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

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

Nome:_____ Matrícula: _____

Prova 1

1) (4.0) Dada a Série de Fibonacci(n)={1,1,2,3,5,8,13,21,34,55,...}, e respeitando a convenção do uso dos registradores:

a) (2.0) Escreva um procedimento **não-recursivo** em Assembly MIPS que a calcule o enésimo termo desta série.
b) (1.0) Escreva o programa principal que leia do teclado o valor inteiro n e escreva "Fibonacci(%d)=%d\n" na tela do console SPIM.
c) (1.0) Escreva a equação que define o número de instruções necessárias ao cálculo de Fibonacci(n) com base em n.

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) sw $t0, BIG($t1)          # Memory [ $t1+BIG ] = $t0
b) bne $t0, SMALL, LABEL     # if ($t0!=SMALL) goto LABEL
c) pop $t0                   # Retira valor da pilha e coloca em $t0, e atualiza o topo ($sp)
d) ble $t0, $t1, LABEL       # if ($t0<=$t1) goto LABEL
e) subi $t0, $t1, BIG        # $t0=$t1-BIG
```

3) (4,5) O seguinte fragmento de código processa dois arrays e produz um valor importante no registrador $v0. Considere que cada array consista de 2500 inteiros, indexados de 0 a 2499, que os endereços base dos arrays estejam em $a0 e $a1, e seus tamanhos (2500) estejam em $a2 e $a3.

```
R  sll $a2, $a2, 2          # desloca $a2 2 BITS à esquerda = X4
R  sll $a3, $a3, 2          # "   $a3  "  "  "  "  "
R  add $v0, $zero, $zero    # zera $v0
R  add $t0, $zero, $zero    # zera $t0
outer: R  add $t4, $a0, $t0 # $t4 = $a0 + $t0  calcula endereço
   I  lw $t4, 0($t4)        # carrega conteúdo $t4
   R  add $t1, $zero, $zero # zera $t1
inner: R  $t3, $a1, $t1 add # $t3 = $a1 + $t1  calcula endereço
   I  lw $t3, 0($t3)        # carrega conteúdo $t3
   I  bne $t3, $t4, skip    # verifica se $t3 ≠ $t4
   I  addi $v0, $v0, 1      # incrementa contador se for igual
skip: I  addi $t1, $t1, 4   # incrementa $t1 endereço
   I  bne $t1, $a3, inner   # verifica se atingiu limite
   I  addi $t0, $t0, 4      # incrementa $t0 endereço
   I  bne $t0, $a2, outer   # verifica se atingiu limite
```

a) (1,5) Acrescente comentários ao código (nesta folha) e diga o que será retornado em $v0. nº de elementos iguais

b) (1,0) 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

A = {0, 1, 2, ..., 2499}
B = {1, 2, 3, ..., 2500}

b.1) (0,9) Quais os tempos que cada processador demora para executar este trecho?
b.2) (0,1) 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 A

1ª Prova 2007-1

Gabarito

1) a)

```
Fibonacci: Li $t1, 1
           Li $t2, 1
           SLTi $t0, $a0, 2
           bne $t0, $ZERO, FORA
           addi $a0, $a0, -1
LOOP:      add $t3, $t1, $t2
           MOVE $t1, $t2
           MOVE $t2, $t3
           addi $a0, $a0, -1
           bne $a0, $ZERO, LOOP
FORA:      move $v0, $t2
           jr $ra
```

b)

```
.data
MES1: .asciiz "Fibonacci("
MES2: .asciiz ") ="
newl: .asciiz "\n"

.TEXT
.globl __START
__START:
           Li $v0, 5      # le n do teclado
           SYSCALL        inteiro
           MOVE $s0, $v0  # salva n em $s0
```

