

**CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA**

Arquitetura de Computadores I – Turmas 4 e 5 (EARTE) – 2020/2

Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br

Primeira Prova – 16 de março de 2021

NOME:	GABARITO
MATRÍCULA:	2650719

Importante: Para esta prova considere que o seu número de matrícula na UFES pode ser representado pelo formato 20*****ZYX₁₀, , sendo Z, Y e X inteiros decimais no intervalo [0..9].

No meu caso: X=9, Y=1, Z=7

1ª Questão – Foi realizado um *dump* dos segmentos de texto (*.text*) e dados (*.data*) da memória do simulador MARS, programado com a linguagem de montagem MIPS em 32 bits, conforme a referência do livro texto da disciplina. O *dump* de ambos os segmentos está apresentado a seguir e foi realizado antes da execução do programa, imediatamente após a sua carga em memória:

.text		
Dump da memória em formato <u>hexadecimal</u> nos endereços de 0x00400000 à 0x00400058.		
Endereço (Hex)	Valor (Hex)	Comentário
00400000	24020004	# addiu \$2, \$0, 0x00000004 (li \$v0, 4)*
00400004	3c011001	# lui \$1, 0x00001001 (la \$a0, msg)*
00400008	34240024	# ori \$4, \$1, 0x00000024
0040000c	0000000c	# syscall
00400010	24020005	# addiu \$2, \$0, 0x00000005 (li \$v0, 5)*
00400014	0000000c	# syscall
00400018	00025821	# Linha 0
0040001c	3c011001	# Linha 1
00400020	34280004	# Linha 2
00400024	3c011001	# Linha 3
00400028	34290014	# Linha 4
0040002c	3c011001	# Linha 5
00400030	8c2a0000	# Linha 6
00400034	11400007	# Linha 7
00400038	8d0c0000	# Linha 8
0040003c	016c6006	# Linha 9
00400040	ad2c0000	# Linha 10
00400044	214affff	# Linha 11
00400048	21080004	# Linha 12
0040004c	21290004	# Linha 13
00400050	0810000d	# Linha 14
00400054	2402000a	# addiu \$2, \$0, 0x0000000a (li \$v0, 10)*
00400058	0000000c	# syscall

* O comentário entre parênteses representa uma instrução sintética (pseudoinstrução) equivalente!

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

.data

 Dump da memória em formato hexadecimal nos endereços de 0x10010000 à 0x10010060.

Endereço (Hex)	Valor	Comentário
10010000	00000004	
10010004	00002000	
10010008	00001000	
1001000c	00000800	
10010010	00000400	
10010014	00000000	
10010018	00000000	
1001001c	00000000	
10010020	00000000	
10010024	72746e45	msg: .asciiz "Entre com o ultimo digito do seu numero de matricula na Ufes: " (a variável msg ocupa os endereços de 0x10010024 a 0x10010062)
10010028	6f632065	
1001002c	206f206d	
10010030	69746c75	
10010034	64206f6d	
10010038	74696769	
1001003c	6f64206f	
10010040	75657320	
10010044	6d756e20	
10010048	206f7265	
1001004c	6d206564	
10010050	69727461	
10010054	616c7563	
10010058	20616e20	
1001005c	73656655	
10010060	0000203a	

Considerando que essas são as informações que estão disponíveis, responda as questões a seguir:

```

.bdata
dim:      .word 4
var:      .word 8192, 4096, 2048, 1024
result:   .space 16
msg:      .asciiz "Entre com o ultimo digito do seu numero de matricula na Ufes: "

.text
li $v0, 4          # system call code for print_str
la $a0, msg        # address of string to print
syscall

