

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI  
CALCOLATORI ELETTRONICI  
NUOVO E VECCHIO ORDINAMENTO DIDATTICO**  
18 Giugno 2004

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

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

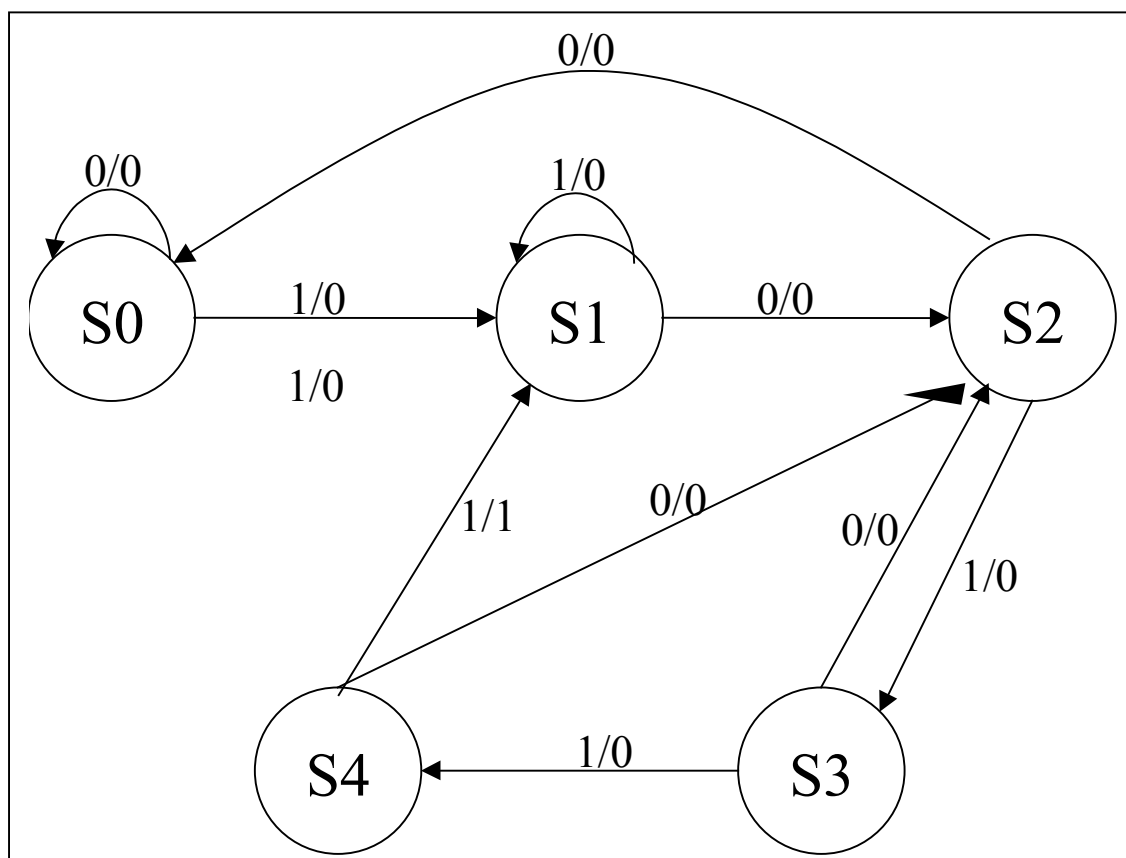
Progettare una rete sequenziale che presenti un ingresso X e un'uscita Z posta a 1 ogni volta che viene riconosciuta la sequenza 10111.

Si richiede:

- (NO: 6 punti – VO: 4 punti) il diagramma degli stati, la tabella di flusso e la tabella delle transizioni;
- (NO: 4 punti – VO: 2 punti) il calcolo delle forme minime delle variabili di eccitazione dei flip flop con le mappe di Karnaugh. Si usino flip flop JK. Calcolare anche la rete combinatoria per l'uscita Z.
- (solo VO: 2 punti) Realizzare un flip flop T a partire da un flip flop JK.

**Soluzione.**

Il diagramma degli stati è il seguente:



La tabella di flusso è data da:

Stato presente	Stato successivo/Uscita	
	X=0	X=1
S0	S0/0	S1/0
S1	S2/0	S1/0
S2	S0/0	S3/0
S3	S2/0	S4/0
S4	S2/0	S1/1

Per codificare 5 stati occorrono tre flip flop. La codifica è la seguente:

S0 → 0 0 0; ...; S4 → 1 0 0. Nel seguito indicheremo ciascun bit della codifica con le lettere A, B, C. L'apice indicherà il bit nell'istante successivo a quello considerato.

