2η Σειρά Ασκήσεων
Συστήματα Μικροϋπολογιστών

Βερέμης Αλέξανδρος 03115063
Σπυρίδων Κρητικός 03116133

1η Άσκηση:

```
PART_A:
            IN 10H              ;DISABLE MEMORY PROTECTION
            LXI H,0900H         ;START OF MEMORY STORAGE
            MVI B,00H           ;SET INITIAL VALUES TO B
            MVI A,00H           ;AND A

NUMS_IN_MEM:
            MOV M,B             ;B HAS THE APPROPRIATE NUMBER TO BE
                                ;STORED
            INR A               ;IF A == 0, THEN 256 HAVE BEEN
            CPI 00H             ;ASSIGNED TO MEMORY AND PART_A IS
            JZ PART_B           ;FINISHED, PART_B STARTS
            INX H               ;ELSE "POINT" TO THE NEXT MEMORY
                                ;STORAGE POSITION
            INR B               ;B += 1, NEXT APPROPRIATE NUMBER
            JMP NUMS_IN_MEM

PART_B:
            LXI H,0900H
            MVI B,00H
            MVI C,00H
            MVI D,08H           ;D TELLS HOW MANY BITS HAVE BEEN
                                ;CHECKED UNTIL NOW
COUNT_ONES:
            MOV A,M             ;READ NUMBER FROM MEMORY
CHECK_BIT:
            MOV E,A             ;E IS USED AS A TEMPORARY VARIABLE
            MOV A,D             ;IF ALL BITS OF CURRENT NUMBER
            CPI 00H             ;HAVE BEEN CHECKED THEN
            JZ CHECK_NEXT       ;CHECK NEXT NUMBER
            MOV A,E             ;ELSE
            RAL                 ;CHECK NEXT BIT THROUGH CARRY
            DCR D               ;ONE LESS BIT TO CHECK
            JNC CHECK_BIT       ;IF CURRENT BIT != 1 CHECK NEXT
            INX B               ;ELSE, BC += 1, INCREASE NUMBER OF
                                ;ONES
            JMP CHECK_BIT
CHECK_NEXT:
            INX H               ;"POINT" TO THE NEXT MEM POSITION
            MOV A,H             ;CHECK IF ALL NUMBERS HAVE BEEN
            CPI 0AH             ;READ
            JZ PART_C           ;IF THEY HAVE, PART_C STARTS
            MVI D,08H           ;ELSE CHECK NEXT NUMBER
            JMP COUNT_ONES

PART_C:
            LXI H,0900H
            MVI D,00H

IN_RANGE:
            MOV A,M             ;READ NUMBER FROM MEMORY
            CPI 10H             ;IS LESS THAN 10H?
```

```
            JC LOOK_NEXT        ;YES, CHECK NEXT NUMBER
            CPI 60H             ;IS GREATER THAN 60H?
            JNC LOOK_NEXT       ;YES, CHECK NEXT NUMBER
            INR D               ;NO, D += 1, INCREASE NUMBER OF
                                ;DESIRED NUMBERS
LOOK_NEXT:
            INX H               ;"POINT" TO THE NEXT NUMBER TO LOOK
            MOV A,H             ;CHECK IF ALL NUMBERS HAVE BEEN
            CPI 0AH             ;READ
            JZ END_PROGRAM              ;IF THEY HAVE, THE PROGRAM ENDS
            JMP IN_RANGE        ;ELSE CONTINUES LOOP

END_PROGRAM:
            END
```

