ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ 2η ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ

ΦΟΙΤΗΤΕΣ

Αθανασίου Ιωάννης / Α.Μ.:03117041 /6ο Εξάμηνο

Καραβαγγέλης Αθανάσιος / Α.Μ.:03117022 / 6ο Εξάμηνο

ΑΣΚΗΣΗ 1η

- Ο κώδικας μας σε assembly του 8085 για κάθε ερώτημα αντίστοιχα δίνεται παρακάτω και επισυνάπτονται και τα εκτελέσιμα προγράμματα στον 8085 «s2askisi1i.8085» , «s2askisi1ii.8085», «s2askisi1iii.8085» αντίστοιχα.

i)

```
IN 10H

;ARXIKOPOIHSH

MVI A,FFH ;A<-255
MVI H,09H ;
MVI L,00H ; for the address

;EPANALHPTIKH

LOOP1:  CPI 00H ; if A<0 ->telos
        JZ MY_END ;  else ->stay in the loop1
        MOV M,A ;   M <- A
        INR L ;      epomeni thesi mnimis
        DCR A ;    A--
        JMP LOOP1 ;   again

MY_END:
       MOV M,A
END
```

ii)

```
IN 10H

MVI H,09H
MVI L,00H
MVI A,FFH
MVI D,00H
MVI E,08H

LIST_LOOP:
               CPI 00H
               JZ LIST_END
               CALL ZEROS_OF_A
               DCR A
               JMP LIST_LOOP

ZEROS_OF_A:
               PUSH PSW
               MOV B,A
               PUSH B
               MVI C,00H
               MOV A,C

LOOP2:         CPI 08H
               JZ END_LOOP2
               MOV A,B
               RRC
               JNC D_INCREASE
D_BACK:
               MOV B,A
               INR C
               MOV A,C
               JMP LOOP2

D_INCREASE:
               INX D
               JMP D_BACK
```

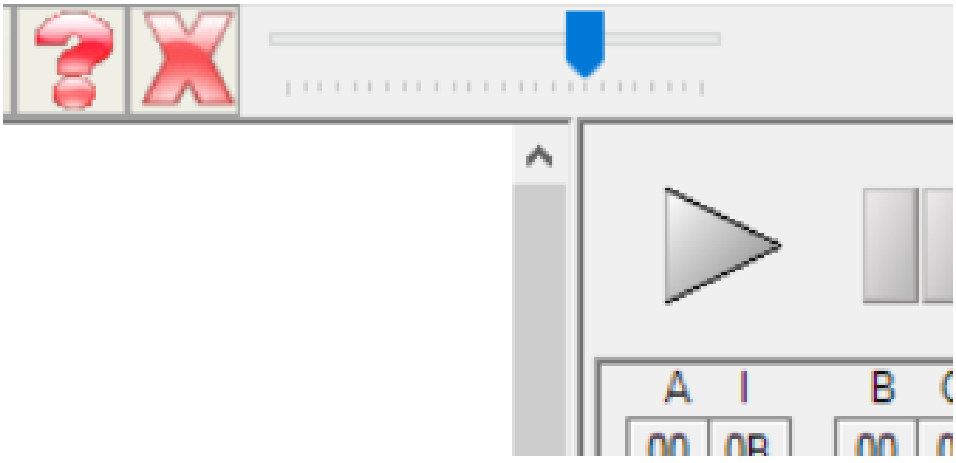
```
END_LOOP2:
                POP PSW
                POP B
                MOV A,B
                RET

LIST_END:
                RST 1
                END
```

iii)

```
IN 10H

;ARXIKOPOIHSH

MVI A,FFH
MVI C,00H
;EPANALHPTIKH

LOOP1:  CPI 20H ; if A<20 ->telos
        JC MY_END ;
        JMP CHECK_70H_SMALLER

CHECK_70H_SMALLER:
        CPI 70H
        JC DEKTO
        JZ DEKTO
        DCR A ;    A--
        JMP LOOP1

DEKTO:
        INR C
        DCR A ;    A--
        JMP LOOP1
MY_END:
       RST 1
       END
```

