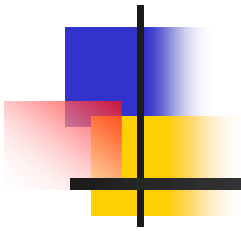

Università degli Studi di Roma "La Sapienza"

Architettura degli elaboratori II

Istruzioni PASCAL in assembly

Come realizzare l'istruzione IF

if *COND* **then** *COD*

Si esegue del codice COD solo se la condizione COND è vera

In assembly:

- Codice per valutare COND e mettere il risultato in un registro
- Salto condizionato per saltare COD se la condizione è *falsa*
- COD, da eseguire *solo* quando la condizione è *vera*

Esempio di traduzione di un IF

if $x > 0$ **then** $y = x*2$

```
In assembly:
.text
lw $t0,x          # Carica x
blez $t0,endif # Salto se COND falsa
# COD:
   mul $t1,$t0,2    # x*2
   sw $t1,y           # Salva $t1 in y
endif: # Fine if

.data
x: .word 5 # valore
y: .word 0 # y = x*2 se x>0; y = 0
```

Come fare l' IF… ELSE

if *COND* **then** *COD1* **else** *COD2*

Si esegue COD1 se un COND è vera; altrimenti si esegue COD2

In assembly:

- Codice per valutare COND e mettere il risultato in un registro
- Salto condizionato per saltare COD1 se la condizione è *falsa*
- COD1, da eseguire *solo* quando COND è vera
- Salto incondizionato per evitare di eseguire *anche* COD2
- COD2, da eseguire *solo* quando la condizione è falsa

Esempio di traduzione di un IF… ELSE

if $x > 0$ **then** $y = x*2$ **else** $y = -x$

```
In assembly:
lw $t0, x          # Carica x
blez $t0, else     # salta se x <= 0
# ramo THEN:
    mul $t1,$t0,2    # x*2
    j end            # Salta alla fine
else:
# ramo THEN:
    neg $t1, $t0     # -x
end:
 sw $t1, y           # memorizza il valore in y

.data
x: .word 5         # valore
y: .word 0         # risultato
```

Istruzione IF Annidati

Si esegue del codice solo se una condizione è vera altrimenti si esegue altro codice

In assembly: applico uno dentro l'altro i due schemi di soluzione già visti

NB: codice complicato e poco chiaro

if x > 0 **then**

if y >= 0 **then** y = x*2 **else** y = -x

else y++

```
In assembly:
lw $t0,x          # Carica x
lw $t1,y          # Carica y
blez $t0,else1    # Salta se cond1 falsa
    bltz $t1,else2     # Salta se cond2 falsa
    mul $t1,$t0,2      # Ramo then2: $t1 = x*2
    j end              # Salta alla fine
    else2:neg $t1,$t0  # Ramo else2: $t1 =-x
    j end              # Salta alla fine
else1:
add $t1,$t1,1     # Ramo else1: $t1=y+1
end: sw $t1, y    # Scrive $t1 in y
```

Istruzione SWITCH

switch *VAR* **case**
V1: *COD1*;
V2: *COD2* ;
…
Vk: *CODk* ;
default: *COD* ;

Ogni ramo testa valori diversi di una stessa variabile ed esegue di conseguenza codice diverso

In assembly: un test per ogni ramo, con una relativa etichetta

Esempio di traduzione di uno SWITCH

switch op **case**
0: y = x*3; 1: y = x+5;
2: y = -x; **default**: y = x

In assembly:

```
        lw $t1,x
        lw $t0,op
        beq $t0,0,case0
        beq $t0,1,case1
        beq $t0,2,case2
        j default
case0:  mul $t2,$t1,3       # $t2 = x*3
        j end
case1:  add $t2,$t1,5       # $t2 = x+5
        j end
case2:  div $t2,$t1,2       # $t2 = x/2
        j end
default: move $t2,$t1       # $t2 = x
end:    sw $t2,y        # Scrive $t2 in y
```

Traduzione più efficiente

Due soli test anziché uno per ogni ramo:

uso la variabile come indice per caricare da una tabella l'indirizzo del codice dove saltare

```
lw $t1,x
lw $t0,op
bge $t0,3,default
blt $t0,0,default
mul $t0,$t0,4              # $t0 = op*4
lw $t0,table($t0)          # $t0 = stable + 4*op
jr $t0                     # salto alla routine giusta
case0:mul $t2,$t1,3        # $t2 = x*3
j end
case1:addi $t2,$t1,5       # $t2 = x+5
j end
case2:div $t2,$t1,2        # $t2 = x/2
j end
default: move $t2,$t1      # $t2 = x
end:sw $t2,y      # Scrive $t2 in y
.data
table: .word case0,case1,case2
```

Istruzione WHILE

while *COND* **do** *COD*

Continua ad eseguire COD finché COND resta vera

N.B.: non so se COND è vera all'inizio; quindi va controllata prima di eseguire COD

In assembly:

- valuta COND e mettere il risultato in un registro
- Salto condizionato per saltare COD se COND è falsa
- COD, da ripetere finché COND resta vera:
  - Salto condizionato al punto 3 per eseguire COD un'altra volta se COND è vera

  oppure
  - Salto al punto 1 in modo da poter eseguire il tutto un'altra volta

Esempio di traduzione di un WHILE

while $x > 0$ **do**
$y = y + x;$
$x = x - 1$

In assembly:

```
lw $t0,x          # Carica x
lw $t1,y          # Carica y
blez $t0,end      # salta se x <= 0
while:            # (Inizio ciclo)
   add $t1,$t1,$t0   # y += x
   sub $t0,$t0,1     # x--
bgtz $t0,while
end:
sw $t0,x          # Salva x
sw $t1,y          # Salva y
```

oppure:

```
lw $t0,x
lw $t1,y
test:
    blez $t0,end
    add $t1,$t1,$t0
    sub $t0,$t0,1
    j test
end:
sw $t0,x
sw $t1,y
```

Istruzione DO-WHILE

Esegue più volte del codice finché una espressione resta vera, con l'espressione vera all'inizio

```
do
    y = y + x;
    x = x – 1
while  x > 0
```

In assembly:

- Codice da ripetere finché la condizione è vera
- Valuta la condizione e mette il risultato in un registro
- Salto condizionato al punto 1 per eseguire un'altra volta se la condizione è *vera*

```
In assembly:

lw $t0,x      # Carica x
lw $t1,y      # Carica y
dowhile:
   add $t1,$t1 $t0 # y += x
   sub $t0,$t0,1   # x--
bgtz $t0,dowhile   # Riesegue il ciclo se
                   # la cond. è vera
sw $t0,x      # Salva x
sw $t1,y      # Salva y
```

Istruzione FOR

for *i* = *V1* **to** *Vk* **do** *COD*

Esegue COD per un certo numero di volte; si può anche usare l'indice *i* (numero di volte eseguite) nel codice.

In assembly:

- Inizializzazione di un registro (da usare come indice) a *V1*
- COD, da ripetere il numero stabilito di volte
- Incremento del registro usato come indice
- Salto condizionato al punto 2 per rieseguire COD, se l'indice è al più *Vk*

Esempio di traduzione di un FOR

for i = 1 **to** 5 **do** y = y + x

In assembly:

```
li $t0,1            # Inizializza indice
lw $t1,x            # Carica x
lw $t2,y            # Carica y
loop:
   add $t2,$t2,$t1          # y += x
   add $t0,$t0,1            # t0++
   ble $t0,5,loop           # Cicla se t0 <= 5
sw $t2,y            # Salva y
```