Calcolatori Elettronici

Parte III: L'organizzazione generale del calcolatore

Prof. Riccardo Torlone

Università Roma Tre

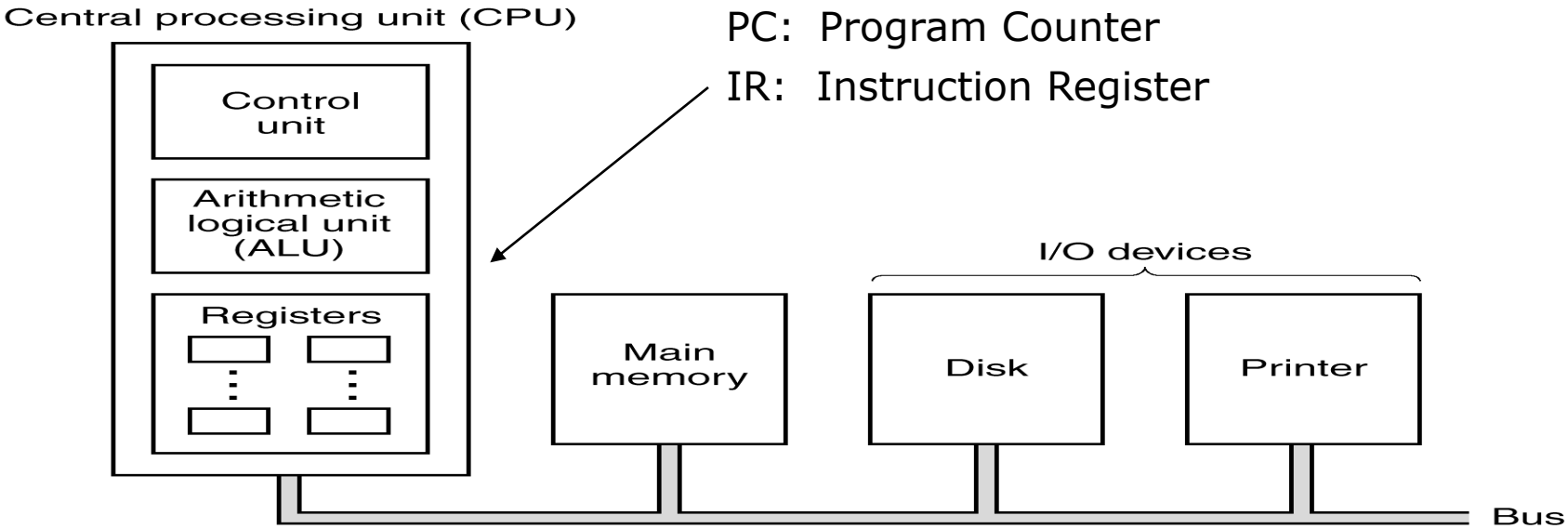
Terminologia di base

- **Calcolatore elettronico**: macchina fatta di dispositivi elettronici che può risolvere problemi eseguendo istruzioni fornitegli
- **Programma**: sequenza di istruzioni in un linguaggio
- **Linguaggio macchina**: eseguibile direttamente da un calcolatore (binario)

Con dispositivi elettronici si possono eseguire direttamente solo un numero limitato di istruzioni semplici (costi)

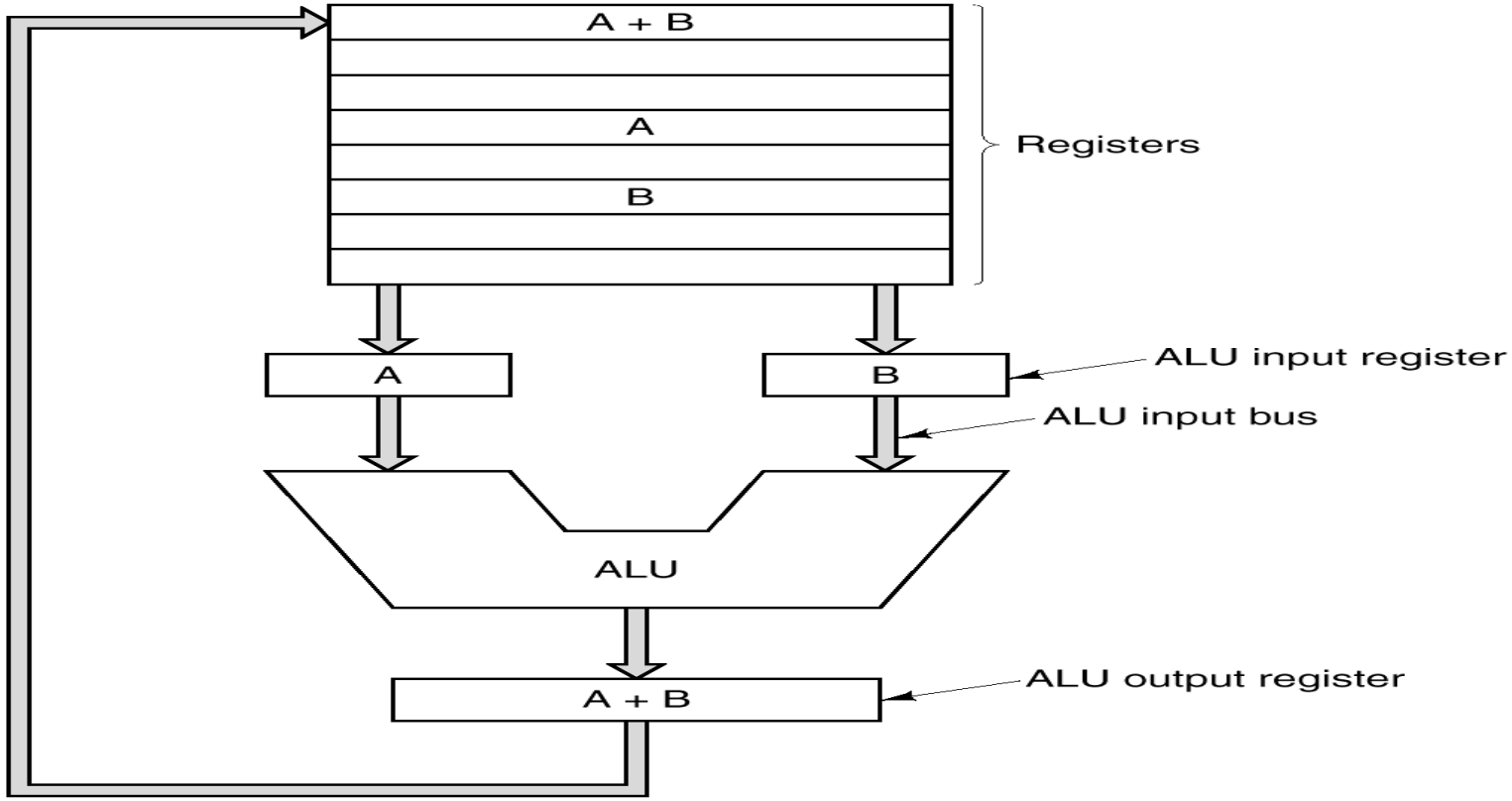
I linguaggi macchina non sono adatti per le persone

Struttura del computer



- La memoria contiene sia i dati che le istruzioni
- Il contenuto dei registri può essere scambiato con la memoria e l'I/O
- Le istruzioni trasferiscono i dati e modificano il contenuto dei registri
- Registri particolari:
  - PC: indirizza la prossima istruzione
  - IR: contiene l'istruzione corrente

Struttura della CPU



- Esecuzione di operazioni aritmetiche e logiche sui dati contenuti nei registri
- Spostamento di dati fra registri e fra registri e memoria
- Ciclo elementare: due operandi sono inviati alla ALU e il risultato e messo in un registro

Il ciclo Fetch-Decode-Execute

L'esecuzione di ciascuna istruzione nella CPU richiede i seguenti passi:

1. Carica l'istruzione da memoria in IR (Instruction Register) (**Fetch**)
2. Incrementa PC (Program Counter)
3. Decodifica l'istruzione (**Decode**)
4. Se l'istruzione usa un dato in memoria calcolane l'indirizzo
5. Carica l'operando in un registro
6. Esegui l'istruzione (**Execute**)
7. Torna al passo **1.** Per l'esecuzione dell'istruzione successiva

Accessi alla memoria sono effettuati sempre al passo 1, e non sempre ai passi 4 e 5

Esecuzione e Interpretazione

Esecuzione diretta

- Le istruzioni possono venire eseguite direttamente dai circuiti hardware
- Approccio molto complesso:
  - Repertorio di istruzioni limitato
  - Progettazione dell'HW complessa
  - Esecuzione molto efficiente

Interpretazione

- L'hardware può eseguire solo alcune operazioni elementari molto semplici dette microistruzioni
- Ciascuna istruzione è scomposta in una successione di microistruzioni poi eseguite dall'hardware
- Vantaggi:
  - Repertorio di istruzioni esteso
  - HW più compatto
  - Flessibilità di progetto

La Microprogrammazione

L'HW può eseguire *microistruzioni*:

- Trasferimenti tra registri
- Trasferimenti da e per la memoria
- Operazioni della ALU su registri

Ciascuna istruzione viene scomposta in una *sequenza di microistruzioni*

L'unità di controllo della CPU esegue un *microprogramma* per effettuare l'*interpretazione* delle istruzioni macchina

Il microprogramma è contenuto in una memoria ROM sul chip del processore

Vantaggi:

- Disegno strutturato
- Semplice correggere errori
- Facile aggiungere nuove istruzioni

CISC e RISC

Architetture **RISC** (*Reduced Instruction Set Computer*):

- Esecuzione diretta
- Repertorio ristretto (alcune decine)
- Istruzioni prevalentemente su registri
- *Una istruzione per ciclo di macchina (del data path)*

Architetture **CISC** (*Complex Instruction Set Computer*)

- Interpretazione tramite microprogramma
- Repertorio esteso (alcune centinaia)
- Istruzioni anche su memoria
- *Molti cicli di macchina per istruzione*

Esempi:

- PowerPC, SPARC, MIPS, ARM: RISC
- VAX (DEC), Pentium II/III/IV/i7 (Intel), AMD: CISC

All'inizio degli anni '80 i progettisti di sistemi veloci riconsiderano l'approccio dell'*esecuzione diretta*

Principi progettuali dei computer moderni

- Eseguire tutte le istruzioni dall'hardware
- Massimizzare la velocità con la quale le istruzioni sono eseguite misurata in MIPS (Millions of Instr. per Second) o XFLOPS (M/G/T floating point oper. per Second)
- Semplificare la decodifica delle istruzioni: formati molto regolari
- Limitare i riferimenti alla memoria (solo LOAD e STORE)
- Ampliare il numero di registri

N.B. Questi principi sono tipici della filosofia RISC ma anche le architetture CISC vi si adeguano, almeno in parte

Introduzione del parallelismo

Vari Tipi di Parallelismo

Il parallelismo è ormai l'unica strada per aumentare le prestazioni
Limite di un'esecuzione sequenziale: velocità della luce (30 cm in 1 nsec)

Due tipi di parallelismo:

