Συστήματα Μικροϋπολογιστών – 2η Σειρά Ασκήσεων

Κυριακόπουλος Γιώργος – el18153
Τζελέπης Σεραφείμ – el18849

1η Άσκηση:

```
START:
    IN 10H
    CALL MAIN
    RST 1

MAIN:
    MVI A,00H    ; Πρώτος αριθμός που θα μπει στη μνήμη.
    LXI H,0900H  ; Αρχική διεύθυνση αποθήκευσης των αριθμών.
    LXI B,0000H  ; Μετρητής μονάδων.
    MVI D,00H    ; Μετρητής αριθμών από 10H έως 60H περιλαμβανομένων.

FIRST:
    CALL SECOND  ; Εκκίνηση καταμέτρησης μονάδων.
    CPI FFH      ; Έλεγχος εάν έχουμε φτάσει στον τελευταίο αριθμό.
    JZ ENDED     ; Έξοδος εάν είμαστε στον τελευταίο αριθμό, ο οποίος
                 ; δεν ανήκει άλλωστε στο εύρος που μας νοιάζει.
    CPI 10H      ; Έλεγχος εάν ο αριθμός μας είναι μικρότερος από 10H.
    JC OUTSIDE   ; Διαδικασία που δεν αυξάνει τον μετρητή D.
    CPI 61H      ; Έλεγχος εάν ο αριθμός είναι μεγαλύτερος από 60H.
    JC INSIDE    ; Διαδικασία που αυξάνει τον μετρητή D και συνεχίζει.
    JMP OUTSIDE  ; Αλλιώς, πάμε στην διαδικασία χωρίς αύξηση.

INSIDE:
    INR D        ; Αύξησε τον μετρητή D κατά 1, εφόσον ο αριθμός
                 ; ανήκει στο επιθυμητό εύρος.
OUTSIDE:
    MOV M,A      ; Τοποθέτηση του αριθμού στη μνήμη.
    INX H        ; Αύξηση της θέσης μνήμης για επόμενη αποθήκευση.
    INR A        ; Αύξηση του αριθμού μας.
    JMP FIRST    ; Επανάληψη της συνολικής διαδικασίας.
ENDED:
    MOV M,A      ; Τοποθέτηση του αριθμού στη μνήμη.
```

```
    RET         ; Επιστροφή στην βασική συνάρτηση START.

SECOND:
    RAR         ; Δεξιά περιστροφή με χρήση του carry flag.
    JNC COUNT1  ; Εάν είναι μονάδα, πήγαινε στον επόμενο έλεγχο.
    INX B       ; Αύξηση του μετρητή μονάδων.

COUNT1:
    RAR         ; Όμοια για τα υπόλοιπα ψηφία.
    JNC COUNT2
    INX B

COUNT2:
    RAR         ; Όμοια για τα υπόλοιπα ψηφία.
    JNC COUNT3
    INX B

COUNT3:
    RAR         ; Όμοια για τα υπόλοιπα ψηφία.
    JNC COUNT4
    INX B

COUNT4:
    RAR         ; Όμοια για τα υπόλοιπα ψηφία.
    JNC COUNT5
    INX B

COUNT5:
    RAR         ; Όμοια για τα υπόλοιπα ψηφία.
    JNC COUNT6
    INX B

COUNT6:
    RAR         ; Όμοια για τα υπόλοιπα ψηφία.
    JNC COUNT7
    INX B

COUNT7:
    RAR         ; Όμοια για τα υπόλοιπα ψηφία.
    JNC COUNT8
    INX B

COUNT8:
```

```
    RAR         ; Τελευταία περιστροφή ώστε να επαναφερθεί το A.
    RET         ; Επιστροφή στην κεντρική συνάρτηση FIRST.

END
```

Με απλή εποπτεία των καταχωρητών αλλά και της μνήμης, αφού τρέξουμε το πρόγραμμα, παρατηρούμε ότι ορθά οι θέσεις 0900 έως 09FF αντιστοιχούν στους αριθμού 0-255 δηλαδή στους δεκαεξαδικούς 0-FF, ο καταχωρητής BC έχει το αποτέλεσμα 0400 δηλαδή το δεκαδικό 1024, που είναι το σωστό αποτέλεσμα που αναμένουμε και τέλος, ο καταχωρητής D έχει ορθά το δεκαεξαδικό 51 που είναι το δεκαδικό 81, το οποίο αποτελεί τους αριθμούς που περιέχονται μεταξύ 10H και 60H δηλαδή μεταξύ 16 και 96 σε δεκαδικό σύστημα.

2η Άσκηση:

```
    MVI A,FFH   ; Αρχική έξοδος για όλα σβηστά (αρνητική λογική).
    STA 3000H   ; Τοποθέτηση της εξόδου στα LEDs.
    MVI D,C8H   ; Μετρητής 200 καθυστερήσεων διάρκειας B.
    LXI B,0064H ; Διάρκεια καθυστέρησης ίση με 100ms (1/10 second).

START:
    LDA 2000H   ; Διάβασμα της εισόδου των dip-switches.
    RAL         ; Απομόνωση του MSB στο Carry Flag.
    JNC OFF1    ; Έλεγχος εάν είναι αρχικά σβηστό.
    JMP START   ; Επανάληψη μέχρι να σβήσει αρχικά, ώστε να έχουμε
                ; την αρχική επιθυμητή ακολουθία OFF->ON->OFF.
OFF1:
    LDA 2000H   ; Διάβασμα της εισόδου των dip-switches.
    RAL         ; Απομόνωση του MSB στο Carry Flag.
    JC ON1      ; Εάν ήταν ανοιχτό τότε περίμενε να σβήσει.
    JMP OFF1    ; Αλλιώς επανάλαβε μέχρι να ανάψει.

ON1:
    LDA 2000H   ; Διάβασμα της εισόδου των dip-switches.
    RAL         ; Απομόνωση του MSB στο Carry Flag.
    JNC TURNON  ; Έλεγχος εάν έσβησε και πάλι, οπότε ανάβουμε έξοδο.
    JMP ON1     ; Αλλιώς επανάλαβε μέχρι να σβήσει.

TURNON:
    MVI A,00H   ; Ετοιμασία εξόδου ώστε να ανάψουν όλα τα LEDs.
    STA 3000H   ; Τοποθέτηση στην έξοδο και άναμμα των LEDs.

OFF2:
    LDA 2000H   ; Διάβασμα της εισόδου των dip-switches.
    RAL         ; Απομόνωση του MSB στο Carry Flag.
    JC ON2      ; Εάν ξανά ανάψει πήγαινε στην ανάλογη διαδικασία.
    CALL DELB   ; Κάλεσμα της καθυστέρησης.
    DCR D       ; Μείωση του μετρητή των καθυστερήσεων.
    MOV A,D     ; Μετακίνηση του μετρητή στον καταχωρητή A.
    CPI 00H     ; Έλεγχος του μετρητή καθυστερήσεων.
    JNZ OFF2    ; Εάν δεν είναι μηδέν τότε επανάλαβε τη διαδικασία.
    JMP TURNOFF ; Εάν είναι μηδέν, σβήσε τα φώτα και ξεκίνα από την
                ; αρχή περιμένοντας να ανάψει πάλι το MSB.
ON2:
    LDA 2000H   ; Διάβασμα της εισόδου των dip-switches.
    RAL         ; Απομόνωση του MSB στο Carry Flag.
```

```
    JNC RESTART ; Εάν ξανασβήσει επανάφερε τον μετρητή καθυστερήσεων.
    CALL DELB   ; Εάν δεν έχει σβήσει πάλι, κάλεσε την καθυστέρηση.
    DCR D       ; Μείωσε το μετρητή καθυστερήσεων.
    MOV A,D     ; Μετακίνηση του μετρητή στον καταχωρητή A.
    CPI 00H     ; Έλεγχος του μετρητή καθυστερήσεων.
    JNZ ON2     ; Εάν δεν είναι μηδέν τότε επανάλαβε τη διαδικασία.
    JMP TURNOFF ; Εάν είναι μηδέν, σβήσε τα φώτα και ξεκίνα από την
                ; αρχή περιμένοντας να ανάψει πάλι το MSB.
RESTART:
    MVI D,C8H   ; Αρχικοποίησε πάλι το μετρητή στο 200.
    JMP OFF2    ; Πήγαινε στη διαδικασία που περιμένει να ανάψει πάλι
                ; το MSB αφού έχει ήδη προηγουμένως σβήσει.
TURNOFF:
    MVI A,FFH   ; Ετοίμασε την έξοδο για να σβήσουν όλα τα LEDs.
    STA 3000H   ; Σβήσε όλα τα LEDs.
    MVI D,C8H   ; Αρχικοποίησε πάλι το μετρητή στο 200.
    JMP OFF1    ; Πήγαινε στη διαδικασία που περιμένει να ανάψει πάλι
                ; το MSB.
END
```

