Architettura degli Elaboratori – A.A. 2009-2010

2° appello – 23 giugno 2010

Riportare su tutti i fogli consegnati nome e cognome, numero di matricola, corso di appartenenza, programma d'esame (NEW, OLD-0, OLD-1).

I risultati saranno pubblicati sulle pagine web dei docenti, appena disponibili, insieme al calendario degli orali.

Domanda 1 (*tutti, eccetto la distinzione sul tipo di architettura della CPU : per* ***NEW*** *e* ***OLD-O*** *è una CPU pipeline con cache dati completamente associativa, per* ***OLD-1*** *una CPU convenzionale con cache completamente associativa***)**

Si compili in assembler D-RISC il seguente programma, che opera su due array *A*, *B* di *N* interi e su un intero *x,*con *i* inizializzato a zero:

```
while(x < MAX) {
  temp = A[i] + B[i];
  if(temp < 0)
    temp = -temp;
  x = x + temp;
  i++;
  if(i == (N-1))
    break; // break => esci dal ciclo immediatamente
}
```

Supponendo che la probabilità che *temp* sia minore di *0* sia trascurabile, che vengano eseguite *N* iterazioni, che *N* = 8K, e che le pagine di memoria virtuale abbiano ampiezza 4K parole:

a) valutare il tempo di completamento in assenza di fault di cache, in funzione di *N* e del ciclo di clock,

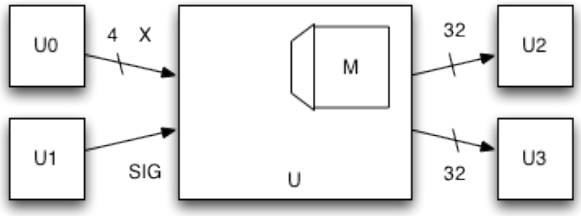
b) valutare il working set della gerarchia di memoria memoria principale - cache ed il numero di fault di cache,

c) mostrare una possibile traccia degli indirizzi *fisici* generati dal programma durante la prima iterazione.

Domanda 2 (*tutti***)**

Un'unità U è interfacciata ad una unità U0 che le fornisce un valore X su 4 bit, ad una unità U1, che le fornisce un segnale di pura sincronizzazione, e a due unità U2 ed U3 cui invia parole di 32 bit.

L'unità U contiene una memoria M con capacità 1K parole da 32 bit.



Per ogni X ricevuto da U0, U conta quante sono le celle di M, con indirizzo *i* tale che *i%16 = X,* che contengono un numero pari e quante quelle che contengono un numero dispari. I risultati (numero_pari e numero_dispari) vengono inviati rispettivamente ad U2 ed U3.

Ogni qualvolta viene rilevato il segnale SIG, il calcolo corrente viene immediatamente *abortito, senza inviare risultati in uscita* e, in seguito alla *prossima* ricezione di un nuovo X, l'invio dei risultati verso U2 ed U3 viene invertito: il numero_pari, precedentemente inviato ad U2 (U3), viene inviato ad U3 (U2), e il numero_dispari, precedentemente inviato ad U3 (U2),, viene inviato ad U2 (U3).

La memoria M ha un tempo di accesso pari a $10t_p$.con t_p ritardo di stabilizzazione di una porta logica con al massimo otto ingressi. Una ALU a 32 bit opera in $5t_p$ e gli eventuali overflow non vanno considerati.

Fornendo adeguate spiegazioni, è richiesto il microprogramma dell'unità U e la valutazione della sua banda di elaborazione per $t_p = 10^{-2}$ nsec, assumendo trascurabile la probabilità che si verifichi un segnale SIG.

Domanda 3

NEW, OLD-0

Si consideri il codice assembler della Domanda 1 e se ne valuti una versione ottimizzata, valutandone il tempo di completamento.

OLD-1

Il codice C della domanda 1 definisce la parte di puro calcolo di un processo Q, espresso in LC, nel quale tale parte di codice è preceduta dalla ricezione dei valori degli array A e B da un processo R. Q opera come un ciclo infinito, continuando a ricevere coppie di array da R, effettuando il calcolo, e così via.

Si assume che la parte di puro calcolo sia sempre conclusa prima che R invii i prossimi valori di A e B.

Valutare di quanto si modifica il tempo di servizio di Q rispetto al valore del tempo di completamento valutato nella Domanda 1.

Per questo insegnamento sono previsti, durante l'intero A.A., 6 appelli ed un numero massimo di 5 prove.

Gli studenti del vecchio ordinamento devono scegliere se presentarsi sul programma di esame 2009-10 con l'aggiunta del Cap. VII (Processi), sez. 5, 6 e 7, oppure sul programma di esame 2008-09. La scelta va fatta la prima volta che lo studente si iscrive all'esame, e vale per tutti gli eventuali successivi appelli dell'a.a. ai quali lo studente si presenti.

*Riportare su tutti i fogli consegnati nome, cognome, numero di matricola, corso di appartenenza, e la sigla **NEW** (per nuovo ordinamento), oppure **OLD-0** (per vecchio ordinamento, programma 2009-10), oppure **OLD-1** (per vecchio ordinamento, programma 2008-09).*

Traccia di soluzione

Domanda 1

a) Il compilato, secondo le regole di compilazione standard (in particolare del costrutto *while do*, in questo caso senza garanzia che il ciclo venga eseguito almeno una volta), è il seguente:

1. **loop: IF ≥ Rx Rmax, cont**
2. **LOAD RbaseA, Ri, R1**
3. **LOAD RbaseB, Ri, R2**
4. **ADD R1, R2, Rtemp**
5. **IF ≥ 0 Rtemp, else**
6. **SUB R0, Rtemp, Rtemp**
7. **else: ADD Rx, Rtemp, Rx**
8. **INCR Ri**
9. **IF = Ri, RNmeno1, cont**
10. **GOTO loop**
11. **cont: …**

Si osservi come solo in seguito ad una ulteriore analisi del codice sarebbe possibile fondere l'istruzione 9 e la 10 in una istruzione come IF = Ri, RN, loop. Nel seguito non considereremo questa ottimizzazione.

NEW,OLD-0

Valutiamo il tempo di completamento con il metodo analitico, in assenza di ottimizzazioni.

I salti condizionati alle istruzioni 1, 9 risultano normalmente non presi (sono presi solo alla fine del ciclo, ripetuto N volte come da specifiche), mentre il GOTO nella 10 è sempre preso. L'IF ≥ nella 5 risulta sempre preso, come da specifiche. Quindi avremo $\lambda=2/9$.

Abbiamo due dipendenze logiche di distanza $k = 1$ (4 su 5 e 8 su 9), nel primo caso con $N_Q = 2$ nel secondo con $N_Q = 1$. Quindi in media $N_Q = 3/2$ e $d_1=2/9$. Il tempo di servizio per istruzione vale:

$$T = (1+\lambda)t + t\, d_1*(N_Q + 1 - 1) = 14\, t/9$$

Per ognuna delle N iterazioni vengono eseguite $9N$ istruzioni, da cui il tempo di completamento in assenza di fault di cache:

$$T_c \sim 14\, N\, t = 28\, N\, \tau$$

OLD-1

I salti condizionati alle istruzioni 1, 9 risultano normalmente non presi (sono presi solo alla fine del ciclo, ripetuto N volte come da specifiche), mentre il GOTO nella 10 è sempre preso. L'IF ≥ nella 5 risulta sempre preso, come da specifiche. Quindi abbiamo 9 istruzioni: 3 di classe IF, 1 di classe GOTO, 2 di classe LOAD, 3 aritmetiche corte. Quindi il tempo necessario ad eseguire le 9 istruzioni è pari a

$$T_{iter} = 9Tch + 3Tex\text{-}if + Tex\text{-}goto + 2\, Tex\text{-}ld + 3\, Tex\text{-}add$$

$$= 9 * (2\tau + tc) + 3(2\tau) + 1(\tau) + 3\,(2\tau+tc) + 3(\tau) = 25\tau+12tc$$

Dove tc è il tempo di accesso in cache, visto dal processore, di valore 3τ. Quindi:

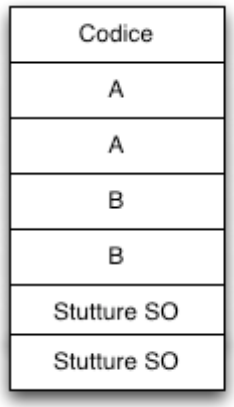
$$T_{iter} = 61\, \tau$$

$$T_c = N\, T_{iter} = 61\, N\, \tau$$

b) Il ciclo scorre i due vettori sequenzialmente, per intero secondo le specifiche. Il working set del programma è quindi costituito dal codice (tutto) e, in ogni istante, da un blocco di A e un blocco di B. Per entrambi i vettori abbiamo infatti località negli accessi, ma non riuso.

c) Dobbiamo considerare le scelte di allocazione compiute dal compilatore nella memoria virtuale del processo.

Con i dati delle specifiche, supponiamo che il compilatore allochi il codice nella prima pagina logica (4K) e di seguito le due pagine logiche di A e le due di B. La memoria virtuale del processo avrà quindi un aspetto come quello in figura (parte sinistra).



IPL	BC	IPF
0		5
1		4
2		0
3		1
4		6
5		9
6		3

La Tabella di Rilocazione potrebbe essere quella della parte destra della figura. Con tale allocazione della memoria principale, gli indirizzi fisici generati durante la prima iterazione, con le ipotesi delle specifiche, sono:

5*4K+0 (IF ≥),

5*4k+1 (prima LOAD),

4*4K+0 (A[0]),

5*4k+2 (seconda LOAD),

1*4K+0 (B[0]),

5*4K+3 (ADD R1),

5*4K+4 (IF ≥ 0),

5*4K+6 (ADD Rx),

5*4K+7 (INCR),

5*4K+8 (IF =),

5*4K+9 (GOTO).

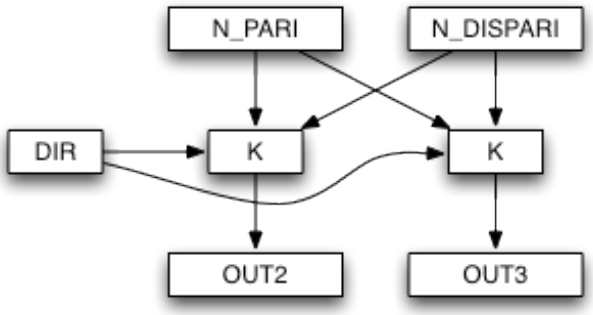
Domanda 2

L'interfaccia con U0 ha un registro X in ingresso da 4 bit, un indicatore RDYin0 e un ACKout0. L'interfaccia con U1 ha solo un RDYsig e un ACKsig, senza registro per il dato, in quanto SIG è un messaggio senza valore (la pura sincronizzazione è data da RDYsig e solo da esso). L'interfaccia con U2 (U3) ha un indicatore RDYout2 (RDYout3), un ACKout2 (ACKout3) e un registro OUT2 (OUT3).

Per scandire il loop utilizziamo un contatore IND da 7 bit. Dei 10 bit CHE servono per indirizzare la memoria, i 4 meno significativi sono il valore di X; quindi gli indirizzi generati sono sempre nella forma IND_m ° X (° operatore di concatenazione).

Per controllare se la parola letta è pari o dispari controlliamo il bit meno significativo. Il numero di posizioni pari (dispari) lo memorizziamo nei registri temporanei N_PARI (N_DISPARI), le cui uscite entrano in OUT2 e OUT3

mediante commutatori, comandati con la tecnica del *controllo residuo* da un registro di un bit DIREZIONE. Tale registro è inizializzato a zero e viene complementato ad ogni ricezione di un SIG:



Il microcodice potrebbe essere il seguente

0. (RDYin, RDYsig = 00) nop, 0;

   (= 10) 0→N_PARI, 0→N_DISPARI, 64→IND, X→TEMPX, reset RDYin, set ACKin, 1;

   (= -1) reset RDYsig, set ACKsig, 0→N_PARI, 0→N_DISPARI, *not*(DIREZIONE) →DIREZIONE, 0

1. (IND_0, RDYsig, ($M[IND_m \circ TEMPX]$)$_{31}$, ACKout2, ACKout3 = -1---) reset RDYsig, set ACKsig,
   0→N_PARI, 0→N_DISPARI, *not*(DIREZIONE) →DIREZIONE, 0;

   (= 000--) N_PARI + 1→N_PARI, IND - 1→IND, 1;

   (= 001--) N_DISPARI + 1→N_DISPARI, IND - 1→IND, 1;

   (= 10-00, = 10-10, = 10-01) nop, 1;

   (= 10-11) set RDYout2, set RDYout3, reset ACKout2, reset ACKout3,
   F(N_PARI, N_DISPARI, DIREZIONE) →(OUT2, OUT3), 0

dove la funzione *F* è definita e implementata come detto sopra mediante controllo residuo su DIREZIONE.

Le variabili di condizionamento sono tutte uscite di registri, tranne la lettura da memoria, acceduta sempre con lo stesso indirizzo: il ritardo di stabilizzazione di ($M[IND_m \circ TEMPX]$)$_{31}$ vale quindi $10t_p$. N_PARI, N_DISPARI e IND vengono scritti con constanti o con uscita di una ALU, quindi richiedono un commutatore in ingresso. Occorrono due ALU per le due somme/sottrazioni, quindi una delle due con un commutatore in ingresso. In conclusione:

$$T_{\omega PO} = 10tp$$

$$T_{\sigma PO} = 5tp + 4tp$$

Per la parte controllo abbiamo due stati interni e 6 variabili di condizionamento, dunque certamente avremo 2tp di ritardo sia per $T_{\omega PC}$ che per $T_{\sigma PC}$. Questo porta ad avere

$$\tau = 10tp + \max\{ 2tp + 9tp, 2tp\} + tp = 22\ tp$$

Agli effetti del tempo di servizio T (considerando trascurabile la probabilità di avere un SIG, come da specifiche), vengono eseguite 1+ 64 + 1 microistruzioni. Quindi:

$$T = 66\ \tau = 1452\ tp = 14{,}52\ nsec$$

La banda di elaborazione vale:

$$B = 1/T = 68{,}9\ Mop/sec$$

Domanda 3

NEW, OLD-0

Le ottimizzazioni tendono a ridurre o annullare le degradazioni dovute alle due dipendenze logiche e alle due istruzioni di salto, descritte nella risposta alla Domanda 1.

Lo spostamento della INCR tra ADD R1 e IF ≥ 0 annulla la seconda dipendenza logica e riduce al minimo la prima. Lo stesso risultato si sarebbe ottenuto usando la INCR per applicare il delayed branch alla IF ≥ 0, ma lasciando inalterata la distanza della prima dipendenza logica. Al GOTO può essere applicato delayed branch spostando, ad esempio la ADD Rx.:

```
loop:   IF ≥   Rx Rmax, cont
        LOAD   RbaseA, Ri, R1
        LOAD   RbaseB, Ri, R2
        ADD    R1, R2, Rtemp
        INCR   Ri
        IF ≥ 0   Rtemp, else
        SUB    R0, Rtemp, Rtemp
else:   IF =   Ri, RN, loop
        GOTO   loop, delayed_branch
        ADD    Rx, Rtemp, Rx
cont:   …
```

Quindi, essendo $\lambda = 1/9$, $d_2 = 1/9$ con $N_Q = 2$, si ha:

$$T = (1+\lambda)t + t\, d_2 * (N_Q + 1 - 2) = 11\, t/9$$

$$T_c \sim 11\, N\, t = 22\, N\, \tau$$

OLD-1

Il codice LC di Q è:

```
Q:: channel in ChanFromR(1); …
        receive( ChanFromR, (A, B) );
        < CODICE SEQUENZIALE >
```

In base alle specifiche, quando R esegue la *send* su **ChanFromR**, il processo Q è in attesa sulla *receive*. Di conseguenza, l'implementazione della *send* copia direttamente i valori di A, B nella variabile targa di Q. Ciò permette di risparmiare la stragrande maggioranza del tempo di esecuzione della *receive*, che si riduce solo a pochi test e scritture nel descrittore di canale, risparmiando la copia di due array (in base alle specifiche, N =16K letture più altrettante scritture in memoria).

Per contro, occorre considerare il ritardo nell'elaborazione di Q dovuto alla *commutazione di contesto*, che va a sommarsi al tempo di puro calcolo nella determinazione del tempo di servizio di Q. Tale ritardo addizionale è dell'ordine di 64 LOAD e 64 STORE, più poche altre istruzioni,

$$\sim 128\, (Tch + Tex\text{-}ld) = 128\, (4\, \tau + 2\, t_a)$$

che, come ordine di grandezza, risulta molto minore rispetto al tempo di copia dei due array. Il *rapporto* è circa (16K LOAD + 16K STORE) / (64 LOAD + 64 STORE) = 256 istruzioni con accesso in memoria, cioè due ordini di grandezza.