Relatório - Trabalho 2
Organização e Arquitetura de Computadores - Turma C

Gabriel Vieira de Arimatéa, 15/0126956

[1]CIC – Universidade de Brasília (UnB)

vieira.arimatea@gmail.com

1. Descrição do problema

O trabalho consite em implementar um simulador da arquitetura MIPS em linguagem C, C++ ou Java, que deveria realizar funções de busca de instrução (*fetch()*), decodificação da instrução (*decode()*) e execução da instrução (*execute()*). as instruções do programa viriam dos arquivos *texte* e *data* obtidos através do montador MARS no formato binário.

2. Descrição sucinta das funções implementadas

Além das funções implementadas no trabalho 1 (**lw**, **lh**, **lhu**, **lb**, **lbu**, **sw**, **sh**, **sb** e **dump_mem**), as funções implementadas foram:

2.1. add

Consiste em somar o conteudo de dois registradores e armazenar em um terceiro;

2.2. sub

Consiste em subtrair o conteudo de dois registradores e armazenar em um terceiro;

2.3. mult

Consiste em multiplicar o conteudo de dois registradores e armazenar o valor em dois registradores: **hi** e **lo**;

2.4. div

Consiste em dividir o conteudo de dois registradores e armazenar o quociente no registrador **lo** e o resto no **hi**;

2.5. and

Realiza a operação lógica *and*, bit a bit, entre dois registradores e armazena o resultado em um terceiro;

2.6. or

Realiza a operação lógica *or*, bit a bit, entre dois registradores e armazena o resultado em um terceiro;

2.7. xor

Realiza a operação lógica *xor*, bit a bit, entre dois registradores e armazena o resultado em um terceiro;

2.8. nor

Realiza a operação lógica *nor*, bit a bit, entre dois registradores e armazena o resultado em um terceiro;

2.9. slt

Caso o primeiro registrador seja menor que o segundo, ele armazena o valor 1 em um terceiro registrador. Caso contrário, armazena 0;

2.10. sll

Armazena em um registrador, o valor de outro registrador deslocado para a esquerda. O campo shamt determina quanto será deslocado;

2.11. slr

Armazena em um registrador, o valor de outro registrador deslocado para a direita. O campo shamt determina quanto será deslocado;

2.12. sra

Armazena em um registrador, o valor de outro registrador deslocado para a direita. O campo shamt determina quanto será deslocado;

2.13. mfhi

Move para um registrador o valor que esta em *hi*;

2.14. mflo

Move para um registrador o valor que esta em *lo*;

2.15. addu

Consiste em somar o conteudo de dois registradores *unsigneds* e armazenar em um terceiro;

2.16. sltu

Caso o primeiro registrador seja menor que o segundo, ele armazena o valor 1 em um terceiro registrador. Caso contrário, armazena 0. Nesse caso não é considerado o bit de sinal;

2.17. jr

Pula para o endereço armazenado em um registrador;

2.18. syscall

Chamada de sistema;

2.19. sltiu

Caso o registrador seja menor que um imediato, ele armazena o valor 1 em um terceiro registrador. Caso contrário, armazena 0. Nesse caso não é considerado o bit de sinal;

2.20. andi

Realiza a operação lógica *and*, bit a bit, entre um registrador e um imediato, armazenando o resultado em um registrador;

2.21. ori

Realiza a operação lógica *ori*, bit a bit, entre um registrador e um imediato, armazenando o resultado em um registrador;

2.22. xori

Realiza a operação lógica *xori*, bit a bit, entre um registrador e um imediato, armazenando o resultado em um registrador;

2.23. lui

Armazena um valor imadiato em um registrador;

2.24. addi

Consiste em somar o conteudo de um registrador com um imediato e armazenar em um segundo registrador;

2.25. slti

Caso um registrador seja menor que um imediato, ele armazena o valor 1 em um terceiro registrador. Caso contrário, armazena 0;

2.26. addiu

Consiste em somar o conteudo de um registrador com um imediato e armazenar em um segundo registrador. Nesse caso não se considera o bit de sinal;

2.27. beq

Caso os dois registradores forem iguals, pc será atualizado para o endereço dado à função;

2.28. bne

Caso os dois registradores não forem iguals, pc será atualizado para o endereço dado à função;

2.29. blez

Caso um registrador for menor ou igual a zero, pc será atualizado para o endereço dado à função;

2.30. bgtz

Caso um registrador for maior que zero, pc será atualizado para o endereço dado à função;

2.31. j

Salta para um determinado endereço da memória;

2.32. jal

Salta para um determinado endereço da memória e armazena a posição da memória que estava no registrador $ra;

3. Testes e resultados

Realziando o teste do código apresentado no roteiro do trabalho, obtive o mesmo resultado entrague no montador MARS.

Porém, ao utilizar o código fornecido pelo professor posteriormente, vi que meu código apresenta um erro no syscall, pois ele imprime todas as informações apresentadas no data a partir do endereço entregue. Problema não resolvido.

Outro erro percebido foi no ori() que extendia o sinal da forma que havia sido escrito anteriormente. Para resolver isso, foi aplicado máscaras e não houve mais problemas quanto a isso.