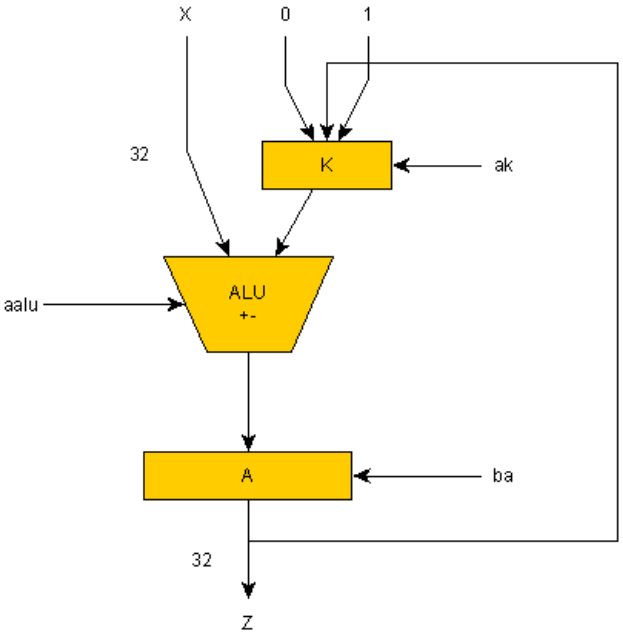
Architettura degli Elaboratori – A.A. 2017–2018 – Secondo appello
14 febbraio 2018

Indicare in alto a destra, su tutti i fogli consegnati, nome, cognome, matricola e corso di appartenenza (A o B). I risultati saranno pubblicati su WEB appena disponibili, insieme al calendario degli orali.

Domanda 1

Si considerino i componenti standard connessi come in figura.
Assumendo che la ALU sia caratterizzata da un ritardo di 5t, si indichi se la rete è una rete di Mealy o di Moore e successivamente:

- si determini il ciclo di clock di tale rete
- si identifichino i componenti in σ, ω ed R
- si indichino gli ingressi, le uscite e i possibili stati interni.



Domanda 2

Si consideri il seguente microcodice:

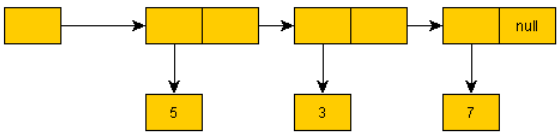
0. (RDY,OP=0-) nop, 0,
   (=10) nop, 1
   (=11) nop, 2
1. IN + R → R, reset RDY, set ACK, 0
2. (ACKout=0) nop, 2
   (=1) IN % 8 → OUT, set RDYout, reset ACKout, 3
3. IN – R → R, set ACK, reset RDY, 0

e si forniscano, con le dovute spiegazioni:

- la descrizione delle operazioni esterne implementate dall'unità firmware che ha il microcodice come programma di controllo
- il tempo di completamento (in funzione di τ)
- un'ottimizzazione del microcodice
- il miglioramento ottenuto in termini del tempo di completamento delle operazioni esterno.

Domanda 3

Si consideri una funzione che ricerca il minimo in una lista costruita a partire da elementi che contengono due puntatori, uno all'elemento successivo della lista ed uno alla cella di memoria che contiene il valore (intero positivo) dell'elemento della lista.



I parametri della funzione sono trasmessi (in ingresso e in uscita) mediante registri generali. Compilare lo pseudo codice in assembler D-RISC e discuterne le prestazioni (tempo di servizio) su un processore D-RISC pipeline standard.

Bozza di soluzione

Domanda 1

La rete sequenziale in figura è di Moore, visto che le uscite (contenuto del registro A, denotata come Z in Figura) dipendono solo dallo stato interno. La rete sequenziale di Moore può essere caratterizzata come segue:

- R è costituito dal solo registro A. Essendo da 32 bit, il numero degli stati interni è 2^{32}.
- La rete ω è nulla (funziona identità)
- La rete σ è costituita dal commutatore e dalla ALU. Ha come ingressi X (32 bit), ak (2 bit), aalu (1 bit) e ba (1 bit) oltre all'uscita del registro di stato interno A
- Gli ingressi sono X (32 bit), ak (2 bit), aalu (1 bit) e ba (1 bit)
- Le uscite sono costituite solo da A (32 bit)
- $T\omega = 0t_p$
- $T\sigma = t_k+t_{alu}$. Il commutatore da 4 ingressi (ne usiamo 3) ha 3 ingressi dati, tutti logicamente da 32 bit, e 2 bit di ingresso di controllo (ak). Al massimo testiamo un ingresso ed entrambi i bit del controllo, dunque serve un solo livello AND. La tabella della verità ha tre righe sole, quindi basta un unico livello di porte OR. Il tempo di stabilizzazione del commutatore è di $2t_p$. Dunqe $T\sigma = 5t_p + 2t_p = 7t_p$.
- $\tau = \max\{T\omega, T\sigma\} + \delta = 8t_p$.

Domanda 2

L'unità interagisce con il mondo esterno mediante due interfacce. Una in ingresso, da cui riceve un valore IN e un codice di operazione OP e una in uscita, su cui invia il contenuto di OUT. Entrambe le interfacce hanno propri indicatori a transizione di livello, dunque potrebbero essere due interfacce separate che lavorano con due unità, U_1 ed U_2, che, rispettivamente, forniscono i dati in ingresso e accettano i dati in uscita, oppure potrebbero interfacciare l'unità considerata verso un'unica unità che fornisce i dati in ingresso ed accetta quelli in uscita in modo indipendente.
Le operazioni esterne implementate sono due:

- OP=0: riceve IN e lo somma al registro interno R
- OP=1: riceve IN, sottrae il valore di R ad IN e memorizza il risultato in R ed invia il valore IN modulo 8 in uscita.

Il tempo di completamento delle due operazioni esterne, funzione di τ, è determinato dal numero di micro istruzioni eseguite (al netto delle micro istruzioni di attesa, nop), dunque:

- 2 micro istruzioni (0. e 1.) per OP=0, e
- 3 micro istruzioni (0. 2. e 3.) per OP=1

Assumendo come equiprobabili le due operazioni il tempo di completamento sarà dato da

$$T_c = p_1 (k_1\tau) + p_2 (k_2\tau) = 0.5 (2\tau) + 0.5 (3\tau) = \tau + 1.5 \tau = 2.5 \tau$$

Il codice può essere evidentemente ottimizzato considerando che :

- La seconda e la terza frase della prima microistruzione sono eliminabili, visto che non sono dovute ad interazioni con il mondo esterno. Possiamo eseguire le micro operazione della 1. nella prima seconda frase al posto della nop e saltare conseguentemente di nuovo alla 0. Possiamo fare il merge della 2. con la terza frase della 0. a patto di inserire ACKout nelle condizione testate dalla nuova 0.
- La micro operazioni della seconda frase della 2. e quelle della 3. non violano le considizioni di Bernestein (W(IN % 8 → OUT, set RDYout, reset ACKout) = {OUT, RDYout, ACKout}, R(IN – R → R, set ACK, reset RDY)={IN}, l'intersezione è vuota). Possiamo quindi unire la 2. e la 3. (prima o dopo il merge della 2. con la 0.)

Quindi il micro codice può essere riscritto in modi diversi.

Versione 1

0. (RDY,OP=0-) nop, 0,
   (=10) IN + R → R, reset RDY, set ACK, 0
   (=11) nop, 2
2. (ACKout=0) nop, 2
   (=1) IN % 8 → OUT, set RDYout, reset ACKout, 3
3. IN – R → R, set ACK, reset RDY, 0

In questo caso $Tc_1 = 1\tau$ e $Tc_2 = 3\tau$, $Tc=2\tau$

Versione 2

0. (RDY,OP, ACKout =0--) nop, 0,
   (=10-) nop, 1
   (=110) nop, 0
   (=111) IN % 8 → OUT, set RDYout, reset ACKout, IN – R → R, set ACK, reset RDY, 0
1. IN + R → R, reset RDY, set ACK, 0

In questo caso $Tc_1 = 2\tau$ e $Tc_2 = 1\tau$, $Tc=\frac{3}{2}\tau$

Versione 3

0. (RDY,OP, ACKout =0--) nop, 0,
   (=10-) IN + R → R, reset RDY, set ACK, 0
   (=110) nop, 0
   (=111) IN % 8 → OUT, set RDYout, reset ACKout, IN – R → R, set ACK, reset RDY, 0

In questo caso $Tc_1 = 1\tau$ e $Tc_2 = 1\tau$, $Tc=1\tau$

Domanda 3

Lo pseudo codice sarà una cosa del tipo:

```
cerca(iniziocoda) {
     p = iniziocoda;
     min = maxint;
     while(p!=null) {
          int val = mem[p.indirizzo];
          if(min > val) min = val;
          p = p.next;
     return(min);
}
```

Assumiamo di avere iniziocoda nel registro Rp, in Rret l'indirizzo di ritorno e di restituire in Rmin il valore cercato.

La funzione sarà chiamata da un codice tipo

```
MOV …,Rp
CALL Rcerca, Rret
MOV Rmin, …
```

La funzione potrebbe essere compilata nel codice D-RISC:

```
cerca:LOAD Rp, R0, Rpun          // riceve Rp,Rret, restituisce Rmin
      LOAD Rpun, R0, Rmin        // inizializza minimo
loop: IF= Rp, R0, fine           // se null ti fermi
      LOAD Rp, R0, Rind          // carica indirizzo val elem coda
```

```
      LOAD Rind, R0, Rval
      IF<= Rmin, Rval, cont
then: ADD Rval, R0, Rmin
cont: LOAD Rp, #1, Rp
      GOTO loop
fine: GOTO Rret
```

La simulazione del codice (relativamente ad una iterazione del while, non la prima ne' l'ultima) mette in evidenza la dipendenza fra le due LOAD, fra la seconda LOAD e la IF<= e la bolla da salto indotta dal GOTO di fine iterazione. La ADD nel ramo then non comporta differenze fra il caso then e l'else. Nel primo caso è decodificata dalla IU al ciclo 9 ed eseguita sulle EU nel ciclo 10. Nel secondo caso (else) viene scartata dalla IU al ciclo 9 e non arriva sulla EU.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
IM	IF=	LD	LD			IF<=			ADD	LD	GOTO	GOTO	IF=	
IU		IF=	LD	LD	LD	LD	IF<=	IF<=	IF<=	ADD	LD	GOTO	GOTO	IF=
DM				LD			LD					LD		
EU					LD			LD			(ADD)		LD	

Considerato che dovremmo eseguire 7 (caso *then*) o 6 (caso *else*) istruzioni per iterazione e invece utilizziamo 12t per eseguire le istruzioni il tempo di servizio sarà $Ts=\frac{12}{7}t$ o $Ts=\frac{12}{6}t=2t$ rispettivamente.