CompSim: Um Ambiente para o Ensino Integrado de Arquitetura e Organização de Computadores

Guilherme Esmeraldo[1], Edson Barbosa Lisboa[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) – Crato, CE – Brazil

[2]Instituto Federal de Educação, Ciência e Tecnologia de Sergipe (IFS) – Aracaju, SE – Brazil

guilhermealvaro@ifce.edu.br, edson.barbosa@ifs.edu.br

Abstract. *In technical, technological and higher courses in the areas of computers and electronics, it is common the presence of disciplines that deal with organizational and architectural aspects of computers. The associated contents are complex and the environment is not always available for the accomplishment of the practical activities. Modern methodologies have explored the use of interactive simulation environments and functionally corresponding to real machines for overcoming difficulties in the teaching-learning process in this area. Thus, this work proposes a virtual tool to make more dynamic and attractive the integrated study of hardware and software in courses of computational technologies.*

Resumo. *Em cursos técnicos, tecnológicos e superiores nas áreas de informática e eletrônica, é comum a presença de disciplinas que tratem de aspectos de organização e arquitetura de computadores. Os conteúdos associados são complexos e nem sempre se dispõe do devido ambiente para a realização das atividades práticas. Metodologias modernas têm explorado o uso de ambientes de simulação interativos e funcionalmente correspondentes às máquinas reais para superar dificuldades no processo de ensino-aprendizagem nessa área Assim, esse trabalho propõe uma ferramenta virtual para dinamizar e tornar mais atraente o estudo integrado de hardware e software em cursos de tecnologias computacionais.*

1. Introdução

Em cursos técnicos, tecnológicos e superiores nas áreas de informática e eletrônica, é comum a presença de disciplinas que tratem de aspectos de organização e arquitetura de computadores, programação próxima ao hardware (ou programação de baixo nível) e como acontece a interação entre essas duas partes de um sistema computacional [Stallings, 2014]. Essas disciplinas são muito importantes, pois, com elas, os alunos estudam os componentes do computador, suas principais funções e como eles interagem entre si, além de métodos de avaliação de desempenho de sistemas computacionais [Duenha and Azevedo, 2016].

O processo de ensino-aprendizagem baseado apenas em aulas expositivas e atividades massivamente teóricas esbarra no nível de abstração que o aluno deve ter, dificultando a absorção de conteúdo e, consequentemente, comprometendo o entendimento do funcionamento de um sistema computacional de forma ampla. Portanto, nessas disciplinas, ao longo dos anos, tem sido comum a adoção de metodologias suportadas pelo uso de ambientes virtuais de simulação (software de simulação) como abordagem pedagógica recomendada [Lourenço and Midorikawa, 2004], pois permitem simplificar e dinamizar, através de práticas em cenários próximos aos reais, o aprendizado de conceitos fundamentais, tais como: ciclo de instrução, programação em baixo nível, modos de endereçamento, protocolos de entrada/saída, arquitetura e organização de dispositivos de Entrada/Saída, mapeamentos e políticas de substituição em memória cache, entre outros.

Na literatura voltada para sistemas digitais e arquitetura de computadores, várias abordagens em ambientes de simulação têm sido propostas. Os ambientes variam em granularidade dos objetos (alguns abrangem desde portas lógicas a sistemas completos), tipos de arquiteturas do processador, foco na programação em linguagem de baixo nível e animação da simulação com interface gráfica. Os ambientes de simulação, como o HADES (*Hamburg Design System*) [Hendrich, 2004] e o DEEDS (*Digital Electronic Education Design Suite*) [Donzellini and Ponta, 2013], possibilitam o estudo desde portas lógicas a sistemas microprocessados completos, em diferentes níveis de abstração. No entanto, a configuração e parametrização de componentes específicos não é algo tão simples, devido à complexidade do ambiente proposto. Abordagens como CPUSIm [Skrien, 2001] e SimuS [Silva and Borges, 2016] incluem interfaces gráficas que permitem a configuração e interação com os componentes de hardware e o ambiente de simulação, e focam essencialmente na exploração da programação em baixo nível. Esses simuladores, no entanto, por não incluírem suporte a alguns tipos de modelos de componentes ou ocultarem detalhes importantes para abstração da simulação, tornam-se ambientes de simulação distantes dos reais. Já os ambientes de simulação como o MipsIT são fortemente baseados no processo de animação da simulação em interface gráfica, vinculando a programação de baixo e médio nível com o comportamento dinâmico das unidades funcionais do processador [Kabir, Bari and Haque, 2011]. No entanto, esse simulador só dá suporte a um único tipo de processador. Por outro lado, simuladores como o MPSoCBench [Duenha and Azevedo, 2016], que apresentam componentes de hardware modelados seguindo todas as características de componentes reais, se tornam muito complexos de configurar e interagir.

O ambiente mais adequado vai depender da metodologia adotada e do perfil do curso a ser ministrado. Nem sempre é uma tarefa fácil, mesmo diante de tantas possibilidades, encontrar uma ferramenta que atende ou se adeque às necessidades e especificidades de determinadas disciplinas, em função das realidades locais, ou que contemple todos os conteúdos abordados em uma única disciplina [Nikolic et al., 2009]. Ressalta-se, neste último caso, é comum o uso de mais de um simulador, de forma a se complementarem. Assim, o objetivo desse trabalho é propor uma ferramenta de simulação para apoio pedagógico através de práticas em disciplinas de arquitetura e

organização de computadores. A ferramenta proposta, chamada de CompSim[1], suporta uma abordagem para estudo integrado dos conceitos relacionados aos componentes de hardware, suas funções e interações em si, bem como sua programação em baixo nível.

2. Materiais e Método

Ao longo dos anos, vêm-se realizando análises nos desempenhos de turmas de uma disciplina de Arquitetura e Organização de Computadores de um curso de Bacharelado em Sistemas de Informação, de uma instituição pública. Em diferentes turmas, foram utilizadas as seguintes abordagens pedagógicas: aulas puramente teóricas com resolução de exercícios; aulas teóricas com práticas laboratoriais de programação de computadores em linguagem de baixo nível; e aulas teóricas com utilização de simuladores de organização e arquiteturas de computadores.

Dessas análises, percebeu-se que os melhores desempenhos foram obtidos ao utilizar-se os simuladores, pois além de oferecerem abstrações das estruturas e comportamentos dos componentes de hardware, simplificando a compreensão dos conceitos relacionados, foi possível aumentar o desempenho e a precisão do processo de ensino-aprendizagem, além de tornar as atividades das disciplinas mais atrativas e interativas. No uso dos simuladores, foram pesquisadas e utilizadas diferentes soluções, sendo que algumas delas focavam em aspectos puramente organizacionais (estruturas e funções dos componentes de hardware) e outras em aspectos de arquitetura (programação em baixo nível). Com isso, percebeu-se a necessidade de compor uma solução que pudesse abordar esses dois aspectos de forma integrada.

Na sequência, realizaram-se novas pesquisas na literatura e discussões com professores de disciplinas correlatas para elencar os tópicos onde os alunos apresentam com frequência as maiores dificuldades de aprendizado, os quais, entre eles estão, por exemplo, programação de baixo nível (*Assembly*), modos de endereçamento de memória e organização da memória cache. Além disso, foi possível também estabelecer os principais requisitos pedagógicos, como suporte à navegação livre, aos conteúdos procedimentais e à interatividade, os quais também motivaram a criação de um novo ambiente de simulação.

Os requisitos levantados direcionaram o desenvolvimento da nova solução aqui proposta. Inicialmente foram criados os modelos de componentes de hardware, considerando como requisitos suporte aos parâmetros de configuração e comportamentos de componentes reais, bem como à eficiência na simulação. Em seguida, após a integração dos componentes, para compor um simulador que possibilite a simulação de um computador completo e realização de testes funcionais, criou-se uma interface gráfica para simplificar a configuração dos componentes, controlar a simulação e exibir os resultados e as descrições dos eventos ocorridos durante uma simulação.

Para o desenvolvimento do simulador, utilizou-se o processo de desenvolvimento ágil XP (Beck, 2000), a linguagem de programação Python[2] e a

1 Vídeo de demonstração do CompSim: <https://www.youtube.com/watch?v=8BaGPvvZ6Zo>.
2 Python. Welcome to Python. Disponível em: <http://www.python.org/>. Acesso em: 07 mar. 2017.

ferramenta Nuitka[3], a qual permite a compilação do simulador para execução em diferentes plataformas operacionais (GNU/Linux e Microsoft Windows).

3. O Simulador CompSim

O simulador aqui proposto possui um ambiente integrado com diferentes recursos gráficos, os quais podem ser vistos na Figura 1.