2η Άσκηση:

```
      MOV D,FFH
      MOV A,D
      STA 3000H           ;Initially, lights are off
      MVI E,96H           ;E = 150 (counter for 30sec)
      LXI B,00C8H         ;BC = 200ms
INIT:
      CALL GET_MSB
      CPI 00H             ;Wait for first OFF
      JZ OFF1
      CALL DELB           ;If it's not yet OFF,wait until it comes
      JMP INIT

OFF1:
      CALL GET_MSB
      CPI 80H                   ;Wait for first ON after OFF
      JZ ON1
      CALL DELB
      JMP OFF1

ON1:
      CALL GET_MSB
      CPI 00H             ;Wait for second OFF
      JZ OFF2
      CALL DELB
      JMP ON1

OFF2:
      CALL GET_MSB
      CPI 01H
      JZ ON2              ;Wait for second ON
      CALL TOGGLE         ;If not second ON, toggle lights
      CALL DELB
      DCR E               ;Decrease timer by 200ms
      MOV A,E
      CPI 00H             ;If 30 secs have not passed
      JNZ OFF2            ;Repeat
      CALL TURN_OFF       ;Else turn off the lights
      JMP OFF1            ;And go to OFF1 routine

ON2:
      CALL GET_MSB
      CPI 00H             ;Wait for third OFF
      JZ OFF3             ;On third OFF, go to OFF3
      CALL TOGGLE         ;Else toggle lights
      CALL DELB           ;delay routine
```

```
       DCR E               ;Same as OFF2
       MOV A,E
       CPI FFH
       JNZ ON2
       CALL TURN_OFF
       JMP OFF1

OFF3:
       MVI E,96H           ;Reset counter to 30sec
       JMP OFF2            ;Go to OFF2 Again

TOGGLE:
       MOV A,D             ;Toggle lights
       CMA                 ;Αντιστροφή των ψηφίων για να αναψουν όλα τα φώτα
       MOV D,A
       STA 3000H
       RET

TURN_OFF:
       MVI A,FFH           ;Turn off lights
       STA 3000H
       RET

GET_MSB:
       LDA 2000H           ;Get MSB from dip switches
       ANI 80H
       RET
```

