UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE FÍSICA E QUÍMICA
DE SÃO CARLOS

DEPARTAMENTO DE FÍSICA E CIÊNCIA DOS MATERIAIS

Projeto de um Microcomputador de 8 bits para aplicações em Pesquisa e Ensino.

Mateus José Martins

Dissertação apresentada ao Instituto de Física e Química de São Carlos, USP, para obtenção do título de Mestre em Física Aplicada

Orientador:

Prof. Dr. Jan Frans Willem Slaets

São Carlos - SP
1990

UNIVERSIDADE DE SÃO PAULO

INSTITUTO DE FÍSICA E QUÍMICA DE SÃO CARLOS

MEMBROS DA COMISSAO JULGADORA DA DISSERTACAO DE MESTRADO DE MATEUS JOSE MARTINS APRESENTADA AO INSTITUTO DE FISICA E QUIMICA DE SAO CARLOS,DA UNIVERSIDADE DE SAO PAULO,EM 18 DE MAIO DE 1990.

COMISSAO JULGADORA:

Prof.Dr.Jan F.W.Slaets

Prof.Dr.Carlos Antonio Ruggiero

Prof.Dr.Antonio Rubens B.de Castro

Cx. Postal, 369 - FONE (0162) 71-1016 - CEP 13.560 - São Carlos - SP - Telex 162374 - FQSC - BR - BRASIL

Dedico,

A minha Esposa,
Virginia
pelo nosso Amor...

...E a meus pais
Manuel e Maria
pelo estímulo e apoio
durante a minha vida

AGRADECIMENTOS

Ao Prof. Dr. Jan Frans Willem Slaets, meu orientador, pela valiosa orientação, estímulo, apoio e principalmente pela amizada e confiança demonstrados no decorrer deste trabalho.

Ao Prof. Dr. Horacio Carlos Panepucci pela confiança transmitida.

Ao amigo Paulo, pelo apoio durante o desenvolvimento desta dissertação.

Aos amigos do Laboratório de Ressonância Magnética Nuclear e Laboratório de Instrumentação Eletrônica: Alberto, Tito, Claudio, Rene, André, Odir, João, Carlos, Lirio, Ivanilda, Marcos, Ailton e secretárias Cris e Lia pela rica amizade e contribuição dada a este trabalho.

Ao FINEP, FIPEC e FBB pelo apoio financeiro.

TABELA DE CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O presente trabalho descreve o desenvolvimento de um microcomputador de 8 bits. O projeto inclui além dos circuitos básicos, lógica adicional para extender a memória contornando o limite normal de endereçamento.

Um disco virtual em RAM e uma interface para "Winchester" foram desenvolvidas para extender a capacidade de armazenamento secundário e a velocidade de execução.

Suporte para o coprocessador AM9511 é fornecido para frequentes cálculos em ponto flutuante.

Rotinas para operações básicas de E/S, manipulação da memória e "Caching" de disco, foram desenvolvidas para suportar o sistema operacional CP/M.

Um monitor residente com montador, desmontador e funções de E/S de alto nível, foi construido para ajudar no desenvolvimento de aplicações dedicadas.

ABSTRACT

The present works describes the development of an 8 bits microcomputer system. The project includes, besides the basic circuity, additional logic for memory extension behind the regular address limit.

A virtual RAM disk and a Winchester interface were developed to extend secondary storage and execution speed.

For floating point intensive calculations support for an AM9511 coprocessor is given.

Routines for basic I/O operations, memory management and disk "Caching" were developed to support the CP/M operating system.

A resident monitor with assembly, disassembly and high level I/O functions was constructed to aid the development of dedicated application.

1 - INTRODUÇÃO

Com o crescente desenvolvimento do conhecimento humano em todas as áreas, a necessidade de ferramentas para cálculos e controle de técnicas envolvidas em cada setor do conhecimento, apresenta extrema importância. Uma dessas ferramentas, que desde o seu aparecimento está revolucionando a humanidade é o *computador.*

Os computadores tornaram-se cada vez mais acessíveis com o aparecimento dos microprocessadores na década de 70, reduzindo a complexidade dos circuitos e permitindo o aparecimento dos microcomputadores.

Hoje em dia quase todos os equipamentos avançados são controlados por microprocessadores, ou dispõem de interfaces para conexão com um microcomputador.

Um dos lugares onde tais equipamentos estão se tornando de uso comum, são os laboratórios de pesquisa. Nestes existem muitas experiências sendo agilizadas, e em algumas destas o microcomputador é de supra importância, como por exemplo para controle dedicado, onde existe a necessidade de monitoração e intervenção frequente com grande rapidez (aplicações em tempo real); ou até mesmo em sistemas para cálculos complexos e longos. Assim com a diminuição do custo os microcomputadores tem suas aplicações cada vez mais amplas, tornando seu uso inevitável, e consequentemente aumentando a necessidade de pessoal qualificado, com conhecimento sobre o funcionamento e interligação do micro a outros equipamentos. Muitas universidades tem cursos para a formação de pessoal, porém estes cursos exigem grandes quantidades de micro-computadores, nos laboratórios de ensino.

Os micros comerciais resolvem em parte esses problemas, mas seu custo é elevado, tornando-se difícil sua aquisição e emprego em muitas aplicações; principalmente se

for utilizado como um controlador em tempo integral de uma experiência, ficando assim "preso", sem possibilidade de utilização em outras aplicações. Além disso, nos laboratórios de ensino, existe a necessidade de micros cujo "hardware" seja resistente e o "software" muito bem documentado.

O microcomputador desenvolvido e descrito nesta dissertação é capaz de ser utilizado em aplicações como um simples controlador, por exemplo: em robos, sistemas de aquisição de dados, controle de experiências, etc; chegando até a formar um completo sistema de processamento, com as ampliações opcionais desenvolvidas. O microcomputador deve ser construido com componentes de fácil obtenção no mercado nacional, e de custo relativamente baixo, para facilitar sua reprodução e manutenção. O mesmo ainda deve permitir sua ampliação e modificação para atender as necessidades de cada usuário e deve ter uma arquitetura simples para ser utilizado em laboratórios de ensino na aprendizagem dos alunos.

Como nenhum computador funciona sem "software", deve-se fornecer concomitantemente com o "hardware" um *monitor*[11], termo utilizado para descrever uma combinação entre um interpretador de comandos e um sistema operacional, que contem uma série de rotinas para controlar o microcomputador, a fim de se executar uma determinada tarefa. Esse monitor deve ser capaz de dar ao usuário todo controle, recursos e facilidades para o manuseio do "hardware", tais como acesso aos registradores da Unidade central de processamento (CPU), à memória e aos dispositivos de entrada e saída; bem como recursos para a confecção do programa dedicado ao controle do experimento. O usuário deve comunicar-se com o monitor de uma maneira simples, devendo este ainda, ter a capacidade de reconhecer vários formatos de dados e se possível possuir uma sintáxe parecida com outros monitores já conhecidos.

Como no "hardware", o monitor deve ser ampliável, assim pode-se acrescentar novas rotinas com facilidade, ou até mesmo instalar um sistema operacional popular, que

contenha vários programas disponíveis no mercado e esteja relativamente bem documentado, como o CP/M[2] ("Control Program for Microcomputers").

O CP/M é um sistema operacional, isto é, um conjunto de programas ou rotinas relacionadas entre si, cujo objetivo é agir como interface entre o usuário ou programa de aplicação e o "hardware", sendo este um dos mais populares sistemas existentes para CPUs de 8 bits. O CP/M propicia um ambiente geral para construção, armazenamento, montagem, compilação e execução de programas em "assembler" ou linguagem de alto nível.

Mesmo utilizando um sistema operacional comercial, o monitor dá suporte para vários formatos e tipos de discos flexíveis, com a possibilidade de utilização até mesmo de discos rígidos "Winchesters" e virtuais. Discos virtuais são memórias, simulando discos normais, nas quais os dados são guardados muito rapidamente, isto é, o usuário tem a impressão de estar utilizando um disco comum cuja velocidade de acesso é muito rápida. Além de vários formatos, os discos flexíveis são equipados com "Caches", isto é, rotinas utilizando algorítimos capazes de armazenar, ler dados antecipadamente e escreve-los somente quando necessário. Tais algorítimos aumentam a velocidade de transferência entre a memória do microcomputador e a unidade de armazenamento.

Para melhorar o sistema e facilitar o usuário, foram desenvolvidos vários utilitários, tais como: AutoRun, Exec, Format, Cls, Dis, Config, Logout, SetSp, Master, Boot, etc., os quais serão vistos posteriormente.

Todas essas implementações foram feitas residentes, sem diminuição do espaço de endereçamento, assim o sistema operacional CP/M pode operar com toda a capacidade disponível, pois a memória utilizada para armazenar os dados, os "buffers", e suas rotinas, são internas ao monitor.

Portanto a presente dissertação, descreve o desenvolvimento e construção desse microcomputador, chamado de micro LIE, e está organizada em 4 capítulos, sendo que:

No primeiro capítulo apresenta a arquitetura típica de um microcomputador, sendo analizados os módulos que o compõem. Depois, é apresentado os motivos da escolha dos principais componentes do presente projeto, tanto na parte do "Hardware" como na do "Software".

No segundo capítulo, apresentam-se ao usuário os módulos básicos que formam o sistema operacional CP/M, utilizado como sistema alternativo. Nele também são mostradas as estruturas que constituem o BIOS, único módulo necessário de ser programado, para a utilização do CP/M.

O terceiro capítulo descreve as características do presente projeto, bem como apresenta todas as partes integrantes deste, tanto do "Hardware", com seus módulos e periféricos opcionais, como do "Software", com seu *monitor* ou sistema operacional e programas aplicativos.

O último capítulo apresenta os resultados e aplicações do projeto, bem como uma avaliação do seu desempenho.

No decorrer desta dissertação alguns termos apesar de terem tradução para o português, apresentam um significado mais expressivo em inglês, sendo assim utilizado.

2. ARQUITETURA

2.1 INTRODUÇÃO

Um microcomputador, como qualquer computador, é constituido de dois componentes principais, o "hardware" e o "software". O "hardware" é o equipamento físico do computador, os circuitos e os dispositivos a ele diretamente relacionados; o "software" é a coleção de programas que controlam o microcomputador.

Este capitulo tenta dar uma visão geral da arquitetura de um microcomputador e o porque da utilização de determinados componentes na sua confecção, bem como da linguagem utilizada para escrever o monitor e as rotinas necessárias à instalação de um sistema operacional.

2.2 VISÃO DA ARQUITETURA

A arquitetura de um computador[TAN] é o projeto de suas principais partes, e como são interligadas.

A Figura 1 representa uma arquitetura típica de um microcomputador.

Os componentes mostrados na Figura 1 são: a unidade central de processamento (UCP), o circuito de temporização, a memória, os circuitos de entrada e saida (E/S), a lógica de controle do duto e o próprio duto ("Bus").

Nos microcomputadores, o microprocessador é a "UCP", algumas vezes também chamada de "microprocessor unit" ("MPU"). O objetivo é decodificar as instruções e usá-las como controle das atividades dentro do sistema, bem como executar todos os cálculos aritiméticos e lógicos.

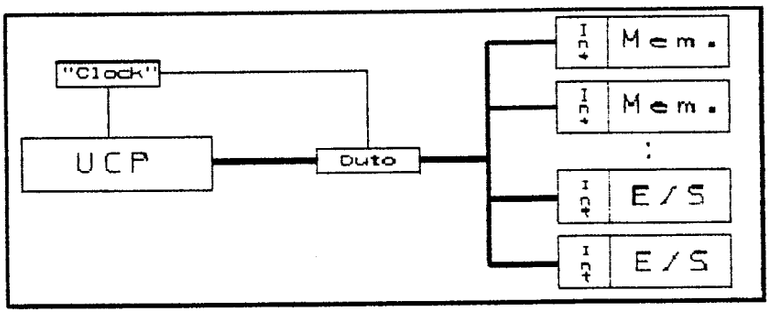


Figura 1 - Arquitetura básica de um microcomputador

2.3 RELÓGIO ("CLOCK")

O circuito de temporização ou "clock", gera os sinais necessários para sincronizar as atividades dentro do microprocessador e da lógica de controle do duto. Esses sinais podem apresentar frequências diferentes ou terem a mesma frequência, porém estarem separados no tempo, isto é, com diferentes fases.

2.4 MEMÓRIA

A memória é usada para armazenar tanto dados, como instruções que são correntemente utilizadas pela UCP.

Normalmente esta é dividida em vários módulos, onde cada um contem alguns milhares de localizações. Cada localização pode conter parte ou o todo de um dado, assim como de uma instrução, e está associada a um indicador chamado endereço da memória ou simplesmente endereço.

A UCP trabalha com sucessíveis entradas de instruções da memória ("fetchs"), executando as tarefas determinadas por estas instruções.

2.5 CIRCUITO DE ENTRADA E SAíDA (E/S)

O circuito de entrada e saída pode consistir de uma variedade de dispositivos para comunicação com o mundo externo e pode armazenar grandes quantidades de informações. Discos "CDs", conversores analógico-digitais (A/D), "mouses", "scanners", mesas digitalizadoras são exemplos de

dispositivos de entrada de dados. Enquanto que impressoras, traçadores gráficos ("plotters"), conversores digital-analógicos (D/A) são dispositivos de saída. Alguns dispositivos tais como terminais de vídeo e papel, têm capacidade de entrada e saída.

Os dispositivos para armazenamento permanente de programas e dados são chamados de unidades de armazenamento em massa. Os tipos mais populares são as fitas "stream" e as unidades de disco.

Embora os discos e fitas possam guardar programas tão bem quanto dados, os programas devem ser transferidos para a memória antes de serem executados.

2.6 O DUTO

O duto ou "Bus" é um conjunto de condutores que conectam a UCP à memória e aos dispositivos de entrada e saída. Esses condutores podem ser constituídos de fios ou trilhas de circuito impresso. Como exatamente, as informações são transmitidas pelo duto é determinada pelas especificações dos sinais.

Normalmente o duto é dividido em três grupos de condutores:

1- As linhas onde circulam os dados

2- As linhas de endereço, que indicam onde as informações são colocadas ou de onde elas devem vir.

3- As linhas de controle, que controlam todas as atividades do duto.

Os sinais do duto devem ser sincronizados, com aqueles fornecidos pelos outros componentes conectados. Os circuitos necessários para conectar o duto a um dispositivo, é chamado de interface. A lógica de controle do duto é a interface para a UCP, e dependendo da complexidade do sistema a lógica de controle do duto pode estar parcialmente

ou totalmente contida na UCP. A interface com a memória consiste a princípio da lógica necessária a decodificação do seu endereço e do circuito que executa a leitura ou escrita.

A interface de entrada e saída pode ser muito simples ou complexa, devendo todas serem capazes de armazenar os dados, receber comandos da UCP e transmitir informações de estado do dispositivo.

Como capacidade extra, as unidades de armazenamento em massa ou interfaces rápidas ("scanners") devem poder comunicar-se com a memória diretamente, acesso direto a memória (DMA), isto requer um habilitador no controle do duto.

A comunicação entre as interfaces de entrada e saída e o duto de dados é feita através de registradores, os quais são chamados de portos de entrada e saída ("I/O Ports").

2.7 ESCOLHA DOS COMPONENTES

Para a elaboração do projeto procurou-se escolher componentes de fácil obtenção no mercado nacional, para tornar o custo relativamente baixo e de fácil reprodução. Outros fatores que devem ser considerados são: a flexibilidade para ampliações, e a confiança. Para a confiança ser alta, a taxa de falhas deve ser a menor possível, como esta é proporcional ao número de componentes no sistema e a complexidade destes, deve-se utilizar o menor número de componentes possíveis.

Embora as UCPs de 16 bits estejam tornando-se cada vez mais utilizadas, seu conjunto de instruções é mais complexo assim como os circuitos necessários para seu funcionamento, ficando mais difícil seu uso em laboratórios de ensino, principalmente para um contato inicial dos alunos. Além disso apresentam um custo maior que as de 8 bits, e como, mesmo hoje em dia, a maior parte dos periféricos são de 8 bits, optou-se pelo uso de uma UCP de 8 bits.

Analizando as UCPs disponíveis no mercado, utilizou-se o Z80 UCP[4] introduzido pela Zilog Inc. em 1976. Sua arquitetura é simples, possuindo uma família de componentes os quais interligam-se com facilidade e apresentam vários recursos já incorporados aos CIs. O Z80 UCP além de ser totalmente compatível com as UCPs 8080 e 8085, apresenta um conjunto maior e mais completo de instruções, facilitando o desenvolvimento do "software".

A família do Z80 inclui:

1- Z80 PIO[5] ("Z80 Paralell I/O controller"), o qual destina-se a servir como interface entre dispositivos periféricos e o Z80 UCP, possuindo dois portos paralelos cada um de 8 bits.

2- Z80 CTC[6] ("Z80 Counter Timer Circuit"), que dispõem de quatro contadores, tanto para contagem de eventos como de tempo, possuindo vários modos de operação e linhas de controle externas.

3- Z80 SIO[7] ("Z80 Serial I/O controler"), é um circuito que possui dois dispositivos que providenciam formatação dos dados para uso em comunicação serial. O circuito é capaz de operar com vários protocolos (IBM Bisync, HDLC, SDLC e outros) e modos, com geração e verificação de CRC ("Cyclic Redundancy Count").

4- Z80 DMA[8] ("Z80 Direct Memory Access"), capaz de transferir dados entre memória e dispositivos diretamente à altas taxas, possuindo quatro modos de operação.

Para diminuir o número de CIs da memória, optou-se pelo uso de memórias dinâmicas ao em vez das estáticas, pois estas apresentam uma maior capacidade de armazenamento por pastilha, além de serem menores em tamanho.

Nas memórias dinâmicas, as células de dados são formadas pelo armazenamento de cargas nas junções de semicondutores, pode-se relacionar esse armazenamento ao de

um capacitor, como esses "capacitores" tendem a se descarregar, é necessário que de tempos em tempos haja um refrescamento ("refresh"), carregando-os novamente. O Z80 já dispõe de parte dos recursos necessários ao "refresh" dessas memórias. Outro problema que esse tipo de memória acarreta é a necessidade de compartilhamento das linhas de endereçamento, porém assim mesmo, a quantidade de CIs necessários é bem menor do que se fossem utilizados memórias estáticas, da mesma forma que o espaço disponível.

Para cada interface, optou-se em utilizar um CI dedicado a esta, diminuindo a complexidade e o número de componentes.

2.8 UMA VISÃO MAIS DETALHADA DO Z80

O Z80 UCP é um "chip" de 40 pinos DIP ("Dual In line Package"), que inclui essencialmente todo o "hardware" e facilidades de "software" do 8080[5] da Intel Inc., com significativas adições. Como no 8080, 16 linhas de endereçamento e 8 linhas de dados bidirecionais, são conectadas diretamente ao Z80. Algumas modificações no projeto reduziram o número de linhas de controle. O Z80 contém apenas um sinal de "clock" ao contrário dos dois para o 8080. Dois pinos a menos são utilizados na alimentação, já que o Z80 precisa apenas de +5 Volts, enquanto o 8080 precisa de +12, +5 e -5 Volts. O Z80 tem duas linhas de interrupção, uma mascarável, isto é, a UCP pode desabilitar, e outra não mascarável, cuja UCP não pode inibir.

O Z80 tem duas vezes mais registradores e instruções que o 8080/8085. Além dos registradores alternativos, dispõem de registradores de índice (IX, IY), que permitem todos os modos de endereçamento, do 8080/8085 e do 6800. O Z80 contém também um novo registrador de 8 bits chamado I, o qual aponta para uma página de memória que deve conter uma tabela, a ser utilizada em requisições de interrupção. Outro registrador novo é o chamado R de 7 bits, usado como contador para o "refresh" das memórias, enquanto a UCP decodifica a instrução a ser executada. O registrador R é

automaticamente incrementado a cada instrução, não tomando qualquer tempo da UCP.

O "software" do Z80 é totalmente compatível a nível de linguagem de máquina, com o 8080 e 8085. Ele dispõem de 158 códigos em linguagem montadora ("Opcodes") e 696 códigos diferentes em liguagem de máquina, sendo que o tamanho das instruções pode variar de 1 a 4 bytes. Enquanto que o 8080 tem poucas instruções com operandos de 16 bits, o Z80 tem praticamente todas as instruções do 8080 para transferência e aritméticas de 8 bits em versões de 16 bits, ele dispõem de instruções para manipulação e teste de bits, tanto em memória como em registrador, bém como instruções para manipulação de blocos de dados, tais como procura e movimentação, com ou sem repetição.

3. O CP/M

3.1 INTRODUÇÃO

O CP/M será abordado neste capítulo, para que sua estrutura seja conhecida e se tenha base para o entendimento do BIOS.

O CP/M ("Control Program for Microcomputers") é um sistema operacional[2] produzido pela companhia americana Digital Research, destinado a ser executado, com algumas alterações, em qualquer microcomputador baseado no 8080, 8085 e Z80, que possua pelo menos 20 Kbytes de memória RAM ("Random Access Memory") e de um até 16 unidades de disco.

O sistema foi desenvolvido em 1973 por Gary Kildall, na época, um consultor de "software" para a INTEL, sendo utilizado até o presente momento nos micros da linha MSX. Por volta de 1975, um número significativo de companhias americanas estavam fazendo microcomputadores. A maioria delas preferiu desenvolver seus próprios sistemas operacionais, que não foram capazes de fazer seus produtos chegarem rapidamente aos consumidores. Ao invés disso, alguns fabricantes ("TARBELL ELECTRONICS, DIGITAL MICROSYSTEMS", etc), contornaram este custoso e demorado desenvolvimento, adotando o CP/M e conseguindo colocar os primeiros sistemas com disco, no mercado.

A chave para tornar o desenvolvimento de "software" financeiramente viável, era desenvolver programas que rodassem em diversos microcomputadores distintos, o CP/M tornou isto possível. Hoje, existe rodando em CP/M uma vasta gama de compiladores e interpretadores para as mais variadas linguagens; editores de texto; processadores de palavras; gerenciadores de bancos de dados e os mais diversos programas de aplicação.

No decorrer do tempo o CP/M sofreu algumas modificações, outros sistemas operacionais foram lançados, como o CDOS[10] da Cromenco Inc., porém estes são totalmente compatíveis com o CP/M apresentando pequenas alterações em relação ao original, assim sendo apenas a versão 2.2 do CP/M será analizada.

3.2 ESTRUTURA INTERNA

O CP/M é normalmente carregado no topo da memória disponível e divide-se logicamente em quatro partes distintas, BIOS, BDOS, CCP e TPA[1,11], como mostrado na Figura 2.

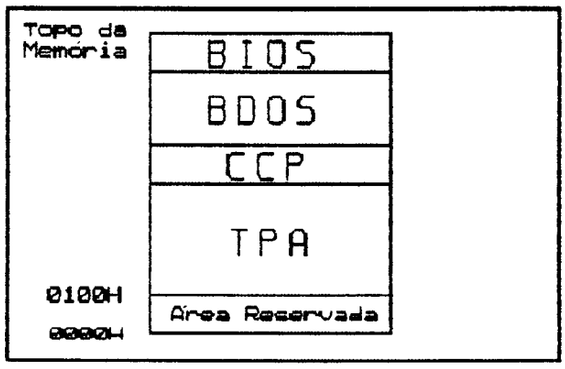


Figura 2 - Disposição do CP/M na memória

O BIOS ("Basic Input/Output System") contém as rotinas básicas de entrada e saída, necessárias ao acesso das unidades de disco, teclado, vídeo, impressora e dispositivos auxiliares[11-13]. Este é o módulo de mais baixo nível, sendo escrito em linguagem "Assembly" e utilizado como interface entre os módulos lógicos do CP/M e o "Hardware".

O acesso ao BIOS é feito através de uma tabela de pontos de entrada ("Jump Table"), localizada no seu início, e que apontam para as subrotinas internas. O BDOS único módulo a chamar o BIOS, localiza as rotinas no BIOS chamando a posição da tabela de "jumps" correspondentes à rotina desejada.

O CP/M pode ser adaptado a qualquer "hardware" particular pela alteração deste módulo. O BIOS, é normalmente escrito pelo fabricante do equipamento e posto no lugar do BIOS original, passando a chamar-se CBIOS ("Custom BIOS"). O CP/M "standard" é fornecido com um BIOS para o módulo de desenvolvimento da INTEL (MDS 800).

Além das primitivas de entrada e saída, o BIOS contém o bloco de parâmetros do disco (DPH/DPB) que define as características de cada disco. Este bloco contém, entre outras, informações sobre: número de setores por trilha, tamanho do bloco em setores, tamanho do disco em blocos, número de entradas no diretório, número de trilhas reservadas para o sistema, etc.

O BIOS contém ainda, duas rotinas de inicialização (BOOT) e recarga (WBOOT), utilizadas quando o sistema é ligado ou reinicializado por programas.

O **BDOS ("Basic Disk Operating System")**, o núcleo propriamente dito do CP/M, é responsável pelo gerenciamento das operações de disco, console e impressora[11,14,15]. O BDOS controla até dezesseis unidades de disco, possibilitando a manutenção de arquivos sequenciais ou randômicos de até 8 Mbytes cada. É ele quem cuida da alocação de espaço em disco; criação, abertura e fechamento de arquivos; escrita e leitura de arquivos; pesquisa de arquivos no diretório; seleção de discos; troca o nome e remove arquivos, etc. Para o BDOS, os arquivos são mantidos em registros de tamanho fixo de 128 bytes.

O BDOS executa 38 funções que são chamadas através de um "CALL" para o ponto de entrada no endereço 0005H. Ao ser executado esse "CALL", o registrador **C** da UCP deve conter o número da função desejada, e o par de registradores **DE**, o parâmetro ou endereço do parâmetro, conforme a função a ser executada.

Entre outras, o BDOS executa as seguintes funções: seleciona uma unidade de disco; cria um arquivo; abre um arquivo; fecha um arquivo; busca arquivos no diretório;

apaga um arquivo; muda o nome de um arquivo; lê ou escreve em um arquivo sequencialmente; lê ou escreve em um arquivo randomicamente; modifica os atributos de um arquivo.

O **CCP ("Console Command Processor")** proporciona a interface entre o usuário e o restante do sistema operacional. Ele é o responsável pela leitura, interpretação e execução dos comandos teclados pelo usuário.

O CCP pode acessar a todos os discos logicamente ligados ao sistema, ele utiliza as letras de A até P, para indicar qual a unidade de disco que está sendo utilizada. Após a inicialização do sistema, o CCP assume a unidade "A" como ativa.

O CCP possui 7 comandos residentes, isto é, são comandos executados pelo próprio CCP. Se o comando emitido pelo usuário não coincidir com estes 7, ele procura no diretório do disco um programa do tipo ".COM", encontrando-o, carrega-o no início da área de programas transientes (TPA), no endereço 0100H, e passa a executá-lo. Os programas assim carregados são chamados de comandos transientes. Após a execução, o programa transiente devolve o controle ao CCP que passa a aguardar novo comando.

A **TPA ("Transient Program Area")**, que inicia no endereço 0100H, conforme mencionado. é a área onde são carregados para a execução, tanto os comandos transientes do sistema como os programas de aplicação.

3.3 BIOS

Além das rotinas de baixo nível de cada dispositivo, e uma tabela de "JUMPs" utilizadas para o BDOS chamar tais rotinas, no BIOS existe também, tabelas que descrevem as características particulares do subsistema de disco, necessárias ao CP/M. A seguir serão descritos os elementos dessas tabelas, para entendimento posterior das implementações efetuadas.

A cada disco é associada uma área de 16 bytes, chamada de "**Disk Parameter Header**" (**DPH**), que tanto provê

informações sobre o disco, como indica áreas de serviço para certas operações do BDOS.

O formato da DPH, para cada disco é dado a seguir:

XLT	0000	0000	0000	DIRBUF	DPB	CSV	ALV
2	2	2	2	2	2	2	2

(em bytes)

onde cada elemento é um valor de 16 bits. O significado de cada um dos campos do DPH está descrito abaixo:

XLT Endereço do vetor de conversão de setores lógicos para físicos. Os discos que utilizam o mesmo fator de conversão partilham a mesma tabela. Se não houver conversão de setor, isto é, o número do setor físico é o mesmo do setor lógico, seu valor deve ser zero (0000H).

0000 Estes 6 bytes são utilizados pelo BDOS para salvar valores intermediários, durante o acesso ao diretório.

DIRBUF Endereço de uma área de 128 bytes para operações com o diretório, dentro do BDOS. Todos os DPHs endereçam a mesma área.

DPB Endereço do bloco de parâmetros de disco "**Disk Parameter Block**". Este bloco, que será detalhado mais adiante, descreve as características de alocação de espaço do disco. Geralmente todos os DPHs endereçam o mesmo DPB, já que na maioria dos casos, todos os drives de um sistema possuem as mesmas características lógicas.

CSV Endereço de uma área de "**Check Sum**" do diretório. Na primeira vez que o BDOS seleciona um drive, ele lê um determinado número de setores do diretório, definido no DPB, calcula o "check sum" de cada um e armazena o resultado nesta área, 1 byte por setor. A partir deste instante, cada setor a ser escrito no diretório pelo BDOS, tem seu "check

sum" comparado com a área CSV, em caso de diferença a unidade de disco é marcada como somente de leitura "Read Only". Isto evita que, inadvertidamente, o operador troque o disco de um drive durante a escrita em um arquivo.

ALV Endereço de uma área usada pelo BDOS para guardar a informação sobre a alocação do espaço em disco, isto é, uma área destinada à montagem da tabela de alocação de blocos. Esta tabela é utilizada pelo BDOS para indicar quais blocos do disco estão disponíveis. A cada bit desta tabela corresponde um bloco do disco. Se o bloco estiver utilizado, o bit correspondente estará acionado, igual a 1, caso esteja livre o bit estará desligado, igual a 0. Ao selecionar um disco, o BDOS lê todas as entradas do diretório do disco nele existentes, e cria a tabela de alocação, ligando os bits correspondentes aos blocos utilizados. A partir desse momento, toda vez que o BDOS precisar de um bloco, ele procura sequencialmente na tabela até achar um bloco livre, então este é marcado como usado. De forma semelhante quando o BDOS libera um bloco, ele calcula a partir do número do bloco, a sua posição na tabela de alocação, e desliga o bit correspondente.

Outra tabela existente, já citada no DPH, é a "Disk Parameter Block" (DPB). Uma particular DPB pode ser endereçada por uma ou mais DPHs.

O formato do DPB é o seguinte:

SPT	BSH	BLM	EXM	DSM	DRM	AL0	AL1	CKS	OFF
2	1	1	1	2	2	1	1	2	2

(em bytes)

onde cada elemento é 1 byte, 8 bits, ou uma palavra, 16 bits, cujo significado é o seguinte:

SPT é o número total de setores, de 128 bytes, por trilha.

BSH é o fator de deslocamento para alocação de blocos. Este parâmetro é utilizado pelo BDOS no cálculo do bloco relativo a um dado registro de arquivo. O BSH é igual ao logarítimo na base dois do número de setores por bloco, BHS = $\log_2$ (Nº de setores por bloco).

BLM é o número de setores por bloco menos um. Este parâmetro é utilizado pelo BDOS no cálculo da trilha e setores físicos correspondentes a um dado registro de arquivo. A Tabela I mostra os valores para BSH e BLM em função dos tamanhos de blocos possíveis.

Tabela I - Valores possíveis para BSH e BLM

TAMANHO DO BLOCO (bytes)	BSH	BLM
1024	3	7
2048	4	15
4096	5	31
8192	6	63
16384	7	127

EXM é a máscara para "extent". Esta máscara é utilizada pelo BDOS durante as comparações das entradas no diretório, para saber qual extenção lógica corresponde uma determinada entrada. A Tabela II mostra os valores possíveis para o campo EXM em função do tamanho do bloco.

Tabela II - Valores possíveis para EXM

TAMANHO DO BLOCO (bytes)	DSM < 256	DSM ≥ 256
1024	0	-
2048	1	0
4096	3	1
8192	7	3
16384	15	7

DSM é o número máximo de blocos que o disco pode conter menos um. Este valor não inclui as trilhas de sistema.

DRM é o número máximo de entradas no diretório menos um, isto é, determina o número máximo de arquivos que um disco pode conter.

AL0, AL1 determina os "clusters" reservados para o diretório, é utilizado para inicializar o mapa de bits.

CKS é o número de setores de diretório que têm seu "Check Sum" calculado. Normalmente, no caso de discos flexíveis todos os setores do diretório devem ter seu "Check Sum" calculado.

Em capítulo a seguir será mostrado todas as rotinas que formam o BIOS.

4. IMPLEMENTAÇÕES

4.1 INTRODUÇÃO

Em capítulos anteriores foram apresentados os motivos da adoção dos elementos principais que compõem este trabalho. Neste capítulo será apresentado o trabalho desenvolvido, analizando-se cada parte separadamente.

O capítulo está dividido em dois tópicos principais o "hardware" e o "software", e estes por sua vez apresentam-se subdivididos nos módulos básicos e em opções que o usuário poderá utilizar, para ampliar a capacidade do micro.

4.2 "HARDWARE" BÁSICO

O sistema desenvolvido inclui vários recursos, com o objetivo de aumentar a versatilidade e potencial de aplicação do micro.

Como existe a necessidade de utilizar uma EPROM, destinada ao código de inicialização e de controle, utilizou-se a implementação de Bancos de memória, a fim de não perder esse espaço para uso em coleta de dados e para o CP/M. Bancos de memória são memórias que ocupam o mesmo espaço de endereçamento, sendo que, para evitar o conflito entre as mesmas, utiliza-se um circuito destinado a colocar em operação apenas um determinado banco por vez.

No projeto utilizou-se dois bancos, sendo o primeiro, chamado Banco 0, destinado a EPROM e uma RAM estática de pequena capacidade (2 Kbytes), utilizada para "Stack", variáveis internas e "buffers" de dados. O segundo banco (Banco 1) utiliza todo espaço de endereçamento (64 Kbytes) sendo contituido por memórias RAM dinâmicas.

Para facilitar a transferência de dados entre bancos, utiliza-se um circuito que além de comutar entre os

bancos, permite formar um espaço de endereçamento composto por partes dos dois bancos. Como recurso extra permite ao usuário expandir o número de bancos, mapeando-os no lugar do banco 1.

Para futuras ampliações ou utilização com equipamentos, o micro foi dotado de amplificadores de sinal ("buffers") em todas as linhas do conector de expansão, isto aumenta o "fan-out" e protege os circuitos internos de qualquer problema com relação aos circuitos externos.

Outro recurso extra instalado no micro, foi a utilização de uma pseudo DMA ao invés de uma DMA, a fim de baixar os custos e manter a possibilidade de utilizar dispositivos de acesso rápido.

A seguir encontra-se uma descrição do "Hardware" do micro LIE, sendo que para facilitar a sua compreensão será descrito as funções básicas de cada bloco.

4.2.1 Diagrama em blocos

Na Figura 3 pode-se ver o Z80-UCP e todos os módulos que compõem o micro desenvolvido, bem como suas inter-ligações.

Os módulos apresentados são: o gerador de temporização ("CLOCK"), o circuito de inicialização ("RESET"), o circuito de seleção de bancos de memória (SELEÇãO BANCO), o banco de memória 0 (EPROM/RAM ESTáTICA), o banco 1 (RAM DINâMICA), o controle do duto ("BUS"), as interfaces e os dispositivos de entrada e saída ("E/S") e por último a fonte de alimentação, com as tensões necessárias ao restante dos módulos.

Como pode ser visto, todas as linhas de controle do Z80-UCP são ativas em baixo, isto é, executam sua função quando o nível nas mesmas é "0". As linhas com números entre colchetes, representam dutos com várias linhas, por exemplo:

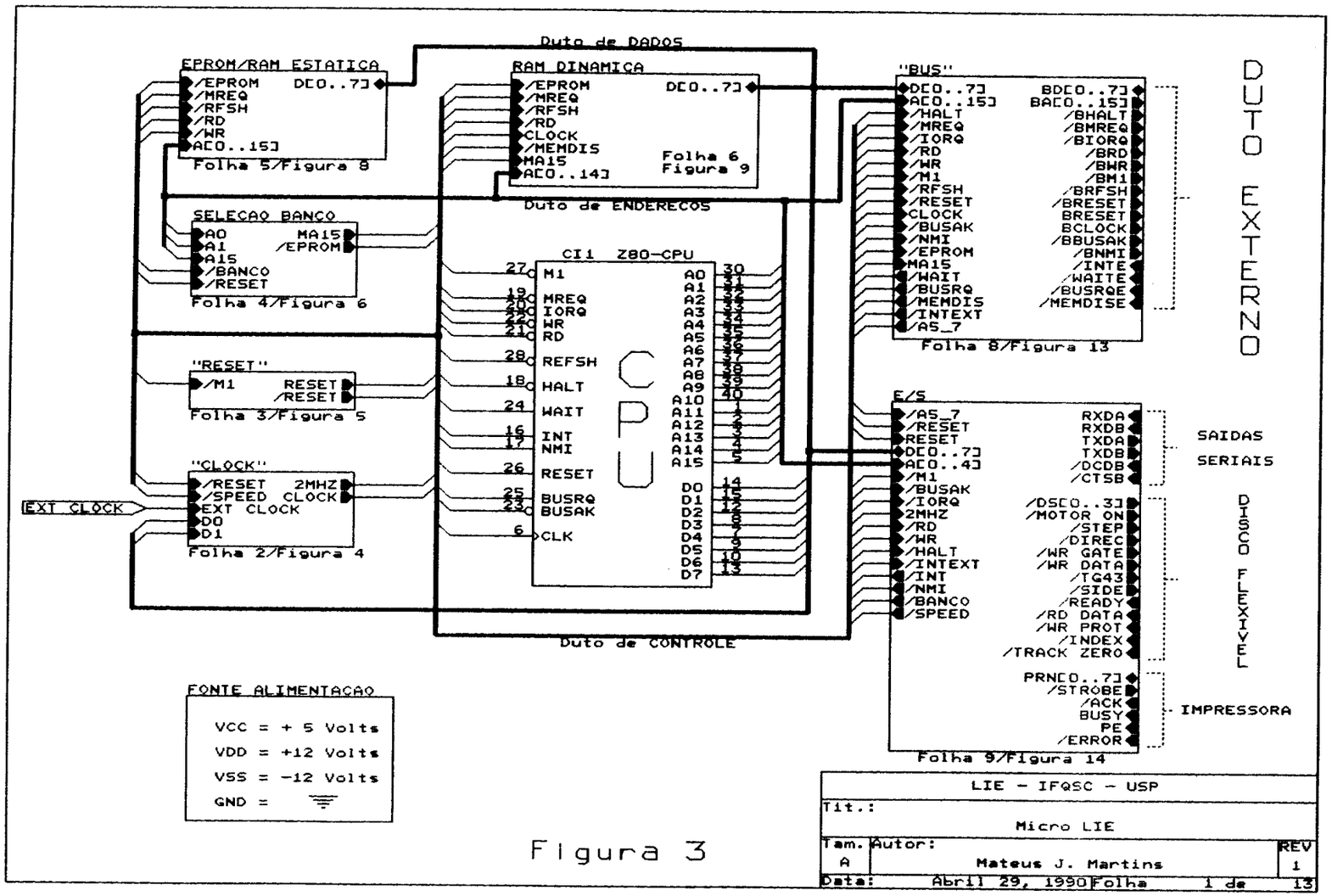

Duto de DADOS
EPROM/RAM ESTATICA
/EPROM
/MREQ
/RFSH
/RD
/WR
AC0..15]
DC0..7]
Folha 5/Figura 8
RAM DINAMICA
/EPROM
/MREQ
/RFSH
/RD
CLOCK
/MEMDIS
MA15
AC0..14]
DC0..7]
Folha 6
Figura 9
Duto de ENDERECOS
"BUS"
DC0..7]
AC0..15]
/HALT
/MREQ
/IORQ
/RD
/WR
/M1
/RFSH
/RESET
CLOCK
/BUSAK
/NMI
/EPROM
MA15
/WAIT
/BUSRQ
/MEMDIS
/INTEXT
/AS_7
BDC0..7]
BAC0..15]
/BHALT
/BMREQ
/BIORQ
/BRD
/BWR
/BM1
/BRFSH
/BRESET
BRESET
BCLOCK
/BBUSAK
/BNMI
/INTE
/WAITE
/BUSRQE
/MEMDISE
Folha 8/Figura 13
DUTO EXTERNO
SELECAO BANCO
A0
A1
A15
/BANCO
/RESET
MA15
/EPROM
Folha 4/Figura 6
CI1 Z80-CPU
M1
MREQ
IORQ
WR
RD
REFSH
HALT
WAIT
INT
NMI
RESET
BUSRQ
BUSAK
CLK
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
D0
D1
D2
D3
D4
D5
D6
D7
CPU
"RESET"
/M1
RESET
/RESET
Folha 3/Figura 5
E/S
/AS_7
/RESET
RESET
DC0..7]
AC0..4]
/M1
/BUSAK
/IORQ
2MHZ
/RD
/WR
/HALT
/INTEXT
/INT
/NMI
/BANCO
/SPEED
RXDA
RXDB
TXDA
TXDB
/DCDB
/CTSB
SAIDAS SERIAIS
/DSC0..3]
/MOTOR ON
/STEP
/DIREC
/WR GATE
/WR DATA
/TG43
/SIDE
/READY
/RD DATA
/WR PROT
/INDEX
/TRACK ZERO
DISCO FLEXIVEL
PRNC0..7]
/STROBE
/ACK
BUSY
PE
/ERROR
IMPRESSORA
Folha 9/Figura 14
"CLOCK"
/RESET
/SPEED
EXT CLOCK
D0
D1
2MHZ
CLOCK
Folha 2/Figura 4
EXT CLOCK
FONTE ALIMENTACAO
VCC = + 5 Volts
VDD = +12 Volts
VSS = -12 Volts
GND =
LIE - IFQSC - USP
Tit.:
Micro LIE
Tam.
A
Autor:
Mateus J. Martins
REV
1
Data:
Abril 29, 1990
Folha
1 de
13

Figura 3

D[0..7], isto significa que existem conjuntamente as linhas D0, D1, D2, D3, D4, D5, D6 e D7.

4.2.2 "Clock"

O circuito de "clock", mostrado na Figura 4 utiliza apenas TTLs[16], é capaz de fornecer várias frequências para a UCP e periféricos. As frequências disponíveis são 1, 2 e 4Mhz, além de uma externa, que o usuário pode utilizar para depuração ou outras finalidades. A comutação das frequências é feita por "software", sendo a mesma síncrona, isto é, o circuito evita a mudança no meio de um pulso de "clock", o que poderia causar problemas com a UCP e a memória, devido ao extreitamento. Após a inicialização, pulso de "reset", o circuito assume a frequência de 2Mhz como a de "clock".

Para modificar a frequência basta efetuar um "OUT 00H", isto é, uma operação de escrita no espaço de E/S no endereço 00H, com o seguinte valor no registrador **A:** 0 para 2Mhz, 1 para 1Mhz, 2 para 4Mhz e 3 para a frequência externa.

4.2.3 "Reset"

O circuito de "reset" destina-se a inicializar a UCP, colocando-a em um estado conhecido. A inicialização inclui:

a) Forçar o registrador de programa (PC) para zero;
b) Desabilita as interrupções;
c) Forçar o registrador I para zero;
d) Forçar o registrador R para zero;
e) Estabelecer o modo de interrupção 0.

A Figura 5, mostra o esquema do circuito de "Reset". O circuito é capaz de fornecer o sinal de "RESET" e seu complemento "/RESET", em duas situações distintas. A primeira, logo após a fase de aplicação da alimentação ("Power Up"), independente das condições do sistema, através do conjunto C3 e R4. A largura do pulso é de aproximadamente 100 milisegundos, o suficiente para que todo o sistema se

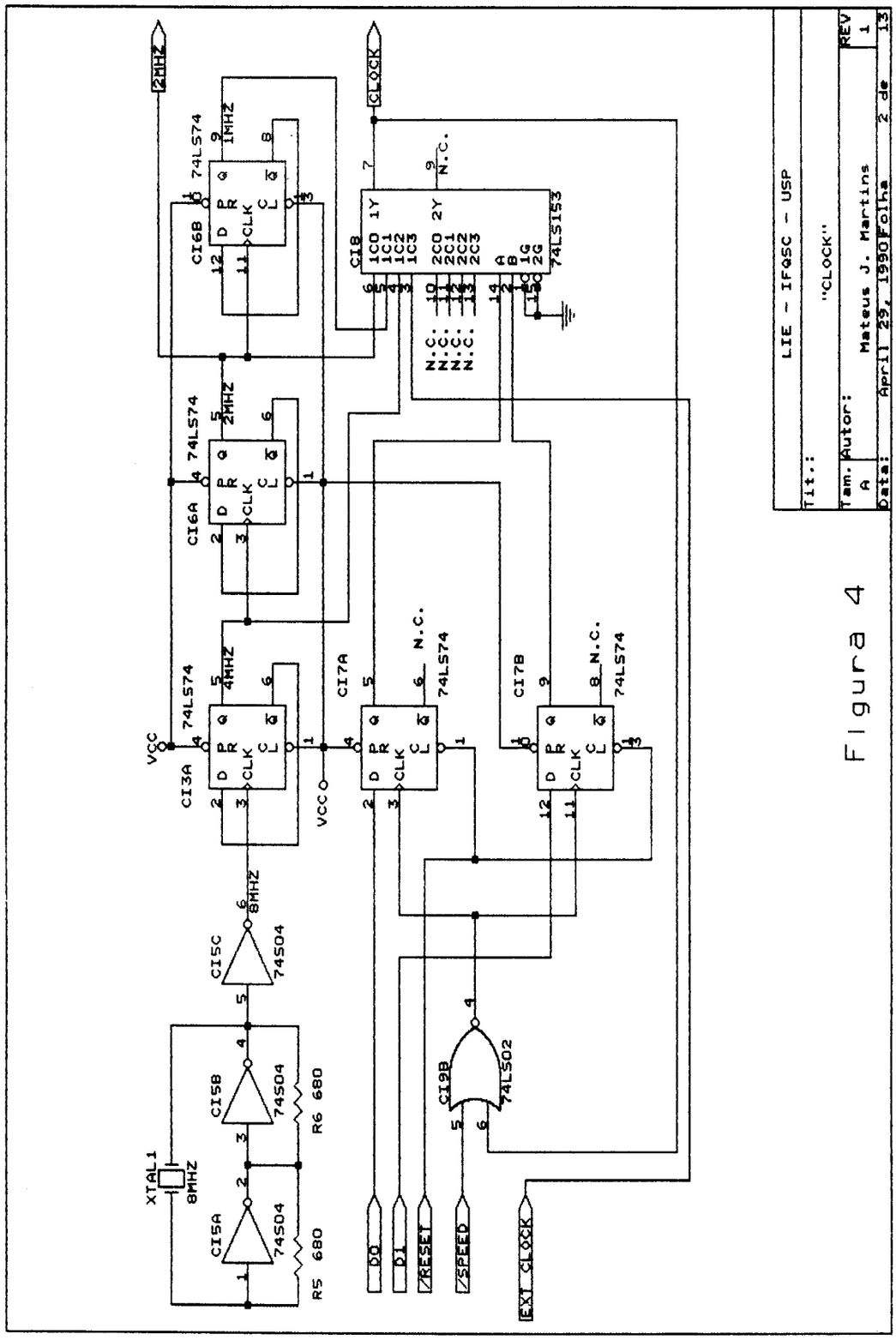

2MHZ
CLOCK
1MHZ
2MHZ
4MHZ
8MHZ
74LS74
CI6B
CI6A
CI3A
CI7A
CI7B
N.C.
CI8
74LS153
1C0 1Y
1C1
1C2
1C3
2C0 2Y
2C1
2C2
2C3
A
B
1G
2G
CI5C
CI5B
CI5A
74S04
CI2B
74LS02
XTAL1
8MHZ
R6 680
R5 680
VCC
D0
D1
/RESET
/SPEED
EXT CLOCK
LIE - IFQSC - USP
Tit.: "CLOCK"
Tam. A
Autor: Mateus J. Martins
Data: April 29, 1990
Folha 2 de 13
REV 1

Figura 4

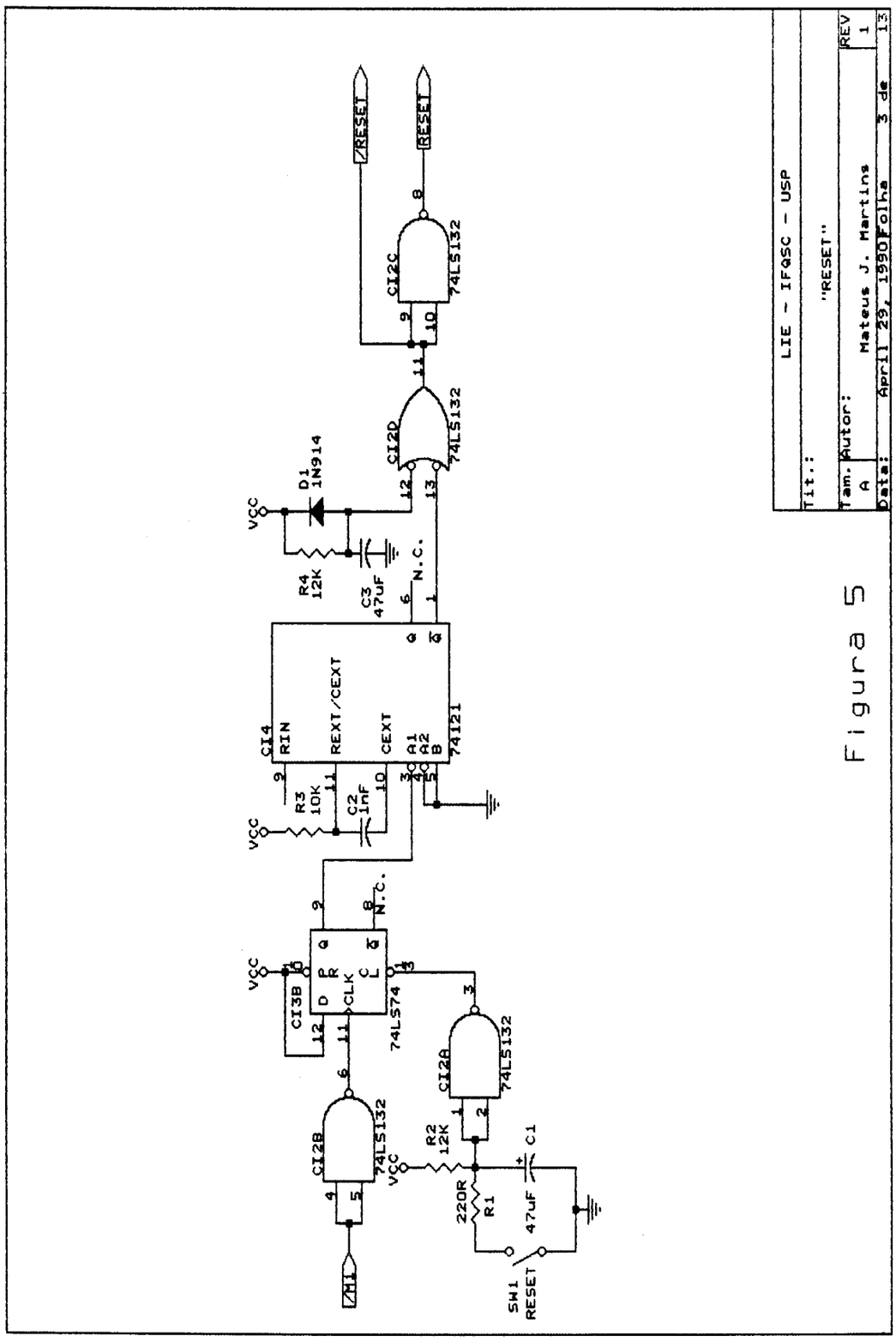

LIE - IFQSC - USP
Tit.:
"RESET"
Tam. Autor:
A
Mateus J. Martins
REV
1
Data:
April 29, 1990 Folha
3 de
13
/RESET
RESET
CI2C
74LS132
CI2D
74LS132
D1
1N914
VCC
R4
12K
C3
47uF
N.C.
CI4
RIN
REXT/CEXT
CEXT
A1
A2
B
74121
R3
10K
C2
1nF
CI3B
D
CLK
74LS74
CI2B
74LS132
CI2A
74LS132
R2
12K
C1
220R
R1
47uF
SW1
RESET
/M1

Figura 5

estabilize. O Diodo D1 força um novo "reset", caso por qualquer motivo, a tensão de alimentação (+5 Volts) diminua, mesmo momentaneamente. A segunda, é quando o usuário pressiona o botão de inicialização manual, porém o sinal só é fornecido a UCP durante o estado de máquina M1, "fetch". Isto ocorre, por que o Z80-UCP apresenta a característica de que se o sinal de "RESET" for aplicado durante o ciclo de "clock" T3, de um ciclo de máquina M1, o sinal /MREQ ficará em um estado indeterminado aproximadamente entre 1 a 10 ciclos de clock. Como no sistema existem memórias dinâmicas, isto causará uma quebra, isto é, um acesso mais curto as memórias podendo destruir os dados dentro delas. Além disso, o circuito evita que o usuário mantenha a UCP em estado de "RESET" durante muito tempo, pois neste estado o refrescamento das memórias é suspenso, o que poderá causar a perda dos dados.

4.2.4 Seleção de banco

Para contornar a limitação do espaço de endereçamento do Z80 UCP, de 64 Kbytes, projetou-se o circuito destinado a selecionar os bancos de memória, cujo esquema pode ser visto na Figura 6.

Cada banco pode conter qualquer valor de memória até o limite de 64 Kbytes, menos o banco 0, como será visto posteriomente. A seleção dos bancos é feita por "software", através de "OUTs" nos portos de E/S entre 04H a 07H, independentemente do valor de qualquer registrador, isto para facilitar a mudança dos bancos. Durante o "RESET" o circuito seleciona a EPROM e a RAM estática (banco 0), como memória inicial do micro.

Embora o banco 0 tenha:
- 16 Kbytes de EPROM divididos em 8 Kbytes de monitor ou EDOS (como será visto no capítulo de "Software"), e 8 Kbytes destinados ao programa do usuário ou para o CP/M dependendo do sistema utilizado;
- 2 Kbytes de RAM estática;

esse banco pode vir a ter até 32 Kbytes distribuidos entre EPROM e RAM estática. Esta limitação se deve por que o

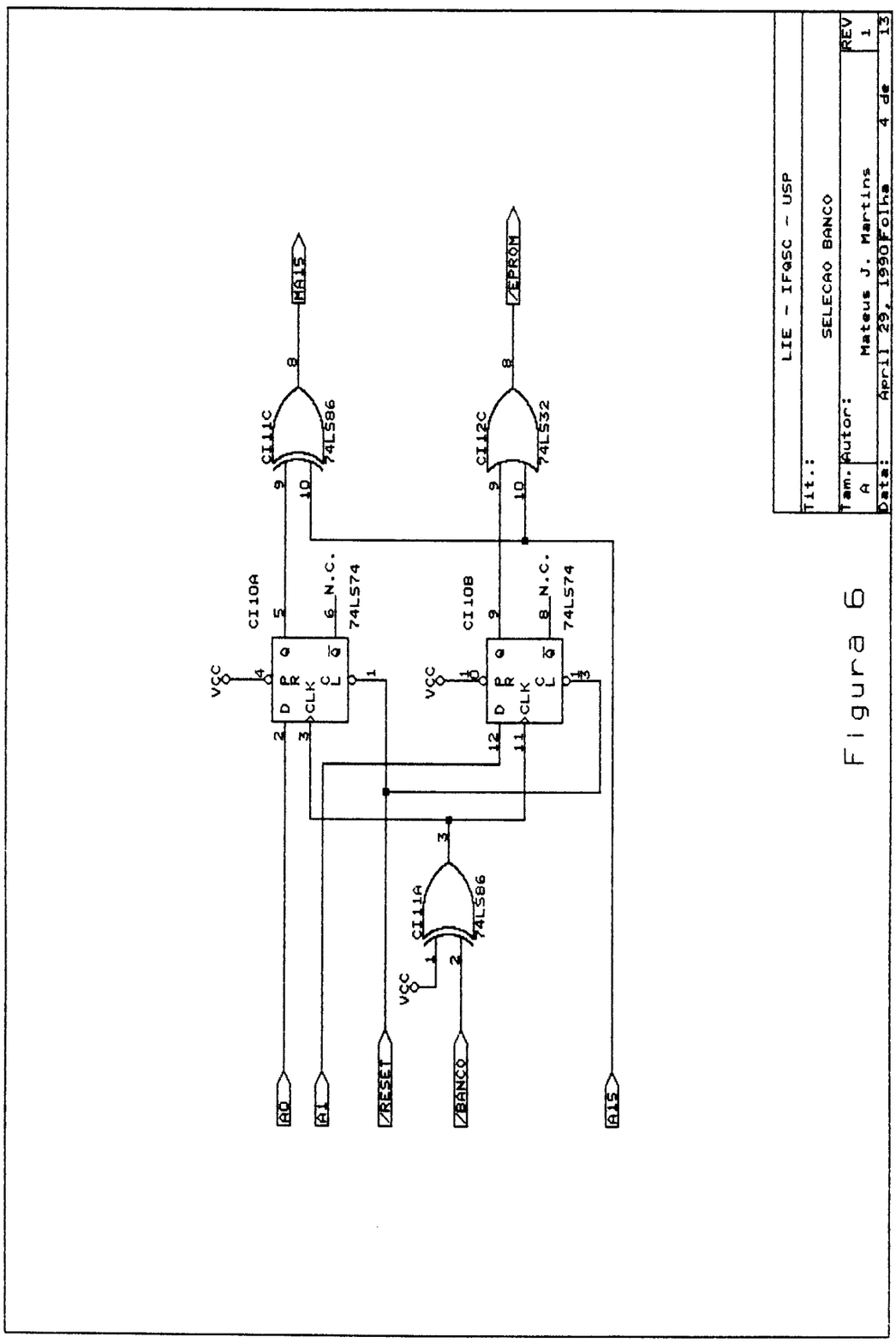

LIE - IFQSC - USP
Tit.:
SELECAO BANCO
Tam. Autor:
A
Mateus J. Martins
REV
1
Data:
April 29, 1990 Folha
4 de
13
MA15
/EPROM
CI11C
74LS86
CI12C
74LS32
CI10A
CI10B
74LS74
N.C.
VCC
D
CLK
PR
CL
Q
CI11A
74LS86
A0
A1
/RESET
/BANCO
A15

Figura 6

espaço de endereçamento acima dos 32 Kbytes (8000H a FFFFH), é utilizado como uma "janela" na qual pode-se "ver" qualquer região do banco 1, ou dos bancos extras instalados pelo usuário. Como essa janela é de 32 Kbytes e os bancos restantes são de 64 Kbytes, o circuito de seleção de bancos permite a inversão das duas metades, podendo assim acessar a região de interesse.

O circuito possui 4 estados possíveis, mostrados na Figura 7. O primeiro selecionado através do "RESET" ou um "OUT" no porto 04H, coloca o banco 0 no espaço de endereçamento inferior da UCP (0000H a 7FFFH) e os 32 Kbytes superiores do banco 1 no espaço de endereçamento superior da UCP (8000H a FFFFH). O segundo estado é selecionado através de um "OUT" no porto 05H, colocando o banco 0 no espaço entre 0000H a 7FFFH e os 32 Kbytes inferiores do banco 1 entre 8000H a FFFFH. O terceiro é selecionado por um "OUT" no porto 06H, colocando o banco 1 inteiramente disponível para a UCP, isto é, os 64 Kbytes de RAM dinâmica no espaço entre 0000H a FFFFH da UCP. O último estado selecionado por um "OUT" no porto 07H, coloca os 32 Kbytes superiores do banco 1 no espaço entre 0000 a 7FFFH e os 32 Kbytes inferiores no espaço entre 8000 a FFFFH da UCP.

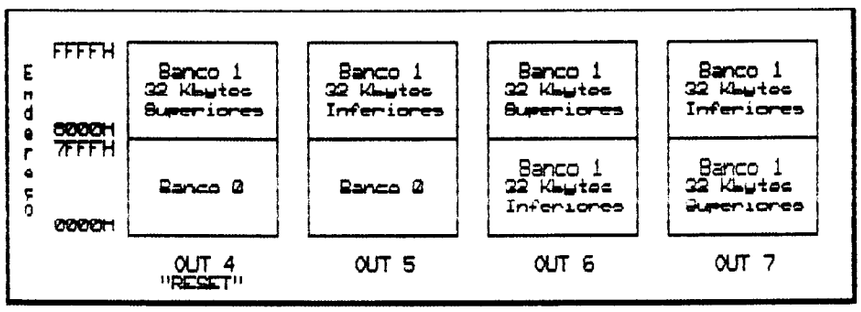


Figura 7 - Seleção dos bancos.

Com o circuito de seleção de bancos o Z80-UCP tem acesso a qualquer região de memória de qualquer banco. O micro permite que mais do que 2 bancos sejam instalados, para isto, o usuário deve acrescentar um circuito para selecionar os bancos extras, utilizando a linha /MEMDIS ("Memory Disable"), que desabilita o banco de RAM dinâmica

(banco 1), dessa maneira a UCP "verá" os bancos extras como sendo o banco 1, e terá tanta memória quanto se deseje.

4.2.5 Eprom/Ram estática

O esquema do banco 0 pode ser visto na Figura 8, ele é composto por um decodificador de endereços, duas EPROMs 2764[16] e uma RAM estática. Quando o micro é utilizado no controle de sistemas, a primeira EPROM é utilizada para o monitor e a segunda para o programa do usuário, sendo que a RAM estática será utilizada para o "Stack" do microprocessador e variáveis internas do monitor, sobrando ainda um espaço para variáveis do programa do usuário. Se for instalado o sistema operacional CP/M, a primeira EPROM conterá o EDOS, parte do CBIOS, e a segunda poderá conter o CP/M com o CBIOS desenvolvido, sendo utilizada a RAM estática para variáveis e "buffers" do EDOS. Caso a EPROM com o CP/M não esteja instalada o EDOS lerá o CP/M da unidade de disco lógico A. Assim o usuário poderá aumentar o CBIOS sem a necessidade de gravar uma nova EPROM.

O monitor ocupa os endereços entre 0000H a 1FFFH, a EPROM do usuário entre 2000H a 3FFFH e a RAM estática entre 4000H a 5FFFH, sendo esta última não decodificada absolutamente, já que a RAM é de 2 Kbytes (4000H a 47FFH).

4.2.6 Ram dinâmica

O esquema do banco 1 está dividido em 2 figuras. A Figura 9 mostra todo o controle e geração de sinais para as RAMs dinâmicas, enquanto a Figura 10 mostra o banco de 64 Kbytes propriamente dito, composto por 8 RAMs de 64 Kbits (4164)[16].

As memórias dinâmicas são formadas por uma matriz de 128 conjuntos de 512 células cada. Quando a primeira parte do endereço, conhecida como endereçamento da linha ("row address"), é recebida pela memória através do sinal "row address strobe" (/RAS), o bit mais alto (MSB) é armazenado e os outros 7 são utilizados para decidir qual dos 128 conjuntos deve ser conectado ao duto interno da memória.

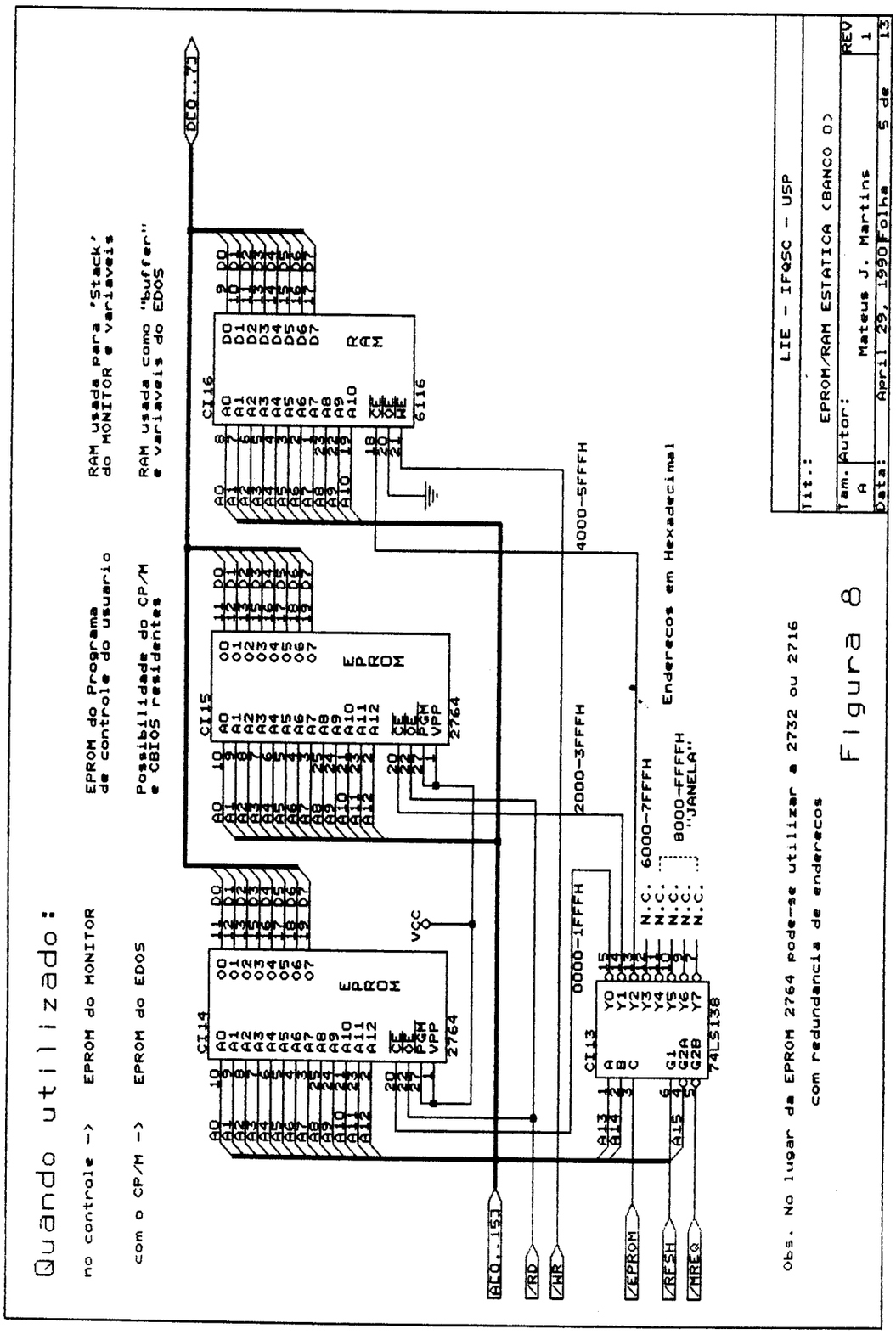
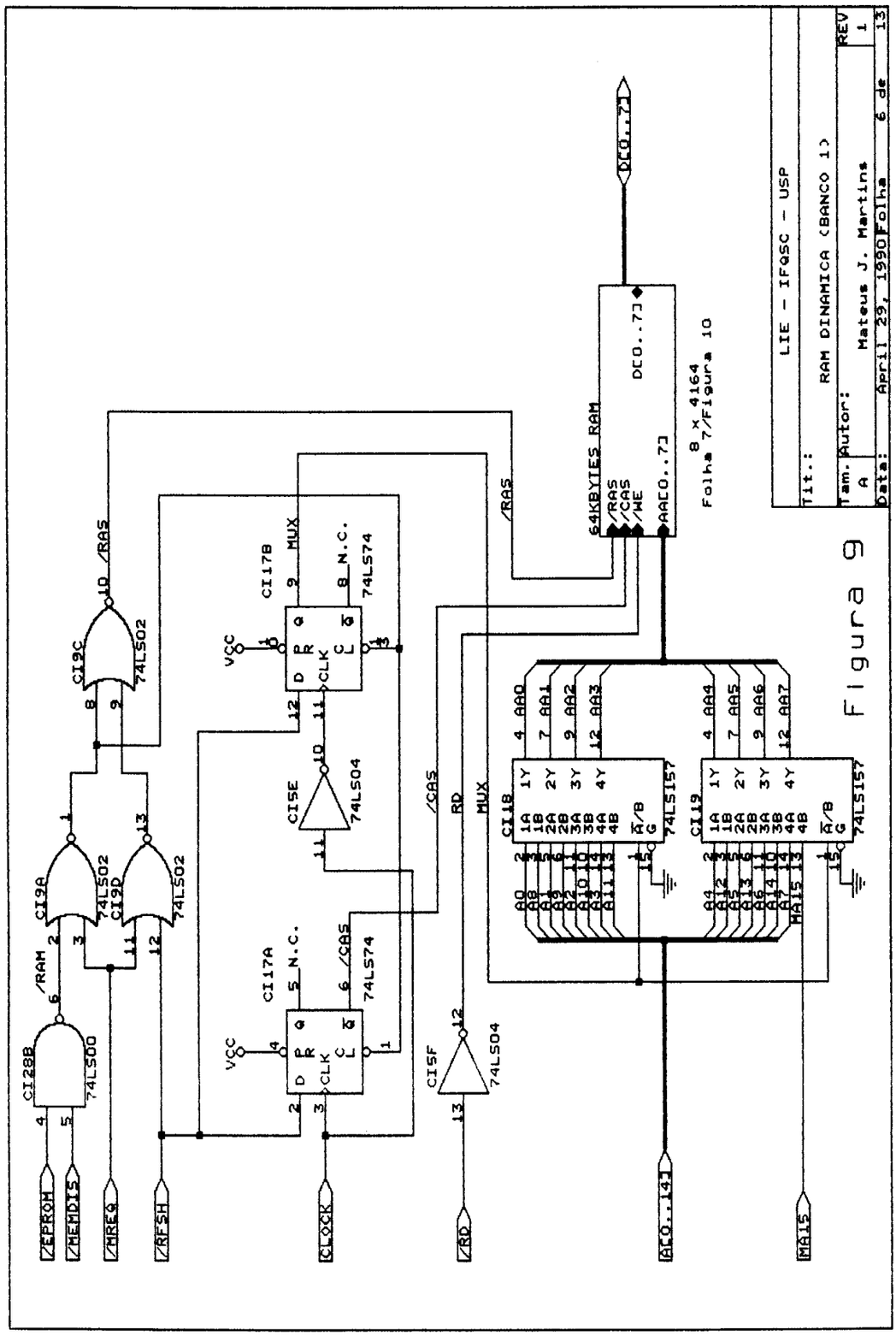

Quando utilizado:
no controle -> EPROM do MONITOR
com o CP/M -> EPROM do EDOS
EPROM do Programa de controle do usuario
Possibilidade do CP/M e CBIOS residentes
RAM usada para 'Stack' do MONITOR e variaveis
RAM usada como "buffer" e variaveis do EDOS
CI14
CI15
CI16
CI13
EPROM
2764
6116
RAM
74LS138
4000-5FFFH
2000-3FFFH
0000-1FFFH
6000-7FFFH
8000-FFFFH
"JANELA"
N.C.
Enderecos em Hexadecimal
DCO..7
DCO..15
/RD
/WR
/EPROM
/RFSH
/MREQ
LIE - IFQSC - USP
Tit.: EPROM/RAM ESTATICA (BANCO 0)
Tam. A
Autor: Mateus J. Martins
Data: April 29, 1990 Folha 5 de 13
REV 1
Obs. No lugar da EPROM 2764 pode-se utilizar a 2732 ou 2716 com redundancia de enderecos

Figura 8


LIE - IFQSC - USP
Tit.: RAM DINAMICA (BANCO 1)
Tam. A
Autor: Mateus J. Martins
Data: April 29, 1990 Folha 6 de 13
REV 1
64KBYTES RAM
8 x 4164
Folha 7/Figura 10
DCO..7]
AACO..7]
/RAS
/CAS
/WE
CI9A 74LS02
CI9B 74LS02
CI9C 74LS02
CI9D 74LS02
CI28B 74LS00
CI17A 74LS74
CI17B 74LS74
CI5E 74LS04
CI5F 74LS04
CI18 74LS157
CI19 74LS157
MUX
/RAM
/ZEPROM
/MEMDIS
/MREQ
/RFSH
CLOCK
/RD
AACO..14]
MAIS
Figura 9

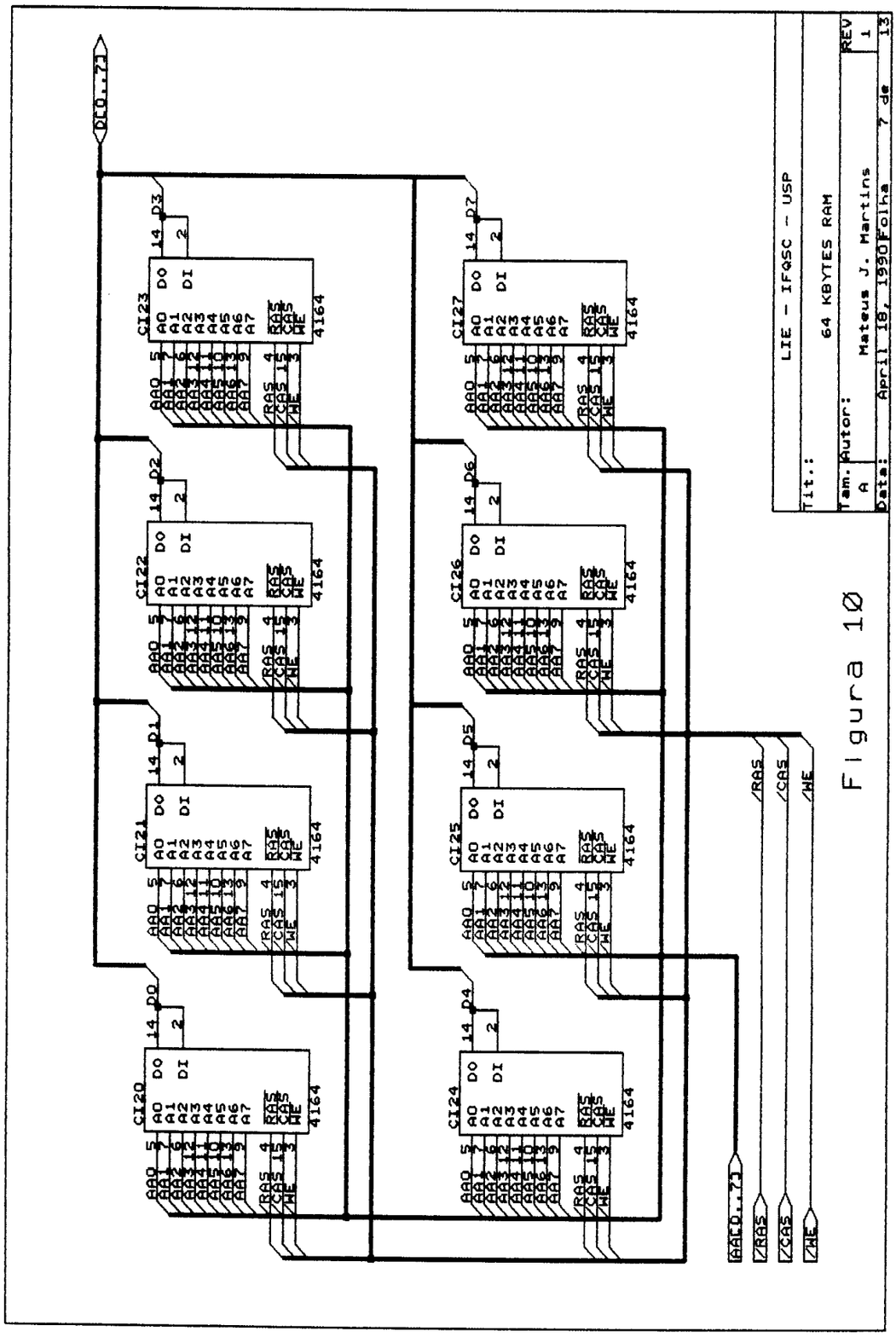

DI0..7
CI20
CI21
CI22
CI23
CI24
CI25
CI26
CI27
4164
LIE - IFQSC - USP
64 KBYTES RAM
Mateus J. Martins
April 18, 1990
DADO..7
/RAS
/CAS
/WE

Figura 10

Como esse duto apresenta uma carga muito maior que uma célula individual, o potencial armazenado na célula é todo perdido, porém antes que isso ocorra, entra em ação amplificadores com realimentação positiva restaurando o potencial original de cada linha do duto, e consequentemente de cada célula ligada ao duto. Há duas implicações nisso, primeiro há um tamanho mínimo do pulso /RAS, se este sinal tornar-se inativo antes das linhas do duto serem recarregadas pelos amplificadores, o conteúdo das células será perdido. Segundo, existe um tempo inativo mínimo do sinal /RAS para que os amplificadores voltem a seu estado balanceado, isto é, intermediário entre os níveis baixo e alto da célula. Para solucionar esses problemas foram utilizadas memórias de 150 nanosegundos (4164-15) e usando a linha /MREQ do Z80-UCP como linha de disparo do sinal /RAS, com isso pode-se utilizar o Z80 operando até 4 Mhz sem problemas.

Após ter-se carregado a primeira parte do endereçamento, ele é mudado para a parte mais significativa (endereçamento da coluna) e o sinal "column address strobe" (/CAS) é acionado, isto fornece os 8 bits mais significativos, juntamente com o bit armazenado quando do sinal /RAS, seleciona uma das 512 células do conjunto para o circuito de retenção. O /CAS também controla o estado dos circuitos de saída da memória. Enquanto este estiver ativo, a saída é habilitada e o bit selecionado é mantido na saída.

O ciclo de escrita é similar, tendo apenas como diferença que o sinal do pino "data imput" é roteado através do circuito selecionador para a célula. Durante um ciclo convencional de escrita o pino de saída da memória contém um dado não válido, como pode ser visto na Figura 11.

Aparentemente o uso de memórias dinâmicas em circuitos associados com memórias estáticas, onde a mesma linha e utilizada como entrada e saída, apresenta problemas. Utilizando-se um "buffer" com "three-state" entre os pinos de entrada e de saída pode-se contornar este problema, porém ao invés disso optou-se por um ciclo de escrita antecipada na memória, forçando o sinal /WE da memória a tornar-se

ativo antes do sinal /CAS. Assim o pino de saída permanece em alta impedância durante as operações de escrita (Figura 12).

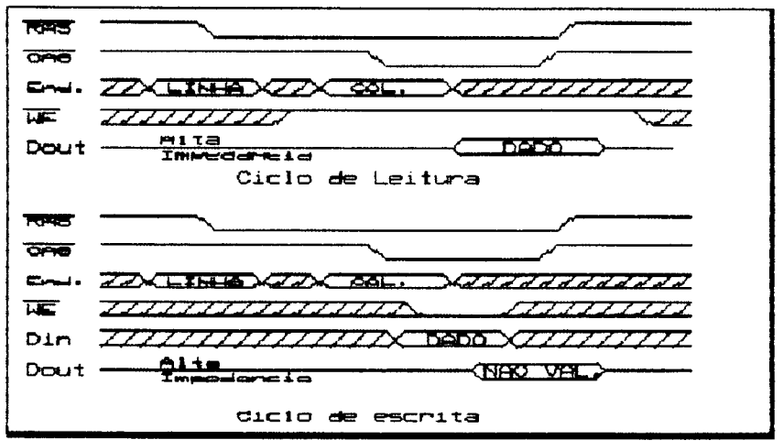


Figura 11 - Ciclo de leitura e escrita da 4164

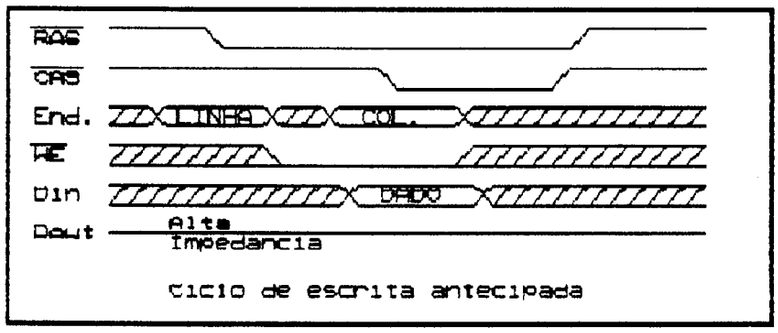


Figura 12 - Ciclo de escrita antecipada (4164)

Esse sinal é obtido invertendo-se a linha /RD e utilizando-a como sinal de escrita para as memórias, já que o sinal /WR não pode ser antecipado.

Para o refrescamento das memórias os 7 bits do Z80-UCP são suficientes, não necessitando de circuitos auxiliares, visto que internamente somente 7 bits são necessários para selecionar um conjunto dos 128 possíveis e os amplificadores encarregam-se de recarregar as 512 células desse conjunto.

4.2.7 Duto

O esquema do controle dos dutos de dados, endereço e controle, encontram-se na Figura 13.

Em todas as linhas da UCP foram instalados "buffers", viabilizando a ligação posterior de interfaces e isolando o "hardware" básico de problemas que ocorram em circuitos externos, montados pelo usuário.

A lógica de controle permite interrupções internas e externas, leitura ou escrita em dispositivos de entrada e saída e em bancos de memória extra. Toda entrada e saída de portos entre 00H a 1FH é reservado aos circuitos internos e qualquer acesso a portos entre 20H a FFH, associados a circuitos externos, destinado ao usuário.

4.2.8 E/S

Os esquemas dos circuitos de entrada e saída encontram-se divididos em 5 figuras. A Figura 14 mostra os circuitos decodificadores de endereço e os blocos básicos, compostos pelas linhas seriais, controlador de discos flexíveis ("FLOPPY") e a impressora padrão "Centronics".