3η Άσκηση:

i)

```
START:
    LDA 2000H   ; Διάβασμα της εισόδου των dip-switches
    MOV B,A     ; Προσωρινή αποθήκευση
    MVI C,00H   ; Μετρητής για 1-8 bits
    MVI D,01H   ; Πρώτη αρχικοποίηση εξόδου σε περίπτωση που
                ; διαβάσουμε 1 στο LSB.
SEARCH:
    INR C       ; Αύξηση του μετρητή των bits.
    MOV A,B     ; Επαναφορά του τρέχοντος στιγμιότυπου στον A.
    ANI 01H     ; Απομόνωση του τελευταίου bit (LSB).
    JNZ FLASH   ; Εάν αυτό είναι 1, πήγαινε να ανάψεις τα LEDs.

    MOV A,B     ; Επαναφορά του τρέχοντος στιγμιότυπου στον A.
    RRC         ; Δεξιά περιστροφή, ώστε να ελέγξω το επόμενο bit.
    MOV B,A     ; Αποθήκευση του τρέχοντος στιγμιότυπου στον B.
    MOV A,D     ; Φέρε την έξοδο στον A για αλλαγή.
    RLC         ; Κάνε μία αριστερή περιστροφή του 1 που υπάρχει.
    MOV D,A     ; Αποθήκευσε τη νέα έξοδο.
    MOV A,C     ; Φέρε το μετρητή στον A για έλεγχο τερματισμού.
    CPI 08H
    JNZ SEARCH  ; Εάν δεν είναι ακόμα ίσος με 8 επανάλαβε.

NONE:
    MVI A,FFH   ; Εάν ο μετρητής γίνει 8 και δεν έχεις βρει 1, τότε
    STA 3000H   ; βάλε στην έξοδο όλα τα LEDs σβηστά και επανάλαβε
    JMP START   ; από την αρχή τη διαδικασία με νέα είσοδο.

FLASH:
    MOV A,D     ; Μετακίνησε την έξοδο στον καταχωρητή A.
    CMA         ; Συμπλήρωμα ως προς 1, λόγω αρνητικής λογικής.
    STA 3000H   ; Πέρασμα στην έξοδο και άναμμα των LEDs.
    JMP START   ; Επανεκκίνηση της διαδικασίας.

END
```

ii)

```
START:
    MVI D,FFH    ; Αρχικοποίηση της εξόδου.
    CALL KIND    ; Κάλεσε το διάβασμα του πληκτρολογίου.
    CPI 09H      ; Έλεγχος εάν πατήθηκε πλήκτρο μικρότερο του 9.
    JNC NONE     ; Εάν όχι τότε σβήσε όλα τα LEDs.
    CPI 00H      ; Έλεγχος εάν πατήθηκε πλήκτρο μεγαλύτερο του 0.
    JZ NONE      ; Εάν όχι τότε σβήσε όλα τα LEDs.
    MOV B,A      ; Προσωρινή αποθήκευση της εισόδου του πληκτρολογίου.

LOOPER:
    MOV A,B      ; Επαναφορά της εισόδου.
    DCR A        ; Μείωση του αριθμού της εισόδου κατά 1.
    CPI 00H      ; Έλεγχος τερματισμού.
    JZ FLASH     ; Εάν ο A είναι ίσος με 0 τότε ανάβουμε τα LEDs.
    MOV B,A      ; Αποθήκευση του στιγμιότυπου της εισόδου.
    MOV A,D      ; Φέρε την έξοδο στον A για αλλαγή.
    STC          ; Θέσε το Carry Flag ίσο με 1.
    CMC          ; Συμπλήρωμα ως προς ένα για το Carry Flag, άρα 0.
    RAL          ; Αριστερή περιστροφή για να μπει 0 στο τέλος του A.
    MOV D,A      ; Αποθήκευση της τροποποιημένης εξόδου.
    JMP LOOPER   ; Επανάλαβε τη διαδικασία.

NONE:
    MVI A,FFH    ; Εάν δεν πατήθηκε κατάλληλο πλήκτρο τότε σβήσε όλα
    STA 3000H    ; τα LEDs.
    JMP START    ; Ξεκίνα έλεγχο από την αρχή με νέα είσοδο.

FLASH:
    MOV A,D      ; Μετακίνησε την έξοδο στον καταχωρητή A.
    CMA          ; Συμπλήρωμα ως προς 1, λόγω αρνητικής λογικής.
    STA 3000H    ; Άναμμα των ανάλογων LEDs.
    JMP START    ; Ξεκίνα έλεγχο από την αρχή με νέα είσοδο.

END
```

iii)

```
    IN 10H      ; Άρση προστασίας μνήμης.

START:
    LXI H,0A00H ; Αρχή των θέσεων αποθήκευσης.
    MVI B,04H   ; Μετρητής για επανάληψη.

LOOPER:
    MVI M,10H   ; Αποθήκευσε το τίποτα.
    INX H       ; Πήγαινε στην επόμενη θέση μνήμης.
    DCR B       ; Μείωσε τον μετρητή.
    JNZ LOOPER  ; Επανάλαβε τη διαδικασία.

LINE0:
    MVI A,FEH   ; Πόρτα σάρωσης = 11111110 – επιλογή γραμμής.
    STA 2800H
    LDA 1800H   ; Διάβασε τις στήλες των πλήκτρων.
    ANI 07H     ; Απομόνωση των 3 τελευταίων ψηφίων.

    MVI C,86H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 06H     ; Έλεγχος εάν πατήθηκε το πρώτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το INSTR_STEP εμφάνισε το.
    MVI C,85H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 05H     ; Έλεγχος εάν πατήθηκε το δεύτερο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το FETCH PC εμφάνισε το.
                ; Αγνοούμε το τρίτο κουμπί της στήλης 0 (HDWR_STEP).
LINE1:
    MVI A,FDH   ; Πόρτα σάρωσης = 11111101 – επιλογή γραμμής.
    STA 2800H
    LDA 1800H   ; Διάβασε τις στήλες των πλήκτρων.
    ANI 07H     ; Απομόνωση των 3 τελευταίων ψηφίων.

    MVI C,84H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 06H     ; Έλεγχος εάν πατήθηκε το πρώτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το RUN εμφάνισε το.
    MVI C,80H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 05H     ; Έλεγχος εάν πατήθηκε το δεύτερο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το FETCH_REG εμφάνισε το.
    MVI C,82H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 03H     ; Έλεγχος εάν πατήθηκε το τρίτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το FETCH_ADDRS εμφάνισε το.
```

```
LINE2:
    MVI A,FBH   ; Πόρτα σάρωσης = 11111011 - επιλογή γραμμής.
    STA 2800H
    LDA 1800H   ; Διάβασε τις στήλες των πλήκτρων.
    ANI 07H     ; Απομόνωση των 3 τελευταίων ψηφίων.

    MVI C,00H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 06H     ; Έλεγχος εάν πατήθηκε το πρώτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 0 εμφάνισε το.
    MVI C,83H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 05H     ; Έλεγχος εάν πατήθηκε το δεύτερο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το STORE/INCR εμφάνισε το.
    MVI C,81H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 03H     ; Έλεγχος εάν πατήθηκε το τρίτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το DECR εμφάνισε το.

LINE3:
    MVI A,F7H   ; Πόρτα σάρωσης = 11110111 - επιλογή γραμμής.
    STA 2800H
    LDA 1800H   ; Διάβασε τις στήλες των πλήκτρων.
    ANI 07H     ; Απομόνωση των 3 τελευταίων ψηφίων.

    MVI C,01H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 06H     ; Έλεγχος εάν πατήθηκε το πρώτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 1 εμφάνισε το.
    MVI C,02H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 05H     ; Έλεγχος εάν πατήθηκε το δεύτερο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 2 εμφάνισε το.
    MVI C,03H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 03H     ; Έλεγχος εάν πατήθηκε το τρίτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 3 εμφάνισε το.

LINE4:
    MVI A,EFH   ; Πόρτα σάρωσης = 11101111 - επιλογή γραμμής.
    STA 2800H
    LDA 1800H   ; Διάβασε τις στήλες των πλήκτρων.
    ANI 07H     ; Απομόνωση των 3 τελευταίων ψηφίων.

    MVI C,04H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 06H     ; Έλεγχος εάν πατήθηκε το πρώτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 4 εμφάνισε το.
    MVI C,05H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 05H     ; Έλεγχος εάν πατήθηκε το δεύτερο κουμπί της στήλης.
```

```
    JZ DISPLAY  ; Εάν πατήθηκε το 5 εμφάνισε το.
    MVI C,06H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 03H     ; Έλεγχος εάν πατήθηκε το τρίτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 6 εμφάνισε το.

LINE5:
    MVI A,DFH   ; Πόρτα σάρωσης = 11011111 - επιλογή γραμμής.
    STA 2800H
    LDA 1800H   ; Διάβασε τις στήλες των πλήκτρων.
    ANI 07H     ; Απομόνωση των 3 τελευταίων ψηφίων.

    MVI C,07H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 06H     ; Έλεγχος εάν πατήθηκε το πρώτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 7 εμφάνισε το.
    MVI C,08H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 05H     ; Έλεγχος εάν πατήθηκε το δεύτερο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 8 εμφάνισε το.
    MVI C,09H   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 03H     ; Έλεγχος εάν πατήθηκε το τρίτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το 9 εμφάνισε το.

LINE6:
    MVI A,BFH   ; Πόρτα σάρωσης = 10111111 - επιλογή γραμμής.
    STA 2800H
    LDA 1800H   ; Διάβασε τις στήλες των πλήκτρων.
    ANI 07H     ; Απομόνωση των 3 τελευταίων ψηφίων.

    MVI C,0AH   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 06H     ; Έλεγχος εάν πατήθηκε το πρώτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το A εμφάνισε το.
    MVI C,0BH   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 05H     ; Έλεγχος εάν πατήθηκε το δεύτερο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το B εμφάνισε το.
    MVI C,0CH   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 03H     ; Έλεγχος εάν πατήθηκε το τρίτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το C εμφάνισε το.

LINE7:
    MVI A,7FH   ; Πόρτα σάρωσης = 01111111 - επιλογή γραμμής.
    STA 2800H
    LDA 1800H   ; Διάβασε τις στήλες των πλήκτρων.
    ANI 07H     ; Απομόνωση των 3 τελευταίων ψηφίων.
```

```
    MVI C,0DH   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 06H     ; Έλεγχος εάν πατήθηκε το πρώτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το D εμφάνισε το.
    MVI C,0EH   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 05H     ; Έλεγχος εάν πατήθηκε το δεύτερο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το E εμφάνισε το.
    MVI C,0FH   ; Ετοιμασία πιθανού κωδικού για προώθηση στο DISPLAY.
    CPI 03H     ; Έλεγχος εάν πατήθηκε το τρίτο κουμπί της στήλης.
    JZ DISPLAY  ; Εάν πατήθηκε το F εμφάνισε το.

    JMP LINE0   ; Εάν δεν πατήθηκε κανένα κουμπί επανάλαβε τους
                ; ελέγχους από την αρχή.
DISPLAY:
    LXI H,0A04H ; Ετοιμάζουμε τη θέση 0A04H.
    MOV A,C     ; Μεταφέρουμε τον κωδικό του κουμπιού στον A.
    ANI 0FH     ; Απομονώνουμε τα 4 LSBs.
    MOV M,A     ; Τα βάζουμε στην 0A04H (5ο bit 7-segment display).
    INX H       ; Πάμε στην επόμενη θέση μνήμης.
    MOV A,C     ; Επαναφέρουμε τον κωδικό του κουμπιού στον A.
    ANI F0H     ; Απομονώνουμε τα 4 MSBs.
    RRC         ; 4 δεξιές περιστροφές για να έρθουν στη σωστή θέση.
    RRC
    RRC
    RRC
    MOV M,A     ; Τα βάζουμε στην 0A05H (6ο bit 7-segment display).

    LXI D,0A00H ; Φορτώνουμε τις θέσεις 0A00H-0A05H για την DCD.
    CALL STDM   ; Καλούμε την STDM.
    CALL DCD    ; Γίνεται η απεικόνιση στο 7-segment display.

    JMP START   ; Επανεκκίνηση της διαδικασίας από την αρχή.

END
```

