ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Συστήματα Μικροϋπολογιστών

2η ομάδα ασκήσεων

Διδάσκων:

Δ. Σούντρης

Ομάδα:

Ειρήνη Δόντη
ΑΜ 03119839

6ο εξάμηνο

Αθήνα 2022

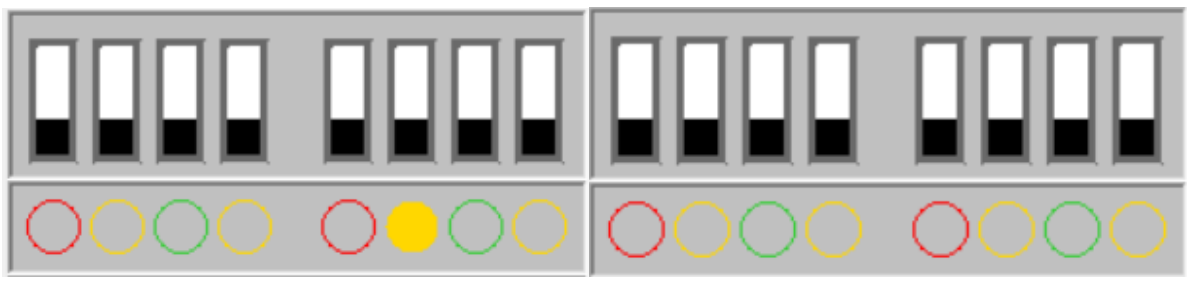
Περιεχόμενα:

1η άσκηση

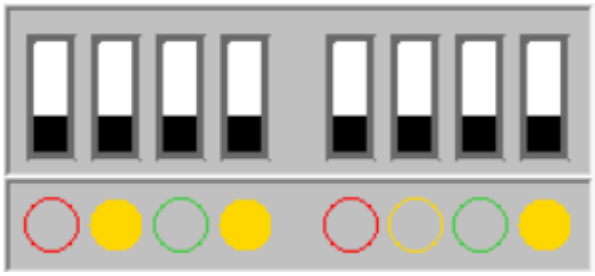
(α) Αν μεταγλωττίσουμε τα πρόγραμμα, προκύπτουν, με σωστό τρόπο, οι παρακάτω θέσεις μνήμης στις οποίες αποθηκεύονται οι ζητούμενοι αριθμοί.

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

(β) Τρέχουμε το πρόγραμμα και τυπώνουμε εναλλάξ το D και το E, με την τιμή τους να καθορίζεται από τα αναμμένα και τα σβηστά LED. Η τιμή του D που τυπώνεται είναι 00000100 και η τιμή του E είναι 00000000. Δηλαδή, η τιμή του DE στο δυαδικό σύστημα είναι 0000010000000000 και στο δεκαεξαδικό ισοδυναμεί με 0400H, ενώ στο δεκαδικό με 1024, όσο δηλαδή είναι και το πλήθος των μηδενικών των αριθμών 0-255.



(γ) Υπολογίζουμε το πλήθος από τους παραπάνω αριθμούς που είναι μεταξύ των αριθμών 0-255. Αποθηκεύουμε το αποτέλεσμα στον καταχωρητή C και επιστρέφουμε την τιμή του μέσω των LED.



Ο C έχει τιμή 01010001, δηλαδή 81 στο δεκαδικό σύστημα. Άρα, έχουμε 81 αριθμούς που ικανοποιούν την παραπάνω συνθήκη. Έτσι, επιβεβαιώνεται η ορθότητα του προγράμματος, εφόσον εξ αρχής γνωρίζαμε ότι το πληθος αυτών των αριθμών είναι 81.

Ο κώδικας που χρησιμοποιήθηκε στην παρούσα άσκηση είναι ο ακόλουθος:

```
;EXERCISE 1

;a

START:
       IN 10H
       MVI A,FFH   ;A valus wew want to store
       LXI H,0900H ;L=00H and H=09H

SAVE:
       MOV M,A   ;store number in memory
       INX H     ;HL=HL+1
       DCR A     ;A=A-1
       CPI 00H   ;if A>0 move to flag save
       JNZ SAVE

;b

       LXI D,0000H ;counter DE -> (if 0 digit (CY=0) -> increment DE)
       MVI C,FFH ;C = 1111 1111

ZEROS:
       MVI B,00H

COUNT_ZEROS:
       MOV A,C ;store number
       RRC
       JC NO_ZERO ;if the digit is 1 (CY=1), -> don't increase the counter -> go to NO_Z
       INX D ;found 0 digit (CY=0) -> increment counter DE

NO_ZERO:
       MOV C,A
       INR B
       MOV A,B
       CPI 08H ;if there are other digits -> check the next one
       JNZ COUNT_ZEROS ;else go to COUNT_ZEROS
       MOV A,C
       CPI 00H ;check if it is the last number
       JZ ZEROS_FINISHED ;if it is the last -> go to ZEROS_FINISHED
       DCR C ;else go to the next number
       JMP ZEROS ;do the same procedure for this number
```

```
ZEROS_FINISHED:
       ;print D or E
       ;MOV A,D
       ;MOV A,E
       CMA
       STA 3000H


;c

       MVI C,00H ;counter -> if 20H<=A<=70H -> counter++
       MVI A,FFH

CHECK:
       CPI 20H ;compare with 20
       JC FALSE_CONDITION ;if A<20 go to FALSE_CONDITION
       CPI 71H ;compare with 71
       JNC FALSE_CONDITION ;if A>70 (A>=71) go to FALSE_CONDITION
       INR C ;20<=A<=70 -> increment counter C

FALSE_CONDITION:
       CPI 00H ;check if the number is the last one
       JZ TOTAL_NUMBERS ;if it is the last -> go to TOTAL_NUMBERS
       DCR A ;else go to the next number
       JMP CHECK ;do the same procedure for this number

TOTAL_NUMBERS:
       ;print C
       MOV A,C
       CMA
       STA 3000H

RST 1

END
```

