Συστήματα Μικροϋπολογιστών

2η Ομάδα Ασκήσεων

Ομάδα:

Κατσιάνης Κωνσταντίνος - 03120183

Κουρής Γεώργιος - 03120116

6ο Εξάμηνο - Μάιος 2023

Άσκηση 1:

(α)

Το πρόγραμμα για το ερώτημα (α) είναι το εξής:

```
       IN 10H        ;deactivating memory protection
       MVI A,00H     ;move immidiate 0 (in hex) to accumulator
       LXI H,0900H   ;load address 0900H into the HL register pair
       MOV M,A       ;move accumulator to address pointed by HL

SAVE_VALUE:
       INX H         ;add 1 to H
       INR A         ;add 1 to the accumulator
       MOV M,A       ;move accumulator to address pointed by HL
       CPI 7FH       ;true if accmulator < 127
       JNZ SAVE_VALUE    ;if true jump to SAVE_VALUE
END
```

Παρακάτω βλεπουμε τα αποθηκευμένα δεδομένα στην RAM:

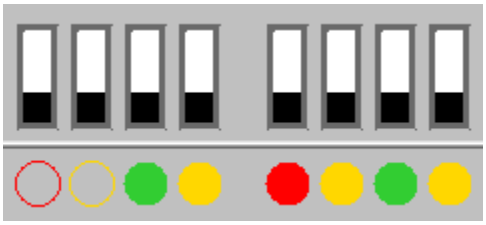
08FC	00	08FD	00	08FE	00	08FF	00	0900	00	0901	01	0902	02	0903	03	0904	04
0905	05	0906	06	0907	07	0908	08	0909	09	090A	0A	090B	0B	090C	0C	090D	0D
090E	0E	090F	0F	0910	10	0911	11	0912	12	0913	13	0914	14	0915	15	0916	16
0917	17	0918	18	0919	19	091A	1A	091B	1B	091C	1C	091D	1D	091E	1E	091F	1F
0920	20	0921	21	0922	22	0923	23	0924	24	0925	25	0926	26	0927	27	0928	28
0929	29	092A	2A	092B	2B	092C	2C	092D	2D	092E	2E	092F	2F	0930	30	0931	31
0932	32	0933	33	0934	34	0935	35	0936	36	0937	37	0938	38	0939	39	093A	3A
093B	3B	093C	3C	093D	3D	093E	3E	093F	3F	0940	40	0941	41	0942	42	0943	43
0944	44	0945	45	0946	46	0947	47	0948	48	0949	49	094A	4A	094B	4B	094C	4C
094D	4D	094E	4E	094F	4F	0950	50	0951	51	0952	52	0953	53	0954	54	0955	55
0956	56	0957	57	0958	58	0959	59	095A	5A	095B	5B	095C	5C	095D	5D	095E	5E
095F	5F	0960	60	0961	61	0962	62	0963	63	0964	64	0965	65	0966	66	0967	67
0968	68	0969	69	096A	6A	096B	6B	096C	6C	096D	6D	096E	6E	096F	6F	0970	70
0971	71	0972	72	0973	73	0974	74	0975	75	0976	76	0977	77	0978	78	0979	79
097A	7A	097B	7B	097C	7C	097D	7D	097E	7E	097F	7F	0980	00	0981	00	0982	00

(β)

