Universidade Federal da Fronteira Sul
Curso de Ciência da Computação
Organização de Computadores

1. Considere o seguinte trecho de programa em linguagem Assembly do RISC-V:

```
            .data       0x10010000    # segmento de dados
palavra1:   .word       13
palavra2:   .word       0x15
```

Indique, em hexadecimal, quais os valores dos seguintes rótulos:

palavra1:
palavra2:

2. Considere o seguinte programa em Assembly do RISC-V:

```
            .data       0x10010000    # segmento de dados
var1:       .word       30
var2:       .word       -50
txt:        .string     "Organizacao de Computadores"

            .text       0x400000
main:
      la   s0, var1
      lw   t3, 0(s0)
      la   s1, var2
      lw   t4, 0(s1)
      sw   t3, 0(s1)
      sw   t4, 0(s0)
      and t1, t1, zero
      and t0, t0, zero
      beq t3, zero, haz
      ori t0, zero, 1
haz:
      slt t1, t4, t3
wrt:  li   t0, 111
      la   t2, txt
      sb   t0, 0(t2)
      li   t0, 99
      sb   t0, 15(t2)
```

a) Indique, em hexadecimal, quais os valores dos seguintes rótulos sabendo que a pseudo-instrução `'la'` utiliza 2 instruções de máquina:

```
var1:
var2:
txt:
main:
haz:
wrt:
```

b) Qual o valor armazenado nas posições de memória indicadas abaixo ao final da execução do programa:

```
      var1:
      var2:
      txt:
```

c) Qual o valor que será atribuído ao campo `'Imm'` da instrução `'beq'` em função do rótulo `'haz'` utilizado na instrução?

3. Considere o seguinte programa em Assembly do RISC-V:

```
            .data       0x10010000    # segmento de dados
byte1:      .byte       30
var1:       .space      2
value1:     .word       10
byte2:      .byte       12
value2:     .word       20
```

a) Quantos bytes foram reservados para o rótulo `'var1'`?
b) O que acontece se o processador executar a instrução `'lw t0, 0(s0)'`, considerando que `s0` possui o endereço de `'value1'`? Qual o motivo? Como resolver essa situação?
c) A situação presente no item b também acontece ao executar a instrução `'lw t1, 0(s1)'`, considerando que `s1` possui o endereço de `'value2'`?

4) Descreva o que acontece com o fluxo de execução do programa e com os registradores durante a chamada das instruções CALL e RET em um programa usando o Assembly do RISC-V.

5) Complete o código abaixo. O código é a parte inicial de um programa com Assembly do RISC-V que percorre o vetor de bytes `'vetor'` e conta o número de bytes cujo valor é igual a 1 e armazena o resultado da contagem na variável `'conta'`. (Pode assumir que a dimensão do vetor sala é sempre 64 bytes).

```
            .data       # segmento de dados
vetor:      .space      64
conta:      .word       0

            .text
main:
```

6. Considere que já se encontra implementado a função `randnum`, a qual devolve um inteiro aleatório entre 0 e um dado número limite:

- argumento: número máximo aleatório pretendido (em a0)
- resultados: inteiro aleatório entre 0 e argumento a0 menos 1

Escreva uma função chamada `Joga`, usando o Assembly do RISC-V, que recebe como argumento um inteiro em a0 e chama a função `randnum` com o argumento 10. Caso o resultado da chamada de `randnum` seja igual ao argumento em a0, a função `Joga` imprime no console a string “Acertou!”. Caso contrário, imprime “Errou!”.

```
randnum:
      addi sp, sp, -4
      sw   ra, 0(sp)
```

```
mv a1, a0
li a7, 42
ecall
lw   ra, 0(sp)
addi sp, sp, 4
ret
```

7. Escreva uma função chamada `swap` utilizando o Assembly do RISC-V. A função recebe em a0 o endereço inicial de um vetor de inteiros, em a1 um índice do vetor e em a2 outro índice do vetor. A função deve trocar de posição os valores presentes em cada um dos índices informados para a função no vetor.

8. Escreva uma função chamada `ordena` utilizando o assembly do RISC-V. A função recebe em a0 o endereço inicial de um vetor de inteiros, em a1 o tamanho do vetor. A função deve ordenar o vetor em ordem crescente de valores. A função `ordena` deve obrigatoriamente utilizar a função swap desenvolvida no exercício anterior.

9. Escreva uma função utilizando o Assembly do RISC-V que remove todas as vogais presentes em uma string. A função recebe em a0 o endereço inicial da string e ao final da sua execução devolve em a0 o endereço inicial da string sem as vogais. Dicas: a) utilize a chamada de sistema 9 para fazer a alocação de memória para nova string sem vogais; b) utilize as instruções lb (load byte) e sb (store byte) para manipular a string na memória; c) o caracter '\0' marca o final da string.

10. O ISA do RISC-V possui vários formatos de instrução (tipo R, I, B, S, J, etc). Para cada uma das instruções codificadas em hexadecimal mostradas abaixo, apresente a instrução em assembly e classifique a instrução quanto ao formato (R, I, B, etc), classe de instrução (aritmética, lógica, desvio condicional, etc) e o modo de endereçamento (registrador, imediato, relativo ao PC, etc).

Código	Instrução	Formato	Classe	Modo de endereçamento
0x00C50513	addi x10 x10 12	I	Aritméticas	Imediato
0x10000517				
0x00A004B3				
0x0122A023				
0x400904B3				
0x0240006F				
0x0005a503				

11. Implementar um programa que lê dois valores inteiros. Após ler os valores o programa deve chamar uma função chamada `multiplica_1` que, realiza a multiplicação dos valores usando somas sucessivas e retorna o resultado. Após o retorno da função o programa deve

imprimir resultado no programa principal. Após isso o programa deve chamar a função `multiplica_2`, que realiza a multiplicação usando deslocamento e soma.

Pseudo-código:

```
Programa:
main{
  ler valor1
  ler valor2

  aux = multiplica_1 (valor1, valor2)
  imprime aux
  aux = multiplica_2 (valor1, valor2)
  imprime aux
}

#multiplica usando somas sucessivas
int multiplica_1 (int num_a, int num_b){
   int x, result = 0;
   for (x =0; x < num_b; x++)
       result = result + num_a;
   return result;
}

# multiplica usando soma e deslocamento
int multiplica_2 (int num_a, int num_b){
   int result = 0;
   while(num_b > 0){
      if((num_b & 1) == 1)
         result = result + num_a;

       num_a = num_a << 1;
       num_b = num_b >> 1;
   }
   return result;
}
```