[illegible]

I.C.M.S.C.

INSTITUTO DE CIÊNCIAS MATEMÁTICAS DE SÃO CARLOS

ESTUDO E AVALIAÇÃO DE UMA ARQUITETURA DE UM MÓDULO DE MEMÓRIA VIRTUAL PARA UM SISTEMA DISTRIBUÍDO FUNCIONAL

Regina Helena Carlucci Santana

UNIVERSIDADE DE SÃO PAULO

SÃO CARLOS - SÃO PAULO
BRASIL

ESTUDO E AVALIAÇÃO DE UMA ARQUITETURA DE UM MÓDULO DE MEMÓRIA VIRTUAL PARA UM SISTEMA DISTRIBUÍDO FUNCIONAL

Regina Helena Carlucci Santana

Orientador:
Prof. Dr. Jan F.W. Slaets

Dissertação apresentada ao Instituto de Ciências Matemáticas de São Carlos, da Universidade de São Paulo, para obtenção do título de Mestre em Ciências de Computação e Estatística

- São Carlos -
1985

COMISSÃO EXAMINADORA

Prof. Dr. Jan F.W. Slaets (IFQSC-USP)
Prof. Dr. Fernão S.R. Germano (ICMSC-USP)
Profa. Dra. Annie France Frere (EESC-USP)

Dedico este trabalho

Ao Marcos pelo incentivo, disponibilidade e compreensão em todas as etapas de nossas vias.

Ao André pela felicidade que nos trouxe.

AGRADECIMENTOS

Ao Professor Dr. Jan F.W. Slaets pela orientação.

Ao Professor Marcos José Santana pelo incentivo e valiosas sugestões.

Ao Professor Dr. Odelar Leite Linhares e Dr. Maximilian Emil Hehl pelo apoio que tem nos dado.

Ao Professor Fernão S.R. Germano pelo apoio, colaboração e incentivo em todas as nossas atividades e em especial na conclusão deste trabalho.

Aos amigos que colaboraram na realização deste trabalho, em especial a Mônica M.R. Germano que não poupou esforços para colaborar na conclusão deste trabalho.

À secretária Sueli Ap. H. Ferreira pela datilografia deste trabalho.

A Carmen L. Paqadigorria pela colaboração prestada na utilização dos equipamentos do Laboratório de Computação do ICMSC.

ESTUDO E AVALIAÇÃO DE UMA ARQUITETURA DE UM MÓDULO DE MEMÓRIA VIRTUAL PARA UM SISTEMA DISTRIBUÍDO FUNCIONAL

RESUMO

Apresenta-se neste trabalho, uma proposta de arquitetura para um módulo de memória virtual, que é compartilhado por todos os demais módulos de um sistema distribuído funcional. A arquitetura sugere algumas características para melhorar o desempenho do módulo, tais como um pequeno cache e outras implementações que visam diminuir o tempo de acesso ao módulo.

A arquitetura sugerida apresenta vantagens em relação às tradicionais, destacando-se a existência de um processador dedicado às tarefas de gerenciamento de memória, simplificando tanto o gerenciamento de memória de um modo geral, quanto a arquitetura dos demais módulos do sistema.

A arquitetura proposta é simulada com o objetivo de avaliar o desempenho global do módulo de memória e, com isto, observarem-se as influências de determinadas características nesse desempenho.

A simulação é implementada modularmente, permitindo fácil alteração nos blocos básicos da arquitetura proposta, podendo-se verificar o comportamento do módulo em função de variações estruturais, tanto a nível de software como de hardware.

Destaca-se também neste trabalho, a conceituação e considerações sobre implementação de memória virtual e cache, compiladas a partir de uma vasta bibliografia.

ARCHITECTURE STUDY AND EVALUATION OF A VIRTUAL MEMORY MODULE FOR A FUNCTIONAL DISTRIBUTED SYSTEM

ABSTRACT

A suggested architecture for a virtual memory module is presented. This module is shared by the other modules of the functional distributed system. This architecture suggests some features that improve the module performance, for example, a little cache and other implementation characteristics, that decrease the module access time.

Such architecture shows some advantages when confronted with more tradicional ones. There is a dedicated processor that makes the memory management. It improves memory management and the architecture of the other modules of the system.

This architecture is simulated to evaluate the memory module performance, so that, the influence of some module features on this performance can be observed.

The simulation has a modular implementation that makes easier changes in the basic architecture blocks, so, the module behavior can be evaluated by structural changes both in the software and in the hardware level.

A conceptuation and consideration on virtual memory and cache implementation, obtained from a vast bibliography, is shown in this work.

ÍNDICE

CAPÍTULO I - INTRODUÇÃO

1 - Considerações Gerais 01

2 - Organização da Dissertação 03

CAPÍTULO II - REVISÃO BIBLIOGRÁFICA

1 - Considerações Iniciais 04

2 - Conceito e Características de Um Sistema Distribuido. 05

3 - Memória 08

4 - Considerações Finais 46

CAPÍTULO III - CONCEITUAÇÃO DO MÓDULO

1 - Considerações Iniciais 48

2 - Características Gerais do Módulo 50

3 - Arquitetura do Módulo 57

4 - Esquema Lógico do Hardware 59

5 - Estrutura Modular do Software 62

6 - Considerações Finais 72

CAPÍTULO IV - DESCRIÇÃO DO MÓDULO

1 - Considerações Iniciais 73

2 - Memória Cache 75

3 - Memória Principal do SDF 88

4 - Memória Secundária 100

5 - Memória Local 112

6 - Considerações Finais 118

CAPÍTULO V - SIMULAÇÃO DO MÓDULO

1 - Considerações Iniciais 121

2 - Simulação Inicial do Módulo de Memória 124

3 - Segunda Simulação do Módulo de Memória 128

4 - Estrutura Modular da Simulação Final 132

5 - Considerações Finais 134

CAPÍTULO V - CONCLUSÕES

1 - Conclusões Finais 139

2 - A Continuidade do Trabalho 144

BIBLIOGRAFIA 145

APÊNDICE

A - ROTINA PARA COLETA DE DADOS 147

B - SIMULAÇÃO DO MÓDULO DE MEMÓRIA VIRTUAL 149

C - ABREVIAÇÕES UTILIZADAS 180

CAPÍTULO I

INTRODUÇÃO

1. *CONSIDERAÇÕES GERAIS*

O tema deste trabalho originou-se do interesse mútuo dos grupos de Hardware e Software básico do ICMSC e de Instrumentação Eletrônica do IFQSC, no desenvolvimento de sistemas computacionais baseados em microprocessadores. Em particular, atualmente, o interesse maior está voltado para o desenvolvimento de um Sistema Distribuído Funcional (SDF), destinado a ambientes computacionais onde exista grande demanda computacional, aliada à necessidade de processamento em tempo real, para controle e monitoração de sistemas físicos.

Esse sistema, objetivo de estudo em {Sa85}, apresenta uma distribuição funcional, caracterizada pela existência de módulos funcionais, para os quais são definidos conjuntos de funções a serem atendidas. A distribuição do processamento é ativada por comandos de alto nível, que circulam por um duto especial, o DCAN (duto de comunicação de alto nível).

Com o objetivo de se implementar para os módulos do SDF, um sistema de gerenciamento de memória com possibilidade de endereçamento virtual sem que com isso aumente-se a complexidade individual de cada módulo (o que é indesejável sob diversos aspectos) pode-se centralizar esse gerenciamento em um módulo especial, dedicado unicamente à implementação e gerenciamento da memória virtual, incluindo também, outras facilidades que visam minimizar os tempos envolvidos no acesso à memória, como é o caso do cache.

O objetivo deste trabalho é o de estudar, de um modo geral, os problemas envolvidos na implementação de sistemas com memória virtual e cache e, em particular, os problemas relacionados com a implementação de um módulo de memória virtual, para um sistema distribuído funcional, no caso o SDF.

Com base no estudo desenvolvido e nos problemas básicos verificados para esse tipo de implementação, o trabalho

apresenta diretrizes para uma arquitetura básica do módulo de memória virtual e destaca as principais características obtidas.

O trabalho procura apresentar, não apenas uma proposta de arquitetura para o módulo de memória virtual, mas também discutí-la exaustivamente, analisando o seu desempenho através de simulações, que levam a alterações significativas, de modo a se obter uma arquitetura com características adequadas ao SDF, tanto do ponto de vista físico como lógico.

A importância do desenvolvimento deste trabalho está relacionada, principalmente, com dois aspectos:

- Relevância do módulo de memoria virtual no SDF - com a implementação deste módulo, os demais módulos do SDF, adquirem a capacidade de endereçamento virtual, sem com isso ter sua complexidade aumentada e sem sobrecarregar seu processador , com tarefas relacionadas ao gerenciamento de memória. Assim, o módulo de memória virtual centralizando o gerenciamento de memória, permite uma menor complexidade nos demais módulos do SDF, facilitando o desenvolvimento do sistema, tanto do ponto de vista de hardware, quanto de software.

- Possibilidade de futuros estudos - o simulador apresentado neste trabalho, permite que se verifique a influência de diversas características no desempenho do módulo. Assim, pode-se verificar o comportamento do módulo de memória quando se introduz novas características no hardware e também novos algoritmos para o gerenciamento da memória virtual. Isso torna possível uma verificação rápida de novas idéias que possam surgir.

2. *ORGANIZAÇÃO DA DISSERTAÇÃO*

No capítulo II é apresentada a revisão bibliográfica da dissertação, que pode ser dividida em duas partes: Estudo do conceito e características de um sistema distribuído, com a finalidade de se estabelecer uma conceituação para melhor situar o sistema que suportará o módulo de memória. Estudo sobre memória, para estabelecer a conceituação necessária na definição do módulo de memória virtual. No estudo sobre memórias destacam-se a conceituação e formas de Implementação de Memória Virtual e Cache e uma verificação dos circuitos integrados que suportam gerenciamento de memória.

No capítulo III são apresentadas as características gerais do módulo de memória e a arquitetura proposta: Destaca-se ainda nesse capítulo a apresentação de um esquema lógico do hardware e a estrutura modular do software.

No capítulo IV é efetuada uma descrição do módulo, discutindo-se as características e formas de implementação dos três níveis de armazenamento propostos neste trabalho: cache, memória principal e memória secundária. A memória local do módulo, utilizada no gerenciamento da memória de todos os módulos do sistema , tem sua arquitetura apresentada nesse capítulo. Destaca-se ainda as justificativas de algumas características do módulo de memória.

No capítulo V é apresentada a simulação do módulo de memória, destacando-se as modificações efetuadas devido a algumas características não desejáveis, observadas nas simulações e a verificação da influência de alguns fatores no desempenho global do sistema.

No capítulo VI são apresentadas as conclusões finais do trabalho, seus aspectos relevantes e sugestões para a sua continuidade.

CAPÍTULO II

REVISÃO BIBLIOGRÁFICA

1. *CONSIDERAÇÕES INICIAIS*

Neste capítulo é apresentado o estudo bibliográfico desenvolvido para a realização deste trabalho. Esse estudo serviu como base para a conceituação do módulo de memória virtual, apresentada no capítulo III.

Os pontos relevantes do estudo abrangem:

a. Estudos de sistemas distribuídos (seção 2) com a finalidade de se estabelecer uma conceituação para melhor situar o sistema onde será utilizado o módulo de memória.

b. Estudo sobre memórias (seção 3) com a finalidade de se estabelecer a conceituação necessária, para a definição do módulo de memória.
   Nesta seção, aborda-se a conceituação e formas de implementação de memória virtual e de memória cache.
   Procura-se então, exemplificar os aspectos estudados, através da apresentação da memória virtual e cache, na família de computadores VAX 11. Finalmente, faz-se uma verificação dos circuitos integrados que suportam gerenciamento de memória.

2. *CONCEITO E CARACTERÍSTICAS DE UM SISTEMA DISTRIBUÍDO*

Os sistemas de processamento de dados distribuídos têm se tornado de grande importância, devido à realidade atual do processamento eletrônico de dados.

Embora todos concordem com a importância desses sistemas, não existe uma definição uniforme para os sistemas distribuídos, que satisfaça a todos os especialistas da área.

Strack, em {St84}, atribui a discordância de conceituação, ao fato de que a distribuição pode ocorrer em diversos níveis do sistema, tais como:

a. arquitetura interna da máquina,

b. disposição física,

c. modo de interconexão,

d. sistema operacional,

e. controle,

f. dados,

g. aplicações.

Enslow, em {En78}, propõe alguns critérios bastante abrangentes para definir o que é um sistema distribuído. Nessa definição, cinco aspectos são considerados:

a. multiplicidade de recursos de propósito geral (recursos físicos e lógicos),

b. distribuição física dos componentes físicos e lógicos devem interagir através de uma rede de comunicação,

c. um sistema operacional de alto nível para unificar e integrar o controle dos componentes distribuídos,

d. transparência do sistema (os serviços devem ser requisitados apenas pelo nome),

e. autonomia cooperativa (caracterizando a operação e iteração dos recursos físicos e lógicos).

Para que um sistema seja considerado distribuído, ele deve satisfazer todos os critérios estipulados. Um sistema onde apenas alguns destes aspectos estão presentes, não pode ser considerado um sistema distribuído {En78}.

No entanto, Enslow observa que a aplicação desses critérios, não deve ser feita de uma maneira binária e sim de uma forma mais flexível. Enslow propõe que os sistemas sejam classificados levando-se em conta três dimensões:

a. descentralização do hardware,

b. descentralização do controle,

c. descentralização da base de dados.

Com isso, pode-se classificar os sistemas segundo o grau de distribuição associado a cada dimensão, estabelecendo uma escala de descentralização do sistema, que vai desde os totalmente centralizados até os totalmente distribuídos.

Fathi, em {Fa83}, aborda o problema de sistemas com microprocessadores distribuídos.

Com o desenvolvimento dos microprocessadores, o emprego de sistemas com múltiplos processadores, tornou-se atrativo. Os sistemas com microprocessadores distribuídos apresentam algumas características das redes de computadores, tendo no entanto, um acoplamento mais rígido e, semelhanças com sistemas multiprocessadores, apresentando um acoplamento mais livre.

As principais características dos sistemas com microprocessadores distribuídos são:

a. autonomia dos elementos,

b. processadores dedicados a tarefa,

c. processadores com complexidade variável,

d. otimização do software e hardware,

e. comunicação à nível de dados,

f. compartilhamento de carga estático.

Fathi, em {Fa83}, faz algumas considerações e conclui que os sistemas com microprocessadores distribuídos devem ser usados em:

a. aplicações em controle de processos com demanda computacional diversificada e processamento em tempo real,

b. aplicações com grande demanda de processamento de entrada e saída de dados,

c. aplicações que necessitam de alta confiabilidade mas que devido a fatores financeiros, ou de espaço, não suportam uma grande redundância.

Em oposição, recomendam-se que esses sistemas devem ser evitados quando:

a. tudo o que se precisa é aumentar a capacidade do sistema,

b. a tarefa global, a ser executada não pode ser dividida em tarefas, com relativa independência.

Comparando a conceituação de sistemas distribuídos, apresentados em {Fa83} e {En78}, observa-se que os sistemas considerados como distribuídos em {Fa83} não satisfazem os critérios de Enslow. No entanto, Fathi, em {Fa83}, apresenta uma caracterização de sistemas com microprocessadores distribuídos, bastante clara e talvez bem mais realista que a sugerida por Enslow, em {En78}.

3. *MEMÓRIA*

3.1. *Algumas Considerações Sobre Endereçamento*

O rápido avanço da tecnologia ocorrido na última década, tem provocado uma sensível mudança na filosofia dos projetos de sistemas computacionais. Soltis, em {So81}, aborda este problema e observa que os sistemas projetados nos anos 80, precisam ser capazes de tirar vantagens desse avanço contínuo, e ainda ser suficientemente flexíveis para assimilação das novas tendências tecnológicas.

Os sistemas de memória, em particular, tem apresentado um avanço tecnológico muito rápido. Quando se trata de memórias, um fator limitante para que um sistema acompanhe as novas tendências, é o número de bits do endereço. A falta de um número suficiente de bits de endereçamento é um problema difícil de ser resolvido.

Assim, um endereçamento adequado é o primeiro ponto de grande importância no projeto de um sistema. Houdek, em {Ho79}, apresenta algumas considerações sobre o desenvolvimento do Sistema IBM/38 e observa que a mais importante decisão do projeto foi empregar uma estrutura de endereços, com bits suficientes para referenciar toda a memória principal e secundária, sem a sua reutilização.

3.2. *Memória Virtual*

3.2.1. *Conceito de Memória Virtual*

Com o aumento da complexidade e tamanho dos programas, surgiu a necessidade de memórias cada vez maiores. Como memórias de acesso rápido são caras, para comportar esta constante necessidade de aumento da memória principal, apareceu a necessidade de uma hierarquização da memória. A hierarquização deve ser de pelo menos dois níveis: uma memória principal, que pode ser diretamente acessada pelo processador, e uma memória secundária, que armazena as partes do programa a que não se está tendo acesso no momento. Com isto, pode-se ter programas bem maiores que a memória principal do computador que os estão executando. Esta idéia, é conhecida como memória virtual e dá ao usuário, a sensa-

ção de ter uma grande memória principal à sua disposição, quando na realidade tem uma memória relativamente pequena.

A conceituação de memória virtual começou a surgir no final dos anos 50, quando foi utilizada no desenvolvimento da máquina ATLAS na Universidade de Manchester. Denning, em {De70}, fala sobre o surgimento e evolução das idéias básicas, chegando aos atuais conceitos de implementação de memória virtual.

É fundamental na idéia de memória virtual a noção de que tem-se um grande espaço de endereçamento, distribuído em vários níveis de dispositivos de memória com diferentes capacidades de armazenamento, custo e velocidade. MacGregor em {Ma83}, apresenta o sistema com três níveis de memória, mostrado na Figura II-1.

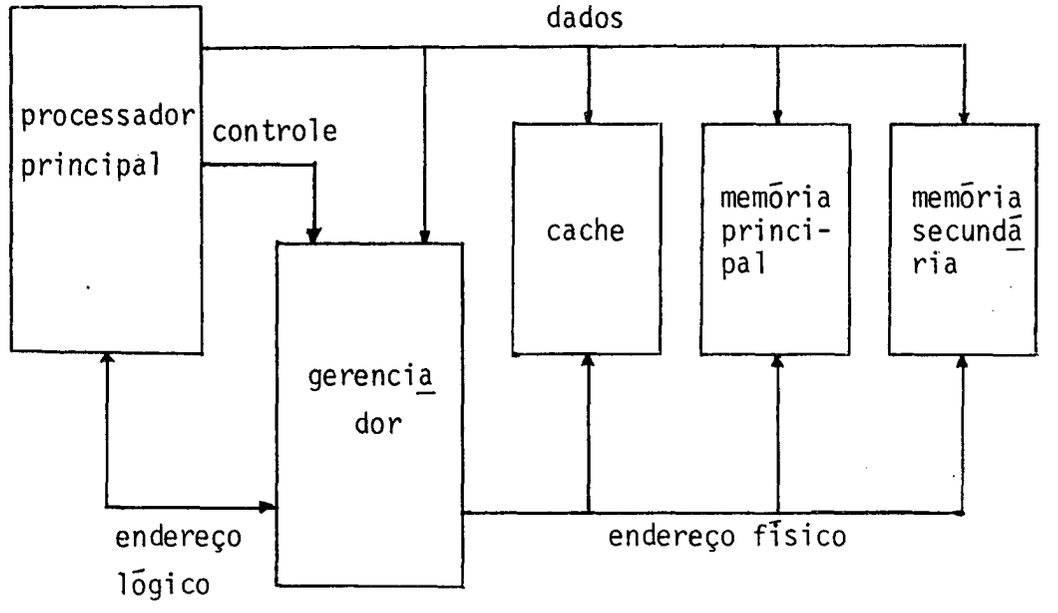


Figura II-1 - Sistema com Três Níveis de Memória

Observa-se, do que foi visto até agora, que o processador pode endereçar um espaço de memória muito maior do que a memória principal pode conter. Isto é, em sistemas com memória virtual, faz-se uma diferenciação entre espaço de endereço e espaço de localização física.

Um endereço usado pelo programador é um endereço virtual e compõe o espaço de endereço. Por outro lado, o endereço utilizado pela memória é chamado endereço real ou endereço físico e compõe o espaço de localização física. O espaço de localização física pode ser bem menor que o espaço de endereços.

3.2.2. *Implementação da Memória Virtual*

Sobre o Mapeamento da Memória

Para a implementação de uma memória virtual é necessário o conhecimento de uma série de informações, tais como: se o endereço a que se quer ter acesso está ou não na memória principal, em caso afirmativo, qual é o seu endereço na memória física, em caso contrário, onde está armazenado, etc.. Para isto, implementa-se uma tabela de mapeamento de endereços, que além de estabelecer uma correspondência entre o endereço virtual e o real, fornece informações da posição de memória que representa.

Denning, em {De70}, aborda o problema de mapeamento, implementando inicialmente, uma tabela que mapeia todos os endereços da memória. Tem-se para isto, um mecanismo de conversão conforme mostrado na Figura II-2.

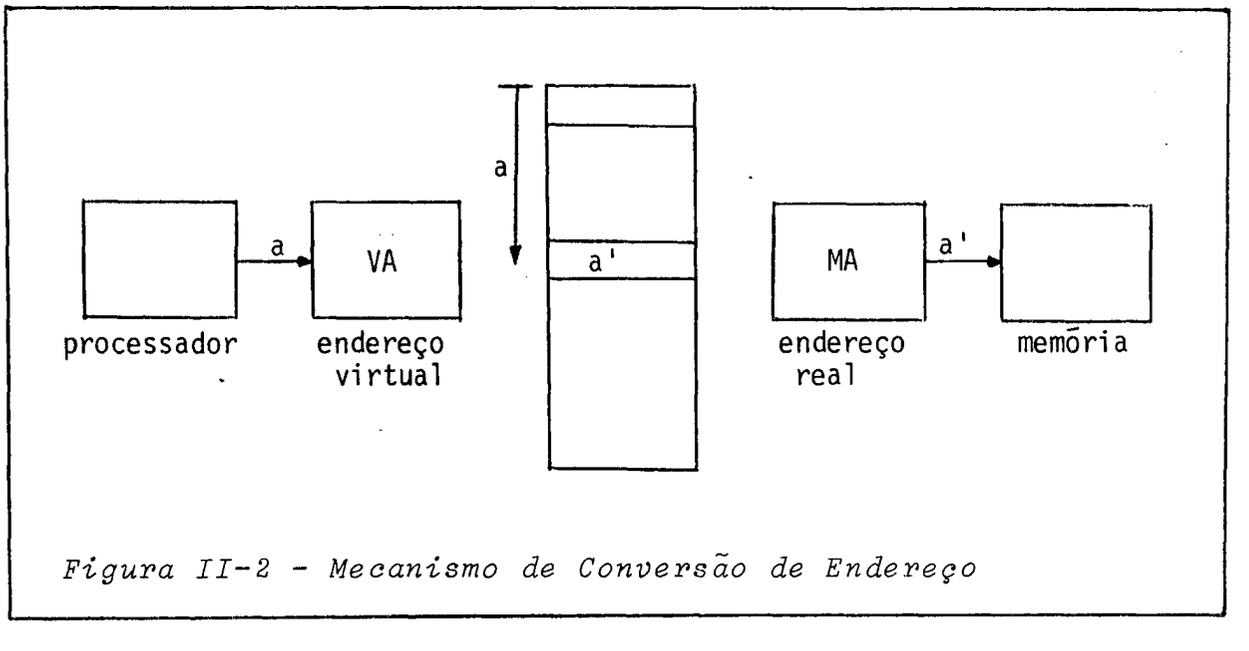


Figura II-2 - Mecanismo de Conversão de Endereço

O processador fornece um endereço (a) a ser convertido. Se o endereço estiver na memória principal, na localização "a" da tabela, tem-se o endereço real (a') relativo a "a". Caso o endereço não esteja na memória principal, existirá uma ausência da linha "a" na tabela.

Analisando esse tipo de mapeamento, Denning chega a conclusão que seria impraticável, pois, seria necessária uma tabela de mapeamento com um entrada para cada endereço da memória. Surge então, a necessidade de agrupar-se a memória em blocos e mapear estes blocos, que seriam indivisíveis do ponto de vista de conversões.

Cada bloco contém um grupo de informações localizadas em um espaço de endereço contínuo da memória. Dependendo de como são formados esses blocos, tem-se diferentes métodos de implementação da memória virtual.

a. segmentação: divide o espaço de endereços em blocos de tamanho arbitrário. Nesse método, os blocos são chamados de segmentos.

b. paginação: todos os blocos terão um tamanho fixo e são chamados de páginas.

c. combinação entre segmentação e paginação.

A. *SEGMENTAÇÃO*

Em um espaço de endereço segmentado, o endereço é composto por dois componentes (s,w), onde s é o segmento e w é a palavra dentro do segmento s.

A.1. *Implementação*

Para implementar o mecanismo de conversão de endereço, para uma memória virtual com segmentação, tem-se uma tabela de segmentos. Na tabela, que também é um segmento, a s-ésima entrada representa o segmento s, contendo as informações referentes ao endereço base do segmento (a) e seu limite (b), caso o segmento esteja na memória principal; se não estiver, estará em branco. Para a conversão de endereço devem-se executar as seguintes operações, como mostra a Figura II-3.

a. carregar s e w nos registradores de segmento e palavra,

b. se a s-ésima entrada estiver em branco, haverá erro por ausência de segmento na memória principal,

c. se w > b (limite de segmento), haverá erro de estouro - tentativa de acesso fora do segmento,

d. carregar (endereço base (a) + (w)) no registrador de endereço da memória.

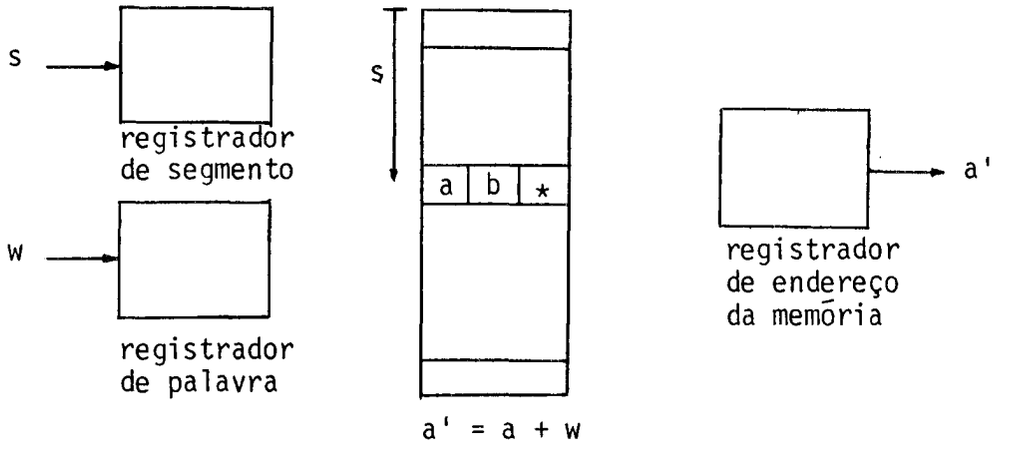


Figura II-3 - Conversão de Endereço para Memória Segmentada

Para cada entrada da tabela de segmentos, além das informações já vistas, existe um campo com bits de proteção (*). Estes bits, podem ser comparados com o tipo de acesso que se pretende fazer no segmento referenciado (leitura, escrita,...).

Caso exista uma discordância será gerado um erro de proteção.

É importante observar que para cada sistema implementado, a tabela de segmentos pode apresentar variações. Shaw apresenta em {Sh74}, o exemplo do Burroughs B5500 que utiliza segmentação. Neste caso, cada entrada da tabela de segmentos contém um bit que indica se o segmento está ou não na memória principal, além das informações já citadas.

A.2. *Estratégias de Alocação*

As estratégias de alocação e de liberação de segmentos em sistemas que utilizam segmentação, apresentam uma série de problemas. Como os blocos são de tamanho variável, no momento da alocação de um bloco na memória, deve-se procurar um es

paço que seja pelo menos igual ao tamanho do bloco. Nem sempre existe este espaço disponível, podendo ocorrer a existência de vários espaços livres, mas não contínuos. Assim, deve-se escolher entre as seguintes estratégias alternativas {Sh74}:

a. política de relocação: onde um ou mais blocos são deslocados, para gerar espaço contínuo,

b. política de compactação: todos os espaços livres da memória são compactados em um único espaço,

c. política de bloqueamento: os blocos que não conseguem espaço livre são colocados em um espaço de espera.

Para o caso da existência de mais de um espaço livre na memória, capaz de comportar o bloco a ser armazenado, existem duas estratégias de alocação:

a. First Fit: onde o primeiro espaço encontrado é utilizado,

b. Best Fit: onde é pesquisado na lista de espaços vazios, aquele onde melhor se encaixa o bloco, deixando um menor espaço vazio.

A estratégia de First Fit é mais fácil e mais rápida de ser implementada. No entanto, o espaço vazio deixado entre o bloco armazenado e o próximo bloco pode ser maior. Esta característica de estratégia First Fit pode ser benéfica ou não, dependendo do caso.

A.3. *Fragmentação Externa*

Os espaços vazios citados no ítem anterior, podem ser às vezes, muito pequenos para comportar novos blocos, gerando o que se denomina fragmentação externa {De70}. Isto é um tipo de fragmentação que ocorre em sistemas que utilizam segmentação, consistindo na formação de lacunas entre blocos armazenados na memória, que são muito pequenas para serem utilizadas, como pode ser observado na Figura II-4.

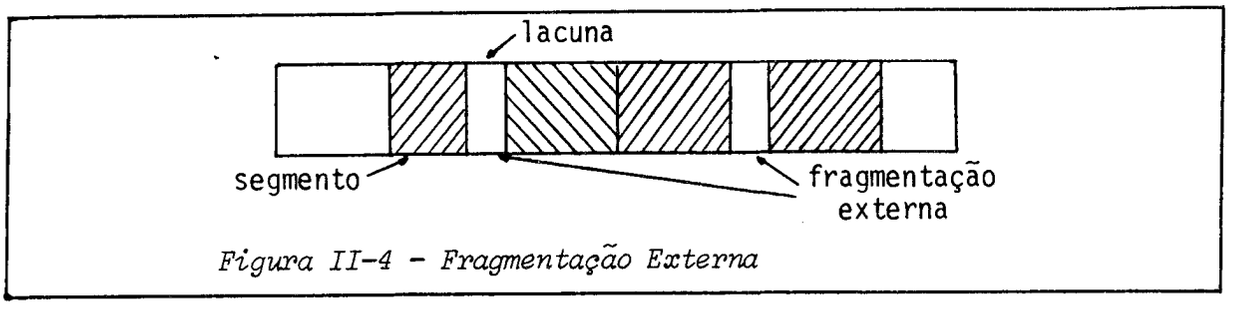


Figura II-4 - Fragmentação Externa

B. *PAGINAÇÃO*

O segundo tipo de implementação de memória virtual é a paginação {Ho72, De70, Sh74} , onde a memória principal é organizada em blocos, denominados páginas físicas, de igual tamanho. O espaço de memória virtual também é organizado em blocos, cujo tamanho é igual ao das páginas físicas, e são denominados páginas. As páginas são as unidades básicas para armazenamento e transferência de informações, além de serem, geralmente, a unidade básica para proteção.

O endereçamento das páginas é feito através de um par (p, w) onde p é o número da página e w o número da palavra dentro da página. Assim, se cada página contém z palavras, um endereço virtual "a" será dado por:

$$a = p.z + w$$

B.1. *Implementação*

Para implementar o mecanismo de conversão de endereços em sistemas que utilizam paginação, utiliza-se uma tabela de páginas, que analogamente à tabela de segmentos, fornece informações sobre o endereço das páginas. Nessa tabela, a p-ésima entrada, que representa a página "p", contém informações sobre o endereço físico da página "p", se esta estiver na memória física; estará em branco caso contrário. Além disso, tem-se os bits de proteção.

Para a conversão de endereço, devem-se executar as seguintes operações, como mostra a Figura II-5:

a. carregar o endereço virtual no registrador de endereços virtuais,

b. determinar a que página virtual o endereço pertence:

$$p = \lceil a/z \rceil,$$

c. determinar a palavra a ser acessada:

$$w = \text{resto}(a/z),$$

d. se a p-ésima entrada estiver em branco, haverá erro por ausência de página na memória principal,

e. carregar (p' + w) no registrador de endereços da memória.

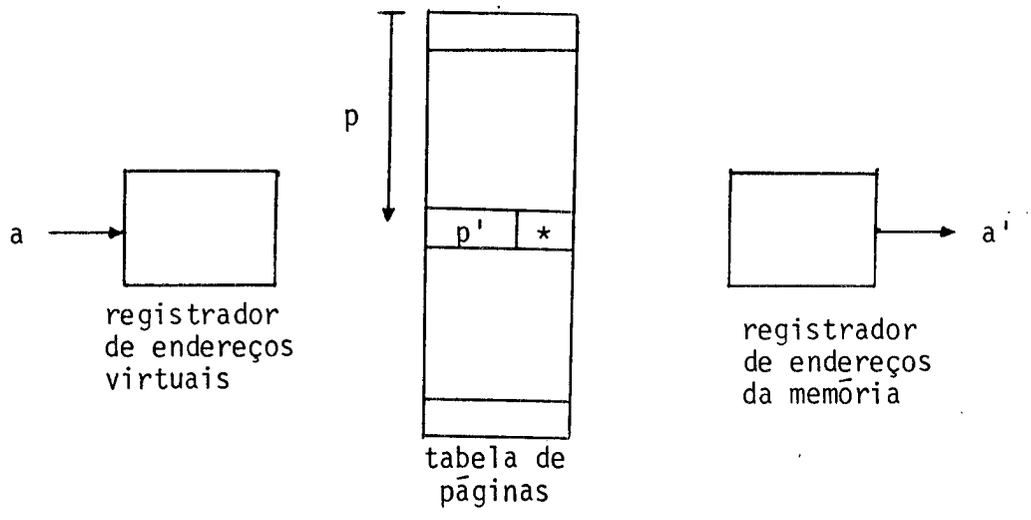


Figura II-5 - Conversão de Endereço para Memória Paginada

Denning, em {De70}, faz considerações de como fica o mecanismo de conversão de endereços, se a tabela de páginas ou de segmentos for armazenada na memória principal, ao invés de ser um componente do mecanismo de conversão de endereços. Nesse caso, deve-se fazer dois acessos à memória. Um primeiro na tabela, para obtenção do endereço do segmento (ou página) desejado e um segundo para o segmento (ou página) que está sendo referenciado. Para minimizar os acessos, pode-se utilizar uma memória associativa, para armazenar a tabela de segmentos ou de páginas.

No caso citado no parágrafo anterior, deve-se manter um registrador contendo o endereço base da tabela (RBT), para que seja possível o acesso.

O primeiro computador a utilizar paginação foi o ATLAS desenvolvido na Universidade de Manchester. A paginação foi utilizada apenas para implementar uma grande memória virtual, não se preocupando com a implementação de multiprogramação. O endereçamento era feito através de uma concentração (n ⊕ m) de bits onde os n bits de mais alta ordem correspondem ao número da página e os m bits de mais baixa ordem ao número de linha.

Um outro exemplo de utilização de paginação é o VAX 11/780, onde se tem uma memória virtual de 4 G bytes, dividida em páginas de 512 bytes {Di79}.

B.2. *Fragmentação Interna*

Uma das grandes vantagens da paginação é evitar a fragmentação externa, porém, surge um novo tipo de fragmentação, denominada fragmentação interna. Na paginação, todas as páginas têm o mesmo tamanho, e os segmentos não têm, necessariamente, um tamanho que é um múltiplo da página. Portanto, pode haver uma sobra na última página que armazena o segmento, gerando um espaço vazio. Na figura II-6, é apresentado um exemplo de fragmentação interna, para um segmento que ocupa k páginas.

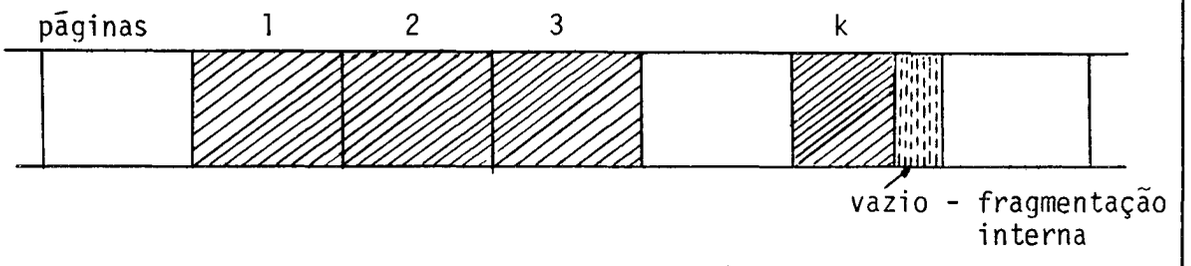


Figura II-6 - Fragmentação Interna

B.3. *Tamanho da Página*

Shaw, em {Sh74}, comenta que o tamanho das páginas é tipicamente de 512 ou 1024 palavras. Denning, em {De70}, e Doran, em{Do76}, entram em maiores detalhes sobre como o tamanho da página pode influenciar no desempenho da máquina.

Existem basicamente, dois fatores que influenciam a escolha do tamanho da página. O primeiro é o problema da fragmentação e o segundo é o problema da eficiência nas operações de transporte das páginas.

Inicialmente, sobre o problema da fragmentação, deve-se considerar que, se o tamanho da página for pequeno, existe uma menor perda de memória, devido à fragmentação interna; porém tem-se um maior número de páginas e, portanto, uma maior tabela

de páginas, ocupando mais espaço de memória.

Denning, em {De70}, faz um estudo sobre as perdas de memória e propõe um tamanho de página ótimo, dado por:

$$z = (2.s_o.(c_1/c_2))^{1/2}$$

onde:

s_o é o tamanho médio do segmento,

c_1 é o custo por perda, de uma palavra para a tabela de segmentos,

c_2 é o custo por perda, de uma palavra pela fragmentação interna.

Considerando-se segmentos de 1000 palavras, e que $c_1 = c_2$, tem-se que $z \leq 45$ palavras, o que é uma página pequena.

Considerando o problema do tempo em operações de transporte, tem-se que se a página for pequena, deve haver um número maior de acessos à memória secundária, e a cada acesso serão transferidas um menor número de palavras. Como o tempo para um transporte depende em grande parte do tempo gasto na localização da página, na memória secundária, e não do tempo e transferência, seria preferível páginas grandes.

Denning, em {De70}, faz uma análise para diversos tipos de armazenamento secundário, e conlui que o tamanho das páginas para minimizar o tempo de transferência, deve ser bem maior do que o obtido para minimizar o problema da fragmentação interna (no mínimo 500 palavras).

Assim, para a determinação do tamanho das páginas, deve ser feito um estudo tentanto balancear o problema de perda de tempo e de perda de memória.

C. *PAGINAÇÃO COM SEGMENTAÇÃO*

Neste método, procura-se unir as vantagens das duas técnicas (segmentação e paginação). A memória será paginada, isto é, dividida em blocos de tamanhos iguais, tendo-se a vantagem do tratamento uniforme {De70} da memória principal, oferecido pela paginação.

A paginação com segmentação, contém uma tabela de segmentos que mapeia todos os segmentos existentes, fornecendo o endereço da tabela de páginas de cada segmento. Na tabela de seg

mentos é implementada a proteção, e é feito o compartilhamento de trechos de processos. Portanto, para o acesso a uma palavra de um segmento, deve-se inicialmente localizar a tabela de páginas do segmento (através da tabela de segmentos), e então, localizar a página que contêm a palavra (através da tabela de páginas). Localizada a página, através do endereço localiza-se a palavra.

Para se ter acesso a palavra w do segmento s, isto é, o endereço virtual (s,w), consideram-se os seguintes passos , ilustrados na Figura II-7.

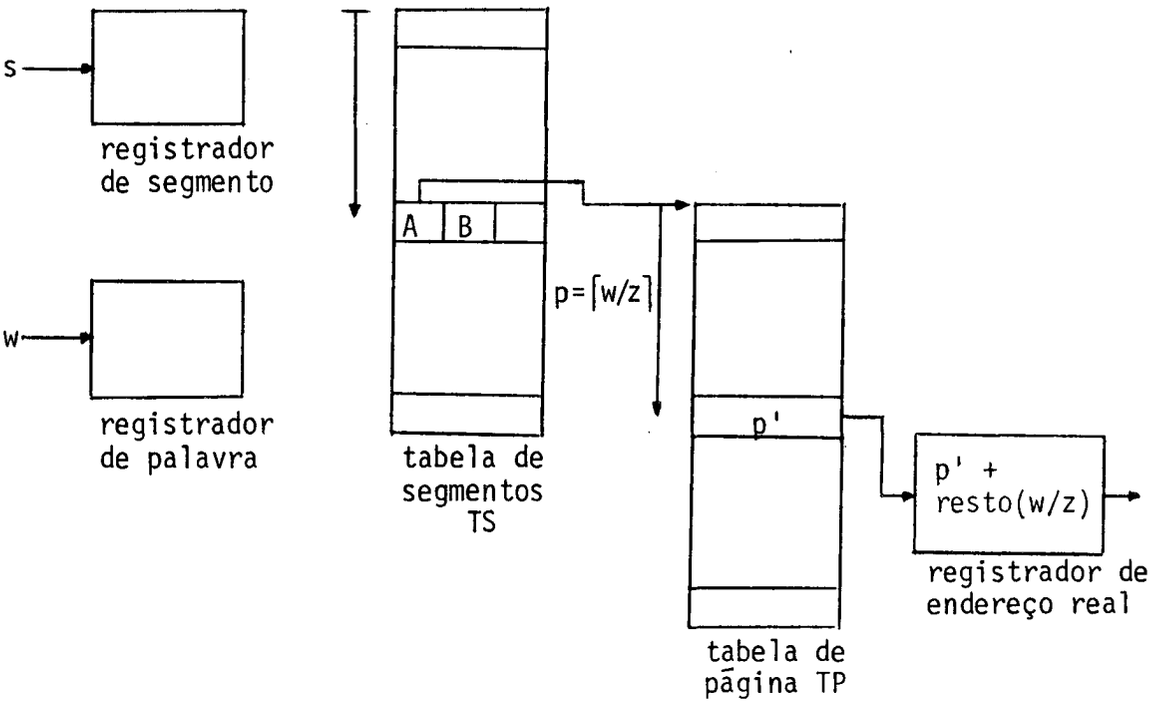


Figura II-7 - Conversão de Endereço para Paginação com Segmentação

a. coloca-se (s,w) no registrador de segmentos e palavras,

b. se a s-ésima entrada da tabela de segmentos estiver em branco, então o segmento s não está na memória,

c. se w for maior que o limite do segmento, haverá erro,

d. localiza-se a tabela de páginas do segmento, fazendo TS[s] + ⌈w/z⌉ onde, TS[s] é o conteúdo da s-ésima linha da tabela e ⌈w/z⌉ é a página a ser acessada,

e. localiza-se o endereço real fazendo:

TP[TS[s] + ⌈w/z⌉] + resto(w/z)

onde:

TP é a tabela de página,

TP[s] + ⌈w/z⌉ é o endereço do início da página.

Como pode ser observado, nesse caso deve-se ter acesso a duas tabelas e depois ao endereço real para a obtenção da informação desejada. Isto é, serão necessários três acessos à memória ao invés de dois como nos outros casos. Assim, a utilização de memória associativa, ou de algum outro mecanismo que minimize estes acessos à memória (tais como registradores dedicados), é essencial.

Pode-se citar como exemplo de utilização desta técnica o GE 645 (MULTICS). O MULTICS mantém um registrador (DBR), que indica o processo que está sendo executado, apontando para sua tabela de segmentos. O endereço virtual contém dois níveis; um indicando o segmento considerado e outro indicando a página procurada. Na figura II-8 tem-se ilustrado o esquema de um acesso no MULTICS.

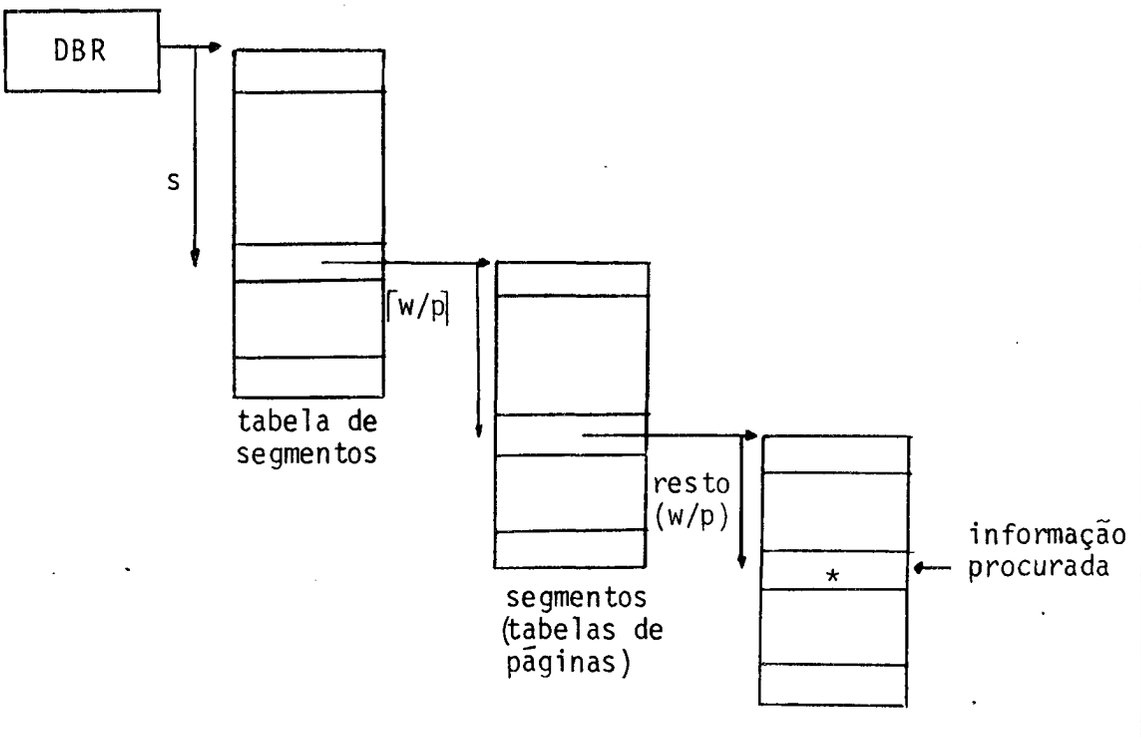


Figura II-8 - Mecanismos de Conversão de Endereços do MULTICS

Requisitos Necessários para Implementação de Uma Memória Virtual

MacGregor, em {Ma83}, analisa os requisitos necessários para suportar uma memória virtual. O processador que faz o gerenciamento da memória virtual, deve ser capaz de executar três funções básicas:

a. reconhecer uma ausência de página,

b. salvar as informações necessárias, para retornar às condições iniciais após a solução da ausência detectada,

c. recarregar o estado que foi salvo, reassumindo o processamento normal.

Pode-se então, dividir o mecanismo de uma memória virtual em quatro passos:

1º Passo: reconhecimento de falha onde a unidade que gerencia a memória virtual deve ser capaz de reconhecer problemas do tipo:

a. ausência de página (acesso a instrução ou operação que não está na memória principal),

b. detecção de erros (problemas na conversão de endereço, no acesso à memória, etc..).

2º Passo: salvar as informações que serão necessárias para reconstruir o estado da máquina, após a correção da falha detectada no 1º passo (isto é necessário porque o passo seguinte será a execução de rotinas que tentarão resolver o problema detectado, provocando as variações no estado da máquina).

3º Passo: execução de rotina para resolver a falha, onde uma rotina é executada para verificar o tipo de problema ocorrido e tomar decisões. Se ocorreu uma ausência de página ou qualquer problema que pode ser resolvido, a rotina executa os reparos necessários, em caso contrário, o processo é suspenso.

4º Passo: recarregar o estado da tarefa anterior e reassumir a execução. Este é o passo mais complexo. Para recarregar o estado anterior é preciso que ele seja: lido da memória, carregado na máquina e que haja uma ava-

liação para verificar o que é preciso ser feito, para retornar a máquina ao estado anterior à ocorrência da falta.

Existem dois métodos básicos para a implementação de memória virtual em um processador, método do reinício da instrução e método da continuação da instrução.

