

# Lezione 5: Architettura dell'elaboratore LC-3

Elementi di Architettura e Sistemi Operativi

Docente: Tiziano Villa

Corso di Laurea in Bioinformatica

Dicembre 2017

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

- Prelievo
- Decodifica
- Calcolo dell'indirizzo
- Prelievo dell'operando
- Esecuzione
- Memorizzazione dei risultati

# Ricorda...

## Che cos'è l'architettura dell'insieme d'istruzioni di un elaboratore (ISA) ?

L'acronimo inglese ISA (Instruction Set Architecture) descrive gli aspetti dell'architettura di un elaboratore che sono visibili al programmatore, quando scrive programmi nel linguaggio della macchina.

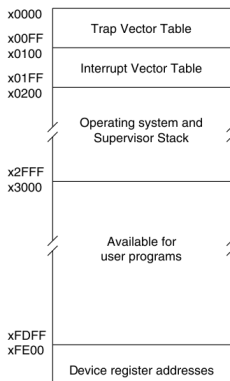
Tali aspetti includono:

- l'organizzazione della memoria
- l'insieme dei registri
- l'insieme delle istruzioni:
  - ▶ codice operativo
  - ▶ tipi di dato
  - ▶ modi di indirizzamento

Vediamoli nel dettaglio...

# LC-3 - Organizzazione della memoria

La memoria di LC-3 ha uno spazio di indirizzamento di  $2^{16}$  (= 65536) locazioni, e indirizzi a 16 cifre binarie. Gli indirizzi di memoria sono numerati da 0 (0x0000) a 65536 (0xFFFF). Gli indirizzi identificano locazioni di memoria, e registri di dispositivi ingresso-uscita mappati sulla memoria. Alcune regioni della memoria sono riservate per usi speciali.



# LC-3 - Insieme dei registri

- LC-3 mette a disposizione 8 registri generali (General Purpose Register, GPR);
- i registri sono a 16 cifre binarie (parola, word);
- l'insieme degli 8 registri (chiamati R0, R1, ..., R7) è identificato da un indirizzo a 3 cifre binarie.

Register 0	(R0)	0000000000000001
Register 1	(R1)	0000000000000011
Register 2	(R2)	0000000000000101
Register 3	(R3)	0000000000000111
Register 4	(R4)	1111111111111110
Register 5	(R5)	1111111111111100
Register 6	(R6)	1111111111111010
Register 7	(R7)	1111111111111000

# LC-3 - Insieme delle istruzioni

- Un'istruzione è identificata da DUE componenti:
  - ▶ **codice operativo:** specifica che cosa l'istruzione chiede di fare alla macchina;
  - ▶ **operandi:** specificano i dati su cui la macchina andrà ad operare;
- Un'istruzione è codificata su 16 cifre binarie. Esempio: vogliamo addizionare i contenuti dei 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		

# LC-3 - Insieme delle istruzioni (OPCODE)

- L'architettura di LC-3 definisce 15 istruzioni, ciascuna identificata da un codice operativo univoco;
- Il **codice operativo** è specificato dalle cifre [12:15] di un'istruzione. Poichè si usano 4 cifre binarie, si possono identificare 16 istruzioni; LC-3 ne specifica 15, lasciandone libera una (codice 1101).
- Ci sono 3 diversi tipi di istruzioni:
  - ▶ *spostamento di dati*: spostano informazioni tra registri e memoria e viceversa;
  - ▶ *operazioni sui dati*: elaborano informazioni;
  - ▶ *controllo*: cambiano il flusso d'esecuzione delle istruzioni.
- In appendice è riportato l'elenco completo delle istruzioni.

# LC-3 - Insieme delle istruzioni (TIPI DI DATO)

- Per *tipo di dato* s'intende una forma di rappresentazione dell'informazione per cui l'architettura definisce dei codici operativi che operano su di essa;
- Ci sono molti modi per rappresentare un'informazione...
- L'architettura di LC-3 utilizza solo la forma **interi in complemento a due**.



