Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2017/2
Prof. Marcus Vinicius Lamar

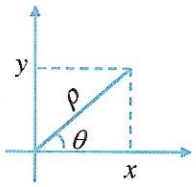
d_0 d_1 / d_2 d_3 d_4 d_5 d_6 d_7 d_8

Nome: GABARITO Matrícula: 17/0123456

Prova 1

(6.0)1) Na simulação de sistemas físicos a conversão entre coordenadas polares e coordenadas retangulares é muito utilizada. Sabendo que:

$$x = \rho.\cos(\theta) \quad y = \rho.\sin(\theta) \quad \sin^2(\theta) + \cos^2(\theta) = 1 \quad \sin(\theta) = \sum_{n=0}^{N\to\infty} \frac{(-1)^n}{(2n+1)!}\theta^{2n+1}$$



(2.0) a) Implemente um procedimento em Assembly MIPS que realize eficientemente a conversão:

(float *x*, float *y*) = P2R (float ρ, float θ); # converte polar em retangular

Considere *N=10*, que os argumentos são passados nos registradores \$f0 (ρ) e \$f1 (θ) e os valores de retorno nos registradores \$f12 (*x*) e \$f13 (*y*).

(1.0) b) Qual deve ser a CPI média do seu procedimento em um processador com uma frequência de clock de 1.5GHz para que seja executado em 150ns com *N=10*?

(1.0) c) Dado o código ao lado, considerando que na sua implementação seja usado $N=2^{31}-1$, quais os valores finais em hexadecimal dos registradores \$f12 e \$f13?

(2.0) d) Considerando que o label P2R seja o endereço 0x00401000 escreva o conteúdo (em hexadecimal) existente na memória nos endereços 0x00400000 a 0x00400028.

Considere .text no endereço 0x004000000 e .data no endereço 0x10010000

```
.data
DAT1: .word 0x41d800000
DAT2: .word 0x3F490FDB
.text
MAIN:   la $t0,DAT2
        lwc1 $f1,0($t0)
        la $t0,DAT1
        lw $s0,0($t0)
        beq $s0,$zero,FIM
        mtc1 $s0,$f0
        jal P2R
FIM:    li $v0,0x0a
        syscall
```

(3.0)2) O montador permite que o programador use diversas pseudo-instruções que tornam o programa em Assembly mais fácil de ser lido. Na ISA MIPS as instruções add e sub detectam implicitamente o overflow chamando a rotina de tratamento de exceções do sistema operacional. Implemente eficientemente as pseudo-instruções abaixo que não devem chamar a rotina de tratamento de exceções:

(1.0) a) nego \$t0,\$t1,OVERFLOW # Negação com detecção explícita de overflow \$t0=-\$t1 e chama jal OVERFLOW (endereço de 32 bits) caso ocorra overflow

(1.0) b) addo \$t0,\$t1,\$t2,OVERFLOW # Soma com detecção explícita de overflow \$t0=\$t1+\$t2 e chama jal OVERFLOW (endereço de 32 bits) caso ocorra overflow

(1.0) c) Escreva uma rotina OVERFLOW que imprima na tela "Ocorreu Overflow no endereço %d", onde %d é o endereço da instrução jal que chamou o procedimento, e termine o programa retornando ao sistema operacional.

(2.0)3) Responda:

(1.0) a) Que medida de desempenho é utilizada atualmente para caracterizar sistemas com grande poder de processamento (supercomputadores)? Por que esta medida é utilizada?

(1.0) b) De acordo com o autor David Patterson, como os modernos sistemas computacionais podem ser classificados? Dê duas características típicas de cada classe.

Que a força esteja com você!

