Datapath de um Processador Multi-ciclo

Guimarães, João Guilherme M.
joaog95@live.com

Muniz, Lucas L. R.
lucaslc01@hotmail.com

23 de junho de 2019

1 Objetivo da prática

A oitava aula prática da disciplina de Laboratório de Arquitetura de Computadores I, teve como objetivo explicar o funcionamento do módulo de Controle e o *datapath* das nove instruções implementadas.

2 Módulo de Controle

A unidade lógica de Controle em um processador multi-ciclo, é responsável por gerenciar os valores assumidos pelas variáveis de controle de cada módulo (*controlAlu, controlMux, etc...*), tendo em vista o ciclo e a instrução em execução.

Segue abaixo parte do código responsável pelo funcionamento do módulo de Controle.

```
    ...
01. case(Step)
02.     2'b00:
03.         begin
04.             writeEnableRegInstruction <= 1'b1;
05.             writeEnableRegisterFile <= 1'b0;
06.             incr_pc  <= 1'b1;
07.             ReadAddressRF1 <= instruction[8:6];
08.             ReadAddressRF2 <= instruction[5:3];
09.             writeEnableRegAddress <= 1'b0;
10.             controlMux <= 2'b10;
11.             W <= 1'b0;
12.          end
13.
14.     2'b01:
```

```
15.         begin
16.             case(instruction[15:12])
17.                 4'b1101:  // 2° Ciclo da Instrução Load
18.                     begin
19.                         incr_pc  <= 1'b0;
20.                         writeEnableRegInstruction <= 1'b0;
21.                         writeEnableRegAddress <= 1'b1;
22.                         controlMux <= 2'b01;
23.                         ReadAddressRF1 <= instruction[8:6];
24.                         ReadAddressRF2 <= instruction[5:3];
25.                     end
26.                 4'b1100:  // 2° Ciclo da Instrução Store
27.                     begin
28.                         incr_pc  <= 1'b0;
     ...
```

Analisando o código acima, é possível perceber a distinção realizada pelo módulo de Controle do ciclo (linhas *02* e *14*) e da instrução (linhas *17* e *26*), para que cada par ciclo-instrução, tenha os valores corretos (linhas *19* à *24*).

3 Descrição das Instruções

Obs.: como o primeiro e o último ciclo são iguais para todas as instruções, eles terão os *datapaths* descritos separadamente.

ciclo 0: este ciclo é responsável por incrementar o valor de PC e realizar o armazenamento da instrução a ser executada, no respectivo registrador.

ciclo 3: é responsável por resetar o valor de qualquer variável de controle crítica, preparar a leitura da próxima instrução a ser executada e caso a instrução atual for de *store*, habilita a escrita do dado na memória.

add(0000) / or(0001) / and(0010) / not(0011): instrução responsável por realizar a operação de soma / *or* bit-a-bit (||) / *and* bit-a-bit (&&) / *not* bit-a-bit (!) dos registradores **Y** e **Z**, e armazenar o valor resultante no registrador **X**.

C1 - consiste em ler os dados dos dois registradores passados como parâmetro e armazenar o resultado da operação aritmética no registrador *regALU*.

C2 - seleciona a porta do *MUX* que tenha o valor de *regALU*, para que este dado seja enviado para o módulo *RF* e armazenado no registrador **X**.

store(1100): instrução responsável por armazenar o dado do registrador **X** na posição de memória que se encontra no registrador **Y**.

C1- leitura dos dados dos registradores **X** e **Y**, e armazenamento da informação de **Y** em *regDadress*, utilizando o controle do *MUX*.

C2- seleciona a porta do *MUX* correspondente ao valor de **X**, para que este seja armazenado em *regDout*.

C3- escreve o dado na memória lendo os registradores *regDout* e *regDadress*.

load(1101): instrução responsável por ler dados da memória, consiste em obter o endereço de memória previamente armazenado no registrador **Y** e salvar o dado lido no registrador **X**.

C1- transfere o dado armazenado no registrador **Y** para o registrador *regDadress*.

C2- leitura do dado da memória utilizando o endereço armazenado em *regDadress*, e o armazena no registrador **X**.

conditional copy(1011): realiza a instrução *copy* se e somente se, o dado armazenado em *regAlu* for 0.

C1- leitura do registrador **Y**.

C2- verificação do valor armazenado por *regAlu*, caso seja 0, escreve o dado de **Y** em **X**, caso contrário, não realiza nenhuma operação.

copy(1110): instrução responsável por copiar o dado no registrador **Y** e enviar para o registrador **X**.

C1- leitura do registrador **Y**.

C2- habilitação da escrita para sobrescrever o dado do registrador **X**.

copy input(1111): instrução responsável por armazenar o valor de *DataIn* no registrador **X**, para isso, é realizado a leitura do próximo endereço de memória armazenado pelo **PC**.

C1- é realizado o incremento do valor de *PC* e o armazena em *regDadress*.

C2- leitura da próxima posição de memória devido ao incremento de *PC* e armazenamento do dado lido no registrador **X**.

*Obs.: todos os processos de escrita em RF utilizam o endereço do registrador **X**, devido a decisão de projeto (processador.v, linha 24, coluna 49).*

4 Conclusão

Como citado anteriormente, o módulo de Controle é responsável por gerenciar os valores assumidos pelas variáveis de controle de cada módulo, e com este papel, é nele que ocorre a criação das nove instruções citadas acima (cada *case* do *switch* de ciclos).

Devido a sua complexidade, o módulo de Controle é deixado como o último a ser implementado, porém somente com sua adesão aos outros módulos, é possível realizar testes no código de forma automatizada (criação de *scripts*), sendo assim, é nesta fase que se encontra os primeiros problemas do desenvolvimento, pois alguma instrução pode implicar em quebra de paradigma. Para evitar problemas desta natureza, é necessário que o projeto do processador multi-ciclo seja bem definido e todas as instruções se complementem.