


La struttura delle istruzioni elementari: il linguaggio Macchina

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

Università degli Studi di Milano


Sommario

Linguaggio macchina

- Le istruzioni in linguaggio assembly devono essere tradotte in linguaggio macchina (cioè in sequenze di 0 e 1) per poter essere eseguite.
- Le istruzioni in linguaggio macchina sono lunghe **32 bit** (come i registri e le parole di memoria).
- Il parsing di un'istruzione in linguaggio macchina viene fatta dalla CPU che ricava le inforamazioni necessarie all'esecuzione dell'istruzione stessa nella fase di decodifica.
- Occorre definire l'architettura delle istruzioni:

  Come vengono raggruppati i bit?

  Cosa viene rappresentato nel singolo bit?

Linguaggio assembly → Linguaggio macchina

L'alfabeto del linguaggio macchina

Codifica binaria

- Codifica posizionale
- 2 simboli, base 2.

$01001101 = 0x2^7 + 1x2^6 + 0x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 77$

Codifica esadecimale

- 16 simboli, base 16.

$0x4D = 4x16^1 + 13x16^0 = 77$

Relazione tra codifica binaria ed esadecimale

0100 -> 4 4 -> 0100

1101 –> D D -> 1101

Sommario

Il linguaggio macchina

Le istruzioni di tipo R

Le istruzioni di tipo I

Le istruzioni di tipo J

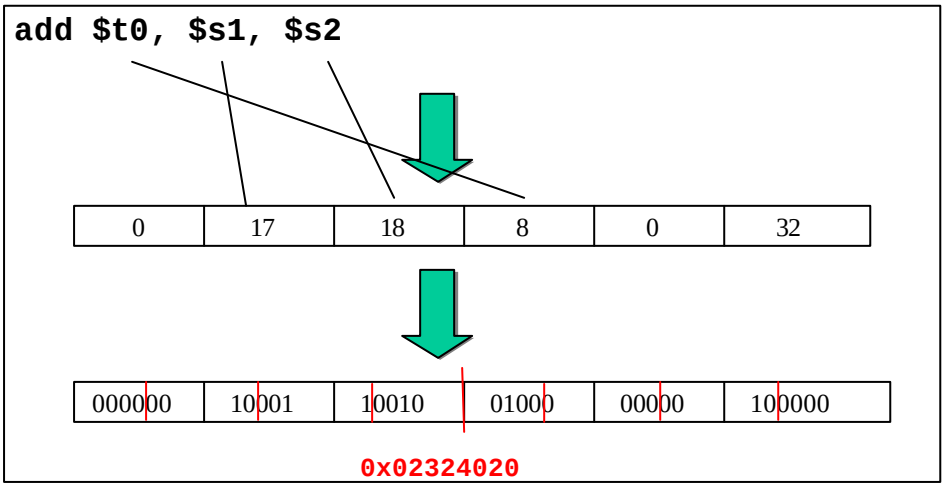
Formato istruzioni di tipo R

Campo di un'istruzione: numero di bit consecutivi contenenti un'informazione per l'esecuzione.

op	rs	rt	rd	shamt	funct
6 bit	5 bit	5 bit	5 bit	5 bit	6 bit

- Ai vari campi sono stati assegnati dei nomi mnemonici:
  - **`op:`** (opcode) identifica il tipo di istruzione
  - **`rs:`** registro contenente il primo operando sorgente
  - **`rt:`** registro contenente il secondo operando sorgente (target)
  - **`rd:`** registro destinazione contenente il risultato
  - **`shamt:`** shift amount (scorrimento)
  - **`funct:`** indica la variante specifica dell'operazione

Istruzioni di tipo R: esempio



Istruzioni di tipo R

- Istruzioni aritmetico-logiche con il tipo di formato visto, vengono chiamate di **tipo R** (registro).
- Esempi:
  - somma, prodotto, divisione
  - shift (scorrimento)
  - AND, OR, NOT
- Le diverse istruzioni aritmetico-logiche di tipo R si distinguono tra loro in base al campo **`funct.`**

