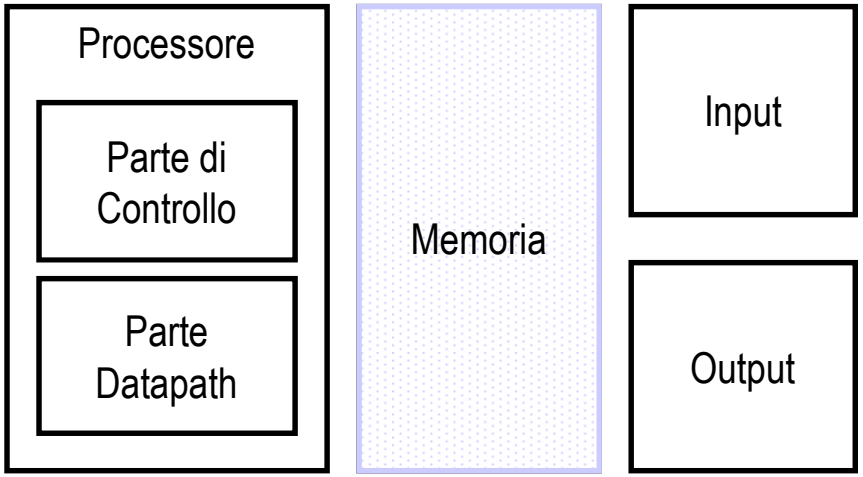
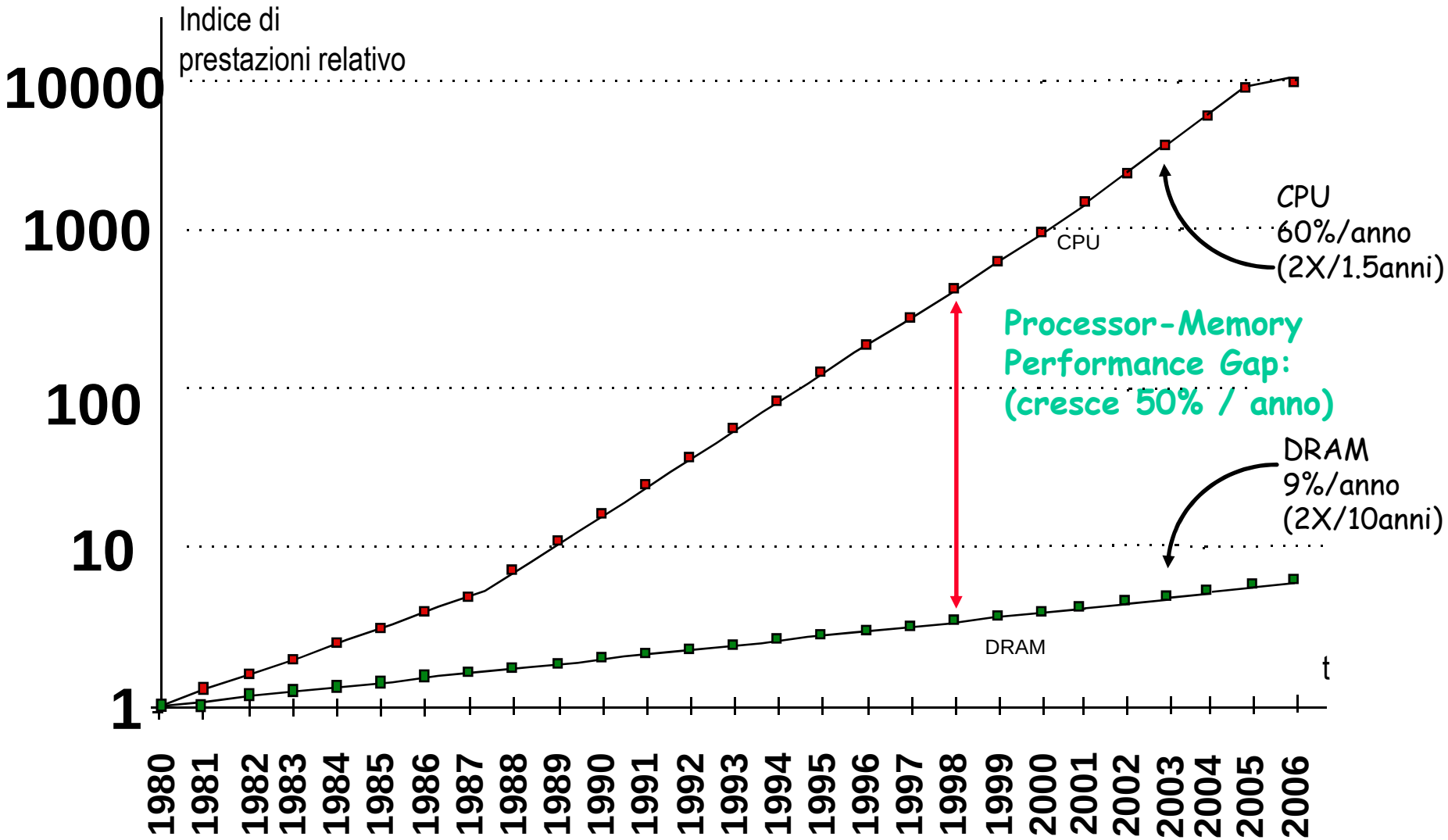
Lezione 17
MECCANISMI DI MIGLIORAMENTO DELLE PRESTAZIONI (1)
Introduzione alla memoria cache

