Architettura degli Elaboratori

a.a. 2012/13 - primo appello, 3 giugno 2013

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

Domanda 1

In una semplice architettura di CPU pipeline scalare, i registri generali sono incapsulati in una unità apposita, RGU, che interagisce con IU ed EU a domanda e risposta. RGU serve una sola richiesta alla volta, dando priorità a IU. Ogni richiesta è relativa a un singolo registro.

IU è rigorosamente in-order ed EU è di latenza unitaria.

Scrivere il microprogramma di RGU e valutarne il ciclo di clock (indicando simbolicamente i ritardi di stabilizzazione interessati) e il tempo di servizio ideale.

Valutare il massimo tempo di risposta di RGU a una richiesta di EU.

Domanda 2

Su una architettura con gerarchia di memoria memoria principale – cache primaria viene eseguito il programma che calcola $\sum_{i=0}^{N-1} F(A[i])$, con *A* array di interi e *F* funzione data.

Sono noti l'ampiezza del blocco, il tempo di servizio di *F* (supposto preponderante rispetto alle altre operazioni del loop), il numero di moduli della memoria principale interallacciata, la latenza di trasmissione dei collegamenti, e la struttura di interconnessione.

Nell'ipotesi che la cache primaria funzioni con prefetching, determinare la minima banda di memoria principale per cui la degradazione di efficienza relativa risulta trascurabile.

Domanda 3

Si consideri l'esecuzione della seguente computazione:

int b, d; int A[N], C[N], X[N];

$\forall$ *i = 0 .. N – 1:*

$X[i] = A[i] * b + C[i] / d$

su una architettura D-RISC con CPU pipeline scalare avente EU a 4 stadi.

Trascurando i fault di cache, confrontare il tempo di completamento del caso di IU in-order e sole ottimizzazioni statiche con quello del caso di IU out-of-order FIFO e sole ottimizzazioni dinamiche, fornendo adeguate spiegazioni.

Soluzione

Domanda 1

RGU contiene la memoria RG[n] dei registri generali e la memoria C[n] dei semafori non-negativi associati a RG. L'algoritmo di sincronizzazione è quello noto (vedi).

Utilizziamo anche una memoria WAIT[n] di booleani per ricordare se IU è in attesa di ricevere il valore di un registro generale: WAIT[i] = 1 sse IU è in attesa di leggere il valore aggiornato di RG[i].

L'interfaccia con IU è composta da RDYIU e RDYOUTIU (per operare a domanda-risposta), INDIU (indirizzo di RG), OPIU (= 0 lettura, = 1 incremento semaforo); quella con EU da RDYEU e RDYOUTEU, INDEU, OPEU (= 0 lettura, = 1 scrittura), VAL (valore da scrivere in RG[INDEU]).

Poiché IU è in-order, una richiesta di lettura viene sospesa se OR(C[INDIU]) = 1.

Poiché EU è di latenza unitaria, non c'è bisogno della sincronizzazione per leggere un registro generale. In caso di scrittura in RG[INDEU], viene anche controllato se WAIT[INDEU] = 1 e se il semaforo associato vale uno (funzione UNO(C[INDEU]) = 1; = 0 altrimenti): in tal caso il valore VAL viene anche inviato a IU.

Il microprogramma consta di una sola microistruzione:

0. (RDYIU, RDYEU, OPIU, OR(C[INDIU]), OPEU, WAIT[INDEU], UNO(C[INDEU]) = 0 0 – – – – –) nop, 0;

   (= 1 – 1 – – – –) reset RDYIU, C[INDIU] + 1 → C[INDIU], set RDYOUTIU, 0;

   (= 1 – 0 0 – – –) reset RDYIU, RG[INDIU] → OUTIU, set RDYOUTIU, 0;

   (= 1 – 0 1 – – –) reset RDYIU, 1 → WAIT[INDIU], 0;

   (= 0 1 – – 0 – –) reset RDYEU, RG[INDEU] → OUTEU, set RDYOUTEU, 0;

   (= 0 1 – – 1 0 –, 0 1 – – 1 1 0) reset RDYEU, VAL → RG[INDEU], C[INDEU] – 1 → C[INDEU], set RDYOUTEU, 0;

   (= 0 1 – – 1 1 1) reset RDYEU, VAL → RG[INDEU], C[INDEU] – 1 → C[INDEU], set RDYOUTEU, VAL → OUTIU, set RDYOUTIU, 0 → WAIT[INDEU], 0

L'unità è realizzata come una singola rete sequenziale di Moore con funzione delle uscite identità. Gli indirizzi (INDIU, INDEU) delle tre memorie passano da un commutatore, ma ovviamente questo non pone problemi di correttezza in quanto la variabile di controllo non può che essere funzione del solo stato interno.

Il ciclo di clock è dato da:

$$\tau = T_\sigma + t_p$$

Il massimo ritardo di stabilizzazione della funzione di transizione dello stato interno vale:

$$T_\sigma = T_{\alpha_K} + T_{K_M} + T_a + T_{ALU}$$

dove:

- T_{α_K} è il ritardo della variabile di controllo dei commutatori degli indirizzi delle memorie:

$$\alpha_K = \overline{RDYIN}\ \ RDYEU$$

- $T_{K_M} = 2t_p$ il ritardo di tali commutatori,
- T_a il tempo di accesso delle memorie (valutato per la lettura; le scritture sono sovrapposte alle letture nelle operazioni che leggono e scrivono in C, oppure hanno comunque ritardo inferiore a quello indicato per T_σ).

Tutte le altre variabili di controllo (β per le interfacce e per le memorie, α della ALU, funzioni OR e UNO) si stabilizzano in parallelo agli altri ritardi.

Quindi:

$$\tau = T_a + T_{ALU} + 4t_p$$

RGU opera come servente nei confronti dei clienti IU, EU in una struttura cliente-servente a domanda-risposta.

Il tempo di servizio ideale di RGU (tempo di servizio di RGU considerata in isolamento) è:

$$T_{RGU} = \tau$$

La latenza della comunicazione non ha impatto in quanto si opera a domanda-risposta e la latenza di trasmissione (inter-chip) è nulla: ad ogni ciclo di clock RGU è in grado di servire una eventuale nuova richiesta e di inviare la relativa risposta.

Il tempo di risposta del servente al generico cliente è in generale dato da:

$$R_Q = W_Q + L_S$$

Il *massimo* tempo $W_{Q-EU-max}$ di attesa in coda di una richiesta di EU è uguale a un ciclo di clock: ciò si verifica nel caso che sia presente anche una richiesta di IU. La latenza L_S di RGU è uguale a un ciclo di clock. Quindi:

$$R_{Q-EU-max} = 2\,\tau$$

Domanda 2

I dati del programma (array A) godono solo della proprietà di località, per cui ogni σ iterazioni si opera su un nuovo blocco di cache. Non ci sono scritture in memoria all'interno del loop. Applicando il prefetching del blocco successivo rispetto a quello corrente, il blocco necessario è già presente in cache, e quindi la degradazione di efficienza è trascurabile, se:

$$T_{trasf} \leq \sigma\, T_F$$

Supponendo di avere una struttura di interconnessione ad albero binario:

$$T_{trasf} = 2\,T_{tr} + \frac{\sigma}{m}\tau_M + \log_2 m\ \ (\tau + T_{tr})$$

Per una struttura costituita da *m* collegamenti diretti, il terzo addendo è $m\tau$. Per il trasferimento di blocchi la banda della memoria vale $B_M = \frac{m}{\tau_M}$. Quindi la condizione richiesta è:

$$B_M \geq \frac{\sigma}{\sigma\, T_F - 2\,T_{tr} - \log_2 m\ \ (\tau + T_{tr})}$$

Domanda 3

Il codice non ottimizzato è (i valori di *b* e *d* sono trasferiti in registri prima di entrare nel loop):

1. LOAD RA, Ri, Ra
2. LOAD RC, Ri, Rc
3. MUL Ra, Rb, Ra
4. DIV Rc, Rd, Rc
5. ADD Ra, Rc, Ra
6. STORE RX, Ri, Ra
7. INCR Ri
8. IF < Ri, RN, 1

che, ad ogni iterazione, soffre della degradazione dovuta al salto (IF), di una dipendenza logica IU-EU di distanza 1 della 7 sulla 8, e di una dipendenza logica IU-EU di distanza 1 della 5 sulla 6. Sulla latenza della dipendenza 5-6 hanno ripercussione le dipendenza logiche EU-EU delle 3 e 4 sulla 5; di queste ultime, poiché 3 e 4 sono indipendenti, ha impatto solo la dipendenza EU-EU della 4 sulla 5, quindi solo la latenza della 4.

Eseguendo questo codice con IU out-of-order FIFO si ottiene un tempo di servizio in assenza di fault di cache (si veda la simulazione grafica):

$$T_{out} = \frac{11}{8}\, t$$

Ottimizzando staticamente il codice si ottiene:

1. LOAD RA, Ri, Ra
2. LOAD RC, Ri, Rc
3. MUL Ra, Rb, Ra
4. DIV Rc, Rd, Rc
5. INCR Ri
6. ADD Ra, Rc, Ra
7. IF < Ri, RN, 1, *delayed_branch*
8. STORE RX, Ri, Ra

Le dipendenze logiche IU-EU 5-7 e 6-8, di distanza $k = 2$ e probabilità $d_k = 1/8$, sono intrecciate: solo la seconda ha effetto essendo certamente caratterizzata da una latenza maggiore. Applicando il modello dei costi:

- per quanto riguarda il solo impatto della 6 sulla 8 (Δ_1): la latenza indotta dalle LOAD appartenenti alla sequenza critica vale $N_{Qk} = 2$, mentre la latenza addizionale della EU nella 6 vale $L_{pipe\text{-}k} = 0$;
- per quanto riguarda l'impatto della sequenza critica (Δ_2), deve essere valutata la dipendenza logica EU-EU della 4 sulla 6, di distanza $h = 2$ e probabilità $d_h = 1/8$, con latenza addizionale della EU data da $L_{pipe\text{-}h} = 4$.

Quindi:

$$\Delta = \Delta_1 + \Delta_2 = t\, d_k \left(L_{pipe-k} + N_{Qk} + 1 - k\right) + t\, d_h \left(L_{pipe-h} + 1 - h\right) = \frac{4}{8}\, t$$

Avendo applicato delayed branch alla IF si ha $\lambda = 0$, per cui:

$$T_{in} = t + \Delta = \frac{12}{8}\, t$$

I tempi di completamento, in assenza di fault di cache, delle versioni out-of-order e in-order valgono rispettivamente:

$$T_{c-out} = 11\, N\, t$$

$$T_{c-in} = 12\, N\, t$$