

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Arquitetura de Computadores I – Turmas 1 e 2 (EARTE) – 2021/2
Prof. Rodolfo da Silva Villaça – rodolfo.villaca@ufes.br
Primeira Prova – 16 de dezembro de 2021

NOME:	DIONATAS SANTOS BRITO
MATRÍCULA:	2019202307

Importante: Para esta prova considere que o seu número de matrícula na UFES pode ser representado pelo formato $20^{*****}ZYX_{10}$, sendo Z, Y e X inteiros decimais no intervalo [0..9].

Z=3

Y=0

X=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 0x00400040.		
Endereço (Hex)	Valor (Hex)	Comentário
00400000	3c011001	# lui \$1, 0x00001001 – la \$a0, numbers*
00400004	34300008	# ori \$16, \$1, 0x00000008
00400008	3c011001	# lui \$1, 0x00001001 – lw \$1, count
0040000c	8c310000	# lw \$17, 0x00000000(\$1)
00400010	00005020	# Linha 0
00400014	0151082a	# Linha 1
00400018	10200005	# Linha 2
0040001c	214a0001	# Linha 3
00400020	8e080000	# Linha 4
00400024	01284820	# Linha 5
00400028	22100004	# Linha 6
0040002c	1000fff9	# Linha 7
00400030	3c011001	# Linha 8
00400034	ac290004	# Linha 9
00400038	2402000a	# Linha 10
0040003c	0000000c	# syscall



CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

* O comentário representa uma instrução sintética (pseudoinstrução) equivalente

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

.data	
count: .word XY res: 0	#Alterar para XY da matrícula
numbers: .word	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203, 205, 207, 209, 211, 213, 215

**CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA**

Considerando que essas são as únicas informações que estão disponíveis, responda:

.data

count: .word 35

res: 0

numbers: .word 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29,
31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63,
65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97,
99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119, 121, 123, 125,
127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147, 149, 151, 153,
155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175, 177, 179, 181,
183, 185, 187, 189, 191, 193, 1

lui \$1, 0x00001001 – la \$a0, numbers*

ori \$16, \$1, 0x00000008

lui \$1, 0x00001001 – lw \$1, count

lw \$17, 0x00000000(\$1)

add \$10 \$0 \$0 # Linha 0

slt \$1 \$10 \$17 # Linha 1

loop; # loop com 5 instruções

beq \$1 \$0 0x0005 # Linha 2 beq \$t0, endloop

addi \$10 \$10 0x0001 # Linha 3

lw \$8 0x0000 \$16 # Linha 4

add \$9 \$9 \$8 # Linha 5

addi \$16 \$16 0x0004 # Linha 6

beq \$0 \$0 0xFFF9 # Linha 7

lui \$1 0x1001 # Linha 8 li \$t1, 0x10010004

sw \$9 0x0004 \$1 # Linha 9

endloop

addiu \$2 \$0 0x000A # Linha 10 li \$v0, 10

syscall



Universidade Federal
do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

a) (1,0) Explique como ocorreu o processo de montagem da instrução **lw \$1, count**

Resposta:

Foi substituída por “lui” e “ori”, a constante excede o valor de 16 bits, então o lui irá colocar os 16 bits mais significativos e o ori, os 16 bits menos significativos, passando assim a constante para os registradores de 32 bits.

Edit

Execute

Text Segment

Program Arguments:

