

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO E VECCHIO ORDINAMENTO DIDATTICO**
28 Settembre 2005

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

ESERCIZIO 1 (NO: 8 punti – VO: 7 punti)

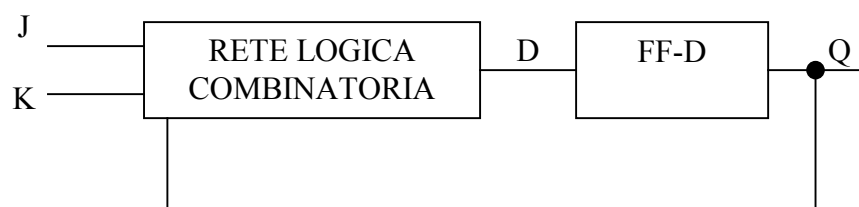
- 1) (3 punti) Spiegare in modo chiaro e sintetico cosa sono ed a cosa servono i flip flop nella progettazione di una rete sequenziale sincrona.
- 2) (NO: 5 punti – VO: 4 punti) Ricavare da un FF-D un FF-JK mediante un'opportuna rete logica.

Soluzione.

1) Vedi dispense del corso (cap.2, pp. 72 e segg.).

2)

E' sufficiente connettere una rete logica all'ingresso del FF-D, secondo lo schema seguente:



J	K	Q	Q'	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

JK \ Q	00	01	11	10
0			1	1
1	1			1

$$D = J\bar{Q} + \bar{K}Q$$

ESERCIZIO 2 (NO: 9 punti – VO: 7 punti)

La memoria di un calcolatore è organizzata con una gerarchia a due livelli cache-primaria (gli indirizzi sono espressi in esadecimale).

- 1) (NO: 4 punti – VO: 2 punti) Si ricavi, dalla sequenza di accessi mostrati in figura, l'hit ratio di cache e l'hit ratio di primaria.

Byte	Recuperato in cache
0F	Sì
AA	
10	
A1	Sì
FA	
90	Sì
65	
80	
B1	Sì
CC	Sì

- 2) (NO: 5 punti – VO: 3 punti) Sapendo che il tempo di accesso alla cache è pari a 4 ns, e quello alla primaria è pari a 40 ns, calcolare il tempo medio di accesso alla gerarchia a due livelli.
- 3) (solo VO: 2 punti) Enunciare e spiegare in modo chiaro e sintetico il principio di località.

Soluzione.

- 1) Dalla definizione di gerarchia di memoria a due livelli, i byte recuperati in cache devono trovarsi anche in primaria, per cui $H_p = 1$.
Dei dieci accessi in tabella, solo cinque sono 'hit' in cache, per cui $H_c = 0.5$.
- 2) Applicando la formula della gerarchia di memoria a due livelli, si ha:
 $T_m = T_c + (1 - H_c) * T_p = 4 + 0.5 * 40 = 24$ ns.
- 3) Vedi dispense del corso.

ESERCIZIO 3 (solo NO: 8 punti)

Implementare in Assembler MIPS una funzione che, dati l'indirizzo iniziale di un vettore v (in \$4) e due indici i e j (rispettivamente in \$5 e in \$6), scambi $v(i)$ con $v(j)$ solo se $v(i) <> v(j)$. Vincolo: si usi il minimo numero di registri, senza preoccuparsi di preservare i valori contenuti nei registri destinati ai parametri della funzione eccetto che per il contenuto di \$4.

Soluzione.

$\$9 \leftarrow v[i]; \$10 \leftarrow v[j]$

```
funzione:  addi $29, $29, -8
           sw $9, 0($29)
           sw $10, 4($29)
           muli $5, $5, 4
           add $5, $5, $4          #indirizzo di v[i] in $5
           muli $6, $6, 4
           add $6, $6, $4          #indirizzo di v[j] in $6
           lw $9, 0($5)
           lw $10, 0($6)
           beq $9, $10, exit      #if v[i]==v[j] exit
           sw $9, 0($6)
           sw $10, 0($5)
exit:      lw $9, 0($29)
           lw $10, 4($29)
           addi $29, $29, 8
           jr $31
```

ESERCIZIO 3 (solo VO: 6 punti)

I trasferimenti di parole a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa di 12 bit 001001101110 (il bit meno significativo è a sinistra), risultata della codifica di una parola di N bit secondo il codice di Hamming. **Spiegando bene ogni passo del ragionamento:**

- 1) (2 punti) calcolare N, supponendo di aver fatto uso del numero minimo di bit di controllo necessario per una stringa di 12 bit;
- 2) (1 punto) scrivere la parola di N bit a partire dalla stringa data;
- 3) (3 punti) indicare eventuali errori nella stringa codificata, specificando quale dei bit è stato alterato.

Soluzione.

- 1) Deve venire rispettata la condizione:

$$2^K \geq N + K + 1 \quad (1),$$

dove K è il numero di bit di controllo inseriti. Essendo $N + K = 12$, si evince dalla (1) che il numero minimo di bit di controllo richiesto è 4. Da cui $N = 8$.

- 2) Nella codifica di Hamming, la sequenza in ingresso presenta la seguente struttura:

c_0	c_1	b_0	c_2	b_1	b_2	b_3	c_3	b_4	b_5	b_6	b_7
0	0	1	0	0	1	1	0	1	1	1	0

Dove $c_0 \dots c_3$ sono i quattro bit costituenti il vettore di controllo, e $b_0 \dots b_7$ gli otto bit trasmessi. La sequenza ricevuta è 10111110.

- 3) Per verificare la presenza di un errore, dobbiamo ricalcolare il vettore di controllo a partire dalla sequenza ricevuta. Si ha:

$$c'_0 = b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0$$

$$c'_1 = b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 1$$

$$c'_2 = b_1 \oplus b_2 \oplus b_3 \oplus b_7 = 0$$

$$c'_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 1$$

Il passo successivo è calcolare il vettore di errore dato dalla differenza dei vettori di controllo c e c' (ricordiamo che somma e differenza tra bit producono lo stesso risultato):

$$e_0 = c_0 \oplus c'_0 = 0$$

$$e_1 = c_1 \oplus c'_1 = 1$$

$$e_2 = c_2 \oplus c'_2 = 0$$

$$e_3 = c_3 \oplus c'_3 = 1$$

Poiché il vettore risultante 1010 non è nullo, vi è un errore nella stringa di 12 bit data e precisamente nella posizione indicata dal vettore di errore tradotto in notazione decimale. Il bit sbagliato è quindi il decimo (b_5), e la parola corretta è 10111010.

ESERCIZIO 4 (NO: 8 punti – VO: 7 punti)

Sia data la seguente lista di processi:

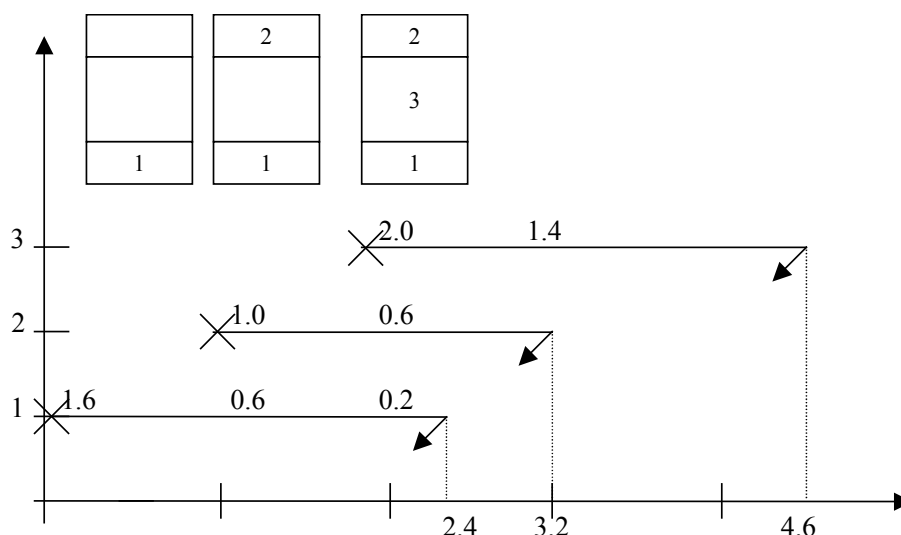
Processo	Tempo di arrivo	Tempo di CPU	Memoria
1	0.0	1.6	120K
2	1.0	1.0	50K
3	1.8	2.0	300K

Testo NO:

La memoria è gestita con partizioni statiche nell'ordine di 100K, 500K e 150K con strategia di allocazione Best Fit.

- (NO: 6 punti – VO: 5 punti) Mostrare, utilizzando il metodo grafico, la sequenza di esecuzione dei processi qualora si impieghi la politica di scheduling FIFO multiprogrammata, indicando in quale partizione i processi vengono allocati (memoria inizialmente vuota).
- (2 punti) Calcolare il tempo di *turnaround* medio e il tempo di *turnaround* pesato medio.

Grafico processo-tempo con politica FIFO multiprogrammata e allocazione Best Fit (gli indirizzi di memoria partono dal basso):



Processo	Arrivo	Fine	Turnaround	WT
1	0.0	2.4	2.4	1.5
2	1.0	3.2	2.2	2.2
3	1.8	4.6	2.8	1.4
Media			2.5	1.7

ESERCIZIO 5 (solo VO: 5 punti)

Spiegare in modo chiaro e sintetico le differenze fra un'architettura CISC e un'architettura RISC.