2η άσκηση

Παρακάτω, παρατίθεται το ζητούμενο πρόγραμμα, το οποίο εκτελέστηκε μέσω της δοσμένης προσομοίωσης:

```
BEGIN:
       LXI B,411BH ; DELAY=0.2 SEC
       MVI E,4BH   ; E= 75 FOR 15 SEC DELAY
       MVI A,00H
       MVI H,00H

       CALL CHECK_LSB    ; CHECK FOR LSB
       JC BEGIN          ; IF LSB IS ON REPEAT

LABEL1:
       CALL CHECK_LSB    ; IF LSB IS OFF THEN REPEAT
       JNC LABEL1

LABEL2:
       CALL CHECK_LSB    ; IF LSB IS ON THEN REPEAT
       JC LABEL2

; WHEN DIP SWITCH LSB IS OFF

INMOVE:
       CPI 04H   ; IF A>=4 CHANGE EXIT
       JC INNEXT ; ELSE MOVE TO FLAG INNEXT
       CALL CHANGE

INNEXT:
       CALL EXIT
       CALL CHECK_LSB ; CHECK FOR LSB
       JC MOVE1       ; IF LSB IS ON MOVE TO FLAG MOVE1
       DCR E
       JZ END15
       JMP INMOVE

; WHEN DIP SWITCH LSB IS ON

MOVE1:
       CPI 04H       ; IF A>=4 CHANGE EXIT
       JC NEXT1      ; ELSE CHANGE EXIT
       CALL CHANGE

NEXT1:
       CALL EXIT
       CALL CHECK_LSB ; CHECK FOR LSB
       JC NEXT2       ; IF LSB IS ON MOVE TO FLAG NEXT2
       MVI E,4BH
       JMP MOVE1


NEXT2:
       DCR E
       JZ END15
       JMP MOVE1

; END OF 15 SEC

END15:
       MVI E,4BH
       MVI A,FFH
       STA 3000H
       MVI A,00H
       MVI H,00H
       CALL CHECK_LSB
       JC LABEL2
       JMP LABEL1
```

```
; SUBROUTINE DELAY (PAGE 69)- DELAY: 12(BC-1) + 15.5 µSEC

DELAY:
       DCX B   ; 6 STATES
       MOV A,B ; 4 STATES
       ORA C   ; 4 STATES
       JNZ DELAY ; 7/10 STATES
       RET       ; 10 STATES

; SUBROUTINE CHECK DIP SWITCH LSB

CHECK_LSB:
       MOV D,A
       LDA 2000H
       RAR       ; IF LSB IS ON THEN CY=1 ELSE CY=0
       MOV A,D
       RET

; SUBROUTINE LED LIGHT SWITCH

CHANGE:
       MOV A,H ; CHANGE EXIT
       CMA
       MOV H,A
       MVI A,00H
       RET

; SUBROUTINE EXIT

EXIT:
       MOV D,A
       MOV A,H
       STA 3000H
       CALL DELAY ; DELAY PROCESS
       MOV A,D
       INR A
       RET

END
```

3η άσκηση

Παρακάτω, παρατίθενται τα ζητούμενα προγράμματα, τα οποία εκτελέστηκαν μέσω της δοσμένης προσομοίωσης:

(i)

```
; Exercise 3 (i)

BEGIN:
       LDA 2000H
       MVI C,00H
       MVI B,00H  ; THESIS COUNTER
       CPI 00H    ; IF ENTRY IS 0
       JZ BEGIN   ; => REPEAT

LABEL:
       INR B
       MOV D,A    ; SAVE A
       MOV A,B
       CPI 09H    ; IF B >= 9 THEN
       JNC LAST   ; MOVE TO LAST FLAG
       MOV A,D    ; RESTORE A
       RAL        ; LEFT ROTATION
       JC FLAG    ; IF B < 9 THEN MOVE TO FLAG
       JMP LABEL  ; JUMP TO LABEL

FLAG:
       CALL DO_FLAG  ; CALL SUBROUTINE
       CALL SAVE     ; CALL SUBROUTINE

LAST:
       CALL EXIT  ; CALL SUBROUTINE
       JMP BEGIN  ; JUMP TO START

; CREATE CARRIER

CAR:
       MOV D,A    ; SAVE A
       MVI A,01H  ; LOAD 1
       CPI 05H    ; A - 5
       MOV A,D    ; SAVE D
       RET        ; STOP SUBROUTINE

; PUSS IN EXIT

SAVE:
       MOV D,A    ; SAVE A
       MOV A,C    ; SAVE C
       CMA        ; COMPLEMENT OF A
       STA 3000H  ; A IS STORED IN 3000H
       MOV A,D    ; SAVE A
       RET        ; STOP SUBROUTINE
```

```
; SUBROUTINE EDIT C

DO_FLAG:
       MOV D,A     ; SAVE A
       MOV A,C     ; SAVE C
       CALL CAR    ; CALL SUBROUTINE ONE
       RAR         ; MOVE RIGHT
       MOV C,A     ; SAVE A
       MOV A,D     ; SAVE D IN A
       DCR B       ; B <- B - 1
       JNZ DO_FLAG ; IF CY = 0, JUMP TO DO_C
       RET         ; STOP SUBROUTINE

; SUBROUTINE EDIT EXIT (DIP SWITCH=OFF)

EXIT:
       LDA 2000H   ; MOVE THE INDEX OF 2000H IN A
       CPI 00H     ; IF A <> 0
       JNZ END0    ; JUMP TO END
       MVI A,FFH   ; 2's COMPLEMENT IS -1
       STA 3000H   ; MOVE A IN 3000H

END0:
       RET         ; STOP SUBROUTINE
END
```

