Relatório 10 - Laboratório de Arquitetura de Computadores

Luiz Junio Veloso Dos Santos - Matricula: 624037

7 de maio de 2019

Exercícios

1. Se tivermos 2 inteiros, cada um com 32 bits, quantos bits podemos esperar para o produto?

   (a) 16
   (b) 32
   (c) 64
   (d) 128

   R: c) 64

2. Quais os registradores que armazenam os resultados na multiplicação?

   (a) high e low
   (b) hi e lo
   (c) R0 e R1
   (d) $0 e $1

   R: b) hi e lo

3. Qual a operação usada para multiplicar inteiros em comp. de dois?

   (a) mult
   (b) multu
   (c) multi
   (d) mutt

   R: a) mult

4. Qual instrução move os bits menos significativos da multiplicação para o reg. 8?

   (a) move $8,lo
   (b) mvlo $8,lo
   (c) mflo $8
   (d) addu $8,$0,lo

   R: c) mflo $8

5. Se tivermos dois inteiros, cada um com 32 bits, quantos bits deveremos estar preparados para receber no **quociente**?

   (a) 16
   (b) 32
   (c) 64
   (d) 128

   R: b) 32

6. Após a instrução div, qual registrador possui o quociente?

   (a) lo
   (b) hi
   (c) high
   (d) $2

   R: a) lo

7. Qual a inst. usada para dividir dois inteiros em comp. de dois?

   (a) dv
   (b) divide
   (c) divu
   (d) div

   R: d) div

8. Faça um arithmetic shift right de dois no seguinte padrão de bits: 1001 1011

   (a) 1110 0110
   (b) 0010 0110
   (c) 1100 1101
   (d) 0011 0111

   R: a) 1110 0110

9. Qual o efeito de um **arithmetic shift right** de uma posição?

   (a) Se o inteiro for unsigned, o shift divide por 2. Se o inteiro for signed, o shift o divide por 2.
   (b) Se o inteiro for unsigned, o shift o divide por 2. Se o inteiro for signed, o shift pode resultar em um valor errado.
   (c) Se o inteiro for unsigned, o shift pode ocasionar um valor errado. Se o inteiro for signed, o shift o divide por 2.
   (d) O shift multiplica o número por dois.

   R: b)

10. Qual sequencia de instruções avalia 3x + 7, onde x é iniciado no reg. $8 e o resultado armazenado em $9?

    (a) ori $3,$0,3
        mult $8,$3
        mflo $9
        addi $9,$9,7

    (b) ori $3,$0,3
        mult $8,$3
        addi $9,$8,7

    (c) ori $3,$0,3
        mult $8,$3
        mfhi $9
        addi $9,$9,7

    (d) mult $8,$3
        mflo $9
        addi $9,$9,7

    R:

Programas

1. Escreva um programa que leia um valor A da memória, identifique se o número é negativo ou não e encontre o seu módulo. O valor deverá ser reescrito sobre o A.

   R:

```
.text
.globl main

main:
    # Colocar endereco base da memoria em t0
    addi $t0,$t0,0x1001
    sll  $t0,$t0,16

    # Obter valor da memoria
    lw   $s0,0($t0)

    # Verificar bit de sinal
    srl  $t1,$s0,31
    andi $t1,$t1,0x0001

    # Se diferente de 0, faz o modulo
    bne  $t1,$zero,nega
    j pos

nega:
    not $s0,$s0
    add $s0,$s0,1

pos:
    sw $s0,0($t0) # Salvar o resultado na memoria

.data
A: .word -42
```

programa13.asm

2. Escreva um programa que leia da memória um valor de temperatura TEMP. Se TEMP >= 30 e TEMP <= 50 uma variável FLAG, também na memória, deverá receber o valor 1, caso contrário, FLAG deverá ser zero.

   R:

```
.text
.globl main

main:
    # Colocar endereco base da memoria em t0
    addi $t0,$t0,0x1001
    sll  $t0,$t0,16

    # Obter valor TEMP da memoria
    lw   $s0,0($t0)

    # Verifica se eh >= 30
    addi $t1,$zero,30
    slt $t1,$s0,$t1 # 1 se for < 30, 0 se for >= 30

    # Se igual a 0, faz o modulo
    beq  $t1,$zero,teste2
    j zero

teste2:
    # Verifica se eh <=50
    addi $t2,$zero,50
    slt $t2,$t2,$s0 # 1 se for 50 < TEMP, 0 se for 50 >= TEMP

    # Se igual a 0, FLAG = 1
    beq $t1,$zero,um
    j zero

um:
    # Colocar um em FLAG
    addi $t3,$zero,1
    sw $t3,4($t0)
    j fim

zero:
    # Colocar zero em FLAG
    sw $zero,4($t0)
    j fim

fim:

.data
TEMP: .word 35
FLAG: .word 0
```

programa14.asm

3. Escrever um programa que crie um vetor de 100 elementos na memória onde $vetor[i] = 2 * i + 1$. Após a ultima posição do vetor criado, escrever a soma de todos os valores armazenados do vetor.

   R:

4. Considere que a partir da primeira posição livre da memória temos um vetor com 100 elementos. Escrever um programa que ordene esse vetor de acordo com o algoritmo da bolha. Faça o teste colocando um vetor totalmente desordenado e verifique se o algoritmo funciona.

   R:

5. $$y = \begin{cases} x^4 + x^3 - 2x^2 & \text{se x for par} \\ x^5 - x^3 + 1 & \text{se x for impar} \end{cases}$$

   Os valores de x devem ser lidos da primeira posição livre da memória e o valor y deverá ser escrito na segunda posição livre.

   R:

6. $$y = \begin{cases} x^3 + 1 & \text{se x > 0} \\ x^4 - 1 & \text{se x <= 0} \end{cases}$$

   Os valores de x devem ser lidos da primeira posição livre da memória e o valor y deverá ser escrito na segunda posição livre.

   R:

7. Escreva um programa que avalie a expressão: $(x * y)/z$.
   Use x = 1600000 (=0x186A00), y = 80000 (=0x13880),
   e z = 400000 (=0x61A80). Inicializar os registradores com os valores acima.

   R:

8. Escreva um programa que gere um vetor com os números ímpares até 100. O valor 1 deverá estar na primeira posição livre da memória.
   Após gerar e armazenar o vetor, seu programa deverá varrer todo o vetor, ler cada termo, somar em uma variável auxiliar e armazenar a última posição a soma de todos os elementos.
   Mostre a tabela de porcentagens das instruções utilizadas.

   R:

9. Escreva um programa que gere um vetor de inteiros até 100. Seu programa deverá armazenar na memória os números pares separados dos ímpares. Armazene primeiro os pares e logo a seguir os ímpares.
   Mostre a tabela de porcentagens das instruções utilizadas.

   R:

10. Para a expressão a seguir, escreva um programa que calcule o valor de k:
    $k = x * y$
    O valor de x deve ser lido da primeira posição livre da memória e o valor de y deverá ser lido da segunda posição livre. O valor de k, após calculado, deverá ainda ser escrito na terceira posição livre da memória.

    R:

11. O mesmo programa anterior, porém: $k = x^y$

    R: