

**PROVA SCRITTA DEL CORSO DI**  
**CALCOLATORI ELETTRONICI**  
**NUOVO ORDINAMENTO DIDATTICO**  
 2 Luglio 2007

**MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI**

**ESERCIZIO 1 (8 punti)**

- (5 punti) Spiegare il funzionamento del flip-flop T sincrono (di tipo master-slave) e definire la sua tabella di transizione degli stati. Si indichino tutte le porte logiche ed i collegamenti.
- (3 punti) Mostrare come derivare un flip flop T sincrono (di tipo master-slave) da un flip flop JK sincrono (di tipo master-slave), a partire dalla tabella di flusso di quest'ultimo.

**Soluzione**

Il FF-T commuta il valore dello stato in esso memorizzato in presenza di un fronte di discesa (o di salita) del segnale in ingresso. Essendo un dispositivo master-slave, la commutazione viene riportata in uscita solo quando il livello del segnale ritorna basso.

La tabella di eccitazione del FF-T è fornita in figura. Dalla sintesi si ha:

T	Q	Q'
0	0	0
0	1	1
1	0	1
1	1	0

T	Q	Q'
0	0	1
0	1	1
1	0	1
1	1	0

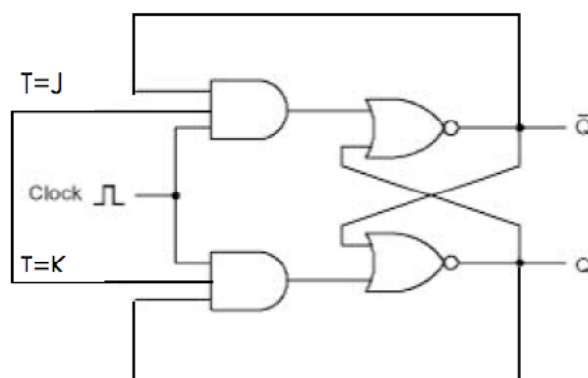
$$Q' = T\bar{Q} + \bar{T}Q$$

$$Q(t + \tau) = (S + Q(t))\bar{R}$$

$$S = T\bar{Q}$$

$$R = TQ$$

$$Q(t + \tau) = (T\bar{Q} + Q)(\bar{T} + \bar{Q}) = T\bar{Q} + \bar{T}Q$$



In figura mostriamo il cosiddetto "latch" T, che opera su livello di segnale anziché su fronte di salita (non funziona quindi come elemento stabile e ritardante). Per ottenere il FF-T è necessario connettere ad un latch T, che fungerà da Master, un secondo latch T, sincronizzato mediante inverter con il primo attraverso il segnale clock. In questo modo, lo stato viene memorizzato dal primo latch quando il livello di sincronizzazione è alto. Sul fronte di discesa del segnale, grazie all'inverter, l'uscita del primo latch viene propagata sul secondo, senza pericolo che possa essere alterata dal livello di segnale in ingresso al primo. Il secondo latch funzionerà quindi da Slave, riportando semplicemente in uscita il valore dello stato precedentemente memorizzato dal Master, in modo stabile, con un ritardo complessivo di un colpo di clock.

Il passaggio da un flip flop JK a un flip flop T è banalmente  $T=J=K$ . Per capirlo è sufficiente confrontare i valori dell'uscita (lo stato successivo  $Q'$ ) dalle tabelle di eccitazione dei due flip flop, che sono esattamente uguali quando  $T=0$  e  $J=K=0$  e quando  $T=1$  e  $J=K=1$ , da cui deriva la relazione  $T=J=K$ .

### ESERCIZIO 2 (8 punti)

- (4 punti) Sia dato un disco con le seguenti caratteristiche: velocità di rotazione pari a 6000 giri al minuto, tempo necessario alla testina per spostarsi da una traccia alla successiva uguale a 0.5 ms, settori da 1 KB, 200 settori per traccia. Calcolare il tempo medio di lettura di un file da 10 KB sapendo che la testina si trova inizialmente in un punto qualunque del disco e che la distanza media tra due settori successivi del file è pari a 2 tracce.
- (4 punti) Utilizzando il tempo medio di accesso al disco calcolato al punto precedente, si calcoli l'hit ratio di cache minimo ( $H_c$ ), relativo ad una gerarchia a 3 livelli (cache, memoria primaria, disco), affinché il tempo medio di accesso alla gerarchia sia 30 ms. Siano dati i seguenti altri dati:  
 $H_p = 0.8$ ;  $T_p = 20$  ms;  $T_c = 10$  ms, dove  $H_p$  è l'hit ratio della memoria primaria,  $T_p$  il tempo medio di accesso alla memoria primaria e  $T_c$  il tempo medio di accesso alla memoria cache.

### Soluzione

- $TROT = 60 / 6000 = 10$  ms  
 $TLAT = TROT / 2 = 5$  ms (tempo di latenza)  
 $Tlett = TROT / 200 = 50$   $\mu$ s (tempo di lettura di un settore)  
 $Tpos = 0.5 \times 2 = 1$  ms  
 Numero di settori richiesti per leggere 10 KB:  $10KB/1KB = 10$   
 Tempo di lettura del file da 10 KB:  
 $= 10 * (TLAT + TPOS + Tlett) = 10 * (5 \text{ ms} + 1 \text{ ms} + 50 \mu\text{s}) \approx 60 \text{ msec}$
- Per calcolare  $H_c$  basta esplicitarlo dalla formula:

$$\bar{T} = H_c T_c + (H_p - H_c)(T_p + T_c) + (1 - H_p)(T_d + T_p + T_c)$$

$$H_c = (T_c + T_p + T_d - H_p T_d - \bar{T}) / T_p = 0.6$$

### ESERCIZIO 3 (10 punti)

Si scriva una funzione Assembler MIPS che implementi la seguente funzione C:

```
int azione(int k, int a, int b, int* c)
{
    switch (k)
    {
        case 1: return a*b; /* moltiplica */
        case 2: return media(a,b); /* richiama media */
        case 3: return c[k]; /* estrai vettore */
    }
}
```

Si utilizzi il vettore di appoggio `jumpTable` visto a lezione (Cap. 6, pag. 26), in cui vengono localizzati gli indirizzi a cui saltare nei vari casi. Il vettore `jumpTable` è un'area di memoria contigua contenente l'indirizzo della prima istruzione da eseguire in funzione dei casi indicati dalla `switch`. Nel codice Assembler questi indirizzi possono essere espressi in forma simbolica (es. si può usare l'etichetta `moltiplica` per esprimere l'indirizzo della prima istruzione da eseguire in case 1). Nel presente esercizio, `jumpTable` sarà quindi caratterizzato come segue:

Indirizzo	Contenuto
<code>jumpTable(0)</code>	indirizzo della prima istruzione relativa a case 1 (es. etichetta <code>moltiplica</code> )
<code>jumpTable(4)</code>	indirizzo della prima istruzione relativa a case 2 (es. etichetta <code>richiama_media</code> )
<code>jumpTable(8)</code>	indirizzo della prima istruzione relativa a case 3 (es. etichetta <code>estrai_vettore</code> )

Nello scrivere la funzione, si consideri che:

- a) `k`, `a`, `b` e l'indirizzo iniziale del vettore `c` sono memorizzati rispettivamente in `$4`, `$5`, `$6`, `$7`;
- b) il valore restituito dev'essere memorizzato in `$7`;
- c) la funzione `media`, esistente, prevede che `a` venga memorizzato in `$4`, `b` in `$5`, ed il valore restituito figuri in `$6`.

**N.B. Tradurre il codice C in modo corretto (es. immaginando di sostituire `switch` con catena `if-else`) ma senza uso di `jumpTable` darà luogo ad un punteggio complessivo massimo di 7 punti.**

#### Soluzione

Nella implementazione della funzione `azione`, che proponiamo, utilizziamo i registri `$9` per contenere il valore `jumpTable[k*4]`, e `$10` per contenere `k*4`.

```
azione:    addi $29, $29, -12
           sw $9, 0($29)
           sw $10, 4($29)
           sw $31, 8($29)
           muli $10, $4, 4
           lw $9, jumpTable($10)
           jr $9
rientro:   lw $9, 0($29)
           lw $10, 4($29)
           lw $31, 8($29)
           addi $29, $29, 12
           jr $31
```

<code>moltiplica:</code>	<code>mul \$7, \$5, \$6</code> <code>j rientro</code>	<code>richiama_media:</code>	<code>move \$4, \$5</code> <code>move \$5, \$6</code> <code>jal media</code> <code>move \$7, \$6</code> <code>j rientro</code>
<code>estrai_vettore:</code>	<code>add \$10, \$7, \$10</code> <code>lw \$7, 0(\$10)</code> <code>j rientro</code>		

#### ESERCIZIO 4 (7 punti)

Si consideri un calcolatore in cui la CPU esegue  $10^5$  istruzioni/s. L'esecuzione di una istruzione richiede 5 cicli di clock, 3 dei quali tengono occupato il bus di sistema. Si ipotizzi che il 75% dell'Instruction Rate sia usato dalla CPU per eseguire programmi che non contengono trasferimenti di I/O. L'ampiezza della linea dati del bus è pari a 32 bit.

Si consideri il caso in cui il trasferimento dei dati avvenga mediante IO da programma, con le seguenti 4 istruzioni:

- 1) LOAD parola dalla periferica al registro CPU
  - 2) STORE parola da registro CPU a memoria
  - 3) generazione indirizzo di memoria successivo
  - 4) conteggio dati da trasferire.
- a) (3 punti) Calcolare la massima frequenza di trasferimento dati ottenibile (espressa in kB/s) fra una periferica collegata al bus di sistema e la memoria principale mediante I/O da programma, sapendo che una parola è pari a 32 bit.
- b) (4 punti) Calcolare la massima frequenza di trasferimento dati ottenibile (espressa in kB/s) nel caso in cui si usi la modalità "transparent" DMA. Si ipotizzi che una operazione di lettura/scrittura della memoria richieda un ciclo di clock e che una parola sia pari a 32 bit.

#### Soluzione

- a) Nel caso di trasferimento mediante I/O da programma, per trasferire una parola occorrono 4 istruzioni. La CPU è impegnata per il 75% del tempo a eseguire istruzioni che non coinvolgono l'I/O, dunque può usare solo il 25% del tempo per eseguire istruzioni di trasferimento dati con periferiche. In termini di istr./sec questo tempo è pari a  $0.25 \times 10^5 \text{ istr./s} = 2.5 \times 10^4 \text{ istr./s}$ . Dal momento che per trasferire una parola servono quattro istruzioni, la velocità di trasferimento è pari a:

$2.5 \times 10^4 \text{ istr./s} / (4 \text{ istr./parola}) = 6250 \text{ parole/s}$ . La dimensione di una parola è pari a 32 bit (4 byte), da cui si ricava la velocità di trasferimento di **24.41 kB/s**,

- b) Nel caso di 'transparent DMA' posso trasferire i dati tutte le volte che il bus di sistema è libero. Nel caso in esame questo tempo è pari alla somma del 25% del tempo lasciato libero dall'esecuzione di istruzioni che non coinvolgono I/O, più i due cicli/istruzione in cui il bus è libero. Pertanto durante il 75% del tempo posso trasferire una parola/istr.:

$0.75 \times 2 \text{ parole/istr} \times 10^5 \text{ istr./s} = 1.5 \times 10^5 \text{ parole/s}$

Nel restante 25% del tempo posso trasferire 5 parole/istr.:

$0.25 \times 5 \text{ parole/istr.} \times 10^5 \text{ istr./s} = 1.25 \times 10^5 \text{ parole/s}$

In **totale**, nel caso di trasferimento con DMA la velocità totale di trasferimento è pari a:  **$(1.5 + 1.25) \times 10^5 \text{ parole/s} = 2.75 \times 10^5 \text{ parole/s} = 1074.22 \text{ kB/s}$**