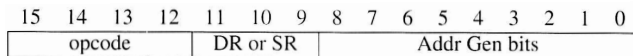
# LC-3 - Insieme delle istruzioni (MODI D'INDIRIZZAMENTO)

- Il *modo d'indirizzamento* è il meccanismo per specificare dove si trova un operando;
- LC-3 definisce cinque modi di indirizzamento:
  - ▶ indirizzo all'interno dell'istruzione stessa (literal o immediate)
  - ▶ indirizzo in registri
  - ▶ indirizzo in memoria (tre modi):
    - ★ relativo al contatore di programma (PC-relative)
    - ★ indiretto
    - ★ base + scostamento (offset)

# Spostamento di dati e modi di indirizzamento

# LC-3 - Insieme delle istruzioni (METODI DI INDIRIZZAMENTO)

- LC-3 definisce 7 istruzioni per spostare dati: LD, LDR, LDI, LEA, ST, STR e STI;
- In generale il formato delle istruzioni di load e store è:



- Nota: le cifre binarie [0:8] si usano per generare l'indirizzo (*l'address generation bits*), cioè rappresentano l'informazione usata per calcolare l'indirizzo a 16 cifre binarie che costituisce il secondo operando dell'istruzione. **Il codice operativo specifica come si devono interpretare queste cifre binarie.**

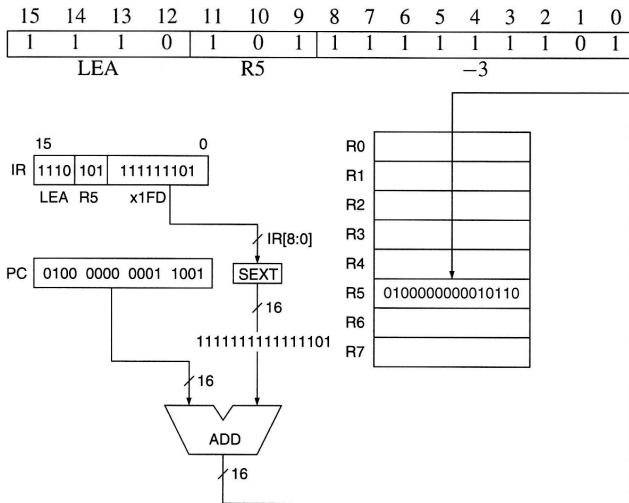
## LC-3 - Indirizzamento immediato (1)

- Si usa *SOLO* con l'istruzione di caricamento effettivo di un indirizzo (Load Effective Address LEA);

L'istruzione LEA (codice operativo = 1110) carica nel registro specificato nelle posizioni [9:11] il valore ottenuto sommando il valore del PC incrementato con il valore indicato nelle posizioni [0:8] dell'istruzione (rappresentato su 16 cifre binarie).

- Di solito l'istruzione LEA è usata per inizializzare un registro con un indirizzo molto vicino all'indirizzo di memoria che si sta usando;
- Esempio: supponiamo che la locazione di memoria 0x4018 contenga l'istruzione "LEA R5, #-3" e il PC+1 contenga 0x4019; dopo l'esecuzione dell'istruzione LEA all'indirizzo 0x4018, R5 conterrà 0x4016.

# LC-3 - Indirizzamento immediato (2)



## LC-3 - Indirizzamento relativo al contatore di programma (PC-relative) (1)

- Le istruzioni **LD** (codice operativo = 0010) e **ST** (codice operativo = 0011) specificano l'indirizzamento relativo al contatore di programma (PC-relative);
- Si chiama così perchè le posizioni [0:8] specificano lo scostamento da sommare al contatore di programma (PC);

L'indirizzo di memoria a cui fa riferimento l'operazione è calcolato sommando al valore del PC incrementato il valore, esteso su 16 cifre binarie, indicato dalle posizioni [0:8] dell'istruzione.