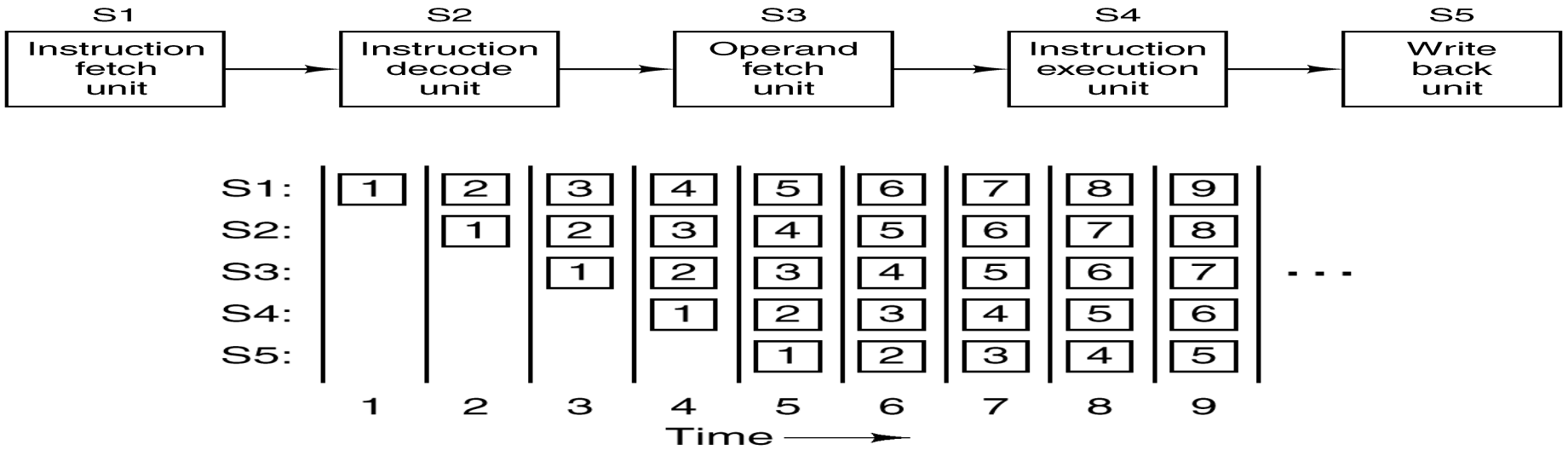
A) a livello di istruzioni

- Diverse istruzioni eseguite insieme
- Diverse fasi della stessa istruzione eseguite insieme

B) a livello di processori

- Molti processori lavorano insieme allo stesso problema
- Fattori di parallelismo molto elevati
- Diversi tipi di interconnessione e di cooperazione (più o meno stretta)

Parallelismo a livello di istruzioni: Pipelining



- Ciascuna istruzione è divisa in fasi
- L'esecuzione avviene in una *pipeline* a più stadi
- Più istruzioni in esecuzione contemporanea
- Una istruzione completata per ogni ciclo

N.B. Si guadagna un fattore pari al numero di stadi della pipeline

Caratteristiche di una pipeline

Una pipeline consente un compromesso tra:

- Latenza: tempo per eseguire una istruzione
- Ampiezza di banda: numero di istruzioni completate per unità di tempo misurata in MIPS (milioni di istruzioni al secondo) - oggi in GFLOPS o TFLOPS (10^9 o 10^{12} istruzioni in virgola mobile al secondo)

Con:

- Velocità di clock = T nsec (periodo del segnale di clock)
- Numero di stadi = n

Abbiamo:

- **Latenza** = nT
- **Ampiezza di banda** = 1 istr. ogni T nsec, ovvero: $10^9/T$ istr. ogni sec., ovvero: $1000/T$ MIPS

Architetture Superscalari

Architetture nelle quali si avviano più istruzioni (4-6) insieme

Si aumenta il parallelismo avendo più di una pipeline nel microprocessore

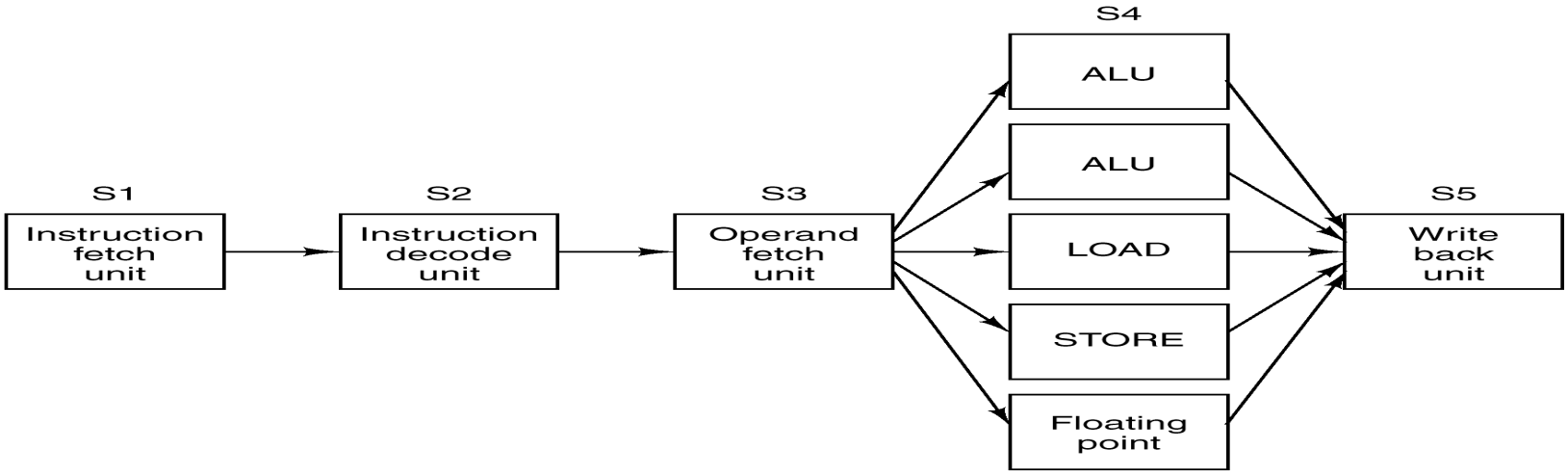
Le pipeline possono essere specializzate:

- Una versione dell'i7 ha diverse pipeline a più stadi
- Può eseguire fino a 6 micro-istruzioni in parallelo

Problema: compatibilità dell'esecuzione parallela

- Indipendenza tra le istruzioni
- Ciascuna istruzione non deve utilizzare i risultati dell'altra

Unità Funzionali Multiple



- Variante: solo lo stadio più lento della pipeline (che condiziona la velocità) viene parallelizzato
- La CPU contiene al suo interno diverse unità funzionali indipendenti
- Architettura adottata nei processori Intel Core

Parallelismo a livello di processori

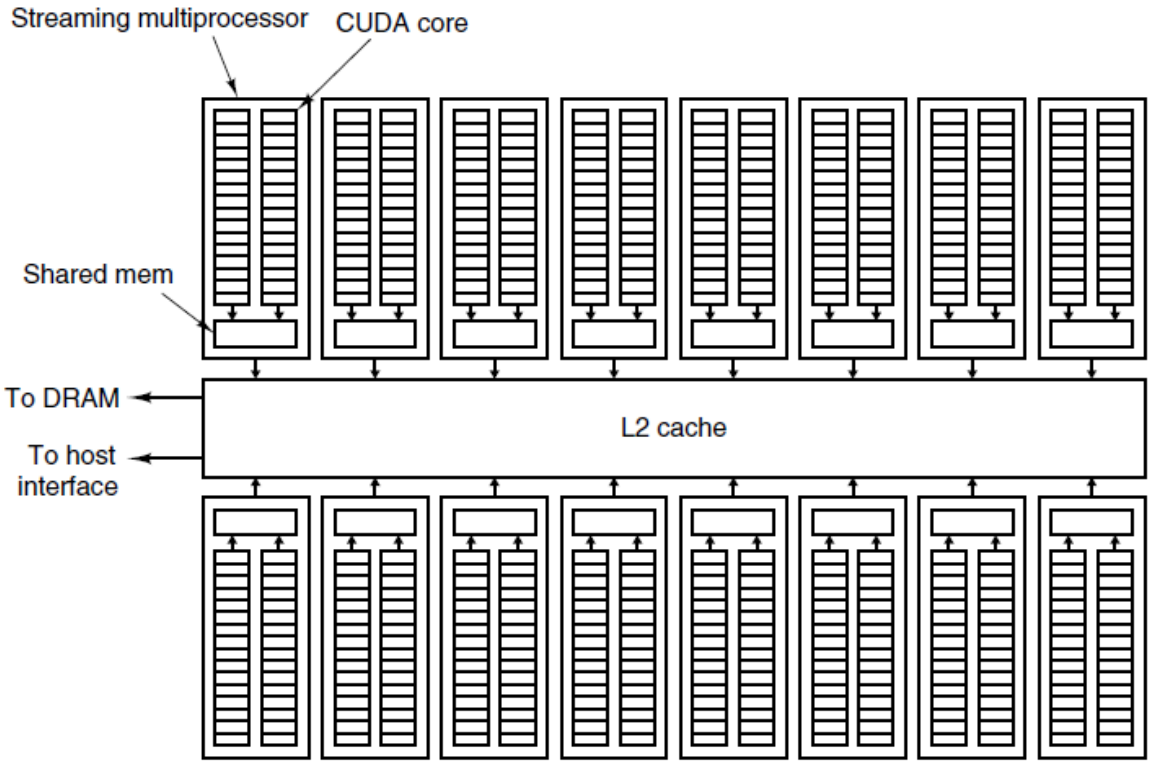
- Miglioramento delle prestazioni con parallelismo a livello di istruzioni: 5-10 volte
- Per migliorare ancora: CPU multiple
- Approcci:
  - Data parallelism (SIMD)
    - Processori matriciali
    - Processori vettoriali
    - GPU
  - Task parallelism (MIMD)
    - Multiprocessori
    - Multicore
    - Multicomputer

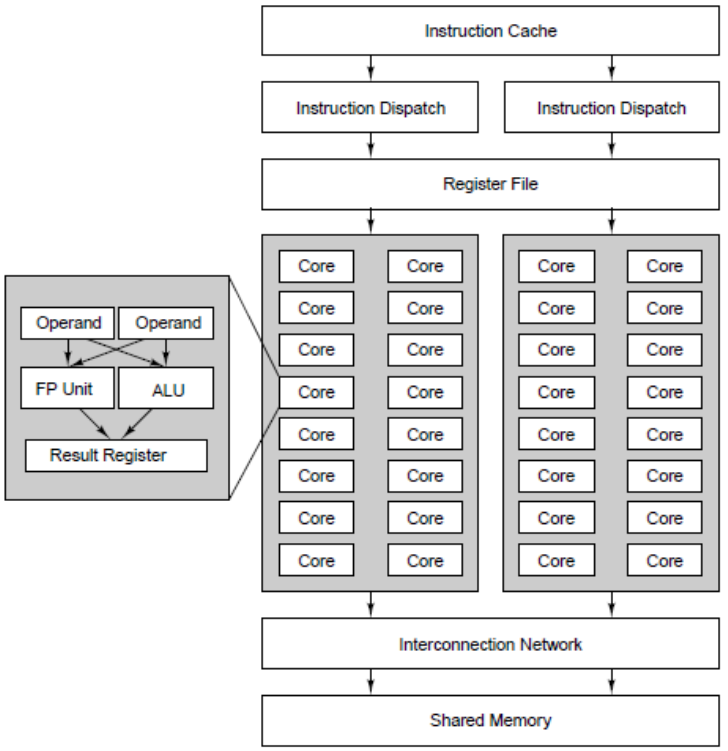
```
[ || (int i : 100) array[i]++; ]
```

```
[ a++; || b+c; ]
```

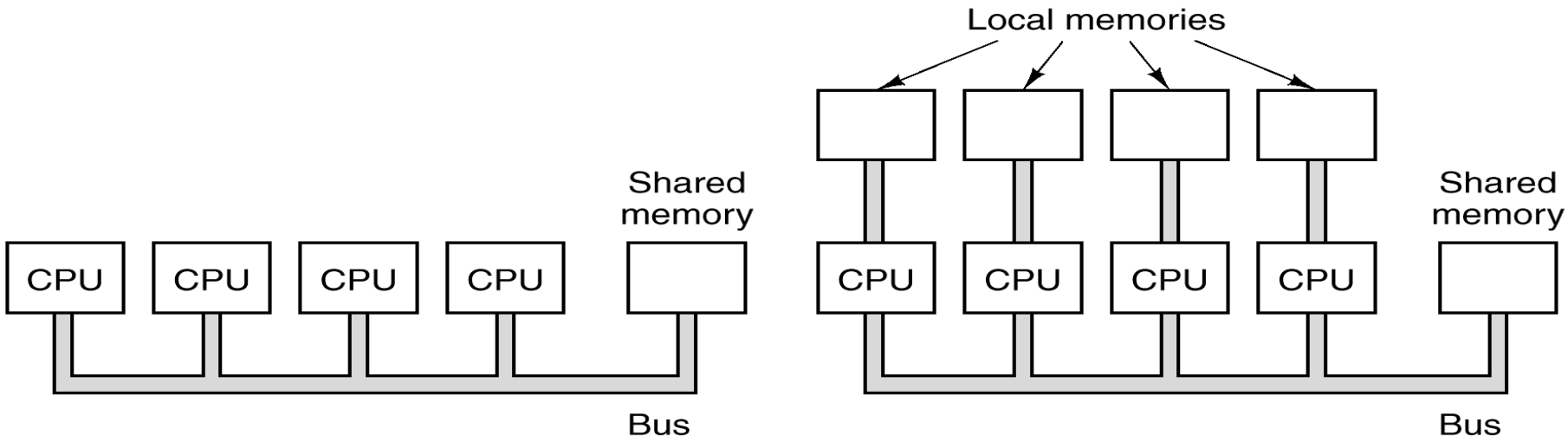
GPU

- Operazioni comuni su pixel, vertici, archi, figure
- Es.: Nvidia Fermi GPU (2009)
  - 16 processori stream SIMD
  - Ogni processore ha 32 core
  - Fino a 512 operazioni per ciclo di clock





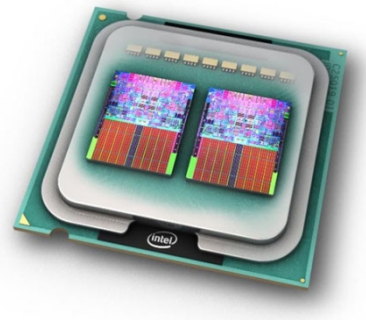
Multiprocessori



- Le CPU lavorano indipendentemente
  - Shared memory: il bus può divenire collo di bottiglia
  - Private memory: contiene il codice e parte dei dati
- Scambio dati tramite la shared memory

Architetture multicore

- La CPU è composta da più core, ovvero da più nuclei di processori fisici montati sullo stesso package
- Ogni core:
  - è un processore indipendente
  - può essere dotato di cache autonoma
- Architetture omogenee (core identici) o eterogenee
- Ogni core può essere multiscalare
- Accoppiamento dei core:
  - stretto: shared cache
  - lasco: private cache
- Nascono a partire dal 2003:
  - IBM: PowerPC
  - Intel: Pentium D, Core 2, Core I3-i5-i7
  - AMD: Opteron, Athlon, Phenom

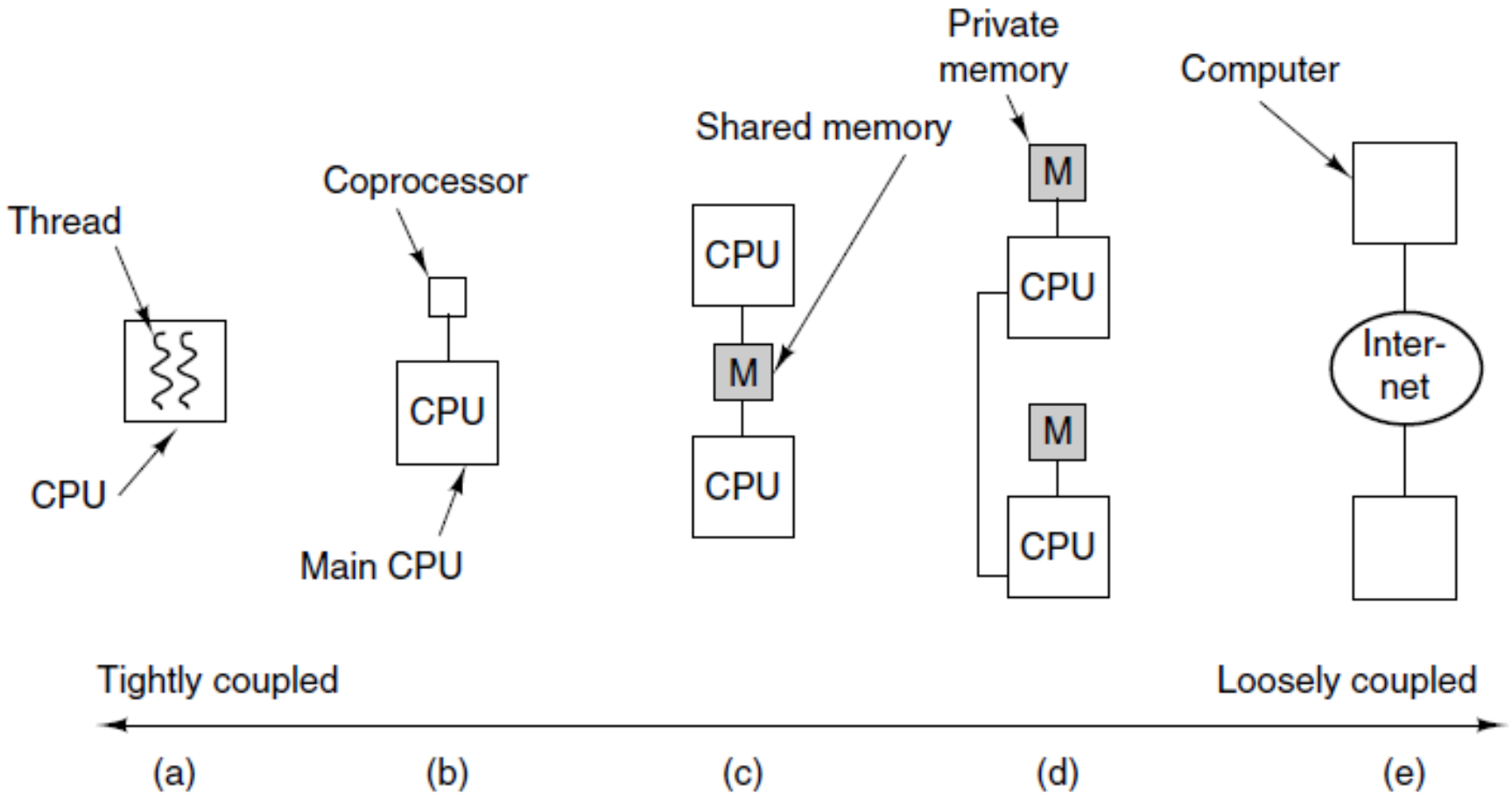


Multicomputer

- I singoli elementi sono normali Workstation o PC
- Comunicazione tramite scambio di messaggi (shared nothing)

MIMD (Multiple Instruction Multiple Data)

Le varie forme di parallelismo



La Memoria Centrale

- Contiene sia i programmi che i dati
- Memorizzazione binaria (bit)
- Cella (o locazione): unità indirizzabile
  - byte: 8 bit (minimo indirizzabile)
  - word: insieme di K byte (K dipende dall'architettura)
- Indirizzo (della cella): tramite il quale la CPU accede al dato nella cella
- Indirizzi binari a m bit: spazio di indirizzamento 2^m celle

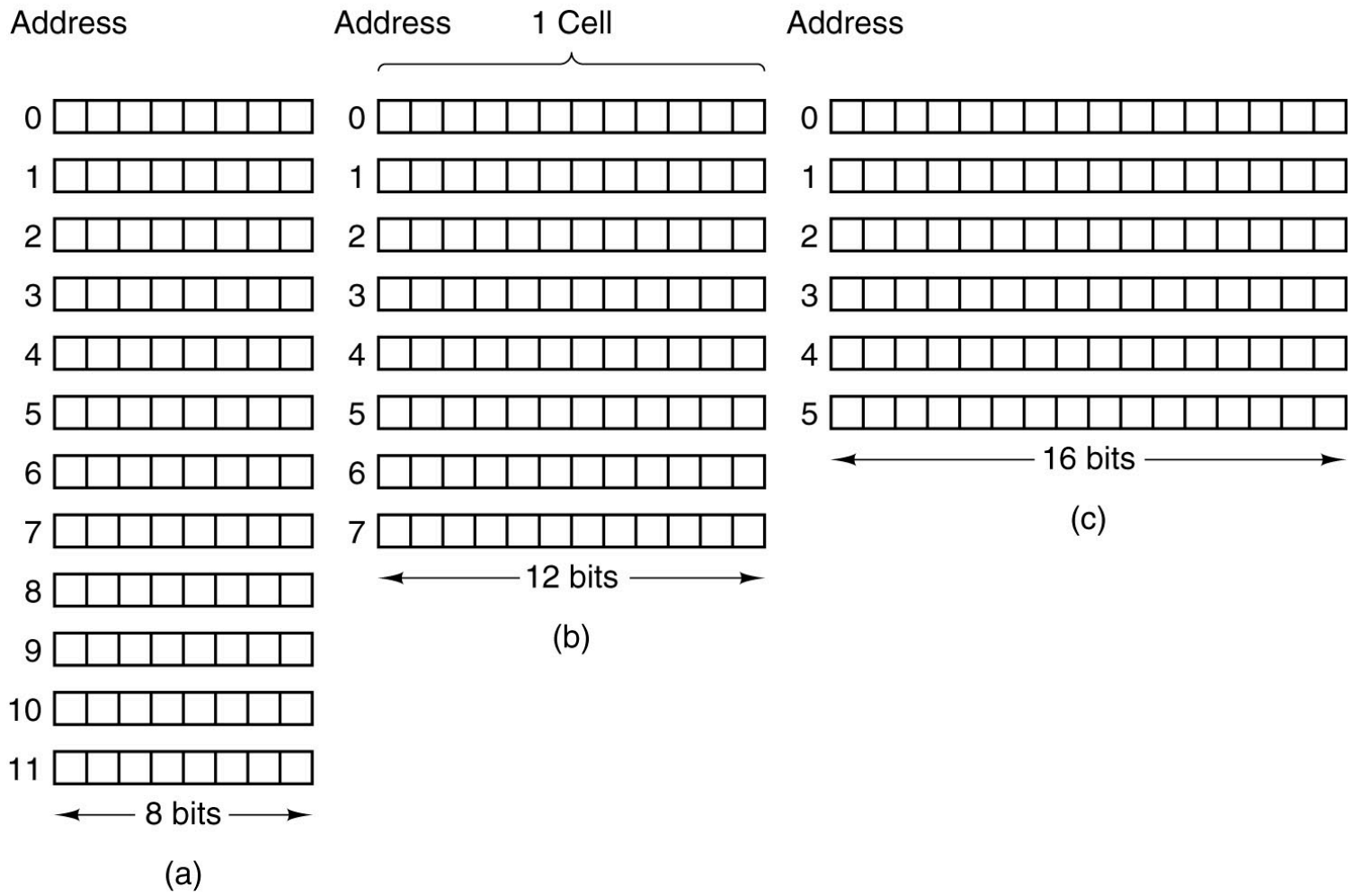
ES Pentium IV

- Architettura a 32 bit
- Registri e ALU a 32 bit
- Word di 4 byte 32 bit
- Indirizzi a 32 bit
- Spazio indirizzabile 2^{32} = 4 GB
  (64GB con opportuni accorgimenti)

Organizzazione della memoria

Diverse possibilità

- Esempio con 96 bit totali:



Dimensione locazioni di memoria

Diverse soluzioni possibili

Computer	Bits/cell
Burroughs B1700	1
IBM PC	8
DEC PDP-8	12
IBM 1130	16
DEC PDP-15	18
XDS 940	24
Electrologica X8	27
XDS Sigma 9	32
Honeywell 6180	36
CDC 3600	48
CDC Cyber	60

Codici a correzione di errore

Tecniche per garantire maggiore affidabilità nella registrazione / trasmissione di informazioni binarie

Recupero degli errori hardware tramite codifiche ridondanti

Codifiche con n = m + r bit

- n bit complessivi codifica
- m bit dati
- r check bit (ridondanti)

Si utilizza solo un sottoinsieme delle codifiche (codifiche valide)

ES **Codice** con n=10, m=2, r=8

0000000000
0000011111
1111100000
1111111111

2^m= 4 codifiche valide (su 2^{10})

Distanza di Hamming

Distanza di Hamming tra due codifiche: numero di bit diversi:

0101 e 1001 sono a distanza 2

Distanza di Hamming di un codice: h = distanza di Hamming minima tra due codifiche valide del codice

