Architettura dei calcolatori

A cura di:
Luca Breveglieri[*] Giacomo Buonanno[#]
Roberto Negrini[*] Giuseppe Pozzi[*] Donatella Sciuto[*]

[*] DEI, PoliMI, Milano
[#] LIUC, Castellanza (VA)

breveglieri,negrini,pozzi,sciuto@elet.polimi.it
buonanno@liuc.it
- versione del 24 marzo 2003 -

Organizzazione strutturata dei calcolatori

- **Bibliografia**:

  Tanenbaum A. S., Goodman J. R,
  "**Architettura dei computer** - Un approccio strutturato",
  Prentice Hall International, 2000 - paragrafo 1.1

- **Sommario**:
  - Introduzione (**perché un'organizzazione strutturata?**).
  - **I livelli nei calcolatori moderni**.
  - Conclusioni (**hardware vs. software**, qual è il punto giusto di separazione tra l'uno e l'altro?).

Alcuni concetti introduttivi

- **Algoritmo**
  descrizione di come si risolve un problema
- **Programma**
  algoritmo scritto in modo che possa essere eseguito da un calcolatore (**linguaggio di programmazione**)
- **Linguaggio macchina**
  linguaggio **effettivamente** "compreso" da un calcolatore, caratterizzato da
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore

Linguaggio di programmazione vs linguaggio macchina

```
...
scanf("%d",&max);
for
(i=1;i<100;i++)
{ scanf("%d",&N);
  if (N > max)
    max = N;
}
...
```

```
...
00000000 00000001 00000000
00000001
00000001 00000010 00000000
00000001
00000000 00000011 00000000
00000001
00000000 00000100 00000000
01100100
00000010 00000011 00000100
00000000
00000011 00000000 00000000
00000110
00000001 00000101 00000000
00000001
...
```

Problema: come ridurre la distanza tra questi due livelli

Soluzione: organizzare il calcolatore come una serie di macchine astratte (structured computer organization)

Esecutori e linguaggi

- Un esecutore è definito in base a tre elementi:
  - l'insieme delle **operazioni** che è capace di compiere;
  - l'insieme delle **istruzioni** che capisce (**sintassi**);
  - quali **operazioni** associa ad ogni **istruzione** che riconosce (**semantica**).
- Il calcolatore "capisce" le istruzioni che fanno parte del linguaggio macchina (che indichiamo con **L0**).
- L'obiettivo è quello di definire un linguaggio **L1** più facile da utilizzare rispetto al linguaggio macchina **L0**:
  - utilizzare le **istruzioni** di **L0** come **operazioni** di **L1**;
  - definire l'insieme delle istruzioni che fanno parte di **L1**;
  - definire quali operazioni vengono associate a quali istruzioni.

Come passare da L1 a L0

- **Traduzione** (o **compilazione**)
  - Un apposito programma (**compilatore**) traduce il programma $\mathbf{P_{L1}}$, scritto in linguaggio **L1**, in un programma $\mathbf{P_{L0}}$, scritto in linguaggio **L0**;
  - il nuovo programma $\mathbf{P_{L0}}$ viene quindi eseguito.
- **Interpretazione**
  - Un apposito programma (**interprete**) esamina il programma $\mathbf{P_{L1}}$, scritto in linguaggio **L1**, e, istruzione per istruzione, lo traduce nel linguaggio **L0** e lo esegue.

Interpretazione

- Il controllo è sempre nelle mani dell'**interprete**
- Dati dell'interprete
  - Programma $\mathbf{P_{L1}}$ scritto nel linguaggio **L1**
  - Dati del programma $\mathbf{P_{L1}}$
- ☺ Versatile e comodo in fase di debugging
  - Se si identifica un errore si può cambiare il codice e proseguire senza dover ripartire da zero
- ☹ Prestazioni ridotte
  - Ogni istruzione deve essere tradotta ogni volta che viene eseguita (e.g. cicli)

Compilazione

- Il **compilatore** mantiene il controllo solo nella prima fase (traduzione da $\mathbf{P_{L1}}$ a $\mathbf{P_{L0}}$).
- Dati del compilatore:
  - programma $\mathbf{P_{L1}}$ scritto nel linguaggio **L1** (input);
  - programma $\mathbf{P_{L0}}$ scritto nel linguaggio **L0** (output).
- ☺ Buone prestazioni:
  - si esegue $\mathbf{P_{L0}}$ scritto in linguaggio **L0**;
  - durante la traduzione si può ottimizzare il codice.
- ☹ In caso di modifica del codice (e.g. debugging) è necessario ricompilare il programma modificato.

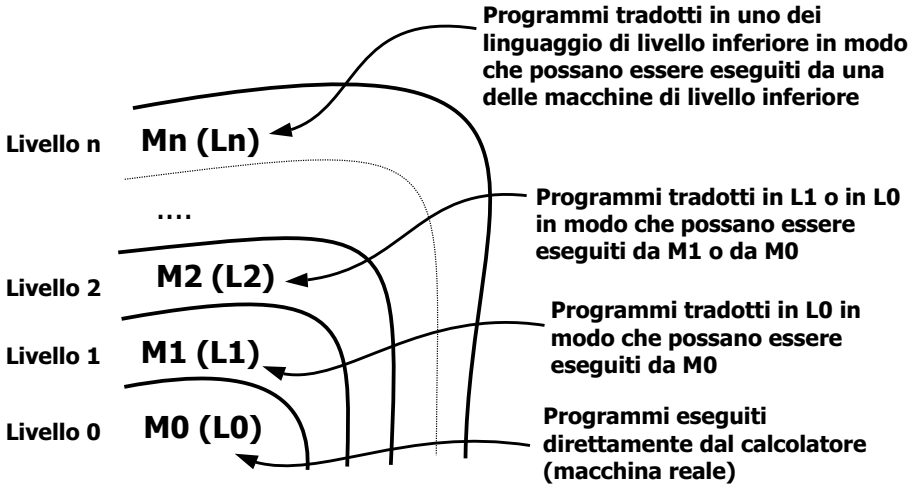
Macchina virtuale M1

- Il calcolatore **reale** (macchina **M0**) riconosce programmi scritti nel linguaggio **L0**
  - realizzare una **macchina reale** capace di comprendere il linguaggio **L1** sarebbe troppo costoso e/o poco efficiente
- La compilazione o l'interpretazione permettono di realizzare una **macchina virtuale** **M1** capace di comprendere il linguaggio **L1**.
- Problema:
  - per rendere comoda ed efficiente la traduzione, la distanza tra **L1** e **L0** (e quindi tra **M1** e **M0**) non può essere elevata;
  - non è detto che **M1** sia la soluzione desiderata (potrebbe essere ancora troppo distante dal livello "umano").

Macchina virtuale M2

- È possibile ripetere lo stesso procedimento ipotizzando che **M1** sia il punto di partenza (invece di **M0**) per definire una macchina virtuale **M2** capace di eseguire programmi scritti in un linguaggio **L2**.
- Per la traduzione dei programmi scritti nel linguaggio **L2** esistono due possibilità:
  - tradurli in **L1** per eseguirli con la **macchina virtuale** **M1**;
  - tradurli direttamente in **L0** in modo che possano essere eseguiti dalla **macchina reale** **M0**.
- Iterando si possono definire ulteriori macchine virtuali **M3**, **M4**, ... **MN** fino a raggiungere il livello di "**usabilità**" desiderato.

Macchina a più livelli



Relazione tra una macchina e il relativo linguaggio macchina

- **La macchina definisce il linguaggio.**
  Una macchina (reale o virtuale) permette di definire il **linguaggio macchina** ad essa associato come l'insieme di tutte le istruzioni che la macchina stessa è in grado di eseguire.
- **Il linguaggio definisce la macchina.**
  Un linguaggio permette di definire la macchina (reale o virtuale) ad esso associata come l'esecutore capace di comprendere tutti i programmi scritti in quel linguaggio.

Calcolatore a *n* livelli

- Per scrivere i programmi per il livello *n* non è necessario conoscere come viene effettuata la traduzione e quindi l'esecuzione.
- I programmi possono essere
  - eseguiti direttamente dalla macchina reale;
  - tradotti direttamente nel linguaggio L0;
  - interpretati da un interprete che viene a sua volta interpretato da un altro interprete ...;
  - ...
- La conoscenza dei livelli intermedi è importante per chi voglia capire
  - come **funziona** un calcolatore;
  - come si **progetta** una macchina virtuale.

I livelli nei moderni calcolatori

5. **Livello applicativo**
4. **Livello del linguaggio assemblatore**
3. **Livello del sistema operativo**
2. **Livello del microprogramma**
1. **Livello della microarchitettura**
0. **Livello logico**
-1. ... Livello dei dispositivi ...
-2. ... (fisica dello stato solido) ...

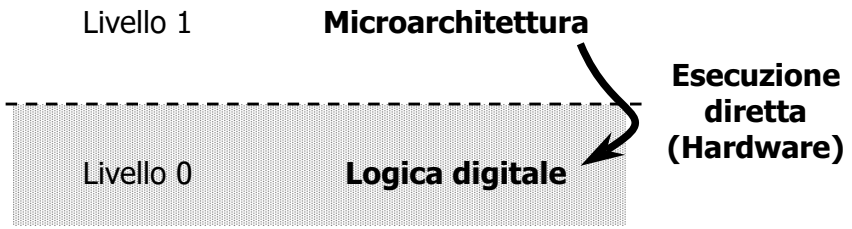
Livello dei dispositivi

- **Transistor** che formano i circuiti elettronici di cui è composto un calcolatore.
- Raggiunge un livello di dettaglio che viene in genere trascurato nella progettazione dei calcolatori.
- Gli stessi transistor vengono descritti a diversi livelli
  - switch, cioè interruttori che possono essere aperti o chiusi;
  - dispositivi caratterizzati da un comportamento approssimato da funzioni di II grado;
  - dispositivi caratterizzati da una serie di equazioni empiriche.
- A un livello ancor più basso ci si occupa della struttura dei transistor (**fisica dello stato solido**).