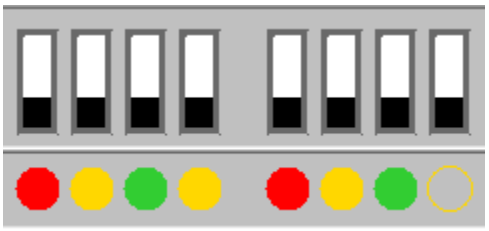
Συμπληρώνουμε το πρόγραμμα του ερωτήματος (α) για να ικανοποιούνται τα ζητούμενα του (β) ερωτήματος.

```
       IN 10H        ;deactivating memory protection
       MVI A,00H     ;move immidiate 0 (in hex) to accumulator
       LXI H,0900H   ;load address 0900H into the HL register pair
       LXI B,0000H   ;initialise BC (ones)
       MOV M,A       ;move accumulator to address pointed by HL

SAVE_VALUE:
       INX H         ;add 1 to H
       INR A         ;add 1 to the accumulator
       MOV M,A       ;move accumulator to address pointed by HL
       CPI 7FH       ;true if accmulator < 127
       JNZ SAVE_VALUE    ;if true jump to SAVE_VALUE

START_B:
       MOV A,M       ;move address pointed by HL to accumulator
       MVI D,09H     ;initialise E wih value 9 (for iteration)

NUM:
       RRC            ;rotate accumulator right
       DCR D          ;subtract 1 from D
       JZ CONTINUE    ;if D=0 continue
       JNC NUM        ;If carry flag = 1 then jump to NUM
       INX B          ;add 1 to the counter
       JMP NUM        ;jump to NUM

CONTINUE:
       RLC           ;rotate accumulator left because last was unneeded
       DCR L         ;decrease L
       JNZ START_B   ;if L != 0 jump to START_B
       MOV A,C       ;move C to the accumulator
       STA 3000H     ;save values and turn on the leds
END
```

Προσθέσαμε έξτρα κομμάτι στον κώδικα για να εμφανίζεται το πλήθος των ψηφίων ανάβοντας τα LEDs. Έχοντας χρησιμοποιήσει MOV A,C τα leds ανοίγουν με τον παρακάτω τρόπο.



Έχοντας χρησιμοποιήσει MOV A,B τα LEDs ανοίγουν με τον παρακάτω τρόπο.



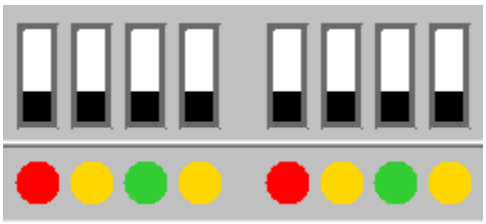
Οι τιμές είναι οι αναμενόμενες.

(γ)

Συμπληρώνοντας και πάλι τον κώδικα του (α) ερωτήματος για να ικανοποιούνται τα ζητούμενα του (γ) παίρνουμε:

```
       IN 10H        ;deactivating memory protection
       MVI A,00H     ;move immidiate 0 (in hex) to accumulator
       LXI H,0900H   ;load address 0900H into the HL register pair
       MOV M,A       ;move accumulator to address pointed by HL

SAVE_VALUE:
       INX H         ;add 1 to H
       INR A         ;add 1 to the accumulator
       MOV M,A       ;move accumulator to address pointed by HL
       CPI 7FH       ;true if accmulator < 127
       JNZ SAVE_VALUE    ;if true jump to SAVE_VALUE

       MVI E,7FH     ;load value 7FH to E
       MVI D,00H     ;load value 0 to D
       MOV A,M       ;move address pointed by HL to the accumulator

CHECK_BOUNDARIES:
       CPI 61H
       JNC CASE_FALSE  ;jump to CASE_FALSE if accumulator > 61H
       CPI 10H
       JC CASE_FALSE   ;jump to CASE_FALSE if accumulator < 10H
       INR D           ;incread D by 1

CASE_FALSE:
       INR L         ;increase L by 1
       MOV A,M       ;move address pointed by HL to the accumulator
       DCR E         ;decrease E by 1
       JNZ CHECK_BOUNDARIES   ;if E != 0 jump to CHEST_BOUNDARIES
       CPI 61H
       JNC FINISH       ;jump to FINISH if accumulator > 61H
       CPI 10H
       JC FINISH       ;jump to FINISH if accumulator < 10H
       INR D         ;increase D by 1
FINISH:
       MOV A,D       ;move D to the accumulator
       STA 3000H     ;turn on the leds
       END
```

Τα LEDs είναι:



Άσκηση 2:

Το πρόγραμμα φαίνεται παρακάτω:

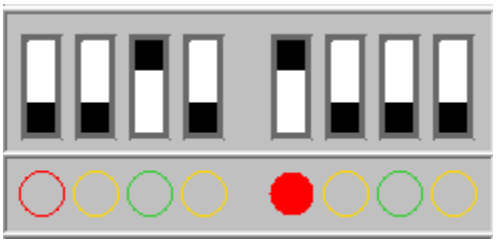
```
       MVI D,C8H     ;D = 200 since we want 200/10 = 20 seconds
       MVI A,FFH     ;we set the accumulator to off as the default
       LXI B,0064H   ;delay of 1/10 seconds
BEGIN:
       LDA 2000H     ;load the input from dip switches to the accumulator
       RLC           ;rotate accumulator left to check MSB
       JNC OFF1      ;if MSB == off is off jump to OFF1
       JMP BEGIN     ;jump to BEGIN
ON1:
       LDA 2000H     ;load the input from dip switches to the accumulator
       RLC           ;rotate accumulator left to check MSB
       JNC OFF2      ;jump to OFF2 if MSB = off (off-on-off)
       JMP ON1       ;jump to ON1
OFF1:
       LDA 2000H     ;load the input from dip switches to the accumulator
       RLC           ;rotate accumulator left to check MSB
       JC ON1        ;if MSB == on jump to ON1
       JMP OFF1      ;Else wait until it's on
OFF2:
       LDA 2000H     ;load the input from dip switches to the accumulator
       RLC           ;rotate accumulator left to check MSB
       JC ON2        ;if MSB == on jump to ON2
       MVI A,00H     ;else set accumulator to on
       STA 3000H     ;then turn on the leds
       CALL DELB     ;call the delb to start counting the seconds
       DCR D         ;decrease D by one
       JNZ OFF2      ;jump to OFF2 if D != 0
       MVI A,FFH     ;set accumulator to off
       STA 3000H     ;turn off the leds
       JMP OFF1      ;jump to OFF1
ON2:
       LDA 2000H     ;load the input from dip switches to the accumulator
       RLC           ;rotate accumulator left to check MSB
       JNC RESTART   ;if MSB != 0 jump to RESTART
       MVI A,00H     ;set accumulator to on
       STA 3000H     ;turn on the leds
       CALL DELB     ;call delb
       DCR D         ;decrease D by one
       JNZ ON2       ;jump to ON2 if D != 0
       MVI A,FFH     ;set accumulator to off
       STA 3000H     ;turn off the lights
       JMP OFF1      ;jump to OFF1
RESTART:
       MVI D,C8H     ;set D to 200 to reset the timer
       JMP OFF2      ;jump to OFF2

END
```

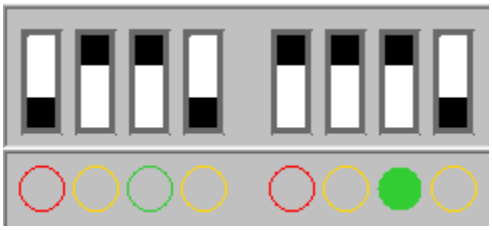
Άσκηση 3:

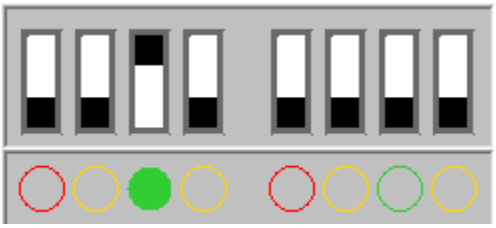
(i)

Το πρόγραμμα φαίνεται παρακάτω:

```
BEGIN:
       MVI D,00H      ;initialize D (D=0)
       MVI E,08H      ;initialize E (E=0)
       LDA 2000H      ;load status of switches
SEARCH:               ;we search for the right most switch that is on
       RRC            ;rotate accumuator right
       DCR E          ;decrease E by one
       JZ NULL        ;jump to NULL if E==0 (all switche off)
       INR D          ;increase D by one
       JNC SEARCH     ;jump to SEARCH if current switch off
       DCR D          ;decrease D by one
       MVI A,FEH      ;set value to accumulator
FOUND:                ;we found the switch!
       RLC            ;rotate accumulator left
       DCR D          ;decrease D by one
       JNZ FOUND      ;jump to FOUND if current switch off
       STA 3000H      ;save the values to the leds (turn them on)
       JMP BEGIN      ;jumpt to BEGIN
NULL:                 ;case that all swithces are off
       MVI A,FFH      ;set value to the accumulator
       STA 3000H      ;save the values to the leds (all are off)
       JMP BEGIN      ;jump to BEGIN
END
```



