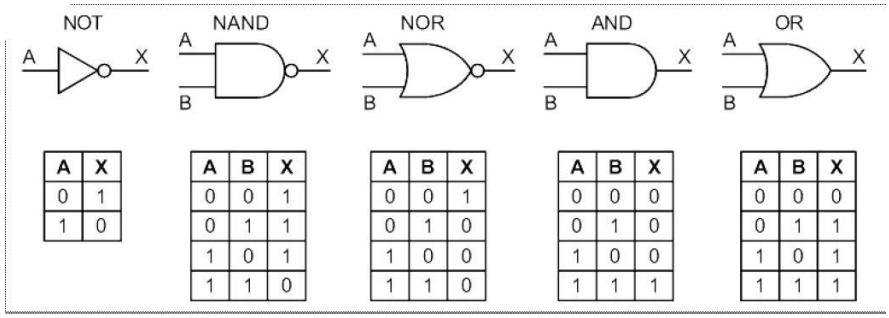
Cenni circuiti, reti combinatorie, reti sequenziali

Porte logiche di base



A	X
0	1
1	0

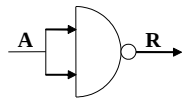
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

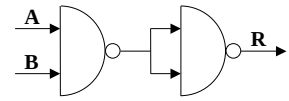
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

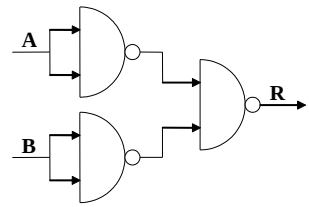
NOT



AND



OR



Quindi NAND o NOR sono complete → circuiti con solo porte NAND o solo porte NOR.

Reti combinatorie

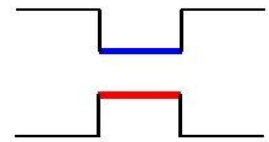
- Rete combinatoria: insieme di porte logiche connesse il cui output in un certo istante è funzione solo dell'input in quell'istante
- N input binari e m output binari
- Ad ogni combinazione di valori di ingresso corrisponde **una ed una sola** combinazione di valori di uscita

Reti combinatorie (segue)

- Vediamo alcuni esempi di circuiti:
  - ✓ I segnali sono discretizzati e di solito assumono solo due stati:

0 / FALSO / [0..1] Volt

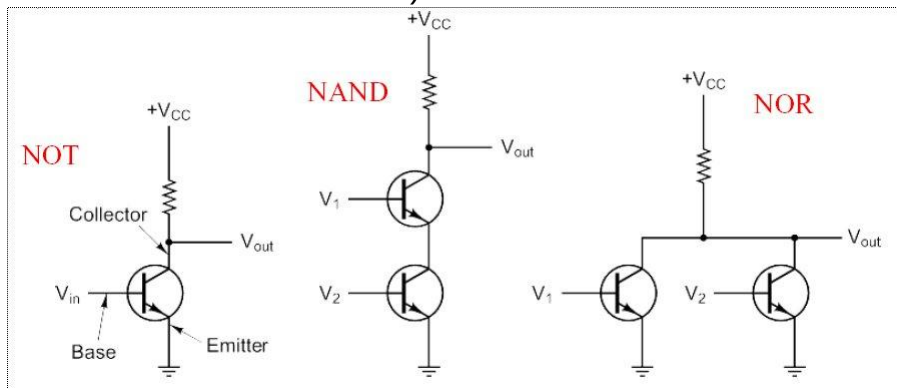
1 / VERO / [2..5] Volt



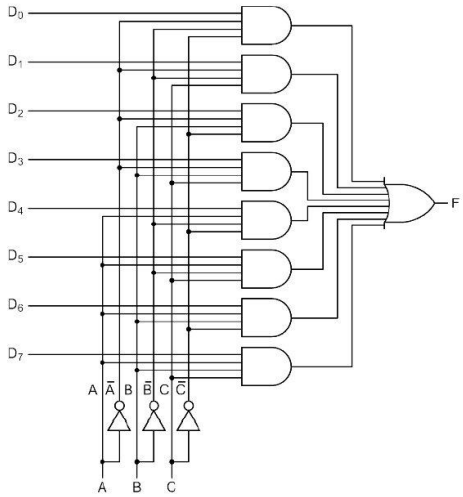
  - ✓ I circuiti più complessi sono realizzati attraverso la combinazione di circuiti semplici (porte logiche)

