Architettura degli Elaboratori

Appello del 28 giugno 2011

Riportare su tutti i fogli consegnati nome, cognome, matricola, corso, programma d'esame. I risultati saranno comunicati appena disponibili sulle pagine web dei docenti.

Domanda 1 (tutti)

Si consideri una computazione corrispondente al seguente algoritmo (con M potenza di 2 e >> 1):

int A[M];

$\forall$ *i = 0 .. M – 1:*

$\forall$ *j = i .. M – 1:*

A[i] = if pari(A[i] + A[j]) then A[i] + A[j] else A[i] – A[j]

Implementarla mediante una unità di elaborazione U connessa ad una gerarchia di memoria memoria principale – cache. La cache, residente sullo stesso chip di U, è associativa, su domanda, Write-Trough, con capacità $\gamma = M/4$ e blocchi di σ parole. La memoria principale è interallacciata con $\sigma/2$ moduli e ciclo di clock uguale a $p\tau$ (con $p >> 1$ e $p << M$, e τ ciclo di clock di U). I collegamenti interchip hanno latenza di trasmissione uguale a τ. U riceve da una unità U_M l'indirizzo base di A e restituisce alla stessa U_M una segnalazione di fine elaborazione.

Scrivere il microprogramma di U e valutare il tempo medio di elaborazione in funzione di M, σ, p e t_p (una ALU ha ritardo $5t_p$). Spiegare le principali scelte di progetto e la valutazione delle prestazioni.

Domanda 2

Implementare la stessa computazione della Domanda 1, eccetto che il calcolo è

A[i] = if pari(A[i] + A[j]) then ***A[i] ∗ A[j]*** *else* ***A[i] / A[j]***

mediante una procedura assembler D-RISC con parametri di ingresso e di uscita aventi significato analogo alla Domanda 1 stessa. La procedura deve essere eseguibile su:

- **NEW, OLD-0:** una CPU pipeline con Unità Esecutiva avente unità funzionali intere con organizzazione pipeline a quattro stadi, e stessa gerarchia di memoria della Domanda 1 per quanto riguarda la cache dati;
- **OLD-1:** una CPU convenzionale con la stessa gerarchia di memoria della Domanda 1.

Compilare la procedura, ottimizzarla ove possibile, e valutarne il tempo di completamento in funzione di M, σ, p e τ. Spiegare come è stato ottenuto il codice e come si è proceduto alla sua valutazione.

Domanda 3 (NEW, OLD-0)

Per una CPU pipeline D-RISC con EU parallela, descrivere il funzionamento dell'Unità EU_Master e, *sulla base di tale descrizione*, determinare il tempo di servizio ideale di detta unità.

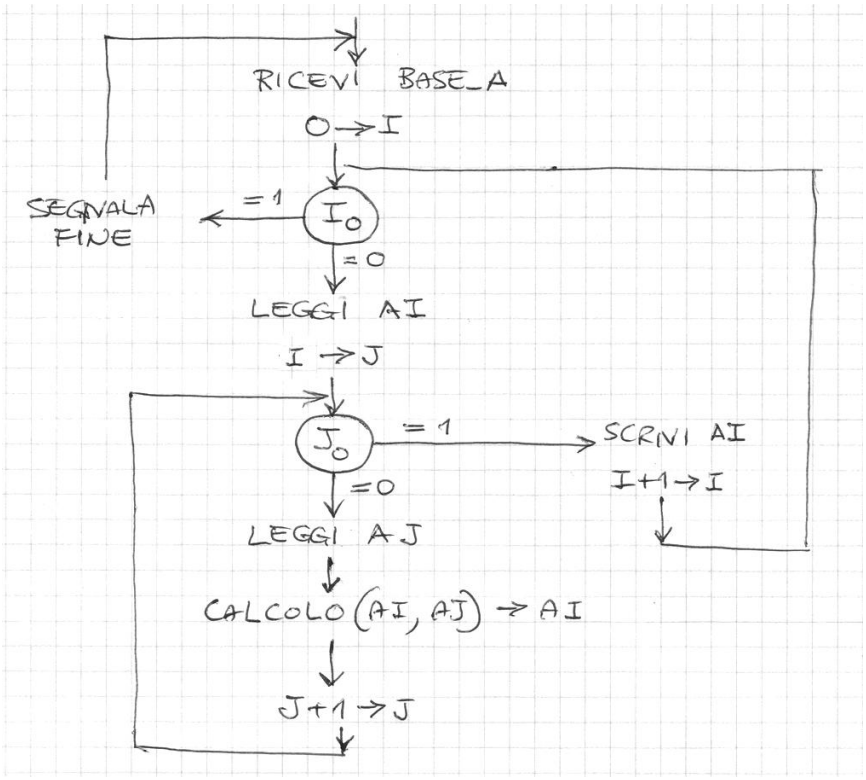
Domanda 3 (OLD-1)

Si consideri una architettura con solo Memory Mapped I/O, in cui ogni unità di I/O dispone di una memoria locale monolitica. Nell'ipotesi che il processore non abbia la possibilità di disabilitare l'utilizzo della cache primaria, determinare il tempo di completamento per effettuare il trasferimento di un array di 100K dalla memoria locale di una unità di I/O alla memoria principale. Esprimere il tempo di completamento in modo parametrico utilizzando simboli opportuni.

Traccia di soluzione

Domanda 1

L'interprete corrisponde al seguente algoritmo:



Supponiamo l'interazione U – U_M a domanda-risposta.

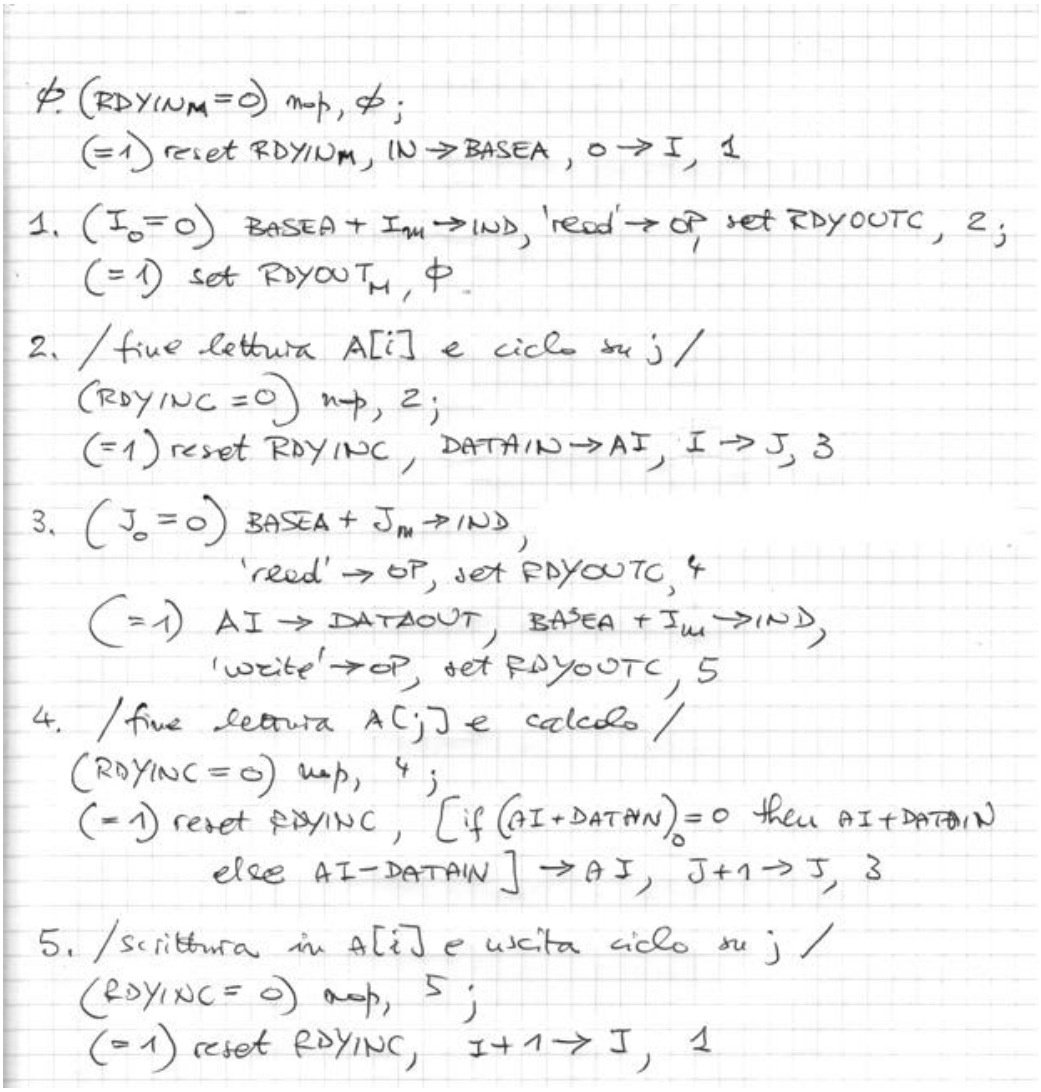
Implementiamo la fase di calcolo mediante controllo residuo: il bit meno significativo di AI + AJ comanda il commutatore di ingresso al registro AI, i cui ingressi sono lo stesso valore di AI + AJ e quello di AI – AJ. Questa soluzione porta ad un ciclo di clock di lunghezza minore o uguale a quello che si avrebbe usando il bit meno significativo di AI + AJ come variabile di condizionamento complessa, e stesso numeri di cicli di clock.

Dal punto di vista della gerarchia di memoria, la computazione, avente tempo di completamento ideale di ordine M^2, ha working set dato da tutti i blocchi di A, in quanto A è interamente riusata nel loop più interno. La cache non può però contenere l'intero array, bensì al più un quarto di esso. Quindi, l'overhead dovuto a trasferimento di blocchi è dell'ordine di M^2. Al più, si può ridurre solo la costante moltiplicativa mantenendo in cache all'incirca un quarto dei blocchi di A con l'opzione "non deallocare" (a firmware, questa condizione si riconosce facilmente). Questa ottimizzazione marginale non verrà applicata nel seguito.

Il microprogramma è descritto nella figura a pagina seguente (l'ottimizzazione marginale è barrata).

Il loop interno (2, 3) viene eseguito in 4τ, tenendo conto del tempo di accesso alla cache in assenza di fault. Il tempo di completamento del loop interno vale quindi $\sim 4\tau * M/2 = 2\tau M$. Il tempo di elaborazione complessivo in assenza di fault vale

$$T_{id} \sim 2 M^2 \tau$$



Il ciclo di clock vale

$$\tau = T_{alu} + T_k + \delta = 8\, t_p$$

in quanto il massimo ritardo per stabilizzare la funzione di transizione dello stato interno di PO è quello della parte di calcolo mediante controllo residuo, e le funzioni di PC si stabilizzano in parallelo a tale funzione.

Il funzionamento Write-Through non introduce alcuna penalità in quanto il tempo di servizio della memoria principale, uguale a $p\tau/4$, è certamente minore del tempo medio di servizio dello stream di richieste di scrittura, uguale a $\sim 2\, M\, \tau$, essendo $p << M$.

Senza l'ottimizzazione marginale, la penalità per il trasferimento dei blocchi vale:

$$T_{fault} = N_{fault} * T_{trasf} = \frac{M^2}{2\sigma}\left(2\,\tau + \frac{\sigma}{4}\,p\,\tau + 4\,\tau\right) \sim \frac{M^2}{2\sigma}\,\frac{\sigma}{4}\,p\,\tau = \frac{M^2\,p\,\tau}{8}$$

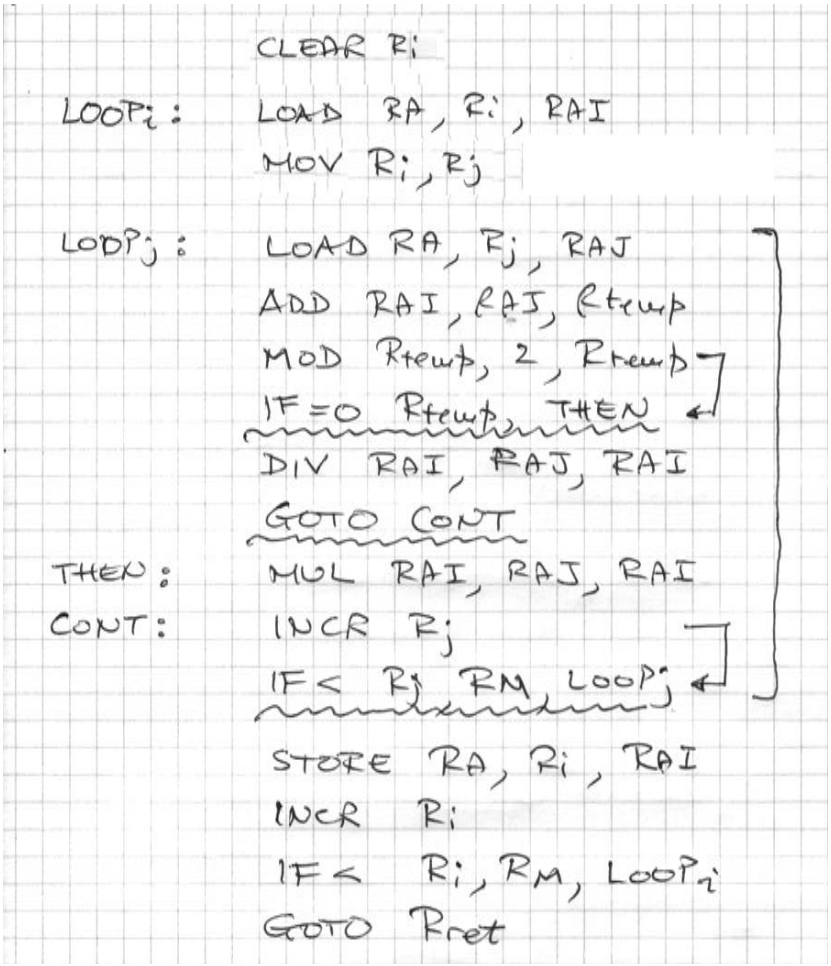
Quindi:

$$T = T_{id} + T_{fault} \sim 2\,M^2\,\tau + \frac{M^2\,p\,\tau}{8} = M^2\,t_p\,(p + 16)$$

in pratica indipendente da σ.

Domanda 2

Il codice compilato senza ottimizzazioni è:



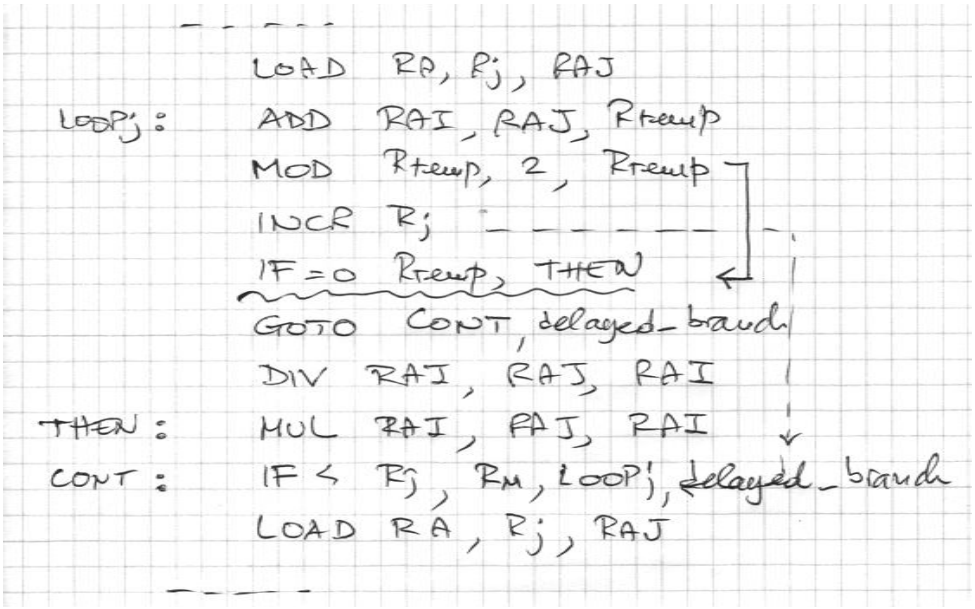
La penalità dovuta ai fault è quella valutata nella Domanda 1.

Il tempo di completamento in assenza di fault si valuta come *M* volte quello del solo loop più interno. La presenza del salto condizionato può essere valutato con una certa probabilità, ad esempio ½.

Per la versione **OLD-1**, il tempo di completamento ideale del loop più interno si valuta in funzione dei tempi di servizio delle varie istruzioni usate (fare).

Per la versione **NEW** e **OLD-0**, le cause di degradazione del loop più interno dipendono dalle due dipendenze logiche segnate in figura e dalle tre istruzioni di salto.

L'ottimizzazione può essere quella mostrata nella figura seguente (da spiegare). Il tempo di servizio non è influenzato dalla latenza delle istruzioni lunghe MUL e DIV, in quanto non appartengono alla sequenza critica della dipendenza logica IU-UE indotta dalla MOD sulla IF = 0 (l'effetto, trascurabile, di tali istruzioni lunghe si avrebbe solo sulla dipendenza indotta sulla STORE nel loop più esterno), mentre l'effetto della latenza della EU parallela si ha a causa dell'istruzione lunga MOD.



Applicando il modello dei costi (con probabilità del predicato IF = 0 uguale a ½):

$$\lambda = \frac{1}{2\,N_{istr}} \qquad \Delta = \frac{5\,t}{N_{istr}} \qquad (spiegare)$$

con N_{istr} = 7,5, da cui:

$$T_{c-id} = \frac{M^2}{2} N_{istr}\ T = \frac{M^2 N_{istr}}{2}[(1+\lambda)\,t + \Delta] = \frac{M^2}{2}(11 + 2\,N_{istr})\,\tau$$

$$T_c = T_{c-id} + T_{fault} = \frac{M^2}{2}(11 + 2\,N_{istr})\,\tau + \frac{M^2\,p\,\tau}{8} = \frac{M^2\,\tau}{8}(p + 104)$$

Può essere interessante confrontare questo valore con quello della Domanda 1.

Domanda 3 (NEW, OLD-0)

Si veda il materiale didattico. Mettere in evidenza:

- la sincronizzazione sui registri generali e float,
- il funzionamento in seguito a richieste di
  - LOAD,
  - aritmetica intera corta,
  - aritmetica intera lunga,
  - aritmetica float,
- l'invio delle modifiche dei registri a IU,
- il possibile funzionamento out-of-order, "controllato" o meno.

Il tempo di servizio ideale vale 1τ.

A causa della latenza delle comunicazioni tra unità, il tempo di servizio effettivo vale 2τ se non si verificano condizioni di attesa, altrimenti è ancora maggiore e dipende dalla latenza delle dipendenze logiche EU-EU e dalla latenza di trasferimento blocchi nella cache dati.

Domanda 3 (OLD-1)

Nelle ipotesi del problema, la funzionalità richiesta può essere implementata soltanto facendo eseguire ad un processo interno il frammento di programma (con N = 100K):

```
LOOP:   LOAD    RA, Ri, Ra
        STORE   RA, Ri, Ra
        INCR    Ri
        IF <    Ri, RN, LOOP
```

Essendo caratterizzato da sola località, il suo tempo di completamento è dominato dai trasferimenti di blocchi da memoria di I/O a cache (LOAD) e, qualunque sia la politica di scrittura, dai trasferimenti di blocchi da cache a memoria principale (STORE):

$$T_{fault} = \frac{N}{\sigma} (T_{trasf-I/O} + T_{trasf-M})$$

dove $T_{trasf\text{-}M}$ dipende dall'architettura della gerarchia di memoria con la memoria principale (organizzazione interallacciata o meno, presenza di cache secondaria o meno, ecc.), mentre, nelle ipotesi del problema, è certamente:

$$T_{trasf-I/O} \sim \sigma\, t_{a-I/O} = \sigma\,(2\,T_{tr-busI/O} + \tau_{MI/O})$$

che causa un overhead di accesso in memoria superiore a quello che avremmo in assenza di cache.