NUMERO: 1468A - ANNO: 2015

A P P U N T I

STUDENTE: De Donno

MATERIA: Architettura dei Sistemi di Elaborazione.
Prof.Mezzalama

Architettura dei sistemi di elaborazione

(Teoria)

A cura di ***Michele De Donno***

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 3 di 484

Presentazione corso

Obiettivi

Analizzare, valutare, progettare i sistemi di elaborazione dai mobile internet device (tablet, smartphone,..), ai personal computer, ai server, ai data center*. E pertanto:

- Comprendere l'architettura dei sistemi basati su (micro)processori, con particolare riferimento alla piattaforma PC/server e sistemi embedded (tablet, mobile) in tutte le sue componenti (processore, bus, memoria, interfacce periferiche).
- Comprendere l'interazione hw-sw di base (kernel s.o).
- Acquisire la conoscenza delle tecniche di progetto per la realizzazione delle interfacce hw e dei driver sw dei principali dispositivi periferici.
- Acquisire le conoscenze per la valutazione ed il dimensionamento dei data center dal punto di vista informatico, energetico, logistico.

*Il Centro Elaborazione Dati (CED) (in inglese **Data Center**) è l'unità organizzativa che coordina e mantiene le apparecchiature ed i servizi di gestione dei dati ovvero l'infrastruttura IT (Information Technology) a servizio di una o più aziende. In alcune realtà può essere denominato Servizio Elaborazione Dati (SED).

Al suo interno possono essere presenti vere e proprie server farm. In alcune realtà sono strutture organizzate su scala industriale ed operano utilizzando una quantità di energia elettrica paragonabile a quella di una città di piccole dimensioni a volte fonte significativa di inquinamento atmosferico sotto forma di Emissioni Diesel.

Il Data Center del Politecnico di Torino consuma 400 KWatt, l'equivalente di circa 133 abitazioni (da 3 KWatt l'una), ed è alimentato da una tensione di 380 Volt.

Temi

Argomenti principali relativi a CPU, Bus e Memoria:

- Processori evoluti (superscalari, hw per virtualizzazione, SoC);
- Processori dedicati(SoC, DSP);
- Assembler x86, ARMx;
- Sistemi di interconnessione (bus, point-to-point);
- Architetture di memoria (DDR3/4, memorie ECC)
- Architetture multiprocessore (coerenza cache, sincronizzazione processi e thread);
- Interfacce intelligenti per periferici (DMA o no-DMA driven);

*A cura di **Michele De Donno***

© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 5 di 484

Concetti introduttivi

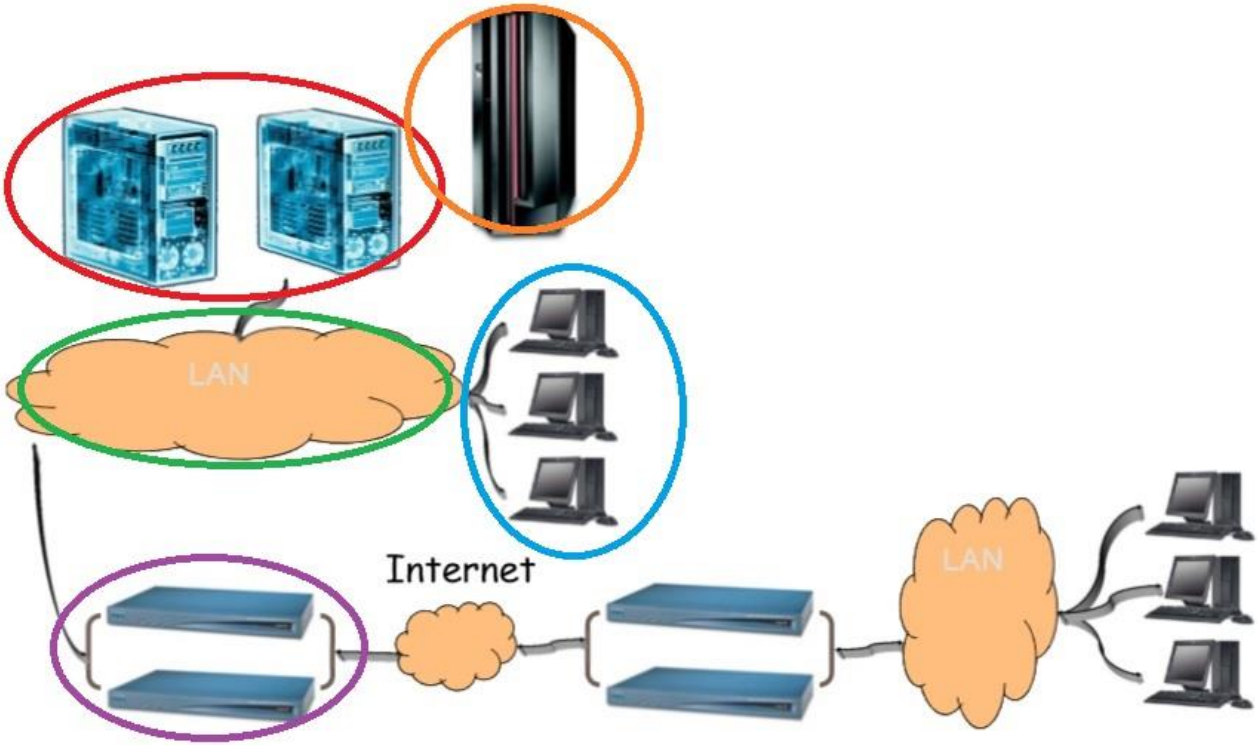
Classificare gli elaboratori

In termini di aree di mercato gli elaboratori possono essere classificati in diversi modi:

- Desktop/portatili;
- Server;
- Mainframe;
- Supercomputer;
- Embedded systems (dai router ai trenini)

Un CED, o data center, è costituito da una serie di componenti:

- Servers;
- Mainframe - è una tipologia di computer caratterizzata da prestazioni di alto livello;
- Client;
- Unità di memorizzazione;
- Rete che interconnette gli elementi;
- Dispositivi di rete;



Un **sistema di elaborazione** è una somma di componenti ognuno con le proprie caratteristiche e prestazioni. L'architetto dei sistemi di elaborazione deve integrarli e configurarli tra loro in maniera equilibrata.

Si tenga in considerazione che ognuno di questi supercomputer necessita di almeno una centrale elettrica che lo alimenti (in genere anche più di una per motivi di ridondanza). Anche per Data Center di grosse dimensioni sono necessarie delle centrali elettriche che li alimentino.

Osservazioni

Alcune interessanti osservazioni possono essere fatte rispetto ai dati analizzati:

- I supercomputer sono costruiti con processori comunemente trovati sui normali dispositivi in commercio (Es. Intel, AMD, IBM);
- Molti dei supercomputer esistenti presentano al loro interno dei processori grafici integrati. Questo perché tali processori risultano essere intrinsecamente vettoriali e quindi intrinsecamente paralleli, cioè adatti a lavorare con una grande mole di dati (ovvero processori SIMD – Single Istruction Multiple Data).

Supercomputer di questo tipo sono tipicamente adoperati nei settori della fisica, chimica, meteorologia, sismologia, etc., come anche nelle industrie (automobilistiche e petrolifere) e in ambito militare.

High Performance Computing in Italia

L'Italia ha fatto il suo debutto nella lista dei primi 10 supercomputer nel 2011 con l'IBM BlueGene/Q installato presso CINECA. Ad oggi (2014) tale dispositivo occupa la posizione numero 17 nella lista dei più potenti supercomputer a livello mondiale, vantando delle prestazioni nell'ordine di 1.72 PFlops/s.

Al numero 11 della classifica mondiale si trova invece il supercomputer ENI HPC2-IBM, con 3 PFlops/s.

In totale, 6 dei 10 migliori supercomputer al mondo sono negli Stati Uniti, uno (il più potente) in Cina, uno in Giappone, uno in Svizzera e uno in Germania.

Cervello e computer: Chi pensa meglio?

Come è possibile correlare il cervello umano con la potenza di calcolo?

I chip odierni sono in grado di eseguire circa 300 miliardi di calcoli al secondo; il cervello umano contiene circa 100 miliardi di neuroni e quello di un cane circa 10 milioni di neuroni.

IBM, a partire dal 2011, ha realizzato chip che riproducono 256 ($=2^8$) neuroni e al massimo 262 144 ($=2^{18}$) sinapsi (ogni neurone umano ha al massimo 100 000 sinapsi).

Un cervello umano completo contiene fino a 100 miliardi di neuroni, ognuno con max 100'000 sinapsi. Si tratterebbe quindi di collegare tra loro circa 400 milioni di chip da 256 neuroni ciascuno, una cifra esorbitante, che quindi rende (fortunatamente?) ancora non troppo vicina la realizzazione di un cervello umano artificiale.

Calcolo delle prestazioni

Evoluzione dell'architettura dei computer

Nelle ultime decadi la progettazione di computer ha tratto vantaggio da due aspetti tecnici fondamentali:

- **Innovazioni architetturali** – cioè come sono articolate le realizzazioni delle funzioni all'interno di un sistema di elaborazione (Esempi di scelte architetturali sono la pipeline, le multilevel cache, etc.).
- **Miglioramenti tecnologici** – ovvero fondamentalmente la larghezza di canale disponibile (larghezza di canale ad oggi = 20 nanometri).

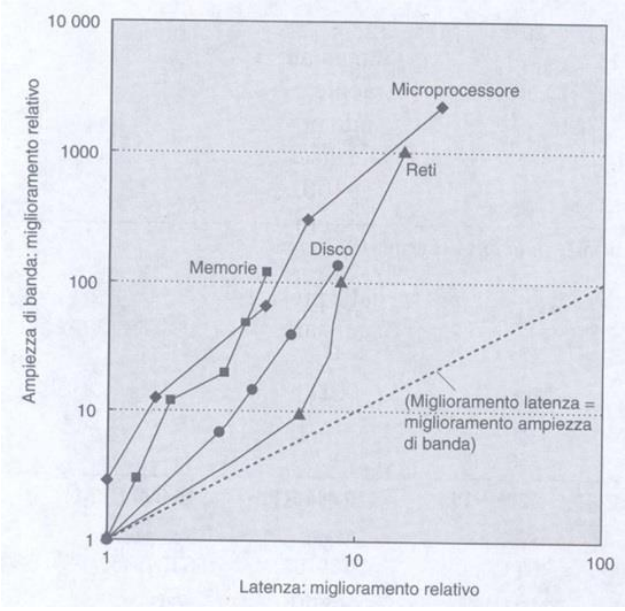
E' stato stimato che la differenza tra i microprocessori con le più alte prestazioni disponibili nel 2001 e quello che si sarebbe ottenuto facendo affidamento unicamente sulla tecnologia, è superiore ad un fattore di 15.

I parametri fondamentali per valutare lo *sviluppo* e le conseguenti *prestazioni* di un qualsiasi elemento che costituisce un sistema di elaborazione, sono:

1. **Throughput** (detto "ampiezza di banda" nell'ambito delle reti): definita come la quantità totale di compiti/azioni/operazioni svolte in un tempo preassegnato.
2. **Latency** (tempo di risposta): definito come il tempo che intercorre dalla richiesta alla effettiva messa in atto di un comando.

I miglioramenti si ottengono mediante approcci **tecnologici** e **architetturali**.

Negli ultimi tempi l'evoluzione tecnologica si è adoperata per migliorare principalmente il throughput, come è facile osservare dal grafico seguente:



La latenza è ancora un elemento da migliorare.

Hard disk vs SSD (Solid State Drive)

Un esempio del miglioramento di prestazioni ottenuto agendo sul latency piuttosto che sul throughput, è l'evoluzione dei dispositivi di storage.

Gli Hard Disk e gli SSD presentano una velocità di trasferimento (throughput) pressoché identica (300 MBps), tuttavia la latenza è completamente diversa (HDD: 12 ms, SSD: 0.1 ms). Ne risultano delle prestazioni notevolmente migliorate degli SSD rispetto agli HDD (HDD: 150 IOPS, SSD: 5000 IOPS).

Questo dimostra come la latenza sia spesso un elemento determinante per migliorare le prestazioni; in pratica questi due tipi di dispositivi presentano una velocità di trasferimento all'incirca uguale (e quindi un throughput pressoché identico), ciò che cambia è il tempo di accesso ai dati (ovvero la latency).

Misurare gli elaboratori

Molteplici sono i parametri da considerare quando si vogliono misurare le prestazioni di un elaboratore, in gran parte dipendenti dal dominio applicativo:

- Potenza di calcolo;
- Potenza grafica;
- Transazioni gestite nell'unità di tempo;
- Rapporto costo/prestazioni;
- Etc.

Quale è meglio?

Vediamo alcune unità di misura che possono essere usate e che misurano aspetti differenti di un calcolatore:

- **MIPS** - Million Instructions Per Second: misura la capacità di un processore di eseguire istruzioni.
  Tale unità di misura è in grado di misurare le prestazioni relative alla **CPU** del calcolatore e non tiene in considerazione tutto ciò che è al di fuori del processore, ovvero BUS, I/O e memoria.
- **FLOPS** – FLoating point Operations Per Second: misura il numero di operazioni in virgola mobile al secondo che il calcolatore è in grado di eseguire.
  Tale unità di misura è in grado di misurare le prestazioni relative alla **capacità computazionale** del calcolatore.
- **IOPS** - Input/Output Operations Per Second: misura il numero di operazioni I/O al secondo che il calcolatore è in grado di eseguire.
  Tale unità di misura è in grado di misurare le prestazioni relative ai **dischi** del calcolatore.

In un sistema composto da più parti, le prestazioni globali del sistema dipendono dalle prestazioni delle singole componenti. E' necessario quindi capire quanto, il miglioramento del singolo componente, influenza e determina il miglioramento complessivo del sistema.

A tal proposito si può affermare che, un aumento di prestazioni del sistema che deriva da un miglioramento locale (interno al sistema), dipende da due fattori:

1. ***Fraction$_{enhanced}$***: la porzione del tempo computazionale che trae vantaggio dal miglioramento locale eseguito sul sistema.
   In pratica indica la percentuale del sistema che è interessata dal miglioramento.
2. ***Speedup$_{enhanced}$***: l'entità del miglioramento sulla singola parte del sistema.
   In pratica indica di quanto si migliora l'elemento specifico.

Tenendo in considerazioni queste grandezze, si può definire il *TimeNew* come segue:

$$execution\ time_{new} = execution\ time_{old} \times \left[(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{speedup_{enhanced}}\right]$$

Dove:

- $execution\ time_{old} \times (1 - fraction_{enhanced})$ → identifica il **tempo** di esecuzione relativo alla parte di sistema **non affetta da miglioramento** (essendo $(1 - fraction_{enhanced})$ la porzione di sistema non interessata dal miglioramento);
- $execution\ time_{old} \times \frac{fraction_{enhanced}}{speedup_{enhanced}}$ → identifica il **tempo** di esecuzione relativo alla parte di sistema sulla quale è stato **eseguito il miglioramento**.

La formulazione definitiva della ***legge di Amdahl*** che indica il miglioramento complessivo ottenuto su un sistema di elaborazione, può essere quindi così definita:

$$\boldsymbol{speedup_{overall} = \frac{execution\ time_{old}}{execution\ time_{new}} =}$$

$$\boldsymbol{= \frac{1}{(1 - fraction_{enhanced}) + \frac{fraction_{enhanced}}{speedup_{enhanced}}}}$$

Scegliere tra due soluzioni: Esempio

Si supponga che siano disponibili due possibili soluzioni per incrementare le prestazioni floating-point di una macchina:

- *Soluzione 1*: Incrementare di un fattore 10 le prestazioni dell'operazione radice quadrata (responsabile del 20% del tempo di esecuzione della macchina) aggiungendo un hardware specifico.
- *Soluzione 2*: Incrementare di un fattore 2 le prestazioni di tutte le operazioni floating-point (responsabili del 50% del tempo di esecuzione della macchina).

Quale soluzione rende la macchina più veloce?

Svolgimento Soluzione 1

$$speedup_1 = \frac{1}{(1-0.2)+\frac{0.2}{10}} = \mathbf{1.22}$$

Svolgimento Soluzione 2

$$speedup_{overall} = \frac{1}{(1-0.5)+\frac{0.5}{2}} = \mathbf{1.33}$$

In questo caso, migliorare di un fattore 2 l'esecuzione di tutte le operazioni in virgola mobile, fornisce uno speedup complessivo superiore rispetto all'altra soluzione.

Risposta: ***Soluzione 2***.

Alcuni "miti" da sfatare

"Il processore conta più di tutto" → **ERRORE**

Esempio: in un sistema orientato ad applicazioni web il 40% del tempo è impiegato nell'esecuzione di istruzioni (CPU) e il 60% in operazioni di I/O (rete, disco).

Sostituendo il nuovo processore con uno 10 volte più veloce, il guadagno di prestazioni (speedup) risulta:

$$speedup = \frac{tempo\ senza\ miglioramento}{tempo\ con\ miglioramento} = \frac{tempo \times (0.6+0.4)}{tempo \times \left(0.6+\frac{0.4}{10}\right)} =$$

$$= \frac{1}{(1-0.4)+\frac{0.4}{10}} = \frac{1}{0.64} = \mathbf{1.56}$$

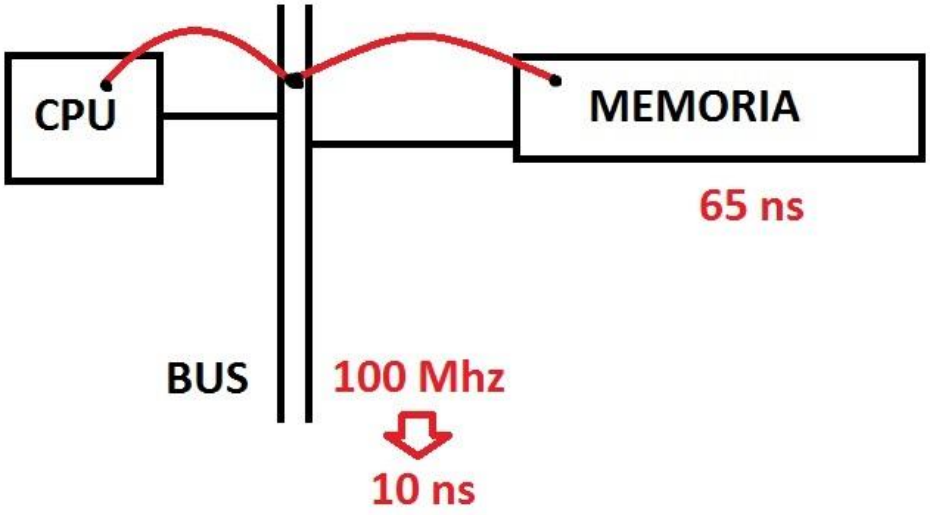
Appare quindi evidente che utilizzato un processore 10 volte più potente rispetto al precedente, il sistema ottiene un miglioramento complessivo inferiore al doppio delle prestazioni.

- Tempo *istruzioni di memoria*:

  Per calcolare il tempo di accesso a memoria bisogna tenere in considerazione sia il tempo di accesso al BUS (e quindi il suo clock) che il tempo di accesso alla memoria stessa.

  Per accedere a memoria è infatti necessario utilizzare un numero intero di cicli di bus, essendo il bus che temporizza l'accesso ad essa. Il numero di cicli di bus da occupare dipende dal tempo necessario per l'effettivo di accesso a memoria*: il tempo di occupazione del bus (ovvero di esecuzione dei cicli di bus) deve essere maggiore o al più uguale al tempo necessario per l'effettivo accesso a memoria*.

  Vediamo un esempio aggiuntivo per chiarire il concetto, nel quale supponiamo un tempo di accesso a memoria di 65 ns e un clock di Bus a 100 Mhz:



  In tale situazione appare evidente che per eseguire un accesso a memoria, che necessità di 65 ns, sarà necessario eseguire almeno 7 cicli di bus (a 100 Mhz), per un totale di 70 ns.

  Dall'esempio mostrato, appare evidente che nel nostro caso (accesso a memoria: 60 ns, clock Bus: 100 Mhz), saranno sufficienti 6 cicli di Bus per il singolo accesso a memoria; per cui:

$$n°\ cicli\ di\ bus\ minimi \times tempo\ di\ ogni\ ciclo\ BUS = 6 \times 10\ ns = \mathbf{60\ ns}$$

- Tempo *istruzioni I/O*:

$$n°\ cicli\ di\ bus\ per\ I/O \times tempo\ di\ ogni\ ciclo\ BUS = 12 \times \frac{1}{100 \times 10^6\ Hz} =$$

$$= 12 \times 0.01 \times 10^{-6} s = 12 \times 10^{-8} s = 120 \times 10^{-9} s = \mathbf{120\ ns}$$

Osservazione

È evidente che, se si migliora una parte del sistema di un fattore *X*, lo speedup totale del sistema dovrà essere necessariamente strettamente minore di *X* ($speedup < X$).

Esempio 4

Si consideri il seguente sistema caratterizzato da:

- Clock di CPU: 1Ghz;
- Clock di Bus: 100Mhz;
- Tempo di accesso alla memoria: 60 ns;
- Tempo di accesso alle interfacce I/O: 12 cicli di bus.

Si calcoli il guadagno prestazionale ottenuto passando ad una CPU con un clock di 2 Ghz (migliora la CPU di un fattore 2).

Svolgimento

In modo analogo a quanto fatto nell'esempio precedente, supponiamo di applicare il miglioramento richiesto:

- Istruzioni nella CPU:

$$30\%\ istruzioni\ totali \times tempo\ istruzioni\ CPU = 30 \times 1\ ns = 30\ ns$$

- Istruzioni in memoria:

$$60\%\ istruzioni\ totali\ \times tempo\ istruzioni\ memoria = 60 \times 60\ ns = 3600\ ns$$

- Istruzioni di I/O:

$$10\%\ istruzioni\ totali \times tempo\ istruzioni\ I/O = 10 \times 120\ ns = 1200\ ns$$

L'esecuzione di 100 istruzioni richiede quindi un totale di **4830 ns**:

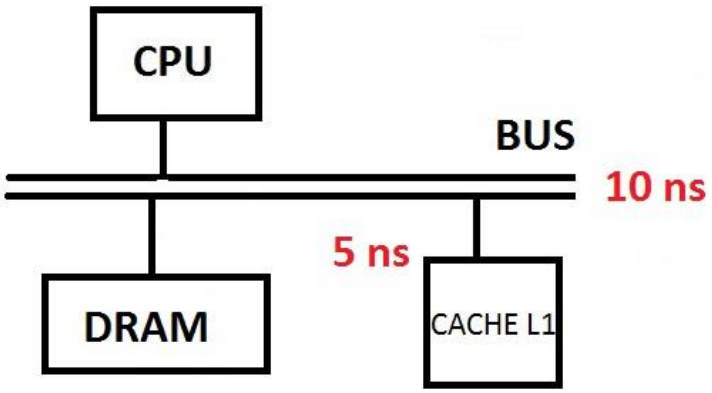
$$30\ ns + 3600\ ns + 1200\ ns = 4830\ ns$$

Il guadagno (**speedup**) ottenuto sul sistema è quindi dato dalla legge di Amdahl:

$$speedup_{overall} = \frac{execution\ time_{old}}{execution\ time_{new}} = \frac{4860}{4830} = \mathbf{1.006 \sim 1}$$

Risposta: ***1***

Ad esempio, nella situazione seguente, il tempo di accesso alla cache sarebbe di 10 ns (tempo di un ciclo di bus):



Curiosità

Il parametro fondamentale di una cache è l'hit rate, in base al quale si decide anche la dimensione della cache stessa.

L'hit rate che è mediamente possibile trovare al giorno d'oggi è del ***98-99%.***

Dall'ipotesi fatta sulla cache ne deriva che il tempo su 100 istruzioni sarà dato dalla somma dei seguenti tempi:

- Istruzioni nella CPU:

$$30\%\ istruzioni\ totali \times tempo\ istruzioni\ CPU = 30 \times 2\ ns = 60\ ns$$

- Istruzioni in memoria cache:

$$95\%\ del\ 60\%\ di\ istruzioni\ \times tempo\ istruzioni\ memoria\ cache =$$

$$= 60 \times 0.95 \times 5\ ns = 285\ ns$$

- Istruzioni in memoria:

$$5\%\ del\ 60\%\ di\ istruzioni\ \times tempo\ istruzioni\ memoria =$$

$$= 60 \times 0.05 \times 60\ ns = 180\ ns$$

- Istruzioni di I/O:

$$10\%\ istruzioni\ totali \times tempo\ istruzioni\ I/O = 10 \times 120\ ns = 1200\ ns$$

L'esecuzione di 100 istruzioni richiede quindi un totale di **1725 ns**:

$$60\ ns + 285\ ns + 180\ ns + 1200\ ns = 1725\ ns$$

Potenza dissipata

Uno dei problemi più rilevanti nella progettazione di un qualunque sistema è quello della potenza dissipata, sia a livello di elaboratore (globale: comprende CPU, memoria, etc.) sia a livello di microprocessore (locale).

Per questo motivo è nata la nuova filosofia del **green IT**.

Informatica Verde (dall'inglese "green computing" or green IT), si riferisce ad un'informatica ecologicamente sostenibile. Si occupa dello studio e della messa in pratica di tecniche di progettazione e realizzazione di computer, server, e sistemi connessi (come ad esempio monitor, stampanti, dispositivi di archiviazione e reti) e sistemi di comunicazione efficienti con impatti ambientali limitati o nulli. La green IT si pone un duplice obiettivo, il raggiungimento di un tornaconto economico e di buone prestazioni tecnologiche, rispettando le nostre responsabilità sociali ed etiche.

In pratica “la green IT è lo studio e l'utilizzo di tecnologie informatiche in modo efficiente".

Basti pensare che nel 1990 il 60% di un costo di un data center dipendeva dalle macchine («costo informatico»). Nel 2014 il 30% del costo dipende dall’informatica, il 70% dall’esercizio (ovvero essenzialmente impianti elettrici e di refrigerazione).

Ne deriva la necessità di progettare e gestire i data center in una logica di risparmio energetico; è quindi necessario avere componenti elettronici che consumano e riscaldano meno. In generale il concetto di green IT si indirizza su:

- **Progettazione Verde** — progettazione a basso consumo energetico e componenti dell'ambiente, computer, server, apparati per il raffreddamento, e data center.
- **Fabbricazione Verde** — realizzare componenti elettronici, computer e altri sottosistemi con un minimo impatto ambientale.

Consumo di energia

I computer consumano una grossa fetta dell’energia prodotta a livello mondiale. A tal proposito, analizziamo alcuni dati:

Un PC/server consuma circa 200 w ed opera per circa 2000 h/anno. Ne deriva un consumo totale di 400 KWh*.

Oggi si possono considerare nel mondo attivi circa 500 milioni di PC. Ne deriva che il consumo energetico totale dei soli pc è di **200 TWh**.

*Wh= Watt per ora.

Per quanto riguarda invece la produzione energetica, sappiamo che la produzione di energia elettrica in Italia è di circa 350 TWh ogni anno.

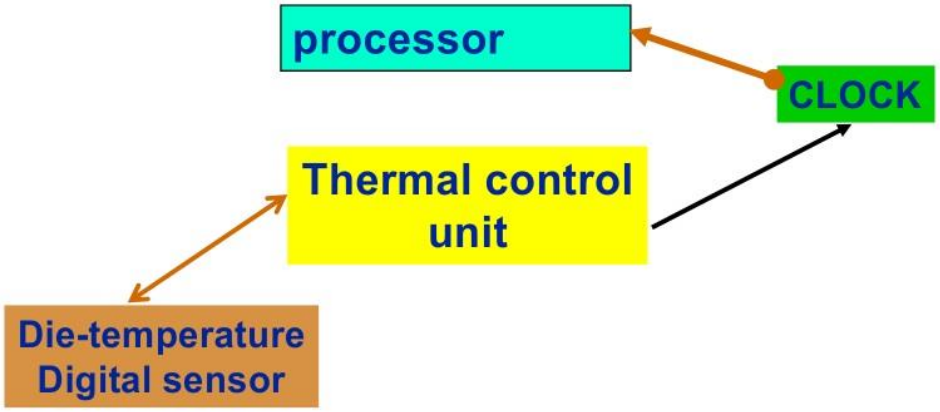
Intel Core

Ad oggi, ogni core dispone di:

- un sensore che misura la temperatura interna del die: **Digital sensor**
- una unità di controllo della temperatura della zona di silicio (die): **Thermal control unit**

Quando viene rilevata una temperatura superiore ad una data soglia (settabile per via software in un registro), l'unità di controllo riduce la frequenza di clock del processore (e quindi la potenza dissipata).

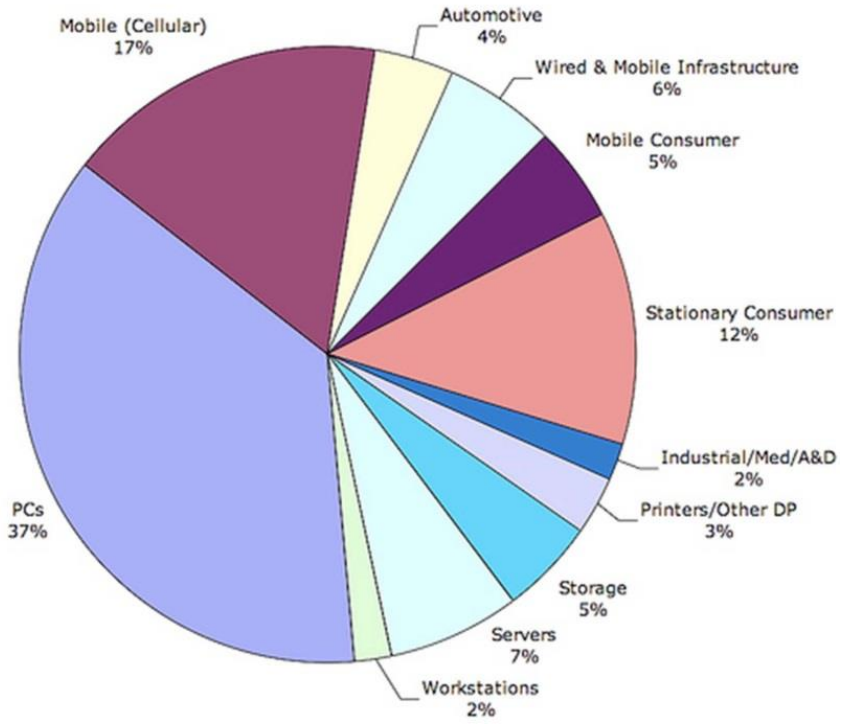
Generalmente ci si riferisce a questa pratica con il nome di: **controllo dinamico della temperatura**.



Appare evidente che l'ideale sarebbe avere una frequenza del processore alta con una tensione bassa al fine di garantire prestazioni elevate e bassi consumi. Tuttavia, essendo le due grandezze direttamente correlate tra loro, si cerca costantemente un compromesso che garantisca delle buone prestazioni e mantenga relativamente bassi i consumi.

I comparti di uso dei processori

Settori che si stanno diffondendo molto negli ultimi anni sono quelli relativi ai SoC e ai DSP. Vediamo un diagramma che mostra più in dettaglio la situazione odierna:



Si osserva che i processori per le macchine general purpose (PC) sono ormai meno del 40% dei processori utilizzati a livello globale.

Inoltre è noto che il mercato dei processori per computer (PC, workstation, ..) è dominato dai processori x86 Intel, con le seguenti quote: Intel 82.8%,AMD 16.9%, Via Technology 0.2%.

Il mercato dei processori per dispositivi PDA, smartphone, tablet è invece basato su sistemi embedded (SoC) che si basano a loro volta su architettura ARM per il 75% del globale. In particolare, Qualcomm detiene il 64% del mercato con un fatturato di 7G$, MediaTek è il secondo produttore con il 12% del mercato, Intel il terzo produttore con 8%.

Appare quindi evidente che i processori x86 e ARM sono ormai quelli più diffusi, ed è il motivo per cui saranno quelli affrontati nel resto del corso.

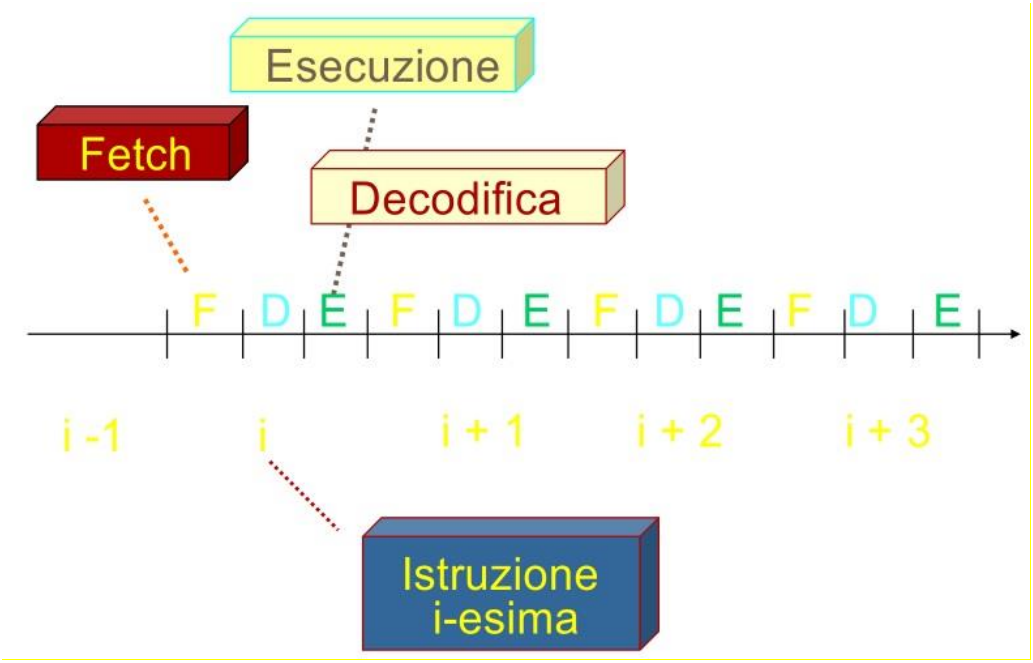
specifiche (special purpose) di controllo digitale. È progettato per interagire direttamente con il mondo esterno tramite un programma residente nella propria memoria interna e mediante l'uso di pin specializzati o configurabili dal programmatore.

** In elettronica digitale un **application specific integrated circuit** (ASIC) è un circuito integrato creato appositamente per risolvere un'applicazione di calcolo ben precisa (specific purpose). All'interno del chip ASIC possono venire integrati vari tipi di componenti come ad esempio transistor bipolari NPN e PNP fino a 80V, N-MOS e P-MOS, CMOS logic, JFET, Foto diodi, Foto transistor e sensori ad effetto Hall.

Confronto tra processori standard e SoC

	System on chip	*Processors on chip*
processor	multiple, simple, heterogeneous	few, complex, homogeneous
cache	one level, small	2-3 levels, extensive
memory	embedded, on chip	very large, off chip
functionality	special purpose	general purpose
interconnect	wide, high bandwidth	often through cache
power, cost	both low	both high
operation	largely stand-alone	need other chips

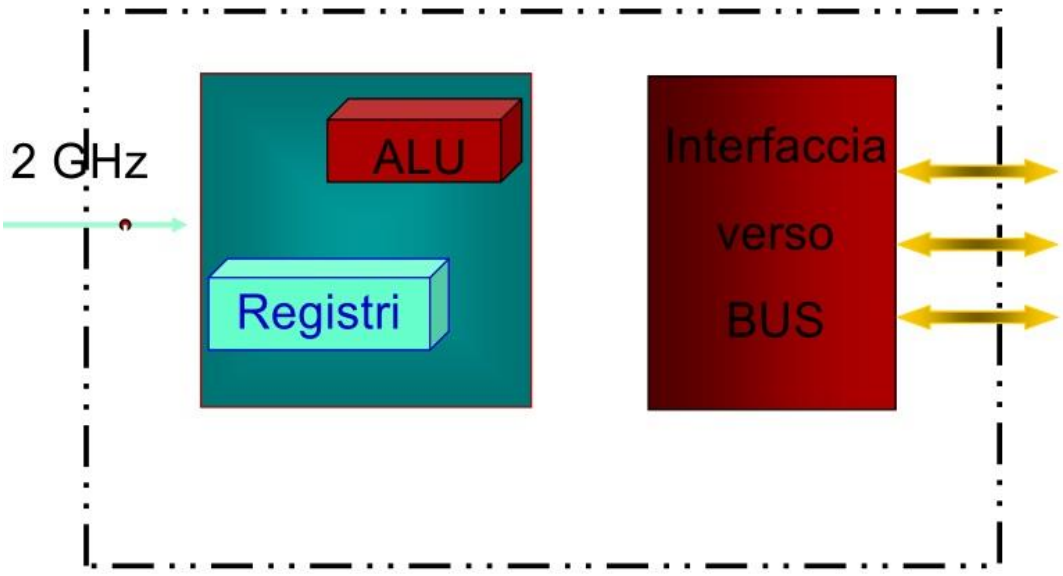
Esecuzione delle istruzioni



Clock interno e di sistema

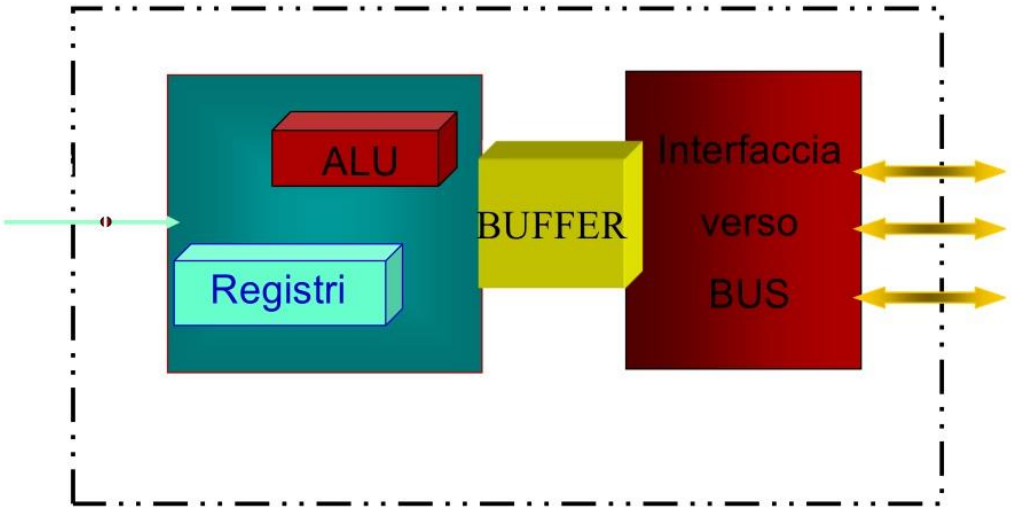
Quando si parla di clock bisogna distinguere tra:

- *clock di CPU* – clock che determina la temporizzazione a livello di CPU
- *clock esterno alla cpu* – possono esserci più clock, uno per ogni sottosistema di bus (livello di bus) presente nel sistema.



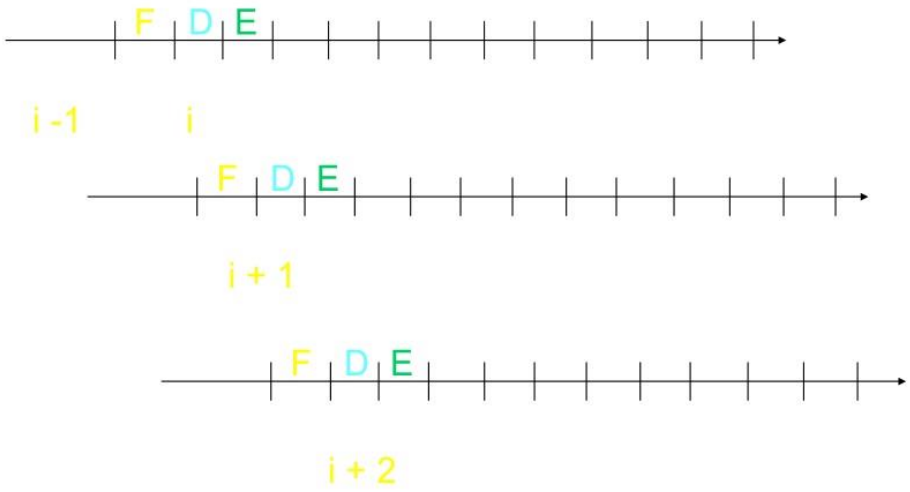
Architettura dei processori – 1° evoluzione

- *Parallelismo tra operazioni interne alla CPU e operazioni esterne (cicli di bus).*
  Per disaccoppiare l'elaborazione dei dati (CPU) dall'accesso alla memoria (dati) è necessario introdurre un sistema di bufferizzazione.



Il buffer introdotto viene detto **prefetch buffer**.

- *Pipeline a livello istruzioni.*

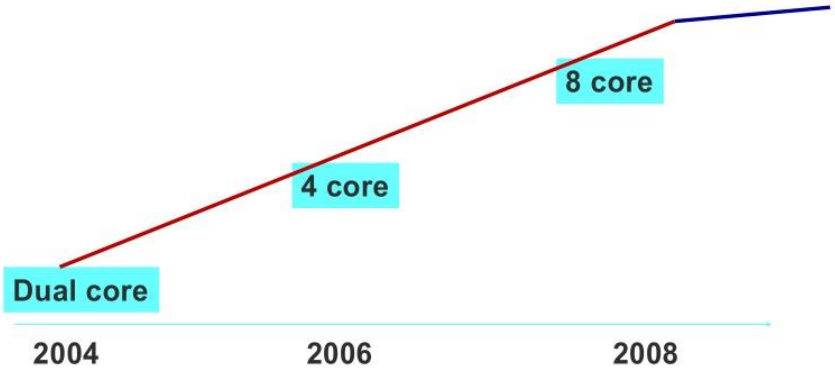


Architettura dei processori – 2° evoluzione

- Architettura a due, quattro, ... core.
- Set di istruzioni a due livelli (assembly level – machine o micro level).
- Parallelismo implicito a livello hw (superscalari,pentium/i7) o compilatore (Itanium-EPIC).

NEW Moore's Law

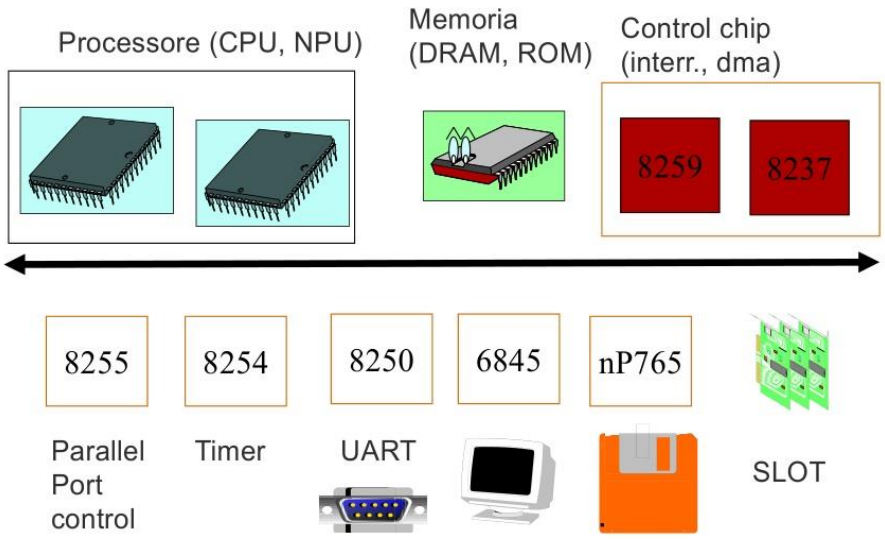
Nuova Legge di Moore: La crescita dei core di CPU all'interno di un microprocessore si duplica ogni 18 mesi.



Perché oggi la curva si è appiattita? Perché vi sono dei limiti al numero di processi paralleli gestibili dai sistemi operativi odierni, per cui, anche all'aumentare del numero di core, non è possibile ottenere miglioramenti significativi delle prestazioni.

Il modello PC (BASE)

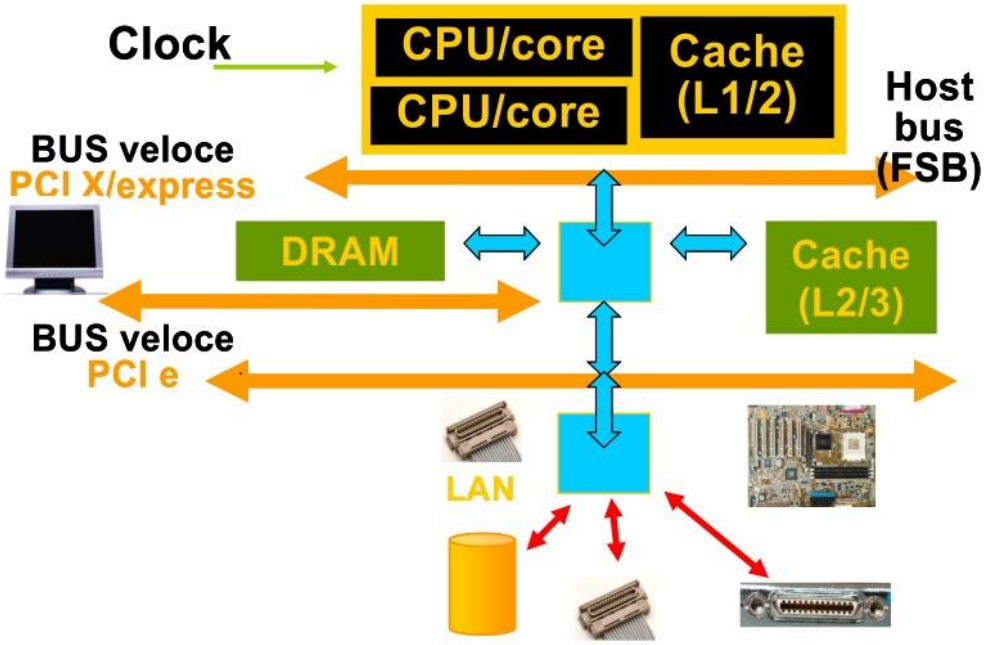
Architettura *single level bus*, nella quale BUS e CPU lavorano alla stessa frequenza (10 MHz in questo caso). Questa è l'architettura a cui fa riferimento il BIOS anche negli attuali PC.



Il chip 8259 è il controllore delle interruzioni.

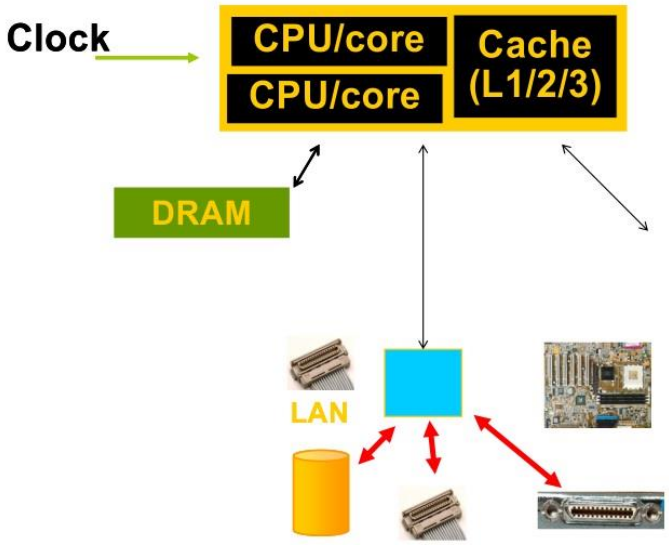
Architettura del PC – 2

Evoluzione dell'architettura PC-1 nella quale si abolisce il Bus ISA e si inserisce un collegamento parallelo diretto con alcune periferiche (Es. tastiera).



Architettura del PC – 3

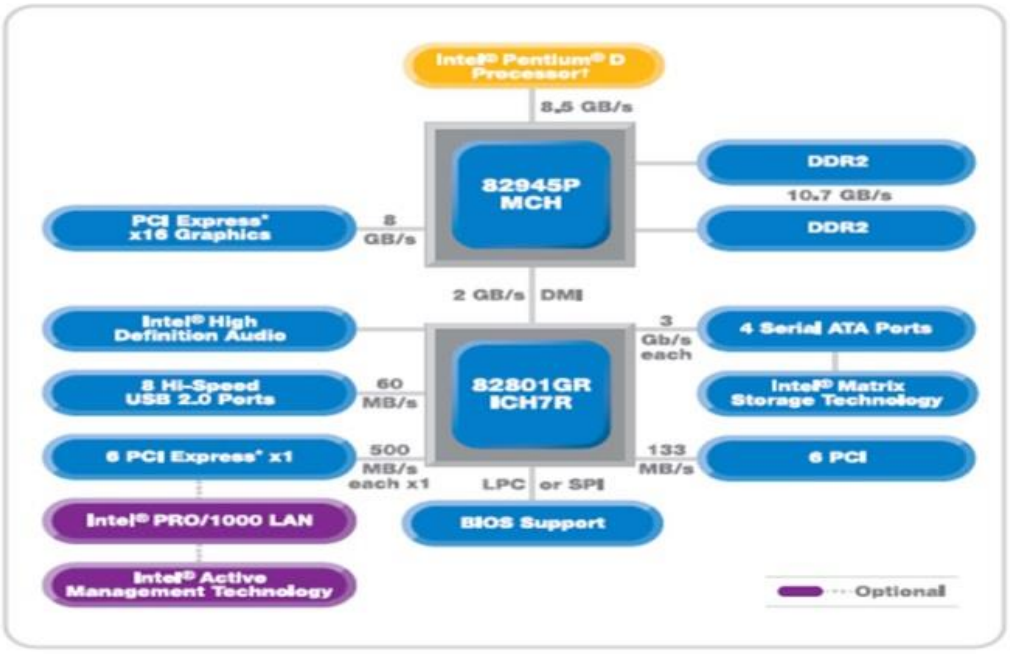
Ultima evoluzione nella quale si elimina un ulteriore livello di bus e si va sempre più verso collegamenti point-to-point.



Ad esempio i processori *i7* odierni hanno un canale diretto con la DRAM e non un bus che fa da intermediario.

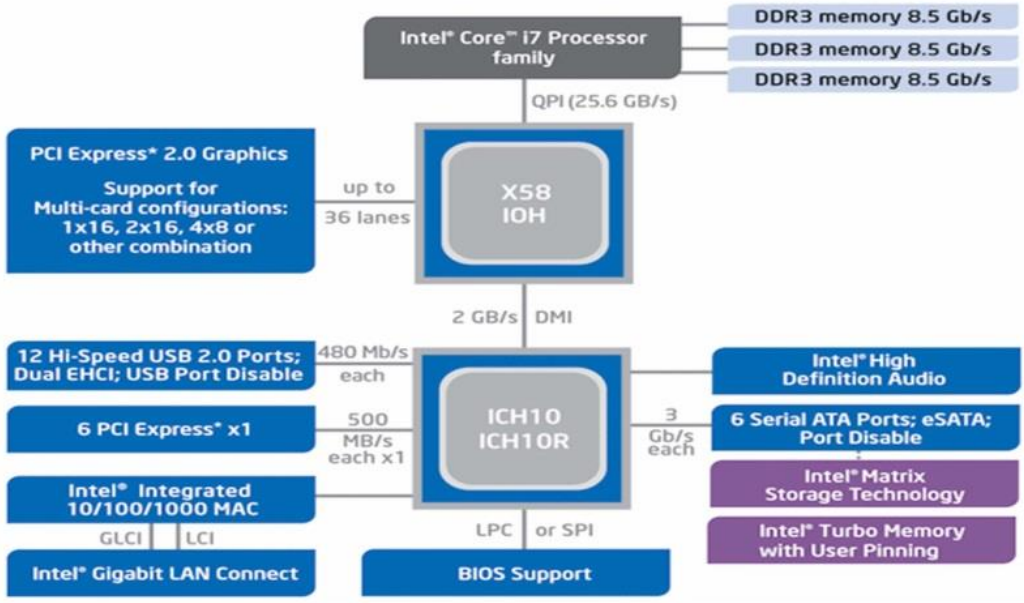
Tuttavia, avere più collegamenti diretti al chip significa aver necessità di un maggior numero di piedini per il chip stesso; un numero più elevato di piedini significa chip di dimensioni notevoli.

- Intel® 945P Express Chipset:



† The Intel® 945P Express Chipset supports the Intel® Pentium® D processor and all other Intel® Pentium® and Intel® Celeron® processors in the LGA775 socket, with scalability for future processor innovations.
*Other names and brands may be claimed as the property of others.

- Core i7:

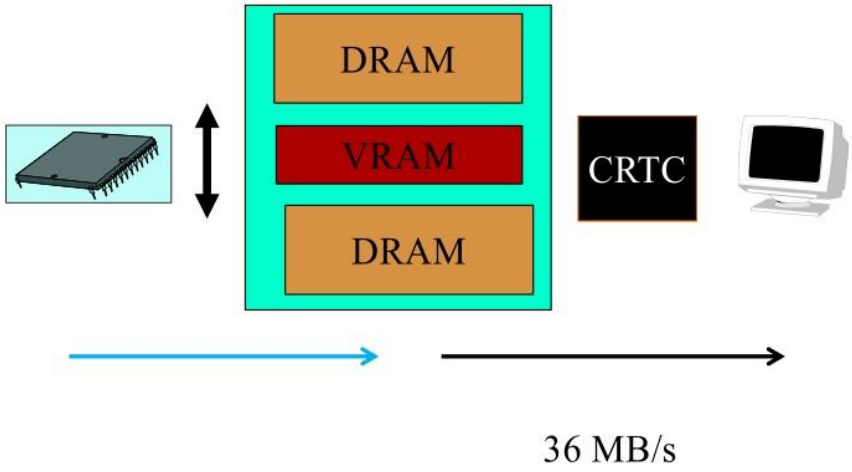


© Proprietà riservata dell'autore - Digitalizzazione e distribuzione a cura del CENTRO APPUNTI - Corso Luigi Einaudi, 55 - Torino / Pagina 41 di 484

appare evidente che sarà necessario inviare 36 MB al secondo:

$$\rightarrow x = \frac{0.72\ MB \times 1\ s}{0.02\ s} = 36\ MB$$

Il **throughput** richiesto è quindi di 36 MB/s.



Caso 1: *sistema sequenziale, monobus*.

Ipotesi:

- CPU: 8086 like, 8MHz (Es. Hitachi HD68000-8);
- Bus: ISA like, 16 bit, 8 Mhz, 4 clock per ciclo.

Supponiamo che il tempo di accesso a memoria sia quello tipico delle attuali DRAM: 60 ns.

Confrontiamo tale tempo col tempo necessario ad eseguire un *ciclo di bus*:

$$ciclo\ di\ bus = n°\ clock\ per\ ciclo \times tempo\ ciclo =$$

$$= 4 \times \frac{1}{8\ Mhz} = 4 \times 0.125\ ms = 0.5\ ms = 500\ ns$$

Essendo $500\ ns \gg 60\ ns$ possiamo considerare il tempo di accesso a memoria pari al tempo necessario ad eseguire un ciclo di bus:

- Tempo di accesso a memoria: 500 ns.

Calcoliamo il *tempo medio necessario per eseguire una istruzione di scrittura in memoria*.

Supponiamo che il numero medio di byte per istruzione sia pari a 2.5 Byte; sapendo che il parallelismo del bus (in questo caso $16\ bit\ =\ 2\ Byte$) indica quanti dati possono essere trasferiti su un ciclo, è possibile dedurre che sono necessari 2 accessi a memoria per la fase di fetch ($2.5\ Byte\ =\ 20\ bit > 16\ bit$).

***Caso 2**: sistema parallelo, monobus + veloce*.

Si supponga di introdurre due miglioramenti:

- Prefetch buffer;
- Bus più veloce (2 clock per ciclo).

Ipotesi:

- CPU: 8086 like, 8MHz (Es. Hitachi HD68000-8);
- Bus: ISA like, 16 bit, 8 Mhz, 2 clock per ciclo:

$$ciclo\ di\ bus = n°\ clock\ per\ ciclo \times tempo\ ciclo = 2 \times \frac{1}{8\ Mhz} = 250\ ns$$

- Tempo di accesso a memoria: 250 ns (discorso analogo a quanto visto in precedenza: $250\ ns \gg 60\ ns$)

Calcoliamo il *tempo medio necessario per eseguire una istruzione*.

Supponiamo che il numero medio di byte per istruzione sia pari a 2.5 Byte, da cui si deduce che sono necessari 2 accessi in memoria per la fase di fetch.

Il tempo medio di esecuzione di una istruzione è dato da: $Tm = Tfetch + Texec$.

Tuttavia, avendo inserito il buffer di prefetch, possiamo considerare nullo il *Tfetch**:

$$Tm = Texec = 250\ ns$$

Il tempo di fetch viene comunque perso, ma grazie al buffer di prefetch, l'operazione di fetch viene eseguita in parallelo alle operazioni successive e quindi Tfetch non è rilevante ai fini del throughput totale.

***N.B.** Il **tempo di esecuzione della singola istruzione è lo stesso** del caso precedente, ciò che cambia è il numero di istruzioni eseguite al secondo.*

Da cui si ricava (in maniera analoga a quanto visto in precedenza): $MIPS = \frac{1\ s}{250 \times 10^{-9} s} = 4$.

Calcoliamo il ***transfer rate del bus*** per verificare l'adeguatezza del bus rispetto alle richieste fatte. Sapendo che per trasferire 16 bit è necessario un ciclo di bus (250 ns):

$$2\ Byte\ (16\ bit) : 250\ ns = x : 1\ s$$

$$\rightarrow x = \frac{2\ Byte\ \times 1\ s}{250 \times 10^{-9}\ s} = 8\ MB$$

Calcoliamo il ***transfer rate del bus*** per verificare l'adeguatezza del bus rispetto alle richieste fatte. Sapendo che per trasferire 16 bit è necessario un ciclo di bus (250 ns):

$$4\ Byte\ (32\ bit): 66\ ns = x : 1\ s$$

$$\rightarrow x = \frac{4\ Byte\ \times 1\ s}{66 \times 10^{-9}\ s} = 60\ MB$$

Per cui: *Transfer rate bus = 60 MB/s*

*In questo caso il **bus** risulta ampiamente sufficiente per soddisfare le richieste (36 MB/s).*

Per l'aggiornamento della memoria video mediante istruzioni (1 istruzione per 2 byte trasferiti), tenendo conto che il *Tm* è pari a 66 ns, è richiesto un tempo (min) pari a:

$$\frac{0.72MB \times 66\ ns}{4^{*}} = \ 12\ ms$$

**Avendo un bus a 32 bit trasferiamo 4 Byte alla volta.*

Si osserva che in questo caso $12\ ms < 20\ ms$, per cui l'intera architettura è sufficiente a soddisfare la richiesta; tuttavia il sistema può essere utilizzato solo per l'operazione di aggiornamento della memoria, infatti *12 ms* è più del *50%* di *20 ms* ("fa solo quello!").

***Caso 4**: cache - multi level bus*.

Si supponga di introdurre i seguenti miglioramenti:

- Cache multilivello;
- Multi level Bus (AGP/PCI-e, PCI, ISA).

Iniziamo con il calcolare il *tempo medio di esecuzione di una istruzione*.

$$Tm = media(\ Tcpu, Tmem, Tio)$$

Dove:

- *Tcpu* = tempo esecuzione istruzioni eseguite solo nella CPU; tempo che dipende dal clock della CPU.
- *Tmem* = tempo esecuzione istruzioni di accesso a memoria; questo tempo dipende come sempre dal tempo di accesso a memoria e dal clock del bus di memoria.
- *Tio* = tempo esecuzione istruzioni di I/O; tempo che dipende dalla velocità dell'**interfaccia** del disco e dal clock del bus di I/O.

Possiamo quindi calcolare il tempo medio di esecuzione di istruzione come segue:

$$Tm = 25\% \times Tcpu + 70\% \times Tmem + 5\% \times Tio =$$

- Bus ISA: $= 0.25 \times 5\ ns + 0.7 \times 15\ ns + 0.05 \times \mathbf{200\ ns} = 22\ ns$
- Bus PCI: $= 0.25 \times 5\ ns + 0.7 \times 15\ ns + 0.05 \times \mathbf{33\ ns} = 13.4\ ns$

Si osservi come anche percentuali di incremento significativo della velocità interna (CPU) non necessariamente modifichino drasticamente le prestazioni. Ad esempio una prestazione del 100% in più (ovvero tempo di *Tcpu* dimezzato) riduce solo di circa il 5% il tempo totale (*Tm*) e quindi influenza di pochissimo i MIPS di sistema.

Osservazione

Se ipotizzassimo di voler calcolare il tempo necessario ad eseguire il refresh dello schermo:

1. Non dovremmo considerare il tempo di accesso alla cache ma solo il tempo di accesso alla VRAM, in quanto la memoria video (VRAM) non viene mai cachata.
2. Bisognerebbe assumere come *Tm* quello calcolato rispetto al bus PCI, in quanto si è ipotizzato che le istruzioni di read/write della memoria video utilizzino il bus PCI.

Pertanto:

Ti = tempo di read/write memoria (60 ns)

Come precedentemente detto, il tempo di esecuzione di istruzioni in memoria dipende anche dal clock del bus, per cui *Ti* è pari a 2 cicli di bus:

$$Ti = 66\ ns$$

Tenendo conto che il *Ti* è pari a 66 ns si ottiene:

$$\frac{0.72MB \times 66\ ns}{4} = 12\ ms$$

Il risultato ottenuto è analogo a quello calcolato nel caso precedente senza cache!

(*Esercizio*: provare a considerare il caso in cui si utilizza il bus AGP<PCI-e)

Si veda il caso di due istruzioni necessarie per leggere da I/O e scaricare in memoria:

IN AH, ind. Lan
MOV mem, AH

Dove:

- La prima istruzione legge da periferico (ind. Lan) nel registro AH;
- La seconda istruzione scrive il registro AH appena letto, in memoria.

Il tempo di esecuzione delle istruzioni non è determinato dal clock del processore (1 ns) ma dalla transazione «fuori» del processore, per cui, ipotizzando un tempo di accesso a memoria estremamente rapido, 30 ns, otteniamo:

- Per l'operazione IN è necessario un ciclo di bus PCI → *60 ns*;
- Per l'operazione MOV supponiamo di poter considerare solo il tempo di accesso alla memoria (ipotizzando ad esempio che la memoria sia direttamente connessa alla CPU e non sia necessario utilizzare un bus più lento) → *30 ns*;

Ne deriva un tempo di esecuzione delle *due istruzioni* pari a *90 ns (>80 ns)*. Questo significa che *il sistema **non** è compatibile con il thoughput del dispositivo!*

Alternativa 1

Si supponga di utilizzare una interfaccia di LAN a **16 bit**. In tale situazione avremo:

$$100\ Mbit : 1\ s = 16\ bit : x\ s$$

$$\rightarrow x = \frac{16\ bit \times 1\ s}{100 \times 10^6\ bit} = 160\ ns$$

Questo vuol dire che arrivano 2 Byte ogni 160 ns, per cui il bus diventa un po' più libero. Inoltre, considerando che per leggere i dati dall'I/O e trasferirli in memoria sono necessari 90 ns (in base ai calcoli fatti in precedenza), *il sistema risulta* in questo caso *compatibile con il thoughput del dispositivo*.

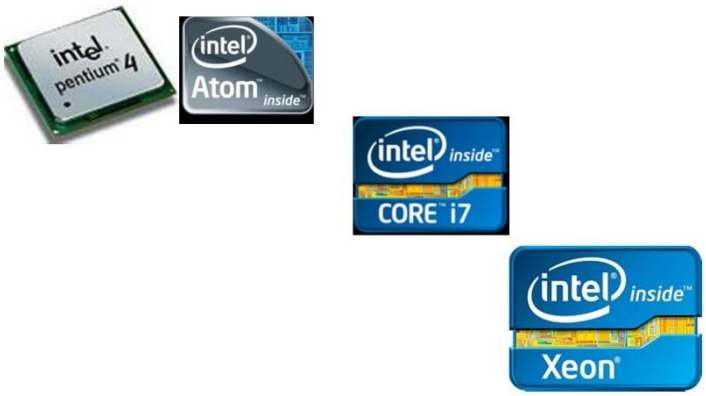
Architettura x86 – 32/64 bit

La famiglia INTEL

La famiglia Intel è costituita al giorno d'oggi da 4 tipi di processori:

- Architettura Pentium 4;
- Architettura Atom: architetture meno performanti;
- Architettura core iX (con X=3,5,7);
- Architettura Xeon.

Sono state elencate in ordine di capacità di elaborazione (Pentium 4 capacità di elaborazione più bassa, Xeon capacità di elaborazione massima).



Vediamo adesso alcune **specifiche tecniche**.

Atom

Processor	Clock Speed(s)	Intro Date(s)	Mfg. Process	Transistors	Cache	Addressable Memory	Bus Speed	Typical Use
Intel® Atom™ Processor Z500	800 MHz	Apr-08	45nm	47 million	512 kB L2	4 GB	400MT/s	MID

Dove:

- MID – Mobile Internet Device;
- Mfg Process: indica il processo di costruzione usato. Si osservi che ad oggi si arriva a valori intorno a *20 nm*.

Si osservi che avere una memoria indirizzabile (*Addressable Memory*) di *X Byte*, implica che il processore produce degli **indirizzi** di $\log_2 X$ bit (Es. $4\ GB \rightarrow \log_2 4GB = 32\ bit$).

Si noti che la soluzione multi-core è diversa dalla tecnologia HT che si applica ad un solo core. Nei sistemi multi-core ad ogni core può essere applicato un processo (programma) diverso, mentre nella tecnologia HT si può "solo" realizzare il multi thread per lo stesso processo.

Pentium 4

Processor	Clock Speed(s)	Intro Date(s)	Mfg. Process	Transistors	Cache	Addressable Memory	Bus Speed	Typical Use
Intel® Pentium® D Processor 800	3.20 - 2.80 GHz	May-05	90nm	230 million	1024 kB L2 Cache	64 GB	800 MHz	Desktop PC

Core i7

Processor	Clock Speed(s)	Intro Date(s)	Mfg. Process	Transistors	Cache	Addressable Memory	Bus Speed	Typical Use
Intel® Core™ i7-965 Extreme Edition	3.20 GHz	Nov-08	45nm	731 million	8 MB	64 GB		Desktop PC

Xeon

Processor	Clock Speed(s)	Intro Date(s)	Mfg. Process	Transistors	Cache	Addressable Memory	Bus Speed	Typical Use
Intel® Xeon® Processor MP X7460	2.66 GHz	Sep-08	45nm	1.90 billion	16 MB	1 TB	1066 MHz	Server

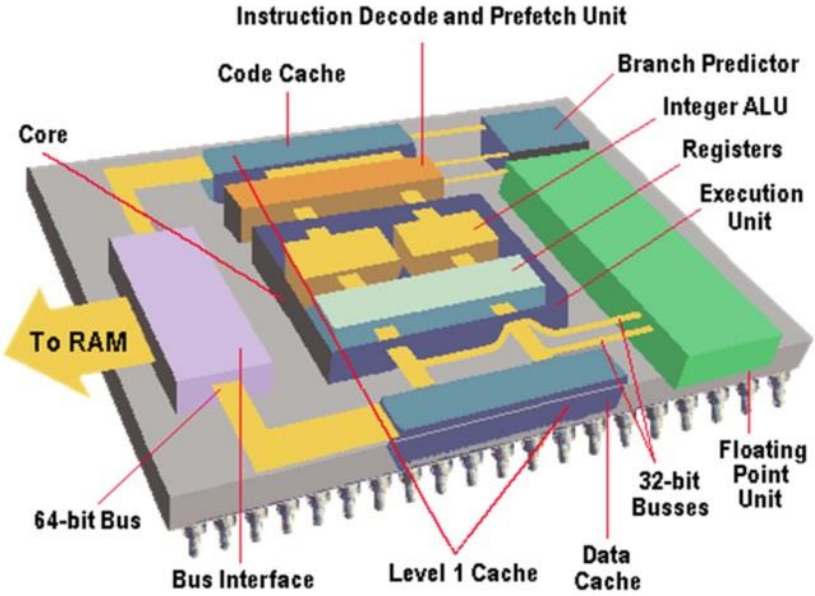
Si osserva che con l'evoluzione dei processori, ciò che è via via aumentato è la quantità di **cache** contenuta all'interno del processore (e di conseguenza il numero di transistor).

Architettura 80x86

Sui dispositivi visti sino ad ora è possibile realizzare 3 tipi di architetture:

- **X86-16** definita anche **IA-16** (Intel e AMD):
  - Indirizzamenti su 16 bit (offset): la singola istruzione è in grado di indirizzare fino a 64 KB (*spazio di indirizzamento* di 64 KB).
  - Registri 8, 16 bit;
  - Nativa su 8086 e 286.

BIOS per la gestione delle periferiche, ma usano delle routine native che sono in modo protetto ma operano su 16 bit.



Caratteristiche Generali del Pentium IV - *Curiosità*

Il Pentium 4 originale, chiamato "Willamette", lavorava ad 1.4GHz e 4.5 GHz ed è stato rilasciato nel Novembre del 200 sulla piattaforma *Socket 423* e successivamente sulla piattaforma *Socket T* con un clock tra 1.5 GHz e 3.8 GHz. Con l'introduzione del Pentium 4, notevole fu il relativo incremento di velocità degli FSB (da 400 MT/s a 1066 MT/s).

Il Pentium 4 è stato anche prodotto in una versione di fascia bassa chiamata *Celeron* (alla quale ci si riferisce con il nome di Celeron 4) e una di fascia alta chiamata *Xeon* (versione destinata alle configurazioni SMP*).

Una versione dual core del Pentium 4 è stata chiamata *Pentium D* e ad oggi ci si riferisce ad essa come *Pentium Dual Core*.

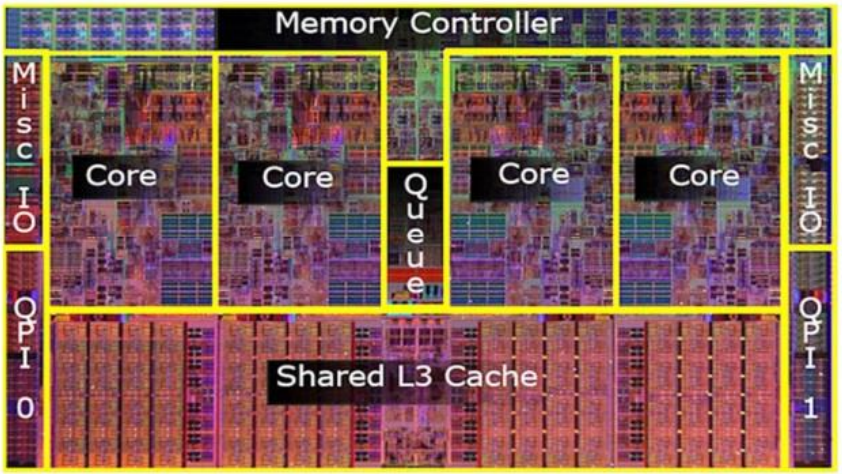
*Il multiprocessore simmetrico (Symmetric MultiProcessing - SMP) è un'architettura hardware dotata di più processori, nella quale questi possono accedere equamente a tutta la memoria RAM disponibile.

Caratteristiche Generali del Core i7

- Versione del 2008, con tecnologia di 22-45 nm;
- Clock: 2.6 – 3.5 GHz
- Numero di core: 4 – 6 core;
- QPI*;
- Hyper-Threading incorporato;
- Architettura EM64.

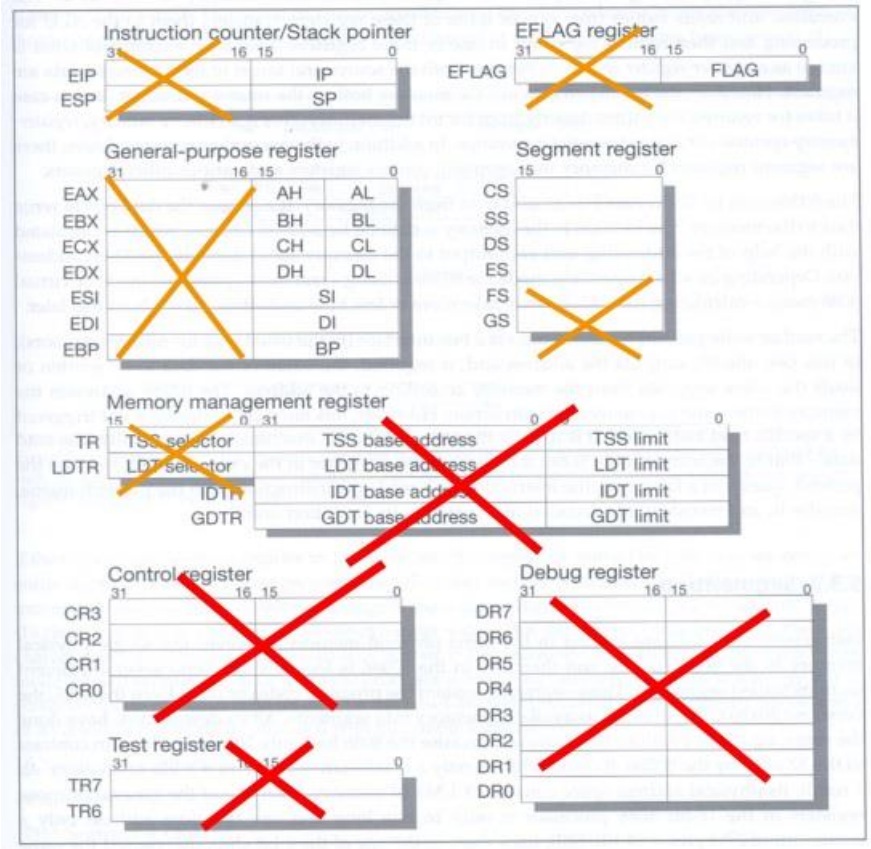
*Con il nome di Intel QuickPath Interconnect o **QPI** (precedentemente conosciuto come Common Systems Interconnect o CSI) Intel indica un nuovo tipo di bus seriale che, a partire dal 2008, è stato introdotto in alcuni processori nei settori desktop e server di marca Intel.

Lo scopo primario del BUS QPI è quello di permettere al processore di comunicare direttamente con i vari altri componenti collegati alla motherboard, beneficiando quindi di una banda passante maggiore e di latenze sempre più ridotte. La caratteristica peculiare di QPI è quella di essere una tecnologia di connessione "point-to-point" che elimina gli svantaggi portati da un solo BUS condiviso tra tutti i processori, il controller memoria e il controller I/O.



I registri in modo reale

I registri disponibili in modo reale sono quelli **non** barrati in figura.



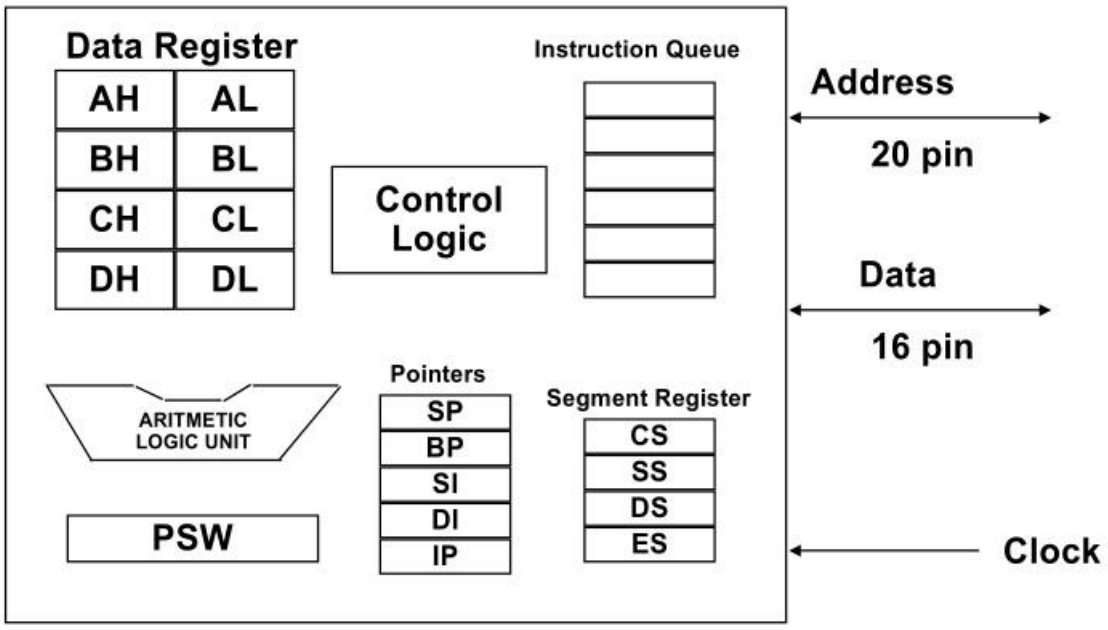
Registri

I registri possono essere suddivisi in 4 gruppi:

- Registri di dato;
- Registri puntatore/di indirizzo;
- Registri di segmento;
- Registri di controllo.

Modello Architetturale in modo reale (8086 like)

Il modello concettuale dell'8086 (e del modo reale in generale) è mostrato in figura.



Si ha:

- Indirizzamento su 20 bit → Memoria totale utilizzabile in modo reale = 1 MB
- Dati su 16 bit

L'architettura è basata su 8 registri *General Purpose* da 8 bit che rappresentano una estensione degli accumulatori. Tali registri sono:

AH	AL
BH	BL
CH	CL
DH	DL

Dove:

- *H* sta per High.
- *L* sta per Low.

Tali registri possono essere combinati tra loro a formare registri da 16 bit (Es. AX=AH+AL, BX=BH+Bl, etc.)

Essendo entrambi registri a 16 bit nell'8086, vuol dire che è possibile indirizzare al più 65536 (64K) segmenti ognuno di 65536 byte (64 KByte).

Registri di Dato

Sono **AX** (Accumulator Register), **BX** (Base Register), **CX** (Count Register) e **DX** (Data Register). Sono utilizzati per memorizzare operandi e risultato delle operazioni.

Possono essere utilizzati come registri da 16 bit oppure come coppie di registri da 8 bit.

- **BX** può anche essere utilizzato nel calcolo di indirizzi, in particolare per specificare l'offset.
- **CX** viene anche utilizzato come contatore da talune istruzioni (Es. loop).
- **DX** contiene l'indirizzo di I/O in alcune istruzioni di I/O.

Registri Puntatore

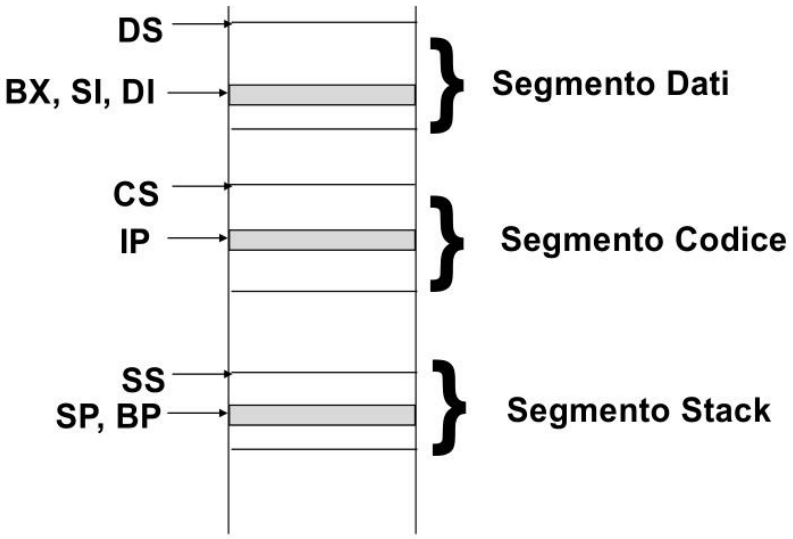
Sono IP, SP, BP, SI e DI.

- **IP** (Instruction Pointer) contiene il puntatore alla prima istruzione da eseguire. IP non può comparire esplicitamente come operando di una istruzione.
- **SP** (stack pointer) contiene il puntatore alla testa dello stack.
- **BP** (Base Pointer) viene utilizzato come base per fare accesso all'interno dello stack.
- **SI** (Source Index) e **DI** (Destination Index) vengono utilizzati come registri indice.

Registri di Segmento

Sono CS, DS, ES e SS. Vengono utilizzati per costruire gli indirizzi fisici con i quali fare accesso in memoria.

Contengono i puntatori all'inizio dei segmenti di codice, di dato, di dato supplementare e di stack, rispettivamente.



Si osservi che sono i bit dell'address bus che limitano la dimensione massima della memoria utilizzabile (Es. Address Bus a 16 bit → Massima memoria indirizzabile = 64 Kbyte).

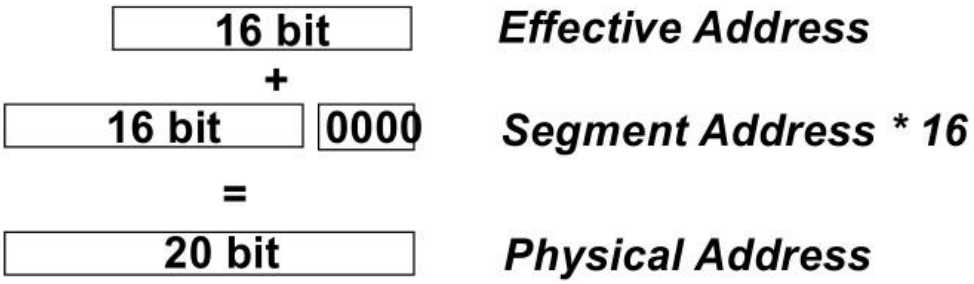
Nell'80x86 esistono due modalità che possono essere utilizzate per tradurre gli indirizzi:

- **Modo Reale**
- **Modo Protetto.**

Nel **modo reale** ogni volta che l'8086 (o un qualsiasi altro processore che opera in modo reale) deve generare un indirizzo da mettere sull'A-bus (physical address), esso esegue una operazione di somma tra il contenuto di un registro puntatore oppure di BX (effective address o offset) ed il contenuto di un registro di segmento (segment address).
La somma avviene dopo aver moltiplicato per 16 (shift a sinistra di 4 posizioni) il contenuto del registro di segmento.

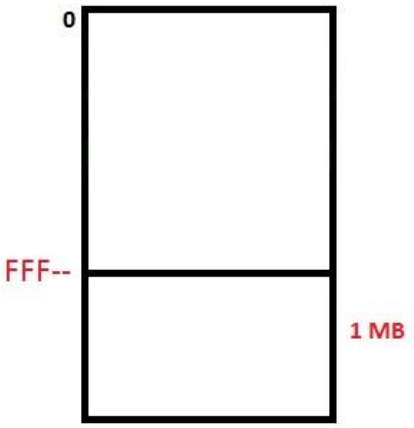
Nel dettaglio:

- Si prende il registro che identifica il segmento (16 bit) e si aggiungono 4 bit a 0 da destra.
- i 20 bit ottenuti vengono sommati al registro che contiene l'offset



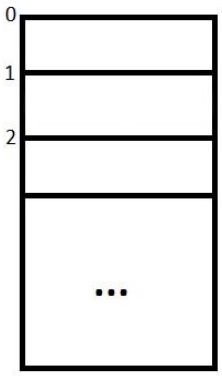
Si ottiene quindi un indirizzo di 20 bit che è la traduzione dell'indirizzo del tipo: *<Segmento>:<Offset>*. Questi 20 bit vengono inviati all'Address bus.

Si osservi che se l'address bus ha più di 20 bit, i bit più significativi (a sinistra) restanti vengono forzati ad 1. Questo significa che la porzione di memoria che viene utilizzata in modo reale è quella alta (ovvero con indirizzi più alti).

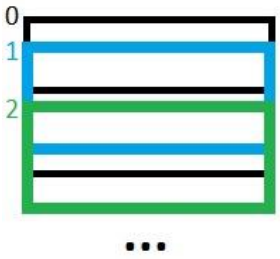


dall'altro di soli 16 byte (Es. 0x00010 → 0x00020). Questa sovrapposizione è stata realizzata per far coesistere dei dati comuni a più segmenti.

- Segmenti contigui:



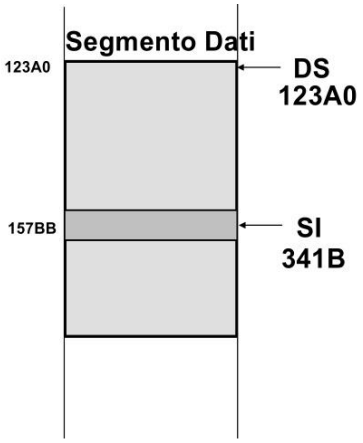
- Segmenti sovrapposti:



Come già accennato, l'utilizzo dei segmenti sovrapposti è utile quando si ha poca memoria fisica a disposizione, in quanto permette di poter condividere delle aree di memoria tra processi differenti senza necessità di dover copiare i dati in più i segmenti (operazione necessaria nel caso si segmenti contigui).

Uso dei Segment Register

Una volta caricato l'indirizzo di testa del segmento in un segment register, tutti gli indirizzi all'interno del segmento sono esprimibili attraverso un registro contenente l'offset.



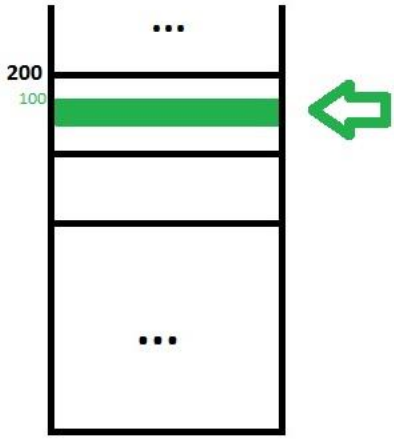
Istruzioni di Jump

Si possono avere diverse istruzioni di **Jump**

- **Jump Far**: si salta ad un indirizzo contenuto in un segmento diverso da quello corrente (cambia sia il segment register che il registro di offset).
  Ad esempio:

  JMP 200:100

  Dove si intende che si vuole eseguire l'istruzione che si trova alla posizione 100 del segmento 200.



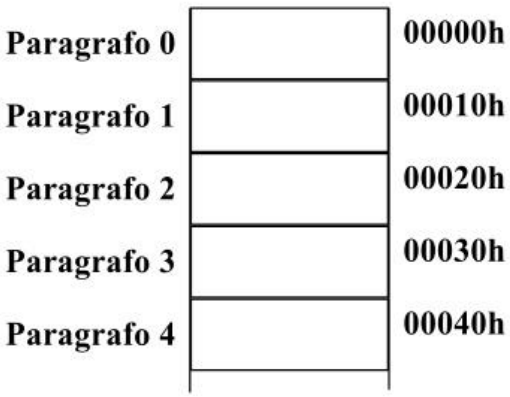
- **Jump Near**: si salta ad un indirizzo interno al segmento corrente (cambia solo il registro di offset).
  Ad esempio:

  JMP IP

  Dove (sapendo che PC=CS+IP) si assume che CS sia pari al valore di CS relativo al PC corrente. In pratica viene solamente calcolato il nuovo offset.

Paragrafi

I gruppi di 16 byte che iniziano ad indirizzi multipli di 16 si definiscono **paragrafi**. La memoria è quindi organizzata in paragrafi.



Flag di Condizione

Vengono automaticamente scritti al termine di varie operazioni:

- **SF** (Sign Flag): coincide con il MSB del risultato dopo una operazione aritmetica;
- **ZF** (Zero Flag): vale 1 se il risultato è nullo, 0 altrimenti;
- **PF** (Parity Flag): vale 1 se il numero di 1 negli 8 bit meno significativi del risultato è pari, 0 altrimenti;
- **CF** (Carry Flag): dopo le operazioni aritmetiche vale 1 se c'è stato riporto (somma) o prestito (sottrazione); altre istruzioni ne fanno un uso particolare;
- **AF** (Auxiliary Carry Flag): usato nell'aritmetica BCD; vale 1 se c'è stato riporto (somma) o prestito (sottrazione) dal bit 3;
- **OF** (Overflow Flag): vale 1 se l'ultima istruzione ha prodotto un overflow.

Flag di Controllo

Possono venire scritti e manipolati da apposite istruzioni, e servono a regolare il funzionamento di talune funzioni del processore:

- **DF** (Direction Flag): utilizzato dalle istruzioni per la manipolazione delle stringhe; se vale 0 le stringhe vengono manipolate partendo dai caratteri all'indirizzo minore, se vale 1 a partire dall'indirizzo maggiore;
- **IF** (Interrupt Flag): se vale 1, i segnali di Interrupt mascherabili vengono percepiti dalla CPU, altrimenti questi vengono ignorati;
- **TF** (Trap Flag): se vale 1, viene eseguita una trap al termine di ogni istruzione.

Accesso alla Memoria

Il modello IA - 16 ha uno spazio di indirizzamento pari a 1 Mbyte.

Il processore è in grado di accedere in un solo passo ad un byte oppure ad una word di memoria, purché questa inizi ad un indirizzo pari. L'accesso a word allineate su indirizzi dispari richiede due cicli di memoria.

Lo Stack

L'80x86 prevede alcune strutture e meccanismi hardware per la gestione di uno stack.

Lo stack corrisponde al segmento di memoria la cui testa è puntata da SS. Il top dello stack (locazione riempita per ultima) è puntato da SP (o ESP, in IA – 32).

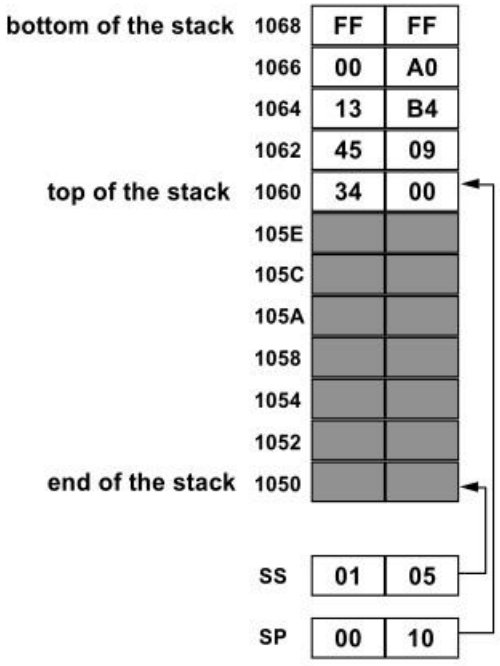
Lo stack cresce dalle locazioni di memoria con indirizzo maggiore verso quelle ad indirizzo minore.

Le istruzioni per manipolare lo stack sono:

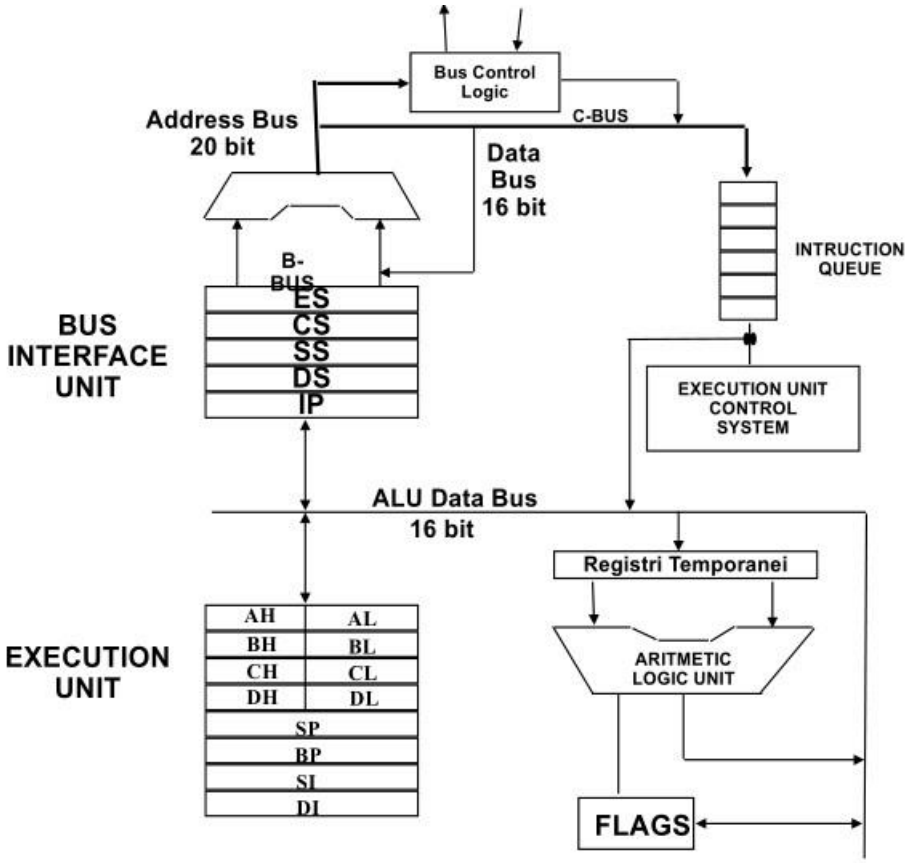
- **PUSH** – scrive dato nello stack all'indirizzo individuato dal puntatore e poi lo incrementa.
  - Ogni operazione di PUSH decrementa di 2 unità SP e scrive una word nella locazione da questo puntata.

- **POP** – legge dallo stack dall'indirizzo individuato dal puntatore e poi lo decrementa.
  - Ogni operazione di POP estrae una word dalla locazione puntata da SP, e successivamente incrementa SP di 2 unità.

Esempio di Stack



Architettura Interna semplificata (modo reale)



Execution Unit (EU)

Provvede alla decodifica ed alla esecuzione delle istruzioni.

Riceve byte per byte le istruzioni dalla BIU, le decodifica, genera gli indirizzi degli operandi (se necessario), li passa alla BIU; una volta ricevuti tutti gli operandi esegue l'istruzione, testa ed aggiorna i flag.

Bus Interface Unit (BIU)

Gestisce tutte le operazioni da e per l'esterno:

- Fetch delle istruzioni;
- Lettura e scrittura operandi e risultati di istruzioni;
- Generazione degli indirizzi;
- Accodamento delle istruzioni;
- La BIU lavora in parallelo con la EU.

Pipeline

Aumento prestazioni

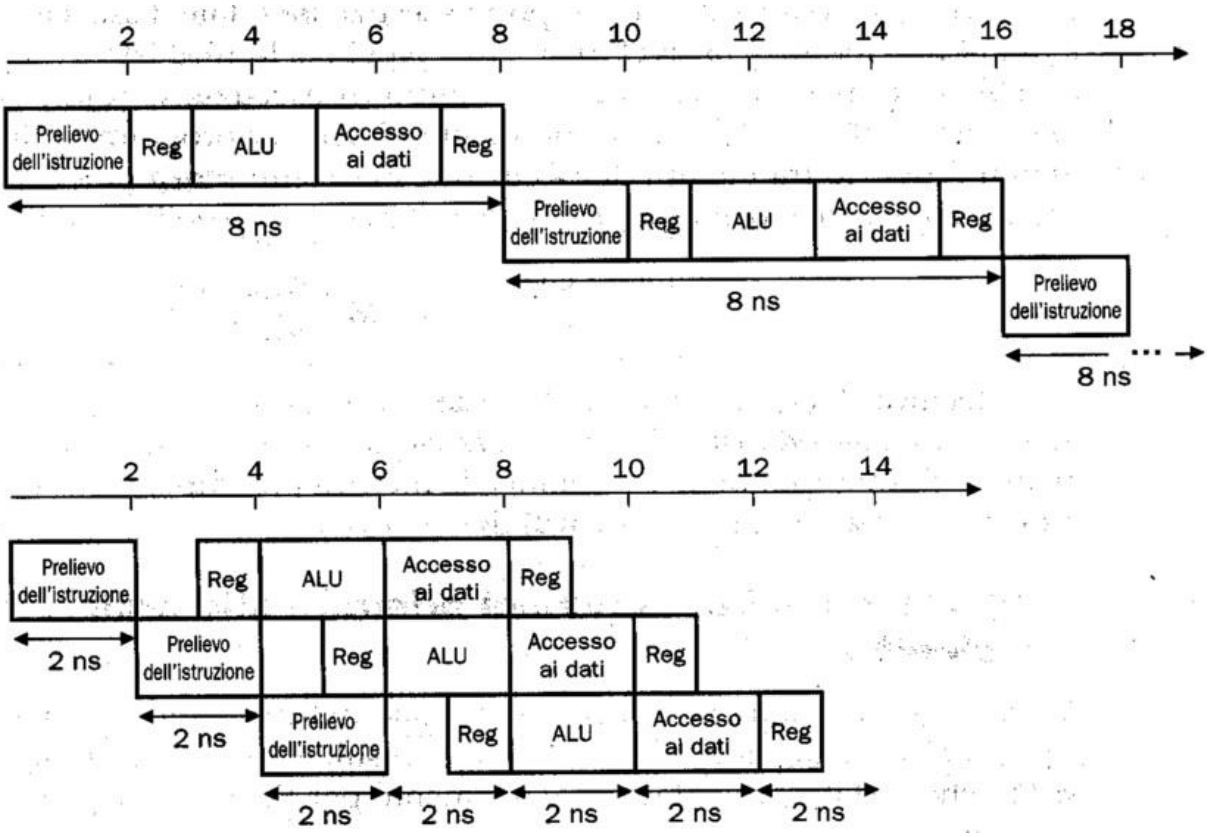
Possono essere applicate diverse soluzioni al fine di aumentare le prestazioni di un sistema:

- Incrementando esecuzione delle singole istruzioni (ILP) mediante **pipeline**;
- Incrementando il numero di istruzioni eseguite in parallelo mediante **processori superscalare** (ILP);
- Incrementando il numero di porzioni di codice (**threads**) eseguite in parallelo (TLP).

Pipeline

Il meccanismo di pipeline è nato agli inizi del Novecento per aumentare la produttività delle catene di montaggio delle automobili.

Per quanto riguarda la pipeline a livello di istruzioni, ogni istruzione può essere divisa in due fasi (Fetch ed Execute) che sono indipendenti (perché afferiscono ad unità indipendenti) e possono essere sovrapposte.



- **Pipeline imperfetta**: Un meccanismo di pipeline imperfetta è costituito da una serie di stadi che eseguono un compito specifico e che hanno un *tempo di esecuzione differente* a seconda dello stadio dell'istruzione.
  In una pipeline di questo tipo non si ha una perfetta sovrapposizione delle istruzioni e sono quindi inevitabili dei "tempi morti".

Il tempo di avanzamento della pipeline ($\Delta \boldsymbol{t_a}$) può essere definito come segue:

$$\Delta t_a = \frac{1}{clock\ pipeline} = max_i\{\Delta t_i\}$$

Dove i vari Δt_i sono i tempi di esecuzione dei singoli stadi.

Questo significa che il tempo di esecuzione di ogni stadio di pipeline è pari al tempo di esecuzione dello stadio più lungo.

Per quanto riguarda la pipeline di istruzioni è necessario fare una distinzione:

- **Tempo di esecuzione di una istruzione** (T_i) che è dato dalla somma dei tempi necessari a percorrere tutti gli stadi di pipeline.
$$T_i = \sum_{j=1}^{N} \Delta t_j$$
- **Numero di istruzioni eseguite** (in un secondo) del processore, indicato come *MIPS* o *througput*, che dipende dal clock della pipeline e dal grado di superscalarità del processore.

La **pipeline** *permette di aumentare il troughput del sistema*, ovvero il numero di istruzioni eseguite, ma *non permette, ovviamente, di ridurre il tempo di esecuzione della singola istruzione*!

Anzi, si osservi che, detto:

- T_s il tempo di esecuzione di una istruzione in un sistema *senza pipeline*;
- T_p il tempo di esecuzione di una istruzione in un sistema *con pipeline*;

Tipicamente risulta:

$$\boldsymbol{T_s \leq T_p}$$

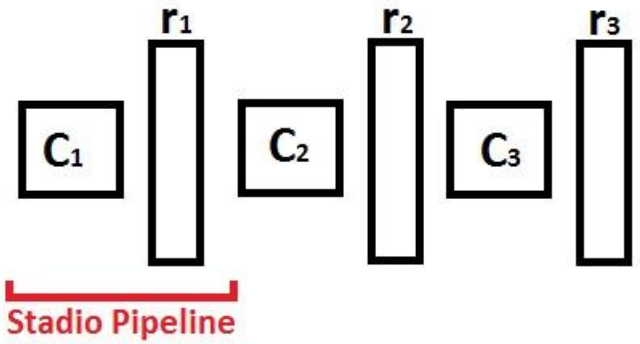
Ciò avviene nel caso di pipeline imperfetta o nel caso di blocco del pipeline perfetto.

Questo significa che il tempo di esecuzione delle **singole istruzioni** in sistemi pipeline è maggiore rispetto ai sistemi senza pipeline, tuttavia il **throughput** finale è migliore nei sistemi con pipeline.

La soluzione ideale è quella in cui ogni coppia C_x,r_x costituisce uno stadio della pipeline, ovvero una situazione nella quale la frequenza di avanzamento della pipeline (f_a) è uguale alla frequenza di clock della pipeline (f_c).

$$f_a = f_c$$

Questo significa che ad ogni colpo di clock si avanza di uno stadio nella pipeline.



Quando $f_a = \frac{f_c}{x}$, vuol dire che uno stadio della pipeline impiega *x* colpi di clock; in tal caso ogni stadio di pipeline è uno *stadio sequenziale* (macchina a stati) e non combinatorio.

Problemi della Pipeline

Le situazioni problematiche che si possono verificare in una pipeline e che portano a delle penalizzazioni di esecuzione sono:

- **Stallo** – situazione nella quale non è possibile terminare l'esecuzione di uno stadio di pipeline entro il tempo di esecuzione fissato per il generico stadio. In tal caso è necessario bloccare tutte le istruzioni presenti nella pipeline finché lo stadio che ha generato lo stallo non termina l'esecuzione.

  Se lo stallo porta ad una penalizzazione di esecuzione pari ad $n \times \frac{1}{f_c}$ per l'istruzione interessata, tutte le istruzioni della pipeline saranno soggette alla stessa penalizzazione di esecuzione.

- **Svuotamento** – situazione nella quale è necessario svuotare la pipeline in quanto è stata riempita di istruzioni che non devono realmente essere eseguite; questo avviene in caso di speculazione errata (Es. salti condizionati).

  Più lunga è la pipeline, peggiore è il problema dello svuotamento, in quanto aumenta il numero di istruzioni che è necessario eliminare dalla pipeline.

Ne deriva che i **MIPS di picco** di una architettura pipeline risultano:

$$\tau_{max} = m \times periodi\ di\ clock\ CPU = \frac{m}{frequenza\ clock\ CPU}$$

$$MIPS = \frac{1}{\tau_{max}} = \frac{frequenza\ clock\ CPU}{m}$$

Essendo m il numero di periodi di clock richiesti per eseguire le operazioni in uno stadio di pipeline.

Nei processori RISC, m in genere vale **1**, ne deriva che un processore a 1 GHz ha $MIPS = 1000\,\frac{istruzioni}{s}$, essendo questo il valore max.

In un **processore superscalare**, con più pipeline in parallelo i MIPS sono dati:

$$MIPS = \frac{frequenza\ clock\ CPU}{m} \times numero\ di\ pipeline\ in\ parallelo$$

Questi calcoli sono fatti nel caso migliore (Best case), ovvero nell'ipotesi in cui non avvengano né stalli né svuotamenti di pipeline.

Dimensionamento Pipeline

Le prestazioni legate all'architettura pipeline dipendono da tre fattori:

- Il **clock** della pipeline, che determina il throughput (istruzioni/s) e il tempo di esecuzione di ciascuna istruzione. Il clock dipende dal tempo impiegato dai singoli stadi.
- Il **numero degli stadi**, che determina la penalità in caso di conflitto/svuotamento.
- Il **numero di pipeline** parallele, che determina il numero complessivo di istruzioni/s.

Pipeline nel Pentium

Nel processore Pentium di base esiste una vera pipeline di 5 stadi:

1. **F** - la fase di **fetch** in cui viene caricata l'istruzione dalla coda istruzioni.
   Nel caso in cui l'istruzione non sia presente nella coda istruzioni è necessario caricarla da (L1) Cache o da RAM, generando almeno uno stallo.

2. **D1** - la fase di **decodifica** dell'istruzione, in cui viene decodificata l'istruzione e vengono eseguiti i controlli di privilegio e di attributi in *modo protetto*; viene in pratica verificata la realizzabilità dell'istruzione.

Processore superscalare

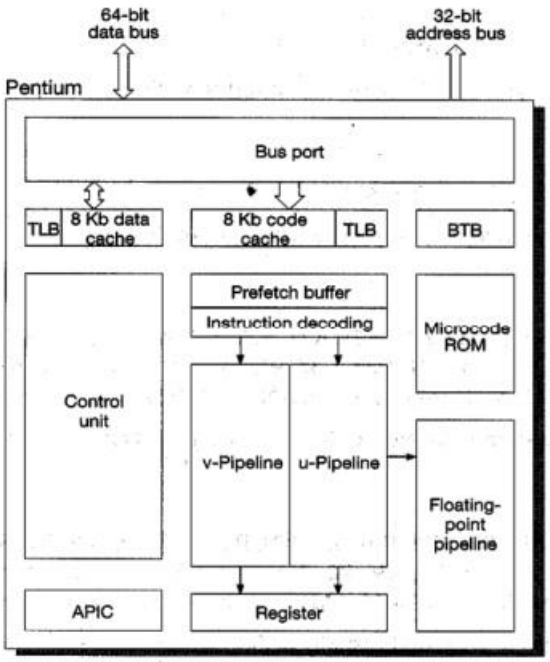
*Per **processore superscalare** si intende generalmente un processore in grado di eseguire più di un'istruzione per ciclo di clock*.

Nei processori a singola CPU o core ciò si ottiene con la duplicazioni delle unità funzionali (ALU, etc) e quelle dedicate alle fasi di fetch e decodifica delle istruzioni. Nella pratica si dotano i processori di più pipeline (un processore superscalare ha almeno 2 pipeline) che operano in parallelo con meccanismi di sincronizzazione tra le stesse per garantire la coerenza dei dati e la sequenza iniziale delle istruzioni.

Accanto a ciò, nei processori a più core si ha un intrinseco parallelismo di esecuzione, dovuto alla pluralità dei core, ognuno dei quali può essere superscalare (cioè avere più pipeline).

I processori della famiglia x86 vanno dalle 2 alle 7 pipeline.

Nella figura successiva è mostrata l'architettura di un core superscalare con 2 pipeline semi simmetriche e nel quale la cache dati e la cache istruzioni sono separate.



Osservando la figura è possibile notare come, all'atto pratico, è come se ci fossero 3 pipeline parallele: 2 per istruzioni intere, 1 per istruzioni FP.

Nel momento in cui la u-Pipeline si accorge che l'istruzione che sta processando è di tipo FP, la inoltra verso la pipeline FP. Il numero di stadi di una pipeline FP è maggiore rispetto alle pipeline intere, in quanto le istruzioni FP hanno bisogno di un numero maggiore di stadi (essendo più complesse); per questo motivo le istruzioni FP hanno un tempo di esecuzione maggiore rispetto a quelle intere.

Processore superscalare: Riepilogo

- Si hanno 3 unità esecutive: due intere (pipeline a 5 stadi) e una floating point (pipeline a 8 stadi);
- Situazioni critiche che possono ridurre fortemente le prestazioni sono:
    - Incompatibilità tra istruzioni su dati e/o operazioni. Esempio di incompatibilità:

$$MOV\ EAX,\ mem \text{ con } ADD\ AX,BX$$

    - Situazioni di stallo per accesso a memoria lenta e/o I/O;
    - Salti condizionati (branch prediction).

Valutazioni prestazioni

Nel caso di *P* pipeline operanti in parallelo:

$$MIPS = \frac{1}{\tau_{max}} \times P = P \times \frac{frequenza\ di\ clock\ CPU}{m}$$

Essendo *m* il numero di periodi di clock richiesti per eseguire le operazioni in uno stadio e *P* il numero di pipeline operanti in parallelo.

Ad esempio:

- $Clock\ processore = 1\ GHz$;
- $m = 1$;
- $pipeline\ parallele = 5$;

Otteniamo che tale processore opera a:

$$MIPS = \frac{1}{1\ GHz} \times 5 = 5 \times 1000 \frac{istruzioni}{s}$$

Esempio di esecuzione di una istruzione in pipeline

Supponiamo di prendere in considerazione l'istruzione:

$$ADD\ AX, [BX]$$

Tale istruzione preleva il contenuto della memoria presente nella locazione *(DS: BX)* e lo somma al contenuto del registro *AX*, scrivendo il risultato ancora nel registro *AX*:

$$AX \leftarrow AX + [\ DS: BX\ ]$$

Si fanno le seguenti ipotesi:

- Pipeline a 5 stadi eseguiti in un colpo di clock.
- $Clock\ CPU\ =\ 1\ GHz \rightarrow Tempo\ di\ percorrenza\ di\ ogni\ stadio\ =\ \frac{1}{1\ GHz} = 1\ ns$
- $Tempo\ di\ accesso\ alla\ L1 =\ 1\ ns$
- $Tempo\ di\ accesso\ alla\ DRAM =\ 60\ ns$

Vediamo cosa succede nella pipeline:

1. **F –** In questa fase viene prelevata l'istruzione da eseguire.
    a. Se la coda di prefetch **non è vuota**, non si generano stalli.
    b. Se la coda di prefetch **è vuota** ed è necessario leggere l'istruzione dalla **cache L1** si dovrà stallare per un numero di colpi di clock sufficienti a leggere il dato dalla cache. Essendo il tempo di accesso alla L1 pari ad 1 ns, è sufficiente un colpo di clock per ottenere l'istruzione: **1 stallo**.
    c. Se la coda di prefetch **è vuota** ed è necessario leggere l'istruzione dalla **DRAM** si dovrà stallare per un numero di colpi di clock sufficienti a leggere il dato dalla cache. Essendo il tempo di accesso alla DRAM pari a 60 ns, sono necessari 60 colpi di clock per ottenere l'istruzione: **60 stalli**.

2. **D1 –** In questa fase viene decodificata l'istruzione e ne viene verificata la realizzabilità. Nel caso in cui l'istruzione non fosse realizzabile (Es. la zona di memoria che si vuole leggere non è disponibile) verrebbe eliminata dalla pipeline; ad ogni modo, non si verificano stalli.

3. **D2 –** In questa fase vengono calcolati gli indirizzi degli operandi e vengono prelevati gli operandi stessi. Ipotizzando di lavorare in modo reale, la paginazione è disabilitata.
   Il secondo operando (*[BX]*) può essere contenuto in:
    a. Cache L1 - facendo un ragionamento analogo a quello fatto per la fase di fetch: **1 stallo**
    b. **DRAM** - facendo un ragionamento analogo a quello fatto per la fase di fetch: **60 stalli**.

FPU e SSE

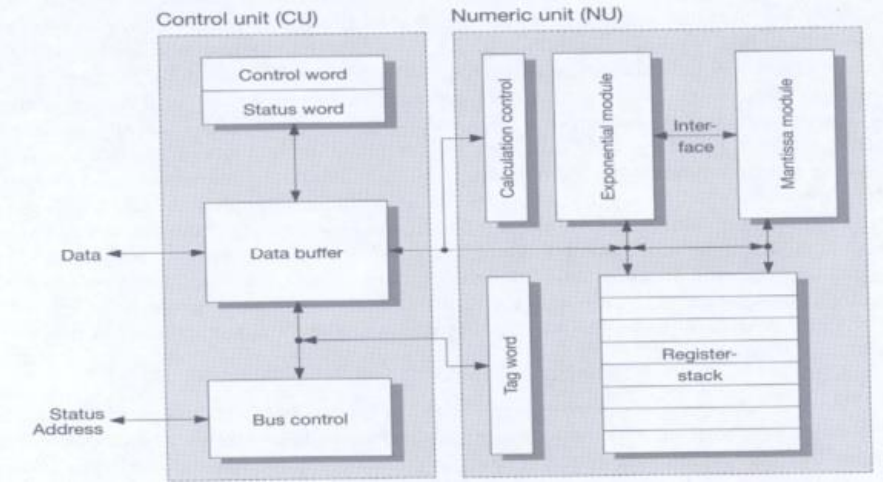
Unità aggiuntive - Registri Speciali

Nell'architettura 80x86 esistono oltre ai citati registri di CPU, registri contenuti in due unita speciali: il coprocessore matematico e l'unita di gestione delle istruzioni di tipo SIMD.

Il coprocessore matematico, **FPU**, fu introdotto fin dall'inizio, mentre le estensioni MMX e SSE vennero introdotte in seguito (Pentium Pro e Pentium IV), anticipate da AMD.

FPU - floating point unit

L'unità floating point è realizzata secondo lo schema mostrato in figura.

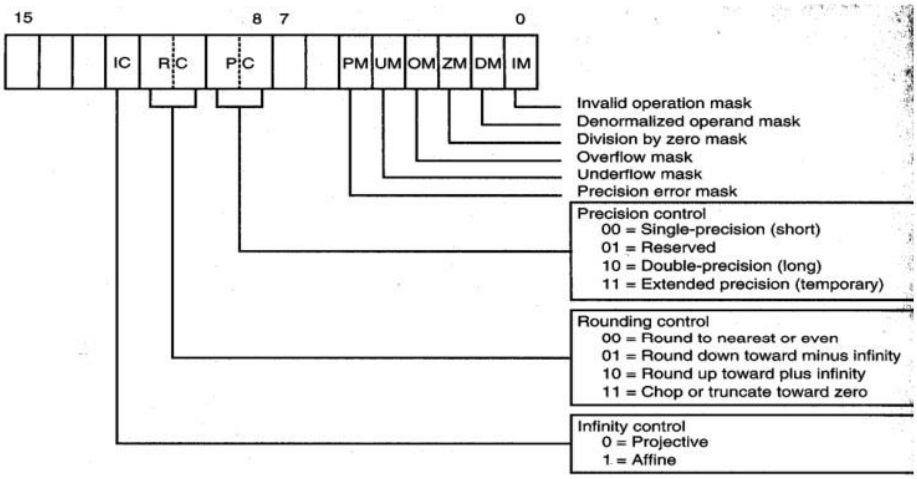


La FPU può essere considerata un vero processore basato su un'**architettura a stack**: ogni operazione viene fatta rispetto al top dello stack (ToS) .

Esegue sia operazioni in fixed point, floating point, sia il calcolo di funzioni trascendentali e trigonometriche, secondo lo standard *IEEE 754*.

FPU: registro controllo

Il registro di controllo permette di definire quale modalità bisogna utilizzare quando si eseguono i calcoli o si interpretano le operazioni.



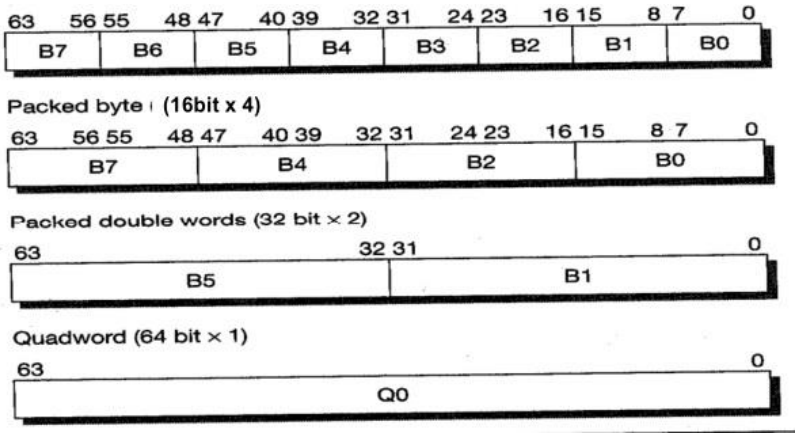
MMX - MultiMedia eXtension or Multiple Math or Matrix Math eXtension

Con **MMX** si fa riferimento ad un set di istruzioni che fanno riferimento a registri aggiuntivi che realizzano una architettura **SIMD** (processori vettoriali). MMX è nato per lavorare su vettori (tipicamente per gestire più pixel insieme nei processori grafici).

I registri interessati sono 8 registri da 64 bit, coincidenti fisicamente con parte dei registri dati della FPU. E' pertanto necessario quando si adoperano tali istruzioni salvare il contenuto dei FPU registers e viceversa.

L'architettura SIMD è fondamentalmente orientata alla grafica (gestione di bit).

I registri da 64 bit (ad oggi 128 bit), detti XMMi, permettono di gestire i seguenti formati:



Ogni registro presenta diversi campi contenenti dati, ognuno caratterizzato da un suo significato semantico.

Esempio

> Somma due vettori di 4 elementi su architettura scalare, con dati espressi su 32 bit in locazioni di memoria adiacenti.

Le operazioni da svolgere sono:

vec_res.x=v1.x+v2.x;
vec_res.y=v1.y+v2.y;
vec_res.z=v1.z+v2.z;
vec_res.w=v1.w + v2.w

Ogni singola operazione richiede 4 istruzioni:

`MOV` ***reg1,v1.x***
`MOV` ***reg2,v2.x***
`ADD` ***reg1,reg2***
`MOV` ***vec_res.x,reg1***

Per un totale di $4 \times 4 = \mathbf{16}$ istruzioni (4 istruzioni per 4 componenti).

> Somma due vettori di 4 elementi su architettura SSE:

`MOVAPS` ***xmm0,address-of-v1***
`ADDPS` ***xmm0,address-of-v2***
`MOVAPS` ***address-of-vec_res,xmm0***

Per un totale di **3** Istruzioni.

Dove gli indirizzi dei vettori puntano al primo elemento del vettore.

Come già accennato, i **registri** nelle architetture SIMD sono identificati con le sigle: `xmm`*I* con *I=0, 1, … , 15.*