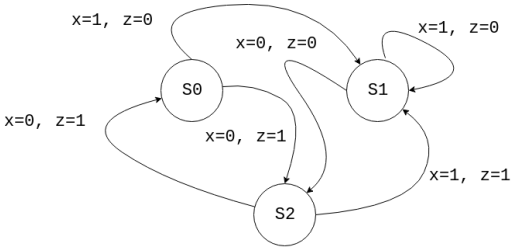
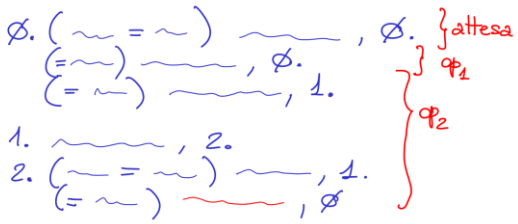
Architettura degli elaboratori – A.A. 2018-2019 – 1° aprile 2019

Riportare in alto a destra di ciascun foglio consegnato Nome, Cognome, numero di Matricola e corso di appartenenza.
I risultati e il calendario degli esami saranno pubblicati su WEB appena disponibili.

Domanda 1

Per l'automa in figura (sotto a sinistra) si fornisca la rete sequenziale che ne implementa le funzionalità, specificandone il tipo (Moore o Mealy) e le prestazioni.





Domanda 2

Si consideri un microcodice con 3 microistruzioni (0. 1. e 2.), che rappresenta il programma di controllo di una unità firmware in grado di eseguire 2 distinte operazioni esterne: op_1 e op_2. La prima microistruzione è una istruzione condizionale con 3 frasi, che utilizza variabili di condizionamento complesse. La prima delle tre frasi attende che venga richiesta un'operazione, la seconda implementa completamente op_1. La terza frase prepara i dati per il calcolo di op_2, che avviene mediante le istruzioni 1. e 2. L'istruzione 1. non prevede test di variabili di condizionamento. L'istruzione 2. è ancora una istruzione condizionale con due frasi: la prima rimanda alla istruzione 1., mentre la seconda conclude op_2 e ritorna all'istruzione 0 (vedi figura in alto a destra).

Si discuta sotto quali condizioni potrebbe risultare conveniente trasformare il microcodice in modo da eliminare le variabili di condizionamento complesse dalla prima microistruzione in funzione:

- del numero k_i di microistruzioni eseguite per op_i (ricordando che $k_1 = 1$) e della probabilità che venga richiesta una certa operazione esterna (p_i),
- dei tempi della parte controllo $T_{\sigma PC}$ e $T_{\omega PC}$
- dei tempi $T_{\omega PO-0}$ e $T_{\omega PO-2}$ necessari a calcolare le variabili di condizionamento delle microistruzioni 0. e 2., e dei tempi $T_{\sigma PO-ij}$ necessari per eseguire ciascuna delle frasi delle tre microistruzioni (con *i* numero di microistruzione e *j* numero della frase nella microistruzione).

Domanda 3

Si fornisca la compilazione in Assembler D-RISC del seguente pseudo-codice e se ne valutino le prestazioni (tempo di completamento) su un processore D-RISC standard con unità slave a due stadi per la moltiplicazione fra interi, assumendo che il passaggio dei parametri da e per la procedura avvenga utilizzando un'area di memoria.

Main	**Funzione**

`X1 = F(a,b,n);`
`Y1 = F(c,d,n);`
`a[0] = X1+Y1;`
`END`
	`int F(int x[], int y[], int n) {`
	`  int sum = 0;`
	`  for(int i=0; i<n; i++) sum+=x[i]*y[i]; return sum;`
	`}`

Bozza di soluzione

Domanda 1

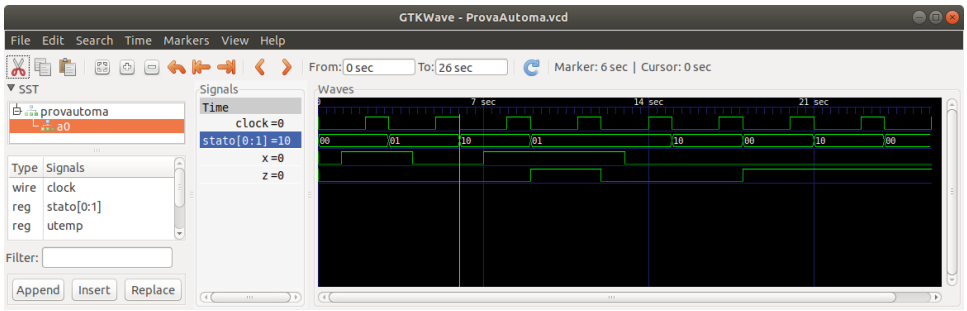
L'automa è di Mealy visto che le uscite dipendono sia dagli ingressi che dallo stato interno. È implementato mediante una rete di Mealy con un registro di stato da 2 bit (00=S0, 01=S1 e 10=S2) con le due componenti ω e σ definite dalle seguenti tabelle di verità (ne riportiamo una sola, la colonna delle uscite (Z) definisce la tabella di verità della ωPC e quella relativa ai nuovi valori del registro di stato interno (S_1' S_2') definisce quella della σPC:

S_0	S_1	X		Z		S_1'	S_0'
0	0	0		1		1	0
0	0	1		0		0	1
0	1	0		1		0	1
0	1	1		0		1	0
1	0	0		1		0	0
1	0	1		0		0	1

Tre ingressi per il livello AND e al massimo 3 ingressi per il livello OR fanno sì che entrambe le reti abbiamo un ritardo di $2t_p$. Questo comporta un ciclo di clock pari a $3t_p$, dal momento che

$$\tau = \max\{T_\omega, T_\sigma\} + \delta$$

Il comportamento dell'automa su una sequenze di input è esemplificato dalla seguente traccia, ricavata programmando l'automa in Verilog e testandone il funzionamento con la sequenza di ingresso: 1 0 1 1 0 0 0 0 …



Domanda 2

La lunghezza del ciclo di clock del microprogramma come descritto è il massimo delle lunghezze necessarie per eseguire le varie frasi del microcodice. Li riportiamo in tabella:

Microistruzione	Frase	Tempo di stabilizzazione
0	1	$T_{\omega PO\text{-}0} + T_{\omega PC} + T_{\sigma PO\text{-}01} + \delta$
0	2	$T_{\omega PO\text{-}0} + T_{\omega PC} + T_{\sigma PO\text{-}02} + \delta$
0	3	$T_{\omega PO\text{-}0} + T_{\omega PC} + T_{\sigma PO\text{-}03} + \delta$

1	1	$0 + T_{\omega PC} + T_{\sigma PO-11} + \delta$
2	1	$T_{\omega PO-2} + T_{\omega PC} + T_{\sigma PO-21} + \delta$
2	2	$T_{\omega PO-2} + T_{\omega PC} + T_{\sigma PO-22} + \delta$

Se spezziamo la 0 in modo da avere due microistruzioni, una (0.0.) che calcola le variabili di condizionamento complesse (magari dopo aver testato il RDY che segnala la presenza di una richiesta di operazione) e una (0.1.) che testa i valori così calcolati, otterremmo una tabella come quella che segue:

Microistruzione	Frase	Tempo di stabilizzazione
0.0	1	$0 + T_{\omega PC} + T_{\omega PO-0} + \delta$
0.1	1	$0 + T_{\omega PC} + T_{\sigma PO-01} + \delta$
0.1	2	$0 + T_{\omega PC} + T_{\sigma PO-02} + \delta$
0.1	3	$0 + T_{\omega PC} + T_{\sigma PO-03} + \delta$
1	1	$0 + T_{\omega PC} + T_{\sigma PO-11} + \delta$
2	1	$T_{\omega PO-2} + T_{\omega PC} + T_{\sigma PO-21} + \delta$
2	2	$T_{\omega PO-2} + T_{\omega PC} + T_{\sigma PO-22} + \delta$

dal momento che il lavoro che prima si faceva nel $T_{\omega PO}$ della prima microistruzione è stato spostato nel $T_{\sigma PO}$ dell'istruzione 0.0 aggiunta, mentra la ex 0. è diventata la 0.1. che testa il registro in cui è stato scritto nella 0.0 il valore della (ex) variabile di condizionamento complessa, e quindi ha ora un $T_{\omega PO}$ nullo e il $T_{\sigma PO}$ originale.

Dobbiamo considerare due casi, a seconda di dove troviamo il massimo valore del ciclo di clock nel programma originale:

- il massimo della terza colonna è in una delle prime tre posizioni della prima tabella. In questo caso il ciclo di clock diminuisce.
  Supponiamo che la diminuzione sia pari a Δ. Va considerato che il numero totale di istruzioni eseguite per la op_2 rimane invariato mentre il numero di quelle eseguite per la op_1 raddoppia. Quindi prima avremmo avuto ($p_1 + p_2 k_2$) istruzioni ciascuna eseguita in un τ, e adesso avremmo ($2 p_1 + p_2 k_2$) istruzioni che costerebbero ciascuna $\tau - \Delta$. Quindi il guadagno avviene per la seconda operazione esterna (ed è pari a $k_2\Delta$) mentre la perdita è per la prima operazione esterna (ed è pari a $2(\tau - \Delta) - \tau = \tau - \Delta$). Se $p_2 k_2 \Delta > p_1 (\tau - \Delta)$ abbiamo un guadagno, altrimenti abbiamo una perdita e la modifica non conviene.
- Il massimo è in una delle ultime tre righe. In questo caso il ciclo di clock non diminuisce ma aumenta il numero di microistruzioni eseguite per la op1 e dunque la cosa non conviene in nessun caso.

Domanda 3

La procedura sarà compilata come segue, assumendo che l'indirizzo dell'area di memoria per il passaggio dei parametri e per l'indirizzo di ritorno siano in Rm e Rr e che nell'area di memoria ad offset consecutivi a partire da 0 si trovino l'indirizzo base di X, quello di Y, l'intero N e la locazione che serve a contenere il risultato:

```
f:    LOADi Rm, #0, Rbasex
      LOADi Rm, #1, Rbasey
      LOADi Rm, #2, Rn
      CLEAR Ri
      CLEAR Rsum
Loop: LOAD Rbasex, Ri, Rxi
      LOAD Rbasey, Ri, Ryi
      MUL Rxi, Ryi, Rtemp
      ADD Rsum, Rtemp, Rtemp
      INC Ri
      IF< Ri, Rn, loop
      STOREi Rm, #3, Rsum
      GOTO Rr
```

Il codice fornito è compilato invece come segue:

```
      STOREi Rm, #0, RbaseA
      STOREi Rm, #1, RbaseB
      STOREi Rm. #2, Rn
      CALL   Rf, Rr
      LOADi  Rm, #3, Rx1
      STOREi Rm, #0, RbaseC
      STOREi Rm, #1, RbaseD
      CALL   Rf, Rr
      LOADi  Rm, #3, Ry1
      ADD    Rx1, Ry1, Rtemp
      STORE  RbaseA, R0, Rtemp
      END
```

L'esecuzione del codice avviene come segue:

- la prima chiamata della procedura richiede 4t più una bolla "da salto" di 1t.

		Prima chiamata					
IM	ST	ST	ST	CALL		LOAD	
IU		ST	ST	ST	CALL		LOAD
DM			ST	ST	ST		
EU master							
EU */ int							

- L'esecuzione del preambolo della funzione (fuori dal loop) richiede 5t (senza bolle)

		Preambolo							
IM	LOAD	LOAD	LOAD	CLR	CLR	LOAD	LOAD	MUL	A
IU		LOAD	LOAD	LOAD	CLR	CLR	LOAD	LOAD	M
DM			LOAD	LOAD	LOAD			LOAD	L
EU master				LOAD	LOAD	LOAD	CLR	CLR	L
EU */ int									

- L'esecuzione di una singola iterazione richiede 8t, avendo avuto cura di invertire la ADD (che dipende dal risultato della MUL (EU-EU) con la INC, che crea una dipendenza sulla IF< di fine ciclo

		Prima iterazione (e successive,esclusa l'ultima)								
IM	LOAD	LOAD	MUL	INC	ADD	IF<			LOAD	
IU		LOAD	LOAD	MUL	INC	ADD	IF<	IF<		LOAD
DM			LOAD	LOAD						
EU master				LOAD	LOAD	MUL	INC	ADD	ADD	INC
EU */ int							MUL	MUL		

		Prima iterazione (e successive,esclusa l'ultima)								
IM	LOAD	LOAD	MUL	INC	ADD	IF<			LOAD	
IU		LOAD	LOAD	MUL	INC	ADD	IF<	IF<		LOAD
DM			LOAD	LOAD						
EU master				LOAD	LOAD	MUL	INC	ADD	ADD	INC
EU */ int							MUL	MUL		

		Prima iterazione (e successive,esclusa l'ultima)								
IM	LOAD	LOAD	MUL	INC	ADD	IF<			LOAD	
IU		LOAD	LOAD	MUL	INC	ADD	IF<	IF<		LOAD
DM			LOAD	LOAD						
EU master				LOAD	LOAD	MUL	INC	ADD	ADD	INC
EU */ int							MUL	MUL		

In realtà, la prima iterazione ha un *t* in più, legato al fatto che c'è una dipendenza fra la CLR Ri e la LOAD RbaseX, Ri, Rxi. Dunque al conto totale va aggiunto un *t*.

- L'ultima iterazione richiede 10t (dipendenza della STORE con la MUL) + 1t di bolla di "salto preso" per il ritorno

		Ultima iterazione									Seconda chiamata					
IM	LOAD	LOAD	MUL	INC	ADD	IF<		ST	GOTO		LOAD	ST	ST	CALL		LOAD
IU		LOAD	LOAD	MUL	INC	ADD	IF<	IF<	ST	GOTO		LOAD	ST	ST	CALL	
DM			LOAD	LOAD						ST			LOAD	ST	St	
EU master				LOAD	LOAD	MUL	INC	ADD	ADD	INC				LOAD		
EU */ int							MUL	MUL								

- La seconda chiamata richiede 4t + 1t di bolla "salto preso"

- Come sopra la funzione richiede 5t + (N-1)*8t + 10t + 1t ovvero 5t (preambolo) + N*8t (iterazioni) +2t (extra ultima iterazione) = 7t + 8Nt

- L'ultima parte del codice chiamante richiede 6t (2t di bolla per la dipendenza IU-EU ADD-STORE)

		Fine main					
IM	LOAD	ADD	ST			END	
IU		LOAD	ADD	ST	ST	ST	END
DM			LOAD				ST
EU master				LOAD	ADD		
EU */ int							

In totale quindi avremmo:

2*(4t + 1t) (doppia chiamata) + 6t (fine main) + 2*(7t+8Nt) + t (dip prima iterazione con CLR) = 31t + 16Nt

A fronte di

- 12 istruzioni del "main"
- 2*(5 + 6*N + 2) istruzioni della funzione

Per un totale di (12+14) + 12N = 26+12N istruzioni.

Calcolando il tempo di servizio sul solo ciclo interno della funziona (che è la parte che pesa di più) abbiamo t = 8t / 6 = 4t/3 con una efficienza di pari a ¾.

Invece come tempo di servizio totale avremmo T = (31+16N)t / (26+12N). Per N grande la parte significativa è quella moltiplicata per N, che è sempre quella di prima e dunque T = 4t/3. Per N piccolo le cose cambiano leggermente. Per esempio, per vettori lunghi 2 abbiamo un T = 63t/50 che è ca. 1.24 t contro l'1.33 t del ciclo interno.