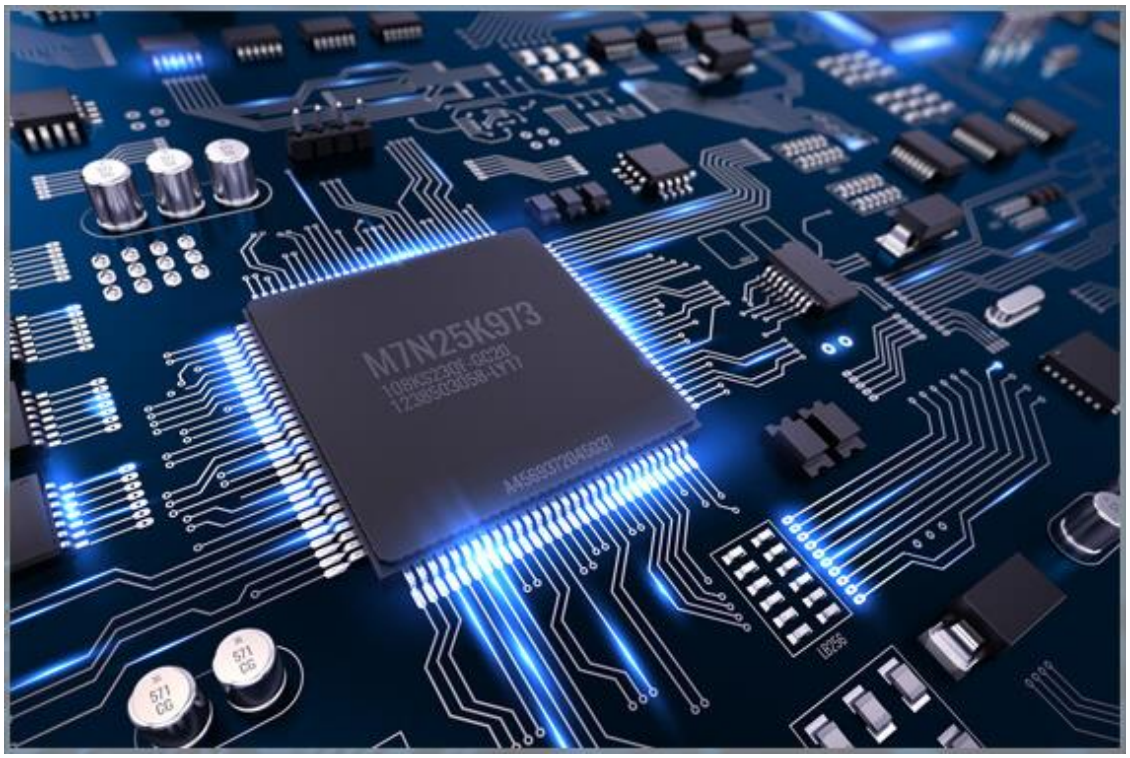
ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

2Η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ



MAY 4, 2021

ΘΟΔΩΡΗΣ ΑΡΑΠΗΣ – EL18028
ΚΡΙΣ ΚΟΥΤΣΗ – EL18905

Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2020-2021

ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

Άσκηση 1

(α)

Το ζητούμενο πρόγραμμα είναι το παρακάτω:

```
;(a)
     START:
          IN 10H
          MVI A,00H     ;A register will hold the value that we want to store
          LXI H,0900H   ;HL registers will hold the address
          MOV M,A
     STORE:
          INR A         ;We increment A and M (HL registers)
          INX H         ;and store the value of A to the address
          MOV M,A       ;stored in M
          CPI FFH       ;If A<255 repeat
          JNZ STORE

          END
```

Προκειμένου να ελέγξουμε αν λειτουργεί το πρόγραμμα όπως θέλουμε, τοποθετούμε τα δεδομένα στην RAM και έχουμε:

