Εθνικο Μετσοβιο Πολυτεχνειο

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧ. ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

2η ΑΣΚΗΣΗ

Αχλάτης Στέφανος-Σταμάτης (03116149)

<el16149@central.ntua.gr>

Καπερώνη Φρειδερίκη (03116685)

<el16685@central.ntua.gr>

Μάιος 2020

Περιεχόμενα

1η Άσκηση

Το πρόγραμμα και τα αντιστοιχα σχόλια όπως γράφτηκε στο TSIK παρατίθεται παρακάτω:

```
PART_A:
       IN 10H ;apenergopoihsh prostasia mnimis
       LXI H,0900H ;arxiki 8esh mnimis
       MVI A,FFH ;arxikopoihsh regA me 255

SAVE_IN_MEM:
       MOV M,A ;o A exei ton ari8mo pros apo8hkeusi
              ;kai ton apo8hkeuw sthn 8esh mnimis
              ;pou deixnei o H
       DCR A ;A<-A-1
       CPI 00H ;an ftasame sto A=0 telos
       JZ PART_B ;kai phgaine sto meros B
       INX H ;an oxi deikse thn epomenh 8esh mnimis
       JMP SAVE_IN_MEM ;kai phgaine pali ekei

PART_B:
       LXI H,0900H ;arxiki 8esh mnimis
       MVI D,00H ; O D exei to plh8os twn 0
       MVI E,00H ;
       MVI C,08H ; O C metraei posa bits exoun
                 ; metri8ei mexri twra

LOAD_PART_B:
       MOV A,M ; diavase ton ari8mo apo tin mnimi

LOOP_PART_B:
       MOV B,A ; B voi8htiki metavliti
       MOV A,C ; an ola ta bit tou ari8mou exoun
       CPI 00H ; elegx8ei pigaine ston epomeno num
       JZ CHECK_NEXT
       MOV A,B ; diaforetika
       RAL     ;check to epomeno bit mesw tou carry
       DCR C   ; mas menoun kata ena ligotera bits
               ; gia check
       JC LOOP_PART_B ; an to bit einai 1 mhn
                      ; aukisieis to plh8os twn 0
       INX D   ; an to bit htan 0 aukise to
               ; pli8os twn 0
       JMP LOOP_PART_B      ; kai pigaine ekei


CHECK_NEXT: ; edw pigainw an ola ta bit tou num
            ; exoun elegx8ei
       INX H ; deikse thn epomenh 8esh mnimis
       MOV A,H ; fortwse sto A ton H
       CPI 0AH ; des an exoun diavastei oloi oi num
       JZ PART_C ; an nai teleiwses
       MVI C,08H ; diaforetika elegkse to epomeno
       JMP LOAD_PART_B ; kai arxikopihse C kai A
PART_C:
       LXI H,0900H ; arxiki 8esi mnimis
       MVI C,00H ; arxikopoisi tou C
                 ; GIA TIN METRISI
IN_RANGE:
       MOV A,M ; diavase ari8mo apo mnimi
       CPI 20H ; einai mikroteros tou 20H?
       JC LOOK_NEXT ; NAI? tote koita ton next num
       CPI 71H ; Einai megaluteros tou 70H?
       JNC LOOK_NEXT ; NAI? tote koita ton next num
       INR C ; An oxi eimai entos oriwn opote aukise
             ; to plh8os ton zitoumenwn ari8mwn

LOOK_NEXT:
       INX H ; deikse tin epomeni 8esh mnimis
       MOV A,H ; check an eidame olous tous num
       CPI 0AH
       JZ TO_END ; An nai tote telos
       JMP IN_RANGE ; an oxi check next num

TO_END:
       END
```

Ας ελέγξουμε την ορθότητα του Α ερωτήματος.
Το αποτέλεσμα στην μνήμη του μικροϋπολογιστή μας είναι το εξής:

08F7	00	08F8	00	08F9	00	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

που είναι το ζητούμενο από την εκφώνηση της άσκησης.

Ας ελέγξουμε την ορθότητα του Β ερωτήματος.

Ο τροποποιημένος κώδικας έτσι ώστε να εκτυπώνεται το D:

```
        JZ LEDS ; an nai teleiwses
        MVI C,08H ; diaforetika elegkse to epomeno
        JMP LOAD_PART_B ; kai arxikopihse C kai A

LEDS:
        MOV A,D
        CMA
        STA 3000H
        END
```

Το περιεχόμενο του D:

Δηλαδή το D περιέχει τον αριθμό 8 (σε δεκαδικό σύστημα αρίθμησης).

Ο τροποποιημένος κώδικας έτσι ώστε να εκτυπώνεται το E:

```
        JZ LEDS ; an nai teleiwses
        MVI C,08H ; diaforetika elegkse to epomeno
        JMP LOAD_PART_B ; kai arxikopihse C kai A

LEDS:
        MOV A,E
        CMA
        STA 3000H
        END
```

Το περιεχόμενο του E:

Δηλαδή το E περιέχει τον αριθμό 0.

Άρα, ο DE περιέχει τον αριθμό (1024)10 , το οποίο είναι σωστό αφού γνωρίζουμε πως από το 0 έως το 2 8 – 1) υπάρχουν 2048 bits από τα οποία τα μισά είναι 0 και τα υπόλοιπα 1.

Ας ελέγξουμε την ορθότητα του Γ ερωτήματος.

Ο τροποποιημένος κώδικας έτσι ώστε να εκτυπώνεται το C:

```
      JZ LEDS ; An nai tote telos
      JMP IN_RANGE ; an oxi check next num

LEDS:
      MOV A,C
      CMA
      STA 3000H
      END
```

Το περιεχόμενο του C:

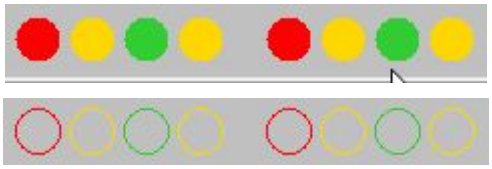
που είναι ο αριθμός 81.
Πράματι οι αριθμοί που βρίσκονται στο διάστημα
[(32)10, (112)10] → [(20)16, (70)16].
Πράματι οι αριθμοί που βρίσκονται στο διάστημα 32-112 μαζί με τους αριθμούς 32 και 112 είναι 81 στο σύνολο.