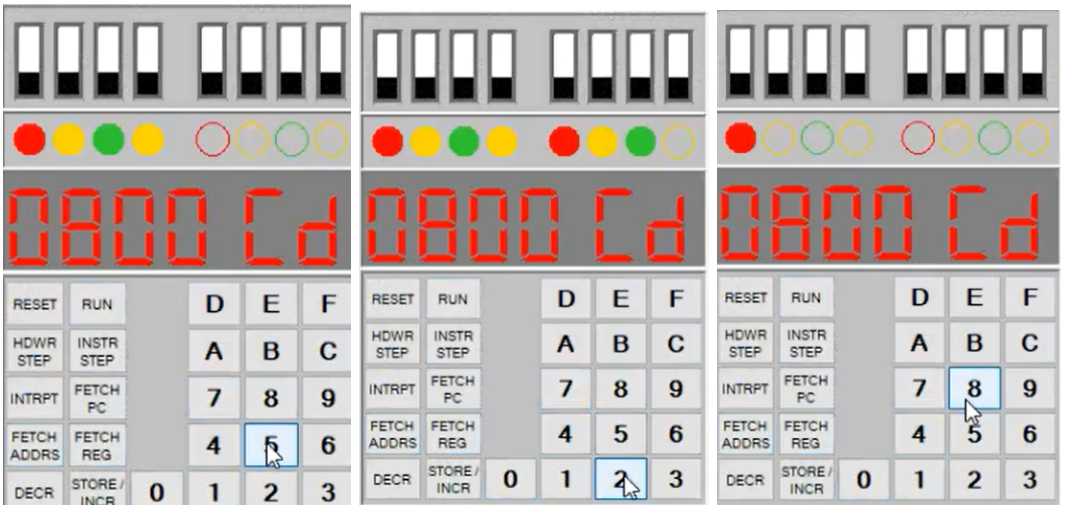




(ii)

Το πρόγραμμα φαίνεται παρακάτω:

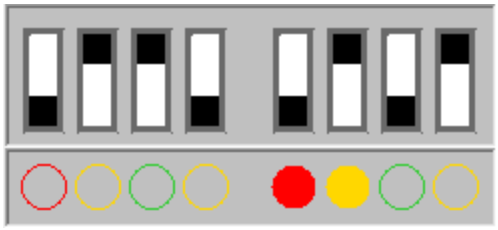
```
BEGIN:
       CALL KIND    ;call subroutine KIND from Î¾Lab notes
       CPI 00H      ;check if accumulator is zero
       JZ NULL      ;if accumulator is zero, go to NULL
       CPI 09H      ;Check if accumulator is equal to 9
       JNC NULL     ;if accumulator is greater than or equal to 9, go to NULL
       MOV E,A      ;save value of accumulator to E
       MVI A,00H    ;clear accumulator A-set to 0
       DCR E        ;decrease E by 1
       JZ TURN_ON   ;if register E is 0, jump to TURN_ON
       INR A        ;increase accumulator by 1

CHECK:
       DCR E        ;decrease E by 1
       JZ TURN_ON   ;if register E is zero, jump to TURN_ON
       RLC          ;rotate accumulator left
       INR A        ;increase accumulator by 1
       JMP CHECK    ;jump to CHECK

TURN_ON:
       STA 3000H    ;turn on the LEDs
       JMP BEGIN    ;jump to BEGIN to check again

NULL:                 ;case: button pressed not 1 to 8
       MVI A,FFH    ;load accumulator with value FFH->LEDs off
       STA 3000H    ;turn on the LEDs (they will be off)
       JMP BEGIN    ;jump to BEGIN to check again
END
```