Istruzioni di tipo R: esempi

Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
`sub $s1, $s2, $s3`	000000	10010	10011	10001	00000	100010

Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
`and $s1, $s2, $s3`	000000	10010	10011	10001	00000	100100

Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
`sll $s1, $s2, 7`	000000	X	10010	10001	00111 (7)	000000

Nome campo	op	rs	rt	rd	shamt	funct
Dimensione	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
`srl $s1, $s2, 7`	000000	X	10010	10001	00111 (7)	000010

Formato R ed operazioni logico-matematiche

Non tutte le operazioni logico-matematico, sono di tipo R.

Le operazioni logico-matematiche di tipo R hanno codice operativo 0.

Non tutte le operazioni con codice operativo 0 sono logico-matematiche (ad esempio ci sono le istruzioni di *jr*, *syscall…*).

Occorre distinguere il funzionamento dell'istruzione elementare dalla sua codifica.

- Codifiche simili (e.g. Tipo R) possono essere condivise da istruzioni di tipo diverso (e.g. aritmetico-logiche, salto).
- Codifiche diverse (e.g. Tipo I e Tipo R) possono essere condivise da istruzioni dello stesso tipo (e.g. add ed addi)

Sommario

Il linguaggio macchina

Le istruzioni di tipo R

Le istruzioni di tipo I

Le istruzioni di tipo J

Linguaggio macchina

op	rs	rt	rd	shamt	funct
6 bit	5 bit	5 bit	5 bit	5 bit	6 bit

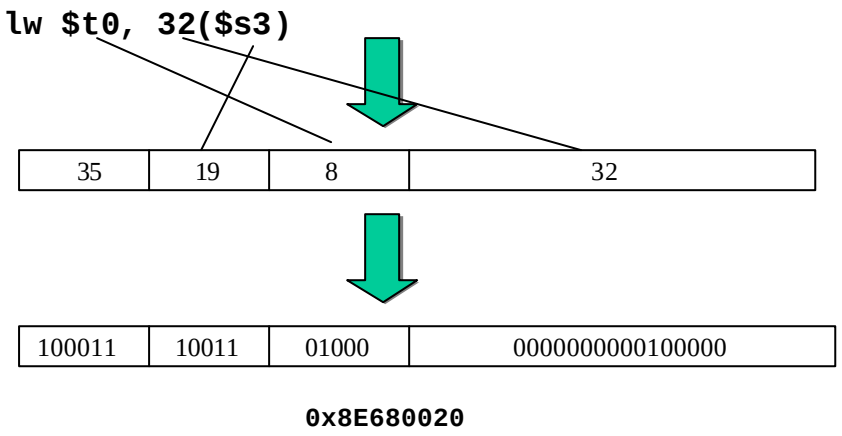
- Il formato delle istruzioni di tipo R non è adatto a rappresentare istruzioni di load/store.
- Al campo indirizzo (offset) delle istruzioni **lw** e **sw** sarebbe riservato un campo di **5 bit** (le costanti sarebbero al massimo di dimensione $2^5 = 32$)
- Per le istruzioni di load/store si utilizza un formato diverso (**tipo I**) utilizzando sempre 32 bit complessivi.

Formato istruzioni di tipo I

op	rs	rt	Indirizzo (=costante)
6 bit	5 bit	5 bit	16 bit

- In questo caso, i campi hanno il seguente significato:
  - **`op`** identifica il tipo di istruzione;
  - **`Rs`** indica il registro sorgente. Nel caso di una lw contiene il registro base;
  - **`rt`** indica il registro target. Nel caso di una lw, contiene il registro destinazione dell'istruzione di caricamento;
  - **`Costante`**. Nel caso di una lw riporta lo spiazzamento (offset).
- Con questo formato una istruzione **lw (sw)** può indirizzare byte nell'intervallo -2^{15} , $+2^{15}-1$ rispetto all'indirizzo base.

