Architettura degli Elaboratori

a.a. 2013/14 - primo appello, 10 giugno 2014

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

Domanda 1

Per un'architettura di CPU D-RISC, pipeline, scalare e FIFO out-of-order, dimostrare che il tempo di calcolo interno dell'unità EU_Master è uguale a un ciclo di clock. La dimostrazione deve essere data descrivendo il funzionamento dell'unità a un livello di dettaglio opportuno.

Domanda 2

La seguente computazione:

int A[M], float B[M];

$\forall$ *i = 0 .. M – 1 :*

B[i] = sqrt (A[i]) + 1

con M = 128K, è eseguita da una CPU D-RISC, pipeline, scalare, con ciclo di clock τ.

L'operazione di radice quadrata è delegata a un co-processore, realizzato come unità di I/O, avente tempo di calcolo interno uguale a 20τ. L'interconnessione tra CPU e unità di I/O è equivalente a due collegamenti dedicati (uno per direzione) con latenza di trasmissione uguale a 2τ. Di tale unità, e della sua cooperazione con la CPU, devono essere descritte le caratteristiche necessarie a comprenderne l'utilizzo che ne viene fatto nel programma compilato.

La gerarchia di memoria è così caratterizzata:

- cache dati primaria e cache secondaria on-chip sono entrambe su domanda, associative e Write-Through, con capacità e ampiezza di blocco rispettivamente uguali a (32K, 8) e (512K, 128);
- memoria principale interallacciata con 8 moduli, ciclo di clock uguale a 20τ, struttura di interconnessione ad albero binario, latenza di trasmissione dei collegamenti uguale a 2τ.

Determinare il tempo di completamento del programma in funzione di M e τ, fornendo adeguate spiegazioni.

Domanda 3

Per una CPU D-RISC, pipeline e scalare esprimere il tempo medio di attesa, nella coda tra IU ed EU, di istruzioni che possono indurre dipendenze logiche IU-EU. Considerare i due casi: EU di latenza unitaria, EU parallela.

Soluzione

Domanda 1

L'unità incapsula la copia principale dei registri generali RG e dei registri float RF, con relativi semafori associati. Ha interfacce d'ingresso da IU, da CD, e dalle unità funzionali (FU), e interfacce di uscita verso le FU e verso IU.

Per il funzionamento FIFO out-of-order, dispone di una memoria di registri (MW) in cui salvare le istruzioni bloccate da dipendenze EU-EU. Ogni locazione di MX contiene anche il numero (CONT) di aggiornamenti attesi. Ogni semaforo ha associato un indirizzo (IND_MX) di tale memoria e un bit di significatività (S).

Il funzionamento complessivo fa uso di non-determinismo nelle comunicazioni, per rilevare vari tipi di eventi, e di parallelismo tra eventi compatibili. Il microprogramma è descritto da una sola microistruzione, quindi l'unità è realizzata come singola rete sequenziale. Ad ogni evento, o sottoinsieme di eventi compatibili, corrisponde una rete combinatoria facente parte della funzione di transizione dello stato interno delle rete sequenziale.

Le variabili di condizionamento, testate tutte contemporaneamente, sono:

- tutti i RDY delle interfacce di ingresso e ACK delle interfacce di uscita,
- 8 bit di codice operativo,
- OR del contenuto di due semafori riferiti parametricamente con gli indirizzi dei registri sorgente,
- condizione che il semaforo associato ad un registro, aggiornato da una FU, diviene zero in seguito all'aggiornamento,
- bit di significatività in tale semaforo,
- OR del contatore CONT riferito parametricamente.

Le singole azioni sono:

1. ricezione di istruzione di Load: attesa del valore da CD e scrittura nel registro destinazione, con incremento del semaforo associato, con inoltro a IU;
2. ricezione di istruzione aritmetica corta su interi: esecuzione, con incremento del semaforo associato al registro destinazione, e inoltro del risultato a IU, oppure messa in attesa (uso di MW, CONT, e aggiornamento di IND_MW e S nel semaforo/nei semafori dei registri sorgente);
3. ricezione di istruzione aritmetica lunga o float: inoltro alla FU relativa dei parametri e del tipo di operazione, con incremento del semaforo associato al registro destinazione, oppure messa in attesa (uso di MW, CONT, e aggiornamento di IND_MW e S nel semaforo/nei semafori dei registri sorgente);
4. ricezione di una coppia (indirizzo registro, valore) dal sottosistema delle FU: aggiornamento del registro e del semaforo associato; se precedentemente il semaforo valeva uno, verifica di CONT di una eventuale istruzione bloccata su tale registro, e conseguente eventuale sblocco.

Insieme a ognuno dei casi 1, 2, 3 è possibile che si verifichi anche il caso 4, che viene servito in parallelo nella stessa microistruzione. L'aggiornamento del registro può sbloccare un'istruzione che si è precedentemente bloccata, la quale viene quindi eseguita nello stesso ciclo di clock insieme all'istruzione ricevuta da IU. Persino l'istruzione, appena ricevuta da IU, potrebbe risultare eseguibile anche se uno dei semafori associati ai registri sorgente vale uno.

Domanda 2

Il co-processore (I/O) funziona con Memory-Mapped I/O. L'interfaccia della CPU con l'unità I/O è nell'unità MMU_D. Ricevendo una richiesta di scrittura (generata da una STORE) I/O utilizza il dato per iniziare, nello stesso ciclo di clock, il calcolo della radice quadrata. La richiesta di lettura (generata da una LOAD) verrà ascoltata da I/O nell'ultimo ciclo di clock di tale esecuzione. Questo garantisce tutto quanto che serve per la cooperazione e la sincronizzazione di CPU e I/O.

L'utilizzo dell'istruzione WAITINT è equivalente ad usare una LOAD (una delle parole del messaggio di interruzione è il risultato), salvo che la latenza risulta maggiore.

Con RI/O si indica un registro generale contenente l'indirizzo logico corrispondente all'indirizzo fisico base della memoria locale dell'unità I/O. La compilazione non ottimizzata è la seguente:

```
LOOP:   LOAD   RA, Ri, Ra
        STORE  RI/O, 0, Ra          // genera richiesta di scrittura //
        LOAD   RI/O, 1, Rb          // genera richiesta di lettura; il dato ricevuto è inviato normalmente a EU //
        INCR   Rb
        STORE  RB, Ri, Rb
        INCR   Ri
        IF < Ri, RM, LOOP
```

Possibile ottimizzazione:

```
LOOP:   1.  LOAD   RA, Ri, Ra
        2.  STORE  RI/O, 0, Ra
        3.  INCR   Ri
        4.  LOAD   RI/O, 1, Rb
        5.  INCR   Rb
        6.  IF < Ri, RM, LOOP, delayed_branch
        7.  STORE  RB, Ri, Rb       // base B decrementata di uno //
```

Dipendenze logiche IU-EU: 1 su 2 (k =1, d_k = 1/7, N_Q = 2), 3 su 6 (non ha effetto), 5 su 7 (k = 2, d_k = 1/7, N_Q = 2). Quest'ultima è condizionata dalla latenza che intercorre tra l'esecuzione della 2 e il completamento della 4; tale latenza è uguale a quella dell'unità I/O più due latenze di trasmissione tra CPU e I/O, complessivamente 24τ. L'effetto della 4 può essere modellato come una dipendenza EU-EU della 2 sulla 4, di distanza h = 2, probabilità d_h = 1/7 e $L_{pipe\text{-}h} = 24\tau = 12t$:

$$\Delta = \Delta_1 + \Delta_2 = \left(\frac{2}{7}t + \frac{1}{7}t\right) + \frac{11}{7}t = \frac{14}{7}t$$

$$T = t + \Delta = \frac{21}{7}t$$

Il tempo di completamento in assenza di fault di cache è quindi:

$$T_{c-0} = 21\ M\ t = 42\ M\ \tau$$

Per quanto riguarda l'effetto della gerarchia di memoria, A e B godono solo di località; B è in solo scrittura e non provoca trasferimento di blocchi. Il meccanismo Write-Through non provoca degradazioni addizionali, essendo il tempo di servizio della memoria interallacciata, per uno stream di scritture a indirizzi consecutivi (= $\tau_M/m = 20\tau/8$), molto minore del tempo di interarrivo anche in assenza di fault di cache (= $T_{c\text{-}0}/M = 42\tau$).

In presenza di una sola struttura dati (A) soggetta a trasferimento blocchi, è ottimizzato l'uso di C2 su domanda: la lettura di *q-1* blocchi-C1 ($q = \sigma_2/\sigma_1 = 16$) da M a C2 è sovrapposto al calcolo e al trasferimento dei q blocchi-C1 da C2 a C1 (verificare la condizione), cioè la penalità per il trasferimento in C2 riguarda solo un blocco-C1. Quindi:

$$N_{fault-C2} = \frac{M}{\sigma_2} \quad , \quad N_{fault-C1} = \frac{M}{\sigma_1}$$

$$T_{trasfM-C2} = 2T_{tr} + \tau_M + lg_2 m\,(\tau + T_{tr}) = 33\tau$$

$$T_{trasfC2-C1} = 2\,\sigma_1\,\tau$$

$$T_{fault} = N_{fault-C2}\,T_{trasfM-C2} + N_{fault-C1}\,T_{trasfC2-C1} = \frac{33}{128}\,M\tau + 2\,M\tau \sim 2.3\,M\tau$$

In conclusione:

$$T_c = T_{c-0} + T_{fault} = 44.3\,M\tau$$

Domanda 3

Si tratta di esprimere la componente (W_Q) del tempo di risposta del servente (R_Q) dipendente dal tempo di servizio ideale di EU, ma non dalla latenza (L_s):

$$R_Q = W_Q(\rho) + L_s$$

In qualunque realizzazione di EU, il suo tempo di servizio ideale è uguale a *t*. Infatti, la realizzazione parallelo-pipeline ha proprio come obiettivo quello di mantenere costante il tempo di servizio, rispetto alla realizzazione con latenza unitaria, pur a scapito della latenza.

Per una macchina Risc, per ogni dipendenza logica di distanza *k* si ha:

$$W_Q(k) = \left(N_{Q_k} - k\right) t$$

dove N_{Q_k} esprime il numero di istruzioni nel sistema {coda IU-EU, EU}: tale parametro vale 2 oppure 1 a seconda che nella sequenza critica sia presente o meno un'istruzione di LOAD (dimostrazione per enumerazione in soli 4 casi).

Complessivamente la media sul programma è quindi data da:

$$W_Q = \sum_{k=1}^{\bar{k}} d_k\; W_Q(k) = t \sum_{k=1}^{\bar{k}} d_k\; \left(N_{Q_k} - k\right)$$