Συστήματα Μικροϋπολογιστών
1η Ομάδα Ασκήσεων
2022

Στεφανής Παναγιώτης (el19096)
Μιχάλης Τσιλιμιγκουνάκης (el19001)

1η Άσκηση:

Το πρόγραμμα είναι φορτωμένο στην μνήμη με αρχή τη διεύθηνση 0800H.
Το πρόγραμμα σε γλώσσα μηχανής είναι το εξής:

06 01 **3A** 00 20 **FE** 00 **CA** 13 08 **1F DA** 12 08 **04 C2** 0A 08 **78 2F 32** 00 30 **CF**

Με την βοήθεια του πίνακα 2 των σημειώσεων το μετατρέπουμε σε εντολές assembly. Επίσης γνωρίζοντας την αρχική θέση μνήμης, αλλά και το μέγεθος κάθε εντολής μπορούμε παράλληλα να έχουμε στο πλάι και την θέση μνήμης κάθε εντολής. Έτσι, έχουμε την εξής ακολουθία εντολών:

```
0800: MVI B, 01H        #B<-01H
0802: LDA 2000H         #A ← MEM[2000H]
0805: CPI 00H           # A-00 (Z=1 IF A==0)
0807: JZ 0813H          #  JUMP 0813H IF A==0
080A: RAR               # Right round shift accumulator with carry
080B: JC 0812H          # IF Z=1 JUMP 0813H (IF LSB OF A IS ZERO)
080E: INR B             # B++
080F: JNZ 080AH         # IF !(B==0) BRANCH AT 0812H
0812: MOV A,B           # A<-B
0813: CMA               # A<-A'
0814: STA 3000H         # A->MEM[3000H]
0817: RST 1             # Branch at address 0008H
```

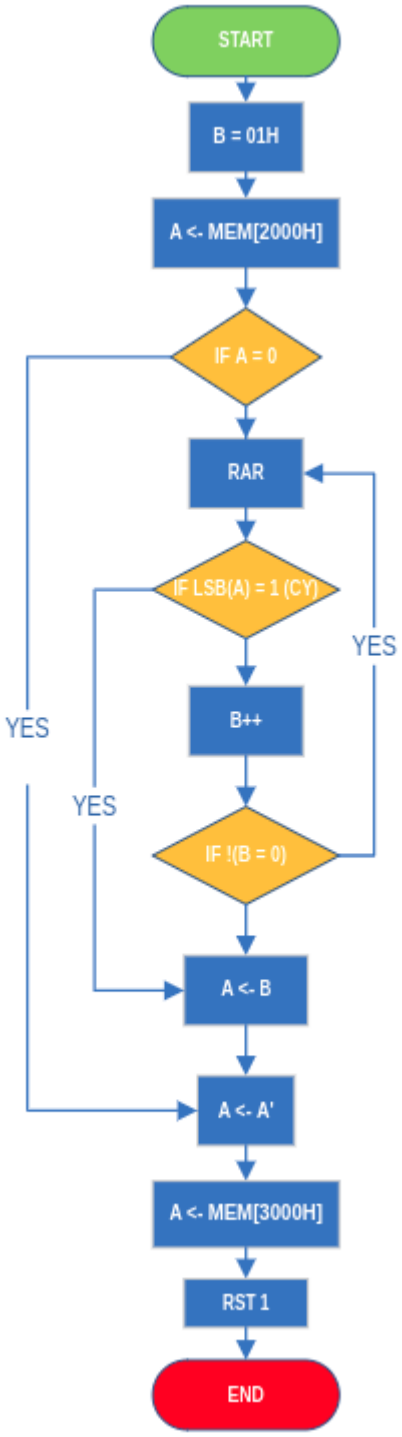
Μπορούμε να δημιουργήσουμε labels για τις διευθύνσεις στις οποίες κάνουμε branch και έτσι το πρόγραμμα μετατρέπεται στο εξής:

```
START:
0800: MVI B, 01H        #B<-01H
0802: LDA 2000H         #A ← MEM[2000H]
0805: CPI 00H           # A-00 (Z=1 IF A==0)
0807: JZ L3             #  JUMP 0813H IF A==0
L1
080A: RAR               # Right round shift accumulator with carry
080B: JC L2             # IF Z=1 JUMP 0813H (IF LSB OF A IS ZERO)
080E: INR B             # B++
080F: JNZ L1            #IF !(B==0) BRANCH AT 0812H
L2:
0812: MOV A,B           # A ← B
L3:
0813: CMA               # A<-A'
0814: STA 3000H         # A->MEM[3000H]
0817: RST 1             # Branch at address 0008H
```