(ii)

```
;  Exercise 3 (ii)


BEGIN:
       LXI B,A2C2H ; BC <- 41666
       MVI D,04H ; D <- 4
       CALL KIND ; A <- ENTRY - READ KEYWORD
       ADI 00H   ; IF A = 0
       JZ BEGIN  ; READ AGAIN - JUMP TO START
       CPI 09H   ; IF A >= 9
       JNC BEGIN ; READ AGAIN - JUMP TO START
       CPI 05H   ; IF A >=5
       JNC MOVE_MSB   ; JUMP TO MSB
       MVI E,F0H
       JMP MOVE_LSB
MOVE_MSB:
       MVI E,0FH
MOVE_LSB:
       CALL EXIT
       JMP BEGIN
EXIT:
       MOV A,E
       STA 3000H
       CALL DELAY
       MVI A,FFH
       STA 3000H
       CALL DELAY
       DCR D
       JZ BEGIN
       JMP EXIT

;DELAY SUBROUTINE - DELAY = 12(BC-1) + 15.5 µSEC

DELAY:
       DCX B     ; 6 STATES
       MOV A,B   ; 4 STATES
       ORA C     ; 4 STATES
       JNZ DELAY ;7/10 STATES
       RET       ; 10 STATES

END
```

(iii)

```
BEGIN:
       IN 10H       ;NO PROTECTION MEMORY
       LXI H,0A00H ; INITIAL MEMORY BLOCK
       MVI B,04H    ; REPEAT

LABEL1:
       MVI M,10H    ; SAVE " "
       INX H
       DCR B
       JNZ LABEL1

; CHECK LINE 0

LINE0:
       MVI A,FEH ; DOOR SCAN 11111110
       STA 2800H
       LDA 1800H ; READ COLUMNS OF KEYS
       ANI 07H   ; KEEP 3 LSB
       MVI C,86H ; C IS MAYBE PASSWORD
       CPI 06H   ; IF A=00000110 (1ST COLUMN BUTTON)
       JZ SHOW   ; PROMOTE CODE
       MVI C,85H ; THE SAME FOR OTHER BUTTONS
       CPI 05H   ; IF A=00000101
       JZ SHOW   ; PROMOTE CODE

; CHECK LINE 1

LINE1:
       MVI A,FDH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,84H
       CPI 06H    ; RUN
       JZ SHOW
       MVI C,80H
       CPI 05H    ; FETCH REG
       JZ SHOW

       MVI C,82H
       CPI 03H    ; FETCH ADDRESS
       JZ SHOW
```

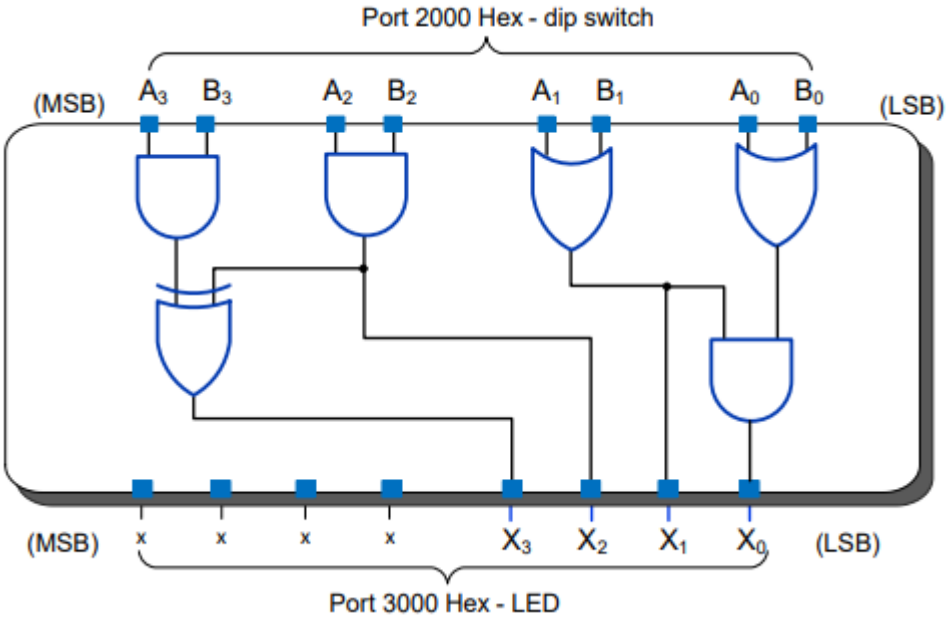
```
; CHECK LINE 2

LINE2:
       MVI A,FBH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,00H
       CPI 06H     ;0
       JZ SHOW
       MVI C,83H
       CPI 05H     ; STORE/INCR
       JZ SHOW

       MVI C,81H
       CPI 03H     ; DECR
       JZ SHOW

; CHECK LINE 3

LINE3:
       MVI A,F7H
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,01H   ; 1
       CPI 06H
       JZ SHOW
       MVI C,02H   ; 2
       CPI 05H
       JZ SHOW

       MVI C,03H   ; 3
       CPI 03H
       JZ SHOW
```

```
; CHECK LINE 4

LINE4:
       MVI A,EFH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,04H
       CPI 06H    ;4
       JZ SHOW
       MVI C,05H
       CPI 05H    ;5
       JZ SHOW

       MVI C,06H
       CPI 03H    ;6
       JZ SHOW

; CHECK LINE 5

LINE5:
       MVI A,DFH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,07H
       CPI 06H    ;7
       JZ SHOW
       MVI C,08H
       CPI 05H    ;8
       JZ SHOW

       MVI C,09H
       CPI 03H    ;9
       JZ SHOW
```

```
; CHECK LINE 6

LINE6:
       MVI A,BFH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,0AH
       CPI 06H    ;A
       JZ SHOW
       MVI C,0BH
       CPI 05H    ;B
       JZ SHOW

       MVI C,0CH
       CPI 03H    ;C
       JZ SHOW

; CHECK LINE 7

LINE7:
       MVI A,7FH
       STA 2800H
       LDA 1800H
       ANI 07H
       MVI C,0DH
       CPI 06H   ; D
       JZ SHOW
       MVI C,0EH
       CPI 05H   ; E
       JZ SHOW

       MVI C,0FH
       CPI 03H   ; F
       JZ SHOW

       JMP BEGIN ; IF THE BUTTON IS NOT PRESSED
                 ; JUMP TO START

; CODE PROMOTION

SHOW:
       LXI H,0A04H
       MOV A,C     ; KEY A
       ANI 0FH     ; KEEP 4 LSBs
       MOV M,A     ; MOVE TO 0A04H - 5TH DIGIT OF
                   ; 7-SEGMENT DISPLAY

       INX H       ; NEXT MEMORY THESIS
       MOV A,C
       ANI F0H     ; KEEP 4 MSBs
       RLC
       RLC         ; MAKE THEM LSBs
       RLC
       RLC
       MOV M,A     ; SAVE TH3 6TH DIGIT OF
                   ; 7-SEGMENT DISPLAY
       LXI D,0A00H ; MOVE BLOCK 0A00H-0A05H
       CALL STDM   ; WHERE DCD READS
       CALL DCD    ; VISUALISE
       JMP BEGIN   ; REPEAT

END
```

4η άσκηση

Υλοποιούμε το παρακάτω κύκλωμα:



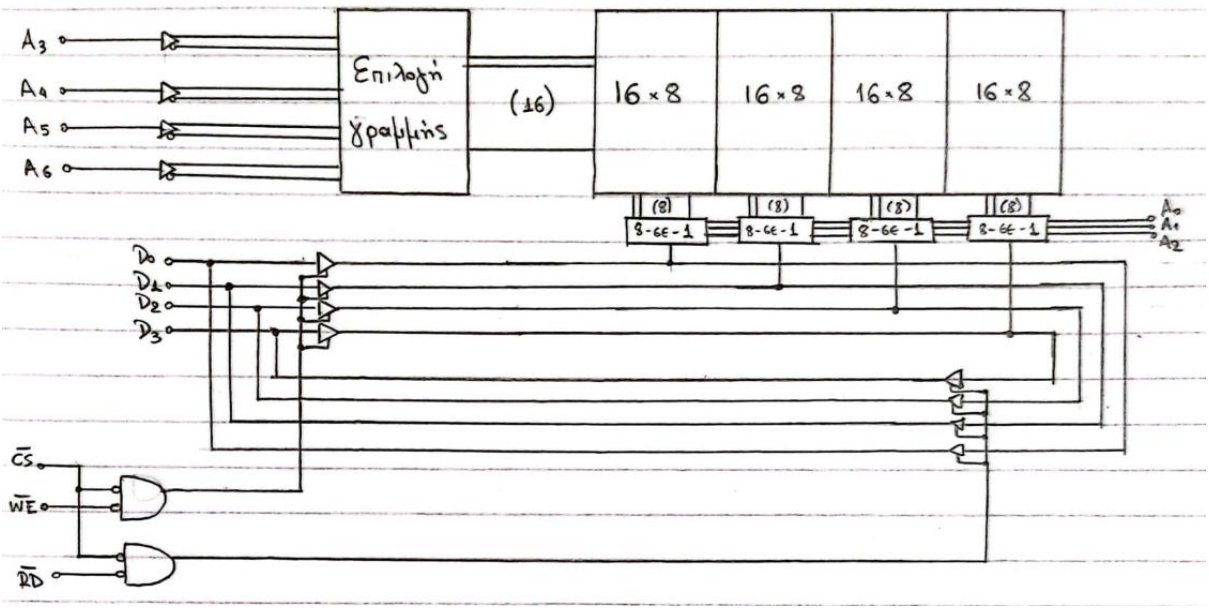
Χρησιμοποιούμε τις εντολές RRC και RLC για να εξασφαλίσουμε τις επιθυμητές θέσεις για τα ψηφία. Μέσω των εντολών ANI απομονώνουμε κάθε φορά τα ψηφία που θέλουμε και ανάμεσα σε αυτά τα στοιχεία εκτελούμε τις λογικές πράξεις για κάθε πύλη. Για την επίτευξη αυτού, χρησιμοποιούμε τις εντολές ORA (λογική πράξη OR), ANA (λογική πράξη AND), XRA (λογική πράξη XOR).

Πιο συγκεκριμένα, δημιουργούμε τον ακόλουθο κώδικα:

```
;EXERCISE 4

BEGIN:

	LDA 2000H
	MOV H,A	;SAVE INITIAL VALUE OF A

MAIN_CODE:

	ANI 01H	;ISOLATE LSB
	MOV L,A	;SAVE LSB IN L
	MOV A,H	;SAVE INITIAL VALUE OF A IN A

	ANI 02H	;ISOLATE 2ND BIT FROM LEFT
	RRC	;MOVE ONE THESIS RIGHT
	ORA L	;A0 OR B0
	MOV D,A	;SAVE A0 OR B0 IN D
	MOV A,H	;SAVE INITIAL VALUE OF A IN A

	ANI 04H	;ISOLATE 3RD BIT FROM LEFT
	RRC
	RRC
	MOV L,A	;SAVE VALUE OF 3RD BIT IN L
	MOV A,H	;SAVE INITIAL VALUE OF A IN A

	ANI 08H	;ISOLATE 4TH BIT FROM LEFT
	RRC
	RRC
	RRC
	ORA L	;A1 OR B1
	MOV L,A	;SAVE A1 OR B1 IN L
	ANA D	;(A0 OR B0)AND(A1 OR B1)
	MOV D,A	;SAVE (A0 OR B0)AND(A1 OR B1) IN D
	MOV A,L	;SAVE A1 OR B1 IN A
	RLC
	ORA D	;A = A OR D
	MOV D,A	;D=A
	MOV A,H	;SAVE INITIAL VALUE OF A IN A

	ANI 10H	;ISOLATE 5TH BIT FROM LEFT
	RRC
	RRC
	RRC
	RRC
	MOV L,A	;SAVE VALUE OF 5TH BIT IN L
	MOV A,H	;SAVE INITIAL VALUE OF A IN A
```

```
        ANI 20H   ;ISOLATE 6TH BIT FROM LEFT
        RLC
        RLC
        RLC
        ANA L     ;A2 AND B2
        MOV L,A   ;SAVE A2 AND B2 IN L
        RLC
        RLC
        ORA D     ;A = A OR D
        MOV D,A   ;D=A
        MOV A,H   ;SAVE INITIAL VALUE OF A IN A

        ANI 40H   ;ISOLATE 7TH BIT FROM LEFT
        RLC
        RLC
        MOV B,A   ;SAVE VALUE OF 7TH BIT IN B
        MOV A,H   ;SAVE INITIAL VALUE OF A IN A

        ANI 80H   ;SAVE VALUE OF 8TH BIT IN B
        RLC
        ANA B     ;A3 AND B3
        XRA L     ;(A3 AND B3) XOR (A2 AND B2)
        RLC
        RLC
        RLC
        ORA D     ;A = A OR D
        MOV D,A   ;D=A



EXIT:
        CMA
        STA 3000H
        JMP BEGIN
END
```

5η άσκηση



Παραπάνω παρουσιάζεται η εσωτερική δομή μιας μνήμης SRAM 128x4 bit. Τα σήματα A3, A4, A5 και A6 καθορίζουν ποιες από τις 16 γραμμές θα επιλεγούν για εγγραφή και για ανάγνωση. Τα D0, D1, D2 και D3 είναι γραμμές δεδομένων που συνδέονται με τον πίνακα της μνήμης μέσω τεσσάρων πολυπλεκτών 8-σε-1. Οι πολυπλέκτες καθορίζουν σε ποια από τις 8 στήλες του καθε μπλοκ (συνδυασμένη με κάποια γραμμή) θα γίνει η ανάγνωση ή η εγγραφή.

Για παράδειγμα, αν είχαμε μια διεύθυνση A0…A6 = 010 0001, επιλέγεται η 2η γραμμή και η 1η τριάδα στη μνήμη.

Για να γίνει εγγραφή στη μνήμη πρέπει να είναι ενεργοποιημένα τα σήματα CS’ και WE’. Πριν την είσοδο τους στην πύλη αντιστρέφονται, οπότε πρέπει αρχικά να είναι και τα δύο 0. Τότε, η έξοδος της αντίστοιχης πύλης AND γίνεται 1 και ενεργοποιούνται οι τέσσερις απομονωτές αριστερά. Μετά την ενεργοποίηση, τα D0, D1, D2, D3 πηγαίνουν στους τέσσερις πολυπλέκτες. Στην συνέχεια, μέσω των σημάτων A0, A1 και A2 καθορίζεται σε ποια από τις 8 στήλες του κάθε μπλοκ θα γίνει η εγγραφή. Τα σήματα A3, A4, A5 και A6 επιλέγουν μία από τις 16 γραμμές μέσω ενός αποκωδικοποιητή. Οπότε, έχουμε ταυτόχρονη εγγραφή των D0, D1, D2 και D3 στα 4 κελιά της μνήμης.

Για να γίνει ανάγνωση από τη μνήμη πρέπει να είναι ενεργοποιημένα τα σήματα CS’ και RD’. Πριν την είσοδο τους στην πύλη αντιστρέφονται, οπότε πρέπει αρχικά να είναι και τα δύο 0. Τότε, η έξοδος της αντίστοιχης πύλης AND γίνεται 1 και ενεργοποιούνται οι τέσσερις απομονωτές δεξιά. Μετά την ενεργοποίηση, οι έξοδοι των τεσσάρων πολυπλεκτών πηγαίνουν στα D0, D1, D2 και D3. Η στήλη του μπλοκ από το οποίο γίνεται η ανάγνωση καθορίζεται από τα σήματα A0, A1 και A2, ενώ η γραμμή από τα A3, A4, A5 και A6. Κάθε πολυπλέκτης διαβάζει ένα bit από το αντίστοιχο μπλοκ και τα συνολικά 4 bits που διαβάστηκαν τοποθετούνται στα D0, D1, D2 και D3.

6η άσκηση

Σχεδιάζουμε ένα σύστημα μνήμης, το οποίο αποτελείται από τα εξής ολοκληρωμένα (που βρίσκονται σε διαδοχικές θέσεις χωρίς κενά):
ROM1: 2k x 8 bit = 2k byte
ROM2: 4k x 8 bit = 4k byte
SRAM1: 2k x 8 bit = 2k byte
SRAM2: 8k x 8 bit = 8k byte
Χρησιμοποιούμε τον μικροεπεξεργαστή 8085, οπότε η αναπαράσταση των δεδομένων απαιτεί λέξεις των 8 bits = 1 byte.

Σχεδιάζουμε το χάρτη μνήμης του συστήματος:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	address	MEM
0	0	**0**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	1000H	ROM1
0	0	**0**	**1**	**0**	1	1	1	1	1	1	1	1	1	1	1	17FFH	2k
0	0	**0**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	1800H	ROM2
0	0	**1**	**0**	**0**	1	1	1	1	1	1	1	1	1	1	1	27FFH	4k
0	0	**1**	**0**	**1**	0	0	0	0	0	0	0	0	0	0	0	2800H	SRAM1
0	0	**1**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	2FFFH	2k
0	0	**1**	**1**	**0**	0	0	0	0	0	0	0	0	0	0	0	3000H	SRAM2
0	1	**0**	**0**	**1**	1	1	1	1	1	1	1	1	1	1	1	4FFFH	8k

Τα bit και A_{15} και A_{14} χρησιμοποιούνται ως επίτρεψη στον αποκωδικοποιητή 3 σε 8 [74LS138] για το (α) ερώτημα και σε λογικές πύλες για το (β) ερώτημα. Τα bits A_{11}, A_{12}, A_{13} χρησιμοποιούνται για την επιλογή του ολοκληρωμένου με τον εξής τρόπο:

ROM1: $A_{13}\,A_{12}\,A_{11} = 010$

ROM2: $A_{13}\,A_{12}\,A_{11} = 011$ ή 100

SRAM1: $A_{13}\,A_{12}\,A_{11} = 101$

SRAM2: $A_{13}\,A_{12}\,A_{11} = 110$ ή 001 (ή 000 ή 111)