Istruzioni di tipo I: esempio



Istruzioni di tipo I: esempi

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`lw $t0, 32 ($s3)`	100011	10011	01000	0000 0000 0010 0000

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`sw $t0, 32 ($s3)`	101011	10011	01000	0000 0000 0010 0000

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`addi $t0, $s3, 32`	001000	10011	01000	0000 0000 0010 0000

Esempio

A[300] = h + A[300]

```
lw $t0, 1200($t1)
add $t0, $s2, $t0
sw $t0, 1200($t1)
```

`$s2` —— h

`$t1` —— Indirizzo base di A

35	9	8	1200		
0	18	8	8	0	32
43	9	8	1200		

100011	01001	01000	0000010010110000		
000000	10010	01000	01000	00000	100000
101011	01001	01000	0000010010110000		

```
0x8D2804B0
0x02484020
0xAD2804B0
```

Versione I di istruzioni aritmetico-logiche

Nome campo	op	rs	rt	"Indirizzo"
Dimensione	6-bit	5-bit	5-bit	16-bit
`addi $s1, $s2, 4`	001000	10001	10001	0000 0000 0000 0100

Nome campo	op	rs	rt	"indirizzo"
Dimensione	6-bit	5-bit	5-bit	16-bit
`slti $t0, $s2, 8`	001010	10010	01000	0000 0000 0000 1000

```
# $t0 = 1 if $s2 < 8
```


Istruzioni di controllo di flusso (salto)

- Il PC viene incrementato di 4 (byte) durante l'esecuzione di un'istruzione.
- Salti condizionati relativi (beq, bne...) – Formato I:
  - Il flusso sequenziale di controllo cambia solo se la condizione è vera.
  - Il calcolo del valore dell'etichetta **`L1`** (indirizzo di destinazione del salto) è relativo al Program Counter (PC).
- Salti incondizionati assoluti (j, jr, jal...) – Formato J:
  - Il salto viene sempre eseguito.
  - L'indirizzo di destinazione del salto è un indirizzo assoluto di memoria.

Istruzioni di salto condizionato

- Salti condizionati relativi:
  - **`beq r1, r2, L1`** (*branch on equal*)
  - **`bne r1, r2, L1`** (*branch on not equal*)

- Salti condizionati relativi:
  - Il flusso sequenziale di controllo cambia solo se la condizione è vera.
  - Il calcolo del valore dell'etichetta **`L1`** (indirizzo di destinazione del salto) è relativo al Program Counter (PC).

Esempio

Assembly		Assembly tradotto
Loop:	**add $t1, $s3, $s3**	**80000: add $t1, $s3, $s3**
	……	**……**
	bne $t0, $s5, Exit	**80016: bne $t0, $s5, 8**
	add $s3, $s3, $s4	**80020: add $s3, $s3, $s4**
	beq $t0,$s5, Loop	**80024: beq $t0,$s5, -28**
Exit:		**80028:**
	……	

2 = (80028 - 80020) / 4

-7 = (80000 - 80028) / 4

Nota: quando si esegue la **bne**, PC punta già all'istruzione successiva (e.g. Bne -> PC = 80020)

Analisi dell'offset

- Nelle architetture MIPS tutte le istruzioni hanno 32 bit (RISC) e sono allineate al MSB. I due bit meno significativi dell'indirizzo delle istruzioni sono quindi sempre 00
- Per l'offset si hanno a disposizione solo 16-bit del campo **`indirizzo`** ⇒ rappresentano un indirizzo di **parola** relativo al PC **(PC-relative word address).**
- Una istruzione di **salto** può indirizzare **parole** nell'intervallo **-2^{15}** , **$+2^{15}-1$** rispetto all'indirizzo base.
- Esempio: **`bne $s0, $s1, L1`**
  - Per esprimere l'etichetta **`L1`** si hanno a disposizione solo 16-bit ⇒ il valore di **`L1`** è calcolato rispetto al Program Counter in modo da saltare di 2^{15} **parole** avanti o indietro rispetto all'istruzione corrente.
- Per il **principio di località** degli indirizzi di memoria è utile calcolare l'indirizzo di destinazione del salto come **offset** rispetto all'istruzione corrente.
- L'indirizzamento relativo al Program Counter permette di fare dei salti condizionati ad aree di memoria il cui indirizzo non è esprimibile con 16-bit.

Istruzioni di salto condizionato: assembly e linguaggio macchina

- I 16-bit del campo indirizzo esprimono **l'offset** rispetto al PC rappresentato in complemento a due per permettere salti in avanti e all'indietro.
- L'offset varia tra **-2^{15}** e **$+2^{15}-1$**
- L'offset rappresenta lo spiazzamento espresso in numero di parole.
- L'etichetta riportata nell'istruzione Assembly rappresenta la locazione di memoria (in byte) a cui l'esecuzione deve saltare.

Esempio: **`bne $s0, $s1, L1`**

L'assemblatore sostituisce l'etichetta **`L1`** con l'indirizzo **di parola** relativo a PC: **(L1-PC)/4**

- Il PC contiene già l'indirizzo dell'istruzione successiva al salto
- La divisione per 4 serve per calcolare lo spiazzamento in numero di parole.

Formato istruzioni di salto condizionato

op	rs	rt	Indirizzo
6 bit	5 bit	5 bit	16 bit

- Nel caso di salti condizionati, i campi hanno il seguente significato:
  - **`op`** identifica il tipo di istruzione;
  - **`rs`** indica il primo registro;
  - **`rt`** indica il secondo registro;
  - **`Indirizzo`** riporta lo spiazzamento (offset). Il valore del campo indirizzo può essere negativo (salti all'indietro)

Esempio

Assembly		Assembly	Linguaggio Macchina
Loop:	**add $t1, $s3, $s3**	**add $t1, $s3, $s3**	**80000: 0 19 19 9 0 32**
	……	**……**	
	bne $t0, $s5, Exit	**bne $t0, $s5, 8**	**80016: 5 8 21 2**
	add $s3, $s3, $s4	**add $s3, $s3, $s4**	**80020:**
	beq $t0,$s5, Loop	**beq $t0,$s5, -28**	**80024: 4 8 21 -7**
Exit:			**80028: (Exit) ...**
	……		

2 = (80028 - 80020) / 4
-7 = (80000 - 80028) / 4

Nota: quando si esegue la **bne**, PC punta già all'istruzione successiva (e.g. Bne -> PC = 80020)

Branch: esempi

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`beq $s1, $s2, 100`	000100	10001	10010	0000 0000 0001 1001

L1 = 100 in byte Codifica su 18 bit: (00) 000 0000 0001 1001(00) in binario.

Nome campo	op	rs	rt	indirizzo
Dimensione	6-bit	5-bit	5-bit	16-bit
`beq $s1, $s2, -100`	000100	10001	10010	1111 1111 1110 0111

L1 = -100 in byte Codifica su 18 bit: (11)111 1111 1110 0111(00) in binario.

Sommario

Il linguaggio macchina

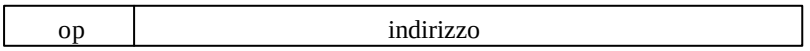
Le istruzioni di tipo R

Le istruzioni di tipo I

Le istruzioni di tipo J

Formato istruzioni di tipo J

- E' il formato usato per le istruzioni di salto incondizionato *(jump)*:



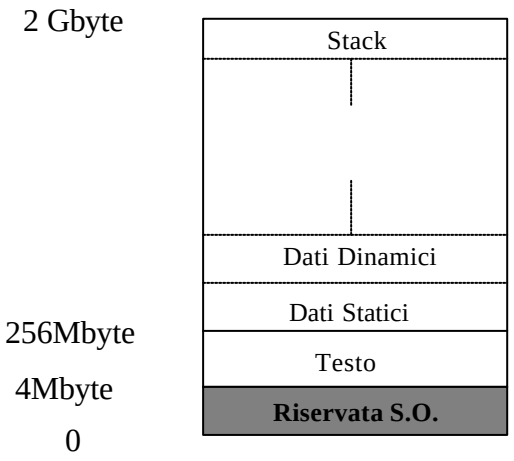
6 bit 26 bit

- In questo caso, i campi hanno il seguente significato:
  - **op** indica il tipo di operazione**;**
  - **indirizzo** (composto da **26-bit**) riporta una parte (26 bit su 32) dell'indirizzo **assoluto** di destinazione del salto.
- I 26-bit del campo **indirizzo** rappresentano un indirizzo di parola **(word address)**

Istruzioni di salto incondizionato

- L'assemblatore sostituisce l'etichetta **`L1`** con i 28 bit meno significativi traslati a destra di 2 (divisione per 4 per calcolare l'indirizzo di parola) per ottenere 26-bit
  - In pratica elimina i due 0 finali
  - Si amplia lo spazio di salto:
    si salta tra 0 e 2^{28} Byte (2^{26} word) = 256 Mbyte.
- I 26-bit di indirizzo nelle jump rappresentano un indirizzo di parola (word address) $\Rightarrow$ corrispondono ad un indirizzo di byte (byte address) composto da 28-bit.
- Poiché il registro PC è composto da 32-bit $\Rightarrow$ l'istruzione jump rimpiazza solo i 28-bit meno significativi del PC, lasciando inalterati i rimanenti 4-bit più significativi.
- La memoria testo è compresa tra 0 e 256Mbyte, i rimanenti 4 bit saranno 0000.

Organizzazione logica della memoria



Esempio

```
Loop:   add $t1, $s3, $s3        80000: 0 19 19 9 0 32        do { t1 = s3*2;
        ……                       ……
        bne $t0, $s5, Exit       80016: 5 8 21  2             if (t1 != s5) break;
        add $s3, $s3, $s4        80020: 0 19 20 19 0 32       s3 +=s4;
        beq $t0, $s5, Loop                80024: 4 8 21 -7    } while (t0 = = s5);
Exit:                            80028: Exit ...
•       ……
```

```
Loop:   add $t1, $s3, $s3        80000: 0 19 19 9 0 32        Loop:   t1 = s3*2
        ……                       ……
        bne $t0, $s5, Exit       80016: 5 8 21  2             if (t1 != s5) break;
        add $s3, $s3, $s4        80020: 0 19 20 19 0 32       s3 +=s4;
        j Loop    (j 80000)      80024: 2 20000               goto Loop;
Exit:                                    80028: Exit ...
        ……
```

Istruzioni di tipo J: esempio

Nome campo	op	indirizzo
Dimensione	6-bit	26-bit
j 32	000010	00 0000 0000 0000 0000 0000 1000

32 = 100(00)

Nome campo	op	indirizzo
Dimensione	6-bit	26-bit
j 80000	000010	00 00000 0001 0100 1110 0010 0000

80000 = 000 0000 0001 0011 1000 1000 00(00)

Salti incondizionati sopra i 256Mbyte

- Per saltare ad indirizzi superiori a 2^{28} Byte si usa l'istruzione:
  `jr rs` (jump register con **formato R**)
- Salta all'indirizzo di memoria **assoluto** contenuto nel registro **`rs`** (spazio di 2^{32} Word cioè 2^{34} byte = 8 Gbyte > intero spazio di memoria)

Formato istruzioni

- I diversi formati (R, I, J) vengono riconosciuti tramite il valore del primo campo, ***codice operativo* (*opcode*),** che indica alla macchina come trattare i rimanenti bit dell'istruzione.

	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	indirizzo		
J	op	indirizzo				

Sommario

Il linguaggio macchina

Le istruzioni di tipo R

Le istruzioni di tipo I

Le istruzioni di tipo J