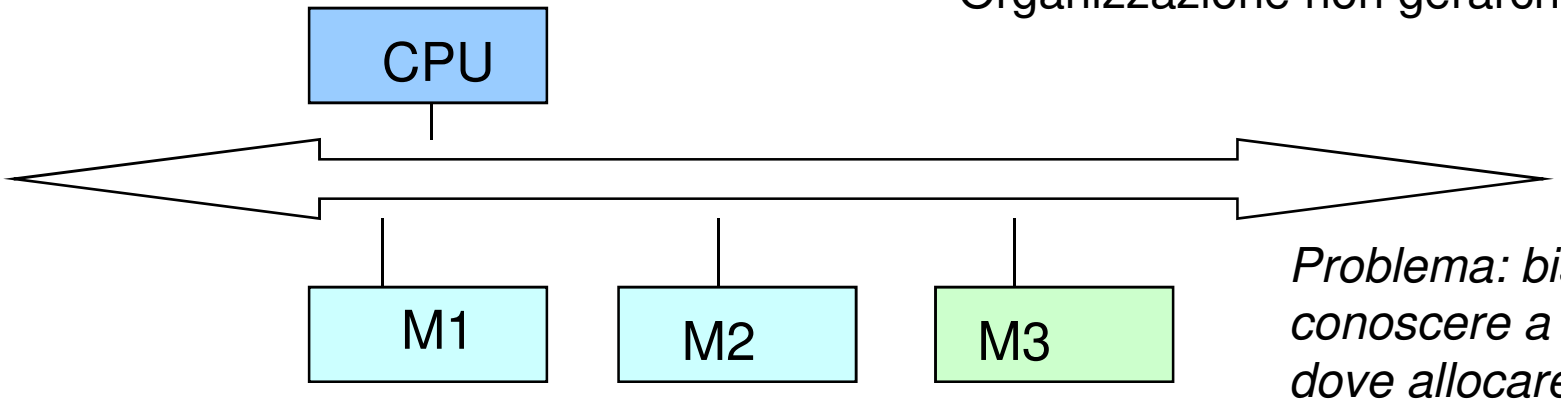
Calcolatori Elettronici

# Gerarchia di memorie

# Organizzazione delle memorie

- *La memoria ideale dovrebbe essere a capacita' infinita, tempo di accesso nullo e costo e consumo nullo*.
- Nei casi reali l'organizzazione del calcolatore comprende diversi tipi di memorie con caratteristiche diverse
- Alcune memorie sono dedicate, ossia sono progettate per contenere specifici tipi di dati (es la EPROM o flash di bootstrap)
- Altre memorie sono general-purpose e possono essere accedute mediante specifici metodi di indirizzamento per leggere o scrivere dati di qualsiasi tipo. Il loro impiego dipende dalle loro caratteristiche e da come il dato viene impiegato durante il funzionamento del calcolatore

Organizzazione non gerarchica



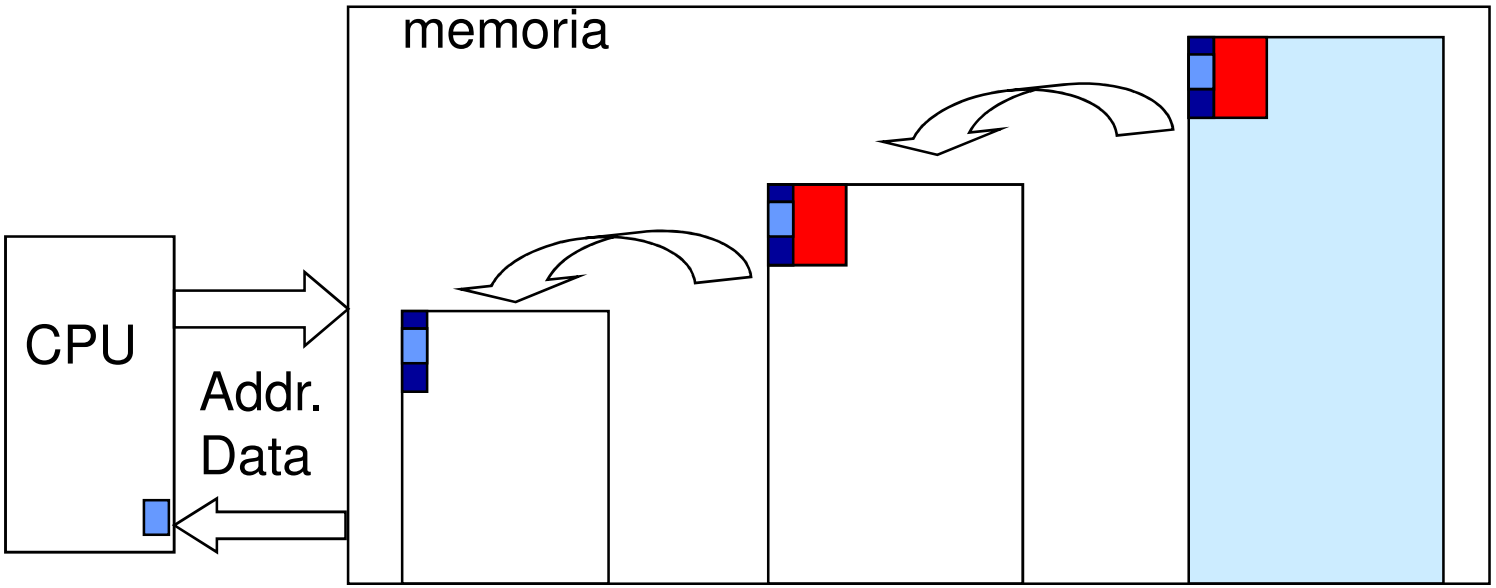

*Problema: bisogna conoscere a priori dove allocare I dati nelle memorie*

Es. nei sistemi embedded: M1 RAM veloce per i dati, M2 RAM veloce per il codice M3 FLASH lenta per il bootstrap Mint interna velocissima.

# Organizzazione delle memorie

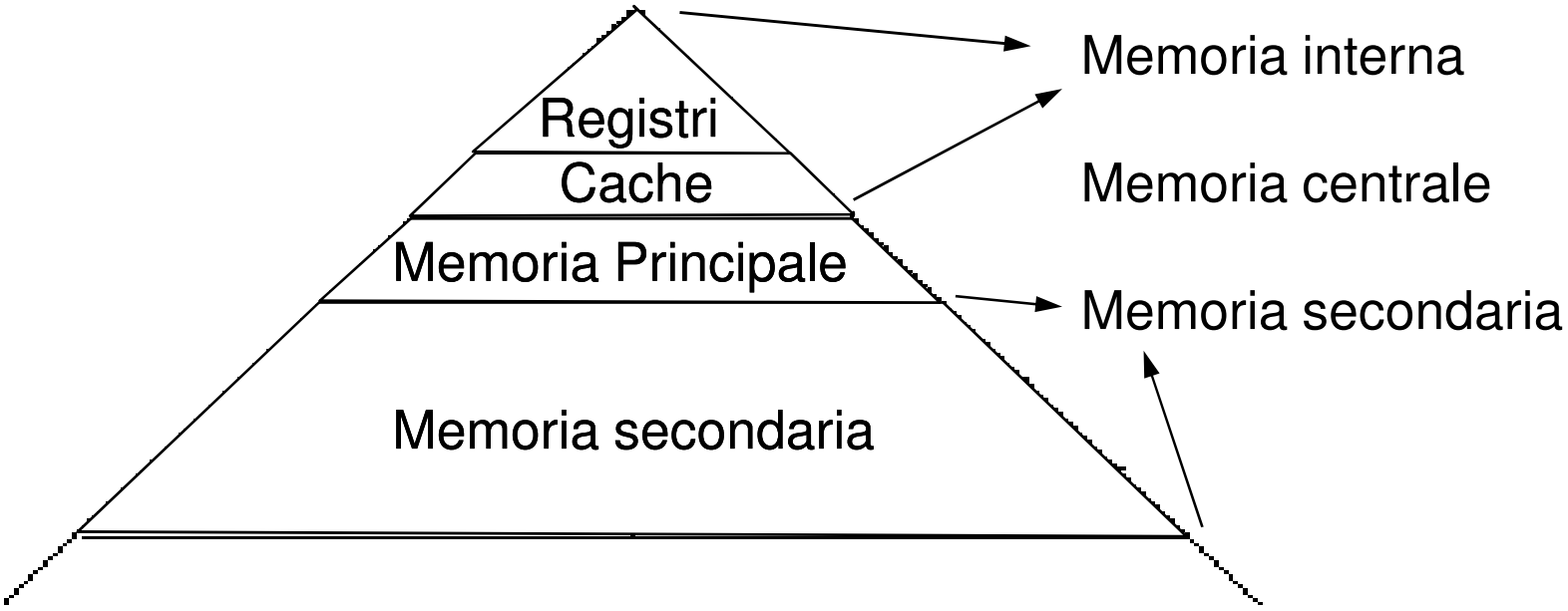
- Organizzazione gerarchica (tipica dei calcolatori general-purpose)

- La CPU vede un solo tipo di memoria, indirizzata direttamente tramite un identificatore (indirizzo)
- L'hardware (MMU) e il sistema operativo gestiscono invece un insieme di memorie organizzate gerarchicamente che contengono repliche dei dati in modo che la CPU trovi ad ogni accesso il dato utile nella memoria più veloce possibile
- **Gerarchia di memorie:** La memoria del calcolatore e' organizzata gerarchicamente in modo da comprendere poche memorie a bassa capacità ed alti costi ma molto veloci e molta memoria  più lenta e di capacità maggiore



# Gerarchia di memoria

- Nella gerarchia di memoria i dati sono temporaneamente trasferiti in memorie sempre più veloci.
- L'obiettivo e' di organizzare le politiche di piazzamento ed accesso dei dati in modo tale da avere i dati più frequentemente usati, o in generale più "utili" virtualmente sempre nelle memorie più veloci.



# Memoria

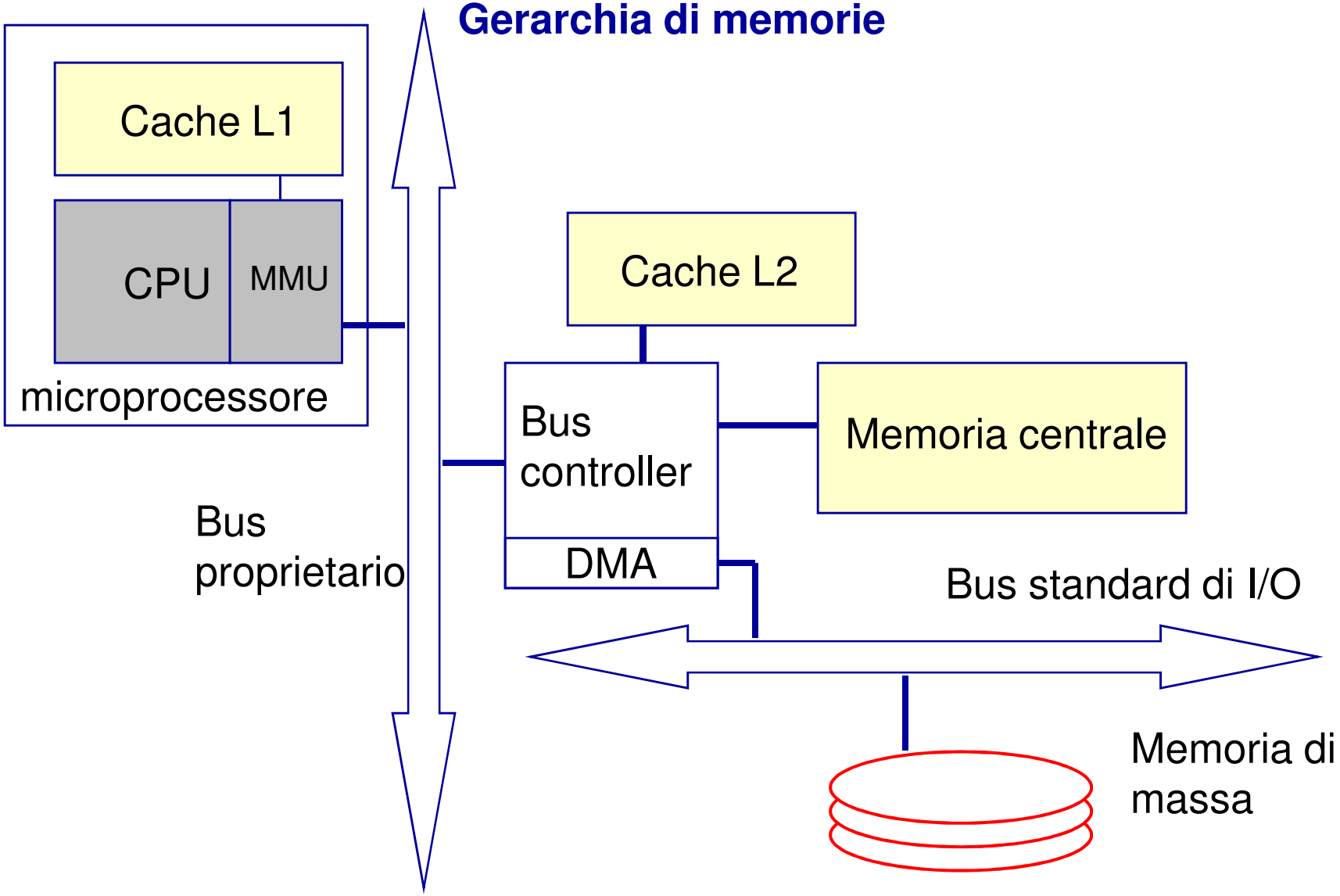
- La **memoria interna** alla CPU è costituita dai registri ed è caratterizzata da: alta velocità (comparabile con quella del processore) e limitate dimensioni (al più qualche migliaio di byte).

- La **memoria centrale o principale** è caratterizzata da,dimensioni molto maggiori della memoria interna alla CPU (fino a qualche Gbyte) ma tempi di accesso più elevati. È accessibile in modo diretto tramite indirizzi.
- Nei sistemi attuali un livello di memorie **cache** e' stato inserito tra CPU e memorie centrali

**Memorie secondarie**:

- In un calcolatore esistono diverse memorie secondarie (o memorie di massa) ad alta capacità bassi costi e non volatili

## Gerarchia di memorie



# Gerarchia di memorie

Regola base dell'efficienza è rendere di massima velocità il caso più frequente

- Sfruttamento della località spaziale e temporale
  Principio di località  un programma in ogni istante utilizza una porzione limitata dello spazio di indirizzamento

- **Località spaziale**: accedendo ad un dato e' assai probabile che si debba accedere ad altri dati localizzati "vicino" nello spazio di indirizzamento

- **Località temporale:** accedendo ad un dato e' assai probabile che si debba riaccedere ad esso in un tempo "vicino"

- Dati che inizialmente si trovano ad un livello più basso
  - (loc. temporale) conviene spostarli in memorie più veloci
  - (loc. spaziale) conviene spostare anche i dati vicini

- **I programmi NON vedono la gerarchia ma referenziano i dati come se fossero sempre in memoria centrale** (a parte per i registri che sono nominati esplicitamente) perciò  bisogna mantenere il più vicino possibile alla CPU dati utilizzati più recentemente

  La gerarchia delle memorie deve prevedere in successione memorie sempre più larghe e più lente per mantenere i dati nei livelli più alti proporzionatamente alla previsione della frequenza d'uso.

# Blocchi di locazione di memoria

Gerarchia di memoria: più livelli

- **Blocco**: unità di informazione minima scambiata fra livelli (ex: 32 bytes)
- **Hit**: un accesso alla memoria che trova l'informazione cercata nel livello superiore
- **Hit Rate**: frequenza di accessi trovati nel livello superiore (h)
- **Miss Rate**: (1 - h) caso di insuccesso (non trovo il dato nella memoria adiacente)
- **Hit Time**: tempo di accesso al livello superiore Th
- **Miss Penalty**: tempo per rimpiazzare un blocco nel livello superiore più il tempo per fornirlo alla CPU. Il "miss penalty", Tmp, può anche essere scomposto nei tempi per reperire il primo dato di un blocco più il tempo per trasferire gli altri. Comprende certamente un Th e un "overhead" nel caso di miss (Tmiss)

- Tacc = h Th + (1-h)Tmp

- Tmp = Th + Tmiss allora **Tacc = Th + (1-h)Tmiss**

Il Tmiss comprende il tempo per reperire il blocco nella gerarchia di livello più basso più (eventualmente trascurabile) il tempo per rimpiazzare i dati dal livello alto al livello più basso (se il livello più alto deve essere "svuotato", cioè se devo salvare i dati che sono in esso presenti, operazione di *write-back*)

# Gerarchia: esempio

- **Esempio**
- Supponiamo un sistema a due livelli in cui per accedere al primo livello (memoria centrale) il tempo di accesso sia 0.1 microsec (100 ns) per ogni dato e per accedere al secondo (ad esempio HD) sia di 0.1 msec.
- Quando si accede al secondo livello il dato deve essere copiato nel primo.

- Ipotizziamo che le probabilità di trovare il dato al primo livello siano del 95% (h).
- Trascuriamo il tempo necessario al processore per capire dove si trova il dato (trascurabile rispetto al tempo di accesso).
- Quanto è il tempo di accesso medio?

- In microsec
- Tacc= 0.95 0.1 + 0.05 (0.1 +100)= 0.095+5.005= 5.1 microsec

- Basta un 5% di penalizzazione di miss che tutto l'accesso medio è fortemente rallentato, nel caso specifico di ben 50 volte rispetto al tempo di hit

- Circa $10^3$ differenza di dimensioni (es. 10K – 10M)
- Circa $10^3$ differenza di tempo di accesso (es. $10^{-7}$s – $10^{-4}$s)

# Gerarchia di memorie

Parametri di progettazione gerarchia di memoria:

- ⇨ Quanti livelli di gerarchia (3?, 4?)
- ⇨ che dimensione e velocità per ogni livello
- ⇨ Quali livelli? (L1, L2, RAM, Disco…).

ogni tipo di memoria si definisce in base al

1) **Piazzamento** del blocco (o funzione di traduzione o mapping): dove può essere allocato il blocco al livello corrente
2) **Identificazione** del blocco: come si può ritrovare il blocco a livello corrente
3) **Rimpiazzamento** del blocco: come si sceglie quale blocco sostituire
4) Strategia di **scrittura**

Ad esempio **a livello di registri**

**L'identificazione** è nominale (mov ax,15)

**il piazzamento e rimpiazzamento, scrittura** e' definito dal compilatore

# Memoria centrale

- **Memoria centrale** è la memoria fisicamente e logicamente collegata direttamente alla CPU.

- Dal punto di vista della gerarchia:

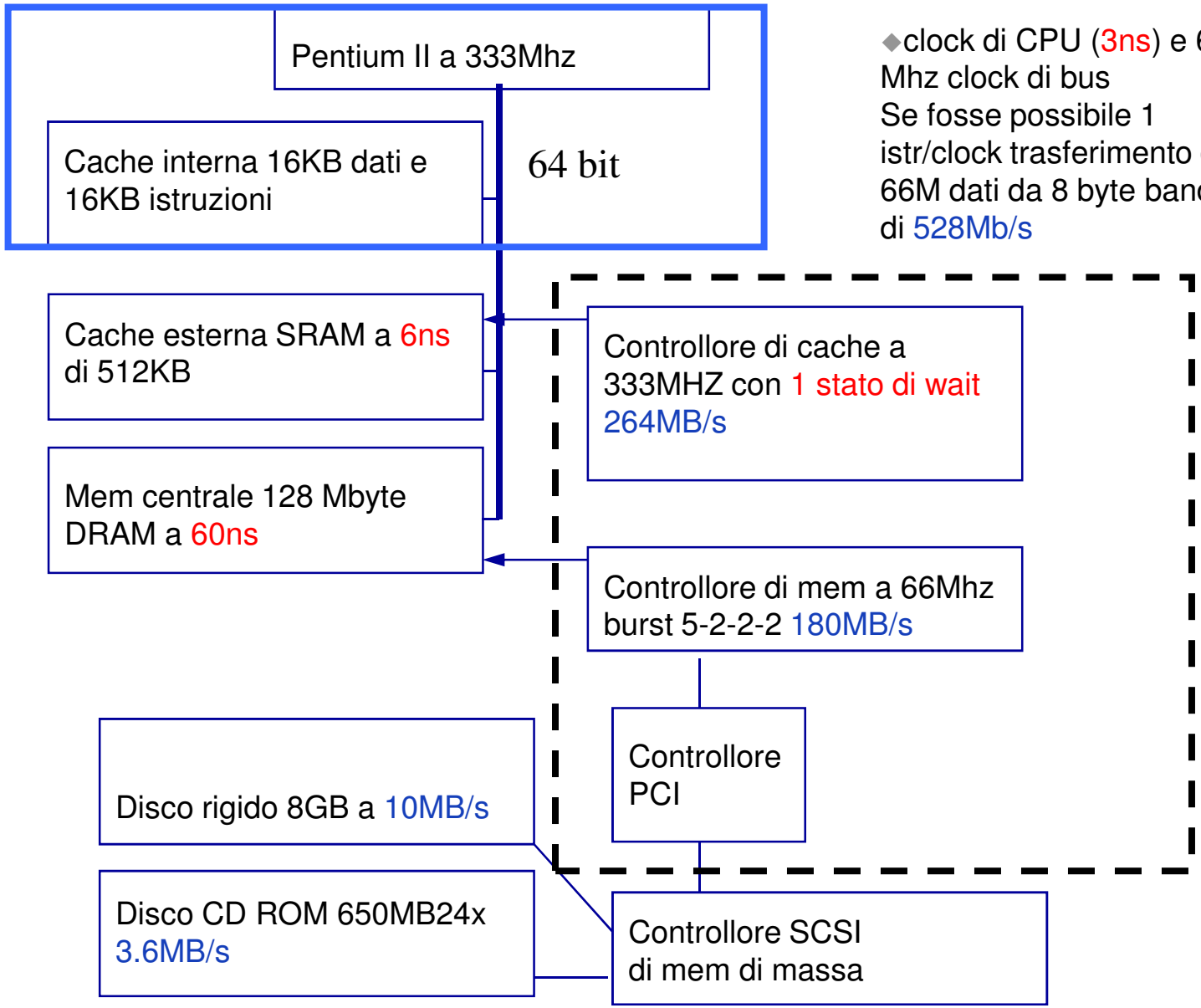
1) **Piazzamento** del blocco: deciso dall'indirizzo nell'istruzione
2) **Identificazione** del blocco: indirizzo (sw, compilatore, SO, traduzione diretta)
3) **Rimpiazzamento** del blocco: deciso dal codice
4) Strategia di **scrittura:** deciso dal codice

Il software (applicativo + SO) ha il controllo dello spazio di indirizzamento

Nota la mappa della memoria

- La CPU legge e scrive su memoria centrale mandando un indirizzo fisico della dimensione pari al suo bus di indirizzi (spazio di indirizzamento) per l'8086 ad esempio 20 bit, creo un indirizzo di 20 bit (indirizzo 1 M).

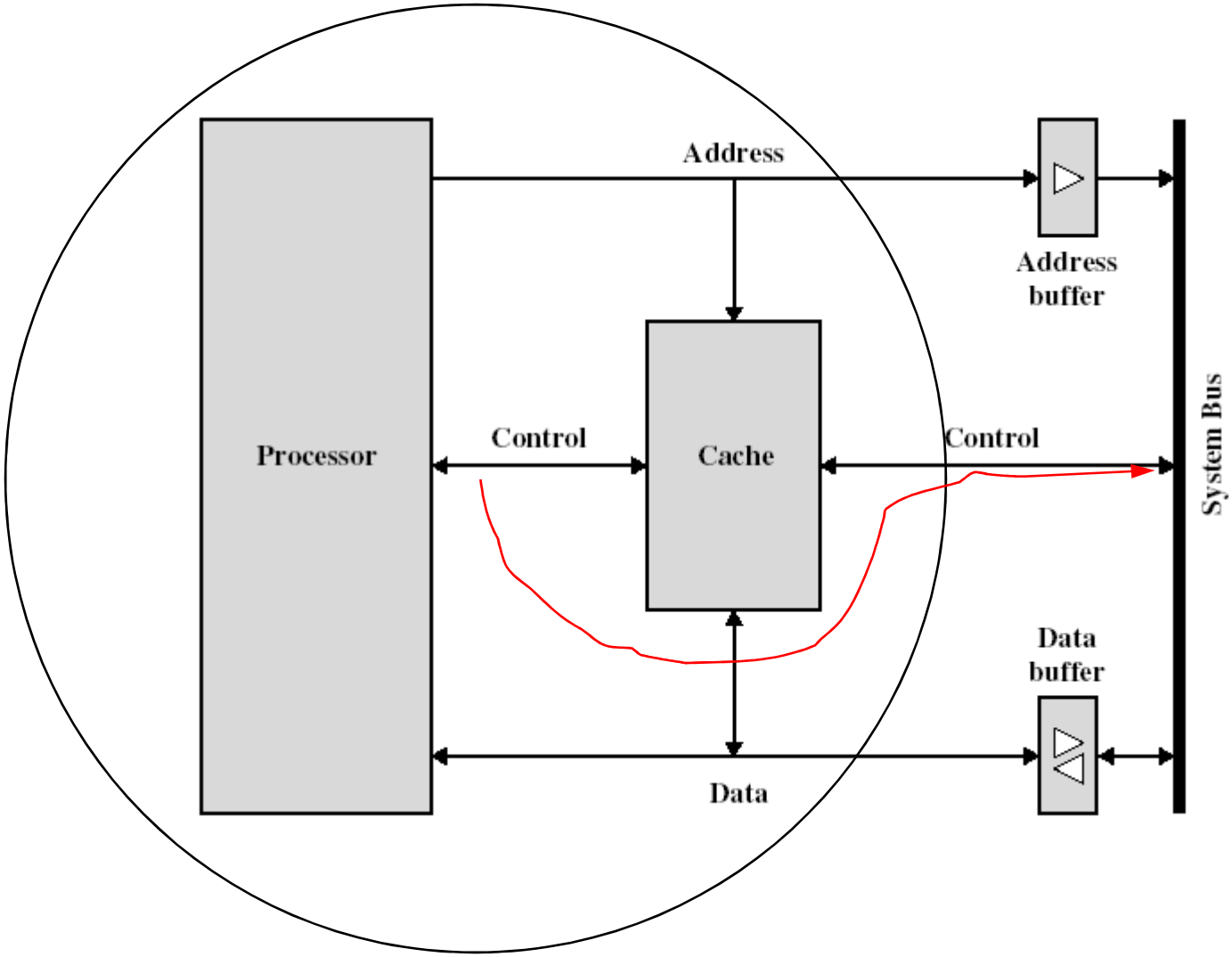
# Esempio



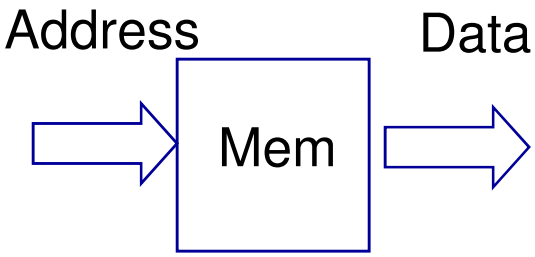

- clock di CPU (3ns) e 66 Mhz clock di bus
  Se fosse possibile 1 istr/clock trasferimento di 66M dati da 8 byte banda di 528Mb/s

