

Lista 3

1)

a) Configurei $t1 = 2$, $t2 = 5$. No final obtivemos $t0 = 7$, como esperado.

IF:

- $PC = PC + 4$.
- A instrução em PC é lida no Instruction Memory.
- A instrução e $(PC + 4)$ são salvos no registrador intermediário IF/ID.

ID:

- A instrução é distribuída pelo Instruction Register, com $rs = 09$ e $rt = 0a$.
- O imediato é estendido para 32 bits.
- O imediato, rs e rt são salvos no registrador intermediário ID/EX.

EX:

- O imediato é multiplicado por 4 e somado com $PC + 4$ (preparação dos branches).
- O valor de rt é selecionado no mux para ser somado com o valor de rs (soma desejada).
- O resultado da soma, sinal zero, rt , e $((\text{imediato} \ll 2) + (PC + 4) + 4)$ são salvos no registrador intermediário EX/MEM.

MEM:

- O resultado do branch é negado no mux do que será escrito no PC.
- O resultado da soma é enviado para ser selecionado como endereço para o Data Memory, com $\text{MemRead} = 0$.
- O valor de rt é enviado para ser escrito na memória, com $\text{MemWrite} = 0$.
- O valor "lido" da memória e o resultado da soma são salvos no registrador intermediário MEM/WB.

WB:

- O resultado da soma é selecionado no mux e enviado o Write Data do Instruction Register.
- O resultado da soma é escrito em rd ($t0 - 08$), com $\text{RegWrite} = 1$.

b) São necessários 5 ciclos.

2)

a) $t1 = 8001fffc$ e $8001fffc = 10\ 00\ 00\ 01$.

IF:

- $PC = PC + 4$.
- A instrução em PC é lida no Instruction Memory.
- A instrução e $(PC + 4)$ são salvos no registrador intermediário IF/ID.

ID:

- A instrução é distribuída pelo Instruction Register, com $rs = 09$, $rt = 08$ e imediato = 0000.

- O imediato é estendido para 32 bits.
- O imediato, rs são salvos no registrador intermediário ID/EX.

EX:

- O imediato é multiplicado por 4 e somado com PC + 4 (preparação dos branches).
- O valor do imediato é selecionado no mux para ser somado com o valor de rs (cálculo do endereço).
- O resultado da soma, sinal zero, rt, e $((\text{imediato} \ll 2) + (\text{PC} + 4) + 4)$ são salvos no registrador intermediário EX/MEM.

MEM:

- O resultado do branch é negado no mux do que será escrito no PC.
- O resultado da soma é enviado para ser selecionado como endereço para o Data Memory, com MemRead = 1.
- O valor de rt é enviado para ser escrito na memória, com MemWrite = 0.
- O valor lido da memória e o resultado da soma são salvos no registrador intermediário MEM/WB.

WB:

- O valor lido na memória é selecionado no mux e enviado o Write Data do Instruction Register.
- O valor lido na memória é escrito em rt ($t0 - 08$), com RegWrite = 1.

- Uma soma é feita na ALU para efetuar o cálculo do endereço que será lido da memória.
- São necessários 5 ciclos.
- Sim, todos os estágios do Pipeline são utilizados.
- a) $t1 = 8001fff8$ e $t0 = 10000001$.

IF:

- $PC = PC + 4$.
- A instrução em PC é lida no Instruction Memory.
- A instrução e $(PC + 4)$ são salvos no registrador intermediário IF/ID.

ID:

- A instrução é distribuída pelo Instruction Register, com rs = 09, rt = 08 e imediato = 0004.
- O imediato é estendido para 32 bits.
- O imediato, rs e rt são salvos no registrador intermediário ID/EX.

EX:

- O imediato é multiplicado por 4 e somado com PC + 4 (preparação dos branches).
- O valor do imediato é selecionado no mux para ser somado com o valor de rs (cálculo do endereço).
- O resultado da soma, sinal zero, rt, e $((\text{imediato} \ll 2) + (\text{PC} + 4) + 4)$ são salvos no registrador intermediário EX/MEM.

MEM:

- O resultado do branch é negado no mux do que será escrito no PC.

- O resultado da soma é enviado para ser selecionado como endereço para o Data Memory, com MemRead = 0.
- O valor de rt é enviado para ser escrito na memória, com MemWrite = 1.
- O valor “lido” da memória e o resultado da soma são salvos no registrador intermediário MEM/WB.

WB:

- Nenhum valor selecionado no mux servirá de algo, já que RegWrite = 0.
- b) Uma soma é feita na ALU para efetuar o cálculo do endereço que será lido da memória.
- c) São necessários 4 ciclos, mas o pipeline força que sejam executados os 5 ciclos.
- d) Não, nem todos os estágios do Pipeline são utilizados.
- f) a) $t1 = 00000000$, $t0 = 00000000$

IF:

- $PC = PC + 4$.
- A instrução em PC é lida no Instruction Memory.
- A instrução e $(PC + 4)$ são salvos no registrador intermediário IF/ID.

ID:

- A instrução é distribuída pelo Instruction Register, com rs = 09, rt = 08 e imediato = 0004.
- O imediato é estendido para 32 bits.
- O imediato, rs e rt são salvos no registrador intermediário ID/EX.

EX:

- O imediato é multiplicado por 4 e somado com $PC + 4$ (preparação dos branches).
- O valor do rt é selecionado no mux para ser comparado com o valor de rs (comparação).
- O resultado da comparação, sinal zero, rt, e $((\text{imediato} \ll 2) + (PC + 4) + 4)$ são salvos no registrador intermediário EX/MEM.

MEM:

- O resultado do branch é escrito no PC.
- A saída da ALU é enviada para ser selecionado como endereço para o Data Memory, com MemRead = 0.
- O valor de rt é enviado para ser escrito na memória, com MemWrite = 0.
- O valor “lido” da memória e a saída da ALU são salvos no registrador intermediário MEM/WB.

WB:

- Nenhum valor selecionado no mux servirá de algo, já que RegWrite = 0.
- b) Uma comparação é feita na ALU para saber se o branch será necessário ou não.
- c) São necessários 4 ciclos, no 5º o valor já será usado.
- d) Não, nem todos os estágios do Pipeline são utilizados.

3)

- a) O registrador t2 recebe o resultado correto no 5º ciclo.
- b) O resultado é necessário para segunda instrução no 3º ciclo.
- c) O problema é que a segunda instrução depende da resposta da primeira instrução, mas essa resposta só está pronta depois da segunda instrução precisar, fazendo com que a segunda instrução não seja executada da forma desejada.
- d) Executado.
- e) Sim, dessa vez executou da forma desejada. Pois acontece o forwarding, ou seja, o resultado da primeira instrução é enviado adiantadamente (antes de ser escrito no registrador) para ser utilizado na segunda instrução.

4)

- a) São necessários 7 ciclos até que o branch ser executado.
- b) Ela é executada normalmente enquanto o branch é executado um ciclo à frente.
- c) O problema é que isso faz com que o addi salve seu cálculo mesmo que a condição de branch seja satisfeita.
- d) Acontece um erro muito similar, a única diferença é que o branch é executado no quinto ciclo, utilizando atalhos por hardware. Mas, ainda assim, a instrução add é finalizada, fazendo com que o programa não siga o que foi intencionado.

5)

- a) Após 6 ciclos, o registrador t1 recebe o valor correto.
- b) A instrução sub precisa ler o valor de t1 do Instruction Register no 5º.
- c) Como esse valor não chegou lá ainda, ele acaba lendo o valor errado. Esse problema pode ser resolvido através de um atalho por hardware, ou de uma reorganização das instruções, ou de uma adição de bolhas (nop), e etc.
- d) O resultado foi bastante inesperado. Aparentemente, no lugar de ser executada a instrução: sub t2, t1, t3; foi executada a instrução: sub t2, t2, t3, ou seja, o tratamento não está sendo feito corretamente.

6)

- a) A funcionalidade do programa é um simples algoritmo de ordenação (aparentemente, um Bubble Sort). Cheguei nessa conclusão executando o código no papel:


OAC

A: 5, 6, 2, 3, 1 (começo de: 8002 0090)

| a0 | a1 | t0 | t1 | t2 | t3 | t4 | t5 | t6 |
|-----------|----|----|----|-----------|-----------|----|----|----|
| 8002 0090 | 5 | 0 | 1 | 8002 0090 | 4 | 5 | 6 | 1 |
| | | 1 | 2 | 8002 0090 | 8002 0094 | 5 | 2 | 0 |
| | | 2 | 3 | 8002 0090 | 8 | 2 | 3 | 1 |
| | | 3 | 4 | 8002 0090 | 8002 0098 | 2 | 1 | 0 |
| | | 4 | 5 | 4 | 12 | 2 | 5 | 0 |
| | | 5 | 2 | 8002 0094 | 8002 0012 | 6 | 3 | 0 |
| | | | 3 | | 16 | 5 | 2 | 0 |
| | | | 4 | | 8002 0016 | 3 | 5 | 0 |
| | | | 5 | | | 6 | 3 | 0 |
| | | | | | | 5 | 5 | 0 |
| | | | | | | 6 | 5 | 0 |

a0: posição 0 do array
 a1: tamanho do array
 t1: j - iterador
 t0: i - iterador

array: 5, 6, 2, 3, 1
 2, 6, 5, 3, 1
 1, 6, 5, 3, 2
 1, 5, 6, 3, 2
 1, 3, 6, 5, 2
 1, 2, 6, 5, 3
 1, 2, 5, 6, 3
 1, 2, 3, 6, 5
 1, 2, 3, 5, 6



(Note que a à medida que fui executando o código, eu pulei alguns passos óbvios como: calcular a posição, onde ele faz:

add t2, t0, t0

add t2, t2, t2

```
add t2, a0, t2)
```

(Um código similar em C seria:

```
for (int i = 0; i < tamanho; i ++)
```

```
    for (int j = i + 1; j < tamanho; j ++)
```

```
        if (!(array[i] <= array[j]))
```

```
            swap(array[i], array[j]);
```

```
)
```

(A utilização de mult reduz um pouco o tempo de execução)

(A execução passo a passo de cada questão pode ser encontrada no link:
<https://1drv.ms/f/s!Av7ZeqaYG5k949tanXHiFky2ueBbTg> ou <https://goo.gl/n6hyMu>)