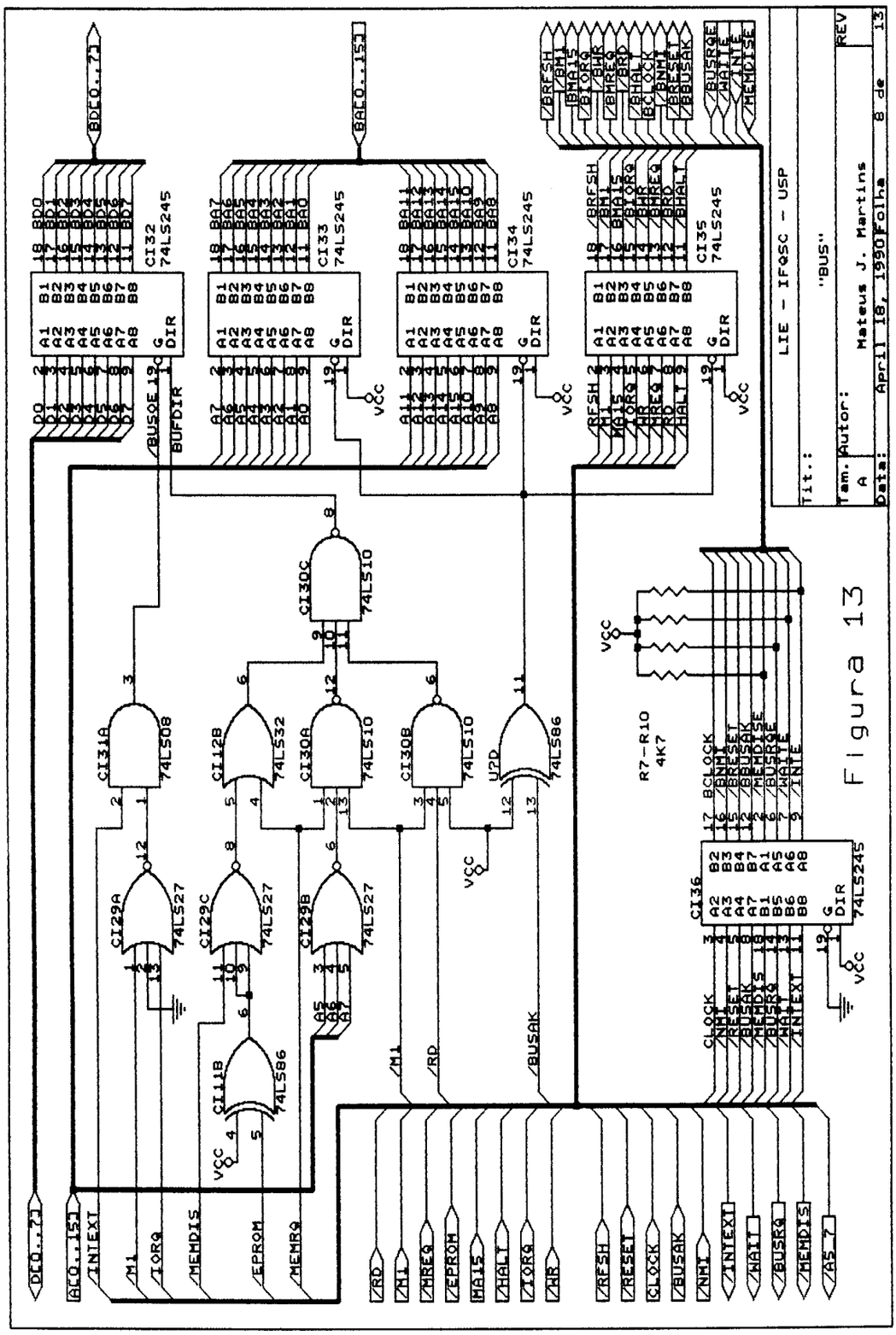
O circuito também fornece o sinal de habilitação para a seleção de bancos (/BANCO) e o sinal para o circuito de "clock" (/SPEED).

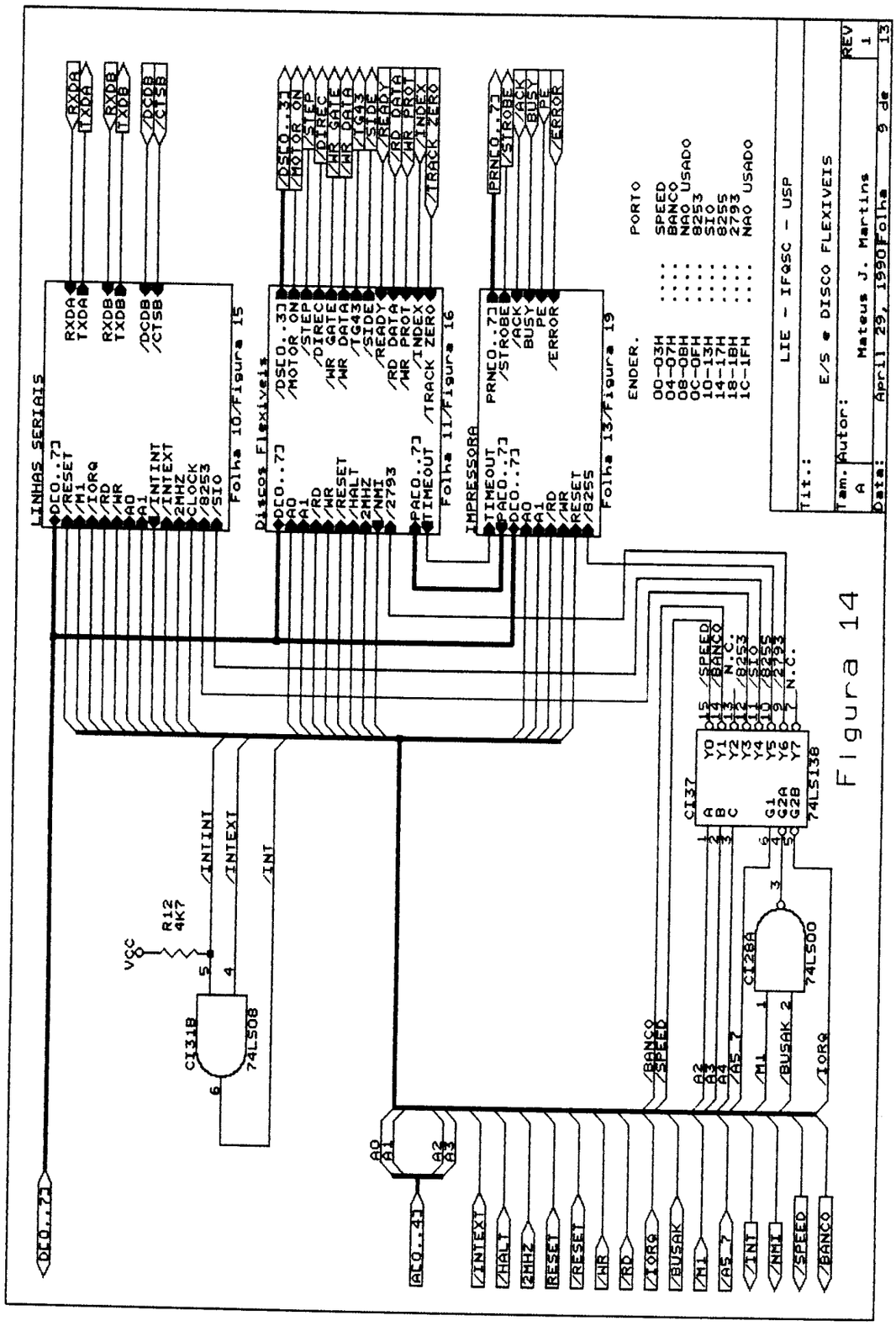
O decodificador de E/S foi colocado entre os endereço baixos de 00H a 1FH, deixando o restante ao usuário, que dessa maneira fica com a maior parte do espaço de endereçamento, facilitando assim o projeto de seu circuito.

A seguir analizar-se-á os módulos básicos mostrados na Figura 14.

4.2.8.1 Linhas Seriais

O circuito contendo 2 linhas seriais padrão RS232C, cujo esquema é mostrado na Figura 15. A primeira linha é utilizada como console tanto para o monitor como para o


LIE - IFSC - USP
Tit.: "BUS"
Autor: Mateus J. Martins
Data: April 18, 1990 Folha 8 de 13
REV
Tam. A
BDO..7
BAO..15
BUFDIR
BUSOE
CI32 74LS245
CI33 74LS245
CI34 74LS245
CI35 74LS245
CI36 74LS245
CI10A 74LS08
CI12B 74LS32
CI30C 74LS10
CI30A 74LS10
CI30B 74LS10
CI11B 74LS86
U2D 74LS86
CI29A 74LS27
CI29C 74LS27
CI29B 74LS27
R7-R10 4K7
VCC
M1
RD
BUSAK
IORQ
MEMDIS
EPROM
MEMRQ
INTEXT
RFSH
RESET
CLOCK
BUSAK
NMI
INTEXT
WAIT
BUSRQ
MEMDIS
A5..7
Figura 13


LINHAS SERIAIS
Folha 10/Figura 15
Discos Flexiveis
Folha 11/Figura 16
IMPRESSORA
Folha 13/Figura 19
PORTO
ENDER.
00-03H SPEED
04-07H BANCO
08-0BH NAO USADO
0C-0FH 8253
10-13H SIO
14-17H 8255
18-1BH 2793
1C-1FH NAO USADO
LIE - IFQSC - USP
E/S e DISCO FLEXIVEIS
Mateus J. Martins
April 29, 1990
R12 4K7
74LS08
74LS138
74LS00
Figura 14

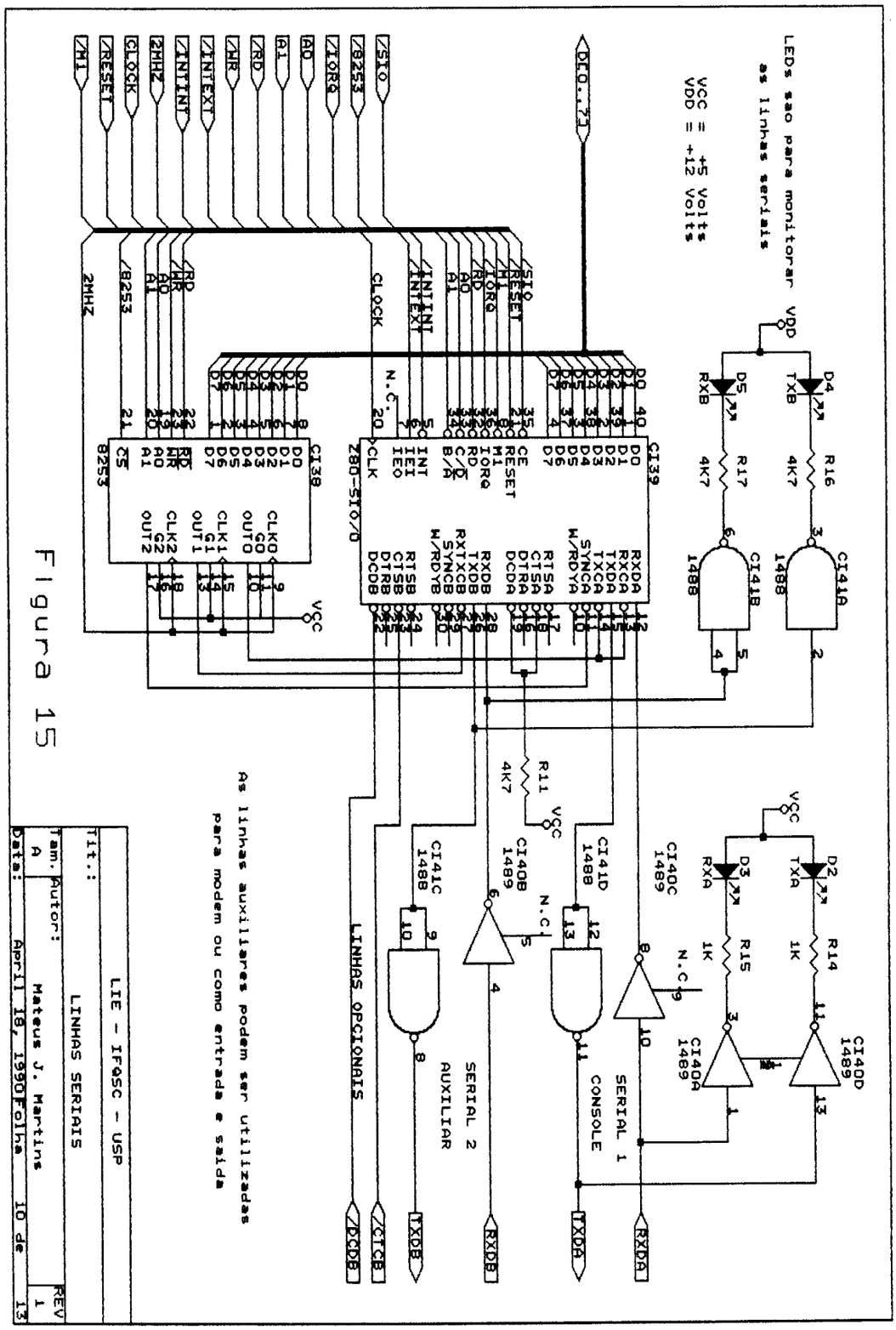

LEDs são para monitorar as linhas seriais
VCC = +5 Volts
VDD = +12 Volts
D[0..7]
/SIO
/8253
/IORQ
/RD
A0
A1
/WR
/INTEXT
/INTINT
2MHZ
CLOCK
/RESET
/M1
CI39
Z80-SIO/0
CI38
8253
CI41B
CI41A
CI40D
CI40C
CI40B
CI40A
CI41C
CI41D
1488
1489
R16
R17
R15
R14
R11
4K7
1K
VDD
VCC
N.C.
As linhas auxiliares podem ser utilizadas para modem ou como entrada e saída
LINHAS OPCIONAIS
SERIAL 2
AUXILIAR
SERIAL 1
CONSOLE
/DCDB
/CTSB
/TXDB
/RXDB
/TXDA
/RXDA
LIE - IFQSC - USP
LINHAS SERIAIS
Tam.
A
Autor:
Mateus J. Martins
Data:
April 18, 1990
Folha
10 de
13
REV
1

Figura 15

CP/M, nesta linha ligou-se um terminal de vídeo. A segunda linha é utilizada como auxiliar, para transferência de dados entre sistemas ou como controle de outros dispositivos associados ao micro. A taxa de transferência das duas linhas seriais é programada por "software" de 10 a 9600 bits por segundo independentemente. Além disso, o circuito possui um contador ligado ao Z80-SIO, e utilizando a versatilidade deste, o usuário pode gerar interrupções em intervalos definidos ou utilizar como relógio por "software".

O circuito dispõem ainda de 4 LEDs usados na monitoração das linhas, de recepção e de transmissão.

4.2.8.2 Controlador de discos flexíveis

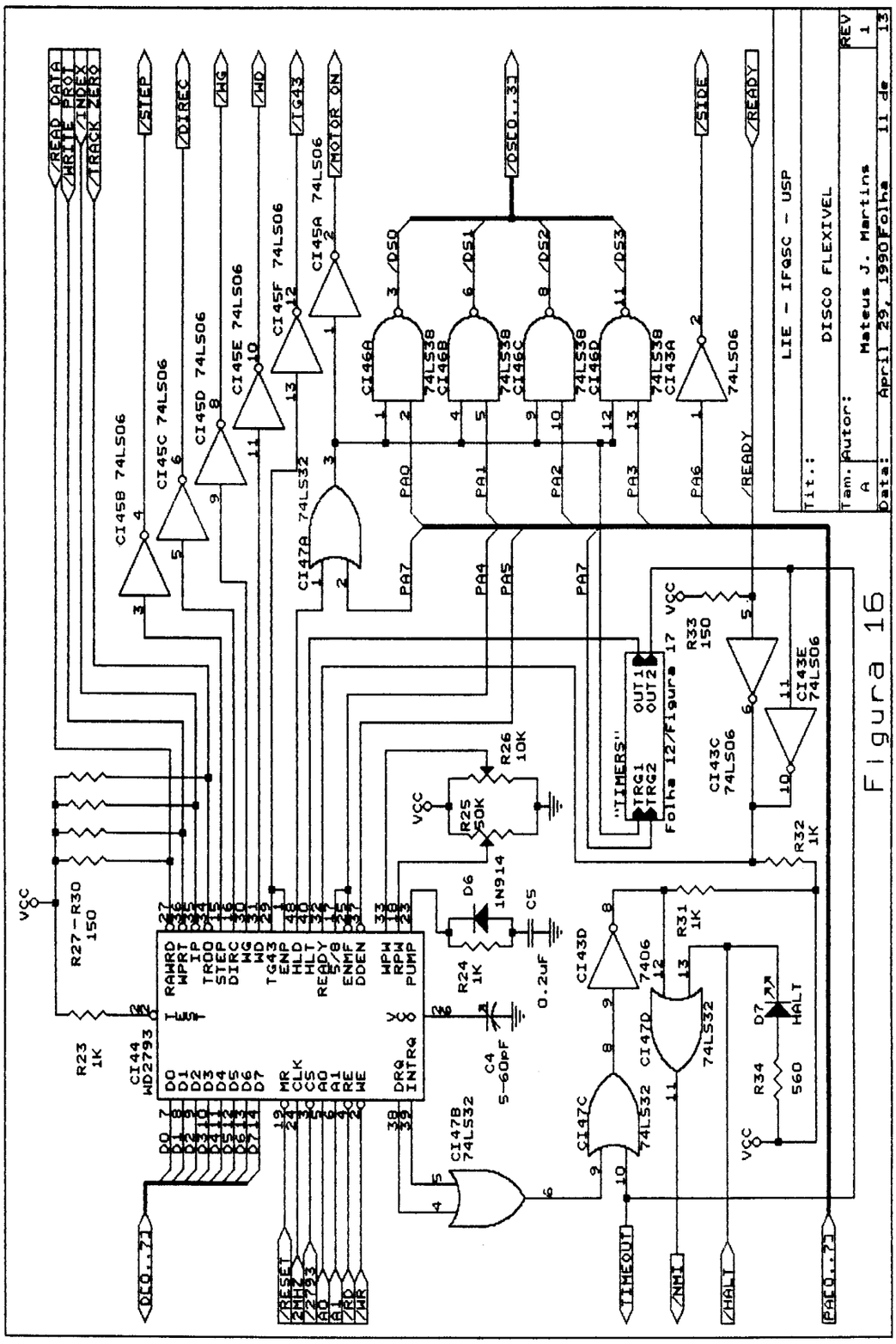
O esquema do circuito controlador de discos flexíveis está dividido em 2 figuras. Na Figura 16 pode-se ver todo o circuito de controle do "floppy" e na Figura 17, os temporizadores utilizados pelo controlador.

Um CI dedidado, o WD2793[19] da "Western Digital", foi utilizado como controlador, que permite ler ou escrever em discos de 5 1/4 ou 8 polegadas, em densidade simples (FM) ou dupla (MFM), com setores de 128, 256, 512 e 1024 bytes.

A programação do tipo de disco, densidade e número de lados é feita por "software", através do porto A de uma 8255, que será vista no ítem interface de impressora.

O circuito mostrado é capaz de controlar até 4 discos flexíveis, com diferentes especificações. Para facilitar o usuário quando estiver utilizando o CP/M, foi desenvolvido um programa chamado **CONFIG** que informa o sistema o tipo de disco instalado pelo usuário. Esse programa possui uma série de formatos dos micros nacionais, afim de compatibilizar a transferência de dados entre esses.

O controlador utiliza dois temporizadores mostrados na Figura 17, sendo o primeiro de 250 milisegundos utilizado para aguardar a estabilização da velocidade do motor do "drive", quando do acionamento deste. O segundo utilizado para cancelar a operação do disco, caso qualquer acesso


LIE - IFQSC - USP
DISCO FLEXIVEL
Mateus J. Martins
April 29, 1990
READ DATA
WRITE PROT
INDEX
TRACK ZERO
STEP
DIREC
HG
WD
TG43
MOTOR ON
DS0..3
SIDE
READY
CI44 WD2793
CI45B 74LS06
CI45C 74LS06
CI45D 74LS06
CI45E 74LS06
CI45F 74LS06
CI45A 74LS06
CI46A 74LS38
CI46B 74LS38
CI46C 74LS38
CI46D 74LS38
CI43A 74LS06
CI47A 74LS32
CI47B 74LS32
CI47C 74LS32
CI47D 74LS32
CI43D 7406
CI43C 74LS06
CI43E 74LS06
"TIMERS"
Folha I2/Figura 17
R23 1K
R24 1K
R25 50K
R26 10K
R27-R30 150
R31 1K
R32 1K
R33 150
R34 560
C4 5-60pF
C5 0.2uF
D6 1N914
D7 HALT
VCC
RESET
TIMEOUT
NMI
HALT
DIO..7
PAO..7
Figura 16

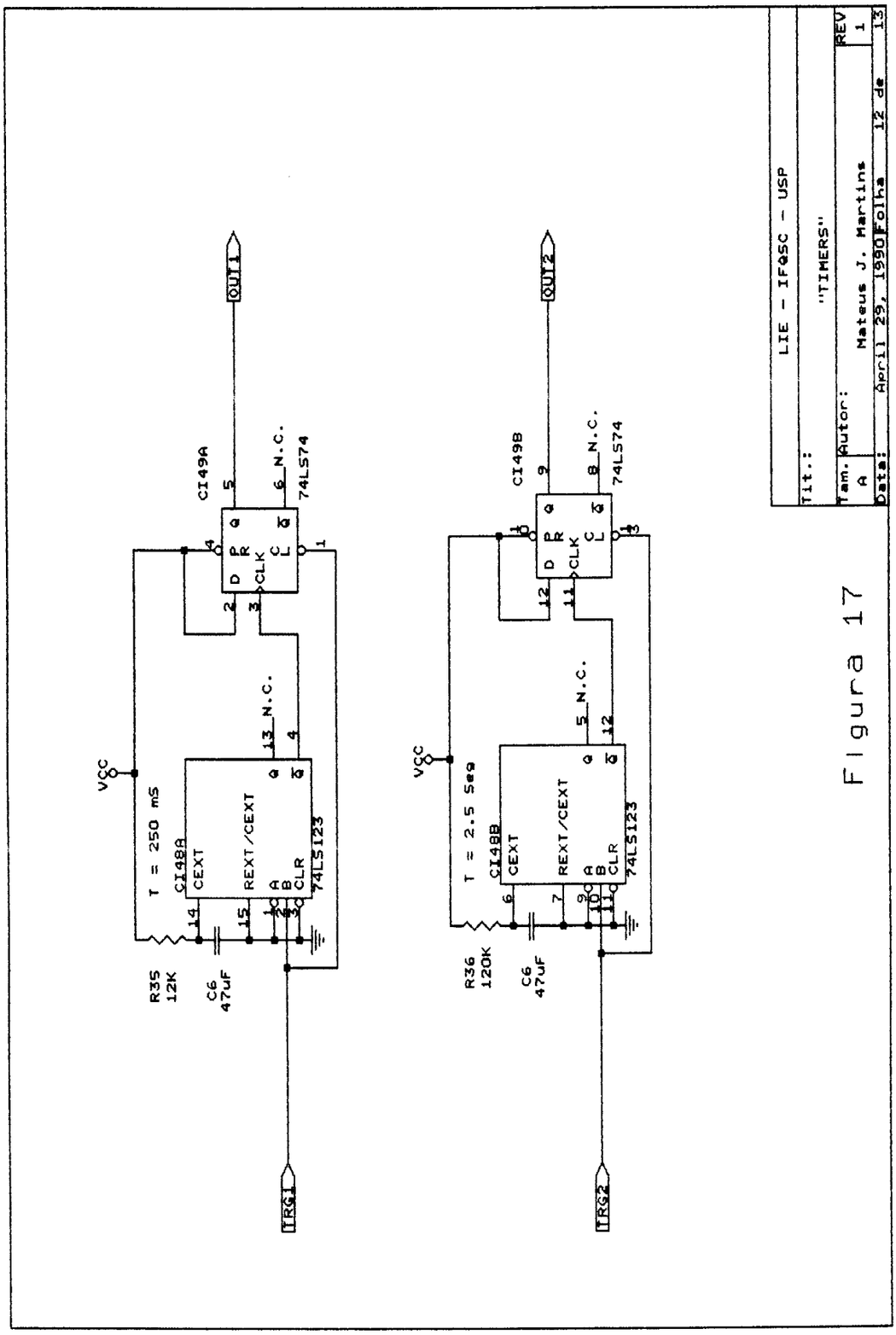

LIE - IFSC - USP
Tit.: "TIMERS"
Tam. A
Autor: Mateus J. Martins
Data: April 29, 1990 Folha 12 de 13
REV 1
OUT1
OUT2
CI49A
CI49B
74LS74
N.C.
D
CLK
R
S
Q
VCC
T = 250 mS
T = 2.5 Seg
CI48A
CI48B
CEXT
REXT/CEXT
A
B
CLR
74LS123
R35 12K
R36 120K
C6 47uF
TRG1
TRG2

Figura 17

passe de 2,5 segundos, funcionando assim como um "Time Out", caso o usuário acione um "drive" inexistente ou com problemas.

Para transferir os dados lidos ou escritos do disco para a memória, utiliza-se normalmente um circuito para acesso direto a memória (DMA), isto devido ao pequeno tempo para tal operação (16 microsegundos em densidade dupla). Porém o uso de DMA aumenta o custo e a complexidade dos circuitos, assim solucionou-se o problema usando-se uma "pseudo DMA", mostrada em detalhes na Figura 18.

Quando a UCP necessita receber ou enviar um setor do disco para o controlador, este entra em "HALT", parando o processamento e aguardando uma interrupção. Quando isso ocorre, a linha "HALT" se torna válida e permite a passagem de uma interrupção não mascarável (NMI). No momento que o controlador requisita uma transferência (DRQ), o processamento do micro continua com a transferência do dado desejado através de uma interrupção NMI. Assim não é necessário testar o controlador todo o tempo, sendo o motivo principal do tempo não ser suficiente para a tranferência, deixando tal operação para o final, onde o tempo agora não é mais importante.

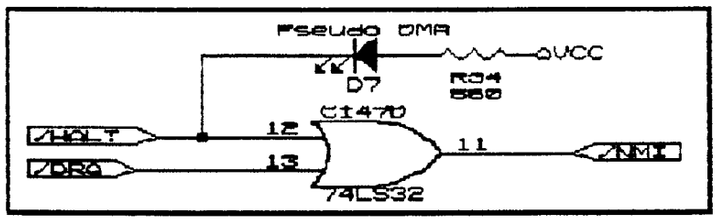


Figura 18 - Pseudo DMA.

Um Led colocado na linha /HALT, indica quando a pseudo DMA esta em operação.

A fim de verificar a validade desse método, pode-se calcular a taxa de transferência de uma rotina de leitura da unidade de disco flexível e compara-la com a taxa de transferência do disco.

Abaixo é fornecido um exemplo de uma rotina para ler dados do controlador de discos flexíveis.

```
        Org 66h         ; Endereço de atendimento da NMI

        RETN            ; Instrução para retornar ao
                        ; programa em HALT

        .

        ; Programa do usuário

        .

        LD HL,ENDBUF    ; Move para o registrador HL o
                        ; endereço do "Buffer" de leitura

        LD B,128        ; Move para o registrador B o
                        ; número de bytes a serem lidos

        LD C,PORTO      ; Move para o registrador C o
                        ; endereço do Porto de leitura

Lp:     HALT            ; Para o processador até ocorrer
                        ; uma interrupção NMI

        INI             ; Recebe o byte lido pelo porto
                        ; indicado em C, armazena o byte no
                        ; endereço indicado por HL,
                        ; incrementa HL, decrementa B e
                        ; coloca os bits da palavra de
                        ; estado de acordo com o conteúdo
                        ; de B

        JP NZ,Lp        ; Se o registrador B não chegou a
                        ; zero, volta ao endereço Lp

        .

        .               ; Continua e verifica o estado

        .
```

Descontado o tempo de inicialização dos registradores B, C e HL, verifica-se que o tempo é principalmente gasto na execução das instruções: INI, HALT, RETN e JP NZ,Lp, que contituem o laço principal da leitura dos 128 bytes. Considerando que a sincronização inicial é feita pela primeira instrução HALT e a respectiva espera pelo sinal NMI, em cada ciclo do laço irá ocorrer um atrazo de no máximo um ciclo de "clock" para o atendimento da interrupção. Então o tempo máximo para a transmissão de um byte é o seguinte:

16 ciclos-t para a instrução INI (ou OUTI)

4 ciclos-t para a instrução HALT

14 ciclos-t para a instrução RETN

10 ciclos-t para a instrução JP NZ,Lp

1 ciclo-t para a sincronização fina

num total de 45 ciclos-t do processador UCP-Z80A. Como já visto a frequência de "clock" de trabalho é de 4 MHz, assim cada ciclo-t tem uma duração de 250 nanosegundos (ns), o que resulta em 11.250 ns para cada byte transferido. Isso garante uma taxa de transferência de 88,8 Kbytes por segundo entre o micro e o controlador de discos flexíveis, o qual tem a mais alta taxa de transferência de dados de 67 Kbytes por segundo, quando se trata de disquetes de 8 polegadas operando em Densidade Dupla. Verifica-se assim, que o microprocessador tem velocidade suficiente para transferir dados ao controlador sem a necessidade de um circuito de DMA.

4.2.8.3 Impressora

A interface paralela, padrão "Centronics", para a impressora encontra-se na Figura 19. Embora o circuito tenha sido destinado a interface com a impressora, ele pode ser reprogramado por "software" pelo usuário para qualquer finalidade, contendo 8 linhas de entrada ou de saída e mais 5 linhas de controle.

O porto A da 8255[1&3], como mencionado anteriormente, é utilizado para o controlador de discos flexíveis, só sendo possível sua utilização caso o usuário não use disco.

O ponto TP1 é utilizado para informar o BIOS, que o usuário instalou um controlador de discos rígidos ("Winchester") com o sistema operacional CP/M, assim o carregamento do sistema operacional se fará pelo "Winchester" ao invés do "Floppy".

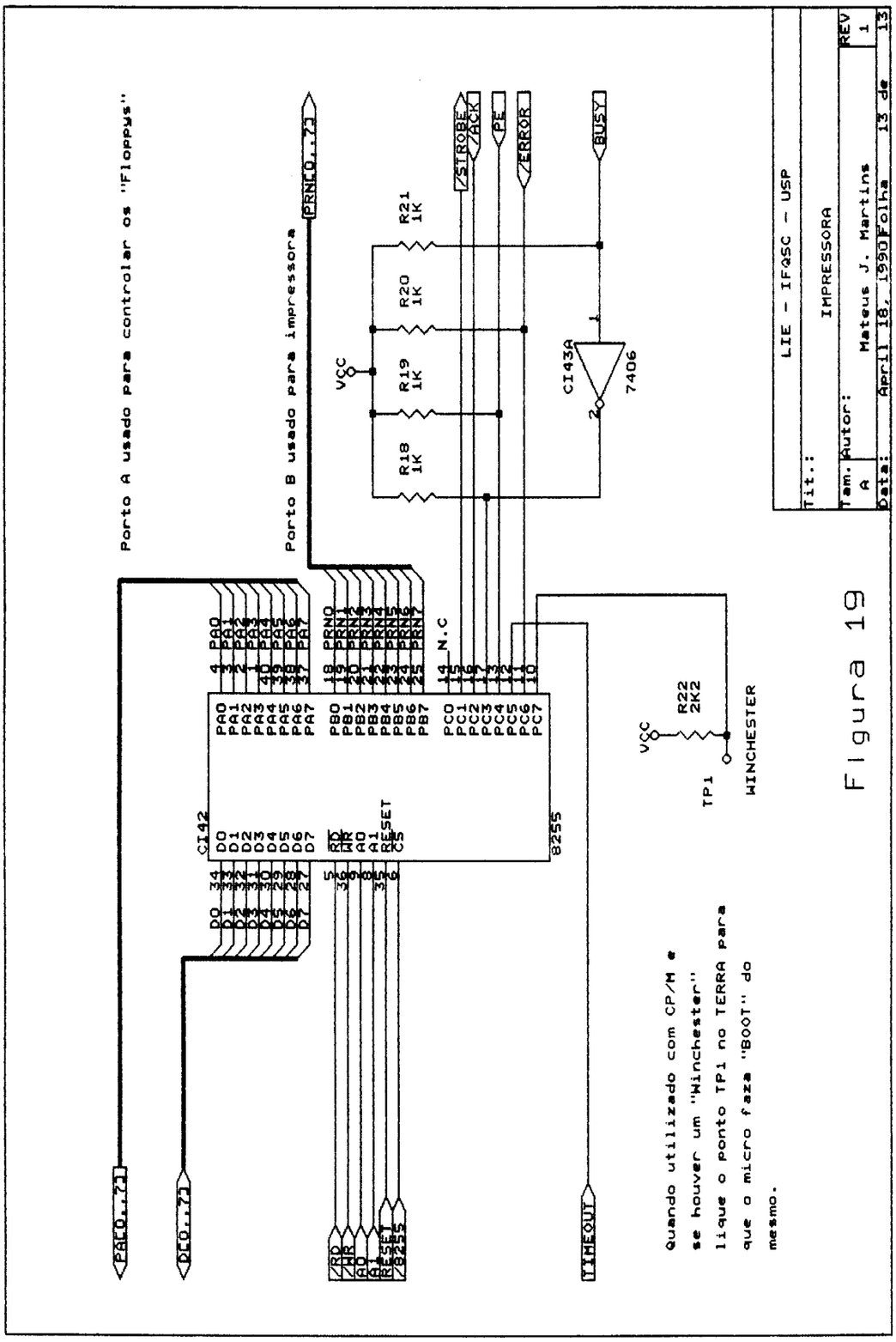

Porto A usado para controlar os "Floppys"
Porto B usado para impressora
PRNO..7
PACO..7
DCO..7
TIMEOUT
/STROBE
/ACK
PE
/ERROR
BUSY
R21 1K
R20 1K
R19 1K
R18 1K
VCC
CI43A
7406
CI42
8255
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
N.C
D0
D1
D2
D3
D4
D5
D6
D7
RD
WR
A0
A1
RESET
CS
R22 2K2
TP1
WINCHESTER
Quando utilizado com CP/M e
se houver um "Winchester"
ligue o ponto TP1 no TERRA para
que o micro faza "BOOT" do
mesmo.
LIE - IFQSC - USP
Tit.:
IMPRESSORA
Tam. A
Autor: Mateus J. Martins
Data: April 18, 1990 Folha 13 de 13
REV 1

Figura 19

4.3 EXPANSÕES PARA O "HARDWARE"

Para melhorar o tempo de processamento e a capacidade de armazenamento, foram projetadas duas interfaces, uma destinada ao controlador de discos rigido e outra para servir como disco virtual, que tem possibilidade de ser utilizado em outras aplicações.

Normalmente os micros que utilizam o sistema operacional CP/M não dispõem de tais interfaces, pelo fato de utilizarem grandes quantidades de memória para a tabela ALV do CBIOS, sobrando pouca memória para os programas aplicativos. Utilizando-se o chaveamento de bancos e rotinas especiais no EDOS, tornou tais interfaces disponíveis para o micro LIE.

Essas interfaces são abordadas a seguir.

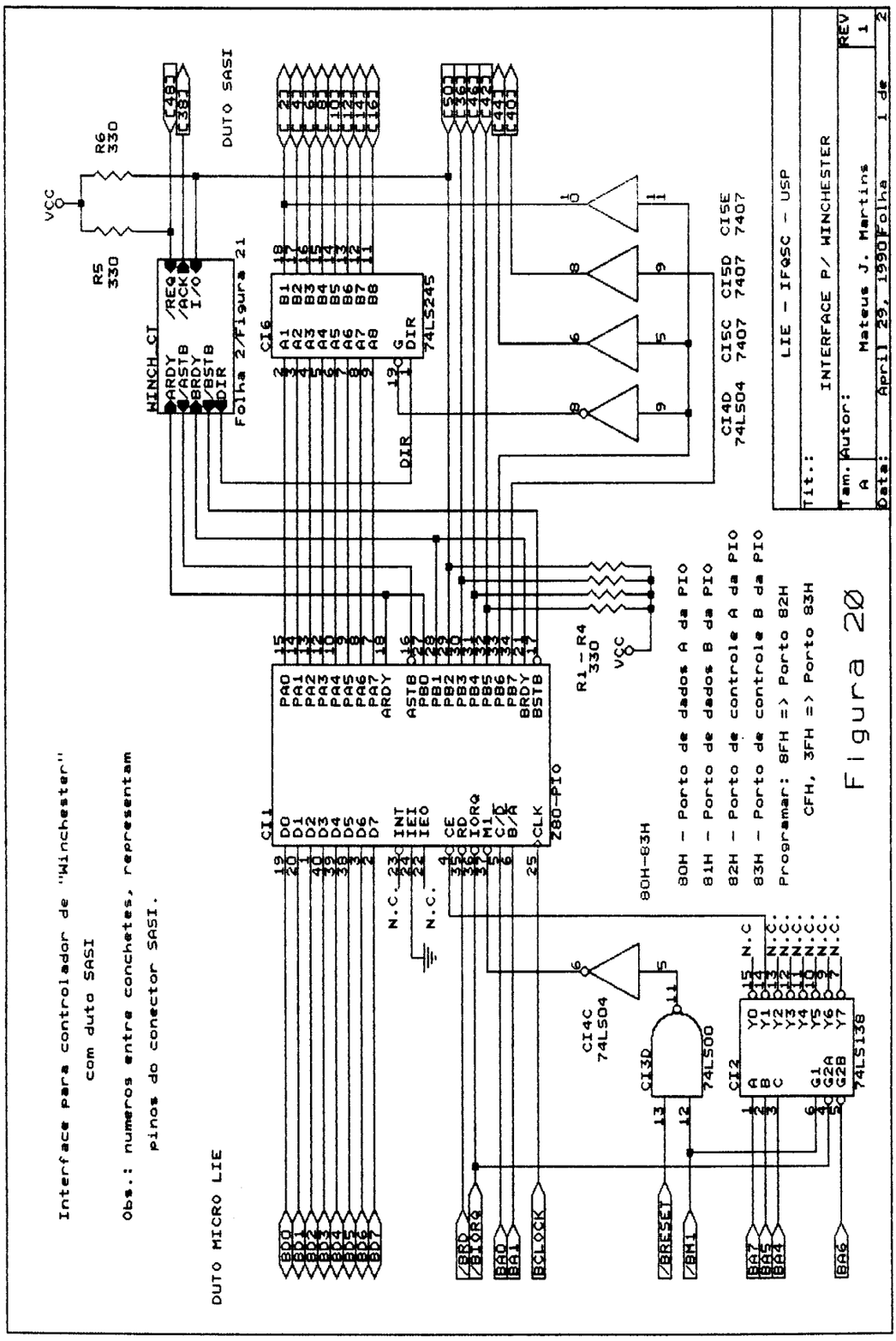
4.3.1 Interface para disco rígido

As Figuras 20 e 21 mostram o esquema do circuito que serve como interface entre o Micro LIE e o controlador de discos rígidos. A Figura 22 mostra o protótipo da interface montada em placa padrão, essa interface converte os sinais do duto do Micro LIE para um duto tipo SASI ("Shugart Associates System Interface"), que é utilizado em controladores de "winchester".

Na Figura 21 tem-se o esquema do circuito destinado a geração dos sinais necessários à transferência de dados, enquanto que a interface propriamente dita encontra-se na Figura 20.

No protótipo foi utilizado um "winchester" de 10 Mbytes (ST412 da Seagate) e um controlador WD1002-SHD que permite o controle de até duas unidades de disco rígido[20]. Pode-se utilizar discos com maior capacidade porém será necessário aumentar o tamanho do bloco de alocação, isto é, o valor do parâmetro BSH e BLM.

O controlador encarrega-se de gerar todos os sinais necessários à transferência de dados entre este e o disco


Interface para controlador de "Winchester"
com duto SASI
Obs.: numeros entre conchetes, representam
pinos do conector SASI.
DUTO MICRO LIE
DUTO SASI
WINCH_CT
/REQ
/ACK
I/O
/RDY
/ASTB
/BRDY
/BSTB
DIR
Folha 2/Figura 21
CI6
74LS245
CI1
280-PIO
CI4C
74LS04
CI3D
74LS00
CI2
74LS138
CI4D
74LS04
CI5C
7407
CI5D
7407
CI5E
7407
R1-R4
330
R5
330
R6
330
VCC
80H-83H
80H - Porto de dados A da PIO
81H - Porto de dados B da PIO
82H - Porto de controle A da PIO
83H - Porto de controle B da PIO
Programar: 8FH => Porto 82H
CFH, 3FH => Porto 83H
LIE - IFQSC - USP
INTERFACE P/ WINCHESTER
Tit.:
Tam.
A
Autor:
Mateus J. Martins
Data:
April 29, 1990
Folha
1 de
REV
1
2
Figura 20

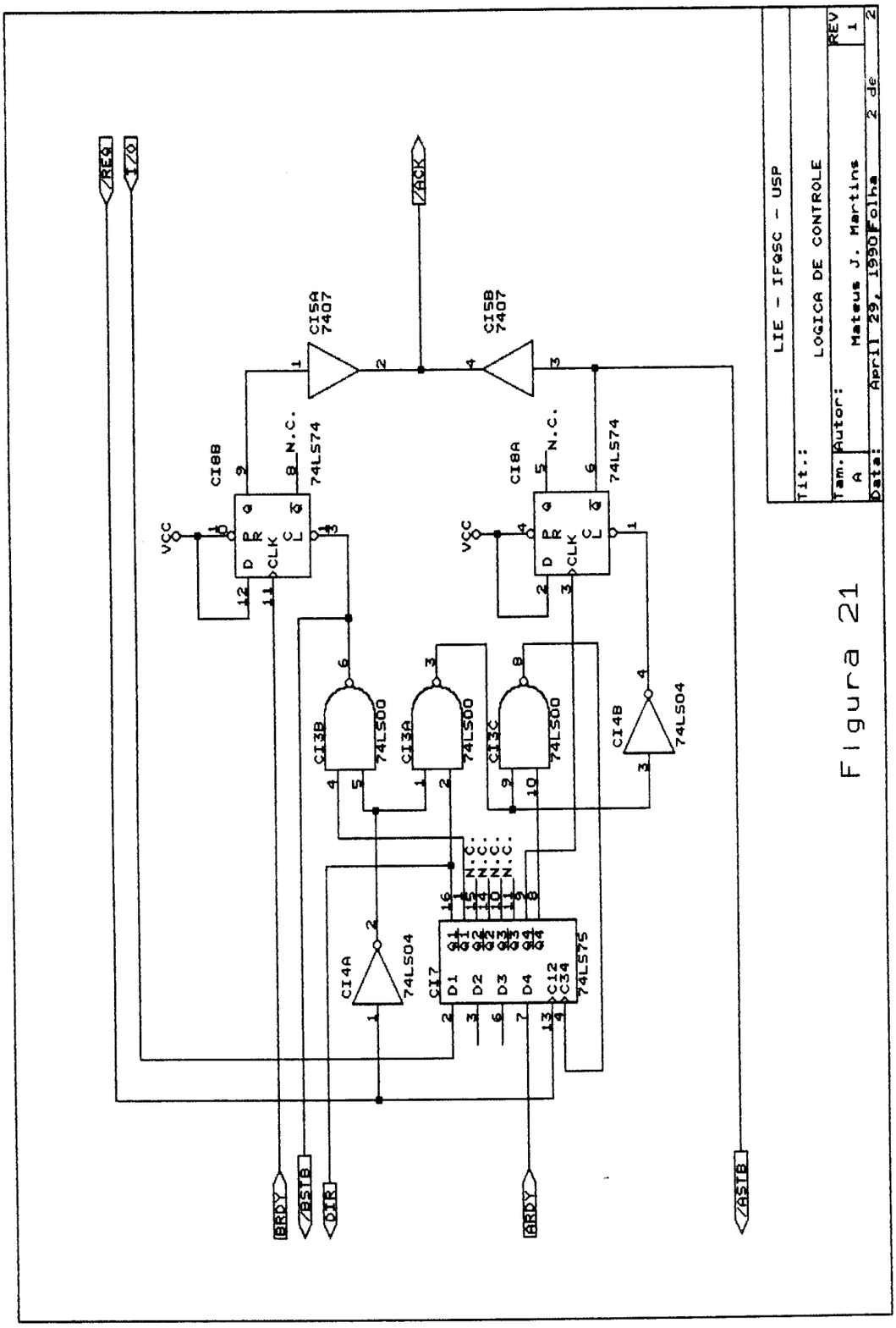

/REQ
IT/0
/ACK
CI5A 7407
CI5B 7407
CI8B
CI8A
74LS74
N.C.
VCC
D PR Q
CLK C L Q
CI3B 74LS00
CI3A 74LS00
CI3C 74LS00
CI4B 74LS04
CI4A 74LS04
CI7 74LS75
D1 D2 D3 D4
C12 C34
/BRDY
/BSTB
DIR
/ARDY
/ASTB
LIE - IFQSC - USP
Tit.: LOGICA DE CONTROLE
Tam. A
Autor: Mateus J. Martins
Data: April 29, 1990
Folha 2 de 2
REV 1

Figura 21

rígido. A comunicação entre o Micro LIE e o controlador é feito através de comandos de alto nível, ou seja, o disco apresenta-se como uma série de setores continuos, onde o micro informa qual setor deseja ler ou escrever.

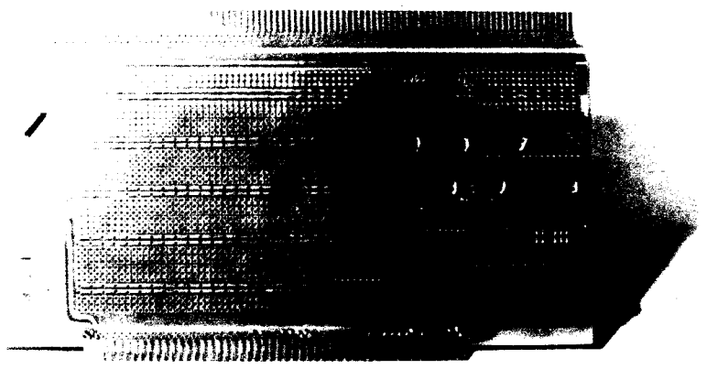


Figura 22 - Interface para disco rigido, montada em placa padrão.

A maior parte do trabalho é executado pelas rotinas internas do EDOS, as quais convertem setores lógicos em físicos e controlam as tabelas do CBIOS. A tabela ALV de alocação de blocos do "Winchester", mantida na RAM estática, é controlada pelo EDOS e trocada com as tabelas do restante dos discos, mantida pelo CBIOS, quando o BDOS necessita de informações deste. Isto é feito pois, o tamanho da área ALV torna-se muito grande com o aumento da capacidade de armazenamento do disco. Assim o uso dessa técnica torna possível a instalação de dispositivos de grande capacidade sem a perda no tamanho da área TPA, pois o CBIOS necessitaria de maior espaço para essas tabelas.

4.3.2 Disco Virtual

O disco virtual nada mais é que um conjunto de bancos de memória, com o qual o "software" simula um disco,

assim o usuário tem a impressão de estar trabalhando como um disco flexível, com setores, trilhas e lado, porém com uma velocidade de acesso muito maior. Os discos virtuais são excelentes para manter dados temporários, por exemplo durante a compilação de um programa ou a coleta de grandes quantidades de dados a altas velocidades.

O circuito do disco virtual é um opcional, e o usuário pode acrescentar no seu sistema sem qualquer modificação, no "hardware" e no "software". O protótipo montado com 512 Kbytes em placa padrão pode ser visto na Figura 23 e seu esquema encontra-se dividido em 4 partes.

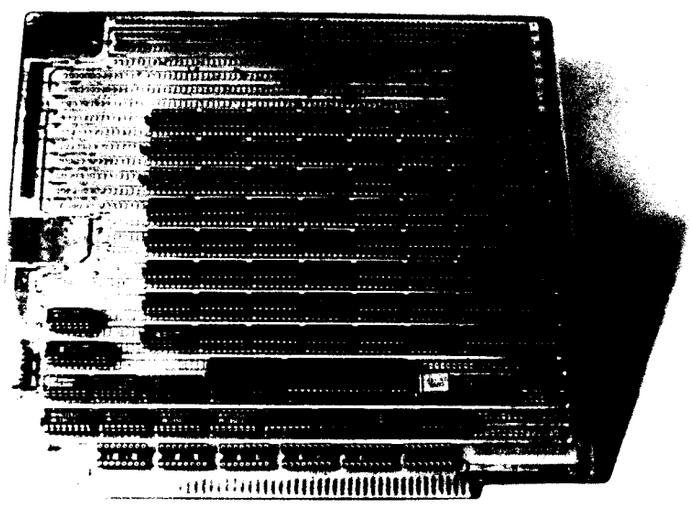


Figura 23 - Protótipo da placa do Disco Virtual.

Na primeira parte, Figura 24, encontra-se o controlador das RAMs dinâmicas, necessário para o refrescamento e geração dos sinais necessários. Foi utilizado um controlador próprio o 8203 da Intel Inc.[17], ao invés do Z80-UCP, isto por que o circuito foi projetado para ser utilizado em outras aplicações sem o microcomputador.

Na segunda parte tem-se a lógica para seleção, o decodificador de endereços e os contadores, Figura 25. Os

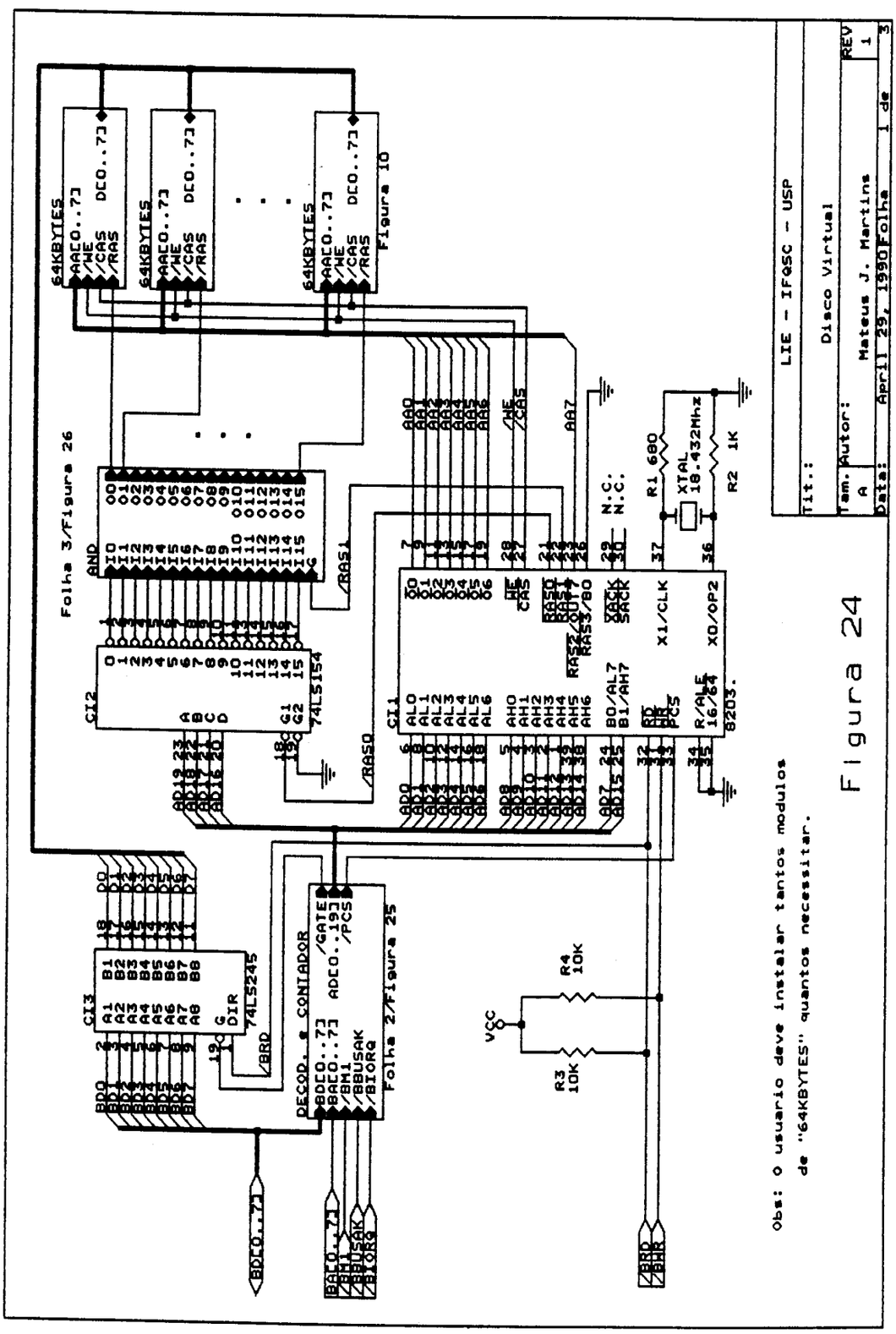

64KBYTES
AACO..7J
/WE
/CAS
/RAS
DCO..7J
Figura 10
Folha 3/Figura 26
74LS154
74LS245
DECOD. e CONTADOR
Folha 2/Figura 25
8203.
R1 680
R2 1K
R3 10K
R4 10K
XTAL 18.432MHz
VCC
N.C.
LIE - IFQSC - USP
Tit.: Disco Virtual
Tam. A
Autor: Mateus J. Martins
Data: April 29, 1990 Folha 1 de 3
REV 1
Obs: O usuario deve instalar tantos modulos
de "64KBYTES" quantos necessitar.

Figura 24

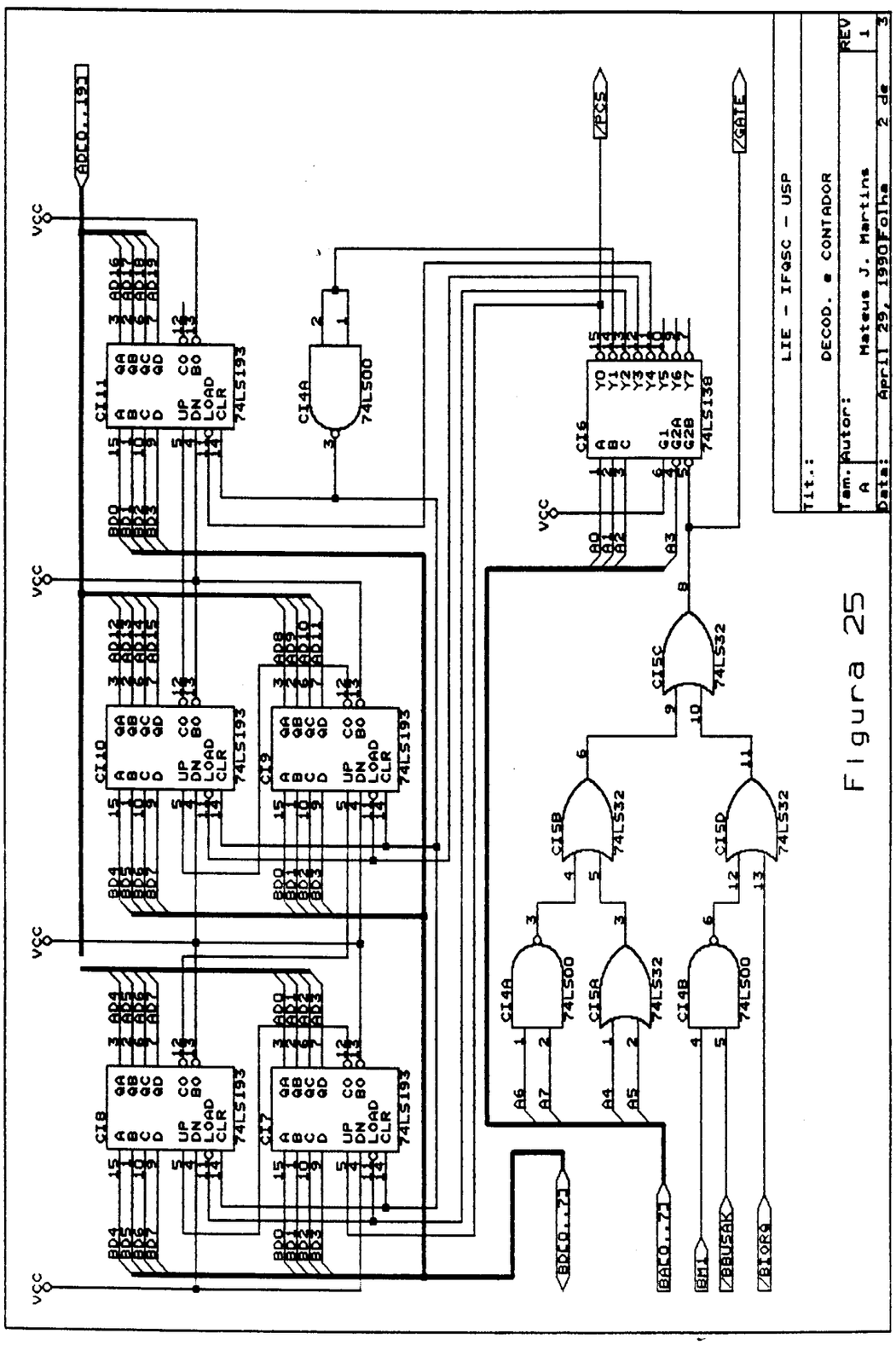

ADC0..15
VCC
CI11
CI10
CI9
CI8
CI7
74LS193
CI4A
74LS00
CI6
74LS138
CI5C
CI5B
CI5D
74LS32
CI4A
CI4B
CI5A
74LS00
74LS32
ZPCS
ZGATE
BDC0..7
BAC0..7
EM1
ZBBUSAK
ZBIORQ
LIE - IFQSC - USP
Tit.:
DECOD. e CONTADOR
Tam. Autor: Mateus J. Martins
A
Data: April 29, 1990 Folha 2 de 3
REV 1

Figura 25

contadores são responsáveis pela geração dos endereços da RAM. Para ler ou escrever na memória, basta carregar os contadores com o endereço inicial dos dados, obtido do número do setor, trilha e lado que deseja-se acessar, e com uma instrução INIR ou OUTIR. Estas últimas são instruções, com um número determinado de repetições, de leitura e escrita em E/S respectivamente. Tais instruções podem ser utilizadas, pois no final de cada acesso ao disco, os contadores são incrementados automaticamente para o novo endereço.

Na terceira parte encontra-se um conjunto de portas "AND", Figura 26, destinadas a geração do sinal /RAS para todas as RAMs.

A última parte do disco virtual contém os bancos de memória RAM, esses bancos são identicos ao da UCP (Figura 10). O disco pode possuir de 64 Kbytes a 1 Mbyte de memória (1 a 16 bancos), bastanto o usuário acrescentar tantos módulos quantos deseje. O protótipo foi construido com 512 Kbytes de capacidade.

Pelo fato de poder-se ler e escrever desde um byte até o disco todo através de um porto de E/S, facilita a programação das rotinas de manejamento do disco virtual no CBIOS.

4.4 CARACTERISTICAS GERAIS "HARDWARE"

A seguir são fornecidos os endereços de todos os dispositivos de E/S (mapeamento de E/S):

-- 00H a 03H porto utilizado para mudar a velocidade de processamento. Escrevendo nesse porto o valor 0, 1, 2, 3, muda-se a velocidade para 2, 1, 4 e externa respectivamente. A velocidade inicial, após o "RESET" é de 2 Mhz.

-- 04H porto para selecionar o banco 0 e a parte superior do banco 1.

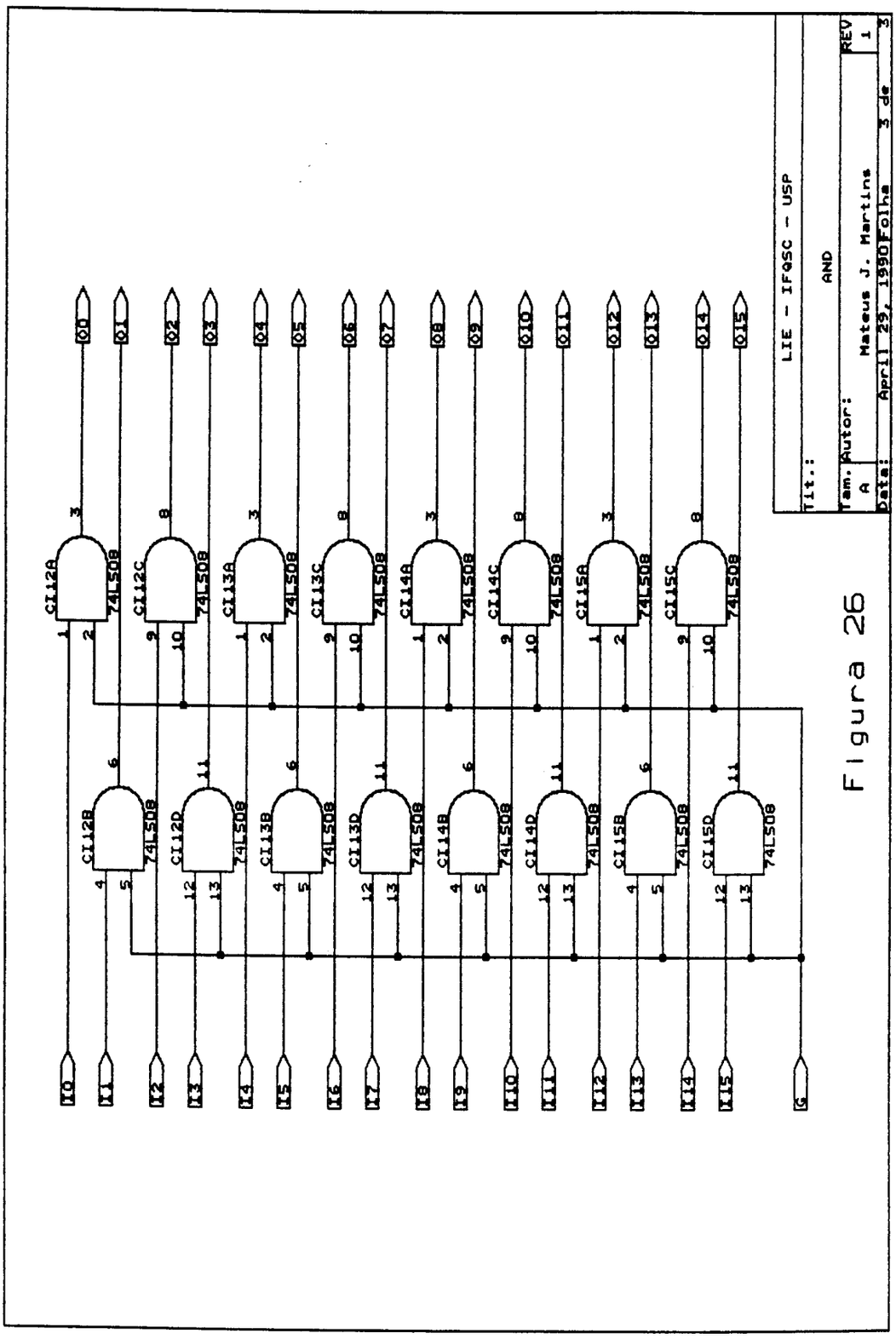

LIE - IFQSC - USP
Tit.: AND
Tam. A
Autor: Mateus J. Martins
Data: April 29, 1990 Folha 3 de 3
REV 1
74LS08
I0
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
I13
I14
I15
O0
O1
O2
O3
O4
O5
O6
O7
O8
O9
O10
O11
O12
O13
O14
O15
G

Figura 26

-- 05H seleciona o banco 0 e a parte inferior do banco 1.

-- 06H seleciona o banco 1 inteiramente.

-- 07H seleciona o banco 1 invertido.

-- 08H a 0BH não é utilizado.

-- 0CH contador 1 da 8253, é utilizado para gerar a frequência da linha serial 1.

-- 0DH contador 2 da 8253, é utilizado para gerar a frequência da linha serial 2.

-- 0EH contador 3 da 8253, é utilizado como gerador de interrupções, através do Z80-SIO.

-- 0FH registrador de controle e comando da 8253.

-- 10H registrador de dados do canal 1 do Z80-SIO.

-- 11H registrador de dados do canal 2 do Z80-SIO.

-- 12H registrador de controle do canal 1 do Z80-SIO.

-- 13H registrador de controle do canal 2 do Z80-SIO.

-- 14H porto A da 8255.

-- 15H porto B da 8255.

-- 16H porto C da 8255.

-- 17H registrador de controle e programação da 8255.

-- 18H registrador de controle e "status" do WD2793.

-- 19H registrador de trilha do WD2793.

-- 1AH registrador de setor do WD2793.

-- 1BH registrador de dados do WD2793.

-- 1CH a 1FH não utilizado.

-- 20H a FFH destinado ao usuário.

O controlador de "Winchester" e o disco virtual ocupam os seguintes portos no espaço do usuário:

-- 80H porto de dados PIO-A do controlador de "Winchester".

-- 81H porto de dados PIO-B.

-- 82H porto de controle PIO-A.

-- 83H porto de controle PIO-B.

-- C0H porto de leitura e escrita do disco virtual.

-- C1H "Clear" do endereço de acesso ao disco virtual.

-- C2H "Load" do endereço de A7 a A0 do disco virtual.

-- C3H "Load" do endereço de A15 a A8.

-- C4H "Load" do endereço de A19 a A16.

A seguir é mostrado o mapa de endereçamento da memória do microcomputador (mapeamento da memória):

-- 0000H a 1FFFH utilizado no monitor banco 0.

-- 2000H a 2FFFH no banco 0, destinado ao programa de controle do usuário.

-- 4000H a 5FFFH no banco 0, utilizado para a RAM estática, essa RAM é de 2 Kbytes embora o espaço seja de 8 Kbytes.

-- 8000H a FFFFH utilizado como janela para os outros bancos, quando o banco 0 esta ativo.

4.5 O "SOFTWARE"

O "software" desenvolvido para o micro LIE, foi dividido em duas partes, o *Monitor* e o *CBIOS/EDOS*. Quando o micro é utilizado no controle de sistemas, o usuário poderá utilizar o monitor para auxilio no desenvolvimento e depuração, pois este é um "software" autônomo contendo um conjunto de comandos e funções que permitem construir rotinas de controle e teste. Caso seja necessário utilizar o CP/M como sistema operacional, devido a grande quantidade de "software" disponível e publicado no mercado, poderá faze-lo com o auxilio do CBIOS/EDOS.

O CBIOS ("Custom Basic Input/Output") serve como interface entre a parte lógica do CP/M (CCP/BDOS) e a parte física do "hardware", e com o complemento do EDOS ("Eprom Disk Operating System") permite utilizar o CP/M com alguns melhoramentos e um maior desempenho.

Também foi desenvolvido uma versão do monitor chamado **DB** para ser utilizado no CP/M, com a finalidade de transportar o ambiente do monitor para o CP/M. Assim o usuário poderá desenvolver programas utilizando os compiladores disponiveis no CP/M e verificar como seria seu funcionamento com o monitor, em baixo do CP/M. Além disso o DB poderá ser utilizado como um ratreador de programas para o CP/M.

4.5.1 Monitor

Monitor é o termo utilizado para descrever um programa que historicamente permitia ao usuário "monitorar" a UCP, os registradores, a memória e a execução de um programa aplicativo. Originalmente os monitores eram pequenos programas, apenas para substituir os velhos painéis feitos de luzes e chaves. Mais tarde, mais e mais funções foram adicionadas, tornando sua utilização mais versátil e útil. Outro passo importante na evolução dos monitores, foi a separação das rotinas de entrada e saida (E/S) do processador de comandos, isto permitia ao usuário a

utilização dessas rotinas, sem a necessidade de duplicação em seu programa.

O monitor desenvolvido para o micro LIE, foi escrito em linguagem montadora do Z80-UCP[21-22] e ocupa 8 Kbytes de EPROM, contendo um editor e interpretador de comandos; montador e desmontador de código para o Z80, com operações aritméticas e lógicas; módulos para execução, monitorização e depuração de programas; além de permitir total acesso aos recursos do microcomputador (memória, registradores, etc).

Sua estrutura pode ser dividida em três partes principais: inicialização, comandos e funções.

A inicialização destina-se a colocar o microprocessador e seus dispositivos, em um estado conhecido e preparado para receber comandos.

Durante fase de inicialização, todos os periféricos são inicializados e a memória RAM do microcomputador é verificada, a fim de certificar-se seu correto funcionamento. Além disso as variáveis utilizadas no monitor são inicializadas, bem como o "stack pointer" do microprocessador e seu modo de interrupção.

Ao final da inicialização, é verificada a existência e a validade da EPROM do usuário, destinada ao programa de controle de um experimento. Caso exista e seja válida a execução é transferida a EPROM, que pode apenas, adicionar novos comandos e/ou funções ao monitor e retornar a execução, ou executar o programa de controle.

A segunda parte do monitor é o interpretador de comandos, cuja função é receber os comandos em forma de cadeias de caracteres e executar as operações necessárias a sua realização. O interpretador recebe controle quando não exista ou não seja válida a EPROM do usuário, ou ainda caso esta retorne ao monitor.

O interpretador contém vários comandos internos, podendo ser ampliável através de comandos externos, que o usuário pode instalar.

A terceira parte do monitor são as funções, as quais são executadas pelo monitor. Essas funções permitem a leitura e escrita em dispositivos de entrada e saida, bem como o controle dos periféricos existentes no micro-computador. Várias dessas funções são disponiveis ao programa do usuário. As funções são chamadas a partir de uma pseudo instrução do Z80-UCP, a '**JSYS n**' ("Jump to System"), onde **n** é um número em hexadecimal e representa uma determinada função do monitor. Essa pseudo instrução é gerada e executada através do "**RST 8**", sendo montada e desmontada pelas rotinas internas do monitor, e portanto totalmente transparente ao usuário.

4.5.1.1 Comandos Internos

O monitor contém uma série de comandos internos, destinados a monitorização dos recursos do microcomputador. Todos os comandos internos começam por uma letra, seguida ou não de parâmetros. Caso haja mais de um parâmetro estes devem ser separados por brancos ou por vírgulas.

Os parâmetros entre colchetes são opcionais, podendo ou não serem fornecidos pelo usuário. Caso não sejam fornecidos, o interpretador utilizará "defaults", que dependem do comando e se fornecidos podem possuir qualquer tipo de combinação e sequência entre as seguintes formas:

Valor + Valor	- Soma inteira de dois valores
Valor - Valor	- Subtração inteira de dois valores
Valor * Valor	- Multiplicação de dois valores inteiros, com resultado inteiro.
Valor / Valor	- Divisão inteira de dois valores
Valor & Valor	- Operação "And" entre dois valores
Valor ! Valor	- Operação "OR" entre dois valores
Valor % Valor	- Módulo entre dois valores, isto é, o resto da divisão entre eles
-Valor	- Complemento de dois do valor
~Valor	- Complemento de um do valor
Valor.	- Valor em decimal (base 10)

@	- Valor do deslocamento (@)
$	- Endereço corrente (PC)
'AB'	- Valor em ASCII das letras.
^RG	- Conteúdo do registrador RG
(Endereço)	- Conteúdo do endereço fornecido
[Parâmetro]	- Utilizado para permitir prioridade nos cálculos, pois as operações são executadas conforme sua ocorrência.

Quando o comando necessitar de dois parâmetros, por exemplo o endereço inicial e o endereco final, pode-se fornecer o endereço inicial e um tamanho ou deslocamento. Exemplo "1000 S300" é identico a "1000 1300". A letra **S** representa "size".

Além dessas formas pode-se utilizar comparações entre valores, cujo resultado é zero (0000H) caso seja falso a comparação ou FFFFH caso verdadeira. As seguintes comparações são válidas:

Valor1 > Valor2	- Valor1 é maior que o Valor2
Valor1 < Valor2	- Valor1 é menor que o Valor2
Valor1 = Valor2	- Valor1 é igual ao valor Valor2
Valor1 <> Valor2	- Valor1 é diferente do Valor2
Valor1 >= Valor2	- Valor1 é maior ou igual ao Valor2
Valor1 <= Valor2	- Valor1 é menor ou igual ao Valor2

O monitor assume que todos os valores estão na base 16, isto é são hexadecimais. A seguir é fornecido um exemplo de um único parâmetro:

(^BC)&FF00+[1024.*2], esse parâmetro é a soma entre 2048 (1024 em decimal vezes 2) e o conteúdo apontado pelo registrador BC "and" com FF00H. Se o conteúdo do registrador BC é 4000H e a posição 4000H da memória contém 1234H, o resultado é 1A00H.

Os comandos internos são:

@ - Relocação de endereços

Esse comando não possui parâmetro, sendo que o valor do endereço a ser relocado é pedido, logo após o comando ser acionado. Esse comando permite que os endereços listados durante a desmontagem das instruções, seja vizualizado como estando em um endereço diferente do qual se encontra. Por exemplo, suponha que montou-se um programa o qual deve ser colocado no endereço inicial 0000H. Porém esse endereço já está sendo ocupado pela EPROM do monitor, assim carrega-se o programa em outra posição, por exemplo em 8000H e utilizando o comando **@**, podemos listar o programa como se estivesse em 0000H, caso contrário os endereços listados seriam em relação a posição 8000H.

A [Endereço] - "Assembler" - Montagem de instruções

Esse comando permite a montagem de instruções, formando assim um programa em linguagem "Assembly". Esse comando pode ou não ser seguido de um endereço onde as instruções seram montadas. Caso o endereço não seja fornecido, o último endereço montado será utilizado, sendo seu valor inicial igual a 8000H. Antes de montar a nova instrução, a existente na posição corrente de memória será desmontada.

Para terminar-se o comando de montagem deve-se utilizar um ponto ".", sendo o endereço de onde terminou-se, utilizado como "default" para a próxima montagem. O montador reconhece todas as instruções da UCP-Z80, bem como a pseudo instrução "**JSYS n**".

B [X] [R] [Endereço] [:Contador] - "BreakPoint"

Esse comando estabelece pontos de parada para a execução do programa do usuário, podendo estabelecer até 12 pontos de parada ao mesmo tempo.

Caso nenhum parâmetro seja fornecido ao comando **B**, este imprimirá todos os pontos de parada estabelecidos. Para

colocar-se um ponto de parada, basta fornecer o endereço, do mesmo, ao comando.

Como opção pode-se fornecer um contador, precedido por ':', que é decrementado de um, a cada vez que se atinge o ponto de parada. Caso o contador chegue a zero este é reinicializado com 1.

A cada vez que a execução alcança o ponto de parada, o processamento é suspenso e os registradores são mostrados bem como a instrução corrente é desmontada. Caso isso não seja desejado, mais sim somente quando o contador chegue a zero, deve-se utilizar a opção de repetição **R**.

Para remover um ponto de parada do programa, deve-se utilizar a opção **X** seguida do endereço do ponto de parada a ser removido. Se nenhum endereço for fornecido todos os pontos de parada são removidos do programa. Os pontos de parada são colocados na memória somente quando o programa é executado, assim não são visíveis quando o programa é listado.

Os pontos de parada são estabelecidos através de uma instrução "**RST 38H**" e como isso exige alteração no programa somente podem ser colocados em RAM. O monitor não deixa que o usuário coloque pontos de parada na EPROM.

O monitor é capaz de diferenciar os pontos de parada de uma instrução "RST 38H" montada pelo usuário. A diferença está em que o monitor nunca removerá a instrução e caso seja executada, o processamento é parado e os registradores são mostrados, porém o contador de programa apontará para a instrução seguinte ao "RST 38H".

C [N] [J] [Contador] - Pula Subrotinas

Esse comando é similar ao "TRACE" utilizado para rastreamento, porém se a instrução corrente, apontada pelo registrador PC, for uma chamada de uma subrotina, esta é executada totalmente, somente após o retorno desta o processamento é parado e os registradores, bem como a instrução corrente, são mostrados. Caso um valor seja

fornecido este é utilizado como um contador do número de instruções a serem executadas, se nenhum valor for fornecido somente uma instrução é executada.

Os registradores são mostrados a cada vez que uma instrução é executada. Isto pode ser desligado pela opção **N**, quando executa-se várias instruções. A opção **J** permite que o programa seja executado, até uma mudança de fluxo ser reconhecida, isto é a UCP tente executar uma instrução JP, JR, CALL, RST, DJNZ ou RET.

D R - "Dump" dos registradores

D [End. Inicial] [[End. Final] ¦ [S Deslocamento]] - Memória

Esse comando possui duas formas, a primeira é utilizada para mostrar os registradores do Z80-UCP, bem como a instrução apontada pelo registrador PC.

A segunda forma mostra o conteúdo da memória em hexadecimal e em ASCII, 16 bytes por linha. Se nenhum parâmetro for fornecido será mostrado o endereço corrente, cujo valor inicial é 8000H, durante 128 bytes. Pode-se fornecer o endereço inicial e o final a serem mostrados, bem como o endereço inicial e o deslocamento, através da opção **S**.

G [Endereço] [/Ponto de parada] - "Go"

Esse comando passa o controle da UCP ao programa do usuário. Caso o parâmetro **Endereço** não seja fornecido, o corrente será utilizado. Como parâmetro opcional pode-se instalar um ponto de parada temporário, que ao ser executado é removido da memória, diferente do comando **B** cujo ponto de parada permanece até ser retirado.

Antes do programa do usuário receber controle os pontos de parada são estabelecidos e a instrução corrente é examinadada para verificar-se sua validade. Caso não seja válida a instrução, o monitor não passará controle, retornando ao interpretador de comandos com um erro.

H Valor1 [Valor2] - Mostra um valor

Para se fazer contas pode-se utilizar esse commando, o qual deve ter no mínimo um parâmetro e como resultado será impresso o valor desse parâmetro em hexadecimal e em decimal. Se dois parâmetros são fornecidos a soma e a subtração dos dois, em hexadecimal e em decimal serão mostrados.

K - Comando para o disco flexível

O comando K possui vários subcomandos, destinados a testar, definir, ler ou escrever no controlador de discos flexíveis. Todos os subcomandos são de baixo nível, dando total acesso ao controlador e ao acionador de discos flexíveis. Os subcomandos de K são:

K P trilha - Posiciona em uma trilha

Este comando posiciona a cabeça do acionador de discos flexiveis na **trilha** pedida. O número da trilha depende do tipo de disco utilizado, caso seja um disco de 8" o número da trilha varia de 0 a 79 em decimal ou 0 a 4FH em hexadecimal. Caso o disco seja de 5" e 1/4, a trilha pode assumir de 0 a 39 em decimal ou de 0 a 27H em hexadecimal.

K H - "Home" - Retorna a cabeça a trilha zero

Este comando recalibra o controlador de discos, fazendo com que o acionador retorne a cabeça de leitura a trilha número zero, mais externa. O Acionador utilizado será o corrente.

K tipo densidade Tam_setor - Define o tipo de disco

Para definir-se o tipo de disco corrente deve-se utilizar este comando. O parâmetro **tipo** pode assumir os valores **8** ou **5**, representado discos de 8" ou 5" e 1/4 respectivamente. O parâmetro **densidade** pode ser **D** ou **S**, representado densidade Dupla ou Simples respectivamente, finalmente o parâmetro **Tam_setor** pode assumir os valores **0**, **1**, **2** ou **3**, onde 0 representa setores de 128 bytes, 1 setores de 256 bytes, 2 setores de 512 bytes e 3 setores de 1024

bytes. Quando o monitor é inicializado os valores iniciais são: tipo 5, densidade D, Tam_setor igual a 2.

K D acionador - Seleciona um acionador

Este comando seleciona um determinado acionador dentre os 4 possíveis como o corrente. Assim o parâmetro **acionador** deve assumir valores entre **1** e **4**. O "default" inicial é **1**.

K L lado - Seleciona o lado do disco

Este comando seleciona qual das cabeças do acionador corrente receberá ou enviará dados. O parâmetro lado pode assumir o valor 0 ou 1. A cabeça inicial a ser selecionada é a 0.

K S step - Seleciona a velocidade de acesso

A taxa com que os pulsos são enviados para o motor de passo, que controla a posição da cabeça de leitura e escrita, é controlada por esse comando. O parâmetro **step** pode assumir valores de **0** a **3**, onde representa taxas de 3, 5, 10 e 15 ms respectivamente. O valor inicial para **step** é 0 ou seja a mais alta taxa de envio de pulsos (3 ms).

K R trilha setor endereço [contador]

Este comando lê um ou mais setores do acionador e cabeça anteriormente definidos. Os parâmetros **trilha setor** e **endereço** representam a trilha, o setor inicial e o endereço onde os dados devem ser colocados. Se o parâmetro **contador** não for fornecido apenas um setor será lido, caso sejá fornecido representa o número de setores a serem lidos.

K W trilha setor endereço [contador]

Este comando é similar ao anterior **"K R"**, possuindo os mesmos parâmetros. Porém o comando ao contrário de ler, escreve um ou mais setores cujos dados devem estar no parâmetro **endereço**.

K A - "Address" - Lê o endereço corrente

Este comando é utilizado para testes no controlador, não possui qualquer parâmetro e uma vez acionado, o controlador é forçado a informar continuamente a posição atual da cabeça do acionador, entrando em laço infinito. O comando só pode ser parado com o acionamento das teclas CTRL C. O lado, trilha, setor, tamanho do setor e CRC, são mostrados continuamente na console, dessa maneira pode-se ajustar o controlar para a melhor leitura dos dados.

L [End. Inicial] [[End. Final] ; [S Deslocamento]] - Lista

O comando acima desmonta uma sequência de instruções da memória, se nenhum parâmetro for fornecido as instruções seram listadas do endereço anterior, por mais 16 bytes. O primeiro parâmetro, se fornecido, informa o endereço onde as instruções devem ser desmontadas. O segundo pode ser o endereço final ou um deslocamento em relação ao endereço inicial. A listagem pode parar e continuar, utilizando-se as teclas **CTRL S** e **CTRL Q.**

M End_Inicial End_final End_destino - Move

O comando **Move** necessita de três parâmetros, sendo que este copia os bytes do endereço inicial até o endereço final para o endereço destino. Após a copia ter sido feita, esta é comparada com a original, qualquer diferença será informada.

Q End_inicial End_final Valor [Valor] ... [Valor] - Procura

O comando **Q** procura um ou uma sequência de bytes na memória, cujo endereço inicial e final devem ser fornecidos. Se a sequência não for encontrada nada é informado, caso contrário os endereços iniciais das sequências encontradas seram mostrados.

A sequência pode ser uma cadeia de caracteres, desde que encontre-se entre apóstrofes.

R End_Inicial End_Final

Esse comando deve ser utilizado para verificar se a memória RAM está perfeita, sem problemas. A verificação é feita do endereço inicial até o endereço final, que devem ser especificados. A sequência de testes é feita, preenchendo-se a memória e testando-a com os seguintes padrões, 00H, FFH, AAH, 55H e finalmente é feito um teste de um por um dos bits. Se nenhum problema for encontrado, o comando não mostrará nenhuma mensagem, caso contrário a posição que o erro ocorreu será mostrada, juntamente com o valor escrito e o valor lido.

S IO Porto - Substitui valor em dispositivos de E/S

S Registrador - Substitui valor de um registrador

S [M] Endereco [Valor] ... [Valor] - Substitui na memória

O comando **S** de substituição possui três formas. A primeira deve ser utilizada para modificar valores em portos ou dispositivos de entrada e saída, seu argumento deve ser um endereço de um dispositivo de E/S (00H a FFH). O valor corrente do dispositivo é mostrado, após o qual um novo valor é pedido. Se não for fornecido esse valor, o porto é novamente lido e um novo valor é pedido, isso se repete até o usuário pressionar um ponto ".".

A segunda forma do comando **S** é a substituição do valor de um registrador. Qualquer registrador pode ser especificado, inclusive o registrador de "flags" e os registradores alterativos. O valor atual do registrador ou das "flags" é mostrado antes do usuário modificar o registrador. No caso das "flags", o usuário deve especificar quais devem estar ligadas.

