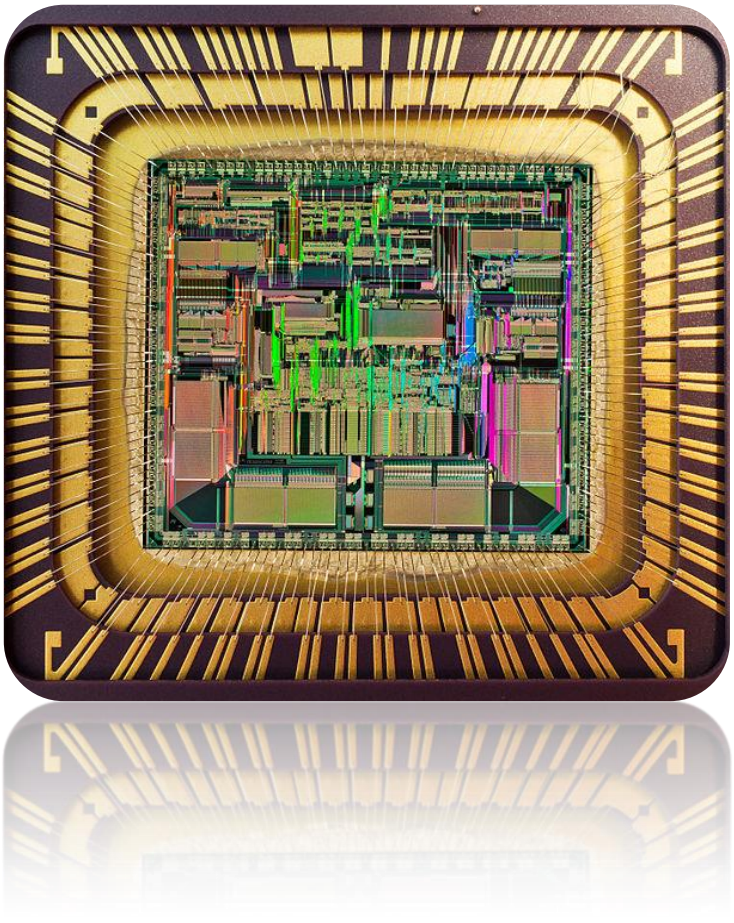
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΥΠΟΛΟΓΙΣΤΩΝ

3η Ομάδα Ασκήσεων

Βαβουλιώτης Γεώργιος
ΑΜ : 03112083
6ο Εξάμηνο

Γιαννόπουλος Αναστάσης
Α.Μ. : 03112176
6ο Εξάμηνο



ΑΣΚΗΣΗ 1

Ο κώδικας σε assembly 8085 για την πρώτη άσκηση φαίνεται παρακάτω:

```
CHECK_INPUT:    LDA 2000H               ;READ INPUT FROM SWITCHES
                MOV E,A                 ;SAVE THE INPUT
                MVI D,00H               ;OUTPUT REGISTER
                CPI 00H                 ;ALL LEDS ARE OFF
                JZ DISPLAY

                MOV A,E                 ;TAKE THE SAVED INPUT
                MVI B,00H               ;1ST ACE'S POSITION
                MVI C,00H               ;COUNTER


FIND_ACE:       RAR                     ;CHECK CURRENT LSB
                INR B                   ;POSITION++
                JC LED_ON       ;IF IS 1 THE LEDS MUST BE TURNED ON
                JNC FIND_ACE            ;ELSE CHECK FOR ACE

DISPLAY:        MOV A,D                 ;A = OUTPUT REGISTER
                CMA
                STA 3000H               ;THE OUTPUT WAS INVERSED
                JMP CHECK_INPUT         ;REPEAT PROCESS

LED_ON:         MVI A,08H           ;NUM OF RIGHT SHIFTS NEED TO DO
                SUB B
                MOV B,A
                INR B
                INR B ;B=08H-POS OF 1STACE+2=NUM OF RIGHT SHIFTS+1

PREPARE_OUTPUT: INR C                   ;COUNTER++
                MOV A,B                 ;A= NUM OF RIGHT SHIFTS + 1
                CMP C                   ;IF SHIFTS DONE THEN
                JZ DISPLAY              ;THEN GO TO OUTPUT
                MOV A,D           ;ELSE PREPARE THA OUTPUT REGISTER
                STC                     ;CY = 1
                RAR                ;WE PUT ACE IN D TO TURN ON LEDS
                MOV D,A                 ;UPDATE D
                JMP PREPARE_OUTPUT      ;CONTINUE PREPARING

END:            END
```

ΑΣΚΗΣΗ 2

Ο κώδικας σε assembly 8085 για την δεύτερη άσκηση φαίνεται παρακάτω:

```
START:          CALL KIND           ;CALLING KIND PROCCESS


                CPI 01H             ;IF INPUT < 1 THEN CY = 1
                JC START            ;CHECK AGAIN
                CPI 09H             ;IF INPUT >= 9 THEN CY = 0
                JNC START           ;CHECK AGAIN

                MVI B,01H           ;FIRST TURN ON THE LS LED
                MVI C,01H           ;C BECOMES 1 (SMALLEST NUMBER)

PREPARE:        CMP C               ;IF C=A
                JZ EXIT             ;FOUND PRESSED NUMBER, SO THE ;APROPRIATE
                                    LED TO TURN ON

                CALL ROTATE         ;ELSE TURN ON THE NEXT MS LED
                INR C               ;INCREASE C,
                JMP PREPARE         ;AND REPEAT THE PROCCESS

EXIT:           MOV A,B             ;OUTPUT IS READY
                CMA                 ;INVERSE THE OUTPUT
                STA 3000H           ;TURN ON THE RIGHT LEDS
                JMP START           ;CHECK FOR NEW INPUT

ROTATE:         PUSH PSW            ;SAVE A AND FLAGS IN STACK
                MOV A,B
                RLC                 ;TURN ON THE NEXT LED
                MOV B,A
                POP PSW             ;REGAIN A AND FLAGS
                RET

END:            END
```

ΑΣΚΗΣΗ 3

Ο κώδικας σε assembly 8085 για την τρίτη άσκηση φαίνεται παρακάτω:

```
      IN 10H            ;WITHOUT MEMORY'S PROTECTION

START:
      MVI A,FEH         ;CHECK LINE 0
      STA 2800H         ;STORE ADDRESS

INSTR_STEP:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0806H       ;CODE FROM KIND: 86.THIS IS THE PRINT VALUE
      CPI 06H           ;CHECK IF THIS IS THE PRESSED BUTTON
      JZ DISPLAY        ;IF IT IS DISPLAY IT IN THE 7-SEGMENT

FTCH_PC:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0805H
      CPI 05H           ;PRESSED BUTTON == 101 ?
      JZ DISPLAY

      ;FOR BUTTON HDWR_STEP WE DO NOTHING BECAUSE IT'S UNUSED

      MVI A,FDH         ;CHECK LINE 1
      STA 2800H

BUTTON_RUN:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0804H
      CPI 06H           ;PRESSED BUTTON == 110 ?
      JZ DISPLAY

FTCH_REG:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0800H
      CPI 05H           ;PRESSED BUTTON == 101 ?
      JZ DISPLAY

FTCH_ADR:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0802H
      CPI 03H           ;PRESSED BUTTON == 011 ?
      JZ DISPLAY

      MVI A,FBH         ;CHECK LINE 3
      STA 2800H

BUTTON0:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0000H
      CPI 06H
      JZ DISPLAY

STORE/INCR:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0803H
      CPI 05H
      JZ DISPLAY
```

```
DECR:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0801H
      CPI 03H
      JZ DISPLAY

      MVI A,F7H         ;CHECK LINE 4
      STA 2800H

BUTTON1:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0001H
      CPI 06H
      JZ DISPLAY

BUTTON2:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0002H
      CPI 05H
      JZ DISPLAY

BUTTON3:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0003H
      CPI 03H
      JZ DISPLAY

      MVI A,EFH         ;CHECK LINE 5
      STA 2800H

BUTTON4:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0004H
      CPI 06H
      JZ DISPLAY

BUTTON5:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0005H
      CPI 05H
      JZ DISPLAY


BUTTON6:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0006H
      CPI 03H
      JZ DISPLAY
      MVI A,DFH         ;CHECK LINE 6
      STA 2800H

BUTTON7:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0007H
      CPI 06H
      JZ DISPLAY
```

```
BUTTON8:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0008H
      CPI 05H
      JZ DISPLAY

BUTTON9:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,0009H
      CPI 03H
      JZ DISPLAY

      MVI A,BFH         ;CHECK LINE 7
      STA 2800H

BUTTONA:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,000AH
      CPI 06H
      JZ DISPLAY

BUTTONB:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,000BH
      CPI 05H
      JZ DISPLAY

BUTTONC:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,000CH
      CPI 03H
      JZ DISPLAY

      MVI A,7FH         ;CHECK LINE 8
      STA 2800H




BUTTOND:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,000DH
      CPI 06H
      JZ DISPLAY

BUTTONE:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,000EH
      CPI 05H
      JZ DISPLAY

BUTTONF:
      LDA 1800H         ;LOAD ADDRESS
      ANI 07H           ;KEEP THE LAST 3 LSB BIT
      LXI D,000FH
      CPI 03H
      JZ DISPLAY

      JMP START         ;PROGRAM NEVER STOPS
```

```
DISPLAY:
      LXI H,0A00H       ;MEMORY TO SAVE THE BUTTON CODE
      MVI B,10H         ;TURN OFF THE 4 LS DIGITS OF 7_SEGNMENT
      MOV M,B
      INX H
      MOV M,B
      INX H
      MOV M,B
      INX H
      MOV M,B
      INX H             ;(H-L)=(D-E)=PRINT VALUE
      MOV M,E
      INX H
      MOV M,D
      LXI D,0A00H       ;STARTING MEMORY POSITION FOR THE EXIT
      CALL STDM         ;CALL THE PRINTING PROCCESSES
      CALL DCD
      JMP START         ;REPEAT

END:  END
```

