Revista Varia Scientia
v.09 , n.16, p. 139-148

Ano de Impressão 2010

André Barbosa Verona[1]
João Angelo Martini[2]
Tiago Lopes Gonçalves[3]

SIMAEAC: UM SIMULADOR ACADÊMICO PARA ENSINO DE ARQUITETURA DE COMPUTADORES

RESUMO: Técnicas de simulação para apoio ao ensino e à pesquisa têm sido cada vez mais empregadas. Através de simuladores podemos prever o comportamento de sistemas antes mesmo de serem criados, viabilizando a previsão de falhas e possibilitando melhorias nos mesmos. Este trabalho apresenta um simulador gráfico baseado no processador 8085 que visa facilitar o processo de ensino-aprendizagem de arquitetura de computadores, através da visualização dos passos e da dinâmica das unidades funcionais no decorrer da execução de uma instrução.
PALAVRAS-CHAVE: Arquitetura de computadores, processador 8085, simulador.

SIMAEAC: AN ACADEMIC SIMULATOR TO TEACH COMPUTER ARCHITECTURE

SUMMARY: Simulation techniques to support teaching and research have been increasingly used. Through simulation we can predict the behavior of systems before they are created, enabling the prediction of failure and possible improvements on them. This paper presents a simulator based on the graphic processor 8085 that has the objective of easing the teaching-learning computer architecture, by viewing the steps and dynamics of the functional units during an instruction.
KEYWORDS: Computer architecture; processor 8085; simulator.

Data de recebimento: 19/07/2009. Data de aceite para publicação: 29/09/2009.
[1] Mestrando em Ciência da Computação, Prof. Colaborador, DIN, Campus Sede, UEM, Maringá, PR, (0xx44) 3261-4040, CEP 87020-900, e-mail: andrebv@din.uem.br.
[2] Doutor em Ciências, Prof. Adjunto, DIN, Campus Sede, UEM, Maringá, PR, CEP 87020-900, e-mail: jangelo@din.uem.br .
[3] Mestre em Ciência da Computação, Professor, FAED, Dois Vizinhos, PR, (0xx46) 3581-5000 – CEP 85660-000, e-mail: tiagolopes@unisep.edu.br.

INTRODUÇÃO

As técnicas de simulação têm facilitado cada vez mais o processo de ensino-aprendizagem do funcionamento de sistemas reais, tais como os microprocessadores. Simuladores são usados para reproduzir ou prever o comportamento de desses sistemas, possibilitando o entendimento e previsão de possíveis falhas. Por meio da reprodução desse comportamento, suas características fundamentais podem ser observadas, facilitando principalmente o ensino e a pesquisa nas mais variadas áreas de conhecimento.

No contexto de arquitetura de computadores, entre outras funções, os simuladores viabilizam o projeto de uma arquitetura de maneira a se obter o melhor desempenho antes mesmo da fabricação do chip. Os simuladores ainda auxiliam no ensino por meio da reprodução fiel da execução de determinado programa, permitindo a exibição dos passos de execução e do comportamento de cada unidade funcional.

Há uma diversidade de simuladores que implementam as mais variadas arquiteturas de microprocessadores, desde as mais simples como o 8085 (GESER, 1989), que utiliza o modelo arquitetural de Von Neumann (NEUMANN, 1993), implementada no Abacus (ZILLER, 2000) e no GNUSim8085 (SRIDHAR, 2002), até as mais complexas, simulando arquiteturas pipelined, como a do processador MIPS64 com o WinMIPS64 (SCOTT, 2006) e arquiteturas superescalares com o SimpleScalar (AUSTIN & BURGER, 1997) e o SATSim (WOLFF & WILLS, 2000).

Este trabalho apresenta um simulador gráfico baseado na arquitetura do processador 8085, semelhante aos simuladores Abacus e GNUSim, porém com uma interface mais amigável e com algumas simplificações que visam melhorar a compreensão da execução de um programa nesse microprocessador.

Esse artigo discute, primeiramente, os fundamentos do modelo da arquitetura de Von Neumann para contextualizar a proposta do simulador desenvolvido neste trabalho. Na seqüência relata os trabalhos relacionados, mostrando algumas características positivas e negativas de cada simulador analisado, seguida da apresentação do simulador proposto neste trabalho e suas principais características. E por fim mostra a conclusão desse estudo além de trabalhos futuros.

FUNDAMENTOS DO MODELO DE VON NEUMANN

O simulador desenvolvido neste trabalho implementa o microprocessador 8085 que é baseado no modelo arquitetural de Von Neumann e que considera uma unidade central de processamento

(CPU) com apenas uma unidade funcional, podendo buscar apenas uma instrução por ciclo de execução. Para o ensino de arquitetura de computadores é interessante omitir, em um primeiro momento, os conceitos de pipeline e superescalar, para que o modelo de Von Neumann possa ser compreendido de forma sucinta.

Os modelos de arquitetura pipelined e superescalar são evoluções do modelo proposto por Von Neumann. Na arquitetura proposta por Von Neumann, uma CPU só teria uma unidade funcional que ficaria responsável pela execução de todos os tipos de instruções suportadas pela mesma. Para tanto, uma instrução deveria passar por várias subunidades funcionais da CPU, tais como subunidade de busca de instrução, subunidade de decodificação de instrução, subunidade de busca de operandos (quando houver) e, por fim, a subunidade de execução da instrução.

Vale ressaltar que nem sempre a instrução faz uso da subunidade de busca de operando, pois existem instruções que não necessitam de operando para serem executadas. Nesse caso, a instrução passa direto da subunidade funcional de decodificação para a subunidade funcional de execução.

Uma limitação desse modelo de arquitetura é que uma segunda instrução só pode ser buscada na memória quando a primeira for concluída, ou seja, quando tiver percorrido todas as subunidades funcionais. Para melhorar o desempenho das CPUs, foram criadas subunidades funcionais independentes, possibilitando que uma segunda instrução não necessite esperar a primeira passar por todas as subunidades funcionais para começar a ser processada. Assim que a primeira instrução sair da primeira subunidade funcional, a segunda instrução pode ser buscada, pois a primeira subunidade (subunidade de busca) estará livre. Tal técnica é denominada de pipeline, e é empregada em praticamente todos os microprocessadores atuais. A Figura 1 apresenta uma comparação entre o modelo de Von Neumann e o modelo usando a técnica de pipeline.

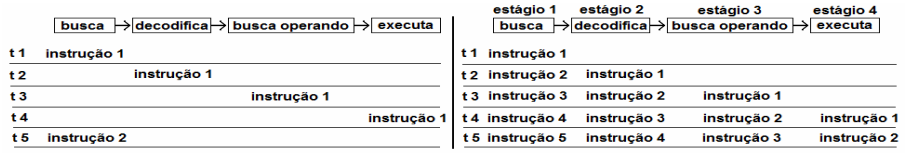

Figura 1 À esquerda o modelo de Von Neumann, e à direita a estratégia pipeline.

Note que com a estratégia pipeline, no instante de tempo t5, a instrução 2 já está no estágio de execução (subunidade de execução),

enquanto no modelo de Von Neumann ela ainda está na subunidade de busca.

Outra inovação nas arquiteturas das CPUs é a técnica superescalar. Essa técnica surgiu no momento em que a inserção de mais de uma unidade funcional se tornou possível. Assim, com a disponibilidade de mais de uma unidade funcional, mais de uma instrução pode ser buscada por vez, possibilitando paralelismo real na execução de instruções. A técnica pipeline também pode ser utilizada em conjunto com a superescalar, melhorando ainda mais o desempenho de uma CPU. Na Figura 2, é possível observar o resultado do emprego dessas técnicas, em conjunto e em comparação com o modelo que emprega exclusivamente o pipeline.

Figura 2 À esquerda o modelo pipeline, e à direita o modelo superescalar com pipeline.

Para fins didáticos é interessante omitir essas duas inovações das CPUs e considerar apenas o modelo de Von Neumann, pois o primeiro desafio de ensino em arquitetura de computadores é explicar o ciclo de execução de uma instrução pela CPU.

Nesse contexto, visando facilitar o primeiro contato dos alunos com o funcionamento de uma CPU, e o ensino da arquitetura de computadores, o presente trabalho apresenta o simulador denominado SIMAEAC, desenvolvido com base no processador Intel 8085, com o intuito de possibilitar aos alunos de cursos de graduação ou pós-graduação maior entendimento sobre toda a complexidade envolvida na execução de instruções por uma arquitetura de computador.

TRABALHOS RELACIONADOS

Análogos ao simulador proposto neste trabalho, encontramos o Abacus (ZILLER, 2000) e o GNUSim8085 (SRIDHAR, 2002) que implementam a arquitetura do microprocessador 8085 da Intel. Esse processador é bastante simples e por isso tem sido usado como processador base para auxiliar no ensino dos princípios de arquitetura de computadores. Ele possui apenas uma unidade funcional, seguindo

o modelo proposto por Von Neumann e não apresenta pipeline. Possui um barramento de endereços de 16 bits, que permite endereçar até 65.536 células de memória, além de um barramento de dados de 8 bits e seus modos de endereçamento são imediato e direto, dependendo da instrução manipulada. O 8085 manipula instruções de 1, 2 ou 3 bytes. Seus registradores de dados são de 8 bits, além dos FLAGS(possui 8, mas utiliza apenas 5 bits) e os registradores SP (Stack Pointer) e PC (Program Counter) podem armazenar 16 bits, pois tratam endereços de memória.

O Abacus foi desenvolvido para plataforma Windows, enquanto que o GNUSim8085 para plataformas Unix e suas distribuições Linux. Ambos exibem todos os registradores encontrados no microprocessador 8085, que são: A, B, C, D, E, H, L, PC, SP e os FLAGS, além de um mapa das instruções que estão sendo executadas e das que serão executadas. Eles mostram também a memória do processador com exibição do conteúdo de cada célula, sendo que a exibição dos dados manipulados pelo microprocessador é no formato hexadecimal e o GNUSim8085 traz opção de conversão para decimal. Uma dificuldade encontrada no Abacus é a inserção de código (instruções) através dos botões disponibilizados. Com isso, para a construção de um programa devemos clicar nos botões referentes às instruções (mnemônicos) uma a uma, o que consome um tempo relativamente alto.

O simulador WinMIPS64 (SCOTT, 2006) simula a arquitetura pipelined do microprocessador MIPS64. Esse simulador apresenta um conjunto de 32 registradores para números inteiros e 32 para números representados em ponto flutuante. Ele possibilita a visualização da instrução passando pelos pipelines e ainda estatísticas relativas à execução, como, por exemplo, quantidade de ciclos que determinada instrução consumiu para passar por todos os estágios do pipeline. A desvantagem do WinMIPS64 é a mesma do Abacus, ou seja, não existe uma área para edição do código de execução.

Para o ensino de arquiteturas superescalares, há o simulador SATSim (WOLFF & WILLS, 2000) que não tem como base nenhuma arquitetura de processador real, o que o torna mais flexível para configurar, por exemplo, o seu número de unidades funcionais.

SIMAEAC

O SIMAEAC - Simulador Acadêmico para Ensino de Arquitetura de Computadores - implementa a arquitetura do processador 8085, através de instruções lógicas e aritméticas, de manipulação de pilha, de chamada e retorno de subrotinas, de desvios condicionais e incondicionais, de transferência de dados entre registradores e entre

registradores, e memória. Exibe os registradores A (Acumulador), B, C, D, E, H, L, PC e SP, todos de 8 bits, e FLAGS, de 5 bits. Ainda, ele possui um campo chamado Programa que detalha instruções executadas e a serem executadas, como endereço na memória, mnemônico, opcode correspondente, além dos dados a serem manipulados; ele também mostra o mapa de memória, que, por motivo de simplificação, teve o número de células reduzido.

