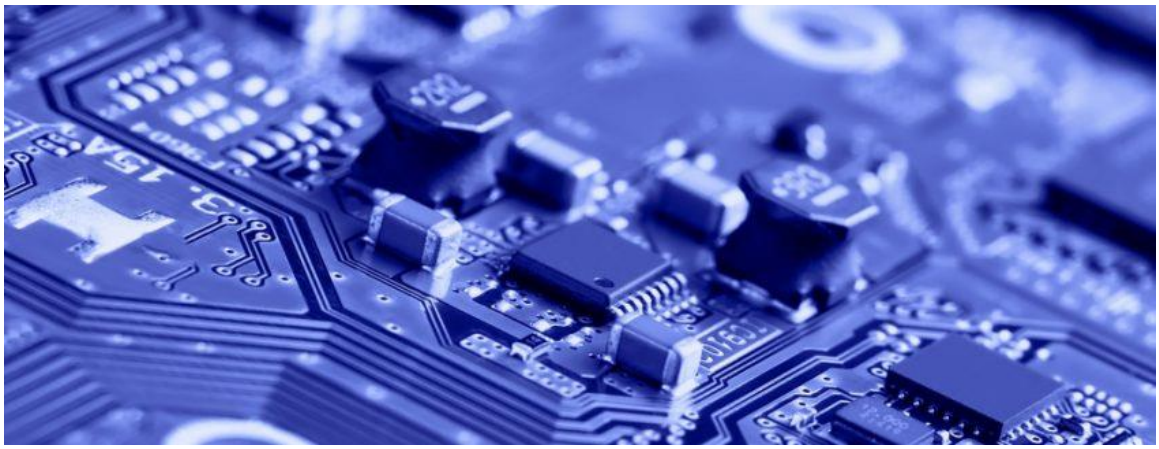


Συστήματα Μικροϋπολογιστών
2η Ομάδα Ασκήσεων

Καμπουγέρης Χαράλαμπος | el20098
Κουστένης Χρίστος | el20227

Άσκηση 1

(α)

Στις παρακάτω εικόνες βλέπουμε το πρόγραμμα που πραγματοποιεί τη λειτουργία που περιγράφεται στην εκφώνηση.

```
START:
       IN 10H ; disable memory protection
       MVI A,00H ; (A) <- 0
       LXI H,0900H ; (H) <- 09 , (L) <- 00
       MOV M,A  ; M((H)(L)) <- (A)
STORE:
       INR A   ; (A) <- (A) + 1
       INX H   ; ((H)(L)) <- ((H)(L)) + 1
       MOV M,A ; M((H)(L)) <- (A)
       CPI 7FH ; Is A < 127 ?
       JC STORE ; If yes then repeat

       END
```

```
START:
0800   DB     IN 10H
0801   10
0802   3E     MVI A,00H
0803   00
0804   21     LXI H,0900H
0805   00
0806   09
0807   77     MOV M,A

STORE:
0808   3C     INR A
0809   23     INX H
080A   77     MOV M,A
080B   FE     CPI 7FH
080C   7F
080D   DA     JC STORE
080E   08
080F   08
```

08F7	00	08F8	00	08F9	00	08FA	00	08FB	00	08FC	00	08FD	00	08FE	00	08FF	00	0900	00	0901	01	0902	02	0903	03
0904	04	0905	05	0906	06	0907	07	0908	08	0909	09	090A	0A	090B	0B	090C	0C	090D	0D	090E	0E	090F	0F	0910	10
0911	11	0912	12	0913	13	0914	14	0915	15	0916	16	0917	17	0918	18	0919	19	091A	1A	091B	1B	091C	1C	091D	1D
091E	1E	091F	1F	0920	20	0921	21	0922	22	0923	23	0924	24	0925	25	0926	26	0927	27	0928	28	0929	29	092A	2A
092B	2B	092C	2C	092D	2D	092E	2E	092F	2F	0930	30	0931	31	0932	32	0933	33	0934	34	0935	35	0936	36	0937	37
0938	38	0939	39	093A	3A	093B	3B	093C	3C	093D	3D	093E	3E	093F	3F	0940	40	0941	41	0942	42	0943	43	0944	44
0945	45	0946	46	0947	47	0948	48	0949	49	094A	4A	094B	4B	094C	4C	094D	4D	094E	4E	094F	4F	0950	50	0951	51
0952	52	0953	53	0954	54	0955	55	0956	56	0957	57	0958	58	0959	59	095A	5A	095B	5B	095C	5C	095D	5D	095E	5E
095F	5F	0960	60	0961	61	0962	62	0963	63	0964	64	0965	65	0966	66	0967	67	0968	68	0969	69	096A	6A	096B	6B
096C	6C	096D	6D	096E	6E	096F	6F	0970	70	0971	71	0972	72	0973	73	0974	74	0975	75	0976	76	0977	77	0978	78
0979	79	097A	7A	097B	7B	097C	7C	097D	7D	097E	7E	097F	7F	0980	00	0981	00	0982	00	0983	00	0984	00	0985	00

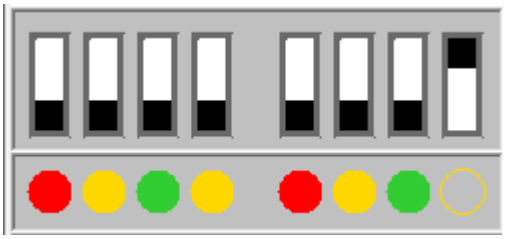
Το πρόγραμμα αυτό αφου απενεργοποιήσει την προστασία μνήμης με την εντολη IN 10H μηδενίζει τον A καταχωρητή (MVI A,00H), μεταφέρει τη διεύθυνση 0900H στο ζεύγος καταχωρητών HL(LXI H,0900H) και μεταφέρει το περιεχόμενο του A στη διεύθυνση μνήμης που περιέχει το ζεύγος καταχωρητών HL(MOV M,A). Στη συνέχεια, αυξάνει κατα 1 το A(INR A) καθώς και τη διεύθυνση που περιέχεται στους HL(INX H).Έπειτα, μεταφέρει το περιεχόμενο του A στην νέα θέση μνήμης (MOV M,A) συγκρίνει το περιεχόμενο του A με τον αριθμό $7F_{16} = 127_{10}$ και αν η τιμή της σημαίας CY γίνει ένα επαναλαμβάνει τα τελευταία βήματα αλλίως τερματίζει.

(β)

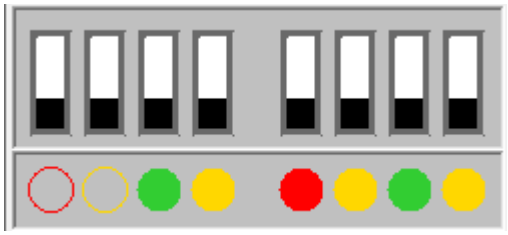
Κάθε θέση μνήμης περιέχει 8 bits. Επομένως, αφού χρησιμοποιήσαμε 128 θέσεις μνήμης και σε κάθε μία τοποθετήσαμε έναν αριθμό το πλήθος των bits για τα παραπάνω δεδομένα είναι 8 * 128 = 1024 bits. Το πλήθος των μονάδων που εντοπίζονται θα πρέπει να είναι 448. Άρα στον διπλό καταχωρητή BC θα πρέπει να αποθηκευτεί ο αριθμός $448_{10} = 111000000_2$. Για να ελέγξουμε το αποτέλεσμα μας μετά την απαρίθμηση των μονάδων στα δεδομένα επιδιώκουμε την εμφάνιση του περιεχομένου των καταχωρητών BC. Όταν το τελευταίο switch εισόδου είναι ON τότε εμφανίζεται το περιεχόμενο του B που περιέχει τα MSBs και όταν είναι OFF εμφανίζεται το περιεχόμενο του C που περιέχει τα LSBs. Πράγματι, διαπιστώνουμε ότι τα LEDs ακολουθώντας

αντίστροφη λογική είναι όλα αναμμένα εκτός από το τελευταίο για τον καταχωρητή B και για τον C όλα αναμμένα εκτός από τα δύο πρώτα. Ακολουθεί το πρόγραμμα σε 8085 περιβάλλον μLAB και τα αποτελέσματα εξόδου.

```
;(a)

START:
       IN 10H ; disable memory protection
       MVI A,00H ; (A) <- 0
       LXI H,0900H ; (H) <- 09 , (L) <- 00
       MOV M,A  ; M((H)(L)) <- (A)
STORE:
       INR A   ; (A) <- (A) + 1
       INX H   ; ((H)(L)) <- ((H)(L)) + 1
       MOV M,A ; M((H)(L)) <- (A)
       CPI 7FH ; Is A < 127 ?
       JC STORE ; If yes then repeat

;(b)

       LXI B,0000H
RESET_COUNTER:
       MOV A,M
       MVI D,09H
CHECK_DIGITS:
       DCR D ; Next digit of the number
       JZ PREVIOUS_ONE ; If we are out of digits next memory location, next number
       RRC ; Search for 1
       JNC CHECK_DIGITS
COUNT:
       INX B ; we have 1 additional 1
       JMP CHECK_DIGITS
PREVIOUS_ONE:
       DCR L ;check the previous memory slot
       JNZ RESET_COUNTER
OUTPUT:
       LDA 2000H
       RRC
       JC MSB
LSB:
       MOV A,C
       STA 3000H
       JMP OUTPUT
MSB:
       MOV A,B
       STA 3000H
       JMP OUTPUT

       END
```



Περιεχόμενο B καταχωρητή



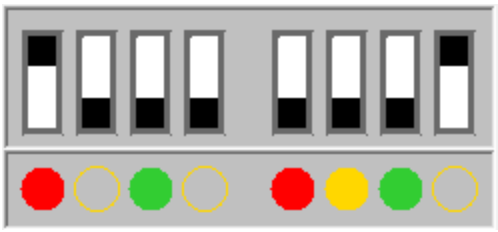
Περιεχόμενο C καταχωρητή

(γ)

Ως συνέχεια του ήδη υπάρχοντος κώδικα φροντίζουμε το πρόγραμμα κατόπιν θέτοντας σε ON κατάσταση το πρώτο από αριστερά dip switch να διαφεύγει από το loop στο οποίο τό είχαμε εισάγει για το ερώτημα (β) ώστε να εμφανίζει δυναμικά το περιεχόμενο των καταχωρητών B και C. Γνωρίζουμε ότι οι αριθμοί που εισήγαμε στη μνήμη είναι ταξινομημένοι και η απάντηση στο ερώτημα πόσοι περιέχονται μεταξύ 10_{hex} και 60_{hex} θα ήταν προφανής. Ωστόσο, επιδιώκουμε μια γενικότερη επίλυση η οποία εντοπίζει το πλήθος όλων των αριθμών στη μνήμη που βρίσκονται στο επιθυμητό διάστημα ακόμη και αν αυτοί είναι τυχαία τοποθετημένοι σε αυτή. Ακολουθεί ο κώδικας που πραγματοποιεί την καταμέτρηση αυτή. Στην εικόνα έχει συμπεριληφθεί και το τέλος του ερωτήματος (β) όπου φαίνεται η ομαλή μετάβαση από τον (β) στο (γ) μέσω του dip switch.

```
OUTPUT:
       LDA 2000H
       RRC
       JC MSB
LSB:
       MOV A,C
       STA 3000H
       LDA 2000H
       RLC
       JC COUNT_NUM
       JMP OUTPUT
MSB:
       MOV A,B
       STA 3000H
       LDA 2000H
       RLC
       JC COUNT_NUM
       JMP OUTPUT

;(c)

COUNT_NUM:
       MVI E,7FH
       MVI D,00H     ;Initialise D
       MOV A,M       ;Load first Number ((H)(L))=0900H from before
STATEMENT:
       CPI 10H       ;If A<10H skip this nummber else continue
       JC SKIP
       CPI 61H       ;If A>60H skip this number else continue and add it to the total
       JNC SKIP
       INR D         ;Count
       JZ CHECK_LAST
SKIP:
       INR L         ;Load next number
       MOV A,M
       DCR E
       JZ CHECK_LAST   ;If E=00H CHECK_LAST else jump to STATEMENT
       JMP STATEMENT
CHECK_LAST:
       CPI 10H
       JC OUTPUT_2
       CPI 61H
       JNC OUTPUT_2
       INR D
OUTPUT_2:
       MOV A,D
       STA 3000H

       END
```

Το πλήθος αποθηκεύται στον D καταχωρητή όπως ζητείται και έπειτα προβάλλεται στην έξοδο. Από τα LEDs διαπιστώνουμε ότι στον D έχει αποθηκευτεί ο αριθμός $01010001_{BIN} = 51_{HEX} = 81_{DEC}$ που είναι και η σωστή απάντηση αφού $60_{HEX} - 10_{HEX} + 1 = 51_{HEX}$.



Άσκηση 2

```
        MVI     D,C8H           ;200 * 0.1s = 20s
        LXI     B,0064H         ;100*10^(-3) = 0.1s

FIRST_CHECK:
        LDA     2000H
        RAL                     ;check MSB -> CY
        JC      FIRST_CHECK     ;if ON repeat
        JMP     FIRST_OFF

FIRST_OFF:
        LDA     2000H
        RAL
        JC      FIRST_ON        ;if ON
        JMP     FIRST_OFF       ;if OFF, repeat until ON

FIRST_ON:
        LDA     2000H
        RAL
        JC      FIRST_ON        ;if ON, repeat until OFF
        JMP     SECOND_OFF      ;if OFF

SECOND_OFF:
        LDA      2000H
        RAL
        JC      SECOND_ON       ;if ON, check for next OFF
        MVI     A,00H           ;if else
        STA     3000H           ;turn on LED
        CALL    DELB            ;wait 0.1sec
        DCR     D               ;decrease D, hence the time
        MOV     A,D
        CPI     00H             ;if D = 0, time is up (Z = 1)
        JZ      AGAIN           ;repeat
        JMP     SECOND_OFF      ;repeat until time is up

SECOND_ON:
        LDA     2000H
        RAL
        JNC     RESTART_TIME    ;if OFF, restart time
        MVI     A,00H
        STA     3000H           ;if ON, turn on LED
        CALL    DELB            ;wait 0.1sec
        DCR     D               ;decrease D
        MOV     A,D
        CPI     00H             ;if D = 0, time is up
        JZ      AGAIN           ;repeat
        JMP     SECOND_ON       ;repeat until time is up


AGAIN:
        MVI     A,FFH           ;turn off LED
        STA     3000H
        MVI     D,C8H           ;restart D = 200
        JMP     FIRST_OFF       ;repeat from the start

RESTART_TIME:
        MVI D,C8H               ;restart time
        JMP SECOND_OFF
END
```

```
0800   16     MVI D,C8H
0801   C8
0802   01     LXI B,0064H
0803   64
0804   00

FIRST_CHECK:
0805   3A     LDA 2000H
0806   00
0807   20
0808   17     RAL
0809   DA     JC FIRST_CHECK
080A   05
080B   08
080C   C3     JMP FIRST_OFF
080D   0F
080E   08

FIRST_OFF:
080F   3A     LDA 2000H
0810   00
0811   20
0812   17     RAL
0813   DA     JC FIRST_ON
0814   19
0815   08
0816   C3     JMP FIRST_OFF
0817   0F
0818   08

FIRST_ON:
0819   3A     LDA 2000H
081A   00
081B   20
081C   17     RAL
081D   DA     JC FIRST_ON
081E   19
081F   08
0820   C3     JMP SECOND_OFF
0821   23
0822   08

SECOND_OFF:
0823   3A     LDA 2000H
0824   00
0825   20
0826   17     RAL
0827   DA     JC SECOND_ON
0828   3C
0829   08
082A   3E     MVI A,00H
082B   00
082C   32     STA 3000H
082D   00
082E   30
082F   CD     CALL DELB
0830   30
0831   04
0832   15     DCR D
0833   7A     MOV A,D
0834   FE     CPI 00H
0835   00
```

```
0836   CA     JZ AGAIN
0837   55
0838   08
0839   C3     JMP SECOND_OFF
083A   23
083B   08

SECOND_ON:
083C   3A     LDA 2000H
083D   00
083E   20
083F   17     RAL
0840   D2     JNC RESTART_TIME
0841   5F
0842   08
0843   3E     MVI A,00H
0844   00
0845   32     STA 3000H
0846   00
0847   30
0848   CD     CALL DELB
0849   30
084A   04
084B   15     DCR D
084C   7A     MOV A,D
084D   FE     CPI 00H
084E   00
084F   CA     JZ AGAIN
0850   55
0851   08
0852   C3     JMP SECOND_ON
0853   3C
0854   08

AGAIN:
0855   3E     MVI A,FFH
0856   FF
0857   32     STA 3000H
0858   00
0859   30
085A   16     MVI D,C8H
085B   C8
085C   C3     JMP FIRST_OFF
085D   0F
085E   08

RESTART_TIME:
085F   16     MVI D,C8H
0860   C8
0861   C3     JMP SECOND_OFF
0862   23
0863   08
```

Άσκηση 3

(i)

```
START:
      LDA 2000H
      MVI B,08H
SEARCH:
      RRC
      JC O_INIT ; if first on switch is found then go to output initialization
      DCR B  ; search the next digit
      JZ NO_LEDS ; if you reach zero then no switch was on so all LEDs OFF
      JMP SEARCH   ; continue searching
O_INIT:
      MVI A,80H ; (A)<- 10000000
      DCR B  ; (B)<- just making sure we have the right place of LED to print
OUTPUT:
      RRC ; move the 1 from end to start
      DCR B ; decrease B until it reaches 0 then we are at the right place
      JNZ OUTPUT
O_FINAL:
      CMA ; (A)<-(A')
      STA 3000H
      JMP START

NO_LEDS:
      MVI A,00H
      CMA ; (A)<-(A')
      STA 3000H
      JMP START

      END
```

(ii)

```
START:
       CALL KIND ; read the number pressed and put it into A reg
       CPI 00H   ;If (A)<=0 is pressed go to off mode
       JZ OFF
       CPI 09H   ;If (A)>=9 is pressed go to off mode
       JNC OFF
       MOV B,A   ;Save A register to B. We will need A somewhere else
       MVI A,00H  ;We initiliaze A
ON_LOOP:
       DCR B
       JZ OUTPUT
       RLC ; basically we multiply by 2 to create one more zero at the LSB
       INR A ; then we add 1 to make the last 0 -> 1
       JMP ON_LOOP
OUTPUT:
       STA 3000H
       JMP START  ; read the keyboard again

OFF:
       MVI A,FFH  ; Do not light up anything
       STA 3000H
       JMP START

       END
```

(iii)

Χρησιμοποιώντας τους παρακάτω πίνακες από το αρχείο “Εισαγωγή στο TSIK” διαπιστώσαμε τα εξής :

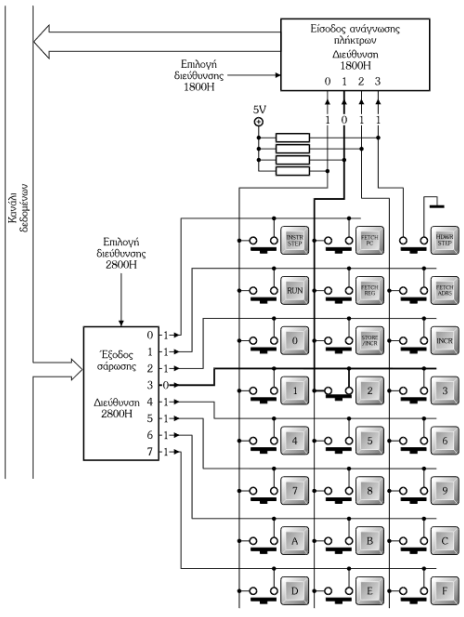
- ✓ Τον κωδικό που αντιστοιχούσε σε κάθε πλήκτρο κάθε γραμμής (Πίνακας 1).
- ✓ Τον δεκαεξαδικό κωδικό του κενού για τα ψηφία της οθόνης που είναι το 10 (Πίνακας 7)
- ✓ Τον αριθμό κάθε γραμμής στην έξοδο σάρωσης ώστε να ξέρουμε ποιο ψηφίο θα μηδενίσουμε.

Πίνακας 1. Κωδικοί των πλήκτρων για τη ρουτίνα KIND

Πλήκτρο	Κωδικός	Πλήκτρο	Κωδικός
0	00	C	0C
1	01	D	0D
2	02	E	0E
3	03	F	0F
4	04	FETCH REG	80
5	05	DECR	81
6	06	FETCH ADRS	82
7	07	STORE/INCR	83
8	08	RUN	84
9	09	FETCH PC	85
A	0A	INSTR STEP	86
B	0B	HDWR STEP	F7

Πίνακας 7. Κωδικοί χαρακτήρων για τη ρουτίνα DCD

Χαρακτήρας	Δεκαεξαδικός κώδικας	Χαρακτήρας	Δεκαεξαδικός κώδικας
0	00	F	0F
1	01	(κενό)	10
2	02	H	11
3	03	L	12
4	04	u	13
5	05	P	14
6	06	o	15
7	07	U	16
8	08	-	17
9	09	c	18
A	0A	I	19
b	0B	8.	1A
c	0C	r	1B
d	0D	-	1C
E	0E		



Σχήμα 1. Διάγραμμα interface του πληκτρολογίου

Με βάση τα παραπάνω καταλήξαμε στον ακόλουθο κώδικα :

```
START:
        IN 10H
        LXI H,0A00H
        MVI B,04H
L1:
        MVI M,10H
        INX H
        DCR B
        JNZ L1

ROW_7:
        MVI A,7FH ; (7F)hex = (01111111)bin to pick the 7TH row
        STA 2800H ;
        LDA 1800H ;
        ANI 07H   ;

        MVI C,0DH  ; D
        CPI 06H
        JZ OUTPUT  ;

        MVI C,0EH  ; E
        CPI 05H
        JZ OUTPUT

        MVI C,0FH  ; F
        CPI 03H ;
        JZ OUTPUT ;

ROW_6:
        MVI A,BFH ; (BF)hex = (10111111)bin to pick the 6TH row
        STA 2800H ;
        LDA 1800H
        ANI 07H


        MVI C,0AH  ; A
        CPI 06H
        JZ OUTPUT  ;

        MVI C,0BH  ; B
        CPI 05H
        JZ OUTPUT

        MVI C,0CH  ; C
        CPI 03H ;
        JZ OUTPUT ;


ROW_5:
        MVI A,DFH ; (DF)hex = (11011111)bin to pick the 5TH row
        STA 2800H ;
        LDA 1800H
        ANI 07H

        MVI C,07H  ; 7
        CPI 06H
        JZ OUTPUT  ;

        MVI C,08H  ; 8
        CPI 05H
        JZ OUTPUT

        MVI C,09H  ; 9
        CPI 03H ;
        JZ OUTPUT ;
```

```
ROW_4:
        MVI A,EFH ; (EF)hex = (11101111)bin to pick the 4TH row
        STA 2800H ;
        LDA 1800H
        ANI 07H


        MVI C,04H  ; 4
        CPI 06H
        JZ OUTPUT  ;

        MVI C,05H  ; 5
        CPI 05H
        JZ OUTPUT

        MVI C,06H  ; 6
        CPI 03H ;
        JZ OUTPUT ;


ROW_3:
        MVI A,F7H ; (F7)hex = (11110111)bin to pick the 3RD row
        STA 2800H ;
        LDA 1800H
        ANI 07H


        MVI C,01H  ; 1
        CPI 06H
        JZ OUTPUT  ;

        MVI C,02H  ; 2
        CPI 05H
        JZ OUTPUT

        MVI C,03H  ; 3
        CPI 03H ;
        JZ OUTPUT ;

ROW_2:
        MVI A,FBH ; (FB)hex = (11111011)bin to pick the 2ND row
        STA 2800H ;
        LDA 1800H
        ANI 07H

        MVI C,00H  ; 0
        CPI 06H
        JZ OUTPUT  ;

        MVI C,83H  ; STORE/INCR
        CPI 05H
        JZ OUTPUT

        MVI C,81H  ; DECR
        CPI 03H ;
        JZ OUTPUT ;

ROW_1:
        MVI A,FDH ; (FD)hex = (11111101)bin to pick the 1ST row
        STA 2800H ;
        LDA 1800H
        ANI 07H

        MVI C,84H  ; RUN
        CPI 06H
        JZ OUTPUT  ;
```

```
        MVI C,80H  ; FETCH REG
        CPI 05H
        JZ OUTPUT

        MVI C,82H  ; FETCH ADDRESS
        CPI 03H ;
        JZ OUTPUT ;


ROW_0:
        MVI A,FEH ; (FE)hex = (11111110)bin to pick the 0 row
        STA 2800H ;
        LDA 1800H
        ANI 07H

        MVI C,86H  ; INSTR STEP
        CPI 06H
        JZ OUTPUT  ;

        MVI C,85H  ; FETCH PC
        CPI 05H
        JZ OUTPUT

        JMP START

OUTPUT:
        LXI H,0A04H
        MOV A,C
        ANI 0FH
        MOV M,A

        INX H
        MOV A,C
          ANI F0H ; (F0)HEX = (1111 0000)BIN
        RLC
        RLC
        RLC
        RLC
        MOV M,A

        LXI D,0A00H
        CALL STDM
        CALL DCD

        JMP START

        END
```

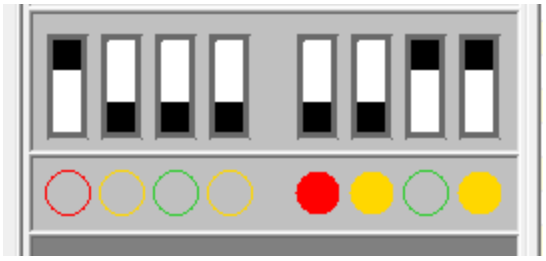
Άσκηση 4

```
START:
            ;X3
      LDA 2000H
      ANI 80H ;A3 (8th switch)
      RRC ;move A3 to 7th switch
      MOV B,A ;save A3(7th) -> B
      LDA 2000H
      ANI 40H ;B3 (7th switch)
      XRA B ;A3 xor B3 -> A
      RRC ;(6th)
      RRC ;(5th)
      RRC ;(4th)
      RRC ;(3rd)
      MOV C,A ;save (A3 xor B3)
      RLC ;(4th)
      MOV D,A ;save A(4th)
            ;X2
      LDA 2000H
      ANI 20H ;A2 (6th)
      RRC ;5th
      MOV B,A ;save A2(5th) -> B
      LDA 2000H
      ANI 10H ;B2 (5th)
      XRA B ;B2 xor A2 -> A
      RRC ;(4th)
      RRC ;(3rd)
      ORA C ;C(A3 xor B3) or A(B2 xor A2)
      ORA D ;[(A3 xor B3) or (A2 xor B2)](3rd) or D(4th)
      MOV D,A ;save A(3rd and 4th)
            ;X1
      LDA 2000H
      ANI 08H ;A1 (4th)
      RRC ;(3rd)
      MOV B,A ;save A1(3rd) -> B
      LDA 2000H
      ANI 04H ;B1(3rd)
      ANA B ;(A1 and B1)
      RRC ;(2nd)
      RRC ;(1st)
      MOV C,A ;save (A1 and B1)
      RLC ;(2nd)
      ORA D ;D(3rd, 4th) or (A1 and B1) (2nd)
      MOV D,A
            ;X0
      LDA 2000H
      ANI 02H ;A0 (2nd)
      RRC ;A0 (1st)
      MOV B,A ;save A0(1st) -> B
      LDA 2000H
      ANI 01H ;B0 (1st)
      ANA B ;(A0 and B0)
      ORA C ;(A0 or B0) or C
      ORA D

      CMA
      STA 3000H
      JMP START
      END
```

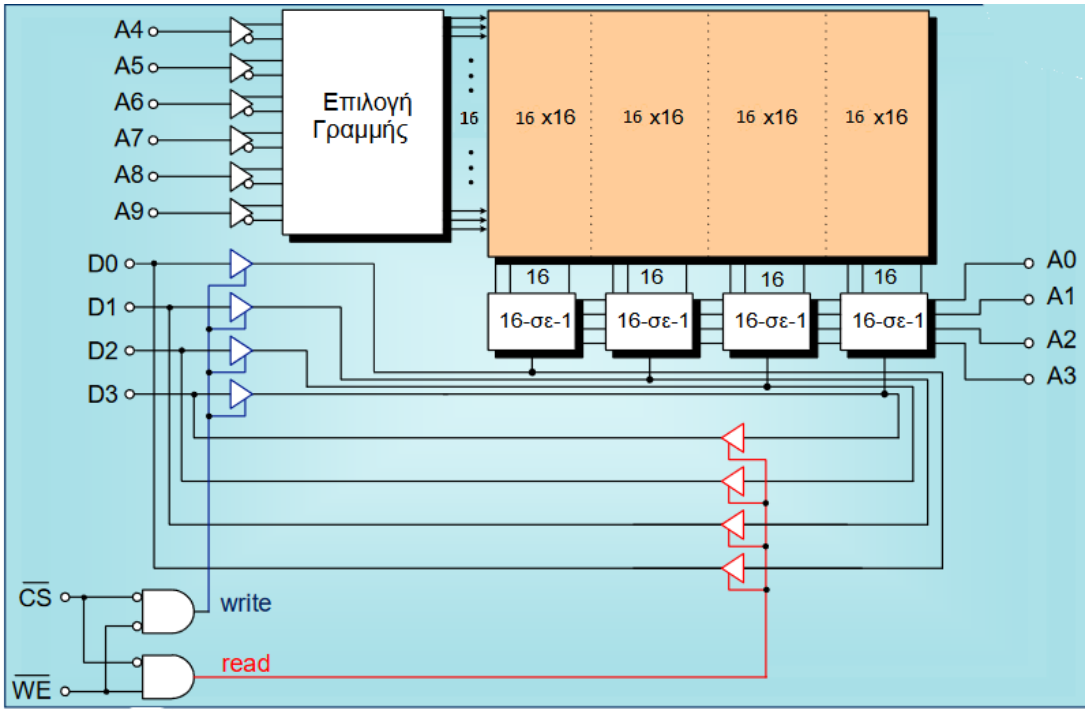
```
START:
0800   3A     LDA 2000H
0801   00
0802   20
0803   E6     ANI 80H
0804   80
0805   0F     RRC
0806   47     MOV B,A
0807   3A     LDA 2000H
0808   00
0809   20
080A   E6     ANI 40H
080B   40
080C   A8     XRA B
080D   0F     RRC
080E   0F     RRC
080F   0F     RRC
0810   0F     RRC
0811   4F     MOV C,A
0812   07     RLC
0813   57     MOV D,A
0814   3A     LDA 2000H
0815   00
0816   20
0817   E6     ANI 20H
0818   20
0819   0F     RRC
081A   47     MOV B,A
081B   3A     LDA 2000H
081C   00
081D   20
081E   E6     ANI 10H
081F   10
0820   A8     XRA B
0821   0F     RRC
0822   0F     RRC
0823   B1     ORA C
0824   B2     ORA D
0825   57     MOV D,A
0826   3A     LDA 2000H
0827   00
0828   20
0829   E6     ANI 08H
082A   08
082B   0F     RRC
082C   47     MOV B,A
082D   3A     LDA 2000H
082E   00
082F   20
0830   E6     ANI 04H
0831   04
0832   A0     ANA B
0833   0F     RRC
0834   0F     RRC
0835   4F     MOV C,A
0836   07     RLC
0837   B2     ORA D
0838   57     MOV D,A
0839   3A     LDA 2000H
```

```
083A   00
083B   20
083C   E6     ANI 02H
083D   02
083E   0F     RRC
083F   47     MOV B,A
0840   3A     LDA 2000H
0841   00
0842   20
0843   E6     ANI 01H
0844   01
0845   A0     ANA B
0846   B1     ORA C
0847   B2     ORA D
0848   2F     CMA
0849   32     STA 3000H
084A   00
084B   30
084C   C3     JMP START
084D   00
084E   08
```

Παραθέτουμε ένα παράδειγμα για είσοδο 10000011, όπως αναμένουμε έχουμε αποτέλεσμα X3=1, X2=1, X1=0,X0=1



Άσκηση 5



Παραπάνω φαίνεται η εσωτερική δομή μίας SRAM 256x4 bit (16x16x4). Από τον πίνακα της μνήμης επιλέγεται με βάση τις γραμμές διεύθυνσης A4-A7 μια από τις 16 γραμμές. Οι γραμμές D0-D3 αποτελούν τις γραμμές δεδομένων, οι οποίες συνδέονται με τον πίνακα της μνήμης μέσω τεσσάρων πολυπλεκτών 16-σε-1. Οι πολυπλέκτες αυτοί επιλέγουν μία από τις 16 τετράδες-στήλες του πίνακα μνήμης (μία ο καθένας), με βάση τις γραμμές διευθύνσεων A0-A3, όπου, σε συνδυασμό με την επιλεγμένη γραμμή του πίνακα, είτε εγγράφονται τα δεδομένα D0-D3 στις θέσεις αυτές, είτε διαβάζονται, δηλαδή μεταφέρονται στις γραμμές D0-D3, τα δεδομένα των θέσεων αυτών.

Για παράδειγμα αν είχαμε μία διεύθυνση A0…A7 = 0110 1001, τότε επιλέγεται η 9η (1001) γραμμή και η 6η (0110) τετράδα του πίνακα μνήμης. Όσον αφορά το πότε γίνεται εγγραφή ή ανάγνωση, αυτό καθορίζεται από τα τρία σήματα $\overline{CS}$, $\overline{RD}$, $\overline{WE}$. Όταν το σήμα $\overline{CS}$ γίνει 0, ενεργοποιείται η λειτουργία της μνήμης. Στη συνέχεια, αν το σήμα $\overline{WE}$ γίνει 0 και το $\overline{RD}$ γίνει 1, τότε θα ενεργοποιηθούν οι απομονωτές με μπλε περίγραμμα και θα έχουμε εγγραφή στην μνήμη. Αν το σήμα $\overline{WE}$ γίνει 1 και το $\overline{RD}$ γίνει 0, τότε θα ενεργοποιηθούν οι απομονωτές με κόκκινο περίγραμμα και θα έχουμε διάβασμα από τη μνήμη.

Άσκηση 6

Για την υλοποίηση χρησιμοποιούμε:

ROM: 2K x 8bit, 2K x 8bit, 4K x 8bit

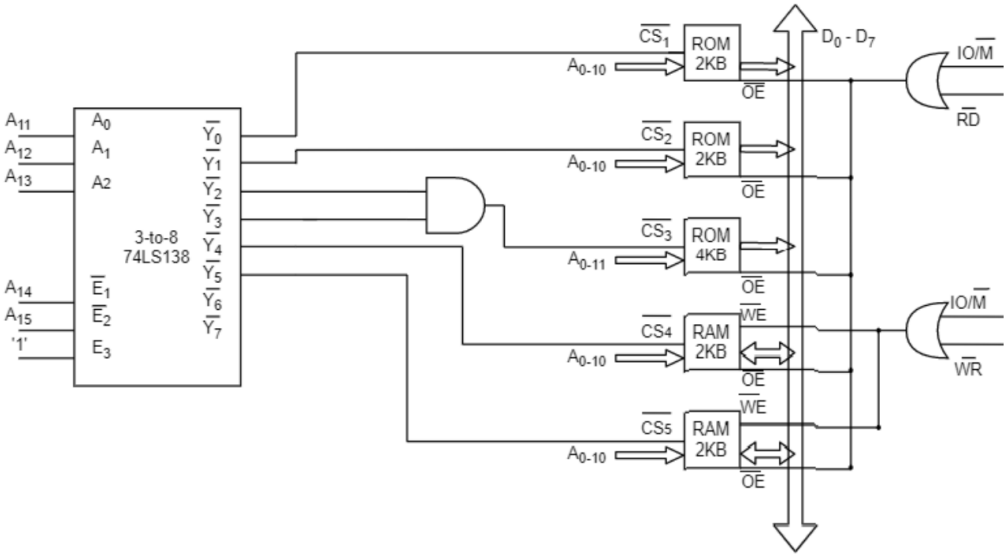
RAM: 2K x 8bit, 2K x 8bit

Υλοποιούμε το χάρτη μνήμης:

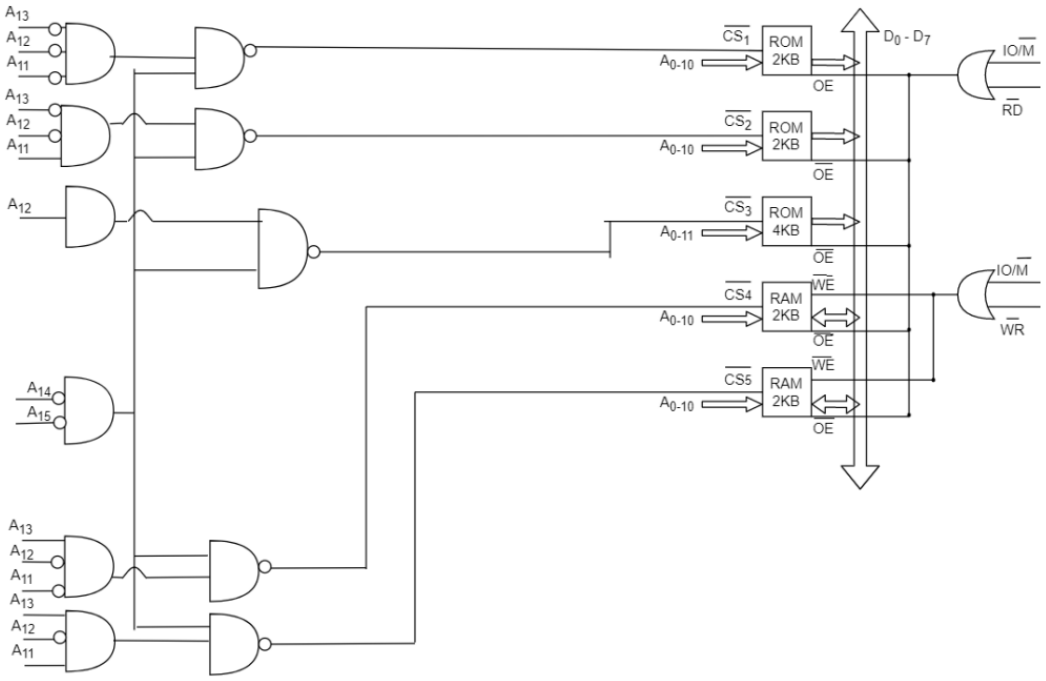
MEMORY	ADDRESS	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	0000H	0	0	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0
	07FFH	0	0	**0**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1
ROM	0800H	0	0	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0
	0FFFH	0	0	**0**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1
	1000H	0	0	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0
	1FFFH	0	0	**0**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1
	2000H	0	0	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0
RAM	27FFH	0	0	**1**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1
	2800H	0	0	**1**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0
	2FFFH	0	0	**1**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1

Παρατηρούμε ότι τα bit $A11$, $A12$, $A13$ χρησιμοποιούνται για την επιλογή του ολοκληρωμένου.

Α) Χρησιμοποιούμε αποκωδικοποιητή 3:8 και λογικές πύλες:



Β) Χρησιμοποιούμε μόνο λογικές πύλες:



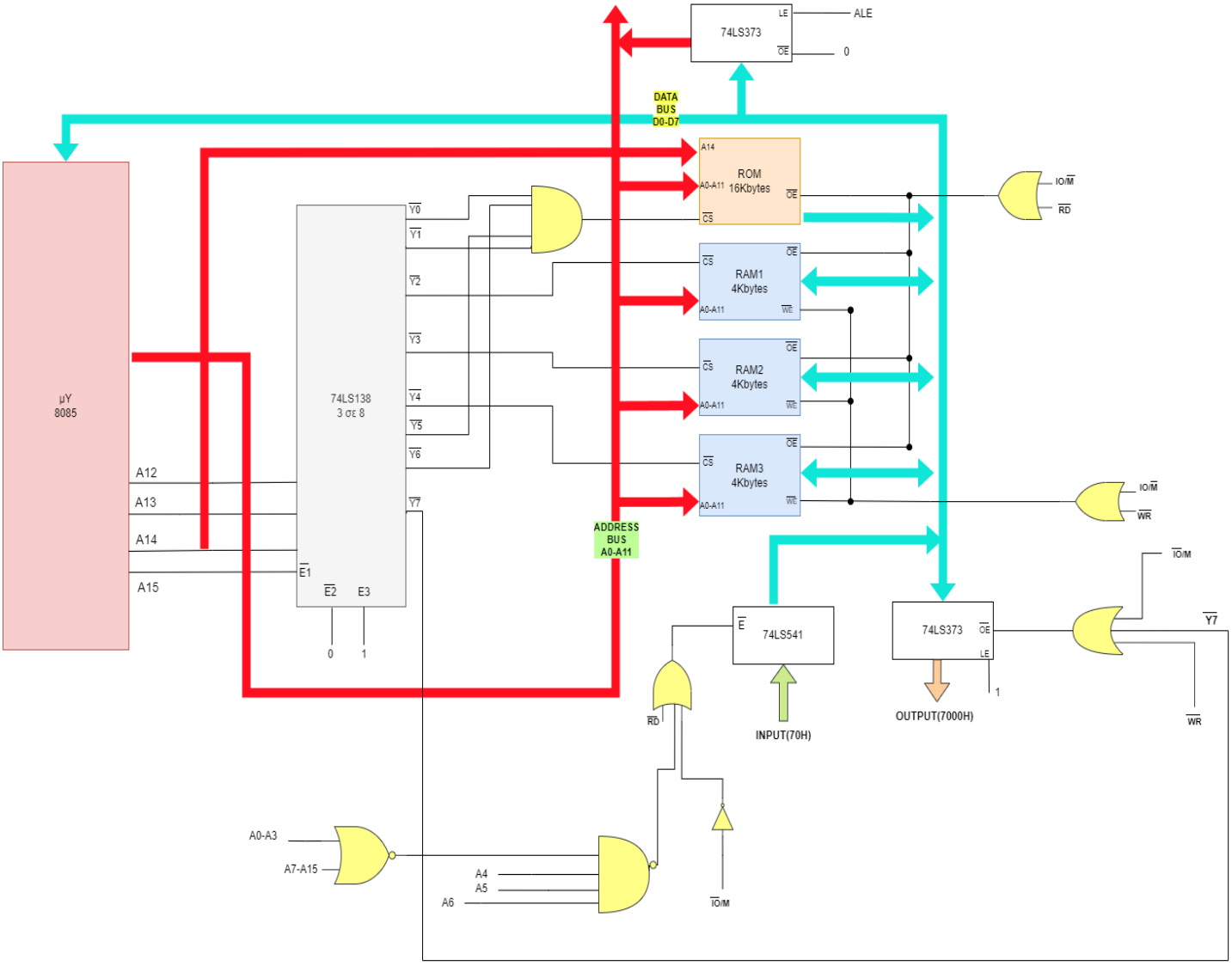
Άσκηση 7

Πρώτα παρουσιάζουμε τον χάρτη μνήμης

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Διεύθυνση	Memory
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	ROM 1 8Kbytes
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000	RAM 1 4Kbytes
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFF	
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3000	RAM 2 4Kbytes
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000	RAM 3 4KBytes
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFF	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000	ROM 2 8Kbytes
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	6FFF	
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	7000	MEMORY I/O

$CS_{ROM} = Y_0 + Y_1 + Y_5 + Y_6$

$CS_{RAM1} = Y_2$, $CS_{RAM2} = Y_3$, $CS_{RAM3} = Y_4$


74LS373
LE
ALE
OE
0
DATA BUS D0-D7
μY 8085
74LS138 3 σε 8
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
A12
A13
A14
A15
E1
E2
E3
0
1
ROM 16Kbytes
A14
A0-A11
CS
OE
RAM1 4Kbytes
RAM2 4Kbytes
RAM3 4Kbytes
WE
IO/M
RD
WR
ADDRESS BUS A0-A11
74LS541
74LS373
INPUT(70H)
OUTPUT(7000H)
A0-A3
A7-A15
A4
A5
A6