

Universidade Federal do Vale do São Francisco – UNIVASF
Curso: Engenharia da Computação

Disciplina: Organização e Arquitetura de Computadores I

Professor: Leonardo Barreto Campos

Lista de Exercícios - III

1. (Patterson & Hannessy 2.4) Por que o MIPS não tem uma instrução de subtração imediata?

2. (Patterson & Hannessy 2.21) Acrescente comentários ao código MIPS a seguir e descreva em uma sentença o que ele calcula. Supondo que \$a0 e \$a1 sejam usados para a entrada e ambos obtenham inicialmente os inteiros a e b, respectivamente. Suponha que \$v0 seja usado para a saída.

```

                                lw    $t1, 1;                #
                                add   $t0, $zero, $zero      #
loop:                          beq   $a1, $zero, final     #
                                add   $t0, $t0, $a0         #
                                sub   $a1, $a1, $t1        #
                                j     loop                  #
final:                          addi $t0, $t0, 100         #
                                add   $v0, $t0, $zero      #
    
```

3. (Patterson & Hannessy 2.22) O seguinte trecho de código processa dois arrays e produz um valor importante no registrador \$v0. Considere que cada array consista em 2500 palavras indexadas de 0 a 2499, que os endereços base dos arrays sejam armazenados em \$a0 e \$a1, respectivamente, e que seus tamanhos (2500) sejam armazenados em \$a2 e \$a3, respectivamente. Acrescente comentários ao código e descreva em uma sentença o que esse código faz. Especificamente, o que será retornado em \$v0?

```

                                sll   $a2, $a2, 2
                                sll   $a3, $a3, 2
                                add   $v0, $zero, $zero
                                add   $t0, $zero, $zero
outer:                          add   $t4, $a0, $t0
                                lw    $t4, 0($t4)
                                add   $t1, $zero, $zero
inner:                          add   $t3, $a1, $t1
                                lw    $t3, 0($t3)
                                bne   $t3, $t4, skip
                                addi $v0, $v0, 1
skip:                            addi $t1, $t1, 4
                                bne   $t1, $a3, inner
                                addi $t0, $t0, 4
                                bne   $t0, $a2, outer
    
```

4. (Patterson & Hannessy 2.23) Supondo que o código do Exercício 3 seja executado em uma máquina com um clock de 2GHz que exija a seguinte quantidade de ciclos para cada instrução:

Instrução	Ciclos
add, addi, sll	1
lw, bne	2

No pior caso, quantos segundos levará a execução desse código?

5. (Patterson & Hannessy 2.10) Implemente o seguinte código C em MIPS, supondo que `set_array` seja a primeira função chamada:

```

1 int i;
2 void set_array(int num)
3 {
4     int array[10];
5     for(i=0; i<10; i++)
6         array[i] = compare(num, i);
7 }
8 int compare(int a, int b)
9 {
10    if (sub(a, b) >= 0)
11        return 1;
12    else
13        return 0;
14 }
15 int sub (int a, int b)
16 {
17    return a - b;
18 }
    
```

Não se esqueça de tratar o stack pointer e o frame pointer corretamente. A variável é alocada na pilha, e `i` corresponde a `$s0`. Desenhe o status da pilha antes de chamar `set_array` e durante cada chamada de função. Indique os nomes dos registradores e variáveis armazenadas na pilha e marque o local de `$sp` e `$fp`.

6. (Patterson & Hannessy 2.26) O programa a seguir tenta copiar words do endereço no registrador `$a0` para o endereço no registrador `$a1`, contando o número de words copeadas no registrador `$v0`. O programa termina de copiar quando encontra uma Word igual a 0. Você não precisa preservar o conteúdo dos registradores `$v1`, `$a0` e `$a1`. Essa Word de término deverá ser copiada, mas não contada.

```

        addi $v0, $zero, 0 # Inicializa contador
loop:   lw $v1, 0($a0)      # Lê próxima word da origem
        sw $v1, 0($a1)    # Escreve no destino
        addi $a0, $a0, 4 # Avança ponteiro p/ próx origem
        addi $a1, $a1, 4 # Avança ponteiro p/ próx destino
        beq $v1, $zero, loop # Repete se a word != zero
    
```

7. (Patterson & Hannessy 2.38) Considere o seguinte trecho de código Java:

```

for (i=0; i<=100; i++)
    a[i] = b[i] + c;
    
```

Suponha que a e b sejam arrays de words e que o endereço base de a esteja em $\$a0$ e o endereço base de b esteja em $\$a1$. O registrador $\$t0$ está associado à variável i e o registrador $\$s0$ ao valor de c . Você também pode considerar que quaisquer constantes de endereço de que precise estejam disponíveis para serem lidas da memória. Escreva o código assembly do MIPS. Quantas instruções são executadas durante a execução desse código se não houver exceções de array fora dos limites? Quantas referências aos dados da memória serão feitas durante a execução?