Architettura degli elaboratori

Leonardo Ganzaroli

Indice

Introduzione

Questi appunti sono derivanti principalmente dalle slide del corso di *Architettura degli elaboratori* che ho seguito durante la laurea Triennale di informatica all'università "La Sapienza".

N.B. Questo corso è il naturale proseguimento di *Progettazione di Sistemi Digitali*, quindi molte cose saranno date per scontate.

1 Argomenti introduttivi

1.1 Cenni storici

Definizione L'alfabeto numerico è un insieme di simboli atti a rappresentare quantità di beni.

I metodi per rappresentare quantità ed effettuare calcoli su di esse esistono da migliaia di anni e nel corso del tempo si sono evoluti fino a diventare automatici:

- **Manuale**
  - **Osso di Lebombo** (circa 35000 a.c.)
    Uno dei più antichi manufatti mai ritrovati ad uso matematico.
  - **Tavoletta d'argilla**
  - **Abaco**
  - **Macchina di Anticitera** (100-150 a.c.)
    Planetario meccanico usato per calcolare le fasi lunari, il movimento dei pianeti, etc...
  - **Sistema decimale posizionale** (1202)

- **Semi-manuale**
  - **Calculating Clock** (1623)
    La prima calcolatrice meccanica.
  - **Pascalina** (1642)
    Macchina che consente di svolgere addizioni e sottrazioni.
  - **Contatore a gradini** (1672)
    Evoluzione della Pascalina che permette di svolgere anche moltiplicazioni e divisioni.
  - **Aritmometro** (1727)
    Miglioramento del contatore a gradini.
  - **Macchina differenziale** (1816)
    Macchina a vapore che permetteva di svolgere anche il calcolo polinomiale.

- **Macchine programmabili**
  - **Macaroni Box** (1889)
    Prima calcolatrice con tastiera che riproduceva il risultato su nastro cartaceo.

- **Tabulator Machine** (1890)
  Primo prototipo di macchina automatica per il censimento, usava schede cartacee perforabili.
- **Macchina di Turing** (1935)
- **Atanasoff Berry Computer** (1942)
  Primo calcolatore totalmente elettronico.
- **Colossus** (1944)
  Primo calcolatore digitale.
- **Mark III** (1950)
  Primo uso di un nastro magnetico come memoria di massa.
- **UNIVersal** (1951)
  Elaboratore ad uso generico, programmi redatti tramite lo *Short Order Code*.
- **RAMAC** (1957)
  Primo uso di un Hard-disk come memoria di massa.
- **CEP** (1961)
  Il primo calcolatore elettronico italiano usato per le ricerche scientifiche.

- **Minicomputer**

  A partire dagli anni sessanta ci fu una nuova generazione di calcolatori con costo e stazza ridotti:

  - **PDP-1** (1960)
  - **Xerox Alto** (1970)
    Aveva un sistema operativo con una rudimentale interfaccia grafica.

- **Microcomputer**

  Un decennio dopo ci fu la nascita dei microprocessori, i più importanti sono:

  - **Intel 4004** (1971)
  - **Intel 8008** (1972)
    Primo uso di un registro apposito per gli indirizzi.
  - **Intel 8085** (1976)
    Sfruttava il DMA e le interruzioni vettorizzate che portarono ad un miglioramento delle comunicazioni I/O e del multitasking.
  - **Intel 8086** (1978)
    Aveva una gestione segmentata della memoria, ciò permetteva la rilocazione dei programmi.

  - **Motorola 68000** (1979)
    Utilizzava un multi-bus per trasferire informazioni eterogenee.
  - **Intel 80286** (1982)
    Primo uso della memoria virtuale.

- **Seconda generazione**
  - **MIPS** (1982)
    Grazie all'uso della *pipeline* aveva delle ottime prestazioni.
  - Nel 1984 Sony e Philips presentano il **CD-ROM**
  - **Intel 80386** (1985)
    Primo uso del cahing della memoria.
  - **ARM2** (1987)
    Processore dotato di un coprocessore matematico.
  - **Intel 80486** (1989)
    Primo uso del *Read-Ahead*.

- **Terza generazione**
  - **Pentium** (1993)
  - **Pentium Pro** (1995)
    Utilizzava la Cache di 2° livello e l'esecuzione *Out of Order*.
  - **Pentium II** (1997)
    Introduzione della tecnologia *MMX* per l'elaborazione audio e video.
  - **Pentium III** (1999)
    Conteneva l'estensione *SSE* che portò ad un potenziamento del coprocessore matematico.

- A partire dal 2001 si iniziò a puntare sul parallelismo tramite processori multipli e multicore:
  - **Pentium D** (2005)
    Dotato di 2 core e cache condivisa tra essi.
  - **Intel i3,i5,i7,i9** (2010-...)
    Tre livelli di Cache e uso dell'*HyperMultiThreading*.

1.2 Richiami di teoria dell'informazione

La teoria pone l'attenzione sul come inviare un messaggio (concatenazione di simboli/segnali/parole) e ricostruirlo poi in modo esatto, inoltre propone uno schema generale di un sistema di comunicazione:

1. **Sorgente** = insieme delle possibili parole
2. **Trasmettitore** = entità che codifica una parola in un segnale
3. **Canale** = mezzo con cui si propaga il segnale
4. **Ricevente** = intermediario che decodifica il segnale
5. **Ricevente**

Figura 1: Schema generale

2 Macchina di Von Neumann

Definizione La macchina di Von Neumann è un modello generale di elaboratore che prevede l'archiviazione di dati e istruzioni nella stessa memoria e la possibilità di effettuare salti nel programma.

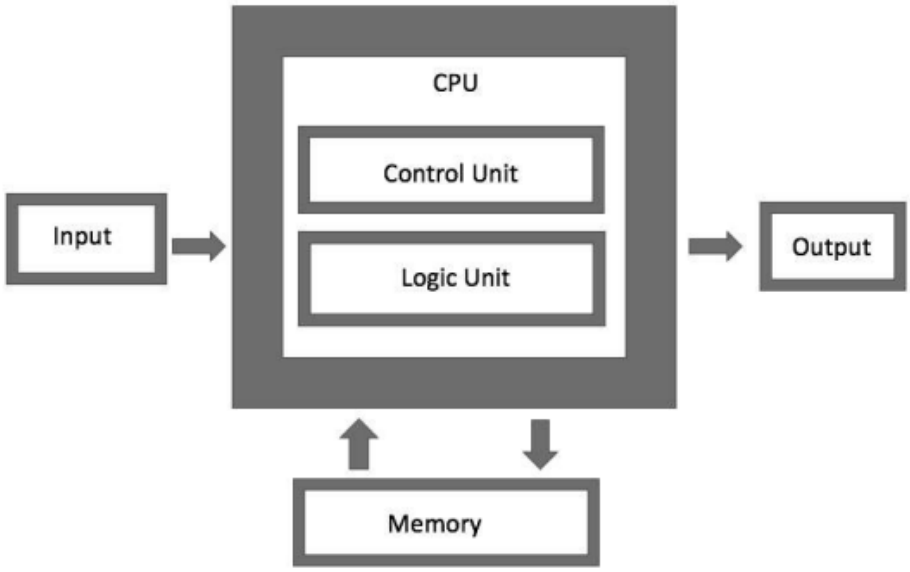


Figura 2: Modello di Von Neumann

2.1 Componenti

- **CU**

  La Control Unit è predisposta a scandire le operazioni elementari necessarie per eseguire ogni istruzione.

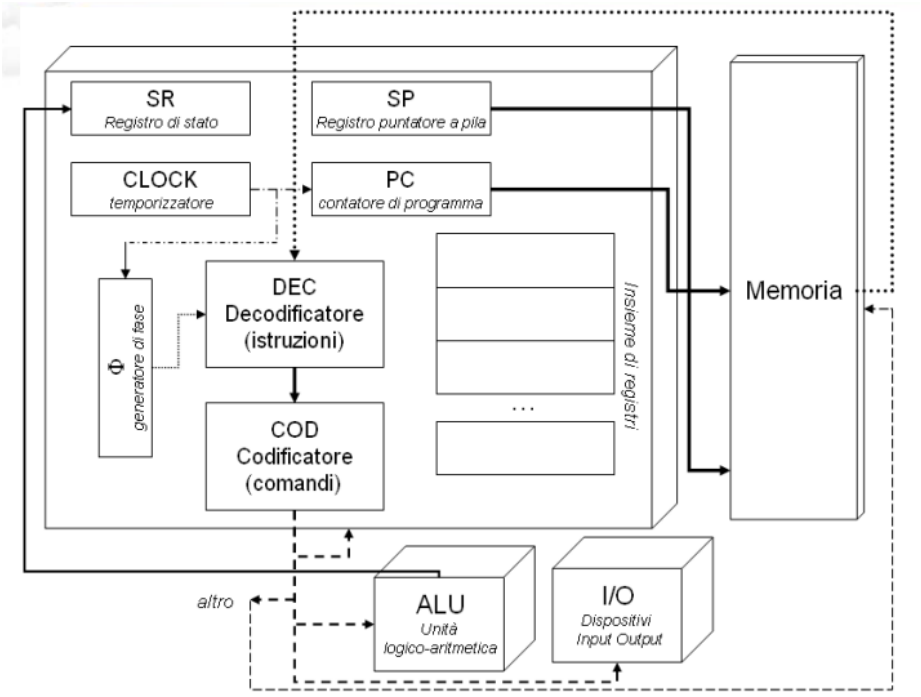


Figura 3: Control Unit

  - I **registri** sono delle piccole memorie presenti internamente alla CU, sono di 2 tipi:
    - Ad uso **speciale**:
      - **Program Counter**
        Contiene l'indirizzo dell'istruzione da seguire attualmente.
      - **Status Register**
        Contiene informazioni riguardo lo stato della CU.
      - **Stack Pointer**
        Contiene l'indirizzo della cima della pila.
    - Ad uso **generale**
      Usati per mantenere risultati di operazioni e/o informazioni di controllo.

