Συστήματα Μικροϋπολογιστών
6ο Εξάμηνο
3η Ομάδα Ασκήσεων

Χαρδούβελης Γεώργιος-Ορέστης
el15100
6ο Εξάμηνο

Κυριάκου Αθηνά
el17405
6ο Εξάμηνο

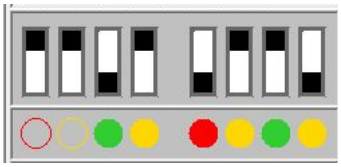
Άσκηση 1:

(i) Στο υποερώτημα αυτό δημιουργήθηκε πρόγραμμα σε Assembly 8085 που λαμβάνει είσοδο από τα dip switches (διεύθυνση μνήμης 2000H), βρίσκει το πρώτο της μηδενικό (με ανάγνωση από τα MSB προς τα LSB) και στην έξοδο (διεύθυνση μνήμης 3000H) ανάβει το LED της θέσης όπου βρέθηκε το πρώτο μηδενικό και όλα τα LED που αντιστοιχούν στα LSB του. Το πρόγραμμα είναι συνεχούς λειτουργίας (LOOP_EXT) και ο κώδικάς του (αρχείο ex1_1.8085) φαίνεται παρακάτω:

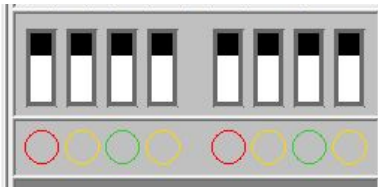
```
LOOP_EXT:
       IN 10H

       MVI B,FFH     ;initialize register B to 1111 1111 (result)
       MVI C,80H     ;initialize register C to 1000 0000 (mask)
       LDA 2000H     ;input from dip switches in register A


       L1:

              MOV D,A       ;(D)<-(A)
              ANA C         ;(A)<-(A) AND (C)
              JZ FINAL      ;if (A)=0, zero is found

              ;if first zero is not yet found
              MOV A,B
              STC           ;CY=1
              CMC           ;CY=0
              RAR           ;CY=1
              MOV B,A       ;LATEST RESULT

              MOV A,C
              CMC           ;CY=0
              RAR           ;CY=0
              CPI 00H       ;compare new mask with zero
              JZ FINAL      ;if it is zero, we have checked all the bits
              MOV C,A       ;LATEST MASK

              MOV A,D
              JP L1

       ;to the final loop if a zero is found or we have checked all the bits

       FINAL:

              MOV A,B
              CMA
              STA 3000H     ;output in LEDs
              JP LOOP_EXT
END
```

Ενδεικτικές έξοδοι από την προσομοίωση είναι:

Για είσοδο: 1101 0110



Για είσοδο: 1111 1111 (κανένα μηδενικό)



(ii) Στο αρχείο ex1_2.8085, δημιουργήθηκε πρόγραμμα συνεχούς λειτουργίας που αναμένει το πάτημα κουμπιού του δεκαεξαδικού πληκτρολογίου και μόνο των πλήκτρων 0-8 μέσω της ρουτίνας KIND. Στην περίπτωση που έχει πατηθεί ένας από τους αριθμούς 0-7, ανάβει το αντίστοιχο LED ενώ αν έχει πατηθεί ο αριθμός 8, αναβοσβήνουν όλα τα LED. Ο κώδικας του προγράμματος είναι:

```
LOOP_EXT:

        IN 10H
        MVI C,00H

        CHECK:
                CALL KIND           ;check if 0-8 is pressed, in register A

                MVI B,00H           ;B is the temp register
                CMP C               ;compare input with 0 button
                JZ FINAL

                MVI B,01H
                INR C
                CMP C               ;compare input with 1
                JZ FINAL

                MVI B,02H
                INR C
                CMP C               ;compare input with 2
                JZ FINAL
```

