UNIVERSIDADE DE SÃO PAULO

SEGUNDO SEMESTRE LETIVO DE 2022

PRIMEIRA PROVA

Escola	EACH	TURMA		Nota do aluno na PROVA
Curso	Sistemas de Informação			
Disciplina	Organização e Arquitetura de Computador II	Data da Prova	25/10/22	
Professor	Clodoaldo Aparecido de Moraes Lima			
Aluno				
No. USP				

1a Questão) (1 ponto)O fragmento de código abaixo processa um array de palavras e retorna 2 valores importantes em $v0 e $v1. Assuma que o array tem armazenado 5.000 palavras indexadas de 0 a 4.999, sabendo que o endereço base do array está armazenado em $a0 e o tamanho do array (5.000) em $a1. Descreva em uma frase o que faz o programa e o que é retornado em $v0 e $v1

[1]		addu $a1, $a1, $a1
[2]		addu $a1, $a1, $a1
[2]		addu $v0, $zero, $zero
[4]		addu $t0, $zero, $zero
[5]	outer:	addu $t4, $a0, $t0
[6]		lw $t4, 0($t4)
[7]		addu $t5, $zero, $zero
[8]		addu $t1, $zero, $zero
[9]	inner:	addu $t3, $a0, $t1
[10]		lw $t3, 0($t3)
[11]		bne $t3, $t4, skip
[12]		addiu $t5, $t5, 1
[13]	skip:	addiu $t1, $t1, 4
[14]		bne $t1, $a1, inner
[15]		slt $t2, $t5, $v0
[16]		bne $t2, $zero, next
[17]		addu $v0, $t5, $zero
[18]		addu $v1, $t4, $zero
[19]	next:	addiu $t0, $t0, 4
[20]		bne $t0, $a1, outer

O código determina qual palavra aparece no array com maior freqüência, retornando a palavra em $v1e o número de vezes que ela aparece em $v0.1

2a Questão) (2.0 Pontos) Escreva um programa em linguagem de montagem (assembly language) do processador MIPS que processa um vetor de números inteiros (VET), transformando cada elemento do vetor em seu valor absoluto. Por exemplo, o valor absoluto do número -43 é +43 e do número +55 é +55. O programa deve obrigatoriamente chamar uma sub-rotina calc_abs que recebe como parâmetro um número e calcula o valor absoluto deste, retornando-o segundo as convenções do MIPS. O valor retornado pela sub-rotina deve ser armazenado, pelo programa principal, na mesma posição do array que continha o elemento original. Utilize a área de dados abaixo para escrever o seu programa.

```
.data
VET: .word 23 -43 55 -9 -7 21 -76 12 -45 -10
TAM: .word 10
```

```
main:
      la $s0,VET
      la $s1,TAM
      lw $s1,0($s1)

loop:
      beq $s1,$zero,fim
      lw $a0,0($s0)
      jal calc_abs
      sw $v0,0($s0)
      addiu $s1,$s1,-1
      addiu $s0,$s0,4
      j loop
fim:
      li $v0,10
      syscall
calc_abs:
      move $v0,$a0
      bgez $v0,fim_ca
      subu $v0,$zero,$v0
fim_ca:
      jr $ra

0,5 carrega vet, tam
0,5 codigo principal
0,5 calc_abs
0,2 uso jal
0,3 finaliza
```

3a Questão) (2 Pontos) Determine o número real de ciclos de clock para executar uma vez o trecho abaixo (do blez $t1,end até uma instrução que salte para trás ou até a última instrução do trecho ser executada, o que acontecer primeiro). Suponha máxima capacidade de resolução de conflitos de dados (isto inclui uma unidade de adiantamento capaz de adiantar dados da saída do terceiro estágio para a entrada do terceiro estágio, da saída do quarto estágio para a entrada do terceiro, e da saída do quarto estágio para a entrada do quarto estágio). Assuma também que está disponível uma estrutura mestre-escravo para acesso ao banco de registradores e um preditor de saltos de 2 bits que inicialmente prevê salto não-realizado. O PC é escrito no quarto ciclo de relógio de cada instrução. Detalhe a execução no diagrama pipeline abaixo, indique todos os adiantamentos de dados que ocorrerem (se ocorrerem) e mostre as bolhas nas posições adequadas, caso estas existam.

BLEZ -- Branches if the register is less than or equal to zero

Os valores iniciais dos registradores pertinentes são: $t0=0x10010000, $t1=0x44 (=68 base 10), $t2=0, $t4=0, $t3=0x100100AA

4a Questão) (2,0 Pontos) Considere a seguinte sequência de instruções, executadas em um datapath com pipeline de 5 estágios:

```
add $5, $2, $1
lw    $3, 4($5)
lw    $2, 0($2)
or    $3, $5, $3
addi  $4,$3,4
```

a) (0,4 Ponto) Na ausência de forwarding ou de detecção de conflito, insira nops para garantir que o código rode corretamente

```
add $5, $2, $1
nops
nops
lw    $3, 4($5)
lw    $2, 0($2)
nops
or    $3, $5, $3
nops
nops
addi   $4,$3,4
```

cada nops 0,08

b) (0,4 ponto) Repita o item anterior usando nops somente quando um conflito não puder ser evitado pela mudança ou rearranjo dessas instruções. Assuma que o registrador $7 é usado para armazenar valores temporários no seu código modificado

O único que dá para mudar é lw $2, 0($2), mas sem ganho

```
add $5, $2, $1
lw    $2, 0($2)
nops
lw    $3, 4($5)
nops
nops
or    $3, $5, $3
nops
nops
addi   $4,$3,4
```

c) 0,3 Se o processador tiver forwarding, mas esquecermos de implementar a unidade de detecção de
conflitos, o que ocorrerá quando este código for executado?

Considerando forwarding, não é preciso bolha

```
add $5, $2, $1
lw    $3, 4($5)
lw    $2, 0($2)
or    $3, $5, $3
addi   $4,$3,4
```

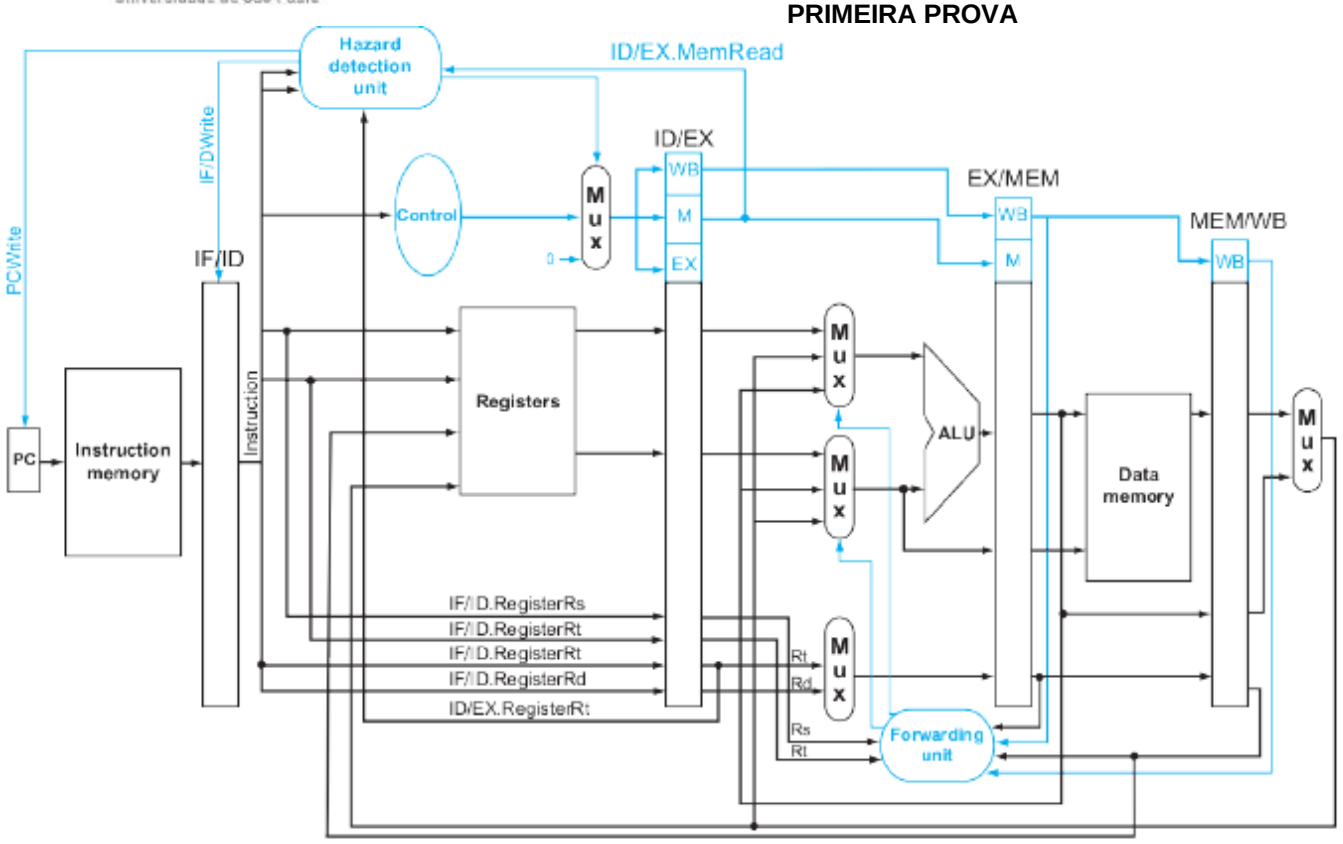
O único problema que poderia ocorrer é se alguém precisasse de $3 logo após (2). Não é o caso, pois o lw em (3) deu o tempo necessário para o forwarding funcionar entre (2) e (4). Então não há problema

d) 0,3 Na presença de forwarding, especifique, para os primeiros 5 ciclos da execução deste código, que sinais são ligados em cada ciclo pelas unidades de detecção de conflitos e de forwarding na figura abaixo:

UNIVERSIDADE DE SÃO PAULO

SEGUNDO SEMESTRE LETIVO DE 2022

PRIMEIRA PROVA



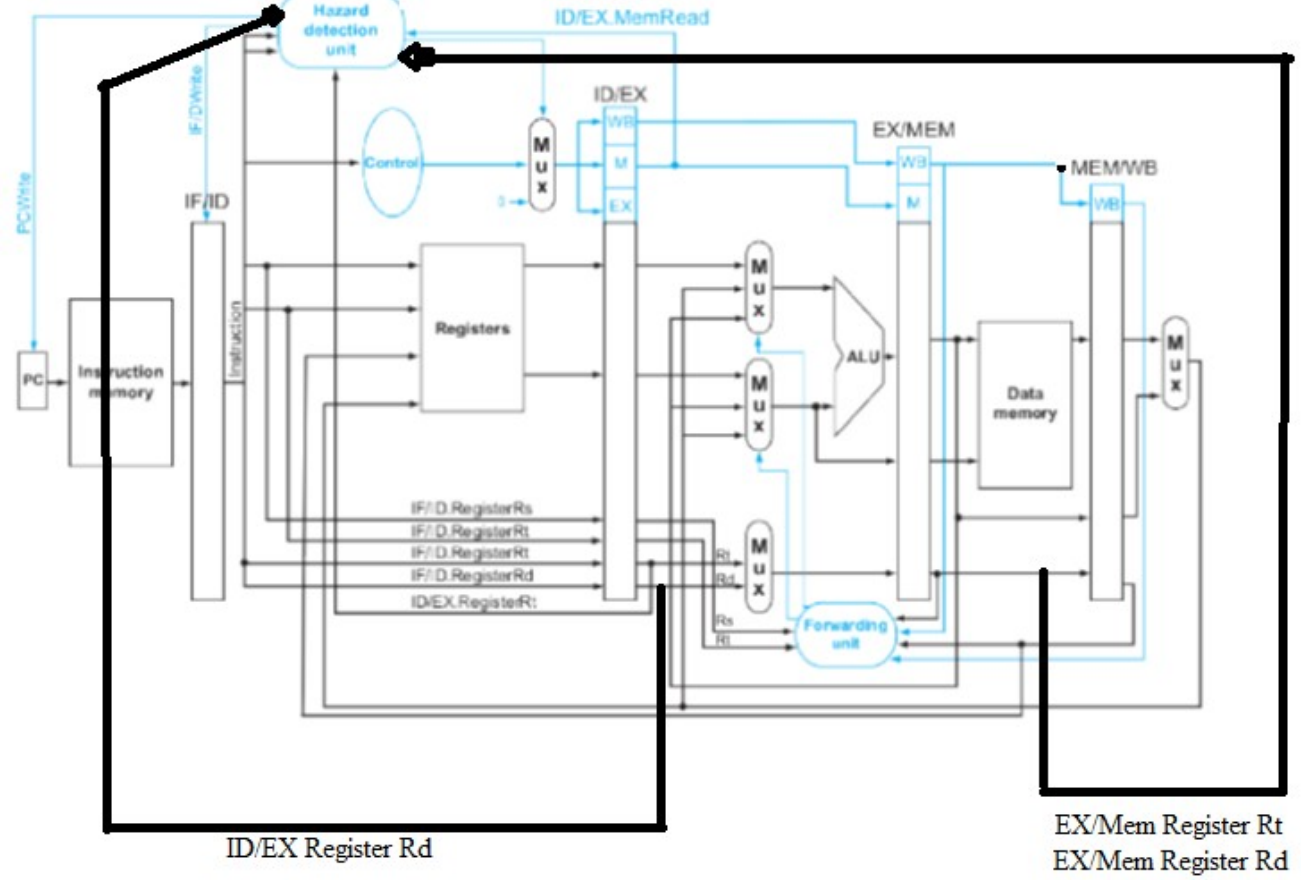
Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

add $5, $2, $1 - no foward no foward
lw $3, 4($5) - forward EX/Mem deslocamento
lw $2, 0($2) - no foward deslocamento
or $3, $5, $3 - no forward Mem/WB
addi $4,$3,4 - EX/Mem deslocamento

Ciclo	PCWrite	IF/IDWrite	ID/EX	ALUin1	ALUin2
1	1	1	0	00 (BR)	00 (BR)
2	1	1	0	10 (EX/Mem)	11 (deslocamento)
3	1	1	0	00 (BR)	11 (deslocamento)
4	1	1	0	00 (BR)	01 Mem/WB
5	1	1	0	10 EX/Mem	11 (deslocamento)

e) 0,3 **Na ausência de forwarding**, que novos sinais de entrada e saída precisaríamos ter na unidade de detecção de conflitos na figura acima? Usando esta sequência de instruções como exemplo, explique porque cada sinal é necessário.

A instrução que está em ID precisa ser parada se depender de um valor fornecido em EX ou MEM. Então precisamos verificar o registrador de destino dessas 2 instruções. Para a instrução em EX, precisamos verifica o Rt da instrução lw e o Rd da instrução formato R. Para instruções em MEM, precisamos verificar o número do registrador de destino (no estágio de EX/MEM). As entradas adicionais a unidade de detecção de conflitos são o registrador Rd de ID/EX e o registrador de saída armazenado em EX/MEM. O registrador Rt em ID/EX já e uma entrada dessa unidade, na figura. Nenhuma saída adicional é necessária, pois podemos parar a pipeline usando os 3 sinais de saída que já estão na figura



(f) 0,3 Para a nova unidade de detecção de conflitos acima, especifique que sinais de saída ela liga em cada um dos 5 primeiros ciclos, durante a execução desse código.

Não tem forwarding

	Ciclo				
	1	2	3	4	5
Add $5,$2, $1	B	DI	EX	M	WB
lw $3,4($5)		BI	DI	X	X
lw $2,0($2)			BI	X	X
or $3,$5,$3					
addi $4,$3,4					

	Sinais		
Ciclo	PCWrite	IF/IDWrite	ID/EX
1	1	1	0
2	1	1	0
3	1	1	0
4	0	0	1
5	0	0	1

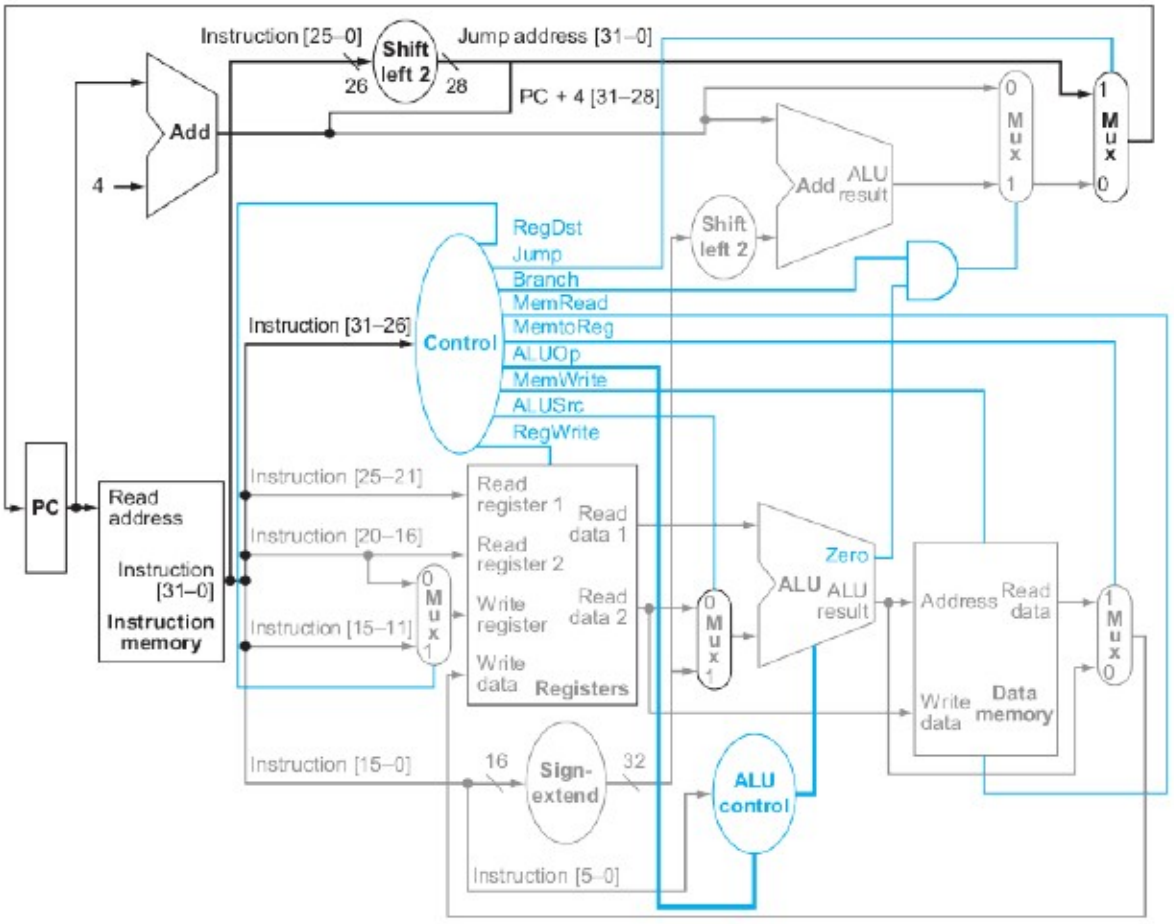
Cada ciclo 0.06
Cada acerto 0,02

5a Questão) (2,0 Pontos) Considere o datapath abaixo:

UNIVERSIDADE DE SÃO PAULO

SEGUNDO SEMESTRE LETIVO DE 2022

PRIMEIRA PROVA



Agora suponha que, em um datapath de ciclo único, a seguinte instrução é trazida da memória: 10101100011000100000000000010100 (trata-se de um \sw rt, desl(rs)"). Assuma que a memória de dados está preenchida com zeros, e que os registradores do processador têm os seguintes valores no início do ciclo no qual a instrução acima é buscada:

r0	r1	r2	r3	r4	r5	r6	r8	r12	r31
0	–1	2	–3	–4	10	6	8	2	–16

a) Quais as saídas da extensão de sinal e da unidade de deslocamento de jumps (\Shift left 2" na figura) para essa instrução?

101011	00011	00010	0000000000010100
opcode	rs	rt	deslocamento

A extensão de sinal usa os bits [15-0] (0000.0000.0001.0100), estendendo o bit de sinal até termos 32b: 0000.0000.0000.0000.0000.0000.0001.0100
A saída da unidade de deslocamento de jumps (shift left 2) é tão somente o deslocamento à esquerda dos bits [25-0]: 0001.1000.1000.0000.0000.0101.0000

b) Quais os valores da entrada da unidade de controle da ALU para essa instrução?
A unidade de controle da ALU recebe os bits [5-0], ou seja, 01.0100, além dos 2 bits de ALUOp (00, conforme tabela na aula 06)

c) Qual o novo endereço do PC após essa instrução ser executada? Mostre (na figura) o caminho que leva à definição desse valor.

Será PC+4. O caminho é PC → PC+4 → Mux do branch → Mux do jump → PC

d) Para cada MUX, mostre os valores de suas saídas durante a execução desta instrução com estes valores de registradores.
Mux dos Registradores 2 se RegDst=1
Mux da ALU 20 (bits [0-15] com extensão de sinal - 3)
Mux da Memória X (Não há como dizer, e não importa)
Mux do Branch PC+4
Mux do Jump PC+4

e) Para a ALU e as duas unidades de soma, quais são os valores de suas entradas de dados?
Somador do PC: PC e 4
Somador do Branch: PC+4 + 80 (bits [15-0] deslocados em 2 esquerda)
ALU: -3 (rs=3) $r3=-3) e 20 (saída do Mux da ALU)

f) Quais os valores de todas as entradas para a unidade de registradores?
RegWrite: 0
Read Register 1: [25-21] = 3
Read Register 2: [20-16] = 2
Write Register: 2 (se RegDst=0) ou 0 ([15-11]) se RegDST=1
Write Data: Não há como dizer a partir dos dados

6a Questão) Limitando nossa atenção a 8 instruções: lw, sw, add, sub, and, or, slt e beq, qual o tempo médio
entre 2 instruções em uma implementação de ciclo único, em que todas as instruções ocupam um
único ciclo de clock, e uma implementação de pipeline, em que cada estágio ocupa um ciclo de clock? O tempo de cada instrução, tanto total quanto a cada estágio, é:

Instruction class	Instruction fetch	Register read	ALU operation	Data access	Register write	Total time
Load word (lw)	200 ps	100 ps	200 ps	200 ps	100 ps	800 ps
Store word (sw)	200 ps	100 ps	200 ps	200 ps		700 ps
R-format (add, sub, AND, OR, slt)	200 ps	100 ps	200 ps		100 ps	600 ps
Branch (beq)	200 ps	100 ps	200 ps			500 ps

Na implementação de ciclo único, o datapath precisa acomodar a instrução mais lenta, e uma só entra no datapath após a outra sair. Então o tempo entre uma instrução e outra _e de 800ps.
No caso da pipeline, e supondo a ausência de problemas, assim que uma instrução deixa a pipeline a seguinte já está entrando no último estágio. Então a diferença de tempo entre 2 instruções é o tempo gasto em cada estágio. Como todo estágio, em nosso modelo, leva o mesmo tempo, eles precisam ser projetados para acomodar o maior estágio. Assim, o tempo entre uma instrução e outra é de 200ps.

Ciclo único –
Tempo entre cada instrução 800 ps
Tempo de cada instrução 800ps
Tempo de cada estagio variável

Pipeline
Tempo entre cada instrução 200 ps
Tempo de cada instrução 200ps
Tempo de cada estágio igual a 200ps