```
Li $V0,4          # imprime 1: string
La $a0, MES1
SYSCALL

Li $V0, 1         # imprime n
MOVE $a0,$S0
SYSCALL

Li $V0, 4         # imprime 2: string
La $a0, MES2
SYSCALL

MOVE $a0,$S0
jal Fibonacci
MOVE $a0, $V0

Li $V0, 1         # imprime Fib(n)
SYSCALL

Li $V0,4          # pula linha
La $a0, NEWL
SYSCALL

Li $V0, 10        # EXIT
SYSCALL
```

c) Nº instruções = $\begin{cases} 5 + (n-1) \times 5 + 2 = 5n + 2 & \text{if } n > 2 \\ n = 0,\ n = 1 \quad \text{Nº instr} = 6 \end{cases}$

2) a) lui $at, BIG[31..16]
addi $at, $at, $X1
sw $t0, BIG[15..0]($at)

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

c) lw $t0, 0($SP)
addi $SP, $SP, 4

d) SLT $at, $X1, $t0
beq $at, $ZERO, LABEL

e) lui $at, BIG[31..16]
ori $at, $at, BIG[15..0]
sub $t0, $X1, $at

3)
TIPO-R = 4 + 2500(2+2500) = 6.255.004

TIPO-I = 0 + 2500 (1+2500 (2+2) + 2499 x 1,1 + 2)

N°GUAU = 2499 TIPO-I = 31.255.000

TIPO-J = 0

b.1) $t_{MP1} = (1 \times 6.255.004 + 2 \times 31.255.000 + 3 \times 0) / 16$
$t_{MP-1} = 0,068765 = 68,765 ms$

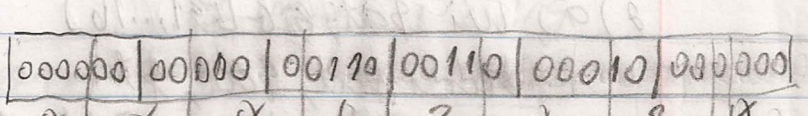
$t_{MP-2} = (3 \times 6.255.004 + 1 \times 31.255.000 + 2 \times 0) / 1,5 G$
$t_{MP-2} = 0,033346 = 33,34 ms$

$t_{MP-3} = (2 \times 6.255.004 + 3 \times 31.255.000 + 1 \times 0) / 2G$
$t_{MP-2} = 0,0531375 = 53,13 ms$

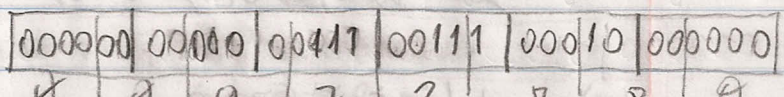
b.2) $\eta = \frac{68,765}{33,34} = 2,0621$

c) (Rt) Rd

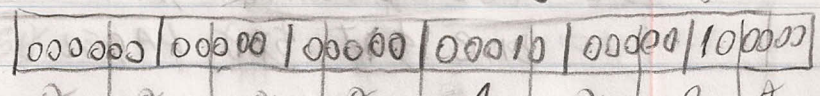
sll $a2, $a2, 2 (6, 6)



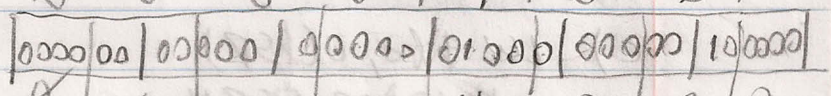
sll $a3, $a3, 2 (7, 7)



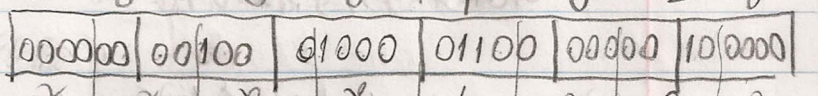
add $v0, $zero, $zero (2)



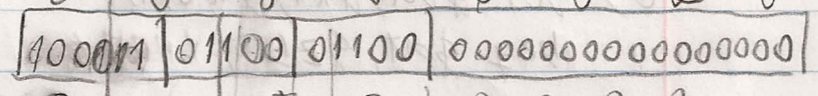
add $t0, $zero, $zero (8)



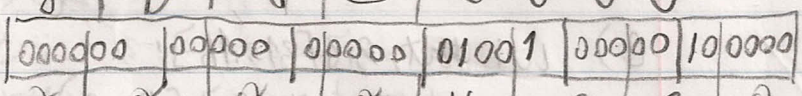
outer: add $t4, $a0, $t0 (12, 4, 8)



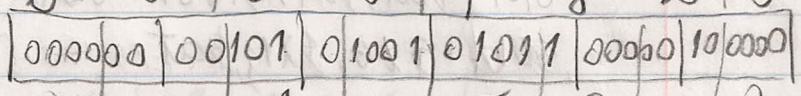
lw $t4, 0($t4) (23h, 12)



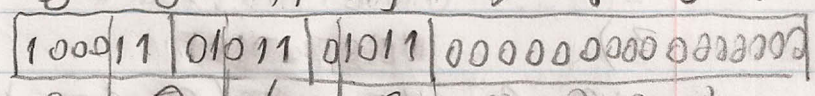
add $t1, $zero, $zero (11, 5, 9)



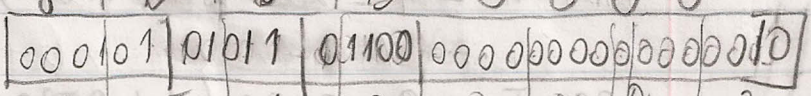
inner: add $t3, $a1, $t1 (11)



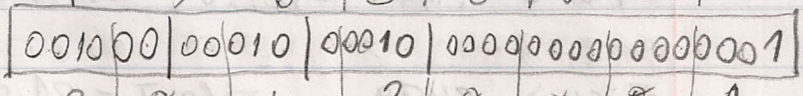
- lw $t3, 0($t3) (11)



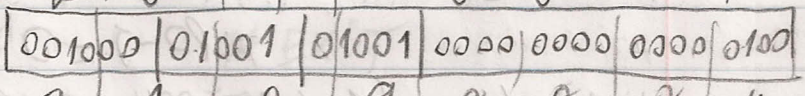
- bne $t3, $t4, skip (11, 12) 4+2



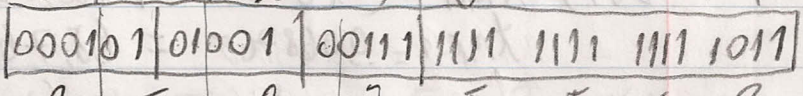
- addi $v0, $v0, 1 (2)



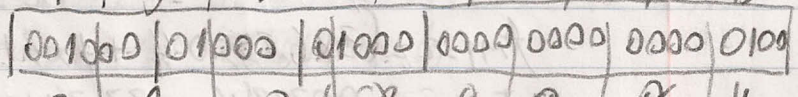
skip - addi $t1, $t1, 4 (9)



- bne $t1, $a3, inner (9, 7) 4-5



- addi $t0, $t0, 4 (8)



- bne $t0, $a2, over (8, 6) 4-10