Tutorato di architettura degli elaboratori

Caching

Andrea Gasparetto – andrea.gasparetto@unive.it

Esercizio 1

Dati una cache con 4096 blocchi, e con dimensione dell'INDEX di 10 b, determinare il grado di associatività della cache stessa.

Determinare la dimensione dell'indirizzo fisico, considerando che la TAG è 18 b, mentre la dimensione della parte dati della cache è 64 KB.

Considerare infine la seguente sequenza di accessi alla memoria (sequenza di lw):

1. 0x000000a4
2. 0x000000ac
3. 0x100000a4
4. 0x100000ac
5. 0x200000a4
6. 0x200000ac
7. 0x300000a4
8. 0x300000ac
9. 0x400000a4

Supporre che tutti blocchi della cache siano non validi. Quali accessi provocano hit, miss, oppure miss con conflitto?

Soluzione

No. di vie = num. blocchi / 2^{INDEX} = 4096 / 1024 = 4.

Dim_blocco = 64 KB / 4096 = 2^{16} / 2^{12} = 2^4

Ind. Fisico = INDEX + TAG + OFFSET = 10 + 18 + $\log_2$ (Dim_blocco) = 28 + 4 = 32 b.

Per quanto riguarda i nove accessi, i 4 b dell'OFFSET corrispondono alla cifra esadecimale meno significativa dell'indirizzo fisico. Inoltre, bisogna considerare che INDEX, ovvero la parte bassa del block address, ovvero la parte bassa dell'indirizzo fisico una volta eliminato OFFSET, è grande 10 b. Quindi, in tutti e nove i casi INDEX rimane costante. Quindi, tutti e nove gli accessi fanno riferimento al solito INDEX, ovvero allo stesso set a 4 vie. Inoltre le coppie di accessi (1,2), (3,4), (5,6), (7,8) fanno riferimento allo stesso blocco. Infatti, se

si considerano gli indirizzi presenti in ogni coppia, si può osservare che cambiano solo i 4 bit meno significativi (l'ultima cifra esadecimale), ovvero l'OFFSET. Gli accessi 1, 3, 5 e 7 sono sicuramente miss, poiché ogni volta cambia il TAG (la parte alta dell'indirizzo). I rispettivi blocchi verranno ospitati in blocchi diversi del set a 4 vie individuato dall'INDEX. Gli accessi 2, 4, 6 e 8 sono invece hit.

Infine, l'ultimo accesso (il 9) è anch'esso un miss a causa del TAG ancora diverso. Quest'ultimo provoca però un conflitto, poiché tutti e 4 i blocchi del set sono a questo punto occupati.

Esercizio 2

Considerare due cache la cui parte dati è di dimensione 128 KB, con blocchi di 64b. Dato un indirizzo fisico di 32b, determinare le organizzazioni delle cache per dimensioni della TAG di 16b e di 18b.

Soluzione

Tag di 16b

OFFSET di 3b

INDEX = 32 – TAG – OFFSET = 32 – 16 – 3 = 13 b

Gli insiemi indirizzabili sono quindi 2^{13} = 8 K

Ogni insieme è composto di 128 KB / 8 K = 16 B, ovvero da 2 (= DimSet / DimBlocco = 16 / 8) blocchi da 8 B = 64 b. L'organizzazione della cache è quindi 2-way associative.

TAG di 18 b

OFFSET di 3b

INDEX = 32 – TAG – OFFSET = 32 – 18 – 3 = 11 b

Gli insiemi indirizzabili sono quindi 2^{11} = 2 K

Ogni insieme è composto di 128 KB / 2 K = 64 B, ovvero da 8 (= DimSet / DimBlocco = 64 / 8) blocchi da 8 B = 64 b. L'organizzazione della cache è quindi 8-way associative.

Esercizio 3

Si supponga che un programma faccia riferimento alla memoria con un puntatore p, inizialmente uguale a p=0x17100004 (ind. fisico di 32 b), e via via incrementato del valore 0x80 (=128 in decimale).

Si supponga di avere una cache di 64 KB (parte dati), con blocchi da 16 B. L'organizzazione della cache può essere diretta o associativa a 2 vie. Calcolare la dimensione dell'INDEX per le due organizzazioni di cache.

Rispetto alla cache diretta, si supponga che questa sia all'inizio vuota. Valutare per quale valore dell'indirizzo (e per quale corrispondente valore di INDEX) si verifica il primo conflitto.

Suggerimento: trasformare l'incremento di indirizzo (0x80) in un incremento al valore dell'INDEX. Si ricorda che il valore di INDEX viene incrementato mod 2^d, dove d è la dimensione in bit dell'INDEX stesso.

La risposta sarebbe diversa per la cache associativa?

Soluzione

Il numero di blocchi della cache è 64K = 16 = 4K. Per la cache associativa, il numero di insiemi è 4K = 2 = 2K.

Per la cache diretta, l'INDEX è di $\log_2 4K = \log_2 2^{12} = 12$ bit, mentre per la cache associativa, l'INDEX è di

$\log_2 2K = \log_2 2^{11} = 11$ bit

Il Block Offset è invece di $\log_2 16 = 4$ bit per entrambe le organizzazioni della cache.

Il programma accede alla cache con la seguente sequenza di indirizzi fisici:

0x17 10 00 04 INDEX= 0

0x17 10 00 84 INDEX= 8

0x17 10 01 04 INDEX=16

0x17 10 01 84 INDEX=24

0x17 10 02 04 INDEX=32

0x17 10 02 84 INDEX=40

0x17 10 03 04 INDEX=48

0x17 10 03 84 INDEX=56

0x17 10 04 04 INDEX=64

Incrementare di 128 il valore di p corrisponde a incrementare di 128 = 16 = 8 il valore dell'INDEX (il block offset rimane sempre uguale a 0).

Cache diretta: all'inizio, rispetto al primo riferimento alla memoria p=0x17100004, abbiamo che INDEX=0. Quando avviene l'n ¡ esimo riferimento, abbiamo INDEX= $(0 + n * 8) \bmod 2^{12}$. Il primo conflitto arriva quando INDEX ritorna uguale a 0, che si verifica quando $n * 8 = n * 2^3 = 2^{12}$, ovvero per $n = 2^9$. L'indirizzo di memoria sarà

0x17100004 + 29¢ 0x80 = 0x17100004 + 0x10000 = 0x17110004

Cache associativa: anche in questo caso all'inizio abbiamo che INDEX=0. Per cui, quando avviene l'n-esimo riferimento, abbiamo che INDEX= $(0 + n * 8) \bmod 2^{11}$. INDEX quindi ritorna uguale a 0 quando $n* 8 = n * 23 = 2^{11}$, ovvero per $n = 2^8$. Nota che questo NON causa un conflitto, perché l'insieme della cache è composto da 2 blocchi. Lo stesso avviene per tutti gli altri riferimenti alla memoria, fino al riferimento corrispondente a $n = 2^9$, quando INDEX ritorna uguale a 0. In questo caso entrambi i blocchi dell'insieme corrispondente a INDEX=0 sono occupati, e si verifica il conflitto. L'indirizzo di memoria sarà

0x17100004 + 29¢ 0x80 = 0x17100004 + 0x10000 = 0x17110004, con INDEX=0.

Esercizio 4

Considerare l'esecuzione di un programma P su di una data CPU. Calcolare il CPI ideale, considerando che, senza gli effetti della cache, il CPI medio delle load/store sarebbe 4.5, il CPI medio delle altre istruzioni sarebbe 2, mentre la percentuale di load/store è del 40%. Considerando i miss della cache si ottiene un CPI reale pari a 3.6. Sapendo che Instruction miss rate = 4%, Data miss rate = 2.5%, determinare il Miss penalty (in cicli).

Calcolare i tempi, reali e ideali, per eseguire P, considerando che IC = 200 Milioni, mentre la frequenza della CPU è di 500 MHz.

Soluzione

$CPI_{ideale} = 0.4 * CPI_{load} + 0.6 * CPI_{altro} = (0.4 * 4.5) + (0.6 * 2) = 3.$

Sia x = Miss penalty. Abbiamo che:

$CPI_{reale} = CPI_{ideale} + 0.04 * x + 0.4 * 0.025 * x$

3.6 (CPI reale) = 3 + 0.04 * x + 0.4 * 0.025 * x

x = 0.6/(0.04 + 0.4 * 0.025) = 12 cicli

Se la CPU è a 500 MHz, allora T = 1/(500 M) s = 2 * 1/G sec. = 2 ns.

Il tempo di esecuzione ideale è quindi $CPI_{ideale} * IC * T = 3 * 200\ M * 2$ ns = 1200 M/G s = 1.2 s.

Il tempo di esecuzione reale è quindi $CPI_{reale} * IC * T = 3.6 * 200\ M * 2$ ns = 1440M/G s = 1.44 s.

Esercizio 5

Abbiamo le seguenti misure relative all'esecuzione di un certo programma su un processore a 2 GHz:

$CPI_{ideale} = 2$ $Perc_{lw/sw} = 20\%$ Data_miss_rate = 30%

Instr_miss_rate = 5% IC=10 Milioni $T_{exe} = 65$ msec

Calcolare il Miss penalty in ns.

Modificando la cache, e mantenendo inalterato il resto del sottosistema di memoria, si osserva un miglioramento del Data miss rate. Se otteniamo un tempo di esecuzione $T_{exe} = 40$ ms, quanto vale il nuovo Data miss rate?

Soluzione

1)

Sappiamo che:

$CPI_{miss} = (Instr\_miss\_rate + Data\_miss\_rate * Perc_{lw/sw})\ miss\_penalty =$

$= (0.05 + 0.06)\ miss\_penalty = 0.11\ miss\_penalty$

$CP\ I = CPI_{ideale} + CPI_{miss} = 2 + 0.11\ miss\_penalty$

$No\_cicli = CPI * IC = (2 + 0.11\ miss\_penalty) * 10^7$

$T = 1/2GHz = 1 / (2 * 10^9)$ s = 0.5 ns

$T_{exe} = No\_cicli * T = (2 + 0.11\ miss\_penalty) * 10^7 * 0.5$ ns =

$= (2 + 0.11\ miss\_penalty) * 5 * 10^6$ ns = $(2 + 0.11\ miss\_penalty) * 5$ ms

Considerando che $T_{exe} = 65$ ms, abbiamo:

$65 = (2 + 0,11 \text{ miss penalty}) * 5$

$65 = 10 + 0.55 \text{ miss penalty}$

$\text{Miss\_penalty} = (65 - 10) / 0.55 = 100$ cicli.

Per esprimere il miss_penalty in ns, basta moltiplicare per T:

$\text{miss\_penalty} = \text{cicli} * T = 100 * 0.5 \text{ ns} = 50 \text{ ns}$

2)

Sia X il nuovo valore del Data_miss_rate:

$CPI_{miss} = (\text{Instr\_miss\_rate} + X * Perc_{lw/sw})\ \text{miss\_penalty} = (0.05 + 0.2 * X)\ 100 = 5 + 20 * X$

$CPI = CPI_{ideale} + CPI_{miss} = 2 + (5 + 20\ X) = 7 + 20\ X$

$\text{No\_cicli} = CP\ I * IC = (7 + 20\ X)\ 10^7$

$T'_{exe} = \text{No\_cicli} * T = (7 + 20\ X)\ 10^7 * 0,5 = (7 + 20\ X) * 5 * 10^6 \text{ ns}$

Metto nell'equazione il tempo di esecuzione dato dalla consegna e calcolo il nuovo Data_miss_rate risolvendo l'equazione per X

$T'_{exe} = (7 + 20\ X) * 5 * 10^6 \text{ ns} = 40 \text{ ms} (== 40 * 10^6 \text{ ns})$

$35 * 10^6 + 10^8\ X = 40 * 10^6$

$10^8\ X = 5 * 10^6$

$X = 5 * 10^{-2} = 0.05 = 5\%$

Esercizio 6

Per una cache diretta composta da 256 blocchi di 16 B, con un indirizzo fisico di 32 b, determinare Index, Tag e Offset.

Quanti bytes di dati, in totale, si possono memorizzare nella cache? (1)

Quanti bytes di memoria sono necessari per realizzare la cache (includendo tag, valid bit, e dati)? (2)

Data la sequenza di accessi alla cache, le cui entry sono all'inizio tutte non valide, individuare quali sono hit e miss, e quali miss provocano un conflitto:

0x1F FA BA C0

0x1F FA BB C0

0x1F FA BA C4

0x1B FA BA C0

Soluzione

INDEX = $\log_2 256 = 8$ b

OFFSET = $\log_2 16 = 4$ b

TAG = $32 - 8 - 4 = 20$ b

1) Size dati = $256 * 16$ B = 2^{12} B = 4 KB

2) Size totale = (TAG+Valid+Blocco) $*$ 256 = (Tag+Valid) $*$ 256 b + Blocco $*$ 256 B =

= (TAG+Valid) $*$ 256 / 8 B + Blocco$*$256

B = $(20 + 1) * 256/8 + 16 * 256 = 4768$ B

3)

0x1F FA BA C0 miss (INDEX=AC)

0x1F FA BB C0 miss (INDEX=BC)

0x1F FA BA C4 hit (INDEX=AC, diverso OFFSET)

0x1B FA BA C0 miss con conflitto (INDEX=AC, diverso TAG)

Esercizio 7

Un computer a 1 GHz, nell'eseguire un certo programma, ha una prestazione ideale di 500 MIPS (senza considerare la cache e i miss relativi).

Calcolare il CPI medio ideale.

Considerando la cache, il CPI reale misurato diventa uguale a 2,48. Conoscendo che il data e l'instruction miss rate sono entrambi uguali al 2%, e che la percentuale di load/store è del 20%, calcolare il miss penalty.

Soluzione

Poiché il processore è a 1 GHz, e 1/2 G (500 M) sono le istruzioni eseguite in un secondo, allora abbiamo che per ogni istruzione impieghiamo in media 2 cicli (CPI=2).

Più formalmente:

Texe = #cicli / Freq = IC$*$CPI / F

MIPS = IC / Texe$*10^6$ = IC$*$F / IC$*$CP I$*10^6$ = F / CP I$*10^6$

Sostituendo, abbiamo che $500 = 1G * 10^{-6}$/CPI, e quindi:

CPI = 10^3 / 500 = 2.

Quello appena calcolato è il CPI ideale, per cui i cicli per istruzione dovuto ai miss corrisponde a

$CPI_{reale} - CPI_{ideale} = 2.48 - 2 = 0.48$

In particolare:

$0.48 = 2\% * Penalty + 2\% * 20\% * Penalty / IC = 0.02 * Penalty + 0.02 * 0.2 * Penalty$

Da cui il miss penalty espresso in cicli corrisponde a:

$Penalty = 0.48 / 0.024 = 20$

Esercizio 8

Per una certa architettura e per un certo compilatore $Comp_A$, i CPI medi per le varie classi di istruzioni (senza considerare l'effetto della cache) sono i seguenti:

$CPI_{l/s} = 2$, $CPI_{aritm} = 1$, $CPI_{fp} = 4$, $CPI_{b/j} = 1.8$

Il codice macchina prodotto da $Comp_A$ contiene istruzioni delle varie classi, che appaiono con le seguenti percentuali:

$Perc_{l/s} = 30\%$, $Perc_{aritm} = 40\%$, $Perc_{fp} = 10\%$, $Perc_{b/j} = 20\%$.

Calcolare il numero di cicli totali come funzione di IC_A, ovvero il numero di istruzioni eseguite dal programma compilato con $Comp_A$.

Impiegando un nuovo compilatore, $Comp_B$, i CPI medi rimangono invariati, mentre le percentuali delle varie classi di istruzioni diventano:

$Perc_{l/s} = 20\%$, $Perc_{aritm} = 60\%$, $Perc_{fp} = 10\%$, $Perc_{b/j} = 10\%$.

Considerando che il numero di cicli totali spesi per l'esecuzione del programma compilato con $Comp_B$ è uguale $1.82 * IC_A$, calcolare IC_B in funzione di IC_A, valutando se il numero di istruzioni eseguite aumenta o diminuisce.

Considerare infine l'effetto della cache sul tempo di esecuzione del caso A.

Si sa che data miss rate=5% e miss penalty=5 cicli, e il numero di cicli realmente necessari per l'esecuzione del programma (compilato con $Comp_A$) è maggiore di quello ideale, ed è pari a $C_A = IC_A * 1.935$. Qual è l'instruction miss rate?

Soluzione

Calcoliamo i due CPI medi:

$$CPI_A = 2 * 0.3 + 1 * 0.4 + 4 * 0.1 + 1.8 * 0.2 = 1.76$$

$$CPI_B = 2 * 0.2 + 1 * 0.6 + 4 * 0.1 + 1.8 * 0.1 = 1.58$$

Il numero di cicli nel primo caso risulta uguale a:

$$C_A = CP\ I_A * IC_A = IC_A * 1.76$$

Quindi il caso A è più veloce del caso B, per il quale è noto che $C_B = 1.82 * IC_A$.

Nel secondo caso abbiamo che:

$$C_B = IC_B * CP\ I_B = IC_B * 1.58$$

Dalla traccia del problema risulta che

$C_B = IC_A * 1.82$, per cui $IC_B * 1.58 = IC_A * 1.82$

e quindi

$IC_B = 1.15 * IC_A$

Vediamo gli effetti della cache. Abbiamo che:

Cicli_miss = cicli_misurati – cicli_ideali = $(IC_A * 1.935) - (IC_A * 1.76) = IC_A * 0.175$.

Ma:

Cicli_miss = ((ist_miss_rate $* IC_A$) + $(0.05 * 0.3 * IC_A)) * miss\_penalty$

Sostituendo:

$IC_A * 0.175 = ((ist\_miss\_rate * IC_A) + (0.05 * 0.3 * IC_A)) * 5$

$0.175 = 5 * ist\_miss\_rate + 5 * 0.015$

Ist_miss_rate = 0.1/5 = 0.02

Quindi l'Instruction miss rate è del 2%.

Esercizio 9

Determinare il numero di cicli totali (ed il relativo CPI) per completare il seguente loop MIPS in esecuzione sulla CPU pipeline vista a lezione (con forwarding e file register speciale, e delayed branch). Per questo calcolo, ignorare i cicli per riempire la pipeline.

Assumere che dati e codice siano presenti in cache all'inizio dell'esecuzione.

```
addi $20, $6, 1024*4          # $6 è l'indirizzo iniziale di un array di word
loop:
    lw $9, 0($6)
    add $9, $9, $10
    sw $9, 0($6)
    addi $6, $6, 4
    bne $6, $20, loop
```

Considerare ora che i dati acceduti non siano presenti in cache, ma che comunque essi siano presenti in memoria principale. Ogni blocco della cache è di 16 word, mentre il cache penalty è di 100 nsec. Ricalcolare il numero di cicli totali e il CPI, considerando una CPU a 500 MHz.

Soluzione

Il loop interno viene eseguito per 1024 volte. Tutte le istruzioni, a meno del riempimento della cache, hanno un CPI uguale a 1, eccetto lw e bne, il cui CPI è 2 (a causa del ciclo aggiuntivo di load delay e di

branch delay). In particolare, il load delay dipende dalla dipendenza RAW, tra l'istruzione lw e l'istruzione add successiva.

Quindi ogni loop impiega 5 + 2 cicli, e il numero di cicli totali è:

#_cicli = 1 + 1024 ∗ 7 = 7169.

CP I = cicli/IC = 7169 / (1 + 5 ∗ 1024) = 7169/5121 = 1.4

Se consideriamo la cache, rispetto alle lw abbiamo 1 miss e 15 hit, poiché ogni blocco della cache è di 16 word. In corrispondenza delle sw abbiamo tutti hit, poiché la word acceduta è stata caricata precedentemente dalla lw. I miss sono quindi

Miss = cicli_del_loop / 16 = 1024/16 = 64

Bisogna esprimere il miss penalty in termini di cicli. Ogni ciclo è

T = 1 / 500 MHz = 2 / 10^9 s = 2 ns

In termini di cicli, abbiamo un miss penalty uguale a 100/T = 100/2 = 50 cicli.

Quindi i cicli spesi sono:

#_cicli = 7169 + 64 ∗ 50 = 10369.

CP I = #_cicli / IC = 10369 / (1 + 5 ∗ 1024) = 10369 / 5121 = 2.02