




**Εθνικό Μετσόβιο Πολυτεχνείο**

**Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών**

**6<sup>ο</sup> Εξάμηνο**

**Συστήματα Μικροϋπολογιστών**

**2<sup>η</sup> ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ**

  
**Χατζή Ήβη**

## Ασκήσεις προσομοίωσης

### 1<sup>η</sup> ΑΣΚΗΣΗ

```
IN 10H
MVI A,00H      ;numbers 0-255 in A
LXI B,0000H    ;number of binary ones in B
MVI D,00H      ;numbers 10H-60H in D
LXI H,0900H    ;addresses
```

```
NUMBER:
  MOV M,A      ;save number A in memory
  MOV E,A      ;save number A in E cause we edit A in ONES loop
```

```
ONES:
  CPI 00H      ;when A is all zeros, stop checking
  JZ CONTINUE
  STC
  CMC          ;set carry to 0
  RAR          ;put each bit in carry and check
  JNC ONES
  INX B        ;found a one
  JMP ONES
```

```
CONTINUE:
  MOV A,E      ;restore number in A
  CPI 10H
  JC NEXT      ;A<10H, check next number
  CPI 60H
  JC ADD       ;A<60H
  JNZ NEXT     ;A>60H, check next number
```

```
ADD:
  INR D        ;10H<=A<=60H so increase D
```

```
NEXT:
  INR A        ;next number
  INX H        ;next space in memory
  CPI 00H      ;when A reaches 256 it rolls back to 00H
  JZ FINISH
  JMP NUMBER
```

```
FINISH:
  END
```

## 2<sup>η</sup> ΑΣΚΗΣΗ

```
LXI B,00C8H           ;delay 200ms=C8H

SET:
    MVI A,FFH
    STA 3000H
    MVI D,64H           ;timer 20s=200ms*100, 100=64H

START:
    LDA 2000H           ;input
    RAL                 ;msb in carry
    CALL DELB           ;delay 200ms
    JC OFF1             ;msb is off
    JMP START

OFF1:
    LDA 2000H           ;get input
    RAL                 ;check msb
    CALL DELB           ;delay 200ms
    JNC OFF1            ;msb is off

ON1:
    LDA 2000H           ;get input
    RAL                 ;check msb
    CALL DELB           ;delay 200ms
    JC ON1              ;msb is on

OFF2:
    MVI A,00H
    STA 3000H           ;light LEDS
    CALL DELB           ;delay 200ms

    LDA 2000H           ;get input
    RAL                 ;check msb
    JC ON2              ;msb is on

    DCR D               ;decrease timer
    MOV A,D
    CPI 00H             ;check if timer reached 0
    JNZ OFF2            ;timer not 0, keep lighting
    JMP SET             ;timer reached 0, reset

ON2:
    DCR D               ;decrease timer
    MOV A,D
    CPI 00H             ;check if timer reached 0
    JZ SET              ;timer reached 0, reset

    LDA 2000H           ;get input
    RAL                 ;check msb
    JC RESETTIMER       ;msb is off, reset timer

    MVI A,00H
    STA 3000H           ;light LEDS
    CALL DELB           ;delay 200ms
```

```

    JMP ON2

RESETTIMER:
    MVI D,64H
    JMP OFF2

END

```

### 3<sup>η</sup> ΑΣΚΗΣΗ

i.

```

START:
    LDA 2000H
    MOV B,A      ;put input in B
    MVI C,00H    ;C is counter
    MVI D,01H    ;set D to 00000001
LOOP1:
    INR C
    MOV A,B
    ANI 01H      ;lsb=1?
    JNZ LIGHT    ;found rightmost 1, light LEDs
    MOV A,B
    RRC          ;move to check next bit
    MOV B,A
    MOV A,D
    RAL          ;rotate D (D holds LED that will light)
    MOV D,A
    MOV A,C
    CPI 08H      ;checked all bits, loop again
    JNZ LOOP1
LIGHT:
    MOV A,D
    CMA
    STA 3000H
    JMP START

END