A última forma altera valores na memória, a opção **M** deve ser fornecida quando houver conflito no reconhecimento do endereço. Por exemplo suponha que se deseje modificar o endereço BCH, porém esse parâmetro pode ser confundido pelo registrador BC, assim deve-se fornecer o comando "S M BC" ou "S 0BC".

Se o argumento **Valor** for especificado a modificação é feita, caso contrário o valor é pedido. Se for pressionada a tecla "-", o endereço será decrementado de um, caso a tecla pressionada seja o retorno de carro o endereço será incrementado de um. Para terminar o comando, deve-se pressionar a tecla ".". O argumento **Valor** pode ser uma cadeia de caracteres, sendo que cada caracter será colocado em uma posição consecutiva ao outro e o próximo endereço disponível será mostrado.

T [N] [J] [Contador] - "Trace"

Esse comando é similar ao comando **C**, porém todas as instruções são rastreadas, mesmo que a instrução seja um "CALL". Esse comando deve ser utilizado quando deseja-se verificar toda a execução do programa. Os parâmetros são identicos ao do comando **C**.

V End_Inicial End_Final End_Comp - "Verify"

Para compara-se dois blocos de bytes deve-se utilizar o comando **V**, o qual necessita de três parâmetros. O Endereço inicial e o final do primeiro bloco e o endereço inicial do segundo bloco. Se nenhuma diferença for encontrada, o comando retorna sem imprimir qualquer mensagem, caso contrário imprime o endereço do primeiro bloco, o valor contido nesse endereço, o valor contido no endereço do segundo bloco e o endereço do segundo bloco, onde houve a diferença.

Z End_Inicial End_final Valor [Valor] ... [Valor] - "Zap"

Esse comando é utilizado para preencher a memória de um endereço inicial até um endereço final, com um byte ou uma sequência de bytes. Os parâmetros **End_Inicial**, **End_Final** e um **Valor** devem ser especificados. O argumento **Valor** pode ser uma cadeia de caracteres entre apóstrofes.

4.5.1.2 Comandos Externos

Como foi visto no tópico anterior, todos os comandos do monitor começam por um simbolo ou uma letra (@, A, B, C,

D, G, H, K, L, M, Q, R, S, T, V e Z), as letras não utilizadas (E, F, I, J, N, O, P, U, W, X e Y) são disponíveis ao usuário para que o mesmo instale seus comandos, chamados de comandos externos. Todos os comandos externos tem vetores em RAM, assim o usuário, ao receber controle, pode modificar tais vetores para apontarem para suas rotinas. Todos os vetores estão em ordem alfabética na memória RAM estática. O usuário através das funções internas, como veremos a seguir, poderá obtem qualquer tipo de parâmetro que deseje para seus comandos. Durante a fase de inicialização do monitor tais vetores apontam para uma rotina de erro a qual imprime um ponto de interrogação "?", um retorno de carro e uma mudança de linha.

4.5.1.3 Funções Internas

O monitor, para poder ter uma boa flexibilidade, utiliza várias rotinas internamente. Com o objetivo de tornar tais rotinas ou funções disponíveis ao usuário, montou-se uma pseudo instrução a **"JSYS n"** ("Jump do System"). Através dessa instrução, cujo monitor monta e desmonta como qualquer outra instrução do Z80-UCP, o usuário pode dispor de uma série de funções, as quais ele teria de duplicar em seu programa de controle. Tais funções são chamadas internas pelo fato de estarem no interior do monitor. O parâmetro **n** define o número da função a ser chamada, esse número pode variar de 00 a FFH. Internamente o monitor dispõem das funções de 00 a 2FH (48 funções). A seguir são listadas as funções internas:

JSYS 00H - END

Essa função termina o programa do usuário e retorna ao monitor.

JSYS 01H - PRG_PER

Quando o usuário desejar programar periféricos poderá utilizar essa função. Antes de chamar essa função o registrador HL deve apontar para uma tabela de programação

de periféricos, quando a função retornar os registradores A, BC, HL e "Flags" estaram modificados.

A tabela de programação de periféricos deve conter um byte com o número de bytes a serem enviados ou zero no final da tabela. Deve ter ainda um byte com o valor do porto para onde os dados seram enviados e finalmente os valores. Exemplo:

```
DB      3               ; Três bytes seram enviados
DB      Porto           ; Porto de comando
DB      Valor1, Valor2, Valor3      ; Valores
DB      0               ; Fim da tabela.
```

JSYS 02H - CON_STT

Essa função retorna o estado da console, no registrador A. Somente o registrador A é alterado, retornando 0 em caso da console não ter nenhum caracter disponível ou FFH caso contrário.

JSYS 03H - CON_INP

A função CON_INP lê um caracter da console e retorna-o no registrador A, sendo que caso não haja nenhum caracter disponível, a função espera até um tornar-se disponível. Somente o registrador A é alterado.

JSYS 04H - CON_OUT

A função CON_OUT envia um caracter do registrador C para a console, somente o conteúdo do registrador A é modificado.

JSYS 05H - SER_STT

SER_STT retorna o estado do dispositivo associado ao porto auxiliar. Somente o conteúdo do registrador A é destruido, sendo que o valor retornado é 0 caso não haja caracter disponível ou FFH caso contrário.

JSYS 06H - SER_INP

A função SER_INP lê um caracter do dispositivo auxiliar se disponível, caso contrário espera até que o seja. O registrador A retorna com o caracter lido, sendo este o único registrador modificado.

JSYS 07H - SER_OUT

A função número 07H envia um caracter para o dispositivo auxiliar cujo código ASCII deve estar no registrador C. Somente o registrador A é alterado.

JSYS 08H - LIST_STT

Essa função retorna o estado da impressora no registrador A, sendo zero quando a impressora não esta disponivel e diferente de zero caso contrário. Somente o registrador A é modificado.

JSYS 09H - LIST_OUT

Essa função imprime um caracter que deve estar no registrador C, na impressora. Caso a impressora esteje desligada, desconectada ou sem papel uma mensagem de erro será enviada a console, a qual esperará pela remoção do erro ou cancelamento desta pelo usuário através de um CTRL C. Os registradores modificados são A, C.

JSYS 0AH - PUT_CAR

Essa função envia um caracter do registrador A para a console. A função PUT_CAR pode ser cancelada por um CTRL C ou bloqueada com CTRL S e CTRL Q, tendo a possibilidade ainda de utilizar o CTRL P para forçar o caracter a ser enviado também a impressora. Nenhum registrador é modificado.

JSYS 0BH - PRINT

Essa função imprime uma cadeia de caracteres terminada por zero (NULO) ou cujo o 7º bit esteja ligado. O

registrador HL deve apontar para a cadeia de caracteres a ser impressa. Os registradores A, C e HL são modificados.

JSYS OCH - PRINT_PC

Essa função é idêntica a anterior, porém a cadeia de caracteres deve estar seguindo a chamada da função. Exemplo:

```
JSYS OC              ; Chama o monitor
DB   'Cadeia de caracteres', 0
...                  ; A execução continua aqui
```

Somente os registradores A e C são modificados. Essa função pode ser cancelada por um CTRL C do usuário.

JSYS ODH - PRINT_OUT

Identica a função anterior, porém o usuário não pode cancelar a saída de caracteres. Deve ser utilizada para enviar mensagens de erro.

JSYS OEH - CR_LF

Imprime um retorno de carro e uma mudança de linha. Somente o registrador A é modificado.

JSYS OFH - UPPER_CASE

Converte um caracter ASCII que representa uma letra minúscula em uma maiúscula. O caracter deve estar no registrador A e somente este é modificado com o resultado da função.

JSYS 10H - OUT_NIB

Essa função converte um Nibble (0 a FH) em um caracter ASCII e imprime-o na console. Os registradores A e C são modificados.

JSYS 11H - OUT_A

A função converte um byte do registrador A em caracteres ASCIIs e imprime-os. Os registradores A e C são modificados.

JSYS 12H - OUT_HL

A função converte um "word" (16bits) contido nos registradores HL em caracteres ASCIIs, dando saída na console. Os registradores A e C são modificados.

JSYS 13H - OUT_DEC

OUT_DEC imprime na console o número convertido de binário para decimal em ASCII, que encontra-se no registrador HL. Somente os registradores A, C e HL são modificados.

JSYS 14H - CHK_ALPHA

Essa função verifica se o caracter contido no registrador A é válido, isto é, se ele é um número ou uma letra. Nenhum registrador é modificado, somente a "flag" CARRY é ativa se o caracter não é válido.

JSYS 15H - CVT_HEX

CVT_HEX converte um caracter ASCII contido no registrador A em um número binário hexadecimal. O registrador A retorna com o valor binário e a "flag" CARRY ligada caso o caracter não possa ser convertido.

JSYS 16H - SPACE

A função 16 imprime um espaço em branco na console, somente o registrador A é modificado.

JSYS 17H - SUM_HL_A

Essa função soma o conteúdo do registrador HL com o conteúdo do registrador A, o resultado fica no registrador HL. Somente o registrador HL é modificado.

JSYS 18H - CMP_HL_DE

A função compara o conteúdo dos registradores HL e DE, somente as "flags" são afetadas. Se HL é igual DE a "flag" ZR será ligada, caso contrário a "flag" ZR estará desligada e a "flag" CARRY indicará se HL é menor que DE

caso esteja ligada e HL menor ou igual a DE se estiver desligada.

JSYS 19H - AND_HL_DE

A função efetua uma operação "AND" de 16 bits com os registradores HL e DE, cujo resultado fica no registrador HL. Somente o registrador HL e as "flags" são afetadas.

JSYS 1AH - OR_HL_DE

A função efetua uma operação "OR" de 16 bits com os registradores HL e DE, cujo resultado fica no registrador HL. Somente o registrador HL e as "flags" são afetadas.

JSYS 1BH - MUL_HL_DE

Essa função multiplica o conteúdo dos registradores HL e DE. O resultado em 32 bits é colocado em DE HL, sendo que os 16 bits mais significativos, dos 32 bits, ficam no registrador DE e o restante no registrador HL. Somente os registradores DE e HL são afetados, juntamente com as "flags".

JSYS 1CH - DIV_HL_DE

Essa função divide o conteúdo do registrador HL pelo de DE. O resto da divisão é colocado no registrador DE e o quociente da divisão é colocado no HL. Somente os registradores DE e HL são afetados, juntamente com as "flags".

JSYS 1DH - ERRO

Imprime uma mensagem de erro, normalmente apenas um ponto de interrogação será impresso "?", juntamente com o um retorno de carro e a mudança de linha. Os registradores afetados são o A e o HL.

JSYS 1EH - GET_LINE

Essa função lê uma cadeia de caracteres terminada por retorno de carro. Os caracteres são colocados no "buffer" interno do monitor. O monitor efetua a edição na

linha caso o usuário erre sua entrada. Os comandos de edição são: O "BackSpace" e o "Delete" apagam um caracter do buffer e da console; o CTRL X apaga a linha inteira; o CTRL P liga e desliga a impressora. Todos os caracteres são convertidos para maiúsculos com exceção de cadeias de caracteres entre apóstrofes e o "TAB" é convertido para espaços em branco.

O registrador DE retorna com o endereço do "buffer", que não deve ter seu valor alterado caso o usuário deseje chamar outras funções para manipular o "buffer". Todos os valores dos registradores principais são destruidos.

JSYS 1FH - GET_INT

A função retorna um inteiro no registrador HL, obtido da cadeia de caracteres lida pela função anterior. Todas as formas de parâmetros vistas no capitulo comandos internos são válidas. O registrador DE não deve ser alterado, pois este é utilizado como ponteiro para o "buffer". Todos os registradores principais são modificados, sendo que o DE apontará para o próximo argumento no buffer.

JSYS 20H - GET_INT_DEF_END

Essa função é similar a anterior porém um valor "default" pode ser fornecido, caso o usuário tenha entrado apenas com um retorno de carro, além disso não deve existir mais nenhum caracter no buffer após a leitura do inteiro. Isto é caso a função GET_INT_DEF_END sejá chamada, deve-se ter apenas um inteiro no buffer seguido de um retorno de carro ou apenas um retorno de carro. Caso isso não ocorra um erro será impresso na console. O registrador HL retornará com o valor sendo o restante dos registradores modificados.

JSYS 21H - GET_INT2

Essa função lê dois inteiros ou um inteiro e um deslocamento. O primeiro valor retorna no registrador HL e a diferença entre o segundo e o primeiro valor, ou o valor do deslocamento, retorna no BC. Todos os registradores principais são alterados.

JSYS 22H - GET_INT3_END

Essa função lê três inteiros e suas variações, como no capítulo comandos internos, sendo que a linha deve conter no final um retorno de carro. O primeiro inteiro retorna no registrador HL, o segundo no BC e o terceiro no DE. Essa função altera todos os registradores, inclusive o DE, assim o ponteiro do buffer é perdido.

JSYS 23H - SKIP_SPACE

A função 23H incrementa o ponteiro DE até apontar para um caracter diferente de branco. Normalmente as funções anteriores à utilizam internamente. Além do registrador DE o registador A é modificado, seu conteúdo é igual ao apontado pelo DE.

JSYS 24H - SKIP_COMMA

Essa função pula brancos e vírgulas. As mesmas observações da função anterior são válidas para esta.

JSYS 25H - END_CMD

A função END_CMD pula espaços em branco e verifica se a linha de comando terminou, isto é, contém um retorno de carro. Caso negativo a função imprime uma mensagem de erro. Os registradores A e DE são alterados.

JSYS 26H - HOME

Retorna a cabeça do acionador de discos corrente para a posição zero. Essa função é similar ao comando "K H". Somente os registradores A e C são alterados, sendo que o estado do controlador é retornado no A.

JSYS 27H - CMD_DSK

CMD_DSK envia um comando contido no registrador C para o controlador de discos. O resultado do comando, estado do controlador WD2793, é retornado no registrador A, sendo este o único registrador modificado.

JSYS 28H - RDY_DSK

Essa função somente retorna ao usuário quando o controlador estiver pronto para aceitar um novo comando, somente o registrador A é modificado com o valor do estado do controlador.

JSYS 29H - SEEK_TRK

SEEK_TRK posiciona a cabeça do acionador corrente na trilha especificada, que deve ser colocada na variável TRK_DSK em RAM. Os registradores A e C são modificados, sendo o estado do controlador após o comando retornado em A.

JSYS 2AH - READ_DSK

Essa função lê um setor da unidade de disco corrente, cuja trilha deve ser posicionada pela função 29H. Todos os registradores principais são alterados, sendo o estado do controlador retornado no registrador A.

JSYS 2BH - WRITE_DSK

Essa função é similar a 2AH, porém escreve um setor na unidade seleciona. Todas as considerações dadas a função 2AH são válidas a essa.

A seguir são fornecidos as características das 6 variáveis, posicionadas na RAM estática, que controlam todos os acessos as unidades de disco.

TRK_DSK ocupa um byte, sendo utilizado pela função 29H para posicionar a unidade selecionada na trilha pedida.

SCT_DSK ocupa um byte, sendo utilizada pelas funções 2AH e 2BH para informar qual o setor que se deseja ler ou escrever respectivamente.

DMA_DSK ocupa 16 bits ("word"), que contém o endereço de onde os dados são lidos ou escritos.

STP_DSK embora ocupe um byte, deve-se colocar nessa variável apenas valores de 0 a 3, representando "step rates" de 3, 5, 10 e 15 ms respectivamente.

PAR_DSK, como a variável anterior, esta ocupar um byte porém deve-se colocar valores de 0 a 3 representando setores de 128, 256, 512 e 1024 bytes respectivamente.

TYP_DSK, essa variável ocupar um byte, porém está dividida em campos. Os bits menos significativos de 0 a 3 controlam qual a unidade de disco receberá comandos. Assim se o bit 0 estiver ligado a unidade zero estará ativa, não se deve acionar mais de uma unidade ao mesmo tempo para evitar conflitos. Se nenhum bit for ligado, nenhuma unidade será seleciona e o controlador retornará um erro de seleção. O bit 4 seleciona unidades de 8" se ativo ou 5" 1/4 caso contrário. O bit 5 controla a densidade, sendo simples se ligado e dupla caso desligado. O bit 6 controla o lado ou cabeça do acionador ativa, se desligado seleciona a cabeça 0, caso contrário a cabeça 1. O bit 7 deve ser sempre 0 pois ligará as unidades de disco, o controle desse bit é feito pelas funções do monitor.

JSYS 2CH - PROC_TAB

PROC_TAB é uma função destinada a procurar uma cadeia de caracteres em uma tabela de cadeias. Para chamar essa função o registrador DE deve apontar para a cadeia de caracteres a ser comparada e o HL para a tabela de caracteres, como resultado o registrador B retorna com o número da cadeia ou a "flag" CARRY ligada caso contrário. Para sinalizar o fim de cada cadeia, o 7 bit desta deve estar ligado e o fim da tabela deve ser sinalizado com um byte zero. Exemplo:

```
        LD HL,TAB       ; HL aponta para a tabela
        LD DE,CAD       ; DE aponta para a cadeia
        JSYS 2CH        ; Procura na tabela
        JP C,OK         ; Pula se encontra
        .               ; Caso contrário ...
        .
        .
TAB:    DC 'ABCDEF'     ; Igual a: DB 'ABCDE','F'+80h
```

```
                DC '0123456789'
                DC 'XR7$H'
                DB 0            ; O fim da tabela é indicado
                                ; por um zero
CAD:            DB 'XR7$H'      ; Caracter a ser procurado
```

Nesse exemplo, o registrador B retorna 2 e a CARRY "flag" ligada.

JSYS 2DH - RAM_TST

Essa função é idêntica ao comando 'V' do monitor, que deve receber no registrador HL o endereço inicial da memória e o número de bytes a serem verificados no registrador BC. A "flag" CARRY retorna ligada se algum erro for encontrado e HL aponta para a localização da memória em que o erro ocorreu. Todos os registadores principais são modificados, e a memória após o teste é preenchida com zero.

JSYS 2EH - CHK_STRING

Essa função compara duas cadeias de caracteres, cujo comprimento deve ser colocado no registrador B, o endereço da primeira cadeia no DE e o da segunda no HL. Se as cadeias forem idênticas a "flag" ZR retorna ligada e os registradores DE e HL apontando para o próximo elemento da cadeia, juntamente com o registrador B igual a zero. Caso as cadeias sejam diferentes, a "flag" ZR retorna desligada e os registradores DE e HL apontam para o caracter que não coincidiu em ambas as cadeias. Os registradores A, B, DE e HL são alterados.

JSYS 2FH - CHK_TAB

CHK_TAB efetua a procura de um caracter apontado por DE, em uma tabela de caracteres apontada por HL e terminadada por um zero. Caso o caracter seja encontrado o CARRY "flag" retorna ligado, caso contrário desligado. Os registradores A, B, DE e HL são afetados, sendo que o registrador B retorna a posição, em relação a tabela, do caracter achado ou o tamanho da tabela caso contrário. O

registrador DE é incrementado por um, caso o caracter apontado por ele seja encontrado e finalmente o registrador HL retorna apontando para o caracter encontrado na tabela ou o fim da tabela caso contrário. Exemplo:

```
        LD HL,TAB       ; HL aponta para a tabela
        LD DE,CAR       ; DE aponta para o caracter
        JSYS 2FH        ; Verifica na tabela
        JP C,OK         ; Pula se encontra
        .               ; Caso contrário ...
        .
TAB:    DB 'XR7$H',0    ; A tabela deve terminar com 0
CAR:    DB '$'          ; Caracter a ser procurado
```

Nesse exemplo, o registrador B retorna 3 a CARRY "flag" ligada, HL apontado para '$' na tabela e DE incrementado de um.

4.5.1.4 Funções Externas

Todas as funções entre 00 a 2FH são internas ao monitor. Qualquer chamada a funções de 30H a FFH simplesmente resultará em um retorno do monitor sem qualquer modificação nos registradores. Tais funções são destinadas a ampliação pelo usuário e dessa maneira são chamadas funções externas.

Se o usuário desejar instalar suas próprias funções, deve modificar a variável tipo "word" MORE_JSYS na memória RAM estática. Essa variável deve apontar para uma rotina, a qual analizará a variável tipo byte TMP_SP, para determinar qual o número da função chamada. Todos os registradores nesse momento, contém os valores passados durante a chamada "JSYS n", assim o usuário pode obter qualquer tipo de parâmetro desejado.

Durante a execução das funções externas pode-se chamar as funções internas, porém a variável TMP_SP será modificada. Deve-se, quando possível, modificar apenas os

registradores essenciais durante a execução das funções, salvando qualquer registrador que não retorne valor útil.

4.5.1.5 Programa de controle

Um dos objetivos do micro LIE é a possibilidade de sua utilização em laboratórios no controle de equipamentos. Tais equipamentos necessitam de um programa de controle, que o usuário pode desenvolver dotado dos recursos obtidos com o monitor. Esse programa pode ser escrito e desenvolvido na própria memória RAM do micro, a fim de que se possa seguir passo a passo sua execução. Quando tal programa estiver totalmente desenvolvido, o usuário poderá gravá-lo em uma EPROM e colocá-la no soquete destinado a EPROM do usuário.

Durante a fase de inicialização do monitor este verifica a existência da EPROM e passa o controle da UCP a mesma através de um "CALL". Essa verificação é feita analizando-se as três primeiras posições da EPROM, que deve conter a cadeia de caracteres '<M>' para ser reconhecida como válida. Se o monitor validar a EPROM, o controle passará para a posição posterior a cadeia de caracteres. Como o endereço inicial da EPROM é 2000H, o controle será passado a posição 2003H.

Tendo o controle da UCP, o programa do usuário pode instalar novos comandos, comandos externos, e novas funções, funções externas, e retornar ao monitor através de um retorno de subrotina "RET". Outra possibilidade do programa do usuário é simplesmente executar o controle dos equipamentos, utilizando as funções do monitor.

Caso o usuário necessite de contadores temporizados, o micro dispõem do terceiro contador da 8253[17] para tal fim. Esse contador é inicializado para gerar interrupções 60 vezes por segundo, embora sua interrupção esteja desligada pela Z80-SIO. O valor do contador e a interrupção podem ser modificados alterando-se os registradores do Z80-SIO e da 8253.

Todas as funções internas em caso erro imprimem, antes de retornar ao monitor, um "?", retorno de carro e mudança de linha. Caso isso não seja desejado, pode-se alterar o valor da variável de 16 bits, END_ERR na memória RAM estática, para apontar para a rotina de erro do usuário. Assim tal rotina pode imprimir qualquer mensagem e retornar ao programa do usuário.

4.5.2 Usando o CP/M

No caso do usuário necessitar de compiladores para linguagens de alto nível e utilitários, este terá ainda a possibilidade de utilizar um sistema operacional mais sofisticado, o CP/M.

Em capítulos anteriores foi visto que para poder-se utilizar tal sistema é necessário o BIOS, módulo de entrada e saída do CP/M. Esse módulo contém todas as rotinas e tabelas de manipulação dos dispositivos físicos, conectados ao microcomputador. Dessa forma o BIOS serve como interface entre a parte lógica do CP/M e a parte física da máquina, que após ser desenvolvido assume o nome de CBIOS.

4.5.2.1 CBIOS

Um dos maiores problemas no projeto do CBIOS é o espaço disponível. Para resultar em um espaço maior aos programas utilitários, área TPA, deve-se gerar o CP/M para utilizar toda a memória RAM dinâmica (64 Kbytes). Neste caso o espaço disponível ao CBIOS será de 1.5 Kbytes, com início no endereço FA00H, sendo que aproximadamente 1 Kbyte será utilizado nas tabelas, o que deixa pouco espaço (0.5 Kbytes) para se escrever todas as rotinas do CBIOS.

Uma das soluções utilizadas em alguns micro-computadores nacionais é reduzir a área TPA destinada aos utilitários, aumentado o espaço para o CBIOS, com o SYSGEN. Essa solução não é desejável, pois isto iria reduzir a performance de vários programas e alguns nem iriam funcionar.

A solução mais utilizada é a de construir o CBIOS o mais simples possível, sem mensagens de erro, rotinas pequenas e pouco eficientes e sem possibilidade de utilizar "Winchesters" devido a grande quantidade de memória destinada a área de montagem da tabela de alocação de blocos (ALV). Além disso, esses CBIOS utilizam um único tipo de acionador de disco e formato do disco, para usarem uma única tabela DPB.

Como o BDOS espera ler setores de 128 bytes, e para obter maior performace nos dispositivos de armazenamento em massa utilizam-se setores de 512 bytes, as rotinas para efetuarem a montagem e desmontagem dos setores físicos em setores lógicos ("Blocking" e "Deblocking") são extremamente simples, algumas até degradam a performance das operações em disco.

Como o objetivo era dotar o CBIOS com algorítimos eficientes e boas mensagens de erro, optou-se em utilizar o recurso do chaveamente dos bancos no CBIOS. Assim este foi projetado para conter todas as tabelas independetemente do tipo e do acionador, dando a possibilidade de utilizar discos com formatos e tipos diferentes. As rotinas do CBIOS simplesmente mudam de banco, e passam o controle a EPROM colocada no banco zero. Tal EPROM foi chamada de EDOS "Eprom Disk Operating System".

O EDOS utiliza a RAM estática como "buffer" para os algorítimos de leitura e escrita e para o armazemamento de informações referentes aos discos e dispositivos. Possui rotinas para o tratamento de erros em todos os dispositivos associados ao micro. Quando qualquer erro ocorre, o EDOS informa o tipo, o dispositivo e o motivo do erro, dando ao usuário a possibilidade de retentar ou retornar com o erro ao BDOS.

Como todas as tabelas que o CP/M necessita, estão na pequena área destinada ao CBIOS, na RAM dinâmica, não haverá assim qualquer problema quanto a mudança nos bancos.

O início do CBIOS é uma tabela de "JUMPs" para as rotinas de baixo nível contidas neste (apêndice C). Essa tabela é utilizada pelo BDOS como ponto de entrada para as rotinas do CBIOS, pois sendo este construido pelo fabricante do "Hardware", o BDOS não tem como saber o endereço de tais rotinas, a não ser por essa tabela. Essa tabela é composta por 17 "JUMPs", sendo que foi adionado mais um para a rotina "SWAP" a qual troca os bancos, dando possibilidade de ampliações do CBIOS pelo usuário.

A tabela de "JUMPs" aponta para as seguintes rotinas:

BOOT: Essa rotina é responsável pela inicialização do sistema durante a partida a frio. O carregador de partida a frio ("Cold Start Loader"), passa o controle para esta rotina, logo após a carga do CP/M. A rotina BOOT deve inicializar todos os periféricos e variáveis utilizadas pelo CP/M, ao término do qual deve passar controle para o módulo CCP do CP/M. Essa rotina é chamada apenas uma vez.

No caso do micro LIE essa rotina apenas inicializa as variáveis utilizadas no CBIOS, já que toda a tarefa principal é feita pelo EDOS quando o micro é ligado.

WBOOT: Responsável pela partida a quente ("Warm Start") do sistema. Isso ocorre quando um CTRL C é pressionado pelo operador. A responsabilidade dessa rotina é recarregar do disco "A" ou da EPROM do usuário todo o CP/M, com exceção do CBIOS, devendo ainda inicializar algumas variáveis na área de parâmetros do sistema.

CONST: Verifica o estado da console. Se houver uma tecla pressionada ou pendente, retorna com o registrador A igual a FFH, caso contrário com zero.

Por ser uma rotina extremamente pequena optou-se por deixá-la no próprio CBIOS, evitando a mudança de bancos.

CONIN: Espera um caracter estar disponivel na console e retorna o código ASCII correspondente no

registrador A. Como na rotina anterior, CONIN não chama o EDOS.

CONOUT: Envia um caracter ASCII, contido no registrador C, para a console. Não chama o EDOS.

LIST: Envia um caracter ASCII recebido no registrador C para a impressora. Chama o EDOS em caso da impressora não estar disponivel para impressão, deixando para o mesmo a análise do problema e sua correção.

PUNCH: Envia um caracter ASCII recebido no registrador C para o dispositivo físico associado ao dispositivo lógico **PUNCH**. Essa rotina não chama o EDOS.

READER: Lê um caracter do dispositivo fisico associado ao lógico **READER** e retorna o código ASCII no registrador A. Não chama o EDOS.

HOME: Posiciona a cabeça de leitura e escrita do drive selecionado na trilha zero. Essa rotina, devido aos algoritimos de "Blocking" e "Deblocking", apenas informa ao EDOS a nova trilha, deixando ao mesmo a responsabilidade do posicionamento.

SELDSK: Seleciona o acionador especificado pelo registrador C para as próximas operações do disco. O valor passado em C varia de 0 para o acionador A, a 15 para o P. SELDSK deve retornar em HL o endereço do DPH associado ao acionador. Caso este não exista ou esteja inoperante o valor retornado em HL será 0000H.

A rotina do micro LIE simplesmente chama o EDOS para selecionar o disco e verificar sua validade. Caso o EDOS retorne com o disco válido, a rotina obterá o endereço do DPH correspondente e passará ao BDOS.

O EDOS é capaz de reconhecer vários formatos e tipos de acionador, bem como um disco virtual e um "Winchester". Após o reconhecimento ele preenche a área de parâmetros no CBIOS com as informações obtidas pelo mesmo.

SETTRK: Posiciona a cabeça de leitura e escrita do acionador corrente na trilha indicada pelo registrador BC. SETTRK não tem, obrigatoriamente, que mover a cabeça, podendo apenas atualizar uma variável, deixando a tarefa para as rotinas READ e WRITE, isto é feito pelo CBIOS do micro LIE, informando apenas o EDOS da trilha desejada.

SETSEC: Informa através do registrador BC, o número do setor a ser lido ou escrito. Assim como SETTRK, o CBIOS apenas informa ao EDOS o setor onde a operação irá ocorrer.

SETDMA: Esta rotina recebe em BC o endereço inicial de 128 bytes a serem utilizados nas próximas operações de leitura e escrita. Durante a inicialização esse endereço é estabelecido em 0080H.

READ: Lê um setor especificado por SETSEC para o endereço especificado por SETDMA. Ao retornar, READ deve passar 00H no registrador A se não houve erro durante a leitura, ou 01H caso contrário.

WRITE: Escreve o conteúdo do endereço espeficado por SETDMA, no setor indicado por SETSEC. O tratamento de erro é feito de mameira similar a do READ.

A tarefa de leitura e escrita é feita pelo EDOS em todas as unidades de disco física, deixando apenas a unidade de disco virtual para o CBIOS, pois devido a sua estrutura as rotinas de leitura e escrita são pequenas.

Devido aos algorítimos utilizados no EDOS, o setor pode já estar contido nos "Buffers" internos do mesmo, no caso de leitura, que simplesmente o transferirá ao endereço pedido. No caso de escrita, os dados serão armazenados internamento ao EDOS até o tamanho de um setor físico. somente após isto, os dados seram escritos fisicamento no disco. Com isso a velocidade de acesso as unidades de disco é aumentada várias vezes. O algorítimo utilizado nas rotinas de "Blocking" e "Deblocking" é o indicado pela DIGITAL RESEARCH.

Em caso de erro, o EDOS tenta no mínimo 10 vezes a operação após o qual imprime uma mensagem a qual mostra a unidade, a operação e a possível causa do erro, deixando ao usuário a possibilidade de retentar ou passar esse erro ao CBIOS.

LISTST: Verifica o estado da impressora. Se a mesma estiver em condições de receber um caracter ASCII, retorna com o registrador A igual a FFH, caso contrário com zero. Como essa rotina é pequena, ela fica contida apenas no CBIOS.

SECTRAN: Executa a translação de setores lógicos para setores físicos. SECTRAN recebe em BC o número do setor lógico e em DE o endereço dos parâmetros residentes no DPH. O número do setor transladado é retornado em HL.

Devido ao tamanho da rotina, esta fica contida somente no CBIOS, deixando a construção da tabela, durante a seleção do acionador, para o EDOS.

SWAP: Essa rotina não faz parte do CBIOS original, sendo inserida para futuras ampliações ou para possível manipulação do usuário. Essa rotina recebe no registrador A o número da função a ser desempenhada pelo EDOS e retorna o resultado no mesmo registrador.

Além dessas funções o CBIOS dispõem de rotinas apenas utilizadas durante a inicialização (BOOT), destinadas a imprimir uma mensagem e de estabelecer no CCP a execução de um programa ou arquivo em lote após a carga do sistema de forma automática.

4.5.2.2 EDOS

O EDOS ("Eprom Disk Operating System") contém todas as rotinas principais do CBIOS, juntamente com as rotinas para a montagem e desmontagem de setores, controle do disco virtual e do disco rígido, cuja listagem pode ser vista no apêndice D.

O EDOS utiliza a RAM estática como "buffer" para suas rotinas e variáveis internas.

Na Figura 27 pode-se ver o fluxo de instruções (linha continua) e de dados (linha pontilhada) durante o procedimento de inicialização. A sequência desse procedimento é representada por números, e será analizada a seguir:

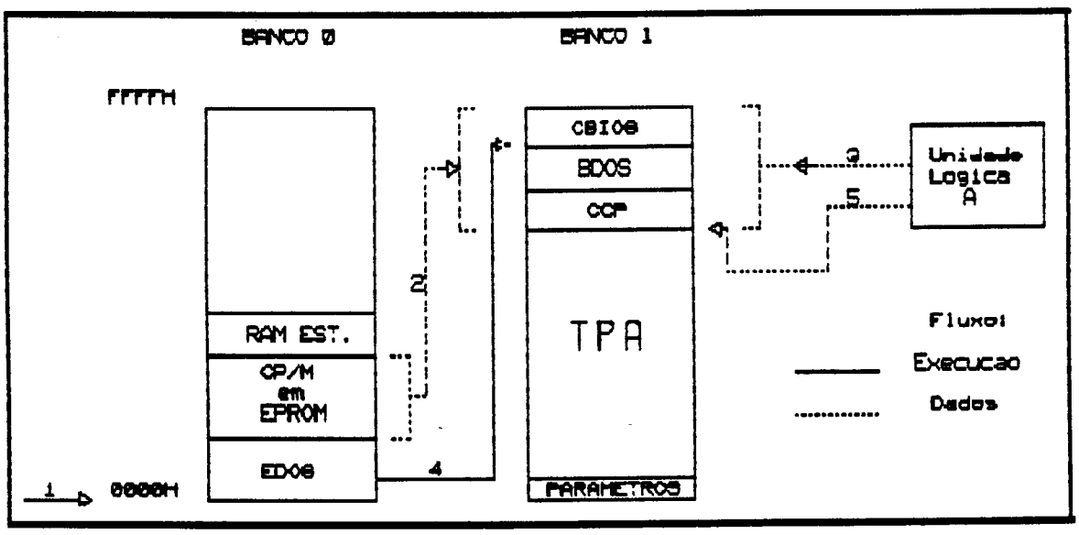


Figura 27 - Fluxo de instruções durante a inicialização.

1) Quando o micro é ligado, a rotina de inicialização do EDOS começa a ser executada, programando os periféricos existentes e verificando a RAM dinâmica. Se nenhum erro é encontrado o passo seguinte é executado, caso contrário uma mensagem de erro é enviada à console para que o operador decida o que fazer, continuar ou verificar novamente a memória.

O próximo passo é instalar a unidade de disco virtual, caso exista. Testando as memórias e inicializando a estrutura a qual o CP/M espera encontrar no disco. A seguir é verificada a existência do disco rígido, que nesse caso assumirá o papel do disco lógico A, deixando a unidade de disco flexível zero como unidade E.

2) Durante a inicialização o EDOS verifica a existência do CP/M na EPROM 1 (EPROM do usuário), copiando-o para a sua posição na RAM dinâmica.

3) A existência de um sistema operacional na unidade de disco lógico A é verificada, caso não exista sistema operacional nem na EPROM nem na unidade lógica A, uma mensagem de erro será exibida.

4) O controle é passado à rotina de "Cold Start" do CBIOS no final da rotina anterior.

5) A rotina "Cold Start" inicializa as variáveis do CP/M na área de parámetros e copia, caso exista, o comando de autoexecução da unidade lógica A para o CCP.

6) O CCP assume o controle passado pelo CBIOS, assim o CP/M passa a ser executado normalmente.

Na Figura 28 tem-se o fluxo durante uma operação de disco.

1) Quando o usuário pede, através do CCP ou de um programa aplicativo, dados ou código para uma unidade lógica, esse pedido é passado para o BDOS através da abertura de um determinado arquivo ou seleção de um disco.

2) O BDOS através das tabelas mantidas no CBIOS, calcula o setor lógico necessário ao usuário, passando-o ao CBIOS.

3) O CBIOS por sua vez transfere o pedido ao EDOS, o qual transforma o setor lógico em setor físico.

4) Caso a RAM estática não contenha o setor físico calculado pelo EDOS, este lerá da unidade física correspondente a unidade lógica selecionada, o setor calculado e transferirá o mesmo para a RAM estática.

5) Se o setor físico estiver presente na RAM estática, então os 128 bytes correspondentes ao setor lógico serão transferidos para o CP/M e o restante permanecerá na RAM estática. Assim no próximo pedido, o EDOS apenas

fornecerá os dados ao sistema e evitará o tempo de acesso ao disco.

6) O EDOS retorna ao CBIOS com informações referentes a execução do comando pedido por este.

7) O CBIOS dotado das informações referentes ao estado do comando ("Status"), transforma esse estado em informações para o BDOS.

8) Esse processo é repetido até o BDOS terminar a sua execução, retornando ao CCP ou programa aplicativo.

Através de programas aplicativos, fornecidos com o CBIOS e o EDOS, o usuário pode informar o tipo de disco, de acionador, formato dos dados, número de setores e trilhas ao EDOS para este poder converter os setores lógicos em setores físicos. Internamente o EDOS contém uma série de tipos de disco, que durante a seleção do acionador ele utilizará no caso do usuário não fazê-lo.

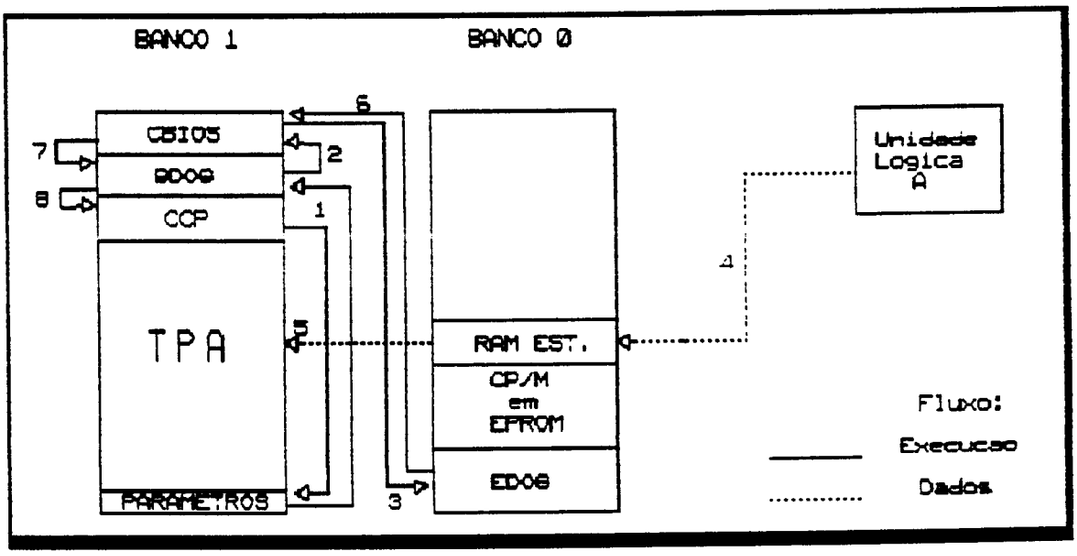


Figura 28 - Fluxo durante uma operação de disco.

Caso o CBIOS passe o controle ao EDOS para selecionar o acionador de disco rígido como o "default",

este trocará a área destinada a alocação de setores e "Check Sum", ALV e CSV, dos discos A, B, C, D e virtual, contida no CBIOS, com a área de alocação do disco rígido contida na RAM estática do EDOS. Isto é feito pois, a área ALV do disco rígido é muito grande devido a sua capacidade (10 Mbytes). Essa troca não afeta o BDOS, pois este só verifica a tabela ALV, após selecionar o disco rígido. Se o CBIOS requisitar outro disco, as tabelas são novamente trocadas entre si. O pequeno tempo gasto em transferir os dados, compensa em muito a possibilidade de utilizar discos rígidos para o armazenamento em massa de dados ou programas.

4.5.2.3 Programas aplicativos

Juntamente com o CBIOS e o EDOS, são fornecidos um conjunto de programas aplicativos, para ajudarem o desenvolvimento de outros programas ou configurarem o sistema. Um arquivo de texto "LEIA.ME" introduzido no disco que contém esses programas explica sua utilização, sintaxe e da alguns exemplos. Todos esses programas foram escritos em linguagem montadora ou em C[illegible].

A seguir será dada a explicação de alguns desses programas, juntamento com sua sintaxe:

@ Arquivo.ext

Ele executa uma série de comandos em lote do arquivo **Arquivo.ext**, com a possibilidade de passar parâmetros de qualquer forma. Esse programa é similar ao SUBMIT junto com o XSUB do CP/M, porém é menor apresenta uma performance maior e pode passar parâmatros que não poderiam ser passados com o SUBMIT e o XSUB originais.

AUTORUN Comando

Esse programa passa o comando **Comando** para o CBIOS, a fim de ser executado após o BOOT a frio do sistema. Esse comando pode ser a execução de um arquivo em lote.

O comando será colocado no disco corrente, assim somente se o disco estiver no acionador A durante a partida a frio, o comando será executado.

BOOT

O programa BOOT executará a reinicialização do sistema, sendo similar ao CTRL C pressionado pelo usuário. Esse comando pode ser executado dentro de um arquivo em lote.

CLS

CLS simplesmente limpa a tela da console, enviando uma sequência de caracteres ao mesmo. Não necessita de parâmatros.

CMP Arquivo1.ext Arquivo2.ext

Esse programa compara dois arquivos em ASCII, imprimindo as linhas que são diferente entre os mesmos.

CONFIG

O programa CONFIG configura as unidades de disco, podendo estabeler o tipo e o formato a ser utilizado. O comando apresenta um menu onde escolhe-se o acionador a ser configurado e uma lista de possíveis formatos. Caso o formato não se encontre na lista o usuário poderá adicionar mais formatos. O formato pedido será passado ao EDOS para sua utilização, e caso o usuário deseje será guardado no disco, para ser utilizado como "default" quando o micro for novamente acionado.

DB [Arquivo.ext]

Esse programa é o *monitor* desenvolvido para o micro LIE, com algumas modificações para utilizar o CBIOS e compilado para funcionar embaixo do CP/M. Com isso pode-se utilizar todos os recursos vistos anteriormente, mesmo com o

CP/M. Caso o argumento **Arquivo.ext** sejá fornecido, o arquivo será carregado para possível rastreamento.

DIS Arquivo.ext

O programa DIS é similar ao TYPE do CP/M, porém o arquivo **Arquivo.ext** será mostrado página a página na console, necessitando a confirmação de mudança de página pelo usuário.

EXC

Caso o operador necessite executar vários comandos em seguida e não deseje criar um arquivo em lote, poderá utilizar esse programa para receber todos os seus comandos e a seguir executa-los sequênciamente, independentemente do número de comandos.

Format Acionador:

Esse programa inicializa um disco virgem, sendo necessário, por motivo de segurança, ser fornecido o acionador onde o disco será inicializado. O programa fornecerá uma lista de possíveis formatos e tipos de disco, que correntemente são suportados. Esse programa é fornecido pois a inicialização de discos depende do controlador utilizado.

LOGOUT

Caso o microcomputador disponha de uma unidade de disco rígido, esse comando posicionará as cabeças do mesmo na última trilha disponível e colocará a UCP em "HALT". Isto é uma boa prática a ser utilizada em casos de mudança de posição do micro, para evitar problemas com o "Winchester".

MASTER Acionador:

Esse comando troca a unidade física, associada a unidade lógica A, por outra unidade física. A unidade lógica

A é na qual o CBIOS lê o sistema operacional durante uma reinicialização a quente (WBOOT). Se instalado o disco virtual ou o disco rígido, isso aumentará a velocidade de carregamento do sistema. O acionador deve ser fornecido e deverá ser associado a uma unidade física.

SETSP [CON ; AUX] Velocidade

O programa SETSP, muda a velocidade da console, CON, ou da linha auxiliar, AUX. As velocidades possíveis são 9600 ou 4800 bits por segundo.

SYSGEN

Esse programa é uma versão modificada do SYSGEN original para tranferir o sistema operacional de um disquete para outro, independentemente do seu tamanho e tipo.

TAB Arquivo1.ext Arquivo2.ext

O programa TAB, remove uma sequência de brancos e substitui os mesmos, por um tabulador. O arquivo **Arquivo1.ext** é a fonte e o destino é o arquivo **Arquivo2.ext.**

XDUMP Arquivo.ext

XDUMP é uma versão aprimorada do programa DUMP do CP/M. Ele mostra os bytes em hexadecimal e em ASCII. sendo mais rápido e ocupando um espaço menor.

4.6 EXEMPLO DE UTILIZAÇÃO

Para facilitar o entendimento do monitor. será mostrado um exemplo de utilização deste, instalando comandos e utilizando suas funções.

O programa exemplo instalará dois comandos externos. E e F, sendo o primeiro para ler um setor lógico e o segundo para escrever. Esses novos comandos transformam os discos em um conjunto de setores continuos, chamados setores lógicos.

ao invés de setores, trilhas e lados. Assim para um disco de 5" 1/4, dupla densidade com setores de 512 bytes, 40 trilhas, 9 setores por lado e 2 lados, ter-se-a 720 (40*2*9) setores lógicos.

A sintaxe dos novos comandos será a seguinte:

E SET_LOG ENDEREÇO [DISCO]

F SET_LOG ENDEREÇO [DISCO]

Para calcular a trilha, lado e setor físico, será utilizado as seguintes formulas:

Trilha = Int (Setor Lógico / (9 * 2))

Lado = Int ((Setor Lógico % 18) / 9)

Setor Físico = ((Setor Lógico % 18) % 9) + 1

Os valores permitidos para as variáveis são: Trilha entre 0 e 39, Lado entre 0 e 1 e Setor Físico entre 1 e 9.

Utilizar-se à região disponível da RAM estática (4130H), para guarda os novos comandos. As linhas a seguir são os comandos utilizados para entrar o programa e testa-lo. Embora o montador de instruções do monitor não aceite comentários no programa, estes foram introduzidos para maior compreenssão, sendo seguidos por ";".

a) Entrada do programa:

```
          A 4130              ; Monte os comandos
4130      CALL 413C           ; Comando E
          JSYS 2A             ; Leia um setor
          RET                 ; Fim do comando E
4136      CALL 413C           ; Comando F
          JSYS 2B             ; Escreva um setor
          RET                 ; Fim do comando F
413C      JSYS 1F             ; (GET_INT) Lê o primeiro
                              ;  valor o setor lógico
          PUSH HL             ; Salva o Setor lógico
```

```
         JSYS 1F              ; (GET_INT) Lê o endereço do
                              ;  "buffer"
         POP BC               ; BC = Set_Log, HL = Endereço
         LD (41B5),DE         ; Salva o ponteiro do "Buffer"
4146     RST 38               ; Utilizado para rastreamento
         LD (4006),HL         ; DMA_DSK
         LD H,B               ; HL = Setor Lógico
         LD L,C
         LD A,L               ; Setor = 0 ?
         OR H
414E     JR NZ,4152
         JSYS 1D              ; Erro
4152     LD DE,40.*9.*2.      ; 40 trilhas, 9 set. e 2 lados
         JSYS 18              ; Compara HL com DE
         JR C,415B            ; Setor < 720, ok
         JSYS 1D              ; caso contrario, Erro
415B     LD DE,9*2            ; 9 setores, 2 lados
415E     JSYS 1C              ; HL / DE
         LD A,L
4166     LD (4004),A          ; TRK_DSK = int( Setor / 18 )
         EX DE,HL
         LD DE,9              ; 9 Setores
         JSYS 1C              ; HL / DE
         LD A,E
         INC A                ; Soma um e armazena em
416C     LD (4005),A          ; SCT_TRK
         LD A,L
         AND 1
         LD B,6
4174     RLCA                 ; Roda para a esquerda
         DJNZ 4174            ; Roda seis vezes
```

```
         PUSH AF
         LD HL,0              ; Unidade "Default" 0
         LD DE,(41B5)         ; Restaura ponteiro
         JSYS 20              ; Obtem valor c/ "default"
4181     LD A,L
         CP 4                 ; Unidades entre 0 a 3
         JR C,4188
         JSYS 1D              ; Erro
4188     OR A
         LD B,A
         LD A,1
         JR Z,4191            ; Drive 0 ?
418E     RLCA                 ; Posiciona o bit
         DJNZ 418E
4191     LD B,A
         POP AF
         OR B
4194     LD (4003),A          ; TYP_DSK
         LD A,0
         LD (4009),A          ; STP_DSK
         LD A,2
         LD (4008),A          ; PAR_DSK
         JSYS 28              ; Disco pronto
         JSYS 26              ; Recalibra
         JSYS 29              ; Posiciona na trilha
41A7     RET
41A8     LD HL,4130           ; Instala comando E
         LD (410C),HL
         LD HL,4136           ; Instala comando F
         LD (410F),HL
```

```
        RET

        .                       ; Fim da montagem
```

b) Execução e rastreamento do programa:

```
        G 41A8                  ; Instala novos comandos

        E 100. 8000             ; Lê setor lógico 100 em
                                ;  decimal, coloca os dados
                                ;  na posição 8000H e disco 0
                                ; Pelo fato da função ter um
                                ;  RST 38, o processamento irá
                                ;  parar com PC = 4147H

        DR                      ; Mostra os valores nos
                                ;  registradores HL e BC em
                                ;  hexadecimal

        H ^BC                   ; Mostra o primeiro valor em
                                ;  hexadecimal e em decimal

        D ^HL S512.             ; Mostra área onde os dados
                                ;  serão sobrepostos pelo
                                ;  setor lógico 100.

        TJ                      ; Executa até um JUMP (414E)

        TN 2                    ; Executa 2 instruções,
                                ;  mostrando os registradores
                                ;  no final, JSYS 18

        C                       ; Executa a função e retorna,
                                ;  verifique as "flags".

        G/415E                  ; Continua a função, parando
                                ;  antes da divisão.
                                ; HL = 64H = 100
                                ; DE = 12H = 18

        H 100./18.              ; Mostra 100 dividido por 18

        C                       ; Executa a divisão e mostra
                                ;  os registradores; HL = 5 e
                                ;  DE = A, resto da divisão

        G/416C                  ; Execute até 416C, A = 2
                                ;  setor físico, HL = 1 lado

        H [100.%18.]/9          ; Mostra se está correto

        BR 4174:4               ; Estabelece um PP no "loop"
                                ;  quando passar 4 vezes

        G                       ; Executa até o Ponto de
                                ;  parada, será mostrado todas
                                ;  as vezes que o PP for
                                ;  atingido
```

```
B                       ; Verifique o contador se
                        ;  chegou a 1, isso indica que
                        ;  o mesmo foi reinicializado

L^PC                    ; Mostra o resto do programa

G/41A7                  ; Executa até o fim, será
                        ;  mostrado mais um PP

S 4146                  ; Retire o RST 38, utilizado
                        ;  apenas para depuração

O                       ; Muda para NOP

.                       ; Termina a substituição

E 100. 8000             ; Execute o comando sem parar

D 8000 S512.            ; Mostra os dados lidos

KD1                     ; Seleciona a unidade 1

KL1                     ; Seleciona lado 1

K5D2                    ; 5" 1/4, Dupla densidade, 512
                        ;  bytes por setor

KR 5 2 9000             ; Lê setor físico que
                        ;  corresponde ao lógico e
                        ;  coloca em 9000H

V 8000 S512. 9000       ; Compara os setores, se tudo
                        ;  OK, não haverá diferenças

Z 8000 S512. 'ISTO EH UM TESTE'

                        ; Preenche a memoria

D 8000 S512.            ;  e mostra-a

F 100. 8000             ; Preenche o setor lógico 100
                        ;  com os novos dados

KL1                     ; Lê o setor fisico

KD1                     ;  correspondente

KR 5 2 9000             ;  e coloca em 9000H

D 9000 S512.            ; Mostra os dados lidos

V 8000 81FF 9000        ; Compara os dados
```

Esse pequeno exemplo, mostra algumas capacidades e facilidades do monitor. A base utilizada pelo monitor é a hexadecimal, sendo necessário introduzir um "." para indicar numeração em decimal.

5. RESULTADOS E CONCLUSÃO

O presente trabalho mostra a viabilidade do desenvolvimento de um microcomputador de 8 bits, baseado na Z80-UCP. O protótipo do "Hardware" foi feito inicialmente em placa padrão com conector de 100 pinos e com ligações em "Wire-Wrap". A versão final foi implementada em circuito impresso, como pode ser visto na Figura 29. Nessa placa foi colocado todo o "Hardware" básico e conectores necessários para suportar até 4 unidades de disco flexível, uma impressora e duas linhas seriais. A placa possui reguladores de tensão própria, não necessitando de fonte estabilizada para seu funcionamento. As unidades flexíveis podem ser de 8" ou 5" 1/4 e os discos permitem vários formatos de gravação.

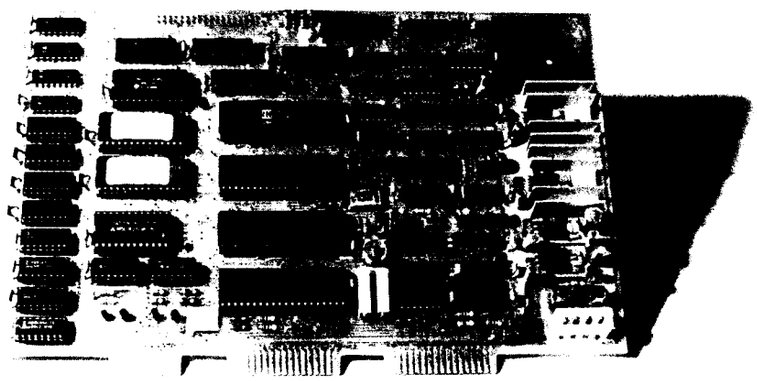
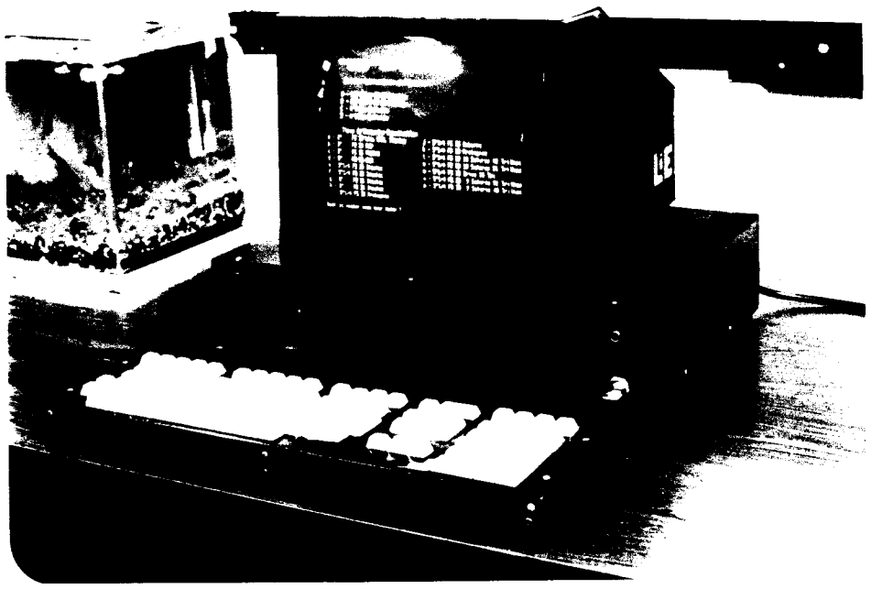


Figura 29 - Projeto Final do "Hardware" Básico

O trabalho desenvolvido não trata somente da construção do "Hardware", como pode ser visto na Figura 30 onde o sistema desenvolvido é visto com 2 acionadores

flexíveis de 5" 1/4, mais inclui tamoém o desenvolvimento de "Software" básico.

a –



b –

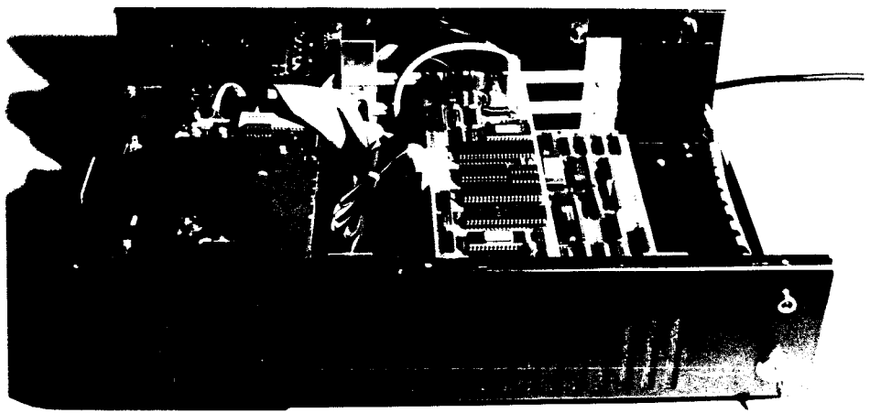


Figura 30 - Configuração utilizada no Laboratório de ensino do DFCM/USP: **a** - Vista externa; **b** - Vista interna

O monitor e depurador desenvolvido neste trabalho, mostrou-se de grande utilidade para o ensino e

desenvolvimento de sistemas de aquisição e controle, onde estas ferramentas se tornam muito valiosas.

Várias técnicas utilizadas em sistemas maiores foram implementadas neste projeto, para melhorar o desempenho e velocidade do sistema. Entre os principais estão o desenvolvimento de um disco virtual em RAM, interface para disco rígido ("Winchester") e a colocação de um coprocessador de ponto flutuante AM9511[27].

A fim de manter a memória principal disponível para aplicações, foi utilizado um complexo esquema de chaveamento de bancos de memória, acompanhado de rotinas de gerenciamento, as quais viabilizam a utilização deste esquema de uma maneira quase transparente ao usuário. Essa técnica foi utilizada também na implementação de um "Cache" para os acionadores de disco flexível e na utilização de um disco rígido, devido a grande área necessária para mapeamento dos blocos do disco (ALV).

Sua primeira aplicação foi no Laboratório de Ressonância Magnética Nuclear (RMN), para aquisição de sinais e FFT. Nesta configuração foi utilizado o coprocessador (AM9511), para aumentar a velocidade de processamento em ponto flutuante. Este equipamento permite a obtenção da FFT de uma linha espectral de NMR, quase que instantaneamente, visualizando o espectro no display do sistema de coleta. A velocidade dada por este equipamento agilizou bastante os trabalhos iniciais da construção do tomógrafo de RMN. Salientamos que foi desenvolvido também uma biblioteca de rotinas para utilização do coprocessador com a linguagem FORTRAN[28]. Na biblioteca original foram substituidos os módulos de ponto flutuante permitindo assim que qualquer programa montado com essa nova biblioteca, utilize o coprocessador de maneira transparente.

Outro projeto baseado no sistema desenvolvido foi um digitalizador e promediador de médias, desenvolvido a partir da versão básica do micro. Trata-se da contrução de um equipamento destinado a aquisição, controle e visualização descrito na dissertação de mestrado DRMN[29].

Uma aplicação em uso até o momento, são os microcomputadores colocados no Laboratório de Ensino do Departamento de Física e Ciência dos Materiais, para a aprendizagem das matérias referentes a microcomputadores. Para esta aplicação o sistema básico foi dotado de 2 acionadores de disco flexível, mostrando-se muito adequado para os cursos iniciais em computação, tanto devido a simplicidade da arquitetura interna e do duto do Z80 facilitando seu interfaceamento, quanto pelas ferramentas de "Software" desenvolvidadas neste trabalho, permitindo assim fácil acesso e manipulação do "Hardware". Lembramos que a isolação do duto externo através de "Buffers" protege o "Hardware" básico de problemas que frequentemente ocorrem em circuitos externos nos laboratórios de ensino.

Mesmo sendo o Z80-UCP um microprocessador de 8 bits, o sistema desenvolvido possui uma alta velocidade de processamento, principalmente quando utilizado com as expansões desenvolvidas, como mostrado acima.

O custo do micro pode ser considerado baixo, já que utiliza poucos CIs sendo a maior parte TTLs, todos de fácil aquisição. Além disso, dependendo da aplicação, algumas de suas partes não são necessárias e portanto não devem ser montadas, baixando ainda mais seu custo.

Todo o "Hardware" e "Software" são disponíveis ao usuário, através de esquemas, desenhos e listagens, tornando fácil efetuar qualquer tipo de modificação ou aprimoramento para uma determinada aplicação, sendo esse um dos principais objetivos dessa dissertação.

O projeto e sua documentação mostraram-se de grande utilidade, servindo como ponto de partida para várias aplicações tanto de microcomputadores de uso geral quanto de sistemas dedicados, inclusive em outras universidades.

6. REFERÊNCIAS BIBLIOGRÁFICAS

1. Dahmke, M., "Microcomputer Operationg Systems", MacGraw-Hill (1982).

2. Soares Filho, V. P., "O sistema operacional CP/M", *Interface*, 3, 38-40 (1983).

3. Liu, Y. e Gibson, G. A., "Microcomputer Systems The 8086/8088 Family Architeture, Programming and Design", 2º ed., Prentice-Hall, 1-5 (1986)

4. "MK 3880 Central Processing Unit", *MOSTEK Z80 Microcomputer Devices Technical Manual*, New England Technical Sales Corporation.

5. "MK 3881 Parallel I/O Controller", *MOSTEK Z80 Microcomputer Devices Technical Manual*, New England Technical Sales Corporation.

6. "MK 3882 Counter Timer Circuit", *MOSTEK Z80 Microcomputer Devices Technical Manual*, New England Technical Sales Corporation.

7. "MK 3884/MK 3885 Serial I/O Controller", *MOSTEK Z80 Microcomputer Devices Technical Manual*, New England Technical Sales Corporation.

8. "MK 3883 Direct Memory Access", *MOSTEK Z80 Microcomputer Systems*, Mostek Corporation (1978).

9. "MCS-80/85 Family User's Manual", Intel Incorporation, oct (1979).

10. "Cromenco CDOS Instruction Manual", Cromenco Inc. (1981).

11. Johnson-Laird, A., "The Programmer's CP/M Handbook", Osborne/MacGraw-Hill (1983).

12. Soares Filho, V. P., "O sistema operacional CP/M", *Interface*, 4, 30-3 (1983).

13. Soares Filho, V. P., "O sistema operacional CP/M", *Interface*, 5, 32-5 (1983).

14. Soares Filho, V. P., "O sistema operacional CP/M", *Interface*, 6, 23-5 (1983).

15. "An introduction to CP/M features and facilities", Digital Research (1978).

16. "The TTL Data Book for Design Engineers", Texas Instruments Inc. (1981).

17. "Peripheral Design Handbook", Intel Corporation, 1, (1981)

18. "The MOS Memory Data Book for Engineers", Texas Instruments Incorporated (1980).

19. "Products Handbook", Western Digital Corporation (1984).

20. "Manual OEM do ST412 e Controlador", Multidigit.

21. Leventhal, L. A. e Saville, W., "Z80 Assembly language subroutines", Osborne/McGrall-Hill (1983).

22. Harrell, D. M., "Operation Codes of the 8080, 8085, and Z80 Processors", *BYTE*, BYTE Publications Inc., March (1980).

23. "Microsoft, ALDS, Assembly Language Development System Reference Manual for Apple II", Microsoft Consumer Produts (1981).

24. Kernighan, B. W. e Ritchie, D. M., "The C Programming language", Prentice-Hall, Inc. (1978).

25. "AZTEC C II User Manual", Manx Software Systens (1983).

26. "LINK-80 Operator's Guide", Digital Research (1980).

27. Furht, B. e Lee, P., "An Efficient Software Driver for Am9511 Arithmetic Processor Implementation", *IEE Micro*, June (1984).

28. "Cromenco FORTRAN IV Instruction Manual", Cromenco Inc. (1982).

29. Dissertação de Mestrado apresentada ao DFCM-IFQSC/USP por Torres Neto, A. em dezembro de 1988.

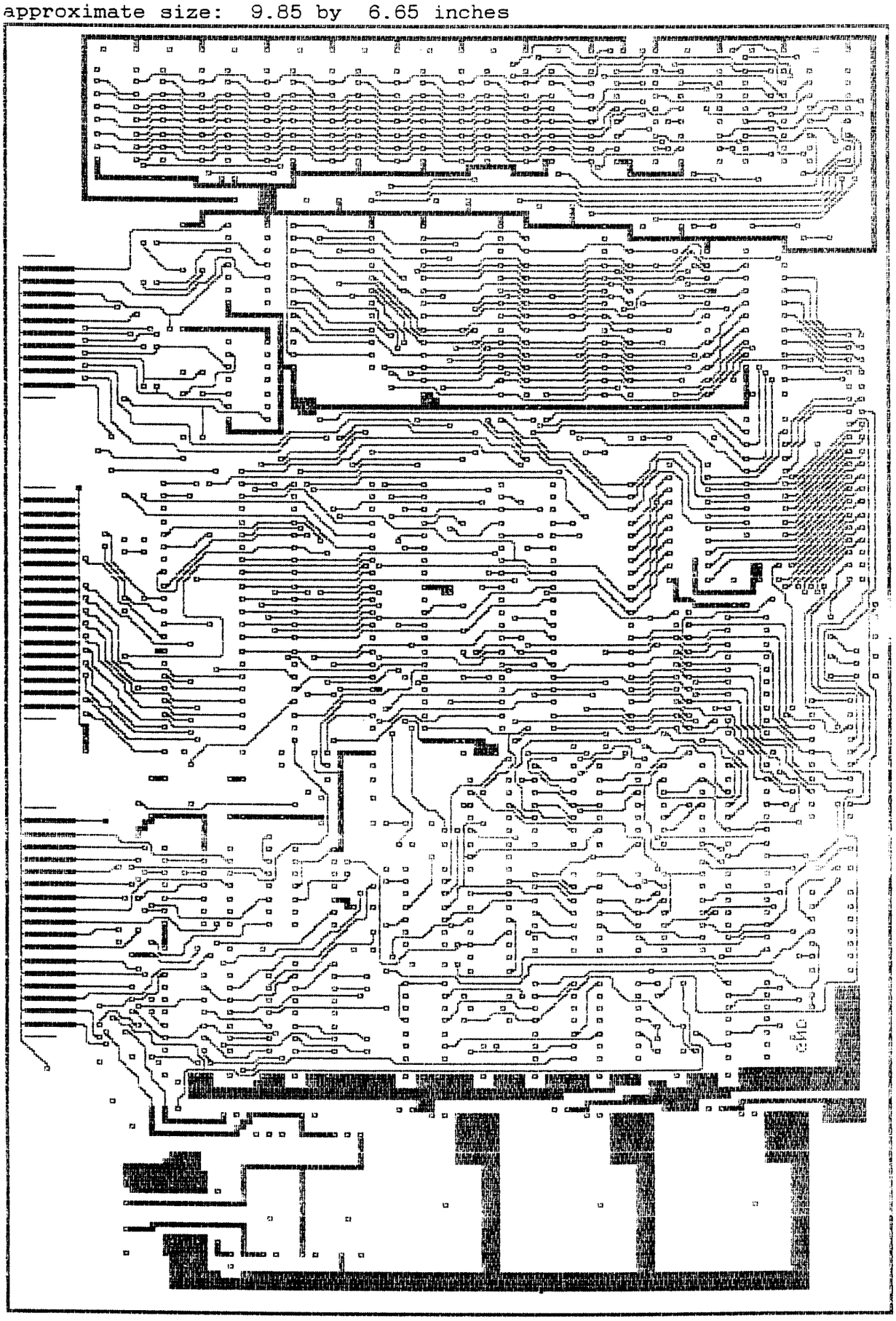
7 — APÊNDICES

APÊNDICE A

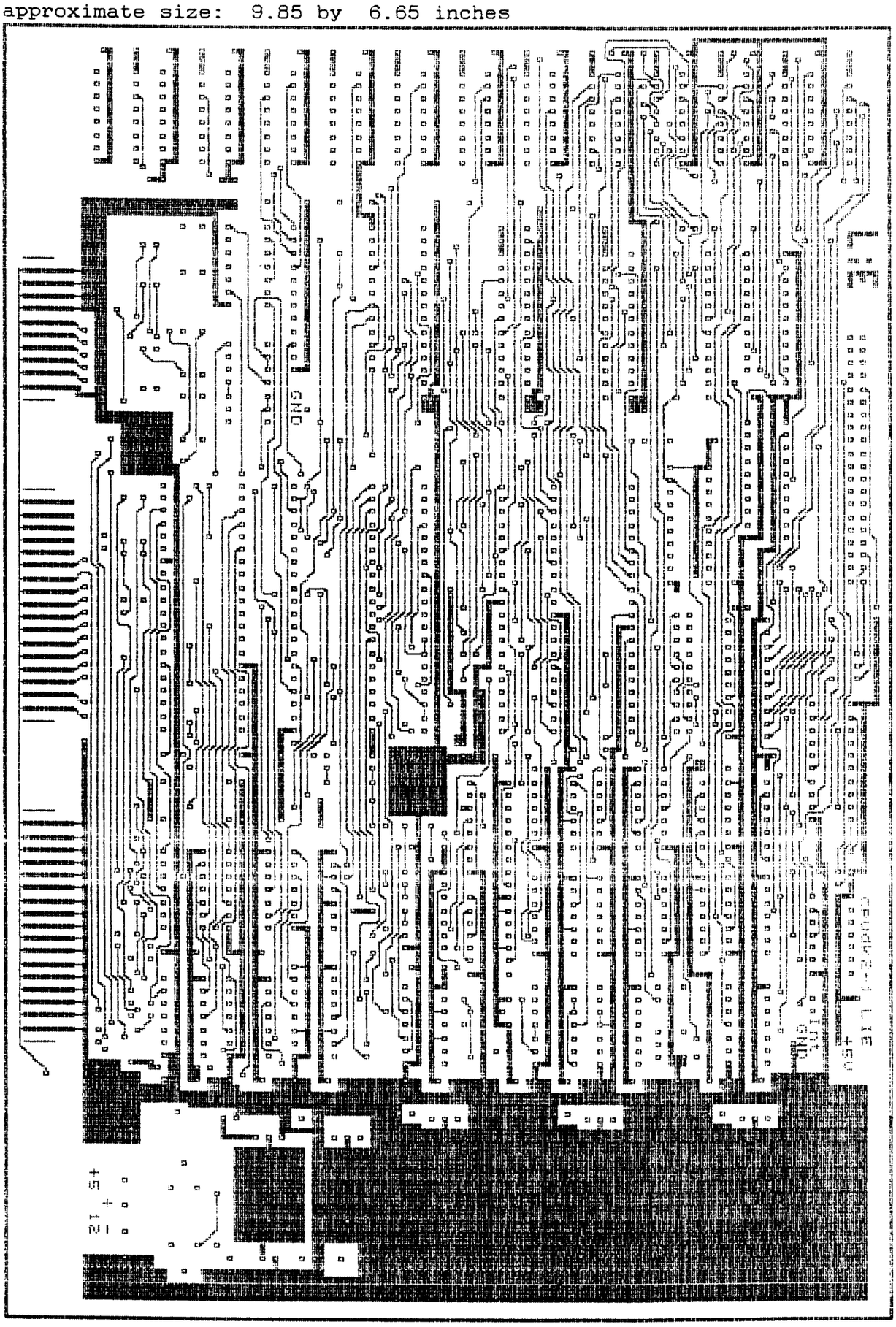
"Layout" da placa

"Hardware" Básico

1X checkplot 16 Apr 90 21:16:06
cpulie.pcb
v1.4 r1 holes: 1638 solder side
approximate size: 9.85 by 6.65 inches



1X checkplot 16 Apr 90 21:17:56
cpulie.pcb
v1.4 r1 holes: 1638 component side
approximate size: 9.85 by 6.65 inches



APÊNDICE B

Listagem do Monitor

```
                        TITLE   Monitor LIE
                ;
                ;##########################################################
                ;
                ; Monitor Expandido, Z80, versao 3.1
                ;
                ; Monitor desenvolvido no laboratorio de RMN, IFQSC-USP.
                ;
                ; Escrito por:  MATEUS J. MARTINS.
                ; Data:         8/2/88.
                ;
                ;##########################################################
                ;
                ;
                ; Definicoes
                ;
000D            CR              EQU     0DH             ; Retorno de carro
000A            LF              EQU     0AH             ; Mudanca de linha
0011            XON             EQU     11H             ; Ctrl S ( segura a tela )
0013            XOFF            EQU     13H             ; Ctrl Q ( solta a tela )
0003            EOC             EQU     3               ; Fim de comando
0010            CTRLP           EQU     10H             ; Ctrl P ( ativa/desativa impressora )
0018            CTRLX           EQU     18H             ; Ctrl X ( limpa o buffer de comando )
0008            BCK_SP          EQU     8               ; Retrocesso
007F            DELETE          EQU     7FH

0050            TAM_BUF         EQU     80              ; Tamanho do buffer de linha
4000            INIC_RAM        EQU     4000H           ; Inicio de RAM estatica banco 0
2000            INIC_EPROM      EQU     2000H           ; Inicio de EPROM do usuario banco 0
8000            POS_INIC        EQU     8000H           ; Inicio de RAM dinamica banco 1
00FF            RESTART         EQU     0FFH            ; Codigo Z80 para INT por software

                ; Comandos do controlador de Floppys
                ;
0008            CMD_HOME        EQU     8               ; Recalibra
0018            CMD_SEEK        EQU     18H             ; Procura de trilha
0080            CMD_READ        EQU     80H             ; Leitura de um setor
00A0            CMD_WRITE       EQU     0A0H            ; Escrita em um setor
00C0            CMD_ADDR        EQU     0C0H            ; Leitura de endereco
00D0            CMD_ABORT       EQU     0D0H            : Comando de cancelamento

                ; Endereco dos portos
                ;
0000            VELOC           EQU     00H             ; Porto de controle de velocidade
0002            VEL4MHZ         EQU     02H             ;       valor para 4 Mhz
0004            ROM             EQU     04H             ; comando para ativar ROM
0006            RAM             EQU     06H             ; comando para ativar RAM
0005            INVBNC          EQU     05H             : comando para inverter a RAM
0004            NRMBNC          EQU     04H             ; comando para restaurar a RAM
0017            C8255           EQU     17H             ; Porto de comando 8255
000C            D8253           EQU     0CH             ; Porto de dados 8253
000F            S8253           EQU     0FH             ; Porto de status 8253
0010            SIO_A_DAT       EQU     10H             ; Porto de dados SIO A
0011            SIO_A_STT       EQU     11H             ; Porto de status SIO A
0012            SIO_B_DAT       EQU     12H             ; Porto de dados SIO B
0013            SIO_B_STT       EQU     13H             ; Porto de status SIO B
0014            DSK_TYP         EQU     14H             ; Porto de controle do floppy
0018            DSK_STT         EQU     18H             ; Porto de status       "
0018            DSK_CMD         EQU     18H             ; Porto de comando      "
0019            DSK_TRK         EQU     19H             ; Porto de trilha       "
```

```
 001A                DSK_SCT      EQU     1AH          ; Porto de setor       "
 001B                DSK_DAT      EQU     1BH          ; Porto de dados       "
 0016                LST_STT      EQU     16H          ; Porto de status da impressora
 0015                LST          EQU     15H          ; Porto de dados       "
 0016                TIME_OUT     EQU     16H          ; Porto do circuito de temporizacao
 0020                BIT_TIME     EQU     20H          ; Bit        "        "

                    ;<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
                    ;          MACROS utilizadas como pseudo instrucoes                 ;
                    ;<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

                                 .SALL
                    +SKIP_WHITE  MACRO                 ; Pula brancos na linha de comando
                    +            RST     10H
                                 ENDM

                    +GET_INT     MACRO                 ; Obtem um inteiro
                    +            RST     18H
                                 ENDM

                    +OUT_HL      MACRO                 ; Imprime o conteudo de HL em ASCII
                    +            RST     20H
                                 ENDM

                    +OUTPUT      MACRO                 ; Imprime o conteudo do Acumulador
                    +            RST     28H           ;       em ASCII
                                 ENDM

                    +PRINT       MACRO                 ; Imprime uma cadeia de caracteres
                    +            RST     30H
                                 ENDM

                    ;
                    ;            Inicialize todo arquivo com FF
                    ;
                                 .SALL
                                 ORG     0
                    +            REPT    1000H         ; Preencha 8Kbytes com FFH
                    +            DEFW    0FFFFH
                    +            ENDM


                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                    ;            Inicio da EPROM do monitor                              ;
                    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

                                 ORG     0
 0000'   F3                      DI                    ; Inibe interrupcoes
 0001'   ED 56                   IM      1             ; estabelece modo 1 de interrupcao
 0003'   C3 0256'                JP      INIT          ; pula para a rotina de inicializacao

                                 ORG     8H            ; Vetor 8 utilizado como pseudo instrucao
 0008'   C3 0071'                JP      ROT_JSYS      ; do Z80, para chamar o sistema

                    ;      Vetores de interrupcao por software
                    ;      Aproveita-se o espaco entre os vetores para colocar algumas cadeias
                    ;      de caracteres.

 000B'   3C 4D 3E   PRONT:       DC      '<M> '        ; Cadeia de caracteres utilizada como
 000E'   A0
                                                       ; 'pronto' do monitor
```

```
                                ORG     10H             ; Subrotina para pular brancos
0010'  C3 042D'                 JP      SKIP_WHITE

0013'  3D 3C 3E     EQ_NE:      DEFM    '=<>',0         ; simbolos para igual e diferente
0016'  00

                                ORG     18H             ; Subrotina para obter um inteiro
0018'  C3 0526'                 JP      GET_INT

001B'  42 50 3D     MSG_BP:     DC      'BP= '          ; Simbolo para Break Point
001E'  A0

                                ORG     20H             ; Subrotina para impressao do HL
0020'  C3 03E6'                 JP      OUT_HL

0023'  3F 3F BF     NOT_CMD:    DC      '???'           ; Simbolo para comando nao valido

                                ORG     28H             ; Subrotina para impressao do Acumulador
0028'  C3 040B'                 JP      OUTPUT

002B'  C0           SYM_@:      DC      '@'             ; Simbolo utilizado no comando de relocacao

                                ORG     30H             ; Impressao de uma cadeia de caracteres
0030'  C3 041C'                 JP      PRINT

                                ORG     38H             ; RST utilizada como BreakPoint
0038'  C3 0C1A'                 JP      BREAK

                    ;===============================================================================
                    ;       Inicio das subrotinas de Entrada e Saida (I/O)                  ;
                    ;===============================================================================

003B'  DB 11        CON_STT:    IN      A,(SIO_A_STT)   ; Verifica se a SIO A ( utilizada como
003D'  0F           CON_STT1:   RRCA                    ;       console tem caracteres disponi-
003E'  9F                       SBC     A,A             ;       veis para a leitura
003F'  C9                       RET

0040'  CD 003B'     GET_CAR:    CALL    CON_STT         ; Le um caracter do console
0043'  28 FB                   JR      Z,GET_CAR
0045'  DB 10                    IN      A,(SIO_A_DAT)
0047'  C9                       RET

0048'  DB 11        OUT_CAR:    IN      A,(SIO_A_STT)   ; Escreve um caracter no console
004A'  E6 04                    AND     4
004C'  28 FA                    JR      Z,OUT_CAR
004E'  79                       LD      A,C
004F'  D3 10                    OUT     (SIO_A_DAT),A
0051'  C9                       RET

0052'  DB 13        SER_STT:    IN      A,(SIO_B_STT)   ; Verifica se a SIO B ( utilizada como
0054'  18 E7                    JR      CON_STT1        ;       dispositivo auxiliar, chamada
                                                        ;       apenas de serial ) tem dados

0056'  CD 0052'     SER_INP:    CALL    SER_STT         ; Le um caracter da serial
0059'  28 FB                    JR      Z,SER_INP
005B'  DB 12                    IN      A,(SIO_B_DAT)
005D'  C9                       RET

005E'  2B 2D 2A     TAB_OPR:    DEFM    '+-*/%&|', 0    ; Simbolos de operadores
0061'  2F 25 26
0064'  7C 00
```

```
                                ORG     66H             ; Vetor de interrupcao por Hardware
0066'  C9                       RET                     ; modo 1, utilizado como Pseudo DMA

0067'  DB 13            SER_OUT:IN      A,(SIO_B_STT)   ; Escreve um caracter na serial
0069'  E6 04                    AND     4
006B'  28 FA                    JR      Z,SER_OUT
006D'  79                       LD      A,C
006E'  D3 12                    OUT     (SIO_B_DAT),A
0070'  C9                       RET

0071'  E3               ROT_JSYS:EX     (SP),HL         ; Rotina JSYS ( JUMP to SYSTEM )
0072'  F5                       PUSH    AF
0073'  7E                       LD      A,(HL)          ; Numero da funcao
0074'  23                       INC     HL
0075'  32 400B                  LD      (TMP_SP),A
0078'  F1                       POP     AF
0079'  E3                       EX      (SP),HL
007A'  E5                       PUSH    HL
007B'  D5                       PUSH    DE
007C'  F5                       PUSH    AF
007D'  3A 400B                  LD      A,(TMP_SP)      ; compara com o numero maximo de
0080'  FE 30                    CP      MAX_JSYS        ; funcoes disponiveis
0082'  38 02                    JR      C,JSYS1
0084'  3E 30                    LD      A,MAX_JSYS      ; se maior utilize as funcoes definidas
                                                        ; pelo usuario
0086'  6F               JSYS1:  LD      L,A             ; use o numero da funcao chamada
0087'  26 00                    LD      H,0             ; como indice na tabela de funcoes
0089'  29                       ADD     HL,HL
008A'  11 0095'                 LD      DE,TAB_JSYS
008D'  19                       ADD     HL,DE
008E'  CD 06D2'                 CALL    LD_HL           ; Carrega em HL o endereco da funcao desejada
0091'  F1                       POP     AF
0092'  D1                       POP     DE
0093'  E3                       EX      (SP),HL
0094'  C9                       RET                     ; Executa a funcao

                        ;               TABELA da Pseudo instrucao JSYS
0095'  0296'            TAB_JSYS:DEFW   START_CR         ; Termina o programa e retorna ao monitor
0097'  00F7'                    DEFW    PRG_FT          ; Programa perifericos
0099'  003B'                    DEFW    CON_STT         ; Le status da console
009B'  0040'                    DEFW    GET_CAR         ; Le um caracter da console
009D'  0048'                    DEFW    OUT_CAR         ; Imprime um caracter na console
009F'  0052'                    DEFW    SER_STT         ; Le status da serial
00A1'  0056'                    DEFW    SER_INP         ; Le um caracter da serial
00A3'  0067'                    DEFW    SER_OUT         ; Imprime um caracter da serial
00A5'  01DF'                    DEFW    LIST_STT        ; Le status da impressora
00A7'  0164'                    DEFW    LIST            ; Imprime um caracter na impressora
00A9'  0132'                    DEFW    PUT_CAR         ; Imprime um caracter na console
00AB'  041C'                    DEFW    PRINT           ; Imprime uma cadeia de caracteres
00AD'  0415'                    DEFW    PRINT_PC        ; Imprime uma cadeia de caracteres (PC)
00AF'  01D0'                    DEFW    PRINT_OUT       ; Imprime uma cadeia de caracteres (PC)
                                                        ;       porem nao pode ser cancelada
00B1'  030A'                    DEFW    CR_LF           ; Imprime um CR e um LF
00B3'  03A9'                    DEFW    UPPER_CASE      ; Converte para letras maiusculas
00B5'  03F4'                    DEFW    OUT_NIB         ; Imprime um Nibble
00B7'  03EB'                    DEFW    OUT_A           ; Imprime o conteudo do acumulador ASCII
00B9'  03E6'                    DEFW    OUT_HL          ; Imprime o conteudo de HL em ASCII
00BB'  03CF'                    DEFW    OUT_DEC         ; Imprime um decimal
00BD'  04B1'                    DEFW    CHK_APHA        ; Verifica se e' Alphanumerico
00BF'  06FE'                    DEFW    CVT_HEX         ; Converte para Hexadecimal
00C1'  0409'                    DEFW    WHITE           ; Imprime espacos em branco
```

```
00C3' 0428'                   DEFW  SUM_HL_A       ; Soma HL com o Acumulador
00C5' 045B'                   DEFW  CMP_HL_DE      ; Comoara HL com DE
00C7' 060F'                   DEFW  HL_AND_DE      ; HL = HL and DE
00C9' 0616'                   DEFW  HL_OR_DE       ; HL = HL or DE
00CB' 05C0'                   DEFW  HL_MUL_DE      ; HL = HL * DE
00CD' 05DD'                   DEFW  HL_DIV_DE      ; HL = HL / DE
00CF' 02C8'                   DEFW  ERROR          ; Imprime simbolo de ERRO
00D1' 0331'                   DEFW  GET_LINE       ; Obtem um linha de comando da console
00D3' 0526'                   DEFW  GET_INT        ; Obtem um inteiro
00D5' 044C'                   DEFW  DEF_END        ; Obtem um inteiro com 'default'
00D7' 04E3'                   DEFW  GET_INT2       ; Obtem dois inteiros
00D9' 04D0'                   DEFW  GET_INT3_END   ; Obtem tres inteiros com fim de comando
00DB' 042D'                   DEFW  SKIP_WHITE     ; Pula Brancos
00DD' 0434'                   DEFW  SKIP_COMMA     ; Pula virgulas
00DF' 0452'                   DEFW  END_CMD        ; Verifica fim de comando
00E1' 1997'                   DEFW  KHOME          ; Recalibra os discos
00E3' 19A7'                   DEFW  CMD_DSK        ; Envia um comando ao controlador de Floppy
00E5' 19AD'                   DEFW  RDY_DSK        ; Verifica Status do controlador
00E7' 199B'                   DEFW  SEEK_TRK       ; Procura uma trilha
00E9' 1919'                   DEFW  READ_DSK       ; Le um setor
00EB' 1907'                   DEFW  WRITE_DSK      ; Escreve um setor
00ED' 0475'                   DEFW  CHKB           ; Verifica igualdade
00EF' 19D5'                   DEFW  RAMTST         ; Executa um teste na RAM
00F1' 01E7'                   DEFW  CHECK_E1       ; Verifica EPROM do usuario
00F3' 0461'                   DEFW  CHK_TAB        ; Verifica comandos em tabelas

0030            MAX_JSYS      EQU   ($-TAB_JSYS)/2 ; Numero maximo de funcoes internas
                                                   ;       do monitor

                ;       Caso usuario deseje pode aumentar o numero de funcoes
                ;               executadas por JSYS

00F5' 412C                    DEFW  MORE_JSYS      ; Expancao do JSYS


00F7' 7E        PRG_PT:       LD    A,(HL)         ; Programa perifericos
00F8' B7                      OR    A
00F9' C8                      RET   Z
00FA' 47                      LD    B,A
00FB' 23                      INC   HL
00FC' 4E                      LD    C,(HL)
00FD' 23                      INC   HL
00FE' ED B3                   OTIR
0100' 18 F5                   JR    PRG_PT

                ;           TABELA para programacao dos perifericos
                ;               existentes na placa

0102'           TAB_PER:
                        ; Inicializa a 8253 ( 3 TIMMERs )

0102' 01 0F 36                DEFB  1, 8253, 36H       ; Canal 1 => Gera Clock Serial (TTY)
0105' 02 0C 1A                DEFB  2, D8253, 26, 00   ; 2 Mhz / 26 = 4800 * 16
0108' 00
0109' 01 0F 76                DEFB  1, 8253, 76H       ; Canal 2 => Gera Clock Serial (PUN RDR)
010C' 02 0D 1A                DEFB  2, D8253+1, 26, 00 ; 2 Mhz / 26 = 4800 * 16
010F' 00
0110' 01 0F B6                DEFB  1, 8253, 0B6H      ; Canal 3 => Gera Clock p/ Relogio
0113' 02 0E 35                DEFB  2, D8253+2, 53, 130 ; 2 Mhz / 33.333 = 60 hz
0116' 82

                        ; Programa a 8255 ( Porto paralelo )
```

```
0117'  01 17 8D          DEFB   1, 08255, 10001101B ; PA saida, PB saida, PC entrada

                         ; Programa a SIO

011A'  08 11 18          DEFB   8, SIO_A_STT, 18H, 04H, 40H, 03H, 0C1H, 05H, 0EAH, 0
011D'  04 4C 03
0120'  C1 05 EA
0123'  00
0124'  08 13 18          DEFB   8, SIO_B_STT, 18H, 04H, 40H, 03H, 0C1H, 05H, 0EAH, 0
0127'  04 4C 03
012A'  C1 05 EA
012D'  00

                         ; Programa o Porto de velocidade

012E'  01 00 02          DEFB   1, VELOC, VEL4MHZ   ; Velocidade CPU = 4 Mhz

                         ; Fim da Programacao de Perifericos

0131'  00                DEFB   0

0132'  F5          PUT_CAR:      PUSH   AF                    ; Imprime um caracter na consola
0133'  C5                        PUSH   BC
0134'  E6 7F                     AND    7FH
0136'  4F                        LD     C,A
0137'  DB 11        FLUX:        IN     A,(SIO_A_STT)  ; Verifica fluxo de controle
0139'  0F                        RRCA                  ;       CTRL_S, CTRL_Q
013A'  30 12                     JR     NC,PUT_C1
013C'  DB 10                     IN     A,(SIO_A_DAT)
013E'  FE 03                     CP     EOC            ; Fim de comando (CTRL_C)
0140'  CA 0296'                  JP     Z,START_CR
0143'  FE 13                     CP     XOFF           ; CTRL_S ?
0145'  20 07                     JR     NZ,PUT_C1
0147'  CD 0040'     WAIT_XON:    CALL   GET_CAR        ; Espere um CTRL_Q
014A'  FE 11                     CP     XON
014C'  20 F9                     JR     NZ,WAIT_XON
014E'  CD 0048'     PUT_C1:      CALL   OUT_CAR        ; Tudo Ok, pode enviar o caracter
0151'  3A 400F                   LD     A,(PRT?)       ; Envia tambem a impressora ?
0154'  A7                        AND    A
0155'  C4 0164'                  CALL   NZ,LIST
0158'  30 07                     JR     NC,PUT_C2
015A'  AF                        XOR    A
015B'  32 400F                   LD     (PRT?),A
015E'  C3 0296'                  JP     START_CR
0161'  C1           PUT_C2:      POP    BC
0162'  F1                       POP    AF
0163'  C9                       RET

0164'  CD 01DF'     LIST:        CALL   LIST_STT       ; Envia um caracter `a impressora
0167'  28 05                     JR     Z,LIST1
0169'  79                        LD     A,C
016A'  D3 15                     OUT    (LST),A
016C'  A7                        AND    A
016D'  C9                        RET
016E'  DB 16        LIST1:       IN     A,(LST_STT)
0170'  CB 67                     BIT    4,A
0172'  28 37                     JR     Z,LIST3
0174'  CD 01D0'                  CALL   PRINT_OUT
0177'  0D 0A 54                  DEFM   CR, LF, 'The printer is OUT OF PAPER.', CR, LF+80H
017A'  68 65 20
017D'  70 72 69
```

```
0180'  6E 74 65
0183'  72 20 69
0186'  73 20 4F
0189'  55 54 20
018C'  4F 46 20
018F'  50 41 50
0192'  45 52 2E
0195'  0D 8A
0197'  DB 16          LIST2:      IN      A,(LST_STT)
0199'  E6 08                      AND     8
019B'  20 C7                      JR      NZ,LIST
019D'  CD 003B'                   CALL    CON_STT
01A0'  28 F5                      JR      Z,LIST2
01A2'  CD 0040'                   CALL    GET_CAR
01A5'  FE 03                      CP      EOC             ; Verifica CTRL_C
01A7'  20 EE                      JR      NZ,LIST2
01A9'  37                         SCF
01AA'  C9                         RET
01AB'  CB 6F          LIST3:      BIT     5,A
01AD'  20 B5                      JR      NZ,LIST
01AF'  CD 01D0'                   CALL    PRINT_OUT
01B2'  0D 0A 54                   DEFM    CR, LF, 'The printer is OFF LINE.', CR, LF+80H
01B5'  68 65 20
01B8'  70 72 69
01BB'  6E 74 65
01BE'  72 20 69
01C1'  73 20 4F
01C4'  46 46 20
01C7'  4C 49 4E
01CA'  45 2E 0D
01CD'  8A
01CE'  18 C7                      JR      LIST2

01D0'  E3             PRINT_OUT:  EX      (SP),HL         ; Imprime uma cadeia de caracteres
                                                          ;       terminada por ZERO ou o 7 bit
                                                          ;       ligado, cujo endereco e'
                                                          ;       apontado pelo PC
01D1'  7E             PRINT_OUT1: LD      A,(HL)          ; Obs. nao pode ser cancelado pelo
01D2'  E6 7F                      AND     7FH             ;       usuario ( CTRL_C )
01D4'  4F                         LD      C,A
01D5'  CD 0048'                   CALL    OUT_CAR
01D8'  7E                         LD      A,(HL)
01D9'  23                         INC     HL
01DA'  07                         RLCA
01DB'  30 F4                      JR      NC,PRINT_OUT1
01DD'  E3                         EX      (SP),HL
01DE'  C9                         RET

01DF'  3A 0016        LIST_STT:   LD      A,(LST_STT)     ; Verifica status da impressora
01E2'  E6 08                      AND     8
01E4'  C9                         RET

01E5'  06 03          CHECK_EPROM: LD     B,3             ; Verifica EPROM
01E7'  1A             CHECK_E1:   LD      A,(DE)
01E8'  BE                         CP      (HL)
01E9'  C0                         RET     NZ
01EA'  13                         INC     DE
01EB'  23                         INC     HL
01EC'  10 F9                      DJNZ    CHECK_E1
01EE'  C9                         RET

01EF'  E1             INIT_RAM:   POP     HL              ; Inicializa a RAM
```

```
01F0'  D9                        EXX
01F1'  01 0200                   LD      BC,STACK-INIC_RAM
01F4'  21 4000                   LD      HL,INIC_RAM
01F7'  11 4001                   LD      DE,INIC_RAM+1
01FA'  36 00                     LD      (HL),0          ; Preenche com zero
01FC'  ED B0                     LDIR
01FE'  D9                        EXX
01FF'  E5                        PUSH    HL
0200'  21 8000                   LD      HL,POS_INIC     ; Estabelece end. inicial para
0203'  22 4006                   LD      (DMA_DSK),HL    ;       transferencia do floppy
0206'  22 401B                   LD      (POSICAO),HL
0209'  22 40BC                   LD      (GO+2),HL
020C'  21 0296'                  LD      HL,START_CR     ; Inicializa User Stack Point
020F'  22 42FE                   LD      (USR_STACK-2),HL
0212'  21 42FE                   LD      HL,USR_STACK-2
0215'  22 40B8                   LD      (REG_SP),HL
0218'  21 40BA                   LD      HL,GO           ; Inicializa JP em RAM
021B'  36 F3                     LD      (HL),0F3H
021D'  23                        INC     HL
021E'  36 C3                     LD      (HL),0C3H
0220'  21 0251'                  LD      HL,ROT_EPROM    ; Transfere Subrotina para a RAM
0223'  11 40BE                   LD      DE,ROT_RAM
0226'  01 0005                   LD      BC,SIZE_ROT
0229'  ED B0                     LDIR
022B'  21 410B                   LD      HL,CMD_NOT      ; Inicializa os comandos nao existentes
022E'  36 C3                     LD      (HL),0C3H       ; na EPROM, destinados ao usuario
0230'  11 02C8'                  LD      DE,ERROR
0233'  ED 53 410C                LD      (CMD_NOT+1),DE
0237'  11 410E                   LD      DE,CMD_NOT+3
023A'  01 001E                   LD      BC,NUM_CMD_NOT-3
023D'  ED B0                     LDIR
023F'  EB                        EX      DE,HL
0240'  36 C9                     LD      (HL),0C9H       ; RET para MORE_JSYS
0242'  AF                        XOR     A
0243'  32 4009                   LD      (STP_DSK),A     ; Step rate = 3ms
0246'  3C                        INC     A
0247'  32 4003                   LD      (TYP_DSK),A     ; Seleciona acionador 1, 5"1/4, Dupla
                                                         ; densidade, lado 0
024A'  D3 14                     OUT     (DSK_TYP),A
024C'  3C                        INC     A
024D'  32 4008                   LD      (PAR_DSK),A     ; Tamhanho do setor = 512 bytes
0250'  C9                        RET

0251'  00              ROT_EPROM:NOP                     ; Subrotina a ser transferida
0252'  A7                        AND     A               ;  para a RAM
0253'  E1                        POP     HL
0254'  23                        INC     HL
0255'  E9                        JP      (HL)

0005                   SIZE_ROT  EQU     $-ROT_EPROM     ; Tamanho da rotina


                       ;*****************************************************************
                       ;               ROTINA DE INICIALIZACAO DO MONITOR              *
                       ;               a execucao do monitor inicia-se aqui            *
                       ;*****************************************************************

0256'  31 4200         INIT:     LD      SP,STACK        ; Inicializa Stack Point
0259'  21 0102'                  LD      HL,TAB_PER      ; Programa interfaces
025C'  CD 00F7'                  CALL    PRG_PT
025F'  CD 01EF'                  CALL    INIT_RAM        ; Inicializa RAM
0262'  21 2000                   LD      HL,INIC_EPROM
```

```
0265'  11 000B'                    LD      DE,PRONT
0268'  CD 01E5'                    CALL    CHECK_EPROM     ; Verifica EPROM do usuario
026B'  CC 2003                     CALL    Z,INIC_EPROM+3
026E'  CD 0415'                    CALL    PRINT_PC        ; Imprime versao do monitor
0271'  0D 0A 09                    DEFM    CR, LF, 9, 9, '<<<  Monitor Micro LIE v3.1 >>>'
0274'  09 3C 3C
0277'  3C 20 20
027A'  4D 6F 6E
027D'  69 74 6F
0280'  72 20 4D
0283'  69 63 72
0286'  6F 20 4C
0289'  49 45 20
028C'  76 33 2E
028F'  31 20 3E
0292'  3E 3E
0294'  0D 8A                       DEFB    CR ,LF+80H

0296'  CD 030A'    START_CR:       CALL    CR_LF
0299'  31 4200     START:          LD      SP,STACK        ; Novo Stack
029C'  21 0307'                    LD      HL,MSG_ERROR    ; Rotina a ser executada em caso de erro
029F'  22 4000                     LD      (END_ERR),HL
02A2'  CD 0CC8'                    CALL    RESTORE_BREAK   ; Restaura pontos de parada
02A5'  CD 0BAE'                    CALL    SLEEP_BREAK     ; e inicializa-os
02A8'  21 000B'                    LD      HL,PRONT        ; Envia pronto ao console
                  +                PRINT
02AC'  CD 0331'                    CALL    GET_LINE        ; Le comando do usuario
                  +                SKIP_WHITE              ; Pega um caracter diferente de branco
02B0'  A7                          AND     A
02B1'  28 E6                       JR      Z,START         ; Comando Nulo ?, volta
02B3'  13                          INC     DE
02B4'  D6 40                       SUB     '@'
02B6'  38 10                       JR      C,ERROR
02B8'  FE 1B                       CP      'Z'-'@'+1
02BA'  30 0C                       JR      NC,ERROR
02BC'  87                          ADD     A,A
02BD'  21 02D1'                    LD      HL,CMD_TAB      ; Verifica na tabela de comandos
02C0'  CD 042B'                    CALL    SUM_HL_A
02C3'  CD 06D2'                    CALL    LD_HL           ; Obtem o endereco do comando
02C6'  18 03                       JR      START3
02C8'  2A 4000     ERROR:          LD      HL,(END_ERR)
02CB'  CD 02D0'    START3:         CALL    JUMP            ; executa o comando
02CE'  18 C9                       JR      START

02D0'  E9          JUMP:           JP      (HL)

                   ;               TABELA contendo o enderecos dos comandos
                   ;               do monitor.
02D1'              CMD_TAB:
                  +        IRP     X,<@,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z>
                  +        DEFW    CMD_&X
                  +        ENDM

0307'  3E 3F       MSG_ERROR:      LD      A,'?'           ; Mensagem em caso de erro
                  +                OUTPUT

030A'  3E 0D       CR_LF:          LD      A,CR            ; Imprime CR e LF
                  +                OUTPUT
030D'  3E 0A                       LD      A,LF
                  +                OUTPUT
0310'  97                          SUB     A
0311'  32 4002                     LD      (CNT_CAR),A
```

```
0314'  C9                        RET

0315'  78          DELCAR:       LD      A,B           ; Apaga um caracter no buffer de linha
0316'  A7                       AND     A             ;       e na console
0317'  C8                       RET     Z
0318'  2B                       DEC     HL
0319'  05                       DEC     B
031A'  3E 08                    LD      A,BCK_SP      ; Retrocesso de carro ( Back Space )
031C'  CD 0132'                 CALL    PUT_CAR
031F'  CD 0409'                 CALL    WHITE         ; Imprime branco
0322'  3E 08                    LD      A,BCK_SP      ; Back Space
0324'  C3 0132'                 JP      PUT_CAR

0327'  77          STORE:       LD      (HL),A        ; Armazena um caracter no buffer
0328'  23                       INC     HL            ; da console
0329'  04                       INC     B
032A'  CD 0132'                 CALL    PUT_CAR
032D'  78                       LD      A,B
032E'  FE 50                    CP      TAM_BUF
0330'  C9                       RET

0331'  21 4020     GET_LINE:    LD      HL,BUF_LIN    ; Le uma linha da console
0334'  06 00                    LD      B,0
0336'  CD 0040'    GET1:        CALL    GET_CAR       ; Le um caracter
0339'  FE 03                    CP      EOC           ; CTRL_C ?
033B'  CA 0296'                 JP      Z,START_CR
033E'  FE 08                    CP      BCK_SP        ; retrocesso ?
0340'  28 04                    JR      Z,DEL_CAR
0342'  FE 7F                    CP      DELETE        ; apaga um caracter ?
0344'  20 05                    JR      NZ,GET2
0346'  CD 0315'    DEL_CAR:     CALL    DELCAR
0349'  18 EB                    JR      GET1
034B'  FE 0D        GET2:        CP      CR
034D'  28 3A                    JR      Z,GET_END
034F'  FE 10                    CP      CTRLP         ; Ativa impressora ?
0351'  20 0A                    JR      NZ,GET3
0353'  3A 400F                  LD      A,(PRT?)
0356'  EE FF                    XOR     0FFH
0358'  32 400F                  LD      (PRT?),A
035B'  18 D9                    JR      GET1
035D'  FE 18        GET3:        CP      CTRLX         ; Limpa o buffer ?
035F'  20 09                    JR      NZ,GET4
0361'  78           GET5:        LD      A,B
0362'  A7                       AND     A
0363'  28 D1                    JR      Z,GET1
0365'  CD 0315'                 CALL    DELCAR
0368'  18 F7                    JR      GET5
036A'  FE 09        GET4:        CP      9             ; Tabulacao (TAB) ?
036C'  20 0E                    JR      NZ,GET6
036E'  3E 20        GET7:        LD      A,' '
0370'  CD 0327'                 CALL    STORE
0373'  30 14                    JR      NC,GET_END
0375'  3E 07                    LD      A,7
0377'  A0                       AND     B
0378'  20 F4                    JR      NZ,GET7
037A'  18 BA                    JR      GET1
037C'  FE 20        GET6:        CP      ' '
037E'  38 B6                    JR      C,GET1
0380'  FE 7F                    CP      7FH
0382'  30 B2                    JR      NC,GET1
0384'  CD 0327'                 CALL    STORE
0387'  38 AD                    JR      C,GET1
```

```
0389'  CD 030A'     GET_END:      CALL   CR_LF          ; Fim
038C'  11 4020                    LD     DE,BUF_LIN
038F'  04                         INC    B
0390'  D5                         PUSH   DE
0391'  0E 00                      LD     C,0
0393'  1A           GET_LIN2:     LD     A,(DE)
0394'  05                         DEC    B
0395'  28 0E                      JR     Z,GET_LIN1
0397'  CB 41                      BIT    0,C
0399'  CC 03A9'                   CALL   Z,UPPER_CASE  ; Converte para letra maiuscula
039C'  12                         LD     (DE),A
039D'  13                         INC    DE
039E'  FE 27                      CP     "'"           ; cadeias de caracteres entre
03A0'  20 F1                      JR     NZ,GET_LIN2   ; apostrofes nao devem ser alteradas
03A2'  0C                         INC    C
03A3'  18 EE                      JR     GET_LIN2
03A5'  97           GET_LIN1:     SUB    A
03A6'  12                         LD     (DE),A
03A7'  D1                         POP    DE
03A8'  C9                         RET

03A9'  FE 61        UPPER_CASE:   CP     'a'           ; Converte para Maiuscula
03AB'  D8                         RET    C
03AC'  FE 7B                      CP     'z'+1
03AE'  D0                         RET    NC
03AF'  E6 5F                      AND    5FH
03B1'  C9                         RET

                    ;
03B2'               +OUT_HL@:     OUT_HL                ; Imprime o registrador HL relocado
03B3'  D5                         PUSH   DE            ;  isto e' com um deslocamento
03B4'  E5                         PUSH   HL            ;  armazenado na variavel VAR_@
03B5'  ED 5B 401E                 LD     DE,(VAR_@)
03B9'  7A                         LD     A,D           ; Deslocamento zero ?
03BA'  B3                         OR     E             ;  nao imprime nada
03BB'  28 0B                      JR     Z,OUT_HL@1
03BD'  CD 0409'                   CALL   WHITE         ; Imprime um branco a diferenca
03C0'  A7                         AND    A             ;  entre o endereco atual e o desloca-
03C1'  ED 52                      SBC    HL,DE         ;  mento e um apostrofe, para indicar
                    +             OUT_HL               ;  endereco relocado
03C4'  CD 03CB'                   CALL   OUT_ASP
03C7'  A7                         AND    A
03C8'  E1           OUT_HL@1:     POP    HL
03C9'  D1                         POP    DE
03CA'  C9                         RET

03CB'  3E 27        OUT_ASP:      LD     A,"'"         ; Imprime um apostrofe
03CD'  18 3C                      JR     OUTPUT

03CF'  CD 03D6'     OUT_DEC:      CALL   OUT_DECIMAL   ; Imprime um numero em decimal
03D2'  3E 2E                      LD     A,'.'         ; O ponto indica valor em decimal
03D4'  18 35                      JR     OUTPUT

03D6'  D5           OUT_DECIMAL:  PUSH   DE            ; Converte de binario para decimal
03D7'  11 000A                    LD     DE,10
03DA'  CD 05DD'                   CALL   HL_DIV_DE
03DD'  7C                         LD     A,H
03DE'  B5                         OR     L
03DF'  C4 03D6'                   CALL   NZ,OUT_DECIMAL ; Rotina recursiva
03E2'  7B                         LD     A,E
03E3'  D1                         POP    DE
03E4'  18 0E                      JR     OUT_NIB
```

```
03E6' 7C          OUT_HL:     LD    A,H          ; Imprime o conteudo de HL
03E7' CD 03EB'                CALL  OUT_A
03EA' 7D                      LD    A,L

03EB' F5          OUT_A:      PUSH  AF           ; Imprime o conteudo do Acumulador
03EC' 1F                      RRA
03ED' 1F                      RRA
03EE' 1F                      RRA
03EF' 1F                      RRA
03F0' CD 03F4'                CALL  OUT_NIB
03F3' F1                      POP   AF

03F4' E6 0F       OUT_NIB:    AND   0FH          ; Imprime um Nibble
03F6' FE 0A                   CP    10
03F8' 38 02                   JR    C,OUT_NIB1
03FA' C6 07                   ADD   A,7
03FC' C6 30       OUT_NIB1:   ADD   A,'0'
03FE' 18 0B                   JR    OUTPUT

0400' CD 0406'    WHITE6:     CALL  WHITE2       ; Imprime espacos em branco
0403' CD 0406'    WHITE4:     CALL  WHITE2
0406' CD 0409'    WHITE2:     CALL  WHITE
0409' 3E 20       WHITE:      LD    A,' '

040B' E5          OUTPUT:     PUSH  HL           ; Imprime um caracter
040C' CD 0132'                CALL  PUT_CAR
040F' 21 4002                 LD    HL,CNT_CAR
0412' 34                      INC   (HL)
0413' E1                      POP   HL
0414' C9                      RET

0415' E3          PRINT_PC:   EX    (SP),HL      ; Imprime uma cadeia de caracteres
                  +           PRINT              ; apontada pelo PC, porem nao pode ser
0417' 20 01                   JR    NZ,PR_PC     ; interrompida pelo usuario
0419' 23                      INC   HL
041A' E3          PR_PC:      EX    (SP),HL
041B' C9                      RET

041C' 0E 00       PRINT:      LD    C,0          ; Imprime uma cadeia de caracteres
041E' 7E          PRINT1:     LD    A,(HL)       ; terminada por NULO ou com o bit 7
041F' A7                      AND   A            ; ligado
0420' C8                      RET   Z            ; Zero ?
                  +           OUTPUT
0422' 0C                      INC   C
0423' 23                      INC   HL
0424' A7                      AND   A
0425' F8                      RET   M            ; Bit 7 ligado ?
0426' 18 F6                   JR    PRINT1

0428' 85          SUM_HL_A:   ADD   A,L          ; HL = HL + ACUMULADOR
0429' 6F                      LD    L,A
042A' D0                      RET   NC
042B' 24                      INC   H
042C' C9                      RET

042D' 1A          SKIP_WHITE: LD    A,(DE)       ; Pula brancos
042E' FE 20                   CP    ' '
0430' C0                      RET   NZ
0431' 13                      INC   DE
0432' 18 F9                   JR    SKIP_WHITE

0434'             +SKIP_COMMA: SKIP_WHITE        ; Pula virgula, se existir
```

```
0435'  FE 2C                    CP      ','
0437'  C0                       RET     NZ
0438'  13                       INC     DE
                        +       SKIP_WHITE
043A'  97                       SUB     A
043B'  C9                       RET


043C'  CD 04E3'  GET_INT2_END:  CALL    GET_INT2        ; Obtem 2 inteiros com fim de linha
043F'  18 11                    JR      END_CMD

0441'            GET_INT2_NOER_END:                     ; Obtem 2 inteiros, sem erro e
                                                        ; com fim de linha
0441'  CD 04E6'                 CALL    GET_INT2_NOER
0444'  18 0C                    JR      END_CMD

0446'            +GET_INT_END:  GET_INT                 ; Obtem um inteiro com fim de linha
0447'  DA 02CB'                 JP      C,ERROR
044A'  18 06                    JR      END_CMD

044C'  E5        DEF_END:       PUSH    HL              ; Pega um inteiro com DEFAULT
                        +       GET_INT
044E'  38 01                    JR      C,DEF_E1
0450'  E3                       EX      (SP),HL
0451'  E1        DEF_E1:        POP     HL

0452'            +END_CMD:      SKIP_WHITE              ; Verifica fim de linha
0453'  A7                       AND     A
0454'  C8                       RET     Z
0455'  FE 0D                    CP      CR
0457'  C2 02CB'                 JP      NZ,ERROR
045A'  C9                       RET

045B'  E5        CMP_HL_DE:     PUSH    HL              ; Compara HL com DE
045C'  A7                       AND     A
045D'  ED 52                    SBC     HL,DE
045F'  E1                       POP     HL
0460'  C9                       RET

0461'  06 00     CHK_TAB:       LD      B,0             ; Compara o conteudo apontado por HL
0463'  7E        CHK_T1:        LD      A,(HL)          ; com o apontado pelo DE
0464'  A7                       AND     A
0465'  C8                       RET     Z
0466'  1A                       LD      A,(DE)
0467'  BE                       CP      (HL)
0468'  28 04                    JR      Z,CHK_T2
046A'  23                       INC     HL
046B'  04                       INC     B
046C'  18 F5                    JR      CHK_T1
046E'  37        CHK_T2:        SCF
046F'  13                       INC     DE
0470'  C9                       RET

0471'  06 80     CHK7B:         LD      B,80H           ; Compara com tabela,
0473'  18 02                    JR      CHKB_1          ;  bit 7 indica se foi encontrado

0475'  06 00     CHKB:          LD      B,0             ; Compara sem o bit 7 ligado
0477'  7E        CHKB_1:        LD      A,(HL)
0478'  A7                       AND     A
0479'  C8                       RET     Z               ; Nao achou, retorna
047A'  CD 0485'                 CALL    CHKB_3          ; Procura na tabela
047D'  38 03                    JR      C,CHKB_2
```

```
047F'  04                        INC   B
0480'  18 F5                     JR    CHKB_1
0482'  CB B8          CHKB_2:    RES   7,B          ; Achou! zera bit 7
0484'  C9                        RET

0485'  C5             CHKB_3:    PUSH  BC
0486'  18 03                     JR    CHKB_5
0488'  C5             CHKB_4:    PUSH  BC
0489'  CB B8                     RES   7,B
048B'  D5             CHKB_5:    PUSH  DE
048C'  1A             CHKB_6:    LD    A,(DE)       ; Compara sem o 7 bit
048D'  AE                        XOR   (HL)
048E'  E6 7F                     AND   7FH
0490'  20 0F                     JR    NZ,CHKB_7    ; Diferente ?
0492'  CB 7E                     BIT   7,(HL)
0494'  23                        INC   HL           ; Igual, avanca proximo caracter
0495'  13                        INC   DE
0496'  28 F4                     JR    Z,CHKB_6     ; Fim?
0498'  CD 04A8'                  CALL  CHKB_9
049B'  30 07                     JR    NC,CHKB_8
049D'  F1                        POP   AF
049E'  37                        SCF
049F'  C1                        POP   BC
04A0'  C9                        RET

04A1'  CD 04C7'       CHKB_7:    CALL  END_7BUF
04A4'  D1             CHKB_8:    POP   DE
04A5'  A7                        AND   A
04A6'  C1                        POP   BC
04A7'  C9                        RET

04A8'  1A             CHKB_9:    LD    A,(DE)
04A9'  CB 78                     BIT   7,B
04AB'  28 04                     JR    Z,CHK_APHA
04AD'  FE 53                     CP    'S'
04AF'  37                        SCF
04B0'  C8                        RET   Z

04B1'  FE 30          CHK_APHA:  CP    '0'          ; Verifica se e' Alphanumerico
04B3'  D8                        RET   C
04B4'  FE 3A                     CP    '9'+1
04B6'  3F                        CCF
04B7'  D0                        RET   NC
04B8'  FE 41                     CP    'A'
04BA'  D8                        RET   C
04BB'  FE 5B                     CP    'Z'+1
04BD'  3F                        CCF
04BE'  C9                        RET

04BF'  04             LOC_TAB:   INC   B            ; Localiza em uma tabela
04C0'  05             LOC_TAB1:  DEC   B            ;  no registrador B contem o numero
04C1'  C8                        RET   Z            ;  de elementos da tabela
04C2'  CD 04C7'                  CALL  END_7BUF     ; Retorna com o endereco
04C5'  18 F9                     JR    LOC_TAB1

04C7'  7E             END_7BUF:  LD    A,(HL)       ; Localiza fim do buffer ou
04C8'  A7                        AND   A            ;  bit 7 ligado
04C9'  C8                        RET   Z
04CA'  7E             END_71:    LD    A,(HL)
04CB'  23                        INC   HL
04CC'  A7                        AND   A
04CD'  F8                        RET   M
```

```
04CE'  18 FA                    JR      END_71

04D0'  CD 04E3'    GET_INT3_END: CALL    GET_INT2        ; Obtem 3 inteiros com fim de linha
04D3'  E5                       PUSH    HL
04D4'  C5                       PUSH    BC
04D5'  CD 0434'                 CALL    SKIP_COMMA
                  +             GET_INT
04D9'  DA 02C8'                 JP      C,ERROR
04DC'  CD 0452'                 CALL    END_CMD
04DF'  EB                       EX      DE,HL
04E0'  C1                       POP     BC
04E1'  E1                       POP     HL
04E2'  C9                       RET

04E3'  37          GET_INT2:    SCF                     ; Obtem 2 inteiros
04E4'  18 01                    JR      GI2_1           ; em caso negativo pule para ERROR

04E6'  A7          GET_INT2_NOER: AND    A               ; Obtem 2 inteiros, retorna em caso
04E7'  08          GI2_1:       EX      AF,AF'          ;  de erro
04E8'  C5                       PUSH    BC
04E9'  E5                       PUSH    HL
                  +             GET_INT
04EB'  30 07                    JR      NC,GI2_2
04ED'  08                       EX      AF,AF'
04EE'  38 0D                    JR      C,GI2_4
04F0'  08                       EX      AF,AF'
04F1'  E1                       POP     HL
04F2'  18 01                    JR      GI2_3
04F4'  F1          GI2_2:       POP     AF
04F5'  CD 0502'    GI2_3:       CALL    GET_SIZE
04F8'  30 06                    JR      NC,GI2_5
04FA'  08                       EX      AF,AF'
04FB'  C1                       POP     BC
04FC'  D0                       RET     NC
04FD'  C3 02C8'    GI2_4:       JP      ERROR
0500'  F1          GI2_5:       POP     AF
0501'  C9                       RET

0502'  CD 0434'    GET_SIZE:    CALL    SKIP_COMMA      ; Verifica se ao inves de fornecer os
0505'  1A                       LD      A,(DE)          ; 2 inteiros foi fornecido 1 inteiro e o
0506'  FE 53                    CP      'S'             ; tamanho
0508'  20 01                    JR      NZ,GS_1
050A'  13                       INC     DE
050B'  E5          GS_1:        PUSH    HL
050C'  F5                       PUSH    AF
                  +             GET_INT
050E'  38 0F                    JR      C,GS_3
0510'  44                       LD      B,H
0511'  4D                       LD      C,L
0512'  F1                       POP     AF
0513'  E1                       POP     HL
0514'  28 07                    JR      Z,GS_2
0516'  79                       LD      A,C
0517'  95                       SUB     L
0518'  4F                       LD      C,A
0519'  78                       LD      A,B
051A'  9C                       SBC     A,H
051B'  47                       LD      B,A
051C'  03                       INC     BC
051D'  A7          GS_2:        AND     A
051E'  C9                       RET
051F'  F1          GS_3:        POP     AF
```

```
0520'  E1                          POP     HL
0521'  CA 02C8'                    JP      Z,ERROR
0524'  37                          SCF
0525'  C9                          RET

0526'                +GET_INT:     SKIP_WHITE              ; Obtem um inteiro
0527'  CD 0592'        GETI1:      CALL    GETI2           ; Le um valor
052A'  D8                          RET     C
052B'  CD 0541'                    CALL    CHK_SYM         ; Comparacoes ?
052E'  D0                          RET     NC
052F'  C5                          PUSH    BC
0530'  E5                          PUSH    HL
0531'  CD 0592'                    CALL    GETI2           ; Le outro valor
0534'  DA 02C8'                    JP      C,ERROR
0537'  EB                          EX      DE,HL
0538'  E3                          EX      (SP),HL
0539'  A7                          AND     A
053A'  ED 52                       SBC     HL,DE
053C'  21 FFFF                     LD      HL,-1
053F'  D1                          POP     DE
0540'  C9                          RET

0541'  E5             CHK_SYM:    PUSH    HL              ; Verifica os simbolos de comparacao
0542'  21 0013'                   LD      HL,EQ_NE        ; Simbolo de IGUAL e DIFERENTE
0545'  CD 0461'                    CALL    CHK_TAB
0548'  30 25                       JR      NC,CHK_S1       ; Nao achou, retorna
054A'  78                          LD      A,B             ; Achou, qual ?
054B'  B7                          OR      A
054C'  28 15                       JR      Z,CHK_S2        ; '=' ?
054E'  1A                          LD      A,(DE)
054F'  FE 3D                       CP      '='
0551'  20 05                       JR      NZ,CHK_S3       ; termina com '=' ?
0553'  13                          INC     DE              ; Sim, '>=' ou '<='
0554'  04                          INC     B
0555'  04                          INC     B
0556'  18 0B                       JR      CHK_S2
0558'  CB 40           CHK_S3:     BIT     0,B
055A'  28 07                       JR      Z,CHK_S2        ; '<' ?
055C'  FE 3E                       CP      '>'
055E'  20 03                       JR      NZ,CHK_S2
0560'  13                          INC     DE              ; '<>'
0561'  06 05                       LD      B,5
0563'  21 0571'        CHK_S2:     LD      HL,TAB_END_CP   ; End. das rotinas de comparacao
0566'  78                          LD      A,B
0567'  87                          ADD     A,A
0568'  CD 0428'                    CALL    SUM_HL_A
056B'  4E                          LD      C,(HL)
056C'  23                          INC     HL
056D'  46                          LD      B,(HL)
056E'  37                          SCF
056F'  E1              CHK_S1:     POP     HL
0570'  C9                          RET

0571'  057D'           TAB_END_CP: DEFW    CP_EQ           ; Tabela contendo os enderecos das
0573'  0587'                      DEFW    CP_LT           ; rotinas de comparacao
0575'  058B'                      DEFW    CP_GE
0577'  0585'                      DEFW    CP_LE
0579'  058D'                      DEFW    CP_GT
057B'  0581'                      DEFW    CP_NE

057D'  28 11           CP_EQ:      JR      Z,CP_TRUE       ; Rotinas de comparacoes;  ( = )
057F'  18 0E                       JR      CP_FALSE
```

```
0581'  20 0D        CP_NE:      JR    NZ,CP_TRUE     ;       ( <> )
0583'  18 0A                    JR    CP_FALSE
0585'  28 09        CP_LE:      JR    Z,CP_TRUE      ;       ( <= )
0587'  38 07        CP_LT:      JR    C,CP_TRUE      ;       ( <  )
0589'  18 04                    JR    CP_FALSE
058B'  28 02        CP_GE:      JR    Z,CP_FALSE     ;       ( >= )
058D'  30 01        CP_GT:      JR    NC,CP_TRUE     ;       ( >  )
058F'  23           CP_FALSE:   INC   HL             ; Caso falso hl = 0
0590'  A7           CP_TRUE:    AND   A              ; Caso verdadeido HL = -1
0591'  C9                      RET

0592'  CD 0641'     GETI2:      CALL  GET_WRD        ; Obtem um valor
0595'  D8                      RET   C
0596'  CD 061D'     GETI3:      CALL  CHK_OPR        ; Verifica Operandos (+,-,*,/,...)
0599'  E5                      PUSH  HL
059A'  C5                      PUSH  BC
059B'  CD 0643'                CALL  GET_WRD1       ; proximo valor
059E'  C1                      POP   BC
059F'  EB                      EX    DE,HL
05A0'  E3                      EX    (SP),HL
05A1'  30 07                   JR    NC,GETI4
05A3'  D1                      POP   DE
05A4'  78                      LD    A,B
05A5'  B1                      OR    C
05A6'  C8                      RET   Z
05A7'  C3 02C8'                JP    ERROR
05AA'  78           GETI4:     LD    A,B
05AB'  B1                      OR    C
05AC'  E5                      PUSH  HL
05AD'  21 05B7'                LD    HL,RET_OPR     ; Ender. para retorno
05B0'  E3                      EX    (SP),HL
05B1'  C5                      PUSH  BC
05B2'  C0                      RET   NZ
05B3'  C1                      POP   BC
05B4'  E3                      EX    (SP),HL
05B5'  E1                      POP   HL
05B6'  19                      ADD   HL,DE
05B7'  D1           RET_OPR:   POP   DE
05B8'  18 DC                   JR    GETI3

05BA'  19           HL_ADD_DE: ADD   HL,DE          ; HL = HL + DE
05BB'  C9                      RET

05BC'  A7           HL_SUB_DE: AND   A              ; HL = HL - DE
05BD'  ED 52                   SBC   HL,DE
05BF'  C9                      RET

05C0'  F5           HL_MUL_DE: PUSH  AF             ; DE HL = HL * DE
05C1'  C5                      PUSH  BC
05C2'  44                      LD    B,H
05C3'  4D                      LD    C,L
05C4'  3E 10                   LD    A,16
05C6'  21 0000                 LD    HL,0
05C9'  29           MLT1:      ADD   HL,HL
05CA'  CB 13                   RL    E
05CC'  CB 12                   RL    D
05CE'  30 07                   JR    NC,MLT2
05D0'  09                      ADD   HL,BC
05D1'  30 04                   JR    NC,MLT2
05D3'  1C                      INC   E
05D4'  20 01                   JR    NZ,MLT2
05D6'  14                      INC   D
```

```
05D7' 3D          MLT2:     DEC   A
05D8' 20 EF               JR    NZ,MLT1
05DA' C1                  POP   BC
05DB' F1                  POP   AF
05DC' C9                  RET

05DD' 7B          HL_DIV_DE: LD   A,E          ; DIVISAO ENTRADA: HL = DIVIDENDO
05DE' B2                  OR    D             ;                  DE = DIVISOR
05DF' 20 07               JR    NZ,DIVD1      ;         SAIDA:   DE = RESTO
05E1' 21 0000             LD    HL,0          ;                  HL = COCIENTE
05E4' 54                  LD    D,H           ; CARRY = 1 EM CASO DE DIV. POR ZERO
05E5' 5D                  LD    E,L
05E6' 37                  SCF
05E7' C9                  RET
05E8' 4D          DIVD1:  LD    C,L
05E9' 7C                  LD    A,H
05EA' 21 0000             LD    HL,0
05ED' 06 10               LD    B,16
05EF' B7                  OR    A
05F0' CB 11       DVLOOP: RL    C
05F2' 17                  RLA
05F3' CB 15               RL    L
05F5' CB 14               RL    H
05F7' E5                  PUSH  HL
05F8' ED 52               SBC   HL,DE
05FA' 3F                  CCF
05FB' 38 01               JR    C,DROP
05FD' E3                  EX    (SP),HL
05FE' 33          DROP:   INC   SP
05FF' 33                  INC   SP
0600' 10 EE               DJNZ  DVLOOP
0602' EB                  EX    DE,HL
0603' CB 11               RL    C
0605' 69                  LD    L,C
0606' 17                  RLA
0607' 67                  LD    H,A
0608' B7                  OR    A
0609' C9                  RET

060A' CD 05DD'    HL_MOD_DE: CALL HL_DIV_DE
060D' EB                  EX    DE,HL
060E' C9                  RET

060F' 7C          HL_AND_DE: LD   A,H          ; HL = HL and DE
0610' A2                  AND   D
0611' 67                  LD    H,A
0612' 7D                  LD    A,L
0613' A3                  AND   E
0614' 6F                  LD    L,A
0615' C9                  RET

0616' 7C          HL_OR_DE: LD    A,H          ; HL = HL or DE
0617' B2                  OR    D
0618' 67                  LD    H,A
0619' 7D                  LD    A,L
061A' B3                  OR    E
061B' 6F                  LD    L,A
061C' C9                  RET

061D' E5          CHK_OPR:  PUSH  HL           ; Verifica operacao a ser efetuada
061E' 21 005E'            LD    HL,TAB_OPR     ; Tabela de operandos
0621' CD 0461'            CALL  CHK_TAB
```

```
0624'  78                        LD      A,B
0625'  21 0631'                  LD      HL,OPR_ROT      ; Tabela das rotinas
0628'  87                        ADD     A,A
0629'  CD 0428'                  CALL    SUM_HL_A
062C'  4E                        LD      C,(HL)
062D'  23                        INC     HL
062E'  46                        LD      B,(HL)
062F'  E1                        POP     HL
0630'  C9                        RET

                    ;            TABELA com os  enderecos das rotinas

0631'  05BA' 05BC'  OPR_ROT:     DEFW    HL_ADD_DE, HL_SUB_DE, HL_MUL_DE, HL_DIV_DE
0635'  05C0' 05DD'
0639'  060A' 060F'               DEFW    HL_MOD_DE, HL_AND_DE, HL_OR_DE, 0
063D'  0616' 0000

0641'  0E 01        GET_WRD:     LD       C,1            ; Obtem um numero
0643'  CD 06FD'     GET_WRD1:    CALL    ASC_BIN         ; Ascii -> binario
0646'  30 2D                     JR      NC,W_HEXDEC
0648'  1A                        LD      A,(DE)
0649'  13                        INC     DE
064A'  2A 401E                   LD      HL,(VAR_@)      ; Offset
064D'  FE 40                     CP      '@'
064F'  C8                        RET     Z
0650'  2A 40BC                   LD      HL,(REG_PC)     ; Ponto corrente ?
0653'  FE 24                     CP      '$'
0655'  C8                        RET     Z
0656'  FE 2D                     CP      '-'             ; Numero negativo ?
0658'  28 57                     JR      Z,N_NEG
065A'  FE 7E                     CP      '~'             ; Complemento de 1 ?
065C'  28 60                     JR      Z,N_CP1
065E'  FE 2B                     CP      '+'             ; Numero positivo ( Default ) ?
0660'  28 DF                     JR      Z,GET_WRD
0662'  FE 5E                     CP      '^'             ; Registrador ?
0664'  28 3C                     JR      Z,N_REG
0666'  FE 28                     CP      '('             ; Conteudo ?
0668'  28 59                     JR      Z,N_CNTD
066A'  FE 5B                     CP      '['             ; Outra operacao ?
066C'  28 69                     JR      Z,N_PUSH
066E'  FE 27                     CP      "'"             ; Valores em ASCII
0670'  28 19                     JR      Z,N_ASC
0672'  1B           GET_WRD2:    DEC     DE
0673'  37                        SCF
0674'  C9                        RET

0675'  D5           W_HEXDEC:    PUSH    DE              ; Obtem um word em HEXA ou em DECIMAL
0676'  13           W_HD1:       INC     DE
0677'  CD 06FD'                  CALL    ASC_BIN         ; Ascii para binario (HEXA)
067A'  30 FA                     JR      NC,W_HD1
067C'  D1                        POP     DE
067D'  FE 2E                     CP      '.'             ; Decimal ?
067F'  28 05                     JR      Z,W_HD2
0681'  CD 06E6'                  CALL    W_HEXA
0684'  A7                        AND     A
0685'  C9                        RET
0686'  CD 0711'     W_HD2:       CALL    W_DEC
0689'  A7                        AND     A
068A'  C9                        RET

068B'  21 0000      N_ASC:       LD      HL,0            ; cadeia de caracteres
068E'  1A           N_ASC1:      LD      A,(DE)
```

```
068F'  FE 27                    CP      "'"
0691'  28 07                    JR      Z,N_ASC3
0693'  A7                       AND     A
0694'  C8                       RET     Z
0695'  65           N_ASC2:    LD      H,L
0696'  6F                      LD      L,A
0697'  13                      INC     DE
0698'  18 F4                   JR      N_ASC1
069A'  13           N_ASC3:    INC     DE
069B'  1A                      LD      A,(DE)
069C'  FE 27                   CP      "'"
069E'  28 F5                   JR      Z,N_ASC2
06A0'  A7                      AND     A
06A1'  C9                      RET

06A2'  CD 0737'     N_REG:     CALL    PRC_REG         ; Numero esta em um registrador
06A5'  30 3C                   JR      NC,JP_ERROR     ; Procura qual ?
06A7'  CD 06D2'                CALL    LD_HL           ; Carrega
06AA'  A7                      AND     A
06AB'  CB 41                   BIT     0,C
06AD'  C0                      RET     NZ
06AE'  26 00                   LD      H,0             ; retorna com zero
06B0'  C9                      RET

06B1'  CD 0641'     N_NEG:     CALL    GET_WRD         ; Numero negativo,
06B4'  2B                      DEC     HL              ; le um numero
06B5'  38 2C        N_NEG1:    JR      C,JP_ERROR
06B7'  7C                      LD      A,H             ; complementa ( Complemento de dois )
06B8'  2F                      CPL
06B9'  67                      LD      H,A
06BA'  7D                      LD      A,L
06BB'  2F                      CPL
06BC'  6F                      LD      L,A
06BD'  C9                      RET

06BE'  CD 0641'     N_CP1:     CALL    GET_WRD         ; Complemento de um
06C1'  18 F2                   JR      N_NEG1

06C3'  78           N_CNTD:    LD      A,B             ; Numero esta no conteudo de uma posicao
06C4'  B1                      OR      C               ; posicao de memoria
06C5'  28 AB                   JR      Z,GET_WRD2
06C7'  CD 0527'                CALL    GETI1           ; Obtem um numero
06CA'  38 17                   JR      C,JP_ERROR
06CC'  1A                      LD      A,(DE)          ; verifica ')'
06CD'  FE 29                   CP      ')'
06CF'  20 12                   JR      NZ,JP_ERROR
06D1'  13                      INC     DE
06D2'  7E           LD_HL:     LD      A,(HL)          ; Pega o conteudo de HL
06D3'  23                      INC     HL
06D4'  66                      LD      H,(HL)          ; HL = ( HL )
06D5'  6F                      LD      L,A
06D6'  C9                      RET

06D7'  CD 0527'     N_PUSH:    CALL    GETI1           ; prioridade nas operacoes
06DA'  38 07                   JR      C,JP_ERROR
06DC'  1A                      LD      A,(DE)
06DD'  FE 5D                   CP      ']'
06DF'  20 02                   JR      NZ,JP_ERROR
06E1'  13                      INC     DE
06E2'  C9                      RET

06E3'  C3 02C8'     JP_ERROR:  JP      ERROR           ; Erro durante o processamento
```

```
06E6'  21 0000      W_HEXA:     LD    HL,0          ; Pega um word em Hexadecimal
06E9'  CD 06FD'     W_HE1:      CALL  ASC_BIN
06EC'  38 0A                    JR    C,W_HE2
06EE'  29                       ADD   HL,HL
06EF'  29                       ADD   HL,HL
06F0'  29                       ADD   HL,HL
06F1'  29                       ADD   HL,HL
06F2'  CD 0428'                 CALL  SUM_HL_A
06F5'  13                       INC   DE
06F6'  18 F1                    JR    W_HE1
06F8'  FE 48        W_HE2:      CP    'H'           ; Pode ter um H no fim do numero
06FA'  C0                       RET   NZ
06FB'  13                       INC   DE
06FC'  C9                       RET

06FD'  1A           ASC_BIN:    LD    A,(DE)        ; Converte de ASCII para binario
06FE'  FE 30        CVT_HEX:    CP    '0'           ; Converte para Hexadecimal
0700'  D8                       RET   C
0701'  FE 3A                    CP    '9'+1
0703'  38 09                    JR    C,CVT_H1
0705'  FE 41                    CP    'A'
0707'  D8                       RET   C
0708'  FE 47                    CP    'F'+1
070A'  3F                       CCF
070B'  D8                       RET   C
070C'  D6 07                    SUB   7
070E'  D6 30        CVT_H1:     SUB   '0'
0710'  C9                       RET

0711'  21 0000      W_DEC:      LD    HL,0          ; Pega um word em decimal
0714'  CD 072C'     W_DEC1:     CALL  CHK_DEC
0717'  13                       INC   DE
0718'  38 0D                    JR    C,W_DEC2
071A'  C5                       PUSH  BC
071B'  29                       ADD   HL,HL
071C'  44                       LD    B,H
071D'  4D                       LD    C,L
071E'  29                       ADD   HL,HL
071F'  29                       ADD   HL,HL
0720'  09                       ADD   HL,BC
0721'  C1                       POP   BC
0722'  CD 0428'                 CALL  SUM_HL_A
0725'  18 ED                    JR    W_DEC1
0727'  FE 2E        W_DEC2:     CP    '.'           ; Deve ter um ponto para indicar decimal
0729'  C8                       RET   Z
072A'  18 B7                    JR    JP_ERROR

072C'  1A           CHK_DEC:    LD    A,(DE)        ; Digito Decimal ?
072D'  FE 30                    CP    '0'
072F'  D8                       RET   C
0730'  FE 3A                    CP    '9'+1         ; entre 0 a 9
0732'  3F                       CCF
0733'  D8                       RET   C
0734'  D6 30                    SUB   '0'
0736'  C9                       RET

0737'  21 1F0A'     PRC_REG:    LD    HL,NAME_REG   ; Procura nos registradores de Software
073A'  CD 0471'                 CALL  CHK7B         ; Procura na tabela
073D'  D0                       RET   NC
073E'  78                       LD    A,B
073F'  80                       ADD   A,B
```

```
0740'  80                        ADD     A,B
0741'  21 1F41'                  LD      HL,SIZ_REG      ; Obtem o tamanho ( Byte, Word )
0744'  CD 0428'                  CALL    SUM_HL_A
0747'  4E                        LD      C,(HL)
0748'  23                        INC     HL              ; Pega o seu conteudo
0749'  CD 06D2'                  CALL    LD_HL
074C'  37                        SCF
074D'  C9                        RET

                    ;************   FIM DO MODULO PRINCIPAL   ****************

                    ;************ INICIO DAS ROTINAS DO MONITOR ****************

074E'  CD 078C'     CMD_DUMP_REG: CALL   OUT_FLAGS       ; Dump dos registradores
                                                         ; Imprime o registrador de Flags (PSW)
0751'  CD 0406'                  CALL    WHITE2
0754'  21 1EC2'                  LD      HL,N_REG_P
0757'  11 1EE2'                  LD      DE,SIZ_R_PR
075A'  06 06                     LD      B,6             ; Imprime os seis registradores
075C'  CD 07D0'     CMD_DUMP1:   CALL    OUT_REG         ; Principais: A, BC, DE, HL, SP, PC
075F'  10 FB                     DJNZ    CMD_DUMP1
0761'  E5                        PUSH    HL
0762'  D5                        PUSH    DE
0763'  FD 2A 40BC                LD      IY,(REG_PC)     ; Desmonta a instrucao apontada pelo
0767'  CD 0DE4'                  CALL    LIST_IY         ;  PC
076A'  D9                        EXX
076B'  08                        EX      AF,AF'
076C'  CD 030A'                  CALL    CR_LF           ; Muda de linha
076F'  CD 079F'                  CALL    OUT_A_FLAGS     ; Mostra as Flags auxilizares
0772'  CD 0406'                  CALL    WHITE2
0775'  D1                        POP     DE
0776'  E1                        POP     HL
0777'  06 07                     LD      B,7             ; Os 7 registradores restantes
0779'  CD 07D0'     CMD_DUMP2:   CALL    OUT_REG         ; A', BC', DE', HL', IX, IY, I
077C'  10 FB                     DJNZ    CMD_DUMP2
077E'  D9                        EXX
077F'  08                        EX      AF,AF'
0780'  A7                        AND     A
0781'  28 06                     JR      Z,CMD_DUMP3
0783'  CD 0400'                  CALL    WHITE6          ; Espacos em branco
0786'  CD 0DCE'                  CALL    CMD_LIST2       ; Imprime o PC relocado
0789'  C3 030A'     CMD_DUMP3:   JP      CR_LF           ; Muda de linha

078C'  3A 40B0      OUT_FLAGS:   LD      A,(REG_F)       ; Imprime as Flags principais
078F'  CD 07A8'                  CALL    OUTFLAGS
0792'  3A 40BA                   LD      A,(60)          ;  e se as interrupcoes estam
0795'  FE F3                     CP      0F3H
0797'  CA 0409'                  JP      Z,WHITE         ;  abilitadas ou nao
079A'  3E 45                     LD      A,'E'
079C'  C3 040B'                  JP      OUTPUT

079F'  3A 40A8      OUT_A_FLAGS: LD      A,(REG_F_)      ; Imprime as Flags Auxiliares
07A2'  CD 07A8'                  CALL    OUTFLAGS        ;       secundarias
07A5'  C3 0409'                  JP      WHITE

07A8'  47           OUTFLAGS:    LD      B,A             ; Imprime o registrador de FLAGS (PSW)
07A9'  3E 53                     LD      A,'S'           ; Sinal
07AB'  CD 07C8'                  CALL    OUTFLAGS1
07AE'  3E 5A                     LD      A,'Z'           ; Zero
07B0'  CD 07C8'                  CALL    OUTFLAGS1
07B3'  CB 10                     RL      B               ; Pula um bit
```

```
07B5' 3E 48                    LD      A,'H'           ; Half Carry ou Aux Carry
07B7' CD 07C8'                 CALL    OUTFLAGS1
07BA' CB 10                    RL      B               ; Pula mais um bit
07BC' 3E 56                    LD      A,'V'           ; Parity/Overflow
07BE' CD 07C8'                 CALL    OUTFLAGS1
07C1' 3E 4E                    LD      A,'N'           ; Subtract
07C3' CD 07C8'                 CALL    OUTFLAGS1
07C6' 3E 43                    LD      A,'C'           ; Carry
07C8' CB 10         OUTFLAGS1: RL      B
07CA' DA 040B'                 JP      C,OUTPUT
07CD' C3 0409'                 JP      WHITE

07D0' C5            OUT_REG:   PUSH    BC              ; Imprime os outros registradores
07D1' D5                       PUSH    DE
                    +          PRINT                   ; Imprime o nome do registrador
07D3' 3E 3D                    LD      A,'='
                    +          OUTPUT                  ;  um sinal de igual
07D6' E3                       EX      (SP),HL
07D7' 5E                       LD      E,(HL)          ; Obtem seu endereco e tipo
07D8' 23                       INC     HL
07D9' 56                       LD      D,(HL)
07DA' 23                       INC     HL
07DB' 7E                       LD      A,(HL)
07DC' 23                       INC     HL
07DD' E5                       PUSH    HL
07DE' A7                       AND     A
07DF' 28 15                    JR      Z,OUT_REG2      ; Tipo Byte ?
07E1' F5                       PUSH    AF
07E2' 1A                       LD      A,(DE)          ; Tipo Word
07E3' 6F                       LD      L,A
07E4' 13                       INC     DE
07E5' 1A                       LD      A,(DE)
07E6' 67                       LD      H,A
07E7' F1                       POP     AF
07E8' 3D                       DEC     A
07E9' 28 08                    JR      Z,OUT_REG1      ; Relocado ?
07EB' CD 03B2'                 CALL    OUT_HL@         ; Relocado, imprime seu valor
07EE' CC 0400'                 CALL    Z,WHITE6
07F1' 18 07                    JR      OUT_REG3
07F3'               +OUT_REG1: OUT_HL                  ; Tipo Word, imprime seu valor
07F4' 18 04                    JR      OUT_REG3
07F6' 1A             OUT_REG2: LD      A,(DE)          ; Tipo Byte, imprime seu valor
07F7' CD 03E8'                 CALL    OUT_A
07FA' CD 0409'       OUT_REG3: CALL    WHITE           ; Imprime um espaco em branco
07FD' D1                       POP     DE
07FE' E1                       POP     HL
07FF' C1                       POP     BC
0800' C9                       RET

0801'               +CMD_S:    SKIP_WHITE              ; Comando de substituicao
0802' 13                       INC     DE
0803' FE 4D                    CP      'M'             ; Substitui na memoria ?
0805' CA 08C6'                 JP      Z,CMD_SM
0808' FE 49                    CP      'I'             ; em portos (I/O)
080A' 20 06                    JR      NZ,CMD_S1
080C' 1A                       LD      A,(DE)
080D' FE 4F                    CP      'O'
080F' CA 0932'                 JP      Z,CMD_SIO
0812' 1B             CMD_S1:   DEC     DE              ; em registradores ?
0813' CD 0737'                 CALL    PRC_REG
0816' D2 08C6'                 JP      NC,CMD_SM       ; Default e' memoria
0819' CD 0452'                 CALL    END_CMD
```

```
081C'  78                        LD      A,B
081D'  FE 1E                     CP      30              ; Flag F'
081F'  28 58                     JR      Z,CMD_SFA
0821'  FE 1F                     CP      31              ; Flag F
0823'  28 44                     JR      Z,CMD_SF
0825'  FE 1D                     CP      29              ; Registrador IP nao pode
0827'  CA 02C8'                  JP      Z,ERROR
082A'  EB                        EX      DE,HL
082B'  21 1F0A'                  LD      HL,NAME_REG     ; Imprime o nome do registrador
082E'  CD 04BF'                  CALL    LOC_TAB
0831'  41              CMD_S2:   LD      B,C
                     +           PRINT
0833'  3E 3D                     LD      A,'='           ; Um sinal de igual
                     +           OUTPUT
0836'  CD 0856'                  CALL    POUT_CDE
0839'  CD 0409'                  CALL    WHITE
083C'  D5                        PUSH    DE
083D'  C5                        PUSH    BC
083E'  CD 0331'                  CALL    GET_LINE        ; Obtem uma linha
                     +           SKIP_WHITE
0842'  A7                        AND     A
0843'  28 0E                     JR      Z,CMD_S3        ; Fim ?
0845'  CD 0446'                  CALL    GET_INT_END     ; Obtem um numero
0848'  44                        LD      B,H
0849'  4D                        LD      C,L
084A'  F1                        POP     AF
084B'  E1                        POP     HL
084C'  71                        LD      (HL),C          ; Armazena-o
084D'  CB 47                     BIT     0,A
084F'  C8                        RET     Z
0850'  23                        INC     HL
0851'  70                        LD      (HL),B
0852'  C9                        RET
0853'  F1              CMD_S3:   POP     AF
0854'  E1                        POP     HL
0855'  C9                        RET

0856'  1A              POUT_CDE: LD      A,(DE)          ; Imprima o conteudo apontado por DE
0857'  CB 40                     BIT     0,B             ; Byte ?
0859'  CA 03EB'                  JP      Z,OUT_A
085C'  6F                        LD      L,A
085D'  13                        INC     DE              ; Word!
085E'  1A                        LD      A,(DE)
085F'  1B                        DEC     DE
0860'  67                        LD      H,A
0861'  CB 48                     BIT     1,B
0863'  CA 03E6'                  JP      Z,OUT_HL        ; Word relocado!
0866'  C3 03B2'                  JP      OUT_HL@

0869'  CD 078C'        CMD_SF:   CALL    OUT_FLAGS       ; Modifica as Flags principais
086C'  3E F3                     LD      A,0F3H          ; Assume interrupcoes desligadas (DI)
086E'  32 40BA                   LD      (GO),A
0871'  37                        SCF
0872'  CD 0884'                  CALL    CMD_SFLAG       ; Modifica
0875'  32 40B0                   LD      (REG_F),A       ; Armazena nas Flags principais
0878'  C9                        RET

0879'  CD 079F'        CMD_SFA:  CALL    OUT_A_FLAGS     ; Modifica as Flags secundarias
087C'  A7                        AND     A
087D'  CD 0884'                  CALL    CMD_SFLAG       ; Modifica
0880'  32 40A8                   LD      (REG_F_),A      ; Armazena nas Flags secundarias
0883'  C9                        RET
```

```
0884'  08           CMD_SFLAG:    EX     AF,AF'          ; Altera o valor da Flags
0885'  06 00                      LD     B,0
0887'  CD 0409'                   CALL   WHITE
088A'  CD 0452'                   CALL   END_CMD         ; Limpa Buffer
088D'  CD 0331'                   CALL   GET_LINE        ; Obtem uma linha
0890'              +CMD_SFLAG1:   SKIP_WHITE             ; Pula brancos
0891'  A7                        AND    A
0892'  78                        LD     A,B
0893'  C8                        RET    Z               ; Fim da linha ?
0894'  C5                        PUSH   BC
0895'  21 08BC'                  LD     HL,SFLAGS       ; Procura a Flag a ser alterada
0898'  CD 0461'                  CALL   CHK_TAB
089B'  D2 02C8'                  JP     NC,ERROR        ; Nao achou, erro
089E'  78                        LD     A,B
089F'  FE 08                     CP     8               ; Flag E ?
08A1'  28 0C                     JR     Z,CMD_SFLAG3
08A3'  47                        LD     B,A
08A4'  04                        INC    B
08A5'  3E 80                     LD     A,80H
08A7'  07           CMD_SFLAG2:  RLCA                    ; Roda ate posicionar na Flag desejada
08A8'  10 FD                     DJNZ   CMD_SFLAG2
08AA'  C1                        POP    BC
08AB'  B0                        OR     B               ; Ativa a Flag
08AC'  47                        LD     B,A
08AD'  18 E1                     JR     CMD_SFLAG1
08AF'  08           CMD_SFLAG3:  EX     AF,AF'
08B0'  D2 02C8'                  JP     NC,ERROR
08B3'  08                        EX     AF,AF'
08B4'  3E FB                     LD     A,0FBH          ; Assume Interrupcao abilitada (EI)
08B6'  32 40BA                   LD     (GO),A
08B9'  C1                        POP    BC
08BA'  18 D4                     JR     CMD_SFLAG1

08BC'  43 4E 56     SFLAGS:      DEFM   'CNV H ZSE',0   ; Flags
08BF'  20 48 20
08C2'  5A 53 45
08C5'  00

08C6'  2A 401B      CMD_SM:      LD     HL,(POSICAO)    ; Substitui em posicoes de
08C9'  CD 044C'                  CALL   DEF_END         ;  memoria
08CC'  22 401B      CMD_SM1:     LD     (POSICAO),HL
08CF'  CD 03B2'                  CALL   OUT_HL@         ; Imprime a posicao relocada
08D2'  CD 0409'                  CALL   WHITE
08D5'  7E                        LD     A,(HL)
08D6'  CD 03EB'                  CALL   OUT_A           ; Seu valor corrente
08D9'  CD 0406'                  CALL   WHITE2
08DC'  E5                        PUSH   HL
08DD'  CD 0331'                  CALL   GET_LINE        ; Obtem uma linha
08E0'  E1                        POP    HL
               +                 SKIP_WHITE             ; Pula os brancos
08E2'  A7                        AND    A
08E3'  20 03                     JR     NZ,CMD_SM2
08E5'  23                        INC    HL
08E6'  18 E4                     JR     CMD_SM1
08E8'  1A           CMD_SM2:     LD     A,(DE)
08E9'  FE 2E                     CP     '.'             ; Fim do comando ?
08EB'  C8                        RET    Z
08EC'  FE 2D                     CP     '-'             ; Retrocede um endereco ?
08EE'  28 19                     JR     Z,CMD_SM6
08F0'  FE 27                     CP     "'"             ; Coloca uma cadeia de caracteres ?
08F2'  28 1E                     JR     Z,CMD_SM7
```

```
08F4'  E5           CMD_SM3:    PUSH    HL
                  +             GET_INT             ; Obtem um valor
08F6'  7D                       LD      A,L         ; Pega os 8 bits menos significativos
08F7'  E1                       POP     HL
08F8'  38 0A                    JR      C,CMD_SM5
08FA'  77                       LD      (HL),A      ; Armazena na posicao
08FB'  23                       INC     HL          ; Muda de posicao
08FC'  22 401B      CMD_SM4:    LD      (POSICAO),HL
08FF'  CD 0434'                 CALL    SKIP_COMMA  ; Pula virgulas
0902'  18 E4                    JR      CMD_SM2
0904'  CD 0452'     CMD_SM5:    CALL    END_CMD     ; Fim de linha
0907'  18 C3                    JR      CMD_SM1
0909'  13           CMD_SM6:    INC     DE          ; Decrementa posicao
090A'  1A                       LD      A,(DE)
090B'  1B                       DEC     DE
090C'  A7                       AND     A
090D'  20 E5                    JR      NZ,CMD_SM3
090F'  2B                       DEC     HL
0910'  18 BA                    JR      CMD_SM1
0912'  13           CMD_SM7:    INC     DE          ; Coloca a cadeia de caracteres
0913'  1A                       LD      A,(DE)
0914'  A7                       AND     A
0915'  28 B5                    JR      Z,CMD_SM1
0917'  FE 27                    CP      "'"         ; ate o fim (apostrofe)
0919'  28 04                    JR      Z,CMD_SM9
091B'  77           CMD_SM8:    LD      (HL),A
091C'  23                       INC     HL
091D'  18 F3                    JR      CMD_SM7
091F'  13           CMD_SM9:    INC     DE
0920'  1A                       LD      A,(DE)
0921'  FE 27                    CP      "'"         ; ou e' para colocar o apostrofe
0923'  28 F6                    JR      Z,CMD_SM8
0925'  FE 22                    CP      '"'         ; No caso de aspas a cadeia deve
0927'  20 D3                    JR      NZ,CMD_SM4  ; terminar com o bit 7 ligado
0929'  13                       INC     DE
092A'  2B                       DEC     HL
092B'  7E                       LD      A,(HL)
092C'  F6 80                    OR      80H         ; Liga o bit 7
092E'  77                       LD      (HL),A
092F'  23                       INC     HL
0930'  18 CA                    JR      CMD_SM4

0932'  13           CMD_SIO:    INC     DE          ; Comando de entrada e saida em portas
0933'  CD 0446'                 CALL    GET_INT_END ; I/O
0936'  26 01                    LD      H,1
0938'  E5           CMD_I2:     PUSH    HL          ; Imprime o endereco da porta (nn)
0939'  3E 28                    LD      A,'('
                  +             OUTPUT
093C'  7D                       LD      A,L
093D'  CD 03EB'                 CALL    OUT_A
0940'  3E 29                    LD      A,')'
                  +             OUTPUT
0943'  3E 3D                    LD      A,'='
                  +             OUTPUT
0946'  7C                       LD      A,H
0947'  A7                       AND     A
0948'  20 09                    JR      NZ,CMD_I1
094A'  4D                       LD      C,L
094B'  ED 78                    IN      A,(C)       ; Le o valor atual e imprime
094D'  CD 03EB'                 CALL    OUT_A
0950'  CD 0409'                 CALL    WHITE
0953'  CD 0331'     CMD_I1:     CALL    GET_LINE    ; Le o novo valor
```

```
                        +               SKIP_WHITE
0957'  E1                               POP     HL
0958'  67                               LD      H,A
0959'  FE 2E                            CP      '.'             ; Fim ?
095B'  C8                               RET     Z
095C'  A7                               AND     A
095D'  28 D9                            JR      Z,CMD_I2
095F'  E5                               PUSH    HL
0960'  CD 0446'                         CALL    GET_INT_END     ; Obtem um inteiro da linha lida
0963'  C1                               POP     BC
0964'  ED 69                            OUT     (C),L           ; Armazena no porto
0966'  60                               LD      H,B
0967'  69                               LD      L,C
0968'  18 CE                            JR      CMD_I2          ; Repete ate o fim (.)

096A'  CD 0452'         CMD_@:          CALL    END_CMD         ; Comando @, armazena o endereco
096D'  21 002B'                         LD      HL,SYM_@        ;  referencia, para as relocacoes
0970'  11 401E                          LD      DE,VAR_@
0973'  0E 01                            LD      C,1
0975'  C3 0831'                         JP      CMD_S2          ; Armazena na variavel

0978'  CD 04D0'         CMD_V:          CALL    GET_INT3_END    ; Comando V, verifica blocos de dados
                                                                ; Pega 3 numeros (Inicio, Fim, Inicio1)
097B'  C5               CMD_V1:         PUSH    BC
097C'  1A                               LD      A,(DE)
097D'  46                               LD      B,(HL)
097E'  B8                               CP      B               ; Compara a primeira posicao
097F'  28 1D                            JR      Z,CMD_V2        ; Ok, continua a comparar
0981'  4F                               LD      C,A             ; Erro, imprime a primeira posicao
0982'  CD 03B2'                         CALL    OUT_HL@         ;  relocada (se ativo)
0985'  CD 0409'                         CALL    WHITE
0988'  78                               LD      A,B
0989'  CD 03EB'                         CALL    OUT_A           ;  o primeiro valor
098C'  CD 0406'                         CALL    WHITE2
098F'  79                               LD      A,C
0990'  CD 03EB'                         CALL    OUT_A           ;  o segundo valor
0993'  CD 0409'                         CALL    WHITE
0996'  EB                               EX      DE,HL
0997'  CD 03B2'                         CALL    OUT_HL@         ;  a segunda posicao relocada
099A'  EB                               EX      DE,HL
099B'  CD 030A'                         CALL    CR_LF           ; Muda de linha
099E'  C1               CMD_V2:         POP     BC
099F'  23                               INC     HL              ; Incrementa a posicao
09A0'  13                               INC     DE
09A1'  0B                               DEC     BC
09A2'  78                               LD      A,B
09A3'  B1                               OR      C               ; Fim ?
09A4'  20 D5                            JR      NZ,CMD_V1
09A6'  C9                               RET

09A7'  CD 04D0'         CMD_M:          CALL    GET_INT3_END    ; Comando M, move dados
09AA'  CD 09AF'                         CALL    MOVEDATA        ; Pega 3 numeros (Inicio, fim, destino)
                                                                ;  move os bytes
09AD'  18 CC                            JR      CMD_V1          ; Verifica se esta tudo bem

09AF'  E5               MOVEDATA:       PUSH    HL              ; Move bytes de uma posicao para outra
09B0'  D5                               PUSH    DE
09B1'  C5                               PUSH    BC
09B2'  ED B0                            LDIR                    ; Fonte = HL, Destino = DE
09B4'  C1                               POP     BC              ; Numero de bytes = BC
09B5'  D1                               POP     DE
09B6'  E1                               POP     HL
```

```
09B7'  C9                          RET

09B8'                +CMD_H:       GET_INT              ; Comando H, soma e subtrai numeros
09B9'  DA 02C8'                    JP      C,ERROR
09BC'  CD 0434'                    CALL    SKIP_COMMA   ; Pega um numero e pula virgulas
09BF'  E5                          PUSH    HL           ;  se existir
                     +             GET_INT              ; Pega outro numero, se existir
09C1'  F5                          PUSH    AF
09C2'  CD 0452'                    CALL    END_CMD      ; Fim
09C5'  F1                          POP     AF
09C6'  EB                          EX      DE,HL
09C7'  E1                          POP     HL
09C8'  38 15                       JR      C,CMD_H1
09CA'  E5                          PUSH    HL
09CB'  D5                          PUSH    DE
09CC'  19                          ADD     HL,DE        ; Soma
                     +             OUT_HL
09CE'  CD 0409'                    CALL    WHITE
09D1'  CD 03CF'                    CALL    OUT_DEC      ; Imprime em Hexa e em Decimal
09D4'  3E 2C                       LD      A,','
                     +             OUTPUT
09D7'  CD 0406'                    CALL    WHITE2
09DA'  D1                          POP     DE
09DB'  E1                          POP     HL
09DC'  A7                          AND     A
09DD'  ED 52                       SBC     HL,DE        ; Subtrai
09DF'                +CMD_H1:      OUT_HL               ; Imprime em Hexa e em Decimal
09E0'  CD 0409'                    CALL    WHITE
09E3'  CD 03CF'                    CALL    OUT_DEC
09E6'  C3 030A'                    JP      CR_LF

09E9'  CD 04E3'      CMD_Q:        CALL    GET_INT2     ; Comando Q, procura uma sequencia
09EC'  C5                          PUSH    BC           ;  de byte na memoria
09ED'  E5                          PUSH    HL
09EE'  CD 0A33'                    CALL    CMD_Z3       ; Obtem o terceiro parametro
09F1'  E1                          POP     HL
09F2'  CD 0A09'      CMD_Q1:       CALL    CMP_DE_HL    ; Compara o conteudo do ender.
09F5'  20 08                       JR      NZ,CMD_Q2    ; Diferente ?
09F7'  C5                          PUSH    BC
09F8'  01 0010                     LD      BC,16        ; Igual, mostra 16 bytes
09FB'  CD 0A82'                    CALL    CMD_D_MEM
09FE'  C1                          POP     BC
09FF'  23            CMD_Q2:       INC     HL           ; avanca endereco
0A00'  E3                          EX      (SP),HL
0A01'  2B                          DEC     HL           ; Calcula a diferenca dos ender.
0A02'  7C                          LD      A,H
0A03'  B5                          OR      L
0A04'  E3                          EX      (SP),HL
0A05'  20 EB                       JR      NZ,CMD_Q1    ; Chegou ao fim ?
0A07'  C1                          POP     BC
0A08'  C9                          RET

0A09'  E5            CMP_DE_HL:    PUSH    HL           ; Compara o conteudo apontado por
0A0A'  D5                          PUSH    DE           ;  DE com o apontado por HL
0A0B'  C5                          PUSH    BC           ;  BC contem o numero de caracteres
0A0C'  1A            CMP_DE_HL1:   LD      A,(DE)
0A0D'  BE                          CP      (HL)         ; Compara ?
0A0E'  20 04                       JR      NZ,CMP_DE_HL2
0A10'  13                          INC     DE           ; Avanca DE e HL
0A11'  23                          INC     HL
0A12'  10 F8                       DJNZ    CMP_DE_HL1   ; Proximo
0A14'  C1            CMP_DE_HL2:   POP     BC
```

```
0A15'  D1                        POP     DE
0A16'  E1                        POP     HL
0A17'  C9                        RET

0A18'  CD 04E3'      CMD_Z:      CALL    GET_INT2        ; Comando Z, Zap
0A1B'  C5                        PUSH    BC
0A1C'  E5                        PUSH    HL              ; Obtem ender. inicial e final
0A1D'  CD 0A33'                  CALL    CMD_Z3          ; Obtem valor ou valores
0A20'  78                        LD      A,B             ;  a serem colocados nos ender.
0A21'  E1                        POP     HL
0A22'  C1                        POP     BC
0A23'  E5                        PUSH    HL
0A24'  EB                        EX      DE,HL
0A25'  ED A0         CMD_Z1:     LDI                     ; Preenche a memoria
0A27'  E2 0A31'                  JP      PO,CMD_Z2
0A2A'  3D                        DEC     A
0A2B'  20 F8                     JR      NZ,CMD_Z1
0A2D'  E1                        POP     HL
0A2E'  ED B0                     LDIR                    ; Repete ate o fim
0A30'  C9                        RET

0A31'  E1            CMD_Z2:     POP     HL              ; Caso ja tenha atingido o fim
0A32'  C9                        RET


0A33'  21 4020       CMD_Z3:     LD      HL,BUF_LIN      ; Le uma linha, pois o valor
0A36'  CD 0434'      CMD_Z4:     CALL    SKIP_COMMA      ;  a ser colocado pode ser um
0A39'  1A                        LD      A,(DE)          ;  inteiro ou uma cadeia de
0A3A'  FE 27                     CP      "'"             ;  caracteres
0A3C'  28 0A                     JR      Z,CMD_Z5
0A3E'  E5                        PUSH    HL
                     +           GET_INT                 ; Le um inteiro
0A40'  7D                        LD      A,L
0A41'  E1                        POP     HL
0A42'  38 1A                     JR      C,CMD_Z9
0A44'  77                        LD      (HL),A
0A45'  23                        INC     HL
0A46'  18 EE                     JR      CMD_Z4
0A48'  13            CMD_Z5:     INC     DE
0A49'  1A            CMD_Z6:     LD      A,(DE)
0A4A'  FE 27                     CP      "'"             ; Cadeia de caracteres ?
0A4C'  28 08                     JR      Z,CMD_Z8
0A4E'  A7                        AND     A
0A4F'  28 0D                     JR      Z,CMD_Z9
0A51'  77            CMD_Z7:     LD      (HL),A          ; Armazena o valor
0A52'  23                        INC     HL
0A53'  13                        INC     DE
0A54'  18 F3                     JR      CMD_Z6
0A56'  13            CMD_Z8:     INC     DE              ; Incrementa ponteiro do buffer
0A57'  1A                        LD      A,(DE)          ;  de linha
0A58'  FE 27                     CP      "'"             ; Colocar apostrofe ?
0A5A'  28 F5                     JR      Z,CMD_Z7        ;  sim
0A5C'  18 D8                     JR      CMD_Z4          ;  nao, analize proximo caracter
0A5E'  CD 0452'      CMD_Z9:     CALL    END_CMD         ; Fim da linha
0A61'  11 4020                   LD      DE,BUF_LIN
0A64'  A7                        AND     A
0A65'  ED 52                     SBC     HL,DE           ; Calcule numero de caracter
0A67'  45                        LD      B,L             ;  a serem armazenados
0A68'  CA 02C8'                  JP      Z,ERROR
0A6B'  C9                        RET

0A6C'  1A            CMD_D:      LD      A,(DE)          ; Comando D, Dump memoria ou reg.
```

```
0A6D'  FE 52                    CP      'R'             ; Registrador ?
0A6F'  20 07                    JR      NZ,CMD_D1
0A71'  13                       INC     DE
0A72'  CD 0452'                 CALL    END_CMD         ; Sim, mostra os registradores
0A75'  C3 074E'                 JP      CMD_DUMP_REG
0A78'  2A 401B          CMD_D1: LD      HL,(POSICAO)    ; Obtem o endereco a ser mostrado
0A7B'  01 0080                  LD      BC,128          ; Default = posicao correte
0A7E'  CD 0441'                 CALL    GET_INT2_NOER_END       ; mostrar 128 bytes
0A81'  37                       SCF

0A82'  C5               CMD_D_MEM:      PUSH    BC              ; Dump da memoria
0A83'  D5                       PUSH    DE
0A84'  E5                       PUSH    HL
0A85'  08                       EX      AF,AF'
0A86'  CD 03B2'         CMD_D_M1:       CALL    OUT_HL@         ; Mostre o endereco relocado
0A89'  CC 0406'                 CALL    Z,WHITE2
0A8C'  CD 0409'                 CALL    WHITE           ; Brancos
0A8F'  11 0000                  LD      DE,0
0A92'  7E               CMD_D_M2:       LD      A,(HL)          ; Mostre 16 bytes
0A93'  23                       INC     HL
0A94'  CD 03E8'                 CALL    OUT_A
0A97'  CD 0409'                 CALL    WHITE
0A9A'  0B                       DEC     BC
0A9B'  1C                       INC     E
0A9C'  7B                       LD      A,E
0A9D'  FE 10                    CP      16
0A9F'  28 09                    JR      Z,CMD_D_M3
0AA1'  E6 03                    AND     3
0AA3'  CC 0409'                 CALL    Z,WHITE
0AA6'  78                       LD      A,B
0AA7'  B1                       OR      C
0AA8'  20 E8                    JR      NZ,CMD_D_M2
0AAA'  CD 0409'         CMD_D_M3:       CALL    WHITE
0AAD'  A7                       AND     A
0AAE'  ED 52                    SBC     HL,DE
0AB0'  7E               CMD_D_M4:       LD      A,(HL)
0AB1'  CD 0ACB'                 CALL    CMD_D_M6        ; ASCII ?
                      +                 OUTPUT                  ; Imprima o caracter
0AB5'  23                       INC     HL
0AB6'  1D                       DEC     E
0AB7'  20 F7                    JR      NZ,CMD_D_M4
0AB9'  08                       EX      AF,AF'
0ABA'  30 03                    JR      NC,CMD_D_M5
0ABC'  22 401B                  LD      (POSICAO),HL    ; Marca a nova posicao corrente
0ABF'  08               CMD_D_M5:       EX      AF,AF'
0AC0'  CD 030A'                 CALL    CR_LF           ; Mude de linha
0AC3'  78                       LD      A,B
0AC4'  B1                       OR      C
0AC5'  20 BF                    JR      NZ,CMD_D_M1
0AC7'  E1                       POP     HL
0AC8'  D1                       POP     DE
0AC9'  C1                       POP     BC
0ACA'  C9                       RET

0ACB'  E6 7F            CMD_D_M6:       AND     7FH             ; Caracter ASCII ?
0ACD'  FE 7F                    CP      7FH
0ACF'  28 03                    JR      Z,CMD_D_M7
0AD1'  FE 20                    CP      20H
0AD3'  D0                       RET     NC
0AD4'  3E 2E            CMD_D_M7:       LD      A,'.'           ; Caso negativo, retorne com '.'
0AD6'  C9                       RET
```

```
                        ; Comando B: Ponto de parada
                        ;
                        ; Estrutura utilizada:
                        ;               DB      Estado
                        ;               DB      Instrucao retirada
                        ;               DW      Endereco do PP
                        ;               DW      Numero de vezes
                        ;
                        ; Estado:       ! 7 ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! 0 !
                        ;                 ^   ^   ^   ^   ^   ^   ^   ^- PP Ativo
                        ;                 ^   ^   ^   ^   ^   ^   ^- Temporario
                        ;                 ^   ^   ^   ^   ^   ^- Uso interno PP breve
                        ;                 ^   ^   ^   ^   ^- Igual ao endereco de parada
                        ;                 ^   ^   ^   ^- Repeticao
                        ;                 ^   ^   ^- RST 38H colocado
                        ;                 ^-  ^-  Sem uso

0AD7'                   +CMD_B:         SKIP_WHITE              ; Comando B, ponto de parada (PP)
0AD8'   B7                              OR      A
0AD9'   28 6F                           JR      Z,CMD_BL        ; Sem opcao ?
0ADB'   13                              INC     DE
0ADC'   FE 58                           CP      'X'             ; Limpa os pontos de parada ?
0ADE'   28 36                           JR      Z,CMD_BX
0AE0'   1B                              DEC     DE
0AE1'   3E 01                           LD      A,1
0AE3'   47              CMD_B1:         LD      B,A
                        +               SKIP_WHITE
0AE5'   A7                              AND     A
0AE6'   C8                              RET     Z
0AE7'   FE 52                           CP      'R'             ; Opcao de repeticao ?
0AE9'   20 03                           JR      NZ,CMD_B2
0AEB'   13                              INC     DE
0AEC'   CB E0                           SET     4,B
0AEE'   C5              CMD_B2:         PUSH    BC
0AEF'   CD 0B89'                        CALL    BREAK_FREE      ; Optem um PP livre
                        +               GET_INT                 ; Le o endereco do PP
0AF3'   DA 02C8'                        JP      C,ERROR
0AF6'   7C                              LD      A,H
0AF7'   FE 20                           CP      20H             ; Na EPROM ?
0AF9'   DA 02C8'                        JP      C,ERROR
0AFC'   DD 75 02                        LD      (IX+2),L        ; Armazena o endereco
0AFF'   DD 74 03                        LD      (IX+3),H
0B02'   CD 0BA1'                        CALL    CMD_B_CNT       ; Le o contador
0B05'   DD 75 04                        LD      (IX+4),L        ; Armazena
0B08'   DD 74 05                        LD      (IX+5),H
0B0B'   CD 0434'                        CALL    SKIP_COMMA      ; Pula virgulas se existir
0B0E'   F1                              POP     AF
0B0F'   DD 77 00                        LD      (IX+0),A
0B12'   E6 0F                           AND     0FH
0B14'   18 CD                           JR      CMD_B1
                        ;
0B16'                   +CMD_BX:        SKIP_WHITE              ; Limpa PP
0B17'   A7                              AND     A
0B18'   28 0E                           JR      Z,CMD_BX2       ; Limpa todos ?
0B1A'                   +CMD_BX1:       GET_INT                 ; Obtem o endereco do PP
0B1B'   DA 0452'                        JP      C,END_CMD
0B1E'   D5                              PUSH    DE
0B1F'   CD 0B29'                        CALL    CMD_BX3         ; Procura e remove
0B22'   D1                              POP     DE
0B23'   CD 0434'                        CALL    SKIP_COMMA
0B26'   18 F2                           JR      CMD_BX1
```

```
0B28'  37           CMD_BX2:    SCF                 ; Limpa trace
0B29'  06 0C        CMD_BX3:    LD      B,12        ; Limpa os PP
0B2B'  DD 21 40C3               LD      IX,BREAKPOINTS
0B2F'  F5           CMD_BX4:    PUSH    AF
0B30'  38 0B                    JR      C,CMD_BX5
0B32'  DD 5E 02                 LD      E,(IX+2)
0B35'  DD 56 03                 LD      D,(IX+3)
0B38'  CD 045B'                 CALL    CMP_HL_DE   ; Compara
0B3B'  20 04                    JR      NZ,CMD_BX6
0B3D'  DD 36 00     CMD_BX5:    LD      (IX+0),0    ; Igual, remove o PP
0B40'  00
0B41'  11 0006      CMD_BX6:    LD      DE,6
0B44'  DD 19                    ADD     IX,DE
0B46'  F1                       POP     AF
0B47'  10 E6                    DJNZ    CMD_BX4
0B49'  C9                       RET
                    ;
0B4A'  06 0C        CMD_BL:     LD      B,12        ; Lista os pontos de parada
0B4C'  DD 21 40C3               LD      IX,BREAKPOINTS
0B50'  DD CB 00     CMD_BL1:    BIT     0,(IX+0)    ; PP ativo ?
0B53'  46
0B54'  28 2B                    JR      Z,CMD_BL3
0B56'  21 001B'                 LD      HL,MSG_BP   ; sim, imprime mensagem
                    +           PRINT
0B5A'  3E 52                    LD      A,'R'       ; Repeticao ?
0B5C'  DD CB 00                 BIT     4,(IX+0)
0B5F'  66
0B60'  20 02                    JR      NZ,CMD_BL2
0B62'  3E 20                    LD      A,' '       ; Nao
0B64'               +CMD_BL2:   OUTPUT
0B65'  CD 0409'                 CALL    WHITE
0B68'  DD 6E 02                 LD      L,(IX+2)
0B6B'  DD 66 03                 LD      H,(IX+3)
0B6E'  CD 03B2'                 CALL    OUT_HL@     ; Imprime endereco do PP
0B71'  CD 0409'                 CALL    WHITE
0B74'  3E 3A                    LD      A,':'       ; Imprime contador
                    +           OUTPUT
0B77'  DD 6E 04                 LD      L,(IX+4)
0B7A'  DD 66 05                 LD      H,(IX+5)
                    +           OUT_HL
0B7E'  CD 0304'                 CALL    CR_LF
0B81'  11 0006      CMD_BL3:    LD      DE,6        ; Proximo PP
0B84'  DD 19                    ADD     IX,DE
0B86'  10 C8                    DJNZ    CMD_BL1
0B88'  C9                       RET

0B89'  06 0C        BREAK_FREE: LD      B,12        ; Localiza um BreakPoint livre
0B8B'  DD 21 40C3               LD      IX,BREAKPOINTS ; Procura o primeiro livre
0B8F'  DD 7E 00     BREAK_FREE1: LD     A,(IX+0)
0B92'  E6 0F                    AND     0FH
0B94'  C8                       RET     Z           ; Achou ?
0B95'  C5                       PUSH    BC
0B96'  01 0006                  LD      BC,6        ; Proximo BP
0B99'  DD 09                    ADD     IX,BC
0B9B'  C1                       POP     BC
0B9C'  10 F1                    DJNZ    BREAK_FREE1
0B9E'  C3 02C8'                 JP      ERROR       ; Nenhum BP livre

0BA1'               +CMD_B_CNT: SKIP_WHITE          ; Le o contador de vezes
0BA2'  21 0001                  LD      HL,1        ; Valor default 1
0BA5'  FE 3A                    CP      ':'         ; Deve-ter ':'
0BA7'  C0                       RET     NZ
```

```
0BA8'  13                        INC     DE
                        +        GET_INT                 ; Le o valor do contador
0BAA'  DA 02C8'                  JP      C,ERROR
0BAD'  C9                        RET

0BAE'  06 0C            SLEEP_BREAK:    LD      B,12            ; Inicializa todos os pontos de parada
0BB0'  DD 21 40C3                LD      IX,BREAKPOINTS
0BB4'  DD 7E 00         SLEEP_BRK1:     LD      A,(IX+0)
0BB7'  E6 F1                     AND     0F1H            ; Desliga os bits de sinalizacao
0BB9'  DD 77 00                  LD      (IX+0),A
0BBC'  11 0006                   LD      DE,6
0BBF'  DD 19                     ADD     IX,DE
0BC1'  10 F1                     DJNZ    SLEEP_BRK1
0BC3'  C9                        RET

0BC4'  AF               CMD_G:           XOR     A               ; Comando G - Go
0BC5'  32 4011                   LD      (TRACE?),A      ; Sem trace e dump
0BC8'  32 4010                   LD      (DUMP?),A
                        +        GET_INT                 ; Obtem o endereco
0BCC'  38 03                     JR      C,CMD_G1        ; Utilizar o PC
0BCE'  22 40BC                   LD      (REG_PC),HL
0BD1'                   +CMD_G1:         SKIP_WHITE
0BD2'  A7                        AND     A
0BD3'  28 0B                     JR      Z,GO_PRG        ; Sem Opcao ?
0BD5'  FE 2F                     CP      '/'             ; Ponto de parada temporario ?
0BD7'  C2 02C8'                  JP      NZ,ERROR
0BDA'  13                        INC     DE              ; Sim, adiciona um PP temporario
0BDB'  3E 02                     LD      A,2
0BDD'  CD 0AE3'                  CALL    CMD_B1
                        ;
0BE0'  AF               GO_PRG:          XOR     A
0BE1'  32 4014                   LD      (VAR_1),A
0BE4'  3A 4010                   LD      A,(DUMP?)
0BE7'  A7                        AND     A
0BE8'  C4 074E'                  CALL    NZ,CMD_DUMP_REG ; Mostra os registradores ?
0BEB'  CD 0CF3'                  CALL    CMD_GBREAK      ; Esta parado em ponto de parada ?
0BEE'  0E 07                     LD      C,7
0BF0'  20 0A                     JR      NZ,GO_ON        ; Nao, execucao normal
0BF2'  3E 01                     LD      A,1             ; Sim, coloca um ponto de parada
0BF4'  32 4014                   LD      (VAR_1),A       ;  apos a instrucao corrente
0BF7'  CD 10DD'                  CALL    NXT_INSTR
0BFA'  0E 08                     LD      C,8
0BFC'  CD 0D36'         GO_ON:           CALL    SET_BRK
0BFF'  31 40A2                   LD      SP,REG_HL_
0C02'  E1                        POP     HL
0C03'  D1                        POP     DE
0C04'  C1                        POP     BC
0C05'  F1                        POP     AF
0C06'  D9                        EXX
0C07'  08                        EX      AF,AF'
0C08'  F1                        POP     AF
0C09'  ED 47                     LD      I,A
0C0B'  FD E1                     POP     IY
0C0D'  DD E1                     POP     IX
0C0F'  F1                        POP     AF
0C10'  C1                        POP     BC
0C11'  D1                        POP     DE
0C12'  E1                        POP     HL
0C13'  ED 7B 4088                LD      SP,(REG_SP)
0C17'  C3 40BA                   JP      GO
                        ;
                        ;       Quando um ponto de parada ou um RST 38H e' atingido a execucao
```

```
                        ;               cai na rotina BREAK.
                        ;
0C1A'  22 40B6          BREAK:          LD      (REG_HL),HL     ; Salva registradores
0C1D'  E1                               POP     HL
0C1E'  2B                               DEC     HL              ; PC = PC - 1,  (RST 38H)
0C1F'  22 40BC                          LD      (REG_PC),HL
0C22'  ED 73 40B8                       LD      (REG_SP),SP
0C26'  31 40B6                          LD      SP,REG_HL
0C29'  D5                               PUSH    DE
0C2A'  C5                               PUSH    BC
0C2B'  F5                               PUSH    AF
0C2C'  DD E5                            PUSH    IX
0C2E'  FD E5                            PUSH    IY
0C30'  ED 57                            LD      A,I
0C32'  67                               LD      H,A
0C33'  2E 00                            LD      L,0
0C35'  E5                               PUSH    HL
0C36'  3E F3                            LD      A,0F3H          ; DI, verifica interrupcoes
0C38'  E2 0C3D'                         JP      PO,BREAK1
0C3B'  3E FB                            LD      A,0FBH          ; EI
0C3D'  32 40BA          BREAK1:         LD      (GO),A
0C40'  08                               EX      AF,AF'          ; Salva registradores secundarios
0C41'  F5                               PUSH    AF
0C42'  D9                               EXX
0C43'  C5                               PUSH    BC
0C44'  D5                               PUSH    DE
0C45'  E5                               PUSH    HL
0C46'  CD 0CC8'                         CALL    RESTORE_BREAK   ; Restaura os pontos de parada
0C49'  3A 4014                          LD      A,(VAR_1)       ; PP interno ?
0C4C'  3D                               DEC     A
0C4D'  28 20                            JR      Z,BREAK2        ; Sim
0C4F'  CD 0C76'                         CALL    BREAK3          ; Achou o PP ?
0C52'  A7                               AND     A
0C53'  28 8B                            JR      Z,GO_PRG        ; Interno
0C55'  E6 83                            AND     83H
0C57'  CA 11A7'                         JP      Z,CMD_TRACE             ; comandos trace ou call ?
0C5A'  2A 40BC                          LD      HL,(REG_PC)     ; Verifica instr. RST
0C5D'  7E                               LD      A,(HL)
0C5E'  FE FF                            CP      RESTART
0C60'  20 04                            JR      NZ,STOP
0C62'  23                               INC     HL              ; Avanca endereco
0C63'  22 40BC                          LD      (REG_PC),HL
0C66'  CD 074E'         STOP:           CALL    CMD_DUMP_REG    ; Mostra Registradores
0C69'  CD 0BAE'                         CALL    SLEEP_BREAK     ; Desarma PP
0C6C'  C3 0299'                         JP      START           ; Volta ao Monitor

0C6F'  32 4014          BREAK2:         LD      (VAR_1),A       ; PP interno
0C72'  0E 07                            LD      C,7
0C74'  18 86                            JR      GO_ON           ; Volta ao normal

0C76'  3E 80            BREAK3:         LD      A,80H
0C78'  08                               EX      AF,AF'
0C79'  AF                               XOR     A
0C7A'  32 4010                          LD      (DUMP?),A       ; Mostra registradores
0C7D'  06 0C                            LD      B,12
0C7F'  DD 21 40C3                       LD      IX,BREAKPOINTS
0C83'  DD 7E 00         BREAK4:         LD      A,(IX+0)        ; Ativo ?
0C86'  E6 07                            AND     7
0C88'  28 0F                            JR      Z,BREAK5
0C8A'  DD 5E 02                         LD      E,(IX+2)        ; Endereco do PP
0C8D'  DD 56 03                         LD      D,(IX+3)
0C90'  2A 40BC                          LD      HL,(REG_PC)     ; Endereco corrente
```

```
0C93'  CD 045B'                    CALL   CMP_HL_DE      ; Igual ?
0C96'  CC 0CA2'                    CALL   Z,BREAK6       ; Sim, sinaliza
0C99'  11 0006       BREAK5:       LD     DE,6           ; Proximo PP
0C9C'  DD 19                       ADD    IX,DE
0C9E'  10 E3                       DJNZ   BREAK4
0CA0'  08                          EX     AF,AF'
0CA1'  C9                          RET

0CA2'  08            BREAK6:       EX     AF,AF'         ; Marca PP encontrado
0CA3'  CB BF                       RES    7,A
0CA5'  08                          EX     AF,AF'
0CA6'  DD 5E 04                    LD     E,(IX+4)       ; Numero de vezes
0CA9'  DD 56 05                    LD     D,(IX+5)
0CAC'  1B                          DEC    DE
0CAD'  7A                          LD     A,D            ; Igual a zero ?
0CAE'  B3                          OR     E
0CAF'  28 11                       JR     Z,BREAK7
0CB1'  DD 73 04                    LD     (IX+4),E       ; Nao, armazena valor
0CB4'  DD 72 05                    LD     (IX+5),D
0CB7'  DD CB 00                    BIT    4,(IX+0)       ; Bit "REPEAT"
0CBA'  66
0CBB'  C8                          RET    Z
0CBC'  3E 01                       LD     A,1            ; Nao mostra registr.
0CBE'  32 4010                     LD     (DUMP?),A
0CC1'  C9                          RET
0CC2'  08            BREAK7:       EX     AF,AF'
0CC3'  DD B6 00                    OR     (IX+0)         ; Flags do PP
0CC6'  08                          EX     AF,AF'
0CC7'  C9                          RET

0CC8'  06 0C         RESTORE_BREAK: LD     B,12           ; Restaura os pontos de parada
0CCA'  DD 21 40C3                  LD     IX,BREAKPOINTS
0CCE'  DD CB 00      RES_LOOP:     BIT    5,(IX+0)
0CD1'  6E
0CD2'  DD CB 00                    RES    5,(IX+0)
0CD5'  AE
0CD6'  28 0F                       JR     Z,NOT_BREAK    ; Ativo ?
0CD8'  DD 6E 02                    LD     L,(IX+2)       ; Endereco
0CDB'  DD 66 03                    LD     H,(IX+3)
0CDE'  3E FF                       LD     A,RESTART
0CE0'  BE                          CP     (HL)           ; Instrucao colocada ?
0CE1'  20 04                       JR     NZ,NOT_BREAK
0CE3'  DD 7E 01                    LD     A,(IX+1)       ; Sim, restaura intrucao
0CE6'  77                          LD     (HL),A
0CE7'  DD CB 00      NOT_BREAK:    RES    3,(IX+0)
0CEA'  9E
0CEB'  11 0006                     LD     DE,6           ; Proximo PP
0CEE'  DD 19                       ADD    IX,DE
0CF0'  10 DC                       DJNZ   RES_LOOP
0CF2'  C9                          RET

0CF3'  06 0C         CMD_GBREAK:   LD     B,12           ; Procura um ponto de parada
0CF5'  DD 21 40C3                  LD     IX,BREAKPOINTS
0CF9'  DD 7E 00      CMD_GBR1:     LD     A,(IX+0)
0CFC'  E6 03                       AND    3              ; Breakpoint ativo ?
0CFE'  28 0D                       JR     Z,CMD_GBR2
0D00'  DD 5E 02                    LD     E,(IX+2)       ; Ender. igual ?
0D03'  DD 56 03                    LD     D,(IX+3)
0D06'  2A 40BC                     LD     HL,(REG_PC)
0D09'  CD 045B'                    CALL   CMP_HL_DE
0D0C'  C8                          RET    Z              ; sim, retorna
0D0D'  11 0006       CMD_GBR2:     LD     DE,6           ; nao, avanca para o proximo
```

```
0D10'  DD 19                    ADD     IX,DE
0D12'  10 E5                    DJNZ    CMD_GBR1
0D14'  97                       SUB     A
0D15'  3C                       INC     A
0D16'  C9                       RET

0D17'  CD 0B89'      BRK_TMP:   CALL    BREAK_FREE      ; Estabelece um PP temporario
0D1A'  DD 36 04                 LD      (IX+4),1
0D1D'  01
0D1E'  DD 36 05                 LD      (IX+5),0        ; Apenas uma vez
0D21'  00
0D22'  DD 75 02                 LD      (IX+2),L        ; Endereco corrente
0D25'  DD 74 03                 LD      (IX+3),H
0D28'  3A 4014                  LD      A,(VAR_1)       ; PP no endereco do PC
0D2B'  A7                       AND     A
0D2C'  3E 08                    LD      A,8             ; Assume igual ao PC
0D2E'  20 02                    JR      NZ,BRK_TMP1
0D30'  3E 04                    LD      A,4             ; Use PP Breve
0D32'  DD 77 00      BRK_TMP1:  LD      (IX+0),A
0D35'  C9                       RET

0D36'  06 0C         SET_BRK:   LD      B,12            ; Coloca o PP
0D38'  DD 21 40C3               LD      IX,BREAKPOINTS
0D3C'  DD 7E 00      SET_BRK1:  LD      A,(IX+0)
0D3F'  A1                       AND     C
0D40'  28 10                    JR      Z,SET_BRK2      ; Ativo ?
0D42'  DD CB 00                 SET     5,(IX+0)        ; PP Armado
0D45'  EE
0D46'  DD 6E 02                 LD      L,(IX+2)
0D49'  DD 66 03                 LD      H,(IX+3)        ; Endereco
0D4C'  7E                       LD      A,(HL)          ; Codigo original
0D4D'  DD 77 01                 LD      (IX+1),A        ;  Salve-o
0D50'  36 FF                    LD      (HL),RESTART    ; Coloque o RST
0D52'  11 0006       SET_BRK2:  LD      DE,6            ; Proximo PP
0D55'  DD 19                    ADD     IX,DE
0D57'  10 E3                    DJNZ    SET_BRK1
0D59'  C9                       RET

0D5A'                +CMD_L:    GET_INT                 ; Comando List, obtem um endereco
0D5B'  30 03                    JR      NC,CMD_L1       ; Se nao fornecido,
0D5D'  2A 401B                  LD      HL,(POSICAO)    ;       use posicao atual
0D60'  E5            CMD_L1:    PUSH    HL
0D61'  FD E1                    POP     IY
0D63'  CD 0434'                 CALL    SKIP_COMMA      ; ','
0D66'  CD 0502'                 CALL    GET_SIZE        ; Tamanho ?
0D69'  30 1F                    JR      NC,CMD_L3
0D6B'  CD 0452'                 CALL    END_CMD         ; Fim
0D6E'  06 10                    LD      B,16            ; 16 bytes de cada vez
0D70'  C5            CMD_L2:    PUSH    BC
0D71'  FD E5                    PUSH    IY
0D73'  E1                       POP     HL
0D74'  E5                       PUSH    HL
0D75'  CD 0DB2'                 CALL    CMD_LIST        ; Lista
0D78'  CD 030A'                 CALL    CR_LF           ; Muda de linha
0D7B'  FD E1                    POP     IY
0D7D'  48                       LD      C,B
0D7E'  06 00                    LD      B,0
0D80'  FD 09                    ADD     IY,BC           ; Proximo
0D82'  FD 22 401B               LD      (POSICAO),IY    ; Posicao atual e' a corrente agora
0D86'  C1                       POP     BC
0D87'  10 E7                    DJNZ    CMD_L2          ; repete ate' o fim
0D89'  C9                       RET
```

```
0D8A'  CD 0452'     CMD_L3:     CALL   END_CMD        ; Fim do comando
0D8D'  60                       LD     H,B
0D8E'  69                       LD     L,C
0D8F'  78                       LD     A,B
0D90'  B1                       OR     C
0D91'  20 01                    JR     NZ,CMD_L4
0D93'  2B                       DEC    HL
0D94'  E5           CMD_L4:     PUSH   HL
0D95'  FD E5                    PUSH   IY
0D97'  E1                       POP    HL
0D98'  E5                       PUSH   HL
0D99'  CD 0DB2'                 CALL   CMD_LIST       ; Lista ate' o fim
0D9C'  CD 030A'                 CALL   CR_LF          ; Muda de linha
0D9F'  FD E1                    POP    IY
0DA1'  58                       LD     E,B
0DA2'  16 00                    LD     D,0
0DA4'  FD 19                    ADD    IY,DE
0DA6'  FD 22 401B               LD     (POSICAO),IY   ; Posicao atual
0DAA'  E1                       POP    HL
0DAB'  A7                       AND    A
0DAC'  ED 52                    SBC    HL,DE
0DAE'  C8                       RET    Z
0DAF'  D8                       RET    C              ; Fim ?
0DB0'  18 E2                    JR     CMD_L4

0DB2'  CD 03B2'     CMD_LIST:   CALL   OUT_HL@        ; Mostra endereco
0DB5'  CC 0409'                 CALL   Z,WHITE        ; Brancos
0DB8'  CD 0409'                 CALL   WHITE
0DBB'  AF                       XOR    A
0DBC'  32 4002                  LD     (CNT_CAR),A    ; Contador de caracteres
0DBF'  CD 0DE4'                 CALL   LIST_IY        ; Desmonta instrucao
0DC2'  A7                       AND    A
0DC3'  C8                       RET    Z
0DC4'  CD 0409'     CMD_LIST1:  CALL   WHITE          ; Branco
0DC7'  3A 4002                  LD     A,(CNT_CAR)    ; Para na posicao mais 16
0DCA'  FE 10                    CP     16
0DCC'  38 F6                    JR     C,CMD_LIST1
0DCE'  ED 5B 401E   CMD_LIST2:  LD     DE,(VAR_@)     ; Relocacao ?
0DD2'  7A                       LD     A,D
0DD3'  B3                       OR     E
0DD4'  C8                       RET    Z
0DD5'  3E 28                    LD     A,'('          ; Sim, imprime endereco relocado
                    +           OUTPUT
0DD8'  A7                       AND    A
0DD9'  ED 52                    SBC    HL,DE
                    +           OUT_HL                ; "(ENDERECO')"
0DDC'  CD 03CB'                 CALL   OUT_ASP
0DDF'  3E 29                    LD     A,')'
0DE1'  C3 040B'                 JP     OUTPUT

0DE4'  AF           LIST_IY:    XOR    A              ; Desmonta a instrucao apontada por
0DE5'  32 401A                  LD     (VAR_T3),A     ;  IY
0DE8'  CD 0E06'                 CALL   LISTIY         ; Procura nas tabelas
0DEB'  30 11                    JR     NC,LIST_IY1
0DED'  C5                       PUSH   BC
0DEE'  CD 10D2'                 CALL   PRINT_MENEMO   ; Imprime menemonico
0DF1'  EB                       EX     DE,HL
0DF2'  CD 02D0'                 CALL   JUMP           ; Executa rotina de tratamento dos
0DF5'  C1                       POP    BC             ;  Operandos
0DF6'  3A 401A                  LD     A,(VAR_T3)
0DF9'  2A 4018                  LD     HL,(VAR_T2)
```

```
0DFC'  37                        SCF
0DFD'  C9                        RET
0DFE'  21 0023'    LIST_IY1:     LD      HL,NOT_CMD      ; Instrucao nao encontrada
                   +             PRINT
0E02'  06 01                     LD      B,1
0E04'  97                        SUB     A
0E05'  C9                        RET

0E06'  AF          LISTIY:       XOR     A               ; Analiza instrucao apontada po IY
0E07'  32 4017                   LD      (VAR_T1),A
0E0A'  FD 7E 00                  LD      A,(IY+0)        ; Pega a instrucao
0E0D'  FE ED                     CP      0EDH            ; Grupo ED
0E0F'  CA 0EBA'                  JP      Z,GRUPO_ED
0E12'  FE DD                     CP      0DDH            ; Grupo DD
0E14'  28 31                     JR      Z,GRUPO_DD
0E16'  FE FD                     CP      0FDH            ; Grupo FD
0E18'  28 31                     JR      Z,GRUPO_FD
0E1A'  FD 7E 00    LISTIY1:      LD      A,(IY+0)
0E1D'  FE CB                     CP      0CBH            ; Grupo CB
0E1F'  CA 0EA7'                  JP      Z,GRUPO_CB
0E22'  21 1BBA'                  LD      HL,TABELA6      ; Procura no grupo de 2 bytes
0E25'  CD 0EE3'                  CALL    CHKTAB_INSTR
0E28'  06 02                     LD      B,2
0E2A'  D8                        RET     C
0E2B'  21 1B1C'                  LD      HL,TABELA4      ; Procura no grupo de 1 byte
0E2E'  CD 0EBE'                  CALL    PROC_MEN
0E31'  06 01                     LD      B,1
0E33'  D8                        RET     C
0E34'  21 1B37'                  LD      HL,TABELA5      ; Procura no resto do grupo
0E37'  CD 0EE3'                  CALL    CHKTAB_INSTR
0E3A'  06 01                     LD      B,1
0E3C'  D8                        RET     C
0E3D'  21 1C06'                  LD      HL,TABELA7      ; Procura no grupo de 3 bytes
0E40'  CD 0EE3'                  CALL    CHKTAB_INSTR
0E43'  D0                        RET     NC
0E44'  06 03                     LD      B,3
0E46'  C9                        RET

0E47'  3E 01       GRUPO_DD:     LD      A,1             ; Assume grupo DD
0E49'  18 02                     JR      GRUPO_DDFD
0E4B'  3E 02       GRUPO_FD:     LD      A,2             ; Assume Grupo FD
0E4D'  32 4017     GRUPO_DDFD:   LD      (VAR_T1),A
0E50'  CD 0E5D'                  CALL    GRUPO_FD_DD     ; Procura, FD e DD tem bytes identicos
0E53'  D0                        RET     NC              ; Nao achou
0E54'  C5                        PUSH    BC
0E55'  CD 0E1A'                  CALL    LISTIY1         ; Analiza o resto
0E58'  F1                        POP     AF
0E59'  80                        ADD     A,B
0E5A'  47                        LD      B,A             ; Tamanho total da instr.
0E5B'  37                        SCF
0E5C'  C9                        RET

0E5D'  FD 23       GRUPO_FD_DD:  INC     IY
0E5F'  21 1AF2'                  LD      HL,TABELA2      ; Procura na tabela com deslocamento
0E62'  CD 0ED8'                  CALL    PROC_BYTE
0E65'  06 02                     LD      B,2             ; 2 bytes
0E67'  D8                        RET     C
0E68'  21 1B0D'                  LD      HL,TABELA3      ; Procura na tabela com ou sem endereco
0E6B'  CD 0ED8'                  CALL    PROC_BYTE
0E6E'  06 01                     LD      B,1             ; 1 byte
0E70'  D8                        RET     C
0E71'  FD 7E 00                  LD      A,(IY+0)
```

```
0E74'  FE CB                    CP    0CBH          ; Subgrupo CB ?
0E76'  20 10                    JR    NZ,GRP_FD_DD
0E78'  FD 7E 02                 LD    A,(IY+2)
0E7B'  FE 36                    CP    036H          ; Exclui 36h
0E7D'  C8                       RET   Z
0E7E'  E6 07                    AND   7
0E80'  FE 06                    CP    06H           ; Subgrupo correto ?
0E82'  20 04                    JR    NZ,GRP_FD_DD
0E84'  06 02                    LD    B,2           ; Sim, 2 bytes
0E86'  37                       SCF
0E87'  C9                       RET
0E88'  A7           GRP_FD_DD:  AND   A             ; Nao achou
0E89'  C9                       RET

0E8A'  FD 23        GRUPO_ED:   INC   IY            ; Grupo ED
0E8C'  21 1C34'                 LD    HL,TABELA8    ; 2 bytes
0E8F'  CD 0EBE'                 CALL  PROC_MEN
0E92'  06 02                    LD    B,2
0E94'  D8                       RET   C
0E95'  21 1C5F'                 LD    HL,TABELA9    ; 2 bytes
0E98'  CD 0EE3'                 CALL  CHKTAB_INSTR
0E9B'  06 02                    LD    B,2
0E9D'  D8                       RET   C
0E9E'  21 1CAB'                 LD    HL,TABEL10    ; 4 bytes
0EA1'  CD 0EE3'                 CALL  CHKTAB_INSTR
0EA4'  06 04                    LD    B,4
0EA6'  C9                       RET

0EA7'  FD E5        GRUPO_CB:   PUSH  IY            ; Grupo CB
0EA9'  FD 23                    INC   IY
0EAB'  3A 4017                  LD    A,(VAR_T1)
0EAE'  A7                       AND   A
0EAF'  28 02                    JR    Z,GRUPO_CB_1
0EB1'  FD 23                    INC   IY
0EB3'  21 1CC0'     GRUPO_CB_1: LD    HL,TABEL11    ; Tabela grupo CB
0EB6'  CD 0EE3'                 CALL  CHKTAB_INSTR
0EB9'  FD E1                    POP   IY
0EBB'  06 02                    LD    B,2           ; 2 bytes
0EBD'  C9                       RET

0EBE'  7E           PROC_MEN:   LD    A,(HL)
0EBF'  FE FF                    CP    0FFH          ; Fim da tabela
0EC1'  C8                       RET   Z
0EC2'  FD BE 00                 CP    (IY+0)        ; Achou ?
0EC5'  28 04                    JR    Z,PROC_MEN1
0EC7'  23                       INC   HL            ; Proxima entrada da tabela
0EC8'  23                       INC   HL
0EC9'  18 F3                    JR    PROC_MEN
0ECB'  23           PROC_MEN1:  INC   HL
0ECC'  4E                       LD    C,(HL)        ; Obtem deslocamento
0ECD'  21 1D8C'                 LD    HL,INSTR
0ED0'  06 00                    LD    B,0
0ED2'  09                       ADD   HL,BC         ; HL => Menemonico
0ED3'  11 0ED7'                 LD    DE,PROC_MEN2  ; Sem rotina de tratamento
0ED6'  37                       SCF
0ED7'  C9           PROC_MEN2:  RET

0ED8'  7E           PROC_BYTE:  LD    A,(HL)        ; Procura byte
0ED9'  A7                       AND   A
0EDA'  C8                       RET   Z
0EDB'  23                       INC   HL
0EDC'  FD BE 00                 CP    (IY+0)        ; Achou ?
```

```
0EDF'  20 F7                     JR      NZ,PROC_BYTE    ; Continua
0EE1'  37                        SCF                     ; CARRY ligada se achou
0EE2'  C9                        RET

0EE3'  FD 7E 00      CHKTAB_INSTR: LD     A,(IY+0)        ; Obtem instrucao
0EE6'  A6                        AND     (HL)            ; Efetua um AND com o primeiro byte
0EE7'  23                        INC     HL              ;       da tabela
0EE8'  BE                        CP      (HL)            ; Compara com o segundo
0EE9'  28 09                     JR      Z,CHKTAB_INS1   ; Igual ?
0EEB'  23                        INC     HL              ; Nao achou, avanca para a
0EEC'  23                        INC     HL              ;       proxima entrada da tabela
0EED'  23                        INC     HL
0EEE'  23                        INC     HL
0EEF'  7E                        LD      A,(HL)          ; Fim da tabela ?
0EF0'  A7                        AND     A
0EF1'  20 F0                     JR      NZ,CHKTAB_INSTR
0EF3'  C9                        RET
0EF4'  23            CHKTAB_INS1: INC    HL              ; Achou
0EF5'  4E                        LD      C,(HL)          ; Reg. C = deslocamento da tabela de
0EF6'  23                        INC     HL              ;       menemonicos
0EF7'  5E                        LD      E,(HL)          ; DE = Rotina de tratamento
0EF8'  23                        INC     HL              ;       dos parametros
0EF9'  56                        LD      D,(HL)
0EFA'  21 1D8C'                  LD      HL,INSTR        ; Tabela de menemonicos
0EFD'  06 00                     LD      B,0
0EFF'  09                        ADD     HL,BC           ; HL = Aponta para o menemonico
0F00'  37                        SCF                     ; Sinaliza que achou
0F01'  C9                        RET

0F02'  CD 1043'      ANAL_RS:    CALL    ANALBIT5_3      ; Analize reg. R,S ( Ex.: LD R,S )
0F05'  CD 10CD'                  CALL    PRT_VRG
0F08'  18 03                     JR      ANAL_AR1

0F0A'  CD 10CA'      ANAL_AR:    CALL    PRT_ACCUM               ; Ex.: LD A,R
0F0D'  C3 104B'      ANAL_AR1:   JP      ANALBIT2_0

0F10'  FD 7E 00      ANAL_B:     LD      A,(IY+0)                ; Analize Bit (Ex.: Bit B,R )
0F13'  E6 38                     AND     38H
0F15'  C3 03EB'                  JP      OUT_A

0F18'  21 1D72'      ANAL_SP_C:  LD      HL,INSTR4       ; '(SP),HL'
0F1B'  18 13                     JR      ANAL__HL

0F1D'  3E 28         ANAL_HL_C:  LD      A,'('           ; '(HL)'
                     +           OUTPUT
0F20'  CD 1093'                  CALL    ANAL_HL
0F23'  3E 29                     LD      A,')'
0F25'  C3 040B'                  JP      OUTPUT

0F28'  21 1D57'      ANAL_DE_HL: LD      HL,INSTR3       ; 'DE,HL'
0F2B'  18 0A                     JR      ANAL_AF_1

0F2D'  21 1D77'      ANAL_SP_HL: LD      HL,INSTR5       ; 'SP,HL'
0F30'                +ANAL__HL:  PRINT
0F31'  C3 1093'                  JP      ANAL_HL

0F34'  21 1D51'      ANAL_AF_AF: LD      HL,INSTR2       ; "AF,AF'"
0F37'  C3 041C'      ANAL_AF_1:  JP      PRINT

0F3A'  CD 1093'      ANAL_HL_DD: CALL    ANAL_HL         ; 'HL,DD'
0F3D'  CD 10CD'                  CALL    PRT_VRG
0F40'  C3 10A6'                  JP      ANAL_DD
```

```
0F43'  CD 0F4B'     ANAL_DD_C_A:  CALL  ANAL_DD_C      ; '(DD),A'
0F46'  18 48                      JR    ANAL_N_A1

0F48'  CD 10CA'     ANAL_A_DD_C:  CALL  PRT_ACCUM      ; 'A,(DD)'
0F4B'  3E 28        ANAL_DD_C:    LD    A,'('
                    +             OUTPUT
0F4E'  CD 10A6'                   CALL  ANAL_DD
0F51'  3E 29                      LD    A,')'
0F53'  C3 040B'                   JP    OUTPUT

0F56'  CD 10CA'     ANAL_A_N:     CALL  PRT_ACCUM      ; 'A,N'
0F59'  18 0F                      JR    ANAL_N

0F5B'  CD 1043'     ANAL_RN:      CALL  ANALBIT5_3     ; 'R,N'
0F5E'  CD 10CD'                   CALL  PRT_VRG
0F61'  3A 4017                    LD    A,(VAR_T1)
0F64'  A7                         AND   A
0F65'  FD 7E 02                   LD    A,(IY+2)
0F68'  20 03                      JR    NZ,ANAL_N1
0F6A'  FD 7E 01     ANAL_N:       LD    A,(IY+1)
0F6D'  C3 03EB'     ANAL_N1:      JP    OUT_A

0F70'  FD 7E 00     ANAL_CC_JR:   LD    A,(IY+0)       ; 'JR cc,Label'
0F73'  E6 18                      AND   18H
0F75'  CD 10BB'                   CALL  ANAL_CC_A
0F78'  CD 10CD'                   CALL  PRT_VRG
0F7B'  FD 4E 01     ANAL_ADD_REL: LD    C,(IY+1)       ; Analize endereco relativo
0F7E'  79                         LD    A,C            ; Ex.: JR Label
0F7F'  17                         RLA
0F80'  9F                         SBC   A,A
0F81'  47                         LD    B,A
0F82'  FD E5                      PUSH  IY
0F84'  E1                         POP   HL
0F85'  09                         ADD   HL,BC
0F86'  23                         INC   HL
0F87'  23                         INC   HL
0F88'  18 1A                      JR    ANAL_ENDER1

0F8A'  FD 7E 01     ANAL_N_A:     LD    A,(IY+1)       ; 'N,A'
0F8D'  CD 03EB'                   CALL  OUT_A
0F90'  CD 10CD'     ANAL_N_A1:    CALL  PRT_VRG
0F93'  3E 41                      LD    A,'A'
0F95'  C3 040B'                   JP    OUTPUT

0F98'  CD 10BB'     ANAL_CC_END:  CALL  ANAL_CC        ; 'CC,ENDER' Ex.: JP cc,Label
0F9B'  CD 10CD'                   CALL  PRT_VRG
0F9E'  FD 6E 01     ANAL_ENDER:   LD    L,(IY+1)
0FA1'  FD 66 02                   LD    H,(IY+2)
0FA4'  3E 02        ANAL_ENDER1:  LD    A,2
0FA6'  32 401A      ANAL_ENDER2:  LD    (VAR_T3),A
0FA9'  22 4018                    LD    (VAR_T2),HL
0FAC'  C3 03E6'                   JP    OUT_HL

0FAF'  CD 10A6'     ANAL_DD_VAL16: CALL ANAL_DD        ; 'DD,VALOR16'
0FB2'  CD 10CD'                    CALL PRT_VRG
0FB5'  18 E7                       JR   ANAL_ENDER

0FB7'  CD 1029'     ANAL_END_CC_A: CALL ANAL_END_C     ; '(ENDERECO),A'
0FBA'  18 D4                       JR   ANAL_N_A1

0FBC'  3E 30        ANAL_IM0:     LD    A,'0'          ; 'IM 0'
```

```
0FBE'  18 06                     JR      ANAL_OUTPUT

0FC0'  3E 31        ANAL_IM1:    LD      A,'1'          ; 'IM 1'
0FC2'  18 02                     JR      ANAL_OUTPUT

0FC4'  3E 32        ANAL_IM2:    LD      A,'2'          ; 'IM 2'
0FC6'  C3 040B'     ANAL_OUTPUT: JP      OUTPUT

0FC9'  21 1D7A'     ANAL_I_A:    LD      HL,INSTR6      ; 'I,A'
0FCC'  18 18                     JR      ANAL_PRINT

0FCE'  21 1D7D'     ANAL_A_I:    LD      HL,INSTR7      ; 'A,I'
0FD1'  18 13                     JR      ANAL_PRINT

0FD3'  21 1D80'     ANAL_R_A:    LD      HL,INSTR8      ; 'R,A'
0FD6'  18 0E                     JR      ANAL_PRINT

0FD8'  21 1D83'     ANAL_A_R:    LD      HL,INSTR9      ; 'A,R'
0FDB'  18 09                     JR      ANAL_PRINT

0FDD'  CD 1043'     ANAL_R_C_C:  CALL    ANALBIT5_3     ; 'R,(C)'
0FE0'  CD 10CD'                  CALL    PRT_VRG
0FE3'  21 1EBE'                  LD      HL,IO_P_C
0FE6'  C3 041C'     ANAL_PRINT:  JP      PRINT

0FE9'  CD 10A6'     ANAL_DD_END_C: CALL  ANAL_DD        ; 'DD,(ENDERECO)'
0FEC'  CD 10CD'                  CALL    PRT_VRG
0FEF'  18 38                     JR      ANAL_END_C

0FF1'  3A 4017      ANAL_B_R:    LD      A,(VAR_T1)     ; 'B,R' Ex.: Bit b,r
0FF4'  A7                        AND     A
0FF5'  20 05                     JR      NZ,ANAL_B_R1
0FF7'  FD 7E 01                  LD      A,(IY+1)
0FFA'  18 03                     JR      ANAL_B_R2
0FFC'  FD 7E 02     ANAL_B_R1:   LD      A,(IY+2)
0FFF'  F5           ANAL_B_R2:   PUSH    AF
1000'  1F                        RRA
1001'  1F                        RRA
1002'  1F                        RRA
1003'  E6 07                     AND     7
1005'  C6 30                     ADD     A,'0'
                   +             OUTPUT
1008'  CD 10CD'                  CALL    PRT_VRG
1008'  F1                        POP     AF
100C'  18 40                     JR      ANALBIT2_0_A

100E'  3A 4017      ANAL_REG:    LD      A,(VAR_T1)
1011'  A7                        AND     A
1012'  20 05                     JR      NZ,ANAL_REG1
1014'  FD 7E 01                  LD      A,(IY+1)
1017'  18 03                     JR      ANAL_REG2
1019'  FD 7E 02     ANAL_REG1:   LD      A,(IY+2)
101C'  18 30        ANAL_REG2:   JR      ANALBIT2_0_A

101E'  CD 1093'     ANAL_HL_END_C: CALL  ANAL_HL        ; 'HL,(ENDERECO)
1021'  CD 10CD'                  CALL    PRT_VRG
1024'  18 03                     JR      ANAL_END_C

1026'  CD 10CA'     ANAL_A_ENDER_C: CALL PRT_ACCUM      ; 'A,(ENDERECO)'

1029'  3E 28        ANAL_END_C:  LD      A,'('          ; '(ENDERECO)'
                   +             OUTPUT
```

```
102C'  FD 6E 01                    LD      L,(IY+1)
102F'  FD 66 02                    LD      H,(IY+2)
1032'  3E 01                       LD      A,1
1034'  CD 0FA6'                    CALL    ANAL_ENDER2
1037'  3E 29                       LD      A,')'
1039'  C3 040B'                    JP      OUTPUT

103C'  21 1EBE'      ANAL_C_C_R:   LD      HL,IO_P_C       ; '(C),R'
                     +             PRINT
1040'  CD 10CD'                    CALL    PRT_VRG

1043'  FD 7E 00      ANALBIT5_3:   LD      A,(IY+0)        ; Analiza bits 5 a 3, registrador
1046'  1F                          RRA
1047'  1F                          RRA
1048'  1F                          RRA
1049'  18 03                       JR      ANALBIT2_0_A
104B'  FD 7E 00      ANALBIT2_0:   LD      A,(IY+0)        ; Analiza bits 2 a 0
104E'  E6 07          ANALBIT2_0_A: AND     7               : Analiza bits 2 a 0 no acumulador
1050'  FE 06                       CP      6               ; (HL) ?
1052'  20 34                       JR      NZ,ANALBT4      ; Procura na tabela de registradores
1054'  3A 4017                     LD      A,(VAR_T1)
1057'  A7                          AND     A
1058'  3E 06                       LD      A,6
105A'  28 2C                       JR      Z,ANALBT4               : Tabela de registradores
105C'  21 1D86'                    LD      HL,INST10       ; Instrucoes com IX
105F'  3A 4017                     LD      A,(VAR_T1)
1062'  3D                          DEC     A
1063'  28 03                       JR      Z,ANALBT1
1065'  21 1D89'                    LD      HL,INST11       ; Instrucoes com IY
1068'                +ANALBT1:     PRINT
1069'  FD 7E 01                    LD      A,(IY+1)        ; Segundo operando
106C'  A7                          AND     A
106D'  FA 1078'                    JP      M,ANALBT2
1070'  3E 2B                       LD      A,'+'           ; Positivo, (IX+n) ou (IY+n)
                     +             OUTPUT
1073'  FD 7E 01                    LD      A,(IY+1)
1076'  18 08                       JR      ANALBT3
1078'  3E 2D          ANALBT2:      LD      A,'-'           ; Negativo, (IX-n) ou (IY-n)
                     +             OUTPUT
107B'  FD 7E 01                    LD      A,(IY+1)
107E'  ED 44                       NEG
1080'  CD 03EB'       ANALBT3:      CALL    OUT_A
1083'  3E 29                       LD      A,')'           ; ')'
1085'  C3 040B'                    JP      OUTPUT

1088'  21 1E60'       ANALBT4:      LD      HL,REGS1        ; Procura na tabela de registradores
108B'  18 36                       JR      LOC_TAB_A

108D'  CD 1029'       ANAL_END_C_HL: CALL   ANAL_END_C      ; '(ENDERECO),HL'
1090'  CD 10CD'                    CALL    PRT_VRG

1093'  3A 4017        ANAL_HL:      LD      A,(VAR_T1)
1096'  21 1E90'                    LD      HL,REGS7        ; Registrador HL
1099'  18 28                       JR      LOC_TAB_A

109B'  21 1E75'       ANAL_DD_DD:   LD      HL,REGS3        ; 'DD,DD'
109E'  18 09                       JR      ANAL_DD1

10A0'  CD 1029'       ANAL_END_C_DD: CALL   ANAL_END_C      ; '(ENDERECO),DD'
10A3'  CD 10CD'                    CALL    PRT_VRG

10A6'  21 1E6C'       ANAL_DD:      LD      HL,REGS2        ; Analize de pares de registradores
```

```
10A9'  FD 7E 00     ANAL_DD1:    LD     A,(IY+0)
10AC'  1F                        RRA
10AD'  1F                        RRA
10AE'  1F                        RRA
10AF'  1F                        RRA
10B0'  E6 03                     AND    3
10B2'  FE 02                     CP     2              ; HL ?
10B4'  28 DD                     JR     Z,ANAL_HL
10B6'  18 0B                     JR     LOC_TAB_A

10B8'  FD 7E 00     ANAL_CC:     LD     A,(IY+0)       ; Analize codigo de condicao
10BB'  1F           ANAL_CC_A:   RRA                    ;    "     no acumulador
10BC'  1F                        RRA
10BD'  1F                        RRA
10BE'  E6 07                     AND    7
10C0'  21 1EA6'                  LD     HL,FLAGS2
10C3'  47           LOC_TAB_A:   LD     B,A            ; Localize na tabela o indexe (A)
10C4'  CD 04BF'                  CALL   LOC_TAB
10C7'  C3 041C'                  JP     PRINT

10CA'  3E 41        PRT_ACCUM:   LD     A,'A'          ; Imprime o Acumulador
                    +            OUTPUT
10CD'  3E 2C        PRT_VRG:     LD     A,','          ; Imprime uma virgula
10CF'  C3 040B'                  JP     OUTPUT

10D2'               +PRINT_MENEMO: PRINT          ; Imprime o menemonico
10D3'  CD 0409'     PRINT_MENEMO1: CALL   WHITE   ; Completa 5 caracteres
10D6'  0C                        INC    C
10D7'  79                        LD     A,C
10D8'  FE 05                     CP     5
10DA'  20 F7                     JR     NZ,PRINT_MENEMO1
10DC'  C9                        RET

10DD'  FD 2A 40BC   NXT_INSTR:   LD     IY,(REG_PC)    ; Verifica a proxima instrucao
10E1'  CD 0E06'                  CALL   LISTIY         ; Instrucao Executavel ?
10E4'  D2 02C8'                  JP     NC,ERROR
10E7'  48                        LD     C,B            ; Tamanho da instrucao
10E8'  06 00                     LD     B,0
10EA'  2A 40BC                   LD     HL,(REG_PC)
10ED'  09                        ADD    HL,BC          ; Coloca um PP apos a instrucao
10EE'  CD 0D17'                  CALL   BRK_TMP        ;  Break Point breve
10F1'  FD 2A 40BC                LD     IY,(REG_PC)
10F5'  21 1CF3'                  LD     HL,TABEL12
10F8'  CD 0EE3'                  CALL   CHKTAB_INSTR   ; Procura Instr. em tabela
10FB'  3F                        CCF
10FC'  D8                        RET    C
10FD'  EB                        EX     DE,HL
10FE'  CD 02D0'                  CALL   JUMP           ; Executa a rotina de tratamento
1101'  DC 0D17'                  CALL   C,BRK_TMP      ; Estabelece um PP se necessario
1104'  37                        SCF
1105'  C9                        RET

1106'  3A 4014      ANAL_CALL:   LD     A,(VAR_1)      ; Tratamento da instrucao CALL
1109'  A7                        AND    A
110A'  20 05                     JR     NZ,ANAL_JUMP
110C'  3A 4011                   LD     A,(TRACE?)
110F'  A7                        AND    A
1110'  C0                        RET    NZ
1111'  FD 6E 01      ANAL_JUMP:   LD     L,(IY+1)       ; Tratamento da instrucao JUMP
1114'  FD 66 02                  LD     H,(IY+2)       ;  endereco
1117'  37                        SCF
1118'  C9                        RET
```

```
1119'  FD 4E 01     ANAL_JR:        LD    C,(IY+1)      ; Tratamento JR e DJNZ
111C'  79                           LD    A,C           ;  deslocamento
111D'  17                           RLA
111E'  9F                           SBC   A,A
111F'  47                           LD    B,A
1120'  2A 40BC                      LD    HL,(REG_PC)   ; Soma com o endereco corrente
1123'  09                           ADD   HL,BC
1124'  23                           INC   HL
1125'  23                           INC   HL
1126'  37                           SCF
1127'  C9                           RET

1128'  2A 40B6     ANAL_JUMP_HL:   LD    HL,(REG_HL)   ; Tratamento JUMP (HL)
112B'  37                           SCF
112C'  C9                           RET

112D'  2A 40AE     ANAL_IX:        LD    HL,(REG_IX)   ; Tratamento do grupo DD (IX)
1130'  18 03                        JR    ANAL_IX_IY

1132'  2A 40AC     ANAL_IY:        LD    HL,(REG_IY)   ; Tratamento do grupo FD (IY)
1135'  FD 7E 01    ANAL_IX_IY:     LD    A,(IY+1)      ; JUMP (IX) ou JUMP (IY)
1138'  FE E9                        CP    0E9H
113A'  37                           SCF
113B'  C8                           RET   Z
113C'  A7                           AND   A
113D'  C9                           RET

113E'  FD 7E 01    ANAL_ED:        LD    A,(IY+1)      ; Tratamento do grupo ED
1141'  FE 4D                        CP    4DH           ; RETI
1143'  28 16                        JR    Z,ANAL_RET
1145'  FE 45                        CP    45H           ; RETN
1147'  28 12                        JR    Z,ANAL_RET
1149'  A7                           AND   A
114A'  C9                           RET

114B'  FD 7E 00    ANAL_RET_C:     LD    A,(IY+0)      ; Tratamento dos RET condicionais
114E'  32 40BE                      LD    (ROT_RAM),A   ;  Caso o RET condicional seja
1151'  2A 40B0                      LD    HL,(REG_F)    ;  verdadeiro, um PP e' estabelecido
1154'  E5                           PUSH  HL            ;  no endereco de retorno, caso
1155'  F1                           POP   AF            ;  contrario o PP nao e' estabelecido
1156'  CD 40BE                      CALL  ROT_RAM
1159'  37                           SCF
115A'  D0                           RET   NC
115B'  3A 4014     ANAL_RET:       LD    A,(VAR_1)     ; Tratamento do RET
115E'  A7                           AND   A
115F'  20 0B                        JR    NZ,PP_RET
1161'  3A 4011                      LD    A,(TRACE?)
1164'  A7                           AND   A
1165'  28 05                        JR    Z,PP_RET
1167'  CD 116C'                     CALL  PP_RET
116A'  E1                           POP   HL
116B'  C9                           RET

116C'  2A 40B8     PP_RET:         LD    HL,(REG_SP)   ; Estabelece PP no endereco de
116F'  5E                           LD    E,(HL)        ;  retorno
1170'  23                           INC   HL
1171'  56                           LD    D,(HL)        ; PP <= (SP)
1172'  EB                           EX    DE,HL
1173'  CD 0D17'                     CALL  BRK_TMP
1176'  A7                           AND   A
1177'  C9                           RET
```

```
1178'                   ANAL_RST:                               ; Tratamento dos RST
1178'  A7               ANAL_JSYS:      AND     A               ; Tratamento do JSYS
1179'  C9                              RET

117A'  3E 01            CMD_C:          LD      A,1             ; Comando Call
117C'  18 01                            JR      CMD_T1

117E'  97               CMD_T:          SUB     A               ; Comando Trace
117F'  32 4011          CMD_T1:         LD      (TRACE?),A
1182'  1A                               LD      A,(DE)
1183'  D6 4E                           SUB     'N'             ; Opcao 'N' ? Nao mostra reg.
1185'  20 01                           JR      NZ,CMD_T2
1187'  13                               INC     DE
1188'  32 4015          CMD_T2:         LD      (FLAG_N),A
118B'  1A                               LD      A,(DE)
118C'  D6 4A                           SUB     'J'             ; Opcao 'J' ? Jump
118E'  20 01                           JR      NZ,CMD_T3
1190'  13                               INC     DE
1191'  32 4016          CMD_T3:         LD      (FLAG_J),A
1194'  21 0001                         LD      HL,1            ; Numero de execucoes, "default" = 1
1197'  CD 044C'                        CALL    DEF_END
119A'  22 4012                         LD      (CNT_TRACE),HL
119D'  97                              SUB     A
119E'  32 4010                         LD      (DUMP?),A       ; Assume nao mostrar REGs.
11A1'  CD 10DD'         CMD_T4:        CALL    NXT_INSTR       ; Verifica a instrucao e estabelece PP
11A4'  C3 0BE0'                        JP      GO_PRG          ; Executa

                        ;       A execucao retorna aqui!

11A7'  CD 0BAE'         CMD_TRACE:     CALL    SLEEP_BREAK     ; Desarma PPs
11AA'  3A 4016                         LD      A,(FLAG_J)      ; Flag J ativa ?
11AD'  A7                              AND     A
11AE'  20 11                           JR      NZ,CMD_TRAC1
11B0'  FD 2A 40BC                      LD      IY,(REG_PC)     ; Sim
11B4'  CD 11E1'                        CALL    CHK_GRP         ; Verifica nos grupos
11B7'  28 08                           JR      Z,CMD_TRAC1
11B9'  21 1D02'                        LD      HL,TABEL13      ; Verifica as instrucoes de mudanca
11BC'  CD 0EE3'                        CALL    CHKTAB_INSTR    ;       de fluxo
11BF'  30 E0                           JR      NC,CMD_T4       ; Tudo bem, continue a execucao
                        ;
                        ;       Instrucoes de mudanca de fluxo

11C1'  2A 4012          CMD_TRAC1:     LD      HL,(CNT_TRACE)  ; Decrementa contador de trace
11C4'  2B                              DEC     HL
11C5'  22 4012                         LD      (CNT_TRACE),HL
11C8'  7C                              LD      A,H
11C9'  B5                              OR      L
11CA'  CA 0C66'                        JP      Z,STOP          ; Chegou no fim, pare!
11CD'  CD 10DD'                        CALL    NXT_INSTR       ; Verifique a instrucao e coloque um PP
11D0'  D2 0C66'                        JP      NC,STOP
11D3'  3A 4015                         LD      A,(FLAG_N)      ; Mostre os registradores ?
11D6'  47                              LD      B,A
11D7'  3A 4010                         LD      A,(DUMP?)
11DA'  B0                              OR      B
11DB'  32 4010                         LD      (DUMP?),A
11DE'  C3 0BE0'                        JP      GO_PRG          ; Continua a execucao

11E1'  FD 7E 00         CHK_GRP:       LD      A,(IY+0)
11E4'  FE ED                           CP      0EDH            ; Grupo ED ?
11E6'  28 0B                           JR      Z,CHK_GRP1
11E8'  E6 DF                           AND     0DFH
```

```
11EA'  FE DD                    CP      0DDH            ; Grupo FD ou DD
11EC'  C0                       RET     NZ
11ED'  FD 7E 01                 LD      A,(IY+1)
11F0'  FE E9                    CP      0E9H            ; JP (IX) ?
11F2'  C9                       RET
11F3'  FD 7E 01         CHK_GRP1:LD     A,(IY+1)
11F6'  E6 F7                    AND     0F7H
11F8'  FE 45                    CP      045H            ; RETN ou RETI
11FA'  C9                       RET

11FB'  2A 401B          CMD_A:  LD      HL,(POSICAO)    ; Comando Assembler
11FE'  CD 044C'                 CALL    DEF_END         ; Le um valor com "default" da posicao
1201'  E5                       PUSH    HL              ;       corrente
1202'  FD E1                    POP     IY
1204'  21 1244'                 LD      HL,CMD_A3       ; End. no caso de erro
1207'  22 4000                  LD      (END_ERR),HL
120A'  ED 73 400B               LD      (TMP_SP),SP     ; Stack corrente
120E'  FD 22 400D       CMD_A1: LD      (TMP_IY),IY     ; IY corrente, no caso de erro
1212'  FD E5                    PUSH    IY
1214'  E1                       POP     HL
1215'  E5                       PUSH    HL              ; Antes de montar a instrucao
1216'  CD 0DB2'                 CALL    CMD_LIST        ;  desmonta a instrucao corrente
1219'  FD E1                    POP     IY
121B'  48                       LD      C,B
121C'  ED 5B 401E               LD      DE,(VAR_@)      ; Relocacao ativa?
1220'  7A                       LD      A,D
1221'  B3                       OR      E
1222'  06 11                    LD      B,17            ; Caso negativo pare na coluna 17
1224'  28 02                    JR      Z,CMD_A2
1226'  06 19                    LD      B,25            ; Caso positivo na coluna 25
1228'  CD 0409'         CMD_A2: CALL    WHITE
122B'  3A 4002                  LD      A,(CNT_CAR)     ; Imprime brancos ate a coluna
122E'  B8                       CP      B               ;       corrente
122F'  38 F7                    JR      C,CMD_A2
1231'  C5                       PUSH    BC
1232'  CD 0331'                 CALL    GET_LINE        ; Le uma linha, instrucao a ser
1235'  C1                       POP     BC              ;       montada
                        +       SKIP_WHITE              ; Pule brancos
1237'  FE 2E                    CP      '.'             ; Fim de montagem ?
1239'  C8                       RET     Z
123A'  A7                       AND     A
123B'  C4 124D'                 CALL    NZ,CMD_ASS      ; Monte a instrucao
123E'  06 00                    LD      B,0
1240'  FD 09                    ADD     IY,BC           ; Proximo endereco
1242'  18 CA                    JR      CMD_A1

                        ;       Em caso de erro a execucao cai aqui

1244'  CD 0307'         CMD_A3: CALL    MSG_ERROR       ; Imprime mensagem de erro
1247'  ED 7B 400B               LD      SP,(TMP_SP)     ; Restabelece Stack Point
124B'  18 C1                    JR      CMD_A1          ; Volta ao comando

124D'                   +CMD_ASS:       SKIP_WHITE      ; Monte uma instrucao
124E'  21 1D8C'                 LD      HL,INSTR        ; Procure na tabela de instrucoes
1251'  CD 0475'                 CALL    CHKB
1254'  D2 02C8'                 JP      NC,ERROR        ; Nao achou, erro
                        +       SKIP_WHITE
1258'  D5                       PUSH    DE
1259'  78                       LD      A,B
125A'  80                       ADD     A,B
125B'  80                       ADD     A,B             ; X 3 -
125C'  21 1A26'                 LD      HL,TABELA1      ; Tabela de rotinas para analize
```

```
125F' CD 0428'                  CALL    SUM_HL_A        ;  dos operandos
1262' 5E                        LD      E,(HL)
1263' 23                        INC     HL
1264' 56                        LD      D,(HL)
1265' 23                        INC     HL
1266' 46                        LD      B,(HL)          ; Byte para analize dos operados
1267' EB                        EX      DE,HL
1268' D1                        POP     DE
1269' E9                        JP      (HL)            ; Execute a rotina de analize

126A' CD 1707'      CMD_A_1:    CALL    GET_REG         ; Verifica argumento, registrador
126D' 38 7D                     JR      C,CMD_A_1_8
126F' CD 172C'                  CALL    GET_C_IXIY      ; (IX) ou (IY)
1272' DA 1389'                  JP      C,CMD_A_1_20
1275' CD 16F4'                  CALL    GET_PAR_REG     ; Par de registradores
1278' DA 13B4'                  JP      C,CMD_A_1_22
127B' CD 1714'                  CALL    GET_REG_IXIY    ; IX ou IY
127E' DA 140D'                  JP      C,CMD_A_1_28
1281' 1A                        LD      A,(DE)          ; Registrador I
1282' FE 49                     CP      'I'
1284' CA 1429'                  JP      Z,CMD_A_1_30
1287' FE 52                     CP      'R'             ; Registrador R
1289' CA 1431'                  JP      Z,CMD_A_1_31
128C' FE 28                     CP      '('
128E' C2 02C8'                  JP      NZ,ERROR
1291' 13                        INC     DE
1292' CD 16F4'                  CALL    GET_PAR_REG     ; (Par registrador)
1295' DA 1440'                  JP      C,CMD_A_1_33
1298' CD 16E4'                  CALL    CMD_A_GET_INT   ; Endereco
129B' CD 179C'                  CALL    CHK_FECHAR      ; Fecha parenteses
129E' CD 1795'                  CALL    CHK_COMMA       ; Virgula, verifica segundo argumento
12A1' CD 16F4'                  CALL    GET_PAR_REG     ; Par de registradores
12A4' 38 34                     JR      C,CMD_A_1_5
12A6' CD 1714'                  CALL    GET_REG_IXIY    ; IX ou IY
12A9' 30 17                     JR      NC,CMD_A_1_3    ; Continua se nao encontrado
12AB' 06 22                     LD      B,22H
12AD' CD 0452'      CMD_A_1_1:  CALL    END_CMD         ; Fim do comando
12B0' 3A 401D                   LD      A,(VAR_L1)
12B3' FD 77 00      CMD_A_1_2:  LD      (IY+00),A       ; Monta a instrucao na memoria
12B6' FD 70 01                  LD      (IY+01),B       ; Segundo argumento
12B9' FD 75 02                  LD      (IY+02),L       ; Endereco
12BC' FD 74 03                  LD      (IY+03),H
12BF' 0E 04                     LD      C,4             ; Tamanho da instrucao 4 bytes
12C1' C9                        RET

12C2' 1A            CMD_A_1_3:  LD      A,(DE)
12C3' FE 41                     CP      'A'             ; Acumulador ?
12C5' C2 02C8'                  JP      NZ,ERROR
12C8' 13                        INC     DE
12C9' 06 32                     LD      B,32H           ; LD (nn),A
12CB' CD 0452'      CMD_A_1_4:  CALL    END_CMD
12CE' FD 70 00                  LD      (IY+00),B       ; Instrucao
12D1' FD 75 01                  LD      (IY+01),L       ; Endereco
12D4' FD 74 02                  LD      (IY+02),H
12D7' 0E 03                     LD      C,3             ; Tamanho da instrucao 3 bytes
12D9' C9                        RET

12DA' FE 20         CMD_A_1_5:  CP      20H             ; HL ?
12DC' 28 0A                     JR      Z,CMD_A_1_7
12DE' C6 43                     ADD     A,43H           ; Soma codigo da instrucao
12E0' 47                        LD      B,A             ;  Grupo ED : LD (nn),RP
12E1' CD 0452'      CMD_A_1_6:  CALL    END_CMD         ; Fim de comando
```

```
12E4'  3E ED                    LD      A,0EDH          ; Grupo ED
12E6'  18 CB                    JR      CMD_A_1_2

12E8'  06 22        CMD_A_1_7:  LD      B,22H           ; LD (nn),HL
12EA'  18 DF                    JR      CMD_A_1_4

12EC'  47           CMD_A_1_8:  LD      B,A             ; Registrador
12ED'  CD 1795'                 CALL    CHK_COMMA       ; Virgula
12F0'  CD 1707'                 CALL    GET_REG         ; Registrador ?
12F3'  30 13                    JR      NC,CMD_A_1_10
12F5'  F5                       PUSH    AF              ; Sim, monta instrucao
12F6'  78                       LD      A,B
12F7'  07                       RLCA                    ; Desloca 3 bits
12F8'  07                       RLCA
12F9'  07                       RLCA
12FA'  47                       LD      B,A
12FB'  F1                       POP     AF
12FC'  80                       ADD     A,B             ; Soma com o segundo registrador
12FD'  C6 40                    ADD     A,40H           ;  + 40H
12FF'  FE 76                    CP      76H             ; 'LD (HL),(HL)' nao existe
1301'  CA 02C8'                 JP      Z,ERROR
1304'  47           CMD_A_1_9:  LD      B,A
1305'  C3 1453'                 JP      CMD_A_2         ; Armazena na memoria

1308'  CD 172C'     CMD_A_1_10: CALL    GET_C_IXIY      ; Reg,(IX+-n) ou Reg,(IY+-n) ?
130B'  30 1E                    JR      NC,CMD_A_1_12
130D'  78                       LD      A,B             ; Reg
130E'  07                       RLCA
130F'  07                       RLCA
1310'  07                       RLCA                    ; Desloca 3 bits
1311'  C6 46                    ADD     A,46H
1313'  FE 76                    CP      76H             ; 'LD (HL),(IX/IY+-n)' nao existe
1315'  CA 02C8'                 JP      Z,ERROR
1318'  47           CMD_A_1_11: LD      B,A
1319'  CD 0452'                 CALL    END_CMD         ; Fim do comando
131C'  FD 70 01                 LD      (IY+01),B       ; Registrador
131F'  FD 71 02                 LD      (IY+02),C       ; Deslocamento
1322'  3A 401D                  LD      A,(VAR_L1)      ; Instrucao
1325'  FD 77 00                 LD      (IY+00),A
1328'  0E 03                    LD      C,3             ; Tamanho 3 bytes
132A'  C9                       RET

132B'  1A           CMD_A_1_12: LD      A,(DE)
132C'  FE 49                    CP      'I'             ; Registrador I
132E'  28 0E                    JR      Z,CMD_A_1_13
1330'  FE 52                    CP      'R'             ; Registrador R
1332'  20 16                    JR      NZ,CMD_A_1_15
1334'  78                       LD      A,B
1335'  FE 07                    CP      7               ; Registrador destino = 'A' ?
1337'  C2 02C8'                 JP      NZ,ERROR
133A'  06 5F                    LD      B,5FH           ; LD A,R
133C'  18 08                    JR      CMD_A_1_14
133E'  78           CMD_A_1_13: LD      A,B
133F'  FE 07                    CP      7               ; Destino = A ?
1341'  C2 02C8'                 JP      NZ,ERROR
1344'  06 57                    LD      B,57H           ; LD A,I
1346'  13           CMD_A_1_14: INC     DE
1347'  C3 145C'                 JP      CMD_A_3         ; Monta a instrucao
134A'  FE 28        CMD_A_1_15: CP      '('             ; Conteudo ?
134C'  28 16                    JR      Z,CMD_A_1_18    ;       sim
134E'  CD 16DA'                 CALL    GET_BYTE        ; Nao, LD reg,n
1351'  78                       LD      A,B
```

```
1352'  07                        RLCA
1353'  07                        RLCA
1354'  07                        RLCA                    ; Desloca 3 bits
1355'  C6 06                     ADD     A,6
1357'  47           CMD_A_1_16:  LD      B,A
1358'  CD 0452'     CMD_A_1_17:  CALL    END_CMD         ; Fim do comando
135B'  FD 70 00                  LD      (IY+00),B       ; Instrucao
135E'  FD 75 01                  LD      (IY+01),L       ; Valor
1361'  0E 02                     LD      C,2             ; Tamanho da instrucao 2 bytes
1363'  C9                        RET

1364'  13           CMD_A_1_18:  INC     DE
1365'  78                        LD      A,B
1366'  FE 07                     CP      7               ; Reg. Destino = A ?
1368'  C2 02C8'                  JP      NZ,ERROR
136B'  CD 16F4'                  CALL    GET_PAR_REG     ; Verica par de registradores
136E'  30 0E                     JR      NC,CMD_A_1_19
1370'  FE 30                     CP      30H
1372'  D2 02C8'                  JP      NC,ERROR        ; 'LD A,(SP)' nao existe
1375'  C6 0A                     ADD     A,0AH
1377'  47                        LD      B,A
1378'  CD 179C'                  CALL    CHK_FECHAR      ; Fecha parenteses
137B'  C3 1453'                  JP      CMD_A_2

137E'  CD 16E4'     CMD_A_1_19:  CALL    CMD_A_GET_INT   ; LD A,(nn)
1381'  CD 179C'                  CALL    CHK_FECHAR
1384'  06 3A                     LD      B,3AH
1386'  C3 12CB'                  JP      CMD_A_1_4       ; Monta a instrucao

1389'  CD 1795'     CMD_A_1_20:  CALL    CHK_COMMA       ; Virgula
138C'  CD 1707'                  CALL    GET_REG         ; Registrador fonte
138F'  30 0A                     JR      NC,CMD_A_1_21
1391'  FE 06                     CP      6               ; (HL) ?
1393'  CA 02C8'                  JP      Z,ERROR
1396'  C6 70                     ADD     A,70H           ; LD (IX/IY+-n),reg
1398'  C3 1318'                  JP      CMD_A_1_11

139B'  CD 16DA'     CMD_A_1_21:  CALL    GET_BYTE        ; LD (IX/IY+-n),n1
139E'  CD 0452'                  CALL    END_CMD         ; Fim do comando
13A1'  3A 401D                   LD      A,(VAR_L1)      ; Instrucao
13A4'  FD 77 00                  LD      (IY+00),A
13A7'  FD 36 01                  LD      (IY+01),36H
13AA'  36
13AB'  FD 71 02                  LD      (IY+02),C       ; n ( deslocamento )
13AE'  FD 75 03                  LD      (IY+03),L       ; n1 ( valor )
13B1'  0E 04                     LD      C,4             ; Instrucao de 4 bytes
13B3'  C9                        RET

13B4'  47           CMD_A_1_22:  LD      B,A
13B5'  CD 1795'                  CALL    CHK_COMMA       ; Virgula
13B8'  21 1E79'                  LD      HL,REGS4
13BB'  CD 0488'                  CALL    CHKB_4          ; Procura na tabela de registradores
13BE'  38 1B                     JR      C,CMD_A_1_24    ;       HL ou AF ?
13C0'  CD 1714'                  CALL    GET_REG_IXIY    ; Registradores IX ou IY
13C3'  30 1F                     JR      NC,CMD_A_1_25
13C5'  78                        LD      A,B
13C6'  FE 30                     CP      30H             ; SP ?
13C8'  20 73                     JR      NZ,GO2_ERROR
13CA'  06 F9                     LD      B,0F9H          ; LD SP,IX/IY
13CC'  CD 0452'     CMD_A_1_23:  CALL    END_CMD         ; Fim do comando
13CF'  3A 401D                   LD      A,(VAR_L1)      ; Instrucao
13D2'  FD 77 00                  LD      (IY+00),A
```

```
13D5'  FD 70 01                LD      (IY+01),B       ; segundo byte
13D8'  0E 02                   LD      C,2             ; 2 bytes de instrucao
13DA'  C9                      RET

13DB'  78           CMD_A_1_24:LD      A,B
13DC'  FE 30                   CP      30H             ; SP ?
13DE'  20 5D                   JR      NZ,GO2_ERROR
13E0'  06 F9                   LD      B,0F9H          ; LD SP,HL
13E2'  18 6F                   JR      CMD_A_2

13E4'  1A           CMD_A_1_25:LD      A,(DE)
13E5'  FE 28                   CP      '('
13E7'  20 17                   JR      NZ,CMD_A_1_27
13E9'  13                      INC     DE
13EA'  CD 16E4'                CALL    CMD_A_GET_INT   ; Le um endereco
13ED'  CD 179C'                CALL    CHK_FECHAR      ; Fecha o parenteses
13F0'  78                      LD      A,B
13F1'  FE 20                   CP      20H             ; HL ?
13F3'  28 06                   JR      Z,CMD_A_1_26
13F5'  C6 4B                   ADD     A,4BH
13F7'  47                      LD      B,A             ; LD reg,(IX/IY+-n)
13F8'  C3 12E1'                JP      CMD_A_1_6

13FB'  06 2A        CMD_A_1_26:LD      B,2AH           ; LD HL,(nn)
13FD'  C3 12CB'                JP      CMD_A_1_4

1400'  CD 16E4'     CMD_A_1_27:CALL    CMD_A_GET_INT   ; Le um inteiro
1403'  CD 0452'                CALL    END_CMD         ; Fim do comando
1406'  3E 01                   LD      A,1
1408'  80                      ADD     A,B             ; LD reg,nn
1409'  47                      LD      B,A
140A'  C3 12CB'                JP      CMD_A_1_4

140D'  CD 1795'     CMD_A_1_28:CALL    CHK_COMMA       ; Virgula
1410'  1A                      LD      A,(DE)
1411'  FE 28                   CP      '('
1413'  20 0C                   JR      NZ,CMD_A_1_29
1415'  13                      INC     DE
1416'  CD 16E4'                CALL    CMD_A_GET_INT   ; (endereco)
1419'  CD 179C'                CALL    CHK_FECHAR
141C'  06 2A                   LD      B,2AH           ; LD IX/IY,(endereco)
141E'  C3 12AD'                JP      CMD_A_1_1

1421'  CD 16E4'     CMD_A_1_29:CALL    CMD_A_GET_INT   ; Obtem um valor 16 bits, nn
1424'  06 21                   LD      B,21H           ; LD IX/IY,nn
1426'  C3 12AD'                JP      CMD_A_1_1

1429'  13           CMD_A_1_30:INC     DE
142A'  CD 1795'                CALL    CHK_COMMA       ; Virgula
142D'  06 47                   LD      B,47H           ; LD I,A
142F'  18 06                   JR      CMD_A_1_32

1431'  13           CMD_A_1_31:INC     DE
1432'  CD 1795'                CALL    CHK_COMMA       ; Virgula
1435'  06 4F                   LD      B,4FH           ; LD R,A
1437'  1A           CMD_A_1_32:LD      A,(DE)
1438'  13                      INC     DE
1439'  FE 41                   CP      'A'             ; Acumulador ?
143B'  28 1F                   JR      Z,CMD_A_3
143D'  C3 02C8'     GO2_ERROR: JP      ERROR

1440'  FE 20        CMD_A_1_33:CP      20H             ; BC, DE ?
```

```
1442'  30 F9                     JR     NC,GO2_ERROR
1444'  C6 02                     ADD    A,2            ; Deslocamento
1446'  47                        LD     B,A
1447'  CD 179C'                  CALL   CHK_FECHAR
144A'  CD 1795'                  CALL   CHK_COMMA
144D'  1A                        LD     A,(DE)
144E'  FE 41                     CP     'A'            ; Acumulador ?
1450'  20 EB                     JR     NZ,GO2_ERROR
1452'  13                        INC    DE
1453'  CD 0452'     CMD_A_2:    CALL   END_CMD        ; Fim do comando
1456'  FD 70 00                  LD     (IY+00),B
1459'  0E 01                     LD     C,1            ; Instrucoes de 1 byte
145B'  C9                        RET

145C'  CD 0452'     CMD_A_3:    CALL   END_CMD        ; Fim do comando
145F'  FD 36 00                  LD     (IY+00),0EDH   ; Grupo ED
1462'  ED
1463'  FD 70 01                  LD     (IY+01),B      ; Segundo byte
1466'  0E 02                     LD     C,2            ; 2 bytes da instrucao
1468'  C9                        RET

1469'  CD 16DA'     CMD_A_4:    CALL   GET_BYTE       ; Le um byte
146C'  C3 1358'                  JP     CMD_A_1_17     ; Monta a instrucao

146F'  21 1E79'     CMD_A_5:    LD     HL,REGS4       ; HL, AF ?
1472'  CD 0488'                  CALL   CHKB_4
1475'  30 4F                     JR     NC,CMD_A_7
1477'  CD 1795'                  CALL   CHK_COMMA      ; Virgula
147A'  CD 16F4'                  CALL   GET_PAR_REG    ; Par de registradores
147D'  D2 02C8'                  JP     NC,ERROR
1480'  F5                        PUSH   AF
1481'  78                        LD     A,B            ; Registrador
1482'  FE 88                     CP     88H
1484'  06 4A                     LD     B,4AH          ; ADC HL,reg
1486'  28 02                     JR     Z,CMD_A_5_1
1488'  06 42                     LD     B,42H          ; SBC HL,reg
148A'  F1           CMD_A_5_1:  POP    AF
148B'  80                        ADD    A,B            ; Soma o par de registradores fontes
148C'  47           CMD_A_5_2:  LD     B,A
148D'  18 CD                     JR     CMD_A_3

148F'  21 1E79'     CMD_A_6:    LD     HL,REGS4       ; HL ou AF ?
1492'  CD 0488'                  CALL   CHKB_4
1495'  38 21                     JR     C,CMD_A_6_3
1497'  CD 1714'                  CALL   GET_REG_IXIY   ; IX ou IY ?
149A'  30 2A                     JR     NC,CMD_A_7
149C'  CD 1795'                  CALL   CHK_COMMA      ; Virgula
149F'  21 1E87'                  LD     HL,REGS6       ; BC, DE, IX, SP
14A2'  3A 401D                   LD     A,(VAR_L1)
14A5'  FE FD                     CP     0FDH           ; IY
14A7'  20 03                     JR     NZ,CMD_A_6_1
14A9'  21 1E7E'                  LD     HL,REGS5       ; BC, DE, IY, SP
14AC'  CD 16F1'     CMD_A_6_1:  CALL   GET_REG_HL     ; Verifica
14AF'  D2 02C8'                  JP     NC,ERROR
14B2'  C6 09                     ADD    A,9
14B4'  47           CMD_A_6_2:  LD     B,A            ; ADD IX/IY,reg
14B5'  C3 13CC'                  JP     CMD_A_1_23

14B8'  CD 1795'     CMD_A_6_3:  CALL   CHK_COMMA      ; Virgula
14BB'  CD 16F4'                  CALL   GET_PAR_REG    ; Par de registradores ?
14BE'  D2 02C8'                  JP     NC,ERROR
14C1'  C6 09                     ADD    A,9            ; ADD HL,reg
```

```
14C3'  C3 1304'                  JP     CMD_A_1_9

14C6'  1A           CMD_A_7:     LD     A,(DE)
14C7'  FE 41                     CP     'A'           ; Acumulador ?
14C9'  20 0B                     JR     NZ,CMD_A_7_2
14CB'  D5                        PUSH   DE
14CC'  13                        INC    DE
14CD'  CD 0434'                  CALL   SKIP_COMMA    ; Virgula
14D0'  28 03                     JR     Z,CMD_A_7_1
14D2'  D1                        POP    DE
14D3'  18 01                     JR     CMD_A_7_2

14D5'  F1           CMD_A_7_1:   POP    AF
14D6'  CD 1707'     CMD_A_7_2:   CALL   GET_REG       ; Registrador simples
14D9'  38 0E                     JR     C,CMD_A_7_3
14DB'  CD 172C'                  CALL   GET_C_IXIY    ; (IX/IY+-n)
14DE'  38 0D                     JR     C,CMD_A_7_4
14E0'  CD 16DA'                  CALL   GET_BYTE
14E3'  78                        LD     A,B
14E4'  C6 46                     ADD    A,46H         ; OR A,n ou AND A,n
14E6'  C3 1357'                  JP     CMD_A_1_16

14E9'  80           CMD_A_7_3:   ADD    A,B
14EA'  C3 1304'                  JP     CMD_A_1_9

14ED'  78           CMD_A_7_4:   LD     A,B
14EE'  C6 06                     ADD    A,6
14F0'  C3 1318'                  JP     CMD_A_1_11

14F3'  CD 1707'     CMD_A_8:     CALL   GET_REG       ; Registrador simples ?
14F6'  38 20                     JR     C,CMD_A_8_2
14F8'  CD 172C'                  CALL   GET_C_IXIY    ; (IX/IY+-n) ?
14FB'  D2 02C8'                  JP     NC,ERROR
14FE'  78                        LD     A,B
14FF'  C6 06                     ADD    A,6           ; RLC, RL (IX/IY+-n)
1501'  47                        LD     B,A
1502'  CD 0452'     CMD_A_8_1:   CALL   END_CMD       ; Fim de comando
1505'  3A 401D                   LD     A,(VAR_L1)    ; Instrucao
1508'  FD 77 00                  LD     (IY+00),A
150B'  FD 36 01                  LD     (IY+01),0CBH  ; Sub grupo CB
150E'  CB
150F'  FD 71 02                  LD     (IY+02),C     ; Deslocamento
1512'  FD 70 03                  LD     (IY+03),B     ; quarto byte
1515'  0E 04                     LD     C,4           ; 4 bytes de instrucoes
1517'  C9                        RET

1518'  80           CMD_A_8_2:   ADD    A,B           ; Registrador
1519'  47           CMD_A_8_3:   LD     B,A
151A'  CD 0452'                  CALL   END_CMD       ; Fim do comando
151D'  FD 70 01                  LD     (IY+01),B
1520'  FD 36 00                  LD     (IY+00),0CBH  ; Grupo CB ( RLC reg )
1523'  CB
1524'  0E 02                     LD     C,2           ; 2 bytes
1526'  C9                        RET

1527'  CD 16BA'     CMD_A_9:     CALL   GET_0_7       ; Le o numero do bit
152A'  CD 1795'                  CALL   CHK_COMMA     ; Virgula
152D'  CD 1707'                  CALL   GET_REG       ; Registrador ?
1530'  38 10                     JR     C,CMD_A_9_1
1532'  CD 172C'                  CALL   GET_C_IXIY    ; (IX+-n) ou (IY+-n)
1535'  D2 02C8'                  JP     NC,ERROR
1538'  7D                        LD     A,L
```

```
1539'  07                          RLCA
153A'  07                          RLCA
153B'  07                          RLCA
153C'  C6 06                       ADD     A,6             ; BIT, RES, SET b,(IX/IY+-n)
153E'  80                          ADD     A,B
153F'  47                          LD      B,A
1540'  18 C0                       JR      CMD_A_8_1

1542'  80              CMD_A_9_1:  ADD     A,B             ; BIT, RES, SET b,reg
1543'  47                          LD      B,A
1544'  7D                          LD      A,L
1545'  07                          RLCA                    ; Monta o segundo byte
1546'  07                          RLCA
1547'  07                          RLCA
1548'  80                          ADD     A,B
1549'  18 CE                       JR      CMD_A_8_3

154B'  D5              CMD_A_10:   PUSH    DE
154C'  CD 177C'                    CALL    CHK_COND_ALL    ; Verifica condicoes
154F'  30 09                       JR      NC,CMD_A_10_1
1551'  80                          ADD     A,B             ; Soma na instrucao
1552'  47                          LD      B,A
1553'  CD 0434'                    CALL    SKIP_COMMA      ; Virgula
1556'  28 04                       JR      Z,CMD_A_10_2
1558'  D1                          POP     DE
1559'  D5                          PUSH    DE
155A'  06 CD           CMD_A_10_1: LD      B,0CDH          ; CALL
155C'  F1              CMD_A_10_2: POP     AF
155D'  CD 16E4'                    CALL    CMD_A_GET_INT   ; Endereco
1560'  C3 12CB'                    JP      CMD_A_1_4

1563'  CD 177C'        CMD_A_11:   CALL    CHK_COND_ALL    ; Verifica condicoes
1566'  30 04                       JR      NC,CMD_A_11_1
1568'  80                          ADD     A,B             ; Soma da instrucao
1569'  47                          LD      B,A
156A'  18 02                       JR      CMD_A_11_2

156C'  06 C9           CMD_A_11_1: LD      B,0C9H          ; RET
156E'  C3 1453'        CMD_A_11_2: JP      CMD_A_2

1571'  D5              CMD_A_12:   PUSH    DE
1572'  CD 177C'                    CALL    CHK_COND_ALL    ; Verifica condicoes
1575'  38 0D                       JR      C,CMD_A_12_2
1577'  D1              CMD_A_12_1: POP     DE
1578'  21 1D44'                    LD      HL,INSTR1       ; (HL), (IX), (IY)
157B'  CD 0475'                    CALL    CHKB
157E'  38 12                       JR      C,CMD_A_12_4
1580'  06 C3                       LD      B,0C3H          ; JP endereco
1582'  18 08                       JR      CMD_A_12_3

1584'  80              CMD_A_12_2: ADD     A,B
1585'  47                          LD      B,A             ; Soma na instrucao base
1586'  CD 0434'                    CALL    SKIP_COMMA      ; Virgula
1589'  20 EC                       JR      NZ,CMD_A_12_1
158B'  F1                          POP     AF
158C'  CD 16E4'        CMD_A_12_3: CALL    CMD_A_GET_INT   ; Endereco
158F'  C3 12CB'                    JP      CMD_A_1_4

1592'  CD 0452'        CMD_A_12_4: CALL    END_CMD
1595'  78                          LD      A,B
1596'  A7                          AND     A
1597'  20 05                       JR      NZ,CMD_A_12_5
```

```
1599'  06 E9                    LD      B,0E9H          ; JP (HL)
159B'  C3 1453'                 JP      CMD_A_2

159E'  06 DD        CMD_A_12_5: LD      B,0DDH          ; IX
15A0'  3D                       DEC     A
15A1'  28 02                    JR      Z,CMD_A_12_6
15A3'  06 FD                    LD      B,0FDH          ; IY
15A5'  2E E9        CMD_A_12_6: LD      L,0E9H          ; JP (IX/IY)
15A7'  C3 1358'                 JP      CMD_A_1_17

15AA'  CD 0434'     CMD_A_13:   CALL    SKIP_COMMA      ; Virgula
15AD'  06 10                    LD      B,10H           ; DJNZ deslocamento
15AF'  18 0E                    JR      CMD_A_14_2

15B1'  CD 1783'     CMD_A_14:   CALL    CHK_COND        ; Condicao
15B4'  38 04                    JR      C,CMD_A_14_1
15B6'  06 18                    LD      B,18H           ; JR deslocamento
15B8'  18 05                    JR      CMD_A_14_2

15BA'  80           CMD_A_14_1: ADD     A,B             ; Soma na condicao
15BB'  47                       LD      B,A
15BC'  CD 1795'                 CALL    CHK_COMMA       ; Virgula
15BF'  CD 16C3'     CMD_A_14_2: CALL    GET_DESL        ; Deslocamento
15C2'  C3 1358'                 JP      CMD_A_1_17

15C5'  CD 16DA'     CMD_A_15:   CALL    GET_BYTE        ; Le um byte
15C8'  7D                       LD      A,L
15C9'  FE 03                    CP      3
15CB'  30 5C                    JR      NC,GO3_ERROR
15CD'  A7                       AND     A
15CE'  28 08                    JR      Z,CMD_A_15_1    ; IM0
15D0'  06 56                    LD      B,56H           ; IM1
15D2'  FE 01                    CP      1
15D4'  28 02                    JR      Z,CMD_A_15_1
15D6'  06 5E                    LD      B,5EH           ; IM2
15D8'  C3 145C'     CMD_A_15_1: JP      CMD_A_3

15DB'  CD 16DA'     CMD_A_16:   CALL    GET_BYTE        ; Le um byte
15DE'  7D                       LD      A,L
15DF'  F5                       PUSH    AF
15E0'  80                       ADD     A,B
15E1'  47                       LD      B,A
15E2'  F1                       POP     AF
15E3'  E6 C7                    AND     0C7H            ; RST n
15E5'  20 42                    JR      NZ,GO3_ERROR
15E7'  C3 1453'                 JP      CMD_A_2

15EA'  CD 1714'     CMD_A_17:   CALL    GET_REG_IXIY    ; IX ou IY ?
15ED'  38 09                    JR      C,CMD_A_17_1
15EF'  CD 16EB'                 CALL    GET_PAR1_REG    ; Par de registradores
15F2'  30 35                    JR      NC,GO3_ERROR
15F4'  80                       ADD     A,B             ; POP reg ou PUSH reg
15F5'  C3 1304'                 JP      CMD_A_1_9

15F8'  78           CMD_A_17_1: LD      A,B
15F9'  C6 20                    ADD     A,20H           ; POP IX ou POP IY
15FB'  C3 14B4'                 JP      CMD_A_6_2

15FE'  CD 1707'     CMD_A_18:   CALL    GET_REG         ; Registrador simples
1601'  30 26                    JR      NC,GO3_ERROR
1603'  FE 06                    CP      6               ; (HL) -
1605'  28 22                    JR      Z,GO3_ERROR
```

```
1607'  07                          RLCA
1608'  07                          RLCA
1609'  07                          RLCA                    ; Desloca 3 bits
160A'  80                          ADD     A,B
160B'  47                          LD      B,A
160C'  FE 78                       CP      78H             ; IN A,(C) ?
160E'  20 10                       JR      NZ,CMD_A_18_1
1610'  CD 1795'                    CALL    CHK_COMMA       ; Virgula
1613'  CD 162C'                    CALL    CMD_A_18_3
1616'  38 0E                       JR      C,CMD_A_18_2
1618'  CD 16DA'                    CALL    GET_BYTE        ; Porto
161B'  06 DB                       LD      B,0DBH          ; IN A,(n)
161D'  C3 1358'                    JP      CMD_A_1_17

1620'  CD 1795'    CMD_A_18_1:    CALL    CHK_COMMA       ; Virgula
1623'  CD 162C'                    CALL    CMD_A_18_3
1626'  DA 145C'    CMD_A_18_2:    JP      C,CMD_A_3
1629'  C3 02CB'    GO3_ERROR:     JP      ERROR
162C'  21 1EBE'    CMD_A_18_3:    LD      HL,IO_P_C       ; (C)
162F'  C3 0488'                    JP      CHKB_4

1632'  CD 162C'    CMD_A_19:      CALL    CMD_A_18_3      ; Instrucao OUT
1635'  30 13                       JR      NC,CMD_A_19_1
1637'  CD 1795'                    CALL    CHK_COMMA
163A'  CD 1707'                    CALL    GET_REG         ; Registrador
163D'  30 EA                       JR      NC,GO3_ERROR
163F'  FE 06                       CP      6
1641'  28 E6                       JR      Z,GO3_ERROR
1643'  07                          RLCA
1644'  07                          RLCA
1645'  07                          RLCA
1646'  80                          ADD     A,B
1647'  C3 148C'                    JP      CMD_A_5_2

164A'  CD 16DA'    CMD_A_19_1:    CALL    GET_BYTE        ; Le um byte
164D'  CD 1795'                    CALL    CHK_COMMA
1650'  1A                          LD      A,(DE)
1651'  FE 41                       CP      'A'
1653'  20 D4                       JR      NZ,GO3_ERROR
1655'  13                          INC     DE
1656'  06 D3                       LD      B,0D3H          ; OUT (n),A
1658'  C3 1358'                    JP      CMD_A_1_17

165B'  21 1D51'    CMD_A_20:      LD      HL,INSTR2       ; EX AF,AF'; EX DE,AF; EX (SP),HL; etc
165E'  CD 0475'                    CALL    CHKB
1661'  D2 02CB'                    JP      NC,ERROR
1664'  48                          LD      C,B
1665'  CD 0452'                    CALL    END_CMD         ; Fim do comando
1668'  06 00                       LD      B,00
166A'  21 1D3A'                    LD      HL,TABEL14      ; Tabela contendo o codigo
166D'  09                          ADD     HL,BC           ;  correspondente
166E'  09                          ADD     HL,BC
166F'  7E                          LD      A,(HL)
1670'  FD 77 00                    LD      (IY+00),A
1673'  0E 01                       LD      C,1             ; Instrucao de 1 byte
1675'  23                          INC     HL
1676'  7E                          LD      A,(HL)
1677'  A7                          AND     A
1678'  C8                          RET     Z
1679'  FD 77 01                    LD      (IY+01),A       ; EX (SP),IX; EX (SP),IY
167C'  0E 02                       LD      C,2             ; Instrucoes de 2 bytes
167E'  C9                          RET
```

```
167F' CD 1714'    CMD_A_21:     CALL   GET_REG_IXIY   ; IX ou IY ?
1682' 38 16                     JR     C,CMD_A_21_1
1684' CD 16F4'                  CALL   GET_PAR_REG    ; Par de registradores ?
1687' 38 1D                     JR     C,CMD_A_21_3
1689' CD 1707'                  CALL   GET_REG        ; Registrador simples ?
168C' 38 25                     JR     C,CMD_A_21_5
168E' CD 172C'                  CALL   GET_C_IXIY     ; (IX/IY+-n) ?
1691' D2 02C9'                  JP     NC,ERROR
1694' 78                        LD     A,B
1695' C6 30                     ADD    A,30H          ; DEC (IX/IY+-n)
1697' C3 1318'                  JP     CMD_A_1_11

169A' 78          CMD_A_21_1:   LD     A,B
169B' 06 23                     LD     B,23H          ; INC IX/IY
169D' FE 04                     CP     4
169F' 28 02                     JR     Z,CMD_A_21_2
16A1' 06 2B                     LD     B,2BH          ; DEC IX/IY
16A3' C3 13CC'    CMD_A_21_2:   JP     CMD_A_1_23

16A6' F5          CMD_A_21_3:   PUSH   AF
16A7' 78                        LD     A,B
16A8' 06 03                     LD     B,3            ; INC par_reg
16AA' FE 04                     CP     4
16AC' 28 02                     JR     Z,CMD_A_21_4
16AE' 06 0B                     LD     B,0BH          ; DEC par_reg
16B0' F1          CMD_A_21_4:   POP    AF
16B1' 18 03                     JR     CMD_A_21_6

16B3' 07          CMD_A_21_5:   RLCA
16B4' 07                        RLCA
16B5' 07                        RLCA                  ; Desloca 3 bits
16B6' 80          CMD_A_21_6:   ADD    A,B            ; INC reg_simples
16B7' C3 1304'                  JP     CMD_A_1_9

16BA' CD 16DA'    GET_0_7:      CALL   GET_BYTE       ; Obtem um byte entre 0 a 7
16BD' 7D                        LD     A,L            ;  Ex.: bit B,6
16BE' FE 08                     CP     8
16C0' 30 26                     JR     NC,GO1_ERROR
16C2' C9                        RET

16C3' CD 16E4'    GET_DESL:     CALL   CMD_A_GET_INT  ; Obtem um deslocamento
16C6' C5                        PUSH   BC             ; Obtem um endereco
16C7' FD E5                     PUSH   IY
16C9' C1                        POP    BC
16CA' A7                        AND    A
16CB' ED 42                     SBC    HL,BC          ; Subtrai do endereco corrente
16CD' 2B                        DEC    HL
16CE' 2B                        DEC    HL
16CF' C1                        POP    BC
16D0' CD 16DD'                  CALL   CHK_BYTE       ; Verifica se deslocamento
16D3' 7C                        LD     A,H            ;  valido
16D4' AD                        XOR    L
16D5' CB 7F                     BIT    7,A
16D7' 20 0F                     JR     NZ,GO1_ERROR
16D9' C9                        RET

16DA' CD 16E4'    GET_BYTE:     CALL   CMD_A_GET_INT  ; Le um byte
16DD' 7C          CHK_BYTE:     LD     A,H            ; Verifica o byte
16DE' A7                        AND    A
16DF' C8                        RET    Z
16E0' 3C                        INC    A
```

```
16E1'  C8                        RET     Z
16E2'  18 04                     JR      GO1_ERROR

16E4'  C5           CMD_A_GET_INT: PUSH  BC              ; Le um inteiro
                   +               GET_INT
16E6'  C1                        POP     BC
16E7'  D0                        RET     NC
16E8'  C3 02C8'     GO1_ERROR:   JP      ERROR

16EB'  E5           GET_PAR1_REG: PUSH   HL
16EC'  21 1E75'                  LD      HL,REGS3
16EF'  18 07                     JR      GET_PAR_REG1
16F1'  E5           GET_REG_HL:  PUSH    HL              ; Procura na tabela apontada por HL
16F2'  18 04                     JR      GET_PAR_REG1
16F4'  E5           GET_PAR_REG: PUSH    HL              ; Le um par de registradores
16F5'  21 1E6C'                  LD      HL,REGS2        ; Tabela de pares de registradores
16F8'  C5           GET_PAR_REG1: PUSH   BC
16F9'  CD 0475'                  CALL    CHKB            ; Procura na tabela
16FC'  30 06                     JR      NC,GET_PAR_REG2 ; Nao achou
16FE'  78                        LD      A,B
16FF'  07                        RLCA                    ; Achou, CARRY ligado
1700'  07                        RLCA
1701'  07                        RLCA
1702'  07                        RLCA                    ; X 16 ( Nibble superior )
1703'  37                        SCF
1704'  C1           GET_PAR_REG2: POP    BC
1705'  E1                        POP     HL
1706'  C9                        RET

1707'               +GET_REG:    SKIP_WHITE              ; Le o registrador
1708'  C5                        PUSH    BC
1709'  E5                        PUSH    HL
170A'  21 1E60'                  LD      HL,REGS1        ; Tabela de registradores
170D'  CD 0475'                  CALL    CHKB            ; Procura na tabela
1710'  78                        LD      A,B
1711'  E1                        POP     HL
1712'  C1                        POP     BC
1713'  C9                        RET

1714'  E5           GET_REG_IXIY: PUSH   HL              ; Verifica reg. IX e IY
1715'  C5                        PUSH    BC
1716'  21 1E92'                  LD      HL,REGS8
1719'  CD 0475'                  CALL    CHKB
171C'  30 0B                     JR      NC,GET_REG_IXY2
171E'  3E DD                     LD      A,0DDH          ; Reg. IX
1720'  05                        DEC     B
1721'  20 02                     JR      NZ,GET_REG_IXY1
1723'  3E FD                     LD      A,0FDH          ; Reg. IY
1725'  32 401D       GET_REG_IXY1: LD     (VAR_L1),A
1728'  37                        SCF
1729'  C1           GET_REG_IXY2: POP    BC
172A'  E1                        POP     HL
172B'  C9                        RET

172C'  E5           GET_C_IXIY:  PUSH    HL              ; Verifica (IX+n), (IY+n), (IX-n) e
172D'  C5                        PUSH    BC              ;  (IY-n)
172E'  1A                        LD      A,(DE)
172F'  FE 28                     CP      '('
1731'  20 45                     JR      NZ,GET_C_IXY6
1733'  D5                        PUSH    DE
1734'  13                        INC     DE
1735'  21 1E92'                  LD      HL,REGS8        ; Verifica IX ou IY
```

```
1738'  CD 0475'                  CALL   CHKB
173B'  30 3A                     JR     NC,GET_C_IXY5
173D'  F1                        POP    AF
173E'  3E DD                     LD     A,0DDH          ; IX
1740'  05                        DEC    B
1741'  20 02                     JR     NZ,GET_C_IXY1
1743'  3E FD                     LD     A,0FDH          ; IY
1745'  32 401D       GET_C_IXY1: LD     (VAR_L1),A
1748'  1A                        LD     A,(DE)
1749'  FE 2B                     CP     '+'             ; (IX+ ou (IY+
174B'  28 0C                     JR     Z,GET_C_IXY2
174D'  FE 29                     CP     ')'             ; )
174F'  21 0000                   LD     HL,0
1752'  28 1D                     JR     Z,GET_C_IXY4
1754'  FE 2D                     CP     '-'             ; (IX- ou (IY-
1756'  C2 02C8'                  JP     NZ,ERROR
1759'  F5            GET_C_IXY2: PUSH   AF
175A'  13                        INC    DE
175B'  CD 16DA'                  CALL   GET_BYTE
175E'  F1                        POP    AF
175F'  FE 2B                     CP     '+'
1761'  28 08                     JR     Z,GET_C_IXY3
1763'  44                        LD     B,H
1764'  4D                        LD     C,L
1765'  21 0000                   LD     HL,0
1768'  A7                        AND    A
1769'  ED 42                     SBC    HL,BC
176B'  1A            GET_C_IXY3: LD     A,(DE)
176C'  FE 29                     CP     ')'
176E'  C2 02C8'                  JP     NZ,ERROR
1771'  13            GET_C_IXY4: INC    DE
1772'  C1                        POP    BC
1773'  4D                        LD     C,L
1774'  E1                        POP    HL
1775'  37                        SCF
1776'  C9                        RET
1777'  D1            GET_C_IXY5: POP    DE
1778'  C1            GET_C_IXY6: POP    BC
1779'  E1                        POP    HL
177A'  A7                        AND    A
177B'  C9                        RET

177C'  21 1EA6'      CHK_COND_ALL: LD   HL,FLAGS2       ; Verifica todas as FLAGS
177F'  0E 07                     LD     C,7
1781'  18 05                     JR     CHK_COND1
1783'  21 1E97'      CHK_COND:   LD     HL,FLAGS1       ; Verifique condicoes (NZ,Z,C,NC, etc)
1786'  0E 03                     LD     C,3
1788'  C5            CHK_COND1:  PUSH   BC
1789'  CD 0475'                  CALL   CHKB
178C'  78                        LD     A,B
178D'  C1                        POP    BC
178E'  D0                        RET    NC
178F'  A1                        AND    C
1790'  07                        RLCA
1791'  07                        RLCA
1792'  07                        RLCA
1793'  37                        SCF
1794'  C9                        RET

1795'  CD 0434'      CHK_COMMA:  CALL   SKIP_COMMA      ; Verifica Virgula
1798'  C8                        RET    Z
1799'  C3 02C8'      GO_ERROR:   JP     ERROR
```

```
179C' 1A          CHK_FECHAR:  LD    A,(DE)          ; Verifica ')'
179D' FE 29                    CP    ')'
179F' 20 F8                    JR    NZ,GO_ERROR
17A1' 13                       INC   DE
17A2' C9                       RET

17A3'             +CMD_K:      SKIP_WHITE            ; Comando K, operacoes com disco
17A4' A7                       AND   A
17A5' CA 02C8'                 JP    Z,ERROR
17A8' 21 17BF'                 LD    HL,TAB_CMDK     ; Tabela de Subcomandos K
17AB' CD 0461'                 CALL  CHK_TAB
17AE' 78                       LD    A,B
17AF' FE 0A                    CP    NUM_CMDK
17B1' D2 02C8'                 JP    NC,ERROR
17B4' 87                       ADD   A,A
17B5' 21 17CA'                 LD    HL,END_CMDK     ; Tabela de enderecos dos Subcomandos K
17B8' CD 0428'                 CALL  SUM_HL_A
17BB' CD 06D2'                 CALL  LD_HL
17BE' E9                       JP    (HL)            ; Executa Subcomando

17BF' 48 35 38    TAB_CMDK:    DEFM  'H58DLAPRWS'    ; Tabela de Subcomandos K
17C2' 44 4C 41
17C5' 50 52 57
17C8' 53
000A              NUM_CMDK     EQU   $ - TAB_CMDK    ; Numero de Subcomandos K
17C9' 00                       DEFB  0

17CA' 17E0'       END_CMDK:    DEFW  CMD_KH          ; Enderecos dos Subcomandos K
17CC' 17F6'                    DEFW  CMD_K5
17CE' 17FD'                    DEFW  CMD_K8
17D0' 1833'                    DEFW  CMD_KD
17D2' 184F'                    DEFW  CMD_KL
17D4' 1866'                    DEFW  CMD_KA
17D6' 17E5'                    DEFW  CMD_KP
17D8' 18AB'                    DEFW  CMD_KR
17DA' 18AF'                    DEFW  CMD_KW
17DC' 189D'                    DEFW  CMD_KS
17DE' 1833'                    DEFW  CMD_KD

17E0' CD 1997'    CMD_KH:      CALL  KHOME           ; Recalibra o disco ( Trilha Zero )
17E3' 18 0B                    JR    CMD_KP1

17E5'             +CMD_KP:     GET_INT               ; Posiciona cabeca na trilha n
17E6' DA 02C8'                 JP    C,ERROR         ; Le n
17E9' 7D                       LD    A,L
17EA' 32 4004                  LD    (TRK_DSK),A
17ED' CD 199B'                 CALL  SEEK_TRK        ; Procura a trilha n
17F0' CD 1985'    CMD_KP1:     CALL  STATUS
17F3' C3 030A'                 JP    CR_LF

17F6' CD 1808'    CMD_K5:      CALL  SIMDUP          ; Tipo: 5 1/4
17F9' CB A7                    RES   4,A
17FB' 18 05                    JR    KBPOL1

17FD' CD 1808'    CMD_K8:      CALL  SIMDUP          ; Tipo: 8 polegadas
1800' CB E7                    SET   4,A
1802' 32 4003     KBPOL1:      LD    (TYP_DSK),A
1805' D3 14                    OUT   (DSK_TYP),A
1807' C9                       RET

1808'             +SIMDUP:     SKIP_WHITE            ; Le a densidade ( S = simples,
```

```
1809'  13                      INC     DE              ;  D = cubla )
180A'  FE 44                   CP      'D'
180C'  20 09                   JR      NZ,SIMD1
180E'  CD 1825'                CALL    CMD_KN
1811'  3A 4003                 LD      A,(TYP_DSK)
1814'  CB AF                   RES     5,A
1816'  C9                     RET
1817'  FE 53          SIMD1:   CP      'S'
1819'  C2 02C8'                JP      NZ,ERROR
181C'  CD 1825'                CALL    CMD_KN
181F'  3A 4003                 LD      A,(TYP_DSK)
1822'  CB EF                   SET     5,A
1824'  C9                     RET

1825'                +CMD_KN:  GET_INT                 ; Tamanho do setor ( 0 = 128 bytes,
1826'  DA 02C8'                JP      C,ERROR         ;                    1 = 256 bytes e
1829'  7D                     LD      A,L             ;                    2 = 512 bytes.
182A'  FE 04                   CP      4
182C'  D2 02C8'                JP      NC,ERROR
182F'  32 4008                 LD      (PAR_DSK),A
1832'  C9                     RET

1833'                +CMD_KD:  SKIP_WHITE              ; Seleciona o Disco n
1834'  E6 0F                   AND     0FH
1836'  CA 02C8'                JP      Z,ERROR
1839'  FE 05                   CP      5
183B'  D2 02C8'                JP      NC,ERROR
183E'  06 01                   LD      B,1
1840'  3D            KDSK1:   DEC     A
1841'  28 04                   JR      Z,KDSK2
1843'  CB 20                   SLA     B
1845'  18 F9                   JR      KDSK1
1847'  3A 4003       KDSK2:   LD      A,(TYP_DSK)
184A'  E6 F0                   AND     0F0H
184C'  B0                      OR      B
184D'  18 B3                   JR      KBPOL1

184F'                +CMD_KL:  GET_INT                 ; Seleciona o Lado do disco ( 0 ou 1 )
1850'  DA 02C8'                JP      C.ERROR
1853'  7D                      LD      A,L
1854'  FE 02                   CP      2
1856'  D2 02C8'                JP      NC,ERROR
1859'  E6 01                   AND     1
185B'  3A 4003                 LD      A,(TYP_DSK)
185E'  CB B7                   RES     6,A
1860'  28 A0                   JR      Z,KBPOL1
1862'  CB F7                   SET     6,A
1864'  18 9C                   JR      KBPOL1

1866'  CD 199B'       CMD_KA:  CALL    SEEK_TRK        ; Le a posicao atual da cabeca do disco
1869'  CD 193F'       KAD3:    CALL    PREP_RDY        ;  continuamente ate CTRL_C ser
186C'  CD 19AD'                 CALL    RDY_DSK         ;  pressionado
186F'  01 061B                  LD      BC,600H+DSK_DAT
1872'  21 4020                  LD      HL,BUF_ID       ; Buffer onde os dados serao colocados
1875'  3E C0                    LD      A,CMD_ADDR
1877'  D3 18                    OUT     (DSK_CMD),A
1879'  76            KAD1:    HALT                    ; Pseudo DMA
187A'  ED A2                    INI
187C'  20 FB                    JR      NZ,KAD1
187E'  3A 4003                  LD      A,(TYP_DSK)
1881'  D3 14                    OUT     (DSK_TYP),A
1883'  CD 1985'                 CALL    STATUS          ; Le status
```

```
1886'  CD 0403'              CALL   WHITE4
1889'  06 06                 LD     B,6
188B'  21 4020               LD     HL,BUF_ID      ; Imprime o conteudo do Buffer
188E'  CD 0409'   KAD2:      CALL   WHITE          ;       trilha atual
1891'  7E                    LD     A,(HL)         ;       lado
1892'  CD 03EB'              CALL   OUT_A          ;       setor
1895'  23                    INC    HL             ;       tamanho do setor
1896'  10 F6                 DJNZ   KAD2           ;       CRC ( Cyclic Redundancy Check )
1898'  CD 030A'              CALL   CR_LF          ; Retorna o carro e muda de linha
189B'  18 CC                 JR     KAD3           ; Repete ...

189D'             +CMD_KS:   GET_INT               ; Modifica velocidade do Step
189E'  DA 02C8'              JP     C,ERROR        ;       0 = 3 ms
18A1'  7D                    LD     A,L            ;       1 = 6 ms
18A2'  FE 04                 CP     4              ;       2 = 10 ms
18A4'  D2 02C8'              JP     NC,ERROR       ;       3 = 15 ms
18A7'  32 4009               LD     (STP_DSK),A
18AA'  C9                    RET

18AB'  3E 01      CMD_KR:    LD     A,1            ; Le um ou mais setores
18AD'  18 01                 JR     KRW

18AF'  AF         CMD_KW:    XOR    A              ; Escreve um ou mais setores

18B0'  32 400A    KRW:       LD     (HST_OP),A
                  +          GET_INT               ; Le a trilha
18B4'  DA 02C8'              JP     C,ERROR
18B7'  7D                    LD     A,L
18B8'  32 4004               LD     (TRK_DSK),A
                  +          GET_INT               ; Le o setor
18BC'  DA 02C8'              JP     C,ERROR
18BF'  7D                    LD     A,L
18C0'  32 4005               LD     (SCT_DSK),A
                  +          GET_INT               ; Le o endereco do Buffer
18C4'  DA 02C8'              JP     C,ERROR
18C7'  22 4006               LD     (DMA_DSK),HL
18CA'  21 0001               LD     HL,1           ; Le o numero de setores com Default =
18CD'  CD 044C'              CALL   DEF_END        ;  a 1 e fim de comando ( CR )
18D0'  7D                    LD     A,L
18D1'  A7                    AND    A
18D2'  C8                    RET    Z
18D3'  CD 199B'              CALL   SEEK_TRK       ; Posiciona na trilha pedida
18D6'  45                    LD     B,L
18D7'  C5         KRW1:      PUSH   BC
18D8'  3A 400A               LD     A,(HST_OP)
18DB'  B7                    OR     A
18DC'  20 05                 JR     NZ,KRW4
18DE'  CD 1907'              CALL   WRITE_DSK      ; Escreve no setor
18E1'  18 03                 JR     KRW5
18E3'  CD 1919'   KRW4:      CALL   READ_DSK       ; Le do setor
18E6'  C4 196C'   KRW5:      CALL   NZ,STATUS_ERR
18E9'  21 0080               LD     HL,80H
18EC'  3A 4008               LD     A,(PAR_DSK)
18EF'  E6 03                 AND    3
18F1'  28 04                 JR     Z,KRW2
18F3'  29         KRW3:      ADD    HL,HL
18F4'  3D                    DEC    A
18F5'  20 FC                 JR     NZ,KRW3
18F7'  EB         KRW2:      EX     DE,HL
18F8'  2A 4006               LD     HL,(DMA_DSK)   ; Calcula novo endereco do Buffer
18FB'  19                    ADD    HL,DE
18FC'  22 4006               LD     (DMA_DSK),HL
```

```
18FF'  21 4005                  LD      HL,SCT_DSK
1902'  34                       INC     (HL)            ; Incrementa setor
1903'  C1                       POP     BC
1904'  10 D1                    DJNZ    KRW1
1906'  C9                       RET

1907'  CD 194C'     WRITE_DSK:  CALL    PREP            ; Rotina de baixo nivel para escrever
190A'  3E A0                    LD      A,CMD_WRITE     ; um setor
190C'  D3 18                    OUT     (DSK_CMD),A
190E'  76           WRT_D2:     HALT                    ; Pseudo DMA para o controlador
190F'  ED A3                    OUTI
1911'  20 FB                    JR      NZ,WRT_D2
1913'  15                       DEC     D
1914'  C2 190E'                 JP      NZ,WRT_D2
1917'  18 10                    JR      RWEND

1919'  CD 194C'     READ_DSK:   CALL    PREP            ; Rotina de baixo nivel para ler
191C'  3E 80                    LD      A,CMD_READ      ;  um setor do disco
191E'  D3 18                    OUT     (DSK_CMD),A
1920'  76           READ_D2:    HALT                    ; Pseudo DMA
1921'  ED A2                    INI
1923'  20 FB                    JR      NZ,READ_D2
1925'  15                       DEC     D
1926'  C2 1920'                 JP      NZ,READ_D2

1929'  DB 16        RWEND:      IN      A,(TIME_OUT)    ; Verifica Time-out
192B'  E6 20                    AND     BIT_TIME
192D'  28 09                    JR      Z,RWEND1
192F'  3E D0                    LD      A,CMD_ABORT     ; Aborta comando
1931'  D3 18                    OUT     (DSK_CMD),A
1933'  0E 00                    LD      C,0
1935'  CD 19A7'                 CALL    CMD_DSK
1938'  3A 4003      RWEND1:     LD      A,(TYP_DSK)
193B'  D3 14                    OUT     (DSK_TYP),A
193D'  18 6E                    JR      RDY_DSK

193F'  3A 4003      PREP_RDY:   LD      A,(TYP_DSK)         ; Verifica disco pronto
1942'  CB FF                    SET     7,A
1944'  D3 14                    OUT     (DSK_TYP),A
1946'  DB 18        PREP1:      IN      A,(DSK_STT)     ; Sinal de READY
1948'  07                       RLCA
1949'  38 FB                    JR      C,PREP1
194B'  C9                       RET

194C'  CD 193F'     PREP:       CALL    PREP_RDY        ; Verifica sinais do controlador
194F'  3A 4005                  LD      A,(SCT_DSK)
1952'  D3 1A                    OUT     (DSK_SCT),A     ; Informa setor
1954'  01 001B                  LD      BC,DSK_DAT
1957'  2A 4006                  LD      HL,(DMA_DSK)    ; Informa endereco para Pseudo DMA
195A'  3A 4008                  LD      A,(PAR_DSK)
195D'  E6 03                    AND     3
195F'  57                       LD      D,A
1960'  C0                       RET     NZ
1961'  06 80                    LD      B,80H
1963'  16 01                    LD      D,1
1965'  C9                       RET

1966'  3E 03        WAIT:       LD      A,3             ; Rotina de espera para o controlador
1968'  3D           WAIT1:      DEC     A               ;       ter tempo de modificar seu
1969'  20 FD                    JR      NZ,WAIT1        ;       registrador de status
196B'  C9                       RET
```

```
196C' CD 1985'    STATUS_ERR:   CALL   STATUS        ; Imprime Status e Setor em caso de
196F' CD 0415'                  CALL   PRINT_PC      ;  erro
1972' 2C 20 53                  DC     ', Setor = '
1975' 65 74 6F
1978' 72 20 3D
197B' A0
197C' 3A 4005                   LD     A,(SCT_DSK)
197F' CD 03EB'                  CALL   OUT_A
1982' C3 030A'                  JP     CR_LF

1985' CD 0415'    STATUS:       CALL   PRINT_PC      ; Imprime o setor Lido ou escrito
1988' 53 74 61                  DC     'Status = '
198B' 74 75 73
198E' 20 3D A0
1991' CD 19AD'                  CALL   RDY_DSK
1994' C3 03EB'                  JP     OUT_A

1997' 0E 08       KHOME:        LD     C,CMD_HOME    ; Retorna disco para trilha ZERO
1999' 18 07                     JR     CPT_CMD

199B' 3A 4004     SEEK_TRK:     LD     A,(TRK_DSK)   ; Posiciona cabeca do disco na trilha
199E' D3 18                     OUT    (DSK_DAT),A   ;  pedida
19A0' 0E 18                     LD     C,CMD_SEEK    ; Comando de posicionamento

19A2' 3A 4009     CPT_CMD:      LD     A,(STP_DSK)   ; Velocidade de avanco da cabeca
19A5' 81                        ADD    A,C
19A6' 4F                        LD     C,A

19A7' CD 19AD'    CMD_DSK:      CALL   RDY_DSK       ; Rotina que envia um comando ao controlador
19AA' 79                        LD     A,C           ;       de discos flexiveis
19AB' D3 18                     OUT    (DSK_CMD),A

19AD' CD 1966'    RDY_DSK:      CALL   WAIT          ; Verifica status do controlador
19B0' DB 18       RDY_D1:       IN     A,(DSK_STT)
19B2' CB 47                     BIT    0,A
19B4' 20 FA                     JR     NZ,RDY_D1
19B6' B7                        OR     A
19B7' C9                        RET

19B8' CD 04E3'    CMD_R:        CALL   GET_INT2      ; Comando R: verifica memoria RAM
19BB' CD 19D5'    CMD_R1:       CALL   RAMTST        ; Efetua os testes na RAM
19BE' D0                        RET    NC            ; Erro ?
                  +             OUT_HL
19C0' CD 0406'                  CALL   WHITE2        ; Imprime o valor e o endereco onde
19C3' 7E                        LD     A,(HL)        ;       ocorreu o erro
19C4' CD 03EB'                  CALL   OUT_A
19C7' CD 030A'                  CALL   CR_LF
19CA' 23                        INC    HL
19CB' EB                        EX     DE,HL
19CC' 09                        ADD    HL,BC
19CD' A7                        AND    A
19CE' ED 52                     SBC    HL,DE
19D0' 44                        LD     B,H
19D1' 4D                        LD     C,L
19D2' EB                        EX     DE,HL
19D3' 18 E6                     JR     CMD_R1

19D5' 78          RAMTST:       LD     A,B           ; Rotina de teste na RAM
19D6' B1                        OR     C
19D7' C8                        RET    Z
19D8' 97                        SUB    A
19D9' CD 1A03'                  CALL   FILCMP        ; Preenche a memoria e testa com 0
```

```
  19DC'   D8                            RET     C
  19DD'   3E FF                         LD      A,0FFH
  19DF'   CD 1A03'                      CALL    FILCMP         ; Preenche e testa com FFH
  19E2'   D8                            RET     C
  19E3'   3E AA                         LD      A,0AAH
  19E5'   CD 1A03'                      CALL    FILCMP         ; Preenche e testa com AAH
  19E8'   D8                            RET     C
  19E9'   3E 55                         LD      A,55H
  19EB'   CD 1A03'                      CALL    FILCMP         ; Preenche e testa com 55H
  19EE'   D8                            RET     C
  19EF'   3E 80         WLKLP:          LD      A,80H          ; Testa um a um os bits
  19F1'   77            WLKLP1:         LD      (HL),A
  19F2'   BE                                    CP      (HL)
  19F3'   37                            SCF
  19F4'   C0                            RET     NZ
  19F5'   0F                            RRCA
  19F6'   FE 80                         CP      80H
  19F8'   20 F7                         JR      NZ,WLKLP1
  19FA'   36 00                         LD      (HL),0
  19FC'   23                            INC     HL
  19FD'   0B                            DEC     BC
  19FE'   78                            LD      A,B
  19FF'   B1                            OR      C
  1A00'   20 ED                         JR      NZ,WLKLP
  1A02'   C9                            RET

  1A03'   E5            FILCMP:         PUSH    HL             ; Preenche a memoria e testa com
  1A04'   C5                            PUSH    BC             ; um valor
  1A05'   5F                            LD      E,A
  1A06'   77                            LD      (HL),A         ; Coloca o valor na primeira
  1A07'   0B                            DEC     BC
  1A08'   78                            LD      A,B
  1A09'   B1                            OR      C              ;       posicao da memoria
  1A0A'   7B                            LD      A,E
  1A0B'   28 05                         JR      Z,COMPARE
  1A0D'   54                            LD      D,H
  1A0E'   5D                            LD      E,L
  1A0F'   13                            INC     DE
  1A10'   ED B0                         LDIR                   ; preenche o resto com o primeiro valor
  1A12'   C1            COMPARE:        POP     BC             ; Compara com o valor colocado
  1A13'   E1                            POP     HL
  1A14'   E5                            PUSH    HL
  1A15'   C5                            PUSH    BC
  1A16'   ED A1         CMPLP:          CPI
  1A18'   20 07                         JR      NZ,CMPER
  1A1A'   EA 1A16'                      JP      PE,CMPLP
  1A1D'   C1                            POP     BC
  1A1E'   E1                            POP     HL
  1A1F'   B7                            OR      A
  1A20'   C9                            RET

  1A21'   C1            CMPER:          POP     BC             ; Erro durante a comparacao
  1A22'   D1                            POP     DE
  1A23'   2B                            DEC     HL
  1A24'   37                            SCF
  1A25'   C9                            RET


                        ;[[[[[[[[[[[[[[[[[[[[[[[[[[[[[]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                        ;
                        ;       TABELAS PARA MONTAR E DESMONTAR AS INSTRUCOES
                        ;
```

```
                        ;[[[[[[[[[[[[[[[[[[[[[[[[[[[[]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
                        ;
                        +TAB1           MACRO   ?END, ?VAL      ;; Macro para montagem das tabelas
                        +               DEFW    ?END            ;; Endereco e valor
                        +               DEFB    ?VAL
                                        ENDM

1A26'                   +TABELA1:       TAB1    CMD_A_5,  088H  ;; ADC
                        +               TAB1    CMD_A_6,  080H  ;; ADD
                        +               TAB1    CMD_A_7,  0A0H  ;; AND
                        +               TAB1    CMD_A_9,  040H  ;; BIT
                        +               TAB1    CMD_A_10, 0C4H  ;; CALL
                        +               TAB1    CMD_A_2,  03FH  ;; CCF
                        +               TAB1    CMD_A_7,  0B8H  ;; CP
                        +               TAB1    CMD_A_3,  0A9H  ;; CPD
                        +               TAB1    CMD_A_3,  0B9H  ;; CPDR
                        +               TAB1    CMD_A_3,  0A1H  ;; CPI
                        +               TAB1    CMD_A_3,  0B1H  ;; CPIR
                        +               TAB1    CMD_A_2,  02FH  ;; CPL
                        +               TAB1    CMD_A_2,  027H  ;; DAA
                        +               TAB1    CMD_A_21, 005H  ;; DEC
                        +               TAB1    CMD_A_2,  0F3H  ;; DI
                        +               TAB1    CMD_A_13, 010H  ;; DJNZ
                        +               TAB1    CMD_A_2,  0FBH  ;; EI
                        +               TAB1    CMD_A_20, 0E3H  ;; EX
                        +               TAB1    CMD_A_2,  0D9H  ;; EXX
                        +               TAB1    CMD_A_2,  076H  ;; HALT
                        +               TAB1    CMD_A_15, 046H  ;; IM
                        +               TAB1    CMD_A_18, 040H  ;; IN
                        +               TAB1    CMD_A_21, 004H  ;; INC
                        +               TAB1    CMD_A_3,  0AAH  ;; IND
                        +               TAB1    CMD_A_3,  0BAH  ;; INDR
                        +               TAB1    CMD_A_3,  0A2H  ;; INI
                        +               TAB1    CMD_A_3,  0B2H  ;; INIR
                        +               TAB1    CMD_A_12, 0C2H  ;; JP
                        +               TAB1    CMD_A_14, 020H  ;; JR
                        +               TAB1    CMD_A_1,  040H  ;; LD
                        +               TAB1    CMD_A_3,  0A8H  ;; LDD
                        +               TAB1    CMD_A_3,  0B8H  ;; LDDR
                        +               TAB1    CMD_A_3,  0A0H  ;; LDI
                        +               TAB1    CMD_A_3,  0B0H  ;; LDIR
                        +               TAB1    CMD_A_3,  044H  ;; NEG
                        +               TAB1    CMD_A_2,  000H  ;; NOP
                        +               TAB1    CMD_A_7,  0B0H  ;; OR
                        +               TAB1    CMD_A_3,  0BBH  ;; OTDR
                        +               TAB1    CMD_A_3,  0B3H  ;; OTIR
                        +               TAB1    CMD_A_19, 041H  ;; OUT
                        +               TAB1    CMD_A_3,  0ABH  ;; OUTD
                        +               TAB1    CMD_A_3,  0A3H  ;; OUTI
                        +               TAB1    CMD_A_17, 0C1H  ;; POP
                        +               TAB1    CMD_A_17, 0C5H  ;; PUSH
                        +               TAB1    CMD_A_9,  080H  ;; RES
                        +               TAB1    CMD_A_11, 0C0H  ;; RET
                        +               TAB1    CMD_A_3,  04DH  ;; RETI
                        +               TAB1    CMD_A_3,  045H  ;; RETN
                        +               TAB1    CMD_A_8,  010H  ;; RL
                        +               TAB1    CMD_A_2,  017H  ;; RLA
                        +               TAB1    CMD_A_8,  000H  ;; RLC
                        +               TAB1    CMD_A_2,  007H  ;; RLCA
                        +               TAB1    CMD_A_3,  06FH  ;; RLD
                        +               TAB1    CMD_A_8,  018H  ;; RR
                        +               TAB1    CMD_A_2,  01FH  ;; RRA
```

```
                        +               TAB1    CMD_A_8,  009H  ;; RRC
                        +               TAB1    CMD_A_2,  00FH  ;; RRCA
                        +               TAB1    CMD_A_3,  067H  ;; RRD
                        +               TAB1    CMD_A_16, 0C7H  ;; RST
                        +               TAB1    CMD_A_5,  098H  ;; SBC
                        +               TAB1    CMD_A_2,  037H  ;; SCF
                        +               TAB1    CMD_A_9,  0C0H  ;; SET
                        +               TAB1    CMD_A_8,  020H  ;; SLA
                        +               TAB1    CMD_A_8,  028H  ;; SRA
                        +               TAB1    CMD_A_8,  038H  ;; SRL
                        +               TAB1    CMD_A_7,  090H  ;; SUB
                        +               TAB1    CMD_A_7,  0A8H  ;; XOR
                        +               TAB1    CMD_A_4,  0CFH  ;; JSYS

                         ;      Instrucoes Grupo DD ou FD, com deslocamento
                         ;              instrucoes complicadas
                         ;
1AF2' 34 35 36           TABELA2:       DEFB    034H, 035H, 036H, 046H, 04EH, 056H, 05EH, 066H, 06EH
1AF5' 46 4E 56
1AF8' 5E 66 6E
1AFB' 70 71 72                          DEFB    070H, 071H, 072H, 073H, 074H, 075H, 076H, 077H, 07EH
1AFE' 73 74 75
1B01' 76 77 7E
1B04' 86 8E 96                          DEFB    086H, 08EH, 096H, 09EH, 0A6H, 0AEH, 0B6H, 0BEH, 000H
1B07' 9E A6 AE
1B0A' B6 BE 00

                         ;      Instrucoes Grupo DD ou FD com ou sem endereco
                         ;
1B0D' 09 19 21           TABELA3:       DEFB    009H, 019H, 021H, 022H, 023H, 029H, 02AH, 02BH, 039H
1B10' 22 23 29
1B13' 2A 2B 39
1B16' E1 E3 E5                          DEFB    0E1H, 0E3H, 0E5H, 0E9H, 0F9H, 000H
1B19' E9 F9 00

                         ;      Instrucoes de comandos de tamanho 1 sem rotina de
                         ;              tratamento
                         ;
1B1C' 76 39              TABELA4:       DEFB    076H, 039H      ; HALT
1B1E' D9 36                             DEFB    0D9H, 036H      ; EXX
1B20' F3 2C                             DEFB    0F3H, 02CH      ; DI
1B22' FB 32                             DEFB    0FBH, 032H      ; EI
1B24' 00 69                             DEFB    000H, 069H      ; NOP
1B26' 07 9E                             DEFB    007H, 09EH      ; RLCA
1B28' 0F AD                             DEFB    00FH, 0ADH      ; RRCA
1B2A' 17 98                             DEFB    017H, 098H      ; RLA
1B2C' 1F A7                             DEFB    01FH, 0A7H      ; RRA
1B2E' 27 26                             DEFB    027H, 026H      ; DAA
1B30' 2F 23                             DEFB    02FH, 023H      ; CPL
1B32' 37 BA                             DEFB    037H, 0BAH      ; SCF
1B34' 3F 10                             DEFB    03FH, 010H      ; CCF
1B36' FF                                DEFB    0FFH

                        +TAB            MACRO   ?BAND, ?BRES, ?DESL, ?END
                        +               DEFB    ?BAND, ?BRES, ?DESL
                        +               DEFW    ?END
                                        ENDM

                         ;      Tabela de comandos de tamanho 1
                         ;
1B37'                   +TABELA5:       TAB     0C0H, 040H, 056H, ANAL_RS
                        +               TAB     0F8H, 080H, 003H, ANAL_AR
```

```
                    +           TAB     0F8H, 088H, 000H, ANAL_AR
                    +           TAB     0F8H, 090H, 0C9H, ANAL_AR
                    +           TAB     0F8H, 098H, 0B7H, ANAL_AR
                    +           TAB     0F8H, 0A0H, 006H, ANAL_AR
                    +           TAB     0F8H, 0A8H, 0CCH, ANAL_AR
                    +           TAB     0F8H, 0B0H, 06CH, ANAL_AR
                    +           TAB     0F8H, 0B8H, 013H, ANAL_AR
                    +           TAB     0C7H, 0C0H, 089H, ANAL_CC
                    +           TAB     0C7H, 0C7H, 0B4H, ANAL_B
                    +           TAB     0FFH, 0C9H, 08BH, PROC_MEN2
                    +           TAB     0CFH, 0C1H, 081H, ANAL_DD_DD
                    +           TAB     0CFH, 0C5H, 084H, ANAL_DD_DD
                    +           TAB     0FFH, 0E3H, 034H, ANAL_SP_C
                    +           TAB     0FFH, 0E9H, 052H, ANAL_HL_C
                    +           TAB     0FFH, 0EBH, 034H, ANAL_DE_HL
                    +           TAB     0FFH, 0F9H, 056H, ANAL_SP_HL
                    +           TAB     0CFH, 003H, 041H, ANAL_DD
                    +           TAB     0CFH, 00BH, 029H, ANAL_DD
                    +           TAB     0C7H, 004H, 041H, ANALBIT5_3
                    +           TAB     0C7H, 005H, 029H, ANALBIT5_3
                    +           TAB     0FFH, 008H, 034H, ANAL_AF_AF
                    +           TAB     0CFH, 009H, 003H, ANAL_HL_DD
                    +           TAB     0EFH, 002H, 056H, ANAL_DD_C_A
                    +           TAB     0EFH, 00AH, 056H, ANAL_A_DD_C
1BB9'  00                       DEFB    0

                    ;       Tabela de comandos de tamanho 2
                    ;
1BBA'               +TABELA6:   TAB     0C7H, 006H, 056H, ANAL_RN
                    +           TAB     0FFH, 0C6H, 003H, ANAL_A_N
                    +           TAB     0FFH, 0CEH, 000H, ANAL_A_N
                    +           TAB     0FFH, 0D6H, 0C9H, ANAL_A_N
                    +           TAB     0FFH, 0DEH, 0B7H, ANAL_A_N
                    +           TAB     0FFH, 0E6H, 006H, ANAL_A_N
                    +           TAB     0FFH, 0EEH, 0CCH, ANAL_A_N
                    +           TAB     0FFH, 0F6H, 06CH, ANAL_A_N
                    +           TAB     0FFH, 0FEH, 013H, ANAL_A_N
                    +           TAB     0FFH, 010H, 02EH, ANAL_ADD_REL
                    +           TAB     0FFH, 018H, 054H, ANAL_ADD_REL
                    +           TAB     0E7H, 020H, 054H, ANAL_CC_JR
                    +           TAB     0FFH, 0D3H, 076H, ANAL_N_A
                    +           TAB     0FFH, 0DBH, 03FH, ANAL_A_N
                    +           TAB     0FFH, 0CFH, 0CFH, ANAL_N
1C05'  00                       DEFB    0

                    ;       Tabela de comandos de tamanho 3
                    ;
1C06'               +TABELA7:   TAB     0C7H, 0C2H, 052H, ANAL_CC_END
                    +           TAB     0C7H, 0C4H, 00CH, ANAL_CC_END
                    +           TAB     0CFH, 001H, 056H, ANAL_DD_VAL16
                    +           TAB     0FFH, 0C3H, 052H, ANAL_ENDER
                    +           TAB     0FFH, 0CDH, 00CH, ANAL_ENDER
                    +           TAB     0FFH, 022H, 056H, ANAL_END_C_HL
                    +           TAB     0FFH, 02AH, 056H, ANAL_HL_END_C
                    +           TAB     0FFH, 032H, 056H, ANAL_END_CC_A
                    +           TAB     0FFH, 03AH, 056H, ANAL_A_ENDER_C
1C33'  00                       DEFB    0

                ; Grupo ED, instrucoes de 1 byte apos ED, sem rotina de tratamento

1C34'  44 66    TABELA8:        DEFB    044H, 066H      ; NEG  -
1C36'  45 92                    DEFB    045H, 092H      ; RETN
```

```
1C38'  4D 8E                    DEFB   04DH, 08EH      ; RETI
1C3A'  67 B1                    DEFB   067H, 0B1H      ; RRD
1C3C'  6F A2                    DEFB   06FH, 0A2H      ; RLD
1C3E'  A0 5F                    DEFB   0A0H, 05FH      ; LDI
1C40'  A1 1C                    DEFB   0A1H, 01CH      ; CPI
1C42'  A2 4B                    DEFB   0A2H, 04BH      ; INI
1C44'  A3 7D                    DEFB   0A3H, 07DH      ; OUTI
1C46'  A8 58                    DEFB   0A8H, 058H      ; LLD
1C48'  A9 15                    DEFB   0A9H, 015H      ; CPD
1C4A'  AA 44                    DEFB   0AAH, 044H      ; IND
1C4C'  AB 79                    DEFB   0ABH, 079H      ; OUTD
1C4E'  B0 62                    DEFB   0B0H, 062H      ; LDIR
1C50'  B1 1F                    DEFB   0B1H, 01FH      ; CPIR
1C52'  B2 4E                    DEFB   0B2H, 04EH      ; INIR
1C54'  B3 72                    DEFB   0B3H, 072H      ; OTIR
1C56'  B8 5B                    DEFB   0B8H, 05BH      ; LDDR
1C58'  B9 18                    DEFB   0B9H, 018H      ; CPDR
1C5A'  BA 47                    DEFB   0BAH, 047H      ; INDR
1C5C'  BB 6E                    DEFB   0BBH, 06EH      ; OUTDR
1C5E'  FF                       DEFB   0FFH

                        ;       Tabela grupo ED, tamanho 2, com rotina

1C5F'                   +TABELA9:       TAB    0E7H, 040H, 03FH, ANAL_R_C_C    ; IN reg,(C)
                        +               TAB    0F7H, 060H, 03FH, ANAL_R_C_C    ; "
                        +               TAB    0FFH, 078H, 03FH, ANAL_R_C_C    ; "
                        +               TAB    0E7H, 041H, 076H, ANAL_C_C_R    ; OUT (C),reg
                        +               TAB    0F7H, 061H, 076H, ANAL_C_C_R    ; "
                        +               TAB    0FFH, 079H, 076H, ANAL_C_C_R    ; "
                        +               TAB    0CFH, 042H, 0B7H, ANAL_HL_DD    ; SBC HL,reg
                        +               TAB    0CFH, 04AH, 000H, ANAL_HL_DD    ; ADC HL,reg
                        +               TAB    0FFH, 046H, 03DH, ANAL_IM0      ; IM0
                        +               TAB    0FFH, 056H, 03DH, ANAL_IM1      ; IM1
                        +               TAB    0FFH, 05EH, 03DH, ANAL_IM2      ; IM2
                        +               TAB    0FFH, 047H, 056H, ANAL_I_A      ; LD I,A
                        +               TAB    0FFH, 057H, 056H, ANAL_A_I      ; LD A,I
                        +               TAB    0FFH, 04FH, 056H, ANAL_R_A      ; LD R,A
                        +               TAB    0FFH, 05FH, 056H, ANAL_A_R      ; LD A,R
1CAA'  00                               DEFB   0

                        ;       Tabela grupo ED, com tamanho 4, com rotina
                        ;
1CAB'                   +TABEL10:       TAB    0EFH, 043H, 056H, ANAL_END_C_DD ; LD (End),BC
                        +               TAB    0FFH, 073H, 056H, ANAL_END_C_DD ; LD (End),HL
                        +               TAB    0EFH, 04BH, 056H, ANAL_DD_END_C ; LD BC,(End)
                        +               TAB    0FFH, 07BH, 056H, ANAL_DD_END_C ; LD HL,(End)
1CBF'  00                               DEFB   0

                        ;       Grupo CB 2 bytes
                        ;
1CC0'                   +TABEL11:       TAB    0F8H, 000H, 09BH, ANAL_REG      ; RLC reg
                        +               TAB    0F8H, 008H, 0AAH, ANAL_REG      ; RRC reg
                        +               TAB    0F8H, 010H, 096H, ANAL_REG      ; RL reg
                        +               TAB    0F8H, 018H, 0A5H, ANAL_REG      ; RR reg
                        +               TAB    0F8H, 020H, 0C0H, ANAL_REG      ; SLA reg
                        +               TAB    0F8H, 028H, 0C3H, ANAL_REG      ; SRA reg
                        +               TAB    0F8H, 038H, 0C6H, ANAL_REG      ; SRL reg
                        +               TAB    0C0H, 040H, 009H, ANAL_B_R      ; BIT b,reg
                        +               TAB    0C0H, 080H, 088H, ANAL_B_R      ; RES b,reg
                        +               TAB    0C0H, 0C0H, 0BDH, ANAL_B_R      ; SET b,reg
1CF2'  00                               DEFB   0
```

```
                        ;       Grupo de instrucoes ou
                        ;       instrucoes de mudanca de fluxo
                        ;
1CF3'                   +TABEL12:       TAB     0FFH, 0DDH, 000H, ANAL_IX       ; Grupo DD
                        +               TAB     0FFH, 0FDH, 000H, ANAL_IY       ; Grupo FD
                        +               TAB     0FFH, 0EDH, 000H, ANAL_ED       ; Grupo ED
1D02'                   +TABEL13:       TAB     0FFH, 0CDH, 000H, ANAL_CALL     ; CALL
                        +               TAB     0FFH, 0C3H, 000H, ANAL_JUMP     ; JP
                        +               TAB     0FFH, 0E9H, 000H, ANAL_JUMP_HL  ; JP (HL)
                        +               TAB     0FFH, 0C9H, 000H, ANAL_RET      ; RET
                        +               TAB     0FFH, 0CFH, 000H, ANAL_JSYS     ; RST 8 (JSYS)
                        +               TAB     0C7H, 0C7H, 000H, ANAL_RST      ; RST 0
                        +               TAB     0C7H, 0C4H, 000H, ANAL_CALL     ; RST N
                        +               TAB     0F7H, 010H, 000H, ANAL_JR       ; DJNZ, JR
                        +               TAB     0E7H, 020H, 000H, ANAL_JR       ; JR cond
                        +               TAB     0C7H, 0C2H, 000H, ANAL_JUMP     ; JP cond
                        +               TAB     0C7H, 0C0H, 000H, ANAL_RET_C    ; RET cond
1D39'   00                              DEFB    0

1D3A'   0008            TABEL14:        DEFW    8
1D3C'   00EB                            DEFW    0EBH
1D3E'   00E3                            DEFW    0E3H
1D40'   E3DD                            DEFW    0E3DDH
1D42'   E3FD                            DEFW    0E3FDH

1D44'   28 48 4C        INSTR1:         DC      '(HL)'
1D47'   A9
1D48'   28 49 58                        DC      '(IX)'
1D4B'   A9
1D4C'   28 49 59                        DC      '(IY)'
1D4F'   A9
1D50'   00                              DEFB    0

1D51'   41 46 2C        INSTR2:         DC      "AF,AF'"
1D54'   41 46 A7
1D57'   44 45 2C        INSTR3:         DC      'DE,HL'
1D5A'   48 CC
1D5C'   28 53 50                        DC      '(SP),HL'
1D5F'   29 2C 48
1D62'   CC
1D63'   28 53 50                        DC      '(SP).IX'
1D66'   29 2C 49
1D69'   D8
1D6A'   28 53 50                        DC      '(SP),IY'
1D6D'   29 2C 49
1D70'   D9
1D71'   00                              DEFB    0

1D72'   28 53 50        INSTR4:         DC      '(SP),'
1D75'   29 AC
1D77'   53 50 AC        INSTR5:         DC      'SP,'
1D7A'   49 2C C1        INSTR6:         DC      'I,A'
1D7D'   41 2C C9        INSTR7:         DC      'A,I'
1D80'   52 2C C1        INSTR8:         DC      'R,A'
1D83'   41 2C D2        INSTR9:         DC      'A,R'
1D86'   28 49 D8        INST10:         DC      '(IX'
1D89'   28 49 D9        INST11:         DC      '(IY'

1D8C'   41 44 C3        INSTR:          DC      'ADC'           ; Tabela de Instrucoes
1D8F'   41 44 C4                        DC      'ADD'
1D92'   41 4E C4                        DC      'AND'
1D95'   42 49 D4                        DC      'BIT'
```

```
1D98'  43 41 4C                  DC      'CALL'
1D9B'  CC
1D9C'  43 43 C6                  DC      'CCF'
1D9F'  43 D0                     DC      'CP'
1DA1'  43 50 C4                  DC      'CPD'
1DA4'  43 50 44                  DC      'CPDR'
1DA7'  D2
1DA8'  43 50 C9                  DC      'CPI'
1DAB'  43 50 49                  DC      'CPIR'
1DAE'  D2
1DAF'  43 50 CC                  DC      'CPL'
1DB2'  44 41 C1                  DC      'DAA'
1DB5'  44 45 C3                  DC      'DEC'
1DB8'  44 C9                     DC      'DI'
1DBA'  44 4A 4E                  DC      'DJNZ'
1DBD'  DA
1DBE'  45 C9                     DC      'EI'
1DC0'  45 D8                     DC      'EX'
1DC2'  45 58 D8                  DC      'EXX'
1DC5'  48 41 4C                  DC      'HALT'
1DC8'  D4
1DC9'  49 CD                     DC      'IM'
1DCB'  49 CE                     DC      'IN'
1DCD'  49 4E C3                  DC      'INC'
1DD0'  49 4E C4                  DC      'IND'
1DD3'  49 4E 44                  DC      'INDR'
1DD6'  D2
1DD7'  49 4E C9                  DC      'INI'
1DDA'  49 4E 49                  DC      'INIR'
1DDD'  D2
1DDE'  4A D0                     DC      'JP'
1DE0'  4A D2                     DC      'JR'
1DE2'  4C C4                     DC      'LD'
1DE4'  4C 44 C4                  DC      'LDD'
1DE7'  4C 44 44                  DC      'LDDR'
1DEA'  D2
1DEB'  4C 44 C9                  DC      'LDI'
1DEE'  4C 44 49                  DC      'LDIR'
1DF1'  D2
1DF2'  4E 45 C7                  DC      'NEG'
1DF5'  4E 4F D0                  DC      'NOP'
1DF8'  4F D2                     DC      'OR'
1DFA'  4F 54 44                  DC      'OTDR'
1DFD'  D2
1DFE'  4F 54 49                  DC      'OTIR'
1E01'  D2
1E02'  4F 55 D4                  DC      'OUT'
1E05'  4F 55 54                  DC      'OUTD'
1E08'  C4
1E09'  4F 55 54                  DC      'OUTI'
1E0C'  C9
1E0D'  50 4F D0                  DC      'POP'
1E10'  50 55 53                  DC      'PUSH'
1E13'  C8
1E14'  52 45 D3                  DC      'RES'
1E17'  52 45 D4                  DC      'RET'
1E1A'  52 45 54                  DC      'RETI'
1E1D'  C9
1E1E'  52 45 54                  DC      'RETN'
1E21'  CE
1E22'  52 CC                     DC      'RL'
1E24'  52 4C C1                  DC      'RLA'
```

```
1E27'  52 4C C3                  DC     'RLC'
1E2A'  52 4C 43                  DC     'RLCA'
1E2D'  C1
1E2E'  52 4C C4                  DC     'RLD'
1E31'  52 D2                     DC     'RR'
1E33'  52 52 C1                  DC     'RRA'
1E36'  52 52 C3                  DC     'RRC'
1E39'  52 52 43                  DC     'RRCA'
1E3C'  C1
1E3D'  52 52 C4                  DC     'RRD'
1E40'  52 53 D4                  DC     'RST'
1E43'  53 42 C3                  DC     'SBC'
1E46'  53 43 C6                  DC     'SCF'
1E49'  53 45 D4                  DC     'SET'
1E4C'  53 4C C1                  DC     'SLA'
1E4F'  53 52 C1                  DC     'SRA'
1E52'  53 52 CC                  DC     'SRL'
1E55'  53 55 C2                  DC     'SUB'
1E58'  58 4F D2                  DC     'XOR'
1E5B'  4A 53 59                  DC     'JSYS'
1E5E'  D3
1E5F'  00                        DEFB   0

1E60'  C2 C3 C4     REGS1:       ASC    "BCDEHL"
1E63'  C5 C8 CC
1E66'  28 48 4C                  DC     '(HL)'
1E69'  A9
1E6A'  C1                        ASC    "A"
1E6B'  00                        DEFB   0

1E6C'  42 C3        REGS2:       DC     'BC'
1E6E'  44 C5                     DC     'DE'
1E70'  48 CC                     DC     'HL'
1E72'  53 D0                     DC     'SP'
1E74'  00                        DEFB   0

1E75'  42 C3        REGS3:       DC     'BC'
1E77'  44 C5                     DC     'DE'
1E79'  48 CC        REGS4:       DC     'HL'
1E7B'  41 C6                     DC     'AF'
1E7D'  00                        DEFB   0

1E7E'  42 C3        REGS5:       DC     'BC'
1E80'  44 C5                     DC     'DE'
1E82'  49 D9                     DC     'IY'
1E84'  53 D0                     DC     'SP'
1E86'  00                        DEFB   0

1E87'  42 C3        REGS6:       DC     'BC'
1E89'  44 C5                     DC     'DE'
1E8B'  49 D8                     DC     'IX'
1E8D'  53 D0                     DC     'SP'
1E8F'  00                        DEFB   0

1E90'  48 CC        REGS7:       DC     'HL'

1E92'  49 D8        REGS8:       DC     'IX'
1E94'  49 D9                     DC     'IY'
1E96'  00                        DEFB   0

1E97'  4E DA        FLAGS1:      DC     'NZ'
1E99'  DA                        DC     'Z'
```

```
1E9A'  4E C3                        DC      'NC'
1E9C'  C3                           DC      'C'
1E9D'  4E C5                        DC      'NE'
1E9F'  45 D1                        DC      'EQ'
1EA1'  47 C5                        DC      'GE'
1EA3'  4C D4                        DC      'LT'
1EA5'  00                           DEFB    0

1EA6'  4E DA        FLAGS2:         DC      'NZ'
1EA8'  DA                           DC      'Z'
1EA9'  4E C3                        DC      'NC'
1EAB'  C3                           DC      'C'
1EAC'  50 CF                        DC      'PO'
1EAE'  50 C5                        DC      'PE'
1EB0'  D0                           DC      'P'
1EB1'  CD                           DC      'M'
1EB2'  4E C5                        DC      'NE'
1EB4'  45 D1                        DC      'EQ'
1EB6'  47 C5                        DC      'GE'
1EB8'  4C D4                        DC      'LT'
1EBA'  4E D6                        DC      'NV'
1EBC'  D6                           DC      'V'
1EBD'  00                           DEFB    0

1EBE'  28 43 A9     IO_P_C:         DC      '(C)'           ; OUT (C),n  , IN (C)
1EC1'  00                           DEFB    0

1EC2'  41 A0        N_REG_P:        DC      'A '
1EC4'  42 43 A0                     DC      'BC '
1EC7'  44 45 A0                     DC      'DE '
1ECA'  48 4C A0                     DC      'HL '
1ECD'  53 D0                        DC      'SP'
1ECF'  50 C3                        DC      'PC'
1ED1'  41 A7                        DC      "A'"
1ED3'  42 43 A7                     DC      "BC'"
1ED6'  44 45 A7                     DC      "DE'"
1ED9'  48 4C A7                     DC      "HL'"
1EDC'  49 D8                        DC      'IX'
1EDE'  49 D9                        DC      'IY'
1EE0'  C9                           DC      'I'
1EE1'  00                           DEFB    0

               +REG                 MACRO   ?END,?TYP       ;; Macro para montar tabela de
               +                    DEFW    REG_&?END       ;;      registradores principais
               +                    DEFB    ?TYP
                                    ENDM

1EE2'          +SIZ_R_PR:           REG     A, 0            ; Registradores Principais
               +                    REG     BC,1            ;  e seu tamanho ( Byte, Word e @Word )
               +                    REG     DE,1
               +                    REG     HL,1
               +                    REG     SP,1
               +                    REG     PC,2
               +                    REG     A_, 0
               +                    REG     BC_,1
               +                    REG     DE_,1
               +                    REG     HL_,1
               +                    REG     IX,1
               +                    REG     IY,1
               +                    REG     I, 0
1F09'  00                           DEFB    0
```

```
1F0A'  42 43 A7    NAME_REG:    DC    "BC'"        ; Nome de todos registradores
1F0D'  44 45 A7                 DC    "DE'"        ;  em sua forma de byte ou word
1F10'  48 4C A7                 DC    "HL'"        ; Apostrofe significa registradores
1F13'  42 C3                    DC    'BC'         ;  alternativos
1F15'  44 C5                    DC    'DE'
1F17'  48 CC                    DC    'HL'
1F19'  41 A7                    DC    "A'"
1F1B'  42 A7                    DC    "B'"
1F1D'  43 A7                    DC    "C'"
1F1F'  44 A7                    DC    "D'"
1F21'  45 A7                    DC    "E'"
1F23'  48 A7                    DC    "H'"
1F25'  4C A7                    DC    "L'"
1F27'  C1                       DC    'A'
1F28'  C2                       DC    'B'
1F29'  C3                       DC    'C'
1F2A'  C4                       DC    'D'
1F2B'  C5                       DC    'E'
1F2C'  C8                       DC    'H'
1F2D'  CC                       DC    'L'
1F2E'  49 D8                    DC    'IX'
1F30'  49 D9                    DC    'IY'
1F32'  53 D0                    DC    'SP'
1F34'  50 C3                    DC    'PC'
1F36'  D8                       DC    'X'
1F37'  D9                       DC    'Y'
1F38'  D3                       DC    'S'
1F39'  D0                       DC    'P'
1F3A'  C9                       DC    'I'
1F3B'  49 D0                    DC    'IP'
1F3D'  46 A7                    DC    "F'"
1F3F'  C6                       DC    'F'
1F40'  00                       DEFB  0

               +REGI            MACRO ?CHAR, ?END  ;; Macro para armazenar os registr.
               +                DEFB  ?CHAR
               +                DEFW  REG_&?END
                                ENDM

1F41'          +SIZ_REG:        REGI  1,BC_        ; Tamanho ( Byte, Word ) e endereco dos
               +                REGI  1,DE_        ;  registradores
               +                REGI  1,HL_
               +                REGI  1,BC
               +                REGI  1,DE
               +                REGI  1,HL
               +                REGI  0,A_
               +                REGI  0,BC_+1
               +                REGI  0,BC_
               +                REGI  0,DE_+1
               +                REGI  0,DE_
               +                REGI  0,HL_+1
               +                REGI  0,HL_
               +                REGI  0,A
               +                REGI  0,BC+1
               +                REGI  0,BC
               +                REGI  0,DE+1
               +                REGI  0,DE
               +                REGI  0,HL+1
               +                REGI  0,HL
               +                REGI  1,IX
               +                REGI  1,IY
               +                REGI  1,SP
```

```
                        +               REGI    3,PC
                        +               REGI    1,IX
                        +               REGI    1,IY
                        +               REGI    1,SP
                        +               REGI    3,PC
                        +               REGI    0,I
                        +               REGI    1,I-1
                        +               REGI    0,F_
                        +               REGI    0,F

1FA1'   4D 61 74                        DEFM    'Mateus J. Martins'     ; Fim do monitor
1FA4'   65 75 73
1FA7'   20 4A 2E
1FAA'   20 4D 61
1FAD'   72 74 69
1FB0'   6E 73

                        ;===============================================================;
                                        .PHASE  INIC_RAM
                        ;===============================================================;
                        ;
                        ;               AREA VARIAVEIS
                        ;
4000    0000            END_ERR:        DEFW    0
4002    00              CNT_CAR:        DEFB    0
                        ;
4003    00              TYP_DSK:        DEFB    0
4004    00              TRK_DSK:        DEFB    0
4005    00              SCT_DSK:        DEFB    0
4006    8000            DMA_DSK:        DEFW    POS_INIC
4008    00              PAR_DSK:        DEFB    0
4009    00              STP_DSK:        DEFB    0
400A    00              HST_OP:         DEFB    0
                        ;
400B    0000            TMP_SP:         DEFW    0
400D    0000            TMP_IY:         DEFW    0
400F    00              PRT?:           DEFB    0
4010    00              DUMP?:          DEFB    0
4011    00              TRACE?:         DEFB    0
4012    0000            CNT_TRACE:      DEFW    0
4014    00              VAR_1:          DEFB    0
4015    00              FLAG_N:         DEFB    0
4016    00              FLAG_J:         DEFB    0
4017    00              VAR_T1:         DEFB    0
4018    0000            VAR_T2:         DEFW    0
401A    00              VAR_T3:         DEFB    0
401B    8000            POSICAO:        DEFW    POS_INIC
401D    00              VAR_L1:         DEFB    0
401E    0000            VAR_@:          DEFW    0
4020                    BUF_ID:
4020                    BUF_LIN:        DEFS    TAM_BUF

4070                                    DEFS    50              ; STACK TEMPORARIO

40A2    0000            REG_HL_:        DEFW    0
40A4    0000            REG_DE_:        DEFW    0
40A6    0000            REG_BC_:        DEFW    0
40A8    00              REG_F_:         DEFB    0
40A9    00              REG_A_:         DEFB    0
40AA    00                              DEFB    0
40AB    00              REG_I:          DEFB    0
40AC    0000            REG_IY:         DEFW    0
```

```
40AE  0000        REG_IX:     DEFW    0
40B0  00          REG_F:      DEFB    0
40B1  00          REG_A:      DEFB    0
40B2  0000        REG_BC:     DEFW    0
40B4  0000        REG_DE:     DEFW    0
40B6  0000        REG_HL:     DEFW    0
40B8  42FE        REG_SP:     DEFW    USR_STACK-2
40BA  F3          GO:         DI
40BB  C3                      DEFB    0C3H            ; Jump
40BC  0000        REG_PC:     DEFW    POS_INIC

40BE  00          ROT_RAM:    NOP                     ; Essa rotina sera transferida para
40BF  A7                      AND     A               ;  a RAM durante a inicializacao do
40C0  E1                      POP     HL              ;  monitor. Encontra-se aqui apenas
40C1  23                      INC     HL              ;  para reservar espaco.
40C2  E9                      JP      (HL)

40C3              +BREAKPOINTS:   REPT    12
                  +               DEFW    0, 0, 0
                  +               ENDM

410B              +CMD_NOT:       IRP     X,<E,F,I,J,N,O,P,U,W,X,Y> ;; Instrucoes destinadas ao
                  +CMD_&X:        JP      ERROR                     ;;  usuario
                  +               ENDM

0021               NUM_CMD_NOT    EQU     $-CMD_NOT               ; Numero de comandos do usuario

412C  C9           MORE_JSYS:     RET
412D  0000                        DEFW    0

4200               STACK          EQU     ($+256) AND 0FF00H
4300               USR_STACK      EQU     STACK+100H
                                  .DEPHASE

                                  END     START
```