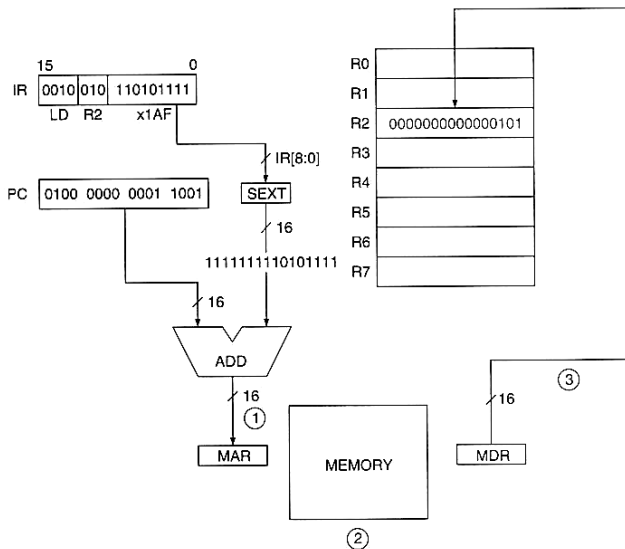
- Se l'operazione è di load, si scrive nel registro indicato dalle posizioni [9:11] il valore contenuto all'indirizzo di memoria calcolato; se è di store, si memorizza all'indirizzo di memoria calcolato il valore contenuto nel registro indicato dalle posizioni [9:11].

## LC-3 - Indirizzamento relativo al contatore di programma (2)

- Esempio: supponiamo che il valore del PC sia 0x4019 e l'istruzione sia "LD R2, x1AF"; in R2 sarà caricato il valore contenuto in memoria all'indirizzo 0x3FC8. *Nota: sono necessari tre passi.*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1
LD				R2			x1AF								

# LC-3 - Indirizzamento relativo al contatore di programma (3)





## LC-3 - Indirizzamento indiretto (1)

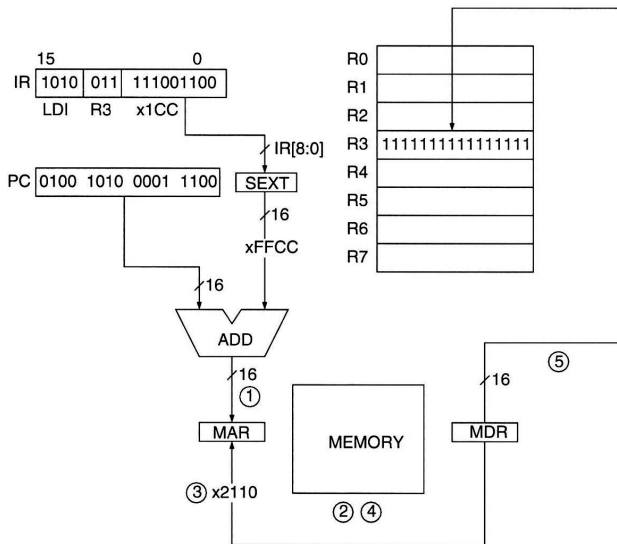
- Le istruzioni **LDI** (codice operativo = 1010) e **STI** (codice operativo = 1011) specificano l'indirizzamento indiretto;
- Funzionano esattamente come le istruzioni LD e ST, ma l'indirizzo calcolato dalla somma tra il PC e il valore specificato nell'istruzione *non è l'indirizzo della parola di memoria da/in cui leggere/scrivere l'operando, **ma è l'indirizzo della parola di memoria che contiene l'indirizzo della parola di memoria da/in cui leggere/scrivere l'operando***;

## LC-3 - Indirizzamento indiretto (2)

- Esempio: supponiamo che l'istruzione in 0x4A1B sia "LDI R3, x1CC", e che l'indirizzo di memoria 0x49E8 (ottenuto sommando il PC incrementato, 0x4A1C, all'estensione su 16 cifre binarie con segno dello scostamento, 0xFFCC) contenga 0x2110; l'esecuzione sarà:
  - ▶ si somma lo scostamento con il PC e si ottiene come risultato l'indirizzo 0x49E8 che si carica nel MAR (fase 1);
  - ▶ si legge la memoria all'indirizzo 0x49E8 e si carica il suo contenuto (0x2110) nel MDR (fase 2);
  - ▶ tale indirizzo (0x2110) e' caricato nel MAR (fase 3);
  - ▶ si legge la memoria all'indirizzo 0x2110 e si carica il suo contenuto 0xFFFF nel MDR (fase 4);
  - ▶ si carica 0xFFFF nel registro di destinazione R3 (fase 5).

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0	0	1	1	1	1	1	0	0	1	1	0	0
LDI				R3				x1CC							

# LC-3 - Indirizzamento indiretto (3)



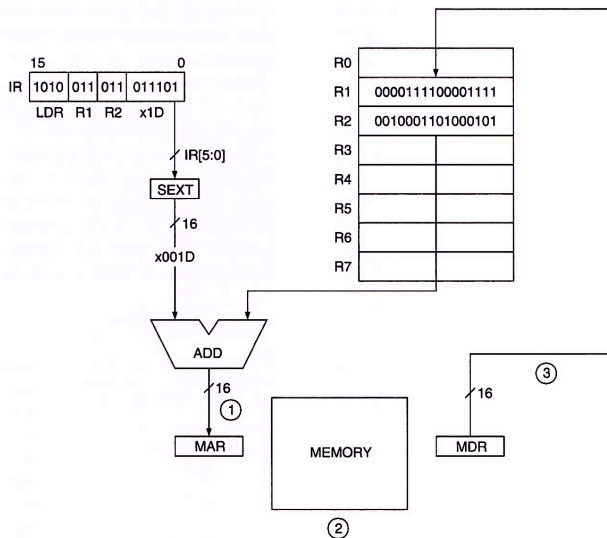
## LC-3 - Indirizzamento base+scostamento (offset) (1)

- Le istruzioni **LDR** (codice operativo = 0110) e **STR** (codice operativo = 0111) specificano l'indirizzamento base+scostamento;

L'indirizzo dell'operando è ottenuto aggiungendo lo scostamento indicato dalle 6 cifre binarie [0:5], esteso a 16 cifre binarie, al registro di base indicato dalle cifre binarie [6:8].

- le 6 cifre binarie utilizzate per rappresentare lo scostamento vanno sempre considerate come interi in complemento a 2 (quindi valori compresi fra -32 e +31);
- Esempio: si supponga che R2 contenga il valore 0x2345, e che l'istruzione da eseguire sia "LDR R1, R2, x1D"; dopo la sua esecuzione, in R1 si avrà il contenuto 0x0F0F della cella di memoria il cui indirizzo 0x2362 è stato calcolato sommando 0x2345 a 0x1D.

# LC-3 - Indirizzamento base+scostamento (offset) (2)



# Operazioni sui dati

- Le operazioni sui dati sono istruzioni che elaborano i dati; possono essere di tipo aritmetico (ADD, SUB, MUL, DIV) o di tipo logico (AND, OR, NOT);
- LC-3 definisce tre istruzioni di questo tipo:
  - ▶ NOT
  - ▶ AND
  - ▶ ADD

## LC-3 - Operazioni sui dati (NOT)

- L'istruzione **NOT** (codice operativo = 1001) è l'unico operatore ad utilizzare un solo operando (operazione unaria);
- Utilizza l'indirizzamento a registro sia per la sorgente che per la destinazione;
- Esegue il complemento cifra a cifra delle 16 cifre binarie specificate nel registro sorgente, e memorizza il risultato nel registro destinazione;
- Esempio: si vuole scrivere 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: le cifre binarie [0:5] sono tutte 1.

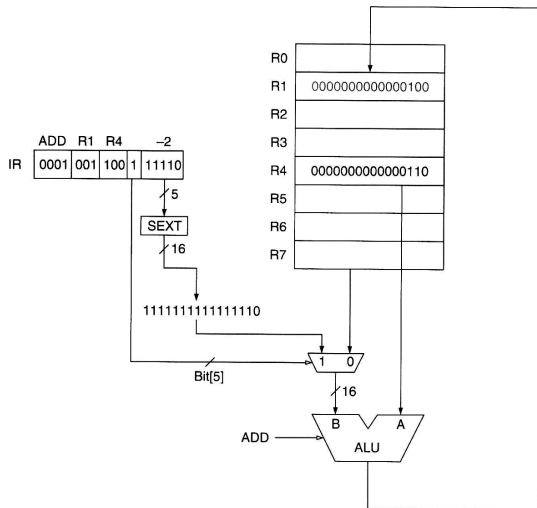


## LC-3 - Operazioni sui dati (AND e ADD)

- Le istruzioni **AND** (codice operativo = 0101) e **ADD** (codice operativo = 0001) operano su due operandi a 16 cifre binarie;
- ADD esegue l'addizione in complemento a due tra gli operandi;
- AND esegue la congiunzione (prodotto) cifra a cifra tra gli operandi;
- A differenza di NOT, il secondo operando sorgente può essere specificato sia utilizzando l'indirizzamento a registro, che immediato. Nel primo caso la cifra [5] è posta a 0, così come le cifre [4:3], mentre le cifre [2:0] indicano il registro; invece nel secondo caso la cifra [5] è posta a 1, e l'operando è contenuto nelle cifre [4:0] dell'istruzione (vedi esempio).

# LC-3 - Operazioni sui dati (AND e ADD)

Esempio di esecuzione dell'istruzione "ADD R1, R4, #-2":



# Controllo

# LC-3 - Controllo

- LC-3 ha 5 operazioni (codice operativo) riservate per il controllo di flusso:
  - ▶ **salto condizionato**
  - ▶ **salto non condizionato**
  - ▶ chiamata a funzione
  - ▶ TRAP
  - ▶ ritorno da un'interruzione (interrupt)
- In questa lezione vedremo solo i salti condizionati e non.

## Nota:

LC-3 ha 3 registri ciascuno costituito da una sola cifra binaria che sono posti a 1 o a 0 ogni volta che si scrive in uno degli otto registri generali. Essi sono:

- Negativo (indicato con N)
- Zero (indicato con Z)
- Positivo (indicato con P)

# LC-3 - Salti condizionati (1)

- L'istruzione di branch è **BR** (codice operativo = 0000);
- Il suo formato è:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	N	Z	P	PCoffset								

- Le cifre binarie [9:11] corrispondono alle tre condizioni N, Z e P; esse sono usati per cambiare il flusso di esecuzione;
- Durante la fase di ESECUZIONE il processore esamina le cifre N, Z, P, e:
  - ▶ se la cifra binaria [11] = 1, si esamina la condizione corrispondente a N;
  - ▶ se la cifra binaria [10] = 1, si esamina la condizione corrispondente a Z;
  - ▶ se la cifra binaria [9] = 1, si esamina la condizione corrispondente a P;

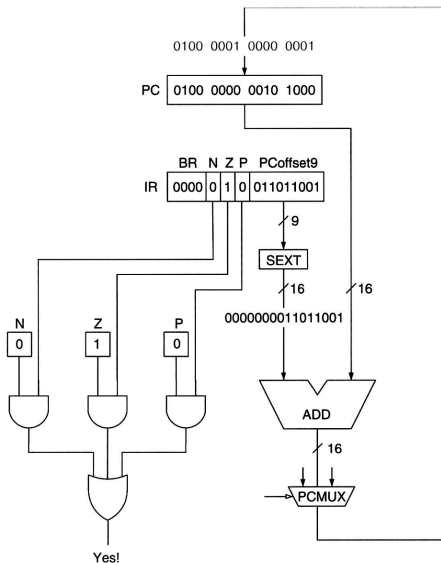
## LC-3 - Salti condizionati (2)

- *Solo se nessuna delle condizioni verificate è vera si lascia immutato il PC;*
- **Altrimenti si riscrive il PC ...**

...  
si aggiorna il PC con il valore calcolato come somma tra il PC incrementato e lo scostamento del PC (PC offset) indicato nell'istruzione.

Esempio: supponiamo che l'istruzione precedente abbia prodotto in un registro generale il valore 0; il PC ha valore 0x4028 e l'istruzione corrente è "BRz x0D9" (0000 0\*1\*0 011011001, Z=1). Dopo l'esecuzione di questa istruzione il PC avrà valore 0x4101 e non 0x4028.

## LC-3 - Salti condizionati (3)



## LC-3 - Salti condizionati (4)

- Se tutte e tre le cifre [11:9] sono 1, si esaminano tutti e tre i codici di condizione, di cui uno sicuramente deve essere uguale a 1. Quindi si aggiorna il PC al nuovo valore calcolato. Questo è un salto incondizionato.
- Se tutte e tre le cifre [11:9] sono 0, non si modifica il PC. Quindi non si salta, ma si prosegue con l'indirizzo presente nel PC.



# LC-3 - Salti non condizionati

- LC-3 mette a disposizione l'istruzione **JMP** (codice operativo = 1100) per eseguire salti non condizionati;
- JMP carica nel contatore di programma (PC) il valore contenuto nel registro indicato dalle posizioni [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							

- Le istruzioni di controllo viste ci permettono di scrivere programmi per LC-3 che contengono costrutti condizionali e cicli;

## Due metodi per il controllo dei cicli:

- ▶ *for*: a priori si conosce il numero di volte in cui si deve eseguire il ciclo; si utilizza quindi un registro contatore (ad esempio R2) che viene decrementato ad ogni ciclo. Esempio: somma dei primi dodici numeri a partire da una locazione di memoria.
- ▶ *while*: a priori non si conosce il numero di volte in cui il ciclo sarà eseguito; si utilizza la tecnica della **sentinella**. Supponiamo ad esempio di voler sommare un insieme di valori tutti positivi (zero escluso) presenti in memoria; al termine di tali valori occorre piazzare una sentinella, che in questo caso potrebbe essere lo zero o un numero negativo. In questo modo si può uscire dal ciclo appena si trova uno zero o un negativo.

- Analizziamo questi esempi...

# LC-3 - Controllo (FOR)

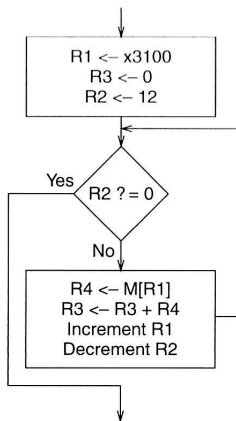
Esempio: somma dei primi dodici numeri memorizzati a partire da una locazione di memoria. Di seguito è riportata una possibile realizzazione nei linguaggi C ed Assembly.

```
int main(){
    int A[20]={1,2,3,4,5,6,7,8,9,10,11,12,0,0,0,0,0,0,0,0};
    int i=0;
    int r=0;
    for(int k=12;k>0;k--){
        r=r+A[i];
        i++;
    }
}
```

```
.ORIG x3000
LEA R1, x0FF ; x3100 indir. A[0] in R1
AND R3,R3,#0 ; R3=0 (variabile r)
AND R2,R2,#0 ; R2=0 (variabile k)
ADD R2,R2,#12 ; k=12
BRz x005      ; salta a x300A
LDR R4,R1,x0 ; contenuto A[i] in R4
ADD R3,R3,R4 ; R3=R3+A[i]
ADD R1,R1,#1 ; i++
ADD R2,R2,#-1 ; k--
BRnzp x1FA ; salta a x3004
.END
```

- LEA R1, x0FF: 0x3001 (PC incrementato) + 0x00FF (scostamento) = 0x3100 (semantica:  $DR \leftarrow (PC+1) + \text{SEXT}(\text{PCoffset})$ )
- BRz x005: 0x3005 (PC incrementato) + 0x0005 (scostamento) = 0x300A
- BRnzp x1FA: 0x300A (PC incrementato) + 0xFFFFA (-6, scostamento) = 0x3004

# LC-3 - Controllo (FOR)



Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
x3000	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1	$R1 \leftarrow x3100$
x3001	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	$R3 \leftarrow 0$
x3002	0	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	$R2 \leftarrow 0$
x3003	0	0	0	1	0	1	0	0	1	0	1	0	1	1	0	0	$R2 \leftarrow 12$
x3004	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	$BRz\ x300A$
x3005	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	$R4 \leftarrow M[R1]$
x3006	0	0	0	1	0	1	1	0	1	1	0	0	0	1	0	0	$R3 \leftarrow R3 + R4$
x3007	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	1	$R1 \leftarrow R1 + 1$
x3008	0	0	0	1	0	1	0	0	1	0	1	1	1	1	1	1	$R2 \leftarrow R2 - 1$
x3009	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0		$BRnzp\ x3004$

# LC-3 - Controllo (WHILE)

Esempio: somma dei numeri maggiori di zero memorizzati a partire da una locazione di memoria. Di seguito è riportata una possibile realizzazione nei linguaggi C ed Assembly.

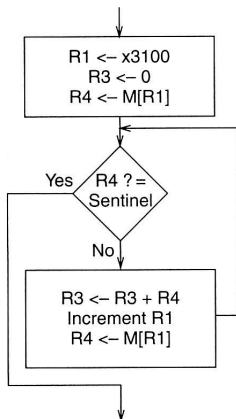
```
int main(){
    int A[20]={1,2,3,4,5,6,7,8,9,10,11,12,-1,0,0,0,0,0,0,0};
    int i=0;
    int r=0;
    while(A[i]>0){ //Nota: -1 è la se
        r=r+A[i];
        i++;
    }
}
```

```
.ORIG x3000
LEA R1, x0FF ; x3100 indir. A[0] in R1
AND R3,R3,#0 ; R3=0 (variabile r)
LDR R4,R1,x0 ; contenuto A[0] in R4
BRn x004 ; salta a x3008
ADD R3,R3,R4 ; R3=R3+A[i]
ADD R1,R1,#1 ; corrisponde a i++
LDR R4,R1,x0 ; contenuto A[i] in R4
BRnzp x1FB ; salta a x3003
.END
```

# LC-3 - Controllo (WHILE)

- LEA R1, x0FF: 0x3001 (PC incrementato) + 0x00FF (scostamento) = 0x3100 (semantica:  $DR \leftarrow (PC+1) + \text{SEXT}(\text{PCoffset})$ )
- BRn x004: 0x3004 (PC incrementato) + 0x0004 (scostamento) = 0x3008
- BRnzp x1FB: 0x3008 (PC incrementato) + 0xFFFFB (-5, scostamento) = 0x3003

# LC-3 - Controllo (WHILE)



Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
x3000	1	1	1	0	0	0	1	0	1	1	1	1	1	1	1	1	R1 ← x3100
x3001	0	1	0	1	0	1	1	0	1	1	1	0	0	0	0	0	R3 ← 0
x3002	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	R4 ← M[R1]
x3003	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	BRn x3008
x3004	0	0	0	1	0	1	1	0	1	1	0	0	0	1	0	0	R3 ← R3+R4
x3005	0	0	0	1	0	0	1	0	0	1	1	0	0	0	0	1	R1 ← R1+1
x3006	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0	0	R4 ← M[R1]
x3007	0	0	0	0	1	1	1	1	1	1	1	1	0	1	1		BRnzp x3003

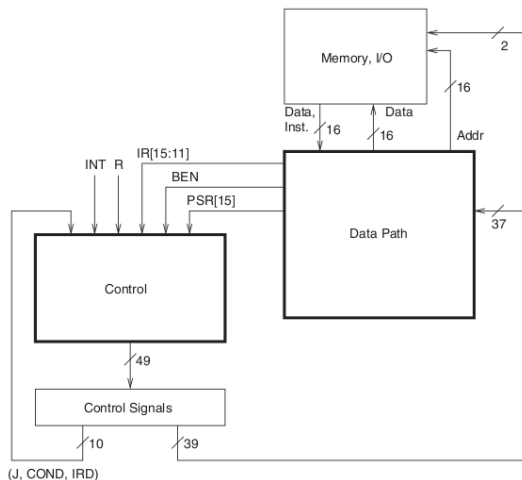


# Approfondimento

- Le due componenti principali di un'architettura sono:
  - ▶ **controllore**: contiene tutte le componenti per generare i segnali di controllo necessari per controllare i componenti logici in ogni ciclo;
  - ▶ **unità esecutiva**: è l'insieme di tutte le componenti che eseguono le istruzioni.
- In questa lezione abbiamo visto le componenti principali dell'unità esecutiva. Vediamo ora la struttura completa dell'unità esecutiva, e un breve cenno al controllore.
- Maggiori dettagli si possono trovare nell'appendice C del testo.

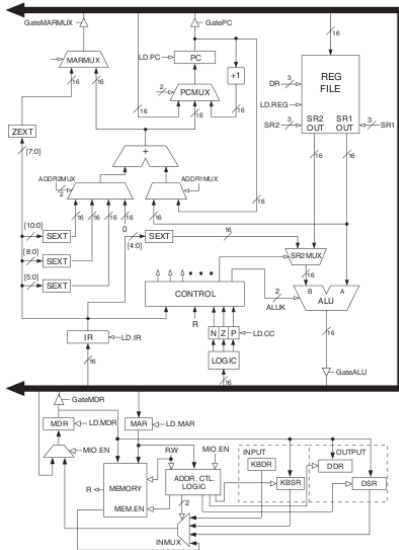
# LC-3 - Approfondimento

- La microarchitettura di LC-3 è la seguente:



- L'unità esecutiva è l'insieme di tutte le componenti che elaborano le informazioni durante il ciclo d'esecuzione;
- Occorre notare che ad ogni componente è associato un segnale di controllo;
- In tutto ci sono 39 segnali di controllo diretto che, ad ogni ciclo di orologio, fanno collaborare le componenti dell'unità esecutiva per processare correttamente i dati;
- Di seguito viene riportato uno schema complessivo dell'unità esecutiva.

## LC-3 - Approfondimento



## LC-3 - Approfondimento

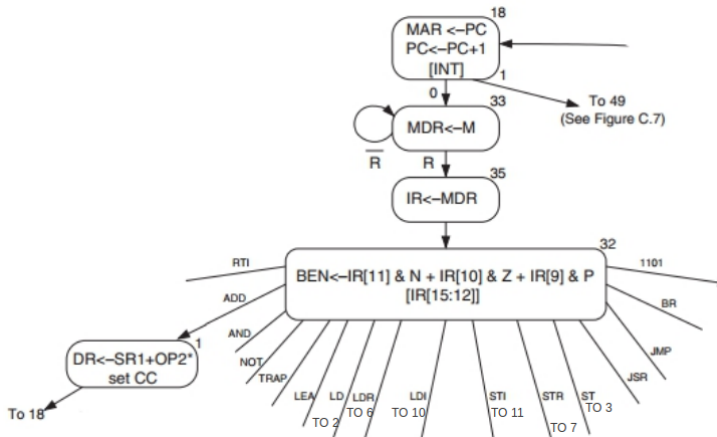
- Il controllore è la componente che stabilisce il comportamento della microarchitettura durante un ciclo di orologio;
- E' implementato da una macchina a stati finiti (MSF, FSM);
- Ogni ciclo di orologio è completamente determinato da 49 segnali (più ulteriori 9); 39 di questi servono per gestire l'unità esecutiva, e 10 segnali servono per determinare lo stato futuro della MSF;
- Complessivamente i 49 segnali si chiamano **microistruzioni**;
- La MSF è composta in tutto da 52 stati distinti, ciascuno dei quali corrisponde ad una microistruzione;

### Nota:

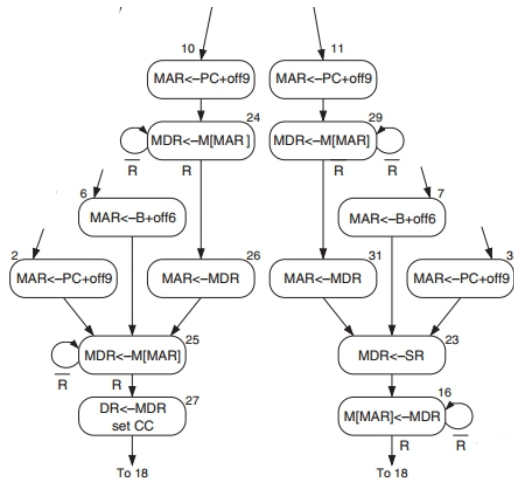
Dalla MSF si ricava quanti cicli di orologio sono effettivamente necessari per eseguire un'istruzione.

# LC-3 - Approfondimento

- Riportiamo di seguito un frammento della MSF:



# LC-3 - Approfondimento





## Elenco completo delle istruzioni di LC-3

# LC-3 - Istruzioni di LC-3

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD	0001				DR			SR1			0	00		SR2		
ADD	0001				DR			SR1			1	imm5				
AND	0101				DR			SR1			0	00		SR2		
AND	0101				DR			SR1			1	imm5				
BR	0000				n	z	p	offset9								
JMP	1100				000			BaseR			000000					
JSR	0100				1	offset11										
JSSR	0100				0	00		BaseR			000000					
LD	0010				DR			offset9								
LDI	1010				DR			offset9								
LDR	0110				DR			BaseR			offset6					
LEA	1110				DR			offset9								
NOT	1001				DR			SR			1111111					
RET	1100				000			111			000000					
ST	0011				SR			offset9								
STI	1110				SR			offset9								
STR	0111				SR			BaseR			offset6					
TRAP	1111															
reserved	1101															
reserved	1000															