# Memorie e cache

- Le memorie cache non sono visibili dal programmatore

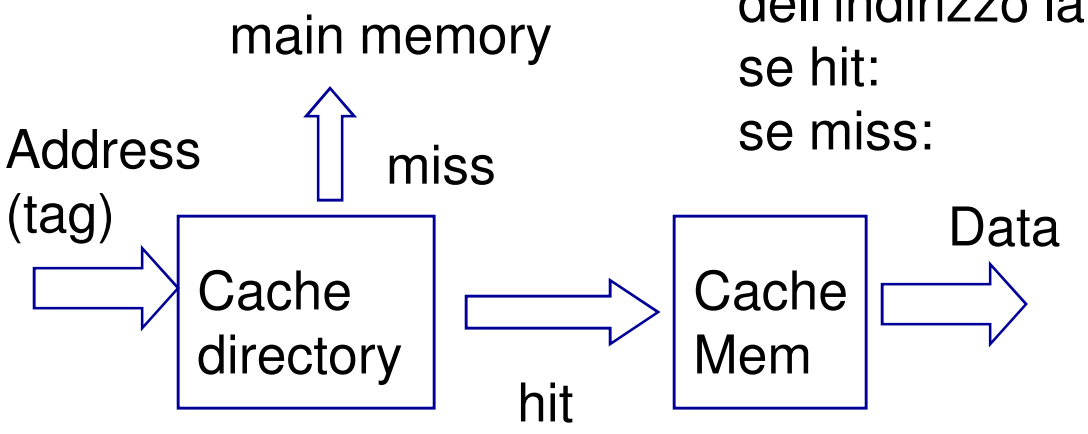


# Es: Memoria principale e cache



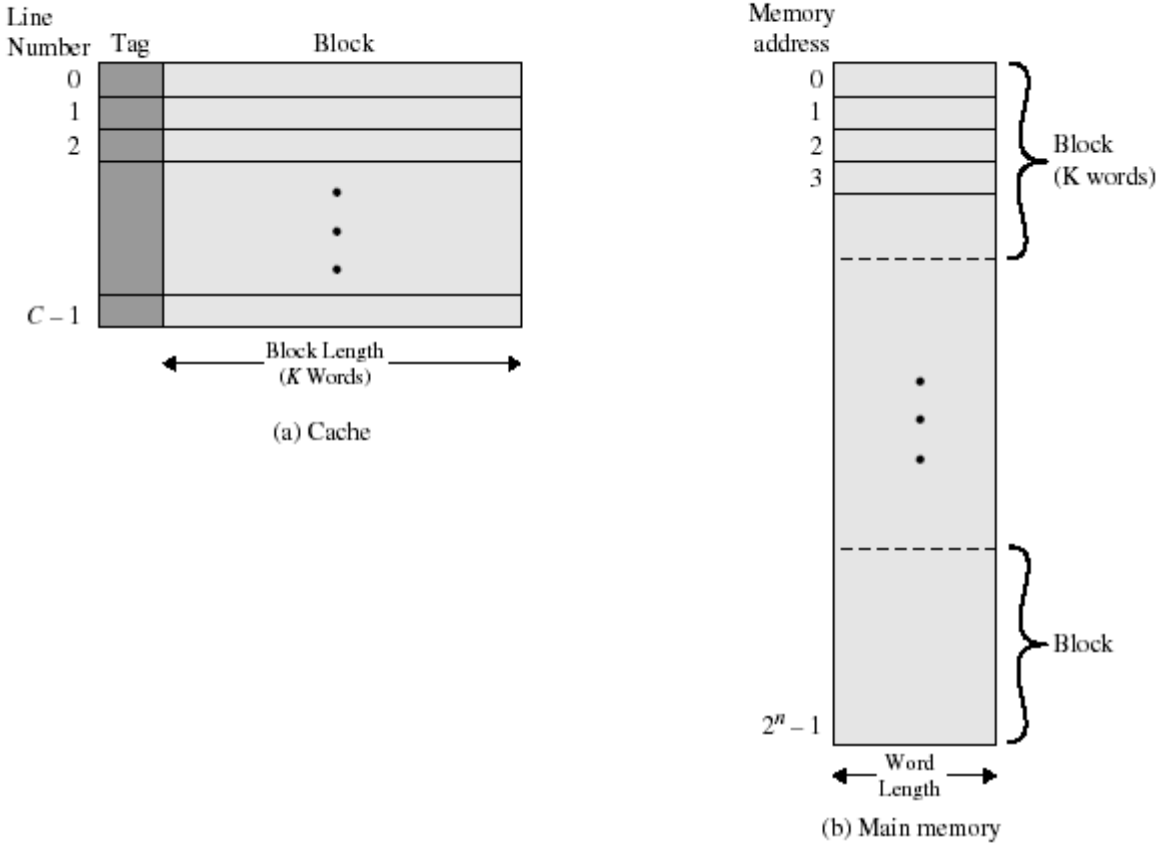

In memoria principale il dato è individuato dall'indirizzo
Tacc costante (es Tacc=80ns)

In cache il dato è individuato da parte dell'indirizzo la cache è più piccola
se hit: Tacc=Thit (es Thit=5ns)
se miss: Tacc= Tmiss



# Cache

- In cache viene ricopiato un blocco di parole normalmente contenute in memoria centrale (supponiamo il parallelismo di parola = n byte così che ogni blocco B contiene nK byte)
- Alla cache RAM è associata la cache directory, che contiene (parti di) indirizzo del blocco corrispondente. La CPU trova il dato confrontando per comparazione di indirizzo.



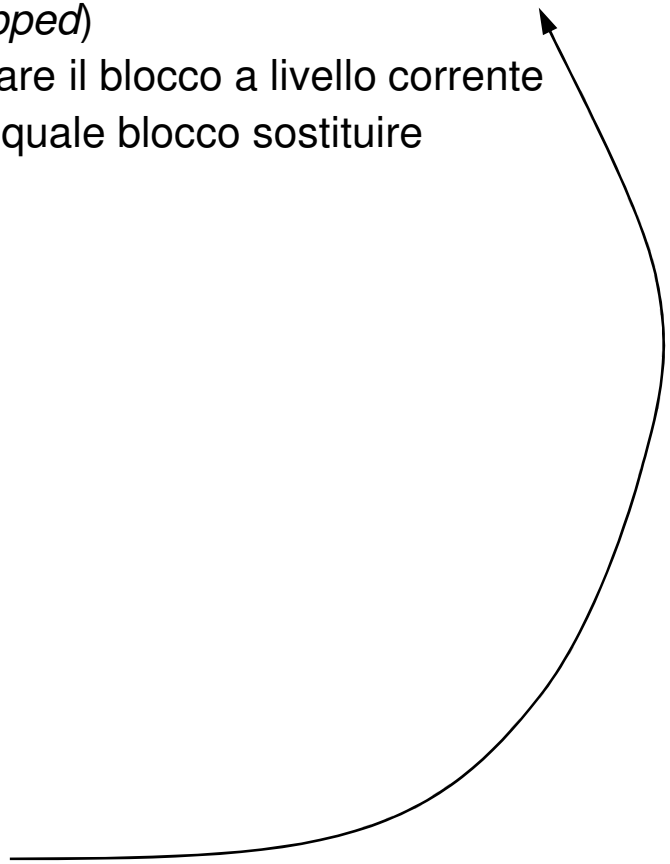
# Memorie cache

Parametri di progettazione delle cache:

1) **Piazzamento** del blocco (o funzione di traduzione o mapping): dove può essere allocato il blocco al livello corrente (ovunque/ *fully associative*, solo in alcune linee/*n way associative*, solo in una linea/ *direct mapped*)
2) **Identificazione** del blocco: come si può ritrovare il blocco a livello corrente
3) **Rimpiazzamento** del blocco: come si sceglie quale blocco sostituire
4) Strategia di **scrittura**

**Classificazione dei miss**

- **Inevitabili**
  Primo accesso a un blocco
- **Capacità**
  Miss che accadono quando si accede a un blocco sostituito
- **Conflitto**
  Miss che accadono poiché i blocchi sono scartati a causa di una specifica strategia di "set-mapping"



# Cache suddivise

- I processori hanno tutti ormai cache divise in
- **cache dati**
- **cache di codice**

- Permette l'accesso contemporaneo al codice(fetch) e ai dati (ld/st)

- Sono cache di primo livello (interne al micro).
- Molti processori prevedono anche cache di secondo livello

- esempio di cache L2 (**Motorola MCM64AF2**)

- Cache di 256 KB, direct mapped:
  - 8 chip da 32 KB ciascuno per i dati
  - 1 chip da 8KB per tag (in realtà è da 32KB anch'esso, ma i pin A13-A14 sono fissi a 0: usato per 1/4 della sua capacità)
- Linee da 32 byte $\Rightarrow$ 8192 set (256K / 32 ovvero 256 K-parole a parallelismo 32 bit)

  5 bit di offset (A0-A4), 13 bit di index di set (A5-A17)

# Tecniche di scrittura

- Scrittura sulle cache
- Gestione della scrittura più complessa
  - Le letture si fanno contemporaneamente al confronto dei tag. Le scritture no (e quindi le scritture sono spesso più lente)

- Se si è in presenza di un "hit"
- **write-through** (store through) si scrive sempre non solo nella cache ma anche nella memoria centrale
- **write-back** (store in, copy-back) si scrive sulla memoria centrale solo quando il blocco deve essere rimpiazzato (quindi si scrive in cache e poi si aggiorna il blocco intero in memoria centrale quando lo si sostituisce)
- In caso di miss si rimpiazzano i blocchi ?
  - Si - **write-allocate** (normalmente usato con il write-back); si alloca sulla cache e poi si modifica
  - No - **no-write-allocate** (normalmente usato con il write-through) si scrive solo nella memoria gerarchicamente più bassa

- nel Pentium esiste un pin esterno per specificare di tipo WT o WB; per usare la politica WB è necessario predisporre una politica e i corrispondenti meccanismi hardware per gestire la coerenza dei dati in memoria e sulle cache in sistemi multiprocessore

# Cache memory

**Table 4.3 Cache Sizes of Some Processors**

| Processor | Type | Year of Introduction | L1 cache[a] | L2 cache | L3 cache |
|---|---|---|---|---|---|
| IBM 360/85 | Mainframe | 1968 | 16 to 32 KB | — | — |
| PDP-11/70 | Minicomputer | 1975 | 1 KB | — | — |
| VAX 11/780 | Minicomputer | 1978 | 16 KB | — | — |
| IBM 3033 | Mainframe | 1978 | 64 KB | — | — |
| IBM 3090 | Mainframe | 1985 | 128 to 256 KB | — | — |
| Intel 80486 | PC | 1989 | 8 KB | — | — |
| Pentium | PC | 1993 | 8 KB/8 KB | 256 to 512 KB | — |
| PowerPC 601 | PC | 1993 | 32 KB | — | — |
| PowerPC 620 | PC | 1996 | 32 KB/32 KB | — | — |
| PowerPC G4 | PC/server | 1999 | 32 KB/32 KB | 256 KB to 1 MB | 2 MB |
| IBM S/390 G4 | Mainframe | 1997 | 32 KB | 256 KB | 2 MB |
| IBM S/390 G6 | Mainframe | 1999 | 256 KB | 8 MB | — |
| Pentium 4 | PC/server | 2000 | 8 KB/8 KB | 256 KB | — |
| IBM SP | High-end server/ supercomputer | 2000 | 64 KB/32 KB | 8 MB | — |
| CRAY MTA[b] | Supercomputer | 2000 | 8 KB | 2 MB | — |
| Itanium | PC/server | 2001 | 16 KB/16 KB | 96 KB | 4 MB |
| SGI Origin 2001 | High-end server | 2001 | 32 KB/32 KB | 4 MB | — |

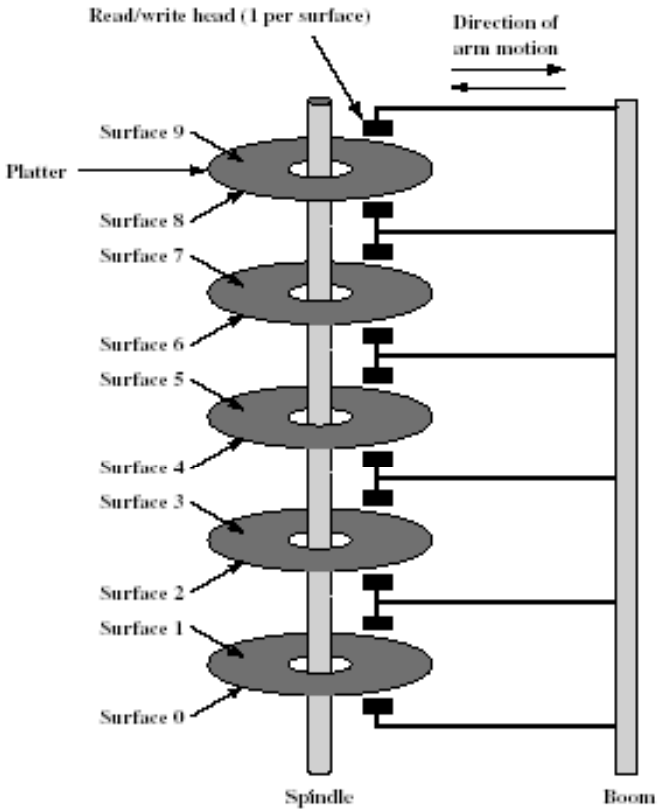
[a] Two values seperated by a slash refer to instruction and data caches

