Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2016/2
Prof. Marcus Vinicius Lamar

d_0 d_1 / d_2 d_3 d_4 d_5 d_6 d_7 d_8

Nome: GABARITO Matrícula: □□/□□□□□□□

Prova 1

(2.0)1) Dado um processador MIPS com frequência de 100MHz, que necessita 1 ciclo de *clock* para a execução das instruções tipo R, 2 ciclos para as instruções tipo J, 3 ciclos para as instruções tipo I e 4 ciclos para as instruções de acesso à memória de dados (*load/stores*). Considerando que o desempenho seja calculado por

$$\eta = \frac{1}{Mem + t_{exec}}$$

onde *Mem* é a quantidade de memória total usada (dados e programas) dada em kiB (kibibytes) e t_{exec} o tempo de execução dado em *ms* (milissegundos). Qual o desempenho deste processador na execução do programa dado?

```
0x00400000 0x34040064
0x00400004 0xAFA40000
0x00400008 0x00842020
0x0040000C 0x8FA40000
0x00400010 0x2084FFFE
0x00400014 0x1480FFFB
```

(3.0)2) Neste semestre, o professor esqueceu de ministrar o conteúdo relativo à ISA e organização do Coprocessador 0 do processador MIPS, a unidade de Ponto Flutuante. Porém, conhecendo como é realizada a representação de um número real no padrão IEEE 754 em Precisão Simples e respeitando a convenção do uso dos registradores, implemente os procedimentos abaixo usando apenas as instruções da CPU principal do MIPS.

(1.0) a) jal ABS # o procedimento recebe em $a0 um número em float e retorna em $v0 seu módulo

(2.0) b) jal CMPLE # o procedimento recebe em $a0 e $a1 dois números em float e retorna $v0=1 se $a0<=$a1 ou $v0=0 caso contrário

(2.0)3) Implemente as pseudo-instruções abaixo:

(1.0)a) seq $t0,$t1,$t2 # se $t1==$t2 então $t0=1 senão $t0=0

(1.0)b) bnzal $t0,LABEL # se $t0!=0 então $ra=PC+4 e PC=LABEL, senão PC=PC+4

(4.0)4) Em Sistemas Embarcados a relação entre custo (medido aqui como o tamanho da memória) e desempenho (medido aqui como o tempo de execução) é um ponto fundamental e que deve ser considerado durante todo o projeto. A memória RAM é um recurso extremamente caro nesses sistemas. Considere que você esteja trabalhando em uma rotina do sistema operacional que necessite que os 32 registradores do banco de registradores do processador MIPS sejam copiados para a memória RAM de dados, a partir do endereço dado por $a0, quando a chamada jal BACKUP seja executada. Dica: Apenas os registradores $at, $k0 e $k1 não precisam ter os valores corretos necessariamente preservados, isto é, podem ser modificados pela rotina BACKUP.

(2.0)a) Caso o custo não seja uma limitação, escreva um procedimento BACKUP que tenha o melhor desempenho possível.

(2.0)b) Caso o desempenho não seja uma limitação, escreva um procedimento BACKUP que ocupe no máximo 64 bytes de memória de programa.

Boa Sorte!

OAC-A PROVA 1 2016/2

GABARITO

1)

			Ciclos
0x00400000	34040064	ori $a0, $zero, 100	3
	AFA40000	sw $a0, 0($s5)	4
	00842020	add $a0, $a0, $a0	1
	8FA40000	lw $a0, 0($s5)	4
	2084FFFE	addi $a0, $a0, -2	3
	1480FFFB	bne $a0, $zero, -5	3

Loop($a0 = 100; $a0 ≠ 0; $a0 = $a0 - 2)

Repete 50 vezes

N° ciclos = 3 + (4+1+4+3+3) × 50 = 753

memória = $\underbrace{6 \times 4}_{\text{código}} + \underbrace{4}_{\text{dado}}$ = 28 bytes = $27,3438 \times 10^{-3}$ KiB

$t_{exec} = 753 \times \frac{1}{100M} = 7,53 \times 10^{-4}$ s = $7,53 \times 10^{-3}$ ms

$$\eta = \frac{1}{27,3438 \times 10^{-3} + 7,53 \times 10^{-3}} = 28,67$$ //

2)

a) ABS: la $v0, 0x7FFF FFFF
and $v0, $a0, $v0
jr $ra

Apagou apenas o bit de sinal

b) Folha em Anexo

A comparação em FLOAT pode ser feita diretamente, invertendo o resultado caso os números sejam negativos.

