ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Συστήματα Μικροϋπολογιστών
Αναφορά 2ης Σειράς Ασκήσεων

Μάιος 2021

Λαγός Αναστάσιος - el13531
Κωνσταντίνος Βασιλάκης - el16504

1 Ασκήσεις Προσομοίωσης

1.1 Άσκηση 1

```
; Store numbers in descending order
; Initialize variables
    IN 10H
    LXI H,08FFH
    MVI A,00H
    MVI B,00H
STORE_NUMBERS:
    INX H       ; Memory location in HL registers
    DCR B   ; Counter
    MOV M,B
    CMP B       ; If A < B continue looping
    JC STORE_NUMBERS


; Count the zeros in all numbers
; A = Current Number
; H = Numbers Counter (00H - FFH)
; L = Loop counter to check each digit of a number (8 loops)
; D-E = Memory Address of Numbers
; B-C = Zeros Counter
    MVI H,00H           ; Initialize Numbers Counter
    LXI D,08FFH         ; Initialize address
    LXI B,0000H         ; Initialie total number of zeros

COUNT_ALL_ZEROS:
    INR H
    ; When we count all numbers and return to 00H again exit
    JZ COUNT_ZEROS_END
    INX D
    LDAX D
    MVI L,08H
COUNT_ZEROS_IN_NUMBER:
    RLC
    JNC FOUND_ZERO
    INX B
FOUND_ZERO:
    ; Decrease D and if it is 0 exit inner loop since the
    ; whole number has been processed else continue with the
    ; same number
    DCR L
    JZ COUNT_ALL_ZEROS
    JMP COUNT_ZEROS_IN_NUMBER
COUNT_ZEROS_END:
```

```
   ; Store the result in memory at 0A22H,0A23H
   MOV H,B
   MOV L,C
   SHLD 0A22H


; Count numbers between 10H and 60H
   MVI A,60H
   SUI 10H
   INR A
; Store the result in memory at 0A2FH
   STA 0A2FH
   MOV D,A


END
```

1.2 Άσκηση 2

```
LXI B,0064H       ;100 * 1ms = 0.1sec
MVI D,00H         ;time counter(200 * 0.1 = 20sec (200 = C8H))


START:
LDA 2000H
ANI 80H ;need MSB specifically to be set. RLC of 20H will jump after few loops
CPI 00H ;OFF
JZ OFF1
JMP START

OFF1:
LDA 2000H
ANI 80H
CPI 80H ;checks if on
JZ ON1
JMP OFF1

ON1:
LDA 2000H
ANI 80H
CPI 00H ;check if off, completing push-button
JZ OFF2
JMP ON1

OFF2:
LDA 2000H
```

```
ANI 80H
CPI 80H
JZ ON2
MVI A,00H
STA 3000H           ;light led
CALL DELB           ;wait for 0.1sec
INR D               ;time counter++
MOV A,D
CPI C8H
JNZ OFF2            ;checks if time reached
MVI A,FFH           ;if true then turn led off
STA 3000H
MVI D,00H           ;reset counter
JMP OFF1            ;back to start

ON2:
LDA 2000H
ANI 80H
CPI 00H
JZ RESTART
MVI A,00H           ;light led
STA 3000H
CALL DELB           ;wait for 0.1sec
INR D               ;time counter++
MOV A,D
CPI C8H
JNZ ON2 ;checks if time reached
MVI A,FFH           ;if true then turn led off
STA 3000H
MVI D,00H           ;reset counter
JMP OFF1            ;back to start

RESTART:
MVI D,00H
JMP OFF2

END
```

1.3 Άσκηση 3a

```
;Loop the input to find the first 1 from the right
; B = Loop counter
; C = Position of first 1 from the right
START:
    LDA 2000H
    MVI B,09H
```

```
    MVI C,00H

; The input from the switches is rotated to the right .When the
; carry flag becomes 1 the position of the one is detected. If the
; all inputs are 0 then the position of the 1 will be considered in
; position 9
LOOP_INPUT:
    RRC
    INR C
    JNC FOUND_ZERO
    JMP CALCULATE_OUTPUT
FOUND_ZERO:
    DCR B
    JNZ LOOP_INPUT

; From the position of the first 1 we calculated above (9th position
; if all inputs are 0) we set A = 0 and CY = 1 and we rotate left through
; the accumulator the needed number of times to make the correct led equal to 1.
; If all are zero we rotate the 1 through all the positions back to the CY so
; all the leds stay switched off.
CALCULATE_OUTPUT:
    MVI A,00H
    INR C
    STC
SHIFT_ONE:
    DCR C
    JZ UPDATE_LEDS
    RAL
    JMP SHIFT_ONE

UPDATE_LEDS:
    CMA
    STA 3000H
    JMP START
END
```

1.4 Άσκηση 3b

```
START:
; Read input from the keyboard. If the number
; is not between 1 and 8 continue
    CALL KIND
    CPI 09H
    JNC START
    CPI 00H
    JZ START
```

```
; We rotate left the Accumulator while adding ones to
; the switched off leds
; A = The output of the leds
; B = Loop counter (1-8)
    MOV B,A
    MVI A,00H
; If B > 1 switch off the rightmost open led
LOOP_INPUT:
    DCR B
    JZ UPDATE_LEDS
    RLC
    ORI 01H
    JMP LOOP_INPUT


UPDATE_LEDS:
    STA 3000H
    JMP START

END
```

1.5 Άσκηση 3c

```
IN 10H

START:
LINE_0:
    MVI A,FEH               ; Line number
    STA 2800H               ; Set line number
    LDA 1800H               ; Read key presses for this line
    ANI 07H                 ; Keep the 3 LSB
    CPI 07H                 ; If we have 111 go to the next line
    JZ LINE_1
    CPI 06H                 ; If the left button is pressed
    MVI C,86H               ; Set its code and display the results
    JZ DISPLAY_RESULTS
    CPI 05H                 ; If the middle button is pressed
    MVI C,85H               ; Set its code and display the results
    JZ DISPLAY_RESULTS
    ;CPI 03H                ; If the right button is pressed
    ;MVI C,00H              ; Set its code and display the results
    ;JZ DISPLAY_RESULTS     ; (This line has only two buttons)

; The code for the other lines is similar to the above
```

```
; The only differences are the line numbers and key codes
LINE_1:
    MVI A,FDH
    STA 2800H
    LDA 1800H
    MVI B,07H
    ANA B
    CPI 07H
    JZ LINE_2
    CPI 06H
    MVI C,84H
    JZ DISPLAY_RESULTS
    CPI 05H
    MVI C,80H
    JZ DISPLAY_RESULTS
    MVI C,82H
    JZ DISPLAY_RESULTS

LINE_2:
    MVI A,FBH
    STA 2800H
    LDA 1800H
    MVI B,07H
    ANA B
    CPI 07H
    JZ LINE_3
    CPI 06H
    MVI C,00H
    JZ DISPLAY_RESULTS
    CPI 05H
    MVI C,83H
    JZ DISPLAY_RESULTS
    MVI C,81H
    JZ DISPLAY_RESULTS

LINE_3:
    MVI A,F7H
    STA 2800H
    LDA 1800H
    MVI B,07H
    ANA B
    CPI 07H
    JZ LINE_4
    CPI 06H
    MVI C,01H
    JZ DISPLAY_RESULTS
    CPI 05H
    MVI C,02H
```

```
    JZ DISPLAY_RESULTS
    MVI C,03H
    JZ DISPLAY_RESULTS

LINE_4:
    MVI A,EFH
    STA 2800H
    LDA 1800H
    MVI B,07H
    ANA B
    CPI 07H
    JZ LINE_5
    CPI 06H
    MVI C,04H
    JZ DISPLAY_RESULTS
    CPI 05H
    MVI C,05H
    JZ DISPLAY_RESULTS
    MVI C,06H
    JZ DISPLAY_RESULTS

LINE_5:
    MVI A,DFH
    STA 2800H
    LDA 1800H
    MVI B,07H
    ANA B
    CPI 07H
    JZ LINE_6
    CPI 06H
    MVI C,07H
    JZ DISPLAY_RESULTS
    CPI 05H
    MVI C,08H
    JZ DISPLAY_RESULTS
    MVI C,09H
    JZ DISPLAY_RESULTS

LINE_6:
    MVI A,BFH
    STA 2800H
    LDA 1800H
    MVI B,07H
    ANA B
    CPI 07H
    JZ LINE_7
    CPI 06H
    MVI C,0AH
```

```
    JZ DISPLAY_RESULTS
    CPI 05H
    MVI C,0BH
    JZ DISPLAY_RESULTS
    MVI C,0CH
    JZ DISPLAY_RESULTS

LINE_7:
    MVI A,7FH
    STA 2800H
    LDA 1800H
    MVI B,07H
    ANA B
    CPI 07H
    JZ DISPLAY_RESULTS
    CPI 06H
    MVI C,0DH
    JZ DISPLAY_RESULTS
    CPI 05H
    MVI C,0EH
    JZ DISPLAY_RESULTS
    MVI C,0FH
    JZ DISPLAY_RESULTS


; Display the results
; C = the key code found in the previous steps
; D = The memory location of the screen data
; to be used by STDM
DISPLAY_RESULTS:
    LXI D,0B10H         ; Arbitrary memory address we write
                        ; the screen data

    ; Write the 4 LSB of the key code to the memory
    ; location 0B14H (5th digit on the screen)
    MOV A,C
    ANI 0FH
    STA 0B14H

    ; Write the 4 MSB of the key code to the memory
    ; location 0B15H (6th digit on the screen)
    MOV A,C
    RRC
    RRC
    RRC
    RRC
    ANI 0FH
    STA 0B15H
```

```
    ; Get and display the results
    CALL STDM
    CALL DCD
    JMP START
END
```