Θεωρούμε εισόδους τα bits A_{11}, A_{12}, A_{13} και τα bits A_{15} και A_{14}. Μόνο όταν $A_{15} = A_{14} = 0$ και $A_{11} \cap A_{14} = 1$, θα μπορέσει να γίνει επιλογή μνήμης μέσω του αποκωδικοποιητή. Οι έξοδοι του αποκωδικοποιητή, προκύπτουν σύμφωνα με τις παρακάτω σχέσεις:

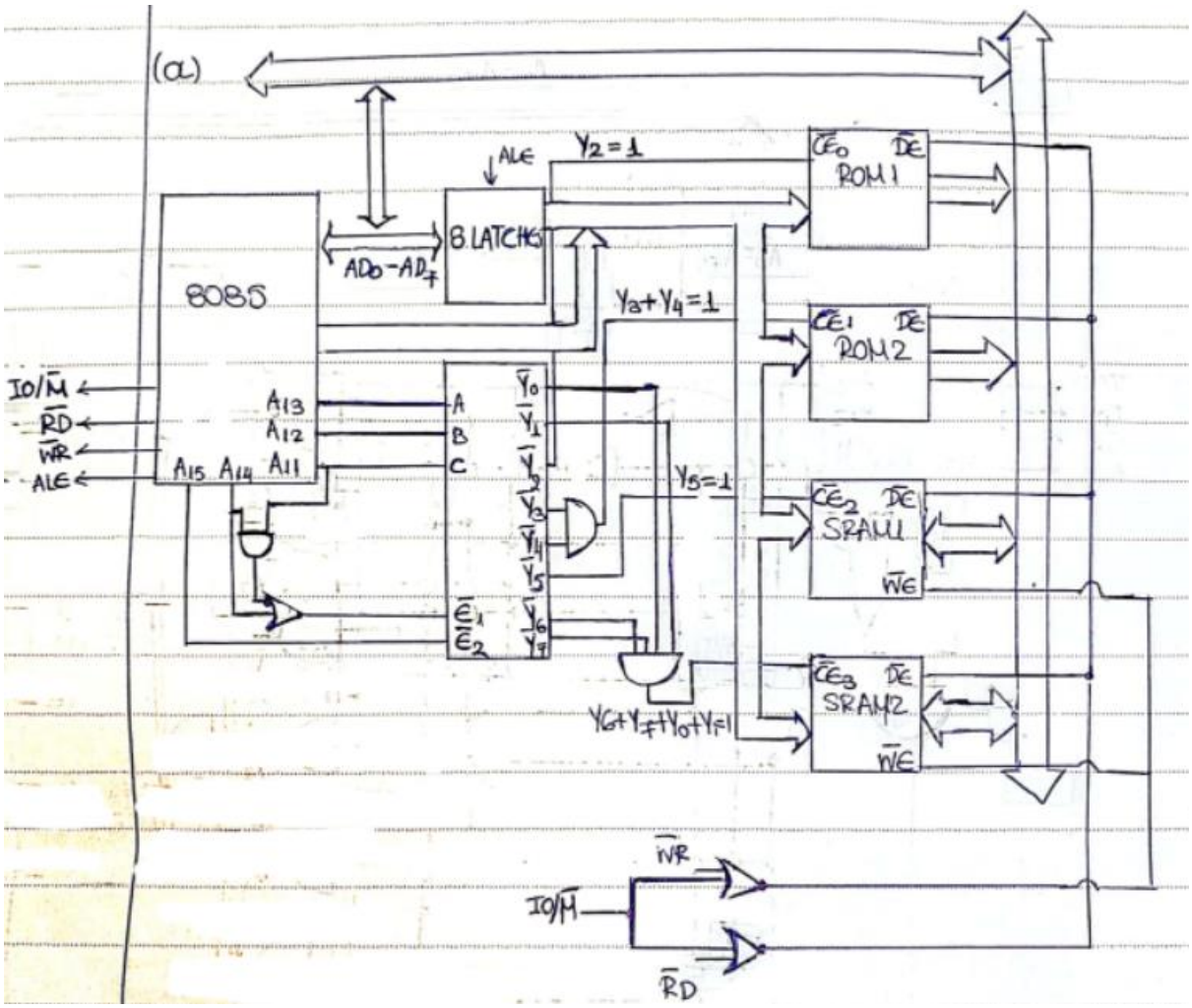
$CE_0 = 1$ όταν $A_{13}\,A_{12}\,A_{11} = 010$, δηλαδή $Y_2 = 1$ ή $\neg Y_2 = 0$

$CE_1 = 1$ όταν $A_{13}\,A_{12}\,A_{11} = 011$ ή 100, δηλαδή $Y_3 + Y_4 = 1$ ή $\neg Y_4 \neg Y_3 = 0$

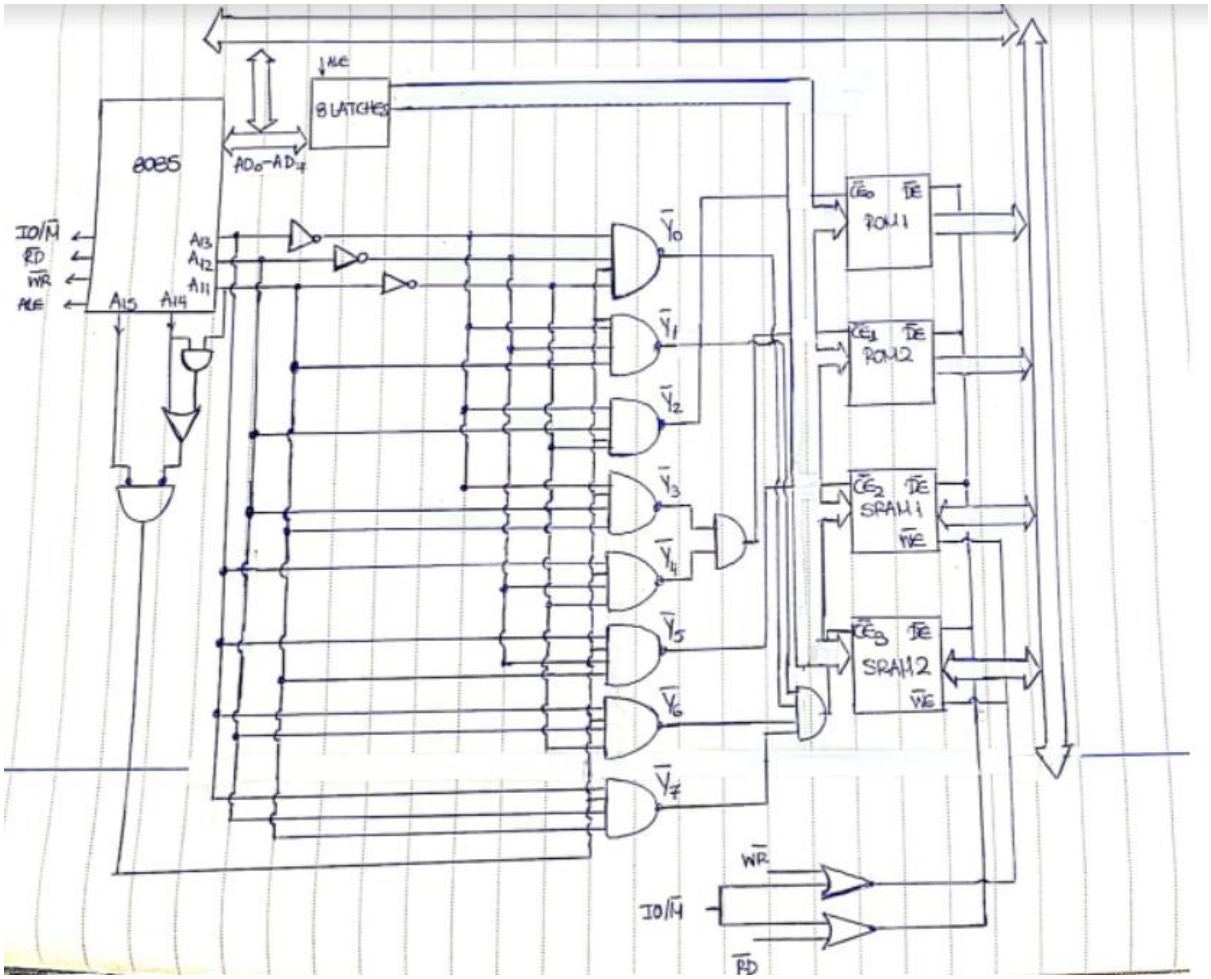
$CE_2 = 1$ όταν $A_{13}\,A_{12}\,A_{11} = 101$, δηλαδή $Y_5 = 1$ ή $\neg Y_5 = 0$

$CE_3 = 1$ όταν $A_{13}\,A_{12}\,A_{11} = 110$ ή 001 (ή 000 ή 111), δηλαδή $Y_0 + Y_1 + Y_6 + Y_7 = 1$ ή $\neg Y_0 \neg Y_1 \neg Y_6 \neg Y_7 = 0$

(α) Παρακάτω, απεικονίζεται ο τελικός σχεδιασμός του συστήματος μνήμης με τον έναν αποκωδικοποιητή 3:8 (74LS138) και λογικές πύλες:



(β) Παρακάτω, απεικονίζεται ο τελικός σχεδιασμός του συστήματος μνήμης με λογικές πύλες:



7η άσκηση

Για τον σχεδιασμό του χάρτη μνήμης μπορούμε να χρησιμοποιήσουμε:

- μια μνήμη ROM των 16Kbytes εκ των οποίων τα 4 θα καταναλωθούν στις θέσεις μνήμης 1000H - 1FFFH και τα υπόλοιπα 12 στις θέσεις μνήμης 5000H - 7FFFH.
- μια μνήμη RAM των 4Kbytes η οποία θα καταναλωθεί στις θέσεις μνήμης 2000H - 2FFFH.
- μια μνήμη RAM των 8Kbytes η οποία θα καταναλωθεί στις θέσεις μνήμης 3000H - 4FFFH.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Memory
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1000H	ROM 4K
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFFH	ROM 4K
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000H	RAM 4K
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFFH	RAM 4K
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3000H	RAM 8K
0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	4FFFH	RAM 8K
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	5000H	ROM 12K
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFFH	ROM 12K

Παρατηρούμε ότι το bit A_{15} έχει για όλες τις θέσεις μνήμης τιμή 0, άρα δεν χρησιμοποιείται για τον προσδιορισμό καμμίας θέσης μνήμης . Συνεπώς, χρησιμοποιείται ως επίτρεψη στον αποκωδικοποιητή που επιλέγει ολοκληρωμένο για προσπέλαση.

Μπορούμε να παρατηρήσουμε ότι:

- Το πλήθος των λέξεων που αποθηκεύονται στην ROM ισούται με $16K = 2^{14}$ λέξεις, και άρα απαιτούνται 14 bits για τη διευθυνσιοδότηση τους. Δηλαδή $A_0 - A_{13}$.
- Το πλήθος των λέξεων που αποθηκεύονται στην 1η RAM ισούται με $4K = 2^{12}$ λέξεις, και άρα απαιτούνται 12 bits για τη διευθυνσιοδότηση τους. Δηλαδή $A_0 - A_{11}$.
- Το πλήθος των λέξεων που αποθηκεύονται στην 2η RAM ισούται με $8K = 2^{13}$ λέξεις, και άρα απαιτούνται 13 bits για τη διευθυνσιοδότηση τους. Δηλαδή $A_0 - A_{12}$.