Άσκηση 4:

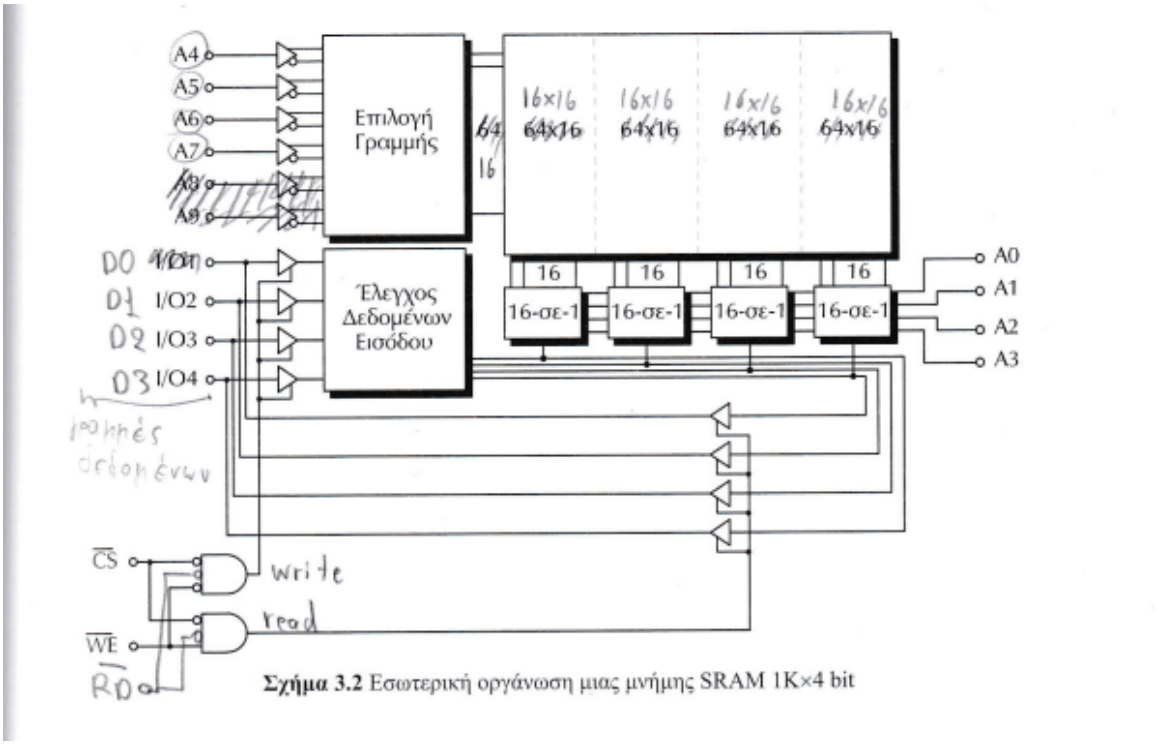
Το πρόγραμμα:

```
ASK_4:
       MVI D,00H     ;initialise D
       LDA 2000H     ;load the input from the switches to the accumulator
       MOV B,A       ;save value from accumulator to B
       ANI 01H       ;B0-save value to accumulator->accumulator AND 00000001
       MOV C,A       ;save accumulator to C
       MOV A,B       ;save B to the accumulator (switches values)
       ANI 02H       ;A0-save value to accumulator->accumulator AND 00000010
       RRC           ;rotate accumulator right
       ANA C         ;accumulator->accumulator AND C
       MOV D,A       ;save accumulator to D
       MOV A,B       ;save B to accumulator
       ANI 04H       ;B1-save value to accumulator->accumulator AND 00000100
       MOV C,A       ;save accumulator to C
       MOV A,B       ;save accumulator to B
       ANI 08H       ;A1-save value to accumulator->accumulator AND 00001000
       RRC           ;rotate accumulator right
       ANA C         ;accumulator->accumulator AND C
       RRC           ;rotate accumulator right
       MOV E,A       ;save accumulator to E
       RRC           ;rotate accumulator right
       ORA D         ;accumulator->accumulator OR D
       ORA E         ;accumulator->accumulator OR E
       MOV D,A       ;save accumulator to D
       MOV A,B       ;save B to accumulator
       ANI 10H       ;B2-save value to accumulator->accumulator AND 00010000
       MOV C,A       ;save accumulator to C
       MOV A,B       ;save B to accumulator
       ANI 20H       ;A2-save value to accumulator->accumulator AND 00100000
       RRC           ;rotate accumulator right
       XRA C         ;accumulator->accumulator XOR C
       MOV E,A       ;save accumulator to E
       MOV A,B       ;save B to accumulator
       ANI 40H       ;B3-save value to accumulator->accumulator AND 01000000
       MOV C,A       ;save accumulator to C
       MOV A,B       ;save B to accumulator
       ANI 80H       ;A3-save value to accumulator->accumulator AND 10000000
       RRC           ;rotate accumulator right
       XRA C         ;accumulator->accumulator XOR C
       RRC           ;rotate accumulator right
       RRC           ;rotate accumulator right
       MOV B,A       ;save accumulator to B
       RRC           ;rotate accumulator right
       ORA D         ;accumulator->accumulator OR D
       MOV D,A       ;save accumulator to D
       MOV A,B       ;save B to accumulator
       ORA E         ;accumulator->accumulator OR E
       RRC           ;rotate accumulator right
       RRC           ;rotate accumulator right
       ORA D         ;accumulator->accumulator OR D
       CMA           ;reverse logic (ls light the LEDs)
       STA 3000H     ;Turn on the correct LEDs
       JMP ASK_4     ;jump to the first line
END
```



Για είσοδο 0110 0101 παίρνουμε την αναμενόμενη έξοδο 0000 1100. Όπως ζητείται, τα 4 πρώτα MSB LEDs είναι πάντα off.

Άσκηση 5:



Σχήμα 3.2 Εσωτερική οργάνωση μιας μνήμης SRAM 1K×4 bit

Μετά από κατάλληλες αλλαγές στην αρχική εικόνα 3.2 του βιβλίου (με μολύβι) παίρνουμε την εσωτερική οργάνωση της μνήμης SRAM 2564 bit. Αρχικά, στην αποκωδικοποίηση έχουμε 16 bit αντί για 64, επομένως μας αρκούν μόνο 4 ψηφία αντί για 6 (A4 - A7 αντί για A4 - A9). Έτσι από τον πίνακα της μνήμης επιλέγεται μία από τις 16 γραμμές. Επίσης οι γραμμές δεδομένων μας είναι οι D0 έως D3. Η επιλογή της τετράδας που θα συνδεθεί στις 4 γραμμές των δεδομένων γίνεται με 4 πολυπλέκτες 16 σε 1 που ελέγχονται από τα bit A0-A3 της διεύθυνσης. Οι πολυπλέκτες υλοποιούνται με διακόπτες και επιτρέπουν τη διέλευση των δεδομένων και προς τις δύο κατευθύνσεις (είτε για ανάγνωση είτε για εγγραφή). Επίσης έχουμε ξεχωριστό σήμα για ανάγνωση δεδομένων, RD.

Για παράδειγμα, άμα έχουμε την διεύθυνση 0011 0111, τότε θα γίνει η επιλογή της 7ης γραμμής και 3ης τετράδας του πίνακα της μνήμης.

Άσκηση 6:

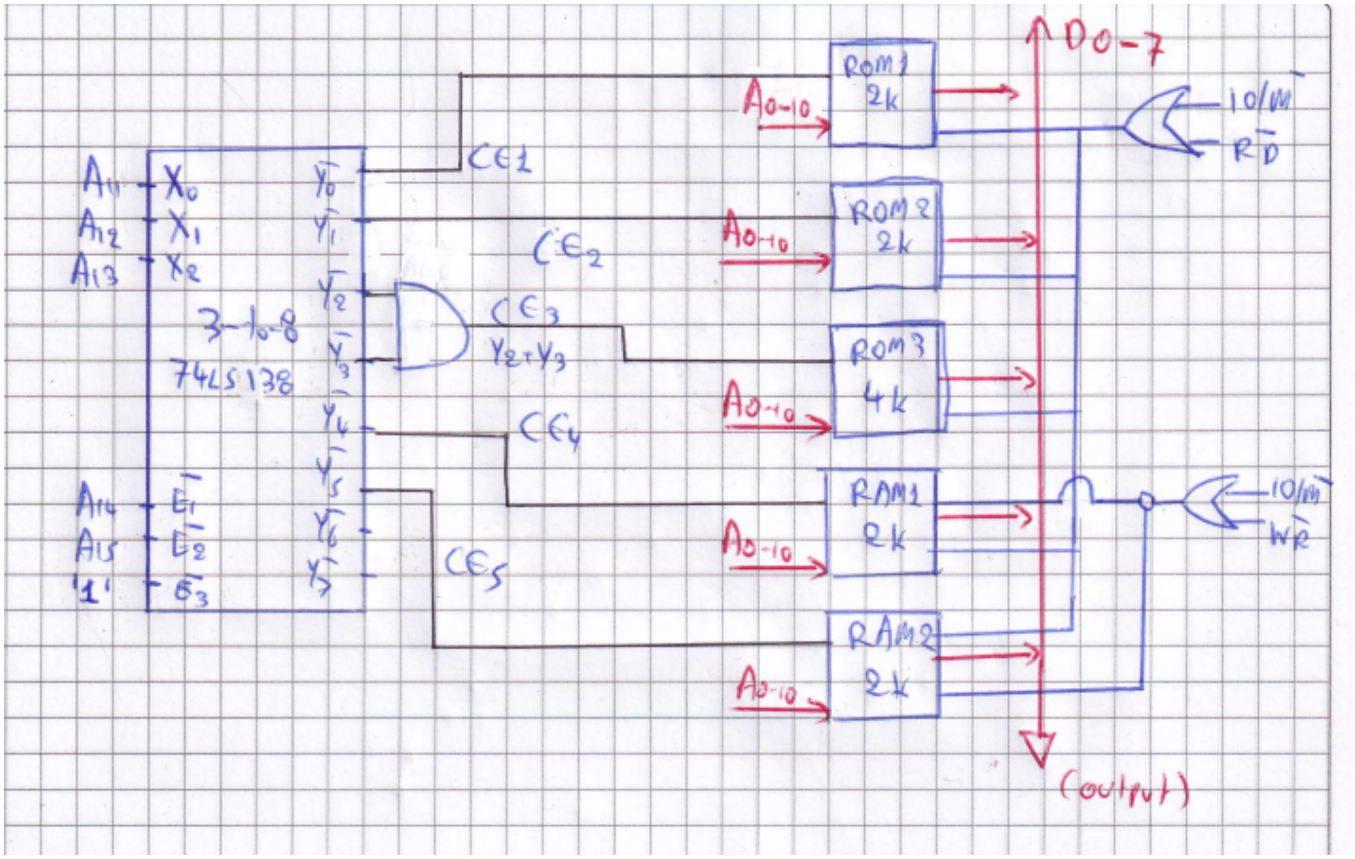
Συμπληρώνουμε τον χάρτη της μνήμης του συστήματος. Έχουμε ότι:

0000 – 1FFFH (8Kbytes) ROM

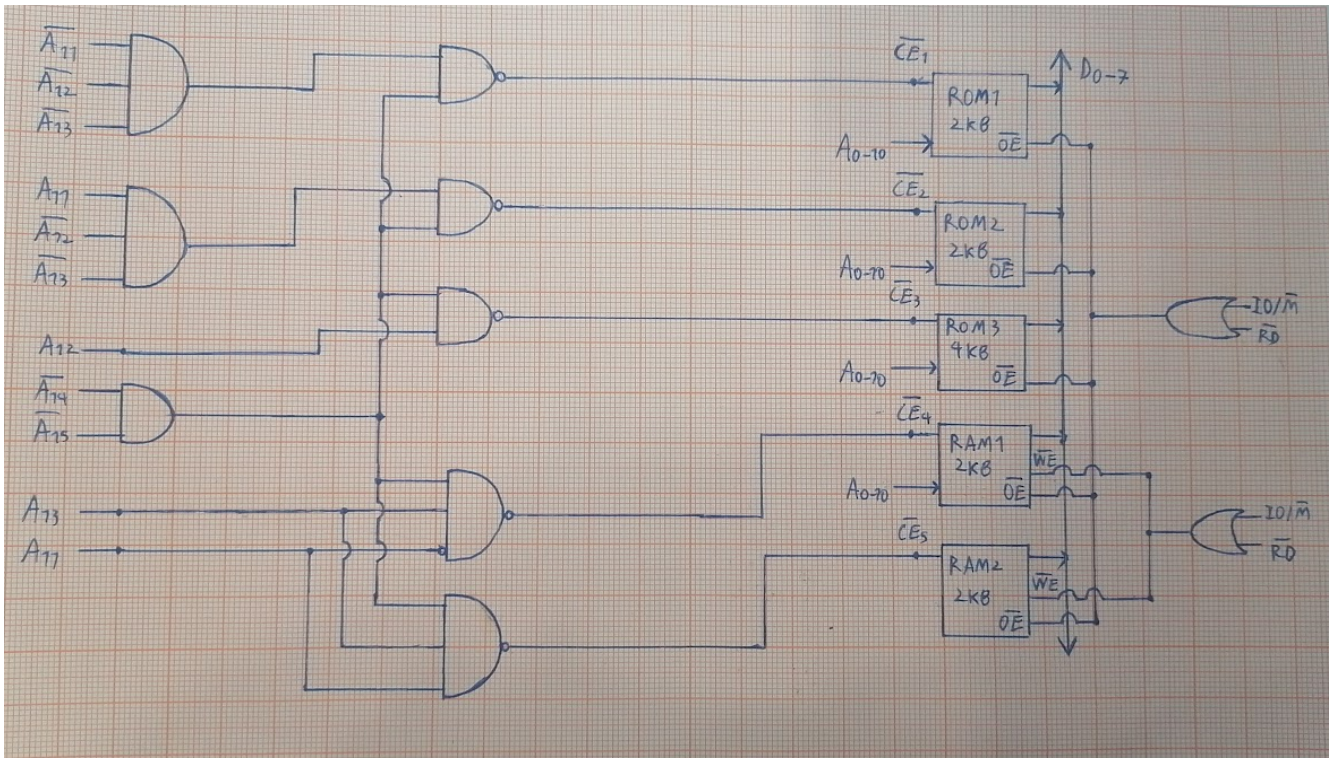
3000 – 2FFFH (4Kbytes) RAM

Μνήμη	Διεύθυνση	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
	0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
ROM2 2k	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
ROM3 4k	2000	0	0	1	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 2k	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 2k	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

(α)



(β)


A11
A12
A13
A11
A12
A13
A12
A14
A15
A13
A11
CE1
ROM1
2kB
OE
A0-10
D0-7
CE2
ROM2
2kB
OE
A0-10
CE3
ROM3
4kB
OE
A0-10
IO/M
RD
CE4
RAM1
2kB
WE
OE
A0-10
CE5
RAM2
2kB
WE
OE
IO/M
RD

Άσκηση 7:

Συμπληρώνουμε τον χάρτη της μνήμης του συστήματος. Έχουμε ότι:

0000-1FFF Hex : ROM (8Kbytes)

2000-4FFF Hex : RAM (12Kbytes)

5000-6FFF Hex : ROM (8Kbytes)

Μνήμη	Διεύθυνση	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROM1 8k	0000	0	0	0	0	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
RAM1 4k	2000	0	0	1	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
ROM2 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
RAM3 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
ROM2 8k	5000	0	1	0	1	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