Trabalho 2: Simulador RISC-V

Leonardo Maximo Silva, 200022172

Sumário

1 Apresentação do Problema

O Trabalho consistiu na Implementação da função run() de um Simulador do Processador RISC-V implementado em C, além da implementação das funções básicas da Linguagem de Programação Assembly em C. Como decisão de implementação do problema, decidiu-se por realizar a implementação das funções em um arquivo próprio chamado "funcoes.h"e inseri-lo no código principal utlizando "#include<funcoes.h>". As funções de memória, realizadas no trabalho 1, foram inseridas em um arquivo denominado "memoria.h", os quais já estavam incluídos no Arquivo.

As Funções foram implementadas no IDE CodeBlocks no Sistema Operacional Windows 10 64 bits.

2 Descrição das Funções implementadas

A seguir, são descritas as funções implementadas

2.1 Run

Utilizando a função já implementada "$load_mem()$", leu-se os dois arquivos binários "code"e "data"obtidos a partir do arquivo "Teste.asm"utilizando, respectivamente, as posições de memória 0x00000000 e 0x00002000, como as posições de início na escrita do array memoria. A variável "$Data_SEGMENT_START$"foi atribuída como sendo o tamanho retornado pela função "$load_mem(”code”, 0x00000000)$". Após essas atribuições, o processador executava em loop a função step() já implementada até o

tamanho da memória do Simulador do Processador implementado ($i < Data_SEGMENT_START$) ou até um comando de parada de programa.

2.2 Add

Realiza a soma simples entre os valores contidos em dois registradores (rs1 e rs2) e coloca o valor no registrador rd. Instrução de tipo R.

2.3 Addi

Realiza a soma entre os valores contidos em um registrador (rs1) e um imediato dado na instrução do tipo I.

2.4 And

Realiza a operação and () bit a bit entre os valores contidos nos registradores rs1 e rs2 e armazena o resultado no registrador rd. Instrução de tipo R.

2.5 Andi

Realiza a operação and () bit a bit entre o valor contido no registrador rs1 e um imediato definido pela instrução de tipo I e armazena o resultado no registrador rd.

2.6 AuiPC

Realiza a soma de um imediato, o qual possui 32 bits, sendo os 12 menos significativos como sendo 0 e os 20 mais significativos definidos na instrução de tipo U, com a instrução localizada no Registrador PC e armazena o valor em um registrador rd. É importante ressaltar que essa instrução não altera o registrador PC e que, no Simulador, o imediato utilizado já está tratado de modo a possuir seus 12 bits menos significativos como 0 e os 20 bits mais significativos definidos na Instrução.

2.7 Beq

Compara os valores contidos em dois registradores rs1 e rs2 e, caso sejam iguais, incrementa a instrução atual do registrador pc (pc-4 devido ao incremento da instrução antes da função execute()) com um valor imediato do tipo SB. Os valores contidos em rs1 e rs2 são comparados considerando a convenção de sinal em complemento de 2.

2.8 Bne

Compara os valores contidos em dois registradores rs1 e rs2 e, caso sejam diferentes, incrementa a instrução atual do registrador pc (pc-4 devido ao incremento da instrução antes da função execute()) com um valor imediato do tipo SB. Os valores contidos em rs1 e rs2 são comparados considerando a convenção de sinal em complemento de 2.

2.9 Bge

Compara os valores contidos em dois registradores rs1 e rs2 e, caso rs1 seja maior ou igual a rs2, incrementa a instrução atual do registrador pc (pc-4 devido ao incremento da instrução antes da função execute()) com um valor imediato do tipo SB. Os valores contidos em rs1 e rs2 são comparados considerando a convenção de sinal em complemento de 2.

2.10 Bgeu

Compara os valores contidos em dois registradores rs1 e rs2 e, caso rs1 seja maior ou igual a rs2, incrementa a instrução atual do registrador pc (pc-4 devido ao incremento da instrução antes da função execute()) com um valor imediato do tipo SB. É importante ressaltar que, nesse caso, os valores contidos em rs1 são comparados sem considerar o sinal, de modo que esses valores sejam interpretados apenas como números não negativos.

2.11 Blt

Compara os valores contidos em dois registradores rs1 e rs2 e, caso rs1 seja menor que rs2, incrementa a instrução atual do registrador pc (pc-4 devido ao incremento da instrução antes da função execute()) com um valor imediato do tipo SB.

2.12 Bltu

Compara os valores contidos em dois registradores rs1 e rs2 e, caso rs1 seja menor que rs2, incrementa a instrução atual do registrador pc (pc-4 devido ao incremento da instrução antes da função execute()) com um valor imediato do tipo SB. É importante ressaltar que, nesse caso, os valores contidos em rs1 são comparados sem considerar o sinal, de modo que esses valores sejam interpretados apenas como números não negativos.

2.13 Jal

Armazena a próxima instrução em um registrador RD (pc+4) e torna o valor no registrador PC como sendo igual à soma do valor contido em um registrador rs1 e um imediato do tipo UJ.

2.14 Jalr

Armazena a próxima instrução em um registrador RD (pc+4) e torna o valor no registrador PC como sendo igual à soma do valor contido em um registrador rs1 e um imediato do tipo I.

2.15 Or

Realiza a operação or (|) bit a bit entre os valores contidos nos registradores rs1 e rs2 e armazena o resultado no registrador rd. Instrução de tipo R.

2.16 Lui

Iguala o valor contido em um registrador rd a um imediato do tipo U que possui 32 bits, sendo seus 12 bits menos significativos 0 e os outros 20 bits dados pela instrução. O imediato recebido por essa instrução já se encontra tratado e igual ao valor desejado.

2.17 Slt

Compara os valores contidos nos registradores rs1 e rs2 de modo que, se rs1 $<$ rs2, armazena 1 em um registrador rd e, caso contrário, armazena 0

em rd. Instrução do tipo R. Os valores contidos em rs1 e rs2 são comparados considerando a convenção de sinal em complemento de 2.

2.18 Sltu

Compara os valores contidos nos registradores rs1 e rs2 de modo que, se rs1 $<$ rs2, armazena 1 em um registrador rd e, caso contrário, armazena 0 em rd. Instrução do tipo R. É importante ressaltar que, nesse caso, os valores contidos em rs1 são comparados sem considerar o sinal, de modo que esses valores sejam interpretados apenas como números não negativos.

2.19 Ori

Realiza a operação and (|) bit a bit entre o valor contido no registrador rs1 e um imediato definido pela instrução de tipo I e armazena o resultado no registrador rd.

2.20 Slli

Armazena o valor de um shift à esquerda de um registrador rs1 e um imediato do tipo I em um registrador rd.

2.21 Srai

Armazena o valor de um shift à direita de um registrador rs1 e um imediato do tipo I em um registrador rd. É importante ressaltar que, nesse caso, os bits inseridos à direita possuem o mesmo valor do bit mais significativo do valor original contido no registrador.

2.22 Srli

Armazena o valor de um shift à direita de um registrador rs1 e um imediato do tipo I em um registrador rd. É importante ressaltar que, nesse caso, os bits inseridos à direita possuem o valor 0, de modo que o valor contido em rs1 pode ser tratado como não possuindo sinal (unsigned).

2.23 Sub

Realiza a subtração entre os valores contidos nos registradores rs1 e rs2 e armazena em um registrador rd. Instrução do tipo R.

2.24 Xor

Realiza a operação xor ($\wedge$) bit a bit entre o valor contido no registrador rs1 e um imediato definido pela instrução de tipo I e armazena o resultado no registrador rd.

2.25 Load Word

Define o valor em um registrador RD como sendo o valor de 32 bits contido em um endereço de memória dado pela soma de um registrador rs1 e um Imediato de 12 bits do tipo S. É importante ressaltar que o endereço deve ser múltiplo 4 para corresponder a um espaço de memória válido.

2.26 Load Byte

Define o valor em um registrador RD como sendo um byte (8 bits) contido em um endereço de memória dado pela soma de um registrador rs1 e um Imediato de 12 bits do tipo S. O Byte é interpretado como tendo um sinal dado em complemento de 2.

2.27 Load Byte Unsigned

Define o valor em um registrador RD como sendo um byte (8 bits) contido em um endereço de memória dado pela soma de um registrador rs1 e um Imediato de 12 bits do tipo S. O Byte é interpretado como não possuindo um sinal (unsigned).

2.28 Store Word

Define o valor em um espaço de memória dado pelos valores contidos em um registrador rs1 e um imediato de 12 bits do tipo S como sendo o valor de 32 bits contido em um registrador rs2. É importante ressaltar que o endereço deve ser múltiplo 4 para corresponder a um espaço de memória válido.

2.29 Store Byte

Define o valor em um espaço de memória dado pelos valores contidos em um registrador rs1 e um imediato de 12 bits do tipo S como sendo o valor de 1 byte (8 bits) contido em um registrador rs2.

2.30 Ecall

A partir dos valores contidos no registrador A0, realiza 3 operações diferentes. Caso o valor contido em A0 seja 1, exibe o valor inteiro contido no registrador A7. Caso A0 seja 4, interpreta o valor contido em A7 como sendo uma palavra codificada em ASCII, foi utilizada a função lb() para realizar o load de um bit e intepretá-lo como um caractere e exibi-lo na tela. Caso A0 seja 10, torna a variável "$stop_prg = 1$"e para a execução do programa.

3 Testes e Resultados

Todos os 22 testes realizados no Código de Teste "Teste.asm"foram realizado com sucesso, conforme pode ser observado em 1.

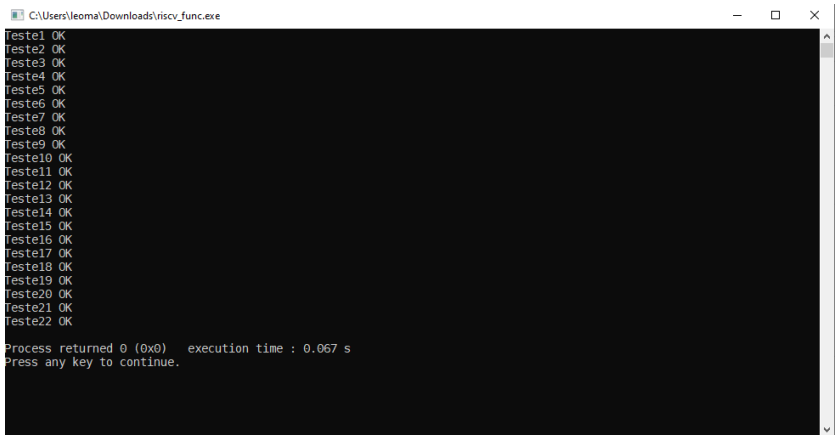

Figura 1: Resultados dos Testes Realizados no Simulador do RISC-V

4 Conclusão

O Simulador do Processador RISC-V foi, portanto, implementado com sucesso a partir dos materiais fornecidos para a sua confecção, de modo que todos os testes realizados em 1 foram realizados com sucesso.