A. *MÉTODO DO REINÍCIO DA INSTRUÇÃO*

Este método é o mais utilizado e caracteriza-se pelo reinício da instrução na qual a falha ocorreu. Isto é, se uma sequência de microinstruções A, B, C de uma instrução, está sendo executada, e após a execução das microinstruções A e B ocorre a ausência da página que contém a microinstrução C, tem-se uma interrupção para trazer essa página para a memória. Com todas as microinstruções na memória, a execução continuará do início da instrução, isto é, as microinstruções A e B serão novamente executadas e logo após será executada a microinstrução C.

Desvantagens do Método de Reinício de Instrução

Quando um recurso visível ao usuário é utilizado como fonte e destino para dados dentro de uma instrução, este método torna-se complicado. Como exemplos dos problemas que ocorrem, podem-se citar:

a. operações aritméticas com precisão estendida. Neste caso, os bits de estado que indicam se houve transporte ou empréstimo, que são utilizados nos cálculos e são variados durante a instrução, devem ser reinicializados,

b. modo de endereçamento com auto-incremento/auto-decremento . Se a ausência é detectada após o incremento ou decremento , esses devem voltar ao valor inicial,

c. acessos a dispositivos de entrada/saída. Se a ausência ocorrer após o acesso, o reinício da instrução pode ocasionar erros se o acesso for novamente executado,

d. tranferência de memória com superposição.

O nível de dificuldade de implementação deste método depende do conjunto de instruções suportado pelo processador e de seus modos de endereçamento. Se existir um número limitado

de situações que poderão ocasionar erros, é possível prever um mecanismo para resolvê-los. No entanto, tendo-se um conjunto de instruções poderoso e variado, com modos de endereçamento igualmente poderosos, ficará bastante complexa a solução dos inúmeros problemas que serão encontrados.

B. *MÉTODO DA CONTINUAÇÃO DA INSTRUÇÃO*

Neste método, ao invés de salvar o estado do início da instrução, salva-se o estado completo da máquina no momento em que a falha é detectada. Após o atendimento da falha, a execução da instrução recomeça do ponto em que estava. Para o exemplo da instrução com microinstruções A, B e C citado no método do reinício, tem-se que, após solucionar a ausência da página que contém a microinstrução C somente essa última será executada. Isto é, a execução continua do ponto em que parou.

Desvantagens do Método da Continuação

a. problemas com as instruções que requerem execução sem interrupção: para esses problemas, deve existir algum mecanismo que force o reinício da instrução. MacGregor, em {Ma83}, cita como exemplo deste tipo de problema, a instrução "teste e atribua valor" (TAS) do MC 68000 que pode ser usada para implementar mecanismos de proteção por semáforos ou outras condições,

b. recursos e tempo utilizados para suportar a necessidade de salvar e depois recarregar o estado interno: o método da continuação requer que o estado interno seja salvo quando ocorrer uma falha. Normalmente, o estado compreende um grande número de registradores e retentores . Isto implica em uma necessidade adicional de recursos e de controle além de um gasto maior de tempo,

c. vulnerabilidade a erros de projeto: é o problema mais complexo associado ao método da continuação, pois as condições que existiam antes das falhas, devem ser recarregadas com exatidão. Assim, os passos para verificar a validade das operações devem ser rigorosos.

Apesar das desvantagens descritas acima, MacGregor, em {Ma83}, posiciona-se favoravelmente ao método da continuação,

pois este parece em geral, mais natural e menos destrutivo para o fluxo da máquina.

Algoritmo para Troca de Página

Quando algum endereço, que não estiver na memória principal for requisitado e não tiver espaço disponível na memória principal, é necessário fazer-se uma troca de páginas entre memória principal e secundária.

Os algoritmos de troca de páginas podem ser divididos em dois grupos.

a. Algoritmos baseados no uso: onde são consideradas referências feitas às informações. Exemplo - LRU (menos recentemente utilizado),

b. Algoritmos não baseados no uso: onde não são consideradas referências feitas às informações. Exemplos - FIFO (primeiro a entrar será o primeiro a sair) e aleatório.

3.2.3. *Proteção da Memória*

Quando foram mostradas as tabelas de segmentos ou páginas, foi citada a existência de alguns bits de proteção. Estes bits fazem parte de proteções contra tipos de acessos para as páginas ou segmentos da memória. As páginas podem ser classificadas por permitirem ou não leitura, escrita ou execução. Pode-se ter desde casos em que nenhum acesso é permitido (não é possível ler, escrever ou executar a página), até casos em que todos os acessos são permitidos, passando por todas as combinações possíveis.

Um exemplo da importância deste tipo de proteção é a que não permite que se escreva nas páginas que contêm o sistema operacional.

Um outro tipo de proteção é a que protege os segmentos de acessos não apropriados. Em todos os acessos é verificado se o endereço requerido realmente faz parte do segmento a que se quer ter acesso. Isto é feito, utilizando-se o endereço base e o comprimento do segmento.

3.3. *Exemplo de Implementação de Memória que Suporta Endereçamento Virtual - Família VAX 11 - (VAX 11/780 e VAX 11/759, Sistema Operacional VAX/VMS)* {Di79, Di80, Le82}

3.3.1. *Implementação da Memória Virtual*

A família VAX 11 tem um espaço de endereçamento virtual de 4 G bytes, que é acessado por um endereço virtual de 32 bits. Este espaço é dividido em páginas de 512 bytes, endereçáveis por 21 dos 32 bits do endereço. O endereçamento da palavra dentro da página é feito pelos 9 bits menos significativos do endereço. Os dois bits mais significativos definem a região da memória a que está tendo acesso.

A memória do VAX é dividida em duas regiões denominadas espaço do processo e espaço do sistema {Di79, Le82}. O espaço do processo é dividido em região Po - programa e região P1 - controle; este espaço pode ser chaveado, tendo-se um para cada processo. O espaço do sistema é também dividido em duas regiões: região do sistema e região reservada. O mapa de endereços do espaço de sistema não é chaveado, sendo comum para todos os processos. O sistema operacional fica no espaço do sistema. Os 2 bits mais significativos do endereço determinam essas regiões conforme mostra a Figura II-9.

Cada página, tanto para as regiões Po e P1, como para a região do sistema, é definida por uma entrada na tabela de páginas (PTE - page-table entry). Cada PTE contém 32 bits com os seguintes campos:

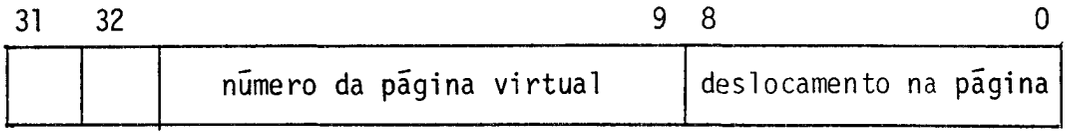
a. PTE <31> - bit de validade: indica se a entrada contém informações válidas,

b. PTE <30:27> - proteção: indica que privilégio é necessário para ler ou escrever na página,

c. PTE <26> : indica se ocorreu uma escrita na página,

d. PTE <25:21>: campo reservado para o sistema operacional,

e. PTE <20:0> - número da página física: localiza a página na memória física.



0 0 - região Po
0 1 - região P1
1 0 - espaço do sistema
1 1 - não usado

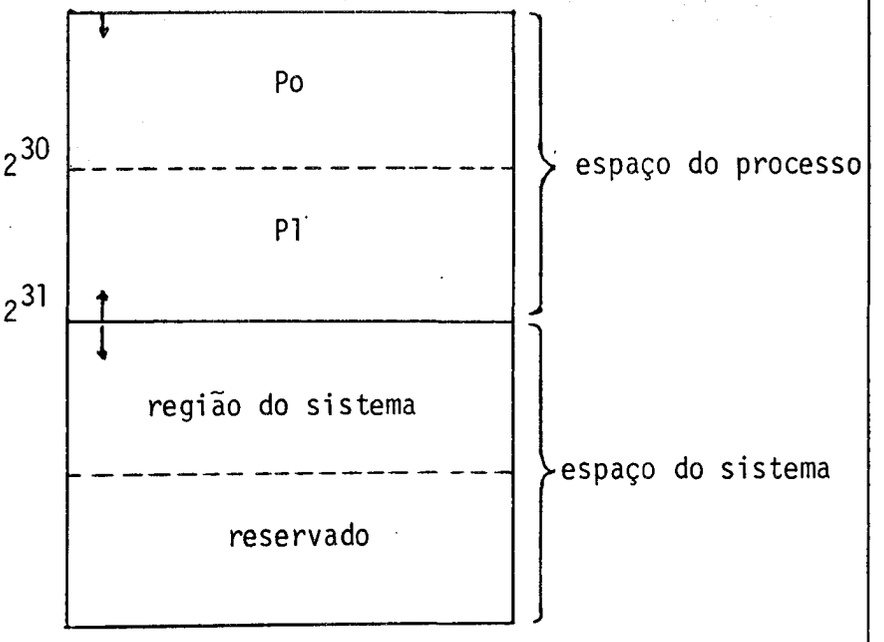


Figura II-9 - Divisão de Memória do VAX

Para o espaço do sistema, existe uma única tabela de páginas, que é definida por dois registradores que contêm seu endereço físico base (SBR) e seu comprimento (SLR). As tabelas de páginas de Po e P1 para cada processo, também são definidas por dois registradores contendo o endereço base (PoBR,P1BR), e o comprimento da tabela (PoLR, P1LR), com a diferença de que o endereço é virtual. A conversão de um endereço virtual do espaço de processo, envolve dois acessos. O primeiro na tabela de páginas do sistema para determinar o endereço da tabela de páginas do processo. O segundo é feito na tabela de páginas do processo, para determinar o endereço físico do elemento especificado.

Como existe uma tabela de páginas de Po e P1 para cada processo, quando ocorre um chaveamento de processo, os registradores PoBR, P1BR, PoLR e P1LR também devem ser chaveados.

Como a tabela de páginas do sistema é compartilhada por todos os processos, os registradores SBR e SLR não são afetados no chaveamento.

3.3.2. *Implementação do Gerenciamento de Memória*

O VAX-VMS contém um limite máximo de páginas, de cada processo, que podem estar na memória principal. Quando este limite é atingido e necessitam-se colocar mais páginas na memória principal, deve-se escolher, dentre aquelas do processo que está requisitando espaço para uma nova página, uma para ser retirada da memória principal.

A escolha é feita pelo processo FIFO, isto é, a primeira que entrar será a primeira a sair. Para cada processo é mantida uma estrutura de dados, chamada lista do conjunto residente, que descreve as páginas contidas no conjunto residente do processo.

Quando uma página é retirada do conjunto residente, ela é inicialmente colocada em uma lista livre, se não houve modificações na página (bit <26> não modificado) ou para uma lista modificada, em caso contrário. Estas listas têm várias finalidades:

a. atuam como cache: muitas páginas que faltam ainda estão na lista livre ou na modificada, não sendo necessário um acesso à memória secundária,

b. escrita empilhada: várias páginas podem ser escritas com um único acesso à memória secundária,

c. leitura empilhada: as páginas ficam arranjadas na memória secundária de modo a ser possível uma leitura empilhada,

d. muitas escritas são economizadas com o término do programa.

3.3.3. *Implementação de Proteção de Página pelo VAX/VMS*

O VAX/VMS suporta quatro modos para controle de acesso:

a. *Kernel:* usado pelo núcleo do sistema operacional para gerenciamento de página, acionamento de entrada e saída e articulador,

b. *Executive:* chamado por serviços do sistema operacional, in

cluindo sistema de gerenciamento de registros.

c. *Supervisor:* usado como interpretador de comandos,

d. *User:* usado por códigos a nível de usuários, utilitários, compiladores, etc..

Cada código contido nos bits <30:27> tem direito a um tipo de acesso dependendo do modo em que está rodando. Uma página pode ser não acessível, acessível só para leitura ou acessível para leitura e escrita. Por exemplo, se os bits de proteção forem 0000 nenhum tipo de acesso é possível, para todos os modos. Para 0010, estando-se no modo Kernel, é possível ler e escrever, para outros modos a página é não acessível. Para 0100 a página é acessível para leitura e escrita para todos os modos.

Tem-se ainda no VAX/VMS uma proteção contra violação do comprimento. Esta proteção é feita através dos registradores que indicam o endereço base e o comprimento da tabela que se está querendo ter acesso.

3.3.4. *Conversão de Endereço*

A rotina de conversão de endereço recebe:

a. endereço virtual,

b. acesso intencionado (leitura ou escrita),

c. modo contra o qual checar este acesso,

e fornece o endereço físico correspondente ao endereço virtual fornecido.

O processo de conversão de endereço para a região do sistema, pode ser subdividido nos passos mostrados na Figura II-10, e descritos a seguir:

a. retirar os bits de 9 a 29 do endereço virtual (<9:29> endereço da página),

b. multiplicar este endereço por quatro. Isto é, forma-se uma palavra com o endereço virtual da página sendo os bits de 2 a 22 e, os demais bits iguais a zero,

c. checar a palavra obtida com o registrador SLR para verificar se não haverá violação (endereço obtido maior que o comprimento da tabela),

d. somar a palavra obtida em b. com o registrador SBR, obten-

do-se o endereço físico da tabela de páginas,

e. na entrada da tabela de páginas obtida em d., tem-se o número da página (bits <20:0>);

f. o endereço físico do dado será a concatenação dos bits obtidos no item e., com o byte de deslocamento do endereço virtual, sendo que os 2 bits mais significativos devem ser zero.

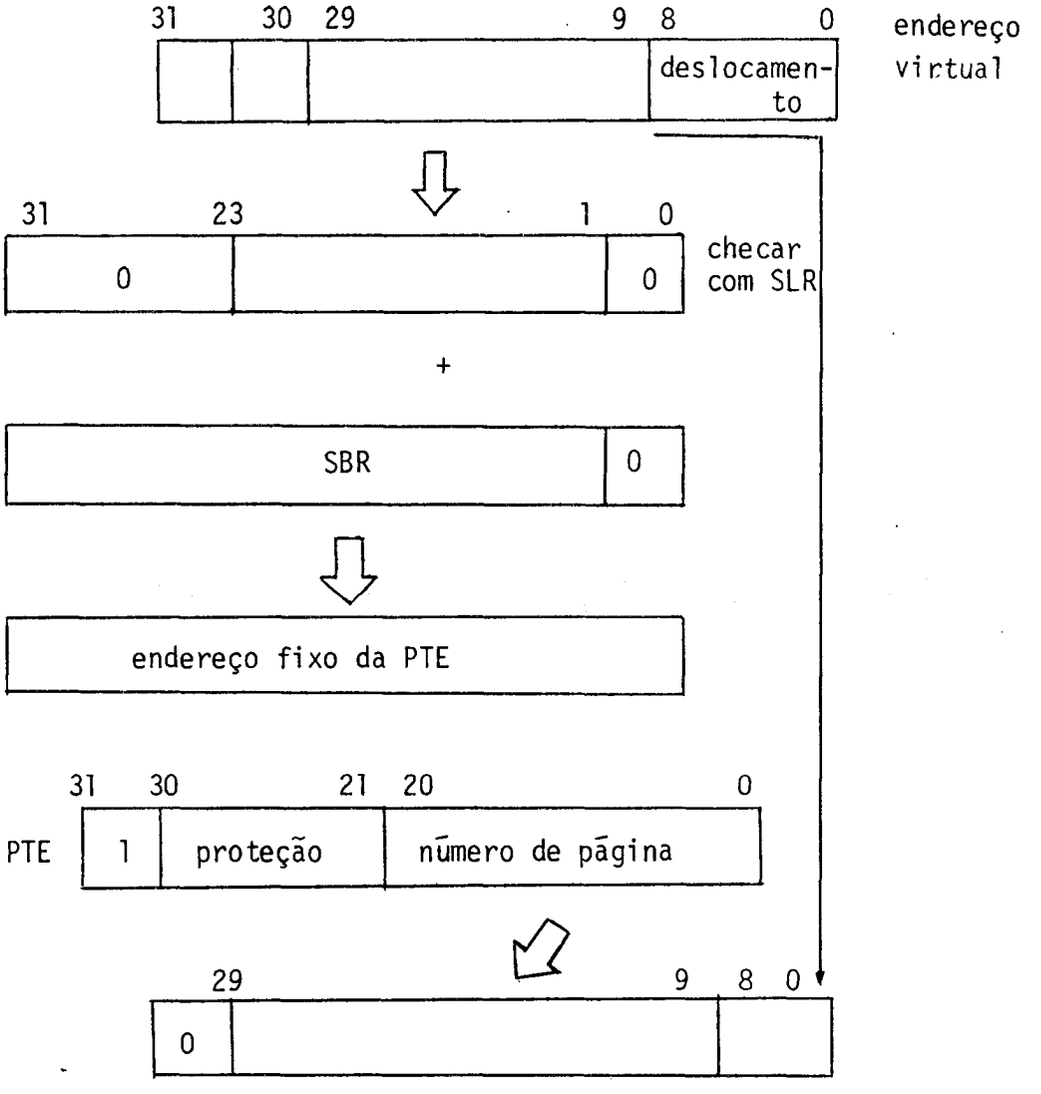


Figura II-10 - Conversão de Endereço no VAX

Quando o endereço a ser convertido está no espaço do processo, o procedimento é parcialmente análogo ao caso anterior. A diferença está em que os endereços-base contidos em P0BR e P1BR são virtuais e portanto, o endereço obtido no item d., será um endereço virtual. Assim, antes de ser utilizado como entrada para a tabela de páginas da região P0 ou P1 a que se refere, deverá haver uma conversão deste endereço para um endereço físico, utilizando agora, uma conversão de endereço para a região do sistema. Obtendo o endereço físico, o processo pode ser completado de maneira análoga à conversão para endereços no espaço do sistema, vista anteriormente.

3.4. *Cache*

Cache são memórias pequenas e de alta velocidade, utilizadas em computadores, para armazenar, temporariamente, porções da memória que estão (ou acredita-se que estejam), frequentemente em uso.

Como o desempenho dos computadores está sempre limitado devido ao tempo de acesso à memória, é interessante que este tempo possa ser minimizado. No entanto, memórias mais rápidas são mais caras, sendo que a relação entre o aumento do custo e a diminuição do tempo de acesso não é compensadora.

Assim, surgiu a idéia do cache, que é uma memória pequena e portanto, não aumenta muito o custo do sistema computacional, mas é bastante rápida (tempo de acesso de cinquenta a cem nano segundos) e, sendo utilizada de modo conveniente, pode diminuir sensivelmente o tempo de acesso às informações.

A memória cache é um outro nível de armazenamento em um sistema, sendo que sua relação com a memória principal pode ser comparada com a relação existente entre memória principal e secundária. Isto é, o cache tem uma capacidade de armazenamento menor que a memória principal, mas um tempo de acesso bem menor, da mesma forma que a memória principal tem uma capacidade menor mas é mais rápida que a memória secundária.

No gerenciamento de uma memória virtual, quando existe uma ausência de página, deve-se fazer um acesso à memória secundária para que a página ausente seja trazida para a memória principal. Da mesma forma quando existe uma ausência no cache, a informação deve ser trazida da memória principal. Nesse caso a u

nidade de transferência de informação é a linha, que corresponde ao conceito de página para memória virtual.

Smith, em {Sm82}, atribui o sucesso das memórias cache ao que ele chama de propriedade de localização. Esta propriedade tem dois aspectos: temporal e espacial.

No aspecto temporal, uma informação que será usada em um futuro próximo, é provável que esteja sendo utilizada no momento. Esse comportamento pode ser esperado devido aos laços dos programas, nos quais tanto os dados quanto as instruções são reutilizados.

No aspecto espacial, tem-se que a porção do espaço de endereços que está em uso é, geralmente, um pequeno número de segmentos desse espaço. Tal comportamento pode ser atribuído a ítens relacionados aos dados, que geralmente são armazendados juntos.

O "modo conveniente de utilização" citado anteriormente, refere-se a algoritmos apropriados para busca, localização e reposição de informações, tamanhos ideias e outras informações. A otimização do projeto de uma memória cache, tem quatro aspectos {Sm82}:

a. minimizar a probabilidade de não encontrar no cache a informação referenciada (taxa de falta),

b. minimizar o tempo de acesso à informação que está no cache,

c. minimizar o atraso no caso de ausência de informação no cache,

d. minimizar a sobrecarga de variação da memória principal.

A seguir pretende-se comentar alguns aspectos importantes para o desenvolvimento de uma memória cache.

Algoritmos de Busca

O algoritmo de busca é utilizado para decidir quando trazer informações para o cache.

Existem, basicamente, dois tipos de algoritmos:

. pré-busca: onde a informação é trazida antes de ser necessária,

. busca em demanda: onde a informação é trazida quando necessária.

Smith, em {Sm82}, apresenta resultados de simula-

ções de busca com e sem pré-busca, concluindo que a pré-busca sempre leva a uma diminuição na taxa de faltas.

No caso de pré-busca, existem três problemas básicos:

a. quando iniciar a pré-busca: existem várias possibilidades:

   i. quando fizer uma leitura ou escrita de dados,

   ii. quando ocorre uma ausência (nesse caso pode-se trazer para o cache a linha que faltou e a(s) próxima(s) linha(s),

   iii. quando se tem um acesso ao último segmento da linha considerada,

   iv. quando um acesso padrão tem sido observado.

b. quais linhas serão trazidas: no caso de memórias cache, como a questão de velocidade é importante, normalmente apenas a linha seguinte à referenciada é considerada na pré-busca. Isto é, se a linha i é referenciada apenas a linha i + 1 é considerada na pré-busca,

c. qual a situação das linhas trazidas por busca, com relação à relocação: neste caso, pode-se ou não alterar seus bits de relocação, sendo que Smith, em {Sm82}, observa que isto causará pouco efeito na taxa de acertos.

Existem, basicamente, três tipos de pré-busca:

a. pré-busca sempre: onde em toda referência à memória, um acesso à linha i, implica em uma pré-busca da linha i + 1,

b. pré-busca em ausências: onde a linha i + 1 só será considerada para pré-busca, se a referência à linha i foi devida a uma ausência.

c. pré-busca rotulada: onde cada linha é rotulada com um bit de referência, que é inicialmente zero. Quando a linha é utilizada por algum programa, esse bit torna-se um, sendo que é novamente zerado quando a linha é retirada do cache. Toda vez que o bit de referência de uma linha i, passa de zero para um, haverá uma pré-busca da linha i + 1.

Smith, em {Sm82}, compara os três tipos de pré-busca citados e conclui que a pré-busca sempre e a rotulada, re-

duzem em 50 a 90% a taxa de falta, em relação ao caso de não se utilizar pré-busca. No caso de pré-busca em ausências, o efeito é bem menor, sendo que a redução da taxa de falta é menos da metade que a dos casos de pré-busca sempre ou rotulada.

Algoritmos de Localização

A memória cache é, normalmente, associativa. Como memórias associativas muito grandes são caras e tendem a ficar mais lentas, o cache é organizado em várias memórias, cada uma dessas memórias é denomida um conjunto e o número de elementos que ela armazena é o tamanho do conjunto. Assim, uma determinada informação apenas precisa ser procurada em um dos conjuntos.

Os algoritmos de localização são utilizados para a determinação do conjunto onde está a informação procurada.

O algoritmo de localização comumente empregado é o de mapeamento do conjunto associativo, onde existe uma função (F) que mapeia o endereço da memória, fornecendo o conjunto. Assim, sendo r(i) o endereço da memória, o conjunto s(i) será obtido por:

$$s(i) = F(r(i))$$

A função de mapeamento mais simples e mais popular, é a denominada seleção de bit. O número de conjuntos (s) é escolhido como sendo uma potência de dois ($s = 2^k$). Existindo dentro de cada linha 2^j bytes, os j bits menos significativos determinam o byte dentro da linha. Os bits de j + 1 até j + k selecionam o conjunto, conforme mostrado na Figura II-11.

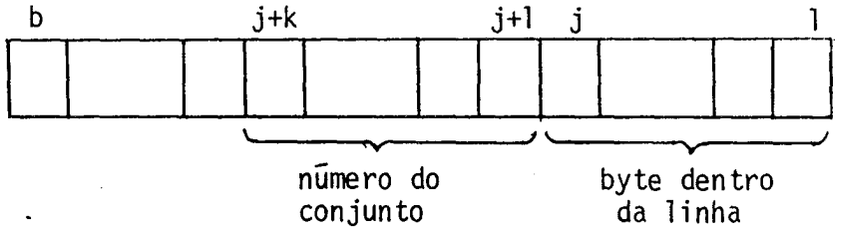


Figura II-11 - Função de Mapeamento por Seleção de Bit

Quanto ao número de elementos por conjunto, Smith, em {Sm82}, sugere um mínimo de quatro a oito para a obtenção de uma taxa de falta aceitável.

Algoritmos de Troca de Linha {Sm82, Ha83}

Quando alguma informação, que não estiver no cache, for requisitada e o cache estiver cheio, alguma linha do cache deve ser selecionada para ser substituída.

Os algoritmos de reposição são os mesmos utilizados para memória virtual.

Smith, em {Sm82}, compara o algoritmo LRU com o FIFO concluindo que a taxa de falta é, aproximadamente, 12% maior quando o algoritmo FIFO é utilizado, em relação à utilização do LRU. Cabe ressaltar no entanto, que o algoritmo FIFO ou o aleatório é muito mais fácil de ser implementado que o LRU.

Tamanho da Linha em Uma Memória Cache

Existem uma série de fatores que devem ser considerados para a determinação do tamanho da linha do cache.

Vantagens de linhas pequenas:

a. tempo para mover a linha da memória principal para o cache é menor,

b. contém um número menor de informações não necessárias.

Vantagens de linhas grandes:

a. se um número maior de informações serão realmente utilizadas, a busca em uma única vez é mais eficiente,

b. o número de linhas no cache é menor, sendo mais fácil gerenciar os conjuntos,

c. o número de elementos por conjunto do cache é menor facilitando a pesquisa associativa.

Quanto à variação da taxa de falta em função do tamanho da linha, depende da utilização das informações e da máquina considerada, pois, utilizando-se linhas grandes, o seu carregamento no cache é mais rápido, mas pode existir também um maior número de informações desnecessárias, poluindo a memória.

Os tamanhos de linhas utilizados variam bastante. Tem-se, por exemplo, que o IBM 3081 utiliza 128 bytes, o VAX

11/780 utiliza 8 bytes e o PDP 11/70 utiliza 4 bytes.

Tamanho do Cache

Considerando apenas a taxa de falta, conlui-se que quanto maior for o cache, menor será a taxa. No entanto, existem outros aspectos que devem ser considerados, entre eles o custo, o tamanho físico e o tempo de acesso.

Smith, em {Sm82}, sugere que a taxa de falta, possa ser calculada, aproximadamente, por:

$$m = a(k^b)$$

onde:

m é a taxa de perda,

a e b são constantes ($b < 0$),

k é a capacidade do cache, em kilobytes.

Smith chegou a esta formulação, através de resultados experimentais. Isto é interessante para que se tenha uma estimativa aproximada do desempenho do cache em projeto. Quanto aos tamanhos típicos para cache, em uso, pode-se citar desde caches de 64 k bytes (como o do NEC ACOS 9000), até cache de 4 k bytes (como o do VAX 11/750) ou de 1 k byte (como o do PDP 11/70).

Algoritmos de Atualização da Memória Principal {Sm82}

Quando a UCP executa alguma modificação, esta deve refletir também na memória principal, uma vez que o cache é apenas um local para armazenamento temporário.

Existem, basicamente, duas técnicas para proceder à variação da memória principal:

. cópia atrasada: onde a modificação é refletida, inicialmente, apenas no cache, e a modificação da memória principal pode ser feita mais tarde,

. cópia durante a modificação: onde a alteração é transmitida imediatamente para a memória principal.

Vários aspectos devem ser levados em consideração na comparação dos algoritmos de variação da memória principal:

a. tráfego na memória principal: a cópia atrasada, quase sem -

pre resulta em menos acessos à memória principal,

b. consistência do cache: se a cópia durante a modificação é utilizada, a memória principal sempre contém uma cópia atualizada de todas as posições de memória,

c. complexidade da lógica: para a cópia atrasada, necessita-se de um bit para indicar quando a linha deve ser reescrita na memória,

d. busca em escrita: no caso da cópia durante a modificação, pode-se ou não fazer uma busca em uma operação de escrita se for o caso. Para a cópia atrasada, sempre será necessário ter-se a linha referida no cache,

e. aclopadores: para a cópia atrasada, um acoplador é necessário para armazenar temporariamente a linha a ser copiada, para evitar interferências com a busca. No caso de cópia durante a modificação, é importante o acoplador para que a UCP não tenha que esperar que a cópia tenha terminado.
No acoplador deve existir o dado a ser armazenado e o endereço.

f. confiabilidade: como na cópia atrasada a memória principal não possui uma cópia atualizada do estado da memória, é necessário ter-se para o cache um código detector de erro. No caso de cópia durante a modificação, o cache pode ser corrigido pela memória principal, bastando portanto, que se tenha uma verificação de paridade.

Smith, em {Sm82}, compara os dois algoritmos e conclui que em termos de taxa de falta, a cópia durante a modificação sempre produz taxas muito maiores. Com respeito ao desempenho, não existe uma escolha muito clara, dependendo bastante da implementação de cada um.

A cópia atrasada é utilizada no IBM 3081, enquanto que a cópia durante a modificação é implementada no IBM 3033, no VAX 11/780 e no PDP 11/70.

Cache de Dados/Instruções

Uma outra estratégia para projeto de cache é a sua divisão em duas partes: uma para dados e a outra para instruções.

Fazendo-se a divisão, tem-se algumas vantagens, como por exemplo, dobrar o atendimento, uma vez que o cache pode a

gora servir dois requisitantes no tempo que só atenderia um. Tem-se também um tempo de acesso menor.

Como desvantagem desse tipo de implementação de memória cache, tem-se o problema da consistência e a ineficiência do uso da memória, uma vez que a fração de memória necessária para dados e instruções variam constantemente.

Smith, em {Sm82}, faz simulações e compara a taxa de falta para cache unificado, cache dividido em duas partes iguais para dados e instruções e cache com uma divisão ótima. Baseado nos resultados experimentais, Smith considera que a taxa de falta pode aumentar significativamente se o cache tem algum tipo de divisão, sendo que o aumento depende da carga de trabalho.

Por outro lado, Hawk, em {Ha83}, argumenta que como as instruções estão quase sempre presentes na parte de instruções, tem-se uma taxa de acertos quase perfeita, permitindo à UCP trabalhar perto de uma velocidade máxima.

Cache do Usuário/Supervisor

O frequente chaveamento entre o estado do usuário e do supervisor resulta, em alguns sistemas, em uma alta taxa de falta, pois, o cache é frequentemente recarregado. Uma forma de resolver o problema é dividir o cache em duas partes: uma parte do usuário e uma parte do supervisor.

Esse tipo de divisão apresenta dois problemas:

- é improvável que a taxa de falta diminuirá, pois, muitas faltas podem ocorrer no estado do supervisor, uma vez que o cache será menor. Além disso, após uma interrupção, não necessariamente voltar-se-á ao mesmo programa que estava sendo executado antes da interrupção,
- as informações usadas pelo usuário e pelo supervisor não são inteiramente distintas.

Início Quente e Início Frio - Efeito da Multiprogramação

Em sistemas com multiprogramação, a frequência de chaveamento das tarefas, tem grande influência sobre a taxa de falta {Sm82}. Isto porque, ao ocorrer um chaveamento, o novo processo que será rodado pode conter nenhuma de suas linhas no cache,

ocasionado muitas faltas. A taxa de falta medida no caso (cache inicialmente vazio de linha do processo) é denominada partida a frio. Em caso contrário (cache inicialmente cheio) é denominada partida quente.

Smith, em {Sm82}, faz alguns experimentos relacionando a taxa de falta com o tamanho da memória e o intervalo de chaveamento (que foi levado em consideração utilizando-se o número de referência à memória por cada processo, antes de ser chaveado (Q)).

Com os resultados do experimento, conclui-se que com grandes valores de Q, a taxa de falta diminui. Isto porque, apesar do cache estar inicialmente vazio, logo estará cheio de linhas do processo, que serão utilizadas várias vezes, diminuindo a taxa de falta, enquanto que para pequenos valores de Q, quando um processo é reiniciado podem existir algumas linhas suas no cache mas, estas e as outras que serão trazidas serão pouco utilizadas, ocasionando uma taxa de falta maior.

Smith, em {Sm82}, propõe algumas soluções para o problema:

. aumentar Q,

. fazer um cache maior, para que vários programas possam manter suas linhas no cache,

. dar preferência a execução de tarefas que contenham várias linhas no cache,

. fazer uma restauração no conjunto de trabalho dos processos,

. criar múltiplos caches,

Consistência de Múltiplos Caches

Tem-se, em alguns sistemas, vários processadores. Se cada processador possuir um ou mais caches, é necessário manter-se uma consistência entre esses caches, pois algumas vezes uma informação pode ser utilizada por vários processadores, estando portanto em vários caches. Assim, quando uma modificação for feita por um processador, isso deve se refletir em todos os lugares onde a informação estiver armazenada. Existem várias propostas para solução do problema, sendo que algumas são apresentadas a seguir {Sm82}:

a. cache compartilhado: onde todos os processadores contêm um único cache, conforme ilustrado na Figura II-12. Normalmente, esta solução não é muito adequada, devido à amplitude de entrada de um único cache, não ser suficiente para suportar vários processadores. Além disso, tem-se também um maior tempo de acesso, pois o cache não é denso o suficiente para suportar vários processadores. Apesar desses problemas, o UNIVAC 1100/80 permite que duas UCPs compartilhem o mesmo cache. VanAken, em {Va82}, aborda também o problema da não minimização de fluxo de informações no duto.

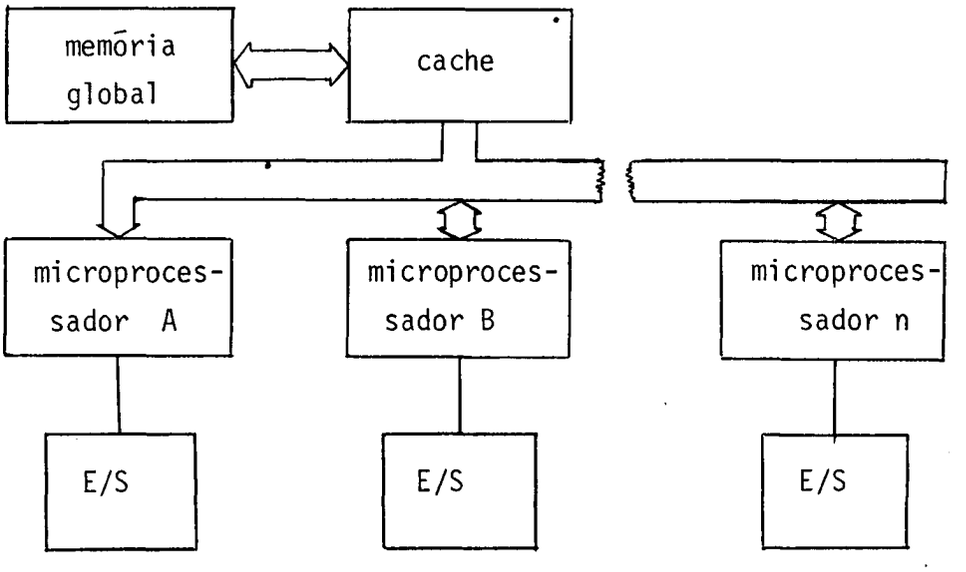


Figura II-12 - Sistema com Vários Processadores e Cache Compartilhado

b. escrita espalhada: onde sempre que a UCP faz uma escrita para o cache, esta deve ser enviada a todos os caches do sistema. Se a informação estiver em algum outro cache, ela deve ser variada ou invalidada. Neste caso, tem-se um cache para cada UCP conforme ilustrado na figura II-13. A invalidação tem sido mais aceita, pois, evita cruzamentos de informações (UCP A varia seu cache e o da UCP B, enquanto UCP B, varia seu cache e o da UCP A), além de minimizar

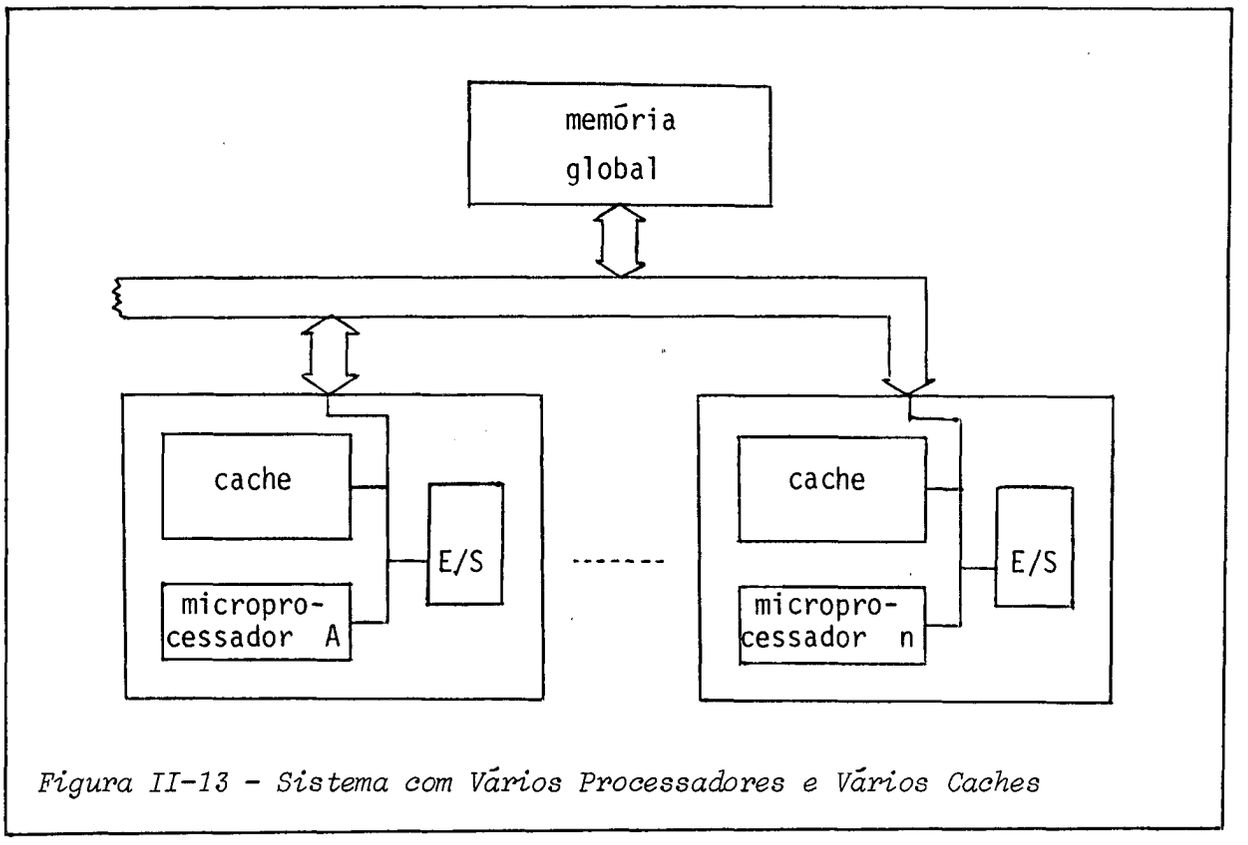


Figura II-13 - Sistema com Vários Processadores e Vários Caches

transferências de dados.

As informações a serem invalidadas são colocadas em uma pilha de endereços a serem invalidados (BIAS) que tem uma alta prioridade nos ciclos para o cache. Esta técnica funciona muito bem para sistemas com até dois processadores, apresentando uma degradação para um número maior. Para esses casos, Smith, em {Sm82}, propõe uma memória filtradora de BIAS (BFM), que seleciona as requisições de invalidação de um mesmo bloco, contidas na BIAS, melhorando o desempenho do sistema,

c. controle por software: onde um controle por software pode garantir a consistência, mantendo as informações que podem ser compartilhadas por vários processadores apenas na memória principal. Isto é, não permitir que estas informações sejam armazenadas no cache,

d. métodos de diretórios: onde é mantido um diretório de todas as linhas da memória principal, que indica quais linhas são compartilhadas e qual cache contém cada linha. Supondo que existam "k" caches, cada linha da memória principal conterá k + 1 bits de controle. Os bits de 1 a k serão "1", se o cache correspondente a ele contiver a linha em questão, e será "0" em caso contrário. O bit k + 1 de cada linha con.

terá "1" se esta linha for modificada por algum processador, sendo que a memória principal e os outros caches ainda não contêm a nova cópia.

Cada cache com a correspondente UCP, contêm um bit-privado, para cada linha que será mudado para "1", quando houver uma modificação na linha executada por esta UCP. Esse bit só se torna "1" quando houver a modificação e o bit k + 1 correspondente a esta linha na memória principal for "0". Havendo a modificação, este bit é modificado para "1". Assim, quando a UCP A, quiser ler a linha i, existem duas possibilidades:

i. $(k + 1)_i = 0$: nesse caso, basta enviar a linha i para a UCP A e fazer o bit correspondente a A da linha i igual a "1",

ii. $(k + 1)_i = 1$: buscar a linha i no cache que contêm a cópia modificada; atualizar a memória principal; invalidar a cópia no cache modificador; enviar a linha i para a UCP A; $(k + 1)i = 0$.

Um problema associado a esse método é o alto custo de uma falta, uma vez que são necessários acessos à memória, além de consultas a outros caches. Pode ocorrer também, problemas com excessos de entrada e saída, sendo necessário um acoplador de Entrada/Saída.

Para minimizar a sobrecarga no método dos diretórios, pode-se pensar em se ter dois caches, sendo que apenas um poderá conter dados compartilhados.

Dos métodos descritos, os mais práticos são o da memória filtrante para a pilha de endereços a serem invalidados e o do diretório, embora também apresentem certos problemas no desempenho.

Amplitude de Entrada da Cache

Corresponde à taxa através da qual os dados podem ser lidos e escritos no cache.

A amplitude de entrada deve ser suficiente para suportar a taxa de execução de instrução e entrada/saída propostas.

Cache em Vários Níveis

Quando um cache começa a ficar muito grande, po-

de-se tentar minimizar os problemas resultantes, dividindo-o em dois níveis: um pequeno, rápido e mais caro por byte, e um maior, um pouco mais lento.

3.5. *Exemplo de Implementação de Cache - Família VAX 11* {Di82}

A UCP do VAX 11/750 prove três sistemas cache: o cache principal, o de conversão de endereço e um de instruções.

O cache principal contém 4 k bytes, mapeados diretamente e com escrita na memória principal, durante as modificações. Fornece ao processador um acesso a dados com alta velocidade, pois armazena os endereços, dados e instruções frequentemente referenciados. Tem uma taxa de acerto de aproximadamente 90%.

O cache de instruções tem um acoplador que habilita a UCP a buscar e decodificar a próxima instrução, enquanto a instrução corrente é executada.

O cache de conversão de endereço contém a conversão dos 512 endereços físicos mais utilizados. Essas 512 localizações são divididas em duas partes, sendo que 256 são para o espaço do sistema e 256 para o espaço dos processos.

O sistema do cache do VAX 11/780 é semelhante ao do VAX 11/750. As diferenças fundamentais são:

a. o tamanho do cache (8 k bytes para o VAX 11/780),

b. a memória cache do VAX 11/780 é associativa por dois conjuntos de caminhos,

c. taxa de acertos maior que 95%,

d. acoplador de conversão de endereços com 128 conversões de endereço virtual para físico, sendo 64 para o espaço do sistema.

3.6. *Circuitos Integrados Que Suportam Gerenciamento de Memória*

Em sistemas que utilizam memória virtual, ou mesmo em sistemas mais simples é necessário um gerenciador de memória, que controle todo mecanismo de conversão de endereços, alocação e relocação dinâmicas e proteção. O mecanismo de gerenciamento pode ser executado por uma unidade externa, localizada entre o processador e a memória e periféricos.

A interligação entre as três unidades é feita através dos dutos de endereços, dados e controle, como pode ser visto na Figura II-14.

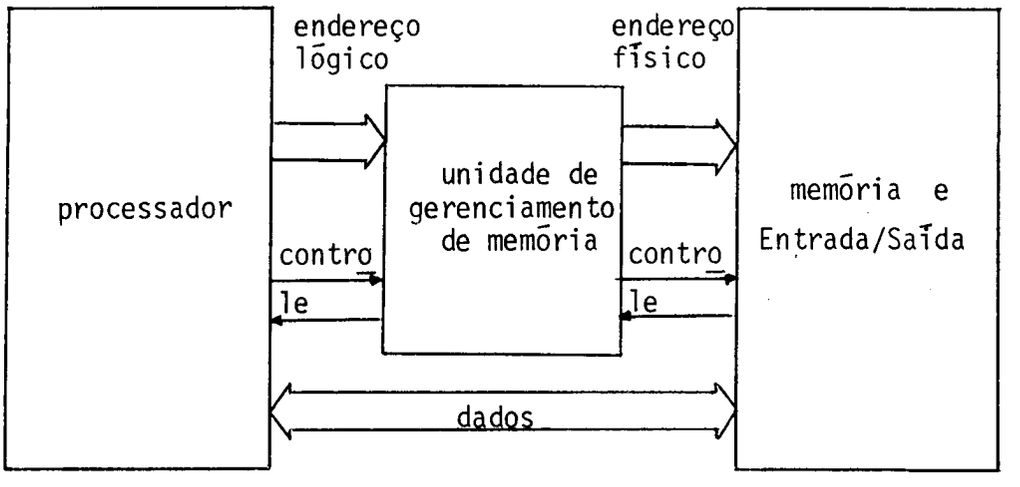


Figura II-14 - Sistema com Gerenciamento de Memória

O mecanismo de gerenciamento de memória, pode ser implementado utilizando-se alguns circuitos integrados disponíveis que executam as funções de uma unidade de gerenciamento de memória. Entre esses integrados, citam-se o MC 68451 da Motorola {Zo81, Mo83}, o iAPX 286/10 da Intel {In82}, o Z8010 da Zilog {St81, Ti85} e o NS 16082 da National Semiconductor Corporation {Ti85}.

Nos próximos ítens, pretende-se apresentar algumas características do MC 68451 e do iAPX 286/10.

3.6.1. *MC 68451*

O MC 68451 é um circuito integrado em larga escala, que combinado com uma UCP e a memória física, constitui um mecanismo de gerenciamento de memória, cujo esquema é apresentado na Figura II-15.

O MC 68451 proporciona ao sistema operacional a capacidade de alocar, controlar e proteger um espaço de endereçamento de 16 M bytes.

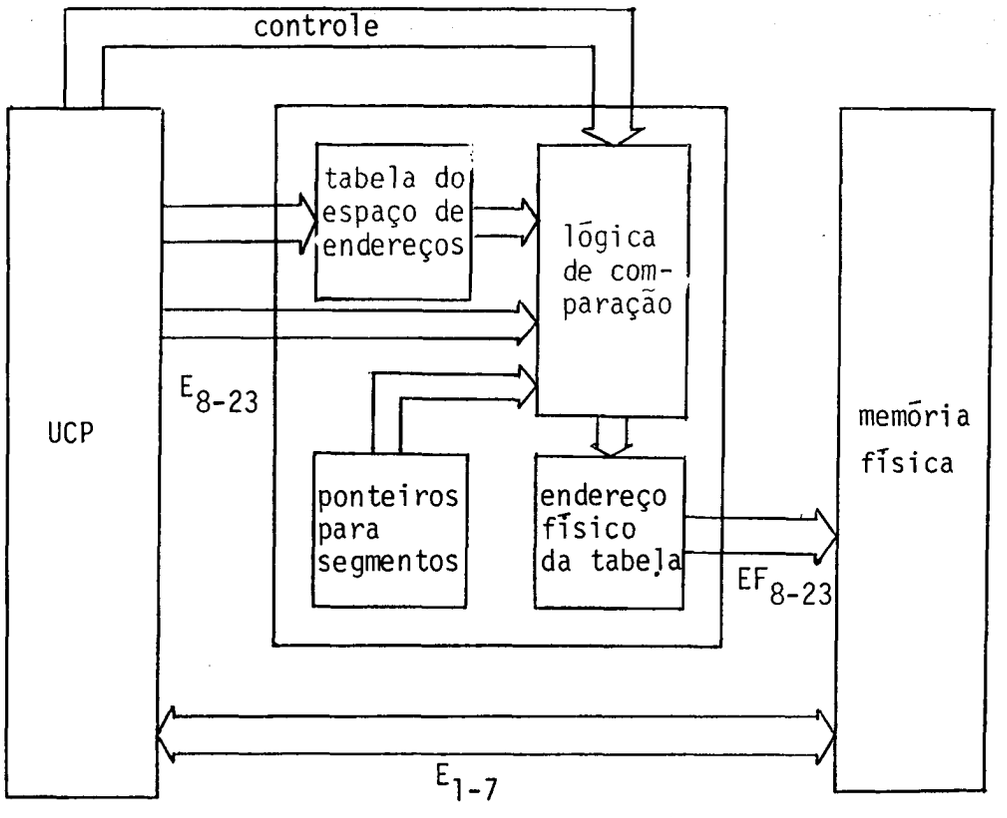


Figura II-15 - Mecanismo de Gerenciamento de Memória Utilizando o MC 68451

As principais características dessa unidade de gerenciamento de memória são:

a. separar o espaço de endereçamento entre espaço do usuário e do supervisor e dentro desses, em espaço de dados e de programa,

b. proporcionar proteção contra escrita,

c. suportar paginação e segmentação,

d. suportar trinta e dois segmentos de tamanho variável,

e. possibilitar a associação de várias unidades para aumentar o número de segmentos,

f. permitir o compartilhamento de segmentos.

Pode-se dividir essa unidade de gerenciamento de memória em três unidades:

a. tabela de espaço de endereços: que contém os números do espaço de endereços, que servem para identificar o processo

que está fazendo o acesso,

b. descritores dos segmentos: compostos de seis registradores, contendo as informações necessárias para descrever o segmento, tais como: o endereço básico para o espaço físico e lógico, estado do segmento, bits sobre os antecedentes de utilização e modificação dos segmentos,

c. lógica de comparação: onde a tabela do espaço de endereço e os descritores dos segmentos, são utilizados para fazer a conversão do endereço lógico para físico, empregando apenas acessos aleatórios à memória e operações lógicas, tornando a conversão mais rápida. Faz também comparações necessárias para verificar se o acesso está correto.

O método de conversão utilizado no MC 68451, torna a operação de chaveamento rápida e eficiente.

3.6.2. *iAPX 286 (80286)*

O iAPX 286 é um microprocessador de alto desempenho e com capacidade de fornecer sistemas multiusuários e multitarefas. Suporta ainda um gerenciador de memória, capaz de mapear até 1 G Byte de espaço de memória virtual por tarefa, em 16 M bytes de memória física. As principais características desse microprocessador, em relação ao gerenciamento de memória são:

a. quatro níveis de proteção de memória,.

b. suporte para memória virtual,

c. gerenciamento de memória integrado.

O iAPX 286/10 é composto por quatro unidades:

a. unidade do duto,

b. unidade de instruções,

c. unidade de execução,

d. unidade de endereçamento.

A unidade de endereçamento é responsável pela conversão do endereço virtual para o endereço físico. O endereço virtual é composto de 32 bits, sendo que os 16 bits de mais alta ordem especificam o segmento e, os 16 bits de mais baixa ordem compõem a variação dentro do segmento.

Utilizando a tabela de descritores e os bits que especificam o segmento, obtém-se o endereço base, que somando à variação dentro do segmento, fornece o endereço físico.

As entradas da tabela de descritores dos segmentos definem o uso da memória. Tipos especiais de descritores definem novas funções para transferência de controle e chaveamento de tarefas.

Os descritores contém ainda:

a. tamanho do segmento (de 1 a 64 k bytes),

b. tipo de acesso (apenas leitura, leitura/escrita, apenas execução e execução/leitura),

c. presença na memória (para sistemas com memória virtual).

Assim, o microprocessador iAPX 286/10, além de desempenhar suas funções como um microprocessador de alto desempenho, é capaz de possibilitar a implementação de sistemas multiusuários e multitarefas, suporta ainda todo o gerenciamento de memória, incluindo capacidade de mapeamento de memória virtual e proteção.

4. *CONSIDERAÇÕES FINAIS*

Com base na revisão bibliográfica apresentada, destacam-se os seguintes pontos:

a. A conceituação geral de sistemas distribuídos é bastante diversificada, com diversos pontos de vista influenciando a classificação dos sistemas. Uma visão mais moderna, inclui os sistemas com microprocessadores dedicados {Fa83}, como distribuídos,

b. A conceituação sobre memória virtual considerada, permite a formação de uma visão geral dos aspectos envolvidos na definição e implementação de sistemas com memória virtual,

c. O estudo sobre cache, destaca as principais características a serem observadas na definição e implementação de um sistema com cache, bem como permite o estabelecimento de uma analogia entre o cache e a memória virtual,

d. As considerações sobre os circuitos integrados que suportam gerenciamento de memória, fornece informações sobre as características gerais destes integrados, indicando os casos em que podem ser convenientemente utilizados.

Assim, através dos pontos abordados na revisão bibliográfica e dos objetivos traçados para o módulo de memória, tem-se estabelecida a base necessária para a conceituação do módulo de memória virtual que é apresentada no capítulo III.

CAPÍTULO III

CONCEITUAÇÃO DO MÓDULO

1. *CONSIDERAÇÕES INICIAIS*

Neste capítulo são apresentadas as características gerais do módulo de memória e a arquitetura proposta. Essas características baseiam-se nas necessidades de um módulo de memória virtual, para um sistema distribuído funcional e no estudo bibliográfico desenvolvido que é apresentado no capítulo II. A descrição e as justificativas das características adotadas, são apresentadas nos capítulos IV e V.

O módulo de memória definido neste trabalho faz parte de um sistema distribuído funcional (SDF), definido em {Sa85}. A arquitetura da configuração básica do sistema é apresentada na figura III-1.

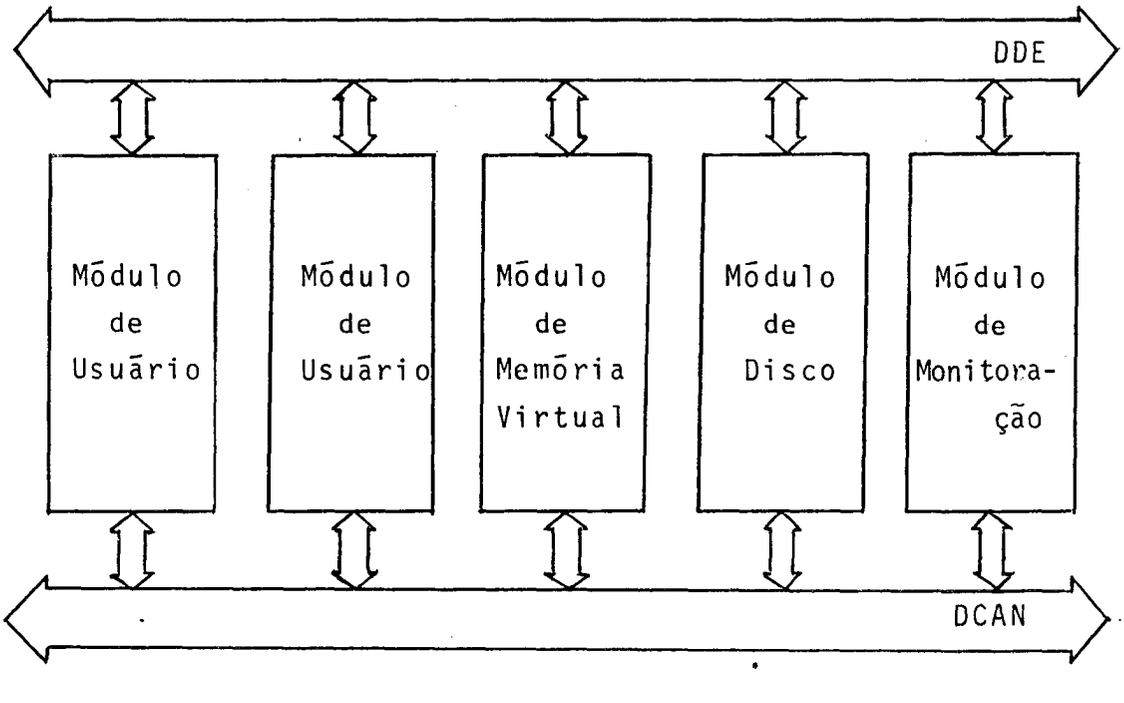


Figura III-1 - Configuração Básica do SDF

O SDF utiliza na comunicação entre os módulos dois dutos; um de comunicação de alto nível (DCAN), por onde circulam os comandos de alto nível do sistema, e outro de comunicação de baixo nível (DDE), por onde circulam dados e endereços. Assim, a comunicação entre o módulo de memória e os outros módulos do sistema, é feita através desses dutos, conforme estipulado em trabalhos prévios {Sa85}.

O SDF é composto por módulos funcionais onde, para cada módulo está pré-estabelecido um conjunto de tarefas ou funções. A existência desses módulos permite uma grande flexibilidade na construção e expansão do SDF.

O módulo de memória virtual permite uma implementação eficiente de memória virtual para todos os módulos do SDF, através da centralização do gerenciamento de memória dos módulos do sistema. O módulo utiliza ainda algumas idéias de cache e outros tipos de implementações que visam melhorar seu desempenho.

A presença desse módulo de memória no SDF permite uma simplificação nos outros módulos do sistema e traz uma maior facilidade na implementação do gerenciamento de memória.

Com o módulo de memória, os demais módulos do SDF podem contar com uma memória que lhes oferece diversos recursos, sem incorrer no ônus de uma arquitetura ou sistema operacional complexos.

A simplificação na implementação do gerenciamento da memória é obtida com a presença de um processador e um sistema de armazenamento secundário dedicados ao módulo de memória.

Um processador dedicado permite a realização de tarefas paralelas a outros processamentos (atualização de tabelas, manutenção de estatísticas, etc..), o que torna a liberação do dado requisitado mais rápida, melhorando o desempenho do módulo.

Uma tarefa que é bastante simplificada, devido ao processador dedicado, é o tratamento de ausência de páginas ocorridas durante a execução de um programa. Nesse caso, o processador do módulo de memória faz todo o tratamento da ausência de página, enquanto o processador do módulo que está solicitando o dado, deve unicamente esperar, pelo dado requisitado. Desta forma, não é necessário ter-se uma rotina para salvar o estado do processador no instante em que ocorre a ausência, e após solucioná-la, ter-se uma outra rotina para restaurar o estado do processa

dor.

Um sistema de memória secundária dedicado ao módulo, torna mais fácil e rápido trocas de páginas entre memórias principal e secundária, uma vez que não é necessário bloquear-se o duto para que a tarefa seja concluída. Além disso, um sistema de armazenamento secundário dedicado a implementação de uma memória virtual, evita saturação no módulo de disco do SDF e no duto do sistema.

A memória principal do módulo de memória é dividida em duas partes:

. Memória do SDF - destinadas aos outros módulos do SDF, sendo dividida por processo, que corresponde a cada programa ativo em cada módulo do sistema.

. Memória local - destinada ao armazenamento das tabelas, rotinas necessárias para o funcionamento do módulo de memória e demais estruturas de dados de apoio. Nesta parte da memória estão armazenadas todas as tabelas, com informações sobre as páginas de cada processo dos módulos do SDF. Podem ter acesso a esta memória o microprocessador local e os módulos do SDF, desde que obedecida a proteção estipulada.

O módulo de memória apresentado neste capítulo, além de permitir aos demais módulos do sistema um endereçamento virtual, mantendo todas as tabelas e executando todas as rotinas necessárias para um perfeito desempenho da memória virtual, apresenta algumas implementações adicionais que visam tornar mais rápido o acesso aos dados requisitados.

Na Seção 2 deste capítulo são apresentadas as características gerais do módulo proposto e em especial as características dos três níveis de armazenamento que o módulo suporta: cache, memória principal e memória secundária. Na seção 3 tem-se a arquitetura do módulo. O esquema lógico do hardware é apresentado na seção 4 e a estrutura do software na seção 5. Nestas seções são feitas algumas sugestões para espaço de memória real, virtual, número de bits a serem utilizados em tabelas e outras características que podem ser modificadas dependendo das disponibilidades no momento da implementação.

2. *CARACTERÍSTICAS GERAIS DO MÓDULO*

Nesta seção são apresentadas as características gerais do módulo de memória virtual, destacando-se a divisão da memória em local e do SDF e a divisão do espaço de endereçamento da memória do SDF, em três níveis.

O espaço de endereçamento do módulo é dividido através do bit mais significativo do endereço, como mostrado na figura III-2, sendo que cada região tem características bastante diferentes.

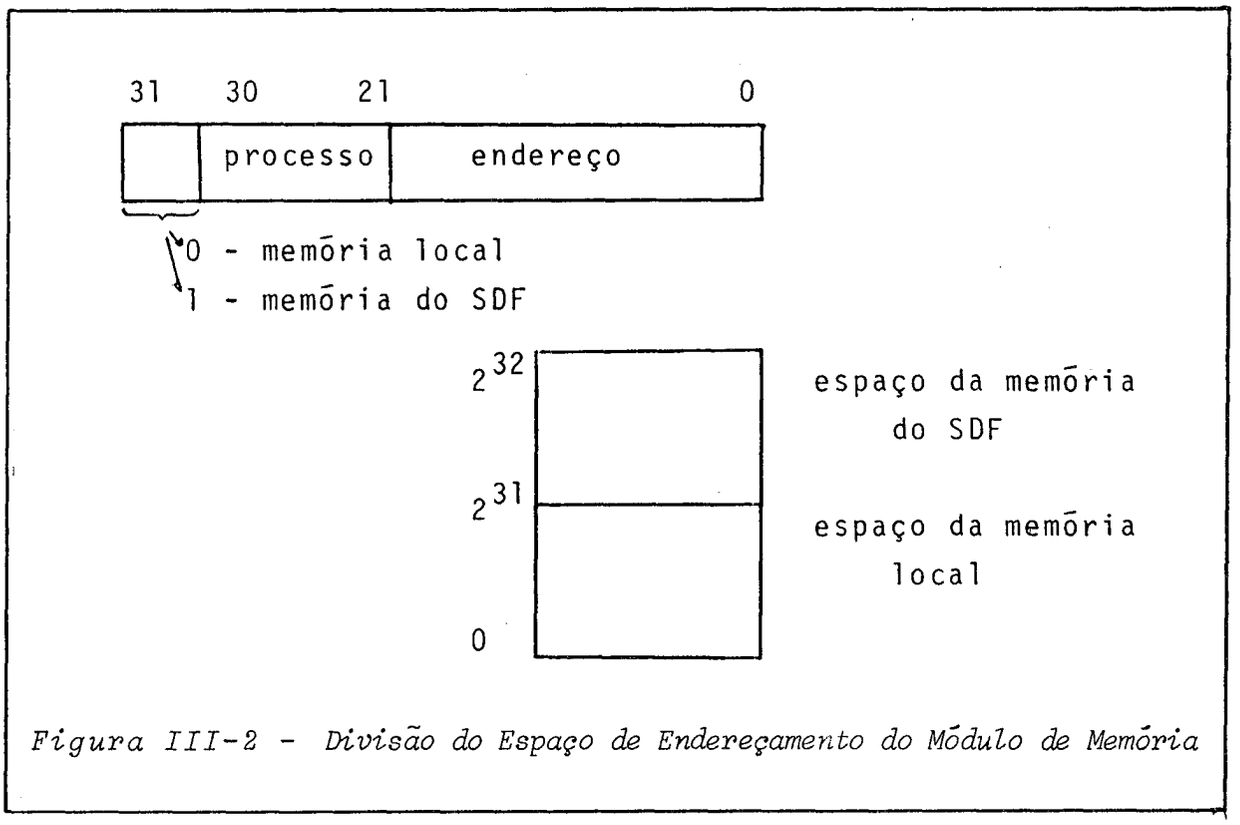


Figura III-2 - Divisão do Espaço de Endereçamento do Módulo de Memória

A. *Memória Local*

Destinada ao armazenamento das estruturas e rotinas utilizadas no módulo, podendo a ela ter acesso o processador local ou outros módulos.

Na memória local não será implementado endereçamento virtual, evitando-se que ocorra uma ausência de página durante a execução de uma rotina. Isso torna bastante rápido o acesso às estruturas de dados, sem ocasionar problemas, uma vez que não é necessário um grande espaço de endereçamento para a memória local.

O endereço local consta do bit mais significativo em zero, seguido do processo que está efetuando o acesso e do endereço a que se quer ter acesso, conforme Figura III-3.

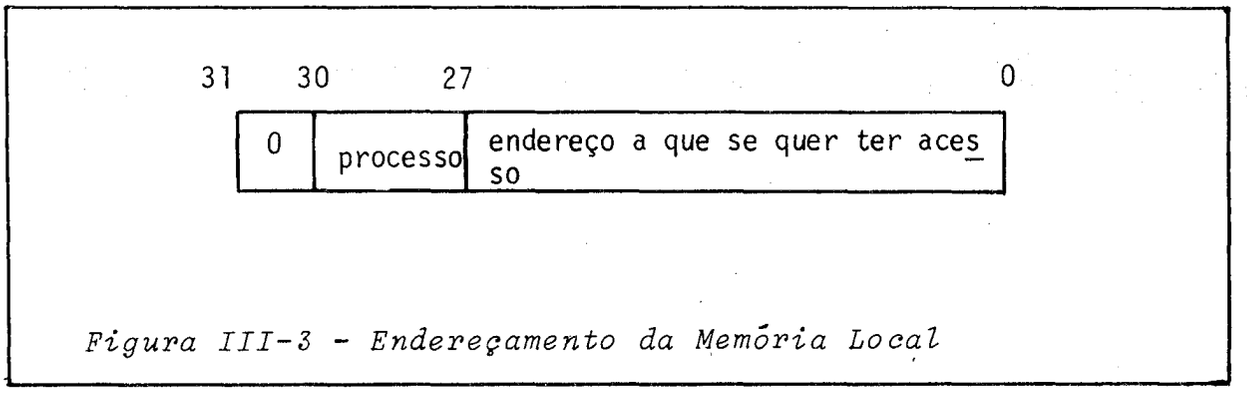


Figura III-3 - Endereçamento da Memória Local

A proteção da memória local é implementada apenas para acessos feitos por outros módulos. O controle é feito por processo, sendo que a região de memória de cada processo pode ser acessível para leitura, para leitura e escrita ou não ser acessível.

B. *Memória do SDF*

Destinada ao armazenamento de dados, códigos e endereços dos módulos do SDF, podendo a ela ter acesso qualquer módulo.

O endereçamento virtual implementado, propicia um grande espaço de endereçamento distribuído em três níveis: cache, memória principal e memória secundária.

Pode-se ter acesso ao espaço de endereçamento virtual do SDF, através de um endereço virtual de 32 bits, sendo que o bit mais significativo deve ser igual a 1. O espaço destinado ao SDF é dividido em páginas de 1024 palavras endereçáveis por 32 bits do endereço virtual (EV<10:20>). O endereçamento da palavra dentro da página é obtido pelos 10 bits menos significativos do endereço virtual (EV<0:9>).

Os bits restantes (EV<27:30>), indicam o processo que está tendo acesso ao módulo de memória.

Assim, o módulo proposto permite um endereçamento virtual de até 2 G bytes, divididos por 16 processos. A figura III-4 mostra a formação do endereço virtual para o espaço de memória do SDF.

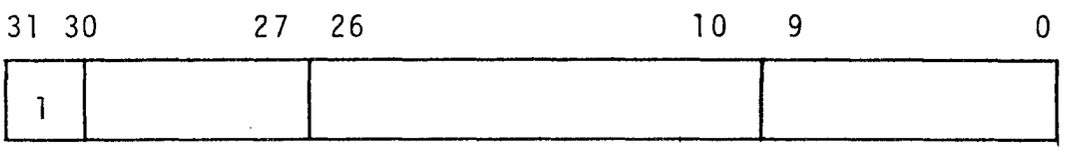


Figura III-4 - Endereço Virtual da Memória do SDF

A proteção da memória do SDF é feita por página e subdividida em 3 tipos:

- página acessível só para leitura
- página acessível só para escrita
- página acessível para leitura e escrita.

A proteção é definida na tabela de páginas virtuais, que é detalhada na seção 5 deste capítulo. A implementação da proteção é feita através do hardware, para tornar possíveis verificações rápidas.

A memória do SDF é distribuída em três níveis de dispositivos de memória com diferentes capacidades de armazenamento, custo e velocidade. O cache bastante rápido (tempo de acesso da ordem de 100 nano segundos), no entanto bem pequeno. A memória principal com tempo de acesso entre 200 nano segundos e 350 nano segundos e a memória secundária, cujo acesso é bastante lento (vários milisegundos), mas com uma grande capacidade de armazenamento.

As características gerais desses três níveis de armazenamento são apresentadas a seguir:

B.1. *Cache*

O objetivo do cache é aumentar a velocidade de obtenção de um dado, através da implementação de memórias pequenas e rápidas, cujo acesso é feito utilizando-se o endereço virtual. Dessa forma evitam-se acessos à memória principal, e a determinação do endereço real.

O cache é dividido por processo, tendo-se um espaço de 64 bits (4 palavras) para cada processo. Assim, a cada acesso, seleciona-se um cache através dos bits do endereço virtual que especificam o processo (EV<27:30>).

A divisão do cache por processos, evita uma diminuição na taxa de acertos, quando ocorre uma diversificação muito grande nos processos que tem acesso ao módulo de memória.

Apesar do cache ser dividido por processo, é localizado no módulo de memória, para que não seja necessária a passagem de 64 bits de dados pelo duto, em cada alteração do cache.

O endereço nesse 1º nível de armazenamento é modificado, toda vez que se tem uma leitura e o dado requisitado não está no cache. Nesse caso o cache é preenchido com o dado do endereço requisitado e mais 3 palavras de posições de memória adjacentes. Esses dados são obtidos com um acesso à memória principal, utilizando o endereço real do dado, sem considerar os 2 bits menos significativos.

As escritas são feitas diretamente na memória principal e o cache é atualizado sempre que a modificação for feita em um endereço que está presente no cache. Tem-se portanto, uma atualização durante a modificação, mantendo-se cache e memória principal sempre consistentes.

B.2. *Memória Principal*

A memória principal do SDF é organizada em quatro blocos de igual tamanho, que são endereçados conjuntamente. A palavra de cada um desses blocos é de 2 bytes.

Como o endereçamento é igual para os quatro blocos, se todos forem selecionados, tem-se uma palavra de 8 bytes (64 bits) na saída da memória principal. Isso facilita a modificação do cache, que pode ser feita com um único acesso à memória principal.

Para acessos a uma palavra (2 bytes) deve-se selecionar (através do selecionador de pastilha) qual dos blocos contêm a posição desejada, utilizando-se os dois bits menos significativos do endereço real. Esse procedimento é detalhado na seção 3 do capítulo IV.

A organização da memória principal do SDF pode ser observada na figura III-5.

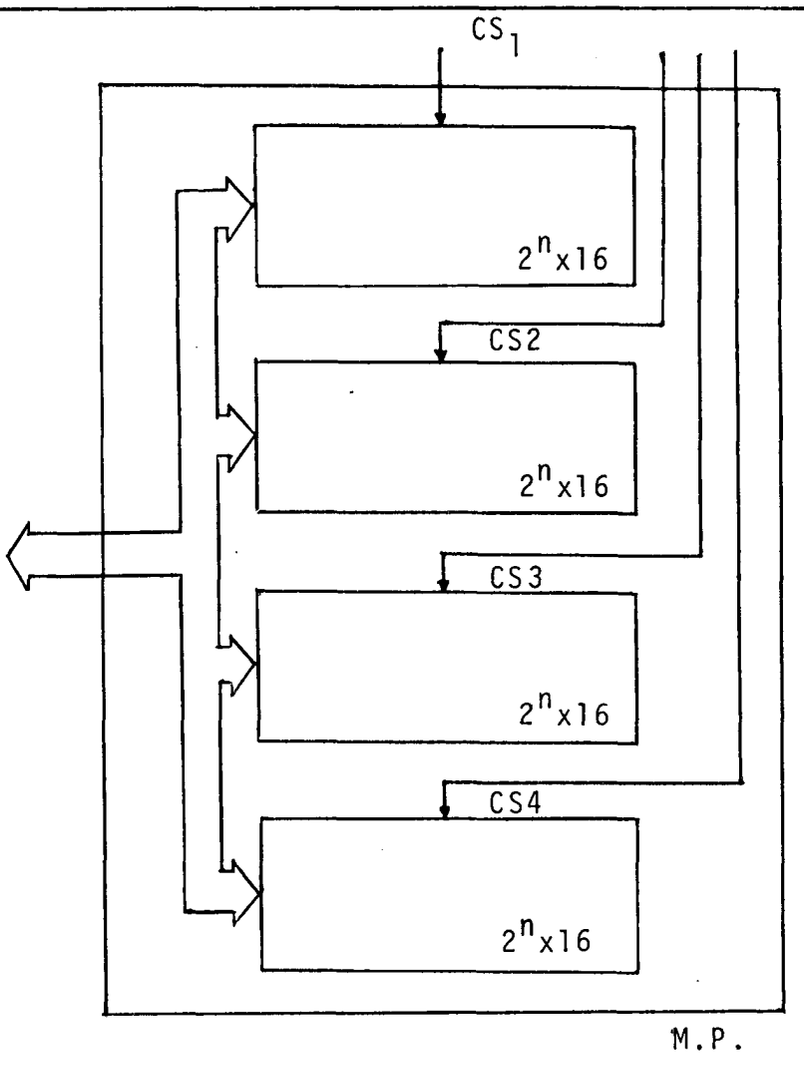


Figura III-5 - Organização da Memória Principal no SDF

A memória principal é dividida, inicialmente, por processo. Isto é, cada processo tem um espaço de memória reservado, cujo tamanho pode variar, dependendo da função do módulo. Apesar da memória ser dividida por processo, é possível, através do software, utilizar o espaço de memória dos processos que não estão ativos. Nesse caso, se um processo que está tendo sua área utilizada for ativado, as páginas que estão naquele espaço de memória são transferidas para a memória secundária. Todo este mecanismo é comandado pelo processador ao módulo de memória, ficando transparente para o usuário.

Dentro deste nível de memória, tem-se ainda uma subdivisão:

. Acessos à última página que teve um acesso para leitura.

Nesse caso, o endereço real da página está armazenado em um registrador, não sendo necessário um acesso à memória local para a determinação do endereço real, tornando a operação mais rápida.

. Acessos a outras páginas

Para os caso em que o endereço real não estiver no registrador, deve-se fazer um acesso à memória local e, utilizando-se o endereço virtual e a tabela de páginas virtuais, determinar o endereço real.

Após determinado o endereço real da página, basta somar a ele o deslocamento (EV<0> a EV<9>), para obter-se o endereço ao qual será feito um acesso na memória do SDF.

É importante observar que todos os mecanismos para obtenção de um dado que está no cache ou na memória principal, não dependem do processador, o que torna o acesso mais rápido.

B.3. *Memória Secundária*

O terceiro nível de memória é o sistema de armazenamento secundário, onde estão todas as páginas que não cabem na memória principal.

O mapeamento da memória secundária é feito diretamente a partir do endereço virtual, facilitando os acessos à ela.

Na implementação da memória virtual, desenvolveu-se diversas idéias, visando a obtenção de um melhor desempenho do módulo de memória. Nesse sentido, podem-se destacar as seguintes características:

. As páginas são guardadas na memória secundária antecipadamente, enquanto o processador está livre.

. É feita uma estatística da retirada de páginas da memória secundária, para que páginas a que se tenham tido acessos, tenham menor prioridade no retorno à memória secundária.

. Para evitar o bloqueio do duto, quando o endereço requisitado não estiver na memória principal, o duto é liberado, en-

quanto é feito o acesso à memória secundária.

. O software necessário para a troca de páginas entre memórias principal e secundária é dividido em duas partes:

- software antes da liberação do dado: Inicialmente executa-se o software necessário para a obtenção do dado requisitado.
- software após a liberação do dado: Após liberar o dado, em paralelo com outros atendimentos que não dependem do processador, executa-se o software para atualização das estruturas de dados necessárias para a manutenção do endereçamento virtual.

Essas características do gerencimaneto de memória virtual são importantes, devido à lentidão de acessos à memória secundária, tornando valiosa toda a tentativa de se evitar ou minimizar os acessos à memória secundária.

3. *ARQUITETURA DO MÓDULO*

Nesta seção é apresentada a arquitetura do módulo de memória, destacando-se o objetivo de cada bloco.

A figura III-6 mostra a arquitetura proposta.

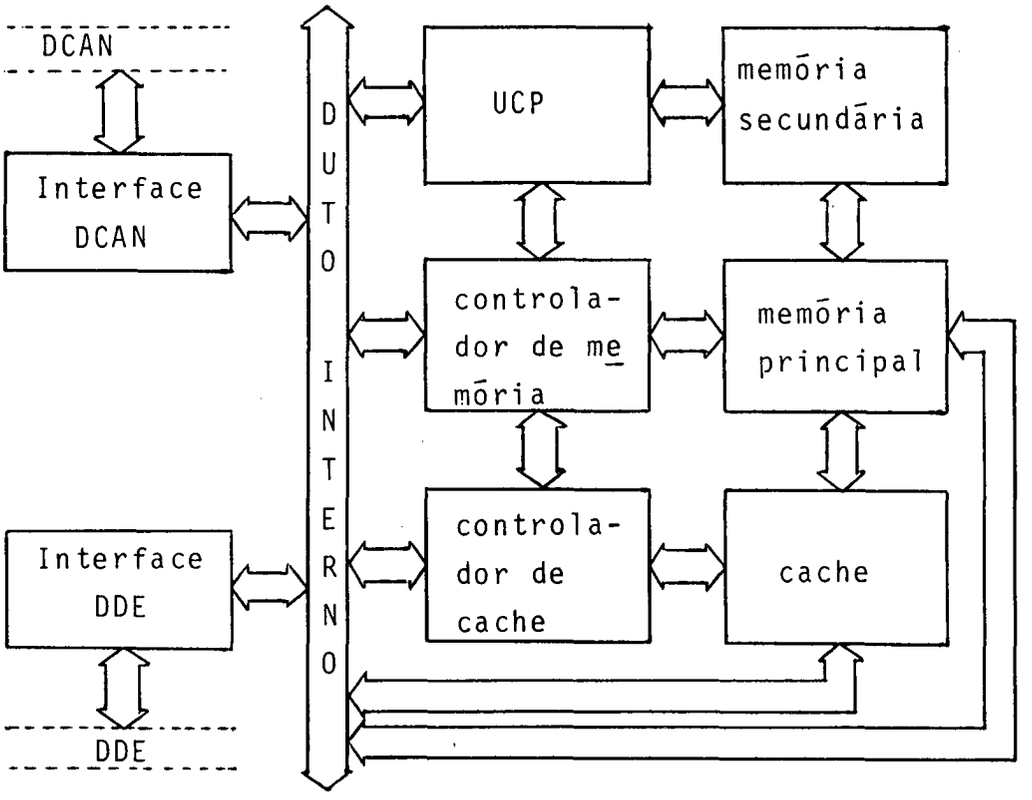


Figura III-6 - Arquitetura do Módulo de Memória

O módulo de memória, quando recebe um pedido de acesso, deve determinar em que nível de armazenamento está o endereço requisitado.

O bloco denominado controlador do cache, tem como objetivo verificar se o endereço requisitado está ou não no cache. Em caso afirmativo, envia um sinal (Sc) ao controlador de memória para que seja interrompida a busca nos outros níveis de memória.

Ao mesmo tempo em que o controlador do cache verifica se o endereço requisitado está no cache, o controlador de

memória verifica se esse está na memória principal. Se o dado não estiver na memória principal e o controlador de memória não receber um sinal do controlador de cache, indicando que o endereço já foi encontrado, é acionada a memória secundária, para que a página contendo o endereço procurado, seja transferida para a memória principal.

Os blocos do cache e da memória principal, como já destacado na seção anterior, constituem os dois primeiros níveis de armazenamento do módulo de memória, sendo que o cache apresenta um tempo de acesso bem baixo, mas com uma capacidade de armazenamento reduzida e a memória principal apresenta uma capacidade maior com um tempo de acesso um pouco mais elevado.

Para os casos em que forem necessários acessos à memória secundária utiliza-se um processador, dedicado ao módulo de memória para fazer todo o gerenciamento da transferência de página, entre as memórias principal e secundária.

O sistema de memória secundária emprega um disco Winchester, com razoável capacidade de armazenamento, tendo no entanto um tempo de acesso bastante grande, quando comparado com os outros dois níveis.

O módulo de memória apresenta ainda dois blocos, um para o interfaceamento entre o módulo e o duto de dados e endereços e um segundo que funciona como interface entre o módulo e o duto de comunicação de alto nível.

4. *ESQUEMA LÓGICO DO HARDWARE*

Nesta seção é apresentado o esquema lógico do hardware, destacando-se os diagramas de blocos do acesso ao cache é a memória principal, que são os níveis de armazenamento que dependem exclusivamente do hardware do módulo.

A figura III-7 mostra esse esquema lógico, destacando os blocos que compõe a arquitetura apresentada na figura III-6.

. Esquema do cache e do seu controlador

Como o cache é dividido por processo, em todo acesso à memória do SDF, deve-se verificar qual processo esta utilizando o módulo de memória, para selecionar qual das linhas de 64 bits, pode conter o dado relativo ao endereço requisitado.

O circuito de seleção de processo, seleciona o comparador apropriado, que por sua vez verifica se um dos endereços que utilizam o cache, coincide com o endereço requisitado.

Com o resultado dessa comparação, os dois bits menos significativos do endereço virtual e algumas informações sobre o acesso, tem-se os dados necessários para um circuito de controle, que verifica se é um acesso que deve utilizar o cache. Em caso afirmativo, coloca-se no duto a palavra adequada, dentre as quatro que ocupam a linha selecionada.

. Esquema do controlador de memória

O controlador de memória utiliza o mesmo bloco de seleção do processo utilizado pelo controlador de cache. Nesse caso, deve ser verificado se o endereço requisitado pertence à página onde foi feita a última leitura.

Para essa verificação é utilizado um circuito contendo:

a. circuito comparador

b. registrador contendo a página onde foi feita a última leitura - registrador de página

c. registrador contendo o endereço real da página referida no

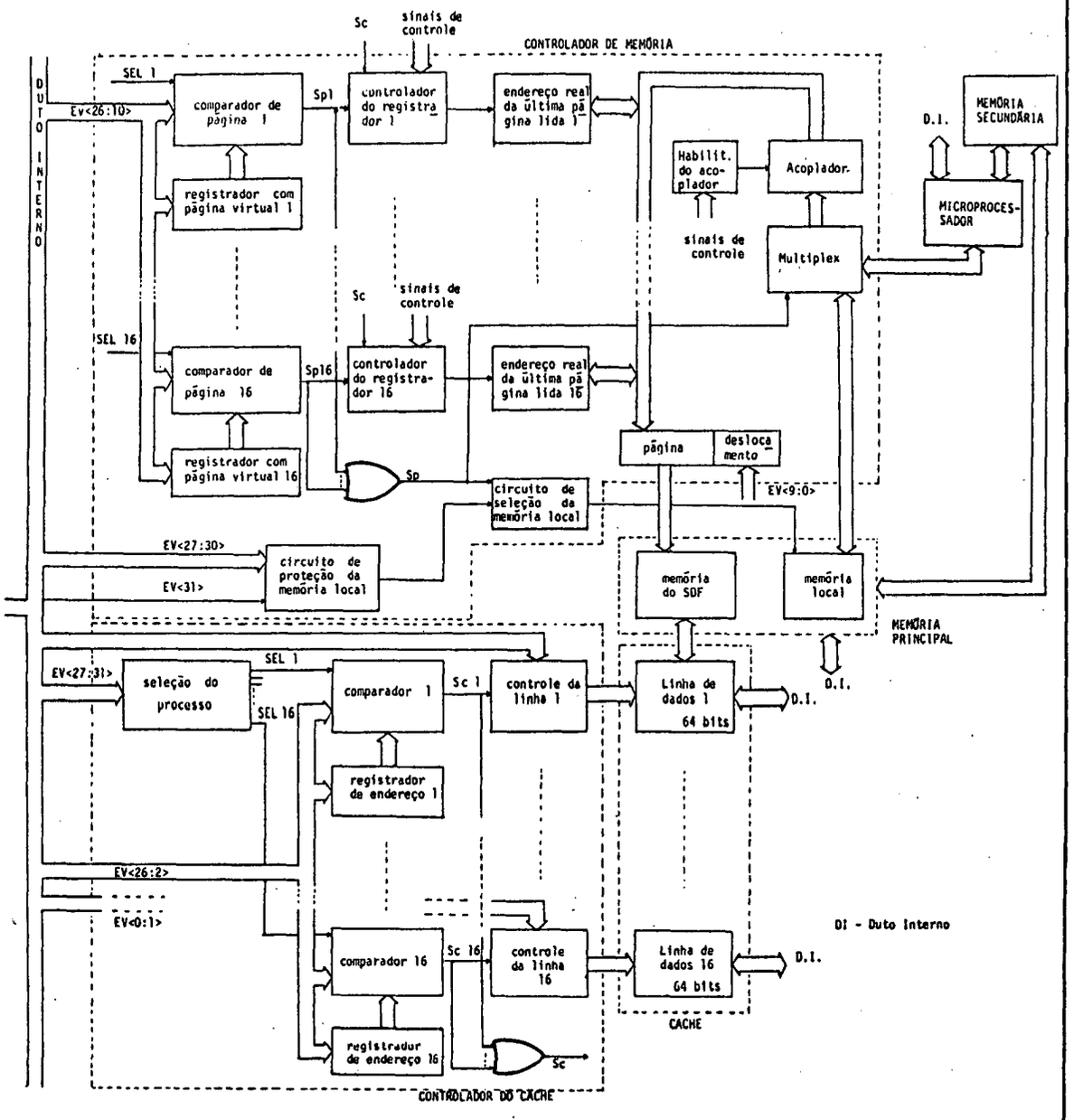


Figura III-7 - Esquema Geral do Módulo de Memória Virtual

Ítem b - registrador de endereço real.

O circuito comparador verifica se a página a que se quer ter acesso, coincide com o conteúdo do registrador de página do processo especificado no endereço virtual.

Utilizando o resultado dessa comparação, pode-se controlar um circuito multiplex, que seleciona o registrador de endereço real relativo ao processo, se a comparação fornecer um resultado verdadeiro e seleciona o endereço fornecido pela memória local, em caso contrário.

Para a obtenção do endereço da página na memória local, basta endereçar essa memória com o endereço obtido somando a página especificada em EV<27:30> ao endereço base do processo na memória local.

Ao endereço real da página obtido, deve-se concatenar o deslocamento dentro da página fornecido pelos bits 0 a 9 do endereço virtual.

Os procedimentos, descritos até o momento nesta seção, são para os casos em que o acesso é para a memória do SDF. Para acessar a memória local é acionado um circuito que verifica se o acesso pretendido é permitido. Em caso afirmativo esse circuito libera a memória local para atender a solicitação, utilizando o endereço recebido pelo duto de dados e endereços.

Os acessos à memória secundária são comandados pelo microprocessador através de um software e estruturas contidos na memória local. Um esquema do software e das estruturas utilizadas são apresentados na seção 5 deste capítulo.

5. *ESTRUTURA MODULAR DO SOFTWARE*

Nesta seção são apresentadas as estruturas de dados necessárias para o gerenciamento da memória virtual e a estrutura do software que o processador local deverá suportar.

5.1. *Estruturas de Dados*

. Tabela de Páginas Virtuais (TPV)

A TPV fornece informações sobre as páginas virtuais, sendo que cada entrada da tabela deve corresponder a uma página virtual e para cada processo deve existir uma TPV. As informações existentes na TPV, contêm os seguintes campos:

TPV<15> - indica se a página virtual relativa a essa entrada está presente na memória principal.

TPV<14> - indica se a página foi modificada ou não. Essa informação é importante no caso de ser necessário utilizar o espaço, que a página está ocupando na memória principal.

TPV<13> - indica a proteção da página em relação a leitura.

TPV<12> - indica a proteção da página em relação a escrita.

TPV<11:00> - indica endereço real da página, no caso em que a página está presente na memória principal. Em caso contrário, este campo não tem significado.

Na figura III-8, pode-se observar a TPV de um processo genérico.

. Tabela de Páginas Reais (TPR)

A TPR fornece informações sobre as páginas da memória principal, sendo que cada entrada da TPR corresponde a uma página real.

A princípio, cada processo possui um certo número de páginas reais destinadas a ele na memória principal. Para cada página tem-se uma entrada na TPR do processo. Um processo que necessite um número maior de páginas pode utilizar as páginas reais de um processo que não esteja inicializado. Essa variação no número máximo de páginas reais para um processo, é controlada

através de software.

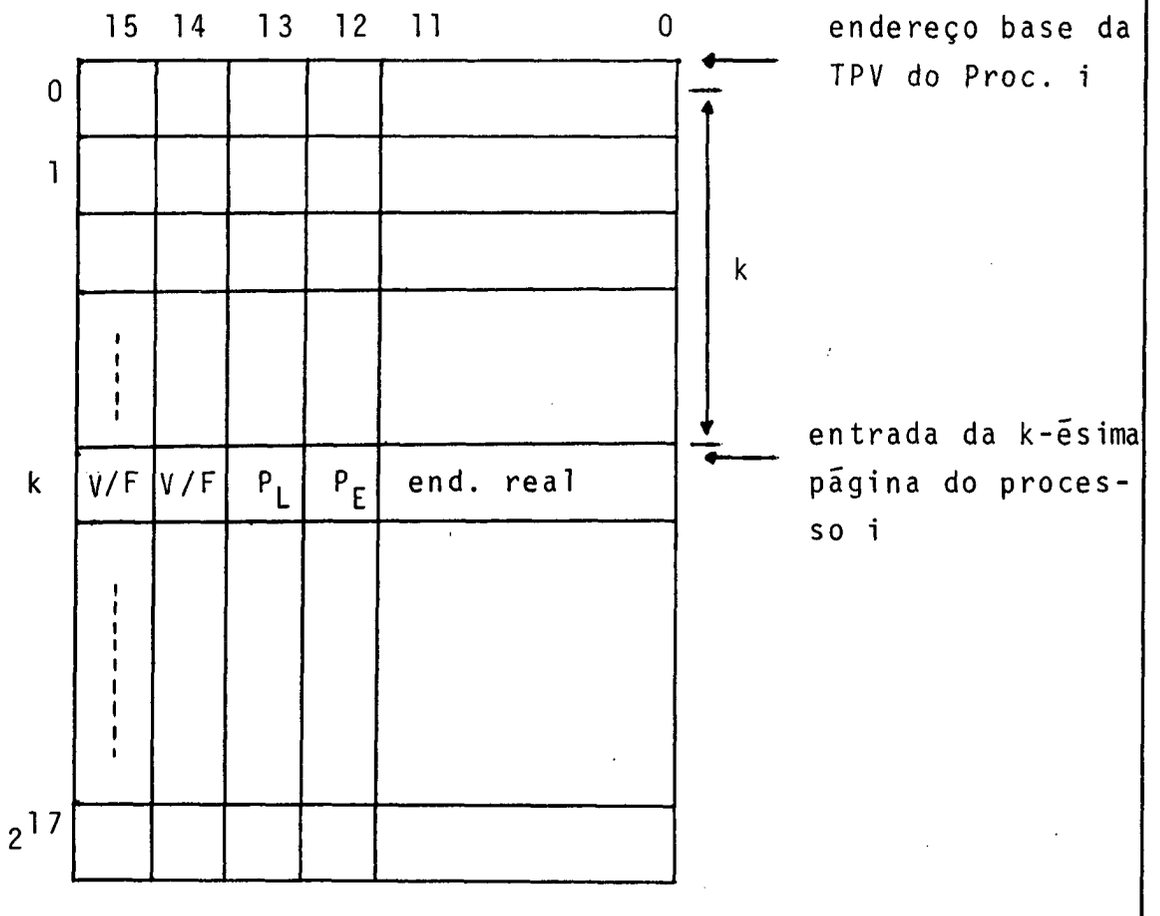


Figura III-8 - TPV de Um Processo Genérico

As informações existentes na TPR estão dispostas nos seguintes campos:

TPR<16> - indica se a página real está sendo ocupada.

TPR<15:4> - esse campo pode ter dois significados:

. se a página estiver ocupada, indica a página virtual que está ocupando essa página real.

. se a página não estiver ocupada, indica a próxima página real desocupada.

TPR<3:0> - indica o processo que está utilizando essa página real.

Nessa estrutura tem-se ainda ponteiros que indicam a primeira página livre de cada processo. Esses ponteiros facilitam a alocação de páginas virtuais.

Na figura III-9, pode-se observar a TPR de um processo genérico, com 2 páginas ocupadas.

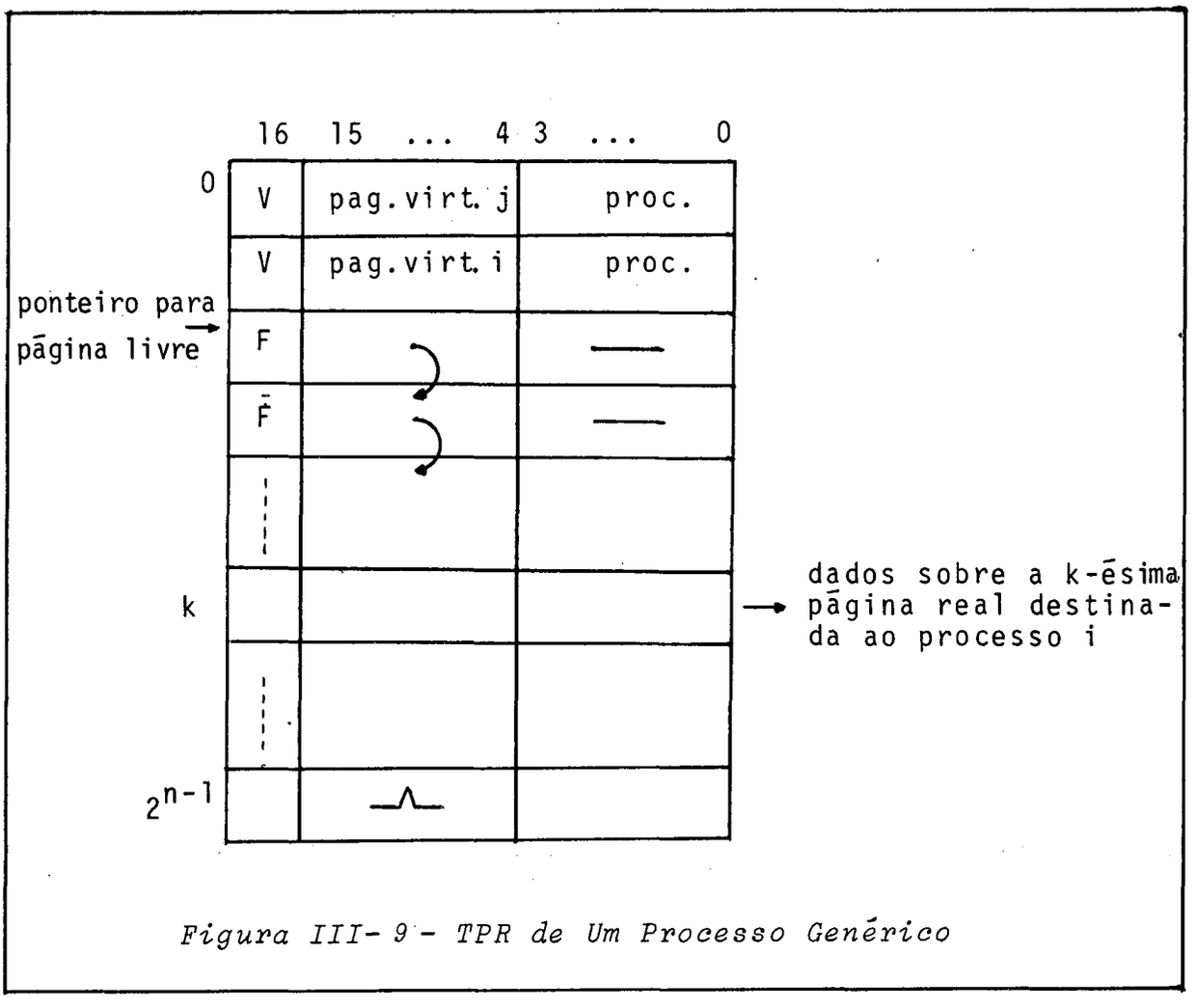


Figura III-9 - TPR de Um Processo Genérico

. Fila estatística (Fila)

Esta estrutura é utilizada para a manutenção de uma estatística de retirada de páginas da memória secundária.

A estatística se baseia na implementação de várias filas (4 por exemplo), contendo as páginas que ocupam a memória real, sendo que cada fila possue diferentes prioridades, na transferência para a memória secundária. Toda vez que uma página é retirada da memória secundária, ela é colocada em uma fila com menor prioridade para retornar ao disco.

