


## Struttura delle cache a indirizzamento diretto

- Ogni posizione della cache include:
  - ▶ **Valid bit** che indica se questa posizione contiene o meno dati validi.
    - 0: posizione di cache non ancora utilizzata
    - 1: posizione di cache occupata con dei dati dalla RAM  
(Quando il calcolatore viene acceso tutte le posizioni della cache sono segnalate come NON valide)
  - ▶ **Campo etichetta (Tag)** contiene un valore che identifica univocamente l'indirizzo del blocco di memoria memorizzato nella posizione della cache  
  
Il valore del Tag include l'indirizzo del blocco in RAM, eccetto i  $k$  bit meno significativi (non mi servono! Sono la posizione nella cache)
  - ▶ **Campo dati** che contiene una copia del blocco in RAM

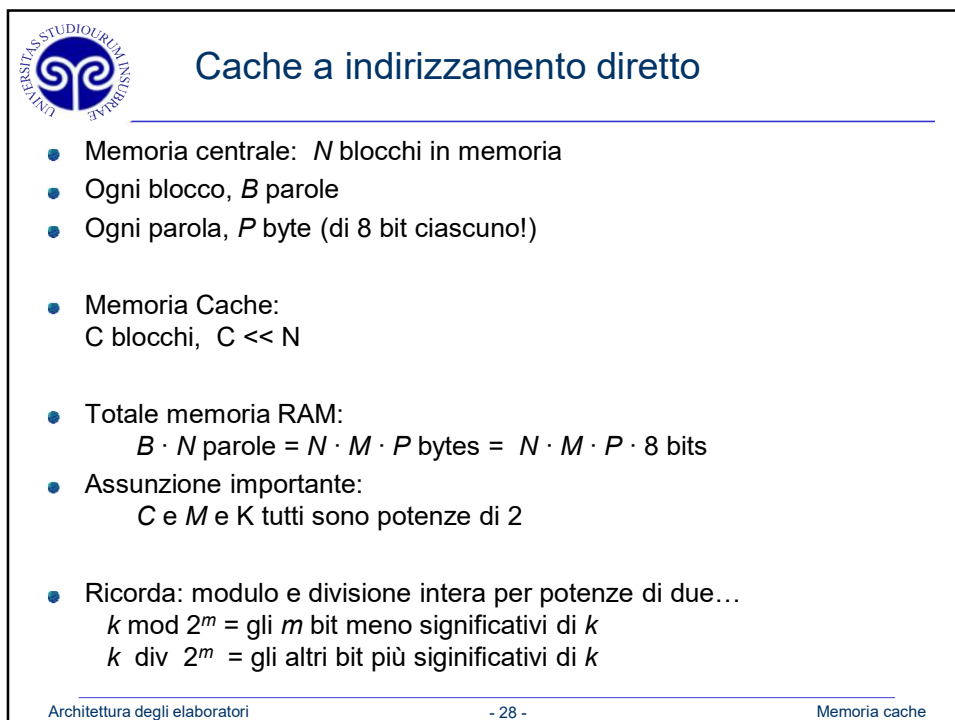
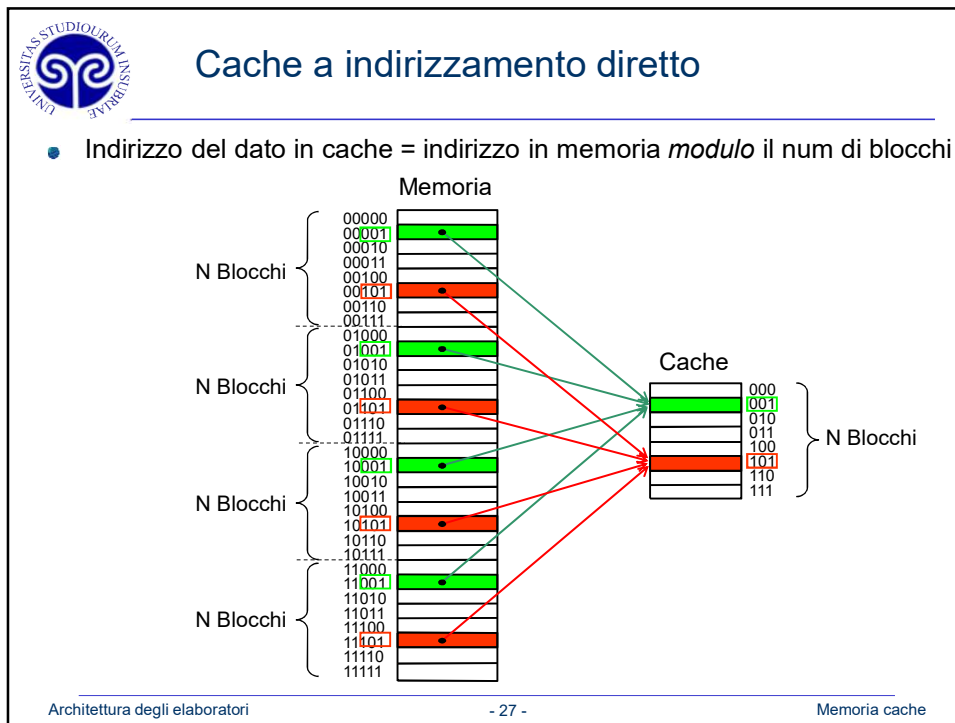
Architettura degli elaboratori - 25 - Memoria cache




## Indirizzamento diretto

- Per sapere se un blocco di un dato indirizzo  $m$  è in cache:
- con  $m$  composto da due sottosequenze di bit  $m_0$  e  $m_1$   
 $m = m_0, m_1$
- accedo al blocco  $m_1$  della cache
- se valid bit = 0 : cache miss
- se valid bit = 1  
analizzo il tag memorizzato nel blocco di cache.  
Se corrisponde a  $m_0$  : cache hit  
Altrimenti: cache miss

Architettura degli elaboratori - 26 - Memoria cache






## Cache a indirizzamento diretto

---

- **Esempio:**
  - ▶ memoria RAM totale: 16 MegaByte =  $2^{24}$  bytes
  - ▶ parole da: 32 bit = 4 bytes =  $2^2$  bytes
  - ▶ blocchi di: 512 parole =  $2^9$  parole =  $2^{11}$  bytes
  - ▶ mem cache (esclusi bit di tag): 128 Kbytes =  $2^{17}$  bytes =  $2^{15}$  parole
- **Domande:**
  - ▶ Lunghezza indirizzo di un byte in RAM? 24
  - ▶ Lunghezza indirizzo di una parola in RAM?  $24 - 2 = 22$
  - ▶ N. blocchi in memoria cache?  $2^{17} / 2^{11} = 2^6 = 64$
  - ▶ N. blocchi in memoria RAM?  $2^{24} / 2^{11} = 2^{13} = \sim 8000$
  - ▶ Bit di tag della cache?  $13 - 6 = 7$

Architettura degli elaboratori
- 29 -
Memoria cache




## Cache a indirizzamento diretto

---

- **Esempio:**
  - ▶ memoria RAM totale: 16 MegaByte =  $2^{24}$  bytes
  - ▶ parole da: 32 bit = 4 bytes =  $2^2$  bytes
  - ▶ blocchi di: 512 parole =  $2^9$  parole =  $2^{11}$  bytes
  - ▶ mem cache (esclusi bit di tag): 128 Kbytes =  $2^{17}$  bytes =  $2^{15}$  parole

TAG del blocco in Cache	indir. blocco in Cache	indice parola nel blocco	byte offset																				
1	0	1	1	0	1	0	0	1	1	1	1	0	1	1	1	1	0	0	0	1	1	0	1
TAG del blocco in Cache							indir. blocco in Cache							indice parola nel blocco							byte offset		
indice blocco in RAM																							
indice parola in RAM																							
indice del byte in RAM																							

Architettura degli elaboratori
- 30 -
Memoria cache



## Indirizzamento nella cache a indirizzamento diretto: esempio

---

**Memory (16 parole)**

Address	
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
A	
B	
C	
D	
E	
F	

**Direct Mapped Cache (4 parole)**


Cache Index	
0	
1	
2	
3	

- Ad ogni indirizzo di memoria corrisponde *una ed una sola* posizione nella cache.
  - ▶ Ogni parola di memoria *può* trovarsi in un'unica posizione della cache
- Ad ogni posizione della cache corrispondono più indirizzi di memoria di livello inferiore

Architettura degli elaboratori

- 31 -

Memoria cache



## Cache a indirizzamento diretto da 4 parole

---

**Memory (16 parole)**

0 (0000)	
1 (0001)	
2 (0010)	
3 (0011)	
4 (0100)	
5 (0101)	
6 (0110)	
7 (0111)	
8 (1000)	
9 (1001)	
A (1010)	
B (1011)	
C (1100)	
D (1101)	
E (1110)	
F (1111)	

**Direct Mapped Cache (4 parole)**

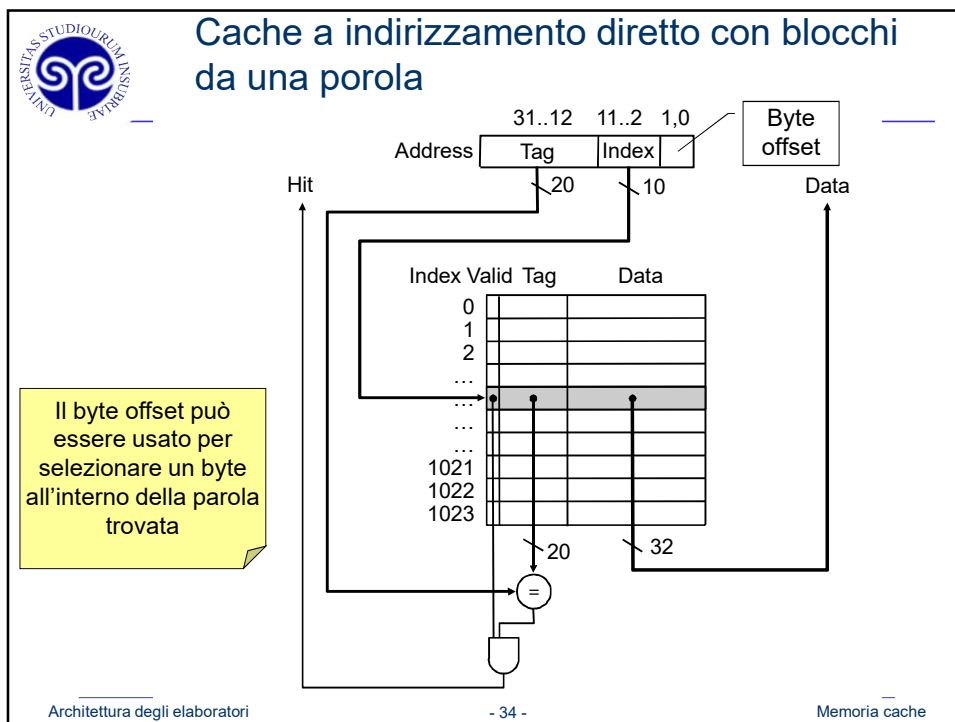
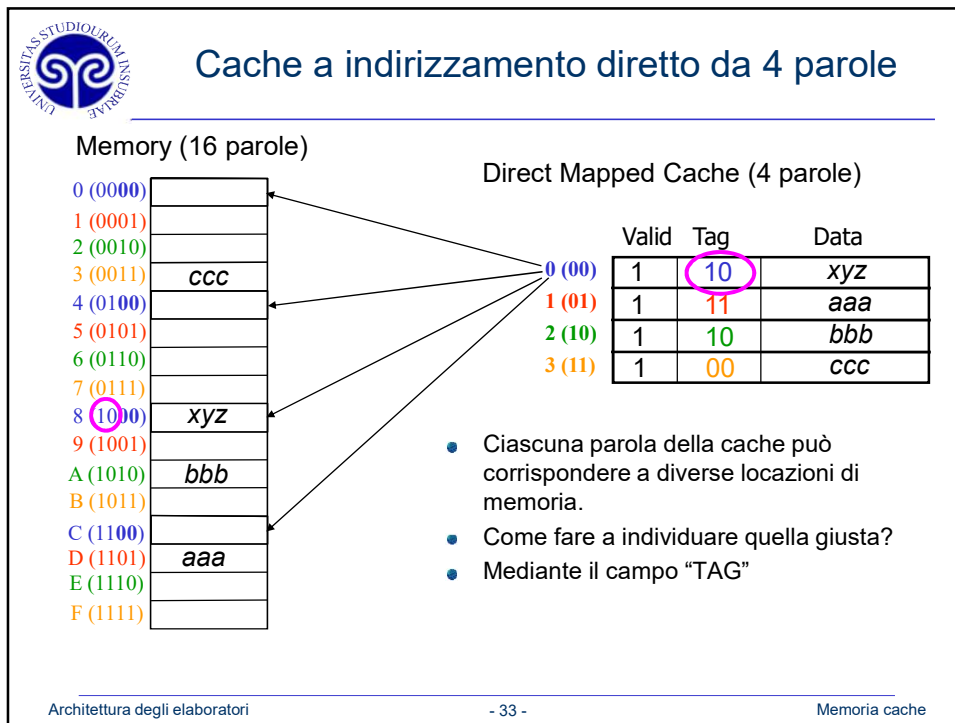
0 (00)	
1 (01)	
2 (10)	
3 (11)	


- Posizione 0 (00) può essere occupata da dati (parole) provenienti da:
  - ▶ Indirizzi di memoria 0, 4, 8, ... etc.
  - ▶ In generale: ogni indirizzo di memoria i cui 2 bit meno significativi dell'indirizzo sono 0
  - ▶ La posizione in cache è data dai due bit meno significativi dell'indirizzo

Architettura degli elaboratori

- 32 -

Memoria cache





## Linee di cache (blocchi) di dimensioni superiori alla parola


---

- Indirizzi di memoria a 32 bit
- Cache a indirizzamento diretto con 4096 ( $2^{12}$ ) linee di cache. Ciascuna linea contiene 4 parole a 4 byte.
- Struttura dell'indirizzo di memoria:
  - ▶ Bit 0 e 1 per individuare il singolo byte
  - ▶ Bit 2 e 3 per individuare una parola nel blocco
  - ▶ Bit 4-15 per individuare il blocco di cache
  - ▶ Bit 16-31 come tag

Architettura degli elaboratori

- 35 -

Memoria cache



## Cache a indirizzamento diretto

---

- La linea di cache di dimensioni maggiori (es. 4 parole) per sfruttare la località spaziale

**Indirizzo (con l'indicazione della posizione dei bit)**

31	30	29	28	27	.....	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
----	----	----	----	----	-------	----	----	----	----	----	----	----	---	---	---	---	---	---	---	---	---	---

↳ Spiazzamento byte

↳ Spiazzamento blocco

Successo

Etichetta 16

Indice 12

VEtichetta 16 bit

Dati 128 bit

4 K elementi

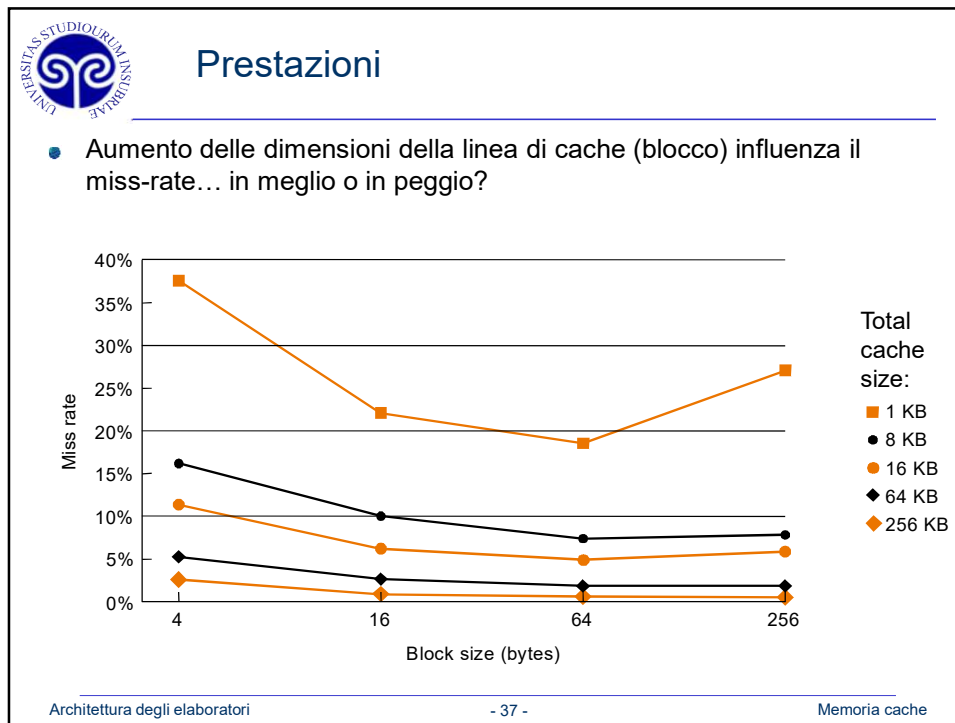
Multiplexer

Dato

Architettura degli elaboratori

- 36 -


Memoria cache



**Prestazioni**

- Dimensione della cache:
  - ▶ diminuisce il miss rate, ma il beneficio diminuisce all'aumentare della dimensione
  - ▶ aumenta il costo (ovviamente) e lo hit-time (memoria più grande)
- Indirizzamento diretto: semplice e veloce
- Indirizzamento associativo: caro oppure lento
- Per compiere tutte queste difficili scelte, un architetto utilizza **profiling**: la simula del comportamento della cache su programmi reali di esempio che consente di valutare il bilancio fra costi e benefici:
  - ▶ serve un **benchmark** di programmi di esempio
  - ▶ il benchmark deve includere programmi di natura molto diversa es una simulazione fisica, un videogioco, un foglio di calcolo, il bootstrap, ...

Architettura degli elaboratori - 38 - Memoria cache




## Hit vs. Miss in lettura

---

- Interazione tra processore e cache: lettura o scrittura di un dato
- Cache hit in lettura
  - ▶ Dato letto dalla cache
  - ▶ Tutto ok. Nient'altro da fare ☺
- Cache miss in lettura
  - ▶ richiesta alla memoria del blocco contenente il dato cercato, copia in cache, ripetizione dell'operazione di lettura in cache
  - ▶ stallo della CPU: durante tutto questo tempo la CPU aspetta

---

Architettura degli elaboratori
- 39 -
Memoria cache



## Hit vs Miss in scrittura


---

- Successo nella scrittura: due strategie possibili
  - ▶ Sostituzione del dato sia in cache sia in memoria (**write-through**)
  - ▶ Scrittura del dato solo nella cache (**write-back**) : (la copia in memoria avverrà in un secondo momento)
- Fallimento nella scrittura (il dato non è in cache):
  - ▶ *stallo* della CPU, mentre:
  - ▶ richiesta del blocco contenente il dato cercato alla memoria, copia in cache, ripetizione dell'operazione di scrittura

---

Architettura degli elaboratori
- 40 -
Memoria cache






## Miglioramento delle prestazioni

---

- Migliorare sia larghezza di banda sia latenza: uso di cache multiple
- Introdurre una cache separata per istruzioni e dati (split cache)
  - ▶ Beneficio:
    - Le operazioni di lettura/scrittura possono essere svolte in modo indipendente in ogni cache
    - ⇒ raddoppia la larghezza di banda della memoria!
  - ▶ Costo:
    - Processore necessita di due porte di collegamento alla memoria

---

Architettura degli elaboratori
- 41 -
Memoria cache



## Prestazioni della cache a indirizzamento diretto

---

- Se due locazioni appartengono ad un blocco di cache diversi che condividono lo stesso slot in cache, allora non potranno mai essere in cache contemporaneamente.
- Cosa succede se un programma lavora per un certo tempo proprio su quelle due locazioni?