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	80	0981	81	0982	82	0983	83	0984	84	0985	85
0986	86	0987	87	0988	88	0989	89	098A	8A	098B	8B	098C	8C	098D	8D	098E	8E	098F	8F	0990	90	0991	91	0992	92	0993	93	0994	94
0995	95	0996	96	0997	97	0998	98	0999	99	099A	9A	099B	9B	099C	9C	099D	9D	099E	9E	099F	9F	09A0	A0	09A1	A1	09A2	A2	09A3	A3
09A4	A4	09A5	A5	09A6	A6	09A7	A7	09A8	A8	09A9	A9	09AA	AA	09AB	AB	09AC	AC	09AD	AD	09AE	AE	09AF	AF	09B0	B0	09B1	B1	09B2	B2
09B3	B3	09B4	B4	09B5	B5	09B6	B6	09B7	B7	09B8	B8	09B9	B9	09BA	BA	09BB	BB	09BC	BC	09BD	BD	09BE	BE	09BF	BF	09C0	C0	09C1	C1
09C2	C2	09C3	C3	09C4	C4	09C5	C5	09C6	C6	09C7	C7	09C8	C8	09C9	C9	09CA	CA	09CB	CB	09CC	CC	09CD	CD	09CE	CE	09CF	CF	09D0	D0
09D1	D1	09D2	D2	09D3	D3	09D4	D4	09D5	D5	09D6	D6	09D7	D7	09D8	D8	09D9	D9	09DA	DA	09DB	DB	09DC	DC	09DD	DD	09DE	DE	09DF	DF
09E0	E0	09E1	E1	09E2	E2	09E3	E3	09E4	E4	09E5	E5	09E6	E6	09E7	E7	09E8	E8	09E9	E9	09EA	EA	09EB	EB	09EC	EC	09ED	ED	09EE	EE
09EF	EF	09F0	F0	09F1	F1	09F2	F2	09F3	F3	09F4	F4	09F5	F5	09F6	F6	09F7	F7	09F8	F8	09F9	F9	09FA	FA	09FB	FB	09FC	FC	09FD	FD
09FE	FE	09FF	FF	0A00	00	0A01	00	0A02	00	0A03	00	0A04	00	0A05	00	0A06	00	0A07	00	0A08	00	0A09	00	0A0A	00	0A0B	00	0A0C	00

Όπως βλέπουμε παραπάνω, πράγματι οι τιμές αποθηκεύονται με ακριβώς με τον τρόπο που ζητείται.

(β)

Το πρόγραμμά μας είναι (ως συνέχεια του προηγούμενου προγράμματος):

```
;(b)
          LXI B,0000H  ;Initialisation of BC
     LOAD:
          MOV A,M      ;M=09FFH from before
          MVI D,09H    ;We set D=9 because we want to check each digit and also the
ALL_DIGITS:           ;the algorithm starts by decreasing D so we "lose" one iteration
          DCR D
          JZ NEXT      ;If D=0 load next number
          RRC
          JNC ALL_DIGITS  ;If CY=1 then count else repeat
COUNT_ONES:
          INX B
          JMP ALL_DIGITS
     NEXT:
          DCR L        ;Decrease L until it reaches zero
          JNZ LOAD     ;If L!=0 Load next else finish

          END
```

Γνωρίζουμε ότι στο εύρος αριθμών 0-255 υπάρχουν 2048 ψηφία, από τα οποία τα μισά (1024) είναι μηδέν και τα άλλα μισά άσσοι. Συνεπώς, προκειμένου να ελέγξουμε την ορθότητα του προγράμματός μας, τρέχουμε το πρόγραμμα δύο φορές, τοποθετώντας πριν το τέλος αρχικά τον κώδικα που φαίνεται στην πρώτη φωτογραφία και ύστερα τον κώδικα που φαίνεται στην δεύτερη. Με αυτόν τον τρόπο μπορούμε να δούμε τον αριθμό που περιέχεται στον διπλό καταχωρητή (BC). Σύμφωνα με την ένδειξη των LED (αντίστροφη λογική), ισχύει: $(BC) = (0400)_{HEX} = (1024)_{DEC}$. Άρα, το πρόγραμμά μας λειτουργεί με τον επιθυμητό τρόπο.

```
NEXT:
     DCR L          ;Decrease L until it reaches zero
     JNZ LOAD       ;If L!=0 Load next else finish
     MOV A,B
     STA 3000H

     END
```

```
NEXT:
     DCR L          ;Decrease L until it reaches zero
     JNZ LOAD       ;If L!=0 Load next else finish
     MOV A,C
     STA 3000H

     END
```