Assim, a escolha das páginas que serão armazenadas

na memória secundária, é feita através de uma fila, mas com prioridade para as páginas que foram trazidas um número menor de vezes, pois estas devem estar sendo menos requisitadas.

Para a implementação dessa estatística, são necessárias duas estruturas de dados:

. Filas das páginas virtuais que ocupam a memória principal. Cada elemento dessas filas deve conter:

Fila<15:12> - página virtual

Fila <11:0> - próximo elemento da fila

Deve-se ter ainda ponteiros indicando o início e final de cada uma das filas. Um esquema dessa estrutura pode ser obervado na figura III-10.

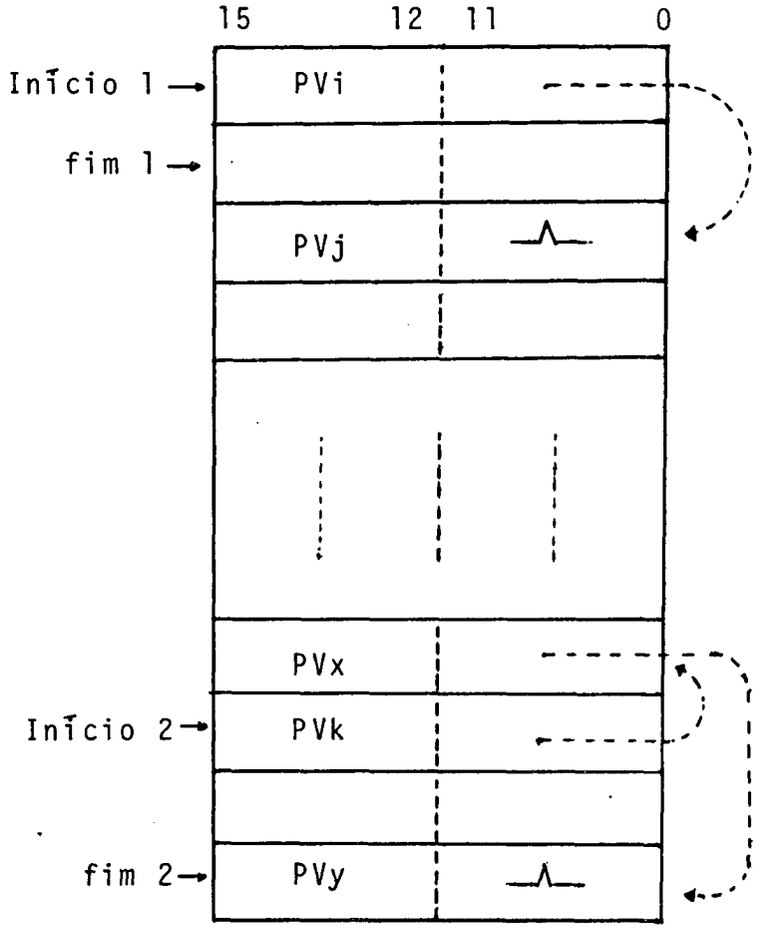


Figura III-10 - Filas das Páginas Virtuais que Ocupam a Memória Principal

. Estatística contendo o número de vezes que cada página virtual já foi retirada da memória secundária. Cada elemento dessa estrutura deve conter:

Estatística <2:0> - número de vezes que a página foi retirada da memória secundária.

Um esquema desta estrutura pode ser observado na figura III-11.

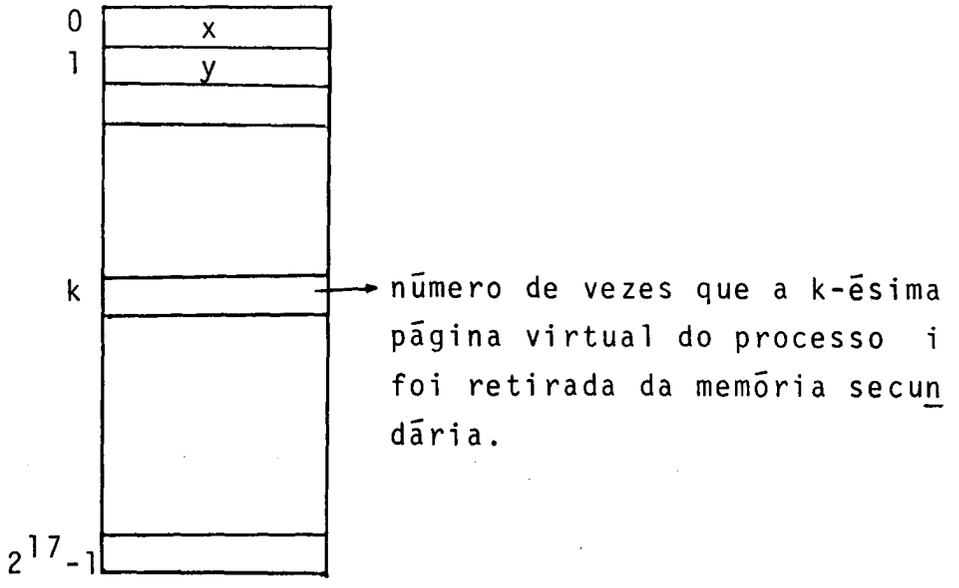


Figura III-11 - Estrutura para Manutenção da Estatística de Retirada de Páginas de Memória Secundária

. Tabela de Processos - TP

A tabela de processos fornece informações sobre a utilização da memória real destinada a cada processo, sendo que cada entrada da tabela corresponde a informações sobre um processo.

As informações contidas na TP, estão nos seguintes campos:

TP<4> - indica se o processo foi inicializado.

TP<3:0> - indica se a memória principal destinada a esse processo está ocupada por outro processo. No ca-

so afirmativo, esse campo informa qual processo está ocupando essa área de memória.

Na figura III-12, tem-se o esquema de uma tabela de processos.

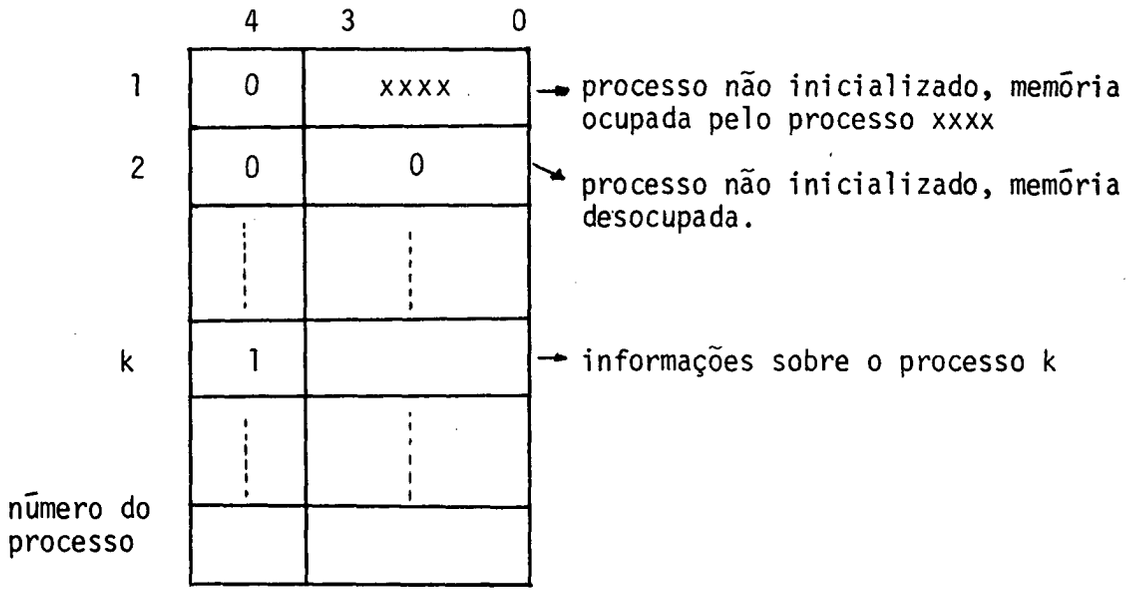


Figura III-12 - Esquema de Uma Tabela de Processos

5.2. *Funções do Software do Módulo de Memória*

O microprocessador dedicado ao módulo de memória deve realizar diversos tipos de tarefas que podem ser solicitadas. Essas terefas devem ficar em uma fila e o atendimento é feito considerando-se uma prioridade pré-estabelecida.

A estrutura do software do módulo de memória pode ser observada na figura III-13.

Na inicialização do sistema, o processador do módulo deve:

- verificar a memória principal, observando a existência de alguma posição com problemas;
- inicializar as tabelas e registradores.

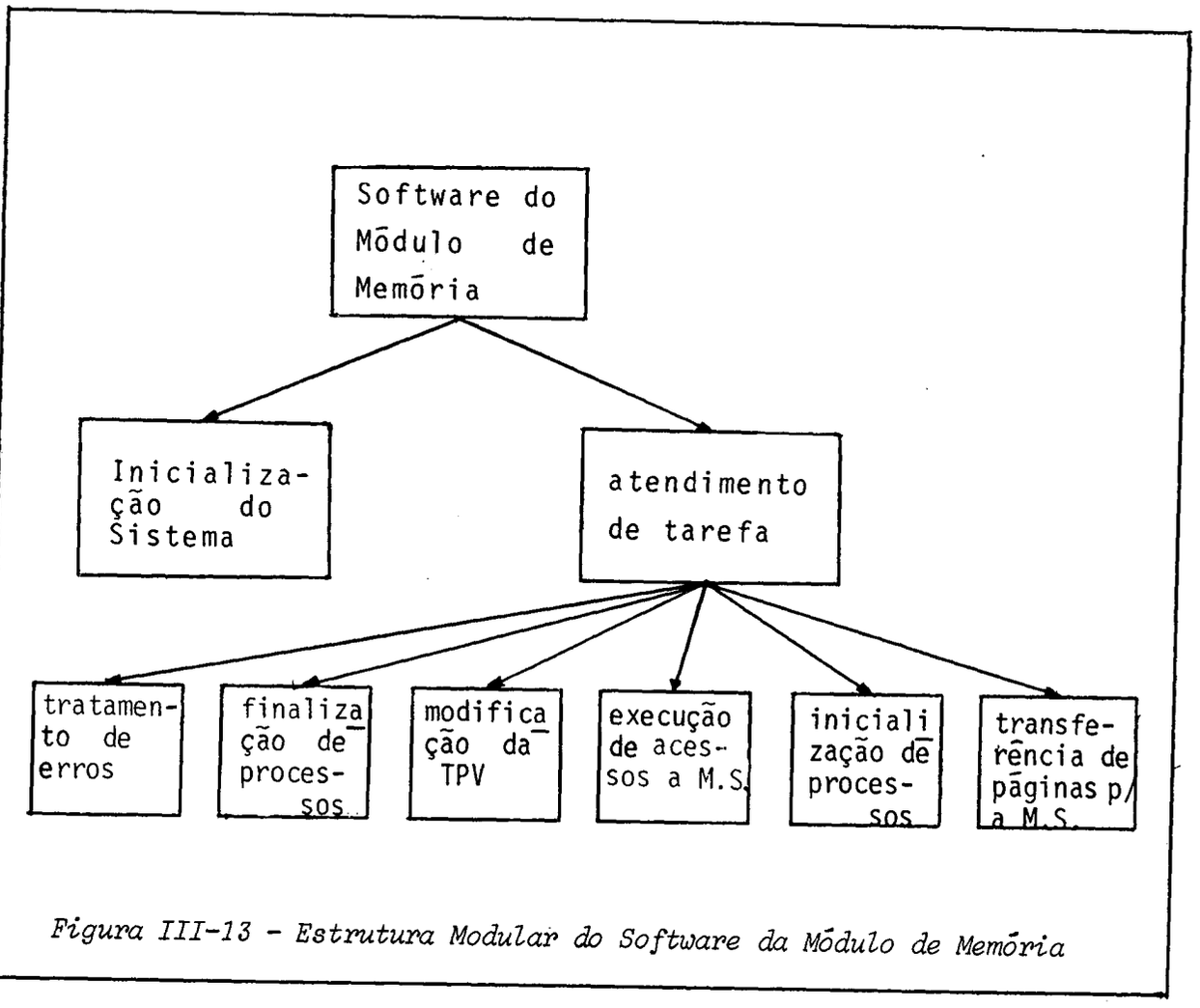


Figura III-13 - Estrutura Modular do Software da Módulo de Memória

Nesta etapa, não é feita a inicialização dos processos, para isso é necessário um comando específico e a inicialização é feita dinamicamente a medida que forem sendo requisitados novos processos.

Após inicializado o sistema, a função do microprocessador passa a ser o atendimento de tarefas solicitadas pelos módulos do SDF ou pelo próprio módulo de memória. Essas tarefas são atendidas levando-se em consideração a prioridade da tarefa. Após o início do atendimento de uma tarefa, esta será completada, mesmo que durante sua execução sejam solicitadas tarefas de maior prioridade. Dentre as tarefas a serem atendidas tem-se:

. Tratamento de erros - Prioridade - 1

Execução de rotinas para o tratamento de diversos tipos de erros como:

- erro na memória principal
- erro na memória secundária

- erro de acesso e outros.

Nesse caso, a prioridade deve ser bastante alta, pois trata-se de finalizar um acesso.

. Finalização de processos - Prioridade - 2

Quando for solicitada uma finalização de processo, o software deve liberar o espaço de memória referente àquele processo, e reinicializar tabelas e registradores.

A prioridade para finalização de processos deve ser alta pois essa operação irá aliviar o sistema, aumentando a memória disponível.

. Modificação da TPV - Prioridade - 3

Em todas as escritas feitas na memória principal é necessário alterar o bit de modificação da entrada da TPV, correspondente à página virtual alterada.

Essa operação é feita através de um software após liberar o duto e é bastante rápida. Deve ser feita antes de qualquer acesso à memória secundária, para que não ocorram erros.

. Execução de Acessos à M.S. - Prioridade - 4

Em todo acesso à páginas que não estejam na memória principal é necessário um software que comande a troca de página. Essa tarefa pode ser dividida entre software anterior e posterior à liberação do dado requisitado. Para antes da liberação do dado, tem-se a estrutura da figura III-14 e para após a liberação do dado, a estrutura representada na figura III-15.

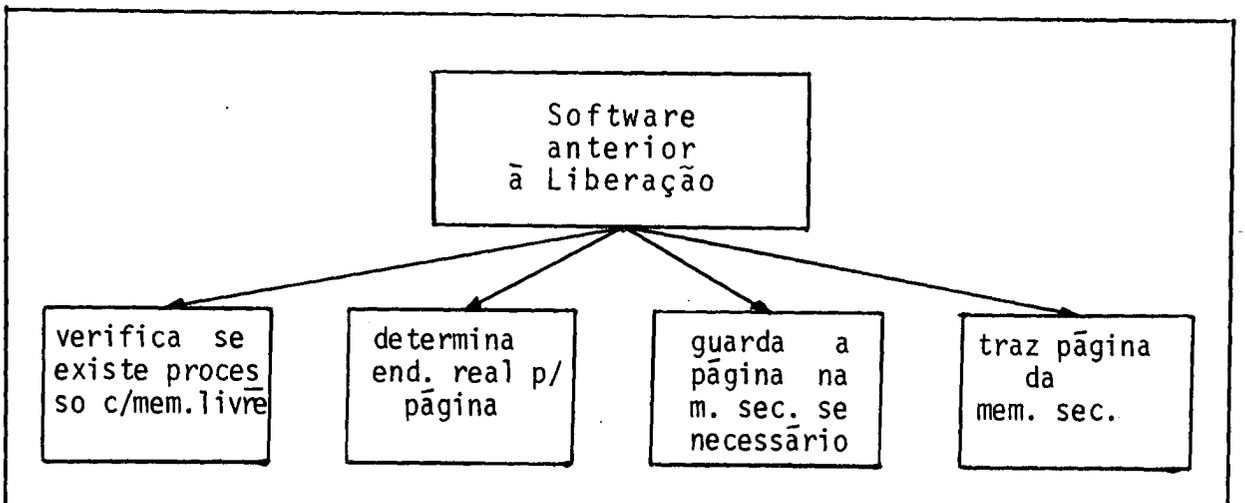


Figura III-14 - Software Anterior à Liberação do Dado em Acessos à Memória Secundária

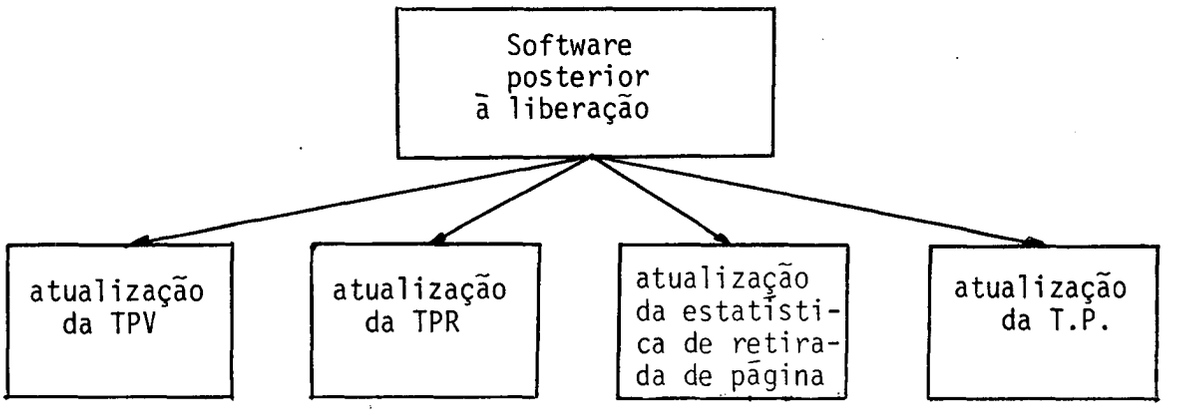


Figura III-15 - Software Posterior à Liberação do Dado em Acessos à Memória Secundária

. Inicialização de Processo - Prioridade - 5

Quando for solicitada uma inicialização de processo, o software deve executar uma série de tarefas que são apresentadas no diagrama da figura III-16.

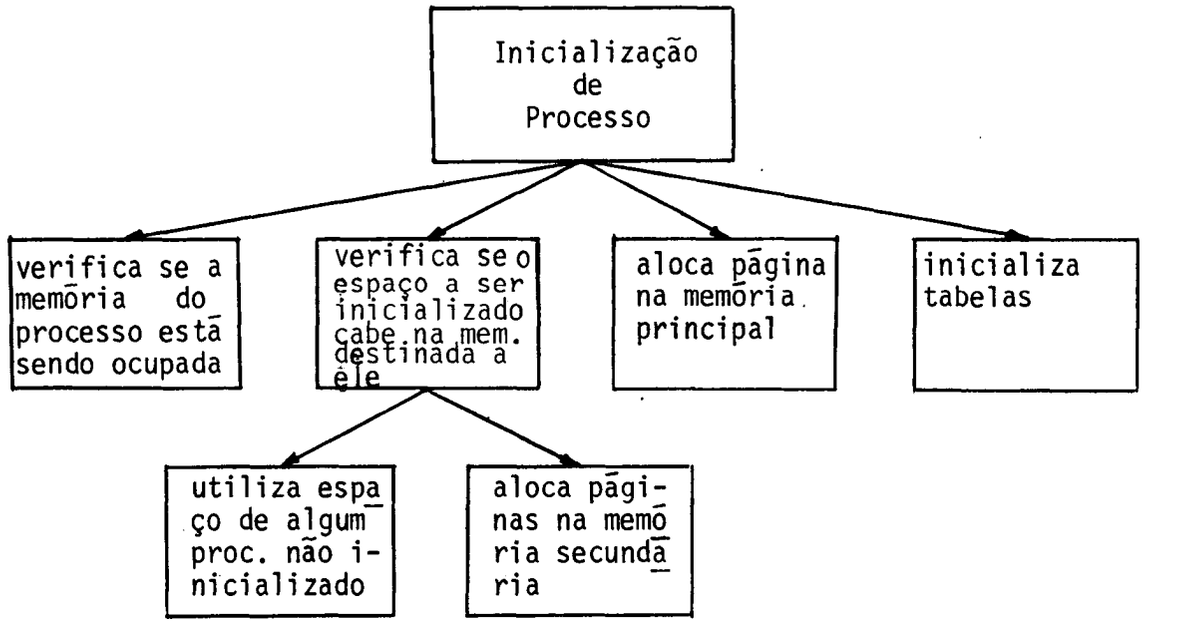


Figura III-16 - Esquema do Software na Inicialização de um Processo

O primeiro passo, a ser executado, é a verificação da disponibilidade do espaço de memória do processo que está sendo inicializado. Caso o espaço esteja sendo ocupado por um outro processo, este deve ser liberado, transferindo as páginas para a memória de um outro processo não inicializados.

Encerrada esta primeira fase, deve-se verificar se o espaço destinado ao processo é suficiente para alocar todas as páginas que devem ser inicializadas. No caso de ser insuficiente, deve-se utilizar o espaço de algum processo não inicializado ou alocar algumas páginas na memória secundária.

Após a definição de quantas páginas poderão ficar na memória principal, deve-se alocar as páginas e inicializar as tabelas.

. Transferência de Páginas para a Memória Secundária

Quando a fila de solicitações estiver vazia, o processador deve utilizar o tempo disponível para guardar as páginas, que sofreram modificações, na memória secundária. Deve-se guardar inicialmente as páginas que estiverem na frente da fila de maior prioridade, para serem retiradas da memória principal.

6. *CONSIDERAÇÕES FINAIS*

Neste capítulo destacam-se de modo geral:

a. O enquadramento do módulo de memória virtual no SDF, com base na arquitetura básica proposta em {Sa85},

b. A definição dos objetivos e importância do módulo de memória virtual,

c. As características gerais do hardware e do software necessários para se alcançar os objetivos do módulo de memória virtual,

d. As diretrizes para definição do software e do hardware necessários.

Com isto, ficam estipuladas as linhas gerais a serem seguidas na descrição e simulação do módulo de memória virtual, que são apresentadas nos capítulos IV e V respectivamente.

CAPÍTULO IV

DESCRIÇÃO DO MÓDULO

1. *CONSIDERAÇÕES INICIAIS*

Neste capítulo é apresentada uma descrição detalhada do módulo de memória, dividida por níveis de armazenamento. Assim, cada seção deste capítulo, apresenta a descrição e discussão de um nível de memória e as justificativas das escolhas efetuadas na fase de definição do módulo. Dessa forma, a seção 2 é dedicada ao cache, enquanto que a memória principal é discutida na seção 3 e a memória secundária na seção 4. Na seção 5 apresenta-se considerações sobre o hardware e o software da memória local do módulo de memória virtual.

Na seção 6 apresentam-se algumas considerações e justificativas gerais sobre o módulo de memória virtual.

O módulo de memória proposto neste capítulo foi simulado e as características decorrentes das conclusões obtidas a partir da simulação, são justificadas no capítulo V.

Para tornar mais claro o funcionamento dos três níveis de memória que compõe o módulo que é apresentado neste capítulo, é necessário ter-se inicialmente uma visão geral de como é tratado um acesso a ele.

Para cada acesso ao módulo de memória, deve-se verificar se o endereço requisitado encontra-se no cache. Em caso afirmativo, deve-se bloquear a busca na memória principal, que é iniciada em paralelo com a verificação do cache. Pode-se descrever os passos de um acesso, da seguinte forma:

- Selecionar o processo a que se está tendo acesso.
- Verificar se o endereço requisitado está no cache e, em paralelo, se ele pertence a última página lida.
- Se o endereço estiver no cache:
  - bloquear a busca na memória principal
  - verificar se não existe erro de acesso

. Habilitar leitura ou escrita no registrador adequado.

. Se o endereço não estiver no cache:

. Se o endereço pertence a última página lida:

. Acionar o registrador que contém o endereço real da página

. Se o endereço não pertencer a última página lida:

. Obter o endereço da TPV na memória local

. Fazer um acesso a memória local, para obter a entrada da TPV relativa a página endereçada

. Se a página não estiver na memória principal

. Acionar o processador para a transferência de página

. Acionar o acoplador para obtenção do endereço real

. Fazer o acesso a memória do SDF.

2. *MEMÓRIA CACHE*

2.1. *Descrição do Esquema Proposto*

O cache do módulo de memória possui uma unidade de transferência de informação entre memória principal e cache, para cada processo, que é a linha. Cada linha do cache contém 4 palavras, sendo que cada palavra é composta de 2 bytes (16 bits).

O esquema proposto para o cache é o apresentado na figura IV-1.

. Considerações Gerais Sobre o Esquema

Em todo acesso ao cache, a primeira operação a ser executada, é a determinação do processo que está tendo acesso ao módulo. Para isso, um circuito demultiplex seleciona um dos 16 processos, através dos 4 bits do endereço virtual que definem o processo.

Nessa seleção, um circuito comparador é acionado e é verificado se o conteúdo do registrador de endereço selecionado, coincide com o endereço virtual requisitado.

Nos casos em que coincide e não existem problemas de proteção, é acionado um circuito habilitador de escrita ou leitura (dependendo do tipo de acesso requerido), que aciona a parte apropriada do cache.

Nos casos em que não coincide, podem-se ter dois procedimentos. Se o acesso pretendido for uma leitura, o circuito habilitador de escrita, habilita a escrita nas quatro palavras da linha do cache e o seu conteúdo é modificado.

Com a modificação do conteúdo do cache, deve-se alterar o conteúdo do registrador de endereço. Dessa forma, toda vez que todas as palavras de uma linha do cache forem alteradas, deve ser fornecido um pulso ao registrador, para que este registre o novo endereço dos dados contidos no cache.

Tanto para o registrador de endereços quanto para o comparador, o endereço considerado corresponde a EV<26:2>. Isto é, os dois bits menos significativos não são considerados, uma vez que na linha do cache tem-se 4 palavras, cujos endereços virtuais são iguais, a menos dos 2 bits menos significativos.

Nos casos em que o registrador de endereços é modi

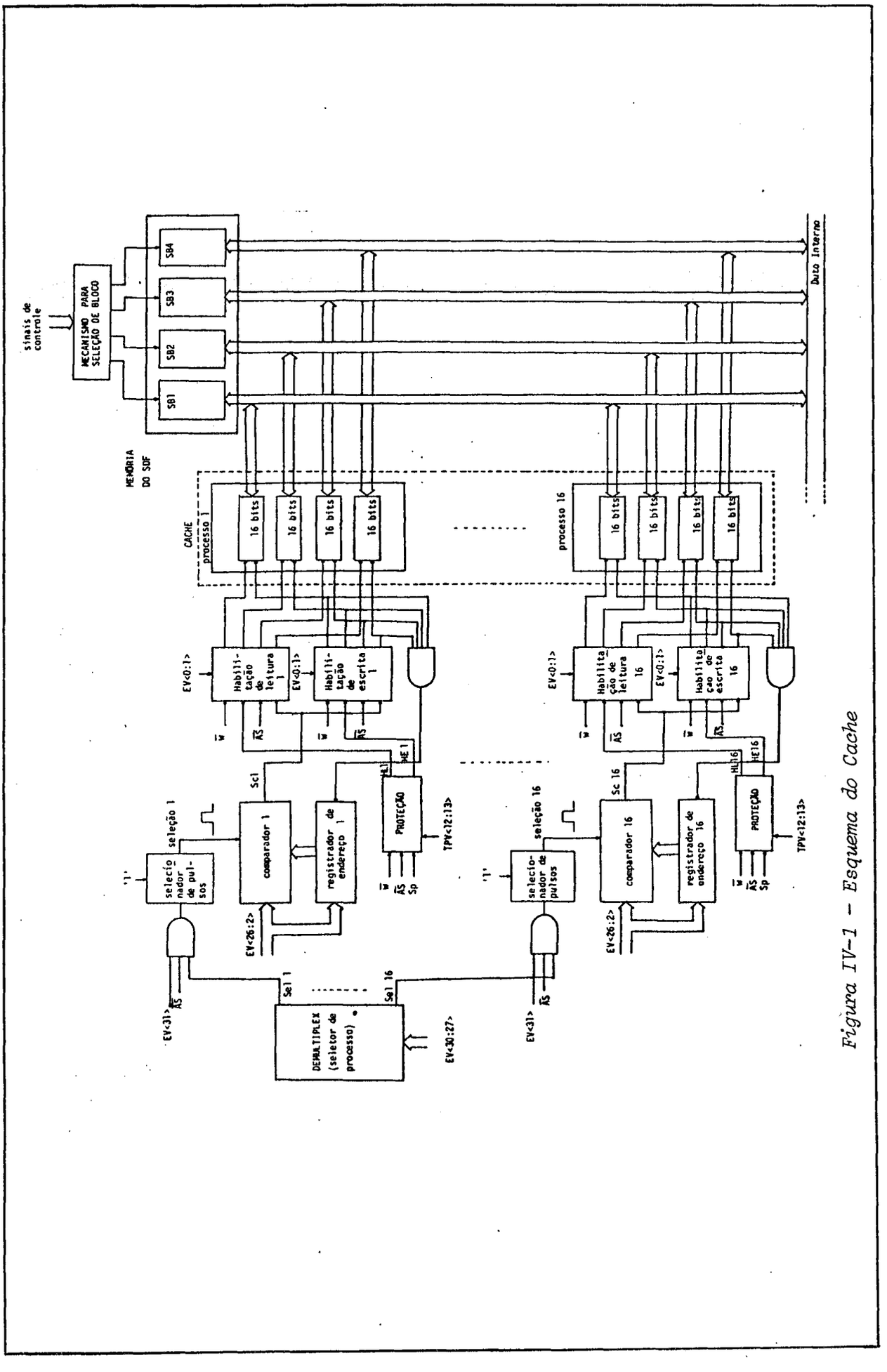


Figura IV-1 - Esquema do Cache

ficado, seu conteúdo passa a ser igual ao da entrada do comparador. Neste instante, o comparador deve estar desativado para que não ocorram erros, pois Sc passaria a indicar que o endereço está no cache. Com a finalidade de manter o comparador ativo apenas o tempo necessário, tem-se um circuito que gera um pulso, toda a vez que uma comparação deve ser efetuada.

. Proteção Contra Acessos não Permitidos.

Quando o endereço requisitado está no cache, não é necessário fazer-se acessos à tabela de páginas. Dessa forma, para determinar se o acesso é permitido, deve-se ter registradores (RL e RE), que contenham o tipo de acesso possível à página a que o endereço do cache pertence. A partir desses registradores verifica-se a proteção dos dados do cache. Na figura IV-2, tem-se o mecanismo proteção para os dados do cache, para um processo genérico.

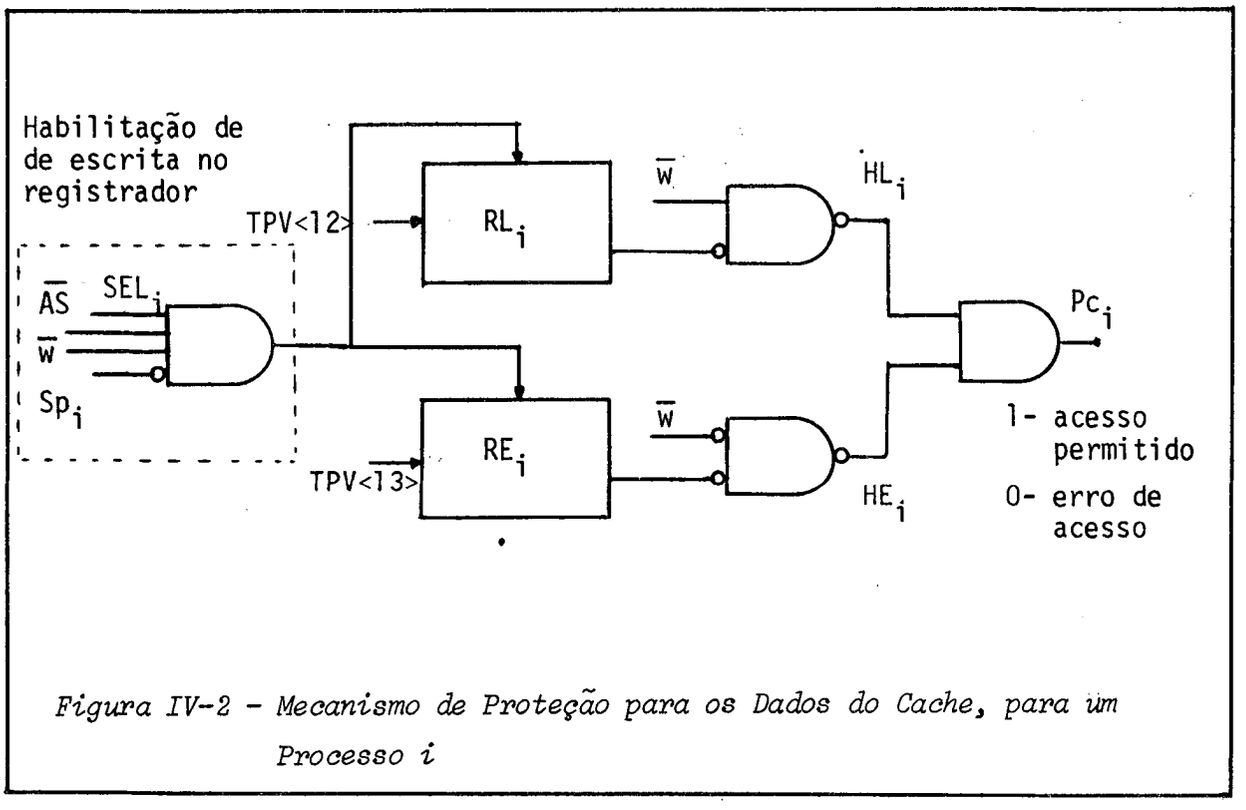


Figura IV-2 - Mecanismo de Proteção para os Dados do Cache, para um Processo i

Nesse circuito, tem-se que HL será "1", se um acesso para leitura estiver correto e de forma análoga HE será "1", nos casos em que se tem uma escrita correta. A saída Pc ficará em nível alto, quando não existirem erros de acesso.

Os registradores RL e RE deverão ser atualizados sempre que a página a que se está tendo acessos para leitura for alterada. Isso porque toda a proteção é feita por página. Assim, RL_i e RE_i deverão assumir os valores de TPV<12> e TPV<13> respectivamente, sempre que se tenha uma leitura no processo i, em página diferente da do último acesso para leitura (Sp = 1).

. Circuito para Habilitação de Leitura no Cache

Os dados armazenados no cache serão lidos apenas quando ocorrer uma leitura em um endereço presente no cache. Nesse caso, deve-se acionar a saída do circuito de habilitação de leitura correspondente à palavra a que se quer ter acesso, do processo selecionado. A leitura de uma palavra do cache depende dos seguintes fatores:

a. a palavra a ser lida deve pertencer ao processo selecionado - Sel = 1,

b. o acesso feito ao módulo de memória deve ser para leitura - $\overline{As}$ = 1 e $\overline{w}$ = 1,

c. o tipo de acesso deve ser permitido - HL = 1,

d. o endereço requisitado deve estar no cache - Sc = 1,

e. deve-se selecionar a palavra dentro da linha, através dos dois bits menos significativos do endereço virtual.

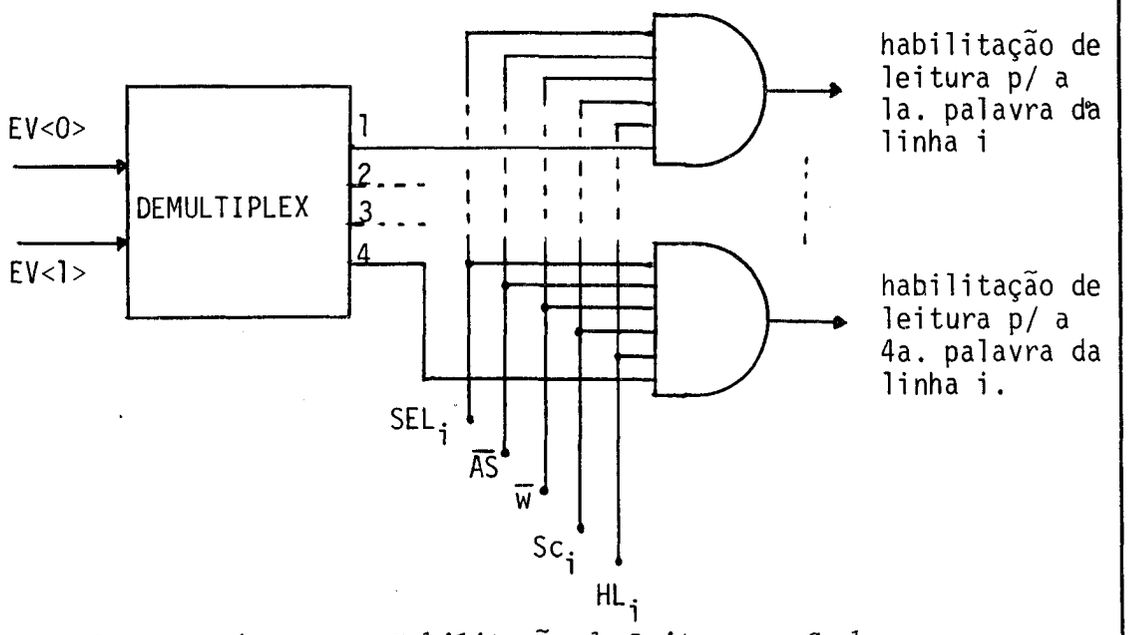
Assim, o mecanismo de habilitação de leitura para um processo genérico é o apresentado na figura IV-3.

. Circuito Para Habilitação de Escrita no Cache

O cache será alterado em dois tipos de operação: Escrita em um endereço que está no cache e leitura em um endereço que não está no cache. O circuito de habilitação de escrita no cache deve prever estes dois casos.

- Escrita em um endereço que está no cache.

Neste tipo de acesso, deve-se atualizar tanto a palavra do cache referente ao endereço virtual requisitado, quanto a memória principal. A habilitação de escrita em uma palavra do cache, depende dos mesmos fatores relacionados para a leitura de



Figuta IV-3 - Mecanismo para Habilitação de Leitura no Cache para um Processo i

uma palavra do cache, alterando-se apenas o ítem b, onde no caso de escrita deve-se ter $\overline{w} = 0$ e $\overline{As} = 1$. Deve-se ainda atualizar a memória principal; este procedimento será descrito na seção 3 deste capítulo.

- Leitura em um endereço que não está no cache.

Neste caso, toda a linha do cache, relativa ao processo a que se está tendo acesso, deve ser modificada. Para isso, deve-se trazer da memória principal as 4 palavras que são endereçadas pelo endereço virtual, retirando-se os dois bits menos significativos. Como toda a linha do cache será modificada, a habilitação de escrita depende somente dos seguintes fatores:

a. o endereço requisitado não está no cache - Sc = 0,

b. o acesso feito ao módulo deve ser para leitura - $\overline{As} = 1$ e $\overline{w} = 1$,

c. a linha do cache que será modificada pertence ao processo selacionado - Sel = 1,

Assim, a escrita em cada palavra do cache deve ser habilitada sempre que ocorrer uma escrita no endereço cujo conteúdo está nessa palavra, ou quando a linha for modificada. Na figura IV-4, tem-se o mecanismo para habilitação de escrita no

cache, para um processo genérico.

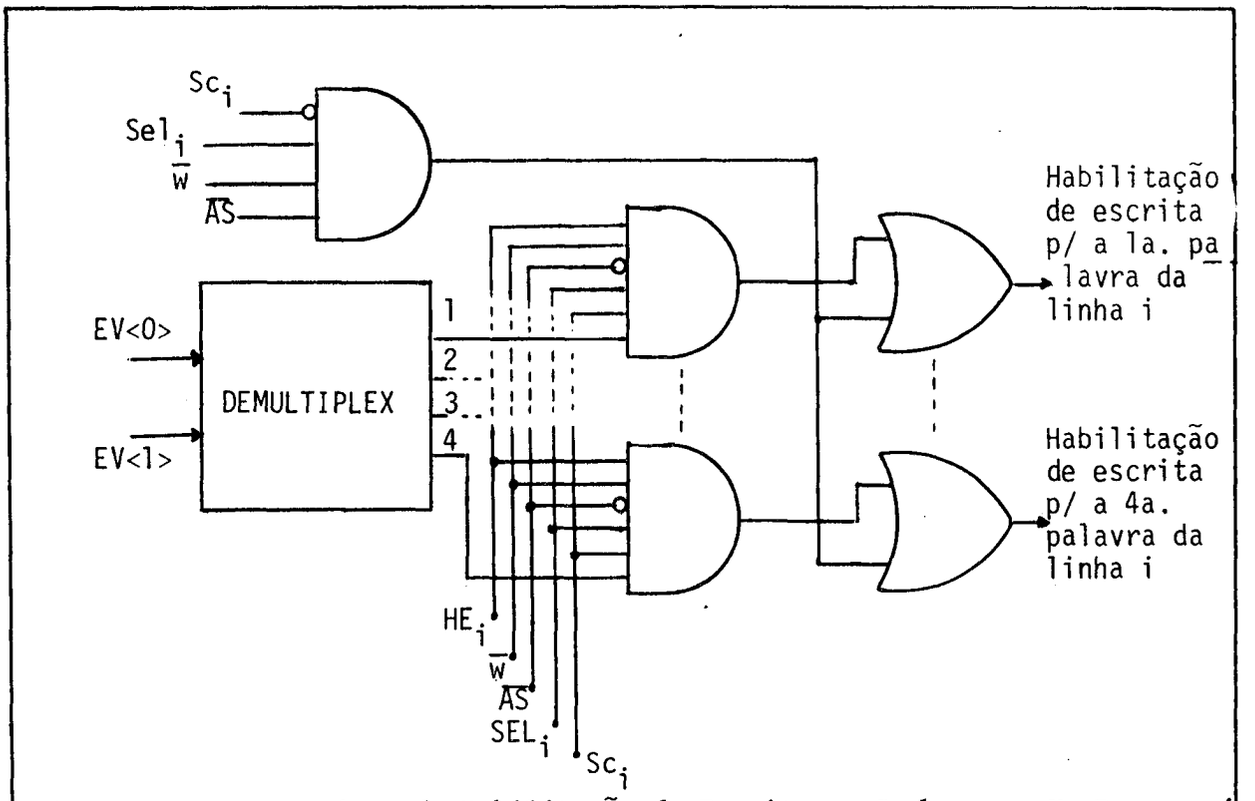


Figura IV-4 - Mecanismo de Habilitação de Escrita no Cache para Um Processo i

. Escrita em Um Endereço que Não Está no Cache.

Em acessos deste tipo, não existe participação do cache, uma vez que a escrita é feita diretamente na memória principal e acessos para escrita não modificam o endereço do cache . Este tipo de acesso será detalhado na seção 3 deste capítulo.

2.2. *Diagrama de Tempo Para Acessos ao Cache*

Para a verificação do tempo de acesso ao cache proposto, é necessário determinarem-se os sinais de controle do duto de baixo nível (DDE), por onde circulam os dados e endereços nos acessos ao módulo de memória virtual, bem como os tempos mínimos entre as operações.

O duto adotado neste trabalho para determinação dos tempos envolvidos é o VMEbus e sua escolha está diretamente relacionada com a proposta apresentada por Santana em {Sa85} e com o fato de se tratar de um duto moderno, adequado a implementação de sistemas distribuídos e bastante considerado atualmente.

As linhas de controle utilizadas na transferência de dados que vem do módulo que solicita o duto são:

a. $\overline{AS}$ - linha de validação de endereço; indica ao escravo (elemento que está recebendo um pedido) que o endereço já está estável,

b. $\overline{W}$ - linha de seleção de escrita ou leitura; indica o tipo de acesso a ser efetuado.

O módulo que recebe a solicitação envia para o módulo origem os seguintes sinais de controle:

a. $\overline{DTACK}$ - indica que o dado foi recebido sem problema, em ciclos de escrita ou que o dado solicitado já está disponível no duto, em ciclos de leitura,

b. $\overline{BERR}$ - indica a ocorrência de um erro no acesso.

Além das linhas de controle, integram ainda o DDE as 16 linhas de dados, que são enviados pelo módulo origem em casos de escrita ou pelo módulo de memória nos ciclos de leitura , tem-se também as linhas de endereço, que no caso do VMEbus, são no máximo 31 linhas disponíveis.

Na figura IV-5, está representado o diagrama de tempos para a leitura de um dado no cache. Tem-se nesse diagrama tanto os sinais que fazem parte do duto, já definidos, como alguns sinais do controlador de cache. O diagrama apresentado é importante tanto para a análise do tempo de acesso ao cache, como para um melhor entendimento de seu funcionamento.

. Descrição dos Tempos Indicados na Figura IV-5.

tempo	descrição	duração(ns)
t_1	tempo necessário para garantir que o endereço está estável	10
t_2	tempo para a seleção do processo (atraso de um demultiplex 4x16)	23
t_3	tempo necessário para que a saída do comparador esteja válida	15.3
t_4	tempo para verificar a proteção	21
t_5	tempo para validação do sinal de habilitação de leitura	23.9

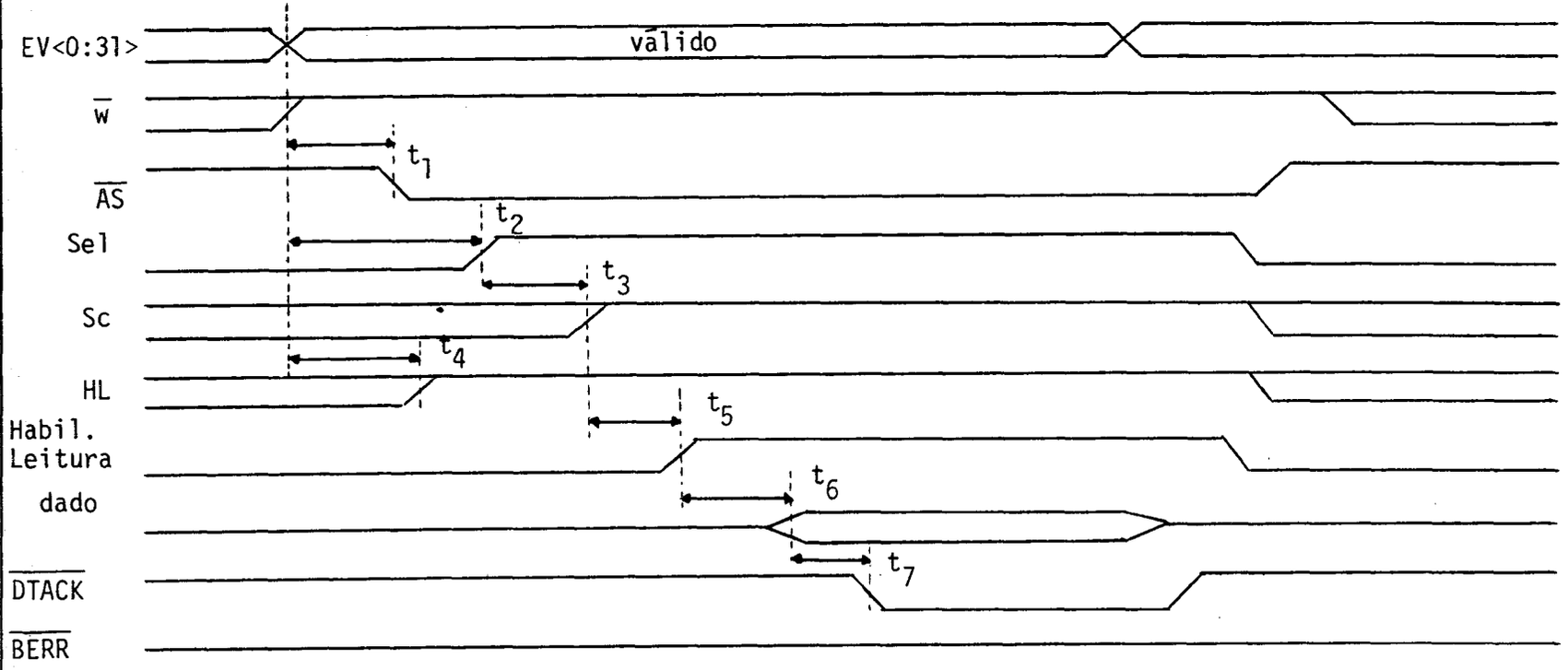


Figura IV-5 - Diagrama de Tempos para Uma Leitura no Cache

tempo	descrição	duração(ns)
t_6	tempo para validação do dado	15
t_7	tempo para garantir que o dado está disponível	0

É importante observar que alguns tempos podem ser sobrepostos, como é o caso dos tempos 1 e 2 do diagrama da Figura IV-5. Somando-se os tempos onde isso não é possível, obtém-se o tempo de acesso ao cache para leituras, que é igual a 77.2 ns.

O diagrama de tempos para uma escrita no cache é apresentado na Figura IV-6.

. Descrição dos Tempos Indicados na Figura IV-6.

tempo	descrição	duração(ns)
t_1	tempo necessário para garantir que o endereço esteja estável	10
t_2	tempo para a seleção do processo	23
t_3	tempo necessário para que a saída do comparador esteja válida	15.3
t_4	tempo para verificar a proteção	21
t_5	tempo para validação do sinal de habilitação de escrita	30.9
t_6	tempo para registrar o dado	24

Cabe observar ainda que a atualização da memória principal, pode ser feita após a liberação do duto, uma vez que o dado já foi registrado no cache.

Para o caso de acessos ao cache para escrita, tem-se um tempo de acesso de 93.2 ns.

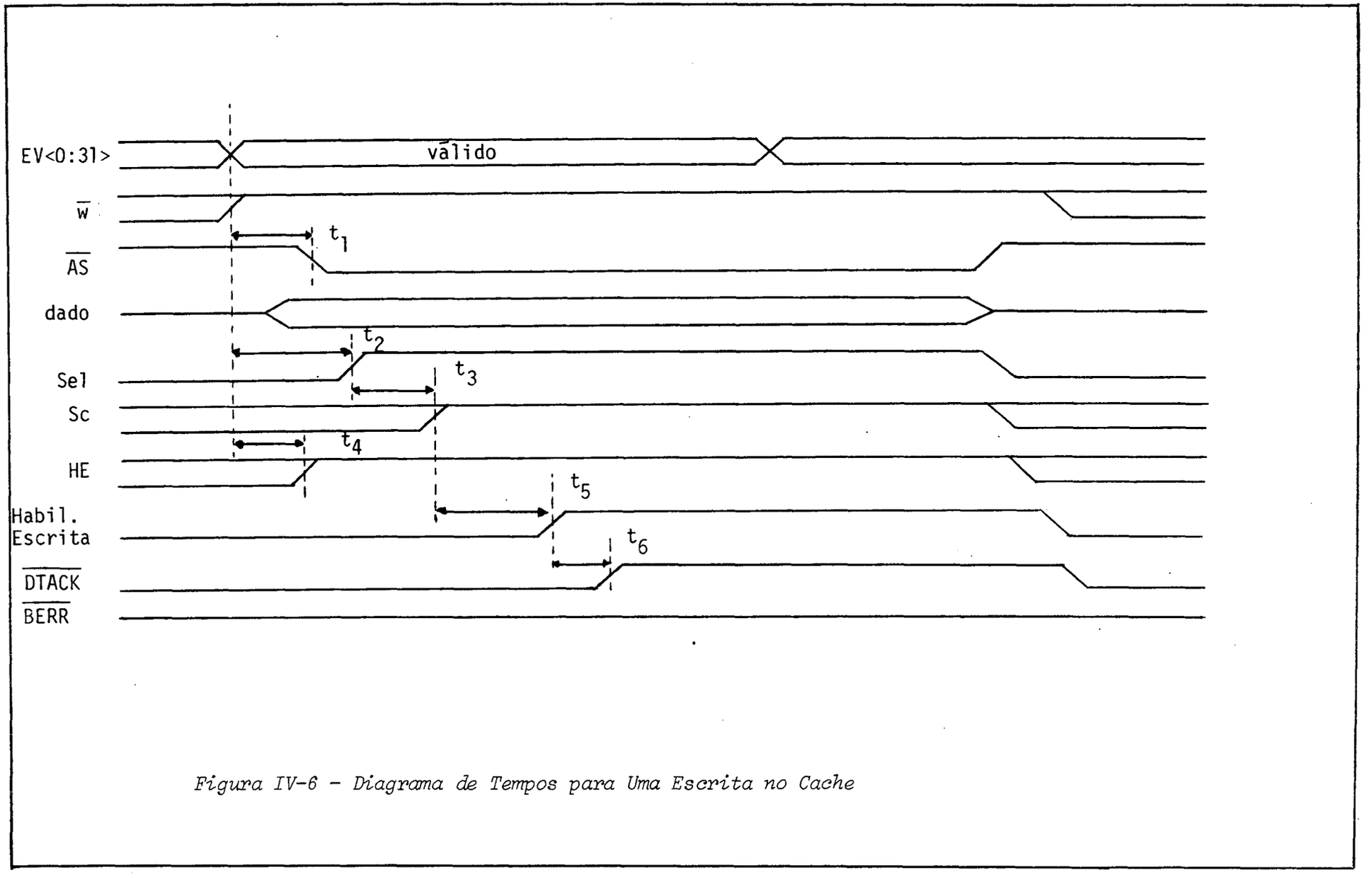


Figura IV-6 - Diagrama de Tempos para Uma Escrita no Cache

2.3. *Justificativas de Algumas Características Adotadas para o Cache*

Neste ítem são apresentadas justificativas de algumas escolhas feitas na fase de definição do módulo de memória.

. Divisão do Cache por Processo

Um cache para o módulo de memória virtual, apresenta o mesmo tipo de problema que um cache para um sistema com multiprogramação. Isto é, ele deve atender pedidos de diferentes processos, que são executados ao mesmo tempo. Assim, sempre que ocorresse uma troca no processo que está utilizando o módulo de memória, não existiriam linhas deste processo no cache, ocasionando muitas faltas. Além disso, as linhas trazidas para o cache podem ser pouco utilizadas, se um novo processo fizer acesso ao módulo.

Optou-se então, por dividir o cache por processo, garantindo assim uma linha para cada processo. Dessa forma, a alteração do processo não terá influência na taxa de acerto do cache.

Uma outra vantagem dessa divisão está na facilidade de implementação dos algoritmos para localização e reposição da informação procurada, que serão melhor discutidos no ítem a seguir.

. Tamanho do Cache e Tamanho da Linha

Tendo-se optado por um cache dividido por processos, é necessário definir-se o número de linhas do cache e o tamanho de cada linha. Para uma implementação inicial optou-se por uma linha por processo, sendo que cada linha é composta por 4 palavras de 16 bits, com isso tem-se um cache que mantém as posições com maior possibilidade de ser requisitadas de cada processo, com uma arquitetura relativamente simples.

A opção de uma linha por processo torna a localização de uma informação bastante simples e rápida, uma vez que basta o número do processo, para obter-se a linha do cache que deve ser considerada. A reposição das linhas do cache é bem simples, principalmente por ter sido adotada a atualização da memória du-

rante a modificação. Assim, para a reposição de uma linha no cache, basta colocar a nova informação nos registradores apropriados.

Com quatro palavras por linhas, tem-se um cache com 128 bytes; a localização da palavra dentro da linha é bem simples e obtém-se uma taxa de acerto razoável, como será discutido no capítulo V.

. Alteração do Cache Apenas em Acessos Para Leitura

Esta característica simplifica bastante o controlador de cache, pois no caso de alteração do cache em acessos para escrita, seriam necessários dois acessos à memória principal. Um primeiro de leitura de quatro palavras para a atualização do cache e um segundo onde seria feita a escrita no cache e na memória principal.

Uma outra justificativa para esta característica é que dificilmente tem-se dois acessos para escrita em uma mesma posição de memória, ao contrário, as escritas são feitas normalmente em posições distantes daquela que se está tendo acesso para leitura. Outro fator a ser considerado é que mais de 80% dos acessos à memória são para leitura {Sm82} e, portanto, esta característica não influencia muito a taxa de acerto do cache.

. Localização do Cache no Módulo de Memória Virtual.

Os problemas citados na revisão bibliográfica, quanto a localização do cache no módulo de memória, são basicamente em relação a minimização do fluxo de informação no duto e do compartilhamento de um mesmo cache por vários processos.

O primeiro problema citado, a atualização do duto, é minimizado pelo pequeno tempo de acesso ao cache. Com isso, o tempo em que o duto fica retido para acessos ao cache é bastante pequeno, não ocasionando problemas, como pode ser observado no capítulo V, na simulação do módulo de memória.

Quanto ao problema de diversos processos compartilharem o mesmo cache, este fica bastante minimizado pela divisão do cache por processos.

A grande vantagem da localização do cache no módu-

lo de memória, é que desta forma não é necessário enviar toda a linha do cache (64 bits) pelo duto, o que ocasionaria a necessidade de um maior número de linhas de dados no duto ou uma serialização das 4 palavras, tornando o acesso mais demorado.

Tem-se ainda que do modo proposto, o módulo de memória centraliza toda a memória não local de todos os outros módulos, ficando a cargo desses apenas o gerenciamento de sua memória local.

3. *MEMÓRIA PRINCIPAL DO SDF*

3.1. *Descrição do Esquema Proposto*

O esquema proposto para o controle dos acessos à memória principal é apresentado na figura IV-7.

. Considerações Gerais Sobre o Esquema.

De forma análoga à do cache, o controlador de memória deve selecionar o processo, ao qual se está tendo acesso. O circuito de seleção é o mesmo utilizado pelo cache.

No caso de acessos à memória principal, deve-se inicialmente verificar se a página que contém o endereço requisitado, foi a última onde se teve acesso para leitura. Nesse caso, o endereço real da página está em um registrador e o endereço real da posição a que se quer ter acesso, pode ser obtido concatenando-se o endereço contido no registrador com o deslocamento da página.

Quando a página requisitada não coincide com a indicada pelo registrador, que é aquela onde se teve a última leitura, o endereço real deve ser obtido na entrada da TPV que se refere à página virtual requisitada, do processo a que se está tendo acesso.

A tabela de páginas virtuais fica armazenada na memória local e o endereçamento, nesse caso, é feito utilizando-se o registrador que contém o endereço base da tabela de páginas, do processo considerado e a página virtual requisitada.

Uma vez feito o acesso à memória local, obtém-se em sua saída a entrada da tabela de páginas virtuais e assim, os dados necessários para determinar se o acesso é permitido (TPV <13:12>) se a página está ou não na memória principal (TPV<15>) e no caso de estar, qual é o seu endereço real (TPV<11:0>).

Supondo que a página está na memória principal, o endereço real pode ser concatenado com o deslocamento na página, para a obtenção do endereço real, de forma análoga ao caso anterior.

O endereço real da página pode, portanto, ser obtido de duas formas:

- a partir da tabela de páginas virtuais: Nesse caso, o endere

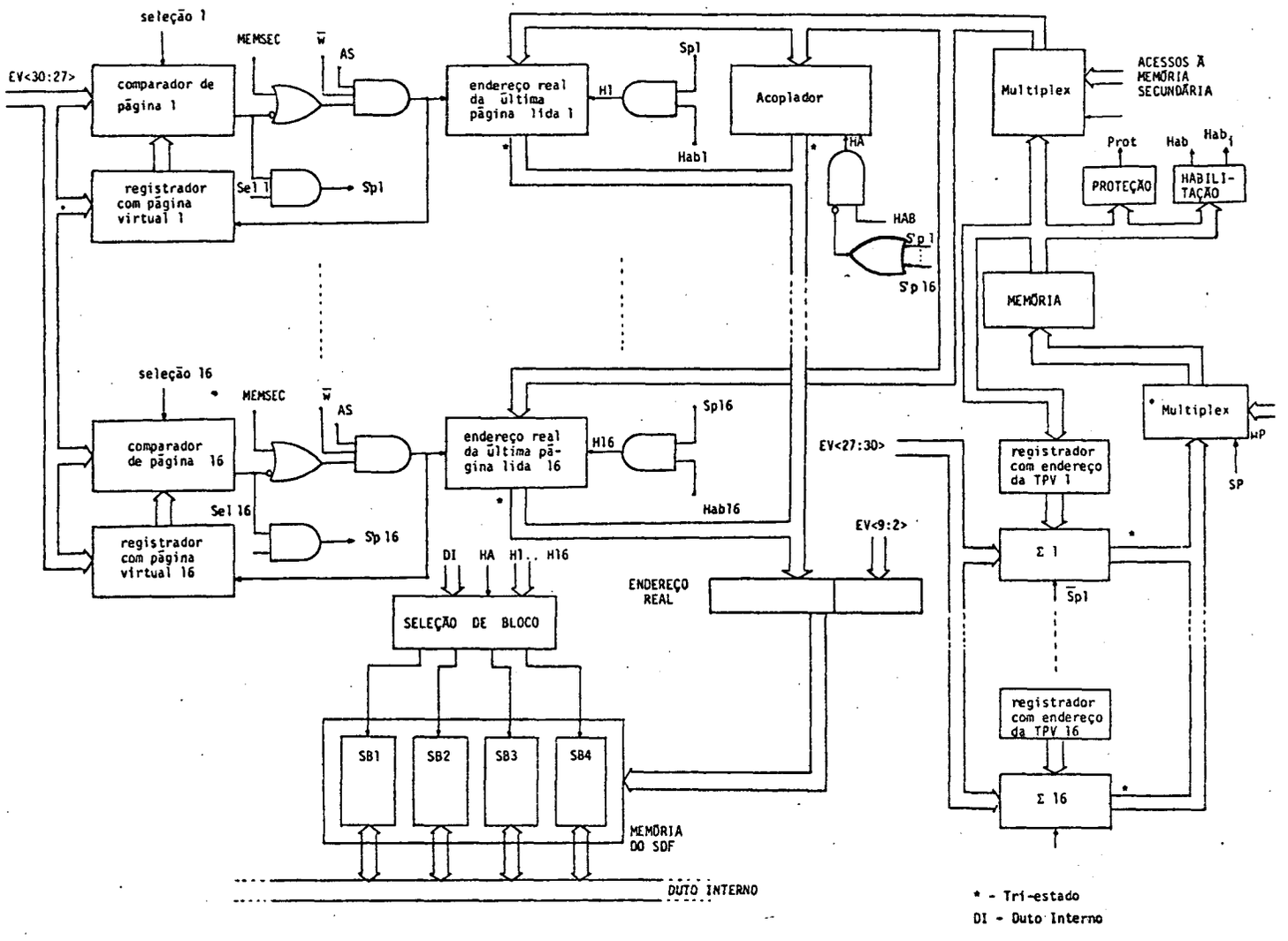


Figura IV-7 - Esquema para Acessos à Memória Principal do SDF

ço passa por um circuito acoplador, com saída em tri-estado, que será habilitada através de um circuito que indica que o endereço real não está no registrador e que o acesso já pode ser feito (Hab = 1);

- a partir do registrador de endereço real: Nesse caso, a saída do registrador, que também é em tri-estado, será habilitada por um circuito que indica, que o endereço real está no registrador (sp = 1), e que o acesso já pode ser feito (Hab_i = 1).

A página cujo endereço real fica armazenado no registrador é alterada, toda a vez que se faz uma leitura em uma outra página. A alteração é feita ativando-se o registrador que contém a página virtual e o que contém seu endereço real, toda vez que se tiver um acesso para leitura, em uma página diferente da que ocupa os registradores.

Como pode ser observado na figura IV-7, o endereço real não utiliza os dois bits menos significativos do deslocamento na página. Isso é devido ao fato de que a memória do SDF é dividida em quatro blocos, que são selecionados pelo circuito seletor de bloco.

. Proteção Contra Acessos Não Permitidos.

A proteção de acesso à memória do SDF é feita de modo diferente, para cada caso descrito no ítem anterior.

Para o caso em que o endereço da página está no registrador de endereço real (Sp = 1), o circuito de proteção é o mesmo que o utilizado no cache, pois, o endereço que está no cache, pertence também à última página onde se teve uma leitura.

Para o caso em que o endereço real está na memória local (Sp = 0), a proteção é feita através dos bits TPV <12> e TPV<13>. O circuito de proteção é o mostrado na figura IV-8.

Nesse último caso, o mecanismo é único para todos os processos e tem algumas de suas entradas vindas da saída da memória local. No primeiro caso, tem-se um circuito para cada processo, devido a necessidade de registradores que indiquem o tipo de proteção para a página, que está no registrador de cada processo.

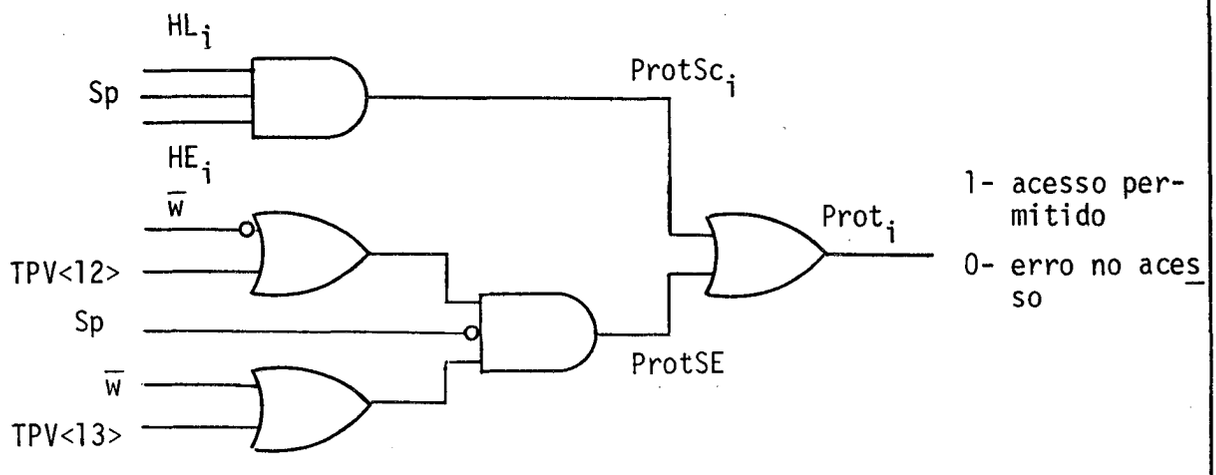


Figura IV-8 - Mecanismo de Proteção para Acessos à Memória Principal

. Mecanismo de Habilitação

Como citado nas considerações gerais sobre o esquema da memória principal, deve-se habilitar a saída do acoplador ou de um dos registradores que contém o endereço real da página a que se está tendo acesso.

Os circuitos de habilitação para os dois casos são bastante semelhantes. A única diferença é devida ao fato de que o acoplador é único para todos os processos e, portanto, deve ser habilitado considerando que qualquer um dos processos pode ser o selecionado, nesse acesso. Os registradores são particulares de cada processo e, portanto, devem ser habilitados apenas quando o processo ao qual estão associados for o selecionado.

As habilitações dependem ainda de que:

- não existam problemas com o acesso desejado;
- o dado não esteja no cache se for um acesso para leitura;
- para o caso de habilitação do acoplador, o endereço real já tenha sido obtido (TPV<15> = 1 ou MEMSEC = 1)
- para o caso de habilitação de um registrador, o processo esteja sendo selecionado (sel = 1).

Na Figura IV-9, tem-se o mecanismo de habilitação, para o registrador que contém endereço real de uma página de um processo genérico e para o acoplador.

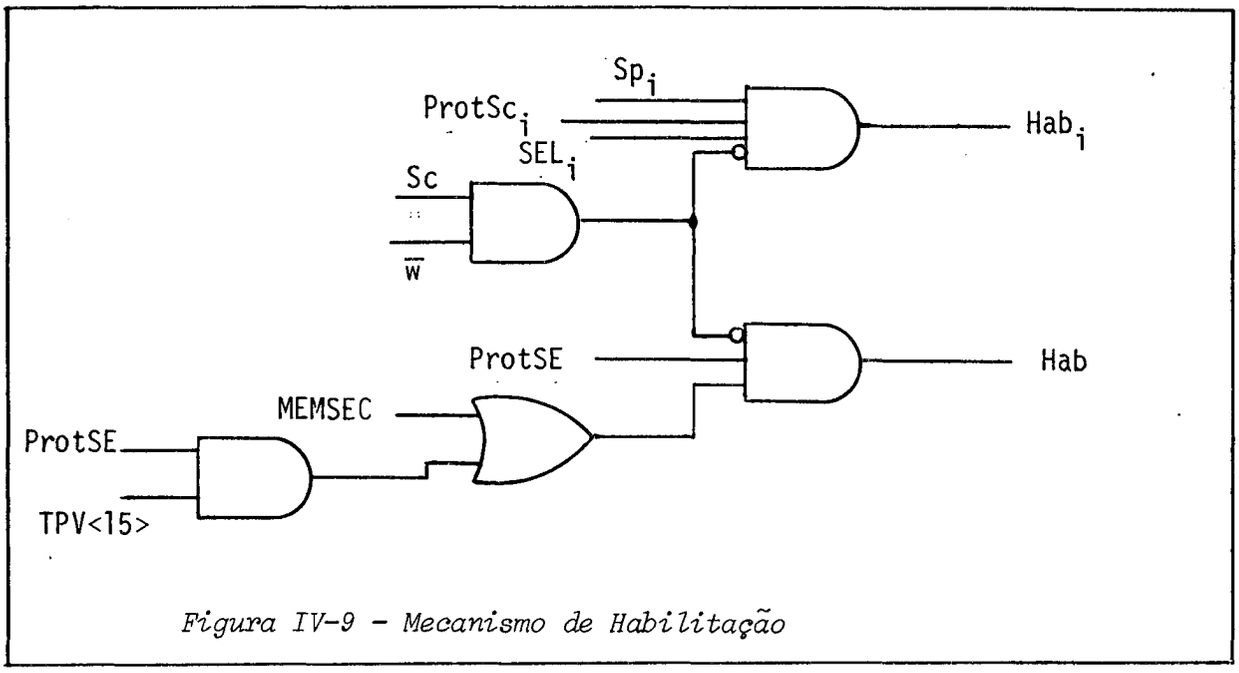


Figura IV-9 - Mecanismo de Habilitação

. Mecanismo Para Seleção de Bloco da Memória Principal

Como citado no capítulo III, a memória principal é dividida em quatro blocos, permitindo o endereçamento de palavras com 64 bits. Para endereçar palavras de 16 bits, devem-se selecioná-las através do selecionador de pastilha. Assim os selecionadores de pastilha de cada bloco devem ser ativiados por um mecanismo de seleção nos seguintes casos:

a. o endereço requisitado encontra-se no cache (Sc = 1) e o acesso é para escrita ($\overline{w}$ = 0): Nesse caso, tem-se uma escrita na memória principal, no bloco definido pelos dois bits menos significativos do endereço virtual. Isto e, deverá ser acionada apenas a seleção de pastilha do bloco referente a EV<0:1>.

b. o endereço não se encontra no cache (Sc = 0) e o acesso é para leitura ($\overline{w}$ = 1): Nesse caso, deve-se ter um acesso para leitura em todos os blocos da memória principal, a fim de obter os dados para alterar o cache.

c. o endereço não se encontra no cache (Sc = 0) e o acesso é para escrita ($\overline{w}$ = 0): Nesse caso, deve-se escrever em um único bloco.

O mecanismo de seleção do bloco de memória a que se quer ter acesso, é o mostrado na figura IV-10, destacando-se cada um dos casos citados.

É importante observar que para qualquer acesso à memória principal deve-se ter o circuito de habilitação em nível alto, pois isso indica que o endereço real já está pronto. Assim, nos três casos citados anteriormente, deve ser incluída essa condição.

Outro fato a ser observado, é que nos casos em que a memória secundária é utilizada, os bits de deslocamento são fornecidos pelo processador, devendo-se ter, portanto, um demultiplexador que selecione os bits através de uma variável (MEM SEC), que indica se o acesso que está acontecendo é ou não controlado pelo microprocessador. A descrição de acessos que envolvem a memória secundária é apresentada em IV-4.

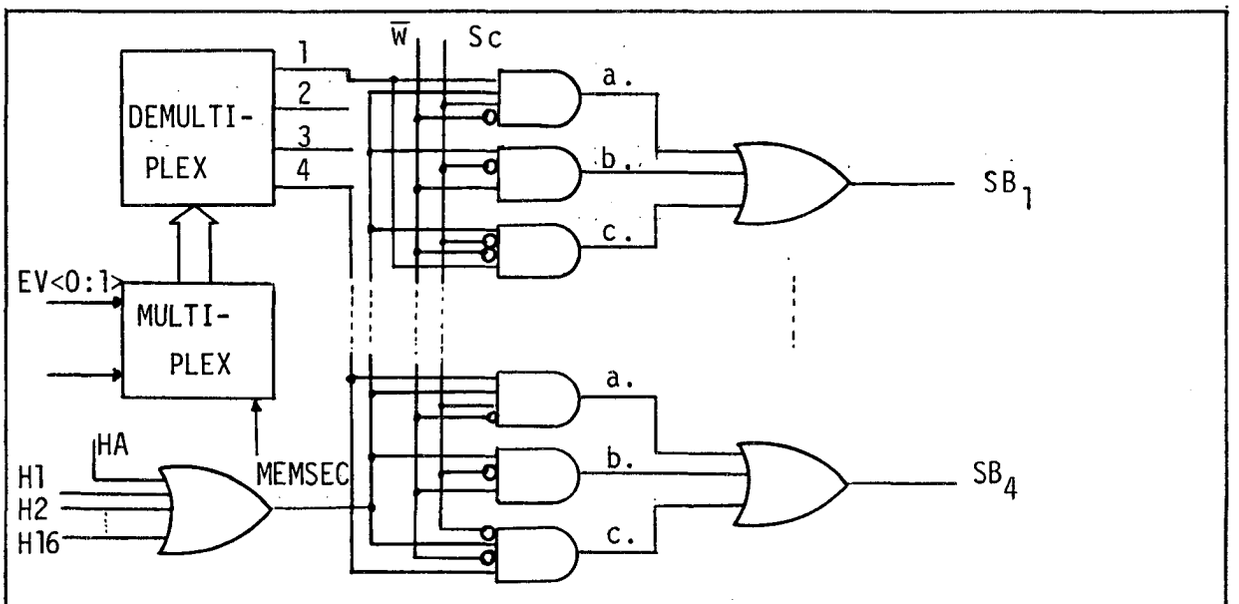


Figura IV-10 - Mecanismo de Seleção dos Blocos da Memória Principal

3.2. *Diagramas de Tempo para Acessos à Memória Principal*

A verificação do tempo de acesso à memória principal, será feita de modo análogo ao utilizado no caso do cache. São apresentados dois diagramas de tempo, uma para os casos em que o endereço real da página está em um registrador e outro para os casos em que está na memória local.

. Diagrama de tempo para o caso em que o endereço real da página está em um registrador.

Na Figura IV-11, tem-se o diagrama de tempo, para leitura em uma página da memória principal.

. Descrição dos Tempos Indicados na Figura IV-11.

tempo	descrição	duração(ns)
t_1	tempo necessário para garantir que o endereço está estável	10
t_2	tempo para seleção do processo	23
t_3	tempo para que a saída do comparador esteja válida	11
t_4	tempo para que a saída do comp. de end. virtual esteja válida	15.3
t_5	tempo para habilitação do registrador	20.7
t_6	tempo para obtenção do endereço real	19
t_7	tempo para obtenção do sinal selecionador de bloco	22.7
t_8	tempo para um acesso à memória principal	120
t_9	tempo para garantir que o dado foi lido	0

Para a obtenção do tempo de acesso, devem-se somar os tempos onde não existe sobreposição, obtendo um tempo de acesso de 201.7 nano segundos.

O diagrama de tempos para escrita na memória principal é análogo ao apresentado na Figura IV-11, sendo que o tempo de acesso é praticamente o mesmo.

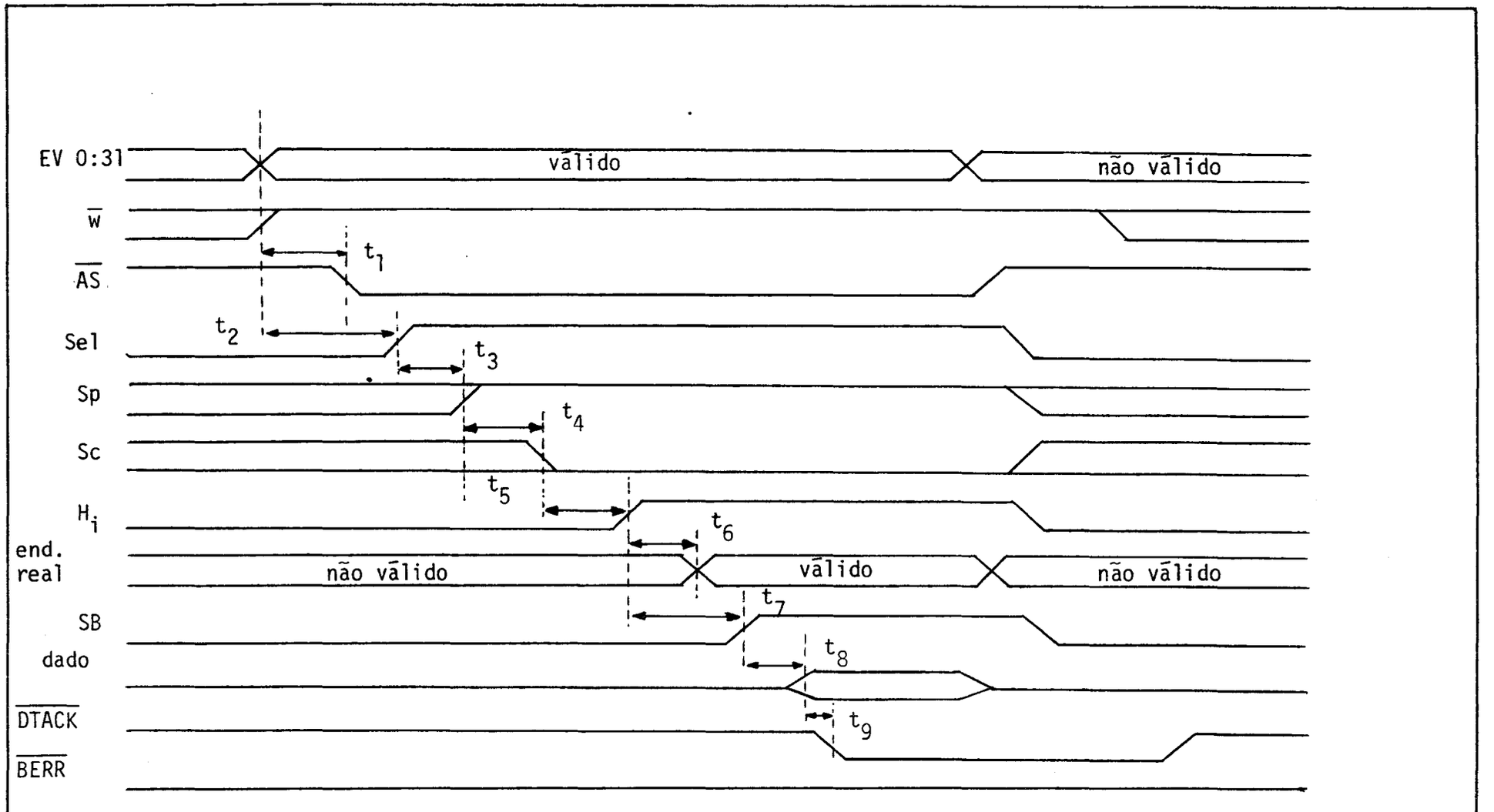


Figura IV-11 - Diagrama de Tempo de uma Leitura na Memória Principal, Quando o Endereço Estiver em um Registrador

. Diagrama de tempo para acessos à memória principal, quando o endereço real da página está na tabela de páginas. na Figura IV-12., tem-se o diagrama de tempo para leitura em uma página que está na memória principal e seu endereço real está na tabela de páginas.

. Descrição dos Tempos Indicados na Figura IV-12

tempo	descrição	duração(ns)
t_1	tempo necessário para garantir que o endereço está estável	10
t_2	tempo para que a saída do comparador de página esteja válida	34
t_3	tempo para que a saída do comparador de endereço virtual esteja válida	38.3
t_4	tempo para a obtenção do endereço da entrada da TPV na memória local	34
t_5	tempo de acesso à memória local para obtenção da entrada da TPV	120
t_6	tempo para habilitar o acoplador	53.4
t_7	tempo para obtenção do endereço real	11
t_8	tempo para obtenção do sinal selecionador de bloco	22.7
t_9	tempo para um acesso à memória principal	120

O tempo de acesso para esse caso é de 380.1 nano segundos, considerando as superposições já citadas.

Também nesse caso, o diagrama de tempo para escrita na memória principal é análogo ao apresentado para leitura, com tempo de acesso praticamente igual.

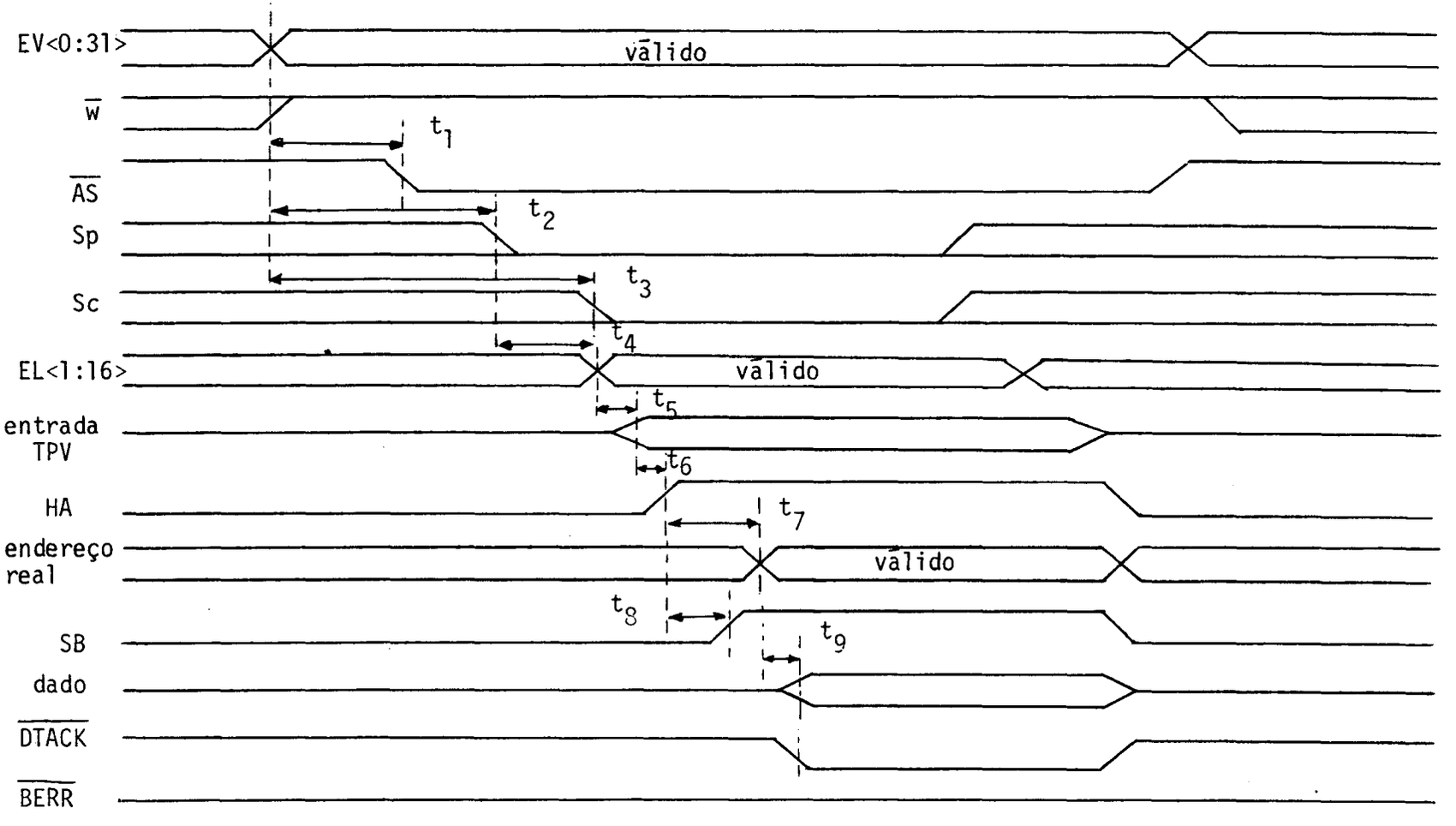


Figura IV-12 - Diagrama de Tempos de Uma Leitura na Memória Principal, Quando o Endereço Real da Página Estiver na Tabela de Páginas

3.3. *Justificativas de Algumas Características Adotadas Para a Memória Principal*

. Sobre o tamanho da palavra da Memória Principal do SDF

A opção de endereçamento de palavras com 64 bits na memória principal do SDF, traz grandes vantagens quando o cache deve ser alterado. Nessa operação, basta endereçar a memória, desprezando-se os dois bits menos significativos do endereço e habilitando-se todos os selecionadores de bloco. Dessa forma, obtém-se os quatro dados que preencherão o cache.

Para acessos a palavras de 16 bits, a organização proposta não apresenta dificuldades, uma vez que o dado pode ser obtido selecionando-se apenas um dos blocos de memória.

. Sobre a manutenção de uma estatística dos acessos ao módulo de memória

Um acesso ao módulo de memória gasta em média 200 ns, quando o endereço requisitado não está na memória secundária. Para manter-se uma estatística dos acessos ao módulo através de um software são necessários alguns μs, em cada acesso, para as atualizações. Isso torna inviável a manutenção de uma estatística por software, uma vez que não existe tempo disponível, para as atualizações entre um acesso e outro.

Decidiu-se então adotar um algoritmo de reposição que não fosse baseado na utilização das páginas. Optou-se pela implementação de uma fila e assim, a página que está há mais tempo na memória é a candidata a ser retirada.

Uma forma de melhorar a taxa de acerto na retirada de páginas da memória principal, é manter uma estatística das páginas que tiveram que retornar para a memória principal. Essas páginas não deveriam ter sido levadas para a memória secundária, uma vez que foi necessário trazê-las novamente.

Assim, o algoritmo para reposição de páginas na memória principal, baseia-se na retirada de páginas da memória secundária e no tempo que uma página já permaneceu na memória principal.

. A utilização de software nos acessos à memória principal

Uma possibilidade para os acessos ao módulo de memória virtual é a de que todo acesso que dependesse da memória local, fosse feito através do microprocessador por software. Dessa forma, ficaria bem simples a obtenção do endereço real de uma página na TPV. No entanto, o tempo gasto para os acessos às páginas da memória principal, seria excessivamente grande: seriam necessários 38 pulsos de relógio para o microprocessador 8088 (por exemplo) fazer o controle da operação, isto é, 8 µsegundos para uma frequência de relógio de 4.77 MHz.

Optou-se por fazer todo o acesso por hardware e utilizar o microprocessador apenas na atualização da TPV, no caso de escritas na memória virtual. Dessa forma, o acesso pode ser atendido em menos de 400 ns, como visto na seção anterior, e o software para atualização é executado após a liberação do duto, quando for necessário.

Portanto, em acessos à memória principal, a utilização do software restringe-se aos casos de escrita na memória principal. Para esses acessos é necessário indicar na entrada da TPV correspondente à página a que se teve acesso, que ela foi modificada.

4. *MEMÓRIA SECUNDÁRIA*

4.1. *Descrição do Esquema Proposto*

Em IV.3.1, descreve-se como o módulo de memória, trata um acesso a uma página que está na memória principal (TPV <15> = 1). Nesta seção é apresentado o procedimento para os casos em que a página a que se quer ter acesso, está na memória secundária.

Nesse caso, o endereço requisitado não é encontrado no cache, a página a que pertence não se encontra no registrador que contém a última página lida e a entrada da TPV, relativa a página do endereço requisitado, indica no bit 15, que a página não está na memória principal. O acesso deve então entrar na fila de atendimento do microprocessador, para que esse se encarregue de fazer a troca de páginas, entre memória principal e secundária. O esquema utilizado para esse procedimento, está apresentado na Figura IV-13.

Após a obtenção da entrada da TPV relativa a página a que se quer ter acesso, da maneira já descrita na seção anterior, deve-se ter TPV<15> = 0, o que indica que a página não está na memória principal. Isso deve ocasionar uma interrupção no microprocessador, caso não tenham ocorrido problemas de proteção, para que a troca de páginas seja colocada na fila para atendimento.

Como visto na seção 5 do capítulo III, toda tarefa que chega ao microprocessador é colocada em uma fila, sendo atendida segundo uma prioridade pré-estabelecida. No atendimento de troca de páginas, o processador encarrega-se de controlar todo o processo de troca, determinação de endereço e atualização de tabelas. Após completada a tarefa, o microprocessador envia um sinal (MEMSEC), indicando que o endereço real da página já está disponível. Os procedimentos do microprocessador são detalhados em IV-4.2.

Como a memória local será utilizada pelo processador e pelo controlador de memória, deve-se ter um demultiplex que controle qual endereço será utilizado em um determinado acesso e um multiplex que envie o dado para quem solicitar.

Esse controle pode ser feito por uma linha de solicitação, controlada pelo microprocessador, que seleciona o ende-

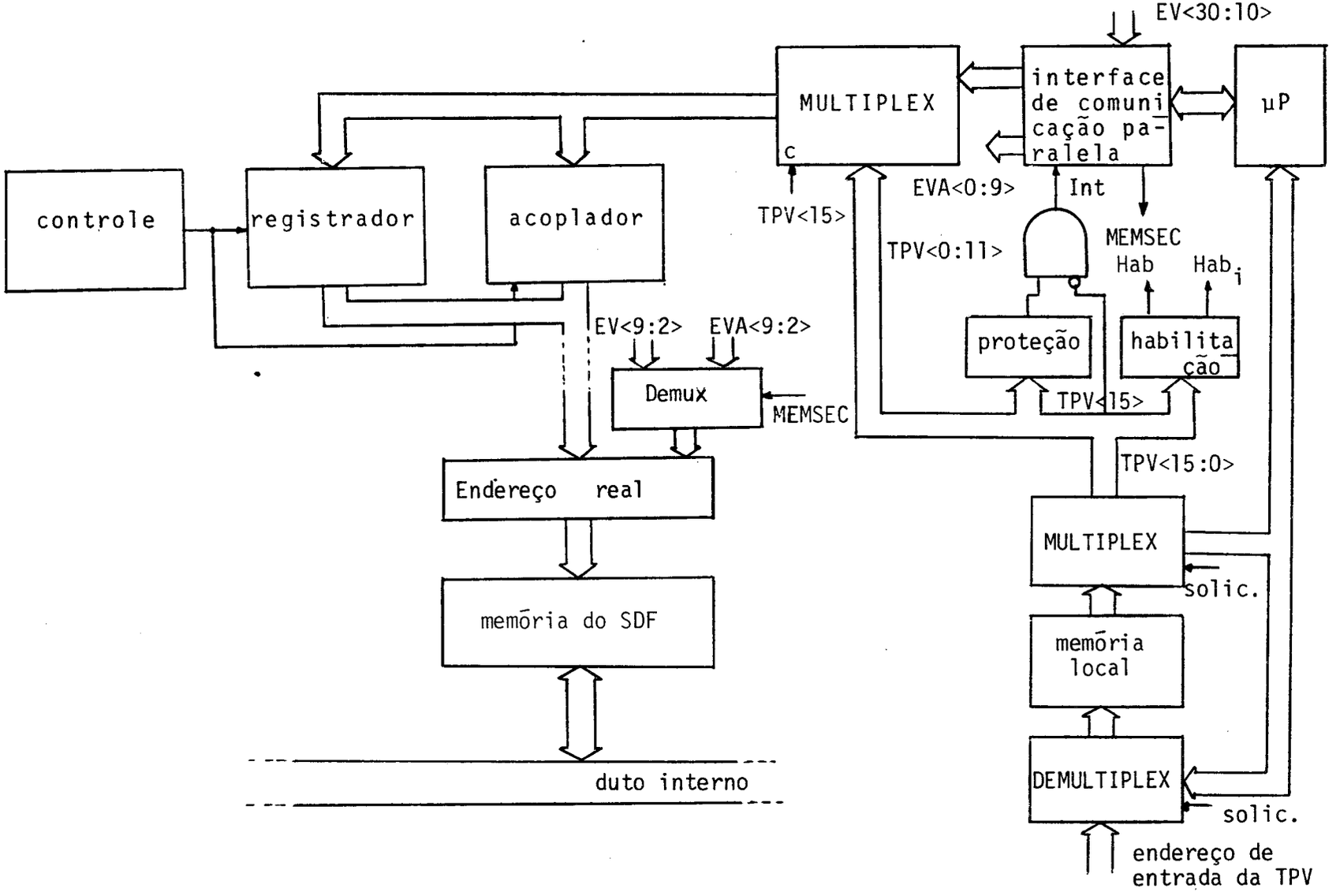


Figura IV-13 - Controle Para a Transferência de Páginas entre as Memórias Principal e Secundária

reço enviado, quando necessita de um acesso à memória local e em caso contrário, libera a memória para acessos do controlador.

É importante lembrar que o microprocessador só é acionado para uma troca de páginas, se o acesso que provocou a troca não apresenta problemas de proteção. Isso é possível porque, mesmo quando a página não está na memória principal, tem-se na TPV o tipo de acesso possível.

4.2. *Software Envolvido nas Trocas de Páginas*

Uma ausência de página na memória principal, provoca uma interrupção no processador, sendo executada uma rotina, cujo algoritmo é mostrado a seguir:

. guardar os dados do acesso em uma fila.

. colocar a tarefa na fila para atendimento, com prioridade 4.

. liberar o duto para outras transferências.

Como pode ser observado no algoritmo apresentado, quando o microprocessador é interrompido para que seja colocado em sua fila, um acesso à memória secundária, executa-se uma rotina que salva os dados relativos ao acesso, e, após colocar a tarefa na fila, libera o duto.

Esse procedimento evita que o duto fique bloqueado por um grande espaço de tempo, tornando possível outras transferências pelo duto e até acessos ao módulo de memória, enquanto o acesso à memória secundária é executado.

Após liberado o duto, pode ser feito qualquer tipo de acesso ao módulo de memória virtual. Para acessos ao cache e memória principal, o módulo fornece respostas normalmente como mostrado nas seções anteriores. Os acessos à memória secundária, serão colocados na fila e atendidos conforme a disponibilidade.

Para os casos de acessos para leitura, após feita a troca de páginas, o módulo de memória endereça o módulo que requisitou a leitura, para completar o acesso. Nos casos de escrita, o dado e a posição onde ele deve ser armazenado são colocados na fila e o módulo requisitante é liberado.

As justificativas da liberação do duto, são apresentadas no capítulo 5 e como o módulo requisitante espera pelo resultado do acesso, é mostrado em IV-4.4.

Quando o processador termina uma tarefa, procura na fila de tarefas pendentes, a de prioridade mais alta.

Quando a tarefa a ser atendida for uma troca de página, a rotina a ser executada pelo processador pode ser dividida em duas partes: uma antes de obter o endereço requerido e uma que pode ser executada após a execução do acesso.

. Rotina que possibilita a execução ao acesso

Esta rotina deve obter espaço na memória principal e, se for o caso, trazer para ela a página a que se quer ter acesso. Um algoritmo simplificado para esta rotina é mostrado a seguir:

- verificar se a página ainda não está na memória principal,
- determinar a página a ser retirada da memória principal, utilizando as filas implementadas,
- verificar o bit de modificação (TPV<14>) da página a ser retirada.
  se TPV<14> = 1 a página foi modificada, então guardá-la na memória secundária,
- liberar o endereço real da página, através da interface de comunicação paralela,
- liberar o endereço e o dado, guardados antes da liberação do duto,
- esperar que não esteja havendo acesso ao módulo,
- obter o duto para enviar a resposta do acesso, no caso de leituras,
- acionar a execução do acesso, através da MEMSEC,
- liberar o duto, no caso de leitura,
- executar rotina de atualização das tabelas.

No algoritmo apresentado, deve-se inicialmente verificar se a página não está na memória principal pois, essa transferência pode ter sido efetuada entre a colocação dessa tarefa na fila e o seu atendimento. Um segundo passo a ser executado é a determinação da página a ser retirada da memória principal. Utiliza-se para isso, a estatística apresentada em IV.3 ,

implementada com as estruturas de dados estipuladas no item c da seção III-5.

. Rotina para atualização das tabelas

Um algoritmo simplificado para execução desta rotina é mostrado a seguir:

- atualizar o campo de validade da TPV, nas entradas relativas às páginas retirada e trazida para a memória principal,
- atualizar o campo de endereço real da TPV, na entrada relativa à página trazida para a memória principal,
- se o acesso foi para escrita, atualizar o campo de modificação da TPV,
- atualizar o campo de página virtual da TPR,
- atualizar a fila para retirada de elementos da memória principal,
- atualizar o número de vezes que a página foi retirada da memória secundária.

A rotina para atualização das tabelas, será executada após a liberação do módulo que requisitou o acesso, mas antes de liberar o processador para atender uma nova tarefa da fila para atendimento.

4.3. *Diagrama de Tempo Para Acessos à Memória Secundária*

No caso de acessos à memória secundária, tem-se três etapas envolvidas:

a. Hardware - circuitos que verificam que a página a que se quer ter acesso, não está na memória principal, ocasionando uma interrupção no microprocessador que fazem o acesso à memória do SDF após a liberação do endereço real. Essa etapa provoca um atraso na resposta que é denominado neste trabalho, tempo de hardware (thard).

b. Software - execução de rotinas que gerenciam a troca de páginas, ocasionando um atraso que é denominado neste trabalho, por tempo de software (tsoft).

c. Acesso à memória secundária - como a página a que se quer

ter acesso não está na memória principal, é necessário um acesso à memória secundária, para transferência da página. Em alguns casos é necessário armazenar a página que será retirada da memória principal, na memória secundária.

A Figura IV-14 mostra o diagrama de tempos para um acesso onde não é necessário guardar a página que está sendo retirada da memória principal, na memória secundária. Esse diagrama detalha os tempos gastos no hardware e localiza os tempos de software e de acesso à memória secundária.

Descrição dos Tempos Indicados na Figura IV-14.

tempo	descrição	duração(ns)
t_1	tempo necessário para garantir que o endereço esta estável	10
t_2	tempo para que a saída do comparador de páginas esteja válida	34
t_3	tempo para que a saída do comparador de endereço virtual esteja válida	38.3
t_4	tempo para obtenção do endereço da entrada da TPV na memória local	34
t_5	tempo de acesso à memória local para obtenção da entrada da TPV	120
t_6	tempo para habilitação da interrupção	19.5
t_7	tempo gasto no software e em acessos à memória secundária	tsoft + tms
t_8	tempo para habilitar o acoplador	41.7
t_9	tempo para obtenção do endereço real	11
t_{10}	tempo para obtenção do sinal selecionador de bloco	22.7

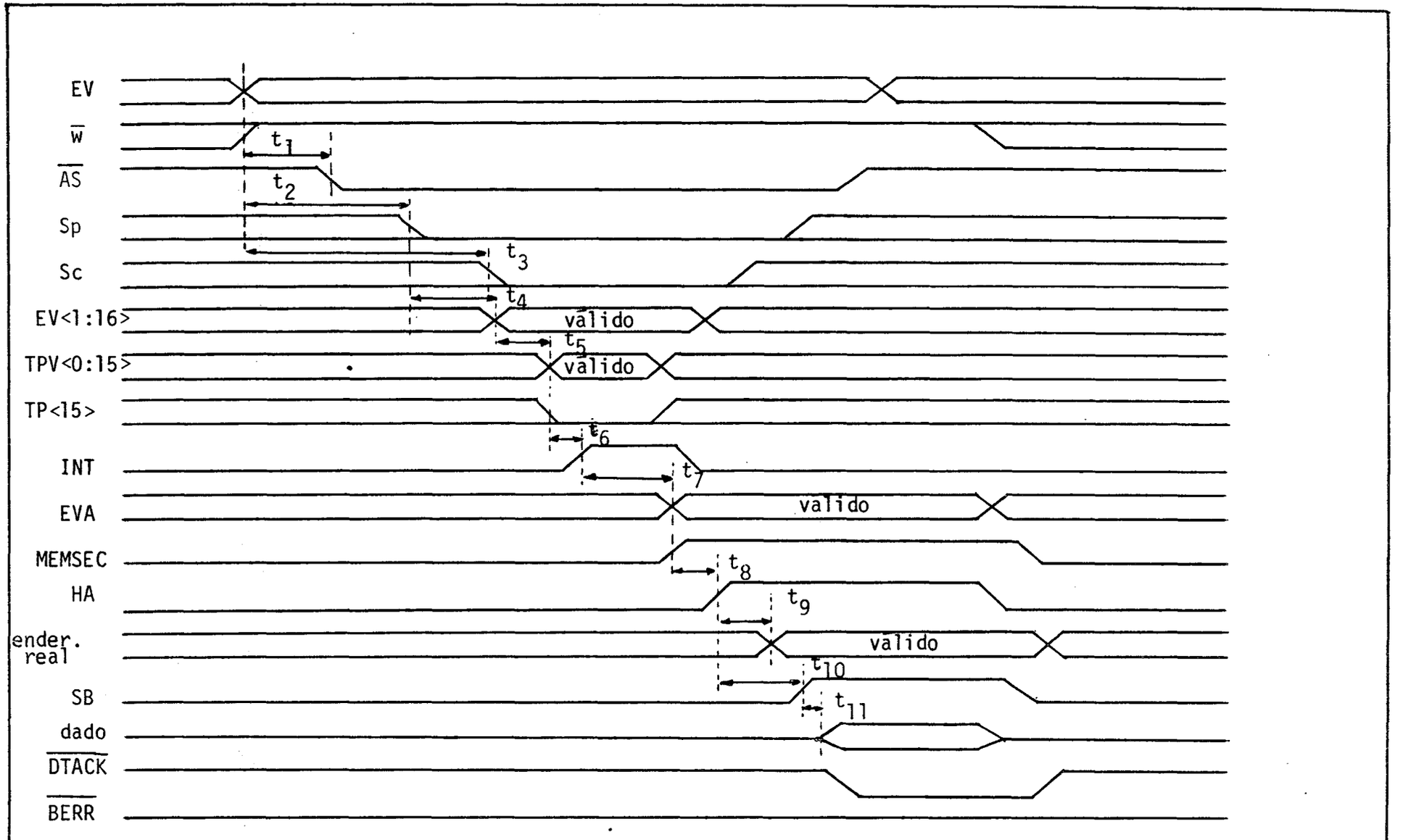


Figura IV-14 - Diagrama de Tempo de uma Leitura com Acesso à Memória Principal

tempo	descrição	duração(ns)
t_{11}	tempo para um acesso à memória principal	120

Somando-se os tempos de hardware, onde não existem sobreposições, obtém-se:

thard = 387.9 nano segundos

. Tempo de Software

O tempo de software é o tempo gasto na execução das rotinas para tratamento de acessos a memória secundária, cujos algoritmos são apresentados em 4.2. Uma forma de avaliar esse tempo, é verificando-se o número de pulsos de relógio necessários para executar cada rotina, em um determinado processador.

Para avaliar o tempo necessário para uma troca de páginas, determina-se o tempo de execução de cada rotina, através do código necessário para executá-la no microprocessador 8086. A partir do código das rotinas obtém-se o número de pulsos de relógio necessários e, considerando-se uma frequência de 4.77 MHz, obtém-se o tempo desejado. Os valores obtidos para as rotinas de acesso à memória secundária são:

- software anterior à liberação do dado: número de pulsos de relógio necessários = 1331; tempo gasto na execução = 279.04 µs.
- software posterior à liberação do dado: número de pulsos de relógio necessários = 2030; tempo gasto na execução = 425.58 µs.

Assim, uma estimativa do tempo de software é de tsoft = 279.04 µs, uma vez que o tempo gasto com o software após a liberação do dado, pode ser sobreposto com outros tempos.

. Tempo para Acessos à Memória Secundária.

O tempo necessário para acessos à memória secundária depende largamente do dispositivo utilizado, tendo-se normalmente, três tempos envolvidos:

a. tempo médio para posicionamento na trilha a que se quer ter acesso,

b. tempo médio de latência,

c. tempo para a transferência das informações.

Os tempos a e b são constantes para cada dispositivo e expressos normalmente, em milisegundos.

O tempo c depende da quantidade de informação transferida e é expresso em número de bits transferidos por segundo.

Para avaliar-se o tempo gasto na transferência de uma página para o disco, considera-se, por exemplo, um Winchester (BR-412 da Flexidisk) de 10 Mbytes.

Para esse disco tem:se:

- tempo médio de acesso = 85 ms
- tempo médio de latência = 8.33 ms
- taxa de transferência=5 Mbits/seg

Como o tamanho das páginas proposto neste trabalho é de 1024 palavras, isto é, 2048 bytes, o tempo de transferência será:

t_t = 3.3 ms

Para esse exemplo tem-se que o tempo necessário para transferir uma página é de 96.63 ms.

Com esse exemplo, pode-se observar que nos casos em que são necessários acessos à memória secundária, o tempo gasto com o hardware e com o software são desprezíveis.

4.4. *Justificativas de Algumas Características Adotadas Para o Gerenciamento da Memória Secundária*

. Transferência de páginas para a memória secundária antecipadamente

Sugere-se neste trabalho, que quando não existirem tarefas a serem executadas, o microprocessador utilize o tempo disponível para transferir páginas, que já foram modificados, para a memória secundária.

O tempo gasto nesta tarefa, praticamente não tem influência nos tempos de acesso e evita, em alguns casos, a necessidade de transferir uma página para a memória secundária, durante um acesso, o que acarretaria uma grande perda de tempo, para o processo que requisitou o acesso e para outros que possam estar na fila para atendimento.

. Mapeamento Direto

Sugere-se neste trabalho, um mapeamento direto entre o endereço virtual e a posição da informação na memória secundária. Essa característica apresenta as seguintes vantagens, em relação ao mapeamento indireto:

a. não é necessário o uso de tabelas para o mapeamento, simplificando as transferências entre memória principal e secundária.

b. no mapeamento indireto, toda página que for retirada da memória principal, pode ser transferida para o disco, mesmo que não tenha sido modificada. Pois, uma outra página pode ter sido armazenada em seu lugar na memória secundária. No mapeamento direto isto não é necessário uma vez que cada página virtual tem seu espaço reservado na memória secundária.

Desvantagens do mapeamento direto:

a. a capacidade da memória secundária deve ser igual a da memória virtual. Em mapeamento indireto pode se mapear um espaço de endereçamento maior em uma memória secundária menor . Isso porém, apresenta a desvantagem de não se poder utilizar todo o espaço de endereçamento virtual.

b. no mapeamento indireto é possível minimizar o tempo de acesso à trilha no disco, escolhendo-se a posição para gravar as páginas, o mais próximo possível da posição da cabeça.

Apesar das desvantagens apresentadas, considera-se que o mapeamento direto é mais adequado, devido à maior facilidade de implementação e à possibilidade de se fazer acessos a todo o espaço de endereçamento virtual.

. Liberação do duto em acesso à memória secundária

Como visto em 4.1, sempre que forem necessárias transferências de páginas entre memória principal e secundária , para a execução de uma leitura, o duto deve ser liberado.

Antes da liberação do duto, deve-se enviar um sinal ao módulo que requisitou o acesso, informando sobre a necessidade de espera para obtenção da página na memória secundária . Tem-se portanto, os seguintes passos na liberação do duto.

a. o módulo de memória verifica que a página não está na memória principal e gera uma interrupção no microprocessador,

b. este sinal de interrupção é enviado ao módulo que requisitou a leitura acionando um biestável que irá gerar pulsos de espera no microprocessador do módulo, que ficará em estado de espera enquanto não receber um novo sinal, informando que o dado já está disponível,

c. quando é completada a transferência da página requisitada, o microprocessador gera o sinal MEMSEC que é enviado ao módulo que requisitou a leitura, fazendo com que ele saia do estado de espera para receber o dado.

O mecanismo utilizado para a liberação do duto é apresentado na Figura IV-15.

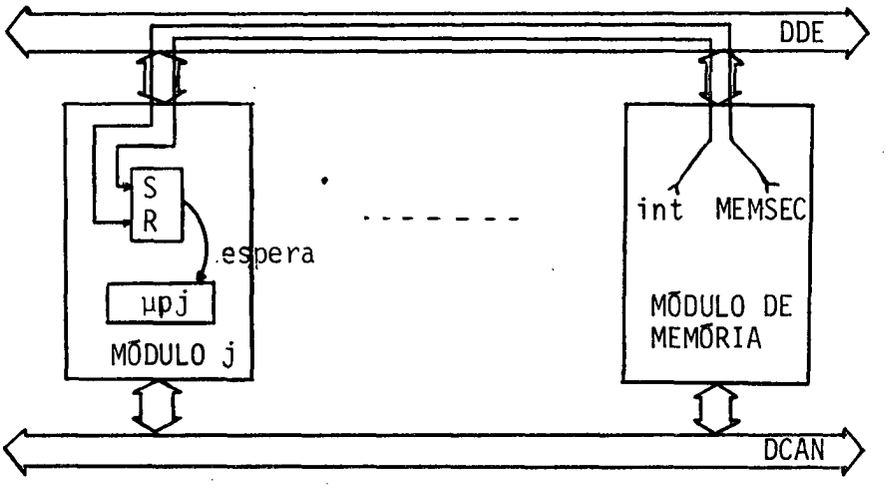


Figura IV-15 - Mecanismo Utilizado na Liberação do Duto

Se durante a espera do dado requisitado, um terceiro módulo quiser fazer uma leitura no módulo em espera, esse acesso pode ser completado em problemas, uma vez que para esse tipo de acesso, pode-se implementar um sistema de exclusão mútua (SEM) que coloca o microprocessador do módulo de onde será lida a informação em estado de espera, deixando a memória livre para leitura. Assim, no caso do microprocessador já estar esperando pelo dado requisitado, tem-se novos pulsos de espera que não modificarão o seu estado. O diagrama de blocos desse sistema, em um módulo genérico, é mostrado na Figura IV-16.

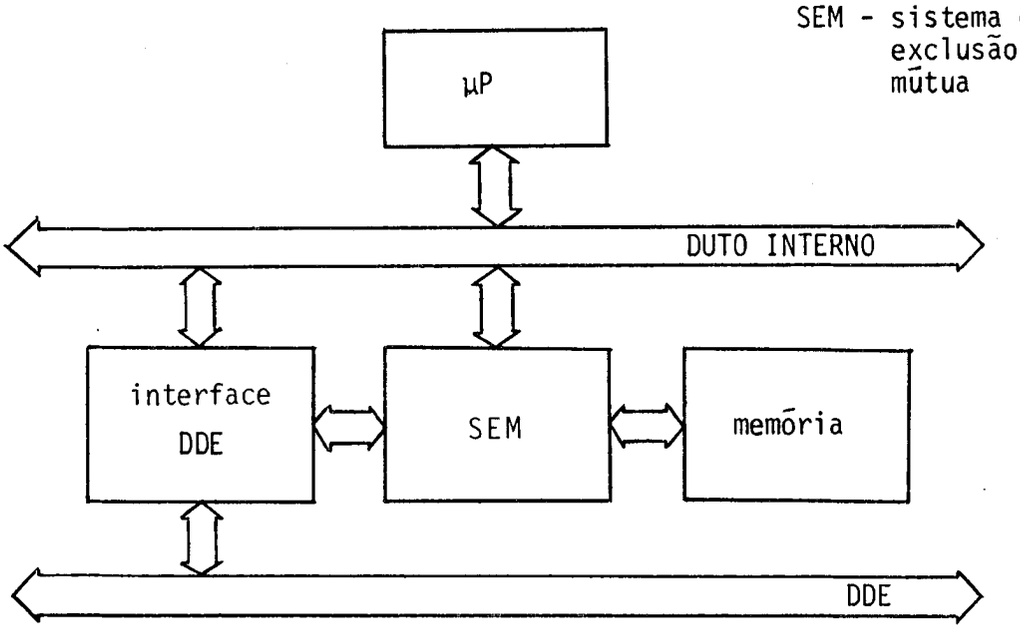


Figura IV-16 - Mecanismo para Utilização da Memória de um Módulo

5. *MEMÓRIA LOCAL*

Nesta seção é apresentada uma descrição do hardware e do software da memória local do módulo de memória virtual. Como visto em seções anteriores a memória local armazena todas as tabelas e rotinas necessárias para o gerenciamento da memória do SDF, sendo que tanto o módulo de memória quanto os outros módulos do sistema podem ter acesso a ela.

5.1. *Acessos à Memória Local*

Na seção 4 deste capítulo descreve-se como o módulo de memória, faz acessos à memória local. A maneira como outros módulos do sistema podem endereçá-la é descrito nesta seção.

Os acessos à memória local são identificados por um endereço virtual, que contenha o bit mais significativo igual a zero. Esse bit deve ser seguido do processo que quer fazer o acesso (EV<27:30>) e do endereço requerido (EV<26:0>).

O tipo de acesso permitido depende do que for estipulado na inicialização do processo. A memória local pode ser acessível para leitura e escrita, só para leitura, só para escrita ou não ser acessível para um dado processo. Os tipos de acessos permitidos para um processo, são armazenados em uma memória associativa, para tornar mais rápido o acesso.

O esquema para acessos à memória local está representado na Figura IV-17. Como pode ser observado, se for requisitado um acesso à memória local e esse for permitido, a linha de acessos de outros módulos (AEX), torna-se igual a 1 e endereça a memória local com o endereço obtido no duto interno. Da mesma forma que o endereço, o dado envolvido no acesso tem seu caminho estipulado através da linha AEX e da linha SOLIC, já descrita na seção anterior.

5.2. *Software Mantido na Memória Local*

Aqui são apresentados algoritmos simplificados das rotinas que compõe o software necessário, para o atendimento dos acessos feitos ao módulo de memória virtual.

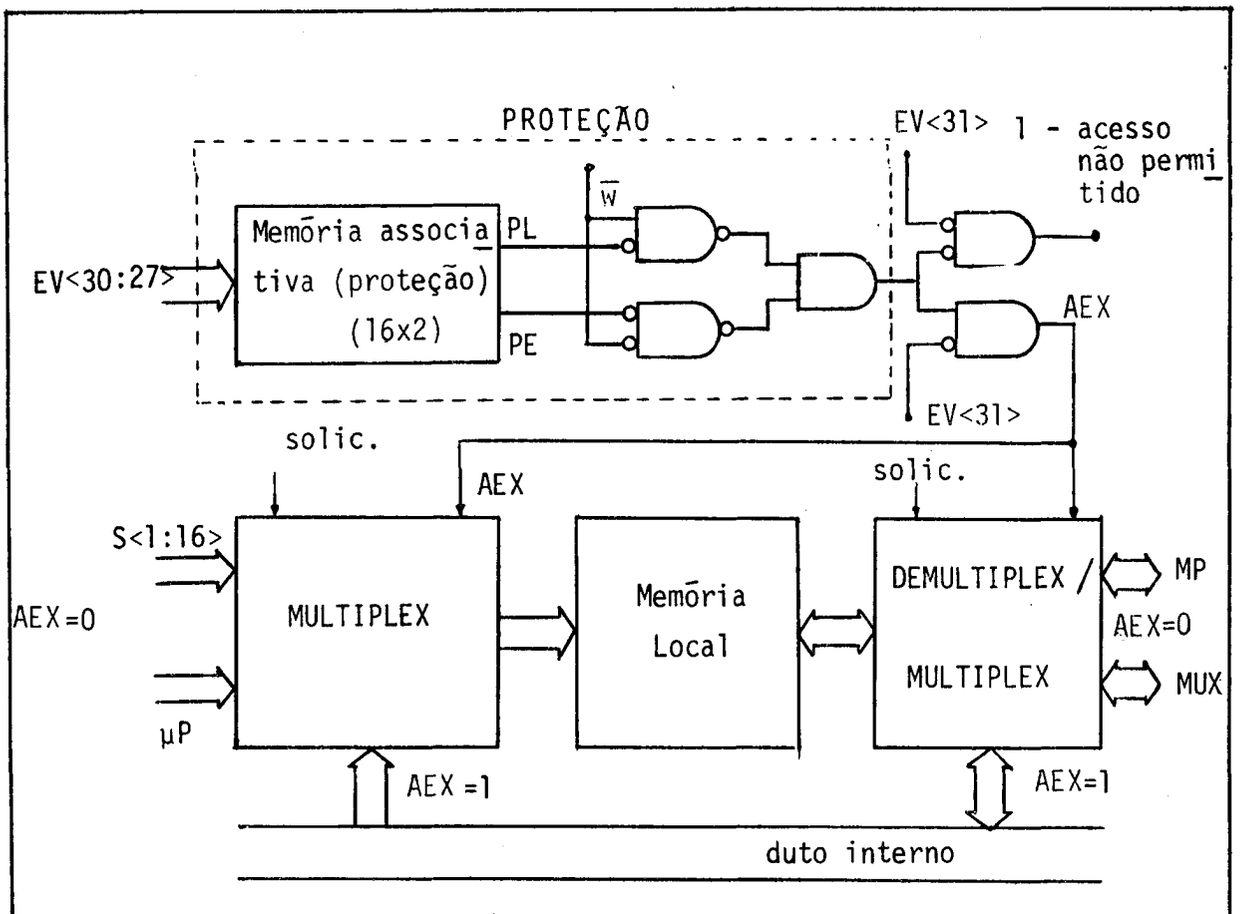


Figura IV-17 - Acessos à Memória Local

5.2.1.*Inicialização do Sistema*

Na inicialização do sistema, deve-se executar o seguinte algoritmo:

a. verificar a memória principal e dividir as páginas por processos através das TPRs,

b. inicializar registradores com endereço do cache e da última página lida, com valores não válidos,

c. alocar a TPV de cada processo na memória local e iniciali - zar os registradores que contêm o endereço básico da tabela de cada processo,

d. invalidar o campo de endereço real da TPV.

No primeiro passo desse algoritmo deve-se alocar as páginas reais para os processos. Essas páginas poderão ser re

manejadas para outros processos, como será detalhado no próximo ítem. A quantidade de páginas alocadas para cada processo pode variar, dependendo da necessidade de cada um e deve ser pré-estabelecida.

No passo b. os registradores que contêm o endereço do cache e da última página lida são inicializados, para que não ocorram coincidências no primeiro acesso ao processo, podendo acarretar erros.

No passo c. as TPV são alocadas na memória local e os registradores contendo seus endereços são, inicializados. É necessária a alocação das TPV, para que seja possível a invalidação do campo de endereço real das entradas das TPV. Essa invalidação torna possível verificar se uma página, já teve algumas acesso.

5.2.2. *Atendimento de Tarefas*

Como visto em seções anteriores, o microprocessador do módulo de memória deve atender a tarefas que ficam em uma fila e que apresentam prioridades. Assim, toda vez que o processador terminar o atendimento de uma tarefa, deve verificar nas filas de mais alta prioridade a existência de outras tarefas a serem executadas. A seguir são detalhadas as tarefas que o microprocessador deve atender.

. Tratamento de Erros

Toda a vez que for gerado pelo Hardware algum tipo de erro deve-se:

a. reconhecer o erro,

b. enviar mensagem apropriada ao módulo que requisitou o acesso,

c. se for erro de memória principal, memória secundária ou qualquer outro tipo de erro no sistema, enviar mensagem ao módulo que monitora o sistema.

d. em casos de erro de memória, retirar a página que contêm o endereço da TPR.

Para cada tipo de erro possível de ocorrer, devem-se ter procedimentos e mensagens apropriadas.

. Finalização de um processo

A finalização de um processo é solicitada, ao módulo de memória virtual, através de um comando de alto nível, enviado pelo DCAN. Nesse caso, deve-se providenciar para que uma nova inicialização dos parâmetros referentes a esse processo (tal como executado na inicialização do sistema) seja efetuada. O algoritmo básico para este caso é:

a. tornar as páginas do processo liberadas, através da TPR e da TP,

b. reinicializar os registradores com endereço do cache e da última página lida, do processo,

c. invalidar o campo de endereço real da TPV do processo.

. Modificação na TPV em escritas na Memória Principal

Em toda escrita na memória principal, deve-se fazer o bit de modificação da TPV (TPV<14>) igual a 1.

. Acessos à Memória Secundária

As rotinas necessárias para a manutenção da memória secundária, foram detalhadas na seção IV.4.2.

. Inicialização de um Processo

Para inicialização de um processo, deve-se receber um comando de alto nível através do DCAN, solicitando a inicialização e informando onde os dados necessários para essa inicialização podem ser obtidos.

Os dados que devem ser transferidos ao módulo de memória para inicializações são:

a. processo onde ocorrerá a inicialização

b. intervalo a ser inicializado

c. proteção das páginas a serem inicializadas

d. informação sobre a possibilidade de se guardar páginas desse processo na memória secundária. No caso de não poder, de

ve ser informado quais processos poderão ter suas áreas ocupadas, pelo processo que está sendo inicializado.

Na inicialização de um processo, deve-se colocar o maior número de páginas possível na memória principal. Se ainda assim, não for possível colocar todas as páginas a serem inicializadas na memória principal, deve-se alocar o restante na memória secundária se não houver proibições.

Podem existir processos que necessitem que todas as suas páginas estejam na memória principal. Neste caso o módulo de memória deve ser sinalizado através dos dados da inicialização. Além de indicar a proibição de transferência de páginas para a memória secundária, deve-se ainda ter informações adicionais sobre quais áreas, pertencentes a outros processos, podem ser utilizadas no caso em que o número de páginas a serem inicializadas, exceder o espaço de memória reservado ao processo em questão.

Tem-se portanto, o seguinte algoritmo para a inicialização:

a. se a memória destinada ao processo a ser inicializado estiver ocupada (TP<5> = 1)

   Então

   a'. se existir algum processo não inicializado
       Então transferir as páginas que estão ocupando a área do novo processo para esta outra área.
       senão transferir as páginas para a memória secundária.

b. se o processo a ser inicializado puder ter página na memória secundária
   Então
   a'. se o espaço disponível para o processo não for suficiente para as páginas a serem inicializadas
       Então
         verificar se cabem em espaços de processos não inicializados

   b'. alocar as páginas que forem possíveis na memória principal e as restantes, se houver, na memória secundária

   Senão

   a'. se as áreas de outros processos, que o processo requisi

tante indicou como área de seu uso, for suficiente.
Então
alocar as páginas a serem inicializadas.
Senão
enviar mensagem para o módulo que requisitou a inicialização, indicando a falta de espaço na memória principal.

c. atualizar a TPV, indicando a proteção de cada página inicializada, sua localização e inicializar o bit de modificação com falso,

d. atualizar a TPR, indicando as páginas reais utilizadas e quais páginas virtuais as ocuparam,

e. inicializar a estatística de retirada de elementos da memória secundária.

Como no passo a. do algoritmo de inicialização, tem-se variações na TPV de outros processos, é necessário bloquear-se os acessos aos módulos de memória, durante essa etapa da inicialização.

. Transferência de Páginas para a Memória Secundária

Sempre que a fila de tarefas a serem atendidas estiver vazia, o processador deve transferir páginas que já foram modificadas, para a memória secundária. Deve portanto executar o seguinte algoritmo:

a. determinar, em cada transferência, os processos que estão utilizando a memória secundária,

b. verificar a fila que contém as páginas com mais baixa prioridade, para serem mantidas na memória principal,

c. verificar qual a primeira página da fila que foi modificada,

d. tornar zero o bit de modificação da página determinada no item c,

e. transferir a página para a memória secundária.

6. *CONSIDERAÇÕES FINAIS*

Com base na descrição do módulo de memória e nas justificativas apresentadas neste capítulo, destacam-se os seguintes pontos:

a. existência de um cache com as seguintes características:
   - . tempo de acesso da ordem de 90 ns,
   - . divisões por processo,
   - . linha alterada apenas com leituras,
   - . escrita na memória durante a modificação,
   - . linhas de 64 bits (4 palavras).

b. memória principal do SDF com as seguintes características:
   - . registrador contendo o endereço real da última página lida,
   - . acessos à memória principal utilizando software apenas para atualização da TPV, em casos de escrita,
   - . tempo de acesso da ordem de 200 ns, quando o endereço real estiver no registrador e de 380 ns, se não estiver,
   - . algoritmo FIFO para reposição de páginas,
   - . as páginas são alocadas na memória principal, durante a inicialização do processo.

c. memória secundária com as seguintes características:
   - . tempo de acesso da ordem de alguns mili-segundos,
   - . rotina que transfere páginas modificadas para a memória secundária, antecipadamente,
   - . liberação do duto no caso de acessos à memória secundária,
   - . software dividido em anterior e posterior à liberação do dado, para diminuir o tempo de acesso,
   - . estatística de retirada de páginas da memória secundária.

d. proteção: implementada através do hardware.

e. memória principal dividida em:
   - . memória do SDF - com endereçamento virtual e dividida em

4 blocos, para facilitar transferências para o cache,

. memória local - contendo as tabelas com informações sobre a memória do SDF e as rotinas necessárias para o gerenciamento do módulo de memória virtual. Acessível tanto pelo módulo de memória como por outros módulos do SDF.

Além das características citadas, existem alguns pontos que ainda devem ser discutidos:

. Possibilidade de uso de circuitos integrados que permitem endereçamento virtual.

Os circuitos integrados que permitem endereçamento virtual encontrados no mercado, o MC 68010 por exemplo, são microprocessadores que reconhecem uma ausência de página na memória principal, quando estão executando alguma rotina. Esses processadores são capazes de fazer todo o tratamento da ausência detectada.

O que torna inviável a utilização desses circuitos integrados no módulo de memória virtual, é que eles apenas podem resolver uma ausência de página, detectada por eles, durante a execução de uma rotina. No caso do módulo de memória, é necessário um procedimento que resolva ausências de páginas ocorridas nos programas, que estão sendo executados nos outros módulos do sistema, por outros processadores, o que não pode ser resolvido com esses integrados dedicados.

. Possibilidade de uso de circuitos integrados que fazem o gerenciamento de memória

As principais funções desses integrados são a relocação e conversão de endereços e a proteção da memória.

Da maneira como é proposto o tratamento do endereço virtual neste trabalho, tem-se na TPV o endereço físico da página, sendo que uma vez obtido esse endereço, é desnecessária a sua relocação. Isso torna o acesso mais rápido, e sua desvantagem é apenas na inicialização do processo, pois se o espaço de memória real do processo a ser inicializado está ocupado, as páginas que estiverem nesse espaço devem ser transferidas para outra área de memória (principal ou secundária), como visto em

IV.5.

É importante observar que a desvantagem citada, ocasiona um maior gasto de tempo na inicialização, mas diminui o tempo gasto nos acessos à memória. Como o número de acessos feitos por um processo é muito maior que o número de inicializações, essa característica torna-se bastante vantajosa. Assim, como a estrutura proposta para o módulo de memória virtual não precisa de relocações de endereços, torna-se desnecessária a utilização desses circuitos integrados para esse fim.

Podem-se ainda citar as seguintes desvantagens na utilização desses integrados:

- o tempo para execução de um ciclo é bastante elevado. Como exemplo, para o MC 68451, segundo Stockton em {St82}, tem-se 625 ns para um ciclo de escrita,
- o número de páginas que esses integrados podem gerenciar é reduzido, sendo necessário o uso de chaveamento de páginas ou então o emprego de um número elevado de circuitos. O MC 68451, por exemplo, permite o gerenciamento de 32 páginas, sendo que cada chaveamento gasta, segundo {St82}, 2 ciclos de escrita.

Quanto ao problema da proteção da memória, a implementação proposta neste trabalho é, através do hardware, de uma forma bastante rápida.

Assim, de acordo com as observações apresentadas, o emprego desses circuitos integrados no módulo de memória virtual proposto neste trabalho, não é recomendado pois isto não traria melhorias no desempenho ou facilidades na implementação do módulo.

Algumas características do módulo de memória virtual descrito neste capítulo, foram resultado da simulação de uma proposta inicial para o módulo, que evoluiu devido às conclusões das simulações. No capítulo V, apresentam-se as simulações efetuadas, seus resultados e as modificações que provocaram. É apresentada também a simulação do módulo conforme descrito neste capítulo e alguns resultados obtidos nessa simulação.

CAPÍTULO V

SIMULAÇÃO DO MÓDULO

1. *CONSIDERAÇÕES INICIAIS*

Neste capítulo são apresentadas simulações do módulo de memória, que foram efetuadas com o objetivo de se avaliar o seu desempenho, sob diversos aspectos e, a partir dos resultados obtidos, proporem-se modificações no módulo de memória.

Na seção 2 tem-se a simulação do módulo da forma como foi proposto inicialmente. Essa simulação indicou a existência de alguns problemas com relação ao desempenho do módulo. Foram feitas determinadas modificações e uma nova simulação, que está apresentada na seção 3, tendo sido demonstrado que ainda existiam certos pontos, que poderiam ser melhorados. Novas modificações foram efetuadas, obtendo-se então o módulo descrito no capítulo IV.

Na simulação final, apresentada na seção 4, foram executados vários testes para determinar-se o desempenho do módulo de memória virtual proposto neste trabalho.

Para que os resultados obtidos na simulação sejam válidos, necessitam-se de dados reais, isto é, os acessos à memória utilizados na simulação devem ter características semelhantes, aos acessos que realmente ocorrem em uma memória.

Assim, os dados utilizados na simulação foram coletados de um sistema baseado no microprocessador 8085 (KMD-85), onde para cada acesso à memória, durante a execução de alguns programas reais, desenvolvidos anteriormente para aplicações distintas, foram coletados dados referentes a:

- tipo do acesso (escrita/leitura)
- endereço a que se quer ter acesso

O esquema básico utilizado na coleta dos dados é o da Figura V-1.

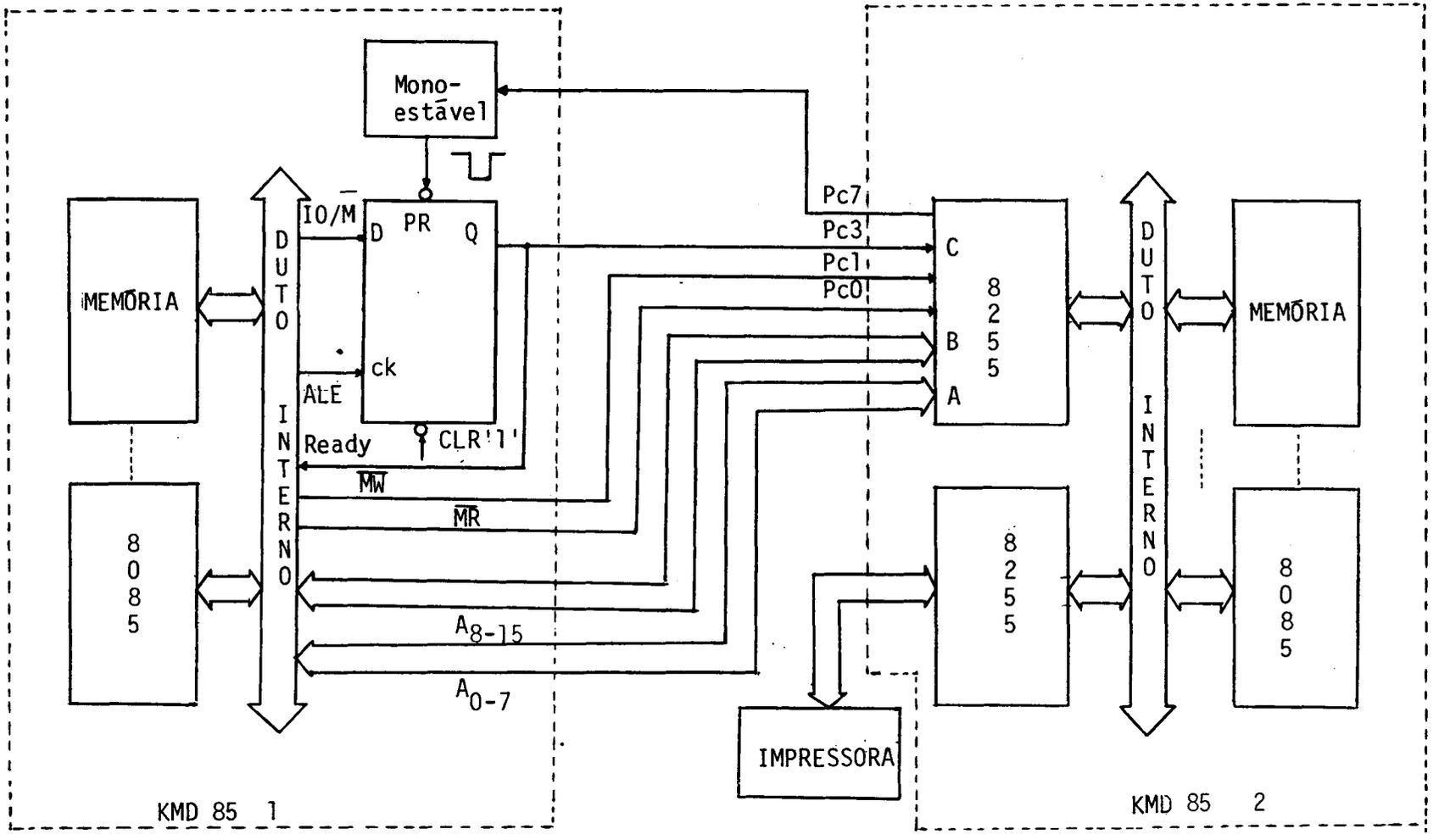


Figura V-1 - Esquema Utilizado para Coletar os Dados da Simulação

Nesse caso, tem-se um sistema KMD-85 executando o programa considerado e um segundo sistema KMD-85, monitorando a execução e coletando informações, toda a vez que o primeiro sistema faz um acesso à memória. Essa monitoração é efetuada por uma rotina desenvolvida para esse fim e que é apresentada no apêndice A.

Foi considerado na simulação um grande número de dados, obtidos a partir desses sistemas, incluindo acessos à memória diretamente ligada ao programa e também à memória ligada ao sistema monitor do equipamento, o que permite uma maior aproximação dos casos reais.

2. *SIMULAÇÃO INICIAL DO MÓDULO DE MEMÓRIA*

2.1. *Estrutura Modular da Simulação*

O módulo de memória originalmente proposto, apresentava algumas características diferentes do módulo descrito no capítulo IV. Entre elas, podem-se citar:

a. as páginas eram alocadas na memória principal, quando fossem endereçadas pela primeira vez,

b. a memória do SDF era dividida por processo e não existia a possibilidade de um processo utilizar a área reservada ao outro, mesmo que esse não estivesse inicializado,

c. o duto não era liberado em acessos que envolviam a memória secundária.

Considerando o módulo de memória original implementou-se uma simulação, cuja estrutura modular está representada na Figura V-2.

A simulação pode atender quatro tipos de comandos:

- Inicialização - que simula um comando para inicializar um processo. Dentro desse bloco, tem-se um procedimento, cuja função é determinar os dados necessários na inicialização,
- Acesso - simula os acessos ao módulo de memória, sendo subdividido em diversas partes que desempenham as funções dos diversos níveis de memória,
- Finalização - executado toda vez que se tem um comando para finalizar um processo,
- Impressão - bloco que não tem função de simular alguma parte do módulo de memória e sim de fornecer relatórios sobre as tabelas mantidas na simulação.

O comando mais complexo dessa estrutura é o que simula um acesso ao módulo de memória . Este comando é subdividido em:

- Pcache - verifica se o endereço a que se quer ter acesso está no cache. Este procedimento utiliza um outro, denominado tacesso, que verifica se o acesso requisitado é permitido.
- Memprin 1 - verifica se a página endereçada, tem seu endere

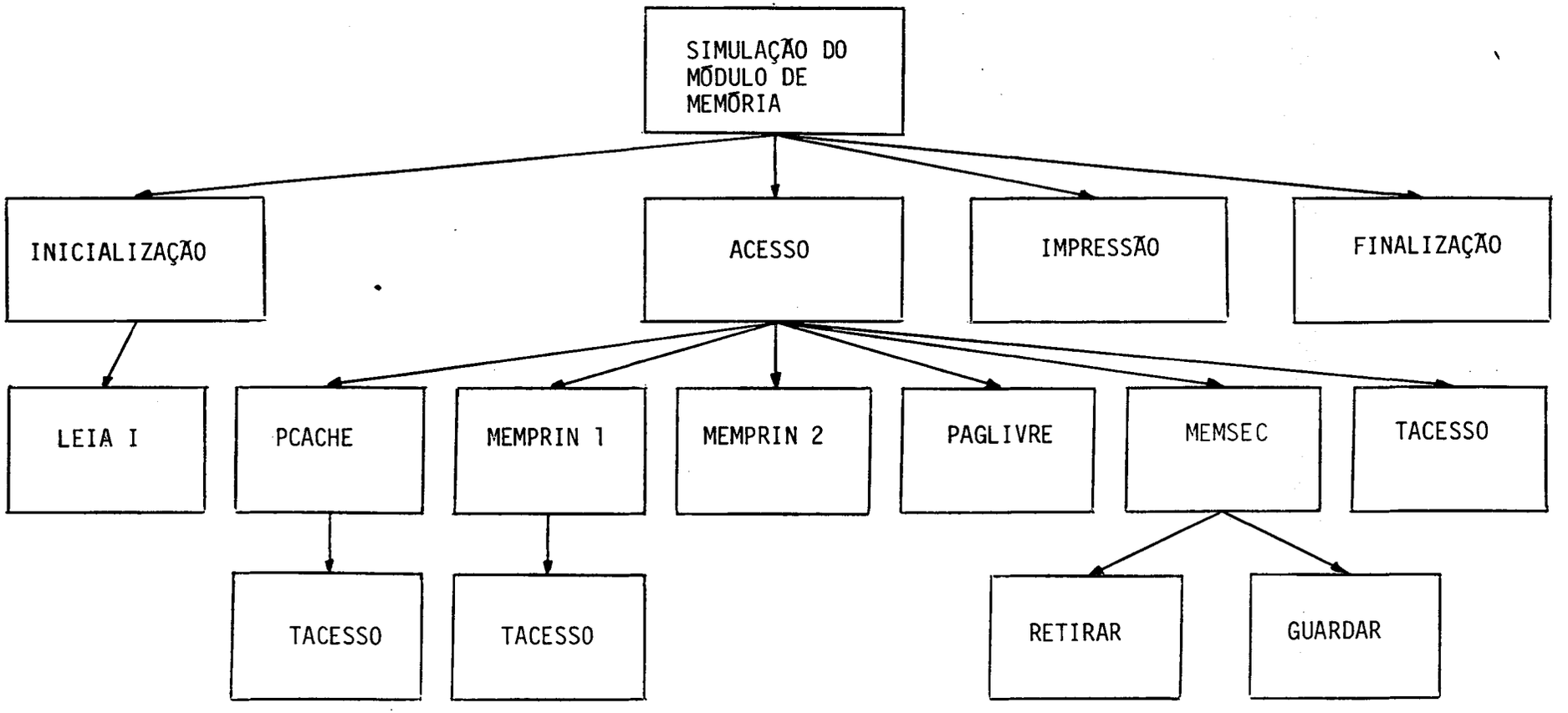


Figura V-2 - Estrutura Modular da Simulação Inicial

ço real em registrador. Esse procedimento também verifica se o tipo de acesso está correto, através do Tacesso.

- Memprin 2 - procedimento que simula acessos à endereços que estão na memória principal.
- Paglivre - procedimento executado quando uma página é endereçada pela primeira vez, e ainda existem páginas livres na memória principal para alocá-la.
- MEMSEC - simula acessos à memória secundária.
- Tacesso - verifica se o acesso pretendido é permitido.

Cada bloco descrito, corresponde a um procedimento do simulador, que foi escrito em linguagem Pascal para o computador PDP 11/45, sistema operacional RSX 11-M.

2.2. *Problemas Indicados pela Simulação*

A execução da simulação descrita no ítem anterior, utilizando dados obtidos conforme descrito na seção 1 deste capítulo, apresentou os resultados mostrados no apêndice B1. Analisando-se tais resultados, pode-se observar que o tempo gasto na alocação de páginas na memória principal, durante o primeiro acesso feito à página, causa um aumento demasiado no tempo total gasto nos acessos simulados.

Para o exemplo apresentado, tem-se uma taxa de acertos, para o cache, de aproximadamente 31% mesmo sendo uma taxa bastante razoável, para um cache de dimensões reduzidas como o sugerido, pode-se observar que a redução de tempo obtida pelo cache, fica pouco significativa perto do tempo gasto pelos outros acessos.

Observa-se ainda que quando existem acessos à memória secundária, o tempo de acesso torna-se bastante grande. Um problema que isso pode acarretar é a manutenção do duto conectado ao módulo de memória, para intervalos de tempo muito grandes, podendo acarretar uma saturação no duto.

Assim, pode-se concluir que os principais problemas apontados pela simulação são:

- a alocação de páginas na memória principal, quando a página for endereçada pela primeira vez, ocasiona um grande atraso no acesso, pois, torna-se necessária a execução de um software para determinação do endereço real da página,

- há necessidade de verificar se o duto não foi conectado ao módulo de memória por um tempo demasiadamente longo, podendo prejudicar o desempenho do módulo.

2.3. *Modificações Propostas*

2.3.1. *No Módulo de Memória*

A alocação de páginas na memória principal, deve ser feita de preferência durante a inicialização do processo. Pretende-se com isso diminuir o tempo gasto em acessos ao módulo de memória.

Com relação à inicialização, o acréscimo de tempo imposto a ela por mais essa tarefa não é significativo, uma vez que este procedimento já faz manipulações com as tabelas dos processos, através de software. Além disso, uma inicialização ocorre raramente, em relação a acessos à memória.

2.3.2. *Na Simulação*

Para verificar se o duto não está sendo bloqueado por um tempo demasiadamente grande, é necessário estimar-se o tempo transcorrido durante os acessos ao módulo de memória.

Com esse tempo e o tempo gasto nos acessos ao módulo de memória, pode-se determinar a porcentagem de tempo em que o duto está bloqueado.

3. *SEGUNDA SIMULAÇÃO DO MÓDULO DE MEMÓRIA*

3.1. *Estrutura Modular da Simulação*

Segundo a modificação proposta no item 2.3.1, deve-se nesta segunda simulação, eliminar a alocação de páginas durante os acessos ao módulo de memória. Essa função era desempenhada pelo procedimento denominado Paglivre, que era executado quando uma página era endereçada pela primeira vez. Assim, esse procedimento teve que ser retirado do comando de acesso e sua parte de alocação de páginas na memória principal, incorporada na rotina de inicialização. Obtém-se dessa forma a estrutura modular representada na Figura V.3.

Para a implementação da modificação proposta no ítem 2.3.3, deve-se determinar o tempo transcorrido. Para isso é necessário analisarem-se dois pontos:

a. porcentagem dos acessos à memória que o módulo que está utilizando o módulo de memória, processa localmente. É interessante que esse dado possa ser facilmente variado, para que se possa analisar como ele interfere no problema observado na simulação. Assim, esse dado deve estar entre as variáveis lidas pelo simulador.

b. tempo que o módulo gasta executando o seu programa, entre os acessos à memória. Como os acessos que um processador faz a memória não são consecutivos, isto é, ele necessita de um tempo para manipular os dados obtidos, antes de fazer novos acessos, este tempo deve ser computado no tempo real, juntamente com o tempo de acesso à memória.
   Para a obtenção do tempo transcorrido, fez-se uma estatística para verificar quantos estados e quantos acessos à memória, tem-se em cada instrução de diversos programas. A partir desses dados, obteve-se o número de estados por acesso à memória, para cada instrução desses programas. Verificou-se então a probabilidade de ocorrrer cada resultado encontrado.
   Assim, para cada acesso que o processador faz à memória, determina-se um número provável de estados, que transcorreram desde o último acesso, considerando as probabilidades estipuladas.