2η Άσκηση

Το πρόγραμμα και τα αντιστοιχα σχόλια όπως γράφτηκε στο TSIK παρατίθεται παρακάτω:

```
INIT:
      MVI B,00H  ; mesw twn bc orixoume thn
      MVI C,64H  ; xronoka8isterisi isi me 100msec
START:
      LDA 2000H ; diavase input
      RAR ; check an to lsb einai 0
      JC START ; an den einai 0 diavase ksana
               ; to input mexri na ginei 0

OFF_TO_ON:        ; edw to lsb einai 0
      LDA 2000H  ; diavase input
      RAR        ; elegxoume to lsb
      JNC OFF_TO_ON ; diavaze to input mexri
                    ; to lsb na ginei 0

ON_TO_OFF:          ; edw to leb htan 1
                    ; kai exei anoiksei mia fora
      LDA 2000H     ; diavase to input
      RAR           ; elegxoume to lsb
      JC ON_TO_OFF  ; an einai akomi 1 tote
                    ; diavase ksana to input
                    ;mexri na ginei 0

LED_ON:      ; otan ftasw edw simainei oti to led
             ; htan off meta on kai twra off
             ; to push-button exei energopoih8ei
      MVI D,00H
      MVI E,67H ; giati 8elw sunolika 15sec kai
                ; uparxoun mikroka8isteriseis
      MVI A,FFH ; gia na einai ola ta led off

DELAY:
      STA 3000H ; kleise ola ta led
      CALL DELB ; wait 100ms
      CMA       ; gia na einai ola ta led on
      STA 3000H ; anoigoun ola ta led
      CALL DELB ; wait 100ms

      DCR E  ; meiwse ton upoloipomeno xrono
      MOV A,E
      CPI 00H ; check an perasan ta 15sec
      JZ LED_OFF ; an perase kleise ola ta led
      MOV A,D ; an oxi, check an o diakoptis
      CPI 00H ;
      JZ LED_IS_ON ; an nai des pou einai o
                   ; diakoptis twra
      CPI 01H      ; einai off?
      JZ DOWN_STATE ; an nai des p eiani twra
      CPI 02H       ; exoume energopoihsei to
                    ; push down button?
      JZ DOWN_UP_STATE ; an nai des pou einai twra

LED_OFF:            ;perasan 15 sec
      MVI A,FFH     ;svise ta led
      STA 3000H
      JMP START     ;programma sinexis leitourgias

LED_IS_ON:
      LDA 2000H ; diavase input
      RAR       ; des to lsb
      JC DELAY  ; o diakoptis anoikse kai 8a doume
                ; an exei kleisei
      INR D     ; D<-D+1

DOWN_STATE:
      LDA 2000H
      RAR
      JNC DELAY ; o diakoptis ekleise afou
                ; prwta eixe anoikse
      INR D

DOWN_UP_STATE:
      LDA 2000H
      RAR
      JC DELAY ; to push-button einai on

UP_DOWN_STATE:
      JMP LED_ON ; opote ananewse ton xrono!
      END
```

Το αποτέλεσμα λοιπόν είναι αφότου ενεργοποιηθεί το push-button τα led του μικρουπολογιστη να αναβοσβήνουν με 100ms και αυτή η ενέργεια διαρκεί για συνολικά 15sec.
Αν ωστόσο ενεργοποιηθεί ξανα το push-down button τότε θα πρέπει να ανανεωθεί ο χρόνος των 15sec.
Έτσι λοιπόν αυτό που βλέπουμε είναι το εξής:



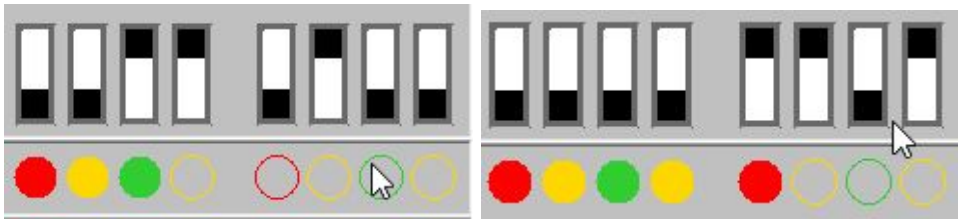
3η Άσκηση

Α Ερώτημα:

Το πρόγραμμα και τα αντιστοιχα σχόλια όπως γράφτηκε στο TSIK παρατίθεται παρακάτω:

```
PART_1:

START:
       LDA 2000H ; read input
       RAL ; check bit8
       JC TURN_1 ; if bit8=1 turn 1 led
       RAL ; check bit7
       JC TURN_2 ; if bit7=1 turn 2 led
       RAL ; check bit6
       JC TURN_3 ; if bit6=1 turn 3 leds
       RAL ; check bit5
       JC TURN_4 ; if bit5=1 turn 4 leds
       RAL ; check bit4
       JC TURN_5 ; if bit4=1 turn 5 leds
       RAL ; check bit3
       JC TURN_6 ; if bit3=1 turn 6 leds
       RAL ; check bit2
       JC TURN_7 ; if bit2=1 turn 7 leds
       RAL ; check bit1
       JC TURN_8 ; if bit1=1 turn 8 leds
       MVI A,FFH ; else all leds turned of
       STA 3000H
       JMP START ; programma sunexoun leitougias

TURN_8:
       MVI A,00H ; gia na anoiksoun ola ta led
       STA 3000H
       JMP START ; programma sunexous leitougias
TURN_7:          ; antisoixa ta upoloipa
       MVI A,01H
       STA 3000H
       JMP START
TURN_6:
       MVI A,03H
       STA 3000H
       JMP START
TURN_5:
       MVI A,07H
       STA 3000H
       JMP START

TURN_4:
       MVI A,0FH
       STA 3000H
       JMP START
TURN_3:
       MVI A,1FH
       STA 3000H
       JMP START
TURN_2:
       MVI A,3FH
       STA 3000H
       JMP START
TURN_1:
       MVI A,7FH
       STA 3000H
       JMP START
END
```

Και μερικά παραδείγματα που δειχνουν την ορθή λειτουργία του προγράμματος μας:

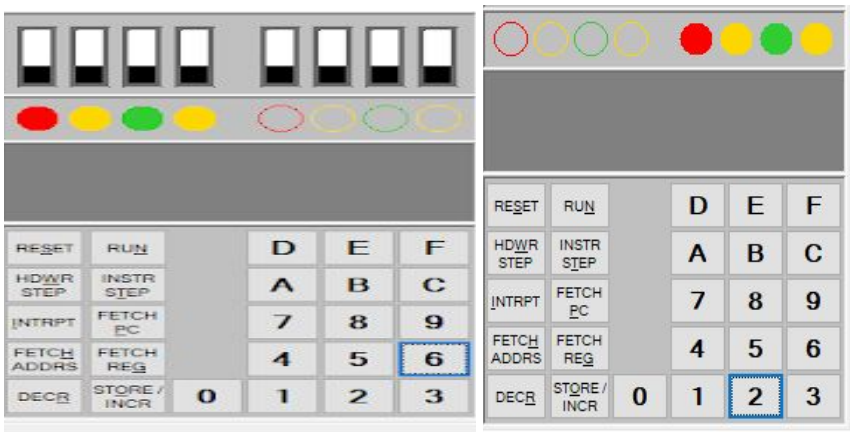


Β Ερώτημα:

Το πρόγραμμα και τα αντιστοιχα σχόλια όπως γράφτηκε στο TSIK παρατίθεται παρακάτω:

```
PART2:
START:
      CALL KIND ; read keyboard
      CPI 01H   ; an einai to 1
      JZ LSB    ; tote anoikse ta lsb
      CPI 02H   ; an einai to 2
      JZ LSB    ; tote anoikse ta lsb
      CPI 03H   ;klp
      JZ LSB
      CPI 04H
      JZ LSB

      CPI 05H  ; an einai to 5
      JZ MSB   ; tote anoikse ta msb
      CPI 06H  ; klp
      JZ MSB
      CPI 07H
      JZ MSB
      CPI 08H
      JZ MSB

      MVI A,FFH ; an den dw8ei kati svista led
      STA 3000H
      JMP START ; kai kane sunexeia loop mexri
                ; na dw8ei

LSB:
      LXI B,01FAH ; stall=500ms
      MVI D,04H   ; counter epeidi 8elw 4 fores
                  ; na naoigokleisoun ta led

LOOP_LSB:
      MVI A,0FH
      CMA         ; anoikse ta 4 lsb
      STA 3000H   ; anoikse ta
      CALL DELB   ; wait 500ms
      MVI A,FFH
      STA 3000H   ; svista ola
      CALL DELB   ; wait 500ms
      DCR D       ; ekana mia epanalispi
                  ; meiwse cntr kata ena
      MOV A,D     ; fortwse to cntr sto A
      JNZ LOOP_LSB  ; an den midenistike ksanakane
                    ; tin diadikasia me ta led
      JMP START     ; alliws phgaine sto start

MSB:
      LXI B,01FAH  ; stall= 500ms
      MVI D,04H

LOOP_MSB:           ; akrivws antisoixa
                    ; mono pou anoigw ta 4 msb
      MVI A,0FH
      STA 3000H     ; edw anoigw ta 4 msb
      CALL DELB
      MVI A,FFH
      STA 3000H
      CALL DELB
      DCR D
      MOV A,D
      JNZ LOOP_MSB
      JMP START
END
```

Πράγματι βλέπουμε την επιθυμητή λειτουργία στο task. Βλέπουμε τα led να αναβοσβήνουν και στις παρακάτω φωτογραφίες φαίνεται όταν κάνουμε screenshot οταν τα led ηταν on για την αντίστοιχη κατηγορία.



Γ Ερώτημα:

Το πρόγραμμα και τα αντιστοιχα σχόλια όπως γράφτηκε στο TSIK παρατίθεται παρακάτω:

```
            IN 10H
            MVI A,10H     ; Κενό στα δεξιότερα 7-Segments
            STA 0B00H
            STA 0B01H
            STA 0B02H
            STA 0B03H
            STA 0B04H
            STA 0B05H
            LXI D,0B00H
            CALL STDM
            CALL DCD


START:      MOV H,A
READ_LN_0:  MVI A,FEH     ; 1111 1110
            CALL READ_COLS
            CPI 06H       ; 110
            JZ DIS_IN_ST  ; Εμφάνιση κατάλληλου κωδικού
            CPI 05H       ; 101
            JZ DIS_F_PC   ; Display Fetch PC


READ_LN_1:  MVI A,FDH     ; 1111 1101
            CALL READ_COLS
            CPI 06H       ; 110
            JZ DIS_RUN
            CPI 05H       ; 101
            JZ DIS_FETC_REG
            CPI 03H       ; 011
            JZ DIS_FETC_ADR


READ_LN_2:  MVI A,FBH     ; 1111 1011
            CALL READ_COLS
            CPI 06H       ; 110
            JZ DIS_0
            CPI 05H       ; 101
            JZ DIS_ST_INC
            CPI 03H       ; 011
            JZ DIS_INCR



READ_LN_3:  MVI A,F7H     ; 1111 0111
            CALL READ_COLS
            CPI 06H       ; 110
            JZ DIS_1
            CPI 05H       ; 101
            JZ DIS_2
            CPI 03H       ; 011
            JZ DIS_3

READ_LN_4:  MVI A,EFH     ; 1110 1111
            CALL READ_COLS
            CPI 06H       ; 110
            JZ DIS_4
            CPI 05H       ; 101
            JZ DIS_5
            CPI 03H       ; 011
            JZ DIS_6
```

```
READ_LN_5:   MVI A,DFH     ; 1101 1111
             CALL READ_COLS
             CPI 06H       ; 110
             JZ DIS_7
             CPI 05H       ; 101
             JZ DIS_8
             CPI 03H       ; 011
             JZ DIS_9

READ_LN_6:   MVI A,BFH     ; 1011 1111
             CALL READ_COLS
             CPI 06H       ; 110
             JZ DIS_A
             CPI 05H       ; 101
             JZ DIS_B
             CPI 03H       ; 011
             JZ DIS_C

READ_LN_7:   MVI A,7FH     ; 0111 1111
             CALL READ_COLS
             CPI 06H       ; 110
             JZ DIS_D
             CPI 05H       ; 101
             JZ DIS_E
             CPI 03H       ; 011
             JZ DIS_F

             MOV A,H
             JMP KEEP_DISPLAY
```

```
READ_COLS:    ; Διαβάζει τις στήλες της γραμμής που έχει οριστεί στον A
              ; Αποθηκεύει τα 3 ψηφία στον A
              STA 2800H     ; Ενεργοποίηση Γραμμής
              LDA 1800H     ; Ανάγνωση στηλών
              MVI B,07H     ; Μάσκα 0000 0111
              ANA B         ; Αποθηκεύσει στον A τα τρία ψηφία
CONT:         RET


DIS_IN_ST:    MVI A,86H
              JMP DISPLAY
DIS_F_PC:     MVI A,85H
              JMP DISPLAY
DIS_RUN:      MVI A,84H
              JMP DISPLAY
DIS_FETC_REG: MVI A,80H
              JMP DISPLAY
DIS_FETC_ADR: MVI A,82H
              JMP DISPLAY
DIS_ST_INC:   MVI A,83H
              JMP DISPLAY
DIS_INCR:     MVI A,81H
              JMP DISPLAY
DIS_0:        MVI A,00H
              JMP DISPLAY
DIS_1:        MVI A,01H
              JMP DISPLAY
DIS_2:        MVI A,02H
              JMP DISPLAY
DIS_3:        MVI A,03H
              JMP DISPLAY
```

```
DIS_4:        MVI A,04H
              JMP DISPLAY
DIS_5:        MVI A,05H
              JMP DISPLAY
DIS_6:        MVI A,06H
              JMP DISPLAY
DIS_7:        MVI A,07H
              JMP DISPLAY
DIS_8:        MVI A,08H
              JMP DISPLAY
DIS_9:        MVI A,09H
              JMP DISPLAY
DIS_A:        MVI A,0AH
              JMP DISPLAY
DIS_B:        MVI A,0BH
              JMP DISPLAY
DIS_C:        MVI A,0CH
              JMP DISPLAY
DIS_D:        MVI A,0DH
              JMP DISPLAY
DIS_E:        MVI A,0EH
              JMP DISPLAY
DIS_F:        MVI A,0FH
              JMP DISPLAY
DISPLAY:      MOV B,A       ; Δημιουργία αντιγράφου
              RRC           ; Απομόνωση MSB HEX ψηφίου
              RRC
              RRC
              RRC
              ANI 0FH       ; Μάσκα 0000 1111
              STA 0B05H     ; Αποθήκευση αριστερότερου ψηφίου

              MOV A,B       ; Απομόνωση LSB HEX ψηφίου
              ANI 0FH       ; Μάσκα 0000 1111
              STA 0B04H
              LXI D,0B00H
              CALL STDM
              CALL DCD
              MOV A,B
              JMP START
KEEP_DISPLAY: LXI D,0B00H
              CALL STDM
              CALL DCD
              JMP START
END
```

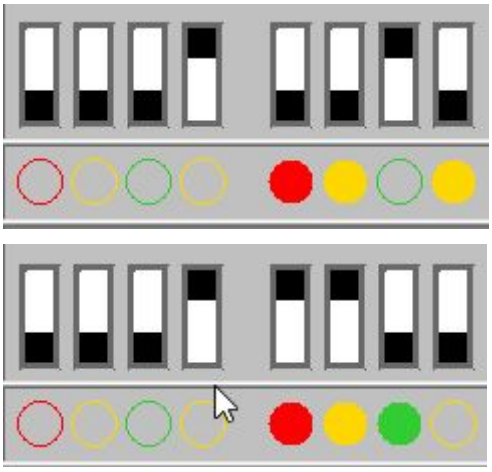
Και παρακάτω παρατίθενται ορισμένα run της εφαρμογής μας:

4η Άσκηση:

Το πρόγραμμα και τα αντιστοιχα σχόλια όπως γράφτηκε στο TSIK παρατίθεται παρακάτω:

```
START:
       LDA 2000H ; read input
       MOV H,A

; X | Y | AND | OR | XOR
; 0 | 0 |  0  | 0  |  0
; 0 | 1 |  0  | 1  |  1
; 1 | 0 |  0  | 1  |  1
; 1 | 1 |  1  | 1  |  0

GATE1_OR:
       ANI 03H
       CPI 00H   ; Compare with 0000 0000
       JNZ OR_1  ; An kai ta dio bit !0 then 1
       MVI B,00H ; diaforetika 0
       JMP GATE2_AND
OR_1:
       MVI B,01H
GATE2_AND:
       MOV A,H
       ANI 0CH
       CPI 0CH
       JZ AND_1      ; compare with 0000 1100
       MVI C,00H     ; an kai ta dio bit=1 then 1
       JMP GATE3_OR  ; else 0
AND_1:
       MVI C,02H
GATE3_OR:
       MOV A,H
       ANI 30H
       CPI 00H       ; compare with 0000 0000
       JNZ OR_2      ;an kai ta dio bit=0 then 1
       MVI D,00H     ;else 0
       MVI L,00H
       JMP GATE4_XOR
OR_2:
       MVI D,04H     ; vale 1 sth swsti 8esh
       MVI L,08H     ; Store it kai sthn epomeni
                     ; thesi

GATE4_XOR:
       MOV A,H
       ANI C0H
       CPI C0H       ; compare with 1100 0000
       JZ AND_2      ; an kai ta duo bit=1 then 1
       MVI E,00H     ; else 0
       MOV A,E       ;8a perasoume ws input sti xor
       XRA L         ;to result ths and kai or
       MOV E,A
       JMP RESULT

AND_2:
       MVI E,08H     ; vale 1 sth swsti 8esh
       MOV A,E
       XRA L
       MOV E,A

RESULT:
       SUB A  ; A=0
       ADD B  ; pros8ese diadoxika ta results
       ADD C
       ADD D
       ADD E
       ANI 0FH ; sta 4msb vale 0
       CMA
       STA 3000H
       JMP START
END
```

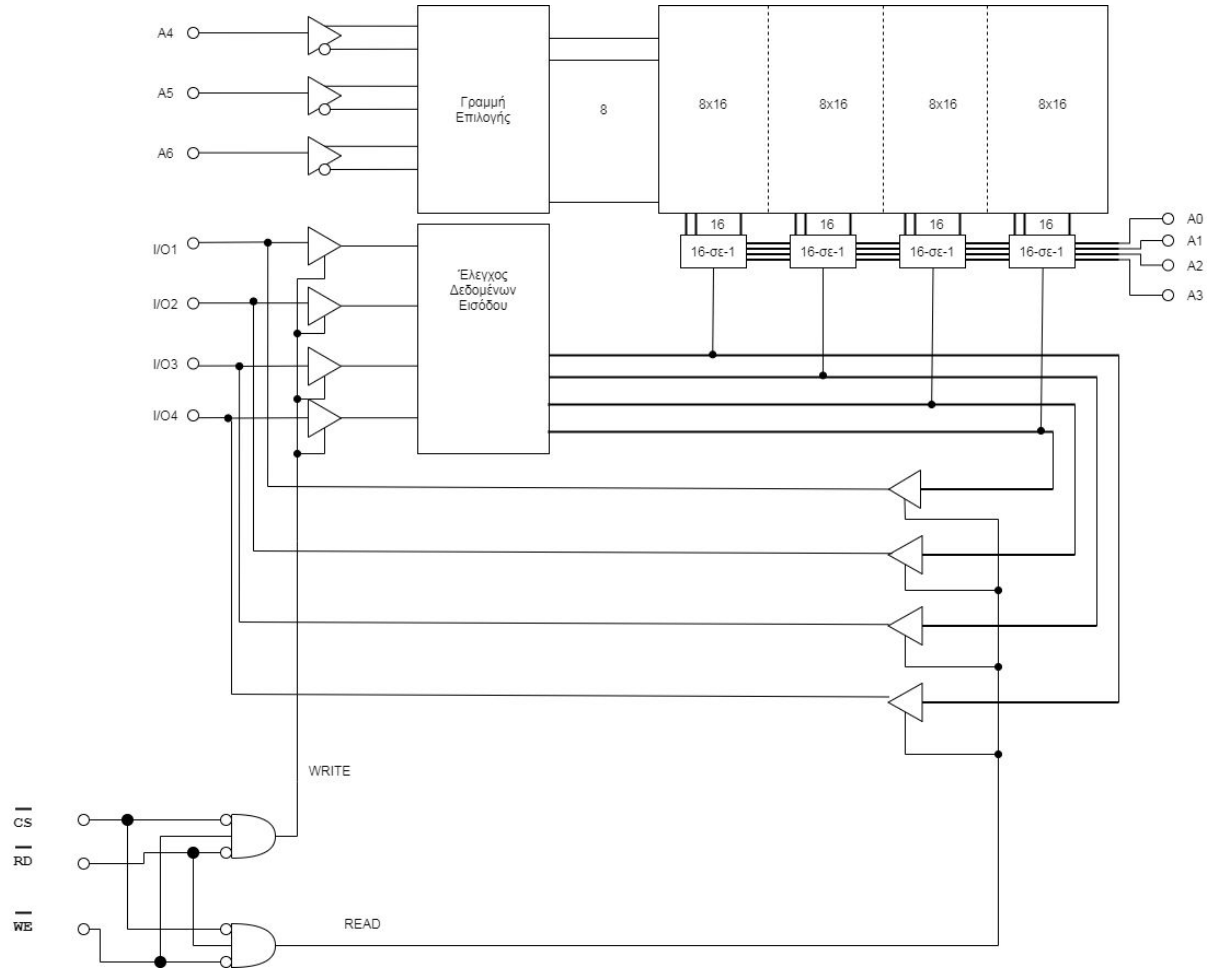
Και παρακάτω παρατίθενται ορισμένα run της εφαρμογής μας:



5η Άσκηση:

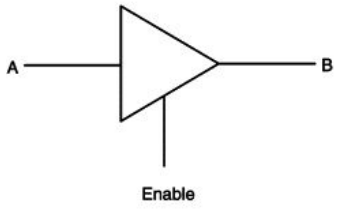
SRAM 128x4 bit

Η μνήμη είναι 128 x 4 bits. Η τετραγωνική διάταξη αποτελείται από 8x64 στοιχεία μνήμης. Από τον πίνακα της μνήμης επιλέγεται με βάση τις γραμμές διεύθυνσης A4-A6 μία από τις 8 γραμμές ($2^3 = 8$). Ο δεύτερος αριθμός, το 4, δηλώνει σε πόσα ίσα μέρη θα είναι χωρισμένη η μνήμη κι ο πρώτος αριθμός, το 128, δηλώνει πόση χωρητικότητα μνήμης έχει το κάθε μέρος. Άρα η μνήμη θα κατακερματίζεται σε 4 μέρη, των 128 bits το καθένα. Κάθε μέρος είναι ένας δισδιάστατος πίνακας με γραμμές και στήλες. Προφανώς το μέγεθος του πίνακα αυτού, πρέπει να ισούται σταθερά με 128 ανεξάρτητα από την κατανομή των γραμμών και των στηλών. Ο λόγος είναι ότι το 128 είναι βασική προϋπόθεση εξ αρχής. Γίνεται φανερό πως οι γραμμές κι οι στήλες μπορούν να κατανεμηθούν με πολλούς τρόπους. Επιλέχθηκε η διάταξη 8x16 για κάθε μέρος. Το πλήθος των γραμμών ισούται με 8 και των στηλών με 16. Η επιλογή αυτή επέφερε την χρήση πολυπλεκτών 16 - σε - 1. Επίσης επέφερε την χρήση 3 bits για τον προσδιορισμό της διεύθυνσης των γραμμών (2 3) και 4 bits για τις στήλες. Τέλος, το γεγονός των 4 μερών επιβάλλει την ύπαρξη και 4 σημάτων I/O για κάθε μία από αυτές. Παρατίθεται η εσωτερική οργάνωση της μνήμης:

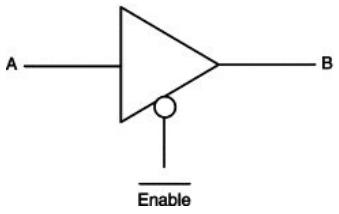


Γραμμές $\overline{CS}$, $\overline{WE}$, $\overline{RD}$

Να σημειωθεί ότι όταν **η είσοδος** $\overline{CS}$ **ισούται με 1,** απομονώνει τόσο την είσοδο όσο και την έξοδο, σύμφωνα μα τον παρακάτω πίνακα τιμών για τρισταθή πύλη (tristate buffer). Εν αντιθέσει, όταν **η είσοδος αυτή ισούται με 0,** "ενεργοποιείται" η συσκευή και η λειτουργία εισόδου/εξόδου.



Enable	A	B
0	0	Z
0	1	Z
1	0	0
1	1	1



$\overline{\text{Enable}}$	A	B
0	0	0
0	1	1
1	0	Z
1	1	Z

[1]

Λειτουργία ανάγνωσης:

$\overline{WE}$, $\overline{RD}$

Για να γίνει ανάγνωση (προφανώς $\overline{CS}$ **= 0**), πρέπει $\overline{RD} = 1$ και $\overline{WE} = 0$, ώστε να ενεργοποιηθεί η δεξιά τετράδα τρισταθών αντιστροφέων και να περάσουν τα bits A0-A3 στις εξόδους 1-4 αντίστοιχα.

Λειτουργία εγγραφής:

Για να γίνει ανάγνωση (προφανώς $\overline{CS}$ **= 0**), πρέπει $\overline{RD} = 0$ και $\overline{WE} = 1$, ώστε να ενεργοποιηθεί η αριστερή τετράδα τρισταθών αντιστροφέων και να γίνει εγγραφή των εισόδων 1-4 στα bits A0-A3

6η Άσκηση:

Στο σύστημα μνήμης γίνεται χρήση του μΕ 8085, άρα για την αναπαράσταση δεδομένων απαιτούνται 8 bits και για τις διευθύνσεις το πολύ 16 bits. Οι 3 γραμμές ελέγχου είναι οι WR(εγγραφή στη μνήμη), RD (ανάγνωση από τη μνήμη) και IO/M(είσοδος-έξοδος(1) ή μνήμη(0)). Η διεύθυνση της συγκεκριμένης θέσης μνήμης δηλώνεται στο bus διευθύνσεων.

Η ROM έχει συνολικά μέγεθος 6KBytes = 4KBytes + 2KBytes = (2^11 + 2^12) · 8bits → χρειαζόμαστε 13 bits. Άρα, η μνήμη ROM χρησιμοποιεί τα bits A0-A12.
Η RAM έχει συνολικά μέγεθος 10KBytes = 8Kbytes + 2KBytes = (2^13 + 2^11)*8bits → χρειαζόμαστε 14 bits. Άρα, η RAM χρησιμοποιεί τα bits A0-A13.

[1] Πίνακας τιμών για τρισταθή πύλη με επίτρεψη θετικής λογικής (πάνω) και αρνητικής λογικής (κάτω).

Από τον παρακάτω πίνακα παρατηρούμε πως 2 μένουν αχρησιμοποίητα, τα A14 και A15, τα οποία αξιοποιούνται από τον αποκωδικοποιητή για την επιλογή του ολοκληρωμένου της υλοποίησης, ως ανεστραμμένες είσοδοι ενεργοποίησης $\overline{E1}$ και $\overline{E2}$ του αποκωδικοποιητή. Τα υπόλοιπα bits χρησιμοποιούνται για την επιλογή μνήμης.

Σελιδοποιούμε κατά 2K (συνολικά 8 σελίδες των 2K), 1 σελίδα αντιστοιχεί στην ROM1-2K, 2 σελίδες η ROM2-4K, 1 σελίδα η SRAM1-2K και 4 σελίδες η SRAM2-8K. Συνεπώς, επιλέγω τις γραμμές A11-A13 ώς 3-άδα εισόδων στον αποκωδικοποιητή 3 σε 8.

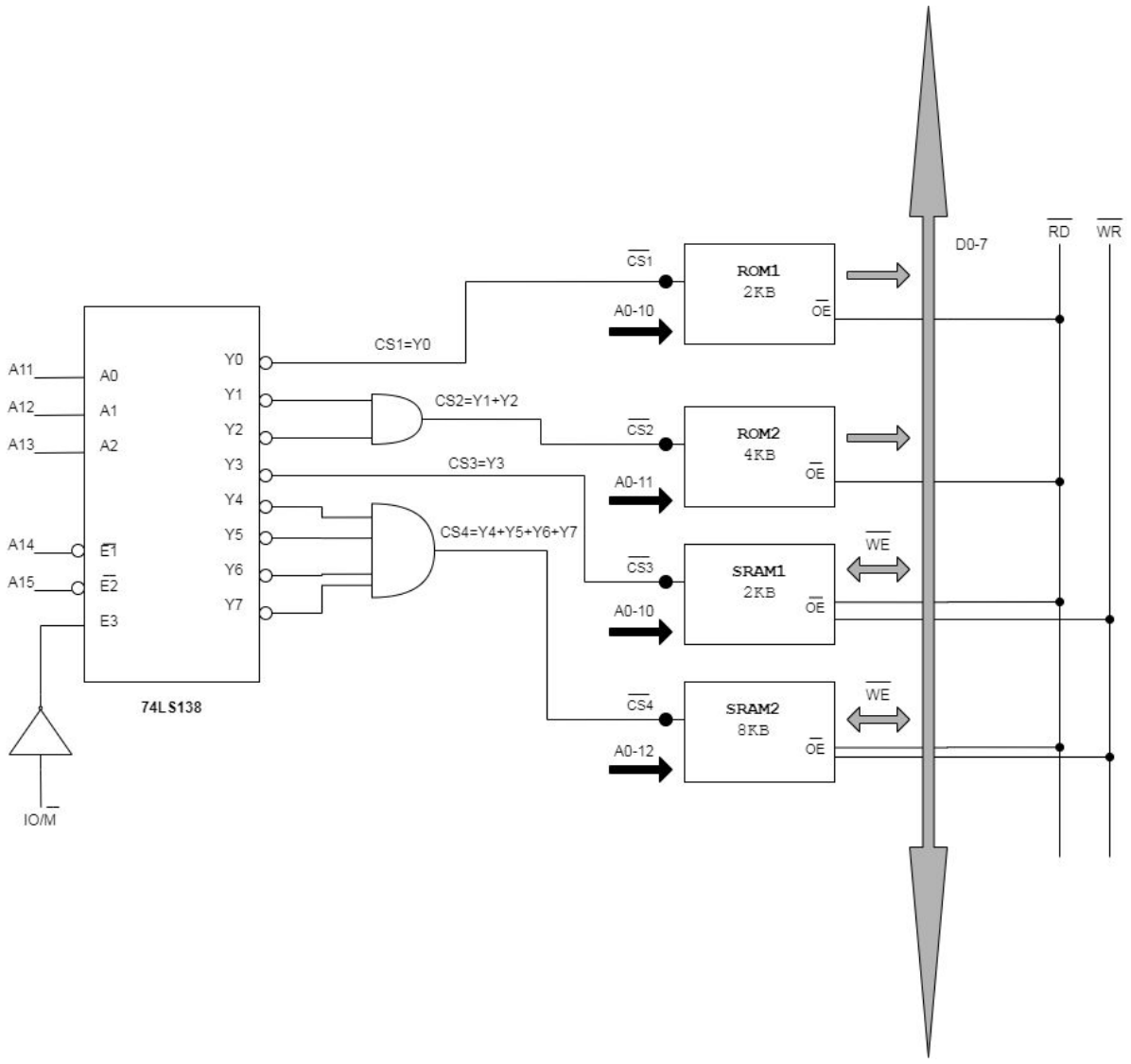
Προφανώς επειδή αναφερόμαστε σε μνήμη χρειαζόμαστε και το $IO/\overline{M}$ από το διάδρομο ελέγχου. (όταν αναφερόμαστε σε μνήμη θέλουμε 0, ενώ για E/E θέλουμε 1 σαν είσοδο)

- **ROM1:** 2K = 2^{11}, άρα χρησιμοποιούμε 11 γραμμές διευθύνσεων, A0-A10
- **ROM2:** 4K = 2^{12}, άρα χρησιμοποιούμε 12 γραμμές διευθύνσεων, A0-A11
- **RAM1:** 2K = 2^{11}, άρα χρησιμοποιούμε 11 γραμμές διευθύνσεων, A0-A10
- **RAM2:** 8K = 2^{13}, άρα χρησιμοποιούμε 13 γραμμές διευθύνσεων, A0-A12

Ο χάρτης μνήμης για RAM και ROM είναι ο εξής:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Address	Memory
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H	ROM1-2Kx 8
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	07FFH	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0800H	ROM2-4Kx 8
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	SRAM1-2K x8
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	SRAM2-8K x8
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH	

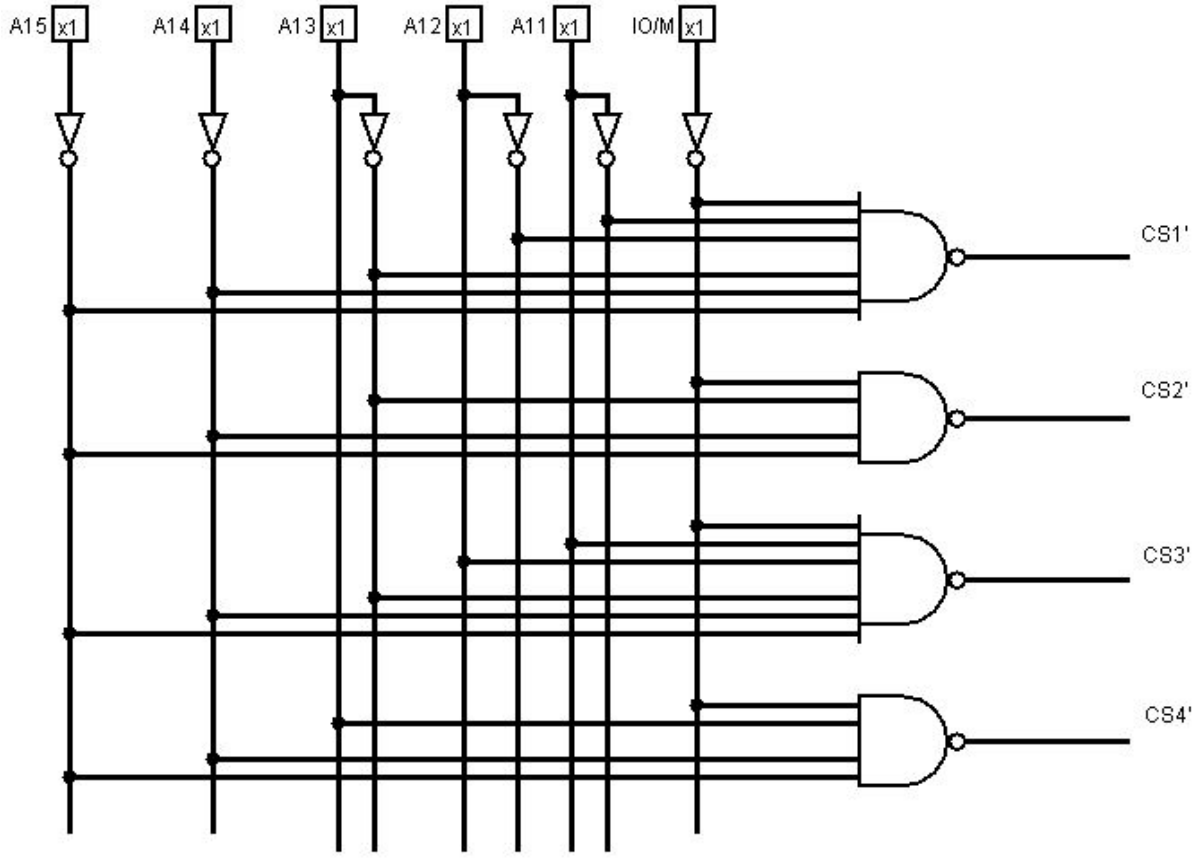
α)χρήση αποκωδικοποιητή 3:8 (74LS138) και λογικές πύλες,



(β) μόνο λογικές πύλες

Να σημειωθεί ότι επειδή έγινε χρήση του προγράμματος logisim τα συμπληρώματα $\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$, $\overline{CS4}$ αντικαταστάθηκαν με τα CS1', CS2', CS3', CS4' λόγω της αδυναμίας του λογισμικού να αποτυπώσει την "-".

Αποτυπώνουμε **μόνο τον αποκωδικοποιητή** σε λογικές πύλες καθώς τα υπόλοιπα στοιχεία (κουτιά ROM, RAM κτλ) δεν αλλάζουν.



- **ROM1:** $\overline{CS1} = \overline{\overline{A_{11}} \cdot \overline{A_{12}} \cdot \overline{A_{13}} \cdot \overline{A_{14}} \cdot \overline{A_{15}} \cdot \overline{M}}$, αφού $A_{11} = A_{12} = A_{13} = A_{14} = A_{15} = 0$
- **ROM2:** $\overline{CS2} = \overline{\overline{A_{13}} \cdot \overline{A_{14}} \cdot \overline{A_{15}} \cdot \overline{M}}$, αφού $A_{13} = A_{14} = A_{15} = 0$
- **RAM1:** $\overline{CS3} = \overline{A_{11} \cdot A_{12} \cdot \overline{A_{13}} \cdot \overline{A_{14}} \cdot \overline{A_{15}} \cdot \overline{M}}$, αφού $A_{13} = A_{14} = A_{15} = 0$ και $A_{11} = A_{12} = 1$
- **RAM2:** $\overline{CS4} = \overline{A_{13} \cdot \overline{A_{14}} \cdot \overline{A_{15}} \cdot \overline{M}}$, αφού $A_{14} = A_{15} = 0$ **και** $A_{13} = 1$

7η Άσκηση:

Εφόσον μας παρέχονται:

- ROM των 16KBytes
- RAMs των 4Kbytes
- RAMs των 8Kbytes

και έχουμε ανάγκη για συνολικά 16KB ROM (4KB και 8KB) και για συνολικά 16KB RAM, **θα τεμαχίσουμε την ROM σε δύο μέρη (4 και 8KB),**

και θα χρησιμοποιήσουμε και τις 2 RAMS που έχουμε στη διάθεση μας, και πρέπει αμέσως μετά το τέλος της διεύθυνσης της RAM 4KB, να ξεκινά η RAM 8KB.

Στο σύστημα μνήμης γίνεται χρήση του μΕ 8085, άρα για την αναπαράσταση δεδομένων απαιτούνται 8 bits και για τις διευθύνσεις το πολύ 16 bits. Οι 3 γραμμές ελέγχου είναι οι WR(εγγραφή στη μνήμη), RD (ανάγνωση από τη μνήμη) και $IO/\overline{M}$ (είσοδος-έξοδος(1) ή μνήμη(0)). Η διεύθυνση της συγκεκριμένης θέσης μνήμης δηλώνεται στο bus διευθύνσεων.

- Η **ROM** έχει συνολικά μέγεθος 16KBytes = $2^{14}\cdot$ 8bits → χρειαζόμαστε 14 bits. Άρα, η μνήμη ROM χρησιμοποιεί τα bits **A0-A13**.
- Η **RAM1- 4KBx8** έχει συνολικά μέγεθος 4KBytes = $2^{12}\cdot$ 8bits → χρειαζόμαστε 12 bits. Άρα, η RAM χρησιμοποιεί τα bits **A0-A11**.
- Η **RAM2- 8KBx8** έχει συνολικά μέγεθος 8KBytes = $2^{13}\cdot$ 8bits → χρειαζόμαστε 13 bits. Άρα, η RAM χρησιμοποιεί τα bits **A0-A12**.

Από τον παρακάτω πίνακα παρατηρούμε πως 1 μένει αχρησιμοποίητο, το **A15**, το οποίο αξιοποιείται από τον αποκωδικοποιητή για την επιλογή του ολοκληρωμένου της υλοποίησης, ως ανεστραμμένη είσοδος ενεργοποίησης $\overline{E2}$ του αποκωδικοποιητή. Τα υπόλοιπα bits χρησιμοποιούνται για την επιλογή μνήμης.

Σελιδοποιούμε κατά 4K (συνολικά 7 σελίδες των 4K), 1 σελίδα αντιστοιχεί στην ROM1-2K, 1 σελίδα η RAM1-4K, 2 σελίδες η RAM2-8K και 3 σελίδες η ROM1-12K. Συνεπώς, επιλέγω τις γραμμές A11-A13 ώς 3-άδα εισόδων στον αποκωδικοποιητή 3 σε 8.

Προφανώς επειδή αναφερόμαστε σε μνήμη χρειαζόμαστε και το $IO/\overline{M}$ από το διάδρομο ελέγχου. (όταν αναφερόμαστε σε μνήμη θέλουμε 0, ενώ για Ε/Ε θέλουμε 1 σαν είσοδο)

Τα Chip Enables έχουν ώς εξής:

- $\overline{CE1} = \overline{Y_0}\cdot\overline{Y_4}\cdot\overline{Y_5}\cdot\overline{Y_6}$
- $\overline{CE2} = \overline{Y_1}$
- $\overline{CE3} = \overline{Y_2}\cdot\overline{Y_3}$

Για την θύρα εισόδου, ένα εξωτερικό κύκλωμα αποκωδικοποίησης συνδυάζει τα σήματα $\overline{RD}$ και $IO/\overline{M}$ και τη διεύθυνση της θύρας και παράγει έναν μοναδικό παλμό επιλογής συσκευής εισόδου, για την θύρα εισόδου.

Ομοίως, για την θύρα εξόδου δεδομένων, εξωτερικό λογικό κύκλωμα αποκωδικοποίησης συνδυάζει τα σήματα $\overline{WR}$ και $IO/\overline{M}$ και τη διεύθυνση της θύρας εξόδου και δημιουργεί έναν μοναδικό παλμό επιλογής συσκευής εξόδου για την θύρα εξόδου. **Επειδή εδώ χρειαζόμαστε μόνο μία θύρα εισόδου και μία θύρα εξόδου, προφανώς δεν χρειάζεται αποκωδικοποιητής διευθύνσεων.** Το σήμα ελέγχου $IO/\overline{M}$ συνδυάζεται με το σήμα $\overline{RD}$ για να δημιουργήσει ένα σήμα ενεργοποίησης εισόδου $\overline{E-IN}$ και με το σήμα $\overline{WR}$ για να δημιουργήσει ένα σήμα ενεργοποίησης εξόδου $\overline{E-OUT}$.

Για την υλοποίηση της πόρτας εξόδου, χρησιμοποιείται ένας μανδαλωτής (ολοκληρωμένο 74LS373), ενώ για την υλοποίηση της πόρτας εισόδου χρησιμοποιείται ένας tri-state buffer (ολοκληρωμένο 74LS540).

Ο χάρτης μνήμης του μΥ-Σ 8085 είναι ο κάτωθι:

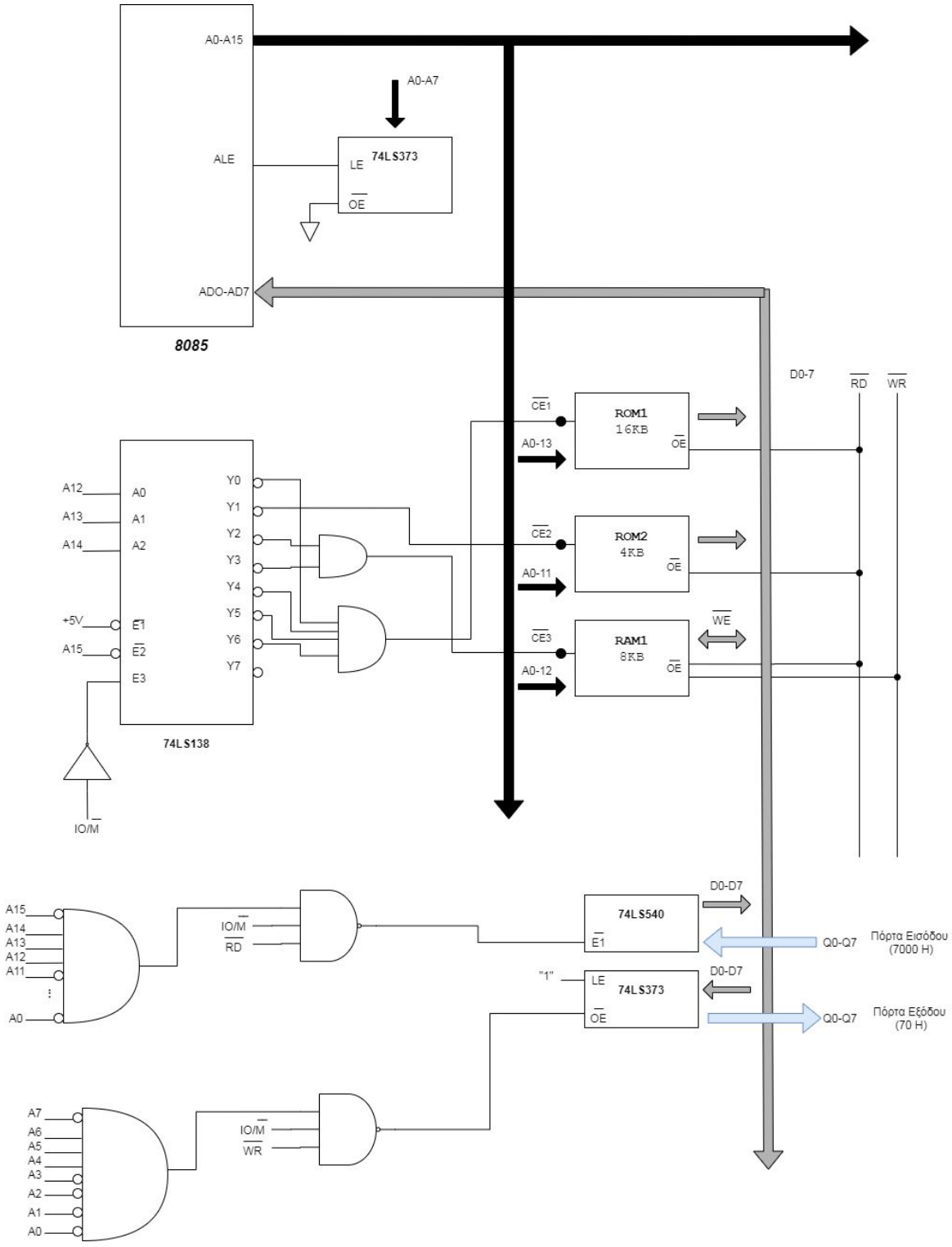
A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Address	Memory
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H	ROM1 FIRST 4KB 8
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0FFFH	
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000H	RAM1-4KBx8
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH	
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H	RAM2-8KBx8
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFFH	
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000H	ROM1 LAST 12KB
0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	6FFFH	

και για τις θύρες Ε/Ε:

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	Address	I/O
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	7000H	Θύρα εισόδου
0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	70H	Θύρα εξόδου

Όπως αναφέρθηκε και νωρίτερα, για τις θύρες θα χρησιμοποιήσουμε μόνο λογικές πύλες στη σχεδίασή μας. Συγκεκριμένα για την είσοδο, θα χρειαστούμε εκτός από τα σήματα RD και $IO/\overline{M}$, τις γραμμές διευθύνσεων A0-A15 (για 7000H) με ανεστραμμένες όσες αντιστοιχούν στο λογικό "0". Ομοίως, για την έξοδο, θα χρειαστούμε εκτός από τα σήματα WR και $IO/\overline{M}$, τις γραμμές διευθύνσεων A0-A7 (για 70H) με ανεστραμμένες όσες αντιστοιχούν στο λογικό "0".

Το πλήρες μΥ-Σ 8085 είναι το εξής:


A0-A15
A0-A7
ALE
74LS373
LE
OE
ADO-AD7
8085
D0-7
RD
WR
CE1
ROM1
16KB
A0-13
OE
A12
A13
A14
A0
A1
A2
Y0
Y1
Y2
Y3
Y4
Y5
Y6
Y7
CE2
ROM2
4KB
A0-11
+5V
A15
E1
E2
E3
74LS138
WE
CE3
RAM1
8KB
A0-12
IO/M
D0-D7
74LS540
Q0-Q7
Πόρτα Εισόδου
(7000 H)
"1"
Πόρτα Εξόδου
(70 H)
A11
A7
A6
A5
A4
A3