

# Lezione 1: L'architettura LC-3

## Laboratorio di Elementi di Architettura e Sistemi Operativi

9 Gennaio 2020

### Ricorda...

Il ciclo di esecuzione di un'istruzione è composto da sei fasi:

FETCH

DECODE

ADDRESS EVALUATION

OPERAND FETCH

EXECUTE

STORE RESULT

### Instruction Set Architecture

Che cos'è l'ISA?

Il termine ISA (Instruction Set Architecture), specifica l'interfaccia tra i comandi software (forniti attraverso un linguaggio di programmazione) e quello che l'hardware è in grado di eseguire (linguaggio macchina).

In particolare specifica:

- l'organizzazione della memoria
- l'insieme dei registri
- l'insieme delle istruzioni:
  - opcode
  - tipi di dato
  - metodi di indirizzamento

Vediamole nel dettaglio...

## Organizzazione della memoria

- La memoria di LC-3 ha uno spazio di indirizzamento di  $2^{16}$  (=65536) locazioni.
- Ogni locazione (word) contiene 16 bit di dati.
- Gli indirizzi sono a 16 bit.
- Gli indirizzi di memoria sono numerati da 0 (0x0000) a 65535 (0xFFFF)

0x0000 0x00FF	Trap Vector Table
0x0100 0x01FF	Interrupt Vector Table
0x0200  0x2FFF 0x3000	Operating System and Supervisor Stack
0xFDFD 0xFE00 0xFFFF	Available for User Program
	Device register addresses

- Non tutti identificano zone di memoria disponibile per l'utente, come riportato in figura.

## Insieme dei registri

- LC-3 mette a disposizione 8 General Purpose Register (GPR);
- i registri sono a 16 bit (word);
- ognuno degli 8 registri (chiamati R0, R1, ..., R7) è identificato da un numero a 3 bit (da R0=000 a R7=111).

Register 0 (R0)	0000111100001111
Register 1 (R1)	0000111111111111
Register 2 (R2)	0000000000001111
Register 3 (R3)	0000110000001111
Register 4 (R4)	0000111100000011
Register 5 (R5)	0000111100001100
Register 6 (R6)	0000111100001001
Register 7 (R7)	0000100000001111

## Insieme delle istruzioni

- Un'istruzione è identificata da DUE componenti:
  - opcode: specifica cosa l'istruzione chiede di fare alla macchina;
  - operandi: specificano i dati su cui la macchina andrà ad operare;
- Ogni istruzione è codificata su 16 bit.
- Esempio: vogliamo fare un'ADD fra i registri R0 e R1 e memorizzare il risultato in R2. L'istruzione sarà:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1
ADD				R2			R0			R1					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	0	1	1	0	0	0	0	0	1
ADD				R2			R3			R1					

- L'ISA di LC-3 definisce 15 istruzioni, ciascuna identificata da un opcode univoco;
- L'opcode è specificato dai bit [12:15] di un'istruzione. Poichè vengono usati 4 bit, si possono identificare 16 istruzioni; LC-3 ne specifica 15, lasciandone libera una (codice 1101).
- Ci sono 3 diversi tipi di istruzioni:
  - operazioni: processano informazioni;
  - spostamento di dati: spostano informazioni tra registri e memoria e viceversa;
  - controllo: cambiano il flusso d'esecuzione delle istruzioni.

## Tipi di dato

- Per tipo *di dato* si intende un modo per rappresentazione l'informazione che sia gestibile dall'ISA
  - ovvero, per la quale l'ISA mette a disposizione istruzioni *in grado di manipolarla*
- Ci sono molti modi per rappresentare un'informazione...
  - caratteri
  - modulo e segno
  - complemento a uno
  - complemento a due
  - virgola mobile
  - ...
- L'ISA di LC-3 utilizza la forma interi in complemento a due

## Metodi di indirizzamento

- Il metodo di indirizzamento è il meccanismo per specificare dove si trova un operando;
- LC-3 supporta cinque modi di indirizzamento:
  - all'interno dell'istruzione stessa (literal o immediate)
  - a registro
  - tre a memoria:
    - PC-relative
    - indiretto
    - base + offset

# Il linguaggio assembly di LC-3

## Abbiamo visto...

- Ogni istruzione è identificata da un insieme di 1 e di 0;
- Ogni locazione di memoria è individuata da un indirizzo a 16 bit.

## Tuttavia...

scrivere programmi sottoforma di 1 e 0 non è certo il massimo. Vediamo quindi come rappresentare in modo più comprensibile i nostri programmi.

## Il linguaggio assembly di LC-3

- è un linguaggio a basso livello;
- ha lo scopo di rendere la programmazione più semplice rimanendo comunque vicino all'ISA;
- vi è una corrispondenza uno a uno tra istruzioni ASM e istruzioni specifiche dell'ISA;
- mette a disposizione nomi simbolici al posto degli opcode delle istruzioni; ad esempio ADD identifica l'opcode 0001;
- permette inoltre di definire nomi simbolici per le locazioni di memoria (symbolic address).

## Istruzioni Assembly

Il formato generale di un'istruzione assembly è:

LABEL	OPCODE	OPERANDS	; COMMENTS
-------	--------	----------	------------

- LABEL: assegna un nome simbolico ad un indirizzo (che può contenere un'istruzione o un indirizzo di memoria).  
È opzionale;
- OPCODE e OPERANDS: sono specifici per ogni istruzione;
- COMMENTS: identifica un commento (come il comando // del C).
- Una label è un nome simbolico che può essere usato per identificare una zona di memoria; viene usata nel programma per fare un riferimento esplicito alla zona di memoria associata; in LC-3 una LABEL può essere lunga al massimo 20 caratteri.
- Come già visto, un'istruzione ha un OPCODE e un certo numero di OPERANDI;
- l'OPCODE:
  - è un nome simbolico che corrisponde ad un'istruzione LC-3;
  - l'idea è che i nomi simbolici (ADD, AND, LDR) sono più facilmente ricordabili rispetto a 0001, 0101, 0110;
- gli OPERANDI: sono specifici per ogni istruzione; ad esempio:
  - BRz AGAIN : se zero, salta all'istruzione AGAIN;
  - ADD R1,R1,#3 : somma 3 al contenuto di R1;
  - Nota: nelle operazioni che richiedono valori costanti, come ad esempio l'istruzione appena vista, si utilizza # per indicare un decimale, x per indicare un esadecimale, e b per indicare un numero binario.
- Nel linguaggio assembly i commenti sono indicati dal ";"
- contengono messaggi in linguaggio naturale che non hanno effetti sul processo di esecuzione;
- ciò che segue il ";" viene completamente ignorato.

Nicola Drago

Elementi di Architettura e sistemi operativi per Bioinformatica

## Le istruzioni di LC-3: operazioni

- Le operazioni sono istruzioni che processano i dati; possono essere di tipo aritmetico (ADD, SUB, MUL, DIV)
- di tipo logico (AND, OR, NOT);
- LC-3 definisce tre operazioni:
  - NOT
  - AND
  - ADD
- Combinandole assieme è possibile ottenere tutte le altre operazioni aritmetiche e logiche.

### NOT

- L'istruzione NOT (opcode = 1001) è l'unica ad utilizzare un solo operando (operazione unaria);
- Utilizza un registro sia per la sorgente che per la destinazione;
- Esegue il complemento bit a bit sui 16 bit del registro sorgente, e memorizza il risultato nel registro destinazione;
- Esempio: si vuole porre nel registro R3 la negazione del registro R5; l'istruzione sarà:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	1	0	1	1	1	0	1	1	1	1	1	1	1
NOT				R3			R5								

- Nota: i bit [0:5] sono tutti 1.

### AND e ADD

- Le istruzioni AND (opcode = 0101) e ADD (opcode = 0001) operano su due operandi a 16 bit;
- L'ADD esegue l'addizione in complemento a due tra gli operandi;
- L'AND esegue l'and bit a bit tra gli operandi;
- A differenza di NOT, il secondo operando sorgente può essere:
  - un registro,
  - un valore costante.
- Nel primo caso il bit [5] vale 0, così come i bit [3:4]. I bit [0:2] contengono il numero del registro.

### Salti non condizionali

- LC-3 mette a disposizione l'istruzione **JMP** (opcode = 1100) per eseguire salti non condizionali;
- JMP carica nel PC il valore contenuto nel registro indicato dai bit [6:8];

Esempio:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
JMP								R2							

# Da codice assembly a codice macchina

## L'assemblatore

Abbiamo visto che il linguaggio assembly permette di scrivere programmi a più alto livello rispetto le istruzioni macchina;

tuttavia un'architettura è in grado di eseguire solo istruzioni in linguaggio macchina: occorre quindi una traduzione;

questo processo di traduzione viene fatto dall'ASSEMBLATORE.

In generale, nel linguaggio assembly vi è una corrispondenza 1 a 1 fra le istruzioni ASM e le istruzioni macchina;

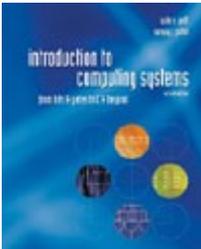
si può quindi pensare di prendere il programma scritto in ASM e, riga per riga, determinare la corrispondente istruzione macchina.

### Esempio:

```
01 .ORIG x3000
02 AND R2,R2,#0 ; reset di R2
...
07 BRp AGAIN
01 x3000: 0101011011100000
...
```

## Riferimenti

*Libro di testo:*



Yale N. Patt, Sanjay J. Patel.  
Introduction to Computing Systems:  
From Bits and Gates to C and Beyond.  
Seconda edizione.  
McGraw-Hill Higher Education, 2003.

<http://www.mhhe.com/patt2>

Contiene il simulatore, il manuale del simulatore e le appendici con l'elenco completo delle istruzioni e la descrizione dell'ISA

## ESERCIZIO 1:

Decodificare le seguenti istruzioni in linguaggio macchina:

0011000001010000: \_X3000: \_ST\_ R0, \_\_\_80\_\_\_\_\_; X3000 + 80

0010001000000111: \_X3001: \_LD\_\_\_\_\_

0010010000000101: \_X3002: \_LD\_\_\_\_\_

0101011011100000: \_\_\_\_\_

0001011011000010: \_\_br np \_\_\_\_\_

0001001001111111: \_\_\_\_\_

0000001111111101: \_\_\_\_\_

1111000000100101: \_\_\_\_\_

0000000000000000: \_\_\_\_\_

0000000000000110: \_\_\_\_\_

(0000) 3050	0011000001010000 ( 4)	.ORIG x3050
(3050) 2207	0010001000000111 ( 5)	LD R1 SIX
(3051) 2405	0010010000000101 ( 6)	LD R2 NUMBER
(3052) 56E0	0101011011100000 ( 7)	AND R3 R3 #0
(3053) 16C2	0001011011000010 ( 11) AGAIN	ADD R3 R3 R2
(3054) 127F	0001001001111111 ( 12)	ADD R1 R1 #-1
(3055) 03FD	0000001111111101 ( 13)	BRP AGAIN
(3056) F025	1111000000100101 ( 15)	TRAP x25
(3057) 0000	0000000000000000 ( 18) NUMBER	.FILL x0000
(3058) 0006	0000000000000110 ( 18) SIX	.FILL x0006