ΑΣΚΗΣΗ 2η

- Ο κώδικας μας σε assembly δίνεται παρακάτω και επισυνάπτεται και το αντίστοιχο αρχείο «s2askisi2.8085».
  Επίσης,για τη σωστή λειτουργία και συγκεκριμένα το σωστό χρόνο των 15 sec πρέπει να θέσουμε τη μπάρα τεχνητής καθυστέρησης όπως φαίνεται παρακάτω:



```
MVI B,01H ;OUR DELAY = 1MS*500 = 0.5 SEC
MVI C,F4H

BEGIN:

       LDA 2000H           ;READ INPUT
       RAR                 ;CHECK LSB
       JC BEGIN            ;IF LSB NOT 0 READ AGAIN THE INPUT

SWITCH_OFF_ON:

       LDA 2000H           ;READ INPUT
       RAR                 ;SWITCH UP -> LSB=1
       JNC SWITCH_OFF_ON   ;IF LSB == 0 CONTINUE READING UNTIL IT BECOMES 1 ELSE CONTINUE

SWITCH_OFF_ON_OFF:

       LDA 2000H           ;READ INPUT
       RAR                 ; CHECKING LSB (WE WANT IT TO BE 0)
       JC SWITCH_OFF_ON_OFF ;CONTINUE READING UNTIL IT BECOMES 0 AGAIN


LEDS_ON:
                           ;WE ARE HERE IF SWITCH GOES DOWN->UP->DOWN SO LEDS ARE ON
       MVI D,00H           ;OFF-ON-OFF FLAG
       MVI E,0FH           ;E = 15 DECIMAL FOR THE DELAY (15(2*0.5) = 15 SEC)


DELAY:
       MVI A,00H           ; SWITCHING ON AND OFF ALL LEDS
       STA 3000H
       CALL DELB           ;0.5 SECOND DELAY
       MVI A,FFH
       STA 3000H
       CALL DELB           ;0.5 SECOND DELAY

       DCR E               ;DECRIMENT BY 1 , STARTING FROM 30
       MOV A,E             ;CHECK IF 15 SECONDS HAVE PASSED
       CPI 00H
       JZ TURN_OFF_LEDS    ;YES? CLOSE LEDS
       MOV A,D             ;NO? CHECK D TO SEE OUR CURRENT SITUATION
       CPI 00H
       JZ LEDS_ARE_ON      ;IF D=0 OUR LEDS ARE ON. ELSE CHECK WHERE IT IS NOW
       CPI 01H
       JZ OFF_ON           ;IF D=1 OUR SWITCH IS ON AFTER BEING OFF.CHECK WHERE IT IS NOW
       CPI 02H ;
       JZ OFF_ON_OFF       ;IF D=2 OUR SWITCH IS OFF-ON-OFF. CHECK WHERE IT IS NOW

TURN_OFF_LEDS:

       MVI A,FFH
       STA 3000H           ;CLOSING ALL LEDS
       JMP BEGIN           ;RESTART OPERATION

LEDS_ARE_ON:

       LDA 2000H           ; READ INPUT
       RAR
       JC DELAY            ;IF SWITCH IS ON THEN GO TO DELAY ELSE WE ARE IN OFF STATE
       INR D               ;STORE OFF STATE =1


OFF_ON:                    ;NOW D = 1
       LDA 2000H
       RAR
       JNC DELAY           ;IF SWITCH IS OFF THEN GO TO DELAY ELSE WE ARE IN OFF-ON STATE
```

```
        INR D               ;STORE OFF - ON STATE = 2


OFF_ON_OFF:                 ; NOW D = 2
        LDA 2000H
        RAR
        JC DELAY            ; IF SWITCH STILL ON GO TO DELAY ELSE WE ARE IN OFF - ON - OFF
        STATE

REMAINS_ON:
        JMP LEDS_ON         ;RELOAD TIME LIMIT
        END
```

ΑΣΚΗΣΗ 3η

- Ο κώδικας μας σε assembly του 8085 για κάθε ερώτημα αντίστοιχα δίνεται παρακάτω και επισυνάπτονται και τα εκτελέσιμα προγράμματα στον 8085 «s2askisi3i.8085» , «s2askisi3ii.8085», «s2askisi3iii.8085» αντίστοιχα.

i)

```
LDA 2000H

MVI C,00H
MOV B,A

MASTER_LOOP:
        MOV A,C

        CPI 08H
        JZ C_EQ_8
        MOV A,B
        RLC
        MOV B,A
        JC FIX_OUTPUT
        INR C
        JMP MASTER_LOOP

FIX_OUTPUT:
        MOV A,C

        CPI 00H
        JZ C_EQ_0

        CPI 01H
        JZ C_EQ_1

        CPI 02H
        JZ C_EQ_2

        CPI 03H
        JZ C_EQ_3

        CPI 04H
        JZ C_EQ_4

        CPI 05H
        JZ C_EQ_5

        CPI 06H
        JZ C_EQ_6

        CPI 07H
        JZ C_EQ_7
```

```
C_EQ_0: MVI A,7FH
        JMP THE_RET

C_EQ_1: MVI A,3FH
        JMP THE_RET

C_EQ_2: MVI A,1FH
        JMP THE_RET

C_EQ_3: MVI A,0FH
        JMP THE_RET

C_EQ_4: MVI A,07H
        JMP THE_RET

C_EQ_5: MVI A,03H
        JMP THE_RET

C_EQ_6: MVI A,01H
        JMP THE_RET

C_EQ_7: MVI A,00H
        JMP THE_RET

C_EQ_8: MVI A,FFH
        JMP THE_RET


THE_RET:STA 3000H
        MVI C,00H
        LDA 2000H
        MOV B,A
        JMP MASTER_LOOP

END
```

ii)

```
MVI B,10H

READ_INPUT:
        CALL KIND
        MVI C,00H
        CPI 05H
        JC FIX_LEDS_1
        CPI 09H
        JNC READ_INPUT
        JMP FIX_LEDS_2

FIX_LEDS_1:
        MVI D,F0H
        JMP LOOP1

FIX_LEDS_2:
        MVI D,00H
        JMP LOOP1

LOOP1:  MOV A,C
        CPI 04H
        JZ READ_INPUT
        MOV A,D
        STA 3000H
        CALL DELB
        MVI A,FFH
        STA 3000H
        CALL DELB
        INR C
        JMP LOOP1

END
```

