Architettura degli Elaboratori

Seconda Prova di Verifica Intermedia - 28 maggio 2014

Riportare tu tutti i fogli, nome, cognome, numero di matricola e corso (A/B). I risultati verranno pubblicati sulle pagine web dei docenti. Le date degli orali saranno pubblicati sulle pagine web dei docenti contestualmente alla pubblicazione dei risultati.

Domanda 1

Si consideri la computazione

```
for(i=0; i<N; i++)
    A[i]=A[i]*C[B[i]%N];
```

con A, B e C vettori di numeri interi di lunghezza N. Per tale computazione: si determini l'insieme di lavoro e il numero di fault, se ne fornisca il codice D-RISC e si calcoli, in funzione di N e τ, il tempo di complemento relativo ad una architettura pipeline con le seguenti caratteristiche:

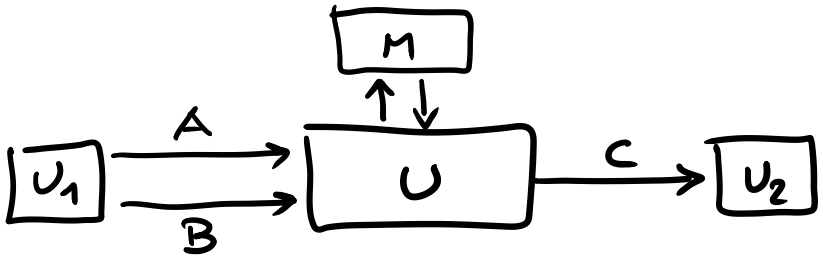
a) EU parallela con EU slave a 4 stadi che calcola moltiplicazioni, divisioni e modulo fra interi,
b) cache dati primaria, associativa su insiemi, con insiemi da 4 blocchi, σ=8, operante su domanda, write-through, capacità complessiva di 32K parole
c) cache secondaria on chip, memoria principale interallacciata con m=4 moduli e tempo di accesso di τ_M=30τ.

Si assuma che tutte le strutture dati necessarie si trovino già nella cache di secondo livello.

Domanda 2

Si consideri una unità firmware U collegata ad una memoria esterna M mediante un'interfaccia di memoria standard. La memoria M è da 4G parole ed ha un tempo di accesso t_M pari a 5 volte il ciclo di clock di U ($t_M = 5\tau$). L'unità riceve due parole A e B da 32 bit da una unità U1 ed invia ad U2 A, se M[A] > M[B], o B, se invece vale M[A]<=M[B]. Di tale unità si determini il tempo di servizio, considerando due casi:

- U1 è pronta all'invio di una nuova coppia <A,B> ogni 5τ
- oppure ogni 15τ.



Traccia di soluzione

Domanda 1

a) Gli accessi ai vettori A e B sono caratterizzati da località. L'accesso al vettore C è caratterizzato (parzialmente) da riuso. Dunque il working set del programma sarà costituito da un blocco di A, un blocco di B, dal codice e da tutto il vettore C.

b) Il numero di fault "fisiologici" sarà dunque al più 3N/8 in aggiunta ai (pochi) fault relativi al codice

c) Il codice in Assembler D-RISC compilato secondo le regole standard studiate nella prima parte del corso sarà:

```
        CLEAR Ri
loop:   LOAD RbaseA, Ri, Rai
        LOAD RbaseB, Ri, Rbi
        MOD  Rbi, RN, Rbi
        LOAD RbaseC, Rbi, Rcbi
        MUL  Rcbi, Rai, Rai
        STORE RbaseA, Ri, Rai
        INC Ri
        IF< Ri, RN, loop
        END
```

La simulazione dell'esecuzione di questo codice sarà dunque:

	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3
IM	LOAD	LOAD	MOD	LOAD	MUL							STORE	INC							IF<	END		LOAD	
IU		LOAD	LOAD	MOD	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	LOAD	MUL	STORE	STORE	STORE	STORE	STORE	STORE	STORE	INC	IF<	IF<	END	LOAD
DM			LOAD	LOAD								LOAD								STORE				
EU m				LOAD	LOAD	MOD							LOAD	MUL										
EU slave							MOD	MOD	MOD	MOD					MUL	MUL	MUL	MUL			INC			

per un tempo di completamento di 22Nt contro un tempo di completamento ideale pari a 8Nt (efficienza 8/22 cioè circa 33%). Il codice si può ottimizzare cercando di ridurre l'effetto della dipendenza MOD → LOAD e MUL → STORE. Possiamo spostare la prima LOAD dopo la MOD e, utilizzando nella store un RbaseA' decrementato di 1, spostare la STORE nel delay slot della IF< trasformata in una IF< …, delayed. Otteniamo dunque il codice:

```
        CLEAR Ri
loop:   LOAD RbaseB, Ri, Rbi
        MOD  Rbi, RN, Rbi
        LOAD RbaseA, Ri, Rai
        INC Ri
        LOAD RbaseC, Rbi, Rcbi
        MUL  Rcbi, Rai, Rai
        IF< Ri, RN, loop, delayed
        STORE RbaseA', Ri, Rai
        END
```

La simulazione dell'esecuzione di questo codice sarà dunque:

	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8
IM	LOAD	MOD	LOAD	INC	LOAD	MUL					IF<	STORE	LOAD						
IU		LOAD	MOD	LOAD	INC	LOAD	LOAD	LOAD	LOAD	LOAD	MUL	IF<	STORE	STORE	STORE	STORE	STORE	STORE	LOAD
DM			LOAD		LOAD						LOAD								STORE
EU m				LOAD	MOD	LOAD	INC					LOAD	MUL						
EU slave						MOD	MOD	MOD	MOD					MUL	MUL	MUL	MUL		

Dunque il tempo di completamento diventerà 17Nt cioè 34Nτ.

A questo tempo di completamento dobbiamo sommare l'overhead dovuto ai fault nella cache primaria. Se assumiamo N tale per cui C possa essere mantenuto interamente nella cache di primo livello, dobbiamo calcolare questo overhead come il numero di fault precedentemente calcolato moltiplicato per il tempo di trattamento del singolo fault. Con le solite assunzioni, il tempo di trattamento di un fault C1 risolto in C2 con C2 sullo stesso chip di C1 può essere approssimato con $2\sigma\tau=16\tau$, e pertanto l'overhead sarà valutabile in $3N*2\sigma\tau/\sigma = 6N\tau$

Dunque il tempo di completamento del programma (seconda versione) potrà essere approssimato (non teniamo conto dei fault relativi al codice né delle istruzioni non appartenenti al corpo del loop, sarà di

$$34N\tau + 6N\tau = 40N\tau$$

Questo significa anche che, per effetto del write through, verranno dirette alla memoria principale richieste di scrittura ogni 40τ (una store per ciclo). La memoria impiega 30τ per effettuare una scrittura e dunque non ci sono problemi legati alla banda della memoria principale ed all'utilizzo della politica write through.

Domanda 2

L'ipotetico microprogramma dell'unità utilizzerà

- una istruzione per ricevere A e B ed utilizzare A come INDirizzo di una OP="read" verso M
- una istruzione per ricevere M[A], salvarlo in un registro temporaneo ed inviare una seconda operazione con INDirizzo B e OP="read"
- una istruzione per ricevere M[B], testare segno(TEMP – DATAIN) e inviare A o B ad U2

e dunque l'unità impiegherà $3\tau+2t_a$ per servire una richiesta. Di conseguenza, il suo tempo di servizio ideale, definito come $\max\{T_a, T_s\}$ sarà pari a 13τ e 15τ rispettivamente, dato che con un tempo di interarrivo di 5τ il massimo sarà dato dal termine relativo al tempo di servizio di U, mentre con un tempo di interarrivo di 15τ il massimo sarà dato da T_a.