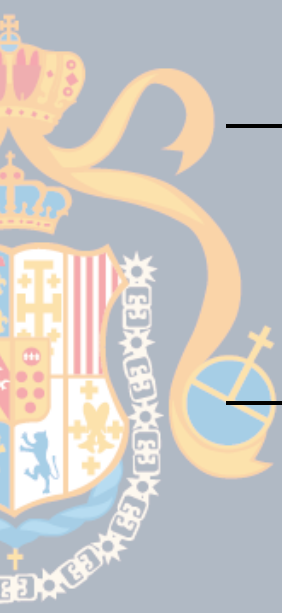

ARCHITETTURA DEGLI ELABORATORI

Docente

Diletta Cacciagrano

Studente

Giorgio Della Roscia

Università degli studi di Camerino

Informatica per la Comunicazione Digitale (L-31)

Scuola di Scienze e Tecnologie

A.A. 2022/2023

Indice

Figure

Tabelle

Approfondimenti

Sezione 1

STORIA DEI CALCOLATORI

Il calcolatore è visualizzabile come una macchina a livelli.

Tabella 1.1: Livelli di un elaboratore

N	LEVEL
5	problem-oriented language
4	assembly language
3	operating system machine
2	instruction set architecture
1	microarchitecture
0	digital logic

La separazione indica che i primi due livelli sono di organizzazione, mentre gli altri di architettura.

Tabella 1.2: Programmazione HW e SW

ARCHITETTURA	ORGANIZZAZIONE
software	hardware
portabilità	performance

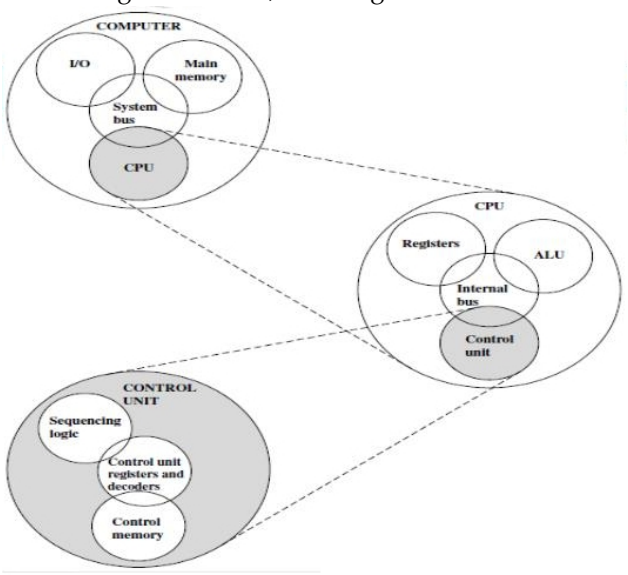

Figura 1.1: Livelli fisici del computer

1.1 Evoluzione dei sistemi di calcolo

Si possono distinguere due macrofasi:

1. era meccanica
   La velocità di elaborazione viene limitata dall'inerzia meccanica. Inoltre era presente un'elevata difficoltà d'uso, una scarsa affidabilità e dei costi elevati.

   > **Esempio** (tecnologia era meccanica). Due invenzioni simbolo di quest'epoca sono state:
   >
   > – Pascalina - 1642 (Blaise Pascal)
   > – stepped reckoner - 1673 (Gottfried Wilhelm Leibniz)
   >
   > La prima effettuava somme algebriche mentre la seconda spaziava anche alle moltiplicazioni e divisioni.

2. era elettronica
   Von Neumann introdusse il calcolatore IAS a base binaria.

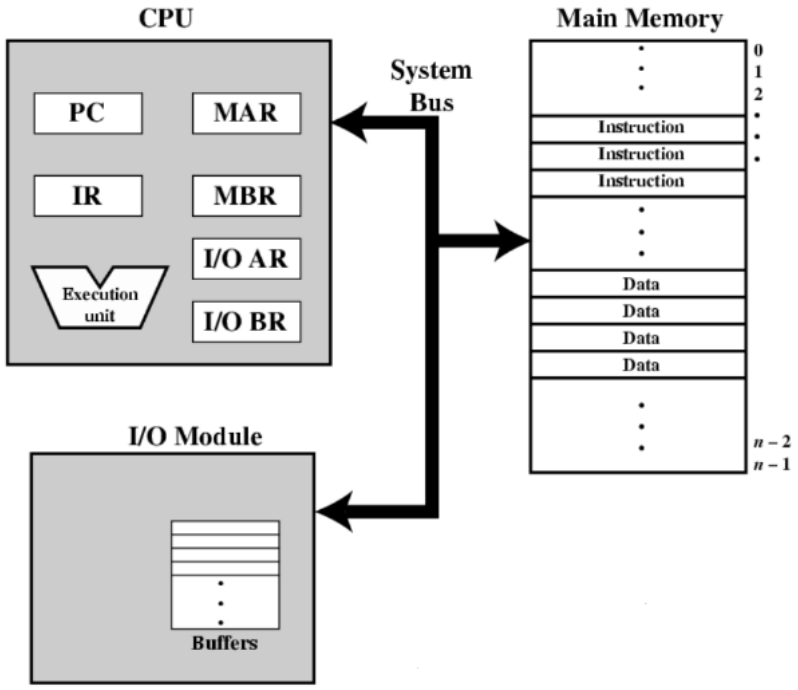

Figura 1.2: Architettura di Von Neumann

1.1.1 Progressi della tecnologia

Il cuore dell'avanzamento ricade sul numero di transistor inseribile su un chip poiché la vicinanza indica velocità.

> **Tesi** (Moore). Gordon Moore pronunciò la seguente frase:
>
> *«Il numero di transistor su chip raddoppierà ogni anno»*
>
> Ciò è stato quasi rispettato poiché l'aumento annuale è del 60%.

Ora la velocità di un chip si misura tramite i GFlops, ossia il numero di istruzioni in virgola mobile eseguite.

1.2 Componenti hardware

Dagli schemi precedenti abbiamo capito che la struttura base di un elaboratore è data da: cpu, memoria, i/o e bus.

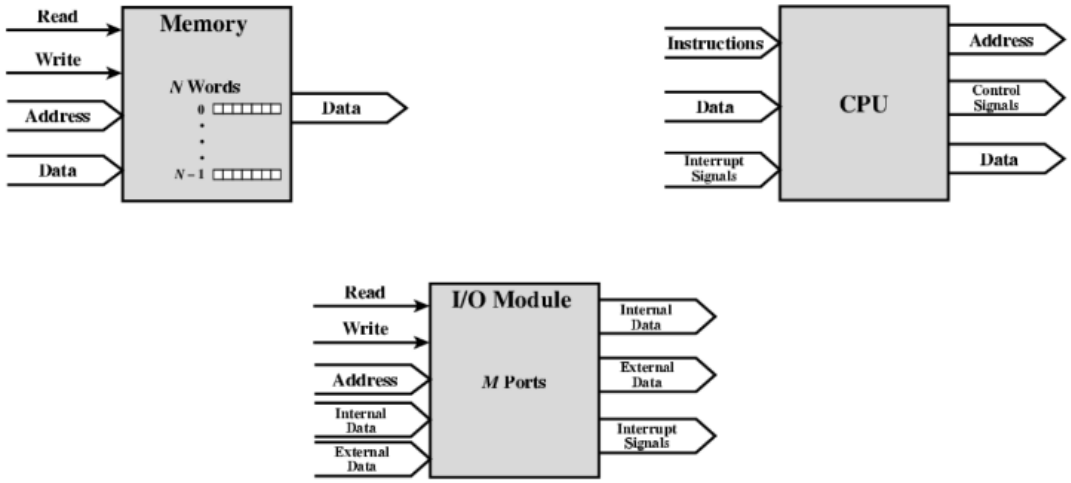

Figura 1.3: Ingessi e uscite dei moduli dell'elaboratore

1.2.1 Bus

Il collegamento fra più dispositivi è garantito da un mezzo di trasmissione condiviso. I segnali che ci viaggiano sono disponibili a tutti e la comunicazione avviene su più canali paralleli.

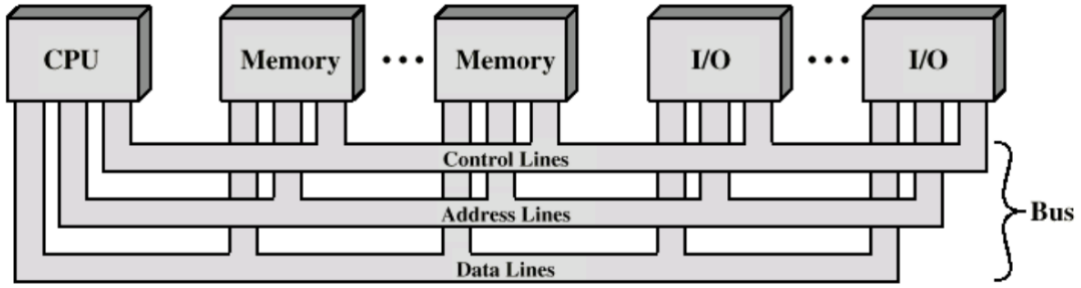

Figura 1.4: Struttura del bus

Funge ad uso esclusivo per ogni modulo quindi per inviare dati si deve ottenere prima l'uso del bus.

1.2.1.1 Arbitraggio del bus

Questo strumento collega diversi elementi; questi si separano in:

a. master, controlla il bus;

b. slave, richiedono l'accesso.

Normalmente il ruolo di master è gestito dalla cpu. I meccanismi di arbitraggio sono:

i. centralizzato;

ii. distribuito.

1.2.1.1.1 Arbitraggio del bus centralizzato

Tramite un'apposita unità funzionale i dispositivi richiedono l'uso del bus e la priorità scende in base alla lontananza dalla prima;

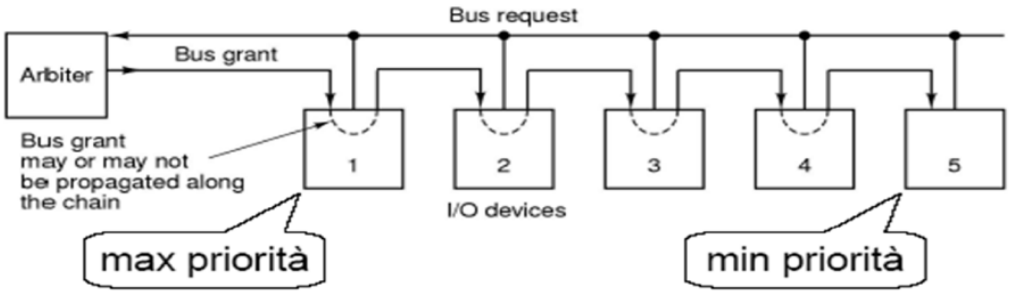

Figura 1.5: Bus centralizzato a festone

L'estensione di questo sistema aggiunge un livello di controllo per gestire le richieste.

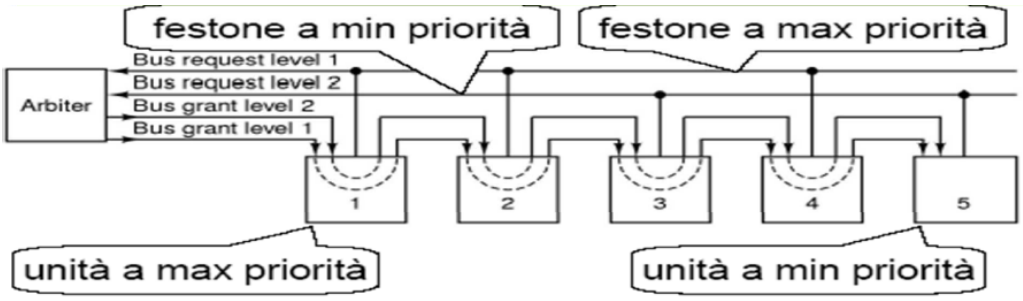

Figura 1.6: Bus centralizzato con richieste

1.2.1.1.2 Arbitraggio del bus distribuito

Il meccanismo di gestione è diffuso su varie linee, ognuna delle quali è accedibile dalla sua singola unità. Le priorità variano in base all'unità.

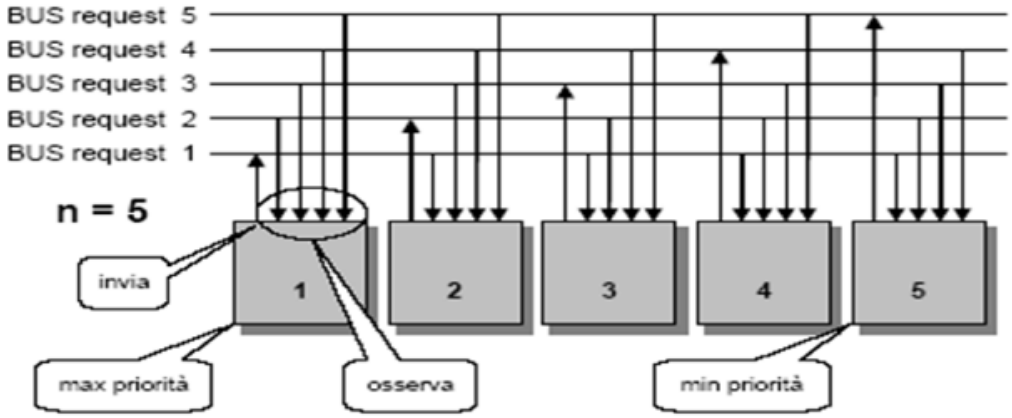

Figura 1.7: Bus distribuito

1.2.1.2 Temporizzazione del bus

Le coordinazione di eventi sul bus è gestita in due modalità di temporizzazione:

- sincrona, si segue il clock di sistema;
- asincrona, l'evento precedente determina l'occorrenza del successivo.