```

ii.

```
START:
    CALL KIND      ;input at A
    CPI 09H        ;if A>=9 get input again
    JNC START
    CPI 00H        ;if A=0 get input again
    JZ START
    MOV B,A        ;put input in B
    MVI D,FFH      ;D=11111111

LOOP2:
    DCR B
    MOV A,D
    RAL            ;rotate D and pad with 0
    MOV D,A
    MOV A,B
    CPI 00H        ;if B>0 keep looping
    JNZ LOOP2
    MOV A,D
    RAR            ;rotate D once to the right cause we have done one extra
rotation
    ORI 80H        ;set MSB=1
    CMA
    STA 3000H
    JMP START
END
```

iii.

```
IN 10H
LXI H,0A30H      ;output at 0A30H, far from program data
MVI M,10H      ;10H means print nothing
INX H          ;store output at 0A30H-0A35H
MVI M,10H
INX H
MVI M,10H      ;0A30H to 0A33H digits 1-3 print nothing
INX H
MVI M,10H
```

START:

LINE\_0:

```
MVI A,FEH      ;FEH = 11111110, select line 0
STA 2800H      ;store in 2800H
LDA 1800H      ;read column
ANI 07H        ;3 lsb have the info
MVI C,86H      ;set C=INSTR STEP
CPI 06H        ;if A=00000110 then button from column 1 is pressed
JZ DISPLAY     ;go to display it
MVI C,85H      ;C=FETCH PC
CPI 05H        ;if A=00000101 then button from column 2 is pressed
JZ DISPLAY
MVI C,F7H      ;C=HRDWR STEP
CPI 03H        ;if A=00000011 then button from column 3 is pressed
JZ DISPLAY
```

LINE\_1:

```
MVI A,FDH
STA 2800H
LDA 1800H
ANI 07H
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:

```
MVI A,FBH
STA 2800H
LDA 1800H
ANI 07H
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:

```
MVI A,F7H
STA 2800H
LDA 1800H
ANI 07H
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:

```
MVI A,EFH
STA 2800H
LDA 1800H
ANI 07H
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:

```
MVI A,DFH
STA 2800H
LDA 1800H
ANI 07H
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:

```
MVI A,BFH
STA 2800H
LDA 1800H
ANI 07H
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:
        MVI A,7FH
        STA 2800H
        LDA 1800H
        ANI 07H
        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,0A34H
        MOV A,C      ;C has output
        ANI 0FH      ;isolate the 4 LSBs
        MOV M,A      ;store them to 0A30H(first digit from left)
        INX H        ;HL++
        MOV A,C
        ANI F0H      ;isolate the 4 MSBs
        RRC          ;shift them to the 4 LSBs
        RRC
        RRC
        RRC
        MOV M,A      ;store them to 0A31(second digit from left)

        LXI D,0A30H    ;move the block 0A30 to 0A35 to the memory
        CALL STDM     ;where the DCD routine reads
        CALL DCD      ;print

        JMP START

END

```



## 4<sup>η</sup> ΑΣΚΗΣΗ

START:

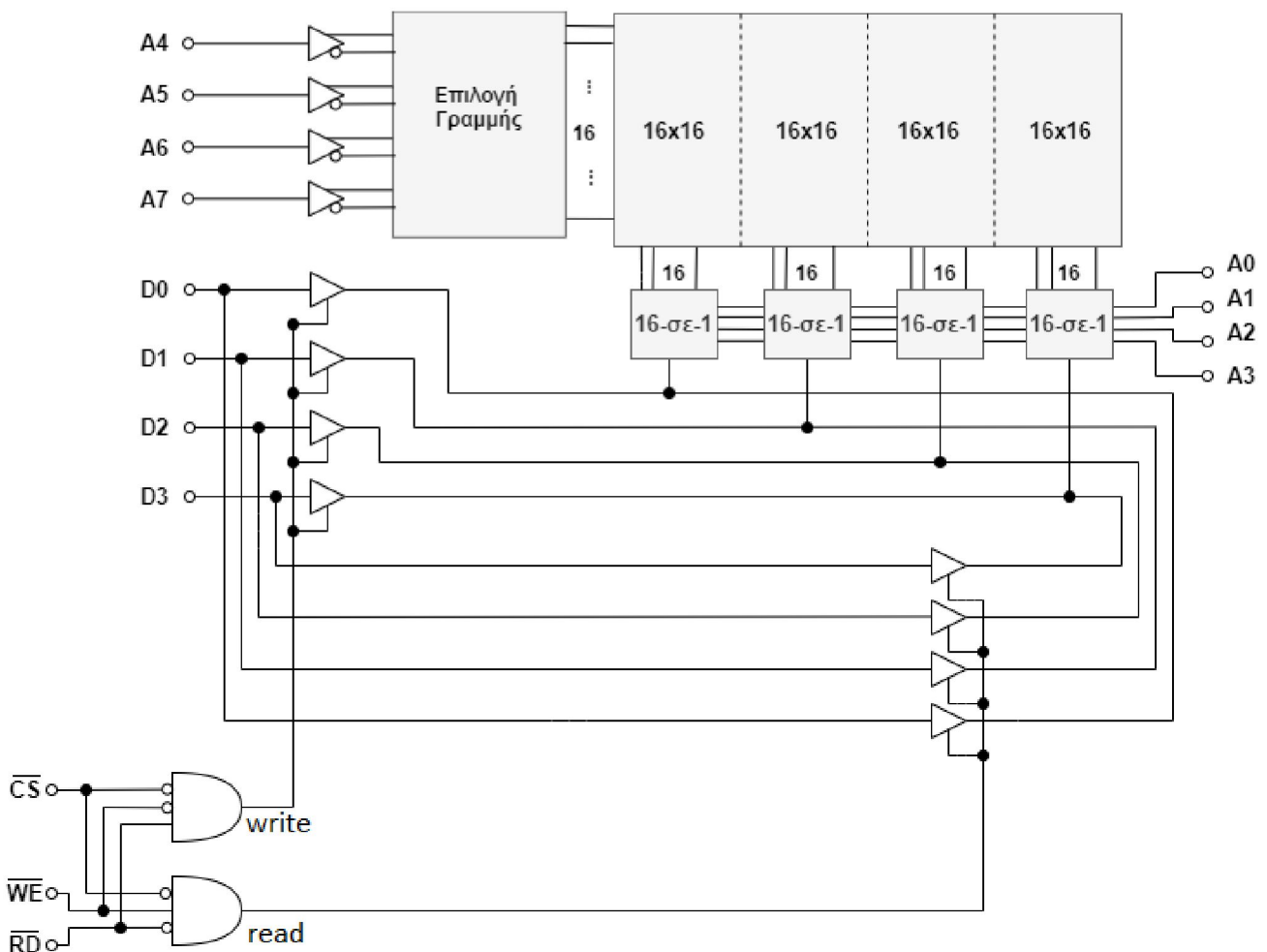
```
MVI D,00H
LDA 2000H
ANI 80H      ;isolate msb to get A3
RRC          ;shift and store in B
MOV B,A
LDA 2000H
ANI 40H      ;isolate bit 01000000 to get B3
ANA B        ;A3 AND B3
RRC          ;shift twice for the OR with (A2 AND B2)
RRC
MOV C,A
RRC          ;shift again to get X3 and store in D
MOV D,A
LDA 2000H
ANI 20H      ;isolate bit 00100000 to get A2
RRC          ;shift and store in B
MOV B,A
LDA 2000H
ANI 10H      ;isolate bit 00010000 to get B2
ANA B        ;A2 AND B2
ORA C        ;(A1 AND B1) OR (A2 AND B2)
RRC          ;shift twice to put in X2
RRC
ORA D        ;update D
MOV D,A
LDA 2000H
ANI 08H      ;isolate bit 00001000 to get A1
RRC          ;shift for XOR and store in B
MOV B,A
LDA 2000H
ANI 04H      ;isolate bit 00000100 to get B1
XRA B        ;A1 XOR B1
RRC          ;shift twice for the XOR with (A0 XOR B0)and store in C
RRC
MOV C,A
RLC          ;shift to put in X1
ORA D        ;update D
MOV D,A
LDA 2000H
ANI 02H      ;isolate bit 00000010 for A0
RRC          ;shift for XOR with B0 and store in B
MOV B,A
LDA 2000H
ANI 01H      ;isolate bit 00000001 for B0
XRA B        ;A0 XOR B0
XRA C        ;(A0 XOR B0) XOR X1
ORA D        ;update D
CMA
STA 3000H
JMP START
END
```

