


Esercizi sulle prestazioni delle memorie cache

Prof. Alberto Borghese
Dott. Massimo Marchi
Dipartimento di Scienze dell'Informazione
alberto.borghese@unimi.it

Università degli Studi di Milano


Sommario

Impatto delle memorie cache
Progettazione delle memorie cache

Tempo medio di accesso alla memoria

TEMPO DI ACCESSO. E' legato al tempo di accesso del livello inferiore di memoria.

TEMPO DI TRASFERIMENTO. E' legato alla larghezza di banda del canale di comunicazione tra i due livelli di memoria (e.g. bus).

Il tempo medio di accesso alla memoria sarà:

$\mathbf{T_{medio}}$ = HIT_RATE * HIT_TIME + MISS_RATE * MISS_TIME =

HIT_TIME*HIT_RATE+MISS_RATE*(HIT_TIME+MISS_PENALTY) =

HIT_TIME*(HIT_RATE+MISS_RATE)+MISS_RATE*MISS_PENALTY =

HIT_TIME + MISS_RATE * MISS_PENALTY

Hit – massimizzo hit_rate
Miss – Minimizzo miss_penalty (costo delle miss -> stalli)

Impatto di una memoria cache

Il tempo di CPU è composto dal tempo richiesto dalla CPU per eseguire il programma e dal tempo che la CPU trascorre in attesa di risposta dal sottosistema di memoria.

T_{CPU} = (#Cicli della CPU in esecuzione + #Cicli di stallo) * T_{Clock}

Ipotesi:
•Tutti gli stalli di memoria sono dovuti al fallimento di accesso alla cache.
•I cicli di clock utilizzati per un accesso alla cache riuscito (HIT) sono inclusi nei cicli di clock della CPU in esecuzione.

CPI = Numero_Cicli_Clock_per_Istruzione =
Numero di Cicli di Clock / Numero di istruzioni.

Impatto di una memoria cache

Cicli_clock_stallo = #Accessi_Memoria*MISS_RATE*MISS_PENALTY

$Tempo_{CPU}$ Programma = (#Cicli_clock + #Cicli_clock_stallo)*T_{clock} =
#Istruzioni*CPI_{exec}*T_{clock} + #Cicli_clock_stallo*T_{clock}

CPI_{con_cache} = CPI_{exec} + #Cicli_clock_stallo / #Istruzioni =
CPI_{exec}+(#Accessi_memoria/#Istruzioni)*MISS_RATE*MISS_PENALTY

Caso ideale: (100% HIT, 0%MISS): CPI_{con_cache} = CPI_{exec}

Caso senza cache: (100% MISS): CPI_{senza_cache}=
CPI_{exec} + (#Accessi_memoria/#Istruzioni)* MISS_PENALTY

Esercizio su cache

Si consideri il VAX-11/780. La MISS_PENALTY è di 6 cicli di clock, mentre tutte le istruzioni impiegano 8.5 cicli di clock se si ignorano i MISS (stalli della memoria). Ipotizzando un MISS_RATE dell'11% e che vi siano in media 2 riferimenti alla memoria per ogni istruzione,

⇒ Qual è l'impatto sulle prestazioni quando viene inserita la cache reale rispetto ad una cache ideale?

⇒ Qual è l'impatto sulle prestazioni tra il caso di cache reale e senza inserimento della cache?

Soluzione esercizio su cache

Dati di ingresso: MISS_PENALTY=6 CPI_{exec}=8.5 MISS_RATE=0,11
#Accessi_memoria/#Istruzioni = 2

CPI_{con_cache} = 8,5 + 0,11 * (2 * 6) = 9,82
$CPI_{con\_cache\_ideale}$ = 8,5 + 0 * (2 * 6)
CPI_{senza_cache}= 8,5 + 1 * (2 * 6) = 20,5

Perdita in prestazioni (speed-up): $CPI_{con\_cache\_ideale}$ / CPI_{con_cache} => 8,5 / 9,82 = 0,865

Guadagno in prestazioni (speed-up): CPI_{senza_cache} / CPI_{con_cache} => 20,5 / 9,82 = 2,087

Sommario

Impatto delle memorie cache
Progettazione delle memorie cache

Esercizio 1

Si progetti una cache di 16KByte a 8 vie per un sistema con indirizzamento al byte di 32bit, bus dati a 32 bit, bus indirizzi a 30bit, e word di 4 byte. L'ampiezza della cache è di 1 word. In quale cella della cache e con quale tag viene memorizzato il dato letto dall'indirizzo 0x40404040 della memoria principale (lw $t0, 0x40404040? Quanta memoria è necessaria per implementare la cache?

Una cache a 8 vie è una cache in grado di memorizzare 8 blocchi diversi per ogni indice.
16Kbyte corrispondono a 16 / 8 = 2KByte per via. La cache è organizzata a matrice. L'ampiezza è di 4 Byte e l'altezza sarà quindi 2KByte / 4 Byte = 512 = Numero di linee.
Servono quindi 11 bit (log2 (2KByte)) per indirizzare i Byte all'interno della cache. Il tag risulta quindi pari a 30-11 = 19 bit.
Gli 11 bit di indirizzamento all'interno della cache vengono suddivisi come segue. L'ampiezza della linea di ciascuna via della cache è pari a 4 Byte, servono quindi 2 bit. Gli altri 11-2 = 9 bit servono per indirizzare la linea (2^9 = 512 linee)

L'indirizzo a 30 bit viene quindi suddiviso come segue:

	tag	**indice**	unused
bit	**29-11**	**10-2**	**1-0**

L'indirizzo **0x40404040 = 01.00 0000 0100 0000 0100 0.000 0100 00.00**$_2$
corrisponde all'**indice 000 0100 00** = **16** ed al **tag** = **000 0000 1000 0000 1000** = **2056**.

Lo spazio fisicamente necessario per implementare la cache è pari a:
19 bit di tag + 32 bit di dato + 1 bit di validate +1 dirty bit per banco = 53 bit per linea per banco.

Il numero totale di bit sarà quindi:
53bit * 512 linee * 8 vie = 217,088 bit (27,136 Byte).

La memoria quindi a fronte di una capacità di memoria di 16KByte, richiede un torale di 26,5 Kbyte.

Esercizio 2

Si progetti una cache di 128 byte a corrispondenza diretta organizzata in linee di 4 word con word di 2 byte per un sistema con indirizzamento al byte a 16 bit, bus indirizzi a 16 bit. Si consideri il tempo di accesso della cache in lettura/scrittura 2ns ed il tempo di accesso in lettura alla DRAM di 60ns. Si consideri anche che la cache lavora in modalità "write-back" e che il tempo per scaricare una linea di cache sia di 100ns. Data la sequenza di accessi 0,2,8,10,132,2 dire quanto vale il tempo di accesso totale nell'ipotesi che i bit di validate siano tutti a 0 all'inizio.

La cache ha un'ampiezza della linea pari a 4 word * 2 Byte / Word = 8 Byte.
Il numero di bit per indirizzare i byte all'interno della cache è qunidi pari a log2 (128) = 7 ed il numero di bit di tag sarà quindi: 16 – 7 = 9.
Il numero di bit necessari ad inidirizzare i Byte all'interno di una linea è pari a:
log(ampiezza linea in Byte) = 3 = (log2 (8)).
Il numero di linee sarà quindi pari a: 7 (#bit per indirizzare i Byte dentro la cache) – 3 = 4.
Questo numero si può ottenere anche dividendo la capacità della cache per la capacità della linea, cioè 128 Byte / 8 Byte = 16 linee => 3 bit.

L'indirizzo viene quindi così suddiviso:

	tag (9 bit)	**indice** (4 bit)	**offset** (2 bit)	unused
bit	**15-7**	**6-3**	**2-1**	**0**

Lo spazio occorrente per implementare questa cache è quindi:
(9 bit di tag + 1 bit di validate + 1 dirty bit * 64 bit per linea) * 16 linee =1184 bit (148 Byte)

Esercizio 2

Si progetti una cache di 128 byte a corrispondenza diretta organizzata in linee di 4 word con word di 2 byte per un sistema con indirizzamento al byte a 16 bit, bus indirizzi a 16 bit. Si consideri il tempo di accesso della cache in lettura/scrittura 2ns ed il tempo di accesso in lettura alla DRAM di 60ns. Si consideri anche che la cache lavora in modalità "write-back" e che il tempo per scaricare una linea di cache sia di 100ns. Data la sequenza di accessi 0 (load),2 (store),8 (load),10 (load),132 (load),4 (store) dire quanto vale il tempo di accesso totale nell'ipotesi che i bit di validate siano tutti a 0 all'inizio.

La cache ha un'ampiezza della linea pari a 4 word * 2 Byte / Word = 8 Byte.
Il numero di bit per indirizzare i byte all'interno della cache è qunidi pari a log2 (128) = 7 ed il numero di bit di tag sarà quindi: 16 – 7 = 9.
Il numero di bit necessari ad inidirizzare i Byte all'interno di una linea è pari a: log(ampiezza linea in Byte) = 3 = (log2 (8)).
Il numero di linee sarà quindi pari a: 7 (#bit per indirizzare i Byte dentro la cache) – 3 = 4.
Questo numero si può ottenere anche dividendo la capacità della cache per la capacità della linea, cioè 128 Byte / 8 Byte = 16 linee => 3 bit.

L'indirizzo viene quindi così suddiviso:

	tag (9 bit)	**indice** (4 bit)	**offset** (2 bit)	unused
bit	**15-7**	**6-3**	**2-1**	**0**

Lo spazio occorrente per implementare questa cache è quindi:
(9 bit di tag + 1 bit di validate + 1 dirty bit * 64 bit per linea) * 16 linee =1184 bit (148 Byte)

Esercizio 2 - accessi

L'accesso a **0 (000000000-0000-00-0)** genera una **miss** perchè la cache è totalmente vuota. il blocco corrispondente con **tag=0** e **indice=0** viene caricato dalla memoria in un solo accesso (64 bit di bus dati). TEMPO: 60ns
L'accesso a **2 (000000000-0000-01-0)** corrisponde allo stesso blocco, **indice=0** quindi abbiamo una **hit** (con dirty bit impostato a 1). TEMPO: 2ns.
L'accesso a **8 (000000000-0001-00-0)** corrisponde al **tag=0**, **indice=1** che non è ancora stato letto quindi è una **miss**. TEMPO: 60ns.
L'accesso a **10 (000000000-0001-01-0)** corrisponde a **tag=0**, **indice=1**; in questo caso è una **hit**. TEMPO: 2ns.
L'accesso a **132 (000000001-0000-10-1)** corrisponde a **tag=1**, **indice=0**; in questo caso è una **miss (**da notare che la linea 0 contiene anche il dirty bit = 1 perchè scritta dalla seconda istruzione. Occorre quindi procedere anche al write-back della linea 0 con tag 0). TEMPO: 60ns + 100ns.
L'accesso a **4 (00000000-0000-10-0)** corrisponde a **tag=0**, **indice=0**, non è ancora presente in cache quindi **miss**. Occorre prima caricare la linea aggiornata dalla memoria e poi operare la scrittura. TEMPO: 60ns.

TEMPO TOTALE = 60 + 2 + 60 + 2 + 160 + 60 = 344 ns.

Esercizio 3

Si progetti una cache a corrispondenza diretta di 64Kbytes, a 4 vie, che utilizzi word di 8 Byte e memorizzi 4 word per ogni linea, bus inidirizzi su 16 bit, indirizzamento al byte. Dimensionare tutti i parametri della cache.

La cache ha una capacità per banco di 64KByte / 4 = 16 Kbyte.
Il numero di bit per indirizzare i byte all'interno della cache è qunidi pari a log2 (16K) = 14 bit ed il numero di bit di tag sarà quindi: 16 – 14 = 2.
L'ampiezza della linea per banco sarà di: 4 Word * 8 Byte / Word = 32 Byte.
L'altezza della cache sarà perciò di 16 Kbyte / 32 Byte = 512 Linee.
Il numero di bit necessari ad inidirizzare i Byte all'interno di una linea è pari a: log2 (32) = 5.
Il numero di linee sarà quindi pari a: 14 -5 = 9.
La parola all'interno della linea viene selezionata dai bit 3 e 4 dell'indirizzo, mentre i bit 0, 1, 2 non vengono utilizzati ed indirizzano uno degli 8 Byte che costituiscono la parola.

La dimensione totale della cache è pari a:
(2 (bit di TAG) + 1 (bit validità) + 1 (dirty bit) + 32 * 8 (dati)) * 512 * 4 = 532,480 bit (66,560 Byte) = 65KByte

Con un tag piccolo, c'è un overhead sull'allocazione di memoria piccolo (1KByte).

La scomposizione dell'indirizzo sarà:

	tag (2 bit)	**indice** (13 bit)	**offset** (2 bit)	unused (1 bit)
bit	**14-15**	**13-5**	**5-4**	**3**

Si noti che in questo caso la RAM può essere copiata tutta in cache, ricopiando i diversi macro blocchi nei diversi banchi della memoria cache.

Sommario

Impatto delle memorie cache
Progettazione delle memorie cache