Από τον πίνακα που φτιάξαμε παραπάνω, παρατηρούμε ότι υπάρχουν κάποιοι συνδυασμοί των bit A_{12} , A_{13} και A_{14} που μπορούν να προσδιορίσουν μοναδικά τις περιοχές της μνήμης που αντιστοιχούν στο κάθε ολοκληρωμένο. Άρα, μπορούν να χρησιμοποιηθούν για την κατάλληλη επιλογή κάποιας εκ των ROM, RAM1 και RAM2.
Συγκεκριμένα:

- Οι συνδυασμοί $A_{14}A_{13}A_{12} = 001,\ A_{14}A_{13}A_{12} = 101,\ A_{14}A_{13}A_{12} = 111$ οδηγούν στην επιλογή της ROM.
- Ο συνδυασμός $A_{14}A_{13}A_{12} = 010$ οδηγεί στην επιλογή της 1ης RAM.
- Οι συνδυασμοί $A_{14}A_{13}A_{12} = 011,\ A_{14}A_{13}A_{12} = 100$ ($A_{14}A_{13}A_{12} = 000,\ A_{14}A_{13}A_{12} = 110$) οδηγούν στην επιλογή της 2ης RAM.

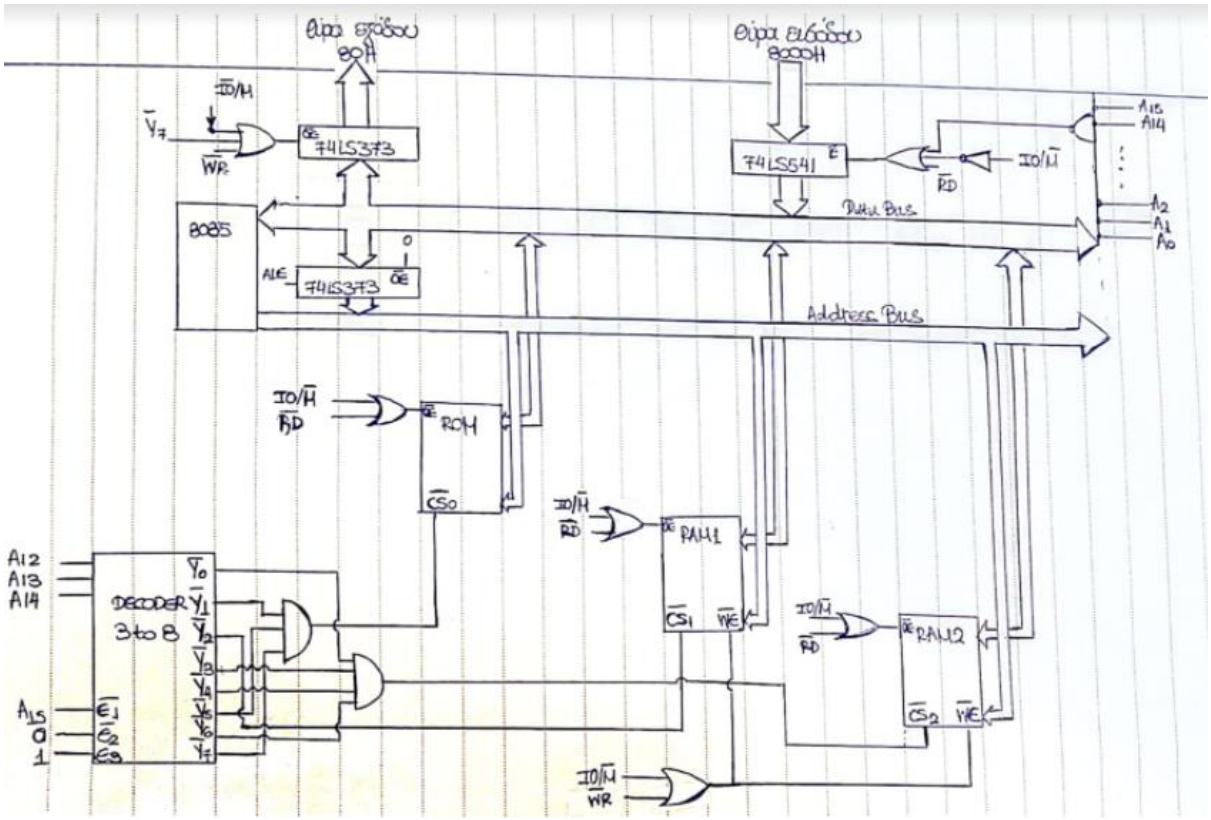
Χρησιμοποιούμε τις παρακάτω σχέσεις για να συνδέσουμε την έξοδο του αποκωδικοποιητή. Από τον χάρτη μνήμης προκύπτει ότι:

$CS_0 = 1$ ή αλλιώς $(CS_0)' = 0$ όταν $A_{13}A_{12}A_{11} = 001,\ A_{13}A_{12}A_{11} = 101,\ A_{13}A_{12}A_{11} = 111$.
Δηλαδή $Y_1 + Y_5 + Y_7 = 1 \Rightarrow Y_1'Y_5'Y_7' = 0$
Άρα $(CS_0)' = Y_1'Y_5'Y_7'$.

$CS_1 = 1$ ή αλλιώς $(CS_1)' = 0$ όταν $A_{13}A_{12}A_{11} = 010$. Δηλαδή $Y_2 = 1 \Rightarrow Y_2' = 0$
Άρα $(CS_1)' = Y_2'$

$CS_2 = 1$ ή αλλιώς $(CS_2)' = 0$ όταν $A_{13}A_{12}A_{11} = 011,\ A_{13}A_{12}A_{11} = 100$. Δηλαδή $Y_3 + Y_4 = 1 \Rightarrow Y_3'Y_4' = 0$
Άρα $(CS_2)' = Y_3'Y_4'$

Σχεδιάζουμε το μΥ-Σ 8085, όπως απεικονίζεται παρακάτω:


8085
74LS373
74LS541
ROM
RAM1
RAM2
DECODER
3 to 8
Address Bus
Data Bus
ALE
IO/M
RD
WR
A12
A13
A14
A15
A2
A1
A0