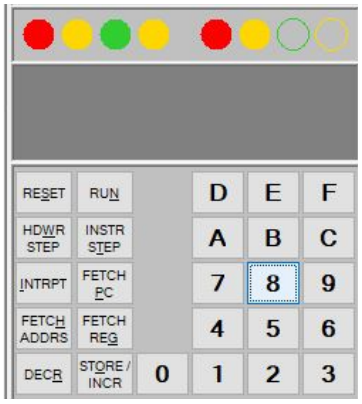
```
        MVI B,04H
        INR C
        CMP C           ;compare input with 3
        JZ FINAL

        MVI B,08H
        INR C
        CMP C           ;compare input with 4
        JZ FINAL

        MVI B,10H
        INR C
        CMP C           ;compare input with 5
        JZ FINAL

        MVI B,20H
        INR C
        CMP C           ;compare input with 6
        JZ FINAL

        MVI B,40H
        INR C
        CMP C           ;compare input with 7
        JZ FINAL

        INR C
        CMP C           ;compare input with 8
        JZ COUNTER

        JMP LOOP_EXT    ;if none of them is pressed, call KIND again


COUNTER: MVI C,0AH      ;counter to repeat the flash routine for the
                        ;LED's blinking 10 times

FLASH:                  ;LED's blinking
        MVI A,00H
        STA 3000H
        NOP
        NOP
        NOP
        MVI A,FFH
        STA 3000H
        NOP
        NOP
        NOP
```

```
        DCR C
        JNZ FLASH
        JMP LOOP_EXT

    FINAL:
        MOV A,B
        CMA
        STA 3000H
        JMP LOOP_EXT

END
```

Ενδεικτικά αποτελέσματα από την προσομοίωση:

Πάτημα του κουμπιού 5:

Στιγμιότυπο από το πάτημα του κουμπιού 8:



(iii) Σε αυτό το υποερώτημα ζητείται να υλοποιηθεί απευθείας ανάγνωση του πληκτρολογίου χωρίς τη χρήση KIND. Αυτό επιτυγχάνεται γνωρίζοντας ότι η σάρωση του πληκτρολογίου του προσομοιωτή γίνεται κατά μία γραμμή κάθε φορά και χρησιμοποιώντας την πόρτα σάρωσης για την επιλογή της επιθυμητής γραμμής και τον συσσωρευτή για την ανάγνωση των δεδομένων στήλης (προσδιορισμός στήλης) από την πόρτα ανάγνωσης.
Η χρήση της εξαψήφιας 7τμημάτων LED οθόνης γίνεται με χρήση της ρουτίνας DCD του προγράμματος monitor και της STDM.

Ο κώδικας του προγράμματος βρίσκεται στο αρχείο ex1_3.8085 και φαίνεται παρακάτω:

```
;go check each line if a button is pressed

IN 10H

INPUT: ;read from the keyboard

        MVI B,07H

        L1:
                MVI A,FEH     ;line of INSTR STEP/FETCH PC
                STA 2800H     ;load to scan port, selection of line
                LDA 1800H     ;selection of column

                ANA B         ;keep only the 3 LSB // CONTENT IN REG A

                CPI 07H       ;if no button is pressed from this line
                JZ L2         ;go check the next line

                ;if a button is pressed

                LXI H,09C0H   ;09C0H address to save the data to be displayed
                MVI M,06H
                INX H         ;addr 09C1
                MVI M,08H
                CPI 06H       ;INSTR STEP (86) // CONTENT IN REG A
                JZ OUTPUT     ;OUTPUT ONLY WHEN THIS BUTTON IS PRESSED

                LXI H,09C0H
                MVI M,05H
                INX H         ;addr 09C1
                MVI M,08H
                CPI 05H       ;FETCH PC (85)
                JZ OUTPUT


        L2:
                MVI A,FDH     ;line of RUN/FETCH REG/FETCH ADRS
                STA 2800H
                LDA 1800H     ;selection of column

                ANA B         ;keep only the 3 LSB // CONTENT IN REG A

                CPI 07H       ;if no button is pressed from this line
```

```
        JZ L3

        ;if a button is pressed

        LXI H,09C0H
        MVI M,04H
        INX H           ;addr 09C1
        MVI M,08H
        CPI 06H         ;RUN (84)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,00H
        INX H           ;addr 09C1
        MVI M,08H
        CPI 05H         ;FETCH REG (80)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,02H
        INX H           ;addr 09C1
        MVI M,08H
        CPI 03H         ;FETCH ADRS (82)
        JZ OUTPUT


L3:
        MVI A,FBH       ;line of 0/STORE INCR/INCR
        STA 2800H
        LDA 1800H       ;selection of column

        ANA B           ;keep only the 3 LSB

        CPI 07H         ;if no button is pressed from this line
        JZ L4

        ;if a button is pressed

        LXI H,09C0H
        MVI M,00H
        INX H           ;addr 09C1
        MVI M,00H
        CPI 06H         ;0 (00)
        JZ OUTPUT

        LXI H,09C0H
```

```
        MVI M,03H
        INX H           ;addr 09C1
        MVI M,08H
        CPI 05H         ;STORE/INCR (83)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,01H
        INX H           ;addr 09C1
        MVI M,08H
        CPI 03H         ;DCR (81)
        JZ OUTPUT


L4:
        MVI A,F7H       ;line of 1/2/3
        STA 2800H
        LDA 1800H       ;selection of column

        ANA B           ;keep only the 3 LSB

        CPI 07H         ;if no button is pressed from this line
        JZ L5

        ;if a button is pressed

        LXI H,09C0H
        MVI M,01H
        INX H           ;addr 09C1
        MVI M,00H
        CPI 06H         ;1 (01)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,02H
        INX H           ;addr 09C1
        MVI M,00H
        CPI 05H         ;2 (02)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,03H
        INX H           ;addr 09C1
        MVI M,00H
        CPI 03H         ;3 (03)
        JZ OUTPUT
```

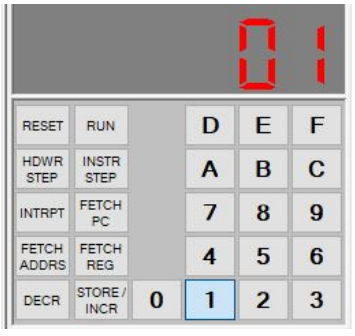
```
L5:
        MVI A,EFH     ;line of 4/5/6
        STA 2800H
        LDA 1800H     ;selection of column

        ANA B         ;keep only the 3 LSB

        CPI 07H       ;if no button is pressed from this line
        JZ L6

        ;if a button is pressed

        LXI H,09C0H
        MVI M,04H
        INX H         ;addr 09C1
        MVI M,00H
        CPI 06H       ;4 (04)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,05H
        INX H         ;addr 09C1
        MVI M,00H
        CPI 05H       ;5 (05)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,06H
        INX H         ;addr 09C1
        MVI M,00H
        CPI 03H       ;6 (06)
        JZ OUTPUT


L6:
        MVI A,DFH     ;line of 7/8/9
        STA 2800H
        LDA 1800H     ;selection of column

        ANA B         ;keep only the 3 LSB

        CPI 07H       ;if no button is pressed from this line
        JZ L7
```

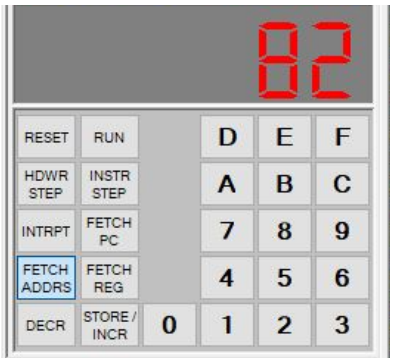
```
        ;if a button is pressed

        LXI H,09C0H
        MVI M,07H
        INX H           ;addr 09C1
        MVI M,00H
        CPI 06H         ;7 (07)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,08H
        INX H           ;addr 09C1
        MVI M,00H
        CPI 05H         ;8 (08)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,09H
        INX H           ;addr 09C1
        MVI M,00H
        CPI 03H         ;9 (09)
        JZ OUTPUT


L7:
        MVI A,BFH       ;line of A/B/C
        STA 2800H
        LDA 1800H       ;selection of column

        ANA B           ;keep only the 3 LSB

        CPI 07H         ;if no button is pressed from this line
        JZ L8

        ;if a button is pressed

        LXI H,09C0H
        MVI M,0AH
        INX H           ;addr 09C1
        MVI M,00H
        CPI 06H         ;A (0A)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,0BH
        INX H           ;addr 09C1
```

```
        MVI M,00H
        CPI 05H         ;B (0B)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,0CH
        INX H           ;addr 09C1
        MVI M,00H
        CPI 03H         ;C (0C)
        JZ OUTPUT


L8:
        MVI A,7FH       ;line of D/E/F
        STA 2800H
        LDA 1800H       ;selection of column

        ANA B           ;keep only the 3 LSB

        CPI 07H         ;if no button is pressed from this line
        JZ INPUT        ;if all checked loopINPUT

        ;if a button is pressed

        LXI H,09C0H
        MVI M,0DH
        INX H           ;addr 09C1
        MVI M,00H
        CPI 06H         ;D (0D)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,0EH
        INX H           ;addr 09C1
        MVI M,00H
        CPI 05H         ;E (0E)
        JZ OUTPUT

        LXI H,09C0H
        MVI M,0FH
        INX H           ;addr 09C1
        MVI M,00H
        CPI 03H         ;F (0F)
        JZ OUTPUT
```

```
OUTPUT:
        ;make the first 4 digits of the 7segment display blank

        MVI B,FFH       ;counter for the DCD loop

        LXI H,09C2H
        MVI M,10H       ;code of blank for DCD is 10H
        INX H           ;addr 09C3
        MVI M,10H
        INX H           ;addr 09C4
        MVI M,10H
        INX H           ;addr 09C5
        MVI M,10H

        LXI D,09C0H
        CALL STDM
        DCD_LOOP:
                CALL DCD
                DCR B
                JNZ DCD_LOOP
                JMP INPUT     ;input displayed, go read again
END
```

Ενδεικτικά αποτελέσματα προσομοίωσης:





Άσκηση 2

Παρακάτω έχουμε ο σύστημα που υπολοιεί μεταφορά δεδομένων από ένα μΥ-Σ 8085 σε ένα άλλο επίσης μΥ-Σ 8085 που η “χειραψία” (handsake) να βασίζεται στη χρήση των γραμμών σειριακής Ε/Ε.
Με βάση τον κώδικα αυτόν επιτρέπεται η μεταφορά 256 δεδομένων που βρίσκονται στη μνήμη του μΥ-Σ 1 με αρχή τη θέση που δείχνει ο καταχωρητής H-L. Ύστερα αποθηκεύονται στη μνήμη του μΥ-Σ 2 με βάση το δικό του καταχωρητή H-L. Η μικρότερη τιμή από τις 256 αποθηκεύεται στον καταχωρητή C του μΥ-Σ 2
Ακολουθούν τα προγράμματα assembly και για τους δύο καταχωρητές.

Για τον μΥΣ1:

```
IN 10H
MVI B,100H
LXI H,0A00H
SEND:
        MVI A,40H
        SIM
        OUT SOD1
SID_0:
        RIM
        ANA 80H
        CPI 80H
        JNZ SID_0
        MOV A,M
        INX H
        OUT DATA1

SID_1:
        RIM
        ANA 80H
        CPI 80H
        JZ SID_1
        DCR BB
```

```
        JZ STOP
        JMP SEND

STOP:
        END
```

Για τον μΥΣ2:

```
IN 10H
MVI C,00H
MVI B,100H
LXI H,0A00H

SID_0:
        RIM
        ANA 80H
        CPI 80H
        JNZ SID_0
        ANA 40H
        SIM
        OUT SOD2

SID_1:
        RIM
        ANA 80H
        CPI 80H
        JZ SID_1
        IN DATA2
        MON M,A
        INX H
        CMO C
        JNC DONT_WANNA
        MOV C,A

DONT_WANNA:
        ANA 40H
        SIM
        ANA 80H
        CPI 80H
        JZ DONT_WANNA
        DCR B
        JZ STOP
        JMP SIS_0

STOP:
        END
```

Άσκηση 3

Για να σχεδιάσουμε το σύστημα μνήμης που περιγράφεται χρειαζόμαστε συνολικά 20kB ROM και (άρα μια των 16kB και μία των 4kB) και 8kB RAM (άλλα 2 των 4kB).

