UNIVERSIDADE FEDERAL DE ALAGOAS – UFAL
INSTITUTO DE COMPUTAÇÃO - IC

Disciplina: ORGANIZAÇÃO E ARQUITETURA DE COMPUTADORES
Curso: Engenharia e Ciência da computação
Docente: Erick de Andrade Barboza
Turma: M Ano: 2016 - 2º Semestre

Prova Módulo 1

ALUNO:

---

1) (2 pontos) Responda verdadeiro (V) ou falso (F) nas assertivas abaixo. Caso seja selecionada a opção F, descreva o erro.

a) Um processador com arquitetura ~~CISC~~ **RISC** possui poucas instruções de máquina, que são executadas diretamente pelo hardware, para que ocorram em alta velocidade. Os processadores ~~RISC~~ **CISC** possuem instruções complexas que são executadas por microprogramas. (**F**)

b) Um dos objetivos de o CISC ter um conjunto mais rico de instruções é poder completar uma tarefa com um conjunto de linhas em Assembly do menor tamanho possível. (**V**)

c) A versão 2 de um determinado processador terá um hardware específico para realizar a operação de multiplicação (multiplicador), que na versão 1 era realizada pela ALU. Portanto, os códigos produzidos pela versão 1 ~~não~~ funcionarão na versão 2, pois a organização do processador foi modificada. (**F**)

d) Os registradores são memórias auxiliares que podem ser de dados (RDM) ou de endereços (REM), sendo que normalmente os de dados têm tamanho ~~menor que~~ **igual** a palavra do processador e os de endereços podem ter tamanhos iguais ou maiores que a palavra. (**F**)

e) O montador é o elemento que converte instruções em linguagem de montagem para linguagem de máquina. (**V**)

2) (2 pontos) (ENADE 2005) Duas máquinas, M1 e M2, implementam um mesmo conjunto de instruções, dos tipos A, B e C. O quadro abaixo mostra o número de ciclos de relógio de que cada máquina necessita para executar cada tipo de instrução. As freqüências dos relógios das máquinas M1 e M2 são, respectivamente, 1 GHz e 500 MHz. Um programa P possui 50% de suas instruções do tipo A, 30% do tipo B e 20% do tipo C.

Tipo de Instrução	Ciclos por instrução para M1	Ciclos por instrução para M2
A	5	3
B	2	1
C	10	4

Da análise da situação exposta, pode-se concluir que o programa P será executado, aproximadamente,

(A) duas vezes mais rápido na máquina M1 do que na máquina M2.
(B) duas vezes mais rápido na máquina M2 do que na máquina M1.
(C) quatro vezes mais rápido na máquina M1 do que na máquina M2.
(D) quatro vezes mais rápido na máquina M2 do que na máquina M1.
(E) no mesmo tempo em ambas as máquinas M1 e M2.

CPI_{M1} = 0,5*5 + 0,3*2 + 0,2*10 = 5,1 -> 6 ciclos para executar o programa P

CPI_{M2} = 0,5*3 + 0,3*1 + 0,2*4 = 2,6 -> 3 ciclos para executar o programa P

$Tempo_{M1} = 6 / 1 \times 10^{6} = 6x10^{-6}$ **$Tempo_{M2} = 3 / 5 \times 10^{-5} = 6x10^{-6}$**

3) (2 pontos) Sobre as implementações monociclo e multiciclo do MIPS responda:

a) Qual a grande diferença entre estas implementações? Qual a principal vantagem de uma sobre a outra?

- **Monociclo = um ciclo para cada instrução; Multiciclo = mais de um ciclo**
- **Multiciclo permite uma melhor utilização do tempo de CPU pois na monociclo o ciclo definido precisa ser grande o suficiente para que a instrução mais longa seja executada.**

b) Em qual das implementações é necessário apenas uma memória? O que permite que essa implementação tenha apenas uma memória ao invés de duas?

- **Na Multiciclo**
- **A memória é utilizada em ciclos de clock distintos**

c) Qual das implementações possui uma unidade de controle mais simples? Justifique.

- **Monociclo, o controle se resume basicamente ao controle da ALU. Multiciclo possui mais multiplexadores e sinais de escrita.**

d) Considerando a implementação multiciclo, quais são as ações realizadas pelo processador na fase 1 (busca de instrução)?

- **Buscar instrução na memória e enviar para o IR**
- **Atualizar o valor do PC (PC += 4)**

4) (2 pontos) Escreva uma subrotina em MIPS que implemente a mesma lógica da rotina swap descrita no código C abaixo:

```
void swap(int v[], int k)
{
      int temp;
      temp = v[k];
      v[k] = v[k+1];
      v[k+1] = temp;
}
```

```
swap: sll $t1, $a1, 2          # reg $t1 = k * 4
      add $t1, $a0, $t1        # reg $t1 = v + (k * 4)
                               # reg $t1 has the address of v[k]
      lw $t0, 0($t1)           # reg $t0 (temp) = v[k]
      lw $t2, 4($t1)           # reg $t2 = v[k + 1]
                               # refers to next element of v
      sw $t2, 0($t1)           # v[k] = reg $t2
      sw $t0, 4($t1)           # v[k+1] = reg $t0 (temp)
      jr $ra                   # return to calling routine
```

5) (2 pontos) O seguinte fragmento de código processa dois vetores e retorna um valor importante no registrador $v0. Assuma que cada array consiste de 2500 palavras indexadas de 0 a 2499, que o endereço base do vetor está salvo em $a0 e $a1 respectivamente, e que os tamanhos deles (2500) estão salvos em $a2 e $a3 respectivamente. Adicione comentários ao código e descreva em uma frase o que esse código faz. Especificamente, o que será retornado em $v0? **Será retornado em v0 a quantidade de números iguais entre os dois arrays.**

```
      sll $a2, $a2, 2            # a2*=4 (multiplica tamanho por 4)
      sll $a3, $a3, 2            # a3*=4 (multiplica tamanho por 4)
      add $v0, $zero, $zero      # v0=0
      add $t0, $zero, $zero      # t0=0
outer:add $t4, $a0, $t0          # t4=a0+t0
      lw $t4, 0($t4)             # t4=array1[t0]
      add $t1, $zero, $zero      # t1=0
inner:add $t3, $a1, $t1          # t3=a1+t1
      lw $t3, 0($t3)             # t3=array2[t1]
      bne $t3, $t4, skip         # se t3!=t4 -> skip
      addi $v0, $v0, 1           # else v0 += 1
                                 # ou seja, conte caso o conteúdo
                                 # retirado dos arrays sejam iguais
skip: addi $t1, $t1, 4           # t1 += 4 (próxima posição do array2)
      bne $t1, $a3, inner        # t1 != a3 -> inner
                                 # não atingiu o tamanho do array2
      addi $t0, $t0, 4           # t0 += 4 (próxima posição do array1)
      bne $t0, $a2, outer        # t0 != a2 -> outer
                                 # não atingiu o tamanho do array1
```