- **Gerarchia di Memoria**
- **Memoria Cache**



Gap Prestazioni Processore-Memoria

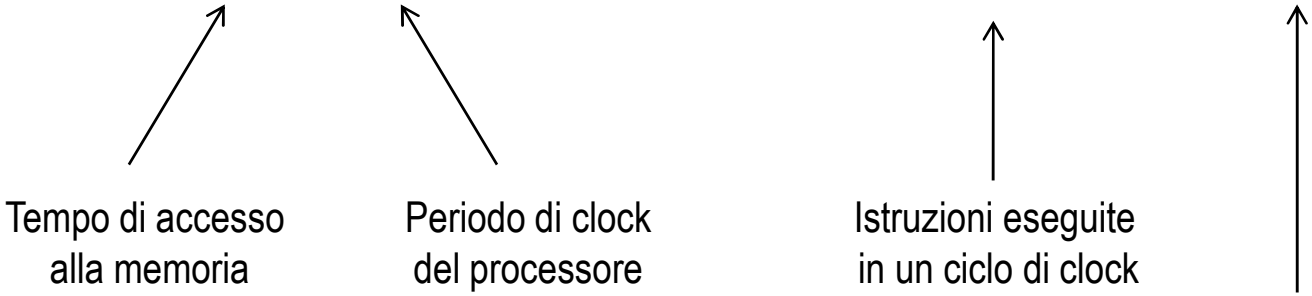


→ Necessita' di nuove architetture per le memorie

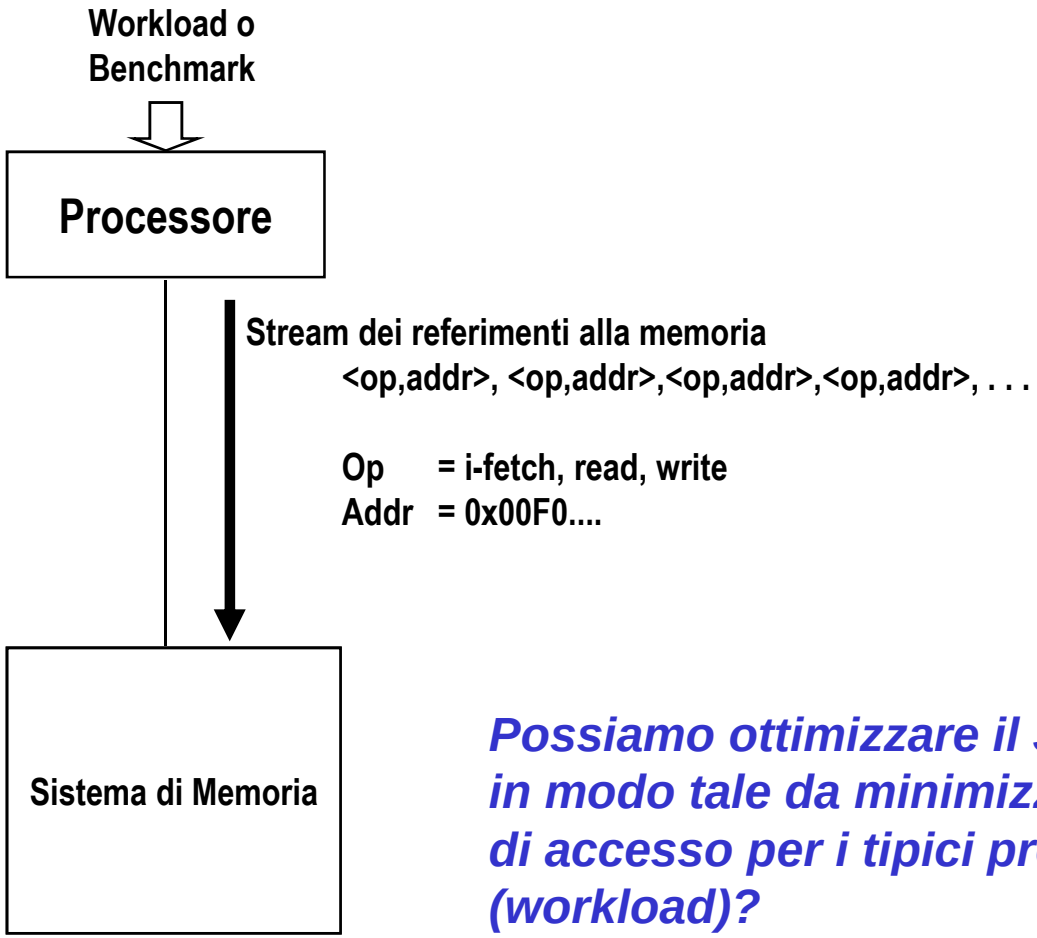
Gap Processore-Memoria: un esempio

Valutiamo il gap Processore-Memoria in termini di istruzioni per eseguite dal processore durante 1 accesso in memoria

Alpha 7000: 340 ns/5.0 ns = 68 clk x 2istr/clk →136 istr.
Alpha 8400: 266 ns/3.3 ns = 80 clk x 4istr/clk →320 istr.
Alpha xxxx: 180 ns/1.7 ns = 108 clk x 6istr/clk →648 istr.



"L'Arte della Progettazione del Sistema di Memoria"



Possiamo ottimizzare il Sistema di Memoria in modo tale da minimizzare il tempo medio di accesso per i tipici programmi che utilizziamo (workload)?

Principio di Localita' dei Programmi

- Un riferimento in memoria tende ad essere ripetuto dopo poco tempo

  → LOCALITA' TEMPORALE

  - Esempio:
    - 0008
    - 1000
    - 0008
    - 2000
    - 0008

- Un riferimento in memoria tende ad essere a locazioni vicine a quelle usate recentemente

  → LOCALITA' SPAZIALE

  - Esempio:
    - 0004
    - 0008
    - 000C
    - 0010
    - 0014

Implicazioni del Principio di Localita'

1) Posso utilizzare piccole memorie veloci per mantenere i riferimenti in memoria piu' utilizzati dal processore

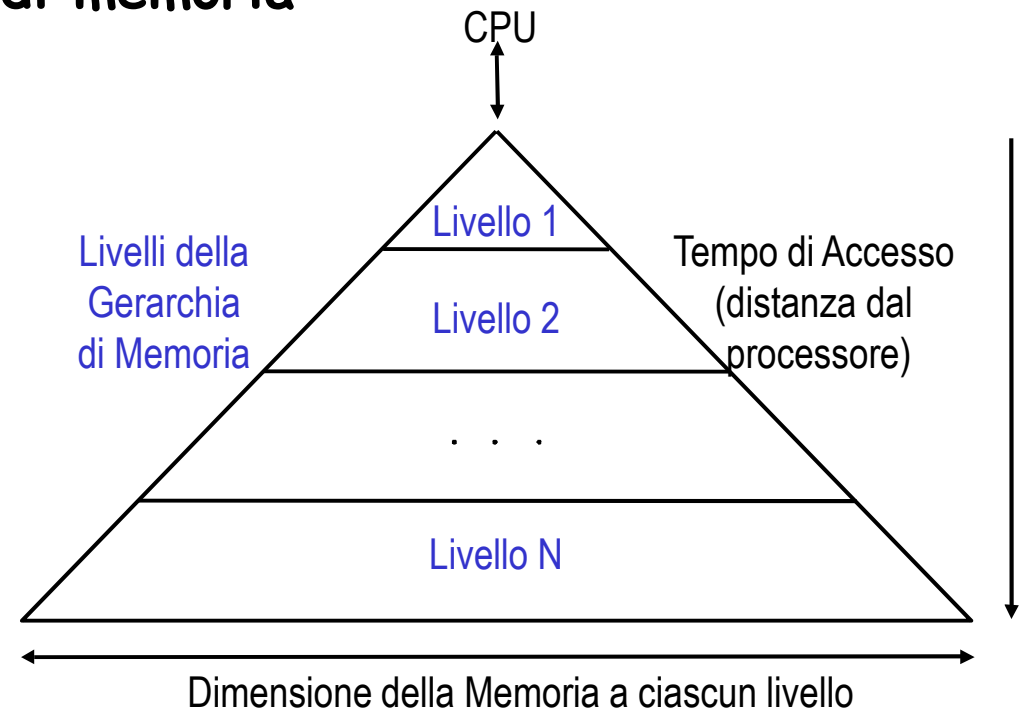
→ MEMORIA CACHE

2) Se i riferimenti utilizzati non possono essere soddisfatti nella memoria veloce

→ creo un secondo livello di memoria un po' piu' grande e un po' meno veloce

3) Posso reiterare questo ragionamento per N livelli di memoria

→ GERARCHIA DI MEMORIA



Gerarchia di Memoria

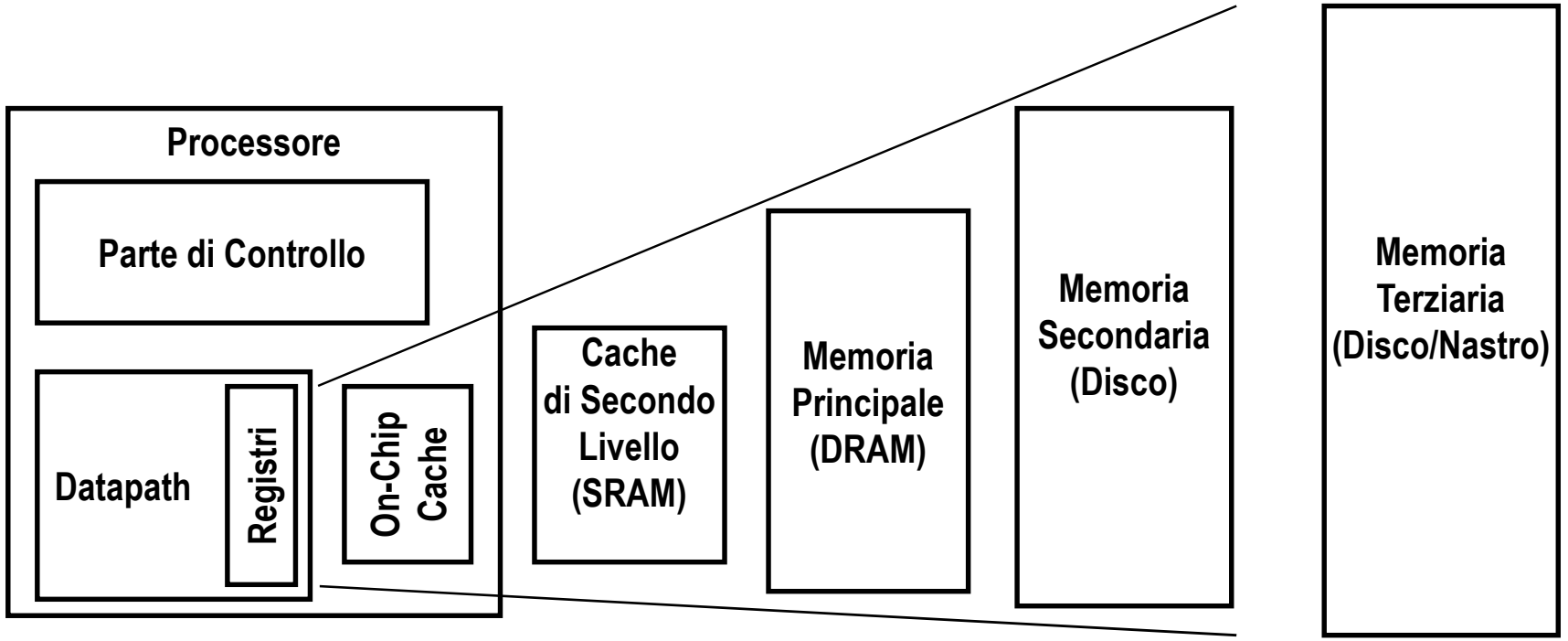
- Presenta al processore una quantita' di memoria pari a tutta quella disponibile all'ultimo livello (realizzata con la tecnologia piu' economica)
- Fornisce un tempo di accesso che si avvicina a quello del primo livello (in cui posso usare la tecnologia piu' veloce possibile). Es. Con 2 soli livelli:
  - Primo Livello: Cache, tecnologia SRAM
  - Secondo Livello: Memoria Principale, tecnologia DRAM
- Matematicamente:
  - h= probabilita' di trovare un dato nel primo livello
    - Maggiore LOCALITA' comporta h→1
  - m= probabilita' di trovare un dato nel livello successivo
  - t_h= tempo di accesso al dato dal primo livello
  - t_m= tempo di accesso al dato nel livello successivo

$\rightarrow t_a = h*t_h + m*t_m$ se $h \rightarrow 1$ allora $t_a \rightarrow t_h$

Terminologia
blocco: insieme di byte a locazioni contigue ed indirizzi multipli della dimensione del blocco
hit: il riferimento puo' essere soddisfatto nel livello di memoria immediatamente successivo
miss: il riferimento richiede un accesso a livelli di memoria oltre il successivo (piu' lenti)

Organizzazione gerarchica della memoria

- Sfruttando il principio di localita' e' possibile:
  - Presentare all'utente tutta la memoria presente al livello della tecnologia meno costosa (es. Disco o nastro)
  - Fornire la velocita' di accesso offerta dalla tecnologia piu' veloce

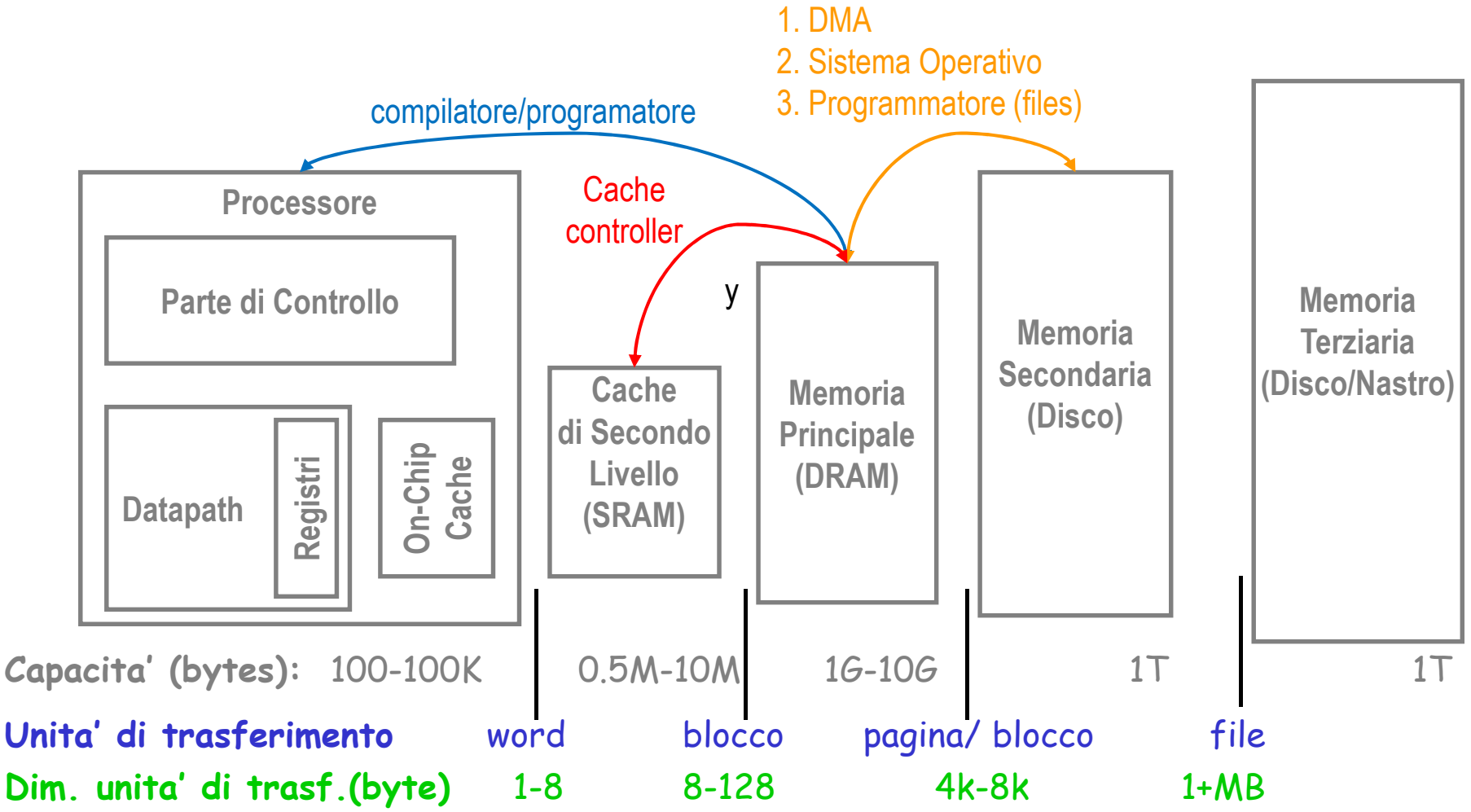


Velocita' (ns):	0.3-1	10-20	100	10,000,000 (10 ms)	10,000,000,000 (10 s)
Capacita' (bytes):	100-100K	0.5M-10M	1G-10G	1T	1T
Costo (EURO/MB):	40	0.5	0.01-0.02	0.0001	0.0001

(Anno 2008)

Unita' di trasferimento ai vari livelli della gerarchia

- Ai livelli piu' alti la dimensione delle unita' di trasferimento e' piu' piccola (trasferimenti piu' veloci)



Memoria Cache

- Scopo della cache e' di memorizzare (per un rapido accesso) copie di blocchi di memoria principale
  - Funzionamento: ogni volta che faccio accesso ad una locazione di memoria, lascio in cache una copia del corrispondente blocco
  - Conseguenza: per il principio di localita' la cache tende a memorizzare i blocchi piu' usati dal processore (working set)
- Normalmente si ha:
  - B = dimensione del Blocco di cache in byte = 2^b
  - C = dimensione della Cache di cache in byte = 2^c
  - M = dimensione della Memoria in byte = 2^m

M >> C

ovvero

$N_M >> N_C$

N_M = Numero di blocchi della Memoria = M/B
N_C = Numero di blocchi della Cache = C/B

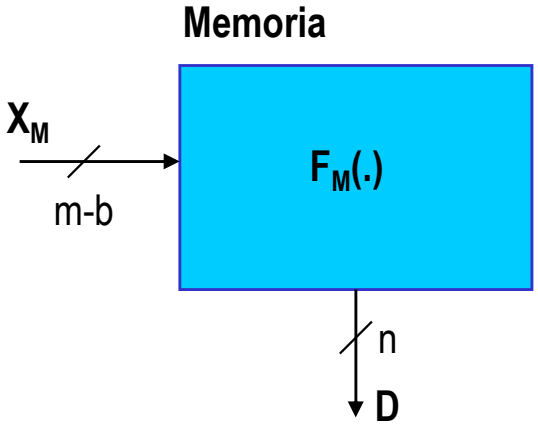
Inoltre, ad un certo istante per lo working set:

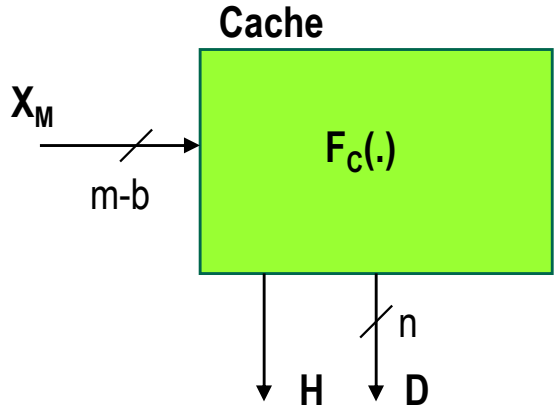
$W_C \subset W_M$

W_C = insieme dei blocchi del mio programma presenti in cache
W_M = insieme dei blocchi (totali) del mio programma in memoria

Modello "Black-Box" della Cache (1)

X_M = indirizzo del blocco di memoria ($0 <= X_M < N_M$) ≡ X / B (X indirizzo al byte)





F_M = Funzione di prelievo del blocco dalla memoria

$D = F_M(X_M)$

F_C = Funzione di prelievo del blocco dalla cache

$D = F_C(X_M) = F_M(X_M)$ se H=1

D=niente se H=0

H = Segnale di Hit

1 se il blocco e' in cache

→ diro' anche che ho avuto un hit

0 se il blocco non e' in cache

→ diro' che ho avuto un miss

F_M : MEM [.]

F_C : ...dipende

Funzionamento della Cache (1)

- Come faccio a sapere se il blocco e' in cache?
  - Definisco una "Funzione di Hit" F_H che vale

  $H = F_H(X_M)$

- Se il blocco e' in cache, come faccio a trovarlo?
  - Per prelevare un blocco dalla memoria facevo $D = F_M(X_M) = MEM[X_M]$
  - Analogamente, per la cache vogliamo recuperare il dato ad un indirizzo X_S di (un array di memoria) cache (con $0<=X_S<N_C$)

  $D = F_{C.D}(X_S) = CACHE[X_S].D$

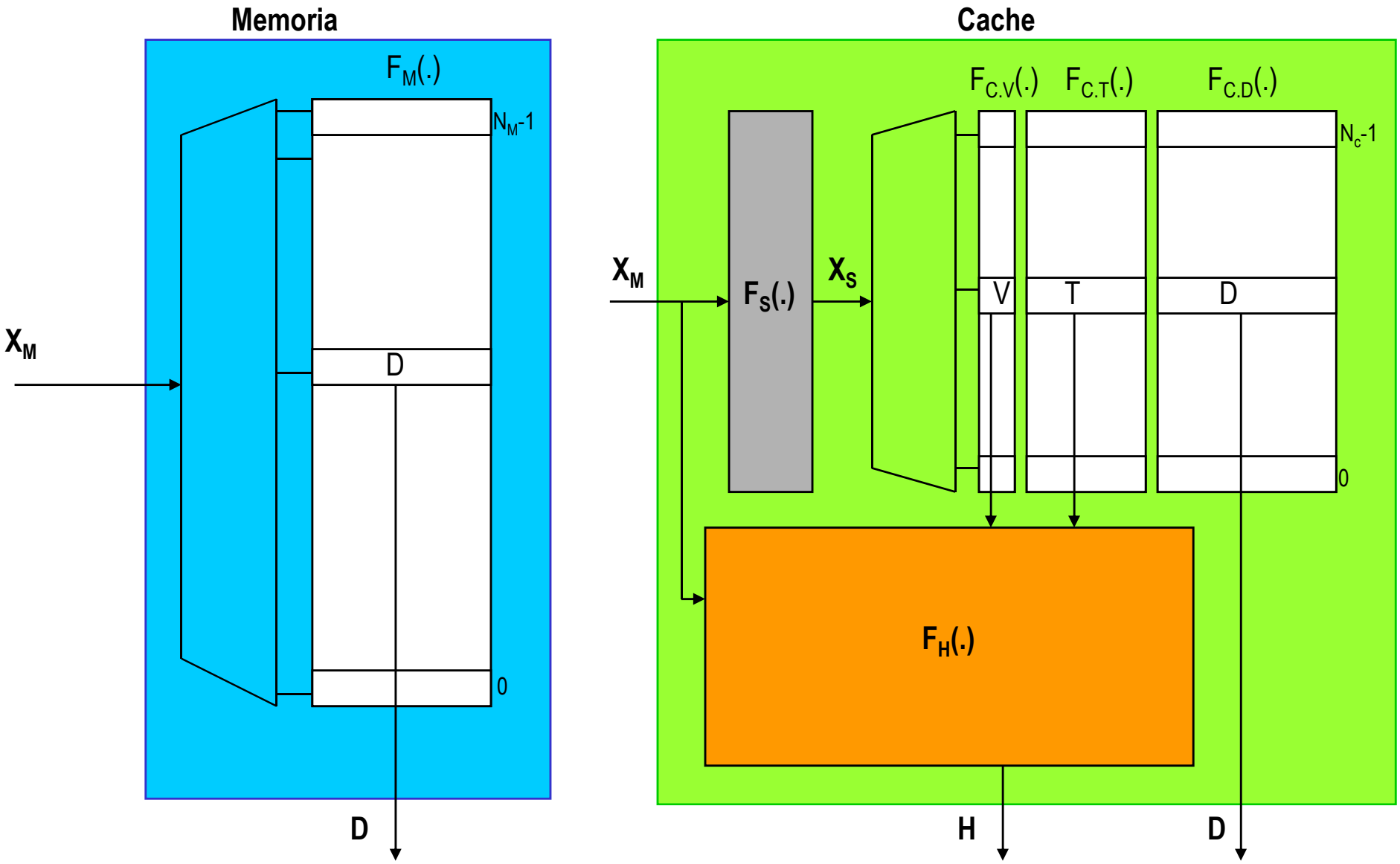
  - Che relazione c'e' fra X_M e X_S (ricordare che $X_S<N_C<<N_M$) ?
    Per essere sicuri che il dato D (che in memoria e' ad indirizzo X_M) sia recuperabile all'indirizzo di cache X_S,
    occorre memorizzare accanto al dato un etichetta (o "tag"):

  $T = F_{C.T}(X_S) = CACHE[X_S].T$

  - Inoltre per essere sicuri che a una data coppia <T,D> corrisponda un dato effettivamente inserito in cache,
    si aggiunge un bit di validita'

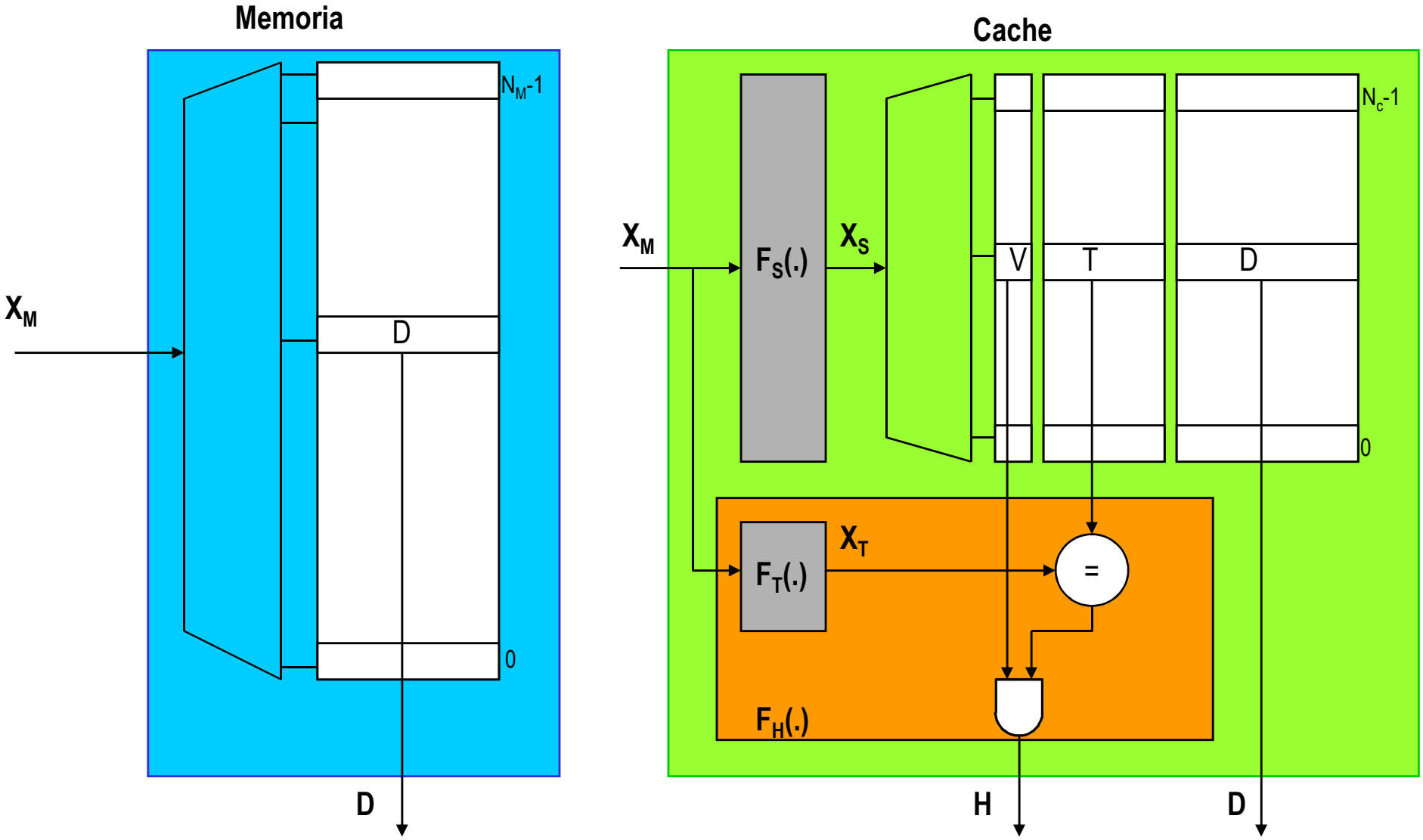
  $V = F_{C.V}(X_S) = CACHE[X_S].V$

Modello "Black-Box" della Cache (2)



F_S: ? F_H: ?

Modello "Black-Box" della Cache (3)



F_S e F_T CARATTERIZZANO IL TIPO DI CACHE

Funzionamento della Cache (2)

- Affinche' questo meccanismo funzioni e' necessario che si possano ricavare in maniera univoca da X_M
  - X_S per indirizzare il dato D dentro la memoria cache
  - X_T per confrontarlo con il tag T memorizzato nella cache

  - L'individuazione del blocco in cache viene quindi fatta combinando due informazioni:
    1) Etichetta (o "tag") associata al blocco:

       $X_T = F_T(X_M)$

    2) Indirizzo di localizzazione in cache:

       $X_S = F_S(X_M)$

- Allora

  $F_H : (X_T == T \;\&\&\; V == 1 \;?\; 1 : 0)$

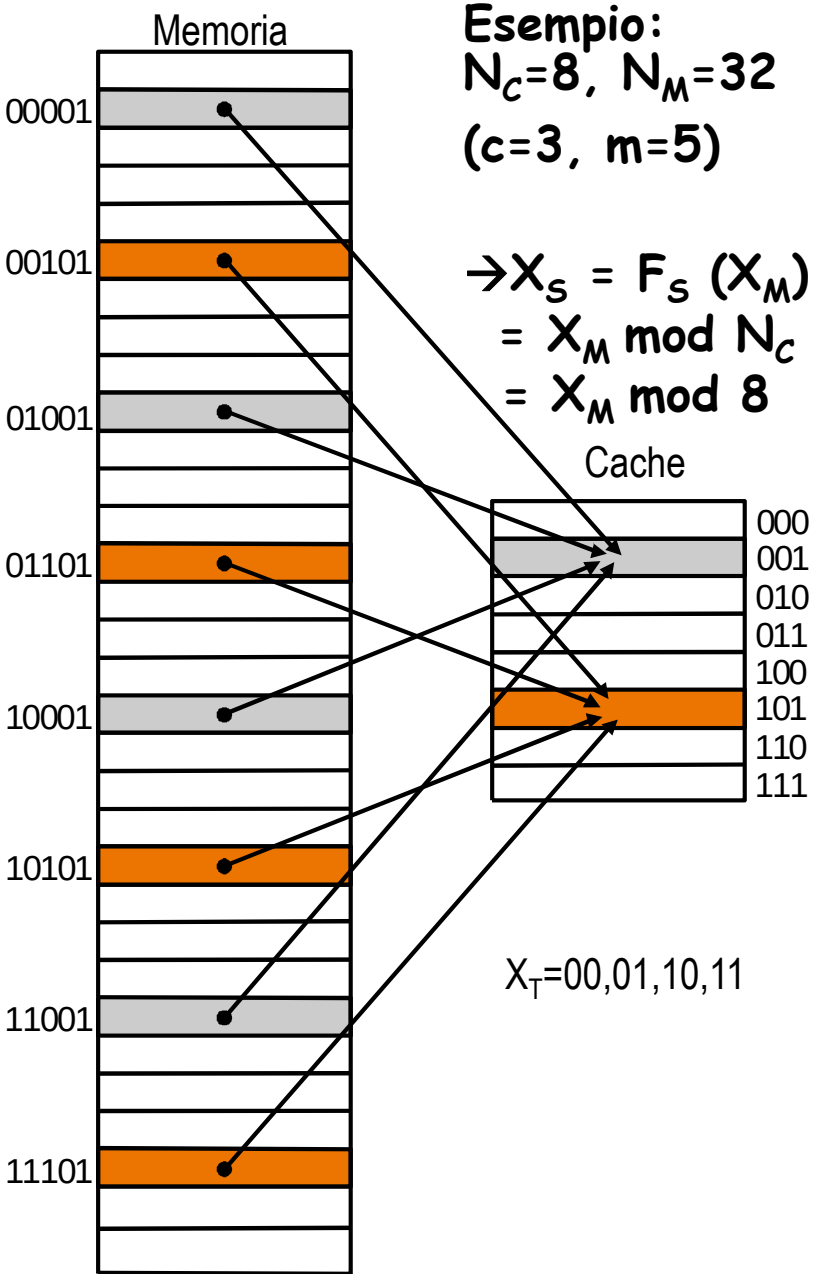
  ovvero

  $F_H : (F_T(X_M) == T \;\&\&\; V == 1 \;?\; 1 : 0)$

F_T e F_S nel caso di cache ad ACCESSO DIRETTO

F_T : (div N_C)

F_S : (mod N_C)



Schema logico di una cache ad accesso diretto

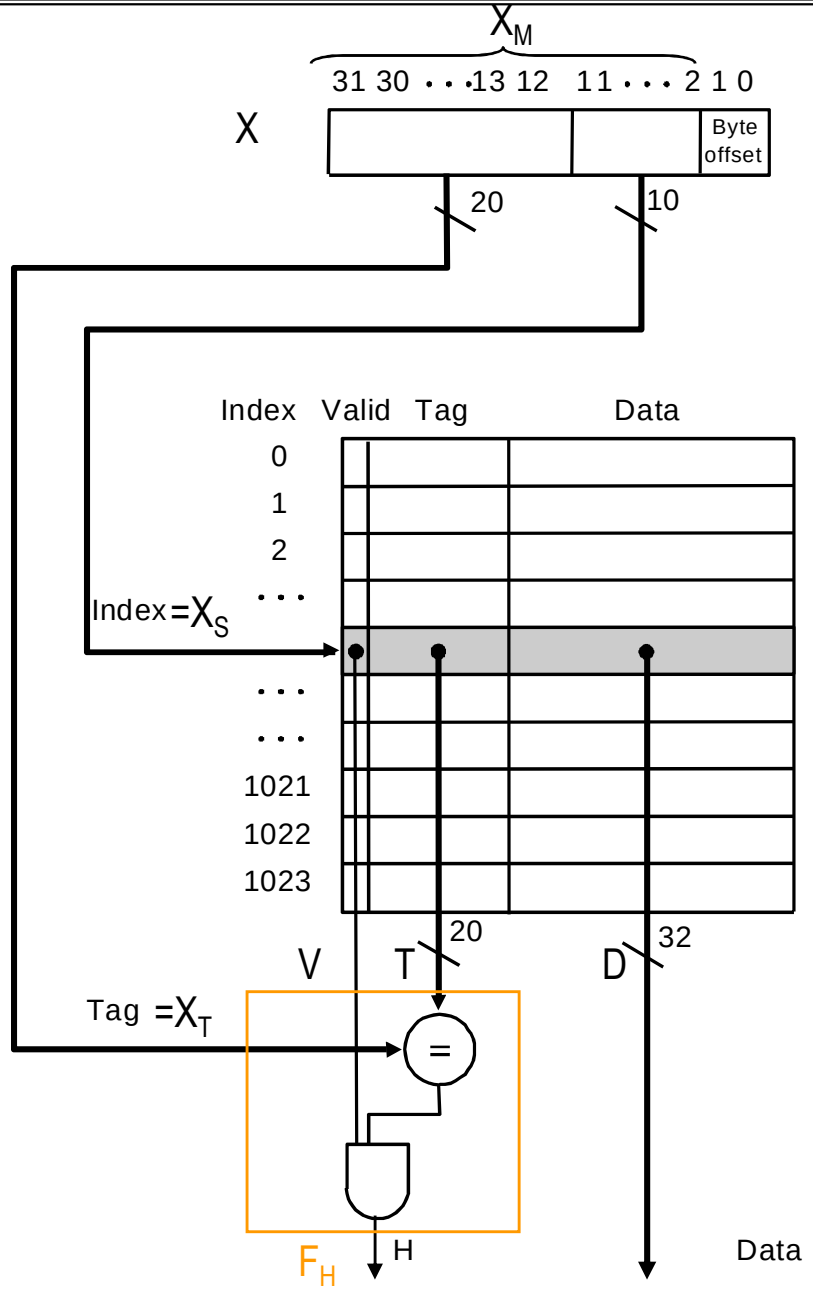
- B = 4 byte
- C = 4 Kbyte
- M = 4 Gbyte

X indirizzo al byte

$X_M = X / B$
$= X / 2^2$

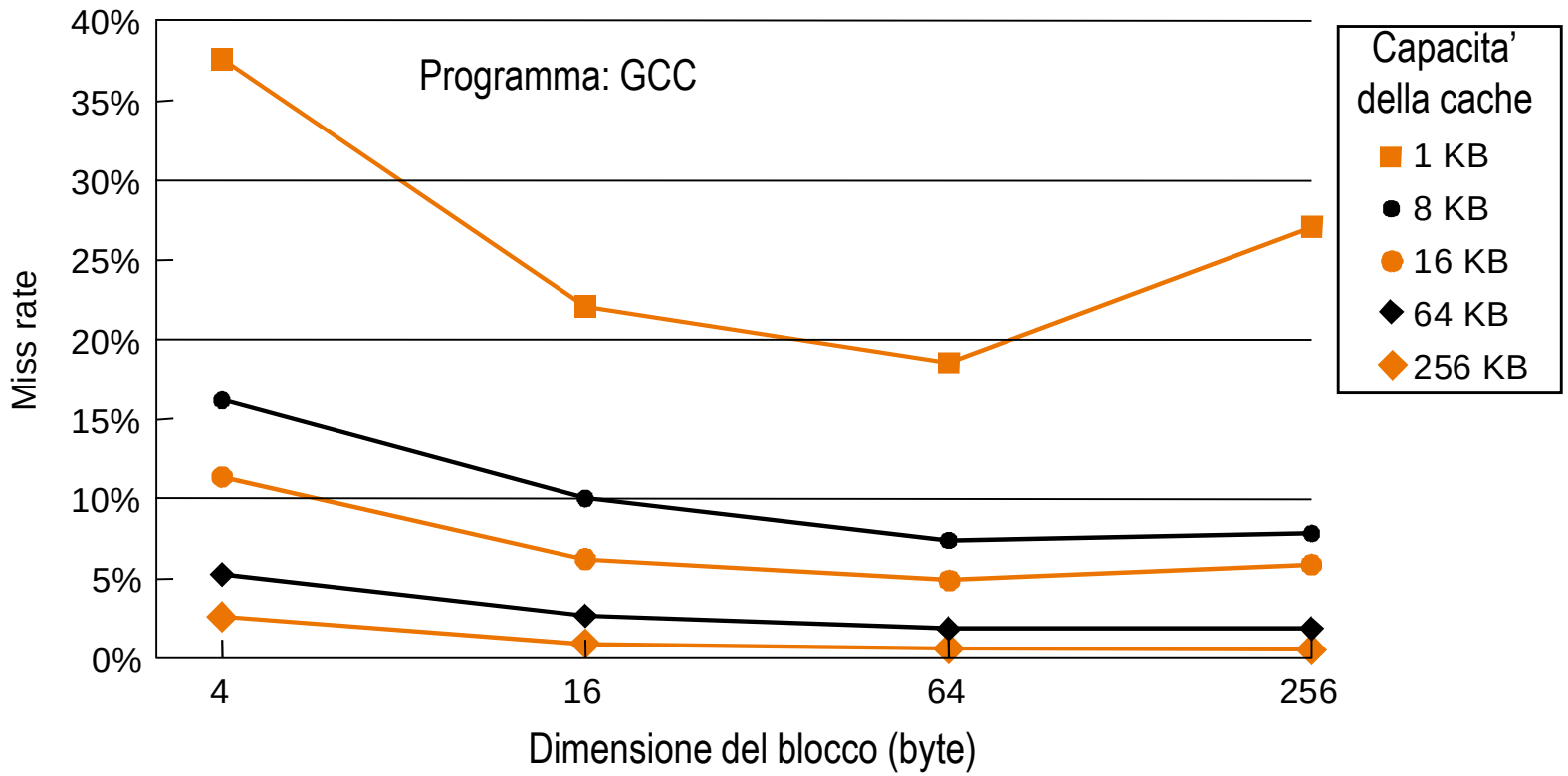
$X_T = X_M / N_C$
$= X_M / (C/B)$
$= X_M / 2^{10}$

$X_S = X_M \bmod N_C$
$= X_M \bmod C/B$
$= X_M \bmod 2^{10}$



Dipendenza del Miss Rate dalla dimensione del blocco

- **Aumentando la dimensione del blocco il miss rate tende a diminuire**



Program	Block size	Miss rate
gcc	4	5.4%
	16	1.9%
spice	4	1.2%
	16	0.4%

Cache a blocchi piu' grandi: sfrutto la localita' spaziale

- B = 16 byte
- C = 64 Kbyte
- M = 4 Gbyte

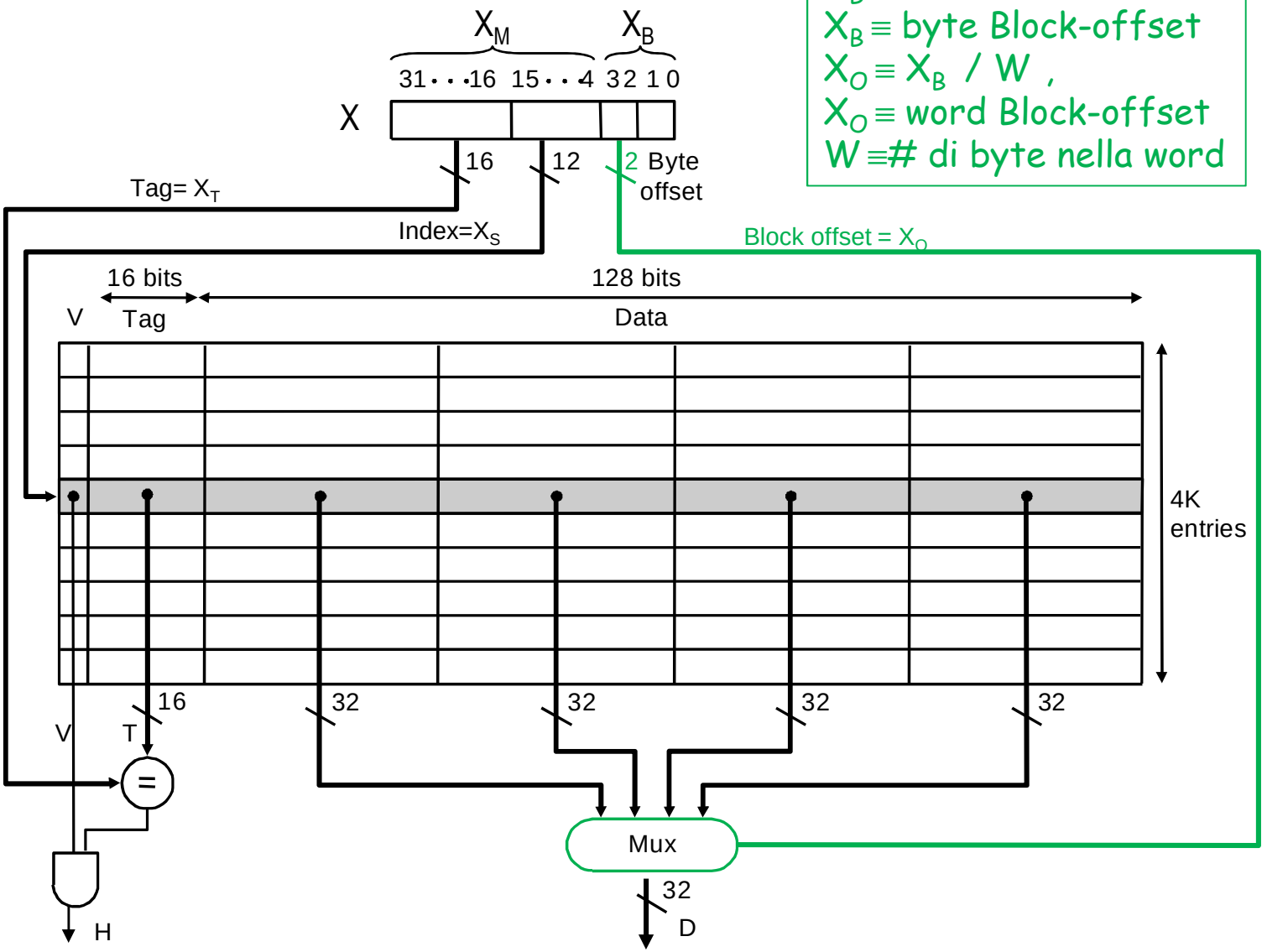
X indirizzo al byte

$X_M = X / B$
$= X / 2^4$

$X_T = X_M / N_C$
$= X_M / (C/B)$
$= X_M / 2^{12}$

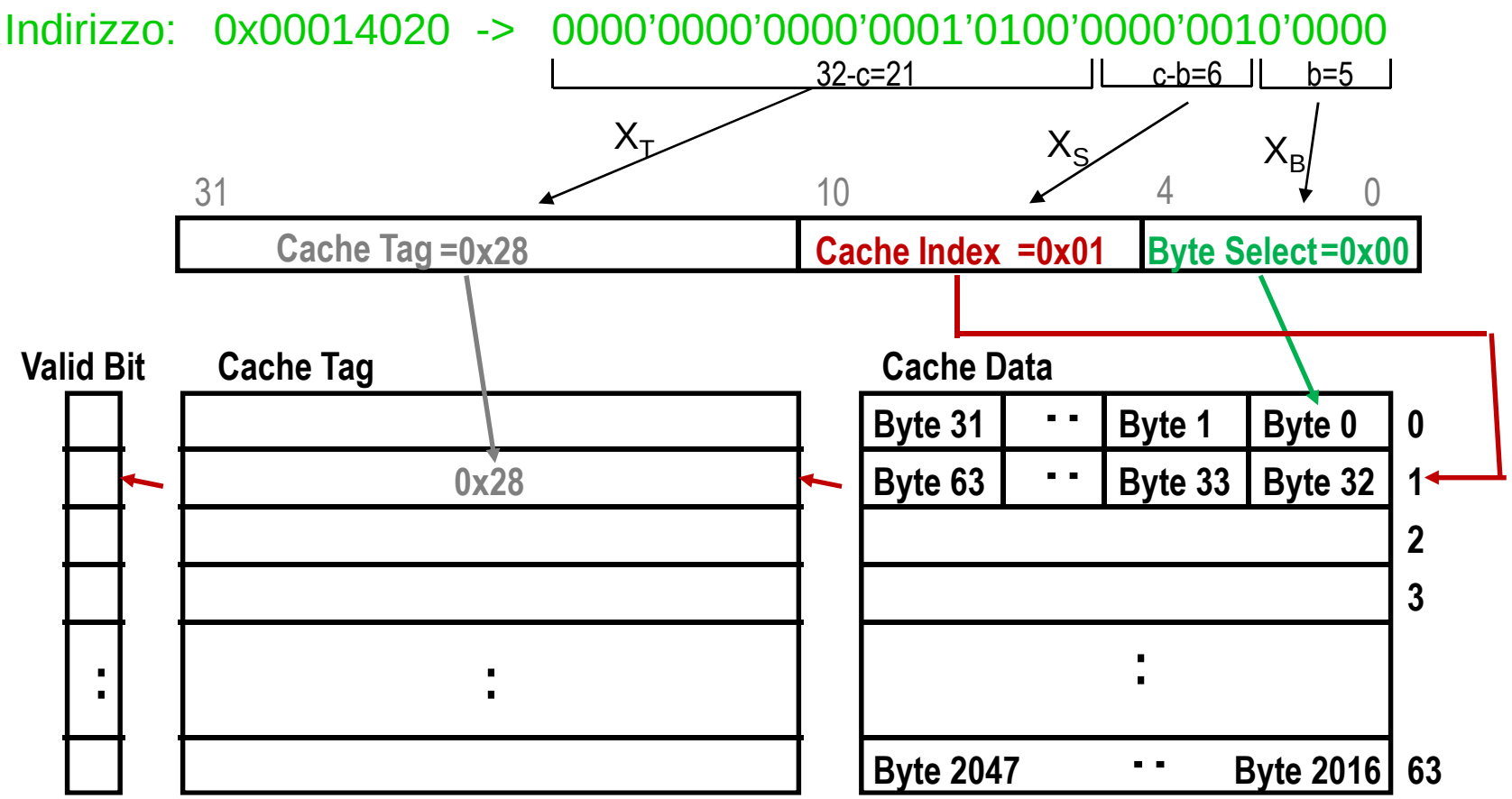
$X_S = X_M \bmod N_C$
$= X_M \bmod C/B$
$= X_M \bmod 2^{12}$

$X_B \equiv X \bmod B$
$X_B \equiv$ byte Block-offset
$X_O \equiv X_B / W$,
$X_O \equiv$ word Block-offset
$W \equiv$# di byte nella word



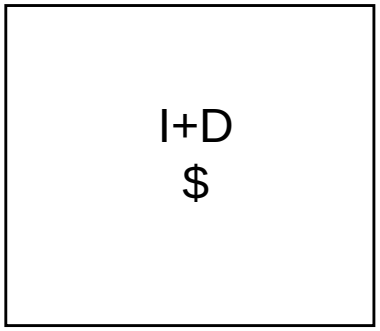
Esempio: cache ad accesso diretto C=2K, B=32

- Per una cache ad accesso diretto di dimensione $C=2^c$ byte e blocco $B=2^b$ byte
  - i (32 – c) bit piu' significativi sono sempre il Tag (X_T)
  - i b bit meno significativi sono sempre il byte-offset nel blocco ($X_B \equiv X \bmod B$)

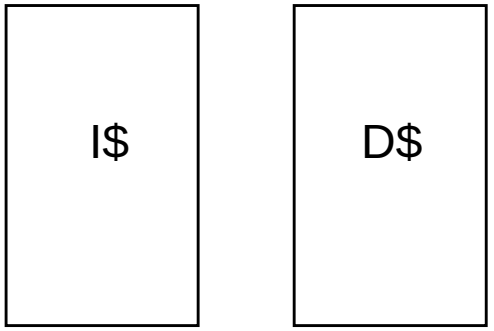


Split Cache

CACHE UNIFICATA



SPLIT CACHE



- La split cache consente di sfruttare maggiormente la localita' spaziale soprattutto nell'accesso ai dati

Program	Block size in words	Instruction miss rate	Data miss rate	Effective combined miss rate
gcc	1	6.1%	2.1%	5.4%
	4	2.0%	1.7%	1.9%
spice	1	1.2%	1.3%	1.2%
	4	0.3%	0.6%	0.4%

Tempo Medio di Accesso alla Memoria (con CACHE)

- **Definiamo**

h = Probabilita' di Hit

m = Probabilita' di Miss = (1-h)

t_{hit} = Tempo di Accesso in caso di Hit

t_{miss} = Tempo di Accesso in caso di Miss

$t_{penalty}$ = Tempo aggiuntivo rispetto al caso di Hit, per servire un Miss (= $t_{miss} - t_{hit}$)

→ **Average Memory Access Time (AMAT):**

$$AMAT = h \times t_{hit} + m \times t_{miss}$$
$$= h \times t_{hit} + m \times (t_{hit} + t_{penalty})$$
$$= t_{hit} + m \times t_{penalty}$$

(MissTime = HitTime + PenaltyTime)

(MissRate + HitRate = 1)

Ovvero Average Memory Access Cycles (AMAC):

$$AMAC = C_{hit} + m \times C_{penalty}$$

($C_{hit} = t_{hit}/T_C$, $C_{penalty} = t_{penalty}/T_C$)

In altri termini, SE NON HO HIT IN CACHE, i cicli "ideali" (C_{hit}) per l'accesso alla memoria sono aumentati dei cicli di penalty ($C_{penalty}$)

Numero medio di riferimenti per istruzione

- **OGNI ISTRUZIONE FA ACCESSO ALLA MEMORIA**
  es. ADD R1, R2, R3 genera 1 riferimento (alla memoria istruzioni)
  LD R1, 0(R2) genera 2 riferimenti (1 alla memoria dati
  +1 alla mem. istruzioni)

- Indichiamo quindi con
  $/R_{INST}$ = Average References per Instruction $\quad \overline{R}_{INST} \stackrel{\Delta}{=} \frac{N_{REF}}{N_{CPU}}$

- $N_{REF,i}$=numero di riferimenti generati dalla istruzione i-esima
  - $N_{REF,i1}$ =2 (es. istruz. LOAD)
  - $N_{REF,i2}$ =1 (es. istruz. ADD)

$$\overline{R_{INST}} = \frac{N_{REF}}{N_{CPU}} = \frac{1}{N_{CPU}} \sum_{i=1}^{N_{CPU}} N_{REF,i}$$

- Nell'esempio precedente:
  - N_{REF} = numero totale di riferimenti = 3
  - N_{CPU} = 2

  →$/R_{INST}$=1.5

Nota: la cache vede al suo ingresso RIFERIMENTI, non istruzioni !

CPI modificato

- QUINDI:
  - Ogni istruzione genera un numero medio di riferimenti /RINST e incorrera' in un ulteriore ritardo Cpenalty per ogni miss che si verifica con una probabilita' m

- In altri termini, detti:

  /RINST = Average References per Instructions

  m = Miss Rate = (# of Misses) / (# of References)

  Cpenalty = Penalty Cycles = $t_{penalty}$ * T_C

il CPI medio si puo' calcolare come

/CPI = /CPIideal + /RINST * m * Cpenalty

$$\overline{R_{INST}} = \frac{N_{REF}}{N_{CPU}} = \frac{1}{N_{CPU}} \sum_{i=1}^{N_{CPU}} N_{REF,i}$$

$$\overline{CPI_{ideal}} = \sum_{K=1}^{T} \overline{CPI_K} = \frac{1}{N_{CPU}} \cdot \sum_{K=1}^{T} N_{CPU,K} \cdot CPI_K$$

Calcolato supponendo sempre hit in cache

Eq. delle prestazioni MODIFICATA

- Equazione delle prestazioni:

$$T_{CPU} = C_{CPU} \cdot T_C = N_{CPU} \cdot \overline{CPI} \cdot T_C$$

- /CPI stavolta e' un "CPI-efficace" = "CPI ideale" +

  + numero cicli medi di stallo (penalty) per istruzione

$$T_{CPU} = N_{CPU} \cdot \left(\overline{CPI}_{ideal} + \frac{N_{REF}}{N_{CPU}} \cdot \overbrace{\overline{R}_{INST}} \cdot m \cdot C_{pen} \right) \cdot T_C$$

$$= T_{CPU,ideal} + N_{REF} \cdot m \cdot C_{pen} \cdot T_C$$

→ Ho due modi per aumentare le prestazioni:

- Diminuire il miss rate
- Diminuire i cicli di penalty

Contributi separati delle istruzioni generiche e L/S

$$\mathrm{T}_{\mathrm{CPU}} = N_{\mathrm{CPU}} \cdot \left(\overline{CPI}_{ideal} + \left(\frac{N_{\mathrm{REF,I}}}{N_{\mathrm{CPU}}} \cdot m_I + \frac{N_{\mathrm{REF,D}}}{N_{\mathrm{CPU}}} \cdot m_D \right) \cdot C_{pen} \right) \cdot \mathrm{T}_{\mathrm{C}}$$

$$m_I = N_{MISS\_DATI} \,/\, N_{REF,I}$$

$$m_D = N_{MISS\_ISTRUZIONI} \,/\, N_{REF,D}$$

$$(**) = N_{\mathrm{CPU}} \cdot \left(\overline{CPI}_{ideal} + \left(m_I + \frac{N_{\mathrm{CPU,L/S}}}{N_{\mathrm{CPU}}} \cdot m_D \right) \cdot C_{pen} \right) \cdot \mathrm{T}_{\mathrm{C}}$$

$$N_{\mathrm{REF,I}} = N_{\mathrm{CPU}}$$

$$N_{\mathrm{REF,D}} = N_{\mathrm{CPU,L/S}}$$

$$= \mathrm{T}_{\mathrm{CPU,ideal}} + \left(N_{\mathrm{CPU}} \cdot m_I + N_{\mathrm{CPU,L/S}} \cdot m_D \right) \cdot C_{pen} \cdot \mathrm{T}_{\mathrm{C}}$$

m= miss combinato dati+istruzioni

$$= \mathrm{T}_{\mathrm{CPU,ideal}} + N_{\mathrm{REF}} \cdot \underbrace{\left(\frac{N_{\mathrm{CPU}} \cdot m_I + N_{\mathrm{CPU,L/S}} \cdot m_D}{N_{\mathrm{REF}}} \right)}_{= m} \cdot C_{pen} \cdot \mathrm{T}_{\mathrm{C}}$$

$$m = \frac{N_{\mathrm{CPU}} \cdot m_I + N_{\mathrm{CPU,L/S}} \cdot m_D}{N_{\mathrm{REF}}}$$

$$= \mathrm{T}_{\mathrm{CPU,ideal}} + N_{REF} \cdot m \cdot C_{pen} \cdot \mathrm{T}_{\mathrm{C}} = \mathrm{T}_{\mathrm{CPU,ideal}} + N_{CPU} \cdot m \cdot \overline{R}_{INST} \cdot C_{pen} \cdot \mathrm{T}_{\mathrm{C}}$$

Nota: per una split cache gli accessi avvengono in parallelo. Quindi: $m_{I,SPLIT} \neq m_I$, $m_{D,SPLIT} \neq m_D$, $m_{SPLIT} \neq m$

Rapporto fra T_{CPU} effettivo ed ideale

$$T_{CPU} = T_{CPU,ideal} + N_{CPU} \cdot m \cdot \overline{R}_{INST} \cdot C_{pen} \cdot T_C$$

$$\frac{T_{CPU}}{T_{CPU,ideal}} = 1 + \frac{N_{CPU} \cdot m \cdot \overline{R}_{INST} \cdot C_{pen} \cdot T_C}{N_{CPU} \cdot \overline{CPI}_{ideal} \cdot T_C} = 1 + \frac{m \cdot \overline{R}_{INST} \cdot C_{pen}}{\overline{CPI}_{ideal}}$$

$$\boxed{\frac{T_{CPU}}{T_{CPU,ideal}} = 1 + \frac{m \cdot \overline{R}_{INST} \cdot C_{pen}}{\overline{CPI}_{ideal}}}$$

Rallentamento della macchina
reale con cache avente miss **m** e penalty $\mathbf{C_{pen}}$
(/R_{INST} e /CPI_{ideal} non dipendono dalla cache)

Esempio: impatto della cache sulle prestazioni

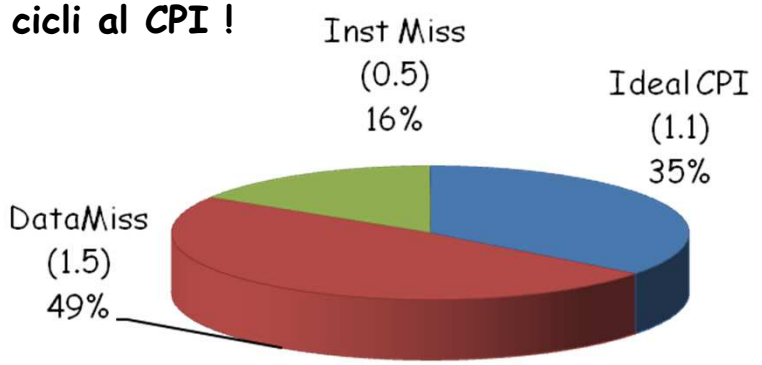
- Supponiamo di avere un processore a 200 MHz (5 ns per ciclo)
  - CPI_{ideal} = 1.1
  - 50% arit/log, 30% ld/st, 20% control
- Supponiamo che il miss-rate per le istruzioni sia l'1%
- Supponiamo che il 10% delle operazioni in memoria subisca miss con una penalty pari a a 50 cicli (di clock)
- Da (**) si ha per il CPI: $CPI = CPI_{ideal} + (m_I + N_{CPU,L/S}/N_{CPU} \times m_D) \times C_{pen}$

$$= 1.1 \text{ (cycles)} + 0.01 \text{ (miss/inst)} \times 50 \text{ (cycles/miss)} + 0.30 \text{ (dmop/inst)} \times 0.10 \text{ (miss/dmop)} \times 50 \text{ (cycles/miss)}$$

$$= 1.1 \text{ cycles} + 0.5 \text{ cycles} + 1.5 \text{ cycle} = 3.1$$

→Per il 65% (2.0/3.1) del tempo il processore attende la mem.!

→Un 1% di miss-rate per le istruzioni aggiunge 0.5 cicli al CPI !



dmop=data memory operation