

# Appunti di Architettura degli Elaboratori

Andreas Araya Osorio

November 4, 2021

## 1 Rappresentazione dei numeri

I numeri possono essere rappresentati in qualsiasi base, noi utilizziamo la base 10 come conseguenza del numero delle nostre dita.

I **calcolatori** utilizzano la base 2, ovvero il binario, che può facilmente essere ricondotta alla condizione di uno stato elettrico o **positivo (1)** o **negativo (0)**

Per la rappresentazione di un numero in qualsiasi base:

**Definizione 1:**

*La sequenza di  $k$  cifre:*

$$d_k \cdot d_{k-1} \cdot \dots \cdot d_1 \cdot d_0 \quad (1)$$

*E questa sequenza la moltiplichiamo per la base scelta:*

$$d_k \times b^k \cdot d_{k-1} \times b^{k-1} \cdot \dots \cdot d_1 \times b^1 \cdot d_0 \times b^0 \quad (2)$$

*dove  $b$  è la base da noi scelta.*

**Definizione 2** (Numero di cifre in una base  $N$ ):

*In una qualsiasi base  $N$  il numero di cifre equivale a:*

$$cifre = N - 1, N - 2, \dots, 1, 0. \quad (3)$$

### 1.1 Binario

Con la base 2 abbiamo le cifre dallo 0 allo 1.

Per la definizione precedente la rappresentazione sarà:

$$d_k \times 2^k \cdot d_{k-1} \times 2^{k-1} \cdot \dots \cdot d_1 \times 2^1 \cdot d_0 \times 2^0 \quad (4)$$

**Definizione 3** (Valore minimo e massimo):

*Il valore minimo e massimo di un numero di  $n$  cifre è:*

- **Valore minimo** = 000000...00( $n$  times) = 0
- **Valore massimo** = 111111...11( $n$  times) =  $2^n - 1$

$$2^{n-1} + 2^{n-2} + \dots + 2^2 + 2^1 + 2^0 = 2^n - 1 \quad (5)$$

ESEMPIO 1.

$$n = 3 \implies 111 = 2^2 + 2^1 + 2^0 = 7 = 2^3 - 1 = 8 - 1 \quad (6)$$

## 1.2 Notazione

**Definizione 4** (BIT):

**Bit** = binary digit, uno dei due simboli (0, 1) del sistema numerico binario.

*Esso è l'unità elementare dell'informazione trattata da un elaboratore.*

*Numeri di 8, 16, 32 bit equivale in base 10 a parlare di numeri a 3, 4, 5, ... cifre*

**Definizione 5** (BYTE):

**Byte** = 8 bit, è una sequenza di bit, convenzionalmente l'unità di misura delle capacità di una memoria.

*Può assumere  $2^8 = 256$  (0 - 255) possibili valori*

**Definizione 6** (Parola):

**Word/Parola** = corrisponde a 16, 32 o 64 bit in base al tipo di IS (Instruction Set), essa è l'unità più piccola di informazione su cui un elaboratore può intervenire.

Nome	Simbolo	Multiplo in base 10	Multiplo in base 2
chilobyte	kB	$10^3$	$2^{10}$
megabyte	MB	$10^6$	$2^{20}$
gigabyte	GB	$10^9$	$2^{30}$
terabyte	TB	$10^{12}$	$2^{40}$
petabyte	PB	$10^{15}$	$2^{50}$

Convenzionalmente si utilizzano come unità di misura:

- il B(yte) per la capacità di una memoria
- il b(it)/s per la velocità di trasmissione di dati.

### 1.3 Decimale

Nella rappresentazione decimale abbiamo le cifre dallo 0 al 9.

Un qualsiasi numero in base decimale si può rappresentare come:

$$d_k \times 10^k \cdot d_{k-1} \times 10^{k-1} \dots \dots d_1 \times 10^1 \cdot d_0 \times 10^0 \quad (7)$$

Binario	Esadecimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9

### 1.4 Ottale

Nella rappresentazione ottale si hanno le cifre dallo 0 al 7.

Questa base viene utilizzata perchè essendo un multiplo di 2 si possono facilmente convertire numeri ottali in binario:

Con 3 cifre si possono rappresentare 8 bit. In questo modo si possono avere numeri più facilmente maneggiabili da umani, rispetto a lunghe stringhe di 0 o 1.

