Simulador de Processador de Computador com propósito didático

Galileu Santos de Jesus, Marco Tulio Chella
Departamento de Computação - Universidade Federal de Sergipe
São Cristovão, Brasil
galilasmb@gmail.com, chella@ufs.br

Abstract. *Project carried out in order to simulate a computer processor didactically. Using the functions and their interconnections as well as the implementation of a software in the Java programming language. Simulating through creation of memory, PC, registers, ALU, AC, IR, decoder and other components. Running through instruction, called word, similar to some of the programming language Assembler, with reference to the simulator Neander.*

Resumo. *Projeto realizado com o objetivo de simular um processador de computador de forma didática. Utilizando as funções e suas interligações, bem como a implementação de um software na linguagem de programação Java. Simulando através de criação de memória, PC, registradores, ULA, AC, IR, Decodificador e outros componentes. Sendo executadas através de instruções, denominada de palavra, semelhantes a algumas da linguagem de programação Assembler, tendo como referência o simulador Neander.*

1. Introdução

Um processador é o cérebro de todo computador, é o responsável pela execução de todas as instruções da máquina, é a parte "mais importante", que faz o controle de todas as outras partes e a execução de instruções, como operações matemáticas, comunicação com dispositivos de saída ou entrada, armazenamento de dados e outros.

> *"a simulação é um processo de projetar um modelo computacional de um sistema real e conduzir experimentos com este modelo com o propósito de entender seu comportamento e/ou avaliar estratégias para sua operação"*
> ***Pegden, C. D.; Shannon, R.E. e Sadowski, R. P. (1990) "Introduction to Simulation Using SIMAN", McGraw Hill, NY, 2 ed.***

Este trabalho apresenta a implementação de um simulador de processador didático, onde o usuário digita o programa em Assembler, executa e visualiza todos os passos que compreendem a execução do processador, auxiliando muito no aprendizado dos alunos.

Funciona basicamente no envio de mensagens, denominadas de palavra, que armazena instruções e dados, enviados e conectados a cada dispositivo por meio de barramentos, armazenando na memória e/ou executando as tarefas passadas. Esta palavra muitas vezes pode variar de acordo com o processador.

2. Trabalhos Relacionados

Existem alguns trabalhos que simulam o funcionamento de processador, porém, não são muitos didáticos. Alguns como Processador LC-3 [1], NEANDER [2] [3], sendo que o último é usado como base neste projeto, entre outros.

3. Desenvolvimento

3.1. Componentes correspondentes ao código em Java, representados na Figura 3

Memória: JTable Memoria;
Decodificador: método chamado: decodificarTabela(){}
Codificador: método chamado: codificar(int c){}
Unidade Lógica Aritmética: método:
BotaoExecutarActionPerformed(ActionEvent evt){}
EAX: variável: long EAX;
EBX: variável: long EBX;
PC: variável: long PC;
AC: variável: long AC;

Barramentos: É a chamada dos métodos para execução dos passos, correspondentes a cada ação gerada. No caso do barramento de dados é utilizado uma variável do tipo "int barramentoDeDados;" Fazendo assim a movimentação da palavra, separando os campos de acordo com o especificado.

3.2. Sintaxe

Foram implementadas 16 funções, a Tabela 1 mostra algumas delas, e como utilizá-las, as restantes estão presentes na janela "Ajuda", da Figura 5.

Função	Sintaxe	Descrição
MOV	MOV 5 10	Move o valor do campo Origem (5) para a posição de memória indicada pelo campo Destino (10), onde quando destino é 63 refere-se a AC, 62 a EAX e 61 a EBX. Também o bit reservado para números negativos dos seis bits do meio.
ADD	ADD 10 11	Soma os valores das posições de memória indicadas pelos campos Origem (10) e Destino (11). O resultado é colocado na posição de memória Destino.
SUB	SUB 10 11	Subtrai do valor da posição de memória indicada pelo campo Destino (11) o valor da posição de memória indicada pelo campo Origem (10). O resultado é colocado na posição de memória Destino.
MUL	MUL 10 11	Multiplica os valores das posições de memória indicadas pelos campos Origem (10) e Destino (11). O resultado é colocado na posição de memória Destino.