4η Άσκηση:

```
START:
    IN 10H
    LDA 2000H   ; Διάβασμα της εισόδου των dip-switches.
    CALL XOR1   ; Κάλεσμα συνάρτησης για έξοδο κάθε πύλης.
    CALL XOR2
    CALL AND1
    CALL AND2
    CALL XOR3
    CALL OR1

    MVI B,00H   ; Προσωρινή κατασκευή της εξόδου.

    LDA 0A04H   ; Διάβασμα από τη μνήμη της εξόδου της XOR3.
    MOV B,A     ; Μετακίνηση της εξόδου της XOR3 (X0) στην έξοδο.

    LDA 0A01H   ; Διάβασμα από τη μνήμη της εξόδου της XOR2.
    RRC         ; Δεξιά περιστροφή ώστε να είναι στην σωστή θέση.
    ADD B       ; Πρόσθεση της εξόδου της XOR2 (X1) στην έξοδο.
    MOV B,A     ; Αποθήκευση της εξόδου στον καταχωρητή B.

    LDA 0A05H   ; Διάβασμα από τη μνήμη της εξόδου της OR1.
    RRC         ; Δύο δεξιές περιστροφές ώστε να είναι στη σωστή
    RRC         ; θέση.
    ADD B       ; Πρόσθεση της εξόδου της OR1 (X2) στην έξοδο.
    MOV B,A     ; Αποθήκευση της εξόδου στον καταχωρητή B.

    LDA 0A03H   ; Διάβασμα από τη μνήμη της εξόδου της AND2.
    RRC         ; Τρεις δεξιές περιστροφές ώστε να είναι στη σωστή
    RRC         ; θέση.
    RRC
    ADD B       ; Πρόσθεση της εξόδου της AND2 (X3) στην έξοδο.
    MOV B,A     ; Αποθήκευση της εξόδου στον καταχωρητή B.

    MOV A,B     ; Μετακίνηση της εξόδου στον καταχωρητή A.
    CMA         ; Συμπλήρωμα ως προς 1, λόγω αρνητικής λογικής.
    STA 3000H   ; Άναμμα των κατάλληλων LEDs.
    JMP START   ; Επανεκκίνηση της διαδικασίας.

XOR1:
    LXI H,0A00H ; Θέση μνήμης για αποθήκευση του αποτελέσματος.
    MOV D,A     ; Προσωρινή αποθήκευση.
```

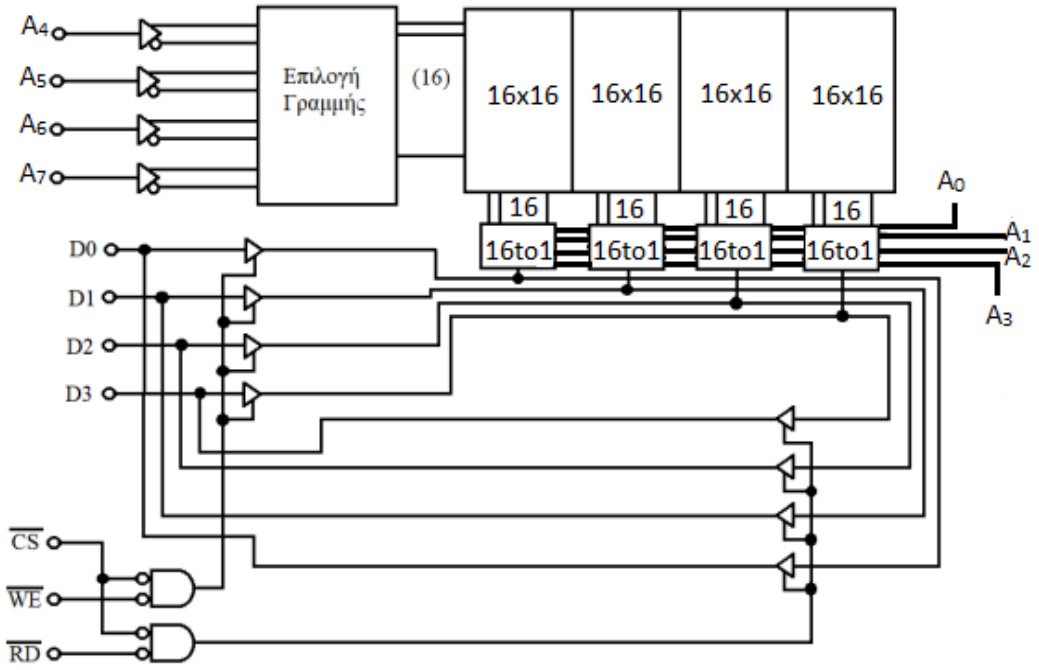
```
    ANI 01H     ; Απομόνωση του 1ου LSB.
    MOV B,A     ; Αποθήκευση του 1ου LSB στον καταχωρητή B.
    MOV A,D     ; Επαναφορά της εισόδου.
    ANI 02H     ; Απομόνωση του 2ου LSB.
    RRC         ; Δεξία περιστροφή ώστε να είναι στην ίδια θέση.
    XRA B       ; XOR με το 1ο LSB.
    MOV M,A     ; Αποθήκευση αποτελέσματος.
    MOV A,D     ; Επαναφορά της εισόδου.
    RET         ; Επιστροφή.

XOR2:
    LXI H,0A01H ; Θέση μνήμης για αποθήκευση του αποτελέσματος.
    MOV D,A     ; Προσωρινή αποθήκευση.

    ANI 04H     ; Απομόνωση του 3ου LSB.
    MOV B,A     ; Αποθήκευση του 3ου LSB στον καταχωρητή B.
    MOV A,D     ; Επαναφορά της εισόδου.
    ANI 08H     ; Απομόνωση του 4ου LSB.
    RRC         ; Δεξία περιστροφή ώστε να είναι στην ίδια θέση.
    XRA B       ; XOR με το 3ο LSB.
    MOV M,A     ; Αποθήκευση αποτελέσματος.
    MOV A,D     ; Επαναφορά της εισόδου.
    RET         ; Επιστροφή.

AND1:
    LXI H,0A02H ; Θέση μνήμης για αποθήκευση του αποτελέσματος.
    MOV D,A     ; Προσωρινή αποθήκευση.

    ANI 10H     ; Απομόνωση του 5ου LSB.
    MOV B,A     ; Αποθήκευση του 5ου LSB στον καταχωρητή B.
    MOV A,D     ; Επαναφορά της εισόδου.
    ANI 20H     ; Απομόνωση του 6ου LSB.
    RRC         ; Δεξία περιστροφή ώστε να είναι στην ίδια θέση.
    ANA B       ; AND με το 5ο LSB.
    MOV M,A     ; Αποθήκευση αποτελέσματος.
    MOV A,D     ; Επαναφορά της εισόδου.
    RET         ; Επιστροφή.

AND2:
    LXI H,0A03H ; Θέση μνήμης για αποθήκευση του αποτελέσματος.
    MOV D,A     ; Προσωρινή αποθήκευση.
```

```
    ANI 40H      ; Απομόνωση του 7ου LSB.
    MOV B,A      ; Αποθήκευση του 7ου LSB στον καταχωρητή B.
    MOV A,D      ; Επαναφορά της εισόδου.
    ANI 80H      ; Απομόνωση του 8ου LSB.
    RRC          ; Δεξία περιστροφή ώστε να είναι στην ίδια θέση.
    ANA B        ; AND με το 7ο LSB.
    MOV M,A      ; Αποθήκευση αποτελέσματος.
    MOV A,D      ; Επαναφορά της εισόδου.
    RET          ; Επιστροφή.

XOR3:
    LDA 0A00H    ; Διάβασμα από τη μνήμη της εξόδου της XOR1.
    MOV B,A      ; Αποθήκευση της εξόδου της XOR1.
    LDA 0A01H    ; Διάβασμα από τη μνήμη της εξόδου της XOR2.
    RRC          ; Δύο δεξιές περιστροφές ώστε να έρθει η δεύτερη
    RRC          ; έξοδος στην ίδια θέση με την πρώτη.
    XRA B        ; XOR των δύο εξόδων.
    STA 0A04H    ; Αποθήκευση αποτελέσματος.
    RET          ; Επιστροφή.

OR1:
    LDA 0A02H    ; Διάβασμα από τη μνήμη της εξόδου της AND1.
    MOV B,A      ; Αποθήκευση της εξόδου της AND1.
    LDA 0A03H    ; Διάβασμα από τη μνήμη της εξόδου της AND2.
    RRC          ; Δύο δεξιές περιστροφές ώστε να έρθει η δεύτερη
    RRC          ; έξοδος στην ίδια θέση με την πρώτη.
    ORA B        ; OR των δύο εξόδων.
    STA 0A05H    ; Αποθήκευση αποτελέσματος.
    RET          ; Επιστροφή.

END
```

