Uma Ferramenta para Exploração do Ensino de Organização e Arquitetura de Computadores

Guilherme A. Esmeraldo
Laboratório de Sistemas Embarcados e Distribuídos (LEDS)
Instituto Federal do Ceará (IFCE)
Crato, Ceará, Brasil
guilhermealvaro@ifce.edu.br

Edson Barbosa Lisboa
Laboratório de Estudos Avançados em Sistemas Eletrônicos (LEA)
Instituto Federal de Sergipe (IFS)
Aracaju, Sergipe, Brasil
edson.barbosa@ifs.edu.br

***Resumo*—As disciplinas que tratam dos aspectos de Organização e Arquitetura de Computadores (OAC) estão presentes nos cursos de computação e engenharias, onde o nível de detalhamento desses conhecimentos varia de acordo com o perfil de cada curso. A complexidade dos conceitos envolvidos demanda metodologias desafiadoras e ferramentas computacionais que facilitem o processo de ensino-aprendizagem em OAC. Assim, este trabalho apresenta um ambiente de simulação que possibilita explorar os conteúdos de OAC em diferentes níveis de detalhes, a depender das necessidades curriculares específicas. Denominada de CompSim, a ferramenta implementa o conceito de projeto baseado em plataforma, apresenta um suporte gráfico intuitivo e atrativo para o processo de ensino-aprendizagem em OAC, o que tem facilitado o entendimento de muitos conceitos abstratos e, por isso, tem sido muito bem avaliada pelo público alvo ao qual tem sido aplicada.**

***Palavras-chave*—Apoio ao Aprendizado; Organização e Arquitetura de Computadores; Simulador; Plataforma Simulável.**

I. INTRODUÇÃO

A disciplina de Organização e Arquiteturas de Computadores (OAC) está entre as principais disciplinas de cursos superiores nas áreas de Computação, Engenharia Eletrônica, dentre outras [6][10-11], e consiste no estudo dos componentes de um sistema computacional, e das respectivas funções, bem como dos aspectos e atributos visíveis ao programador [1], abrangendo os conhecimentos necessários à operação, projeto e programação eficiente de sistemas computacionais [8-9].

Esses conhecimentos variam de acordo com o perfil de cada curso. Existem currículos de referência, como os apresentados em [10] e [11], que trazem propostas de conteúdos das disciplinas de OAC em cursos de Computação. Esses currículos incluem diferentes tópicos, porém cinco deles são apontados como fundamentais a qualquer curso básico, que são: 1) Lógica e Sistemas Digitais; 2) Representação de dados em nível de máquina; 3) Organização da máquina em nível de montagem, 4) Organização e arquitetura de memória; e 5) Interfaceamento e comunicação.

Além desses objetivos, os currículos incluem habilidades práticas que são frequentemente desenvolvidas em laboratório. Essas habilidades permitem que os alunos possam verificar os conhecimentos obtidos em aulas teóricas, pela observação e exploração das características dos sistemas atuais [9]. Em [10], define-se as seguintes habilidades práticas que devem necessariamente ser desenvolvidas em cursos de OAC: 1) Projeto de blocos básicos de um computador; 2) Uso de ferramentas CAD (*Computer-Aided Design*) para projetar blocos básicos; 3) Conversão de dados numéricos entre bases; 4) Escrita de programas simples ao nível de máquina (*Assembly*) para diferentes cenários de uso do computador; 5) Demonstração de como construções de linguagens de alto nível são implementadas ao nível de máquina; e 6) Cálculo dos tempos de acesso aos dados, considerando diferentes configurações de memória principal e cache, bem como diferentes tipos de referência a dados e instruções.

Nesse contexto, percebe-se que os simuladores tornam-se peças-chave para o desenvolvimento das habilidades práticas estabelecidas em cursos de OAC. O uso de simuladores computacionais, ou ambientes virtuais, como prática pedagógica complementar, não é uma atividade nova [14]. Estudos, como os apresentados em [12-15], mostram que, ao se utilizar ambientes virtuais para fins educativos, foi possível aumentar o desempenho acadêmico em cursos de tecnologia e engenharia. Os simuladores são ferramentas importantes no processo de apropriação do conhecimento, pois possibilitam o desenvolvimento de habilidades e experiências práticas, de forma assíncrona, em cenários virtuais que se assemelham aos reais [20]. São também fundamentais para compor laboratórios específicos na ausência de infraestrutura [13] ou ainda quando há necessidade de se reduzir custos, realizar configurações rápidas e obter resultados instantâneos [12]. Os simuladores são particularmente úteis para representação ou abstração de cenários complexos [17-18] e frequentemente abordam os conteúdos presentes no estado da arte [20].

Algumas ferramentas especializadas são adotadas ou adaptadas para compor ambientes virtuais de aprendizagem. Porém, apesar de oferecerem um ambiente completo de projeto, não se pode concluir se, nesses casos, o suporte educacional é efetivo [16].

Desta maneira, considerar os aspectos didático-pedagógicos torna-se o grande desafio no projeto de qualquer ambiente de aprendizagem. Além destes, outros requisitos como, por exemplo, desempenho, facilidade de uso, interface intuitiva, portabilidade, modularidade, acessibilidade, interatividade e independência [12][14], são fatores importantes para sua aplicabilidade e necessários para estimular e permitir que o estudante possa desenvolver seu próprio aprendizado.

Este trabalho apresenta uma abordagem para o processo de ensino-aprendizagem, que considera os aspectos discutidos anteriormente, respaldada pelo apoio pedagógico de uma ferramenta de simulação em organização e arquitetura de computadores. A ferramenta proposta, chamada de CompSim, busca reunir as principais características dos simuladores do estado da arte, bem como o suporte necessário para o desenvolvimento das habilidades práticas constantes nos

currículos de referência em [10-11]. Este artigo está dividido da seguinte maneira: A Seção II apresenta os trabalhos relacionados e como eles contribuíram para o desenvolvimento do projeto aqui proposto, que é apresentado na Seção III. Na Seção IV são apresentados os resultados obtidos até o presente momento. Por fim, a Seção V apresenta as conclusões e demarca os trabalhos futuros.

II. Trabalhos Relacionados

A literatura científica inclui diferentes abordagens em ambientes virtuais, como os apresentados em [2-5][7-9][19] [23-24], para apoio à disciplina de OAC. Pela grande variedade de opções, algumas pesquisas, como as apresentadas em [8-9] [19-20], realizaram comparações entre os simuladores, de acordo com métricas preestabelecidas.

O estudo em [19] considera os seguintes elementos como os fatores principais para comparações de simuladores para OAC: 1) desempenho de simulação; 2) flexibilidade, para suportar diferentes configurações e níveis de abstração dos componentes de hardware; 3) nível de detalhes do modelo de simulação; e 4) precisão dos resultados de simulação.

Segundo [8], define-se as seguintes métricas de comparação: 1) tipo de licença de distribuição, sendo que as livres possuem maior impacto positivo por permitirem modificações no código para alteração ou adição de funcionalidades; 2) suporte à simulação de diferentes tipos de arquitetura; 3) granularidade de parametrização; 4) suporte à simulação de aplicações de usuário e de sistemas completos, incluindo sistema operacional; 5) suporte a *debugging* e *tracing*, pois permitem analisar respectivamente o estado e a carga de trabalho da máquina simulada; e 6) disponibilidade de documentação para suporte ao aprendizado no uso do simulador.

De acordo com [20], foram categorizados os simuladores estudados em: 1) máquinas históricas, onde justificou-se sua importância na compreensão da evolução dos computadores; 2) lógica digital, para uso de abordagens *botton-up* (inicio em transistores MOS até os circuitos combinacionais); 3) máquinas hipotéticas simples, pois permitem focar a atenção em conceitos importantes, desde que não considera detalhes complexos dependentes de arquitetura; 4) conjunto de instruções intermediárias, por incluírem mais detalhes da máquina alvo, trazem um modelo de programação mais realístico; 5) microarquitetura avançada, podem ser utilizados em cursos avançados para investigação do impacto de técnicas de projeto, como *pipelining*, *branch-prediction* e paralelismo ao nível de instrução, sobre o desempenho e complexidade do computador; 6) multiprocessadores, indicados para cursos de OAC mais avançados, permitem o estudo das características de sistemas computacionais multiprocessados, como redes de interconexões e memória compartilhada; 7) subsistemas de memória, utilizados para modelagem e análise de desempenho sob diferentes configurações da hierarquia de memórias e de memória cache.

Nos estudos apresentados por [9], utilizou-se dois grupos de métricas na comparação entre simuladores. O primeiro grupo considerou a cobertura dos tópicos fundamentais recomendados para cursos básicos de OAC [10]. O outro grupo considerou as principais características encontradas nos simuladores avaliados, que são: 1) escopo e complexidade, que refletem as características das arquiteturas estudadas (básica ou avançada); 2) suporte ao projeto de novas arquiteturas; 3) apresentação visual, permite a visualização do trabalho interno de um sistema computacional; 4) fluxo de simulação interativo, onde o usuário pode controlar toda a simulação; 5) granularidade de simulação, onde a abstração da simulação pode ser em nível de *clock*, de instrução ou de programa; 6) detalhes de implementação, que permitem visualizar os detalhes de operação das estruturas de blocos; 7) suporte ao aprendizado a distância, considera o uso do simulador a distância via Internet.

A análise do estado da arte evidencia que não é uma tarefa fácil, mesmo diante de tantas possibilidades, encontrar um ambiente de ensino-aprendizagem que atenda ou se adeque a determinadas disciplinas, no contexto de OAC, em função de realidades locais, metodologia adotada e perfil do curso a ser ministrado. Portanto, a abordagem aqui proposta busca integrar as principais características levantadas nesses estudos comparativos e suporte aos desenvolvimento das habilidades práticas básicas que devem ser realizas em cursos de OAC [10]. A seção a seguir detalha o simulador proposto.

III. O Simulador CompSim

A. *Apresentação*

CompSim é um ambiente integrado para dar suporte ao processo de ensino aprendizado em OAC. Ele possui um ambiente gráfico, que pode ser visto na Figura 1, com os seguintes recursos: 1) parametrização dos componentes de um sistema computacional; 2) edição, análise, montagem e carregamento em memória de código em nível de máquina; 3) gerenciamento da dinâmica das simulações; 4) acompanhamento dos *status* dos componentes de hardware, durante uma simulação; 5) análise das estruturas de dados e pilha do programa, durante uma simulação; e 6) avaliação do desempenho do sistema ao final de uma simulação. O simulador foi codificado e compilado com Python v3.5 e está disponível para os sistemas operacionais MS Windows e GNU/Linux (32 e 64 bits).

