Desenvolvimento e utilização de recursos educacionais abertos para colaborar com o ensino de memória virtual

Carlos Emílio de Andrade Cacho

Carlos Emílio de Andrade Cacho

Desenvolvimento e utilização de recursos educacionais abertos para colaborar com o ensino de memória virtual

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Paulo Sérgio Lopes de Souza

USP – São Carlos
Dezembro de 2015

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

C634d

Cacho, Carlos Emílio de Andrade
Desenvolvimento e utilização de recursos educacionais abertos para colaborar com o ensino de memória virtual / Carlos Emílio de Andrade Cacho; orientador Paulo Sérgio Lopes de Souza. - São Carlos - SP, 2015.
187 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2015.

1. Memória Virtual. 2. Recursos Educacionais Abertos. 3. Ensino. 4. Amnesia. I. Souza, Paulo Sérgio Lopes de, orient. II. Título.

Carlos Emílio de Andrade Cacho

Development and utilization of open educational resources to collaborate with the virtual memory teaching

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Paulo Sérgio Lopes de Souza

USP – São Carlos
December 2015

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus. Serei eternamente grato aos meus pais por todo o apoio, carinho e amor. A minha mãe Maria Cleide que sempre esteve preocupada, mas sempre esteve me apoiando e me guiando. Ao meu pai Roberto Samuel que sempre buscou as melhores formas para me conduzir as minhas conquistas, não seria nada sem eles e tudo que fizeram por mim.

Gostaria de agradecer ao meu irmão Judson por me apoiar e me ajudar sempre quando necessitei. Hoje não estaria aqui se não fosse a sua ajuda, quando morei em Cuiabá. Agradecer também a minha cunhada Lia por me apoiar no tempo que estive na sua casa e por me presentear com a mais linda sobrinha e afilhada que existe. Agradecer a minha irmã Roberta, que me deu várias de suas mesadas (sem saber) e por ser a irmã mais linda que existe.

Não poderia deixar de esquecer da minha tia e segunda mãe Angela Zorzatto, por me ajudar durante os quatro anos da minha graduação e por ser acolhedora, aconselhadora, carinhosa e preocupada (praticamente uma mãe). Também não poderia deixar de falar do meu tio e segundo pai (professor doutor) José Roberto Zorzatto, por todos seus concelhos e por ser minha inspiração para a vida acadêmica, sempre estará em minha memória.

Agradecer a todos da família Andrade e da família Cacho, por sempre ser meu porto seguro em tudo que precisei e por passar momentos felizes ao lados de vocês. Só não irei citar os nomes pois uma folha não seria suficiente.

Agradecer ao meu Orientador Professor Paulo Sérgio Lopes de Souza, por cumprir a sua tarefa de orientação da melhor forma possível, não imaginaria forma de orientação melhor. O nosso trabalho foi definido na nossa primeira reunião e sempre fui guiado da melhor maneira possível. Tenho certeza que ganhei além de uma formação acadêmica, ganhei uma formação pessoal. Ensinamentos que levarei para o resto da vida, muito obrigado por tudo.

Agradecer as pessoas envolvidas no meu projeto, principalmente a Prof.ª Sarita Mazzini Bruschi que foi praticamente a Co-Orientadora desse projeto. Agradecer ao Fernando Tiosso e a Prof.ª Ellen Francine Barbosa, que foram pessoas fundamentais no meu projeto.

Agradecer a todos meus amigos do LaSDPC (Laboratório de Sistemas Distribuídos e Programação Concorrente) e da República Vegas, certamente construí amizades para o resto de minha vida. Foram muitas histórias construída com esse pessoal que não esquecerei tão fácil.

Gostaria de agradecer a todas as pessoas que me apoiaram e me ajudaram de alguma forma, mas são tantas que uma folha não é suficiente. Serei eternamente grato a todos.

“Eu acredito, que às vezes são as pessoas que ninguém espera nada que fazem as coisas que ninguém consegue imaginar”

(Alan Turing)

RESUMO

CACHO, C. E. A. **Desenvolvimento e utilização de recursos educacionais abertos para colaborar com o ensino de memória virtual**. 2015. 187 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

O ensino de computação envolve muitos assuntos que formam a base para uma aprendizagem eficaz. A falta de recursos apropriados torna difícil a apresentação de tais assuntos de forma clara, devido à dinâmica e à complexidade dos mesmos. O ensino de assuntos referentes ao conteúdo de hierarquia de memória e memória virtual é desafiador, porque ambos consideram diferentes aspectos estruturais, funcionais e de desempenho. Muitas abordagens vêm sendo estudadas para tornar o ensino desses assuntos mais atrativo. Uma dessas abordagens considera o desenvolvimento e aplicação de Recursos Educacionais Abertos (REA). Os REA têm sido aplicados com sucesso para ajudar o ensino e a aprendizagem de assuntos desafiadores. Mesmo com conhecimento da possibilidade do uso de REA, muito conteúdos ainda são desprovidos desses recursos. Tendo isso em mente, este trabalho apresenta a transformação do simulador Amnesia em um REA para facilitar o ensino e o aprendizado de memória virtual, simulando aspectos estruturais, funcionais e de desempenho. O foco deste trabalho é melhorar o ensino de Memória Virtual com o auxílio do REA Amnesia. Foram desenvolvidos materiais didáticos para auxiliar os professores e alunos com planos de aula, tutorial de utilização e texto com conteúdo teórico, os quais são disponibilizados juntamente com o REA Amnesia. Para verificar a melhora no ensino e aprendizado foram realizados três experimentos com alunos de graduação e pós-graduação, aplicando avaliações quantitativas e qualitativas. Os experimentos seguiram uma base experimental similar, mas cada experimento teve sua particularidade na aplicação e na análise dos resultados. Os experimentos realizados mostram uma considerável evolução no aprendizado do assunto memória virtual. No primeiro experimento foi possível observar uma melhora 28,8% na quantidade de acertos. No segundo experimento foi possível comparar resultados de alunos que utilizaram o REA Amnesia com alunos que não tiveram nenhuma aula entre as avaliações. Os resultados do terceiro experimento mostram melhorias de até 180% na quantidade de acertos com a utilização do REA Amnesia, para alunos com dificuldades (demonstradas em suas notas), onde o uso do Amnesia se mostrou mais importante.

Palavras-chave: Memória Virtual, Recursos Educacionais Abertos, Ensino, Amnesia.

ABSTRACT

CACHO, C. E. A. **Desenvolvimento e utilização de recursos educacionais abertos para colaborar com o ensino de memória virtual**. 2015. 187 f. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação (ICMC/USP), São Carlos – SP.

The teaching of computer science involves many subjects that form the basis for an effective learning. The lack of adequate educational resources makes difficult to present such subjects clearly, due to the dynamics and complexity of them. The teaching of subjects related to the content of memory hierarchy and virtual memory are challenging, because both present distinct aspects, such as: structural, functional and performance. Many approaches have been studied to make the teaching of these aspects more attractive. One of these approaches considers the development and application of Open Educational Resources (OER). OER have been successfully applied to help the teaching and learning of challenging issues. Even knowing the possibility of using OER, many contents do not use such resources. This work presents the transformation of the Amnesia simulator in an OER to make easier the teaching and learning of the virtual memory subject, simulating structural, functional and performance aspects. The focus of this work is to improve the Virtual Memory teaching with the help of the OER Amnesia. We developed teaching materials to help teachers and students with class plans, tutorial of utilization and text with theoretical content, which are available along with the OER Amnesia. In order to verify the improvement in teaching and learning were conducted three experiments with undergraduate and graduate students, applying quantitative and qualitative evaluations. The experiments followed a similar experimental basis, but each experiment had its particularity in the application and analysis of results. The experiments show a considerable progress in the subject virtual memory learning among the students. In the first experiment, it was possible to observe a 28.8% of improvement in the quantity of hits by students. In the second experiment it was possible to compare results of students who used the OER Amnesia with students who had no class between the two tests applied. The results of the third experiment show improvements of up to 180% in the amount of hits, when using Amnesia, for those students with difficulties (demonstrated in their notes), where the use of Amnesia was more significant.

Key-words: Virtual memory, Open Educational Resources, Education, Amnesia.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS-FONTE

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADL ..... *Advanced Distributed Learning*
Alea. ..... Aleatória
CPU ..... *Central Processing Unit*
Cresc. .... Crescente
Decresc. .. Decrescente
DRAM ... *Dinamic Random Access Memory*
FIFO ..... *First-in First-out*
HMSE ... *Hierarchical Memory System Environment*
I/O ..... *Input/Output*
ICMC .... Instituto de Ciências Matemáticas e de Computação
ISA ..... *Institute for Advanced Studies*
LaSDPC .. Laboratório Sistemas Distribuídos e Programação Concorrente
LFU ..... *Least Frequetly Used*
LOM ..... *Learning Object Metadata*
LOR ..... *Learning Object Repository*
LRU ..... *Least Recently Used*
LTSC .... *Learning Technology Standards Committee*
MD ..... *Magnetic Disk*
MMU .... *Memory Management Unit*
MSE ..... *Memory System for Education*
MVC ..... *Model View Controller*
NRU ..... *Not Recently Used*
OA ..... Objetos de Aprendizagem
OC ..... Organização de Computadores
Pág. ..... Página
RAM ..... *Random Access Memory*
REA ..... Recursos Educacionais Abertos
ROA ..... Repositórios de Objetos de Aprendizagem
SC ..... *Second chance*
SCORM .. *Sharable Content Object Reference Model*
SMA ..... Sistemas Multiagentes

SO ..... Sistema Operacional
SOs ..... Sistemas Operacionais
SRAM ... *Static Random Access Memory*
STI ..... Sistema Tutor Inteligente
TLB ..... *Translation-Lookaside Buffer*
UC ..... unidade de controle
ULA ..... Unidade Lógica Aritmética
USP ..... Universidade de São Paulo
VPR ..... Repositórios Virtuais Privados

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

1.1 Contexto

Segundo as diretrizes curriculares nacionais para os cursos de graduação em Computação "não é um exagero dizer que a vida das pessoas depende de sistemas de computação e de profissionais que os mantêm, seja para dar segurança na estrada e no ar ou ajudar médicos a diagnosticar e tratar problemas de saúde, seja com um papel fundamental no desenvolvimento de novas drogas. Mais frequentemente, profissionais de computação estão trabalhando com especialistas de outras áreas, projetando e construindo sistemas de computação para os mais diversos aspectos da sociedade" (BARONE, 2012).

Garantir a qualidade de ensino superior em áreas tecnológicas como a computação é um desafio. Algumas tecnologias são efêmeras, nascendo e logo desaparecendo ao serem substituídas por outras melhores. Outras, por sua vez, são fundamentadas em conceitos mais profundos e alteram de maneira significativa as bases do conhecimento, alavancando novos desenvolvimentos.

Para qualificar profissionais de computação para o mercado de trabalho ou para a área acadêmica é necessário ensinar as bases da computação nos cursos de graduação. Isso requer que os conhecimentos teóricos e práticos sejam relacionados de modo que novos profissionais sejam formados com bases sólidas. Mas para a transmissão desses conhecimentos surgem alguns obstáculos, que devem ser enfrentados para a boa formação dos alunos.

Em Rocha *et al.* (2010) são apresentados alguns desses obstáculos, como a dificuldade por parte do professor, no que diz respeito à assimilação do conteúdo pelos alunos. Muitas vezes o ritmo dos alunos não acompanha o ritmo em que a matéria é conduzida, acarretando em dificuldades no aprendizado de conceitos fundamentais. Como consequência disso, pode ocorrer o desinteresse pelo conteúdo ministrado, ou até mesmo a aversão à disciplina.

Como descrito em Maia (2001), os recursos disponíveis para professores como giz,

lousa, papel e slides não são suficientes para a apresentação de conteúdos que necessitam de dinamismo, pois a apresentação do mesmo fica limitada a uma pequena sequência de eventos que tentam representar um grande número de situações. A falta de recursos apropriados torna difícil a apresentação do conteúdo de forma clara, devido à sua dinâmica e complexidade. Isso acaba limitando o aluno a apenas visualizar o que está sendo transmitido, sem muita interação com o conteúdo.

Exemplos desse problema podem ser encontrados em matérias como Sistemas Operacionais (SOs) e Organização de Computadores (OC), que são essenciais para todos os profissionais de computação. Hierarquia de memória e memória virtual são assuntos essenciais presentes em ambas matérias, SOs e OC, e estudados sob perspectivas distintas (TANENBAUM; BOS, 2014) (PATTERSON; HENNESSY, 2014). Enquanto OC apresenta os assuntos sob as perspectivas de hardware, SOs considera o ponto de vista de software.

O ensino da hierarquia de memória é um desafio, uma vez que aborda a relação de diferentes níveis de memória, que são baseados em princípios de localidade espacial e temporal (PATTERSON; HENNESSY, 2014). As memórias menores, mais rápidas e mais caras são colocadas próximas da CPU para manter os dados frequentemente referenciados. Por outro lado, as memórias caracterizadas com atrasos superiores, maior capacidade de armazenamento e mais baratas são colocados longe da CPU para dar suporte aos dados não frequentemente referenciados. Cada memória tem o seu modo particular de funcionamento e interação com as outras, o que provê um cenário cheio de detalhes correlacionados. Os níveis de hierarquia de memória ensinados são registradores, memórias cache, principal (RAM) e secundária (Disco).

O Sistema Operacional (SO) gerencia a memória lógica e oferece proteção, realocação e permite a execução de processos maiores do que a memória principal disponível (TANENBAUM; BOS, 2014). Esta gerência é baseada na abstração conhecida como memória virtual. A memória virtual oferece a cada processo o seu próprio espaço de endereçamento, dividido em páginas (uma sequência contígua de endereços) que são armazenadas na memória principal ou no disco. Para um ensino adequado dessa abstração, os seguintes assuntos devem ser compreendidos pelos estudantes: página virtual, quadro de página, tabela de página, *Memory Management Unit* (MMU), endereço virtual, endereço físico, falhas página, *Translation-Lookaside Buffer* (TLB), tabela de páginas multinível, tabela de página invertida, algoritmo de substituição de página, segmentação, segmentação paginada, entre outros.

1.2 Motivação

Muitos estudantes consideram os assuntos relacionados à memória virtual complexos e difícil de assimilar (DICKINSON, 2000). Muitas abordagens vêm sendo estudadas para tornar as disciplinas (por exemplo SOs e OC) mais atrativas para que os alunos tenham uma formação sólida, dentro do prazo estipulado pelos planos pedagógicos dos seus cursos.

Tornar o ensino mais atrativo em conteúdos que necessitem de dinamismo na apresentação e interação com o aluno, acaba tornando-se um desafio, que pode ser atacado com o desenvolvimento de ferramentas de ensino (HAMAWAKI; PELEGRINI, 2009). Para seguir nesta linha, o desenvolvimento de ferramentas precisa ser cauteloso para que o custo seja minimizado e que o conteúdo seja abordado de forma abrangente e profunda. Duas alternativas amplamente investigadas atualmente nesse contexto são Objetos de Aprendizagem (OA) (LOM WG12, 2009) e Recursos Educacionais Abertos (REA) (UNESCO; Commonwealth of Learning, 2011).

Os OA representam um artefato de ensino que pode ser reutilizável posteriormente. O *Learning Objects Metadata Workgroup* define os OA como uma "entidade" desenvolvida com as tecnologias disponíveis, digital ou não, capaz de ser usada, reusada ou referenciada durante a aprendizagem (LOM WG12, 2009). Uma das definições de OA determina que os objetos são: "qualquer recurso digital que possa ser reutilizado no suporte ao ensino" (WILEY, 2003).

REA representam uma evolução dos OA e são vistos como materiais de ensino, aprendizado e pesquisa em qualquer mídia (impressa ou digital) que estão sob domínio público ou licenciados de maneira aberta, permitindo a utilização e adaptação por terceiros. REA podem incluir cursos completos, partes de cursos, módulos, livros didáticos, artigos de pesquisa, vídeos, *software*, entre outros materiais que possam ser utilizados no ensino (UNESCO; Commonwealth of Learning, 2011).

Mesmo com a abordagem de OA e REA, a disponibilização de conteúdos para alguns ambientes de ensino ainda é escassa, como é o caso de OA para o ensino de memória virtual. Algumas ferramentas de ensino tentam suprir essa falta de objetos, mas a grande maioria das ferramentas encontradas tem código fechado e não aborda todas as minúcias do conteúdo de memória virtual.

O projeto Amnesia possui um simulador de mesmo nome e visa complementar o ensino do conteúdo de hierarquia de memória e memória virtual de uma forma interativa e dinâmica. O simulador está em desenvolvimento desde 2007, por alunos de graduação e de mestrado do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP). Desde 2013 estão sendo investigadas novas características de OA e REA para incorporação no simulador. O simulador Amnesia está organizado em três módulos principais, que são módulo CPU, cache e memória virtual e considera os registradores internos da CPU, caches multinível, memória principal, disco e a abstração da memória virtual paginada (OLIVEIRA *et al.*, 2008).

1.3 Objetivos

O objetivo principal deste trabalho é colaborar com o ensino de computação, com o desenvolvimento e a utilização de OA e REA. Mais especificamente, espera-se melhorar o ensino de memória virtual nos cursos de computação utilizando o Amnesia. Para isso, é desenvolvido o REA Amnesia para auxiliar o professor no ensino de memória virtual, procurando prover assim

melhorias na qualidade de ensino deste conteúdo.

Outros objetivos desse trabalho são corrigir o máximo de defeitos encontrados no simulador, para torná-lo robusto e transformar o simulador Amnesia em um REA. Além disso são desenvolvidos planos de aula e tutorial de utilização do simulador e texto com conteúdo teórico para que o simulador possa ser utilizado mais facilmente pelos alunos. Todos os materiais produzidos estão disponibilizados para que possam ser utilizados em outros contextos de ensino.

Para verificar as melhorias no aprendizado dos alunos providas pelo REA Amnesia é realizada uma avaliação quantitativa. Essa avaliação consiste na aplicação de dois testes, antes e depois da utilização do Amnesia, para observar a evolução no aprendizado. Além disso, é realizada uma avaliação qualitativa, para verificar a opinião dos alunos referente ao Amnesia.

É importante observar que a utilização do Amnesia não visa substituir o professor. A ideia é que o Amnesia seja utilizado como material de apoio ao aprendizado do aluno, tendo em vista uma maior participação do mesmo nas aulas ministradas. Os alunos podem adquirir os materiais para reforçar o conhecimento adquiridos nas aulas de memória virtual.

1.4 Organização da dissertação

Esta dissertação está dividida da seguinte forma:

- No Capítulo 2 são apresentados os conceitos de OA, abordando assuntos como padrões, repositórios, avaliação de OA e conceitos de REA. No final desse capítulo são apresentados alguns OA que são utilizados no ensino de computação.

- No Capítulo 3 são apresentados os conceitos da arquitetura de von Neumann e da hierarquia de memória, além de descrever as memórias que compõem essa hierarquia: registradores, memória cache, memória principal e memórias secundárias.

- No Capítulo 4 é apresentada a gerência de memória, abordando os conceitos de alocação de processo e partições de memória. Em seguida é apresentada a memória virtual abordando os conceitos de paginação e TLB.

- No Capítulo 5 é apresentado o simulador Amnesia que é o simulador utilizado neste trabalho. Em seguida, são apresentados simuladores de hierarquia de memória e memória virtual encontrados na literatura. No final é realizada uma comparação entre os simuladores.

- No Capítulo 6 são apresentadas as implementações e readequações de funcionalidades realizadas no Amnesia, necessárias antes da incorporação de características de OA e REA no Amnesia.

- No Capítulo 7 são apresentados os experimentos realizados como forma de avaliação do Amnesia. São apresentados os resultados das avaliações quantitativas e qualitativas realizadas em três experimentos com alunos.
- No Capítulo 8 são realizadas as conclusões do trabalho, ressaltando os resultados obtidos, destacando as contribuições deste trabalho e apresentado os trabalhos futuros do projeto.

CAPÍTULO

2

OBJETOS DE APRENDIZAGEM E RECURSOS EDUCACIONAIS ABERTOS

2.1 Considerações inicias

Nesse capítulo são abordados conceitos e características dos OA. São apresentados alguns dos conceitos recorrentes na literatura, como padrões, repositórios, avaliações de OA e a apresentação dos conceitos de REA. Em seguida são apresentados alguns OA encontrados na literatura para diferentes tipos de ensino e para a computação.

2.2 Conceitos e características

Os OA não possuem uma definição universalmente aceita, mas uma das definições mais utilizadas, porposta por *Learning Technology Standards Committee* (LTSC), apresenta-os como: "Qualquer entidade, digital ou não digital, que possa ser utilizada, reutilizada ou referenciada durante o aprendizado suportado por tecnologias" (LTSC, 2002).

Segundo Sosteric e Hesemeier (2002) a definição feita por LTSC é muito abrangente e acaba englobando "tudo", como por exemplo um computador ou um teclado. Para ter-se uma definição mais restritiva, esse trabalho de 2002 apresenta a seguinte definição: "Um objeto de aprendizagem é um arquivo digital (imagem, filme, etc.) destinado e utilizado em fins pedagógicos, o que inclui internamente ou via associação, sugestões sobre o contexto adequado para sua utilização"(SOSTERIC; HESEMEIER, 2002).

Wiley (2000), afirma que um objeto de aprendizagem é qualquer recurso digital que pode ser reutilizado como apoio à aprendizagem. Tarouco, Fabre e Tamusiunas (2003), por sua vez, apresentam a ideia básica que objetos aprendizagem são como blocos com os quais é construído o contexto da aprendizagem.

Como descrito em Barbosa (2014), além das diversas definições, existe uma divergência em relação à terminologia usada nos OA. Algumas das terminologias encontradas na literatura são objetos ativos, objetos educacionais, recursos de aprendizagem, objetos de informações reutilizáveis, objetos de aprendizagem reutilizáveis, entre outras.

Segundo Galafassi, Gluz e Galafassi (2013), a definição de OA não é unânime porque eles são considerados uma tecnologia recente. O mesmo se aplica à divergência em relação aos demais termos citados. Apesar disso, existe um consenso entre os vários trabalhos disponíveis na literatura, os quais determinam que a reutilização é fundamental para compreender o significado de um objeto de ensino.

Um objeto de aprendizagem deve possuir um objetivo muito bem definido, para proporcionar a aprendizagem do conteúdo e a evolução do raciocínio de quem o utiliza, além de ser facilmente reutilizado em outros contextos de aprendizagem.

Segundo Longmire (2000), a abordagem de OA deve satisfazer as necessidades de aprendizagem imediatas e as necessidades futuras baseadas no curso que o Objeto se insere. Há vários argumentos para a concepção e desenvolvimento de OA com foco na reutilização, incluindo o seguinte:

- Flexibilidade: os OA que foram desenvolvidos para serem utilizados em vários contextos, podem ser reutilizados mais facilmente do que o OA que têm de ser reescrito para cada novo contexto.

- Facilidade de atualizações, pesquisas e gerenciamento de conteúdo: os OA devem seguir um padrão (Seção 2.3) para facilitar a atualização, busca e gestão de conteúdo, filtrando e selecionando apenas o conteúdo relevante para um determinado fim.

- Customização: os OA devem proporcionar flexibilidade e também uma fácil customização. Como os objetos são independentes, a ideia de utilização dos mesmos em diversos contextos é estimulada e esperada. Cada contexto diferente pode utilizar os objetos e arranjá-los da maneira que lhe for conveniente, montando seus próprios conteúdos programáticos para a utilização do objeto.

- Interoperabilidade: a ideia que um objeto de aprendizado possa ser criado e utilizado por diferentes plataformas de ensino e por diferentes pessoas, aumenta ainda mais as vantagens dos objetos aprendizagem. Dessa forma, são considerados padrões para o armazenamento dos OA, proporcionando a reutilização, não apenas em cursos destinados inicialmente para aqueles objetos, mas também para outros cursos em que se deseje ou necessite utilizar objetos.

- Facilidade na aprendizagem baseada em competências: a reutilização dos OA em diversos contextos e diversas vezes ao longo do tempo de vida do objeto faz com que, de maneira

espontânea, ele receba e transmita novos conhecimentos, provocando assim uma melhora significativa na qualidade do ensino. Isso é considerada uma vantagem da utilização de OA.

- Aprimoramento do conteúdo: o valor do conteúdo de um Objeto de Aprendizagem é melhorado toda vez que for reutilizado, pois outros usuários podem aprimorar o conteúdo disponibilizado. Isso não só se reflete no conteúdo inicial, mas também em conteúdo de colaboradores, evitando assim, gastos de tempo de *re-design* e desenvolvimento, na possibilidade de fornecê-los a parceiros em diferentes contextos de ensino .

Todos esses argumentos mostram que a utilização de OAs vem para facilitar e melhorar a qualidade do conteúdo disponível para os mais diferentes tipos de ensino, proporcionando a professores e alunos diversas ferramentas facilitadoras.

Os OA podem ser classificados de acordo com suas características. Segundo Longmire (2000) as quatro categorias de classificação que descrevem um OA são:

- Objetos instrucionais: artigos de workshops/seminários e slides de aulas;
- Objetos de colaboração: exercícios monitorados (individuais e em grupo), salas de discussão;
- Objetos de prática: ferramentas de simulações de hardware e software;
- Objetos de avaliação: provas e testes.

As características dos OA são definidas a partir dos metadados[1] de um padrão de objeto de aprendizagem. Alguns padrões de OA são definidos na próxima Seção.

2.3 Padrões

Os OA necessitam realizar uma descrição do seu conteúdo para que esses objetos sejam localizados com maior facilidade e reutilizados em diversos contextos de ensino, para isso são utilizados os metadados. Para que essa descrição seja realizada é necessária a adoção de um padrão de OA.

Uma descrição mais formal de metadados caracteriza-os como informações sobre um determinado conteúdo (dados base). Sendo assim, metadados são dados que descrevem completamente os dados base que representam, permitindo ao usuário decidir sobre a utilização desses dados da melhor forma possível (GALAFASSI; GLUZ; GALAFASSI, 2013).

[1] "Metadados são informações que acrescem aos dados e que têm como objetivo informar-nos sobre eles para tornar mais fácil a sua organização"(METADADOS, 2014).

Alguns padrões de OA são basicamente um conjunto de metadados que visam tornar um objeto de aprendizagem reutilizável por diferentes usuários e em diferentes contextos, além de torná-lo acessível onde é feita a catalogação do objeto para ser disponibilizado em repositórios.

Segundo LTSC (2001), os padrões de objetos aprendizagem devem incidir sobre um conjunto mínimo de atributos necessários para permitir que tais objetos possam ser gerenciados, localizados e avaliados. Nesta Seção são apresentadas as características dos padrões de OA que têm uma maior influência nos trabalhos encontrados na literatura. Os padrões apresentados são: *Learning Object Metadata* (LOM) e o *Sharable Content Object Reference Model* (SCORM).

Outros padrões conhecidos são: IMS-Learning Design (IMS Global Learning Consortium, 2008), Ariadne (EPFL; KUL; PROJECT, 1999), Dublin Core (KUNZE; BAKER, 2007), CanCore (FRIESEN; INITIATIVE, 2005).

2.3.1 Learning Object Metadata (LOM)

LOM é um padrão de OA desenvolvido e mantido pela LTSC desde 1997 e descrito em (LTSC, 2002). Este padrão oferece instâncias de metadados para os autores de objeto de aprendizagem e descreve características relevantes do objeto.

Alguns dos metadados do padrão descrevem as principais características dos OA como: nome do objeto, tamanho, localização, conhecimento necessário, tipo do objeto, autor, atributos pedagógicos, termos de distribuição, entre outros. Casali *et al.* (2013) afirmam que o propósito dos metadados em OA é apoiar a reutilização desses objetos para ajudar a descoberta e para facilitar a sua interoperabilidade.

Segundo LTSC (2001), os objetivos do padrão LOM são:

- Proporcionar aos alunos e/ou instrutores facilidade na pesquisa, avaliação, aquisição e utilização de OA.
- Habilitar o compartilhamento e a troca de objetos entre os sistemas de aprendizagem.
- Permitir o desenvolvimento de OA em unidades que possam ser combinadas de maneira significativa.
- Permitir que ferramentas de ensino possam de forma automática e dinâmica compor lições personalizadas aos alunos de forma individual.
- Permitir que vários OA possam trabalhar juntos dentro de um ambiente de aprendizagem, para que cada objeto possa complementar o conteúdo um do outro.
- Requisitar a documentação do desenvolvimento do objetivo de aprendizagem existente ou de novos OA agregados.

- Permitir que os OA ofereçam o suporte a todas as formas de distribuição de aprendizagem: sem fins lucrativos e com fins lucrativos.
- Capacitar ensino, formação e organizações do aprendizado, tanto do setor público quanto do privado, para expressar o conteúdo educacional e padrões de desempenho em um formato padronizado e independente do conteúdo propriamente dito.
- Fornecer aos pesquisadores padrões que suportem a coleta e o compartilhamento de dados, sobre a aplicabilidade e a eficiência de OAs.
- Definir um padrão que seja simples, mas extensível a vários domínios e competências, de modo a ser facilmente e amplamente adotado em aplicações.
- Prover suporte à segurança e autenticação necessária para a distribuição e o uso de OA.

O esquema LOM é definido como uma hierarquia de metadados, como pode ser visto na Figura 1. A hierarquia inclui categorias de metadados e metadados simples. Na base de dados do esquema LOM só os nós folha têm valores individuais a serem definidos através de seu espaço de valor e tipo de dados. As categorias de metadados do esquema LOM não têm valores individuais. Consequentemente, eles não têm espaço de valor ou tipo de dados para ser preenchido.

O esquema LOM descreve os metadados oferecidos para os OA. Os metadados agregados e simples também podem ser visualizadas na Figura 1.

Existem nove categorias de metadados no esquema do padrão LOM para definição dos OAs: *General*, *Lifecycle*, *Meta-metadata*, *Technical*, *Educational*, *Rights*, *Relation*, *Annotation* e *Classification*. Estas categorias são descritas a seguir:

- *General*: categoria de informações gerais que descreve o objeto de aprendizagem como um todo.
- *Lifecycle*: categoria de ciclo de vida dos recursos, relacionada com a história e o estado atual do objeto de aprendizagem, além daqueles que afetam este objeto de aprendizagem durante a sua evolução.
- *Meta-Metadata*: categoria de informação sobre a própria instância de metadados.
- *Technical*: categoria referente aos requisitos técnicos e características técnicas do objeto de aprendizagem.
- *Educational*: categoria referente às características educacionais e pedagógicas do objeto de aprendizagem.
- *Rights*: categoria dos direitos de propriedade intelectual e condições de uso para o objeto de aprendizagem.

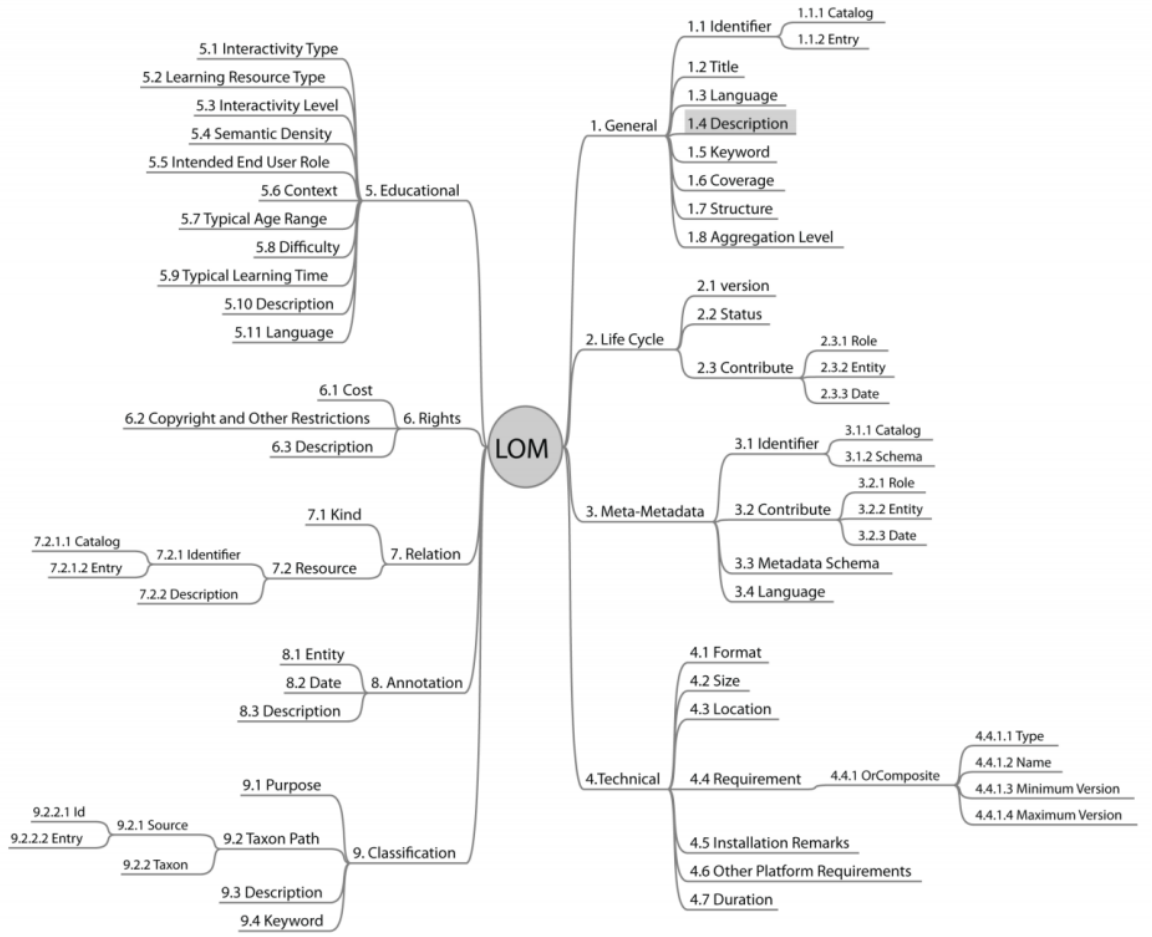


Figura 1 – Representação do esquema hierárquico de metadados do padrão LOM (BARKER; CAMPBELL, 2010)

- *Relation*: categoria das características que definem a relação entre o objeto de aprendizagem e outros objetos relacionados.

- *Annotation*: categoria que fornece comentários sobre o uso educacional do objeto de aprendizagem, além de informações sobre quando e por quem os comentários foram criados.

- *Classification*: categoria que descreve o objeto de aprendizagem em relação a um sistema de classificação específico.

Essas categorias de metadados formam a base do esquema LOM. Em Ferlin *et al.* (2010) foi realizada uma comparação entre o padrão LOM e outros padrões, descritos no final da Seção anterior. Foi constatado que nove metadados do padrão LOM estão presentes em todos os padrões. As categorias que contêm esses metadados são: *General*, *Lifecycle*, *Technical* e *Educational*. Esta comparação reflete o quanto esses elementos são importantes para a catalogação de um objeto de aprendizagem.

2.3.2 Sharable Content Object Reference Model (SCORM)

O padrão de objeto de aprendizagem SCORM é uma compilação de especificações técnicas para OA. O padrão SCORM foi criado e é mantido pela *Advanced Distributed Learning* (ADL), um grupo de pesquisa patrocinado pelo departamento de defesa dos Estados Unidos (ADL, 2011).

Segundo a ADL (2011) o SCORM integra um conjunto de normas técnicas, especificações e orientações destinadas a atender aos requisitos no nível de acessibilidade, reutilização, interoperabilidade, durabilidade de conteúdo e sistemas.

A Figura 2 apresenta o conjunto de especificações do SCORM 1.2, que é divido em três seções: visão geral (*The SCORM Overview*), modelo de agregação de conteúdo (*The SCORM Content Aggregation Model*) e ambiente de execução (*The SCORM Runtime Environment*).

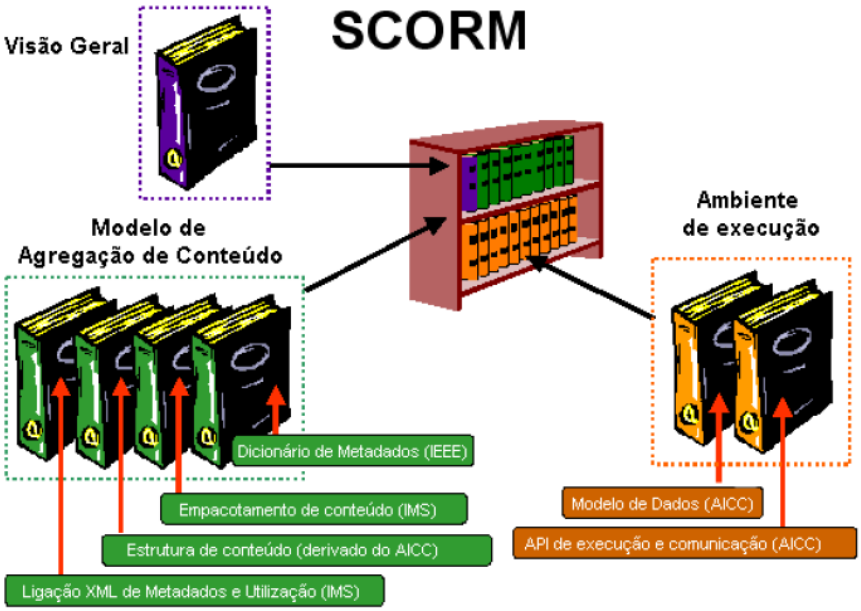


Figura 2 – Conjunto de especificação SCORM 1.2 (FABRE; TAROUCO; TAMUSIUNAS, 2013)

Segundo Rustici (2009), a seção "Ambiente de Execução" especifica como o conteúdo do objeto de aprendizagem deve ser descrito para ser colocado em um repositório. O "Modelo de Agregação de Conteúdo" tem a especificação de como deve ser empacotado o conteúdo para que ele possa ser importado para o repositório. Isso envolve a criação de arquivos XML que descrevam o conteúdo do objeto de aprendizagem. Um objeto empacotado é constituído pelos arquivos de recursos do objeto e o arquivo *imsmanifest.xml*, comprimidos em um arquivo do tipo *zip*.

Segundo Fabre, Tarouco e Tamusiunas (2013), o SCORM define duas categorias de metadados: metadados do objeto e metadados do estudante. A primeira categoria, descrita no Modelo de Agregação de Conteúdo do SCORM, descreve metadados de conteúdo e recursos

do objeto, que fornecem meios para busca e recuperação de objetos em repositórios. A segunda categoria, descrita na especificação do Ambiente de Execução do SCORM, é utilizada para rastrear os perfis e o desempenho dos estudantes.

2.4 Repositórios de objetos de aprendizagem

Repositórios, de maneira geral, são bases de dados *online* que reúnem de maneira organizada arquivos de diversos formatos, que resultam em uma série de benefícios tanto para os pesquisadores quanto às instituições ou para sociedade. Alguns repositórios proporcionam maior visibilidade aos resultados de pesquisas e possibilitam a preservação da memória científica de sua instituição.

Os Repositórios de Objetos de Aprendizagem (ROA) ou *Learning Object Repository* (LOR) fornecem uma interface através da qual os OA são armazenados e acessados de forma segura e confiável, para a manutenção e preservação do acervo digital. Assim, podem ser distribuídos para professores e alunos que queriam utilizar os OA.

Alguns LORs exigem que os OA sejam acompanhados de metadados, os quais são utilizados para descrever os OA com informações como: nome do objeto, autor, características pedagógicas, entre outras. Os metadados podem ser adquiridos com aplicação de alguns dados padrões (apresentados na Seção 2.3) ou são oferecidos pelo próprio repositório.

Em Neven e Duval (2002) é realizado um estudo exploratório de LORs existentes, onde são apontadas as diferenças entre LORs, quais são as exigências de cada LOR, entre outros fatores. Alguns dos repositórios existentes e apresentados no estudo são:

- ARIADNE[2] é um LOR que vem sendo desenvolvido desde 1996. Ele é uma rede hierarquizada de nós replicados, onde os OAs e os metadados dos objetos são replicados juntamente para todos os nós da rede. O LOR exige que os metadados sejam providos no padrão LOM.

- SMETE[3] é um LOR de apoio à pesquisa de recursos para a área de exatas nos cursos de matemática, computação, engenharias, entre outros. O usuário pode criar um perfil pessoal que dá acesso a um espaço de trabalho chamado bookmarks. Os OA são recomendados com base nos acessos anteriores de um usuário ao LOR. Membros com interesses semelhantes podem ser identificados.

- Lydia[4] é uma rede comercial de OA, em que permite os usuários a procurar por OAs desde que tenham se registrado no site. Os metadados de cada objeto de aprendizagem

[2] Alliance of Remote Instructional Authoring and Distribution Networks for Europe - <http://www.ariadne-eu.org/>

[3] science, math, engineering and technology education - <http://www.smete.org/smete/>

[4] <http://www.lydialearn.com/>

descrevem as propriedades e o valor a ser pago pelo objeto. Organizações podem criar seus próprios Repositórios Virtuais Privados (VPR). Um VPR permite que as organizações possam introduzir conteúdo privado e oferece acesso livre a todo o conteúdo disponível no servidor central.

- MERLOT II[5] é um LOR criado e mantido pela *California State University*. O LOR é descrito como uma comunidade *on-line* de materiais de aprendizagem. O serviço foi desenvolvido pelo corpo docente, sendo utilizado pela comunidade internacional de educação. A coleção de Merlot é composta por mais de 40.000 objetos classificados em 19 categorias, que incluem: estudo de caso, ePortfólio, LOR, curso online, ferramenta de rede social, animação, ferramenta de avaliação, entre outras.

- LABVIRT[6] é um LOR da Universidade de São Paulo - USP, coordenada pela Faculdade de Educação. Nesse repositório são disponibilizadas simulações feitas pela equipe do LabVirt, em relação a Física e Química para alunos de ensino médio das escolas da rede pública.

- BIOE[7] é um repositório criado em 2008 pelo Ministério da Educação, em parceria com o Ministério da Ciência e Tecnologia, a Rede Latino-Americana de Portais Educacionais, a Organização dos Estados Ibero-Americanos, entre outras parcerias. Esse repositório internacional tem o propósito de manter e compartilhar recursos educacionais digitais de livre acesso, sem distinção de área de ensino. Os objetos são classificados em oito categorias, que são: animações/simulações, áudios, experimentos práticos, hipertextos, imagens, mapas, *softwares* educacionais e vídeos.

Embora bastante difundidos e usados na literatura, alguns estudos vêm mudando esse quadro de que os LORs são os únicos locais para o armazenamento de OA. Em Nash (2005) são apresentadas algumas alternativas para a localização de OA, como *wikipedia*, *weblogs*, *podcasts*, entre outros. Em Buzzetto-More (2013) é utilizado o YouTube como plataforma para distribuição de OA em forma de vídeo aula.

2.5 Avaliação

Segundo Williams (2000), a avaliação é parte integrante da concepção de um objeto de aprendizagem. Ela ajuda a identificar as necessidades presentes nos objetos, considerando alternativas para tratar essas necessidades (incluindo outros OA), como o desenvolvimento de protótipos, além da produção e entrega de material pedagógico extra. A importância da avaliação do objeto de aprendizagem é apresentada em muitos trabalhos, como em Rocha e Campos (1993) e Morais (2003). Nesses trabalhos existe um consenso que os docentes devem dedicar uma

[5] *Multimedia Educational Resource for Learning and Online Teaching* - <http://www.merlot.org/>
[6] Laboratório Didático Virtual - <http://www.labvirt.fe.usp.br/>
[7] Banco Internacional de Objetos Educacionais - <http://objetoseducacionais2.mec.gov.br/>

atenção criteriosa à avaliação dos OA para garantir o melhor aproveitamento dos mesmos nas práticas pedagógicas.

Em Almeida, Chaves e Coutinho (2010) e Santos e Amaral (2012), são realizadas avaliações de OA, sendo que essas avaliações apresentam os principais problemas presentes nesses objetos. Os problemas apresentados são: qualidade do conteúdo, falta de aprofundamento da temática (abordagem superficial), a ausência de contextualização e a falta de utilização de exemplos, analogias e situações lúdicas que possam facilitar a assimilação e fixação do conteúdo trabalhado.

Existem alguns métodos de avaliação de OA, como a proposta de uma Técnica de Inspeção de Conformidade Ergonômica de Software Educacional por Cybis *et al.* (1998), que apresenta um conjunto de questões que visam orientar os avaliadores na tarefa de inspecionar as qualidades pedagógicas do objeto de aprendizagem. Esse método apresenta três módulos, que são:

- Módulo de Classificação: o objetivo desse módulo é classificar o objeto de aprendizagem quanto à sua modalidade, abordagem pedagógica e habilidades cognitivas exigidas dos alunos.
- Módulo de Avaliação: o objetivo desse módulo é avaliar a capacidade do material em relação ao auxílio do aprendizado específico através de uma inspeção de conformidade do objeto de aprendizagem, segundo um conjunto de padrões ergonômicos e pedagógicos.
- Módulo de Contextualização: tem como objetivo auxiliar no processo de tomada de decisão sobre a aquisição, mediante a adequação do objeto de aprendizagem ao contexto específico do ensino.

Outro método de avaliação é LORI, proposto por (NESBIT; BELFER; LEACOCK, 2002). Esse método avalia um objeto, referente aos seguintes itens: qualidade do conteúdo, adaptação ao aluno, acessibilidade, reutilização, entre outros fatores. Os avaliadores do objeto de aprendizagem estipulam notas de 1 a 5 para cada item e no final é determinada uma média de cada item para apresentar as qualidades e o que precisa ser melhorado no objeto.

Em GAMA (2007) é proposto um método de avaliação onde são aplicados questionários aos alunos, ao professor e ao desenvolvedor. As avaliações são focadas nas expectativas e reação dos alunos com o uso do objeto, além de avaliar usabilidade e características pedagógicas do objeto. Os Questionários propostos por GAMA (2007) são apresentados no Anexo A.

2.6 Recursos Educacionais Abertos (REA)

"Abertura é o único futuro para Objetos de Aprendizagem" (WILEY, 2007). Partindo desse ideal foi desenvolvido o termo REA que engloba os conceitos de OA e abertura.

O conceito de abertura é referente ao licenciamento do conteúdo. Isso significa que o detentor do direito autoral daquele objeto decidiu compartilhar com a sociedade parte de seus direitos de autor, como os direitos de cópia, reprodução, redistribuição, recombinação, entre outros.

Segundo Wiley (2010), a abertura de REA deve fazer com que recursos de ensino e aprendizagem sejam fornecidos gratuitamente sob uma licença que concede ao usuário uma permissão para se envolver em atividades chamada de "4R",que são:

- Reutilizar: direito de utilizar o conteúdo em sua forma inalterada ou modificada.
- Revisar: direito de adaptar, ajustar, modificar ou alterar o próprio conteúdo.
- Remixar: direito de combinar o conteúdo original ou revisado com outro conteúdo para criar algo novo.
- Redistribuir: direito de compartilhar cópias do conteúdo original, das revisões ou as combinações com outros conteúdos.

Para realizar o licenciamento de conteúdo e a proteção dos direitos autorais é necessária a inclusão de licenças de *copyright* nos conteúdos que serão disponibilizados. As licenças e *copyright* mais utilizadas nos REA são:

- *Creative Commons*[8] (CC): compõem um conjunto de licenças de *copyright*, publicadas pela organização sem fins lucrativos chamada de *Creative Commons*, fundada em 2001. As obras que podem incluir a licença CC compreendem livros, peças, filmes, músicas, artigos, entre outras.
- *General Public License*[9] (GPL): Modelo de licença com maior utilização por parte de projetos de software livre, devido à sua adoção para o projeto GNU[10]. A licença é indicada exclusivamente para *software*. A GPL baseia-se em nas liberdades de execução do programa, de estudo do código, de redistribuição de cópias e do aperfeiçoamento de programas.
- GNU *Free Documentation License*[11] (GFDL): Uma variação da GPL para ser utilizada com documentos. A GFDL permite que textos, apresentações, conteúdo de páginas WEB, entre outros documentos, sejam distribuídos e reaproveitados, mantendo os direitos autorais e restringindo que essa informação seja usada de maneira indevida.

8 <http://creativecommons.org/>
9 <http://www.gnu.org/licenses/licenses.html#GPL>
10 GNU é um SO distribuído de forma livre
11 <http://www.gnu.org/licenses/fdl-1.3.html>

Segundo Hylén (2006) a definição mais usada atualmente de REA é "Recursos Educacionais Abertos são materiais digitalizados oferecidos de forma livre e abertamente para educadores, estudantes e autodidatas para o uso e reuso no ensino, no aprendizado e na pesquisa." Ainda nesse trabalho são apresentados três desafios presentes em REA, que são: conscientização sobre questões dos direitos de autorais, garantias de qualidade dos REA e sustentabilidade das iniciativas de disponibilização REA.

Inciativas de disponibilização de REA são plataformas onde alunos e professores podem adquirir REA para as mais distintas áreas. Em Barbosa (2014), são apresentadas algumas plataformas de sucesso no cenário mundial em que podem ser encontrados REA. Algumas delas são: *OpenLearn*[12] da *Open University*, *ConneXions* da RiceUniversity (Burrus, 2010), a *Open Learning Initiative*[13] da *Carnegie Mellon University* (STRADER; THILLE, 2012), entre outras. No Brasil, dentre as iniciativas de destaque, temos o Projeto REA-Brasil[14] e o Portal Educação[15].

Ainda que várias iniciativas de disponibilização de recursos educacionais digitais visem ao compartilhamento de informações e conhecimento que não estejam configuradas inteiramente como REA, elas estão muito próximas do cerne do movimento (SANTOS, 2013). Os ROA (Apresentados na Seção 2.4) podem ser considerados iniciativas de distribuição de recursos educacionais.

Existe um grande esforço para que REA sejam empregados no ensino superior. A UNESCO, por exemplo, disponibilizou um texto com a finalidade de incentivar governos e instituições a investirem em produção, adaptação e uso de REA, com a finalidade de melhorar o ensino e consequentemente reduzir custos (UNESCO; Commonwealth of Learning, 2011).

2.7 Objetos de aprendizagem na computação

Existem objetos de aprendizagem que podem ser utilizados no ensino fundamental (BARBOSA; FERNANDES, 2010) e no ensino médio (JUNIOR; FREITAS, 2013) (ARANTES; MIRANDA; STUDART, 2010). Também há muitos OA que podem ser aplicados no ensino superior (GONZáLEZ; ANJOS; CUNDA, 2014).

Em Alharbi, Henskens e Hannaford (2011) são apresentadas as áreas mais abordadas pelos OA contidos no repositório MERLOT II. Algumas áreas abordadas são: Agricultura, Astronomia, Biologia, Física e Computação. Mas o foco do trabalho de 2011 está em explorar os OA para a área de computação. Esse estudo cita a existência de 750 OA para a área de computação no repositório.

Nesta Seção são apresentados alguns OA que foram encontrados na literatura e que

12 <http://www.open.edu/openlearn>
13 <http://oli.cmu.edu/>
14 <http://rea.net.br/site>
15 <http://educacaoaberta.org>

podem ser utilizados nos diversos cursos de tecnologias e outros contextos. Os objetos são:

- FAdo[16] (MOREIRA, 2005) é um objeto de aprendizagem que possui um ambiente interativo para a manipulação simbólica de linguagens formais. FAdo permite a edição e a visualização de autômatos finitos e fornece uma interface ao usuário para alguns algoritmos de conversão e reconhecimento de expressões regulares.

- REDBOOL (FOURNIER-VIGER; NAJJAR, 2006) é um objeto de aprendizagem para o ensino de redução booleana. Objeto é referente ao assunto de expressões algébricas booleanas e suas simplificações, por meio de regras de redução.

- *Live Programming* (WU *et al.*, 2011) é um objeto de aprendizagem para o ensino de programação que visa melhorar a base introdutória no ensino de programação de computadores. O objeto de aprendizagem emprega um modelo de aprendizagem, chamado *Interactive Learning Model*, que fornece visualização e interatividade na aprendizagem de uma percepção e da prática integrada. A plataforma de aprendizagem é baseada em nuvem.

- Lógica de programação (JESUS *et al.*, 2007) é um objeto de aprendizagem para o ensino de lógica de programação com uso de "português estruturado"[17] . Os assuntos abordados no objeto de aprendizagem são: algoritmos, variáveis, operadores, estrutura de seleção e estrutura de repetição. O ambiente de criação do curso permite a inclusão de objetos interativos como animações, textos e imagens.

- VisualED (COSTA, 2011) é um objeto de aprendizagem que auxilia no ensino da linguagem Pascal com foco na disciplina de Estruturas de Dados. O professor constrói um roteiro por meio do qual o objeto gera ilustrações gráficas que orientam o aluno a visualizar o funcionamento das estruturas de dados. Os temas abordados pelo objeto são: vetores, lista encadeada, fila, pilha, grafos, entre outros.

Existem outros OA, que não têm descrição em trabalhos científicos, mas estão disponibilizados em repositórios e podem ser utilizados (PEREIRA *et al.*, 2010). Existem outras ferramentas de ensino que não são classificadas como OA, mas poderiam ser descritas com tais (CARBONE; LL, 1998).

2.8 Considerações finais

Transformar materiais de ensino (*slides*, textos, simuladores, entre outros) em OA e disponibilizá-los em ROA é uma contribuição para a educação. Os materiais disponibilizados em

[16] É um acrônimo de Finite Automata devoted oracle - <http://fado.dcc.fc.up.pt/>
[17] Uma pseudolinguagem em português, conhecida também como Portugol.

ROA, podem ser localizados com maior facilidade e utilizados por professores e alunos que têm interesse nos assuntos abordados pelo material, realizando uma reutilização do conhecimento contido nesse material. O uso e reuso de tais materiais polarizam uma constante avaliação e evolução, criando-se um ciclo virtuoso de melhoria constante do conhecimento.

A estrutura envolvida no conceito de OA torna esse conceito concreto e robusto. Para que um material de ensino possa ser transformado em um objeto de aprendizagem é necessário aplicar um padrão e disponibilizar em repositório, para que reutilização desse OA seja efetiva.

Tornar o simulador Amnesia um REA e disponibilizá-lo em um repositório, faz com que a utilização do Amnesia não fique limitada a poucas pessoas e instituições. A reutilização do Amnesia provê uma grande contribuição para ensino de hierarquia de memória e memória virtual.

CAPÍTULO

3

HIERARQUIA DE MEMÓRIA

3.1 Considerações iniciais

Neste capítulo são elencados os principais conceitos que formam a base do REA Amnesia, apresentado na Seção 5.2. Os conceitos apresentados neste capítulo são sobre a arquitetura de von Neumann e hierarquia de memória, destacando o subsistema de memória, desde registradores até a memória secundária, incluindo os conceitos de gerência de memória e memória virtual.

3.2 Arquitetura de von Neumann

A arquitetura de von Neumann é uma arquitetura de computador digital com o conceito de programa armazenado, o qual é formado por três componentes principais: processador ou unidade de processamento central (*Central Processing Unit* - CPU); memória para armazenamento de instruções e dados; entrada e saída (*Input/Output* - I/O) para permitir a comunicação com outros dispositivos ou com o ser humano (STALLINGS, 2006).

O conceito de programa armazenado determina que instruções e dados são representados na memória como números binários. Desta maneira, o processador pode realizar a leitura de instruções e dados da mesma forma diretamente da memória.

Os projetistas do ENIAC[1] e John von Neumann (que era consultor do ENIAC na época), desenvolveram a ideia de programa armazenado; paralelamente, Alan Turing desenvolveu uma ideia similar. A concretização do conceito ocorreu em 1945, com a proposta de um novo computador, o EDVAC[2].

[1] *Electronic Numerical Integrator And Computer* - Foi o primeiro computador eletrônico de propósito geral. (WEIK, 1961)

[2] *Electronic Discrete Variable Automatic Computer* - utilizava o sistema binário e possuía arquitetura de von Neumann.

Em Stallings (2006) é apresentado o esboço da proposta realizada por von Neumann para a criação da arquitetura proposta von Neumann. A descrição completa dessa proposta encontra-se em Neumann (1945). Baseado nesta proposta, em 1946 von Neumann e seus colegas começaram a desenvolver uma arquitetura de computador conhecida como *Institute for Advanced Studies* (ISA) da Universidade de Princeton.

Um protótipo foi construído em 1952 com todas as características de um computador de propósito geral, além do conceito de programa armazenado. O protótipo serviu de modelo para todos os computadores de propósito geral subsequentes até hoje, com raras exceções (STALLINGS, 2006). A arquitetura IAS é apresentada na Figura 3.

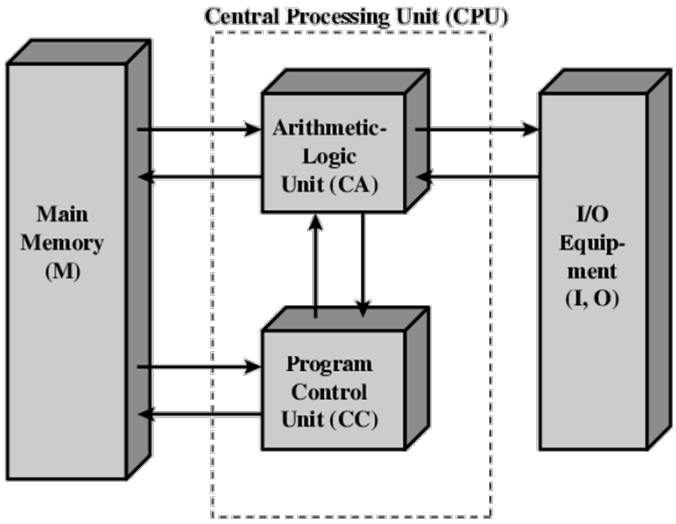


Figura 3 – arquitetura IAS baseada na arquitetura de von Neumann (STALLINGS, 2006).

Os componentes da arquitetura IAS são:

- *Main Memory* (M): conhecida como memória principal, é o local onde são armazenados e acessados as instruções e os dados dos programas;
- *Artithimetic-Logic Unit* (CA): conhecida como unidade lógica e aritmética, é a unidade que realiza operações Aritméticas/Lógicas básicas, como soma, subtração, multiplicação AND, OR, entre outras;
- *Program Control Unit* (CC): conhecida como unidade de controle, é a unidade responsável por interpretar as instruções de um programa vindas da memória e coordenar a sequência de execução dessas instruções;
- *I/O equipment* (I/O): conhecido como módulo de entrada e saída, é responsável por coordenar operações de entrada e saída de dados de dispositivos periféricos.

Os componentes de um computador precisam comunica-se entre si, portanto, deve haver caminhos para a ligação entre os módulos. Segundo (STALLINGS, 2006), a coleção desses caminhos

que conectam os módulos de um computador é chamada de estrutura de interconexão. Essa estrutura depende de como será realizada a troca de informações entre os módulos.

O barramento é uma estrutura de interconexão utilizada nos computadores atuais. Diversos componentes podem ser conectados em um barramento, onde somente um componente pode realizar a transmissão de dados para outro componente por vez. Caso mais de um componente transmita dados no barramento ao mesmo tempo, os dados podem ser sobrepostos, corrompendo-os e, assim, inviabilizando a leitura dos mesmos.

O barramento que liga a CPU à memória na arquitetura de von Neumann torna-se um gargalo, pois somente ocorre uma transmissão por vez; isso limita seriamente a velocidade de processamento. A CPU é continuamente forçada a esperar por dados que precisam ser transferidos para ou a partir da memória. Como a velocidade da CPU e o tamanho da memória têm aumentado mais rapidamente que a taxa de transferência do barramento, o gargalo se tornou um problema, cuja gravidade aumenta a cada geração de CPU.

A partir da observação do comportamento dos programas foi descrito um princípio que tenta minimizar o gargalo entre a CPU e a memória. Esse princípio é baseado no fato que a grande maioria dos programas executam repetidamente alguns trechos de código e acessam muitas vezes dados que estão armazenados próximos na memória. Observando esse comportamento dos programas foi observado o princípio de localidade.

Segundo Patterson e Hennessy (2014), o princípio de localidade diz que os programas acessam uma pequena parte do seu espaço de endereçamento em um instante de tempo e tendem a ficar acessando essa pequena parte do espaço de endereçamento por um tempo. O princípio de localidade é dividido em:

- Localidade Temporal: quando uma posição do espaço de endereçamento é referenciada, ela tende a ser referenciada novamente num futuro próximo. Isso ocorre, por exemplo, devido ao uso de *loops* e contadores no código.

- Localidade Espacial: quando uma posição do espaço de endereçamento é referenciada, as posições próximas tendem a ser os próximos endereços referenciados. Isso ocorre, por exemplo, devido ao uso de vetores, pilhas e busca sequencial de instruções no ciclo de busca.

Segundo Patterson e Hennessy (2014), para aproveitar o princípio de localidade temporal e espacial, as memórias de um computador podem ser construídas como uma hierarquia de memória. Aspectos da hierarquia de memória são descritos na próxima Seção.

Backus (1977) descreve uma maneira de solucionar o problema do gargalo entre a CPU e a memória. Nesse trabalho foi cunhado o termo “gargalo de von Neumann” e para converter esse problema pode-se utilizar uma memória cache entre o processador e a memória principal. Como

as memórias caches baseiam-se no princípio de localidade temporal e espacial, o problema de desempenho pode ser aliviado, até certo ponto e dependendo do padrão de acesso.

A memória cache e a memória principal fazem parte da hierarquia de memória e serão descritas com mais riqueza de detalhes na próxima Seção.

3.3 Características da hierarquia de memória

Nessa Seção são apresentadas as memórias presentes nos computadores atuais e como elas estão dispostas de forma hierárquica. São apresentados conceitos de registradores, de memória cache, de memória principal e de memória secundária. Além disso, é apresentado como o trabalho em conjunto desses componentes faz com que a hierarquia de memória seja utilizada até hoje.

Segundo Tanenbaum e Bos (2014), idealmente, todo computador deveria ter uma memória infinitamente grande, rápida e não-volátil (não perde os dados quando sua fonte de energia é retirada) e de baixo custo. Com a tecnologia atual, contudo, não há memórias que tenham baixo custo e alto benefício. Como alternativa para solucionar esse problema surgiu a hierarquia de memória.

Segundo Stallings (2006), uma arquitetura de computador não deve ser dependente de um componente e/ou de tecnologias de memórias, pois as memórias possuem as seguintes características: quanto mais rápido o tempo de acesso, mais alto é o seu custo por bit e menor será sua capacidade. Por esse fato se faz a necessidade do uso de uma hierarquia de memória.

A hierarquia de memória de um computador está definida da seguinte forma, quanto mais próxima a memória é do processador maior é seu custo, menor sua capacidade, maior é sua velocidade e maior será a frequência de acesso à memória feita pelo processador. Essa disposição de cada componente de memória pode ser visualizada na Figura 4.

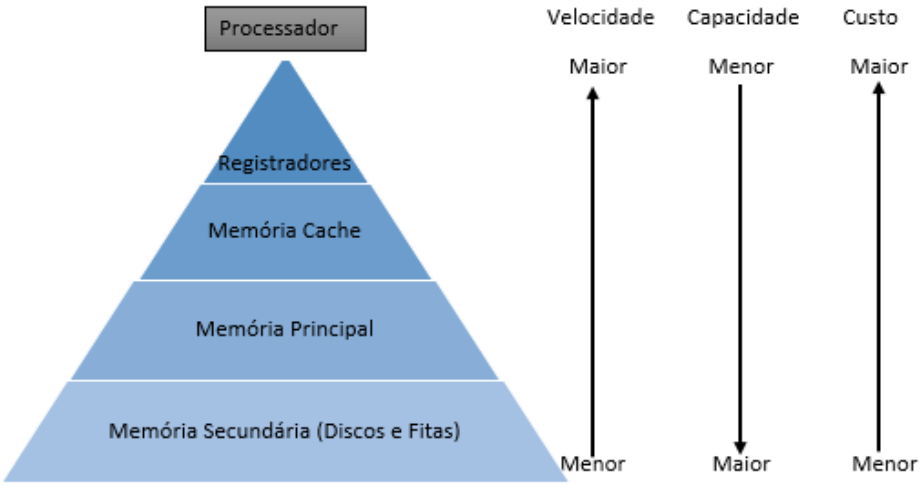


Figura 4 – Hierarquia de memória relação entre velocidade, capacidade e custo (PATTERSON; HENNESSY, 2014).

Segundo Patterson e Hennessy (2014) devido a diferenças em relação a custo e tempo de acesso, torna-se vantajoso construir uma hierarquia de memória. Sendo assim as memórias mais rápidas devem ser mantidas próximas ao processador.

Existem três tecnologias principais utilizadas na construção de memórias presentes na hierarquia de memória. Hennessy e Patterson (2011).

- *Static Random Access Memory* (SRAM): Unidade de memória em que os bits são armazenados estaticamente (como nos *flip-flops*[3]) e, desde que haja energia, o valor pode ser mantido indefinidamente. São circuitos integrados (normalmente *arrays* de memória) com uma única porta de acesso para leitura/escrita. SRAMs possuem um tempo de acesso fixo para cada dado e são utilizadas na construção das memórias cache.

- *Dinamic Random Access Memory* (DRAM): memória em que o valor em uma célula é mantido e armazenado devido à carga de um capacitor[4]. Um único transistor[5] é utilizado para ler essa carga ou para escrever sobre a carga armazenada. Como as DRAMs utilizam apenas um único transistor por bit de armazenamento, logo utiliza menos área por bit de memória, portanto tem uma maior densidade em relação às SRAM. A tecnologia DRAM é utilizada na construção da memória principal.

- *Magnetic Disk* (MD): os discos magnéticos são lâminas de metal ou de vidro recobertas com material magnético de gravação em ambos os lados. O MD não é volátil, ou seja, caso sua fonte de energia seja retirada, as informações salvas não são perdidas, diferentemente das tecnologias SRAM e DRAM que são tecnologias voláteis. A tecnologia MD é utilizada na construção de disco rígido.

Uma outra tecnologia é a SSD[6] que é utilizada na construção de memória secundária. Além de SSD existem algumas outras tecnologias de memória, mas elas não são detalhadas aqui, pois não pertencem ao foco deste trabalho. Uma relação entre as tecnologias SRAM, DRAM e MD é apresentada na Tabela 1. Nessa tabela é realizada uma comparação ao tempo de acesso em nano segundos (ns) e custo (US$) no ano de 2010.

Tabela 1 – relação entre tecnologia, tempo de acesso e custo (HENNESSY; PATTERSON, 2011)

Tecnologia	Tempo de acesso	US$ por GB
SRAM	0,5 a 2,5 ns	4.000 a 6.000
DRAM	28 a 36 ns	30 a 60
MD	5.000.000 a 20.000.000 ns	0,20 a 2

[3] Um elemento da memória onde a saída é igual ao valor do estado armazenado dentro do elemento e para o qual o estado é alterado apenas em uma transição do *clock*. (PATTERSON; HENNESSY, 2014)

[4] É um componente que armazena energia num campo elétrico.

[5] O transistor é um dispositivo em estado sólido, feito de silício.(STALLINGS, 2006)

[6] *Solid-state drive* - para armazenamento não volátil de dados.

Observa-se na Tabela 1 que a SRAM é cara, mas o tempo de acesso é rápido. Supondo que uma memória RAM seja produzida com a tecnologia SRAM, uma memória de 4GB, custaria de 16.000 US$ a 24.000 US$, elevando o valor do computador para usuário final. Por esse motivo essa tecnologia é utilizada para construção de memórias caches, pois um tamanho padrão de memória cache é 512KB a 16 MB.

O mesmo acontece com a tecnologia MD: suponha que seja construída uma memória RAM de 4GB. Com essa tecnologia, o custo seria de 0,80 US$ a 8 US$, mas o tempo de acesso seria de 0,005 a 0,02 segundos deixando o computador lento.

A hierarquia de memória faz com que as memórias mais próximas do processador se tornem um subconjunto de um nível mais baixo da hierarquia. Quando um dado é requisitado na hierarquia de memória, a busca pelo dado inicia no nível mais alto e, caso esse nível não tenha o dado requisitado, ele requisita uma cópia para o nível abaixo dele, sendo que o nível mais baixo realiza o mesmo processo, até que o dado seja encontrado. Antes da entrega do dado ao processador o dado é salvo nos níveis acima.

3.3.1 Registradores

Um registrador é a menor unidade de memória. Geralmente, o tamanho de um registrador é referente à arquitetura do computador (32 e 64 Bits). Nas máquinas de propósito geral atuais os registradores armazenam informações que serão utilizadas com maior frequência pelo programa executado no processador. Os registradores são utilizados para realizar a execução de instruções, para a manipulação dos dados e para armazenar os resultados das operações.

Existem dois tipos de registradores: propósito geral e propósito específico. Os registradores de propósito geral são utilizados na manipulação de dados de programas, podendo ser utilizados pela ULA para armazenar os dados de entrada ou são destinados a receber os resultados das operações realizadas na ULA.

Os registradores de propósito específico são registradores utilizados pelo processador para manter ponteiros de memória, ponteiros referentes à pilha ou então são registradores auxiliares nas manipulações de dados.

Registradores permitem acessos são mais rápidos aos dados devido à tecnologia utilizada na produção dos mesmos (*flip-flops*) e porque eles são mantidos dentro do processador - isto faz com que o acesso seja mais veloz por diminuir o gasto de acesso ao barramento; ao contrário das outras memórias que disputam o acesso ao barramento.

3.3.2 Memória Cache

Segundo Stallings (2006) a memória cache visa a obter maior a velocidade maior de acesso e ao mesmo tempo ter uma capacidade consideravelmente grande a custo próximo da memória principal. A memória cache é uma memória que realiza o intermédio dos acessos do

processador à memória principal. A memória cache contém uma cópia de partes da memória principal que são mais acessadas pelo processador.

Segundo Tanenbaum (2007) utilizando-se do princípio de localidade, quando uma instrução ou dado é referenciado, este é trazido da memória principal para a cache. Deste modo, a próxima vez que o processador for utilizar o dado ou instrução, o acesso será mais rápido, desde de que a instrução ou dado esteja ainda presente na cache.

Para o melhor entendimento do funcionamento da memória cache, algumas terminologias precisam ser definidas como:

- Palavra (*Word*): unidade de organização da memória, normalmente medida em bits (por exemplo, palavra de 32 bits). O tamanho da palavra, no caso geral, refere-se ao tamanho das instruções e dados; pode também representar o tamanho de um número inteiro (vide máquinas de 16 e 32 bits).

- Bloco (*Block*): constitui uma ou mais palavras, mas é um número fixo definido pela quantidade de palavras que podem ser escritas em um bloco da memória cache.

- Conjunto (*Set*): constitui um ou mais blocos de tamanho fixo. A mudança no tamanho dos conjuntos influencia a forma de mapeamento da memória cache.

Em Stallings (2006) o mapeamento da memória principal para memória cache é descrito da seguinte forma: a memória principal é constituída de 2^n palavras endereçáveis com n bits e cada bloco da memória tem k palavras, ou seja, a memória principal possui $M = 2^n/k$ blocos.A memória cache possui C blocos de k palavras, sendo que o número de blocos da cache é menor que número de blocos da memória principal$(C << M)$.

Quando o processador requisita a leitura de uma palavra para a memória, primeiramente é verificado se o bloco que contém a palavra está na memória cache. Caso esteja (*cache hit*) a palavra é entregue para o processador. Caso contrário (*cache miss*) a memória cache requisita o bloco referente à palavra buscada para a memória principal. A memória cache salva o bloco e atualiza as informações para futuras buscas desse bloco e entrega a palavra para o processador (vide Figura 5).

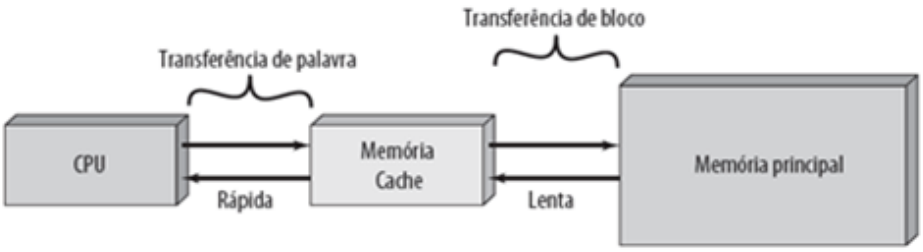


Figura 5 – memória cache e tipos de transferência de dados (STALLINGS, 2006).

Em (STALLINGS, 2006) é descrito como é realizado o mapeamento da memória principal na memória cache, pois como o número de blocos da memória cache é menor que número de blocos da memória principal, é necessário mapear os blocos da memória principal para os blocos da cache. Para exemplificar as formas de mapeamento de memória cache será descrita uma arquitetura de cache exemplo:

- Possui uma memória endereçada a *byte*
- Tamanho da palavra (*Length Word*) de 4 bytes, ou seja, $LW = 2^2$ bytes;
- A cache possui 2 palavras/bloco ($W = 2$);
- A cache possui 16 blocos ($B = 16$);
- O mapeamento de cada bloco é dado pela função *Hash* abaixo:

$$i = j \mod m \tag{3.1}$$

  - i: índice do conjunto da cache
  - j: índice do bloco da memória principal
  - m: número total de conjuntos da memória cache

Existe três modos de mapeamento da memória, que são: mapeamento direto, mapeamento associativo por conjunto e mapeamento totalmente associativo. Primeiramente será explicado o mapeamento associativo por conjunto, pois os conceitos aplicados nesse modo de mapeamento podem ser aplicados aos outros.

O mapeamento associativo por conjunto divide a memória cache em conjuntos contendo blocos. Vamos supor uma memória cache com 4 conjuntos com 4 blocos por conjunto. Cada endereço do bloco da memória principal é mapeado para um conjunto da memória cache que pode escolher aleatoriamente entre os blocos disponíveis do conjunto. No momento de uma escrita de um bloco é utilizada a equação 3.1, para determinar qual o conjunto a ser escrito.

No momento do acesso de uma palavra, o endereço (E) de busca é dividido em quatro campos, sendo: o deslocamento do byte (*Byte offset* = E/LW) caso o endereçamento seja a *byte*, deslocamento da palavra (*Word offset* = $(E \mod LW)/W$), valor do conjunto ($SET = (E \mod (LW * W))/B$) e rótulo ($TAG = E \mod (LW * W * B)$); se memória fosse endereçada à palavra não seria necessário o *Byte offset*. Um Exemplo da divisão do endereço pode ser visualizado na Figura 6.

O endereço é dividido para verificar se o dado ou instrução está ou não presente na memória cache. Para buscar o bloco na memória principal é utilizada a parte de *TAG* e *SET* que formam o endereço do bloco na memória. A verificação do endereço buscado na memória cache segue a seguinte sequência:

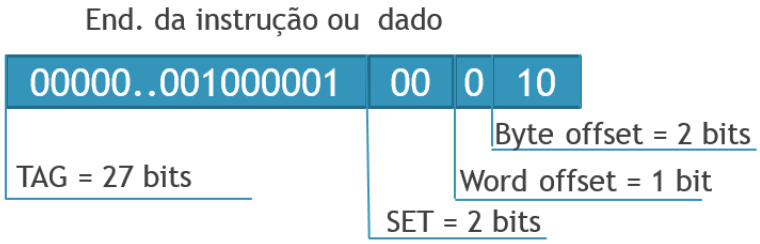


Figura 6 – modo de divisão do endereço de busca.

- O SET é utilizado para determinar o conjunto da memória cache;
- A TAG é utilizada para verificar se o bloco a ser acessado está na memória cache, realizando uma comparação da TAG contida na memória cache com a TAG do endereço acessado. No momento da escrita do bloco na cache a TAG é salva para futuras comparações.
- O *Word offset* é utilizado para dizer qual é a palavra do bloco requisitada pelo processador.
- O *Byte offset* é utilizado pelo processador para dizer qual o byte está sendo requisitado. Alguns processadores endereçam somente palavras, sendo que assim o *byte offset* não é necessário.

O bit de validade (V) indica se aquele bloco está ocupado (1) ou não (0). A representação da estrutura da memória cache descrita aqui pode ser visualizada na Figura 7.

Cache

SET	Block 0					Block 3			
	V	TAG	Word 0	Word 1	...	V	TAG	Word 0	Word 1
0	0				...	0			
1	1	0000	10100...0	1010...0	...	1	0001	10001...0	010010...1
2	1	0010	10101...0	1001...1	...	1	0011	01010...0	010100...1
3	0				...	0			

Figura 7 – memória Cache associativa por conjunto.

O mapeamento direto possui somente um bloco por conjunto. Deste modo, o bloco da memória principal é mapeado diretamente para um bloco na cache. O mapeamento totalmente associativo possui todos os blocos em somente um conjunto. Nessa abordagem, qualquer bloco da memória principal pode ocupar qualquer bloco da memória cache que esteja livre.

Os modos de mapeamento totalmente associativo e associativo por conjunto necessitam de políticas de substituição. No caso da escrita de um bloco em conjunto da cache, quando todos os blocos do conjunto estiverem com dados, é necessário escolher algum bloco para ser substituído. Algumas políticas de substituição conhecidas são:

- Aleatória: o bloco a ser removido é escolhido aleatoriamente.
- *First-in First-out* (FIFO): o bloco a ser removido é o primeiro que foi escrito no bloco da cache ou do conjunto.
- *Least Frequetly Used* (LFU): o bloco a ser removido é o que tem a menor frequência de acesso, ou seja, é o bloco menos requisitado no conjunto.
- *Least Recently Used* (LRU): o bloco a ser removido é o que está há mais tempo sem ser referenciado, ou seja, está há mais tempo sem ser requisitado pelo processador.

Segundo Stallings (2006) atualmente é possível colocar memória cache no mesmo chip do processador, reduzindo assim o uso do barramento externo com o processador. Alguns processadores mais novos colocam até duas memórias caches (L1 e L2) dentro chip do processador e uma memória cache (L3) externa. Cada nível de cache contém porções de instruções ou dados do nível mais baixo.

Quando o processador requisita um dado, o pedido é feito para L1; caso ele não tenha o dado, o pedido é propagado para o nível abaixo, neste caso é L2, que repete o processo. Cada nível de cache pode ser *Unified* (dados e instruções em uma única cache) e/ou *Split* (dados e instruções estão em caches separadas). Isso pode ser visto na Figura 8.

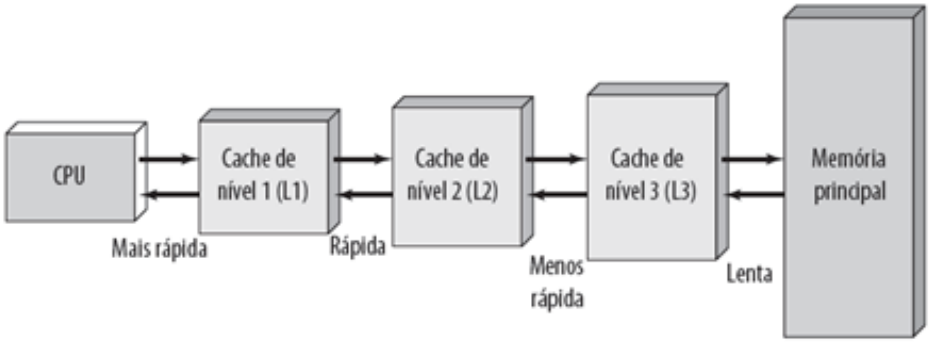


Figura 8 – níveis de cache (STALLINGS, 2006).

A utilização de mais de um nível de cache tende a diminuir a penalidade por falta. Em (PATTERSON; HENNESSY, 2014) a penalidade por falta é descrita como o tempo necessário para buscar um bloco nos níveis mais baixos daquele que ocorreu a falta, incluindo o tempo de acesso, tempo de transmissão e tempo de inserção do bloco no nível que sofreu a falta.

A penalidade por falta em uma cache simples é determinada pelo tempo de buscar o bloco na memória principal. Já em cache multiníveis, a penalidade por falta é minimizada caso o bloco seja encontrado nos níveis de cache abaixo antes de chegar à memória principal.

De uma forma geral, espera-se que as memórias cache maximizem a quantidade de *cache hits*. Uma boa memória cache tem uma taxa de acerto (*HIT RATE*) alta. Essa taxa é dada em porcentagem pelo número de acertos dividido pelo número total de acessos à cache.

3.3.3 Memória principal

A memória principal é uma memória intermediária da hierarquia, comumente chamada de *Random Access Memory* (RAM). A memória RAM é utilizada para armazenar os processos que estão ativos no momento, além de reservar espaço para o SO.

Um processo é caracterizado como um programa em execução. Um processo é composto de área de dados, de instruções e uma pilha de dados dinâmicos. Caso o processador solicite um dado ele é buscado na memória RAM e levado para a cache.

A memória RAM é uma memória volátil, ou seja, ela deve ser provida de energia constantemente; caso o fornecimento seja interrompido os dados são perdidos. Como caso comum e por essa volatilidade, a memória RAM deve ser utilizada como armazenamento temporário.

Segundo Stallings (2006), a memória principal é dividida em duas partes uma área para o SO e outra para um processo que está em execução, isso em sistemas de monoprogramação. Em sistemas de multiprogramação a memória é dividida em área do SO e área de usuário, onde a área do usuário é dividida de modo que possa ser utilizada por vários processos.

Certamente, é preciso que a memória principal mantenha o maior número de processos ativos que queiram utilizar o processador. Uma possibilidade seria aumentar o tamanho da memória, mas os processos têm requisitado mais memória, enquanto o custo da memória principal por bit tem diminuído mais lentamente. Sendo assim, memórias maiores são ocupadas por processos maiores e não por maior quantidade de processos (PATTERSON; HENNESSY, 2014).

A outra possibilidade é a troca de processos (*swapping*), onde os processos que não estão na memória principal são armazenados tipicamente na memória secundária, numa fila de longo prazo ou numa fila intermediária, como apresentado na Figura 9.

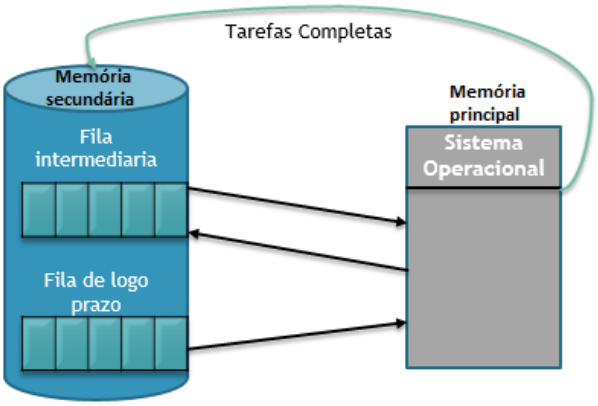


Figura 9 – Filas de processos no Disco (STALLINGS, 2006).

Na Figura 9, vê-se que a fila de longo prazo mantém os processos recém criados prontos para executar no processador e a fila intermediária contém os processos já criados, mas retirados

da memória principal, pois foram atender uma requisição de entrada e saída (ressalta-se que a saída de processos da memória RAM, de fato, nem sempre é necessária; se não houver alta demanda sobre a memória, os processos continuam hospedados nela). Como caso geral, considera-se que, na tentativa de evitar que o processador fique ocioso é realizado o *swapping*, trazendo um processo de umas das filas (intermediária ou longo prazo) que queira utilizar o processador para a memória principal e levado um ou mais processos que estão aguardando uma requisição de entrada e saída na memória principal para fila intermediária. Quando um processo termina ele é removido da memória principal e escrito no disco.

A subdivisão da memória e o *swapping* são abordados de forma mais detalhada no próximo capítulo (Seção 4.2), pois essas técnicas fazem parte da gerência de memória.

3.3.4 Memória secundária

A memória secundária é uma memória não-volátil, ou seja, ela não perde os dados salvos quando sua fonte de energia é retirada. Além disso, têm maior capacidade de armazenamento devido ao custo menos elevado em relação à memória principal. No entanto, o tempo de acesso às suas informações é consideravelmente alto.

Os tipos de memórias secundárias mais conhecidas são: discos rígidos (HDs), discos flexíveis (disquetes), cartões de memória, pen drives (*Flash Drives*), *CD-ROM*, *DVD-ROM*, fitas magnéticas, entre outros.

Como descrito na Seção anterior, a memória secundária tem um papel fundamental na gerência de memória, pois pode armazenar as filas de processos, evitando que o processador fique ocioso. Quando o processo era maior que o tamanho total da memória principal isso era um problema para a gerência de memória, resolvido pelo programador.

Segundo Tanenbaum e Bos (2014) uma solução para esse problema foi adotada em 1960, chamada de *overlays* (sobreposições). Nesta solução, o processo é divido em sobreposições (porção de código e dados) e armazenado na memória secundária; uma sobreposição é trazida para a memória principal somente no momento do uso.

A divisão do processo era feita pelo programador, apesar dessa não ser uma tarefa trivial. Em pouco tempo essa tarefa passou a ser realizada pelo SO, mais especificamente um método criado em 1961 conhecido como memória virtual passou a realizar essa tarefa. Segundo (PATTERSON; HENNESSY, 2014) a memória virtual é uma técnica que usa memória principal como a "cache" da memória secundária. No próximo capítulo serão descritos os métodos *overlays* (Seção 4.2.5) e memória virtual (Seção 4.3) de forma mais detalhada.

3.4 Considerações finais

A hierarquia de memória é utilizada até hoje, pois é uma forma eficiente de diminuição de acessos às memórias mais lentas. As memórias mais próximas do processador tendem a manter os dados mais acessados pelo processador, evitando assim que sejam realizados acessos às memórias mais lentas com muita frequência.

Os detalhes apresentados neste capítulo são abordados nos cursos de computação geralmente na matéria de organização e arquitetura de computadores. Por se tratar de um conteúdo extenso e complexo deve ser apresentado para o aluno de forma gradual. Uma forma de facilitar a apresentação desse conteúdo e utilizar ferramentas e simuladores de ensino.

Um dos conteúdos apresentados pelo simulador Amnesia é o conteúdo de hierarquia de memória. Existem trabalhos na literatura de experimentos com alunos realizados com o módulo memória cache (TIOSSO *et al.*, 2014) do simulador Amnesia. O conteúdo de memória cache é um dos conteúdos mais extensos presentes na hierarquia de memória. O simulador Amnesia realiza um apanhado das principais questões que devem ser apresentadas desse conteúdo.

Com a abordagem da hierarquia de memória se faz necessário apresentar o conteúdo de gerência de memória. Tal gerência deve manter o controle de quais partes das memórias que estão alocadas e quais não estão, para destiná-las (as memórias) aos processos quando eles requisitarem espaço e liberá-las quando eles terminarem (ou quando forem necessárias pelo SO). Alguns assuntos abordados neste capítulo são descritos com maior riqueza de detalhes no próximo capítulo. Os assuntos são: gerência de memória, *overlays* e memória virtual. O estudo desses conceitos foi essencial para o desenvolvimento deste trabalho.

CAPÍTULO

4

GERÊNCIA DE MEMÓRIA E MEMÓRIA VIRTUAL

4.1 Considerações iniciais

Neste capítulo são abordados conceitos referentes à memória virtual, desde do gerenciamento de memória, abordando questões de alocação de processos, *swapping* e os *overlays*. É realizada também uma contextualização da memória virtual, descrevendo paginação e o uso da TLB.

4.2 Gerência de memória

Segundo Tanenbaum e Bos (2014) a memória é um dos recursos mais importantes do computador e por esse motivo deve haver um gerenciamento cauteloso. A parte do SO que gerencia a hierarquia de memória é conhecida como gerenciador de memória; sua principal função é gerenciar a alocação de processos na memória e liberar o espaço quando os processos finalizam.

O sistema de gerenciamento de memória pode ser separado em duas classes: sistema de gerenciamento de memória sem troca de processos e sistema com troca de processos. Os sistemas que não realizam trocas de processos são SO monoprogramados. O processo é colado na memória e retirado somente quando termina a sua execução. Os três modos de endereçamento de sistema de monoprogramação são mostrados na Figura10 .

Dentre os modos de endereçamento apresentados na Figura 10, o modo de endereçamento (A) foi utilizado em computadores de grande porte, o modo (B) é utilizado em sistemas embarcados como computadores de mão (palmtop) e o modo de endereçamento (C) foi utilizado com os primeiros computadores pessoais (SO MS-DOS).

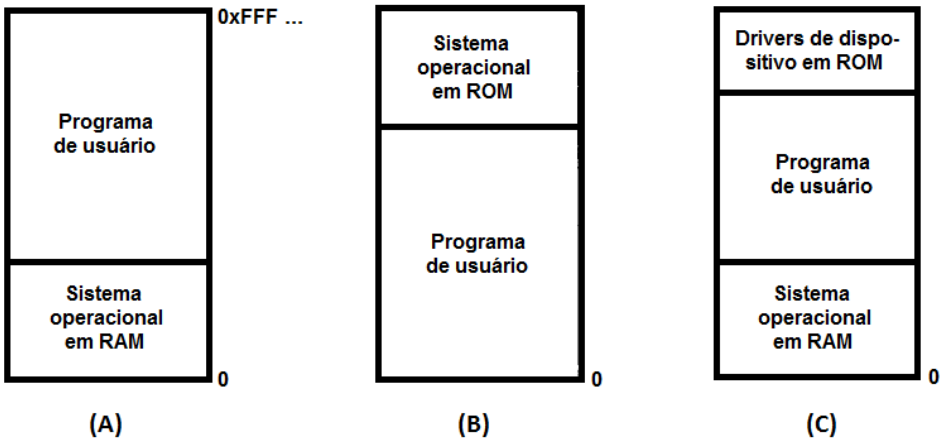


Figura 10 – Modo endereçamento de sistemas de monoprogramação e divisão de memória (STALLINGS, 2006).

Sistemas de gerenciamento de memória com troca de processos são utilizados em SO multiprogramados. A multiprogramação permite que múltiplos processos concorrentes sejam abertos no SO, dando a impressão de estarem em execução simultânea no processador. A maneira mais simples de realizar a multiprogramação do ponto de vista da memória, consiste simplesmente em dividir a memória em *n* partições de mesmo tamanho ou de tamanhos diferentes para alocação dos processos, como pode ser visto na Figura 11 (A) e (B) respectivamente.

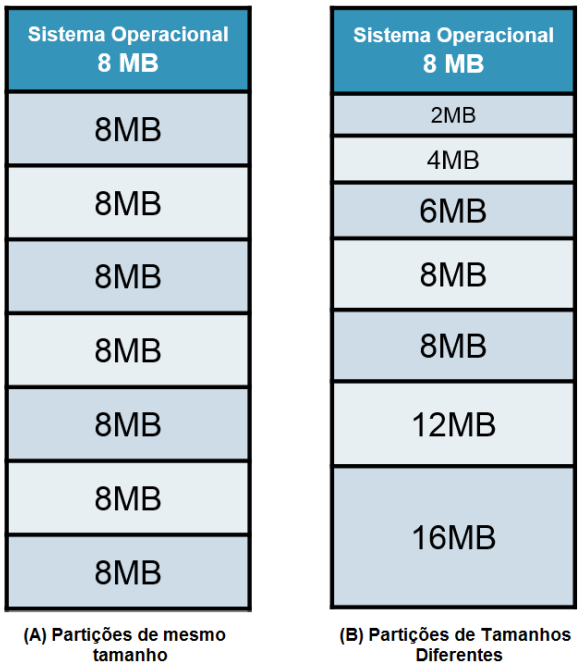


Figura 11 – Particionamento fixo (STALLINGS, 2006).

4.2.1 Alocação de processos em partições fixas

No modo de particionamento apresentado na Figura 11(A), os processos podem ocupar uma ou mais partições da memória, mas caso um processo não ocupe inteiramente uma partição,

o espaço restante desta partição não pode ser utilizado por nenhum outro processo. Quando isso ocorre é dito que a memória está sofrendo fragmentação interna. Uma tentativa de diminuir a fragmentação interna foi particionar a memória em tamanhos diferentes, como pode ser visto na Figura 11(B).

Na chegada dos processos ao sistema, se não houver um espaço na memória para alocar os processos, eles são mantidos em filas. Os processos podem ser mantidos em múltiplas filas, uma para cada partição da memória ou uma fila única para todas as partições.

A desvantagem da ordenação em múltiplas filas é que algumas filas podem ficar vazias e outras com muitos processos, pois o processo é colocado na fila da partição de tamanho ideal. Essa alternativa tenta diminuir a ocorrência da fragmentação interna.

A solução para a desvantagem de filas vazias em múltiplas filas é utilizar uma fila única de processos para todas as partições; quando uma partição for liberada, o primeiro processo da fila ocupa essa partição. A desvantagem dessa solução é que processos pequenos podem ocupar uma partição muito grande, causando assim fragmentação interna. A possível solução para essa desvantagem é fazer uma busca na fila para selecionar um processo com melhor tamanho que possa utilizar o espaço liberado.

4.2.2 Alocação de processos em espaços variáveis

Um outro modo de gerenciamento da memória é utilizar espaços na memória do tamanho exato do processo, como pode ser visto na Figura 12. Este modo elimina a fragmentação interna, mas pode ocorrer a fragmentação externa, que é apresentada a seguir.

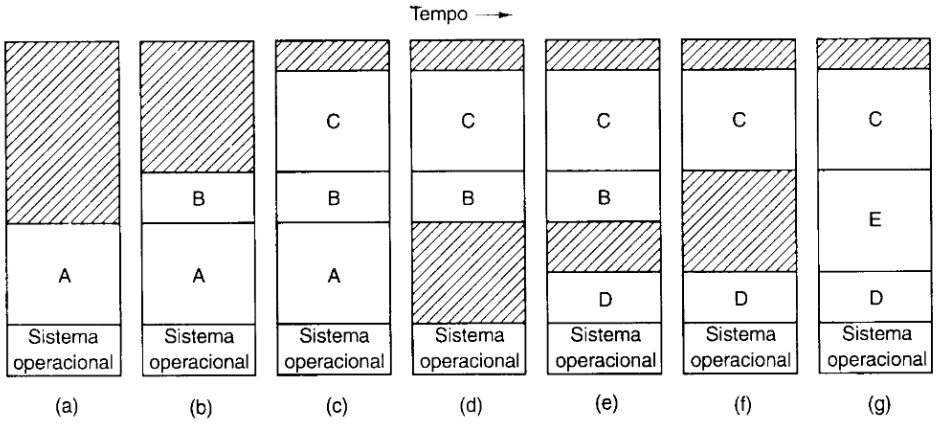


Figura 12 – particionamento de tamanho variável (TANENBAUM; BOS, 2014).

Na Figura 12 é apresentada a alocação dos processos ao longo do tempo, em (a), (b) e (c) são alocados na memória os processos A, B e C em sequência. Em (d), o processo A finaliza e libera o espaço na memória, em seguida o processo D é alocado, como pode ser visto em (e). Nesse momento, está ocorrendo fragmentação externa da memória, pois há espaços pequenos que não estão contíguos na memória, inviabilizando a alocação de um outro processo maior que

os mesmos. Somente com a finalização do processo B, como visto em (f), é liberada memória suficiente para alocar o processo E, como pode ser visto em (g).

Uma alternativa para diminuir a ocorrência da fragmentação externa é realizar uma compactação de memória, mais especificamente mover todos os processos para os endereços mais baixos livres, deixando os espaços contíguos. No entretanto essa alternativa não é viável, pois é gasto muito tempo para mover todos os processos contidos na memória.

Esse modo de gerenciamento de memória deve utilizar um modo de controle dos espaços disponíveis. Existem dois modos para o controle de espaços que são: mapa de bits e lista encadeada. Os dois modos exigem a criação de uma estrutura de dados que deve ser mantida na memória, ocupando assim espaço na memória principal.

4.2.3 Gerenciamento de memória com mapas de bits

No mapa de bits cada unidade de alocação é representada por um bit no mapa de bits, e este bit vale 0 se a unidade de alocação está vazia ou 1 caso contrário. O controle dos espaços disponíveis e ocupados no mapa de bits considera que a memória é mapeada em unidades de alocação de mesmo tamanho.

Cada unidade de alocação pode representar uma palavra ou ter vários *KBytes*. A utilização de unidades de alocação maiores que o tamanho de uma palavra acaba gerando fragmentação interna, sendo aconselhadas unidades de alocação menores.

O tamanho da unidade de alocação define o tamanho do mapa de bits: quanto menor a unidade de alocação maior o mapa de bits. Exemplo de uma unidade de alocação de 4 bytes (32 bits) é representada por 1 bit no mapa de bits, logo uma memória de 4 GBytes, necessita de 128 Mbytes para o mapa de bits ser armazenada na memória principal.

A desvantagem do mapa de bits é a busca por espaços para a alocação de um processo, que pode ser lenta. Quando um processo de tamanho (k) precisa ser alocado na memória, este deve procurar uma sequência de zeros no mapa de no mínimo o mesmo tamanho (k). Muitas das vezes é necessário passar por todo o mapa de bits, para encontrar um espaço suficiente para o processo.

4.2.4 Gerenciamento de memória com lista encadeada

A gerência de memória pode manter uma lista encadeada de segmentos alocados e/ou disponíveis da memória. Em uma estrutura genérica, cada elemento da lista pode ser composto por quatro campos, sendo que o primeiro campo contém uma letra ou um bit sendo H (*Hold*) ou 0 para segmento livre e P (*process*) ou 1 para segmento ocupado. O segundo campo contém o endereço inicial do bloco livre ou ocupado. O terceiro campo apresenta a quantidade de blocos livres ou ocupados. O quarto campo é um ponteiro para o próximo elemento da lista.

A busca por espaços livres para alocação de um processo é feita da seguinte forma: a lista é percorrida e a cada elemento da lista é verificado se o elemento contém H ou P; caso P o segmento que está ocupado então busca-se o próximo elemento da lista até que se encontre um elemento com H. Em seguida é verificada a quantidade de blocos livres; caso a quantidade de blocos disponíveis seja de no mínimo o tamanho do o processo, ele será alocado no endereço contido no segundo campo.

Caso o processo não ocupe o segmento inteiro, o elemento da lista é dividido em dois, sendo um contendo P com o início e o tamanho do novo processo alocado na memória e outro com H com o início e a quantidade de blocos que o processo não ocupou. Os algoritmos que permitem a escolha de segmentos livres com espaço suficiente para um processo ocupar são:

- *First fit* (O primeiro que couber): busca na lista de segmentos o primeiro segmento suficientemente grande que caiba o processo. Após a alocação a busca volta para o início da lista.
- *Next fit* (O próximo que couber): similar ao *First fit*, com a diferença que a busca sempre inicia no último elemento do segmento alocado.
- *Best Fit* (O que menor couber): realiza uma busca inteira na lista, verificando todos os elementos livres e escolhe aquele de menor quantidade de blocos livres que seja suficiente para o processo.
- *Worst fit* (o que pior couber): realiza uma busca inteira na lista, verificando todos os elementos livres e escolhe o segmento que tiver a maior quantidade de blocos de espaço livre.

Os algoritmos *First fit* e *Next Fit* são comumente algoritmos mais rápidos, pois somente no pior caso é percorrida a lista inteiramente na busca de um espaço. Os algoritmos *best fit* e *worst fit* sempre percorrem a lista inteira para encontrar um espaço. Esses dois algoritmos apresentam um desempenho ruim, em relação à velocidade de processamento da lista, mas tendem a minimizar a fragmentação externa.

4.2.5 Overlays

Outro problema nas abordagens de partição de memória ocorria quando o processo tinha um tamanho maior que o tamanho da memória principal, ou seja, não há como alocar o processo inteiro na memória principal. Segundo Tanenbaum e Bos (2014) isso foi resolvido dividindo o programa em porções denominadas sobreposições (ou *overlays*).

Todo o gerenciamento das sobreposições era de responsabilidade do programador, sendo que ele deveria coordenar onde as sobreposições seriam alocadas na memória principal e memória

secundária, como seriam realizadas as trocas das sobreposições entre as memórias e em quais momentos essas trocas deveriam ocorrer.

Para remover esse transtorno do programador, em 1961 um grupo de pesquisadores em Manchester, Inglaterra, propôs um método para executar o processo de sobreposição automaticamente (TANENBAUM; BOS, 2014). Esse método está descrito em Fotheringham (1961), e mais tarde passou a ser conhecido como memória virtual.

4.3 Memória virtual

Historicamente, houve três motivações para a criação da memória virtual: permitir o compartilhamento eficiente da memória entre vários programas (endereçamento relativo), remover o transtorno do programador em relação ao tamanho do programa (permitir que um único processo não precisasse estar totalmente na memória principal para ser executado) e oferecer mais segurança aos processos em execução (PATTERSON; HENNESSY, 2014).

A primeira motivação ainda é predominante: a memória total exigida por todos os programas pode ser muito maior que a quantidade de memória principal disponível. Segundo o princípio de localidade (descrito na Seção 3.2) somente algumas porções dos processos ativos necessita estar presente na memória principal, o que favorece a proposta de não manter, necessariamente, todo o processo na memória.

A segunda motivação se deve ao fato do tempo gasto na criação e gerenciamento das sobreposições em um programa. A memória virtual foi criada para retirar esse trabalho do programador e fazer com que o SO realize essa tarefa.

A terceira motivação é permitir mais segurança dos processos, impedindo que um processo possa acessar o espaço de endereçamento de outros processos.

Segundo Patterson e Hennessy (2014) a memória virtual é uma gerência dois níveis da hierarquia de memória: a memória principal e a memória secundária. Embora os conceitos aplicados na memória virtual e na cache sejam similares, suas terminologias são diferentes. Um bloco de memória virtual é chamado de página e uma falha na memória virtual é chamada de falta de página.

A memória virtual oferece para cada processo uma área de endereçamento referente à arquitetura do computador. Supondo uma arquitetura de 32 bits, ou seja para cada processo a memória virtual oferece 4GB de endereços. Mesmo com uma memória principal menor que esse valor, a memória virtual pode endereçar partes dos processos na memória principal e partes na secundária, as quais juntas geralmente têm espaço suficiente para armazenar vários processos.

Os acessos à memória virtual são feitos por meio de endereços virtuais. Uma combinação de hardware (MMU) e *software* realiza a tradução dos endereços virtuais para endereços reais de página. Caso o endereço virtual traduzido não enderece uma página na memória principal, o SO

tenta trazer a página da memória secundária para a memória principal. Com o endereço real, é possível fazer o acesso ao dado ou instrução presente na memória principal. Existem dois modos de memória virtual: por paginação e por segmentação.

4.3.1 Paginação

Nesse modo de memória virtual os processos observam a memória principal como um espaço de endereçamento virtual, dividido em unidades denominadas páginas, de forma análoga ao modo de gerência de memória com alocação de processos em partições fixas (Seção 4.2.1).

Algumas das diferenças entre paginação e partições fixas são que os tamanhos comuns das páginas são de 4KB a 64 KB; em partições fixas os tamanhos comuns são de 2MB a 16MB, Além disso, na paginação os processos não necessitam ficar inteiramente na memória principal, como é o caso de partições fixas.

A carga de um processo para execução pode ser feita de diferentes maneiras. Em uma delas, o processo é inicialmente carregado em disco, sendo dividido em páginas (STALLINGS, 2006). O SO controla as páginas livres da memória (usando o mapa de bits ou lista encadeada) como pode ser visto na Figura 13(A).

Utilizando um modo de paginação, cópias das páginas são trazidas para a memória principal e o SO cria uma tabela de páginas endereçadas por endereços virtuais, armazenando o respectivo endereço real da página. Além disso, as páginas não necessitam ficar em sequência na memória principal como pode ser visto na Figura 13(B).

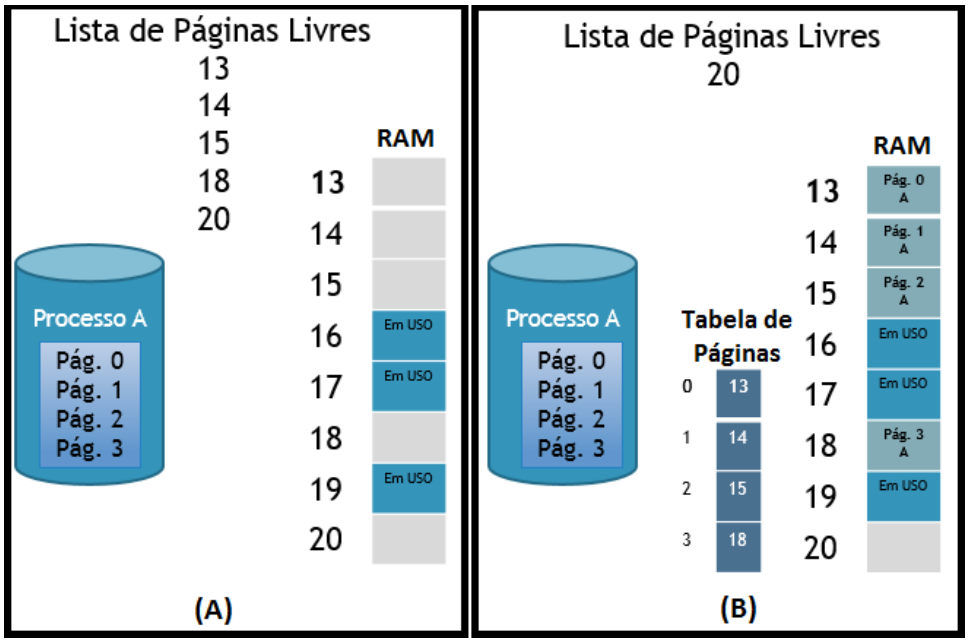


Figura 13 – paginação de um processo(STALLINGS, 2006).

Cada processo tem a sua própria tabela de páginas. Todas as tabelas de páginas dos processos ativos são mantidas na memória principal. O processador sabe qual é a tabela de

páginas do processo que está em execução no momento, pois existe um registador especial que contém o endereço para o início da tabela de páginas do processo ativo.

A memória virtual define quando e como as páginas são trazidas da memória secundária para a memória principal de acordo com um modo de paginação. Os modos mais conhecidos são:

- Paginação Simples: o modo mais simples, que traz todas as páginas do processo que couber na memória principal de uma só vez. Não é muito eficiente, pois gasta muito tempo para trazer todas as páginas da memória e muitas páginas ficarão ociosas por um longo tempo de acordo com o princípio de localidade.

- Paginação por demanda: esse modo de paginação traz somente a página quando requisitada, ou seja, as páginas são mantidas na memória secundária e somente quando o processador requisitar um acesso a uma página que não se encontra na memória principal, a página é trazida da memória secundária (de acordo com o princípio de localidade essa técnica tende a ter um bom desempenho).

- pré-paginação: carrega um conjunto de páginas de um processo na memória principal antes que o processo entre em execução. Sendo assim, tenta evitar a ocorrência de falta excessiva de páginas.

As tabelas de páginas são acessadas pelos endereços virtuais. As tabelas contêm endereços reais das páginas que estão na memória principal. A forma de acesso à tabela página pode ser visualizada na Figura 14.

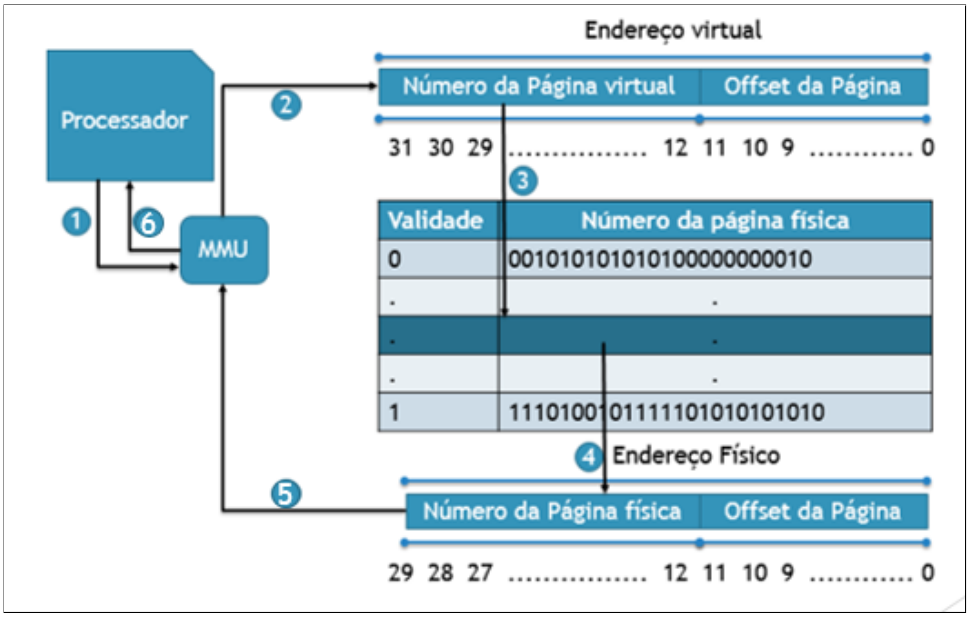


Figura 14 – tradução de endereço (PATTERSON; HENNESSY, 2014).

Conforme mostra a Figura 14, o processador deseja realizar a tradução de um endereço virtual. Em (1) o endereço virtual é salvo na MMU, em (2) é realizada a divisão do endereço separando o número da página virtual e o *offset* da página, que representa o deslocamento dentro da página real.

Em (3) o campo número de página virtual é somado ao endereço de início da Tabela de Páginas, de modo a apontar para um endereço na tabela de páginas do processo. Nesse local é verificado o bit de validade que diz se página está presente na memória principal (bit igual a 1) ou na memória secundária (bit igual a 0). Caso a página esteja na memória secundária é dito que houve uma falta de páginas. Em seguida, há uma exceção, o SO entra em ação de modo que a página é trazida para memória principal, o campo número de página física na tabela é atualizado e o bit de validade é alterado.

Em (4) é concatenado o número da página física com *offset* da página separado em (2). Em (5) é salvo o endereço real na MMU e em (6) o endereço real é entregue para o controlador de acesso à memória no processador.

Uma tabela de página pode possuir, além do bit de validade, outros bits, para serem utilizados em determinados momentos. Alguns dos bits comumente encontrados em uma tabela de página são:

- Bit de referência: é utilizado para dizer se aquela página foi acessada pelo processador (bit = 1) ou não (bit = 0). Em alguns SOs, é realizada uma varredura para trocar os bits de referência que estão por um longo tempo sinalizados com 1, mas em páginas não acessadas recentemente.

- Bit de modificação: esse bit indica se a página foi modificada em operações de escrita realizadas pelo processador. Caso a página tenha sido modificada o bit é 1; caso não o bit é 0. Esse bit é verificado no momento em que a página deve ser retirada da memória principal, pois se a página foi modificada ela deve ser reescrita na memória secundária; caso contrário não.

- Bits de proteção: esses bits dizem quais são os tipos de acessos permitidos na página. Uma das formas contém somente um bit que diz se a página é de leitura (0) ou escrita (1). Outra possibilidade, mais detalhada, possui três bits, sendo um de leitura, um de escrita e um de execução.

Quando uma página é trazida para a memória principal, é atualizada a respectiva linha da tabela de páginas do seu processo; mas quando não há espaço disponível para sua alocação deve ser realizada uma troca de página. A seleção de qual página deve ser removida da memória principal para dar acesso à nova página trazida da memória secundária é feita pelos seguintes algoritmos:

- Algoritmo Ótimo: é simplesmente um conceito de um algoritmo de substituição de página, pois a sua implementação é impraticável. Somente é utilizado como forma de comparação para os outros algoritmos. O algoritmo remove a página que não vai ser referenciada novamente ou que está mais distante de ser referenciada novamente. Para que isso seja implementado é necessário saber quais os próximos passos que o processo irá tomar.
- Algoritmo *First-In First-Out* (FIFO): é mantida uma fila, sendo que as páginas que são trazidas para a memória são colocadas na fila na ordem de chegada. Logo, quando uma página precisa ser trazida para a memória principal e não há espaço, a página que está à frente na fila é retirada da memória para liberar espaço, e a página que entrar no lugar é colocada no final da fila. Esse é um algoritmo simples e de baixo custo.
- Algoritmo *Not Recently Used* (NRU): para esse algoritmo são usados os bits de referência e de modificação. As páginas são divididas em classes e no momento da remoção escolhe-se aleatoriamente uma página correspondente à menor classe presente na memória. As classes são:
  - Classe 0: referência (0) e modificação (0)
  - Classe 1: referência (0) e modificação (1)
  - Classe 2: referência (1) e modificação (0)
  - Classe 3: referência (1) e modificação (1)
- Algoritmo *Second chance* (SC): é uma modificação do algoritmo FIFO, onde antes de remover uma página que está na frente da fila é verificado o seu bit de referência e caso esteja sinalizado com 1, a página é colocada no final da fila e seu bit é modificado para 0. Repete-se o processo até que localize uma página com bit de referência sinalizado com 0 para ser removida.
- Algoritmo *Least Recently Used* (LRU): Existem vários modos de implementar esse algoritmo. Um modo de implementação é utilizar um contador, incrementado automaticamente após cada *tick* de *clock*. Além disso, cada entrada da tabela de páginas deve ter um campo que armazena o valor do contador. Quando a página é referenciada ela armazena o valor do contador na sua linha da tabela de páginas. Quando uma página for substituída é escolhida a página com o menor valor no campo contador armazenado.

Existem outros algoritmos que são variações desses acima descritos. A escolha do algoritmo deve levar em conta o desempenho, custo em questões de hardware e implementação.

4.3.2 Translation-Lookaside Buffer (TLB)

A memória virtual trouxe grandes contribuições para a gerência de memória, mas com a memória virtual são necessários, no mínimo, dois acessos à memória principal para obter uma

instrução ou dado. Há um acesso para traduzir o endereço virtual para endereço real; outro acesso para obter o dado ou instrução. Isso reflete negativamente no desempenho dos computadores.

Segundo Tanenbaum e Bos (2014) a solução foi equipar computadores com um dispositivo de hardware para mapear endereços virtuais para endereços físicos sem passar pela tabela de páginas. Esse dispositivo é uma memória cache denominada *Translation-Lookaside Buffer* (TLB), localizado internamente na MMU.

Segundo Patterson e Hennessy (2014) a TLB é uma cache onde são mantidos os resultados do mapeamento de endereços virtuais para endereços reais das páginas. A TLB possui pouca quantidade de entradas e geralmente tem o mapeamento totalmente associativo para tentar minimizar a ocorrência de faltas e agilizar as consultas. Cada entrada da TLB contém: bit de validade (V) e bit de referência (R), além da TAG de endereço virtual da página e número da página real do respetivo endereço virtual.

O endereço virtual no momento da busca é dividido em deslocamento do endereço dentro da página (*page offset*) e número da página virtual (*page frame*). *Page frame* é utilizado para comparar com a TAG contida na TLB ou para buscar o endereço real na tabela de páginas. *Page offset* é concatenado com o endereço real adquirido na TLB ou na tabela de páginas, para buscar o dado ou instrução na cache ou memória principal. A TLB e a divisão do endereço virtual podem ser vistas na Figura 15.

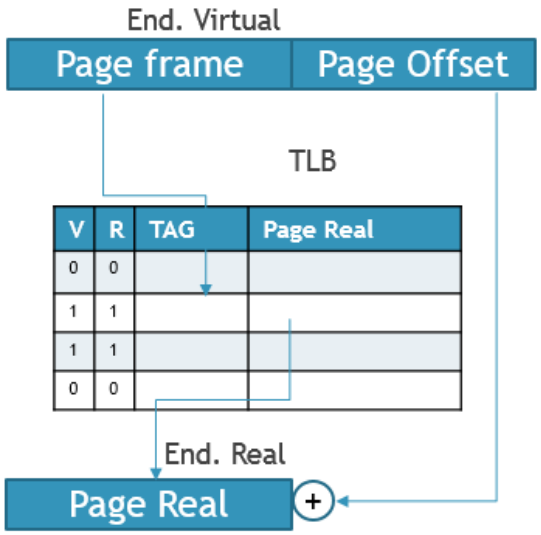


Figura 15 – Tradução de endereço com TLB(PATTERSON; HENNESSY, 2014).

Quando o processador vai traduzir um endereço virtual para o real, ele realiza a consulta respectivamente na TLB e na tabela de páginas; caso a TLB contenha o endereço real, ela retorna imediatamente ao processador. Nesse caso, a busca na tabela de páginas é cancelada. Mas, se a TLB não contiver o endereço real, ela aguarda a tradução ser feita usando a tabela de páginas e quando o endereço real for ser entregue ao processador, a TLB também atualiza as informações para futuras buscas do endereço real.

Com o endereço real, é realizado um acesso à cache para verificar se ela contém o dado. Caso não contenha é realizado o acesso à página na memória principal referente ao endereço real, para ser entregue a instrução ou dado para o processador. Antes de entregar a instrução ou dado para o processador, a cache é atualizada. As operações de acesso à TLB e acesso à cache podem ser visualizadas na Figura 16.

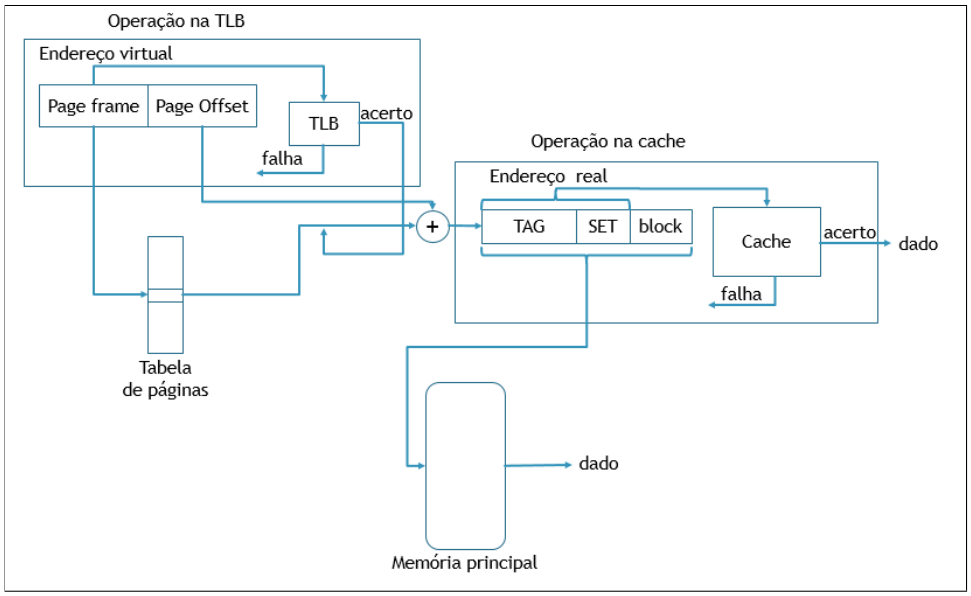


Figura 16 – Fluxograma de execução (STALLINGS, 2006).

Segundo Patterson e Hennessy (2014), quando o processador requisita um dado ou instrução, no pior caso essa requisição pode gerar falta na TLB, na tabela de páginas e consequentemente na cache. Isso ocorre pois deve existir uma consistência entre as memórias. Uma informação não pode estar na cache e na TLB sem que a respectiva página esteja na memória principal.

Supondo que um endereço seja traduzido pela TLB, mas ao acessar a memória principal a página não esteja lá, o dado ou a instrução que seriam passados para a cache e depois ao processador seriam totalmente erradas.

O SO deve garantir essa consistência entre as memórias. No momento que uma página é retirada da memória principal, as informações salvas na TLB e na cache da respectiva página são removidas pelo SO, zerando o bit de validade.

4.4 Considerações finais

A gerência de memória cuida de alocar porções de memória para os processos ativos. Ela passou por uma grande evolução. Inicialmente os processos tinham que ser colocados de forma integral na memória principal, mas partes dos processos ficavam ociosas na memória principal. Além disso, quando o tamanho em bytes dos processos era maior que o tamanho da memória principal, era necessário realizar partições nos processos, chamadas de *overlays*.

A memória virtual é a evolução direta da técnica de particionamento de processos (*overlays*). O conceito referente à memória virtual considera o uso de páginas como unidades de alocação na memória principal (*Page Frames*). Todas as páginas atribuídas a um processo não precisam estar na memória principal concomitantemente e dispostas de forma contíguas. O mapeamento dessas páginas é realizado pela tabela de páginas. A tabela de páginas é endereçada por endereços virtuais e possui os endereços reais das páginas.

O conteúdo gerência de memória e memória virtual é normalmente apresentado na matéria de SOs, nos cursos de graduação em computação. Por se tratar de conteúdo complexo e extenso ele dever ser apresentado com cuidado para que os alunos não fiquem com dúvidas. Nas aulas sobre memória virtual devem ser apresentados aspectos estruturais, funcionais e de desempenho.

O simulador Amnesia possui um módulo de memória virtual que pode auxiliar o professor na apresentação do conteúdo. O simulador Amnesia é apresentado no capítulo 5. Além disso, foram desenvolvidos planos de aulas, tutoriais de utilização, conteúdo teórico e um banco de questões. Esses materiais foram desenvolvidos para serem utilizados juntamente com o Amnesia e estão descritos no capítulo 6.

CAPÍTULO

5

SIMULADOR AMNESIA

5.1 Considerações iniciais

Amnesia caracteriza-se por ser um simulador de hierarquia de memória e memória virtual na arquitetura de von Neumann. Sua função é demonstrar o funcionamento dos registradores de um processador, memórias caches, principal e virtual de forma didática, com o objetivo de facilitar e melhorar a aprendizagem nas disciplinas de OCs e SOs.

O simulador está em desenvolvimento desde 2007, por alunos de graduação e mestrado do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP), orientados pelos professores: Prof. Dr. Paulo Sérgio Lopes de Souza e Prof[a]. Dr[a]. Sarita Mazzini Bruschi. A partir de 2013 estão sendo investigadas e incorporadas características de OA e REA no simulador, além de tornar o simulador robusto com a aplicação de testes funcionais.

Existem alguns trabalhos no projeto Amnesia que realizaram experimentos e obtiveram resultados que foram publicados. Em Oliveira *et al.* (2008) é apresentada uma descrição do simulador Amnesia, apresentadas as características de cada módulo, e algumas formas de utilização do simulador. Em Tiosso *et al.* (2014) o simulador Amnesia começa a ser transformado em Objeto de Aprendizagem, mas o foco do trabalho está no módulo de memória cache. Além disso, foram realizados estudos experimentais que demonstraram uma evolução na aprendizagem dos alunos que utilizaram OA Amnesia no conteúdo de memória cache (TIOSSO *et al.*, 2014).

5.2 Conceitos e funcionalidades

O simulador foi produzido utilizando a linguagem JAVA e com os conceitos do padrão de arquitetura de software *Model View Controller* (MVC). Uma das características mais importante para escolha da linguagem JAVA é a sua portabilidade, pois um programa escrito em JAVA é executável em diferentes SO, sobre a máquina virtual JAVA disponível.

A escolha do padrão de arquitetura (*design pattern*) MVC se deve ao fato desse padrão separar a lógica da aplicação da interface com o usuário. Sendo assim, torna-se mais fácil de desenvolver, editar e testar separadamente cada parte do simulador. Uma representação do padrão MVC pode ser vista na Figura 17. Essa representação está descrita em (BURBECK, 1992) da seguinte forma:

- *Model*: gerencia o comportamento dos dados do domínio da aplicação, responde às requisições de informações sobre seu estado (geralmente vindas da *view*), e reponde às instruções de alteração de estados (geralmente vindas do *controller*)
- *View*: gerencia a interface gráfica e/ou texto da aplicação
- *Controller*: interpreta as entradas do usuário para comandar mudanças no *model* e/ou na *View*.

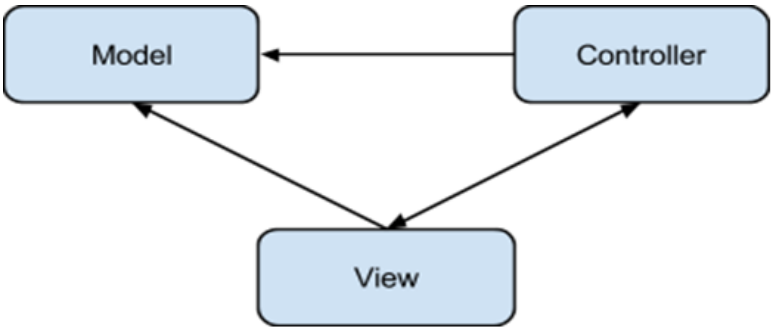


Figura 17 – representação do padrão de software MVC

O Simulador Amnesia aborda os conceitos relacionados à hierarquia de memória e memória virtual, que são abordados nas disciplinas dos cursos de computação. Os conceitos são abordados no simulador tanto por sua interface de texto como por sua interface gráfica, sendo a interface gráfica mais utilizada, por ser mais didática. As suas funcionalidades estão divididas em três módulos, apresentados a seguir:

- Módulo processador: baseado na arquitetura MIPS multiciclo de 32 Bits proposta por (PATTERSON; HENNESSY, 2014). O módulo Processador apresenta todas as características essenciais empregadas na definição de um processador MIPS, como sua unidade de controle (UC), Unidade Lógica Aritmética (ULA) e banco de registradores.
- Módulo cache: responsável por simular memórias caches definidas pelo usuário. As caches podem ser do tipo *Unified* (dados e instruções em uma única cache) ou *Split* (dados e instruções estão em caches separadas). Além disso, as memórias caches podem ser configuradas de acordo com seu modo de mapeamento (mapeamento direto, mapeamento associativo por conjunto e mapeamento totalmente associativo). Os tamanhos de blocos da cache podem ser configurados.

- Módulo memória virtual: responsável por simular uma memória virtual paginada. O módulo é altamente configurável. O usuário pode configurar o tamanho das páginas, o algoritmo de substituição de páginas e adicionar uma TLB ou não. As configurações da TLB são similares às configurações do módulo cache, pois a TLB é uma memória cache.

Os três módulos podem ser executados em conjunto, permitindo a simulação da hierarquia de memória desde dos registradores até o disco. Esta união de módulos exemplifica como um endereço virtual é convertido em um endereço real e como acessar as memórias presentes na hierarquia para obtenção de um dado ou instrução. O módulo memória virtual é muito importante para o desenvolvimento desse trabalho e será melhor detalhado na Subseção 5.2.1.

O simulador Amnesia possui ferramentas auxiliares, que ajudam os alunos na manipulação de seus arquivos de entrada, além de gerar uma nova forma de entrada de dados que são os arquivos de rastros (*Trace*). As ferramentas auxiliares são:

- Conversor de código *Assembly* MIPS para binário: responsável por converter um código *Assembly* MIPS em um código binário (hexadecimal) e vice-versa, pois o módulo processador trabalha com código binário; mas para o aluno é mais fácil desenvolver um código em *Assembly* MIPS;
- Gerador de arquivos de rastro: permite que o simulador Amnesia gere um arquivo de rastro a partir de uma execução. Os arquivos são similares a um *log*, ou seja, os pedidos de leitura ou escrita feitos pelo processador para a memória são salvos em um arquivo, que segue o padrão DIN proposto por (HILL; SMITH, 1989), os arquivos podem ser utilizados para realizar novas execuções. Cada linha do arquivo de rastro é composta de uma dupla: rótulo (decimal) e endereço (hexadecimal). Qualquer outra informação é vista como um comentário. Um exemplo pode ser visto na Figura 18. Os rótulos possíveis são:
  - Rótulo “0”: leitura de dados;
  - Rótulo “1”: gravação de dados;
  - Rótulo “2”: busca de instrução;
  - Rótulo “3”: registro escape (tratado como tipo de acesso desconhecido);
  - Rótulo “4”: registro escape (operação de cache *flush*).

As configurações empregadas em cada módulo podem ser definidas por um arquivo XML chamado arquitetura, o qual pode ser gerado manualmente pelo usuário ou então, com o auxílio de uma ferramenta auxiliar, que está disponibilizada no Amnesia.

Há duas formas de realizar uma simulação. Uma delas é carregar uma arquitetura no simulador, em seguida carregar um arquivo binário que passou pelo conversor de *assembly*/binário, como pode ser visto na Figura 19(a). A segunda é carregar um arquivo de rastro, como pode ser

```
2 408fcc      Há um rótulo seguido
2 408fd0      de um espaço e um endereço
1 426754      (em hexadecimal) por linha,
0 7ffd8be4    podendo assim o restante
2 426730      de cada linha ser usado
2 426750      como comentário.
```

Figura 18 – exemplo do conteúdo de um arquivo de rastro (MORAES; SOUZA; BRUSCHI, 2011)

visto na Figura 19(b). A simulação pode ser executada passo a passo ou toda de uma vez. No final da simulação são geradas estatísticas da simulação e no primeiro caso é gerado um arquivo de rastro que pode ser reutilizado em nova execução.

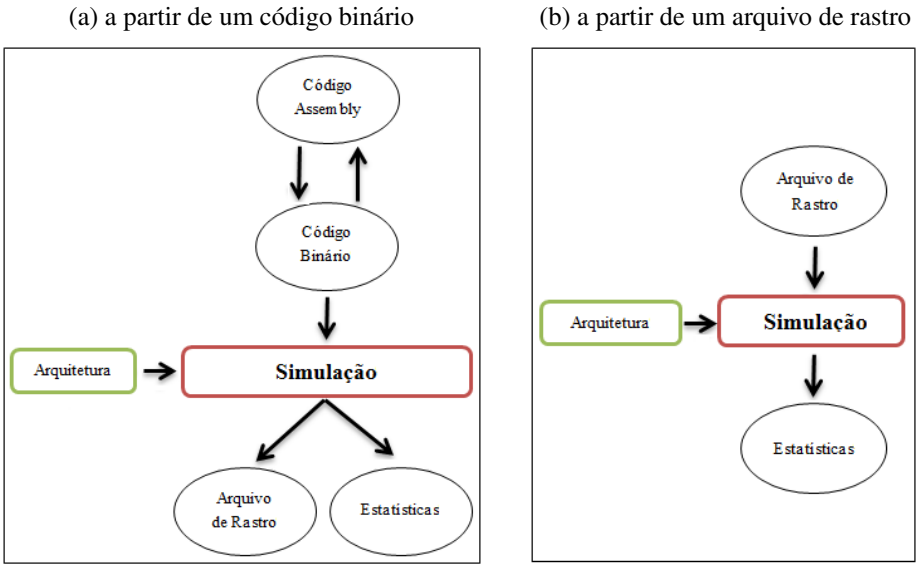


Figura 19 – Modos de funcionamento do simulador Amnesia

O simulador Amnesia somente inicia uma simulação quando um arquivo de arquitetura com configuração correta dos módulos é carregado seguindo de um arquivo binário ou trace válidos. Alguns módulos podem ser omitidos, se não serão utilizados naquela simulação. Exemplo de um arquivo de arquitetura (código 2) e um arquivo de trace (código 3) são encontrados no Apêndice A.

A interface do simulador Amnesia pode ser visualizada na Figura 20, que foi dividida em quatro partes: na parte A da Figura 20, está localizada a barra de menus, local onde o usuário pode acessar para carregar uma nova arquitetura, carregar um arquivo binário, abrir o conversor de *assembly*/binário, entre outras opções.

Na parte B da Figura 20 estão localizadas as caixas de seleção de cada componente. Os componentes são: CPU, registradores, cache, tabela de páginas, TLB, entre outros. No canto inferior da parte B estão localizados os botões RUN e PAUSE. O botão RUN é utilizado para realizar a execução de um programa de uma vez, mas quando o botão PAUSE é acionado, o botão RUN é modificado para STEP, nesse momento a execução é feita passo a passo.

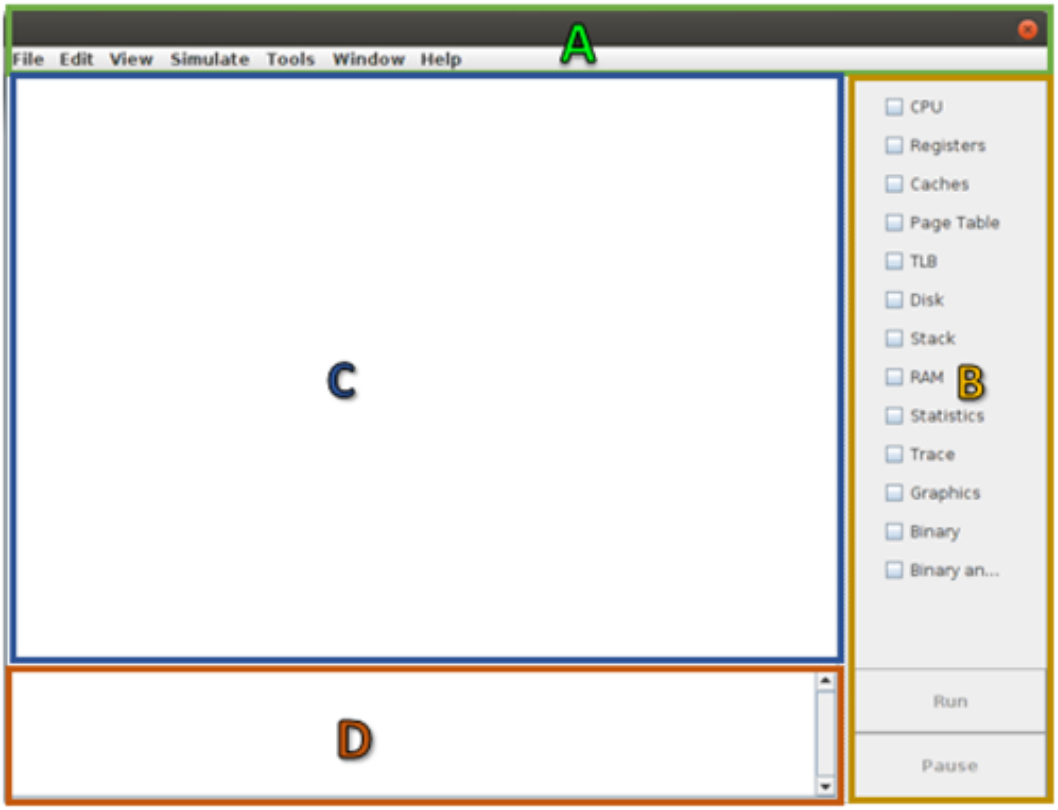


Figura 20 – Interface Gráfica do simulador Amnesia

Quando uma das caixas de seleção é ativada, o componente é exibido na área de visualização; essa área é apresentada na parte C da Figura 20. Cada componente pode ser movimentado e redimensionado pelo usuário de forma livre nessa área.

Na parte D da Figura 20 está localizado o terminal de saída, onde são apresentadas informações relacionadas ao carregamento de arquivo de arquitetura e arquivo binário, caso ocorra um erro no carregamento ou o carregamento ocorra como esperado. Nesse local também são apresentados detalhes da simulação como acesso à TLB, acessos à memória cache e traduções de endereço de páginas.

Na Figura 21 é apresentada uma forma de disposição dos componentes. Tais componentes podem ser utilizados no momento de uma simulação de memória virtual. Os componentes apresentados são: TLB, *Page Table* (Tabela de Páginas) e *Disk* (Disco).

Os componentes como Disco, TLB e Tabela de Páginas são apresentados na Figura 21, e foram posicionados de forma a ter uma boa visualização dos componentes na área de visualização, lembrando que os componentes podem ser livremente movimentados na área. No terminal de saída pode ser visualizada a informação de acesso a TLB (isso ocorre depois de um passo na simulação).

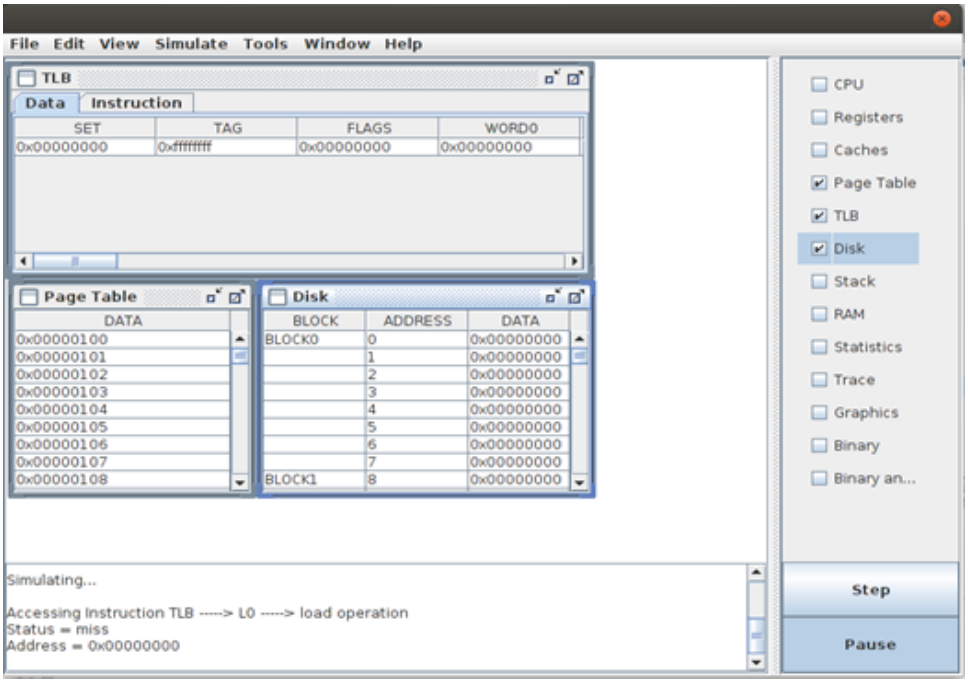


Figura 21 – Amnesia realizando uma simulação de memória virtual

5.2.1 Módulo memória virtual

O simulador Amnesia possui representações das estruturas presentes no conceito de memória virtual, como memória principal, tabela de páginas, disco e TLB. A memória principal é dividida em *page frame*, que possui os dados ou instruções do processo. Caso não seja possível alocar todas as páginas na memória principal, elas são alocadas no disco, a tabela de páginas e endereçada pela *virtual pages* e possui as *page frame* que compõem o endereço real.

As funcionalidades que podem ser estudadas no simulador Amnesia referentes ao conceito de memória virtual são tradução de endereço (tabela de página ou TLB), troca de páginas (*paging*), políticas de substituição, entre outras. Os diferentes impactos no desempenho podem ser observados no simulador devido à inclusão de estruturas (tamanho de página e TLB) e escolhas de funcionalidades (política de substituição).

Mesmo com todos esses aspectos possíveis de serem apresentados pelo simulador Amnesia, ele apresentava sérios problemas de robustez. O simulador não apresentava saídas válidas para grande parte das tentativas de simulações que tentavam ser realizadas. Na Seção 6.2 serão apresentados soluções para tornar o simulador Amnesia um *software* robusto.

O simulador Amnesia somente apresenta a memória virtual paginada, sendo assim não simula a memória virtual segmentada. Outros assuntos não apresentados pelo simulador são tabela de páginas invertidas e tabela de páginas multiníveis, técnicas estas utilizadas na diminuição do espaço ocupado pela tabela de páginas na memória principal. Esses assuntos podem ser abordados de forma teórica com os alunos.

5.3 Simuladores de memória virtual

Nesta Seção são apresentados Simuladores de hierarquia de memória e memória virtual utilizados no ensino. A literatura apresenta muitos simuladores interessantes para o contexto deste trabalho, como simuladores de memória cache, de memória virtual e de hierarquia de memória. Até o momento não foram encontrados OA ou REA de hierarquia de memória ou de memória virtual, embora todos simuladores encontrados na literatura sejam empregados no ensino.

Os simuladores encontrados foram: Dinero IV (EDLER; HILL, 1997), Web-MHE (COUTINHO; MENDES; MARTINS, 2007), *Hierarchical Memory System Environment* (HMSE) (DJORDJEVIC; MILENKOVIC; PRODANOVIC, 1998), *Memory System for Education* (MSE) (DJORDJEVIC, 2005) e SOsim (MAIA; PACHECO, 2003).

Dinero IV (EDLER; HILL, 1997): um simulador de memória cache, muito utilizado na literatura, que possui muitas outras versões remodeladas. O simulador não possui material didático próprio e possui somente o módulo cache, o que permite variações como cache *Split* e *unified*, além de caches multiníveis.

O simulador não possui interface gráfica e todas as simulações são realizadas por linha de comando. As simulações também podem ser realizadas passo a passo ou de forma direta com um arquivo de entrada. O simulador possui suporte para vários formatos de arquivos de entrada, dentre eles: binários e rasto no formato DIN (HILL, 1989). No momento de uma execução, o usuário pode requisitar a geração de um arquivo de rastro, para ser utilizado em uma próxima execução. Cada linha do arquivo possui 3 informações: endereço, tipo de acesso (leitura ou escrita) e tamanho (caso escrita).

Web-MHE (COUTINHO; MENDES; MARTINS, 2007): caracteriza-se como um ambiente web *opensource* que tem como objetivo auxiliar o aprendizado de hierarquia de memória. Este ambiente possui os seguintes módulos: memória cache, memória virtual com TLB e memória principal. O Simulador foi desenvolvido em JAVA e HTML para ser utilizado tanto *online* como *offline*. As simulações podem ser geradas passo-a-passo, onde o aluno pode acompanhar o que está acontecendo em cada módulo separadamente.

O simulador ainda possui tutoriais didáticos, além de *scripts* de configuração e simulação. O modo de utilização do simulador trata simplesmente de carregar um *script* para gerar a simulação de um determinado módulo. Em (MENDES; COUTINHO; MARTINS, 2006) é apresentada uma outra forma de execução da simulação, utilizando um gerador de arquivo de rastro. Cada linha gerada no arquivo corresponde a um acesso à memória e possui três campos: endereço (número), escrita (W) ou leitura (R) e instrução (I) ou dado (D).

Em Coutinho, Mendes e Martins (2007) foi realizada uma avaliação com os alunos para verificar se o simulador Web-MHE ajudaria a melhorar o conhecimento do assunto de hierarquia de memória. Os alunos foram divididos em duas turmas, de acordo com resultados de

um primeiro teste. Em seguida, uma das turmas utilizou o simulador com a ajuda de material didático e a outra classe não teve nenhuma intervenção. Após isso, as duas turmas responderam um segundo teste, onde houve uma melhora na aprendizagem 10% da turma que utilizou o simulador em relação ao desempenho da outra turma.

HMSE (DJORDJEVIC; MILENKOVIC; PRODANOVIC, 1998) é um simulador de hierarquia de memória que possui os módulos: memória cache, memória principal e memória virtual com TLB. As simulações são realizadas no nível de componentes lógicos e cada módulo é apresentado de maneira isolada. O simulador possui somente a versão *desktop*.

A execução do simulador pode ser feita de dois modos: no primeiro, o usuário realiza pedidos de leitura ou escrita para a memória e no final os pedidos podem ser salvos em um arquivo. No segundo modo, os arquivos de pedidos podem ser utilizados para realizar novas execuções. Este trabalho apresenta sete descrições de experimentos de laboratório, alguns deles com foco em memória virtual, mas nenhum dos experimentos apresenta os resultados de avaliações com os alunos.

MSE (DJORDJEVIC, 2005) é um simulador voltado para o ensino e possui material didático. O simulador possui os módulos memória principal, memória cache e memória virtual com TLB. As simulações são realizadas de maneira isolada, ou seja, a simulação de um componente acontece de forma isolada do restante.

O simulador possui uma versão *web* e uma versão *desktop* escrita em Visual Basic. As simulações são realizadas utilizando uma tabela de requerimento. Nesta tabela, o usuário faz o papel do processador realizando pedidos de leitura ou escrita para memória. Cada requerimento é composto por: tipo da requisição (leitura ou escrita), dado (caso de escrita), endereço e tempo de espera.

Em Djordjevic (2005) são descritos sete exercícios para serem utilizados com o simulador com textos explicativos que ajudam o aluno compreender os conteúdos de hierarquia de memória e memória virtual. Os autores acreditam que houve uma melhoria na média de estudantes 7,89 para 8,23, em testes realizados durante o curso, devido a uma média de 18 acessos por dia no site do simulador.

SOsim (MAIA; PACHECO, 2003) é um simulador para o suporte à aprendizagem de SOs. O simulador possui uma interface gráfica interativa, onde o aluno pode adicionar processos *CPU Bound* e *I/O Bound* com diferentes prioridades. A simulação apresenta os processos disputando a CPU. O foco da simulação é a gerência de processos, apresentando os processos nas filas de pronto, execução e espera.

A configuração possível no módulo de memória virtual é a escolha da política de busca de páginas (por demanda ou antecipada); não são possíveis escolhas do algoritmo de substituição de páginas e do tamanho de páginas. O simulador e materiais de apoio para utilização do mesmo

encontram-se disponíveis para download[1].

Em Maia e Pacheco (2003) é realizada uma avaliação qualitativa sobre o simulador SOsim, onde os alunos recebem atividades a serem realizadas com o auxílio do simulador em laboratório. A conclusão apresentada sobre essa avaliação é que os alunos acreditam que o simulador ajudou na visualização dos conceitos e problemas relacionados à área de SO.

Os Simuladores apresentados nessa Seção, com exceção do Dinero IV e SOsim, compartilham de um mesmo problema: a falta de disponibilidade. Alguns artigos descrevem repositórios e sites onde os simuladores podem ser encontrados, mas ao tentar acessar esses repositórios ou sites o simulador foi retirado ou o próprio site não existe mais.

Os simuladores SOsim e Dineiro IV estão disponíveis, mas somente o simulador Dineiro IV possui o código aberto. Como o simulador Dinero IV possui somente o módulo memória cache e interface texto, para a sua utilização há a necessidade de se desenvolver os módulos memória virtual e CPU, para que a mesma possa ser utilizada no ensino de memória virtual. Além desses módulos também será necessário desenvolver uma interface gráfica para facilitar a utilização por parte dos alunos.

Na Tabela 2 é realizada uma comparação entre o simulador Amnesia e os simuladores apresentados nesta Seção, com relação aos modos de simulação, formas de entradas de dados, apresentação de resultados, a iteração com o usuário, entre outras informações. Dentre os simuladores apresentados, o REA Amnesia é o único que realiza simulação do módulo CPU; nos outros simuladores o foco das simulações é somente a cache. A apresentação para os alunos do módulo CPU é importante, pois é na CPU que estão os registradores, primeiro nível da hierarquia de memória .

O simulador Amnesia e o único simulador dentre os apresentados que realiza a simulação dos módulos em conjunto, mostrando para os alunos como é a iteração das memórias no decorrer de uma simulação. Quando o processador realiza um pedido de leitura de dado ou instrução para a memória, é apresentado para o usuário como esse pedido se propaga entre as memórias até que o dado ou instrução seja encontrado e entregue para o processador.

Dois simuladores apresentam interface WEB, mas como o simulador Amnesia está disponibilizado para *download*, alunos e professores podem ter acesso ao simulador para instalação em seus computadores pessoais. Além disso, o simulador foi escrito em JAVA, logo, pode ser utilizado em vários SO diferentes.

Com a disponibilidade do simulador Amnesia de forma gratuita e com o código aberto, professores de outros cursos e outras instituições podem adicionar componentes, reportar defeitos, agregar conteúdo pedagógico. Com exceção do simulador SOsim, os outros simuladores pesquisados não estão disponíveis para *download*.

O simulador Amnesia e o único que oferece um conjunto de planos de aula relacionados

[1] Site do simulador: <http://www.training.com.br/sosim/>

Tabela 2 – Comparativo entre simuladores de hierarquia de memória e memória virtual

Características \ Simuladores	Amnesia	Dinero	Web-MHS*	MSE*	HMSE*	SOsim
Módulo CPU	X					
Módulo Cache	X	X	X	X	X	
Módulo memória virtual	X		X	X	X	X
Simulação em conjunto dos módulos	X					
Versão *desktop*	X	X		X	X	
Versão web			X	X		
Trace	X	X	X	X	X	
Estatísticas	X	X	X	X	X	X
Acesso e/ou *Download*	X	X	X	X	X	X
Código aberto	X	X	X			
Material didático	X		X	X	X	X
Planos de aula	X					
OA / REA	X					
Avaliação quantitativa do módulo memória virtual	X					
Avaliação qualitativa do módulo memória virtual	X					X

com o tema da memória virtual, que auxiliam o professor na apresentação do conteúdo de memória virtual em aulas com o simulador Amnesia. Em um trabalho anterior foram desenvolvidos e disponibilizados planos de aula do conteúdo de memória cache, para ser utilizado juntamente com módulo de cache do simulador Amnesia.

Os simuladores apresentados que têm um módulo de memória virtual não realizaram qualquer avaliação quantitativa para verificar se o módulo ajuda na compreensão do assunto. Como o assunto memória virtual é complexo, uma avaliação quantitativa é necessária para observar se o simulador auxilia na apresentação do assunto. Na Seção 7.3 são apresentados os resultados de avaliações quantitativas com alunos que utilizaram o Amnesia.

No trabalho SOsim (MAIA; PACHECO, 2003), os resultados apresentados da avaliação qualitativa foram obtidos a partir de uma avaliação completa do simulador, onde um grupo de alunos realizaram atividades com o apoio simulador e responderam um questionário com sete perguntas genéricas sobre o simulador. A avaliação não separou dados de atividades realizadas com módulo de memória virtual. Na Seção 7.4 são apresentados os resultados de uma avaliação qualitativa com alunos que utilizaram o módulo de memória virtual do Amnesia.

* Não foi possível verificar as características das execuções desses simuladores, pois não estavam mais disponíveis nos links descritos nos artigos. As características discutidas e apresentadas na Tabela 2 foram obtidas a partir dos artigos disponíveis.

5.4 Considerações finais

O simulador Amnesia é o simulador mais completo dentre os simuladores encontrados na literatura. Além disso, muitos simuladores apresentam somente uma descrição em artigo, mas os simuladores não estão disponibilizados para download. Houve a tentativa de contato com os responsáveis pelos artigos, para verificar se era possível a liberação de uma versão do simulador ou o código fonte, mas não houve retorno.

Como o intuito deste trabalho é ajudar o ensino de memória virtual, o módulo memória virtual do simulador Amnesia se mostra suficiente para desempenhar essa tarefa. Com todos os aspectos estruturais, funcionais e de desempenho que podem ser apresentados pelo simulador, é possível cobrir grande parte dos assuntos de memória virtual.

Antes de utilizar o Amnesia com os alunos, foi necessário realizar uma manutenção no seu código fonte, aumentando sua robustez, melhorando a sua interface e incorporando recursos que facilite a aprendizagem do conteúdo de memória virtual. Tais soluções são apresentadas na Seção 6.2.

CAPÍTULO

6

INCORPORAÇÃO DE CONCEITOS DE APRENDIZAGEM NO SIMULADOR AMNESIA

6.1 Considerações iniciais

O simulador Amnesia se destaca dos demais simuladores de aspectos de OC e SO encontrados na literatura, mas não era uma *software* robusto. Então, antes de utilizar o simulador com os alunos foi necessário realizar um conjunto de testes e correções para deixar o simulador funcional e robusto. O Amnesia possui três módulos e algumas ferramentas auxiliares. Os testes tiveram como foco o módulo memória virtual, mesmo assim, algumas correções foram necessárias nos outros módulos. Tais correções estão descritas no decorrer desse capítulo.

Nesse capítulo são apresentados os testes realizados no simulador Amnesia e como foram incorporadas as características de REA no simulador, viabilizando sua utilização em aula e redistribuição. Além disso, foram desenvolvidos outros materiais de apoio para o ensino de memória virtual.

6.2 Readequações de funcionalidades

Um software robusto tem a capacidade de reagir adequadamente a condições anormais (MEYER, 1997), ou seja, dadas entradas válidas ou inválidas o sistema deve apresentar saídas válidas. O simulador Amnesia não apresentava essa característica, então foi necessário realizar melhorias em seu código para obter um simulador com maior qualidade. Segundo Rocha e Campos (1993) o controle de qualidade de produtos técnicos (*softwares*) pode trazer benefícios tanto no que se refere ao desempenho do produto como nos resultados da utilização deste pelos alunos.

Tabela 3 – Total de testes realizados para cada combinação de memória

	Classe de Equivalência Memórias Presentes no Teste					Arquivo de Rastro Modo de Acesso						
Sessão de Teste	Memória Principal	Tabela de Pag.	Disco	TLB	Cache	Cresc.		Decresc.		Alea.		Total
						R	W	R	W	R	W	
Primeira	X	X				4	4	4	4	4	4	24
Segunda	X	X	X			8	8	8	8	8	8	48
Terceira	X	X		X		8	8	8	8	8	8	48
Quarta	X	X	X	X		16	16	16	16	16	16	96
Quinta	X	X			X	8	8	8	8	8	8	48
Sexta	X	X	X		X	16	16	16	16	16	16	96
Sétima	X	X		X	X	16	16	16	16	16	16	96
Oitava	X	X	X	X	X	32	32	32	32	32	32	96
												648

Para melhorar a qualidade do simulador Amnesia foram realizados testes funcionais. Segundo Fabbri, Vincenzi e Maldonado (2007) o teste funcional é uma técnica de testes conhecida como caixa preta, onde são fornecidas as entradas e as saídas geradas são validadas em conformidade com os objetivos especificados. Essa técnica de teste avalia o sistema segundo o ponto de vista do usuário. Para realização dos testes deve ser criado um conjunto de casos de teste e para que esse conjunto não seja muito amplo são definidas classes de equivalência.

As classes de equivalência têm por objetivo agrupar uma quantidade viável de entradas válidas para a atividade de teste (FABBRI; VINCENZI; MALDONADO, 2007). As classes de equivalência definidas foram os componentes de memórias necessários para simulações de memória virtual no Amnesia. Para cada classe foram definidos dois tamanhos (pequeno e grande) que eram combinados entre si. Para cada combinação de classe definiram-se formas de acessos crescentes, decrescentes e aleatórios, descritas nos arquivos de rastro. Além disso, os arquivos de rastro definiram o tamanho do processo e os modos de acesso (leitura e escrita) realizados.

Realizando uma combinação de todas as classes com seus respectivos tamanhos de processos obtiveram-se os conjuntos de testes apresentados na Tabela 3. Os testes foram realizados em sessões e no momento em que o teste apresentava uma falha, a sessão era paralisada para identificar e corrigir o defeito. Em seguida, todas as sessões eram reiniciadas, para verificar se as alterações realizadas no código não causariam novas falhas.

Após todas as sessões de testes terem sido executadas, foram visualizadas 116 falhas, onde os defeitos foram encontrados e corrigidos. Aproximadamente 18% dos testes apresentaram falhas e grande parte delas estava relacionada à classe *virtualMemory*, pois tratava-se da classe mais estressada nos testes.

Também foram realizados testes com entradas inválidas, pois dada uma entrada inválida o sistema deveria retornar uma saída válida. Sendo assim, dado um arquivo de arquitetura ou

arquivo de rastro inválido, o simulador Amnesia deveria apresentar uma mensagem de erro, que fosse explicativa para o usuário. Foi definido um conjunto de entradas inválidas, para observar quais as mensagens de erro que o Amnesia apresentava.

As mensagens de erro foram reformuladas, pois não se mostravam representativas. Além disso, foi adicionada na mensagem de erro uma possível solução, que guiaria o usuário para verificar os equívocos nos arquivos de arquitetura ou rastro. Um exemplo das mensagem de erro mais representativa é apresentado na Figura 22.

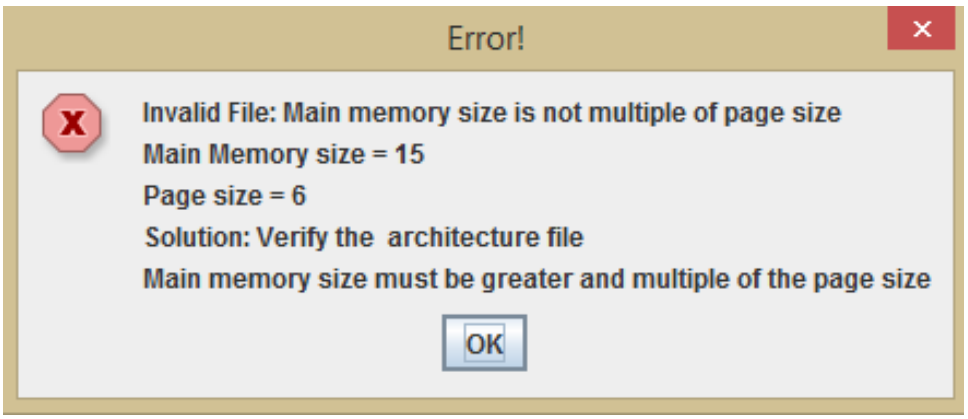


Figura 22 – Exemplo de mensagem de erro informando uma possível solução

6.3 Implementações de funcionalidades

Em muitas classes foram realizadas modificações de métodos, na tentativa de melhorar a legibilidade do código, principalmente na classe *virtualMemory*. Um exemplo de melhora de legibilidade pode ser visto no apêndice B, onde são apresentados o antes (código 5) e o depois (código 6) das modificações realizadas no método *translation*, da classe *virtualMemory*.

Foi realizado um planejamento de readequação do módulo *Page Table*, pois o mesmo envolvia muitas classes a serem modificadas. Isso refletia tanto na *interface* quanto nas classes que controlavam esse módulo. O módulo *Page Table* estava sendo criado em uma porção da memória principal e possuía um componente próprio na *interface*. O módulo *Page Table* precisava ser removido do componente memória principal e ser apresentado somente em seu próprio componente.

Com essa alteração pode-se simular as duas formas de alocação da tabela de paginas, simulando as alocações em um conjunto de registradores ou na área de SO da memória principal. Segundo Tanenbaum e Bos (2014) a forma mais simples (pelo menos conceitualmente) é ter uma tabela de página única que consiste em um conjunto de registradores, com entradas para cada página virtual, indexada pelo número da página virtual. A outra abordagem de alocação da tabela de página, considera alocar inteiramente a mesma na memória principal, onde todo o hardware necessário é um registrador único que aponta para início da tabela. Após as alterações,

para diferenciar as duas abordagens, esta mudança consistiu apenas na mudança dos parâmetros sobre o tempo de acesso.

Foram realizadas melhorias na *interface* do simulador Amnesia, deixando uma *interface* mais intuitiva e de fácil compreensão, para apresentação do conteúdo de memória virtual. Além disso, foram realizadas melhorias na apresentação dos componentes, adicionando informações que podem ajudar a entender como funcionam os acessos de leitura e de escrita. A comparação das *interfaces* Antiga e Nova é realizada destacando alguns pontos nas Figuras 23 e 24.

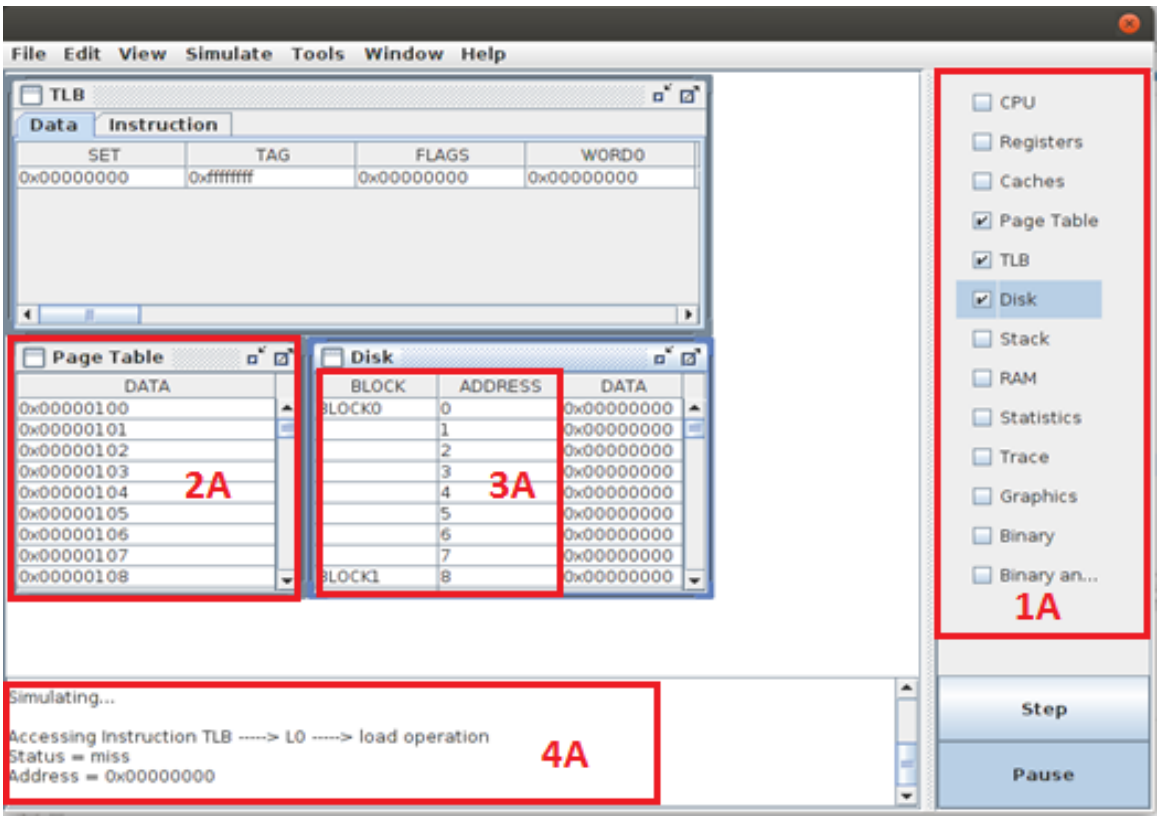


Figura 23 – Interface do simulador Amnesia antes das modificações

Os pontos sinalizados nas Figuras 23 e 24 destacam as alterações realizadas na *interface*. Em 1A são apresentados todos os componentes que podem ser simulados no Amnesia, em 1B foram removidos os componentes que não são necessários nas simulações de memória virtual. Em 2A podemos ver o conteúdo do componente *Page Table*, que concatenava as informações dos bits R (Referência), M (Modificado) e V (Validade) com o *Frame Page* do endereço real. Isso dificultava a ligação de qual *Frame Page* continha o conteúdo da página virtual; em 2B as informações foram separadas colocando cada uma em uma coluna, além de adicionar os campos *Virtual Page* (índice da página virtual) e REP (campo utilizado pelos algoritmos de substituição).

Outros pontos destacados nas Figuras 23 e 24, são referentes à inclusão de informações de índices. Em 3B e 4B são apresentadas as informações de índice referente à divisão em páginas da memória principal e disco, respectivamente (essa informação não era apresentada em 3A). Em 4A é apresentada uma informação de acesso à TLB no terminal de saída. Em 5B essa informação

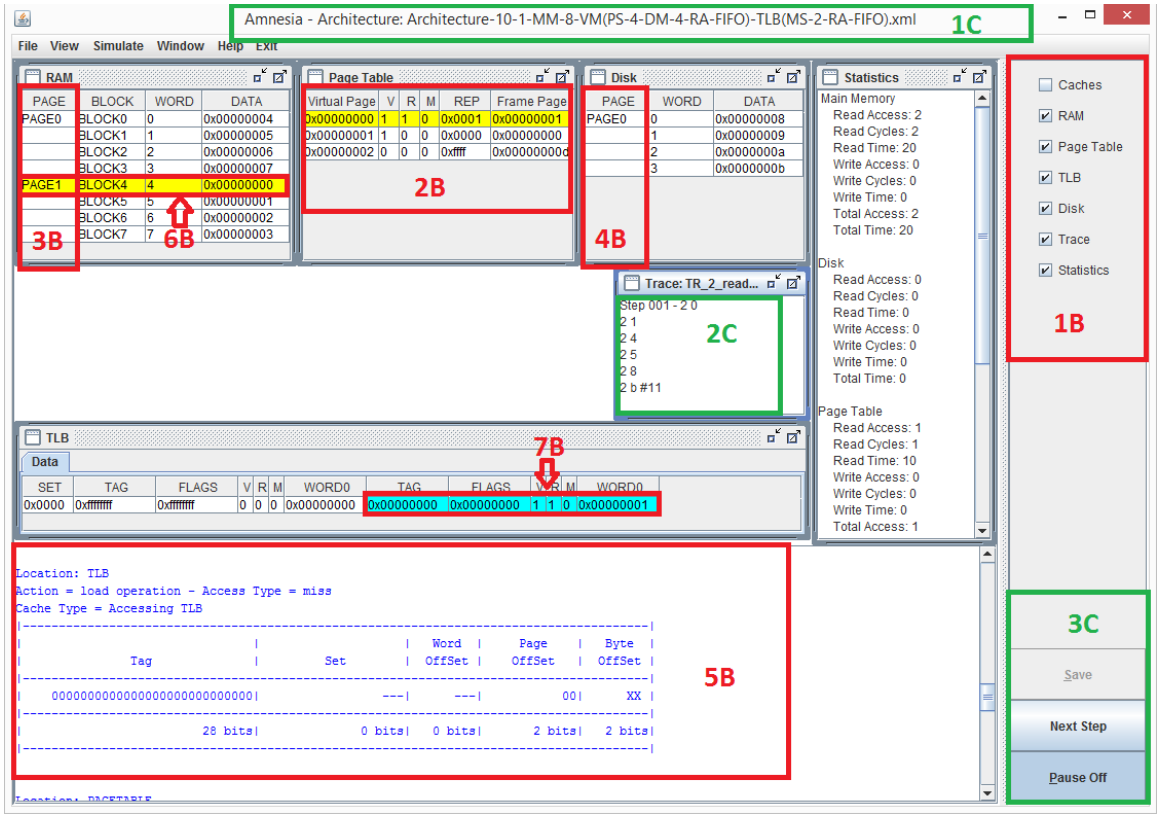


Figura 24 – Interface do simulador Amnesia após as modificações

foi reformulada apresentado a dispersão de bits, onde o endereço virtual de 32 bits é dividido nos campos *TAG*, *SET*, *Word Offset*, *Page Offset* e *Byte Offset*, permitindo assim a explicação do que cada campo significa para os alunos.

Algumas alterações foram herdadas do trabalho de mestrado de (TIOSSO, 2015). Essas alterações estão sinalizadas pela letra C na Figura 24. Em 1C pode-se ver o nome do arquivo de arquitetura, apresentado no topo da janela do simulador. Em 2C destacam-se os passos já executados no arquivo de rastro (com a palavra *Step*). Em 3C o botão *SAVE* é ativado no final da simulação e quando acionado salva um *log* com todas as informações do arquivo de arquitetura, do rastro, acessos às memórias, entre outras informações daquela simulação. No Apêndice A é apresentado um exemplo de *log* (Código 4) da execução.

Em 6B e 7B são apresentados os acessos às memórias destacadas por cores inclusas no componente, referentes ao último passo executado. Essa modificação foi sugerida por um aluno participante do primeiro experimento. As cores destacam em cada componente de memória os locais em que os acessos de escrita e/ou leitura ocorrerão naquele passo de execução. A cor amarela indica um acesso de leitura, a cor azul indica um acesso de escrita, a cor verde indica um acesso de leitura e um de escrita na mesma memória naquele passo de instrução.

Após carregar um arquivo de arquitetura os componentes descritos no arquivo são automaticamente abertos e pré-dispostos na área de visualização (Figura 20 na área C) como mostra a Figura 24. Essa pré-disposição e o redimensionamento dos componentes possibilita que

o Amnesia seja executado em computadores com resolução de tela de no mínimo 1024 x 768 *pixels*. A pré-disposição, o auto carregamento e o redimensionamento dos componentes foram implementados neste trabalho de mestrado.

As correções dos defeitos do simulador Amnesia foram realizadas antes da incorporação dos conceitos de REA. Na próxima Seção serão apresentados os conceitos de REA incorporados no simulador Amnesia.

6.4 Incorporação dos conceitos de REA

Foram incorporadas características de REA no módulo de memória virtual, pelo fato que REA é uma evolução direta de dois conceitos de OA. Os OA têm foco na reutilização em diferentes contextos de ensino; mas os REA podem ser reutilizados, revisados, recombinados e redistribuídos, além de aumentar as chances do Amnesia ser adquirido por alguém que deseje utilizar o conhecimento.

Para incorporar os conceitos de REA no Amnesia, foi incluída a licença de software GNU GPL, pelo fato de ser uma licença específica para software. Em Foundation (2014) são apresentados os passos para inclusão da licença GPL, que são:

- Incluir um aviso de *Copyright* no topo de todos os arquivos fontes do software, seguido do ano do lançamento e os nomes de todas as pessoas envolvidas no desenvolvimento do software;
- Inserir um texto pré-definido em todos os arquivos fontes do software, que se trata de uma declaração de permissão de cópias. Esse texto pode ser visto no Código 1;
- Salvar uma cópia da licença GPL[1] em arquivo chamado "*COPYING*", que deve ser distribuído juntamente com todos os arquivos de fontes do software.

Código-fonte 1: Exemplo de inclusão da licença GPL no código fonte

```
1  /*
2   * Copyright (c) 2015 André Crepaldi Geiger Smidt,
3   * Bruno Henrique de Oliveira,
4   * Carlos Emilio de Andrade Cacho,
5   * Daniel Elias Machado Junho,
6   * Eder Leão Fernandes,
7   * Fernando Tiosso,
8   * Gabriel Bim,
9   * Gabriel de Barros Paranhos da Costa,
10  * Guilherme Rodrigues Buzo,
```

[1] <http://www.gnu.org/licenses/gpl.txt>

```
11  * Jordan Herbert Santos,
12  * Matheus Pedroso de Moraes,
13  * Paulo Sérgio Lopes de Souza and
14  * Sarita Mazzini Bruschi.
15  *
16  *      Amnesia is free software: you can redistribute it and/or
17  *      modify it under the terms of the GNU General Public License
18  *      as published bythe Free Software Foundation, either version 3
19  *      of the License, or (at your option) any later version.
20  *
21  *      Amnesia is distributed in the hope that it will be useful, but
22  *      WITHOUT ANY WARRANTY; without even the implied warranty of
23  *      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
24  *      GNU General Public License for more details.
25  *
26  *      You should have received a copy of the GNU General Public
27  *      Licensealong with Amnesia.  If not, see
28  *      <http://www.gnu.org/licenses/>.
29  */
```

Com essas medidas pôde-se distribuir o código fonte do Amnesia para que outras pessoas colaborem com a manutenção do código, além das atividades dos 4R (vide Seção 2.6). Agora o simulador Amnesia pode ser considerado um REA. Além da inclusão da licença de *software* foi necessário disponibilizar o Amnesia em algum repositório, para que ele fosse apresentado e adquirido por outras pessoas com maior facilidade. Para disponibilizar o REA Amnesia foi criado um site do projeto Amnesia, apresentado na Seção 6.6.

Além do REA Amnesia existem materiais de apoio que são distribuídos juntamente com REA Amnesia (apresentado na Seção 6.5). Esses materiais foram também transformados em REA. Como a licença GPL é mais utilizada em softwares, a licença aplicada nesses materiais foi a *Creative Commons*. Um exemplo da licença é apresentado na Figura 25.

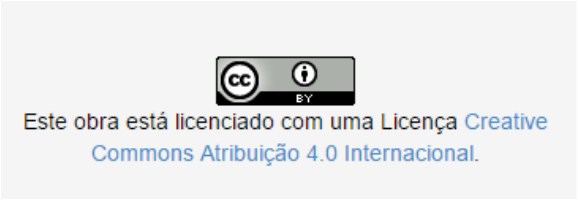


Figura 25 – Exemplo de licença *Creative Commons*

6.5 Desenvolvimento de material de apoio

Os materiais de apoio desenvolvidos neste trabalho têm por objetivo auxiliar professores e alunos no ensino e aprendizado do conteúdo de memória virtual com a ajuda do REA Amnesia.

Foram criados planos de aulas que ajudam os professores na condução de sua aula. Além disso, foram criados o manual de utilização do REA e o texto com conteúdo teórico sobre assunto de memória virtual que ajudam os alunos a entender o funcionamento do REA e a estudarem sozinhos os assuntos memória virtual.

Os planos de aula são compostos de atividades práticas e têm a intenção de tornar o aprendizado interativo, dinâmico e mais atraente. Os planos também cobrem de forma mais ampla e aprofunda os aspectos estruturais, funcionais e de desempenho, que são difíceis de ser abordados no método tradicional de ensino.

Os planos de aula têm objetivos claros e descrevem o que os alunos irão aprender na Aula. As atividades estão relacionadas diretamente com os objetivos que pretende atingir e, além disso, os planos de aula consideram o nível de conhecimento prévio dos alunos que estarão presentes na Aula. Alguns exemplos de atividades são: os benefícios da memória virtual para um sistema, os custos de uma memória virtual para um sistema, diferenças entre os algoritmos de substituição de página e o impacto da TLB no desempenho. Os planos de aula também descrevem os recursos necessários para apresentar o conteúdo.

Foram desenvolvidos seis planos de aula em versões em inglês e português com a seguinte estrutura: **Título** relacionado com a finalidade ou o contexto da aula; **Público alvo** para qual o plano de aula foi projetado; **Motivação** para ensinar o conteúdo; **Objetivo geral** e os **objetivos específicos** a serem alcançados em aula; **Duração** das atividades planejadas; **Desenvolvimento** do conteúdo, que sintetiza o que deve ocorrer na aula por meio das atividades; **Síntese integradora** com os principais conhecimentos trabalhados em aula com os alunos; **Recursos** necessários para conduzir a aula, e **referências** que podem ser usadas pelos professores e alunos para um estudo mais aprofundado do conteúdo.

Os seis planos de aula foram divididos em dois grupos de alunos: com e sem o conhecimento prévio do tema trabalhado na aula. Os planos de aula são:

1. Funcionamento do REA Amnesia;

2. Uso da tabela de páginas e a atividade de paginação no contexto de memória virtual: estrutura, funcionalidade e desempenho em memória virtual;

3. Políticas de substituição de páginas e uso da TLB no contexto de memória virtual;

4. Introdução à memória virtual - conceitos básicos sobre a tabela de páginas, paginação e impacto no desempenho;

5. Introdução às políticas de substituição de páginas no contexto de memória virtual;

6. Introdução ao uso de TLB no contexto de memória virtual.

O plano de aula (1) introduz o REA Amnesia para os alunos, e deve ser utilizado para ambos os grupos de alunos. Os planos de aula (2) e (3) pertencem ao grupo de alunos que tiveram uma aula de memória virtual e precisam apenas fixar o conteúdo. Os planos de aula (4), (5) e (6) pertencem ao grupo sem o conhecimento prévio e são utilizados para introduzir os conteúdos para os alunos por meio de atividades com REA Amnesia. O plano de aula (1) na versão em português é apresentado no apêndice E.

Outro material disponibilizado juntamente com REA Amnesia é o tutorial de utilização do REA Amnesia. Esse material está vinculando ao REA Amnesia e pode ser acessado no menu *HELP* na barra de menus (Figura 20 área A) do REA Amnesia ou aberto na pasta *AmnesiaHelp*. Por se tratar de um Arquivo em HTML pode ser aberto em qualquer navegador WEB.

O conteúdo do tutorial tem por objetivo oferecer um suporte técnico aos usuários iniciantes do REA Amnesia, considerando essencialmente o uso do módulo Memória Virtual. O tutorial apresenta os requisitos necessários para o funcionamento do REA e apresenta de forma ilustrativa o passo a passo para realização de uma simulação. Além disso, apresenta para os usuários, como eles podem criar os seus próprios arquivos de arquitetura e de rastro, para realizar simulações com REA Amnesia. O tutorial de utilização é apresentado no Apêndice G.

O texto referente ao conteúdo teórico, disponibilizado juntamente com REA Amnesia, trata-se de uma síntese dos assuntos referentes à memória virtual. Nesse texto são abordados temas como: motivação da utilização da memória virtual; características referentes à paginação; como é realizada uma tradução de endereço; políticas de substituição; motivação para utilização da TLB; entre outros temas. O conteúdo teórico é um texto ilustrativo similar ao apresentado na Seção 4.3.

O REA Amnesia, os planos de aulas, o tutorial de utilização e o conteúdo teórico estão disponibilizados no site do Projeto Amnesia, que está apresentado na próxima Seção.

6.6 Site do projeto Amnesia

O site do Projeto Amnesia[2] é um local centralizador das informações do Projeto Amnesia. Na página inicial há uma breve descrição do projeto. O site foi desenvolvido utilizando a plataforma Wordpress[3] e na Figura 26 é apresentada a página inicial do site.

Além de informações do projeto Amnesia, o site apresenta a importância de se estudar hierarquia de memória, publicações realizadas, autores envolvidos e requisitos para o *donwload*. Todo o conteúdo do site está em português e em inglês. Além disso, possui uma galeria de fotos do REA Amnesia.

A área de *download* disponibiliza os executáveis e os planos de aula tanto de memória

[2] <http://amnesia.lasdpc.icmc.usp.br/>
[3] <https://br.wordpress.org/>

Figura 26 – Pagina inicial do site Amnesia

cache quanto de memória virtual. Além disso, quem desejar obter o código fonte do REA Amnesia pode entrar em contato e requisitar a liberação. Este contato nos permite ter um controle dos interessados pelo código fonte do Amnesia, apenas para fins estatísticos. A área de contato pode ser utilizada pelos usuários do REA Amnesia para enviar considerações referentes à utilização do REA ou relatório de defeitos encontrados.

6.7 Considerações finais

Nesse capítulo foram apresentadas todas as correções e incorporações de características realizadas no REA Amnesia, antes da utilização com os alunos. Essas correções foram necessárias para a liberação de um software mais robusto, que se encontra disponível no site Amnesia, para quem deseja aprender e ensinar o conteúdo de hierarquia de memória e principalmente de memória virtual, que é o foco deste trabalho.

Todas essas contribuições técnicas apresentadas nesse capítulo foram essenciais para ter um REA Amnesia funcional e com diversos materiais e recursos para facilitar o ensino do conteúdo de memória virtual. Todo esse trabalho foi colocado à prova nos experimentos

realizados com os alunos. Os resultados desses experimentos são apresentados no próximo Capítulo. Além disso, os alunos que utilizaram o REA apresentaram suas opiniões referente ao REA e propuseram mudanças, sendo que uma delas já foi apresentada na Seção 6.2.

A incorporação dos conceitos de REA ajudaram o Amnesia a ser utilizado em outros contextos de ensino, sendo assim, uma contribuição para ensino do conteúdo de memória virtual e memória cache. Além disso, esperamos obter colaboradores para o projeto Amnesia, tanto na produção de planos de aulas como para correções no código fonte do Amnesia, caso o mesmo apresente falhas não observadas nos testes.

CAPÍTULO

7

AVALIAÇÃO DO REA AMNESIA

7.1 Considerações iniciais

O objetivo desta avaliação é verificar a eficácia do REA Amnesia no auxílio ao ensino do conteúdo de memória virtual durante o uso real em sala de aula. Para atingir esse objetivo foram realizados três experimentos. Os resultados dos experimentos foram analisados para verificar se o REA Amnesia melhorou, de fato, a aprendizagem dos alunos no conteúdo proposto.

A avaliação também serviu para obter opiniões dos alunos em questões relacionadas à utilização do REA Amnesia. Algumas dessas questões tratam da facilidade de utilização do REA, da compreensão dos assuntos ministrados, da sequência de atividades apresentadas na aula, da adequação de cada atividade considerando o objetivo proposto para as aulas e da satisfação dos alunos em utilizar o REA.

O primeiro experimento contou com a participação de alunos de pós-graduação (mestrado e doutorado) do Laboratório Sistemas Distribuídos e Programação Concorrente (LaSDPC) do ICMC/USP. Os alunos já haviam tido aulas sobre memória virtual em disciplinas de SOs mas estavam sem contato com o conteúdo de memória virtual a um ano no mínimo, tanto na graduação quanto na pós-graduação.

O segundo experimento contou com a participação de alunos de pós-graduação do ICMC/USP e o terceiro experimento contou com a participação de alunos de graduação do curso de Engenharia de Computação. Tais alunos estavam cursando a disciplina de SOs e tiveram uma aula teórica do assunto de memória virtual antes do início do experimento.

7.2 Descrição dos experimentos

Cada experimento teve sua particularidade na aplicação, mas possuía uma base similar, compostas das seguinte etapas: (1) aplicação de um pré-teste (Apêndice C) com questões

relacionadas ao conteúdo de memória virtual (2), aplicação de um questionário de expectativa de uso (Apêndice D), (3) Aula com a utilização do REA Amnesia, (4) aplicação de um questionário de reação ao uso (Apêndice D), e (5) aplicação de um pós-teste (Apêndice C) com questões diferentes do pré-teste, mas com um mesmo grau de dificuldade.

A aplicação dos testes (1 e 5) faz parte da avaliação quantitativa (Seção 7.3), que verifica se ocorreu uma melhora no conhecimento do aluno com o auxílio do REA. A aplicação dos questionários (2 e 4) compõe a avaliação qualitativa (Seção 7.4), com objetivo de verificar as opiniões dos alunos referentes a utilização do REA, se as expectativas dos alunos foram atendidas e se eles acreditam que o REA traz melhorias no aprendizado, entre outras questões que podem ser analisadas.

O primeiro experimento contou com a participação de nove alunos que não tinham contato com o conteúdo ministrado havia um ano e meio, no mínimo. Nesse experimento, portanto, o REA Amnesia teve o objetivo de relembrar os conceitos já estudados pelos alunos. Esse experimento foi dividido em duas aulas ministradas de forma expositiva com o auxílio do REA Amnesia e contou com a utilização de seus respectivos planos de aula. Ao todo foram desenvolvidas 11 atividades abordando os assuntos de memória virtual. A primeira aula apresentou os assuntos de motivação, aspectos estruturais, funcionais e de desempenho. A segunda aula apresentou os assuntos de políticas de substituição de páginas e TLB.

O segundo experimento contou com a participação de onze alunos, sendo que os alunos tiveram um contato com o assunto de memória virtual em uma aula teórica no curso de SO. Nesse experimento, portanto, o REA Amnesia teve o objetivo sedimentar o conhecimento adquirido nas aulas teóricas. O experimento teve uma aula ministrada de forma expositiva com o auxílio do REA Amnesia e contou com a utilização de um plano de aula com seis atividades desenvolvidas dentro do conteúdo de memória virtual. Os assuntos abordados foram políticas de substituição de páginas e TLB.

O terceiro experimento contou com a participação de quarenta e nove alunos que tiveram uma aula prévia do conteúdo no curso de SO, similar aos alunos do segundo experimento. Sendo assim, o REA Amnesia teve o objetivo de sedimentar o conhecimento adquirido e a aula contou com os mesmos planos do segundo experimento. O diferencial desse experimento é que os alunos foram separados em duas turmas, onde uma turma teve a aula com o REA Amnesia e a outra turma teve uma aula teórica de forma tradicional com outro professor.

A abordagem utilizada no terceiro experimento for similar à utilizada por Coutinho, Mendes e Martins (2007), no qual os alunos foram divididos em quatro grupos baseados no nível de conhecimento prévio (Alto, Bom, Médio e Baixo, separados nos grupos A, B, C e D respectivamente) avaliados no pré-teste. Metade dos alunos de cada grupo foram separados em duas turmas (1 e 2), ou seja, cada turma possuía a mesma quantidade de alunos com o mesmo nível de conhecimento prévio. A Turma 1 teve a aula com auxílio do REA Amnesia e a Turma 2 teve uma aula teórica tradicional. O esquema completo é apresentado na Figura 27.

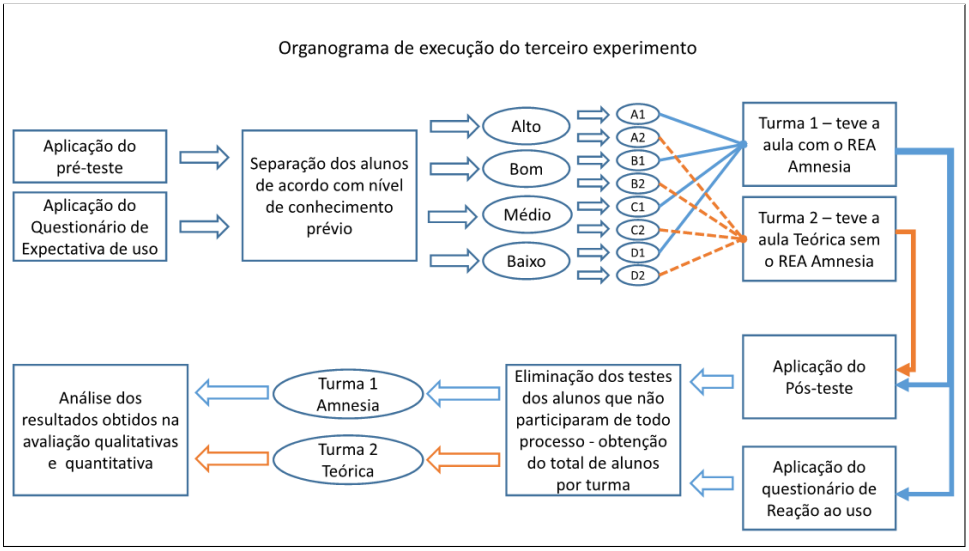


Figura 27 – Organograma de execução do experimento similar a Coutinho, Mendes e Martins (2007)

Conforme a descrição apresentada na Figura 27, cada turma participa de uma aula distinta. Em seguida é aplicado o pós-teste para as duas turmas e a Turma 1 responde um questionário de reação ao uso da REA Amnesia. Não é necessária a aplicação desse questionário para a Turma 2, pois essa turma não tem contato com o REA Amnesia. Os alunos que não participaram de alguma das etapas do experimento, tiveram seus testes e questionários desconsiderados na etapa da análise dos experimentos.

A avaliação realizada por Coutinho, Mendes e Martins (2007) separou as duas turmas, sendo que uma turma teve o acesso à ferramenta WEB-MHE para realizar atividades com auxílio de manuais de utilização e a outra turma não teve nenhuma intervenção no aprendizado. Em seguida as duas turmas continuaram o processo apresentado na Figura 27. A diferença deste trabalho de mestrado é que o terceiro experimento realiza uma intervenção no aprendizado das duas turmas, podendo assim obter uma comparação de duas abordagens de ensino distintas.

7.3 Avaliação quantitativa

O primeiro experimento iniciou com a aplicação do pré-teste para verificar o conhecimento prévio dos alunos sobre os assuntos relacionados à memória virtual. Em seguida, deu-se início à primeira aula com o REA Amnesia e no dia seguinte foi realizada uma segunda aula. O pós-teste foi aplicado em um terceiro dia.

O pré-teste e o pós-teste foram compostos de 15 questões cada, separadas pelos assuntos de memória virtual. Com duas questões de motivação, duas de aspectos estruturais, três de aspectos funcionais, duas de impacto no desempenho, três de políticas de substituição e duas questões sobre TLB. No mínimo uma questão por assunto foi considerada de dificuldade alta ou média.

Analisando a Figura 28 nota-se que houve uma melhora nos acertos em questões do pós-teste, de 73 para 94, ou seja, um crescimento de 28,8%. Em contrapartida, houve uma diminuição 27,1% de repostas erradas, de 48 para 35. Tais resultados podem ser um indicativo que o REA Amnesia ajudou os alunos a consolidarem o aprendizado do conteúdo de memória virtual.

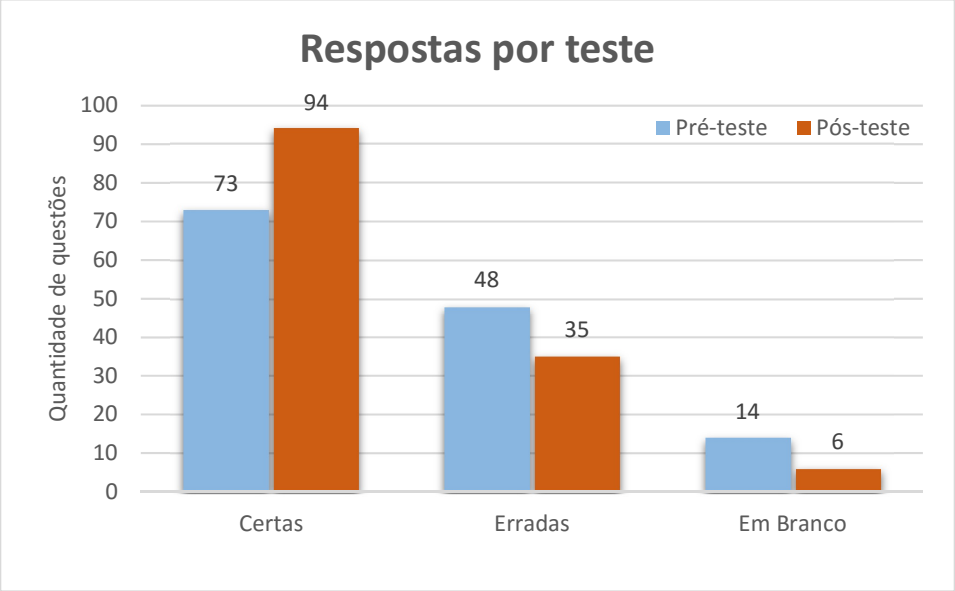


Figura 28 – Gráfico de respostas por teste do primeiro experimento

Ainda na Figura 28 observa-se uma diminuição de 57,1% na quantidade de respostas deixadas em branco de 14 para 6. Aos alunos foi recomendado que deixassem as respostas das questões em branco caso não tivessem certeza da resposta. Dessa maneira, conclui-se que os alunos tiveram maior confiança em responder as questões do pós-teste.

O gráfico *boxplot* na Figura 29 mostra a concentração de 50% dos valores dos acertos, representados pela extremidade inferior (1º quartil, ou percentil de 25%) e superior (3º quartil, ou percentil de 75%) da caixa, a mediana das notas (barra central da caixa), a maior e a menor nota (barras verticais com intimidades) e, caso existam, os *outliers* (representadas por pontos no gráfico).

Nesta Figura 29 observa-se que a concentração das notas no pré-teste ficou entre 7 e 9 acertos com a mediana em 8; no pós-teste houve uma melhora do resultado, a concentração ficou entre 8 e 12 acertos, com a mediana em 11. Analisando esse resultado temos uma concentração de acertos maior no pós-teste, além de uma mediana maior, evidenciando que os alunos tiveram uma melhora significativa no conhecimento.

Na Figura 30, pode ser observada uma melhora nos resultados do pré-teste para o pós-teste na quantidade de acertos por assuntos. Isso pode ser verificado nas questões de motivação, aspectos estruturais, aspectos funcionais, impacto no desempenho e políticas de substituição, as quais tiveram melhora nos resultados. Destas, destacam-se as questões sobre aspectos estruturais

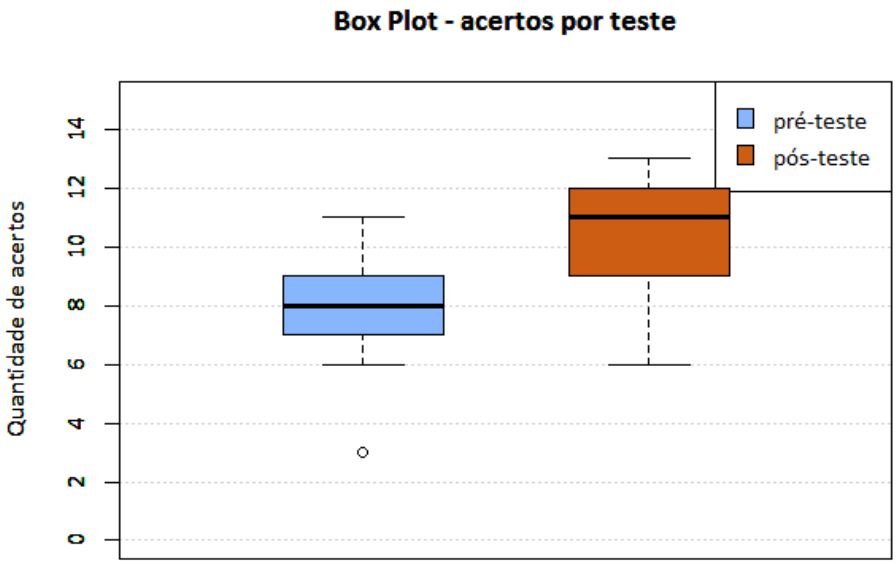


Figura 29 – *Boxplot* de acertos por testes do primeiro experimento

de memória virtual (2 questões por testes) e políticas de substituição (3 questões por testes) que tiveram uma melhora de 47,1% e 40,9%, respectivamente.

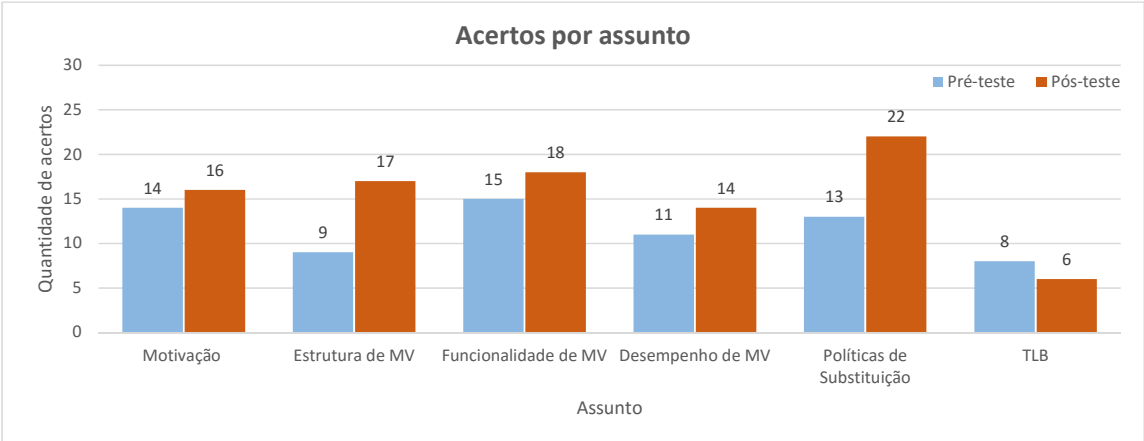


Figura 30 – Gráficos de acertos por assunto do primeiro experimento

Ainda na Figura 30, observa-se uma queda do pré-teste para o pós-teste na quantidade de acertos referente à TLB. Alguns fatores que podem explicar o ocorrido são o nível de dificuldade do pós-teste, dificuldade de entendimento do pós-teste, dificuldades pontuais ou falta de uma ênfase maior com o uso do REA nesses assuntos. No entanto, o experimento realizado não nos permite apontar com exatidão o principal causador desta queda pontual. Tal aspecto foi melhor investigado no próximo experimento.

O segundo experimento contou com a participação de 11 alunos que tiveram contato recente com o assunto de memória virtual. O pré-teste e pós-teste contêm 10 questões cada, relacionadas aos assuntos de políticas de substituição e TLB. Os assuntos abordados possuem

um conteúdo extenso que poderia ser melhor explorado com o auxílio do REA Amnesia, dando um foco maior no assunto de TLB, devido aos resultados não favoráveis obtidos no primeiro experimento. Além disso, houve algumas modificações nos planos de aula, melhorando a apresentação do assunto TLB.

Para este segundo experimento, somente 7 alunos participaram da aula com auxílio do REA Amnesia, pois a participação da aula com o REA foi voluntária. Isso nos permitiu realizar uma análise de dois cenários. O cenário 1 contou com 7 alunos que utilizaram o REA Amnesia; o cenário 2 contou com 4 alunos que não tiveram nenhuma aula (sobre o conteúdo memória virtual) entre os testes. A Figura 31 mostra a quantidade de questões respondidas de forma correta, errada e deixadas em branco para o segundo experimento nos dois cenários. Lembrando que foi recomendado aos alunos que deixassem as respostas em branco caso não tivessem certeza da resposta.

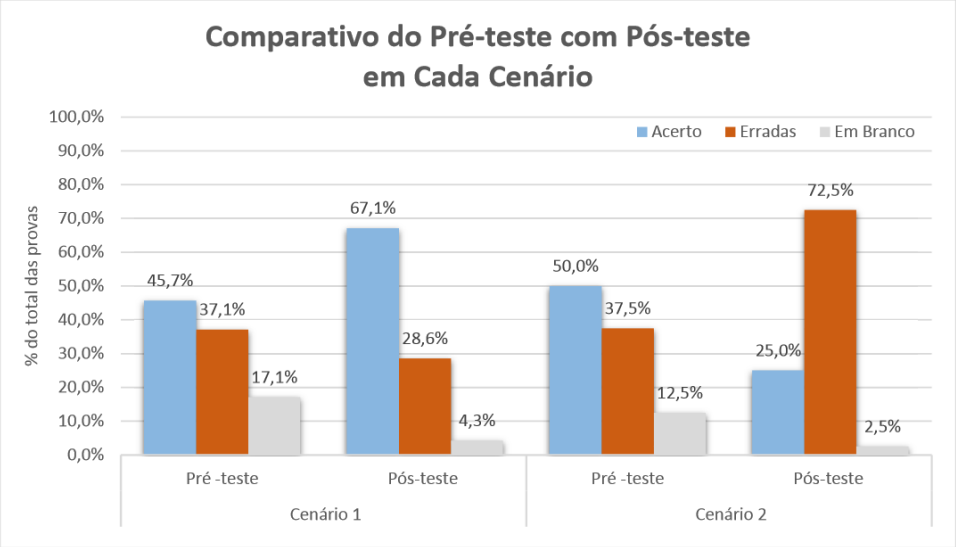


Figura 31 – Gráficos de respostas do segundo experimento

Podemos observar na Figura 31 que a quantidade de acertos no pré-teste dos alunos do cenário 1 é de 45,7%, no pós-teste houve uma melhora de 21,4% na quantidade de questões certas, e consequentemente, uma diminuição de 8,5% na quantidades de questões erradas do pré-teste (37,1%) para o pós-teste (28,6%). Além disso, ocorreu uma maior confiança no momento de responder as questões, pois somente 4,3% das questões do pós-teste foram deixadas em branco, contra 17,1% deixadas no pré-teste.

O cenário 2 apresentou um quadro inverso, em que a quantidade de questões certas no pré-teste foi de 50%, tendo uma queda no pós-teste de metade na quantidade questões certas ficando em 25%. Além disso, a quantidade de questões erradas foi de 72,5% no pós-teste, apresentado uma queda desempenho dos alunos. Seria muito interessante ministrar uma aula para esses alunos com ou sem o REA Amnesia, com o intuito de auxiliar a absorção do conteúdo de memória virtual.

Na Figura 32 podemos ver a distribuição dos resultados dos testes no gráfico *boxplox* do Cenário 1 e 2, com os resultados do pré-teste e pós-teste. No Cenário 1 podemos observar que a concentração das notas ficou entre 3 e 6 acertos com mediana em 4; já no pós-teste, a concentração ficou entre 6 e 7 acertos e com mediana em 7, ou seja, os alunos obtiveram uma melhora nos resultados do pós-teste com auxílio do REA Amnesia

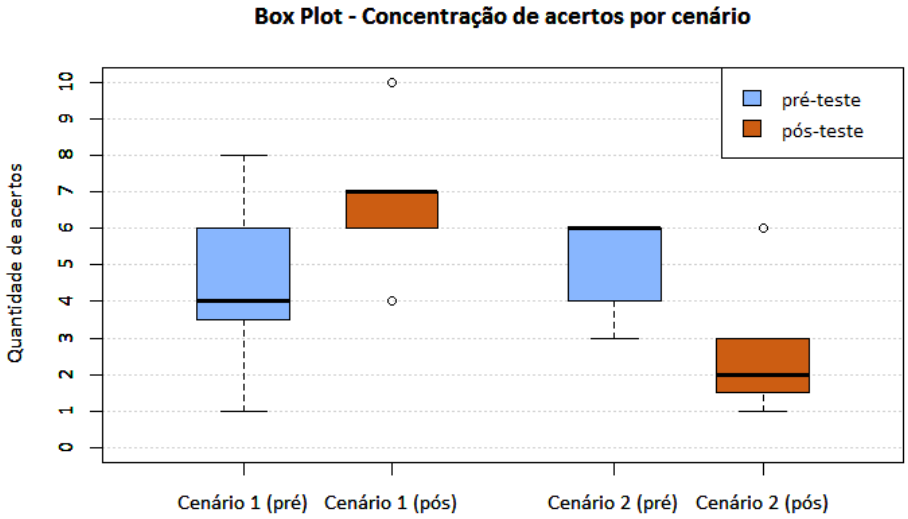


Figura 32 – *Boxplot* de acertos dos dois cenários do segundo experimento

Ainda na Figura 32 podemos ver que a menor e a maior nota do cenário 1 no pré-teste foram 1 e 8 respectivamente; no pós-teste podemos ver dois *outliers*, representando dois alunos que obtiveram a menor nota (4) e a nota máxima (10) respectivamente. Com os resultados foi constatado que 6 de 7 alunos ficaram com nota acima da média, mostrando assim uma melhora significativa nos resultados com auxílio do REA Amnesia.

No pré-teste do Cenário 2 a concentração das notas estava em 4 e 6 com mediana em 6, ou seja, as notas do pré-teste do cenário 2 estavam melhores do que as notas do pré-teste no cenário 1. No pós-teste a nota do cenário 2 ficou extremamente baixa entre 1 e 3 com mediana em 2 e somente com 1 aluno (nota 6) acima da média. Esse resultado mostra, que os alunos que não tiveram sequer uma aula teórica sobre o conteúdo de memória virtual, tiveram uma queda no desempenho.

Na Figura 33 são apresentados os resultados de acertos por assuntos. No cenário 1 podemos ver que houve uma melhora nos resultados das questões certas em relação aos assuntos abordados (políticas de substituição e TLB), essa melhora é mais evidente no resultado do assunto TLB, onde houve uma melhora de 37% na quantidade de questões certas do pré-teste para o pós-teste.

A melhora significativa no resultado do assunto TLB, é devido ao foco empregado na apresentação desse assunto neste experimento. Como no primeiro experimento tivemos um resultado desfavorável no assunto TLB (Figura 30), este experimento mostrou que um reforço no

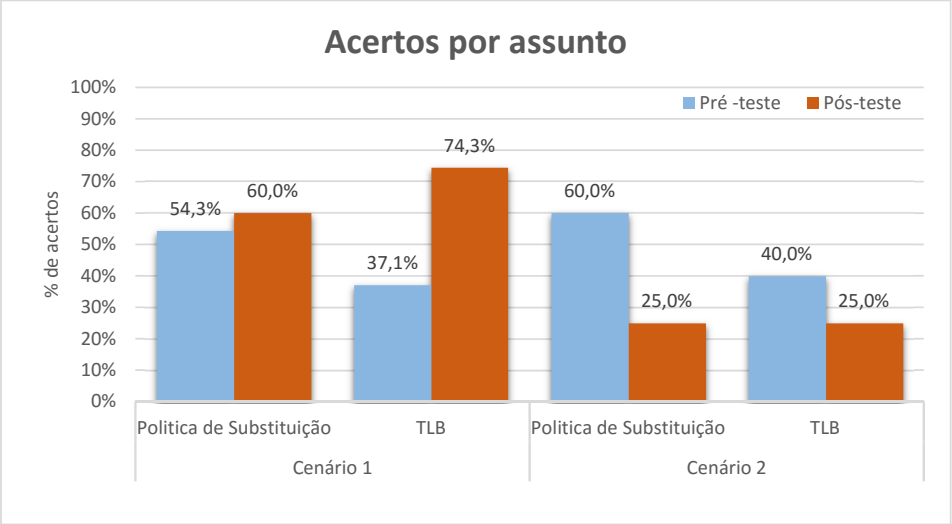


Figura 33 – Gráficos de respostas do segundo experimento

assunto com o auxílio REA pode apresentar resultados positivos. Isso mostra que para melhorar de fato o aprendizado dos alunos, não basta apenas usar um REA; deve-se planejar o uso e empregar o REA de maneira cuidadosa que os objetivos do ensino sejam alcançados.

Ainda na Figura 33, quando comparados o pré-teste e pós-teste do cenário 2, podemos observar uma queda nos resultados tanto de políticas de substituição como para TLB, mais acentuada no caso do assunto políticas de substituição, que apresentou uma queda de 35% do pré-teste para pós-teste. A queda no resultado do assunto TLB foi de 15% do pré-teste para o pós-teste.

A quantidade reduzida de alunos do primeiro e segundo experimento e os resultados negativos no cenário 2 do segundo experimento motivaram a realização do terceiro experimento. O terceiro experimento teve a participação de quarenta e nove alunos, o que nos permitiu realizar um experimento diferenciado dos anteriores. O pré e o pós-teste possuíam 10 questões cada, relacionadas aos assuntos de políticas de substituição e TLB. A nota do pré-teste serviu para realizar a separação dos alunos em grupos. O grupo A (Alto) foi formado por alunos que tiveram de 10 e 9 acertos; grupo B (Boa) por alunos que tiveram 8 e 7 acertos; grupo C (Média) por alunos que tiveram 6 e 5 acertos; e grupo D (Baixo) por alunos que tiveram 4 acertos ou menos. A distribuição dos alunos nos grupos ficou desta maneira:

- Grupo A (Alto): 6 alunos
- Grupo B (Boa): 24 alunos
- Grupo C (Média): 12 alunos
- Grupo D (Baixo): 7 alunos

Os grupos foram divididos em duas turmas sendo vinte e cinco alunos para Turma 1 e vinte e quatro alunos para Turma 2. Ressalta-se quatro alunos que não participaram do pós-teste, sendo assim, os testes e questionários desses alunos foram desconsiderados na fase de análise dos resultados. A turma 1 teve três alunos e a turma 2 teve um aluno que não participaram do pós-teste. Para análise dos resultados a separação dos grupos e turmas ficou da seguinte forma:

- Grupo A1 (Turma 1 – Alta): 2 alunos (1 faltante).
- Grupo A2 (Turma 2 – Alta): 3 alunos.
- Grupo B1 (Turma 1 – Boa): 12 alunos.
- Grupo B2 (Turma 2 – Boa): 12 alunos.
- Grupo C1 (Turma 1 – Média): 6 alunos.
- Grupo C2 (Turma 2 – Média): 6 alunos.
- Grupo D1 (Turma 1 – Baixa): 2 alunos (2 faltantes).
- Grupo D2 (Turma 2 – Baixa): 2 alunos (1 faltante).

A Turma 1 contou com a participação de vinte e dois alunos e a Turma 2 contou com a participação de 23 alunos. A Turma 1 teve a aula em laboratório onde cada aluno em seu respectivo computador acompanhou a aula com o auxílio do REA Amnesia. A Turma 2 teve uma aula teórica de forma tradicional onde o professor utilizou recursos como giz e quadro, sem o auxílio do REA Amnesia. Após três dias da realização das aulas foi aplicado o pós-teste, para evitar que o experimento sofresse influência do conteúdo que foi recém revisto.

Na Figura 34 é realizado um comparativo dos resultados obtidos no pré-teste e pós-teste por cada turma. Observa-se que as duas turmas obtiveram resultados positivos sendo que no pré-teste as duas turmas acertaram 68% de questões, mas no pós-teste a Turma 1 obteve 81% de acerto e a Turma 2 obteve 78% de acerto, uma evolução de 20,2% da Turma 1 contra 13,9% da Turma 2. Embora a evolução da Turma 1 tenha sido maior, a diferença entre elas não foi considerada significativa.

Na Figura 35 observa-se a distribuição das notas no gráfico *boxplot*. Com a realização a separação dos alunos nas duas turmas a concentração das notas do pré-teste foram similares, somente com um diferencial de dois *outliers*, na Turma 1 um aluno com a nota 1 e na Turma 2 um aluno com nota 2.

Na Figura 35 as diferenças entre o resultado do pós-teste da aula com Amnesia e o resultado da aula teórica são pequenas. Podemos ver que a concentração de 50% das notas da Turma 1 ficou entre 7,5 a 9 e na aula teórica ficou entre 7 e 9, com mediana em 8 nas duas turmas.

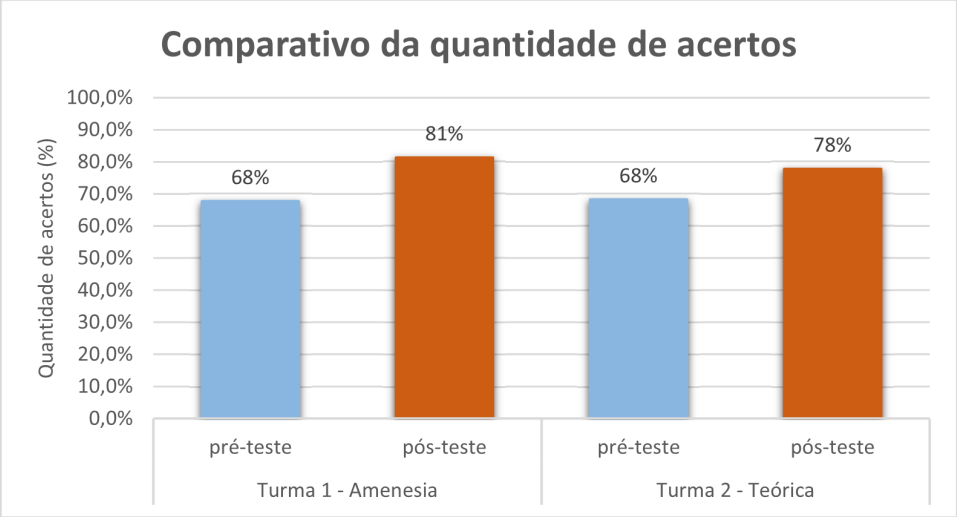


Figura 34 – Gráficos de respostas por aulas do terceiro experimento

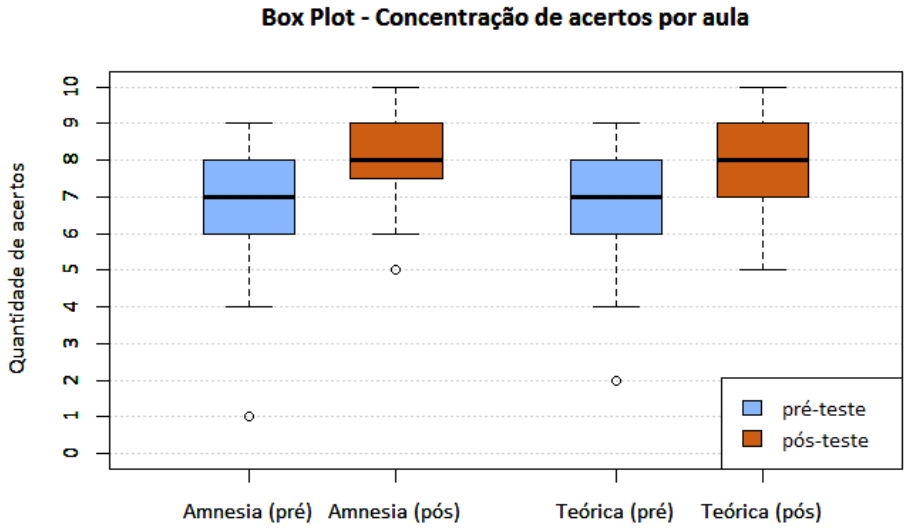


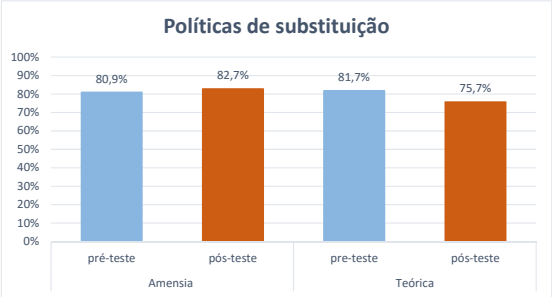
Figura 35 – *boxplot* de acertos por aulas do terceiro experimento

Além disso, existe um *outlier* com nota 5 no resultado do pós-teste da Turma 1; a Turma 2 teve mais de um aluno com nota 5.

Na Figura 36(a) observam-se os resultados de questões certas no assunto políticas de substituição, realizando uma comparação do pré-teste e pós-teste tanto para os alunos que participaram da aula com o Amnesia como para a aula tradicional. Comparação similar é apresentada na Figura 36(b) para os resultados de questões certas no assunto TLB.

Na Figura 36(a) identifica-se que a aula com auxílio do REA Amnesia obteve uma melhora de 2% no entendimento no assunto de políticas de substituição, a aula tradicional teve uma leve queda de 6%, possivelmente causada pela falta de recursos na apresentação desse assunto, ou porque os alunos da aula tradicional tiveram dificuldades de entender o funcionamento de alguma das políticas de substituição apresentadas.

(a) Acertos no assunto política de substituição



(b) Acertos no assunto TLB

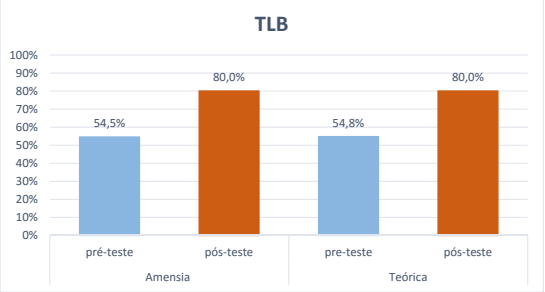


Figura 36 – Comparativo do total de acertos por assunto para cada Turma

Na apresentação do assunto política de substituição, é necessário apresentar parte da estrutura da memória virtual e as funcionalidades de cada política de substituição de páginas, focando em mostrar o impacto no desempenho causado por cada política de substituição. Muitas das vezes, uma troca de página errada ou uma contabilização errada apresenta resultados incoerentes, confundindo o entendimento dos alunos em determinada política de substituição.

Na Figura 36(b) percebe-se que as duas turmas obtiveram uma melhora nos resultados do assunto TLB. Os alunos da Turma 1 e 2 no pré-teste estavam com 55% de questões certas e no pós-teste 80% das questões certas. Para apresentar esse assunto é necessário apresentar a estrutura da TLB similar à estrutura da memória cache, os conceitos funcionais de tradução de endereço e a melhora no desempenho devido à utilização da TLB.

Na Figura 37 observam-se as médias de acertos nos diferentes grupos, realizando uma comparação entre os grupos de cada turma (1 e 2). Observa-se que todos os grupos obtiveram uma melhora na média das notas, com exceção do grupo A1, que permaneceu com a mesma média, sendo uma das maiores média entre os outros grupos.

Ainda na Figura 37 é apresentada uma grande evolução dos Grupos D1 e D2 onde as médias melhoraram 180% e 100% respectivamente, realizando um resgaste desses alunos que estavam abaixo da média. Estes resultados mostram a importância dos recursos educacionais para os estudantes que enfrentam dificuldades de aprendizagem e dependem de uma boa aula (em outras palavras, uma aula mais dinâmica, interativa e atraente).

7.4 Avaliação qualitativa

A avaliação qualitativa é composta de dois questionários, o primeiro questionário de expectativa de uso do REA, aplicado antes da utilização do REA. Os alunos apresentam qual é o seu grau de conhecimento no assunto de memória virtual, o que esperam do REA Amnesia, se já tiveran contato com algum outro REA anteriormente ao experimento, entre outras questões.

O segundo questionário de reação ao uso foi aplicado após a utilização do REA, com o qual os alunos descreveram suas experiências com o REA Amnesia. Os alunos são questionados

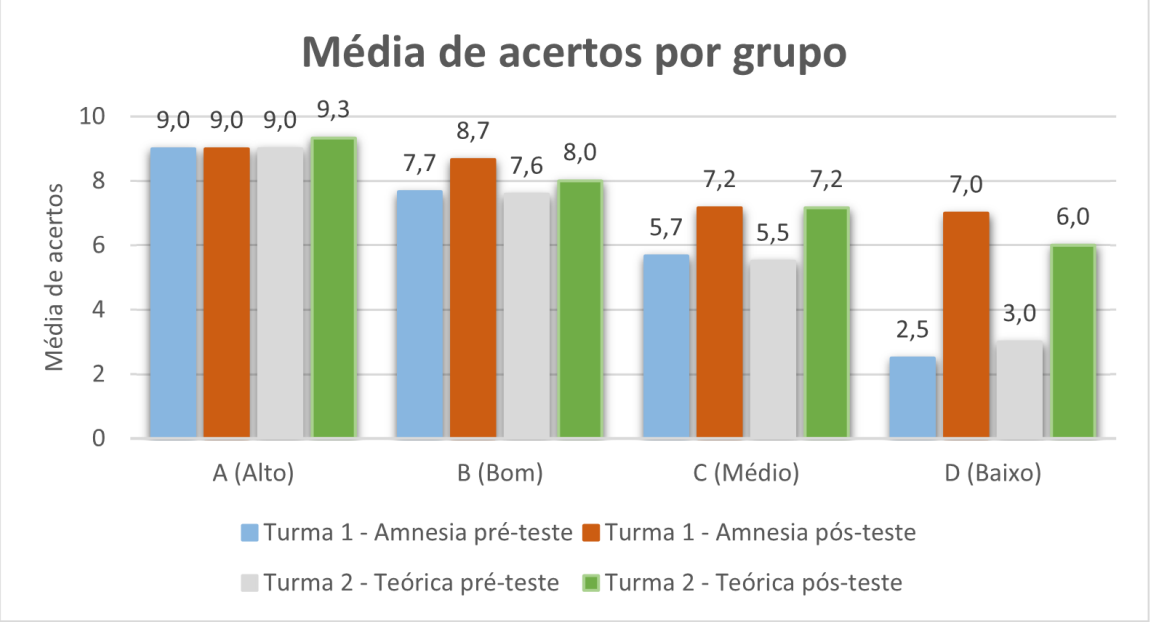


Figura 37 – Media de acertos por grupo

em relação ao que eles acreditam que seu conhecimento sobre o assunto de memória virtual foi melhorado, se suas expectativas foram atendidas, se o aluno utilizaria o REA novamente, entre outras questões. Além disso, perguntou-se aos alunos se eles se depararam com alguma falha no Amnesia, ou se encontraram dificuldade de utilização.

Somente os alunos que utilizaram o REA Amnesia responderam aos questionários dessa avaliação, sendo assim, há um total de 38 participantes. O primeiro experimento contou com a participação de 9 alunos, o segundo experimento teve a participação de 7 alunos e no terceiro experimento há a participação de 22 alunos na avaliação qualitativa. A identificação dos alunos era opcional nessa avaliação para evitar que os alunos se sentissem constrangidos em dar a sua opinião em algumas questões.

Os resultados da avaliação qualitativa composta pelos questionários de expectativa e reação ao uso são apresentados nas Tabelas 4 e 5. Após a realização do primeiro experimento foi realizada uma alteração no questionário de expectativa de uso adicionando 3 questões (Questões 6, 7 e 8). Com a alteração, as novas questões abordaram o conhecimento dos alunos sobre a matéria de memória virtual, conhecimento de outros REAs em quais outras matérias o aluno gostaria de usar um REA. As novas questões foram aplicadas no segundo e terceiro experimentos.

Tabela 4 – respostas do questionário de expectativa de uso

1) O que você acha da ideia de ter à disposição um software ou Recurso Educacional Aberto (REA) em qualquer lugar e hora para poder resolver algum problema ligado à disciplina?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Muito Bom	89%	71%	86%

Bom	11%	29%	14%
Regular	0%	0%	0%
Ruim	0%	0%	0%
Péssimo	0%	0%	0%
2) Quanto ao uso de REA para auxiliar o conhecimento,você acha que:			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Vai melhorar o seu conhecimento	89%	71%	95%
Vai permitir a voc saber pouca coisa a mais	11%	29%	5%
Vai deixar você no mesmo nível de conhecimento	0%	0%	0%
Não vai fazer diferença	0%	0%	0%
3) Na sua opinião, o uso de um REA pode despertar o interesse do usuário sobre o assunto?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	44%	57%	59%
Parcialmente	44%	43%	36%
Não	0%	0%	0%
No se aplica ou no sabe	11%	0%	5%
4) Na expectativa de uso de um REA, você se sente:			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Temeroso	0%	14%	0%
Indiferente	22%	14%	5%
Motivado	44%	29%	41%
Confiante	33%	43%	55%
5) Na sua opinião, o REA vai permitir o desenvolvimento de um conhecimento novo?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	56%	57%	50%
Parcialmente	33%	14%	36%
Não	0%	14%	5%
Não se aplica ou não sabe	11%	14%	9%
6) Você já conhece o assunto que será abordado pelo REA Amnesia, isto é, memória virtual?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Não conheço o assunto.		0%	0%
Parcialmente		86%	64%
Sim, já conheço.		14%	36%
7) Você já usou algum REA antes?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Nunca		71%	73%
Sim		29%	27%

8) Você gostaria de usar REA em outros contextos e/ou assuntos que irá estudar?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim		71%	68%
Parcialmente		14%	9%
Não		0%	0%
Não se aplica ou não sabe		14%	23%

Tabela 5 – respostas obtidas nos questionários de reação ao uso

1) O seu conhecimento sobre o conteúdo de memória virtual antes de utilizar o Recurso Educacional Aberto (REA) Amnesia era:			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Muito Bom	11%	0%	0%
Bom	33%	43%	64%
Regular	56%	57%	32%
Ruim	0%	0%	5%
Péssimo	0%	0%	0%
2) O REA Amnesia é compreensível e de fácil, manuseio?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	67%	0%	27%
Parcialmente	22%	71%	68%
Não	0%	29%	5%
No se aplica ou no sabe	11%	0%	0%
3) Você considera, em relação ao grau de conhecimento adquirido com o uso do REA Amnesia, que saiu:			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sabendo menos do que antes	0%	0%	5%
Sabendo a mesma coisa que antes	11%	0%	5%
Sabendo pouca coisa a mais	33%	71%	77%
Sabendo muito mais do que antes	56%	29%	14%
4) Em relação à motivação de utilizar REA Amnesia para auxiliar o conhecimento de memória virtual, pode ser considerada que:			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Aumentou	78%	29%	45%
Permaneceu igual a quando você não conhecia o REA	11%	0%	23%
Diminuiu	0%	14%	5%
Sofreu momentos de altos e baixos.	11%	57%	27%

5) Na sua opinião, o uso do REA Amnesia despertou o interesse sobre os conteúdos de memória virtual contidos nele?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	78%	29%	50%
Parcialmente	11%	43%	36%
Não	0%	29%	14%
Não se aplica ou não sabe	11%	0%	0%
6) A aula se tornou mais interessante com o uso do REA Amnesia?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	67%	57%	41%
Parcialmente	22%	14%	45%
Não	0%	29%	14%
Não se aplica ou não sabe	11%	0%	0%
7) A atividade é apropriada e o uso do REA Amnesia facilitou a sua compreensão sobre o conteúdo de memória virtual?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	67%	29%	59%
Parcialmente	22%	57%	36%
Não	0%	14%	5%
Não se aplica ou não sabe	11%	0%	0%
8) As respostas de todas as operações realizadas no REA Amnesia estavam de acordo com suas expectativas?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	44%	43%	64%
Parcialmente	56%	0%	18%
Não	0%	14%	9%
Não se aplica ou não sabe	0%	43%	9%
9) Você se sente seguro quanto aos resultados obtidos no REA Amnesia?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	67%	57%	82%
Parcialmente	11%	14%	9%
Não	0%	29%	5%
Não se aplica ou não sabe	22%	0%	5%
10) Está confiante e saberá usar o conhecimento adquirido pelo REA Amnesia em uma futura prática?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	33%	29%	59%

Parcialmente	0%	57%	36%
Não	0%	14%	5%
Não se aplica ou não sabe	67%	0%	0%
11) O uso do REA Amnesia aumentou o nível de seu conhecimento no conteúdo de memória virtual?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	67%	29%	41%
Parcialmente	22%	71%	55%
Não	11%	0%	5%
Não se aplica ou no sabe	0%	0%	0%
12) Na sua opinião, sobre o REA Amnesia			
Qual a relevância:	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Muito Bom	33%	43%	36%
Bom	67%	43%	64%
Regular	0%	14%	0%
Ruim	0%	0%	0%
Péssima	0%	0%	0%
Qual a adequação do conteúdo memória virtual:	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Muito Bom	44%	29%	45%
Bom	56%	43%	50%
Regular	0%	14%	5%
Ruim	0%	14%	0%
Péssima	0%	0%	0%
Qual a facilidade na compreensão das atividades:	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Muito Bom	33%	0%	14%
Bom	67%	43%	45%
Regular	0%	43%	36%
Ruim	0%	14%	5%
Péssima	0%	0%	0%
13) Você usaria o REA Amnesia novamente?			
	Exp. 1 (%)	Exp. 2 (%)	Exp. 3 (%)
Sim	78%	57%	77%
Parcialmente	11%	0%	14%
Não	0%	43%	9%
Não se aplica ou não sabe	11%	0%	0%

No primeiro experimento, os resultados apresentados no questionário de expectativa de uso (Tabela 4) mostra que 44% dos alunos estavam motivados em utilizar o REA Amnesia e

33% estavam confiantes. Além disso, 89% acreditavam que seu conhecimento no conteúdo iria melhorar, porém, só 44% acreditava no aumento do interesse pelo assunto. A provável falta de interesse pelo assunto se deve ao fato de que eram alunos de pós-graduação que realizam trabalhos de pesquisa em outras áreas. Além disso, 89% achou muito boa a ideia de ter um REA sempre à disposição para resolver problemas ligados às disciplinas.

Os resultados do questionário de reação ao uso (Tabela 5) no primeiro experimento mostram que 88% dos alunos notou que houve um reforço de seu conhecimento no conteúdo de memória virtual, sendo que 55% dos alunos relataram que seu conhecimento aumentou com o uso do REA e 33% disseram que tinham aprendido algo a mais. Além disso, 77% afirmaram que a motivação em utilizar o Amnesia aumentou depois do seu uso e ainda que despertou interesse sobre o conteúdo. A motivação em utilizar o REA se manteve e ajudou no conhecimento dos alunos, refletindo, consequentemente, na melhora dos resultados do pós-teste. Na opinião dos alunos, o REA Amnesia mostrou-se relevante, com conteúdo adequado e de fácil compreensão, de modo que 77% disseram que o utilizariam novamente.

O questionário de reação ao uso permitiu que os alunos informassem falhas ocorridas nas simulações do REA Amnesia. As falhas que foram apresentadas são:

- O tutorial de utilização não abria no SO Linux.
- No OS X, quando era aberto um arquivo de arquitetura, uma janela de escolha de arquivos "extra" aparecia, tendo que ser fechada.

Como os testes foram realizados em uma máquina com o SO *Windows*, esses detalhes não foram observados previamente nos testes. As correções foram realizadas e nenhuma outra falha foi relata nos outros experimentos.

No segundo experimento os resultados apresentados no questionário de expectativa de uso (Tabela 4) mostram que 43% dos alunos estavam confiantes em utilizar o REA Amnesia e 33,3% estavam motivados. Além disso, 71% acreditavam que seus conhecimentos iriam aumentar, porém, só 57% acreditavam no aumento do interesse pelo assunto. 86% dos alunos declararam conhecer parcialmente o assunto, ou seja, era possível melhorar o conhecimento desses alunos. Além disso, 71% deles nunca utilizaram um REA e deram sugestões de matérias e assuntos que poderiam utilizar REA para facilitar o aprendizado. Algumas das sugestões foram: computação paralela, algoritmos de semáforos, mutex, multiprocessadores, escalonamento de tarefas, circuitos digitais e tratamento de interrupções.

No questionário de reação ao uso (Tabela 5), 71% dos alunos relataram que saíram sabendo um pouco a mais, e 29% disseram que seus conhecimentos (sabendo mais do que antes) aumentaram com a utilização do REA. Ou seja, 100% dos alunos notaram um reforço nos seus conhecimentos no conteúdo de memória virtual. Além disso, 57% disseram que a aula se tornou mais interessante com o uso REA Amnesia e 57% se sentiram seguros quanto aos resultados

obtidos com REA Amnesia. Sendo assim, o REA atendeu grande parte das expectativas dos alunos nesse experimento.

Os resultados apresentados no questionário de expectativa de uso (Tabela 4) mostraram que 95% dos alunos acreditava na melhoram dos seus conhecimentos. Além disso, 59% acreditavam no aumento do interesse pelo assunto. Mesmo 64% dos alunos argumentaram que conheciam parcialmente o assunto. Um dado interessante apresentado é que 27% dos alunos já tinham utilizado anteriormente algum REA. Os alunos destacaram alguns dos REA utilizados anteriormente, como codeAcademy, karel *the robot*, JFLAP, JDSP, entre outros. Alguns alunos descreveram os REA utilizados, algumas das descrições foram:

- "Utilizei um REA em redes, que mostrava a diferença ente *Go-back-N* e repetição seletiva."
- "Simulador de algoritmos de ordenação e comunicação via TCP."
- "Simuladores de algoritmos de ordenação, estrutura de dados (Árvore), etc."
- "Simulador de transmissão de pacotes TCP."

Ainda no terceiro experimento no questionário de expectativa de uso, 68% dos alunos apresentaram temas em que poderiam ser adotados REA, como redes, arquiteturas de computadores, Arquivos, *Threads*, memória cache, árvores, busca em grafos, microeletrônica, entre outros.

No questionário de reação ao uso (Tabela 5) do terceiro experimento, 64% dos alunos relataram que seus conhecimentos antes de utilizar o Amnesia era bom e 32% afirmaram que eram regulares. 77% dos alunos disseram que saíram sabendo pouca coisa a mais e 14% disseram que saíram sabendo muito mais do que antes. Ou seja, 91% dos alunos disseram que o Amnesia ajudou a reforçar o conhecimento do conteúdo de memória virtual. Isso pode ser evidenciado na análise dos resultados do terceiro experimento (Figura 34) onde houve uma evolução de 13%. 59% dos alunos consideraram a atividade apropriada para o uso do REA e estavam confiantes em utilizar os conhecimentos adquiridos com REA Amnesia em futuras práticas. 77% apontaram que utilizariam o REA Amnesia novamente.

Nos três experimentos, o questionário de reação ao uso permitiu indicar sugestões de alterações. As sugestões apresentadas foram analisadas verificando a possibilidade da alteração antes de cada novo experimento ou antes da disponibilização do REA no site. Algumas das alterações sugeridas foram:

1. Utilizar cores e *highlights* nos campos para fixar a atenção dos alunos.
2. Implementação do botão *back* no qual seria possível retornar às execuções anteriores.
3. Animações durante a execução das instruções.

4. Explorar mais algoritmos de substituição de páginas nas arquiteturas.

5. Colocar dicas quando passar o *Mouse* por cima dos componentes.

6. Opção de criação de arquivos de arquitetura sem ter que manipular o arquivo XML, apenas escolhendo as configurações das memórias em um componente.

7. Adicionar o número da *Virtual Page* no componente RAM (essa informação é apresentada no componente *Page Table*, mas facilitaria aparecer nos dois componentes).

A sugestão (1) foi apresentada no primeiro experimento e as alterações foram realizadas e utilizadas nos experimentos seguintes. As sugestões (2), (3), (4) e (5) foram apresentadas no segundo experimento e não houve tempo hábil ainda para realizar mais modificações. A sugestão (6) foi apresentada no terceiro experimento. O módulo de criação de arquiteturas existe no REA Amnesia, mas ainda apresenta falhas, então é necessário passar por uma correção de defeitos antes da liberação do componente. A sugestão (7) também foi apresentada no terceiro experimento, mas essa modificação envolve uma alteração nas dimensões dos componentes, podendo danificar a propriedade de resolução de tela mínima de 1024 x 768 *pixels*.

7.5 Considerações finais

Nos três experimentos realizados, observou-se uma evolução na aprendizagem do conteúdo de memória virtual com a realização de uma avaliação quantitativa. Cada experimento teve a sua particularidade na abordagem da aplicação. O primeiro experimento apresentou todo o conteúdo de memória virtual abordado pelo REA Amnesia, o segundo experimento teve foco em dois assuntos de memória virtual e no terceiro foi realizada uma comparação com uma aula tradicional.

Com as abordagens diferenciadas nos três experimentos foi possível obter resultados relevantes para o REA Amnesia. O primeiro experimento mostrou que o REA Amnesia ajuda os alunos a relembrar os assuntos de memória virtual (os resultados mostraram uma melhora de 28,8%). O segundo experimento apresentou um reforço de 21% no conhecimento dos alunos que utilizaram o REA Amnesia, contra uma queda de 50% no conhecimento dos alunos que não tiveram nenhuma aula entre os testes. Com o terceiro experimento foi possível obter dados mais concisos, onde verificou-se uma evolução de até 180% no conhecimento dos alunos que mais necessitavam de reforço no aprendizado.

A avaliação qualitativa mostrou que os alunos estavam motivados em utilizar o REA Amnesia e que esse interesse se manteve após a realização dos experimentos. Por meio destes questionários os alunos apresentaram suas opiniões sobre o REA Amnesia, além de apresentarem outras matérias e assuntos que seriam interessante de serem utilizados REA. Além disso,

apontaram sugestões de pontos estes que poderiam ser melhorados no REA Amnesia, pontos estes que não foram observados no processo de manutenção do simulador (Seção 6.2).

CAPÍTULO

8

CONCLUSÕES

8.1 Conclusão geral

Existem muitos desafios que devem ser enfrentados pelos professores de Sistemas Operacionais que desejarem ensinar o conteúdo de memória virtual de maneira aprofundada. Tais desafios relacionam-se à dificuldade do conteúdo, que envolve aspectos estruturais, funcionais e de desempenho. O módulo memória virtual do simulador Amnesia apresenta-se como uma alternativa, o qual auxilia o ensino de memória virtual simulando aspectos essenciais desse conteúdo.

Por acreditar que o simulador Amnesia não deveria ficar restrito a somente uma instituição de ensino, buscaram-se alternativas para a divulgação do simulador, como uma ferramenta didática. Dentre as soluções pesquisadas, optou-se pelo desenvolvimento de um REA. O conceito de REA fornece os direitos de adaptá-los, combiná-los e compartilhar cópias, além das vantagens de reutilização, uma vez que REA é a evolução direta de Objetos de Aprendizagem (OA).

Antes da utilização e disponibilização do Amnesia foi necessária a realização de um conjunto de testes funcionais para torná-lo um software robusto. Com a realização dos testes funcionais foi possível identificar falhas antes da aplicação com os alunos. Além disso foram realizadas mudanças no código fonte do Amnesia, melhorando a sua legibilidade. Ainda foram realizadas mudanças na interface do Amnesia, tornado-o mais intuitivo para os alunos.

Após as mudanças no código fonte do simulador Amnesia foi realizada a inclusão de uma licença de software (GPL) e a disponibilização do Amnesia. Sendo assim, o simulador Amnesia pode ser considerado um REA para o ensino de hierarquia de memória e memória virtual. O REA Amnesia demonstra o funcionamento das memórias caches, principal e virtual de forma didática, com o objetivo de facilitar e melhorar a aprendizagem nas disciplinas que apresentem esse conteúdo.

Foram desenvolvidos planos de aulas, manual de utilização do REA e texto com conteúdo

teórico, que também podem ser considerados REA disponibilizados juntamente com Amnesia. Foi criado um site para realizar a disponibilização de todos os REAs desenvolvidos nesse projeto e os OA desenvolvidos no projeto de mestrado de Tiosso (2015).

As avaliações realizadas no Amnesia tiveram o objetivo de verificar a eficácia do REA no ensino dos assuntos referentes ao conteúdo de memória virtual. Foram realizados três experimentos com abordagens distintas com alunos de graduação e pós-graduação. Os experimentos possuem uma base experimental similar, composta de uma avaliação quantitativa e uma qualitativa, além de uma aula com o auxílio do REA Amnesia.

Com o resultado da avaliação quantitativa foi possível observar que houve um reforço no conhecimento dos alunos em relação aos assuntos de memória virtual. Os alunos mostraram um melhor rendimento após a utilização do REA, sedimentando o aprendizado nesse assunto. No primeiro experimento o REA Amnesia ajudou os alunos a relembrarem o conteúdo que foi visto previamente. O segundo experimento ajudou a sedimentar o conteúdo de memória virtual já abordado em sala.

O terceiro experimento permitiu obter dados mais concisos, devido ao maior número de alunos que participaram do mesmo. Foi realizado um experimento dividindo os alunos em duas turmas, onde uma turma teve a aula com o REA Amnesia e outra teve uma aula da forma tradicional. Os resultados mostraram que a aula com o REA contribuiu significativamente para o ensino e a aprendizagem de alunos, apresentado ganhos em até 180% nos acertos de alunos com notas baixas.

Com a avaliação qualitativa foi possível observar se as expectativas dos alunos foram atendidas com utilização do REA Amnesia. Os alunos se mostraram confiantes em utilizar o REA, confirmando que o uso do REA atendeu suas expectativas (73% dos alunos disseram que utilizariam o REA novamente). Nessa avaliação, os alunos apresentaram suas sugestões e falhas identificadas no REA, contribuindo com o aprimoramento do Amnesia.

As aulas com o REA Amnesia podem ser consideradas mais dinâmicas, interativas e atraentes para os alunos. Em aulas no estilo tradicional, os alunos são condicionados a anotar o que professor fala ou escreve no quadro, dificultando o entendimento dos aspectos referentes ao assunto de memória virtual. Um ponto importante é que as aulas com o REA Amnesia tendem a ser mais longas, pois inicialmente deve ser apresentado o funcionamento do REA, antes da apresentação dos assuntos de memória virtual. As aulas com o REA Amnesia também exigem um planejamento cuidadoso, onde os planos de aula oferecidos juntamente com REA pretendem facilitar essa tarefa.

O intuito do desenvolvimento do REA Amnesia é oferecer uma ferramenta para auxiliar o professor no ensino de hierarquia de memória e memória virtual. Como são conteúdos que devem ser apresentados de forma dinâmica, é extremamente importante ter à disponibilidade uma ferramenta que auxilie o professor na aplicação dessa tarefa. Os professores podem utilizar

laboratórios de computadores para facilitar o controle da apresentação do REA Amnesia, mas essa abordagem fica a critério do professor. Os alunos podem obter o REA Amnesia para reforçar os conhecimentos aprendido em sala em atividades extraclasse.

8.2 Contribuições

A principal contribuição deste trabalho de mestrado é o desenvolvimento e a disponibilização de um REA para o ensino do conteúdo de memória virtual. Os REAs do projeto Amnesia auxiliam tanto os professores, com a disponibilização dos planos de aula, quantos os alunos que aprendem de uma forma mais dinâmica o conteúdo de memória virtual.

Os experimentos realizados neste trabalho conseguiram melhorar o conhecimento dos alunos nos assuntos relacionados à memória virtual. Os resultados positivos obtidos nos experimentos mostram que esses alunos conseguiram absorver o conteúdo ministrado e estavam confiantes em utilizar os conhecimentos adquiridos com REA Amnesia em futuras práticas.

Uma contribuição indireta deste trabalho foi a criação de novos REAs, pelos alunos que participaram do segundo e terceiro experimento. Esses alunos estavam cursando disciplinas de Sistemas Operacionais na graduação e pós-graduação, e desenvolveram REAs de diferentes assuntos relacionados a SO. Tais REA foram disponibilizados na web[1], onde também se pode ter acesso ao REA Amnesia. Um aspecto interessante dessa experiência foi verificar que alguns dos alunos que participaram do desenvolvimento desses REA relataram que o Amnesia foi a inspiração para seu trabalho.

8.3 Produção científica

Este trabalho de mestrado possibilitou a produção de três artigos. O primeiro artigo produzido com o título "Amnesia: Um Recurso Educacional Aberto para o Ensino de Memória Virtual", foi aceito para o *Congreso Chileno de Educación Superior en Computación* (CCESC 2015) e deverá ser apresentado em novembro.

O segundo artigo produzido foi submetido para o *ACM Symposium On Applied Computing* (ACM SAC 2016) *Track on Intelligent, Interactive and Innovative Learning Environments*, com o título *"An Interactive Approach for Teaching of Virtual Memory Using Open Educational Resources"*. Tal artigo encontra-se em avaliação no período em que esta dissertação está sendo escrita.

O primeiro artigo tem o foco no REA Amnesia e o segundo artigo tem o foco no ensino do conteúdo com auxílio do REA Amnesia. O primeiro artigo apresenta detalhes do módulo memória virtual do REA Amnesia e possui análise de dois experimentos (o primeiro e segundo

[1] http://rea.lasdpc.icmc.usp.br/pt/

experimentos do Capítulo 7). O segundo artigo relata nossa experiência no ensino de memória virtual com o apoio REA Amnesia, para uma aula mais interativa com os alunos. A análise dos resultados com o terceiro experimento também é apresentada nesse artigo.

Outro trabalho também produzido durante o desenvolvimento deste projeto de mestrado foi *"Amnesia: a Learning Object for Memory Hierarchy Teaching"*, o qual foi aceito para o *Frontiers in Education* (FIE 2015). Neste trabalho aborda-se o uso do módulo memória cache do Amnesia.

8.4 Trabalhos futuros

Apesar de todos os experimentos realizados neste trabalho, ainda há a necessidade de novos experimentos com outras abordagens e metodologias de ensino com o suporte do Amnesia. Novos experimentos podem ser realizados por outras instituições e por outros pesquisadores, em colaboração ao projeto Amnesia, enriquecendo a pesquisa na área de educação em computação.

As sugestões de alterações (Seção 7.4) apresentadas pelos alunos que participaram dos experimentos devem ser analisadas com mais cuidado em trabalhos futuros. As sugestões do segundo e terceiro experimentos não foram implementadas ainda por falta de tempo hábil, mas são sugestões de grande valia.

Futuros projetos podem explorar o ensino com auxílio dos três módulos disponíveis no Amnesia, principalmente com módulo cache e memória virtual, pois são módulos robustos. O módulo CPU e as ferramentas auxiliares como Conversor de código *Assembly* MIPS e Gerador de arquivos de rastro ainda não foram utilizados com abrangência e profundidade. Inicialmente são necessárias correções nessas ferramentas de software para que seja possível utilizá-las no ensino. O módulo CPU e as ferramentas podem ainda ser utilizados nas disciplinas de linguagem *assembly*, arquitetura de computadores e organização de computadores.

Outra investigação a ser explorada no projeto Amnesia é voltada ao conhecimento individual dos alunos. Como os exercícios são desenvolvidos para reforçar e sedimentar o conhecimento de uma turma, identificou-se a possibilidade de utilizar o Amnesia adaptando-o para cada aluno que o utiliza, transformando-o em um Sistema Tutor Inteligente (STI). Através da inclusão de Sistemas Multiagentes (SMA) e da análise de diferentes necessidades e estilos de aprendizagem dos usuários, o conhecimento obtido poderia mensurado pelo Amnesia em cada atividade realizada, transformando-o em um objeto inteligente de aprendizagem (TIOSSO, 2015). Essa é a descrição de um projeto de doutorado já em andamento.

REFERÊNCIAS

ADL. **SCORM Users Guide for Instructional Designers**. 2011. Disponível em: <http://www.adlnet.gov/wp-content/uploads/2011/12/SCORM_Users_Guide_for_ISDs.pdf>. Citado na página 37.

ALHARBI, A.; HENSKENS, F.; HANNAFORD, M. Computer science learning objects. **International Conference on e-Education, Entertainment and e-Management**, n. September, p. 326–328, 2011. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6137817>. Citado na página 42.

ALMEIDA, R.; CHAVES, A.; COUTINHO, F. Avaliação da usabilidade e da qualidade do conteúdo de Objetos de Aprendizagem digitais sobre o sistema digestório. **II Simpósio Nacional de Ensino de Ciência e Tecnologia**, 2010. Disponível em: <http://www.sinect.com.br/anais2010/artigos/TIC/214.pdf>. Citado na página 40.

ARANTES, A. R.; MIRANDA, M. S.; STUDART, N. Objetos de aprendizagem no ensino de física: usando simulações do PhET. **Física na Escola**, p. 27–31, 2010. Disponível em: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Objetos+de+aprendizagem+no+ensino+de+f?sica:+usando+simula??es+do+PhET\#0>. Citado na página 42.

BACKUS, J. Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs. **Communications of the ACM**, 1977. Disponível em: <http://dl.acm.org/citation.cfm?id=359579>. Citado na página 47.

BARBOSA, E. F. **Módulos Educacionais e Ambientes de Ensino e Aprendizagem: Contribuições e Perspectivas**. 2014. Texto para obtenção do Título de Professor Livre Docente. Citado 2 vezes nas páginas 32 e 42.

BARBOSA, J.; FERNANDES, A. Objetos de Aprendizagem: análise de seu uso em uma sala de aula do ensino fundamental. **Anais do Workshop de Informática na Escola**, n. 2004, p. 1117–1126, 2010. Disponível em: <http://ceie-sbc.tempsite.ws/pub/index.php/wie/article/view/2034>. Citado na página 42.

BARKER, P.; CAMPBELL, L. M. Metadata for Learning Materials: An Overview of Existing Standards and Current Developments. **Technology, Instruction, Cognition and Learning**, v. 7, n. 3-4, p. 225–243, 2010. ISSN 15400182. Citado 2 vezes nas páginas 13 e 36.

BARONE, P. M. V. B. C. **Diretrizes Curriculares Nacionais para os cursos de graduacao em Computacao**. 2012. 27 p. Disponível em: <http://portal.mec.gov.br/index.php?option=com_docman&task=doc_download&gid=11205&Itemid=>. Citado na página 25.

BURBECK, S. **How to use Model-View-Controller (MVC)**. 1992. Disponível em: <http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html>. Citado na página 74.

BUZZETTO-MORE, N. An Examination of Undergraduate Student's Perceptions and Predilections of the Use of YouTube in the Teaching and Learning Process. **Interdisciplinary Journal of Knowledge and Learning Objects**, v. 10, p. 17–32, 2013. Disponível em: <http://www.ijello.org/Volume10/IJELLOv10p017-032Buzzetto0437.pdf>. Citado na página 39.

CARBONE, A.; LL, J. K. A survey of methods used to evaluate computer science teaching. **ACM SIGCSE Bulletin**, p. 41–45, 1998. Disponível em: <http://dl.acm.org/citation.cfm?id=283014>. Citado na página 43.

CASALI, A.; DECO, C.; ROMANO, A.; TOMé, G. An Assistant for Loading Learning Object Metadata : An Ontology Based Approach. v. 9, 2013. Citado na página 34.

COSTA, F. H. G. da. **Objeto de Aprendizagem para o ensino de Estruturas de Dados**. Brasília: [s.n.], 2011. Disponível em: <http://monografias.cic.unb.br/dspace/handle/123456789/315>. Citado na página 43.

COUTINHO, L. M. N.; MENDES, J. L. D.; MARTINS, C. A. P. d. S. Avaliação quantitativa do uso de um ambiente open-source de auxílio ao aprendizado de hierarquia de memória. **Workshop sobre eduacação em Computação**, p. 27–36, 2007. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2007/003.pdf>. Citado 4 vezes nas páginas 13, 79, 98 e 99.

CYBIS, W. d. A.; PIMENTA, M. S.; SILVEIRA, M. C.; GAMEZ, L. Uma Abordagem Ergonômica para o Desenvolvimento de Sistemas Interativos. p. 1–10, 1998. Citado na página 40.

DICKINSON, J. Operating systems projects built on a simple hardware simulator. **ACM SIGCSE Bulletin**, v. 32, n. 1, p. 320–324, mar. 2000. ISSN 00978418. Disponível em: <http://portal.acm.org/citation.cfm?doid=331795.331878>. Citado na página 26.

DJORDJEVIC, J. A Memory System for Education. **The Computer Journal**, v. 48, n. 6, p. 630–641, jun. 2005. ISSN 0010-4620. Disponível em: <http://comjnl.oxfordjournals.org/cgi/doi/10.1093/comjnl/bxh133>. Citado 2 vezes nas páginas 79 e 80.

DJORDJEVIC, J.; MILENKOVIC, A.; PRODANOVIC, S. A hierarchical memory system environment. **Proceedings of the 1998 Workshop on Computer Architecture Education**, 1998. Disponível em: <http://dl.acm.org/citation.cfm?id=1275205>. Citado 2 vezes nas páginas 79 e 80.

EDLER, J.; HILL, M. D. **Dinero IV Trace-Driven Uniprocessor Cache Simulator**. 1997. Disponível em: <http://pages.cs.wisc.edu/~markhill/DineroIV/>. Citado na página 79.

EPFL; KUL; PROJECT the A. **ARIADNE Educational Metadata Recommendation**. 1999. Disponível em: <http://vs.fernuni-hagen.de/methoden/ils/Organisation/ariadne.html>. Citado na página 34.

FABBRI, S. C. P. F.; VINCENZI, A. M. R.; MALDONADO, J. C. Teste funcional. In: ____. São Paulo, SP: Elsevier Editora Ltda, 2007. (In: Maldonado, J. C; Jino, M.; Delamaro, M. E. (Org.). Introdução ao Teste de Software.), p. 47–76. Citado na página 86.

FABRE, M.-C. J. M.; TAROUCO, L. M. R.; TAMUSIUNAS, F. R. SCORM (Sharable Content Object Reference Model). 2013. Citado 2 vezes nas páginas 13 e 37.

FERLIN, J.; KEMCZINSKI, A.; MURAKAMI, E.; HOUNSELL, M. d. S. Metadados Essenciais: Uma Metodologia para Catalogação de Objetos de Aprendizagem no Repositório Digital ROAI. **Anais do Workshop de Informática na Escola**, p. 1147–1156, 2010. Disponível em: <http://ceie-sbc.tempsite.ws/pub/index.php/wie/article/view/2037>. Citado na página 36.

FOTHERINGHAM, J. Dynamic storage allocation in the Atlas computer, including an automatic use of a backing store. **Communications of the ACM**, 1961. Disponível em: <http://dl.acm.org/citation.cfm?id=366800>. Citado na página 64.

FOUNDATION, F. S. **GNU General Public License**. 2014. Disponível em: <http://www.gnu.org/copyleft/gpl.html>. Citado na página 90.

FOURNIER-VIGER, P.; NAJJAR, M. A cognitive and logic based model for building glass-box learning objects. **Interdisciplinary Journal of Knowledge and Learning Objects**, v. 2, 2006. Disponível em: <http://www.editlib.org/p/44815/>. Citado na página 43.

FRIESEN, N.; INITIATIVE, C. Interoperability and Learning Objects : An Overview of E-Learning Standardization. v. 1, 2005. Citado na página 34.

GALAFASSI, F. P.; GLUZ, J. a. C.; GALAFASSI, C. Análise Crítica das Pesquisas Recentes sobre as Tecnologias Objetos Aprendizagem e Ambientes Virtuais de Aprendizagem. **Revista Brasileira de Informática na Educação**, v. 21, n. 03, mar. 2013. ISSN 1414-5685. Disponível em: <http://www.br-ie.org/pub/index.php/rbie/article/view/2353>. Citado 2 vezes nas páginas 32 e 33.

GAMA, C. D. Método de construção de objetos de aprendizagem com aplicação em métodos numéricos. 2007. Disponível em: <http://www.ppgmne.ufpr.br/arquivos/teses/9.pdf>. Citado 7 vezes nas páginas 23, 40, 181, 183, 184, 185 e 187.

GONZáLEZ, L. A. S.; ANJOS, E. S. dos; CUNDA, A. V. da. **Implementação de Objetos de Aprendizagem para Disciplinas de Análise Estrutural dos Cursos de Engenharia**. 2014. 1–4 p. Disponível em: <http://www.ufrgs.br/salao_ead_grad/salao2008/anais/anais_08_pdf/a7/implementacao_de_objetos_de_aprendizagem_para_disciplinas_de_analise_estrutural.pdf>. Citado na página 42.

HAMAWAKI, M. H.; PELEGRINI, C. D. M. As ferramentas do ensino a distancia e suas contribuições para a eficácia no processo de aprendizagem do aluno. **CEPPG**, 2009. Disponível em: <http://www.portalcatalao.com/painel_clientes/cesuc/painel/arquivos/upload/temp/b7632647fce4a8a50fda143156336f90.pdf>. Citado na página 27.

HENNESSY, J. L.; PATTERSON, D. A. **Computer Architecture, Fifth Edition: A Quantitative Approach**. 5th. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011. ISBN 012383872X, 9780123838728. Citado 2 vezes nas páginas 17 e 49.

HILL, M. D. **dinero - cache simulator, version III (Enhanced Version)**. 1989. Disponível em: <http://www.ece.cmu.edu/~ece548/tools/dinero/man/dinero.htmhttp://www.ece.cmu.edu/~ece548/tools/dinero/src/doc.h>. Citado na página 79.

HILL, M. D.; SMITH, A. J. Evaluating Associativity in CPU Caches. **IEEE Trans. Comput.**, IEEE Computer Society, Washington, DC, USA, v. 38, n. 12, p. 1612–1630, 1989. ISSN 0018-9340. Disponível em: <http://dx.doi.org/10.1109/12.40842>. Citado na página 75.

HYLéN, J. Open educational resources: Opportunities and challenges. **Proceedings of Open Education**, 2006. Disponível em: <http://library.oum.edu.my/oumlib/sites/default/files/file_attachments/odl-resources/386010/oer-opportunities.pdf>. Citado na página 42.

IMS Global Learning Consortium, I. IMS Global Learning Consortium Intellectual Property Rights Policy. p. 1–13, 2008. Citado na página 34.

JESUS, A. N. a. D.; LOPES, D. L.; PERIN, F. R.; aO, J. M. C.; PIMENTEL, E. P. OBJETOS DE APRENDIZAGEM NO ENSINO DE LÓGICA DE PROGRAMAÇÃO. **Informática Aplicada**, III, p. 37–42, 2007. Disponível em: <http://seer.uscs.edu.br/index.php/revista_informatica_aplicada/article/view/748/611>. Citado na página 43.

JUNIOR, N. I. d. F.; FREITAS, N. M. C. D. Objetos de aprendizagem para o ensino da história: uma busca na web. **Revista Latino-Americana de História**, v. 2, 2013. Disponível em: <http://projeto.unisinos.br/rla/index.php/rla/article/viewFile/232/185>. Citado na página 42.

KUNZE, J.; BAKER, T. The Dublin Core Metadata Element Set. 2007. Disponível em: <http://tools.ietf.org/html/rfc5013.html>. Citado na página 34.

LOM WG12. **IEEE Standard for Learning Object Metadata**. 2009. Disponível em: <http://standards.ieee.org/findstds/standard/1484.12.1-2002.html>. Citado na página 27.

LONGMIRE, W. A primer on learning objects. **Learning Circuits**, 2000. Disponível em: <http://vcampus.uom.ac.mu/orizons/html/Res270704/LOR-RLO/Longmire-RLO-primer.doc>. Citado 2 vezes nas páginas 32 e 33.

LTSC. **IEEE Learning Technology Standards Committee (LTSC)**. 2001. Disponível em: <http://web.archive.org/web/20010415201839/http://ltsc.ieee.org/wg12/>. Citado na página 34.

____. **Draft standard for learning object metadata**. 2002. 1–44 p. Disponível em: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Draft+Standard+for+Learning+Object+Metadata\#0http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Draft+standard+for+learning+object+metadata\#0>. Citado 2 vezes nas páginas 31 e 34.

MAIA, L.; PACHECO, A. A simulator supporting lectures on operating systems. In: **33rd Annual Frontiers in Education, 2003. FIE 2003.** IEEE, 2003. v. 2, p. F2C_13–F2C_17. ISBN 0-7803-7961-6. ISSN 0190-5848. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1264701>. Citado 4 vezes nas páginas 79, 80, 81 e 82.

MAIA, L. P. **SOsim: Simulador para o Ensino de Sistemas operacionais**. 2001. Disponível em: <http://www.ctesop.com.br/andreia/SistemasdeInforma??o/SistemasOperacionais/SOSIM/sosim_tese.pdf>. Citado na página 25.

MENDES, J. L. D.; COUTINHO, L. M. N.; MARTINS, C. A. P. S. Web memory hierarchy learning and research environment. In: **Proceedings of the 2006 Workshop on Computer Architecture Education: Held in Conjunction with the 33rd International Symposium on Computer Architecture**. New York, NY, USA: ACM, 2006. (WCAE '06). Disponível em: <http://doi.acm.org/10.1145/1275620.1275629>. Citado na página 79.

METADADOS. **O que são metadados?** 2014. Disponível em: <http://www.metadados.pt/index.php/oquesaometadados>. Citado na página 33.

MEYER, B. **Object-oriented Software Construction**. [S.l.]: Prentice Hall PTR, 1997. ISBN 9780136291558. Citado na página 85.

MORAES, M. P. de; SOUZA, P. S. L. . de; BRUSCHI, S. M. . **Usando Arquivos de Rastro no Projeto Amnesia**. 2011. 548 p. Citado 2 vezes nas páginas 13 e 76.

MORAIS, R. X. T. de. SOFTWARE EDUCACIONAL: A IMPORTÂNCIA DE SUA AVALIAÇÃO E DO SEU USO NAS SALAS DE AULA. **flf.edu.br**, p. 1–52, 2003. Disponível em: <http://www.flf.edu.br/revista-flf/monografias-computacao/monografia-rommel-xenofonte.pdf>. Citado na página 39.

MOREIRA, N. Interactive Manipulation of Regular Objects with FAdo . p. 335–339, 2005. Citado na página 43.

NASH, S. Learning objects, learning object repositories, and learning theory: Preliminary best practices for online courses. **Interdisciplinary Journal of Knowledge and Learning Objects**, v. 1, 2005. Disponível em: <http://www.editlib.org/p/44877/>. Citado na página 39.

NESBIT, J.; BELFER, K.; LEACOCK, T. Learning Object Review Instrument (LORI) User Manual. p. 1–12, 2002. Citado na página 40.

NEUMANN, J. V. First draft of a report on the EDVAC. **IEEE Annals of the History of Computing**, v. 15, n. 4, p. 27–75, 1945. ISSN 1058-6180. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=238389>. Citado na página 46.

NEVEN, F.; DUVAL, E. Reusable Learning Objects: A Survey of LOM-based Repositories. In: **Proceedings of the Tenth ACM International Conference on Multimedia**. New York, NY, USA: ACM, 2002. (MULTIMEDIA '02), p. 291–294. ISBN 1-58113-620-X. Disponível em: <http://doi.acm.org/10.1145/641007.641067>. Citado na página 38.

OLIVEIRA, B. H.; SANTOS, J. H.; SOUZA, P. S. L. de; BRUSCHI, S. M.; SOUZA, S. R. S. D. Amnésia : Um Simulador de Hierarquia de Memória. **Workshop sobre Educação em Arquitetura de Computadores**, p. 13–16, 2008. Disponível em: <http://www.ppgee.pucminas.br/weac/2008/PDF/WEAC-2008-Artigo-03.pdf>. Citado 2 vezes nas páginas 27 e 73.

PATTERSON, D. A.; HENNESSY, J. L. **Computer Organization and Design, Enhanced: The Hardware/Software Interface**. Elsevier Science, 2014. ISBN 9780128012857. Disponível em: <https://books.google.com.br/books?id=G7IMAwAAQBAJ>. Citado 13 vezes nas páginas 13, 26, 47, 48, 49, 54, 55, 56, 64, 66, 69, 70 e 74.

PEREIRA, L. F. D.; LAPOLLI, F.; SAMPAIO, F. F.; MOTTA, C. L. R.; OLIVEIRA, C. E. T. Ateliê de Objetos de Aprendizagem: Uma Abordagem para o Ensino de Computação em Cursos Técnicos. **Revista Brasileira de Informática na Educação**, v. 18, 2010. Disponível em: <http://ceie-sbc.tempsite.ws/pub/index.php/rbie/article/view/1225>. Citado na página 43.

ROCHA, A. A. R.; CAMPOS, G. H. B. D. G. AVALIAÇÃO DA QUALIDADE DE SOFTWARE EDUCACIONAL. **Aberto, ano**, n. 1987, 1993. Disponível em: <http://www.emaberto.inep.gov.br/index.php/emaberto/article/viewFile/845/757http://emaberto.inep.gov.br/index.php/emaberto/article/view/845/757>. Citado 2 vezes nas páginas 39 e 85.

ROCHA, P.; FERREIRA, B.; MONTEIRO, D.; NUNES, D. d. S. C.; GóES, H. C. d. N. Ensino e aprendizagem de programação: análise da aplicação de proposta metodológica baseada no sistema personalizado de ensino. **RENOTE**, 2010. Disponível em: <http://www.seer.ufrgs.br/renote/article/view/18061/106>. Citado na página 25.

RUSTICI, M. **SCORM 1.2 Visão geral para desenvolvedores**. 2009. Disponível em: <http://scorm.com/pt/scorm-explicou/t?cnicos-scorm/scorm-12-overview-for-developers/>. Citado na página 37.

SANTOS, A. I. dos. **Recursos Educacionais Abertos no Brasil: o estado da arte, desafios e perspectivas para o desenvolvimento e inovação**. [s.n.], 2013. 88 p. ISBN 978-85-60062-64-5. Disponível em: <http://unesdoc.unesco.org/images/0022/002279/227970por.pdf>. Citado na página 42.

SANTOS, M.; AMARAL, L. AVALIAÇÃO DE OBJETOS VIRTUAIS DE APRENDIZAGEM NO ENSINO DE MATEMÁTICA. **Revista de Ensino de Ciências e Matemática**, p. 83–93, 2012. Disponível em: <http://revistapos.cruzeirodosul.edu.br/index.php/rencima/article/view/109>. Citado na página 40.

