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

Esempi di programmi in MIPS

Assegnamenti condizionali

Marco Tarini

139

Come tradurre a basso livello il seguente brano di codice?

```
var maggiorenne bool
maggiorenne = ( età < 18 )

…

if maggiorenne { … }
```

Uso dei registri:
`$s4` : maggiorenne
`$s5` : età

```
...
slt $s4 $s5 18
...
```

- 140 -

140

I boolean a basso livello

- Una variabile di tipo "boolean" a basso livello è solo un intero (su n bits) che vale 0 (falso) oppure 1 (vero)
- Come assegnarlo (o inizializzarlo), ad es., per il caso visto sopra?
  - Si potrebbe pensare di usare un costrutto if-then-else ("se età < 18 allora assegnalo a 1 altrimenti assegnalo a 0")
  - Per questo caso molto comune, gli assembly (come MIPS) mettono a disposizione un costrutto apposito
  - risparmia istruzioni e soprattutto salti!
- **set-on-less-then** $d $a $b
  - Mette il registro $d a 1 se $a < $b, e a 0 altrimenti
  - Ricorda: «to set» [un bit] = mettere quel bit a 1
  - Nota: viene testata una condizione (minoranza fra due registri), ma non è un salto! Niente è un costrutto di controllo di flusso.
  - E' solo un'operazione della ALU come un'altra, fra due operatori ($a e $b) il cui risultato (0 oppure 1) è memorizzato in un registro ($d)

141

Compara: istruzione di "branch on"…

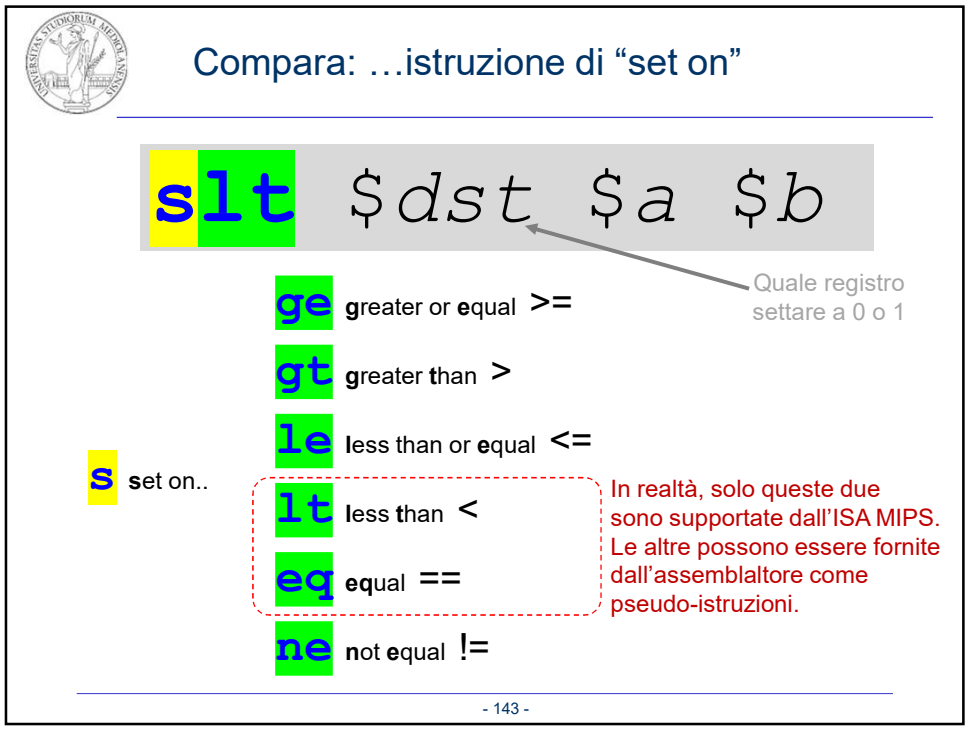
```
blt  $a  $b  imm
```

quante istruzioni saltare (se si salta)