No processador original os endereços variam de 0000 a FFFF, enquanto no SIMAEAC eles vão de 00 até 8F. Essa quantidade de memória implementada foi a que permitiu a visualização sem necessidade de rolagem da barra lateral. Dessa forma, a visualização da célula em que se encontra a instrução sendo executada está sempre disponível.

Com essa alteração, instruções que manipulam memória além dos registradores SP e PC devem manipular 8 bits. O registrador de FLAGS possui 8 bits, porém, no 8085 apenas 5 foram utilizados, ficando os 3 restantes a serem utilizados em uma possível atualização dessa arquitetura. Na Figura 3, temos a interface do SIMAEAC.

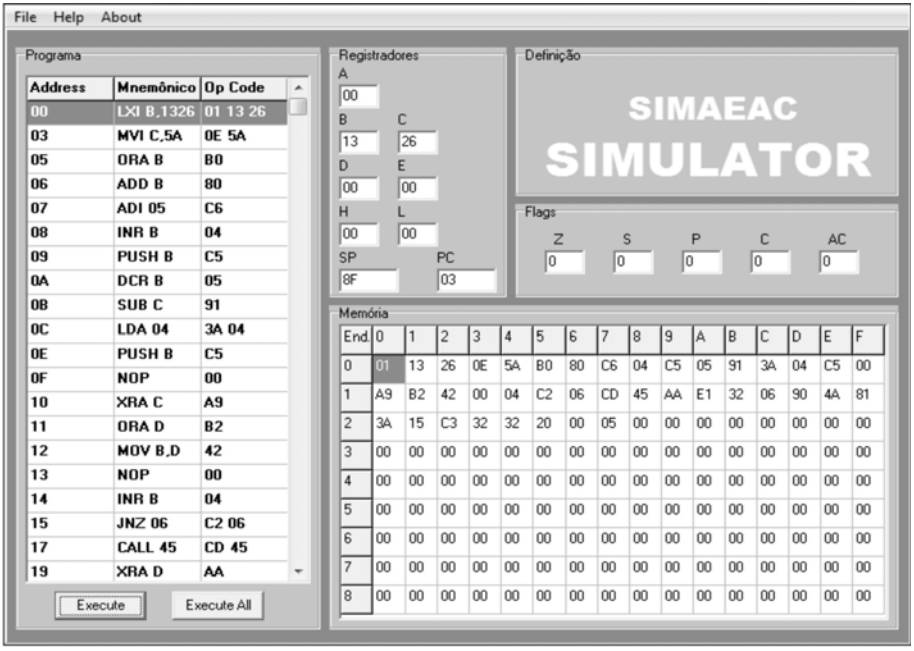

Figura 3 Interface do SIMAEAC.

É válido notar que os valores são exibidos em hexadecimal para facilitar o entendimento, pois em binário a compreensão seria

prejudicada. Com isso, cada registrador ou célula de memória exibe apenas dois caracteres (4 bits podem representar 1 caractere em hexadecimal). Outra característica interessante do SIMAEAC é uma tarja azul que acompanha tanto o campo Programa quanto o campo Memória. No campo Programa essa tarja marca toda a linha de uma instrução sendo executada (endereço, mnemônico, opcode), e na Memória, ela marca a célula da instrução que será executada naquele momento. Essa é uma característica interessante e que facilita bastante o acompanhamento da execução das instruções, melhorando a compreensão e o aprendizado da dinâmica de execução de instruções. Para a execução das instruções existem dois botões, o Execute (executa apenas a instrução sublinhada), e o Execute All (executa todas as instruções do programa).

Mais uma simplificação visando melhorar a usabilidade do SIMAEAC foi a adoção do bloco de notas (.txt) para criação do programa de execução. Com isso, podemos desenvolver os programas com os mnemônicos no bloco de notas, depois abrirmos o SIMAEAC para carregá-los e em seguida executá-los. Para que isso fosse possível, foi desenvolvido um analisador léxico, responsável pela leitura e interpretação desse arquivo do bloco de notas. No momento em que o programa contido no bloco de notas é carregado para o simulador, o campo Programa é atualizado e cada campo recebe o valor correspondente. A memória também é atualizada e os dados e os valores dos opcodes (valores que foram obtidos através da conversão dos mnemônicos escritos no programa do bloco de notas) são inseridos nos locais correspondentes.

O SIMAEAC disponibiliza suporte para uma grande quantidade de instruções do processador 8085. Para instruções de transferência de dados, temos as instruções MOV r1,r2 (move o dado contido no registrador r2 para o registrador r1). Temos ainda a instrução MVI r1,dado8 (move um dado de 8 bits para o registrador r1) e a LXI r1,dado16 (move um dado de 16 bits para os registradores r1 + seu par). Uma peculiaridade da instrução LXI é que r1 só pode ser B (B+C), D (D+E) ou H (H+L), pois sempre precisaremos de 2 registradores de 8 bits para manipular um dado de 16 bits.

Para dar suporte às instruções lógicas, foram implementadas as instruções ORA r1 (OU lógico bit a bit entre o conteúdo do registrador r1 e o conteúdo do registrador A), ANA r1 (E lógico bit a bit entre o conteúdo do registrador r1 e o conteúdo do registrador A) e XRA r1 (OU EXCLUSIVO lógico bit a bit entre o conteúdo do registrador r1 e o conteúdo do acumulador).

Para instruções aritméticas, temos as instruções ADD r1 (soma o conteúdo do registrador r1 com o conteúdo do registrador A), ADI dado8

(soma o dado de 8 bits com o dado contido no registrador A), INR r1 (incrementa em uma unidade o conteúdo do registrador r1), DCR r1 (decrementa em uma unidade o conteúdo do registrador r1) e SUB r1 (subtrai do conteúdo do registrador A o conteúdo do registrador r1).

Para manipulação da pilha, temos implementadas as instruções PUSH r1 (empilha um dado de 16 bits representado pelo par de registradores r1 + seu par) e POP r1 (desempilha um dado de 16 bits para o par de registradores r1 + seu par). É importante lembrar que para essas duas instruções o valor de r1 só pode ser B (B+C), D (D+E) ou H (H+L). A pilha no SIMAEAC está implementada no final da memória, da última célula para as penúltimas (8F, 8E, 8D, 8C), e o registrador que manipula a pilha é o SP (stack pointer), que é alterado a cada instrução que manipula essa pilha.

Instruções de chamada e retorno de subrotinas foram implementadas com as instruções CALL endereço (desvia a execução normal do programa para a instrução do endereço denominado endereço) e RET (retorna para a execução da instrução imediatamente após a instrução CALL). A cada chamada de subrotina, o valor do registrador PC é armazenado na pilha, para que o processador possa saber onde deverá voltar a executar (endereço de retorno) quando uma subrotina terminar sua execução.

Foram implementados dois tipos de instruções de desvio, o condicional e o incondicional. O desvio incondicional é dado pela instrução JMP endereço, que quando executada desvia o fluxo normal de execução do programa para o endereço acompanhado da instrução. A instrução condicional foi implementada pelo mnemônico JNZ endereço, que no momento da execução desvia para o endereço que o acompanha se a FLAG Z for diferente de zero (JNZ = jumper if not zero flag).

Além das instruções de transferência de dados entre registradores, foram implementadas as instruções de transferência entre registradores e a memória. Essas instruções são sempre realizadas com o registrador A. Para cópia dos dados contidos em uma célula de memória para esse registrador, temos a instrução LDA endereço. Já para copiarmos o conteúdo do registrador A para uma célula de memória, temos a instrução STA endereço. A cada execução dessas instruções, um dado de 8 bits é manipulado. Ainda, essas duas instruções apresentam modo de endereçamento direto, pois o dado a ser manipulado na execução não é o endereço que acompanha a instrução, e sim o conteúdo desse endereço.