Αν θέλαμε να κάνουμε το πρόγραμμα να τρέχει ατέρμονα, θα χρειαζόταν να αλλάζαμε την εντολή “RST 1” που μας επιστρέφει στην προκαθορισμένη θέση 0008H, για την εντολή “JMP START”. Έτσι κατά την ολοκλήρωση του προγράμματος, η ροή θα μεταφερόταν πάλι στην αρχή του, κάνοντας το να τρέξει από την αρχή κ.ο.κ.

```
START:
0800: MVI B, 01H          #B<-01H
0802: LDA 2000H           #A ← MEM[2000H]
0805: CPI 00H             # A-00 (Z=1 IF A==0)
0807: JZ L3               #  JUMP 0813H IF A==0
L1
080A: RAR                 # Right round shift accumulator with
carry
080B: JC L2               # IF Z=1 JUMP 0813H (IF LSB OF A
IS ZERO)
080E: INR B               # B++
080F: JNZ L1              #IF !(B==0) BRANCH AT 0812H
L2:
0812: MOV A,B             # A ← B
L3:
0813: CMA                 # A<-A’
0814: STA 3000H           # A->MEM[3000H]
0817: JMP START           # Branch at START
```

Τρέχοντας το παραπάνω πρόγραμμα στον προσομοιωτή Tsik, παρατηρούμε πως τυπώνει στην στα LED (σε δυαδική μορφή) την θέση του μικρότερου ενεργού (On) dip switch. Δηλαδή αν έχουμε on το 5ο dip switch (και όλα τα υπόλοιπα δεξιά του είναι off) τότε στα LED θα δούμε τον αριθμό 00000101 (5 σε δυαδική μορφή).



2η Άσκηση:

Παρακάτω ακολουθεί ο κώδικας του προγράμματος της δεύτερης άσκησης. Περιληπτικά ο κώδικας που υλοποιήσαμε ακολουθεί την εξής προσέγγιση:
Αρχικά στο START ελέγχουμε το 2ο LSB ,αν είναι on , επιστρέφουμε συνεχώς στο START κρατώντας το προηγούμενο LED σταθερά αναμμένο. Όταν το 2ο LSB γίνει off προχωράμε στο L1.
Στο L1 αυτό που κάνουμε είναι να βγάζουμε την εκάστοτε έξοδο στα LED και μέχρι κάθε φορά να φτάσουμε στο 8ο LED κάνουμε αριστερή περιστροφή για να προχωρήσουμε στο επόμενο LED. Όταν φτάσουμε στο 8ο LED χρειάζεται να ελέγξουμε το 1ο LSB για να δούμε αν θα πρέπει να συνεχίσουμε κάνοντας κυκλική κίνηση ή να ανάβουμε τα LED προς την αντίθετη κατεύθυνση , γι' αυτό πηδάμε στο L2.
Στο L2 ελέγχουμε ξανά αρχικά το 2ο LSB κι αν είναι «on» λουπάρουμε εκεί , αλλιώς ελέγχουμε το 1ο LSB αν είναι off πηγαίνουμε στο L1 για να συνεχίσουμε την αριστερή ολίσθηση (κυκλική κίνηση). Διαφορετικά βγάζουμε έξοδο στα LED και κάνουμε δεξιά ολίσθηση. Όταν φτάσουμε στο 1ο LED με την ίδια λογική πηγαίνουμε στο START για να αρχίσουμε πάλι την αριστερή ολίσθηση (μέχρι να ξαναφτάσουμε στο 8ο LED κ.ο.κ.) ,αλλιώς αν δεν έχουμε φτάσει στο 1ο LED πηδάμε ξανά στο L2.

```
IN 10H ;memory protection off
LXI B,01F4H ;delay 500ms
MVI D,FEH
START:
       LDA 2000H ; load dip switches
       RRC
       RRC ; cy = 2nd LSB
       JC START ; If CY=1, 2nd LSB is on, loop start to keep led steady
L1:
       MOV A,D ; load led position in accumulator
       STA 3000H ; output led position
       CALL DELB ; add delay
       RLC ; move led to next (left) position
       MOV D,A ; store led position in d
       CPI 7FH ; check if led is in the 8th position
       JZ L2 ; if yes go to l2 to check dipswitch lsb
       JMP START ; jump to start to move the led again
L2:
       LDA 2000H ; load dipswitches again
       RRC
       RRC
       JC L2 ; If CY=1, 2nd LSB is on, loop L2 to keep led steady
       RRC
       RRC
       RRC
       RRC
       RRC
       RRC
       RRC
       JNC L1 ; if CY=0 go to L1 for circular mode
       MOV A,D ; load led position
       STA 3000H ; output led position
       CALL DELB ; add delay
       RRC ; right shift led position
       MOV D,A ; store led position
       CPI FEH ; check if first led is on
       JZ START ; if yes go to L1 to start left shifting
       JMP L2 ; if not, check the lsb and act according to the LSB
       END
```

3η Άσκηση:

Στην άσκηση αυτή καλούμαστε να επεκτείνουμε το παράδειγμα της σελίδας 84 του βιβλίου που μετατρέπει ένα δυαδικό αριθμό σε δεκαδικό για να μπορεί να δέχεται και αριθμούς μεγαλύτερους του 99.
Στον παρακάτω κώδικα αρχικά στο START ελέγχουμε αν ο αριθμός είναι μικρότερος του 100.
Αν ναι, υλοποιούμε παρόμοια διαδικασία με του παραδείγματος απλά αντί να αποθηκεύουμε τις μονάδες και τις δεκάδες σε διευθύνσεις μνήμης, τις αποθηκεύουμε σε δύο καταχωρητές και στη συνέχεια μέσω της λογικής πράξης OR συνενώνουμε τους δύο καταχωρητές με τέτοιο τρόπο ώστε να βρίσκονται οι δεκάδες στα 4 MSB του Α και οι μονάδες στα 4 LSB του Α. Στη συνέχεια με την εντολή “STA 3000H” τυπώνουμε στα LEDs τον «αριθμό» της εισόδου του dip switch.
Αν ο αριθμός είναι μεγαλύτερος του 99 πηγαίνουμε στο GRTR99 και ελέγχουμε αν είναι και μεγαλύτερος του199. Αν ναι, πηγαίνουμε στο GRTR199 και φορτώνοντας τον Α αρχικά με τον αριθμό 0FH (00001111) ανάβουμε τα 4 LSB των LED και ξαναφορτώνοντας τον με τον αριθμό 00H (00000000) σβήνουμε τα 4 LSB των LED. Με αυτό τον τρόπο πετυχαίνουμε τα αναβοσβήνουν τα ζητούμενα LED μέχρι να «έρθει» στην είσοδο νέος αριθμός μικρότερος του 199.
Αν ο αριθμός είναι μεγαλύτερος 99 , αλλά μικρότερος του 200 , αφαιρούμε από τον αριθμό αυτό το 100 και στη συνέχεια πηδάμε στο SMLR100 ώστε να αναπαραστήσουμε στην έξοδο τις δεκάδες και τις μονάδες του αριθμού μειωμένου κατά 100 (x-100) .

```
LXI B,01F4 ; time delay 500ms
START:
       LDA 2000H ; load dip switches
       CPI 64H ; check if number is less than 100
       JNC GRTR99 ; if yes go to greater99
       MVI E,FFH ; initialize E with -1
SMLR100:
       INR E ; add 1 ten
       SUI 0AH ; subtract 1 ten from A
       JNC SMLR100 ; if A>0, keep subtracting
       ADI 0AH ; else, correct the negative remainder
       MOV D,A ;D has units
       MOV A,E ;E has tens (and store them in A)
       RLC ;move tens to the 4 MSBs
       RLC
       RLC
       RLC
       ORA D ; logical addition to show the units 4 LSBs (A+D)
       CMA ; complement due to inverse led logic
       STA 3000H ; output
       JMP START ; jump start to check for a new number
GRTR99:
       MVI E,FFH ; initialize E with -1
       CPI C8H ; check if number >=200
       JNC GRTR199 ; if yes, go to greater199
       SUI 64H ; else, subtract 100
       JMP SMLR100 ; output the value of {input}-100
GRTR199:
       MVI A,0FH ; set 4 LED LSBs on
       CMA
       STA 3000H ; output
       CALL DELB ;delay
       MVI A,00H ; set 4 LED LSBs off
       CMA
       STA 3000H ;output
       CALL DELB ;delay
       JMP START ;go to start and check for a new number
       END
```

4η Άσκηση

- Για την πρώτη τεχνολογία το κόστος σχεδίασης είναι 15.000€ και το κόστος ανά τεμάχιο για την κατασκευή 20€ (10€ για τα IC και 10€ για την πλακέτα). Έτσι το κόστος ανά τεμάχιο είναι:

$$Κόστος_1 = \frac{15000}{x} + 20\,€$$

- Για την δεύτερη τεχνολογία το κόστος σχεδίασης είναι 7.000€ και το κόστος ανά τεμάχιο για την κατασκευή 60€ (50€ για τα IC και 10€ για την πλακέτα). Έτσι το κόστος ανά τεμάχιο είναι:

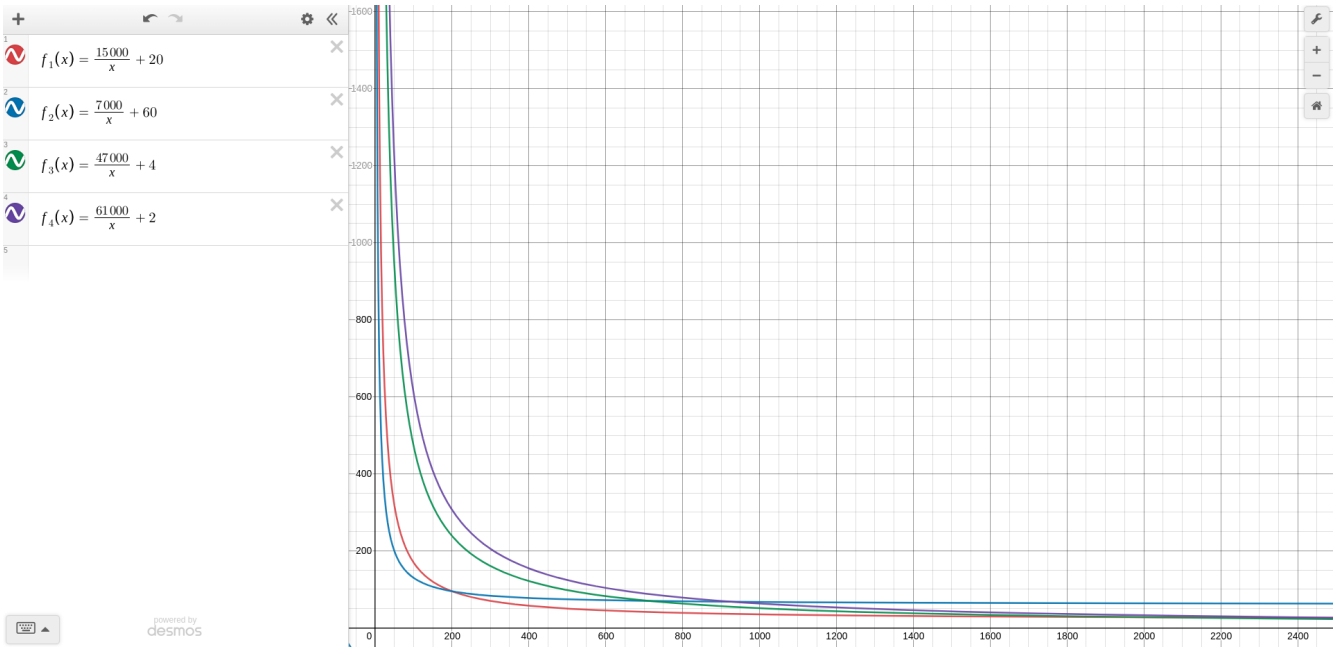
$$Κόστος_2 = \frac{7000}{x} + 60\,€$$

- Για την τρίτη τεχνολογία το κόστος σχεδίασης είναι 47.000€ και το κόστος ανά τεμάχιο για την κατασκευή 4€ (2€ για τα IC και 2€ για την πλακέτα). Έτσι το κόστος ανά τεμάχιο είναι:

$$Κόστος_3 = \frac{47000}{x} + 4\,€$$

- Για την τέταρτη τεχνολογία το κόστος σχεδίασης είναι 61.000€ και το κόστος ανά τεμάχιο για την κατασκευή 2€ (1€ για τα IC και 1€ για την πλακέτα). Έτσι το κόστος ανά τεμάχιο είναι:

$$Κόστος_4 = \frac{61000}{x} + 2\,€$$



Παρατηρούμε πως αν ο αριθμός των τεμαχίων είναι μικρότερος των 200 μας συμφέρει η δεύτερη τεχνολογία (FPGAs).
Για αριθμό τεμαχίων από 200 μέχρι 2000 μας συμφέρει η πρώτη τεχνολογία.
Από 2000 μέχρι 7000 τεμάχια μας συμφέρει η τρίτη τεχνολογία.

Από τα 7000 τεμάχια και πάνω μας συμφέρει η τέταρτη τεχνολογία.
Δεν θα υπάρξει κάποια ακόμα τομή των καμπυλών λόγω του ότι διατηρούν μονοτονία στους θετικούς ακεραίους.

Για να εξαφανιστεί η πρώτη τεχνολογία μειώνοντας το κόστος IC ανά τεμάχιο στην τεχνολογία FPGA, θα πρέπει να κάνουμε την δεύτερη επιλογή πιο συμφέρουσα στην περιοχή τεμαχίων που συμφέρει η πρώτη (200 έως 2000). Λύνουμε δηλαδή την ανίσωση:

$\frac{15000}{n}+20>\frac{7000}{n}+(x+10)$, όπου n ο αριθμός τεμαχίων και χ το κόστος των IC.

$\frac{15000}{n}+20>\frac{7000}{n}+(x+10)\rightarrow x<\frac{8000}{n}+10$

Αυτό θα πρέπει να ισχύει για κάθε n ανήκει στο [200,2000] και επειδή το δεξί μέλος είναι φθίνουσα συνάρτηση του n, αρκεί να ισχύει για το πάνω όριο του διαστήματος, δηλαδή για n=2000, x < 14€.

Άρα αν τα IC στην τεχνολογία με τα FPGA κοστίζουν 14€ και κάτω, τότε η πρώτη τεχνολογία δεν συμφέρει για καμία περιοχή του αριθμού τεμαχίων.

Παρατηρούμε πως η κάθε τεχνολογία συμφέρει για διαφορετικό αριθμό παραγόμενων τεμαχίων. Τεχνολογίες με ακριβό κόστος σχεδίασης (αν αυτό συνεπάγεται με μικρό κατασκευής και συναρμολόγησης ανά τεμάχιο) συμφέρουν για μεγάλο αριθμό παραγόμενων τεμαχίων. Από την άλλη, για λίγα παραγόμενα τεμάχια συμφέρουν τεχνολογίες που παρά το υψηλό κόστος συναρμολόγησης και κατασκευής ανά τεμάχιο δεν έχουν υψηλό σχεδιαστικό κόστος. Αυτό συμβαίνει διότι το κόστος σχεδίασης “μοιράζεται” κατά κάποιο τρόπο σε κάθε παραγόμενο/πωλούμενο τεμάχιο.

Τέλος αναφοράς

Μιχάλης Τσιλιμιγκουνάκης (el19001)
Παναγιώτης Στεφανής (el19096)