

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

16 Luglio 2003

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

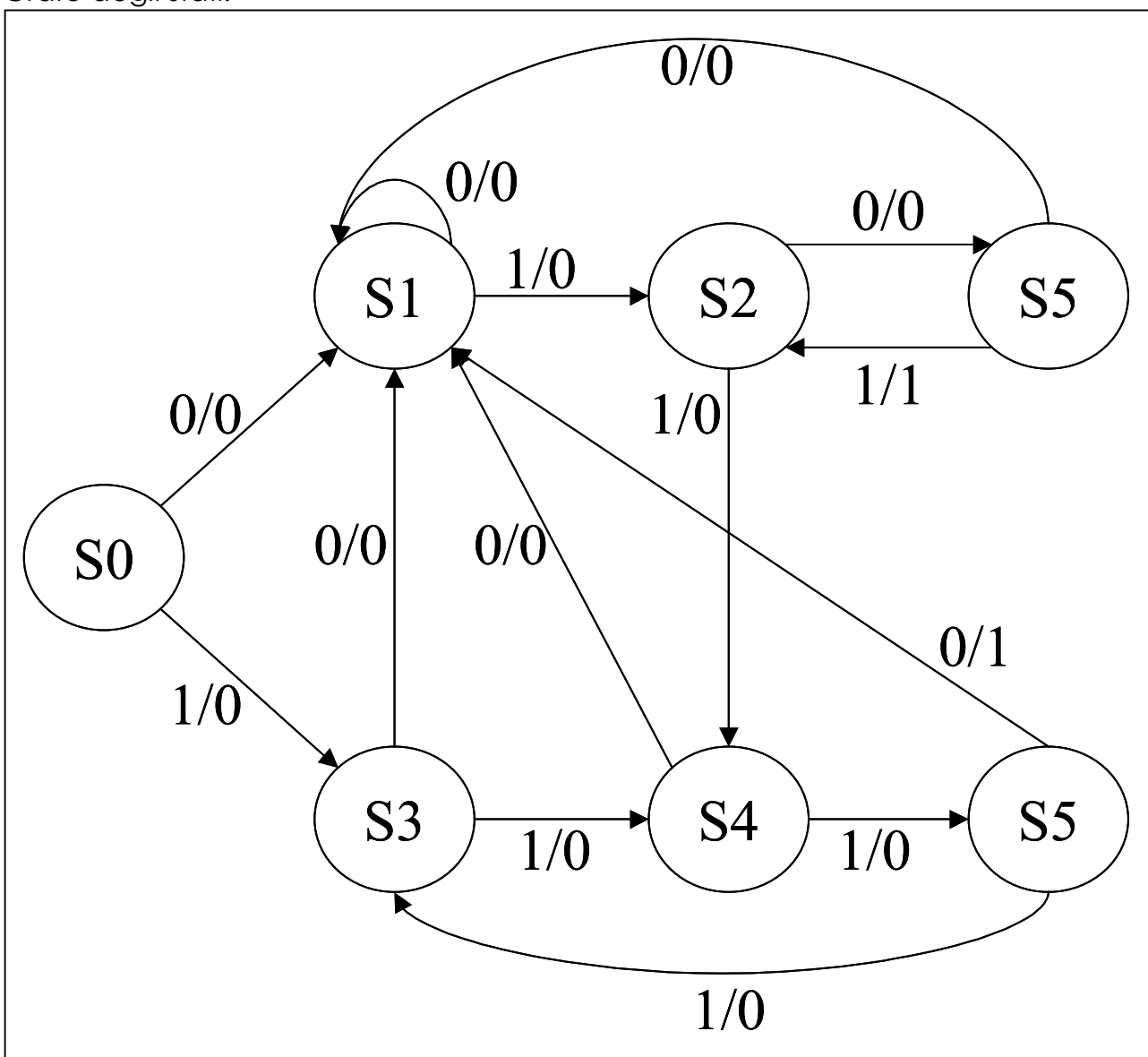
ESERCIZIO 1 (NO: 10 punti – VO: 8 punti)

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.

- (a) (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;
- (b) (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:



ERRATA CORRIGE. Il nodo S5 in basso deve essere chiamato S6. L'arco che parte da S6 in corrispondenza di X=1 deve ritornare a S6.

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	S6/0

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	1	0	1	0	0	0	0
1	1	1	0	D	D	D	D	D	D	0
1	1	1	1	D	D	D	D	D	D	0

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

$$T_A = \overline{B}\overline{C}\overline{X} + \overline{A}\overline{X} + \overline{A}BX + AC$$

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

$$T_B = \overline{A}B + \overline{B}X + B\overline{X}$$

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

$$T_C = CX + \overline{C} \cdot \overline{X} + \overline{A} \cdot \overline{B} \cdot \overline{C}$$

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

ESERCIZIO 2 (8 punti)

Si consideri un calcolatore che dispone di una memoria principale di 256 Mbyte e di una memoria cache di 512 Kbyte. E' possibile accedere al singolo byte.

1. (VO: 3 punti - NO: 4 punti) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache nel caso venga usata la modalità di indirizzamento

- Diretto, con blocchi da 32 byte

- "associativo su insiemi", e ciascun insieme contenga sedici blocchi da 8 byte

2. (VO: 3 punti – NO: 4 punti) Si considerino le due parole di indirizzo rispettivamente 311F5x8 e 11yF50A. Calcolare i valori delle cifre x e y tali per cui le due parole si trovano:

a) Nella stessa "linea" di cache nel caso di indirizzamento diretto.

b) Nello stesso insieme nel caso di indirizzamento associativo su insiemi a sedici vie.

3. (solo VO: 2 punti) Spiegare in modo chiaro e sintetico vantaggi e svantaggi dei due metodi di indirizzamento della cache presentati nei punti precedenti.

Soluzione.

1. <TAG 9 bit > <CACHE INDEX 14 bit> <OFFSET 5 bit>

<TAG 13 bit > <CACHE INDEX 12 bit> <OFFSET 3 bit>

2.

311F5x8 → 0011 0001 0001 1111 0101 xxxx 1000

11yF50A → 0001 0001 yyyy 1111 0101 0000 1010

Indirizzamento diretto:

311F5x8 → 0011 0001 0|001 1111 0101 xxx|x 1000

11yF50A → 0001 0001 y|yyy 1111 0101 000|0 1010

Quindi y=[1,9], x=[0,1].

Associativo su insiemi:

311F5x8 → 0011 0001 0001 1|111 0101 xxxx 1|000

11yF50A → 0001 0001 yyyy 1|111 0101 0000 1|010

In tal caso y può assumere qualunque valore, x=0.

3. Vedi dispense del corso

ESERCIZIO 3 (solo NO: 8 punti)

E' dato il seguente codice Assembler MIPS.

<pre>F: addi \$29, \$29, -16 sw \$8, 0(\$29) sw \$9, 4(\$29) sw \$10, 8(\$29) sw \$11, 12(\$29) lw \$6, 0(\$4) move \$10, \$0 A: addi \$10, \$10, 1 slt \$8, \$10, \$5 beq \$8, \$0, B addi \$4, \$4, 4</pre>	<pre> lw \$11, 0(\$4) slt \$9, \$11, \$6 beq \$9, \$0, A move \$6, \$11 j A B: lw \$8, 0(\$29) lw \$9, 4(\$29) lw \$10, 8(\$29) lw \$11, 12(\$29) addi \$29, \$29, 16 jr \$31</pre>
---	---

Spiegare il significato di ogni istruzione e individuare qual è la funzione svolta dal codice presentato. Si ricordi che lo stack pointer è contenuto in \$29, che \$31 è tipicamente usato per il ritorno da procedura (contiene il PC).

(La risposta è valida anche traducendo il codice MIPS in C)

Soluzione. Il seguente blocco:

```
Fun:      addi $29, $29, -16
        sw $8, 0($29)
        sw $9, 4($29)
        sw $10, 8($29)
        sw $11, 12($29)
```

effettua il salvataggio dei registri \$8...\$11 nello stack (puntato da \$29).

Le istruzioni successive:

```
        lw $6, 0($4)
        move $10, $0
```

caricano in \$6 il valore puntato da \$4, e azzerano il contenuto di \$10.

Il frammento di codice etichettato con 'A':

```
A:      addi $10, $10, 1
        slt $8, $10, $5
        beq $8, $0, B
        addi $4, $4, 4
        lw $11, 0($4)
        slt $9, $11, $6
        beq $9, $0, A
        move $6, $11
        j A
```

aggiorna il valore \$10 e controlla se \$10 risulta minore di \$5. Se no, salta a B.

Se sì, preleva il valore contenuto nella locazione adiacente a \$4 e lo confronta con \$11. Se \$11 < \$6, allora esso viene spostato in \$6, altrimenti si ritorna ad A e si accede alla successiva locazione puntata da \$4.

Il frammento di codice etichettato con 'B' effettua il ripristino dei parametri dallo stack, e ritorna al chiamante.

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

Il codice proposto ha dunque lo scopo di trovare il più piccolo numero entro una sequenza di \$5 celle di memoria adiacenti, il cui indirizzo iniziale si trova in \$4. Tale minimo viene depositato in \$6.

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 seguente che rappresenta una parola codificata con il codice di Hamming (il bit meno significativo è a sinistra): 011101010001.

1. (1 punto) Quale è la dimensione della parola trasmessa (escludendo i bit di codice)?
2. (4 punti) Verificare se la stringa ricevuta contiene un errore e in caso affermativo correggerlo
3. (1 punto) Scrivere la parola corretta che si voleva trasmettere?

Soluzione.

La stringa trasmessa contiene 12 bit, dunque i bit di codice sono 4 e la parola trasmessa ha dimensione di 8 bit (il numero di bit di codice m è scelto come il più piccolo intero che soddisfa la relazione $m + b \leq 2^m - 1$, dove b indica il numero di bit della parola da codificare. In questo caso $m + b = 12$ e il più piccolo intero che soddisfa la relazione precedente è 4).

Se indichiamo con b i bit della parola inviata e con c i bit di codice, la stringa di 12 bit deve essere interpretata nel modo seguente:

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	1	1	1	0	1	0	1	0	0	0	1

Il calcolo dei bit di errore è il seguente:

$$e_0 = c_0 \oplus b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$e_1 = c_1 \oplus b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 1 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1$$

$$e_2 = c_2 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_7 = 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$e_3 = c_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 1 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 0$$

Posizione dell'errore = 7

Nella stringa il 7° bit (che corrisponde a b_3) è errato.

La parola di 8 bit trasmessa (bit meno significativo a sinistra) è dunque: 10110001.

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

L'ampiezza della linea dati del bus di un calcolatore è pari a 32 bit. La frequenza del clock della CPU è di 625 MHz.

- 1) (NO: 4 punti – VO: 3 punti) Ipotezzando che il bus sincrono abbia la stessa frequenza di clock della CPU, la durata di una trasmissione sul bus impieghi 4 cicli di clock, e che il tempo di ciclo della memoria sia 80 ns, illustrare chiaramente il protocollo di lettura su bus sincrono utilizzando l'opportuno grafico, indicando il tempo complessivo di trasferimento di una parola da 32 bit. (3 punti)
- 2) (3 punti) Se le istruzioni di CPU necessarie ad effettuare il trasferimento dati da una periferica alla memoria nel caso in cui i trasferimenti periferica-calcolatore vengano gestiti mediante IO da programma richiedono 6 cicli di clock, calcolare la massima velocità di trasferimento (in bit/s) fra periferica e calcolatore che è possibile raggiungere effettuando i trasferimenti mediante IO da programma.

Soluzione:

- 1) La durata di un ciclo di clock è pari a $1/(625 \text{ MHz}) = 1.6 \text{ ns}$

La lettura su un bus sincrono avviene secondo il protocollo seguente:

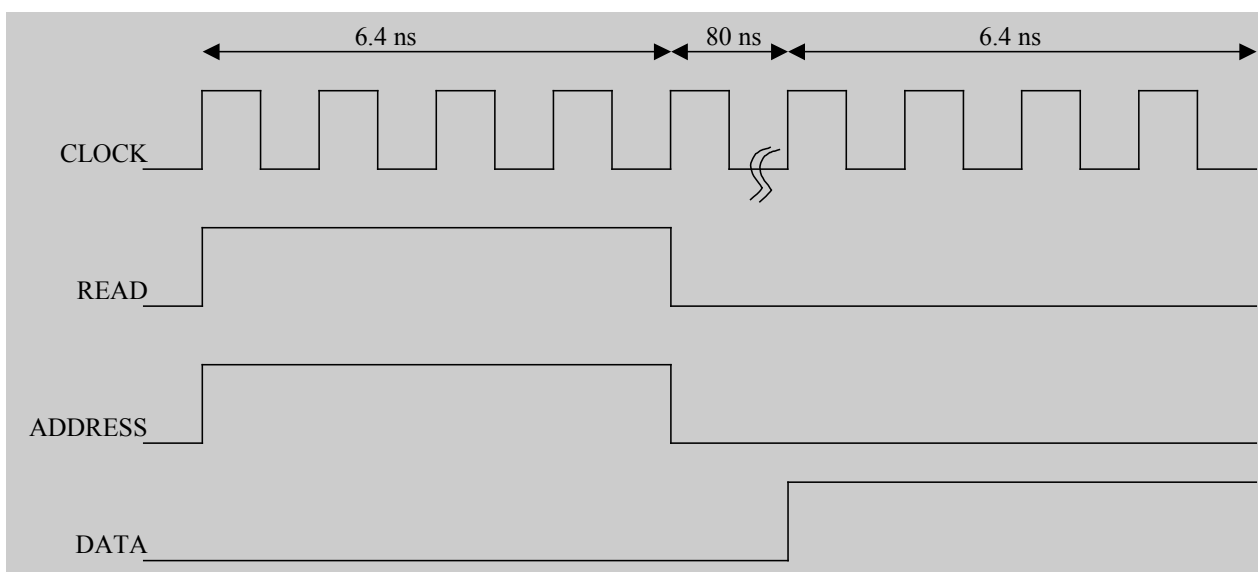
- Segnale di READ sulla linea di controllo e contemporaneamente l'indirizzo della locazione in cui risiede il dato sulla linea indirizzi:

4 cicli di clock = 6.4 ns

- Lettura della parola dalla memoria: **80 ns**
- Trasferimento della parola dalla memoria:
se la parola da leggere ha ampiezza pari a 32 bit

trasferimento della parola = 6.4 ns

Tempo totale per leggere una parola dalla memoria = (12.8 + 80) ns



- 2) Sono necessari 6 cicli di clock per trasferire un blocco di 32 bit, cioè si trasferisce un blocco con la frequenza di 625/6 MHz. La massima velocità di trasferimento è dunque pari a $(32 \cdot 625 \cdot 10^6)/6 \text{ bit/s} = 3333 \cdot 10^6 \text{ bit/s}$.

ESERCIZIO 5 (solo VO: 5 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.

Nella 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.