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

Professor: Fabiano Hessel

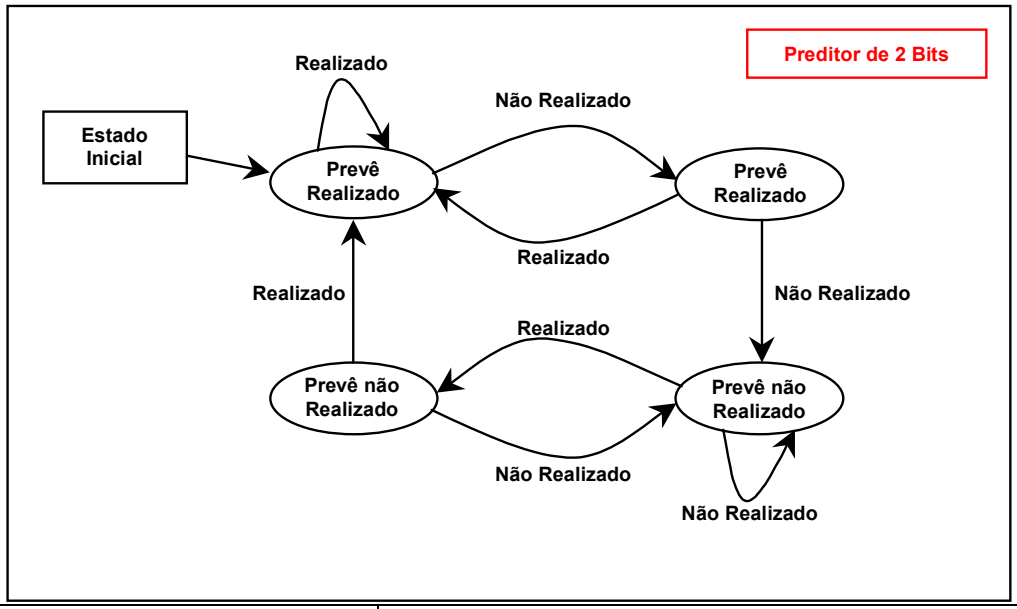
Aluno: 10/novembro/2008

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

1. Suponha a eliminação do sinal de controle MemtoReg no bloco de dados do MIPS. Em seu lugar passaria a ser utilizado o sinal MemRead. Em sua opinião, quais instruções não continuariam funcionando corretamente? Considere na sua resposta os caminhos de dados: mono-ciclo e multi-ciclo. Justifique a sua resposta. Use como referência a implementação disponível nas transparências do Capítulo 5 do livro texto.

Instruction	RegDst	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUp0
R-format	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. Dado o trecho de programa abaixo, e assumindo uma implementação do MIPS com máxima capacidade de solução de conflitos de dados – *forward* (inclusive estrutura mestre-escravo no banco de registradores) e com um preditor de saltos de 2 bits (máxima capacidade de solução de conflitos de controle), com diagrama de transição de estados e estado inicial também dados abaixo, execute as seguintes tarefas:
   a) Simule a execução completa do programa. Use o diagrama pipeline. Assuma para a simulação que a unidade de adiantamento é simples, ela pode apenas adiantar dados que já estejam no processador (na saída do estágio 3 ou do estágio 4) para a entrada da ULA. Ou seja, assuma que não é possível adiantar dados para o estágio 4. *[2 pontos]*
   b) No final do sexto 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]*
   c) O que a unidade de adiantamento (*forward*) está fazendo durante o quarto ciclo de execução? Se algumas comparações estiverem sendo feitas, mencione-as. *[1 ponto]*




```
        addi    $t4, $zero, 2
root :  add     $t1, $t2, $t3
        lw      $t3, 100($t1)
        sw      $t3,  200($t1)
        subi    $t4, $t4, 1
        beq     $t4, $t3, root
        addi    $t3, $t3, 100
```

Conteúdos iniciais da memória e dos registradores relevantes:

$t1=100, $t2=100, $t3=100, $t4=100
Mem [100-103]= 002345ABH
Mem [200-203]= ABFF5BE0H
Mem [300-303]= 00000000H
Mem [400-403]= 00CD5F00H

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 $t3, 100($t1)																									
sw $t3, 200($t1)																									
subi $t4, $t4, 1																									
beq $t4, $t3, root																									
addi $t3, $t3, 100																									

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 adição de uma nova instrução no conjunto de instruções do MIPS chamada **ADDM**. Esta instrução permite que instruções aritméticas façam acesso direto à memória.

   Por exemplo:

```
addm   $t2, 100($t3)    # $t2 = $t2 + Memória[$t3+100]
```

   Escreva um parágrafo explicando porque seria difícil adicionar esta instrução no pipeline do processador MIPS visto em aula.

