```
        MOV BX,0
        MOV CX,STRLEN
CICLO_EXT:
        MOV AL,STRINGA[BX]
        MOV SI,0
CICLO_INT:      (*)
        MOV AH,LUT[SI]
        CMP AH,AL       SE REPLACE_VOWEL
        INC SI
        CMP SI,5  ; le vocali sono 5!
        JNE CICLO_INT
        JMP NEW_EXT_ITERATION
REPLACE_VOWEL: MOV STRINGA[BX],'*'
NEW_EXT_ITERATION: INC BX
        LOOP CICLO_EXT
```

NOTA BENE: riporto solo le soluzioni in forma scarna, e quasi senza commento. Lo studente dovrà invece risolvere gli esercizi ordinatamente, e con adeguato spazio ai commenti.

indirizzo logico

	DS			
2C6A:0000	50='P'	41='A'	50	45
0004	52	4F	50	4F
0008	4C	49	19	00
000C	A8	07	C6	36
0010	19	00	A8	07
0014	C6	36	19	00
0018	A8	07	C6	36
001C	01	00	02	00

(N.B. l'8086 ha ordinamento "Little-endian")

(*) Con LUT DB 'AEIOU'

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

Calcolatori Elettronici I — a.a. 2005–2006

Compito del 5 aprile 2006

1

Cognome e Nome dello studente: Carlo Colombo

1. rappresentazione, CPU, hardware/software 8086

(a). Data la seguente definizione di segmento dati in assembly 8086:

```
0000  DATI SEGMENT
0000  STRINGA DB 'PAPEROPOLI'        <- alloca 10 byte
000A  STRINGA_END LABEL BYTE
000A  STRLEN EQU STRINGA_END-STRINGA   36C6h
000A  BUFFER DW 3 DUP (25 07A8h,0011011011000110B)   <- alloca 3x6=18 byte
001C  VAR DW 1,2                       <- alloca 2x2 = 4 byte
0020  DATI ENDS
```

Location counter

= 32 = 10 + 18 + 4

+ = 0019h

1. Indicare il valore del location counter in corrispondenza di ciascuna riga del segmento;
2. Determinare l'indirizzo fisico di VAR nel caso in cui DS sia pari a *2C6Ah*; 2C6A0 + 0001C = 2C6BCh
3. Riportare in esadecimale (si ricordi che 'A'=*41h*) il contenuto completo della memoria allocata per il segmento.

(b). Data inoltre l'istruzione `mov VAR, offset VAR`:

indirizzamento diretto su VAR; indirizzamento immediato

nel byte all'indirizzo 2C6BCh viene scritto 1Ch al posto di 01h

1. Fornirne il significato, specificando i modi di indirizzamento coinvolti e mostrando come cambia il contenuto della memoria dopo la sua esecuzione;
2. Servendosi della tabella sotto riportata, trovare la codifica di macchina in esadecimale dell'istruzione, sapendo che la sua codifica generale è [*1100011w*] [*mod 000 r/m*] [*disp-lo*] [*disp-hi*] [*data-lo*] [*data-hi*];
3. Determinare il numero di cicli di bus richiesti dal fetch e dall'esecuzione dell'istruzione, sapendo che il valore del registro IP ad essa corrispondente durante l'esecuzione è *6AB3h*.

1 | 00 | 110 | 1C | 00 | 1C | 00

	mod					
r/m	(00)	01	10	11		reg
000	BX+SI	BX+SI+D8	BX+SI+D16	AL	AX	000
001	BX+DI	BX+DI+D8	BX+DI+D16	CL	CX	001
010	BP+SI	BP+SI+D8	BP+SI+D16	DL	DX	010
011	BP+DI	BP+DI+D8	BP+DI+D16	BL	BX	011
100	SI	SI+D8	SI+D16	AH	SP	100
101	DI	DI+D8	DI+D16	CH	BP	101
110	D16	BP+D8	BP+D16	DH	SI	110
111	BX	BX+D8	BX+D16	BH	DI	111
				w=0	w=1	

indirizzamento diretto (1° operando)

poiché l'istruzione è ad indirizzo dispari, la BIU ne preleverà prima un solo byte, poi 2 word ad indirizzi pari, e poi 1 byte ad indirizzo pari. In tutto 1+2+1=4 cicli per il fetch, cui va aggiunto 1 ciclo per lo store del dato, che è a ind. pari. Totale: 5 cicli

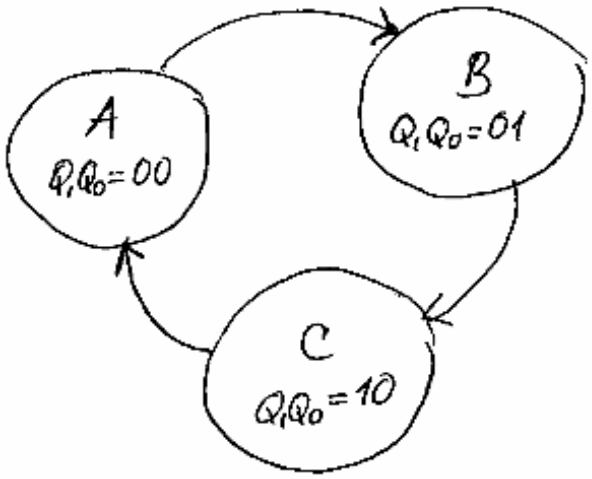
(c). Infine, scrivere un programma assembly 8086 che, scandendo carattere per carattere la stringa STRINGA, trasformi in '*' tutte le vocali in essa presenti.

2. reti combinatorie e sequenziali 0%

(a). Disegnare il diagramma degli stati e la tabella di verità per un contatore binario "up" modulo 3.

(b). Progettare il contatore di cui sopra facendo uso di flip-flop di tipo "D" e di multiplexer.

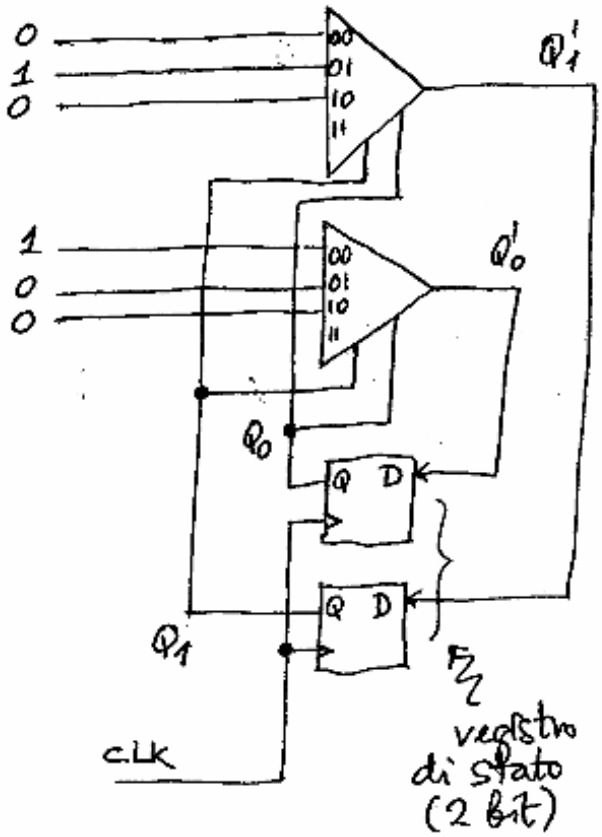
(c). Quali modifiche bisogna apportare al progetto per consentire sia il conteggio "up" che quello "down"?



Q_1 Q_0	Q_1' Q_0'
0 0	0 1
0 1	1 0
1 0	0 0
1 1	X X

Sono funzioni combinatorie di 2 variabili che possono essere realizzate in vari modi. Ad es. con multiplexer (compito #1), o con porte elementari (Compito #2)

Realizzazione con multiplexer



Per contare sia "up" che "down", bisogna introdurre un ingresso ($U/\overline{D}$), per consentire la scelta del verso (avanti o all'indietro) di percorrenza dell'automa. La tabella avrà 3 ingressi, e quindi la rete combinatoria che determina la transizione di stato sarà un po' più complicata.

Realizzazione con porte

Se si pongono i don't care a 0, si ottiene

$$Q_1' = \overline{Q}_1 Q_0 \quad , \quad Q_0' = \overline{Q}_1 \overline{Q}_0 = \overline{Q_1 + Q_0}$$

Se il don't care su Q_1' è posto a 1, c'è un'ulteriore semplificazione:

$$Q_1' \equiv Q_0$$