Macros:

GET_INT	OUTPUT	OUT_HL	PRINT	REG
REGI	SKIP_WHITE	TAB	TAB1	

Symbols:

104B'	ANALBIT2_0	104E'	ANALBIT2_0_A	1043'	ANALBIT5_3
1068'	ANALBT1	1078'	ANALBT2	1080'	ANALBT3
1088'	ANALBT4	0F7B'	ANAL_ADD_REL	0F37'	ANAL_AF_1
0F34'	ANAL_AF_AF	0F0A'	ANAL_AR	0F0D'	ANAL_AR1
0F48'	ANAL_A_DD_C	1026'	ANAL_A_ENDER_C	0FCE'	ANAL_A_I
0F56'	ANAL_A_N	0FD8'	ANAL_A_R	0F10'	ANAL_B
0FF1'	ANAL_B_R	0FFC'	ANAL_B_R1	0FFF'	ANAL_B_R2
1106'	ANAL_CALL	10B8'	ANAL_CC	10BB'	ANAL_CC_A
0F98'	ANAL_CC_END	0F70'	ANAL_CC_JR	103C'	ANAL_C_C_R
10A6'	ANAL_DD	10A9'	ANAL_DD1	0F4B'	ANAL_DD_C
0F43'	ANAL_DD_C_A	109B'	ANAL_DD_DD	0FE9'	ANAL_DD_END_C
0FAF'	ANAL_DD_VAL16	0F28'	ANAL_DE_HL	113E'	ANAL_ED
0F9E'	ANAL_ENDER	0FA4'	ANAL_ENDER1	0FA6'	ANAL_ENDER2
1029'	ANAL_END_C	0FB7'	ANAL_END_CC_A	10A0'	ANAL_END_C_DD
108D'	ANAL_END_C_HL	1093'	ANAL_HL	0F1D'	ANAL_HL_C
0F3A'	ANAL_HL_DD	101E'	ANAL_HL_END_C	0FBC'	ANAL_IM0
0FC0'	ANAL_IM1	0FC4'	ANAL_IM2	112D'	ANAL_IX
1135'	ANAL_IX_IY	1132'	ANAL_IY	0FC9'	ANAL_I_A
1119'	ANAL_JR	1178'	ANAL_JSYS	1111'	ANAL_JUMP
1128'	ANAL_JUMP_HL	0F6A'	ANAL_N	0F6D'	ANAL_N1
0F8A'	ANAL_N_A	0F90'	ANAL_N_A1	0FC6'	ANAL_OUTPUT
0FE6'	ANAL_PRINT	100E'	ANAL_REG	1019'	ANAL_REG1
101C'	ANAL_REG2	115B'	ANAL_RET	114B'	ANAL_RET_C
0F5B'	ANAL_RN	0F02'	ANAL_RS	1178'	ANAL_RST
0FD3'	ANAL_R_A	0FDD'	ANAL_R_C_C	0F18'	ANAL_SP_C
0F2D'	ANAL_SP_HL	0F30'	ANAL__HL	06FD'	ASC_BIN
0008	BCK_SP	0020	BIT_TIME	0C1A'	BREAK
0C3D'	BREAK1	0C6F'	BREAK2	0C76'	BREAK3
0C83'	BREAK4	0C99'	BREAK5	0CA2'	BREAK6
0CC2'	BREAK7	40C3	BREAKPOINTS	0BB9'	BREAK_FREE
0B8F'	BREAK_FREE1	0D17'	BRK_TMP	0D32'	BRK_TMP1
4020	BUF_ID	4020	BUF_LIN	0017	C8255
01E7'	CHECK_E1	01E5'	CHECK_EPROM	0471'	CHK7B
0475'	CHKB	0477'	CHKB_1	0482'	CHKB_2
0485'	CHKB_3	0488'	CHKB_4	048B'	CHKB_5
048C'	CHKB_6	04A1'	CHKB_7	04A4'	CHKB_8
04A8'	CHKB_9	0EF4'	CHKTAB_INS1	0EE3'	CHKTAB_INSTR
04B1'	CHK_APHA	16DD'	CHK_BYTE	1795'	CHK_COMMA
1783'	CHK_COND	1788'	CHK_COND1	177C'	CHK_COND_ALL
072C'	CHK_DEC	179C'	CHK_FECHAR	11E1'	CHK_GRP
11F3'	CHK_GRP1	061D'	CHK_OPR	056F'	CHK_S1
0563'	CHK_S2	0558'	CHK_S3	0541'	CHK_SYM
0463'	CHK_T1	046E'	CHK_T2	0461'	CHK_TAB
096A'	CMD_@	11FB'	CMD_A	120E'	CMD_A1
1228'	CMD_A2	1244'	CMD_A3	00D0	CMD_ABORT
00C0	CMD_ADDR	124D'	CMD_ASS	126A'	CMD_A_1
154B'	CMD_A_10	155A'	CMD_A_10_1	155C'	CMD_A_10_2
1563'	CMD_A_11	156C'	CMD_A_11_1	156E'	CMD_A_11_2
1571'	CMD_A_12	1577'	CMD_A_12_1	15B4'	CMD_A_12_2
158C'	CMD_A_12_3	1592'	CMD_A_12_4	159E'	CMD_A_12_5
15A5'	CMD_A_12_6	15AA'	CMD_A_13	15B1'	CMD_A_14
15BA'	CMD_A_14_1	15BF'	CMD_A_14_2	15C5'	CMD_A_15
15D8'	CMD_A_15_1	15DB'	CMD_A_16	15EA'	CMD_A_17
15F8'	CMD_A_17_1	15FE'	CMD_A_18	1620'	CMD_A_18_1
1626'	CMD_A_18_2	162C'	CMD_A_18_3	1632'	CMD_A_19
164A'	CMD_A_19_1	12AD'	CMD_A_1_1	1308'	CMD_A_1_10
1318'	CMD_A_1_11	132B'	CMD_A_1_12	133E'	CMD_A_1_13

1346'	CMD_A_1_14	134A'	CMD_A_1_15	1357'	CMD_A_1_16
1358'	CMD_A_1_17	1364'	CMD_A_1_18	137E'	CMD_A_1_19
12B3'	CMD_A_1_2	1389'	CMD_A_1_20	139B'	CMD_A_1_21
13B4'	CMD_A_1_22	13CC'	CMD_A_1_23	13DB'	CMD_A_1_24
13E4'	CMD_A_1_25	13FB'	CMD_A_1_26	1400'	CMD_A_1_27
140D'	CMD_A_1_28	1421'	CMD_A_1_29	12C2'	CMD_A_1_3
1429'	CMD_A_1_30	1431'	CMD_A_1_31	1437'	CMD_A_1_32
1440'	CMD_A_1_33	12CB'	CMD_A_1_4	12DA'	CMD_A_1_5
12E1'	CMD_A_1_6	12E8'	CMD_A_1_7	12EC'	CMD_A_1_8
1304'	CMD_A_1_9	1453'	CMD_A_2	165B'	CMD_A_20
167F'	CMD_A_21	169A'	CMD_A_21_1	16A3'	CMD_A_21_2
16A6'	CMD_A_21_3	16B0'	CMD_A_21_4	16B3'	CMD_A_21_5
16B6'	CMD_A_21_6	145C'	CMD_A_3	1469'	CMD_A_4
146F'	CMD_A_5	148A'	CMD_A_5_1	148C'	CMD_A_5_2
148F'	CMD_A_6	14AC'	CMD_A_6_1	14B4'	CMD_A_6_2
14B8'	CMD_A_6_3	14C6'	CMD_A_7	14D5'	CMD_A_7_1
14D6'	CMD_A_7_2	14E9'	CMD_A_7_3	14ED'	CMD_A_7_4
14F3'	CMD_A_8	1502'	CMD_A_8_1	1518'	CMD_A_8_2
1519'	CMD_A_8_3	1527'	CMD_A_9	1542'	CMD_A_9_1
16E4'	CMD_A_GET_INT	0AD7'	CMD_B	0AE3'	CMD_B1
0AEE'	CMD_B2	0B4A'	CMD_BL	0B50'	CMD_BL1
0B64'	CMD_BL2	0B81'	CMD_BL3	0B16'	CMD_BX
0B1A'	CMD_BX1	0B2B'	CMD_BX2	0B29'	CMD_BX3
0B2F'	CMD_BX4	0B3D'	CMD_BX5	0B41'	CMD_BX6
0BA1'	CMD_B_CNT	117A'	CMD_C	0A6C'	CMD_D
0A78'	CMD_D1	19A7'	CMD_DSK	075C'	CMD_DUMP1
0779'	CMD_DUMP2	0789'	CMD_DUMP3	074E'	CMD_DUMP_REG
0AB6'	CMD_D_M1	0A92'	CMD_D_M2	0AAA'	CMD_D_M3
0AB0'	CMD_D_M4	0ABF'	CMD_D_M5	0ACB'	CMD_D_M6
0AD4'	CMD_D_M7	0A82'	CMD_D_MEM	410B	CMD_E
410E	CMD_F	0BC4'	CMD_G	0BD1'	CMD_G1
0CF9'	CMD_GBR1	0D0D'	CMD_GBR2	0CF3'	CMD_GBREAK
09B8'	CMD_H	09DF'	CMD_H1	0008	CMD_HOME
4111	CMD_I	0953'	CMD_I1	0938'	CMD_I2
4114	CMD_J	17A3'	CMD_K	17F6'	CMD_K5
17FD'	CMD_K8	1866'	CMD_KA	1833'	CMD_KD
17E0'	CMD_KH	184F'	CMD_KL	1825'	CMD_KN
17E5'	CMD_KP	17F0'	CMD_KP1	18AB'	CMD_KR
189D'	CMD_KS	18AF'	CMD_KW	0D5A'	CMD_L
0D60'	CMD_L1	0D70'	CMD_L2	0D8A'	CMD_L3
0D94'	CMD_L4	0DB2'	CMD_LIST	0DC4'	CMD_LIST1
0DCE'	CMD_LIST2	09A7'	CMD_M	4117	CMD_N
410B	CMD_NOT	411A	CMD_O	411D	CMD_P
09E9'	CMD_Q	09F2'	CMD_Q1	09FF'	CMD_Q2
19B8'	CMD_R	19BB'	CMD_R1	0080	CMD_READ
0801'	CMD_S	0812'	CMD_S1	0831'	CMD_S2
0853'	CMD_S3	0018	CMD_SEEK	0869'	CMD_SF
0879'	CMD_SFA	0884'	CMD_SFLAG	0890'	CMD_SFLAG1
08A7'	CMD_SFLAG2	08AF'	CMD_SFLAG3	0932'	CMD_SIO
08C6'	CMD_SM	08CC'	CMD_SM1	08E8'	CMD_SM2
08F4'	CMD_SM3	08FC'	CMD_SM4	0904'	CMD_SM5
0909'	CMD_SM6	0912'	CMD_SM7	091B'	CMD_SM8
091F'	CMD_SM9	117E'	CMD_T	117F'	CMD_T1
1188'	CMD_T2	1191'	CMD_T3	11A1'	CMD_T4
02D1'	CMD_TAB	11C1'	CMD_TRAC1	11A7'	CMD_TRACE
4120	CMD_U	0978'	CMD_V	097B'	CMD_V1
099E'	CMD_V2	4123	CMD_W	00A0	CMD_WRITE
4126	CMD_X	4129	CMD_Y	0A18'	CMD_Z
0A25'	CMD_Z1	0A31'	CMD_Z2	0A33'	CMD_Z3
0A36'	CMD_Z4	0A48'	CMD_Z5	0A49'	CMD_Z6
0A51'	CMD_Z7	0A56'	CMD_Z8	0A5E'	CMD_Z9
1A21'	CMPER	1A16'	CMPLP	0A09'	CMP_DE_HL

0A0C'	CMP_DE_HL1	0A14'	CMP_DE_HL2	045B'	CMP_HL_DE
4002	CNT_CAR	4012	CNT_TRACE	1A12'	COMPARE
003B'	CON_STT	003D'	CON_STT1	19A2'	CPT_CMD
057D'	CP_EQ	058F'	CP_FALSE	058B'	CP_GE
058D'	CP_GT	0585'	CP_LE	0587'	CP_LT
0581'	CP_NE	0590'	CP_TRUE	000D	CR
030A'	CR_LF	0010	CTRLP	0018	CTRLX
070E'	CVT_H1	06FE'	CVT_HEX	000C	D8253
0451'	DEF_E1	044C'	DEF_END	0315'	DELCAR
007F	DELETE	0346'	DEL_CAR	05E8'	DIVD1
4006	DMA_DSK	05FE'	DROP	0018	DSK_CMD
001B	DSK_DAT	001A	DSK_SCT	0018	DSK_STT
0019	DSK_TRK	0014	DSK_TYP	4010	DUMP?
05F0'	DVLOOP	04CA'	END_71	04C7'	END_7BUF
0452'	END_CMD	17CA'	END_CMDK	4000	END_ERR
0003	EOC	0013'	EQ_NE	02C8'	ERROR
1A03'	FILCMP	1E97'	FLAGS1	1EA6'	FLAGS2
4016	FLAG_J	4015	FLAG_N	0137'	FLUX
0336'	GET1	034B'	GET2	035D'	GET3
036A'	GET4	0361'	GET5	037C'	GET6
036E'	GET7	0527'	GETI1	0592'	GETI2
0596'	GETI3	05AA'	GETI4	16BA'	GET_0_7
16DA'	GET_BYTE	0040'	GET_CAR	172C'	GET_C_IXIY
1745'	GET_C_IXY1	1759'	GET_C_IXY2	176B'	GET_C_IXY3
1771'	GET_C_IXY4	1777'	GET_C_IXY5	1778'	GET_C_IXY6
16C3'	GET_DESL	0389'	GET_END	0526'	GET_INT
04E3'	GET_INT2	043C'	GET_INT2_END	04E6'	GET_INT2_NOER
0441'	GET_INT2_NOER_EN	04D0'	GET_INT3_END	0446'	GET_INT_END
03A5'	GET_LIN1	0393'	GET_LIN2	0331'	GET_LINE
16EB'	GET_PAR1_REG	16F4'	GET_PAR_REG	16FB'	GET_PAR_REG1
1704'	GET_PAR_REG2	1707'	GET_REG	16F1'	GET_REG_HL
1714'	GET_REG_IXIY	1725'	GET_REG_IXY1	1729'	GET_REG_IXY2
0502'	GET_SIZE	0641'	GET_WRD	0643'	GET_WRD1
0672'	GET_WRD2	04E7'	GI2_1	04F4'	GI2_2
04F5'	GI2_3	04FD'	GI2_4	0500'	GI2_5
40BA	GO	16E8'	GO1_ERROR	143D'	GO2_ERROR
1629'	GO3_ERROR	1799'	GO_ERROR	0BFC'	GO_ON
0BE0'	GO_PRG	0E88'	GRP_FD_DD	0EA7'	GRUPO_CB
0EB3'	GRUPO_CB_1	0E47'	GRUPO_DD	0E4D'	GRUPO_DDFD
0EBA'	GRUPO_ED	0E4B'	GRUPO_FD	0E5D'	GRUPO_FD_DD
050B'	GS_1	051D'	GS_2	051F'	GS_3
05BA'	HL_ADD_DE	060F'	HL_AND_DE	05DD'	HL_DIV_DE
060A'	HL_MOD_DE	05C0'	HL_MUL_DE	0616'	HL_OR_DE
05BC'	HL_SUB_DE	400A	HST_OP	2000	INIC_EPROM
4000	INIC_RAM	0256'	INIT	01EF'	INIT_RAM
1D86'	INST10	1D89'	INST11	1D8C'	INSTR
1D44'	INSTR1	1D51'	INSTR2	1D57'	INSTR3
1D72'	INSTR4	1D77'	INSTR5	1D7A'	INSTR6
1D7D'	INSTR7	1D80'	INSTR8	1D83'	INSTR9
0005	INVBNC	1EBE'	IO_P_C	06E3'	JP_ERROR
0086'	JSYS1	02D0'	JUMP	1802'	K8POL1
1879'	KAD1	188E'	KAD2	1869'	KAD3
1840'	KDSK1	1847'	KDSK2	1997'	KHOME
18B0'	KRW	18D7'	KRW1	18F7'	KRW2
19F3'	KRW3	18E3'	KRW4	18E6'	KRW5
06D2'	LD_HL	000A	LF	0164'	LIST
016E'	LIST1	0197'	LIST2	01AB'	LIST3
0E06'	LISTIY	0E1A'	LISTIY1	0DE4'	LIST_IY
0DFE'	LIST_IY1	01DF'	LIST_STT	04BF'	LOC_TAB
04C0'	LOC_TAB1	10C3'	LOC_TAB_A	0015	LST
0016	LST_STT	0030	MAX_JSYS	05C9'	MLT1
05D7'	MLT2	412C	MORE_JSYS	09AF'	MOVEDATA

001B'	MSG_BP	0307'	MSG_ERROR	1F0A'	NAME_REG
0CE7'	NOT_BREAK	0023'	NOT_CMD	0004	NRMBNC
000A	NUM_CMDK	0021	NUM_CMD_NOT	10DD'	NXT_INSTR
0688'	N_ASC	068E'	N_ASC1	0695'	N_ASC2
069A'	N_ASC3	06C3'	N_CNTD	06BE'	N_CP1
06B1'	N_NEG	06B5'	N_NEG1	06D7'	N_PUSH
06A2'	N_REG	1EC2'	N_REG_P	0631'	OPR_ROT
07A8'	OUTFLAGS	07C8'	OUTFLAGS1	040B'	OUTPUT
03EB'	OUT_A	03CB'	OUT_ASP	079F'	OUT_A_FLAGS
0048'	OUT_CAR	03CF'	OUT_DEC	03D6'	OUT_DECIMAL
078C'	OUT_FLAGS	03E6'	OUT_HL	03B2'	OUT_HL@
03C8'	OUT_HL@1	03F4'	OUT_NIB	03FC'	OUT_NIB1
07D0'	OUT_REG	07F3'	OUT_REG1	07F6'	OUT_REG2
07FA'	OUT_REG3	4008	PAR_DSK	401B	POSICAO
8000	POS_INIC	0856'	POUT_CDE	116C'	PP_RET
0737'	PRC_REG	194C'	PREP	1946'	PREP1
193F'	PREP_RDY	00F7'	PRG_PT	041C'	PRINT
041E'	PRINT1	10D2'	PRINT_MENEMO	10D3'	PRINT_MENEMO1
01D0'	PRINT_OUT	01D1'	PRINT_OUT1	0415'	PRINT_PC
0ED8'	PROC_BYTE	0EBE'	PROC_MEN	0ECB'	PROC_MEN1
0ED7'	PROC_MEN2	000B'	PRONT	400F	PRT?
10CA'	PRT_ACCUM	10CD'	PRT_VRG	041A'	PR_PC
014E'	PUT_C1	0161'	PUT_C2	0132'	PUT_CAR
0006	RAM	19D5'	RAMTST	19B0'	RDY_D1
19AD'	RDY_DSK	1920'	READ_D2	1919'	READ_DSK
1E60'	REGS1	1E6C'	REGS2	1E75'	REGS3
1E79'	REGS4	1E7E'	REGS5	1E87'	REGS6
1E90'	REGS7	1E92'	REGS8	40B1	REG_A
40A9	REG_A_	40B2	REG_BC	40A6	REG_BC_
40B4	REG_DE	40A4	REG_DE_	40B0	REG_F
40A8	REG_F_	40B6	REG_HL	40A2	REG_HL_
40AB	REG_I	40AE	REG_IX	40AC	REG_IY
40BC	REG_PC	40B8	REG_SP	00FF	RESTART
0CC8'	RESTORE_BREAK	0CCE'	RES_LOOP	05B7'	RET_OPR
0004	ROM	0251'	ROT_EPROM	0071'	ROT_JSYS
40BE	ROT_RAM	1929'	RWEND	1938'	RWEND1
000F	S8253	4005	SCT_DSK	199B'	SEEK_TRK
0056'	SER_INP	0067'	SER_OUT	0052'	SER_STT
0D36'	SET_BRK	0D3C'	SET_BRK1	0D52'	SET_BRK2
08BC'	SFLAGS	1817'	SIMD1	1808'	SIMDUP
0010	SIO_A_DAT	0011	SIO_A_STT	0012	SIO_B_DAT
0013	SIO_B_STT	0005	SIZE_ROT	1F41'	SIZ_REG
1EE2'	SIZ_R_PR	0434'	SKIP_COMMA	042D'	SKIP_WHITE
0BAE'	SLEEP_BREAK	0BB4'	SLEEP_BRK1	4200	STACK
0299'	START	02CB'	START3	0296'	START_CR
1985'	STATUS	196C'	STATUS_ERR	0C66'	STOP
0327'	STORE	4009	STP_DSK	042B'	SUM_HL_A
002B'	SYM_@	1CAB'	TABEL10	1CC0'	TABEL11
1CF3'	TABEL12	1D02'	TABEL13	1D3A'	TABEL14
1A26'	TABELA1	1AF2'	TABELA2	1B0D'	TABELA3
1B1C'	TABELA4	1B37'	TABELA5	1BBA'	TABELA6
1C06'	TABELA7	1C34'	TABELA8	1C5F'	TABELA9
17BF'	TAB_CMDK	0571'	TAB_END_CP	0095'	TAB_JSYS
005E'	TAB_OPR	0102'	TAB_PER	0050	TAM_BUF
0016	TIME_OUT	400D	TMP_IY	400B	TMP_SP
4011	TRACE?	4004	TRK_DSK	4003	TYP_DSK
03A9'	UPPER_CASE	4300	USR_STACK	4014	VAR_1
401E	VAR_@	401D	VAR_L1	4017	VAR_T1
4018	VAR_T2	401A	VAR_T3	0002	VEL4MHZ
0000	VELOC	1966'	WAIT	1968'	WAIT1
0147'	WAIT_XON	0409'	WHITE	0406'	WHITE2
0403'	WHITE4	0400'	WHITE6	19EF'	WLKLP

19F1'	WLKLP1	1907'	WRITE_DSK	190E'	WRT_D2
0711'	W_DEC	0714'	W_DEC1	0727'	W_DEC2
0676'	W_HD1	0686'	W_HD2	06E9'	W_HE1
06F8'	W_HE2	06E6'	W_HEXA	0675'	W_HEXDEC
0013	XOFF	0011	XON		

No Fatal error(s)

APÊNDICE C

Listagem do CBIOS

```
                        TITLE   Mixed Density BIOS WINCHESTER
                ;
                ;       ###########################################################
                ;       #                                                         #
                ;       #           BIOS LIE Z80, versao 1.1 para Disco           #
                ;       #                                                         #
                ;       #  Bios desenvolvido no laboratorio de LIE, IFQSC-USP.    #
                ;       #  Versao utilizando rotinas do monitor EDOS.             #
                ;       #                                                         #
                ;       #  Escrito por: Mateus J. Martins.                        #
                ;       #  Data:               04/07/1988                         #
                ;       #                                                         #
                ;       ###########################################################


0040            memsz   equ     64              ; Tamanho da memoria em Kbytes
B000            bias    equ     (memsz-20)*1024 ; Deslocamento do CP/M
2000            cpmrom  equ     2000h           ; Endereco do CP/M em EPROM
1600            cpmsiz  equ     1600h           ; Tamanho do CP/M
E400            ccp     equ     3400h+bias      ; Ender. inicial do CCP
E488            ccpcnt  equ     ccp+88h         ; Contador e Buffer de
E407            ccpbuf  equ     ccp+7h          ;   caracteres do CCP
EC06            bdos    equ     ccp+806h        ; Ender. inicial do BDOS
FA00            bios    equ     ccp+cpmsiz      ; Ender. inicial do BIOS
0000            jboot   equ     0               ; Variaveis utilizadas pelo
0003            iobyte  equ     3               ;   CP/M, na area inicial
0004            cdrv    equ     4
0005            jbdos   equ     5
0030            edos    equ     30h             ; Ponto de entrada do EDOS
0038            jinva   equ     38h             ; RST 38 utilizado como instrucao
                                                ;  nao valida
0080            buffer  equ     80h             ; Ender. da DMA default
0094            byteIO  equ     94h             ; Byte de controle

                ;##########################################
                ;       Ender. de rotinas no EDOS         #
                ;##########################################

4052            track   equ     4052h           ; Variaveis do EDOS
4054            sector  equ     4054h           ;  utilizadas nos algoritmos
4055            dmaadr  equ     4055h           ;  de montagem e desmontagem
4057            hstact  equ     4057h           ;  de setores.
4058            hstwrt  equ     4058h
4059            unacnt  equ     4059h

0005            dsk     equ     5               ; Numero de discos maximo

000D            cr      equ     13              ; Retorno de carro
000A            lf      equ     10              ; Mudanca de linha
001B            esc     equ     27              ; Utilizado para apagar a tela
0007            bell    equ     07              ; Campainha
0000            null    equ     00              ; Fim de cadeias de caracteres

0010            data    equ     10h             ; Porto serial A, SIO-A
0011            stat    equ     11h
0012            datal   equ     12h             ; Porto serial B, SIO-B
0013            statl   equ     13h

00C0            dskrd   equ     0c0h            ; Portos de controle
00C0            dskwrt  equ     0c0h            ; do disco virtual
```

```
00C1                dsklow      equ     0c1h
00C2                dskmid      equ     0c2h
00C3                dskhi       equ     0c3h
00C3                dskclr      equ     0c3h

0016                lstst       equ     16h         ; Impressora
0015                lst         equ     15h

0004                eprom       equ     04h         ; Porto para chaveamento
0006                ram         equ     06h         ;  dos bancos

                    ;       Comandos executador pelo EDOS

0000                cmdoff      equ     0           ; Imprimir a mensagem 'off line'
0001                cmdout      equ     1           ; Imprimir 'out of paper'
0002                cmdchg      equ     2           ; Trocar o disco mestre
0003                cmdsel      equ     3           ; Selecionar disco
0004                cmdrd       equ     4           ; Ler um setor
0005                cmdwrt      equ     5           ; Escrever um setor
0006                cmdinv      equ     6           ; Imprimir 'Jump invalido'
0007                cmdwbr      equ     7           ; Imprimir 'Boot error'
0008                cmdrdp      equ     8           ; Retornar os parametros do disco
0009                cmdwdp      equ     9           ; Salvar os parametros do disco
000A                cmdini      equ     10          ; Reinicializar o sistema


                    ;#########################################################

                    ;           Start of BIOS code

                                .PHASE  bios

FA00  C3 FBF3       jcboot:     jp      boot        ; Partida a frio
FA03  C3 FBB8       jwboot:     jp      wboot       ; Partida a quente
FA06  C3 FAF6                   jp      const       ; Le o estado da console
FA09  C3 FAFB                   jp      conin       ; Le um caracter da console
FA0C  C3 FB12                   jp      conout      ; Escreve um caracter na console
FA0F  C3 FB32                   jp      list        ; Imprime um caracter na impressora
FA12  C3 FB71                   jp      punch       ; Escreve um caracter no porto AUX
FA15  C3 FB97                   jp      reader      ; Le um caracter do porto AUX
FA18  C3 FC6A                   jp      home        ; Posiciona na trilha zero
FA1B  C3 FC7E                   jp      seldsk      ; Seleciona o acionador
FA1E  C3 FC44                   jp      settrk      ; Seleciona a trilha
FA21  C3 FC4D                   jp      setsec      ; Seleciona o setor
FA24  C3 FC56                   jp      setdma      ; Endereco para operacoes em disco
FA27  C3 FCAD                   jp      read        ; Le um setor
FA2A  C3 FCB1                   jp      write       ; Escreve um setor
FA2D  C3 FB2A                   jp      listst      ; Le o estado da impressora
FA30  C3 FC5F                   jp      sectra      ; Translada setores
FA33  C3 FC39                   jp      swap        ; Muda para o EDOS

FA36                dpbase      equ     $           ; Endereco inicial das tabelas

FA36  0000 0000     dph0:       defw    0,0         ; Ender. da tabela de translacao
FA3A  0000 0000                 defw    0,0         ; area auxiliar
FA3E  FCF5 FA96                 defw    dirbuf, dpb0 ; Buffer diretorio, Ender. DPB
FA42  FDB3 FD75                 defw    csv0,   alv0 ; check, alloc vectors

FA46  0000 0000     dph1:       defw    0,0         ; Ender. da tabela de translacao
FA4A  0000 0000                 defw    0,0         ; area auxiliar
FA4E  FCF5 FAA6                 defw    dirbuf, dpb1 ; Buffer diretorio, Ender. DPB
FA52  FE11 FDD3                 defw    csv1,   alv1 ; check, alloc vectors
```

```
FA56  0000 0000  dph2:   defw  0,0          ; Ender. da tabela de translacao
FA5A  0000 0000          defw  0,0          ; area auxiliar
FA5E  FCF5 FAB6          defw  dirbuf, dpb2 ; Buffer diretorio, Ender. DPB
FA62  FE6F FE31          defw  csv2,   alv2 ; check, alloc vectors

FA66  0000 0000  dph3:   defw  0,0          ; Ender. da tabela de translacao
FA6A  0000 0000          defw  0,0          ; area auxiliar
FA6E  FCF5 FAC6          defw  dirbuf, dpb3 ; Buffer diretorio, Ender. DPB
FA72  FECD FE8F          defw  csv3,   alv3 ; check, alloc vectors

FA76  0000 0000  dphWin: defw  0,0          ; DPH do disco rigido
FA7A  0000 0000          defw  0,0
FA7E  FCF5 FAD6          defw  dirbuf, dpbWin
FA82  0000 FD75          defw  0,      alvWin

FA86  0000 0000  dphVir: defw  0,0          ; DPH do disco virtual
FA8A  0000 0000          defw  0,0
FA8E  FCF5 FAE6          defw  dirbuf, dpbVir
FA92  0000 FEED          defw  0,      alvVir

FA96  0040       dpb0:   defw  64           ; Setores por trilha
FA98  04                 defb  4            ; Deslocamento do bloco
FA99  0F                 defb  15           ; Mascara do bloco
FA9A  01                 defb  1            ; Marcara da extencao
FA9B  009B               defw  155          ; Tamanho do disco - 1
FA9D  003F               defw  63           ; Numero de entradas no Dir.
FA9F  80                 defb  128          ; alloc0
FAA0  00                 defb  0            ; alloc1
FAA1  0010               defw  16           ; Tamanho do Check
FAA3  0001               defw  1            ; Deslocamento
FAA5  06                 defb  6            ; Tipo do disco, nao faz parte do
                                            ;  DPB, adicionado.

FAA6  0040       dpb1:   defw  64           ; Setores por trilha
FAA8  04                 defb  4            ; Deslocamento do bloco
FAA9  0F                 defb  15           ; Mascara do bloco
FAAA  01                 defb  1            ; Marcara da extencao
FAAB  009B               defw  155          ; Tamanho do disco - 1
FAAD  003F               defw  63           ; Numero de entradas no Dir.
FAAF  80                 defb  128          ; alloc0
FAB0  00                 defb  0            ; alloc1
FAB1  0010               defw  16           ; Tamanho do Check
FAB3  0001               defw  1            ; Deslocamento
FAB5  06                 defb  6            ; Tipo do disco, nao faz parte do
                                            ;  DPB, adicionado.

FAB6  0040       dpb2:   defw  64           ; Setores por trilha
FAB8  04                 defb  4            ; Deslocamento do bloco
FAB9  0F                 defb  15           ; Mascara do bloco
FABA  01                 defb  1            ; Marcara da extencao
FABB  009B               defw  155          ; Tamanho do disco - 1
FABD  003F               defw  63           ; Numero de entradas no Dir.
FABF  80                 defb  128          ; alloc0
FAC0  00                 defb  0            ; alloc1
FAC1  0010               defw  16           ; Tamanho do Check
FAC3  0001               defw  1            ; Deslocamento
FAC5  06                 defb  6            ; Tipo do disco, nao faz parte do
                                            ;  DPB, adicionado.

FAC6  0040       dpb3:   defw  64           ; Setores por trilha
FAC8  04                 defb  4            ; Deslocamento do bloco
```

```
FAC9  0F                        defb  15          ; Mascara do bloco
FACA  01                        defb  1           ; Marcara da extencao
FACB  009B                      defw  155         ; Tamanho do disco - 1
FACD  003F                      defw  63          ; Numero de entradas no Dir.
FACF  80                        defb  128         ; alloc0
FAD0  00                        defb  0           ; alloc1
FAD1  0010                      defw  16          ; Tamanho do Check
FAD3  0001                      defw  1           ; Deslocamento
FAD5  06                        defb  6           ; Tipo do disco, nao faz parte do
                                                  ;  DPB, adicionado.

FAD6  0044          dpbWin:     defw  17*4        ; 16 setores de 512 bytes
FAD8  04                        defb  4           ; blocos de 2 kbytes
FAD9  0F                        defb  15          ; DPB para o Winchester
FADA  00                        defb  0
FADB  144C                      defw  5197-1
FADD  03FF                      defw  1024-1
FADF  FF                        defb  0ffh
FAE0  FF                        defb  0ffh
FAE1  0000                      defw  0
FAE3  0001                      defw  1
FAE5  81                        defb  81h         ; tipo

FAE6  0100          dpbVir:     defw  256         ; virtual disk
FAE8  04                        defb  4           ; ( 512 kbytes, max= 1Mbyte )
FAE9  0F                        defb  15          ; DPB para o disco Virtual
FAEA  01                        defb  1
FAEB  00FF                      defw  255
FAED  007F                      defw  127
FAEF  C0                        defb  0c0h
FAF0  00                        defb  0
FAF1  0000                      defw  0
FAF3  0000                      defw  0
FAF5  80                        defb  80h         ; virtual

                    ;       Rotinas de entrada e saida

FAF6  DB 11         const:      in    a,(stat)    ; Le o estado da console
FAF8  0F                        rrca
FAF9  9F                        sbc   a,a
FAFA  C9                        ret
                    ;
FAFB  3A 0003       conin:      ld    a,(iobyte)  ; Le um caracter da console
FAFE  E6 03                     and   3
FB00  3D                        dec   a
FB01  CA FB9F                   jp    z,Xreader
FB04  3D                        dec   a
FB05  CA FB9F                   jp    z,Xreader
FB08  CD FAF6       Xconin:     call  const
FB0B  28 FB                     jr    z,Xconin
FB0D  DB 10                     in    a,(data)
FB0F  E6 7F                     and   7fh
FB11  C9                        ret
                    ;
FB12  3A 0003       conout:     ld    a,(iobyte)  ; Escreve um caracter na console
FB15  E6 03                     and   3
FB17  3D                        dec   a
FB18  CA FB79                   jp    z,Xpunch
FB1B  3D                        dec   a
FB1C  28 20                     jr    z,Xlist
FB1E  DB 11         Xconout:    in    a,(stat)
```

```
FB20  E6 04                   and   4
FB22  28 FA                   jr    z,Xconout
FB24  79                      ld    a,c
FB25  E6 7F                   and   7fh
FB27  D3 10                   out   (data),a
FB29  C9                      ret
                    ;
FB2A  DB 16         liststs:  in    a,(lstst)     ; Le o estado da impressora
FB2C  E6 08                   and   8
FB2E  C8                      ret   z
FB2F  3E FF                   ld    a,-1
FB31  C9                      ret
                    ;
FB32  3A 0003       list:     ld    a,(iobyte)    ; Imprime um caracter na
FB35  E6 C0                   and   0c0h          ;  impressora
FB37  28 E5                   jr    z,Xconout
FB39  FE 40                   cp    40h
FB3B  CA FB79                 jp    z,Xpunch
FB3E  CD FB2A       Xlist:    call  liststs
FB41  28 04                   jr    z,list1
FB43  79                      ld    a,c
FB44  D3 15                   out   (lst),a
FB46  C9                      ret
FB47  DB 16         list1:    in    a,(lstst)
FB49  CB 67                   bit   4,a
FB4B  28 1C                   jr    z,list3
                    ;
FB4D  3E 01                   ld    a,cmdout      ; Chama o EDOS para imprimir
FB4F  C5            predos:   push  bc            ; a mensagem de erro
FB50  CD FC39                 call  swap
FB53  C1                      pop   bc
FB54  DB 16         list2:    in    a,(lstst)
FB56  E6 08                   and   8
FB58  20 E4                   jr    nz,Xlist
FB5A  CD FAF6                 call  const
FB5D  28 F5                   jr    z,list2
FB5F  CD FAFB                 call  conin
FB62  FE 03                   cp    3
FB64  CA FBB8                 jp    z,wboot
FB67  18 EB                   jr    list2
FB69  CB 77         list3:    bit   6,a
FB6B  20 D1                   jr    nz,Xlist
                    ;
FB6D  3E 00                   ld    a,cmdoff
FB6F  18 DE                   jr    predos

FB71  3A 0003       punch:    ld    a,(iobyte)    ; Imprime um caracter no
FB74  E6 30                   and   30h           ; dispositivo auxiliar
FB76  CA FB1E                 jp    z,Xconout
FB79  CD FB86       Xpunch:   call  Ctrl$s
FB7C  DB 13                   in    a,(stat1)
FB7E  E6 04                   and   4
FB80  28 F7                   jr    z,Xpunch
FB82  79                      ld    a,c
FB83  D3 12                   out   (data1),a
FB85  C9                      ret
                    ;
FB86  DB 13         Ctrl$s:   in    a,(stat1)     ; Verifica CTRL S e CTRL Q
FB88  0F                      rrca                ; no dispositivo auxiliar
FB89  D0                      ret   nc
FB8A  DB 12                   in    a,(data1)
FB8C  FE 13                   cp    13h
```

```
FB8E  C0                        ret     nz
FB8F  CD FB97       Ctrl$q:     call    reader
FB92  FE 11                     cp      11h
FB94  20 F9                     jr      nz,ctrl$q
FB96  C9                        ret
                    ;
FB97  3A 0003       reader:     ld      a,(iobyte)      ; Le um caracter do disp.
FB9A  E6 0C                     and     0ch             ;  auxiliar
FB9C  CA FB08                   jp      z,Xconin
FB9F  DB 13         Xreader:    in      a,(stat1)
FBA1  E6 01                     and     1
FBA3  28 FA                     jr      z,Xreader
FBA5  DB 12                     in      a,(data1)
FBA7  E6 7F                     and     7fh
FBA9  C9                        ret
                    ;
FBAA  3E 07         wberr:      ld      a,cmdwbr        ; Chama o EDOS para imprimir
FBAC  CD FC39                   call    swap            ; uma mensagem de erro no BOOT
FBAF  18 07                     jr      wboot

FBB1  C1            msginv:     pop     bc              ; EDOS imprime mensagem de
FBB2  0B                        dec     bc              ; jump invalido
FBB3  3E 06                     ld      a,cmdinv
FBB5  CD FC39                   call    swap
                    ;
                    ;
                    ;       Rotinas de controle dos discos
                    ;
FBB8  D3 06         wboot:      out     (ram),a         ; Carrega o sistema partida a quente
FBBA  31 0080                   ld      sp,buffer
FBBD  1E 00                     ld      e,0
FBBF  0E 00                     ld      c,0
FBC1  CD FC7E                   call    seldsk
FBC4  7C                        ld      a,h
FBC5  B5                        or      l
FBC6  28 E2                     jr      z,wberr
FBC8  01 0000                   ld      bc,0
FBCB  CD FC44                   call    settrk
FBCE  21 E400                   ld      hl,ccp
FBD1  01 2C04                   ld      bc,2c04h        ; 44 setores comecando do 4
FBD4  E5            wbloop:     push    hl
FBD5  C5                        push    bc
FBD6  CD FC4D                   call    setsec
FBD9  44                        ld      b,h
FBDA  4D                        ld      c,l
FBDB  CD FC56                   call    setdma
FBDE  CD FCAD                   call    read
FBE1  C1                        pop     bc
FBE2  E1                        pop     hl
FBE3  20 C5                     jr      nz,wberr
FBE5  11 0080                   ld      de,80h
FBE8  19                        add     hl,de
FBE9  0C                        inc     c
FBEA  10 E8                     djnz    wbloop
FBEC  3A E400                   ld      a,(ccp)
FBEF  FE C3                     cp      0c3h
FBF1  20 B7                     jr      nz,wberr

FBF3  AF            boot:       xor     a               ; Inicializa variaveis do EDOS
FBF4  D3 04                     out     (eprom),a       ;  Partida a Frio, a maior parte
FBF6  32 4057                   ld      (hstact),a      ;  do servico foi feito pelo EDOS
FBF9  32 4059                   ld      (unacnt),a
```

```
FBFC  32 4058                 ld    (hstwrt),a
                      ;
FBFF  D3 06                   out   (ram),a       ; Inicializa variaveis do CP/M
FC01  3A 0003                 ld    a,(iobyte)
FC04  E6 FC                   and   0fch          ; retorna console = crt:
FC06  32 0003                 ld    (iobyte),a
FC09  3E C3                   ld    a,0c3h
FC0B  32 0000                 ld    (jboot),a
FC0E  32 0005                 ld    (jbdos),a
FC11  32 0038                 ld    (jinva),a
FC14  21 FA03                 ld    hl,jwboot
FC17  22 0001                 ld    (jboot+1),hl
FC1A  21 EC06                 ld    hl,bdos
FC1D  22 0006                 ld    (jbdos+1),hl
FC20  21 FBB1                 ld    hl,msginv
FC23  22 0039                 ld    (jinva+1),hl
FC26  31 0080                 ld    sp,buffer
FC29  CD FCEB                 call  signon
FC2C  01 0080                 ld    bc,buffer
FC2F  CD FC56                 call  setdma
FC32  3A 0004                 ld    a,(cdrv)
FC35  4F                      ld    c,a
FC36  C3 E400                 jp    ccp
                      ;
FC39  21 FC41       swap:     ld    hl,swapr      ; Chama o EDOS
FC3C  D3 04                   out   (eprom),a
FC3E  C3 0030                 jp    edos
FC41  D3 06         swapr:    out   (ram),a
FC43  C9                      ret

FC44  D3 04         settrk:   out   (eprom),a     ; Armazena a trilha no EDOS
FC46  ED 43 4052              ld    (track),bc
FC4A  D3 06                   out   (ram),a
FC4C  C9                      ret
                      ;
FC4D  79            setsec:   ld    a,c           ; Armazena o setor no EDOS
FC4E  D3 04                   out   (eprom),a
FC50  32 4054                 ld    (sector),a
FC53  D3 06                   out   (ram),a
FC55  C9                      ret
                      ;
FC56  D3 04         setdma:   out   (eprom),a     ; Armazena o Ender. para DMA
FC58  ED 43 4055              ld    (dmaadr),bc   ;  no EDOS
FC5C  D3 06                   out   (ram),a
FC5E  C9                      ret
                      ;
FC5F  60            sectra:   ld    h,b           ; Translada setores
FC60  69                      ld    l,c
FC61  7A                      ld    a,d
FC62  B3                      or    e
FC63  C8                      ret   z
FC64  EB                      ex    de,hl
FC65  09                      add   hl,bc
FC66  6E                      ld    l,(hl)
FC67  26 00                   ld    h,0
FC69  C9                      ret
                      ;
FC6A  D3 04         home:     out   (eprom),a     ; Informa o EDOS, trilha = 0
FC6C  21 0000                 ld    hl,0
FC6F  22 4052                 ld    (track),hl
FC72  3A 4058                 ld    a,(hstwrt)
FC75  B7                      or    a
```

```
FC76  20 03              jr    nz,homel
FC78  32 4057            ld    (hstact),a
FC7B  D3 06     homel:   out   (ram),a
FC7D  C9                 ret
                ;
FC7E  79        seldsk:  ld    a,c          ; Chama o EDOS para selecionar o disco
FC7F  FE 06              cp    dsk+1
FC81  30 1E              jr    nc,seld2
FC83  3E 03     seld1:   ld    a,cmdsel
FC85  CD FC39            call  swap
FC88  FE FF              cp    0ffh         ; Erro de selecao ?
FC8A  28 15              jr    z,seld2
FC8C  32 FCAC            ld    (diskvt),a
FC8F  21 FA86            ld    hl,dphvir
FC92  FE 05              cp    dsk
FC94  C8                 ret   z
FC95  11 FA36            ld    de,dpbase    ; nao
FC98  6F                 ld    l,a
FC99  26 00              ld    h,0
FC9B  29                 add   hl,hl
FC9C  29                 add   hl,hl
FC9D  29                 add   hl,hl
FC9E  29                 add   hl,hl
FC9F  19                 add   hl,de
FCA0  C9                 ret
                ;
FCA1  21 0000   seld2:   ld    hl,0         ; erro
FCA4  AF                 xor   a
FCA5  32 0004            ld    (cdrv),a     ; Disco corrente A
FCA8  32 FCAC            ld    (diskvt),a
FCAB  C9                 ret
                ;
FCAC  00        diskvt:  defb  0
                ;
FCAD  3E 04     read:    ld    a,cmdrd      ; Pede ao EDOS para ler um setor
FCAF  18 02              jr    rdwrt

FCB1  3E 05     write:   ld    a,cmdwrt     ; Pede ao EDOS para escrever
                                           ;  um setor
FCB3  F5        rdwrt:   push  af
FCB4  3A FCAC            ld    a,(diskvt)   ; Disco virtual ?
FCB7  FE 05              cp    dsk
FCB9  28 06              jr    z,dskvir
FCBB  F1                 pop   af
FCBC  CD FC39            call  swap
FCBF  B7                 or    a
FCC0  C9                 ret
                ;
FCC1  D3 04     dskvir:  out   (eprom),a    ; Le ou escreve
FCC3  3A 4054            ld    a,(sector)   ;  um setor do disco virtual
FCC6  47                 ld    b,a
FCC7  E6 01              and   1
FCC9  0F                 rrca
FCCA  D3 C1              out   (dsklow),a   ; Monta o endereco para
FCCC  3A 4052            ld    a,(track)    ; o disco virtual
FCCF  0F                 rrca
FCD0  D3 C3              out   (dskhi),a
FCD2  78                 ld    a,b
FCD3  1F                 rra
FCD4  D3 C2              out   (dskmid),a
FCD6  2A 4055            ld    hl,(dmaadr)
FCD9  D3 06              out   (ram),a
```

```
FCDB  01 80C0                    ld    bc,80c0h      ; 128 bytes, port C0h
FCDE  F1                         pop   af
FCDF  FE 05                      cp    cmdwrt
FCE1  3E 00                      ld    a,0           ; leitura ou escrita ?
FCE3  20 03                      jr    nz,dskvrd
FCE5  ED B3                      otir                ; Escreve
FCE7  C9                         ret
FCE8  ED B2            dskvrd:   inir                ; Le
FCEA  C9                         ret
                       ;
                       ; Rotina apenas executada durante a inicializacao do BIOS
                       ;
FCEB  3E C9            signon:   ld    a,0c9h        ; JUMP
FCED  32 FCEB                    ld    (signon),a
FCF0  3E 94                      ld    a,ByteIO
FCF2  32 0003                    ld    (IOByte),a
FCF5  CD FD8D                    call  print
FCF8  0D 0A 0A                   defb  cr,lf,lf,'
FCFB  20 20 20
FCFE  20 20 20
FD01  20 20 20
FD04  20 20 20
FD07  20 20 20
FD0A  20 20 20
FD0D  4D 69 63                   defm  'Micro LIE v1.1, 64k CP/M vers 2.2'
FD10  72 6F 20
FD13  4C 49 45
FD16  20 76 31
FD19  2E 31 2C
FD1C  20 36 34
FD1F  6B 20 43
FD22  50 2F 4D
FD25  20 76 65
FD28  72 73 20
FD2B  32 2E 32
FD2E  0D 0A 0A                   defb  cr,lf,lf,null
FD31  00
FD32  AF                         xor   a
FD33  32 0004                    ld    (cdrv),a      ; Drive corrente
                       ;
FD36  0E 00                      ld    c,0
FD38  1E 00                      ld    e,0
FD3A  CD FC7E                    call  seldsk
FD3D  7D                         ld    a,l
FD3E  B4                         or    h
FD3F  C8                         ret   z
FD40  01 0000                    ld    bc,0
FD43  CD FC44                    call  settrk
FD46  01 FD9B                    ld    bc,buftmp     ; DMA corrente
FD49  CD FC56                    call  setdma

FD4C  CD FD53                    call  autorun       ; Prepara o CCP para executar
                                                     ;  um comando
FD4F  CD FD75                    call  setdpb        ; Acerta o DBP
FD52  C9                         ret

FD53  0E 00            autorun:  ld    c,0           ; Le o comando do disco corrente
FD55  CD FC4D                    call  setsec
FD58  CD FCAD                    call  read
FD5B  C0                         ret   nz
FD5C  21 FD9B                    ld    hl,buftmp
FD5F  7E                         ld    a,(hl)
```

```
FD60  A7                    and   a
FD61  C8                    ret   z
FD62  F8                    ret   m
FD63  11 E407               ld    de,ccpbuf       ; Passa-o para o buffer do CCP
FD66  23                    inc   hl
FD67  77                    ld    (hl),a
FD68  3C                    inc   a
FD69  4F                    ld    c,a
FD6A  06 00                 ld    b,0
FD6C  ED B0                 ldir
FD6E  21 E408               ld    hl,ccpbuf+1
FD71  22 E488               ld    (ccpcnt),hl
FD74  C9                    ret

FD75  0E 02       setdpb:   ld    c,2             ; Pega o DPB default para os discos
FD77  CD FC4D               call  setsec
FD7A  CD FCAD               call  read            ; Le do disco corrente
FD7D  C0                    ret   nz
FD7E  21 FD9B               ld    hl,buftmp
FD81  7E                    ld    a,(hl)
FD82  B7                    or    a
FD83  F8                    ret   m
FD84  11 FA96               ld    de,Dpb0
FD87  01 0050               ld    bc,5*16
FD8A  ED B0                 ldir
FD8C  C9                    ret

FD8D  E3          print:    ex    (sp),hl         ; Imprime mensagens terminadas
FD8E  7E          pmsg:     ld    a,(hl)          ; por nulo
FD8F  23                    inc   hl
FD90  B7                    or    a
FD91  28 06                 jr    z,pmsg1
FD93  4F                    ld    c,a
FD94  CD FB12               call  conout
FD97  18 F5                 jr    pmsg
FD99  E3          pmsg1:    ex    (sp),hl
FD9A  C9                    ret

FD9B              buftmp    equ   $               ; Ender. do buffer temporario

                  $eject
```

```
                        ;
                        ;
                        ;################################################
                        ;
                        ;               Area de variaveis
                        ;
                        ;################################################

                                .dephase
                                .phase  0ffffh-(650+128)

FCF5    dirbuf:         defs    128             ; Buffer do diretorio

FD75    alvWin:         ; Area de alocacao de setores para o winchester

FD75    alv0:           defs    62      ; alocacao de setores disco A
FDB3    csv0:           defs    32      ; e check sum
FDD3    alv1:           defs    62      ; disco B
FE11    csv1:           defs    32
FE31    alv2:           defs    62      ; disco C
FE6F    csv2:           defs    32
FE8F    alv3:           defs    62      ; disco D
FECD    csv3:           defs    32

FEED    alvVir:         defs    32      ; disco virtual

FF0D    endData         equ     $

                        ;               Fim do BIOS
                                .dephase
                                END
```

Macros:

Symbols:

FD75	ALV0	FDD3	ALV1	FE31	ALV2
FE8F	ALV3	FEED	ALVVIR	FD75	ALVWIN
FD53	AUTORUN	EC06	BDOS	0007	BELL
B000	BIAS	FA00	BIOS	FBF3	BOOT
0080	BUFFER	FD9B	BUFTMP	0094	BYTEIO
E400	CCP	E407	CCPBUF	E488	CCPCNT
0004	CDRV	0002	CMDCHG	000A	CMDINI
0006	CMDINV	0000	CMDOFF	0001	CMDOUT
0004	CMDRD	0008	CMDRDP	0003	CMDSEL
0007	CMDWBR	0009	CMDWDP	0005	CMDWRT
FAFB	CONIN	FB12	CONOUT	FAF6	CONST
2000	CPMROM	1600	CPMSIZ	000D	CR
FDB3	CSV0	FE11	CSV1	FE6F	CSV2
FECD	CSV3	FB8F	CTRL$Q	FB86	CTRL$S
0010	DATA	0012	DATA1	FCF5	DIRBUF
FCAC	DISKVT	4055	DMAADR	FA96	DPB0
FAA6	DPB1	FAB6	DPB2	FAC6	DPB3
FA36	DPBASE	FAE6	DPBVIR	FAD6	DPBWIN
FA36	DPH0	FA46	DPH1	FA56	DPH2
FA66	DPH3	FA86	DPHVIR	FA76	DPHWIN
0005	DSK	00C3	DSKCLR	00C3	DSKHI
00C1	DSKLOW	00C2	DSKMID	00C0	DSKRD
FCC1	DSKVIR	FCE8	DSKVRD	00C0	DSKWRT
0030	EDOS	FF0D	ENDDATA	0004	EPROM
001B	ESC	FC6A	HOME	FC7B	HOME1
4057	HSTACT	4058	HSTWRT	0003	IOBYTE
0005	JBDOS	0000	JBOOT	FA00	JCBOOT
0038	JINVA	FA03	JWBOOT	000A	LF
FB32	LIST	FB47	LIST1	FB54	LIST2
FB69	LIST3	FB2A	LISTST	0015	LST
0016	LSTST	0040	MEMSZ	FBB1	MSGINV
0000	NULL	FD8E	PMSG	FD99	PMSG1
FB4F	PREDOS	FD8D	PRINT	FB71	PUNCH
0006	RAM	FCB3	RDWRT	FCAD	READ
FB97	READER	4054	SECTOR	FC5F	SECTRA
FC83	SELD1	FCA1	SELD2	FC7E	SELDSK
FC56	SETDMA	FD75	SETDPB	FC4D	SETSEC
FC44	SETTRK	FCEB	SIGNON	0011	STAT
0013	STAT1	FC39	SWAP	FC41	SWAPR
4052	TRACK	4059	UNACNT	FBAA	WBERR
FBD4	WBLOP	FBB8	WBOOT	FCB1	WRITE
FB08	XCONIN	FB1E	XCONOUT	FB3E	XLIST
FB79	XPUNCH	FB9F	XREADER		

No Fatal error(s)

APÊNDICE D

Listagem do EDOS

```
                        TITLE   EDOS "Eprom Disk Operating System"

                ;       ************************************************************
                ;
                ;               Sistema Destinado ao Novo Micro LIE - EPROM 1
                ;
                ;       Escrito por:    MATEUS J. MARTINS.
                ;       Data:           07/04/1988
                ;
                ;       Alterado: 22/03/1989 Colocacao dos drives sem READY.
                ;
                ;       ************************************************************

                        .Z80

0001            STPRT   EQU     1

                ;       Comandos para o controlador de discos WD2793

0018            CMSEEK  EQU     018H
00C0            CMADDR  EQU     0C0H
0008            CMHOME  EQU     008H
0080            CMREAD  EQU     080H
00A0            CMWRITE EQU     0A0H
00D0            CMABORT EQU     0D0H
004C            CMSTPI  EQU     04CH
006C            CMSTPO  EQU     06CH

                ;       Ender. do CP/M

E400            DEFCPM  EQU     0E400H
FA00            CPMBT   EQU     0FA00H

                ;       Caracteres ASCIIs

000D            CR      EQU     0DH
000A            LF      EQU     0AH
0007            BELL    EQU     07H
0000            NULL    EQU     00H

                ;       Porto de controle de velocidade

0000            VELOC   EQU     00H
0002            VEL4M   EQU     02H

                ;       Porto de controle dos bancos

0004            ROM     EQU     04H
0006            RAM     EQU     06H
0005            INVBNC  EQU     05H
0004            NRMBNC  EQU     04H

                ;       Z80-SIO - Seriais

0010            SIOAD   EQU     10H
0011            SIOAS   EQU     11H
0012            SIOBD   EQU     12H
0013            SIOBS   EQU     13H

                ;       8255    - Porto paralelo
```

```
0017                    C8255           EQU     17H

                        ;       8253    - Temporizador

000C                    D8253           EQU     0CH
000F                    S8253           EQU     0FH

                        ;       Ender. do controlador de discos

0014                    DSKTYP          EQU     14H
0018                    DSKSTT          EQU     18H
0018                    DSKCMD          EQU     18H
0019                    DSKTRK          EQU     19H
001A                    DSKSEC          EQU     1AH
001B                    DSKDAT          EQU     1BH

                        ;       Porto de controle do disco virtual

00C0                    DSKRD           EQU     0C0h            ; Reg. Leitura disco VIRTUAL
00C0                    DSKWRT          EQU     0C0h            ; Reg. Escrita disco VIRTUAL
00C3                    DSKCLR          EQU     0C3h            ; Cmd de Clear nos reg. de end.

                        ;       Porto de controle da impressora

0015                    LST             EQU     15H
0016                    LSTSTT          EQU     16H

                        ;       Portos auxiliares

0016                    TIMEOUT         EQU     16H
0016                    BOOT            EQU     16H

                        ;       Comandos de escrita fornecidos pelo BDOS ao BIOS
                        ;        e passados ao EDOS pelo mesmo

0000                    WRALL           EQU     0       ; Escrita comum
0001                    WRDIR           EQU     1       ; Escrita em diretorio
0002                    WRUAL           EQU     2       ; Escrita alocando setores
0004                    WRFLU           EQU     4       ; Descarga dos buffers

                        ;       Deslocamentos em relacao as tabelas

0000                    TYP             EQU     0       ; tipo
0001                    TRK             EQU     1       ; trilha
0002                    TRKTOP          EQU     2       ; trilha maxima
0003                    SECTOP          EQU     3       ; setor maximo
0004                    PARDSK          EQU     4       ; parametro do disco
0005                    BCKCNT          EQU     5       ; contador de blocos
0006                    SECMSK          EQU     6       ; marcara
0006                    SECSHF          EQU     6       ; deslocamento da mascara
0007                    CPMSPT          EQU     7       ; parametro para o CP/M
0008                    STEP            EQU     8       ; passo
0009                    TRAN            EQU     9       ; translacao
0000                    TRKID           EQU     0       ; trilha corrente
0003                    SIZEID          EQU     3       ; tamanho

                        ;       Macros

                        ACTDSK          MACRO           ; Aciona o disco
                                        RST     08H
                                        ENDM
```

```
                        WAITDSK     MACRO           ; Desliga o disco
                                    RST     28H
                                    ENDM

                        OUTPUT      MACRO           ; Imprime um caracter
                                    RST     10H
                                    ENDM
                        ;
                        INPUT       MACRO           ; Le um caracter
                                    RST     18H
                                    ENDM
                        ;
                        PRTMSG      MACRO           ; Imprime uma cadeia de caracteres
                                    RST     38H
                                    ENDM
                        ;
                        PRINT       MACRO   MSG     ; Imprime e posiciona a cadeia
                                    PRTMSG
                                    DEFM    MSG
                                    DEFB    NULL
                                    ENDM
                        ;
                        PRINTL      MACRO   MSG     ; Imprime a cadeia com CR e LF
                                    PRTMSG
                                    DEFB    CR,LF
                                    DEFM    MSG
                                    DEFB    NULL
                                    ENDM
                        ;
                                    .LIST


                                    ORG     0               ; Inicio do EDOS

0000'  F3                           DI                      ; Confirma os defaults
0001'  D3 04                        OUT     (ROM),A
0003'  ED 56                        IM      1
0005'  C3 0105'                     JP      START

                                    ORG     8H              ; Ativa o disco corrente

0008'  FD 7E 00                     LD      A,(IY+TYP)
000B'  F6 80                        OR      80H
000D'  D3 14                        OUT     (DSKTYP),A
000F'  C9                           RET
                        ;
                                    ORG     10H             ; Imprime um caracter

0010'  C3 0048'                     JP      COUT
                        ;
                                    ORG     18H             ; Le um caracter

0018'  E7               CINP:       RST     20H
0019'  28 FD                        JR      Z,CINP
001B'  DB 10                        IN      A,(SIOAD)
001D'  C9                           RET
                        ;
                                    ORG     20H             ; Le o estado da console

0020'  DB 11            CSTAT:      IN      A,(SIOAS)
0022'  0F                           RRCA
```

```
0023'  9F                        SBC     A,A
0024'  C9                        RET
                    ;
                                 ORG     28H             ; Desliga os discos

0028'  FD 7E 00                  LD      A,(IY+TYP)
002B'  D3 14                     OUT     (DSKTYP),A
002D'  C9                        RET
                    ;
                                 ORG     30H             ; Ponto de entrada do EDOS

0030'  C3 02D0'                  JP      DISCO
                    ;
                                 ORG     38H             ; Imprime um string

0038'  E3           PMSG:        EX      (SP),HL
0039'  F5                        PUSH    AF
003A'  C5                        PUSH    BC
003B'  7E           PMSG1:       LD      A,(HL)
003C'  23                        INC     HL
003D'  B7                        OR      A
003E'  28 04                     JR      Z,PMSG2         ; byte zero ?
0040'  4F                        LD      C,A
                    +            OUTPUT
0042'  18 F7                     JR      PMSG1
0044'  C1           PMSG2:       POP     BC
0045'  F1                        POP     AF
0046'  E3                        EX      (SP),HL
0047'  C9                        RET


0048'  F5           COUT:        PUSH    AF              ; Saida de um caracter ASCII
0049'  DB 11        COUT1:       IN      A,(SIOAS)
004B'  E6 04                     AND     4
004D'  28 FA                     JR      Z,COUT1
004F'  79                        LD      A,C
0050'  D3 10                     OUT     (SIOAD),A
0052'  F1                        POP     AF
0053'  C9                        RET

0054'  7E           PRGPT:       LD      A,(HL)          ; Programa os perifericos
0055'  B7                        OR      A
0056'  C8                        RET     Z
0057'  47                        LD      B,A
0058'  23                        INC     HL
0059'  4E                        LD      C,(HL)
005A'  23                        INC     HL
005B'  ED B3                     OTIR
005D'  18 F5                     JR      PRGPT

                                 ORG     66H             ; Utilizado no pseudo DMA
0066'  ED 45                     RETN

0068'  01 0F 36     PERIF:       DEFB    1,SB253,36H     ; Canal 1=> Gera Clock Serial (TTY)
006B'  02 0C 1A                  DEFB    2,DB253,1AH,00  ; 2 Mhz / 26 = 4800 * 16
006E'  00
006F'  01 0F 76                  DEFB    1,SB253,76H     ; Canal 2=> Gera Clock Serial(PUN RDR)
0072'  02 0D 1A                  DEFB    2,DB253+1,1AH,00 ; 2 Mhz / 26 = 4800 * 16
0075'  00
0076'  01 0F B6                  DEFB    1,SB253,0B6H    ; Canal 3=> Gera Clock p/ Relogio
0079'  02 0E 35                  DEFB    2,DB253+2,53,130 ; 2 Mhz / 33.333 = 60 hz
007C'  82
```

```
                        ;
007D'  01 17 8D                 DEFB    1,C8255,10001101B  ; PA saida, PB saida, PC entrada
                        ;
0080'  08 11 18                 DEFB    8,SIOAS,18H,04H,4CH,03H,0C1H,05H,0EAH,0
0083'  04 4C 03
0086'  C1 05 EA
0089'  00
008A'  08 13 18                 DEFB    8,SIOBS,18H,04H,4CH,03H,0C1H,05H,0EAH,0
008D'  04 4C 03
0090'  C1 05 EA
0093'  00
                        ;
0094'  01 00 02                 DEFB    1,VELOC,VEL4M       ; Velocidade CPU = 4 Mhz

0097'  01 14 00                 DEFB    1,DSKTYP, 0         ; Desliga discos
009A'  00                       DEFB    0                   ; Fim de Programacao de Perifericos


009B'  3E 05            WAIT:   LD      A,5             ; Pequeno atraso
009D'  3D               WAIT1:  DEC     A               ;  para o controlador de discos
009E'  20 FD                    JR      NZ,WAIT1
00A0'  C9                       RET

00A1'  DB 16            CHKTOUT:IN      A,(TIMEOUT)     ; Verifica temporizador do disco
00A3'  E6 20                    AND     20H             ;  se o acionador esta com
00A5'  C8                       RET     Z               ;  problemas
00A6'  3E D0                    LD      A,CMABORT       ; Se problemas cancela o comando
00A8'  D3 18                    OUT     (DSKCMD),A
00AA'  0E 00                    LD      C,0
00AC'  CD 00B2'                 CALL    DSKCOM
00AF'  F6 80                    OR      80H
00B1'  C9                       RET

00B2'  CD 00B8'         DSKCOM: CALL    DSKRDY          ; Envia um comando ao controlador
00B5'  79                       LD      A,C
00B6'  D3 18                    OUT     (DSKCMD),A

00B8'  CD 009B'         DSKRDY: CALL    WAIT            ; Verifica se o controlador
00BB'  DB 18            RDY1:   IN      A,(DSKSTT)      ; esta pronto
00BD'  CB 47                    BIT     0,A
00BF'  20 FA                    JR      NZ,RDY1
00C1'  B7                       OR      A
00C2'  C9                       RET

00C3'  CD 009B'         DSKOK:  CALL    WAIT            ; Retorna o estado do controlador
00C6'  DB 16            DSKOK1: IN      A,(TIMEOUT)     ; junto com o bit de TIMEOUT
00C8'  E6 20                    AND     20H
00CA'  C0                       RET     NZ
00CB'  DB 18                    IN      A,(DSKSTT)
00CD'  E6 80                    AND     80H
00CF'  20 F5                    JR      NZ,DSKOK1
00D1'  C9                       RET

00D2'  11 000A          PDEC:   LD      DE,10           ; Imprime em decimal
00D5'  14               PD0:    INC     D
00D6'  93                       SUB     E
00D7'  30 FC                    JR      NC,PD0
00D9'  83                       ADD     A,E
00DA'  15                       DEC     D
00DB'  5F                       LD      E,A
00DC'  7A                       LD      A,D
00DD'  B7                       OR      A
```

```
00DE'  28 04                    JR      Z,PD1
00E0'  C6 30                    ADD     A,'0'
00E2'  4F                       LD      C,A
                        +       OUTPUT
00E4'  7B               PD1:    LD      A,E
00E5'  C6 30                    ADD     A,'0'
00E7'  4F                       LD      C,A
                        +       OUTPUT
00E9'  C9                       RET

00EA'  7C               OUT_HL: LD      A,H             ; Imprime em hexadecimal
00EB'  CD 00EF'                 CALL    OUT_A
00EE'  7D                       LD      A,L
00EF'  F5               OUT_A:  PUSH    AF
00F0'  1F                       RRA
00F1'  1F                       RRA
00F2'  1F                       RRA
00F3'  1F                       RRA
00F4'  CD 00F8'                 CALL    OUTNB
00F7'  F1                       POP     AF
00F8'  E6 0F            OUTNB:  AND     0FH
00FA'  FE 0A                    CP      10
00FC'  38 02                    JR      C,OUTNB1
00FE'  C6 07                    ADD     A,7
0100'  C6 30            OUTNB1: ADD     A,'0'
0102'  4F                       LD      C,A
                        +       OUTPUT
0104'  C9                       RET

                        ; ===========================================================================
                        ;
                        ;       Inicio real do EDOS

0105'  31 4800          START:  LD      SP,PILHA        ; Estabelece a pilha

0108'  21 0068'                 LD      HL,PERIF        ; Programa os dispositivos
010B'  CD 0054'                 CALL    PRGPT

                        +       PRINTL  '               ! Micro LIE - Edos V1.2 !'

012D'  CD 02C1'                 CALL    VRTCLR          ; Verifica o disco virtual

0130'  CD 0163'                 CALL    TSTRAM          ; Testa a RAM dinamica

0133'  CD 022E'                 CALL    COPSYS          ; Copia o CP/M da EPROM para a RAM

0136'  CD 09A4'                 CALL    DSKIN           ; Inicializa as variaveis do disco

0139'  CD 0249'  STRT1:         CALL    LDDSK           ; Carrega o CP/M do disco A se existir
013C'  20 08                    JR      NZ,STRT2
013E'  3E FF                    LD      A,-1            ; Nao existe, pula para o BIOS
0140'  32 406B                  LD      (SKIP),A
0143'  C3 FA00                  JP      CPMBT

0146'  DB 16            STRT2:  IN      A,(BOOT)        ; Winchester existe ?
0148'  E6 80                    AND     80H
014A'  20 0E                    JR      NZ,NOSYST       ; nao

014C'  3A 406A                  LD      A,(MASTER)      ; Sim, troca o disco mestre
014F'  FE 04                    CP      4
0151'  28 07                    JR      Z,NOSYST
```

```
0153'  3E 04                    LD      A,4             ; Winchester = A
0155'  32 406A                  LD      (MASTER),A      ; Disco A    = E
0158'  18 DF                    JR      STRT1

015A'  AF           NOSYST:     XOR     A               ; Sem sistema na EPROM e no disco
015B'  32 406A                  LD      (MASTER),A      ; imprime uma mensagem
015E'  CD 03C8'                 CALL    ERRMSG
0161'  18 D6                    JR      STRT1           ; Tenta novamente

0163'  CD 01F2'     TSTRAM:     CALL    TSTRM           ; Testa a memoria RAM dinamica
0166'  38 08                    JR      C,TSTERR        ; Erro ?
0168'  D3 05                    OUT     (INVBNC),A
016A'  CD 01F2'                 CALL    TSTRM           ; Testa o outro banco
016D'  D3 04                    OUT     (NRMBNC),A
016F'  D0                       RET     NC
0170'              +TSTERR:     PRINTL  'Warning : RAM error at address '
0193'  CD 00EA'                 CALL    OUT_HL
0196'              +TSTER1:     PRINTL  'Do you want to continue (boot)? (Y/N)'
                   +            INPUT
01C0'  E6 5F                    AND     5FH
01C2'  4F                       LD      C,A
                   +            OUTPUT
01C4'  FE 59                    CP      'Y'             ; Boot ?
01C6'  C8                       RET     Z
01C7'  FE 4E                    CP      'N'
01C9'  20 CB                    JR      NZ,TSTER1
                   +            PRINTL  'Checking RAM forever...'
01E6'  CD 01F2'     TSTR1:      CALL    TSTRM           ; Testa continuamente a RAM
01E9'  D3 05                    OUT     (INVBNC),A
01EB'  CD 01F2'                 CALL    TSTRM
01EE'  D3 04                    OUT     (NRMBNC),A
01F0'  18 F4                    JR      TSTR1

01F2'  21 8000      TSTRM:      LD      HL,8000H        ; Rotina para testar o banco
01F5'  01 8000                  LD      BC,8000H        ; superior da RAM 8000H a FFFFH
01F8'  AF                       XOR     A               ; Testa com zero
01F9'  CD 020B'                 CALL    FILCMP
01FC'  D8                       RET     C
01FD'  3E 55                    LD      A,55H           ; Testa com 55H
01FF'  CD 020B'                 CALL    FILCMP
0202'  D8                       RET     C
0203'  3E AA                    LD      A,0AAH          ; Testa com AAH
0205'  CD 020B'                 CALL    FILCMP
0208'  D8                       RET     C
0209'  3E FF                    LD      A,0FFH          ; Testa com FFH

020B'  E5           FILCMP:     PUSH    HL              ; Testa com um determinado padrao
020C'  C5                       PUSH    BC
020D'  5F                       LD      E,A
020E'  77                       LD      (HL),A
020F'  0B                       DEC     BC
0210'  78                       LD      A,B
0211'  B1                       OR      C
0212'  7B                       LD      A,E
0213'  28 05                    JR      Z,COMPAR
0215'  54                       LD      D,H
0216'  5D                       LD      E,L
0217'  13                       INC     DE
0218'  ED B0                    LDIR                    ; Preenche
021A'  C1           COMPAR:     POP     BC
021B'  E1                       POP     HL
021C'  E5                       PUSH    HL
```

```
021D'  C5                     PUSH    BC
021E'  ED A1        CMPLP:    CPI                     ; Testa
0220'  20 07                  JR      NZ,CMPER
0222'  EA 021E'               JP      PE,CMPLP
0225'  C1                     POP     BC
0226'  E1                     POP     HL
0227'  B7                     OR      A
0228'  C9                     RET
                      ;
0229'  C1           CMPER:   POP     BC             ; Erro de comparacao
022A'  D1                     POP     DE
022B'  2B                     DEC     HL
022C'  37                     SCF                    ; retorna com CARRY flag ligada
022D'  C9                     RET


022E'  3E C3        COPSYS:   LD      A,0C3H         ; Copia o sistema se existir na
0230'  32 FA00                LD      (CPMBT),A      ; Eprom do usuario
0233'  21 015A'               LD      HL,NOSYST
0236'  22 FA01                LD      (CPMBT+1),HL

0239'  21 2000                LD      HL,2000H
023C'  3E C3                  LD      A,0C3H
023E'  BE                     CP      (HL)           ; Existe ?
023F'  C0                     RET     NZ

0240'  01 1BFF                LD      BC,1BFFH       ; Sim, copia
0243'  11 E400                LD      DE,DEFCPM
0246'  ED B0                  LDIR

0248'  C9                     RET

0249'  AF           LDDSK:   XOR     A              ; Carrega os parametros defaults
024A'  32 4057                LD      (HSTACT),A
024D'  32 4059                LD      (UNACNT),A
0250'  32 4058                LD      (HSTWRT),A
0253'  1E 00                  LD      E,0
0255'  0E 00                  LD      C,0
0257'  CD 0911'               CALL    SELDSK         ; Seleciona o disco A
025A'  E6 F0                  AND     0F0H
025C'  C0                     RET     NZ             ; Erro, retorna

025D'  21 0000                LD      HL,0
0260'  22 4052                LD      (SEKTRK),HL    ; Trilha = 0
0263'  3E 01                  LD      A,1
0265'  32 4054                LD      (SEKSEC),A     ; Setor = 1
0268'  21 E300                LD      HL,DEFCPM-100H ; Ender. acima do CP/M
026B'  22 4055                LD      (DMAADR),HL

026E'  CD 0453'               CALL    READ           ; Le  o setor

0271'  B7                     OR      A
0272'  C0                     RET     NZ             ; Erro, retorna

0273'  21 E300                LD      HL,DEFCPM-100H

0276'  7E                     LD      A,(HL)         ; O disco contem os parametros ?
0277'  B7                     OR      A
0278'  FA 0283'               JP      M,OPCAO        ; Nao

027B'  11 4000                LD      DE,4000H       ; Sim copia-os para a area do
027E'  01 0050                LD      BC,NUMPAR*5    ;  EDOS
```

```
0281'  ED B0                    LDIR

0283'  3E 03        OPCAO:      LD      A,3             ; Setor = 3
0285'  32 4054                  LD      (SEKSEC),A
0288'  CD 0453'                 CALL    READ            ; Le
028B'  B7                       OR      A
028C'  C0                       RET     NZ

028D'  3A E300                  LD      A,(DEFCPM-100H) ; O primeiro byte tem o numero de
0290'  B7                       OR      A               ;  setores a serem transferidos
0291'  F2 0296'                 JP      P,OPCA1
0294'  AF                       XOR     A
0295'  C9                       RET
0296'  2A E303      OPCA1:      LD      HL,(DEFCPM-100H+3)
0299'  E5                       PUSH    HL                      ; End. empinhado
029A'  C8                       RET     Z
029B'  47                       LD      B,A                     ; # de setores a transferir
029C'  0E 04                    LD      C,4
029E'  2A E301                  LD      HL,(DEFCPM-100H+1)      ; Ender. a ser tranferido

02A1'  3E FF                    LD      A,-1                    ; Pula mensagens de erro
02A3'  32 406B                  LD      (SKIP),A

02A6'  C5           SIGLOP:     PUSH    BC              ; Transfere os setores pedidos
02A7'  E5                       PUSH    HL
02A8'  79                       LD      A,C
02A9'  32 4054                  LD      (SEKSEC),A      ; Setor
02AC'  22 4055                  LD      (DMAADR),HL     ; DMA
02AF'  CD 0453'                 CALL    READ            ; Le
02B2'  E1                       POP     HL
02B3'  C1                       POP     BC
02B4'  B7                       OR      A
02B5'  28 02                    JR      Z,SIGLP1
02B7'  E1                       POP     HL
02B8'  C9                       RET

02B9'  11 0080      SIGLP1:     LD      DE,80H          ; DMA = DMA + 128
02BC'  19                       ADD     HL,DE
02BD'  0C                       INC     C
02BE'  10 E6                    DJNZ    SIGLOP          ; Repete ate o fim
02C0'  C9                       RET                     ; Pula para o endereco especificado


02C1'  21 1000      VRTCLR:     LD      HL,4096         ; Incializa o disco virtual
02C4'  DB C3                    IN      A,(0C3H)
02C6'  3E E5        VRTCL:      LD      A,0E5H          ; Valor = E5H
02C8'  D3 C0                    OUT     (0C0H),A
02CA'  2B                       DEC     HL
02CB'  7C                       LD      A,H
02CC'  B5                       OR      L
02CD'  20 F7                    JR      NZ,VRTCL
02CF'  C9                       RET

02D0'  22 4070      DISCO:      LD      (USRHL),HL      ; Rotina chamada pelo BIOS
02D3'  ED 73 4072               LD      (USRSP),SP
02D7'  31 4800                  LD      SP,PILHA
02DA'  21 030C'                 LD      HL,RETURN
02DD'  E5                       PUSH    HL
02DE'  B7                       OR      A
02DF'  28 70                    JR      Z,MSGOFF        ; funcao 0
02E1'  3D                       DEC     A
02E2'  28 38                    JR      Z,MSGOUT        ; funcao 1
```

```
02E4'  3D                        DEC     A
02E5'  28 2D                     JR      Z,CHGDSK        ; funcao 2.
02E7'  3D                        DEC     A
02E8'  CA 0911'                  JP      Z,SELDSK        ; Seleciona disco, funcao 3
02EB'  F5                        PUSH    AF
02EC'  CD 05F0'                  CALL    SETIY
02EF'  F1                        POP     AF
02F0'  3D                        DEC     A
02F1'  CA 0453'                  JP      Z,READ          ; Le um setor, funcao 4
02F4'  3D                        DEC     A
02F5'  CA 045E'                  JP      Z,WRITE         ; Escreve um setor, funcao 5
                       ;
02F8'  3D                        DEC     A
02F9'  CA 0385'                  JP      Z,MSGINV        ; Funcao 6, jump invalido
02FC'  3D                        DEC     A
02FD'  CA 03AD'                  JP      Z,MSGWBR        ; Funcao 7, Erro no boot
0300'  3D                        DEC     A
0301'  CA 0414'                  JP      Z,RETDPB        ; Funcao 8, retorna DPB
0304'  3D                        DEC     A
0305'  CA 0421'                  JP      Z,SAVDPB        ; Funcao 9, salva DPB
0308'  3D                        DEC     A
0309'  CA 0105'                  JP      Z,START         ; Reinicializa

030C'  ED 7B 4072    RETURN:     LD      SP,(USRSP)      ; Retorna ao BIOS
0310'  2A 4070                   LD      HL,(USRHL)
0313'  E9                        JP      (HL)
                       ;
0314'  79            CHGDSK:     LD      A,C             ; Seleciona um acionador
0315'  FE 06                     CP      6
0317'  D0                        RET     NC
0318'  32 406A                   LD      (MASTER),A
031B'  C9                        RET
                       ;
031C'                +MSGOUT:    PRTMSG
031D'  0D 0A 07                  defb    CR,LF,BELL,BELL
0320'  07
0321'  54 68 65                  defm    'The printer is OUT OF PAPER, please LOAD it.'
0324'  20 70 72
0327'  69 6E 74
032A'  65 72 20
032D'  69 73 20
0330'  4F 55 54
0333'  20 4F 46
0336'  20 50 41
0339'  50 45 52
033C'  2C 20 70
033F'  6C 65 61
0342'  73 65 20
0345'  4C 4F 41
0348'  44 20 69
034B'  74 2E
034D'  0D 0A 00                  defb    CR,LF,NULL
0350'  C9                        RET
                       ;
0351'                +MSGOFF:    PRTMSG
0352'  0D 0A 07                  defb    CR,LF,BELL,BELL
0355'  07
0356'  54 68 65                  defm    'The printer is OFF LINE, please CONNECT it.'
0359'  20 70 72
035C'  69 6E 74
035F'  65 72 20
0362'  69 73 20
```

```
0365'  4F 46 46
0368'  20 4C 49
036B'  4E 45 2C
036E'  20 70 6C
0371'  65 61 73
0374'  65 20 43
0377'  4F 4E 4E
037A'  45 43 54
037D'  20 69 74
0380'  2E
0381'  0D 0A 00                    defb    CR,LF,NULL
0384'  C9                          RET
                        ;
0385'  C5               MSGINV:    PUSH    BC              ; Jump invalido
                       +           PRTMSG
0387'  0D 0A 07                    defb    CR,LF,BELL
038A'  4A 75 6D                    defm    'Jump invalid at address '
038D'  70 20 69
0390'  6E 76 61
0393'  6C 69 64
0396'  20 61 74
0399'  20 61 64
039C'  64 72 65
039F'  73 73 20
03A2'  00                          defb    NULL
03A3'  E1                          POP     HL
03A4'  CD 00EA'                    CALL    OUT_HL
                       +           PRTMSG
03A8'  48                          defm    'H'
03A9'  0D 0A 00                    defb    CR,LF,NULL
03AC'  C9                          RET
                        ;
03AD'                  +MSGWBR:    PRINTL  'Warning: Error on BOOT!'
03C8'                  +ERRMSG:    PRINTL  'Non-System disk or disk error'
                       +           PRINTL  'Replace and strike any key when ready'
                       +           INPUT
0413'  C9                          RET

0414'  01 0050         RETDPB:     LD      BC,NUMPAR*5     ; Retorna o DPB
0417'  21 4000                     LD      HL,TABDSK
041A'  CD 042F'        RETDP1:     CALL    POKE
041D'  EA 041A'                    JP      PE,RETDP1
0420'  C9                          RET

0421'  01 0050         SAVDPB:     LD      BC,NUMPAR*5     ; Salva o DPB
0424'  21 4000                     LD      HL,TABDSK
0427'  EB                          EX      DE,HL
0428'  CD 0441'        SAVDP1:     CALL    PEEK
042B'  EA 0428'                    JP      PE,SAVDP1
042E'  C9                          RET

042F'  CB 7A           POKE:       BIT     7,D             ; Localiza o banco e transfere
0431'  28 03                       JR      Z,POKE1
0433'  ED A0                       LDI
0435'  C9                          RET
0436'  CB FA           POKE1:      SET     7,D
0438'  D3 05                       OUT     (INVBNC),A
043A'  ED A0                       LDI
043C'  D3 04                       OUT     (NRMBNC),A
043E'  CB BA                       RES     7,D
0440'  C9                          RET
```

```
0441'  CB 7C       PEEK:     BIT   7,H          ; Localiza o banco e traz os dados
0443'  28 03                 JR    Z,PEEK1
0445'  ED A0                 LDI
0447'  C9                    RET
0448'  CB FC       PEEK1:    SET   7,H
044A'  D3 05                 OUT   (INV2NC),A
044C'  ED A0                 LDI
044E'  D3 04                 OUT   (NRMBNC),A
0450'  CB BC                 RES   7,H
0452'  C9                    RET

                   ;         ==================================
                   ;
                   ;          Rotinas de blocking e deblocking
                   ;
                   ;         ==================================

0453'  3E 01       READ:     LD    A,1            ; Le um setor logico
0455'  32 4067               LD    (READOP),A
0458'  3C                    INC   A
0459'  32 4069               LD    (WRTYPE),A     ; Tipo escrita nao alocando
045C'  18 5E                 JR    ALLOC

                   ;
045E'  AF          WRITE:    XOR   A              ; Escrita de um setor logico
045F'  32 4067               LD    (READOP),A
0462'  79                    LD    A,C
0463'  32 4069               LD    (WRTYPE),A     ; Tipo
0466'  FE 04                 CP    WRFLU
0468'  CA 0567'              JP    Z,FLUSH        ; Descarrega os buffers ?
046B'  FE 02                 CP    WRUAL          ; Aloca clusters
046D'  20 11                 JR    NZ,CHKUNA
046F'  FD 7E 05              LD    A,(IY+BCKCNT)
0472'  32 4059               LD    (UNACNT),A
0475'  11 405C               LD    DE,UNADSK      ; procura = fornecidos
0478'  21 4051               LD    HL,SEKDSK
047B'  01 0004               LD    BC,4
047E'  ED B0                 LDIR
0480'  3A 4059     CHKUNA:   LD    A,(UNACNT)
0483'  B7                    OR    A
0484'  28 36                 JR    Z,ALLOC
0486'  3D                    DEC   A
0487'  32 4059               LD    (UNACNT),A
048A'  11 4051               LD    DE,SEKDSK
048D'  21 405C               LD    HL,UNADSK
0490'  1A                    LD    A,(DE)
0491'  BE                    CP    (HL)           ;Mesmo Disco
0492'  20 28                 JR    NZ,ALLOC
                   ;
0494'  13                    INC   DE
0495'  23                    INC   HL
0496'  1A                    LD    A,(DE)         ;Mesma Trilha
0497'  BE                    CP    (HL)
0498'  20 22                 JR    NZ,ALLOC
049A'  13                    INC   DE
049B'  23                    INC   HL
049C'  1A                    LD    A,(DE)
049D'  BE                    CP    (HL)
049E'  20 1C                 JR    NZ,ALLOC
                   ;
04A0'  13                    INC   DE
04A1'  23                    INC   HL
```

```
04A2'  1A                        LD      A,(DE)          ;Mesmo Setor
04A3'  BE                        CP      (HL)
04A4'  20 16                     JR      NZ,ALLOC
                         ;
04A6'  34                        INC     (HL)
04A7'  7E                        LD      A,(HL)
04A8'  FD BE 07                  CP      (IY+CPMSPT)
04AB'  38 09                     JR      C,NODVF
04AD'  36 00                     LD      (HL),0
04AF'  2A 405D                   LD      HL,(UNATRK)
04B2'  23                        INC     HL
04B3'  22 405D                   LD      (UNATRK),HL
04B6'  AF              NODVF:    XOR     A
04B7'  32 4066                   LD      (RSFLAG),A
04BA'  18 08                     JR      RWOPER
04BC'  AF              ALLOC:    XOR     A
04BD'  32 4059                   LD      (UNACNT),A
04C0'  3C                        INC     A
04C1'  32 4066                   LD      (RSFLAG),A
                         ;
04C4'  AF              RWOPER:   XOR     A               ; Le e escreve
04C5'  32 4065                   LD      (ERFLAG),A
04C8'  3A 4054                   LD      A,(SEKSEC)
04CB'  FD 46 06                  LD      B,(IY+SECSHF)
04CE'  CB 38           RWOP2:    SRL     B
04D0'  38 04                     JR      C,RWOP1
04D2'  CB 3F                     SRL     A
04D4'  18 F8                     JR      RWOP2
04D6'  32 405B         RWOP1:    LD      (SEKHST),A
04D9'  21 4057                   LD      HL,HSTACT
04DC'  7E                        LD      A,(HL)
04DD'  36 01                     LD      (HL),1
04DF'  B7                        OR      A
04E0'  28 25                     JR      Z,FILHST        ; Preenche os buffers
04E2'  3A 4051                   LD      A,(SEKDSK)
04E5'  21 4060                   LD      HL,HSTDSK
04E8'  BE                        CP      (HL)
04E9'  20 15                     JR      NZ,NOMATC
04EB'  3A 4052                   LD      A,(SEKTRK)
04EE'  23                        INC     HL
04EF'  BE                        CP      (HL)
04F0'  20 0E                     JR      NZ,NOMATC
04F2'  3A 4053                   LD      A,(SEKTRK+1)
04F5'  23                        INC     HL
04F6'  BE                        CP      (HL)
04F7'  20 07                     JR      NZ,NOMATC
04F9'  3A 405B                   LD      A,(SEKHST)
04FC'  23                        INC     HL
04FD'  BE                        CP      (HL)
04FE'  28 32                     JR      Z,MATCH
0500'  3A 4058         NOMATC:   LD      A,(HSTWRT)
0503'  B7                        OR      A
0504'  C4 0573'                  CALL    NZ,WRTHST
0507'  3A 4051         FILHST:   LD      A,(SEKDSK)      ; Preenche os buffers
050A'  32 4060                   LD      (HSTDSK),A
050D'  2A 4052                   LD      HL,(SEKTRK)
0510'  22 4061                   LD      (HSTTRK),HL
0513'  3A 405B                   LD      A,(SEKHST)
0516'  32 4063                   LD      (HSTSEC),A
0519'  3A 4066                   LD      A,(RSFLAG)
051C'  B7                        OR      A
051D'  28 0F                     JR      Z,FILHS1
```

```
051F'  FD 7E 00              LD      A,(IY+TYP)
0522'  E6 20                 AND     20H
0524'  EE 20                 XOR     20H
0526'  47                    LD      B,A
0527'  3A 4067               LD      A,(READOP)
052A'  B0                    OR      B
052B'  C4 0576'              CALL    NZ,RDRHST       ; Le antes de escrever
052E'  AF          FILHS1:   XOR     A
052F'  32 4058               LD      (HSTWRT),A
0532'  3A 4054     MATCH:    LD      A,(SEKSEC)
0535'  FD 46 06              LD      B,(IY+SECMSK)
0538'  05                    DEC     B
0539'  A0                    AND     B
053A'  26 00                 LD      H,0
053C'  6F                    LD      L,A
               +             REPT    7
               +             ADD     HL,HL
               +             ENDM
0544'  11 407C               LD      DE,BUFDSK       ; Transfere os dados
0547'  19                    ADD     HL,DE
0548'  ED 5B 4055            LD      DE,(DMAADR)
054C'  3A 4067               LD      A,(READOP)
054F'  B7                    OR      A
0550'  20 06                 JR      NZ,RWMOVE
0552'  3C                    INC     A
0553'  32 4058               LD      (HSTWRT),A
0556'  EB                    EX      DE,HL
0557'  AF                    XOR     A
0558'  CD 06F0'    RWMOVE:   CALL    TRANSF
055B'  3A 4069               LD      A,(WRTYPE)
055E'  FE 01                 CP      WRDIR
0560'  20 0C                 JR      NZ,RWEND
0562'  3A 4065               LD      A,(ERFLAG)
0565'  B7                    OR      A
0566'  C0                    RET     NZ
0567'  AF          FLUSH:    XOR     A
0568'  32 4058               LD      (HSTWRT),A
056B'  CD 0573'              CALL    WRTHST
056E'  3A 4065     RWEND:    LD      A,(ERFLAG)
0571'  B7                    OR      A
0572'  C9                    RET
               ;
               ;
0573'  AF          WRTHST:   XOR     A               ; Escrita do setor fisico
0574'  18 02                 JR      RDRWRT
0576'  3E 01       RDRHST:   LD      A,1             ; Leitura do setor fisico
0578'  32 4068     RDRWRT:   LD      (HSTOP),A
057B'  3A 4060               LD      A,(HSTDSK)
057E'  FE 04                 CP      4
0580'  CA 0A97'              JP      Z,RDWTWC
0583'  FD E5                 PUSH    IY
0585'  CD 05F3'              CALL    SETIYA
0588'             +NEWRW:    ACTDSK
0589'  CD 00C3'              CALL    DSKOK
058C'  20 30                 JR      NZ,FATERR
058E'  3E 0A                 LD      A,10            ; Numero de tentativas
0590'  32 4064     RWHST:    LD      (HSTTRY),A
0593'  CD 0603'              CALL    SEEKTR          ; Procura a trilha
0596'  CD 0647'              CALL    SEEKSC          ; Acerta o setor
               +             ACTDSK
059A'  21 407C               LD      HL,HSTBUF
059D'  3A 4068               LD      A,(HSTOP)
```

```
05A0'  B7                        OR      A
05A1'  28 05                     JR      Z,RWHST1
05A3'  CD 0744'                  CALL    RDRDSK          ; Leitura fisica
05A6'  18 03                     JR      RWHST2
05A8'  CD 077C'      RWHST1:     CALL    WRTDSK          ; Escrita fisica
05AB'  B7            RWHST2:     OR      A
05AC'  28 3F                     JR      Z,NOERR
05AE'  07                        RLCA
05AF'  38 0D                     JR      C,FATERR
05B1'  3A 4064                   LD      A,(HSTTRY)      ; Erro, tenta novamente
05B4'  3D                        DEC     A
05B5'  28 07                     JR      Z,FATERR
05B7'  F5                        PUSH    AF
05B8'  CD 0728'                  CALL    CHECK           ; Mesma trilha
05BB'  F1                        POP     AF
05BC'  18 D2                     JR      RWHST
05BE'  3A 406B       FATERR:     LD      A,(SKIP)        ; Se interno, pula mensagens
05C1'  B7                        OR      A
05C2'  28 23                     JR      Z,FTSKP
05C4'  CD 07D4'                  CALL    ERROR
05C7'                +FATQ:      PRINTL  'Retry, Fail? ' ; Usuario deside o que fazer
                     +           INPUT
05D9'  C5                        PUSH    BC
05DA'  4F                        LD      C,A
                     +           OUTPUT
05DC'  C1                        POP     BC
05DD'  E6 5F                     AND     5FH
05DF'  FE 52                     CP      'R'             ; Retentar ?
05E1'  28 A5                     JR      Z,NEWRW
05E3'  FE 46                     CP      'F'             ; Retornar com o erro ?
05E5'  20 E0                     JR      NZ,FATQ
05E7'                +FTSKP:     WAITDSK
05E8'  3E FF                     LD      A,-1
05EA'  32 4065                   LD      (ERFLAG),A      ; Erro!
05ED'  FD E1         NOERR:      POP     IY
05EF'  C9                        RET

05F0'  3A 4051       SETIY:      LD      A,(SEKDSK)      ; Seleciona os parametros
05F3'  87            SETIYA:     ADD     A,A             ; do disco pedido
05F4'  87                        ADD     A,A
05F5'  87                        ADD     A,A
05F6'  87                        ADD     A,A
05F7'  C5                        PUSH    BC
05F8'  4F                        LD      C,A
05F9'  06 00                     LD      B,0
05FB'  FD 21 4000                LD      IY,TABDSK
05FF'  FD 09                     ADD     IY,BC
0601'  C1                        POP     BC
0602'  C9                        RET
                     ;
0603'  CD 00B8'      SEEKTR:     CALL    DSKRDY          ; Avanca para a trilha pedida
0606'  FD 7E 01                  LD      A,(IY+TRK)
0609'  D3 19                     OUT     (DSKTRK),A
060B'  3A 4061                   LD      A,(HSTTRK)
060E'  FD BE 02                  CP      (IY+TRKTOP)
0611'  F5                        PUSH    AF
0612'  CD 065F'                  CALL    SHIFT           ; Muda de lado
0615'  F1                        POP     AF
0616'  38 03                     JR      C,SEEKT1
0618'  FD 96 02                  SUB     (IY+TRKTOP)
061B'  FD 4E 04      SEEKT1:     LD      C,(IY+PARDSK)
061E'  CB 69                     BIT     5,C
```

```
0620'  28 02                    JR      Z,SEEKT2
0622'  CB 27                    SLA     A
0624'  FD BE 01      SEEKT2:    CP      (IY+TRK)
0627'  C8                       RET     Z

0628'  FD 77 01                 LD      (IY+TRK),A
062B'  D3 1B                    OUT     (DSKDAT),A
062D'  3E 1B                    LD      A,CMSEEK        ; Envia comando ao controlador
062F'  FD B6 08                 OR      (IY+STEP)
0632'  4F                       LD      C,A
0633'  CD 00B2'                 CALL    DSKCOM
0636'  F5                       PUSH    AF

0637'  FD 7E 04                 LD      A,(IY+PARDSK)
063A'  CB 6F                    BIT     5,A
063C'  28 07                    JR      Z,SEEKT3

063E'  FD 7E 01                 LD      A,(IY+TRK)
0641'  CB 3F                    SRL     A
0643'  D3 19                    OUT     (DSKTRK),A
0645'  F1            SEEKT3:    POP     AF
0646'  C9                       RET

0647'  CD 00B8'      SEEKSC:    CALL    DSKRDY
064A'  3A 4063                  LD      A,(HSTSEC)
064D'  FD BE 03                 CP      (IY+SECTOP)
0650'  F5                       PUSH    AF
0651'  38 03                    JR      C,SEEKS1
0653'  FD 96 03                 SUB     (IY+SECTOP)
0656'  CD 066F'      SEEKS1:    CALL    TRASEC
0659'  32 406E                  LD      (AUX1),A
065C'  D3 1A                    OUT     (DSKSEC),A
065E'  F1                       POP     AF
                     ;
065F'  FD 7E 00      SHIFT:     LD      A,(IY+TYP)      ; Muda de lado
0662'  47                       LD      B,A
0663'  CB B0                    RES     6,B
0665'  38 02                    JR      C,SHIFT1
0667'  CB F0                    SET     6,B
0669'  A8            SHIFT1:    XOR     B
066A'  C8                       RET     Z
066B'  FD 70 00                 LD      (IY+TYP),B
066E'  C9                       RET
                     ;
066F'  47            TRASEC:    LD      B,A             ; Translada o setor
0670'  FD 7E 04                 LD      A,(IY+PARDSK)
0673'  E6 1C                    AND     1CH
0675'  20 02                    JR      NZ,TRAS1
0677'  78                       LD      A,B
0678'  C9                       RET
0679'  FE 04         TRAS1:     CP      4
067B'  20 03                    JR      NZ,TRAS2
067D'  78                       LD      A,B
067E'  3C                       INC     A
067F'  C9                       RET
0680'  FE 1C         TRAS2:     CP      1CH
0682'  20 0A                    JR      NZ,TRA3
0684'  FD 6E 09                 LD      L,(IY+TRAN)
0687'  FD 66 0A                 LD      H,(IY+TRAN+1)
068A'  16 00                    LD      D,0
068C'  18 0C                    JR      TRA4
068E'  0F            TRA3:      RRCA
```

```
068F'  16 00                    LD      D,0
0691'  5F                       LD      E,A
0692'  21 069A'                 LD      HL,TRATAB-4
0695'  19                       ADD     HL,DE
0696'  7E                       LD      A,(HL)
0697'  23                       INC     HL
0698'  66                       LD      H,(HL)
0699'  6F                       LD      L,A
069A'  58               TRA4:   LD      E,B
069B'  19                       ADD     HL,DE
069C'  7E                       LD      A,(HL)
069D'  C9                       RET
                        ;
069E'  06A8'            TRATAB: DEFW    TAB1            ; Endereco das tabelas de translacao
06A0'  06C2'                    DEFW    TAB2            ;       default
06A2'  06CC'                    DEFW    TAB3
06A4'  06D6'                    DEFW    TAB4
06A6'  06E0'                    DEFW    TAB5
                        ;
                        ;###############################################################################
                        ;
06A8'  01 07 0D         TAB1:   DEFB    1,7,13,19,25,5,11,17,23,3,9,15,21
06AB'  13 19 05
06AE'  0B 11 17
06B1'  03 09 0F
06B4'  15
06B5'  02 08 0E                 DEFB    2,8,14,20,26,6,12,18,24,4,10,16,22      ; 26 SETORES 1:6
06B8'  14 1A 06
06BB'  0C 12 18
06BE'  04 0A 10
06C1'  16
06C2'  01 03 05         TAB2:   DEFB    1,3,5,7,9,2,4,6,8,10                    ; 10 SETORES 1:2
06C5'  07 09 02
06C8'  04 06 08
06CB'  0A
06CC'  01 04 07         TAB3:   DEFB    1,4,7,10,2,5,8,3,6,9                    ; 10 SETORES 1:3
06CF'  0A 02 05
06D2'  08 03 06
06D5'  09
06D6'  01 05 09         TAB4:   DEFB    1,5,9,2,6,10,3,7,4,8                    ; 10 SETORES 1:4
06D9'  02 06 0A
06DC'  03 07 04
06DF'  08
06E0'  01 0C 07         TAB5:   DEFB    1,12,7,2,13,8,3,14,9,4,15,10,5,16,11,6  ; 16 SETORES 1:11
06E3'  02 0D 08
06E6'  03 0E 09
06E9'  04 0F 0A
06EC'  05 10 0B
06EF'  06
                        ;
                        ;###############################################################################
                        ;

06F0'  ED 4B 4055       TRANSF: LD      BC,(DMAADR)     ; Transfere os dados
06F4'  CB 78                    BIT     7,B
06F6'  01 0080                  LD      BC,128
06F9'  28 03                    JR      Z,TRAN1
06FB'  ED B0                    LDIR
06FD'  C9                       RET
06FE'  D3 05            TRAN1:  OUT     (INVBNC),A
0700'  B7                       OR      A               ; READ=1 , WRITE=0
0701'  20 11                    JR      NZ,TRAN2
```

```
0703'  CB FC                    SET     7,H
0705'  ED A0            TRAN4:  LDI
0707'  E2 0725'                 JP      PO,TRAN3
070A'  7C                       LD      A,H
070B'  B5                       OR      L
070C'  20 F7                    JR      NZ,TRAN4
070E'  CB FC                    SET     7,H
0710'  D3 04                    OUT     (NRMBNC),A
0712'  18 F1                    JR      TRAN4
0714'  CB FA            TRAN2:  SET     7,D
0716'  ED A0            TRAN5:  LDI
0718'  E2 0725'                 JP      PO,TRAN3
071B'  7A                       LD      A,D
071C'  B3                       OR      E
071D'  20 F7                    JR      NZ,TRAN5
071F'  CB FA                    SET     7,D
0721'  D3 04                    OUT     (NRMBNC),A
0723'  18 F1                    JR      TRAN5
0725'  D3 04            TRAN3:  OUT     (NRMBNC),A
0727'  C9                       RET
                        ;
                        ;
072B'  CD 08DE'         CHECK:  CALL    READID          ; Verifica onde a cabeca do
072B'  20 07                    JR      NZ,NOCHE        ;  acionador esta posicionada
072D'  3A 4074                  LD      A,(BUFFID+TRKID)
0730'  FD BE 01                 CP      (IY+TRK)        ; Mesma trilha retorna
0733'  C8                       RET     Z
0734'  FD 36 01         NOCHE:  LD      (IY+TRK),0      ; Reposiciona na trilha zero
0737'  00
0738'  3E 08                    LD      A,CMHOME
073A'  FD B6 08         CHKSTP: OR      (IY+STEP)
073D'  4F                       LD      C,A
073E'  CD 00B2'                 CALL    DSKCOM
0741'  E6 9B                    AND     10011011B
0743'  C9                       RET


                        ;
0744'  CD 00B8'         RDRDSK: CALL    DSKRDY          ; Leitura dos setores
0747'  01 B01B                  LD      BC,2000H+DSKDAT
074A'  FD 7E 04                 LD      A,(IY+PARDSK)
074D'  E6 03                    AND     3
074F'  28 17                    JR      Z,RDDSK
0751'  06 00                    LD      B,0             ; SETOR DE 256 BYTES ?
0753'  3D                       DEC     A
0754'  28 12                    JR      Z,RDDSK
0756'  3E 80                    LD      A.CMREAD        ; Nao 512
0758'  D3 18                    OUT     (DSKCMD),A
075A'  76               RDD1:   HALT                    ; Pseudo DMA
075B'  ED A2                    INI
075D'  C2 075A'                 JP      NZ,RDD1
0760'  76               RDD2:   HALT
0761'  ED A2                    INI
0763'  C2 0760'                 JP      NZ.RDD2
0766'  18 09                    JR      RDD3
                        ;
0768'  3E 80            RDDSK:  LD      A,CMREAD        ; Le 256 bytes
076A'  D3 18                    OUT     (DSKCMD),A
076C'  76               RDD:    HALT
076D'  ED A2                    INI
076F'  20 FB                    JR      NZ,RDD
```

```
0771'  CD 07AF'     RDD3:       CALL    COMPLT


0774'  CD 00A1'     RWHEND:     CALL    CHKTOUT         ; Verifica TIMEOUT
0777'  C0                       RET     NZ
                   +            WAITDSK
0779'  C3 00B8'                 JP      DSKRDY
                   ;
077C'  CD 07AF'     WRTDSK:     CALL    COMPLT
077F'  CD 00B8'                 CALL    DSKRDY
0782'  01 801B                  LD      BC,8000H+DSKDAT
0785'  FD 7E 04                 LD      A,(IY+PARDSK)
0788'  E6 03                    AND     3
078A'  28 17                    JR      Z,WRDSK         ; setor maior que 256 ?
078C'  06 00                    LD      B,0
078E'  3D                       DEC     A
078F'  28 12                    JR      Z,WRDSK
0791'  3E A0                    LD      A,CMWRITE       ; sim
0793'  D3 18                    OUT     (DSKCMD),A
0795'  76           WRT1:      HALT
0796'  ED A3                    OUTI
0798'  C2 0795'                 JP      NZ,WRT1
079B'  76           WRT2:      HALT
079C'  ED A3                    OUTI
079E'  C2 079B'                 JP      NZ,WRT2
07A1'  18 0A                    JR      WRT3

07A3'  3E A0        WRDSK:      LD      A,CMWRITE       ; Nao
07A5'  D3 18                    OUT     (DSKCMD),A
07A7'  76           WRD:       HALT
07A8'  ED A3                    OUTI
07AA'  C2 07A7'                 JP      NZ,WRD
07AD'  18 C5        WRT3:       JR      RWHEND

07AF'  FD 7E 04     COMPLT:     LD      A,(IY+PARDSK)
07B2'  CB 7F                    BIT     7,A
07B4'  C8                       RET     Z
07B5'  C5                       PUSH    BC
07B6'  E5                       PUSH    HL
07B7'  E6 03                    AND     3
07B9'  01 8001                  LD      BC,8001H            ; 128 Bytes
07BC'  28 07                    JR      Z,COMPL2
07BE'  06 00                    LD      B,0                 ; 256 bytes
07C0'  3D                       DEC     A
07C1'  28 02                    JR      Z,COMPL2
07C3'  0E 02                    LD      C,2                 ; 512 bytes
07C5'  21 407C      COMPL2:     LD      HL,BUFDSK
07C8'  7E           COMPL1:     LD      A,(HL)
07C9'  2F                       CPL
07CA'  77                       LD      (HL),A
07CB'  23                       INC     HL
07CC'  10 FA                    DJNZ    COMPL1
07CE'  0D                       DEC     C
07CF'  20 F7                    JR      NZ,COMPL1
07D1'  E1                       POP     HL
07D2'  C1                       POP     BC
07D3'  C9                       RET

07D4'              +ERROR:      PRINTL  '$'             ; Envia mensagem em caso de erro
07DA'  3A 4068                  LD      A,(HSTOP)
07DD'  B7                       OR      A
07DE'  20 09                    JR      NZ,ERR1         ; Leitura ou escrita ?
```

```
                        +               PRINT   'WRITE'
07E7'   18 06                           JR      ERR
07E9'                   +ERR1:          PRINT   'READ'
07EF'                   +ERR:           PRINT   ' error'
07F7'   DB 18                           IN      A,(DSKSTT)      ; Estado do controlador
07F9'   CB 7F                           BIT     7,A
07FB'   28 0F                           JR      Z,ERR2
                        +               PRINT   ', not ready.'
080B'   C9                              RET
080C'   CB 57           ERR2:           BIT     2,A             ; Tipo do erro
080E'   28 0D                           JR      Z,ERR3
                        +               PRINT   ', lost data'
081D'   CB 5F           ERR3:           BIT     3,A
081F'   28 0D                           JR      Z,ERR4
                        +               PRINT   ', crc error'
082E'   CB 67           ERR4:           BIT     4,A
0830'   28 14                           JR      Z,ERR5
                        +               PRINT   ', record not found'
0846'   CB 6F           ERR5:           BIT     5,A
0848'   28 0F                           JR      Z,ERR6
                        +               PRINT   ', record type'
0859'   CB 77           ERR6:           BIT     6,A
085B'   28 11                           JR      Z,ERRF
                        +               PRINT   ', write protect'
086E'                   +ERRF:          PRINT   ' on drive '
087A'   3A 4060                         LD      A,(HSTDSK)      ; Disco
087D'   C6 41                           ADD     A,'A'
087F'   4F                              LD      C,A
                        +               OUTPUT
                        +               PRTMSG
0882'   2E 0D 0A                       DEFB    '.',CR,LF,'Error on head ',NULL
0885'   45 72 72
0888'   6F 72 20
088B'   6F 6E 20
088E'   68 65 61
0891'   64 20 00
0894'   FD CB 00                       BIT     6,(IY+TYP)      ; Cabeca
0897'   76
0898'   0E 30                           LD      C,'0'
089A'   28 01                           JR      Z,ERRF1
089C'   0C                              INC     C
089D'                   +ERRF1:         OUTPUT
                        +               PRINT   ', track '      ; Trilha em decimal
08A8'   FD 7E 01                        LD      A,(IY+TRK)
08AB'   CD 00D2'        ERRF2:          CALL    PDEC
                        +               PRINT   ' ('
08B2'   FD 7E 01                        LD      A,(IY+TRK)      ; Trilha em hexadecimal
08B5'   CD 00EF'                        CALL    OUT_A
                        +               PRINT   'h) and sector '
08C8'   3A 406E                         LD      A,(AUX1)        ; Setor
08CB'   CD 00D2'                        CALL    PDEC
                        +               PRINT   ' ('
08D2'   3A 406E                         LD      A,(AUX1)
08D5'   CD 00EF'                        CALL    OUT_A
                        +               PRINT   'h).'
08DD'   C9                              RET

08DE'                   +READID:        ACTDSK          ; Le a trilha corrente do acionador
08DF'   CD 00C3'                        CALL    DSKOK
08E2'   C0                              RET     NZ
08E3'   CD 00B8'                        CALL    DSKRDY
08E6'   C5                              PUSH    BC
```

```
08E7' D5                        PUSH    DE
08E8' E5                        PUSH    HL
08E9' 16 03                     LD      D,3
08EB' 01 061B       RID:        LD      BC,600H+DSKDAT
08EE' 21 4074                   LD      HL,BUFFID       ; Buffer temporario
08F1' 3E C0                     LD      A,CMADDR
08F3' D3 18                     OUT     (DSKCMD),A
08F5' 76            RID1:       HALT                    ; Transfere 6 bytes
08F6' ED A2                     INI
08F8' 20 FB                     JR      NZ,RID1

08FA' CD 00A1'                  CALL    CHKTOUT
08FD' 20 0A                     JR      NZ,RID2
08FF' CD 00B8'      RID3:       CALL    DSKRDY          ; Verifica controlador
0902' E6 9B                     AND     09BH
0904' 28 03                     JR      Z,RID2
0906' 15                        DEC     D
0907' 20 E2                     JR      NZ,RID
0909' F5            RID2:       PUSH    AF
                  +             WAITDSK
090B' F1                        POP     AF
090C' E1                        POP     HL
090D' D1                        POP     DE
090E' C1                        POP     BC
090F' B7                        OR      A
0910' C9                        RET
                    ;
                    ;
0911' 79            SELDSK:     LD      A,C             ; Disco a ser selecionado
0912' B7                        OR      A
0913' 20 05                     JR      NZ,SELCH6
0915' 3A 406A                   LD      A,(MASTER)      ; Master ?
0918' 18 07                     JR      SELCH1
091A' 21 406A       SELCH6:     LD      HL,MASTER
091D' BE                        CP      (HL)
091E' 20 01                     JR      NZ,SELCH1
0920' AF                        XOR     A
0921' 32 406C       SELCH1:     LD      (AUX),A
0924' 4F                        LD      C,A
0925' FE 04                     CP      4
0927' 28 52                     JR      Z,SELWIN        ; Winchester
0929' FE 05                     CP      5
092B' 28 5A                     JR      Z,SELVIR        ; Disco virtual
092D' CD 05F3'                  CALL    SETIYA
                  +             WAITDSK
0931' CB 43                     BIT     0,E
0933' 20 26                     JR      NZ,SELDS1

0935' CB FF                     SET     7,A
0937' D3 14                     OUT     (DSKTYP),A      ; Ativa o acionador
0939' CD 00B8'                  CALL    DSKRDY          ; Verifica se Ok
093C' CD 00C3'                  CALL    DSKOK
093F' 28 05                     JR      Z,SELD1
                  +             WAITDSK
0942' 3E FF                     LD      A,-1            ; Erro
0944' B7                        OR      A
0945' C9                        RET
0946' 3E 08         SELD1:      LD      A,CMHOME        ; Recalibra
0948' FD B6 08                  OR      (IY+STEP)
094B' 4F                        LD      C,A
094C' CD 00B2'                  CALL    DSKCOM
094F' EE 04                     XOR     4
```

```
0951'  E6 84                    AND     10000100B
                          +     WAITDSK
0954'  3E FF                    LD      A,-1
0956'  C0                       RET     NZ
0957'  AF                       XOR     A
0958'  FD 77 01                 LD      (IY+TRK),A
095B'  3A 4051          SELDS1: LD      A,(SEKDSK)
095E'  FE 04                    CP      4
0960'  20 0A                    JR      NZ,SELD0
0962'  3A 406C                  LD      A,(AUX)
0965'  FE 04                    CP      4
0967'  C4 0B02'                 CALL    NZ,SWAP
096A'  18 08                    JR      SELD01
096C'  3A 406C          SELD0:  LD      A,(AUX)
096F'  FE 04                    CP      4
0971'  CC 0B02'                 CALL    Z,SWAP
0974'  3A 406C          SELD01: LD      A,(AUX)
0977'  32 4051                  LD      (SEKDSK),A
097A'  C9                       RET
                        ;
097B'  CB 43            SELWIN: BIT     0,E             ; Seleciona o disco rigido
097D'  20 DC                    JR      NZ,SELDS1
097F'  CD 0A50'                 CALL    SELWNC
0982'  28 D7                    JR      Z,SELDS1
0984'  3E FF                    LD      A,-1
0986'  C9                       RET
                        ;
0987'  CB 43            SELVIR: BIT     0,E             ; Seleciona o disco virtual
0989'  20 D0                    JR      NZ,SELDS1
098B'  DB C3                    IN      A,(DSKCLR)
098D'  DB C0                    IN      A,(DSKRD)
098F'  47                       LD      B,A
0990'  DB C3                    IN      A,(DSKCLR)
0992'  AF                       XOR     A
0993'  D3 C0                    OUT     (DSKWRT),A
0995'  DB C3                    IN      A,(DSKCLR)
0997'  DB C0                    IN      A,(DSKRD)
0999'  B7                       OR      A
099A'  3E FF                    LD      A,-1
099C'  C0                       RET     NZ
099D'  DB C3                    IN      A,(DSKCLR)
099F'  78                       LD      A,B
09A0'  D3 C0                    OUT     (DSKWRT),A
09A2'  18 B7                    JR      SELDS1
                        ;

                        ;
09A4'  AF               DSKIN:  XOR     A               ; Inicializa as variaveis do EDOS
09A5'  32 406B                  LD      (SKIP),A
09A8'  32 406A                  LD      (MASTER),A
09AB'  32 4051                  LD      (SEKDSK),A

09AE'  21 0B1F'                 LD      HL,FDDD5        ; Assume disco A 5" 1/4
09B1'  11 4000                  LD      DE,TABDSK       ; densidade dupla
09B4'  01 0010                  LD      BC,NUMPAR
09B7'  ED B0                    LDIR
09B9'  21 4000                  LD      HL,TABDSK
09BC'  11 4010                  LD      DE,TABDSK+NUMPAR
09BF'  01 0030                  LD      BC,NUMPAR*3
09C2'  ED B0                    LDIR

09C4'  01 0401                  LD      BC,0401H
```

```
09C7'  11 0010                 LD      DE,NUMPAR
09CA'  21 4000                 LD      HL,TABDSK
09CD'  7E           DSKIN1:    LD      A,(HL)
09CE'  B1                      OR      C
09CF'  77                      LD      (HL),A
09D0'  CB 01                   RLC     C
09D2'  19                      ADD     HL,DE
09D3'  10 F8                   DJNZ    DSKIN1

09D5'  21 0B2F'                LD      HL,WINTAB       ; tabela do winchester
09D8'  11 4040                 LD      DE,TABDSK+NUMPAR$4
09DB'  01 0010                 LD      BC,NUMPAR
09DE'  ED B0                   LDIR

09E0'  DB 16                   IN      A,(BOOT)        ; Winchester existe ?
09E2'  E6 80                   AND     80H
09E4'  C0                      RET     NZ              ; Nao, retorna

09E5'  21 0A44'    SETPARWIN:  LD      HL,PRGWNC       ; Sim. programa a interface entre
09E8'  CD 0054'                CALL    PRGPT           ; o micro e o controlador

09EB'  DB 80                   IN      A,(80H)

                   +           PRINTL  'Winchester ... '

0A00'  CD 0A50'    SPLP:       CALL    SELWNC          ; Espera o winchester atingir a
0A03'  20 FB                   JR      NZ,SPLP         ; velocidade correta

                   +           PRINT   'Ready'

0A0C'  CD 0A63'                CALL    SELW
0A0F'  3E F3                   LD      A,0F3H          ; Programa o controlador
0A11'  CD 0A72'                CALL    OUTWIN
0A14'  06 05                   LD      B,5
0A16'  3E FF                   LD      A,-1
0A18'  CD 0A72'    SETPA1:     CALL    OUTWIN
0A1B'  10 FB                   DJNZ    SETPA1
0A1D'  21 0B3F'                LD      HL,WINPAR
0A20'  06 08                   LD      B,8
0A22'  7E          SETPA2:     LD      A,(HL)
0A23'  2F                      CPL
0A24'  CD 0A72'                CALL    OUTWIN
0A27'  23                      INC     HL
0A28'  10 F8                   DJNZ    SETPA2
                   ;
0A2A'  CD 0A7C'    STATWIN:    CALL    INPWIN          ; Estado do controlador do disco
0A2D'  4F                      LD      C,A             ; rigido
0A2E'  DB 81       STTWIN:     IN      A,(81H)
0A30'  E6 22                   AND     22H
0A32'  20 FA                   JR      NZ,STTWIN
0A34'  DB 80                   IN      A,(80H)
0A36'  A1                      AND     C
0A37'  F5                      PUSH    AF
0A38'  DB 81       STTWC1:     IN      A,(81H)
0A3A'  E6 08                   AND     8
0A3C'  20 FA                   JR      NZ,STTWC1
0A3E'  F1                      POP     AF
0A3F'  3C                      INC     A
0A40'  C8                      RET     Z
0A41'  3E FF                   LD      A,-1
0A43'  C9                      RET
```

```
0A44' 01 82 8F     PRGWNC:   DEFB   1,082H,08FH          ; Tabela para a interface entre
0A47' 02 83 CF               DEFB   2,083H,0CFH,03FH     ; micro e controlador disco
0A4A' 3F
0A4B' 02 81 00               DEFB   2,081H,000H,0FFH     ; rigido
0A4E' FF
0A4F' 00                     DEFB   0

0A50' CD 0A63'     SELWNC:   CALL   SELW          ; Seleciona o controlador do Winchester
0A53' 3E FF                  LD     A,-1
0A55' C0                     RET    NZ
0A56' 06 06                  LD     B,6
0A58' CD 0A72'     SELWI1:   CALL   OUTWIN
0A5B' 10 FB                  DJNZ   SELWI1

0A5D' 18 CB                  JR     STATWIN


0A5F' 3E 7F        RESWIN:   LD     A,7FH          ; Reset controlador
0A61' 18 02                  JR     SELRES
                   ;
0A63' 3E 80        SELW:     LD     A,80H                ; RET (0) = OK
                   ;
0A65' D3 81        SELRES:   OUT    (81H),A
0A67' 3E FF                  LD     A,0FFH
0A69' D3 81                  OUT    (81H),A
0A6B' DB 81                  IN     A,(81H)
0A6D' E6 34                  AND    034H
0A6F' EE 34                  XOR    034H
0A71' C9                     RET
                   ;
0A72' 4F           OUTWIN:   LD     C,A            ; Envia ao controlador
0A73' DB 81        OUTWC1:   IN     A,(81H)
0A75' 0F                     RRCA
0A76' 38 FB                  JR     C,OUTWC1
0A78' 79                     LD     A,C
0A79' D3 80                  OUT    (80H),A
0A7B' C9                     RET
                   ;
0A7C' DB 81        INPWIN:   IN     A,(81H)        ; Le do controlador
0A7E' E6 02                  AND    2
0A80' 20 FA                  JR     NZ,INPWIN
0A82' DB 80                  IN     A,(80H)
0A84' C9                     RET
                   ;
0A85' 2A 4061      WCALC:    LD     HL,(HSTTRK)    ; Calcula a posicao do setor
0A88' 54                     LD     D,H
0A89' 5D                     LD     E,L
0A8A' 29                     ADD    HL,HL          ;X2
0A8B' 29                     ADD    HL,HL          ;X4
0A8C' 29                     ADD    HL,HL          ;X8
0A8D' 29                     ADD    HL,HL          ;X16
0A8E' 19                     ADD    HL,DE          ;X17
0A8F' 16 00                  LD     D,0
0A91' 3A 4063                LD     A,(HSTSEC)
0A94' 5F                     LD     E,A
0A95' 19                     ADD    HL,DE          ; TRILHA*17+SETOR
0A96' C9                     RET
                   ;
                   ;
0A97' 3E 0A        RDWTWC:   LD     A,10           ; Le do winchester
0A99' 32 4064      RWWINN:   LD     (HSTTRY),A     ; Numero de tentativas
0A9C' CD 0A63'               CALL   SELW
```

```
0A9F'  20 56                    JR      NZ,RWWINERR
0AA1'  06 F7                    LD      B,0F7H          ; Comando ce leitura
0AA3'  3A 4068                  LD      A,(HSTOP)
0AA6'  B7                       OR      A
0AA7'  20 02                    JR      NZ,RWWIN
0AA9'  06 F5                    LD      B,0F5H          ; Comando de escrita
0AAB'  78               RWWIN:  LD      A,B
0AAC'  CD 0A72'                 CALL    OUTWIN
0AAF'  3E FF                    LD      A,-1
0AB1'  CD 0A72'                 CALL    OUTWIN          ; Envia ao controlador
0AB4'  CD 0AB5'                 CALL    WCALC
0AB7'  7C                       LD      A,H
0AB8'  2F                       CPL
0AB9'  CD 0A72'                 CALL    OUTWIN
0ABC'  7D                       LD      A,L
0ABD'  2F                       CPL
0ABE'  CD 0A72'                 CALL    OUTWIN
0AC1'  3E FE                    LD      A,0FEH
0AC3'  CD 0A72'                 CALL    OUTWIN
0AC6'  3C                       INC     A
0AC7'  CD 0A72'                 CALL    OUTWIN
0ACA'  21 407C                  LD      HL,HSTBUF
0ACD'  06 00                    LD      B,0
0ACF'  3A 4068                  LD      A,(HSTOP)
0AD2'  B7                       OR      A
0AD3'  20 10                    JR      NZ,RWWIN1
0AD5'  7E               RWWIN2: LD      A,(HL)          ; Envia os dados
0AD6'  CD 0A72'                 CALL    OUTWIN
0AD9'  23                       INC     HL
0ADA'  10 F9                    DJNZ    RWWIN2
0ADC'  7E               RWWIN3: LD      A,(HL)
0ADD'  CD 0A72'                 CALL    OUTWIN
0AE0'  23                       INC     HL
0AE1'  10 F9                    DJNZ    RWWIN3
0AE3'  18 0E                    JR      RWWIN4
0AE5'  CD 0A7C'         RWWIN1: CALL    INPWIN          ; Le os dados
0AE8'  77                       LD      (HL),A
0AE9'  23                       INC     HL
0AEA'  10 F9                    DJNZ    RWWIN1
0AEC'  CD 0A7C'         RWWIN5: CALL    INPWIN
0AEF'  77                       LD      (HL),A
0AF0'  23                       INC     HL
0AF1'  10 F9                    DJNZ    RWWIN5
0AF3'  CD 0A2A'         RWWIN4: CALL    STATWIN         ; Verifica o estado do controlador
0AF6'  C8                       RET     Z
0AF7'  3A 4064          RWWINERR: LD    A,(HSTTRY)
0AFA'  3D                       DEC     A               ; Erro
0AFB'  20 9C                    JR      NZ,RWWINN
0AFD'  3D                       DEC     A
0AFE'  32 4065                  LD      (ERFLAG),A
0B01'  C9                       RET
                        ;
                        ;
FD75                    ALVSWAP EQU     0FD75H

0B02'  3A 406B          SWAP:   LD      A,(SKIP)        ; Troca as tabelas do ALV e CVS
0B05'  B7                       OR      A               ; do BIOS com a do EDOS WINCHESTER
0B06'  C8                       RET     Z
0B07'  01 028A                  LD      BC,650
0B0A'  11 FD75                  LD      DE,ALVSWAP
0B0D'  21 427C                  LD      HL,HSTSWAP
0B10'  7E               SWAPL:  LD      A,(HL)
```

```
0B11'  F5                      PUSH    AF
0B12'  1A                      LD      A,(DE)
0B13'  77                      LD      (HL),A
0B14'  F1                      POP     AF
0B15'  12                      LD      (DE),A
0B16'  13                      INC     DE
0B17'  23                      INC     HL
0B18'  0D                      DEC     C
0B19'  20 F5                   JR      NZ,SWAPL
0B1B'  05                      DEC     B
0B1C'  20 F2                   JR      NZ,SWAPL
0B1E'  C9                      RET


0010                NUMPAR     EQU     16      ; Total de parametros


0B1F'  00 00 28     FDDD5:     DEFB    0,0,40,8,00000110B,16,4,64,0    ; 5" 1/4 Dupla Dens.
0B22'  08 06 10
0B25'  04 40 00
0B28'  00 00 00                DEFB    0,0,0,0,0,0,0
0B2B'  00 00 00
0B2E'  00

0B2F'  00 00 00     WINTAB:    DEFB    0,0,0,0,00000010B,16,4,64,0     ; Winchester
0B32'  00 02 10
0B35'  04 40 00
0B38'  00 00 00                DEFB    0,0,0,0,0,0,0
0B3B'  00 00 00
0B3E'  00

0B3F'  01 32 04     WINPAR:    DEFB    1,32H,4,0,80H,0,40H,0BH         ; Progr. Controlador
0B42'  00 80 00
0B45'  40 0B


                    ;###########################################################
                    ;
                    ;
0B47'  45 64 6F             DEFM    'Edos - V1.2 - 22/03/89 '       ; Ultima Alteracao
0B4A'  73 20 2D
0B4D'  20 56 31
0B50'  2E 32 20
0B53'  2D 20 32
0B56'  32 2F 30
0B59'  33 2F 38
0B5C'  39 20
0B5E'  4D 61 74             DEFM    'Mateus J. Martins'
0B61'  65 75 73
0B64'  20 4A 2E
0B67'  20 4D 61
0B6A'  72 74 69
0B6D'  6E 73
                    ;
                    ;
                    ;###########################################################
                    ;
                    ;
                            .PHASE  4000H           ; RAM estatica

                    ;       Variaveis do EDOS
```

```
4800                    PILHA           EQU     $+800H          ; "Stack Point"

4000                    TABDSK:         DEFS    5 * 16          ; 4 unidades de disco

4050                    TYPCOR:         DEFS    1
4051                    SEKDSK:         DEFS    1
4052                    SEKTRK:         DEFS    2
4054                    SEKSEC:         DEFS    1
4055                    DMAADR:         DEFS    2
4057                    HSTACT:         DEFS    1
4058                    HSTWRT:         DEFS    1
4059                    UNACNT:         DEFS    1
405A                    RTRACK:         DEFS    1
                        ;
405B                    SEKHST:         DEFS    1
                        ;
405C                    UNADSK:         DEFS    1
405D                    UNATRK:         DEFS    2
405F                    UNASEC:         DEFS    1
                        ;
4060                    HSTDSK:         DEFS    1
4061                    HSTTRK:         DEFS    2
4063                    HSTSEC:         DEFS    1
4064                    HSTTRY:         DEFS    1
                        ;
4065                    ERFLAG:         DEFS    1
4066                    RSFLAG:         DEFS    1
4067                    READOP:         DEFS    1
4068                    HSTOP:          DEFS    1
4069                    WRTYPE:         DEFS    1
406A                    MASTER:         DEFS    1
                        ;
406B                    SKIP:           DEFS    1
406C                    AUX:            DEFS    2
406E                    AUX1:           DEFS    2
4070                    USRHL:          DEFS    2
4072                    USRSP:          DEFS    2
4074                    BUFFID:         DEFS    8

407C                    HSTBUF:
407C                    BUFDSK:         DEFS    512             ; "Buffer" de setor

427C                    HSTSWAP:        DEFS    650             ; Area reservada a ALV do Winchester


4506                    DATAEND         EQU     $

                        ;       Fim do EDOS

                                        END
```

Macros:

ACTDSK INPUT OUTPUT PRINT PRINTL
PRTMSG WAITDSK

Symbols:

04BC'	ALLOC	FD75	ALVSWAP	406C	AUX
406E	AUX1	0005	BCKCNT	0007	BELL
0016	BOOT	407C	BUFDSK	4074	BUFFID
0017	C8255	0728'	CHECK	0314'	CHGDSK
073A'	CHKSTP	00A1'	CHKTOUT	0480'	CHKUNA
0018'	CINP	00D0	CMABORT	00C0	CMADDR
0008	CMHOME	0229'	CMPER	021E'	CMPLP
0080	CMREAD	0018	CMSEEK	004C	CMSTPI
006C	CMSTPO	00A0	CMWRITE	021A'	COMPAR
07C8'	COMPL1	07C5'	COMPL2	07AF'	COMPLT
022E'	COPSYS	0048'	COUT	0049'	COUT1
FA00	CPMBT	0007	CPMSPT	000D	CR
0020'	CSTAT	000C	D8253	4506	DATAEND
E400	DEFCPM	02D0'	DISCO	4055	DMAADR
00C3	DSKCLR	0018	DSKCMD	00B2'	DSKCOM
001B	DSKDAT	09A4'	DSKIN	09CD'	DSKIN1
00C3'	DSKOK	00C6'	DSKOK1	00C0	DSKRD
00B8'	DSKRDY	001A	DSKSEC	0018	DSKSTT
0019	DSKTRK	0014	DSKTYP	00C0	DSKWRT
4065	ERFLAG	07EF'	ERR	07E9'	ERR1
080C'	ERR2	081D'	ERR3	082E'	ERR4
0846'	ERR5	0859'	ERR6	086E'	ERRF
089D'	ERRF1	08AB'	ERRF2	03C8'	ERRMSG
07D4'	ERROR	05BE'	FATERR	05C7'	FATO
081F'	FDDD5	020B'	FILCMP	052E'	FILHS1
0507'	FILHST	0567'	FLUSH	05E7'	FTSKP
4057	HSTACT	407C	HSTBUF	4060	HSTDSK
4068	HSTOP	4063	HSTSEC	427C	HSTSWAP
4061	HSTTRK	4064	HSTTRY	4058	HSTWRT
0A7C'	INPWIN	0005	INVBNC	0249'	LDDSK
000A	LF	0015	LST	0016	LSTSTT
406A	MASTER	0532'	MATCH	0385'	MSGINV
0351'	MSGOFF	031C'	MSGOUT	03AD'	MSGWBR
0588'	NEWRW	0734'	NOCHE	05ED'	NOERR
0500'	NOMATC	04B6'	NOOVF	015A'	NOSYST
0004	NRMBNC	0000	NULL	0010	NUMPAR
0296'	OPCA1	0283'	OPCAO	00F8'	OUTNB
0100'	OUTNB1	0A73'	OUTWC1	0A72'	OUTWIN
00EF'	OUT_A	00EA'	OUT_HL	0004	PARDSK
00D5'	PDO	00E4'	PD1	00D2'	PDEC
0441'	PEEK	0448'	PEEK1	0068'	PERIF
4800	PILHA	003B'	PMSG	003B'	PMSG1
0044'	PMSG2	042F'	POKE	0436'	POKE1
0054'	PRGPT	0A44'	PRGWNC	0006	RAM
076C'	RDD	075A'	RDD1	0760'	RDD2
0771'	RDD3	0768'	RDDSK	0744'	RDRDSK
0576'	RDRHST	0578'	RDRWRT	0A97'	RDWTWC
00BB'	RDY1	0453'	READ	08DE'	READID
4067	READOP	0A5F'	RESWIN	041A'	RETDP1
0414'	RETDPB	030C'	RETURN	08EB'	RID
08F5'	RID1	0909'	RID2	08FF'	RID3
0004	ROM	4066	RSFLAG	405A	RTRACK
056E'	RWEND	0774'	RWHEND	0590'	RWHST
05A8'	RWHST1	05AB'	RWHST2	0558'	RWMOVE
04D6'	RWOP1	04CE'	RWOP2	04C4'	RWOPER
0AA8'	RWWIN	0AE5'	RWWIN1	0AD5'	RWWIN2
0ADC'	RWWIN3	0AF3'	RWWIN4	0AEC'	RWWIN5

0AF7'	RWWINERR	0A99'	RWWINN	000F	S8253
0428'	SAVDP1	0421'	SAVDPB	0006	SECMSK
0006	SECSHF	0003	SECTOP	0656'	SEEKS1
0647'	SEEKSC	061B'	SEEKT1	0624'	SEEKT2
0645'	SEEKT3	0603'	SEEKTR	4051	SEKDSK
405B	SEKHST	4054	SEKSEC	4052	SEKTRK
0921'	SELCH1	091A'	SELCHG	096C'	SELD0
0974'	SELD01	0946'	SELD1	095B'	SELDS1
0911'	SELDSK	0A65'	SELRES	0987'	SELVIR
0A63'	SELW	0A58'	SELWI1	097B'	SELWIN
0A50'	SELWNC	05F0'	SETIY	05F3'	SETIYA
0A18'	SETPA1	0A22'	SETPA2	09E5'	SETPARWIN
065F'	SHIFT	0669'	SHIFT1	02A6'	SIGLOP
02B9'	SIGLP1	0010	SIOAD	0011	SIOAS
0012	SIOBD	0013	SIOBS	0003	SIZEID
406B	SKIP	0A00'	SPLP	0105'	START
0A2A'	STATWIN	0008	STEP	0001	STPRT
0139'	STRT1	0146'	STRT2	0A38'	STTWC1
0A2E'	STTWIN	0B02'	SWAP	0B10'	SWAPL
06A8'	TAB1	06C2'	TAB2	06CC'	TAB3
06D6'	TAB4	06E0'	TAB5	4000	TABDSK
0016	TIMEOUT	068E'	TRA3	069A'	TRA4
0009	TRAN	06FE'	TRAN1	0714'	TRAN2
0725'	TRAN3	0705'	TRAN4	0716'	TRAN5
06F0'	TRANSF	0679'	TRAS1	0680'	TRAS2
066F'	TRASEC	069E'	TRATAB	0001	TRK
0000	TRKID	0002	TRKTOP	0196'	TSTER1
0170'	TSTERR	01E6'	TSTR1	0163'	TSTRAM
01F2'	TSTRM	0000	TYP	4050	TYPCOR
4059	UNACNT	405C	UNADSK	405F	UNASEC
405D	UNATRK	4070	USRHL	4072	USRSP
0002	VEL4M	0000	VELOC	02C6'	VRTCL
02C1'	VRTCLR	009B'	WAIT	009D'	WAIT1
0A85'	WCALC	0B3F'	WINPAR	0B2F'	WINTAB
0000	WRALL	07A7'	WRD	0001	WRDIR
07A3'	WRDSK	0004	WRFLU	045E'	WRITE
0795'	WRT1	079B'	WRT2	07AD'	WRT3
077C'	WRTDSK	0573'	WRTHST	4069	WRTYPE
0002	WRUAL				

No Fatal error(s)

PROJETO DE UM MICROCOMPUTADOR DE 8 BITS PARA APLICAÇÕES EM PESQUISA E ENSINO

Introdução:

-=- Expansão da Informática.

-=- Uso em Pesquisa (Sistemas Dedicados).

-=- Ensino (Arquitetura Simples).

-=- Micros Comerciais.

* Alto Custo.

* Manutenção (sem esquemas).

Objetivos:

-=- Construção de um microcomputador destinado a pesquisa e ensino.

-=- Custo Baixo.

-=- Facilidades de modificação (Esquemas).

-=- Fácil manutenção.

-=- Uso em sistemas Dedicados.

-=- Arquitetura Simples.

-=- Facilidade de interfaceamento.

-=- Possibilidade de Expansões.

Porque 8 bits ?

-=- UCPs 16 bits são mais complexas.

* Maior quantidade de CIs.

* Custo Maior.

* Arquitetura para primeiro contato complexa.

-=- Dispositivos de 8 bits (Impressoras, plotters, etc).

Porque o Z80 ?

-=- Facilmente disponível no mercado nacional.

-=- Custo mais baixo.

-=- Arquitetura simples, interfaces

-=- Possui recursos de "hardware" internos, diminui o numero de componentes.

-=- Conjunto de instruções maior, operações de 8 e 16 bits, "loops", transferências de dados.

-=- Sintaxe das instruções mais simples:

Z80	8080/8085
LD	MVI, MOV, LXI, LDAX, STAX, SHLD, LHLD.

Arquitetura:

-=- "Hardware" Básico e Expansões.

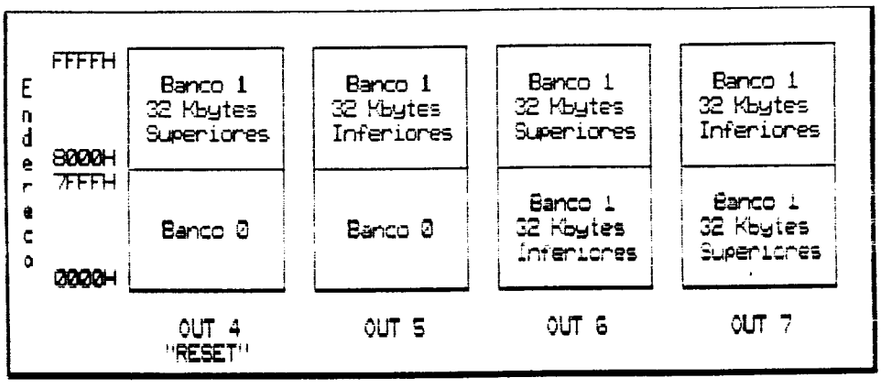
-=- Diagrama em blocos (forma mais didática).

-=- "Clock", mudança por "software" e sincrona (evitar perda de memória).

-=- "Reset", sincrono (evitar perda do conteúdo da memória).

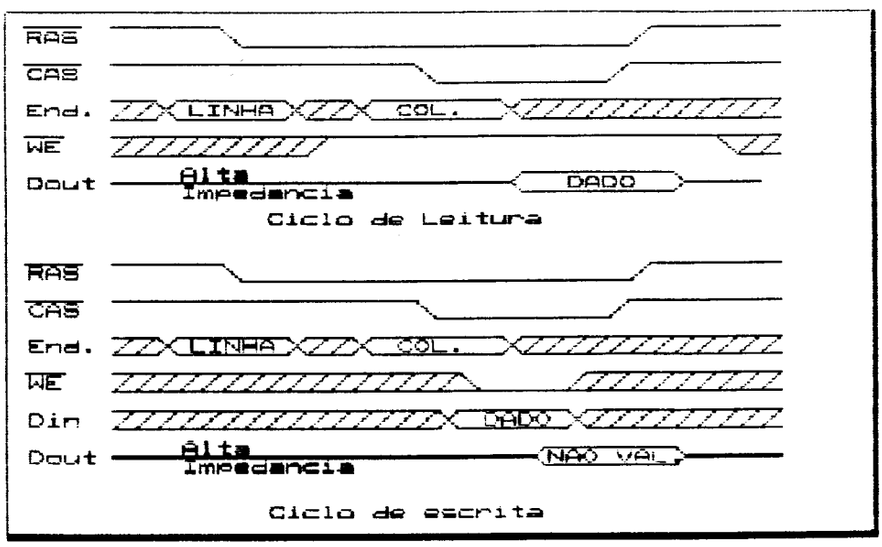
-=- Uso de Bancos de memória (mais capacidade).

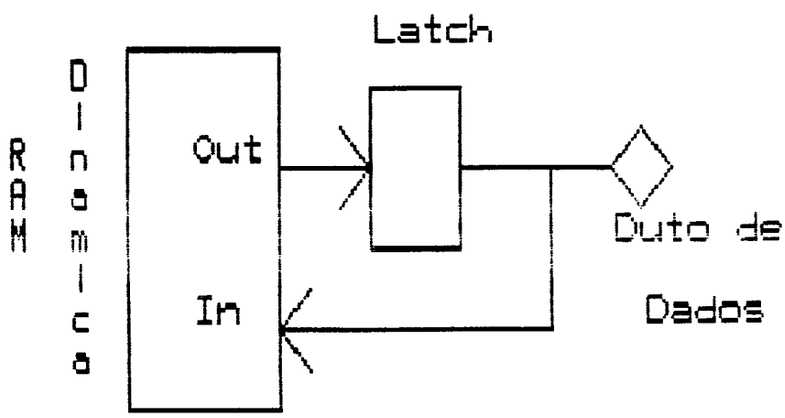
-=- Seleção de Bancos.

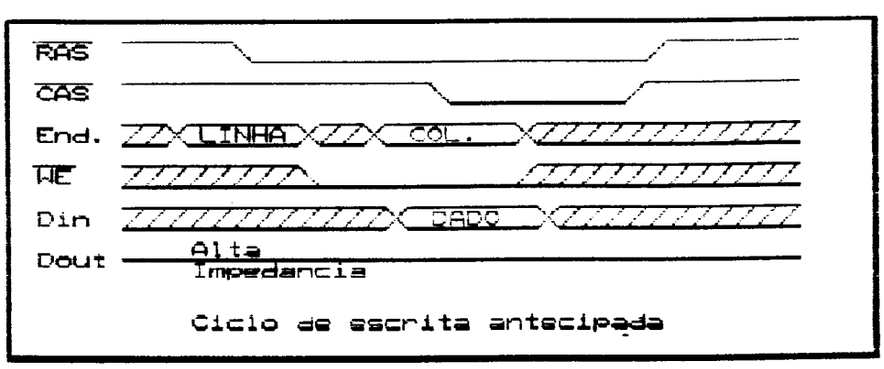


-=- Possibilidade de vários bancos.

-=- RAM dinâmica, uso de ciclos de escrita antecipada.





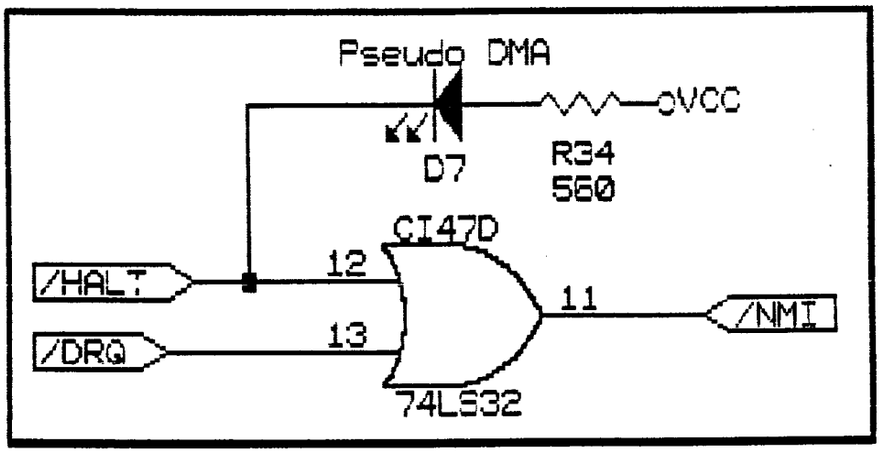


-=- Duto com "Buffers" e distinção entre dispositivos internos e externos.

-=- Uso de poucos endereços de E/S (00H a 1FH), usuário (20H a FFH).

-=- 2 linhas seriais (RS232C).

-=- Até 4 unidades de disco flexível 5"1/4 e/ou 8" (Pseudo DMA).




```
ORG NMI                    Rotina de leitura
RETN  <---------------       LP: HALT
  |--------------------->          INI
                                   JP NZ,LP
```

-=- Para 4Mhz, Taxa = 88,8 Kbytes/seg, Taxa do disco = 67 Kbytes/seg

-=- Saida paralela "Centronics" para impressora ou outro uso.

Interface para Winchester:

-=- Maior capacidade de Armazenamento.

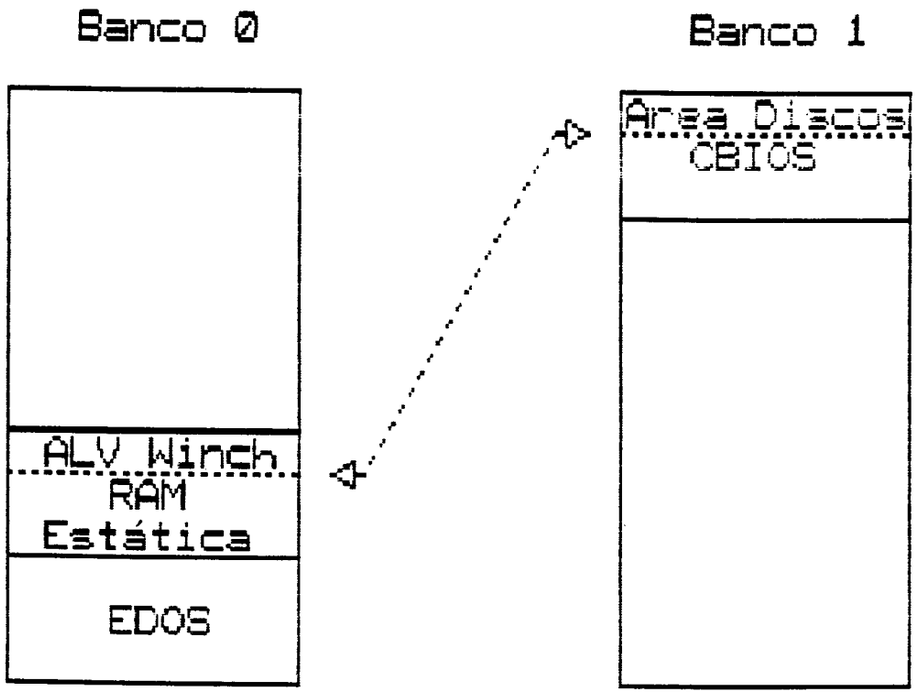
-=- Maior velocidade de acesso ao disco (Warm Boot no CP/M).

-=- Interface Simples.

-=- Problemas com a grande área de ALV.

* Winchester 10 Mbytes => ALV = 640 bytes (Bloco de 2 Kbytes).

-=- Solucionado pelo EDOS.



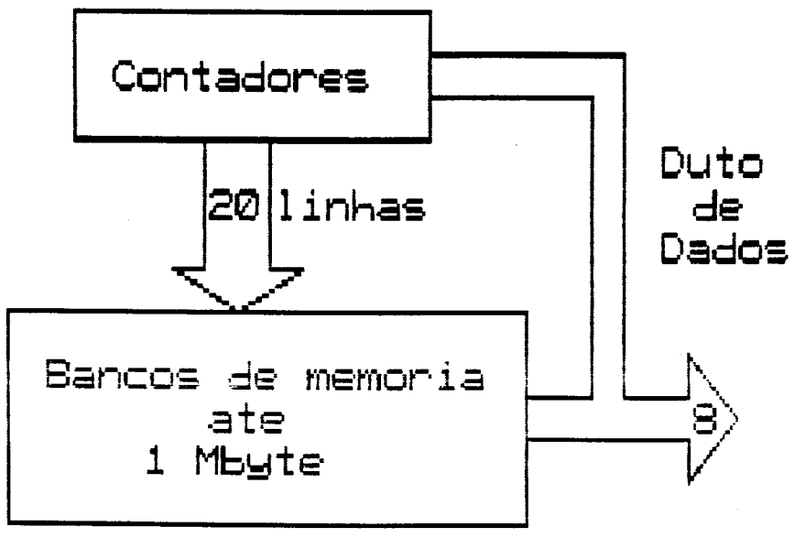
Disco Virtual:

-=- Uso de um controlador próprio (outras aplicações).

-=- Até 1 Mbyte sem modificação.

-=- Acesso através de portos de E/S.

-=- Utilização para arquivos temporários (Compilação, aquisição de dados, etc).



-=- Contadores incrementados a cada leitura ou escrita.

* Trilha [0..0F], Setor [0..FF], Lado [0..1], Tamanho = 128 bytes.

* Simples algoritimos.

"Software":

-=- Monitor, uso para controle.

-=- Uso com CP/M, sistema mais conhecido para micros de 8 bits.

Monitor:

-=- Monitorização dos recursos do micro (Registradores, memória, E/S).

-=- Utilizado para Desenvolvimento (Montadores e desmontadores em "Assembly").

-=- Testes de programas.

-=- Apóio à programação (Funções internas)

-=- Sintaxe simples, recursiva.

-=- Vários recursos.

Valor + Valor	-Valor
Valor - Valor	~Valor
Valor * Valor	@
Valor / Valor	$
Valor & Valor	'AB'
Valor ! Valor	^RG
Valor % Valor	(Endereço)
Valor.	[Parâmetro]

Exemplo:

D ^BC+512. (^HL)*3

-=- Comandos internos.

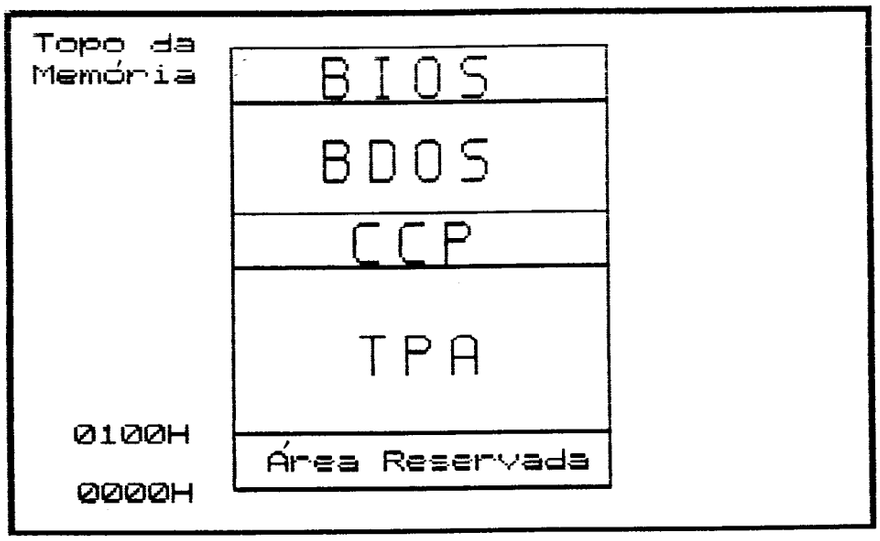
* Mostrar registradores.
* Montar e desmontar instruções.
* Rástrear programas ("Break Points", "Trace Jump", "Trace Call").
* Comandar interfaces.
* Executar Programas.
* Procurar bytes na memória.
* Preencher memória, etc.

-=- Comandos Externos, programados pelo usuário.

-=- Funções internas (42), através de uma pseudo instrução 'JSYS n'.

-=- Funções externas, programadas pelo usuário.

CP/M:



-=- CP/M para 64 Kbytes possui área reservada ao BIOS de 1.5 Kbytes.

-=- Diminuir a área de TPA.

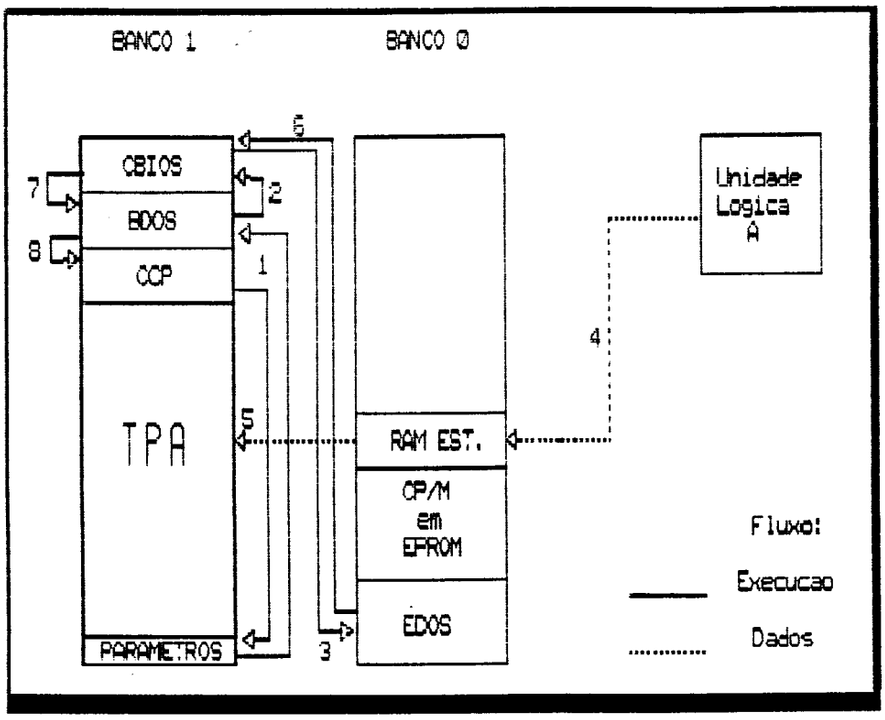
-=- BIOS utiliza um tipo de disco apenas.

-=- Possui mensagens pouco esclarecedoras.

-=- Algoritimos para "BLOCK" e "DEBLOCK" pouco eficientes.

-=- Não possui capacidade para Winchester.

-=- EDOS soluciona os problemas.



1 - Chamada ao BDOS, 2 - Setor Fisico CBIOS,
3 - Pedido ao EDOS, 4 - Transferencia disco,
5 - Transferencia CP/M, 6 - "Status ao CBIOS,
7 - "Parâmetros ao BDOS, 8- "Status" Programa

-=- Desenvolvido vários programas para apoio ao usuário e configuração do sistema, escritos em C e "Assembly".

Conclusão:

-=- Micro de baixo custo (TTL).

-=- Disponibilidade de esquemas.

-=- Facilmente modificavél.

-=- Listagens.

-=- "Layouts".

-=- Utilizado nos laboratórios de ensino.

-=- Utilizado em Laboratários de pesquisa.

PROJETO DE UM MICROCOMPUTADOR DE 8 BITS PARA APLICAÇÕES EM PESQUISA E ENSINO

Introdução:

-=- Expansão da Informática.

-=- Uso em Pesquisa (Sistemas Dedicados).

-=- Ensino (Arquitetura Simples).

-=- Micros Comerciais.

* Alto Custo.

* Manutenção (sem esquemas).

Objetivos:

-=- Construção de um microcomputador destinado a pesquisa e ensino.

-=- Custo Baixo.

-=- Facilidades de modificação (Esquemas).

-=- Fácil manutenção.

-=- Uso em sistemas Dedicados.

-=- Arquitetura Simples.

-=- Facilidade de interfaceamento.

-=- Possibilidade de Expansões.

Porque 8 bits ?

-=- UCPs 16 bits são mais complexas.

* Maior quantidade de CIs.
* Custo Maior.
* Arquitetura para primeiro contato complexa.

-=- Dispositivos de 8 bits (Impressoras, plotters, etc).

Porque o Z80 ?

-=- Facilmente disponível no mercado nacional.

-=- Custo mais baixo.

-=- Arquitetura simples, Interfaces

-=- Possui recursos de "hardware" internos, diminui o numero de componentes.

-=- Conjunto de instruções maior, operações de 8 e 16 bits, "loops", transferências de dados.

-=- Sintaxe das instruções mais simples;

Z80	8080/8085
LD	MVI, MOV, LXI, LDAX, STAX, SHLD, LHLD.

Arquitetura:

-=- "Hardware" Básico e Expansões.

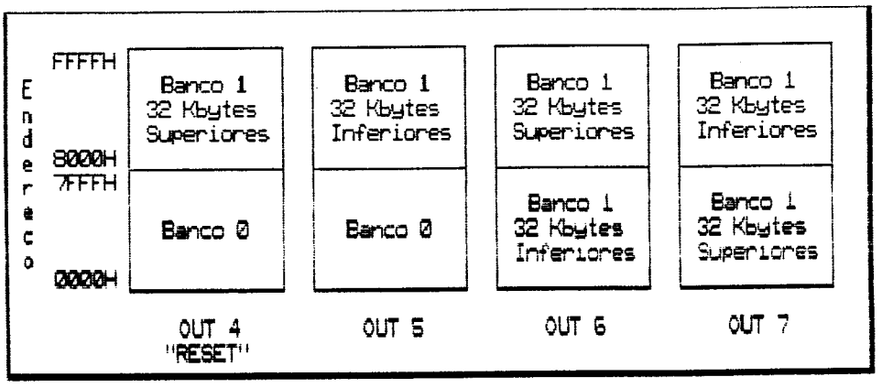
-=- Diagrama em blocos (forma mais didática).

-=- "Clock", mudança por "software" e sincrona (evitar perda de memória).

-=- "Reset", sincrono (evitar perda do conteúdo da memória).

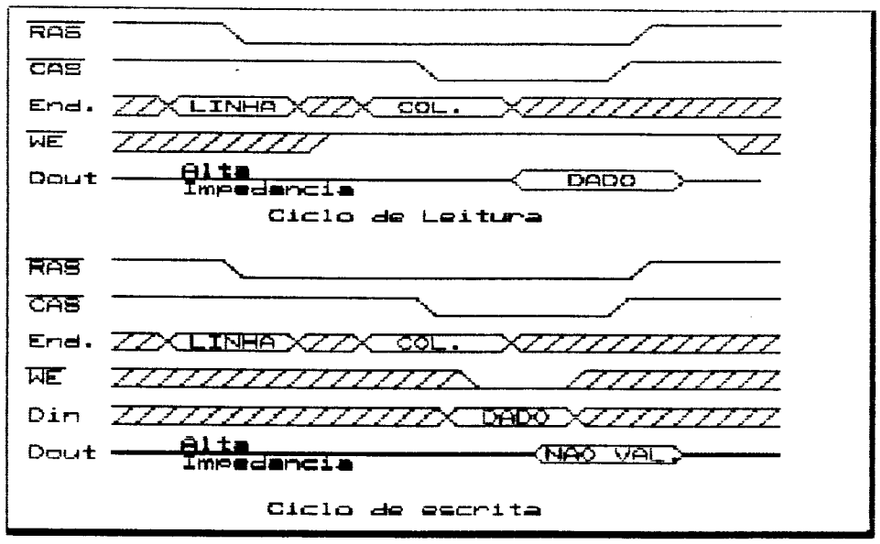
-=- Uso de Bancos de memória (mais capacidade).

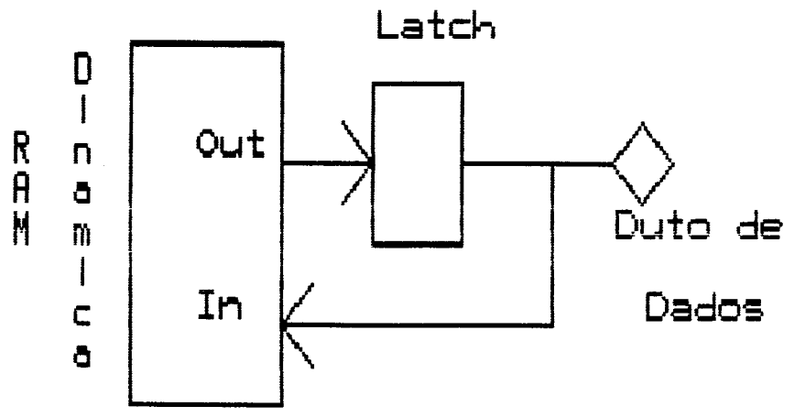
-=- Seleção de Bancos.



-=- Possibilidade de vários bancos.

-=- RAM dinâmica, uso de ciclos de escrita antecipada.



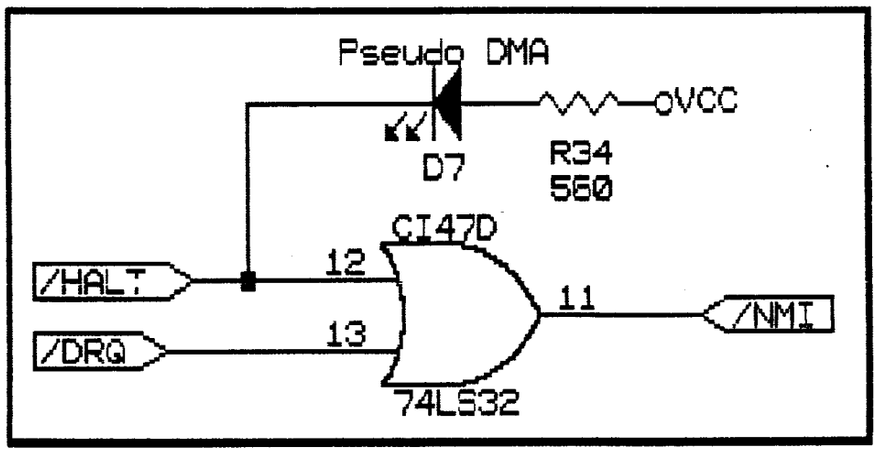


-=- Duto com "Buffers" e distinção entre dispositivos internos e externos.

-=- Uso de poucos endereços de E/S (00H a 1FH), usuário (20H a FFH).

-=- 2 linhas seriais (RS232C).

-=- Até 4 unidades de disco flexível 5"1/4 e/ou 8" (Pseudo DMA).




```
ORG NMI                    Rotina de leitura
RETN  <---------------      LP: HALT
  |-------------------->          INI
                                 JP NZ,LP
```

-=- Para 4Mhz, Taxa = 88,8 Kbytes/seg, Taxa do disco = 67 Kbytes/seg

-=- Saida paralela "Centronics" para impressora ou outro uso.

Expansões

Interface para Winchester:

-=- Maior capacidade de Armazenamento.

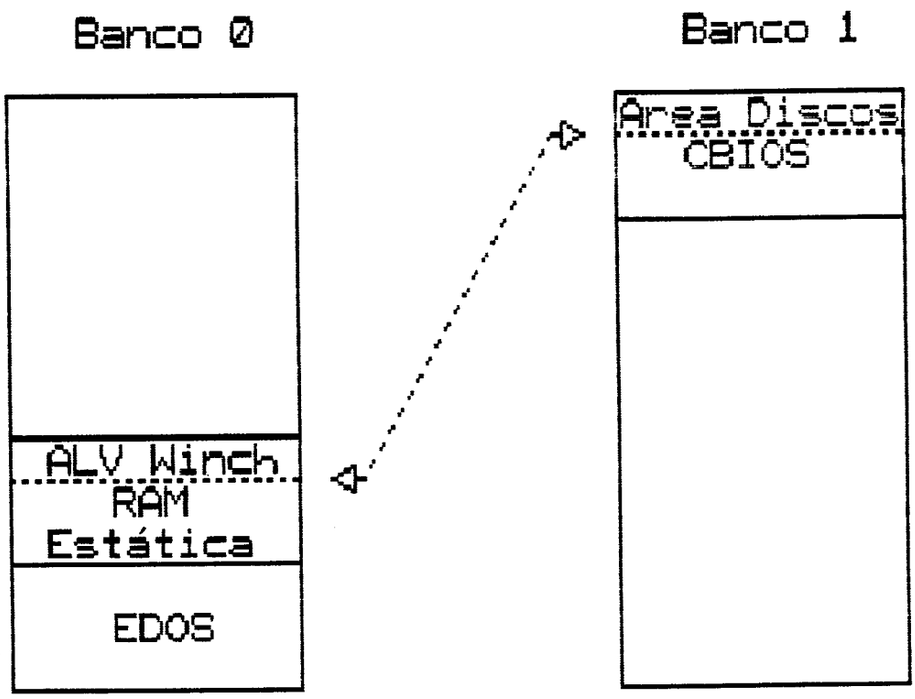
-=- Maior velocidade de acesso ao disco (Warm Boot no CP/M).

-=- Interface Simples.

-=- Problemas com a grande área de ALV.

* Winchester 10 Mbytes => ALV = 640 bytes (Bloco de 2 Kbytes).

-=- Solucionado pelo EDOS.



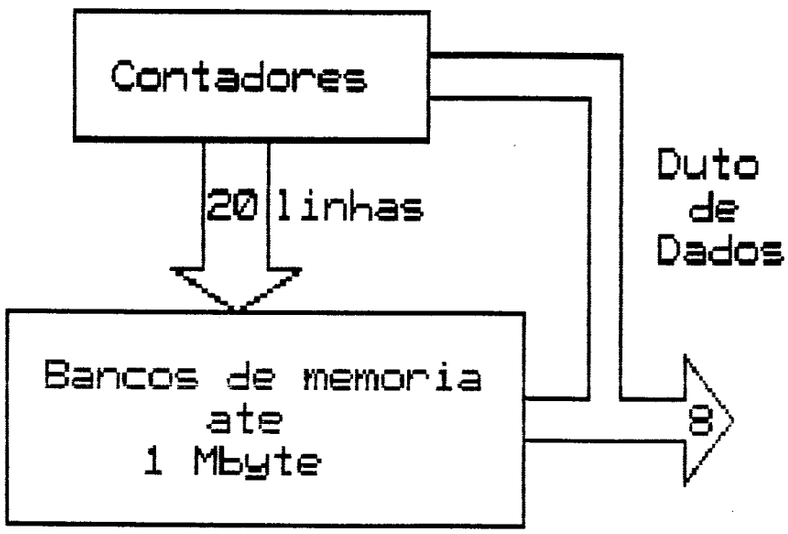
Disco Virtual:

-=- Uso de um controlador próprio (outras aplicações).

-=- Até 1 Mbyte sem modificação.

-=- Acesso através de portos de E/S.

-=- Utilização para arquivos temporários (Compilação, aquisição de dados, etc).



-=- Contadores incrementados a cada leitura ou escrita.

* Trilha [0..0F], Setor [0..FF], Lado [0..1], Tamanho = 128 bytes.
* Simples algoritimos.

"Software":

-=- Monitor, uso para controle.

-=- Uso com CP/M, sistema mais conhecido para micros de 8 bits.

Monitor:

-=- Monitorização dos recursos do micro (Registradores, memória, E/S).

-=- Utilizado para Desenvolvimento (Montadores e desmontadores em "Assembly").

-=- Testes de programas.

-=- Apóio à programação (Funções internas)

-=- Sintáxe simples, recursiva.

-=- Vários recursos.

Valor + Valor	-Valor
Valor - Valor	~Valor
Valor * Valor	@
Valor / Valor	$
Valor & Valor	'AB'
Valor ¦ Valor	^RG
Valor % Valor	(Endereço)
Valor.	[Parâmetro]

Exemplo:

D ^BC+512. (^HL)*3

-=- Comandos internos.

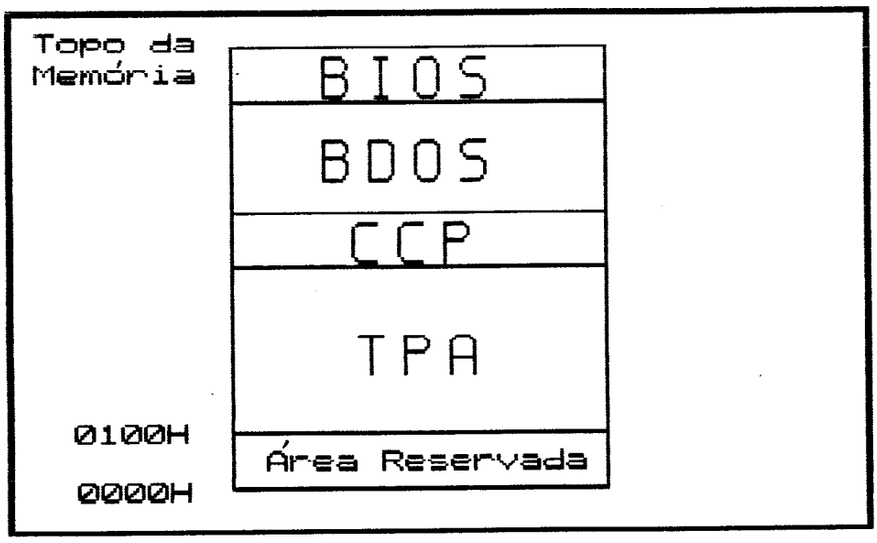
* Mostrar registradores.
* Montar e desmontar instruções.
* Rastrear programas ("Break Points", "Trace Jump", "Trace Call").
* Comandar interfaces.
* Executar Programas.
* Procurar bytes na memória.
* Preencher memória, etc.

-=- Comandos Externos, programados pelo usuário.

-=- Funções internas (42), através de uma pseudo instrução 'JSYS n'.

-=- Funções externas, programadas pelo usuário.

CP/M:



-=- CP/M para 64 Kbytes possui área reservada ao BIOS de 1.5 Kbytes.

-=- Diminuir a área de TPA.

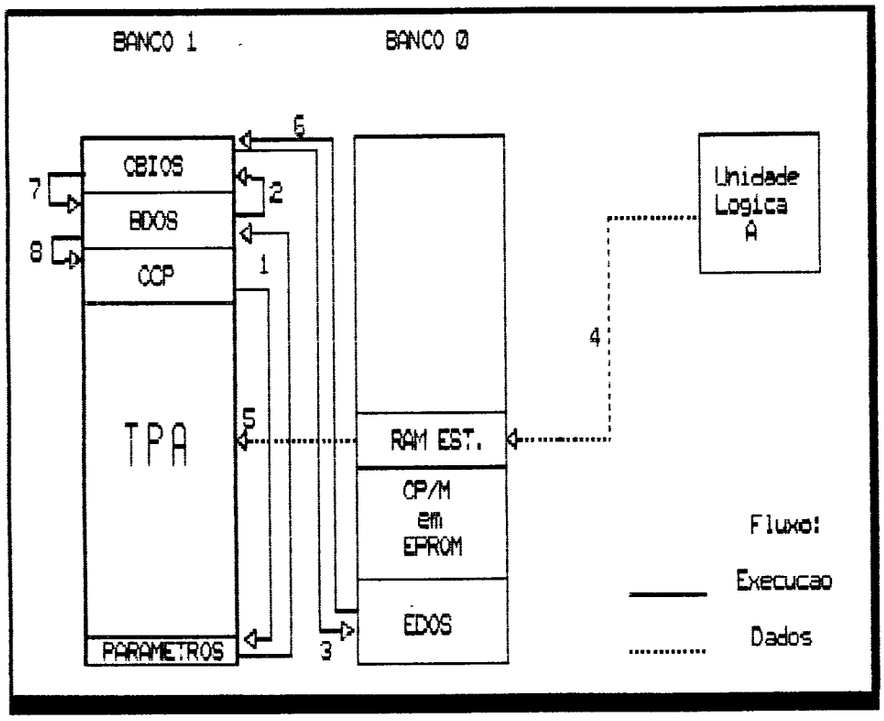
-=- BIOS utiliza um tipo de disco apenas.

-=- Possui mensagens pouco esclarecedoras.

-=- Algorítimos para "BLOCK" e "DEBLOCK" pouco eficientes.

-=- Não possui capacidade para Winchester.

-=- EDOS soluciona os problemas.



1 - Chamada ao BDOS, 2 - Setor Físico CBIOS,
3 - Pedido ao EDOS, 4 - Transferência disco,
5 - Transferencia CP/M, 6 - "Status ao CBIOS,
7 - "Parâmetros ao BDOS, 8- "Status" Programa

-=- Desenvolvido vários programas para apoio ao usuário e configuração do sistema, escritos em C e "Assembly".

Conclusão:

-=- Micro de baixo custo (TTL).

-=- Disponibilidade de esquemas.

-=- Facilmente modificavél.

-=- Listagens.

-=- "Layouts".

-=- Utilizado nos laboratórios de ensino.

-=- Utilizado em Laboratários de pesquisa.