## Θεωρητικές Ασκήσεις

### 5<sup>η</sup> ΑΣΚΗΣΗ:

Η μνήμη SRAM 256x4 bit μπορεί να δεχθεί 256 λέξεις μεγέθους 4 bit η καθεμία. Τη χωρίζουμε σε 4 τμήματα μεγέθους 256 bits, ώστε στο καθένα να βρίσκεται ένα bit της λέξης. Για κάθε τμήμα διαστάσεις 16x16. Επομένως, για την επιλογή της στήλης χρειαζόμαστε  $\log_2 16 = 4$  ακροδέκτες διεύθυνσης A0-A3 και τέσσερις πολυπλέκτες 16-σε-1, ενώ για την επιλογή της γραμμής χρειαζόμαστε  $\log_2 16 = 4$  ακροδέκτες διεύθυνσης A4-A7 και έναν αποκωδικοποιητή 4-σε-16. Οι ακροδέκτες D0-D3 αποτελούν τους ακροδέκτες εισόδου/εξόδου. Τα σήματα  $\overline{CS}$ ,  $\overline{WE}$ ,  $\overline{RD}$ , καθορίζουν ποια λειτουργία (εγγραφή/ανάγνωση) θα επιτελεστεί ενεργοποιώντας ή απενεργοποιώντας τους κατάλληλους τρισταθείς buffers.

Παρουσιάζουμε της εσωτερική οργάνωση της μνήμης SRAM 256x4 bit.



### Παράδειγμα: λειτουργία ανάγνωσης:

Έστω ότι θέλουμε να διαβάζουμε τη λέξη που βρίσκεται στη θέση μνήμης με διεύθυνση 00101100.

Για να ενεργοποιηθεί η λειτουργία μνήμης πρέπει να εφαρμοστεί ένας αρνητικός παλμός στον ακροδέκτη CS. Η λειτουργία ανάγνωσης (και η αποτροπή εγγραφής) σηματοδοτείται με την εφαρμογή αρνητικού παλμού στον ακροδέκτη RD και θετικού παλμού στον ακροδέκτη WS. Με αυτόν τον τρόπο η έξοδος της κάτω πύλης AND γίνεται 1, επιτρέποντας τη διέλευση δεδομένων από τους τρισταθείς buffers που μεταφέρουν τα δεδομένα από τους πολυπλέκτες προς τα D0-D3, ενώ η έξοδος τις άνω πύλης AND γίνεται 0 θέτοντας σε κατάσταση υψηλής αντίστασης τους buffers που μεταφέρουν δεδομένα από τα D0-D3 προς τους πολυπλέκτες, δηλαδή αποτρέποντας τη διέλευση από αυτούς.

Η διεύθυνση από την οποία θέλουμε να διαβάσουμε τίθεται ως είσοδος στους ακροδέκτες A0-A7. Μέσω των A4-A7 και του αποκωδικοποιητή επιλέγεται η γραμμή που αντιστοιχεί στο 0010 (2 στο δεκαδικό). Μέσω των A0-A3 επιλέγεται και περνάει στην έξοδο κάθε πολυπλέκτη το bit που βρίσκεται στη στήλη που αντιστοιχεί στο 1100 (12 στο δεκαδικό). Στη συνέχεια τα bits αυτά διέρχονται από τους buffers και οδηγούνται στις εξόδους D0-D3.

Δηλαδή, με αυτόν τον τρόπο διαβάζουμε 4 bits, ένα από κάθε τμήμα μνήμης. Καθένα βρίσκεται στη 2<sup>η</sup> γραμμή και 12<sup>η</sup> στήλη του αντίστοιχου τμήματος.

### Παράδειγμα: λειτουργία εγγραφής:

Έστω ότι θέλουμε να εγγράψουμε τη λέξη 0110 στη διεύθυνση 10111101.

Για να ενεργοποιηθεί η λειτουργία μνήμης πρέπει να εφαρμοστεί ένας αρνητικός παλμός στον ακροδέκτη CS. Η λειτουργία εγγραφής (και η αποτροπή ανάγνωσης) σηματοδοτείται με την εφαρμογή θετικού παλμού στον ακροδέκτη RD και αρνητικού παλμού στον ακροδέκτη WS. Με αυτόν τον τρόπο η έξοδος της άνω πύλης AND γίνεται 1, επιτρέποντας τη διέλευση δεδομένων από τους τρισταθείς buffers που μεταφέρουν τα δεδομένα από τα D0-D3 προς τους πολυπλέκτες, ενώ η έξοδος τις άνω πύλης AND γίνεται 0 θέτοντας σε κατάσταση υψηλής αντίστασης τους buffers που μεταφέρουν δεδομένα από τους πολυπλέκτες προς τα D0-D3, δηλαδή αποτρέποντας τη διέλευση από αυτούς.

Κάθε bit της προς εγγραφή λέξης εφαρμόζεται στον αντίστοιχο ακροδέκτη εισόδου D0-D3 και φτάνει μέσω του buffer στον αντίστοιχο πολυπλέκτη. Η διεύθυνση εγγραφής τίθεται ως είσοδος στους ακροδέκτες A0-A7. Μέσω των A4-A7 και του αποκωδικοποιητή επιλέγεται η γραμμή που αντιστοιχεί στο 1011 (11 στο δυαδικό). Μέσω A0-A3 κάθε bit που φτάνει στους πολυπλέκτες οδηγείται στη στήλη που αντιστοιχεί στο 1101 (13 στο δυαδικό) του αντίστοιχου τμήματος μνήμης.

Δηλαδή, με αυτόν τον τρόπο το bit της λέξης που τίθεται στον D0 εγγράφεται στο πρώτο εξ αριστερών τμήμα μνήμης στην 11<sup>η</sup> γραμμή και 13<sup>η</sup> στήλη, το bit της λέξης που τίθεται στον D1 εγγράφεται στο δεύτερο τμήμα μνήμης στην 11<sup>η</sup> γραμμή και 13<sup>η</sup> στήλη, κ.ο.κ.

Σε περίπτωση που σταλεί αρνητικός παλμός ταυτόχρονα στα WE και WD όλοι οι buffers τίθενται σε κατάσταση υψηλής αντίστασης αποτρέποντας τη διέλευση δεδομένων, προστατεύοντας με αυτόν τον τρόπο την πληροφορία.

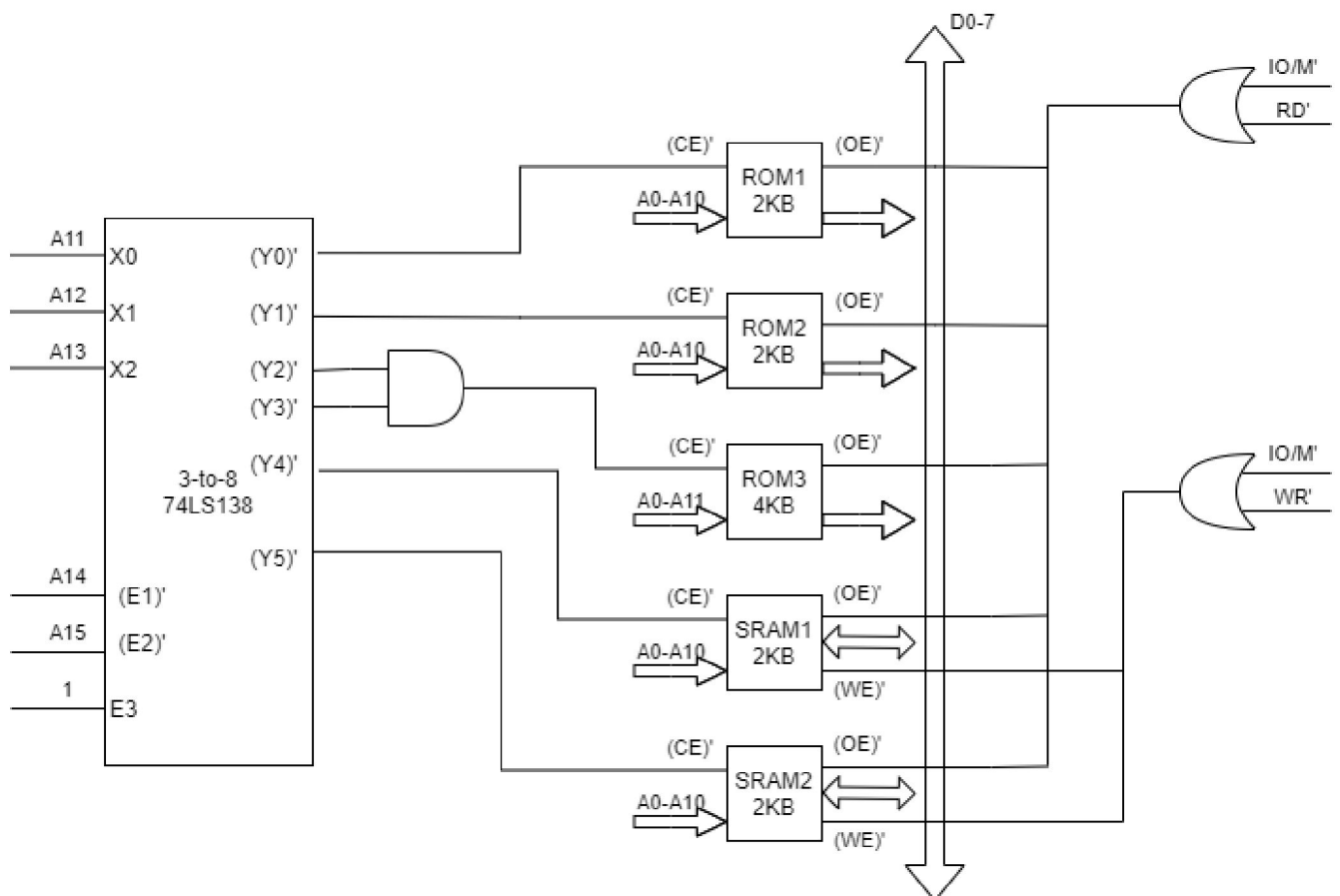
## 6<sup>η</sup> ΑΣΚΗΣΗ

Χάρτης μνήμης (2KB page, 6 pages):

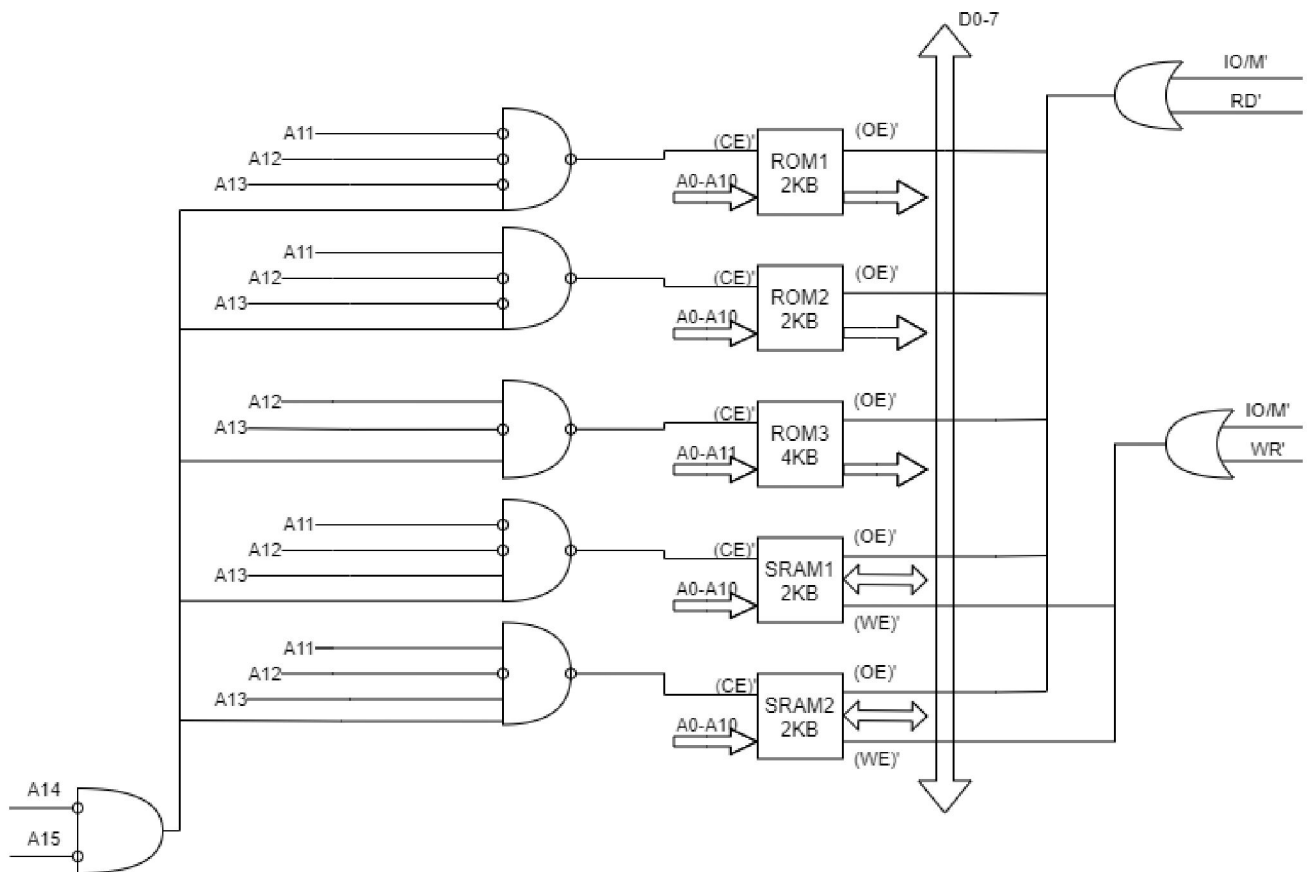
Memory	Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROM1 2K	0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	07FF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
ROM2 2K	0800	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
ROM3 4K	1000	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	1FFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
SRAM1 2K	2000	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	27FF	0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1
SRAM2 2K	2800	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
	2FFF	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1

Λογικά διαγράμματα:

(α) Με αποκωδικοποιητή:



**(β) Μόνο με πύλες:**



## 7<sup>η</sup> ΑΣΚΗΣΗ

Χάρτης μνήμης (4KB page, 7 pages):

Memory	Address	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ROM1 12K	0000	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0
	2FFF	0 0 1 0	1 1 1 1	1 1 1 1	1 1 1 1
RAM1 4K	3000	0 0 1 1	0 0 0 0	0 0 0 0	0 0 0 0
	3FFF	0 0 1 1	1 1 1 1	1 1 1 1	1 1 1 1
RAM2 4K	4000	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0
	4FFF	0 1 0 0	1 1 1 1	1 1 1 1	1 1 1 1
RAM3 4K	5000	0 1 0 1	0 0 0 0	0 0 0 0	0 0 0 0
	5FFF	0 1 0 1	1 1 1 1	1 1 1 1	1 1 1 1
ROM2 4K	6000	0 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0
	6FFF	0 1 1 0	1 1 1 1	1 1 1 1	1 1 1 1

Λογικό διάγραμμα:

