Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Συστήματα Μικροϋπολογιστών

Ακαδημαϊκό έτος 2019-2020

Αναφορά 2^{ης} Σειράς Ασκήσεων

Στοιχεία σπουδαστών

Όνομα: Μαντζούτας Ανδρέας Α.Μ. : 03117108

Όνομα: Τσιτσής Αντώνιος Α.Μ. : 03117045

Εξάμηνο: 6^{ο}

Ημερομηνία: 12/05/2020

1η Άσκηση

```
IN 10H
LXI H,0900H  ;ARXIKH ADDRESS 0900
MVI A,FFH    ;O A PERIEXEI TO 255

LOOP1:
      MOV M,A      ;APOTHIKEUSI TOY ARITHMOU STON HL
      DCR A        ;NEXT NUMBER
      INX H        ;NEXT ADDRESS
      MOV M,A      ;HL)=NEXT NUMBER
      CPI 00H
      JNZ LOOP1    ;OSO DIAFORO TO 0, SUNEXIZE NA BAZEIS ARITHMOUS

;B ERWTHMA
      LXI D,0008H  ;ARIKOPOISI TOU DE 0008H, EPEIDI TO 0 EXEI 8 MIDENIKA
      MVI A,00H
      MVI B,00H
      MVI C,00H

NEXT_NUM:
      MOV A,B
      MVI C,00H    ;KRATA TO PLHTHOS TWN BIT POU CHECKARISTHKAN GIA TO AN
EINAI 1 H 0
      INR A        ;NEXT NUMBER
      MOV B,A      ;SAVE A
      CPI FFH      ;IF NUMBER = 255 THEN FINISH
      JNZ CHECKBITS       ;ELSE CHECK THE NUMBER
      JZ THIRD     ;TELOS B ERWTHMATOS


CHECKBITS:
      INR C        ;C = CURRENT BIT
      MOV A,C
      CPI 09H      ;IF C =09H, NUM IS OVER
      MOV A,B
      JNZ FINDZEROS       ;ELSE CONTINUE
      JZ NEXT_NUM

FINDZEROS:
      RAR          ;CY = LSB
      JC CHECKBITS ;IF CY = 1 CHECK NEXT BIT
      INX D        ;ELSE WE HAVE ZERO
      JNC CHECKBITS       ;CHECK THE NEXT BIT


THIRD:
      MVI C,01H
      MVI B,20H    ;PLHTHOS ARITHMWN

CALCULATE:
      INR B        ;NEXT NUMBER
      INR C        ;PLHTHOS=PLHTHOS+1
      MOV A,B
      CPI 70H             ;IF B=70H FINISH
      JNZ CALCULATE       ;ELSE CONTINUE CALCULATING
      RST 1
END
```

Ο κώδικας επισυνάπτεται με όνομα ask1.8085

2η Άσκηση

Για την επίτευξη του ζητούμενου, χρησιμοποιήσαμε την ρουτίνα DELB, βάζοντας στον διπλό καταχωρητή BC την τιμή 0FFF H, ενώ χρησιμοποιήσαμε και τον καταχωρητή E ως μετρητή «επαναλήψεων». Έπειτα από μερικές δοκιμές, ορίσαμε την τιμή του E σε 12 H και κάθε μια φορά που αναβοσβήνουν τα λαμπάκια, μειώνουμε την τιμή του κατά 1. Να σημειωθεί ότι τη μπάρα καθυστέρησης την αφήσαμε στην αρχική της θέση.

