Aula 6 - Instruções de desvio

Na aula passada:

- Representações das expressões do tipo R e tipo I
- Shift

Instruções de desvio

Há duas instruções de desvio: - beq: branch if equal - bne: branch if not equal

Estrutura da instrução:

```
beq/bne $s0, $s1, label
```

O label indica um endereço de memória, esse instrução é do tipo I. No caso do beq:

- Se $s0 = $s1, desvie para a label. Na prática, a instrução calcula 's0 - $s1' e verifica se o valor é 0 ou diferente de 0.

Instruções de comparação

Temos a seguinte instrução, e sua versão imediata: - slt: set on less than - slt: set on less than immediate

Estrutura da instrução:

```
slt $t0, $s0, $s1
```

- Se $s0 < $s1: $t0 = 1
- Senão: $t0 = 0

Observação: slt é instrução do tipo R e slti é do tipo I.

E as demais comparações?

Sabendo fazer: ==, != e <, como podemos fazer <=, >= e >?

- a > b é equivalente a b < a, basta inverter a ordem dos registradores na instrução slt/slti.
- <= é equivalente a !>.
- >= é equivalente a !<.

Exemplo: - Verificar `$s0 <= $s1`:

```
slt $t0, $s1, $s0 # $t0 = $s1 < $s0, ou seja, $t0 = 1 se s1 >= s0, ou seja, s0 <= s1.
beq $t0, $zero, verdadeiro # vai para verdadeiro se s0 = s1.
```

> Obs: Existem as pseudoinstruções: blt (branch on less than), ble (branch on less or equal) e bgt (branch on greater than) e bge (branch on greater or equal). Embora existam, não são implementadas naturalmente na arquitetura.

Desvio incondicional

- j label: jump para label.

Essa instrução é do tipo J.

Formato de instrução do tipo J: | op | endereco | | — | — | | 6 bits | 26 bits |

Compilando ifs:

Ex:

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

f, g, h, i, j => $s0, $s1, $s2, $s3, $s4

```
    beq $s3, $s4, soma
    j subtracao
soma:
    add $s0, $s1, $s2
subtracao:
    sub $s0, $s1, $s2
```

Compilando laços

```
for (i=0, f=0;i<h;i++)
    f+=i;
```

```
int i=0;
int f=0;
while(i<h){
    f = f + i;
    i = i + 1;
}
```

f, h, i -> $s0, $s1, $t0

```
    add $t0, $zero, $zero # i=0
    add $s0, $zero, $zero # f=0
loop:
    slt $t1, $t0, $s1 # t1 = i<h
    beq $t1, $zero, exit
    add $s0, $s0, $t0
    addi $t0, $t0, 1
    j loop
exit:
```