Esercitazione su cache

Prestazioni e memoria: Esercizio 1

Si consideri l'esecuzione di un programma *P* su di una data CPU. Il *CPI* ideale è pari a *3*, ma considerando i *miss* della cache si ottiene un *CPI* reale pari a *3.6*. Sapendo inoltre che *Miss penalty* = 12 cicli e che *Instruction miss rate = 4%* determinare *Data miss rate* per il programma considerato, tenendo conto che la percentuale di load/store è del 40%.

Prestazioni e memoria: Esercizio 1

Si consideri l'esecuzione di un programma *P* su di una data CPU. Il *CPI* ideale è pari a *3*, ma considerando i *miss* della cache si ottiene un *CPI* reale pari a *3.6*. Sapendo inoltre che *Miss penalty* = 12 cicli e che *Instruction miss rate = 4%* determinare *Data miss rate* per il programma considerato, tenendo conto che la percentuale di load/store è del 40%.

$$\begin{aligned} \textit{cicli tot.} &= CPI_{reale} \bullet IC \\ &= CPI_{ideale} \bullet IC + \\ &\quad (IC \bullet \textit{perc. load/store} \bullet \textit{Data miss rate} \bullet \textit{Miss penalty}) + \\ &\quad (IC \bullet \textit{Instruction miss rate} \bullet \textit{Miss penalty}) \end{aligned}$$

$$\begin{aligned} CPI_{reale} &= \textit{cicli tot.} / IC = \\ &= CPI_{ideale} + \textit{perc. load/store} \bullet \textit{Data miss rate} \bullet \textit{Miss penalty} + \\ &\quad + \textit{Instruction miss rate} \bullet \textit{Miss penalty} = \\ &= 3 + 0.4 \bullet \textit{Data miss rate} \bullet 12 + 0.04 \bullet 12 = \\ &= 3 + 4.8 \bullet \textit{Data miss rate} + 0.48 \end{aligned}$$

Data miss rate = 3.6 - 3.48 / 4.8 = 0.025 cioè *2.5%*

Gerarchie di memoria: Cache

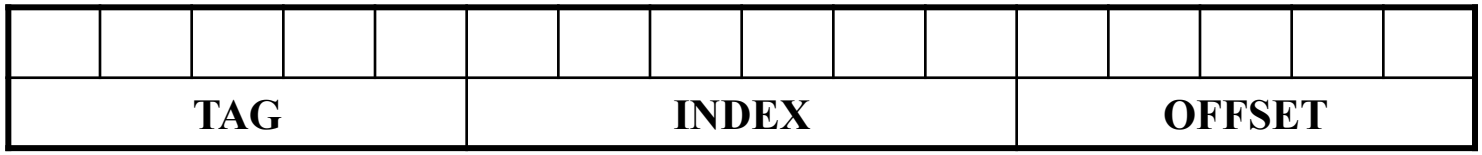
Se l'indirizzo fisico è di *m bit* e la dimensione del blocco è 2^n, allora gli *m - n bit* più significativi rappresentano l'indirizzo del blocco, i rimanenti l'offset all'interno del blocco.

- Cache diretta

Ogni blocco di memoria può essere inserito in un'unica entry della cache.

- Cache k-way associative

I blocchi della cache sono suddivisi in insiemi di dimensione *k*. Ogni blocco di memoria può essere inserito in uno degli *k* blocchi di un insieme (riduce la possibilità di conflitti).



Gerarchie di memoria: Cache

Consideriamo ora i problemi relativi al dimensionamento, organizzazione, indirizzamento della memoria cache …

Se l'indirizzo (memoria principale) è di *m bit* e la dimensione del blocco è 2^n, allora gli *m - n bit* più significativi rappresentano l'indirizzo del blocco, i rimanenti l'offset all'interno del blocco.

Cache diretta

Ogni blocco di memoria può essere inserito in un unico blocco della cache

Cache block index = Indirizzo del blocco **mod** *#(cache blocks)*

Se *#(cache blocks)* = 2^k allora il *Cache block index è* dato dai *k bit* meno significativi del *Mem. block address.*

Gerarchie di memoria: Cache

Cache n-way associative

I blocchi della cache sono suddivisi in *set* (insiemi) di dimensione *n*. Ogni blocco di memoria può essere inserito in uno degli *n* blocchi di un set (riduce la possibilità di conflitti).

Il numero dei set di blocchi in cache è

$$\#(\text{cache sets}) = \#(\text{cache blocks}) / n$$

Mentre, per un dato blocco di memoria, l'indice del set nella cache sarà:

$$\text{Cache set index} = \text{Mem. block address} \ \textbf{mod} \ \#(\text{cache sets})$$

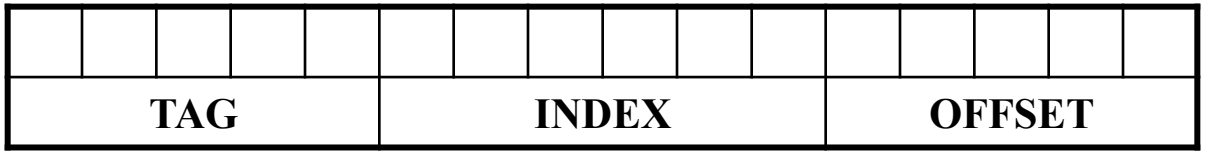
Se $\#(\text{cache sets}) = 2^k$ allora il *Cache set index è* dato dai *k bit* meno significativi del *Mem. block address.*

Gerarchie di memoria: Cache

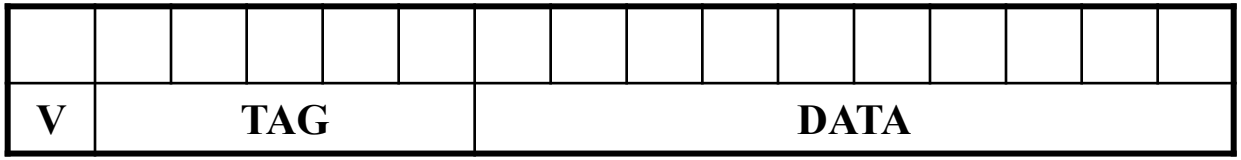
Come si fa a sapere se il contenuto della cache è il dato cercato?

Le entry della cache contengono le informazioni necessarie:

Indirizzo



Cache entry
(pos. INDEX)



Si reperisce la cache entry di indirizzo INDEX. Se questa è valida e il suo TAG coincide con quello dell'indirizzo → hit, il dato è corretto. Altrimenti si è verificato un miss.

Per cache *associative* occorre confrontare il TAG dell'indirizzo con quello di *ogni* entry del *set* indirizzato da INDEX.

Cache: Esercizio 2

Considerare una cache *4-way associative*, con parte dati di *8 KB* organizzata in blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

a. Determinare la suddivisione dell'indirizzo fisico nei campi TAG, INDEX, OFFSET

Cache: Esercizio 2

Considerare una cache *4-way associative*, con parte dati di *8 KB* organizzata in blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

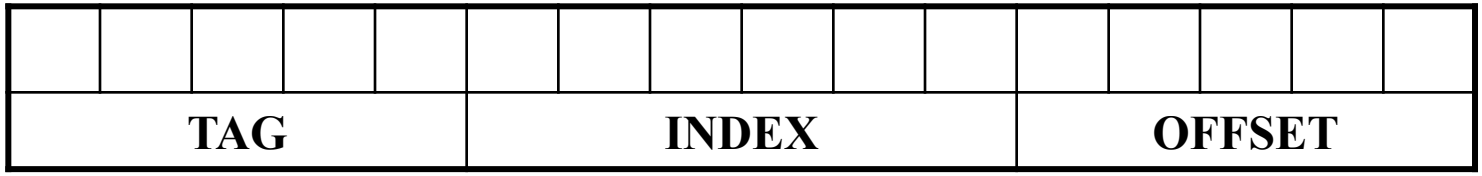
a. Determinare la suddivisione dell'indirizzo fisico nei campi TAG, INDEX, OFFSET

1 Blocco = 32 B $\rightarrow$ *OFFSET* = $log_2 32 = 5$ *bit*

Tot. blocchi = Cache size / Block size = 8KB / 32B = $2^{13} / 2^5 = 2^8$ *blocchi*

Poiché la cache è 4-way associative si hanno $2^8/2^2 = 2^6$ set

Quindi *INDEX = 6 bit* e i rimanenti *5 bit* sono per *TAG*



Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* e avente blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

b. Stabilire se gli indirizzi 0xAFAF e 0xAFB0 sono mappati nello stesso insieme della cache.

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* e avente blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

b. Stabilire se gli indirizzi 0xAFAF e 0xAFB0 sono mappati nello stesso insieme della cache.

			TAG	INDEX	OFFSET
0xAFAF	= 1010 1111 1010 1111	=	10101	111101	01111
0xAFB0	= 1010 1111 1011 0000	=	10101	111101	10000

Quindi i contenuti dei due indirizzi compaiono nello stesso set

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* e avente blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

b. Stabilire se gli indirizzi 0xAFAF e 0xAFB0 sono mappati nello stesso insieme della cache.

			TAG	INDEX	OFFSET
0xAFAF	= 1010 1111 1010 1111	=	10101	111101	01111
0xAFB0	= 1010 1111 1011 0000	=	10101	111101	10000

Quindi i contenuti dei due indirizzi compaiono nello stesso set

c. Supponendo che il contenuto di 0xAFAF sia nella cache, cosa accade se tentiamo di leggere il contenuto di 0xAFB0 ?

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* e avente blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

b. Stabilire se gli indirizzi 0xAFAF e 0xAFB0 sono mappati nello stesso insieme della cache.

			TAG	INDEX	OFFSET
0xAFAF	= 1010 1111 1010 1111	=	10101	111101	01111
0xAFB0	= 1010 1111 1011 0000	=	10101	111101	10000

Quindi i contenuti dei due indirizzi compaiono nello stesso set

c. Supponendo che il contenuto di 0xAFAF sia nella cache, cosa accade se tentiamo di leggere il contenuto di 0xAFB0 ?

I due indirizzi hanno identici INDEX e TAG, quindi sono nello stesso blocco !! Pertanto se il contenuto di 0xAFAF si trova nella cache, vi sarà anche quello di 0xAFB0.

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* e avente blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

d. Cosa succede invece per gli indirizzi 0xAFAF e 0xA7B0 ?

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* e avente blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

d. Cosa succede invece per gli indirizzi 0xAFAF e 0xA7B0 ?

					TAG	INDEX	OFFSET
0xAFAF =	1010	1111	1010	1111	=10101	111101	01111
0xA7B0 =	1010	0111	1011	0000	=10100	111101	10000

Anche in questo caso, i contenuti dei due indirizzi si trovano nello stesso set.

Ma ora i due indirizzi hanno TAG diversi!!

Quindi, se il contenuto di 0xAFAF è nella cache, non è detto che ci sia anche quello di 0xAFB0 [anzi nel caso di cache a corrispondenza diretta (direct mapped) uno escluderebbe l'altro].

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* con blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

						TAG	INDEX	OFFSET
0xAFAF	=	1010	1111	1010	1111	= 10101	111101	01111
0xA7B0	=	1010	0111	1011	0000	= 10100	111101	10000

e ... se la cache fosse *2-way associative*?

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* con blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

						TAG	INDEX	OFFSET
0xAFAF	=	1010	1111	1010	1111	= 10101	111101	01111
0xA7B0	=	1010	0111	1011	0000	= 10100	111101	10000

e ... se la cache fosse *2-way associative*?

Tot. blocchi = 2^8

Dato che la cache è 2-way associative si hanno $2^8 / 2 = 2^7$ set

quindi INDEX = 7 bit ... e i due indirizzi considerati sono su set diversi!

						TAG	INDEX	OFFSET
0xAFAF	=	1010	1111	1010	1111	= 1010	1111101	01111
0xA7B0	=	1010	0111	1011	0000	= 1010	0111101	10000

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* e avente blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

f. Come sono ripartiti i campi TAG, INDEX e OFFSET se invece la cache fosse *completamente associativa*?

Cache: Esercizio 2 (continua)

Considerare una cache *4-way associative* in cui la parte dati è di *8 KB* e avente blocchi da *32 B*. L'indirizzo fisico è di *16 bit*.

f. Come sono ripartiti i campi TAG, INDEX e OFFSET se invece la cache fosse *completamente associativa*?

1 Blocco = 32 B $\rightarrow$ *OFFSET* $= \log_2 32 = 5$ *bit*

Tot. blocchi = Cache size / Block size = 8KB / 32B $= 2^{13}/2^5 = 2^8$ *blocchi*

Poichè la cache è completamente associativa vi è un unico set con 2^8 blocchi. Pertanto *INDEX = 0 bit* e *TAG = 16 - 5 = 11 bit*

Cache e conflitti

Memoria cache: qualcosa sui conflitti...

Cache e conflitti: Esercizio 3

Si consideri una cache *1-way associative* (diretta) e si assuma che l'indirizzo fisico sia di *16 bit*, suddiviso in *4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG.
Si consideri inoltre una sequenza di accessi in lettura alla memoria, agli indirizzi seguenti:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

Si supponga che inizialmente ogni blocco nella cache sia *non valido*.

a. Quali saranno i blocchi *validi* alla fine della sequenza di letture?

Cache e conflitti: Esercizio 3

Si consideri una cache *1-way associative* (diretta) e si assuma che l'indirizzo fisico sia di *16 bit*, suddiviso in *4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG.
Si consideri inoltre una sequenza di accessi in lettura alla memoria, agli indirizzi seguenti:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

Si supponga che inizialmente ogni blocco nella cache sia *non valido*.

a. Quali saranno i blocchi *validi* alla fine della sequenza di letture?

I blocchi di cache interessati dalle operazioni di lettura sono quelli con INDEX = 5 (cioè il sesto blocco) e INDEX = C (cioè il tredicesimo blocco). Questi saranno quindi gli unici blocchi *validi*.

Cache e conflitti : Esercizio 3 (continua)

Si consideri una cache *1-way associative* (diretta) e si assuma che l'indirizzo fisico sia di *16 bit*, suddiviso in *4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG. Si consideri inoltre la sequenza di accessi in lettura:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

b. Si verificano conflitti durante la sequenza di lettura? Se sì, quali?

Cache e conflitti : Esercizio 3 (continua)

Si consideri una cache *1-way associative* (diretta) e si assuma che l'indirizzo fisico sia di *16 bit*, suddiviso in *4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG. Si consideri inoltre la sequenza di accessi in lettura:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

b. Si verificano conflitti durante la sequenza di lettura? Se sì, quali?

Gli accessi 1 e 3 fanno riferiscono lo stesso blocco di cache, ma anche di memoria sottostante (stesso TAG), quindi non vi è alcun conflitto.

L'accesso 4 è in conflitto con 2 (stesso INDEX, TAG diversi). Quindi 4 rimuoverà dalla cache il blocco corrispondente a 2.

L'accesso 5 è in conflitto con il 3 e 1, quindi rimuoverà dalla cache il blocco corrispondente agli accessi 3 e 1.

Cache e conflitti : Esercizio 3 (continua)

Si consideri una cache *1-way associative* (diretta) e si assuma che l'indirizzo fisico sia di *16 bit*, suddiviso in *4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG. Si consideri inoltre la sequenza di accessi in lettura:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

c. Come si presenta la cache (blocchi *validi*) alla fine della sequenza di lettura?

Cache e conflitti : Esercizio 3 (continua)

Si consideri una cache *1-way associative* (diretta) e si assuma che l'indirizzo fisico sia di *16 bit*, suddiviso in *4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG. Si consideri inoltre la sequenza di accessi in lettura:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

c. Come si presenta la cache (blocchi *validi*) alla fine della sequenza di lettura?

	VALID	TAG	DATA
INDEX: 5	1	8F	Blocco all'indirizzo specificato in 5
INDEX: C	1	10	Blocco all'indirizzo specificato in 4

Cache e conflitti : Esercizio 3 (continua)

c. Si consideri ancora la sequenza di accessi in lettura:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

ma si supponga che, con la stessa composizione dell'indirizzo (*4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG), la cache sia *2-way associative*. Si verificano conflitti durante la sequenza di lettura? Se si, quali?

Cache e conflitti : Esercizio 3 (continua)

c. Si consideri ancora la sequenza di accessi in lettura:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

ma si supponga che, con la stessa composizione dell'indirizzo (*4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG), la cache sia *2-way associative*. Si verificano conflitti durante la sequenza di lettura? Se si, quali?

Questa volta INDEX si riferisce ai set, e ogni set contiene due blocchi (quindi in totale i blocchi sono 32).

Visto che nella sequenza ogni set è riferito al più due volte, *non* si verificheranno conflitti.

Cache e conflitti : Esercizio 3 (continua)

Sequenza di letture (*4 bit* di OFFSET, *4 bit* di INDEX e *8 bit* di TAG), …

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

La situazione dei blocchi validi alla fine della sequenza di lettura è:

Set con INDEX = 5

Blocco	VALID	TAG	DATA
0	1	7F	Blocco all'indirizzo specificato in 1 e 3
1	1	8F	Blocco all'indirizzo specificato in 5

Set con INDEX = C

Blocco	VALID	TAG	DATA
0	1	3C	Blocco all'indirizzo specificato in 2
1	1	10	Blocco all'indirizzo specificato in 4

Cache e conflitti: Esercizio 3 (continua)

d. Si consideri ancora la sequenza di accessi in lettura:

1. `0x7F5A`
2. `0x3CC7`
3. `0x7F5B`
4. `0x10C2`
5. `0x8F50`

e si assuma di avere una cache *2-way associative*, ma con lo stesso spazio dati della prima (2^4 blocchi di 2^4 *B*). Si verificano conflitti durante la sequenza di lettura?

Cache e conflitti: Esercizio 3 (continua)

d. Si consideri ancora la sequenza di accessi in lettura:

1. `0x7F5A` → `0111 1111 0 101 1010`
2. `0x3CC7` → `0011 1100 1 100 0111`
3. `0x7F5B` → `0111 1111 0 101 1011`
4. `0x10C2` → `0001 0000 1 100 0010`
5. `0x8F50` → `0100 1111 0 101 0000`

e si assuma di avere una cache *2-way associative*, ma con lo stesso spazio dati della prima (2^4 blocchi di 2^4 *B*). Si verificano conflitti durante la sequenza di lettura?

Cache e conflitti: Esercizio 3 (continua)

d. Si consideri ancora la sequenza di accessi in lettura:

1. `0x7F5A` → `0111 1111 0 101 1010`
2. `0x3CC7` → `0011 1100 1 100 0111`
3. `0x7F5B` → `0111 1111 0 101 1011`
4. `0x10C2` → `0001 0000 1 100 0010`
5. `0x8F50` → `0100 1111 0 101 0000`

e si assuma di avere una cache *2-way associative*, ma con lo stesso spazio dati della prima (2^4 blocchi di 2^4 *B*). Si verificano conflitti durante la sequenza di lettura?

I 2^4 blocchi saranno suddivisi in 2^3 set, quindi indirizzati da un INDEX di *3 bit*. OFFSET resta di *4 bit* ed i rimanenti *9 bit* formano il TAG.

Ancora una volta nella sequenza ogni set è riferito al più due volte, quindi *non* si verificheranno conflitti

1. e 3. fanno riferimento allo stesso blocco

Tracce di accesso alla cache dati

Si supponga che un programma sia stato tracciato nei sui accessi alla memoria fisica effettuati da istruzioni di `lw` e `sw`. In particolare, gli accessi tracciati sono i seguenti (ind. a 32 b):

```
0x17 11 00 00
0x17 11 00 04
0x17 20 00 00
0x17 20 00 04
0x17 20 10 04
0x17 20 10 08
0x17 20 10 0a
...
```

La cache dati di primo livello del sistema è di 64 KB, con blocchi da 16 B.
Rispondere alle seguenti domande considerando le due possibili organizzazioni della cache: diretta oppure associativa a 16 vie

1. Per ogni indirizzo tracciato, quali sono TAG, INDEX e OFFSET?
2. Se la cache è inizialmente vuota, quali sono i riferimenti di tipo *miss* o *hit*, e tra i miss quali provocano dei conflitti con il rimpiazzo del blocco?

Tracce di accesso alla cache dati (cont.)

- Il numero di blocchi della cache, ovvero il numero di insiemi nella cache diretta, è pari a 64K/16 = 4K = 2^{12}.
- Per la cache associativa a 16 vie, il numero di insiemi è invece 4K/16 = 2^{8}.
- Per la cache diretta, l'INDEX è di log 2^{12} = 12 bit.
- Per la cache associativa, l'INDEX è di log 2^{8} = 8 bit.
- Il Block Offset è invece di log 16 = 4 bit per entrambe le organizzazioni della cache

Tracce di accesso alla cache dati (cont.)

Cache diretta, TAG, INDEX e OFFSET:

```
0x17 11 00 00    TAG=1711 INDEX =000 OFFSET=0
0x17 11 00 04    TAG=1711 INDEX =000 OFFSET=4
0x17 20 00 00    TAG=1720 INDEX =000 OFFSET=0
0x17 20 00 04    TAG=1720 INDEX =000 OFFSET=4
0x17 20 10 04    TAG=1720 INDEX =100 OFFSET=4
0x17 20 10 08    TAG=1720 INDEX =100 OFFSET=8
0x17 20 10 0a    TAG=1720 INDEX =100 OFFSET=a
...
```

Miss e hit:

`0x17 11 00 00`	miss
`0x17 11 00 04`	hit (stesso blocco del rif. precedente)
`0x17 20 00 00`	miss con conflitto (stesso index, tag differente)
`0x17 20 00 04`	hit (stesso blocco del rif. precedente)
`0x17 20 10 04`	miss
`0x17 20 10 08`	hit (stesso blocco del rif. precedente)
`0x17 20 10 0a`	hit (stesso blocco del rif. precedente)

```
...
```

Tracce di accesso alla cache dati (cont.)

Cache associativa a 16 vie, TAG, INDEX e OFFSET:

```
0x17 11 00 00 TAG=17110 INDEX =00 OFFSET=0
0x17 11 00 04 TAG=17110 INDEX =00 OFFSET=4
0x17 20 00 00 TAG=17200 INDEX =00 OFFSET=0
0x17 20 00 04 TAG=17200 INDEX =00 OFFSET=4
0x17 20 10 04 TAG=17201 INDEX =00 OFFSET=4
0x17 20 10 08 TAG=17201 INDEX =00 OFFSET=8
0x17 20 10 0a TAG=17201 INDEX =00 OFFSET=a
...
```

Miss e hit:

`0x17 11 00 00`	miss
`0x17 11 00 04`	hit (stesso blocco del rif. precedente)
`0x17 20 00 00`	miss con conflitto sull'insieme (stesso index, tag differente)
`0x17 20 00 04`	hit (stesso blocco del rif. precedente)
`0x17 20 10 04`	miss con conflitto sull'insieme (stesso index, tag differente)
`0x17 20 10 08`	hit (stesso blocco del rif. precedente)
`0x17 20 10 0a`	hit (stesso blocco del rif. precedente)

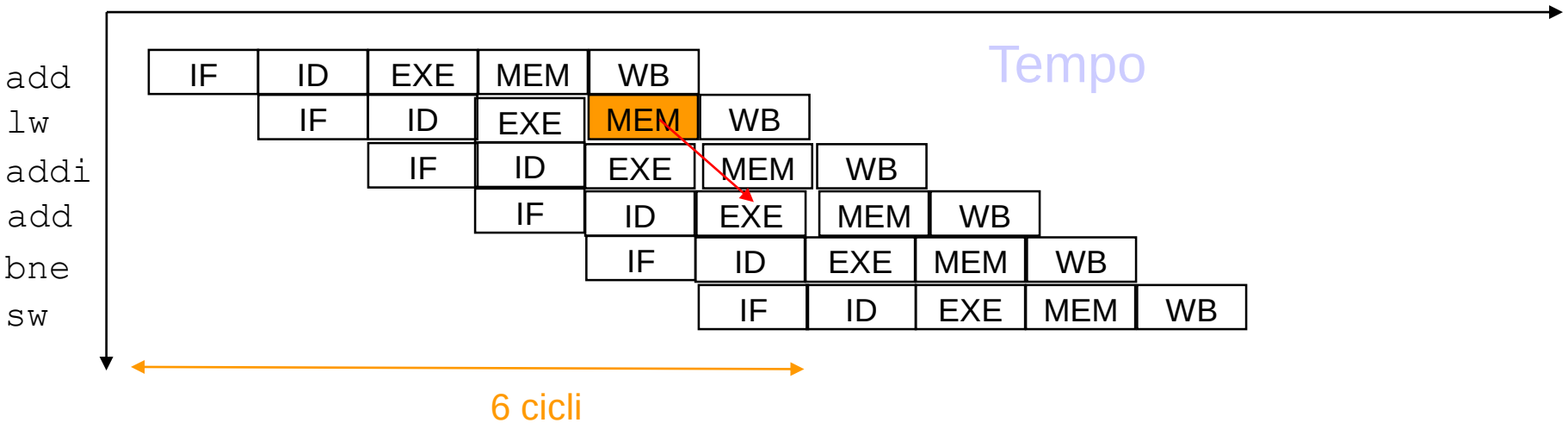
```
...
```

In questo caso, i miss con conflitto sull'insieme non provocano il rimpiazzo del blocco, in quanto ogni insieme contiene ben 16 blocchi.

Pipeline, CPI, e stalli di accesso alla memoria

- Si consideri il loop ottimizzato (vedi es. su pipeline)
- Il diagramma seguente si verifica se la memoria ha un costo di accesso di 1 ciclo di clock (stadio MEM)

```
Start:
  add $t1, $s0, $t0
  lw $t2, 0($t1)
  addi $t0, $t0, 4
  add $t2, $t2, $t3
  bne $t0, $s3, start
  sw $t2, 0($t1)
```

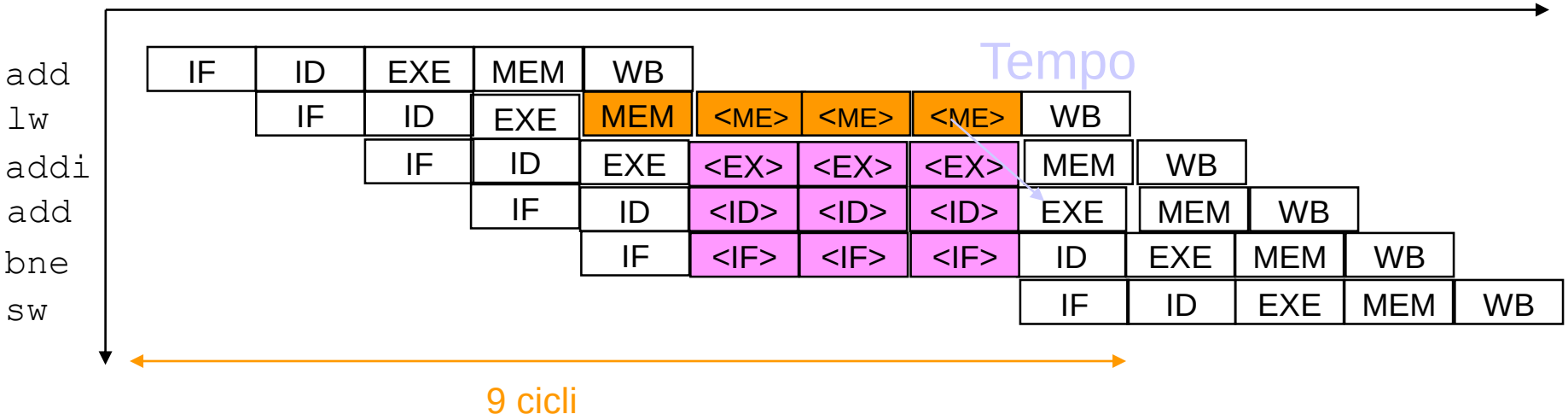


Pipeline, CPI, e stalli di accesso alla memoria

- Abbiamo finora considerato una memoria dati il cui l'accesso costa 1 ciclo di clock (stadio MEM)
- Se in presenza di miss la pipeline andasse in stallo, poiché l'accesso costa 4 cicli, come cambierebbe il diagramma temporale di esecuzione di un ciclo del loop? (si noti che il miss si può avere solo sulla lw, mentre la successiva sw provoca un hit)

Pipeline, CPI, e stalli di accesso alla memoria

- Abbiamo finora considerato una memoria dati il cui l'accesso costa 1 ciclo di clock (stadio MEM)
- Se in presenza di miss la pipeline andasse in stallo, poiché l'accesso costa 4 cicli, come cambierebbe il diagramma temporale di esecuzione di un ciclo del loop? (si noti che il miss si può avere solo sulla lw, mentre la successiva sw provoca un hit)



Cicli totali per eseguire il loop in presenza di miss =
6 cicli normali + 3 cicli di stallo = 9 cicli

Pipeline, CPI, e stalli di accesso alla memoria

- Consideriamo una cache ad accesso diretto con blocchi da 16 B, che all'inizio del loop non contiene nessuna word del vettore acceduto
- Calcolare il CPI totale del loop precedente (ottimizzato)

Pipeline, CPI, e stalli di accesso alla memoria

- Consideriamo una cache ad accesso diretto con blocchi da 16 B, che all'inizio del loop non contiene nessuna word del vettore acceduto
- Calcolare il CPI totale del loop precedente (ottimizzato)
- Soluzione:
  - Poiché i blocchi si accedono in sequenza in word da 4 B, abbiamo sempre un miss e 3 hit
  - Cicli totali per eseguire il loop in presenza di miss = 9
  - Cicli totali per eseguire il loop in presenta di hit = 6
  - Cicli totali per le 256 iterazioni = 0.25 * 256 * 9 + 0.75 * 256 * 6 = 64*9 + 192*6 = 1728
  - CPI = cicli totali / IC = 1728 / (256 * 6) = 1.125

Pipeline, CPI, e stalli di accesso alla memoria

- Consideriamo una cache ad accesso diretto con blocchi da 16 B, che all'inizio del loop non contiene nessuna word del vettore acceduto
- Calcolare il CPI totale del loop precedente (ottimizzato)
- Soluzione:
  - Poiché i blocchi si accedono in sequenza in word da 4 B, abbiamo sempre un miss e 3 hit
  - Cicli totali per eseguire il loop in presenza di miss = 9
  - Cicli totali per eseguire il loop in presenta di hit = 6
  - Cicli totali per le 256 iterazioni = 0.25 * 256 * 9 + 0.75 * 256 * 6 = 64*9 + 192*6 = 1728
  - CPI = cicli totali / IC = 1728 / (256 * 6) = 1.125

- La risposta sarebbe stata diversa se la cache avesse un'organizzazione differente rispetto all'associatività ?

Pipeline, CPI, e stalli di accesso alla memoria

- Consideriamo una cache ad accesso diretto con blocchi da 16 B, che all'inizio del loop non contiene nessuna word del vettore acceduto
- Calcolare il CPI totale del loop precedente (ottimizzato)
- Soluzione:
  - Poiché i blocchi si accedono in sequenza in word da 4 B, abbiamo sempre un miss e 3 hit
  - Cicli totali per eseguire il loop in presenza di miss = 9
  - Cicli totali per eseguire il loop in presenta di hit = 6
  - Cicli totali per le 256 iterazioni = 0.25 * 256 * 9 + 0.75 * 256 * 6 = 64*9 + 192*6 = 1728
  - CPI = cicli totali / IC = 1728 / (256 * 6) = 1.125

- La risposta sarebbe stata diversa se la cache avesse un'organizzazione differente rispetto all'associatività ?
- Soluzione:
  - La risposta è indipendente dal grado di associatività della cache ! Solo località spaziale, influenzata dalla dimensione del blocco. Solo miss certi (compulsory)