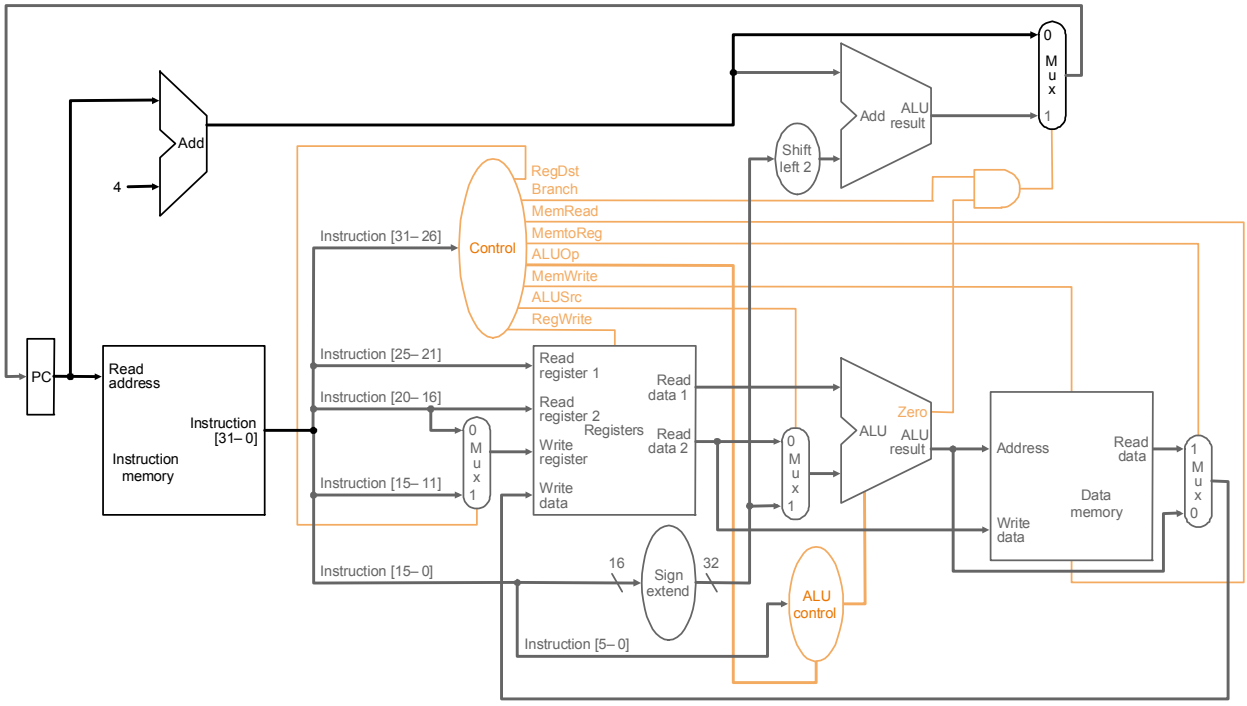
**Prova P2** **Disciplina: Organização e Arquitetura de Computadores II**

**Professor: Fabiano Hessel – Ney Calazans**

Aluno: 10/junho/2009

**Valor das questões: 1) 3 pontos 2) 4 pontos 3) 3 pontos**

1. Considere o bloco de dados mono-ciclo da arquitetura MIPS e o seu bloco de controle desenhado abaixo. A seguir, observe os sinais de controle representados na tabela abaixo do desenho. Determine se existe algum sinal de controle, com exceção do sinal MemtoReg (cuja compatibilidade com MemRead já foi estudada em aula) que possa ser eliminado e substituído por um outro sinal de controle existente sem alterar a funcionalidade de nenhuma instrução. Justifique.




| Instr / Sinal | RegDst | ALUSrc | Memto Reg | Reg Write | Mem Read | Mem Write | Branch | ALUOp1 | ALUp0 |
|---|---|---|---|---|---|---|---|---|---|
| Formato R | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| sw | X | 1 | X | 0 | 0 | 1 | 0 | 0 | 0 |
| beq | X | 0 | X | 0 | 0 | 0 | 1 | 0 | 1 |

2. Dados:
   - o trecho de programa no quadro abaixo e os conteúdos iniciais de registradores e posições de memória relevantes;
   - o diagrama pipeline vazio que segue o trecho;
   - a organização pipeline do MIPS desenhada em seguida,

| | |
|---|---|
| **addi $t4, $zero, 2**<br>**root : add $t1, $t2, $t3**<br>**lw $t3, 0x100($t1)**<br>**sw $t3, 0x200($t1)**<br>**subi $t4, $t4, 2**<br>**beq $t4, $t3, root**<br>**addi $t3, $t3, 0x100** | Conteúdos iniciais da memória e dos registradores relevantes:<br>$t1=0x100, $t2=0x100, $t3=0x100, $t4=0x100<br>Mem [0x100-0x103]= 0x002345AB<br>Mem [0x200-0x203]= 0x0000000A<br>Mem [0x300-0x303]= 0x00000000<br>Mem [0x400-0x403]= 0x00CD5F00 |

Execute as seguintes tarefas:

a) Simule a execução completa do programa. Use o diagrama pipeline fornecido após o quadro. [2 pontos]

b) O que a unidade de adiantamento (*forward*) está fazendo durante o quinto ciclo de execução? Se algumas comparações estiverem sendo feitas, mencione-as. [1 ponto]

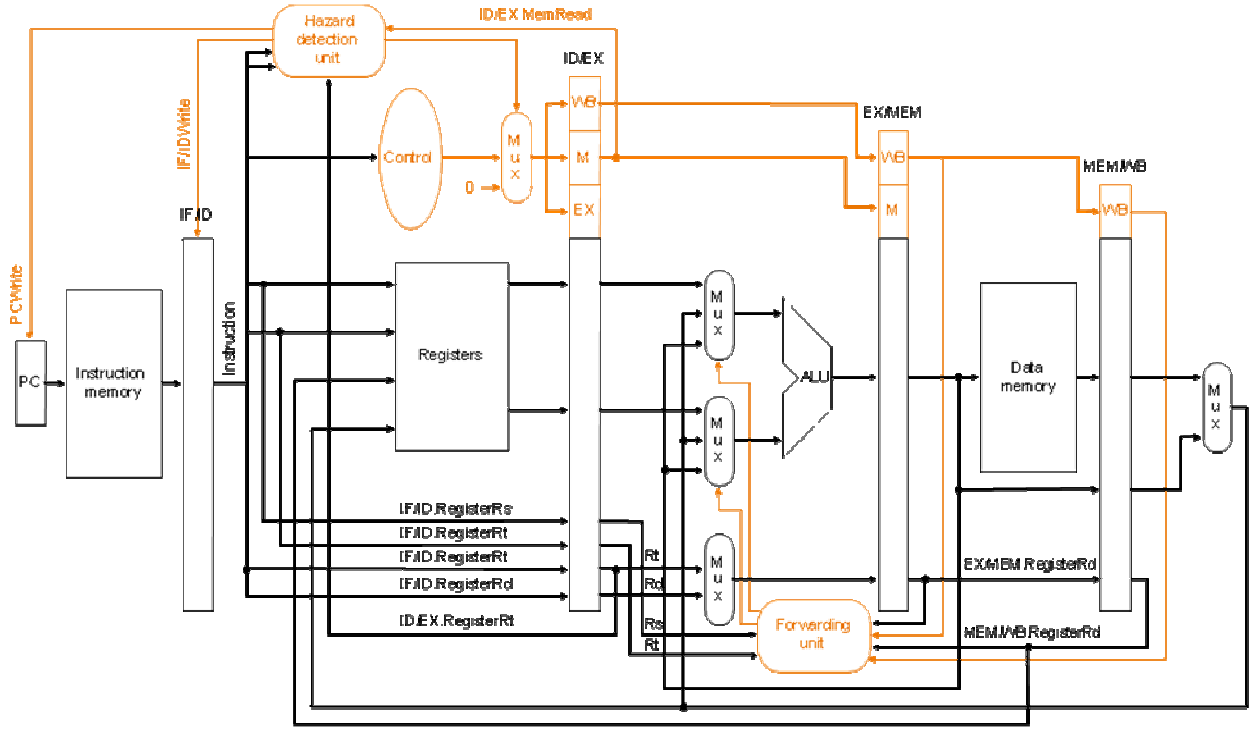
c) No final do vigésimo ciclo de execução do trecho de programa, qual(ais) registradores estão sendo lidos e qual(ais) está(ão) sendo escrito(s) (lembre-se dos estágios em que estas operações ocorrem no pipeline!). [1 ponto]

| INSTRUÇÃO | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **addi $t4, $zero,2** | | | | | | | | | | | | | | | | | | | | | | | | | |
| **root: add $t1, $t2, $t3** | | | | | | | | | | | | | | | | | | | | | | | | | |
| **lw $t3,0x100($t1)** | | | | | | | | | | | | | | | | | | | | | | | | | |
| **sw $t3,0x200($t1)** | | | | | | | | | | | | | | | | | | | | | | | | | |
| **subi $t4, $t4, 2** | | | | | | | | | | | | | | | | | | | | | | | | | |
| **beq $t4, $t3, root** | | | | | | | | | | | | | | | | | | | | | | | | | |
| **addi $t3, $t3,0x100** | | | | | | | | | | | | | | | | | | | | | | | | | |

**Convenções**: **X - bolha** **F - flush do pipeline**

**- - estágio não usado** **⟶ - adiantamento ou leitura após escrita no mesmo ciclo**

**Estágios do pipeline: B (Busca), D (Decodificação), E (Execução) M (Memória) W (Write-back)**



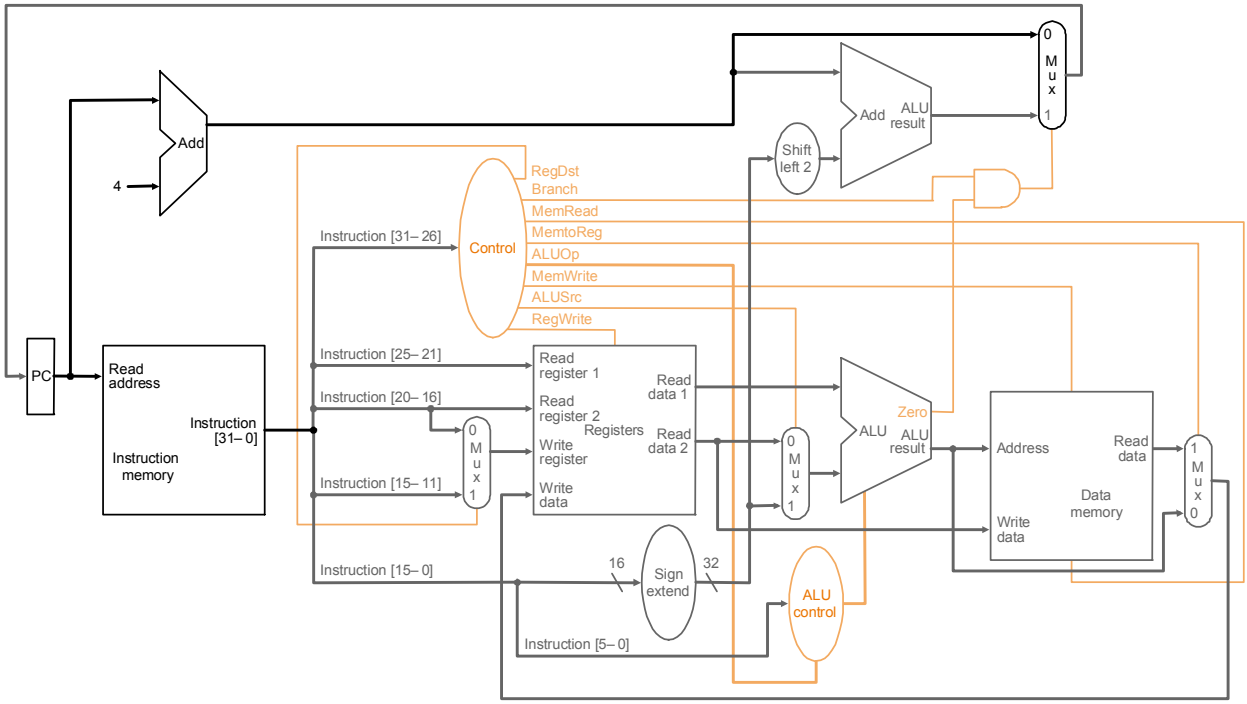

3. Considere a seguinte modificação na arquitetura do conjunto de instruções do processador MIPS: suponha que as instruções de acesso à memória de dados a palavra (**`lw`**/**`sw`**) não mais usam endereçamento base-deslocamento para acesso ao dado que deverá ser escrito no registrador destino, mas apenas endereçamento direto a registrador. As antigas instruções **`lw`** e **`sw`** passam a ser pseudo-instruções e deverão ser implementadas através de duas instruções. Por exemplo, uma pseudo-instrução **`lw $t0,100($t1)`** seria implementada pela seqüência de instruções:

```
addi   $at,$t1,100 # Adiciona o deslocamento ao registrador base, gerando o endereço de acesso
lw     $t0,$at     # Usa a nova forma da instrução lw, ao invés do tradicional. Lê conteúdo de memória do
                   # endereço contido em $at e o copia para $t0 lw $t0, 100($t1)
```

Quais modificações deveriam ser realizadas no Bloco de Dados e no Bloco de Controle para dar suporte a esta nova instrução? Considere a implementação monociclo do MIPS, a mesma da questão desta prova.

# Gabarito da P2 de 10/junho/2009

1. Considere o bloco de dados mono-ciclo da arquitetura MIPS e o seu bloco de controle desenhado abaixo. A seguir, observe os sinais de controle representados na tabela abaixo do desenho. Determine se existe algum sinal de controle, com exceção do sinal MemtoReg (cuja compatibilidade com MemRead já foi estudada em aula) que possa ser eliminado e substituído por um outro sinal de controle existente sem alterar a funcionalidade de nenhuma instrução. Justifique.




| Instr / Sinal | RegDst | ALUSrc | Memto Reg | Reg Write | Mem Read | Mem Write | Branch | ALUOp1 | ALUp0 |
|---|---|---|---|---|---|---|---|---|---|
| Formato R | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| lw | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| sw | X | 1 | X | 0 | 0 | 1 | 0 | 0 | 0 |
| beq | X | 0 | X | 0 | 0 | 0 | 1 | 0 | 1 |

Solução: Observando a tabela, nota-se que o sinal RegDst é compatível com o sinal ALUOp1, e portanto pode ser substituído pelo último. Também Branch e ALUOp0 podem ser substituídos um pelo outro, pois ambos geram os mesmos valores em todas as situações.

2. Dados:
   - o trecho de programa no quadro abaixo e os conteúdos iniciais de registradores e posições de memória relevantes;
   - o diagrama pipeline vazio que segue o trecho;
   - a organização pipeline do MIPS desenhada em seguida,

| | | |
|---|---|---|
| | **addi** | **$t4, $zero, 2** |
| **root :** | **add** | **$t1, $t2, $t3** |
| | **lw** | **$t3, 0x100($t1)** |
| | **sw** | **$t3, 0x200($t1)** |
| | **subi** | **$t4, $t4, 2** |
| | **beq** | **$t4, $t3, root** |
| | **addi** | **$t3, $t3, 0x100** |

Conteúdos iniciais da memória e dos registradores relevantes:

$t1=0x100, $t2=0x100, $t3=0x100, $t4=0x100
Mem [0x100-0x103]= 0x002345AB
Mem [0x200-0x203]= 0x0000000A
Mem [0x300-0x303]= 0x00000000
Mem [0x400-0x403]= 0x00CD5F00

Execute as seguintes tarefas:

a) Simule a execução completa do programa. Use o diagrama pipeline fornecido após o quadro. [2 pontos]

*Resposta:*

| INSTRUÇÃO | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **addi $t4, $zero,2** | B | D | E | M | W | | | | | | | | | | | | | | | | | | | | |
| **add $t1, $t2, $t3** | | B | D | E | M | W | | | | | | B | D | E | M | W | | | | | | | | | |
| **lw $t3,0x100($t1)** | | | B | D | E | M | W | | | | | | B | D | E | M | W | | | | | | | | |
| **sw $t3,0x200($t1)** | | | | B | D | X | X | E | M | W | | | | B | D | X | X | E | M | W | | | | | |
| **subi $t4, $t4, 2** | | | | | B | X | X | D | E | M | W | | | | B | X | X | D | E | M | W | | | | |
| **beq $t4, $t3, root** | | | | | | | | B | D | E | M | W | | | | | | B | D | E | M | W | | | |
| **addi $t3, $t3,0x100** | | | | | | | | | B | D | E | F | F | | | | | | B | D | E | M | W | | |

**Convenções**: **X - bolha** **F - flush do pipeline**
**- - estágio não usado** **⟶ - adiantamento ou leitura após escrita no mesmo ciclo**
**Estágios do pipeline: B (Busca), D (Decodificação), E (Execução) M (Memória) W (Write-back)**

b) O que a unidade de adiantamento (*forward*) está fazendo durante o quinto ciclo de execução? Se algumas comparações estiverem sendo feitas, mencione-as. *[1 ponto]*

*Resposta:* A unidade de adiantamento está realizando duas ações em paralelo:

*1* - Comparando os registradores-fonte da instrução LW $t3,0x100($t1) com os registradores-destino das instruções ADD $t1, $t2, $t3 e ADDI $t4, $zero, 2.

*2* – Adiantando o valor correto de $t1da instrução ADD $t1, $t2, $t3 para a instrução LW $t3,0x100($t1).

c) No final do vigésimo ciclo de execução do trecho de programa, qual(ais) registradores estão sendo lidos e qual(ais) está(ão) sendo escrito(s) (lembre-se dos estágios em que estas operações ocorrem no pipeline!). *[1 ponto]*

*Resposta:*

Lido: $t3

Escrito: $t3

3. Considere a seguinte modificação na arquitetura do conjunto de instruções do processador MIPS: suponha que as instruções de acesso à memória de dados a palavra (**`lw`**/**`sw`**) não mais usam endereçamento base-deslocamento para acesso ao dado que deverá ser escrito no registrador destino, mas apenas endereçamento direto a registrador. As antigas instruções **`lw`** e **`sw`** passam a ser pseudo-instruções e deverão ser implementadas através de duas instruções. Por exemplo, uma pseudo-instrução **`lw $t0,100($t1)`** seria implementada pela seqüência de instruções:

```
addi   $at,$t1,100 # Adiciona o deslocamento ao registrador base, gerando o endereço de acesso
lw     $t0,$at     # Usa a nova forma da instrução lw, ao invés do tradicional. Lê conteúdo de memória do
                   # endereço contido em $at e o copia para $t0 lw $t0, 100($t1)
```

Quais modificações deveriam ser realizadas no Bloco de Dados e no Bloco de Controle para dar suporte a esta nova instrução? Considere a implementação monociclo do MIPS, a mesma da questão 1) desta prova.

*Resposta:* A principal modificação consiste em perceber que não será mais necessário passar através da ULA (cálculo do endereço) para só então acessar a memória de dados. Seria necessário acrescentar uma ligação direta entre a saída Read data 1 do banco de registradores e o barramento de endereços da memória (assumindo que o formato da instrução não será modificado). A saída da ULA não precisa mais ser conectada ao barramento de endereços da memória de dados. O bloco de controle não precisa ser modificado, porém alguns sinais de controle podem ser *don't cares* para as instruções de acesso à memória, tais como ALUSrc, ALUOp0 e ALUOp1.