Università degli Studi di Milano «La Statale»
Dipartimento di Informatica

Esempi di programmi in MIPS

Controllo di flusso (parte 1)

Marco Tarini

85

Controllo di flusso: nei linguaggi ad alto livello

Come in C / C++ / Java / Go

Costrutti condizionali:

- if (…) then { … }
- if (…) then { … } else { … }

Costrutto switch:

- switch (expr) {
  case 1: …; case 2: …; case 3: … ;
  default: …;
  }

86

Controllo di flusso: nei linguaggi ad alto livello

Cicli (loops):

- while (*condizione*) { *fai qualcosa* }
- do { *fai qualcosa* } while (*condizione*)
- for (*init* ; *condiz* ; *passo*) {
  fai qualcosa
  }

87

Controllo di flusso: nei linguaggi basso livello

Ciascuno dei costrutti sopra, in un linguaggio assembly (a basso livello), è realizzato attraverso istruzioni di

- **Jump**
- **Branch**

Cioè: i salti (condizionati e non) rappresentano l'*unico* costrutto di controllo di flusso dei linguaggi a basso livello.

Vediamo degli esempi di uso.

88

Un primo esempio:

- Utilizziamo un salto per non effettuare una divisione, se il divisore è zero

```
.text:
li $s0 10
li $s1 0

# qui altro codice, che potenzialmente
# modifica $s1

beq $s1 $zero +2

div $s0 $s1
mflo $s2    # s2 diventa $s1/$s0

nop
```

90

Un primo esempio: note

- La **beq** (**branch on equal**) effettua il salto solo se $s1 (il divisore) è, in quel momento, uguale a (il valore del registro) zero.
- In caso contrario: niente salto, e l'esecuzione prosegue con l'istruzione successiva, come normale
- La destinazione del salto è di **2** istruzioni sotto: cioè salta a 2 istruzioni dopo l'istruzione successiva
- Quindi, il salto evita le due istruzioni che effettuano la divisione
- Il salto arriva dunque fino all'istruzione «**nop**» (no operation) in fondo
- Il salto, dunque, incrementa il PC di 4x2 = 8 byte *in aggiunta* al normale incremento di 4 byte (ricorda: ogni istruzione occupa 4 byte)
- Some sappiamo**,** il valore +2 è memorizzato del campo **immediate** (di 16 bit) dell'istruzione **beq**
- Se questo fosse stato un valore negativo, sarebbe saltato indietro, al di sopra della **beq**

91

Come trovare l'indirizzo di destinazione di un salto (jump o branch?)

- Attraverso le etichette!
- Nota: le etichette possono essere usate tanto nel segmento `.code` quanto nel segmento `.data`
- In entrambi i casi, registrano l'indirizzo di memoria del dato o dell'istruzione immediatamente successiva

92

If – Then: esempio

Codice Go:

```
età := 16
limiteEtà := 18
prezzo := 100
/* sconto per i minorenni */
if età < limiteEtà  { prezzo = 80 }
prezzo += 5   // tasse (per tutti)
```

```
.data
eta: .word 16
limiteEta: .word  18
prezzo: .word  100

.text
          lw $s0 eta          # uso $s0 per «eta»
          lw $s1 limiteEta    # uso $s1 per «limiteEta»
          lw $s2 prezzo       # uso $s2 per «prezzo»

          bge $s0 $s1 saltaQui
          li $s2 80

saltaQui: addi $s1 $s1 5      # tasse
          sw $s1 prezzo
```

93

If – Then: esempio (note)

- Invece di testare la condizione
  (**età < limiteEtà**),
  testiamo se vale *il suo opposto*
  (**età ≥ limiteEtà**, cioè età «greater equal» limite),
- In questo caso, saltiamo a piè pari il «**ramo then**»
  (che consiste nella sovrascrittura di prezzo con il valore 80)
- In caso contrario, il codice prosegue normalmente dentro al nel ramo then, che è posto di seguito alla branch
- In entrambi i casi, si passa poi a eseguire il codice che segue l'etichetta (l'incremento del prezzo di 5, e poi la sua memorizzazione in RAM)
- Nota: non mi è necessario contare quante istruzioni devo saltare per «passare sopra» al ramo then, grazie al meccanismo delle etichette
  - Menomale, perché il conteggio sarebbe reso difficile dalle pseudoisruzioni

94

Riassunto: cosa fa per noi l'assembler nel tradurre un comando branch

1. Traduce da etichetta a *target address* di 32 bit
2. Calcola la differenza fra
   indirizzo a cui si trova l'istruzione da cui si salta
   al quello in cui si trova il target a cui salta
   - Espresso come come numero di istruzioni
     (numero di byte diviso 4)
   - Nota: questo numero può essere un positivo o negativo
   - Meno 1: è richiesto il numero di istruzioni da saltare
     oltre al normale incremento del PC di 4 byte
3. Codifica il risultato nel campo immediate dell'istruzione
   - In complemento a 2

GRAZIE, assembler!

95

If – Then: esempio (note aggiuntive - ripasso)

- Come sappiamo, le istruzioni **`lw`** e **`sw`** (**load word** e **store word**), quando prendono, come secondo parametro, un'etichetta, sono pseudo-istruzioni equivalenti ad una coppia di istruzioni:
  - La prima, per assegnare l'indirizzo indicato dall'etichetta in un registro temporaneo (si usa ovviamente **`$at`**)
  - L'altra per effettuare la lettura (load) o scrittura (store) in RAM all'indirizzo contenuto nel registro temporaneo

```
lw $s2 prezzo
```

```
la $at prezzo
lw $s2 ($at)
```

  - la **`la`** (**load address**) è a sua volta una pseudoistruzione sostituita da un'altra coppia di istruzioni (ciascuna per copiare 2 dei 4 byte in **`$at`**)
- Nota sintattica: le etichette vanno terminate col «due punti» quando sono definite, ma non quando sono usate

96

Un esempio «tossico»

```
.text
b -1
```

Questo mini-programma consiste in un'unica istruzione che salta (sempre) a… se stessa!

- **`b`** = **branch** (salto condizionato) ma con condizione "always" (quasi un ossimoro)
- Destinazione del salto: un'istruzione *sopra* quella successiva
- E' un loop infinito minimale
- Alternativa equivalente:

```
.text
beq $zero $zero -1
```

97

Esercizi 1/2

- Come al solito, testare il codice visto su mipsweb, e osservarne l'assembly e l'esecuzione
- Scrivere un programma in Assembly che: date quattro variabili in RAM **a**, **b**, **c**, **d, max** (di valore iniziale 12, 15, 20, e 18)
  1. Copia il valore di **a** in $s0
  2. Se il valore di **b** è *strettamente maggiore* di $s0, lo copia in $s0 (sovrascrivendolo)
  3. Ora lo stesso, con **c**
  4. Ora lo stesso, con **d**

  Usa registri *temporanei* per memorizzare i valori letti nei passi 2,3,4
- Test: verifica il valore di $s0 alla fine del programma sarà il valore *massimo* delle quattro variabili
  - verifica che questo sia il caso modificando il valore iniziale delle quattro variabili in RAM
- Modifica il programma in modo che il valore di $s0 sia il valore minimo dei quattro

98

Esercizi 2/2

Nell'esempio visto con "età" e "limite età", modifica la condizione:

- effettua lo sconto solo se l'età è esattamente di 18 anni
- (più difficile e fantasioso): effettua lo sconto se l'età è *compresa* fra 14 e 18 anni *inclusi* (Suggerimento: "salta" lo sconto se è l'età è strettamente maggiore di 18… e anche se è strettamente minore di 14!)

Verifica la correttezza del codice eseguendolo dopo aver inizializzato l'età a vari valori.

99