

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

11 Gennaio 2008

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

ESERCIZIO 1 (NO: 10 punti – VO: 8 punti)

Progettare una rete sequenziale che ricevendo in ingresso una sequenza di bit X , produca un'uscita Z posta a 1 solo in corrispondenza della sequenza 0101 oppure 1110.

1. (NO: 6 punti – VO: 5 punti) disegnare il grafo degli stati, la tabella di flusso, e la tabella delle transizioni con l'utilizzo di flip flop T;
2. (NO: 4 punti – VO: 3 punti) minimizzare le funzioni di transizione dello stato attraverso le mappe di Karnaugh. Indicare anche l'espressione algebrica dell'uscita Z .

Soluzione

Grafo degli stati:

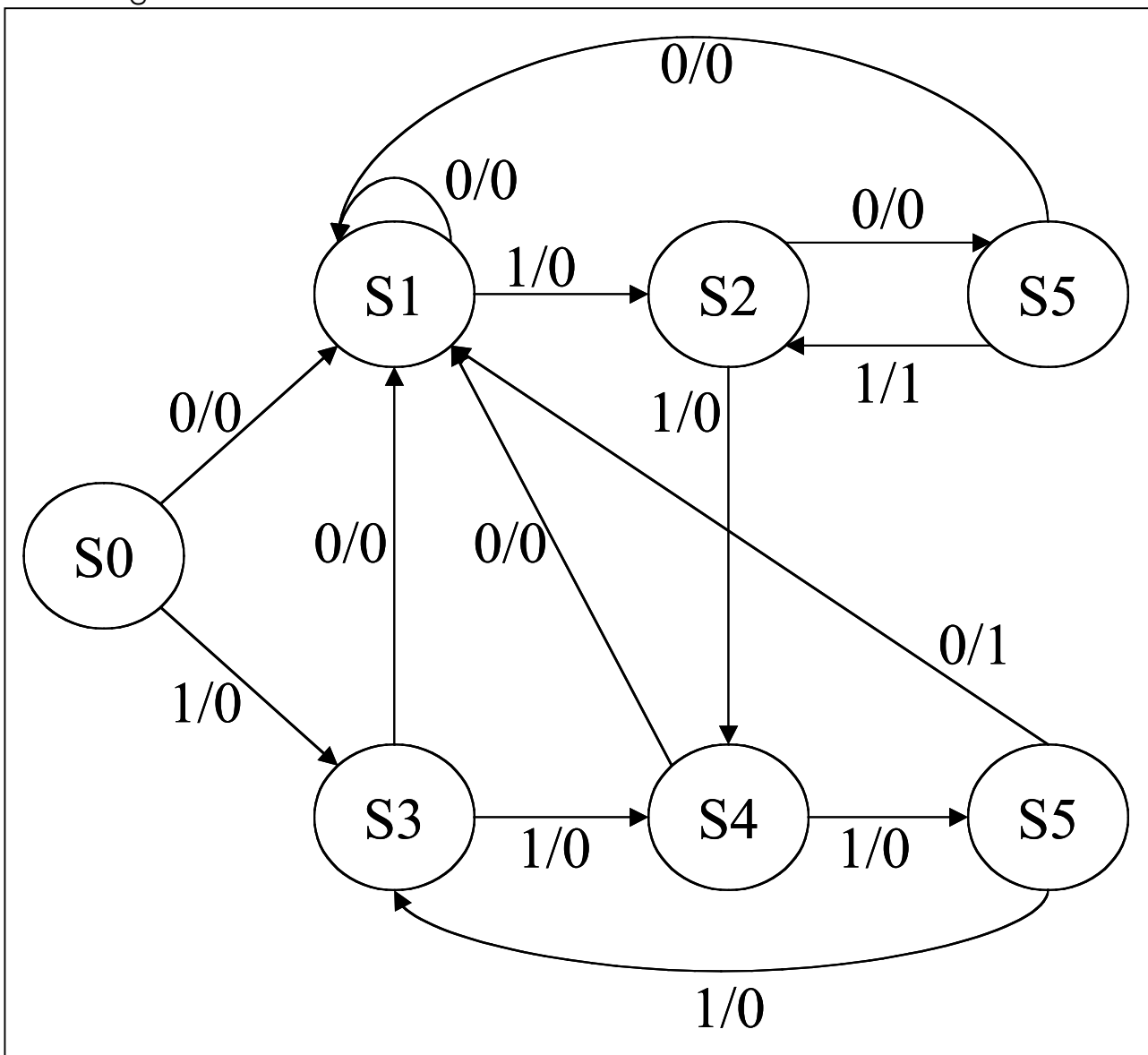


Tabella di flusso:

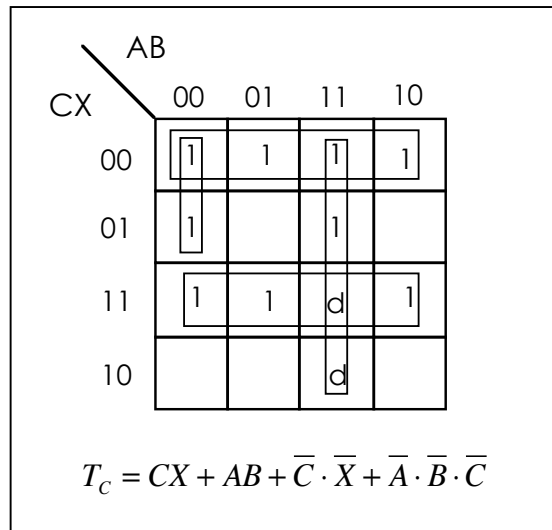
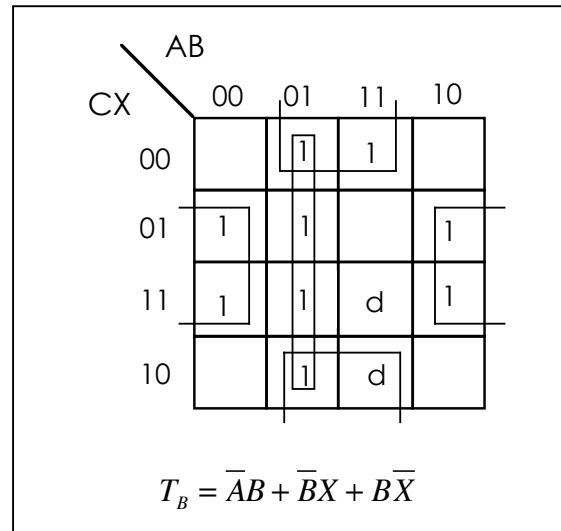
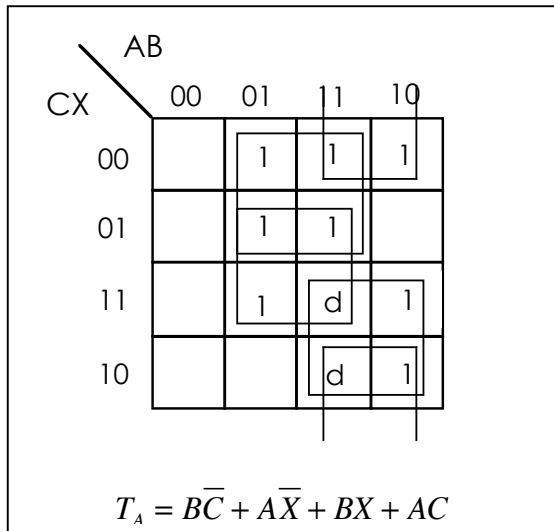
Stato presente (ABC)	Stato futuro/Uscita	
	X=0	X=1
S0 (000)	S1/0	S3/0
S1 (001)	S1/0	S2/0
S2 (010)	S5/0	S4/0
S3 (011)	S1/0	S4/0
S4 (100)	S1/0	S6/0
S5 (101)	S1/0	S2/1
S6 (110)	S1/1	S3/0

Tabella di eccitazione FF-T:

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

Tabella delle transizioni:

A	B	C	X	A'	TA	B'	TB	C'	TC	Z
0	0	0	0	0	0	0	0	1	1	0
0	0	0	1	0	0	1	1	1	1	0
0	0	1	0	0	0	0	0	1	0	0
0	0	1	1	0	0	1	1	0	1	0
0	1	0	0	1	1	0	1	1	1	0
0	1	0	1	1	1	0	1	0	0	0
0	1	1	0	0	0	0	1	1	0	0
0	1	1	1	1	1	0	1	0	1	0
1	0	0	0	0	1	0	0	1	1	0
1	0	0	1	1	0	1	1	0	0	0
1	0	1	0	0	1	0	0	1	0	0
1	0	1	1	0	1	1	1	0	1	1
1	1	0	0	0	1	0	1	1	1	1
1	1	0	1	0	1	1	0	1	1	0
1	1	1	0	D	D	D	D	D	D	0
1	1	1	1	D	D	D	D	D	D	0



Infine, per quanto riguarda l'uscita: $Z = \overline{A}\overline{B}CX + ABC \cdot \overline{X}$.

ESERCIZIO 2 (NO: 8 punti - VO: 6 punti)

Si consideri il seguente formato per la rappresentazione binaria dei numeri in virgola mobile: 24 bit, con esponente a 8 bit in eccesso **125**, mantissa frazionaria e normalizzata in segno e valore (1.M).

1. (NO: 2 punti – VO: 2 punti) Si calcoli il minimo e il massimo numero **positivo** rappresentabili, escluso lo zero.
2. (NO: 3 punti – VO: 2 punti) Si rappresentino nel formato dato i numeri 130.25 e 120.75.
3. (NO: 3 punti – VO: 2 punti) Si sommino i due numeri seguendo i passi usati nell'algoritmo dei calcolatori.

Soluzione.

1. Minimo numero: 2^{-125} .
Massimo numero: $(2-2^{-15}) * 2^{130}$.
2. $130.25 = 10000010.01 = 1.000001001 * 2^{-7}$.
 $120.75 = 1111000.11 = 1.11100011 * 2^{-6}$.

	S	Esponente	Mantissa
130.25	0	10000100	000001001000000
120.75	0	10000011	111000110000000

3. Allineando le mantisse e sommando si ottiene:

$$\begin{array}{r}
 130.25 = 1.000001001 * 2^{-7} + \\
 120.75 = 0.111100011 * 2^{-7} = \\
 \hline
 1.111101100 * 2^{-7} = 251
 \end{array}$$

La cui rappresentazione è:

	S	Esponente	Mantissa
251	0	10000100	111101100000000

ESERCIZIO 3 (solo NO: 8 punti)

Si scriva il codice Assembler MIPS di una funzione che, dati tre vettori di N interi u, v, w, scriva nella posizione w(i) il valore u(i)+v(i), se u(i)<v(i), e il valore u(i)-v(i), altrimenti. Si consideri che gli indirizzi iniziali dei vettori u, v, w siano memorizzati in \$4, \$5, \$6, rispettivamente, e che N sia memorizzato in \$7.

In altri termini il codice MIPS può implementare la funzione C:

```
void elabora(int *u, int *v, int *w, int N)
{
    int i;
    for(i=0; i<N; i++)
        if(u(i)<v(i))
            w(i)=u(i)+v(i);
        else
            w(i)=u(i)-v(i);
}
```

Soluzione.

$\$8 \leftarrow i$; $\$9 \leftarrow u(i) < v(i)$
 $\$10 \leftarrow u(i)$; $\$11 \leftarrow v(i)$; $\leftarrow \$12 \leftarrow w(i)$

```
elabora: addi $29, $29, -20
          sw $8, 0($29)
          sw $9, 4($29)
          sw $10, 8($29)
          sw $11, 12($29)
          sw $12, 16($29)
          move $8, $0
for:      beq $8, $7, exit
          lw $10, 0($4)
          lw $11, 0($5)
          slt $9, $10, $11
          bne $9, $0, sum
          sub $12, $10, $11
continue: sw $12, 0($6)
          addi $8, $8, 1
          addi $4, $4, 4
          addi $5, $5, 4
          addi $6, $6, 4
          j for

exit:     lw $8, 0($29)
          lw $9, 4($29)
          lw $10, 8($29)
          lw $11, 12($29)
          lw $12, 16($29)
          addi $29, $29, 20
          jr $31

sum:      add $12, $10, $11
          j continue
```