ES

0000000000
0000011111
1111100000
1111111111

} Distanza di Hamming del codice h=5

- Per **rilevare** errori su k bit occorre che sia:
  - almeno h = k + 1 ovvero k ≤ h - 1
- Per **correggere** errori su k bit occorre che sia:
  - almeno h = 2k + 1 ovvero k ≤ (h – 1)/2

Codici a correzione di errore (Esempio)

ES

Codice con n=10, m=2, r=8

0000000000
0000011111
1111100000
1111111111

} Distanza di Hamming = 5

h=5=k+1 $\Rightarrow$ E' possibile *rilevare* errori <u>quadrupli</u>

0000011111 $\rightarrow$ 1111011111

1111011111 viene riconosciuto come errato

h=5=2k+1 $\Rightarrow$ E' possibile *correggere* errori <u>doppi</u>

0000011111 $\rightarrow$ 1100011111

1100011111 viene corretto in 0000011111

Rilevazione di errore singolo (controllo di parità)

- Nel caso più semplice si vogliono solo rilevare errori singoli
- Basta aggiungere un solo check bit r=1, n=m+1
- Bit di parità: scelto in modo che il numero complessivo di 1 nella codifica sia sempre pari (o dispari)
- Questo codice ha distanza h=2
- Errore rilevato da circuiti molto semplici
- Le memorie segnalano *parity error* quando un errore si manifesta

ES. 11011010 bit di parità:1 → 110110101 OK

01100101 bit di parità:0 → 011011010 Error

Correzione di errore singolo

- m data bit, r check bit, n bit totali
- 2^m codifiche valide
- n codifiche errate a distanza 1 da ciascuna delle valide
- Ogni codifica valida ne richiede in tutto n+1

ES:

La codifica:

0000

Richiede le codifiche errate:

1000

0100

0010

0001

Correzione di errore singolo

- Se ogni codifica valida ne richiede n+1 deve essere:

$$(n+1)\ 2^m \leq 2^n \quad \text{cioè} \quad (m+r+1) \leq 2^r$$

Word size	Check bits	Total size	Percent overhead
8	4	12	50
16	5	21	31
32	6	38	19
64	7	71	11
128	8	136	6
256	9	265	4
512	10	522	2

N.B. Al crescere di m l'*overhead* scende

Esercizio 1

Riferendosi all'organizzazione generale di un calcolatore, indicare se le seguenti affermazioni sono vere o false.

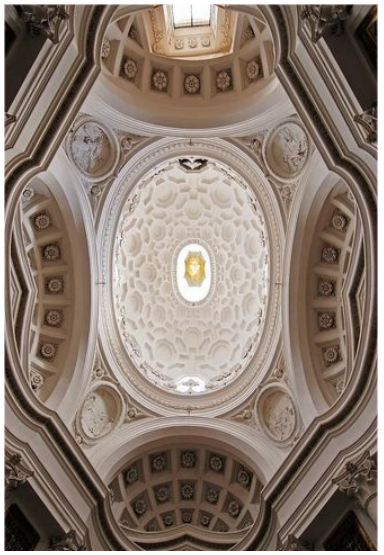
- Nelle architetture RISC le istruzioni macchina vengono tradotte in microistruzioni che vengono poi eseguite dall'hardware. **FALSO**
- Le tecnica del pipeline non è compatibile con una architettura superscalare. **FALSO**
- Una architettura con indirizzi a 16 bit con indirizzamento al byte non può gestire una memoria più grande di 64KB. **VERO**
- In processore con pipeline a 5 stadi e un clock con periodo di 2 nsec una istruzione macchina richiede 10 nsec per essere eseguita. **FALSO**
- Un processore con pipeline a 5 stadi e un clock con periodo di 5 nsec ha un'ampiezza di banda di 200 MIPS. **VERO**
- L'ampiezza di banda (numero di istruzioni eseguite al secondo a regime) di un processore con pipeline non dipende dal numero di stadi della pipeline. **VERO**
- In una architettura con pipeline sono necessari più cicli di clock per completare una istruzione macchina. **VERO**
- In linea di principio, se si raddoppia la frequenza del clock si dimezza la latenza e si raddoppia l'ampiezza di banda. **VERO**

Esercizio 2

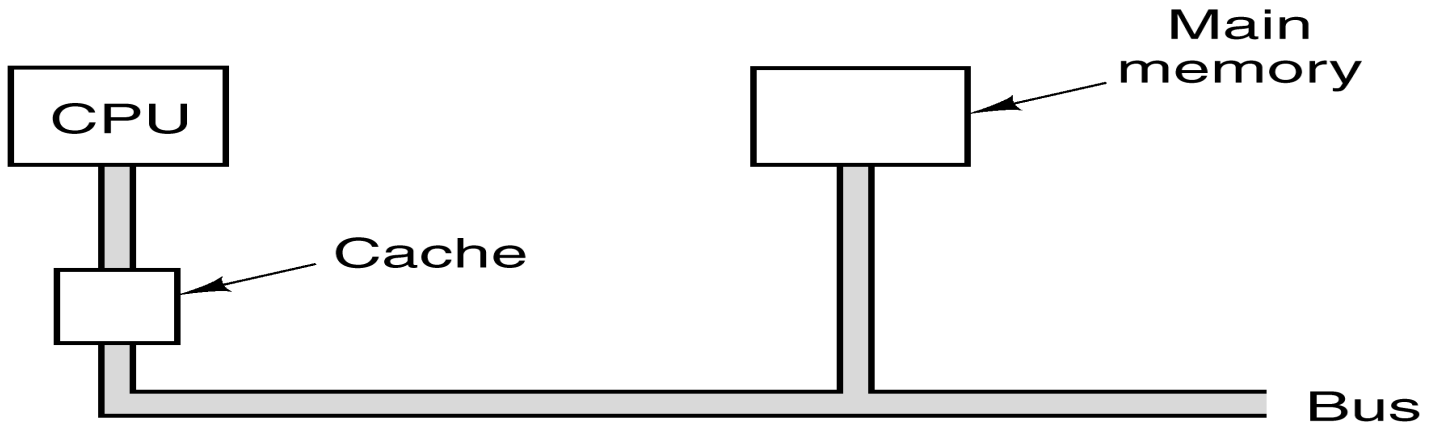
Con riferimento ai codici a rilevazione e correzione di errore indicare se le seguenti affermazioni sono vere o false.

- La distanza di Hamming [illegible]a codifica e il suo complemento a uno e pari alla lunghezza [illegible] codifica. VERO
- Con distanza di Hammin[illegible] possibile correggere 2 errori. FALSO
- Il numero di bit di controllo necessari per rilevare un errore singolo su un codice a 8 [illegible]inore rispetto al numero bit di controllo necessari per un codice a 16 bit. FALSO
- La distanza di Hamm[illegible] codice composto solo dalle parole 1100, 0011 e 1111 è [illegible] FALSO
- La percentuale di bit di controllo rispetto alla lunghezza complessiva di un codice [illegible]ezione di errore singolo diminuisce all'aumentare della lunghezza del codice. VERO
- Per rilevare r errori è ne[illegible] che un codice abbia una distanza di Hamming pa[illegible]1. FALSO
- Un bit di parità pe[illegible] solo di rilevare errori singoli. VERO
- Se in una parola si commet[illegible]errore singolo ma si conosce la sua posizione, il bit di parit[illegible]ficiente a correggerlo. VERO

Caching…



Memorie Cache



- La memoria è sempre più lenta della CPU e tende a rallentarla
- Memorie veloci sono disponibili ma solo per piccole dimensioni
- La *cache* (da *cacher*) funziona alla velocità del processore, e quindi *nasconde* la "lentezza" della memoria
- Contiene le ultime porzioni di memoria acceduta: se la CPU vuole leggere una di esse evita un accesso a memoria
- Funziona bene a causa della *località* degli accessi

Cache Hit Ratio

Se una parola viene letta *k* volte di seguito, *k – 1* volte sarà trovata in cache

- Cache hit ratio:

$$H = (k - 1) / k$$

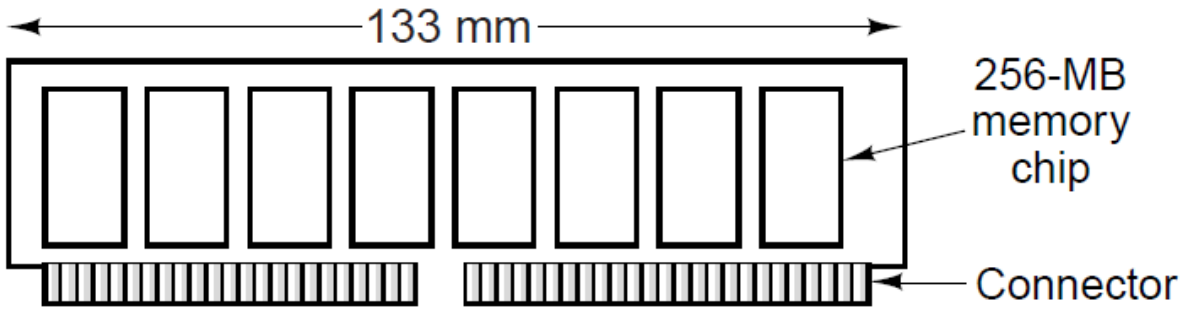
- Tempo medio di accesso a memoria:
  - *m*: tempo di accesso della memoria
  - *c*: tempo di accesso della cache

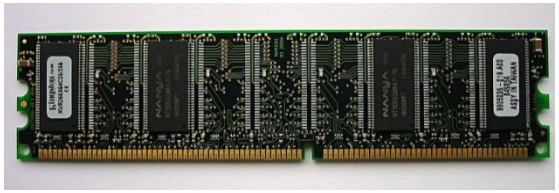
$$A = c + (1 - H)m$$

La memoria è organizzata in blocchi

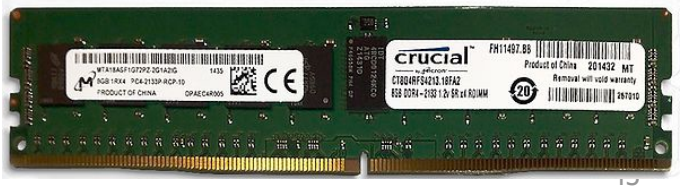
Per ogni *cache miss* un intero blocco è spostato in cache

Tipologie schede memoria

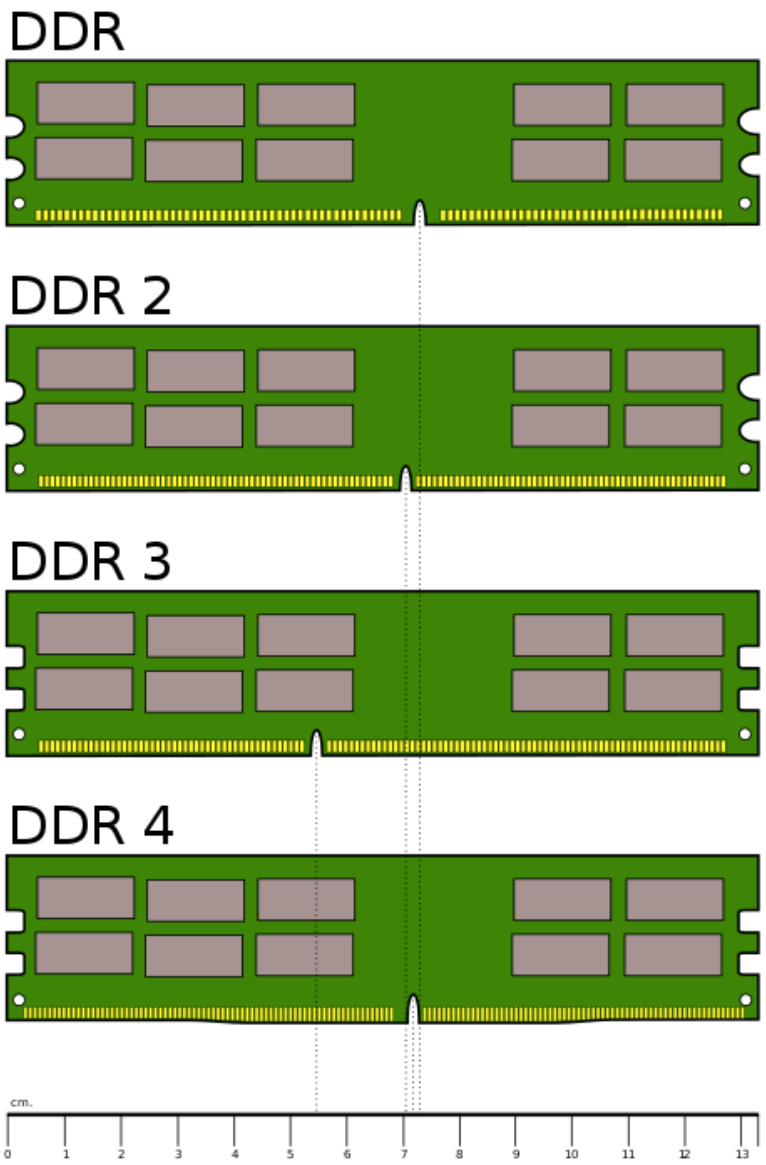


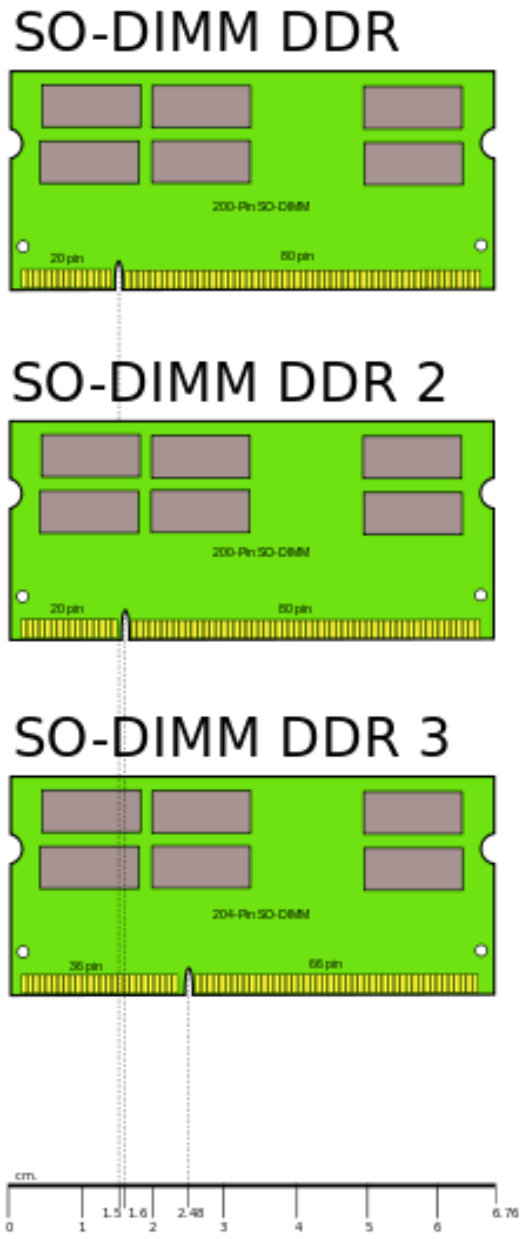


- SIMM (Single Inline Memory Module)
  - 72 piedini, 32 bit, 8-16 chip, 128 MByte
  - A coppie nel Pentium (bus dati 64 bit)
- DIMM (Double Inline Memory Module)
  - 120/240 piedini, 64 bit, 8 chip, 256 MByte
- SO-DIMM (Small Outline DIMM)
  - Per notebook di dimensioni più piccole
- DDR, DDR2, DDR3, (M)DDR4 (Double Data Rate): introducono un meccanismo di pipeline nella lettura/scrittura, fino a 288 pin.
- Alcune hanno bit di parità altre no

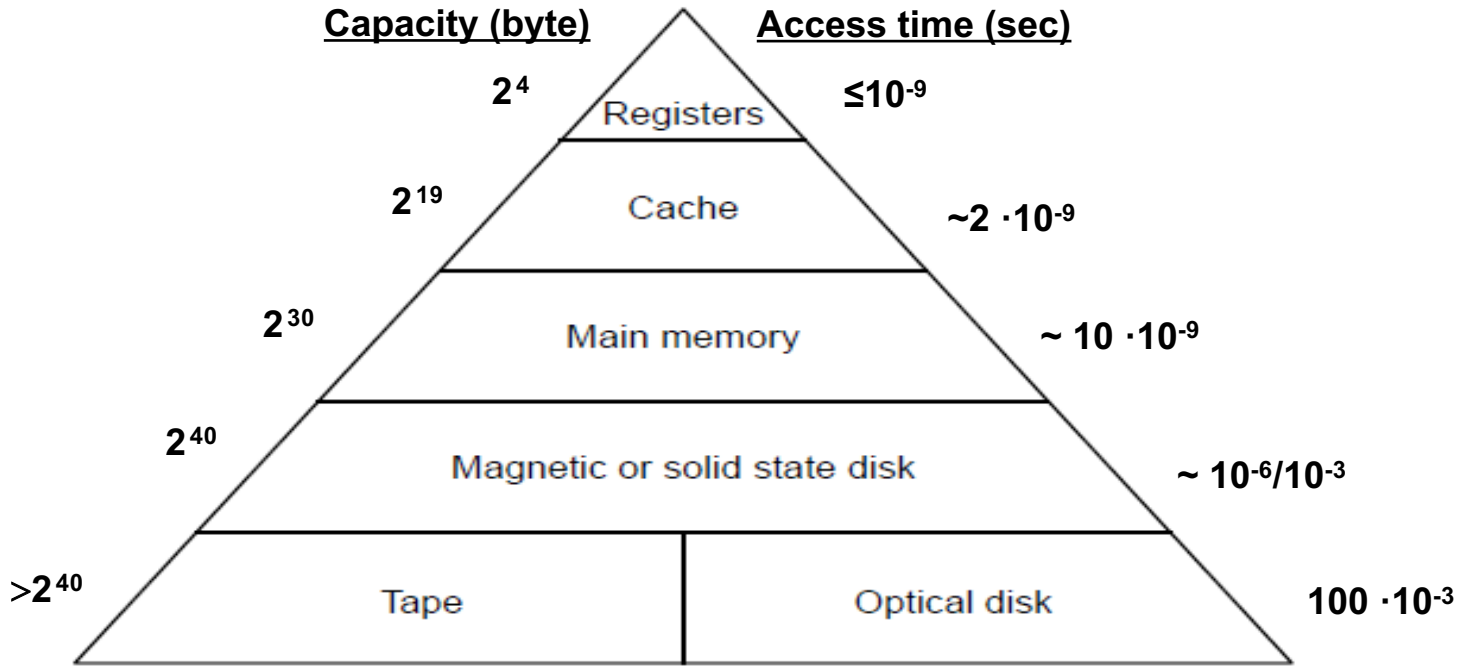


DDR a confronto





Gerarchie di memoria



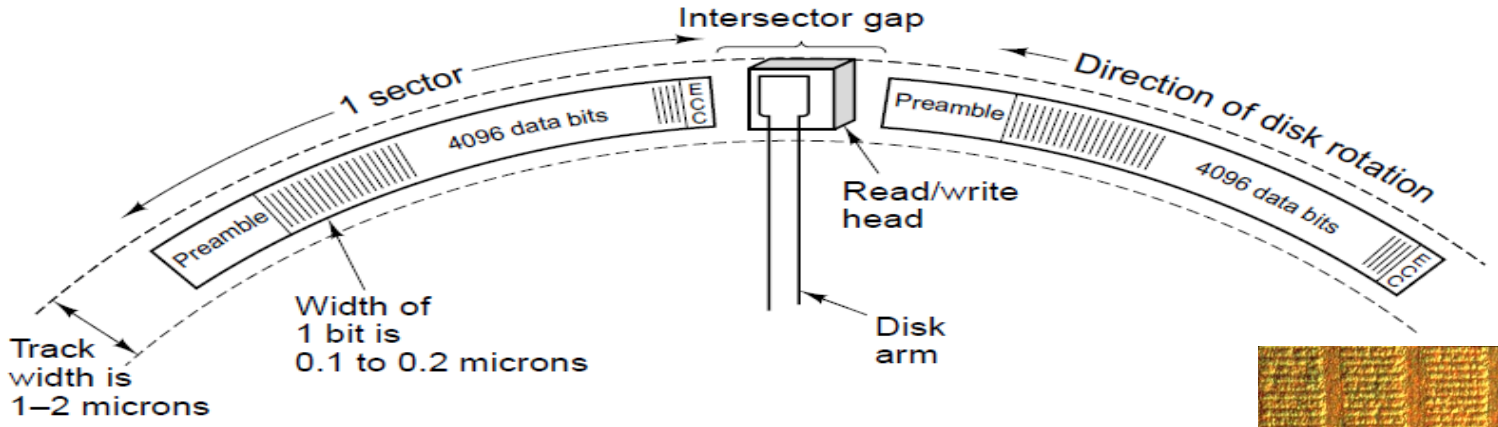
Scendendo nella gerarchia:

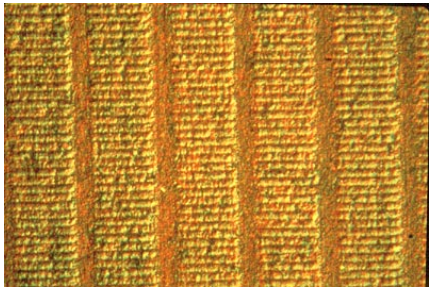
- Cresce il *tempo di accesso*
- Aumenta la *capacità*
- Diminuisce il *costo per bit*

Solo il livello più alto della gerarchia è a contatto con la CPU

Migrazione dei dati fra livelli della gerarchia

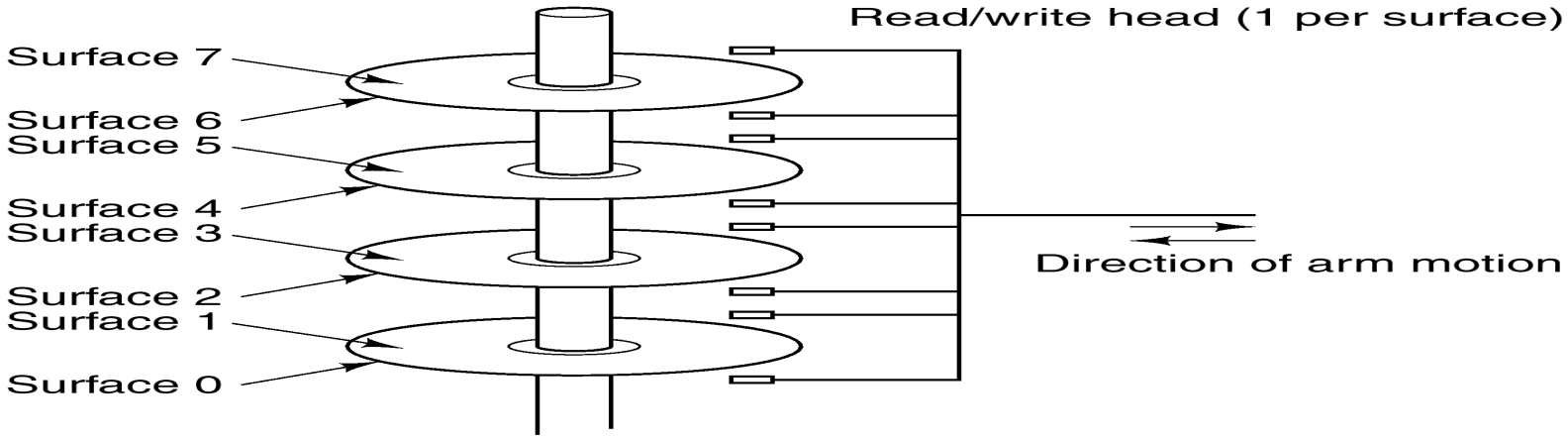
Dischi magnetici





- Dimensione: <10cm, Densità: 25Gb/cm
- Registrazione seriale su *tracce* concentriche
- 50.000 tracce/cm (larghe ~200nm)
- Dischi ad alta densità con bit registrati *perpendicolarmente*
- Tracce divise in *settori* contenenti i dati, un *preambolo* e un *ECC* (Error-Correcting Code) (la *capacità formattata* scende del 15%)
- Velocità di rotazione costante (5.400-10.800 RPM)
- Velocità di trasferimento di 150 MB/sec (1 settore in 3.5 µsec)
- *Burst rate*: velocità da quando la testina è sopra il primo bit
- *Sustained rate*: velocità di trasferimento in un certo intervallo

Dischi magnetici (2)

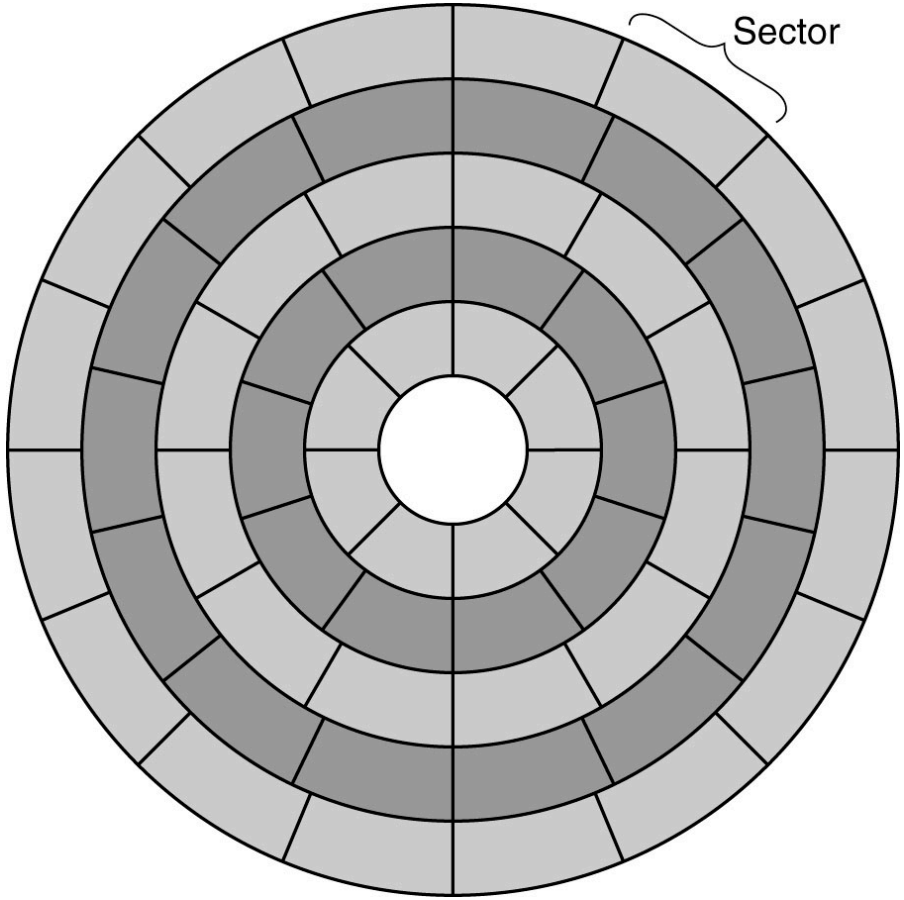


- *Cilindro*: insieme di tracce sulla stessa verticale
- *Tempo di seek* t_{seek}: spostamento delle testine sul cilindro desiderato, dipende in parte dalla distanza (~5-10ms)
- *Tempo di latency* t_{lat}: spostamento sul settore desiderato (~3-6ms)
- Tempo di accesso:

$$t_{acc} = t_{see} + t_{lat}$$

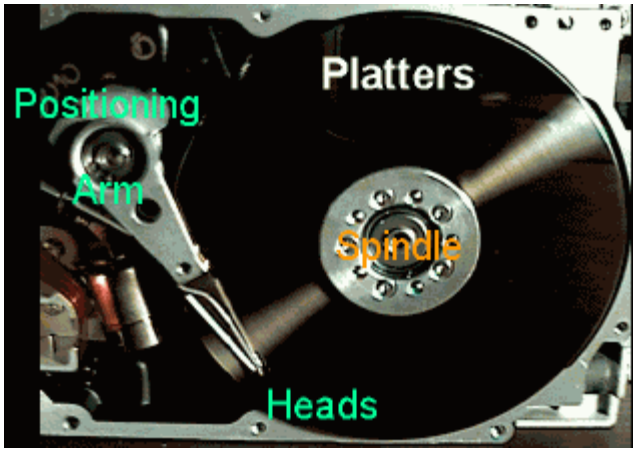
Organizzazione dei dati su disco

Densità di registrazione variabile con il raggio della traccia (~ 25 Gbit/cm)



La gestione è fatta da **controllori di disco** (CPU specializzate)

Un hard disk



Dischi IDE e EIDE

- IDE: standard nato con il PC XT IBM
  - Limite di 16 testine, 63 settori e 1024 cilindri: in tutto 504 MB, transfer rate: ~4MB/sec
- EIDE estende lo standard mediante lo schema LBA (Logical Block Addressing) che prevede 2^{28} settori
  - Totale di $2^{28} \times 2^{9}$B = 128GB
  - 2 controllori - 4 dischi per controllore
  - transfer rate più alta ~17MB/sec
- ATA-3 (AT Attachment) a 33MB/sec
- ATAPI-5 (ATA PAcket Interface) a 66MB/sec
- ATAPI-6 a 100MB/sec
  - LBA a 48 bit – Massimo: $2^{48} \times 2^{9}$B=128PB
- ATAPI-8 e successivi: basato su SATA (Serial ATA)
  - connettori a meno bit (da 80 a 7), tensioni più basse (0.5V), velocità maggiori (>500MB/sec)
- SCSI: Controller e interfaccia più intelligente, Bus con connessione *daisy chain,* versione moderna: Serial attached SCSI (>10Gb/sec)

Dischi RAID

Problema: miglioramento lento delle prestazioni dei dischi (1970: t_{seek}=50ms; 2018: t_{seek}=5-10ms)

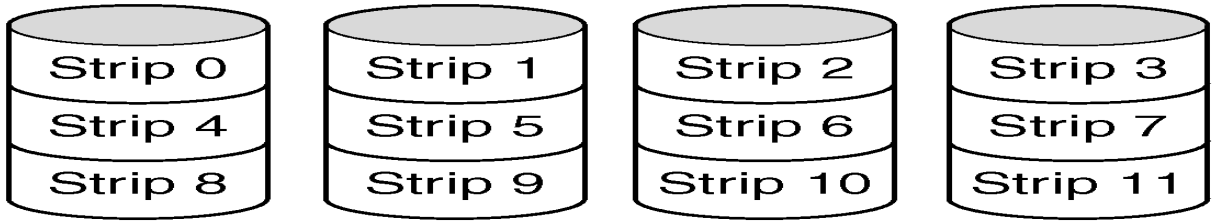
Soluzione: **RAID** (Redundant Array of Inexpensive Disks)

- Dividere i dati su più dischi
- Parallelizzare l'accesso
- Aumentare il data rate
- Introdurre una resistenza ai guasti

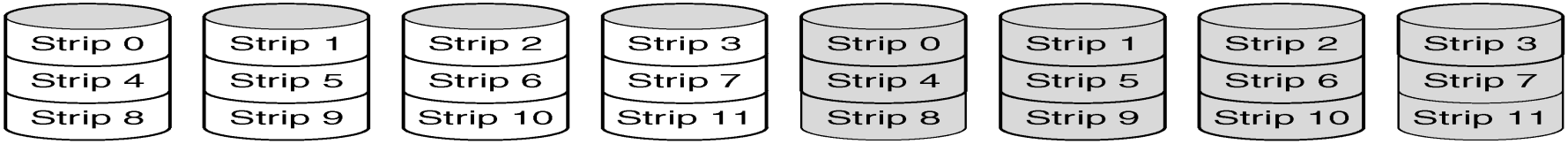
Contrapposti a **SLED** (Single Large Expensive Disk)

Data Striping: dati consecutivi nello stesso file vengono "affettati" e disposti su dischi diversi, dai quali possono essere letti (e scritti) in parallelo

RAID Level 0 e Level 1

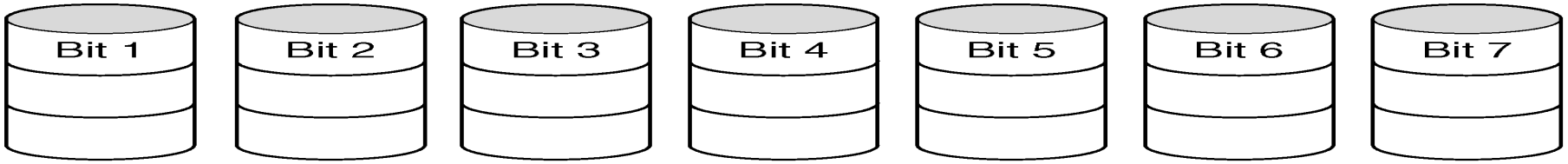


- Su n dischi si può guadagnare un fattore n sia in lettura che in scrittura
- Lo MTBF (*Mean Time Between Failures*) peggiora
- Non c'è ridondanza: non è un vero RAID



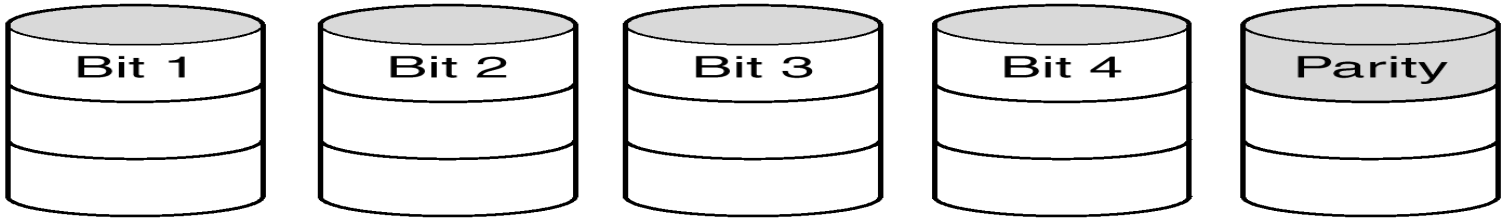
- Ciascun disco è duplicato: shadowing
- Ottime prestazioni soprattutto in lettura: molte possibilità di bilanciare carico
- Eccellente resistenza ai guasti
- Supportato anche da vari SO (Es. Windows)

RAID Level 2



- *Striping* a livello di word o di byte
- Esempio: un *nibble* (mezzo byte) più 3 bit: codice di Hamming a 7 bit
- Registrazione ad 1 bit per ogni disco
- Rotazione dei dischi sincronizzata
- Resiste a guasti semplici
- Guadagna un fattore 4 in read e write
- Forte *overhead* (nell'esempio 75%)
- Ha senso con molti dischi:
  - 32 bit+(6+1) parità $\Rightarrow$ 39 dischi
  - Overhead del 19%
  - Guadagna un fattore 32 in read e write

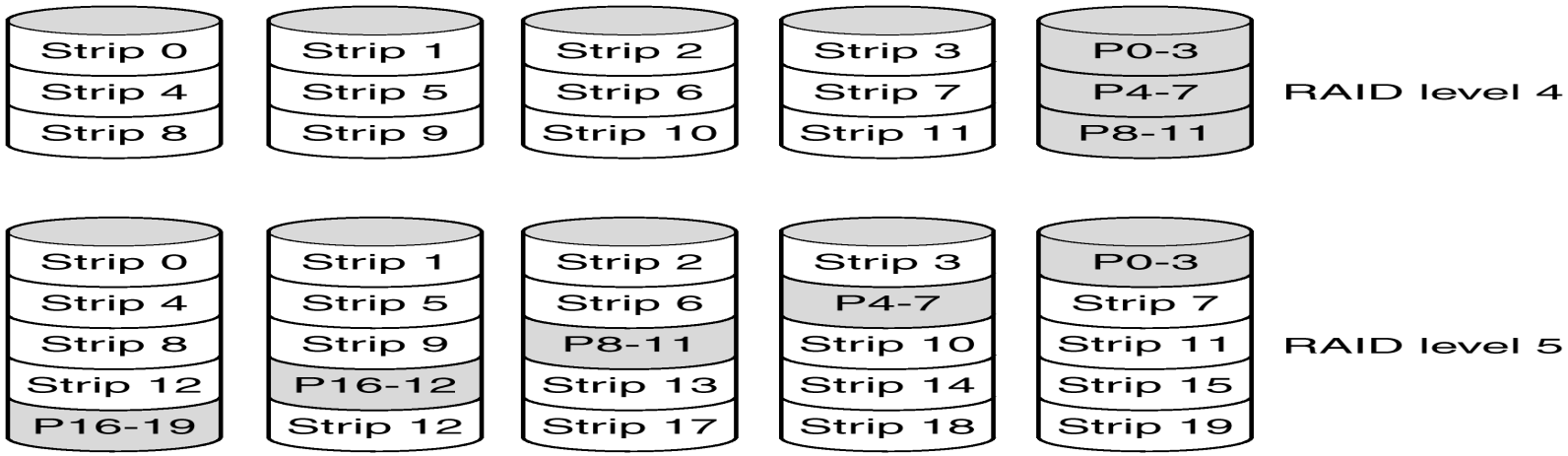
RAID Level 3



- Versione semplificata di RAID 2
- Resiste a guasti semplici! Il bit di parità, *sapendo quale drive è rotto*, consente la correzione
- *Overhead* abbastanza contenuto

RAID 2 e 3 offrono un'eccellente data rate ma permettono di gestire solo una operazione su disco per volta perché ciascuna operazione coinvolge tutti i dischi

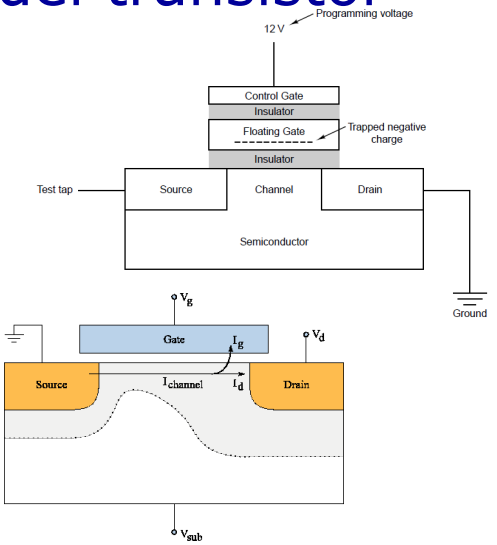
RAID 4 e RAID 5



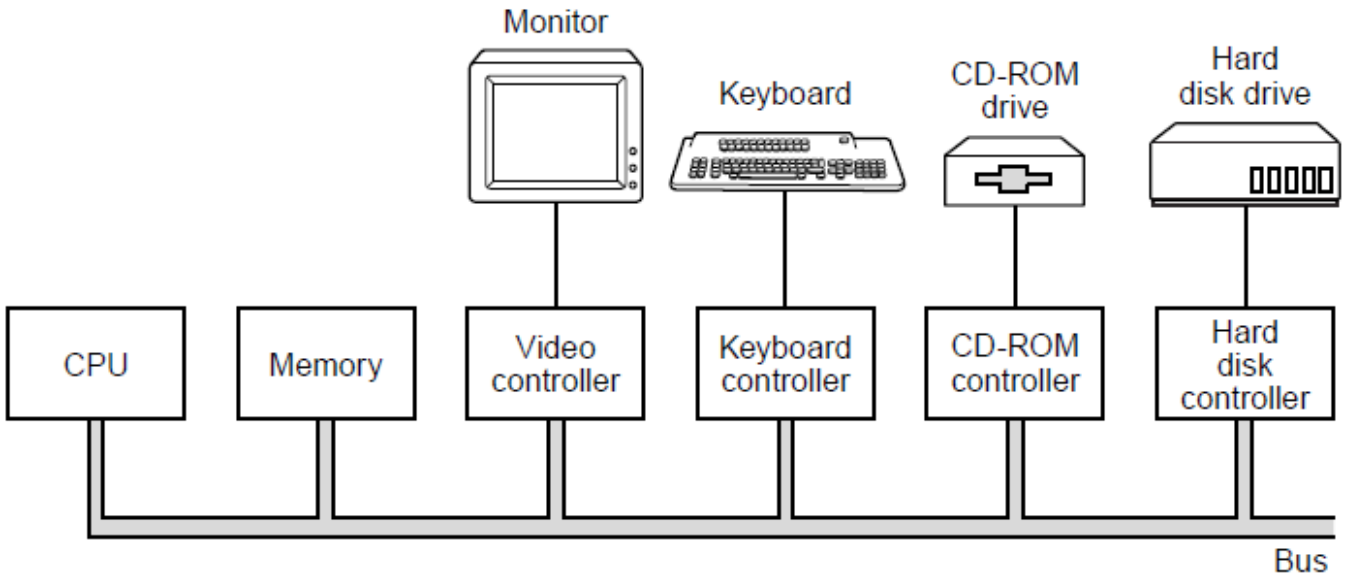
- *Striping* a livello di blocco: drive non sincronizzati
- RAID 4: la *strip* nell'ultimo disco contiene i bit di parita dell'insieme di bit omologhi di tutte le altre *strip*
- Resiste a guasti singoli (vedi RAID 3)
- Se una sola *strip* è scritta occorre leggere tutte le altre per calcolare la parità
- Il disco di parità è il collo di bottiglia
- RAID 5 distribuisce le *strip* di parità

Unità a stato solido (SSD)

- Basata sul fenomeno "Hot-carrier injection" dei transistor
- Celle di memoria flash a stato solido
- Montate sopra un normale transistor
- Applicando una tensione al CG:
  - Il FG si carica (no alimentazione)
  - Aumenta la tensione di commutazione
  - Test di commutazione a basso voltaggio
- Tempi di trasferimento: >200MB/sec
- Adatto a dispositivi mobili
- Costi più alti: ~1c/GB → ~1€/GB
- Maggiore "failure rate": ~ 100.000 Write
- Wear leveling: distribuzione uniforme delle scritture sulle celle dell'unità
- Aumento di capacità con celle multilivello
- Versione moderna: 3D XPoint

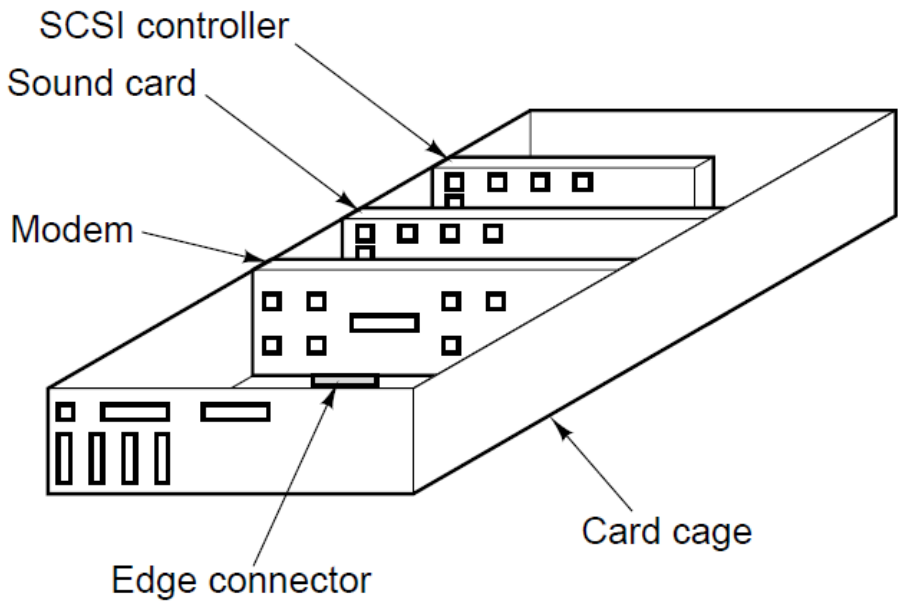


Dispositivi di I/O



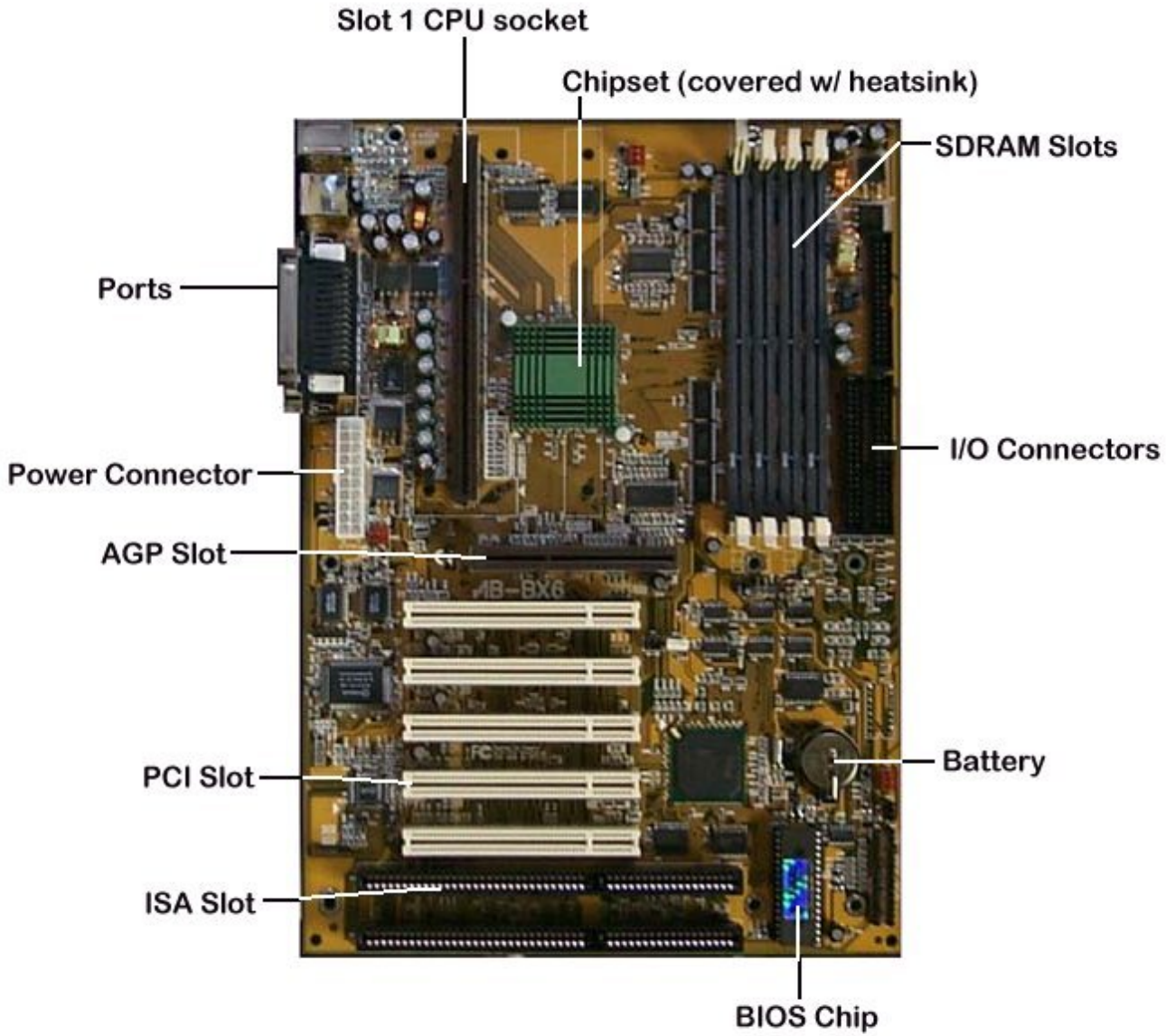
- I dispositivi di I/O sono connessi al bus tramite *controller*
- I controller gestiscono autonomamente i trasferimenti da e per la memoria: DMA (Direct Memory Access)
- Possono comunicare con la CPU tramite le *interruzioni*
- Il bus è condiviso da CPU e controller, e gli accessi sono regolati da un arbitro

Struttura fisica del PC

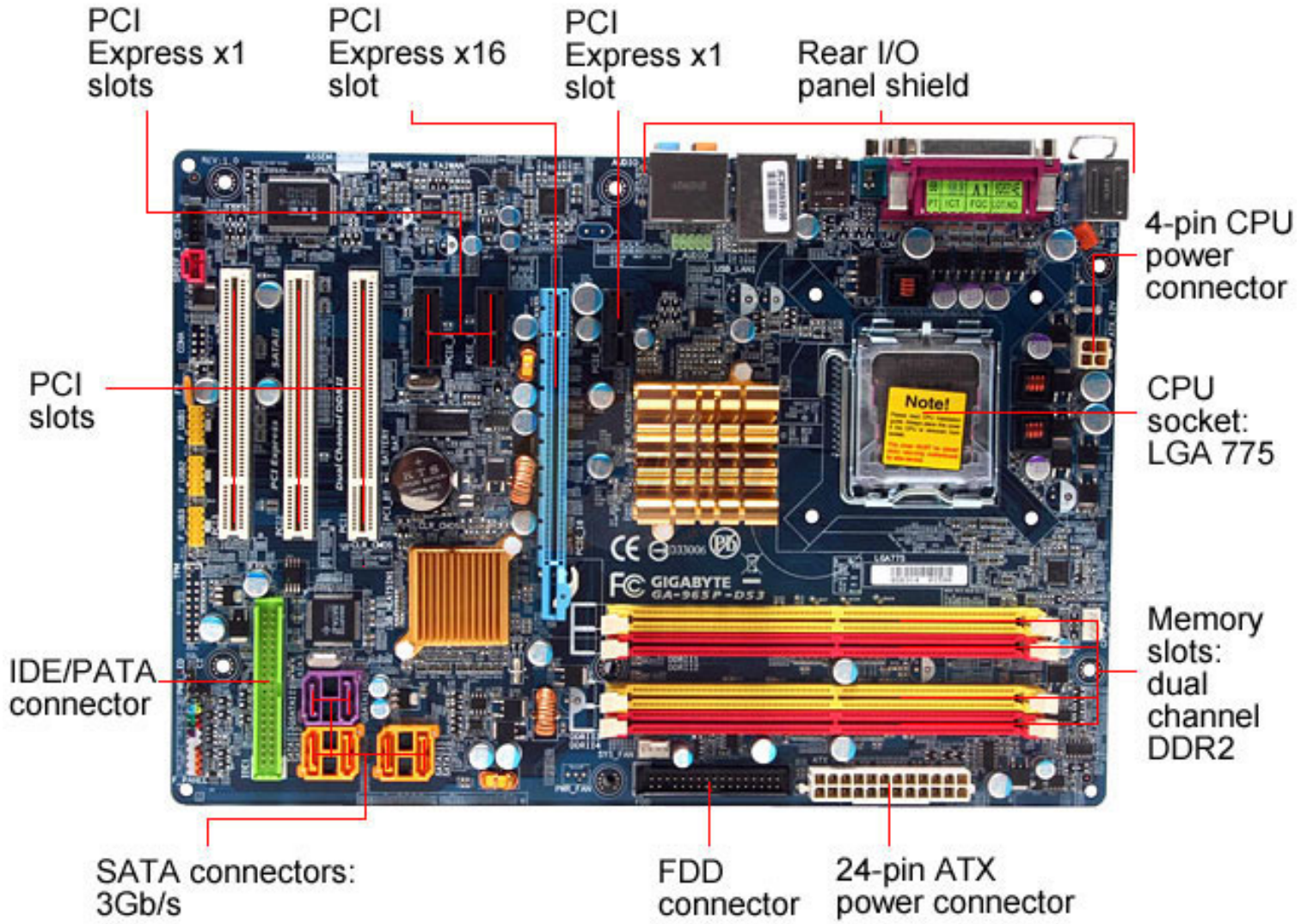


- La base della struttura è costituita dalla *Scheda Madre (Mother Board)*
- Sulla scheda madre sono la CPU, il *Chipset,* il bus e vari connettori per la memoria e i dispositivi di I/O
- Il bus è costituito da una serie di piste sul circuito stampato
- Spesso sono presenti più bus, secondo diversi standard
- Le schede di I/O vengono inserite nei connettori

Una schema madre



Scheda madre "moderna"



Esercizio 3

Si consideri una CPU con pipeline a 6 stadi che lavora a una frequenza di 400 Mhz e in cui ogni stadio viene eseguito in un ciclo di clock; indicare se le seguenti affermazioni sono vere o false.

- A regime VERO dizioni ideali la CPU completa un'istruzione ogni 2.5 nsec.
- Una istruzione richiede 10 nsec per ess FALSO uita.
- L'ampie FALSO anda della CPU è di 500 MIPS.
- La latenza della CPU è VERO nsec.
- In linea di principio, se la freque VERO clock aumenta a 800 Mhz si raddoppia l'ampiezza di banda.
- In linea di principio VERO requenza del clock scende a 200 Mhz si raddoppia la latenza.
- Il tempo di esecuzione di u VERO amma di 3 istruzioni è di 20 nsec.
- In linea di pri FALSO gliendo uno stadio si aumenta la latenza e si diminuisce l'a di banda.

Esercizio 4

Si consideri un programma che confronta il contenuto di una variabile X con tutti gli elementi di un vettore di interi A. Il vettore è composto da 5 elementi di 4 byte memorizzati in locazioni contigue della memoria principale mentre X è memorizzato in un'altra zona della memoria principale. L'esecuzione del programma avviene su un microprocessore che dispone di una cache con tempo di accesso di 2 nsec e di una memoria con tempo di accesso di 20 nsec. Si assuma che i trasferimenti tra memoria e cache avvengano per blocchi di 16B.

- Indicare la percentuale di successo nell'accesso alla cache (cache hit ratio) per la variabile X
- Indicare il tempo necessario per il primo accesso alla variabile X, espresso in nanosecondi.
- Indicare il tempo medio di accesso alla variabile X, espresso in nanosecondi.
- Indicare il cache hit ratio complessivo (percentuale globale di successo nell'accesso alla cache) e il tempo medio di accesso alla memoria del programma;
- Assumendo che il confronto di due elementi sia eseguito dal microprocessore in 1 nsec, indicare il tempo complessivo necessario all'esecuzione del programma, espresso in nanosecondi.

Esercizio 4.1

Si consideri un programma che confronta il contenuto di una variabile X con tutti gli elementi di un vettore di interi A. Il vettore è composto da 5 elementi di 4 byte memorizzati in locazioni contigue della memoria principale mentre X è memorizzato in un'altra zona della memoria principale. L'esecuzione del programma avviene su un microprocessore che dispone di una cache con tempo di accesso di 2 nsec e di una memoria con tempo di accesso di 20 nsec. Si assuma che i trasferimenti tra memoria e cache avvengano per blocchi di 16B.

- Indicare la percentuale di successo nell'accesso alla cache (cache hit ratio) per la variabile X

Esercizio 4.2

Si consideri un programma che confronta il contenuto di una variabile X con tutti gli elementi di un vettore di interi A. Il vettore è composto da 5 elementi di 4 byte memorizzati in locazioni contigue della memoria principale mentre X è memorizzato in un'altra zona della memoria principale. L'esecuzione del programma avviene su un microprocessore che dispone di una cache con tempo di accesso di 2 nsec e di una memoria con tempo di accesso di 20 nsec. Si assuma che i trasferimenti tra memoria e cache avvengano per blocchi di 16B.

- Indicare il tempo necessario per il primo accesso alla variabile X, espresso in nanosecondi.

Esercizio 4.3

Si consideri un programma che confronta il contenuto di una variabile X con tutti gli elementi di un vettore di interi A. Il vettore è composto da 5 elementi di 4 byte memorizzati in locazioni contigue della memoria principale mentre X è memorizzato in un'altra zona della memoria principale. L'esecuzione del programma avviene su un microprocessore che dispone di una cache con tempo di accesso di 2 nsec e di una memoria con tempo di accesso di 20 nsec. Si assuma che i trasferimenti tra memoria e cache avvengano per blocchi di 16B.

- Indicare il tempo medio di accesso alla variabile X, espresso in nanosecondi.

Esercizio 4.4

Si consideri un programma che confronta il contenuto di una variabile X con tutti gli elementi di un vettore di interi A. Il vettore è composto da 5 elementi di 4 byte memorizzati in locazioni contigue della memoria principale mentre X è memorizzato in un'altra zona della memoria principale. L'esecuzione del programma avviene su un microprocessore che dispone di una cache con tempo di accesso di 2 nsec e di una memoria con tempo di accesso di 20 nsec. Si assuma che i trasferimenti tra memoria e cache avvengano per blocchi di 16B.

- Indicare il cache hit ratio complessivo (percentuale globale di successo nell'accesso alla cache) e il tempo medio di accesso alla memoria del programma;

Esercizio 4.5

Si consideri un programma che confronta il contenuto di una variabile X con tutti gli elementi di un vettore di interi A. Il vettore è composto da 5 elementi di 4 byte memorizzati in locazioni contigue della memoria principale mentre X è memorizzato in un'altra zona della memoria principale. L'esecuzione del programma avviene su un microprocessore che dispone di una cache con tempo di accesso di 2 nsec e di una memoria con tempo di accesso di 20 nsec. Si assuma che i trasferimenti tra memoria e cache avvengano per blocchi di 16B.

- Assumendo che il confronto di due elementi sia eseguito dal microprocessore in 1 nsec, indicare il tempo complessivo necessario all'esecuzione del programma, espresso in nanosecondi.

Esercizio 5

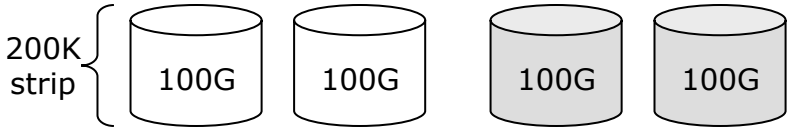
Illustrare la composizione e funzionamento di un'unità RAID di 200 GB (spazio utilizzabile di memoria fisica) e con blocchi (strip) di 512 KB, con riferimento:

(A) ad una configurazione di livello 1 con 4 dischi,
(B) ad una configurazione di livello 2,
(C) ad una configurazione di livello 4 con 5 dischi e
(D) ad una configurazione di livello 5 con 3 dischi.

Indicare in entrambi i casi la dimensione effettiva di memoria fisica necessaria per la realizzazione (in numero di byte).

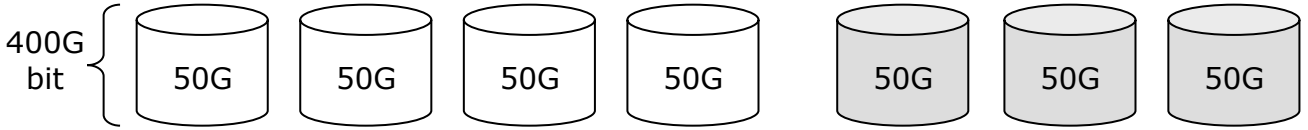
Soluzione esercizio 5

(A)



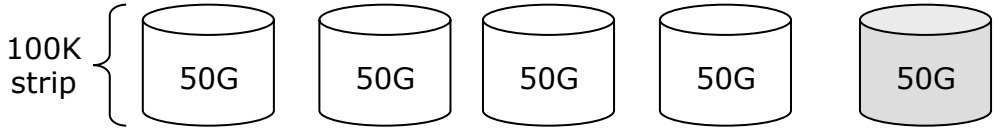
Totale
400GB

(B)



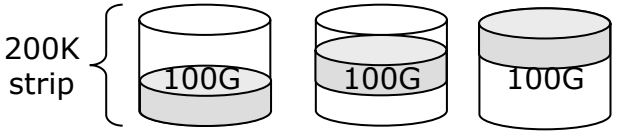
Totale
350GB

(C)



Totale
250GB

(D)



Totale
300GB