Συστήματα Μικροϋπολογιστών
1η Ομάδα Ασκήσεων
2022

Στεφανής Παναγιώτης (el19096)
Μιχάλης Τσιλιμιγκουνάκης (el19001)

1η Άσκηση

Στην άσκηση αυτή αποθηκεύουμε στην μνήμη, από την θέση 0900H και πάνω, τους αριθμούς από το 255 μέχρι το 0. Ενδιάμεσα ελέγχουμε πόσα μηδενικά έχει ο κάθε αριθμός κάνοντας 8 right shifts και αυξάνοντας τον counter των μηδενικών (καταχωρητής DE) ανάλογα.. Δύο ακόμα έλεγχοι κοιτάνε πότε ένας αριθμός είναι μεταξύ του 20 και του 70 (συμπεριλαμβάνοντας τα άκρα) και αυξάνουν τον καταχωρητή C ανάλογα.

```
IN 10H ;memory protection off
MVI A,FFH ; accu = 255
MVI C,00H ; 20-70 numbers counter
LXI D,0000H ; zeros counter
LXI H,0900H ; first number memory address
FOR:
        MVI B,09H ;b=9 to count the zeros of current number
        MOV M,A ; store number in memory

L1:     DCR B
        JZ L2 ; if we have iterated through every digit
        RRC ; else shift right
        JC L1 ; if digit = 1 go to L1 to check the next digit
        INX D ; else count +1 zeroz
        JMP L1 ; go to L1 to check next digit

L2:     CPI 71H ; check if number is <=70
        JNC L3 ; if no go to L3 to store next number
        CPI 20H ; else check if number is >=20
        JC L3 ; if no go to L3 to store next number
        INR C ; else increase counter for numbers between 20 and 70

L3:     DCR A ; decrease remaining numbers counter
        INX H ; point to next memory location
        CPI FFH ; check if we stored every number in range [0,255]
        JNZ FOR ; if no go to FOR
        RST 1
        END
```

2η Άσκηση

Το πρόγραμμα εδώ περιμένει το LSB των διακοπτών να γίνει on, ενώ πριν ήταν off και μετά το ανάποδο. Ενώ δηλαδή ήταν on να γίνει off. Έτσι κάνει branch και ξεκινάει να αναβοσβήνει τα LED. Αν ενδιάμεσα σε αυτή την λειτουργία έρθει πάλι διαδοχή off-on-off στο LSB των DIP, κάνουμε reset έναν καταχωρητή που μετράει πόσες φορές έχουν ανάψει και σβήσει τα LED.

```
 1         LXI B,00C8H     ;Load registers BC with 200 for 200ms delay
 2 START:
 3         LDA 2000H       ;Load DIP switches
 4         RRC             ;Right shift to check the LSB
 5         JNC LSBOFF      ;If LSB is off go to LSBOFF label
 6         JMP START       ;Else check again (loop)
 7 LSBOFF:
 8         LDA 2000H       ;load again
 9         RRC             ;right shift to get LSB
10         JC LSBON        ;if lsb is on, go to LSBON label
11         JMP LSBOFF      ;else check again
12 LSBON:
13         LDA 2000H       ;load DIP switches again
14         RRC             ;right shift to get LSB
15         MVI E,25H       ;E=37 to turn on and off LEDs ~15sec (2*37*200ms=~75*200ms)
16         JNC LED         ;If lsb is off turn on leds
17         JMP LSBON       ;else wait for the lsb to turn off
18 LED:
19         LDA 2000H       ;load DIP
20         RRC             ;get LSB
21         JC LED2         ;if LSB=1 go LED2 to check if we have to reset timer
22         MVI A,00H       ;else a=0 to turn on LEDs
23         STA 3000H       ;output leds
24         CALL DELB       ;add delay
25         MVI A,FFH       ;a=1 to turn off all leds
26         STA 3000H       ;output leds
27         CALL DELB       ;add delay
28         DCR E           ;decrease turn-on/off counter
29         JNZ LED         ;if timer!=0 go to LED to flash leds again
30         JMP START       ;else go to start to check DIP
31 LED2:
32         LDA 2000H       ;load DIPs
33         RRC             ;right shift to get LSB
34         JNC RESET_TIMER ;if LSB=0 go to RESET_TIMER to reset timer
35         MVI A,00H       ;A=00000000 to turn on LEDs
36         STA 3000H       ;output LEDs
37         CALL DELB       ;add delay
38         MVI A,FFH       ;A=11111111 to turn off all LEDs
39         STA 3000H       ;output LEDs
40         CALL DELB       ;add delay
41         DCR E           ;decrease flash times counter
42         JNZ LED2        ;if counter!=0 go to LED2 to flash again
43         JMP START       ;else go to START to check DIPs
44 RESET_TIMER:
45         MVI E,25H       ;reset timer
46         JMP LED         ;go to LED
47 FINISH:
48         END
```

3η Άσκηση

i) Το πρόγραμμα αυτό ανιχνεύει ποιο είναι το αριστερότερο ενεργό DIP switch, και στην συνέχεια ανάβει το αντίστοιχο LED και όλα τα αριστερά του. Αυτό το κάνει με χρήση μιας μάσκας, με την οποία σε κάθε επανάληψη κάνουμε λογικό AND με τον A, ώστε να σβήσουν τα δεξιά LED, που βρίσκονται δεξιότερα του ενεργού DIP switch.

```
START:  MVI B,08H       ;Initialize B with 8 decimal
        LDA 2000H       ;Load switch input

L1:     RLC             ;Left shift
        JC POSITION     ;If Carry is 1,go to find the position
        DCR B           ;Decrease B by 1
        JZ TURNOFF      ;If B is 0, we want all leds OFF
        JMP L1          ;Else jump to L1 (loop)

POSITION:
        MVI A,FEH       ;11111110 --> A
        MVI C,FEH       ;11111110 --> C , C is gonna be a mask
        DCR B           ;Decrease B by 1
        JZ TURNALL      ;If B is 0 we want all leds ON

CHECK:  DCR B           ;Loop here while B > 0
        JZ LED          ;if B=0 go to LED to turn on the right LEDs
        RLC             ;Right shift A
        ANA C           ;A logical AND C to turn off next LSB LED
        MOV C,A         ;C=A to refresh mask and use it again next time
        JMP CHECK       ;go to CHECK

LED:    CMA             ;Complement of A (inverse led logic)
        STA 3000H       ;Output LEDs
        JMP START       ;Go to START to check DIP switches

TURNALL:
        MVI A,00H       ;all leds ON
        STA 3000H       ;Output LEDs
        JMP START       ;Jump to START for continuous operation


TURNOFF:
        MVI A,FFH       ;all leds OFF
        STA 3000H       ;Output LEDs
        JMP START       ;Jump to START for continuous operation
        END
```

ii) Το πρόγραμμα αυτό αναμένει το πάτημα ενός hardware key. Αν το κουμπί είναι μεταξύ 1 και 4 αναβοσβήνουν τα 4 LSB των LED για 4 φορές. Αν έχει πατηθεί πλήκτρο μεταξύ 5 έως 8 αναβοσβήνουν τα 4 MSB LEDs 4 φορές. Αυτό γίνεται με την ρουτίνα “KIND” η οποία περνάει στον A τον κωδικό του πλήκτρου που πατήθηκε. Με 2 iterators στην συνέχεια οι οποίοι αρχικοποιούνται σε 8 και 4 και μειώνονται σε κάθε επανάληψη, ελέγχουμε σε ποια από τις 2 ομάδες ανήκει το πλήκτρο που πατήθηκε.

```
        LXI B,01F4H

START:  CALL KIND       ;detect an keyboard stroke and store the value in A
        MVI D,08H       ;D=8 because we want to detect only the buttons 1-8
        MVI E,04H       ;E=4 to check if the button is between 1 and 4 or
                        ;between 5 and 8

L1:     CMP D
        JZ LED_MSB      ;If A=D before E=0 go to turn on the MSBs
        DCR D           ;decrease by 1 the D in each iteration
        DCR E           ;decrease by 1 the E in each iteration
        JNZ L1          ;If E=0 go to check if the button is between 1-4

        MVI E,04H       ;E=4
L2:     CMP D
        JZ LED_LSB      ;If A=D before E=0 go to turn on the MSBs
        DCR D   ;decrease by 1 the D in each iteration
        DCR E           ;decrease by 1 the E in each iteration
        JZ START        ;if E=0 jump to START because no button between 1-8
                        ;has been pressed
        JMP L2          ;else jump again to L2

LED_LSB:
        MVI E,04H       ;E=4 to turn on/off 4 times
L3:     MVI A,F0H       ;A=11110000 to turn on 4 LSBs
        STA 3000H       ;Output LEDs
        CALL DELB       ;Add Delay
        MVI A,FFH       ;A=11111111 to turn off 4 LSBs
        STA 3000H       ;Output LEDs
        CALL DELB       ;Add delay
        DCR E           ;decrease E
        JZ START        ;If E=0 jump to start
        JMP L3          ;else jump to L3

LED_MSB:
        MVI E,04H       ;E=4 to turn on/off 4 times
L4:     MVI A,0FH       ;A=00001111 to turn on 4 MSBs
        STA 3000H       ;Output LEDs
        CALL DELB       ;Add delay
        MVI A,FFH       ;A=11111111 to turn off 4 MSBs
        STA 3000H       ;Output LEDs
        CALL DELB       ;Add delay
        DCR E           ;Decrease E
        JZ START        ;If E=0 go to start to check for key stroke
        JMP L4          ;Else, go to L4 to flash again
        END
```

iii) Το πρόγραμμα εδώ αναμένει από το πληκτρολόγιο να πατηθεί κάποιο πλήκτρο και στην συνέχεια εκτυπώνει τον κωδικό του στην οθόνη.

```
START:  IN 10H      ;Memory protection off
        LXI H,0A00H ;Point to 0800H memory location (the first storing block)
        MVI B,04H   ;For 4 blanks in the first four segments

BLANK:  MVI M,10H   ;Hex code of blank
        INX H       ;Increase HL register
        DCR B       ;Decrease B
        JNZ BLANK   ;If B!=0 go to Blank to add one more blank

LINE0:  MVI A,FEH   ;choose the line 0
        STA 2800H   ;
        LDA 1800H   ;load to A the reading button's port (porta anagnosis pliktron)
        ANI 07H     ;we need only the first three digits
        MVI C,86H   ;Code for INSTR STEP
        CPI 06H     ;check if A=00000110 i.e. the INSTR STEP button is pushed
        JZ SHOW     ;If yes jump to show in order to display it in 7-segments
        MVI C,85H   ;Code for FETCH PC
        CPI 05H     ;check if A=00000101 i.e. the FETCH PC button is pushed
        JZ SHOW     ;If yes jump to show in order to display it in 7-segments
        ;MVI C,F7H  ;These commands is in annotations because
        ;CPI 03H    ;we want to ignore the HDWR STEP
        ;JZ SHOW    ;

LINE1:  MVI A,FDH   ;choose the line 1
        STA 2800H   ;
        LDA 1800H   ;load to A the reading button's port (porta anagnosis pliktron)
        ANI 07H     ;we want to check only the first three digits
        MVI C,84H   ;Code for RUN
        CPI 06H     ;check if A=00000110 i.e. the RUN button is pushed
        JZ SHOW     ;If yes jump to show in order to display it in 7-segments
        MVI C,80H   ;COde for FETCH REG
        CPI 05H     ;check if A=00000101 i.e. the FETCH REG button is pushed
        JZ SHOW     ;If yes jump to show in order to display it in 7-segments
        MVI C,82H   ;Code for FETCH ADRS
        CPI 03H     ;check if A=00000011 i.e. the FETCH ADRS button is pushed
        JZ SHOW     ;If yes jump to show in order to display it in 7-segments

LINE2:  MVI A,FBH   ;Repeat the same process as above for the next lines
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,00H   ;For 0
        CPI 06H
        JZ SHOW
        MVI C,83H   ;For STORE/INCR
        CPI 05H
        JZ SHOW
        MVI C,81H   ;For DECR
        CPI 03H
        JZ SHOW

LINE3:  MVI A,F7H
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,01H   ;For 1
        CPI 06H
        JZ SHOW
        MVI C,02H   ;For 2
        CPI 05H
        JZ SHOW
        MVI C,03H   ;For 3
        CPI 03H
        JZ SHOW
```

```
LINE4:  MVI A,EFH
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,04H   ;For 4
        CPI 06H
        JZ SHOW
        MVI C,05H   ;For 5
        CPI 05H
        JZ SHOW
        MVI C,06H   ;For 6
        CPI 03H
        JZ SHOW

LINE5:  MVI A,DFH
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,07H   ;For 7
        CPI 06H
        JZ SHOW
        MVI C,08H   ;For 8
        CPI 05H
        JZ SHOW
        MVI C,09H   ;For 9
        CPI 03H
        JZ SHOW

LINE6:  MVI A,BFH
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,0AH   ;For A
        CPI 06H
        JZ SHOW
        MVI C,0BH   ;For B
        CPI 05H
        JZ SHOW
        MVI C,0CH   ;For C
        CPI 03H
        JZ SHOW
```

```
LINE7:  MVI A,7FH
        STA 2800H
        LDA 1800H
        ANI 07H
        MVI C,0DH   ;For D
        CPI 06H
        JZ SHOW
        MVI C,0EH   ;For E
        CPI 05H
        JZ SHOW
        MVI C,0FH   ;For F
        CPI 03H
        JZ SHOW
        JMP START   ;If no button is pushed loop to start to load again

SHOW:   LXI H,0A04H ;load to H the correct memory position
        MOV A,C     ;Move to A the code for the button that is being pushed
        ANI 0FH     ;We keep the 4 LSBs
        MOV M,A     ;And store them in 0A04H i.e. in the fifth postion of 7-segment display
        INX H       ;Next memory position
        MOV A,C
        ANI F0H     ;We keep the 4 MSBs
        RLC         ;Transfer them in LSBs's position
        RLC
        RLC
        RLC
        MOV M,A     ;Store them in 0A05H i.e. in the last postion of 7-segment display
        LXI D,0800H ;Load to D the fisrt position of block 0A00H - 0A05H to be ready for DCD
        CALL STDM
        CALL DCD    ;display the code
        JMP START   ;Loop to start for continuous operation
        END
```

4η Άσκηση

Στο πρόγραμμα αυτό υλοποιούμε τις boolean πράξεις τους σχήματος και εμφανίζουμε τα αποτελέσματα στο αντίστοιχο LED. Σαν ροή προγράμματος, υπολογίζουμε κάθε φορά μία πράξη και αποθηκεύουμε το αποτέλεσμα της προσωρινά στον B και στην συνέχεια τα υπόλοιπα αποτελέσματα γίνονται OR με το περιεχόμενο του B (τα αποτελέσματα των προηγούμενων πράξεων).

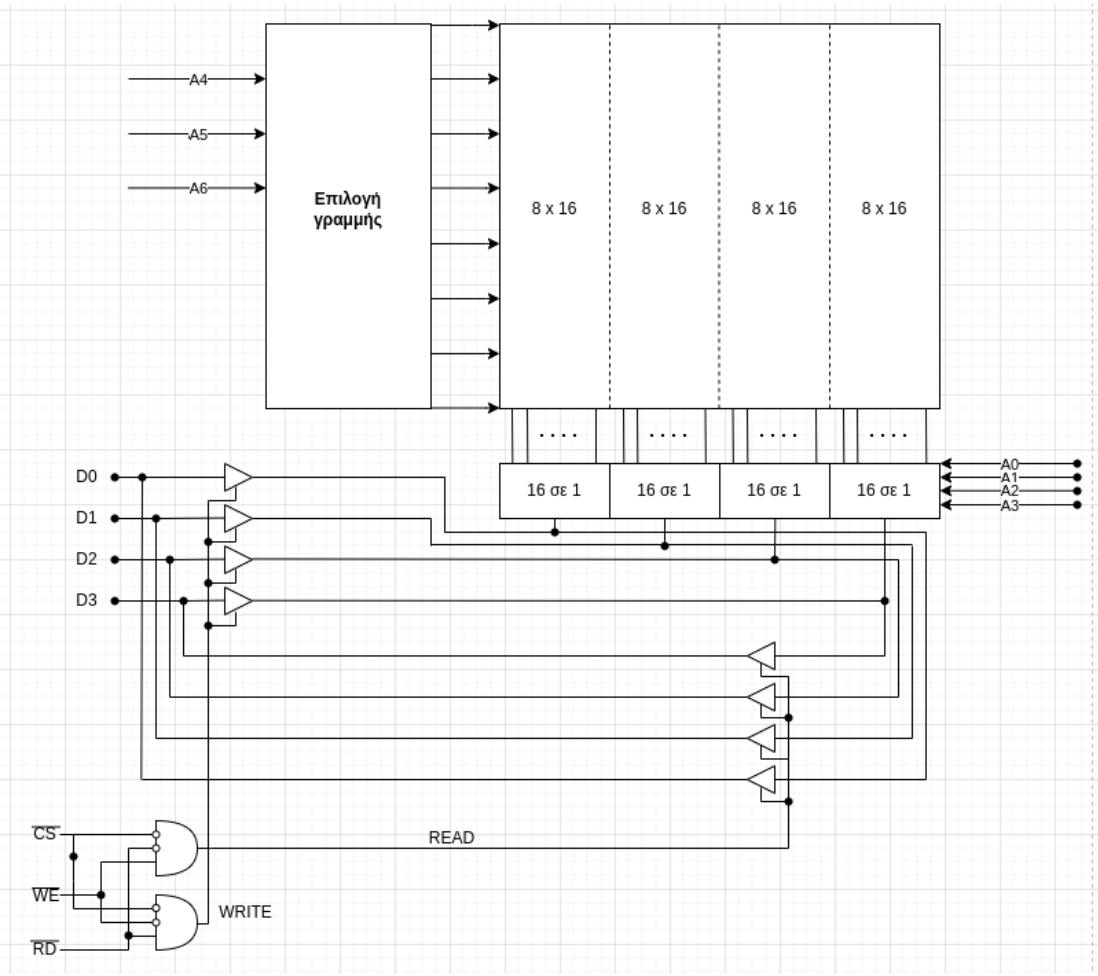
```
START:
LDA 2000H       ;load DIP
ANI 08H         ;isolate 4th digit (00001000)
MOV B,A         ;A1 is now in B
LDA 2000H       ;load DIP
ANI 04H         ;isolate 3rd digit (00000100)
RLC             ;shift it left to OR it with A1
ORA B           ;A1 OR B1
RRC             ;double right shift to put it in second LED
RRC
MOV B,A         ;X1 is in B

LDA 2000H       ;load DIP
ANI 01H         ;isolate 1st digit (00000001)
MOV C,A         ;C has B0
LDA 2000H       ;load DIP
ANI 02H         ;isolate 2nd digit (00000010)
RRC             ;A has A0
ORA C           ;A0 or B0
RLC             ;shift it left to AND it with X1
ANA B           ;X0 = (A0 or B0) AND X1
RRC             ;shift it right to put in first LED
ORA B           ;add result in B
MOV B,A         ;B now has 0,0,0,0,0,0,x1,x0

LDA 2000H       ;load DIP
ANI 10H         ;isolate 5th digit (00010000)
MOV C,A         ;C has B2
LDA 2000H       ;load DIP
ANI 20H         ;isolate 6th digit (00100000)
RRC             ;shift it right to AND with B2
ANA C           ;C = B2 AND A2
MOV D,A         ;D has A2 AND B2 (to use it later for X3)
RRC             ;shift right 2 times to put in X2 position
RRC             ;
ORA B           ;add X2 to B
MOV B,A         ;B now has 0,0,0,0,0,X2,X1,X0

LDA 2000H       ;load DIP
ANI 40H         ;isolate 7th digit (01000000)
MOV C,A         ;C has B3
LDA 2000H       ;load DIP
ANI 80H         ;isolate 8th digit (10000000)
RRC             ;shift right to perform AND with B3
ANA C           ;A = A3 AND B3
RRC             ;double right shift to perform XOR with X2
RRC             ;
XRA D           ;A = (A3 AND B3) XOR X2
RRC             ;shift right to put in X3 position
ORA B           ;add B result to A
CMA             ;complement A (inverser LED logic)
STA 3000H       ;output LEDs
JMP START       ;run again
END
```

5η Άσκηση



Στην παραπάνω φαίνεται η εσωτερική δομή μιας SRAM 128x4bit. Τα bit A0 έως A7 είναι τα bit διεύθυνσης ενώ τα bit D0 εώς D3 αποτελούν την γραμμή δεδομένων.

Έστω ότι θέλουμε να διαβάσουμε κάτι στην διεύθυνση 1111001. Μέσω των τριών MSB (A4,A5,A6) επιλέγουμε μία απο τις 8 γραμμές της SRAM. Οι 4 πολυπλέκτες 16 σε 1, οι οποίοι ελέγχονται από τα 4 LSB (A3-A2-A1-A0) θα επιλέξουν μία τετράδα που θα συνδεθεί στις γραμμές των δεδομένων. Στο παράδειγμα μας θα επιλεγεί η 8η γραμμή και τα 4 LSB θα επιλέξουν την 6η θέση κάθε πολυπλέκτη.
Όσον αφορά το πότε γίνεται ανάγνωση ή εγγραφή, αυτό καθορίζεται από τα τρία σήματα (CS)’ , (WE)’ και (RD)’.

Αν θέλουμε να διαβάσουμε από την συγκεκριμένη θέση θα είναι 0 τα σήματα (CS)’ και (RD)’ και 1 το (WE)’ άρα θα ενεργοποιηθεί η κάτω AND πύλη που αφορά το READ και αυτή με την σειρά της θα ενεργοποιήσει την κάτω τετράδα απομονωτών που αναφέρονται στο διάβασμα.

Αντίστοιχα αν θέλουμε να γράψουμε στην συγκεκριμένη θέση, θα έχουμε 0 στα (CS)’ και (WE)’ και 1 στο (RD)’. Έτσι θα ενεργοποιηθεί η πάνω AND πύλη και αυτή με την σειρά της θα ενεργοποιήσει την πάνω τετράδα απομονωτών που επιτρέπουν την εγγραφή.

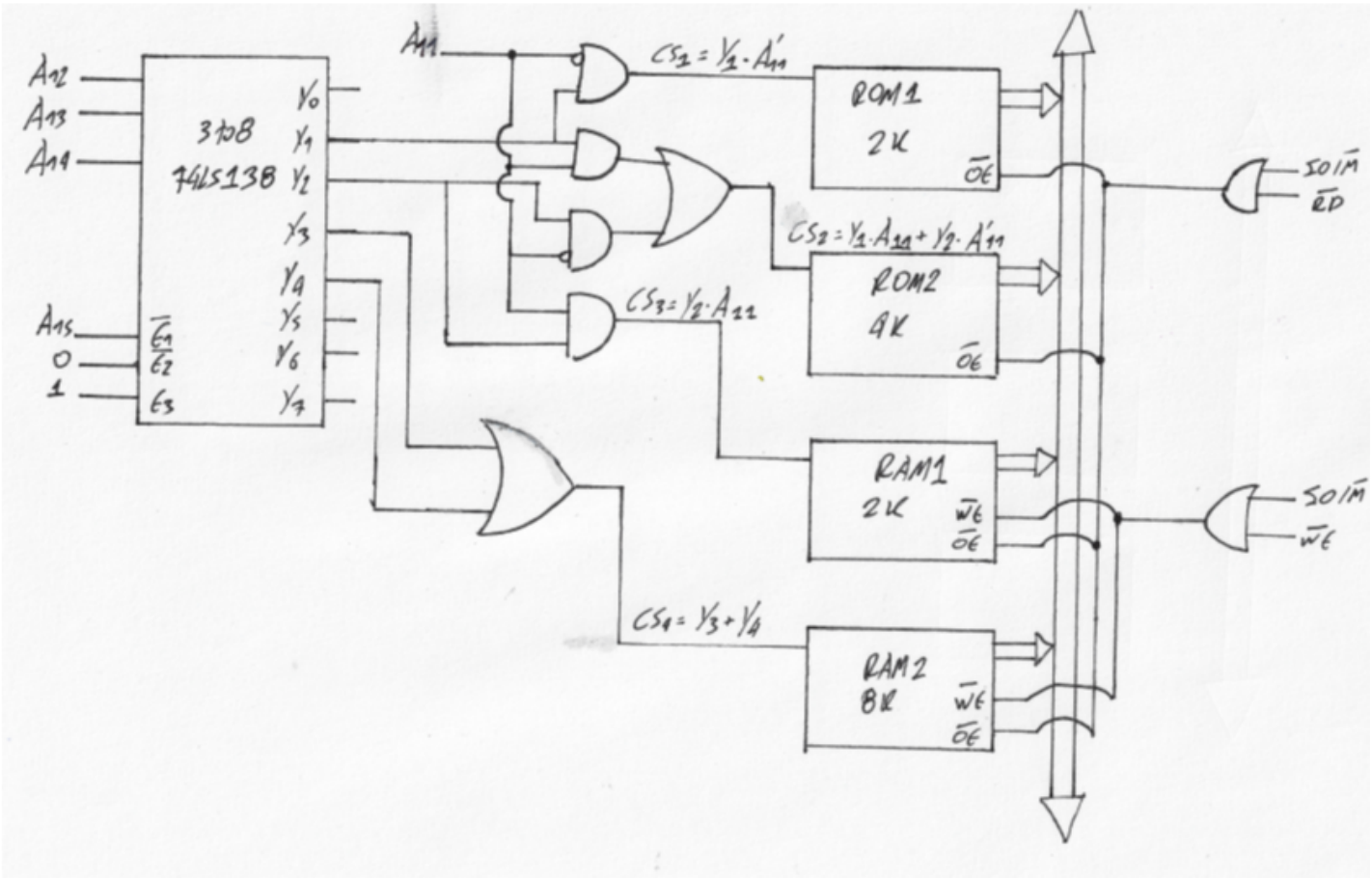
Για την ανάγνωση και την εγγραφή χρησιμοποιούμε 2 πύλες AND, 3ων εισόδων η κάθε μία, ώστε να μην γίνει ούτε ανάγνωση ούτε εγγραφή αν έρθει ταυτόχρονα σήμα (λανθασμένο) εγγραφής και ανάγνωσης.

6η Άσκηση

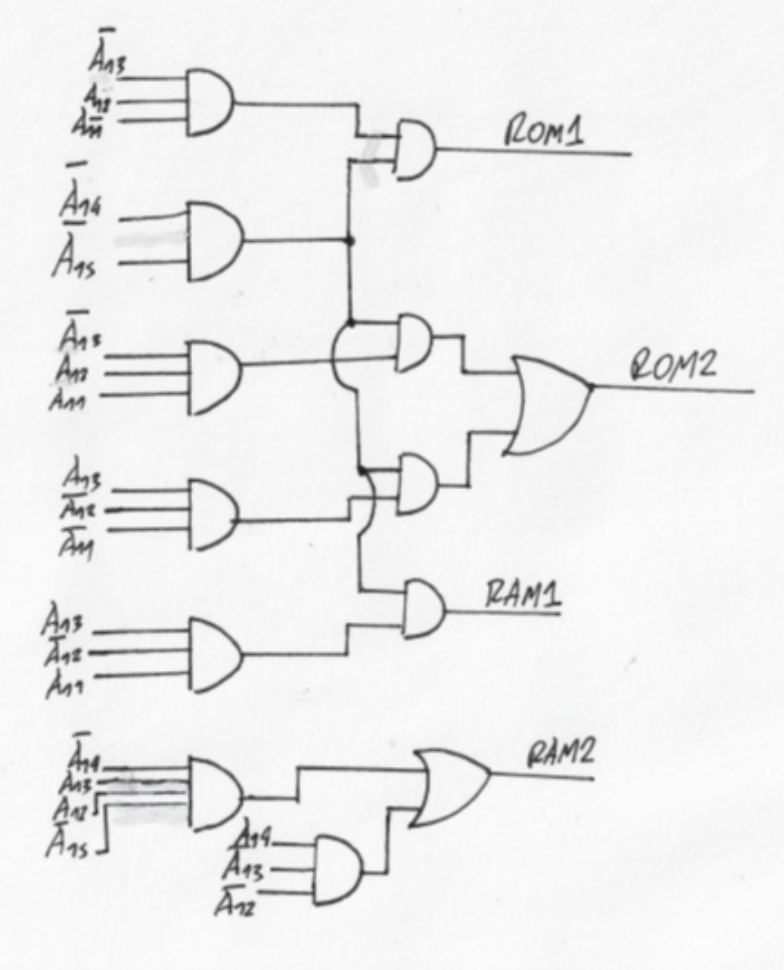
Ο χάρτης μνήμης είναι ο εξής:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Chip
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000H	ROM1 2K
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	17FFH	
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1800H	ROM2 4K
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	27FFH	
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	2800H	RAM 2K
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFFH	
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3000H	RAM 8K
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFFH	

α) Υλοποιούμε το σύστημα μνήμης του 8085 με έναν αποκωδικοποιητή 3 σε 8 και λογικές πύλες
Για να γίνεται η ενεργοποίηση του κάθε memory chip θα χρησιμοποιήσουμε τα A11, A12, A13, A14. Τα 3 πρώτα θα αποτελέσουν την είσοδο του αποκωδικοποιητή.



β) Υλοποιούμε πάλι τον ίδιο χάρτη μνήμης, ωστόσο τώρα χρησιμοποιούμε μόνο λογικές πύλες.

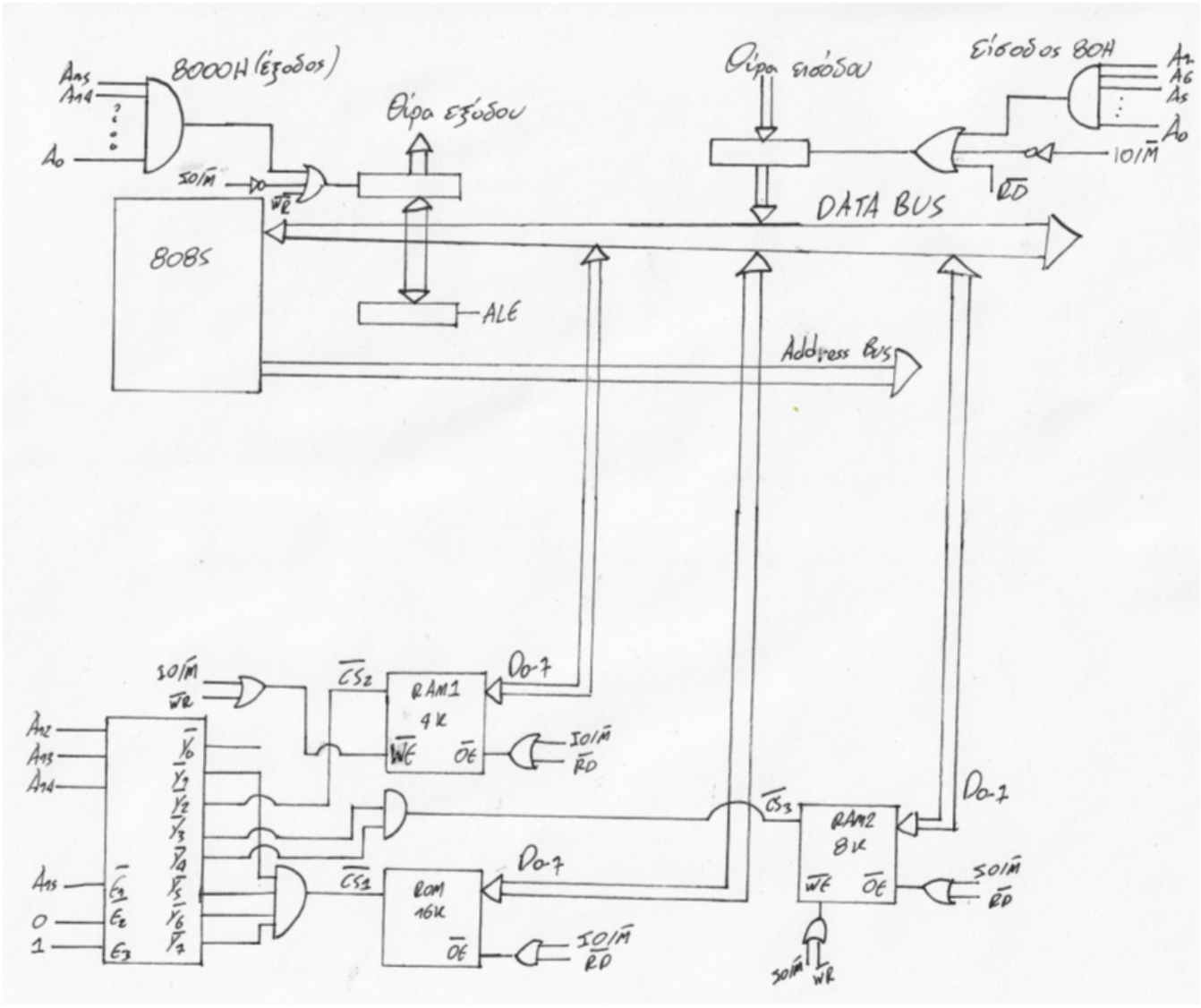


7η Άσκηση

Ο χάρτης μνήμης του μΥ-Σ 8085:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Chip
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000H	ROM1 4K
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H	RAM1 4K
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFFH	
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3000H	RAM2 8K
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFFH	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000H	ROM 12K
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFFH	

Πάλι στον αποκωδικοποιητή θα στείλουμε τα A12, A13, A14.


8000H (έξοδος)
Θύρα εξόδου
Θύρα εισόδου
Είσοδος 80Η
DATA BUS
8085
ALE
Address Bus
IO/M
WR
RD
RAM1
4K
RAM2
8K
ROM
16K
D0-7