Ε.Μ.Π. - ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
ΑΚΑΔ. ΕΤΟΣ 2019-2020

Πεγειώτη Νάταλυ
03117707

2η ΟΜΑΔΑ ΑΣΚΗΣΕΩΝ
ΓΙΑ ΤΟ ΜΑΘΗΜΑ "Συστήματα Μικροϋπολογιστών"

Ασκήσεις Προσομοίωσης

1η ΑΣΚΗΣΗ

Κώδικας σε assembly

```
        IN 10H          ;Άρση προστασίας της μνήμης
        LXI H,0900H     ;Αρχικοποίηση της διεύθυνσης μνήμης στη θέση 0900H
        MVI A,FFH       ;Αρχικοποίηση A=255

LOOP_A: MOV M,A         ;Αντιγραφή του περιεχομένου του A στη μνήμη
        INX H           ;Μετάβαση στην επόμενη θεση μνήμης
        DCR A           ;Μείωση του περιεχομένου του A κατά 1
        CPI 00H         ;Σύγκριση του A με το 0
        JNZ LOOP_A      ;Αν το A είναι διάφορο του 0, τότε άλμα στην ετικέτα
                        LOOP_A
        MOV M,A         ;Αντιγραφή του 0 στη μνήμη


PART_B: LXI H,09FFH     ;Αρχικοποίηση της διεύθυνσης μνήμης στη θέση 09FFH
        LXI D,0000H     ;Αρχικοποίηση του DE με 0

NXT_NUM: MOV A,M        ;Αντιγραφή του περιεχομένου της μνήμης στον A
        MVI B,09H       ;Ορισμός B = 9, έτσι ώστε να πραγματοποιηθούν
                        8 επαναλήψεις

NXT_BIT: DCR B          ;Μειώση μετρητή B
        JZ MEM          ;Αν B = 0 (8 επαναλήψεις), τότε άλμα στην ετικέτα MEM
        RRC             ;Δεξιά περιστροφή και μετακίνηση bit στο CY για έλεγχο
        JC NXT_BIT      ;Αν CY = 1, τότε άλμα στην ετικέτα NXT_BIT
        INX D           ;Αύξηση του D κατά 1
        JMP NXT_BIT     ;Άλμα στην ετικέτα NXT_BIT

MEM:    DCR L           ;Μετάβαση στην προηγούμενη θέση μνήμης
        JNZ NXT_NUM     ;Αν η νέα θέση μνήμης είναι διάφορη της 0900H, τότε άλμα
                        στην ετικέτα NXT_NUM
```

```
PART_C:     MVI C,00H          ;Αρχικοποίηση του καταχωρητή C με 0
            LXI H,09FFH        ;Αρχικοποίηση της διεύθυνσης μνήμης στη θέση 09FFH

COMPARE:    MOV A,M            ;Αντιγραφή του περιεχομένου της μνήμης στον A
            CPI 70H            ;Σύγκριση του περιεχομένου του A με το 70H
            JZ COUNT           ;Αν A=70H, τότε άλμα στην ετικέτα COUNT
            JNC NXT_NUM_C      ;Αν A>70H, τότε άλμα στην ετικέτα NXT_NUM_C
            CPI 20H            ;Σύγκριση του περιεχομένου του A με το 20H
            JNC COUNT          ;Αν A>=20H, τότε άλμα στην ετικέτα COUNT
            JMP NXT_NUM_C      ;Άλμα στην ετικέτα NXT_NUM_C

COUNT:      INR C              ;Αύξηση του μετρητή C, αφού ο αριθμός A περιλαμβάνεται
                               στα όρια

NXT_NUM_C:  DCR L              ;Μετάβαση στην προηγούμενη θέση μνήμης
            JNZ COMPARE        ;Αν η νέα θέση μνήμης είναι διάφορη της 0900H, τότε
                               άλμα στην ετικέτα COMPARE

            XCHG               ; Ανταλλαγή του περιεχομένου των  DE και HL
            SHLD 0AAAH         ; Αποθήκευση του HL αρχίζοντας από τη θέση 0AAAH
            MOV A,C            ; Αντιγραφή του περιεχομένου του C στον A
            STA 0AACH          ; Αποθήκευση του A στη θέση 0AACH

END
```

(α) Μετά την εκτέλεση του προγράμματος η μνήμη έχει την ακόλουθη μορφή:

08FA	00	08FB	00	08FC	00	08FD	00	08FE	00	08FF	00	0900	FF	0901	FE	0902	FD	0903	FC
0904	FB	0905	FA	0906	F9	0907	F8	0908	F7	0909	F6	090A	F5	090B	F4	090C	F3	090D	F2
090E	F1	090F	F0	0910	EF	0911	EE	0912	ED	0913	EC	0914	EB	0915	EA	0916	E9	0917	E8
0918	E7	0919	E6	091A	E5	091B	E4	091C	E3	091D	E2	091E	E1	091F	E0	0920	DF	0921	DE
0922	DD	0923	DC	0924	DB	0925	DA	0926	D9	0927	D8	0928	D7	0929	D6	092A	D5	092B	D4
092C	D3	092D	D2	092E	D1	092F	D0	0930	CF	0931	CE	0932	CD	0933	CC	0934	CB	0935	CA
0936	C9	0937	C8	0938	C7	0939	C6	093A	C5	093B	C4	093C	C3	093D	C2	093E	C1	093F	C0
0940	BF	0941	BE	0942	BD	0943	BC	0944	BB	0945	BA	0946	B9	0947	B8	0948	B7	0949	B6
094A	B5	094B	B4	094C	B3	094D	B2	094E	B1	094F	B0	0950	AF	0951	AE	0952	AD	0953	AC
0954	AB	0955	AA	0956	A9	0957	A8	0958	A7	0959	A6	095A	A5	095B	A4	095C	A3	095D	A2
095E	A1	095F	A0	0960	9F	0961	9E	0962	9D	0963	9C	0964	9B	0965	9A	0966	99	0967	98
0968	97	0969	96	096A	95	096B	94	096C	93	096D	92	096E	91	096F	90	0970	8F	0971	8E
0972	8D	0973	8C	0974	8B	0975	8A	0976	89	0977	88	0978	87	0979	86	097A	85	097B	84
097C	83	097D	82	097E	81	097F	80	0980	7F	0981	7E	0982	7D	0983	7C	0984	7B	0985	7A
0986	79	0987	78	0988	77	0989	76	098A	75	098B	74	098C	73	098D	72	098E	71	098F	70
0990	6F	0991	6E	0992	6D	0993	6C	0994	6B	0995	6A	0996	69	0997	68	0998	67	0999	66
099A	65	099B	64	099C	63	099D	62	099E	61	099F	60	09A0	5F	09A1	5E	09A2	5D	09A3	5C
09A4	5B	09A5	5A	09A6	59	09A7	58	09A8	57	09A9	56	09AA	55	09AB	54	09AC	53	09AD	52
09AE	51	09AF	50	09B0	4F	09B1	4E	09B2	4D	09B3	4C	09B4	4B	09B5	4A	09B6	49	09B7	48
09B8	47	09B9	46	09BA	45	09BB	44	09BC	43	09BD	42	09BE	41	09BF	40	09C0	3F	09C1	3E
09C2	3D	09C3	3C	09C4	3B	09C5	3A	09C6	39	09C7	38	09C8	37	09C9	36	09CA	35	09CB	34
09CC	33	09CD	32	09CE	31	09CF	30	09D0	2F	09D1	2E	09D2	2D	09D3	2C	09D4	2B	09D5	2A
09D6	29	09D7	28	09D8	27	09D9	26	09DA	25	09DB	24	09DC	23	09DD	22	09DE	21	09DF	20
09E0	1F	09E1	1E	09E2	1D	09E3	1C	09E4	1B	09E5	1A	09E6	19	09E7	18	09E8	17	09E9	16
09EA	15	09EB	14	09EC	13	09ED	12	09EE	11	09EF	10	09F0	0F	09F1	0E	09F2	0D	09F3	0C
09F4	0B	09F5	0A	09F6	09	09F7	08	09F8	07	09F9	06	09FA	05	09FB	04	09FC	03	09FD	02
09FE	01	09FF	00	0A00	00	0A01	00	0A02	00	0A03	00	0A04	00	0A05	00	0A06	00	0A07	00

(β) Ο συνολικός αριθμός μηδενικών των παραπάνω δεδομένων υπολογίστηκε όπως αναμενόταν 0400H, δηλαδή 1024.

0AAA	00	0AAB	04

(γ) Το πλήθος των αριθμών μεταξύ 20H και 70H συμπεριλαμβανομένων υπολογίστηκε όπως αναμενόταν 51H, δηλαδή 81.

0AAC	51

2η ΑΣΚΗΣΗ

Κώδικας σε assembly

```
            IN 10H          ; Άρση προστασίας της μνήμης
            LXI B,00C8H     ; Καταχώρηση BC = 200 (200*1ms=0,2s)

START:      MVI D,4BH       ; Αρχικοποίηση του καταχωρητή D = 75 (75*200*1ms=15s)
            MVI E,00H       ; Αρχικοποίηση του καταχωρητή E = 00H
            MVI A,FFH       ; Αρχικοποίηση του καταχωρητή A = FFH
            STA 3000H       ; Αρχικοποίηση της κατάστασης των LED (διεύθυνση 3000H)

CHECK1:     CALL DELB       ; Προσθήκη delay = BC*1ms = 200*10^-3 = 0,2s
            LDA 2000H       ; Φόρτωση του περιεχομένου της θέσης 2000H (dip switch) στον A
            ANI 01H         ; Απομόνωση του LSB του A και καταχώρηση του στον ίδιο
            CPI 01H         ; Σύγκριση του A με 1, αν ισούνται τότε Z = 1, αλλιώς  Z = 0
            JNZ CHECK1      ; Αν Z = 0, τότε άλμα στην ετικέτα CHECK1 (μέχρι να γίνει 1)

CHECK2:     CALL DELB       ; Προσθήκη delay = BC*1ms = 200*10^-3 = 0,2s
            LDA 2000H       ; Φόρτωση του περιεχομένου της θέσης 2000H (dip switch) στον A
            ANI 01H         ; Απομόνωση του LSB του A και καταχώρηση του στον ίδιο
            CPI 00H         ; Σύγκριση του A με 0, αν ισούνται τότε Z = 1, αλλιώς  Z = 0
            JNZ CHECK2      ; Αν Z = 0, τότε άλμα στην ετικέτα CHECK1 (μέχρι να γίνει 0)


LIGHTS1:    CALL DELB       ; Προσθήκη delay = BC*1ms = 200*10^-3 = 0,2s
            LDA 2000H       ; Φόρτωση του περιεχομένου της θέσης 2000H (dip switch) στον A
            ANI 01H         ; Απομόνωση του LSB του A και καταχώρηση του στον ίδιο
            CPI 01H         ; Σύγκριση του A με 1, αν ισούνται τότε Z = 1, αλλιώς  Z = 0
            JZ LIGHTS2      ; Αν Z = 1,τότε άλμα στην ετικέτα LIGHTS2

            MOV A,E         ; Καταχώρηρη του E (προηγούμενη κατάσταση) στον A
            CMA             ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
            STA 3000H       ; Ενημέρωση της κατάστασης των LED  (διεύθυνση 3000H)
            MOV E,A         ; Καταχώρηρη του E (τρέχουσα κατάσταση) στον E
            DCR D           ; Μείωση του D κατα 1
            JNZ LIGHTS1     ; Αν το D είναι διάφορο του 0, τότε άλμα στην ετικέτα LIGHTS1

            JMP START       ; Άλμα στην ετικέτα START


LIGHTS2:    CALL DELB       ; Προσθήκη delay = BC*1ms = 200*10^-3 = 0,2s
            LDA 2000H       ; Φόρτωση του περιεχομένου της θέσης 2000H (dip switch) στον A
            ANI 01H         ; Απομόνωση του LSB του A και καταχώρηση του στον ίδιο
            CPI 00H         ; Σύγκριση του A με 0, αν ισούνται τότε Z = 1, αλλιώς  Z = 0
            JNZ LIGHTS3     ; Αν Z = 0, τότε άλμα στην ετικέτα LIGHTS3
            MVI D,4BH       ; Αρχικοποίηση του καταχωρητή D = 75 (75*200*1ms=15s)
            JMP LIGHTS1     ; Άλμα στην ετικέτα LIGHTS
```

```
LIGHTS3:  MOV A,E     ; Καταχώρηρη του E (προηγούμενη κατάσταση) στον A
          CMA         ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
          STA 3000H   ; Ενημέρωση της κατάστασης των LED (διεύθυνση 3000H)
          MOV E,A     ; Καταχώρηρη του A (τρέχουσα κατάσταση) στον E
          DCR D       ; Μείωση του D κατα 1
          JNZ LIGHTS2 ; Αν το D είναι διάφορο του 0, τότε άλμα στην ετικέτα LIGHTS2

          JMP START   ; Άλμα στην ετικέτα START

HLT

END
```

3η ΑΣΚΗΣΗ

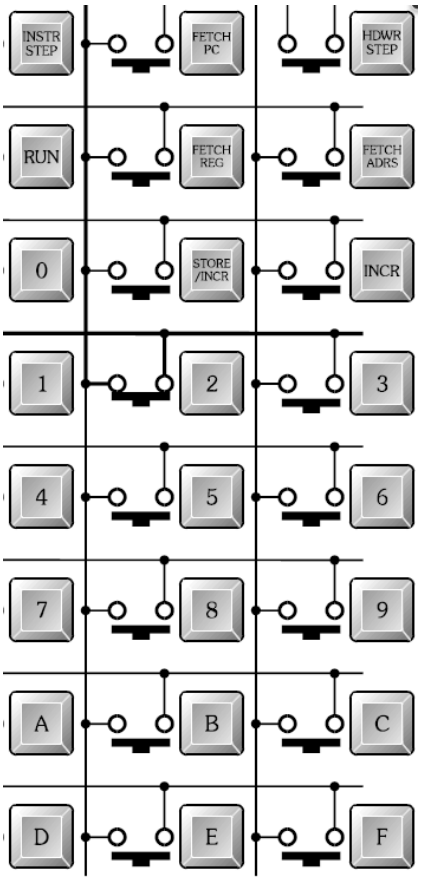
i. Κώδικας σε assembly

```
            IN 10H          ; Άρση προστασίας της μνήμης

START:      LDA 2000H       ; Φόρτωση του περιεχομένου της θέσης 2000H (dip switch) στον A
            MVI C,80H       ; Αρχικοποίηση του C (έξοδος) με την τιμή 80H (10000000)
            MVI D,08H       ; Αρχικοποίηση του D (μετρητής) με την τιμή 08H

REPEAT:     RLC             ; Αριστερή περιστροφή του A και αποθήκευση του MSB στον CY
            JC OUT          ; Αν CY = 1, τότε άλμα στην ετικέτα OUT
            MOV B,A         ; Αντιγραφή του περιεχομένου του A στον B
            MOV A,C         ; Αντιγραφή του περιεχομένου του C (έξοδος) στον A
            INR A           ; Αύξηση του A κατά 1
            RRC             ; Δεξιά περιστροφή του A
            MOV C,A         ; Αποθήκευση του A στον C (έξοδος)
            MOV A,B         ; Επαναφορά της τιμής του B στον A
            DCR D           ; Μείωση της τιμή του μετρητή D κατά 1
            JZ OUT          ; Αν ο D είναι ίσος με 0, τότε άλμα στην ετικέτα OUT
            JMP REPEAT      ; Άλμα στην ετικέτα REPEAT

OUT:        MOV A,C         ; Μεταφορά του περιεχομένου του C (έξοδος) στον A
            CMA             ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
            STA 3000H       ; Ενημέρωση της κατάστασης των LED (διεύθυνση 3000H)
            JMP START       ; Άλμα στην ετικέτα START

END
```

ii. Κώδικας σε assembly

```
        IN 10H          ; Άρση προστασίας της μνήμης
        LXI B,01F4H     ; Καταχώρηση BC = 500 (500*1ms=0,5s)

START:  MVI D,04H       ; Αρχικοποίηση του καταχωρητή D = 4
        CALL KIND       ; Διάβασμα αριθμού από το πληκτρολόγιο και καταχώρηση
                        του αντίστοιχου κωδικού στον A
        CPI 00H         ; Σύγκριση του A με το 0, αν ισούνται τότε Z=1, αλλιώς Z=0
        JZ START        ; Αν Z = 1, τότε άλμα στην ετικέτα START
        CPI 05H         ; Σύγκριση του A με το 5, αν A<5 τότε CY=1, αλλιώς CY=0
        JC LSB          ; Αν CY = 1, τότε άλμα στην ετικέτα LSB
        CPI 09H         ; Σύγκριση του A με το 9, αν A<9 τότε CY=1, αλλιώς CY=0
        JC MSB          ; Αν CY = 1, τότε άλμα στην ετικέτα MSB
        JMP START       ; Άλμα στην ετικέτα START


LSB:    MVI A,0FH       ; Καταχώρηση της τιμής 0FH (00001111) στον A
        CMA             ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
        STA 3000H       ; Ενημέρωση της κατάστασης των LED (διεύθυνση 3000H)
        CALL DELB       ; Προσθήκη delay = BC*1ms = 500*10^-3 = 0,5s
        MVI A,00H       ; Καταχώρηση της τιμής 00H (00000000) στον A
        CMA             ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
        STA 3000H       ; Ενημέρωση της κατάστασης των LED (διεύθυνση 3000H)
        CALL DELB       ; Προσθήκη delay = BC*1ms = 500*10^-3 = 0,5s
        DCR D           ; Μείωση του περιεχομένου του D κατά 1
        JNZ LSB         ; Αν η τιμή του D είναι διάφορη του 0, τότε άλμα στην LSB
        JMP START       ; Άλμα στην ετικέτα START


MSB:    MVI A,F0H       ; Καταχώρηση της τιμής 0FH (11110000) στον A
        CMA             ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
        STA 3000H       ; Ενημέρωση της κατάστασης των LED (διεύθυνση 3000H)
        CALL DELB       ; Προσθήκη delay = BC*1ms = 500*10^-3 = 0,5s
        MVI A,00H       ; Καταχώρηση της τιμής 00H (00000000) στον A
        CMA             ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
        STA 3000H       ; Ενημέρωση της κατάστασης των LED (διεύθυνση 3000H)
        CALL DELB       ; Προσθήκη delay = BC*1ms = 500*10^-3 = 0,5s
        DCR D           ; Μείωση του περιεχομένου του D κατά 1
        JNZ MSB         ; Αν η τιμή του D είναι διάφορη του 0, τότε άλμα στην MSB
        JMP START       ; Άλμα στην ετικέτα START

END
```

iii. Για την υλοποίηση της συγκεκριμένης άσκησης αποθηκεύουμε τον κωδικό κάθε πλήκτρου σε ένα πίνακα στη μνήμη ο οποίος υλοποιείται σύμφωνα με το interface του πληκτρολογίου ως ακολούθως:



806H	805H	807H
804H	800H	802H
00H	803H	801H
01H	02H	03H
04H	05H	06H
07H	08H	09H
0AH	0BH	0CH
0DH	0EH	0FH

Αρχίζοντας από τη θέση 0AC0H κάθε κελί του πίνακα αποθηκεύεται σε δύο συνεχόμενες θέσεις μνήμης δηλαδή σε δύο bytes.

Κώδικας σε assembly

```
IN 10H          ; Άρση προστασίας της μνήμης
LXI H,0806H     ; Στις επόμενες γραμμές αποθηκεύονται οι κωδικοί των
SHLD 0AC0H      ; πλήκτρων όπως εξηγήθηκε πιο πάνω
LXI H,0805H     ; Στην πρώτη θέση μνήμης που αντιστοιχεί σε κάθε πλήκτρο
SHLD 0AC2H      ; αποθηκεύονται τα 8 LSBs (1ο byte) του κωδικού ενώ στη
LXI H,0807H     ; δεύτερη τα 8 MSBs (2ο byte) του κωδικού
SHLD 0AC4H
LXI H,0804H
SHLD 0AC6H
LXI H,0800H
SHLD 0AC8H
LXI H,0802H
SHLD 0ACAH
LXI H,0000H
```

```
        SHLD 0ACCH
        LXI H,0803H
        SHLD 0ACEH
        LXI H,0801H
        SHLD 0AD0H

        MVI C,0FH       ; Για την αποθήκευση των δεκαεξαδικών αριθμών μπορεί
        MVI B,01H       ; να χρησιμοποιηθεί ένα loop
        LXI H,0AD2H
SAVE_NUM: MOV M,B
        INX H
        MVI M,00H
        INX H
        INR B
        DCR C
        JNZ SAVE_NUM

        MVI B,05H       ; Καταχώρηση BC = 5 ώστε να υπάρχει delay
        CALL DELB       ; για να μην λαμβάνεται η εντολή run ως το
                        ; πλήκτρο του οποίου ζητείται ο κωδικός
        MVI C,00H       ; Αρχικοποίηση των 8 LSBs του ζεύγους BC
        MVI B,00H       ; Αρχικοποίηση των 8 MSBs του ζεύγους BC
        MVI E,FEH       ; Αρχικοποίηση του καταχωρητή E με την πρώτη γραμμή
                        ; του πληκτρολογίου FEH (11111110)
LOOP1:  MOV A,E         ; Φόρτωση της τρέχουσας γραμμής στον A
        STA 2800H       ; Αποθήκευση της γραμμής στην πόρτα σάρωσης (2800H)
        LDA 1800H       ; Φόρτωστη στον A της πόρτας ανάγνωσης (1800H)
        ANI 07H         ; Απομόνωση των 3 LSBs
        RRC             ; Δεξιά περιστροφή (έλεγχος του δεξιότερου bit – LSB)
        JNC GO_OUT      ; Αν το LSB ήταν 0, τότε άλμα στην ετικέτα GO_OUT
        INR C
        INR C           ; Αύξηση του καταχωρητή C κατά 2
        RRC             ; Δεξιά περιστροφή (έλεγχος του δεξιότερου bit – 2ο LSB)
        JNC GO_OUT      ; Αν το 2ο LSB ήταν 0, τότε άλμα στην ετικέτα GO_OUT
        INR C
        INR C           ; Αύξηση του καταχωρητή C κατά 2
        RRC             ; Δεξιά περιστροφή (έλεγχος του δεξιότερου bit – 3ο LSB)
        JNC GO_OUT      ; Αν το 3ο LSB ήταν 0, τότε άλμα στην ετικέτα GO_OUT
        INR C
        INR C           ; Αύξηση του καταχωρητή C κατά 2
        MOV A,E         ; Φόρτωση της τρέχουσας γραμμής στον A
        RLC             ; Αριστερή περιστροφή του A (επόμενη γραμμή)
        MOV E,A         ; Αποθήκευση της καινούριας γραμμής στον E
```

```
        CPI FEH         ; Σύγκριση του A με την πρώτη γραμμή (11111110)
        JNZ LOOP1       ; Αν δεν ισούνται, τότε άλμα στην ετικέτα LOOP1
        MVI C,00H       ; Αρχικοποίηση των 8 LSBs του ζεύγους BC
        JMP LOOP1       ; Άλμα στην ετικέτα LOOP1

GO_OUT: LXI H,0AC0H     ; Φόρτωση στο ζεύγος HL της πρώτης θέσης του πίνακα
        DAD B           ; Προσθήκη στο HL του ζεύγους BC (για υπολογισμό της
                          σωστής διεύθυνσης μνήμης)
        MOV A,M         ; Φόρτωση στον A του περιεχομένου της θέσης που είναι
                          αποθηκευμένη στο HL (δηλαδή του 1ου από τα δύο bytes κωδικού)
        STA 0B04H       ; Αποθήκευση του αριθμού αυτού σε μια θέση μνήμης (0B04H)
        INX H           ; Μετάβαση από την τρέχουσα θέση μνήμης στην επόμενη
        MOV A,M         ; Φόρτωση στον A του περιεχομένου της θέσης που είναι
                          αποθηκευμένη στο HL (δηλαδή του 2ου από τα δύο bytes κωδικού)
        STA 0B05H       ; Αποθήκευση του αριθμού αυτού στην επόμενη θέση μνήμης
                          από αυτήν που αποθηκεύτηκε το 1ο byte κωδικού (0B05H)
        MVI A,10H       ; Φόρτωση στον A της τιμή 10H
        STA 0B03H       ; Αποθήκευση της τιμής του A στις 4 προηγούμενες θέσεις
        STA 0B02H       αυτών που αποθηκεύσαμε τα 2 bytes κωδικού πλήκτρου
        STA 0B01H
        STA 0B00H
        LXI D,0B00H     ; Φόρτωση στον καταχωρητή D της διεύθυνσης της τελευταίας
                          θέσης μνήμης που χρησιμοποιήθηκε για αποθήκευση (0B00H)
        CALL STDM       ; Αρχίζοντας από τη διεύθυνση που είναι αποθηκευμένη στο
                          ζεύγος DE, αποθήκευση των 6 πρώτων bytes στις θέσεις
                          0BF0H-0BF5H (RAM)

LOOP2:  CALL DCD        ; Εμφάνιση των bytes που βρίσκονται στη RAM (0BF0-0BF5)
                          στο display αρχίζοντας από δεξιά
        JMP LOOP2       ; Άλμα στην ετικέτα LOOP2 για διατήρηση του αποτελέσματος
                          στην οθόνη

HLT

END
```

4η ΑΣΚΗΣΗ

Κώδικας σε assembly

```
        10H         ; Άρση προστασίας της μνήμης
START:  LDA 2000H   ; Φόρτωση του περιεχομένου της θέσης 2000H στον A
        MOV B,A     ; Αποθήκευση της εισόδου (2000H) στον καταχωρητή B
        MVI D,00H   ; Αρχικοποίηση του καταχωρητή D = 00H (00000000),
                    ; όπου D η έξοδος

GATE_1: RLC         ; Αριστερή περιστροφή του A
        ANA B       ; Πράξη A and B και αποθήκευση του αποτελέσματος στον A
        RLC         ; Αριστερή περιστροφή του A και αποθήκευση του MSB στον CY
        JNC GATE_2  ; Αν CY = 0 (A3 and B3 = 0), τότε άλμα στην GATE_2
        MVI D,08H   ; Ενημέρωση D = 08H (00001000)

GATE_2: MOV A,B     ; Φόρτωση του περιεχομένου του B (είσοδος) στον A
        RLC         ; Αριστερή περιστροφή του A
        ORA B       ; Πράξη A or B και αποθήκευση του αποτελέσματος στον A
        RLC         ; Αριστερή περιστροφή του A
        RLC         ; Αριστερή περιστροφή του A
        RLC         ; Αριστερή περιστροφή του A και αποθήκευση του MSB στον CY
        MVI E,00H   ; Καταχώρηση της τιμής 00H (00000000) στον E
        JNC XOR     ; Αν CY = 0 (A2 or B2 = 0), τότε άλμα στην XOR
        MVI E,0CH   ; Καταχώρηση της τιμής 00H (00001100) στον E

XOR:    MOV A,D     ; Φόρτωση του περιεχομένου του D (έξοδος) στον A
        XRA E       ; Πράξη A xor E και αποθήκευση του αποτελέσματος στον A
        MOV D,A     ; Ενημέρωση του καταχωρητή D (έξοδος)

GATE_3: MOV A,B     ; Φόρτωση του περιεχομένου του B (είσοδος) στον A
        RRC         ; Δεξιά περιστροφή του A
        ANA B       ; Πράξη A and B και αποθήκευση του αποτελέσματος στον A
        RRC         ; Δεξιά περιστροφή του A
        RRC         ; Δεξιά περιστροφή του A
        RRC         ; Δεξιά περιστροφή του A και αποθήκευση του LSB στον CY
        MVI E,00H   ; Καταχώρηση της τιμής 00H (00000000) στον E
        JNC EXIT_3  ; Αν CY = 0 (A1 and B1 = 0), τότε άλμα στην EXIT_3
        MVI E,02H   ; Καταχώρηση της τιμής 02H (00000010) στον E

EXIT_3: MOV A,D     ; Φόρτωση του περιεχομένου του D (έξοδος) στον A
        ORA E       ; Πράξη A or E και αποθήκευση του αποτελέσματος στον A
        MOV D,A     ; Ενημέρωση του καταχωρητή D (έξοδος)
```

```
GATE_4:  MOV A,B      ; Φόρτωση του περιεχομένου του B (είσοδος) στον A
         RRC          ; Δεξιά περιστροφή του A
         ORA B        ; Πράξη A or B και αποθήκευση του αποτελέσματος στον A
         RRC          ; Δεξιά περιστροφή του A και αποθήκευση του LSB στον CY
         MVI E,00H    ; Καταχώρηση της τιμής 00H (00000000) στον E
         JNC EXIT_4   ; Αν CY = 0 (A0 or B0 = 0), τότε άλμα στην EXIT_3
         MVI E,01H    ; Καταχώρηση της τιμής 01H (00000001) στον E

EXIT_4:  MOV A,D      ; Φόρτωση του περιεχομένου του D (έξοδος) στον A
         ORA E        ; Πράξη A or E και αποθήκευση του αποτελέσματος στον A


LIGHTS:  CMA          ; Συμπλήρωμα ως προς 1 του A και καταχώρηση του στον ίδιο
         STA 3000H    ; Ενημέρωση της κατάστασης των LED (διεύθυνση 3000H)
         JMP START    ; Άλμα στην ετικέτα START

END
```

Θεωρητικές Ασκήσεις

5η ΑΣΚΗΣΗ

SRAM 128 λέξεις × 4 bits

Υπάρχουν 128 (2^7) διευθύνσεις, μία για κάθε λέξη.

Συνολικό πλήθος των bits: $128 \times 4 = 2^7 \times 2^2 = 2^9 = 512$ bits

Η ψηφίδα είναι βέλτιστο να είναι τετραγωνική, στην συγκεκριμένη περίπτωση όμως αυτό δεν μπορεί να πραγματοποιηθεί (2^9 bits), επομένως γίνεται προσπάθεια να πλησιάζει η ψηφίδα όσο το δυνατό περισσότερο στο τετράγωνο.
Συνεπώς η ψηφίδα κατασκευάζεται με τη μία πλευρά αποτελούμενη από 2^4 bits (γραμμές) και τη δεύτερη από 2^5 bits (στήλες).

Η κάθε λέξη αποτελείται από 4 bits, επομένως η ψηφίδα πλάτους 32 bits χωρίζεται κατακόρυφα σε 4 τμήματα των 8 bits.
Έτσι για κάθε μία από τις 16 (2^4) οριζόντιες γραμμή που επιλέγεται, μεταφέρονται 4 ομάδες των 8 bits στις εισόδους 4 πολυπλεκτών.

Ο κάθε πολυπλέκτης είναι 8 εισόδων, και 3 επιλογέων, ενώ 1 bit προκύπτει στην έξοδο του καθενός από αυτούς.
Άρα 3 bits από τη διεύθυνσή (και συγκεκριμένα τα λιγότερο σημαντικά A_0, A_1 και A_2) οδηγούν τους επιλογείς των πολυπλεκτών.

Τα υπόλοιπα 4 bits της διεύθυνσης (A_6, A_5, A_4, A_3) οδηγούν έναν αποκωδικοποιητή 4 σε 16, από τον οποίο προκύπτουν οι 16 γραμμές εισόδοι της ψηφίδας.

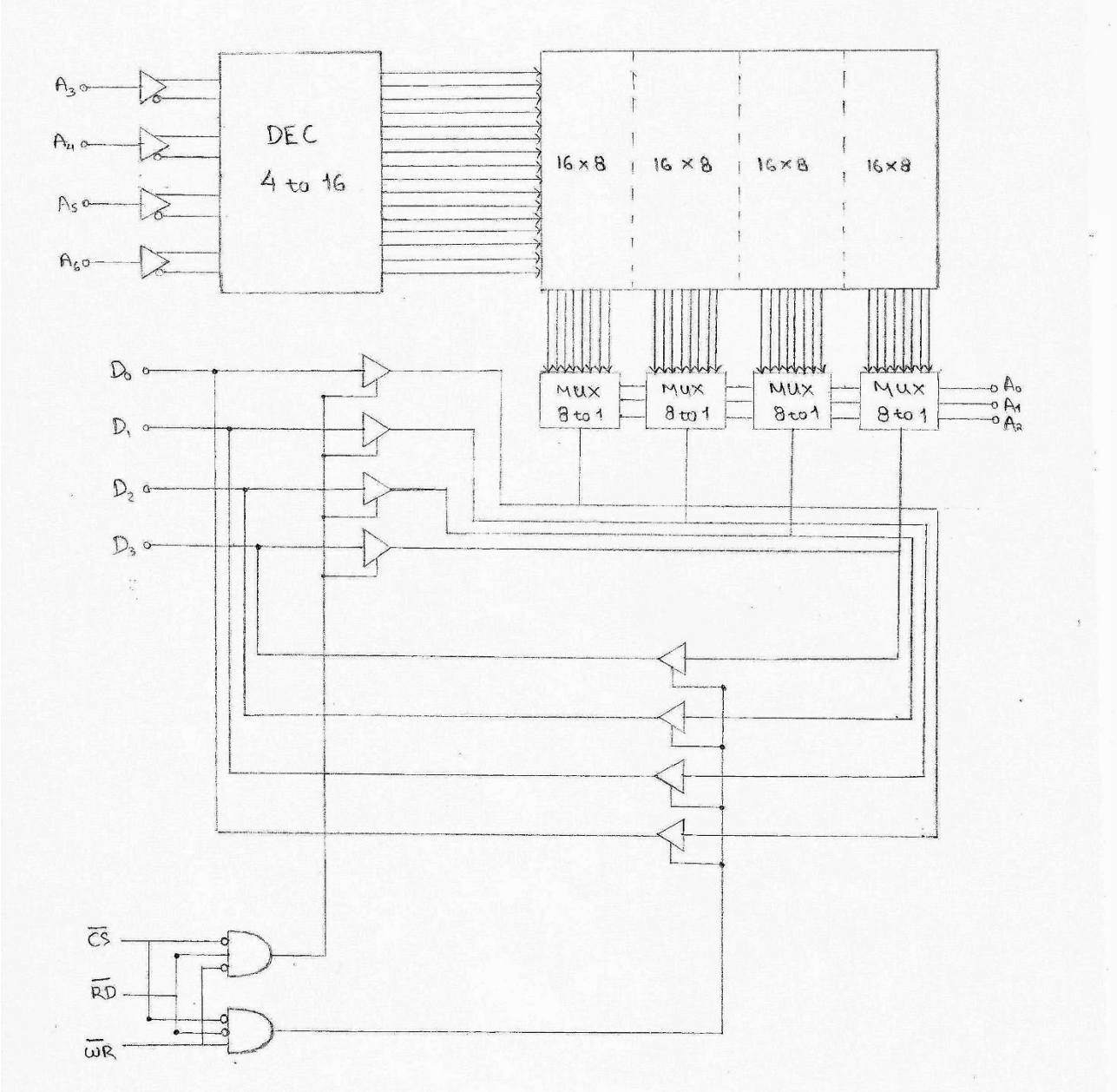
Έτσι χρησιμοποιώντας 7 bits (3 LSBs στους πολυπλέκτες και 4 MSBs στον αποκωδικοποιητή), προκύπτουν οι $2^7 = 128$ διευθύνσεις της μνήμης SRAM.

Όσον αφορά τα σήματα εισόδου, θεωρούμε ότι τα $\overline{WE}$ και $\overline{RD}$ πρέπει να είναι πάντα συμπληρωματικά μεταξύ τους (δηλαδή δεν μπορούν να είναι ταυτόχρονα και τα δύο 1 ή 0).
Η είσοδος $\overline{CS}$ απομονώνει την είσοδο και την έξοδο.
Όλα τα σήματα είναι αρνητικής λογικής, δηλαδή δουλεύουν όταν βρίσκονται στο λογικό 0.

Οι λειτουργίες της μνήμης συνοψίζονται στον ακόλουθο πίνακα:

$\overline{CS}$	$\overline{WE}$	$\overline{RD}$	Λειτουργία
1	X	X	Απομονώνει
0	0	1	Εγγραφή
0	1	0	Ανάγνωση

Η εσωτερική οργάνωση μιας μνήμης SRAM 128×4 είναι η ακόλουθη:



Παράδειγμα εγγραφής στη μνήμη

Έστω ότι επιθυμούμε την εγγραφή της λέξης 0111 στη διεύθυνση 0101010 της μνήμης.

Τότε πρέπει: $\overline{CS} = 0$ (επιτρεψιμότητα λειτουργίας)

$\overline{WE} = 0$ (εγγραφή)

$\overline{RD} = 1$ (όχι ανάγνωση)

Με τα σήματα αυτά στις εισόδους των AND πυλών, η πάνω πύλη έχει σαν έξοδο λογικό 1, ενώ η κάτω πύλη έχει σαν έξοδο λογικό 0.

Το λογικό 1 από την έξοδο της πάνω AND οδηγείται στους ακροδέκτες Gate των τρισταθών πυλών που βρίσκονται στις γραμμές εισόδου (ακροδέκτες D_0, D_1, D_2, D_3), και έτσι επιτρέπει να περάσουν τα σήματα από την είσοδο στην έξοδο των πυλών αυτών.

Το λογικό 0 από την έξοδο της κάτω AND οδηγείται στους ακροδέκτες Gate των τρισταθών πυλών που έχουν ως εισόδους τις εξόδους των 4 πολυπλεκτών. Έτσι απομονώνονται οι γραμμές αυτές από το δίαυλο ώστε να μην μπορούν να επηρεάσουν τα σήματα που έρχονται από τις γραμμές εισόδου (ακροδέκτες D_0, D_1, D_2, D_3).

Η λέξη 0111 οδηγείται στους ακροδέκτες D_0, D_1, D_2 και D_3.

Τα 3 LSBs της διεύθυνσης 0101010, δηλαδή ξεκινώντας από δεξιά τα 0,1 και 0, οδηγούνται στους ακροδέκτες A_0, A_1 και A_2 αντίστοιχα.

Τα υπόλοιπα 4 MSBs δηλαδή 1, 0, 1 και 0 οδηγούνται στους ακροδέκτες A_3, A_4, A_5 και A_6 αντίστοιχα, επιλέγονας έτσι τη σωστή γραμμή της ψηφίδας.

Παράδειγμα ανάγνωσης από τη μνήμη

Έστω ότι επιθυμούμε την ανάγνωση της λέξης που βρίσκεται στη διεύθυνση 0101010 της μνήμης.

Τότε πρέπει: $\overline{CS} = 0$ (επιτρεψιμότητα λειτουργίας)

$\overline{WE} = 1$ (όχι εγγραφή)

$\overline{RD} = 0$ (ανάγνωση)

Με τα σήματα αυτά στις εισόδους των AND πυλών, η πάνω πύλη έχει σαν έξοδο λογικό 0, ενώ η κάτω πύλη έχει σαν έξοδο λογικό 1.

Το λογικό 0 από την έξοδο της πάνω AND οδηγείται στους ακροδέκτες Gate των τρισταθών πυλών που βρίσκονται στις γραμμές εισόδου (ακροδέκτες D_0, D_1, D_2, D_3), και έτσι δεν επιτρέπεται να περάσουν τα σήματα από την είσοδο στην έξοδο των πυλών αυτών.

Τα 4 MSBs της διεύθυνσης 0101010 ξεκινώντας από δεξιά, δηλαδή 1, 0, 1 και 0 οδηγούνται στους ακροδέκτες A_3, A_4, A_5 και A_6 αντίστοιχα, καθορίζοντας έτσι τη σωστή γραμμή της ψηφίδας.

Η ψηφίδα εξάγει τη σωστή οριζόντια γραμμή των 32 bits ομαδοποιημένη σε 4 ομάδες των 8 bits, και η κάθε ομάδα από αυτές οδηγείται στις εισόδους ενός από τους πολυπλέκτες.

Τα 3 LSBs της διεύθυνσης 0101010, δηλαδή ξεκινώντας από δεξιά τα 0,1 και 0, οδηγούνται στους ακροδέκτες A_0, A_1 και A_2 αντίστοιχα, καθορίζοντας έτσι ποιο από τα 8 bits στην είσοδο κάθε πολυπλέκτη θα περάσει στην έξοδο κάθε ενός από αυτούς.

Τα 4 bits που εξάγονται από τους πολυπλέκτες φτάνουν στις εισόδους των τρισταθών πυλών που βρίσκονται κάτω δεξιά στο σχήμα.

Ταυτόχρονα στα gates αυτών των πυλών φτάνει το λογικό 1 από την έξοδο της κάτω πύλης AND, επιτρέποντας έτσι στα 4 bits των εισόδων να περάσουν στις εξόδους των πυλών και να καταλήξουν στους ακροδέκτες εξόδους D_0, D_1, D_2, και D_3 της μνήμης.

6η ΑΣΚΗΣΗ

Η διευθυνσιοδότηση των θέσεων μνήμης για κάθε τμήμα του συστήματος μνήμης παρουσιάζεται στον ακόλουθο χάρτη μνήμης:

			A15	A14	**A13**	**A12**	**A11**	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
ROM	1ο IC	0000H	0	0	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0
	2K×8	07FFH	0	0	**0**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1
	2ο IC	0800H	0	0	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0
	4K×8	17FFH	0	0	**0**	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1
RAM	1ο IC	1800H	0	0	**0**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0
	2K×8	1FFFH	0	0	**0**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1
	2ο IC	2000H	0	0	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0
	8K×8	3FFFH	0	0	**1**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1

Στον πιο πάνω πίνακα φαίνονται με έντονα γράμματα τα bits της διεύθυνσης (A_{13}, A_{12}, A_{11}) που είναι καθοριστικά για την επιλογή του τμήματος μνήμης που χρησιμοποιείται σε κάθε περίπτωση.

(α) Στην πρώτη περίπτωση, τα τρία αυτά bits A_{13}, A_{12} και A_{11}, οδηγούν τις εισόδους X_0, X_1 και X_2 ενός αποκωδικοποιητή 3:8 (74LS138), ο οποίος παράγει όλους τους δυνατούς συνδυασμούς τους στις εξόδους του
$\overline{Y_0} - \overline{Y_7}$.
Σύμφωνα με τον χάρτη μνήμης χρησιμοποιώντας τις κατάλληλες λογικές πύλες στις εξόδους του αποκωδικοποιητή, επιλέγεται η επίτρεψη ή απότρεψη της λειτουργίας των τμημάτων του συστήματος μνήμης που ενδείκνυνται σε κάθε περίπτωση.

Οι εισόδοι $\overline{E_1}$ και $\overline{E_2}$ του αποκωδικοποιητή τροφοδοτούνται αντίστοιχα από τα bits A_{14} και A_{15} της διεύθυνσης.
Η είσοδος E_3 του αποκωδικοποιητή δέχεται ένα σήμα $IO/\overline{M}$, το οποίο τον ενεργοποιεί μόνο όταν είναι 1.

Τα bits A_{10} έως A_0 αποτελούν τα δεδομένα που ρέουν στον δίαυλο δεδομένων εισόδου.
Η κάθε μνήμη ανάλογα με την είσοδο της (διεύθυνση) εξάγει τα επιθυμητά δεδομένα στον δίαυλο εξόδου.

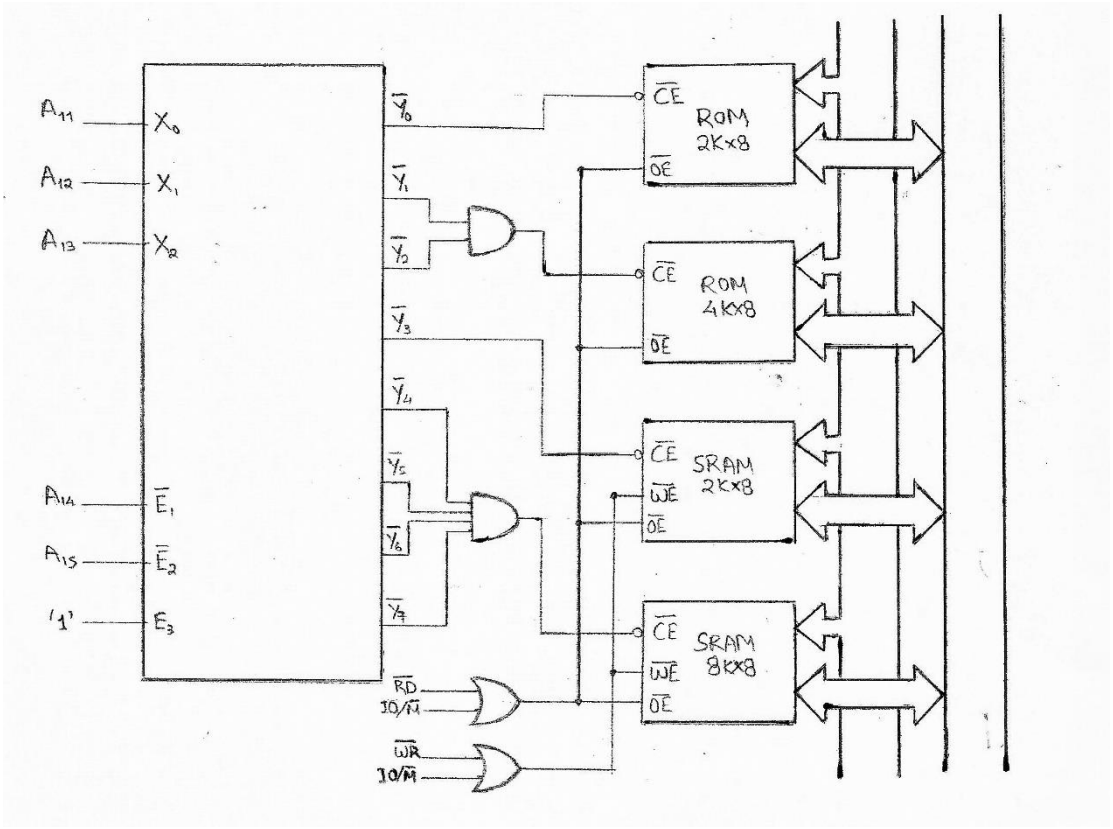
Επιπλέον όλα τα ICs δέχονται τα σήματα $\overline{CE}$ και $\overline{OE}$, τα οποία δίνουν την επίτρεψη στο IC για να λειτουργήσει και την επίτρεψη για να διαβάσει αντίστοιχα.

Οι μνήμες RAM έχουν μία επιπρόσθετη γραμμή ελέγχου $\overline{WE}$, η οποία δίνει την επίτρεψη για εγγραφή.

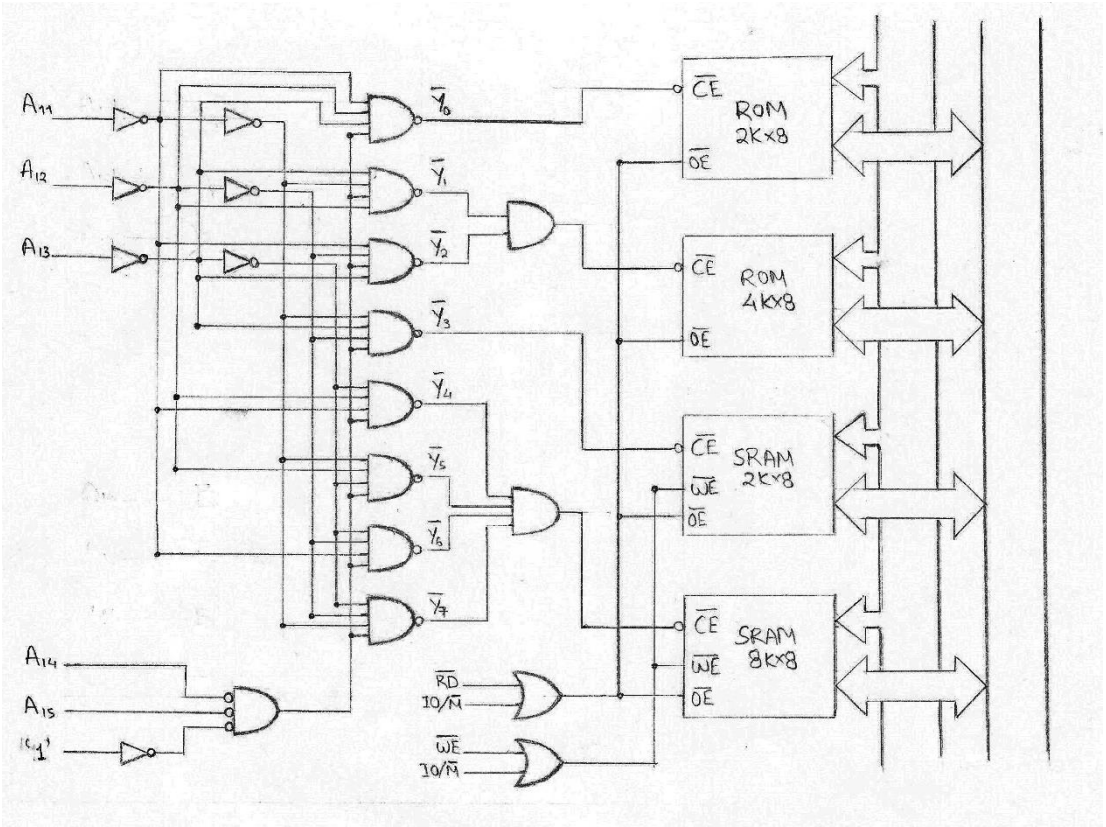
(β) Στη δεύτερη περίπτωση ο αποκωδικοποιητής αντικαθίσταται από λογικές πύλες, ενώ όλες οι υπόλοιπες υλοποιήσεις παραμένουν αναλλοίωτες.

Το σύστημα μνήμης για τις περιπτώσεις (α) και (β) παρουσιάζεται ακολούθως:

(α)



(β)



7η ΑΣΚΗΣΗ

Η διευθυνσιοδότηση των θέσεων μνήμης για κάθε τμήμα του συστήματος μνήμης παρουσιάζεται στον ακόλουθο χάρτη μνήμης:

			A15	**A14**	**A13**	**A12**	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
ROM	1ο IC 16K×8	0000H	0	**0**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
		0FFFH	0	**0**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	1
		4000H	0	**1**	**0**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
		6FFFH	0	**1**	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1	1
RAM	1ο IC 4K×8	1000H	0	**0**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
		1FFFH	0	**0**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	1
	2ο IC 8K×8	2000H	0	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	0
		3FFFH	0	**0**	**1**	**1**	1	1	1	1	1	1	1	1	1	1	1	1
Memory map I/O		7000H	0	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0
Standard I/O		70H	0	**0**	**0**	**0**	0	0	0	0	0	1	1	1	0	0	0	0