```

**CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA**

```

li $v0, 5          # system call code for read_int, result in $t3
syscall

addu $t1, $0, $2      # Linha 0 – move $t3, $v0
lui $t1, 0x00001001    # Linha 1 – la $t0, var
ori $t1, $1, 0x00000004    # Linha 2
lui $t1, 0x00001001    # Linha 3 – la $t1, result
ori $t1, $1, 0x00000014    # Linha 4
lui $t1, 0x00001001    # Linha 5 – lw $t2, dim
lw $t2, 0x00000000($t1)    # Linha 6
loop:
    beq $t2, $0, 0x00000007    # Linha 7 – beq $t2, $zero, endloop
    lw $t3, 0x00000000($t1)    # Linha 8 – lw $t4, 0($t0)
    sllv $t4, $t2, $t1        # Linha 9 – sllv $t4, $t4, $t3
    sw $t4, 0x00000000($t1)    # Linha 10 – sw $t4, 0($t1)
    addi $t2, $t2, -1         # Linha 11 – addi $t2, $t2, -1
    addi $t1, $t1, 4           # Linha 12 – addi $t0, $t0, 4
    addi $t1, $t1, 4           # Linha 13 – addi $t1, $t1, 4
    j 0x00400034            # Linha 14 – j loop
endloop:

li $v0, 10
syscall

```

- a) (2,0) Explique o processo de decodificação das instruções presentes nas linhas “Linha X” e “Linha Y+5” presentes no segmento de texto (.text) fornecido nesta questão.
- Importante: X, Y vêm do seu número de matrícula.
 - Restrição: Se X for igual a Y+5 (por exemplo: X=5 e Y=0) então decodifique as linhas “Linha X” e “Linha Y+4”.

Consultar Livro, Página 1-50, Figura A.10.2: “MIPS opcode map”

Linha 6 (Y+5):

Valor: $0x8c2a0000 = (1000\ 1100\ 0010\ 1010\ 0000\ 0000\ 0000\ 0000)_2$

- op (MSB 6 bits): $(100011)_2 = 0x23 = (35)_{10} = \text{lw}$
- rt: $(01010)_2 = (10)_{10} = \$10 = \$t2$
- rs: $(00001)_2 = (01)_{10} = \$1 = \$at$
- address: $(0000000000000000)_2 = 0x0000 = (0)_{10} = 0$
- $\text{lw } \$t2, 0x00000000(\$at) \quad \# \$t2 = \text{MEM}[0x00000000+\$at]$

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Linha 9 (X):

Valor: $0x016c6006 = (0000\ 0001\ 0110\ 1100\ 0110\ 0000\ 0000\ 0110)_2$
 – op (MSB 6 bits): $(000000)_2 = 0x00 = (0)_{10}$ = Arithmetic and Logical Operation
 – function (LSB 6 bits): $(000110)_2 = 0x06 = (6)_{10}$ = srlv
 – rs: $(01011)_2 = (11)_{10} = \$t3$
 – rt: $(01100)_2 = (12)_{10} = \$t4$
 – rs: $(01100)_2 = (12)_{10} = \$t4$
 $\rightarrow \text{srlv } \$t4, \text{St4}, \$t3 \quad \# \$t4 = \$t4 >> \$t3$

Linha 7 (Z):

Valor: $0x11400007 = (0001\ 0001\ 0100\ 0000\ 0000\ 0000\ 0000\ 0111)_2$
 – op (MSB 6 bits): $(000100)_2 = 0x04 = (4)_{10}$ = beq
 – rs: $(01010)_2 = (10)_{10} = \$t2$
 – rt: $(00000)_2 = (00)_{10} = \$zero$
 – offset: $(0000\ 0000\ 0000\ 0111)_2 = 0x0007 = (7)_{10}$
 $\rightarrow \text{beq } \$t2, \$zero, 0x0007 \quad \# \text{if } (\$t2 == \$zero) \text{ PC} = \text{PC} + 28 \text{ (saltar 7 instruções)}$

Linha 14:

Valor: $0x0810000d = (0000\ 1000\ 0001\ 0000\ 0000\ 0000\ 0000\ 1101)_2$
 – op (MSB 6 bits): $(000010)_2 = 0x02 = (2)_{10}$ = j
 – target: $(00\ 0001\ 0000\ 0000\ 0000\ 0000\ 1101)_2 \rightarrow (0000\ 0100\ 0000\ 0000\ 0000\ 0011\ 0100)_2$
 $\rightarrow j\ 0x00400034 \quad \# \text{PC} = 0x00400034$

b) (1,0) Identifique o tipo e os valores iniciais das variáveis armazenadas nos endereços de 0x10010000 a 0x10010020 do segmento de dados (.data).