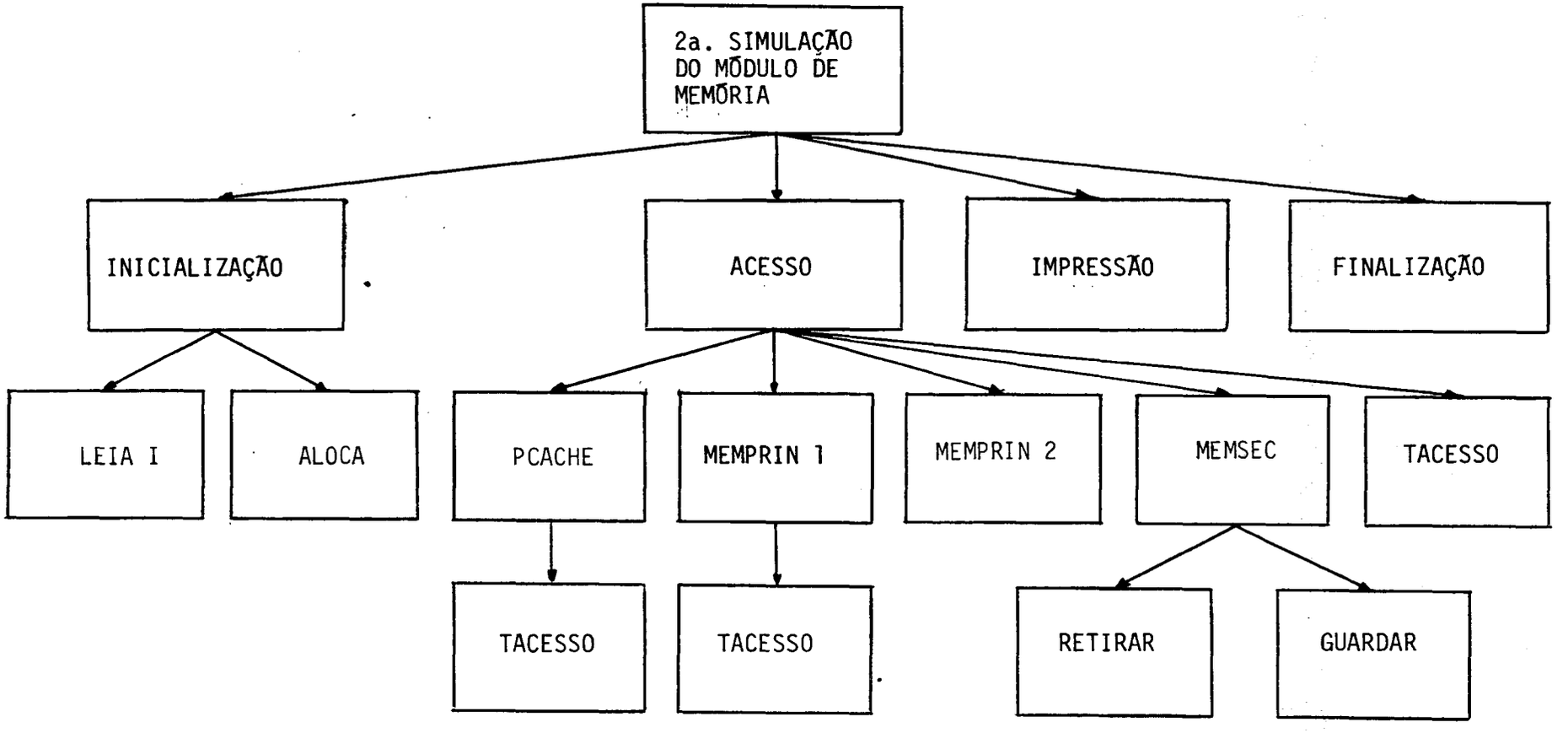


Figura V-3 - Estrutura Modular da Segunda Simulação

Considerando-se os pontos levantados nos itens a e b, pode-se determinar o tempo trasncorrido entre um acesso e outro, que um processo fez ao módulo de memória.

3.2. *Problemas Indicados pela Segunda Simulação*

Os resultados da execução desta nova simulação, apresentados no apêndice B2, permitem a observação dos seguintes pontos:

- Com a alocação das páginas na memória principal durante a inicialização do processo, o tempo gasto nos acessos ao módulo de memória diminui sensivelmente, principalmente, nos casos em que não existem acessos à memória secundária.
- A presença do cache diminui bastante o tempo gasto em acessos ao módulo de memória. Para os casos em que não se têm acessos à memória secundária, a redução do tempo gasto é da ordem de 31%.
- Quanto ao tempo que o duto permanece bloqueado por acessos ao módulo de memória, pode-se observar que:
  - Quando a memória secundária não é utilizada, a porcentagem de tempo que o duto fica bloqueado não deve levá-lo a uma saturação. Para os casos em que 50% dos acessos à memória são processados pelo módulo de memória e utilizando-se os tempos obtidos no capítulo IV, tem-se que o duto será bloqueado por esse processo durante aproximadamente 30% do tempo.
  - Quando a memória secundária é utilizada, a porcentagem de tempo em que o duto fica bloqueado, é bastante elevada podendo levá-lo à saturação, com apenas dois processos sendo executados.
- O tempo gasto em acessos à memória secundária é muito elevado, mostrando uma necessidade de minimização desses acessos e de uma melhor utilização da memória principal.

3.3. *Modificações Propostas*

Visando solucionar os problemas discutidos na seção 3.2, sugerem-se as seguintes modificações:

a. liberação do duto sempre que existirem acessos à memória se

cundária, para evitar que o duto fique bloqueado por muito tempo. As implicações da liberação do duto, já foram discutidas no capítulo IV.

b. permitir a utilização das áreas de memória de processos não inicializados. Dessa forma, tem-se um melhor aproveitamento da memória principal, e uma minimização dos acessos à memória secundária.

4. *ESTRUTURA MODULAR DA SIMULAÇÃO FINAL*

Com base nas modificações sugeridas no ítem 3.3. , obtém-se o módulo de memória descrito na capítulo IV e uma simulação final, cuja listagem é apresentada no apêndice B5.

Uma modifcação introduzida no módulo nesta simulação, foi a liberação do duto nos casos em que são necessários acessos à memória secundária. A maneira como essa liberação foi incluída no módulo de memória, foi apresentada em IV.4.

Para a implementação dessa característica no simulador, utilizaram-se três tempos:

. tduto - tempo em que o duto esteve bloqueado por acessos ao módulo de memória.

. tmm - tempo em que o módulo esteve ocupado no atendimento de acessos à memória do SDF.

. treal - tempo transcorrido durante a simulação.

Em tduto são considerados os tempos gastos para o atendimento de endereços que estão no cache ou na memória principal. Para acessos à memória secundária, é adicionado a esse tempo um tempo necessário para se colocar o acesso na fila de pedidos a serem atendidos e para liberar o duto.

Em tmm são considerados, além dos tempos de tduto, o tempo gasto nos acessos à memória secundária.

Treal é determinado considerando-se os fatores apresentados nos itens a e b da seção V.3.1.

Um algoritmo mostrando como é determinado cada um desses tempos e também como são considerados os tempos gastos em processamentos paralelos, é apresentado no apêndice B4.

A segunda modificação introduzida no módulo, foi decorrente da sugestão proposta no ítem b da seção V.3.3, que é a utilização do espaço de memória de processos não inicializados.

Para isso é necessário modificar-se a inicialização e a finalização dos processos. Na inicialização, quando o espaço destinado ao processo não é suficiente para se inicializar todas as páginas requisitadas, deve-se verificar a existência de processos não inicializados, e ocupar seus espaços de memória. É necessário ainda, verificar se o espaço destinado ao processo

que está sendo inicializado não está sendo ocupado por outro processo.

Na finalização deve-se liberar a área de memória reservada para o processo que está sendo finalizado, e outras que ele possa ter ocupado.

A estrutura modular da simulação final é a mesma apresentada na Figura V.3, para a segunda simulação, tendo-se modificações apenas dentro dos blocos da estrutura.

5. *CONSIDERAÇÕES FINAIS*

Como visto neste capítulo, a simulação do módulo de memória descrito no capítulo IV, permitiu a observação de algumas características não desejáveis, que levaram a modificações no módulo proposto inicialmente, chegando-se ao módulo descrito neste trabalho.

Dentre as alterações realizadas devido à simulação, cabe ressaltar:

- As páginas passaram a ser alocadas na memória principal, durante a inicialização do processo e não no primeiro acesso à página, como feito anteriormente,
- O duto passou a ser liberado nos acessos que dependem da memória secundária,
- Os espaços da memória principal, reservados a processos que não estão inicializados, passaram a ser utilizados por outros processos que deles necessitem.

A simulação mostra-se uma importante ferramenta, para a verificação da influência de algumas características, no desempenho global do sistema proposto. Dentre elas tem-se:

- Influência do tamanho da página no tempo gasto em acessos ao módulo de memória.

Através da simulação do módulo de memória para páginas de 256, 512 e 1024 palavras (simulações 1, 2 e 3 do apêndice A.3), obtiveram-se os resultados apresentados na tabela 1.

tamanho da página	tempo gasto para 2044 acessos (μs)
256	316.7
512	315.2
1024	310.4

Tabela 1

Alterando-se o tamanho da página de 256 para 1024 palavras, obtém-se, na simulação executada, uma diminuição inferior a 2%, no tempo gasto em acessos ao módulo de memória.

A escolha de página com 1024 palavras é justificada por facilitar os acessos à memória secundária e, principalmente, por diminuir consideravelmente o tamanho das tabelas para a manutenção do endereçamento virtual.

. Influência do cache no tempo gasto em acessos ao módulo de memória

Através das simulações 3,4,5 e 6 onde a execução foi efetuada com e sem cache e com e sem a utilização da memória secundária, obtiveram-se os tempos gastos para execução de 2044 acessos apresentados na tabela 2.

	com acessos a MS	sem acessos a MS
com cache	3.3968 s	310.4 µs
sem cache	3.3979 s	451.7 µs

Tabela 2

Para casos em que não existem acessos à memória secundária, tem-se que o cache reduz em 31% o tempo gasto em acessos ao módulo de memória., Para casos em que existem acessos à memória secundária, a influência do cache é bem menor, o que pode ser explicado pelo fato de que a economia de tempo ocasionada pelo cache, é de ordem de nano segundos, enquanto que um acesso a memória secundária é da ordem de mili segundos.

Através das simulações pode-se observar ainda, que o cache proposto neste trabalho, embora bastante simples e pequeno apresenta uma taxa de acerto acima de 50%.

. Influência de acesso à memória secundária no tempo gasto no módulo de memória e no tempo em que o duto fica bloqueado.

Analisando-se a tabela 2, pode-se verificar o aumento significativo do tempo gasto em acessos ao módulo de memória, nos casos em que se tem acesso à memória secundária.

Esses dados mostram a importância da liberação do duto, quando a memória secundária está envolvida nos acessos. Na tabela 3, tem-se a porcentagem de tempo em que o duto permanece bloqueado quando ele não é liberado (caso 1) e quando ele é liberado (caso 2), para diferentes tempos de acesso a memória secundária (simulações 5,13 e 14).

Simulação	tempo acesso	% tempo livre caso 1	% tempo livre caso 2	tempo gasto no módulo
5	96.53 ms	0.13	99.5	3.39 s
13	31.93 ms	0.41	98.5	1.14 s
14	76.53 ms	0.16	99.4	2.70 s

tabela 3

Pode-se observar através das colunas 3 e 4 que a liberação do duto em acessos a memória secundária é de grande importância.

. Influência da porcentagem de acessos executados na memória local de um módulo no,tempo em que o duto permanece livre, sem acessos ao módulo de memória

Para a análise desta característica, tem-se as simulações de 7 a 10, que fornecem os resultados apresentados na tabela 4.

Analisando-se a tabela 4, nota-se que a porcentagem de acessos à memória local dos módulos que utilizam o módulo de memória, tem uma grande influência sobre o número de processos que podem utilizar o duto sem levá-lo à saturação.

% processamento local	% tempo livre
0	87.8
20	89.6
40	92.4
60	95.1
80	97.6
90	98.8

tabela 4

. Influência do tempo de acesso à memória principal no desempenho do módulo de memória.

A diminuição do tempo de acesso à memória principal, resulta em um melhor desempenho do módulo de memória, mas também em um aumento no custo das memórias. A relação entre o tempo de acesso à memória principal e o tempo gasto para a execução de uma série de acessos ao módulo pode ser observado através das simulações 3, 15, 16, 17 e 18. Um resumo dos resultados obtidos é apresentado na tabela 5.

tempo de acesso a MP	tempo gasto em 2044 acessos
100 ns	287.8 µs
120 ns	310.4 µs
150 ns	344.4 µs
230 ns	434.9 µs
300 ns	514.0 µs

tabela 5

Como pode ser observado na tabela 5, o tempo de acesso à memória principal, varia linearmente com o tempo gasto no módulo de memória.

As modificações efetuadas na simulação, descrita neste capítulo, foram de fácil implementação, devido a estrutura modular empregada no desenvolvimento do sistema de simulação, que dá aos módulos características adequadas, tanto do ponto de vista de coesão (relacionamento intra-modular), como de acoplamento (relacionamento inter-modular).

No esquema modular empregado, cada tipo de acesso ao módulo de memória é tratado particularmente, por um conjunto coeso de procedimentos.

CAPÍTULO VI

CONCLUSÕES

1. *CONCLUSÕES FINAIS*

1.1. *Sobre o Estudo Bibliográfico*

Com o estudo desenvolvido sobre sistemas distribuídos, observa-se uma grande diversificação quanto a conceituação desses sistemas. Destaca-se, nesse estudo, a conceituação moderna de Fathi, em {Fa83}, para sistemas com microprocessadores distribuídos.

O estudo desenvolvido permitiu a formação de uma visão geral sobre os sistemas distribuídos e, com isso, a conceituação de um sistema distribuído funcional (SDF), como apresentado em {Sa85}. A partir dessa conceituação, foi possível a definição adequada do módulo de memória virtual, para o SDF.

Do estudo sobre memória destacam-se a conceituação e os aspectos de implementação de memória virtual e cache. Em especial, a abordagem abrangente sobre memória virtual apresentada em {De70} e as características a serem observadas para uma implementação eficiente de cache, apresentada em {Sm82}.

A familiarização com os conceitos de memória virtual e cache foi de grande importância, para tornar possível a definição do módulo de memória, com o desempenho desejado.

Destaca-se ainda na revisão bibliográfica, a apresentação de um estudo sobre circuitos integrados que suportam gerenciamento de memória. Esse estudo permitiu uma análise da possibilidade de utilização desses integrados, no módulo de memória virtual, através da comparação das características apresentadas por cada um, com as obtidas no módulo proposto neste trabalho.

1.2. *Sobre a Proposta Apresentada Para o Módulo de Memória Virtual*

A proposta apresentada neste trabalho para o módulo de memória virtual, satisfaz às diretrizes básicas estipula -

das inicialmente, no que se refere a:

- obtenção das características desejadas para o módulo de memória virtual,
- obtenção de um módulo com baixa complexidade,
- atendimento dos requisitos estipulados na proposta do sistema distribuído funcional (SDF), em {Sa85}.

Destaca-se no módulo proposto, a presença de um processador dedicado ao gerenciamento da memória do módulo, o que simplifica tanto o gerenciamento de memória de um modo geral, quanto à estrutura particular dos outros módulos do sistema. A presença do módulo de memória com processador dedicado, oferece ao sistema a possibilidade de endereçamento virtual, sem sobrecargas para o processador dos demais módulos.

No caso da não utilização de um processador dedicado, os problemas envolvidos no gerenciamento da memória, teriam que ser tratados pelo processador local, acarretando inevitáveis interrupções ao processamento normal, com a consequente necessidade de se salvar o estado atual do processador, antes do atendimento à interrupção e restaurá-lo posteriormente.

Além disso, também a manutenção das tabelas e estatísticas é facilitada pela presença do módulo de memória virtual, com seu processador dedicado, pois, essas tarefas (inevitáveis em sistemas com memória virtual) podem, como foi abordado neste trabalho, ser processadas em paralelo com acessos de outros módulos do sistema, ao módulo de memória virtual.

Outro ponto a destacar é a presença de um sistema de armazenamento secundário dedicado ao módulo de memória, evitando que o módulo de disco do sistema fique sobrecarregado e que o duto fique bloqueado, durante as trasferências de páginas entre as memórias principal e secundária.

A divisão da memória em três níveis: cache, memória principal e memória secundária, possibilita uma diminuição no tempo de acesso ao módulo. Os acessos ao cache e à memória principal, dependem apenas do hardware, sendo portanto, da ordem de nano segundos. A única dependência do microprocessador nesses tipos de acesso é para a atualização da tabela de páginas virtuais, em escritas na memória principal, que pode ser feita após a liberação do duto.

O cache proposto neste trabalho é bastante simples, compreendendo 16 linhas com 4 palavras cada uma, sendo que para cada processo é reservada uma linha.

As alterações das informações nele armazenadas, ocorrem apenas para os casos de leituras em posições de memória não presentes no cache ou nos casos de escritas em posições que estão presentes no cache. Nesse caso, a atualização da informação é efetuada tanto no cache como na memória principal.

A organização da memória principal do SDF em 4 blocos distintos, facilita a modificação do cache, sem comprometer o desempenho do módulo de memória nos acessos normais.

Uma característica a ser destacada nos acessos à memória principal é a presença de um registrador para cada processo, contendo o endereço real da última página lida. O tempo de acesso para os casos em que o endereço real é encontrado nesses registradores, é reduzido em 50%, em relação aos acessos em que o endereço real deve ser localizado na tabela de páginas virtuais.

Ressalta-se ainda que a relocação de páginas na memória principal é efetuada através de um algoritmo FIFO, levando-se em consideração o número de vezes, que as páginas foram retiradas da memória secundária.

A presença de um microprocessador dedicado ao módulo de memória virtual, permite a execução de uma série de tarefas em paralelo com novos acessos à memória e outros processamentos locais, nos demais módulos do sistema. Assim, tarefas como a manutenção de tabelas e estatísticas ou transferências antecipadas de páginas para a memória secundária, podem ser executadas pelo processador durante o seu tempo livre, reduzindo a influência dos acessos à memória secundária, no tempo total de processamento de cada módulo.

O mecanismo de proteção é implementado em hardware, na proposta apresentada, o que concorre para a obtenção de um sistema rápido e eficiente de controle dos acessos à memória.

1.3. *Sobre a Simulação do Módulo de Memória Virtual*

A simulação do módulo de memória virtual, implementada neste trabalho, destina-se tanto à depuração do módulo, como à verificação da influêcia de certas características no seu desempenho global.

Na depuração, obtiveram-se através da simulação, importantes características do módulo de memória, tais como:

- liberação do duto durante acessos à memória secundária,
- alocação de páginas na memória principal, durante a inicialização,
- utilização mais apropriada da memória principal, possibilitando o aproveitamento da área de memória reservada a processos não inicializados.

Na verificação do desempenho do módulo, observou-se através da simulação, a influência de algumas características, apresentadas em V.5. Entre elas citam-se:

- influência do cache,
- influência do tempo de acesso à memória principal,
- influência do tempo de acesso à memória secundária,
- influência do tamanho da página.

No desenvolvimento do sistema de simulação empregou-se uma estrutura modular, permitindo que futuras alterações no módulo de memória ou mesmo que o teste de novas características, possam ser facilmente executados, pois, no esquema modular empregado, cada tipo de acesso ao módulo é tratado por um conjunto coeso de procedimentos. Assim, para alterar-se, por exemplo, as características de um dos níveis de memória, basta alterar o procedimento correspondente, que simula o nível de armazenamento a ser modificado.

Dessa forma, o sistema de simulação apresentado, não deve ser tomado apenas como uma ferramenta de apoio, no desenvolvimento inicial do módulo de memória virtual, mas também como um importante meio para futuros testes, visando alterações e adaptações voltadas à expansão ou assimilação de avanços tecnológicos, entre outros.

1.4. *Sobre os Pontos Relevantes e Contribuições do Trabalho*

Como pontos relevantes e contribuições do trabalho, destacam-se:

a. a proposta do módulo de memória virtual para um sistema com as características propostas em {Sa85}.

b. a proposta de um cache simples e pequeno, sem necessidade de algoritmos de relocação, com fácil localização da posição de memória requerida e com taxa de acerto acima de 50%.

c. utilização de um processador dedicado ao gerenciamento de memória, permitindo simplificações consideráveis tanto a nível de gerenciamento de memória, quanto a nível de organização dos demais módulos do sistema.

d. a presença de uma memória principal, que pode endereçar palavras de 64 bits ou de 16 bits, facilitando a modificação do cache.

e. a implementação de um esquema de memória virtual onde toda a parte de acessos à memória principal é efetuada por hardware, reduzindo-se o tempo de acesso à memória, sem um grande aumento na complexidade do módulo.

f. o desenvolvimento de um sistema modular para a simulação do módulo de memória proposto, que permite adaptações simples para verificação de características sugeridas futuramente.

g. compilação de uma série de informações sobre memória virtual e cache.

2. *A CONTINUIDADE DO TRABALHO*

Os grupos envolvidos no desenvolvimento do sistema distribuído funcional, tem trabalhado em pesquisas voltadas à implementação final do sistema. Sendo assim, alguns módulos funcionais destinados ao SDF estão sendo estudados.

Este trabalho está inserido nesse contexto e, portanto, constitui uma proposta inicial de um dos módulos do SDF.

Em continuidade à proposta aqui apresentada sugerem-se:

- a implementação e avaliação de uma versão experimental, a ser acoplada a módulos com características compatíveis com as estipuladas para o SDF.
- a evolução da arquitetura proposta neste trabalho, em face aos testes executados na versão experimental, inicialmente através de simulações e posteriormente através de novas implementações físicas, de modo a se melhorar o desempenho global do módulo de memória virtual do SDF.

BIBLIOGRAFIA

{De70} - DENNING,J.P. - Virtual Memory. Computing Surveys, v.2, nº 3, Sep. 1970.

{Di79} - Digital VAX 11/780. Hardware Handbook. Digital's Sales Support Literature Group. 1979.

{Di80} - Digital VAX Handbook. Digital's New Products Marketing, 1980.

{Di82} - Digital VAX Technical Summary. Digital Equipment Corporation, 1982.

{Do76} - DORAN,R.W. - Virtual Memory. Computer, Oct. 1976.

{En78} - ENSLOW,P.H. - What is a "Distributed" Data Processing System?. Computer, Jan. 1978.

{Fa83} - FATHI,E.T. & KRIEGER,M. - Multiple Microprocessor Systems: What, Why and When. Computer, Mar. 1983.

{Ha83} - HAWK,R.L. - Cache Memory Increases Instruction Execution Speed for Superminis. Computer Technology Review, Sum. 1983.

{Ho72} - HOARE,C.A.R. & McKLANG,R.M. A Survey of Management Techniques In Hoare,C.A.R. and Perrott, R.H. (eds) - Operating Systems Techniques. Academic Press, 1972.

{Ho79} - HOUDEK,M.E. & MITCHELL,G.R. - Hash Index Helps Manage Large Virtual Memory. Eletronics, Mar (15). 1979.

{In82} - Intel Corporation - iAPX/10 High Performance Microprocessor with Memory Management and Protection. Advance Information, 1982.

{Le82} - LEVY,H.M. & LIPMAN,P.H. - Virtual Memory Management in the VAX/VMS Operating System. Computer, Mar. 1982.

{Ma83} - MacGREGOR,D. & MOTHERSOLE,D.S. - Virtual Memory and the MC 68010. IEEE Micro, Jun. 1983.

{Mo83} - Motorola Semiconductors 1 G bit Microprocessors Data Manual. Motorola Inc., 1983.

{Rh82a} - RHODES,C. - Caches Keep Main Memories from Slowing Down Fast CPUs. Eletronic Design, Jan. 1982.

{Rh82b} - RHODES,C.; CHUN,J. & HERNDON,T. - Cache Memory Functions Surface in VLSI Chip. Eletronic Design, Feb. 1982.

{Sa85} - SANTANA,M.J. - Descrição, Anâlise e Considerações Sobre a Implementação de Um Duto Paralelo de Comunicação de Alto Nîvel Para Um Sistema Distribuîdo Funcional , 1985.

{Sh74} - SHAW,A.C. - The Logical Design of Operating Systems. Prentice Hall Inc., 1974.

{Sm82} - SMITH,A.J. - Cache Memories. Computing Surveys, v.14, nº 3, Sep. 1982.

{So81} - SOLTIS,F.G. - Design of a Small Business Data Processing Systems. Computer, Sep. 1981.

{St81} - STEVENSON,D. - Virtual Memory on the Z 8003 in Laliotis, T. (ed) VLSI in the Laboratory the Office the Factory the Home. Compcon, Feb. 1981.

{St82} - STOCKTON,J.F. - The M68451 Memory Management Unit. Electronic Engineering, May. 1982.

{St84} - STRACK,J. - Sistemas de Processamento Distribuîdo. Livros Tēcnicos e Cientîficos Editora S.A., 1984.

{Ti85} - TITUS,C.A.; TITUS,J.A.; BALDWIN,A.; SCALON,L. & HUBIN, W.N. - Microprocessadores de 16 bits. Editora Campus Ltda, 1985.

{Va82} - VanAKen,J. - Match Cache Architecture to the Computer System. Eletronic Design, Mar. (4). 1982.

{Zo81} - ZOLNOWSKY,J. - Philosophy of the MC 68451 Memory Management Unit in Latiots,T. (ed) VLSI in the Laboratory the Office the Factory the Home. Compcon, Feb. 1981.

APÊNDICE

A. Rotina Utilizada para Coleta de Dados Reais Sobre Acessos à Memória, para Utilização na Simulação do Módulo de Memória Virtual.

```
                        ;
                        ;       ROTINA PARA COLETA DE DADOS
                                ORG 4000H

4000'  3E 93                    MVI     A,93H   ; programacao da 8255
4002'  D3 17                    OUT     17H     ;
4004'  01 01FF                  LXI     B,01FFH ; variavel para o loop
4007'  21 4100                  LXI     H,4100H ; HL- indica a partir de qual posicao de memoria,
                                                ;     os dados coletados sao armazenados
400A'  3E 80                    MVI     A,80H   ; linha de controle do monoestavel
400C'  D3 16                    OUT     16H
400E'  3E 00                    MVI     A,00    ; linhas de controle
4010'  D3 16                    OUT     16H
4012'  3E 80                    MVI     A,80H
4014'  D3 16                    OUT     16H
4016'  3E 90                    MVI     A,90H
4018'  D3 16                    OUT     16H


401A'                   LOOP1:                  ; verifica se existem acessos 'a memoria
401A'  DB 16                    IN      16H
401C'  E6 08                    ANI     08H
401E'  C2 401A'                 JNZ     LOOP1

                                                ; verifica se ja comecou o programa
4021'  DB 15                    IN      15H
4023'  E6 F0                    ANI     0F0H
4025'  FE 40                    CPI     40H
4027'  CA 4035'                 JZ      ESTE
                                                ; se nao comecou libera o micro e retorna para loop1
402A'  3E 01                    MVI     A,01
402C'  D3 16                    OUT     16H
402E'  3E 90                    MVI     A,90H
4030'  D3 16                    OUT     16H
4032'  C3 401A'                 JMP     LOOP1
4035'                   ESTE:                   ; se comecou, ler os dados e armazenar a partir
4035'  DB 15                    IN      15H     ; da posicao 4100H
4037'  77                       MOV     M,A
4038'  23                       INX     H
4039'  DB 14                    IN      14H
403B'  77                       MOV     M,A
403C'  23                       INX     H
403D'  DB 16                    IN      16H
403F'  E6 03                    ANI     03H
4041'  77                       MOV     M,A
4042'  23                       INX     H

                                                ; verifica fim do loop
4043'  0B                       DCX     B
4044'  78                       MOV     A,B
4045'  B1                       ORA     C
```

```
4046'  CA 405B'          JZ      FIM

                                 ; libera o microprocessador
4049'  3E 01             MVI     A,01
404B'  D3 16             OUT     16H
404D'  3E 90             MVI     A,90H
404F'  D3 16             OUT     16H
4051'                LOOP2:              ; verifica se existe novo acesso 'a memoria
4051'  DB 16             IN      16H
4053'  E6 08             ANI     08H
4055'  C2 4051'          JNZ     LOOP2
4058'  C3 4035'          JMP     ESTE
405B'                FIM:                ; volta ao estado inicial
405B'  3E 80             MVI     A,80H
405D'  D3 16             OUT     16H
405F'  3E 00             MVI     A,00
4061'  D3 16             OUT     16H
4063'  3E 80             MVI     A,80H
4065'  D3 16             OUT     16H
4067'  E7                RST     4
```

B. Simulação do Módulo de Memória Virtual

B.1. Resultados Obtidos com a Simulação Inicial

1.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  4
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 16
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .46830579E-03 SEGUNDOS
  .NUMERO DE ACESSOS AO CACHE -      1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   225
     TAXA DE ACERTO -    .11007827E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2041
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

2.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  4
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 16
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - FALSE
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .60955714E-03 SEGUNDOS
  .NUMERO DE ACESSOS AO CACHE -         0
     TAXA DE ACERTO -    .00000000E+00%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -  1816
     TAXA DE ACERTO -    .88845407E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   225
     TAXA DE ACERTO -    .11007827E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2041
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

3.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 16
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - FALSE .
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .34000846E+01 SEGUNDOS
  .NUMERO DE ACESSOS AO CACHE -         0
     TAXA DE ACERTO -    .00000000E+00%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -  1816
     TAXA DE ACERTO -    .88845407E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   199
     TAXA DE ACERTO -    .97358119E+01%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2015
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   9
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -  26
```

4.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 16
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .34000268E+01 SEGUNDOS
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   199
     TAXA DE ACERTO -    .97358119E+01%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2015
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   9
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -  26
```

B.2. Resultados Obtidos com a Segunda Simulação

1.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  4
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 16
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .31044205E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .52369197E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
    -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .94072091E+02%
    -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .94072091E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -    .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -     0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

2.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  4
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 16
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - FALSE
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .45167064E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .49578772E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
    -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .90889883E+02%
    -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .90889883E+02%
  .NUMERO DE ACESSOS AO CACHE -        0
     TAXA DE ACERTO -    .00000000E+00%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -  1816
     TAXA DE ACERTO -    .88845407E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -    .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -     0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

3.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 16
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - FALSE
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .34938302E+01 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .34978663E+01 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
    -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .11539000E+00%
    -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .99505835E+02%
  .NUMERO DE ACESSOS AO CACHE -         0
     TAXA DE ACERTO -    .00000000E+00%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -  1816
     TAXA DE ACERTO -    .88845407E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   200
     TAXA DE ACERTO -    .97847354E+01%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2016
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -    28
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   9
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -  27
```

4.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 16
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .34932899E+01 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .34981486E+01 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
    -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .13889439E+00%
    -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .99519681E+02%
  .NUMERO DE ACESSOS AO CACHE -      1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   200
     TAXA DE ACERTO -    .97847354E+01%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2016
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -    28
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   9
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -  27
```

B.3. Resultados Obtidos com a Simulação Final

1.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 256 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  8
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO - 100
  .NUMERO MAXIMO DE PROCESSOS -  5
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .31671109E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .50593801E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .93740171E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .93740171E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   640
     TAXA DE ACERTO -    .31311154E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   263
     TAXA DE ACERTO -    .12866929E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

2.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 512 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  4
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .31527816E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .49552927E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .93637597E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .93637597E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   648
     TAXA DE ACERTO -    .31702545E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   255
     TAXA DE ACERTO -    .12475538E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

3.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -   .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -   .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -   .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -   .31044205E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -   .49292526E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -   .93702089E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -   .93702089E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -   .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -   .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -   .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

4.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - FALSE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -   .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -   .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -   .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -   .45167064E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -   .47079200E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -   .90406203E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -   .90406203E+02%
  .NUMERO DE ACESSOS AO CACHE -        0
     TAXA DE ACERTO -   .00000000E+00%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -  1816
     TAXA DE ACERTO -   .88845407E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -   .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

5.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -  108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .33967956E+01 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -  .34014821E+01 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .13777393E+00%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .99503540E+02%
  .NUMERO DE ACESSOS AO CACHE -  1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -  675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -  200
     TAXA DE ACERTO -    .97847354E+01%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2016
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -  28
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -  8
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -  27
```

6.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - FALSE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -  108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .33970636E+01 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .34014520E+01 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .12902051E+00%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .99497342E+02%
  .NUMERO DE ACESSOS AO CACHE -  0
     TAXA DE ACERTO -    .00000000E+00%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -  1816
     TAXA DE ACERTO -    .88845407E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -  200
     TAXA DE ACERTO -    .97847354E+01%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2016
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -  28
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -  8
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -  27
```

7.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -   0%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -   .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -   .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -   .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -   .31044205E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -   .25517923E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -   .87834435E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -   .87834435E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -   .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -   .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -   .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

8.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  20%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -   .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -   .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -   .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -   .31044205E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -   .29981039E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -   .89645457E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -   .89645457E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -   .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -   .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -   .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

9.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  40%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .31044205E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .41088099E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .92444533E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .92444533E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -    .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

10.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  60%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .31044205E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .62758382E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .95053410E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .95053410E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -    .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

11.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  80%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  1132
  .NUMERO DE ACESSOS PARA LEITURA -  1082
  .NUMERO DE ACESSOS PARA ESCRITA -    50
  .TEMPO GASTO NO MODULO DE MEMORIA -    .17144137E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .71992393E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .97618639E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .97618639E+02%
  .NUMERO DE ACESSOS AO CACHE -      618
     TAXA DE ACERTO -    .54593640E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   400
     TAXA DE ACERTO -    .35335689E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   114
     TAXA DE ACERTO -    .10070671E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  1132
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

12.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  90%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  1132
  .NUMERO DE ACESSOS PARA LEITURA -  1082
  .NUMERO DE ACESSOS PARA ESCRITA -    50
  .TEMPO GASTO NO MODULO DE MEMORIA -    .17144137E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .13876981E-01 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .98764574E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .98764574E+02%
  .NUMERO DE ACESSOS AO CACHE -      618
     TAXA DE ACERTO -    .54593640E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   400
     TAXA DE ACERTO -    .35335689E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   114
     TAXA DE ACERTO -    .10070671E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  1132
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

13.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .30330001E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .96700000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .11374242E+01 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .11421116E+01 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .41041770E+00%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .98525214E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   200
     TAXA DE ACERTO -    .97847354E+01%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2016
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -  28
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   8
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -  27
```

14.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .73330008E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .12062501E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .26979559E+01 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .27023699E+01 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .16334113E+00%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .99371469E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   200
     TAXA DE ACERTO -    .97847354E+01%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2016
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -  28
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   8
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -  27
```

15.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .10062499E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .28782388E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .48709819E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .94091093E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .94091093E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -    .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

16.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .15125000E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .34437584E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .47879589E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .92807507E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .92807507E+02%
  .NUMERO DE ACESSOS AO CACHE -     1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -    .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

17.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .23125000E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .43485515E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .50768717E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .91434633E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .91434633E+02%
  .NUMERO DE ACESSOS AO CACHE -      1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -    .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

18.

```
SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL

 CARACTERISTICAS DO MODULO:
  .TAMANHO DAS PAGINAS - 1024 PALAVRAS
  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL POR PROCESSO -  2
  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA POR PROCESSO -  50
  .NUMERO MAXIMO DE PROCESSOS - 10
  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS POR PROCESSO - TRUE
  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS LOCALMENTE -  50%
  .TEMPO DE ACESSO A MEMORIA SECUNDARIA -    .93300007E-01 SEGUNDOS
  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA -    .50000000E+07   BITS/SEG.
  .TEMPO DE ACESSO A MEMORIA PRINCIPAL -    .30250001E-06 SEGUNDOS

 RESULTADOS OBTIDOS COM A SIMULACAO
  .NUMERO DE ACESSOS AO MODULO DE MEMORIA -  2044
  .NUMERO DE ACESSOS PARA LEITURA -  1936
  .NUMERO DE ACESSOS PARA ESCRITA -   108
  .TEMPO GASTO NO MODULO DE MEMORIA -    .51403009E-03 SEGUNDOS
  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO -    .50739874E-02 SEGUNDOS
     PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE
      -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .89869397E+02%
      -COM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA -    .89869397E+02%
  .NUMERO DE ACESSOS AO CACHE -      1141
     TAXA DE ACERTO -    .55821919E+02%
  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA -   675
     TAXA DE ACERTO -    .33023482E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL COM O ENDERECO REAL NA TPV -   228
     TAXA DE ACERTO -    .11154599E+02%
  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE -  2044
  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA -   0
  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA -   0
```

B.4 - Algoritmo Utilizado para a Atualização dos Tempos na Simulação de um Acesso ao Módulo de Memória Virtual.

. Variáveis utilizadas

Local - número de acessos à memória local, efetuados por um módulo do SDF, entre dois acessos consecutivos ao módulo de memória virtual.

Tmm (ST) - tempo gasto no módulo de memória.

Tduto - tempo em que o duto esteve bloqueado com acessos ao módulo de memória.

Treal - tempo transcorrido durante os acessos ao módulo de memória.

ΔT - tempo em que o processador permaneceu desocupado.

Tanterior - tempo em que as tarefas relativas ao acesso anterior, foram concluídas.

TMP1 - tempo gasto em um acesso, quando o endereço real da página a que se quer ter acesso está em um registrador (excluindo-se o tempo de acesso à Memória Principal).

TMP2 - tempo gasto em um acesso, quando o endereço real da página a que se quer ter acesso, está na memória local (excluindo-se o tempo de acesso à memória principal).

TcacheE - tempo de escrita no cache.

TcacheL - tempo de leitura no cache.

Tlibduto - tempo gasto na liberação do duto.

Ttrazer = tguardar - tempo gasto para localizar uma página no disco.

Ttrans - tempo gasto para transferir uma página entre as memórias principal e secundária.

Tsoft - tempo gasto com o software em um acesso à memória secundária.

Thard - tempo gasto com o hardware em um acesso à

memória secundária.

Tsoftp - tempo gasto com o software paralelo em um acesso à memória secundária.

Tamp - tempo de acesso à memória principal.

Temp - tempo gasto na atualização da TPV em uma escrita na memória principal.

fc - frequência do relógio.

Tampag - tamanho da página.

. Algoritmo

```
Para i = 1 até Local faça
  Treal = Treal + estados/fc
Se o endereço está no cache
Então
  Se w̄ = 1
  Então
    Tduto = Tduto + TcacheL
    Tmm = Tmm + TcacheL
  Senão
    Tduto = Tduto + TcacheE
    Tmm = Tmm + TcacheE
    Tanterior = Tanterior + Temp
Senão
  Se o endereço está na última página lida
  Então
    Tduto = Tduto + Tmp1 + Tamp
    Tmm = Tmm + tmp1 + Tamp
    Se w̄ = 0
    Então
      Tanterior = Tanterior + Temp
  Senão
    Se o endereço está na memória principal
    Então
      Tmm = Tmm + Tmp2 + 2*Tamp
      Tduto = Tduto + Tmp2 + 2*Tamp
      Se w̄ = 0
      Então
```

```
    Tanterior = Tanterior + Temp
Senão (endereço está na memória secundária)
  Tmm = Tmm + Thard + 2*Tamp
  Tduto = Tduto + Thard + 2*Tamp
  ΔT = Treal - Tanterior
  Se ΔT < 0
  Então
    Tmm = Tmm - ΔT
    Tduto = Tduto - ΔT
    Treal = Treal - ΔT
  Se o bit de modificação da página a ser guardada for i-
     gual a 1
  Então
    Tmm = Tmm + Tguardar + Tampag *16/Ttrans
    Treal = Treal + Tguardar + Tampag *16/Ttrans
    Tduto = Tduto + Tlibduto
  Se a página a que se quer ter acesso já foi endereçada
  Então
    Tmm = Tmm + Ttrazer + Tampag*16/Ttrans
    Treal = Treal + Ttrazer + Tampag*16/Ttrans
    Se o duto ainda não foi liberado
    Então
      Tduto = Tduto + Tlibduto
  Tmm = Tmm + Tsoft
  Tduto = Tduto + Tsoft
  Treal = Treal + Tsoft
  Tanterior = Treal + Tsoftp
```