– **Generatore di fase**
Scandisce le varie di fasi di esecuzione delle operazioni elementari, quest'ultime si possono riassumere in:

1. **Caricamento**
   Lettura dell'indirizzo nel PC.
2. **Decodifica**
   Riconoscimento di operazione ed operandi.
3. **Esecuzione**
4. **Spostamento**

Ogni istruzione ha bisogno di un certo numero di cicli macchina per essere eseguita, questi possono essere diversi e dipendono da molti fattori.

– **Transcodificatore**
Riconosce le istruzioni e genera gli opportuni comandi.

- **ALU**
  L'unità logico-aritmetica si occupa di effettuare operazioni logiche ed aritmetiche.

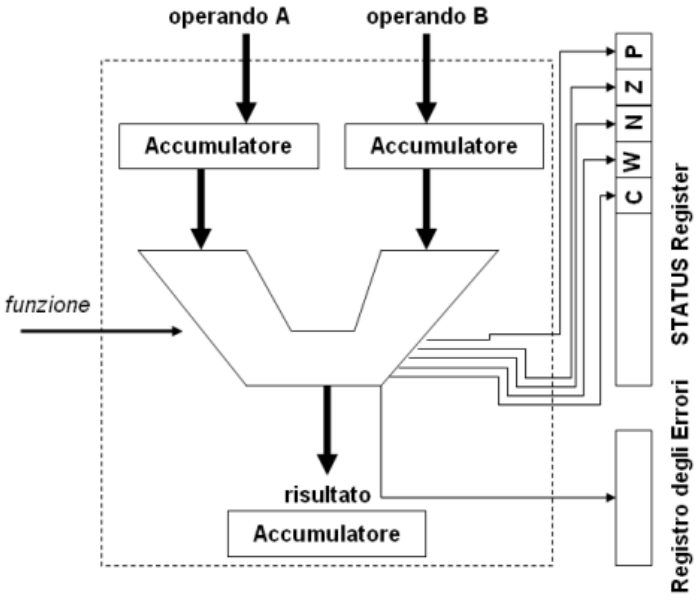


Figura 4: ALU

- **Accumulatori**

  Sono dei registri "trasparenti" che ospitano gli operandi/soluzioni durante le operazioni.

- **Registro degli errori**

  Usato per indicare situazioni non risolvibili.

- **Funzione**

  Un input che indica l'operazione da eseguire.

- **Flags**

  Output che forniscono informazioni sull'ultima operazione eseguita.

- **Memoria**

  Ha delle aree riservate contenenti informazioni essenziali per il funzionamento e per operazioni particolari. Nel caso di memorie con dimensioni importanti si ricorre ad una divisione logica per semplificare l'accesso, essa viene divisa in banchi e blocchi e l'indirizzo viene sezionato.

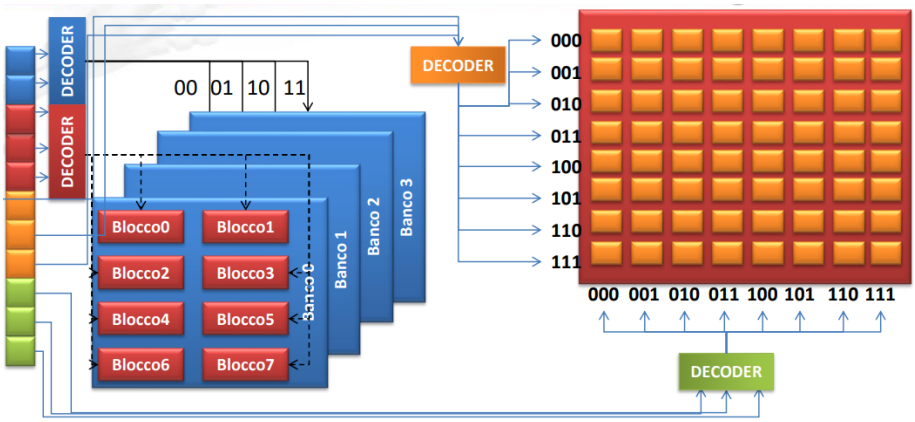
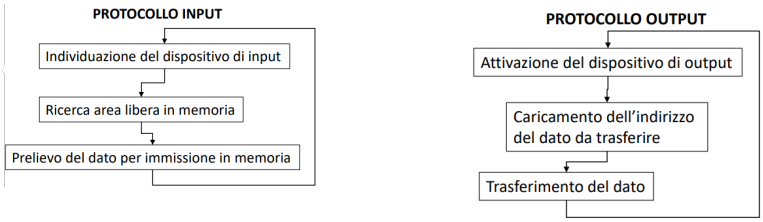


Figura 5: 4 banchi, 8 blocchi

- **Dispositivi I/O**

  Permettono di collegare l'elaboratore con il mondo esterno in diversi modi, un problema di questa interazione è il fatto che i dispositivi hanno delle velocità di trasferimento delle informazioni estremamente minore rispetto alle componenti interne e ciò porta problemi di sincronizzazione.

Visto il possibile trasferimento di dati è opportuno usare un protocollo per regolarlo:



In ogni caso un dispositivo va interconnesso ad un modulo I/O, esso fa da intermediario con il processore:

– **INPUT**

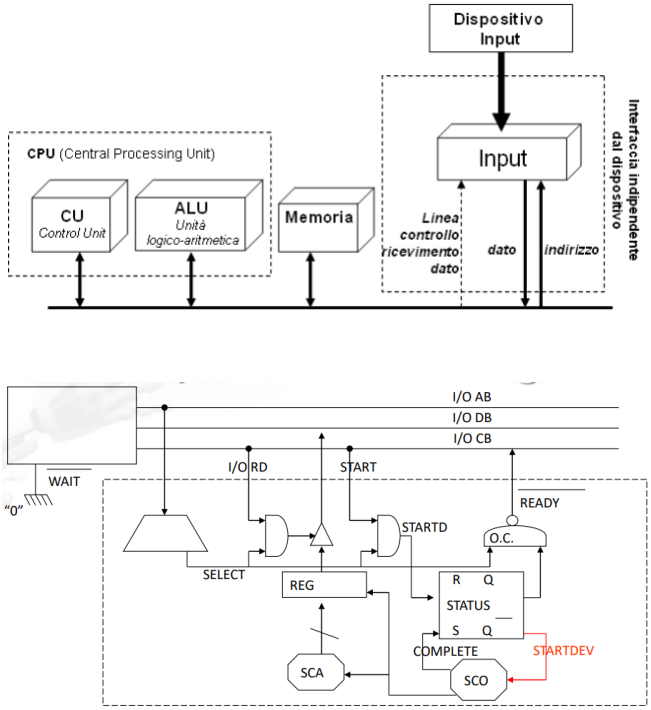


Figura 6: Controller Input dettagliato

Protocollo:

1. il processore invia sull'I/O Address bus l'indirizzo del dispositivo e ne esamina lo stato tramite la linea di controllo $\overline{\text{READY}}$
2. Se il dispositivo non è pronto il processore deve attendere e tornare al punto 1, se è pronto va al punto 3
3. START resetta il flip-flop STATUS e in tale stato rimane per tutta la durata delle operazioni
4. Quando il dato è disponibile in REG, il dispositivo genera il segnale COMPLETE, settando STATUS
5. Nel frattempo il processore, in attesa del dato, esamina il flip-flop STATUS campionando il segnale READY
6. Se $\overline{\text{READY}} = 1$ il processore deve attendere e tornare al punto 5, altrimenti invia il segnale di controllo IO/RD per trasferire il dato presente in REG nella locazione di memoria che ospiterà il valore