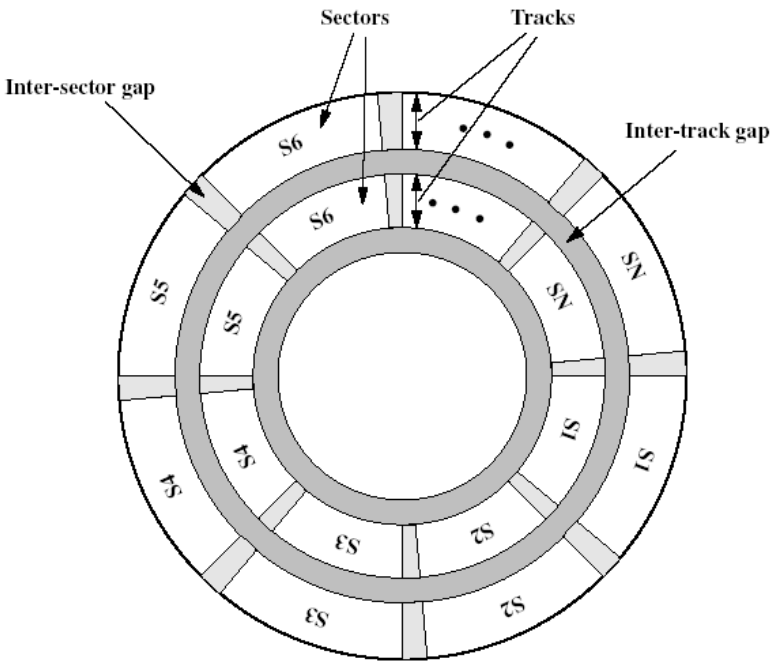
[b] Both caches are instruction only; no data caches

# Memorie a disco magnetico

- **Memorie secondarie**:
- In un calcolatore esistono diverse memorie secondarie (o memorie di massa) ad alta capacità bassi costi e non volatili
- **Ad es:**
- **Memoria a disco magnetico**
- L'elemento di memoria è un disco ricoperto di materiale magnetico, su cui esistono una serie di *tracce* concentriche.
- L'unità di memoria può essere costituita da più dischi: in tal caso essi sono connessi ad un unico asse e ruotano a velocità costante. Ogni superficie (*faccia*) è dotata di una testina in grado di muoversi radialmente fin sulla traccia desiderata. Le varie testine si muovono di solito in maniera solidale.
- L'insieme delle tracce ad uguale distanza dal centro poste su facce diverse è denominato *cilindro*.
- Ogni traccia contiene la stessa quantità di informazione, ed è quindi caratterizzata da una densità crescente andando verso il centro. Ogni traccia è organizzata in *settori*, corrispondenti all'unità di trasferimento.
- Sono essenzialmente di 2 tipi:
- · a disco rigido (hard disk)
- · a disco flessibile (floppy disk).

# Hard disk


Read/write head (1 per surface)
Direction of arm motion
Surface 9
Platter
Surface 8
Surface 7
Surface 6
Surface 5
Surface 4
Surface 3
Surface 2
Surface 1
Surface 0
Spindle
Boom


Sectors
Tracks
Inter-sector gap
Inter-track gap

# Hard disk

- Hard-Disk NEC D2257: dischi e testine sono sigillati in una scatola a tenuta. Si compone di 5 dischi da 8'', su cui vengono utilizzate 8 facce, ognuna con la sua testina.
- Caratteristiche

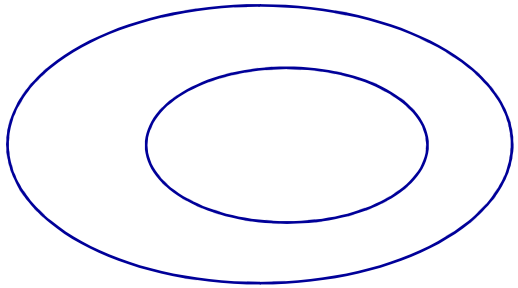
| | |
|---|---|
| #tracce per superficie | 1024 |
| max densità di memorizzazione | 9420 bit/inch |
| capacità per traccia | 20,480 byte |
| capacità totale | 167.7 Mbyte |
| velocità di rotazione | 3510 giri/min |
| Average Seek Time | 20 ms |

- Il tempo di accesso tA è determinato da:
- · tS: tempo per posizionare la testina sulla traccia opportuna (seek time, o tempo di ricerca); è
- nullo se ogni traccia ha la sua testina;
- · tL: tempo per posizionare la testina sul dato, all'interno della traccia ( latency time)
- · tD: tempo per leggere serialmente i dati ( data-transfer time, o tempo di trasferimento).
- Si ha quindi che
- **tA= tS+ tL+ tD**

- Supponendo di voler trasferire un blocco di 1kbyte di memoria, il tempo di accesso è pari a (20 +
- 8,55 + 0,85) msec = 29,4 msec

# ACCESSO AL DISCO

- I dischi sono organizzati in anelli concentrici o TRACCE (da 500 a 2000 tracce) separati da intervalli vuoti GAP) per l'allineamento delle testine
- la densita' (bit x cm) credce con le tracce interne per avere lo stesso numero di bit per traccia.
- In ogni traccia ci stanno tanti settori che costituiscono un blocco di fati dam leggere o scrivere (ogni traccia da 10 a 100 settori)
- dipende dalla formattazione del s.o.
- Es una traccia con 30 settori di 600 byte di cui 512 dati e altri di controllo
- caratteristiche
- **testina** : fissa ( 1 per traccia), mobile (1 per superf)
- **lati**: densita' singola o doppia
- **Piatti**: singolo, multiplo
- **meccanica testina**: contatto (floppy), aerodinamico (winchester)



# ACCESSO AL DISCO

Tempi di ritardo

0) attesa del dispositivo....

1) **tempo di ricerca** (seek) per posizionare la testina sulla traccia

Ts=mxn+s

m costante del drive del disco, n numero di tracce attraversate, s di avviamento meccanico

m=0,3-0,1 ms  s= 20-3 ms

2) **latenza rotazionale**: per arrivare al settore corretto

hd di solito 3600rpm,1 rotazione in 16,7 ms ritado medio 8.3ms

Tr=1/2r  r giri al secondo

floppy 300-600 rpm, ritardo medio 100-200 ms

1+2 si dicono tempo di accesso

3) **tempo di trasferimento**

Tt= B/rN B byte da trasferire, N n di byte per traccia

T= Ts+Tr+Tt

# ACCESSO AL DISCO

**Organizzazione sequenziale**, il file e' diviso in settori e tracce adiacenti

- si consideri un disco con Ts medio dichiarato di 20ms con velocità di trasferimento di 1MB/s e tracce di 32 settori da 512 byte.
- Si vuole leggere un file da 128KB

- 128KB--> 256 settori--> 8 traccie adiacenti

- per leggere la prima traccia Ts=20ms
- Tr=8.3ms (medio a 3600 rpm)
- lettura dei primi 32 settori di una traccia Tt=16.7 msec
- T=20+8.3+16.7=45ms
- se le altre tracce seguono non c'e piu' da aggiungere Ts ma solo 8.3+16.7=25
- cosi' Ttot=45+7x25=220ms=**0,22sec**

**Organizzazione random**

- per ogni settore T=20+8.3+0.5= 28.8 msec
- Ttot= 28.8 x 256 =7373ms= **7.37sec**

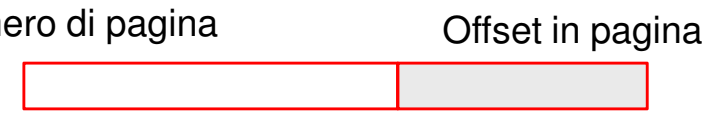
# Memoria virtuale

- Tecnica gestita dal sistema operativo per fare in modo che il processore veda virtualmente tutto lo spazio di indirizzamento malgrado la memoria centrale sia limitata.