3η Άσκηση:

```
EXERCISE_I:

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 ;ALL DIGITS HAVE TO BE ZERO IN ORDER TO TURN ON ALL THE LEDS
              STA 3000H
              JMP START

TURN_7:
              MVI A,01H
```

```
              STA 3000H
              JMP START
TURN_6:
              MVI A,03H ;THE DIGITS THAT ARE NOT ZERO DO NOT TURN ON THE EQUIVALENT
LEDS
              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

EXERCISE_II:

START:

       CALL KIND           ;READ KEYBOARD
       CPI 01H             ;
       JZ TURN_1           ;TURN ON LED 1
       CPI 02H             ;
       JZ TURN_2           ;TURN ON LED 2
       CPI 03H             ;
       JZ TURN_3           ;TURN ON LED 3
       CPI 04H             ;
       JZ TURN_4           ;TURN ON LED 4
       CPI 05H             ;
       JZ TURN_5           ;TURN ON LED 5
       CPI 06H             ;
       JZ TURN_6           ;TURN ON LED 6
       CPI 07H             ;
       JZ TURN_7           ;TURN ON LED 7
       CPI 08H             ;
       JZ TURN_8           ;TURN ON LED 8
       MVI A,FFH
       STA 3000H
       JMP START

TURN_1:
       MVI A,FEH
       STA 3000H
       JMP START

TURN_2:
       MVI A,FDH
       STA 3000H
```

```
       JMP START
TURN_3:
       MVI A,FBH
       STA 3000H
       JMP START
TURN_4:
       MVI A,F7H
       STA 3000H
       JMP START
TURN_5:
       MVI A,EFH
       STA 3000H
       JMP START
TURN_6:
       MVI A,DFH
       STA 3000H
       JMP START
TURN_7:
       MVI A,BFH
       STA 3000H
       JMP START
TURN_8:
       MVI A,7FH
       STA 3000H
       JMP START

       END


EXECISE_III:

       IN 10H              ;DISABLE MEMORY PROTECTION

       LXI H,0A02H         ;"POINT" TO MEMORY 0A02H

       MVI M,10H           ;STORE NOTHING TO "PRINT"
       INX H               ;POINT TO THE NEXT MEMORY CELL 0A03H
       MVI M,10H           ;STORE NOTHING TO "PRINT"
       INX H               ;POINT TO THE NEXT MEMORY CELL 0A04H
       MVI M,10H           ;STORE NOTHING TO "PRINT"
       INX H               ;POINT TO THE NEXT MEMORY CELL 0A05H
       MVI M,10H           ;STORE NOTHING TO "PRINT"
START:

LINE_0:                    ;LINE INSTR STEP, FETCH PC, HRDWR STEP
       MVI A,FEH
       STA 2800H           ;READ LINE
       LDA 1800H           ;READ COLUMNS
       ANI 07H             ;SET THE 5 MSB'S = 0
       MVI C,86H
       CPI 06H             ;INSTR STEP
       JZ DISPLAY
       MVI C,85H
       CPI 05H             ;FETCH PC
       JZ DISPLAY
       MVI C,F7H
       CPI 03H             ;HRDWR STEP
       JZ DISPLAY

LINE_1:                    ;LINE RUN, FETCH REG, FETCH ADRS
       MVI A,FDH
       STA 2800H           ;READ LINE
       LDA 1800H           ;READ COLUMNS
```

```
       ANI 07H             ;SET THE 5 MSB'S = 0
       MVI C,84H
       CPI 06H             ;RUN
       JZ DISPLAY
       MVI C,80H
       CPI 05H             ;FETCH REG
       JZ DISPLAY
       MVI C,82H
       CPI 03H             ;FETCH ADRS
       JZ DISPLAY

LINE_2:                    ;LINE 0, STORE/INCR, DECR
       MVI A,FBH
       STA 2800H           ;READ LINE
       LDA 1800H           ;READ COLUMNS
       ANI 07H             ;SET THE 5 MSB'S = 0
       MVI C,00H
       CPI 06H             ;0
       JZ DISPLAY
       MVI C,83H
       CPI 05H             ;STORE/INCR
       JZ DISPLAY
       MVI C,81H
       CPI 03H             ;DECR
       JZ DISPLAY

LINE_3:                    ;LINE 1, 2, 3
       MVI A,F7H
       STA 2800H           ;READ LINE
       LDA 1800H           ;READ COLUMNS
       ANI 07H             ;SET THE 5 MSB'S = 0
       MVI C,01H
       CPI 06H             ;1
       JZ DISPLAY
       MVI C,02H
       CPI 05H             ;2
       JZ DISPLAY
       MVI C,03H
       CPI 03H             ;3
       JZ DISPLAY

LINE_4:                    ;LINE 4, 5, 6
       MVI A,EFH
       STA 2800H           ;READ LINE
       LDA 1800H           ;READ COLUMNS
       ANI 07H             ;SET THE 5 MSB'S = 0
       MVI C,04H
       CPI 06H             ;4
       JZ DISPLAY
       MVI C,05H
       CPI 05H             ;5
       JZ DISPLAY
       MVI C,06H
       CPI 03H             ;6
       JZ DISPLAY

LINE_5:                    ;LINE 7, 8, 9
       MVI A,DFH
       STA 2800H           ;READ LINE
       LDA 1800H           ;READ COLUMNS
       ANI 07H             ;SET THE 5 MSB'S = 0
       MVI C,07H
       CPI 06H             ;7
```

```
       JZ DISPLAY
       MVI C,08H
       CPI 05H             ;8
       JZ DISPLAY
       MVI C,09H
       CPI 03H             ;9
       JZ DISPLAY

LINE_6:                    ;LINE A, B, C
       MVI A,BFH
       STA 2800H           ;READ LINE
       LDA 1800H           ;READ COLUMNS
       ANI 07H             ;SET THE 5 MSB'S = 0
       MVI C,0AH
       CPI 06H             ;A
       JZ DISPLAY
       MVI C,0BH
       CPI 05H             ;B
       JZ DISPLAY
       MVI C,0CH
       CPI 03H             ;C
       JZ DISPLAY

LINE_7:                    ;LINE D, E, F
       MVI A,7FH
       STA 2800H           ;READ LINE
       LDA 1800H           ;READ COLUMNS
       ANI 07H             ;SET THE 5 MSB'S = 0
       MVI C,0DH
       CPI 06H             ;D
       JZ DISPLAY
       MVI C,0EH
       CPI 05H             ;E
       JZ DISPLAY
       MVI C,0FH
       CPI 03H             ;F
       JZ DISPLAY

       JMP START           ;IF NO BUTTON IS PRESSED, CHECK AGAIN

DISPLAY:

       LXI H,0A00H

       MOV A,C             ;( C) := BUTTON'S CODE
       ANI 0FH             ;ISOLATE THE 4 LSB'S
       MOV M,A             ;STORE THEM TO MEMORY
       INX H               ;POINT TO THE NEXT MEMORY CELL 0A01H
       MOV A,C             ;( C) := BUTTON'S CODE
       ANI F0H             ;ISOLATE THE 4 MSB'S
       RRC                 ;SHIFT THEM TO THE 4 LSB'S
       RRC
       RRC
       RRC
       MOV M,A             ;STORE THEM TO MEMORY

       LXI D,0A00H         ;MEMORY ADDRESS WHERE THE MESSAGE IS
       CALL STDM
       CALL DCD

       JMP START

       END
```

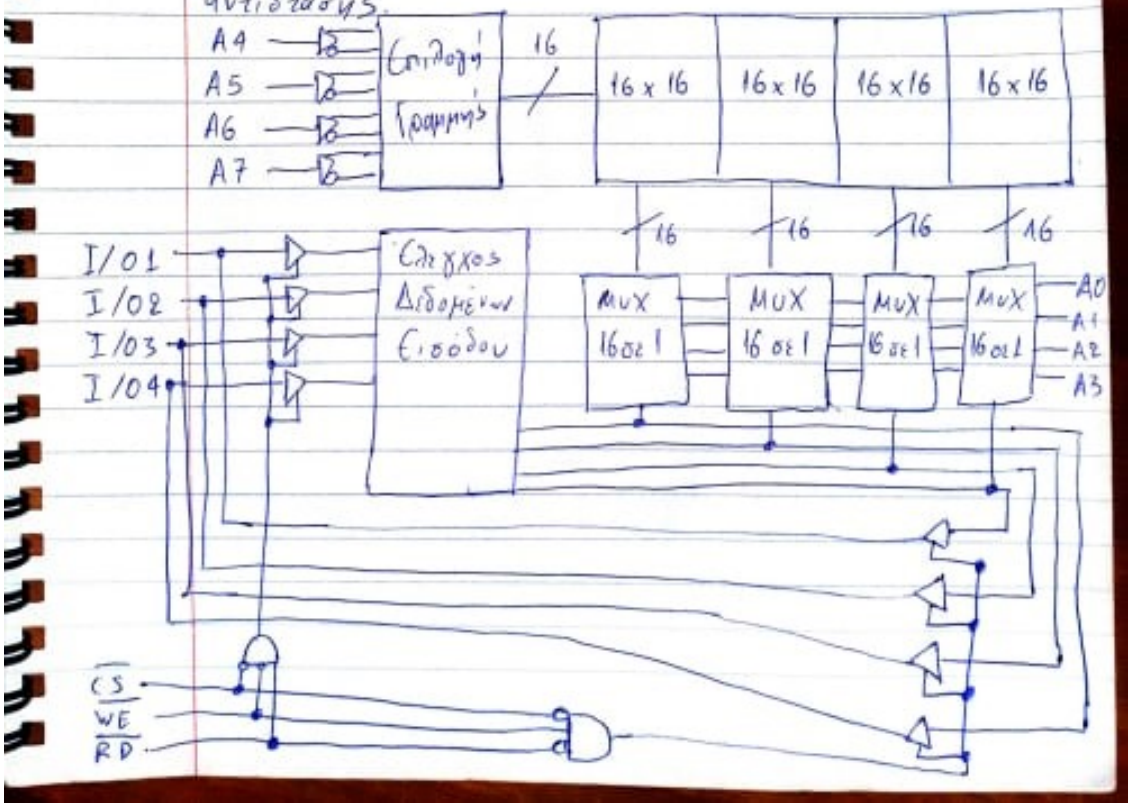
4η Άσκηση:

```
MVI C,00H    ; THE FIRST MS LEDS ALWAYS OFF THAT'S WHY
             ; WE LOAD 00H and not F0H


;Calculate X0_a
       ;Load B0
       LDA 2000H
       ANI 01H             ;Mask for 1st LSB digit
       CALL GET_DIGIT
       MOV B,A
       ;Load A0
       LDA 2000H
       ANI 02H
       CALL GET_DIGIT
       ;X0 = A0 | B0
       ORA B
       ;Store X0_a for next operation
       MOV D,A

;Calculate X1
       ;Load B1
       LDA 2000H
       ANI 04H
       CALL GET_DIGIT
       MOV B,A
       ;Load A1
       LDA 2000H
       ANI 08H
       CALL GET_DIGIT
       ;X1 = A1 | B1
       ORA B
;Calculate X0
       XRA D
       ;Store this bit in C
       ORA C
       MOV C,A
       ;X1 = A1 | B1
       ORA B
       ;Shift and store
       RLC
       ORA C
       MOV C,A



;Calculate X2
       ;Load B2
       LDA 2000H
       ANI 10H
       CALL GET_DIGIT
```

```
       MOV B,A
       ;Load A2
       LDA 2000H
       ANI 20H
       CALL GET_DIGIT
       ;X2 = A2 & B2
       ANA B
       ;Shift and store
       RLC
       RLC
       ORA C
       MOV C,A

;Calculate X3
       ;Load B3
       LDA 2000H
       ANI 40H
       CALL GET_DIGIT
       MOV B,A
       ;Load A3
       LDA 2000H
       ANI 80H
       CALL GET_DIGIT
       ;X3 = A3 & B3
       ANA B
       ;Shift and store
       RLC
       RLC
       RLC
       ORA C
       MOV C,A

;Move C to LEDs in order to display the right turned on LEDs
       MOV A,C
;Complement A because LEDs are on when bit is 0
       CMA
       STA 3000H
       HLT



GET_DIGIT:
       JPE A0        ;If result had even parity, this
       JMP A1        ;means it was all zeros => digit was 0
A0:    MVI A,00H
       RET
A1:    MVI A,01H
       RET

       END
```

5η Άσκηση:

1. Παρακάτω φαίνεται η οργάνωση μιας SRAM 256 x 4 bit. Για τη διευθυνσιοδότηση αυτής της μνήμης θέλουμε 8 bit διεύθυνσης αφού $2^8 = 256$, δηλαδή A7 - A0. Για να έχουμε την καλύτερη δυνατή ισορροπία μεταξύ γραμμών και στηλών (τετράδων bit) επιλέγουμε οργάνωση που περιλαμβάνει ίσες γραμμές και στήλες, δηλαδή 16 γραμμές (16 x 16 = 256) και 16 τετράδες bit. Επίσης, για να μην γίνεται να ενεργοποιηθούν τόσο οι απομονωτές για την ανάγνωση, όσο και για τη διαγραφή (τα σήματα (RD)' και (WE)' καθορίζονται έτσι ώστε να μην ενεργοποιούνται ταυτόχρονα, οπότε δεν είναι εύκολο να συμβεί το παραπάνω) αν και τα δύο σήματα γίνουν 0, τότε όλοι οι απομονωτές μεταβαίνουν σε κατάσταση υψηλής αντίστασης.



Έστω ότι θέλουμε να διαβάσουμε τα δεδομένα που βρίσκονται στην τετράδα με διεύθυνση 0010 0110. Από τη διεύθυνση αυτή των 8 bit τα 4 πρώτα μας δίνουν τη γραμμή όπου βρίσκονται, δηλαδή τη γραμμή 0010= 2, η οποία γραμμή επιλέγεται από τον επιλογέα γραμμών. Τα υπόλοιπα 4 bit μας δείχνουν ότι κάθε bit της τετράδας βρίσκεται στη στήλη 0110 = 6 του καθενός από τα 4 κελιά από στήλες του ενός bit. Με βάση αυτά τα bit A3-A0, στην έξοδο των πολυπλεκτών βρίσκονται τα δεδομένα από τη στήλη αυτή, σχηματίζοντας έτσι την τετράδα που θέλουμε. Επειδή έχουμε (CS)' = 0, (WE)' = 1 και (RD)' = 0, η αριστερή συστοιχία απομονωτών οδηγείται σε κατάσταση υψηλής αντίστασης εξόδου λόγω απενεργοποίησης της αριστερής πύλης AND, ενώ η δεξιά πύλη και η δεξιά συστοιχία απομονωτών ενεργοποιούνται και συνδέεται η έξοδος των πολυπλεκτών (δηλαδή η τετράδα που διαβάζουμε) με το δίαυλο δεδομένων.

Ομοίως, αν θέλουμε να κάνουμε εγγραφή σε αυτή τη θέση μνήμης, η επιλογή γραμμής και στήλης γίνεται με τον ίδιο τρόπο, με τους πολυπλέκτες να συνδέουν την έξοδό τους με την κατάλληλη θέση μνήμης. Επειδή τώρα έχουμε (WE)' = 0 και (RD)' = 1, η αριστερή συστοιχία πολυπλεκτών είναι αυτή που ενεργοποιείται, οπότε ο δίαυλος δεδομένων, ο οποίος περιέχει την τετράδα προς εγγραφή, συνδέεται μέσω του κυκλώματος με τα κελιά μνήμης που αντιστοιχούν στην επιθυμητή τετράδα, οπότε και ολοκληρώνεται η εγγραφή.

6η Άσκηση:

Η ROM ξεκινά από τη διεύθυνση 0000H και υλοποιείται χρησιμοποιώντας 2 ολοκληρωμένα μνήμης 4K8 bit (ROM) και 1 ολοκληρωμένο μνήμης 2K8 bit (ROM) -συνολικά 3 ICs. Η μνήμη RAM καταλαμβάνει 6 KB και ακολουθεί χωρίς κενά διευθύνσεων την ROM, ενώ αποτελείται απο μια μνήμη 2K8 και μια 4K8 SRAMs (2 ICs).

Σύμφωνα με τις προδιαγραφές που μας δίνονται, οι μνήμες μαζί με τις διευθύνσεις που καταλαμβάνουν:

Μνήμη	Διευθύνσεις
ROM 1 (4K)	0000H - 0FFFH
ROM 2 (4K)	1000H - 1FFFH
ROM 3 (2K)	2000H - 27FFH
SRAM 1 (2K)	2800H - 2FFFH
SRAM 2 (4K)	3000H - 3FFFH

Τα I.C. των 2Kbyte δέχονται ως είσοδο τα bit διεύθυνσης A10-A0 (11 bits), ενώ τα άλλα I.C. (4KB) δέχονται τα A11-A0 (12 bits). Για τον χάρτη μνήμης του συγκεκριμένου συστήματος εργαζόμαστε όπως παρακάτω:

Για τη $1_η$ ROM ξεκινάμε από τη διεύθυνση 0000 0000 0000 0000 και καταλήγουμε στη διεύθυνση 0000 1111 1111 1111 (μεταβάλλονται τα bits A0-A11)

A15	A14	**A13**	**A12**	**A11**	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0
0	0	**0**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1

Για τη $2^η$ ROM ξεκινάμε από τη διεύθυνση 0001 0000 0000 0000 και καταλήγουμε στη διεύθυνση 0001 1111 1111 1111 (μεταβάλλονται τα bits A0-A11)

A15	A14	**A13**	**A12**	**A11**	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0
0	0	**0**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1

Για τη $3^η$ ROM ξεκινάμε από τη διεύθυνση 0010 0000 0000 0000 και καταλήγουμε στη διεύθυνση 0010 1111 1111 1111 (μεταβάλλονται τα bits A0-A10)

A15	A14	**A13**	**A12**	**A11**	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0
0	0	**1**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1

Για τη $1^η$ RAM ξεκινάμε από τη διεύθυνση 0010 1000 0000 0000 και καταλήγουμε στη διεύθυνση 0010 1111 1111 1111 (μεταβάλλονται τα bits A0-A10)

A15	A14	**A13**	**A12**	**A11**	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	**1**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0
0	0	**1**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1

Για τη $2^η$ RAM ξεκινάμε από τη διεύθυνση 0011 0000 0000 0000 και καταλήγουμε στη διεύθυνση 0011 1111 1111 1111 (μεταβάλλονται τα bits A0-A11)

A15	A14	**A13**	**A12**	**A11**	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0
0	0	**1**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1