B.5. Listagem da Simulação Final

```
PROGRAM MEMORIA(INPUT,OUTPUT);
   (* SIMULACAO DE UM MODULO DE MEMORIA QUE SUPORTA ENDERECAMENTO
      VIRTUAL E IMPLEMENTACAO DE CACHES POR PROCESSO
      ROTINAS UTILIZADAS:
      INICIALIZA  - PROCEDIMENTO PARA FAZER AS INICIALIZACOES
                    NECESSARIAS PARA INICIO DE UM PROCESSO
        LEIAI     - PROCEDIMENTO PARA LEITURA DOS DADOS NECESSARIOS
                    NA NICIALIZACAO
        ININP     - PROCEDIMENTO PARA INICIALIZAR TABELAS NÁ ALOCACAO
                    DE PAGINAS NA MEMORIA PRINCIPAL
      ACESSO      - PROCEDIMENTO QUE EXECUTA ACESSOS A MEMORIA DO SDF
        TCESSO    - FUNCAO QUE VERIFICA SE UM ACESSO E' PERMITIDO
        PCACHE    - FUNCAO QUE VERIFICA SE O ENDERECO REQUERIDO ESTA
                    NO CACHE
        MEMPRIN1  - FUNCAO QUE VERIFICA SE O ENDERECO A QUE SE PRETENDE
                    TER ACESSO, PERTENCE A ULTIMA PAGINA A QUE SE TEVE
                    ACESSO PARA LEITURA
        MEMPRIN2  - PROCEDIMENTO PARA OS CASOS EM QUE O ENDERECO
                    REQUERIDO ESTA NA MEMORIA PRINCIPAL
        MENSEC    - PROCEDIMENTO PARA OS CASOS EM QUE A PAGINA NAO ESTA
                    NA MEMORIA PRINCIPAL
          RETIRAR - FUNCAO PARA DETERMINAR QUAL PAGINA DA MEMORIA
                    PRINCIPAL DEVE SER RETIRADA
          GUARDAR - PROCEDIMENTO PARA GUARDAR UMA PAGINA NA MEMORIA
                    SECUNDARIA
      IMPRESSAO    - PROCEDIMENTO PARA IMPRIMIR TABELAS E FILAS
      FINALIZA    - PROCEDIMENTO PARA FINALIZAR UM PROCESSO *)
   CONST
     MAXMS=50;  (*  NUMERO MAXIMO DE PAGINAS NA MEMORIA VIRTUAL  *)
     MAXMP=2;   (*  NUMERO MAXIMO DE PAGINAS NA MEMORIA PRINCIPAL  *)
     NPROC=10;  (*  NUMERO DE PROCESSOS QUE O SISTEMA COMPORTA  *)
     TAMPAG=1024;(*  TAMANHO DAS PAGINAS  *)
     MAXMPT=20;  (*  NUMERO TOTAL DE PAGINAS NA MEMORIA PRINCIPAL  *)
   TYPE
     TMSB=ARRAY[1..NPROC,1..MAXMS]OF BOOLEAN;
     TMSI=ARRAY[1..NPROC,1..MAXMS]OF INTEGER;
     TN=ARRAY[1..NPROC]OF INTEGER;
     TNR=ARRAY[1..NPROC] OF REAL;
     TABPROC=RECORD
               INIC:ARRAY[1..NPROC] OF BOOLEAN;
               PROC:ARRAY[1..NPROC] OF INTEGER;
             END;
     TP=RECORD
          VAL:TMSB;
          MODI:TMSB;
          PROTL:TMSB;
          PROTE:TMSB;
          ENDER:TMSI;
        END;
     TPR=RECORD
           VAZIO:ARRAY[1..NPROC,1..MAXMPT] OF BOOLEAN;
           PAGINA:ARRAY[1..NPROC,1..MAXMPT] OF INTEGER;
         END;
     FILLA=RECORD
             PAGVIRTUAL:INTEGER;
             PONT:^FILLA;
           END;
     PONTEIRO=ARRAY[1..NPROC,1..4]OF ^ FILLA;
     PONTFILA=RECORD
                COM:PONTEIRO;
                FIM:PONTEIRO;
                NOVOEND:^FILLA;
              END;
     ENDERECO=RECORD
                MEM:0..1;
                PROC:1..NPROC;
                ENDER:INTEGER;
              END;
     PROBABILIDADE=ARRAY[0..100] OF INTEGER;
```

```
70      VAR
71        TABPRO:TABPROC;  (*  TABELA QUE FORNECE INFORMACOES SOBRE A AREA DE
72                             MEMORIA RESERVADA A CADA PROCESSO
73          INIC  --> INDICA SE O PROCESSO FOI INICIALIZADO
74          PROC  --> INDICA SE A AREA DE MEMORIA ESTA SENDO OCUPADA E EM
75                    CASO AFIRMATIVO, QUAL PROCESSO ESTA OCUPANDO A AREA *)
76        ETP:TP;   (*  REPRESENTA A TABELA DE PAGINAS VIRTUAIS
77          VAL   --> INDICA A PRESENCA OU NAO DA PAGINA VIRTUAL NA
78                    MEMORIA PRINCIPAL
79          MODI  --> INDICA SE A PAGINA FOI MODIFICADA OU NAO
80          PROTL --> INDICA A PROTECAO DA PAGINA EM RELACAO A LEITURA
81          PROTE --> INDICA A PROTECAO DA PAGINA EM RELACAO A ESCRITA
82          ENDER --> INDICA O ENDERECO DA PAGINA VIRTUAL NA MEMORIA
83                    PRINCIPAL ( SE VAL = TRUE )  *)
84        ETPR:TPR;   (*  REPRESENTA A TABELA DE PAGINAS DA MEMORIA REAL
85          VAZIO  --> INDICA SE A PAGINA ESTA OCUPADA
86          PAGINA --> INDICA A PROXIMA PAGINA VAZIA(SE VAZIO = TRUE)
87                     OU QUAL PAGINA VIRTUAL ESTA OCUPANDO ESTA
88                     PAGINA REAL (SE VAZIO = FALSE) *)
89        ENDE:ENDERECO;   (*  ENDERECO A QUE SE TERA' ACESSO NA MEMORIA
90                             PRINCIPAL
91          MEM   --> INDICA SE O ACESSO E' NA MEMORIA LOCAL DO MODULO
92                    (MEM = 0) OU SE E' NA MEMORIA DO SDF (MEM = 1)
93          PROC  --> INDICA O PROCESSO ONDE SE ESTA TENDO UM ACESSO
94          ENDER --> INDICA O ENDERECO ONDE SE ESTA TENDO UM ACESSO  *)
95        PF:PONTFILA;   (*  CONTEM PONTEIROS PARA AS FILAS QUE SERAO
96                         UTILIZADAS NA RETIRADA DE PAGINAS DA MEMORIA PRINCIPAL
97          COM     --> PONTEIROS PARA  OS COMECOS DAS FILAS
98          FIM     --> PONTEIROS PARA OS FINAIS DAS FILAS
99          NOVOEND --> PONTEIRO AUXILIAR NA INSERCAO DE ELEMENTOS NA FILA*)
100       COMANDO:CHAR;  (*  INDICA QUAL COMANDO SERA EXECUTADO  *)
101       MAXPV,LOCAL,N,ENDCACHE,LIMITE,LPV,ENDMP1:TN;
102         (* MAXPV[I]     -->  INDICA A PAGINA VIRTUAL DE MAIOR VALOR
103                               INICIALIZADA PARA O PROCESSO I
104            LOCAL[I]    -->  PORCENTAGEM DE PROCESSAMENTO NO MODULO I
105            N[I]        -->  INDICA QUANTAS PAGINAS DO PROCESSO I JA'
106                              FORAM INICIALIZADAS
107            ENDCACHE[I] -->  INDICA O ENDERECO BASICO DOS DADOS QUE ESTAO
108                              NO CACHE DO PROCESSO I
109            LIMITE[I]   -->  INDICA QUANTAS PAGINAS DO PROCESSO I ESTAO
110                              NA MEMORIA PRINCIPAL
111            LPV[I]      -->  PONTEIRO PARA A LISTA DE PAGINAS VAZIAS
112                              DO PROCESSO I
113            ENDMP1[I]   -->  INDICA O ENDERECO DA ULTIMA PAGINA A QUE SE
114                              TEVE ACESSO PARA LEITURA NO PROCESSO I *)
115       FC:TNR;  (*  FC[I]  -->INDICA A FREQUENCIA DO CLOCK DO
116                                PROCESSADOR I  *)
117       PROB:PROBABILIDADE;  (*  VARIAVEL UTILIZADA NA DETERMINACAO DO
118                                  NUMERO DE ESTADOS DA INSTRUCAO  *)
119       EXCACHE:BOOLEAN;  (*  EXISTENCIA DO CACHE  *)
120       NP,I,J,AUXPL,AUXPE:INTEGER;
121         (* NP          --> NUMERO DE PAGINAS EM UM COMANDO DE
122                             INICIALIZACAO
123            I,J         --> VARIAVEIS AUXILIARES EM LOOPS
124            AUXPE,AUXPL-->  VARIAVEIS AUXILIARES NA LEITURA DA PROTECAO
125                            DAS PAGINAS *)
126       NMS,NCACHE,NMP1,NMP2,NGUARDAR,NTRAZER,NACESSO,LEITURA,ESCRITA:
127         INTEGER;
128         (* NMS      --> NUMERO DE ACESSOS A MEMORIA SECUNDARIA
129            NCACHE   -->NUMERO DE ACESSO AO CACHE
130            NMP1     -->NUMERO DE ACESSO A ULTIMA PAGINA A QUE SE
131                         TEVE ACESSO PARA LEITURA
132            NMP2     -->NUMERO DE ACESSO A MEMORIA PRINCIPAL
133            NGUARDAR-->NUMERO DE PAGINAS GUARDADA NA MEMORIA SECUNDARIA,
134                       ANTES DA LIBERACAO DO DADO
135            NTRAZER -->NUMERO DE PAGINAS TRAZIDA DA MEMORIA SECUNDARIA
136            NACESSO -->NUMERO DE ACESSOS AO MODULO DE MEMORIA
137            LEITURA -->NUMERO DE ACESSOS PARA LEITURA
138            ESCRITA -->NUMERO DE ACESSOS PARA ESCRITA  *),
```

```
139       RESTO,TTRAZER,TGUARDAR,TTRANS,ST,TDUTO,TREAL,TANTERIOR,TAMP:REAL;
140         (* RESTO     --> FRACAO DE INSTRUCAO QUE AINDA NAO FOI EXECUTADA
141            TTRAZER   --> TEMPO GASTO PARA TRAZER UMA PAGINA DA MEMORIA
142                          SECUNDARIA
143            TGUARDAR --> TEMPO GASTO PARA GUARDAR UMA PAGINA NA MEMORIA
144                          SECUNDARIA
145            TTRANS    --> TAXA DE TRANSFERENCIA ENTRE MEMORIA PRINCIPAL E
146                          MEMORIA SECUNDARIA ( EM MEGA BITS POR SEGUNDO )
147            ST        --> SOMATORIA DOS TEMPOS GASTOS NOS ACESSOS A MEMORIA
148            TDUTO     --> TEMPO QUE O DUTO FICOU BLOQUEADO COM ACESSOS AO
149                          MODULO DE MEMORIA
150            TREAL     --> INDICA O TEMPO REAL
151            TANTERIOR--> VALOR DE ST NO ACESSO ANTERIOR A MEMORIA SECUN -
152                          DARIA
153            TAMP      --> TEMPO DE ACESSO A MEMORIA PRINCIPAL  *)
154       ESTAT:TMSI;
155         (* ESTAT  -->  VARIAVEL QUE MANTEM UMA ESTATISTICA DA RETIRADA
156                        DE PAGINAS DA MEMORIA SECUNDARIA  *)
157     PROCEDURE INICIALIZA (VAR ETP:TP;VAR ETPR:TPR;VAR ENDE:ENDERECO;
158                     VAR NP:INTEGER;VAR ESTAT:TMSI;VAR PF:PONTFILA;
159                     VAR N,LIMITE,LPV,MAXPV,LOCAL:TN;VAR FC:TNR;
160                     VAR TABPRO:TABPROC);
161       TYPE
162         INICIAL=RECORD
163                   EI:INTEGER;
164                   EF:INTEGER;
165                   PL:BOOLEAN;
166                   PE:BOOLEAN;
167                 END;
168         ANTE=^FILLA;
169       VAR
170         X,ENDREAL,J,I,AUX,AUX2,NUMPAG,PVI,PVF,PV,PL,PVFIM:INTEGER;
171           (* X,J,I,AUX2--> VARIAVEIS AUXILIAR PARA LOOPS
172              ENDREAL    --> ENDERECO REAL
173              AUX        --> VARIAVEL QUE INDICA QUAL PROCESSO ESTA NAO
174                             INICIALIZADO, PODENDO CEDER SUA AREA DE
175                             MEMORIA
176              NUMPAG     --> NUMERO DE PAGINAS QUE AINDA DEVEM SER
177                             INICIALIZADAS
178              PVI        --> NUMERO DA  PAGINA INICIAL A SER INICIALIZADA
179              PVF        --> NUMERO DA PAGINA FINAL A SER INICIALIZADA
180              PV         --> INDICA A PRIMEIRA PAGINA DAS QUE AINDA DEVEM
181                             SER INICIALIZADAS
182              PL         --> INDICA O NUMERO DE PAGINAS DE UM PROCESSO QUE
183                             ESTAO LIVRES
184              PVFIM      --> INDICA A ULTIMA PAGINA A SER INICIALIZADA NA
185                             AREA CONSIDERADA *)
186         INI:INICIAL;    (*  DADOS PARA A INICIALIZACAO
187           EI --> ENDERECO INICIAL DO INTERVALO CONSIDERADO
188           EF --> ENDERECO FINAL DO INTERVALO CONSIDERADO
189           PL --> BIT DE PROTECAO CONTRA LEITURA
190           PE --> BIT DE PROTECAO CONTRA ESCRITA  *)
191         ANT:ANTE;  (* VARIAVEL AUXILIAR NA INICIALIZACAO DAS FILAS  *)
192       PROCEDURE LEIAI(VAR ENDE:ENDERECO;VAR INI:INICIAL);
193         VAR
194           J:INTEGER;    (*  VARIAVEL AUXILIAR PARA LOOPS  *)
195         BEGIN
196           READ(ENDE.PROC);
197           READLN(INI.EI,INI.EF,AUXPL,AUXPE);
198           IF AUXPL=0
199           THEN INI.PL:=FALSE
200           ELSE INI.PL:=TRUE;
201           IF AUXPE=0
202           THEN INI.PE:=FALSE
203           ELSE INI.PE:=TRUE;
204         END;
205       PROCEDURE ININP(VAR ENDE:ENDERECO;VAR LPV:TN;VAR PF:PONTFILA;
206                  VAR ETPR:TPR;VAR ETP:TP;PAGVIRT,AUX1,AUX:INTEGER);
207         VAR
```

```
208           ENDREAL:INTEGER;
209             (* ENDREAL--> REPRESENTA O ENDERECO REAL  *)
210         BEGIN
211           ENDREAL:=(AUX1-1)*MAXMP*TAMPAG+(AUX-1)*TAMPAG;
212           (*  ATUALIZACAO DA ETPR  *)
213           LPV[AUX1]:=ETPR.PAGINA[AUX1,AUX];
214           ETPR.PAGINA[AUX1,AUX]:=PAGVIRT;
215           ETPR.VAZIO[AUX1,AUX]:=FALSE;
216           (*  ATUALIZACAO DA ETP *)
217           ETP.VAL[ENDE,PROC,PAGVIRT]:=TRUE;
218           ETP.ENDER[ENDE,PROC,PAGVIRT]:=ENDREAL;
219             ETP.MODI[ENDE,PROC,PAGVIRT]:=FALSE;
220           (*   ATUALIZACAO DA FILA  *)
221           NEW(PF.NOVOEND);
222           PF.NOVOEND^.PAGVIRTUAL:=PAGVIRT;
223           PF.NOVOEND^.PONT:=NIL;
224           IF PF.COM[ENDE,PROC,1]=NIL
225           THEN
226             PF.COM[ENDE,PROC,1]:=PF.NOVOEND
227           ELSE
228             PF.FIM[ENDE,PROC,1]^.PONT:=PF.NOVOEND;
229           PF.FIM[ENDE,PROC,1]:=PF.NOVOEND;
230       END;
231       BEGIN
232         LEIAI(ENDE,INI);
233         PVI:=INI.EI DIV TAMPAG +1;
234         PVF:=INI.EF DIV TAMPAG + 1;
235         NP:=PVF-PVI+1;
236         N[ENDE,PROC]:=N[ENDE,PROC]+NP;
237         IF PVF > MAXPV[ENDE,PROC]
238         THEN MAXPV[ENDE,PROC]:=PVF;
239         (* INICIALIZACAO DA TABELA DE PAGINAS E DA VARIAVEL ESTAT  *)
240         FOR J:=PVI TO PVF DO
241           BEGIN
242             ETP.PROTL[ENDE,PROC,J]:=INI.PL;
243             ETP.PROTE[ENDE,PROC,J]:=INI.PE;
244             ESTAT[ENDE,PROC,J]:=1;
245             ETP.MODI[ENDE,PROC,J]:=FALSE;
246           END;
247         IF N[ENDE,PROC]< MAXMP
248         THEN
249           LIMITE[ENDE,PROC]:=N[ENDE,PROC]
250         ELSE
251           LIMITE[ENDE,PROC]:=MAXMP;
252         (*  INICIALIZACAO DOS PONTEIROS PARA AS FILAS  *)
253         IF N[ENDE,PROC] = NP
254         THEN
255           BEGIN
256             FOR J:=2 TO 4 DO
257               BEGIN
258                 PF.COM[ENDE,PROC,J]:=NIL;
259                 PF.FIM[ENDE,PROC,J]:=NIL;
260               END;
261             READLN(LOCAL[ENDE,PROC],FC[ENDE,PROC]);
262           END;
263         IF TABPRO.PROC[ENDE,PROC] <> 0
264         THEN  (*  AREA DE MEMORIA DO PROCESSO A SER INICIALIZADO ESTA
265                   OCUPADA  *)
266           BEGIN
267             I:=1;
268             AUX:=0;
269             WHILE I <= NPROC DO
270               IF TABPRO.INIC[I]
271               THEN
272                 I:=I+1
273               ELSE
274                 IF TABPRO.PROC[I] = 0
275                 THEN
276                   BEGIN
```

```
277                   AUX:=I;
278                   I:=NPROC+1;
279                 END
280               ELSE
281                 I:=I+1;
282           IF AUX <> 0
283           THEN  (*  TRANSFERIR AS PAGINAS PARA OUTRO ESPACO DA M.P. *)
284             BEGIN
285               I:=1;
286               WHILE I<= LIMITEENDE.PROC] DO
287                 BEGIN
288                   AUX2:=LPVEAUX];
289                   ENDREAL:=(AUX-1)*MAXMP * TAMPAG + (AUX2-1) * TAMPAG;
290                   LPVEAUX]:=ETPR.PAGINAEAUX,AUX2];
291                   ETPR.PAGINAEAUX,AUX2]:=ETPR.PAGINAEENDE.PROC,I];
292                   ETPR.VAZIOEAUX,AUX2]:=FALSE;
293                   ETP.ENDEREABPRO.PROCEENDE.PROC]
294                     ,ETPR.PAGINAEENDE.PROC,I]]:=ENDREAL;
295                   I:=I+1;
296                 END;
297               TABPRO.PROCEAUX]:=TABPRO.PROCEENDE.PROC];
298               LIMITEEAUX]:=LIMITEEENDE.PROC];
299             END
300           ELSE  (*  GUARDAR NA MEMORIA SECUNDARIA  *)
301             BEGIN
302               I:=1;
303               WHILE I<= LIMITEEENDE.PROC] DO
304                 BEGIN
305                   X:=ETPR.PAGINAEENDE.PROC,I];
306                   ETP.VALEABPRO.PROCEENDE.PROC],X]:=FALSE;
307                   ETP.MODIEABPRO.PROCEENDE.PROC],X]:=FALSE;
308                   J:=1;
309                   WHILE J <= 4 DO
310                     BEGIN
311                       IF PF.COMEABPRO.PROCEENDE.PROC],J]^.PAGVIRTUAL
312                          =X
313                       THEN
314                         BEGIN
315                           PF.COMEABPRO.PROCEENDE.PROC],J]:=
316                           PF.COMEABPRO.PROCEENDE.PROC],J]^.PONT;
317                           IF ANT ^.PONT=NIL
318                           THEN
319                             PF.FIMEABPRO.PROCEENDE.PROC],J]:=ANT;
320                         END
321                       ELSE
322                         BEGIN
323                           PF.NOVOEND:=PF.COMEABPRO.PROCEENDE.PROC],
324                                J]^.PONT;
325                           ANT:=PF.COMEABPRO.PROCEENDE.PROC],J];
326                           WHILE PF.NOVOEND <> NIL DO
327                             BEGIN
328                               IF PF.NOVOEND^.PAGVIRTUAL=X
329                               THEN
330                                 ANT^.PONT:=PF.NOVOEND^.PONT;
331                               IF ANT^.PONT = NIL
332                               THEN
333                                 PF.FIMEABPRO.PROCEENDE.PROC],J]:=ANT;
334                               PF.NOVOEND:=PF.NOVOEND^.PONT;
335                               ANT:=ANT^.PONT;
336                             END;
337                         END;
338                       J:=J+1;
339                     END;
340                   I:=I+1;
341                 END;
342             END;
343           LPVEENDE.PROC]:=1;
344           FOR I:=1 TO MAXMP DO
345             BEGIN
```

```
346                 ETPR.VAZIO[ENDE.PROC,I]:=TRUE;
347                 ETPR.PAGINA[ENDE.PROC,I]:=I+1;
348               END;
349             ETPR.PAGINA[ENDE.PROC,MAXMP]:=-1;
350           END;
351         PL:=MAXMP-LPV[ENDE.PROC]+1;
352         IF(PL <= MAXMP) AND (PL >= NP)
353         THEN  (* TODAS AS PAGINAS CABEM NA MEMORIA PRINCIPAL  *)
354           BEGIN
355             FOR J:=PVI TO PVF DO
356               BEGIN
357                 I:=LPV[ENDE.PROC];
358                 IF ETP.ENDER[ENDE.PROC,J] = 0
359                 THEN INIMP(ENDE,LPV,PF,ETPR,ETP,J,ENDE.PROC,I);
360               END;
361             TABPRO.INIC[ENDE.PROC]:=TRUE;
362             TABPRO.PROC[ENDE.PROC]:=0;
363           END
364         ELSE
365           BEGIN
366             IF PL <= MAXMP
367             THEN  (*  PARTE DAS PAGINAS CABEM NA MEMORIA PRINCIPAL  *)
368               BEGIN
369                 PV:=PVI;
370                 FOR J:=1 TO PL DO
371                   BEGIN
372                     IF ETP.ENDER[ENDE.PROC,PV] = 0
373                     THEN INIMP(ENDE,LPV,PF,ETPR,ETP,PV,ENDE.PROC,
374                                LPV[ENDE.PROC]);
375                     PV:=PV + 1;
376                   END;
377                 TABPRO.INIC[ENDE.PROC]:=TRUE;
378                 TABPRO.PROC[ENDE.PROC]:=0;
379               END
380             ELSE
381               PV:=PVI;
382             NUMPAG:=PVF-PV+1;
383             WHILE NUMPAG > 0 DO
384               BEGIN
385                 AUX:=0;
386                 I:=1;
387                 WHILE I <= NPROC DO
388                   IF TABPRO.INIC[I]
389                   THEN
390                     I:=I+1
391                   ELSE
392                     IF TABPRO.PROC[I] = 0
393                     THEN
394                       BEGIN
395                         AUX:=I;
396                         I:=NPROC+1;
397                       END
398                     ELSE
399                       I:=I+1;
400                 IF AUX <> 0
401                 THEN  (* EXISTE PROCESSO NAO INICIALIZADO  *)
402                   BEGIN
403                     IF MAXMP >= NUMPAG
404                     THEN
405                       BEGIN
406                         PVFIM:=PVF;
407                         LIMITE[AUX]:=PVF-PV+1;
408                       END
409                     ELSE
410                       BEGIN
411                         PVFIM:=PV+MAXMP-1;
412                         LIMITE[AUX]:=MAXMP;
413                       END;
414                     FOR J:=PV TO PVFIM DO
```

```
415                       IF ETP.ENDER[ENDE.PROC,J] = 0
416                       THEN
417                         ININP(ENDE,LPV,PF,ETPR,ETP,J,AUX,LPV[AUX]);
418                     NUMPAG:=NUMPAG-MAXMP;
419                     PV:=PV+MAXMP;
420                     TABPRO.PROC[AUX]:=ENDE.PROC;
421                   END
422                 ELSE
423                   FOR J:=PV TO PVF DO
424                     BEGIN
425                       IF ETP.ENDER[ENDE.PROC,J]=0
426                       THEN
427                         BEGIN
428                           ETP.VAL[ENDE.PROC,J]:=FALSE;
429                           ETP.ENDER[ENDE.PROC,J]:=-1;
430                           ETP.MODI[ENDE.PROC,J]:=FALSE;
431                           NUMPAG:=0;
432                         END;
433                     END;
434             END;
435         END;
436     END;
437   PROCEDURE ACESSO(VAR ENDE:ENDERECO;VAR ST,TANTERIOR,TREAL:REAL;
438                 VAR ETP:TP;VAR ETPR:TPR;VAR PF:PONTFILA;
439                 VAR LPV,ENDCACHE,ENDMP1,LOCAL:TN;VAR FC:TNR);
440     CONST
441       TEMP=7.3E-6;   (*  TEMPO GASTO PARA ATUALIZAR A TABELA DE PAGINAS
442                          EM CASOS DE ESCRITA NA MEMORIA PRINCIPAL  *)
443     TYPE
444       LOGICO=0..1;
445     VAR
446       PRECACHE,PREMP1,VALCACHE,VALMP1,VALTP,PRL,PRE:BOOLEAN;
447         (* PRECACHE--> INDICA A PRESENCA (OU AUSENCIA) DO ENDERECO
448                        A QUE SE ESTA TENDO ACESSO NO CACHE
449            PREMP1  --> INDICA SE A PAGINA EM QUE SE ESTA TENDO UM
450                        ACESSO COINCIDE COM A ULTIMA PAGINA ACESSADA PARA
451                        LEITURA
452            VALCACHE--> INDICA SE O ACESSO PRETENDIDO A UM DADO QUE
453                        ESTA NO CACHE E VALIDO
454            VALMP1  --> INDICA SE O ACESSO PRETENDIDO PARA UM DADO QUE
455                        ESTA NA ULTIMA PAGINA QUE TEVE UM ACESSO PARA
456                        LEITURA E' VALIDO
457            VALTP   --> INDICA SE O ACESSO PRETENDIDO PARA UMA PAGINA
458                        VIRTUAL E' VALIDO
459            PRL     --> INDICA A PROTECAO CONTRA LEITURA DA ULTIMA
460                        PAGINA ONDE SE TEVE ACESSO PARA LEITURA
461            PRE     --> INDICA A PROTECAO CONTRA ESCRITA DA ULTIMA
462                        PAGINA ONDE SE TEVE ACESSO PARA LEITURA  *)
463       AUX,J,PAGVIRT,ENDBASE:INTEGER;
464         (* AUX,J  --> VARIAVEIS AUXILIARES
465            PAGVIRT--> PAGINA VIRTUAL ONDE SE TEVE UM ACESSO
466            ENDBASE--> ENDERECO BASE NO CACHE PARA O ENDERECO QUE ESTA
467                       RECEBENDO UM ACESSO  *)
468       AUXJ:REAL;
469         (* AUXJ  --> VARIAVEL AUXILIAR  *)
470       WBARRA:LOGICO;
471         (* WBARRA  --> INDICA O TIPO DE ACESSO PRETENDIDO
472                        WBARRA = 1 -> LEITURA
473                        WBARRA = 0 -> ESCRITA *)
474       FUNCTION TACESSO(WBARRA:LOGICO;PL,PE:BOOLEAN):BOOLEAN;
475         BEGIN
476           IF((WBARRA=1)AND PL) OR ((WBARRA = 0) AND PE)
477           THEN
478             TACESSO:=TRUE
479           ELSE
480             TACESSO:=FALSE;
481         END;
482       FUNCTION PCACHE(VAR ENDCACHE:TN;ENDE:ENDERECO;VAR PAGVIRT
483                   :INTEGER;WBARRA:LOGICO;ETP:TP;VAR VALCACHE,PRL,
```

```
484                 PRE:BOOLEAN;VAR ST,TANTERIOR:REAL):BOOLEAN;
485           CONST
486             TCACHEL=7.72E-8;(* TEMPO GASTO NA OBTENCAO DE UM DADO QUE ESTA
487                               NO CACHE  *)
488             TCACHEE=9.32E-8;(* TEMPO GASTO NA ESCRITA DE UM DADO NO CACHE*)
489           BEGIN
490             ENDBASE:=ENDE.ENDER-ENDE.ENDER MOD 4;
491             IF ENDBASE=ENDCACHE[ENDE.PROC]
492             THEN   (*  ENDERECO REQUERIDO ESTA NO CACHE  *)
493               BEGIN
494                 VALCACHE:=TACESSO(WBARRA,PRL,PRE);
495                 IF VALCACHE
496                 THEN
497                   BEGIN
498                     IF WBARRA = 1
499                     THEN
500                       BEGIN
501                         TDUTO:=TDUTO+TCACHEL;
502                         ST:=ST+TCACHEL;
503                       END
504                     ELSE
505                       BEGIN
506                         TDUTO:=TDUTO+TCACHEE;
507                         ST:=ST+TCACHEE;
508                       END;
509                   END;
510                 NCACHE:=NCACHE+1;
511                 IF WBARRA = 0
512                 THEN
513                   BEGIN
514                     TANTERIOR:=TANTERIOR+TEMP;
515                     ETP.MODI[ENDE.PROC,PAGVIRT]:=TRUE;
516                   END;
517                 PCACHE:=TRUE;
518               END
519             ELSE   (*  ENDERECO REQUERIDO NAO ESTA NO CACHE  *)
520               BEGIN
521                 PCACHE:=FALSE;
522                 IF WBARRA = 1
523                 THEN    (*  TROCAR O ENDERECO DO CACHE  *)
524                   BEGIN
525                     IF ENDCACHE[ENDE.PROC] = -1
526                     THEN
527                       BEGIN
528                         PRE:=ETP.PROTE[ENDE.PROC,PAGVIRT];
529                         PRL:=ETP.PROTL[ENDE.PROC,PAGVIRT];
530                       END;
531                     ENDCACHE[ENDE.PROC] :=ENDBASE;
532                   END;
533               END;
534           END;
535         FUNCTION MEMPRIN1(VAR ENDE:ENDERECO;VAR PAGVIRT:INTEGER;
536                     WBARRA:LOGICO;VAR ENDMP1:TN;ETP:TP;VAR VALMP1,PRL,
537                     PRE:BOOLEAN;VAR ST,TANTERIOR:REAL):BOOLEAN;
538           CONST
539             TMP1=8.1E-8;(*TEMPO GASTO NA OBTENCAO DE UM DADO QUE ESTA NA
540                           ULTIMA PAGINA ONDE SE TEVE ACESSO PARA LEITURA *)
541           BEGIN
542             IF PAGVIRT=ENDMP1[ENDE.PROC]
543             THEN    (*  PAGINA EM QUE SE TEM O ACESSO COINCIDE COM A ULTIMA
544                         A QUE SE TEVE ACESSO  *)
545               BEGIN
546                 VALMP1:=TACESSO(WBARRA,PRL,PRE);
547                 IF VALMP1
548                 THEN
549                   BEGIN
550                     TDUTO:=TDUTO+TMP1+TAMP;
551                     ST:=ST+TMP1+TAMP;
552                   END;
```

```
553             MEMPRIN1:=TRUE;
554             IF WBARRA = 0
555             THEN
556               BEGIN
557                 TANTERIOR:=TANTERIOR+TEMP;
558                 ETP.MODI[ENDE.PROC,PAGVIRT]:=TRUE;
559               END;
560             NMP1:=NMP1+1;
561           END
562         ELSE   (*  PAGINA ONDE SE TEM O ACESSO NAO COINCIDE  COM ACESSO
563                    ANTERIOR  *)
564           BEGIN
565             MEMPRIN1:=FALSE;
566             IF WBARRA=1
567             THEN
568               BEGIN
569                 ENDMP1[ENDE.PROC]:=PAGVIRT;
570                 PRE:=ETP.PROTE[ENDE.PROC,PAGVIRT];
571                 PRL:=ETP.PROTL[ENDE.PROC,PAGVIRT];
572               END;
573           END;
574       END;
575     PROCEDURE MEMPRIN2(VAR ST,TANTERIOR:REAL;WBARRA:LOGICO;VAR ETP:TP;
576                        ENDE:ENDERECO;PAGVIRT:INTEGER);
577       CONST
578         TMP2=1.401E-7; (* TEMPO GASTO NA OBTENCAO DE UM DADO QUE ESTA
579                           NA MEMORIA PRINCIPAL  *)
580       BEGIN
581         ST:=ST+TMP2+2*TAMP;
582         TDUTO:=TDUTO+TMP2+2*TAMP;
583           IF WBARRA = 0
584           THEN
585             BEGIN
586               TANTERIOR:=TANTERIOR+TEMP;
587               ETP.MODI[ENDE.PROC,PAGVIRT]:=TRUE;
588             END;
589         NMP2:=NMP2+1;
590       END;
591     PROCEDURE MEMSEC(VAR PF:PONTFILA;VAR ETP:TP;VAR TANTERIOR,ST
592                     ,TREAL:REAL;VAR ESTAT:TMSI;WBARRA:LOGICO;VAR FC:TNR;
593                     ENDE:ENDERECO;PAGVIRT:INTEGER;VAR LOCAL:TN);
594       CONST
595         THARD=1.479E-7;      (* TEMPO GASTO COM O HARDWARE, QUANDO A
596                                 PAGINA ONDE O ACESSO E' FEITO ESTA' NA
597                                 MEMORIA SECUNDARIA  *)
598         TSOFT=2.7904E-4;    (* TEMPO GASTO COM O SOFTWARE, ANTES DE
599                                 LIBERAR O DADO REQUISITADO, QUANDO NAO
600                                 EXISTE ESPACO NA MEMORIA PRINCIPAL *)
601         TSOFTP=4.2558E-4;   (* TEMPO GASTO COM O SOFTWARE, APOS LIBERAR
602                                 O DADO REQUISITADO, QUANDO NAO EXISTE
603                                 ESPACO NA MEMORIA PRINCIPAL *)
604         TLIBDUTO=1.0E-5;   (*  TEMPO GASTO PARA LIBERAR O DUTO  *)
605       TYPE
606         AUXI=^FILLA;
607       VAR
608         PAGINAV,FILAX,ENTPR,NPAGUAR:INTEGER;
609           (* PAGINAV  --> PAGINA VIRTUAL QUE SERA RETIRADA DA MEMORIA
610                           PRINCIPAL
611              FILAX    --> FILA QUE SERA UTILIZADA PARA OBTENCAO DO
612                           ELEMENTO A SER RETIRADO
613              ENTPR    --> ENTRADA DA TPR QUE SERA ALTERADA
614              NPAGUAR  --> NUMERO DE PAGINAS QUE FOI POSSIVEL GUARDAR
615                           DURANTE VARTEMPO *)
616         VARTEMPO:REAL;
617           (* VARTEMPO --> TEMPO DISPONIVEL ENTRE DOIS ACESSOS A
618                           MEMORIA SECUNDARIA  *)
619         AUX,AUXT:AUXI;
620           (* AUX  --> VARIAVEL AUXILIAR UTILIZADA NA ATUALIZACAO DA
621                       FILA
```

```
622               AUXT --> VARIAVEL AUXILIAR UTILIZADA NA VERIFICAO DAS
623                        PAGINAS QUE FORAM GUARDADAS NA MEMORIA
624                        SECUNDARIA DURANTE O TEMPO DISPONIVEL *)
625           LIBDUTO:BOOLEAN;
626         FUNCTION RETIRAR(VAR PF:PONTFILA):INTEGER;
627           VAR
628             AUXILIAR:INTEGER;
629           BEGIN
630             AUXILIAR:=1;
631             WHILE PF.COMCENDE.PROC,AUXILIAR] = NIL DO
632             AUXILIAR:=AUXILIAR + 1;
633             RETIRAR:=AUXILIAR;
634           END;
635         BEGIN
636           LIBDUTO:=FALSE;
637           NMS:=NMS+1;
638           ST:=ST + THARD+2*TAMP;
639           TDUTO:=TDUTO+THARD+2*TAMP;
640           VARTEMPO:=TREAL - TANTERIOR;
641           IF VARTEMPO > 0
642           THEN  (* SOFTWARE PARALELO COMPLETO - VERIFICA-SE O NUMERO DE
643                    PAGINAS GUARDADAS DURANTE O TEMPO OCIOSO  *)
644             BEGIN
645               AUXT:=PF.COMCENDE.PROC,RETIRAR(PF)];
646               NPAGUAR:=TRUNC(VARTEMPO / (TGUARDAR+TAMPAG*16/TTRANS));
647               WHILE(NPAGUAR <> 0) AND (AUXT <> NIL) DO
648                 BEGIN
649                   IF ETP.MODICENDE.PROC,AUXT^.PAGVIRTUAL]
650                   THEN
651                     BEGIN
652                       ETP.MODICENDE.PROC,AUXT^.PAGVIRTUAL]:=FALSE;
653                       NPAGUAR:=NPAGUAR - 1;
654                     END;
655                   AUXT:=AUXT^.PONT;
656                 END;
657             END
658           ELSE   (*  SOFTWARE PARALELO NAO COMPLETO  *)
659             BEGIN
660               ST:=ST - VARTEMPO;
661               TREAL:=TREAL - VARTEMPO;
662               TDUTO:=TDUTO-VARTEMPO;
663             END;
664           (*  INICIO DO SOFTWARE ANTES DA LIBERACAO DO DADO  *)
665           FILAX:=RETIRAR(PF);
666           WHILE FILAX < 5 DO
667             BEGIN
668               AUX:=PF.COMCENDE.PROC,FILAX];
669               REPEAT
670                 PAGINAV:=AUX^.PAGVIRTUAL;
671                 AUX:=AUX^.PONT;
672               UNTIL (AUX = NIL) OR (PAGINAV <> ENDMP1CENDE.PROC]);
673               IF (AUX = NIL) AND (PAGINAV = ENDMP1CENDE.PROC])
674               THEN
675                 REPEAT
676                   FILAX:=FILAX+1;
677                 UNTIL PF.COMCENDE.PROC,FILAX] <> NIL
678               ELSE
679                 FILAX:=5;
680             END;
681           FILAX:=RETIRAR(PF);
682           IF ETP.MODICENDE.PROC,PAGINAV]
683           THEN
684             BEGIN
685               ETP.MODICENDE.PROC,PAGINAV]:=FALSE;
686               ST:=ST+TGUARDAR+TAMPAG*16/TTRANS;
687               TREAL:=TREAL+TGUARDAR+TAMPAG*16/TTRANS;
688               LIBDUTO:=TRUE;
689               TDUTO:=TDUTO+TLIBDUTO;
```

```
690               NGUARDAR:=NGUARDAR+1;
691             END;
692           IF ETP.ENDER[ENDE,PROC,PAGVIRT] >= 0
693           THEN
694             BEGIN
695               ST:=ST+TTRAZER+TAMPAG*16/TTRANS;
696               TREAL:=TREAL+TTRAZER+TAMPAG*16/TTRANS;
697               NTRAZER:=NTRAZER+1;
698               IF NOT LIBDUTO
699               THEN TDUTO:=TDUTO+TLIBDUTO;
700               ESTAT[ENDE,PROC,PAGVIRT]:=ESTAT[ENDE,PROC,PAGVIRT]+1;
701             END;
702           (*  FINAL DO SOFTWARE ANTES DA LIBERACAO DO DADO  *)
703           (*  INICIO DO SOFTWARE APOS A LIBERACAO DO DADO   *)
704           ETP.VAL[ENDE,PROC,PAGINAV]:=FALSE;
705           ETP.VAL[ENDE,PROC,PAGVIRT]:=TRUE;
706           IF WBARRA = 0
707           THEN ETP.MODI[ENDE,PROC,PAGVIRT]:=TRUE;
708           ETP.ENDER[ENDE,PROC,PAGVIRT]:=ETP.ENDER[ENDE,PROC,PAGINAV];
709           ENTPR:=1;
710           WHILE ETPR.PAGINA[ENDE,PROC,ENTPR] <> PAGINAV DO
711             ENTPR:=ENTPR+1;
712           ETPR.PAGINA[ENDE,PROC,ENTPR]:=PAGVIRT;
713           (* ATUALIZACAO DA FILA EM QUE A PAGINA FOI RETIRADA  *)
714           AUX:=PF.COM[ENDE,PROC,FILAX];
715           PF.COM[ENDE,PROC,FILAX]:=PF.COM[ENDE,PROC,FILAX]^.PONT;
716           IF PF.COM[ENDE,PROC,FILAX]=NIL
717           THEN
718             PF.FIM[ENDE,PROC,FILAX]:=NIL;
719           IF ESTAT[ENDE,PROC,PAGVIRT]>4
720           THEN
721             ESTAT[ENDE,PROC,PAGVIRT]:=4;
722           IF PF.FIM[ENDE,PROC,ESTAT[ENDE,PROC,PAGVIRT]] <> NIL
723           THEN
724             PF.FIM[ENDE,PROC,ESTAT[ENDE,PROC,PAGVIRT]]^.PONT:=AUX
725           ELSE
726             PF.COM[ENDE,PROC,ESTAT[ENDE,PROC,PAGVIRT]]:=AUX;
727           PF.FIM[ENDE,PROC,ESTAT[ENDE,PROC,PAGVIRT]]:=AUX;
728           PF.FIM[ENDE,PROC,ESTAT[ENDE,PROC,PAGVIRT]]^.PONT:=NIL;
729           PF.FIM[ENDE,PROC,ESTAT[ENDE,PROC,PAGVIRT]]^.PAGVIRTUAL:=PAGVIRT;
730           ST:=ST+TSOFT;
731           TDUTO:=TDUTO+TSOFT;
732           TREAL:=TREAL+TSOFT;
733           TANTERIOR:=TREAL+TSOFTP;
734           (*  FINAL DO SOFTWARE APOS A LIBERACAO DO DADO  *)
735         END;
736       BEGIN
737         AUXJ:=LOCAL[ENDE,PROC]/(100-LOCAL[ENDE,PROC]);
738         J:=TRUNC(AUXJ)+1;
739         RESTO:=RESTO+AUXJ-TRUNC(AUXJ);
740         IF RESTO >= 1
741         THEN
742           BEGIN
743             J:=J+1;
744             RESTO:=RESTO-1;
745           END;
746         WHILE J>0 DO
747           BEGIN
748             TREAL:=TREAL+PROB[(RUNTIME+J*100+NACESSO MOD 50) MOD 100 +1]
749             /FC[ENDE,PROC];
750             J:=J-1;
751           END;
752         NACESSO:=NACESSO+1;
753         READ(WBARRA);
754         IF WBARRA = 1
755         THEN LEITURA:=LEITURA + 1
756         ELSE ESCRITA:=ESCRITA + 1;
757         WITH ENDE DO
758         READLN(MEM,PROC,ENDER);
```

```
759         IF ENDE.MEM = 1
760         THEN   (*  ACESSO A MEMORIA DO SDF  *)
761           BEGIN
762             VALTP:=TRUE;
763             PAGVIRT:=ENDE.ENDER DIV TAMPAG + 1;
764             IF EXCACHE
765             THEN
766               PRECACHE:=PCACHE(ENDCACHE,ENDE,PAGVIRT,WBARRA,ETP,
767               VALCACHE,PRL,PRE,ST,TANTERIOR)
768             ELSE
769               PRECACHE:=FALSE;
770             IF NOT PRECACHE
771             THEN
772               BEGIN
773                 PREMP1:=MEMPRIN1(ENDE,PAGVIRT,WBARRA,ENDMP1,ETP,
774                 VALMP1,PRL,PRE,ST,TANTERIOR);
775                 IF NOT PREMP1
776                 THEN
777                   BEGIN
778                     VALTP:=TACESSO(WBARRA,ETP.PROTL[ENDE.PROC,PAGVIRT],
779                     ETP.PROTE[ENDE.PROC,PAGVIRT]);
780                     IF VALTP
781                     THEN
782                       IF ETP.VAL[ENDE.PROC,PAGVIRT]=TRUE
783                       THEN
784                         MEMPRIN2(ST,TANTERIOR,WBARRA,ETP,ENDE,PAGVIRT)
785                       ELSE
786                         MEMSEC(PF,ETP,TANTERIOR,ST,TREAL,ESTAT,
787                           WBARRA,FC,ENDE,PAGVIRT,LOCAL);
788                   END;
789               END;
790             IF(PRECACHE AND NOT VALCACHE) OR (PREMP1 AND NOT VALMP1) OR
791             (NOT VALTP)
792             THEN
793               WRITELN('ERRO DE ACESSO');
794           END
795         ELSE
796           WRITELN('ACESSO A MEMORIA COMUM ');
797       END;
798     PROCEDURE IMPRESSAO(VAR ETP:TP;VAR ETPR:TPR;VAR NP:INTEGER;VAR N,
799                   LIMITE,MAXPV:TN;VAR ESTAT:TMSI;VAR PF:PONTFILA;
800                   ENDE:ENDERECO;TABPRO:TABPROC);
801       TYPE
802         AUXI=^FILLA;
803       VAR
804         J1,I:INTEGER;
805         I2:AUXI;
806       BEGIN
807         WHILE NOT EOLN DO
808           BEGIN
809             READ(ENDE.PROC);
810             WRITELN('       PROCESSO-',ENDE.PROC:3);
811             WRITELN('TABELA DA PAGINAS VIRTUAIS');
812             WRITELN('VALIDADE MODIFICACAO PROT.LEIT. PROT.ESCR. ENDERECO'
813             );
814             FOR J1:=1 TO MAXPV[ENDE.PROC] DO
815               WRITELN(ETP.VAL[ENDE.PROC,J1]:8,ETP.MODI
816                 [ENDE.PROC,J1]:12,ETP.PROTL[ENDE.PROC,J1]:11,
817                 ETP.PROTE[ENDE.PROC,J1]:11,ETP.ENDER[ENDE.PROC,J1]:9);
818             WRITELN;
819             WRITELN('TABELA DAS PAGINAS REAIS');
820             WRITELN('VAZIO  PAGINA');
821             FOR J1:=1 TO LIMITE[ENDE.PROC] DO
822               WRITELN(ETPR.VAZIO[ENDE.PROC,J1]:6,
823                 ETPR.PAGINA[ENDE.PROC,J1]:5);
824             FOR I:=1 TO NPROC DO
825               IF TABPRO.PROC[I] = ENDE.PROC
826               THEN
827                 FOR J1:=1 TO LIMITE[I] DO
```

```
828                 WRITELN(ETPR.VAZIO[I,J1]:6,ETPR.PAGINA[I,J1]:5);
829             WRITELN;
830             WRITELN('TABELA DE PROCESSOS - INFORMACOES SOBRE AS AREAS ',
831               'DE MEMORIA');
832             WRITELN('INICIALIZADO  PROCESSO');
833             FOR J1:=1 TO NPROC DO
834               WRITELN(TABPRO.INIC[J1]:10,TABPRO.PROC[J1]:10);
835             WRITELN;
836             WRITELN('ESTATISTICA SOBRE A RETIRADA DE PAGINAS DA MEMORIA'
837               ,' SECUNDARIA');
838             FOR J1:=1 TO MAXPV[ENDE.PROC] DO
839               WRITELN(ESTAT[ENDE.PROC,J1]);
840             WRITELN;
841             WRITELN('FILAS PARA RETIRADA DE PAGINAS DE MEMORIA',
842               ' PRINCIPAL');
843             J1:=1
844             WHILE J1<5 DO
845               BEGIN
846                 WRITELN('   FILA ',J1:2);
847                 I2:=PF.COM[ENDE.PROC,J1];
848                 WHILE I2<> NIL DO
849                   BEGIN
850                     WRITELN(I2^.PAGVIRTUAL:9);
851                     I2:=I2^.PONT;
852                   END;
853                 J1:=J1+1;
854               END;
855           END;
856         READLN;
857       END;
858     PROCEDURE FINALIZA(VAR N,LPV,ENDCACHE,ENDMP1,MAXPV:TN;ENDE:ENDERECO;
859                   VAR TABPRO:TABPROC);
860       VAR
861         I,J:INTEGER;
862       BEGIN
863         READLN(ENDE.PROC);
864         N[ENDE.PROC]:=0;
865         LPV[ENDE.PROC]:=1;
866         ENDCACHE[ENDE.PROC]:=-1;
867         ENDMP1[ENDE.PROC]:=-1;
868         MAXPV[ENDE.PROC]:=0;
869         FOR J:=1 TO MAXMP DO
870           BEGIN
871             ETPR.VAZIO[ENDE.PROC,J]:=TRUE;
872             ETPR.PAGINA[ENDE.PROC,J]:=J+1;
873             ETPR.PAGINA[ENDE.PROC,MAXMP]:=-1;
874           END;
875         FOR J:=1 TO MAXMS DO
876           ETP.ENDER[ENDE.PROC,J]:=0;
877         TABPRO.INIC[ENDE.PROC]:=FALSE;
878         FOR I:=1 TO NPROC DO
879           IF TABPRO.PROC[I] = ENDE.PROC
880           THEN
881             BEGIN
882               LPV[I]:=1;
883               TABPRO.PROC[I]:=0;
884               TABPRO.INIC[I]:=FALSE;
885               FOR J:=1 TO MAXMP DO
886                 BEGIN
887                   ETPR.VAZIO[I,J]:=TRUE;
888                   ETPR.PAGINA[I,J]:=J+1;
889                 END;
890               ETPR.PAGINA[ENDE.PROC,MAXMP]:=-1;
891               FOR J:=1 TO MAXMS DO
892                 ETP.ENDER[I,J]:=0;
893             END;
894       END;
895     BEGIN
896       (* INICIALIZACOES  *)
```

```
897       READLN(I,TGUARDAR,TTRAZER,TTRANS,TAMP);
898       IF I=1
899       THEN EXCACHE:=TRUE
900       ELSE EXCACHE:=FALSE;
901       FOR I:=1 TO NPROC DO
902         BEGIN
903           FOR J:=1 TO MAXMP DO
904             BEGIN
905               ETPR.VAZIO[I,J]:=TRUE;
906               ETPR.PAGINA[I,J]:=J+1;
907             END;
908           ETPR.PAGINA[I,MAXMP]:=-1;
909           FOR J:=1 TO MAXMS DO
910             ETP.ENDER[I,J]:=0;
911           N[I]:=0;
912           LPV[I]:=1;
913           ENDCACHE[I]:=-1;
914           ENDMP1[I]:=-1;
915           MAXPV[I]:=0;
916           TABPRO.PROC[I]:=0;
917           TABPRO.INIC[I]:=FALSE;
918         END;
919       TREAL:=0;
920       TOUTO:=0;
921       ST:=0;
922       NCACHE:=0;
923       NMP1:=0;
924       NMP2:=0;
925       NGUARDAR:=0;
926       NTRAZER:=0;
927       NMS:=0;
928       NACESSO:=0;
929       ESCRITA:=0;
930       LEITURA:=0;
931       TANTERIOR:=0;
932         (* INICIALIZACAO DO VETOR DE PROBABILIDADES *)
933         FOR I:=1 TO 39 DO
934           PROB[I]:=3;
935         FOR I:=40 TO 95 DO
936           PROB[I]:=4;
937         FOR I:=96 TO 99 DO
938           PROB[I]:=5;
939         PROB[I]:=10;
940         RESTO:=0;
941       (* EXECUCAO DOS COMANDOS  *)
942       WHILE NOT EOF DO
943         BEGIN
944           READ(COMANDO);
945           CASE(COMANDO)OF
946             'I':INICIALIZA(ETP,ETPR,ENDE,NP,ESTAT,PF,N,LIMITE,LPV,
947                 MAXPV,LOCAL,FC,TABPRO);
948             'A':ACESSO(ENDE,ST,TANTERIOR,TREAL,ETP,ETPR,PF,LPV,ENDCACHE,
949                 ENDMP1,LOCAL,FC);
950             'P':IMPRESSAO(ETP,ETPR,NP,N,LIMITE,MAXPV,ESTAT,PF,ENDE,
951                 TABPRO);
952             'F':FINALIZA(N,LPV,ENDCACHE,ENDMP1,MAXPV,ENDE,TABPRO);
953             OTHERS:WRITELN('ERRO - TENTATIVA DE EXECUCAO DE COMANDO',
954                    'NAO EXISTENTE');
955           END;
956         END;
957       WRITELN('SIMULACAO DE UM MODULO COM MEMORIA VIRTUAL');
958       WRITELN;
959       WRITELN(' CARACTERISTICAS DO MODULO:');
960       WRITELN('  .TAMANHO DAS PAGINAS - ',TAMPAG:3,' PALAVRAS');
961       WRITELN('  .NUMERO DE PAGINAS NA MEMORIA PRINCIPAL',
962         ' POR PROCESSO - ',MAXMP:2);
963       WRITELN('  .NUMERO DE PAGINAS NA MEMORIA SECUNDARIA',
964         ' POR PROCESSO - ',MAXMS:3);
```

```
965       WRITELN('  .NUMERO MAXIMO DE PROCESSOS - ',NPROC:2);
966       WRITELN('  .EXISTENCIA DE UM CACHE COM QUATRO PALAVRAS',
967         ' POR PROCESSO - ',EXCACHE);
968       WRITELN('  .PORCENTAGEM DOS ACESSOS A MEMORIA PROCESSADOS '
969         ,'LOCALMENTE - ',LOCALENDE.PROC]:3,'%');
970       WRITELN('  .TEMPO DE ACESSO A MEMORIA SECUNDARIA - ',TGUARDAR,
971         ' SEGUNDOS');
972       WRITELN('  .TAXA DE TRANSF. PARA MEMORIA SECUNDARIA - ',
973         TTRANS:7:3,'   BITS/SEG.');
974       WRITELN('  .TEMPO DE ACESSO A MEMORIA PRINCIPAL - ',TAMP,
975         ' SEGUNDOS');
976       WRITELN;
977       WRITELN(' RESULTADOS OBTIDOS COM A SIMULACAO');
978       WRITELN('  .NUMERO DE ACESSOS AO MODULO DE MEMORIA ',
979         '- ',NACESSO:5);
980       WRITELN('  .NUMERO DE ACESSOS PARA LEITURA - ',LEITURA:5);
981       WRITELN('  .NUMERO DE ACESSOS PARA ESCRITA - ',ESCRITA:5);
982       WRITELN('  .TEMPO GASTO NO MODULO DE MEMORIA - ',ST,' SEGUNDOS');
983       WRITELN('  .TEMPO TRANSCORRIDO DURANTE A EXECUCAO - ',TREAL,
984         ' SEGUNDOS');
985       WRITELN('      PORCENTAGEM DE TEMPO EM QUE O DUTO ESTEVE LIVRE ');
986       WRITELN('       -SEM SUA LIBERACAO EM ACESSOS A MEMORIA SECUNDARIA',
987         ' - ',(TREAL-ST)/TREAL*100,'%');
988       WRITELN('       -COM SUA LIBERACAO EM ACESSOS A MEMORIA ',
989         'SECUNDARIA - ',(TREAL-TDUTO)/TREAL*100,'%');
990       WRITELN('  .NUMERO DE ACESSOS AO CACHE - ',NCACHE);
991       WRITELN('      TAXA DE ACERTO - ',NCACHE /
992         NACESSO*100,'%');
993       WRITELN('  .NUMERO DE ACESSOS NA ULTIMA PAGINA LIDA - ',
994         NMP1:5);
995       WRITELN('      TAXA DE ACERTO - ',NMP1 / NACESSO*100,'%');
996       WRITELN('  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL',
997         ' COM O ENDERECO REAL NA TPV - ',NMP2:5);
998       WRITELN('      TAXA DE ACERTO - ',
999         NMP2/NACESSO*100,'%');
1000      WRITELN('  .NUMERO DE ACESSOS A MEMORIA PRINCIPAL + CACHE - ',
1001        NCACHE+NMP1+NMP2:5);
1002      WRITELN('  .NUMERO DE ACESSOS A MEMORIA SECUNDARIA - ',NMS:3);
1003      WRITELN('  .NUMERO DE PAGINAS  GUARDADAS NA MEMORIA SECUNDARIA',
1004        ' - ',NGUARDAR:3);
1005      WRITELN('  .NUMERO DE PAGINAS TRAZIDAS DA MEMORIA SECUNDARIA ',
1006        '- ',NTRAZER:3);
1007    END.
```

C. ABREVIAÇÕES UTILIZADAS

$\overline{AS}$ - Validação de Endereço

$\overline{BERR}$ - Indica a ocorrência de erros

DCAN - Duto de Comunicação de Alto Nível

DDE - Duto de Dados e Endereços

$\overline{DTACK}$ - Fornece informações sobre a disponibilidade do dado

EV - Endereço Virtual

Hab - Habilitação do acoplador para obtenção do endereço real a que se quer ter acesso

Hab_i - Habilitação do registrador contendo o endereço real da última página lida do processo i, para obtenção do endereço real a que se quer ter acesso

HE_i - Indica se a escrita pretendida no processo i é permitida

HL_i - Indica se a leitura pretendida no processo i é permitida

MEMSEC - Sinal enviado pelo processador do módulo de memória virtual, indicando que a página requisitada já foi transferida para a memória principal

MP - Memória Principal

MS - Memória Secundária

PC_i - Indica se ocorreram erros de acesso no cache i

Sc - Indica se o endereço requisitado está no cache

Sc_i - Indica se o endereço requisitado está na linha i do cache

SDF - Sistema Distribuído Funcional

Sel_i - Indica se o processo i está tendo um acesso

Sp - Indica se o endereço requisitado pertence a uma página, que tem seu endereço real em um registrador

Sp_i - Indica se o endereço requisitado pertence a última página, onde se teve um acesso para leitura, no pro

cesso i

TP - Tabela de Processos

TPR - Tabela de Páginas Reais

TPV - Tabela de Páginas Virtuais

$\overline{w}$ - tipo de acesso pretendido ($\overline{w}$ = 1 → leitura; $\overline{w}$ = 0 → escrita)