dim: .word 4 # Endereço 0x10010000
 \rightarrow Vide Linha 5, 6 a instrução lw

var: .word 8192, 4096, 2048, 1024 # Endereços 0x10010004, 0x10010008
0x1001000c, 0x10010010
 \rightarrow Vetor com 4 word: var[0], var[1], var[2], var[3]
 \rightarrow var = 0x10010004 (endereço base do vetor)

result: .space 16 # Endereços 0x10010014, 0x10010018
0x1001001c, 0x10010020
 \rightarrow Vetor com 4 word: result[0], result[1], result[2], result[3]
 \rightarrow result = 0x10010014

*** outra resposta válida ***

result: .word 0, 0, 0, 0 # Endereços 0x10010014, 0x10010018
0x1001001c, 0x10010020
 \rightarrow Vetor com 4 word: result[0], result[1], result[2], result[3]
 \rightarrow result = 0x10010014

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

c) (1,0) Execute o programa armazenado até o final no simulador MARS. Considerando o último dígito do seu número de matrícula como entrada, após essa execução apresente os valores contidos no segmento de dados (.data) da memória nos endereços de 0x10010000 a 0x10010020.

No meu caso, X=9.

O programa desloca o valor em cada posição “i” do vetor “var” em 9 bits à direita (srlv), e armazena o resultado na posição “i” do vetor “result”. Ambos, “var” e “result” estão no segmento de dados da memória.

Endereço (Hex)	Valor	Comentário
10010000	00000004	
10010004	00002000	
10010008	00001000	
1001000c	00000800	
10010010	00000400	
10010014	00000010	$0x00002000 / 2^9 = 8192/512 = 16 = 0x00000010$
10010018	00000008	$0x00001000 / 2^9 = 4096/512 = 8 = 0x00000008$
1001001c	00000004	$0x00000800 / 2^9 = 2048/512 = 4 = 0x00000004$
10010020	00000002	$0x00000400 / 2^9 = 1024/512 = 2 = 0x00000002$

d) (1,0) Considere que a instrução contida no endereço 0x0040003c foi alterada para 0x016c6004. Execute o programa armazenado até o final no simulador MARS. Considerando o último dígito do seu número de matrícula como entrada, após essa execução apresente os valores contidos no segmento de dados (.data) da memória nos endereços de 0x10010000 a 0x10010020.

Valor: 0x016c6004 = (0000 0001 0110 1100 0110 0000 0000 0100)₂

- op (MSB 6 bits): $(000000)_2 = 0x00 = (0)_{10}$ = Arithmetic and Logical Operation

- function (LSB 6 bits): $(000110)_2 = 0x04 = (6)_{10} = \text{slvv}$

$$-\text{rs: } (01011)_2 = (11)_{10} = \$t3$$

$$- rt: (01100)_2 = (12)_{10} = \$t4$$

- rs: $(01100)_2 = (12)_{10} = \$t4$

→ sllv \$t4, St4, \$t3 # \$t4 = \$t4<<\$t3

Ou seja, agora deslocaremos à esquerda.

**CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA**

Endereço (Hex)	Valor	Comentário
10010000	00000004	
10010004	00002000	
10010008	00001000	
1001000c	00000800	
10010010	00000400	
10010014	00400000	$0x00002000 * 2^9 = 8192 * 512 = 4194304 = 0x00400000$
10010018	00200000	$0x00001000 * 2^9 = 4096 * 512 = 2097152 = 0x00200000$
1001001c	00080000	$0x00000800 * 2^9 = 2048 * 512 = 1048576 = 0x00100000$
10010020	00000002	$0x00000400 * 2^9 = 1024 * 512 = 524288 = 0x00080000$

e) (1,0) Considere que a instrução contida no endereço 0x0040003c foi alterada para 0x016c6004. Haverá *overflow* após a execução do programa se a entrada (valor lido em \$v0 no endereço 0x00400014) for igual a 24_{10} ? Justifique sua resposta.

Não, pois a instrução `sllv` não gera *overflow*.

2ª Questão – Considere o programa “questao2.c”, em linguagem C, conforme código a seguir:

```

// Início da declaração de variáveis (seção .data)
int dim=2;           //declara uma variável chamada dim, inteiro 32 bits, inicializada com valor 2
float a[] = {0.ZYX, -1.YZX}; // declara um vetor de float (PF single) chamado a, com 2 posições.
                            // O vetor a é inicializado com 2 números reais 0.ZYX e -1.YZX
float b[] = {-2.XZY, 3.XYZ}; // declara um vetor de float (PF single) chamado b, com 2 posições.
                            // O vetor b é inicializado com 2 números reais -2.XZY e 3.XYZ
float r[] = {0, 0};        // declara um vetor de float (PF single) chamado r, com 2 posições.
                            // O vetor r é inicializado com valores 0.0 em duas 2 posições

void main(void) {
    for (int i=0; i<dim; i++) { // Início do programa (seção .text)
        // Declara um loop que começa com valor i=0 (i é inteiro, 32 bits) e
        // repete enquanto i<dim. i é incrementado a cada iteração
        r[i] = a[i] + b[i];      // Soma os valores nas posições i (i=0 e i=1) dos vetores a e b (ou
                                // seja a[i]+b[i]) e armazena o resultado na mesma posição i do vetor
                                // r (ou seja, r[i] = a[i] + b[i])
    }
}

```

a) (1,0) Utilizando a linguagem de montagem do MIPS, tendo como alvo o simulador MARS de 32 bits, apresente um código em linguagem de montagem (“questao2.asm”) que represente o programa “questao2.c”.

Dica 1: nesta questão você deve apenas traduzir o programa de C para ASM, não é preciso gerar código de máquina (binário ou hexa).

Dica 2: se o programa em C original não realiza operações de entrada de dados (`scanf`) e saída de dados (`printf`), seu programa na linguagem de montagem MIPS também não precisará fazer isso! Não faça o que não foi solicitado!

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

```
.data
dim:    .word 2
a:      .float 0.719, -1.197
b:      .float -2.971, 3.917
r:      .space 8

.text
lw $t0, dim          # Linha 0
la $t1, a            # Linha 1
la $t2, b            # Linha 2
la $t3, r            # Linha 3

loop:
beq $t0, $zero, endloop      # Linha 4
l.s $f0, 0($t1) # $f0 = MEM[a + 0]  # Linha 5
l.s $f1, 0($t2) # $f0 = MEM[b + 0]  # Linha 6
add.s $f0, $f0, $f1        # Linha 7
s.s $f0, 0($t3) # MEM[r + 0] = $f1  # Linha 8

addi $t0, $t0, -1          # Linha 9
addi $t1, $t1, 4            # Linha 10
addi $t2, $t2, 4            # Linha 11
addi $t3, $t3, 4            # Linha 12
j loop                      # Linha 13
endloop:

li $v0, 10
syscall
```

b) (1,5) Altere os valores de X, Y e Z nas variáveis a e b do programa “questao2.asm” da letra a) de acordo com os valores correspondentes do seu número de matrícula (ver início da prova). Execute o seu programa “questao2.asm” no simulador MARS. Apresente um *dump* do conteúdo dos endereços de memória, no segmento de dados (.data) que representam a variável r. Explique os valores encontrados nesses endereços e como interpretar esses valores no formato IEEE 754.

