8' Esercitazione

https://politecnicomilano.webex.com/meet/gianenrico.conti

13 maggio 2021

Gian Enrico Conti

Gerarchia di memoria, Cache

Outline

- **Memorie Cache**
  - Struttura
  - Problemi
  - Tipologie di memorie cache
    - Indirizzamento diretto
    - Completamente associativa
    - Set associativa a N vie

Memorie Cache

- La **memoria cache** è una memoria **veloce** relativamente **piccola**, non visibile al software che memorizza i dati più recentemente usati della **memoria principale** del sistema.

Qualche rif. ai tempi (Intel 2016)

Table 1

L2 cache	256 KB	4 nanoseconds	1 TB/second Sometimes shared by two cores
L3 cache	8 MB or more	10x slower than L2	>400 GB/second
MCDRAM		2x slower than L3	400 GB/second
Main memory on DDR DIMMs	4 GB-1 TB	Similar to MCDRAM	100 GB/second
Main memory on Intel Omni-Path Fabric	Limited only by cost	Depends on distance	Depends on distance and hardware
I/O devices on memory bus	6 TB	100x-1000x slower than memory	25 GB/second
I/O devices on PCIe bus	Limited only by cost	From less than milliseconds to minutes	GB-TB/hour Depends on distance and hardware

Memorie Cache

- Il funzionamento della Cache Memory si basa principalmente su due *principi di località*:
  - **Località temporale**
    - Dati recentemente usati hanno un'alta probabilità di essere nuovamente usati a breve.
  - **Località spaziale**
    - Se un dato viene referenziato, è molto probabile che dati adiacenti siano a breve a loro volta acceduti.

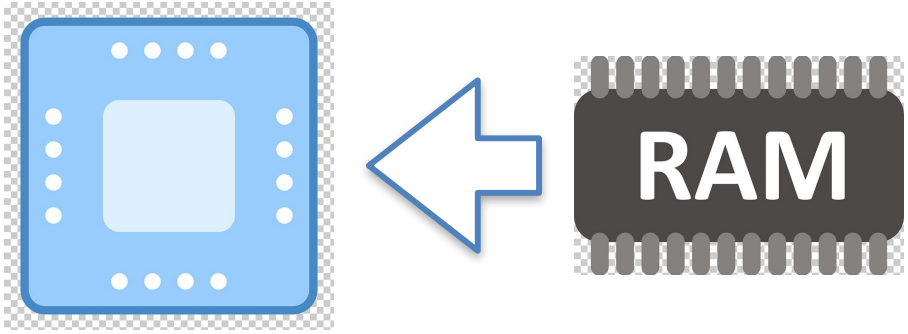
- Accesso al dato in Cache:
  - **Cache Hit** Dato presente in cache
  - **Cache Miss** Dato non presente in cache

Struttura di Cache

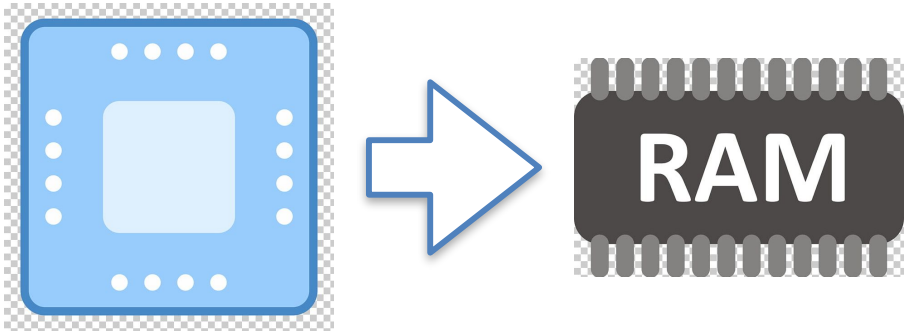
- Ogni linea di cache contiene, in aggiunta al blocco di dati:
  - **Tag** - identica univocamente un blocco in cache
  - **D (dirty bit)** - quando e settato ad 1, indica che il blocco in cache e stato modificato, e viceversa (facoltativo)
  - **V (validity bit)** - quando e settato ad 1, indica che il blocco in cache e valido, e viceversa

Problemi Essenziali

- **Dove (block placement)**
  - Dove caricare un blocco proveniente da un livello gerarchico inferiore?



- **Come (block identication)**
  - Come individuare un blocco in un livello gerarchico superiore?



Problemi Essenziali

- **Quale (block replacement)**
  - Quale blocco sostituire in caso di miss per fare posto ad un blocco del livello gerarchico sottostante?
    - FIFO
    - LRU
    - RANDOM
- **Politica di Scrittura (write policy)**
  - Come gestire le modiche dei blocchi?
    - Write back
    - Write through

Cache a Indirizzamento Diretto

- Ciascun blocco di RAM va mappato su un preciso blocco di cache
- Struttura dell'indirizzo di RAM:

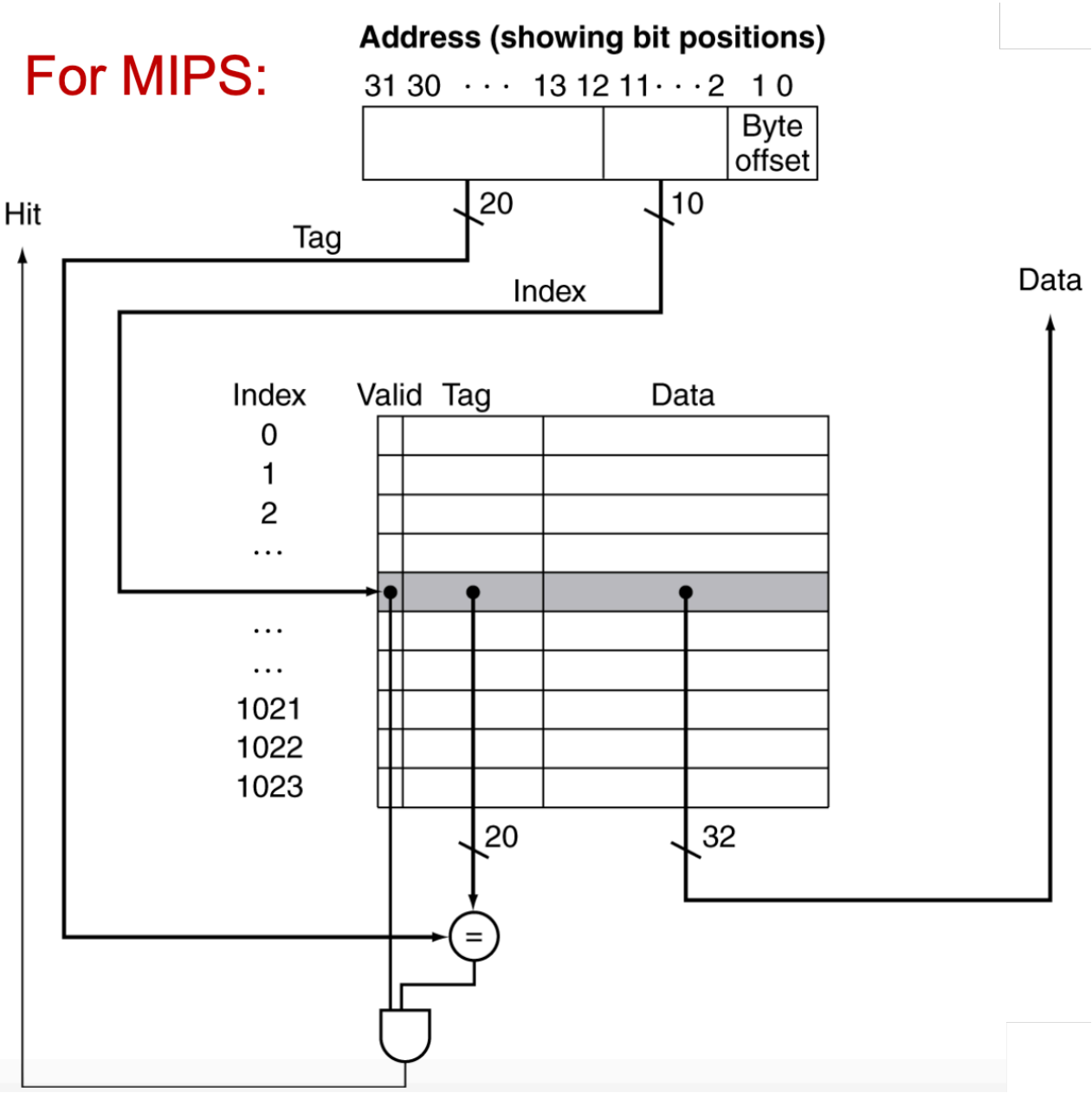
- **DOVE (block placement)**
  - (indirizzo blocco) MOD (numero blocchi in cache)
  - Equivale a considerare il campo INDICE dell'indirizzo in RAM

- **COME (block identication)**
  - Dato l'indirizzo in RAM, si considera il campo INDICE, che indichera il blocco di cache in cui cercare il dato. Se in quel blocco il tag e uguale al tag dell'indirizzo in RAM e il validity bit e settato ad 1, allora il dato e presente in cache ed e valido.

[Oss: il dato va cercato in una sola linea di cache!]

Cache a Indirizzamento Diretto MIPS

For MIPS:



Word: 4 byte

Tag: 20

Quindi..

Cache Completamente Associativa

- Ciascun blocco di RAM puo essere mappato in qualsiasi blocco di cache
- Struttura dell'indirizzo di RAM:

- **DOVE (block placement)**
  - In qualsiasi blocco di cache

- **COME (block identication)**
  - Dato l'indirizzo in RAM, si considera il campo TAG, che indichera il blocco di RAM da cercare in cache. Tale campo deve essere confrontato con l'omonimo campo in tutte le linee di cache. Se si trova un matching sul campo TAG e il validity bit e settato ad 1, allora il dato e presente in cache ed e valido.

[Oss: il dato va cercato in tutte le linee di cache!]

Cache Set Associativa

- Ciascun blocco di RAM va mappato in uno qualsiasi dei blocchi di un preciso set nella cache
- Struttura dell'indirizzo di RAM:

- **DOVE (block placement)**
  - (indirizzo blocco) MOD (numero set in cache)
  - Equivale a considerare il campo SET dell'indirizzo in RAM

- **COME (block identication)**
  - Dato l'indirizzo in RAM, si considera il campo SET, che indichera il set nella cache in cui cercare il dato. All'interno del set individuato, bisogna effettuare una ricerca associativa sul campo TAG di tutte le linee di cache in quel set. Se si trova un matching sul campo TAG e il validity bit e settato ad 1, allora il dato e presente in cache ed è valido. [Oss: il dato va cercato in tutti i blocchi di un solo set!]

Ex warm up 1 "Direct Mapped"

Dato un sistema con:

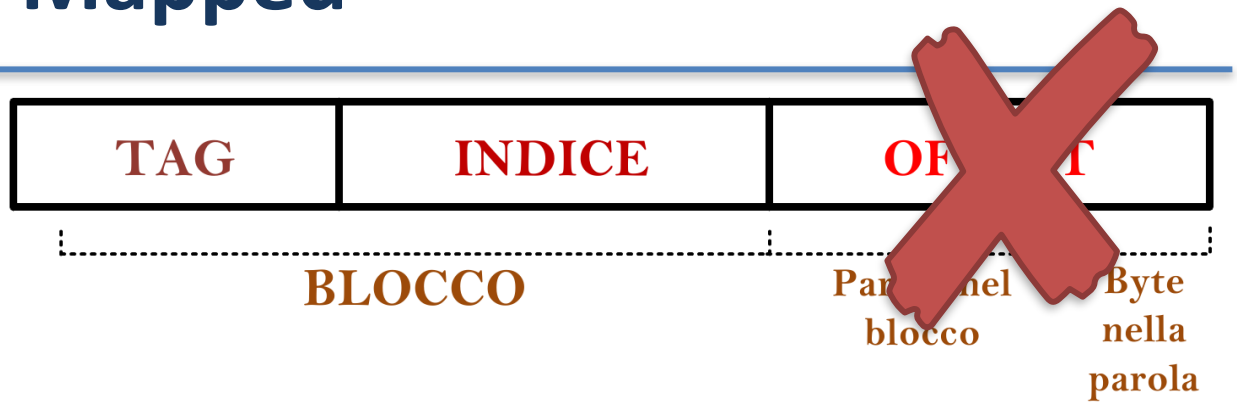
- 8 blocchi
- Ogni blocco da 1 byte (word == byte)

Calcolare in quale blocco ed in quale posizione viene mappato

l' indirizzo 22

Ex warm up 1 "Direct Mapped"

- 8 blocchi
- Ogni blocco da 1 byte (word == byte)



- non ho offset

- 8 blocchi = 2^3 = 3 bit x identificare blocco

Quindi indice = 3 bit

...

Ex warm up 1 "Direct Mapped"

- 8 blocchi
- Ogni blocco da 1 byte (word == byte)
- non ho offset
- 8 blocchi = 2^3 = 3 bit x identificare blocco

- Quindi indice = 3 bit

22 = 10 110 $_b$

10 110 blocco 6, nessuna pos. (Byte!)

Ex warm up 1 "Direct Mapped"

- 8 blocchi
- Ogni blocco da 1 byte (word == byte)
- non ho offset
- 8 blocchi = 2^3 = 3 bit x identificare blocco

- Quindi indice = 3 bit

22 = 10 110 $_b$

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	Mem[10110]
111	N		

10 110 blocco 6, nessuna pos. (Byte!)

Ex warm up 2 "Direct Mapped"

Dato un sistema a 32 bit con:

- 64 blocchi
- Ogni blocco da 16 bytes

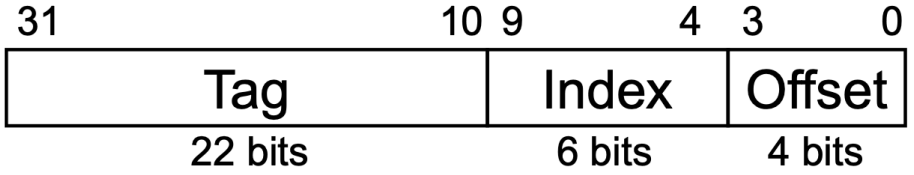
Calcolare in quale blocco ed in quale posizione viene mappato
l' indirizzo 1200

Ex warm up 2 "Direct Mapped"

Dato un sistema a 32 bit, con:

- 64 blocchi
- Ogni blocco da 16 bytes

Calcolare in quale blocco ed in quale posizione viene mappato
l' indirizzo 1200 = 0100 1011 0000



16 byte x blocco -> 4 bit x offset

64 blocchi -> 6 bits x Index

... 22 x tag.

Ex warm up 2 "Direct Mapped"

Dato un sistema a 32 bit, con:

- 64 blocchi
- Ogni blocco da 16 bytes

Calcolare in quale blocco ed in quale posizione viene mappato
l' indirizzo 1200 = 01 001011 0000

64 blocchi -> 6 bits x offset

16 byte x blocco -> 4 bit x offset

TOLGO Offset (i.e.) cancello 4 bit "bassi" i.e. SHIFTO x 4...

Index = 75

- "*(indirizzo blocco) MOD (numero blocchi in cache) "* = **75 MOD 64 = 11**

MOD: al max 64... e cerco...

Ex 1

Sia data un'architettura con:

- RAM da 1GB,

- cache da 1MB e blocchi da 512 byte, indirizzata al byte

1) Specificare la struttura dell'indirizzo di memoria nel caso di cache ad indirizzamento diretto, completamente associativa e set associativa a 4 vie.

2) Calcolare la dimensione totale della cache nei tre casi indicati al punto 1, sapendo che la cache `e stata ottimizzata con l'aggiunta del dirty bit.

3) Ipotizzando la cache inizialmente vuota, si consideri il caso in cui la CPU debba caricare in cache per 3 volte di seguito una sequenza di 800KB di dati, memorizzati in modo consecutivo a partire dall'inzirizzo 0 di RAM. Sapendo che la politica di sostituzione dei blocchi `e LRU, quale delle 3 soluzioni di cache indicate al punto 1 `e piu` conveniente?

Ex 1.1 direct address

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

RAM → 1GB = 2^{30} → 30 bit per l'indirizzo

CACHE → 1MB = 2^{20}

blocco → 512byte = 2^9 → 9 bit per l'offset

#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ → 21 bit per blocco in RAM

#blocchi in cache = dim.Cache/dim.Blocco = $2^{20}/2^9 = 2^{11}$ → 11 bit per Indice

Ex 1.1 direct address

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

RAM → 1GB = 2^{30} → 30 bit per l'indirizzo

CACHE → 1MB = 2^{20}

blocco → 512byte = 2^9 → 9 bit per l'offset

#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ → 21 bit per blocco in RAM

#blocchi in cache = dim.Cache/dim.Blocco = $2^{20}/2^9 = 2^{11}$ → 11 bit per Indice

#bit tag = #bit indirizzo – #bit offset – #bit indice = 30 – 9 – 11 = 10

Ex 1.1 direct address

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

RAM → 1GB = 2^{30} → 30 bit per l'indirizzo

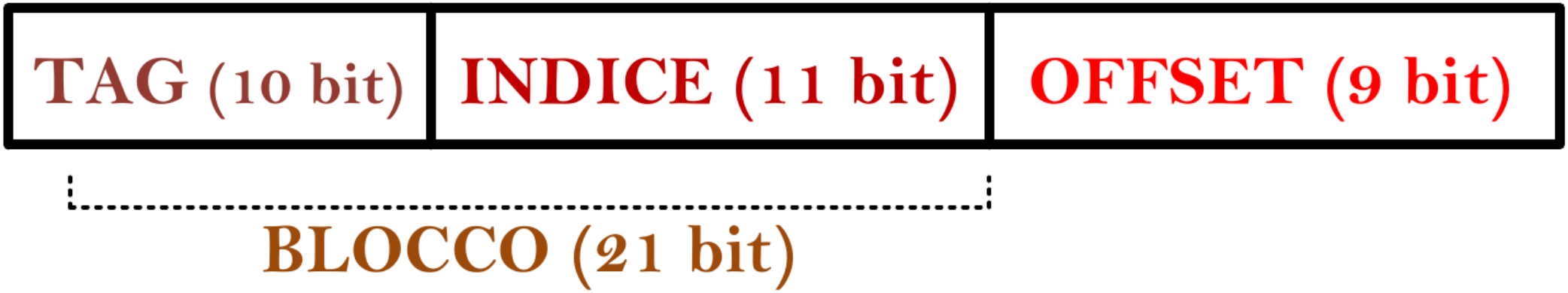
CACHE → 1MB = 2^{20}

blocco → 512byte = 2^9 → 9 bit per l'offset

#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ → 21 bit per blocco in RAM

#blocchi in cache = dim.Cache/dim.Blocco = $2^{20}/2^9 = 2^{11}$ → 11 bit per Indice

#bit tag = #bit indirizzo – #bit offset – #bit indice = 30 – 9 – 11 = 10



Ex 1.1 fully associative

Sia data un'architettura con: RAM da 1GB, cache da 1MB e blocchi da 512 byte, indirizzata al byte

RAM → 1GB = 2^{30} → 30 bit per l'indirizzo

CACHE → 1MB = 2^{20}

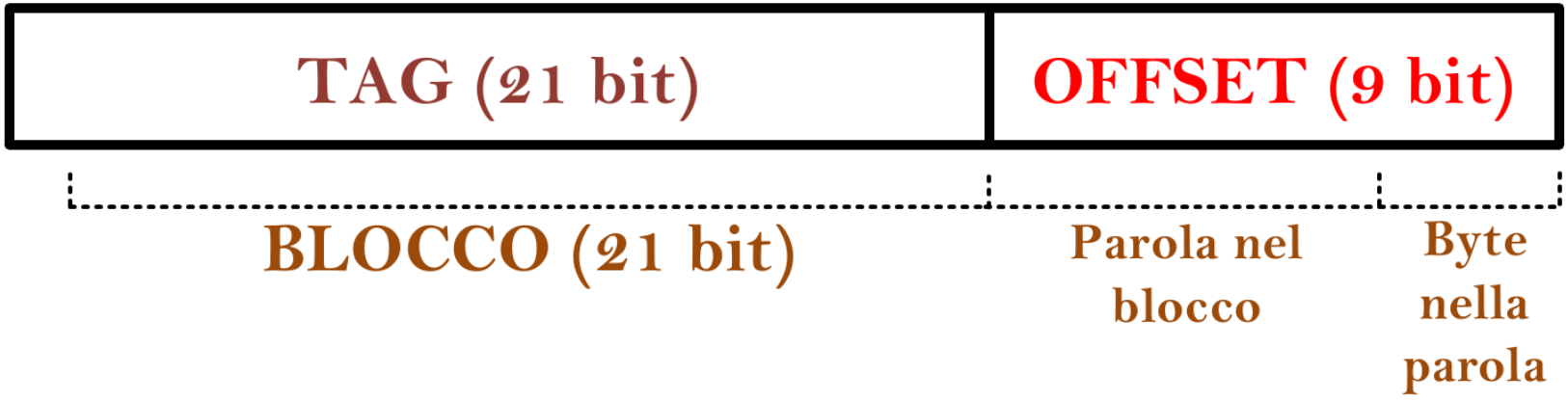
blocco → 512byte = 2^9 → 9 bit per l'offset

#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ → 21 bit per blocco in RAM

(Come sopra..)

NO INDEX..

#bit tag = #bit indirizzo – #bit offset – #bit indice = 30 – 9 = 21



Ex 1.1 fully associative 4 ways

Sia data un'architettura con: RAM da 1GB, cache da 1MB e blocchi da 512 byte, indirizzata al byte

RAM → 1GB = 2^{30} → 30 bit per l'indirizzo

CACHE → 1MB = 2^{20}

blocco → 512byte = 2^9 → 9 bit per l'offset

#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ → 21 bit per blocco in RAM

(Come sopra..)

$$\text{dim.set} = \#\text{vie} * \text{dim.blocco} = 2^2 * 2^9 = 2^{11}$$

Ex 1.1 fully associative 4 ways

Sia data un'architettura con: RAM da 1GB, cache da 1MB e blocchi da 512 byte, indirizzata al byte

RAM → 1GB = 2^{30} → 30 bit per l'indirizzo

CACHE → 1MB = 2^{20}

blocco → 512byte = 2^9 → 9 bit per l'offset

#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ → 21 bit per blocco in RAM

(Come sopra..)

dim.set = #vie ∗ dim.blocco = $2^2 * 2^9 = 2^{11}$

#set in cache = dim.cache/dim.set = $2^{20}/2^{11} = 2^9$ → 9 bit per set in cache

Ex 1.1 fully associative 4 ways

Sia data un'architettura con: RAM da 1GB, cache da 1MB e blocchi da 512 byte, indirizzata al byte

RAM → 1GB = 2^{30} → 30 bit per l'indirizzo

CACHE → 1MB = 2^{20}

blocco → 512byte = 2^9 → 9 bit per l'offset

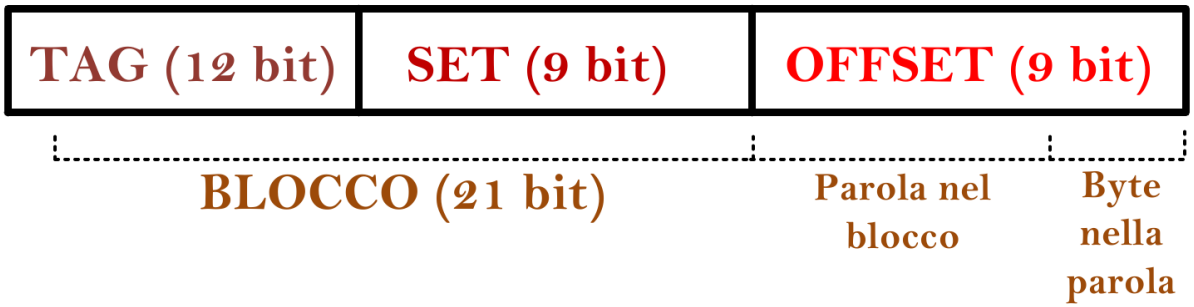
#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ → 21 bit per blocco in RAM

(Come sopra..)

dim.set = #vie ∗ dim.blocco = $2^2 * 2^9 = 2^{11}$

#set in cache = dim.cache/dim.set = $2^{20}/2^{11} = 2^9$ → 9 bit per set in cache

#bit tag = #bit indirizzo – #bit offset – #bit set = 30 – 9 – 9 = 12



Ex 1.2

Sia data un’architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

2) Calcolare la dimensione totale della cache nei tre casi indicati al punto 1, sapendo che la cache `e stata ottimizzata con l’aggiunta del dirty bit.

...

Ex 1.2 direct access Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

blocco → 512byte = 2^9 → 9 bit per l'offset

10 bit di tag.

#blocchi in cache = dim.Cache/dim.Blocco = $2^{20}/2^9 = 2^{11}$ → 11 bit per Indice

2 bit x Dirty e validity:

dim.tot. = #blocchi $*$ (dim.Blocco + dim.Tag + 2bit)

Ex 1.2 direct access Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

blocco → 512byte = 2^9 → 9 bit per l'offset

10 bit di tag.

#blocchi in cache = dim.Cache/dim.Blocco = $2^{20}/2^9 = 2^{11}$ → 11 bit per Indice

2 bit x Dirty e validity

dim.tot. = #blocchi $*$ (dim.Blocco + dim.Tag + 2bit)

dim.tot. (in byte) = $2^{11} * (2^9$**byte** + 10bit + 2bit)

Raccolgo...

Ex 1.2 direct access Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

blocco → 512byte = 2^9 → 9 bit per l'offset

10 bit di tag.

#blocchi in cache = dim.Cache/dim.Blocco = $2^{20}/2^9 = 2^{11}$ → 11 bit per Indice

2 bit x Dirty e validity

dim.tot. = #blocchi $*$ (dim.Blocco + dim.Tag + 2bit)

dim.tot. (in byte) = $2^{11} * (2^9$**byte** + 10bit + 2bit)

Raccolgo...

dim.tot. (in byte) = $2^{11} * (2^9$**byte** + 10bit + 2bit) =

2^{20} byte + $2^{11} *$ (10bit + 2bit) = ...

Ex 1.2 direct access Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

$$= 2^{20}\text{byte} + (2^{11}\text{byte} * 12\text{bit}) =$$

$$= 1\text{MB} + 2^{11} * (2^{3}\text{bit} + 2^{2}\text{bit}) =$$

$$= 1\text{MB} + 2^{14}\text{bit} + 2^{13}\text{bit} =$$

$$= 1\text{MB} + 2^{11}\text{byte} + 2^{10}\text{byte} =$$

$$= 1\text{MB} + 2\text{KB} + 1\text{KB} = 1024\text{KB} = 1,003\text{MB}$$

Ex 1.2 fully associative Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

$$\text{dim.tot.} = \#\text{blocchi} * (\text{dim.Blocco} + \text{dim.Tag} + 2\text{bit})$$

Era:

CACHE $\rightarrow$ 1MB = 2^{20}

blocco $\rightarrow$ 512byte = 2^9 $\rightarrow$ 9 bit per l'offset

#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ $\rightarrow$ 21 bit per blocco in RAM

Ex 1.2 fully associative Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

CACHE → 1MB = 2^{20}

blocco → 512byte = 2^9 → 9 bit per l'offset

#blocchi in RAM = dim.RAM/dim.Blocco = $2^{30}/2^9 = 2^{21}$ → 21 bit per blocco in RAM

$$\text{dim.tot.} = \#\text{blocchi} * (\text{dim.Blocco} + \text{dim.Tag} + 2\text{bit})$$

$$= 2^{11} * (2^9\text{byte} + 21\text{bit} + 2\text{bit})$$

Ex 1.2 fully associative Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

$$\text{dim.tot.} = \#\text{blocchi} * (\text{dim.Blocco} + \text{dim.Tag} + 2\text{bit})$$

$$= 2^{11} * (2^{9}\text{byte} + 21\text{bit} + 2\text{bit}) =$$

$$= 2^{20}\text{byte} + (2^{11} * 23\text{bit}) =$$

.. un po' di passaggi...

$$= 1\text{MB} + 2^{11} * (2^{4}\text{bit} + 2^{2}\text{bit} + 2\text{bit} + 1\text{bit}) =$$

$$= 1\text{MB} + 2^{15}\text{bit} + 2^{13}\text{bit} + 2^{12}\text{bit} + 2^{11}\text{bit} =$$

$$= 1\text{MB} + 2^{12}\text{byte} + 2^{10}\text{byte} + 2^{9}\text{byte} + 2^{8}\text{byte} =$$

$$= 1\text{MB} + 4\text{KB} + 1\text{KB} + 512\text{byte} + 256\text{byte} =$$

$$= 1005,768\text{KB} \approx 1,006\text{MB}$$

Ex 1.2 fully associative Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

.. un po' di passaggi...

$= 1\text{MB} + 2^{11} * (2^4\text{bit} + 2^2\text{bit} + 2\text{bit} + 1\text{bit}) =$

$= 1\text{MB} + 2^{15}\text{bit} + 2^{13}\text{bit} + 2^{12}\text{bit} + 2^{11}\text{bit} =$

$= 1\text{MB} + 2^{12}\text{byte} + 2^{10}\text{byte} + 2^9\text{byte} + 2^8\text{byte} =$

$= 1\text{MB} + 4\text{KB} + 1\text{KB} + 512\text{byte} + 256\text{byte} =$

$= 1005,768\text{KB} \approx \mathbf{1,006MB}$

Ex 1.2 - 4 ways associative Cache Size

Sia data un’architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

2) Calcolare la dimensione totale della cache nei tre casi indicati al punto 1, sapendo che la cache `e stata ottimizzata con l’aggiunta del dirty bit.

...

Ex 1.2 - 4 ways associative Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

2) Calcolare la dimensione totale della cache nei tre casi indicati al punto 1, sapendo che la cache `e stata ottimizzata con l'aggiunta del dirty bit.

$$\text{dim.tot.} = \#\text{blocchi} * (\text{dim.Blocco} + \text{dim.Tag} + 2\text{bit}) =$$

per questo caso valeva:

$$\text{dim.set} = \#\text{vie} * \text{dim.blocco} = 2^2 * 2^9 = 2^{11}$$

$$\#\text{set in cache} = \text{dim.cache/dim.set} = 2^{20}/2^{11} = 2^9 \rightarrow 9 \text{ bit per set in cache}$$

...

Ex 1.2 - 4 ways associative Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

dim.set = #vie $*$ dim.blocco = $2^2 * 2^9 = 2^{11}$

#set in cache = dim.cache/dim.set = $2^{20}/2^{11} = 2^9 \rightarrow$ 9 bit per set in cache

#bit tag = ... = 12

$$\text{dim.tot.} = \#\text{set} * (\text{dim.Blocco} + \text{dim.Tag} + 2\text{bit}) =$$

$$= 2^{11} * (2^9\text{byte} + 12\text{bit} + 2\text{bit}) =$$

...

Ex 1.2 - 4 ways associative Cache Size

Sia data un'architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

dim.set = #vie $*$ dim.blocco = $2^2 * 2^9 = 2^{11}$

#set in cache = dim.cache/dim.set = $2^{20}/2^{11} = 2^9 \rightarrow$ 9 bit per set in cache

#bit tag = ... = 12

$$\text{dim.tot.} = \#\text{set} * (\text{dim.Blocco} + \text{dim.Tag} + 2\text{bit}) =$$

$$= 2^{11} * (2^9\text{byte} + 12\text{bit} + 2\text{bit}) =$$

$$= 2^{20}\text{byte} + (2^{11} * 14\text{bit}) =$$

$$= 1\text{MB} + 2^{11} * (2^3\text{bit} + 2^2\text{bit} + 2\text{bit}) =$$

$$= 1\text{MB} + 2^{14}\text{bit} + 2^{13}\text{bit} + 2^{12}\text{bit} =$$

$$= 1\text{MB} + 2^{11}\text{byte} + 2^{10}\text{byte} + 2^9\text{byte} =$$

$$= 1\text{MB} + 2\text{KB} + 1\text{KB} + 512\text{byte} =$$

$$= 1003,512\text{KB} \approx 1,035\text{MB}$$

Ex 1.3

Sia data un’architettura con: - RAM da 1GB, - cache da 1MB e blocchi da 512 byte, indirizzata al byte

3) Ipotizzando la cache inizialmente vuota, si consideri il caso in cui la CPU debba caricare in cache per 3 volte di seguito una sequenza di 800KB di dati, memorizzati in modo consecutivo a partire dall’inzirizzo 0 di RAM. Sapendo che la politica di sostituzione dei blocchi `e LRU, quale delle 3 soluzioni di cache indicate al punto 1 `e piu` conveniente?

...

Ex 1.3

3) Ipotizzando la cache inizialmente vuota, si consideri il caso in cui la CPU debba caricare in cache per 3 volte di seguito una sequenza di 800KB di dati, memorizzati in modo consecutivo a partire dall'indirizzo 0 di RAM. Sapendo che la politica di sostituzione dei blocchi `e LRU, quale delle 3 soluzioni di cache indicate al punto 1 `e piu` conveniente?

OSS:

a) va calcolato il cache miss (inizio "vuota)

b) x ogni caso blocco = 512byte = 2^9 → 9 bit per l'offset

Calcoliamo quanti blocchi...

Ex 1.3

3) Ipotizzando la cache inizialmente vuota, si consideri il caso in cui la CPU debba caricare in cache per 3 volte di seguito una sequenza di 800KB di dati, memorizzati in modo consecutivo a partire dall'indirizzo 0 di RAM. Sapendo che la politica di sostituzione dei blocchi `e LRU, quale delle 3 soluzioni di cache indicate al punto 1 `e piu` conveniente?

OSS:

a) va calcolato il cache miss (inizio "vuota)

b) x ogni caso blocco = 512byte = 2^9 → 9 bit per l'offset

Calcoliamo quanti blocchi...

= 800KB/512byte = 1562.5

→ 1563 blocchi dalla RAM alla cache.

.. Ci stanno in CACHE?

Ex 1.3

3) Ipotizzando la cache inizialmente vuota, si consideri il caso in cui la CPU debba caricare in cache per 3 volte di seguito una sequenza di 800KB di dati, memorizzati in modo consecutivo a partire dall'indirizzo 0 di RAM. Sapendo che la politica di sostituzione dei blocchi `e LRU, quale delle 3 soluzioni di cache indicate al punto 1 `e piu` conveniente?

OSS:

a) va calcolato il cache miss (inizio "vuota)

b) x ogni caso blocco = 512byte = 2^9 → 9 bit per l'offset

blocchi = 800KB/512byte = → 1563 blocchi dalla RAM alla cache.

Ci stanno in CACHE?

La cache puo' contenere fino a:

1MB/512byte = 1953, 125 → 1953 blocchi , OK!

Ex 1.3 direct access

Prima sequenza di caricamento:

Tutti i 1563 blocchi causano **miss** in quanto la cache era inizialmente vuota

Nota: tutti i blocchi hanno una precisa posizione in cache, e andranno memorizzati dalla linea 0 alla linea 1562 di cache.

Ex 1.3 direct access

Prima sequenza di caricamento:

Tutti i 1563 blocchi causano **miss** in quanto la cache era inizialmente vuota

2' sequenza di caricamento:

I 1563 blocchi sono gi`a in cache → nessun miss

3' sequenza di caricamento:

I 1563 blocchi sono gi`a in cache → nessun miss

Ex 1.3 direct access

1' sequenza di caricamento:

Tutti i 1563 blocchi causano **miss** in quanto la cache era inizialmente vuota

2' sequenza di caricamento:

I 1563 blocchi sono gi`a in cache → nessun miss

3' sequenza di caricamento:

I 1563 blocchi sono gi`a in cache → nessun miss

Totale: 1563 miss

Ex 1.3 fully associative

Prima sequenza di caricamento:

Tutti i 1563 blocchi causano **miss** in quanto la cache era inizialmente vuota.

I blocchi possono essere mappati in qualsiasi posizione in cache.

Supponendo di memorizzare ciascun blocco di RAM nella prima linea di cache libera, abbiamo 1563 miss

“a freddo ”.

Ex 1.3 fully associative

1' sequenza di caricamento:

Tutti i 1563 blocchi causano **miss** in quanto la cache era inizialmente vuota.

I blocchi possono essere mappati in qualsiasi posizione in cache.

Supponendo di memorizzare ciascun blocco di RAM nella prima linea di cache libera, abbiamo 1563 miss

“a freddo ”.

Seconda e terza sequenza di caricamento:
I 1563 blocchi sono gi`a in cache → nessun miss

Totale: 1563 miss

Ex 1.3 4 ways..

Prima sequenza di caricamento:

Tutti i 1563 blocchi causano **miss** in quanto la cache era inizialmente vuota

Ex 1.3 4 ways..

Nota:

i blocchi possono essere mappati in qualsiasi posizione di un **preciso** set in cache.

In cache ci sono: $2^9 = 512$ set.

Quindi, i blocchi di RAM 0, 512, 1024 e 1536 vanno nel set 0, ...

Ex 1.3 4 ways..

Nota:

i blocchi possono essere mappati in qualsiasi posizione di un **preciso** set in cache. In cache ci sono: $2^9 = 512$ set.

blocchi di RAM 0, 512, 1024 e 1536 vanno nel set 0

i blocchi di RAM 26, 538, 1050 e **1562** vanno nel set 26

i blocchi di RAM 27, 539 e 1051 vanno nel set 511

...

(1562 ultimo)

Ex 1.3 4 ways..

Nota:

..

Sui set dallo 0 al 26 sono mappati 4 blocchi di RAM, mentre dal 27 al 511 sono mappati 3 blocchi di RAM.

Supponendo di memorizzare ciascun blocco di RAM nella prima linea di cache libera del set associato, abbiamo 1563 miss a freddo.

Ex 1.3 4 ways..

Seconda e terza sequenza di caricamento:

I 1563 blocchi sono già in cache → nessun miss

Totale: 1563 miss

Ex 1.3 4 final thoughts

NOTA:

era possibile arrivare alla stessa soluzione osservando che la quantità di dati da caricare in cache (800 KB)

è inferiore alla capacità totale della cache

quindi, a prescindere dalla politica scelta, a partire dalla 2' sequenza di caricamento, i dati sono già **tutti** in cache.

In conclusione, in questo caso specifico le tre soluzioni di cache sono equivalenti in quanto causano lo stesso numero di cache miss.

Ex 2

Sia data un’architettura composta da una memoria RAM da 4KB, cache da 512 byte e blocchi da 128 byte.

Si consideri la seguente sequenza di richieste di lettura (R) e scrittura (W) della memoria:

R 000000010010

R 000000010011

R 100100010100

W 000011111111

W 111110001010

W 011110010010

W 110000000110

Ex 2

Specificare il contenuto della cache a seguito delle richieste indicate, nei seguenti casi:

1 cache ad indirizzamento diretto, politica di sostituzione FIFO e politica di scrittura write back (con dirty bit). Cosa cambierebbe cambiando la politica di sostituzione in questo caso?

2 cache completamente associativa, politica di sostituzione LRU e politica di scrittura write back (con dirty bit)

3 cache set associativa a 2 vie, politica di sostituzione RANDOM e politica di scrittura write through

Ex 2

Iniziamo con un po' di calcoli:

RAM da 4KB → 12 bit per indirizzo
blocchi da 128 byte = 2^7 byte → 7 bit per offset

Ex 2

Iniziamo con un po' di calcoli:

RAM da 4KB → 12 bit per indirizzo
blocchi da 128 byte → 7 bit per offset

#Blocchi in cache = dim.Cache/dim.Blocco = 512/128 =

$$2^9/2^7 = 2^2$$

→ 2 bit per l'indice nel caso di cache ad **indirizzamento diretto**.

Ex 2

Iniziamo con un po' di calcoli:

dim.Set=#vie$*$dim.Blocco = $2*128==2*2^7=2^8$

set in cache = dim.Cache/dim.Set = $2^9/2^8$ = 2

→ 1 bit per set nel caso di cache set-associativa a 2 vie

OSS: nulla da calcolare x fully assoc.

Ex 2: Struttura indirizzo - tag indici offset

cache ad indirizzamento diretto: Tag=3bit, Indice=2bit, Offset=7bit

cache completamente associativa: Tag=5bit, Offset=7bit *(no index..)*

cache set-associativa a 2 vie: Tag=4bit, Set=1bit, Offset=7bit

Soliti tricks coi bit disponibili... 12 bit.

Ex 2: Struttura indirizzo - tag indici offset

cache ad indirizzamento diretto: Tag=3bit, Indice=2bit, Offset=7bit

cache completamente associativa: Tag=5bit, Offset=7bit *(no index..)*

cache set-associativa a 2 vie: Tag=4bit, Set=1bit, Offset=7bit

Soliti tricks coi bit disponibili... 12 bit.

Ex 2:

Cache ad indirizzamento diretto, politica di sostituzione FIFO e politica di scrittura write back

		BLOCCO 0				BLOCCO 1				BLOCCO 2				BLOCCO 3				
Richiesta	E	V	D	TAG	dato	V	D	TAG	dato	V	D	TAG	dato	V	D	TAG	dato	NOTE
R 000000010010	M	1	0	000	0	0	0	-	-	0	0	-	-	0	0	-	-	
R 000000010011	H	1	0	000	0	0	0	-	-	0	0	-	-	0	0	-	-	
R 100100010100	M	1	0	000	0	0	0	-	-	1	0	100	18	0	0	-	-	
W 000011111111	M	1	0	000	0	1	1	000	1	1	0	100	18	0	0	-	-	
W 111110001010	M	1	0	000	0	1	0	000	1	0	0	100	18	1	1	111	31	
W 011110010010	M	1	0	000	0	1	0	000	1	0	0	100	18	1	1	011	15	Copia blocco 31 in RAM
W 110000000110	M	1	1	110	24	1	0	000	1	0	0	100	18	1	1	111	15	

Ex 2:

Cache completamente associativa, politica di sostituzione LRU e politica di scrittura write back

		BLOCCO 0				BLOCCO 1				BLOCCO 2				BLOCCO 3				
Richiesta	E	V	D	TAG	dato	V	D	TAG	dato	V	D	TAG	dato	V	D	TAG	dato	NOTE
R 000000010010	M	1	0	00000	0	0	0	-	-	0	0	-	-	0	0	-	-	
R 000000010011	H	1	0	00000	0	0	0	-	-	0	0	-	-	0	0	-	-	
R 100100010100	M	1	0	00000	0	1	0	10010	18	0	0	-	-	0	0	-	-	
W 000011111111	M	1	0	00000	0	1	0	10010	18	1	1	00001	1	0	0	-	-	
W 111110001010	M	1	0	00000	0	1	0	10010	18	1	1	00001	1	1	1	111	31	
W 011110010010	M	1	1	00000	15	1	0	10010	18	0	0	00001	1	1	1	111	31	(LRU)
W 110000000110	M	1	1	00000	15	1	1	11000	24	0	0	00001	1	1	1	111	15	(LRU)

Ex 2:

Cache set associativa a 2 vie, politica di sostituzione RANDOM e politica di scrittura write through

		SET 0						SET 1						
		BLOCCO 0			BLOCCO 1			BLOCCO 2			BLOCCO 3			
Richiesta	E	V	TAG	dato	V	TAG	dato	V	TAG	dato	V	TAG	dato	NOTE
R 000000010010	M	1	0000	0	0	-	-	0	-	-	0	-	-	
R 000000010011	H	1	0000	0	0	-	-	0	-	-	0	-	-	
R 100100010100	M	1	0000	0	1	1001	18	0	-	-	0	-	-	
W 000011111111	M	1	0000	0	1	1001	18	1	0000	1	0	-	-	Copia blocco 1 in RAM
W 111110001010	M	1	0000	0	1	1001	18	1	0000	1	1	1111	31	Copia blocco 31 in RAM
W 011110010010	M	1	0000	0	1	1001	18	1	0111	15	1	1111	31	Copia blocco 15 in RAM
W 110000000110	M	1	1100	24	1	1001	18	1	0111	15	1	1111	31	Copia blocco 24 in RAM

Ex 3:

Data la seguente tabella:

	L1	L2
SIZE	32 KB	1MB
HitRate	80%	90%
HitTime	1ns	27ns
MissPenalty	10ns	750 ns

Calcolare il tempo di accesso TA nel caso di architettura rispettivamente con singolo e doppio livello di cache.

2 Quale MissPenalty deve avere L1 perchè il tempo di accesso dell'architettura con singolo livello di cache sia pari a 13ns, lasciando invariati tutti gli altri parametri?

3 Quale HitRate deve avere L2 perch`e il tempo di accesso dell'architettura con due livelli di cache sia pari a 4 ns, lasciando invariati tutti gli altri parametri?

Ex 3:

Con singolo livello di cache:

	L1	L2
SIZE	32 KB	1MB
HitRate	80%	90%
HitTime	1ns	27ns
MissPenalty	10ns	750 ns

$$T_A = \text{hit time}_{L1} + \text{miss rate}_{L1} * \text{miss penalty}_{L1}$$

$$= 1\text{ns} + (1 - 0.8) * 10\text{ns} = 3\text{ns}$$

Ex 3:

Con doppio livello di cache:

	L1	L2
SIZE	32 KB	1MB
HitRate	80%	90%
HitTime	1ns	27ns
MissPenalty	10ns	750 ns

$$T_A = \text{hit time}_{L1} + \text{miss rate}_{L1} * (\text{hit time}_{L2} + \text{miss rate}_{L2} * \text{miss penalty}_{L2}) =$$

$$1ns + 0.2 * (27ns + 0.1 * 750ns)$$

$$= 21.4ns$$

Ex 3.2:

Quale MissPenalty deve avere L1 perchè il tempo di accesso dell'architettura con singolo livello di cache sia pari a 13ns, lasciando invariati tutti gli altri parametri?

	L1	L2
SIZE	32 KB	1MB
HitRate	80%	90%
HitTime	1ns	27ns
MissPenalty	10ns	750 ns

$$T_A = \text{hit time}_{L1} + \text{miss rate}_{L1} * \text{miss penalty}_{L1} = 13\text{ns} \rightarrow$$

$$1\text{ns} + (1 - 0.8) * \text{miss penalty}_{L1} = 13\text{ns} \rightarrow$$

$$\text{miss penalty}_{L1} = 12/0.2 = 60\text{ns}$$

Ex 3.3:

Quale HitRate deve avere L2 perchè il tempo di accesso dell'architettura con due livelli di cache sia pari a 4 ns, lasciando invariati tutti gli altri parametri?

	L1	L2
SIZE	32 KB	1MB
HitRate	80%	90%
HitTime	1ns	27ns
MissPenalty	10ns	750 ns

Imponiamo 4 ns nella formula:

$$T_A = \text{hit time}_{L1} + \text{miss rate}_{L1} * (\text{hit time}_{L2} + \text{miss rate}_{L2} * \text{miss penalty}_{L2}) = \mathbf{4ns}$$

Ex 3.3:

Quale HitRate deve avere L2 perchè il tempo di accesso dell'architettura con due livelli di cache sia pari a 4 ns, lasciando invariati tutti gli altri parametri?

	L1	L2
SIZE	32 KB	1MB
HitRate	80%	90%
HitTime	1ns	27ns
MissPenalty	10ns	750 ns

Imponiamo 4 ns nella formula:

$$T_A = \text{hit time}_{L1} + \text{miss rate}_{L1} * (\text{hit time}_{L2} + \text{miss rate}_{L2} * \text{miss penalty}_{L2}) = \mathbf{4ns}$$

Sostituiamo:

$$1ns + 0.2 * (27ns + (1 - \text{hit rate}_{L2}) * 750ns) = 4ns$$

$$27ns + (1 - \text{hit rate}_{L2}) * 750ns = 1 \ ..$$

Altri passaggi..

Ex 3.3:

	L1	L2
SIZE	32 KB	1MB
HitRate	80%	90%
HitTime	1ns	27ns
MissPenalty	10ns	750 ns

$\text{hit rate}_{L2} = 1 + 12/750 > 100\%!!!$

Quindi non è possibile ottenere un tempo di accesso

con doppio livello di cache pari a 4 ns modificando solo l'hit rate L2.

Ex 4:

Sia data un'architettura composta da CPU e cache, avente miss penalty pari a 6 cicli di clock, CPI ideale pari a 8.5 cicli di clock e miss rate pari a 11%.

Supponendo che per ciascuna istruzione siano necessari 2 accessi in memoria per recuperare i dati, calcolare il CPI reale, lo speedup del sistema ideale rispetto a quello reale, il CPI del sistema privo di cache e lo speedup del sistema privo di cache rispetto a quello dotato di cache.

Ex 4:

Sia data un'architettura composta da CPU e cache, avente miss penalty pari a 6 cicli di clock, CPI ideale pari a 8.5 cicli di clock e miss rate pari a 11%. Supponendo che per ciascuna istruzione siano necessari 2 accessi in memoria per recuperare i dati, calcolare il CPI reale, lo speedup del sistema ideale rispetto a quello reale, il CPI del sistema privo di cache e lo speedup del sistema privo di cache rispetto a quello dotato di cache.

Soluzione:

$$CPI_{REALE} = \text{CPI IDEALE} + \text{miss rate} * \text{miss penalty} * \#\text{riferimenti memoria} =$$

$$8.5 + 0.11 * 6 * 3 = 10.48$$

$$speedup_{\text{ideale vs reale}} = (CPI_{REALE} * IC * f)/(CPI_{IDEALE} * IC * f) =$$

$$10.48/8.5 \approx 1.23$$

$$CPI_{NO\ CACHE} = \text{CPI IDEALE} + \text{miss penalty} * \#\text{riferimenti memoria} = 8.5 + 6 * 3 = 26.5$$

$$speedup_{\text{nocache vs reale}} = (CPI_{REALE} * IC * f) / (CPI_{NO\ CACHE} * IC * f) = 10.48/26.5 \approx 0.39$$

Ex 5:

Sia data un'architettura composta da una memoria RAM da 4Mparole, indirizzata a parole, una cache da 1Kparola e blocchi da 64 parole, dove ogni parola è composta da 4 byte.

A) Strutturare gli indirizzi di memoria nel caseo di cache ad indirizzamento diretto, cache completamente associativa e cache set associativa a 2 vie.

B) Ipotizzando la cache inizialmente vuota, si consideri il caso in cui la CPU debba caricare in cache 1025 parole memorizzate in modo adiacente nella memoria RAM a partire dall'indirizzo 0, ripetendo la sequenza di caricamento per 5 volte consecutive. Calcolare il numero di cache miss in caso di cache ad indirizzamento diretto e di cache completamente associativa, con politica di sostituzione LRU.

Ex 5:

1' parte:

dim.RAM = 4Mparole = 2^{22}parole → 22 bit per l'indirizzo

dim.Cache = 1Kparola = 2^{10}parole

dim.Blocco = 64parole = 2^{6}parole → 6 bit per offset

#blocchi in$_c$ache = $2^{10}/2^{6} = 2^{4}$ → 4 bit per campo indice in caso di cache ad indirizzamento diretto

dim.Set =$2*2^{6} = 2^{7}$

#set in cache = $2^{10}/2^{7} = 2^{3}$ → 3 bit per campo set in caso di cache set-associativa a 2 vie

Struttura dell'indirizzo in presenza di cache ad inbdirizzamento diretto: Tag=12 bit, Indice=4 bit, Offset=6 bit

Struttura dell'indirizzo in presenza di cache completamente associativa: Tag=16 bit, Offset=6 bit

Struttura dell'indirizzo in presenza di cache set-associativa a 2 vie: Tag=13 bit, Set=3 bit, Offset=6 bit

Ex 5:

2' parte:

Bisogna caricare 1025 parole, memorizzate a partire dall'indirizzo 0 di memoria, per 5 volte consecutive.

1025 parole = 1024 + 1 parole = (210 + 1) parole

Dato che ciascun blocco `e grande 26 parole:
210/26 = 24 → 210 parole stanno in 24 blocchi, piu` un blocco per l'ultima parola.

Se la cache `e ad indirizzamento diretto risulta:

1o ciclo: 1024 miss a freddo + 1 miss per sostituzione (la 1025a parola prende il posto della prima)

2o , 3o , 4o , 5o ciclo: 1 miss per la prima parola che prende il posto della 1025a + un miss per la 1025a parola che prende il posto della prima

Totale: 1025 + 2*4 = 1033 miss

Ex 5 Osservazioni:

Quando la memoria è indirizzata a word, la parola è la più piccola unità di memoria indirizzabile. Questa soluzione ha il vantaggio di utilizzare indirizzi di memoria piu` compatti, che occuperanno meno spazio nella memoria istruzioni.

Quando invece la memoria `e indirizzata al byte, l'unit`a di risoluzione della memoria `e il byte. Le parole possono comunque essere indirizzate, ma l'indirizzo di memoria richiede alcuni bit aggiuntivi rispetto al caso precedente. L'approccio piu` diffuso `e quello in cui la dimensione di una parola `e un multiplo intero della dimensione del byte.