SOSTERIC, M.; HESEMEIER, S. When is a Learning Object not an Object: A first step towards a theory of learning objects. **International Review of Research in Open and Distance Learning**, v. 3, n. 2, 2002. Disponível em: <http://www.irrodl.org/index.php/irrodl/article/viewArticle/106>. Citado na página 31.

STALLINGS, W. **Computer Organization and Architecture: Designing for Performance**. [S.l.]: Pearson Prentice Hall, 2006. ISBN 9780131856448. Citado 13 vezes nas páginas 13, 45, 46, 48, 49, 50, 51, 52, 54, 55, 60, 65 e 70.

STRADER, R.; THILLE, C. **The open learning initiative: Enacting instruction online**. [s.n.], 2012. 201–213 p. ISBN 978-1-933046-00-6. Disponível em: <http://www.educause.edu/Resources/GameChangersEducationandInform/Chapter15TheOpenLearningInitia/249985>. Citado na página 42.

TANENBAUM, A. S. **Organização estruturada de computadores**. [S.l.]: Pearson Prentice Hall, 2007. ISBN 9788576050674. Citado na página 51.

TANENBAUM, A. S.; BOS, H. **Modern Operating Systems**. Pearson Education, 2014. ISBN 9780133592511. Disponível em: <https://books.google.com.br/books?id=wa1GAwAAQBAJ>. Citado 10 vezes nas páginas 13, 26, 48, 56, 59, 61, 63, 64, 69 e 87.

TAROUCO, L. M. R.; FABRE, M.-C. J. M.; TAMUSIUNAS, F. R. Reusabilidade de objetos educacionais. **RENOTE**, p. 1–11, 2003. Disponível em: <http://www.seer.ufrgs.br/renote/article/view/13628/0>. Citado na página 31.

TIOSSO, F. **Utilização de objetos de aprendizagem para melhoria da qualidade do ensino de hierarquia de memória**. 102 p. Tese (Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional) — Universidade de São Paulo (USP), 2015. Citado 3 vezes nas páginas 89, 118 e 120.

TIOSSO, F.; BRUSCHI, S. M.; SOUZA, P. S. L. de; BARBOSA, E. F. Amnesia : um Objeto de Aprendizagem para o Ensino de Hierarquia de Memória. **XXV Simpósio Brasileiro de Informática na Educação**, p. 80–89, 2014. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/2932/2662>. Citado 2 vezes nas páginas 57 e 73.

UNESCO; Commonwealth of Learning. **Guidelines for Open Educational Resources (OER) in Higher Education**. [S.l.]: http://www.col.org/PublicationDocuments/Guidelines_OER_-HE.pdf, 2011. Citado 2 vezes nas páginas 27 e 42.

WEIK, M. H. **The ENIAC Story**. 1961. Citado na página 45.

WILEY, D. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. **Vasa**, v. 2830, n. 435, p. 1–35, 2003. Disponível em: <http://medcontent.metapress.com/index/A65RM03P4874243N.pdfhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Connecting+learning+objects+to+instructional+design+theory:+A+definition,+a+metaphor,+and+a+taxonomy\#0>. Citado na página 27.

____. **Openess, localization, and the future of learning objects**. 2007. Disponível em: <http://opencontent.org/presentations/bcnet07/>. Citado na página 40.

____. The Open Future. Openness as Catalyst for an Educational Reformation. **Educause review**, July/Augus, p. 15–20, 2010. ISSN ISSN-1527-6619. Citado na página 41.

WILEY, D. A. Learning object design and sequencing theory. n. June, 2000. Disponível em: <http://www.citeulike.org/group/2668/article/675045>. Citado na página 31.

WILLIAMS, D. Evaluation of learning objects and instruction using learning objects. 2000. Disponível em: <http://penta3.ufrgs.br/objetosaprendizagem/32williams.doc>. Citado na página 39.

WU, B.; QIAN, K.; BHATTACHARYA, P.; GUO, M.; HU, W. Live Programming Learning Objects on Cloud. **2011 IEEE 11th International Conference on Advanced Learning Technologies**, Ieee, p. 362–363, jul. 2011. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5992345>. Citado na página 43.

APÊNDICE

A

ARQUIVOS DE ENTRADA E SAÍDA DO REA AMNESIA

Código-fonte 2: Exemplo de arquivo de arquitetura para o REA Amnesia

```
1  <!-- Inicio de Cabeçalho do arquivo de arquitetura não modificar -->
2  <?xml version="1.0" encoding="ISO-8859-1"?>
3  <!DOCTYPE AmnesiaConfiguration SYSTEM "Configuration/amnesia.dtd">
4  <?xml-stylesheet type="text/css" href="teste.css"?>
5  <!-- Fim do Cabeçalho -->
6
7  <!-- Inicio da configuracao-->
8  <AmnesiaConfiguration>
9      <Processor>
10         <processorContains>0</processorContains>
11         <createTraceFile>0</createTraceFile>
12     </Processor>
13
14     <Trace>
15     <!-- Tamanho da palavra 4 Bytes, modificações substituir os dois
       valores-->
16         <wordSize>4</wordSize>
17     </Trace>
18     <CPU>
19     <!-- Tamanho da palavra 4 Bytes, modificações substituir os dois
       valores-->
20         <wordSize>4</wordSize>
21     </CPU>
22
23     <!-- Memória principal-->
24     <MainMemory>
```

```
25          <!-- valores em Bytes manter valor de blockSize igual o valor
        lineSize -->
26          <blockSize>1</blockSize>
27          <memorySize>32</memorySize>
28          <ciclesPerAccessRead>1</ciclesPerAccessRead>
29          <ciclesPerAccessWrite>2</ciclesPerAccessWrite>
30          <timeCicle>10</timeCicle>
31      </MainMemory>
32
33      <!-- Memória virtual-->
34    <VirtualMemory>
35        <!-- o tamanho de páginas deve ser multiplo de 2 -->
36        <!-- e menor que tamanho da memoria pricipal -->
37          <pageSize>4</pageSize>
38          <!-- o tamanho do disco deve ser multiplo do tamanho de uma p
       ágina -->
39          <diskMemorySize>16</diskMemorySize>
40          <diskCiclesPerAccessRead>1</diskCiclesPerAccessRead>
41          <diskCiclesPerAccessWrite>2</diskCiclesPerAccessWrite>
42          <timeCicle>100</timeCicle>
43          <!-- Algoritmos de substituicao possiveis NRU, LRU e FIFO -->
44          <pageTableReplacementAlgorithm>FIFO</
       pageTableReplacementAlgorithm>
45
46      <TLBType>unified</TLBType>
47      <unifiedTLB>
48          <!-- o tamanho da TLB não necessita ser multiplo de 2 -->
49        <memorySize>4</memorySize>
50        <ciclesPerAccessRead>1</ciclesPerAccessRead>
51        <ciclesPerAccessWrite>2</ciclesPerAccessWrite>
52        <timeCicle>1</timeCicle>
53        <!-- Algoritmos de substituicao possiveis LRU e FIFO -->
54        <replacementAlgorithm>LRU</replacementAlgorithm>
55      </unifiedTLB>
56    </VirtualMemory>
57 </AmnesiaConfiguration>
58 <!-- Fim da configuracao -->
```

Código-fonte 3: Exemplo de arquivo trace para o REA Amnesia

```
1 1 0
2 1 4
3 1 8
4 0 c   // inicio do loop
5 0 d
6 2 10
7 2 11  // Em todos os arquivos traces
8 2 14  // cada linha é composta de 2 números;
9 2 15  // o primeiro número diz o tipo de acesso
10 0 c  // o segundo número o endereço a ser acessado
11 0 d  // Tudo o que vier depois desse dois números
12 2 10 // é visto como um comentário
13 2 11
14 2 14
15 2 15 // Os tipos de acessos possíveis são:
16 0 c  // ''0'': leitura de dados;
17 0 d  // ''1'': gravação de dados;
18 2 10 // ''2'': busca de instrução;
19 2 11
20 2 14 // Os endereços são números em hexadecimal
21 2 15
22 0 c
23 0 d
24 2 10
25 2 11
26 2 14
27 2 15   // fim loop
28 0 16
29 1 16
30 1 17   // fim programa
```

Código-fonte 4: Exemplo de arquivo log do REA Amnesia

```
1
2 -------------------------------------
3 --- File created by Amnesia tool ---
4 -------------------------------------
5
6
7 ----------------------------------
8 --- Architecture Configuration ---
9 ----------------------------------
10
11 --- Main Memory ---
12
```

```
13 Memory size: 8 word(s)
14 Block size: 1 word(s)/block
15 Cycles per access read: 1 unity
16 Cycles per access write: 2 unity
17 Time cycle: 10 time unity
18
19 --- Virtual Memory ---
20 Page size: 4 word(s)
21 Replacement algorithm: FIFO
22
23 ---      Disk     ---
24 Memory size: 4 word(s)
25 Cycles per access read: 1 unity
26 Cycles per access write: 2 unity
27 Time cycle: 100 time unity
28
29 ---       TLB      ---
30 Memory size: 2 Frame Page(s)
31 Cycles per access read: 1 unity
32 Cycles per access write: 2 unity
33 Time cycle: 1 time unity
34 Replacement algorithm: FIFO
35
36 --------------------------
37 ---       Trace       ---
38 --------------------------
39
40 2 0
41 2 a
42 2 a
43
44 ------------------------------
45 ---      Execution Log     ---
46 ------------------------------
47
48 Location: MAINMEMORY
49 Action = Stored data in Main Memory
50 Block = 0
51
52
53 Location: MAINMEMORY
54 Action = Stored data in Main Memory
55 Block = 1
56
57
58 Location: MAINMEMORY
59 Action = Stored data in Main Memory
```

```
Block = 2


Location: MAINMEMORY
Action = Stored data in Main Memory
Block = 3


Location: MAINMEMORY
Action = Stored data in Main Memory
Block = 4


Location: MAINMEMORY
Action = Stored data in Main Memory
Block = 5


Location: MAINMEMORY
Action = Stored data in Main Memory
Block = 6


Location: DISK
Action = Storing page in Disk
Address = 0x00000000 - Page 0


------------------ Step: 001 to 003 || Trace: 2 0 ------------------

-------------------   Beginning of translation   -------------------

Location: TLB
Action = load operation - Access Type = miss
Cache Type = Accessing TLB
|-------------------------------------------------------------------|
|                              |       |   Word  |   Page  |  Byte  |
|               Tag            |  Set  |  OffSet |  OffSet | OffSet |
|-------------------------------------------------------------------|
| 00000000000000000000000000000|    ---|      ---|       00|     XX |
|-------------------------------------------------------------------|
|                      28 bits| 0 bits|   0 bits|   2 bits|  2 bits|
|-------------------------------------------------------------------|

Location: PAGETABLE
Action = loading data in the Page Table
Virtual Page = 0x00000000
```

```
Page Fault

----------------------      Beginning of paging
     ----------------------

Location: DISK
Action = Loading page from Disk and storing page in the aux vector
|--------------------|       |--------------------|
|     aux vector     |  <--  |    disk    - Page 0|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000000|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000001|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000002|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000003|
|--------------------|       |--------------------|


Location: DISK
Action = Loading page from Main Memory and storing page in Disk
|--------------------|       |--------------------|
|    disk    - Page 0|  <--  |Main Memory - Page 0|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000004|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000005|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000006|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000007|
|--------------------|       |--------------------|


Location: MAINMEMORY
Action = Store page in the Main Memory
|--------------------|       |--------------------|
|Main Memory - Page 0|  <--  |     aux vector     |
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000000|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000001|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000002|
|--------------------|       |--------------------|
|                xxxxx|       |          0x00000003|
```

```
|--------------------|       |--------------------|


Location: PAGETABLE
Action = Update the Page Table
|--------------------|       |--------------------|
|          0x00000000|  <->  |          0x00000001|
|--------------------|       |--------------------|


-------------------   End of paging   -------------------

Location: TRANSLATION
Action = Translation Address  - Using = Page Table
|-----------------------------------------------------|
|                  Virtual Address                    |
|-----------------------------------------------------|
|         Virtual Page          |PageOffSet|ByteOffSet| -->
|-------------------------------|----------|----------|
|  00000000000000000000000000000|        00|       XX |
|-----------------------------------------------------|

|-----------------------------------------------------|
|                    Real Address                     |
|-----------------------------------------------------|
|           Frame Page          |PageOffSet|ByteOffSet|
|-------------------------------|----------|----------|
|  00000000000000000000000000000|        00|       XX |
|-----------------------------------------------------|

Location: TLB
Action = Store Operation - Access Type = miss
Cache Type = TLB
|-------------------------------------------------------------------|
|                               |       |   Word  |   Page  |  Byte  |
|               Tag             |  Set  |  OffSet |  OffSet | OffSet |
|-------------------------------------------------------------------|
| 00000000000000000000000000000|    ---|      ---|       00|     XX |
|-------------------------------------------------------------------|
|                      28 bits| 0 bits|   0 bits|   2 bits|  2 bits|
|-------------------------------------------------------------------|

-------------------   End of translation   -------------------

Location: MAINMEMORY
Action = Loading data from Main Memory
Block = 0
```

```
200
201
202 ------------------ Step: 002 to 003 || Trace: 2 a ------------------
203
204 -------------------   Beginning of translation   -------------------
205
206 Location: TLB
207 Action = load operation - Access Type = miss
208 Cache Type = Accessing TLB
209 |-----------------------------------------------------------------------|
210 |                               |       |   Word  |   Page  |  Byte   |
211 |               Tag             |  Set  |  OffSet |  OffSet | OffSet  |
212 |-----------------------------------------------------------------------|
213 | 00000000000000000000000010|    ---|      ---|       10|     XX  |
214 |-----------------------------------------------------------------------|
215 |                      28 bits| 0 bits|   0 bits|   2 bits|  2 bits|
216 |-----------------------------------------------------------------------|
217
218 Location: PAGETABLE
219 Action = loading data in the Page Table
220 Virtual Page = 0x00000002
221
222 Location: TRANSLATION
223 Action = Translation Address  - Using = Page Table
224 |-----------------------------------------------------|
225 |                  Virtual Address                    |
226 |-----------------------------------------------------|
227 |         Virtual Page          |PageOffSet|ByteOffSet| -->
228 |-------------------------------|----------|----------|
229 |  00000000000000000000000010|        10|       XX |
230 |-----------------------------------------------------|
231
232 |-----------------------------------------------------|
233 |                    Real Address                     |
234 |-----------------------------------------------------|
235 |           Frame Page          |PageOffSet|ByteOffSet|
236 |---------------------------------------|----------|----------|
237 |  00000000000000000000000001|        10|       XX |
238 |-----------------------------------------------------|
239
240
241 Location: TLB
242 Action = Store Operation - Access Type = miss
243 Cache Type = TLB
244 |-----------------------------------------------------------------------|
245 |                               |       |   Word  |   Page  |  Byte   |
246 |               Tag             |  Set  |  OffSet |  OffSet | OffSet  |
```

```
|-------------------------------------------------------------------|
| 00000000000000000000000010|    ---|      ---|       10|     XX |
|-------------------------------------------------------------------|
|                       28 bits| 0 bits|   0 bits|   2 bits|  2 bits|
|-------------------------------------------------------------------|

-------------------   End of translation   -------------------

Location: MAINMEMORY
Action = Loading data from Main Memory
Block = 6


------------------ Step: 003 to 003 || Trace: 2 a ------------------

-------------------   Beginning of translation   -------------------

Location: TLB
Action = load operation - Access Type = hit
Cache Type = Accessing TLB
|-------------------------------------------------------------------|
|                             |       |   Word  |   Page  |  Byte  |
|               Tag           |  Set  |  OffSet |  OffSet | OffSet |
|-------------------------------------------------------------------|
| 00000000000000000000000010|    ---|      ---|       10|     XX |
|-------------------------------------------------------------------|
|                       28 bits| 0 bits|   0 bits|   2 bits|  2 bits|
|-------------------------------------------------------------------|

Location: TRANSLATION
Action = Translation Address  - Using = TLB
|-------------------------------------------------|
|                  Virtual Address                |
|-------------------------------------------------|
|        Virtual Page          |PageOffSet|ByteOffSet| -->
|------------------------------|----------|----------|
|  00000000000000000000000010|        10|       XX |
|-------------------------------------------------|

|-------------------------------------------------|
|                    Real Address                 |
|-------------------------------------------------|
|           Frame Page         |PageOffSet|ByteOffSet|
|------------------------------------------|----------|
|  00000000000000000000000001|        10|       XX |
|-------------------------------------------------|

```

```
294 --------------------   End of translation   --------------------
295
296 Location: MAINMEMORY
297 Action = Loading data from Main Memory
298 Block = 6
299
300
301 ------------------------------
302 ---        Statistics        ---
303 ------------------------------
304
305 Main Memory
306   Read Access: 13
307   Read Cycles: 13
308   Read Time: 130
309   Write Access: 10
310   Write Cycles: 20
311   Write Time: 200
312   Total Access: 23
313   Total Time: 330
314
315 Disk
316   Read Access: 1
317   Read Cycles: 1
318   Read Time: 100
319   Write Access: 1
320   Write Cycles: 2
321   Write Time: 200
322   Total Time: 300
323
324 Page Table
325   Read Access: 2
326   Read Cycles: 2
327   Read Time: 20
328   Write Access: 2
329   Write Cycles: 4
330   Write Time: 40
331   Total Access: 2
332   Page Fault: 1
333   Total Time: 60
334
335 TLB
336   Read Access: 3
337   Read Cycles: 3
338   Read Time: 3
339   Write Access: 2
340   Write Cycles: 4
```

```
341   Write Time: 4
342   Total Access: 3
343   Total Hit: 1
344   Total Miss: 2
345   Hit rate: 0.33333334
346   Total Time: 7
347
348  Total Time: 637
```

APÊNDICE

B

MELHORIA DE LEGIBILIDADE NO MÉTODO TRANSLATION

Código-fonte 5: Metódo *translation* antes das alterações de legibilidade

```
1 public Word translation(Word virtualPageNumber) {
2     Block physicalPageBlock = new Block(getBlockSize(), super.
    getWordSize());
3
4     // 1 - verify if this is a valid page number.
5     if (virtualPageNumber.getLongValue() > numberOfPages) {
6         System.out.println("ERROR: Invalid page number " +
    virtualPageNumber.getLongValue() + ".");
7
8         return null;
9     }
10
11    // 2 - retrieve the associated entry from the page table or any
    of the TLB.
12    if (dataTLB != null) {
13        // verify if we are in an instruction cicle and if we have an
     instruction TLB.
14
15        if ((!dataCycle) && instructionTLB != null) {
16            physicalPageBlock.setBlock(instructionTLB.start(
    virtualPageNumber));
17
18        } else {
19            physicalPageBlock.setBlock(dataTLB.start(
    virtualPageNumber));
20
21        }
```

```
22      } else {
23          physicalPageBlock.setBlock(pageTable.load(virtualPageNumber,
    1));
24
25      }
26      // stores the position of the page entry inside the block.
27
28      int position = (int) ((virtualPageNumber.getLongValue() +
    pageTable.getRegisterOffset()) % getBlockSize());
29      // stores the page entry to be manipulated.
30      Word pageEntry = physicalPageBlock.getWord(position);
31      // the physical page number.
32      Word physicalPageNumber = new Word(super.getWordSize());
33      // mask to separate flags from physical frame.
34      long mask = ((1 << physicalFrameAddressSize) - 1);
35
36      // 3 - the page is in main memory?
37
38      if (pageInMainMemory(pageEntry)) {
39          // 3.1.1 - remove flags.
40          physicalPageNumber.setWord(pageEntry.getLongValue() & mask);
41
42      } else {  //troca de páginas
43
44          // 3.2.1 - load the page from the disk.
45          pageIn = disk.load(pageEntry.getLongValue());
46
47          // 3.2.2 - query for address.
48
49          // stores the address where the page will be placed.
50          Word pageNewAddress = new Word(super.getWordSize());
51
52          // stores the page entry of the page to be paged out.
53          Word pageOutEntry;
54
55          // page out position.
56          int pageOutPosition;
57
58          // page out block
59          Block pageOutBlock = new Block(getBlockSize(), getWordSize())
    ;
60
61          // stores the virtual address of the selected page.
62          Word pageOutAddress = new Word(super.getWordSize());
63
64          // 3.2.3 - query the page to be paged out.
65          pageOutAddress.setWord(pageTable.getReplacementPosition());
```

```
66
67          // 3.2.4 - query the page entry.
68          if (dataTLB != null) {
69              // verify if we are in an instruction cycle and if we
    have an instruction TLB.
70              if ((!dataCycle) && instructionTLB != null) {
71
72                  // if the data is in dataTLB and not in
    instructionTLB
73                  if (!(instructionTLB.searchData(pageOutAddress)) &&
                      (dataTLB.searchData(pageOutAddress)))
74                  {
75
76                      instructionTLB.start(pageOutAddress, dataTLB.
    start(pageOutAddress));
77                      pageOutBlock.setBlock(instructionTLB.start(
    pageOutAddress));
78
79                  } else { // if data isn't in dataTLB
80
81                      pageOutBlock.setBlock(instructionTLB.start(
    pageOutAddress));
82
83                  }
84
85              } else {
86                  if (instructionTLB != null) {
87                      // if the data is in instructionTLB and not in
    dataTLB
88                      if (!(dataTLB.searchData(pageOutAddress)) &&
                                (instructionTLB.searchData(
    pageOutAddress)))
89                      {
90                          dataTLB.start(pageOutAddress, instructionTLB.
    start(pageOutAddress));
91                          pageOutBlock.setBlock(dataTLB.start(
    pageOutAddress));
92
93                      } else {// if data isn't in instructionTLB
94                          pageOutBlock.setBlock(dataTLB.start(
    pageOutAddress));
95                      }
96
97                  } else {
98                      pageOutBlock.setBlock(dataTLB.start(
    pageOutAddress));
99                  }
```

```
100             }
101
102         } else {
103             pageOutBlock.setBlock(pageTable.load(pageOutAddress, 1));
104         }
105
106         pageOutPosition = (int) pageOutAddress.getLongValue() %
    getBlockSize();
107         pageOutEntry = pageOutBlock.getWord(pageOutPosition);
108
109         // 3.2.5 - get the physical page address.
110         pageNewAddress.setWord((pageOutEntry.getLongValue() & mask) *
     getPageSize());
111
112
113         // 3.2.6 - invalidate caches
114         control.invalidateCaches(pageNewAddress, pageSize);
115
116         // 3.2.7 - prepare the page to be paged out.
117         pageOut = this.loadPageFromMainMemory(pageNewAddress);
118
119
120         // 3.2.2.6 - page out.
121         Word diskAddress = new Word(super.getWordSize());
122         diskAddress.setWord(disk.store(pageOut));
123
124         //making pageTable entry to the pageOut
125         dateTable(pageOutAddress, diskAddress, pageOutBlock,
    pageOutPosition, false, false, false,true,false);
126         // }
127
128         if (controller != null) {
129
130             String location = String.format("Storing page into
    PageTable");
131             String addressString = String.format("\n %s", printer.
    formatHexString(Long.toHexString(pageNewAddress.getLongValue())));
132             //controller.pause("virtualmemory|"+location+"|address",
    addressString );
133
134         }
135
136         // 3.2.8 - store the page into main memory.
137         storePageInMainMemory(pageNewAddress, pageIn);
138
139         // 3.2.9 - retrieve the frame.
```

```
140         physicalPageNumber.setWord(getPageNumber(pageNewAddress.
      getLongValue()) & mask);
141      }
142
143      // 4 - updating page table block
144
145      /* if( dataTLB!=null ) {
146          // verify if we are in an instruction cycle and if we have an
       instruction TLB.
147          if ( (!dataCycle) && instructionTLB!=null ){
148              physicalPageBlock.setBlock(instructionTLB.start(
      virtualPageNumber));
149          }
150          else{
151              physicalPageBlock.setBlock(dataTLB.start(
      virtualPageNumber));
152          }
153      }else{
154          physicalPageBlock.setBlock(pageTable.load(virtualPageNumber
      ,1));
155      }*/
156
157       // System.out.println("Physical Page Block 2");
158       // physicalPageBlock.getBlock().printBlock();
159           // 5 - update table.
160      updateTable(virtualPageNumber, physicalPageNumber,
      physicalPageBlock, position, true, writeCycle, true, true,false);
161              // pageTable.print();
162              // System.out.println("instructionTLB ");
163              // instructionTLB.printContents();
164              // System.out.println("dataTLB ");
165              // dataTLB.printContents();
166
167          return physicalPageNumber;
168
169 }// translation.
```

Código-fonte 6: Metódo *translation*, após as alterações para melhoria da legibilidade

```
1 private Word translationNewVersion(Word virtualPageNumber, long
      pageOffSet){
2      Block physicalPageBlock = new Block(super.getBlockSize(), super.
      getWordSize());
3      boolean needUpdateTLB = false;
4      boolean needUpdatePAgeTable = false;
5      boolean foundInTLB = false;
6      // the physical page number.
```

```
7     Word realPageNumber = new Word(super.getWordSize());
8     long mask = ((1 << physicalFrameAddressSize) - 1);
9
10    if(!verifyValidateVirtualPage(virtualPageNumber))
11        return null;
12
13    //try finding real address in TLB
14    physicalPageBlock.setBlock(SearchTLB(virtualPageNumber,
      pageOffSet));
15
16    // if not find real address in TLB,
17    // going to finding in PageTable
18    if(physicalPageBlock.checkEmpty()) {
19        physicalPageBlock.setBlock(SearchPageTable(virtualPageNumber)
      );
20
21        if(dataTLB != null) {
22
23            needUpdateTLB = true;
24        }
25
26    // found in TLB
27    } else if(dataTLB != null){
28
29        foundInTLB = true;
30        needUpdatePAgeTable = false;
31        if(!verifyPageInMainMemory(virtualPageNumber,
      physicalPageBlock)) {
32
33            // not found in Main Memory, but address stay in TLB
34            System.out.println("error: not found in Main Memory, but
      address stay in  TLB\n" + "this is not possible , because the TLB
      should only contain addresses of data in the main " + "memory ");
35        }
36    }
37
38    int position = (int) ((virtualPageNumber.getLongValue() +
      pageTable.getRegisterOffset()) \% getBlockSize());
39    pageEntry = physicalPageBlock.getWord(position);
40
41    // verify in main memory case not found in TLB, but found in Page
       Table
42    // if necessary swap pages
43    if(!foundInTLB && !verifyPageInMainMemory(virtualPageNumber,
      physicalPageBlock)){
44
45        // active update in PageTable
```

```
46          needUpdatePAgeTable = true;
47
48          // modify page entry, realize swap pages
49          realPageNumber.setWord(swapPage(pageOffSet));
50
51          // print in log translation from page table
52          control.addTranslationLog("Translation Address", "Page Table"
    , virtualPageNumber.getLongValue(), realPageNumber.getLongValue(),
     pageOffSet, pageOffSetSize);
53      }
54
55      else if (!foundInTLB){
56
57          realPageNumber.setWord(pageEntry.getLongValue() & mask);
58          // print in log translation from page table
59          control.addTranslationLog( "Translation Address", "Page Table
    ",                      virtualPageNumber.getLongValue(),
    realPageNumber.getLongValue(), pageOffSet, pageOffSetSize);
60
61      }else{
62
63          realPageNumber.setWord( pageEntry.getLongValue() & mask );
64          // print in log translation from TLB
65          control.addTranslationLog( "Translation Address", "TLB",
    virtualPageNumber.getLongValue(), realPageNumber.getLongValue() ,
    pageOffSet, pageOffSetSize);
66      }
67
68      if(needUpdatePAgeTable) {
69
70          UpdatePageTable(virtualPageNumber, realPageNumber,
    physicalPageBlock, pageOffSet);
71      }
72      else{
73          int blockPosition = (int) ((virtualPageNumber.getLongValue()
    + pageTable.getRegisterOffset()) % mainMemory.getBlockSize());
74          updateTable(virtualPageNumber, realPageNumber,
    physicalPageBlock, position,blockPosition, true, writeCycle, true,
     false, false, pageOffSet);
75      }
76
77      if(needUpdateTLB) {
78          UpdateTLB(virtualPageNumber, realPageNumber,
    physicalPageBlock, pageOffSet);
79          needUpdateTLB = false;
80      }
81
```

```
82      return realPageNumber;
83  }// translation.
```

APÊNDICE

C

TESTES

C.1 Pré-teste

Nome: _____

Número USP (Opcional): _____

1. Qual a política de substituição de páginas que remove a página que está há mais tempo sem ser referenciada?

   a) LFU (Least Frequently used).

   b) FIFO (First In First Out).

   c) LRU (Least Recently Used).

   d) FCFS (First Come First Served).

   e) NRU (Not Recently Used).

2. Qual a política de substituição de páginas que remove a primeira página virtual que foi carregada na memória principal?

   a) FCFS (First Come First Served).

   b) NRU (Not Recently Used).

   c) FIFO (First In First Out).

   d) WsClock (Working Set Clock).

   e) LRU (Least Recently Used).

3. Qual das políticas de substituição é mais custosa de ser implementada em hardware e por quê?

a) NRU – pois todas as páginas modificadas devem ser escritas imediatamente no disco.

b) FIFO – pois deve ser implementada uma lista ligada em hardware.

c) NRU – pois os bits de controle podem gastar um espaço exagerado.

d) LRU – implementada com um contador em hardware e o valor desse contador deve ser armazenado em cada página presente na memória principal.

e) N.D.A.

4. Suponha uma memória principal com 4 molduras de páginas (page frames). Inicialmente, as páginas virtuais 1, 2, 3 e 5 são carregadas, respectivamente nas molduras de página 0, 1, 2 e 3. A moldura 0 foi a primeira a ser carregada, a moldura 2 a segunda, a moldura 3 a terceira e a moldura 5 a última. A política de substituição de páginas é a LRU. Considere que um processo realiza a seguinte sequência de acessos 1, 1, 2, 3, 4, 5, 6 e 4. Quais serão as 4 páginas que estarão, respectivamente, nas molduras de página 0, 1, 2 e 3, ao final da execução?

a) 5, 6, 3 e 4.

b) 4, 6, 3 e 5.

c) 1, 2, 3 e 4.

d) 5, 2, 6 e 4.

e) 4, 6, 5 e 3.

5. Considerando o conceito de políticas de substituição, é incorreto afirmar que:

a) No momento de uma substituição, a política FIFO não leva em consideração se a página foi acessa e/ou modificada.

b) A política NRU separa as páginas em classes de acordo com os bits de Referência (R) e Modificado (M), considerando somente as páginas que estão com o bit de Validade (V) em 1.

c) A política NRU remove a página que pertencente à classe mais alta.

d) A política LRU remove a página que há mais tempo está sem ser referenciada.

e) A política de substituição LRU tem um custo alto para ser implementada em hardware.

6. Qual técnica é utilizada para diminuir os acessos à memória principal quando é necessário fazer uma tradução de um endereço?

a) Duplicar a Tabela de Páginas.

b) Tabela de Páginas duplamente encadeadas.

c) Aumentar o tamanho das páginas para o limite da cache.

d) Utilizar uma TLB.

e) Diminuir o tamanho das páginas.

7. Com um acerto (hit) na TLB, a moldura de página (Frame Page) é concatenada com:

a) Word Offset.

b) Page Offset.

c) Endereço Real.

d) Virtual Page.

e) N.D.A.

8. Com a falha (miss) na TLB é obtido o:

a) Endereço Virtual.

b) Endereço Real.

c) Page Offset.

d) Virtual Page.

e) N.D.A.

9. Tendo uma TLB totalmente associativa com 4 blocos e 2 palavras por blocos, quantos bits são necessários para representar a TAG em uma arquitetura 32 bits? Considere que o tamanho da página seja 4 palavras.

a) 29

b) 26

c) 24

d) 30

e) 31

10. Quando uma página presente na memória principal deve ser substituída e essa página está presente em um bloco da TLB, é correto afirmar que:

a) O bloco da TLB deve ser alterado para conter o endereço do disco.

b) Nada, pois quando for acessado esse bloco da TLB a inconsistência de informação será detectada e consertada.

c) O bloco só é sinalizado como inválido se não existir outros blocos livres na TLB.

d) O bloco da TLB só é invalidado caso um bloco na cache seja invalidado também.

e) O bloco deve ser sinalizado com bit de inválido

C.2 Pós-teste

1. Uma política de substituição deve ser utilizada quando:

    a) Uma página que está presente na memória secundária deve ser trazida para a memória principal, mas não há mais nenhuma moldura de página livre.

    b) Uma página que está presente na memória principal deve ser levada para a memória secundária, mas não há mais espaço na memória secundária.

    c) Uma página é removida da memória principal e deve haver uma substituição das informações contidas na TLB.

    d) Uma página é modificada e deve ser substituída imediatamente.

    e) N.D.A.

2. Qual é a política de substituição que utiliza dois bits para separar as páginas em 4 classes distintas?

    a) NRU (Not recently used).

    b) WsClock (Working Set Clock).

    c) FCFS (First Come First Served).

    d) LRU (Least recently used).

    e) N.D.A.

3. Considerando o conceito de políticas de substituição, é incorreto afirmar:

    a) A política LRU remove a página que tem a menor quantidade de referências.

    b) No momento de uma substituição, a política FIFO não leva em consideração se a página foi acessa e/ou modificada.

    c) A política NRU separa as páginas em classes de acordo com os bits de Referência (R) e Modificado (M), considerando somente as páginas que estão com o bit de Validade (V) em 1.

    d) A política de substituição LRU tem um custo alto para ser implementada em hardware.

    e) N.D.A.

4. Qual alternativa não é uma forma de implementação da política de substituição LRU?

    a) Segunda chance.

    b) Aging

    c) Com um contador.

d) Lista encadeada.

e) N.D.A.

5. Suponha uma memória principal com 4 molduras de páginas (page frame). Inicialmente as páginas virtuais 1, 2, 3 e 5 são carregadas, respectivamente nas molduras de página 0, 1, 2 e 3. A política de substituição de páginas é a FIFO. Considere que um processo realiza a seguinte sequência de acessos 1, 1, 3, 2, 4, 5, 6 e 4. Quais serão as 4 páginas que estarão, respectivamente, nas molduras de página 0, 1, 2 e 3, ao final da execução?

   a) 4, 6, 3 e 5.

   b) 1, 2, 3 e 4.

   c) 3, 5, 4 e 6.

   d) 5, 6, 3 e 4.

   e) N.D.A.

6. Com a utilização de memória virtual, qual das alternativas é uma motivação para a utilização da TLB?

   a) Com a memória virtual são necessários 2 acessos à memória principal, um para obtenção do endereço real e outro para leitura ou escrita do dado ou instrução.

   b) O tempo gasto para realizar 1 acesso à memória principal aumentou.

   c) São realizados mais 2 acessos de escrita na memória principal, além do acesso para obtenção do dado e/ou instrução.

   d) Após a utilização da memória virtual não houve queda de desempenho, mas a TLB é utilizada para melhorar ainda mais o desempenho.

   e) N.D.A.

7. A tradução de endereço virtual para o endereço real é feita pela MMU (Memory Management Unit). Essa tradução é realizada acessando:

   a) A TLB ou a Tabela de Páginas.

   b) Somente a TLB.

   c) Somente o Disco.

   d) Somente a Tabela de Páginas.

   e) N.D.A.

8. Com um acerto (hit) na TLB é obtido o:

   a) Endereço Real.

   b) Endereço Virtual.

c) Page Offset.

d) Virtual Page.

e) N.D.A.

9. Uma falha (miss) na TLB:

a) Determina que não foi possível realizar a tradução endereço a partir da TLB.

b) Determina que a página virtual está no disco.

c) Determina que não é necessário acessar a Tabela de Páginas para realizar a tradução daquele endereço.

d) Determina que não é necessário acessar a Memória principal para obter o dado ou instrução.

e) Implica que os dados contidos na TLB estão corrompidos, devendo ser removidos da TLB.

10. Tendo uma TLB com 4 blocos e 2 palavras por blocos, quantos bits são necessários para representar o Word offset em uma arquitetura 32 bits?

a) 1

b) 2

c) 4

d) 6

e) 8

APÊNDICE

D

QUESTIONÁRIOS DE EXPECTATIVA E REAÇÃO AO USO

D.1 Expectativa de uso - Questionário utilizado nos Experimentos

1. O que você acha da ideia de ter a disposição um software ou Recursos Educacionais Abertos (REA) em qualquer lugar e hora para poder resolver algum problema ligado a disciplina?

   a) Muito Bom

   b) Bom

   c) Indiferente

   d) Ruim

   e) Péssimo

2. Quanto ao uso do recurso educacional aberto para auxiliar o conhecimento, pode ser considerado:

   a) Vai melhorar o seu conhecimento

   b) Vai saber pouca coisa a mais

   c) Vai se manter na mesma

   d) Não vai fazer diferença

3. Na sua opinião, o uso do REA Amnesia vai despertar o interesse do usuário sobre o assunto?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica ou não sabe

Justifique sua resposta: _____

_____

_____

4. Na expectativa de uso do REA Amnesia, você se sente:

a) Temeroso, pois você não se sente à vontade em lidar com REA.

b) Indiferente, para você tanto faz usar ou não um software.

c) Motivado, em poder aprender mais com o uso da tecnologia.

d) Confiante, pois você acredita que o uso vai facilitar a sua aprendizagem.

5. Na sua opinião, o REA vai permitir o desenvolvimento de um conhecimento novo?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica ou não sabe

Justifique sua resposta: _____

_____

_____

6. Na expectativa para o uso do REA Amnesia, você pretende usar em experiências futuras?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica ou não sabe

7. Gostaria de fazer algum comentário sobre a expectativa de uso do REA?

_____

_____

_____

_____

_____

D.2 Reação ao uso - Questionário utilizado nos Experimentos

1. O seu conhecimento sobre o conteúdo de memória virtual antes de utilizar o Recurso Educacional Aberto (REA) Amnesia era:

   a) Muito bom

   b) Bom

   c) Regular

   d) Ruim

   e) Péssimo

2. O REA Amnesia é compreensível e de fácil manuseio?

   a) Sim

   b) Parcialmente

   c) Não

   d) Não se aplica

3. Você considera, em relação ao grau de conhecimento adquirido com o uso do REA Amnesia, que saiu:

   a) Sabendo menos do que antes

   b) Sabendo a mesma coisa que antes

   c) Sabendo pouca coisa a mais

   d) Sabendo muito mais do que antes

4. Em relação a motivação de utilizar REA Amnesia para auxiliar o conhecimento de memória virtual, pode ser considerada que:

   a) Aumentou

   b) Permaneceu igual de quando você não conhecia o REA

   c) Diminuiu

   d) Sofreu momentos de altos e baixos.

5. Na sua opinião, o uso do REA Amnesia despertou o interesse sobre os conteúdos de memória virtual contidos nele?

   a) Sim
   b) Parcialmente
   c) Não
   d) Não se aplica

6. A aula se tornou mais interessante com o uso do REA Amnesia?

   a) Sim
   b) Parcialmente
   c) Não
   d) Não se aplica

7. A atividade é apropriada e o uso do REA Amnesia facilitou a sua compreensão sobre o conteúdo de Memória Virtual?

   a) Sim
   b) Parcialmente
   c) Não
   d) Não se aplica

   Discorra sobre sua opinião:

   _____

   _____

   _____

8. As respostas de todas as operações realizadas no REA Amnesia estavam de acordo com suas expectativas?

   a) Sim
   b) Parcialmente
   c) Não
   d) Não se aplica

   Discorra sobre sua opinião:

   _____

   _____

   _____

9. Você se sente seguro quanto aos resultados obtidos no REA Amnesia?

   a) Sim

   b) Parcialmente

   c) Não

   d) Não se aplica

   Discorra sobre sua opinião:

   _____

   _____

   _____

10. Está confiante e saberá usar o conhecimento adquirido pelo REA Amnesia em uma futura prática?

    a) Sim

    b) Parcialmente

    c) Não

    d) Não se aplica

    Discorra sobre sua opinião:

    _____

    _____

    _____

11. O uso do REA Amnesia aumentou o nível de seu conhecimento no conteúdo de memória virtual?

    a) Sim

    b) Parcialmente

    c) Não

    d) Não se aplica

    Discorra sobre sua opinião:

    _____

    _____

    _____

12. Na sua opinião, sobre o REA Amnesia

    Qual a relevância:

    ( ) Muito bom ( ) Bom ( ) Regular ( ) Ruim ( ) Péssimo

    Qual a adequação do conteúdo memória virtual:

    ( ) Muito bom ( ) Bom ( ) Regular ( ) Ruim ( ) Péssimo

    Qual a facilidade na compreensão das atividades:

    ( ) Muito bom ( ) Bom ( ) Regular ( ) Ruim ( ) Péssimo

13. Você usaria o REA Amnesia novamente?

    a) Sim
    b) Parcialmente
    c) Não
    d) Não se aplica

    Discorra sobre sua opinião:

    _____

    _____

    _____

14. Gostaria de fazer algum comentário sobre o uso do REA Amnesia?

    _____

    _____

    _____

    _____

15. Você identificou algum defeito e/ou gostaria de sugerir melhorias no REA Amnesia?

    _____

    _____

    _____

    _____

    _____

APÊNDICE

E

PLANOS DE AULA

E.1 Plano de aula 1 - Funcionamento do Recurso Educacional Aberto Amnesia

Público alvo

Alunos sem um conhecimento prévio do Recurso Educacional Aberto (REA) Amnesia. Motivação do Plano de Aula: o aluno não tem como utilizar o REA Amnesia adequadamente, sem saber como ele funciona. Esta aula é motivada pela necessidade de transmitir aos alunos as principais características do REA e suas funcionalidades.

Objetivo Geral

Ensinar aos alunos o como utilizar o REA Amnesia.

Objetivos Específicos

O aluno deverá ser capaz de:

1. Iniciar a execução do REA, identificar os seus principais componentes (outros módulos – processador/rastro, cache, memória virtual - menus, janelas para interação, arquivos de entrada/saída, modos de execução).

2. Conhecer o fluxo básico de execução, identificando, inclusive entradas inválidas para a simulação.

3. Identificar as principais informações inseridas no log de saída, contendo os acessos à memória.

4. Identificar as métricas usadas nas informações relativas ao desempenho.

5. Executar simulações simples no REA Amnesia.

6. Criar os seus próprios arquivos de arquitetura em XML e arquivos de rastro.

Duração

A aula tem uma duração de uma hora, se todas as atividades forem apresentadas. Se o professor somente apresentar a atividade de Operações Básicas do REA Amnesia, tem uma duração 5 - 10 minutos.

Desenvolvimento

A aula será desenvolvida com atividades práticas usando o REA Amnesia em um computador. Cada atividade descrita a seguir aborda um ou mais objetivos a serem atingidos.

At. 1: Operações básicas no Amnesia

Atividades-Operações Básicas

At. 2: Geração do arquivo de Arquitetura em XML

a) **Motivação**: Essa atividade apresenta quais as informações devem estar presentes no arquivo XML para cada componente, fazendo que informações corretas gerem uma arquitetura válida para ser aceita pelo REA Amnesia.

b) **Objetivo**: Fazer com que os alunos entendam como a arquitetura é descrita no arquivo XML, para que futuramente eles possam criar as suas próprias arquiteturas.

c) **Detalhamento**: Será aberto um arquivo de arquitetura (02-2) em um editor de texto, onde serão apresentadas as seguintes informações:

   i. Cabeçalho do arquivo XML, que deve ser igual em todos os arquivos XML carregados pelo REA Amnesia
   <?xml version="1.0"encoding="ISO-8859-1"?>
   <!DOCTYPE AmnesiaConfiguration SYSTEM "Configuration/amnesia.dtd»
   <?xml-stylesheet type="text/css"href="teste.css"?>

   ii. Início da descrição da arquitetura com a tag <AmnesiaConfiguration>

   iii. Apresentar as características do processador que estaria acessando a hierarquia de memória. Tais acessos podem ser feitos por um arquivo de rastro, resultante da execução em um processador. Mesmo usando tal arquivo de rastro, as características do processador precisam ser fornecidas. Tais características devem ser mantidas no arquivo XML.
   <Processor>
   <processorContains>0</processorContains>
   <createTraceFile>0</createTraceFile>
   </Processor>

iv. Informações de tamanho da palavra em BYTES. Os valores devem ser iguais nas duas tags <wordSize>.
<Trace>
<wordSize>4</wordSize>
</Trace>
<CPU>
<wordSize>4</wordSize>
</CPU>

v. Informações da memória principal: Informações de tamanho de bloco, tamanho da memória, tempo de ciclos entre outros.
<MainMemory>
<blockSize>1</blockSize>
<memorySize>16</memorySize>
<ciclesPerAccessRead>1</ciclesPerAccessRead>
<ciclesPerAccessWrite>2</ciclesPerAccessWrite>
<timeCicle>10</timeCicle>
</MainMemory>

vi. Informações da memória virtual que são: tamanho de páginas em quantidade de palavras, tamanho do disco em quantidade de palavras e tempo para acessos à leitura e escrita.
<VirtualMemory>
<pageSize>4</pageSize>
<diskMemorySize>24</diskMemorySize>
<diskCiclesPerAccessRead>1</diskCiclesPerAccessRead>
<diskCiclesPerAccessWrite>2</diskCiclesPerAccessWrite>
<timeCicle>100</timeCicle>
<pageTableReplacementAlgorithm>FIFO</pageTableReplacementAlgorithm>
<TLBType>none</TLBType>
</VirtualMemory>

vii. Finalizar a configuração com a tag: </AmnesiaConfiguration>

viii. Importante salientar que para todas as arquiteturas, as configurações dos parâmetros ciclesPerAccessRead, ciclesPerAccessWrite e timeCicle serão constantes conforme a seguir:

Configuração / Memória	ciclesPerAccessRead	ciclesPerAccessWrite	timeCicle
Cache	1	2	1
Principal	1	2	10
Disco	1	2	100
TLB	1	2	1

ix. Aplicar mudanças na arquitetura para que o aluno crie seu próprio arquivo de arquitetura. Sugestões de mudanças são:
   A. Mudar o tamanho da memória principal de 16 para 32.
   B. Mudar o tamanho da página de 4 para 8.
   C. Mudar o tamanho do disco de 24 para 16.
x. Salvar arquivo com as mudanças realizadas dar um nome para a nova arquitetura.
xi. Carregar nova arquitetura e observar mudanças na interface do REA.

d) **Arquivos**
   i. Arquitetura: Architecture-02-2-MM-16-VM(PS-4-DM-4-RA-FIFO)-TLB(none)

e) **Pontos a destacar:**
   i. É esperado que o aluno entenda como descrever um arquivo de arquitetura XML, além disso, incentivar que o aluno busque informações de como descrever as informações de outras memórias que podem ser descritas no REA Amnesia como Cache e TLB.

At. 3: Geração do arquivo de rastro

a) **Motivação**: Os arquivos de rastro contêm a relação cronológica de acessos à memória em uma execução anterior. Tais arquivos são usados para determinar a ordem os acessos devem ocorrer, simulando o uso da hierarquia de memória no Amnesia. Os alunos precisam saber como os arquivos de rastro são organizados de modo a usá-los adequadamente.

b) **Objetivo**: Fazer com que os alunos entendam como os acessos à memória são descritos no arquivo de rastro, para que futuramente eles possam criar os seus próprios arquivos de rastro.

c) **Detalhamento**: Será aberto o arquivo de rastro (6) em um editor de texto, onde serão apresentadas as seguintes informações:
   i. Cada linha do arquivo de rastro é referente a 1 acesso à memória.
   ii. Cada linha é composta por dois números (tipo de acesso e endereço a ser acessado) o que vem após isso e ignorado como comentário.
   iii. *(se não apresentado na primeira atividade)* Os tipos de acessos possíveis são:
      - "0": Leitura de dados;
      - "1": Gravação de dados;
      - "2": Busca de instrução;
   iv. Os endereços estão em hexadecimal.
   v. Verificar que o acesso ao endereço mais alto, representa o tamanho do processo.
      - Supondo que há um acesso de leitura ao endereço f (2 f).

- O processo terá 16 palavras (f em decimal é 15. O processo é de 0 até 15 dados)
- O tamanho da memória deve ser no mínimo do tamanho do processo (ex: 16 palavras).
- O tamanho total de memória é o tamanho total da RAM mais o tamanho total do disco.

d) **Arquivos**:

i. Rastro: TR_6_read_and_write_30_rand_PS_24

e) **Pontos a destacar**:

i. Espera-se que o aluno entenda como descrever um arquivo de rastro.

At. 4: Arquitetura com entradas inválidas

a) **Motivação**: verificar que há restrições em relação aos valores de tamanho de cada componente. Deve-se evitar valores que possam causar erro na construção arquitetura.

b) **Objetivo**: Com essa tarefa é possível notar que há algumas restrições de valores para descrever as memórias presentes em cada arquitetura, além de observar que algumas combinações de valores geram erros.

c) **Detalhamento**: O aluno irá carregar uma arquitetura (with-errors) com erros implantados propositalmente, para no momento que os alunos tentarem carregar a arquitetura os erros são apresentados e corrigidos. As sequências de erros são:

i. Tamanho de palavra nas tags <Trace> e <CPU> são diferentes
ii. Tamanho da memória principal não é múltipla do tamanho de páginas
iii. O Tamanho do disco não é múltiplo do Tamanho da página
iv. Política de substituição não é uma entrada válida
v. Tamanho da página não é base 2. Após essa correção indicar que há outro erro e solicitar que os alunos identifiquem esse erro. O erro em questão é que o tamanho da memória principal não será múltiplo do tamanho da página.
vi. Após essas correções a arquitetura irá carregar corretamente.
vii. *(Opcional)* Abrir novamente a arquitetura e deixar os alunos injetarem erros na arquitetura e verificar quais os resultados obtidos com o carregamento da arquitetura.

d) **Arquivos**:

i. Arquiteturas: Architecture-with-errors.

e) **Pontos a destacar**:

i. É esperado com essa atividade que os alunos entendam como construir as suas próprias arquiteturas, evitando os valores que possam causar erros na arquitetura.

ii. Não é necessário realizar uma execução na arquitetura.

At. 5: Arquivo de Rastro com entradas inválidas

a) Motivação: Verificar que há algumas restrições em relação aos valores de tamanho de cada componente. Tentar evitar valores que possam causar erro na construção da arquitetura.

b) Objetivo: Ressaltar que há restrições de valores para os tipos de acesso, além de observar os valores inválidos para o arquivo.

c) Detalhamento: O aluno irá abrir um arquivo de rastro (with-errors) com erros implantados propositalmente, para no momento que os alunos tentarem carregar o arquivo de rastro, os erros sejam apresentados e corrigidos. As sequências de erros são:

i. Um comentário em uma posição errada do arquivo de rastro
ii. Um valor de tipo de acesso inválido
iii. Endereço com valor negativo
iv. Um endereço hexadecimal inexistente (g)
v. Endereço maior que tamanho total da memória.
vi. Após essas correções o arquivo de rastro deve carregar corretamente.
vii. *(Opcional)* Abrir novamente o arquivo de rastro e deixar os alunos injetarem erros no arquivo e verificar quais os resultados obtidos com o carregamento do arquivo de rastro.
viii. Após as correções realizar uma execução direta e verificar o resultado.

d) **Arquivos:**

i. Arquiteturas: Architecture-with-errors - com as correções realizadas.
ii. Rastro: Trace_with_error

e) **Pontos a destacar**:

i. É esperado com essa atividade que os alunos entendam como construir os seus próprios arquivos de rastro, evitando os valores que podem causar erros.
ii. Realizar a execução somente para verificar que os erros foram removidos.

At. 6: ***Exercício***: Desenvolver um arquivo de arquitetura e um arquivo de rastro para realizar uma simulação no REA Amnesia, após finalizar a simulação realizar uma coleta dos tempos das memórias presentes na arquitetura.

a) O arquivo de arquitetura deve ter:

i. Tamanho de palavra 4
ii. Memória Principal de tamanho 16 palavras
iii. Tamanho de bloco 1 palavra.

iv. Memória Virtual com tamanho de página 16

v. Tamanho de disco 32 palavras.

vi. Com Política de substituição LRU.

vii. Sem TLB.

b) O arquivo Trace deve conter:

i. Acessos de Leitura aos endereços pares de 0 a 1C.

ii. Acesso de Escrita aos endereços impares de 0 a 1f.

c) Realizar a simulação direta e coletar os dados de:

i. Read Time, Write Time e Total Time nas estatísticas: Memória RAM, Disk e page Table.

ii. Total Time da simulação.

iii. Escrever situação final da tabela de páginas.

d) **Objetivo**: Fazer com que os alunos fixem os conhecimentos desta aula sobre o funcionamento do REA Amnesia para que nas próximas aulas não restem dúvidas de como operar a REA, criar arquivos de arquitetura e arquivos de rastro.

Síntese

Essa aula apresentou a interface do REA Amnesia com o usuário, onde são realizadas as simulações de hierarquia de memória e memória virtual. Também foi apresentado como descrever uma arquitetura no arquivo XML e como é descrito um arquivo de rastro. Além disso, foram apresentados possíveis defeitos nos arquivos de arquitetura e nos arquivos de rastro e como solucionar esses defeitos.

Recursos

Os recursos necessários para esta aula são computadores com o REA Amnesia e um projetor multimídia.

Referências

Hill, M. D. (1989). dinero - cache simulator, version III (Enhanced Version). http://www.ece.cmu.edu/ ece548/tools/dinero/src/doc.h, [accessed on May 25].

APÊNDICE

F

ATIVIDADES

F.1 Operações básicas no Amnesia

1. **Motivação**: Os alunos necessitam de um primeiro contato com o REA Amnesia de modo a incentivar a sua utilização.
2. **Objetivo**: Apresentar o REA Amnesia para os alunos, fazendo com que os mesmos manipulem a interface para observar como as simulações são feitas.
3. **Detalhamento**:
   a) Realizar o download do REA.
   b) Descompactar o arquivo ZIP contendo o REA.
   c) Carregar o REA.
   d) Apresentar a interface e em seguida realizar uma descrição verbal de como o REA funciona:
      i. Para realizar uma simulação inicialmente deve ser carregado um arquivo XML com a arquitetura.
      ii. Detalhar o conteúdo do arquivo de arquitetura, como: tamanho da palavra em BYTES, tamanho da memória RAM em palavras e tamanho de página em palavras.
   e) Em seguida deve ser carregado um arquivo de rastro. A descrição verbal do arquivo de rastro é:
   f) Um arquivo de rastro é um arquivo no padrão DIN [Hill 1989], onde cada linha e composta por dois números, onde o primeiro diz o tipo de acesso e segundo o endereço a ser acessado. (Opcional) Os tipos de acessos são:
      i. “0”: Leitura de dados;

ii. “1”: gravação de dados;

iii. “2”: Busca de instrução;

g) Carregar um arquivo de Arquitetura (02-2) que contenha alguns componentes para que os alunos evidenciem que os componentes são aberto de acordo com o que está descrito no arquivo XML.

h) Carregar um arquivo de rastro (5) para que os alunos verifiquem que os botões RUN e PAUSE ON serão ativados. Descrever a funcionalidade de cada botão.

i) Realizar a execução direta e verificar que ao final da execução o botão SAVE é ativado.

j) Salvar e abrir o log da execução (o log é salvo na pasta local do REA Amnesia).

4. Arquivos

a) Arquitetura: Architecture-02-2-MM-16-VM(PS-4-DM-4-RA-FIFO)-TLB(none)

b) Rastro: TR_5_read_and_write_20_rand_PS_18

5. Pontos a destacar:

a) Com essa atividade é esperado que o aluno entenda o funcionamento do REA e se familiarize com a interface.

b) Aprender como realizar a execução de uma simulação.

c) Salvar um log e verificar as informações no log.

d) O resultado final da execução é:

Taxas / Memória	Acessos de leitura	Acessos de escrita	Page fault (Page table)	Tempo Total
Page Table	20	6	3	320
Principal	44	26		960
Disco	3	3		900
				TT: 1860

retornar ao Plano de aula 1

APÊNDICE

G

TUTORIAL DE UTILIZAÇÃO

Introdução

Este documento tem por objetivo oferecer um suporte técnico a usuários iniciantes no Recurso Educacional Aberto (REA) Amnesia, considerando essencialmente o uso do módulo "Memória Virtual". Ele apresenta os requisitos mínimos para a instalação e uso do Amnesia. Este documento considera o uso de arquivos de rastro no padrão **DIN** (Hill, 1989) para a geração dos acessos às memórias.

Requisitos

1. Monitor de vídeo: as configurações mínimas de resolução é 1024 x 768px
2. JAVA: pode ser obtido através do link:
   a) <https://java.com/pt_BR/download/index.jsp>
   b) É importante ressaltar que todos os testes realizados no REA Amnesia foram realizados com JAVA 7 ou superior. Caso possua uma versão inferior a esta instalada, por favor, obtenha a versão mais atual.
3. REA Amnesia: pode ser obtido através do link:
   a) <http://amnesia.lasdpc.icmc.usp.br/downloads/>
4. Descompactar o arquivo ZIP para qualquer pasta do seu computador.
5. Não separar o arquivo Amnesia.jar das pastas **AmnesiaHelp** e **Configuration**, essas pastas são necessárias para o bom funcionamento do REA.

Execução

1. **Executar o REA**

   Abrir a pasta onde o REA foi descompactado. Em seguida, executar o arquivo **jar** referente ao REA Amnesia conforme mostra a Figura G.1. Os demais arquivos existentes na pasta referem-se às arquiteturas (arquivos **XML** de configuração de ambiente) e rastro de programas (arquivos **TXT** que se baseiam em traços de execuções **DIN** de programas). Os arquivos de arquitetura bem como os rastros, serão explicados a seguir.

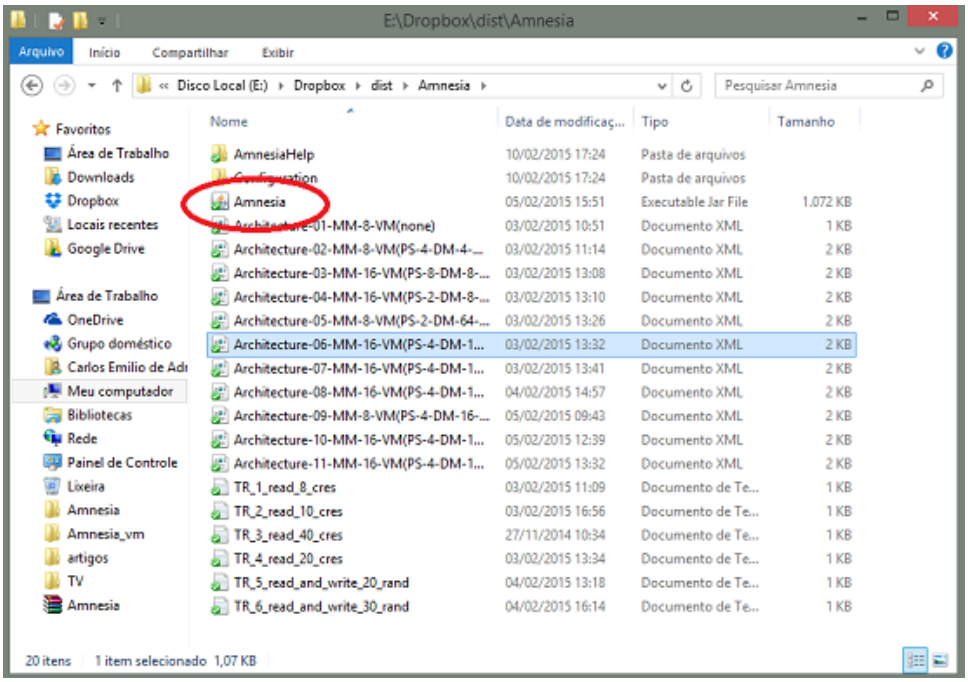


Figura G. 1 – Pasta Local do REA Amnesia

Com o REA carregado, a interface gráfica será semelhante à apresentada na Figura G.2. Caso não consiga abrir a ferramenta, muitas das vezes isso se deve ao fato do JAVA não estar instalado no computador. Volte para etapa de requisitos deste tutorial e realize a instalação do JAVA conforme o item 2.

A Figura G.3 mostra a interface gráfica dividida em 4 áreas. Essa nomenclatura de áreas será utilizada no decorrer desse tutorial.

Na área A da Figura G.3 está localizada a barra de ferramentas, local que permite o carregamento de uma arquitetura previamente configurada, programas representados pelos seus arquivos de rastro, entre outras opções.

Na área B da Figura G.3 estão localizadas as caixas de seleção de cada componente. Os componentes são: Memória Cache, Memória RAM,Tabela de Páginas(*Page Table*), TLB, Disco(*Disk*), Estatísticas (*Statistics*) e Rastro (Trace).

No canto inferior da área B estão localizados os botões **RUN**, **PAUSE ON** e **SAVE**. O botão RUN é utilizado para realizar a execução de um programa de uma vez. No entanto, se

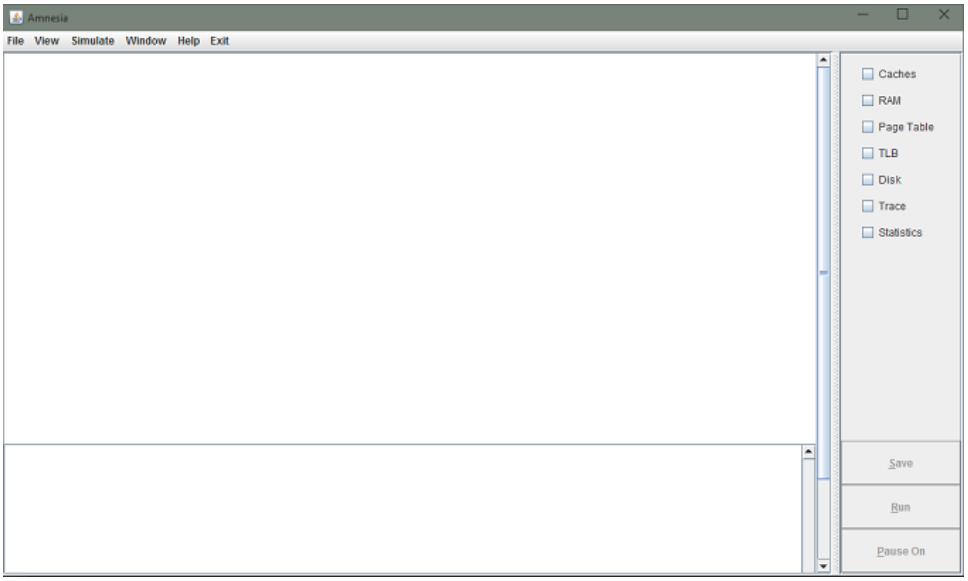


Figura G. 2 – Interface gráfica do REA Amnesia

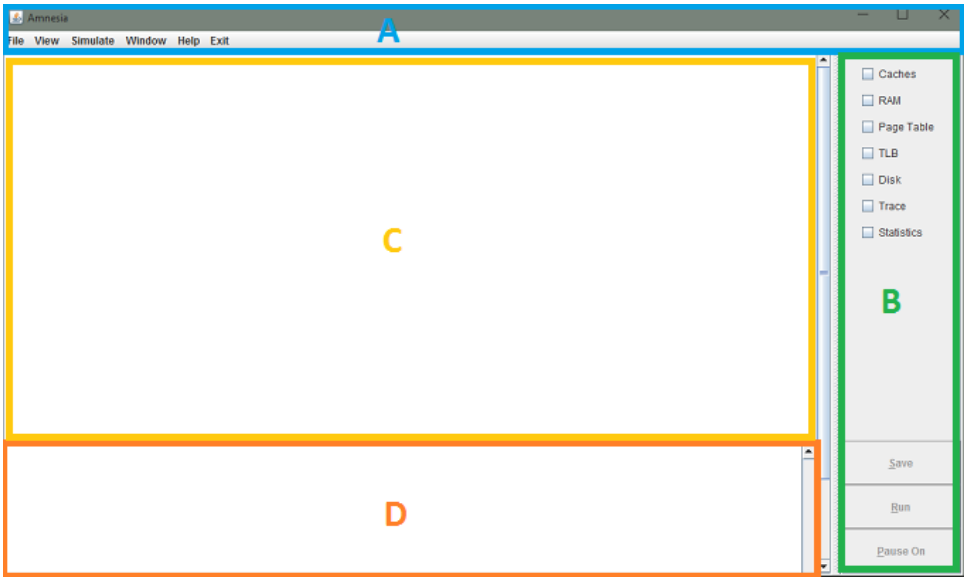


Figura G. 3 – Divisão de áreas do REA Amnesia

o botão **PAUSE ON** é acionado, o botão **RUN** é modificado para **NEXT STEP** e a partir deste momento a execução é feita passo a passo. Após o término da execução, o usuário pode acionar o botão **SAVE** para salvar um **log** contendo as informações da arquitetura utilizada, arquivo de rastro executado, espalhamento dos bits de cada endereço de memória requisitado e as estatísticas de acesso às memórias cache e principal.

Quando uma das caixas de seleção é ativada, o componente selecionado é exibido no campo de visualização, como pode ser visualizado na área C da Figura G.3. Cada componente pode ser movimentado e redimensionado pelo usuário de forma livre nesse campo de visualização. Os componentes são acionados e posicionados automaticamente quando o arquivo de arquitetura é carregado.

O terminal de saída está localizado na área D da Figura G.3, onde são apresentadas informações relacionadas ao carregamento de arquivo de arquitetura e arquivo de rastro. Nesse local também são apresentados detalhes da simulação como: acessos à tabela de páginas, tradução de endereço, troca de páginas, informações de acertos ou falhas nos acessos, entre outras informações.

2. **Abrir um arquivo de Arquitetura**

   Para iniciar uma simulação é necessária a indicação de uma arquitetura mantida em um arquivo formatado no padrão XML. Podem ser configurados nesse arquivo: atributos de processador, memória cache, principal e virtual. Para realizar uma simulação com o objetivo de analisar o funcionamento da memória virtual, os atributos do processador, da memória principal e virtual devem ser obrigatoriamente especificados.

   Desta forma, para o processador deve-se configurar o tamanho da palavra em bytes. Para a memória principal deve-se configurar seu tamanho em palavras, o número de palavras por bloco, o número de ciclos para cada acesso de leitura, o número de ciclos para cada acesso de escrita e o tempo de cada ciclo. Para a memória virtual devem ser especificado o tamanho da páginas, tamanho do disco,número de ciclos para cada acesso de leitura no disco , número de ciclos para cada acesso de escrita no disco, definir o algoritmo de substituição de páginas e características da TLB se desejar que tenha uma.

   Para carregar a arquitetura é necessário abrir o menu **FILE>*LoadArchitecture*** **(CTRL + L)** localizado na área A. Uma caixa de seleção irá aparecer para que seja selecionado a o arquivo **XML**, como mostra a Figura G.4.

   Selecione uma arquitetura e acione o botão **Open**. É importante ressaltar que os arquivos de arquitetura devem estar na pasta local do REA Amnesia, caso contrário o erro “Connection Refused” será apresentado.

3. **Abrir um arquivo de trace**

   Em seguida um arquivo de rastro, que também é chamado de Trace, deve ser carregado. Um arquivo trace contém os acessos realizados por um programa à memória, acessos

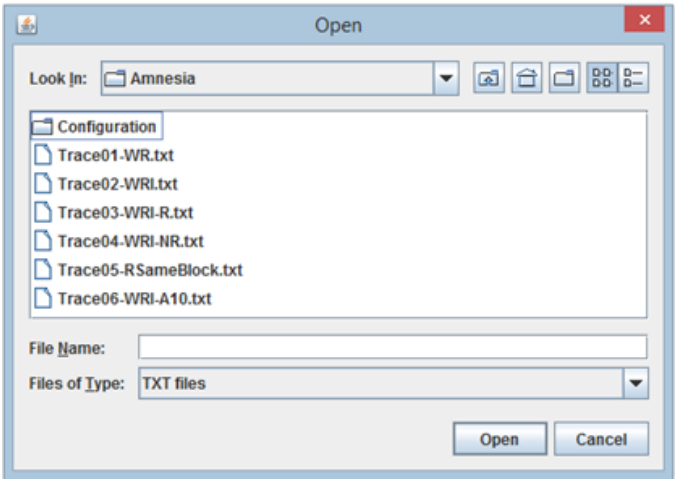


Figura G. 4 – Caixa de seleção de arquivo de arquitetura

de leitura, escrita ou busca de instruções. Para abrir um arquivo de trace, acesse o menu **FILE>Load Trace (CTRL + T)** na área A. Nesse momento será aberta uma caixa de seleção similar a apresentada na Figura G.5.

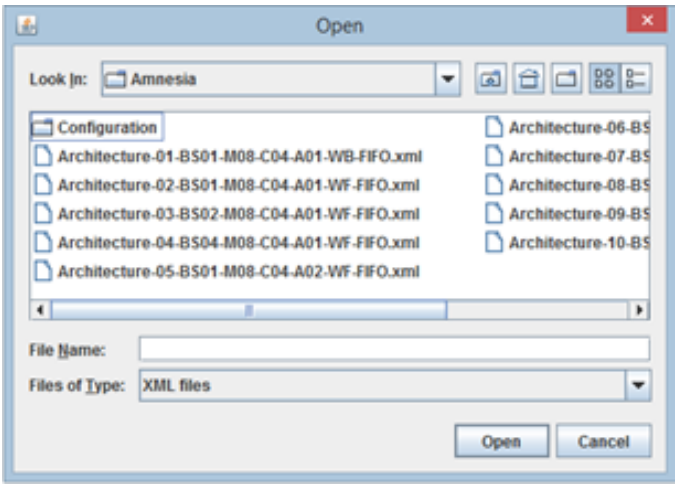


Figura G. 5 – Caixa de seleção de um arquivo Trace

Os arquivos traces são arquivos do tipo **txt** contendo os acessos realizados na memória principal. É importante ressaltar que os arquivos de trace, assim como os arquivos de arquitetura, também devem estar na pasta local do REA Amnesia, caso contrário o erro "*Connection Refused*" será apresentado.

Após o carregamento dos arquivos de arquitetura e **trace**, os componentes presentes na arquitetura são automaticamente abertos e posicionados mas eles podem ser ativados e desativados através das caixas de seleções localizadas na área B. Os componentes serão apresentados na área C, como na Figura G.6. Os componentes podem ser redimensionados e movimentados livremente na área C.

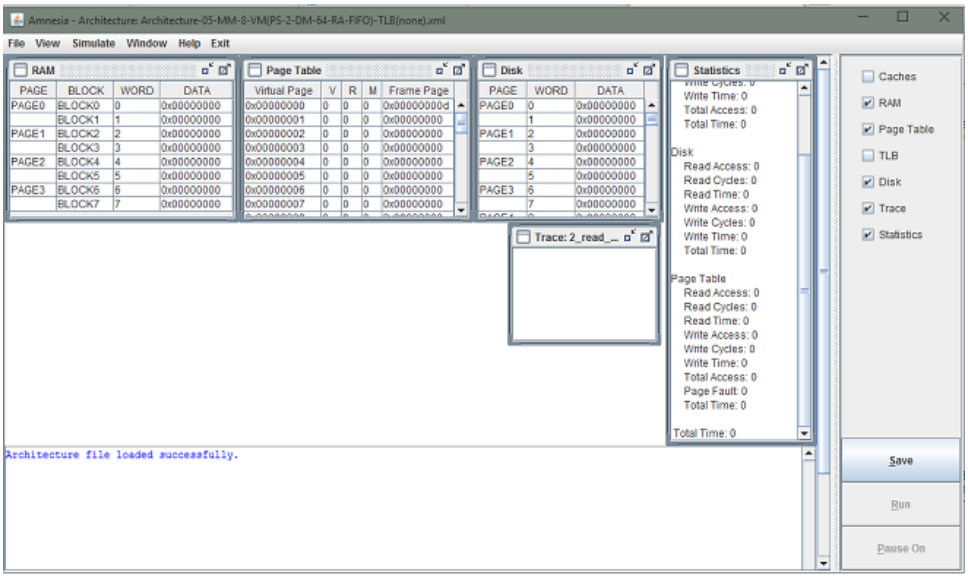


Figura G. 6 – Apresentação dos Componentes: RAM,Page Table, Disk, Staticts e Trace

4. **Execução direta**

Após o carregamento da arquitetura e do arquivo trace os botões **RUN** e **PAUSE ON** são ativados na área B. Caso deseje realizar uma execução direta, acione o botão **RUN** e desta forma todos os passos do trace serão executados. Após a execução, os botões **RUN** e **PAUSE ON** serão desativados e botão **SAVE** será ativado, como pode ser observado na Figura G.7.

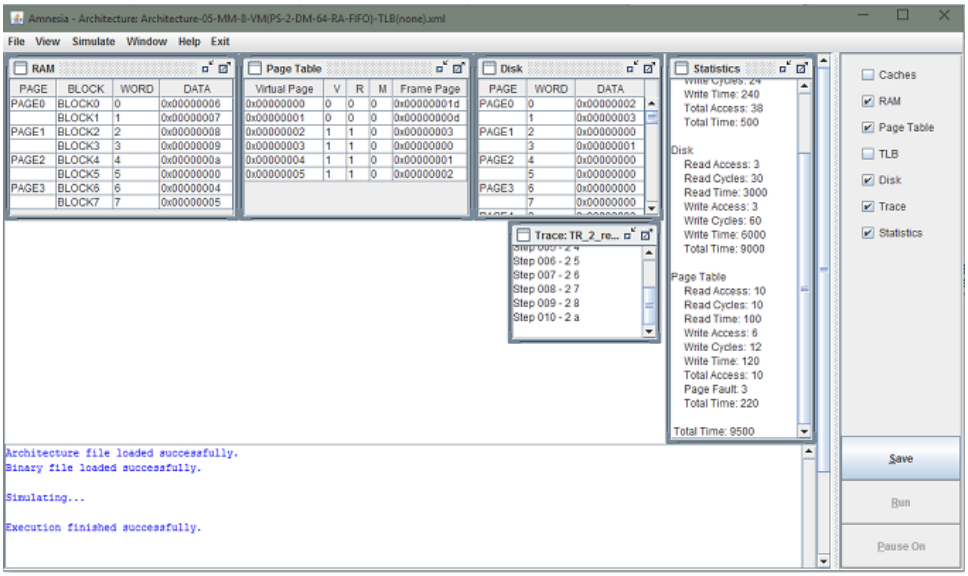


Figura G. 7 – Execução direta da simulação

Observe que as estatísticas de execução do trace são apresentadas no componente **Statistics**. Além disso, os componentes **RAM**, **Page Table**, **Disk**, **Staticts** e **Trace** são apresentados no seu estado final de execução na área D:

Architecture file loaded successfully...(Carregamento da arquitetura realizada com sucesso)

Binary file loaded successfully.....(Carregamento do arquivo trace realizada com sucesso)

Simulating.....(Realizando a Simulação)

Execution finished successfully.....(Execução realizada com sucesso)

5. **Execução Passo a Passo**

   Além da execução direta, é possível realizar a execução passo a passo de um programa a partir de um arquivo de rastro, que provê um *log* detalhado e em tempo real da alocação das informações da memória principal e memória virtual, possibilitando visualizar o tradução de endereço requisitado e troca de páginas para cada instrução executada. Siga os passos até o item 3 deste documento e depois clique sobre o botão **PAUSE ON**. Os botões **RUN** e **PAUSE ON** passarão a apresentar as novas funcionalidades **NEXT STEP** e **PAUSE OFF** respectivamente, como pode ser observado na Figura G.8.

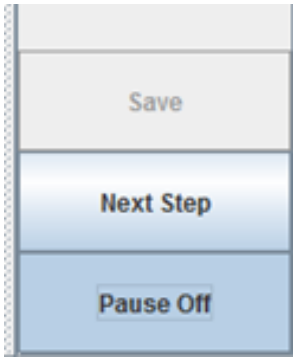


Figura G. 8 – Alteração do botão Run para Next Step

   Execute cada passo do arquivo trace pressionando o botão **NEXT STEP** e acompanhe o *log* detalhado da execução no terminal de saída (Área D), como apresentado na Figura G.9.

6. **Salvamento do *log* de execução**

   Após o término da simulação, um *log* pode ser gravado contendo as informações da arquitetura utilizada, código binário ou arquivo de rastro executado, traduções de endereços realizados, troca de páginas realizados acesso às memórias cache, principal e virtual. Tanto no caso de uma execução direta (**Item 4**) quanto em uma execução passo a passo (**Item 5**), os botões **PAUSE ON**, **RUN** ou **NEXT STEP** são desativados permanecendo ativo somente o botão **SAVE** como pode ser observado na Figura G.10.

   Caso deseje salvar o *log* da execução clique no botão **SAVE**. Assim, o *log* será salvo na pasta local do REA com a seguinte nome:Log-Amnesia-<ANO>-<MÊS>-<DIA>-<HORA>-<MINUTO>-<SEGUNDO>.log

Informações complementares

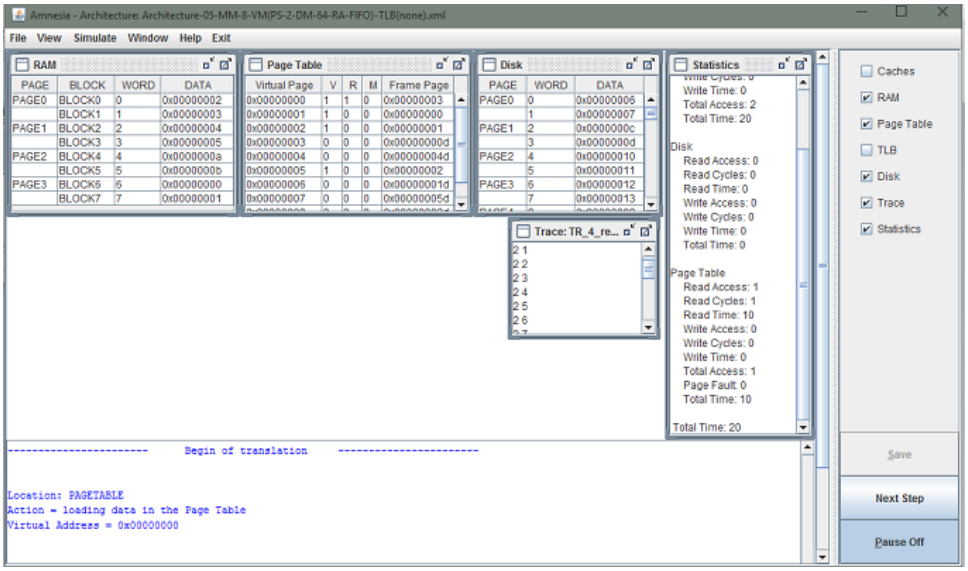


Figura G. 9 – Execução passo a passo de uma simulação

Figura G. 10 – Botão Save habilitado no fim da simulação

- **Arquivos de arquitetura**

  Para iniciar uma simulação é necessária a indicação de uma arquitetura mantida em um arquivo formatado no padrão **XML**. Nesse arquivo, atributos de processador, memória cache, principal e virtual podem ser configurados.

  Para realizar uma simulação com o objetivo de analisar o funcionamento da memória virtual, os atributos do processador, da memória virtual e principal devem ser obrigatoriamente especificados. Desta forma, para o processador deve-se configurar o tamanho da palavra em bytes. Para cada nível da memória deve-se configurar o seu tamanho em palavras, o número de ciclos para cada acesso de leitura, o número de ciclos para cada acesso de escrita, o tempo de cada ciclo, a política de substituição (FIFO ou LRU) e a algumas outras características.

  Para a memória principal deve-se configurar seu tamanho em palavras, o número de palavras por bloco, o número de ciclos para cada acesso de leitura, o número de ciclos para cada acesso de escrita e o tempo de cada ciclo. Um exemplo de um arquivo de arquitetura e apresentado no Código 2

Lembrando que:

O arquivo deve ser salvo no formato XML e outros formatos não são reconhecidos.

O arquivo dever ser salvo na pasta local do REA.

- **Arquivos de trace**

  O arquivo trace ou arquivo de rastro pode ser escrito seguindo algumas orientações:

  Cada linha deste arquivo é composta de uma dupla: rótulo (decimal) e endereço (hexadecimal). Qualquer outra informação é vista como um comentário. Um exemplo (Código 3) pode ser visto na Figura G.11. Os rótulos possíveis são:

  – Rótulo “0”: leitura de dados;
  – Rótulo “1”: gravação de dados;
  – Rótulo “2”: busca de instrução;
  – Rótulo “3”: registro escape (tratado como tipo de acesso desconhecido);
  – Rótulo “4”: registro escape (operação de cache *flush*).

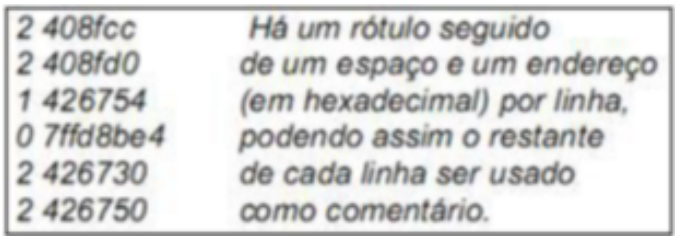


Figura G. 11 – Exemplo de um arquivo Trace (Moraes et al, 2011)

Lembrando que:

O arquivo deve ser salvo no formato TXT e outros formatos não são reconhecidos.

O arquivo deve ser salvo na pasta local do REA.

- **Referências**

Hill, M. D. (1989). **dinero - cache simulator, version III (Enhanced Version)**. Retrieved May 25, 2014, Disponível em: http://www.ece.cmu.edu/ ece548/tools/dinero/src/doc.h

Moraes, M. P; SOUZA, P. S. L. ; Bruschi, S.M. Usando Arquivos de Rastro no Projeto Amnesia. São Carlos: IX Simpósio Internacional de Iniciação Científica da Universidade de São Paulo (SIIC/USP), ICMC – USP 2011. (Resumo de Iniciação Científica).

ANEXO

A

QUESTIONÁRIOS PROPOSTOS POR GAMA

A.1 Expectativa de uso proposto por GAMA (2007)

1. Você costuma usar o computador?

   a) Diariamente

   b) Mais de 3 vezes por semana

   c) Menos de 3 vezes por semana

   d) Você não tem o costume de usar o computador

2. Você tem acesso a Web em casa?

   a) Sim

   b) Parcialmente

   c) Não

   d) Não se aplica

   Caso a resposta seja negativa, qual é o local em que acessa?

   a) No trabalho

   b) Na escola

   c) Em Lan House

   d) Em casa de amigos, parentes...

   e) Outros: Especifique _____

3. O que você acha da ideia de ter a disposição um software ou objeto educacional em qualquer lugar e hora para pode resolver algum problema ligado a disciplina?

a) Muito Bom

b) Bom

c) Indiferente

d) Ruim

e) Péssimo

4. Quanto ao uso do objeto educacional para auxiliar a aprendizagem, pode ser considerada:

a) Vai melhorar o seu conhecimento

b) Vai se manter na mesma

c) Vai saber pouca coisa a mais

d) Não vai fazer diferença

5. Na sua opinião o uso deste objeto vai despertar o interesse do usuário sobre o assunto?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica ou não sabe

6. Na expectativa de uso do objeto você se sente:

a) Temeroso, pois você não se sente a vontade em lidar com a tecnologia.

b) Indiferente, para você tanto faz usar ou não um software

c) Motivado, em poder aprender mais com o uso da tecnologia

d) Confiante, pois você acredita que o uso vai facilitar a sua aprendizagem

7. Na sua opinião o objeto educacional vai permitir o desenvolvimento de um conteúdo novo?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica ou não sabe

8. Na expectativa para o uso do objeto educacional, você pretende usar várias vezes em experiências futuras?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica ou não sabe

9. Gostaria de fazer algum comentário sobre a expectativa de uso do objeto educacional?

_____

_____

_____

_____

_____

A.2 Reação ao uso, proposto por GAMA (2007)

1. Você costuma usar o computador?

   a) Diariamente

   b) Mais de 3 vezes por semana

   c) Menos de 3 vezes por semana

   d) Você não tem o costume de usar o computador

2. Você tem acesso a Web em casa?

   a) Sim

   b) Parcialmente

   c) Não

   d) Não se aplica

   Caso a resposta seja negativa, qual é o local em que acessa?

   a) No trabalho

   b) Na escola

   c) Em Lan House

   d) Em casa de amigos, parentes...

   e) Outros: Especifique _____

3. Usa frequentemente a Web para os estudos?

   a) Sim diariamente

   b) Mais de três vezes por semana

   c) Menos de três vezes por semana

   d) Você não usa

4. Gosta da ideia de poder acessar em casa o objeto educacional em qualquer hora quando precisasse e sem custo?

   a) Sim

   b) Parcialmente

   c) Não

   d) Não se aplica

5. O seu conhecimento sobre o conteúdo antes de utilizar o objeto era:

a) Muito bom

b) Bom

c) Regular

d) Ruim

e) Péssimo

6. O objeto é compreensível e de fácil manuseio?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica

7. Você considera, em relação ao grau de conhecimento adquirido com o uso do objeto educacional, que saiu:

a) Sabendo menos do que antes

b) Sabendo a mesma coisa que antes

c) Sabendo pouca coisa a mais

d) Sabendo muito mais do que antes

8. Em relação a motivação de utilizar os objetos educacionais para auxiliar na aprendizagem ela pode ser considerada:

a) A mesma de quando você não conhecia o objeto (software

b)

c) Diminuiu

d) Aumentou

e) Sofreu momentos de altos e baixos.

9. Na sua opinião o uso deste objeto de aprendizagem despertou o interesse sobre os conteúdos contidos nele?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica

10. A aula se tornou mais interessante com o uso do objeto?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica

11. A atividade é apropriada e o uso do O.E. facilitou a sua compreensão sobre o conteúdo?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica

12. As respostas de todas as operações realizadas no objeto estavam de acordo com suas expectativas?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica

13. Você se sente seguro quanto aos resultados obtidos no objeto?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica

14. Quando se deparou com algum erro teve suporte técnico que lhe ajudou a resolver o problema?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica

15. Está confiante e saberá usar o conhecimento adquirido pelo objeto em uma futura prática?

a) Sim

b) Parcialmente

c) Não

d) Não se aplica

16. O uso do objeto educacional aumentou o nível de seu conhecimento?

    a) Sim
    b) Parcialmente
    c) Não
    d) Não se aplica

17. O gráfico contido no objeto ajudou a compreender melhor a base do conteúdo?

    a) Sim
    b) Parcialmente
    c) Não
    d) Não se aplica

18. Qual a sua opinião sobre o objeto educacional:

    Quanto a relevância: ( )muito importante ( )importante ( )não se aplica

    Quanto a adequação: Sequência: ( )boa ( )regular ( )ruim

    Compreensão: ( )fácil ( )difícil

19. Você usaria o objeto educacional novamente?

    a) Sim
    b) Parcialmente
    c) Não
    d) Não se aplica

20. Gostaria de fazer algum comentário sobre o uso do objeto educacional?

    _____

    _____

    _____

    _____

    _____