ΑΣΚΗΣΗ 4

Ο κώδικας σε assembly 8085 για την τέταρτη άσκηση φαίνεται παρακάτω:

```
      IN 10H
START:
      MVI A,FEH   ;CHECK LINE 0
      STA 2800H   ;STORE ADDRESS

INSTR:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,8082H
      CPI 06H
      JZ DISPLAY

FTCH_PC:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,8092H
      CPI 05H
      JZ DISPLAY

      MVI A,FDH
      STA 2800H

BUTTON_RUN:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,8099H
      CPI 06H
      JZ DISPLAY

FTCH_REG:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,80C0H
      CPI 05H
      JZ DISPLAY

FTCH_ADR:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,80A4H
      CPI 03H
      JZ DISPLAY

      MVI A,FBH
      STA 2800H


BUTTON0:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C0C0H
      CPI 06H
      JZ DISPLAY
```

```
STORE/INCR:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,80B0H
      CPI 05H
      JZ DISPLAY

DECR:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,80F9H
      CPI 03H
      JZ DISPLAY

      MVI A,F7H
      STA 2800H

BUTTON1:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C0F9H
      CPI 06H
      JZ DISPLAY

BUTTON2:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C0A4H
      CPI 05H
      JZ DISPLAY

BUTTON3:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C0B0H
      CPI 03H
      JZ DISPLAY

      MVI A,EFH
      STA 2800H

BUTTON4:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C099H
      CPI 06H
      JZ DISPLAY
BUTTON5:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C092H
      CPI 05H
      JZ DISPLAY
```

```
BUTTON6:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF
                  ;PRESSED BUTTON IS FOUND
      LXI D,C082H
      CPI 03H
      JZ DISPLAY

      MVI A,DFH
      STA 2800H

BUTTON7:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C0F8H
      CPI 06H
      JZ DISPLAY

BUTTON8:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C080H
      CPI 05H
      JZ DISPLAY

BUTTON9:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C090H
      CPI 03H
      JZ DISPLAY

      MVI A,BFH
      STA 2800H

BUTTONA:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C088H
      CPI 06H
      JZ DISPLAY

BUTTONB:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C083H
      CPI 05H
      JZ DISPLAY
BUTTONC:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C0A7H
      CPI 03H
      JZ DISPLAY

      MVI A,7FH
      STA 2800H
```

```
BUTTOND:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C0A1H
      CPI 06H
      JZ DISPLAY

BUTTONE:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C086H
      CPI 05H
      JZ DISPLAY

BUTTONF:
      LDA 1800H   ;LOAD ADDRESS
      ANI 07H     ;SAVE IN B THE BIN NUMBER 0000 0111 TO CHECK IF PRESSED
                  ;BUTTON IS FOUND
      LXI D,C08EH
      CPI 03H
      JZ DISPLAY

      JMP START


DISPLAY:
      MVI A,20H   ;1o MSB
      STA 2800H
      MOV A,D
      STA 3800H   ;OUTPUT
      MVI A,10H   ;2o MSB
      STA 2800H
      MOV A,E
      STA 3800H   ;OUTPUT
      MVI A,FFH   ;SWITCH OFF
      STA 3800H
      JMP START

END:  END
```

Άσκηση 5

Κώδικας 8085 για το μΥ-Σ 1:

```
      LXI D,0100H ; METRITIS GIA TA 256 DEDOMENA
BEGIN:
      MVI A,C0H   ; A = 11000000
      SIM         ; SOD1 = 1
WAIT1:
      RIM         ; ANAMONH MEXRI SOD2 = 1
      ANI 80H
      JZ WAIT1

SEND:
      MVI A,40H   ; A = 01000000
      SIM         ; SOD1 = 0
      MOV A,M     ;(A)<-(H)(L)
      OUT DATA1   ; APOSTOLH DEDOMENWN 8 BIT

WAIT0:
      RIM         ; ANAMONH MEXRI SOD2 = 0
      ANI 80H
      JNZ WAIT0
      DCX D       ; OTAN OLOKLHRWTHEI H APOSTOLH (D)<-(D)-1
      JZ FINISH   ; AN O D MHDENISTEI, STAMATAEI H APOSTOLH
      INX H       ; ALLIWS EKKINEI H APOSTOLH NEOY DEDOMENOY
      JMP START

FINISH:
      END
```

Κώδικας 8085 για το μΥ-Σ 2:

```
      LXI D,0100H ; METRITIS GIA TA 256 DEDOMENA
BEGIN:
      RIM         ; ANAMONH MEXRI SOD1 = 1
      ANI 80H
      JZ  START

READY:
      MVI A,C0H   ; A=11000000
      SIM         ; SOD2 = 1

WAIT0:
      RIM         ; ANAMONH MEXRI SOD1 = 0
      ANI 80H
      JNZ WAIT0
      IN  DATA 2  ; READ INPUT
      MOV M,A     ; (H)(L)<-(A)
      DCX D       ; (D)<-(D)-1
      JZ  FINISH  ; AN O D MHDENISTEI, STAMATAEI H APOSTOLH
      INX H       ; ALLIWS AKKOLOYTHEI LHPSH NEOY DEDOMENOY
      JMP START

FINISH:
      END
```

Μέγιστη Ταχύτητα Μεταφοράς Δεδομένων:

Για τον υπολογισμό της μέγιστης ταχύτητας μεταφοράς δεδομένων απαιτείται πρώτα η μέτρηση των κύκλων CPU που απαιτούνται για την αποστολή ενός byte. Επειδή ζητείται μέγιστη ταχύτητα, υποθέτουμε ότι δεν υπάρχει καμία καθυστέρηση κατά την αποστολή δεδομένων από το ένα μΥ-Σ στο άλλο.
Έτσι υπολογίζουμε ότι για την αποστολή ενός byte απαιτούνται 49 κύκλοι CPU. Επειδή ο συγκεκριμένος μΕ 8085 λειτουργεί στα 5MHz, κάθε κύκλος διαρκεί 0,2 μsec. Επομένως για την αποστολή ενός byte απαιτούνται συνολικά 49*0,2 = 9,8 μsec. Άρα σε 1sec πραγματοποιείται η αποστολή 10^6/9,8 = 102040,8163265 bytes.
Τελικά η *μέγιστη ταχύτητα μεταφοράς* δεδομένων είναι ***102,041 Mbps*****.**

Άσκηση 6

Στην άσκηση αυτή θα σχεδιάσουμε ένα μΥ-Σ 8085 με την εξής χάρτη μνήμης:

0000-1FFF Hex : EPROM
2000-3FFF Hex : RAM
4000-5FFF Hex : EPROM
6000 Hex : θύρα εξόδου (Memory map I/O)
60 Hex : θύρα εισόδου (Standard I/O)
80 Hex : θύρα εισόδου-εξόδου (Standard I/O)

Όπως είναι γνωστό, ο 8085 έχει 16-bit διευθύνσεις και 8-bit περιεχόμενα διευθύνσεων. Προκύπτει ο πίνακας:

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	bits
EPROM0																	
Αρχή	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Bin
		0				0				0				0			Hex
Τέλος	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	Bin
		1				F				F				F			Hex
SRAM0																	
Αρχή	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Bin
		2				0				0				0			Hex
Τέλος	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	Bin
		2				F				F				F			Hex
SRAM1																	
Αρχή	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	Bin
		3				0				0				0			Hex
Τέλος	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	Bin
		3				F				F				F			Hex
EPROM1																	
Αρχή	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Bin
		4				0				0				0			Hex
Τέλος	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	Bin
		5				F				F				F			Hex

