Architettura degli Elaboratori

a.a. 2012/13 - terzo appello, 5 settembre 2013

Riportare nome, cognome, numero di matricola e corso A/B

Domanda 1

Una unità di elaborazione U è così definita:

1. contiene due memorie A[1K], B[4K] di interi;
2. riceve nondeterministicamente messaggi da U0 e da U1, con priorità a U0, e invia messaggi a U2;
3. per ogni indirizzo J di A, A[J] è *significativo* se e solo se è stato aggiornato con un valore X inviato da U0 con il messaggio (X, J);
4. per ogni indirizzo I di A ricevuto da U1, *se e quando A[I] è significativo* U invia a U2 il numero di locazioni di B il cui contenuto è minore di A[I] *e* A[I] diviene nuovamente non significativo;
5. non è necessario che l'ordine dei messaggi inviati a U2 sia lo stesso dei messaggi ricevuti da U1.

Scrivere il microprogramma di U con l'obiettivo di massimizzare la banda offerta alle richieste di U1.

Ulteriore specifica comunicata in aula all'inizio della prova: gli indirizzi inviati da U1 sono tutti distinti.

Domanda 2

Si consideri l'utilizzo che viene fatto di una memoria esterna interallacciata in una architettura con cache. Sia *m* il numero di moduli. L'interfaccia di memoria esterna della CPU permette di inviare richieste del tipo (indirizzo fisico, parola, operazione).

Confrontare la banda offerta e la latenza della soluzione consistente in *m* unità distinte con la soluzione consistente in una sola unità contenente *m* componenti logici memoria, considerando le modalità Write-Back e Write-Through.

Nel caso di *m* unità distinte è necessario specificare la struttura che rende possibile la comunicazione dalla CPU a tali unità, nell'ipotesi di usare solo collegamenti dedicati.

Domanda 3

Valutare il tempo di completamento del programma che esegue la computazione:

int A[N], B[N], C[N];

$\forall$ *i = 0 .. N – 1:*

C[i] = A[i] ∗ B[1 + A[i] % N] + 1

per una CPU pipeline, scalare, in order, con

- comunicazioni a doppia bufferizzazione,
- Unità Esecutiva a 4 stadi,
- cache dati primaria su domanda, Write-Through, blocchi di ampiezza $\sigma = 8$,
- cache secondaria on-chip con prefetching,
- memoria esterna interallacciata con σ moduli e ciclo di clock uguale a 200τ.

Soluzione

Domanda 1

Utilizziamo una memoria S[1K] con locazioni di 2 bit per indicare lo stato di ogni locazione di A:

- S[J] = 00 se A[J] è non significativo e non è stata ricevuta da U1 una richiesta per A[J];
- S[J] = 01 se A[J] è non significativo ed è stata ricevuta da U1 una richiesta per A[J]. Quindi U si ricorda di essere in attesa di poter operare su A[J] quando verrà eventualmente reso significativo con una comunicazione da U0. In altre parole, U opera *out-of-order*: ricevendo da U1 una richiesta per A[J] non aggiornato, prosegue a ricevere nuove richieste da U1: Questo permette di rispettare il requisito sulla banda offerta a U1 ed è in linea con la caratteristica 5;
- S[J] = 1- se A[J] è significativo.

Siano (X, J) i registri di interfaccia di ingresso da U0 e I quello da U1. Poiché è necessario usare una variabile di condizionamento complessa per testare S[I], non ci sono ripercussioni sulla lunghezza del ciclo di clock usando la variabile di condizionamento complessa *segno(contenuto di locazione di B – valore)* per l'elaborazione su B (punto 4) allo scopo di minimizzare il numero di cicli di clock.

0. (RDY0, RDY1, S[I] = 0 0 - -) nop, 0;

   // richiesta da U1 soddisfacibile //

   (= 0 1 1 -) reset RDY1, set ACK1, 00 → S[I], A[I] → V, 0 → IND, 0 → C, 1;

   // richiesta da U1 non soddisfacibile //

   (= 0 1 0 -) reset RDY1, set ACK1, 01 → S[I], 0;

   // messaggio da U0, non attesa su A[J] //

   (= 1 - - 0) reset RDY0, set ACK0, X → A[J], 10 → S[I], 0;

   // messaggio da U0, attesa su A[J] //

   (= 1 - - 1) reset RDY0, set ACK0, 00 → S[J], X → V, 0 → IND, 0 → C, 1

1. (IND_0, segno (B[INDm] – V, ACK2) = 0 0 -) IND + 1 → IND, 1;

   (= 0 1 -) IND + 1 → IND, C + 1 → C, 1;

   (= 1 – 0) nop, 1;

   (= 1 – 1) C → OUT2, set RDY2, reset ACK2, 0

Domanda 2

Lettura blocco

Per questa operazione le due soluzioni hanno la stessa banda, uguale a m/τ_M con τ_M ciclo di clock della memoria, in quanto in entrambe le soluzioni vengono lette m parole contemporaneamente in un singolo ciclo di clock.

La latenza della lettura blocco è costante o logaritmica in *m* a seconda della struttura verso la CPU. Nella soluzione con *m* unità distinte, per poter usare solo collegamenti dedicati la struttura della memoria interallacciata deve prevedere una ulteriore unità di interfaccia che riceve la richiesta dalla CPU e invia *m* richieste di lettura identiche alle *m* unità. La presenza di questa unità aumenta la latenza della quantità $\tau + T_{tr}$ rispetto alla soluzione con singola unità.

Scrittura

La soluzione con *m* unità distinte ha una banda massima uguale a m/τ_M, mentre quella con singola unità ha banda uguale a $1/\tau_M$. Poiché la CPU invia una parola alla volta, tutto questo è valido sia per cache Write-Back (scrittura blocco) che per cache Write-Through, per la quale la banda effettiva è uguale alla massima quando le scritture siano a indirizzi consecutivi. Per scritture a indirizzi non consecutivi con Write Through la banda con *m* unità distinte è stimabile come $\sqrt{m}\,\tau_M$.

Per la latenza, nel caso Write-Back con *m* unità distinte la penalità dell'unità di interfaccia (che serve da smistatore delle richieste) è pagata solo per il primo accesso, dopo di che la latenza di tale unità di interfaccia all'accesso.

Domanda 3

Dal punto di vista della gerarchia di memoria, le strutture A e C godono di sola località, mentre su B può avere *anche* un certo riuso che sfrutteremo con l'annotazione non_deallocare. Data la funzione per indirizzare B, non è però predicibile *quanto* il riuso venga effettivamente esplicitato: per la valutazione, possiamo solo considerare il caso peggiore di non riuso, che porta ad un insieme di lavoro consistente nel blocco corrente di A, B, C, con un numero di fault uguale a $3N/\sigma$ che, essendo C in sola scrittura, nella valutazione si riduce a $3N/\sigma$.

La cache secondaria on-chip con prefetching garantisce la presenza dei blocchi di A. Quella dei blocchi di B non è garantita, non essendo riferiti in modo consecutivo. È però importante ricordare che i blocchi di C2 sono più ampi di quelli di C1 di circa un ordine di grandezza, e questo aumenta in modo significativo la probabilità di trovare in C2 i blocchi richiesti da C1 anche per B. Per questo motivo supporremo di avere probabilità di fault trascurabile per C2 da cui:

$$T_{fault} = N_{fault}\, T_{trasf} = 4\, N\, \tau$$

Ovviamente, una valutazione più precisa è valutata in modo apprezzabile.

Il programma assembler base:

```
1. LOOP: LOAD   RA, Ri, Ra
2.       MOD    Ra, RN, Rindex
3.       INCR   Rindex
4.       LOAD   RB, Rindex, Rb, non_deallocare
5.       MUL    Ra, Rb, Ra
6.       INCR   Ra
7.       STORE  RC, Ri, Ra
8.       INCR   Ri
9.       IF <   Ri, RN, LOOP
```

contiene le dipendenze logiche IU-EU 3 → 4, 6 → 7, 8 → 9. La 3 → 4 risente della dipendenza logica EU-EU 2 → 3. La 6 → 7 risente della dipendenza logica EU-EU 5 → 6. Applicando delayed branch alla 9 mediante lo spostamento della STORE e spostando la 8 tra le 3 e la 4 (con base di C anticipata), si ottiene una delle possibili versioni ottimizzate:

```
1. LOOP: LOAD   RA, Ri, Ra
2.       MOD    Ra, RN, Rindex
3.       INCR   Rindex
4.       INCR   Ri
5.       LOAD   RB, Rindex, Rb, non_deallocare
6.       MUL    Ra, Rb, Ra
7.       INCR   Ra
8.       IF <   Ri, RN, LOOP, delayed_branch
9.       STORE  RC, Ri, Ra
```

Rimane l'effetto

- della dipendenza logica IU-EU 3 → 5, distanza k = 2, con probabilità d_k = 1/9, N_Q = 2 e $L_{pipe\text{-}k}$ = 0, che risente della dipendenza logica EU-EU 2 → 3, di distanza h = 1, con probabilità d_h = 1/9 e $L_{pipe\text{-}h}$ = 4;
- della dipendenza logica IU-EU 7 → 9, di distanza k = 2, con probabilità d_k = 1/9, N_Q = 2 e $L_{pipe\text{-}k}$ = 0, che risente della dipendenza logica EU-EU 6 → 7, di distanza h = 1, con probabilità d_h = 1/9 e $L_{pipe\text{-}h}$ = 4.

Quindi, essendo $t = \tau$ con comunicazioni a doppia bufferizzazione:

- ritardo dovuto a dipendenze logiche:

$$\Delta = \Delta_1 + \Delta_2 = 2\, t\, d_k \left(L_{pipe-k} + N_{Qk} + 1 - k\right) + t\, d_h \left(L_{pipe-h} + 1 - h\right) = \frac{10}{9}\,\tau$$

- tempo di servizio per istruzione in assenza di fault di cache: $T_0 = t + \Delta = \frac{19}{9}\,\tau$
- tempo di completamento in assenza di fault di cache: $T_{c0} = 9\,N\,T_0 = 19\,N\,\tau$
- tempo di completamento senza l'effetto Write Through: $T_c = T_{c0} + T_{fault} = 23\,N\,\tau$

Il tempo di interarrivo delle richieste di scrittura alla memoria esterna è uguale a

$$23\tau$$

Il tempo di servizio in scrittura della memoria esterna con modalità Write Through, essendo le scritture a indirizzi consecutivi, è uguale a

$$200\tau/\sigma = 25\tau$$

quindi la memoria esterna rappresenta il collo di bottiglia e il tempo di completamento è in realtà dato da:

$$T_c = 25\,N\,\tau$$