– **OUTPUT**

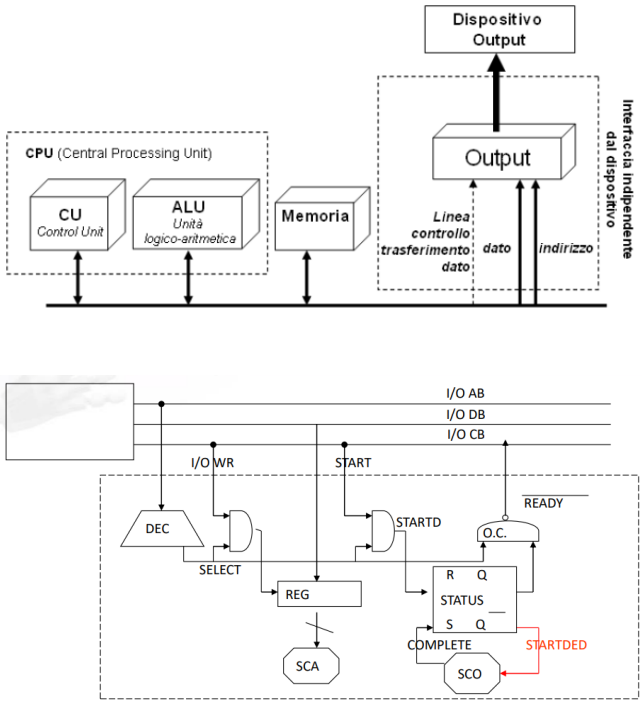


Figura 7: Controller Output dettagliato

Protocollo:

1. il processore invia sull'I/O Address bus l'indirizzo del dispositivo e ne esamina lo stato tramite la linea di controllo $\overline{\text{READY}}$
2. Se il dispositivo non è pronto il processore deve attendere e tornare al punto 1, se è pronto va al punto 3
3. Il processore trasferisce il contenuto dalla locazione di memoria al registro d'interfaccia del dispositivo
4. Il processore invia il segnale START che resetta il flip-flop STATUS e in tale stato rimane per tutta la durata delle operazioni, quando il dato è stato letto il dispositivo genera il segnale COMPLETE che setta STATUS
5. Nel frattempo il processore, in attesa del dato, esamina il flip-flop STATUS campionando il segnale $\overline{\text{READY}}$
6. Se READY = 1 il processore deve attendere e tornare al punto 5, altrimenti può eseguire un'altra istruzione

Per quanto riguarda l'accesso da/ai dati e l'indirizzamento dei dispositivi si usano 2 tecniche:

1. **I/O Canonico**
   Si riserva uno spazio indipendente con specifiche istruzioni.
2. **I/O programmato**
   Si riserva una parte della memoria centrale.

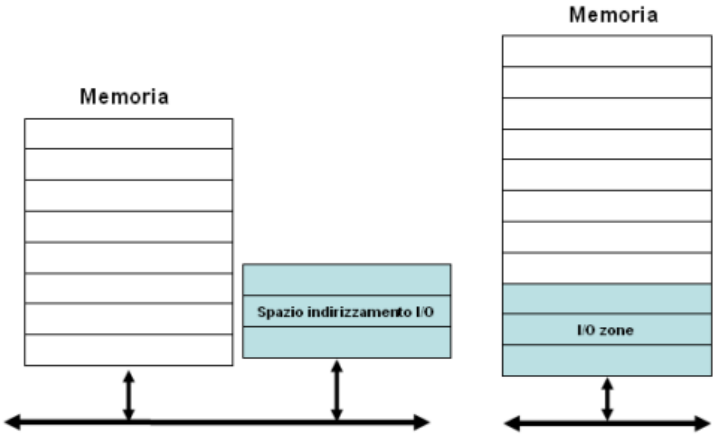
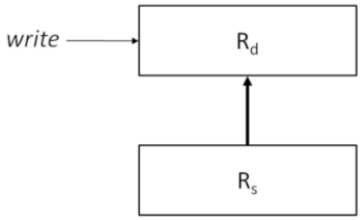
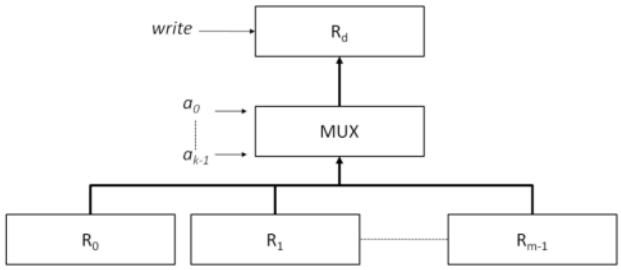


Figura 8: Canonico e Programmato

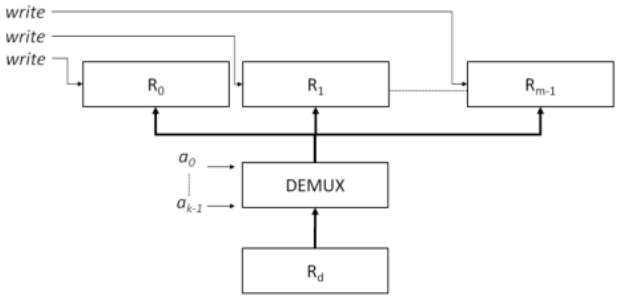
- **Interconnessioni**
  - **Punto a punto**
    Si usa per effettuare trasferimenti da un registro ad un altro.



  - **Multiplexer**
    Si usa per effettuare trasferimenti da molti registri ad uno.

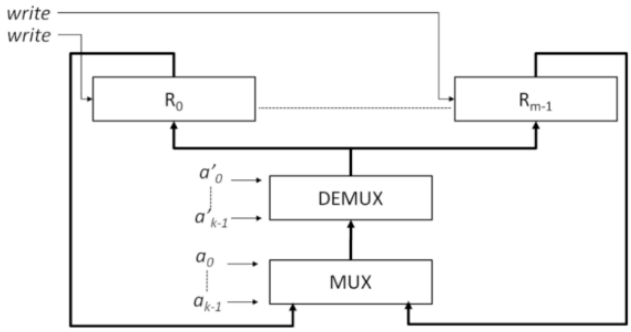


  - **Demultiplexer**
    Si usa per effettuare trasferimenti da un registro a molti.



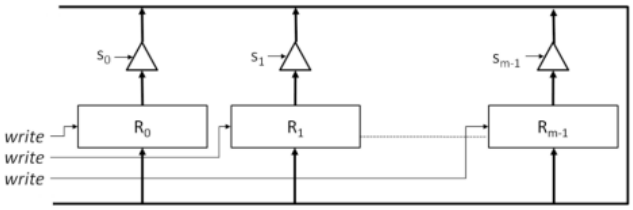
– **Mesh**

Permette di connettere molti componenti tra loro.



– **Bus**

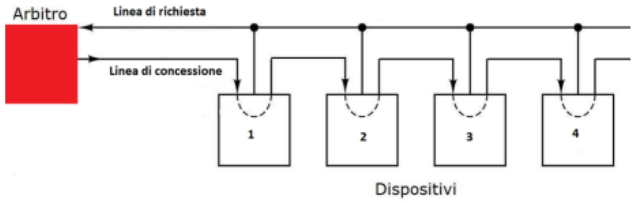
Fascio di n linee, tramite il buffer tristate si può scegliere quale attivare.

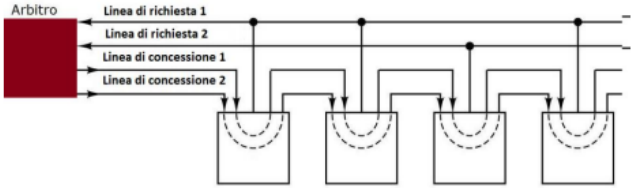


Oggigiorno vengono usati bus multipli o multiplexati che permettono di averne uno tra processore e memoria ed almeno un altro per le periferiche.

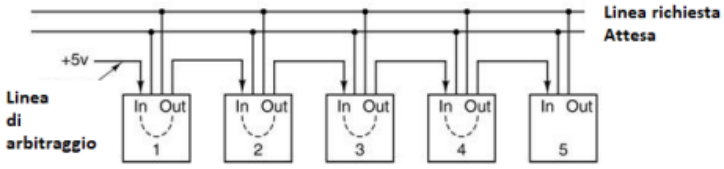
Per risolvere il problema delle richieste simultanee si utilizza l'arbitraggio:

* **Centralizzato**
  Un'entità chiamata arbitro decide qual'è il prossimo dispositivo usando il *daisy chaining*, è possibile usare diverse linee con diverse priorità.





* **Decentralizzato**
  Si usano diverse linee con diverse priorità, quando un dispositivo deve usare il bus invia un segnale sulla linea di richiesta ed analizza le linee per sapere se è il prossimo a poterlo usare.



2.2 Macchina di Harvard

Definizione La macchina di Harvard è un altro modello di elaboratore nato basandosi sull'elaboratore *Harvard Mark I*, a differenza del modello di Von Neumann prevede 2 memorie diverse per le istruzioni e i dati.

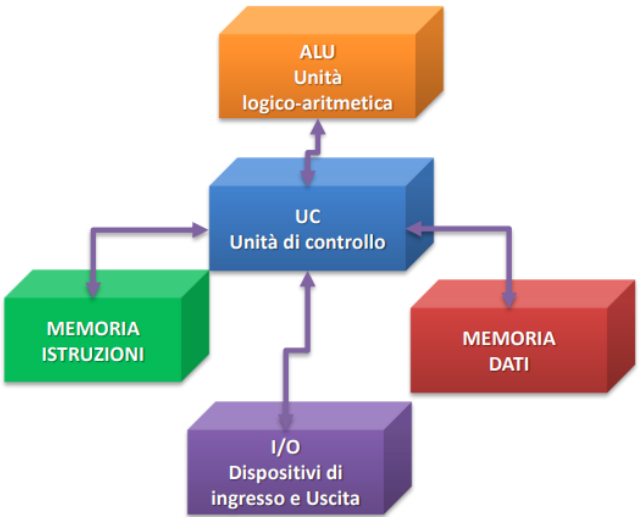


Figura 9: Modello di Harvard

3 Programmi

L'esecuzione di un programma è l'ultimo passo di un processo che inizia con la scrittura dello stesso in un linguaggio simbolico di alto livello, le 3 entità che generalmente sono coinvolte in questo processo sono:

1. **Compilatore**

   Trasforma il programma in linguaggio assembly, ottimizza il codice ed effettua un eventuale riordino delle istruzioni.

2. **Assemblatore**

   Traduce il programma assembly in un file oggetto che è una combinazione di istruzioni macchina, dati ed altre informazioni.

I file oggetto sono divisi solitamente in 6 sezioni:

(a) **Object File Header**
Contiene le locazioni e le dimensioni delle altre sezioni.

(b) **Text Segment**
Contiene le istruzioni.

(c) **Data Segment**
Contiene i dati.

(d) **Relocation information**
Identifica istruzioni/dati che dovranno essere rilocati dal linker.

(e) **Symbol Table**
Contiene i simboli non ancora definiti.

(f) **Debugging information**
Contiene informazioni per il debug.

Quando sono disposti in memoria è presente inoltre uno spazio apposito in cui si scambiano dati.

3. **Linker**

   Il linker crea il file eseguibile partendo da quello oggetto, nel caso un programma ne abbia diversi allora essi vengono opportunamente collegati.

Una volta che il file eseguibile è pronto e si vuole eseguirlo entra in gioco il **Loader** che si occupa di caricarlo in memoria.

Un'alternativa a questo processo è l'**interprete** che traduce direttamente le istruzioni senza bisogno di compilazione al costo di maggiore memoria e minore velocità di esecuzione.

3.1 Istruzioni

Definizione Un'istruzione è una stringa binaria che indica all'elaboratore i compiti da svolgere, è divisa in sottostringhe definiti campi:

- **OPCODE**

  Specifica il tipo di operazione.

- **Operando**

  Può contenere un dato, un riferimento ad esso o l'etichetta di un registro.

Il tipo di suddivisione individua il formato dell'istruzione:

- Lunghezza fissa

  Le istruzioni hanno una dimensione prefissata.

- Lunghezza variabile

  La dimensione cambia in base all'operazione.

Quando si programma si utilizzano le istruzioni assembly che sono una rappresentazione simbolica delle istruzioni macchina, infatti tra le 2 esiste un legame uno a uno. In alcuni casi esistono delle pseudoistruzioni che sono composte da più istruzioni elementari. Generalmente sono costituite da:

- L'indirizzo dell'istruzione in memoria
- Un'etichetta
- L'istruzione, comprende:
  - Un codice mnemonico
  - Il modo di indirizzamento
- Eventuali commenti

Esattamente come nei linguaggi di alto livello è possibile definire una **Macro** che permette di accorpare più istruzioni sotto un'etichetta, ovviamente quando si procede ad assemblare il programma sia queste che le pseudoistruzioni vengono sostituite.

Prima di vedere le istruzioni generalmente presenti è opportuno definire le principali *Flag* relative alla ALU:

- **C** (Carry), 1 se l'ultima operazione ha prodotto un riporto/prestito
- **N** (Negative), 1 se l'ultima operazione ha prodotto un risultato negativo
- **Z** (Zero), 1 se l'ultima operazione è nulla
- **W** (Overflow), 1 se l'ultima operazione ha generato un Overflow
- **P** (Parity), 1 se l'ultima operazione ha dato un risultato con un numero pari di 1

3.1.1 Classi

Segue una carrellata di istruzioni generali:

Codice mnemonico	Operandi	Funzione
Spostamento		
LOAD	sorg,dest	Copia il valore della sorgente(memoria) nel destinatario
STORE	sorg,dest	Copia il valore della sorgente nel destinatario(cella di memoria esplicita)
MOVE	sorg,dest	Sposta il contenuto della sorgente nel destinatario (entrambi registri)
PUSH	sorg	Sposta il contenuto della sorgente in cima alla Stack
POP	dest	Sposta l'elemento in cima alla Stack nella destinazione
Aritmetiche		
ADD	dest,sorg,sorg	Inserisce nella destinazione la somma dei valori delle sorgenti
CMP	dest,sorg,sorg	Inserisce nella destinazione la comparazione dei valori delle sorgenti
NEG	dest,sorg	Inserisce nella destinazione la negazione del valore della sorgente
Logiche		
AND	dest,sorg,sorg	Inserisce nella destinazione l'AND dei valori delle sorgenti
OR	dest,sorg,sorg	Inserisce nella destinazione l'OR dei valori delle sorgenti
XOR	dest,sorg,sorg	Inserisce nella destinazione lo XOR dei valori delle sorgenti
NOT	dest,sorg	Inserisce nella destinazione il NOT del valore della sorgente
SL	reg,k	Shift a sinistra di k posti del registro
SR	reg,k	Shift a destra di k posti del registro
ROL	reg,k	Rotazione a sinistra di k posti del registro
ROR	reg,k	Rotazione a destra di k posti del registro
Implicite		
CLRC	-	Imposta a 0 la flag C
CLRN	-	Imposta a 0 la flag N
CLRZ	-	Imposta a 0 la flag Z
CLRW	-	Imposta a 0 la flag W
SETC	-	Imposta a 1 la flag C
SETN	-	Imposta a 1 la flag N
SETZ	-	Imposta a 1 la flag Z
SETW	-	Imposta a 1 la flag W
Salto		
BEQZ	sorg,ind	Se il valore della sorgente è 0 salta all'indirizzo
BGT	sorg1,sorg2,ind	Se sorg1>sorg2 salta all'indirizzo
BLT	sorg1,sorg2,ind	Se sorg1<sorg2 salta all'indirizzo
J	ind	Salta all'indirizzo
JSR	ind	Salva il valore di PC nella stack e salta all'indirizzo di subroutine
RET	-	Ripristina il valore di PC recuperandolo nella Stack
Comando		
HALT	-	Interruzione di sistema
NOP	-	Nessuna operazione
BREAK	-	Interruzione di programma

3.1.2 Modi di indirizzamento

Per poter esprimere dove risiede un operando esistono 2 modi:

1. **Implicito**

   In alcune macchine gli operandi si trovano in posizioni prestabilite, non serve specificarli.

2. **Assoluto** (Esplicito)

   Si indica il suo indirizzo.

Definizione L'indirizzo effettivo è un indirizzo che prende 2 significati diversi in base al tipo di operazione:

1. Nelle istruzioni logico-aritmetiche e di trasferimento dati è l'indirizzo degli operandi interessati
2. In un'istruzione di salto è l'indirizzo dove si vuole andare.

Definizione Un'etichetta è il designatore simbolico di un indirizzo in memoria.

Esistono molti modi per indicare l'indirizzo di un operando:

- **Immediato**

  L'operando si trova nella posizione immediatamente successiva all'istruzione.

- **Diretto**

  Si usa l'indirizzo effettivo in memoria.

- **A registro**

  Si usa l'etichetta del registro.

- **Indiretto**

  Si usa un indirizzo che punta ad un altro indirizzo.

- **Indiretto a registro**

  Come quello a registro ma esso contiene un indirizzo.

- **Indiretto differito**

  Come il precedente ma l'indirizzo punta ad un altro indirizzo.

- **Con spiazzamento**

  Come quello indiretto a registro ma ad esso si aggiunge un offset.

- **Relativo**

  L'indirizzo è dato dal PC ed un offset.

- **Pre/Post incremento/decremento**

  Come l'indiretto a registro ma il contenuto dello stesso viene incrementato/decrementato prima/dopo l'esecuzione dell'istruzione.

3.2 Interruzioni

Definizione Un'interruzione è un evento che cambia la normale sequenza di esecuzione di un programma, le possibili cause possono essere molteplici e non necessariamente dovute ad errori.

Gli elaboratori possono avere 2 tipi di sistemi per gestire le eccezioni:

1. **Mascherabile**

   Un'interruzione può essere interrotta da un'altra interruzione.

2. **Non mascherabile**

   Un'interruzione deve essere gestita prima di gestirne un'altra.

Quando si verifica un'interruzione il controllo viene passato ad un gestore che svolge le azioni appropriate e alla fine fa riprendere il programma dalla posizione in cui si trovava, questo sistema deve quindi garantire che:

- Non ci siano interferenze sul processo interrotto, il contesto viene salvato

  Le fasi di commutazione del contesto sono protette, ossia non possono essere interrotte.

- Il dispositivo generatore venga individuato
- Multiple interruzioni vengano gestite secondo una priorità

L'uso di questo sistema implica che in memoria siano presenti tante routine di servizio (ISR) quanti sono i dispositivi e le possibili interruzioni software, queste richiedono che ci sia una corretta identificazione dei dispositivi.

3.2.1 Identificazione

Ci sono 2 modi diversi per individuare le interruzioni:

1. **Polling**

   Un dispositivo ha un registro che segnala se è presente un'interruzione, la CPU alla fine dell'esecuzione di un'operazione li controlla tutti, questo si può implementare tramite una porta OR o un collegamento *wired OR* di porte open collector.

2. **Int. vettorizzata**

   Una versione migliorata del Polling dove i dispositivi hanno un ulteriore registro che contiene un codice identificativo inviato al processore quando si verifica un'interruzione, questo valore solitamente punta all'inizio della routine necessaria.

Definizione Il driver di una periferica è l'insieme di routine che servono per gestire le interazioni con esso.

3.2.2 Classificazione

Esistono 2 principali metodi di classificazione, il primo in base all'origine:

- Int. Hardware

  Proviene da un componente esterno.

- Int. Software

  Proviene dal programma stesso:

  1. **Trap**
     Richiesta dal programmatore
  2. **Eccezione**
     Dovuta ad errori.

Il secondo in base alla sua relazione con il clock:

- Asincrone (Interne/Esterne)

  Potrebbero accadere in qualsiasi momento.

- Sincrone (Trap)

  Il loro accadimento è noto.

4 RISC/CISC e microprogrammazione

4.1 Microprogrammazione

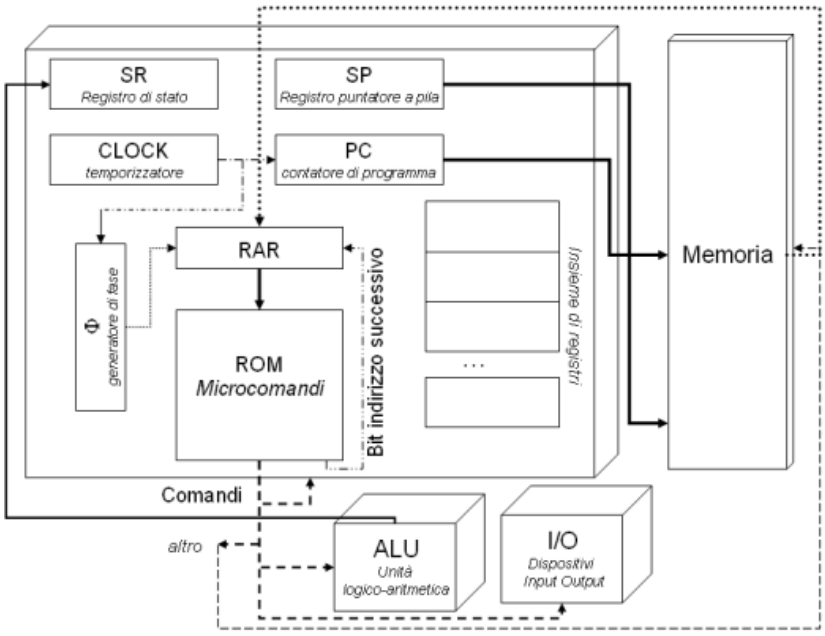
Alla fine degli anni cinquanta si dotò la CU di un decodificatore sequenziale e di una ROM in cui archiviare le sequenze dei comandi, l'idea era di far corrispondere un'istruzione all'esecuzione di un microprogramma scritto nella ROM.

In questo caso il processore fa da interprete ad un linguaggio micro-programmato, la CU dovrà svolgere la seguente trasformazione:

Micro-programma → Insieme di micro-istruzioni → Insieme di micro-operazioni

I componenti essenziali di questa macchina sono:

- Il generatore di indirizzo (della micro-istruzione)
- Il registro RAR (essenzialmente un program counter interno)



Il micro-codice si può categorizzare in base alla sua struttura:

- **Orizzontale**

  Si massimizzano la flessibilità e la parallelizzazione.

- **Verticale**

  Si privilegia la compattezza del codice.

- **Diagonale**

  Una combinazione delle precedenti.

4.2 CISC

Le macchine CISC (Complex Instruction Set Computing) nacquero alla fine degli anni settanta, hanno un set di istruzioni complesse numeroso e spesso a lunghezza variabile.

L'uso di istruzioni complesse era giustificato in quegli anni a causa della tecnologia disponibile:

- Compilatori basilari
- RAM costosa
- No Cache
- Facile sostituzione della ROM contenente il set

Ovviamente le istruzioni di questo tipo presentano degli svantaggi:

- Molti cicli di clock per l'esecuzione
- Nel caso le operazioni non vengano scomposte serve una grande quantità di componenti per eseguirle
- Ogni istruzioni richiede accessi ed operazioni multiple, soprattutto nel giusto ordine

4.3 RISC

Le macchine RISC (Reduced Instruction Set Computer) nacquero nel 1980, questi nuovi processori avevano un set di istruzioni semplici e limitate in modo da poter sfruttare anche la pipeline.

	Dimensione istr.	Formato istr.	Durata istr.	Complessità istr.	Num. registri	Modi ind.	Operazioni
CISC	Variabile	Variabile	Variabile	Complessa	Pochi	Complessi	Operandi in memoria
RISC	Fissa	Uniforme	Fissa	Semplice	Molti	Semplici	Op. ALU solo tra registri

Tabella 1: Comparazione

5 Pipeline

Definizione La canalizzazione è una tecnica che consiste nello scomporre una rete combinatoria inserendo dei registri di disaccoppiamento, questo permette di aumentare la frequenza del clock e svolgere più fasi in parallelo.

In una macchina di tipo RISC ogni istruzione viene eseguita in 5 fasi:

1. **Fetch**
2. **Decode**
3. **Execution**
4. **Memory**
5. **Write Back**

Normalmente è attivo solo un "blocco" del circuito alla volta (associato ad una fase), invece tramite la pipeline ogni blocco elabora la fase di un'istruzione e passa subito alla successiva.

Tutto questa risulta più arduo da implementare nel caso di macchine con istruzioni complesse e con tempi di esecuzione variabili, questo sia perché la frequenza del clock va regolata in modo da permettere l'esecuzione di ogni istruzione in parallelo sia perché alcune fasi potrebbero sovrapporsi tra di loro con istruzioni di lunghezza non fissa.

Esecuzione RISC																				
Δt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Istruzione 1	F	D	E	M	WB																
Istruzione 2		F	D	E	M	WB															
Istruzione 3			F	D	E	M	WB														
Istruzione 4				F	D	E	M	WB													
Istruzione 5					F	D	E	M	WB												

Esecuzione CISC																					
Δt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Istruzione 1	F	D	D	L	L	L	E	E	M	WB											
Istruzione 2				F	D	D	L	L	L	E	M	WB									
Istruzione 3						F	D	D	D	D	L	L	E	M	M	WB					
Istruzione 4										F	F	D	L	L	L	E	M	WB			
Istruzione 5												F	F	D	D	L	L	L	E	M	WB

Figura 10: Esempio di esecuzione

5.1 Hazard

Ovviamente con questo sistema si potrebbero presentare dei problemi durante l'esecuzione (Hazard), si possono classificare in 3 categorie:

1. **Sul controllo**

   Un salto o un'interruzione cambiano il flusso di esecuzione.

2. **Sui dati**

   Un dato necessario sta venendo ancora usato in un'istruzione precedente.

3. **Strutturali**

   Non sono sufficienti le risorse Hardware.

Per cercare di risolvere il problema si possono utilizzare diversi metodi:

- **Forwarding**

  Se il dato serve nell'istruzione successiva si può inserire una scorciatoia che permetta di passare il dato prima della fase WB.

- **Stallo**

  In caso non sia possiibile usare il Forwarding si inseriscono degli stalli che ritardano l'esecuzione di qualche ciclo.

- **Riordino**

  Viene cambiata la sequenza delle istruzioni per evitare gli stalli, bisogna però mantenere la semantica.

- **Anticipo**

  I salti incondizionati si possono riconoscere in una fase precedente risparmiando un'istruzione.

- **Branch Prediction**

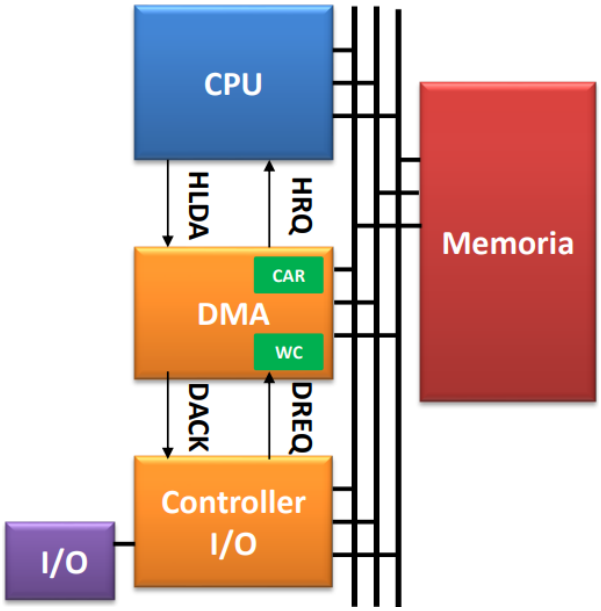
  La CPU osserva i salti e cerca di pre-caricare l'istruzione da eseguire.

6 DMA

La soluzione migliore per interfacciarsi con dispositivi I/O è l'accesso diretto alla memoria, tramite un apposito controllore è possibile trasferire dati da/a il dispositivo senza dover passare ogni volta per il processore.

Questo controllore contiene:

- Il registro **CAR** (Current Address Register)
  Funge da puntatore alla locazione di memoria.
- Il **WC** (Word Counter)
  Contiene la quantità di dati da trasferire.
- Un registro di stato
- Delle linee di comando per gestire il trasferimento dei dati



Per trasferire dei dati si segue il seguente procedimento:

1. Si accede alla locazione di memoria
2. Si trasferisce una parola tramite il Bus
3. Si incrementa/decrementa il CAR
4. Si decrementa il WC
5. Quando WC= 0 crea un interruzione per segnalare la fine del trasferimento

7 Memorie

A causa del costo e delle diverse prestazioni si stabilisce una gerarchia tra le diverse memorie usate in un elaboratore:

1. **Registri**
2. **Cache**
3. **RAM**
4. **Memorie non volatili**

Questa organizzazione a livelli permette ad un programmatore di avere una memoria che sia veloce e capiente contemporaneamente.

7.1 RAM

La RAM è il tipo di memoria più usata come memoria centrale, ne esistono 2 tipi:

SRAM	**DRAM**
Accesso rapido	Accesso più lento
Costo alto	Costo minore
Maggiore consumo elettrico	Minore consumo elettrico
Circuiti complessi	Circuiti semplici
Bassa densità delle celle	Alta densità delle celle

Tabella 2: Confronto tra RAM

7.2 Cache

L'inserimento di questa memoria si basa sul principio di località:

- Località **temporale**

  Quando si usa un elemento in memoria molto spesso viene riutilizzato a breve.

- Località **spaziale**

  Quando si usa un elemento in memoria molto spesso vengono usati gli elementi ad esso vicini.

Fisicamente è organizzata in linee:

V	TAG	DATA

I campi presenti sono:

- **V**

  Indica se la linea contiene dei dati.

- **TAG**

  Identifica quale parola della memoria centrale è nella linea.

- **DATA**

  La parola effettiva.

Quando il processore richiede un certo dato viene controllata la Cache prima della memoria centrale, se non presente il valore viene anche caricato nella Cache e questo permette di diminuire il tempo medio per trovare un dato.

Quando un dato nella Cache viene modificato bisogna aggiornare la memoria di livello inferiore, ci sono 2 modi:

- **Write Through**

  Il blocco in memoria viene aggiornato ad ogni modifica.

- **Write Back**

  Il blocco in memoria viene aggiornato solo quando viene sostituito.

7.2.1 Tipi

Esistono diversi tipi di Cache, la differenza sta nel come fanno corrispondere un indirizzo locale con uno della memoria centrale:

- **Direct Mapped**

  $$\text{Ind. parola in Cache} \;=\; \text{Ind. parola in memoria} \mod (\text{Num. linee Cache})$$

- **Associativa a n-vie**

  Vengono definiti un certo numero x di insiemi di n parole, per trovare l'insieme in cui si trova una parola:

  $$\text{Insieme} \;=\; \text{Ind. parola in memoria} \mod x$$

- **Completamente associativa**

  Una parola può essere inserita in qualsiasi posizione.

7.2.2 Politiche di rimpiazzo

Quando si verifica un MISS e la Cache è piena bisogna stabilire (tranne nella Direct Mapped) quale degli elementi presenti può essere sostituito con il nuovo:

- **Sostituzione casuale**
- **LRU** (Least Recently Used)

  Si usa un opportuno contatore che tiene traccia del tempo che un elemento passa nella Cache, si rimuove quello più "vecchio".

- **LFU** (Least Frequently Used)

  Si usa un opportuno contatore che tiene traccia di quante volte un elemento è stato usato, quello usato meno frequentemente viene rimosso.

7.3 Allocazione dei processi

Definizione Un processo è un programma presente in memoria centrale e che è in corso di esecuzione.

Ogni processo ha un **PCB** (Process Control Block) associato che contiene:

- Lo stato del processo
- Il suo program counter
- I valori nei registri
- Informazioni sullo scheduling
- Informazioni sulla memoria
- Informazioni di contabilità
- Informazioni I/O

Considerando la quantità di processi normalmente attivi c'è necessità di allocare la memoria in modo efficiente, ci sono svariati metodi:

- A partizione **singola**

  Ci può essere un solo processo allocato alla volta.

- A partizioni **multiple**

  - A dimensione **fissa**

    Quando un programma deve essere caricato si crea una partizione.
  - A dimensione **variabile**

    Una partizione viene allocata dinamicamente in base al processo.

  In entrambi i casi è presente un problema definito frammentazione (risolvibile tramite la compattazione):

  - **Interna**

    Viene allocata della memoria eccessiva per ogni processo e in presenza di molti diventa uno spreco importante di memoria.
  - **Esterna**

    C'è abbastanza memoria per un processo ma non è contigua.

Nel caso un programma richieda più memoria della massima partizione possibile si può ricorrere all'*Overlay*, esso permette di caricare in memoria solamente la parte di programma attualmente usata e scambiarla con opportune operazioni quando necessario (poco efficiente a causa degli scambi).

7.3.1 Memoria virtuale

Definizione La memoria virtuale è una tecnica che consente di eseguire programmi anche se non possono essere contenuti pienamente della memoria.

Per funzionare usa una tecnica chiamata *Paginazione* che permette ad un programma di avere uno spazio degli indirizzi non per forza contiguo, funziona dividendo la memoria in "frame" ed il programma in "pagine" e caricando poi quest'ultime nei frame quando necessario.

Facendo così si ottiene uno spazio di indirizzi logici, essi sono tradotti in indirizzi fisici usando la tabella delle pagine:

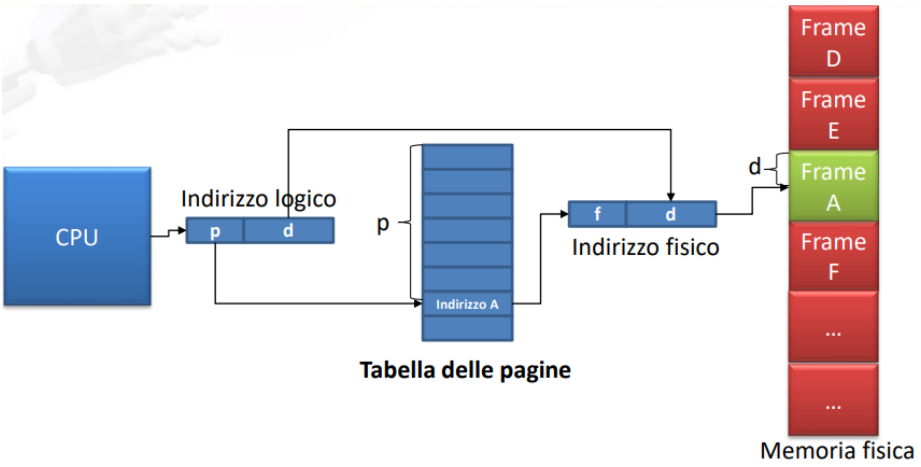


Figura 11: Traduzione degli indirizzi

Un esempio di funzionamento è il seguente:

1. Il processore richiede un indirizzo logico
2. Viene cercata la pagina nella tabella
3. Non è presente, si genera un'interruzione
4. Controllando il PCB si cerca l'indirizzo, se è valido passa al punto successivo
5. Si cerca un frame libero
6. Viene caricata la pagina in quel frame tramite I/O e si aggiornano le tabelle
7. L'istruzione viene riavviata

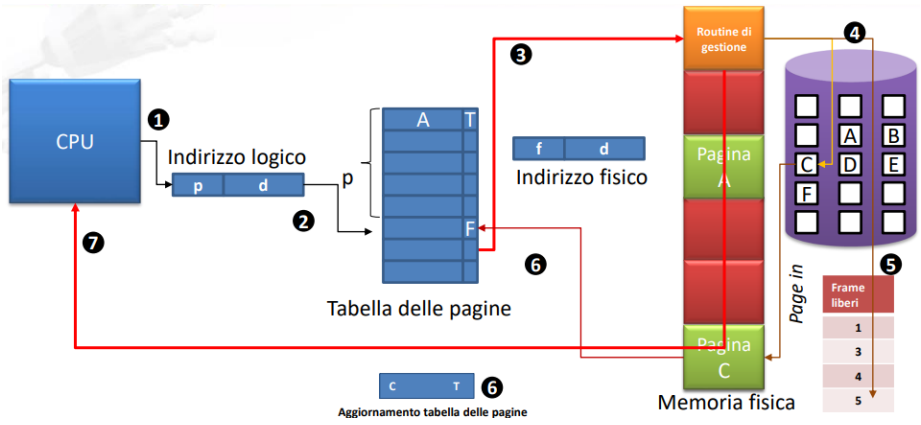


Figura 12: Rappresentazione grafica

Come si è visto se una pagina non è presente in memoria va caricata dalla memoria secondaria (On-Demand Paging), in alternativa è possibile usare il *Prepaging* che consiste nel caricare a "blocchi" le pagine contigue.

Nel caso ci sia bisogno di inserire una pagina nella memoria piena si adottano diversi metodi per scegliere il rimpiazzo:

- **LRU** (Come per la Cache)
- **FIFO**

  Si scambia con la pagina presente in memoria da più tempo.
- **CLOCK**

  Le pagine vengono organizzate in una coda circolare insieme ad un bit che indica se sono state usate, quando bisogna scambiare si ispeziona tutta la coda e:

  – Se il bit è 1 viene settato a 0

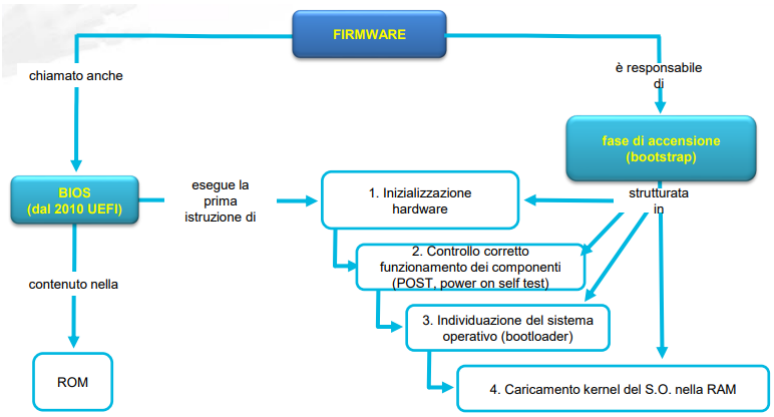
  – Se il bit è 0 viene sostituita la pagina
- **OPT** (Soluzione ottimale utopistica)

8 Sistemi operativi (Fondamenti)

Definizione Il Firmware è un programma eseguibile che sia avvia automaticamente dopo l'accensione della macchina, tipicamente risiede su una EEPROM.

I principali compiti che svolge sono:

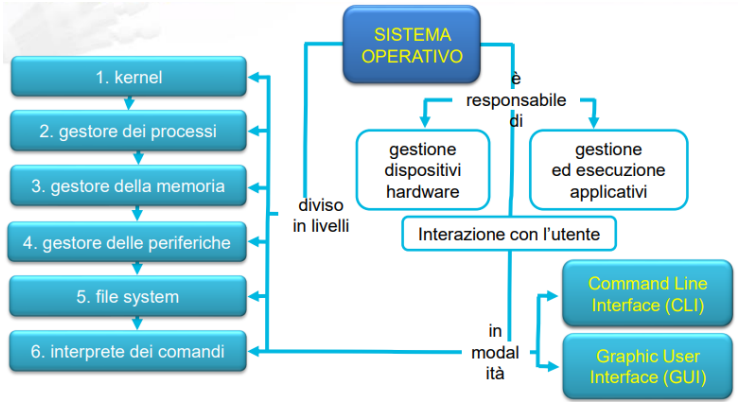
1. Controllare il funzionamento dei vari componenti
2. Eseguire il *Bootloader*, ossia caricare in memoria il Kernel (Nucleo) del Sistema Operativo



Una volta caricato il S.O. rimane sempre in esecuzione ed il suo Kernel è sempre in memoria centrale.

Definizione Il sistema operativo è un insieme di programmi che:

- Gestisce le componenti fisiche dell'elaboratore
- Fornisce un ambiente in cui è possibile installare, gestire ed eseguire i programmi applicativi
- Permette l'interazione utente-elaboratore senza che esso conosca tutti i dettagli dell'architettura



Essenzialmente deve gestire interamente la macchina svolgendo tutti i compiti:

Figura 13: Compiti principali

9 Architetture Multiprocessore e multicore

Definizione I sistemi multiprocessore e multicore sono rispettivamente definiti come:

- "Sistema con 2 o più processori operanti in parallelo"
- "Sistema composto da 2 o più core (nucleo del processore) nello stesso chip"

Entrambe sono architetture che permettono il calcolo parallelo, risulta utile dare una classificazione delle architetture viste fin'ora tramite la tassonomia di Flynn:

- **SISD**

  Quella classica, un solo processore.

- **MISD** (Instruction Level Parallelism)

  Sono presenti più processori che svolgono un istruzione ciascuno contemporaneamente sullo stesso dato.

- **SIMD** (Data Level Parallelism)

  Contiene più processori tra cui si individua quello "Master", esso rilascia un flusso di istruzioni che vengono eseguite una alla volta da ogni processore su dati diversi.

- **MIMD** (Thread Level Parallelism)

  Ogni processore è autonomo, ha la propria CU e la sua memoria locale.

9.1 Gestione della memoria

Nel caso multiprocessore è possibile avere una memoria:

- **Condivisa**
  - La memoria è la stessa per tutti i processori
  - Può accedere un solo processore alla volta
  - Eventuali zone comuni vanno gestite dal programmatore

  Ce ne sono 2 tipi:

  - **Uniform Memory Acess**
  - **Non Uniform Memory Acess**
    La memoria viene divisa in zona comune e zona veloce.
- **Distribuita**
  - La memoria è distribuita fisicamente tra i processori
  - Per comunicare tra loro i processori usano il *Message Passing*

  Ce ne sono 2 tipi:

  - **NO-Remote Memory Access**
    Tutte le memorie locali sono private.
  - **Cache Only Memory Access**
    Presenti solo le memorie Cache.

9.2 Gestione dei processi

Definizione Un thread è una suddivisione di un processo in 2 o più sottoprocessi.

Definizione Il multithreading è la capacità di un processore di passare da un thread all'altro.

I 3 tipi principali di Multithreading sono:

1. **Coars Grained MultiThreading**

   Si esegue un thread alla volta, appena si presenta uno stallo prolungato si passa ad un altro.

2. **Fine Grained MultiThreading**

   Ad ogni ciclo di clock vengono alternati i thread.

3. **Simultaneous MultiThreading**

   Una combinazione dei precedenti.

10 Memorie di massa

Le memorie di massa sono classificabili in base al tipo di accesso:

- **Sequenziale**
- **Diretto**
- **Indicizzato**

E al loro materiale di composizione:

- **Magnetico**
- **Ottico**
- **A stato solido**

Definizione Un documento digitale è una collezione di dati organizzata secondo un certo formato.

Definizione Un supporto digitale è una memoria su cui viene archiviato un documento digitale.

Definizione Un oggetto digitale è un supporto digitale contenente un documento digitale ed una sua descrizione.

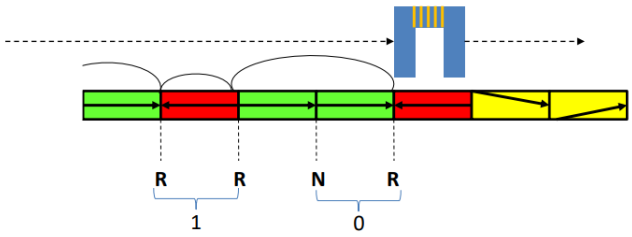
10.1 Nastro magnetico

Il nastro magnetico è formato da uno strato di particelle magnetiche depositate su un supporto flessibile.

I componenti principali sono:

- **Rivestimento dorsale**
- **Substrato**
- **Strato magnetico**
- **Legante**

Ogni particella ha un verso di magnetizzazione, un insieme di particelle può mutare il suo verso in presenza di un campo magnetico esterno. La scrittura quindi avviene sfruttando un elettromagnete per cambiare l'orientamento, la lettura invece leggendo le variazioni del campo magnetico.



Il nastro viene diviso in tracce e a loro volta in blocchi, un blocco odierno è formato da:

1. Preambolo
2. Marcatore di inizio
3. Sezione dati
4. Indirizzo del blocco
5. Informazioni per il rilevamento e la correzione di errori

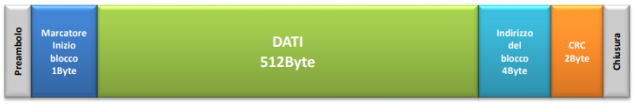


Figura 14: Blocco LTO

I principali tipi sono:

- Super Digital Linear Tape
- Super Advanced Intelligent Tape
- Linear Tape Open (STANDARD)

10.2 Disco magnetico

Il disco magnetico è formato da uno strato di particelle magnetiche depositate su un supporto fisso.

I componenti principali sono:

- **Substrato**
- **Strato magnetico**

2-3 dischi magnetici, il sistema di movimentazione e le testine sono sigillati in una custodia metallica a tenuta stagna chiamata Hard-Disk:

Figura 15: Hard-Disk

Il funzionamento è simile al nastro magnetico.

Il disco ha 2 facce, una faccia viene divisa in tracce e a loro volta in settori, più settori contigui formano un cluster. La quantità di dati nel settore può essere variabile o costante (nelle parti esterne ce ne sono di più).

Una traccia è formata da:

1. Preambolo
2. Sincronizzazione
3. Indirizzo del blocco
4. Sezione dati
5. Informazioni per il rilevamento e la correzione di errori

Figura 16: Blocco

10.3 Disco ottico

Il disco ottico ha una struttura composta da materiali eterogenei.

I componenti principali sono:

- **Substrato**
- **Strato dati**
- **Strato riflettente**
- **Strato protettivo**

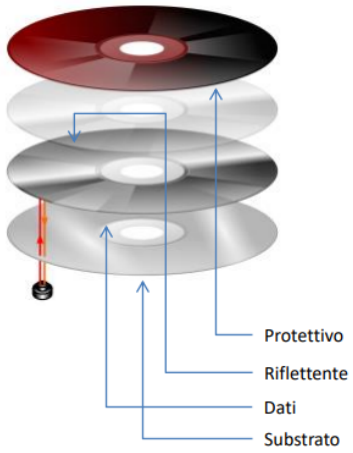


Figura 17: Struttura

I dati vengono rappresentati da depressioni (pit) e spianate (land) sulla superficie e sono disposti a spirale, essa viene divisa in settori. Essendo nati per conservare la musica sono strutturati in modo da avere una densità dei dati costante che permette una velocità di lettura lineare costante.

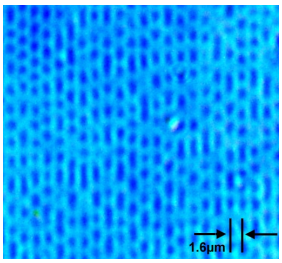


Figura 18: Superfice al microscopio

Per essere letti e scritti viene usato un laser che seguendo la traccia viene riflesso in modo variabile a causa della superficie e letto da un fotodiodo:

- La transizione da pit a land (o viceversa) indica un 1
- Le parti "piatte" indicano una serie di 0

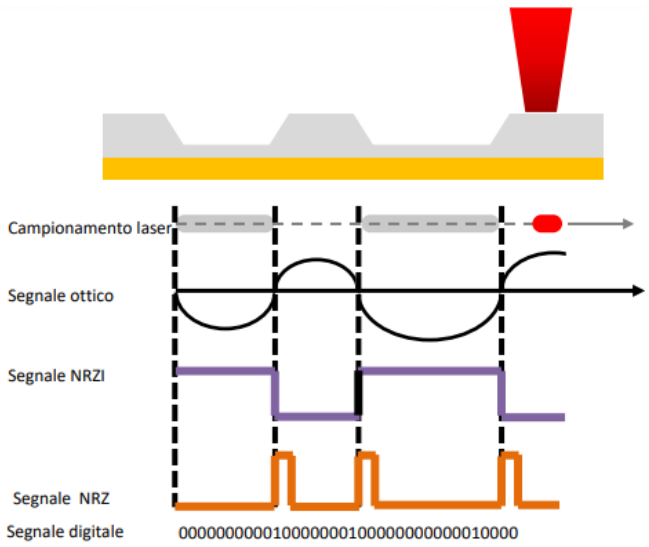


Figura 19: Lettura

La scrittura varia in base al tipo di disco, partendo da un file ISO:

- **Sola lettura** (CD-ROM,DVD-ROM,BR-ROM)

  Si imprimono i pit su un disco master (in vetro o zinco) e poi si effettua uno stampo versando la base in policarbonato allo stato fuso.

- **Registrabili una volta** (CD-R,DVD-R,BR-R)

  Si brucia una pellicola organica grazie al masterizzatore, la fusione della pellicola (che copre lo strato riflettente) produce delle aree riflettenti che vanno a ripetere la condizione di opacità e riflettività dei pit e dei land.

- **Riscrivibili** (CD-RW,DVD-RW,BR-RW)

  Si usa del materiale cristallino, quando il laser è alla massima potenza il materiale si scoglie creando una zona amorfa e perdendo di riflettività, questa differenza con il materiale "normale" va a simulare i pit e i land.

  Per farlo ritornare vuoto basta scaldare le zone amorfe a circa 200°C, questo fa ritornare la riflettività del materiale.

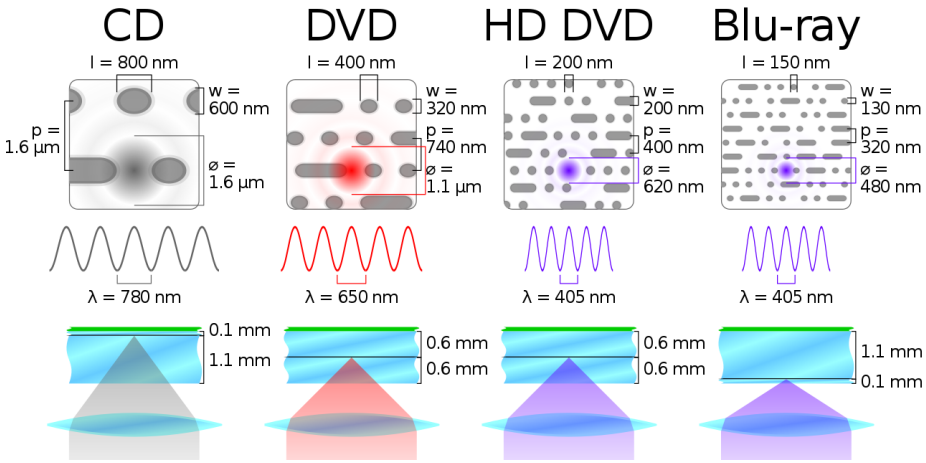
Inserendo una tabella dei contenuti nella parte centrale è possibile migliorare la ricerca dei file sul disco indicandone la posizione.

Una particolarità è l'uso della modulazione Eight-to-fourteen (nei CD), la sua caratteristica è il trasformare un byte in una parola di 14 bit avente gli 1 separati da 2-10 0.

I principali tipi sono:

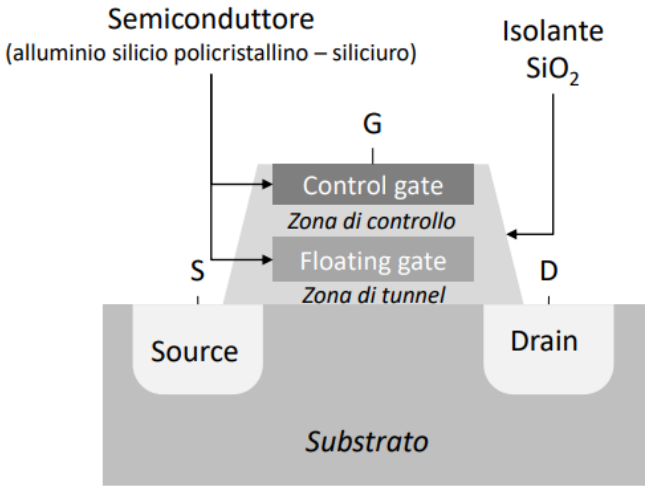
- **CD**
- **DVD**
- **Blu-ray**
- **HVD**

Ogni tipo usa laser e lenti diverse:



10.4 A stato solido

Le memorie a stato solido sono delle matrici di celle di memoria, nella maggior parte si usa un transistor a griglia fluttuante per mantenere l'informazione:



Per scrivere si applicano 7V al Drain e 12V al Control Gate, per cancellare invece 9V al Gate e 6V alla Source.

Per leggere si applica 1V al Drain e se è presente un flusso di elettroni allora vale 1.

Le celle si raggruppano in chunk, i chunk in pagine e le pagine in blocchi, i dati vengono recuperati con accesso casuale.

Data la struttura la scrittura/modifica/cancellazione può avvenire solamente a blocchi, se una pagina deve essere cancellata sarà solamente contrassegnata come non valida. Questo porta inevitabilmente alla frammentazione interna, per ovviare al problema la memoria svolge periodicamente dei controlli tramite *Garbage Collection*.

Ogni volta che una cella viene scritta/cancellata la sua zona di tunnel si degrada, per migliorare l'aspettativa di vita della memoria si usa l'*Overprovisioning* mantenendo uno spazio non allocato, in questo modo si riesce a garantire un numero di celle sostituibili adeguato.

10.5 Digital Preservation

Definizione La preservazione digitale è l'insieme di attività volte a garantire la durata nel tempo e la conservazione delle informazioni in formato digitale.

Ogni tipo di memoria di massa può essere adatta o meno per preservare i dati nel lungo periodo:

Supporto	Aspettativa di vita	Capacità di archiviazione	Affidabilità progettuale	Obsolescenza tecnologica
Nastro magnetico	Lunga	Grande	Elevata	Bassa
Disco magnetico	Medio-bassa	Grande	Medio-bassa	Medio-elevata
Disco ottico	Medio-bassa	Media	Inesistente	Alta
Mem. stato solido	Bassa	Medio-alta	Scarsa	Bassa