iii)

```
IN 10H
                        ;WE WILL CHECK EVERY LINE TO SEE WHAT HAS BEEN PRESSED
                        ;AND IF IT HAS BEEN PRESSED WE'LL PRINT IT

LINE1:

MVI A,7FH               ;01111111 -> WE ARE CHOOSING LINE 1
STA 2800H               ; STORE IT AT 2800H THAT CHOOSES THE LINE
LDA 1800H               ;LOADING MEMORY FROM 1800H
ANI 07H                 ;"AND WITH" 00000111 BECAUSE WE WANT TO KEEP THE 3 LAST DIGITS
RAR
JNC D                   ; IF 1ST BIT=0 WE HAVE PRESSED D
RAR
JNC E                   ; SAME FOR E
RAR
JNC F                   ; SAME FOR F

LINE2:

MVI A,BFH               ;10111111 -> CHOOSING LINE 2
STA 2800H               ;STORE IT AT 2800H THAT CHOOSES THE LINE
LDA 1800H               ;LOADING MEMORY FROM 1800H
ANI 07H                 ;"AND WITH" 00000111 BECAUSE WE WANT TO KEEP THE 3 LAST DIGITS
RAR
JNC A                   ; IF 1ST BIT=0 WE HAVE PRESSED A
RAR
JNC B                   ; SAME FOR B
RAR
JNC C                   ; SAME FOR C

                        ;SAME PROCEDURE FOR THE NEXT LINES TOO
LINE3:

MVI A,DFH
STA 2800H
LDA 1800H
ANI 07H
RAR
JNC N7
RAR
JNC N8
RAR
JNC N9


LINE4:

MVI A,EFH
STA 2800H
LDA 1800H
ANI 07H
RAR
JNC N4
RAR
JNC N5
RAR
JNC N6

LINE5:

MVI A,F7H
STA 2800H
LDA 1800H
ANI 07H
RAR
JNC N1
RAR
JNC N2
```

```
RAR
JNC N3

LINE6:

MVI A,FBH
STA 2800H
LDA 1800H
ANI 07H
RAR
JNC N0
RAR
JNC STORE_INCR
RAR
JNC DECR


LINE7:

MVI A,FDH
STA 2800H
LDA 1800H
ANI 07H
RAR
JNC RUNB
RAR
JNC FETCH_REG
RAR
JNC FETCH_ADDRESS

LINE8:

MVI A,FEH
STA 2800H
LDA 1800H
ANI 07H
RAR
JNC INSTR_STEP
RAR
JNC FETCH_PC
RAR
JNC HDWR_STEP

NONE_PRESSED:
MVI D,10H                    ;DELETING 7 SEGMENT
MVI E,10H
JMP PRINT

D:
MVI D,0DH
MVI E,00H
JMP PRINT

E:
MVI D,0EH
MVI E,00H
JMP PRINT

F:
MVI D,0FH
MVI E,00H
JMP PRINT

A:
MVI D,0AH
MVI E,00H
JMP PRINT

B:
```

```
MVI D,0BH
MVI E,00H
JMP PRINT

C:
MVI D,0CH
MVI E,00H
JMP PRINT

N7:
MVI D,07H
MVI E,00H
JMP PRINT

N8:
MVI D,08H
MVI E,00H
JMP PRINT

N9:
MVI D,09H
MVI E,00H
JMP PRINT

N4:
MVI D,04H
MVI E,00H
JMP PRINT

N5:
MVI D,05H
MVI E,00H
JMP PRINT

N6:
MVI D,06H
MVI E,00H
JMP PRINT

N1:
MVI D,01H
MVI E,00H
JMP PRINT

N2:
MVI D,02H
MVI E,00H
JMP PRINT

N3:
MVI D,03H
MVI E,00H
JMP PRINT

N0:
MVI D,00H
MVI E,00H
JMP PRINT

STORE_INCR:
MVI D,03H
MVI E,08H
JMP PRINT

DECR:
MVI D,01H
MVI E,08H
JMP PRINT
```

```
RUNB:
MVI D,04H
MVI E,08H
JMP PRINT

FETCH_REG:
MVI D,00H
MVI E,08H
JMP PRINT

FETCH_ADDRESS:
MVI D,02H
MVI E,08H
JMP PRINT

INSTR_STEP:
MVI D,06H
MVI E,08H
JMP PRINT

FETCH_PC:
MVI D,05H
MVI E,08H
JMP PRINT

HDWR_STEP:
JMP LINE1

PRINT:
LXI H,0A00H                 ; MEMORY POSITION FOR PRINTING
MVI B,10H                   ; DELETING 4 LSB OF 7 SEGMENT DISPLAY
MOV M,B
INX H
MOV M,B
INX H
MOV M,B
INX H
MOV M,B
INX H                       ; POSITION FOR 2ND MSB OF OUR BUTTON
MOV M,D                     ; THAT IS STORED IN REGISTER D
INX H                       ; POSITION FOR THE MSB OF OUR BUTTON
MOV M,E                     ; THAT IS STORED IN REGISTER E
LXI D,0A00H
CALL STDM
CALL DCD
JMP LINE1
END
```

ΑΣΚΗΣΗ 4η

- Ο κώδικας μας σε assembly δίνεται παρακάτω και επισυνάπτεται και το αντίστοιχο αρχείο «s2askisi4.8085».

```
START:
MVI B,00H                   ; WE KEEP TEMPORARILY THE RESULT OF THE GATE
LDA 2000H                   ; LOADING THE INPUT FROM REGISTER A
STC
CMC                         ; CY=0
MVI C,00H                   ; INITIALIZE THE REGISTERS WE ARE GOING TO USE
MVI D,00H
MVI E,00H
MVI H,00H
RAR                         ; CHECK 1ST SWITCH
JC 1ST_OR_FROM_1ST_BIT
RAR                         ; CHECK 2ND SWITCH
JC 1ST_OR_1
JNC 1ST_OR_RESULT
```

```
1ST_OR_FROM_1ST_BIT:            ; SINCE WE FOUND A 1 IN THE 1ST BIT
RAR                             ; WE NEED ONE MORE RIGHT ROTATION (SKIP 2ND SWITCH)

1ST_OR_1:                       ; IF OR=1
INR B                           ; B = B + 1

1ST_OR_RESULT:
MOV C,B                         ; STORE RESULT IN C
MVI B,00H                       ; INITIALIZE B FOR NEXT GATE
RAR                             ; CHECK 3RD SWITCH
JNC 1ST_AND_0_FROM_1ST_BIT      ; IF 0 -> AND = 0
RAR                             ; CHECK 4TH SWITCH
JNC 1ST_AND_RESULT              ; IF 0 -> AND = 0
INR B                           ; ELSE AND = 1 ,DO B = B + 1
JC 1ST_AND_RESULT               ; CY = 1 IF AND = 1 (B1 = A1 = 1)

1ST_AND_0_FROM_1ST_BIT:
RAR                             ; SKIP 4TH SWITCH

1ST_AND_RESULT:
MOV D,B                         ; STORE RESULT IN D
MVI B,00H                       ; INITIALIZE B
RAR                             ; CHECK 5TH SWITCH
JC 2ND_OR_FROM_1ST_BIT          ; IF CY = 1 -> OR = 1
RAR                             ; CHECK 6TH SWITCH
JC 2ND_OR_1                     ; IF CY = 1 -> OR = 1
JNC 2ND_OR_RESULT

2ND_OR_FROM_1ST_BIT:
RAR                             ; SKIP 6TH SWITCH

2ND_OR_1: ; IF OR = 1
INR B ; B = B + 1

2ND_OR_RESULT:
MOV E,B                         ; STORE RESULT IN E
MVI B,00H                       ; INITIALIZE B FOR NEXT GATE
RAR                             ; CHECK 7TH SWITCH
JNC 2ND_AND_0_FROM_1ST_BIT      ; IF CY = 0 -> AND2 = 0
RAR                             ; CHECK 8TH AND LAST SWITCH
JNC 2ND_AND_RESULT              ; IF CY = 0 -> AND2 = 0
INR B                           ; ELSE AND = 1 , DO B = B + 1
JC 2ND_AND_RESULT               ; RESULT OF AND2 = 1

2ND_AND_0_FROM_1ST_BIT:
RAR                             ; SKIP 8TH SWITCH

2ND_AND_RESULT:
MOV H,B                         ; STORE RESULT OF AND2 IN H;
MOV B,A                         ; STORE A IN B TEMPORARILY
MOV A,H                         ; TAKE THE RESULT OF AND 2
CMP E                           ; AND COMPARE IT WITH THE RESULT OF E
JZ XOR_0                        ; IF THEY ARE THE SAME XOR = 0
MVI H,01H                       ; H = 1
JNZ XOR_1                       ; XOR = 1

XOR_0:
MVI H,00H                       ; H = 0

XOR_1:
CMC                             ; CY=0
MVI A,00H                       ; A = 00000000
ADD H                           ; A = 0000000V WHERE X = OR1
RAL
ADD E                           ; A = 000000VZ WHERE Y = AND1
RAL
ADD D                           ; A = X0000VZY WHERE Z = OR2
RAL
ADD C                           ; A = 0000VZYX WHERE V = XOR
```

```
CMA
STA 3000H                          ; LEDS ON
JMP START
END
```

ΑΣΚΗΣΗ 5

Η μνήμη που σχεδιάσαμε είναι μεγέθους 128x4bits. Επομένως (128=2^7), χρειάζονται 7 bits-επιλογείς (A0, A1, ...A6) για την επιλογή της εκάστοτε θέσης μνήμης όπου θα γίνει εγγραφή ή ανάγνωση. Επιλέγουμε να αναπαραστήσουμε τα 4 bits της κάθε λέξης με τα σήματα D0, D1, D2, D3.

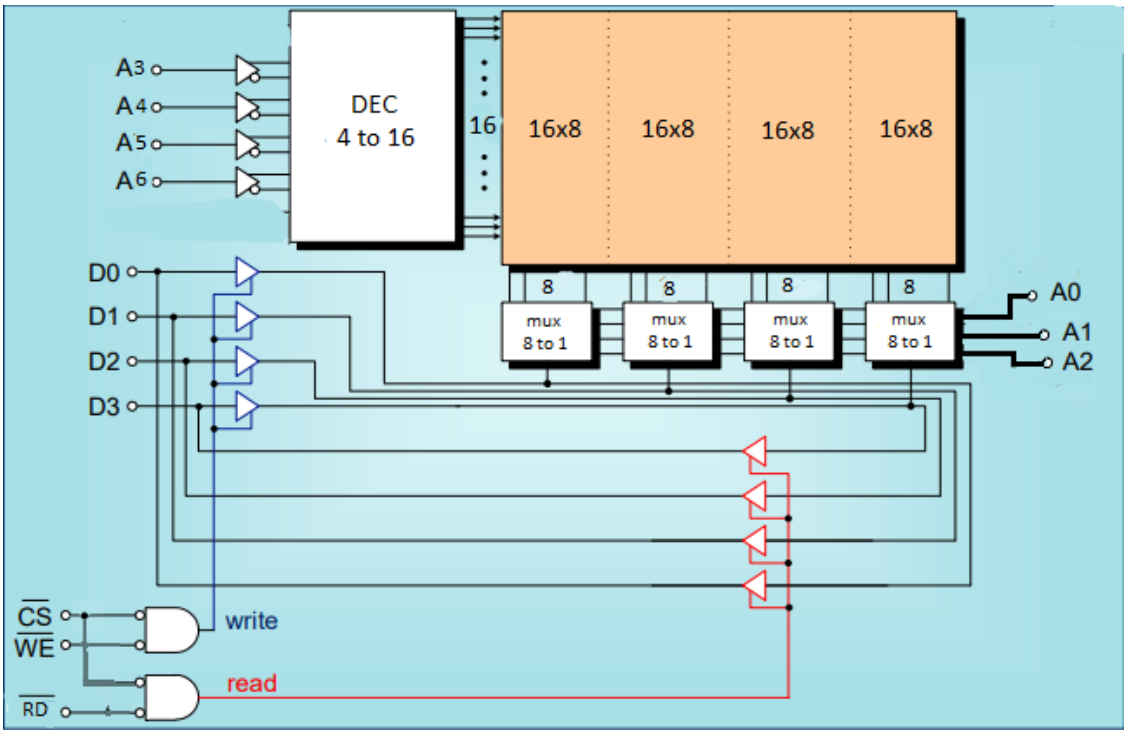
Επιλέγουμε η μνήμη να αποτελείται από 4 υποτμήματα που το καθένα έχει 4 γραμμές και 8 στήλες, συνολικά δηλαδή έχουμε μία διάταξη 16x32.

Τα σήματα A3, A4, A5, A6 χρησιμοποιούνται για την επιλογή της γραμμής της θέσης μνήμης μέσω ενός αποκωδικοποιητή 4 σε 16. Τα σήματα A0, A1, A2 χρησιμοποιούνται ως κοινοί επιλογείς τεσσάρων πολυπλεκτών 8 σε 1 για την επιλογή μίας από τις 8 στήλες από κάθε ένα από τα 4 υποτμήματα της μνήμης. Ετσι, μέσω των σημάτων A0 έως και A7 προσδιορίζεται μία μοναδική θέση (τετράδα από bits).

Τα υπόλοιπα στοιχεία του κυκλώματος χρησιμοποιούνται για να προσδιορίσουμε εάν πρόκειται για ανάγνωση από την μνήμη ή εγγραφή προς αυτήν.Πιο συγκεκριμένα, στην περίπτωση που τα σήματα CS (chip select) και WE (write enable) ισούνται ταυτόχρονα με 1, γίνεται εγγραφή λέξης στη μνήμη. Τότε ενεργοποιείται η αριστερή τετράδα των buffers του κυκλώματος και η "πορεία" της πληροφορίας γίνεται από τα αριστερά προς τα δεξιά, δηλαδή κάθε bit από τα D0 έως D3 φορτώνεται στην υποδεικνυόμενη θέση μνήμης από τους Ai επιλογείς της.

Ομοίως, στην περίπτωση που τα σήματα CS και RD (read) ισούνται ταυτόχρονα με 1, γίνεται ανάγνωση λέξης από τη μνήμη. Ενεργοποιείται η δεξιά τετράδα των buffers και έτσι η λέξη που περιέχεται στη θέση μνήμης που ορίζεται από τους επιλογείς Ai μεταφέρεται στις υποδοχές D0, D1, D2, D3 ("πορεία" της πληροφορίας από τα δεξιά προς τα αριστερά).

Θεωρούμε ότι τα σήματα CS, WE, RD δεν γίνονται ταυτόχρονα ίσα με 1, αφού δεν γίνεται να γράψουμε και να διαβάσουμε ταυτόχρονα από τη μνήμη.



ΑΣΚΗΣΗ 6η

Στη συγκεκριμένη άσκηση μας ζητείται η κατασκευή ενός συστήματος μνήμης για μΥ σύστημα του μΕ 8085. Το σύστημα αυτό θα περιέχει μνήμη ROM 6kByte και μνήμη RAM 10kByte.

Η μνήμη ROM θέλουμε να ξεκινάει από τη θέση 0000H κι αμέσως μετά να βρίσκεται ο χώρος για τη μνήμη RAM ,έτσι θα χρησιμοποιήσουμε πρώτα τα ολοκληρωμένα της ROM κι έπειτα της RAM. Επιλέγουμε αυθαίρετα να χρησιμοποιήσουμε πρώτα τα chip μεγαλύτερης χωρητικότητας κι έπειτα αυτά μικρότερης, χωρίς να επηρεάζεται η λειτουργία του συστήματος μας αν κάναμε αντίθετη επιλογή.

Το πρώτο chip έχει διαστάσεις 4k x 8bit = 2^{12} x 8bits άρα χρειαζόμαστε 12 bits διεύθυνσης για τον προσδιορισμό του τμήματος της σελίδας της ROM που θέλουμε να προσπελάσουμε. Ο χάρτης μνήμης για το συγκεκριμένο chip είναι ο εξής:

ROM 1(4Kx8bit)				
Αρχή Διευθύνσεων				
BIN	0000	0000	0000	0000
HEX	0	0	0	0
Τέλος Διευθύνσεων				
BIN	0000	1111	1111	1111
HEX	0	F	F	F

Το επόμενο chip ROM που χρησιμοποιούμε έχει διαστάσεις 2K = 2^{11} x 8bits , άρα χρειαζόμαστε 11 bit διεύθυνσης για να προσδιοριστεί το τμήμα της σελίδας που θέλουμε από αυτό .έτσι ο χάρτης μνήμης αυτή τη φορά διαμορφώνεται ως εξής.

ROM 2(2Kx8bit)				
Αρχή Διευθύνσεων				
BIN	0001	0000	0000	0000
HEX	1	0	0	0
Τέλος Διευθύνσεων				
BIN	0001	0111	1111	1111
HEX	1	7	F	F

Στη συνέχεια τοποθετούμε τα chip της μνήμης RAM .Το 1ο εξ' αυτών έχει διαστάσεις 8K = 2^{13} x 8bit ενώ το 2ο έχει διαστάσεις 2K = 2^{11} x8bit δηλαδή χρειάζονται 13 και 11 bit αντίστοιχα για την παράσταση μίας διεύθυνσής τους. Οι χάρτες μνήμης για τα 2 chip της RAM είναι:

SRAM 1(8Kx8bit)				
Αρχή Διευθύνσεων				
BIN	0001	1000	0000	0000
HEX	1	8	0	0
Τέλος Διευθύνσεων				
BIN	0011	0111	1111	1111
HEX	3	7	F	F

SRAM 2(2Kx8bit)				
Αρχή Διευθύνσεων				
BIN	0011	1000	0000	0000
HEX	3	8	0	0
Τέλος Διευθύνσεων				
BIN	0011	1111	1111	1111
HEX	3	F	F	F

Από τα παραπάνω προκύπτει ο συνολικός χάρτης μνήμης , που θα είναι ο παρακάτω.

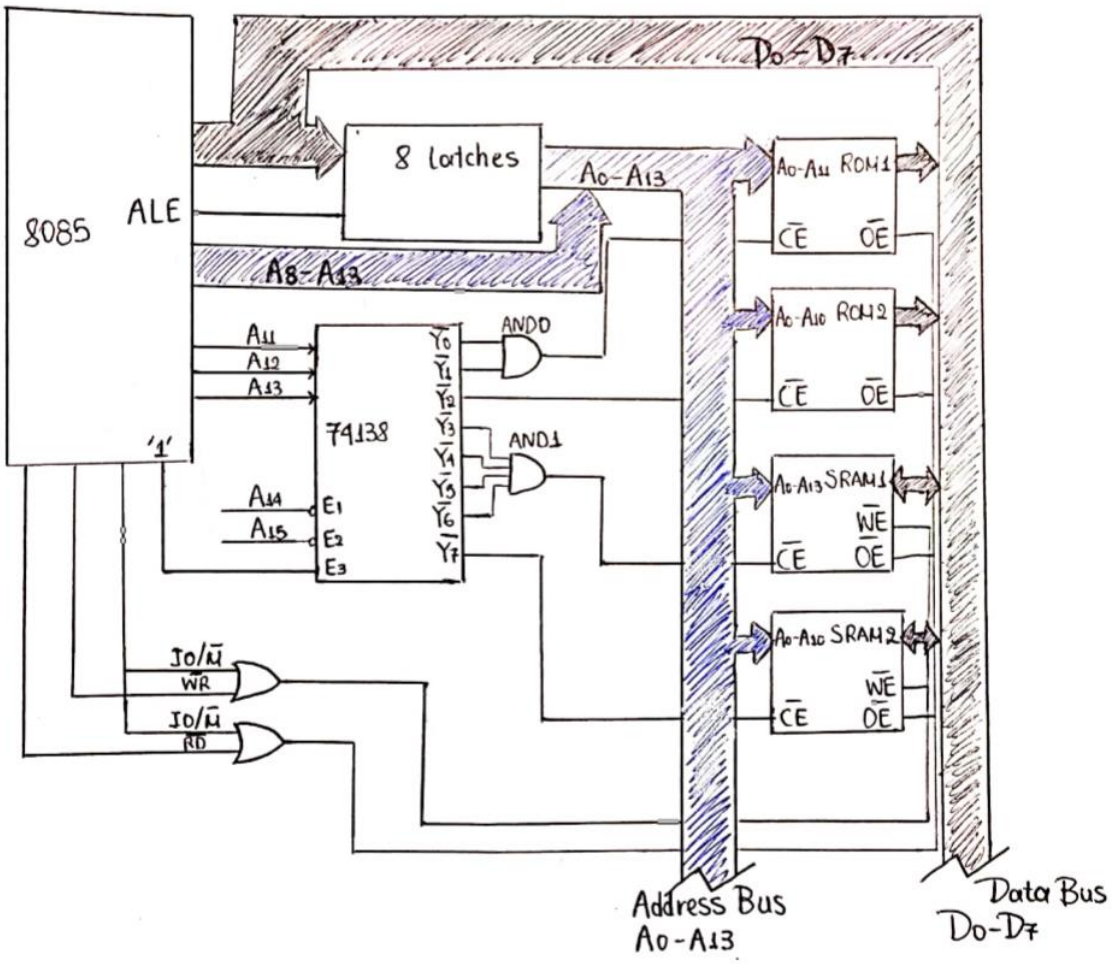
ROM 1	0000	0000	0000	0000
	0000	1111	1111	1111
ROM2	0001	0000	0000	0000
	0001	0111	1111	1111
SRAM1	0001	1000	0000	0000
	0011	0111	1111	1111
SRAM2	0011	1000	0000	0000
	0011	1111	1111	1111

Από τον παραπάνω πίνακα παρατηρούμε ότι με τη χρήση των 5 MSB μπορούμε να προσδιορίζουμε το επιθυμητό chip. Ακόμη βλέπουμε ότι τα ψηφία που είναι σημειωμένα με μπλε είναι πάντα 0 (A15-A14).Άρα διαφοροποίηση υπάρχει μόνο στα A11-A12-A13. Έτσι, δικαιολογείται η χρήση ενός αποκωδικοποιητή 3 σε 8 74LS138 ο οποίος έχει 3 εισόδους ενεργοποίησης (E1-E3) και τις 3 εισόδους(A11-A13) που δίνουν τις 8 εξόδους . Στις 2 εισόδους ενεργοποίησης θα θέσουμε τα bit A15-A14 αφού περάσουν από έναν αντιστροφέα ,ώστε η μνήμη να μην είναι προσπελάσιμη στην περίπτωση που τα MSB είναι 11,01 ή 00 και αφού τότε η επίτρεψη του αποκωδικοποιητή είναι στο λογικό 0 λόγω των αντιστροφέων. Για να επιλέξουμε ποια έξοδος του αποκωδικοποιητή ενεργοποιεί το αντίστοιχο chip σχεδιάζουμε τον παρακάτω πίνακα.

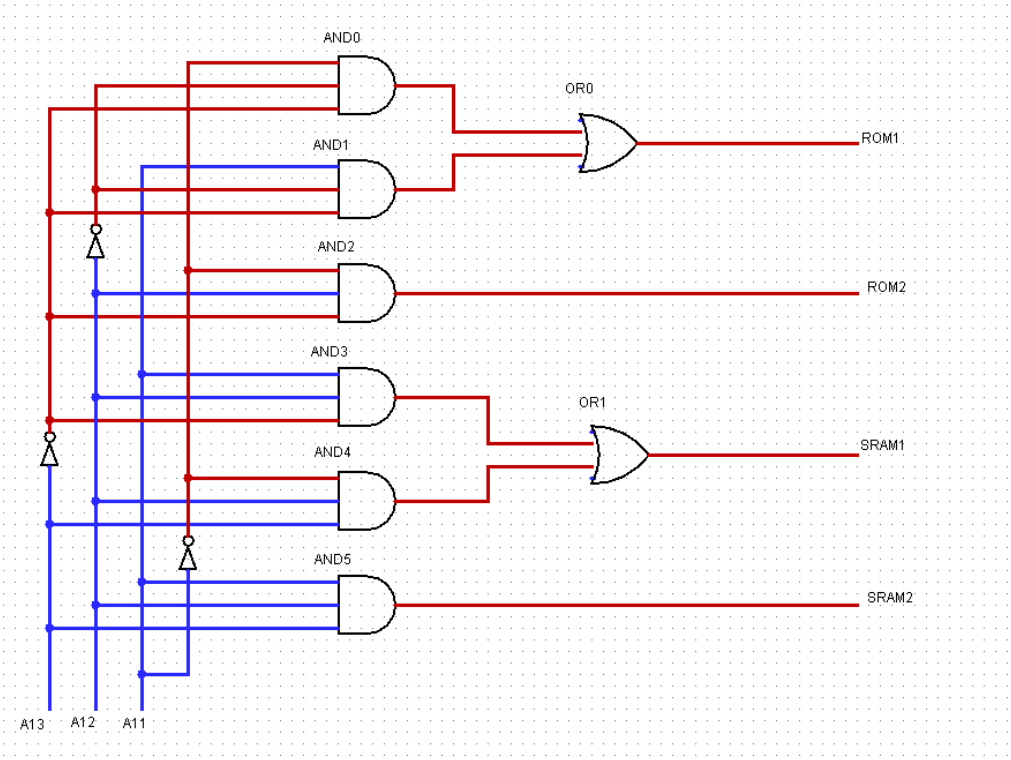
A_{13}	A_{12}	A_{11}	ROM1	ROM2	SRAM1	SRAM2
0	0	0	**1**	0	0	0
0	0	1	**1**	0	0	0
0	1	0	0	**1**	0	0
0	1	1	0	0	**1**	0
1	0	0	0	0	**1**	0
1	0	1	0	0	**1**	0
1	1	0	0	0	**1**	0
1	1	1	0	0	0	**1**

Όταν 2 ή παραπάνω 3άδες που δημιουργούνται από τα bit A13-A12-A11 ενεργοποιούν το ίδιο chip ,τότε εισάγονται σε μία AND της οποίας η έξοδος οδηγείται στο chip που θέλουμε να ενεργοποιηθεί. Αυτό γίνεται διότι ο αποκωδικοποιητής έχει συμπληρωματικές εξόδους και στο CS' πρέπει να εισαχθεί 0 για να ενεργοποιηθεί. Ακόμη σύμφωνα με τον πίνακα με τις αρχικές και τελικές διευθύνσεις στο ADDRESS BUS πρέπει να κυκλοφορούν τα bits A0-A13 για να μπορούν να προσδιοριστούν πλήρως κόλες οι λέξεις που υπάρχουν στη μνήμη (γιατί παραδείγματος χάρη βλέπουμε ότι στην SRAM1 έχουμε διαφορά στα 14 LSB από την αρχική στην τελική διεύθυνση. Έτσι η τελική διάταξη έχει ως εξής:

(a) Με αποκωδικοποιητή 3 σε 8 74LS138 και λογικές πύλες



(b) Μόνο με λογικές πύλες όπου θεωρούμε ότι η διάταξη παραμένει ίδια εκτός από το κομμάτι του αποκωδικοποιητή 3 σε 8 και των λογικών πυλών που αντικαθίστανται μόνο από λογικές πύλες σε διάταξη που φαίνεται παρακάτω.



ΑΣΚΗΣΗ 7η

Στην άσκηση αυτή μας ζητείται να σχεδιάσουμε ένα μΥ-Σ 8085 με τον εξής χάρτη μνήμης:

0000-0FFF Hex : ROM (4Kbytes)

1000-3FFF Hex : RAM (12Kbytes)

4000-6FFF Hex : ROM (12Kbytes)

7000 Hex : θύρα εισόδου (Memory map I/O)

70 Hex : θύρα εξόδου (Standard I/O)

Φτιάχνουμε τον χάρτη μνήμης του συστήματος και καταλήγουμε στο συμπέρασμα ότι κάθε σελίδα μνήμης θα έχει μέγεθος 4Kbytes.

Ο χάρτης είναι ο εξής:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	address	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	ROM1
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	00FF	4 Kbytes
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000	RAM2
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF	12 Kbytes
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000	
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	
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3FFF	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000	ROM2
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFF	12 Kbytes
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000	
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	
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	Θύρα εισόδου

Τα bits βάση των οποίων διακρίνουμε τις διαδοχικές σελίδες μνήμης είναι τα 4 MSB. Ως επιτρέψεις του πολυπλέκτη επιλέγουμε το MSB-A15 (δύο φορές) αφού η τιμή του είναι μηδενική για όλες τις σελίδες και το λογικό ‘1’. Οι τρεις είσοδοι του πολυπλέκτη θα είναι τα bits A12, A13, A14 μέσω των οποίων διακρίνουμε τις διαδοχικές σελίδες. Παρατηρούμε ότι οι δύο ROM των 4 και 12 Kbytes του χάρτη μνήμης θα συνδυαστούν στην ROM των 16 Kbytes που δίνεται και η οποία θα αποτελείται από τις σελίδες μνήμης 0,4,5,6. Η RAM των 12 Kbytes του χάρτη θα χωριστεί σε δύο μνήμες RAM , μία των 4 Kbytes, η οποία θα αποτελείται από την σελίδα 2 και μία των 8 Kbytes, η οποία θα αποτελείται από τις σελίδες 5,6.
Επομένως, συνδέουμε τις αντίστοιχες εξόδους του πολυπλέκτη με τα CS’ των μνημών. Το address bus θα “μεταφέρει” τα bits A0-A14, ενώ το data bus θα “μεταφέρει” τα bits AD0-AD7.
Στην διεύθυνση 7000H θα τοποθετηθεί η θύρα εισόδου. Η αντίστοιχη έξοδος του πολυπλέκτη , καθώς και τα σήματα RD και IO/M’ (RD=1, IO/M’=0), θα οδηγηθούν στον καταχωρητή 74LS240 που θα χρησιμοποιηθεί για τη ρύθμιση της εισόδου (Memory Map I/O).
Η έξοδος των δεδομένων θα πραγματοποιηθεί μέσω του flip-flop 74LS373, το ρολόι του οποίου θα συνδεθεί κατάλληλα με τα σήματα WR, IO/M’ (WR=1, IO/M’=1) και τη διεύθυνση 70H (Standard I/O).

Το κύκλωμα άρα αποκτά την εξής μορφή: