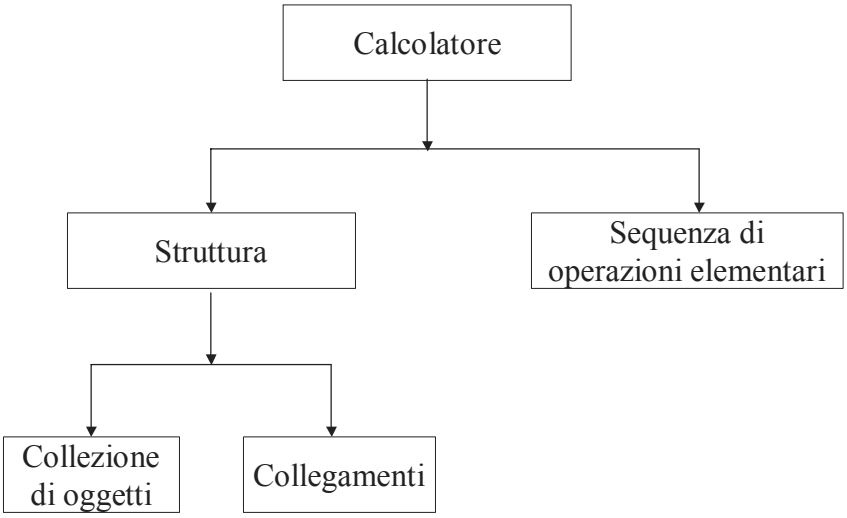
L'architettura di riferimento degli elaboratori

Architettura
degli Elaboratori e delle Reti
Turno I

Alberto Borghese
Università degli Studi di Milano
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

L'architettura di riferimento (Descrizione della collezione di oggetti)

Descrizione di un elaboratore



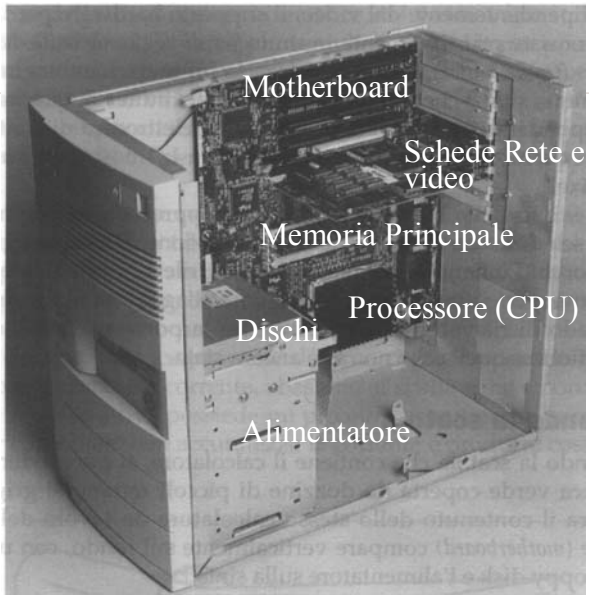
Architettura di riferimento degli elaboratori

Elabora in modo adeguato un input per produrre l'output.

Dispositivi di Input/Output

- Le unità di *ingresso* (tastiera del terminale video, mouse o altri dispositivi grafici di ingresso, ecc.) permettono al calcolatore di acquisire informazioni dall'ambiente esterno.
- Le unità di *uscita* (monitor grafico del terminale video, stampanti, ecc.) consentono al calcolatore di comunicare i risultati ottenuti dall'elaborazione all'ambiente esterno.

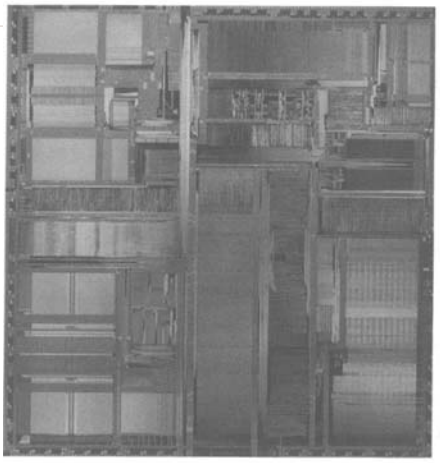
Cosa c'è dentro un elaboratore?

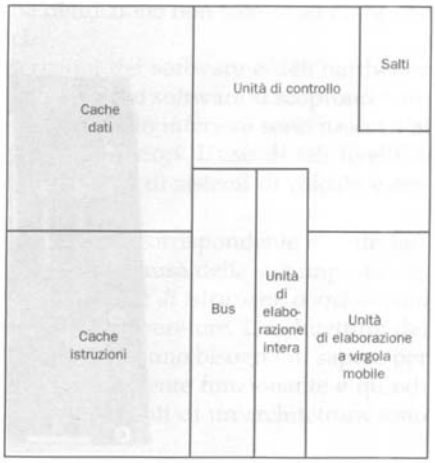


Architettura di riferimento degli elaboratori (Architettura di Von Neumann)

- Elementi principali di un calcolatore:
  - Unità centrale di elaborazione (*Central Processing Unit - CPU*)
  - Memoria di lavoro o memoria principale *(Main Memory - MM)*
- Sulla motherboard:menti principali di un calcolatore:
  - Bus di sistema (dati, indirizzi, controllo)
  - Interfacce per i dispositivi di *Input/Output - I/O*: il terminale, la memoria di massa (di solito dischi magnetici), le stampanti, …

Componenti di un processore Pentium

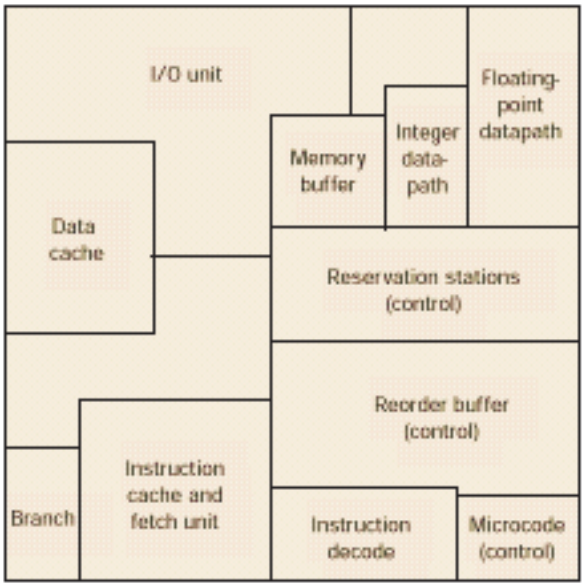




- 3,3 Milioni di transistor su 91mm^2

Il processore Pentium Pro

- 5,5 Milioni di transistor su 306mm^2 (2cm x 1.5cm) con cache esterna da 31 milioni.



Unità centrale di elaborazione *(Central Processing Unit - CPU)*

- La *CPU* provvede ad eseguire le istruzioni che costituiscono i diversi programmi elaborati dal calcolatore.
- Eseguire un'istruzione vuol dire operare delle scelte, eseguire dei calcoli a seconda dell'istruzione e dei dati a disposizione.

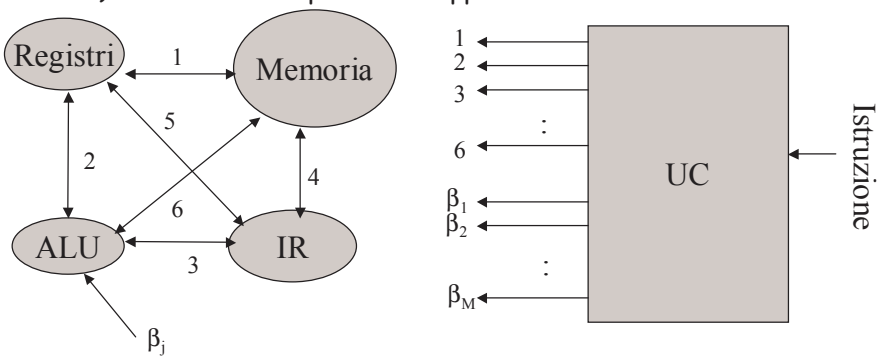
Elementi principali della CPU

- Banco di registri *(Register File)* ad accesso rapido, in cui memorizzare i dati di utilizzo più frequente. Il tempo di accesso ai registri è circa 10 volte più veloce del tempo di accesso alla memoria principale;
- Registro *Program counter (PC)*. Contiene l'indirizzo dell'istruzione corrente da aggiornare durante l'evoluzione del programma, in modo da prelevare dalla memoria la corretta sequenza di istruzione;
- Registro *Instruction Register (IR)*. Contiene l'istruzione in corso di esecuzione.

- Unità per l'esecuzione delle operazioni aritmetico-logiche (*Arithmetic Logic Unit - ALU*). I dati forniti all'*ALU* possono provenire da registri oppure direttamente dalla memoria, a seconda delle modalità di indirizzamento previste;
- Unità aggiuntive per elaborazioni particolari come unità aritmetiche per dati in virgola mobile *(Floating Point Unit – FPU)*, sommatori ausiliari, ecc.;

L'unità di controllo

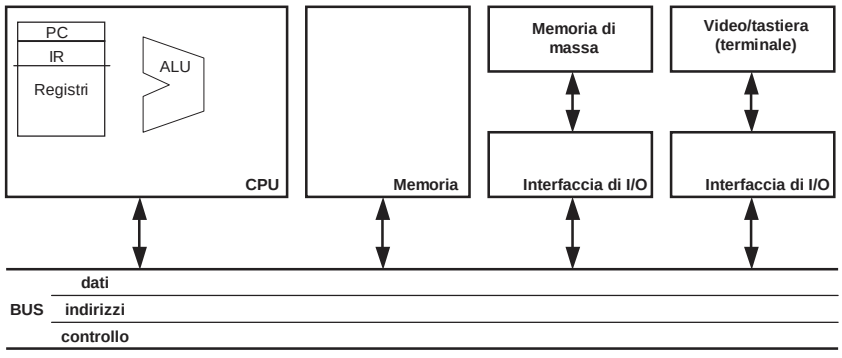
- Unità di controllo coordina i flussi di informazione:
- 1) abilitando le vie di comunicazione opportune a seconda dell'istruzione in corso di esecuzione.
- 2) selezionando l'operazione opportuna delle ALU.



Collegamenti bidirezionali tra i dispositivi: n (n-1) → non praticabile.

Collegamento tra i vari componenti (Architettura di Von Neumann)

Connessione *a nodo comune (bus)*.



Numero di collegamenti: 2n.

Tutte le unità del calcolatore sono connesse al bus.

Bus di sistema

- Permette la comunicazione tra le diverse unità del calcolatore ed è generalmente composto da tre parti logiche:
-
  - *Bus dati*, comprende le linee per trasferire dati e istruzioni da/verso I dispositivi (la memoria). In generale, la dimensione del bus dati è tale da garantire il trasferimento contemporaneo di una o più parole di memoria;
  - *Bus indirizzi*, su cui la *CPU* provvede a trasmettere l'indirizzo da cui prelevare il dato nel caso di lettura dalla memoria, oppure in cui depositarlo nel caso di scrittura nella memoria (esempio la cella di memoria).
  - *Bus di controllo*, dove transitano le informazioni ausiliarie per la corretta definizione delle operazioni da compiere (per esempio l'indicazione che si vuole effettuare una *lettura* piuttosto che una *scrittura*) e per la sincronizzazione tra *CPU* e memoria.

Esempio di utilizzo del bus

- Esempio: operazione di lettura dalla memoria. La *CPU* fornisce l'indirizzo della parola desiderata sul bus indirizzi, quindi viene richiesta l'operazione di lettura attivando il bus di controllo. Quando la memoria ha completato la lettura della parola richiesta, il dato viene trasferito sul bus dati e la *CPU* può prelevarlo ed utilizzarlo nelle sue elaborazioni.
- La struttura del bus può essere realizzata secondo diverse topologie di interconnessione.
- Il bus può esser utilizzato per un solo trasferimento alla volta ⇒ in ogni istante soltanto due unità *(Master e Slave)* possono usare il bus.
- Nelle architetture di riferimento l'unità Master è solitamente la CPU. Esistono alcune schede di I/O particolarmente performanti che possono diventare anch'esse bus-master.
- Le linee di controllo del bus vengono utilizzate per inviare più richieste contemporanee di utilizzo del bus che vengono gestite dalla logica di *arbitraggio* del bus.


Bus di sistema & buffer

- Principali vantaggi della struttura a bus singolo: elevata flessibilità e bassi costi.
- I dispositivi collegati al bus variano in termini di velocità dell'esecuzione delle operazioni ⇒ necessario un meccanismo di sincronizzazione per garantire il trasferimento efficiente delle informazioni sul bus.
- Tipicamente all'interno delle unità che utilizzano il bus sono presenti dei registri di buffer per mantenere l'informazione durante i trasferimenti e non vincolarsi alla velocità del dispositivo più lento connesso al bus.


Riassunto

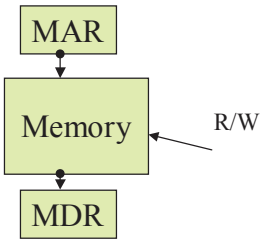
Architettura di riferimento:

CPU contiene: UC, Registri (IR, PC…), ALU.

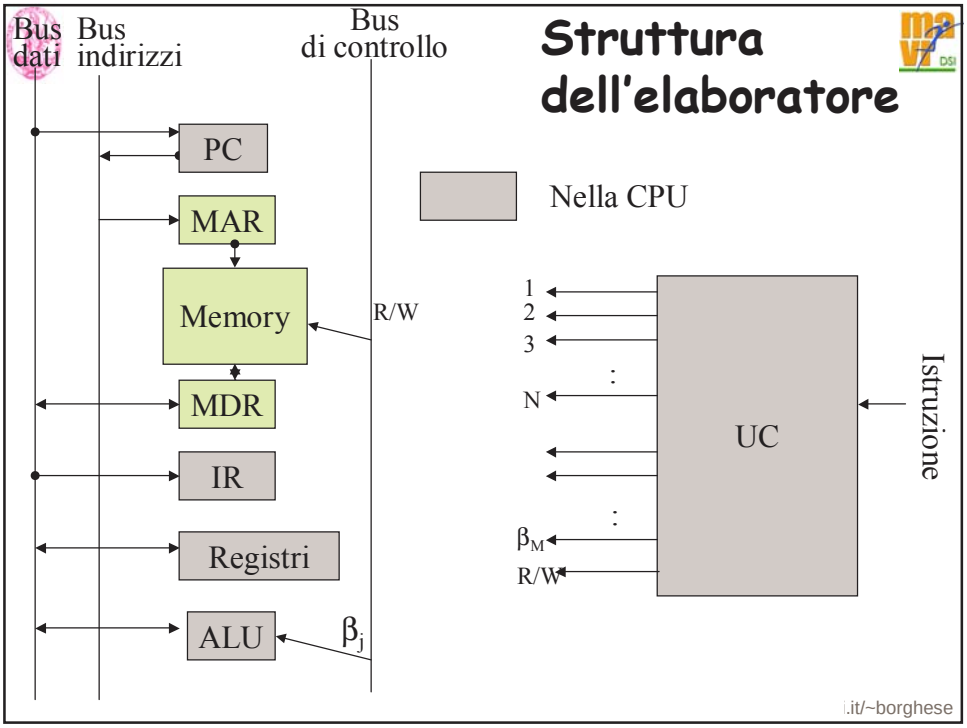
Collezione di oggetti collegati da BUS.

Interfaccia processore-memoria

- **MAR – Memory Address Register:** registro degli indirizzi della memoria per memorizzare l'indirizzo della posizione della memoria principale in cui o da cui i dati o istruzioni devono essere trasferiti.
- **MDR – Memory Data Register:** registro dei dati della memoria per memorizzare i dati che devono essere scritti in memoria o letti dalla memoria e le istruzioni lette dalla memoria nella locazione indicata dall'indirizzo specificato.



Struttura dell'elaboratore



Terminologia

Bit = binary digit.

- 1 byte = 8 bit.
- 1kbyte = 2^{10}byte = 1,024
- 1Mbyte = 2^{20}byte = 1,048,576.
- 1Gbyte = 2^{30}byte = 1,073,741,824.
- 1Tbyte = 2^{40}byte = 1,099,511,627,776.

- Parola (word) numero di bit trattati come un unicum dall'elaboratore.
- Le parole oggi arrivano facilmente a 64bit (Bus PCI-64).

Indirizzi nella memoria principale

- La memoria è organizzata in *parole* o *word* composte da *n-bit* che possono essere caricate e memorizzate con una singola operazione di lettura/scrittura della memoria. (*n* è chiamata *lunghezza di parola,* tipicamente da *16* a *64* bit).
- Ogni parola di memoria è associata ad un indirizzo composto da *k-bit.*
- I 2^k indirizzi (corrispondenti a 2^k parole) costituiscono lo *spazio di indirizzamento* del calcolatore. Ad esempio un indirizzo composto da *32-bit* genera uno spazio di indirizzamento di 2^{32} o *4G* parole = *16Gbyte*.
- Compito principale consiste nel contenere i moduli attivi del sistema operativo ed i processi in esecuzione (completi di istruzioni e dati).

Memoria Principale

- In genere, la dimensione della parola di memoria coincide con la dimensione dei registri contenuti nella *CPU*, in modo da poter caricare una parola di memoria in un registro della *CPU*. Se anche il bus dati è largo come la parola di memoria $\Rightarrow$ l'operazione di *load/store* avviene in un singolo ciclo.
- Le memorie in cui ogni locazione può essere raggiunta in un breve e prefissato intervallo di tempo misurato a partire dall'istante in cui si specifica l'indirizzo desiderato, vengono chiamate memorie ad accesso casuale *(Random Access Memory – RAM)*
- Nelle RAM il *tempo di accesso alla memoria* (tempo necessario per accedere ad una parola di memoria) è *fisso* e *indipendente* dalla posizione della parola alla quale si vuole accedere.

Indirizzi nella memoria principale

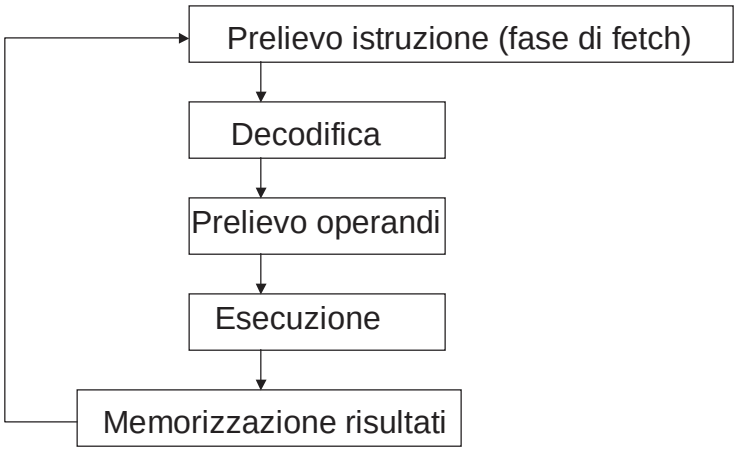
Indirizzo	$\Leftarrow$ n-bit $\Rightarrow$	Parola
0		Parola 0
1		Parola 1
…		…
i	b_{n-1} ……b_1 b_0	Parola i
…		…
2^k-1		Parola 2^k-1