Ο χάρτης μνήμης έχει ως εξής:

Μνήμες	Διευθύνσεις	A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0
ROM1	Αρχή BIN HEX	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
	Τέλος BIN HEX	0 0 1 0 2	1 1 1 1 F	1 1 1 1 F	1 1 1 1 F
RAM1	Αρχή BIN HEX	0 0 1 1 3	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
	Τέλος BIN HEX	0 0 1 1 3	1 1 1 1 F	1 1 1 1 F	1 1 1 1 F
RAM2	Αρχή BIN HEX	0 1 0 0 4	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
	Τέλος BIN HEX	0 1 0 0 4	1 1 1 1 F	1 1 1 1 F	1 1 1 1 F
ROM1-ROM2	Αρχή BIN HEX	0 1 0 1 5	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
	Τέλος BIN HEX	0 1 1 0 6	1 1 1 1 F	1 1 1 1 F	1 1 1 1 F

Συγκεκριμένα, εφόσον χρησιμοποιούμε 2 μνήμες ROM, 16kB και 4kB αλλά δεν είναι συνεχόμενες οι περιοχές στη μνήμη, οι ROM1 και ROM2 θα έχουν ως εξής:

Μνήμες	Διευθύνσεις	A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0
ROM1_A	Αρχή BIN HEX	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
	Τέλος BIN HEX	0 0 1 0 2	1 1 1 1 F	1 1 1 1 F	1 1 1 1 F
ROM1_B	Αρχή BIN HEX	0 1 0 1 5	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0

	Τέλος BIN HEX	0 1 0 1 5	1 1 1 1 F	1 1 1 1 F	1 1 1 1 F
ROM2	Αρχή BIN HEX	0 1 1 0 6	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0
	Τέλος BIN HEX	0 1 1 0 6	1 1 1 1 F	1 1 1 1 F	1 1 1 1 F

Το bit A15 θα χρησιμοποιηθεί ως σήμα (enable) για τον αποκωδικοποιητή ενώ τα bits A12-A14 για την επιλογή μεταξύ των ολοκληρωμένων.

Συγκεκριμένα, στον αποκωδικοποιητή θα έχουμε το εξής:

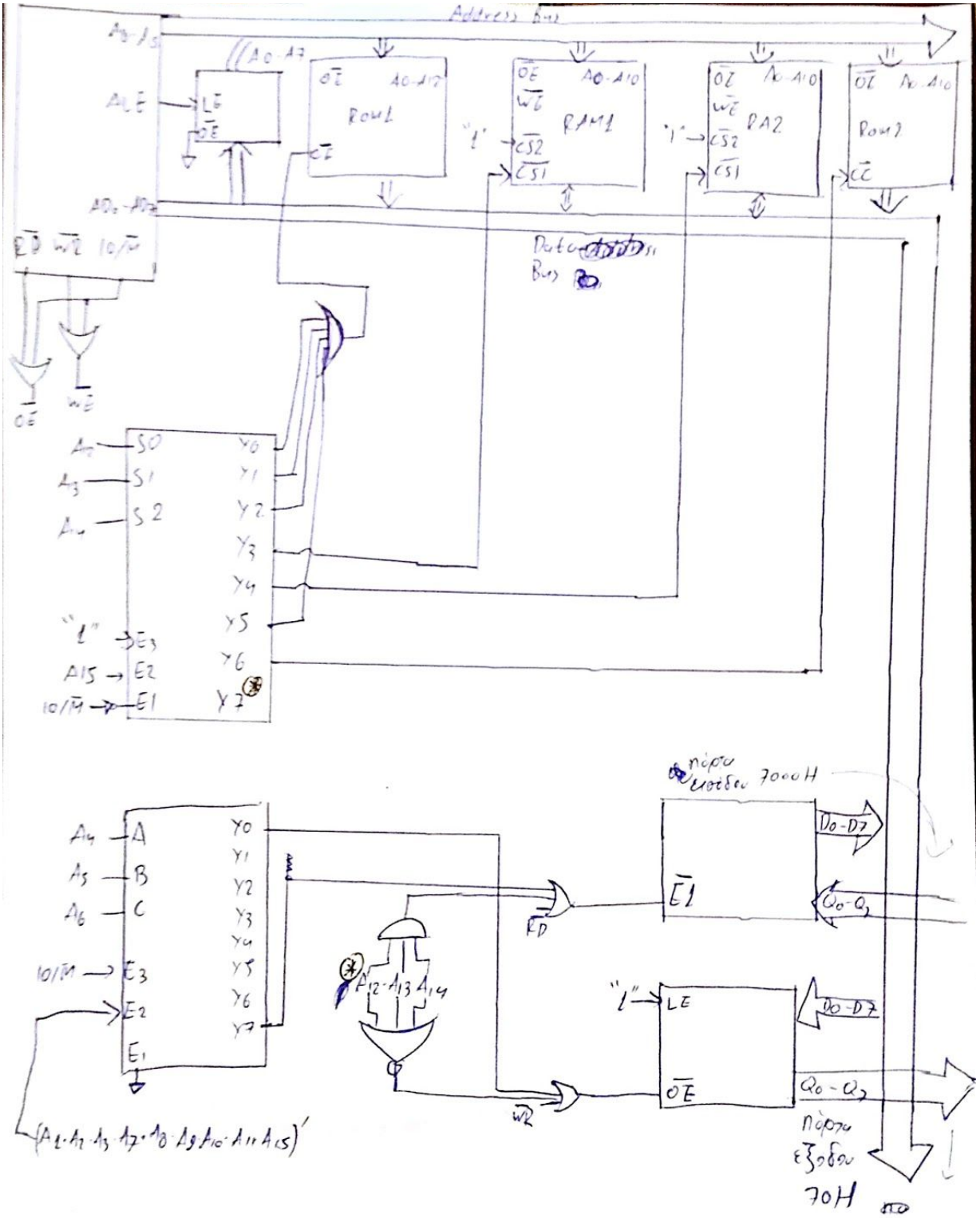
A14 A13 A12	Μνήμη
0 0 0	ROM1
0 0 1	
0 1 0	
0 1 1	RAM1
1 0 0	RAM2
1 0 1	ROM1
1 1 0	ROM2
1 1 1	-

Ακόμη, από την εκφώνηση έχουμε τα εξής για την E/E.

Διευθύνσεις	A_{15} A_{14} A_{13} A_{12}	A_{11} A_{10} A_9 A_8	A_7 A_6 A_5 A_4	A_3 A_2 A_1 A_0	
7000H	0 1 1 1	0 0 0 0	0 0 0 0	0 0 0 0	Memory Map I/O - θύρα εισόδου
70H	0 0 0 0	0 0 0 0	0 1 1 1	0 0 0 0	Standard I/O - θύρα εξόδου

Θα χρησιμοποιήσουμε λοιπόν τα A0, A1, A2, A3, A7, A8, A9, A10, A11, A15 ως είσοδο επίτρεψης, με μια NOR για να ελεγξουμε αν κάποιο από αυτά είναι 1, τα A4, A5, A6 ώστε να επιλεγεί η σωστή θύρα ενώ και τα A12, A13, A14 μπορούν να είναι καταλυτικά ώστε να επιλεγεί η σωστή θύρα στον αποκωδικοποιητή

Λαμβάνοντας όλα τα παραπάνω υπόψη, σχεδιάζουμε το παρακάτω σύστημα του μΥ συστήματος:



Άσκηση 4:

Οι μακροεντολές είναι ένα είδος υποπρογράμματος που χρησιμοποιείται για να μην επαναλαμβάνεται σε ένα πρόγραμμα η συγγραφή κώδικα συμβολικής γλώσσας που χρησιμοποιείται συχνά. Ωστόσο, στον αντικειμενικό κώδικα αυτό το τμήμα κώδικα περιλαμβάνεται όσες φορές καλείται η μακροεντολή από το κύριο πρόγραμμα.

α) Η μακροεντολή SWAP Nible Q θέλουμε όταν καλείται να εναλλάσσει το χαμηλότερης αξίας HEX ψηφίο του καταχωρητή-ορίσματος Q με το υψηλότερης αξίας ψηφίο του. Επίσης να πραγματοποιεί την ίδια εναλλαγή στο περιεχόμενο της θέσης μνήμης που δείχνει το ζεύγος καταχωρητών H-L, χωρίς να επηρεάζονται τα περιεχόμενα των υπόλοιπων καταχωρητών γενικού σκοπού.

```
SWAP Nible MACRO Q
        PUSH PSW    ;αποθήκευση περιεχομένου καταχωρητών A,F στη στοίβα
        MOV A,Q
        RLC
        RLC
        RLC
        RLC
        MOV Q,A     ;Q change is made
        MOV A,M
        RLC
        RLC
        RLC
        RLC
        MOV M,A     ;change in the memory content pointed by H-L registers is made
        POP PSW     ;επαναφορά περιεχομένου καταχωρητών A,F
ENDM
```

Σημείωση: στην περίπτωση που ο καταχωρητής Q είναι ο H ή ο L, πρώτα θα γίνει εναλλαγή των HEX ψηφίων ψηλότερης και μικρότερης αξίας του και μετά εναλλαγή στο περιεχόμενο της θέσης μνήμης που δείχνει το ζεύγος τους. Αυτό με την προϋπόθεση ότι μετά την πρώτη εναλλαγή θα παραμείνουμε σε διεύθυνση που αντιστοιχεί στην περιοχή προγράμματος χρήστη.

β) Μακροεντολή RHLR Q,R για περιστροφή του περιεχομένου του κρατουμένου CY και των καταχωρητών Q,R κατά μία θέση δεξιά. Δηλαδή, θέλουμε να υλοποιηθεί η λειτουργία (CY - Q - R):

Bit_17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 ->1 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2

```
RHLR MACRO Q,R

        PUSH B      ;αποθήκευση περιεχομένου του καταχωρητή B στη στοίβα
        MOV B,A     ;σώζω το περιεχόμενο του B στον A
        MOV A,Q
        RAR         ;δεξιά περιστροφή του καταχωρητή A μέσω της σημαίας κρατουμένου
                    ; Bit_9 17 16 15 14 13 12 11 10

        MOV Q,A     ;(Q)<-(A) και CY=Bit_9 (δεν επηρεάζεται η σημαία κρατουμένου)

        MOV A,R
        RAR         ;δεξιά περιστροφή του καταχωρητή A μέσω της σημαίας κρατουμένου
                    ; Bit_1 9 8 7 6 5 4 3 2

        MOV R,A     ;(R)<-(A)
        MOV A,B     ;επαναφορά του περιεχομένου του καταχωρητή A
        POP B       ;επαναφορά του περιεχομένου του καταχωρητή B από τη στοίβα

ENDM
```

Σημείωση: Στην παραπάνω υλοποίηση θεωρήθηκε ότι δε μας ενδιαφέρει να διατηρήσουμε το περιεχόμενο των flags πριν και μετά την εκτέλεση της μακροεντολής, εφόσον δεν αναφέρεται κάτι τέτοιο στην εκφώνηση της άσκησης, για αυτό δεν πραγματοποιήθηκε PUSH/POP PSW. Σώσαμε στη στοίβα μόνο το περιεχόμενο του καταχωρητή B για να μπορέσουμε να διατηρήσουμε το περιεχόμενο του A.