Binario	Ottale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

### 1.5 Esadecimale

Nella rappresentazione esadecimale si hanno 15 cifre: dallo 0, ..., 9, A, ..., F.

Un simbolo (cifra) in questa base rappresenta 4 cifre binarie (4 bit).

Con 4 cifre esadecimali si possono rappresentare 16 bit.

Binario	Esadecimale
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

## 1.6 Algebra di Boole

Essa viene utilizzata per la specifica di funzioni logiche.

Una qualsiasi variabile può assumere 2 valori: **vero** o **falso**

**Definizione 7** (Operazioni logiche di base):

$$\begin{aligned}
 A \text{ AND } B &= A \cdot B \\
 A \text{ OR } B &= A + B \\
 \text{NOT } A &= \overline{A}
 \end{aligned}
 \tag{8}$$

$A$	$B$	$A \text{ AND } B$	$A \text{ OR } B$	$\text{NOT } A$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Si indica con:  $\oplus = xor$ , è l'or esclusivo, esso è vero solo quando una delle due variabili è vera, non quando lo sono entrambe.

$\overline{A \cdot B} = nand$ , l'opposto dell'AND classico

Grazie a XOR e NAND si possono rappresentare tutte le altre funzioni logiche attraverso delle combinazioni di questi due.

A	B	$\overline{A}$	$A \cdot B$	$A + B$	$\overline{A \cdot B}$	$\overline{A + B}$	$A \oplus B$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	1	0	1
1	1	0	1	1	0	0	0

## 2 APPROACH

There's a difference between

**Computer architecture** and **Computer organization**.

- C.Architecture = attributes of a system visible to a programmer.  
**ISA** = Instruction set architecture, is a synonym of C.A.
- C.Organization = operational units and their interconnections that realize the architectural specifications. Examples:
  - instruction set
  - number of bits used to represent various data types
  - I/O mechanisms and techniques for addressing memory

### 2.1 Structure and function

A modern computer is a hierarchical system, and is a set of interrelated subsystems. These have, in turn, have subsystems of their own until we reach the lowest level of subsystems. There's a huge difference between:

- **structure** the way components are interrelated
- **function** the operation of each individual

The way that is considered most efficient and the clearest approach for describing pc components is: **TOP-DOWN**

### 3 Componenti principali

Le componenti principali di un elaboratore sono:

- CPU
- Memoria
- Sistemi di I/O
- Interconnessioni

**Definizione 8** (Architettura di Von Neumann):

*Secondo questo tipo di architettura, un elaboratore è composto di questi principali componenti:*

- *dati e istruzioni in memoria*
- *memoria accessibile per indirizzo*
- *esecuzione sequenziale delle istruzioni*

**Definizione 9** (Programma cablato):

*Conaiste nel costruire i componenti logici in modo tale che il risultato sia quello voluto e non può essere modificato in seguito.*

*Vuol dire "programmare" a livello hardware, ovvero con le componenti fisiche. Non è un sistema flessibile, esegue solo operazioni predeterminate.*

**Definizione 10** (Programma):

*Un programma è una sequenza di passi*

*Ogni passo corrisponde ad un'operazione logica.*

*Ogni operazione determina un diverso insieme di segnali di controllo.*

**Definizione 11** (Programmazione software):

*Nasce con Von Neumann, si parte da un hardware generico, si ha una parte che preleva il codice di una istruzione, è generale: L'hardware di cui parliamo si dice **general purpose**, utile a vari scopi. Si hanno poi dei segnali di controllo corrispondenti.*

*Questo sistema è molto più flessibile di quello "cablato".*

*La CPU assume delle funzioni diverse ovvero:*

- *interprete delle istruzioni*
- *generico modulo per operazioni aritmetico logiche = ALU*

*I segnali di controllo sono necessari per far eseguire al giusto modulo la giusta operazione: ALU*

*ALU prende segnali di controllo ed esegue le istruzioni codificate*

*In questo sistema si ha la codifica delle istruzioni e la decodifica delle istruzioni.*

**Definizione 12** (Memoria principale nell'architettura di Von Neumann):

*Si ha la possibilità di salti oltre che all'esecuzione sequenziale(in serie)*

*Per esempio con le operazioni che richiedono accesso a più dati in memoria nello stesso momento.*

*Inoltre essa ha il compito di immagazzinare dati e istruzioni*

ESEMPIO 2.