(γ)

Γράφουμε το πρόγραμμα με την λογική ότι τα νούμερα (0-255) μπορεί να έχουν αποθηκευτεί σε τυχαίες θέσεις στο κομμάτι μνήμης 0900H – 09FFH επιπλέον υποθέτουμε ακόμα ότι κάποιο νούμερο μπορεί να υπάρχει περισσότερες από μία φορές ή να μην υπάρχει καθόλου. Χωρίς αυτές τις υποθέσεις, το η απάντηση θα ήταν ξεκάθαρη (μια απλή αφαίρεση). Η απάντηση που θα θέλαμε στο συγκεκριμένο πρόβλημα είναι:

$(60)_{HEX} – (10)_{HEX} + 1 = (51)_{HEX}$ ή $(96)_{DEC} – (16)_{DEC} + 1 = (81)_{DEC}$

Ελέγχουμε την ορθότητα του προγράμματός μας εκτυπώνοντας την τιμή του καταχωρητή D, εφαρμόζοντας την αλλαγή που φαίνεται κάτω από το πρόγραμμα. Η ένδειξη των LED συμφωνεί με τα παραπάνω αποτελέσματα, άρα το πρόγραμμα λειτουργεί σωστά.

```
;(c)
          MVI E,FFH
          MVI D,00H     ;Initialise D
          MOV A,M       ;Load first Number (M=0900H from before)
STATEMENT:
          CPI 10H       ;If A<10H DONT_COUNT else continue
          JC DONT_COUNT
          CPI 61H       ;If A>60H DONT_COUNT else continue
          JNC DONT_COUNT
          INR D         ;Count
DONT_COUNT:
          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 END
          CPI 61H
          JNC END
          INR D
     END:
          END
```

Στην ταμπέλα END κάνουμε την ακόλουθη αλλαγή προκειμένου να ελέγξουμε την ορθότητα του προγράμματος:

```
     END:
          MOV A,D
          STA 3000H
          END
```

Το πρόγραμμα βρίσκεται στο αρχείο *Άσκηση_1.8085*.

Άσκηση 2

Το πρόγραμμά μας είναι το ακόλουθο:

```
     LXI B,0064H    ;We call delb with 64Hms = 1/10 sec delay
START:
     LDA 2000H      ;Load input from dip switches to A
     RLC            ;Rotate left to check MSB
     JNC OFF        ;If MSB is off then go to OFF
     JMP START
OFF:                ;We check if the switch is turned on
     LDA 2000H
     RLC
     JC ON1         ;If it's on then go to ON1
     JMP OFF        ;Else wait until it's on
ON1:                ;In order to light up the LEDs we have to switch off
                    ;the MSB, but when that happens we'll have to call delb
     MVI D,C8H      ;200 times(we want them on for 20s), that's why D = 200
     LDA 2000H
     RLC
     JNC OPEN       ;If MSB turned off then push button(off-on-off) is activated
     JMP ON1
OPEN:
     LDA 2000H      ;If MSB turns on the 20s timer keeps on going, but if that
     RLC            ;happens then we have to check whether it turns off or not.
     JC ON_AGAIN    ;If it does so, then we have to reset the timer
     MVI A,00H
     STA 3000H      ;Turn on all LEDs
     CALL DELB
     DCR D          ;Decrease D
     JNZ OPEN       ;If D = 0 then 20s passed
     MVI A,FFH
     STA 3000H      ;Turn off all LEDs and start checking again
     JMP OFF
ON_AGAIN:           ;Getting here means that if the MSB switch turns off then the
     LDA 2000H      ;timer has to reset.If the MSB stays on the whole time then we
     RLC            ;repeat the same process as above
     JNC RESTART
     MVI A,00H
     STA 3000H
     CALL DELB
     DCR D
     JNZ ON_AGAIN
     MVI A,FFH
     STA 3000H
     JMP OFF
RESTART:            ;If on-off occured while we had the LEDs on then reset the timer
     MVI D,C8H
     JMP OPEN

     END
```