A partire dalla tabella di eccitazione del flip flop JK:

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

A	B	C	X	A'	Ja	Ka	B'	Jb	Kb	C'	Jc	Kc	Z
0	0	0	0	0	0	D	0	0	D	0	0	D	0
0	0	0	1	0	0	D	0	0	D	1	1	D	0
0	0	1	0	0	0	D	1	1	D	0	D	1	0
0	0	1	1	0	0	D	0	0	D	1	D	0	0
0	1	0	0	0	0	D	0	D	1	0	0	D	0
0	1	0	1	0	0	D	1	D	0	0	0	D	0
0	1	1	0	0	0	D	1	D	0	0	D	1	0
0	1	1	1	1	1	D	0	D	1	0	D	1	0
1	0	0	0	0	D	1	1	1	D	0	0	D	0
1	0	0	1	0	D	1	0	0	D	1	1	D	1
1	0	1	0	D	D	D	D	D	D	D	D	D	D
1	0	1	1	D	D	D	D	D	D	D	D	D	D
1	1	0	0	D	D	D	D	D	D	D	D	D	D
1	1	0	1	D	D	D	D	D	D	D	D	D	D
1	1	1	0	D	D	D	D	D	D	D	D	D	D
1	1	1	1	D	D	D	D	D	D	D	D	D	D

Ora possiamo disegnare le mappe di Karnaugh

		AB			
		00	01	11	10
CX	00			d	d
	01			d	d
	11		1	d	d
	10			d	d

$J_A = BCX$

		AB			
		00	01	11	10
CX	00	d	d	d	1
	01	d	d	d	1
	11	d	d	d	d
	10	d	d	d	d

$K_A = 1$

		AB			
		00	01	11	10
CX	00		d	d	1
	01		d	d	
	11		d	d	d
	10	1	d	d	d

$J_B = C\bar{X} + A\bar{X}$

		AB			
		00	01	11	10
CX	00	d	1	d	d
	01	d		d	d
	11	d	1	d	d
	10	d		d	d

$K_B = \bar{C} \cdot \bar{X} + CX$

		AB			
		00	01	11	10
CX	00			d	
	01	1	1	d	1
	11	d	d	d	d
	10	d	d	d	d

$J_C = X$

		AB			
		00	01	11	10
CX	00	d	d	d	d
	01	d	d	d	d
	11		1	d	d
	10	1	1	d	d

$K_C = B + \bar{X}$

Infine, per quanto riguarda l'uscita Z:

$$Z = A \cdot \overline{B} \cdot \overline{C} \cdot X$$

Volendo utilizzare anche i don't care:

AB					
CX		00	01	11	10
	00			d	
	01			d	1
	11			d	d
	10			d	d

$$Z = AX$$

c)

Per realizzare un flip flop T da un JK, è sufficiente connettere fra loro gli ingressi J e K, come si può vedere confrontando la tabella delle transizioni del JK con quella del T:

	J	K	Q	Q'	T	Q	Q'
{	0	0	0	0	0	0	0
	0	0	1	1	0	1	1
	0	1	0	0	1	0	1
	0	1	1	0	1	1	0
{	1	0	0	1			
	1	0	1	1			
	1	1	0	1			
	1	1	1	0			

**ESERCIZIO 2 (NO: 8 punti – VO: 7 punti)**

Si consideri una memoria primaria costituita da 128 parole e una memoria cache costituita da 32 parole. Il metodo di indirizzamento della cache sia quello completamente associativo con blocchi di 8 parole.

- (1) (1 punto) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache.
- (2) (NO: 5 punti – VO: 4 punti) Si considerino le seguenti chiamate ad altrettante parole (indirizzi espressi in decimale): 125, 58, 60, 113, 70, 27, 59, 111, 5, 92. Si indichi il contenuto della cache, ovvero quali byte occupano i relativi blocchi di cache, dopo l'ultima chiamata, considerando la **LRU** come strategia di rimpiazzamento dei blocchi.
- (3) (2 punti) Si consideri una gerarchia di memoria a tre livelli costituita da: cache, primaria e disco. Se l'hit ratio di cache è pari a 0.9, l'hit ratio di primaria è pari a 0.95, i tempi di accesso a cache, primaria e disco valgono rispettivamente, 5 nsec, 50 nsec e 5 msec, esprimere il tempo medio di accesso alla gerarchia **in nanosecondi**.