Como pode ser observado na interface do SIMAEAC, temos as letras Z, S, P, C, AC, que representam as FLAGS. A letra Z é a FLAG de zero, que assume valor 1 quando o resultado de uma operação lógica ou aritmética é igual a zero. A letra S é a FLAG de sinal cujo valor é 1

quando o dado resultante de uma operação lógica ou aritmética é um número negativo. A letra P é a FLAG de paridade e assume valor 1 quando o dado resultante de uma operação lógica ou aritmética contém um número par de bits 1. A letra C corresponde a FLAG de carry do bit 7 e as letras AC ao carry do bit 3. Elas recebem valor 1 quando em uma operação aritmética ocorre transbordo do bit 7 para fora do número ou quando ocorre transbordo do bit 3 para o bit 4 do número manipulado.

O SIMAEAC está sendo utilizado na Universidade Estadual de Maringá como ferramenta de auxílio ao aprendizado da arquitetura de computadores nas disciplinas de Arquitetura de Computadores I do curso de Ciência da Computação e Arquitetura e Organização de Computadores II do curso de Informática, ambas do Departamento de Informática.

Com o auxílio desse simulador, os alunos vêm conseguindo diminuir o tempo gasto para compreensão da dinâmica de execução de uma instrução, além de entenderem com mais facilidade como estão organizadas as unidades funcionais, a relação exata entre um mnemônico e um opcode, e visualizar o que acarreta a execução das mais variadas instruções. Essa ferramenta está disponível para download em www.din.uem.br/~andrebv.

CONCLUSÕES E TRABALHOS FUTUROS

Entender como determinada arquitetura de computador funciona é uma tarefa difícil se nos valermos apenas de teoria. Para melhorar esse entendimento, são usados os simuladores que reproduzem com perfeição toda a dinâmica de execução das instruções em uma arquitetura de computador. Visando o melhoramento desse processo de ensino-aprendizem, foi desenvolvido o SIMAEAC, um simulador simples baseado no modelo de Von Neumann. Esse simulador apresenta diversas características que facilitam o ensino, tais como: a tarja azul que acompanha as células de memória que estão sendo lidas naquele ciclo, o espaço de memória reduzido, garantindo que sempre será possível visualizar a célula lida em cada ciclo, o uso do bloco de notas para escrita dos programas a serem executados, facilitando a criação e alteração do código fonte a ser lido, o fato de não ser necessária a instalação do simulador na máquina que apenas executando o código objeto já está pronto para uso, e a gratuidade para utilização. Com esses fatores positivos, espera-se que esse simulador possa auxiliar alunos de graduação e pós-graduação no estudo de fundamentos de arquitetura de computadores, atuando como ferramenta auxiliar nas aulas teóricas.

REFERÊNCIAS

AUSTIN, T.M.; BURGER, D. **The SimpleScalar Tool Set, Version 2.0**. University of Wisconsin-Madison Computer Sciences Department, Technical Report, n. 1342, 1997.

GESER, S. A Specification of the Intel 8085: A Case Study. In: **Lectures Notes in Computer Science**. New York, EUA: Springer, 1989. p. 347-401.

SCOTT, M. (2006) **WinMips64, version 1.5**. School of Computing Dublin City University, Ireland. Disponível em:

< http://www.computing.dcu.ie/~mike/winmips64.html>. Acesso em: jul. 09.

SRIDHAR, Z. **GNUSim8085, versão 1.3**, 2002. Disponível em: <http://gnusim8085.sourceforge.net/index.php/Main_Page>. Acesso em: jun. 09.

VON NEUMANN, J., First Draft of a Report on the EDVAC. In: **Annals of the History of Computing**, v. 15, 4 ed., 1993. p. 27-75,

WOLFF, M.; WILLS, L. SATSim: A Superscalar Architecture Trace Simulator Using Interactive Animation. In: WCAE: WORKSHOP ON COMPUTER ARCHITECTURE EDUCATION, 2000. Vancouver, Canada.

ZILLER, R. **Microprocessadores:** Conceitos Importantes, 2 ed. Florianópolis, Brasil: EEL – UFSC, 2000.

VARIA SCIENTIA

Versão eletrônica disponível na internet:

www.unioeste.br/saber