Ministério da Educação
UNIVERSIDADE DE BRASÍLIA
Instituto de Ciências Exatas
Dep. Ciências da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma B 2007/1
Prof.: Marcus Vinicius Lamar

Nome: _____ Matrícula: _____

Prova 1

```
int sort (int v[], int n)
{
    int i,t,k;

    k=0;  t=1;
    while(t==1)
    {
        t=0;
        for(i=0;i<n-1;i++)
            if(v[i]>v[i+1])
            {
                swap(v,i);
                t=1;
                k++;
            }
    }
    return k;
}
```

1) (5.0) Dado a função em C ao lado, onde swap é o procedimento de troca visto em aula.

a) (2,0) Respeitando a convenção do uso de registradores, compile o procedimento para Assembly MIPS

b) (3,0) Caso v[10]={0, 1, 2, 3, 4, 5, 6, 9, 7, 8}
b.1) (0,5) Qual o valor retornado no registrador $v0 ?
b.2) (1,0) Qual o número de instruções necessário a ordenação deste vetor (considere também as instruções do procedimento swap) ?
b.3) (0,5) Em um processador MIPS uniciclo com *clock* de 1GHz, qual o tempo necessário a esta ordenação?
b.4) (1,0) Para um vetor de dimensão 10, qual o máximo tempo esperado para a ordenação com o processador do item anterior?

2) (2,5) A vida do programador em Assembly MIPS é bastante facilitada pelo montador SPIM, uma vez que o mesmo implementa de maneira automática, várias pseudo-instruções que são bastante úteis. Dado que BIG é uma constante imediata de 32 bits, SMALL uma constante de 16 bits, LABEL um endereço, implemente as seguintes pseudo-instruções:

a) lw $t0, BIG($t1) # $t0=Memory [$t1+BIG]
b) beq $t0, SMALL, LABEL # if ($t0==SMALL) goto LABEL
c) push $t0 # Coloca o valor de $t0 na pilha e atualiza o topo ($sp)
d) bgt $t0, $t1, LABEL # if ($t0>$t1) goto LABEL
e) addi $t0, $t1, BIG # $t0=$t1+BIG

3) (3,5) Para o seguinte trecho de código Assembly:

```
        R add $t0, $zero, $zero     # Zera acumulador
R+I <-  ble $a1, $zero, finish      # se a1<0 termina
        R add $t0, $t0, $a0         # acumula a0
        I subi $a1, $a1, 1          # a1=a1-1 a1 vezes
        J j loop                    # vai pro loop
finish: I addi $t0, $t0, 100        # soma 100 ao produto a0xa1
        R add $v0, $t0, $zero       # coloca no reg. de saída
```

Onde $a0 e $a1, correspondem a valores inteiros e $v0 o valor de saída do procedimento.
Calcula o produto de a0xa1 e soma 100.

a) (0,7) Acrescente comentários ao código (nesta folha) e descreva em uma sentença o que ele calcula.

b) (0,8) Sabendo que temos três implementações do processador MIPS usando diferentes números de ciclos de *clock* para cada tipo de instrução:

Instrução	MP-1 (1GHz)	MP-2 (1,5GHz)	MP-3 (2 GHz)
Tipo-R	1	3	2
Tipo-I	2	1	3
Tipo-J	3	2	1

Considerando $a0=10 e $a1=20:

b.1) (0,5) Quais os tempos que cada processador demora para executar este trecho?
b.2) (0,3) Qual o fator de desempenho conseguido pela 2ª versão do processador em relação à 1ª versão?

c) (2,0) Sabendo que o código acima está localizado na memória no início do segmento de texto, endereço 0x00400000, realize a tradução para linguagem de máquina, escrevendo o conteúdo da memória de programa em hexadecimal.

BOA SORTE!

OAC - TURMA B

1º PROVA 2007-1

GABARITO

1) a)

```
SORT: addi $SP, $SP, -24
      sw $ra, 20($SP)
      sw $s4, 16($SP)
      sw $s3, 12($SP)
      sw $s2, 8($SP)
      sw $s1, 4($SP)
      sw $s0, 0($SP)
      move $s0, $a0
      addi $s1, $a1, 1
      move $s2, $ZERO
      li $s3, 1
WHILE: beq $s3, $ZERO, EXIT
      move $s3, $ZERO
      move $s4, $ZERO
FOR:  slt $t0, $s4, $s1
      beq $t0, $ZERO, WHILE
      sll $t0, $s4, 2
      add $t0, $s0, $t0
      lw $t1, 0($t0)
      lw $t2, 4($t0)
      slt $t0, $t2, $t1
      beq $t0, $ZERO, ENDFOR
      move $a1, $s4
      move $a0, $s0
      jal SWAP
      li $s3, 1
      addi $s2, $s2, 1
ENDFOR: addi $s4, $s4, 1
      j FOR

EXIT: lw $s0, 0($SP)
      lw $s1, 4($SP)
      lw $s2, 8($SP)
      lw $s3, 12($SP)
      lw $s4, 16($SP)
      lw $ra, 20($SP)
      addi $SP, $SP, 24
      move $v0, $s2
      jr $ra

SWAP: sll $t1, $a1, 2
      add $t1, $a0, $t1
      lw $t0, 0($t1)
      lw $t2, 4($t1)
      sw $t2, 0($t1)
      sw $t0, 4($t1)
      jr $ra
```

b)

b.1) $v0 armazena o número de trocas (swap) realizados

b.2) 235 instruções

$$n^{o}\text{ instruções} = 11 + 3 + \underbrace{7\times10}_{7\text{ FOR}} + \underbrace{2\times22}_{2\text{ FOR c/ SWAP}} + 2 + \underbrace{3 + 9\times10 + 2 + 1}_{\text{2º WHILE } (t=0)} + 9$$

(1º WHILE (t=1): $3 + 7\times10 + 2\times22 + 2$; 9 FOR: 9×10)

b.3) $\text{tempo} = 235 \times \frac{1}{1\times10^{9}} = 235\times10^{-9} = 235\,ns$

b.4) Pior caso: 1511 instruções

$$n^{o}\text{ instruções} = 11 + (3+9\times22+2) + (3+8\times22+1\times10+2) + (3+7\times22+2\times10+2) + (3+6\times22+3\times10+2) + (3+5\times22+4\times10+2) + (3+4\times22+5\times10+2) + (3+3\times22+6\times10+2) + (3+2\times22+7\times10+2) + (3+1\times22+8\times10+2) + (3+9\times10+2) + 1 + 9$$

$t_{MAX} = 1511 \times 1\times10^{-9} = 1{,}511\,\mu s$

2)

a)
```
lui $at, BIG[31...16]      # 16 bits mais significativos de BIG
ori $at, $at, BIG[15...0]  # 16 bits menos significativos
add $at, $t1, $at
lw $t0, 0($at)
```

outra solução:
```
lui $at, BIG[31..16]
addi $at, $at, $t1
lw $t0, BIG[15..0]($at)
```

b) addi $at, $ZERO, SMALL
beq $at, $t0, LABEL

c) addi $SP, $SP, -4
sw $t0, 0($SP)

d) SLT $at, $t1, $t0 # t1 < t0
bne $at, $ZERO, LABEL

e) lui $at, BIG[31..16]
ori $at, $at, BIG[15..0]
add $t0, $t1, $at

3) a1=20 → executa 20 vezes o loop

Cuidar: ble $a1, $ZERO, FINISH ⟹ SLT $at, $ZERO, $a1
beq $at, $ZERO, FINISH

Logo:
Instruções = 1 + 20 x 5 + 2 + 2 = 105
Tipo-R = 1 + 20 x (1+1) + 1 + 1 = 43
Tipo I = 20 x (1+1) + 1 + 1 = 42
Tipo-J = 20 x (1) = 20

b.1)
187 ciclos
$t_{MP1} = [1\times43 + 2\times42 + 3\times20] / 1G = 187ns$

211 ciclos
$t_{MP2} = (3\times43 + 1\times42 + 2\times20) / 1{,}5G = 140{,}66ns$

232 ciclos
$t_{MP3} = [2\times43 + 3\times42 + 1\times20) / 2G = 116ns$

b.2) $n = \frac{187}{140{,}606} = 1{,}329$ → 32,9% mais rápido

c)

Instruction	OP	RS	RT	RD	SHAMT	FUNCT	Hex
add $t0, $zero, $zero (↳8)	000000	00000	00000	01000	00000	100000	00 00 40 20

LOOP: ble $a1, $zero, Finish :

↳ { SLT $at, $zero, $a1 — |000000|00000|00101|00001|00000|101010| — 00 05 08 2A

beq $at, $zero, Finish (↳+4) — |000100|00001|00000|0000000000000100| — 1 0 2 0 0 0 0 4

add $t0, $t0, $a0 (rd, rs, rt) — |000000|01000|00001|01000|00000|100000| — 0 1 0 1 4 0 2 0

addi $a1, $a1, -1 — |001000|00101|00101|1111111111111111| — 2 0 A 5 F F F F

j LOOP (0x00400004, ↳2 bits→) — |000010|00 0001 0000 0000 0000 0000 0001| — 0 8 1 0 0 0 0 1

Finish: addi $t0, $t0, 100 — |001000|01000|01000|0000000001100100| — 2 1 0 8 0 0 6 4

add $v0, $t0, $zero — |000000|01000|00000|00010|00000|100000| — 0 1 0 0 1 0 2 0

Address	Instruction
0x0040 0000	00 00 40 20
0x0040 0004	00 05 08 2A
0x0040 0008	10 20 00 04
0x0040 000C	01 04 40 20
0x0040 0010	20 A5 FF FF
0x0040 0014	08 10 00 01
0x0040 0018	21 08 00 64
0x0040 001C	01 00 10 20