**Soluzione.**

(a)

Con 128 parole, ogni indirizzo è formato da 7 bit così suddivisi secondo il metodo completamente associativo:

< Block frame 4 bit > < Offset 3 bit >

(b)

Blocco 0	Blocco 1	Blocco 2	Blocco 3
88 – 95	56 – 63	104 – 111	0 – 7

(c) Con tutti i dati a nostra disposizione è sufficiente valutare la formula:

$$\bar{T} = H_C T_C + (H_P - H_C)(T_P + T_C) + (1 - H_P)(T_D + T_P + T_C)$$

Quindi:

$$\bar{T} = 0.9 \cdot 5 + 0.05 \cdot 55 + 0.05 \cdot 5000000 = 250007.25 ns$$

### ESERCIZIO 3 (solo NO: 8 punti)

Si scriva il codice Assembler MIPS di una funzione che, ricevendo in ingresso l'indirizzo iniziale di un vettore di interi  $v$ , la sua dimensione  $N$ , un secondo vettore di interi  $w$ , la sua dimensione  $M$ , scriva nel vettore  $w$  il numero di occorrenze degli interi da 0 a  $M-1$ . In altri termini  $w[j]$  conterrà il numero di occorrenze del valore  $j$  nel vettore  $v$ . Si assuma che il contenuto di  $w$  sia inizializzato a 0.

Allocazione dei parametri di ingresso:  $\&v[0] \rightarrow \$4$ ,  $N \rightarrow \$5$ ,  $\&w[0] \rightarrow \$6$ ,  $M \rightarrow \$7$ .

Es. il codice MIPS potrebbe implementare il seguente codice C:

```
void occorrenze(int v[], int N, int w[], int M)
{
    int i,j;

    for(i=0; i<N; i++)
    {
        j=v[i];
        if((j<M) && (j>=0))
            w[j]++;
    }
}
```

### Soluzione.

```
$8 ← i; $9 ← v[i]; $10 ← w[j]; $11 ← (v[i]<M); $12 ← (v[i]<0);
$13 ← &w[j]
```

```
occorrenze:  addi $29, $29, -24      #salvataggio contesto
              sw $8, 0($29)
              sw $9, 4($29)
              sw $10, 8($29)
              sw $11, 12($29)
              sw $12, 16($29)
              sw $13, 20($29)
              move $8, $0           #i=0
for:         beq $8, $5, exit
              lw $9, 0($4)          #j=v[i]
              addi $4, $4, 4        #aggiorno indici
              addi $8, $8, 1
              slt $11, $9, $7       #$11 ← (v[i]<M)
              beq $11, $0, for      #if !($11) then for
              slt $12, $9, $0       #$12 ← (v[i]<0)
              bne $12, $0, for      #if ($12) then for
              muli $13, $9, 4
              add $13, $13, $6      #calcolo indirizzo di w[j]
              lw $10, 0($13)       #
              addi $10, $10, 1      #w[j]++
              sw $10, 0($13)       #
              j for
exit:        lw $8, 0($29)         #ripristino contesto
              lw $9, 4($29)
              lw $10, 8($29)
              lw $11, 12($29)
              lw $12, 16($29)
              lw $13, 20($29)
              addi $29, $29, 24
              jr $31              #ritorno a chiamante
```

**ESERCIZIO 3 (solo VO: 6 punti)**

Si supponga di disporre di tre architetture: a pila, a uno e a due indirizzi. Per ognuna di queste si abbiano le seguenti istruzioni:

A pila		A un indirizzo		A due indirizzi	
Istruzione	Semantica	Istruzione	Semantica	Istruzione	Semantica
PUSH X	$M[X] \rightarrow \text{push}$	STORE X	$\text{ACC} \rightarrow M[X]$	MOV X1, X2	$M[X1] \rightarrow M[X2]$
POP X	$\text{pop} \rightarrow M[X]$	LOAD X	$M[X] \rightarrow \text{ACC}$	ADD X1, X2	$M[X1] + M[X2] \rightarrow M[X2]$
ADD	$\text{pop} + \text{pop} \rightarrow \text{push}$	ADD X	$\text{ACC} + M[X] \rightarrow \text{ACC}$	DIV X1, X2	$M[X1] / M[X2] \rightarrow M[X2]$
DIV	$\text{pop1} / \text{pop2} \rightarrow \text{push}$	DIV X	$\text{ACC} / M[X] \rightarrow \text{ACC}$	MUL X1, X2	$M[X1] * M[X2] \rightarrow M[X2]$
MUL	$\text{pop} * \text{pop} \rightarrow \text{push}$	MUL X	$\text{ACC} * M[X] \rightarrow \text{ACC}$		

ACC è il registro accumulatore della macchina a un indirizzo.

$M\{X\}$  indica il dato nella locazione di memoria X.

Si scriva, per ognuna delle tre macchine, la sequenza delle istruzioni necessarie per realizzare la seguente operazione (rispettando le precedenze delle operazioni):

$$Z = A / B + B * (A + C)$$

Le lettere indicano le locazioni di memoria dove si trovano i dati.

Suggerimento: Dove necessario, si faccia uso di ulteriori locazioni dove introdurre i risultati parziali, pena la perdita dei dati iniziali nelle locazioni date.

**Soluzione.**

A pila		A un indirizzo		A due indirizzi	
PUSH B	M[C] → push	LOAD A	M[A] → ACC	MOV B, P	M[B] → M[P]
PUSH A	M[D] → push	DIV B	ACC + M[B] → ACC	DIV A, P	M[A] + M[P] → M[P]
DIV	pop1 / pop2 → push	STORE Z	ACC → M[Z]	MOV A, Z	M[A] → M[Z]
PUSH A	M[A] → push	LOAD A	M[A] → ACC	ADD C, Z	M[C] + M[Z] → M[Z]
PUSH C	M[C] → push	ADD C	ACC + M[C] → ACC	MUL B, Z	M[B] * M[Z] → M[Z]
ADD	pop + pop → push	MUL B	ACC * M[B] → ACC	ADD P, Z	M[P] + M[Z] → M[Z]
PUSH B	M[B] → push	ADD Z	ACC + M[Z] → ACC		
MUL	pop * pop → push	STORE Z	ACC → M[Z]		
ADD	pop + pop → push				
POP Z	pop → M[Z]				

**ESERCIZIO 4 (NO: 7 punti – VO: 6 punti)**

Si consideri un campo di 5 bit, utilizzati per la rappresentazione dei numeri in virgola mobile secondo i seguenti formati:

- (1) 1 bit per segno, 2 bit per l'esponente (in eccesso 1), 2 bit per la mantissa, rappresentazione con bit intero implicito (1.M)
- (2) 1 bit segno, 3 bit per l'esponente (in eccesso 3), 1 bit per la mantissa, rappresentazione con bit intero implicito (1.M)
- (a) (1 punto) Indicare il numero di valori rappresentabili nei due casi;
- (b) (2 punti) Indicare il minimo e il massimo valore positivo rappresentabili, escluso lo zero, dettagliando i passaggi intermedi;
- (c) (NO: 4 punti – VO: 3 punti) Rappresentare il valore 5 nei due casi, specificando a quale valore esso venga approssimato nel caso in cui non sia rappresentabile esattamente.

**Soluzione.**

- (a) Indipendentemente dal formato, il numero di valori rappresentabili dipende dal numero di bit a disposizione. Essendo 5 bit, possiamo rappresentare  $2^5 = 32$  valori.

- (b)
- |                     |                        |  |
|---------------------|------------------------|--|
| Rappresentazione 1. | Minimo: $2^{-1}=0.5$ . | Massimo: $2^2 * (2-2^{-2}) = 2^3-1 = 7$      |
| Rappresentazione 2. | Minimo: $2^{-3}$ .     | Massimo: $2^4 * (2-2^{-1}) = 2^5 - 2^3 = 24$ |

- (c)

$$5_{10} = 101_2 = 1.01 * 2^2$$

	S	E	M
Rappresentazione 1:	0	11	01
Rappresentazione 2:	0	101	0

Si può notare facilmente che, essendo la mantissa della rappresentazione 2 formata da un solo bit, l'1 meno significativo viene "troncato" e il valore 5, pertanto, non è rappresentato esattamente ma approssimato al valore più vicino, che si ottiene riconvertendo il numero in virgola mobile. Il valore con cui viene approssimato è dunque 4.

**ESERCIZIO 5 (solo VO: 5 punti)**

- (1) Descrivere la classificazione di Flynn delle architetture parallele, spiegando che tipo di calcolatore ricade in ciascuna architettura. (3 punti)
- (2) Illustrare le misure di "speedup" e "efficienza" usate per valutare le prestazioni di una architettura multiprocessore. (2 punti)

**Soluzione**

Vedi dispense del corso.