Universidade Federal da Paraíba
Centro de Informática

Arquitetura de Computadores II

Relatório Unidade de Controle

Alunos Vinícius Matheus Veríssimo da Silva - 11409543
Matheus Lima Moura de Araújo - 11409518

Professor Alisson Vasconcelos de Brito

João Pessoa, 20 de abril de 2017

Conteúdo

1 Introdução

O presente trabalho, proposto durante as aulas da disciplina de Arquitetura de Computadores II na Universidade Federal da Paraíba, tem como principal objetivo a implementação de uma Unidade de Controle Micro programada.

1.1 Ambiente de Testes

Na implementação desse trabalho foi utilizadas a linguagem de programação *Python*. Os testes foram realizados em uma máquina com 8GB de memória DDR3, processador Intel® CoreTM i7-6500U CPU @ 2.50GHz x 4 e sistema operacional Ubuntu 16.04.

2 Descrição do Problema

Definir e implementar uma Unidade de Controle Micro programada que possuam em seu conjunto instruções para pelo menos:

- Executar 4 operações aritméticas;
- Saltos condicional e incondicional;
- As duas etapas da execução devem ser bem definidas e separadas: 1) ciclo de busca e 2) ciclo de execução
- Registradores devem ser modelados como variáveis do programa;
- Memória deve ser modelada como um array ou outra coleção equivalente;
- Apresentar um pequeno programa como exemplo que utilize as instruções implementadas.

3 Arquitetura

A memória de dados e instruções foram divididas em estruturas de dados distintas, a memória de dados foi limitada a 16 espaços de endereçamento e a memória de instruções é equivalente ao tamanho do programa que está sendo executado.

A quantidade de registradores de dados foi limitada a 4 (R0,R1,R2,R3), além do IR (registrador de instruções) e do PC (contador de programa).

Foram definidas flags para a operação de comparação de valores, as flags são um conjunto de variáveis booleanas que guardam os resultados das comparações, como: comparação quanto a igualdade ou superioridade dos valores, ou simplesmente saber se um dado valor é igual a 0.

4 Sintaxe

A sintaxe desenvolvida contém treze operações, atendendo todos os requisitos do problema.

4.1 Operações Aritméticas

O *destino* pode ser tanto a memória de dados quanto algum registrador. A *fonte* pode ser tanto um registrador como um valor constante.

ADD *destino*, *fonte*, *fonte*
SUB *destino*, *fonte*, *fonte*
MULT *destino*, *fonte*, *fonte*
DIV *destino*, *fonte*, *fonte*

4.2 Saltos Incondicionais

O *address* pode ser tanto o índice da instrução contido na memória de instrução ou o programador pode criar um rótulo com o nome que preferir. Este rótulo deve ser único em uma linha do programa.

JMP *address*

Exemplo com uso de rotulação:

ADD *R0*, *R1*, *R2*
JMP *ROTULO*
MULT *R0*, *R1*, *R1*
ROTULO
MULT *R0*, *R2*, *R2*

Exemplo com uso do índice da memória de instrução:

ADD *R0*, *R1*, *R2*
JMP *MI[3]*

MULT *R0*, *R1*, *R1*
MULT *R0*, *R2*, *R2*

4.3 Saltos Condicionais

Para utilizar os operadores de saltos condicionais é necessário utilizar o operador **CMP** que pode ser executado de duas maneiras:

CMP *fonte1*, *fonte2* - na qual a *fonte* é obrigatoriamente um registrador;

CMP *fonte1* - para comparar se é igual a zero.

Após o operador de comparação, algumas flags são acionadas. A partir destas flags o salto é realizado ou não. O conjunto de flags contém 4 bits e são distribuídos da seguinte maneira:

0001 - *Significa que fonte1 == fonte2*
0010 - *Significa que fonte1 > fonte2*
0100 - *Significa que fonte1 < fonte2*
1000 - *Significa que fonte1 == zero*

Após o uso do operador de comparação, utiliza-se um dos operadores de salto condicional, de acordo com o objetivo do programador. Os operadores condicionais são:

JE *address* - Salte se bit 0 de *flags* estiver ativo.
JG *address* - Salte se bit 1 de *flags* estiver ativo.
Jl *address* - Salte se bit 2 de *flags* estiver ativo.
JZ *address* - Salte se bit 3 de *flags* estiver ativo.

4.4 MOV, LOAD, STORE

A instrução **MOV** pode ser feita entre registradores ou adicionar um valor constante a um registrador. Exemplo:

MOV *R0*, *R1*
MOV *R0*, *5*

A instrução **LOAD** é feita apenas entre o registrador e a memória de dados, passando o dado da memória para o registrador. Exemplo:

LOAD *R0*, *MD[3]*

A instrução **STR** é feita entre a memória de dados e o registrador ou entre a memória de dados e algum valor constante, com o objetivo de salvar um certo dado na memória. Exemplo:

STR *MD[2]*, *R0*
STR *MD[2]*, *5*

5 Exemplos de programas

5.1 Fatorial de 5

Exemplo de um programa que calcula o fatorial de 5.

MOV R0, 1
MOV R1, 5
MOV R3, R1
LOOP
MULT R3, R0, R3
ADD R0, R0, 1
CMP R0, R1
JE FIM
JMP LOOP
FIM
STORE MD[0], R3

5.2 Fibonacci

Exemplo de programa para calcular os 10 primeiros números da sequência de Fibonacci. Cada valor da sequência vai sendo guardado no registrador R3 e sendo sobrescrito pelo seu sucessor.

MOV R0,0
MOV R1,1
MOV R2,10
LOOP
ADD R3,R1,R0
MOV R0,R1
MOV R1,R3
SUB R2,R2,1
CMP R2
JZ FIM
JMP LOOP
FIM