1.6 Άσκηση 4

```
START:
LDA 2000H

AND1:
MOV B,A ;store address for later use
ANI 80H ;A3 set?
RRC             ;rotate to d6 so i can perform and operation
MOV C,A ;store in c
MOV A,B ;retrieve address
ANI 40H ;B3 set?
ANA C
RRC             ;LEDs x4-x7 should be off
RRC
RRC
MOV D,A

AND2:
MOV A,B ;retrieve address
ANI 20H ;A2 set?
RRC             ;rotate to d5 so i can perform and operation
MOV C,A ;store in c
MOV A,B ;retrieve address
ANI 10H ;B2 set?
ANA C
RRC
RRC

AND3:
ORA D           ;or operation of the gate
ORA D           ;update total number
MOV D,A ;store in D

XOR1:
MOV A,B ;retrieve address
ANI 08H ;A1 set?
RRC
MOV C,A
```

```
MOV A,B
 ANI 04H ;B1 set?
XRA C           ;XOR operation
RRC
ORA D           ;update total number
MOV D,A

XOR2:
MOV A,B
 ANI 02H ;A0 set?
RRC
MOV C,A
MOV A,B
 ANI 01H ;B0 set?
XRA C           ;XOR operation

XOR3:
XRA D           ;last xor gate
ORA D           ;update total number
CMA             ;negative logic
STA 3000H
JMP START

END
```

2 Θεωρητικές Ασκήσεις

2.1 Άσκηση 5

Δίνουμε ένα ελαφρώς διαφοροποιημένο διάγραμμα από αυτό που υπάρχει στις διαφάνειες. Η μνήμη αυτή είναι οργανωμένη σε 4 blocks των 32x8 bits. Κάθε block περιέχει μία εκ των τεσσάρων τιμών μιας τετράδας. Η επιλογή γίνεται μέσω των A0 έως A7. Τα A3 εως A7 επιλέγουν μία από τις 32 γραμμές της μνήμης. Για κάθε επιλογή γραμμής τα blocks βγάζουν 8 bit το κάθε ένα σε 4 αποκωδικοποιητές 8 σε 1 οι οποίοι με την σειρά τους μέσω των τιμών A0 έως A2 επιλέγουν ένα bit ο καθένας για να σχηματιστεί η σωστή τετράδα κατά την ανάγνωση. Σε αντίθετη φορά έχουμε αντίστοιχα εγγραφή. Οι λειτουργίες τις εγγραφής και της ανάγνωσης ελέγχονται με τα κατάλληλα σήματα ελέγχου. Αν έχουμε ξεχωριστό σήμα εγγραφή και ανάγνωσης αυτά θα ενεργοποιούν ξεχωριστά τους μπλε buffers (εγγραφή) ή κόκκινους (ανάγνωση) όπως φαίνεται στο figure 1.

