CALCOLATORI ELETTRONICI per Ing. INFORMATICA (C. COLOMBO)

Prove scritte del 6/2/2002 e 15/2/2002 :
soluzioni di alcuni esercizi

① [RAPPRESENTAZIONE DELL'INFORMAZIONE E ARITMETICA BINARIA]

La rappresentazione ASCII (si veda la tabella) fa corrispondere ad ogni carattere un codice di 7 bit. Ad esempio, il codice per '8' è $(38)_{16}$, ovvero, in notazione funzionale, $\alpha(\text{'8'}) = (38)_{16}$. Si calcoli il numero binario in rappresentazione in complemento a 2 corrispondente alla differenza $\alpha(\text{'3'}) - \alpha(\text{'9'})$. A quale numero decimale corrisponde?

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

$\alpha('3') = 33H = 00110011B$
$\alpha('9') = 39H = 00111001B$,

dove abbiamo esteso a 8 bit la rappr. ASCII

Rappresentiamo nella forma in compl. a 2 il numero $-\alpha('9')$, ottenendo $11000111B$ Sommiamo ad $\alpha('3')$, ottenendo

$$\begin{array}{r} 00110011 + \\ 11000111 = \\ \hline 11111010 \end{array} \quad (= \beta)$$

β è un numero negativo (MSB=1) Il numero decimale ad esso corrispondente è

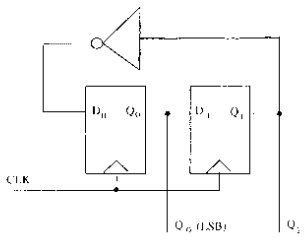
$$-2^{8-1} + 1\times2^6 + 1\times2^5 + 1\times2^4 + 1\times2^3 + 1\times2^1 =$$

$$= -128 + 64 + 32 + 16 + 8 + 2 = -128 + 122 = -6$$

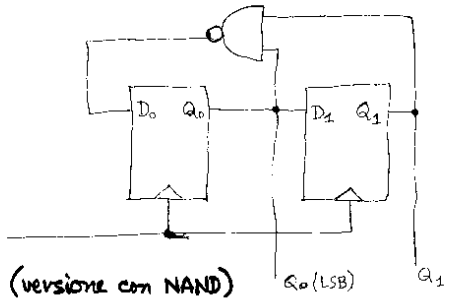
■

(2) [RETI LOGICHE SEQUENZIALI]

L'uscita della macchina sequenziale sincrona riportata in figura è data dalla parola Q_1Q_0 (Q_0 = LSB). Al tempo $t = 0$, subito prima dell'arrivo del primo colpo di clock, la parola di uscita vale $P_0 = (00)_2 = (0)_{10}$. Stabilire il valore decimale di P_i $i = 1, \ldots 9$. Quanti stati distinti ha la macchina? Ogni quanti colpi di clock le uscite si ripetono?



(versione con NOT)



(versione con NAND)

a) versione con NOT

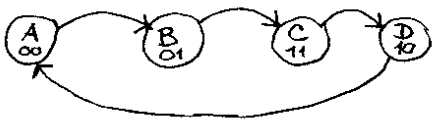
L'equazione caratteristica del flip-flop D è $Q' = D$, ovvero lo stato al tempo $t+1$ è pari all'ingresso al tempo t. Possiamo quindi enumerare le uscite dei flip-flop a partire dalla condizione iniziale secondo il seguente schema:

t	$Q_1 Q_0$	$Q_1' = Q_0$	$Q_0' = \overline{Q_1}$	$(P_t)_{10}$
0	0 0	0	1	0
1	0 1	1	1	1
2	1 1	1	0	3
3	1 0	0	0	2
4	0 0	0	1	0
5				
6	(...)	(...)	(...)	(...)
7				
8				
9				

Nota: L'evoluzione delle uscite mantiene le stesse caratteristiche (ordine, ripetitività) indipendentemente dalle condizioni iniziali

È evidente che la sequenza d'uscita si ripete ogni 4 colpi di clock. Dunque le prime 10 uscite sono 0132013201.

Gli stati distinti della macchina sono 4; l'automa evolve secondo il diagramma sotto riportato:

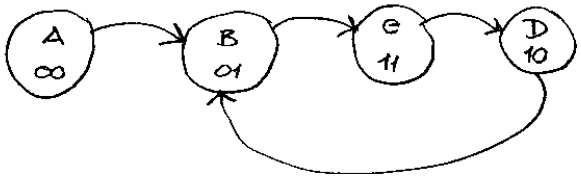


b) Versione con NAND

La tabella precedente si modifica come segue:

t	$Q_1 Q_0$	$Q_1' = Q_0$	$Q_0' = \overline{Q_1 Q_0}$	$(P_t)_{10}$
0	0 0	0	1	0
1	0 1	1	1	1
2	1 1	1	0	3
3	1 0	0	1	2
4	0 1	1	1	1
5				
6	(...)	(...)	(...)	(...)

Notiamo che, rispetto al caso precedente, l'evoluzione del sistema dipende dalla condizione di inizializzazione in maniera leggermente più complessa



Infatti lo stato A (=00) non è ottenibile per alcun valore di t, qualora $P_0 \in \{B, C, D\}$. Quindi il numero di stati distinti attraversati dall'automa dipende dalle condizioni iniziali potrà essere 4, oppure 3. In ogni caso, a regime le uscite si ripetono dopo soli 3 colpi di clock: ... 132132132 ....

③ [MEMORIZZAZIONE DI DATI E CODICE IN LINGUAGGIO MACCHINA]

Si indichi la lunghezza (in byte) di ciascuna riga di codice assembler 8086 sotto riportata (indicare lo spazio occupato sia dalle dichiarazioni di variabili che dalle istruzioni).

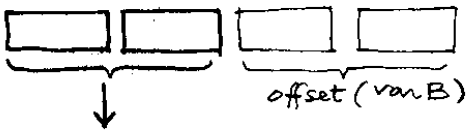
	Codice		Lunghezza
Dati	SEGMENT		0
varA	DW	27H	2
varB	DW	24H	2
varC	DB	0A3	1
Dati	ENDS		0
Codice	SEGMENT		0
...	...	...	
	SUB	BX, BX	2
	MOV	varB, BX	4
LABEL:	ADD	BX, 1023	4
...	...	...	
Codice	ENDS		0
	Totale		15

Note

1) le direttive ASSEMBLY servono alla preparazione del codice macchina da parte dell'ASSEMBLER: ad esse non corrisponde alcuno spazio di memoria
2) DW e DB allocano rispettivamente 2 e 1 byte di memoria
3) "varB" è un indirizzo di variabile, cui corrisponde nell'8086 una word che specifica l'OFFSET di varB nel segmento dati.

Si supponga, per semplicità che codice operativo e modi di indirizzamento degli operandi occupino i primi due byte di ciascuna istruzione.

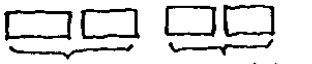
(cfr. nota 3.)



in questi due byte è contenuta tutta l'informazione necessaria per:

a) identificare l'operazione da svolgere (es. MOV, SUB, etc.)
b) conoscere numero e tipo di operandi (numero: 1 o 2, tipo word o byte)
c) nel caso di operandi specificati attraverso registri, identificare i registri (nell'esempio: BX)
d) conoscere i modi di indirizzamento degli operandi; ciò è importante per determinare la lunghezza complessiva dell'istruzione (nell'es., dalla lettura del contenuto dei primi due byte, si sa che bisogna fare il fetch di altri 2 byte per poter eseguire l'istruzione

4) il numero 1023 è specificato con indirizzamento immediato: occupa una WORD nel segmento di codice. Nota: anche l'istruzione ADD BX, 1 prende 4 byte nel segmento di codice, nonostante il fatto che l'operando immediato possa in teoria essere rappresentato con un solo byte. Infatti, la dimensione del dato immediato deve sempre accordarsi a quella del secondo operando (che, in questo caso, è un registro a 16 bit)

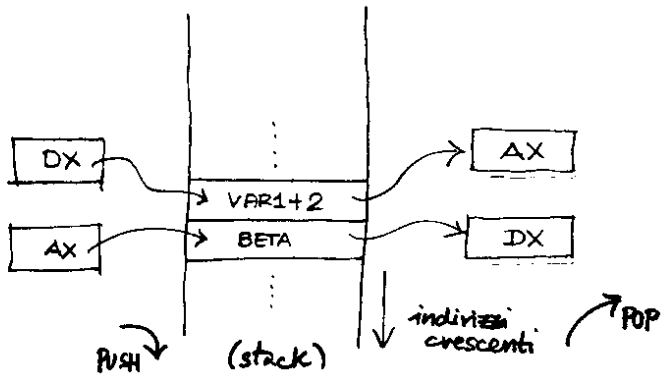


④ [ASSEMBLY 8086]

Qual è il contenuto dei registri AX ed DX dopo l'esecuzione del seguente spezzone di codice

```
beta DW A34DH
var1 DW A82BH
. . .
code_sgm segment
mov AX, beta        ; in AX il          della variabile beta
push AX             ;    A34DH  =  il contenuto di AX viene poi posto nello stack.
mov DX, var1
add AX, 2
add DX, 2           ; DX contiene, dopo questa istruzione, il valore var1+2 = A82DH;
push DX             ; tale valore viene qui posto nello stack...
pop AX              ; ...da dove viene prelevato e posto in AX (quindi AX= A82DH)
pop DX              ; Infine prelievo dallo stack: DX= A34DH
. . .
code_sgm ends
```

NOTA: L'istruzione ADD AX,2 non ha effetto sul risultato, poiché essa segue il push di AX.



Risposta:

AX = A82DH

DX = A34DH

⑤ [8086]

Si consideri un'architettura 8086. Tenendo presenti le caratteristiche di accesso alla memoria in tale architettura, si indichi il numero di cicli di bus necessari per leggere:

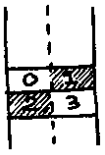
un byte all'indirizzo 00410 → 1

un byte all'indirizzo 00313 → 1

una word all'indirizzo 0020C → 1

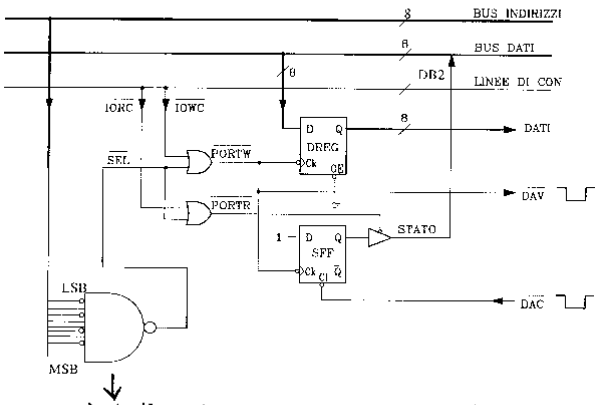
una word all'indirizzo 0020F → 2

La lettura di Word a indirizzi dispari richiede due cicli di bus. Nel primo $\overline{BHE} = 0$ e $A_0 = 1$, nel secondo $\overline{BHE} = 1$ e $A_0 = 0$. La parola viene letta un byte alla volta



⑥ [INPUT/OUTPUT]

Si consideri l'interfaccia di I/O rappresentata in figura in cui le linee del bus dati sono indicate con DB7 (MSB), ..., DB1, DB0 (LSB).



ind. di porta: 01011010B = 5AH

Si legga da programma lo stato dell'interfaccia. In caso di stato attivo (bit di stato = 1) si scriva il valore 37h nel registro DREG, altrimenti si scriva su tale registro il valore 49h.

Indicare la sequenza di istruzioni.

L'interfaccia possiede un unico indirizzo di porta 5AH (cfr. Figura). La scrittura su tale porta (istruzione OUT) permette di trasferire un dato di 8 bit nel registro dati DREG; la lettura (istr. IN) permette di acquisire lo stato, che viaggia sul DB2 del bus dati

```
        IN AL, 5AH              ; Lettura (dello stato)
        AND AL, 00000100B       ; si isola il bit di stato (DB2)
        JNZ STATUS_ON           ; controllo stato
STATUS_OFF: MOV AL, 49H         ; caso stato = 0
        JMP OUTPUT_AL           ;
STATUS_ON: MOV AL, 37H          ; caso stato = 1
OUTPUT_AL: OUT 5AH, AL          ; scrittura (del dato)
```

(17) [8086]

In un'architettura 8086 (organizzazione di memoria little endian) ~~il vettore~~ la tabella delle interruzioni è memorizzato a partire dall'indirizzo 00000h. L'indirizzo di ciascuna routine di interruzione è indicato considerando prima l'IP e poi il CS. Il seguente schema rappresenta ipotetici valori contenuti in una porzione di tale ~~vettore~~ tabella. Si determini l'indirizzo fisico (su 20 bit) della routine di gestione dell'interruzione numero 9. Si indichino nella tabella i valori di IP e di CS corrispondenti a tale routine. Si supponga successivamente di voler impiegare una nuova routine per gestire ~~tale~~ l'interruzione, che si trova a partire dall'indirizzo fisico 12344h. Riportare nella terza colonna del seguente schema i nuovi valori contenuti ~~nel vettore~~ nella tabella delle interruzioni (più soluzioni sono possibili).

	Indirizzo Memoria	Valore Attuale	Valore Futuro	
	...	...		
	00020h	00h		
	00021h	12h		
	00022h	06h		
	00023h	2Ah		
IP_9	00024h	10h	04	IP_9^{new}
IP_9	00025h	03h	00	IP_9^{new}
CS_9	00026h	01h	34	CS_9^{new}
CS_9	00027h	12h	12	CS_9^{new}
	00028h	07h		
	00029h	1Eh		
	0002Ah	02h		
	0002Bh	30h		
	0002Ch	10h		
	0002Dh	1Ah		
	...		...	

L'indirizzo del vettore di interruzione è in decimale

(00000H +) 9 × 4 = 36

9 → INT TYPE

4 → numero di byte di ogni vettore: 2 byte per IP, 2 byte per CS.

In binario, moltiplicare per 4 equivale a fare lo shift a sinistra di due cifre

0000 1001 ⟶ 0010 0100 ovvia 09H × 4 = 24H

(0 9 → 2 4)

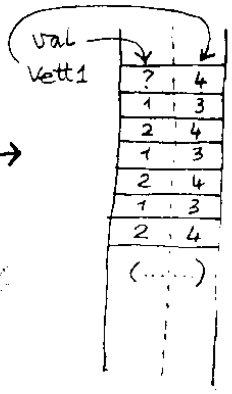
Segue (little endian) che IP_9 = 0310H e CS_9 = 1201H La routine dell'int 9 è dunque all'indirizzo fisico 12010 + 00310 = = 12320H. Infine, una possibile soluzione CS:IP compatibile con l'indirizzo fisico 12344H è 1234 0004

(8) [ASSEMBLY 8086]

Quale valore contiene val al termine dell'esecuzione delle seguenti linee di codice assembler 8086?

```
DSEG    SEGMENT
val     DB          ?
vett1   DB          11 DUP (4, 1, 3, 2) ;
DSEG    ENDS
CSEG    SEGMENT
        ...         ...
        MOV         AH, 0
        MOV         CX, 11
        MOV         val, 0
        MOV         SI, 0
ciclo:  MOV         AL,vett1[SI]
        ADD         val, AL
        INC         SI
        LOOP        ciclo
        SUB         val, 0
...     ...
CSEG    ENDS
```

duplica fin 11 volte in memoria il quartetto di byte 4,1,3,2 ⇝

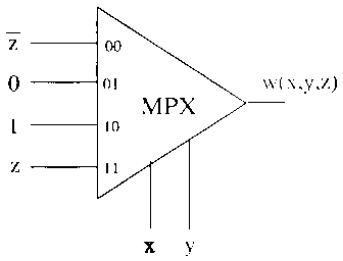


Il programma esegue la somma dei primi 11 byte del vettore vett1, ponendo il risultato in val. Il risultato è dunque val = 4+1+3+2+4+1+3+2+4+1+3 = 28

NOTA L'istruzione LOOP genera un salto all'etichetta "ciclo" ogni volta che, decrementato di 1 il valore di CX, trova che CX > 0. Dato che inizialmente è CX = 11, vengono eseguiti esattamente 10 salti a "ciclo"

(9) [ALGEBRA DELLE RETI E RETI COMBINATORIE]

Si faccia riferimento allo schema basato su multiplexer riportato nella parte sinistra della figura. Indicare il valore dell'uscita w in corrispondenza di tutte le configurazioni d'ingresso xyz, completando la tabella della verità nella parte destra della figura. Scrivere la funzione $w(x, y, z)$ nella forma canonica "somma di prodotti", e trovarne - ove sia possibile - un'espressione semplificata.



x y z	w(x,y,z)
000	1
001	0
010	0
011	0
100	1
101	1
110	0
111	1

$w(x,y,z) = x\bar{y}\bar{z} + xy\bar{z} + x\bar{y}z + xyz =$

$= \bar{y}\bar{z}(\underbrace{\bar{x}+x}_{1}) + xz(\underbrace{\bar{y}+y}_{1}) = xz + \bar{y}\bar{z}$