Universidade Federal Rural do Semiarido
Departamento de Computação
Ciência da Computação
Prof. Sílvio Fernandes

Lista de Exercícios de Revisão da I Unidade

Utilize o MARS para todos os exercícios

1. Realize a tradução de C para MIPS. Considere que as variáveis *f* e *g* sejam dadas e possam ser consideradas inteiros de 32 bits, conforme declarado em um programa C.
    a) f = -g -f;
    b) f = g + (-f -5);
2. Realize a tradução de C para MIPS. Suponha que as variáveis *f*, *g*, *h*, *i* e *j* sejam atribuídas aos registradores $s0, $s1, $s2, $s3 e $s4, respectivamente. Considere que o endereço de base dos arrays A e B estejam nos registradores $s6 e $s7, respectivamente.
    a) f = g + h + B[4];
    b) f = g – A[B[4]];
3. Escreva um programa em Assembly MIPS que mapeia uma variável do tipo string, str1, para o registrador $s0. Esse programa utiliza uma função *strlen* que recebe como parâmetro o ponteiro para a string e retorna o comprimento dela, em quantidade de caracteres, sem contabilizar o marcador de fim de string (‘/0’). Implemente essa função e teste no seu programa, considerando todas as convenções de implementação de funções.

4. Simule a execução do código a seguir no MARS:

```
.data
        i: .word  5         # valor 5 armazenado em i
        msg_result: .asciiz "O reultado é "

.text
main:
        # carregando variaveis
        la  $t0, i          # endereco de "teste" em $t0
        lw  $a0, 0($t0)             # carrega o valor de "i" em $a0
        jal func

        #salvando o resulado em $s0
        add   $s0, $v0, $zero     # resultado final

        #imprime o resultado
        la   $a0, msg_result # endereco de "msg_result" em $a0
        li   $v0, 4         # especiica o servico de impressao de string
        syscall             # faz a chamada de system para imprimir a string

        add   $a0, $s0, $zero  # resultado final
        li   $v0, 1         # especifica o servido de impressao de inteiros
        syscall             # imprime o valor de resultado

        # Terminando o programa
        li   $v0, 10        # system call for exit
        syscall             # we are out of here.


#####################
# Funcao
####################
func:
                addi  $sp, $sp, -8   #ajusta pilha para 2 itens
                sw $ra, 4($sp)   # salva o endereço de retorno
                sw $a0, 0($sp) # salva o argumento n
                #Condição
                slti    $t0, $a0, 1         # teste para n < 1
                beq $t0, $zero, L1     # se n>=1 vai para L1
                #Senão for maior que 1, devolve o valor 1.
                addi $v0, $zero, 1 # retorna 1
                addi $sp, $sp, 8     # retira 2 itens da pilha
                jr      $ra             #retorna para depois de jal
                #Se for maior que 1
        L1:     addi $a0, $a0, -1  #arg1 = n – 1;
                jal     func            #chama func(n-1);
                #Restaurando registradores.
                #A próxima instrução é onde func retorna.
                lw $a0, 0($sp)     #retorna de jal: restaura n
                lw $ra, 4($sp)     #restaura endereço de retorno
                addi $sp, $sp, 8  #ajusta stack pointer

                #Devolvendo o novo $v0
                mul $v0, $a0, $v0  # retorna n * func( n - 1)
                jr $ra   # retorna para o procedimento que o chamou
```

b) Quais são os endereços de memória (na ordem que são usados) os dados colocados na pilha são armazenados?

c) Quais os valores dos registradores $sp, $ra e $pc em cada armazenamento na pilha?

d) O que muda se o valor da variável “i”, no início do programa, mudar para 10?