Esecuzione delle istruzioni

Codifica dati e istruzioni

- Dati e istruzioni vengono manipolate dal calcolatore dopo essere state opportunamente *codificate*.
- L'istruzione è suddivisa in *campi* (*field*):
  - il campo *codice operativo* indica il tipo di operazione;
  - gli altri campi indicano gli indirizzi degli operandi oppure gli operandi stessi. Gli indirizzi possono riferirsi ad indirizzi di memoria o ai registri della CPU.
- Le *modalità di indirizzamento* indicano le diverse modalità attraverso le quali far riferimento agli operandi nelle istruzioni.

Ciclo di esecuzione di un'istruzione



Lettura dell'istruzione (fetch)

- Istruzioni e dati risiedono nella memoria principale, dove sono stati caricati attraverso un'unità di ingresso.
- L'esecuzione di un programma inizia quando il registro PC punta alla prima istruzione del programma.
- Il contenuto del PC viene trasferito nel MAR e un segnale di controllo per la lettura (READ) viene inviato alla memoria.
- Trascorso il tempo necessario all'accesso in memoria, la parola indirizzata (in questo caso la prima istruzione del programma) viene letta dalla memoria e trasferita nel registro MDR.
- Il contenuto del registro MDR (istruzione) viene trasferito nel registro IR.

Decodifica ed esecuzione dell'istruzione

- L'istruzione contenuta nel registro IR viene decodificata ed eseguita.
- Se l'istruzione deve essere svolta dall'unità aritmetico-logica è necessario recuperare gli operandi richiesti, che possono risiedere nei registri di uso generale oppure in memoria.
- Se un operando risiede in memoria, deve essere prelevato caricando l'indirizzo dell'operando nel registro MAR e attivando un ciclo di READ della memoria.
- L'operando letto dalla memoria viene posto nel registro MDR per essere trasferito alla ALU, che esegue l'operazione.

Scrittura in memoria

- Il risultato dell'operazione può essere memorizzato nei registri ad uso generale oppure in memoria.
- Se il risultato dell'operazione deve essere posto in memoria, esso viene caricato nel registro MDR. L'indirizzo della posizione di memoria in cui scrivere il risultato viene caricato nel registro MAR e si attiva un ciclo di scrittura (WRITE) della memoria.
- Mentre viene eseguita un'istruzione, il contenuto del PC viene incrementato in modo da puntare alla prossima istruzione da eseguire.
- Non appena è terminata l'esecuzione dell'istruzione corrente, si preleva l'istruzione successiva dalla memoria.

Classi di ISA

Accumulator (1 register):

1 address	add A	acc ← acc + mem[A]
1+x address	addx A	acc ← acc + mem[A + x]

Stack:

0 address	add	tos ← tos + next

General Purpose Register:

2 address	add A B	EA(A) ← EA(A) + EA(B)
3 address	add A B C	EA(A) ← EA(B) + EA(C)

Load/Store:

3 address	add Ra Rb Rc	Ra ← Rb + Rc
	load Ra Rb	Ra ← mem[Rb]
	store Ra Rb	mem[Rb] ← Ra

Comparazione per numero di istruzioni

Comparing Number of Instructions

Code sequence for C = A + B for four classes of instruction sets:

Stack	Accumulator	Register (register-memory)	Register (load-store)
Push A	Load A	Load R1,A	Load R1,A
Push B	Add B	Add R1,B	Load R2,B
Add	Store C	Store C, R1	Add R3,R1,R2
Pop C			Store C,R3

Architetture LOAD/STORE

- Il numero dei registri ad uso generale (ad esempio 32 registri da 32 bit ciascuno) non è sufficientemente grande da consentire di memorizzare tutte le variabili di un programma ⇒ ad ogni variabile viene assegnata una locazione di memoria nella quale trasferire il contenuto del registro quando questo deve essere utilizzato per contenere un'altra variabile.
- *Architetture LOAD/STORE:* gli operandi dell'ALU possono provenire soltanto dai registri ad uso generale contenuti nella CPU e **non** possono provenire dalla memoria. Sono necessarie apposite istruzioni di:
  - *caricamento (LOAD)* dei dati da memoria ai registri;
  - *memorizzazione (STORE)* dei dati dai registri alla memoria.

CPU di tipo RISC (*Reduced Instruction Set Computer*)

- Ispirate al principio di eseguire soltanto istruzioni semplici: le operazioni complesse vengono scomposte in una serie di istruzioni più semplici da eseguire in un ciclo base ridotto, con l'obiettivo di migliorare le prestazioni ottenibili dalle *CPU CISC*.
- Caratterizzate da istruzioni molto semplificate.
- Gli operandi dell'*ALU* possono provenire dai registri ma *non* dalla memoria. Per il trasferimento dei dati da memoria ai registri e viceversa si utilizzano delle apposite operazioni di caricamento (*load*) e di memorizzazione (*store)*
  ⇒ *architetture load/store*.

CPU di tipo RISC (*Reduced Instruction Set Computer*)

- *CPU* relativamente semplice ⇒ si riducono i tempi di esecuzione delle singole istruzioni, che sono però meno potenti delle istruzioni *CISC*.
- Dimensione *fissa* delle istruzioni ⇒ più semplice la gestione della fase di prelievo *(fetch)* e della codifica delle istruzioni da eseguire.

CPU di tipo CISC (*Complex Instruction Set Computer*)

- Caratterizzate da elevata complessità delle istruzioni eseguibili ed elevato numero di istruzioni che costituiscono l'insieme delle istruzioni.
- Numerose modalità di indirizzamento per gli **operandi** dell'*ALU* che possono provenire da registri oppure da memoria, nel qual caso l'indirizzamento può essere diretto, indiretto, con registro base, ecc.

CPU di tipo CISC (*Complex Instruction Set Computer*)

- Dimensione *variabile* delle istruzioni a seconda della modalità di indirizzamento di ogni operando ⇒ complessità di gestione della fase di prelievo o *fetch* in quanto a priori non è nota la lunghezza dell'istruzione da caricare.
- Elevata complessità della *CPU* stessa (cioè dell'hardware relativo) in termini degli elementi che la compongono con la conseguenza di rallentare i tempi di esecuzione delle operazioni. Elevata profondità dell'albero delle porte logiche, utilizzato per la decodifica.

Utilizzo architettura Intel 80x86: le 10 istruzioni più frequenti

Rank	instruction	Integer Average Percent total executed
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
	Total	96%

Simple instructions dominate instruction frequency

Interruzioni

- Oltre a trasferire dati/istruzioni tra memoria e processore, il calcolatore acquisisce dati dai dispositivi di ingresso e invia dati ai dispositivi di uscita attraverso apposite istruzioni che gestiscono i trasferimenti di I/O.
- Il normale flusso di esecuzione di un programma può essere interrotto da un segnale di interruzione *(INTERRUPT)* per una richiesta di intervento che un dispositivo di I/O manda al processore.

Interruzioni

- Il processore fornisce il servizio richiesto mediante l'esecuzione di una procedura di servizio delle interruzioni *(Interrupt Service Routine)* che deve salvare in memoria lo stato del processore prima di servire l'interruzione: il contenuto del PC, dei registri ad uso generale e alcune informazioni di controllo vengono salvati in memoria.
- Quando la procedura di servizio dell'interruzione viene completata, lo stato del processore viene ripristinato in modo che il programma interrotto possa proseguire.

Riassunto

La memoria è definita in termini di capacità.
Il ciclo di esecuzione delle istruzioni: fetch, decodifica, esecuzione.
Architetture RISC e CISC.
Interruzione dell'esecuzione.