- **Rilocazione**: il Sistema operativo rialloca gli indirizzi dei dati e del codice in zone di memoria definite in base a politiche fissate
- **Segmentazione** Il sistema operativo segmenta la memoria in segmenti logici che alloca in memoria centrale, anche parzialmente sovrapposti. Gestiti dalla MMU a tempo di esecuzione sommando il registro di segmento al registro delle pagine

## La Memoria Virtuale è:

- **Paginazione**. Lo spazio di indirizzamento della memoria centrale viene diviso in blocchi di dimensione fissata (pagine fino a 4M di dimensione) contigui.
- L'indirizzo può sempre essere diviso in un indice di pagina e in un offset rispetto alla pagina
-




- Non tutte le pagine sono allocate in memoria ma lo sono virtualmente. Se non sono in memoria sono sulla memoria secondaria e vengono ricopiate in memoria centrale quando richiesto (ad esempio da HD a RAM)

# Memoria virtuale

**1) Identificazione delle pagine**

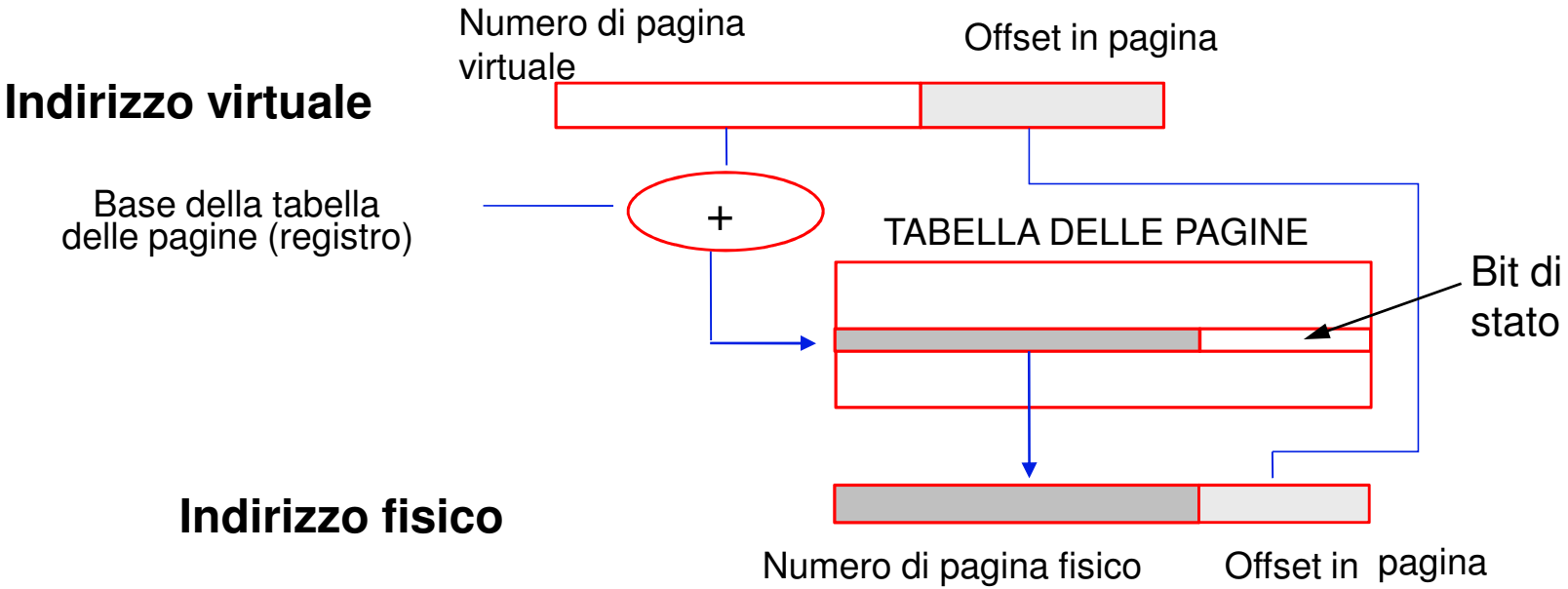
- Traduzione dell'indirizzo (virtuale -> fisico)
- Indiretto con  tabelle delle pagine
- Traduzione "cached" in un buffer di traslazione che tiene memoria della riallocazione

**2) Rimpiazzamento** delle pagine

- Quasi sempre del tipo Least Recently Used
- "Working set"

**3) Strategie di scrittura**

- Write-back (usando un bit di pagina sporca)



# Memoria Virtuale

- La CPU esegue il programma in cui è codificato **l'indirizzo virtuale** (che copre tutto lo spazio di indirizzamento possibile) deciso dal compilatore.

- La CPU prima di usare l'indirizzo per un accesso alla memoria (cache oppure memoria centrale) traduce l'indirizzo virtuale in fisico. Viene gestito dalla MMU che accede alla memoria centrale ad un indirizzo dove e' allocata la tabella delle pagine, usa l'indirizzo (parte del) virtuale come puntatore
- Se alla locazione indicata c'e' l'indirizzo fisico (bit di stato valid) lo usa per accedere alla memoria (prima cache e poi memoria).
- Se non c'e' (bit dirty) l'indirizzo fisico corrispondente inizia una fase di page fault che chiama il SO il quale rimpiazza la pagina da memoria secondaria a memoria centrale e scrive l'indirizzo fisico sulla tabella delle pagine
- Due accessi. Accesso in memoria per la tabella e accesso per il dato

- La tabella delle pagine può essere troppo lunga (ad esempio indirizzo a 32 bit con pagine di 4Kbyte, tabella delle pagine contiene 2^20 indirizzi), occupa troppa memoria. Quindi un indirizzamento ad una tabella che punta alle tabelle delle pagine (due livelli di tabella delle pagine ognuno da 2^10 indirizzi)→ tre accessi in memoria
- Translation Lookaside Buffer
- Cache con indirizzi virtuali

# Pentium II

- Pentium II Hardware per la segmentazione e paginazione

- **Unsegmented unpaged**
  - virtual address = physical address
  - Low complexity
  - High performance

- **Unsegmented paged**
  - Memory viewed as paged linear address space
  - Protection and management via paging
  - **Berkeley UNIX**

- **Segmented unpaged**
  - Collection of local address spaces
  - Protection to single byte level
  - Translation table needed is on chip when segment is in memory

- **Segmented paged**
  - Segmentation used to define logical memory partitions subject to access control
  - Paging manages allocation of memory within partitions
  - **Unix System V**