🕛

Aula 06 - 12/12/22 - Deslocamento e Instruções de desvio

Deslocamento Lógico

Ex.: 1011 → srl 1 → 0101 , inseriu o 0 a esquerda e jogou o 1 da direita fora (bit menos significativo).

srl * → deslocamento lógico, em que * é a quantidade de casas deslocadas.

Deslocamento Aritmético

Ex.: 1011 → sra 1 → 1101 , inseriu o 1 (conserva o sinal) e jogou o 1 da direita para fora (bit menos significativo).

sra * → deslocamento aritmético, em que * é a quantidade de casas deslocadas.

Ex.: 0101 → sra 1 → 0010 , joga o 1 (LSB = Lower Significant Bit) para fora.

Instruções de desvio

Desvio condicional

- desvia o fluxo se uma condição for satisfeita
- beq rs, rt ,label
  - Se rs == rt, desvia p/ a instrução rotulada por label.
- bne rs, rt, label (branch if not equal)
  - Se rs ≠ rt, desvia p/ a instrução rotulada por label.
  - rs e rt = conteúdos dos registradores.

Desvio incondicional

- j label (jump)
  - desvia p/ a instrução com rótulo label.

Formato tipo J

op	endereço
6 bits	26 bits

Exemplo:

```
// Linguagem C
if (i == j) f = g+h;
else f = g-h;
```

```
; Linguagem Assembly
; Mapeamento:
f - $t0
g - $t1
h - $t3
i - $t4
j - $t5

; 1° usando beq
beq $t4, $t5, soma
sub $t0, t1, $t3 ;f= g-h
j sair ;jump

soma:
  add $t0, $t1, $t3 ;f = g+h
sair:
  ...

; 2° usando bne
bne $t4, $t5, sub
add $t0, $t1, $t3
j sair ;jump
sub:
  sub $t0, $t1, $t3
sair:
...

; 3° usando beq e bne
beq $t4, $t5, soma
```

```
bne $t4, $t5, subtrai
soma:
  add $t0, $t1, $t3
  j sair ;jump
subtrai:
  sub $t0, $t1, $t3
sair:
...
```

Exemplo

```
// Linguagem C
int i, k, m;
i = 0; // move $t0, $zero
while (i != m)
{
  k = k+1;
  i++;
}
```

```
; Linguagem Assembly
; Mapeamento
i - $t0
j - $t1
k - $t2
m - $s0

; Solução não adequada:
soma:
  addi $t2, $t2, 1 ;k = k+1
  addi $t0, $t0, 1 ;i = i+1
bne $t0, $s0, soma

; Solução para fazer como se fosse o do while:
laco: beq $t0, $s0, sair ; vejo quando ele deve ser encerrado
addi $t2, $t2, 1 ; k = k+1
addi $t0, $t0, 1 ; i++
j laco ;jump
```

Obs: bge (≥), bgt (>), ble (≤) e blt (<) são pseudoinstruções.

O hardware p/ executá-las seria mais lento.

- verificar = e ≠ requer apenas uma instrução, enquanto >, ≥, <, ≤ requerem mais.

Instruções de comparação

- slt / slti (set on less than immediate) rd, rs, rt / const
- Se rs < rt, rd = 1, caso contrário, rd = 0.

Ex.: if (i < j) k++;

```
; Mapeamento
i - $t0
j - $t1
k - $t2

slt $t9, $t0, $t1 ; se i < j, $t9 = 1
beq $t9, $zero , sair
addi $t2, $t2, 1 ; k++

sair:
...
```

Para pensar:

- Como fazer >, ≥ e ≤ usando slt ?
- Como implementar as pseudoinstruções bge, bgt, ble e blt ?