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	14: la \$a0, numbers
<input type="checkbox"/>	0x00400004	0x34240008	ori \$4,\$1,0x00000008	
<input type="checkbox"/>	0x00400008	0x3c011001	lui \$1,0x00001001	15: lw \$1, count
<input type="checkbox"/>	0x0040000c	0x8c210000	lw \$1,0x00000000(\$1)	
<input type="checkbox"/>	0x00400010	0x8c310000	lw \$17,0x00000000(\$1)	16: lw \$17, 0x00000000(\$1)
<input type="checkbox"/>	0x00400014	0x00005020	add \$10,\$0,\$0	18: add \$t2 \$zero \$zero
<input type="checkbox"/>	0x00400018	0x0151082a	slt \$1,\$10,\$17	19: slt \$at \$t2 \$s1
<input type="checkbox"/>	0x0040001c	0x10200005	beq \$1,\$0,0x00000005	21: beq \$at,\$zero, jump
<input type="checkbox"/>	0x00400020	0x214a0001	addi \$10,\$10,0x0000...	22: addi \$t2 \$t2 0x0001
<input type="checkbox"/>	0x00400024	0x8e080000	lw \$8,0x00000000(\$16)	23: lw \$t0 0x0000(\$s0)

Como visto na imagem, o lui carrega o 0x00001001 e o ori carrega o 0x00000000 e em seguida é atribuído ao registrador \$1 (\$at).

Ainda relacionando a tabela, temos que o lw se dividiu em duas instruções nativas e corresponde a 0x3c011001 e , seu processo de montagem ocorreu da seguinte forma:

*0x3c011001 = 001111 00000 00001 0001000000000001-> lui \$at 0x1001
op =15 e o rt = at*

001111	00000	0001	0001000000000001
op	rb	rt	n [imediato]

**CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA**

b) (1,0) Decodifique as instruções presentes nas linhas “Linha X” e “Linha Y+1” presentes no segmento de texto (.text) fornecido nesta questão.

– Restrição: Se X for igual a Y+1 (por exemplo: X=5 e Y=4) então decodifique as linhas “Linha X” e “Linha Y+2”.

x = 7 -> beq\$0 \$0 0xFFF9

#Linha 7 = 0001000000000000111111111111001

000100	00000	00000	111111111111001
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>16 bit n[imediato]</i>

y = 0 + 1 = 1 -> slt \$1 \$10 \$17

Linha 1=00000001010100010000100000101010

000000	01010	10001	00001	00000	101010
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>rd</i>	<i>0</i>	<i>slt</i>

c) (1,0) Execute o programa armazenado até o final no simulador MARS. Após essa execução apresente o valor contido no endereço da variável res e identifique esse endereço. O que faz esse programa?



Universidade Federal
do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Simulador MARS interface showing assembly code and data segment.

Text Segment

Bkpt	Address	Code	Basic	
	0x00400000	0x3c011001	lui \$1,4097	14:
	0x00400004	0x34300008	ori \$16,\$1,8	
	0x00400008	0x3c011001	lui \$1,4097	15:
	0x0040000c	0x8c310000	lw \$17,0(\$1)	
	0x00400010	0x00005020	add \$10,\$0,\$0	17:
	0x00400014	0x0151082a	slt \$1,\$10,\$17	19:
	0x00400018	0x10200005	beq \$1,\$0,5	21:
	0x0040001c	0x214a0001	addi \$10,\$10,1	22:
	0x00400020	0x8e080000	lw \$8,0(\$16)	23:
	0x00400024	0x01284820	add \$9,\$9,\$8	24:

Labels

Label	Address
numbers	0x10010008
res	0x10010004
count	0x10010000

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value
0x10010000	70	4900	1	3	
0x10010020	13	15	17	19	
0x10010040	29	31	33	35	
0x10010060	45	47	49	51	
0x10010080	61	63	65	67	
0x100100a0	77	79	81	83	
0x100100c0	93	95	97	99	
0x100100e0	109	111	113	115	
0x10010100	125	127	129	131	
0x10010120	141	143	145	147	

Navigation buttons: Previous, Next, Address: 0x10010000 (.data), Hexadecimal Addresses

Com $X = 7$ e $Y = 0$ [$XY=70$]

O valor contido no endereço é igual a 4900

O programa tem um comportamento como um loop for, e a quantidade de vezes que vai rodar é o `count`, que vai todas as variáveis que estão presentes na "lista" até "`count`"

d) (1,0) Considere que a instrução contida no endereço 0x0040002c foi alterada para 0x08100005 Execute o programa armazenado até o final no simulador MARS. Após essa execução apresente o valor contido no endereço da variável `res`. Houve alteração? Explique a diferença entre as duas instruções (a original no item c e a modificada nesta questão).

A instrução no endereço 0x0040002c é `beq $zero $zero exit` e equivale a um jump, que compara os valores em `$0` e `$0`, e caso seus valores sejam iguais, o programa recebe o comando de "exit", como o registrador está sendo comparado com ele mesmo, ele sempre ira realizar o jump e caso o endereço seja alterado para 0x08100005 iria mudar a

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

instrução e a nova seria igual a j 0x0100005, que por não realizar comparações entre registradores igual a anterior, e assim tornaria mais eficiente

e) (1,0) Ignore o XY da variável count nesta questão. Qual seria o maior valor de count que não geraria overflow na execução deste programa? Justifique sua resposta.
Não mu

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 variavel chamada dim, inteiro 32 bits, inicializada com valor 2
int a[] = {X, -Y}; // declara um vetor de inteiros, chamado a, com 2 posições.
// O vetor a é inicializado com 2 números inteiros X e -Y
int b[] = {-Z, X}; // declara um vetor de inteiros, chamado b, com 2 posições.
// O vetor b é inicializado com 2 números reais -Z e X
int r1[] = {0, 0}; // declara um vetor de inteiros, chamado r1, com 2 posições.
// O vetor r1 é inicializado com valores 0 nas 2 posições
double r2[] = {0, 0} // declara um vetor de float (PF duplo) chamado r2, com 2 posições.
// O vetor r2 é inicializado com valores 0 nas 2 posições
void main(void) { // Início do programa (seção .text)
    for (int i=0; i<dim; i++) { // 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
        r1[i] = a[i] % b[i]; // Calcula o resto da divisão dos 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 r1 (ou seja, r1[i] = a[i]
% b[i])
        r2[i] = (double) a[i] / (double) b[i]; // Calcula a divisão real dos 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 r2 (ou seja, r2[i] = a[i] / b[i])
    }
}
```

Altere os valores de X, Y e Z nas variáveis a e b do programa “questao2.asm” de acordo com os valores correspondentes do seu número de matrícula.

a) (1,5) 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”.

**CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA**

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!

b) (1,5) 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 as variáveis r1 e r2. Quais são esses endereços? Explique os valores encontrados nesses endereços e como interpretar esses valores no formato IEEE 754.

3ª Questão – Identifique a instrução em linguagem de montagem do MIPS e apresente as representações binárias e hexadecimal das instruções descritas pelos seguintes campos:

a) (1,0) op=0x0, rs=0x3, rt=0x2, rd=0x3, shamt=0x0, funct=0x34

Em Binário = 00000000011000100001100000110100

Em hexa = 00621834

<i>000000</i>	<i>00011</i>	<i>00010</i>	<i>0001100000</i>	<i>110100</i>
<i>SPECIAL</i>	<i>\$v1</i>	<i>\$v0</i>	<i>code</i>	
<i>op</i>	<i>rs</i>	<i>rt</i>	<i>shamt</i>	<i>funct</i>

Equivalente a Instrução = teq \$v1 \$v0

b) (1,0) op=0x23, rs=0x1, rt=0x2, const=0x4

Em Binário = 10001100001000100000000000000100

Em hexa = 8c220004 (hex)

<i>100011</i>	<i>00001</i>	<i>00010</i>	<i>0000000000000100</i>
<i>LW</i>	<i>\$at</i>	<i>\$v0</i>	<i>offset</i>
<i>lw</i>	<i>rb</i>	<i>rt</i>	<i>offset</i>

Equivalente a Instrução = lw \$v0, 4(\$at)



Universidade Federal
do Espírito Santo

CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA

Boa Prova!