Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2012/2
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

(4.0) 1) Respeitando a convenção do uso dos registradores, compile e monte, i.e. codifique em linguagem de máquina (hexadecimal), a partir do endereço .text 0x00400000 o programa ao lado. Coloque a resposta final no verso desta folha.

Obs.: Não se preocupe com a codificação do segmento de dados.

```
void main(void)
{
   int k;

   printf("k=");
   scanf("%d",&k);
   printf("r=%d\n",proc(2*k,k));
}

int proc(int a, int b)
{
     if(a<=b)  return (a+b);
     else  return (a+proc(a-b,a+b)-b);
}
```

(2.0) 2) O montador Mars implementa várias pseudo-instruções Assembly que são muito úteis aos programadores. Às vezes, algumas instruções reais não são implementadas propositalmente, devendo ser emuladas como as pseudo-instruções.

Escreva um código alternativo para as instruções abaixo:

(1.0)a) lb $t0,Imm($t1) # Carrega em $t0 o byte presente em (Imm+$t1) considerando-o em complemento de 2

(0.5)b) lhu $t0,Imm($t1) # Carrega em $t0 a half-word presente em (Imm+$t1) considerando-a sem sinal

(0.5)c) lui $t0,Imm # Coloca o Imm na porção mais significativa de $t0

(1.5) 2) No estudo de desempenho de sistemas computacionais, a sua definição pode ser extremamente genérica. Neste curso usamos apenas o tempo de execução do programa do usuário para medir o desempenho. No entanto, vc trabalha em uma indústria que fabrica tablets de alto desempenho, e neste caso a dimensão (D em polegadas), o peso do produto (P em gramas) e a capacidade da bateria (C em mA/h) são tão importantes quanto o tempo de execução (t em segundos). Estes fatores podem possuir relações bastante complexas entre si D(P,C,t), P(D,C,t), C(D,P,t) e t(P,D,C). Para os seus superiores, o ideal é ter um tablet de 7" com tela sensível ao toque capacitiva de AMOLED que tenha o menor peso possível, uma bateria que dure o máximo possível e execute o programa do usuário o mais rápido possível. Usando esses fatores crie uma função para a medir o desempenho.

(1.5) 3) Sobre as arquiteturas Harvard e Von Neumann:
(0.5)a) O que caracteriza estas arquiteturas?
(0.5)b) Quais as vantagens e desvantagens de cada uma.
(0.5)c) Qual é usada hoje em dia nos modernos processadores?

(1.5) 4) Suponha que melhoramos uma máquina fazendo todas as instruções de ponto flutuante serem executadas cinco vezes mais rápido. Estamos procurando um benchmark para mostrar essa nova unidade de ponto flutuante e queremos que o benchmark geral mostre um aumento de velocidade de 3 vezes. Um benchmark que estamos considerando é executado durante 100 segundos com o hardware de ponto flutuante antigo. Quanto do tempo de execução as instruções de ponto flutuante teriam que considerar para produzir nosso aumento de velocidade desejado nesse benchmark?

BOA SORTE!

Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores
Prof. Marcus Vinicius Lamar

Nome: Gabarito Matrícula: _____

Endereço (Hexa)	Conteúdo(Hexa)
0x0040 0000	0x2402 0004
0x0040 0004	0x3C01 1001
0x0040 0008	0x3424 0000
0x0040 000C	0x0000 000C
0x0040 0010	0x2402 0005
0x0040 0014	0x0000 000C
0x0040 0018	0x0002 1021
0x0040 001C	0x0010 2040
0x0040 0020	0x0010 2821
0x0040 0024	0x0C10 0014
0x0040 0028	0x0002 3821
0x0040 002C	0x2402 0004
0x0040 0030	0x3C01 1001
0x0040 0034	0x3424 0003
0x0040 0038	0x0000 000C
0x0040 003C	0x2402 0001
0x0040 0040	0x0011 2021
0x0040 0044	0x0000 000C
0x0040 0048	0x2402 000A
0x0040 004C	0x0000 000C
0x0040 0050	0x23BD FFF4
0x0040 0054	0xAFBF 0008
0x0040 0058	0xAFA5 0004
0x0040 005C	0xAFA4 0000
0x0040 0060	0x0085 402A
0x0040 0064	0x1500 000C
0x0040 0068	0x1085 000B
0x0040 006C	0x0004 4021
0x0040 0070	0x0085 2022
0x0040 0074	0x0105 2820
0x0040 0078	0x0C10 0014
0x0040 007C	0x8FBF 0008
0x0040 0080	0x8FA5 0004
0x0040 0084	0x8FA4 0000
0x0040 0088	0x23BD 000C
0x0040 008C	0x0044 1020
0x0040 0090	0x0045 1022
0x0040 0094	0x03E0 0008
0x0040 0098	0x0085 1020
0x0040 009C	0x23BD 000C
0x0040 00A0	0x03E0 0008
0x0040 00A4	
0x0040 00A8	
0x0040 00AC	
0x0040 00B0	
0x0040 00B4	
0x0040 00B8	
0x0040 00BC	
0x0040 00C0	
0x0040 00C4	
0x0040 00C8	

OAC-A 1º Prova 20212

Gabarito

1)

```
.data
STR1: .asciiz "K="
STR2: .asciiz "r="
NL: .asciiz "\n"
.TEXT
MAIN: Li $V0,4          # imprime STR1   addiu $2,$0,4
      la $a0,STR1       lui $1,0x1001 / ori $4,$1,0
      SYSCALL
      Li $V0,5          # le K   addiu $2,$0,5
      SYSCALL
      move $s0,$V0      # Salva em $0   addu $16,$0,$2
      SLL $a0,$s0,1     # $a0=2xK
      move $a1,$s0      # $a1=K   addu $5,$0,$16
      jal PROC
      move $s1,$V0      # addu $17,$0,$2
      Li $V0,4          # imprime resulj   addiu $2,$0,4
      La $a0,STR2       lui $1,0x1001 / ori $4,$1,3
      SYSCALL
      Li $V0,1          # addiu $2,$0,1
      move $a0,$s1      # addu $4,$0,$17
      SYSCALL
      Li $V0,10         # exit   addiu $2,$0,10
      SYSCALL
```

0,25 | 0,25 | 0,5

```
PROC: addi $SP, $SP, -12      # salva na pilha
      sw $ra, 8($SP)          } 0,25
      sw $a1, 4($SP)
      sw $a0, 0($SP)
      slt $t0, $a0, $a1       } 0,5
      bne $t0, $ZERO, PULA    # a0<a1?
      beq $a0, $a1, PULA      # a0 = a1?
      move $t0, $a0           } 0,5
      sub $a0, $a0, $a1
      add $a1, $t0, $a1
      jal PROC                # $a0 = $a0 - $a1 e
                                $a1 = $a0 + $a1
      lw $ra, 8($SP)          } 0,25
      lw $a1, 4($SP)          # Retira pilha
      lw $a0, 0($SP)          # PROC + $a0 - $a1
      addi $SP, $SP, 12
      add $v0, $v0, $a0
      sub $v0, $v0, $a1
      jr $ra

PULA: add $v0, $a0, $a1       # retorna $a0 + $a1   } 0,5
      addi $SP, $SP, 12       # libera pilha
      jr $ra
```

Assembler → 1,0

2) a) `Lb $t0, Imm($t1)`

LITTLE endian	BIG endian
`LW $t0, Imm($t1)`	`LW $t0, Imm($t1)`
`SLL $t0, $t0, 24`	`SRA $t0, $t0, 24`
`SRA $t0, $t0, 24`	

b) LHU $t0, Imm($t1)

LITTLE ENDIAN	BIG ENDIAN
LW $t0, Imm($t1)	LW $t0, Imm($t1)
ANDI $t0, $t0, 0xFFFF	SRL $t0, $t0, 16

c) LUI $t0, Imm

addi $t0, $ZERO, Imm
SLL $t0, $t0, 16

2)

$$d = \frac{C(D,P,t)}{(D(P,C,x)-7)^2 \times P(D,C,t) \times t(P,D,C)}$$

desempenho ótimo

$d \to \infty$ {
- $C \to \infty$ (Bateria Dure ∞)
- $D \to 7$ (Display de 7")
- $P \to 0$ (Pese 0 gramas)
- $t \to 0$ (Executo o Programa em 0 seg)

3)

a) HARVARD: Programas e Dados em memórias diferentes

VON NEUMANN: Prog e Dados na mesma memória

b)

	VANTAGEM	DESVANTAGEM
HARVARD	Acesso + rápido à Largura de Banda	- Acesso proibido à memória - 2 dispositivos físicos
VON NEUMANN	- Prog. tratado como dado	- Largura de Banda - Executa dado

c) É usada a Arquitetura HARVARD modificada, implementada através da Hierarquia de memórias cache:

4) $\begin{cases} 100 = t_{não\ afetado} + t_{FPU\ antiga} \\ \frac{100}{3} = t_{não\ afetado} + t_{FPU\ nova} \end{cases}$

$t_{FPU\ nova} = \frac{t_{antiga}}{5}$

$\frac{100}{3} = t_{não\ afetado} + \frac{t_{FPU\ antiga}}{5}$

$100 - \frac{100}{3} = t_{FPU\ antiga} - \frac{t_{FPU\ antiga}}{5}$

$\frac{200}{3} = \frac{4}{5} t_{FPU\ antiga}$

$t_{FPU\ antiga} = \frac{1000}{12} = 83{,}333$

$t_{não\ afetado} = 16{,}6666$

Logo 83,33% do tempo //