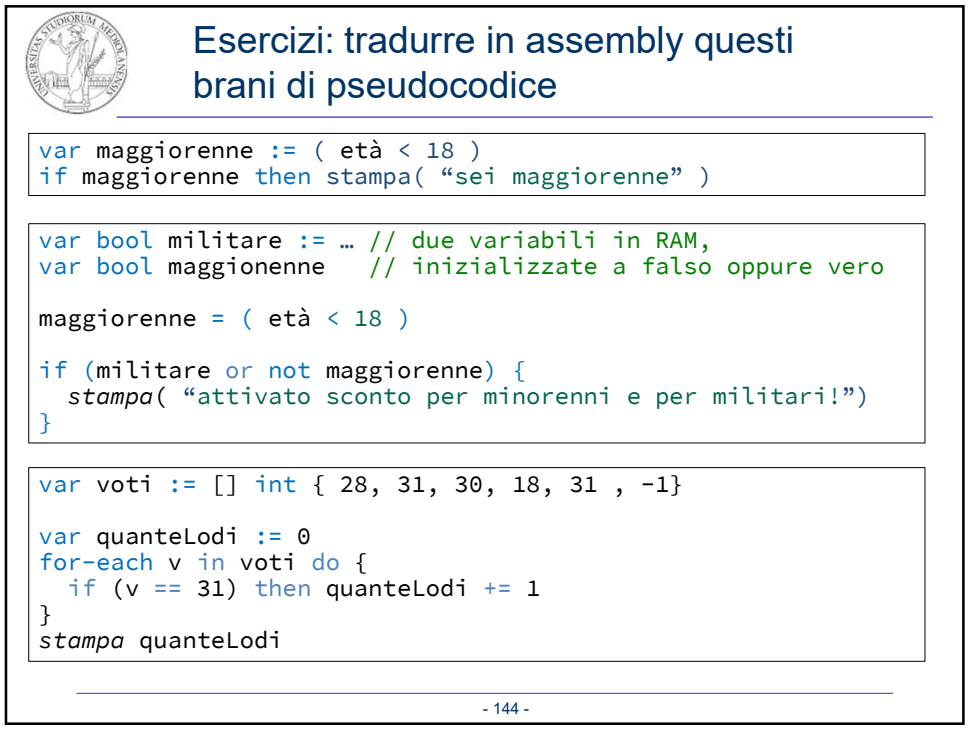
b branch on

- **ge** greater or equal >=
- **gt** greater than >
- **le** less than or equal <=
- **lt** less than <
- **eq** equal ==
- **ne** not equal !=

142


Compara: …istruzione di “set on”
slt $dst $a $b
Quale registro settare a 0 o 1
ge greater or equal >=
gt greater than >
le less than or equal <=
s set on..
lt less than <
eq equal ==
ne not equal !=
In realtà, solo queste due sono supportate dall’ISA MIPS. Le altre possono essere fornite dall’assemblaltore come pseudo-istruzioni.
- 143 -

143


Esercizi: tradurre in assembly questi brani di pseudocodice
var maggiorenne := (età < 18)
if maggiorenne then stampa(“sei maggiorenne”)
var bool militare := … // due variabili in RAM,
var bool maggionenne // inizializzate a falso oppure vero
maggiorenne = (età < 18)
if (militare or not maggiorenne) {
stampa(“attivato sconto per minorenni e per militari!”)
}
var voti := [] int { 28, 31, 30, 18, 31 , -1}
var quanteLodi := 0
for-each v in voti do {
if (v == 31) then quanteLodi += 1
}
stampa quanteLodi
- 144 -

144

Note alle soluzioni (vedi le soluzioni date sul sito)

Secondo esercizio: (sconto per militari e maggiorenni)

- Trattiamo l'espressione logica a guardia dell'if come qualsiasi altra expr
  - per esempio quelle matematiche, come 3 x (10+6)
- Cioè:
  - calcoliamo ogni sotto-espressione (come «not maggionenne»)…
  - Qui, attraverso operazioni logiche,
  - …e ne memorizziamo il valore in registri temporanei
- Quando il valore dell'espressione logica completa è pronta, la usiamo come condizione per un branch
  - cioè, per saltare il ramo then se vale false, cioè 0
- Sottoproblema: come calcolare il not di un «bool» (un intero che vale 1 o 0)?
  - Il costrutto «not» (not bit a bit) non va bene, perché inverte tutti i bit, e vero, 1, (0…01 in binario) diverrebbe 1…10
  - Invece, effettuiamo un XOR con 0….01, che inverte solo il bit meno significativo

- 145 -

145

Note alle soluzioni (vedi le soluzioni date sul sito)

Terzo esercizio: (contare quante lodi ci sono nel libretto)

- E' necessaria una scansione lineare di un vettore
  - cioè processare tutti i suoi elementi in un ciclo,
  - uscendone quando troviamo la guardia -1
- Per ogni elemento letto, bisogna sommare 1 al numero di lodi, ma solo quando quell'elemento vale 31
  - Prima idea: usare un if-then: «if (elemento-letto == 31) then incrementa numero-di-lodi»
  - Idea più semplice: possiamo sommare al numero di lodi il valore «booleano» (quindi, intero 0 o 1) settato come «uguaglianza a 31»
  - Infatti: se l'elemento vale 31, l'ugualianza vale, e quindi sommo 1; altrimenti, sommo 0
  - Nota: questo *non* sarebbe consentito in molti linguaggi ad alto livello, in cui un bool non è un intero e non è usabile come tale
  - (non posso sommare un int ad un bool)

- 146 -

146