OAC-A PROVA 1 2017/2

GABARITO

1)

a) FOLHA em ANEXO (2,0)

b) Num de Instruções de P2R

$I = 13 + 7 + 10 \times N + 2$

$I = 22 + 10N$ $N = 10 \rightarrow I = 122$

LOGO:

$t_{exec} = I \times CPI \times T$

$150n = 122 \times CPI \times \frac{1}{1,5G}$

$CPI = 1,84$ (1,0)

c) CONSIDERANDO $N \rightarrow \infty$ p/ d8 = 6

$\$f0 = 0x41600000 = 14,0$

$\$f1 = 0x3F490FDB = 0,7853982 = \frac{\pi}{4}$

0011 1111 0100 1001 0000 1111 1101 1011

126 1,10010010000111111011011

7E = -1 1,5707963705

COMO

$sen(\pi/4) = cos(\pi/4) = 0,7071067811 87$

LOGO

$\$f12 = \$f13 = 14 \times \frac{\sqrt{2}}{2} = 7\sqrt{2} = 9,899494 93659$

$E = 3 \rightarrow EXP_{ARITC} = 130$

$M = 9,899 / 8 = 1,237436867 07$

$\$f12 = \$f13 = 0X411E6455$ (0,5)

(0,5)

apcode
00|0011, 100400

d) P2R: 0x0040 1000 >> 2 → 0000 0000 0100 0000 0001 0000 0000 0100

0x0040 0000	lui $at, 0x1001	0x3C01 1001
04	ori $t0, $at, 4	0x34280004
08	lwc1 $f1, 0($t0)	0xC5010000
0C	lui $at, 0x1001	0x3C01 1001
10	ori $t0, $at, 0	0x3428 0000
14	lw $s0, 0($t0)	0x8D10 0000
18	beq $s0, $zero, 2	0x1200 0002
1C	mtc1 $s0, $f0	0x4490 0000
20	jal P2R	0x0C 100400 //
24	addiu $v0, $zero, 10	0x2402 000A
0x0040 0028	syscall	0x0000 000C

(0,2) cada

2)

a) bgeo $t0, $t1, OVERFLOW

↳ Único caso de OVERFLOW $t1 = 0x8000 0000

↳ maior negativo possível não tem a representação positiva.

```
    lui $at, 0x8000          (1,0)
    bne $t1, $at, PULA
    jal OVERFLOW
PULA: subu $t0, $zero, $t1   Nega sem overflow
```

b) addo $t0, $t1, $t2, OVERFLOW

método A + B > 0
(-A) + (-B) < 0 } não tem
A + (-B) } overflow
(-A) + B

Folha em ANEXO (1,0)

3)

a) x FLOPS : Operações em ponto Flutuante por seg. porque essas instruções possuem capacidade/complexidade semelhantes entre si.

(1,5)

b) Pessoais : - Geralmente 1 único usuário
- Software de terceiros

Servidores : - Recursos compartilhados vários usuários
- Software específico

Embarcados : - Fins específicos
- Software de difícil customização

(0,1) cada
+ 0,1

2)

c) .Data
TXT: .asciiz "Ocorreu OVERFLOW no Endereço"
(0,2)

.Text
OVERFLOW: la $a0, TXT (0,2)
li $v0, 4
syscall
addi $a0, $ra, -4 (0,2)
li $v0, 1
syscall (0,2)
li $v0, 10
syscall (0,2)

```
1: .eqv N 10
2:
3: .data
4: DAT1: .word 0x41100000
5: DAT2: .word 0x3F490FDB
6: UM:   .float 1.0
7:
8: .text
9: MAIN:    la $t0,DAT2
10:     lwc1 $f1,0($t0)
11:     la $t0,DAT1
12:     lw $s0,0($t0)
13:     beq $s0,$zero,FIM
14:     mtc1 $s0,$f0
15:     jal P2R
16: FIM:    li $v0,0x0a
17:     syscall
18:
19:
20: P2R:    addi $sp,$sp,-4     # $f0 raio  $f1 angulo
21:     sw $ra,0($sp)
22:     l.s $f4,UM      # numero 1.0
23:                 # pseudo
24:     jal SENO        # resultado em $f9
25:     mul.s $f3,$f9,$f9   # sin^2
26:     sub.s $f3,$f4,$f3   # 1-sin^2
27:     sqrt.s $f3,$f3      # cos=sqrt
28:     mul.s $f12,$f0,$f3  # R*cos
29:     mul.s $f13,$f0,$f9  # R*sin
30:     lw $ra,0($sp)
31:     addi $sp,$sp,4
32:     jr $ra
33:
34: SENO:   li $t0,N    # $f1 é o angulo
35:     mov.s $f20,$f4  # contador em float
36:     mov.s $f9,$f1   # 1o termo do somatorio
37:     mul.s $f7,$f1,$f1   # x^2
38:     neg.s $f7,$f7       # -x^2
39:     mov.s $f5,$f1       # x^(2n+1)
40:     mul.s $f6,$f4,$f4  # 0! fatorial
41: LOOP:   beq $t0,$zero,FIMS
42:     addi $t0,$t0,-1
43:     add.s $f20,$f20,$f4 # contador
44:     mul.s $f6,$f6,$f20
45:     add.s $f20,$f20,$f4 # contador+1
46:     mul.s $f6,$f6,$f20  # fatorial
47:     mul.s $f5,$f5,$f7   # (-1)^n * x^(2n+1)
48:     div.s $f21,$f5,$f6  # x^(2n+1)/(2n+1)!
49:     add.s $f9,$f9,$f21  # resultado do somatório
50:     j LOOP
51: FIMS:   jr $r
```

```
1: # Questão 2
2: .data
3: STR: .asciiz "Ocorreu Overflow no endereco "
4:
5: .text
6:  li $t1,-10
7:  #la $t1,0x80000000
8:
9: #    a) nego $t0,$t1,OVERFLOW
10:     lui $at,0x8000      # o únco caso que pode dar overflow é 100000...0000
11:     bne $t1,$at,PULA
12:     jal OVERFLOW
13: PULA:   subu $t0,$zero,$t1  # nega sem overflow
14:
15:
16:
17:     li $t1,0x7FFFFFF0
18: #   li $t2,20
19:     li $t2,2
20:
21:     j SO
22:
23: #   b) addo $t0,$t1,$t2,OVERFLOW  Solução Trivial
24: ST: addi $sp,$sp,-12    # salva registradores usados
25:     sw $t3,0($sp)
26:     sw $t4,4($sp)
27:     sw $t5,8($sp)
28:     lui $at,0x8000      # máscara do último bit
29:     and $t3,$t1,$at     # sinal de A
30:     and $t4,$t2,$at     # sinal de B
31:     addu $t0,$t1,$t2    # resultado A+B sem sinalizar overflow
32:     and $t5,$t0,$at     # sinal do resultado
33:     bne $t3,$t4,PULAX   # sinais diferentes não tem overflow
34:     beq $t3,$t5,PULAX   # sinal das parcelas igual ao do resultado não tem overflow
35:     jal OVERFLOW
36: PULAX:  lw $t5,8($sp)       # recupera registradores
37:     lw $t4,4($sp)
38:     lw $t3,0($sp)
39:     addi $sp,$sp,12
40:
41:
42: #  b) Solução otimizada
43: SO: addu $t0,$t1,$t2    # calcula a soma sem detecção de overflow
44:     xor $at,$t1,$t2     # xor dos bits mais signifcativos dos operandos
45:     slt $at,$at,$zero   # se xor=1 então sinais diferentes <0
46:     bne $at,$zero, NO_OVER  # se sinais diferentes não há overflow
47:     xor $at,$t0,$t1     # xor dos bits mais sig do resultado e um operando
48:     slt $at,$at,$zero   # se xor=1 então sinais diferentes <0
49:     beq $at,$zero,NO_OVER   # se forem bits iguais não há overflow
50:     jal OVERFLOW
51
```

```
52: NO_OVER: li $v0,10
53:       syscall
54:
55:
56: #    c)  Procedimento OVERFLOW
57: OVERFLOW:    la $a0,STR
58:          li $v0,4
59:          syscall
60:          addi $a0,$ra,-4      # Endereço do jal
61:          li $v0,1
62:          syscall
63:          li $v0,10
64:          syscal
```

: