**Συστήματα Μικροϋπολογιστών**

1η Ομάδα Ασκήσεων

Ιωάννης Τσαντήλας

03120883

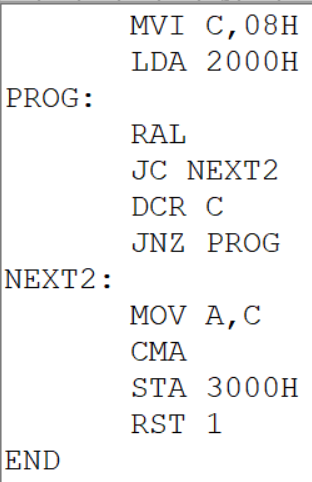
**1η Άσκηση**

Η μετάφραση σε assembly είναι:

|  |  |  |
| --- | --- | --- |
| 0Ε 08 | MVI C,08H | Move immediate the value 08H to C |
| 3Α 00 20 | **LDA** 2000H | **Load** the contents of mem\_loc 2000H into A |
| 17 | **RAL** | **Rotate left** A’s contents |
| DΑ 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:

Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

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

Εικόνα που περιέχει κείμενο, τραπέζι

Περιγραφή που δημιουργήθηκε αυτόματα

**2η Άσκηση**

|  |  |
| --- | --- |
| IN 10H |  |
| LXI B,01F4H | **Load immediate** the values B=01H, C=F4H (01F4Hex = 500Dec) **\*** |
| 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 = 01F4Hex = 500Dec, καθυστερεί 0.5 seconds.

**3η Άσκηση**

|  |  |
| --- | --- |
| LXI B,01F4H | Load immediate the values B=01H, C=F4H (01F4Hex = 500Dec) |
| PROG: |  |
| LDA 2000H | **Load** the value of mem\_loc 2000H (i.e., input) to A |
| CPI C8H | (C8Hex = 200Dec)  **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 | (64Hex = 100Dec)  **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 | (FFHex = 25510 = 1111 1111Bin) |
| MAIN: |  |
| INR D | **Increase** D’s contents by 1 |
| SUI 0AH | **Decrease** A’s contents by 10 (0AHex = 10Bin)  **If**(Result>A) **then** (C=1) |
| JNC MAIN |  |
| ADI 0AH | **Increase** A’s contents by 10 (0AHex = 10Bin).  **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 (0FHex = 0000 1111Bin) → 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 (0FHex = 0000 1111Bin) → 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 (0FHex = 1111 0000Bin) → 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 (FFHex = 1111 1111Bin) → 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:

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

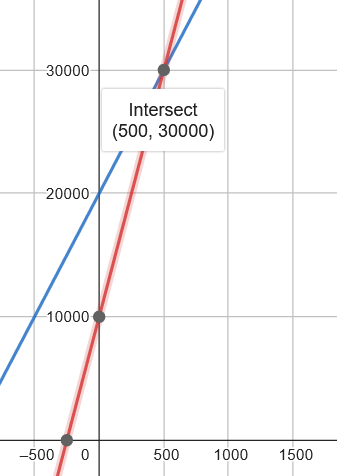
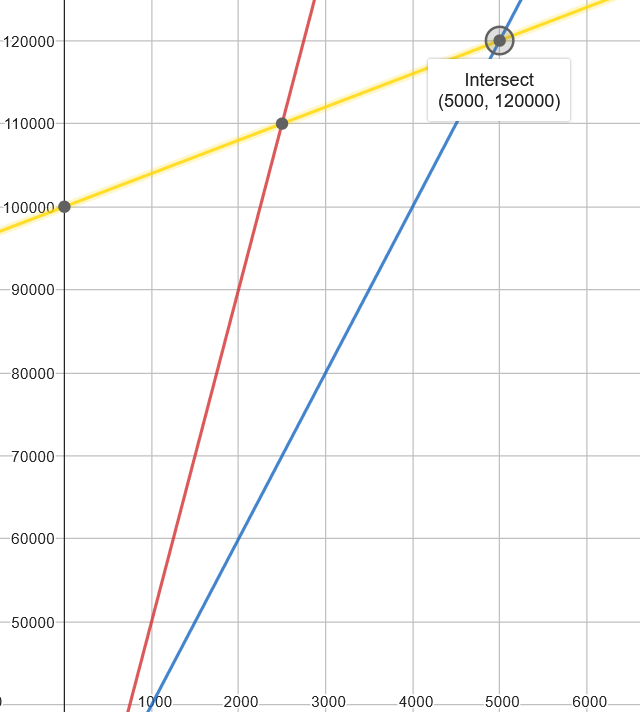
Οι γραφικές αυτών των εξισώσεων:

Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Τα σημεία τομής που μας ενδιαφέρουν:

Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Επομένως, για πλήθος:

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

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

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

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

Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα