SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Architettura degli Elaboratori
18 – Cache multilivello

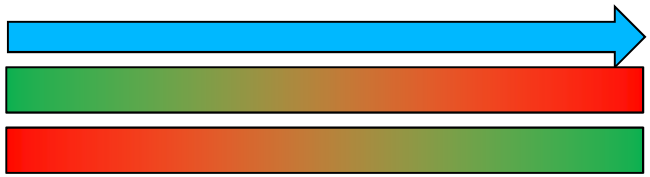
Prof. Andrea Sterbini – sterbini@di.uniroma1.it

Argomenti

- **Argomenti della lezione**
  - Cache multilivello
  - Esercizi

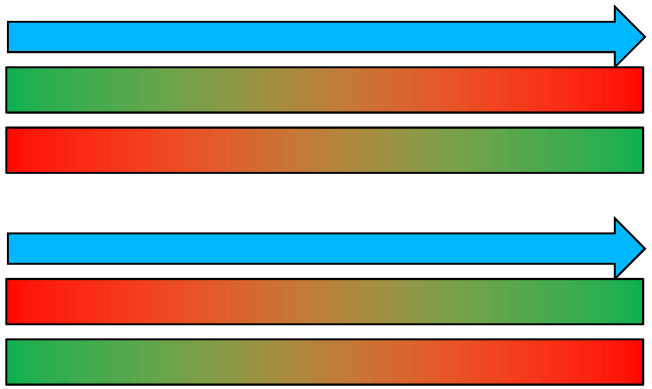
Argomenti

- **Argomenti della lezione**
  - Cache multilivello
  - Esercizi
- **Influenza dei parametri della cache**
- Maggior dimensione del blocco
  - Maggior tempo di MISS (trasf. da RAM)
  - Minor percentuale di MISS di tipo **cold start**



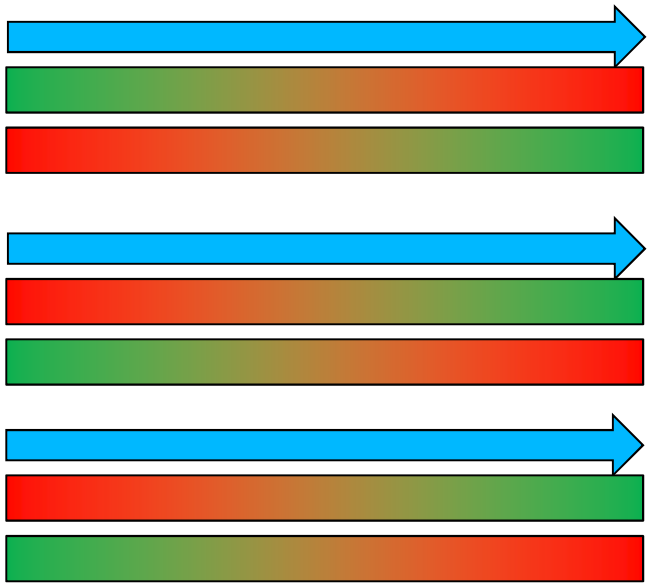
Argomenti

- **Argomenti della lezione**
  - Cache multilivello
  - Esercizi
- **Influenza dei parametri della cache**
- Maggior dimensione del blocco
  - Maggior tempo di MISS (trasf. da RAM)
  - Minor percentuale di MISS di tipo **cold start**
- Maggior dimensione della cache (# di linee)
  - Minor percentuale di MISS di **capacità**
  - Maggior costo (più memoria)



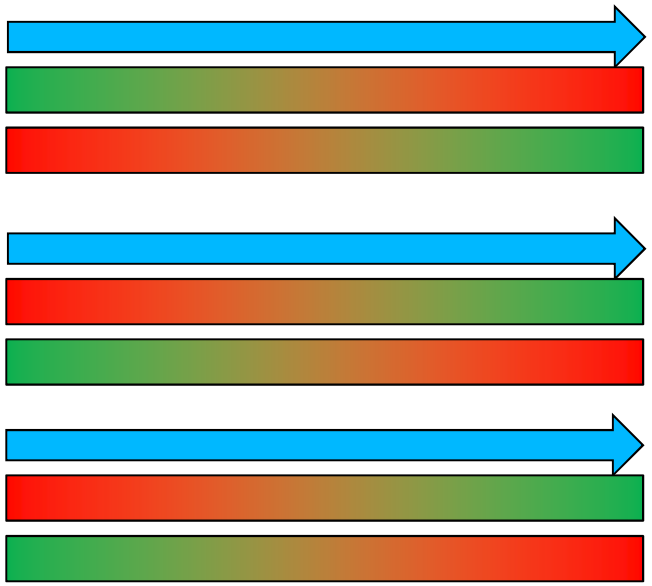
Argomenti

- **Argomenti della lezione**
  - Cache multilivello
  - Esercizi
- **Influenza dei parametri della cache**
- Maggior dimensione del blocco
  - Maggior tempo di MISS (trasf. da RAM)
  - Minor percentuale di MISS di tipo **cold start**
- Maggior dimensione della cache (# di linee)
  - Minor percentuale di MISS di **capacità**
  - Maggior costo (più memoria)
- Maggior Associatività (# di vie)
  - Minor percentuale di MISS di **conflitto**
  - Maggior costo (più comparatori)



Argomenti

- **Argomenti della lezione**
  - Cache multilivello
  - Esercizi
- **Influenza dei parametri della cache**
- Maggior dimensione del blocco
  - Maggior tempo di MISS (trasf. da RAM)
  - Minor percentuale di MISS di tipo **cold start**
- Maggior dimensione della cache (# di linee)
  - Minor percentuale di MISS di **capacità**
  - Maggior costo (più memoria)
- Maggior Associatività (# di vie)
  - Minor percentuale di MISS di **conflitto**
  - Maggior costo (più comparatori)



- **IDEA:** migliorare le performance del sistema usando **più livelli di cache** ottimizzate con parametri diversi (e costi diversi)

Diversa associatività

- 1 via, 4 set per via

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M					

Diversa associatività

- 1 via, 4 set per via

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

- 2 vie, 2 set per via

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

- 2 vie, 2 set per via

Blocchi da 1 word, 4 linee, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
H/M					

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

- 2 vie, 2 set per via

Blocchi da 1 word, 4 linee, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
H/M	M cold	M cold	HIT	M cold	M conf.

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

- 2 vie, 2 set per via
- **MISS = 80%**

Blocchi da 1 word, 4 linee, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
H/M	M cold	M cold	HIT	M cold	M conf.

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

- 2 vie, 2 set per via
- **MISS = 80%**

Blocchi da 1 word, 4 linee, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
H/M	M cold	M cold	HIT	M cold	M conf.

- 4 vie, 1 set per via

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

- 2 vie, 2 set per via
- **MISS = 80%**

Blocchi da 1 word, 4 linee, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
H/M	M cold	M cold	HIT	M cold	M conf.

- 4 vie, 1 set per via

Blocchi da 1 word, 4 linee, 4 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
tag	0	8	0	6	8
H/M					

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

- 2 vie, 2 set per via
- **MISS = 80%**

Blocchi da 1 word, 4 linee, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
H/M	M cold	M cold	HIT	M cold	M conf.

- 4 vie, 1 set per via

Blocchi da 1 word, 4 linee, 4 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
tag	0	8	0	6	8
H/M	M cold	M cold	HIT	M cold	HIT

Diversa associatività

- 1 via, 4 set per via
- **MISS = 100%**

Blocchi da 1 word, 4 linee, direct mapped					
#blocco	0	8	0	6	8
index	0	0	0	2	0
tag	0	2	0	1	2
H/M	M cold	M cold	M conf.	M cold	M conf.

- 2 vie, 2 set per via
- **MISS = 80%**

Blocchi da 1 word, 4 linee, 2 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
index	0	0	0	0	0
tag	0	4	0	3	4
H/M	M cold	M cold	HIT	M cold	M conf.

- 4 vie, 1 set per via
- **MISS = 60%**

Blocchi da 1 word, 4 linee, 4 vie con rimpiazzo LRU					
#blocco	0	8	0	6	8
tag	0	8	0	6	8
H/M	M cold	M cold	HIT	M cold	HIT

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

4 vie
strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco														
indice linea														
tag														
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit **Index=log2(2)=1 bit** **tag=32–offset-–index = 32–5–1 = 26 bit**

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco														
indice linea														
tag														
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset-–index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco														
indice linea														
tag														
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset–index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	#blocco = ⌊ address / dim. blocco ⌋ (/32)													
indice linea														
tag														
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset-–index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco														
indice linea														
tag														
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset-–index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea														
tag														
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset–-index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea	indice linea = #blocco % #insiemi (%2)													
tag	tag = #blocco / #insiemi (/2)													
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset-–index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea														
tag														
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

4 vie

strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset-–index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea	0	1	0	0	0	0	0	1	0	0	0	0	0	1
tag	2	4	28	2	8	5	28	1	8	31	5	8	28	4
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset–-index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea	0	1	0	0	0	0	0	1	0	0	0	0	0	1
tag	2	4	28	2	8	5	28	1	8	31	5	8	28	4
Hit/Miss	HIT se stesso tag e stesso indice entro 4 (vie)													

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

4 vie

strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset-–index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea	0	1	0	0	0	0	0	1	0	0	0	0	0	1
tag	2	4	28	2	8	5	28	1	8	31	5	8	28	4
Hit/Miss														

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

  4 vie
  strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset-–index = 32–5–1 = 26 bit

DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea	0	1	0	0	0	0	0	1	0	0	0	0	0	1
tag	2	4	28	2	8	5	28	1	8	31	5	8	28	4
Hit/Miss	M	M	M	H	M	M	H	M	H	M	H	H	H	H

Esercizio per casa

- Per una cache con:
  - blocchi da **8** word = 32 byte = 256 bit
  - **2** insiemi per via

4 vie
strategia di rimpiazzo **LRU**

1) Calcolate la **dimensione in bit**

Offset=log2(32)=5 bit Index=log2(2)=1 bit tag=32–offset-–index = 32–5–1 = 26 bit

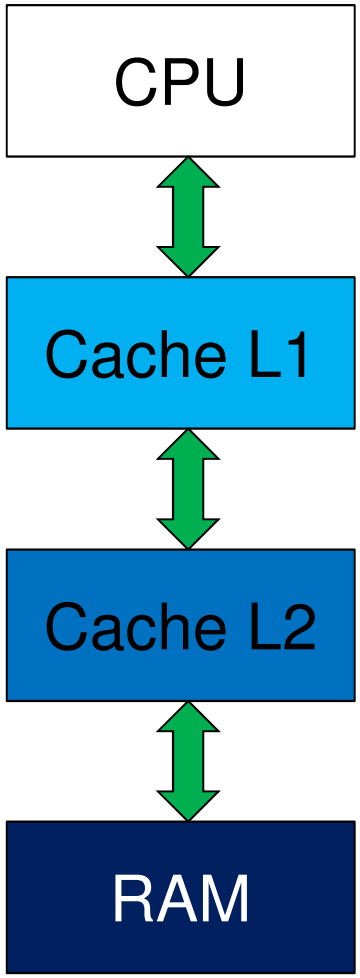
DIM = 4 vie * 2 set * (tag + validità + blocco) = 8 * (26 + 1 + 256) = 8*283=2264 bit