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Endereço (Hex)	Valor	Comentário
10010014	c01020c4	r[0] = 0.719 + (-2.971) em notação IEEE 754
10010018	402e147b	r[1] = -1.197 + 3.917 em notação IEEE 754

Endereço: 0x10010014

Valor: 0xc01020c4 = (1100 0000 0001 0000 0010 0000 1100 0100)₂

- sinal (MSB 1 bit): 1 (negativo)
- exponente (8 bits): (10000000)₂ = 128 = 127+1 → expoente = 1
- mantissa (23 bits): (00100000010000011000100)₂ = 1056964

Endereço: 0x10010018

Valor: 0x402e147b = (0100 0000 0010 1110 0001 0100 0111 1011)₂

- sinal (MSB 1 bit): 0 (positivo)
- exponente (8 bits): (10000000)₂ = 128 = 127+1 → expoente = 1
- mantissa (23 bits): (01011100001010001111011)₂ = 3019899

c) (1.5) Numere, começando em 0, todas linhas de código do segmento de texto (.text) do seu programa “questão2.asm” da letra a) desta questão. Identifique a Linha Z correspondente ao seu número de matrícula (ver início da prova) do segmento de texto (.text) do seu programa e explique como essa linha deverá ser montada em linguagem de máquina (instruções básicas do MIPS) para carga em memória. Apresente o resultado no formato binário ou hexadecimal.

Linha #7

add.s \$f0, \$f0, \$f1

- op (MSB 6 bits): 17 = 0x11 = (010001)₂
- op2 (5 bits): 16 = 0x10 = (10000)₂
- ft (5 bits): \$f1 = (0)₁₀ = (00000)₂
- fs (5 bits): \$f0 = (0)₁₀ = (00000)₂
- fd (5 bits): \$f0 = (1)₁₀ = (00001)₂
- extra (6 bits): (0)₁₀ = 0x00 = (000000)₂
- (0100 0110 0000 0001 0000 0000 0000)₂ = 0x46010000 = add.s \$f0, \$f0, \$f1

*** outro exemplo ***

Linha #11

addi \$t2, \$t2, 4

- op (MSB 6 bits): 8 = (001000)₂
- rs (5 bits): \$t2 = (10)₁₀ = (01010)₂
- rt (5 bits): \$t2 = (10)₁₀ = (01010)₂
- imm (16 bits): (4)₁₀ = (0000 0000 0000 0100)₂
- (0010 0001 0100 1010 0000 0000 0000 0100)₂ = 0x214a0004 = addi \$t2, \$t2, 4

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Em alguns casos a Linha Z representa uma pseudoinstrução, neste caso, ela precisará ser dividida em 2 ou mais instruções sintéticas.

Linha #5

A pseudoinstrução l.s \$f0, 0(\$t1) é montada como:

lui \$1, 0x00000000

addu \$1, \$1, \$9

lwc1 \$f0, 0x00000000(\$1)

lui \$1, 0x00000000

– op (MSB 6 bits): $(15)_{10} = (001111)_2$

– op2 (5 bits): $(0)_{10} = 0x00 = (00000)_2$

– rt (5 bits): $$1 = (1)_{10} = (00001)_2$

– imm (16 bits): $(0)_{10} = 0x0000 = (0000 0000 0000 0000)_2$

→ lui \$1, 0x00000000 = $(0011\ 1100\ 0000\ 0001\ 0000\ 0000\ 0000\ 0000)_2 = 0x3c010000$

addu \$1, \$1, \$9

– op (MSB 6 bits): $(0)_{10} = (000000)_2$

– rs (5 bits): $(1)_{10} = (00001)_2$

– rt (5 bits): $(9)_{10} = (01001)_2$

– rd (5 bits): $(1)_{10} = (00001)_2$

– pad (5 bits): $(0)_{10} = 0x00 = (00000)_2$

– inst (6 bits): $(33)_{10} = 0x21 = (100001)_2$

→ addu \$1, \$1, \$9 = $(0000\ 0000\ 0010\ 1001\ 0000\ 1000\ 0010\ 0001)_2 = 0x00290821$

lwc1 \$f0, 0x00000000(\$1)

– op (MSB 6 bits): $(49)_{10} = (110001)_2$

– rs (5 bits): $(1)_{10} = (00001)_2$

– rt (5 bits): $(0)_{10} = (00000)_2$

– offset (16 bits): $(0)_{10} = (0000\ 0000\ 0000\ 0000)_2$

→ addu \$1, \$1, \$9 = $(1100\ 0100\ 0010\ 0000\ 0000\ 0000\ 0000\ 0000)_2 = 0xc4200000$

Boa Prova!