Architettura degli Elaboratori—A.A. 2016-201
Appello Straordinario—7 aprile 2017

Indicare su tutti i fogli consegnati Nome, Cognome, Matricola e Corsi di appartenenza. I risultati e il calendario degli orali saranno pubblicati su web appena disponibili

Domanda 1

Si consideri il seguente microcodice:

0. (RDYin=0) NOP, 0;
   (=1) I+1→I, 1
1. M[IN] →X, IN→TEMP, 2
2. (segno(X-K),ACKout=0-) M[TEMP]+1 →M[TEMP], set ACKin, reset RDYin, 0
   (=11) X → OUTm, I→OUTi, set RDYout, reset ACKout, set ACKin, reset RDYin, 0
   (=10) nop, 2

che interfaccia una unità U a due unità U_1 (che spedisce dati ad U nel registro di interfaccia IN) ed U_2 (che riceve dati da U tramite i due registri OUT_m e OUT_i) e se ne valuti una possibile ottimizzazione. I è un registro interno inizializzato a 0, K è un registro interno.

Successivamente si discuta la complessità della parte controllo prima e dopo l'ottimizzazione.

Domanda 2

Un codice D-RISC implementa l'algoritmo espresso dallo pseudo-codice:

```
float a[N],b[i];
for(int i=1; i<N-1; i++)
  b[i]  = (a[i-1]+a[i]+a[i+1])/3 ;
```

e viene eseguito su un'architettura D-RISC pipeline con unità per la somma/sottrazione in virgola mobile da 2 stadi e per la moltiplicazione/divisione in virgola mobile da 4 stadi.

Si fornisca il working set del programma e il minimo numero di fault richiesti, indicando, se necessario, le assunzioni sulla cache.

Successivamente, si fornisca il tempo di servizio per l'esecuzione del programma.

Facoltativo: si consideri l'esecuzione dello stesso codice su un processore che in aggiunta alle caratteristiche precedenti supporti multithreading a 2 vie in modalità interleaving e si valuti nuovamente il tempo di servizio assumendo che un primo thread calcoli la prima metà dei valori dell'array B e un secondo thread calcoli la seconda metà dei valori di B.

Traccia di soluzione

Domanda 1

Dall'analisi del codice vediamo che:

- Le condizioni di Bernstein ci dicono che non vi sono dipendenze fra le operazioni della seconda frase delle 0. e quelle della 1., che quindi possono essere raggruppate in un'unica micro operazione
- Fra la 1. E la 2. C'è una dipendenza su X e TEMP, che però possono essere entrambe eliminate utilizzando M[IN] al posto di X e IN al posto di TEMP

Dal che si deduce che si sarebbe potuta utilizzare una unica micro istruzione:

1. (RDYin,segno(M[IN]-K,ACKout=0--) nop, 1
   (=10-)i+1 → i, M[IN] +1 →M[IN], set Ackin, reset RDYin, 1
   (=111) M[IN] →OUTm, I+1→OUTi, I+1→I, set ACKin, reset RDYin, set RDYout, reset ACKout, 1
   (=110) nop, 1

Il che riduce la complessità della PC da rete sequenziale a rete combinatoria (per la sola produzione delle variabili di controllo a partire dalle sole variabili di condizionamento).

Domanda 2

Il programma scorre in lettura l'array A e in scrittura l'array B, accedendo a locazioni successive. Il working set è quindi costituito da una linea di B e da una linea di A. Quando la posizione i+1-esima di A finisce nella linea successiva a quella corrente, la linea entra nel working set e quella corrente si appresta ad uscirne.

Il numero minimo di fault risulterebbe quindi pari al numero di fault per il codice (1 o poche unità a seconda della dimensione della linea di cache e del codice compilato per il ciclo) più N/σ fault per A e altrettanti per B. Essendo il vettore B scritto completamente, tranne la prima e l'ultima posizione, il fault relativo alla prima e all'ultima linea richiede un accesso al livello successivo della gerarchia di memoria (e costa dunque come un fault in lettura) mentre tutte le linee intermedie fra queste due richiedono la sola allocazione in cache di primo livello anziché l'allocazione + l'accesso in memoria per la lettura. Se supponessimo di avere a disposizione una sottosistema cache che supporta il prefetching, l'utilizzo della LOAD/STORE col flag PREFECTH permetterebbe di ridurre a 1 il numero di fault sia per A che per B.

Il codice compilato del programma è il seguente:

```
        ADD R0, #1, Ri
Loop:   ADD Ri, #-1, Rsotto
        LOAD RbaseA, Rsotto, R1
        LOAD RbaseA, Ri, R2
        ADDFP R1, R2, R3
```

ADD Ri, #1, Rsopra
LOAD RbaseA, Rsopra, R4
ADDFP R3, R4, R5
DIVFP R5, #3, R6
STORE RbaseB, Ri, R6
INC Ri
SUB RN, #1, Rnmeno1
IF< Ri, RN, Loop
END

Si può notare che c'è un invariante (calcolo che non dipende dalla variabili di iterazione i) nel corpo del loop, che può essere pertanto rimosso. E' il caso del calcolo di N-1 alla fine del ciclo. Questa istruzione può essere portata prima dell'inizio del ciclo.

La simulazione del codice (originale) porta a un tempo di completamento della singola iterazione di ca. 25t a fronte di 12t teorici (12 istruzioni nel corpo del ciclo). Il tempo così lungo è dovuto alle dipendenze indotte dal calcolo degli indici sugli accessi in memoria e dal calcolo del risultato sulla store. Entrambe possono essere mitigate con spostamenti di codice (calcolo degli indici prima dei relativi accessi, con gli altri accessi frapposti, spostamento della STORE nel delay slot del salto condizionale).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
IM	ADD	LD		LD	ADD	ADD	LD			ADD	DIV	ST									INC	SUB	IF<		END	ADD		
IU		ADD	~~LD~~	LD	LD	ADD	ADD	~~LD~~	~~LD~~	LD	ADD	DIV	~~ST~~	~~ST~~	~~ST~~	~~ST~~	~~ST~~	~~ST~~	~~ST~~	~~ST~~	ST	INC	SUB	~~IF<~~	IF<	~~END~~	ADD	
DM					LD	LD					LD											ST						
EU			ADD			LD	LD	ADD	ADD			LD	ADD	~~DIV~~	~~DIV~~	DIV							INC	SUB				ADD
EU+-FP									ADD	ADD				ADD	ADD													
EU*/FP																	DIV	DIV	DIV	DIV								

Parte facoltativa:

In questo caso le piccole bolle legate al calcolo degli indici per gli accessi al vettore A sono mitigate dall'interleaving, mentre la bolla legata al calcolo del nuovo valore di b[i] non viene ridotta significativamente. Da notare che il fatto di avere una unica copia delle EU floating point non è un problema, perché per sua natura l'interleaving sfrutta la struttura pipeline delle unità facendo partire comunque un'operazione ogni *t*. Di fatto risusciamo a completare una interazione di entrambi i thread in 31t, a fronte dei 25t per una sola iterazione di un solo thread precedente.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
IM	ADD	ADD	LD	LD	LD	LD	ADFP	ADFP	ADD	ADD	LD	LD	LD	ADFP	ADFP	DIVFI	DIVFI	ST							ST	INC	INC	SUB	SUB	IF<	IF<	ADD	ADD	
IU		ADD	ADD	LD	LD	LD	LD	ADFP	ADFP	ADD	ADD	~~LD~~	LD	LD	ADFP	ADFP	DIVFI	DIVFI	ST						ST	ST	INC	INC	SUB	SUB	IF<	IF<	ADD	ADD
DM					LD	LD	LD	LD						LD	LD											ST	ST							
EU			ADD	ADD		LD	LD	LD	LD	ADFP	ADFP	ADD	ADD		LD	LD	ADFP	ADFP	~~DIVFI~~	DIVFI	DIVFP							INC	INC	SUB	SUB			ADD
EU+-FP											ADFP	ADFP						ADFP	ADFP															
												ADFP	ADPF						ADFP	ADFP															
EU*/FP																					DIFVI	DIFVI	DIFVI	DIFVI	DIFVP									
																						DIFVI	DIFVI	DIFVI	DIFVP									