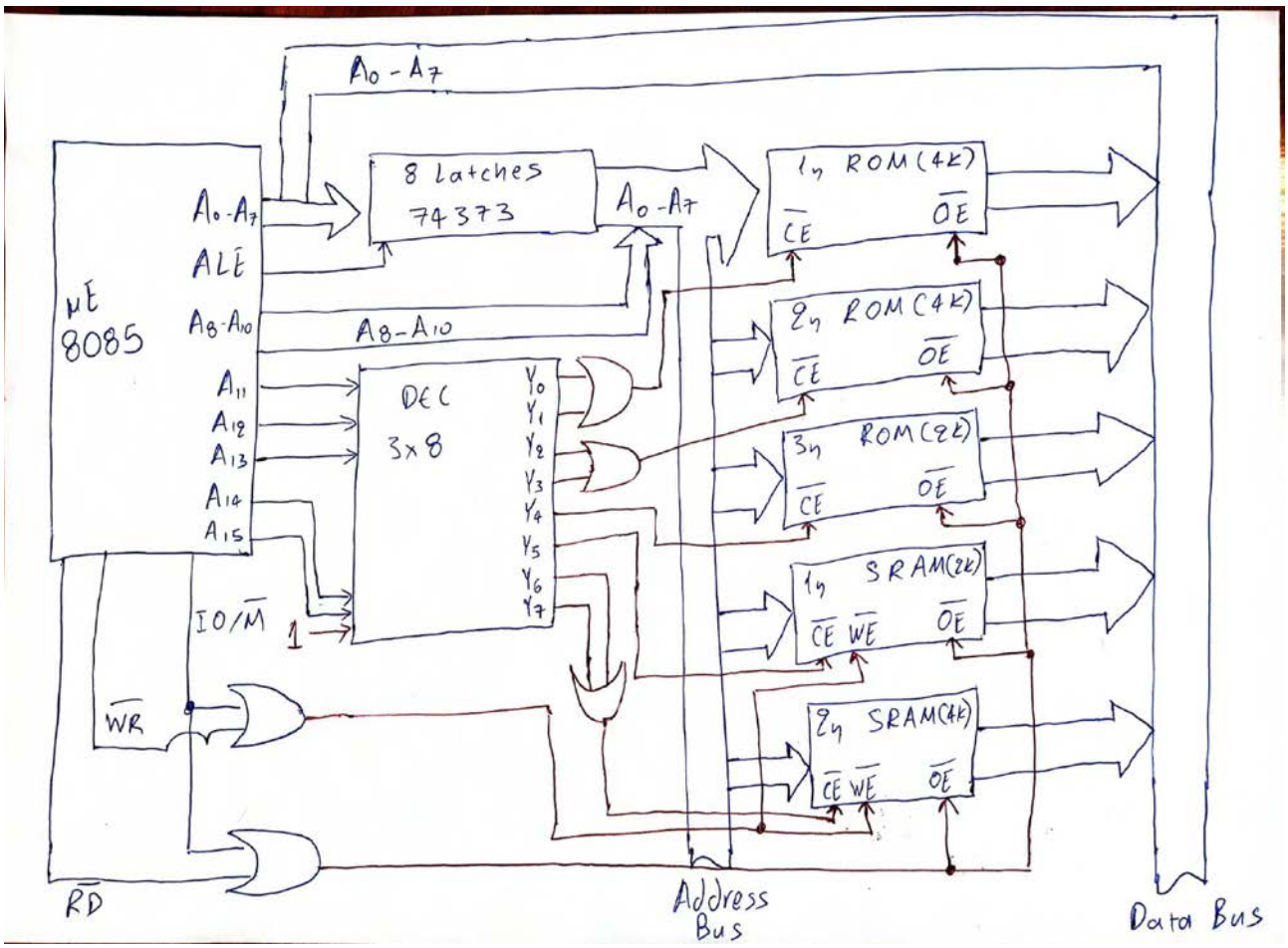
Από τα παραπάνω συμπεραίνουμε ότι τα μοναδικά bits που δεν μεταβάλλονται σε καμία περίπτωση είναι τα A15, A14. Επομένως για τη δημιουργία του αποκωδικοποιητή θα αναστρέψουμε τις εισόδους των bits αυτών (που θα έχουν πάντα τιμή μηδέν) ώστε να έχουν ρόλο εισόδου επίτρεψης και φυσικά θα έχουμε επιπλέον την καθιερωμένη είσοδο επίτρεψης στο λογικό 1. **Άρα, Τα bits A14,A15 χρησιμοποιούνται ως επίτρεψη στον αποκωδικοποιητή.**
Επίσης προκύπτει ότι υπεύθυνα για την ενεργοποίηση της κατάλληλης μνήμης είναι τα bits A13, A12 και A11.

ROM 1 : A13’ A12’
ROM 2 : A13’ A12
ROM 3 : A13 A12’ A11’
SRAM 1 : A13 A12’ A11
SRAM 2 : A13 A12

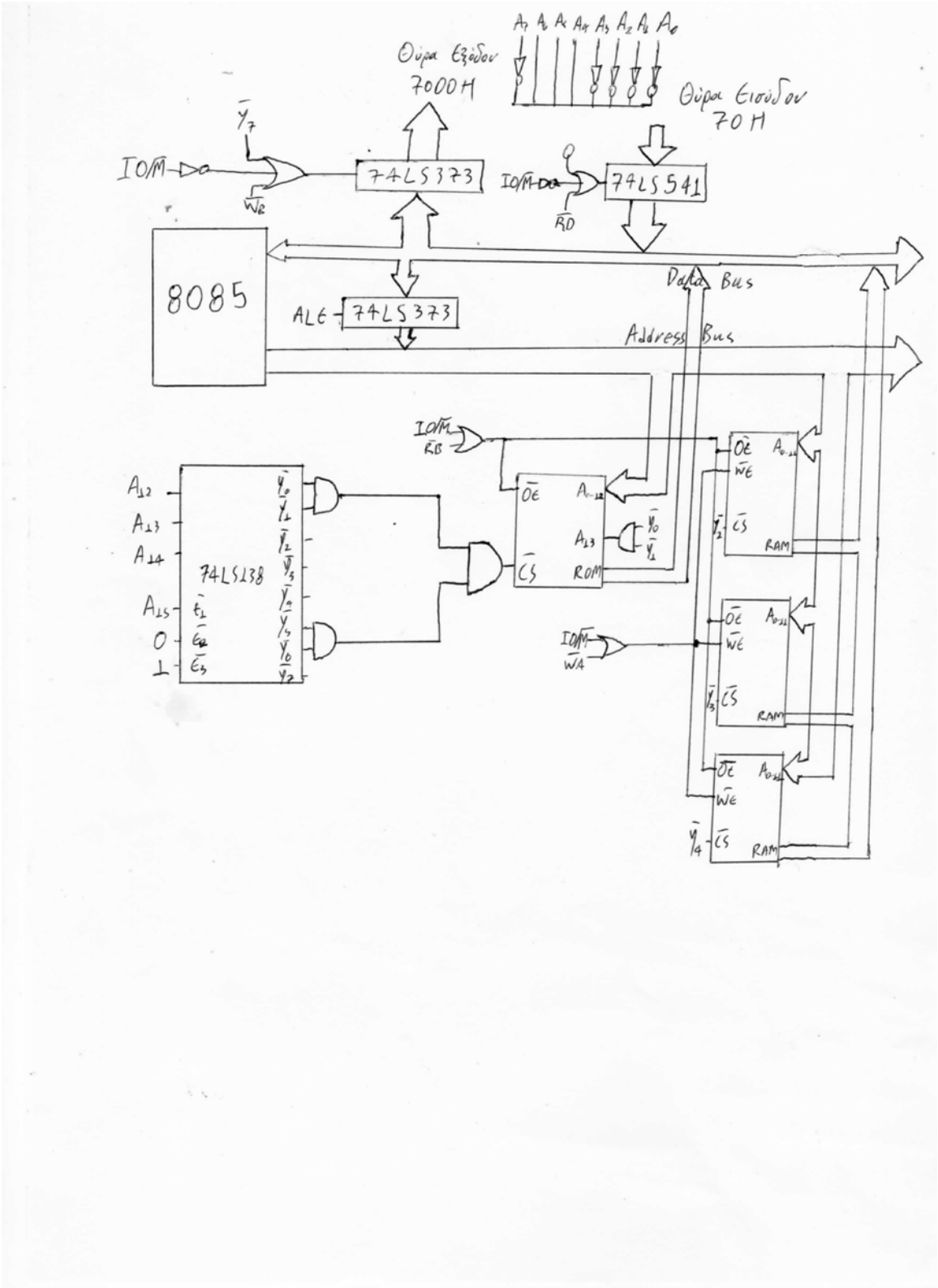
Οι συνδυασμοί που μπορούμε να έχουμε και οι αντίστοιχες έξοδοι του αποκωδικοποιητή βρίσκονται στον παρακάτω πίνακα:

A13	A12	A11	ΕΞΟΔΟΣ	ΜΝΗΜΗ
0	0	0	Y0	ROM1
0	0	1	Y1	ROM1
0	1	0	Y2	ROM2
0	1	1	Y3	ROM2
1	0	0	Y4	ROM3
1	0	1	Y5	SRAM1
1	1	0	Y6	SRAM2
1	1	1	Y7	SRAM2

Από τα συνολικά 64 KB της μνήμης δεν χρησιμοποιούνται τελικά τα 64 – 16 = 48 Kbytes από τις RAM και τις ROM.
Το σχέδιο του συστήματος μνήμης:



7η Άσκηση:


Θύρα Εξόδου
7000H
Θύρα Εισόδου
70H
74LS373
74LS541
8085
ALE
74LS373
Data Bus
Address Bus
74LS138
ROM
RAM
RAM
RAM