Prova de Arquitetura de Computador

Aluno: Data:

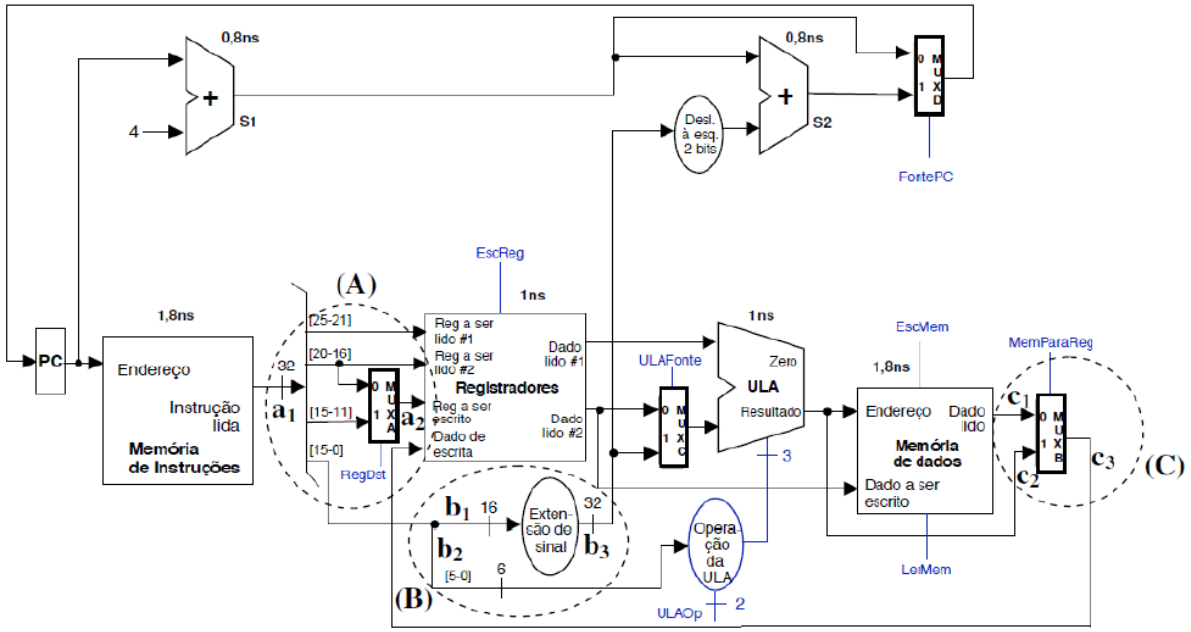
1a Questão) (2,5 ponto) Suponha que a seguinte sequência de instruções deva ser executada na aquitetura abaixo.

Endereço	Instrução	Observações
FA00:0000	lw $s1, 64($t0)	Os registradores $t0-$t7 são numerados de 8 a 15 (temporários) e os de $s0-$s7 de 16 a 23. O opcode de lw é 35. O opcode sub é 0, com 34 nos bits [5-0]. O valor (em hexa) de $t0 = 8AFF:0000. Campos de bits não utilizados/desnecessários para a instrução terão valor 0.
FA00:0004	lw $s0, 60($t0)	
FA00:0008	Sub $s2, $s1, $s0	

Memória

Endereço	Contéudo	Valores em binário em cada posição de memória.
8AFF:0038	0000 0000 0000 0000 0000 0000 0010 0001	
8AFF:003C	0000 0000 0000 0000 0000 0000 0001 1001	
8AFF:0040	0000 0000 0000 0000 0000 0000 0001 0100	
8AFF:0044	0000 0000 0000 0000 0000 0000 0000 1100	

Mostre quais os valores dos conjuntos de bits (a_1 a c_3) que estão sendo enviados em cada uma das partes ao executar cada instrução.



Instrução	Opcode	Rs	Rt	Rd	Shamt	Funct
Posição	31-26	25-21	20-16	15-11	11-6	5-0
lw $s1, 64($t0)	35	8	17	64		
lw $s0, 60($t0)	35	8	16	60		
Sub $s2, $s1, $s0	0	17	16	18	0	34

Solução

Instrução	Opcode	Rs	Rt	Rd	Shamt	Funct
Posição	31-26	25-21	20-16	15-11	11-6	5-0
lw $s1, 64($t0)	100011	01000	10001	0000 0000 0100 0000		
lw $s0, 60($t0)	100011	01000	10000	0000 0000 0011 1100		
Sub $s2, $s1, $s0	000000	10001	10000	10010	00000	100010

0.8

lw $s1, 64($t0)	
a1	100011 01000 10001 0000 0000 0100 0000
a2	10001

b1	0000 0000 0100 0000
b2	000000
b3	0000 0000 0000 0000 0000 0000 0100 0000
c1	0000 0000 0000 0000 0000 0000 0001 0100 (20 em binário)
c2	8 A F F :0 0 4 0 0 0 0 0 :0 0 4 0
c3	0000 0000 0000 0000 0000 0000 0001 0100

c2 - saida da ULA (soma $t0 + deslocamento b3)

$t0	=	8	A	F	F	:0	0	0	0
Des		0	0	0	0	:0	0	4	0
c2		8	A	F	F	:0	0	4	0

0.8

lw $s0, 60($t0)	
a1	100011 01000 10000 0000 0000 0011 1100
a2	10000
b1	0000 0000 0011 1100
b2	11 1100
b3	0000 0000 0000 0000 0000 0000 0011 1100
c1	0000 0000 0000 0000 0000 0000 0001 1001 (25 em binário)
c2	8 A F F :0 0 3 C 0 0 0 0 :0 0 3 C
c3	0000 0000 0000 0000 0000 0000 0001 1001

c2 - saida da ULA (soma $t0 + deslocamento b3)

$t0	=	8	A	F	F	:0	0	0	0
Des		0	0	0	0	:0	0	3	C
c2		8	A	F	F	:0	0	3	C

0.9

Sub $s2, $s1, $s0	
a1	000000 10001 10000 10010 00000 100010
a2	10010
b1	10010 00000 100010
b2	100010
b3	0000 0000 0000 0000 10010 00000 100010
c1	xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx 0,15 (colocou zero 0.05)
c2	1111 1111 1111 1111 1111 1111 1111 1011 0,10
c3	1111 1111 1111 1111 1111 1111 1111 1011 0,15

c2 - saida da ULA ($s1 - $s0)

$s1	=	0000 0000 0000 0000 0000 0000 0001 0100
$s0	=	0000 0000 0000 0000 0000 0000 0001 1001
c2		0000 0000 0000 0000 0000 0000 0000 0101 (5)
		1111 1111 1111 1111 1111 1111 1111 1010 (inverte)
		1111 1111 1111 1111 1111 1111 1111 1011 (soma 1) 0,15

2a Questão) (1,5 ponto) Considere o bloco de dados monociclo apresentado da questão anterior. Suponha que existe uma falha, que afeta apenas o multiplexador mais à direita no desenho (C), que gera a entrada de dados do Banco de Registradores. A falha é que este componente sempre deixa passar a sua entrada c2 para a saída, independente do valor do sinal de controle MemParaReg. Diga quais instruções ainda podem ser executadas corretamente, justificando sua resposta. Alguma instrução pode ser executada de forma correta às vezes e de forma incorreta outras vezes? Se sim, qual instrução é esta e em que condições tais situações ocorrem?

As únicas instruções que podem ser afetadas são as que fazem acesso à memória (1,0 ponto). Mais ainda apenas as que leem dado da memória, pois a escrita não passa pelo multiplexador com falha. Logo, todas as instruções podem ser executadas sem nenhuma falha, exceto a instrução LW (1,0 ponto). A única possibilidade de a falha ser mascarada seria no caso em que o endereço da memória ao qual se quer fazer acesso conter um dado exatamente igual ao endereço onde ele está armazenada, um fato bastante incomum (0,5).

Não acessam memoria (Formato R - 0.25 (funciona), Branch- 0.25 (funciona), Jump - 0.25 (funciona))

Acessam memória (Store word - 0.25 (funciona), Load word - 0.25 (não funciona))

Endereço igual ao valor - 0,25

3a Questão) (1,5 Ponto) Analise o código do programa abaixo e explique o que este programa faz. Inicie mapeando que registradores são utilizados, dizendo como cada um é inicializado e conclua descrevendo a função destes no código do programa. Comente semanticamente o código. Com os dados fornecidos, que valor é escrito na memória de dados ao final da execução do programa?

```
            .text
            .globl      main
main:       la          $t0,VET
            la          $t1,TAM
            lw          $t1,0($t1)
            add         $t2,$zero,$zero
            add         $t3,$zero,$zero
loop:       beq         $t1,$zer0,end
            lw          $t4,0($t0)
            andi        $t4,$t4,1
            beq         $t4,$zero, opcao
            addi        $t3,$t3,1
            j           loopend
opcao:      addi        $t2,$t2,1
loopend:    addi        $t0,$t0,4
            addi        $t1,$t1,-1
            j           loop
end:        la          $t0,R
            sw          $t2,0($t0)
            la          $t0,T
            sw           $t3,0($t0)
            li $v0,10
            syscall
.data
VET: .word 23 -43 55 -9 -7 21 -76 12 -45 -10
TAM: .word 10
R: .word 0
T: .word 0
```

0,35 Processa um vetor de números inteiros (VET), contando quantos elementos pares e ímpares contém o vetor. Os resultados são armazenados nas variáveis R (numeros PARES) e T (números IMPARES) em memória.
R = 3 0,2
T = 7 0,2
0,75 (descrição geral do código - 0,3; descrição andi 0.15, la 0.15, lw 0,15)

4a Questão) (2,0 ponto) Traduza a implementação abaixo para MIPS

Enquanto i<> 10
v[i+1] ← v[i] +1;

```
      se v[i+1] <> 10
        a++;
      fimse;
      i ← i+1
      b ← c*2 -d;
fim enquanto
```

v[ ], a, b, c, d e i estão armazendos nos registrados $s0, $s1, $s2, $s3, $s4 e $s5

```
        add $t1, $zero, 10   #adciona 10 em $t5
Loop:   beq $s5, $t1, FIM    #verifica se i é igual a 10
        sll $t2, $s5, 2 #multiplica i por 4
        add $t2, $t2, $s0 # soma a base
        lw $t3, 0($t2) #carrega v[i]
        addi $t3, $t3, 1 #v[i]+1
        sw $t3, 4($t2)   # v[i+1] = v[i]+1
        beq $t3, $t1, FSE    #IF
        addi $s1, $s1, 1 #a = a+1
FSE:    addi $s5, $s5, 1 # i = i + 1
        add $t6, $s3, $s3
        sub $s2, $t6,$s4
j        Loop
FIM:
```

Esqueceu de multiplicar indice por 4
Cada instrução – 2/14

5a Questão) (2,5 ponto) Assuma que o seguinte código é executado sobre um processador pipeline com 5 estágio, com adiantamento e um preditor de desvio.

```
          add $1  $5, $3
Label1:   sw  $1, 0 ($2)
          add  $2, $2, $3
          beq  $2, $4, Label 1 // Não tomado
          add  $5, $5, $1
          sw $1, 0 ($2)
```

a) (1,0 ponto) Desenhe o diagrama de execução para este código, assumindo que todo desvio é tomado pelo preditor, que não há unidade de adiantamento e que o PC é escrito no terceiro ciclo de relógio de cada instrução em caso de desvio.

Convenções: X – bolha, F - flush do pipeline, -- para estágio não usado, ---> adiantamento ou leitura após escrita no mesmo ciclo. Estágios do pipeline: BI(Busca), DI (Decodificação), EX (Execução) MEM (Memória) WB (Writeback)

Instrução.....	1.	2.	3.	4.	5.	6.	7.	8.	9.	10	11	12	13	14	15	16
add $1 $5, $3	BI	DI	EX	---	W											
sw $1, 0 ($2)		BI	DI	X	X	EX	M	---								
add $2, $2, $3			BI	X	X	DI	EX	---	W							
beq $2, $4, Label 1						BI	DI	X	X	EX	---	---				
add $5, $5, $1							BI	X	X	DI	F	F	F			
sw $1, 0 ($2)										BI	F	F	F	F		
add $5, $5, $1											BI	DI	EX	---	W	
sw $1, 0 ($2)												BI	DI	EX	M	---

cada instrução correta 1,0 / 8
considerou nop - descontar 0.1
erro o numero de bolhas 0.15
cada 2 erro de notação - 0.05
bolha lugar errado - 1 bolha- 0.15, 2 bolha -0.2
adiantamento 0.1
cada duas instruções que não seguiram a convençao - 0.05

b) (1,0 ponto) Desenhe o diagrama de execução para este código, assumindo que um preditor de saltos de 2 bit que inicialmente prevê salto realizado, que há unidade de adiantamento e que o PC é escrito no terceiro ciclo de relógio de cada instrução em caso de desvio.

Instrução.....	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12	13	14	13	14	15
add $1 $5, $3	BI	DI	EX	---	W												
sw $1, 0 ($2)		BI	DI	**X**	**X**	EX	M	---									
add $2, $2, $3			BI	**X**	**X**	DI	EX	---	W								
beq $2, $4, Label 1						BI	DI	EX	---	---							
add $5, $5, $1							BI	DI	F	F	F						
sw $1, 0 ($2)								BI	F	F	F	F					
add $5, $5, $1									BI	DI	EX	---	W				
sw $1, 0 ($2)										BI	DI	EX	M	---			

cada instrução 1,0/8
Não adicionou bolha - 0.1

c) (0,5 ponto) Qual é o speed-up alcançado pelo item b) em relação ao item a).

$$speed\_up = \frac{16}{14} = 1.14$$