Συστήματα Μικροϋπολογιστών

1η Ομάδα Ασκήσεων

Ιωάννης Τσαντήλας

03120883

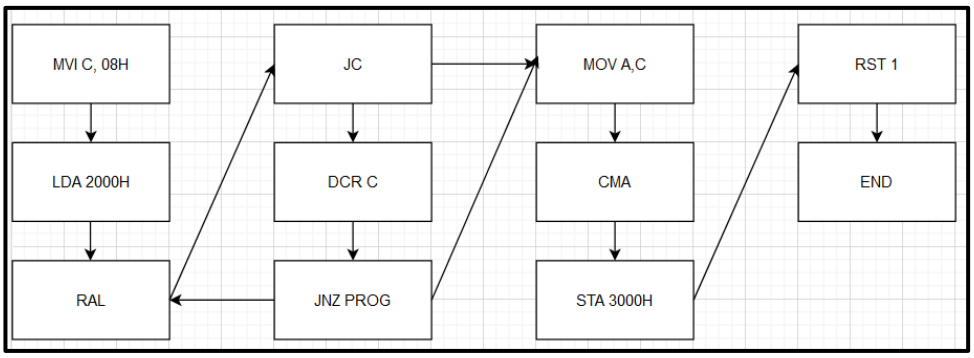
1η Άσκηση

Η μετάφραση σε assembly είναι:

0E 08	**MVI** C,08H	**Move immediate** the value 08H to C
3A 00 20	**LDA** 2000H	**Load** the contents of mem_loc 2000H into A
17	**RAL**	**Rotate left** A's contents
DA 0D 08	**JC** 080DH	**If**(CY==1) **then** (jump to mem_loc 080DH)
0D	**DCR** C	**Decrement** C's contents by 1
C2 05 08	**JNZ** 0805H	**If**(Z==0) **then** (jump to mem_loc 0805H)
79	**MOV A,C**	**Copy** C's contents to A
2F	**CMA**	**Complement** A
32 00 30	**STA** 3000H	**Store** A's contents to mem_loc 3000H
CF	**RST 1**	**Restart 1**

Περνάμε το πρόγραμμα στο Simulator, αλλάζοντας τις διευθύνσεις των jump με ετικέτες και παραθέτουμε το αντίστοιχο διάγραμμα ροής:

```
        MVI C,08H
        LDA 2000H
PROG:
        RAL
        JC NEXT2
        DCR C
        JNZ PROG
NEXT2:
        MOV A,C
        CMA
        STA 3000H
        RST 1
END
```



Για να τρέχει απ' άπειρον, μπορούμε να προσθέσουμε ένα unconditional jump στο τέλος, το οποίο θα μας επαναφέρει στην αρχή του προγράμματος:

```
PROG:
        MVI C,08H
        LDA 2000H
NEXT1:
        RAL
        JC NEXT2
        DCR C
        JNZ NEXT1
NEXT2:
        MOV A,C
        CMA
        STA 3000H
        RST 1
        JMP PROG
END
```

2η Άσκηση

IN 10H	
LXI B,01F4H	**Load immediate** the values B=01H, C=F4H ($01F4_{Hex} = 500_{Dec}$) *
MVI E,01H	**Move immediate** the value E=01H (i.e., nº of LED to be lit)
PROG:	
LDA 2000H	**Load** the value of mem_loc 2000H (i.e., input) to A
MOV D,A	
RAR	**Rotate right** A's contents, so that CY=LSB
JC PROG	**If**(CY==1) **then** (jump PROG)
MOV A,D	
CALL DELB	**Wait** for 0.5secs **
RAL	**Rotate left** A's contents, so that CY=MSB
JNC RIGHT	**If**(CY==0) **then** (we'd turn RIGHT) **else** (we'd turn LEFT)
LEFT:	
MOV A,E	**Copy** E's contents (i.e., nº of LED to be lit) to A
CMA	**Complement** A's contents (reverse logic LEDs)
STA 3000H	**Store** A's contents to mem_loc 3000H (i.e., output)
CMA	**Complement** A's contents (original A's contents)
RLC	**Rotate left** A's contents, to find next LED to be lit
MOV E,A	
JMP PROG	
RIGHT:	
MOV A,E	**Copy** E's contents (i.e., nº of LED to be lit) to A
CMA	**Complement** A's contents (reverse logic LEDs)
STA 3000H	**Store** A's contents to mem_loc 3000H (i.e., output)
CMA	**Complement** A's contents (original A's contents)
RRC	**Rotate right** A's contents, to find next LED to be lit
MOV E,A	
JMP PROG	
END	

***/**Important Note:**

Η εντολή **CALL DELB** καλεί την (implemented στο mLab) υπορουτίνα **DELB**, η οποία προκαλεί καθυστέρηση ίση με την τιμή του ζεύγους BC επί 1 millisecond, και αφού BC = $01F4_{Hex} = 500_{Dec}$, καθυστερεί 0.5 seconds.

3η Άσκηση

LXI B,01F4H	**Load immediate** the values B=01H, C=F4H ($01F4_{Hex} = 500_{Dec}$)
PROG:	
LDA 2000H	**Load** the value of mem_loc 2000H (i.e., input) to A
CPI C8H	($C8_{Hex} = 200_{Dec}$) **If**(A≥200) **then** (C=0) **Else if** (A<200) **then** (C=1)
JNC MSB_ERR	**If**(C==0 → A≥200) **then** (jump MSB_ERR) **Else** (we'll check the next boundary)
CPI 64H	($64_{Hex} = 100_{Dec}$) **If**(A≥100) **then** (C=0) **Else if** (A<100) **then** (C=1)
JNC LSB_ERR	**If**(C==0 → A≥100) **then** (jump LSB_ERR)
MVI D,FFH	($FF_{Hex} = 255_{10} = 1111\ 1111_{Bin}$)
MAIN:	
INR D	**Increase** D's contents by 1
SUI 0AH	**Decrease** A's contents by 10 ($0A_{Hex} = 10_{Bin}$) **If**(Result>A) **then** (C=1)
JNC MAIN	
ADI 0AH	**Increase** A's contents by 10 ($0A_{Hex} = 10_{Bin}$). **If**(Result>A) **then** (C=1)
MOV E,A	**Copy** A's contents to E
MOV A,D	**Copy** D's contents to A
RLC	**Rotate left** A's contents 4 times, so MSD = MSB
RLC	
RLC	
RLC	
ADD E	**Add** E's contents to A, so LSD = LSBs
CMA	**Complement** A's contents (exit is reverse logic)
STA 3000H	**Store** A's contents to mem_loc 3000H (i.e., output)
JMP PROG	
MSB_ERR:	
MVI A,0FH	**Move immediate** the value A=0FH ($0F_{Hex} = 0000\ 1111_{Bin}$) → MSBs ON, LSBs OFF
STA 3000H	**Store** A's contents to mem_loc 3000H (i.e., output)
CALL DELB	**Wait** for 0.5secs
MVI A,FFH	**Move immediate** the value A=0FH ($0F_{Hex} = 0000\ 1111_{Bin}$) → MSBs ON, LSBs OFF
STA 3000H	**Store** A's contents to mem_loc 3000H (i.e., output)
CALL DELB	**Wait** for 0.5secs
JMP PROG	
LSB_ERR:	
MVI A,F0H	**Move immediate** the value A=F0H ($0F_{Hex} = 1111\ 0000_{Bin}$) → MSBs OFF, LSBs ON
STA 3000H	**Store** A's contents to mem_loc 3000H (i.e., output)
CALL DELB	**Wait** for 0.5secs
MVI A,FFH	**Move immediate** the value A=FFH ($FF_{Hex} = 1111\ 1111_{Bin}$) → MSBs OFF, LSBs OFF
STA 3000H	**Store** A's contents to mem_loc 3000H (i.e., output)
CALL DELB	**Wait** for 0.5secs
JMP PROG	
END	

4η Άσκηση

Γενικά:

$$Total\_Cost = Initial\_Cost + (IC\_Cost + Constr\_Cost) * N$$

Όπου Ν είναι το πλήθος των τεμαχίων. Έχουμε λοιπόν ανά περίπτωση:

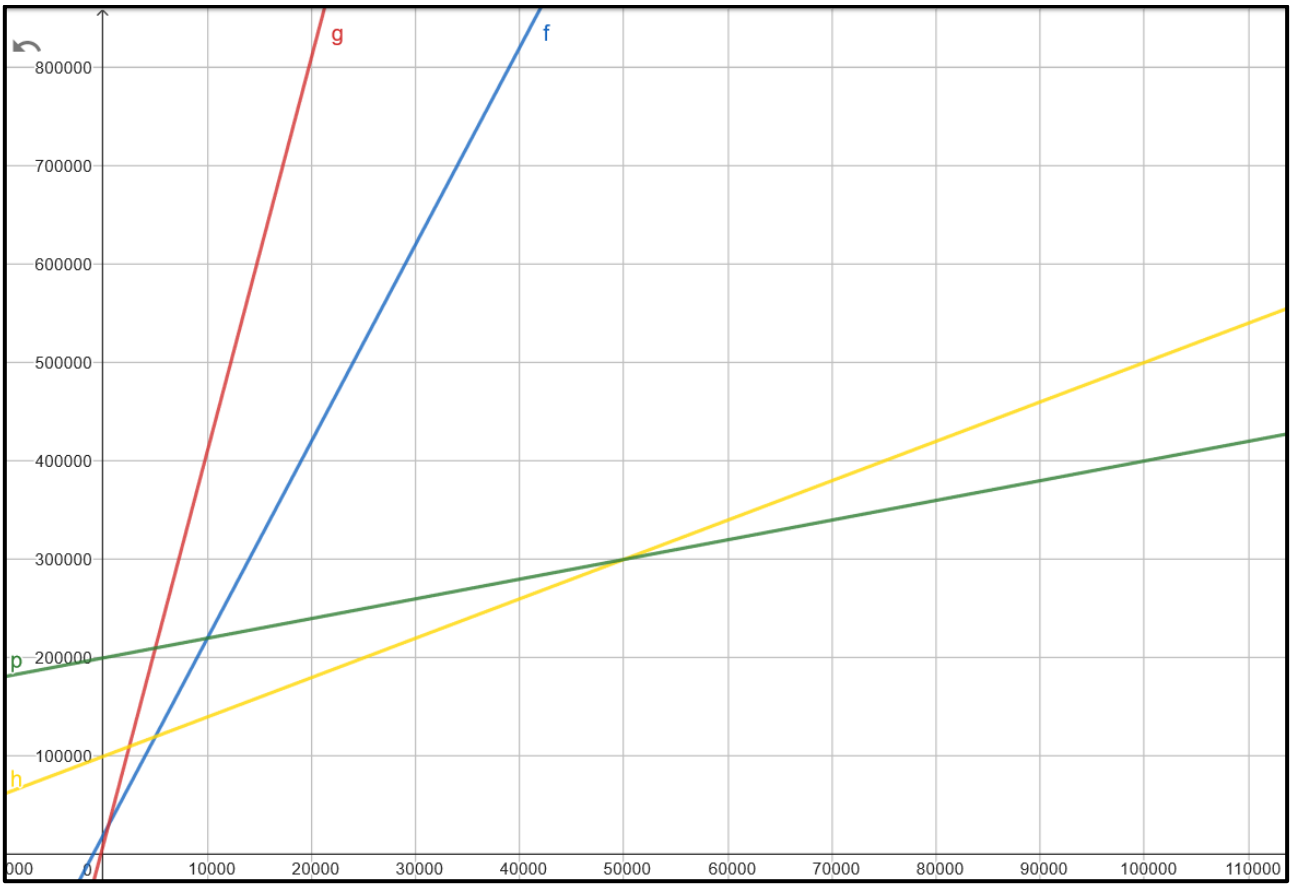
$$TC1 = 20.000 + 20 * N$$

$$TC2 = 10.000 + 40 * N$$

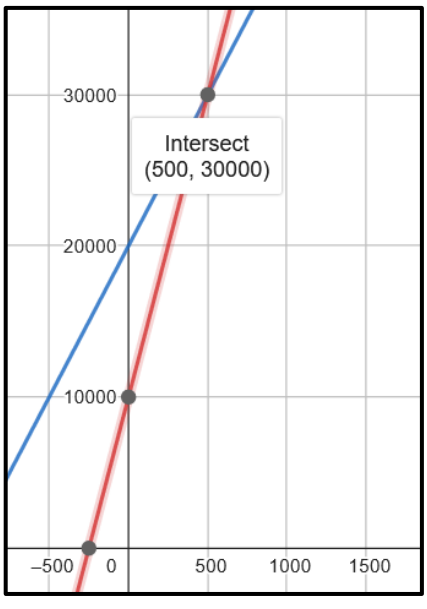
$$TC3 = 100.000 + 4 * N$$

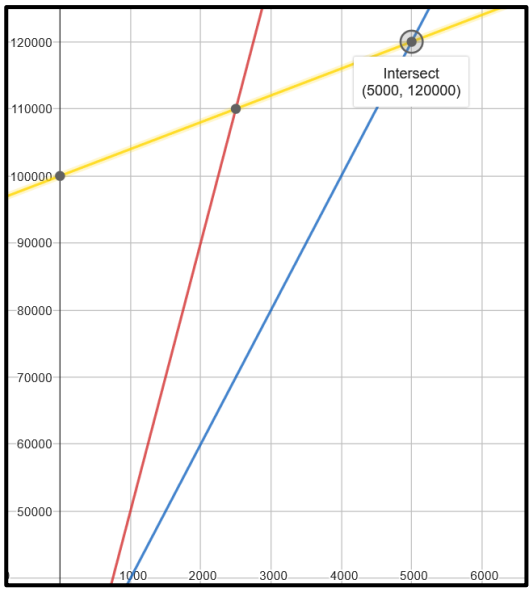
$$TC4 = 200.000 + 2 * N$$

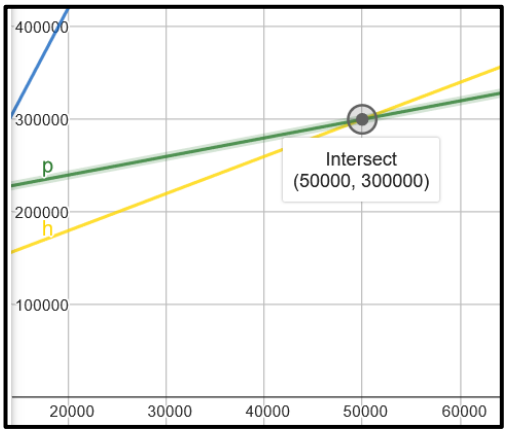
Οι γραφικές αυτών των εξισώσεων:



Τα σημεία τομής που μας ενδιαφέρουν:







Επομένως, για πλήθος:

1. N ≤ 500, συμφέρει η 2η τεχνολογία.
2. 500 ≤ N ≤ 5.000, συμφέρει η 1η τεχνολογία.
3. 5.000 ≤ N ≤ 50.000, συμφέρει η 3η τεχνολογία.
4. 50.000 ≤ N, συμφέρει η 4η τεχνολογία.

Για να «εξαφανίσουμε» την 1η τεχνολογία (μπλε γραμμή), ουσιαστικά θα θέλουμε στο δεύτερο διάστημα (500 ≤ N ≤ 5.000), να προτιμούμε την 2η τεχνολογία (κόκκινη). Για να συμβεί αυτό, θα πρέπει:

$$TC2' = 10.000 + (IC\_Cost' + 10) * N$$

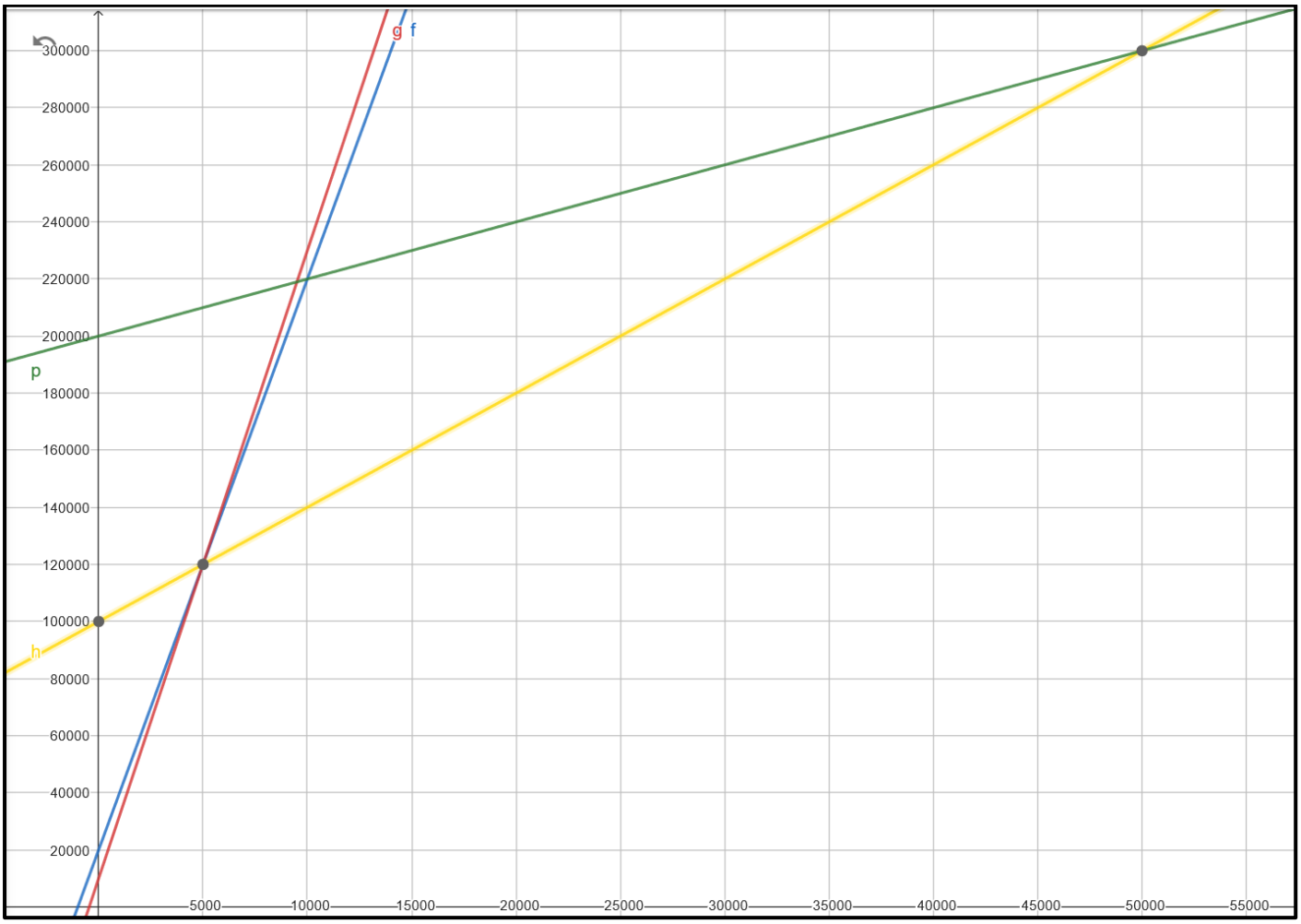
$$TC2'(5.000) \leq 120.000$$

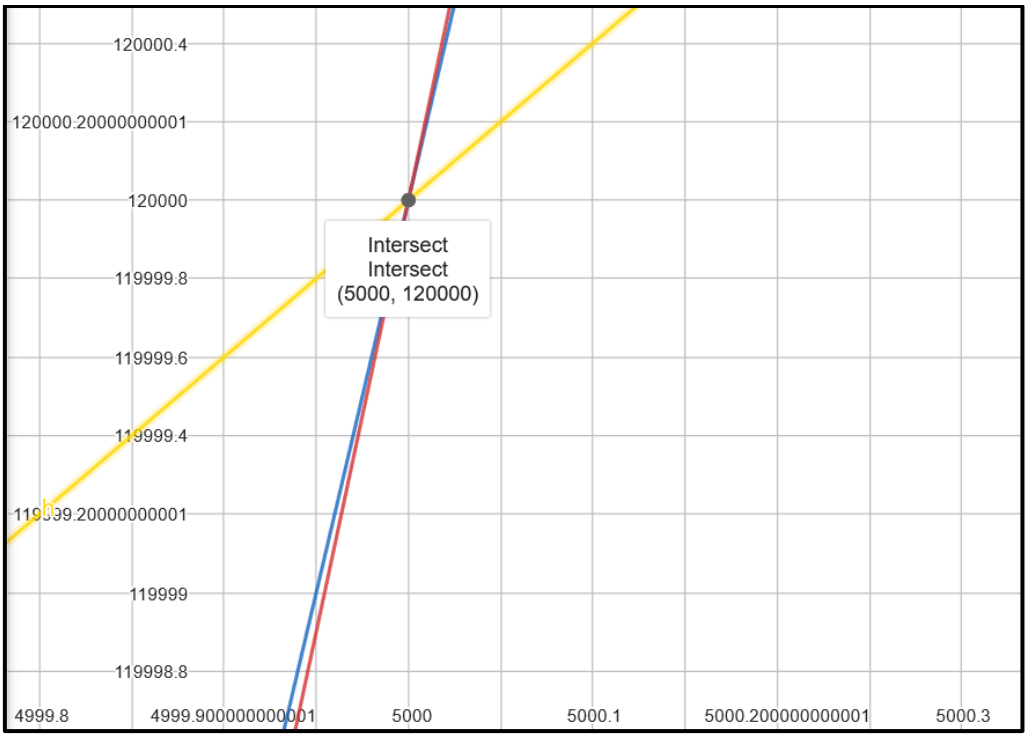
Όπου 120.000 είναι η τεταγμένη του σημείου τομής της 1ης τεχνολογίας (μπλε) με την 3η (κίτρινη). Δηλαδή:

$$10.000 + (IC\_Cost' + 10) * 5.000 \leq 120.000 \rightarrow IC\_Cost' = 12€$$

$$\rightarrow TC2' = 10.000 + 22 * N$$

Άρα, σε καμία περίπτωση δεν προτιμούμε την 1η τεχνολογία (μπλε):




120000.4
120000.20000000001
120000
119999.8
119999.6
119999.4
119999.20000000001
119999
119998.8
4999.8
4999.900000000001
5000
5000.1
5000.200000000001
5000.3
Intersect
Intersect
(5000, 120000)