Trabalho 1 de Organização e Arquitetura de Computadores 1/2018

Cristiano Silva Júnior: 13/0070629

29 de Março de 2018

1 Introdução

Este é o relato do projeto 1, onde foi pedido para implementar algumas funções de acesso à memória em C utilizando a mesma lógica de acesso da ISA MIPS. A saber, foram determinadas as seguintes funções para implementação:

- *lw*, que lê uma *word* (4 *bytes*) da memória;
- *lh*, que lê uma *half word* (2 *bytes*) com sinal;
- *lhu*, que lê uma *half word* sem sinal;
- *lb*, que lê 1 *byte*;
- *lbu*, que lê 1 *byte* sem sinal;
- *sw*, que escreve uma *word* na memória;
- *sh*, que escreve uma *half word* na memória;
- *sb*, que escreve um *byte* na memória;

Os parâmetros das funções em C devem imitar os parâmetros das funções na linguagem *assembly* original.

Aqui, cabe esclarecer que um *byte* será definido, para os fins deste projeto, como uma lista de tamanho 8 de variáveis booleanas (indivualmente chamadas de bits). Logo, uma *word*, que contém 4 *bytes*, é uma lista de 32 bits. Considerando um *byte* b como sendo

$$b := \{b_i\}_{i=1}^{8}$$

, então algumas operações comumente definidas para variáveis booleanas podem ser extrapoladas para bytes. No caso, vamos trabalhar com as operações "e"e "ou", definidas respectivamente como:

$$a \wedge b := \{a_i \wedge b_i\}_{i=1}^{8}$$

$$a \vee b := \{a_i \vee b_i\}_{i=1}^{8}$$

Uma nova operação a ser definida é o deslocamento (*shift*), que pode ser escrito como sendo direcionado para a direita:

$$b \gg x := \{\top \wedge b_{i-x}\}_{i=1}^{8}$$

ou para a esquerda:

$$b \ll x := \{\top \wedge b_{i+x}\}_{i=1}^{8}$$

onde $x \in \mathbb{N}$.

Neste contexto, vamos definir que -1 representa a lista onde todos os elementos são $\top$, ou seja, todos os bits são verdadeiros. Além disso, 0 representa

2 Metodologia

O principal objeto a ser manipulado neste procedimento é a memória. No caso, resolvi implementar a memória como um *array* de *words* com um tamanho fixo de 4096 *words*. As instruções na ISA MIPS acessam a memória *byte* a *byte* [1], ou seja, cada endereço se refere a um byte diferente. Desta forma, para ler e escrever na memória, certas operações se fazem necessárias para implementar as funções.

Para ler *bytes* e da memória, preferi utilizar uma lógica de máscara. O valor lido v do k-ésimo byte de um endereço a da memória de words M será

$$v = \big(M\lfloor a \rfloor \gg 8k\big) \wedge -1$$

indicando que o valor lido originalmente da memória será ajustado e mascarado para se recuperar o *byte* em questão. A mesma operação pode ser utilizada para *half words*.

Para escrever na memória, a mesma operação de leitura pode ser utilizada, mas agora adaptando a máscara para o deslocamento de endereço desejado e levando o dado a ser guardado em consideração. Neste caso, a função do valor guardado v para um dado d será:

$$v = \big(M[a] \wedge P\big) \vee d \ll 8k$$

onde P é a máscara correspondente para o deslocamento necessário. Essa operação também pode ser utilizada para *half words* porém máscaras próprias deverão ser utilizadas.

Essas funções assumem algumas situações:

- O endereço a se refere a uma posição de um byte dentro da memória;
- k é positivo, e menor que 4 para *bytes* ou par para *half words*. Valores fora dessa faixa sempre retornam o valor 0.

3 Resultados

A implementação encontra-se no código fonte em anexo ao trabalho. O código é escrito em **ANSI C** e pode ser compilado chamando o programa *make* em qualquer sistema operacional com o pacote **GCC**.

4 Conclusão

As operações realizadas no trabalho serviram para modelar matematicamente o acesso à memória realizada por operações da ISA MIPS. Nota-se que o endereçamento original, apesar de ser feito a *word*, precisa respeitar a posição dos *bytes* individuais, necessitando de atenção na construção do hardware necessário.

5 Referência Bibliográfica

1. "MIPS32® Architecture For Programmers". Volume II.