Somma con 2 numeri in locazione di memoria diverse

### 3.1 CPU

Essa non deve solo eseguire istruzioni ma anche gestire dei segnali di controllo e gestire delle risorse.

Composta da Vari componenti principali:

- **btEU** = execution unit = alu
- **btIR** = instruction register, registro che contiene l'istruzione da eseguire successivamente a quella nel PC.
- **PC** = program counter, puntatore all'istruzione indirizzo dell'istruzione da eseguire presente nella memoria.
- **MAR** = memory address register, registro di interfaccia con il bus di sistema, contiene solo registri
- **MBR** = memory buffer register, contiene solo dati.  
Il MAR e il MBR mantengono le informazioni fino a che non è disponibile il bus di sistema per essere impiegato.
  - in caso di lettura raccolgono il dato dal bus.
  - in caso di scrittura contengono il dato.
- **I/O AR** indirizzo periferica con cui scambiare dati, specificare periferica
- **I/O BR** raccolta dati

Quando si ha un salto nell'esecuzione delle istruzioni incrementa l'indirizzo del PC

Inoltre è presente un buffer nel modulo I/O: è una memoria interna al sistema di input output, esso serve perchè la CPU invia dati troppo velocemente rispetto ed esso non può riceverli alla stessa velocità, per questo il buffer dell'I/O, mantiene in memoria i dati inviati dalla più veloce CPU

### 3.2 Ciclo CPU esecuzione

- **Fetch:** reperimento, prelievo dell'istruzione dalla memoria
- **Execute:** esecuzione dell'istruzione prelevata dalla memoria

Il registro **PC** contiene l'indirizzo di memoria della cella di Memoria contenente l'istruzione da eseguire. Quando si ha un prelievo di istruzioni dalla memoria, si ha un incremento del PC.

L'istruzione prelevata viene messa in IR poi viene eseguita.

### 3.3 Tipi di Operazioni

1. Processore-memoria: trasferimento dati dalla CPU alla Memoria R/W
2. Processore-I/O: trasferimento dati da CPU a I/O R/W
3. Elaborazione dati: operazioni logiche e aritmetiche sui dati operazioni della ALU
4. Controllo: può alterare la sequenza delle istruzioni, per esempio il salto

ESEMPIO 3.

Parola = 16bit

Istruzione = 16bit

Codici operativi = 4 bit a sinistra =  $2^4$  combinazioni = 16

- 0001 carica in AC (accumulatore) una cella di M
- 0010 scrive in M il contenuto di AC
- 0101 somma una cella di M ad AC



### 3.4 Ciclo di esecuzione

1. Instruction Address Calculation
2. Instruction Fetch
3. Instruction operation decoding
4. Operand address calculation
5. Operand Fetch
6. Data operation
7. Operand address calculation
8. Operand Store

Per l'esecuzione di una singola operazione, bisogna prima eseguire una serie di altri sottocompiti, che devono essere ben eseguiti.

Per questo è essenziale **l'unità di controllo**, per accertarsi che ogni operazione venga eseguita correttamente e nel giusto ordine.

La CPU può eseguire **più operazioni** momentaneamente, mantenendo ogni sua parte attiva , breve introduzione al concetto di pipeline

### 3.5 Interruzioni

Il meccanismo tramite il quale dei moduli possono interrompere la normale di sequenza di esecuzione.

Il ciclo di esecuzione con interruzioni è differente: al **termine** di un ciclo si ha il controllo delle interruzioni,

- se ce ne sono, esse vengono risolte
- se non ce ne sono, il ciclo ricomincia

Tipi di interruzioni:

- Program, overflow, divisione per zero
- Timer, da un timer interno alla CPU
- I/O, termine di un'operazione di I/O
- Guasto Hardware

Si interrompe per

- efficienza elaborazione

Ciclo interruzione:

- viene aggiunto al ciclo di esecuzione
- la cpu controlla (fetch) le interruzioni pendenti
- se non ce ne sono, prende la prossima istruzione
- se ce ne sono:
  - sospende esecuzione
  - salva contesto
  - imposta il pc all'indirizzo di inizio del programma di gestione
  - esegue il programma di gestione dell'hardware
  - rimette il contesto al suo posto e continua il programma interrotto

Lunghezza di attesa:

- **breve attesa**, tempo di operazione di I/O minore del tempo tra due istruzioni WRITE
- **lunga attesa**,

### 3.5.1 Interruzioni multiple

In caso di **interruzioni multiple**: esistono vari livelli di interruzione e differenti tipi di politica di gestione.

Esistono interruzioni di alto livello e basso livello, a seconda dell'importanza che hanno per il funzionamento del sistema.

**Politiche di interruzione:**

- Disabilitare le interruzioni:
  - La CPU **ignora** le altre interruzioni e gestisce la prima e "maschera" le altre
  - Le interruzioni rimangono **pendenti**
  - vengono gestite nell'**ordine** in cui arrivano
- Definire le priorità

- Interruzioni di bassa priorità vengono interrotte quando si presentano quelle di alta priorità
- Quando è stata gestita la priorità di alto livello viene maneggiata quella di basso livello
- Si hanno delle interruzioni **annidate**

### 3.6 Connessioni

Tutti i componenti **devono** essere connessi

Esistono vari tipi di connessioni per vari tipi di componenti

- CPU
- Memoria
- I/O

### 3.7 Bus

Tutti i dispositivi sono collegati dal bus di sistema

Il bus:

1. collega **2 o più** dispositivi
2. mezzo trasmissione condiviso
3. un segnale trasmesso ad un bus è disponibile a tutti i dispositivi
4. arbitro bus: solo un dispositivo alla volta può trasmettere
5. varie linee di comunicazione ( trasmettono uno 0 o un 1)
6. varie linee trasmettono in parallelo numeri binari. Un bus da 8 bit trasmette un dato di 8 bit

#### 3.7.1 Bus di sistema:

- connette cpu, i/o, M
- da 50 a qualche centinaio di linee
- 3 gruppi di linee
  1. bus dati
  2. indirizzi
  3. controllo

### 3.7.2 Bus dati:

- trasporta dati o istruzioni
- ampiezza  $\rightarrow$  efficienza del sistema
  - con poche linee  $\rightarrow$  accessi in memoria

### 3.7.3 Bus indirizzi

- indica sorgente o destinazione dati
- l'ampiezza determina la massima quantità di M indirizzabile

### 3.7.4 Bus controllo

- per controllare accesso, uso linee dati e indirizzi
  1. M write
  2. M read
  3. richiesta bus
  4. bus grant
  5. interrupt request
  6. clock

Bus usage: se un modulo vuole inviare dati ad un altro:

- bus grant
- data transfer

se un module vuole ricevere dati da un altro:

- bus grant
- trasferire una richiesta all'altro modulo sulle linee di controllo
- attendere invio dati

### 3.7.5 Bus singoli e multipli

- singolo bus = ritardo e congestione
- vari bus = risoluzione problema

### 3.8 Temporizzazione

- Coordinazione degli eventi su un bus
- Sincrona
  - clock determined events
  - single clock line
  - single sequence is a clock cycle
  - every device connected to the bus can read the clock line
  - every event starts at the beginning of a clock cycle

### 3.9 Memoria

Tutte le locazioni di memoria sono suddivise in blocchi.

La memoria è suddivisa in 2 tipi differenti:

- Cache la più veloce e suddivisa in diversi livelli
  - L1 cache
  - L2 cache
  - L3 cache
- Ram più lenta della cache ma più capiente

La memoria Ram è composta da:

1. indirizzo di memoria
2. blocco di memoria

Il numero di parole in un blocco è una potenza di 2.

Una parola è composta da 4 byte, possiamo identificare i primi 14 bit come "indirizzo" del bit, mentre i restanti 2 come identificativi del bit.

#### 3.9.1 Gerarchia di memoria

Un blocco di memoria richiesto dalla CPU può essere presente **hit** o non presente **miss** in memoria. (generalmente è presente).

