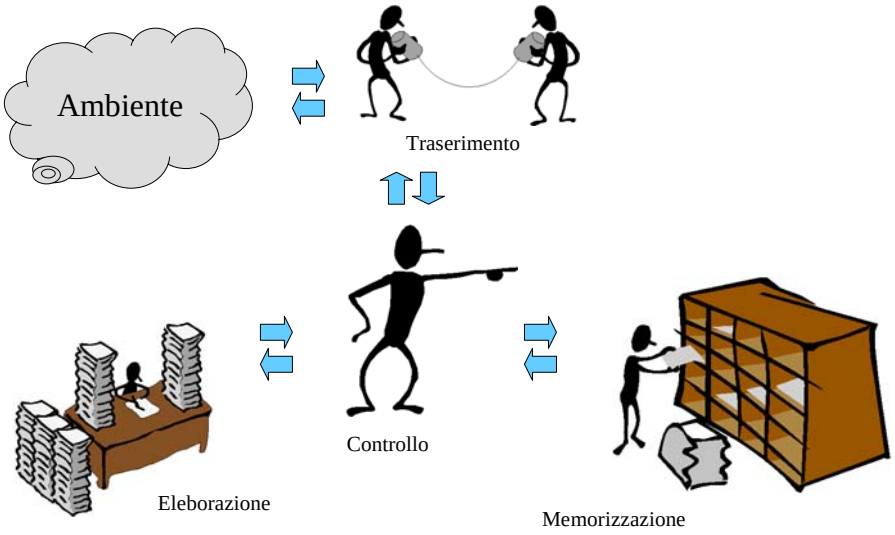
Introduzione all'Architettura dei Calcolatori Elettronici

Maurizio Palesi

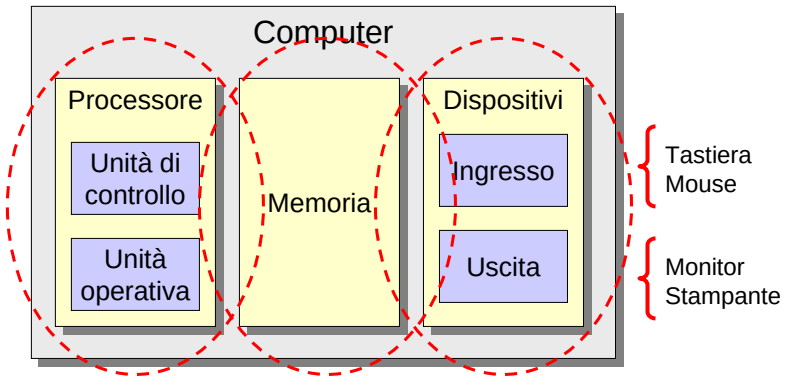
Caratteristiche Fondamentali

- Capacità di eseguire sequenze di istruzioni memorizzate
- **Calcolatore** = Unità di Elaborazione + Unità di Controllo
  - ➔1. Preleva le istruzioni dalla memoria
  - ➔2. Interpreta i codici di istruzione
  - ➔3. Effettua le azioni che questi prevedono
- **Programma** = Insieme organizzato di istruzioni

Vista funzionale di un Calcolatore



Componenti di un Computer

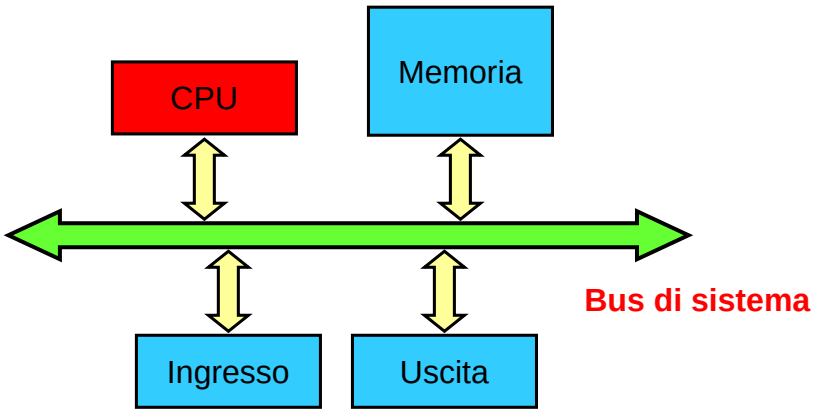


Unità di elaborazione e controllo o Central Processing Unit (CPU)

Contiene le istruzioni da eseguire e i dati su cui operare

Insieme di dispositivi che consentono la comunicazione con e da il mondo esterno

Organizzazione Generale



Bus e Master-Slave

- Il bus è una linea a cui sono contemporaneamente connesse le unità del calcolatore e che consente il trasferimento di dati tra tali unità
  - ➔Problema: contesa su un mezzo condiviso!
  - ➔Soluzione: CPU = master, periferiche = slave

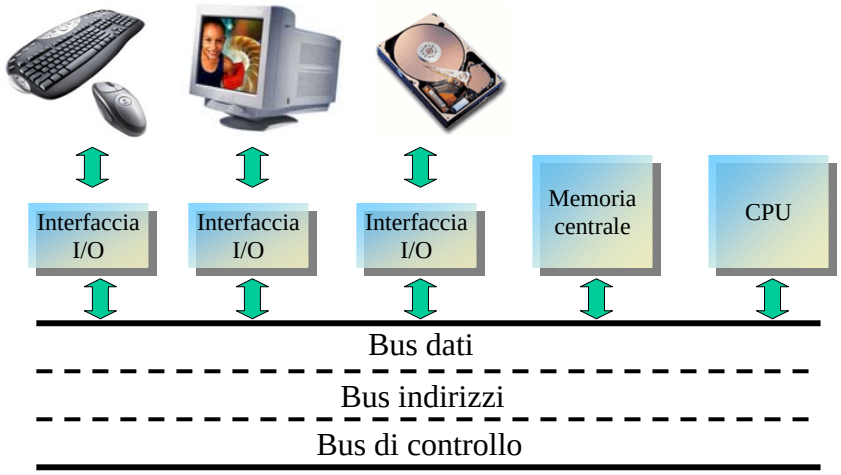
Bus e Master-Slave - Pregi

- Semplicità: 1 sola linea di connessione ∀ # di dispositivi
- Estendibilità: nuovi dispositivi possono essere aggiunti tramite un'interfaccia al bus senza influenzare l'HW preesistente
- Standardizzabilità: definizione di normative che consentono a periferiche di costruttori diversi di interagire correttamente

Bus e Master-Slave - Difetti

- Lentezza: l'uso in mutua esclusione del bus inibisce almeno parzialmente la parallelizzazione delle operazioni di trasferimento di dati tra dispositivi
- Limitata capacità: al crescere del numero di dispositivi la presenza di una sola linea comporta un limite alla capacità di trasferire dati
- Sovraccarico della CPU: l'unità centrale viene coinvolta in tutte le operazioni di trasferimento di dati

Lo Schema di Riferimento

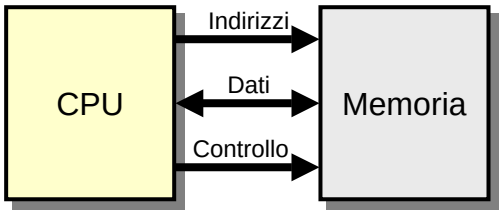


Tipi di Bus

- Bus dati: utilizzato per trasferire dati (es. fra memoria e CPU, fra CPU e interfacce di I/O)
- Bus indirizzi: che identifica la posizione delle celle di memoria un cui la CPU va a scrivere o leggere
- Bus di controllo: in cui transitano i segnali di controllo che consentono di selezionare le unità coinvolte in un trasferimento dati (sorgente e destinazione), di definire la direzione dello scambio (scrittura o lettura)

Architettura di Von Neumann

- Burks, Goldstein e Von Neumann sono stati i primi a proporre che il codice del programma potesse essere memorizzato nella stessa memoria dei dati



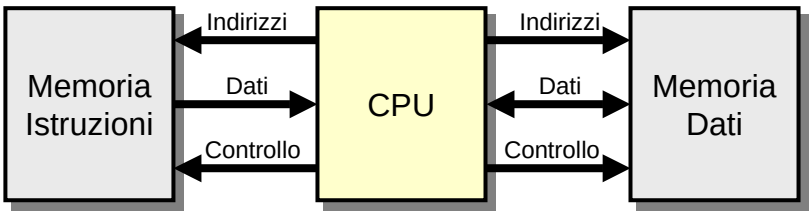
- Memoria indifferenziata per dati o istruzioni
- Solo l'interpretazione da parte di CPU stabilisce se una data configurazione di bit è da riguardarsi come un dato o come un'istruzione

Collo di Bottiglia Von Neumann

- L'organizzazione di Von Neumann è quella più popolare
- Consente al processore di manipolare i programmi in modo più semplice
- **Svantaggi**
  - ➔ La limitata larghezza di banda della memoria ha un'impatto negativo sulla velocità di esecuzione dell'applicazione
  - ➔ Questo fenomeno è noto come "Von Neumann bottleneck"

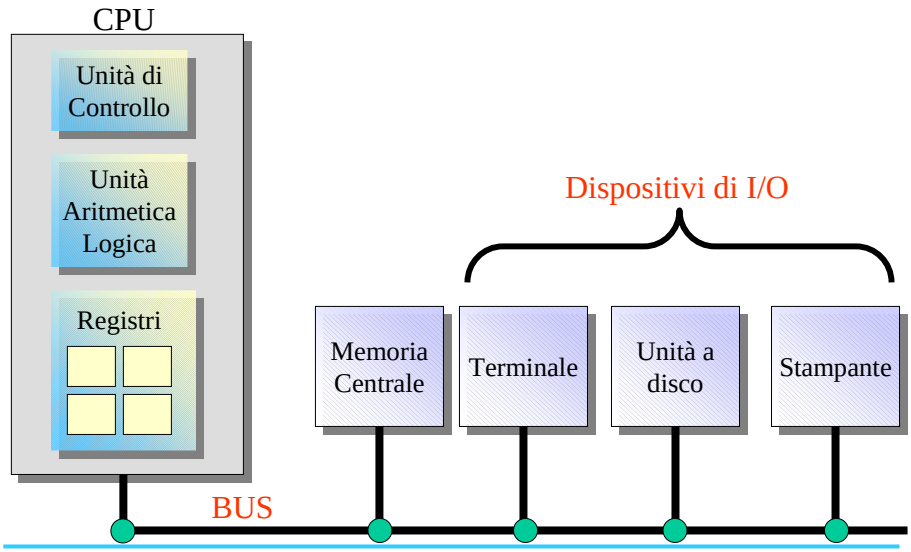
Architettura Harward

- Altre organizzazioni memorizzano dati e programmi in memorie diverse



- E' principalmente utilizzata nei processori ad alte prestazioni e nelle architetture dedicate per applicazioni di elaborazione digitale dei segnali (DSP)

Organizzazione Tipica (bus oriented)



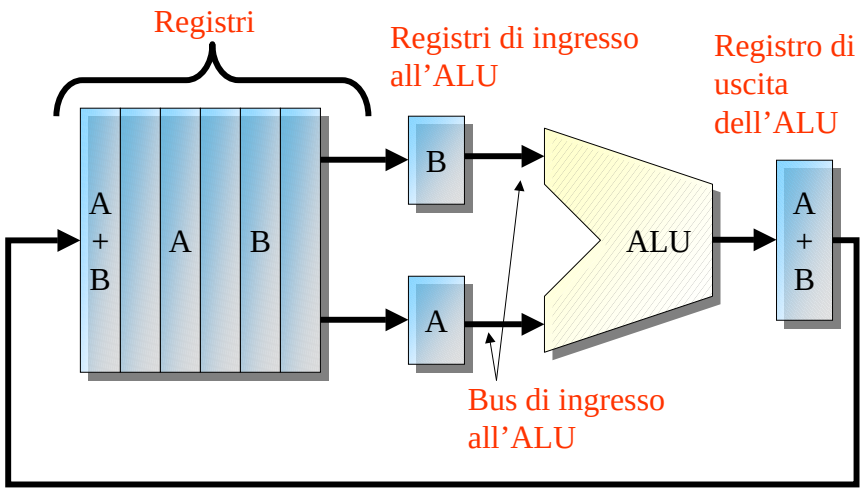
Tre Tipologie di Istruzioni

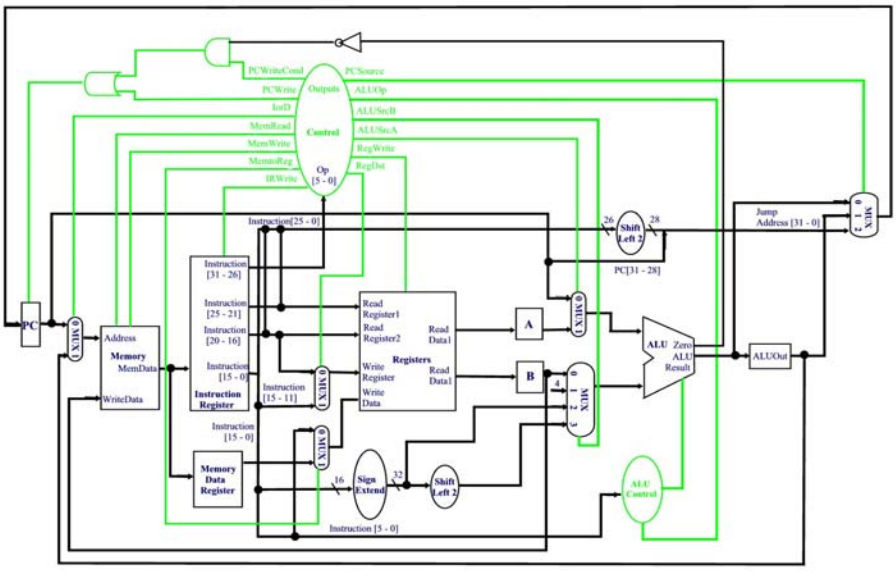
- Istruzioni Aritmetico Logiche (Elaborazione dati)
  - Somma, sottrazione, divisione, …
  - And, Or, Xor, …
  - Maggiore, minore, uguale, maggiore uguale, …
- Controllo del flusso delle istruzioni
  - Sequenza
  - Selezione
  - Ciclo a condizione iniziale, a condizione finale, …
- Trasferimento di informazione
  - Trasferimento dati e istruzioni tra CPU e memoria
  - Trasferimento dati e istruzioni tra CPU e dispositivi di I/O

Elementi di una CPU

- Unità di controllo
  - Legge le istruzioni dalla memoria e ne determina il tipo
- Unità aritmetico-logica
  - Esegue le operazioni necessarie per eseguire le istruzioni
- Registri
  - Memoria ad alta velocità usata per risultati temporanei
  - Determina il parallelismo della CPU
  - Esistono registri generici e registri specifici
    - ✓ Program Counter (PC)
    - ✓ Instruction Register (IR)
    - ✓ …

Struttura del "data path"





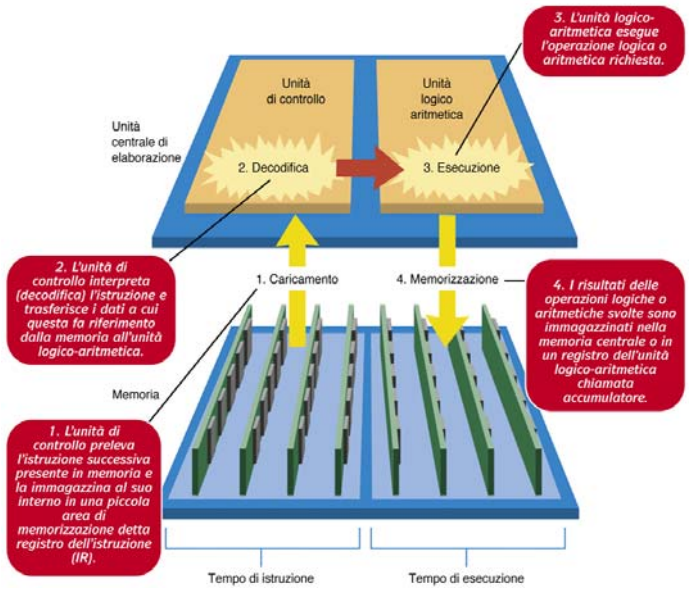
L'Esecutore

- Un calcolatore basato sull'architettura di Von Neumann esegue un programma sulla base dei seguenti principi
  - ➔Dati e istruzioni sono memorizzati in una memoria unica che permette sia la scrittura che la lettura
  - ➔I contenuti della memoria sono indirizzati in base alla loro posizione
  - ➔Le istruzioni vengono eseguite in modo sequenziale

Linguaggio Macchina e Assembly

- Linguaggio macchina
  - ➔Rudimentale
  - ➔Il concetto di tipo di dato è quasi assente
  - ➔Il numero di operandi è limitato
  - ➔Il numero di operazioni previste è ridotto

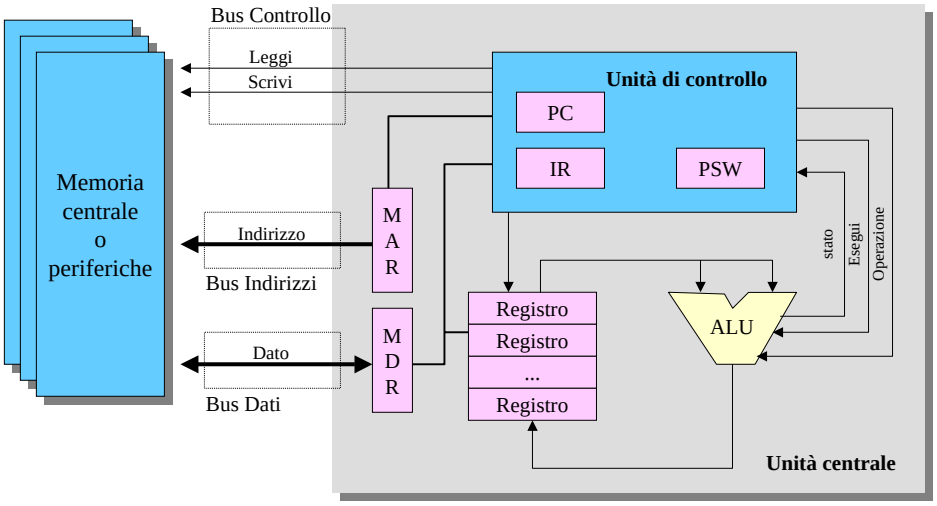
Struttura di una istruzione della CPU	codice operativo	op1	op2
Specifica analoga alla codifica in assembly	SOMMA	Reg1	Reg2
Codifica in un ipotetico linguaggio macchina	10000011	001	010



Esecuzione delle Istruzioni

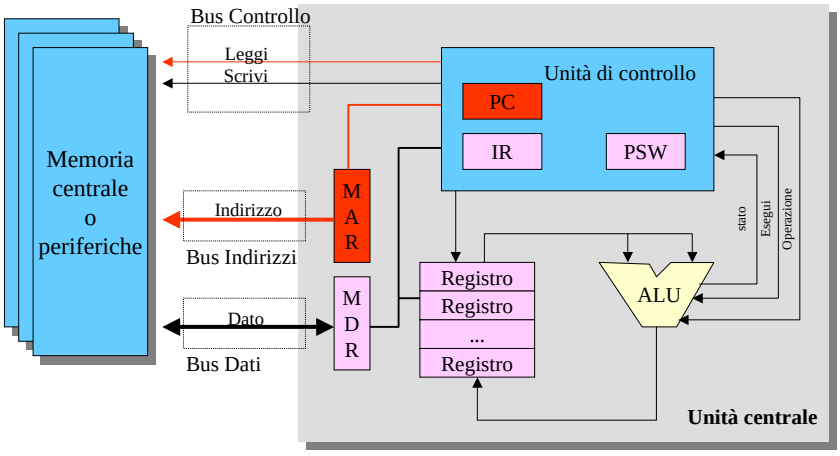
- Ciclo Fetch-Decode-Execute
  - Prendi l'istruzione corrente dalla memoria e mettila nel registro istruzioni (**IR**) [Fetch]
  - Incrementa il program counter (**PC**) in modo che contenga l'indirizzo dell'istruzione successiva
  - Determina il tipo dell'istruzione corrente [Decodifica]
  - Se l'istruzione usa una parola in memoria determina dove si trova
  - Carica la parola, se necessario, in un registro della CPU
  - Esegui l'istruzione [Execute]
  - Torna al punto 1.

Struttura Semplificata di una CPU



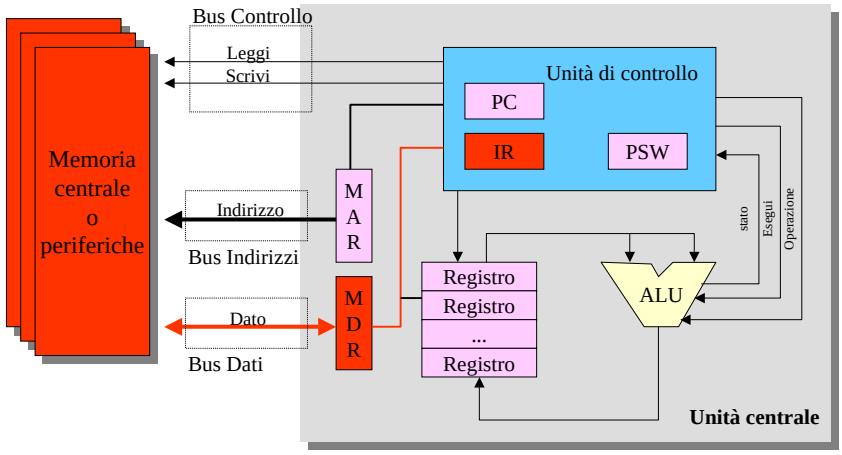
Esempio: Lettura dalla Memoria

- Fase di Fetch (1 di 2)



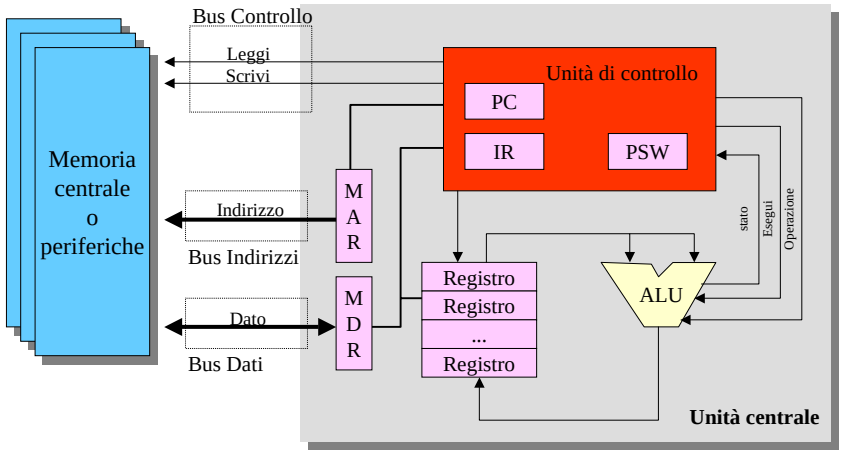
Esempio: Lettura dalla Memoria

- Fase di Fetch (2 di 2)



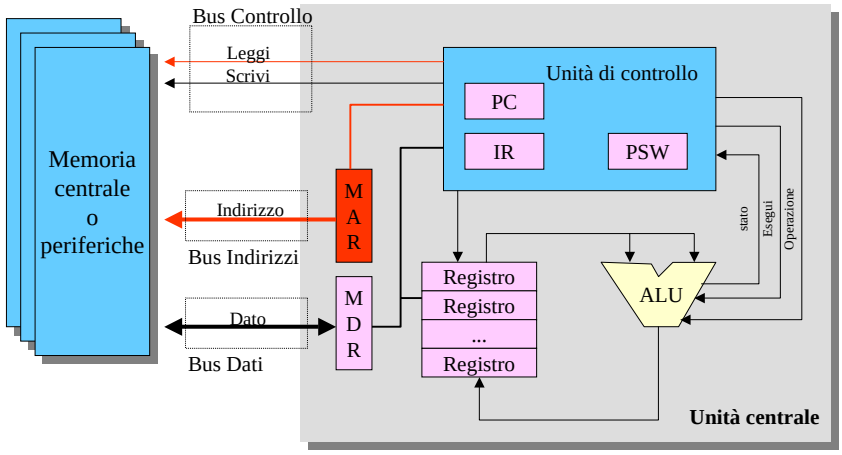
Esempio: Lettura dalla Memoria

- Decodifica



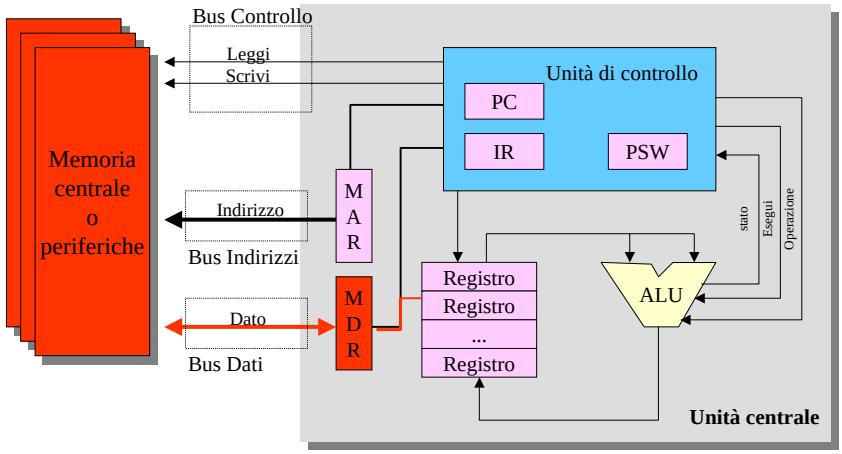
Esempio: Lettura dalla Memoria

- Esecuzione (1 di 2)



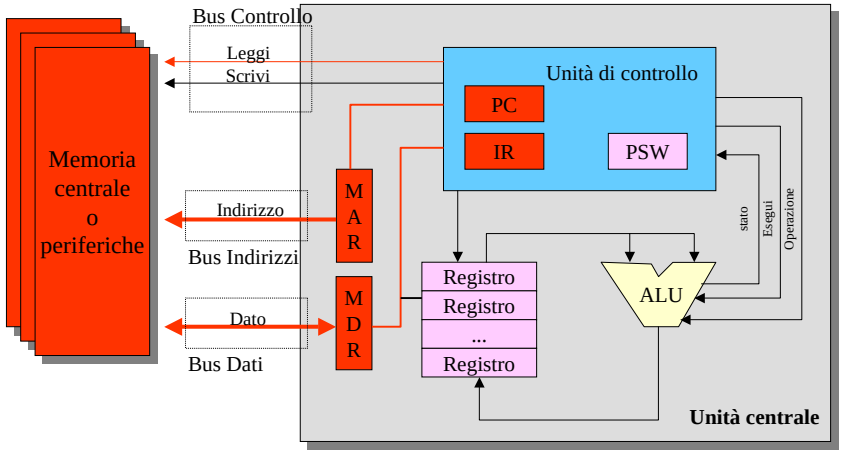
Esempio: Lettura dalla Memoria

- Esecuzione (2 di 2)



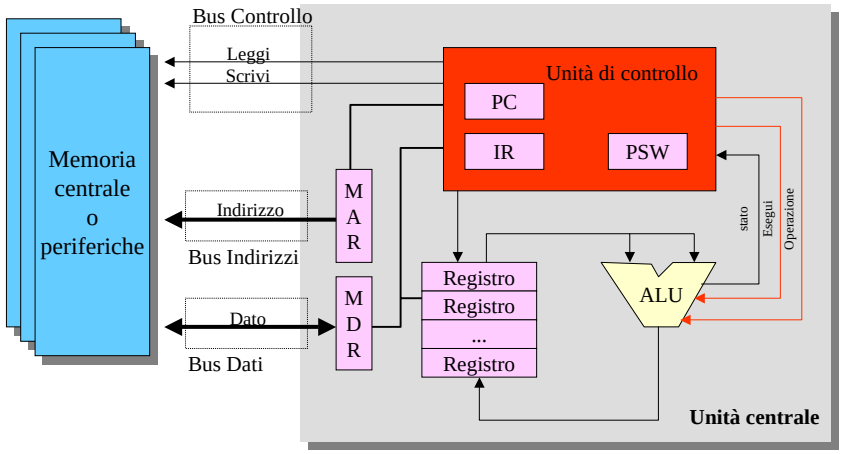
Esempio: Somma tra due registri

- Fetch (come prima)



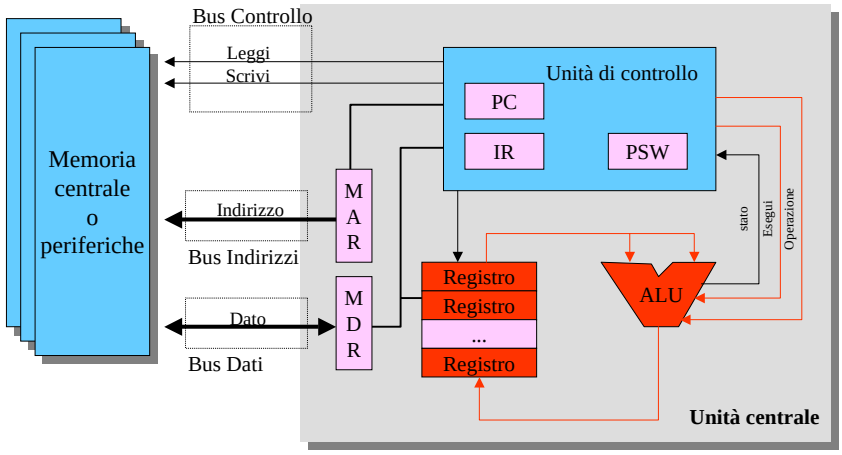
Esempio: Somma tra due registri

- Decodifica

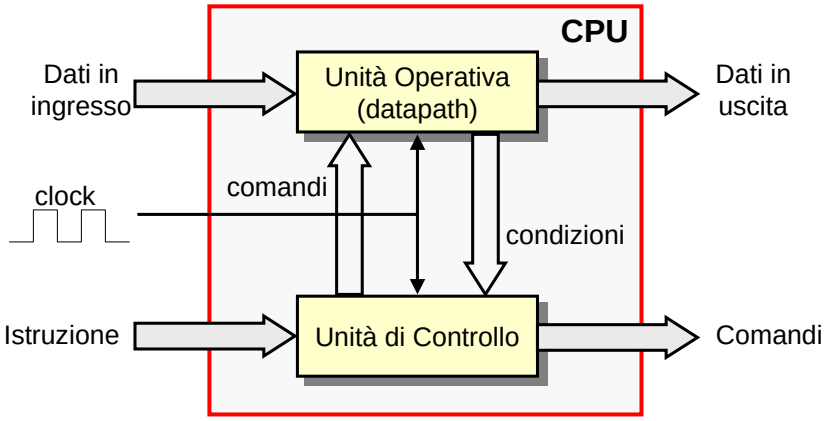


Esempio: Somma tra due registri

- Esecuzione

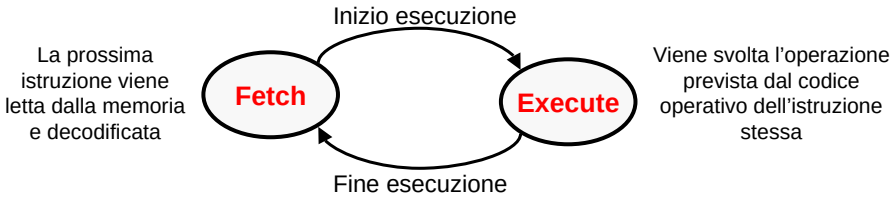


Struttura di una CPU



Fetch-Esecuzione

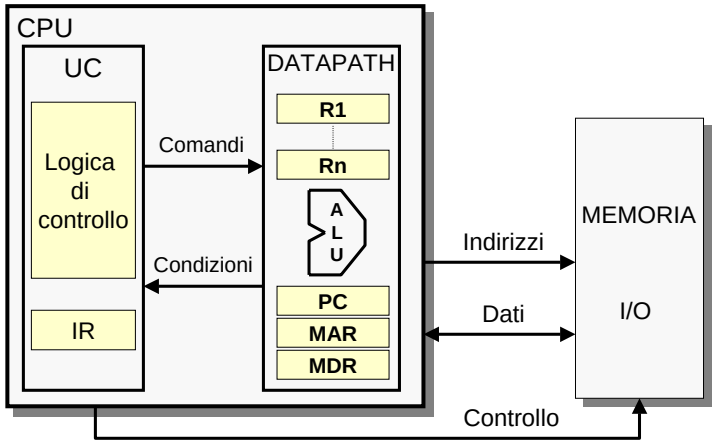
- Fetch
  - ➔ Prelievo e decodifica dell'istruzione
  - ➔ Fase comune a tutte le istruzioni
- Esecuzione
  - ➔ Fase in cui vengono eseguite le azioni previste dal codice di operazione
  - ➔ Fase diversa da istruzione a istruzione



Il Programma

- Programma = Sequenza di istruzioni
- Le istruzioni sono in memoria a indirizzi contigui
- Occorre un registro per memorizzare l'indirizzo della prossima istruzione da eseguire
  - ➔ Usualmente denominato Program Counter (PC)
- A termine dell'esecuzione di un'istruzione, PC deve puntare alla prossima istruzione
  - ➔ Le istruzioni sono a lunghezza fissa (stesso # di bytes)
    - ✓ PC è incrementato di una quantità pari a tale numero
  - ➔ Le istruzioni hanno lunghezza variabile
    - ✓ PC deve essere incrementato di volta in volta della dimensione in byte dell'istruzione appena eseguita
  - ➔ Le istruzioni di salto hanno l'effetto di aggiornare il PC con l'indirizzo di destinazione del salto

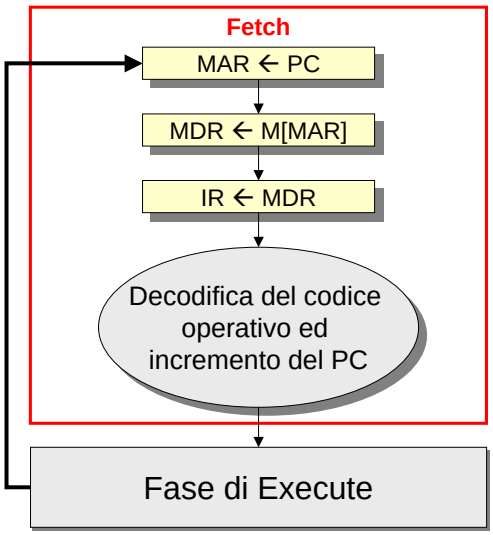
Elementi Fondamentali di una CPU



Registri di CPU

- **IR**: Usato per contenere l'istruzione in corso di esecuzione
  - Caricato in fase di fetch
  - Rappresenta l'ingresso che determina le azioni svolte durante la fase di esecuzione
- **PC**: Tiene traccia dell'esecuzione del programma
  - Contiene l'indirizzo di memoria in cui è memorizzata la prossima istruzione da eseguire
- **MAR**: contiene l'indirizzo della locazione di memoria da leggere o scrivere
  - La dimensione di MAR determina l'ampiezza dello spazio di memoria fisica
  - Dalla fine degli anni '80 vengono prodotti microprocessori con bus indirizzi a 32 bit
- **MDR**: Registro attraverso il quale viene scambiata l'informazione tra la memoria e la CPU
  - Tradizionalmente la dimensione di MDR dà la misura del grado di parallelismo della macchina (8, 16, 32, 64 bit)
- **R0, R1,...Rn**: Registri di uso generale

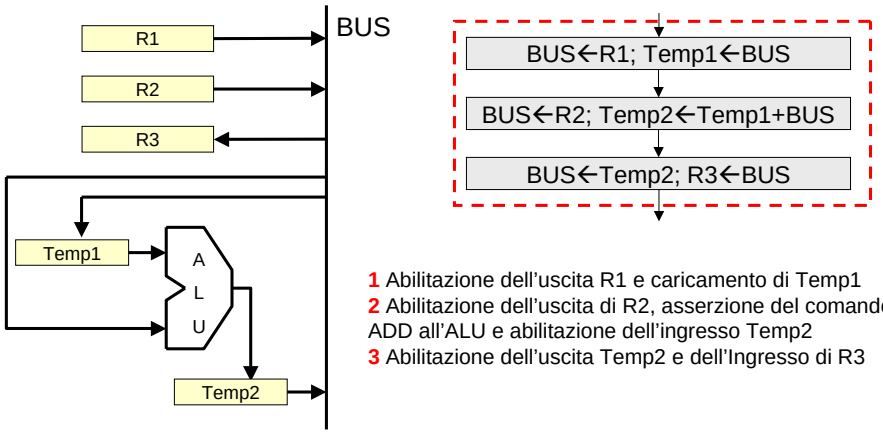
Diagramma a Stati della Fase di Fetch



Ogni stato ha durata di un periodo di clock, eccetto il secondo il quale può richiedere più cicli a causa della latenza della memoria

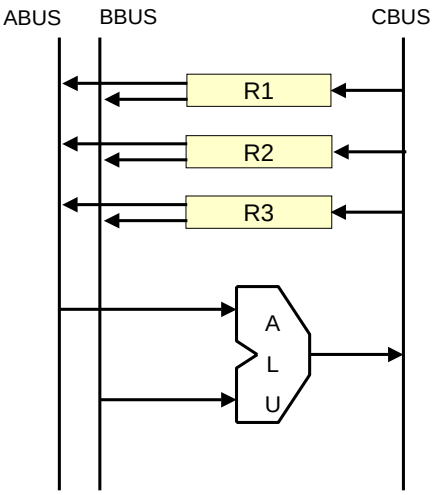
Fase di Execute (1 bus)

```
ADD R3, R1, R2
```

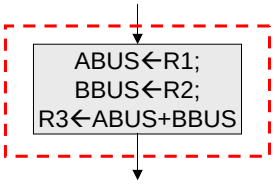


1 Abilitazione dell'uscita R1 e caricamento di Temp1
2 Abilitazione dell'uscita di R2, asserzione del comando ADD all'ALU e abilitazione dell'ingresso Temp2
3 Abilitazione dell'uscita Temp2 e dell'Ingresso di R3

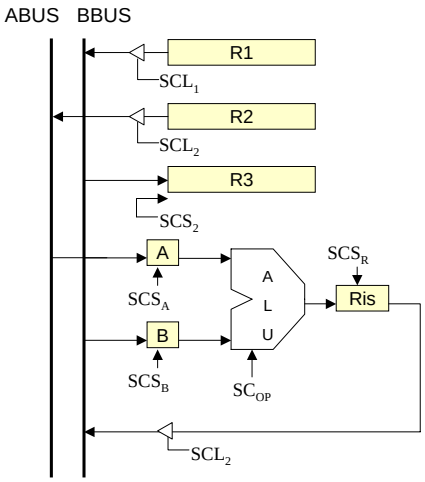
Fase di Execute (3 bus)



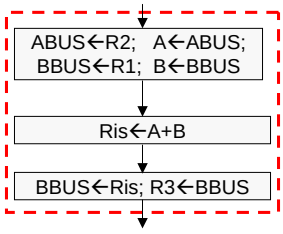
ADD R3, R1, R2



Fase di Execute (2 bus)

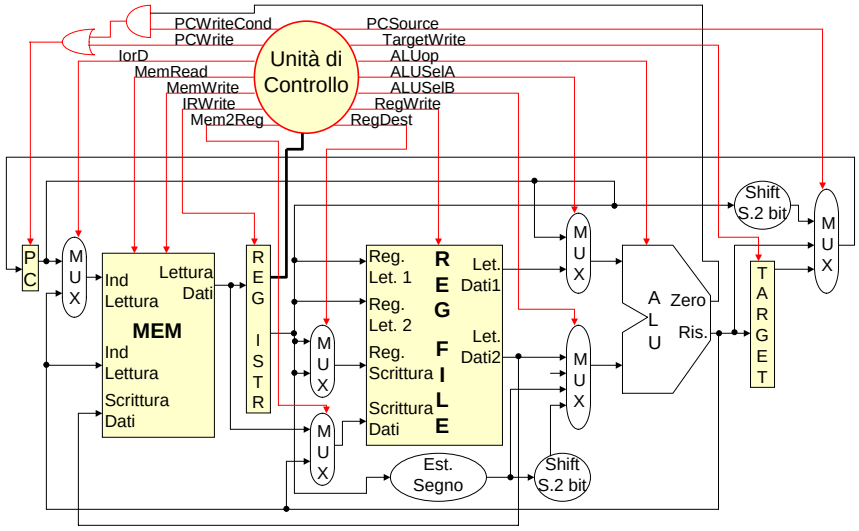


ADD R3, R1, R2



Nota: E' l'Unità di Controllo che ha il compito di pilotare i segnali di abilitazione SCL e SCS

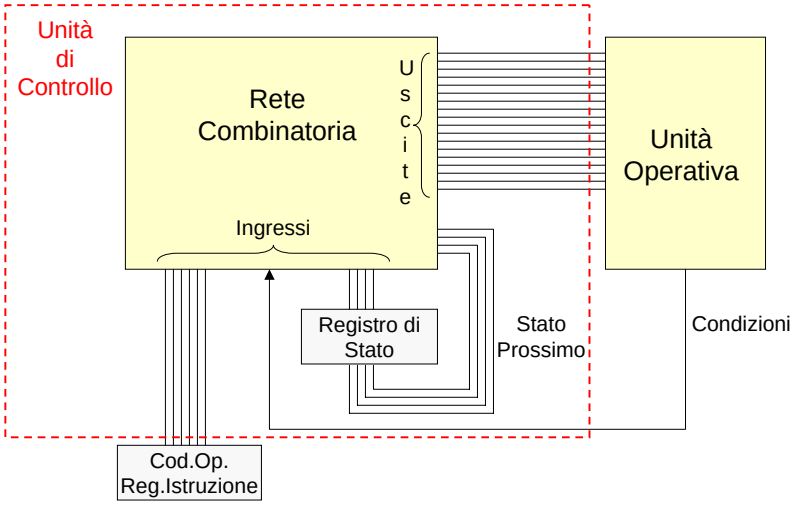
Controllo + Datapath



L'Unità di Controllo

- Realizzazione Cablata
- Realizzazione Microprogrammata

Unità di Controllo
Realizzazione Cablata

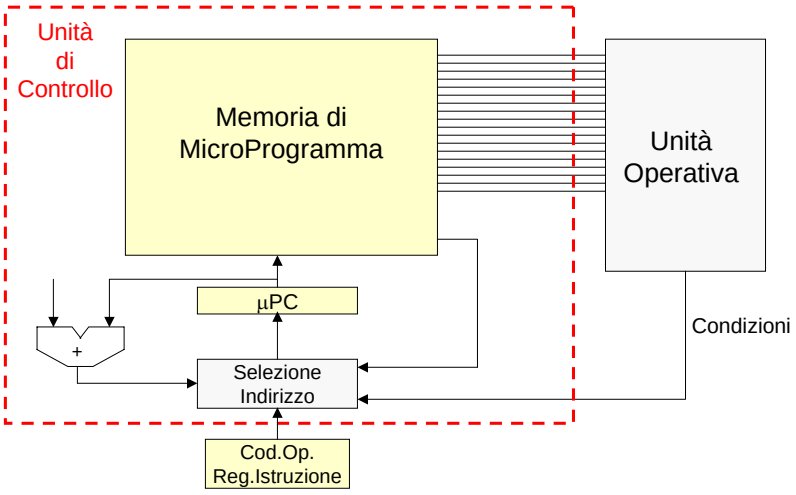


Unità di Controllo
Realizzazione Cablata

- Progettazione
  - ➔Seguendo il classico flusso di sintesi di una rete sequenziale
    - ✓Ingressi: IR, Stato
    - ✓Uscite: comandi, stato prossimo
  - ➔Uso di ROM
    - ✓Ingressi (indirizzi alla ROM): IR, stato di UO, stato di UC
    - ✓Uscite: comandi, ingressi di eccitazione dei FF di stato
- Misura della complessità di UC
  - ➔#stati × #ingressi × #uscite

Unità di Controllo

Realizzazione Microprogrammata



Unità di Controllo

Realizzazione Microprogrammata

- Tecnica affermatasi negli anni ‘70
- UC è una sorta di calcolatore nel calcolatore
- La memoria di controllo contiene le microistruzioni
- μPC: contatore di microprogramma
  - ➔ Contiene l’indirizzo della prossima microistruzione
  - ➔ All’inizio della fase di fetch μPC contiene l’indirizzo (I0) del tratto di microprogramma corrispondente al fetch
  - ➔ Alla fine della fase di fetch μPC viene aggiornato con il contenuto (o ad una opportuna decodifica) di IR in modo da puntare alla microroutine che effettua le azioni richieste dalla particolare istruzione
  - ➔ Al termine, μPC viene di nuovo caricato con (I0)

Unità di Controllo
Cablata vs. Microprogrammata

- Fino a fine anni '60: logica cablata (PDP8, HP 2116)
- Anni '70: microprogrammazione (VAX, Z80, 8086, 68000)
  - ➔ Repertorio di istruzioni molto esteso e variato: CISC
  - ➔ Il VAX 11/789 (Digital) e il 370/168 (IBM) avevano oltre 400.000 bit di memoria di controllo
- Dagli anni '80 si è tornati alla logica cablata
  - ➔ Affermazione delle macchine RISC
- Istruttivo è esaminare l'evoluzione dell'architettura Intel: da CISC a (praticamente) RISC

CISC

- **CISC**: **C**omplex **I**nstruction **S**et **C**omputing
- Un repertorio di istruzioni esteso è preferibile perché:
  - ➔ Istruzioni potenti semplificano la programmazione
  - ➔ Riduce il gap tra linguaggio di macchina e linguaggio di alto livello
  - ➔ L'uso efficiente della memoria (all'epoca era costosa) era la preoccupazione principale:
    - ✓ Meglio avere codici compatti
    - ✓ Essendo (allora) la memoria di controllo molto più veloce della memoria centrale, portare funzionalità nella prima avrebbe migliorato le prestazioni della macchina

RISC

- Memorie RAM
  - ➔ Molto più veloci delle precedenti a nuclei
- Cache
  - ➔ Riducono ulteriormente i tempi di esecuzione
- Comportamento dei programmi
  - ➔ L'80% delle istruzioni eseguite corrispondeva al solo 20% del repertorio
  - ➔ Conviene investire nella riduzione dei tempi di esecuzione di quel 20%, anziché aggiungere raffinate istruzioni, quasi mai usate, ma responsabili dell'allungamento del tempo di ciclo di macchina
- Conviene costruire processori molto veloci, necessariamente con repertori semplici, e contare sull'ottimizzazione del compilatore
- **RISC**: **R**educed **I**nstruction **S**et **C**omputing

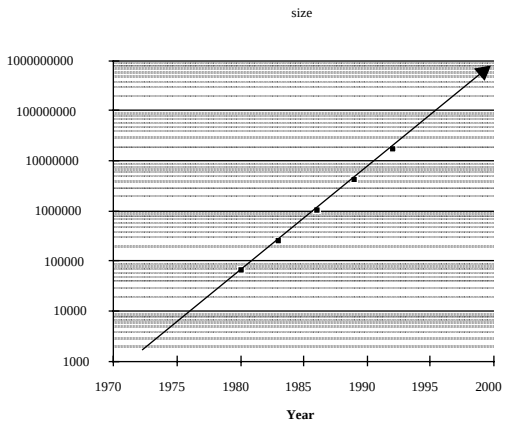
RISC - Criteri di Progettazione

- Le istruzioni devono essere semplici
  - ➔ Se l'introduzione di una operazione di macchina fa crescere del 10% il periodo di clock, allora essa deve produrre una riduzione di almeno un 10% del numero totale di cicli eseguiti
- Con memorie attuali
  - ➔ Non c'è vantaggio a spostare le funzionalità a livello di microcodice
    - ✓ Ciò ha solo l'effetto di rendere più difficoltose modifiche e cambiamenti
  - ➔ Molto meglio modificare una libreria di sistema che modificare una memoria di controllo

RISC - Criteri di Progettazione

- Tutte le istruzioni occupano lo stesso spazio di memoria (una parola)
- Ristretto numero di formati
  - ➔ L'interpretazione del codice avviene attraverso un semplice decodificatore (una rete AND-OR)
  - ➔ La codifica “ordinata” consente accorgimenti per velocizzare l'esecuzione (pipeline), difficilmente applicabili a repertori di istruzioni complesse
- La semplificazione del repertorio tende a far aumentare la dimensione del codice
  - ➔ Non è un problema, vista la tendenza alla riduzione dei costi e all'aumento della densità delle memorie
  - ➔ Dal punto di vista della velocità i guadagni che si ottengono nel semplificare le istruzioni sono superiori all'effetto negativo del maggior numero di istruzioni per programma

Capacità di Memoria (DRAM singolo chip)



Single Chip DRAM		
Year	Size	Cyc. Time
1980	64 Kb	250 ns
1983	256 Kb	220 ns
1986	1 Mb	190 ns
1989	4 Mb	165 ns
1992	16 Mb	145 ns
1996	64 Mb	120 ns
1999	256 Mb	100 ns
2002	1 Gb	80 ns

Incremento 1,4 per anno

4000X dal 1980

RISC - Criteri di Progettazione

- Conclusioni
  - ➔Progetto di un'architettura che preveda solo operazioni tra registri (non registro/memoria o memoria/memoria) e operazioni di lettura/scrittura in memoria molto semplici con poche modalità di indirizzamento
    - ✓Architetture Load/Store
  - ➔Il compilatore deve fare il miglior uso possibile dei registri e tenere il più possibile le variabili nei registri
    - ✓CPU con elevato numero di registri