UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA - DEPARTAMENTO DE INFORMÁTICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Prof. Anderson Faustino da Silva

6895 - Arquitetura e Organização de Computadores II

Segunda Prova

Ricardo Henrique Brunetto RA: 94182

Maringá
Agosto - 2017

Introdução

O presente documento contém as questões respondidas da segunda prova aplicada pelo Prof. Anderson Faustino da Silva na disciplina de Arquitetura e Organização de Computadores II para a turma de Bacharelado em Ciência da Computação de 2015. Tal documento fora desenvolvido unicamente e explicitamente para o propósito avaliativo.

O conteúdo aqui citado é advindo, além das anotações em aula e materiais disponibilizados pelo professor, da gama de referências bibliográficas por ele recomendadas e encontradas pelo autor.

Questão 01

Para a resolução da questão foi desenvolvido um código em assembly para MIPS32 que realiza a cópia de um subvetor em outro vetor. São argumentados à função `copy` os seguintes dados:

- **$a0** - Endereço do primeiro vetor
- **$a1** - Endereço do segundo vetor
- **$a2** - Índice inicial
- **$a3** - Índice final

O programa, então, copiará os elementos entre, e inclusive, `$a2` e `$a3` do vetor cujo endereço inicial está em `$a0` para o vetor cujo endereço inicial está em `$a1`. O detalhamento do código será omitido aqui, uma vez que o mesmo o faz com comentários.

```
.data
exit_val: .int  10
.text
# Copia um subvetor de v1 para v2
# Recebe como parametros
# a0 - endereco do primeiro vetor (primeiro elemento do vetor)
# a1 - endereco do segundo vetor (primeiro elemento do vetor)
# a2 - indice inicial
# a3 - indice final
# Copia o subvetor a0[a2 ... a3] para a1[0 ... a3-a2]
copy:
  sub $v0, $a3, $a2          # $v0 sera o contador
  add $a0, $a0, $a2          # coloca $a0 em $a0[$a2]
  looping:
    lw $at, 0($a0)           # carrega o elemento do subvetor para $at
    sw $at, 0($a1)           # armazena o elemento no vetor
    addu $a0, $a0, 4         # avanca 4 bytes em a0
    addu $a1, $a1, 4         # avanca 4 bytes em a1
    sub $v0, $v0, 1          # $v0--
    bgtz $v0, looping        # if ($v0 > 0) looping
fim:
  li $v0, exit_val
  syscall
```

../assembly.s

Tabela 1: My caption

ER	**Busy**	**Op**
add1		
add2		
add3		
mul1		
mul2		
load1		
load2		
load3		
load4		
load5		
store1		
store2		
store3		
store4		
store5		

Referências