Architettura degli Elaboratori

Appello del 17 gennaio 2013

Si prega di consegnare l'elaborato in modo leggibile.

Domanda 1 (tutti)

Una unità di elaborazione U gestisce una coda FIFO di *N* parole (*N* potenza di 2) nei confronti di una unità produttore U1 e di una unità consumatore U2. U esegue una sola operazione possibile (inserzione o estrazione) alla volta con priorità al produttore. Le interfacce con U1 e U2 sono di una parola. Ogni messaggio inviato da U1/richiesto da U2 è caratterizzato da una propria dimensione *L* in parole, con $L < N$.

Scrivere e spiegare il microprogramma con il vincolo di minimizzare il numero medio di cicli di clock.

Domanda 2 (tutti)

a) Dire se le seguenti affermazioni sono vere o false, spiegando la risposta:

1. in un sistema contenente una gerarchia di memoria complessivamente costituita dai livelli M_n, M_{n-1}, …, M_1, l'indirizzo generato dall'interprete firmware del processo in esecuzione è riferito al livello M_n e viene via via tradotto, nell'ordine, nell'indirizzo di M_{n-1}, di M_{n-2}, …, di M_1. Nella risposta utilizzare anche esempi di livelli tipici;
2. in qualunque sottogerarchia memoria principale – cache, la cui definizione preveda che un blocco di memoria principale possa essere allocato solo in un determinato blocco di cache, l'indirizzo di cache è costituito da una stringa di bit consecutivi all'interno dell'indirizzo di memoria principale;
3. il microprogramma di una unità cache completamente associativa controlla che il blocco di memoria principale riferito sia attualmente allocato nel blocco di cache il cui identificatore è costituito da una stringa di bit consecutivi all'interno dell'identificatore del blocco di memoria principale.

b) Per una unità cache associativa su insiemi con otto blocchi per insieme, implementare la funzione di transizione dello stato interno della Parte Operativa relativa al registro destinato a contenere l'indirizzo di cache, e valutarne il ritardo di stabilizzazione.

Domanda 3 (NEW, OLD-0)

Determinare le dipendenze logiche, e valutarne l'impatto sul tempo di completamento, per il seguente codice da eseguire su una architettura pipeline scalare con EU avente unità funzionali in pipeline con 4 stadi:

```
LOAD   RX, Rj, Rx
MUL    Rx, Ry, Rx
DIV    Rx, Rz, Ri
INCR   Ri
LOAD   RA, Ri, Rc
STORE  RB, Rx, Rd
```

Domanda 3 (OLD-1)

In un sistema D-RISC con Memory Mapped I/O, il processore delega l'esecuzione di una funzione *F* o di una funzione *G* ad una unità P-I/O. Le funzioni operano su un intero fisicamente allocato nella memoria locale di P-I/O, il cui indirizzo logico è passato dal processore, e modificano la locazione stessa, senza effettuare ulteriori accessi in memoria. Il processore non attende il risultato. Non è disponibile il supporto alla comunicazione tra processi. Spiegare

a) come il processore comunica la richiesta a P-I/O,

b) come P-I/O può essere implementata senza effettuare ulteriori accessi in memoria oltre ai due richiesti per l'esecuzione di *F* o *G*, distinguendo due tipi di implementazione:

1. una singola unità firmware dedicata,
2. un elaboratore general-purpose.

Soluzione

Domanda 1 (tutti)

La prima parola di una richiesta di inserzione contiene il valore di *L*; dopo questa parola U1 invia le *L* parole del messaggio. La richiesta di estrazione contiene il valore di *L*; successivamente, se la richiesta è soddisfatta, U invia *L* parole a U2

Il nondeterminismo è controllato anche in funzione del numero di posizioni disponibili (registro VUOTE, inizializzato a N) e occupate (registro PIENE, inizializzato a 0). Nella prima microistruzione si usano, quindi, le variabili di condizionamento *segno (IN1 – VUOTE)* e *segno (IN2 – PIENE).*

L'inserzione e estrazione sono eseguite come loop di *L* passi di comunicazione, ognuno controllato da una microistruzione: questo garantisce il minimo numero di cicli di clock con il vincolo che le interfacce sono su singola parola, indipendentemente dal tempo di elaborazione di U1 e U2.

La coda è implementata da un vettore Q[N] gestito circolarmente con puntatori di inserzione INS ed estrazione ESTR di $log_2 N$ bit, inizializzati a zero.

0. (RDY1, RDY2, segno (IN1 – VUOTE), segno (IN2 – PIENE) = 0 0 – –, 1 0 1 –, 0 1 – 1) nop, 0;

   (= 1 – 0 –) //accettazione inserzione //

   reset RDY1, set ACK1, IN1 – 1→ I, VUOTE – IN1 → VUOTE, PIENE + IN1 → PIENE, 1;

   (= 0 1 – 0, 1 1 1 0) // accettazione estrazione //

   reset RDY2, set RDYOUT2, IN2 – 1 → I, VUOTE + IN2 → VUOTE, PIENE – IN2 → PIENE, 2;

1. (I_0, RDY1 = 0 0) nop, 1; (= 1 –) "nop", 0;

   (= 0 1) reset RDY1, set ACK1, IN1 → Q[INS], INS + 1 → INS, I – 1 → I, 1

2. (I_0, RDY2 = 0 0) nop, 1; (= 1 –) "nop", 0;

   (= 0 1) reset RDY2, set RDYOUT2, Q[ESTR] → OUT, ESTR + 1 → ESTR, I – 1 → I, 2

Domanda 2 (tutti)

a1: La situazione tipica è la gerarchia MV, M, C3, C2, C1. È vero che l'indirizzo generato dall'interprete firmware del processo in esecuzione è riferito al livello M_n (indirizzo logico), ma è in generale falso che esso venga via via tradotto, nell'ordine, nell'indirizzo di M_{n-1}, di M_{n-2}, …, di M_1. Nell'esempio, una volta tradotto l'indirizzo di MV in indirizzo di M, quest'ultimo viene tradotto in indirizzo di C1, ed eventualmente lo stesso indirizzo di M potrà venire tradotto in indirizzo di C2 (fault in C1) o di C3 (fault in C2), in quanto C3, C2, C1 sono una ulteriore gerarchizzazione di uno stesso livello (concettualmente esiste un livello di cache).

a2. Falso in generale. Nel metodo diretto quanto affermato è vero solo utilizzando, nella definizione della legge di corrispondenza, una funzione opportuna ed efficiente da implementare (modulo il numero di blocchi di C1), ma in generale qualunque altra funzione definibile staticamente va bene.

a3. Falso. L'identificatore del blocco di cache non può essere contenuto nell'identificatore del blocco di memoria principale, in quanto la funzione di corrispondenza è stabilita dinamicamente ad ogni allocazione/deallocazione di blocco.

b. Detto IND il registro contente l'indirizzo di memoria e INDC quello per indirizzare la memoria cache, la funzione richiesta è definita come segue:

in_{INDC} = *when* $\beta_{INDC} = 1$ *do*

IND.SET ° f (IND.TAG, TAB(IND.SET)) ° IND.DISPL

dove la funzione *f* è composta dal confronto di IND.TAG con gli 8 tag nella posizione IND.SET della memoria TAB e dalla funzione *g* trasforma gli 8 risultati del confronto nei 3 bit che identificano il blocco di cache all'interno dell'insieme. La funzione *g* è così definita:

> detti $x_0, \ldots, x_7$ i risultati negati del confronto e z_0, z_1, z_2 i bit dell'identificatore del blocco all'interno dell'insieme, per la tabella di verità è sufficiente considerare specificato lo stato di uscita corrispondente agli 8 stati d'ingresso aventi un solo bit a 1 e tutti gli altri bit a 0. Da questa si ricava l'espressione logica per z_0, z_1, z_2, ognuno dei quali è costituito dall'OR di 4 termini AND facilmente riconoscibili, ognuno con 8 variabili.

La funzione *f* si stabilizza con un ritardo dato da:

- accesso a TAB: t_{TAB}
- ex-OR su parola per i confronti: $2t_p$
- OR bit a bit di una parola risultato del confronto: $2t_p$
- rete combinatoria che implementa la funzione *g*: $2t_p$

assumendo come di regola che il ritardo t_p sia significativo per porte con al massimo 8 ingressi.

Quindi il ritardo richiesto vale

$$T_{\sigma PO/INDC} = t_{TAB} + 6t_p$$

Domanda 3 (NEW, OLD-0)

1. LOAD RX, Rj, Rx
2. MUL Rx, Ry, Rx
3. DIV Rx, Rz, Ri
4. INCR Ri
5. LOAD RA, Ri, Rc
6. STORE RB, Rx, Rd

Si tratta di valutare il massimo valore del ritardo Δ tra quello Δ_a della dipendenza indotta dalla 2 sulla 6 e Δ_b della dipendenza indotta dalla 4 sulla 5, tenendo conto della latenza della EU e del parallelismo/sequanzializzazione tra operazioni nelle sequenze critiche.

Applicando il modello dei costi si ha:

- dipendenza IU-EU di 2 su 6: distanza k = 4, probabilità $d_k = 1/6$, $L_{Qk} = 2$, $L_{pipe\text{-}k} = 4$, nessuna istruzione lunga nella sequenza critica, da cui $\Delta_1 = 3t/6$, $\Delta_2 = 0$, $\Delta_a = 3t/6$;
- dipendenza IU-EU di 4 su 5: distanza k = 1, probabilità $d_k = 1/6$, $L_{Qk} = 2$, $L_{pipe\text{-}k} = 0$, da cui $\Delta_1 = 2t/6$. Inoltre c'è impatto delle dipendenze EU-EU della 2 sulla 3 e della 3 sulla 4, entrambe di distanza h = 1, probabilità $d_h = 1/6$ e $L_{pipe\text{-}h} = 4$, quindi $\Delta_2 = 8t/6$ e $\Delta_b = \Delta_1 + \Delta_2 = 10t/6$.

In conclusione,

$$\Delta = \Delta_b = 10t/6$$

Così valutato si tratta del parametro usato per determinare il tempo di servizio per istruzione. Agli effetti del *tempo di completamento*, il ritardo che viene pagato (bolla complessiva) è uguale a

$$10t = 20\tau$$

Si noti che, applicando il funzionamento out-of-order, è possibile ridurre tale ritardo a $9t$.

Domanda 3 (OLD-1)

a) La richiesta (*fun, ind*) viene comunicata semplicemente come:

STORE Ri/o, 0, Rfun

STORE Ri/o, 1, Rind

dove RG[Rfun] contiene l'identificatore della funzione *F/G*, e RG[Rind] l'indirizzo logico del parametro di ingresso.

b) Per soddisfare il vincolo richiesto, P-I/O non effettua le scritture in memoria locale dei due dati ricevuti (*fun, ind*) dal Bus di I/O, bensì opera direttamente su tali valori, ignorando l'indirizzo fisico contenuto nella richiesta.

1. Se P-I/O è una unità specializzata a firmware, con memoria locale nella propria PO, i due valori (*fun, ind*) sono scritti in registri e usati normalmente nel microprogramma di *F/G*.
2. Se P-I/O è un elaboratore general-purpose, con propria CPU (CPU-I/O) e propria memoria locale, esso dispone anche di una propria unità di I/O: questa è, analogamente al caso *a)*, una unità specializzata a firmware connessa al Bus di I/O, la quale riceve i due valori (*fun, ind*) senza scriverli in memoria locale e li comunica con una interruzione a CPU-I/O. Lo handler dell'interruzione esegue la funzione *F/G*.