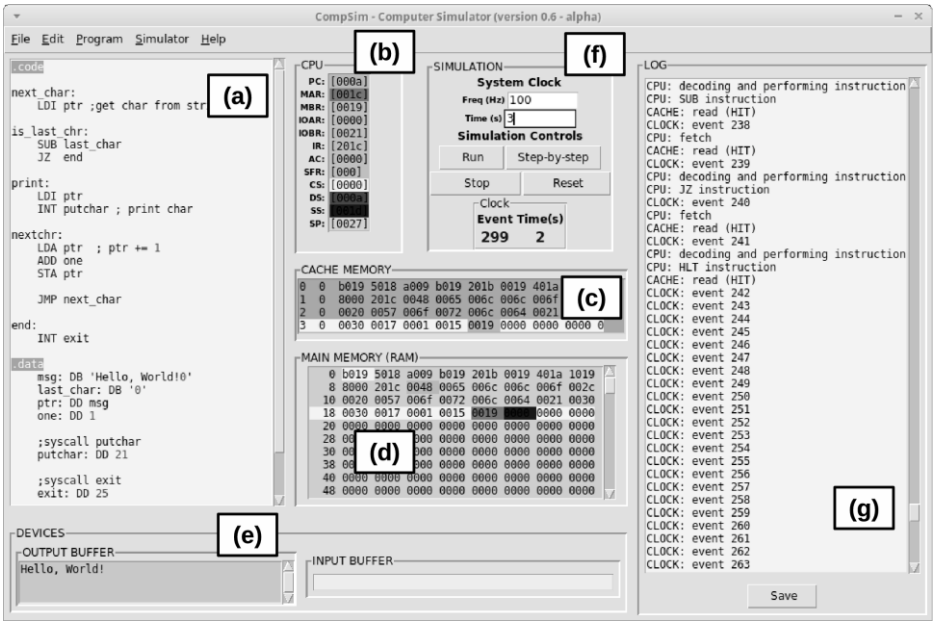

Figura 1. Ambiente Gráfico do simulador CompSim.

Na Figura 1, observa-se os seguintes componentes gráficos: (a) **Editor de código** - com ele é possível codificar uma nova aplicação em baixo nível para execução no simulador; (b) **CPU** - são exibidos, em tempo de execução, os estados assumidos pelos registradores do processador; (c) **CACHE MEMORY** - pode-se observar os estados das respectivas linhas de memória cache; (d) **MAIN MEMORY** - são exibidos os conteúdos de todos os endereços da memória principal; (e) **DEVICES** - são exibidos os conteúdos dos *buffers* de entrada (INPUT BUFFER) e de saída (OUTPUT BUFFER), implementados com comportamentos semelhantes aos de um teclado e de um monitor, respectivamente; (f) **SIMULATION –** esse componente possibilita definir a frequência do relógio do sistema (*clock*) e o tempo total de simulação, bem como controlar a simulação, através dos controles "RUN", "STEP-BY-STEP" (execução passo-a-passo, por ciclo de relógio), "STOP" e "RESET"; (g) **LOG** - neste componente são exibidas as descrições dos eventos gerados pelos componentes de hardware, durante uma simulação. Essas descrições podem incluir, por exemplo, a contagem dos ciclos de relógio, ações de decodificação de instrução, tipos de acessos à memória, ações de substituição de linhas de cache, entre outros.

3 Nuitka. Nuitka Home. Disponível em: <http://www.nuitka.net/>. Acesso em: 07 mar. 2017.

Além desses recursos, o simulador inclui um Montador (*Assembler*), que realiza análises léxica e sintática, para validar um novo programa em linguagem de máquina, e, a partir desta, gerar código binário executável. As subseções a seguir apresentam descrições mais detalhadas dos modelos propostos de componentes de hardware e do montador.

3.1 Descrição dos Modelos de Componentes de Hardware

O processador proposto consiste de um modelo teórico de 16-bits, com as seguintes características: inclui um contador de programa, uma unidade lógica e aritmética (ULA), inclui espaço de endereçamento diferenciado para entrada/saída e para acesso à memória principal; suporte aos modos de endereçamento imediato, direto, indireto, registrador e implícito; um banco de registradores para suportar as operações de busca de instruções do programa, decodificação de instrução, operações aritméticas e lógicas, através de um registrador acumulador e da ULA, operações de entrada e saída e de acesso à memória e pilha de programa; inclui dezesseis 16 instruções de baixo nível; e suporte a operandos inteiros de 16-bits com sinalização (*signed int*) e a strings (*bytes*).

O modelo de memória principal (RAM) suporta palavras de 16-bits, com blocos de 8 palavras cada, e pode ser configurado para incluir de 64 à 512 blocos, obtendo assim as capacidades de armazenamento mínima de 8KB e máxima de 64KB. O modelo de memória cache proposto possui número de linhas parametrizável, onde cada linha suporta 8 palavras de 16-bits (configuração necessária para guardar um bloco da memória proposta). No componente de memória cache, pode-se ainda configurar o tipo de mapeamento, as políticas de atualização e de substituição. Os dispositivos de entrada e saída apresentam comunicação serial e são utilizados para interfacear, durante a simulação, o computador teórico com o usuário. Por fim, o barramento de sistema realiza a interconexão física entre todos os modelos de componentes de hardware propostos.

3.2 A Linguagem de Baixo Nível e o Montador

De forma a simplificar o aprendizado em programação em nível de hardware, criou-se uma linguagem de programação de baixo nível composta por apenas 16 instruções, as quais são representadas por mnemônicos/ semelhantes aos de uma linguagem *Assembly* de uma arquitetura real. Algumas instruções não incluem operandos e outras apenas um operando (instruções com número de operandos reduzido são mais simples de compreender e de utilizar em programas) e são utilizadas em ações de movimento de dados, operações lógicas e aritméticas, acesso à pilha do programa, alteração do fluxo de execução do programa e para encerramento de execução. O Montador (*Assembly*) proposto segue o mesmo comportamento de montadores de dois passos. No primeiro passo, o montador realiza as análises léxica e sintática, a construção da tabela de símbolos e de uma representação intermediária do código. Já no segundo passo, a partir da tabela de símbolos e da representação intermediária do programa, o montador gera o programa binário final para a arquitetura alvo.

No simulador, o montador pode ser utilizado de duas maneiras. Na primeira, ele realiza uma análise no código do programa e apresenta um relatório. No relatório, são exibidos a tabela de símbolos, os endereços dos segmentos de texto, dados e pilha, bem como o código binário gerado. Na segunda forma, o montador gera o código binário e o mesmo é copiado para a memória principal, permitindo assim o início da simulação (execução do código nos componentes de hardware do simulador).

3.3 Integração entre Interface Gráfica e Componentes de Hardware

A interface gráfica do simulador está integrada aos componentes de hardware de forma que é possível configurar, executar e compreender as interações entre os componentes, durante uma simulação. A Figura 2 apresenta, em detalhes, os componentes CPU, CACHE MEMORY, MAIN MEMORY, SIMULATION e LOG.

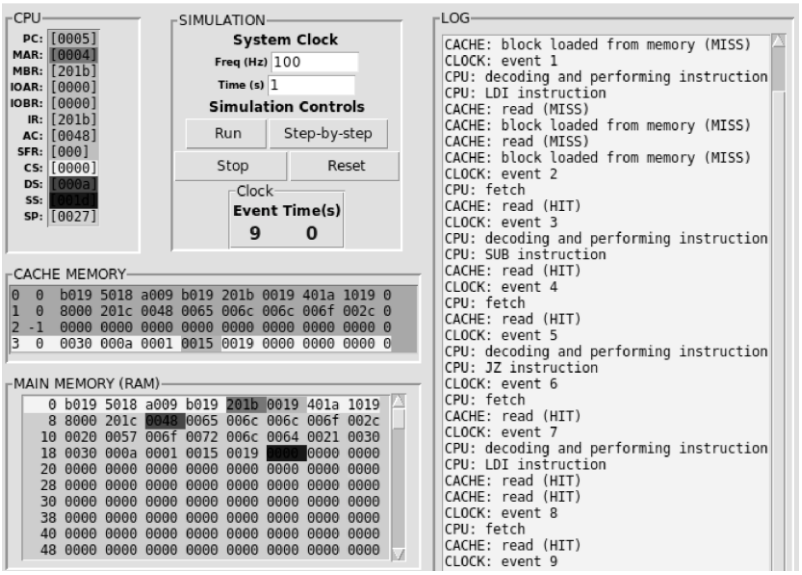

Figura 2. Interação entre componentes de hardware e interface do CompSim.

Na Figura 2, podemos ver, à esquerda e acima, no componente gráfico CPU, que os valores de alguns dos registradores estão sendo destacados em diferentes cores (verde, amarelo, azul, vermelho, etc.). Os valores destacados, dependendo do registrador, referem-se a determinados endereços da memória principal, durante uma simulação. Os endereços de memória principal apontados por esses registradores serão também destacados com as respectivas cores, no componente MAIN MEMORY (exibido abaixo e à esquerda, na Figura 2). Este recurso permite o acompanhamento dinâmico de quais endereços estão sendo acessados pelos diferentes registradores de endereçamento à memória, em uma simulação do tipo STEP-BY-STEP. Além disso, no componente CACHE MEMORY (exibido ao centro e à esquerda, na Figura 2), são destacadas a linha e a palavra que estão sendo acessadas em um determinado momento.

Ainda na Figura 2, ao centro e acima, observa-se os controles de simulação (SIMULATION). Com eles é possível gerar os estímulos do relógio de sistema, bem como parar e reiniciar uma simulação. Por fim, do lado direito, ainda na Figura 2, o componente LOG apresenta descrições detalhadas dos eventos de cada componente de

hardware, na medida em que ocorrem durante uma simulação. Na Figura 2, no relatório de LOG, podemos ver descrições de eventos de relógio de sistema, processador, memória cache e vídeo.

4. Resultados Esperados e Trabalhos Futuros

O uso de simuladores é uma prática pedagógica frequente em disciplinas de organização e arquitetura de computadores. Acredita-se que simulador aqui proposto permitirá otimizar o processo de ensino-aprendizagem, pois, através de uma abordagem para estudo integrado dos conceitos relacionados aos componentes de hardware e à programação em baixo nível, pode-se estabelecer um programa de disciplina que aborde todos os conteúdos práticos necessários e que seja focado na aprendizagem significativa. A tendência em ambientes virtuais de simulação é dar suporte a diferentes arquiteturas de processadores, plataformas de hardware especificadas em linguagem de descrição de hardware (HDL) e possibilitar a interação com outros ambientes por meio de cossimulação. Adicionalmente, é fundamental o ambiente gerenciar a metodologia pedagógica de forma interativa com os atores envolvidos em tempo real, possibilitando uma comunicação eficiente e integração com ambientes de educação à distância, como por exemplo, o Moodle.

Referências

Beck, K. (2000) Extreme programming explained: embrace change. addison-wesley professional.

Donzellini, G. and Ponta, D. (2013) From Gates to FPGA: Learning Digital Design with Deeds. In: Proceedings of the Third Interdisciplinary Engineering Design Education Conference – IEDEC. pp.41-48.

Duenha, L. and Azevedo, R. (2016) Utilização dos Simuladores do MPSoCBench para o Ensino e Aprendizagem de Arquitetura de Computadores. In: International Journal of Computer Architecture Education (IJCAE), V. 5, n. 1. pp 26-31.

Hendrich, N. (2002) From CMOS-Gates to Computer Architecture: Lessons Learned from Five Years of Java-Applets. In: Proceedings of the 4th European Workshop on Microelectronics Education, EWME. Pp 23-24.

Kabir, M. T., Bari, M. T. and Haque, A. L. (2011) VisiMips: Visual Simulator of MIPS32 Pipelined Processor. In: International Conference on Computer Science & Education (ICCSE). p. 788–793.

Lourenço, A. E. and Midorikawa, E. T. (2004) Ensino de arquitetura de computadores utilizando simuladores completos. Congresso Brasileiro de Engenharia - COBENGE 2004, Brasila.

Nikolic, B., Radivojevic, Z., Djordjevic, J. and Milutinovic, V. (2009) A Survey and Evaluation of Simulators Suitable for Teaching Courses in Computer Architecture and Organization. IEEE Transactions on Education, Vol. 52, No. 4.

Silva, G. P and Borges, J. A. dos S. (2016) SimuS: Um Simulador para o Ensino de Arquitetura e Organização de Computadores. In: International Journal of Computer Architecture Education (IJCAE). V. 5, n. 1. pp 7-12.

Skrien, D. (2001) CPU Sim 3.1: A tool for simulating computer architectures for computer organization classes. In: Journal on Educational Resources in Computing (JERIC), 1(4).

Stallings, W. (2014) Operating Systems: Internals and Design Principles| Ed: 8. Pearson.