$T_a$ : Tempo medio di accesso ad un dato in memoria cache

$$T_a = T_h \times P_h + T_m(1 - P_h) \quad (9)$$

$T_h$ : tempo di accesso ad un dato presente in cache  $T_m$ : tempo medio di accesso ad un dato **non** in cache (dimensione blocco)  $P_h$ : probabilità di hit

*Tecnica generale*

1. Suddivisione della memoria centrale in blocchi logici
2. dimensionamento della cache in multiplo di blocchi
3. ogni indirizzo emesso dalla cpu
  - hit  $\iff$  il dato viene fornito immediatamente alla cpu
  - miss
    - (a) la cache richiede il dato al livello inferiore
    - (b) viene posto in cache
    - (c) viene fornito alla cpu

**Definizione 13 (associazione diretta / direct mapping):**

*Ogni blocco del livello inferiore può essere allocato solo in una specifica posizione **linea/slot** del livello superiore*

1. ***ILS*** = indirizzo di livello superiore
2. ***ILI*** = indirizzo di livello inferiore
3.  $ILS = ILI \bmod N$

1. *vantaggi*

- semplicità traduzione indirizzo *ILI* a *ILS*
- determinazione velocità hit o miss

2. *svantaggi*

- necessità di contraddistinguere blocco in *ILS*
- swap frequenti per accesso a dati di blocchi adiacenti

**Definizione 14 (associazione completa / fully associative):**

*Ogni blocco del livello inferiore può essere posto in qualunque posizione del livello superiore.*

*Ad una cache di  $N$  blocchi viene associata una tabella di  $N$  posizioni contenenti il numero di blocco effettivo (tag)*

- *vantaggi: massima efficienza di allocazione*
- *molto tempo per la corrispondenza ILS-ILI e della verifica hit/miss*

**Definizione 15 (associazione a N-gruppi / N-way set associative):**  
*Ogni blocco di un certo insieme di blocchi del livello inferiore può essere allocato liberamente in uno specifico gruppo di blocchi del livello superiore*

ESEMPIO 4.

Per una cache di 32 linee con un  $N$  equivalente a 2, ogni gruppo avrà 16 linee.

*Questo tipo di associazione è una via di mezzo fra gli altri due tipi. La cache composta da  $R$  gruppi di  $N$  posizioni di blocco, si affiancano  $R$  tabelle di  $N$  elementi contenenti i tag. Ha una buona efficienza di allocazione, nonostante abbia una certa complessità*

**Definizione 16 (Politiche di rimpiazzo dei blocchi):**

*Quando si ha un miss, come si decide quale blocco della cache dobbiamo rimpiazzare? Nell'associazione diretta non ci si pone questo problema, perchè ogni linea della cache corrisponde un blocco della memoria centrale.*

1. *casuale, viene occupato lo spazio omogeneamente, facile implementazione*
2. *First-In-First-Out(FIFO), il blocco rimasto più a lungo in cache, complicata implementazione*
3. *Least Frequently Used(LFU), il blocco con meno accessi, complicata implementazione hardware*
4. *Least Recently Used(LRU), il blocco con l'accesso più distante, per preservare quelli accessi più recentemente, implementazione difficile.*

*A minor quantità di cache si hanno migliori prestazioni con il rimpiazzo LRU. Ad aumentare il livello di cache è sempre meno significativo il miglioramento offerto da queste tecnologie.*

La scrittura dati determina incoerenza tra il blocco in cache e quello nei livelli inferiori

**Definizione 17 (write through):**

1. *scrittura contemporanea in cache e livello inferiore*
2. *aumento traffico per frequenti scritture nel medesimo blocco, dati coerenti fra blocchi*

3. *si ricorre a buffer asincroni verso la memoria*

**Definizione 18** (write back): 1. *scrittura in memoria inferiore differita al rimpiazzo del blocco di cache corrisp.*

2. *occorre ricordare operazioni di scrittura nel blocco*

3. *ottimizzazione del traffico tra livelli*

4. *periodi di incoerenza*

Occorre ricordare che tra memoria centrale (RAM) e cache si passano **BLOCCHI** e non **PAROLE**.

ESEMPIO 5 (scenario problematico). • più dispositivi connessi allo stesso bus con cache locale

- memoria centrale condivisa

Nessun tipo di "write" (through, back) può assicurare coerenza.

Possibili soluzioni

- **monitoraggio del bus con write through**, controllori intercettano modifiche locazioni condivise
- **trasparenza hardware**, hardware aggiuntivo: modifica a RAM = modifica a cache
- **memoria non cacheable**, solo una porzione è condivisa e non cacheable

Si può estendere il discorso fatto a livelli più alti prendendo in considerazione come memorie la ram e la memoria di tipo swap, basata sui dischi di archiviazione.