CI212 - Organização e Arquitetura de Computadores
Primeira Lista de Exercícios [v1.0, 25ago05]

Cap2: 4,6,13,29-31,49-51, Cap3: 7,9,10,27,28,35-40,42-44

1) Justifique a codificação (formato e largura dos campos) das instruções do processador MIPS.

2) Qual o problema mais sério com a codificação das instruções do 80x86? Qual a diferença com relação ao conjunto de instruções do MIPS? Como esta(s) diferença(s) afetam o desempenho dos processadores?

3) Descreva *todos* os modos de endereçamento suportados pela linguagem de montagem do processador MIPS. Desenhe diagramas mostrando os componentes do endereço efetivo.

4) [2.38] Considerando o modo de endereçamento PC-relativo, explique em que circunstâncias um montador não conseguiria traduzir/implementar diretamente a instrução de desvio no trecho de código abaixo.

```
agora:  beq $s0, $s1, depois
        ...
depois: add $s0, $s0, $s0
```

5) Descreva o mecanismo de suporte a subrotinas/funções no conjunto de instruções do MIPS. Dê um exemplo de uso indicando o estado da pilha antes da chamada, durante a execução e após o retorno da subrotina/função.

6) Uma técnica de programação sugere que um programa deve ser implementado como um conjunto de funções, onde o código de cada função não excede o tamanho de uma página de papel (15-25 linhas de código mais comentários). Considerando as convenções de código de máquina do MIPS, discuta a eficiência de código com muitas chamadas de função. O que pode ser feito para melhorar a eficiência do código? Quem deve fazê-lo?

7) [2.37] Pseudoinstruções não são parte do conjunto de instruções MIPS embora apareçam freqüentemente em programas codificados em assembly. Para cada pseudoinstrução na tabela abaixo escreva uma seqüência mínima de código com instruções reais do MIPS que efetuem o mesmo trabalho. Pode ser necessário usar $at para algumas seqüências. 'gde' representa uma constante que necessita de 32 bits para ser representada e 'peq' uma constante que pode ser representada em 16 bits.

pseudoinstrução	semântica (o que efetua)
move $t1, $t2	$t1 = $t2
clear $t5	$t5 = 0
beq $t1, peq, L	if ($t1 == peq) goto L
beq $t1, gde, L	if ($t1 == gde) goto L
li $t1, peq	$t1 = peq
li $t2, gde	$t1 = gde
ble $t3, $t4, L	if ($t3 <= $t4) goto L
bgt $t5, $t6, L	if ($t3 > $t4) goto L
bge $t5, $t6, L	if ($t3 >= $t4) goto L
addi $t0, $t2, gde	$t0 = $t2 + gde
lw $t5, gde($t2)	$t5 = M[$t2 + gde)
abs $t3, $t2	$t3 = ($t3 >= 0 ? $t3 : compl2($t3)) [3 instr]

10) Mostre o código MIPS necessário para implementar o seguinte comando C:

```
int x[NNN], y[MMM];
a = x[10] + x[ y[3] ];
```

11) [3.30,31] Considerando os padrões de bits abaixo, mostre o que eles representam se forem (a) inteiro em complemento de 2, (b) inteiro sem sinal, (c) número em ponto flutuante (float), (d) instrução do MIPS.

```
1010 1101 0001 0000 0000 0000 0000 0010
0010 0100 1001 0010 0100 1001 0010 0100
```