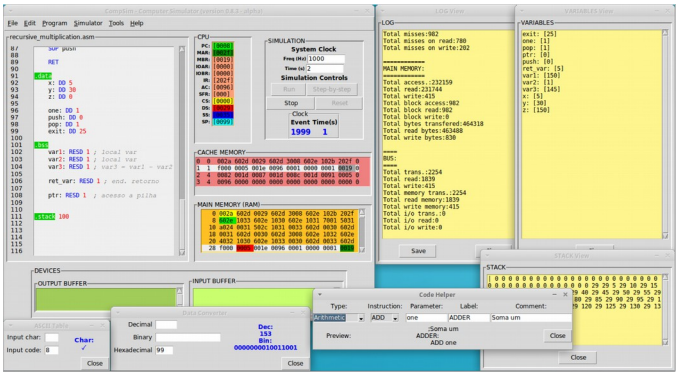

Fig. 1. Ambiente gráfico do CompSim.

B. *Organização da Plataforma*

O simulador CompSim inclui uma plataforma pré-definida de hardware simulável e parametrizável, chamada de Mandacaru, que segue a abordagem de Projeto Baseado em Plataformas [22], a qual estabelece uma arquitetura de hardware pré-definida, que pode ser customizada de acordo com requisitos de uma determinada aplicação.

A plataforma Mandacaru inclui os seguintes componentes de hardware: um processador conceitual, uma memória RAM, uma memória cache, periféricos de entrada (Teclado) e de saída (Vídeo), bem como uma estrutura de comunicação composta por dois barramentos compartilhados, sendo um deles de sistema e o outro de periféricos. A Figura 2 apresenta a organização da plataforma Mandacaru.

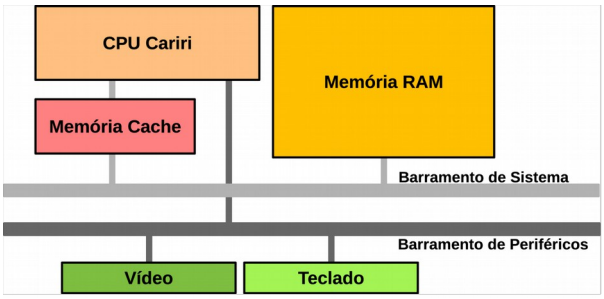

Fig. 2. Plataforma Mandacaru.

O modelo de processador proposto, chamado de Cariri, inclui:

- Palavra de 16-bits (dados e endereços);
- Unidade Lógica e Aritmética (ALU);
- Unidade de Controle (*Control Unit*);
- Somador (*Adder*) para cálculo do endereço da próxima instrução;
- Banco de registradores, para suportar busca/decodificação de instrução, operações lógicas/aritméticas, de entrada/saída, de acesso à memória e pilha de programa e de transferência de controle. São eles:
  - Acumulador (AC);
  - Apontador de instruções (PC);
  - Registrador de Instrução (IR);
  - Endereçamento e *Buffer* de Memória (MAR e MBR);
  - Endereçamento e *Buffer* de I/O (IOAR e IOBR);
  - Segmentos de Código, Dados e Pilha (CS, DS e SS);
  - Apontador de Pilha (SP); e
  - *Flags* de *status* de operação (SFR), com códigos para "Zero", "*Signal/Negative*" e "*Carry/Overflow*".

A Figura 3 apresenta o *datapath* do processador Cariri. Na figura, as linhas em vermelho representam os caminhos de dados, as linhas em azul os caminhos de endereços e as linhas em verde os sinais de controle. Ainda na Figura 3, podemos observar as interconexões entre os blocos *Adder*, *Control Unit* e ALU, com os registradores do processador.

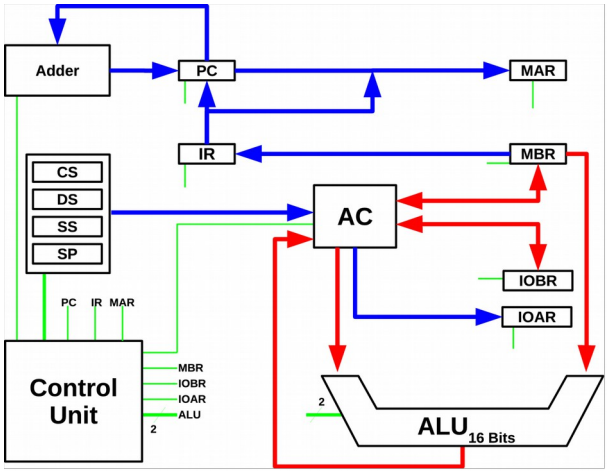

Fig. 3. Datapath do processador Cariri.

O modelo proposto para a memória RAM permite armazenar dados e instruções de 16 bits e possui endereços de 16 bits, sendo que os 13 bits mais significativos são utilizados para endereçamento de linha e os 3 bits menos significativos para endereçamento de coluna. O modelo de memória proposto é parametrizável, sendo possível endereçar de 64 a 8k linhas de memória, onde cada linha possui 8 colunas, totalizando-se assim uma faixa que varia de 512 a 64k diferentes endereços. A Figura 4 apresenta um diagrama da organização da memória RAM proposta, onde observa-se, ao centro, a Matriz de Dados, organizada em 8k linhas x 8 colunas x palavras de 16 bits; à esquerda, as linhas de endereço; e, à direita, o barramento de dados. As linhas em azul representam os caminhos de endereço e as em vermelho os caminhos de dados.

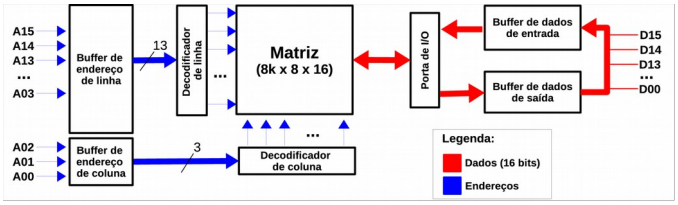

Fig. 4. Organização da memória RAM.

O modelo de memória cache proposto possui número parametrizável de linhas (de 4 a 128), onde cada linha pode armazenar 8 palavras de 16 bits (configuração que permite guardar um bloco do modelo proposto para a memória RAM). Na cache proposta, pode-se ainda configurar o tipo de mapeamento (*Direct Mapped*, *Fully Associative* e *Set Associative*), as políticas de atualização (*Write-Through*, *Write-Back*, *Write Allocate* e *Write Around*) e de substituição (FIFO, LRU e *Random*).

Esses recursos podem ser compreendidos como apoio ao aprendizado em: 1) projetos de sistemas digitais, ao tratar, por exemplo, do projeto dos blocos básicos do processador, como ULA, UC e registradores; 2) organização básica da máquina de Von Neumann, bem como de processadores, tipos e hierarquia

de memória, periféricos e interconexões; 3) entrada/saída (I/O), incluindo os conceitos de *handshaking*, *buffering*, entrada e saída programada/orientada à interrupção e protocolos de comunicação.

C. Arquitetura do Processador

A arquitetura do processador Cariri inclui um projeto simples, mas com o suporte necessário para conduzir o aprendizado de programação em nível de máquina para solução de problemas de baixa ou alta complexidade.

De uma forma geral, o processador Cariri inclui as seguintes características:

- Arquitetura baseada em acumulador;
- 16 instruções monociclo de baixo nível (*Instruction Set Architecture – ISA*);
- Operandos inteiros de 16 bits com sinalização (*signed* int) e caracteres (bytes);
- Suporte aos modos de endereçamento imediato, direto, indireto, via registrador e implícito; e
- Espaço de endereçamento diferenciado para entrada/saída e memória principal.

As Figuras 5(a) e 5(b) ilustram, respectivamente, os formatos propostos para representação de dados do tipo inteiro de 16-bits com sinalização (formato em complemento de dois) e caractere, que utiliza apenas os 8 bits menos significativos (os demais bits não são utilizados).

Nas instruções do processador Cariri, os 4 bits mais significativos indicam o código da operação (*opcode*) e os demais bits codificam o operando, como pode ser visto na Figura 5(c). A Tabela I apresenta todas as instruções, com os respectivos códigos e descrições.

A ISA inclui instruções para operações aritméticas (ADD e SUB), lógicas (NAND e SHIFT), de transferência de dados (MOV, LDA, STA, LDI, STI e SOP), de transferência de controle (JMP, JZ, JN, CALL, RET e INT) e de entrada/saída (INT).

Para assegurar a simplicidade do modelo proposto e superar a limitação do número de instruções, a ISA do processador foi projetada para garantir a versatilidade funcional, de forma que:

- Suporta diferentes modos de endereçamento para incluir maior número de recursos no processador, como por exemplo: manipulação de pilha de programa e de estruturas de dados; implementação de construções de linguagens de alto nível, como funções, condicionais e interações; e diferentes métodos de entrada/saída.
- Suporta, através de um programa em baixo nível para execução no processador, a implementação de operações mais complexas. Por exemplo, as operações lógicas AND, OR e NOT podem ser implementadas a partir da instrução NAND (*universal gate*), assim como as operações aritméticas MULT e DIV podem ser implementadas com as instruções ADD, SUB e SHIFT.
- Agrupa mais de uma operação, em uma única instrução, desde que estejam no mesmo contexto. Por exemplo, a instrução SOP agrupa as operações de pilha de programa PUSH e POP (ver descrição da instrução SOP na Tabela I).

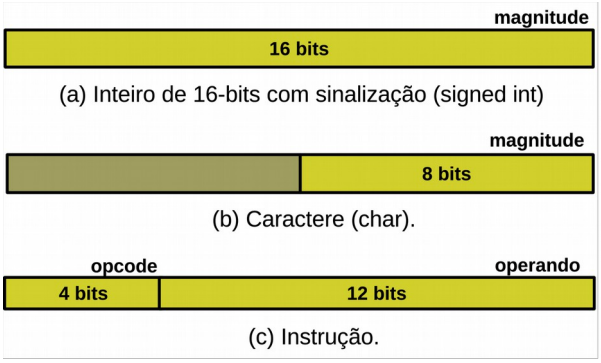

Fig. 5. Formato de dados e instruções no processador Cariri.

TABELA I. ISA DO PROCESSADOR CARIRI.

Código	Instrução	Descrição
0000	LDA X	Carrega da memória para AC.
0001	STA X	Grava AC em memória.
0010	INT X	Executa uma das seguintes pseudo-rotinas: GetChar, PutChar e Halt.
0011	CALL X	Chamada de procedimento.
0100	ADD X	Soma AC com operando em memória.
0101	SUB X	Subtrai operando em memória de AC.
0110	SOP X	Adiciona/remove AC no/do topo da pilha de programa. O operando em memória indica *push* ou *pop*.
0111	MOV I MOV R	Carrega operando no AC.
1000	JMP X	Desvia a execução do programa para X.
1001	JN X	Desvia a execução do programa para X, se a flag 'Signal' = '1'.
1010	JZ X	Desvia a execução do programa para X, se a flag 'Zero' = '1'.
1011	LDI Y	Carrega da memória para AC
1100	STI Y	Grava AC em memória.
1101	NAND X	Operação binária NAND entre AC e um operando em memória.
1110	SHIFT X	Operação binária Shift em AC. O operando em memória aponta o sentido da operação.
1111	RET	Retorno de procedimento.

Legenda: I: Imediato, X: Direto, Y: Indireto, R: Registrador.

Além das instruções apresentadas na Tabela I, o montador (*Assembler*) da linguagem de programação do processador Cariri inclui as pseudo-instruções (diretivas) na Tabela II para definição de dados. Com elas, é possível alocar variáveis e sequências (*arrays*) de inteiros e bytes.

TABELA II. PSEUDO-INSTRUÇÕES DO MONTADOR DO PROCESSADOR CARIRI.

Pseudo-Instruções	Descrição
DD	Reserva memória e armazena um número inteiro de 16 bits.
DB	Reserva memória e armazena uma string (conjunto de bytes).
RESD	Reserva memória para armazenar um ou mais números inteiros de 16 bits.
RESB	Reserva memória para armazenar um ou mais bytes.

Os modos de endereçamento suportados são dependentes de instrução. O modo de endereçamento imediato é apenas suportado pela instrução MOV, a qual permite copiar números inteiros, no intervalo de 0 a 4000, para o acumulador. A instrução MOV também é a única que suporta o endereçamento via registrador – copia operando do registrador referenciado para o acumulador –, sendo que o intervalo 4090 a 4093 é utilizado para referenciar os registradores CS, DS, SS e SP, respectivamente. O modo de endereçamento indireto é apenas suportado pelas instruções LDI e STI. A instrução RET é a única que não necessita de operando e, portanto, não suporta modo de endereçamento. As demais instruções suportam apenas o modo de endereçamento direto. Apesar da instrução SOP considerar endereçamento direto, ela também utiliza o endereçamento implícito no acesso à pilha do programa.

Para a comunicação com periféricos, diversos trabalhos, como os apresentados em [4-5][23-24], fazem uso de abstrações nos processadores para simular as operações de entrada/saída de sistemas operacionais. Os objetivos incluem a possibilidade de ensino das técnicas de entrada/saída em estágios preliminares do curso e economia de memória RAM do simulador (normalmente, os modelos de memórias propostos possuem capacidades de armazenamento bastante limitadas) [16], bem como permitir um *feedback* mais rápido nos programas desenvolvidos pelos estudantes que interagem como o computador (entrada de dados e visualização de resultados).

No CompSim, também considerou-se o uso de abstrações para composição das operações de entrada/saída. Porém, ao contrário dos simuladores citados, incluiu-se apenas duas operações básicas: 1) leitura de um byte do *buffer* de entrada para o acumulador; e 2) escrita de um byte do acumulador para um *buffer* de saída. Com o suporte dessas duas operações, outras mais complexas podem ser implementadas pelos estudantes, como, por exemplo, leitura/escrita de strings ou de números inteiros sinalizados, tornando-se um exercício prático de programação com aprofundamento teórico em entrada e saída programada (*polling*) e por interrupções.

A seguir é possível ver, como exemplo, o programa "*Hello, world!*" codificado na linguagem de baixo nível do processador Cariri:

```
1   .code
2       get_next_char:
3           LDI ptr     ; get next char from string
4
5       is_last_char:
6           SUB last_char  ; compare with '0'
7           JZ  end
8
9       print_char:
10          LDI ptr
11          INT putchar   ; print char
12
13      point_next_char:
14          MOV 1
15          ADD ptr
16          STA ptr       ; ptr += 1
17
18          JMP get_next_char
19
20      end:
21          INT exit
22
23  .data
24          msg:     DB 'Hello, World!0'
25          last_char: DB '0'
26
27          ptr:     DD msg  ; string pointer
28
29          putchar: DD 21   ;putchar code
30          exit:    DD 25   ;exit code
31
32  .stack 10
```

Fig. 6. Exemplo de programa para o processador Cariri.

A Figura 7 apresenta um *snapshot* do CompSim durante a execução do programa exemplo da Figura 6. Na Figura 7, pode-se observar, em sua parte inferior, à esquerda, no componente gráfico *Output Buffer*, a saída do programa (impressão parcial da string "*Hello, World!*").

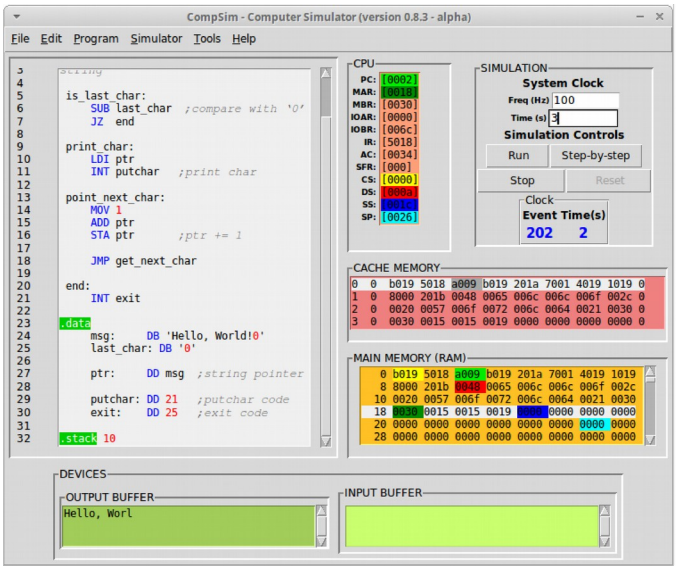

Fig. 7. Detalhe de componentes gráficos do CompSim.

D. Interface Gráfica

A interface do CompSim, que está integrada à plataforma Mandacaru e ao montador, apresenta os seguintes componentes gráficos:

- Editor de Código: utilizado para desenvolvimento de uma nova aplicação em baixo nível para execução na plataforma Mandacaru. O editor inclui exibição de número de linhas e de nome de arquivo, *sintax highlight* para a linguagem do processador Cariri, recursos de *Undo/Redo* e *Cut/Copy/Paste*, teclas de atalho, *Code Helper* (assistente para auxílio na construção das instruções com a sintaxe correta) e integração com o montador (destaca na cor vermelha uma linha do programa que, após uma análise, apresenta uma instrução com algum erro léxico, sintático e/ou semântico). A Figura 7 mostra alguns dos recursos do editor de código, como *sintax highlight* e numeração de linhas do código fonte do programa exemplo na Figura 6;
- CPU: exibe, em tempo de execução, os estados dos registradores do processador Cariri. Este componente distingue, em cores diferenciadas, cada um dos registradores de endereçamento à memória, sendo PC na cor verde-claro, MAR em verde-escuro, CS em amarelo, DS em vermelho, SS em azul-escuro e SP em azul-claro, como pode ser visto na Figura 7;
- *Cache Memory*: exibe os estados das linhas de memória cache, em tempo de execução, destacando a linha e a coluna de acordo com palavra buscada pelo processador. No cenário da Figura 7, está sendo destacada a terceira palavra da primeira linha;
- *Main Memory* (RAM): exibe os estados das posições da memória principal, em tempo de execução, destacando as posições referenciadas pelos registradores de endereçamento do processador Cariri, de acordo com as respectivas cores. Na Figura 7, pode-se observar as diferentes posições de memória referenciadas pelos registradores de endereçamento do processador Cariri, de acordo com as respectivas cores (por exemplo, o endereço 0x0018, destacado na cor verde-escura, está sendo endereçado pelo registrador MAR);
- *Devices*: inclui componentes gráficos para interações com o programa (entrada/saída de dados), durante uma simulação. Esses componentes estão integrados aos periféricos “Teclado” (*Input Buffer*) e “Vídeo” (*Output Buffer*) da plataforma Mandacaru;
- *Simulation*: inclui controles para configuração e gerenciamento das simulações;
- *Logs*: exibe informações de eventos e estatísticas de simulação gerados por cada componente da plataforma (permite gravação dessas informações em arquivo);
- *Variables*: exibe os valores, ou sequências de valores, assumidos pelas variáveis do programa durante uma simulação;
- *Stack*: exibe o conteúdo da pilha do programa durante uma simulação;
- *Platform Customization*: painel para customização dos parâmetros dos componentes da plataforma Mandacaru;
- *Assembly Report*: analisa o código do programa do usuário (editor de código) e exibe um relatório com: tabela de símbolos, endereços de memória dos segmentos de código, dados e pilha, tamanho da pilha de programa e o respectivo código de máquina;
- Outros aplicativos: consulta de código e/ou de símbolo na Tabela ASCII, conversão entre bases numéricas (Decimal, Binária e Hexadecimal).

IV. Resultados Preliminares

Utilizou-se o simulador CompSim nas práticas laboratoriais de uma turma com 20 alunos da disciplina de OAC de um curso de Bacharelado em Sistemas de Informação.

No experimento, os alunos concordaram em avaliar o simulador nas seguintes dimensões: 1) **Suporte educacional**, que abrangeu i) cobertura dos cinco tópicos básicos apontados nos currículos de referência em [10-11], ii) suporte ao desenvolvimento das habilidades práticas definidas em [10] e iii) suporte pedagógico; e 2) **Experiência de uso do simulador**.

Para essa avaliação, formulou-se uma Rubrica [21] com 9 indicadores de qualidade nos contextos de Suporte Educacional e 2 indicadores para a Experiência de Uso do Simulador. São eles:

1.1 Conteúdo curricular: relação entre as práticas realizadas com os componentes curriculares da disciplina.

1.2 Pensamento de alto nível: possibilidade de construção de novos conhecimentos, através da análise, avaliação e síntese.

1.3 Nível de Graduação: se o nível de conteúdos abordados no simulador é apropriado para o público-alvo.

1.4 Ambiente de laboratório: espaço nas aulas em laboratório para desenvolvimento das atividades práticas.

1.5 Componentes de avaliação: suporte aos estudantes para autoavaliação do nível de aprendizado.

1.6 Materiais de apoio: disponibilidade de materiais complementares, como apostilas, vídeos, exemplos de códigos-fonte, entre outros.

1.7 Efetividade no ensino: efetividade no suporte ao ensino-aprendizagem dos conteúdos abordados.

1.8 Desempenho do estudante: taxa de aprendizado é apropriada às necessidades individuais dos estudantes.

1.9 Interatividade: relação entre a atividade do usuário com o sistema e o suporte ao estímulo/motivação para estudos.

2.1 Amigabilidade: design limpo, fácil de navegar e dispõe de meios para auxílio ao estudante no aprendizado.

2.2 Acessibilidade: informações gráficas são devidamente rotuladas, fontes são consistentes e fáceis de ler, e se estão sendo empregados diferentes estilos de aprendizagem e níveis de habilidade.

Cada um dos indicadores incluiu os seguintes aspectos de qualidade, com respectivos escores: Ruim (1 ponto), Razoável (2 pontos), Bom (3 pontos) e Excelente (4 pontos).

A Tabela III apresenta os resultados da aplicação da rubrica com a turma de OAC. Nela, podemos observar que todos os alunos avaliaram que os conteúdos curriculares da disciplina foram cobertos nas práticas com simulador (1.1) e que ele oferece meios para construção de novos conhecimentos (1.2).

TABELA III. RESULTADOS DA AVALIAÇÃO PELOS ALUNOS.

Indicador	Média Escores	Qualidade Média
1.1 Conteúdo curricular	4	Excelente
1.2 Pensamento de alto nível	4	Excelente
1.3 Nível de graduação	3,9	Excelente
1.4 Ambiente de laboratório	3,9	Excelente
1.5 Componentes de avaliação	3,7	Excelente
1.6 Materiais de apoio	3,7	Excelente
1.7 Efetividade no ensino	3,7	Excelente
1.8 Desempenho do estudante	3,6	Excelente
1.9 Interatividade	3,5	Bom
2.1 Amigabilidade	3,4	Bom
2.2 Acessibilidade	3,6	Excelente

Ainda na Tabela III, a maioria dos alunos considerou o conceito “excelente” para: conteúdos abordados foram adequados ao nível de graduação (1.3), que houve espaço em laboratório para desenvolvimento das atividades práticas (1.4), as atividades desenvolvidas no simulador permitiram que eles pudessem autoavaliar o aprendizado (1.5), os materiais de apoio utilizados foram suficientes (1.6), uso do simulador foi efetivo para os processos de ensino-aprendizagem (1.7) e foi possível assimilar os conteúdos na medida em que foram apresentados (1.8); e “bom” no que se refere à relação entre interatividade entre usuário/simulador e o estímulo para estudos (1.9). Já para os indicadores de experiência de uso do simulador, a maioria dos alunos consideraram que sua interface é amigável (2.1) e o suporte à acessibilidade é excelente (2.2).

Ressalta-se ainda que, em relação às turmas anteriores da mesma disciplina, houve um acréscimo de motivação e produtividade e, consequentemente, no desempenho geral final. As atividades em laboratório, com o uso do simulador CompSim, trataram desde demonstrações básicas de aspectos de projetos de sistemas digitais e de organização de computadores, até o desenvolvimento de aplicações em nível de máquina para:

- Operações matemáticas: multiplicação e divisão utilizado somas ou subtrações sucessivas, respectivamente, ou com deslocamento de bits;
- Operações lógicas: AND, OR e NOT;
- Construções de linguagens de programação de alto nível: IF/ELSE e FOR/WHILE; e
- Funções diversas, com passagem de parâmetros, retorno de resultado e recursividade: máximo divisor comum, contagem de total de números primos dentro de um determinado intervalo, sequência de Fibonacci e busca linear.

Ao final da disciplina, houve ainda espaço para um desafio de projeto de sistema computacional, com a proposta de que os estudantes deveriam codificar uma aplicação com função recursiva para cálculo de potência e, através de customizações no código-fonte da aplicação e nas configurações da plataforma Mandacaru, otimizar o desempenho do sistema.

Por fim, os *feedbacks* dos estudantes, durante toda a disciplina, foram muito importantes para identificação/correção de bugs, adição de novos recursos, produção de materiais didáticos complementares e otimizações no desempenho do simulador.

V. CONCLUSÕES

Este artigo apresentou o CompSim, uma ferramenta para suporte ao ensino aprendizado em Organização e Arquitetura de Computadores. O simulador proposto, que segue a abordagem de Projetos Baseados em Plataforma, inclui uma interface gráfica integrada a uma plataforma de hardware simulável, que permite simplificar a configuração dos componentes de hardware, programação do processador, execução e avaliação de desempenho de novos sistemas computacionais. O CompSim foi utilizado nas práticas laboratoriais e avaliado por uma turma de uma disciplina de Organização e Arquitetura de Computadores. Os resultados mostraram que a ferramenta proposta apresentou suporte educacional efetivo e qualidade na experiência de uso, bem como constatou-se um acréscimo na motivação para estudos avançados e no desempenho final da turma.

Os trabalhos futuros incluem: a adição de novos componentes simuláveis de hardware e implementação da plataforma Mandacaru em FPGA, estabelecendo assim cenários reais e fluxo completo de projetos de sistemas computacionais baseados em plataformas; o suporte de diferentes tipos de dados, um compilador de linguagem de alto nível e um sistema operacional de tempo real para o processador Cariri; novos recursos de interface gráfica, como diagramas dinâmicos de organização de processador e de memórias, bem como de temporização de barramento; cossimulação com ferramentas de projeto de sistemas computacionais e plataformas de prototipação de projetos eletrônicos; novos materiais didáticos para OAC; e, por fim, disponibilizar todos os códigos-fonte assim que o mesmo obtenha maturidade e o CompSim esteja estável, criando assim mais oportunidades para discussões/implementações de novas ideias e novas colaborações técnico-científicas.

REFERÊNCIAS

[1] W. Stallings, “Arquitetura e Organização de Computadores”. 8a Ed. Prentice Hall, 2010.

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

[3] A. E. Lourenço, E. T. Midorikawa, “Ensino de Arquitetura de Computadores Utilizando Simuladores Completos”. Congresso Brasileiro de Engenharia - COBENGE 2004, Brasila, 2004.

[4] M. R. D. Ullmann, A. C. G. Inocência, E. D. M. Neto, M. S. Freitas, P. A. F. Júnior, “NeanderSIM: Simulador Gráfico de Apoio ao Ensino de Arquitetura de Computadores”. XXXIV Congresso da Sociedade Brasileira de Computação (CSBC), 2014.

[5] G. P. Silva, J. A. dos S. Borges, “SimuS – Um Simulador Para o Ensino de Arquitetura de Computadores”. International Journal of Computer Architecture Education (IJCAE), v. 5, no. 1, 2016. pp 7-12.

[6] V. B. da Silva,, J. F. P. Cheiran, “Análise do uso de microcontroladores como ferramenta de apoio ao ensino-aprendizagem de Arquitetura de Computadores”. International Journal of Computer Architecture Education (IJCAE), v. 4, no. 1, 2015. pp 1-4.

[7] TEMURTAS, H. O.; GULBAG, A. BZK.SAU: Implementing a Hardware and Software-based Computer Architecture Simulator for Educational Purpose. International Conference On Computer Design And Appliations (ICCDA 2010), 2010.

[8] P. H. M. M. Penna, H. C. Freitas, “Análise e Avaliação de Simuladores de Sistemas Completos para o Ensino de Arquitetura de Computadores”. International Journal of Computer Architecture Education (IJCAE), v. 2, no. 1, 2013. pp 13-16.

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

[10] Association for Computing Machinery, IEEE Computer Society, “Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, December, 2013”, https://www.acm.org/education/CS2013-final-report.pdf. Acesso em: 03 abr. 2017.

[11] Sociedade Brasileira de Computação, “Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação, 2005”, http://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/760-curriculo-de-referencia-cc-ec-versao2005. Acesso em: 03 abr. 2017.

[12] M. D. R. Uribe, A. J. Magana, J.-H. Bahk, A. Shakouri, “Computational Simulations as Virtual Laboratories for Online Engineering Education: A Case Study in the Field of Thermoelectricity”. Computer Applications in Engineering Education, Vol. 24(3), 2016. pp. 428–442.

[13] I. A. Garcia, C. L. Pacheco, J. N. Garcia, “Enhancing education in electronic sciences using virtual laboratories developed with effective practices”. Computer Applications in Engineering Education, Vol. 22(2), 2014. pp. 283–296.

[14] B. Balamuralithara, P. C. Woods, “Virtual laboratories in engineering education: The simulation lab and remote lab”. Computer Applications in Engineering Education, Vol. 17(1), 2009. pp. 108–118.

[15] H. S. Tan, K. C. Tan, L. Fang, M. L. Wee, C. Koh, “Using Simulations to Enhance Learning and Motivation in Machining Technology”. Proceedings of The 17th International Conference on Computers in Education (ICCE), 2009.

[16] A. J. Magana, S. P. Brophy, G. M. Bodner, “An exploratory study of engineering and science students’ perceptions of nanohub.org simulation tools”, International Journal of Engineering Education, Vol. 28(5), 2012. Pp.1019–1032.

[17] D. Srivastava, S. M. Atluri, “Computational nanotechnology: A current perspective”. Computer Modeling in Engineering and Sciences. Vol. 3(5), 2002. pp. 531–538.

[18] Bahk, J. H.; Youngs, M.; Yazawa, K.; Shakouri, A.; Pantchenko, O. An online simulator for thermoelectric cooling and power generation. Frontiers in education Eonference, IEEE, 2013.

[19] A. Akram, L. Sawalha, “x86 computer architecture simulators: A comparative study”. In IEEE 34th International Conference on Computer Design (ICCD), IEEE, 2016.

[20] G. S. Wolffe, W. Yurcik, H. Osborne, M. A. Holliday, “Teaching computer organization/architecture with limited resources using simulators”. Proceedings of the 33rd SIGCSE technical symposium on Computer science education, ACM SIGCSE Bulletin. Vol. 34, No. 1, 2002.

[21] M. Yuan, M. Recker, “Not all rubrics are equal: A review of rubrics for evaluating the quality of open educational resources. The International Review of Research in Open and Distributed Learning, 16(5), 2015.

[22] A. Davare, D. Densmore, T. Meyerowitz, A. Pinto, A. Sangiovanni-Vincentelli, G. Yang, H. Zeng, Q. Zhu, “A next-generation design framework for platform-based design”. Conference on using hardware design and verification languages (DVCon) (Vol. 152), 2007.

[23] D. Skrien, "CPU Sim 3.1: A tool for simulating computer architectures for computer organization classes." Journal on Educational Resources in Computing (JERIC) 1, no. 4 (2001): 46-59.

[24] L. Null, J. Lobur, "MarieSim: The MARIE computer simulator."Journal on ducational Resources in Computing (JERIC) 3, no. 2 (2003): 1.