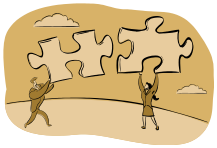
Esercizi Cache

organizzazione e tecniche di allocazione

Es1: Si consideri una cache di 4KB con associazione a gruppi a 8 vie (8-way set associative) in congiunzione con una memoria centrale di 1MB.

Supponendo che un blocco sia di dimensione 64B, si dica come un indirizzo di memoria è suddiviso in campi e a quanto ammonta la dimensione di ogni campo.



Soluz.:

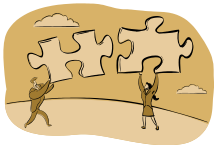
- trattandosi di una cache con associazione a gruppi, l'indirizzo di memoria centrale deve essere suddiviso nei campi tag, set, e parola.
- la memoria centrale è di 1MB, cioè 2^{20} byte; pertanto un indirizzo di memoria centrale è espresso in 20 bit.
- la dimensione del campo parola è individuato univocamente dalla dimensione del blocco, che è di 64B, cioè 2^6 byte; pertanto il campo parola è di 6 bit.
- una cache di 4KB possiede 2^{12} byte; ogni linea deve contenere un blocco e quindi impegna 2^6 byte; quindi la cache contiene $2^{12}/2^6 = 2^6$ linee. Poiché un insieme deve contenere 8 linee, il numero di insiemi della cache è pari a $2^6/2^3 = 2^3$. Pertanto il campo set è di 3 bit.
- la dimensione del campo tag sarà dunque: 20 -3 -6 = 11 bit

Esercizi Cache

organizzazione e tecniche di allocazione

Es2: Si consideri una cache di 16KB con associazione a gruppi a 4 vie (4-way set associative) e dimensione di linea di 32B.

Supponendo che il campo tag sia di 12 bit, si dica quale è la dimensione massima (in byte) di memoria principale che la cache è in grado di gestire, assumendo il singolo byte come unità di indirizzamento della memoria.



Soluz.:

- per calcolare la quantità massima di memoria principale gestibile, bisogna calcolare il numero di bit totali che esprimono una generica locazione di memoria.
- trattandosi di una cache con associazione a gruppi, l'indirizzo di memoria centrale deve essere suddiviso nei campi tag, set, e parola.
- sappiamo che il campo tag è di 12 bit; quindi occorre calcolare la dimensione dei campi set e parola.
- la dimensione del campo parola è individuato univocamente dalla dimensione del blocco, che è di 32B, cioè 2^5 byte; pertanto il campo parola è di 5 bit.
- una cache di 16KB possiede 2^{14} byte; ogni linea deve contenere un blocco e quindi impegna 2^5 byte; quindi la cache contiene $2^{14}/2^5 = 2^9$ linee. Poiché un insieme deve contenere 4 linee, il numero di insiemi della cache è pari a $2^9/2^2 = 2^7$. Pertanto il campo set è di 7 bit.
- quindi la dimensione massima di memoria gestibile è: 2^{12+7+5}, cioè 16MB

Esercizi Cache

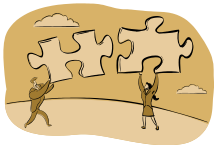
organizzazione e tecniche di allocazione

Es3: Si consideri una cache di 4KB con associazione a gruppi a 4 vie (4-way set associative) in congiunzione con una memoria centrale di 256KB.

Supponendo che un blocco sia di dimensione 64B, si dica:

a) se le locazioni di memoria con indirizzi (in esadecimale) **30E5C** e **17A87** hanno la possibilità di essere caricate all'interno dello stesso set di linee;

b) se in cache è presente la locazione con indirizzo **05ABC**, quali altre locazioni sono sicuramente presenti nella cache.



Soluz.:

a) procedendo come visto negli esercizi precedenti, abbiamo che un indirizzo di memoria è decomponibile in un campo parola di 6 bit, un campo set di 4 bit, ed un campo tag di 8 bit.

• le due locazioni di indirizzo **30E5C** e **17A87** possono trovarsi nello stesso insieme se il loro campo set è identico. Quindi basta controllare se i bit da 10 a 7 (a partire da destra) sono identici:

30E5C = (su 18 bit) 110000111001011100

17A87 = (su 18 bit) 010111101010000111

• non essendo identici, la risposta è no.

b) le altre locazioni che necessariamente saranno presenti con la locazione di indirizzo **05ABC** sono quelle all'interno del medesimo blocco.

• poiché **05ABC** = (su 18 bit) 000101101010111100 , tutte le locazioni con indirizzo da 000101101010000000 (hex **05A80**) a 000101101010111111 (hex **05ABF**) si troveranno simultaneamente in cache.

Es4: Sia data la seguente sequenza di indirizzi in lettura (l) o scrittura (s) emessi dalla CPU:

	Indirizzo	l/s	dato scritto (in esadecimale)
1	000100000000	l	
2	000100001000	l	
3	000100001100	s	B1
4	000100001100	l	
5	000100010000	s	B4
6	000100010000	l	
7	000100010100	s	B7

Si assuma che la dimensione di parola coincida con un byte, e la presenza di una cache di ampiezza 16B, dimensione di blocco 4B, inizialmente vuota, e ad associazione a 2 vie (con politica di rimpiazzo LRU e politica di scrittura write-through).
Si assuma che la memoria abbia il contenuto esadecimale mostrato di seguito:

ind	byte	ind	byte	ind	byte	ind	byte
100	0C	101	00	102	07	103	02
104	00	105	00	106	00	107	00
108	AE	109	13	10A	A1	10B	23
10C	A1	10D	42	10E	90	10F	75
110	B9	111	16	112	00	113	00
114	0A	115	07	116	03	117	71

ind = indirizzo

Si mostri come sia il contenuto della cache che il contenuto della memoria cambia.

Soluzione:

Poiché un blocco è costituito da 4B, e la cache è di 16B, si avranno in cache 16/4 = 4 linee.

Essendo l'associatività a due linee (2 vie), la cache sarà costituita da due insiemi (set 0 e set 1) ognuno di 2 linee.

Quindi i 12 bit di indirizzo saranno suddivisi nel seguente modo:

- i 2 bit meno significativi individueranno il byte all'interno del blocco;
- il terzo bit da destra individuerà l'insieme (set 0 o set 1);
- i restanti bit costituiranno il campo tag.

Mostriamo di seguito l'evoluzione del contenuto della cache e della memoria.

Per la cache, nel caso in cui tutte e due le linee di un insieme (set) siano libere, si sceglie la linea con indirizzo minore per la allocazione (scelta arbitraria: si poteva usare un criterio diverso).

In caso di miss per una operazione di scrittura, si assume la politica "write allocate", cioè si porta prima in cache il blocco che contiene la parola da scrivere e poi si effettua la scrittura.

Codifica della soluzione

```
ind. rif.              cache dati                  modifica memoria
memoria          set  0          set  1            mem[ind.] = cont.
   hex        [  linea  0 ]   [  linea  2 ]
 binario      t: tag          t: tag
              r: rif.         r: rif.
              [  linea  1 ]   [  linea  3 ]
              t: tag          t: tag
              r: rif.         r: rif.
```

```
    100       [0C|00|07|02]
000100000000  t:000100000
              r:miss
              [           ]
              t:
              r:

    108       [0C|00|07|02]
000100001000  t:000100000
              r:
              [AE|13|A1|23]
              t:000100001
              r:miss
```

write allocate

prima del write

modifica memoria

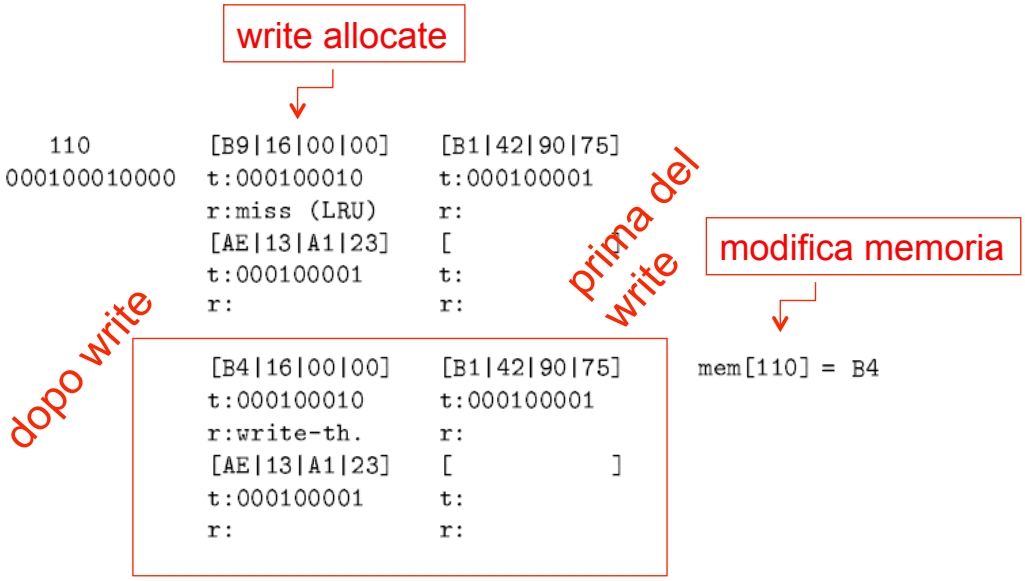
```
    10C       [0C|00|07|02]   [A1|42|90|75]
000100001100  t:000100000     t:000100001
              r:              r:miss
              [AE|13|A1|23]   [           ]
              t:000100001     t:
              r:              r:
```

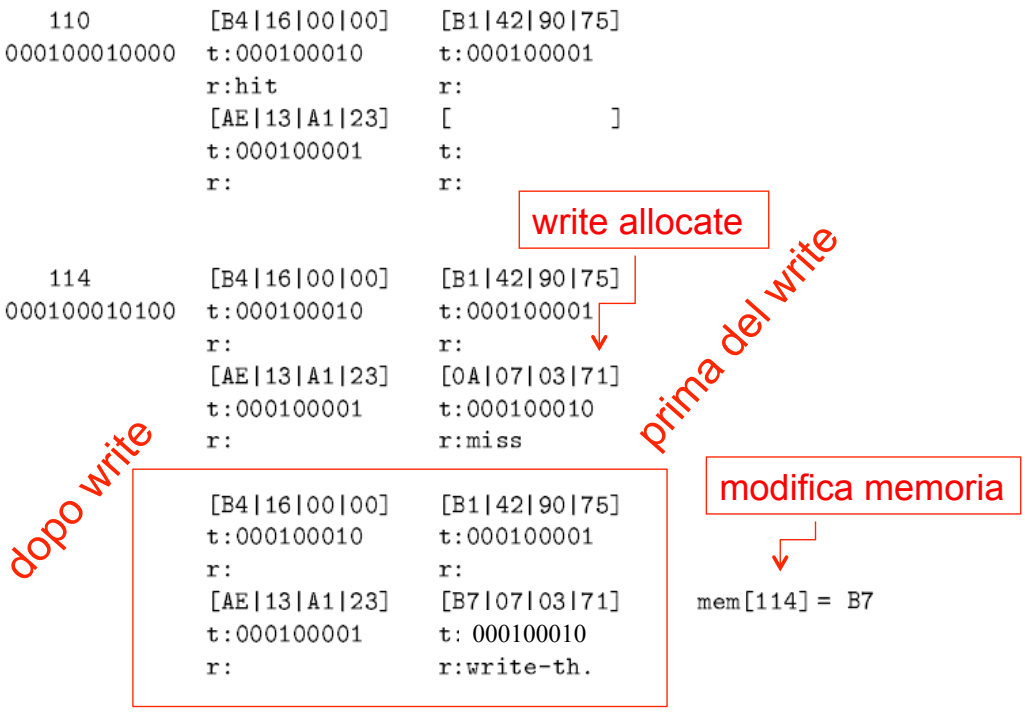
dopo write

```
              [0C|00|07|02]   [B1|42|90|75]
              t:000100000     t:000100001
              r:              r:write-th.
              [AE|13|A1|23]   [           ]
              t:000100001     t:
              r:              r:
```

```
mem[10C] = B1
```

```
    10C       [0C|00|07|02]   [B1|42|90|75]
000100001100  t:000100000     t:000100001
              r:              r:hit
              [AE|13|A1|23]   [           ]
              t:000100001     t:
              r:              r:
```


write allocate
110
000100010000
[B9|16|00|00]
t:000100010
r:miss (LRU)
[AE|13|A1|23]
t:000100001
r:
[B1|42|90|75]
t:000100001
r:
[
t:
r:
prima del write
modifica memoria
dopo write
[B4|16|00|00]
t:000100010
r:write-th.
[AE|13|A1|23]
t:000100001
r:
[B1|42|90|75]
t:000100001
r:
[ ]
t:
r:
mem[110] = B4


110
000100010000
[B4|16|00|00]
t:000100010
r:hit
[AE|13|A1|23]
t:000100001
r:
[B1|42|90|75]
t:000100001
r:
[ ]
t:
r:
write allocate
114
000100010100
[B4|16|00|00]
t:000100010
r:
[AE|13|A1|23]
t:000100001
r:
[B1|42|90|75]
t:000100001
r:
[0A|07|03|71]
t:000100010
r:miss
prima del write
dopo write
[B4|16|00|00]
t:000100010
r:
[AE|13|A1|23]
t:000100001
r:
[B1|42|90|75]
t:000100001
r:
[B7|07|03|71]
t: 000100010
r:write-th.
modifica memoria
mem[114] = B7