

Architettura degli elaboratori – A.A. 2016-17

Terzo appello—12 giugno 2017

Riportare nome, cognome, numero di matricola e corso di appartenenza in alto a destra su tutti i fogli consegnati.

I risultati saranno pubblicati via web appena disponibili

Domanda 1

Si consideri un programma che, dati N vettori V_1, \dots, V_N ciascuno di valori in virgola mobile e di N posizioni, calcola un vettore Max di N posizioni la cui i -esima posizione è la somma dei massimi dei vettori V_1, \dots, V_i . Gli indirizzi di partenza dei vettori V_1, \dots, V_N sono contenuti in un vettore IND di N posizioni. Lo pseudo codice potrebbe essere:

```
float runmax=0;
for(int v=0; v<N; v++) {
    float max = (IND[v])[0];
    for(int i=1; i<N; i++)
        if (max < (IND[v])[i]) (max=IND[v])[i];
    runmax += max;
    res[v] = runmax;
}
```

Si fornisca il codice D-RISC del programma e quindi, assumendo di lavorare in un sistema con gerarchia di memoria a 2 livelli (cache set associativa a 2 vie, $\sigma=8$, 1K insieme, on chip, memoria principale interallacciata con 4 moduli da 1M parole ciascuna, $\tau_M = 50\tau$, off chip, $t_{tr}=4\tau$):

- se ne valutino le prestazioni su un'architettura D-RISC pipeline con unità EU_{slave} che calcola somma e sottrazione tra numeri in virgola mobile in $2t$
- se ne fornisca il working set e il numero di fault di cache
- si ottimizzi il codice prodotto, stimando il guadagno ottenuto in termini di tempo di servizio

Domanda 2

Si consideri un sistema costituito da una unità firmware U connessa ad una gerarchia di memoria simile a quella dell'esercizio precedente. U riceve in ingresso messaggi (IND, DIM) , dove IND è un indirizzo di memoria, e DIM è un intero positivo, e restituisce in uscita alla stessa unità che ha inviato la richiesta il numero degli elementi dell'array avente indirizzo base IND e dimensione DIM che hanno la proprietà di essere dispari.

Si dettagliano le interfacce dell'unità U e se ne fornisca il tempo medio di elaborazione in funzione di t_p , ritardo di una porta logica con al più 8 ingressi.

Traccia di soluzione

Domanda 1

Il codice che risulta dalla compilazione secondo le regole standard è il seguente:

```

1          ADDF R0, R0, Rrunmax          // runmax = 0
2          ADD R0, R0, Rv                // v = 0
3  LOOPV:  LOAD Rinds, Rv, Rcurr         // recupero (IND[v])
4          LOAD Rcurr, R0, Rmax          // max = (IND[v])[0]
5          ADD R0, #1, Ri                 // ri = 1
6  LOOPI:  LOAD Rcurr, Ri, Rvi           // vi = (IND[v])[i]
7          IF>= Rmax, Rvi, CONT           // se è già il massimo, continua
8  THEN:   ADDF Rvi, R0, Rmax            // max = vi
9  CONT:   INC Ri                         // i++
10         IF< Ri, RN, LOOPI             // fine ciclo for i
11         ADDF Rmax, Rrunmax, Rrunmax    // runmax += max
12         STORE Rres, Rv, Rrunmax       // res[v] = runmax
13         INC Rv                          // v++
14         iF<Rv, RN, LOOPV              // fine ciclo for v
15         END                            // fine programma

```

Consideriamo la sola esecuzione a regime del ciclo più interno. In questo caso la simulazione avviene come segue:

- a) caso di iterazione che non trova un nuovo minimo.

Abbiamo 4 istruzioni, che richiedono $9t$ per completare, dunque con un'efficienza pari a $\varepsilon = 4/9 = 0.44$

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
IM	LD	IF			ADC	INC	IF			LD				
IU		LD	IF	IF	ADC	INC	IF	IF		LD				
DM			LD									LD		
EUm				LD				INC					LD	
EU+-														

- b) caso di iterazione che trova un nuovo minimo.

Abbiamo 5 istruzioni, che richiedono sempre $9t$ per completare, dunque un'efficienza pari a $\varepsilon = 5/9 = 0.55$, leggermente migliore.

	0	1	2	3	4	5	6	7	8	9	10	11	12
IM	LD	IF			ADC	INC	IF			LD			
IU		LD	IF	IF	ADC	INC	IF	IF		LD			
DM			LD									LD	
EUm				LD			ADC	INC					LD
EU+-							ADC	ADD					

Al di là dell'efficienza, entrambi i casi richiedono $9t$ per il completamento della singola iterazione.

Il working set del programma sarà costituito da:

- i. la pagina corrente di IND e RES (scorrimento sequenziale in sola lettura (IND) o scrittura (RES), strutture dati con località e senza riuso)
- ii. la pagina corrente del vettore Vi (scorrimento sequenziale in sola lettura, struttura dati con località e senza riuso)
- iii. il codice (accessi con località e riuso)
- iv. i valori scalari (max, runmax)

Il numero di fault “fisiologici” sarà dunque pari a $2N/\sigma + N*N/\sigma$ oltre ai fault per il codice (2 o 3, nel nostro caso, a seconda dell’allineamento in memoria).

Il codice del ciclo più interno si può ottimizzare osservando che la INC Ri può essere anticipata immediatamente dopo la LOAD Rcurr, Ri, Rvi, mitigando l’effetto della dipendenza indotta dalla LOAD sulla IF>= e annullando l’effetto della dipendenza indotta dalla INC Ri sulla IF<. Questa semplice operazione fa scendere a $7t$ il tempo necessario a concludere un’iterazione.

		0	1	2	3	4	5	6	7	8	9	10
IM		LD	INC	IF		ADC	IF		LD			
IU			LD	INC	IF	IF	ADC	IF		LD		
DM				LD							LD	
EUm					LD	INC		ADD				LD
EU+-									ADC	ADD		

A questo punto si potrebbe anche utilizzare un delayed branch nella IF< Ri, Rn, LOOPi, utilizzando una LOAD Rcurr, Ri, Rvi nel delay slot e saltando a LOOPi+1:

```

LOOPi:  LOAD Rcurr, Ri, Rvi           // vi = (IND[v])[i]
LOOPi'': INC Ri                       // i++
        IF>= Rmax, Rvi, CONT         // se è già il massimo, continua
THEN:   ADD Rvi, R0, Rmax            // max = vi
CONT:   IF< Ri, RN, LOOPi'', delayed // fine ciclo for i
        LOAD Rcurr, Ri, Rvi         // delay slot, prima istruzione dell'iterazione

```

Questo porta a $6t$ il tempo necessario per la singola iterazione, con una efficienza pari a 0.66, ancora migliore alle precedenti.

			0	1	2	3	4	5	6	7	8	9
IM		LD	INC	IF>=		ADC	IF<	LD	INC			
IU			LD	INC	IF>=	IF>=	ADC	IF<	LD	INC		
DM				LD						LD		
EUm					LD	INC		ADD			LD	INC
EU+-									ADC	ADD		

In termini di tempo di servizio, le ottimizzazioni mostrate portano l’iniziale $T = 9t/4$ a $T = 6t/4$.

Quanto detto fin’ora vale per il tempo ideale di completamento. Va aggiunto il tempo necessario trattare i fault. Ciascun fault costa

$$(\tau + 2T_{tr}) + \sigma(\tau_M)/m + 4\tau = 113 \tau$$

Il fault si verifica ogni 8 iterazioni, quindi il tempo medio di completamento delle iterazioni del ciclo più interno sarà pari a

$$(8 \cdot (6t) + 113\tau) / 8 = 12\tau + 113\tau / 8 = 12\tau + 14.12\tau = 16.12\tau$$

Domanda 2

U avrà in ingresso dall'unità cliente i registri IND, DIM e l'indicatore a transizione di livello RDYin ed in uscita verso la stessa unità il registro OUTC e l'indicatore a transizione di livello ACKin. Questa interfaccia verrà utilizzata dall'unità cliente utilizzando un protocollo a domanda risposta.

U avrà un'interfaccia verso il sottosistema di memoria con i registri in uscita INDMEM e OP, in ingresso ESITO e DATAIN, e gli indicatori a transizione di livello in uscita RDYm e in ingresso ACKm.

Il microcodice relativo alla unità in questione può essere scritto come segue:

- ```

0. (RDYin = 0) nop, 0 // nessuna richiesta, attendi
// richiesta di servizio, salva i registri di interfaccia in registri generali (per poterli riscrivere con ALU),
(=1) IND → RIND, DIM → RDIM,
// inizializza il contatore di elementi dispari
0 → C,
// ed effettua una richiesta al sottosistema di memoria per l'indirizzo corrente
IND → INDMEM, "read" → OP, set RDYm, 1
1. (ACKm, or(ESITO), zero(RDIM), DATAIN0=1---) nop, 1 // attesa dati da sottosistema di memoria
(=11--) ... , i // trattamento errore di lettura in memoria
(=101-) C → OUTC, reset RDYin, set ACKout, reset RDYin, set ACKin, 0 // termina ciclo e comunica
(=1000) RDIM-1 → RDIM, IND+1 → IND, 1 // dato pari, procedi
(=1001) RDIM-1 → RDIM, IND+1 → IND, C+1 → C, 1 // dato dispari, incr. e procedi

```

La parte controllo avrà tipicamente un tempo di stabilizzazione sia per la  $\omega$ PC che per la  $\sigma$ PC di  $2t_p$  (2 stati (1 bit) e 4 variabili di condizionamento (4 bit) fanno cinque (<8) bit di ingresso per il livello AND; 6 frasi fanno cinque (<8) ingressi (al più) per il livello OR). Per la  $\omega$ PO abbiamo uno zero(RDIM). Al massimo, la dimensione del vettore potrà essere 512M, dunque l'OR che calcola lo zero sarà su 29 bit (2 livelli di porte OR =>  $2t_p$ ). Per la  $\sigma$ PO abbiamo tutte scritte in registri con un commutatore in ingresso che sceglie fra un risultato della ALU (e.g. C+1) e una costante (e.g. 0) il valore da scrivere nel registro, impiegando un  $t_k + t_{alu}$ , ovvero un  $2t_p + 5t_p = 7t_p$ . Le ultime frasi della 1 utilizzano sia il calcolo di zero(RDIM) che la scrittura di un registro con il valore di una ALU tramite un commutatore, il che porta il ciclo minimo di clock a  $2t_p$  (per lo zero(...) a 29 bit) più  $2t_p + 7t_p$  (il  $T_{OPC} + t_k + t_{alu}$ ) più  $\delta$ , il che fa sì che  $\tau = 2t_p + 9t_p + t_p = 12t_p$ . A questo va aggiunto  $t_a$  (attesa nella 1. Che ACKm sia diverso da 0). In caso di cache hit, essendo la cache ad indirizzamento diretto, il  $t_a$  costa  $2\tau$  (tempo di accesso a una cache set associativa on-chip, in questo caso assumiamo che non ci sia traduzione indirizzi). In caso di fault paghiamo tutto il tempo di rimpiazzo della linea di cache dalla memoria principale, già calcolato per l'esercizio precedente.

L'esecuzione dell'unica operazione esterna richiederà dunque  $1\tau$  (per la 0.) più DIM ( $\tau + t_a$ ) (per la 1.)