---

Architettura degli elaboratori
- 42 -
Memoria cache

**Prestazioni con cache a indirizzamento diretto: caso ottimo**

4 Byte Direct Mapped Cache

Cache Index	Valid	Tag	Data
0 (00)	1	00	
1 (00)	1	00	
2 (00)	1	00	
3 (00)	1	00	

Un programma accede ciclicamente a queste locazioni

I dati sono in cache: il programma accede ciclicamente a queste locazioni

- Una volta caricati i blocchi in cache non si hanno più miss
- Massimo vantaggio (nessun accesso a memoria centrale)

Architettura degli elaboratori - 43 - Memoria cache

**Prestazioni con cache a indirizzamento diretto: caso pessimo**


4 Byte Direct Mapped Cache

Cache Index	Valid	Tag	Data
0 (00)	1	00	
1 (00)	1	00	
2 (00)	1	00	
3 (00)	1	00	

Un programma accede ciclicamente a queste locazioni

- In cache c'è la riga con tag 00 quando serve quella con tag 01.
- Viceversa quando serve quella con tag 00, in cache c'è quella con tag 01.
- Prestazioni perfino peggiori che senza cache!

Architettura degli elaboratori - 44 - Memoria cache




## Analisi delle prestazioni (modello semplificato)

- Tempo di esecuzione = (cicli di esecuzione + cicli di stallo) × periodo del ciclo
  - ▶ Si ha un ciclo di stallo quando la CPU deve attendere il caricamento della cache a causa di un miss
- Cicli di stallo = #miss × #cicli per miss = (# istruzioni × miss rate) × miss penalty
  - ▶ miss penalty = #cicli per ogni miss
- Due modi per migliorare le prestazioni:
  - ▶ ridurre miss rate
  - ▶ ridurre il miss penalty

---

Architettura degli elaboratori - 45 - Memoria cache




## Ridurre il miss rate

- Un modo per ridurre il miss rate consiste nel consentire a qualunque combinazione di blocchi di stare contemporaneamente in cache
- Meglio adattandosi così alle esigenze dei programmi.

---

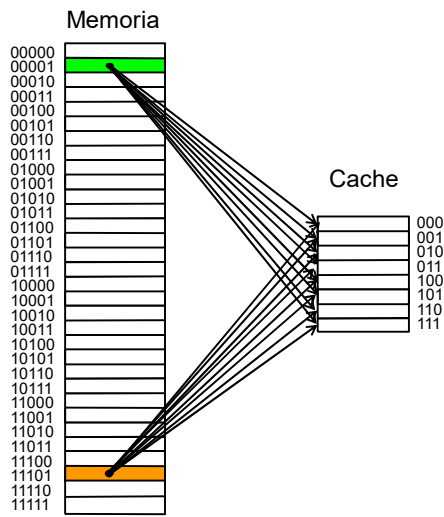
Architettura degli elaboratori - 46 - Memoria cache



## Cache completamente associative

---


- Un blocco può essere memorizzato *in qualunque posizione* della cache.
- Non esiste una relazione fissa tra indirizzo di memoria del dato e posizione in cache



Architettura degli elaboratori

- 47 -

Memoria cache



## Cache (completamente) associativa: indirizzamento

---

(Fully) associative cache memory.  
Qui: 8 blocchi da  $2^M$  byte ciascuno

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data


N-M     $2^M$   
bit    byte

- Se la RAM è di  $2^N$  byte (l'indirizzo RAM è di N bit) e il blocco è di  $2^M$  byte
  - ▶ M bit di indirizzo per spiazzamento (offset) byte nel blocco
  - ▶ gli altri N-M bit di indirizzo sono etichetta (tag)
- Questo è indipendente dalla dimensione della cache

Architettura degli elaboratori

- 48 -

Memoria cache



## Cache (completamente) associativa: indirizzamento

---


- Esempio:
  - ▶ Memoria di 64KByte
    - 16 bit di indirizzo
  - ▶ Cache contenente k blocchi da  $2^4 = 16$  byte ciascuno
- Struttura dell'indirizzo:
  - ▶ I 4 bit meno significativi individuano il byte all'interno del blocco da 16 byte memorizzato nella cache
  - ▶ 12 bit più significativi: etichetta

Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data	Tag	Data
-----	------	-----	------	-----	------	-----	------	-----	------	-----	------	-----	------	-----	------

} 12bit
} 16byte

Dato un indirizzo qualunque, i suoi 12 bit più significativi devono essere confrontati con tutti i tag dei blocchi in cache

Architettura degli elaboratori
- 49 -
Memoria cache



## Cache (completamente) associativa

---

- Esempio:
  - ▶ memoria RAM totale = 16 MegaByte =  $2^{24}$  bytes
  - ▶ parole da 32 bit = 4 bytes =  $2^2$  bytes
  - ▶ blocchi di : 512 parole =  $2^9$  parole =  $2^{11}$  bytes
  - ▶ mem cache (esclusi bit di tag) = *qualsiasi numero di blocchi*

1	0	1	1	0	1	0	0	1	1	1	1	0	1	1	1	1	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

TAG del blocco in Cache

indice parola nel blocco


byte offset

indice blocco in RAM

indice parola in RAM

indice del byte in RAM

Architettura degli elaboratori
- 50 -
Memoria cache




## Cache (completamente) associativa

---

- Cercare un dato nella cache richiede il *confronto di tutte le etichette* presenti in cache con l'etichetta dell'indirizzo di memoria richiesto
  - ▶ Per consentire prestazioni decenti la ricerca avviene in parallelo (HW replicato: costoso!)
- Etichetta viene trovata: cache hit. (procedo come prima)
- Etichetta non viene trovata: cache miss.  
Quindi stallo: accedo alla RAM, e memorizzo il blocco acceduto nella cache. Dove?  
*Con le cache associative, possiamo/dobbiamo scegliere!*
- Se la cache non è ancora piena: scelgo un blocco vuoto qualsiasi di cache e ci copio il blocco della RAM (insieme con il suo tag)
- Se la cache è piena, è necessario sostituire un dato. Quale?
  - ▶ Scelta casuale, oppure
  - ▶ Scelta del dato utilizzato meno di recente (strategia «LRU», Least Recently Used)

Architettura degli elaboratori
- 51 -
Memoria cache



## Cache set-associative


---

- Per avere i vantaggi delle cache associative, riducendone i costi.
- Dividere i blocchi nella cache in *linee*, (o insiemi, o set) ciascuna linea =  $n$  blocchi
- Ogni blocco di RAM può andare in un'unica *linea* di cache
  - ▶ Scelta con lo stesso meccanismo dell'indirizzamento diretto
- Ogni linea comprende  $n$  blocchi
  - ▶ Un blocco dato può occupare qualsiasi posizione nella linea
  - ▶ Scelto con lo stesso meccanismo delle cache associative
- Una cache set-associativa in cui un blocco può andare in  $n$  posizioni si chiama «set-associativa a  $n$  vie». (« $n$ -ways set associative»)

Es: two-way set associative

set	tag	data	tag	data
0				
1				
2				
3				

Architettura degli elaboratori
- 52 -
Memoria cache




## Cache set-associative

- Ogni blocco della memoria corrisponde –come nella cache ad indirizzamento diretto– ad uno *set* della cache (set = insieme = «linea» della cache)
- il blocco può essere messo in uno qualsiasi degli  $n$  elementi di questo insieme
- Combina la modalità a indirizzamento:
  - ▶ diretto per scegliere il set
  - ▶ completamente associativa scegliere il blocco all'interno del set.

---

Architettura degli elaboratori - 53 - Memoria cache




## Indirizzamento nelle cache set-associative

- Un indirizzo di memoria di  $N$  bit è suddiviso in 4 campi:
  1.  $B$  bit meno significativi per individuare il byte all'interno della parola di  $2^B$  byte (detto offset del byte)
  2.  $W$  bit per individuare la parola all'interno del blocco di  $2^W$  parole
  3.  $M$  bit per individuare il set (insieme, linea) di cache
  4.  $N-(M+W+B)$  come etichetta
- Come nell'indirizzamento diretto

---

Architettura degli elaboratori - 54 - Memoria cache



## 4-way set associative Cache: esempio


---

- Dati:
  - ▶ memoria RAM totale : 16 MegaByte =  $2^{24}$  bytes
  - ▶ parole da : 32 bit
  - ▶ blocchi di : 512 parole
  - ▶ tot mem cache (senza tag) : 64Kb
  
- Di quanti set è composta la cache?
- Come si scompone questo indirizzo?

1	0	1	1	0	1	0	0	1	1	1	1	0	1	1	1	1	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

indice del byte in RAM

Architettura degli elaboratori
- 55 -
Memoria cache



## 4-way set associative Cache: esempio

4  
blocchi  
in ogni  
set

dim  
dei  
blocchi

---

- Dati:
  - ▶ memoria RAM totale : 16 MegaByte =  $2^{24}$  bytes
  - ▶ parole da : 32 bit = 4 bytes =  $2^2$  bytes
  - ▶ blocchi di : 512 parole =  $2^9$  parole =  $2^{11}$  bytes
  - ▶ tot mem cache (senza tag) : 64Kb =  $2^{16}$  bytes =  $2^3 \times 2^2 \times 2^{11}$  bytes

1	0	1	1	0	1	0	0	1	1	1	1	0	1	1	1	1	0	0	0	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

TAG da memoriz.  
in Cache

indir. del Set  
in Cache


indice parola  
nel blocco

byte  
offset

indice del byte in RAM

Architettura degli elaboratori
- 56 -
Memoria cache





## Cache set-associativa

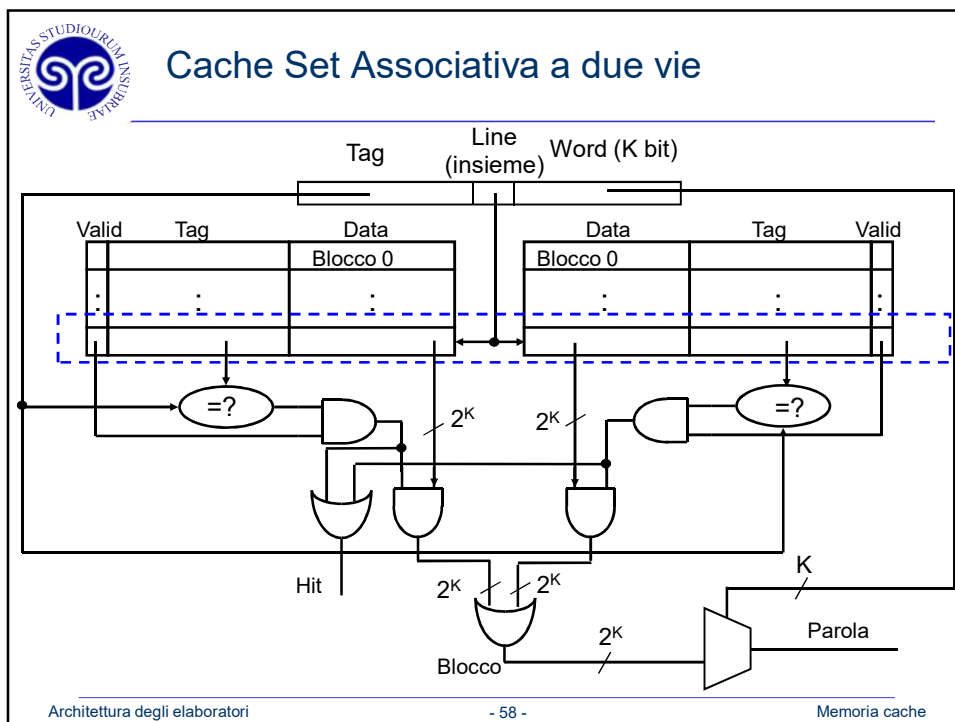
---


- Cache a due vie: insiemi (linee) di 2 blocchi
- Equivale ad avere due cache a indirizzamento diretto che operano in parallelo
- La parte di indirizzo che individua l'insieme seleziona i due blocchi della cache
- Le due etichette vengono confrontate in parallelo con quella dell'indirizzo cercato
- Il dato viene selezionato in base al risultato dei due confronti

Architettura degli elaboratori

- 57 -

Memoria cache





### Cache set associativa a 4 vie

#### Altro esempio.

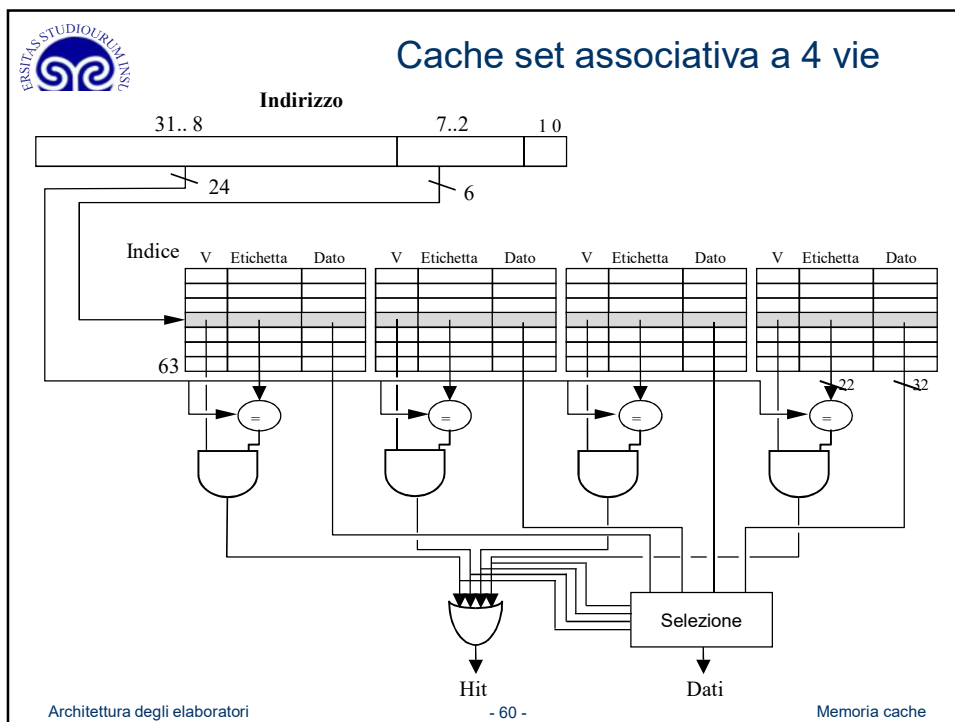
---

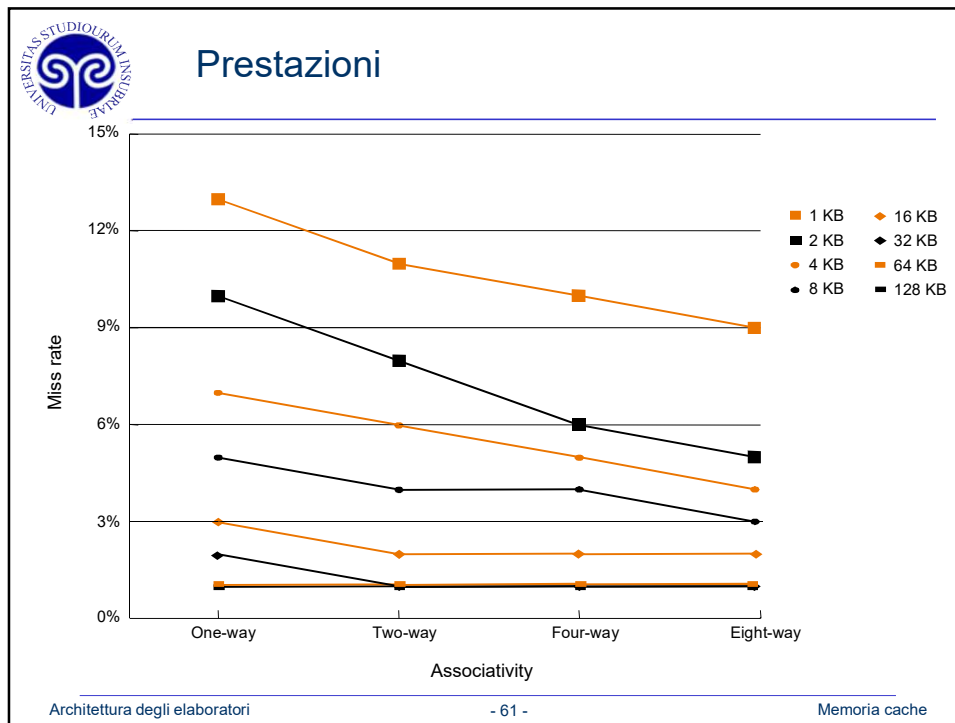
- Indirizzo di memoria: 32 bit
- Memoria cache 1KByte indirizzabile per byte, 1 parola da 4 Byte per blocco
  - ▶ Cioè ogni linea contiene un insieme di 4 blocchi, ciascuno da 4 byte (totale 16 byte)
  - ▶ Le linee sono  $1024/16 = 64 (=2^6)$
- Organizzazione dell'indirizzo:
  - ▶ Bit 0 e 1 per indirizzare i byte nella parola da 4 byte
  - ▶ Bit 2-7 indirizza dell'insieme nella cache
  - ▶ Bit 8-31 etichetta

Architettura degli elaboratori

- 59 -

Memoria cache






**Confronto tra diverse organizzazioni di cache**

- Cache set-associativa a N vie vs. Cache a indirizzamento diretto:
  - ▶ N comparatori vs. 1 per verificare che il tag sia quello giusto
  - ▶ Un ritardo dovuto al MUX aggiuntivo per i dati
  - ▶ Dati sono disponibili solo DOPO il segnale di Hit/Miss
- In una cache a indirizzamento diretto, il blocco di cache richiesto è disponibile PRIMA del segnale di Hit/Miss:
  - ▶ Possibile ipotizzare un successo e quindi proseguire. Si recupera successivamente se si trattava in realtà di un fallimento.

Architettura degli elaboratori - 62 - Memoria cache




## Conclusioni: 4 domande su gerarchia di memoria

---

- Q1: Dove si colloca un blocco nel livello di memoria superiore?  
(*Posizionamento del blocco*)
- Q2: Come si identifica un blocco che si trova nel livello superiore?  
(*Identificazione del blocco*)
- Q3: Quale blocco deve essere sostituito nel caso di un fallimento?  
(*Sostituzione del blocco*)
- Q4: Cosa succede durante una scrittura?  
(*Strategia di scrittura*)

---

Architettura degli elaboratori - 63 - Memoria cache




## Posizionamento del blocco

---

- Indirizzamento diretto:
  - ▶ Posizione univoca:  
[ indirizzo di memoria ] *modulo* [ numero dei blocchi in cache ]
- Completamente associativa:
  - ▶ Posizione qualunque all'interno della cache
- Set associativa
  - ▶ Insieme determinato univocamente come  
[indirizzo di memoria/numero dei blocchi]  
*modulo*  
[numero degli insiemi]
  - ▶ Posizione qualunque all'interno dell'insieme scelto

---

Architettura degli elaboratori - 64 - Memoria cache




## Identificazione del blocco

---

- Indirizzamento diretto:
  - ▶ Indice memoria inferiore determina posizione nella cache
  - ▶ Si confronta etichetta trovata con quella cercata etichetta e si verifica che bit «valido» = 1
- Completamente associativo:
  - ▶ Confronta etichetta in ogni blocco.
- Set-associativo
  - ▶ Identifica insieme
  - ▶ Confronta etichette dell'insieme e verifica bit valido

Architettura degli elaboratori
- 65 -
Memoria cache



## Sostituzione del blocco (quale blocco sostituire)


---

- Cache a indirizzamento diretto:
  - ▶ Nessuna scelta: è definito dall'indirizzo nelle
- Cache set associative o completamente associative:
  - ▶ Casuale, oppure
  - ▶ LRU (Least Recently Used)

**Cache miss:**  
esempio di risultato empirico di misurato su un benchmark:

Associatività	2-way		4-way		8-way	
Dimensione	LRU	Casuale	LRU	Casuale	LRU	Casuale
16 KB	5.2%	5.7%	4.7%	5.3%	4.4%	5.0%
64 KB	1.9%	2.0%	1.5%	1.7%	1.4%	1.5%
256 KB	1.15%	1.17%	1.13%	1.13%	1.12%	1.12%

Architettura degli elaboratori
- 66 -
Memoria cache




## Strategie di scrittura di un blocco

---

- **Write through**
  - ▶ L'informazione viene scritta sia in Cache che in Main Memory
- **Write back** (detta anche: copy back)
  - ▶ L'informazione viene scritta in cache.
  - ▶ La main memory viene aggiornata solo quando il blocco viene rimosso dalla cache
    - cioè quando quel blocco viene sostituito da un altro blocco
    - oppure con un'apposita operazione di cache «flush», da eseguire ad esempio prima di azzerare la cache

---

Architettura degli elaboratori
- 67 -
Memoria cache




## Strategie di scrittura: write back

---

- Per ogni blocco di cache è necessario mantenere un bit «Modificato» che indica se il blocco in cache è stato modificato o meno
  - ▶ detto anche bit «dirty»:
    - 0 = il blocco è una copia esatta, *pulita*, della main RAM
    - 1 = il blocco è stato *sproccato* da una scrittura
- Quando un blocco «dirty» viene sostituito, deve prima essere copiato definitivamente nella main RAM
- Nota: fino ad allora, se tale blocco venisse letto, verrà trovato in cache, e quindi, correttamente, la lettura restituirà il valore modificato (anche se la modifica non ha ancora raggiunto la main RAM)

---

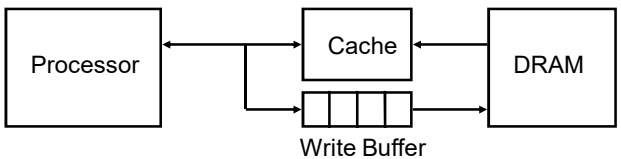
Architettura degli elaboratori
- 68 -
Memoria cache



## Strategie di scrittura: Write Through


---

- per non aumentare troppo i tempi di scrittura dovuti alle inferiori prestazioni della memoria di livello inferiore, Write Through viene realizzato con buffer di scrittura



- ▶ Processore: scrive i dati in cache e nel buffer di scrittura
- ▶ Controllore di memoria: scrive i contenuti del buffer in memoria

Architettura degli elaboratori
- 69 -
Memoria cache




## Write through vs Write back

---

- Write through
  - ▶ ☹ Semplice da implementare
  - ▶ ☹ Scarsa efficienza quando la stessa parola viene aggiornata più volte di fila
- Write back
  - ▶ ☹ Più complessa da implementare
  - ▶ ☺ Si evitano aggiornamenti ripetuti delle stesse celle di memoria: l'aggiornamento avviene una volta sola
  - ▶ ☹ Efficienza non ottimale quando si ricopia un intero blocco contenente sia parole modificate che parole non modificate.

Architettura degli elaboratori
- 70 -
Memoria cache




## Miss in lettura

---

- Se si ha un miss in lettura il blocco corrispondente viene caricato in memoria
- Strategia **Read back**
  - ▶ Prima si carica il blocco in cache *interamente*.
  - ▶ Poi la lettura avviene dal blocco di cache (come nel cache hit)
- Strategia **Load through** (o **early restart**)
  - ▶ Prima si manda alla cache la parola cercata, e da qui al processore (che può proseguire l'esecuzione)
  - ▶ Poi la lettura del resto del blocco prosegue *in parallelo*

Architettura degli elaboratori
- 71 -
Memoria cache



## Tempo di accesso alla memoria

---

- Tempo medio di accesso alla memoria =
 
$$\text{hit\_rate} \times \text{hit\_time} + \text{miss\_rate} \times \text{miss\_penalty}$$


di solito, questo secondo termine domina la somma
- Dove:
  - ▶ Tasso di successo = **hit\_rate**
  - ▶ Tasso di fallimento = **miss\_rate** = 1 - **hit\_rate**
  - ▶ **hit\_time** ≈ tempo di accesso alla cache (e ricerca del blocco)
  - ▶ **miss\_penalty** = penalità di fallimento =
 
$$\text{tempo di fallire la ricerca del blocco in cache} \approx \text{hit\_rate} + \text{tempo di accesso alla main memory} + \text{tempo di copiare un intero blocco in cache (e trovare dove copiarlo)}$$

prima, bisogna fallire la ricerca in cache

MOLTO GRANDE

Architettura degli elaboratori
- 72 -
Memoria cache

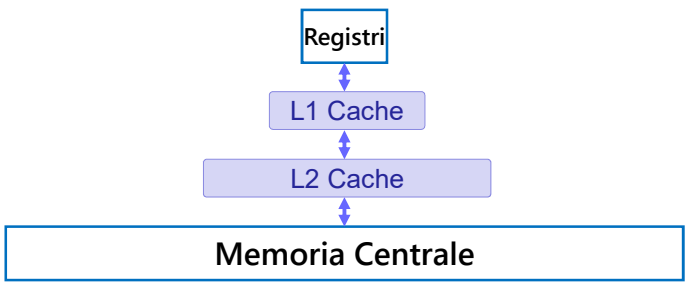




## Riduzione della penalt  di fallimento: cache multilivello

---


- Aggiunta di un secondo livello di cache
- Es, quando ci sono due livelli:
  - L1 Cache (primaria)
  - L2 Cache (secondaria)
- La penalt  di fallimento di L1 si riduce, se il dato   disponibile in L2



```

graph TD
    R[Registri] <--> L1[L1 Cache]
    L1 <--> L2[L2 Cache]
    L2 <--> MC[Memoria Centrale]
            
```

Architettura degli elaboratori
- 73 -
Memoria cache

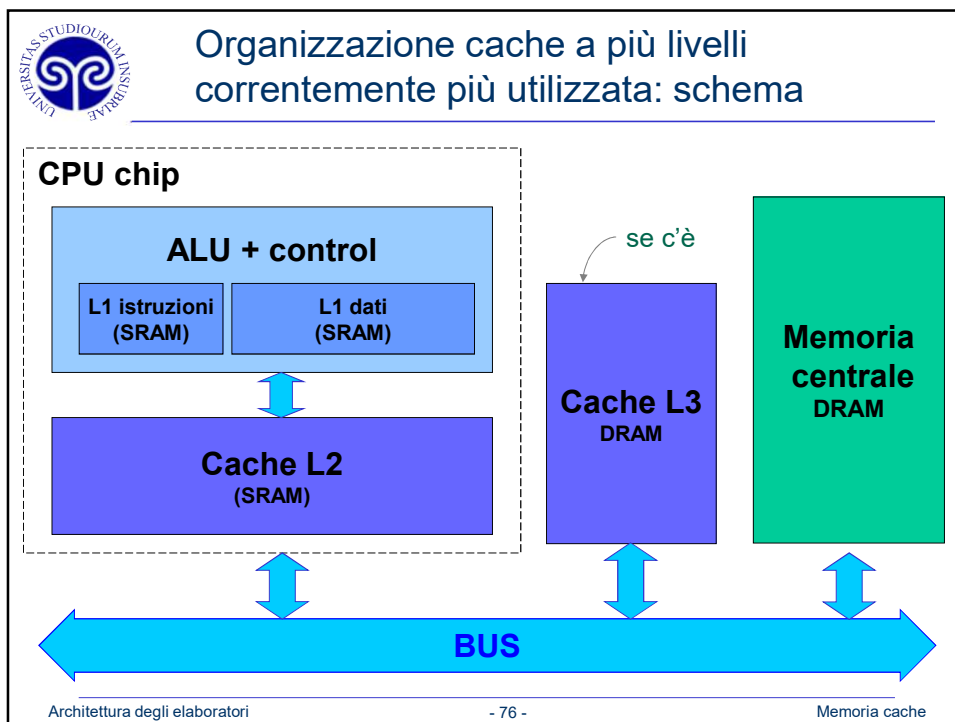
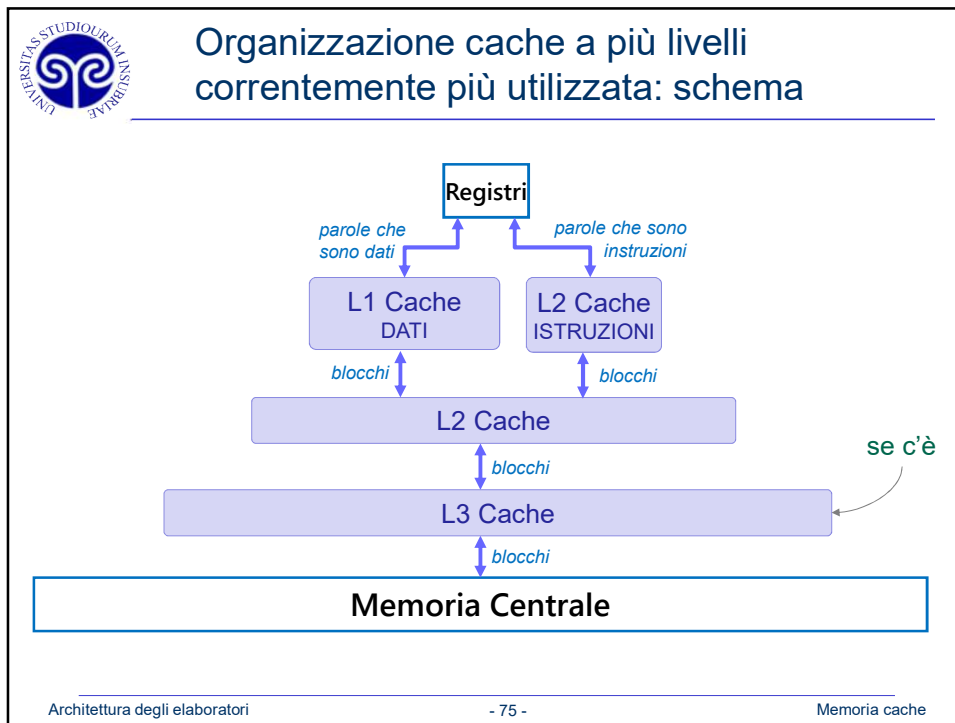



## Riduzione della penalt  di fallimento: cache multilivello

---

- Cache miss su L1? Cercare in L2.
  - Poi, caricare il blocco su L1.
  - Miss anche su L2? Caricare blocco su L2 (dalla RAM), quindi su L1
- Conviene la cache multilivello?
  - difficile dirlo a priori: per fortuna c'  il profiling!
- Nelle architetture moderne, tipicamente:
  - ci sono due livelli di cache L1 e L2 (a volte L3)
  - sono entrambi SRAM
  - sono entrambi a bordo del microchip del processore
    - se c'  un L4 (di solito, NO),   DRAM e in un chip separato
    - come un tempo era anche la L1-Cache
  - «split cache»:
    - ci sono due cache L1, specializzate per:
      - istruzioni (L1-Instruction)
      - dati (L2-Data)

Architettura degli elaboratori
- 74 -
Memoria cache





## Prestazioni con due livelli di cache

---

**Tempo medio di accesso alla memoria =**


$$\text{hit\_rate\_di\_L1} \times \text{hit\_time\_a\_L1} +$$

$$\text{miss\_rate\_di\_L1} \times \left( \text{hit\_rate\_di\_L2} \times (\text{hit\_time\_a\_L1} + \text{hit\_time\_a\_L2}) \right.$$

$$\left. \text{miss\_rate\_di\_L2} \times \text{penalità\_di\_fallimento\_L2} \right)$$


---

Architettura degli elaboratori - 77 - Memoria cache




## Take-home messages

---

- Questa lezione sulle cache ha compreso due importanti messaggi:
  - ▶ quando guarderemo la progettazione di CPU, ricordiamoci che ogni accesso in memoria main (lettura o scrittura) **può richiedere una quantità di tempo molto variabile** e difficilmente predicibile
    - alta nei rari, ma possibili, casi di cache miss, (e ancora peggio se cache miss multipli, su cache multilivello)
    - ordini di grandezza di differenza!
  - ▶ Nella nostra attività di programmatori software, ricordiamoci che la **cache chorence** può avere un grosso impatto nella performance dei programmi
    - programmi con pattern di accesso ai dati randomici hanno tempi di esecuzione più lenti di interi ordini di grandezza

---

Architettura degli elaboratori - 78 - Memoria cache



## Esercizio

In un sistema di memoria con due livelli di cache, si ha:

- hit rate di livello 1 = 90%, tempo di accesso 1 ns;
- hit rate di livello 2 = 90%, tempo di accesso 5 ns;
- tempo di accesso alla memoria principale: 100 ns.

Si calcoli il tempo medio di accesso a memoria.

Soluzione (in nanosecondi)

$$\begin{aligned} &0.9 \times 1 \\ &+ \\ &0.1 \times ( \\ &\quad 0.9 \times (1+5) \\ &\quad + \\ &\quad 0.1 \times (1+5+100) \\ &\quad ) \end{aligned}$$

---

Architettura degli elaboratori - 79 - Memoria cache