5η Άσκηση:

Η λειτουργία ανάγνωσης της μνήμης απαιτεί ενεργοποιημένα τα σήματα CS’ και RD’. Κατά αυτόν τον τρόπο η έξοδος της κάτω πύλης AND, η οποία είναι συνδεμένη με την επίτρεψη στα τρισταθή buffers που “κοιτάνε” προς τα αριστερά, γίνεται 1 και έτσι οι έξοδοι των πολυπλεκτών 16 σε 1 προωθείτε στα D0, D1, D2, D3. Οι πολυπλέκτες διαβάζουν ο καθένας από το αντίστοιχο block ένα bit του οποίου η γραμμή δίνεται από τους επιλογείς A4, A5, A6, A7 και στήλη μέσω των επιλογέων A0, A1, A2, A3 και τα 4bits που θέλουμε να διαβάσουμε από την μνήμη περνιούνται στα D0, D1, D2, D3.

Η λειτουργία εγγραφής της μνήμης απαιτεί ενεργοποιημένα τα σήματα CS’ και WE’. Κατά αυτόν τον τρόπο η έξοδος της πάνω πύλης AND, η οποία είναι συνδεμένη με την επίτρεψη στα τρισταθή buffers που “κοιτάνε” προς τα δεξιά, γίνεται 1 και έτσι τα D0, D1, D2, D3 προωθούνται στους πολυπλέκτες 16 σε 1. Εκεί έχουμε ως επιλογείς τα A0, A1, A2, A3 οπού καθορίζουν σε ποια από τις 16 στήλες του κάθε block θα γίνει η εγγραφή ενώ τα σήματα A4, A5, A6, A7 επιλέγουν μέσω του αποκωδικοποιητή της γραμμή εγγραφής και έτσι γράφονται ταυτόχρονα τα δεδομένα D0, D1, D2, D3 στο αντίστοιχο κελί μνήμης για κάθε block.



6η Άσκηση:

Το σύστημα μνήμης προς σχεδίαση αποτελείται από τα εξής ολοκληρωμένα, που βρίσκονται σε διαδοχικές θέσεις χωρίς κενά :

2 x 2KBytes ROM

1 x 4KBytes ROM

2 x 2KBytes SRAM

Τα οποία συνολικά μας δίνουν την ζητούμενη μνήμη των 8KBytes ROM και 4KBytes RAM.

Ο χάρτης μνήμης θα είναι ο εξής:

Μνήμη	Διεύθυνση	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ROM1 2K	0000 H	***0 0 0 0***	***0*** 0 0 0	0 0 0 0	0 0 0 0
	07FF H	***0 0 0 0***	***0*** 1 1 1	1 1 1 1	1 1 1 1
ROM2 2K	0800 H	***0 0 0 0***	***1*** 0 0 0	0 0 0 0	0 0 0 0
	0FFF H	***0 0 0 0***	***1*** 1 1 1	1 1 1 1	1 1 1 1
ROM3 4K	1000 H	***0 0 0 1***	***0*** 0 0 0	0 0 0 0	0 0 0 0
	1FFF H	***0 0 0 1***	***1*** 1 1 1	1 1 1 1	1 1 1 1
RAM1 2K	2000 H	***0 0 1 0***	***0*** 0 0 0	0 0 0 0	0 0 0 0
	27FF H	***0 0 1 0***	***0*** 1 1 1	1 1 1 1	1 1 1 1
RAM2 2K	2800 H	***0 0 1 0***	***1*** 0 0 0	0 0 0 0	0 0 0 0
	2FFF H	***0 0 1 0***	***1*** 1 1 1	1 1 1 1	1 1 1 1

Άρα όπως φαίνεται έχουμε:

$A_{13}A_{12}A_{11}$= 000 => ROM1

$A_{13}A_{12}A_{11}$= 001 => ROM2

$A_{13}A_{12}A_{11}$= 010 και $A_{13}A_{12}A_{11}$= 011 => ROM3

$A_{13}A_{12}A_{11}$= 100 => RAM1

$A_{13}A_{12}A_{11}$= 101 => RAM2

α) Αν κάνουμε χρήση αποκωδικοποιητή 3:8 (74LS138) και λογικών πυλών θα χρησιμοποιήσουμε τον αποκωδικοποιητή με εισόδους A_{13} A_{12} A_{11} και τα A_{15} A_{14} ως επιτρέψεις άρα οι έξοδοι του αποκωδικοποιητή με τα αντίστοιχα CE(Chip Enable) με τον παρακάτω τρόπο

CE_0 = 1 => $(CE_0)'$ = 0 όταν $A_{13}A_{12}A_{11}$= 000 => Y_0 = 1 => $(Y_0)'$ = 0 => $(CE_0)'$ = $(Y_0)'$

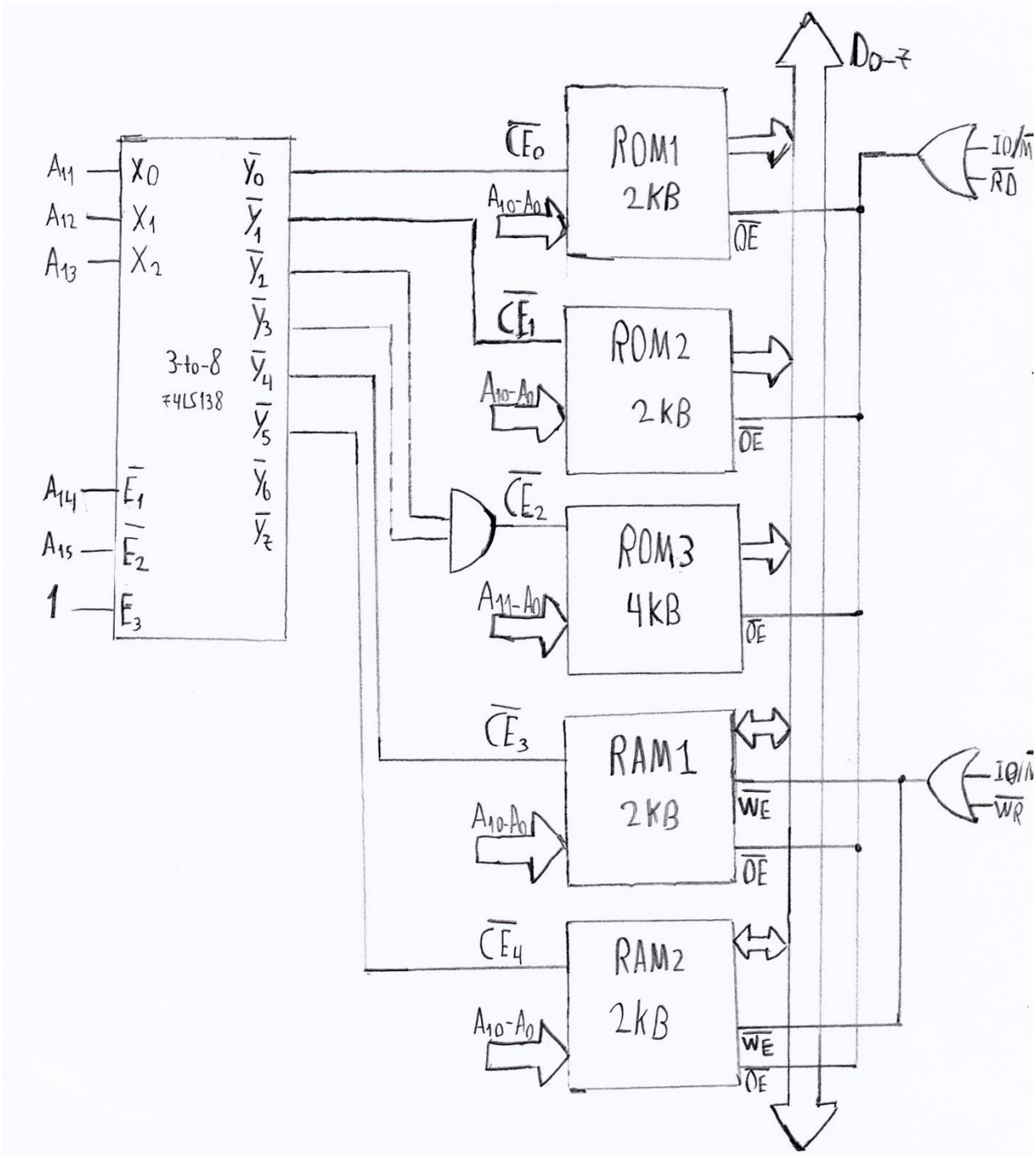
CE_1 = 1 => $(CE_1)'$ = 0 όταν $A_{13}A_{12}A_{11}$= 001 => Y_1 = 1 => $(Y_1)'$ = 0 => $(CE_1)'$ = $(Y_1)'$

$CE_2 = 1 \Rightarrow (CE_2)' = 0$ όταν $A_{13}A_{12}A_{11} = 010$ και $A_{13}A_{12}A_{11} = 011 \Rightarrow Y_2 + Y_3 = 1 \Rightarrow (Y_2)'(Y_3)' = 0 \Rightarrow (CE_2)' = (Y_2)'(Y_3)'$

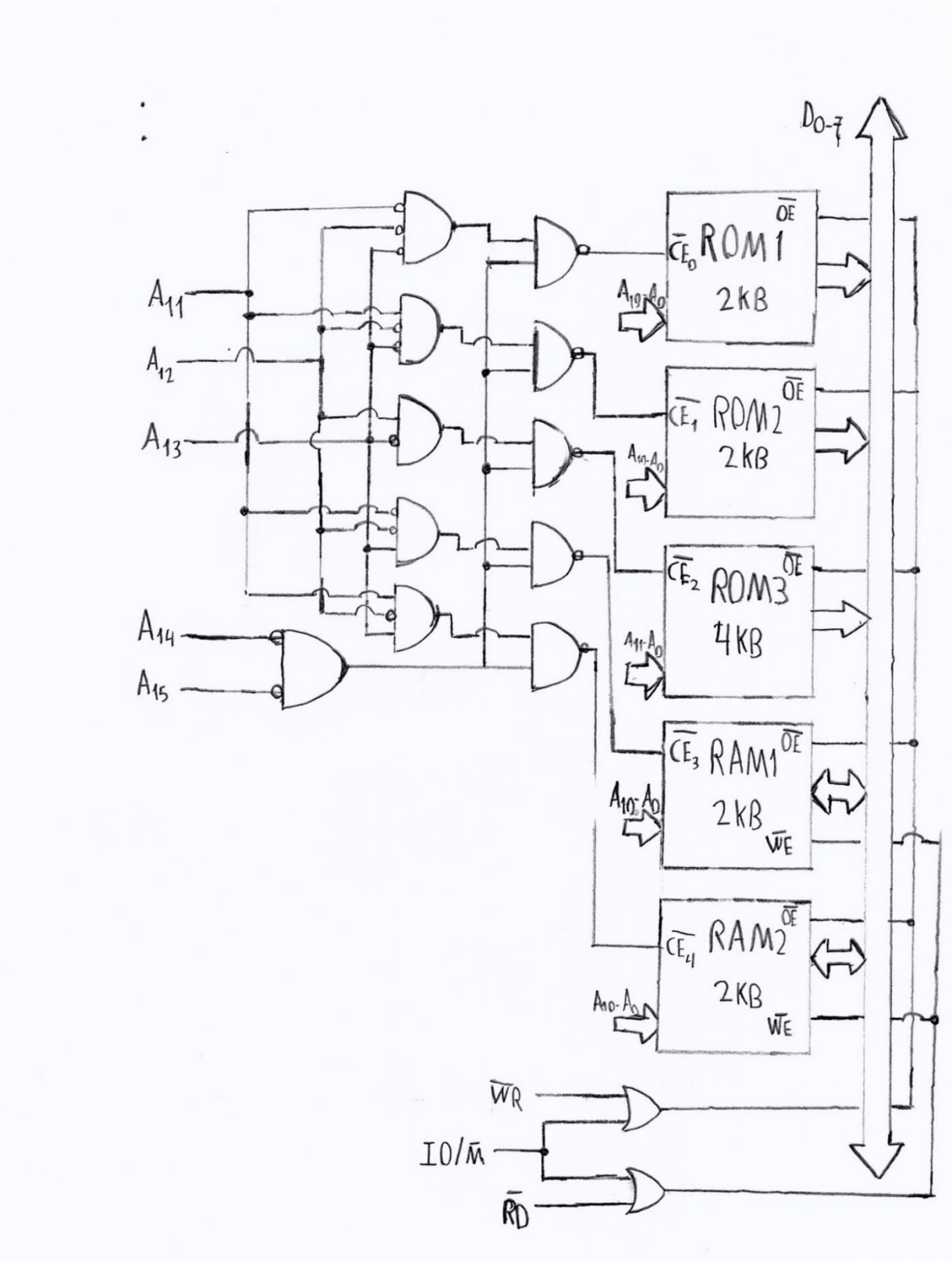
$CE_3 = 1 \Rightarrow (CE_3)' = 0$ όταν $A_{13}A_{12}A_{11} = 100 \Rightarrow Y_4 = 1 \Rightarrow (Y_4)' = 0 \Rightarrow (CE_3)' = (Y_4)'$

$CE_4 = 1 \Rightarrow (CE_4)' = 0$ όταν $A_{13}A_{12}A_{11} = 101 \Rightarrow Y_5 = 1 \Rightarrow (Y_5)' = 0 \Rightarrow (CE_4)' = (Y_5)'$

Άρα έχουμε :



β) Αν κάνουμε χρήση μόνο λογικών πυλών τότε θα έχουμε:



7η Άσκηση:

Για τη σχεδίαση του συστήματος μνήμης υπάρχουν τα εξής ολοκληρωμένα διαθέσιμα:

1xROM 16KBytes

3xRAM 4KBytes

Για τον ζητούμενο σχεδιασμό μνήμης θα έχουμε πρώτα τα 12Kbytes της ROM μετά θα έχουμε διαδοχικά τις τρείς RAM οι οποίες συντελούν τα συνολικά ζητούμε 12KBytes RAM και τέλος θα χρησιμοποιήσουμε τα υπολειπόμενα 4KBytes της ROM. Όλα τα παραπάνω συντελούν τον εξής χάρτη μνήμης:

Μνήμη	Διεύθυνση	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
ROM 12K	0000 H	***0 0 0 0***	0 0 0 0	0 0 0 0	0 0 0 0
	2FFF H	***0 0 1 0***	1 1 1 1	1 1 1 1	1 1 1 1
RAM1 4K	3000 H	***0 0 1 1***	0 0 0 0	0 0 0 0	0 0 0 0
	3FFF H	***0 0 1 1***	1 1 1 1	1 1 1 1	1 1 1 1
RAM2 4K	4000 H	***0 1 0 0***	0 0 0 0	0 0 0 0	0 0 0 0
	4FFF H	***0 1 0 0***	1 1 1 1	1 1 1 1	1 1 1 1
RAM3 4K	5000 H	***0 1 0 1***	0 0 0 0	0 0 0 0	0 0 0 0
	5FFF H	***0 1 0 1***	1 1 1 1	1 1 1 1	1 1 1 1
ROM 4K	6000 H	***0 1 1 0***	0 0 0 0	0 0 0 0	0 0 0 0
	6FFF H	***0 1 1 0***	1 1 1 1	1 1 1 1	1 1 1 1

Για την διευθυνσιοδότηση των λέξεων που αποθηκεύονται στις RAM των 4KBytes χρειαζόμαστε 12 Bits ενώ για την αντίστοιχη διευθυνσιοδότηση της ROM θα χρειαστούμε 14 Bits. Το A_{15} παραμένει σταθερό και συνεπώς δεν χρησιμοποιείται για τον προσδιορισμό καμίας θέσης μνήμης συνεπώς θα το χρησιμοποιήσουμε ως επίτρεψη στον αποκωδικοποιητή. Για την επιλογή ολοκληρωμένου θα χρησιμοποιήσουμε τα bit A_{12} A_{13} A_{14} με την παρακάτω λογική(σύμφωνη με τον χάρτη μνήμης) :

$A_{14}A_{13}A_{12}$= 000, $A_{14}A_{13}A_{12}$= 001, $A_{14}A_{13}A_{12}$= 010 και $A_{14}A_{13}A_{12}$= 110 => ROM

$A_{14}A_{13}A_{12}$= 011 => RAM1

$A_{14}A_{13}A_{12}$= 100 => RAM2

$A_{14}A_{13}A_{12}$= 101 => RAM3

Αν κάνουμε χρήση αποκωδικοποιητή 3:8 (74LS138) και λογικών πυλών θα χρησιμοποιήσουμε τον αποκωδικοποιητή με εισόδους A_{14} A_{13} A_{12} και το A_{15} ως επίτρεψη άρα οι έξοδοι του αποκωδικοποιητή με τα αντίστοιχα CE(Chip Enable) με τον παρακάτω τρόπο:

ROM: $CE_0 = 1 \Rightarrow (CE_0)' = 0$ όταν $A_{14}A_{13}A_{12}= 000$, $A_{14}A_{13}A_{12}= 001$, $A_{14}A_{13}A_{12}= 010$ και $A_{14}A_{13}A_{12}= 110 \Rightarrow Y_0 + Y_1 + Y_2 + Y_6 = 1 \Rightarrow Y_0'Y_1'Y_2'Y_6' = 0 \Rightarrow (CE_0)' = Y_0'Y_1'Y_2'Y_6'$

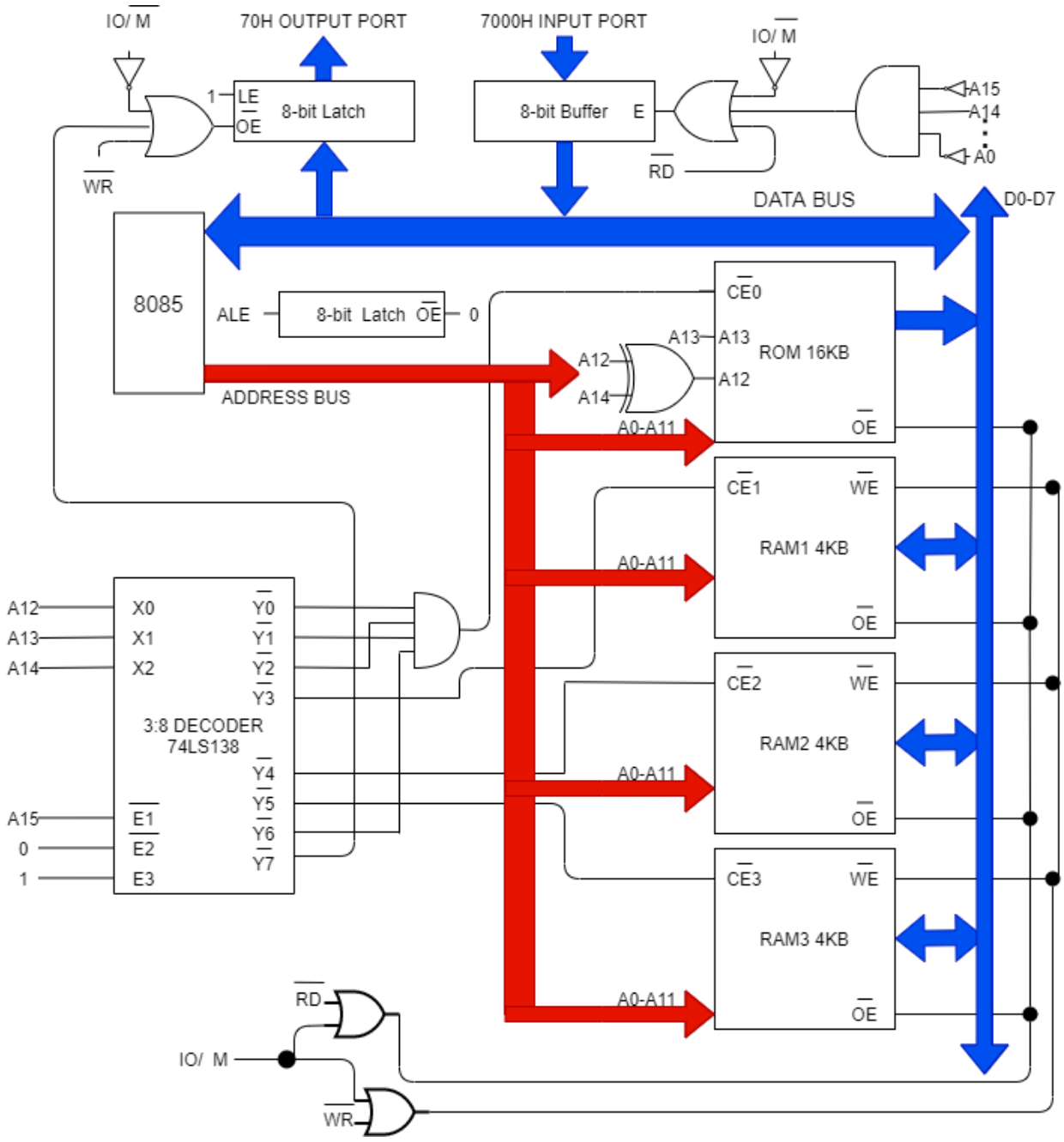
RAM1: $CE_1 = 1 \Rightarrow (CE_1)' = 0$ όταν $A_{14}A_{13}A_{12}= 011 \Rightarrow Y_3 = 1 \Rightarrow (Y_3)' = 0 \Rightarrow (CE_1)' = (Y_3)'$

RAM2: $CE_2 = 1 \Rightarrow (CE_2)' = 0$ όταν $A_{14}A_{13}A_{12}= 100 \Rightarrow Y_4 = 1 \Rightarrow (Y_4)' = 0 \Rightarrow (CE_2)' = (Y_4)'$

RAM3: $CE_3 = 1 \Rightarrow (CE_3)' = 0$ όταν $A_{14}A_{13}A_{12}= 101 \Rightarrow Y_5 = 1 \Rightarrow (Y_5)' = 0 \Rightarrow (CE_3)' = (Y_5)'$

Για την επίτρεψη του Latch της θύρας εισόδου 7000H κάνουμε χρήση πύλης AND με 16 εισόδους, η οποία δίνει στην έξοδο 1 αν: $A_{15}A_{14}A_{13}A_{12}A_{11}A_{10}A_9A_8A_7A_6A_5A_4A_3A_2A_1A_0$ = 0111000000000000 Για την επίτρεψη του Latch της θύρας εισόδου 70H κάνουμε χρήση του $\bar{Y}_7$

Έτσι έχουμε:


IO/ M
70H OUTPUT PORT
7000H INPUT PORT
IO/ M
1
LE
OE
8-bit Latch
8-bit Buffer
E
A15
A14
A0
WR
RD
DATA BUS
D0-D7
8085
ALE
8-bit Latch
OE
0
CE0
A13
A13
ROM 16KB
A12
A12
A14
ADDRESS BUS
A0-A11
OE
CE1
WE
RAM1 4KB
A0-A11
OE
A12
A13
A14
X0
X1
X2
Y0
Y1
Y2
Y3
3:8 DECODER
74LS138
Y4
Y5
Y6
Y7
A15
0
1
E1
E2
E3
CE2
WE
RAM2 4KB
A0-A11
OE
CE3
WE
RAM3 4KB
A0-A11
OE
RD
IO/ M
WR