Σχόλιο: Απ' τον παραπάνω πίνακα φαίνεται ότι για τον προσδιορισμό όλων των διευθύνσεων είτε EPROM είτε RAM, χρειάζονται ακριβώς **13 bits** (από A_0 - A_{12}). Συνέπεια αυτού είναι οτι κάθε περιοχή μνήμης του χάρτη μνήμης έχει μέγεθος 2^{13}B = 8KB. Η μνήμη RAM επιλέχθηκε ως μια δυάδα από SRAM 4K x 8 bits σύμφωνα με τα διαθέσιμα υλικά. Η μνήμη EPROM επιλέχθηκε με μέγεθος 16KB, αφού συγκροτείται από δύο περιοχές στον χάρτη μνήμης, την περιοχή από 0000 - 1FFFH και την περιοχή από 4000 - 5FFFH μεγέθους 8KB η κάθε μία. Περισσότερες λεπτομέρειες υπάρχουν παρακάτω. Σημειώνεται πως η RAM είναι μία συνεχής περιοχή μνήμης. Επίσης, από τον χάρτη μνήμης προκύπτει οτι υο A_{15} χρησιμεύει ως είσοδος επίτρεψης, ενώ τα bits από A_{12} - A_{14} χρειάζονται για τον μοναδικό προσδιορισμό κάθε ολοκληρωμένου κυκλώματος μνήμης. Συμπερασματικά, εξάγεται ο ακόλουθος

Πίνακας Αποκωδικοποίησης:

A/A	A14=A	A13=B	A12=C	EPROM	SRAM0	SRAM1
0	0	0	0	1	0	0
1	0	0	1	1	0	0
2	0	1	0	0	1	0
3	0	1	1	0	0	1
4	1	0	0	1	0	0
5	1	0	1	1	0	0
6	1	1	0	0	0	0
7	1	1	1	0	0	0

Ο παραπάνω πίνακας έχει πεδίο ορισμού A, B, C και εξόδους τα enable των μνημών. Ο αποκωδικοποιητής 3 σε 8 είναι αντίστροφης λογικής, όπως και τα enable των μνημών. Γι' αυτό οι άσοι είναι μηδενικά και τα μηδενικά άσοι στη πραγματικότητα. Οι τύποι των λογικών συναρτήσεων στη μορφή αθροίσματος ελαχιστόρων είναι:

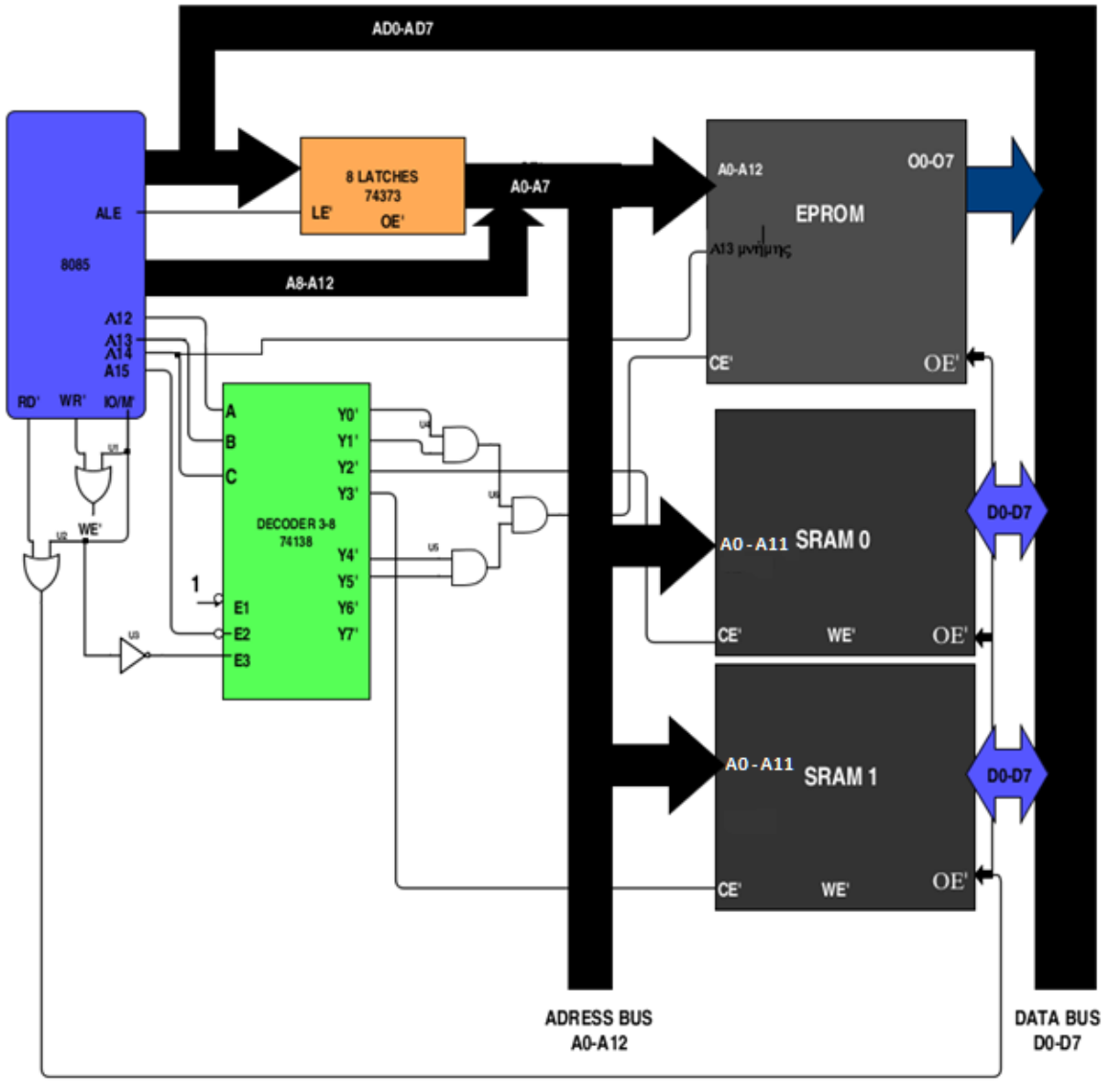
- **EPROM = Σ(0,1,4,5)**
- **SRAM0 = Σ(2)**
- **SRAM1 = Σ(3)**

Για να αποφευχθούν όλα αυτά τα προβλήματα γίνεται μετατόπιση της δεύτερης περιοχής στα όρια μιας μνήμης των 16KB. Συγκεκριμένα γίνεται χρήση μιας 16KB μνήμης που έχει διευθύνσεις από 0000 - 3FFFH. Η πρώτη περιοχή του χάρτη μνήμης και της EPROM προφανώς αντιστοιχεί επακριβώς στην περιοχή από 0000 - 1FFFH της μνήμης. Η τρίτη περιοχή του χάρτη μνήμης από 4000 - 5FFFH αντιστοιχίζεται στην περιοχή από 2000 - 3FFFH της μνήμης. Ένα θέμα που ανακύπτει τώρα είναι πως θα διακρίνονται οι διευθύνσεις της πρώτης περιοχής του χάρτη μνήμης από την τρίτη. Παρατηρώντας τον αρχικό χάρτη μνήμης διαπιστώνεται πως οι δύο περιοχές της EPROM ως προς τα bits A_{11} - A_{14}, διαφέρουν μόνο ως προς το A_{14}. Όταν αυτό ισούται με 0 τότε λαμβάνεται η πρώτη περιοχή, ενώ όταν αυτό ισούται με 1, λαμβάνεται η δεύτερη περιοχή. Το ίδιο προκύπτει κι από τον χάρτη μνήμης της μνήμης των 16KB με την διαφορά οτι τον ρόλο του A_{14} έχει το A_{13}, γεγονός αναμενόμενο. Αναμενόμενο, γιατί μια μνήμη 16KB απαιτεί 14 bits από A_0 - A_{13} για τον προσδιορισμό όλων των δυνατών διευθύνσεων. Στην σχεδίαση λοιπόν του συστήματος στο A_{13} που αφορά στην μνήμη εισέρχεται το A_{14} από τον 8085. Στον επόμενο χάρτη τα ονόματα EPROM1, EPROM2 αναφέρονται στις δύο περιοχές της EPROM του αρχικού χάρτη αντίστοιχα.

Χάρτης Μνήμης 16 KB :

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	*bits*
EPROM0																	
Αρχή	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**Bin**
		0				0				0				0			**Hex**
Τέλος	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	**Bin**
		1				F				F				F			**Hex**
EPROM1																	
Αρχή	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	**Bin**
		2				0				0				0			**Hex**
Τέλος	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**Bin**
		3				F				F				F			**Hex**

Έτσι έχουμε τη σχεδίαση:

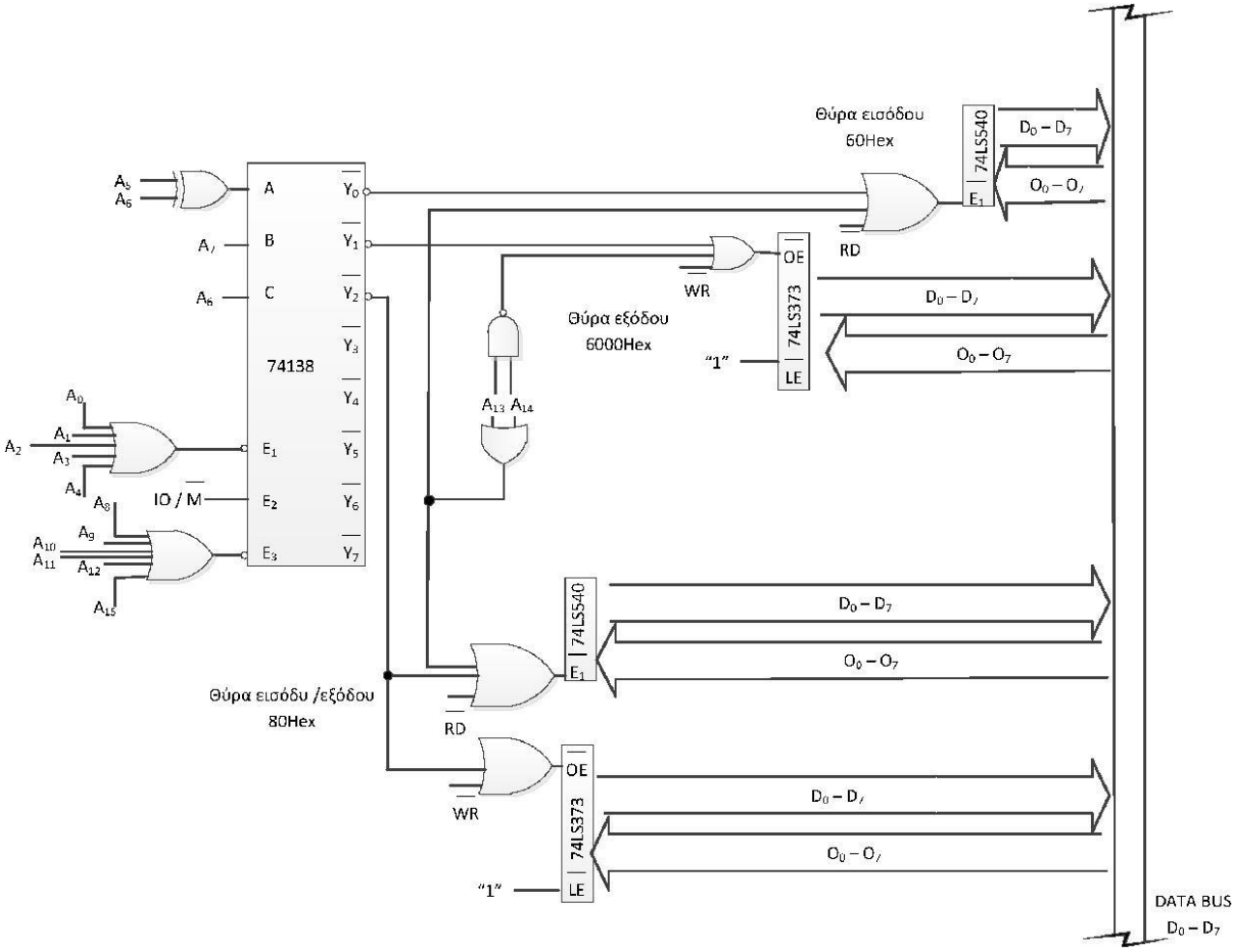


Είσοδος – έξοδος μΥ-Σ:

Τέλος, για την είσοδο και την έξοδο του συστήματος απορρέει ο χάρτης μνήμης,

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
6000H																
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
60H																
	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
80H																
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Τα bits χωρίζονται σε τρεις ομάδες, κάθε μια εκ των οποίων χρωματίζεται διαφορετικά. Η πράσινη ομάδα χρησιμεύει ως είσοδος επίτρεψης (όπως φαίνεται λεπτομερέστερα στο σχήμα). Η μπλε ομάδα χρειάζεται, ώστε μετά την αποκωδικοποίηση να επιλέγεται η σωστή θύρα σύμφωνα με τις τιμές των A_{13}, A_{14}. Τέλος, η κόκκινη ομάδα καθορίζει τις τιμές στη είσοδο του αποκωδικοποιητή. Διευκρινίζεται πως το σχήμα που αφορά στην Ε/Ε, ως κύκλωμα βρίσκεται μαζί με το υπόλοιπο κύκλωμα που παρουσιάζεται παραπάνω. Η ξεχωριστή παρουσίαση έγινε για να δοθεί η προσοχή που αρμόζει σε κάθε μέρος του μΥ συστήματος και για να είναι πιο ευδιάκριτη η σχεδίαση.



Άσκηση 7

(α) Η υλοποίηση της μακροεντολής SWAP Q,R φαίνεται παρακάτω :

```
SWAP MACRO Q,R

      MOV A,R
      MOV R,Q
      MOV Q,A
ENDM
```

(β) Η υλοποίηση της μακροεντολής FILL ADDR,L,K φαίνεται παρακάτω :

```
FILL MACRO ADDR,L,K

      LXI H,ADDR  ;FIRST ELEMENT
      MVI C,L     ;STORE LENGTH OF MEMORY PART

LOOP:
      MVI M,K     ;FILL THE MEMORY PART
      INX H       ;NEXT CELL OF MEMORY
      DCR C       ;(C)=(C)-1
      JNZ LOOP    ;IF THE MEMORY PART IS NOT FULL
END:
      ENDM
```

(γ) Η υλοποίηση της μακροεντολής RHLL φαίνεται παρακάτω :

```
RHLL MACRO n

    PUSH B            ;VAZW TON BC STIN STOIVA
    PUSH H            ;VAZW TON HL STIN STOIVA
    MVI B,n           ;(B) = n(>0)
    MVI C,00H         ;COOUNTER->NUM OF SHIFTS

KEEP_SHIFTING:
    MOV A,H
    RAL               ;SHIFT LEFT H
    JC CARRY_H_1      ;IF CY=1 THEN THE LAST BIT OF L IS 1
    JNC CARRY_H_0     ;ELSE THE LAST BIT OF L IS 0

CONTINUE:
    INR C             ;COUNTER++
    MOV A,C
    CMP A,B           ;IF THE n SHIFTS DONE
    JZ FINISH         ;THEN FINISH
    JNZ KEEP_SHIFTING ;ELSE KEEP SHIFTING

CARRY_H_1:
    MVI E,01H         ;E -> FLAG TO REMEMBER THE CY OF H
    MOV A,L
    RAL               ;SHIFT LEFT L
    JC CARRY_L_1      ;IF CY=1 THEN THE LAST BIT OF H IS 1
    JNC CARRY_L_0     ;ELSE THE LAST BIT OF H 0
```

```
CARRY_H_0:
  MVI E,00H          ;E -> FLAG TO REMEMBER THE CY OF H
  MOV A,L
  RAL                ;SHIFT LEFT L
  JC CARRY_L_1       ;IF CY=1 THEN THE LAST BIT OF H IS 1
  JNC CARRY_L_0      ;ELSE THE LAST BIT OF H IS 0

CARRY_L_1:
  MOV A,H
  ORI 01H            ;THE LAST BIT OF H = 1
  MOV H,A
  JMP CHECK          ;CHECK THE FLAG E

CARRY_L_0:
  MOV A,H
  ANI FEH            ;DO THE LAST BIT OF H = 0
  MOV H,A
  JMP CHECK          ;CHECK THE FLAG E

CHECK:
  MOV A,E
  CPI 00H            ;IF FLAG = 0 THEN
  JZ NEXT0           ;THE LAST BIT OF L = 0
  JNZ NEXT1          ;ELSE DO THE LAST BIT OF L = 1

NEXT0:
  MOV A,L
  ANI FEH            ;THE LAST BIT OF L = 0
  MOV L,A
  JMP CONTINUE

NEXT1:
  MOV A,L
  ORI 01H            ;THE LAST BIT OF L = 1
  MOV L,A
  JMP CONTINUE

FINISH:
  POP H              ;VGAZW TON HL APO TIN STOIVA
  POP B              ;VGAZW TON BC AP TIN STOIVA
  ENDM
```

Ιδέα της υλοποίησης: Ο παραπάνω κώδικας αξιοποιεί τα παρακάτω:

- Το κρατούμενο που προκύπτει απ' την ολίσθηση του H είναι το τελευταίο bit του L.
- Το κρατούμενο που προκύπτει απ' την ολίσθηση του L είναι το τελευταίο bit του H.

Σχόλιο: Θα μπορούσαμε για εξοικονόμηση χώρου να ορίσουμε την μακροεντολή `SET_THE_LAST_BIT_OF R,n` η οποία παίρνει έναν καταχωρητή R και θέτει το τελευταίο του bit ίσο με n = 00H ή 01H. Η εν λόγω μακροεντολή θα είχε ως εξής:

```
SET_THE_LAST_BIT_OF MACRO R,n
       MVI A,n
       CPI 00H
       JZ SET_0
       JNZ SET_1
   SET_0:
       MOV A,R
       ANI FEH
       MOV R,E
       JMP FINISH
   SET_1:
       MOV A,R
       ORI 01H
       MOV R,E
       JMP FINISH
   FINISH:
       ENDM
```