1η Σειρά Ασκήσεων

Ειρήνη Στρουμπάκου 03121183
Χαράλαμπος Παπαδόπουλος 03120199

1) Το πρόγραμμα σε γλώσσα assembly είναι το εξής:

```
START:
       MVI B,01H
       LDA 2000H
       CPI 00H
       JZ FIRST

THIRD:
       RAR
       JC SECOND
       INR B
       JNZ THIRD

SECOND:
       MOV A,B

FIRST:
       CMA
       STA 3000H
       RST 1

END
```

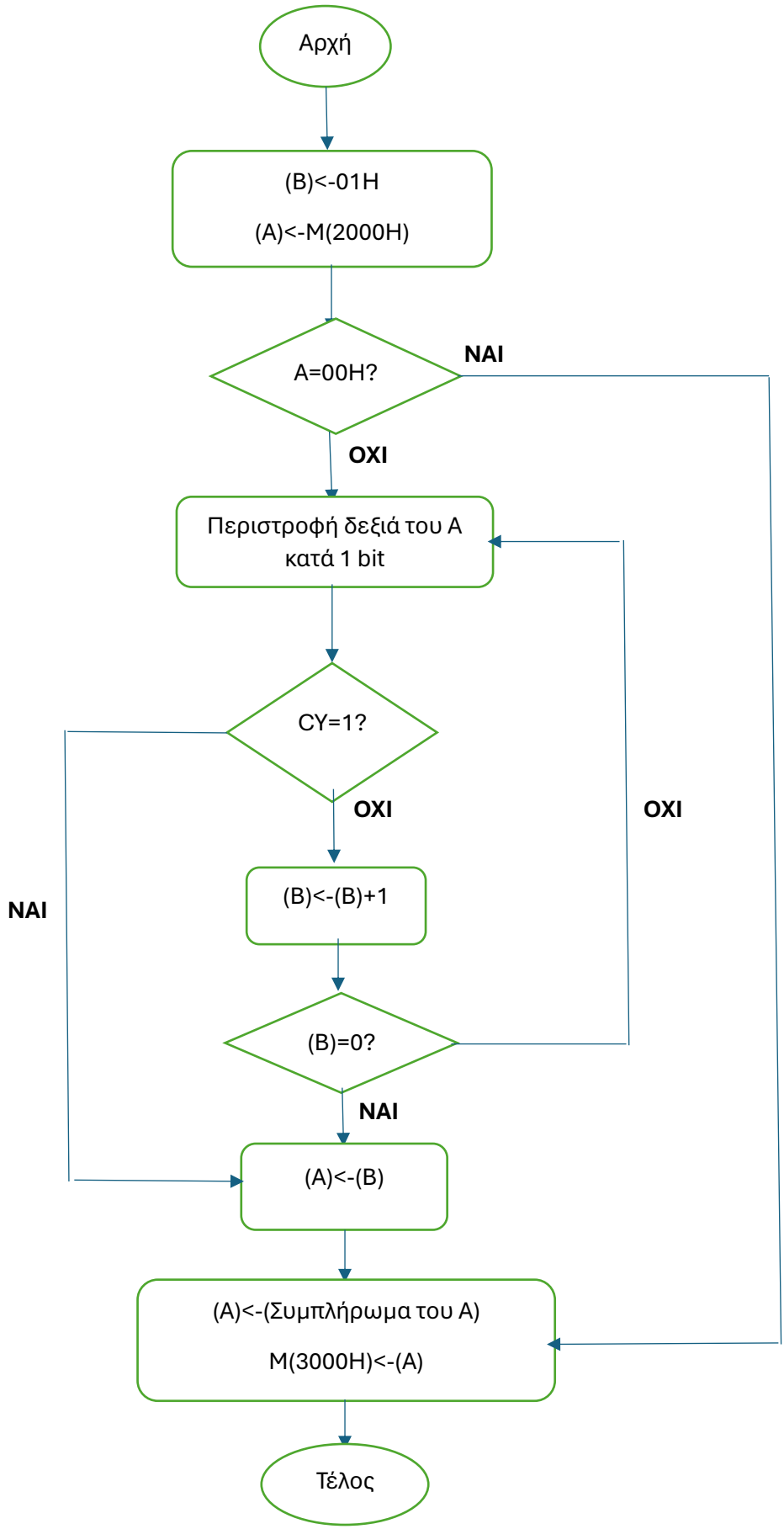
Και εάν κάνουμε την μετάφρασή του παίρνουμε πράγματι τον ακόλουθο κώδικα:

```
0800   06     MVI B,01H
0801   01
0802   3A     LDA 2000H
0803   00
0804   20
0805   FE     CPI 00H
0806   00
0807   CA     JZ FIRST
0808   13
0809   08

THIRD:
080A   1F     RAR
080B   DA     JC SECOND
080C   12
080D   08
080E   04     INR B
080F   C2     JNZ THIRD
0810   0A
0811   08

SECOND:
0812   78     MOV A,B

FIRST:
0813   2F     CMA
0814   32     STA 3000H
0815   00
0816   30
0817   CF     RST 1
```

Παρατηρούμε ότι το πρόγραμμα επιτελεί την ακόλουθη λειτουργία:
Αρχικά μέσω της διεύθυνσης 2000H φορτώνεται στην είσοδο ένας δυαδικός αριθμός. Έπειτα κάνουμε right shift(RAR) τον δυαδικό αριθμό μέχρι το MSB του νέου αριθμού που προκύπτει από τα shifts να γίνει 1. Κατά την διάρκεια αυτών των shifts, αυξάνουμε έναν counter B για να δούμε πόσες μετακινήσεις χρειάστηκαν. Τέλος, φορτώνουμε

στην έξοδο το συμπλήρωμα(CMA) του counter B, οπότε λόγω της αρνητικής λογικής των LEDS, τελικά στην έξοδο παίρνουμε τον αριθμό των shifts που χρειάστηκαν.

Διάγραμμα Ροής:



Για να αποκτήσει συνεχή λειτουργία το πρόγραμμα, αφαιρούμε την εντολή RST 1 και στη θέση της βάζουμε την εντολή JMP START, όπου με START συμβολίζουμε πλέον την αρχή του κώδικα. Προκύπτει, επομένως ο ακόλουθος κώδικας:

```
START:
       MVI B,01H
       LDA 2000H
       CPI 00H
       JZ FIRST

THIRD:
       RAR
       JC SECOND
       INR B
       JNZ THIRD

SECOND:
       MOV A,B

FIRST:
       CMA
       STA 3000H
       JMP START

END
```

2)

```
START:
       IN 10H
       LXI H,0001H              ;Rightmost led
       MOV A,L
       CMA
       STA 3000H
       MVI A,01H                ;Turning on rightmost led

LOOP1:
       LXI B,FFFFH              ;Delay
       MOV E,A                  ;Saving the position of the last lit led
       CALL ENABLE              ;Checking if the second last LSB is on
       LDA 2000H                ;Input
       CPI 01H                  ;Comparing input to 00000001B
       JZ DIP_ON
       JNZ DIP_OFF

ENABLE:
       LDA 2000H                ;Constantly checking input to see whether  enable changed
       ANI 02H                  ;Compare input to enable-bit
       JNZ ENABLE
       RET                      ;Return to main program

DIP_ON:
       MOV A,D                  ;D is our loop counter
                                ;For 8<= D <15 the LEDs move right.
       CPI 0EH
       JNC RESET_COUNTER        ;For D = 15 we reset the counter
       CPI 07H
       JC LEFT                  ;For 0<= D <7 the LEDs move left.
       JNC RIGHT                ;For 8<= D <15 the LEDs move right.
```

```
DIP_OFF:
       MOV A,E
       MVI L,07H
       CALL CHECK_BIT            ;With this subroutine we are aiming to get the
                                 ;position of the LED so in case the dip-switch
                                 ;switched back on we can resume our counter from that point.
       MOV D,L
       JMP LEFT

CHECK_BIT:
       RLC
       JC FOUND_BIT
       DCR L
       JNZ CHECK_BIT

FOUND_BIT:
       RET

RESET_COUNTER:
       MVI D,00H

LEFT:
       MOV A,E
       RLC                       ;Rotate the LED to the left
       MOV E,A
       JMP DELAY

RIGHT:
       MOV A,E                   ;Rotate the LED to the right
       RRC
       MOV E,A
       JMP DELAY

DELAY:
       DCX B
       MOV A,B
       ORA C
       JNZ DELAY

END:
       MOV A,E
       INR D                     ;Increment the Counter
       CMA
       STA 3000H
       CMA
       JMP LOOP1
       END
```

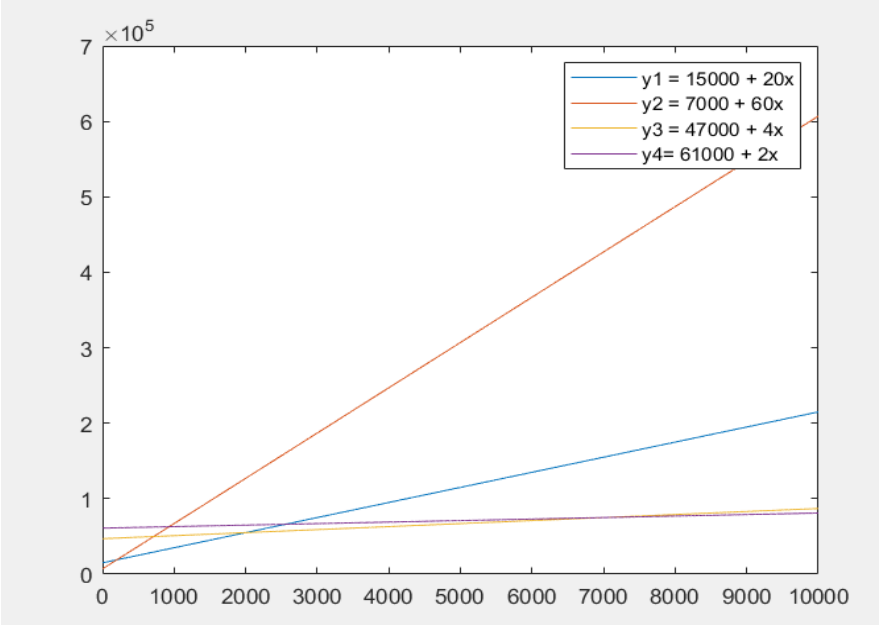
3) Ο κώδικας για την 3η άσκηση μαζί με σχόλια για την λειτουργία του είναι ο εξής:

```
START:
       LDA 2000H
       MVI B,FFH     ;Store -1 to B

HUND:
       CPI C7H       ;Check if greater than 199
       JNC LEDS      ;If yes go to loop LEDS
       CPI 64H       ;Check if greater than 100
       JC DECA       ;If not go to DECA
       SUI 64H       ;If yes subtract 100
```

```
DECA:
       INR B          ;Increase the counter of tens
       SUI 0AH        ;Subtract 10
       JNC DECA       ;If positive again loop
       ADI 0AH        ;If negative add 10
       MOV C,A        ;Store the ones in C

SUM:
       MOV A,B        ;Store the tens in A
       RLC
       RLC
       RLC
       RLC            ;Create 2^4
       ADD C          ;Make of the final number with tens and ones
       CMA
       STA 3000H
       JMP RESTART

LEDS:
       MVI A,00H      ;Turn on all LEDs
       STA 3000H
       JMP RESTART

RESTART:
       RST 1
       JMP START

END
```

4) Για την άσκηση αυτή σχεδιάζουμε τα διαγράμματα για τις ακόλουθες συναρτήσεις:

f1(x) = 15000 + 20x
f2(x) = 7000 + 60x
f3(x) = 47000 + 4x
f4(x) = 61000 + 2x
με x τα τεμάχια



Για να βρω τις περιοχές χαμηλότερου κόστους για κάθε τεχνολογία αρχικά βρίσκω τα σημεία τομής των τεχνολογιών:

a) f1(x)=f2(x) => x=200
b) f1(x)=f3(x) => x=2000
c) f1(x)=f4(x) => x=2555
d) f2(x)=f3(x) => x=714
e) f2(x)=f4(x) => x=931
f) f3(x)=f4(x) => x=7000

Οπότε με βάση τις τιμές αυτές και τα διαγράμματα παρατηρούμε ότι:
Αρχικά η 2 έχει χαμηλότερο κόστος μέχρι να συναντήσει την 1 στο σημείο x=200. Έπειτα, έχει η 1 μέχρι να συναντήσει την 3 στο σημείο x=2000. Έπειτα, έχει η 3 μέχρι να συναντήσει την 4 στο σημείο x=7000. Τέλος, μετά από το σημείο x=7000 χαμηλότερο κόστος έχει η 4.
Τελικά:
$1^{η}$: 200<x<2000
$2^{η}$: 0<x<200
$3^{η}$: 2000<x<7000
$4^{η}$: x>7000

Για να εξαφανιστεί η $1^{η}$ τεχνολογία πρέπει f2'(x)<f1(x) =>
7000+(λ+10)x<15000+20x => $\lambda < \frac{8000}{x} + 10$
Αυτό όμως πρέπει να ισχύει για κάθε x, οπότε και για x->άπειρο. Για
x->άπειρο προκύπτει $\lambda \leq 10$(Βάζω ισότητα, γιατί δεν γίνεται όντως να πάρω άπειρα κομμάτια, οπότε παίρνω όριο στο άπειρο. Επομένως, για οποιαδήποτε τιμή του νέου κόστους μέχρι 10€ η $1^{η}$ τεχνολογία εξαφανίζεται.