Architettura degli Elaboratori

Appello del 8 giugno 2011 e seconda prova di verifica intermedia

Domanda 1 (tutti)

Una unità di elaborazione U, comunicante con una unità di ingresso e una di uscita, è descritta dal seguente microprogramma:

0. (RDYIN = 0) nop, 0;

   (= 1) reset RDYIN, set ACKIN, IN $\rightarrow$ X, 0 $\rightarrow$ I, 0 $\rightarrow$ S, 1;

1. (I_0, segno(A[I_m] – X), zero(A[I_m] – X), ACKOUT = 0 0 0 –) X $\rightarrow$ A[I_m], S + 1 $\rightarrow$ S, I + 1 $\rightarrow$ I, 1;

   (= 0 0 1 –, 0 1 – –) I + 1 $\rightarrow$ I, 1;

   (= 1 – – 0) nop, 1;

   (= 1 – – 1) S $\rightarrow$ OUT, set RDYOUT, reset ACKOUT, 0

Detto t_p il ritardo di una porta logica con al massimo 8 ingressi, le ALU e la memoria A hanno ritardo di $5t_p$.

a) Definire le funzioni delle uscite e, relativamente alla memoria A, di transizione dello stato interno della Parte Operativa, e mostrare le corrispondenti parti della struttura.

b) Scrivere una versione equivalente del microprogramma che faccia uso di controllo residuo per comandare la scrittura in A e in S, e mostrare le modifiche alla struttura della Parte Operativa.

c) Determinare la differenza nei tempi di elaborazione della versione data e della versione con controllo residuo.

Domanda 2 (NEW, OLD-0) e Domanda 1 della seconda prova di verifica intermedia

Un programma sequenziale è definito dal seguente algoritmo:

int A[N];

$\forall$ *i = 0 .. N – 1:*

if A[i % C] = 0 then A[i] = 0 else A[i] = A[i] ∗ A[i] + 1

con N = 512K, C = 2K.

Compilarlo con ottimizzazioni e determinarne il tempo di completamento per una CPU pipeline scalare D-RISC con le seguenti caratteristiche:

- Unità Esecutiva di latenza unitaria per ogni operazione;
- cache dati associativa, di capacità 32K parole, blocchi di 8 parole, scritture con il metodo write-through. La memoria principale è interallacciata con 4 moduli e ciclo di clock uguale a 100τ;
- cache secondaria on-chip di capacità 1Mega parole;

supponendo

- trascurabile la probabilità di eseguire il ramo *then*,
- che, all'atto dell'esecuzione del processo, l'array A sia interamente presente in cache secondaria.

Domanda 2 della seconda prova di verifica intermedia

Si discutano le caratteristiche della Unità Esecutiva Master per una Unità Esecutiva contenente tre unità pipeline dedicate, rispettivamente, alla moltiplicazione e divisione in virgola fissa (4 stadi), addizione e sottrazione (4 stadi) e moltiplicazione e divisione (8 stadi) in virgola mobile. In particolare:

- le interazioni con Unità Istruzioni e con Memoria Dati,
- la gestione dei registri generali e in virgola mobile,
- l'implementazione delle dipendenze sui dati.

Domanda 3 (OLD-1)

Con riferimento al processore D-RISC ed al modello a processi LC si discutano:

- le cause delle commutazioni di contesto
- l'implementazione della commutazione di contesto che porta in esecuzione il primo processo nella lista dei processi pronti

mettendo in evidenza l'utilizzo delle strutture dati del supporto e le istruzioni assembler utilizzate.

Traccia di soluzione

Vengono riportati solo gli elementi essenziali per comprendere la soluzione, che va completata con adeguate spiegazioni, codici e schemi.

Per molte spiegazioni e per il formalismo si rimanda al libro di testo

Domanda 1 (tutti)

a) La PO comprende tre ALU, delle quali una dedicata alle funzioni segno e zero corrispondenti a due variabili di condizionamento allo scopo di garantire la condizione necessaria per la correttezza..

La funzione delle uscite della PO consta delle funzioni corrispondenti alle 5 variabili di condizionamento e delle funzioni identità delle uscite esterne OUT, RDYOU e ACKIN:

x_0 = RDYIN

x_1 = segno(A[I_m] – X)

x_2 = zero(A[I_m] – X)

x_3 = ACKOUT

z_1 = ACKIN

z_2 = OUT

z_3 = RDYOUT

La funzione di transizione dello stato interno, relativamente ad A, deve evidenziare che la transizione di stato interno di *A[j]* avviene se e solo se, in corrispondenza dell'impulso di clock, la scrittura è abilitata *e l'indirizzo Im è uguale a j:*

$in_{A[j]}$ = **when** $\beta_A = 1$ **and** I_m = j **do** X

b) Il controllo residuo richiesto comporta la modifica del microprogramma come segue, eliminando le variabili di condizionamento segno(A[I_m] – X) e zero(A[I_m] – X)::

0. (RDYIN = 0) nop, 0;

   (= 1) reset RDYIN, set ACKIN, IN → X, 0 → I, **0 → S| f_S,**, 1;

1. **(I_0, ACKOUT = 0 –) X → A[I_m] | f_A, S + 1 → S | f_S,** I + 1 → I, 1;

   (= 1 0) nop, 1;

   (= 1 1) S → OUT, set RDYOUT, reset ACKOUT, 0

Le funzioni booleane f_A e f_S forniscono i valori delle variabili di controllo per abilitare la scrittura in A e S rispettivamente. Esse possono essere definite nel modo seguente:

detto

$$b = \overline{segno(A[Im]- X)} \ \ \overline{zero(A[Im]- X)}$$

e detta *y* la variabile dello stato interno della PC:

$$f_A = y\, I_0\, b$$

$$f_{S=} \bar{y}\, RDYIN + f_A$$

In altri termini, sono state ricavate le espressioni logiche delle variabili di controllo per abilitare la scrittura in A e in S.

c) I due microprogrammi sono eseguiti nello stesso numero di cicli di clock. Il ciclo di clock dell'unità data vale 20 t_p, in quanto

$$T_{\omega PO} = 10\ t_p,$$

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

$$T_{\sigma PO} = 7\ t_p$$

(incremento di S, I, che prevale sulla scrittura di X in A)

Il ciclo di clock dell'unità con controllo residuo vale 19 t_p, in quanto

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

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

$$T_{\sigma PO} = 16\ t_p$$

(stabilizzazione di f_A *in* $11t_p$ seguita dalla stabilizzazione del selezionatore di A per la scrittura di X)

Domanda 2 (NEW, OLD-0) e Domanda 1 della seconda prova di verifica intermedia

Analizziamo la computazione per determinare il working set della cache dati primaria. L'array A è complessivamente caratterizzato da località, mentre la parte iniziale di 2K è caratterizzata anche da riuso. Date le dimensioni, questa parte può essere contenuta tutta in cache. Il numero di fault è quindi solo N/σ. Essendo $2\sigma\tau$ la latenza di trasferimento di un blocco da cache secondaria a cache primaria, la penalità dovuta ai fault di cache vale $T_{fault} = 2N\tau$.

Il codice D-RISC senza ottimizzazioni è:

```
LOOP: 1. MOD   Ri, RC, Rmod
      2. LOAD  RA, Rmod, Ra1, non_deallocare
      3. IF = 0 Ra1, THEN
      4. LOAD  RA, Ri, Ra
      5. MUL   Ra, Ra, Ra
      6. INCR  Ra
      7. STORE RA, Ri, Ra
      8. GOTO CONT
THEN: 9. CLEAR Ra
     10. STORE RA, Ri, Ra
CONT: 11. INCR  Ri
     12. IF <   Ri, RN, LOOP
     13. END
```

Le degradazioni per la CPU pipeline data sono dovute a due salti (8, 12; la 3 provoca un salto con probabilità trascurabile per ipotesi) e quattro dipendenze logiche: 1 su 2 (k = 1, NQ = 1), 2 su 3 (k = 1, NQ = 2), 6 su 7 (k = 1, NQ = 2), 11 su 12 (k = 1, NQ = 1).

L'effetto dei due salti può essere annullato con delayed branch: invertendo la 7 e la 8, e replicando la 1 dopo la 12.

Spostando l'istruzione INCR Ri dopo la 2, a condizione di usare una base decrementata di 1 per A nella seconda LOAD (registro RAdec), si elimina la quarta dipendenza logica, rimane inalterata la prima, e la seconda e terza divengono di distanza k = 2:

```
        1. MOD   Ri, RC, Rmod
LOOP:   2. LOAD   RA, Rmod, Ra1, non_deallocare
        3. INCR   Ri
        4. IF = 0 Ra1, THEN
        5. LOAD   RAdec, Ri, Ra
        6. MUL    Ra, Ra, Ra
        7. INCR   Ra
        8. GOTO CONT, delayed_branch
        9. STORE   RA, Ri, Ra
THEN:  10. CLEAR   Ra
       11. STORE   RA, Ri, Ra
CONT:  12. IF <   Ri, RN, LOOP
       13. MOD   Ri, RC, Rmod
       14. END
```

Le istruzioni eseguite ad ogni iterazione sono $N_{istr} = 10$. Applicando il modello dei costi con EU di latenza unitaria:

$$\lambda = 0$$

$$\Delta = \frac{3}{10}\, t$$

$$T = \frac{13}{10}\, t$$

$$T_c = N_{istr}\ N\ T + T_{fault} = 28\, N\, \tau$$

Le scritture in memoria hanno una frequenza uguale a $1/28\tau$. La banda della memoria principale è maggiore di tale frequenza: $4/100\tau = 1/25\tau$, per cui nessuna degradazione ha luogo nell'applicare la tecnica del Write-Through, che comporta la scrittura in parallelo in cache primaria e in memoria principale, oltre che in cache secondaria.

Domanda 2 della seconda prova di verifica intermedia

Facendo riferimento alla dispensa, la risposta consta dei seguenti punti:

1. descrizione dello schema della EU;
2. forme di parallelismo adottate;
3. funzionamento della EU_Master per ogni istruzione ricevuta: caso delle operazioni aritmetiche corte, caso delle LOAD, caso delle operazioni complesse;
4. sincronizzazione sui registri generali e in virgola mobile mediante semafori;
5. comunicazioni di aggiornamento di registri alla IU;
6. ottimizzazioni in seguito a dipendenze EU-EU: mantenendo la EU in-order, è possibile eseguire la prima istruzione in coda se abilitata (ed eventualmente più di una purché in ordine rigidamente FIFO), realizzando una EU out-of-order cercando in coda una o più istruzioni abilitate in qualunque posizione, mantenendone l'ordine relativo.

Domanda 3 (OLD-1)

In LC le cause di commutazione di contesto corrispondono alle situazioni di sospensione di un processo nell'esecuzione di comandi *send* e *receive*.

Espandere l'implementazione della procedura (eseguita a interruzioni disabilitate) secondo i seguenti passi:

1. salvataggio dei registri generali e di IC (Rret) del processo running;
2. estrazione del primo PCB in lista pronti, facendo uso di appropriate tecniche per risolvere il problema delle strutture condivise riferite indirettamente;
3. ripristino dell'immagine dei registri generali e dell'immagine di IC, anch'essa in un registro generale;
4. valutazione delle informazioni da passare alla MMU, facendo uso di appropriate tecniche per risolvere il problema delle strutture condivise riferite indirettamente e di strutture dati del supporto del processo che deve passare in esecuzione;
5. esecuzione dell'istruzione START_PROCESS.