2η Ομάδα Ασκήσεων
Συστήματα Μικροϋπολογιστών

Ανδρέας Στάμος
Αριθμός Μητρώου: *03120****

Μάιος 2023

Περιεχόμενα

1 Άσκηση 1

1.1 Ερώτημα α

```
IN 10H ;disable memory protection
MVI A,00H ;counter = 0
LXI H,0900H ;initial address to save numbers

BYTE:
CPI 80H ;if counter == 80: end
JZ EXIT
```

```
MOV M,A ;move counter to memory
INR A ;++counter
INX H ;++memory address
JMP BYTE ;repeat loop

EXIT:
END
```

1.2 Ερώτημα β

```
IN 10H ;disable memory protection
MVI A,00H ;counter = 0
LXI H,0900H ;initial address to save numbers

BYTE:
CPI 80H ;if counter == 80: end
JZ ASK2
MOV M,A ;move counter to memory
INR A ;++counter
INX H ;++memory address
JMP BYTE ;repeat loop

ASK2:
LXI H,0900H ;initial address of numbers
LXI B,0000H ;counter = 0
BYTE2:
MVI A,80H ;this is for the check below
CMP L ;if memory address == 0980H then EXIT
JZ EXIT
INR L ;++memory address
MOV A,M ;A=*(memory address)
MVI D,09H ;bitcounter = 9

BIT: ;while --bitcounter != 0 (will repeat 8 times)
DCR D ;--bitcounter
JZ BYTE2 ;if bitcounter == 0 then byte is complete
RLC ;MSB to CY and left shift of A
JNC BIT ;if msb == 0: go to next bit
INX B ;if msb == 1: ++counter
JMP BIT

EXIT:
RST 1 ;stop show that we can see the result at the register pair
 ↪ [BC]
END
```

1.3 Ερώτημα γ

```
IN 10H ;disable memory protection
MVI A,00H ;counter = 0
```

```
LXI H,0900H ;initial address to save numbers

BYTE:
CPI 80H ;if counter == 80: end
JZ ASK3
MOV M,A ;move counter to memory
INR A ;++counter
INX H ;++memory address
JMP BYTE ;repeat loop

ASK3:
LXI H,0900H ;initial address of numbers
LXI B,0000H ;counter = 0
BYTE2:
MVI A,80H ;this is for the check below
CMP L ;if memory address == 0980H then EXIT
JZ EXIT
INR L ;++memory address
MOV A,M ;A=*(memory address)

CPI 10H ;if A-0x10<0 go to next byte
JC BYTE2
CPI 61H ;if A-0x61>=0 go to next byte
JNC BYTE2
INX B ;++counter
JMP BYTE2

EXIT:
RST 1 ;stop show that we can see the result at the register pair
 ↪ [BC]
END
```

2 Άσκηση 2

```
IN 10H ;disable memory protection
LXI B,0064H ;delay step of 100ms

OFF0: ;state where nothing has happened
MVI A,FFH
STA 3000H ;turn off all leds
LDA 2000H
RLC
JC ON0 ;check if msb dip is on
JMP OFF0

ON0: ;state where msb dip is on
LDA 2000H
RLC
JNC OFF1 ;check if msb dip is off
```

```
JMP ONO

OFF1: ;state where msb dip is OFFed after has been ONed
MVI D,C8H
MVI A,00H
STA 3000H

OFF1_0: ;while in state OFF1 looping for 200*100ms=20sec to check for extending time
DCR D ;decrease the loop variable
JZ OFF0 ;if out of time (D==0) return to initial off state
CALL DELB ;delay 100ms
LDA 2000H ;check if msb dip is on
RLC
JC OFF1_1
JMP OFF1_0

OFF1_1: ;state where msb dip is ONed while before in OFF state but LEDs were ON
DCR D ;decrease the loop variable
JZ OFF0 ;if out of time (D==0) return to initial off state
CALL DELB ;delay 100ms
LDA 2000H ;check if msb dip is off
RLC
JNC OFF1
JMP OFF1_1

END
```

3 Άσκηση 3

3.1 Ερώτημα i

```
L1:
MVI B,FEH
LDA 2000H ;reading the dip switches

BYTE: ;looping every bit inside bytes of dip switches
RRC
JC SHOW ;if found the rightmost bit that is 1 goto show
MOV C,A
MOV A,B
CPI 7FH ;if the ON led reached the leftmost posotion, then we've checked the whole byte and the loop is over
JZ SHOWNONE
RLC ;move the ON led one position left
MOV B,A
MOV A,C
JMP BYTE
```

```
16
17  SHOW: ;showing the leds
18  MOV A,B
19  STA 3000H
20  JMP L1
21
22  SHOWNONE: ;do not show any leds.
23  MVI A,FFH
24  STA 3000H
25  JMP L1
26
27  END
```

3.2 Ερώτημα ii

```
1   L1:
2   CALL KIND ;inputing from the keyboard into A
3
4   CPI 01H ;if A<1 show no leds
5   JC SHOWNONE
6   CPI 09H ;if A>=9 show no leds
7   JNC SHOWNONE
8
9   MOV B,A
10  MVI A,00H
11
12  NUMBER: ;looping over the value of B
13  DCR B
14  JZ SHOW
15  RLC ;shifting left by adding 1 to the rightmost position
16  INR A
17  JMP NUMBER
18
19  SHOW: ;showing the leds
20  STA 3000H
21  JMP L1
22
23  SHOWNONE: ;showing no leds
24  MVI A,FFH
25  STA 3000H
26  JMP L1
27
28  END
```

3.3 Ερώτημα iii

```
1   IN 10H
2   MVI A,1CH
3   STA 0B00H
4   STA 0B01H
```

```
STA 0B02H
STA 0B03H
STA 0B04H
STA 0B05H

L1:

;inputing from keyboard

;line 0
MVI A,FEH
STA 2800H
LDA 1800H
ANI 07H
CPI 06H ;checking if right button was clicked
JNZ CONT_0_0
MVI B,86H
JMP SHOW
CONT_0_0:
CPI 05H ;checking if middle button was clicked
JNZ CONT_0_1
MVI B,85H
JMP SHOW
CONT_0_1:
CPI 03H ;checking if left button was clicked
JNZ CONT_0_2
MVI B,F7H
JMP SHOW
CONT_0_2:


;line 1
MVI A,FDH
STA 2800H
LDA 1800H
ANI 07H
CPI 06H
JNZ CONT_1_0
MVI B,84H
JMP SHOW
CONT_1_0:
CPI 05H
JNZ CONT_1_1
MVI B,80H
JMP SHOW
CONT_1_1:
CPI 03H
JNZ CONT_1_2
MVI B,82H
JMP SHOW
```

```
CONT_1_2:


;line 2
MVI A,FBH
STA 2800H
LDA 1800H
ANI 07H
CPI 06H
JNZ CONT_2_0
MVI B,00H
JMP SHOW
CONT_2_0:
CPI 05H
JNZ CONT_2_1
MVI B,83H
JMP SHOW
CONT_2_1:
CPI 03H
JNZ CONT_2_2
MVI B,83H
JMP SHOW
CONT_2_2:


;line 3
MVI A,F7H
STA 2800H
LDA 1800H
ANI 07H
CPI 06H
JNZ CONT_3_0
MVI B,01H
JMP SHOW
CONT_3_0:
CPI 05H
JNZ CONT_3_1
MVI B,02H
JMP SHOW
CONT_3_1:
CPI 03H
JNZ CONT_3_2
MVI B,03H
JMP SHOW
CONT_3_2:


;line 4
MVI A,EFH
STA 2800H
```

```
LDA 1800H
ANI 07H
CPI 06H
JNZ CONT_4_0
MVI B,04H
JMP SHOW
CONT_4_0:
CPI 05H
JNZ CONT_4_1
MVI B,05H
JMP SHOW
CONT_4_1:
CPI 03H
JNZ CONT_4_2
MVI B,06H
JMP SHOW
CONT_4_2:


;line 5
MVI A,DFH
STA 2800H
LDA 1800H
ANI 07H
CPI 06H
JNZ CONT_5_0
MVI B,07H
JMP SHOW
CONT_5_0:
CPI 05H
JNZ CONT_5_1
MVI B,08H
JMP SHOW
CONT_5_1:
CPI 03H
JNZ CONT_5_2
MVI B,09H
JMP SHOW
CONT_5_2:


;line 6
MVI A,BFH
STA 2800H
LDA 1800H
ANI 07H
CPI 06H
JNZ CONT_6_0
MVI B,0AH
JMP SHOW
```

```
155 CONT_6_0:
156 CPI 05H
157 JNZ CONT_6_1
158 MVI B,0BH
159 JMP SHOW
160 CONT_6_1:
161 CPI 03H
162 JNZ CONT_6_2
163 MVI B,0CH
164 JMP SHOW
165 CONT_6_2:
166
167
168 ;line 7
169 MVI A,7FH
170 STA 2800H
171 LDA 1800H
172 ANI 07H
173 CPI 06H
174 JNZ CONT_7_0
175 MVI B,0DH
176 JMP SHOW
177 CONT_7_0:
178 CPI 05H
179 JNZ CONT_7_1
180 MVI B,0EH
181 JMP SHOW
182 CONT_7_1:
183 CPI 03H
184 JNZ CONT_7_2
185 MVI B,0FH
186 JMP SHOW
187 CONT_7_2:
188
189 ;if no button was clicked the result is already in memory
190 JMP SHOWNOMEMS
191
192 SHOW:
193 ;splitting A=XYH to XH and YH (two characters)
194 MOV A,B
195 ANI 0FH
196 STA 0B04H
197 MOV A,B
198 RRC
199 RRC
200 RRC
201 RRC
202 ANI 0FH
203 STA 0B05H
204 ;showing to screen
```

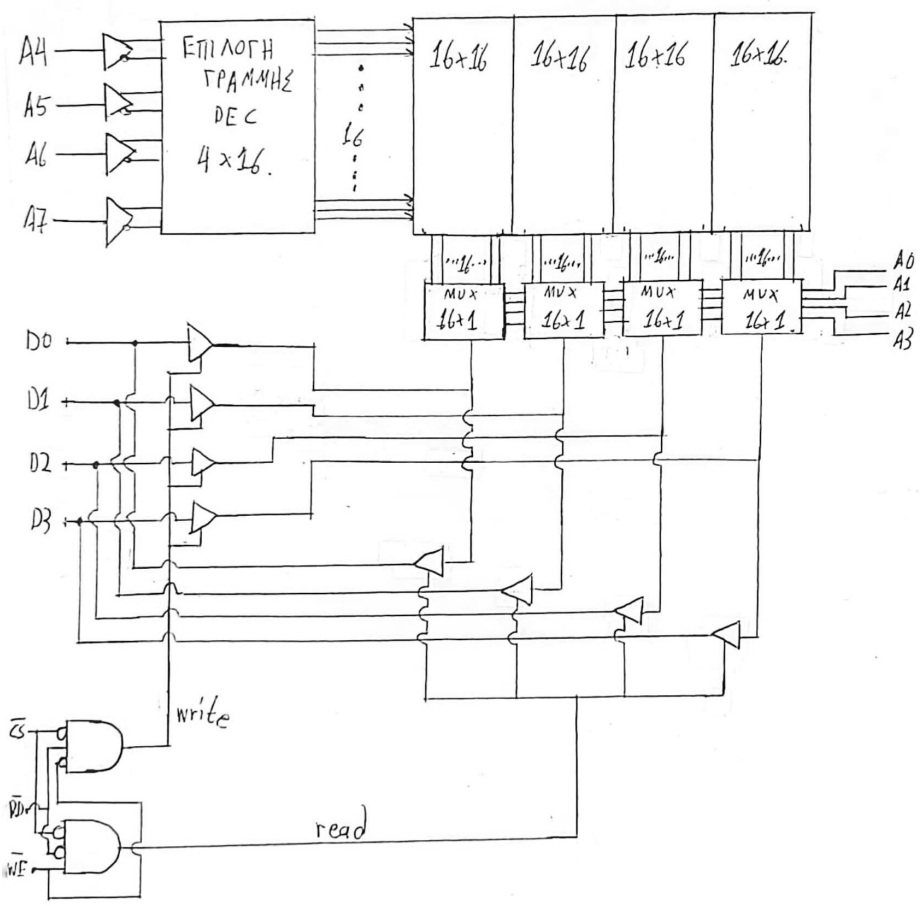
```
205 CALL STDM
206 SHOWNOMEMS:
207 LXI D,0B00H
208 CALL DCD
209 JMP L1
210
211 END
```

4 Άσκηση 4

```
1  IN 10H
2  L1:
3
4  LDA 2000H ;inputting from dip switches
5  MOV B,A
6
7  ;getting bits 0 and 1, ANDing them
8  ;and store result at pos 0 of C
9  ANI 01H
10 MOV D,A
11 MOV A,B
12 ANI 02H
13 RRC
14 ANA D
15 MOV C,A
16
17 ;getting bits 2 and 3, ANDing them
18 ;and store result at pos 1 of C
19 MOV A,B
20 ANI 04H
21 MOV D,A
22 MOV A,B
23 ANI 08H
24 RRC
25 ANA D
26 RRC
27 ORA C
28 MOV C,A
29
30
31 ;getting bits 4 and 5, XORing them
32 ;and store result at pos 2 of C
33 MOV A,B
34 ANI 10H
35 MOV D,A
36 MOV A,B
37 ANI 20H
38 RRC
39 XRA D
```

```
40  RRC
41  RRC
42  ORA C
43  MOV C,A
44
45
46  ;getting bits 6 and 7, XORing them
47  ;and store result at pos 3 of C
48  MOV A,B
49  ANI 40H
50  MOV D,A
51  MOV A,B
52  ANI 80H
53  RRC
54  XRA D
55  RRC
56  RRC
57  RRC
58  ORA C
59  MOV C,A
60
61  ;ORing bits 1 and 3 of C only with bits 2 and 4 of C
62  RRC
63  ANI 05H
64  ORA C
65
66  CMA ;complement because of the complement logic of leds
67  STA 3000H ;showing to leds
68
69  JMP L1
70
71  END
```

5 Άσκηση 5



Σχήμα 1: Εσωτερική οργάνωση μνήμης άσκησης 5

Η εσωτερική οργάνωση της μνήμης SRAM 256x4 bit σχεδιάστηκε στο σχήμα 5. Από τις γραμμές A4-A7, μέσω ενός αποκωδικοποιητή επιλέγεται η γραμμή διεύθυνσης. Από τις γραμμές A0-A3, μέσω 4 πολυπλεκτών 16x1 επιλέγεται η 4-αδα στηλών που αντιστοιχεί στα αιτούμενα 4 bits. Οι γραμμές D0-D4 αποτελεούν τις γραμμές δεδομένων (ενώνονται με τις εξόδούς των πολυπλεκτών). Τα σήματα $\overline{RD}$, $\overline{WE}$ (προσοχή πως δίνεται το συμπλήρωμα ως είσοδος) καθορίζουν αν θα συμβεί ανάγνωση ή εγγραφή στην μνήμη. Τέλος το σήμα $\overline{CS}$ (προσοχή πως δίνεται το συμπλήρωμα ως είσοδος) καθορίζει αν η μνήμη θα αλληλεπιδράσει με τις γραμμές δεδομένων, ανεξάρτητα από τα RD, WE.

Με σκοπό ένα παράδειγμα λειτουργίας της μνήμης, έστω η διεύθυνση μνήμης A0-A7 = 0101 0111. Επιλέγεται η 7η (A4-A7 = 1001) γραμμή και η 5η (A0-A3 = 0101) τετράδα στηλών. Αν $RD = 1$ και $WE = 0$ οι εξόδοι των πολυπλεκτών οδηγούνται προς τις γραμμές D0-D3, ενώ αν $RD = 0$ και $WE = 1$ οι γραμμές D0-D3 οδηγούνται προς τους πολυπλέκτες. Επίσης, ως προς την είσοδο CS, αν

$CS = 0$ η μνήμη δεν λαμβάνει ούτε στέλνει δεδομένα, ενώ αν $CS = 1$ ακολουθεί την προαναφερθείσα λειτουργία.

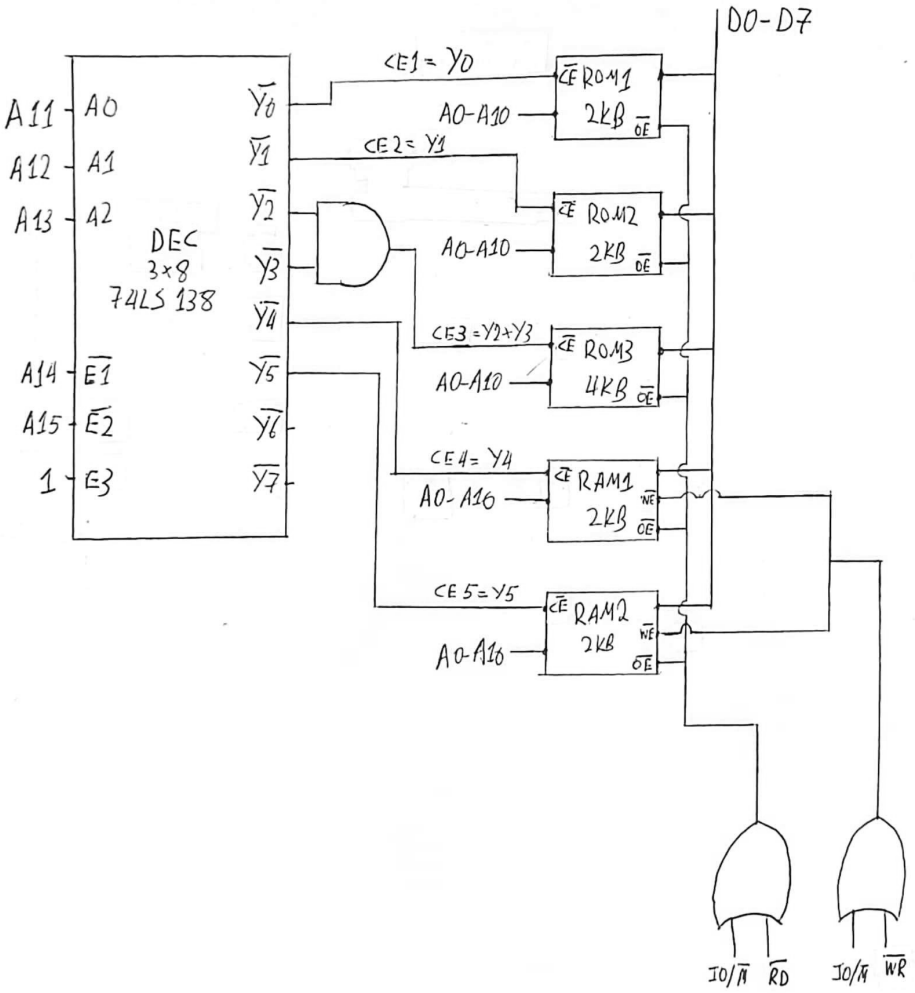
6 Άσκηση 6

Ο χάρτης μνήμης φαίνεται στην εικόνα 6. Παρατηρούμε πως κάθε chip μπορεί να επιλεγεί με χρήση των bits A11-A13.

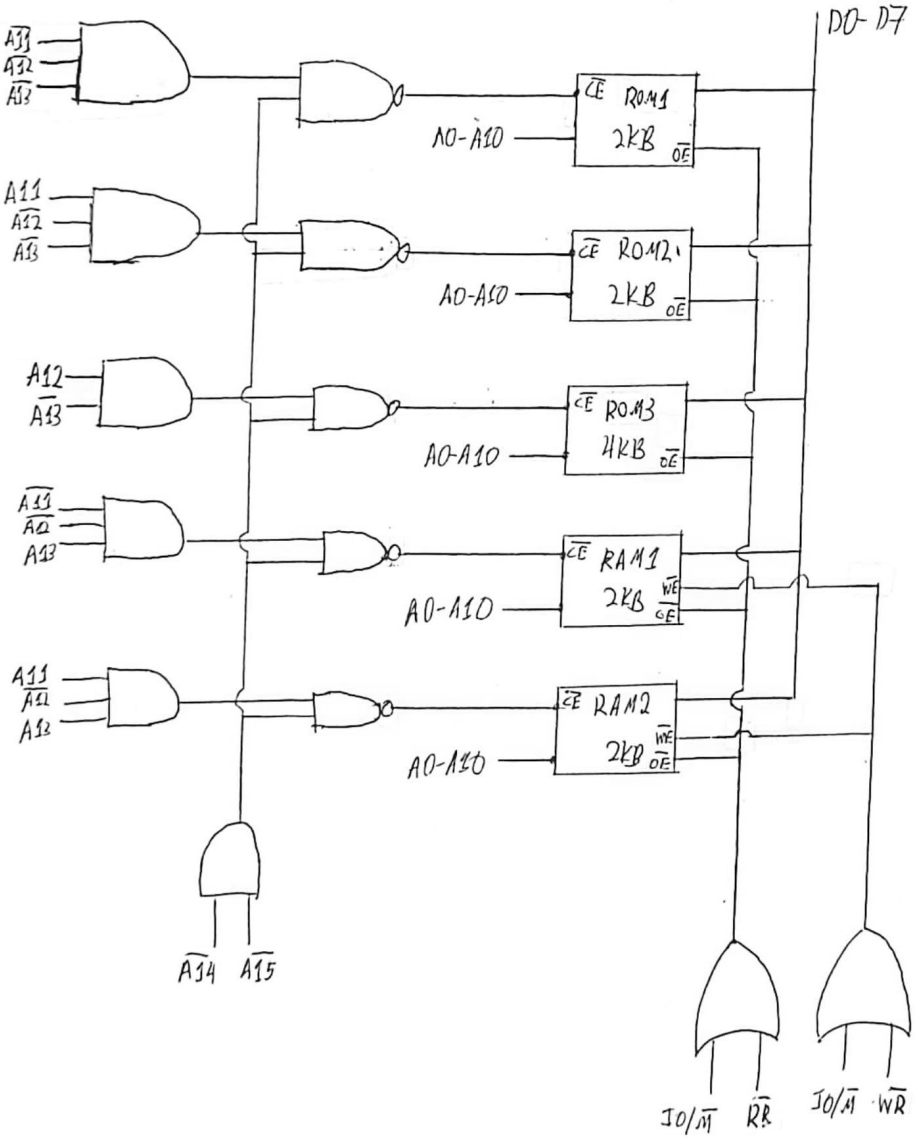
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Διεύθυνση HEX	Chip
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	ROM1-2K
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FF	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0800	ROM2-2K
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFF	
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000	ROM3-4K
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000	RAM1-2K
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	27FF	
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0000	RAM2-2K
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	07FF	

Σχήμα 2: Χάρτης μνήμης για το σύστημα της Άσκησης 6. Με κόκκινο χρώμα απεικονιζόνται τα απαιτούμενα bits για την επιλογή του σωστού chip ανά διεύθυνση

Η υλοποίηση του ζητούμενου συστήματος μνήμης με χρήση αποκωδικοποιητή φαίνεται στην εικόνα 6, ενώ με χρήση μόνο λογικών πυλών φαίνεται στην εικόνα 6.



Σχήμα 3: Σύστημα άσκησης 6 με χρήση αποκωδικοποιητή



Σχήμα 4: Σύστημα άσκησης 6 με χρήση μόνο λογικών πυλών

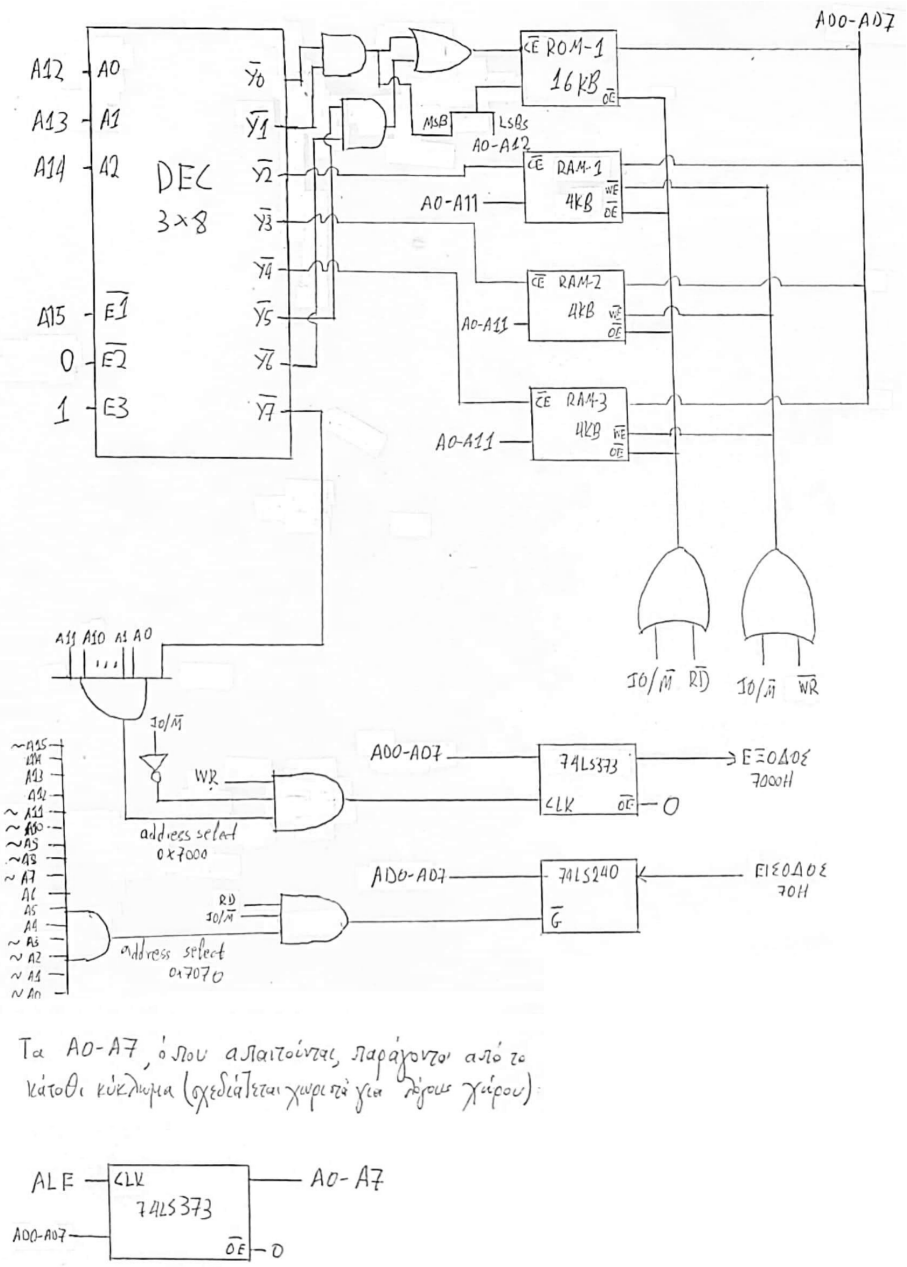
7 Άσκηση 7

Ο χάρτης μνήμης φαίνεται στην εικόνα 7.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Διεύθυνση HEX	Chip
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000	ROM-16K
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF	(1st half)
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000	RAM1-4K
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	RAM2-4K
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000	RAM3-4K
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFF	
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000	ROM-16K
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	6FFF	(1st half)
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	7000	Output

Σχήμα 5: Χάρτης μνήμης για το σύστημα της Άσκησης 7. Με κόκκινο χρώμα απεικονιζόνται τα απαιτούμενα bits για την επιλογή του σωστού chip ανά διεύθυνση

Η υλοποίηση του ζητούμενου μΥ-Σ φαίνεται στην εικόνα 7.



Σχήμα 6: Σύστημα άσκησης 7