Architettura degli Elaboratori

Appello del 21 febbraio 2011

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). I risultati verranno pubblicati sulle pagine web del corso/dei docenti appena disponibili.

Domanda 1 (tutti)

Un sistema a livello firmware consta delle unità U, U1, CU, M.

Le unità U e CU sono realizzate sullo stesso chip. CU è una unità cache completamente associativa, operante su domanda, write-through, capacità 64K parole, blocchi di 16 parole.

M è una unità contenente un componente logico memoria di capacità massima 4G parole, interallacciato con 4 moduli. M ha ciclo di clock uguale a 10τ, con τ ciclo di clock di U.

I collegamenti interchip hanno latenza di trasmissione uguale a 5τ.

U è collegato solo a U1 e a CU.

U riceve di U1 messaggi (OP, A, B, C) con OP di 1 bit e A, B, C ognuno di 32 bit.

Se OP = 0, A rappresenta un indirizzo di M, mentre B e C sono due numeri interi. I valori B e C devono essere scritti nelle locazioni di indirizzo A e A+1.

Se OP = 1, A, B, C rappresentano gli indirizzi base in M di tre array di N = 16K interi ciascuno. U esegue il seguente algoritmo:

$\forall\ i = 0\ ..\ N-1:$

{ $C[i] = 0;$

$\forall\ j = 0\ ..\ N-1:$

$C[i] = \text{if } pred(B[j])$

$\text{then } C[i] - B[j] + A[i] / 2048$

$\text{else } C[i] + B[j] - A[i] \ \%\ 4096$

}

La funzione booleana *pred(X)* restituisce vero se e solo se *X* è un numero positivo, minore di 256, esprimibile come $2^h + 2^k$, con $h \neq k$.

I numeri sono rappresentati in complemento a due. È noto il ritardo di stabilizzazione t_p di una porta logica con al più 8 ingressi. Una ALU ha ritardo di stabilizzazione uguale a $5t_p$.

Il requisito è la *minimizzazione del tempo medio di elaborazione della seconda operazione esterna (OP = 1) per quanto riguarda tanto il valore ideale in assenza di fault di cache, quanto il valore effettivo considerando i fault di cache.*

É richiesto il microprogramma di U e la valutazione dei tempi medi di elaborazione di entrambe le operazioni esterne in funzione di τ e N**.**

Prima del microprogramma deve essere riportata la spiegazione di quali scelte sono effettuate per soddisfare il suddetto requisito sui tempi di elaborazione.

Ogni rete combinatoria utilizzata, che sia diversa da una ALU o un commutatore, deve essere completamente definita e implementata.

Domanda 2 (NEW, OLD-0)

Per una macchina D-RISC con CPU pipeline, compilare con ottimizzazioni un programma simile alla computazione della Domanda 1:

```
int A[N], B[N], C[N];
∀ i = 0 .. N – 1:
        { C[i] = 0;
          ∀ j = 0 .. N – 1:
                C[i] = if (B[j] < 0)
                                then C[i] – B[j] + A[i] / 2048
                                else C[i] + B[j] – A[i] % 4096
        }
```

Solo agli effetti della valutazione, si assuma trascurabile la probabilità che *B[j] < 0*.

Valutare il tempo di completamento per una architettura con la stessa cache primaria e memoria principale della Domanda 1 e senza cache secondaria.

Domanda 2 (OLD-1)

Si consideri il supporto a tempo di esecuzione della *comunicazione sincrona* tra processi LC.

Scrivere e spiegare il codice D-RISC della fase di *copia* del messaggio, tanto per il supporto della *send* quanto per quello della *receive*.

All'inizio del codice richiesto, nel registro *Rch* è presente l'indirizzo logico base del descrittore di canale e nel registro *Rmsg* l'indirizzo logico base del valore del messaggio, oppure nel registro *Rvtg* l'indirizzo logico base della variabile targa.

Valutare il tempo medio di elaborazione della copia in funzione della lunghezza *L* del messaggio in parole.

L'architettura dispone di cache primaria e memoria principale come nella Domanda 1 e non ha cache secondaria.

Soluzione

Domanda 1 (tutti)

Il requisito è che il tempo medio di elaborazione della seconda operazione esterna:

$$T_1 = T_{1\text{-}id} + T_{fault}$$

sia minimizzato tanto per quanto riguarda $T_{1\text{-}id}$ quanto T_{fault}. In entrambi i casi, trattandosi di un doppio loop della stessa cardinalità e con i calcoli concentrati nel loop più interno, si tratta di minimizzare il tempo di elaborazione del loop più interno

Riguardo alla minimizzazione di $T_{1\text{-}id}$, si osserva che il calcolo richiesto è una espressione condizionale, la cui realizzazione più efficiente è un'unica rete combinatoria in modo da minimizzare il prodotto numero di cicli di clock per ciclo di clock, senza variabili di condizionamento complesse.

Detti *a* = A[i], *b* = B[i], *c* = C[i], la rete *F(a, b, c)* è cosi definita:

- viene utilizzata una ALU1 per eseguire *r1 = c + b* oppure *r1 = c – b*, ed una ALU2 in cascata per eseguire *c = r1 – r2* oppure *c = r1 + r2*, dove *r2* è l'uscita di un commutatore avente in ingresso la parola A/2048 (11 zeri concatenati ai 21 bit più significativi di *a*) e la parola A%4096 (20 zeri concatenati ai 12 bit meno significativi di *a*);
- il commutatore, ALU1 e ALU2 sono controllati dall'uscita della rete combinatoria che implementa *pred(c)*, così definita: i 24 bit più significativi di *c* uguali a zero e, degli 8 bit meno significativi, due e solo due uguali a uno e gli altri 6 a zero (*scrivere l'espressione logica*).

Il ritardo della rete *pred* è $4t_p$, quindi il ritardo della rete *F* è di $14t_p$.

Nella scrittura del microprogramma occorre, ovviamente, procurare i valori di *a, b* interagendo a domanda risposta con CU e, all'uscita del loop più interno, scrivere il valore di *c* in cache. Poiché la cache è write-through, tutte le scritture avvengono anche in memoria (a cura dell'unità CU). Si verifica facilmente che la banda delle richieste di scrittura (una ogni loop esterno) è minore della banda della memoria interallacciata.

La computazione è caratterizzata da solo località per gli array A e C e da località e riuso per B. L'insieme di lavoro è quindi costituito da tutti i blocchi di B e dal blocco corrente di A e di C. Tale insieme di lavoro può risiedere permanentemente in cache, la cui capacità è maggiore di $N + 2\sigma$. A questo scopo, le richieste di lettura degli elementi di B sono accompagnate dall'opzione "non deallocare", così che ogni blocco di B, una volta caricato in cache in seguito al fault sul primo accesso, vi risieda permanentemente. Quindi, il numero di fault è dato da

$$N_{fault} = 2\,\frac{N}{\sigma}$$

senza considerare quelli sugli elementi di C, in quanto non provocano trasferimento di blocco.

L'applicabilità del riuso di B è importante, in quanto la penalità per i fault di cache scende da ordine $O(N^2)$ a $O(N)$, quindi trascurabile rispetto al tempo di elaborazione ideale che è $O(N^2)$.

La latenza di trasferimento di un blocco da memoria principale a cache vale:

$$T_{trasf} = 2\,T_{tr} + \frac{\sigma}{m}\,\tau_M + m\tau = 54\,\tau$$

Quindi

$$T_{fault} = N_{fault} * T_{trasf} = \frac{2 * 54}{16}\,N\,\tau \sim 7\,N\,\tau$$

La seconda operazione esterna consta solo di due scritture interagendo con CU a domanda risposta. Sulla prima viene generato un fault di cache, che non ha effetto sulle prestazioni essendo il blocco usato in sola scrittura ed essendo aggiornato in memoria attraverso il meccanismo write-through (le due scritture avvengono in moduli distinti di M).

Microprogramma:

0. (RDY1, OP = 0 -) nop, 0;
   (= 1 0) reset RDY1, set ACK1, A → IND, B → DATAOUT, C → C1, write → OPC,set RDYOUTC, 1;
   (= 1 1) reset RDY1, set ACK1, A → INDA, B → INDB, C → INDC, 0 → I, 3

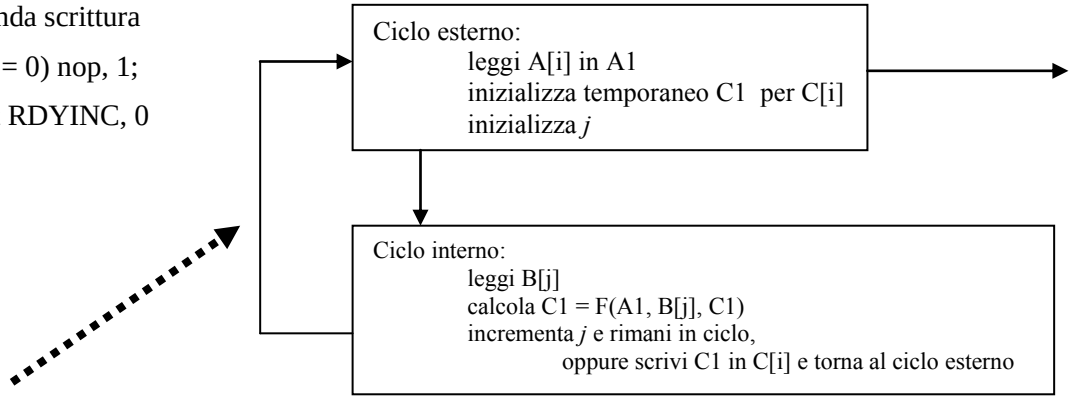
Prima operazione esterna:

attesa fine prima scrittura e richiesta seconda scrittura:

1. (RDYINC = 0) nop, 1;
   (= 1) reset RDYINC, IND + 1 → IND, C1 → DATAOUT, write → OPC, set RDYOUTC, 2

attesa fine seconda scrittura

2. (RDYINC = 0) nop, 1;
   (= 1) reset RDYINC, 0



Seconda operazione esterna:

controllo ciclo esterno, richiesta lettura A[i]; all'uscita del ciclo esterno si torna in 0

3. (I_0 = 0) INDA → IND, read → OPC, set RDYOUTC, INDA + 1 → INDA, 4;
   (= 1) "nop", 0

attesa fine lettura A[i], inizializzazione indice *j* loop interno, inizializzazione temporaneo C1 per C[i]

4. (RDYINC = 0) nop, 4;
   (= 1) reset RDYINC, DATAIN → A1, 0 → J, 0 → C1, 5

controllo loop interno, richiesta lettura B[j]; all'uscita del ciclo interno richiesta scrittura C[i]

5. (J_0 = 0) INDB → IND, (read, *non_deallocare*) → OPC, set RDYOUTC, 6;
   (= 1) INDC → IND, C1 → DATAOUT, write → OPC, INDC + 1 → INDC, 7

attesa fine lettura B[j], calcolo della funzione F, si continua nel loop interno

6. (RDYINC = 0) nop, 6;
   (= 1) reset RDYINC, *F(A1, DATAIN, C1)* → C1, INDB + 1 → INDB, J + 1 → J, 5

attesa fine scrittura C[i], si continua nel ciclo esterno

7. (RDYINC = 0) nop, 7;
   (= 1) reset RDYINC, I + 1 → I, 3

Essendo il tempo di accesso in cache $t_c = 2\tau$, i tempi medi ideali di elaborazione sono:

$$T_{0\text{-}id} = 3\tau + 2t_c = 7\tau$$

$$T_{1\text{-}id} \sim N^2 (2\tau + t_c) = 4 N^2 \tau$$

I tempi di elaborazione effettivi sono:

$$T_0 = T_{0\text{-}id} = 7\tau$$

$$T_1 = T_{1\text{-}id} + T_{fault} \sim T_{1\text{-}id} = 4 N^2 \tau$$

Per la valutazione del ciclo di clock:

$$T_{\omega PO} = 0$$

$$T_{\omega PC} = T_{\sigma PC} = 2t_p$$

$T_{\sigma PO}$ (ritardo della funzione F + commutatore all'ingresso di C1) = $16t_p$

$$\tau = 19\ t_p$$

Domanda 2 (NEW, OLD-0)

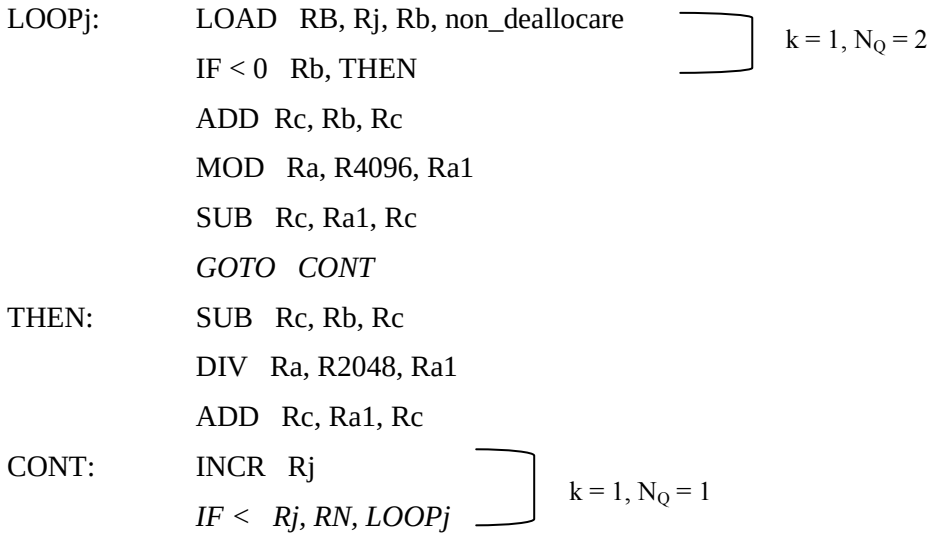
Il codice D-RISC senza ottimizzazioni è:

```
LOOPi:   CLEAR   Rc
         LOAD    RA, Ri, Ra
         CLEAR   Rj

LOOPj:   LOAD    RB, Rj, Rb, non_deallocare
         IF < 0  Rb, THEN
         ADD     Rc, Rb, Rc
         MOD     Ra, R4096, Ra1
         SUB     Rc, Ra1, Rc
         GOTO    CONT
THEN:    SUB     Rc, Rb, Rc
         DIV     Ra, R2048, Ra1
         ADD     Rc, Ra1, Rc
CONT:    INCR    Rj
         IF <    Rj, RN, LOOPj

         STORE   RC, Ri, Rc
         INCR    Ri
         IF <    Ri, RN, LOOPi
         END
```

Alla luce di quanto già discusso nella Domanda 1, ottimizziamo il loop interno. Se la generica iterazione di questo loop ha tempo di completamento T_{iter}, allora il tempo di completamento complessivo è dato da:

$$T_c = N^2 T_{iter} + T_{fault} \sim N^2 Titer$$

Nel codice del loop interno

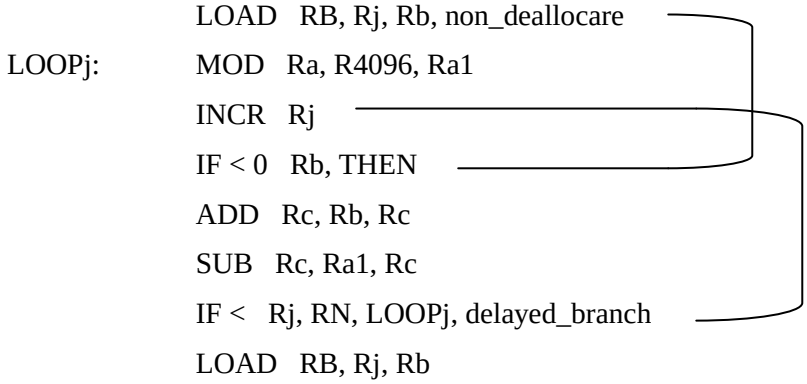


si rilevano le dipendenze logiche indicate e istruzioni che provocano salto, GOTO e IF < Ri, in quanto la probabilità che IF< 0 provochi salto è trascurabile.

Per ottimizzare,

- replichiamo codice nei due rami *then* ed *else*, per eliminare GOTO,
- applichiamo la tecnica del delayed branch alla IF < Rj,
- anticipiamo INCR Rj e DIV/MOD per aumentare la distanza delle dipendenze logiche:

Considerando solo il ramo else nel loop interno:



Il risultato è quindi:

$$\lambda = 0 \qquad \Delta = 0 \qquad T = t = 2\tau \qquad \varepsilon = 1$$

$$T_{iter} = 7\,T = 14\tau$$

$$T_c = \sim 14\,N^2\,\tau$$

La valutazione dell'impatto della memoria cache è identico a quanto visto nella Domanda 1, quindi trascurabile sul tempo di completamento grazie al riuso di B.

Domanda 2 (OLD-1)

Facciamo riferimento alla struttura tipica del descrittore di canale: nell'ordine, campo *wait*, lunghezza *L*, riferimento a variabile targa o messaggio, riferimento al PCB del processo in attesa.

Con l'implementazione alla pari, sia nella *send* che nella *receive*, in modo del tutto duale, viene copiato il messaggio direttamente nella variabile targa.

Assumiamo che il metodo per trattare le strutture condivise riferite indirettamente sia quello degli *indirizzi logici coincidenti.*

Il codice della *send* è è:

```
            LOAD   Rch, 1, RL
            LOAD   Rch, 2, Rvtg          nella receive sarebbe: LOAD  Rch, 2, Rmsg
            CLEAR  Ri
LOOP:       LOAD   Rmsg, Ri, Rx
            STORE  Rvtg, Ri, Rx
            INCR   Ri
            IF <   Ri, RL, LOOP
```

Il tempo di completamento di questo programma è dato da:

$$T_{copy} = T_{id} + T_{fault}$$

dove, essendo $t_c = 3\tau$:

$$T_{id} = 3\,T_{ch} + 2\,T_{ex\text{-}load} + T_{ex\text{-}corte} + L\,(4\,T_{ch} + 2\,T_{ex\text{-}load} + T_{ex\text{-}corte} + T_{ex\text{-}if}) = (26 + 33\,L)\,\tau$$

Possiamo assumere che il descrittore di canale risieda già in cache (è stato acquisito all'atto della lettura del campo *wait*), ma dobbiamo considerare che il messaggio e la variabile targa non risiedano in cache. Essendo strutture caratterizzate da sola località, abbiamo (T_{trasf} è quello valutato nella Domanda 1):

$$N_{fault} = 2\,\frac{L}{\sigma} = \frac{L}{8}$$

$$T_{fault} = N_{fault} * T_{trasf} \sim 7\,L\,\tau$$

La valutazione dei tempi di trasferimento blocco anche per le scritture è motivato dal fatto che, in questo caso, la banda della memoria interallacciata non è sufficiente a far fronte ad una sequenza di *L* parole molto ravvicinate tra loro. È come se ogni blocco della variabile targa dovesse essere trasferito.

Quindi:

$$T_{copy} = (26 + 40\,L)\,\tau$$