Memória do MIPS
Simulador

Thiago Santos Matos, 17/0063666

[1] Dep. Ciência da Computação – Universidade de Brasília (UnB)
CiC 116394 - Organização e Arquitetura de Computadores - Turma C

thiago.matos882@gmail.com

1. Introdução

O programa é capaz de simular as instruções de acesso a memória do MIPS, foi desenvolvido em linguagem C e compilado no Windows (GCC-6.3.0-1).

2. Simulador

2.1. Problema

O acesso à memória no MIPS só ocorre por meio das instruções *load* e *store*, sendo *store* a instrução que armazena dados na memória e *load* a instrução que lê dados da memória. Os dados são armazenados na memória em formato hexadecimal, e os endereços da memória são dados na forma de byte. Considerando o funcionamento da memória e das instruções de acesso, um vetor será utilizado para representar uma memória de 16KBytes, sendo que cada índice desse vetor contém os 32 bits necessários para armazenar uma *word* em formato hexadecimal.

2.2. Funções

void sw(uint32_t address, int16_t kte, int32_t dado): Verifica a validade do endereço e deslocamento recebidos, respectivamente *address* e *kte*, e se válido, armazena a *word* no endereço selecionado.
void sh(uint32_t address, int16_t kte, int16_t dado): Verifica a validade do endereço e deslocamento recebidos, respectivamente *address* e *kte*, e se válido, armazena a *half word* no endereço selecionado.
void sb(uint32_t address, int16_t kte, int8_t dado): Verifica a validade do endereço e deslocamento recebidos, respectivamente *address* e *kte*, e se válido, armazena o *byte* no endereço selecionado.
int32_t lw(uint32_t address, int16_t kte): Verifica a validade do endereço e deslocamento recebidos, respectivamente *address* e *kte*, e se válido, retorna a *word* presente no endereço selecionado.
int32_t lh(uint32_t address, int16_t kte): Verifica a validade do endereço e deslocamento recebidos, respectivamente *address* e *kte*, e se válido, retorna a *half word* presente no endereço selecionado.
uint32_t lhu(uint32_t address, int16_t kte): Verifica a validade do endereço e deslocamento recebidos, respectivamente *address* e *kte*, e se válido, retorna a *half word* sem sinal presente no endereço selecionado.
int32_t lb(uint32_t address, int16_t kte): Verifica a validade do endereço e deslocamento recebidos, respectivamente *address* e *kte*, e se válido, retorna o *byte* presente no

endereço selecionado.
uint32_t lbu(uint32_t address, int16_t kte): Verifica a validade do endereço e deslocamento recebidos, respectivamente *address* e *kte*, e se válido, retorna o *byte* sem sinal presente no endereço selecionado.
void dump_mem(uint32_t add, uint32_t size): Imprime o conteúdo armazenado na memória entre *add* e *add+size*.

3. Análise dos Resultados

Foi realizado o procedimento de testes proposto na especificação deste trabalho, implementado na função **void specification_test()**.

```
void specification_test() {
    memset(mem, 0, MEM_SIZE);

    sb(0, 0, 0x04); sb(0, 1, 0x03); sb(0, 2, 0x02); sb(0, 3, 0x01);
    sb(4, 0, 0xFF); sb(4, 1, 0xFE); sb(4, 2, 0xFD); sb(4, 3, 0xFC);
    sh(8, 0, 0xFFF0); sh(8, 2, 0x8C);
    sw(12, 0, 0xFF);
    sw(16, 0, 0xFFFF);
    sw(20, 0, 0xFFFFFFFF);
    sw(24, 0, 0x80000000);

    dump_mem(0, 28);

    printf("\n");

    lb(0,0); lb(0,1); lb(0,2); lb(0,3);
    lb(4,0); lb(4,1); lb(4,2); lb(4,3);
    lbu(4,0); lbu(4,1); lbu(4,2); lbu(4,3);
    lh(8,0); lh(8,2);
    lhu(8,0); lhu(8,2);
    lw(12,0); lw(16, 0); lw(20,0);
}
```

Figure 1. Código da função void specification_test()

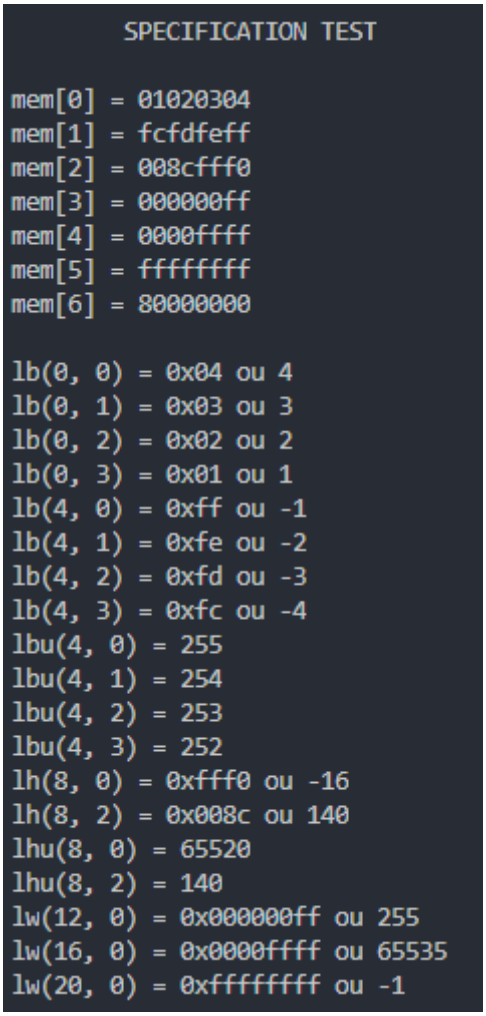

Figure 2. Execução da função void specification_test()

Foram realizados também outros testes implementados na função **void other_test()** com casos de falha e uso dos retornos das operações de *load* para *store*.

```c
void other_test() {
    memset(mem, 0, MEM_SIZE);

    int32_t l;

    sw(0, 1, 0xFFFF7777);
    sw(0, 0, 0xFFFF7777);
    l = lw(0, 0);
    sw(4, 0, l);

    sh(8, 1, 0xFFFF);
    sh(8, 2, 0xFFFF);
    l = lh(8, 2);
    sh(8, 0, l);

    sb(12, 1, 0x55);
    l = lb(12, 1);
    sb(12, 3, l);

    dump_mem(0, 16);
}
```

Figure 3. Código da função void other_test()

```
        OTHER TEST

Endereco Invalido
lw(0, 0) = 0xffff7777 ou -34953
Endereco Invalido
lh(8, 2) = 0xffff ou -1
lb(12, 1) = 0x55 ou 85
mem[0] = ffff7777
mem[1] = ffff7777
mem[2] = ffffffff
mem[3] = 55005500
```

Figure 4. Execução da função void other_test()