Το πρόγραμμα βρίσκεται στο αρχείο *Άσκηση_2.8085*

Άσκηση 3

Τα προγράμματά μας για κάθε ερώτημα δίνονται παρακάτω:

(i)

```
START:
          MVI D,08H      ;D = 8
          LDA 2000H      ;Load input from dip switches to A
          MVI B,00H      ;B = 0
CHECK:                   ;Starting from LSB to MSB we find the first
                         ;dip switch that's on
          RRC
          DCR D          ;Decrease D
          JZ TURNOFF     ;If D = 0 then no dip switch was on so turn off
                         ;all LEDs and start again
          INR B          ;Increase B (B is equal to the current position that
                         ;we're checking)
          JNC CHECK      ;If a dip switch is on then stop looping
          MVI A,FEH
          DCR B
TURN_ON:
          RLC            ;Rotate left until we reach the correct position
          DCR B
          JNZ TURN_ON
          STA 3000H      ;Turn on the LED
          JMP START      ;Start checking again
TURNOFF:                 ;Getting here means that no dip switch was on
          MVI A,FFH
          STA 3000H      ;Turn off all LEDs
          JMP START

          END
```

Το πρόγραμμα βρίσκεται στο αρχείο *Άσκηση_3i.8085*

(ii)

```
START:
     CALL KIND
     CPI 00H        ;If we press 0 then go to OFF
     JZ OFF
     CPI 09H        ;If we press 9 then go to OFF
     JNC OFF
     MOV B,A        ;Save A to register B
     MVI A,00H      ;A = 0
     DCR B          ;Decrease B
     JZ OPEN        ;If B = 0 then open all LEDs (We pressed  1)
     INR A          ;Increase A
REPEAT:
     DCR B
     JZ OPEN        ;If B = 0 then go to OPEN with the current data of A
     RLC            ;Rotate left
     INR A
     JMP REPEAT
OPEN:
     STA 3000H      ;Open the LEDs starting from the number that we pressed
                    ;up to the MSB
     JMP START      ;Start checking again
OFF:                ;If we pressed 0 or 9 then turn off all LEDs and go to START
     MVI A,FFH
     STA 3000H
     JMP START

     END
```

Το πρόγραμμα βρίσκεται στο αρχείο *Άσκηση_3ii.8085*

(iii)

```
START:
       IN 10H          ; άρση προστασίας μνήμης
       LXI H,0A00H     ; 0A00H = η αρχή του μπλοκ αποθήκευσης
       MVI B,04H       ; απλός επαναλήπτης
L1:
       MVI M,10H       ; αποθήκευσε "κενο" (4 φορές)
       INX H
       DCR B
       JNZ L1
LINE0:
       MVI A,FEH       ; πόρτα σάρωσης = 11111110 - επιλογή γραμμής
       STA 2800H
       LDA 1800H       ; διάβασε τις στήλες των πλήκτρων
       ANI 07H         ; κρατάμε μόνο τα 3 LSB (περιέχουν την πληροφορία)
       MVI C,86H       ; C = πιθανός κωδικός
       CPI 06H         ; A ?= 00000110 (δηλ. πατήθηκε το κουμπί της
                       ; 1ης στήλης [INSTR_STEP])
       JZ SHOW         ; αν ναι, προώθησέ τον κωδικό του
                       ; στην έξοδο των 7-segment display
       MVI C,85H       ; ομοίως για όλα τα πιθανά κουμπιά
       CPI 05H         ; A ?= 00000101 (δηλ. πατήθηκε το κουμπί της
                       ; 2ης στήλης [FETCH PC])
       JZ SHOW
                       ; αγνοούμε το κουμπί HDWR_STEP
LINE1:
       MVI A,FDH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,84H
       CPI 06H         ; RUN
       JZ SHOW
       MVI C,80H
       CPI 05H         ; FETCH_REG
       JZ SHOW
       MVI C,82H
       CPI 03H         ; FETCH_ADDRS
       JZ SHOW
LINE2:
       MVI A,FBH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,00H
       CPI 06H         ; 0
       JZ SHOW
       MVI C,83H
```

```
        CPI 05H         ; STORE/INCR
        JZ SHOW
        MVI C,81H
        CPI 03H         ; DECR
        JZ SHOW
LINE3:
        MVI A,F7H
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,01H       ; 1
        CPI 06H
        JZ SHOW
        MVI C,02H       ; 2
        CPI 05H
        JZ SHOW
        MVI C,03H       ; 3
        CPI 03H
        JZ SHOW
LINE4:
        MVI A,EFH
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,04H
        CPI 06H         ; 4
        JZ SHOW
        MVI C,05H
        CPI 05H         ; 5
        JZ SHOW
        MVI C,06H
        CPI 03H         ; 6
        JZ SHOW
LINE5:
        MVI A,DFH
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,07H
        CPI 06H         ; 7
        JZ SHOW
        MVI C,08H
        CPI 05H         ; 8
        JZ SHOW
        MVI C,09H
        CPI 03H         ; 9
```

```
LINE6:
       MVI A,BFH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,0AH
       CPI 06H       ; A
       JZ SHOW
       MVI C,0BH
       CPI 05H       ; B
       JZ SHOW
       MVI C,0CH
       CPI 03H       ; C
       JZ SHOW
LINE7:
       MVI A,7FH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,0DH
       CPI 06H       ; D
       JZ SHOW
       MVI C,0EH
       CPI 05H       ; E
       JZ SHOW
       MVI C,0FH
       CPI 03H       ; F
       JZ SHOW
       JMP START     ; αν δεν πατήθηκε κουμπί, επανάλαβε τους ελέγχους
SHOW:
       LXI H,0A04H          ; ετοιμάζουμε τη θέση 0A04H
       MOV A,C              ; κωδικός --> A
       ANI 0FH              ; κρατάμε τα 4 LSBs
       MOV M,A              ; τα βάζουμε στη θέση 0A04H
                            ; δηλ. στο πέμπτο ψηφίο των 7-segment display
       INX H                ; επόμενη θέση μνήμης
       MOV A,C
       ANI F0H              ; κρατάμε τα 4 MSBs
       RLC
       RLC                  ; τα κάνουμε LSBs
       RLC
       RLC
       MOV M,A              ; τα αποθηκεύουμε στο έκτο (αριτερότερο)
                            ; ψηφίο των 7-segment display
       LXI D,0A00H          ; μετακίνηση του block 0A00H - 0A05H
                            ; στο σημείο που διαβάζει η DCD
       CALL STDM
       CALL DCD            ; απεικόνιση
       JMP START           ; επανάληψη

       END
```

Το πρόγραμμα βρίσκεται στο αρχείο *Άσκηση_3iii.8085*

Άσκηση 4

Το πρόγραμμά μας είναι το ακόλουθο:

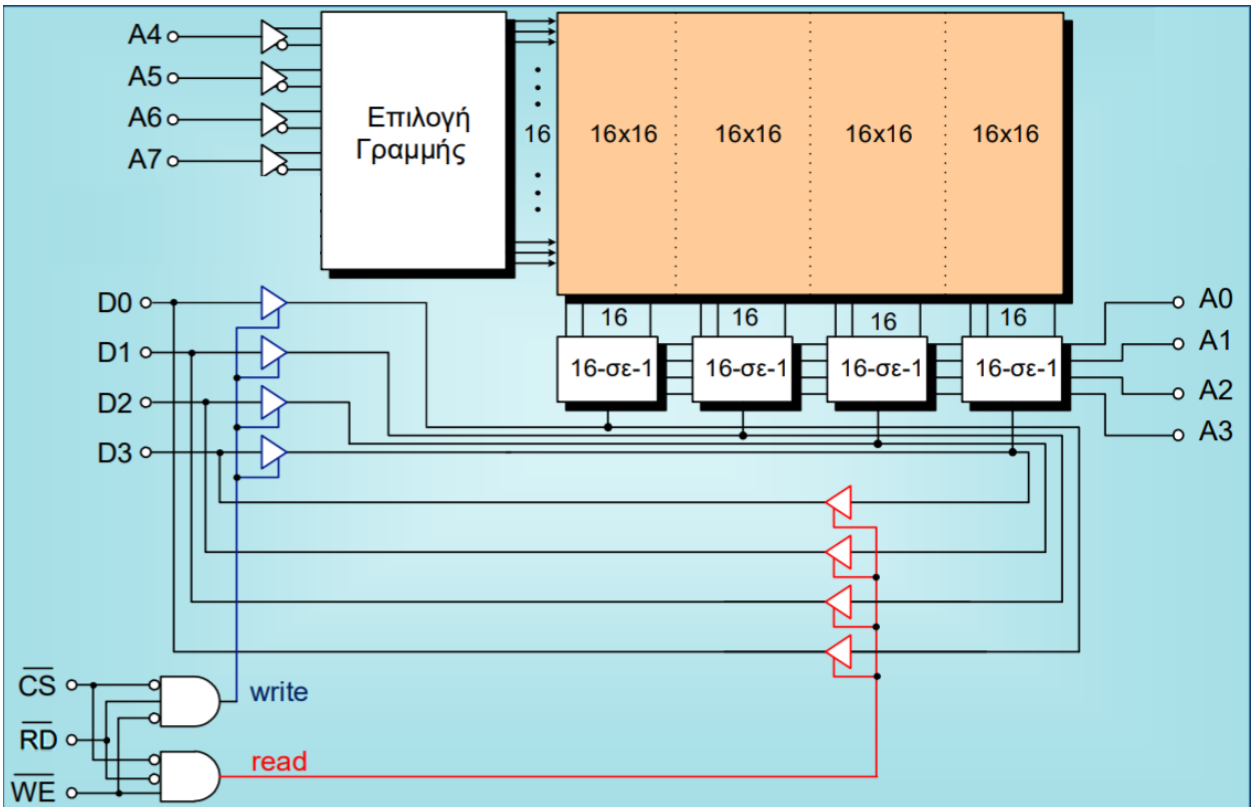
```
START:
     LDA 2000H      ;Load input from dip switches to A
     MOV B,A        ;Save A to register B
A0_B0:
     ANI 01H        ;A = A AND 00000001
     MOV C,A        ;C = A
     MOV A,B        ;A is equal to the input of dip switches
     ANI 02H        ;A = A AND 00000010
XOR0:
     RRC            ;Rotate right to get the output at LSB
     XRA C          ;A XOR C
     MOV D,A        ;Save the answer
A1_B1:
     MOV A,B
     ANI 04H        ;A = A AND 00000100
     MOV C,A        ;C = A
     MOV A,B
     ANI 08H        ;A = A AND 00001000
XOR1:
     RRC
     XRA C          ;A XOR C
     RRC            ;Rotate right to get X1 at 2nd LSB
     MOV E,A        ;E = A XOR C (output of XOR1)
     RRC            ;Rotate right to get X0 at LSB
     XRA D          ;(A XOR C) XOR D
     ORA E          ;Save X1 at 2nd LSB
     MOV D,A        ;Save X0 at LSB
A2_B2:
     MOV A,B
     ANI 10H        ;A = A AND 00010000
     MOV C,A        ;C = A
     MOV A,B
     ANI 20H        ;A = A AND 00100000
     RRC
AND:
     ANA C          ;A AND C
     MOV E,A        ;E = A AND C (output of A2_B2)
A3_B3:
     MOV A,B
     ANI 40H        ;A = A AND 01000000
     MOV C,A        ;C = A
     MOV A,B
     ANI 80H        ;A = A AND 10000000
     RRC
AND1:
     ANA C          ;A AND C (output of A3_B3)
     RRC
     RRC
     MOV B,A
     RRC
     ORA D
     MOV D,A        ;Save X3 at 4th LSB
```

```
OR:
     MOV A,B
     ORA E          ;(A AND C(output of A3_B3)) OR E
     RRC
     RRC
     ORA D          ;Save X2 at 3rd LSB
LED:
     CMA            ;Inverse logic
     STA 3000H      ;Turn on the correct LEDs
     JMP START      ;Start checking again

     END
```

Το πρόγραμμα βρίσκεται στο αρχείο *Άσκηση_4.8085*

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 μνήμες 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 $9^{η}$ (1001) γραμμή και η $6^{η}$ (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.

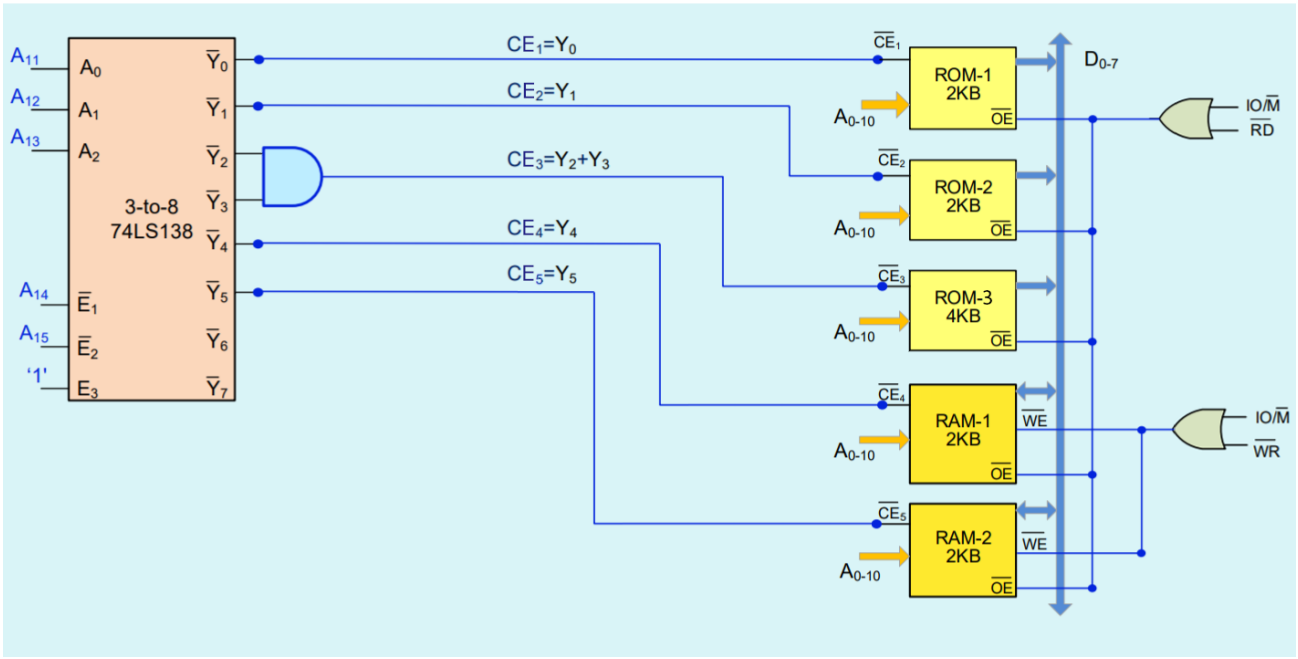
Άσκηση 6

Ο χάρτης μνήμης του συστήματος μνήμης είναι:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Memory
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	ROM1-2K
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FF	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0800	ROM2-2K
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF	
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000	ROM3-4K
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	RAM1-2K
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	27FF	
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	2800	RAM2-2K
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFF	

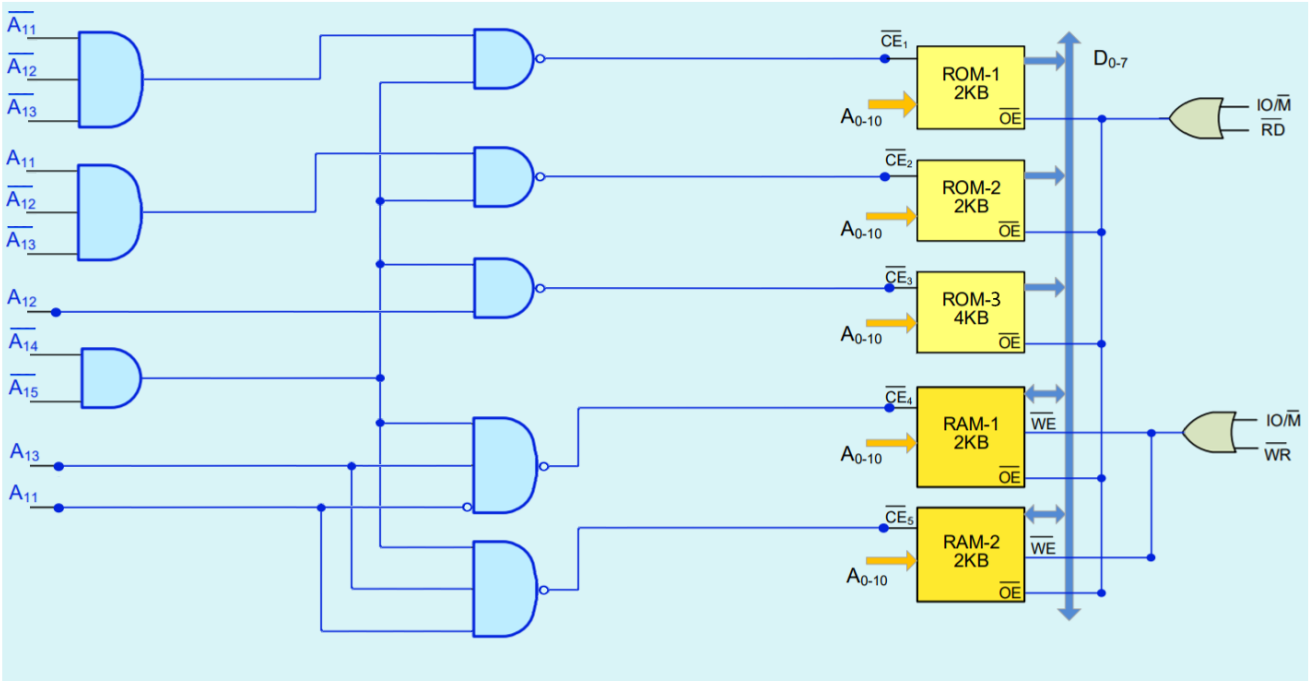
(α)

Το σύστημα μνήμης υλοποιημένο με αποκωδικοποιητή και λογικές πύλες:



(β)

Το σύστημα μνήμης υλοποιημένο μόνο με λογικές πύλες:



Άσκηση 7

Ο χάρτης μνήμης είναι:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Memory
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	ROM1-12K
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	RAM1-4K
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	RAM2-4K
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	RAM3-4K
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF	
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000	ROM1-4K
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	6FFF	

Το σύστημα μνήμης είναι: