Organização e Arquitetura de Computadores Trabalho 2: Funções de Acesso a Memória do MIPS

Aluno: Gabriel Martins de Miranda
Matricula: 13/0111350
Turma: C

1. Observações

→ Compile digitando make no terminal.

→ São gerados 2 executáveis: main e teste. Com teste tem-se o código de teste.

2. Objetivos

→ Implementar as funções de acesso à memória encontradas no MIPS (load byte, load half, load word, store byte, store half e store word).

→ Mostrar resultados obtidos de um código de teste.

3. Parâmetros

• *É importante* destacar que o tipo int8_t é definido pelo compilador como um char, e por isso aprensenta diversos problemas, incluindo quando se vai fazer shift. Devido a isto, foi substituido por um int comum.

→ uint32_t address = endereço que se deseja acessar.
→ uint16_t kte = constante adicionada ao endereço.
→ int dado = byte a ser escrito na memória.
→ int16_t dado = meia-palavra a ser escrita na memória.
→ int32_t dado = palavra a ser escrita na memória.

3. Funções

→ *int* lb(uint32_t address, uint16_t kte);

- *t0 (celula de memoria) => t0 = mem[celula]*
- *t1 (offset sem mascara) => t1 = t0 >> offset*8*
- *s0 (offset com mascara) => s0 =mascara_byte & t1*

→ int16_t lh(uint32_t address, uint16_t kte);

- *t0 (celula de memoria) => t0 = mem[celula]*
- *t1 (offset sem mascara) => t1 = t0 >> offset*8*
- *s0 (offset com mascara) => s0 = mascara_half & t1*

→ int32_t lw(uint32_t address, uint16_t kte);

- *t0 (celula de memoria) => t0 = mem[celula]*
- *t1 (celula de memoria) => t1 = t0*
- *s0 (celula de memoria) => s0 = t1*

→ void sb(uint32_t address, uint16_t kte, int dado);

- *t0 (celula de memoria) => t0 = mem[celula]*
- *t1 (byte a ser escrito com offset) => t1 = dado << offset*8*
- *s0 (novo valor da celula de memoria) =>*

  *s0 = ((~(mascara_byte << (offset*8)) & t0) | t1)*

→ void sh(uint32_t address, uint16_t kte, int16_t dado);

- *t0 (celula de memoria) => t0 = mem[celula]*
- *t1 (byte a ser escrito com offset) => t1 = dado << offset*8*
- *s0 (novo valor da celula de memoria) =>*

  *s0 = ((~(mascara_half << (offset*8)) & t0) | t1)*

→ void sw(uint32_t address, uint16_t kte, int32_t dado);

- *t0 (novo valor da celula de memoria) => t0 = dado*
- *t1 (novo valor da celula de memoria) => t1 = t0*
- *s0 (novo valor da celula de memoria) => s0 = t1*