3) a) seq $t0, $t1, $t2

```
add $t0, $zero, $zero
bne $t1, $t2, PULA
addi $t0, $zero, 1
PULA:
```

b) bntal $t0, LABEL

```
beq $t0, $zero, PULA
jal LABEL
PULA:
```

4) Folha em Anexo

a) melhor desempenho é a solução TRIVIAL que ocupa mais memória

b) Usa Hi e Lo como registradores temporários

modifica a instrução:

sw $zero, 0($a0)

↳ varia de 4 em 4

↳ varia de 1 em 1

```
# Questão 2

.text
    j B
A:  li $v0,6
    syscall
    mfc1 $a0,$f0    # le um float
    jal ABS
    mtc1 $v0,$f12   # imprime o modulo do float
    li $v0,2
    syscall
#   j A         # para testar com vários valores

B:  li $v0,6
    syscall
    mfc1 $a0,$f0    # le um float $a0
    li $v0,6
    syscall
    mfc1 $a1,$f0    # le um float $a1
    jal CMPLE
    move $a0,$v0   # imprime comparação
    li $v0,1
    syscall
    j B

#############################################################
ABS:    la $v0,0x7FFFFFFF
    and $v0,$a0,$v0
    jr $ra

#############################################################
CMPLE:  li $v0,1
    beq $a0,$a1,FIM     # testa se são iguais
    lui $t1,0x8000      # máscara do sinal
    and $t0,$t1,$a0     # sinal de $a0
    and $t0,$t0,$a1     # sinal de $a1 & sinal de $a0 = se os $a0 e $a1 <0
    srl $t0,$t0,31      # coloca o sinal no bit 0
    slt $v0,$a0,$a1     # compara os valores inteiros de $a0 e $a1
    xor $v0,$v0,$t0     # inverte resultado da comparação caso os 2 sejam negativos
FIM:    jr $ra
#############################################################
```

```
.text
    move $a0,$sp
#   jal BACKUP_A
    jal BACKUP_B
    li $v0,10
    syscall

BACKUP_A: #sw $0,0($a0)   # para ser o mais rapido não precisa salvar $zero, $at, $k0 e $k1
    #sw $1,4($a0)
    sw $2,8($a0)
    sw $3,12($a0)
    sw $4,16($a0)
    sw $5,20($a0)
    sw $6,24($a0)
    sw $7,28($a0)
    sw $8,32($a0)
    sw $9,36($a0)
    sw $10,40($a0)
    sw $11,44($a0)
    sw $12,48($a0)
    sw $13,52($a0)
    sw $14,56($a0)
    sw $15,60($a0)
    sw $16,64($a0)
    sw $17,68($a0)
    sw $18,72($a0)
    sw $19,76($a0)
    sw $20,80($a0)
    sw $21,84($a0)
    sw $22,88($a0)
    sw $23,92($a0)
    sw $24,96($a0)
    sw $25,100($a0)
    #sw $26,104($a0)
    #sw $27,108($a0)
    sw $28,112($a0)
    sw $29,116($a0)
    sw $30,120($a0)
    sw $31,124($a0)
    jr $ra

BACKUP_B: la $k0,LOOP       # Endereço LOOP
                #
      mthi $k0      # salva em HI
      la $k0,0x00010004 # incremento da instrução
                #
      mtlo $k0      # salva em LO
      lui $k0,0xAC80    # instrução sw $zero,0($a0)
      li $k1,32     # contador
LOOP:   sw $zero,0($a0)     # local da instrução variável
    mflo $at        # recupera o incremento
    add $k0,$k0,$at     # calcula a próxima instrução
    mfhi $at        # recupera o endereço LOOP
    sw $k0,0($at)       # atualiza a instrução variável
    addi $k1,$k1,-1     # decrementa o contador
    bne $k1,$zero,LOOP  # verifica
    jr $ra          # retorna
```