Παραθέτουμε τον κώδικα της άσκησης, ο οποίος και επισυνάπτεται με όνομα ask2.8085.

```
LXI B,0FFFH                ;FOR DELB

BEGIN:

       LDA 2000H           ;INPUT
       RAR                 ; LSB = CY
       JC BEGIN            ;IF LSB=1, SWITCH UP, KEEP READING UNTIL SWITCH
DOWN

LSB0:

       LDA 2000H           ;READ INPUT
       RAR
       JNC LSB0      ;READ INPUT UNTIL SWITCH IS UP


LSB1:

       LDA 2000H
       RAR
       JC LSB1       ;WE HAD OFF-ON. READ UNTIL WE HAVE OFF AGAIN


LEDS_ON:              ;OFF-ON-OFF
       MVI D,00H             ;(D)=0 --> LSB IS DOWN
       MVI E,12H             ;COUNTER FOR TIME



DELAY:
       MVI A,00H
       STA 3000H             ;00000000, ALL LEDS OPEN
       CALL DELB
       MVI A,FFH             ;11111111, ALL LEDS CLOSE
       STA 3000H
       CALL DELB
       DCR E
```

```
       MOV A,E
       CPI 00H
       JZ LEDS_OFF          ;IF E =0 TIME IS OVER AND LEDS HAVE TO GO OFF
       MOV A,D
       CPI 00H
       JZ CHECK             ;IF D=0, CHECK IF THERE WAS ANY CHANGE ON THE
SWITCH
       CPI 01H
       JZ CHECKIFDOWN       ;IF D=1, SWITCH IS ON,
                            ;CHECK IF IT WENT DOWN AGAIN TO REFRESH


LEDS_OFF:

       MVI A,FFH
       STA 3000H            ;ALL LEDS OFF
       JMP BEGIN


CHECK:

       LDA 2000H
       RAR
       JNC DELAY
       INR D                ;IF LSB=1, SWITCH WENT ON AGAIN -> D=1

       JMP DELAY


CHECKIFDOWN:

       LDA 2000H
       RAR
       JC DELAY             ;IF LSB=1, NO CHANGES
       JMP LEDS_ON          ;ELSE LSB=0 AND MUST REFRESH TIME AND D

END
```

3η Άσκηση

i)

```
EXERCISE_1:
START:
       LDA 2000H ;READ INPUT
       RAL ;CHECK BIT7
       JC TURN_1 ;IF BIT7 = 1 THEN TURN ON 1 LED
       RAL ;CHECK BIT6
       JC TURN_2 ;IF BIT6 = 1 THEN TURN ON 2 LEDS
       RAL ;CHECK BIT5
       JC TURN_3 ;IF BIT5 = 1 THEN TURN ON 3 LEDS
       RAL ;CHECK BIT4
       JC TURN_4 ;IF BIT4 = 1 THEN TURN ON 4 LEDS
       RAL ;CHECK BIT3
       JC TURN_5 ;IF BIT3 = 1 THEN TURN ON 5 LEDS
       RAL ;CHECK BIT2
       JC TURN_6 ;IF BIT2 = 1 THEN TURN ON 6 LEDS
       RAL ;CHECK BIT1
       JC TURN_7 ;IF BIT1 = 1 THEN TURN ON 7 LEDS
       RAL ;CHECK BIT0
       JC TURN_8 ;IF BIT0 = 1 THEN TURN ON 8 LEDS
       MVI A,FFH
       STA 3000H
       JMP START

TURN_8:
       MVI A,00H
       STA 3000H
       JMP START
TURN_7:
       MVI A,01H
       STA 3000H
       JMP START
TURN_6:
       MVI A,03H
       STA 3000H
       JMP START
TURN_5:
       MVI A,07H
       STA 3000H
       JMP START
TURN_4:
       MVI A,0FH
       STA 3000H
       JMP START
TURN_3:
       MVI A,1FH
       STA 3000H
       JMP START
TURN_2:
       MVI A,3FH
       STA 3000H
       JMP START
TURN_1:
       MVI A,7FH
       STA 3000H
       JMP START
END
```

ii)

```
START:
       CALL KIND
       CMP H
       JZ START
       MOV H,A             ;SWZW THN TIMH GIA NA STAMATHSW AN DEN ALLA3EI
       LXI B,0244H ;
       MVI E,00H
       JMP DECISION

DECISION:
       CPI 01H             ;
       JZ TURN_1           ;
       CPI 02H             ;
       JZ TURN_1           ;
       CPI 03H             ;
       JZ TURN_1           ;
       CPI 04H             ;
       JZ TURN_1           ;AN MIKROTERO H ISO TOU TESSERA TOTE PRWTA 4
       CPI 05H             ;
       JZ TURN_2           ;
       CPI 06H             ;
       JZ TURN_2           ;
       CPI 07H             ;
       JZ TURN_2           ;
       CPI 08H             ;
       JZ TURN_2           ;AN APO 5 MEXRI 8 TOTE TELEYTAIA 4
       MVI A,FFH
       STA 3000H
       JMP START

TURN_2:
       MOV C,E
       INR C
       MOV E,C             ;METRHTHS 4 FORWN ANABOSBHSHS
       MOV A,C
       CPI 05H
       JZ START
       MVI A,0FH
       STA 3000H
       LXI B,0A00H
       CALL DELB
       MVI A,FFH
       STA 3000H
       CALL DELB
       JMP TURN_2

TURN_1:
       MOV C,E
       INR C
       MOV E,C             ;METRHTHS 4 FORWN ANABOSBHSHS
       MOV A,C
       CPI 05H
       JZ START
       MVI A,F0H
       STA 3000H
       LXI B,0A00H
       CALL DELB
       MVI A,FFH
       STA 3000H
       CALL DELB
       JMP TURN_1

END
```