ESERCIZIO 3 (solo VO: 7 punti)

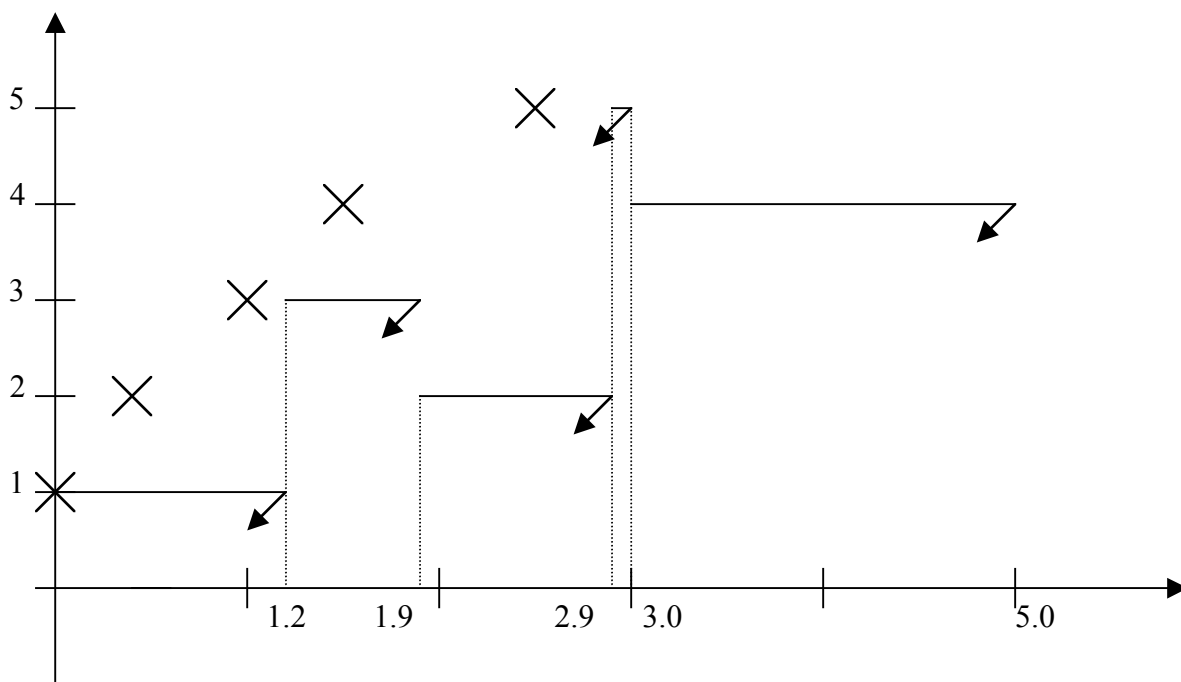
Sia data la seguente lista di processi:

Processo	Tempo di arrivo	Tempo di CPU
1	0.0	1.2
2	0.4	1.0
3	1.0	0.7
4	1.5	2.0
5	2.5	0.1

1. (4 punti) Mostrare, utilizzando il metodo grafico, la sequenza di esecuzione dei processi qualora si impieghi la politica di scheduling SJF monoprogrammata.
2. (3 punti) Calcolare il tempo di *turnaround* medio e il tempo di *turnaround* pesato medio.

Soluzione.

Grafico processo-tempo con politica SJF monoprogrammata:



Processo	Arrivo	Inizio	Fine	Turnaround	WT
1	0.00	0.00	1.20	1.20	1.00
2	0.40	1.90	2.90	2.50	2.50
3	1.00	1.20	1.90	0.90	1.29
4	1.50	3.00	5.00	3.50	1.75
5	2.50	2.90	3.00	0.50	5.00
Media				1.72	2.31

ESERCIZIO 4 (NO: 7 punti – VO: 5 punti)

I trasferimenti di parole a/dalla memoria di un calcolatore sono codificati utilizzando il codice di Hamming. Si consideri la parola di 7 bit 0110101 (il bit meno significativo è a sinistra). **Spiegando bene ogni passo del ragionamento:**

- 1) (NO: 1 punto – VO: 1 punto) calcolare il minimo numero di bit di controllo necessari per la codifica della parola;
- 2) (NO: 3 punti – VO: 2 punti) codificare la parola data;
- 3) (NO: 3 punti – VO: 2 punti) imporre un errore nel quinto bit della **parola inizialmente data**. Spiegare come l'errore viene rivelato e corretto per mezzo della codifica di Hamming.

Soluzione.

- 1) Deve essere rispettata la condizione:

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

dove K è il numero di bit di controllo inseriti. Essendo N=7, il numero minimo di bit di controllo richiesto è 4.

- 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
		0		1	1	0		1	0	1

Dove c₀...c₃ sono i quattro bit costituenti il vettore di controllo, e b₀...b₇ gli otto bit trasmessi. Tali bit si ottengono con le seguenti operazioni

$$c_0 = b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0 \oplus 1 \oplus 0 \oplus 1 \oplus 1 = 1$$

$$c_1 = b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$$

$$c_2 = b_1 \oplus b_2 \oplus b_3 = 1 \oplus 1 \oplus 0 = 0$$

$$c_3 = b_4 \oplus b_5 \oplus b_6 = 1 \oplus 0 \oplus 1 = 0$$

La stringa codificata è 10001100101.

- 3) Nell'ipotesi di un errore sul quinto bit della stringa iniziale, la stringa ricevuta risulta: 10001100**0**01. Per rivelare questo errore, bisogna ricalcolare i bit di controllo:

$$c'_0 = b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$$

$$c'_1 = b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 = 0$$

$$c'_2 = b_1 \oplus b_2 \oplus b_3 = 1 \oplus 1 \oplus 0 = 0$$

$$c'_3 = b_4 \oplus b_5 \oplus b_6 = 0 \oplus 0 \oplus 1 = 1$$

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):

$$e_0 = c_0 \oplus c'_0 = 1$$

$$e_1 = c_1 \oplus c'_1 = 0$$

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

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

Poiché il vettore risultante 1001 non è nullo, vi è un errore nella stringa di 12 bit e precisamente nella posizione indicata dal vettore di errore tradotto in notazione decimale (posizione 9). Il bit sbagliato nella stringa codificata è quindi b_4 , che può venire dunque corretto.

ESERCIZIO 5 (solo VO: 7 punti)

Sia data la sequenza di istruzioni RISC (A, B, X, Y sono riferimenti a locazioni di memoria):

MOV R1, A

MOV R2, B

ADD R3, R1, R2

MOV R4, X

MOV R5, Y

ADD R6, R4, R5

ADD R7, R3, R4

ADD R8, R6, R5

Individuare le istruzioni che operano su dati dipendenti. Spiegare perché possono creare dei "ritardi" nell'esecuzione della pipeline e proporre delle tecniche per risolvere questo problema.

Soluzione.

Nelle sequenza di istruzioni assegnata si possono individuare due tipi di dipendenza: il primo riguarda le due istruzioni 'ADD R3, R1, R2' e 'ADD R6, R4, R5' la cui esecuzione dipende dalla velocità del caricamento dei registri R1, R2, R4 ed R5 con i dati letti dalla memoria. Le due istruzioni di somma possono essere bloccate per un certo numero di cicli in attesa che vengano caricati i valori corretti nei registri che contengono gli addendi. Per ovviare a questo inconveniente il compilatore può riordinare il codice spostando la prima istruzione di somma subito prima della seconda istruzione di somma. In questo modo l'istruzione 'ADD R3, R1, R2' viene chiamata dopo un certo numero di cicli dall'istruzione MOV consentendo il caricamento dei registri R1 e R2. La seconda istruzione di ADD a questo punto viene chiamata più tardi rispetto all'organizzazione originaria del codice, riducendo così il ritardo nell'esecuzione. Le successive istruzioni di somma operano su dati dipendenti nel senso che gli addendi di una somma sono i risultati di somme precedenti. In questo caso per evitare ritardi si usa una tecnica 'hardware' detta 'data forwarding' che consente di inviare i risultati di una istruzione nei registri di ingresso della ALU senza aspettare la scrittura dei registri di CPU.