Università degli Studi di Milano
Corso di Laurea in
Informatica per la Comunicazione Digitale
Architettura degli Elaboratori

Esempi di programmi in MIPS

Controllo di flusso (parte 2)

Marco Tarini

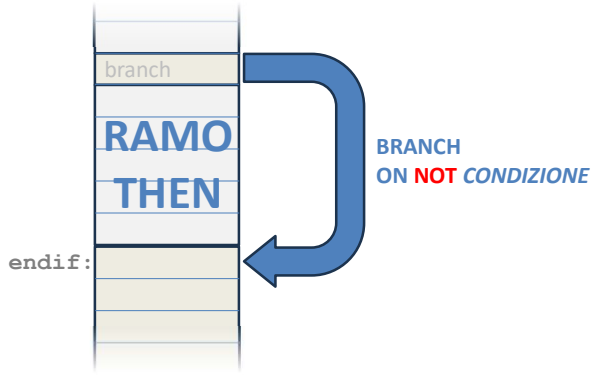
107

Da alto a basso livello

- Vediamo come tradurre
  a basso livello (in **Assembly**)
  i tipici costrutti di **controllo di flusso**
  forniti dai linguaggi ad alto livello (Go, C, etc)
  - E' il lavoro del **complilatore**
  - I modi che vediamo non sono certo unici
  - Sta al complilatore trovare il modo più ottimizzato

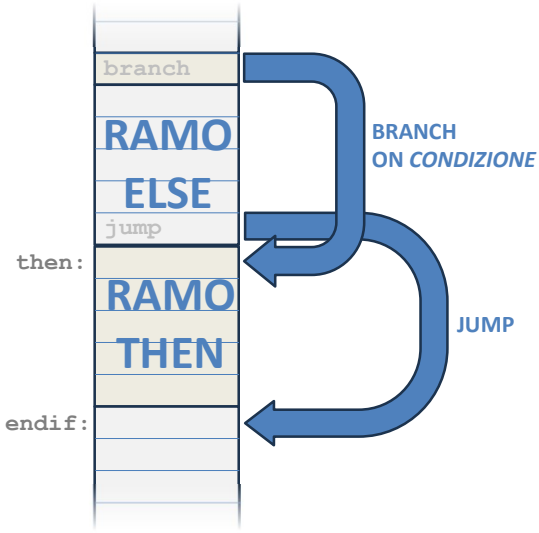
108

if <condizione> then …



109

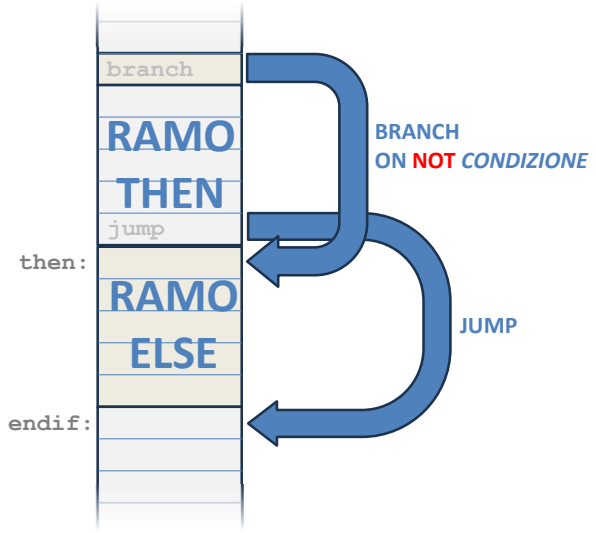
if <condizione> then … else …



110

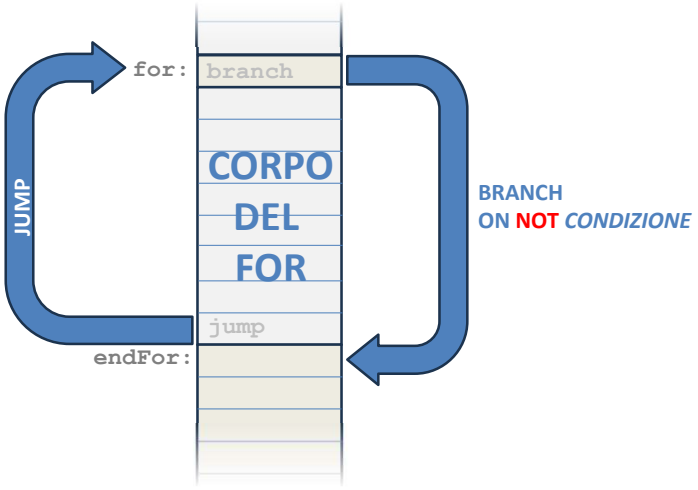
In alternativa…

if <condizione> then … else …



111

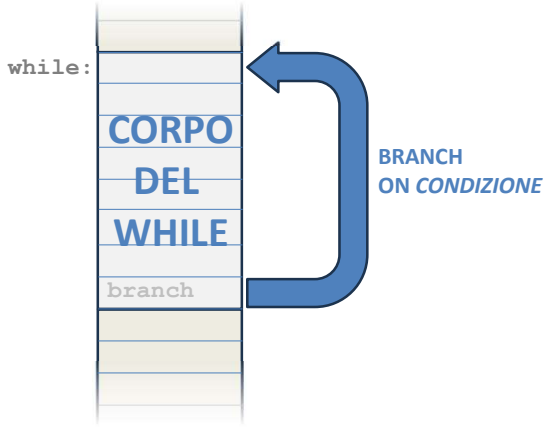
for <condizione> do …



112

Costrutto in C/C++/C#/Java/JavaScript (ma non in Go):

do {…} while <condizione>

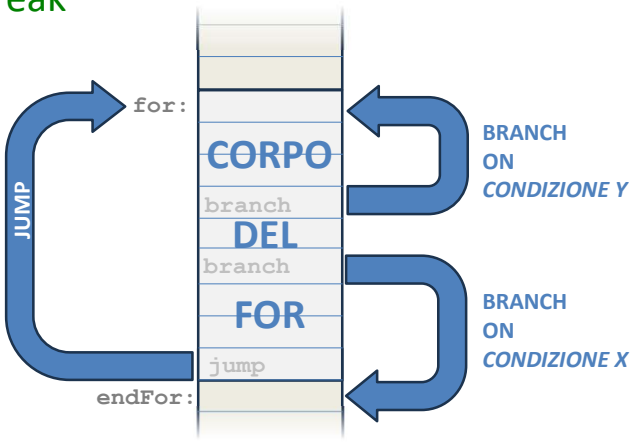


(la differenza con **for** del Go è che il test è alla fine delle iteraz, quindi ne viene fatta almeno una)

113

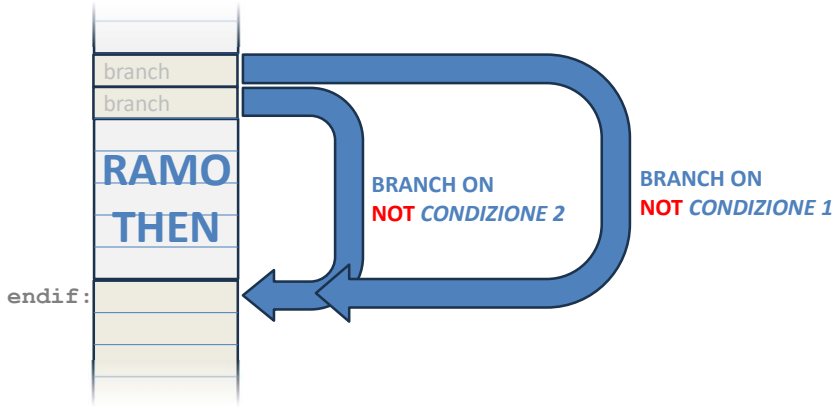
ciclo infinito, in go

```
for {
  …
  if condizioneY continue
  …
  if condizioneX break
  …
}
```



114

if <condizione1> and <condizione2> then …



(appena non vale la 1° o la 2° condizione, si sorvola il ramo then…)

115

Controllo di flusso nei linguaggi ad alto livello (una nota)

Alcuni linguaggi ad alto livello, come il C e il Go, prevedono anche un costrutto per il controllo di flusso del tipo:

goto <locazione> (da "go to" = vai a)

che è il diretto equivalente di una **Jump** in linguaggio macchina/assembly

La teoria ci dice: non è mai *necessario* usare il goto.

- Bastano sempre gli altri costrutti disponibili (if-then-else, loop come for o while, switch-case, chiamate di funzioni…)

Le buone pratiche di programmazione raccomandano: mai usare i goto.

- Perché portano facilmente a scrivere "*spaghetti*" code
- E' uno dei principii della programmazione detta *ben strutturata*
- Come tutte le prassi di programmazione (che è un'arte), ammette eccezioni

Tornando a basso livello (p.e. al MIPS): siamo invece obbligati ad usare i salti.
Non c'è altro, per controllare il flusso del programma.

(Vedi anche: https://xkcd.com/292/)

116

Esercizi 1/2

- Improvvisati «compilatore» e traduci in MIPS-32 i seguenti brani di (pseudo) codice C e Go
  - NB: servirà anche la lezione successiva su «sistema operativi»
  - Prova il tuo codice su https://mipsweb.di.unimi.it

```
do{
   Scrivi «inserisci un numero (o negativo per uscire)»;
   leggi numero;
}
while (numero >= 0);
```

```
età := 16
limiteEtà := 18
prezzo := 0

if età < limiteEtà  {
  prezzo = 80
} else {
  prezzo = 100
}

prezzo += 5 // tasse
```

```
int età = 16 ;
int prezzo = 36;

/* sconto del 20% *
 * per teenagers! */

if età è fra 13 e 19 (compresi)
{
  prezzo = prezzo * 80 / 100 ;
}
```

esprimilo con 2 condizioni in and!

118

Esercizi 2/2

\n = «accapo»

```
scrivi(«indovina la mia età...\n»)
for {
   leggi(numero);
   if (numero == 35) break
   scrivi(«\nsbagliato, ritenta!\n»)
}
scrivi(«giusto!\n»)
```

Questo elemento è la guardia, o terminatore. Segnala che la lista di voti è terminata.
NB: non è un voto!

Più difficile:

```
voti := [] int { 30, 18, 30, 18, -1 }
somma := 0
for i := 0 ;; i++ {
   x := voti[i]
   if x < 0 break // l'array di voti è finito
   somma += x
}
media := somma / 4
```

119