iii)

```
       IN 10H

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


       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


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
```

Οι ασκήσεις επισυνάπτονται με ονόματα ask3a.8085, ask3b.8085 και ask3c.8085 αντίστοιχα.

4η Άσκηση

Η γενική ιδέα πίσω από την άσκηση ήταν να ταυτίζουμε κάθε φορά το LSB με το CY, κάνοντας rotate right και ανάλογα με το αν είναι 0 ή 1, πηγαίναμε στο κατάλληλο μπλοκ, έως ότου έχουμε ελέγξει και τα 8 bit(ή στην περίπτωση της αριστερής AND είναι 0 το πρώτο bit, οπότε και δεν ελέγχουμε το 8ο). Είχαμε αρχικοποιήσει τον καταχωρητή B, ώστε να περιέχει τον αριθμό που θα πρέπει να εκτυπωθεί στο τέλος, και ανάλογα με το αποτέλεσμα κάθε πύλης τον αυξάναμε ή τον αφήναμε σταθερό.

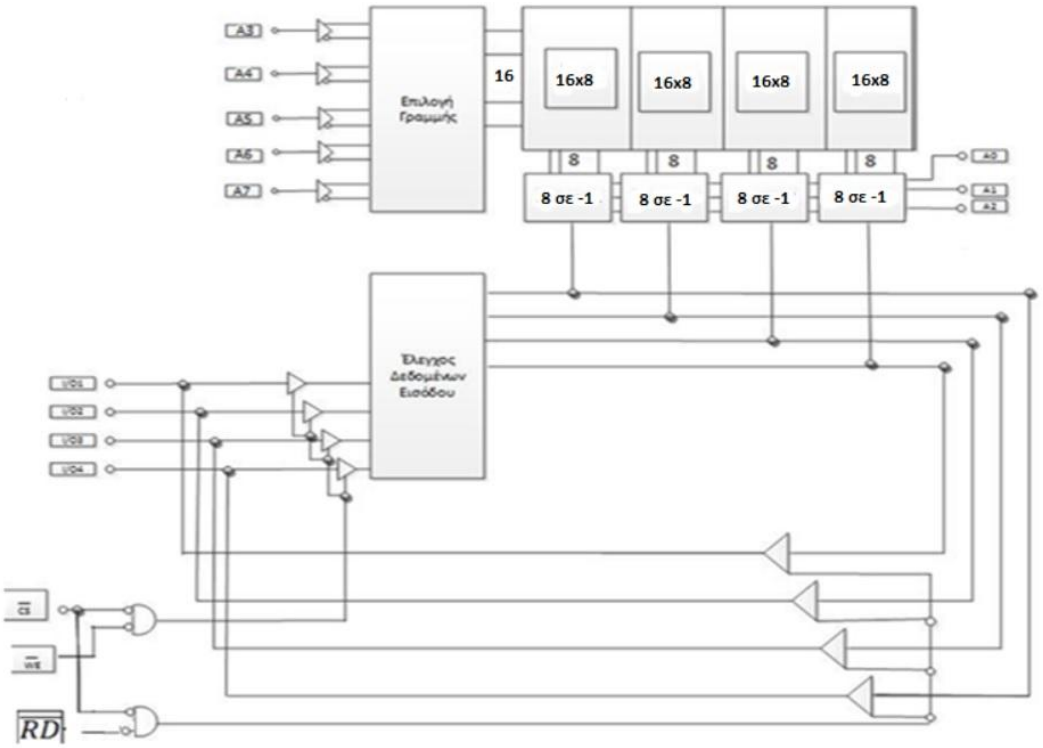
Παραθέτουμε τον κώδικα, ο οποίος και επισυνάπτεται με όνομα ask4.8085.

```
BEGIN:
       LDA 2000H
       MVI B,00H
       MVI C,00H
       MVI D,00H
       STC
       CMC                 ;CY = 0
       RAR                 ;LSB -> CY
       JC OR1_1            ;IF LSB = 1 -> OR GATE = 1
       RAR
       JC OR1_1            ;IF CY = 1 -> 1ST OR GATE = 1
       JNC OR1_0           ;ELSE IST 0R GATE = 0

OR1_1:
       INR B
       RAR                 ;IF 1ST BIT IS 1, OR RESULT=1. AND WE
                           ;DONT CARE ABOUT 2ND BIT. RAR SO WE CAN
                           ;PROCEED TO THIRD BIT


OR1_0:
       MOV D,B             ;D HAS THE RESULT NOW
       MVI B,00H
       RAR                 ;GET 3RD BIT
       JNC AND1_0          ;AND GATE RESULT = 0
       RAR
       JNC AND1_0          ;AND GATE RESULT =0
       JC AND1_1           ;IF CY=1, BOTH BITS ARE 1 AND GATE RESULT = 1


AND1_1:
       INR B
       INR B               ;B = 2, SO IN BIN IT IS 10

AND1_0:
       MOV E,A             ;SAVE CONTENT OF A
       MOV A,D             ;MOVE RESULT TO A
       ADD B               ;ADD B TO RESULT
       MOV D,A             ;MOVE RESULT TO D
       MOV A,E             ;RESTORE A
       MVI B,00H