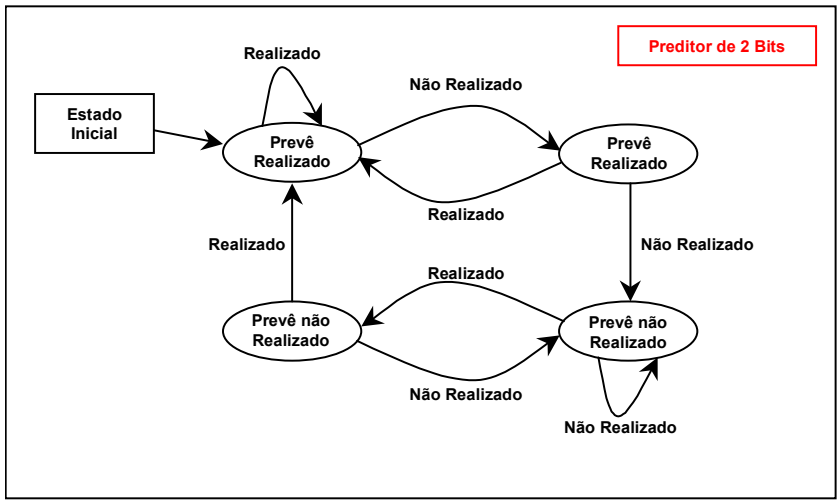
Gabarito da P2 de 10/novembro/2008

1. Suponha a eliminação do sinal de controle MemtoReg no bloco de dados do MIPS. Em seu lugar passaria a ser utilizado o sinal MemRead. Em sua opinião, quais instruções não continuariam funcionando corretamente? Considere na sua resposta os caminhos de dados: mono-ciclo e multi-ciclo. Justifique a sua resposta. Use como referência a implementação disponível nas transparências do Capítulo 5 do livro texto.

Resposta: Observando a tabela, vemos que os dois sinais são idênticos, exceto para as instruções sw e beq, onde o sinal MemtoReg é don't care (ou seja, qualquer valor para estes sinais, neste casos, pode ser usado). Assim, a modificação terá como efeito que na implementação monociclo todas as instruções continuam funcionando corretamente. Na implementação multiciclo contudo, como o sinal MemRead é ativado no quarto ciclo da execução e desativado em todos os demais, e como o sinal MemtoReg é ativado no quinto ciclo e desativado em todos os demais, usar MemRead no lugar de MemtoReg terá como efeito o funcionamento incorreto da instrução **`lw`**, pois esta lerá corretamente o dado da memória, mas não conseguirá ter este valor escrito no banco de registradores, pois o multiplexador não conectará o dado lido da memória à entrada do banco de registradores, mas a saída da ULA. Todas as demais instruções funcionarão sem problema.

2. Dado o trecho de programa abaixo, e assumindo uma implementação do MIPS com máxima capacidade de solução de conflitos de dados (inclusive estrutura mestre-escravo no banco de registradores) e com um preditor de saltos de 2 bits (máxima capacidade de solução de conflitos de controle), com diagrama de transição de estados e estado inicial também dados abaixo, execute as seguintes tarefas:

   a) Simule a execução completa do programa. Use o diagrama pipeline. Assuma para a simulação que a unidade de adiantamento é simples, ela pode apenas adiantar dados que já estejam no processador (na saída do estágio 3 ou do estágio 4) para a entrada da ULA. Ou seja, assuma que não é possível adiantar dados para o estágio 4; *[2 pontos]*



	addi	**$t4, $zero, 2**
root :	**add**	**$t1, $t2, $t3**
	lw	**$t3, 100($t1)**
	sw	**$t3, 200($t1)**
	subi	**$t4, $t4, 1**
	beq	**$t4, $t3, root**
	addi	**$t3, $t3, 100**

Conteúdos iniciais da memória e dos registradores relevantes:

$t1=100H, $t2=100H, $t3=100h, $t4=100H
Mem [100H-103H]= 002345ABH
Mem [200H-203H]= ABFF5BE0H
Mem [300H-303H]= 00000000H
Mem [400H-403H]= 00CD5F00H

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	F	F	F	F										
lw $t3, 100($t1)			B	D	E	M	W																		
sw $t3, 200($t1)				B	D	X	X	E	M	W															
subi $t4, $t4, 1					B	X	X	D	E	M	W														
beq $t4, $t3, root								B	X	X	D	E	M	W											
addi $t3, $t3, 100												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) No final do sexto ciclo de execução do trecho abaixo, 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]*

No 6º. ciclo:

Lidos: nenhum

Escritos: $t1

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

A unidade de adiantamento está comparando os registradores-fonte da instrução ADD $t1, $t2, $t3 com o destino da instrução ADDI $t4, $zero, 2

d) Qual o conteúdo dos registradores *$t1, $t2, $t3,* e *$t4*; e das posições de memória Mem[100H-103H], Mem[200H-203H], Mem[300H-303H], e Mem[400H-403H] ao final da execução completa do programa. *[1 ponto]*

$t1 = 200 $t2 = 100 $t3 = 100 $t4 = 1

Mem [100H-103H]= 002345ABH
Mem [200H-203H]= ABFF5BE0H
Mem [300H-303H]= 00000000H
Mem [400H-403H]= 00000000H

3. Considere a adição de uma nova instrução no conjunto de instruções do MIPS chamada **ADDM**. Esta instrução permite que instruções aritméticas façam acesso direto à memória.

Por exemplo:

```
addm   $t2, 100($t3)    # $t2 = $t2 + Memória[$t3+100]
```

Escreva um parágrafo explicando porque seria difícil adicionar esta instrução no pipeline do processador MIPS visto em aula.

Resposta: A instrução ADDM necessita 6 ciclos: busca, decodificação/leitura dos registradores, cálculo do endereço de memória do operando, leitura da memória, adição e escrita do registrador destino.

Isto significa que precisamos de um estágio de pipeline a mais (e todas as outras instruções deverão executar um ciclo a mais). Isto pode aumentar a latência das instruções, mas não aumentará o *throughput*. Além disto, haverá um aumento de custo devido à adição de hardware (forward, controle, etc.) para o novo estágio.