Architettura degli Elaboratori – Seconda prova di verifica intermedia

a.a. 2012-13, 28 maggio 2013

Riportare nome, cognome, numero di matricola e corso A/B

Domanda 1

a) Dato un sistema di elaborazione a livello firmware, mostrare come viene valutato il tempo di servizio ideale e il tempo di servizio effettivo di una unità di elaborazione che ne fa parte.

Si consideri un elaboratore con CPU D-RISC avente architettura pipeline scalare con ciclo di clock τ e comunicazioni a singola bufferizzazione. Dimostrare che è possibile progettare l'Unità Istruzioni in modo tale che il suo tempo di servizio ideale sia uguale a 2τ.

b) La definizione dell'architettura di cui al punto *a)* è completata come segue:

- il set di istruzioni include le operazioni su reali, inclusa la radice quadrata (SQRT);
- Unità Esecutiva parallela con unità funzionali per interi e per reali aventi organizzazione pipeline, rispettivamente a 4 stadi e a 8 stadi;
- cache dati primaria operante su domanda, write-through, capacità di 32 Kparole, blocchi di 8 parole;
- cache secondaria on chip operante con prefetching;
- memoria esterna interallacciata con 8 moduli e ciclo di clock uguale a 80τ, con collegamenti inter-chip aventi latenza di trasmissione uguale a 5τ.

Si consideri la seguente computazione:

int X[N][N]; int FILTRO[N]; float Y[N][N];

$\forall$ *i = 0 .. N–1 :*

$\forall$ *j = 0 .. N–1 :*

Y[i][j] = F(X[i][j], FILTRO[j])

dove $N = 4000$, e la funzione reale $F(a, b)$, con a e b interi, calcola:

$$F(a,b) = \frac{\sqrt{a\,b\ +\ b^2}}{a^2\ +\ b^2}$$

Valutare il tempo di completamento in funzione di N e τ, assumendo che la probabilità di fault della cache secondaria sia trascurabile. Fornire adeguate spiegazioni. La compilazione deve essere ottimizzata.

c) Verificare se l'ipotesi del punto *b)* sulla probabilità di fault della cache secondaria è realistica o meno. Per dare la risposta occorre anche che venga definito come è interconnessa la memoria esterna alla CPU.

Soluzione

a) Il tempo di servizio *ideale* di un'unità (in generale, un modulo) facente parte di un sistema è valutato considerando l'unità in isolamento (o con tempo di interarrivo piccolo a piacere) e tenendo conto della latenza di comunicazione:

$$T_{id} = \max(T_{calc}, L_{com})$$

dove T_{calc} è il tempo medio di elaborazione della funzione affidata all'unità misurato dall'istante in cui sono disponibili i dati d'ingresso a quando sono pronti i risultati, e L_{com} la latenza di comunicazione. Il tempo di servizio *effettivo* coincide con il tempo di interpartenza ed è dato da:

$$T = \max(T_{id}, T_A)$$

dove T_A è il tempo di interarrivo.

Nel caso della CPU data, con interfacce a singola bufferizzazione, è $L_{com} = 2\tau$. Si tratta quindi di dimostrare che è possibile progettare IU con $T_{calc} \leq 2\tau$. In effetti, il microprogramma di IU è eseguibile in un singolo ciclo di clock, corrispondente ad una microistruzione nella quale si valutano contemporaneamente le seguenti condizioni logiche:

1. presenza di nuova istruzione da IM,
2. validità dell'istruzione in base all'identificatore unico,
3. possibile situazione di delayed branch in corso,
4. istruzione corrispondente al codice operativo,
5. contenuto uguale a zero di semafori associati agli eventuali registri generali usati in lettura da IU,
6. verità della condizione di salto in caso di istruzione di salto condizionato,
7. presenza di un nuovo valore inviato da EU,
8. contenuto diventato uguale a zero di semafori associati ai registri generali modificati in seguito all'evento 7 e usati in lettura da IU in istruzioni bloccate,
9. interruzione.

Le condizioni del tipo 1, 7 (e 9) sono valutate in modo nondeterministico, in modo che IU sia sempre pronta a servire messaggi da IM e da EU anche se presenti contempoaneamente.

In conseguenza delle combinazioni significative delle condizioni logiche suddette, IU svolge, nella stessa microistruzione, tutte le azioni necessarie:

A. incremento di semafori associati a registri destinazione di istruzioni aritmetico-logiche o LOAD,
B. calcolo di indirizzi base più indice per dati in LOAD e STORE,
C. modifica di IC,
D. comunicazioni verso IM, DM, EU a seconda dei casi,
E. effettuazione delle stesse azioni per istruzioni sbloccate in seguito all'evento 8,
F. inizio del trattamento interruzione.

(Al più, per ragioni di riduzione della complessità, si può eseguire le azioni E, F in un ulteriore ciclo di clock, senza conseguenze apprezzabili sul tempo di servizio ideale.)

b) Dal punto di vista della gerarchia di memoria, l'insieme di lavoro per i dati è costituito dal blocco corrente di X e di Y (solo località) e da tutti i blocchi di FILTRO (località e riuso). Il riuso su FILTRO è applicabile a C1 essendo $N <$ capacità di C1; il compilatore inserisce l'annotazione opportuna (non_deallocare). Essendo Y in sola scrittura, il numero complessivo di fault è dato da:

$$N_{fault} = N_{fault-A} + N_{fault-FILTRO} = \frac{N^2}{\sigma} + \frac{N}{\sigma} \sim \frac{N^2}{\sigma}$$

Anche per applicare il prefetching a C2 il compilatore inserisce opportune annotazioni (prefetching_C2).

Il codice compilato è quello di un *for* il cui corpo è l'espressione di *F(a, b)* (operante su alcuni registri in virgola fissa R e altri in virgola mobile RF) conclusa da una STOREF. Il codice ottimizzato segue lo schema adottato in benchmark analoghi:

- applicazione del delayed ranch al controllo del ciclo, spostando la STOREF dopo la IF <,
- anticipazione dell'incremento dell'indice, inizializzando il registro base di Y all'indirizzo logico base meno uno.

```
LOOP_i:   CLEAR   Rj
LOOP_j:   LOAD    RX, Rj, Ra, prefetching_C2
          LOAD    RFILTRO, Rj, Rb, non_deallocare, prefetching_C2
          INCR    Rj
          MUL     Rb, Rb, Rbb
          MUL     Ra, Ra, Raa
          MUL     Ra, Rb, Rab
          ADD     Rab, Rbb, Rab
          SQRT    Rab, RFnumeratore
          ADD     Raa, Rbb, Rdenominatore
          DIVF    RFnumeratore, Rdenominatore, RFrisultato
          IF <    Rj, RN, LOOP_j, delayed_branch
          STOREF  RY, Rj, RFrisultato
          ADD     RX, RN, RX
          ADD     RY, RN, RY
          INCR    Ri
          IF <    Ri, RN, LOOP_i
          END
```

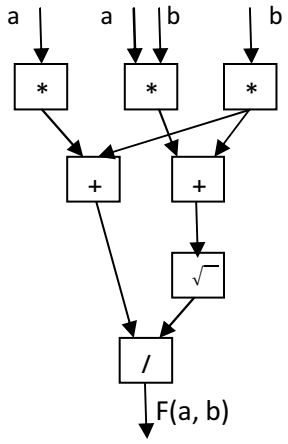
del quale valutiamo solo il loop più interno di 12 istruzioni. L'unica degradazione è dovuta alla dipendenza logica IU-EU indotta dalla DIVF sulla STOREF ($k = 2$, $N_{Qk} = 2$, $d_k = 1/12$, $L_{pipe\text{-}k} = 8$), da cui il tempo di servizio per istruzione in assenza di fault di cache:

$$T_0 = t + \Delta_1 + \Delta_2$$

dove:

$$\Delta_1 = t\, d_k \left(L_{pipe-k} + N_{Qk} + 1 - k\right) = \frac{9}{12}\, t$$

Per quanto riguarda Δ_2, sulla DIVF viene indotta una catena di dipendenze logiche EU-EU del quale occorre valutare le dipendenze sui dati e le relative latenze. Applicando le condizioni di Bernstein al codice di *F(a,b)* si ottiene il grafo di ordinamento parziale (data-flow) di figura.



Il ramo avente latenza maggiore è quello che comprende una moltiplicazione intera, addizione intera, radice quadrata (reale), divisione reale.

Le dipendenze EU-EU su DIVF da considerare, tutte con probabilità d_h = 1/12, sono quindi: SQRT su DIVF (distanza h = 2, $L_{pipe\text{-}h}$ = 8), ADD Rab, Rbb, … su SQRT (distanza h = 1, $L_{pipe\text{-}h}$ = 0), MUL Ra, Rb, … su ADD Rab, Rbb, Rab (distanza h = 1, $L_{pipe\text{-}h}$ = 4).

Quindi

$$\Delta_2 = t \sum_h d_h \left(L_{pipe-h} + 1 - h\right) = \frac{11}{12}\, t$$

In conclusione:

$$T_0 = \frac{32}{12}\, t$$

Il tempo di completamento in assenza di fault di cache vale:

$$T_{c-0} \sim N\, T_{c-inner} = 12\, N^2\, T_0 = 32\, N^2\, t = 64\, N^2\, \tau$$

Nel *caso peggiore* di non considerare la sovrapposizione tra calcolo e trasferimento del blocco in seguito a fault di C1:

$$T_{fault} = N_{fault} * T_{trasfC2-C1} = \frac{N^2}{\sigma} * 2\, \sigma\, \tau = 2\, N^2\, \tau$$

$$T_c = T_{c-0} + T_{fault} = 66\, N^2\, \tau$$

Per una valutazione più precisa, si consideri che il trasferimento di blocchi da C2 a C1 si sovrappone parzialmente al calcolo di alcune istruzioni: per $i > 0$, dopo la richiesta a C2 del trasferimento del blocco di X, vengono eseguite LOAD FILTRO, INCR e MUL Rb, Rb, Rbb, che si sovrappongono al trasferimento del blocco di X, mentre MUL Ra, Ra, Raa deve attendere il trasferimento. Quindi, sul tempo di completamento si risparmiano *3t* ogni 8 iterazioni:

$$T_{fault} = N_{fault} * T_{trasfC2-C1} = \frac{N^2}{\sigma} * (2\, \sigma\, \tau - 6\tau) = 1.25\, N^2\, \tau$$

$$T_c = T_{c-0} + T_{fault} = 65.25\, N^2\, \tau$$

È necessario valutare l'effetto delle scritture in Y con il metodo write-through. Ogni scrittura in Y viene effettuata in parallelo in C1, C2 e M. Il tempo di interpartenza delle richieste è dato dal tempo di servizio di una iterazione del loop più interno:

$$T_{richiesta} = 12\, T_0 = 32\, t = 64\, \tau$$

che rappresenta il tempo di interarrivo delle richieste di scrittura a M. Il tempo di servizio di M per accessi a indirizzi consecutivi è uguale a:

$$T_{offerta} = \frac{1}{B_{M-max}} = \frac{\tau_M}{m} = 10\tau$$

Anche tenendo conto che M è impegnata in trasferimenti di blocchi di X verso C2, si ha che la banda offerta è senz'altro maggiore della banda richiesta, per cui vale il tempo di completamento valutato sopra.

c) Valutiamo se è valida l'assunzione che ogni richiesta di blocco di C1 a C2 trovi il blocco già in C2. La cache secondaria opera con prefetching sui blocchi di X (e di FILTRO): poiché i blocchi di C2 sono ampi un multiplo di σ parole, ogni richiesta di C2 a M è relativa a più blocchi di σ parole. Man mano che C2 riceve un blocco di σ parole, questo diviene disponibile su richiesta per C1.

Il tempo di interarrivo delle richieste di un blocco di X da C1 a C2 è dato dal tempo di servizio di σ iterazioni del loop più interno:

$$T_{richiestaC1-C2} = \sigma\, T_{c-inner} = 8 * 12\ T_0 = 8 * 32\ t = 512\,\tau$$

La latenza di un trasferimento di un blocco di σ parole da M a C2 è data da:

- con struttura di interconnessione consistente semplicemente in $m = 8$ collegamenti diretti:

$$L_{trasf-64} = 2\,T_{tr} + \frac{\sigma}{m}\tau_M + m\tau = 98\,\tau$$

- con struttura di interconnessione (più realistica) ad albero binario con $m = 8$ foglie:

$$L_{trasf-64} = 2\,T_{tr} + \frac{\sigma}{m}\tau_M + \log_2 m\ (\tau + T_{tr}) = 108\,\tau$$

In entrambi i casi, tale latenza si sovrappone completamente all'elaborazione degli 8 blocchi precedenti nella CPU, per cui l'assunzione fatta in *b)* è realistica.