Revisão para a Primeira VA de Infraestrutura de Hardware – Parte 2

1)

5.1 Formatos de instruções

Nos formatos das instruções MIPS (Figura 5.4), podemos reduzir o campo do código de operação de 6 para 5 bits e o campo da função de 6 para 4 bits. Dado o número de instruções MIPS e as funções necessárias, essas mudanças não limitarão o projeto; isto é, ainda teremos códigos de operações e códigos de funções suficientes. Os 3 bits ganhos dessa forma podem ser usados para estender os campos `rs`, `rt` e `rd` de 5 para 6 bits cada.

a. Liste dois efeitos positivos dessas mudanças. Justifique sua resposta.
b. Liste dois efeitos negativos dessas mudanças. Justifique sua resposta.

2)

Exemplo 5.3: Compilando estruturas if-then-else Linguagens de alto nível possuem uma construção *if-then-else* que nos permite realizar uma ou duas computações, dependendo de quando uma determinada condição é satisfeita. Mostre uma seqüência de instruções MiniMIPS correspondente à seguinte declaração if-then-else:

```
if (i <= j) x = x + 1; z = 1; else y = y - 1; z = 2 * z;
```

Solução: Isso é muito similar à declaração if-then, exceto que precisamos de instruções correspondentes à parte do "else" e de uma maneira de saltar a parte do else, quando a parte "then" for executada.

```
        slt    $t0, $s2, $s1        # j < i? (inverso da condição)
        bne    $t0, $zero, else     # desvio em j < i para a parte "else"
        addi   $t1, $t1, 1          # início da parte "then": x = x + 1
        addi   $t3, $zero, 1        # z = 1
        j      endif                # salta a parte "else"
else:   addi   $t2, $t2, -1         # início da parte "else": y = y - 1
        add    $t3, $t3, $t3        # z = z + z
endif: ...
```

Note que cada uma das duas seqüências de instruções, correspondentes às partes "then" e "else" da declaração condicional, podem ser arbitrariamente longas.

5.9 Desvio condicional

Modifique a solução do Exemplo 5.3 para que a condição testada seja:

a. `i < j`

b. `i >= j`

c. `i + j <= 0`

d. `i + j > m + n`

3)

5.5 Compilação de um comando de chaveamento (*switch/case*)

Um comando de chaveamento (switch/case) permite desvio em múltiplos caminhos baseado em valores de uma variável inteira. No exemplo seguinte, a variável de chaveamento `s` pode assumir um entre três valores em [0, 2] e uma ação diferente é especificada para cada caso.

```
switch (s)  {
        case 0:  a = a + 1;  break;
        case 1:  a = a - 1;  break;
        case 2:  b = 2 * b:  break;
}
```

Mostre como essa instrução pode ser compilada em instruções assembly do MiniMIPS.

4)

6.15 Conversão ASCII/inteiro

a. Um número decimal é armazenado na memória na forma de uma cadeia ASCII terminada com um caráter nulo. Escreva um procedimento MiniMIPS que receba o endereço de início da cadeia ASCII no registrador `$a0` e retorne o valor inteiro equivalente no registrador `$v0`. Ignore a possibilidade de *overflow* e assuma que o número pode iniciar com um dígito ou um sinal (+ ou –).

5)

9.11 Conversão entre bases numéricas

Converta cada um dos seguintes números das suas bases indicadas, para as representações nas bases 2 e 16.

a. Números na base 3: 1021; 1021 2210; 1021 2210 2100 1020
b. Números na base 5: 302; 302 423; 302 423 140
c. Números na base 8: 534; 534 607; 534 607 126 470
d. Números na base 10: 12; 5 655; 2 550 276; 76 545 336; 3 726 755
e. Números na base 12: 9a5; b0a; ba95; a55a1; baabaa

6)

9.12 Conversão entre bases numéricas

Converta cada um dos seguintes números de ponto fixo, das suas bases indicadas para a representação na base 10.

a. Números na base 2: 10,11; 1011,0010; 1011 0010,1111 0001
b. Números na base 3: 10,21; 1021,2210; 1021 2210,2100 1020
c. Números na base 8: 53,4; 534,607; 534 607,126 470
d. Números na base 12: 7a,4; 7a4,539; 7a4 593,1b0
e. Números na base 16: 8,e; 8e,3a; 823a,51c0

7)

9.13 Conversão entre bases numéricas

Converta cada um dos seguintes números de ponto fixo, das suas bases indicadas para as representações nas bases 2 e 16.

a. Números na base 3: 10,21; 1021,2210; 1021 2210,2100 1020
b. Números na base 5: 30,2; 302,423; 302 423, 140
c. Números na base 8: 53,4; 534,607, 534 607,126 470
d. Números na base 10: 1,2; 56,55; 2 550,276; 76 545,336; 3 726,755
e. Números na base 12: 9a,5; b,0a; ba,95; a55,a1; baa,baa

8)

11.2 Multiplicação sem sinal

Multiplicar os seguintes números binários de 4 bits, sem sinal. Apresente seu trabalho no formato da Figura 11.2a.

a. $x = 1001$ e $y = 0101$

b. $x = 1101$ e $y = 1011$

c. $x = 0{,}1001$ e $y = 0{,}0101$

9)

11.10 Multiplicação por constantes

Usando apenas deslocamentos e instruções de soma e subtração, imagine um procedimento eficiente para multiplicação de cada uma das constantes seguintes. Assuma operandos sem sinal de 32 bits e assegure que resultados intermediários não excederão o intervalo dos números sem sinal de 32 bits. Não modifique os registradores de operandos e use apenas um outro registrador para resultados parciais.

a. 13
b. 43
c. 63
d. 135