Livello 0: la logica digitale

- La macchina è formata da **porte logiche**.
- Ogni porta riceve in ingresso dei segnali binari (cioè segnali che possono essere 0 o 1) e calcola una semplice funzione (**AND**, **OR**, ...).
- Alcune porte, collegate opportunamente, possono formare una **memoria di un bit** (bistabile).
- Combinando **N** memorie di un bit si può formare un **registro** capace di memorizzare un numero binario (non più grande di $\mathbf{2^N-1}$).
- Combinando le porte si realizzano i circuiti che formano i calcolatori.

Livello 1: la microarchitettura /1



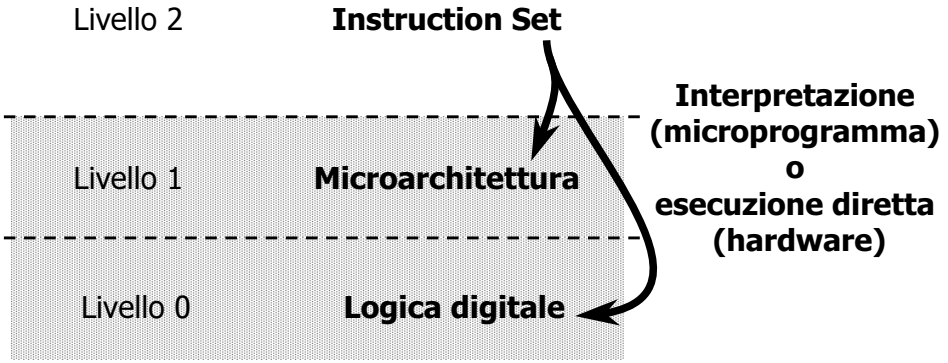
Livello 1: la microarchitettura /2

- Elaborazione – **Data Path**
  - registri **general purpose** come memoria locale;
  - Arithmetic Logic Unit (**ALU**) capace di eseguire semplici operazioni aritmetico-logiche;
  - Elementi di connessione tra registri e ALU.
- Controllo
  - registri dedicati al controllo (**PC**, **IR**, ...);
  - Unità di controllo – **Control Unit** che può **microprogrammata** o **cablata**.

Control Unit: cablata o microprogrammata?

- **Control Unit Microprogrammata**
  - il funzionamento del **Data Path** viene controllato da un programma chiamato microprogramma;
  - si tratta di un interprete delle istruzioni di livello 2;
  - era la soluzione più diffusa in passato, tanto che il termine μ-programmazione era a volte sinonimo di μ-architettura.
- **Control Unit Cablata**
  - il funzionamento del **Data Path** viene controllato direttamente tramite dispositivi hardware;
  - la sequenza di operazioni associate alle istruzioni di livello 2 non viene generata da un interprete ma viene gestita direttamente via hardware.

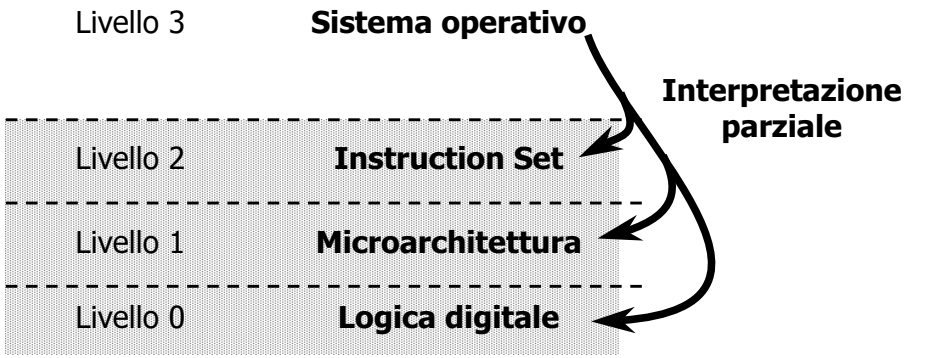
Livello 2: Instruction Set Architecture /1



Livello 2: Instruction Set Architecture /2

- Insieme delle istruzioni che possono essere **comprese** dalla **μ-architettura** (la **μ-architettura** agisce da interprete dell'**Instruction Set**).
- È il livello cui si fa riferimento quando si descrive il "**linguaggio macchina**" di un calcolatore.

Livello 3: il sistema operativo /1



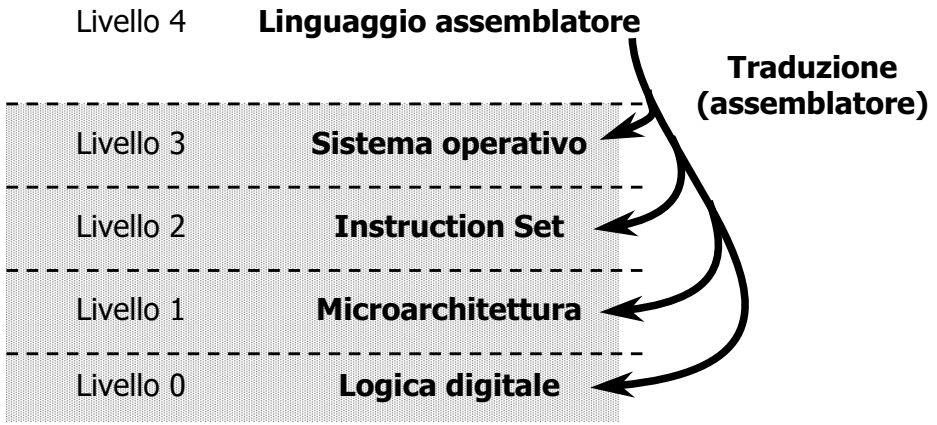
Livello 3: il sistema operativo /2

- Livello "**ibrido**":
  - comprende molte istruzioni che si trovano già al livello 2;
  - comprende anche un insieme di **istruzioni aggiuntive**;
  - ha una **diversa organizzazione della memoria**;
  - esegue **più programmi** contemporaneamente.
- Le nuove funzionalità sono eseguite da un **interprete** che viene definito "**sistema operativo**".
- Le istruzioni identiche a quelle del livello 2 vengono eseguite direttamente dalla **microarchitettura**.

Livelli bassi (1, 2, 3) vs livelli alti (4, 5)

- I livelli bassi supportano il funzionamento dei **compilatori** e degli **interpreti** utilizzati ai livelli alti:
  Livelli **bassi** ⇔ programmazione di **sistema**
  Livelli **alti** ⇔ programmazione di **applicazioni**
- Per migliorare le prestazioni:
  - i **livelli 2 e 3** vengono sempre **interpretati**;
  - i **livelli 4 e 5** vengono (quasi) sempre **compilati**.
- Per questioni di efficienza e di "usabilità":
  - i linguaggi dei **livelli 2 e 3** sono **numerici**;
  - i linguaggi dei **livelli 4 e 5** sono composti di **parole** e abbreviazioni che hanno un senso per l'uomo.

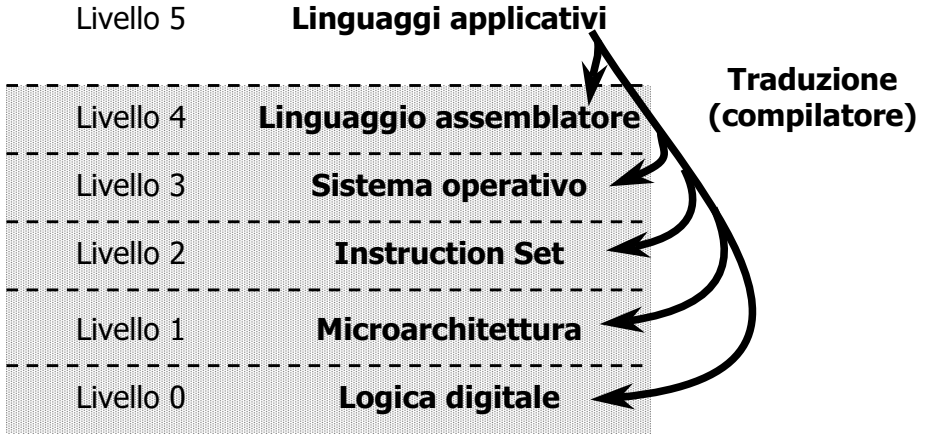
Livello 4: il linguaggio assemblatore /1



Livello 4: il linguaggio assemblatore /2

- Rappresentazione **simbolica** di uno dei livelli sottostanti.
  - I linguaggi numerici dei livelli "bassi" sono difficili da usare per un programmatore (... errori di "trascrizione" ...).
  - A **ogni** istruzione del linguaggio assemblatore corrisponde **una** istruzione del linguaggio macchina.
- I programmi in linguaggio assemblatore vengono **tradotti** in un linguaggio di livello inferiore e poi eseguiti. Il programma che esegue la traduzione si chiama **assemblatore**.

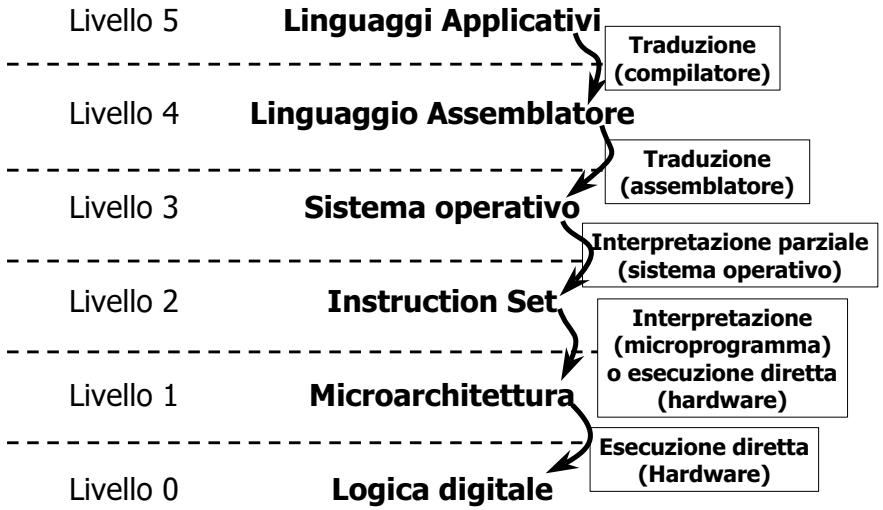
Livello 5: i linguaggi applicativi /1



Livello 5: i linguaggi applicativi /2

- Linguaggi di alto livello come **C**, **C++**, **Java**, **BASIC**, **LISP**, **Prolog**, …
- Sono utilizzati per la realizzazione di **programmi applicativi**.
- Il più delle volte la traduzione è affidata a un **compilatore**, mentre in alcuni casi si usa un **interprete** (e.g. Java).

I livelli: uno schema riassuntivo



I livelli: conclusioni

- I calcolatori sono progettati come una **serie di livelli** ognuno dei quali si **basa** sui **livelli precedenti**.
- Ogni **livello** rappresenta una **diversa astrazione** con strutture dati e funzionalità diverse.
- L'insieme di tipi di dati, operazioni e caratteristiche di ogni livello prende il nome di **ARCHITETTURA**.
- La descrizione dell'architettura di un livello presenta l'insieme delle **caratteristiche visibili all'utente** di quel livello (e.g. ad un programmatore di applicazioni interessa sapere quanta è la memoria disponibile)

Hardware vs. Software

- **Hardware**: oggetti **tangibili** che compongono un calcolatore (circuiti elettronici, memoria, dispositivi di I/O, ...).
- **Software**: **programmi**, indipendentemente dal supporto su cui sono memorizzati (HD, RAM, CD, ...).
- Hardware e software sono **logicamente equivalenti**
  - qualsiasi operazione effettuata dal software può essere inglobata nell'hardware ("hardware = software pietrificato");
  - qualsiasi istruzione eseguita dall'hardware può essere simulata dal software;
  - la decisione di realizzare una funzione in hardware o in software dipende da parametri quali il costo, la velocità, l'affidabilità, la frequenza di variazione, ...

Tappe fondamentali nell'evoluzione dell'architettura dei calcolatori

- **Bibliografia**:

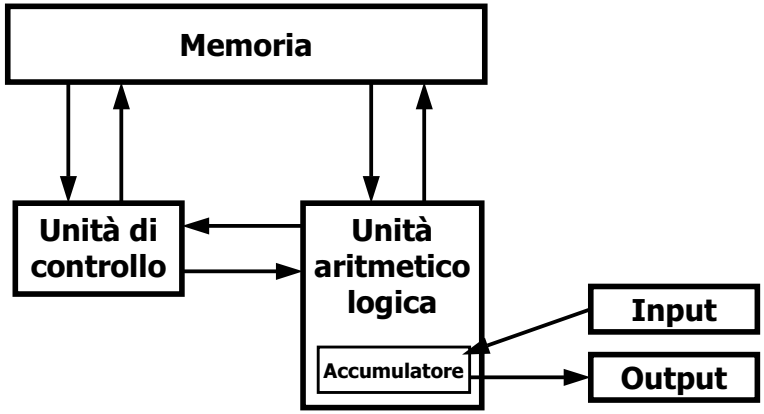
  Tanenbaum A. S., Goodman J. R,
  "**Architettura dei computer** - Un approccio strutturato",
  Prentice Hall International, 2000 - **paragrafi 1.2 e 1.3**

- **Sommario**:
  1. La macchina di von Nuemann.
  2. Il transistor.
  3. I circuiti integrati e la legge di Moore.
  4. I calcolatori disponibili.

John von Neumann

- Partecipa al progetto ENIAC.
- Due intuizioni fondamentali:
  - memorizzare i **programmi in forma digitale** nella stessa memoria dei dati per rendere più semplice la programmazione (rispetto all'utilizzo di cavi e interruttori);
  - utilizzare l'**aritmetica binaria** invece di quella decimale (due valvole per bit invece di dieci per cifra).
- Il suo progetto (**macchina di von Neumann**) è ancora oggi alla base di quasi tutti i calcolatori digitali.

Macchina di von Neumann /1



Macchina di von Neumann /2

- Progetto effettivo (**macchina IAS**)
  - Memoria composta da **4096 parole** di **40 bit** contenenti
    - **due istruzioni di 20 bit** ciascuna (8 bit per indicare il tipo d'istruzione e 12 bit per indirizzare una delle 4096 parole), oppure
    - un numero **intero con segno di 40 bit**;
  - Unità aritmetico logica dotata di un registro **accumulatore** di **40 bit**
  - Esempi di istruzioni:
    - aggiungi il contenuto di una cella di memoria all'accumulatore;
    - trasferisci il contenuto dell'accumulatore in una cella di memoria;
    - … …

Il transistor

- Inventato ai **Bell Labs** nel **1948** da John Bardeen, Walter Brattain e William Shockley:
  - nel giro di 10 anni rivoluziona la ricerca sui calcolatori;
  - alla fine degli anni '50 i calcolatori a valvole sono già obsoleti.
- **TX-0** (Transistorized eXperimental computer 0)
  - evoluzione del Whirlwind I (MIT);
  - progettato solo come prototipo del più sofisticato TX–2 che però non ebbe mai successo.
- **Digital Equipment Corporation** (**DEC**)
  - fondata nel 1957 da Kenneth Olsen, uno dei ricercatori del progetto TX-0, per vendere circuiti stampati;
  - nel 1961 realizza il PDP–1, il primo **minicalcolatore**.

I circuiti integrati (1958)

- Sviluppo della tecnologia d'integrazione:
  - **decine** (**SSI**), **centinaia** (**MSI**) e **migliaia** (**LSI**) di transistor sono integrati sullo stesso pezzo di silicio (**chip**);
  - possibilità di realizzare calcolatori **più piccoli**, **più veloci** e **meno costosi** dei loro predecessori.
- Due famiglie di calcolatori rappresentative:
  - **360** di IBM (modelli 30, 40, 50 e 65)
    - grazie alla **microprogrammazione** emulava sia il 1401 che 7094;
    - **multiprogrammazione** (più programmi in memoria);
    - spazio di **indirizzamento di 224 byte** (16MB), limite che rimane nelle famiglie 370, 4300, 3080 e 3090 fino agli anni '80.
  - **PDP-11** di DEC
    - evoluzione a 16 bit del PDP-8

Very Large Scale Integration

- 10^5–10^7 transistor integrati per chip.
- Passaggio dai **minicalcolatori**, alle **workstation**, ai **Personal Computer** (PC):
  - usati per applicazioni **fortemente interattive** (elaborazione testi, fogli elettronici, …);
  - in origine proposti come kit da assemblare, senza software;
  - due architetture principali:
    - **Apple** (basato su CPU Motorola e PowerPC)
      - primo PC, progettato da Steve Jobs e Steve Wozniak nel '78,
      - **architettura proprietaria**!
    - **IBM** e compatibili (CPU Intel e SW Microsoft – "Wintel")
      - realizzato utilizzando componenti "off the shelf",
      - **architettura di dominio pubblico**, quindi replicabile da altri (cloni)!

Ulteriori evoluzioni

- Sviluppo della tecnologia d'integrazione:
  - aumenta il numero dei transistor e quindi la complessità dei microprocessori;
  - sviluppo di architetture capaci di svolgere compiti sempre più complessi (CISC).
- Attenzione alla velocità e alla semplicità dell'hardware per consentire una maggior efficienza del software (RISC vs CISC).
- Esecuzione in parallelo di diverse istruzioni
  - CPU pipeline;
  - CPU superscalari.

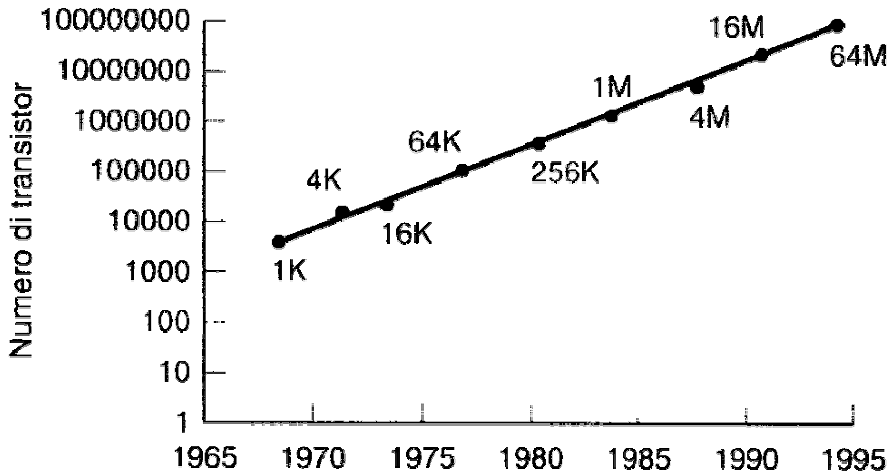
Legge di Moore

Osservazione fatta da Gordon Moore nel 1965:

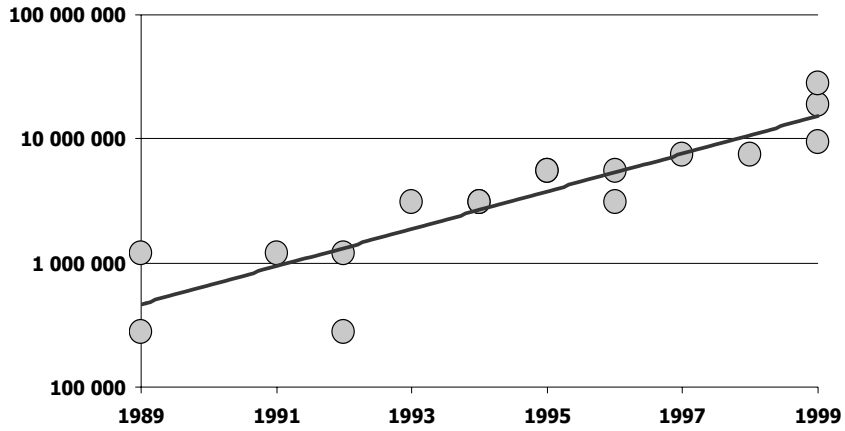
il numero dei transistor per cm^2 raddoppia ogni X mesi

In origine X era 12. Correzioni successive hanno portato a fissare **X=18**. Questo vuol dire che c'è un incremento di circa **il 60% all'anno**.

Densità dei chip di memoria



Transistor [CPU Intel]



Legge di Moore e progresso

- Il progresso della tecnologia provoca un **aumento del numero di transistor** per cm^2 e quindi per chip.
- Un maggior numero di transistor per chip permette di produrre **prodotti migliori** (sia in termini di prestazioni che di funzionalità) **a prezzi ridotti**.
- I prezzi bassi stimolano la nascita di **nuove applicazioni** (e.g. non si fanno video game per computer da milioni di $).
- Nuove applicazioni aprono **nuovi mercati** e fanno nascere **nuove aziende**.
- L'esistenza di tante aziende fa **crescere la competitività** che, a sua volta, stimola il **progresso della tecnologia** e lo sviluppo di **nuove tecnologie**.

Evoluzione del software

- Legge del software di Nathan (Nathan Myhrvold, top executive Microsoft): "Il software è come un gas che si espande per riempire il contenitore che lo contiene."
  - Anni '80: elaborazione di testi eseguita da programmi di poche decine di kilobyte di memoria (e.g. troff).
  - Oggi: word processor di decine di megabyte (e.g. word).
  - La continua aggiunta di nuove funzionalità al software esistente giustifica la costante richiesta di processori più veloci, memorie più grandi e più ampie capacità di I/O.
- Anche le altre tecnologie si sono evolute di pari passo:
  - I dischi rigidi sono passati da 10MB (1982) a 40GB (2003)
  - I modem analogici sono passati da 300 bit/sec a 56 Kbit/sec.

Quantità vs. qualità

- Cambiare di un ordine di grandezza la **quantità** significa cambiare anche la **qualità**:
  - un'auto in grado di raggiungere una velocità di 1000 km/h nel deserto del Nevada è una macchina **fondamentalmente diversa** da un'auto che fa 100 km/h sull'autostrada;
  - un grattacielo di 100 piani non è solo un edificio di 10 piani un po' più grande.
- Nei computer le differenze sono di **diversi ordini di grandezza**.
- I miglioramenti procurati dalla legge di Moore possono essere utilizzati in modi diversi:
  - costruire **calcolatori sempre più potenti a prezzo costante**;
  - costruire lo **stesso calcolatore a prezzi** ogni anno **più convenienti**.

Calcolatori disponibili /1

Tipo	Prezzo ($)	Applicazione tipica
Calcolatore monouso	1	Biglietti di auguri
Calcolatore dedicato	10	Orologi, automobili, …
Calcolatore per videogiochi	100	Videogiochi personali
Calcolatore per PC	1 K	PC da tavolo o portatile
Server	10 K	Server di rete
Reti di workstation	100 K	Centro di calc. dipartimentale
Mainframe	1 M	Database di una banca
Supercalcolatore	10 M	Previsioni del tempo

I prezzi sono solo indicativi.

Calcolatori disponibili /2

- **Calcolatori monouso**:
  - chip singoli incollati all'interno dei biglietti di auguri;
  - si tratta in pratica di calcolatori usa e getta.
- **Sistemi embedded** (calcolatori dedicati):
  - calcolatori che si trovano in telefoni, televisori, forni, auto, …
  - questi calcolatori contengono un processore, meno di un megabyte di memoria e qualche funzione di I/O.
- **Videogame**
  - normali calcolatori con particolari capacità grafiche, ma software limitato e poche possibilità di estensione; fanno parte di questa categoria anche i PDA;
  - contengono un processore, alcuni megabyte di memoria, un tipo di schermo (anche un televisore) e poco di più.

Calcolatori disponibili /3

- **Personal computer** (PC) o **workstation**:
  - dotati di alcune centinaia di megabyte di memoria, di un disco fisso contenente alcuni gigabyte di dati, drive CD–RW e DVD, modem, scheda audio e altre periferiche;
  - dotati di sistemi operativi elaborati, molte opzioni di espansione e una vasta gamma di software disponibile.
- **Server di rete**
  - si tratta di PC o workstation potenziati utilizzati come server di rete sia per le reti locali che per Internet;
  - esistono sia in configurazione con processore unico che con più processori, hanno alcuni gigabyte di memoria, molti gigabyte di spazio sul disco fisso e interfacce di rete ad alta velocità.

Calcolatori disponibili /4

- **NOW (Networks of Workstations)** o
  COW (Cluster of Workstations)
  - composti da PC o workstation normali collegate con reti ad elevate prestazioni (qualche gigabit/sec) e funzionanti con software speciale, che permette a tutte le macchine di lavorare insieme su un unico problema;
  - architetture sono facilmente scalabili (da alcune macchine ad alcune migliaia) e sono paragonabili a minisupercomputer.
- **Mainframe**
  - calcolatori grandi come una stanza, in uso fin dagli anni '60;
  - non sono più veloci di server potenti, ma solitamente hanno più capacità di I/O e sono dotate di grandi insiemi di dischi
  - sono macchine estremamente costose, che vengono spesso mantenute per via dell'ingente investimento esistente in termini di software, dati, procedure operative e personale.

Calcolatori disponibili /5

- **Supercomputer**
  - hanno CPU velocissime, molti gigabyte di memoria centrale, dischi e reti molto veloci.
  - Recentemente molti supercomputer sono diventati macchine altamente parallele non molto diverse dai COW, ma con componenti più veloci e più numerosi.
  - I supercomputer vengono utilizzati per risolvere problemi di calcolo molto complicati in campi scientifici e ingegneristici:
    - simulazione di uno scontro fra galassie,
    - sintesi di nuovi farmaci,
    - modelli del comportamento dell'aria attorno alle ali di un aereo.

Esempi di famiglie di calcolatori

- **Bibliografia**:

  Tanenbaum A. S., Goodman J. R,
  "**Architettura dei computer** - Un approccio strutturato",
  Prentice Hall International, 2000 - **paragrafo 1.4**
- **Sommario**:
  - Introduzione al **Pentium II**.
  - Introduzione all'**UltraSparc II**.
  - Introduzione a **picoJava II**.

Java

- **Obiettivo: permettere agli utenti WWW di caricare programmi binari via Internet e di eseguirli come se fossero pezzi di pagine Web.**
- Problemi:
  - **Sicurezza**
    Evitare che un programma possa interferire con l'ambiente in cui viene eseguito (virus, privacy, …);
  - **Portabilità**
    Consentire l'esecuzione su qualsiasi macchina
- Soluzione: **Java**, un nuovo linguaggio
  - orientato agli **oggetti** e ispirato a C++;
  - eseguito da una **Java Virtual Machine (JVM).**

Java Virtual Machine

- Memoria formata da **parole di 32 bit**;
- Capace di **eseguire 226 istruzioni**
  - la maggior parte sono semplici (RISC)
  - alcune sono complesse e richiedono più cicli di memoria.
- **Interprete JVM**
  - esecutore capace di eseguire un programma Java;
  - **scritto in C** (e quindi compilato ed eseguito su qualsiasi macchina dotata di compilatore C);
  - **lento**!!
- Alternativa: **compilatore JIT (Just In Time)**
  - scaricare il programma Java, compilarlo e poi eseguirlo;
  - **ritardo** nell'esecuzione causato dalla compilazione.

I microprocessori JVM

- Sono CPU che eseguono **direttamente** i programmi binari JVM.
- **Non servono strati intermedi** né di interpretazione software né di compilazione JIT.
- Le architetture **picoJava** I e II erano state pensate, in origine, per il mercato dei **sistemi embedded**.
- Flessibilità di implementazione:
  - si può **aggiungere o togliere l'unità floating–point**;
  - si possono **variare le dimensioni delle cache**;
  - …
- Vantaggi:
  - possibilità di **cambiare le funzionalità** mentre è operativo.

I microprocessori JVM

- L'architettura **picoJava II** è alla base di una serie di chip
  - Sun microJava 701 (riferimento nell'ambito del corso)
  - vari altri chip di produttori autorizzati dalla Sun.
- PicoJava II è dotato di due unità opzionali:
  - una **cache**
  - un'unità **floating point**.