Universidade de Brasília

Instituto de Ciências Exatas

Departamento de Ciência da Computação

116394 - Organização e Arquitetura de Computadores

Relatório - Trabalho I: Simulador MIPS 1ª parte

Aluno:
Lucas Santos - 14/0151010

Professor:
Ricardo Jacobi

11 de setembro de 2016

1 Descrição do Problema

O trabalho consiste na elaboração de um simulador da arquitetura MIPS, como o *MARS*. A primeira parte do trabalho aborda a carga de instruções e dados gerados pelo montador *MARS*. O simulador eleborado deve ler arquivos binários contendo as instruções e os dados para sua memória e exibir seu conteúdo na tela.

As instruções deste trabalho começam no endereço $0x00000000$ e se encerram no endereço $0x00000044$, enquanto os dados no endereço $0x00002000$ e acabam no endereço $0x0000204c$. A memória simulada é definida com o tamanho de $4K$Words, ou seja, $16K$Bytes. Tomando como exemplo, o código a seguir:

```
1  .data
2
3  primos: .word   1,3,5,7,11,13,17,19
4  size:   .word   8
5  msg:    .asciiz "Os oito primeiros numeros primos sao: "
6  space:  .ascii  " "
7
8  .text
9          la $t0, primos # Carrega o endereco inicial do array de primos
10         la $t1, size # Carrega o endereco do size
11
12         lw $t1, 0($t1) # Carrega size em t1
13
14         li $v0, 4 # Impressao da msg
15         la $a0, msg
16         syscall
17
18 loop:
19         beq $t1, $zero, exit # Se percorreu todo o array, encerra
20
21         li $v0, 1 # Impressao de um inteiro
22         lw $a0, 0($t0) # Inteiro a ser exibido
23         syscall
24
25         li $v0, 4 # Impressao do space
26         la $a0, space
27         syscall
28
29         addi $t0, $t0, 4 # Incrementa indice array
30         addi $t1, $t1, -1 # Decrementa contador
31         j loop
32
33 exit:
34         li $v0, 10 # Termino do programa
35         syscall
```

Ao executar o código no montador *MARS*, os dados e as instruções gerados, exibidos a seguir em hexadecimal, são ilustrados nas Figuras 1 e 2. Os dados gerados serão armazenados a partir do endereço $0x00002000$ na memória simulada, enquanto as instruções a partir do endereço $0x00000000$.

Para exibir o conteúdo da memória simulada na tela, a elaboração de funções se faz necessária. Tais funções serão apresentadas e descritas na próxima seção.

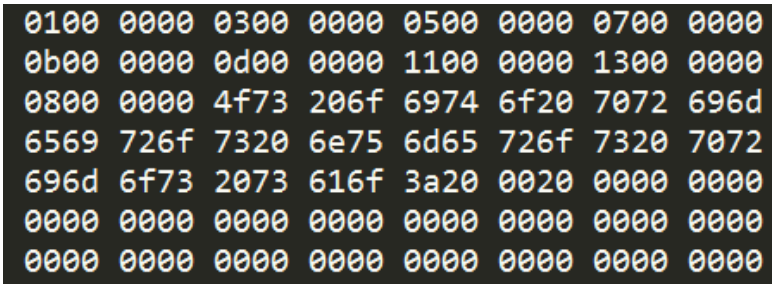

Figura 1: Conteúdo do arquivo gerado *data.bin*.

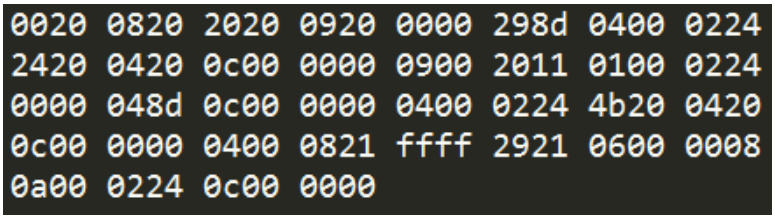

Figura 2: Conteúdo do arquivo gerado *text.bin*.

2 Descrição das Funções Implementadas

2.1 Funções de carregamento na memória

2.1.1 *Load Data - void loadData(FILE *ponteiroArquivo)*

Função que carrega o conteúdo do arquivo *data.bin* a partir do endereço $0x00002000$ para a memória simulada.

2.1.2 *Load Text - void loadText(FILE *ponteiroArquivo)*

Função que carrega o conteúdo do arquivo *text.bin* a partir do endereço $0x00000000$ para a memória simulada.

2.2 Funções de Leitura

2.2.1 *Load Word - int32 lw(uint32 endereco, int16 deslocamento)*

Função que lê uma palavra na memória, ou seja, 4 bytes. Se o endereço ou o deslocamento não forem múltiplos de 4, a função retorna o código de erro $0xfffffffe$, -2 em decimal. Caso contrário ela retorna a palavra armazenado no endereço+deslocamento.

Se o deslocamento for igual a 0, a palavra atual é lida, se o deslocamento for igual a 4, a próxima palavra é lida.

2.2.2 *Load Halfword - int32 lh(uint32 endereco, int16 deslocamento)*

Função que lê uma meia-palavra na memória, ou seja, 2 bytes. Se o endereço não for múltiplo de 4 ou o deslocamento não for múltiplo de 2, a função retorna o código de erro $0xfffffffe$, -2 em decimal. Caso contrário ela retorna a meia-palavra armazenada no endereço+deslocamento.

Se a meia-palavra for negativa, o retorno é uma palavra em complemento de 2. Ex: $0x1000 = -8$, retorno $= 0x11111000 = -8$.

Se o deslocamento for igual a 0, a meia-palavra menos significativa é lida, se o deslocamento for igual a 2, a meia-palavra mais significativa é lida.

2.2.3 *Load Halfword Unsigned - int32 lhu(uint32 endereco, int16 deslocamento)*

Função que lê uma meia-palavra na memória, ou seja, 2 bytes. Se o endereço não for múltiplo de 4 ou o deslocamento não for múltiplo de 2, a função retorna o código de erro $0x$fffffffe, -2 em decimal. Caso contrário ela retorna a meia-palavra armazenada no endereço+deslocamento.

Se o deslocamento for igual a 0, a meia-palavra menos significativa é lida, se o deslocamento for igual a 2, a meia-palavra mais significativa é lida.

2.2.4 *Load Byte - int32 lb(uint32 endereco, int16 deslocamento)*

Função que lê um byte na memória, ela retorna o conteúdo armazenado no endereço+deslocamento.

Se o byte for negativo, o retorno é uma palavra em complemento de 2. Ex: $0x10 = -2$, retorno $= 0x11111110 = -2$.

Se o deslocamento for igual a 0, o byte menos significativo é lido, se o deslocamento for igual a 1, o segundo byte menos significativo é lido, se o deslocamento for igual a 2, o segundo byte mais significativo é lido, se o deslocamento for igual a 3, o byte mais significativo é lido.

2.2.5 *Load Byte Unsigned - int32 lbu(uint32 endereco, int16 deslocamento)*

Função que lê um byte na memória, ela retorna o conteúdo armazenado no endereço+deslocamento.

Se o deslocamento for igual a 0, o byte menos significativo é lido, se o deslocamento for igual a 1, o segundo byte menos significativo é lido, se o deslocamento for igual a 2, o segundo byte mais significativo é lido, se o deslocamento for igual a 3, o byte mais significativo é lido.

2.3 Funções de Escrita

2.3.1 *Store Word - void sw(uint32 endereco, int16 deslocamento, int32 dado)*

Função que escreve uma palavra na memória, ou seja, 4 bytes. Se o endereço ou o deslocamento não forem múltiplos de 4, a função retorna o código de erro $0x$fffffffe, -2 em decimal. Caso contrário ela armazena o dado no endereço+deslocamento.

Se o deslocamento for igual a 0, o dado é armazenado na palavra atual, se o deslocamento for igual a 4,o dado é armazenado na próxima palavra.

2.3.2 *Store Halfword - void sh(uint32 endereco, int16 deslocamento, int16 dado)*

Função que escreve uma meia-palavra na memória, ou seja, 2 bytes. Se o endereço não for múltiplo de 4 ou o deslocamento não for múltiplo de 2, a função retorna o código de erro $0x$fffffffe, -2 em decimal. Caso contrário ela armazena o dado no endereço+deslocamento.

Se o deslocamento for igual a 0, o dado é armazenado na meia-palavra menos significativa, se o deslocamento for igual a 2, o dado é armazenado na meia-palavra mais significativa.

2.3.3 *Store Byte - void sb(uint32 endereco, int16 deslocamento, int8 dado)*

Função que escreve um byte na memória, ela armazena o dado no endereço+deslocamento.

Se o deslocamento for igual a 0, o dado é armazenado no byte menos significativo, se o deslocamento for igual a 1, o dado é armazenado no segundo byte menos significativo, se o deslocamento for igual a 2, o dado é armazenado no segundo byte mais significativo, se o deslocamento for igual a 3, o dado é armazenado no byte mais significativo.

3 Testes e Resultados

Os testes foram desenvolvidos no arquivo de testes, os resultados obtidos foram iguais aos resultados obtidos pelo *MARS*, portanto o programa desenvolvido foi considerado satisfatório.