       RAR                 ;BIT 5
       JC OR2_1            ;IF CY = 1 -> 2ND OR IS 1
       RAR
       JC OR2_1
       JNC OR2_0


OR2_1:
       INR B
       INR B
       INR B
```

```
       INR B
       INR C               ;IF 1, C =1. WILL USE FOR XOR
       RAR

OR2_0:
       MOV E,A
       MOV A,D
       ADD B
       MOV D,A
       MOV A,E
       MVI B,00H
       RAR                 ;BIT 7
       JNC AND2_0          ;IF CY =0 -> AND = 0
       RAR
       JC AND2_1           ;IF CY=1 -> AND 1
       JNC AND2_0

AND2_1:
       MOV A,C
       CPI 01H             ;AND=1. IF C =1 THEN XOR = 0
       JZ XOR0
       JNZ XOR1

AND2_0:
       MOV A,C
       CPI 01H             ;AND =0. IF C = 0, THEN XOR =0. ELSE 1
       JNZ XOR0
       JZ XOR1

XOR1:
       MVI A,00H
       INR A
       INR A
       INR A
       INR A
       INR A
       INR A
       INR A
       INR A
       ADD D
       JMP FINISH

XOR0:
       MVI A,00H
       MOV A,D

FINISH:
       CMA
       STA 3000H
       RST 1
END
```

5η Άσκηση

Για την παρουσίαση της εσωτερικής οργάνωσης μιας μνήμης SRAM 128x4 bit, βασιστήκαμε πάνω στην απεικόνιση των διαφανειών. Το x4 υποδηλώνει ότι η μνήμη χωρίζεται σε 4 μέρη, και το 128 την χωρητικότητα της κάθε μνήμης. Στην περίπτωση μας, λοιπόν, θα χρησιμοποιήσουμε 4 μνήμες χωρητικότητας 16x8 η κάθε μία, και 4 πολυπλέκτες 8 σε 1. Επομένως, η εσωτερική οργάνωση μιας τέτοιας μνήμης είναι η ακόλουθη:



Όσον αφορά τη λειτουργία, κατά την ανάγνωση και την εγγραφή έχουμε:

- **Ανάγνωση**: Οι είσοδοι RD και CS είναι 0, ενώ η είσοδος WE είναι 1. Αυτό έχει ως αποτέλεσμα το αποτέλεσμα της πάνω πύλης AND να είναι 0, ενώ της κάτω 1. Έτσι, εμποδίζεται το σήμα απ' το να καταγραφεί στην μνήμη, ενώ ταυτόχρονα επιτρέπεται η ανάγνωση.
- **Εγγραφή**: Στην περίπτωση της εγγραφής, οι είσοδοι CS και WE είναι 0, ενώ η είσοδος RD είναι 1. Έτσι, το αποτέλεσμα της κάτω AND είναι 0, και της πάνω 1. Με αυτόν τον τρόπο ενεργοποιούνται οι απομονωτές που επιτρέπουν τη διέλευση προς τις γραμμές I/O, ενώ αποτρέπεται η εγγραφή.

6η Άσκηση

Αφού ζητείται η κατασκευή ενός συστήματος μνήμης για μΥ σύστημα του μΕ 8085, και το σύστημα αυτό πρέπει να περιέχει μνήμη ROM 6kByte και μνήμη RAM 10 kByte, ο χώρος τη μνήμης θα χωριστεί σε δύο ROM (ROM 1 2Kx8bits και ROM 2 4Kxbits) και δύο SRAM (SRAM 1 2Kxbits και SRAM 2 8Kx8bits).
Αφού τις κατατάξουμε με την σειρά που προαναφέρθηκαν και εφόσον η αρχή διευθήνσεων της μίας αρχίζει μετά το τέλος διευθύνσεων της προηγούμενης (με αρχή διευθύνσεων της ROM το 0000H) θα έχουμε τον ακόλουθο χάρτη μνήμης σε δυαδικό :

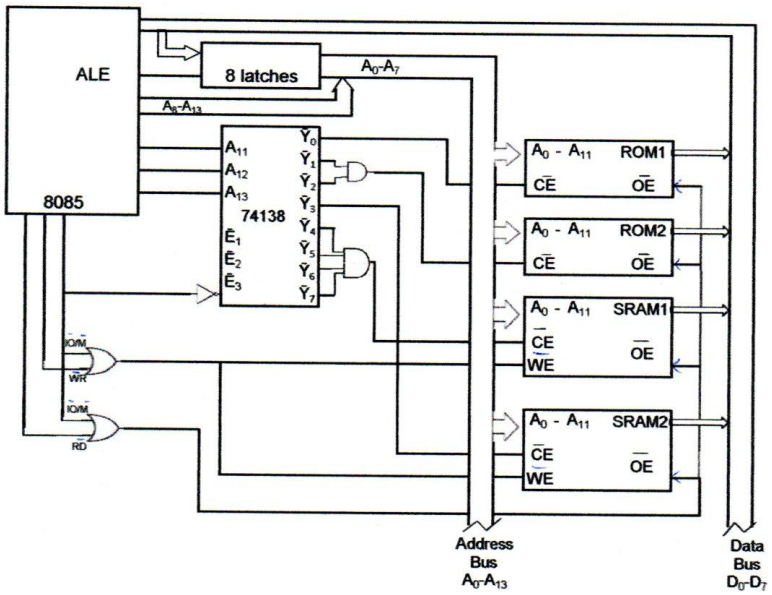
Χάρτης Μνήμης					
ROM 1 (2Kx8bits)	Αρχή διευθύνσεων	0000	0000	0000	0000
	Τέλος διευθύνσεων	0000	0111	1111	1111
ROM 2 (4Kxbits)	Αρχή διευθύνσεων	0000	1000	0000	0000
	Τέλος διευθύνσεων	0001	0111	1111	1111
SRAM 1 (2Kxbits)	Αρχή διευθύνσεων	0001	1000	0000	0000
	Τέλος διευθύνσεων	0001	1111	1111	1111
SRAM 2 (8Kx8bits)	Αρχή διευθύνσεων	0010	0000	0000	0000
	Τέλος διευθύνσεων	0011	1111	1111	1111

Για να για να ελέγξουμε την εκάστοτε επιλογή των μνημών παρατηρούμε οτι τα A13-A11 είναι αυτά που αλλάζουν, και συνεπώς παράγουμε τον ακόλουθο χάρτη “Karnaugh”.

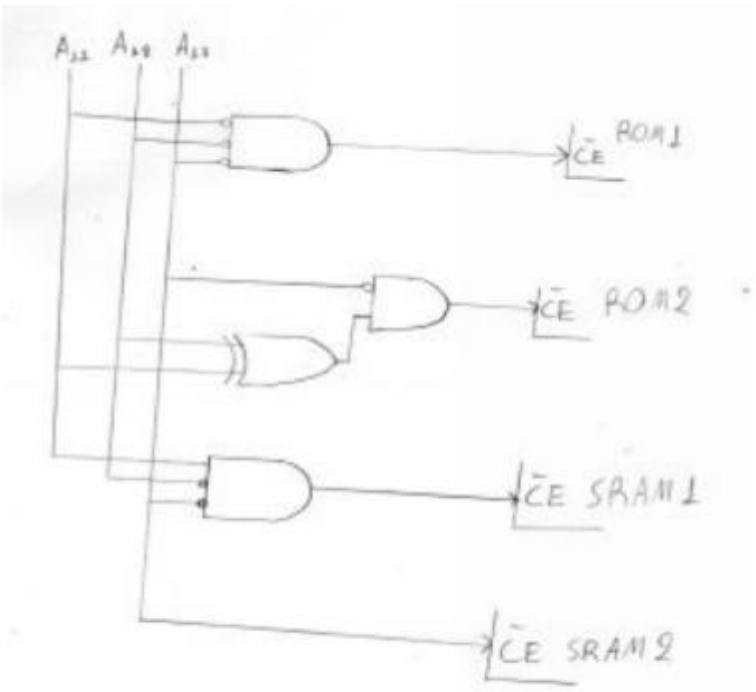
A13	A11 A12	0	1
0	0	ROM 1	ROM 2
0	1	ROM 2	SRAM 1
1	0	SRAM 2	SRAM 2
1	1	SRAM 2	SRAM 2

Με βάση τον παραπάνω πίνακα, που ουσιαστικά είναι ο πίνακας αποκωδικοποίησης, λαμβάνουμε την ακόλουθη αναπαράσταση της μνήμης:

α) Με αποκωδικοποιητή και λογικές πύλες (OR και AND) :



β) Για την αναπαράστασή του συστήματος μνήμης χωρίς αποκωδικοποιητή, αρκεί να βάλουμε στη θέση του αποκωδικοποιητή το παρακάτω λογικό κύκλωμα:



7η Άσκηση

Και σε αυτήν την άσκηση εργαστήκαμε με παρόμοιο τρόπο με την προηγούμενη και προέκυψε ο παρακάτω χάρτης μνήμης:

ROM 0 16Kbytes				
Αρχη διευθύνσεων				
Binary	0000	0000	0000	0000
Hexadecimal	0	0	0	0
Τέλος διευθύνσεων				
Binary	0000	1111	1111	1111
Hexadecimal	0	F	F	F

RAM 1 4Kbytes				
Αρχη διευθύνσεων				
Binary	0001	0000	0000	0000
Hexadecimal	1	0	0	0
Τέλος διευθύνσεων				
Binary	0001	1111	1111	1111
Hexadecimal	1	F	F	F

RAM 2 4Kbytes				
Αρχη διευθύνσεων				
Binary	0010	0000	0000	0000
Hexadecimal	2	0	0	0
Τέλος διευθύνσεων				
Binary	0010	1111	1111	1111
Hexadecimal	2	F	F	F

RAM 3 4Kbytes				
Αρχη διευθύνσεων				
Binary	0011	0000	0000	0000
Hexadecimal	3	0	0	0
Τέλος διευθύνσεων				
Binary	0011	1111	1111	1111
Hexadecimal	3	F	F	F

ROM 1 16Kbytes				
Αρχη διευθύνσεων				
Binary	0100	0000	0000	0000
Hexadecimal	4	0	0	0
Τέλος διευθύνσεων				
Binary	0100	1111	1111	1111

ROM 2 16Kbytes				
Αρχη διευθύνσεων				
Binary	101	0000	0000	0000
Hexadecimal	5	0	0	0
Τέλος διευθύνσεων				
Binary	0101	1111	1111	1111
Hexadecimal	5	F	F	F

ROM 3 16Kbytes				
Αρχη διευθύνσεων				
Binary	110	0000	0000	0000
Hexadecimal	6	0	0	0
Τέλος διευθύνσεων				
Binary	0110	1111	1111	1111
Hexadecimal	6	F	F	F

Έπειτα, επειδή έχουμε συγκεκριμένες μνήμες στη διάθεσή μας, «ενώσαμε» τις ROM1, ROM2 και ROM0 σε μία ROM με μέγεθος 16 kByte, ενώ χρησιμοποιήσαμε 2 RAM, μία 4kBytes και μία 8 kBytes.

Επομένως, λαμβάνοντας υπόψιν και την είσοδο-έξοδο, το σύστημα μνήμης που προκύπτει είναι το ακόλουθο: