Architettura degli Elaboratori, 2011-12

Prima prova di verifica intermedia

19 dicembre 2011

Domanda 1

Una unità di elaborazione U contiene al suo interno un componente logica memoria M, di capacità 64K parole, con organizzazione interallacciata su quattro moduli, ognuno con tempo di accesso $5t_p$. U riceve in ingresso una parola X e un indirizzo INDIN di M, e invia in uscita un valore booleano PRES e un indirizzo INDOUT.

Tutte le parole di M hanno valore diverso. INDIN possiede la proprietà che INDIN *mod* 4 = 0.

Se il valore X è uguale a una delle parole del blocco di quattro parole avente indirizzo base INDIN, allora PRES assume il valore vero e INDOUT assume il valore dell'indirizzo della locazione avente contenuto uguale a X; altrimenti, PRES assume il valore falso e il valore di INDOUT non è significativo.

Scrivere il microprogramma e valutare il ciclo di clock, con il vincolo che il tempo di elaborazione di U sia uguale a un ciclo di clock.

Domanda 2

Per una generica unità di elaborazione, si indichino con S_{PC} e S_{PO} lo stato interno della Parte Controllo e della Parte Operativa rispettivamente. Dire se lo stato interno successivo dei due automi può essere espresso come:

$$S_{PO}(t+1) = f(S_{PC}(t), S_{PO}(t))$$

$$S_{PC}(t+1) = g(S_{PC}(t), S_{PO}(t))$$

con *f* e *g* funzioni opportune, e dimostrare la risposta.

Domanda 3

Si consideri la seguente computazione:

int X[N], Y[N], W[N], Z[N];

$\forall i = 0 .. N-1$: *if (X[i] = Y[i] then W[i] = X[i] + Y[i] else Z[i] = X[i] – Y[i]*

a) Determinarne - in funzione di N, del ciclo di clock e del tempo di accesso in memoria - il tempo di completamento per un elaboratore C1 avente macchina assembler D-RISC e per un elaboratore C2 avente macchina assembler D-RISC con in più la seguente istruzione rappresentata su due parole:

IF&CALC = Ra, Rb, Rc, Rd, Re, Rf, Rg, Rh

la cui semantica normale è:

if (MV[RG[Ra] + RG[Rb]] = MV[RG[Rc] + RG[Rd]])

then MV[RG[Ra] + RG[Rb]] + MV[RG[Rc] + RG[Rd]] $\rightarrow$ *MV[RG[Re] + RG[Rf]]*

else MV[RG[Ra] + RG[Rb]] $-$ *MV[RG[Rc] + RG[Rd]]* $\rightarrow$ *MV[RG[Rg] + RG[Rh]]*

b) Spiegare il contenuto dei file prodotti a tempo di compilazione per un sistema con memoria virtuale con paginazione.

Soluzione

Nota: *le domande di questa prova sono esattamente del tipo degli, e anche direttamente riconducibili agli, esercizi e quesiti delle raccolte 1 e 2, che sono stati fortemente raccomandati. In particolare:*

- *la Domanda 1 ricalca l'esercizio 6-3 della raccolta 1, applica quanto richiesto nei quesiti 5a e 5bis, ed usa un sottoinsieme dei concetti e tecniche applicate in qualunque esercizio e quesito sulla progettazione di unità, nonché di reti combinatorie;*
- *la Domanda 2 richiede capacità di spiegare e formalizzare concetti base, analogamente a quanto richiesto con i quesiti 1 e 3 della raccolta 1;*
- *la Domanda 3, che non pone alcuna difficoltà algoritmica e di compilazione, ricalca l'esercizio 5 della raccolta 2, e, nella parte b), richiede la spiegazione di concetti base di compilazione e creazione di processi di cui all'esercizio 1 della raccolta 2 stessa.*

Nella soluzione che segue, viene data la traccia essenziale, da completare con le altre spiegazioni sul tipo di quelle date negli esercizi delle raccolte di cui sopra.

Domanda 1

La memoria interallacciata data permette la lettura di un blocco di 4 parole in un singolo tempo di accesso. Ricordando la struttura di un indirizzo:

displ (14 bit)	mod (2 bit)

grazie alla proprietà INDIN *mod* 4 = 0, la lettura del blocco si ottiene indirizzando contemporaneamente tutti e quattro i moduli con *INDIN.displ*.

Il vincolo che l'operazione esterna sia eseguibile in un singolo ciclo di clock richiede che sia realizzabile come rete combinatoria la funzione *F* che prende in ingresso X e il blocco di indirizzo base INDIN e restituisce i valori *presenza*, *out.displ* e *out.mod* da scrivere in *PRES*, *INDOUT.displ* e *INDOUT.mod* rispettivamente. Questa funzione è ottenuta come segue:

$$\forall i = 0..3: \; x_i = OR(X \oplus M_i(INDIN.displ)$$

$$presenza = \overline{x_0\, x_1\, x_2\, x_3}$$

$$out.mod_0 = x_0\, \overline{x_1}\, x_2\, x_3 + \overline{x_0}\, x_1\, x_2\, x_3$$

$$out.mod_1 = x_0\, x_1\, \overline{x_2}\, x_3 + \overline{x_0}\, x_1\, x_2\, x_3$$

$$out.displ = INDIN.displ$$

Complessivamente la funzione *F*, inclusa la lettura dai moduli di memoria, ha ritardo uguale a *11* t_p.

Il microprogramma, di una sola microistruzione, è il seguente:

0. (RDYIN, ACKOUT = 0 0, 0 1, 1 0) nop, 0;
   (= 1 1) reset RDYIN, set ACKIN, reset ACKOUT, set RDYOUT,
   F(X, INDIN.displ, M_0[INDIN.displ], M_1[INDIN.displ], M_2[INDIN.displ], M_3[INDIN.displ])
   $\rightarrow$ *(PRES, INDOUT.displ, INDOUT.mod)*, 0

L'unità è realizzata come singola rete sequenziale, che si stabilizza nel tempo dato dal ritardo di stabilizzazione della funzione *F* (la ex-σ_{PO}), in quanto la produzione del segnale β per le interfacce in funzione dei valori RDYIN e ACKOUT (la ex-ω_{PC}) si sovrappone alla *F* stessa. Quindi:

$$\tau = T_F + \delta = 12\, t_p$$

Domanda 2

È certamente vero che le due funzioni dello stato interno successivo hanno nel loro dominio *esclusivamente* gli stati interni presenti di entrambi gli automi, in quanto, *essendo uno dei due automi (PO) di Moore*, il valore stabile degli stati interni presenti di PC e PO all'inizio di un generico ciclo di clock è tutto e solo quanto occorre per produrre il valore degli stati successivi di tali automi attraverso la loro interazione durante il ciclo di clock stesso. Questa è solo la spiegazione concettuale. La dimostrazione formale è la seguente:

indicando con Γ lo stato delle variabili di controllo e con X lo stato delle variabili di condizionamento, si ha:

$$S_{PO}(t+1) = \sigma_{PO}\big(S_{PO}(t), \Gamma(t)\big) = \sigma_{PO}\Big(S_{PO}(t), \omega_{PC}\big(S_{PC}(t), X(t)\big)\Big) = \sigma_{PO}\Big(S_{PO}(t), \omega_{PC}\big(S_{PC}(t), \omega_{PO}(S_{PO}(t))\big)\Big)$$

$$S_{PC}(t+1) = \sigma_{PC}\big(S_{PC}(t), \mathrm{X}(t)\big) = \sigma_{PC}\Big(S_{PC}(t), \omega_{PO}\big(S_{PO}(t)\big)\Big)$$

da cui l'asserto.

Domanda 3

a) Con le dovute spiegazioni sull'allocazione e inizializzazione dei registri generali, e applicando le regole di compilazione, il codice compilato per C1 è:

```
LOOP:   LOAD   RX, Ri, Rx
        LOAD   RY, Ri, Ry,
        IF =   Rx, Ry, THEN
        SUB    Rx, Ry, Rx
        STORE  RZ, Ri, Rx
        GOTO   CONT
THEN:   ADD    Rx, Ry, Rx
        STORE  RW, Ri, Rx
CONT:   INCR   Ri
        IF <   Ri, RN, LOOP
        END
```

il cui tempo di completamento (nel caso più sfavorevole che venga sempre eseguito il ramo *else* di 8 istruzioni) è:

$$T_{c1} = N\,(\,11\,t_a + 29\,\tau\,)$$

In particolare, degli 11 accessi in memoria per iterazione, 8 sono per la lettura di istruzioni e 3 per acceso ai dati. I 29 τ addizionali si ottengono facilmente considerando che 2 sono spesi in ogni chiamata istruzione e decodifica, 2 per ogni LOAD o STORE, 2 per ogni IF, e 1 per istruzioni aritmetiche corte e GOTO.

Per C2, l'istruzione IF&CALC è esattamente corrispondente all'elaborazione richiesta in ogni iterazione.

La compilazione è quindi:

```
LOOP:   IF&CALC   RX, Ri, RY, Ri, RW, Ri, RZ, Ri
        INCR   Ri
        IF <   Ri, RN, LOOP
        END
```

Il tempo di elaborazione di IF&CALC, rappresentata su due parole, è dato da $Tch = 2\tau + t_a$ più quattro accessi in memoria (ognuno dei quali costa $\tau + t_a$), il primo per leggere la seconda parola e gli altri 3 per i dati, più un ciclo di clock per il confronto, uno per l'operazione (+ o –) e uno per la fine scrittura e test interruzioni. Nel microprogramma, la chiamata della doppia parola può essere effettuata in parte in ch0, come per C1, e in parte all'inizio del microprogramma di esecuzione, come segue:

ch0. … IC → IND, …

ch1. … DATAIN → IR, …< decodifica >

if&calc0. IC + 1 → IND, 'read' → OPM, set RDYOUT, IC + 2 → IC, if&calc1

if&calc1. (RDYIN, or(ESITO) = …) …

(= 1 0) reset RDYIN, DATAIN → IR2, RG[IR.Ra] + RG[IR.Rb] → IND, 'read' → OPM, set RDYOUT, if&calc2

…

Quindi:

$$T_{IF\&CALC} = 5\ t_a + 9\ \tau$$

Da cui:

$$T_{c2} = N\ (7\ t_a + 16\ \tau)$$

Oltre al guadagno sul numero di accesi in memoria per la chiamata istruzione (nonostante IF&CALC sia rappresentata su doppia parola), c'è anche un guadagno sul numero di cicli di clock aggiuntivi dovuto alla parallelizzazione di microistruzioni nell'interprete di IF&CALC, tipicamente una richiesta di accesso in memoria viene fatta nello stesso ciclo di clock nel quale si conclude l'accesso precedente (per questo motivo un accesso in memoria costa $\tau + t_a$).

b) Si veda l'esercizio 1 della raccolta 2.

In particolare:

- il *file eseguibile* contiene una pagina per il codice, N/σ pagine per ognuno degli array X, Y, W, Z (con σ ampiezza di pagina), una pagina per PCB e una pagina per TABRIL del processo, lo spazio di n_{max} pagine per tutti i PCB del numero massimo di processi attivi, codice e strutture dati del supporto a tempo di esecuzione dei processi (primitive di cooperazione e relative strutture, scheduling a basso livello inclusi puntatori di accesso alla lista pronti, trattamento eccezioni, trattamento interruzioni e relative tabelle). Far notare quli strutture dati sono inizializzate a tempo di compilazione e quali verranno modificate in fase di creazione e caricamento;
- il *file di configurazione* contiene informazioni che servono per la creazione e il caricamento del processo, tra le quali: identificatore, indirizzo logico base e numero di pagine per ogni struttura dati condivisa e identificatori dei processi con i quali è condivisa; indirizzo base del PCB del processo; indirizzo base e numero di pagine di TABRIL; annotazioni sulle pagine da trasferire in memoria all'atto del caricamento; eventuali annotazioni per l'allocazione fisica in memorie locali di unità di I/O per implementare il Memory Mapped I/O.