2.2 Άσκηση 6

Το λογικό διάγραμμα μνήμης φαίνεται στο fig2. Ενώ οι δυο διαφορετικές υλοποιήσεις, αποκωδικοποιητής με λογικές πύλες και μόνο λογικές πύλες, στα fig3 και fig4 αντίστοιχα.

2.3 Άσκηση 7

Ο χάρτης μνήμης φαίνεται στο table 1. Ο μΥ-Σ 8085 μαζί με τις επιμέρους μνήμες και IO και τις διασυνδέσεις μεταξύ τους φαίνονται παρακάτω στο fig5.

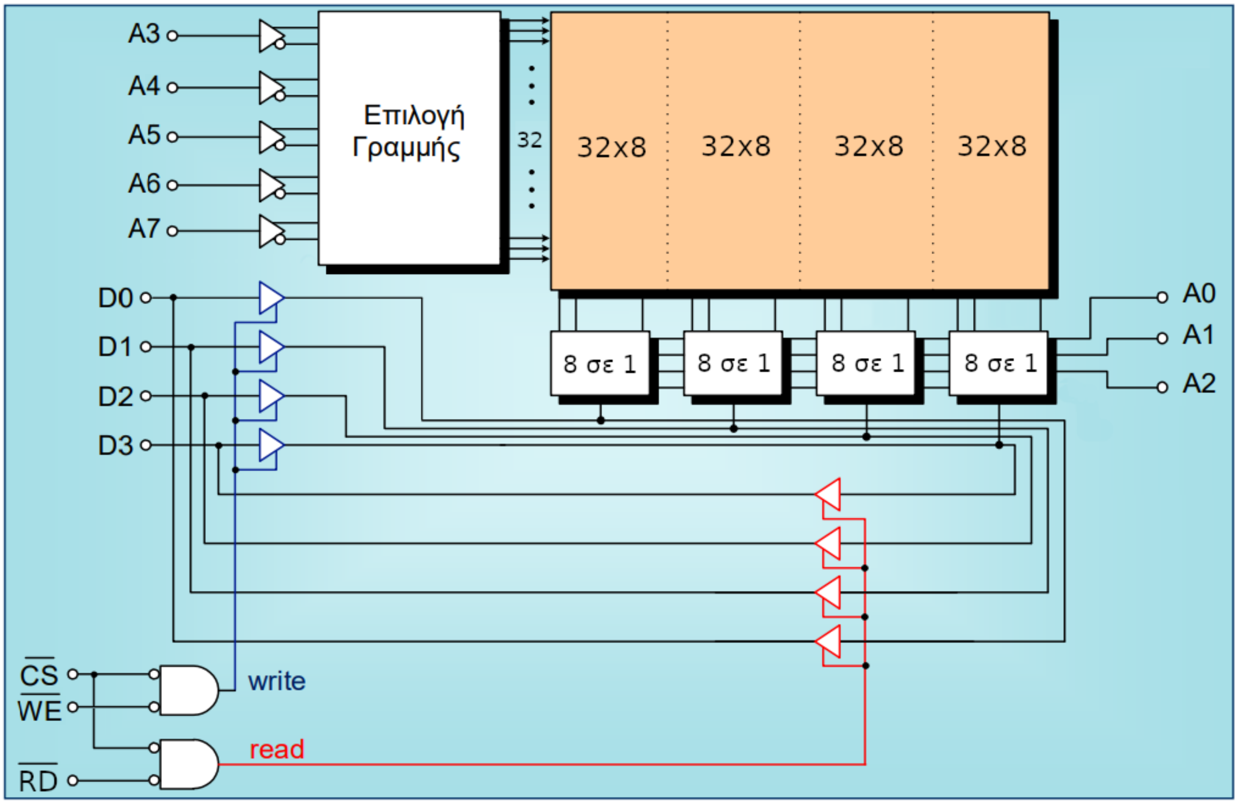

Figure 1: Εσωτερική οργάνωση μιας μνήμης SRAM 256x4 bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Memory
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000H	2K - ROM1
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	2K - ROM2
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	4K - ROM3
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	2K - RAM1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	27FFH	
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	2800H	2K - RAM2
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	2FFFH	

Figure 2: Χάρτης Μνήμης για την άσκηση 6

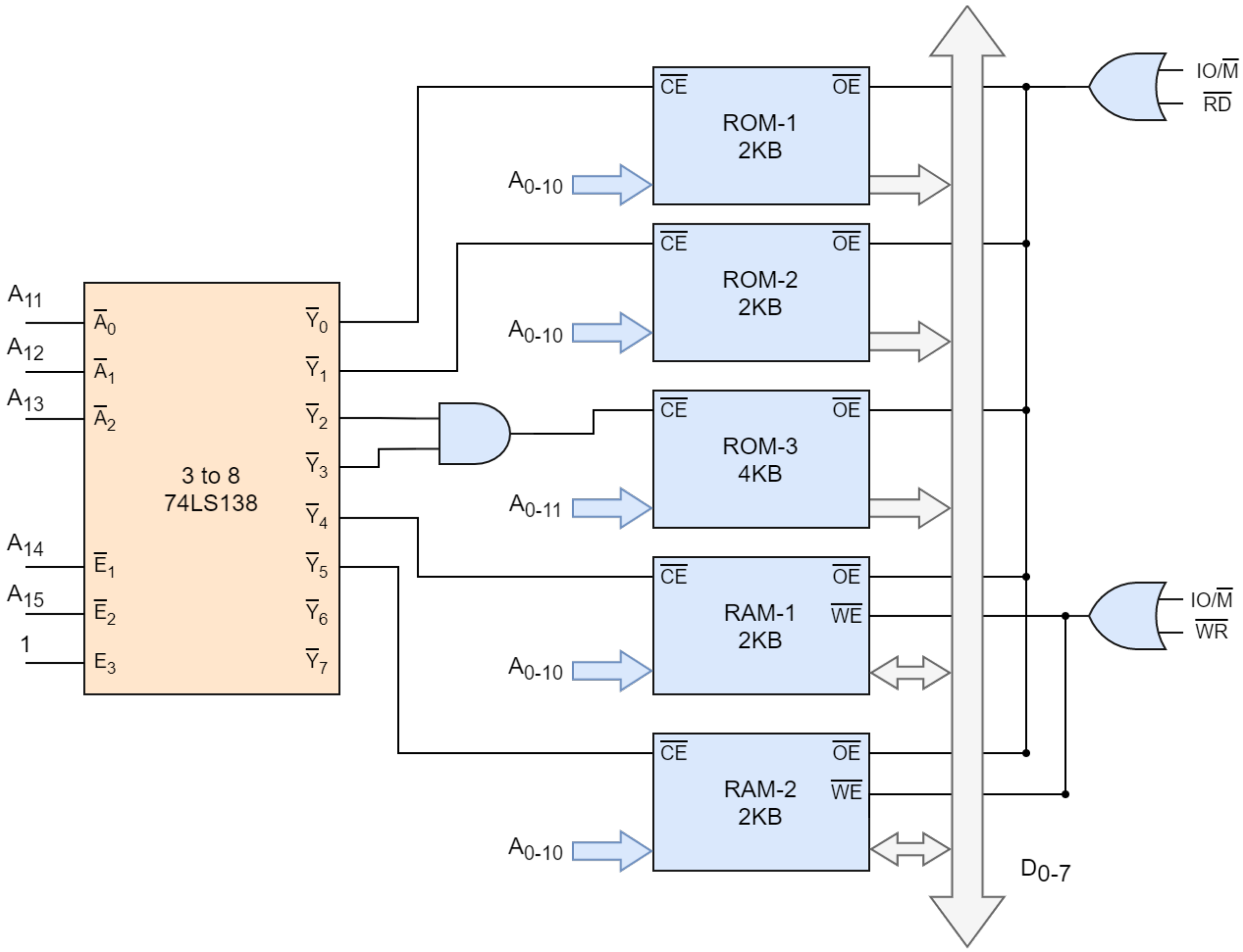

Figure 3: Σύστημα μνήμης με χρήση αποκωδικοποιητή 3:8 και λογικές πύλες.

Memory	Address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROM 12KBytes	0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2FFF	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
RAM1 4KBytes	3000	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	3FFF	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
RAM2 4KBytes	4000	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4FFF	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
RAM3 4KBytes	5000	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	5FFF	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
ROM 4KBytes	6000	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	6FFF	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1

Table 1: Χάρτης μνήμης μΥ-Σ 8085 για την άσκηση 7

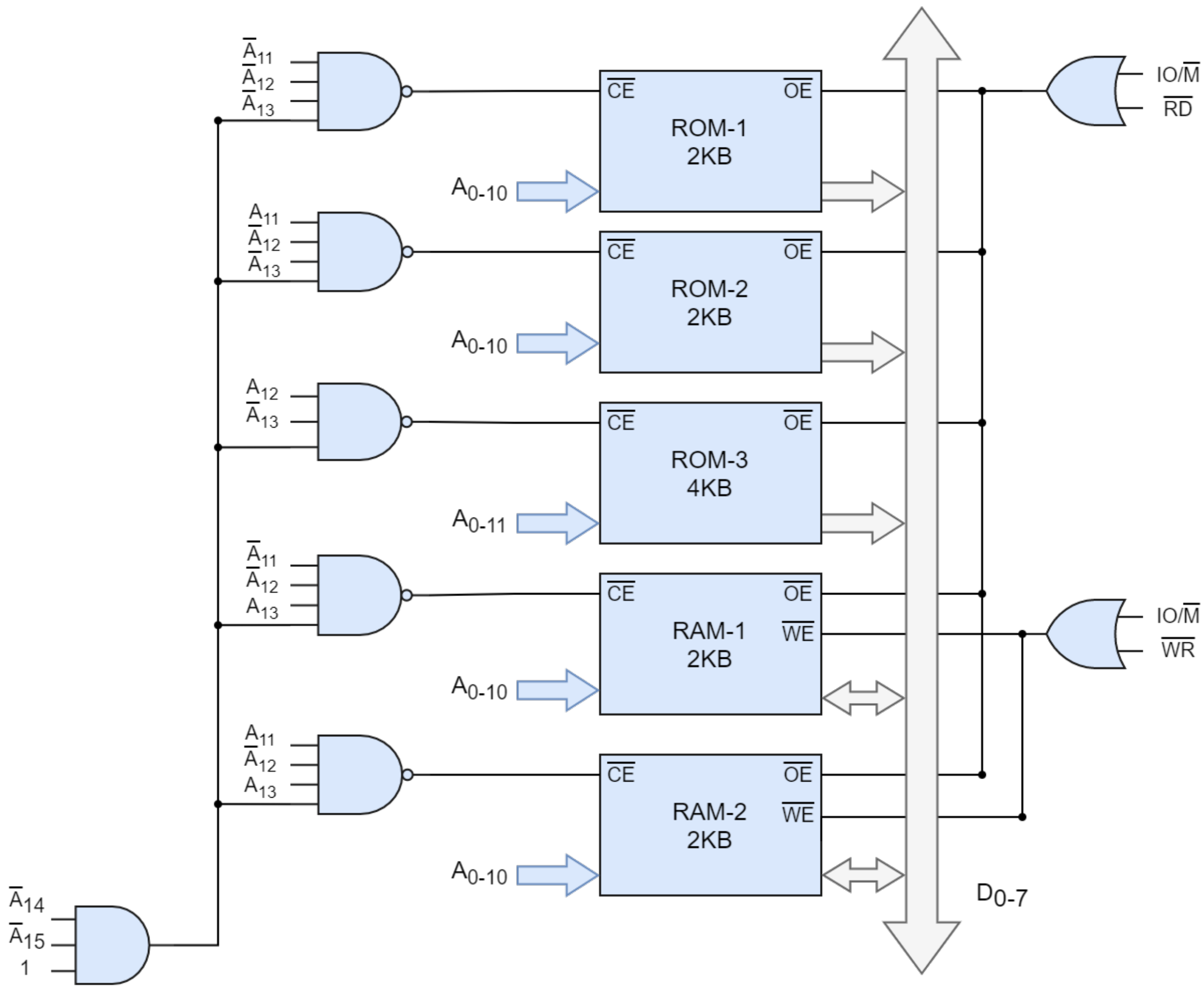

Figure 4: Σύστημα μνήμης με χρήση λογικών πυλών.

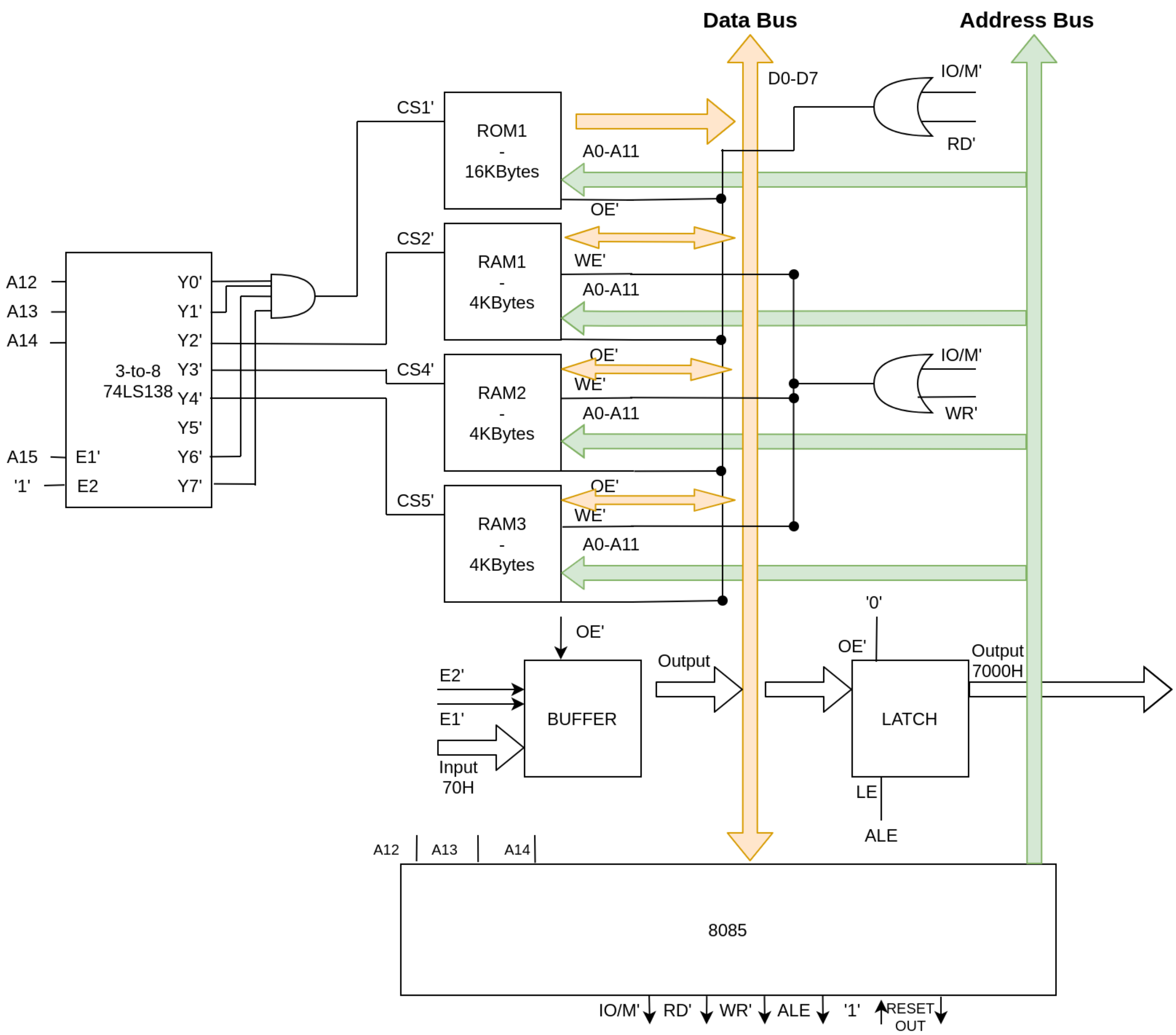

Figure 5: μΥ-Σ 8085