DIV	DIV 10 11	Divide o valor da posição de memória indicada pelo campo Destino (11) pelo valor da posição de memória indicada pelo campo Origem (10). O resultado é colocado na posição de memória Destino.
OR	OR 10 11	Executa a operação lógica OR bit-a-bit nos valores das posições de memória indicadas pelos campos Origem (10) e Destino (11). O resultado é colocado na posição de memória Destino.
AND	AND 10 11	Executa a operação lógica AND bit-a-bit nos valores das posições de memória indicadas pelos campos Origem (10) e Destino (11). O resultado é colocado na posição de memória Destino.
JUMP	JUMP 10	Atribui ao PC o valor indicado pelo campo Destino.
JUMPZ	JUMPZ 10	Atribui ao PC o valor indicado pelo campo Destino (10), se o valor de AC - acumulador - for zero.
JUMPN	JUMPN 10	Atribui ao PC o valor indicado pelo campo Destino (10), se o valor de AC - acumulador - for negativo.
HALT	HALT	Para a execução do programa.

Tabela 1. Representação de algumas funções.

3.3. Funcionamento

O processador foi implementado na linguagem de programação Java, possuindo interface gráfica para uma melhor aplicação e entendimento do usuário. Possui especificamente cinco janelas. A janela principal, Ajuda, Digitar, Diagrama e Decodificar Memória.

Para entender o funcionamento interno, é necessário entender como funciona a palavra. Esta foi usada com a metade de 4 bytes ou seja, 16 bits, sendo os 4 bits mais significativo destinado a instrução e o restante divido em duas partes, cada uma com 6 bits, sendo reservado para os dados. Como mostram as Figura 1 e Figura 2.

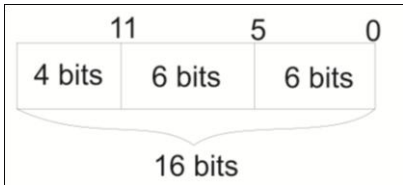

Figura 1. Tamanho da Palavra.

Instrução de 4 bits, possuindo assim até 16 instruções (2^4). Dados possuem 12 bits – Dado1 + Dado2.

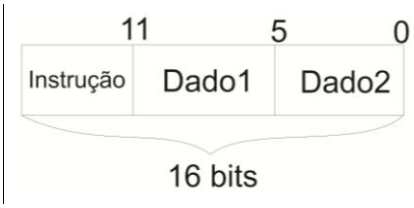

Figura 2. Divisão da Palavra.

A palavra é enviada para a memória, contendo a instrução e os dados, de acordo com os campos da Figura 2. Indo em direção da Unidade de controle, é decodificada e executada assim que o PC – Contador de Programa - indicar.

O diagrama abaixo (Figura 3) mostra toda a representação do processador, onde há a interligação dos componentes para funcionamento do mesmo.

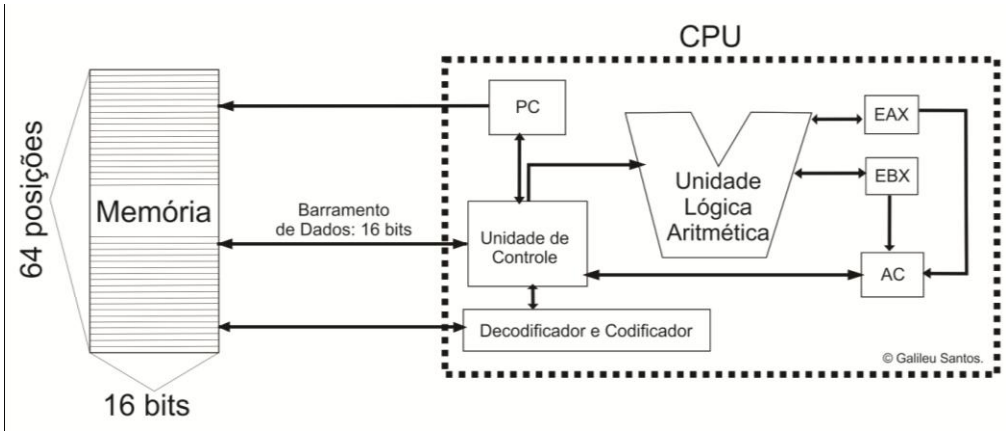

Figura 3. Diagrama de blocos do processador.

A Memória contém um espaço de endereçamento de 16 bits – 4 bytes, com 64 posições.

Há dois registradores, chamados de EAX e EBX, são os responsáveis por armazenamento de dados. E o AC – acumulador, responsável por armazenar a execução da última instrução.

A unidade de controle é a responsável por receber os dados e executar as ações, advindas da memória, que são decodificadas ou codificadas.

O PC – contador de programa é o responsável por manter a ordem de execução dos passos.

A unidade lógica aritmética é a responsável por todas as operações que ocorrem dentro do processador, como por exemplos, operações matemáticas, a memória é acessar de acordo com seu valor, onde o mesmo possui uma variância de 1 a 64, de acordo com o tamanho da memória.

A palavra é o elemento principal do funcionamento do processador, é importante saber como cada posição é interpretada. Primeiramente é necessário entender o conceito de deslocamento de bits, e operação lógica. O deslocamento é simples, é feito a partir do operador >> ou <<, representando respectivamente deslocamento pra direito e pra esquerda. Feito em Java da seguinte forma: numero >> 12; Representa um deslocamento de 12 posições para direita, esta é para capturar o valor referente a operação a ser executada. Antes disso é feito um "e" condicional, para capturar somente o resultado que está nos quatros bits mais significativo, o mesmo acontece com o restante da palavra, onde é feita pelo código abaixo:

```
inst = inst >>12;  /*  Atribui a variável inst o valor contido nos quatro bits mais significativo */

dado1 = (dado1 & 4032)>>6;  /*atribui a variável dado1, os 6 bits do meio.*/

dado2 = (dado2 & 63); /* atribui a variável dado2 os 6 bits iniciais. Esquerda pra direita. */
```

O valor 4032_{10} = 0000111111000000_2 correspondente da linha acima, corresponde ao "e" bit a bit do valor contigo na variável, e >> desloca 6 posições pra direita, restando assim o valor que está daposição 5 a 11. O processo similar acontece para o dado2, a diferença é que é 63_{10} =

0000000000111111_2 correspondente aos 6 primeiros bits. Uma observação para os seis bits do meio é que é reservado um dos bits como identificador de números negativos – sinal, ficando assim representados por cinco bits os restantes dos números.

Desta forma é separado cada campo pertencente a uma variável e atribuído a novas três, sendo o campo instrução, e dois para dados, onde são analisados e executados. Lembrando que os comandos que não possuem os dois campos, são tratados de formas diferentes, como o HALT, que não possui as duas opções, ou seja, ficam inicializadas como tudo zero.

Este é o principal enfoque do simulador, o restante é analisar o conteúdo obtido e assim realizar as operações de acordo com as mesmas.

4. Resultados

Após a implementação em Java, vem a execução. A janela principal, como mostra a Figura 4, contém todos os botões que permite a simulação – através do botão Executar, todas as funções, localizada no lado direito da janela, e as funções abaixo, como os que abrem outras janelas: Digitar, Ajuda, Diagrama e Decodificar Memória.