2) Calcolate nella sequenza di accessi qui sotto **quali sono HIT e quali sono MISS**

address	130	310	1800	150	519	342	1820	127	529	2000	325	516	1794	317
#blocco	4	9	56	4	16	10	56	3	16	62	10	16	56	9
indice linea	0	1	0	0	0	0	0	1	0	0	0	0	0	1
tag	2	4	28	2	8	5	28	1	8	31	5	8	28	4
Hit/Miss	M	M	M	H	M	M	H	M	H	M	H	H	H	H

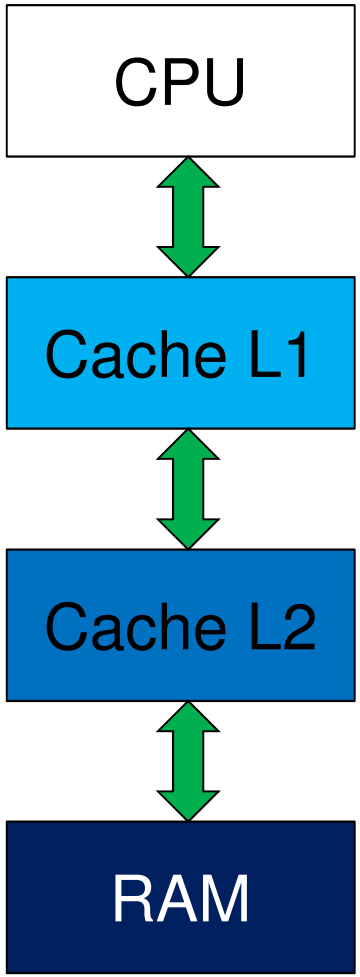
Cache multi-livello

- In una memoria multi-livello:
- ogni **HIT** fornisce i dati a tutti i livelli superiori
- solo le **MISS** fanno richieste al livello subito inferiore
- ogni accesso impiega il tempo del livello più basso che ha risposto



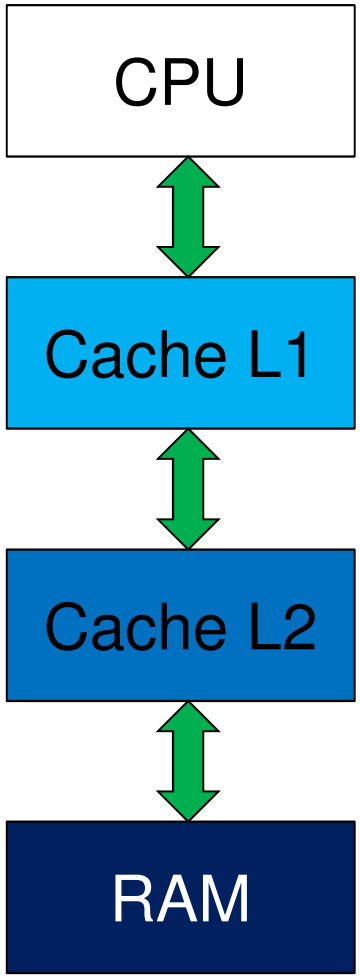
Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**



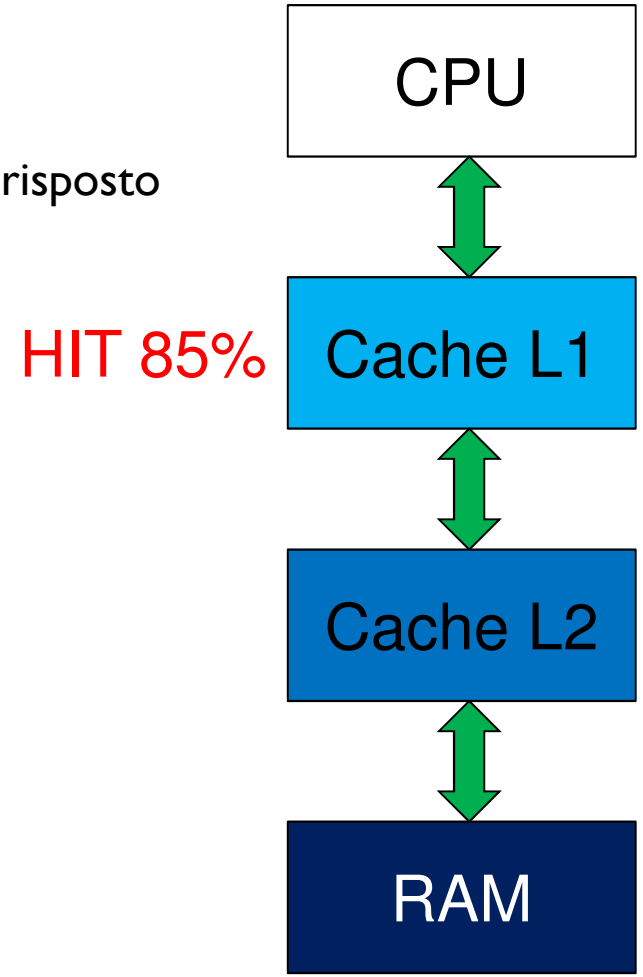
Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%



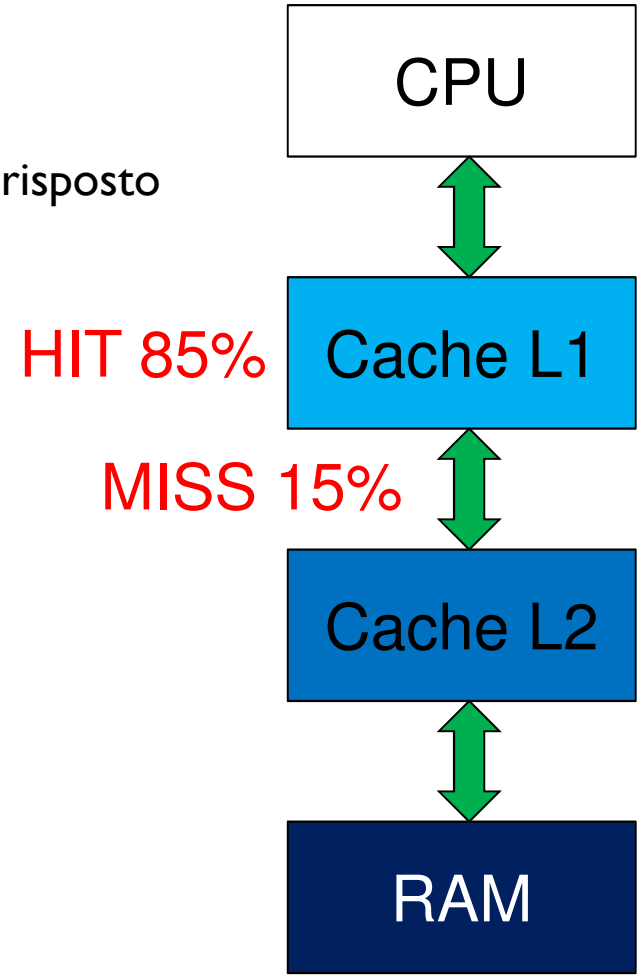
Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%



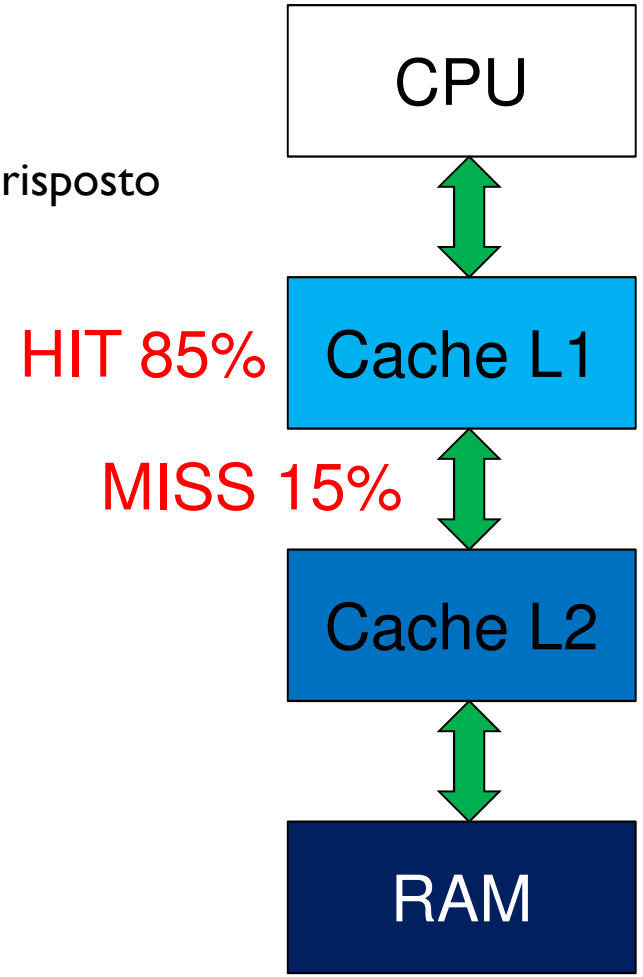
Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%



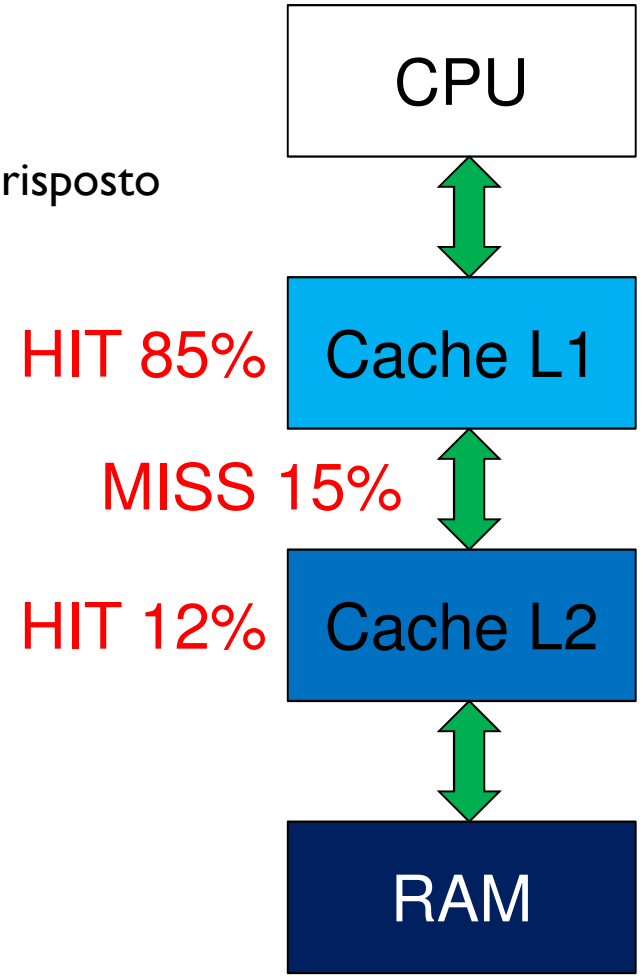
Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%
  perc. di MISS su L2 = 20% → HIT su L2 = 80%



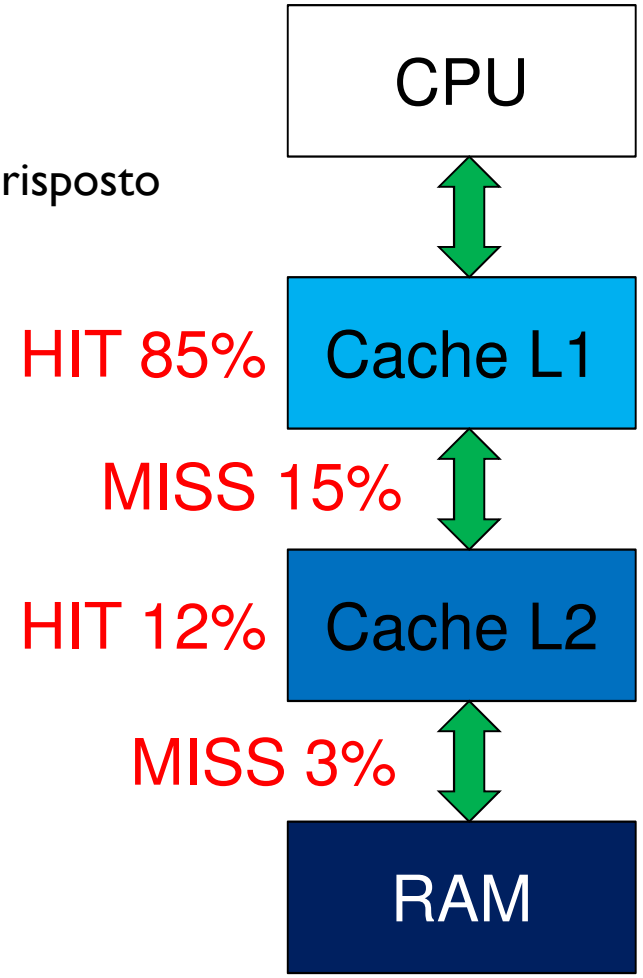
Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%
  perc. di MISS su L2 = 20% → HIT su L2 = 80%



Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%
  perc. di MISS su L2 = 20% → HIT su L2 = 80%

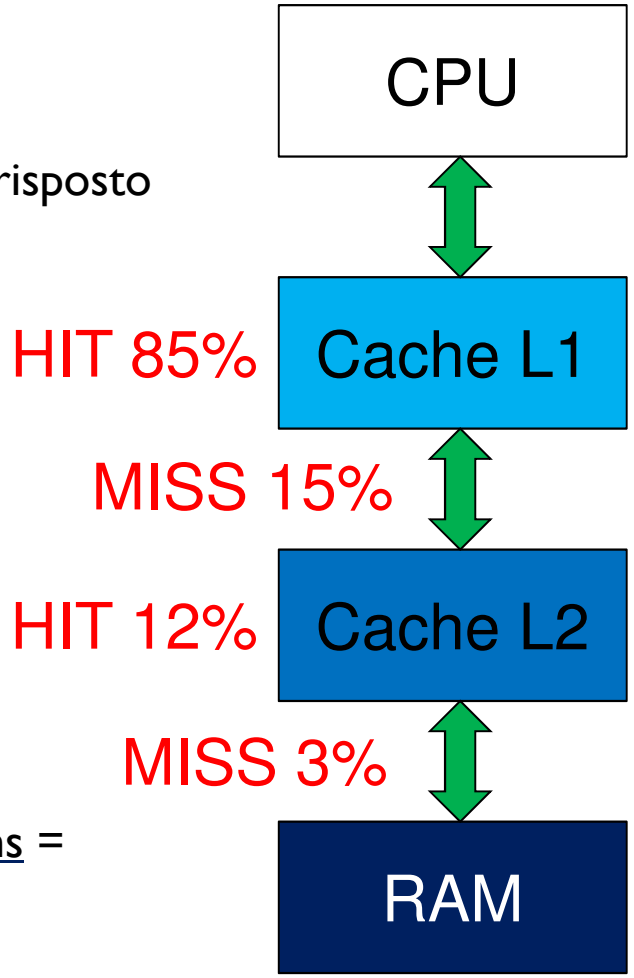


Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%
  perc. di MISS su L2 = 20% → HIT su L2 = 80%

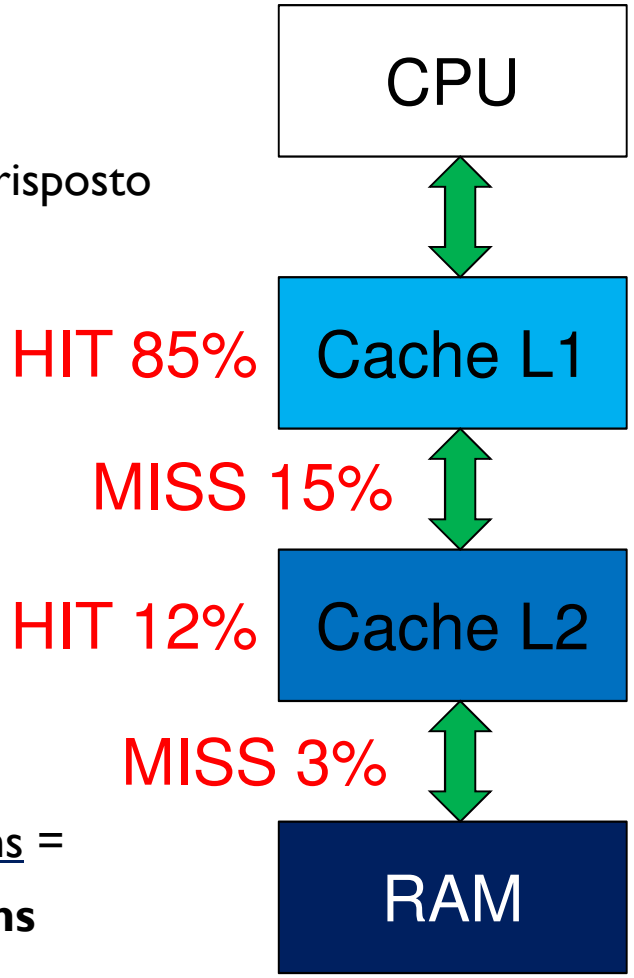
Il tempo medio di un accesso in memoria sarà

85% * 2ns + 15% * 80% * 30ns + 15% * 20% * 100ns =



Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%
  perc. di MISS su L2 = 20% → HIT su L2 = 80%



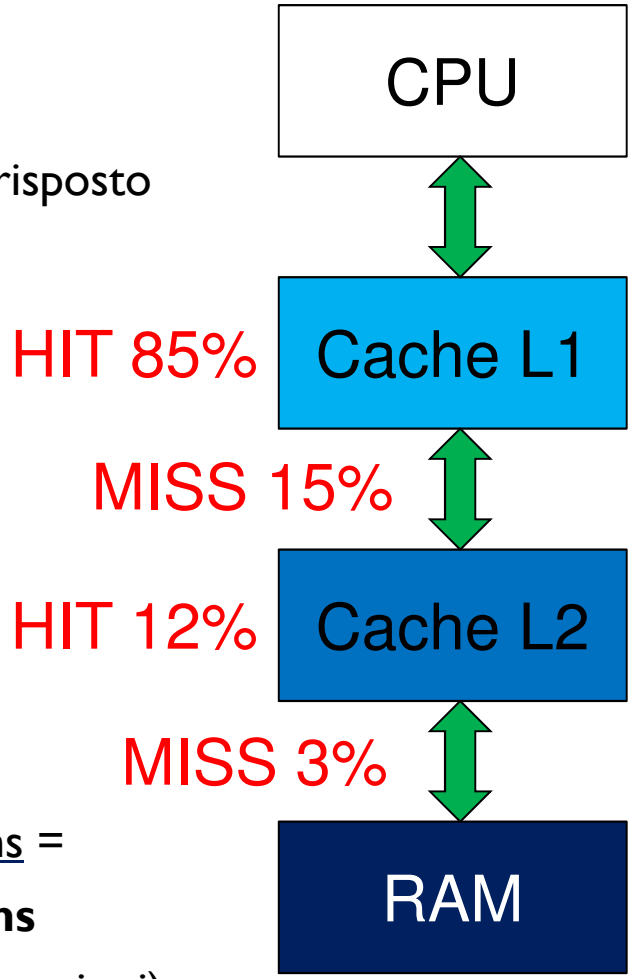
Il tempo medio di un accesso in memoria sarà

85% * 2ns + 15% * 80% * 30ns + 15% * 20% * 100ns =

= 1.7ns + 3.6 ns + 3 ns = **8.3 ns**

Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%
  perc. di MISS su L2 = 20% → HIT su L2 = 80%



Il tempo medio di un accesso in memoria sarà

85% * 2ns + 15% * 80% * 30ns + 15% * 20% * 100ns =

= 1.7ns + 3.6 ns + 3 ns = **8.3 ns**

Ovvero 8.3 * 2 = **16.6** clock per accesso (1 ogni 5.5 istruzioni)

Cache multi-livello

- In una memoria multi-livello:
  - ogni **HIT** fornisce i dati a tutti i livelli superiori
  - solo le **MISS** fanno richieste al livello subito inferiore
  - ogni accesso impiega il tempo del livello più basso che ha risposto
- **Esempio** di calcolo dei tempi se supponiamo che:
  - HIT su L1 = **2ns**
  - HIT su L2 = **30ns**
  - accesso a RAM (ovvero MISS su L2) = **100ns**
  - clock della CPU da **2 Ghz** e **3 CPI**
- **Dati:** perc. di MISS su L1 = 15% → HIT su L1 = 85%
  perc. di MISS su L2 = 20% → HIT su L2 = 80%

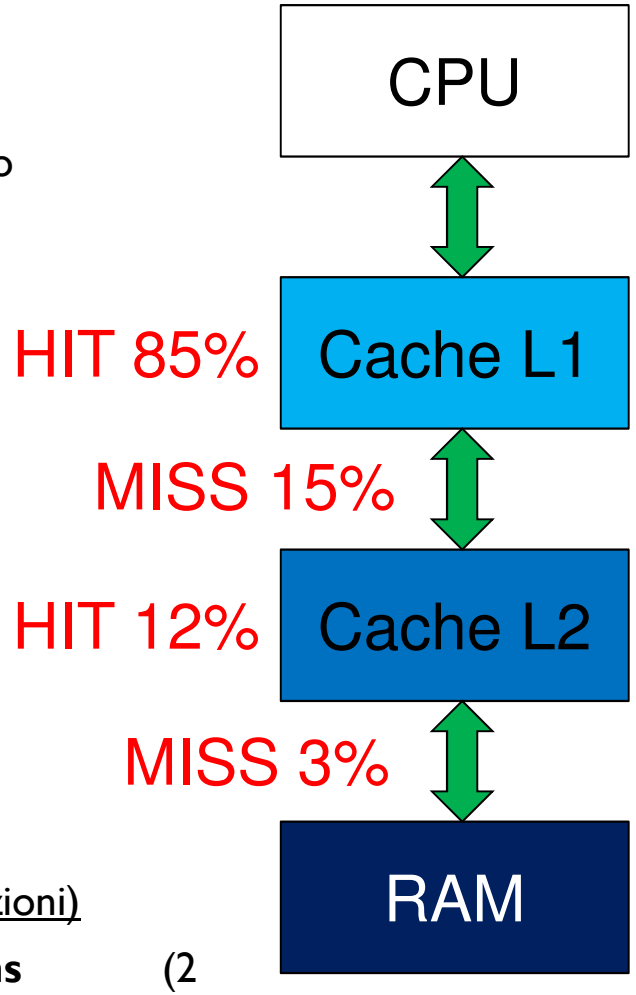
Il tempo medio di un accesso in memoria sarà

85% * 2ns + 15% * 80% * 30ns + 15% * 20% * 100ns =

= 1.7ns + 3.6 ns + 3 ns = **8.3 ns**

Ovvero 8.3 * 2 = **16.6** clock per accesso (1 ogni 5.5 istruzioni)

Con la sola L1: 85% * 2ns + 15% * 100 ns = 1.7ns + 15ns = **16.7 ns** (2 volte più lento)



Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#														
	index														
	tag														
	H/M														
L2	block#														
	index														
	tag														
	H/M														

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#														
	index														
	tag														
	H/M														
L2	block#														
	index														
	tag														
	H/M														

#blocco = ⌊ address / dim. blocco ⌋ (/8)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index														
	tag														
	H/M														
L2	block#														
	index														
	tag														
	H/M														

#blocco = ⌊ address / dim. blocco ⌋ (/8)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index														
	tag														
	H/M														
L2	block#														
	index														
	tag														
	H/M														

tag = #blocco / #insiemi (/4)

indice linea = #blocco % #insiemi (%4)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M														
L2	block#														
	index														
	tag														
	H/M														

tag = #blocco / #insiemi (/4)

indice linea = #blocco % #insiemi (%4)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M														
L2	block#														
	index														
	tag														
	H/M														

HIT se stesso tag e stesso indice entro 1 (via)

MISS di cold start: 1° accesso
MISS di conflitto: HIT se fully-associative
MISS di capacità: MISS anche se fully-associative

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#														
	index														
	tag														
	H/M														

HIT se stesso tag e stesso indice entro 1 (via)

MISS di cold start: 1° accesso
MISS di conflitto: HIT se fully-associative
MISS di capacità: MISS anche se fully-associative

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#														
	index														
	tag														
	H/M														

#blocco = ⌊ address / dim. blocco ⌋ (/64)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index														
	tag														
	H/M														

#blocco = ⌊ address / dim. blocco ⌋ (/64)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index														
	tag														
	H/M														

tag = #blocco / #insiemi (/8)

indice linea = #blocco % #insiemi (%8)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M														

tag = #blocco / #insiemi (/8)

indice linea = #blocco % #insiemi (%8)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M														

HIT se stesso tag e stesso indice entro 2 (vie)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT	HIT	HIT	HIT

HIT se stesso tag e stesso indice entro 2 (vie)

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT	HIT	HIT	HIT

- **Tempo impiegato:** **2 L1 HIT + 7 L2 HIT + 5 L2 MISS**

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT	HIT	HIT	HIT

- **Tempo impiegato:** **2 L1 HIT + 7 L2 HIT + 5 L2 MISS**
- **Se L1 HIT = 1ns L2 HIT = 10ns L2 MISS = 100ns e clock = 1Ghz con 1 CPI**

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT	HIT	HIT	HIT

- **Tempo impiegato:** **2 L1 HIT + 7 L2 HIT + 5 L2 MISS**
- **Se L1 HIT = 1ns L2 HIT = 10ns L2 MISS = 100ns e clock = 1Ghz con 1 CPI**
- **Tempo totale = 2 * 1ns + 7 * 10ns + 5 * 100ns = 572 ns**

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT	HIT	HIT	HIT

- **Tempo impiegato:** **2 L1 HIT + 7 L2 HIT + 5 L2 MISS**
- **Se L1 HIT = 1ns L2 HIT = 10ns L2 MISS = 100ns e clock = 1Ghz con 1 CPI**
- **Tempo totale = 2 * 1ns + 7 * 10ns + 5 * 100ns = 572 ns**
- **Tempo medio = 572ns /14 = 40.8ns ovvero 40.8 colpi di clock medi per accesso**

Esempio

- Cache **L1** con blocchi da **2 word**, **direct mapped** con **4 set** (linee)
- Cache **L2** con blocchi da **16 word**, **2 ways** con **8 set** per way e **politica di rimpiazzo LRU**

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	140	66	137	257	64	11	257	39	12	39	137	11	39	13
	index	0	2	1	1	0	3	1	3	0	3	1	3	3	1
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	M conf	M conf	M cold
L2	block#	17	8	17	32	8	1		4	1		17	1	4	1
	index	1	0	1	0	0	1		4	1		1	1	4	1
	tag	2	1	2	4	1	0		0	0		2	0	0	0
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT	HIT	HIT	HIT

- **Tempo impiegato:** **2 L1 HIT + 7 L2 HIT + 5 L2 MISS**
- **Se L1 HIT = 1ns L2 HIT = 10ns L2 MISS = 100ns e clock = 1Ghz con 1 CPI**
- **Tempo totale = 2 * 1ns + 7 * 10ns + 5 * 100ns = 572 ns**
- **Tempo medio = 572ns /14 = 40.8ns ovvero 40.8 colpi di clock medi per accesso**
- **NOTA: si potrebbero eliminare le 3 MISS di conflitto cambiando l'associatività di L1**

Proviamo a cambiare qualcosa

- Cache L1:

  blocco da 4 word (16 byte) — per diminuire i MISS cold start

  2 ways — per diminuire i MISS conflict

  2 set per via — (stesso numero totale di blocchi)

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	70	33	68	128	32	5	128	19	6	19	68	5	19	6
	index	0	1	0	0	0	1	0	1	0	1	0	1	1	0
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	HIT	HIT	HIT
L2	block#	17	8	17	32	8	1		4	1		17			
	index	1	0	1	0	0	1		4	1		1			
	tag	2	1	2	4	1	0		0	0		2			
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT			

Proviamo a cambiare qualcosa

- Cache L1:
  - blocco da 4 word (16 byte) — per diminuire i MISS cold start
  - 2 ways — per diminuire i MISS conflict
  - 2 set per via — (stesso numero totale di blocchi)

cold

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	70	33	68	128	32	5	128	19	6	19	68	5	19	6
	index	0	1	0	0	0	1	0	1	0	1	0	1	1	0
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	HIT	HIT	HIT
L2	block#	17	8	17	32	8	1		4	1		17			
	index	1	0	1	0	0	1		4	1		1			
	tag	2	1	2	4	1	0		0	0		2			
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT			

Proviamo a cambiare qualcosa

- Cache L1:
  - blocco da 4 word (16 byte) — per diminuire i MISS cold start
  - 2 ways — per diminuire i MISS conflict
  - 2 set per via — (stesso numero totale di blocchi)

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	70	33	68	128	32	5	128	19	6	19	68	5	19	6
	index	0	1	0	0	0	1	0	1	0	1	0	1	1	0
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	HIT	HIT	HIT
L2	block#	17	8	17	32	8	1		4	1		17			
	index	1	0	1	0	0	1		4	1		1			
	tag	2	1	2	4	1	0		0	0		2			
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT			

cold

conf.

Proviamo a cambiare qualcosa

- Cache L1:
  - blocco da 4 word (16 byte) — per diminuire i MISS cold start
  - 2 ways — per diminuire i MISS conflict
  - 2 set per via — (stesso numero totale di blocchi)

cold

conf.

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	70	33	68	128	32	5	128	19	6	19	68	5	19	6
	index	0	1	0	0	0	1	0	1	0	1	0	1	1	0
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	HIT	HIT	HIT
L2	block#	17	8	17	32	8	1		4	1		17			
	index	1	0	1	0	0	1		4	1		1			
	tag	2	1	2	4	1	0		0	0		2			
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT			

Totale: 5 L1 HIT + 4 L2 HIT + 5 L2 MISS

Proviamo a cambiare qualcosa

- Cache L1:
  - blocco da 4 word (16 byte) — per diminuire i MISS cold start
  - 2 ways — per diminuire i MISS conflict
  - 2 set per via — (stesso numero totale di blocchi)

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	70	33	68	128	32	5	128	19	6	19	68	5	19	6
	index	0	1	0	0	0	1	0	1	0	1	0	1	1	0
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	HIT	HIT	HIT
L2	block#	17	8	17	32	8	1		4	1		17			
	index	1	0	1	0	0	1		4	1		1			
	tag	2	1	2	4	1	0		0	0		2			
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT			

cold

conf.

Totale: 5 L1 HIT + 4 L2 HIT + 5 L2 MISS

Tempo: 5 * 1ns + 4 * 10ns + 5*100ns = 545ns (inv. di 572ns, **quasi 5% in meno**)

Proviamo a cambiare qualcosa

- Cache L1:
  - blocco da 4 word (16 byte) — per diminuire i MISS cold start
  - 2 ways — per diminuire i MISS conflict
  - 2 set per via — (stesso numero totale di blocchi)

cold

conf.

	Address	1120	531	1100	2056	512	95	2060	317	101	315	1099	89	319	104
L1	block#	70	33	68	128	32	5	128	19	6	19	68	5	19	6
	index	0	1	0	0	0	1	0	1	0	1	0	1	1	0
	tag	35	16	34	64	16	2	64	9	3	9	34	2	9	3
	H/M	M cold	M cold	M cold	M cold	M cold	M cold	HIT	M cold	M cold	HIT	M conf	HIT	HIT	HIT
L2	block#	17	8	17	32	8	1		4	1		17			
	index	1	0	1	0	0	1		4	1		1			
	tag	2	1	2	4	1	0		0	0		2			
	H/M	M cold	M cold	HIT	M cold	HIT	M cold		M cold	HIT		HIT			

Totale: 5 L1 HIT + 4 L2 HIT + 5 L2 MISS

Tempo: 5 * 1ns + 4 * 10ns + 5*100ns = 545ns (inv. di 572ns, **quasi 5% in meno**)

Esercizio: Cambiate L1 ed L2 mantenendo la stessa dimensione totale