Nei cambi di stato del clock, da zero a uno o viceversa, la variazione non è immediata ma flette leggermente dati i limiti fisici.

1.3 Prestazioni

La valutazione delle performance hardware riguarda i tempi.

$$\boxed{textprestazioni_x > \text{prestazioni}_y \rightarrow T_x > T_y \rightarrow \frac{1}{T_x} > \frac{1}{T_y}}$$

Il numero di clock impiegati per eseguire un'istruzione è dato da:

$$\boxed{CPI = \frac{n_{\text{cicli}}}{n_{\text{istruzioni}}}}$$

Il tempo di CPU è strettamente legato al concetto precedente:

$$\boxed{T_{CPU} = CPI \cdot n_{\text{istruzioni}} \cdot T_{\text{clock}}} \qquad \boxed{T_{CPU} = t_{\text{medio}} \cdot n_{\text{istruzioni}}}$$

Esercizio (calcolo CPI). Calcolare il valore di CPI di un programma di 400.000 istruzioni. L'esecuzione necessita di $T_{CPU} = 1,2s$ e l'elaboratore ha frequenza $1MHz$.

$$n_{\text{cicli}} = T_{CPU} \cdot f_{\text{clock}} = 1,2s \cdot 1 \cdot 10^6 \frac{1}{s} = 1,2 \cdot 10^6$$

$$CPI = \frac{n_{\text{cicli}}}{n_{\text{istruzioni}}} = \frac{1,2 \cdot 10^6}{4 \cdot 10^5} = 3$$

1.3.1 Valutazione delle prestazioni

Tesi (Amdahl). La velocità di un programma è limitata dalla frazione di codice che non può essere parallelizzata,

$$\text{aumento di } v = \frac{t \text{ per eseguire il programma su un processore}}{t \text{ per eseguire il programma su } N \text{ processori}}$$

$$\boxed{\frac{T(1-f) + T \cdot f}{T(1-f) + \frac{T \cdot f}{N}} \implies \frac{1}{(1-f) + \frac{f}{N}}}$$

a. Se f è piccolo, l'utilizzo di processori paralleli ha un effetto ridotto.

b. Se N tende all'infinito, l'aumento di processori paralleli porta alla diminuzione di vantaggi.

Il tempo di esecuzione viene calcolato sfruttando il numero di istruzioni da svolgere, il valore di CPI e la frequenza:

$$\boxed{ExeTime = \frac{IC \cdot CPI}{F}}$$

1.3.1.1 Speed Up

Tesi (Amdahl)**.** Il miglioramento delle prestazioni globali ottenuto con un miglioramento particolare dipende dalla frazione di tempo in cui esso viene eseguito.

$$\boxed{speedup = \frac{1}{1 - F + \frac{F}{S}}}$$

$S = speedup_{\text{migliorato}}$ $F =$ frazione

Lo *speedup* fra due calcolatori si calcola secondo

$$\boxed{speedup = \frac{T_1}{T_2} = \frac{1}{1 - F_m + \frac{F_m}{S_m}}}$$

ciò ponendo sempre al denominatore il valore maggiore.

Sezione 2

MEMORIE

Parte del calcolatore che immagazzina i dati. Esistono molteplici tipologie di memorie che cooperano in ogni elaboratore. Questo è dovuto alla compensazione fra vantaggi e svantaggi di ciascuna di esse.

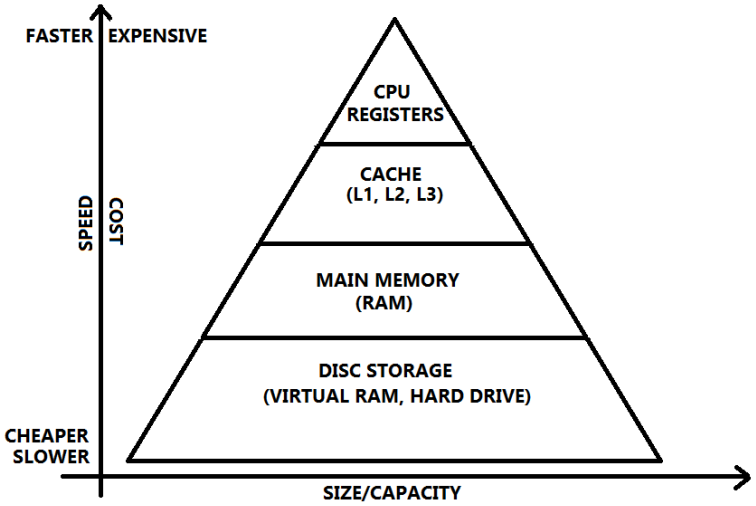

Figura 2.1: Gerarchia delle memorie

2.1 Caratteristiche delle memorie

Tabella 2.1: Elenco elementi distintivi delle memorie

CARATTERISTICA	DEFINIZIONE
locazione	processore, interna o esterna
capacità	dimensione e numero word
unità di trasferimento	word o blocco
metodo di accesso	sequenziale, diretto, casuale o associativo
prestazioni	tempi e velocità
modello fisico	semiconduttore, magnetico o ottico
caratteristiche fisiche	volatile o meno, riscrivibile o meno

2.1.1 Velocità di trasferimento

L'accesso casuale alla memoria ha velocità pari all'inverso del tempo di ciclo. Invece quello non casuale è regolato dalla seguente relazione:

$$\boxed{T_N = T_A + \frac{N}{R}}$$

- T_N =tempo medio per leggere/scrivere N bit
- T_A =tempo di accesso medio
- N =numero di bit
- R =rate di trasferimento in bit per secondo

2.2 Tipologie di memorie

Distinguiamo varie memorie distribuite su tre livelli: contenute nel processore, interne ed esterne. In ogni caso, ognuna di queste è suddivisa in blocchi contenenti a loro volta delle parole.

2.2.1 Memorie contenute nel processore

Sono fondamentalmente i registri e la cache. Hanno velocità di accesso sorprendenti ma pagano in capienza e costi.

2.2.1.1 Registri

Le istruzioni vengono gestite celermente. I registri di controllo di stato sono quattro:

- PC (*Program Counter*)
- IR (*Instruction Register*)
- MAR (*Memory Address Register*)
- MBR (*Memory Buffer Register*)

Si cita inoltre il registro PSW (*Program Status Word*) che contiene una panoramica dell'esecuzione in atto. Esistono diversi metodi di indirizzamento degli operandi nei registri:

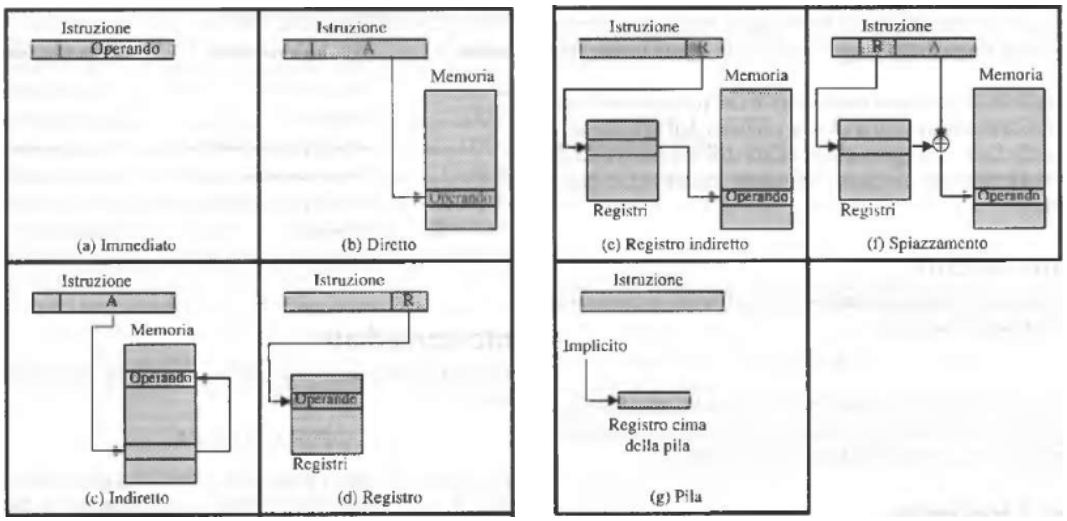

Figura 2.2: Schemi di indirizzamento nei registri

Le istruzioni compiono il ciclo illustrato nella seguente immagine:

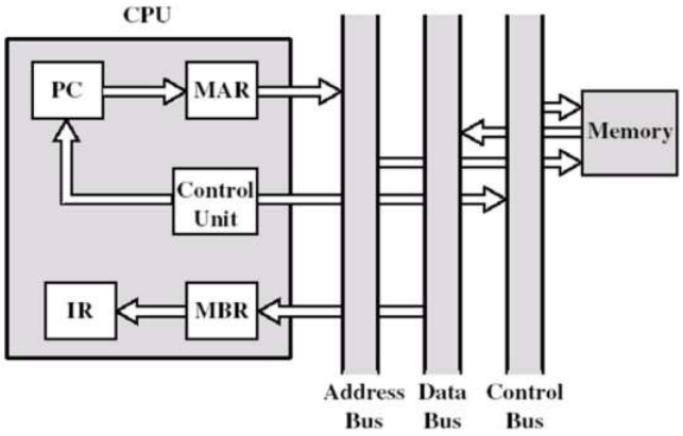

Figura 2.3: Ciclo vitale delle istruzioni

2.2.1.2 Cache

Memoria che si interpone fra la CPU e la RAM.

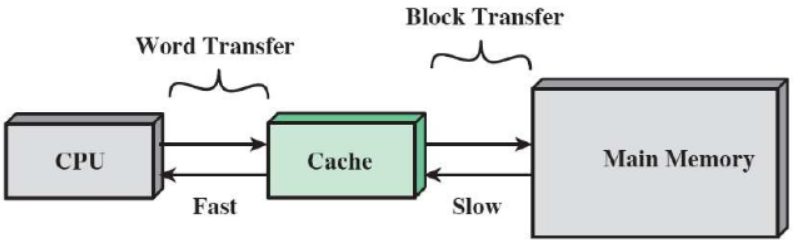

Figura 2.4: Posizionamento della Cache

Possiede grandi valori in termini di velocità ma ad un costo elevato. Il tempo di accesso è determinato da:

$$\boxed{T_s = (T_1 \cdot H) + (1 - H) \cdot (T_1 + T_2) = T_1 + (T_2 \cdot (1 - H))}$$

- T_1 =tempo di accesso alla cache
- T_2 =tempo di accesso alla RAM
- H =hit ratio (dato trovato su ricercato)

La cache è divisa in livelli.

2.2.1.2.1 Associazione diretta

Il *Direct Mapping* alloca ogni blocco della memoria inferiore in una sola posizione di quella superiore.

2.2.1.2.2 Associazione completa

Ogni blocco della memoria inferiore può essere mappato in uno qualsiasi di quella superiore.

2.2.1.2.3 Associazione a gruppi

Ogni blocco della memoria inferiore può essere mappato in un insieme di blocchi di quella superiore.

2.2.2 Memorie interne

Sono legate alla scheda madre dell'elaboratore e, senza di esse, questo non si accenderebbe neppure.

2.2.2.1 RAM (*Random Access Memory*)

Contiene parole di dati di rapido accesso utili per la sessione corrente. Viene organizzata in 2^k parole di M bit ciascuna. La capacità è dunque data da:

$$\boxed{C = 2^k \cdot M}$$

L'indirizzo di memoria è strutturato in modo tale da definire il posizionamento della cella tramite la riga e la colonna indicate.

2.2.2.1.1 SRAM e DRAM

La RAM si divide in SRAM (*Static RAM*) e DRAM (*Dynamic RAM*).

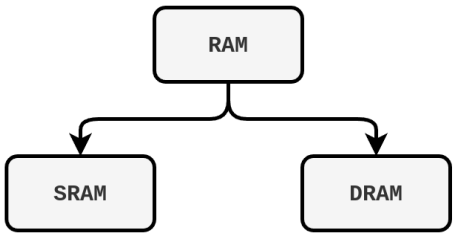

Figura 2.5: Divisione RAM in SRAM e DRAM

La prima, statica, è veloce ma costosa mentre la seconda richiede un refresh periodico.

2.2.2.1.2 Codici d'errore

Per contrastare ipotetici errori dovuti a malfunzionamenti si sfruttano:

a. codici rilevatori
   individuano un cambiamento grazie a un bit di parità quindi non localizza l'errore;

b. codici correttori
   se fornita una ridondanza riesce a sistemare l'errore.

Chiaramente il secondo metodo è più sicuro ma richiede un costo, di bit adoperati, nettamente superiore.

2.2.2.1.3 Codice di Hamming

Ricade fra gli ECC (*Error-Correcting Codes*) ed è basato sull'aggiunta di un numero di bit di controllo alla stringa originale:

$$\boxed{2^K - 1 \geq N + K}$$

- K = numero minimo di bit di controllo;
- N = numero di bit della stringa originale.

I bit di controllo, posizionati nella stringa ogni potenza del due, segnalano l'errore e ne identificano la locazione.

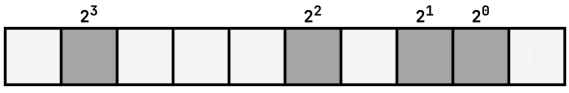

Figura 2.6: Posizione dei bit di controllo

In generale, il bit in posizione n viene monitorato da quei bit di controllo C_i tali che:

$$\boxed{\sum C_i \cdot 2^i = n}$$

Dunque un certo bit è identificato nelle eventuali variazioni dalla somma dei bit di parità.

2.2.2.2 ROM (*Read Only Memory*)

Accessibile in sola lettura e non volatile. Dunque il contenuto resta invariato dopo la fabbricazione. Contiene librerie per funzioni usate di frequente e programmi di sistema.

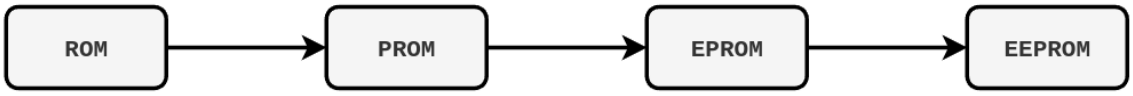

Figura 2.7: Derivate della ROM

Tabella 2.2: Memorie di sola lettura

ACRONIMO	NOME	DESCRIZIONE
ROM	*Read Only Memory*	programmata in fabbricazione
PROM	*Programmable ROM*	programmabile post-produzione
EPROM	*Erasable PROM*	riscrivibile se ripristinata
EEPROM	*Electrically EPROM*	riscrivibile

2.2.3 Memorie esterne

Dischi esterni quali HDD o SSD. Vengono inseriti per conferire grande capienza ad un costo ribassato.

2.2.3.1 Dischi magnetici

Una testina legge e scrive su di un piatto circolare rotante. Questa solitamente è fissa ma in certi casi può muoversi nelle tracce.

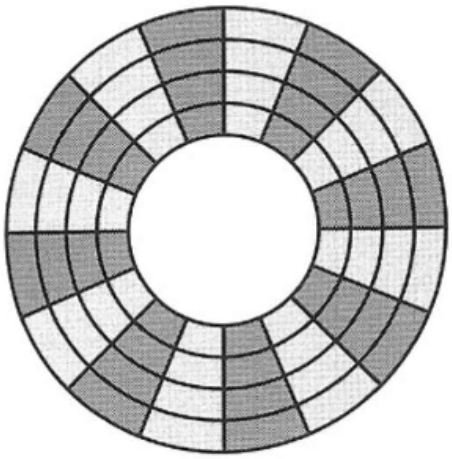

Figura 2.8: Organizzazione tracks

L'organizzazione a circonferenza rende i dati salvati internamente di inferiore velocità d'accesso rispetto a quelli esterni.

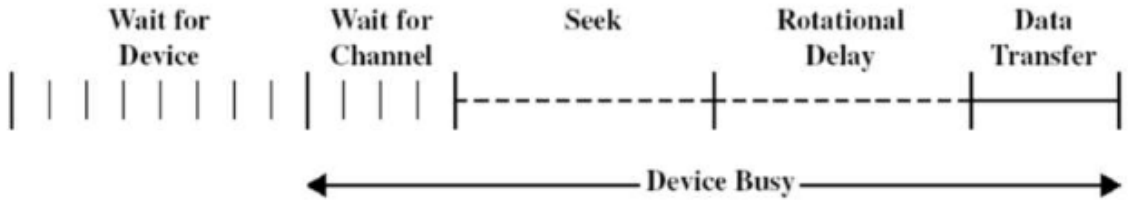

Figura 2.9: Prestazioni di accesso al disco

Si intuisce che un'organizzazione sequenziale dei dati riduce i tempi di lavoro.

2.2.3.2 RAID

Servono a garantire delle ridondanze di dati in modo tale da contrastare eventuali problemi che ne causerebbero la perdita. Esistono varie tipologie di RAID (*Redundant Array of Independent Disk*):

- **RAID 0**, spartizione dei dati su due dischi;
- **RAID 1**, copia dei dati su un secondo disco;
- **RAID 2**, viene sfruttato il codice di Hamming;
- **RAID 4**, il disco di parità contiene l'intero blocco anziché una parte;
- **RAID 5**, i blocchi di parità vengono direttamente inseriti nei dischi dati;
- **RAID 6**, il blocco di parità viene smezzato in due dischi;
- **RAID 10**, composizione in sequenza dei raid 0 e 1.

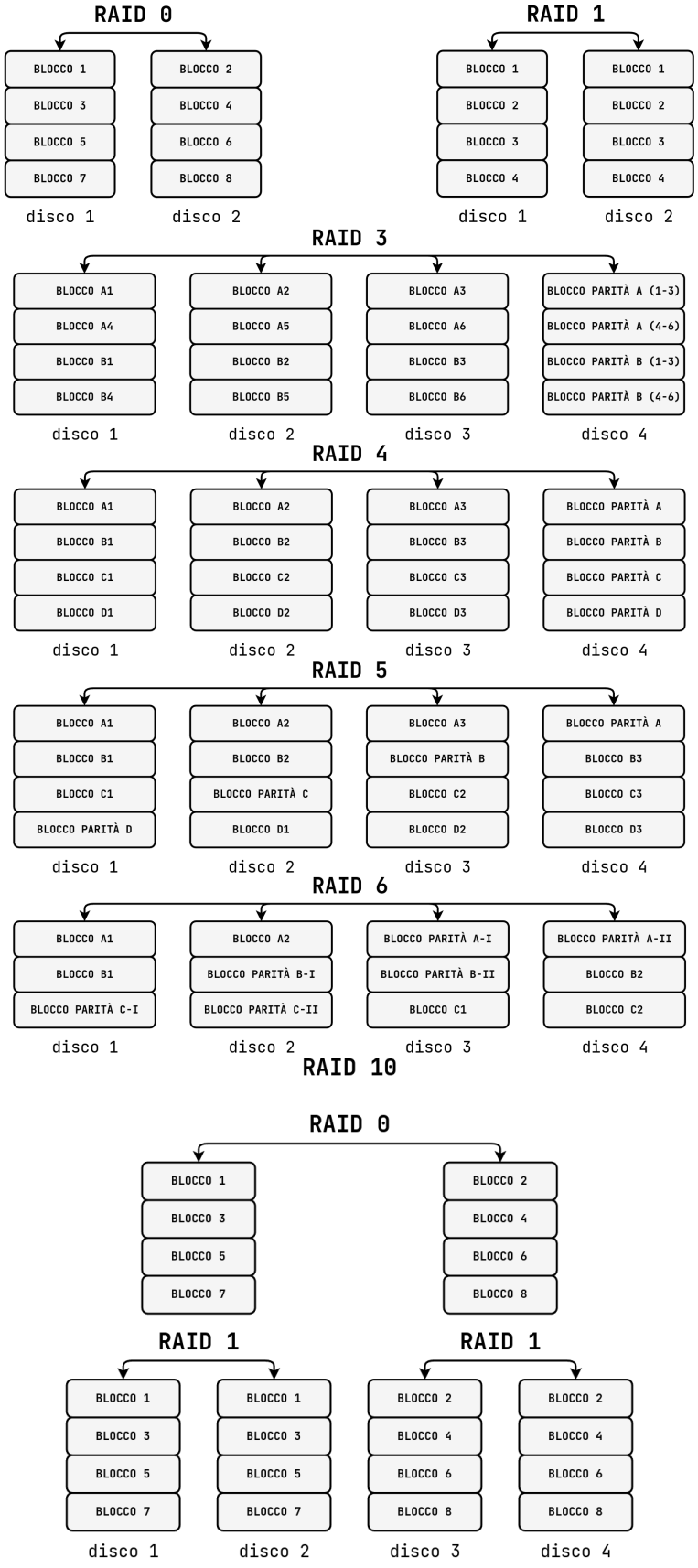

Figura 2.10: RAID 0, 1, 3, 4, 5, 6, 10

Sezione 3

I/O

La complessità maggiore legata ai moduli di I/O è che non sono sincronizzati, in termini di invio di richieste, con la CPU. Intervengono quindi dei controllori costituiti da registri il cui accesso è simile a quello in memoria. Il contenuto di questi può essere:

- dati;
- comandi;
- informazioni.

A seconda del ruolo il modulo assume però due nomi differenti:

a. I/O controller (bassa complessità)

b. I/O processor (alta complessità)

Ci sono tre metodologie per la comunicazione:

- programmata
- guidata da interrupt
- *Direct Memory Access*

Tabella 3.1: Tecniche di I/O

TRANSFER	NO INTERRUPTS	INTERRUPTS
CPU	programmed	interrupt-driven
DIRECT	/	DMA

3.1 Programmed I/O

La CPU controlla periodicamente un bit flag che segnala le richieste dell'I/O.

3.1.1 Memory mapped I/O

Per identificare in modo univoco una periferica le si associa un indirizzo di memoria mappandola. Si avrà quindi la distinzione fra moduli memory mapped ed isolated.

Tabella 3.2: Moduli I/O memory mapped e isolated

MEMORY MAPPED	ISOLATED
potenziale sfruttato	istruzioni dedicate
memoria limitata	memoria libera

3.2 Interrupt-driven I/O

Arriva un segnale di interrupt alla CPU che può decidere di accettarlo o meno in base alla criticità delle operazioni che sta compiendo. Si ha poi un registro PSW (*Program Status Word*) contenente le informazioni sullo stato della CPU. Per interruzioni multiple si assegna il bus alla periferica con maggiore priorità. Viene inoltre effettuato un polling per ricercare il modulo che ha emesso la subroutine.

3.3 Direct Memory Access I/O

Come soluzione agli interrupt troppo frequenti si installa un chip hardware che permette di accumulare diversi segnali in un singolo interrupt e può accedere direttamente alla memoria. Esistono due modalità di trasferimento:

i. block transfer (veloce, DMA come master del bus dati);

ii. transparent (lenta, DMA richiede l'accesso al bus).

Sezione 4

CPU

La *Central Processing Unit* è dedita allo svolgimento di istruzioni e lo fa servendosi di sue componenti interne.

4.1 Ciclo della CPU

Ogni microprocessore compie ripetutamente due azioni:

i. fetch (reperimento dell'istruzione);

ii. execute (esecuzione dell'istruzione).

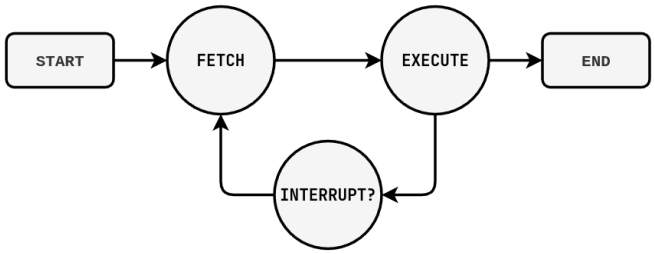

Figura 4.1: Ciclo della CPU

Alla seconda viene associato un controllo di eventuali interruzioni. Inoltre, possono essere eseguite quattro tipologie di operazioni:

1. **cpu-memoria**, trasferimento dati tra cpu e memoria
2. **cpu-i/o**, trasferimento dati tra cpu e dispositivi di i/o
3. **elaborazione dati**, operazione logica o aritmetica sui dati
4. **controllo**, alternamento della sequenza delle istruzioni (salti)

4.1.1 Esecuzione parallela di istruzioni

L'esecuzione di istruzioni parallele avviene secondo i seguenti steps:

1. fetch della prossima istruzione dalla memoria all'IR;
2. cambio del contenuto del PC per puntare la prossima istruzione;
3. determina il tipo di istruzione presa;
4. se l'istruzione usa word in memoria determina la locazione;
5. esegue l'istruzione.

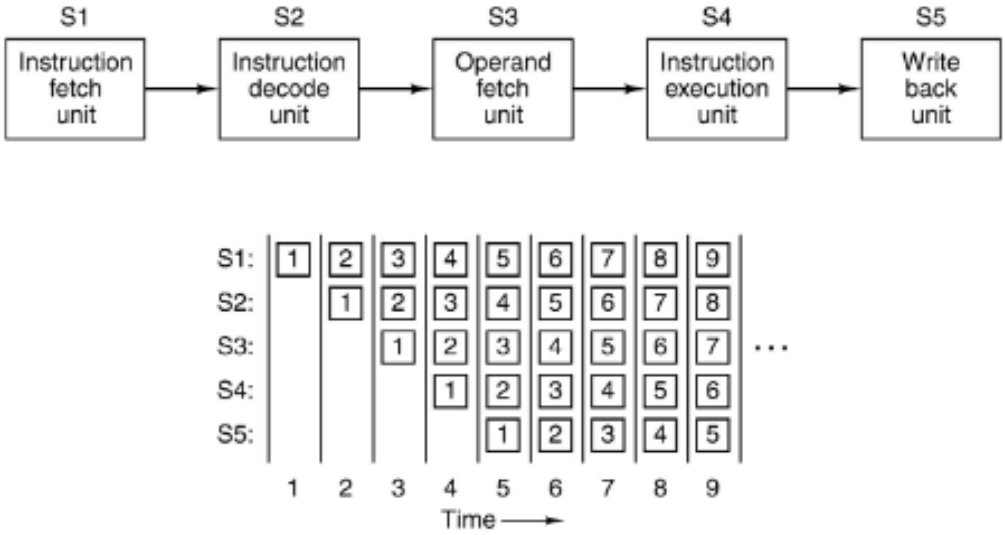

Figura 4.2: Esecuzione parallela delle istruzioni

4.1.1.1 Ciclo della CPU con interruzioni

Dopo aver eseguito il fetch-execute si controlla se è presente un istruzione di interrupt del tipo:

- program (overflow, division by zero)
- timer (interno alla cpu)
- i/o (fine operazione i/o)
- hardware (guasto, mancanza alimentazione)

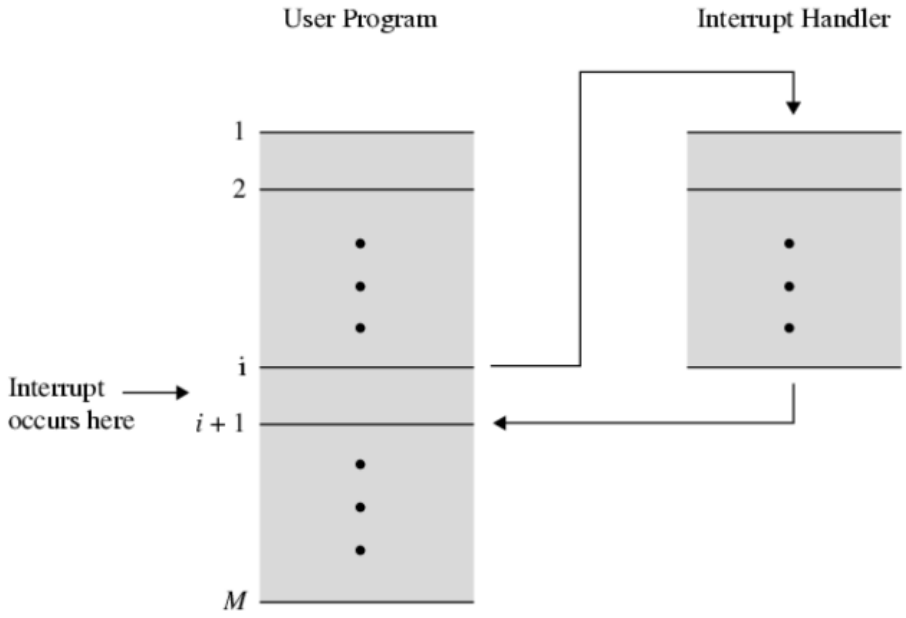

Figura 4.3: Schema interrupt

Esistono inoltre due tecniche di organizzazione per eseguire interruzioni multiple, esse possono infatti esser disposte in modo:

a. **sequenziale**, mentre la CPU gestisce un'interruzione ignorerà tutte le altre. Queste restano pendenti e vengono eseguite secondo il principio FCFS (*First Come First Served*);

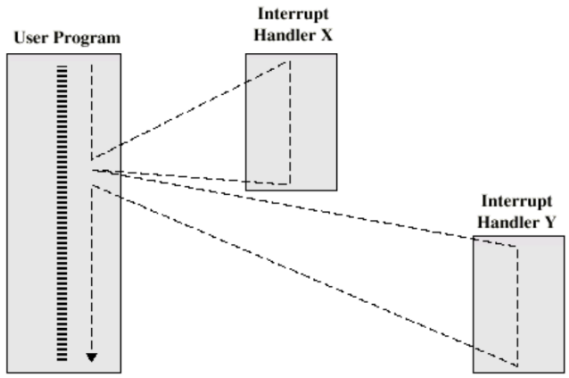

Figura 4.4: Interruzioni multiple

b. **annidato**, l'interruzione ne contiene al suo interno altre eventuali interruzioni.

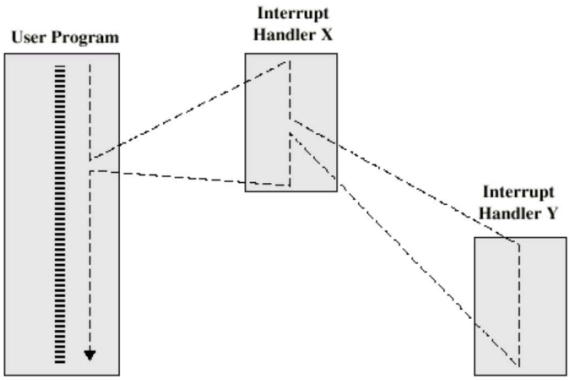

Figura 4.5: Interruzioni annidate

4.2 ALU

Unità che esegue le operazioni aritmetiche e logiche sui dati.

Figura 4.6: Schema grafico componente ALU

In ingresso vengono forniti dati, tramite registri, e l'unità di controllo. Sono poi restituiti i risultati ed eventuali flag di controllo.

4.2.1 Complemento a due

In questo caso si procede seguendo tali passaggi:

1. conversione base binaria;
2. complemento a uno;
3. incrementazione.

Da notare è che gli ultimi due passaggi vengono effettuati se, e solo se, il numero di partenza era negativo. Il bit più significativo del numero ha peso per quanto riguarda il segno:

$$0 \rightarrow \text{positivo} \qquad 1 \rightarrow \text{negativo}$$

4.2.2 Adders

Esistono diversi circuiti per eseguire somme fra operandi:

- **Half Adder**, dati due bit in ingresso presenta in uscita la somma e il riporto
- **Full Adder**, dati due bit e il risporto in ingresso presenta in uscita la somma e il riporto

Tabella 4.1: I/O Full Adder

A	B	CARRY IN	SUM	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

4.3 Livello ISA

Si pone come connessione fra software e hardware ed accoglie il linguaggio macchina.

4.3.1 Linguaggio macchina

Ogni istruzione è formata dai seguenti elementi:

- codice operativo (*op code*);
- indirizzo degli operandi di ingresso;
- indirizzo del risultato;
- riferimento alla prossima istruzione.

Il formato è schematizzabile in questo modo

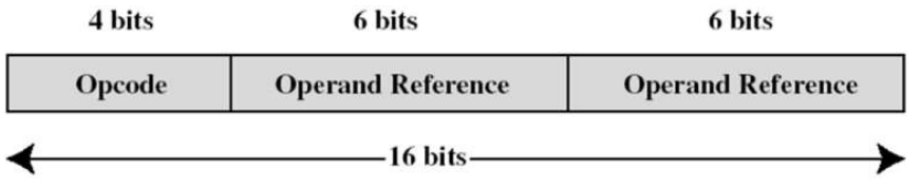

Figura 4.7: Rappresentazione dell'istruzione

In linea generale, non esiste un unico formato per tutte le istruzioni. Esistono quattro classi di queste ultime:

i. elaborazione dati
ii. memorizzazione
iii. i/o
iv. controllo

4.4 Pipeline

Si introduce il concetto di prefetch.

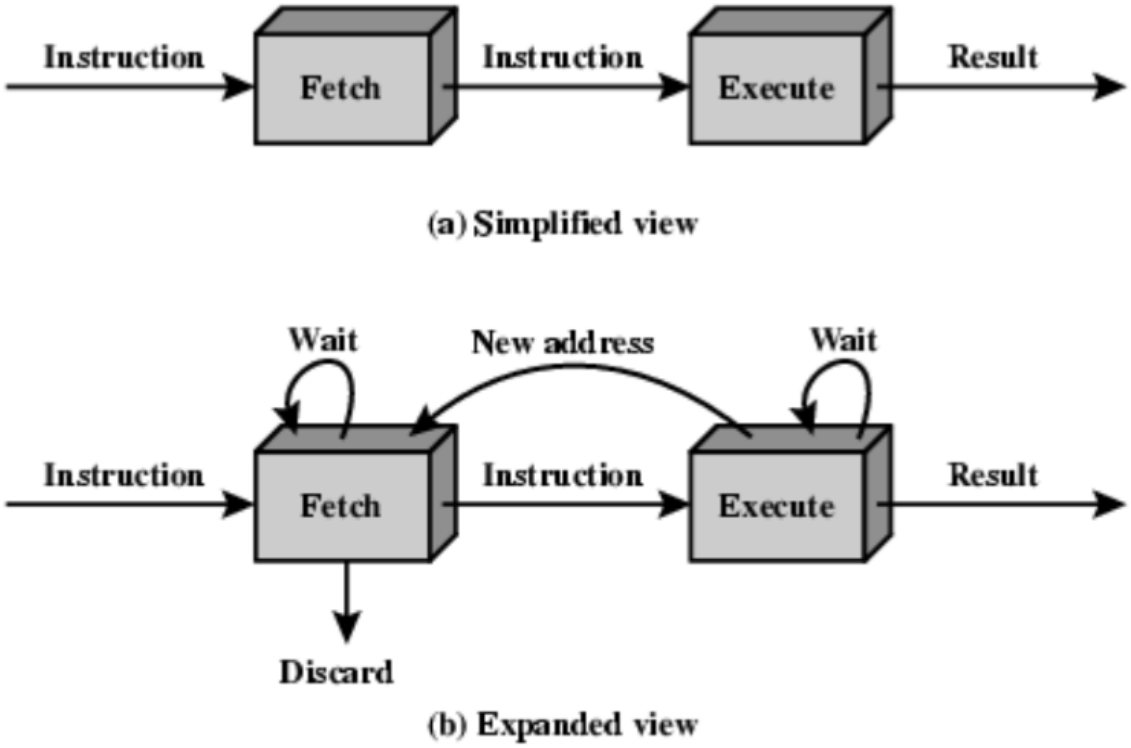

Figura 4.8: Pipeline a due stadi

Si eseguono contemporaneamente diverse fasi per sfruttare al meglio il processore:

i. fetch
ii. decodifica
iii. calcolo operandi
iv. fetch operandi
v. esecuzione
vi. scrittura risultato

4.4.1 Temporizzazione

Assumendo che le sei fasi sopra citate abbiano la stessa durata e non vi siano conflitti nell'accesso alla memoria si ha un diagramma di questo tipo:

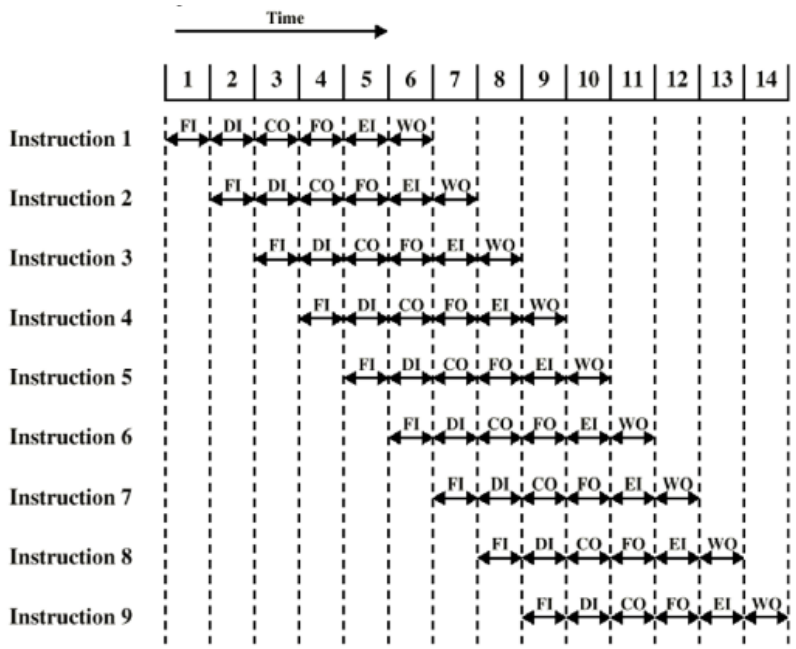

Figura 4.9: Temporizzazione della pipeline

Occorre però considerare che le fasi creano uno sbilanciamento in quanto hanno durata differente. Una variante del primo metodo analizzato si ha con salto condizionato.

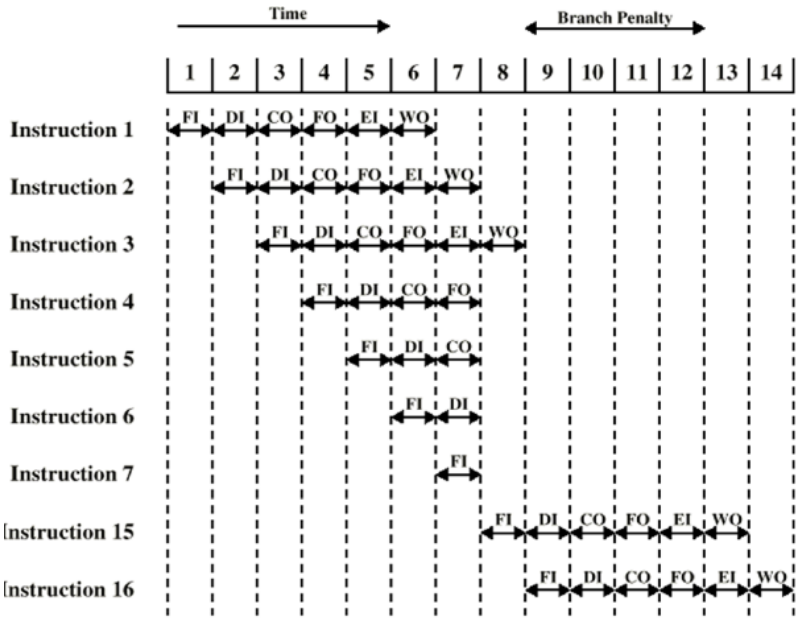

Figura 4.10: Temporizzazione della pipeline con salto

In questo secondo caso l'istruzione 3 contiene un salto alla 15.

Bibliografia

[Sta10] William Stallings. Architettura e organizzazione dei calcolatori. 8th ed. Pearson, 2010. ISBN: 9788871925974.