Reti combinatorie (segue)

- Porte Logiche:
  - ✓ Sono realizzate tramite transistor (sono in pratica interruttori automatici)

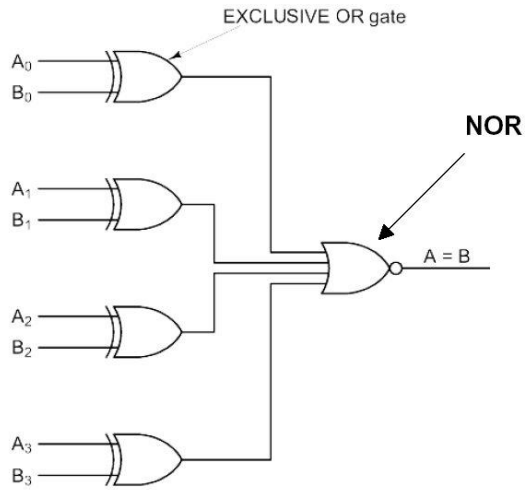


Multiplexer (o selettore) 2^n a 1



- Solo uno degli ingressi viene trasferito all'output
- n ingressi di controllo: indicano l'ingresso da trasferire
  - ✓ 2^n linee di input (D_0 - D_7)
  - ✓ n linee di controllo (A,B,C)
  - ✓ 1 linea di output (F)
- ✓ Per ogni combinazione degli ingressi di controllo, 2^n -1 delle porte AND hanno uscita 0, l'altra fa uscire l'ingresso

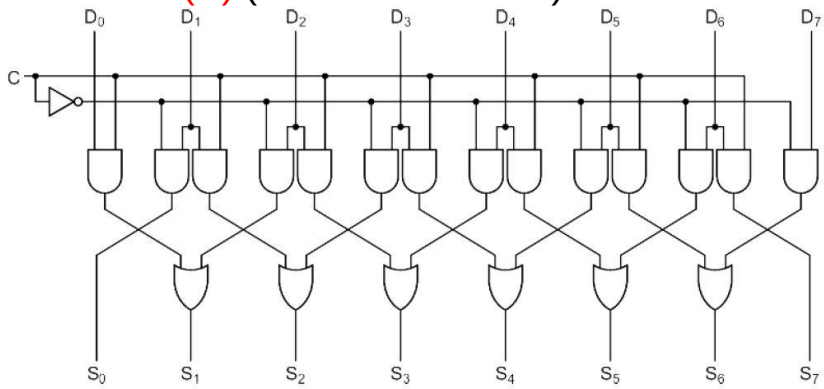
Comparatore a più bit



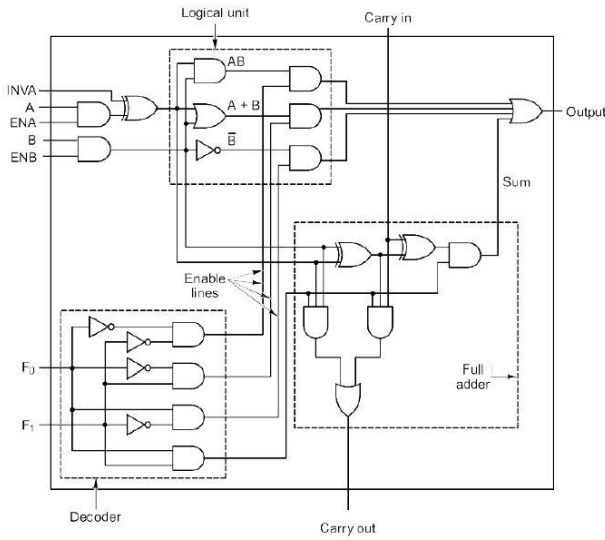
- ✓ Comparatori ad 1 bit vengono collegati tramite una porta NOR
- ✓ L'output vale 1 solo se tutti gli output dei singoli comparatori ad 1 bit valgono 0
- ✓ (Ai=Bi) per ogni i, cioè A=B

Traslatore (shifter)

- ✓ Trasla i bit in ingresso (D) di una posizione, a sinistra o a destra a seconda del valore del bit di controllo (C) (C=1 shift a destra)



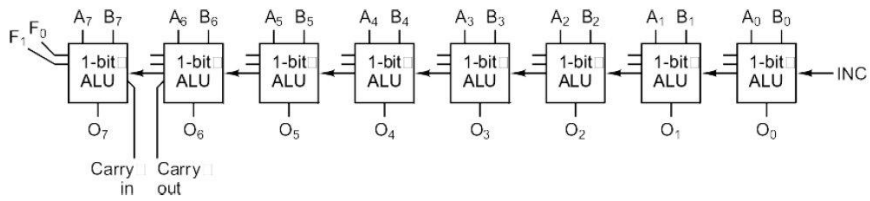
ALU a 1 bit



- ALU ad 1bit che realizza 4 operazioni (selezionate da F_0 e F_1)
- AB, A or B, not(B), A+B
- ENA, ENB: per forzare a 0 gli input A e B
- INVA, INVB: per invertire gli input

ALU a n bit

- Si ottiene concatenando n ALU ad 1 bit
- F_0 e F_1 collegati a tutte le ALU
- Riporto intermedio propagato da una ALU alla successiva
- INC (corrispondente al carry in della ALU “0”) permette di sommare 1 al risultato in caso di addizione



Reti combinatorie

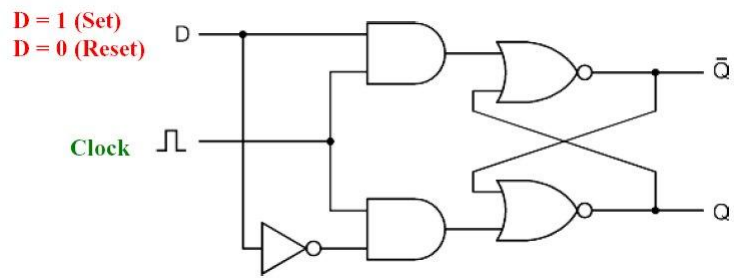
- Utili per implementare la ALU e la connessione tra parti della CPU
- Non sono in grado di memorizzare uno stato, quindi non possono essere usate per implementare la memoria
- Per questo servono le reti sequenziali
  - L’output dipende non solo dall’input corrente, ma anche dalla storia passata degli input

Flip flop

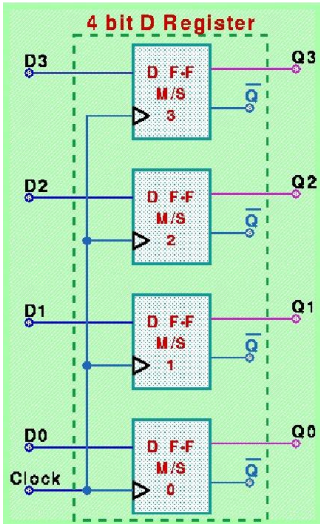
- Forma più semplice di una rete sequenziale
- Tanti tipi, ma due proprietà per tutti:
  - Bistabili:
    - Possono trovarsi in uno di due stati diversi
    - In assenza di input, rimangono nello stato in cui sono
    - Memoria per un bit
  - Due output
    - Uno è sempre il complemento dell'altro

Flip flop D

- Un solo input (D)
- Usa segnale di clock per stabilizzare l'output (sincronizzazione)
- Quando clock =0, gli output dei due AND sono 0 (stato stabile)
- Quando clock=1, gli input sono uno l'opposto dell'altro ➔ Q=D



Registro di tipo D



- è il circuito sincrono più semplice che realizza un registro
- Memorizzazione (store): dati presentati in ingresso e clock da 0 a 1 (uscita riproduce ingresso)
- Mantenimento (hold): clock da 1 a 0 (poi costante); l'uscita rimane invariata indipendentemente dal valore degli ingressi

(Super)Cenni di Microprogrammazione

Controllo

- La Parte (o Unità) Controllo (PC) della CPU si fa carico di realizzare il flusso di controllo appropriato per ogni istruzione tramite l'invio di opportuni segnali di controllo alla Parte Operativa

- **Requisiti funzionali** (cioè le funzioni che la PC deve eseguire):
  - Definire gli elementi di base del processore
  - Definire le micro-operazioni che il processore esegue
  - Determinare le funzioni che la PC deve effettuare per l'esecuzione delle micro-operazioni

Elementi Base

- Come visto in precedenza gli elementi di base sono:
  - ALU
  - Registri
  - Bus dati interno
  - Bus dati esterno
  - Unità di controllo

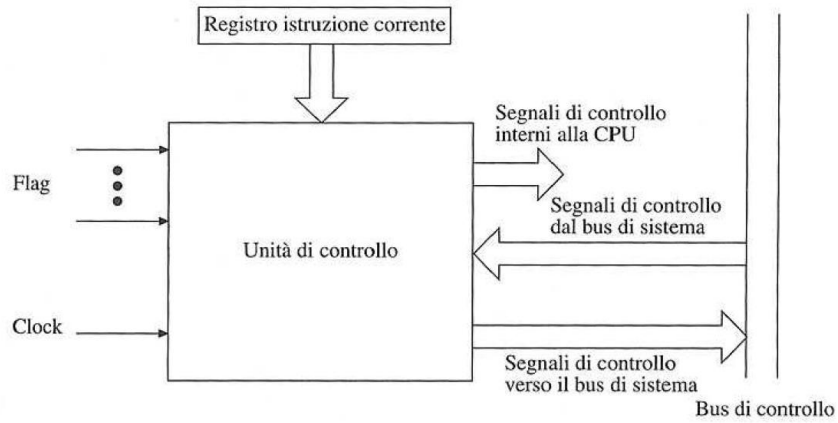
Tipologie di micro-istruzioni

- Trasferimento dati da un registro all'altro
- Trasferimento dati da un registro ad un'interfaccia esterna
- Trasferimento dati da un'interfaccia esterna ad un registro
- Esecuzione di una operazione aritmetica o logica, che utilizzi i registri come input e output

Funzioni della PC

- Quindi i compiti base della PC sono:
  - **Serializzazione**: determina la "giusta" sequenza di micro-operazioni da eseguire in funzione del codice operativo dell'istruzione
  - **Esecuzione**: provoca l'esecuzione di micro-operazioni
- La realizzazione di questi compiti base passa attraverso la generazioni di opportuni *segnali di controllo*

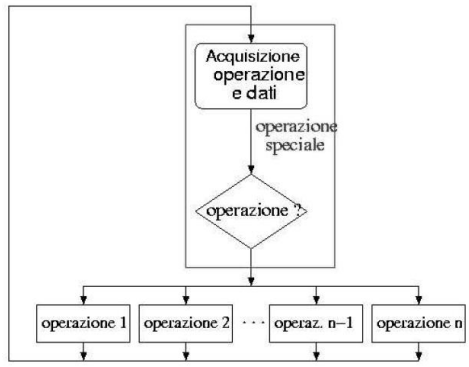
Segnali di Controllo



Realizzazione della PC

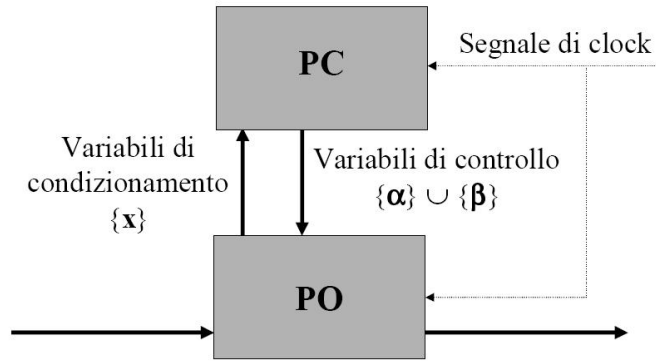
- Ci sono due alternative:
  - **Cablata**:
    - si realizza direttamente tramite circuiti digitali (livello di astrazione 0);
    - soluzione tipica di architetture RISC;
  - **Microprogrammata** (la trattiamo di seguito):
    - si realizza tramite microprogrammazione (livello di astrazione 1);
    - soluzione tipica di architetture CISC;
    - permette una maggior flessibilità in fase di progettazione: rende facile modificare le sequenze di micro-operazioni

Livello Firmware

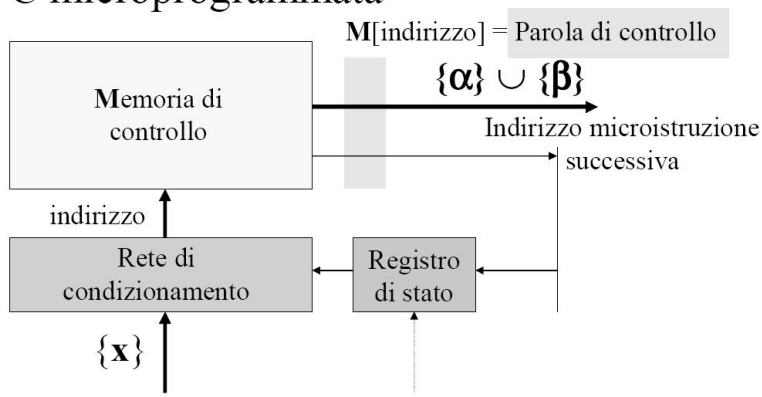


- Il *μprogramma* di una unità riunisce i frammenti di programma delle diverse operazioni (esterne e speciale)
- Il *μprogramma* ha una struttura ciclica in cui si alterna l'esecuzione della operazione speciale con l'esecuzione della operazione esterna il cui codice e dati da eleborare sono stati acquisiti dalla operazione speciale

Livello Firmware (segue)



- PC microprogrammata



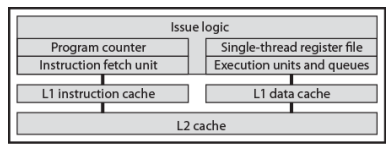
- {α} ∪ {β} designa la microoperazione richiesta

Calcolatori Multicore

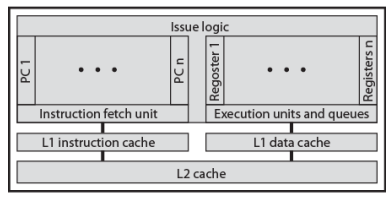
Prestazioni hardware

- I microprocessori hanno visto una crescita esponenziale delle prestazioni
  - Miglioramento della organizzazione
  - Incremento della frequenza di clock
- Crescita del parallelismo
  - Pipeline
  - Pipeline parallele (superscalari)
  - superscalari + replicazione banco registri → Multithreading simultaneo (SMT)
- Problemi
  - Maggiore complessità richiede logica più complessa
  - Aumento dell'area del chip per supportare il parallelismo
    - Più difficile da progettare, realizzare e verificare (debug)

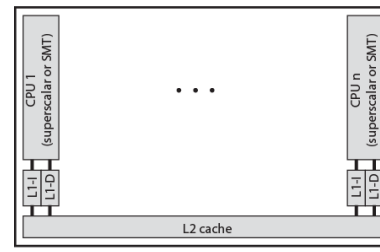
Organizzazioni alternative del chip



(a) Superscalar

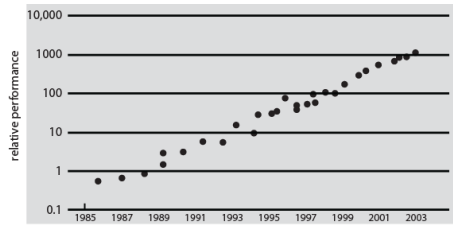


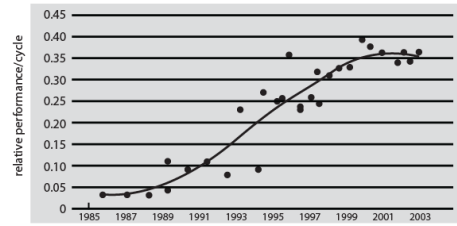
(b) Simultaneous multithreading

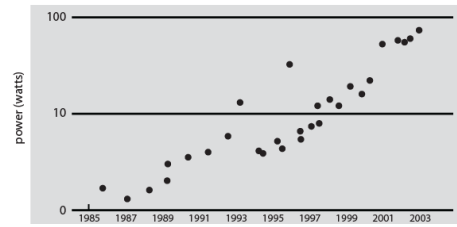


(c) Multicore

Tendenze Hardware Intel



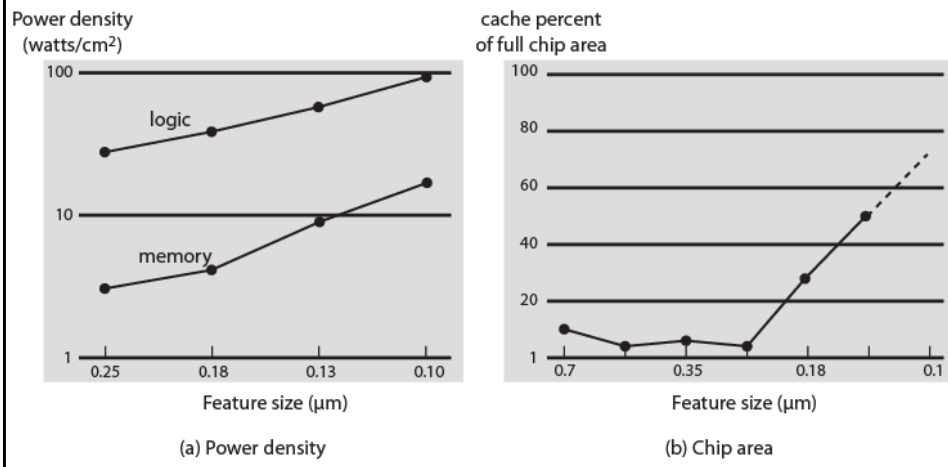




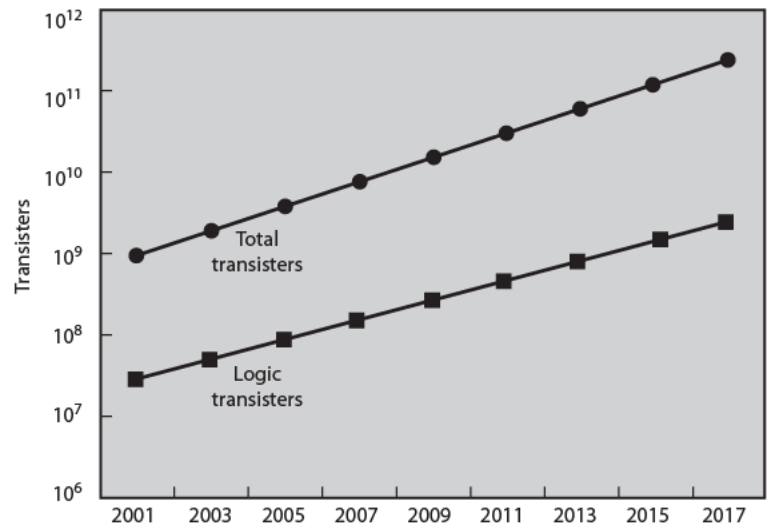
Incremento in complessità

- La potenza cresce esponenzialmente con la densità del chip e la frequenza del clock
  - Rimedio: usare più spazio per la cache
    - Meno densa
    - Richiede molta meno potenza (ordini di magnitudine)
- Nel 2015
  - 100 miliardi di transistor in 300mm^2 sul “die” (chip)
    - Cache di 100MB
    - 1 miliardo di transistor per la logica
- Regola di Pollack:
  - Le prestazioni sono all’incirca proporzionali alla radice quadrata dell’incremento in complessità
    - Il raddoppio in complessità restituisce il 40% in più di prestazione
- Architetture multicore hanno il potenziale per ottenere un miglioramento quasi lineare
- Improbabile che un core possa utilizzare efficacemente tutta la memoria cache

Considerazioni sulla potenza e sulla memoria



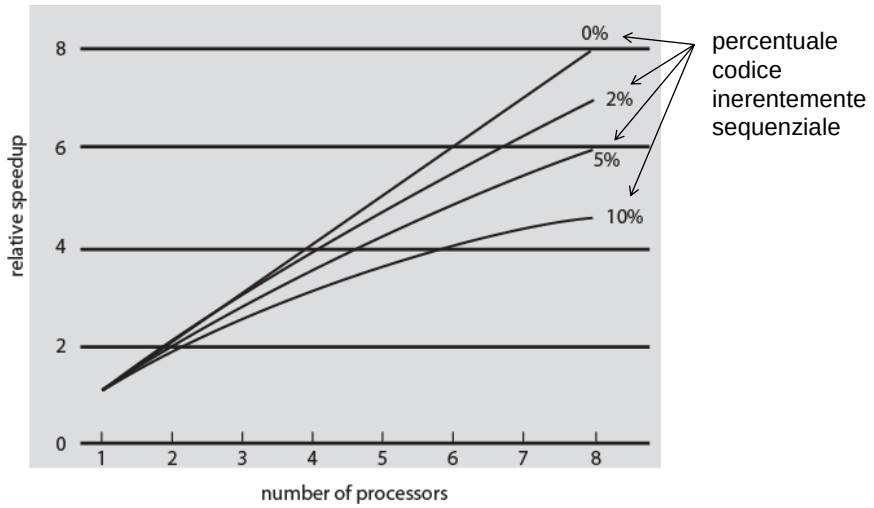
Utilizzo dei Transistor



Prestazioni del Software

- I vantaggi prestazionali dipendono dallo sfruttamento efficace delle risorse parallele
- Anche una piccola quantità di codice seriale ha un impatto significativo sulle prestazioni
  - Il 10% di codice intrinsecamente seriale eseguito su un sistema a 8 processori dà un incremento di prestazioni di solo 4,7 volte
- Overhead dovuto alla comunicazione, distribuzione del lavoro e mantenimento della coerenza della cache
- Alcune applicazioni effettivamente sfruttano i processori multicore

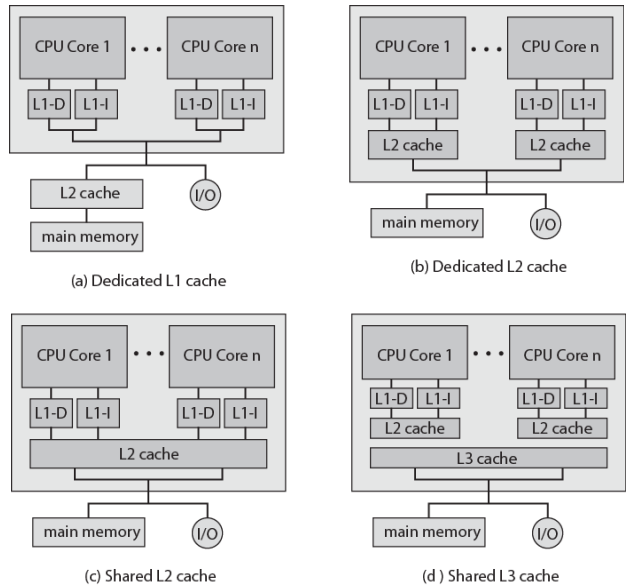
Prestazioni del Software



Organizzazione Multicore

- Numero di core per chip
- Numero di livelli di cache per chip
- Quantità di cache condivisa

Possibili alternative

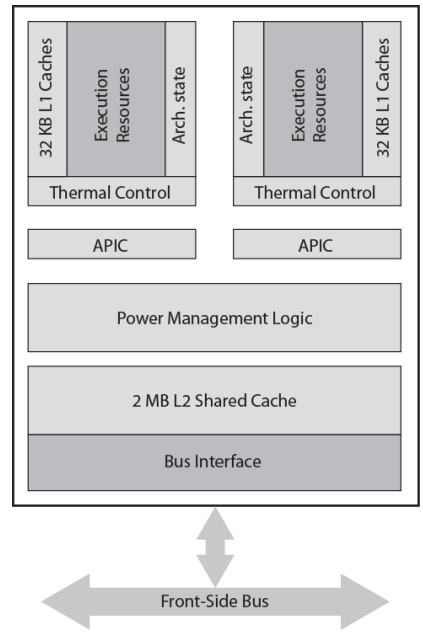


Vantaggi di una Cache L2 condivisa

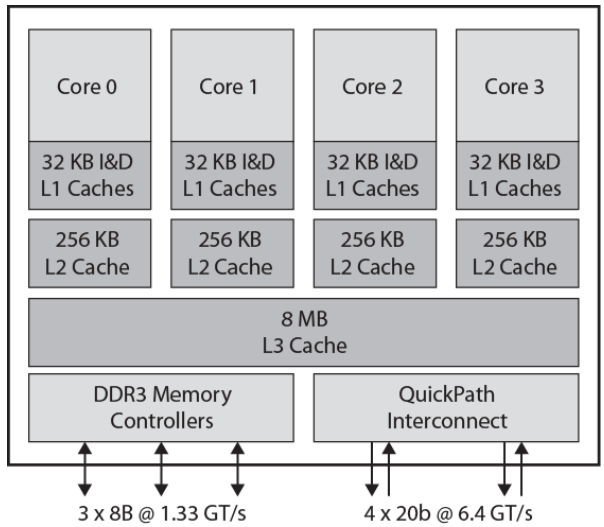
- Riduzione (accidentale) del numero di miss totali
- Dati condivisi da più core non sono replicati a livello di cache (a livello 2, ma possibile replicazione a livello 1)
- Con appropriati algoritmi di sostituzione dei blocchi, la quantità di cache dedicata ad ogni core è dinamica
  - Thread con minore località possono utilizzare più spazio di cache
- Comunicazione fra processi (anche in esecuzione su core diversi) facilitata dall'utilizzo della memoria condivisa
- Problema della coerenza della cache confinata al L1
- Cache L2 dedicate danno però un accesso alla memoria più rapido
  - Migliori prestazioni per thread con forte località
- Anche una cache L3 condivisa può migliorare le prestazioni

Intel Core Duo

- 2006
- core superscalari
- cache L2 condivisa
- cache L1 dedicata
- unità di controllo termico
- controllori di interruzioni programmabili (APIC)
- logica di controllo della potenza
- Interfaccia bus



Intel Core i7



- Novembre 2008
- core SMT
- cache L3 condivisa
- cache L1, L2 dedicate
- controllore memorie DDR3 sul chip
- Logica di connessione con controllo di coerenza della cache molto efficiente e veloce (banda totale di 25.6GB/s)

Prestazioni dei processori multicore