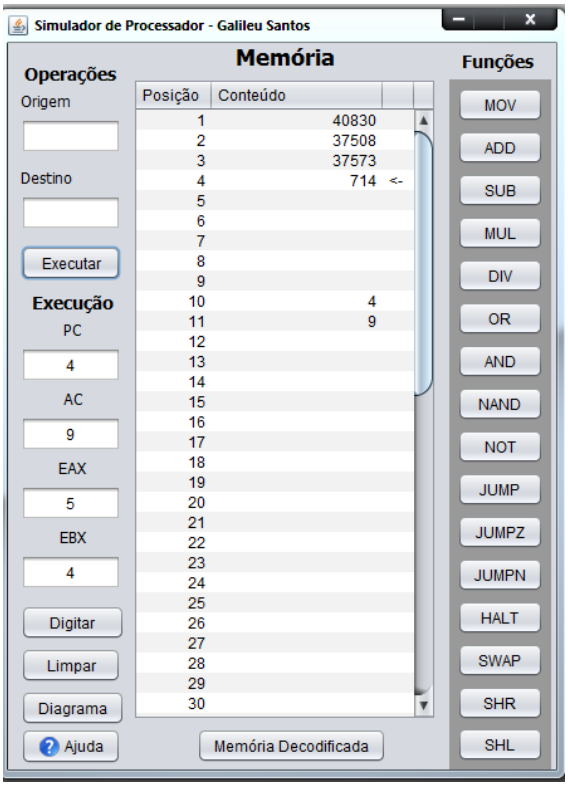

Figura 4. Janela Principal – carregada inicialmente.

Janela "Ajuda", traz uma breve explicação sobre o funcionamento do programa, apresentando ao usuário todo o simulador e o uso de cada função. Como mostra a Figura 5.

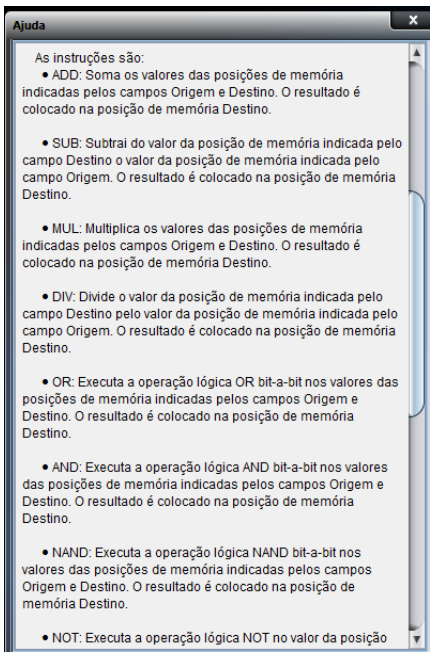

Figura 5. Janela Ajuda – aberta ao clicar em Ajuda.

Janela "Memória Decodificada" permite que o usuário visualize as instruções que estão na memória, e mostrando os passos que estão sendo executados, através da seta <-. Como mostra a Figura 6.

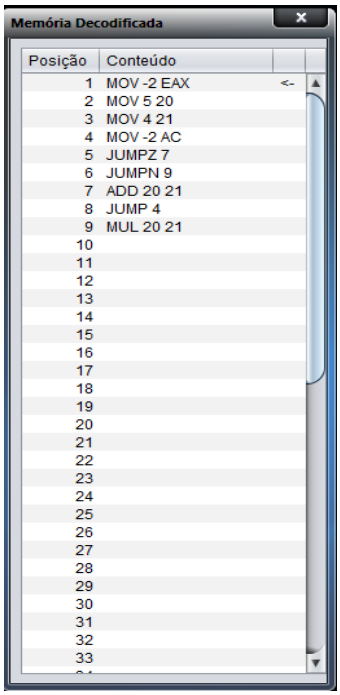

Figura 6. Memória Decodificada – aberta ao clicar no botão de mesmo nome.

A janela "Interpretador" permite que o usuário digite todas as funções que serão simuladas, esta, por sua vez, possui o botão Interpretar, que funciona basicamente como interpretador, interpretando as palavras digitadas e "convertendo" o que foi digitado em instrução, caso siga os requisitos. Como mostra a Figura 7.

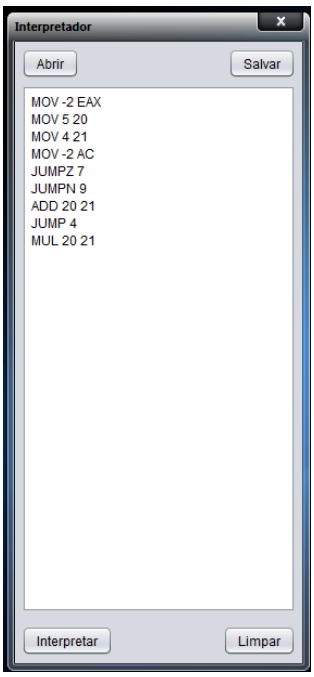

Figura 7. Janela Interpretador – aberta ao clicar em Digitar.

Há também a opção de clicar, onde o usuário digita no campo Origem e Destino, sendo respectivamente representados pelo Dado1e Dado2, em seguida na função que deseja executar. Como mostra a Figura 8. No campo Origem possui o valor 5 e o Destino o valor 11, em seguida o botão MOV foi pressionado, representando a seguinte palavra: MOV 5 11, onde irá mover o valor 5 para a posição de memória 11, representado na memória por uma palavra do tipo inteiro, que é codificado e mostrado na janela "Memória Decodificada", como mostra a Figura 6.

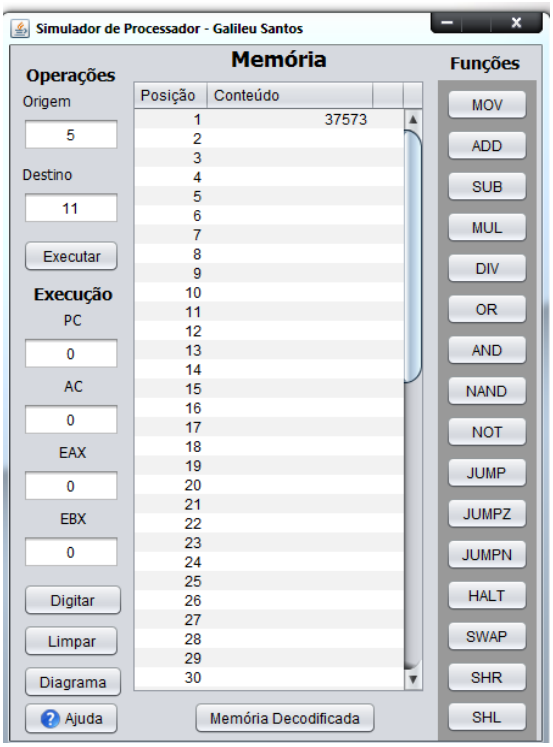

Figura 8. Executando a operação MOV.

A execução do programa é feita a partir do carregamento das instruções na memória e ao clicar no botão Executar, é feita as respectivas ações.

A figura abaixo (Figura 9) mostra a execução passo a passo da realização de algumas instruções representadas no diagrama, sendo o mesmo interativo, mudando ao passo que as instruções são alteradas.

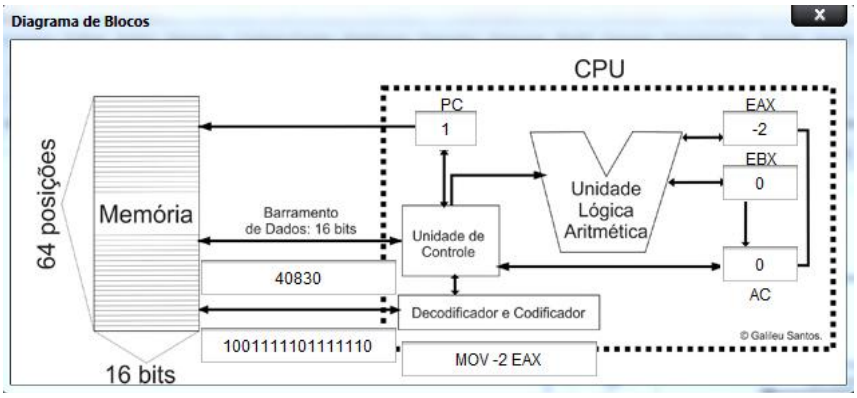

Figura 9. Diagrama de Blocos – aberto ao clicar em Diagrama.

A figura 10 mostra todas as janelas em execução, todos os componentes em interação, sendo eles, os citados anteriormente. Realizando testes digitando na janela “Digitar”, executando o código digitado em todo o simulador, seguindo os passos e suas respectivas representações.

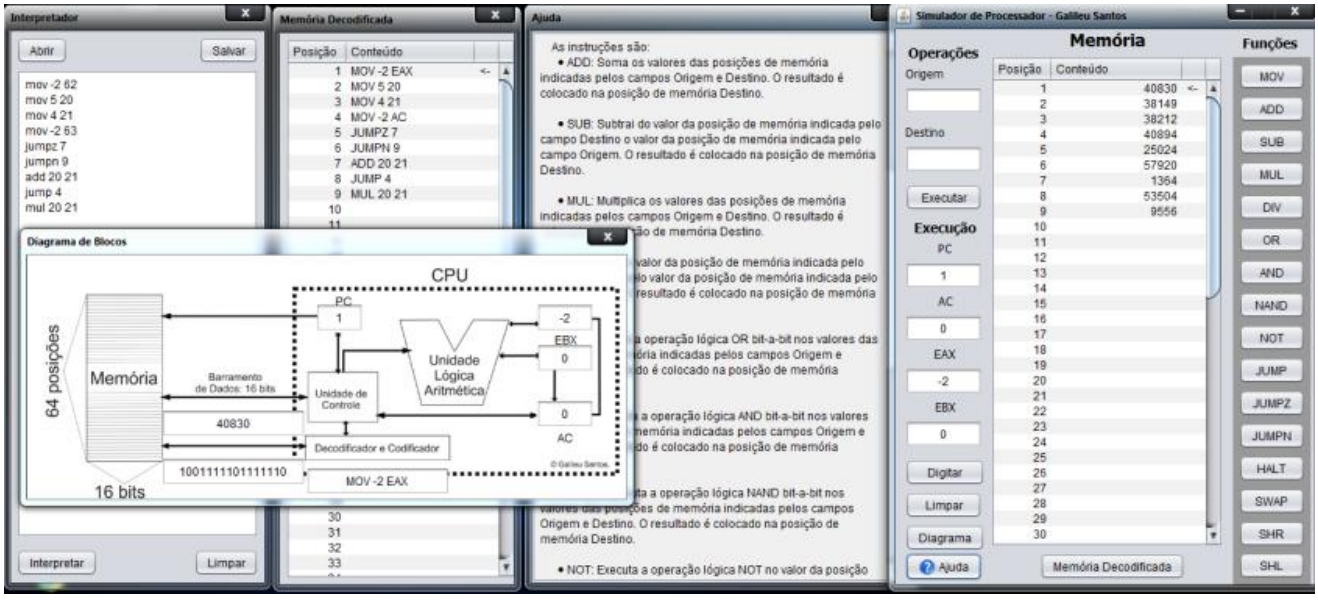

Figura 10. Janelas de todos os componentes do simulador em execução.

5. Conclusão

O objetivo foi cumprido, desde a realização do pré-projeto até sua realização, sempre há algumas mudanças na execução, o que não deixou de acontecer. Foram implementadas todas as funções citadas no pré-projeto, bem como toda a interligação do diagrama, utilizando a linguagem de programação Java, com interface gráfica, auxiliando e facilitando o aprendizado dos alunos que assim utilizem deste projeto, bem como dos professores usando-o em sala de aula como conteúdo didático.

Referências

[1] http://highered.mcgraw-hill.com/sites/0072467509/

[2] http://inf.ufrgs.br/~vbuaraujo/sw/neander/neander.html

[3] http://www.dcc.ufrj.br/~gabriel/neander.php

[4] http://www.ppgee.pucminas.br/weac/2008/PDF/WEAC-2008-Artigo-05.pdf

[5] ftp://ftp.inf.ufrgs.br/pub/inf107/

[6] http://highered.mcgraw-hill.com/sites/0072467509/

Todos os links foram acessados em Agosto-Outubro de 2013.