

Συστήματα Μικροϋπολογιστών

1^η ομάδα ασκήσεων

Νασοπούλου Ελένη – 03121087

Άσκηση 1

➔ Disassembly

Αρχικά, πραγματοποιούμε την διαδικασία αποκωδικοποίησης (disassembly) της γλώσσας μηχανής, αξιοποιώντας τον πίνακα 2 του παραρτήματος 2 των σημειώσεων.

Machine language	8085 Assembly
06 01	MVI B,01H
3A 00 20	LDA 2000H
FE 00	CPI 00H
CA 13 08	JZ 0813H
1F	RAR
DA 12 08	JC 0812H
04	INR B
C2 0A 08	JNZ 080AH
78	MOV A,B
2F	CMA
32 00 30	STA 3000H
CF	RST 1

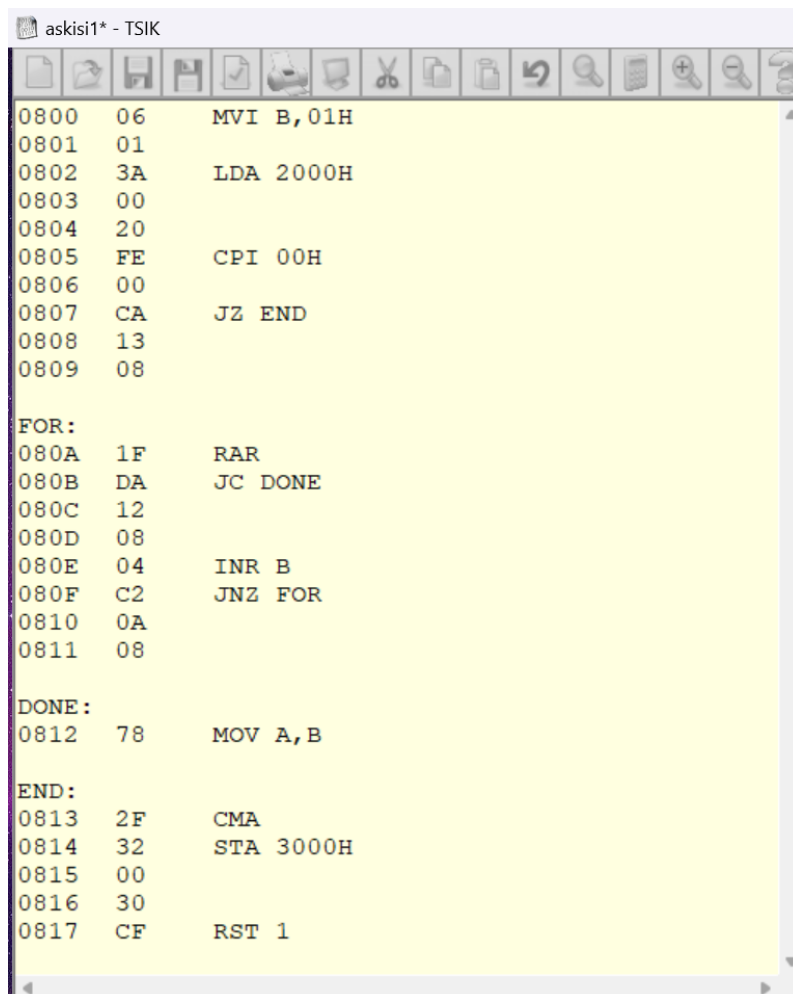
Στον παρακάτω πίνακα φαίνεται πιο αναλυτικά η παραπάνω διαδικασία, καθώς αντιστοιχούμε κάθε διεύθυνση στο περιεχόμενο της σε γλώσσα μηχανής και σε assembly 8085, υποθέτοντας ότι το πρόγραμμα είναι φορτωμένο στη μνήμη με αρχή τη διεύθυνση 0800H. Επιπλέον, προσθέτουμε συμβολικές διευθύνσεις ως εξής:

- 080AH: FOR
- 0812H: DONE
- 0813H: END

Address	Machine Language	Address lable	Assembly
0800	06 - opcode		MVI B,01H
0801	01 - data		
0802	3A - opcode		LDA 2000H
0803	00 - data		
0804	20 - data		
0805	FE - opcode		CPI 00H
0806	00 - data		
0807	CA - opcode		JZ END
0808	13 - data		
0809	08 - data		
080A	1F - opcode	FOR	RAR
080B	DA - opcode		JC DONE

080C	12 - data		
080D	08 - data		
080E	04 - opcode		INR B
080F	C2 - opcode		JNZ FOR
0810	0A - data		
0811	08 - data		
0812	78 - opcode	DONE	MOV A,B
0813	2F - opcode	END	CMA
0814	32 - opcode		STA 3000H
0815	00 - data		
0816	30 - data		
0817	CF - opcode		RST 1

Η παραπάνω ανάλυση μπορεί να επαληθευτεί μέσω του προσομοιωτή, πατώντας το INSTR STEP, το οποίο δίνει τα εξής αποτελέσματα:



```

askisi1* - TSIM
0800 06 MVI B,01H
0801 01
0802 3A LDA 2000H
0803 00
0804 20
0805 FE CPI 00H
0806 00
0807 CA JZ END
0808 13
0809 08

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

DONE:
0812 78 MOV A,B

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

```

➔ Flowchart

Στη συνέχεια προκειμένου να υλοποιήσουμε με επιτυχία το διάγραμμα ροής του κώδικα προχωράμε σε ανάλυση του assembly κώδικα.

Addr label	Assembly	Επεξήγηση
	MVI B,01H	Αρχικοποίηση του καταχωρητή B με 1
	LDA 2000H	Φόρτωση της εισόδου στον καταχωρητή A
	CPI 00H	Σύγκριση εισόδου με το 0
	JZ END	Αν (είσοδος = 0), τότε άλμα στην END
FOR:	RAR	Περιστροφή μέσω κρατουμένου (το flag CY παίρνει την τιμή του LSB)
	JC DONE	Αν (CY = 1), τότε άλμα στην DONE
	INR B	Αύξηση του B κατά 1
	JNZ FOR	Αν (B != 0), τότε άλμα στο FOR *
DONE:	MOV A,B	Προώθηση του B στον A
END:	CMA	Αντιστροφή του A
	STA 3000H	Προώθηση του A στην έξοδο
	RST 1	Διακοπή προγράμματος

* Αυτό το άλμα θα γίνεται πάντα, καθώς το B είναι αδύνατο να λάβει την τιμή 0 (ξεκινάει από την τιμή 1 και σταδιακά αυξάνεται)

Έπειτα από αυτή την ανάλυση, μπορούμε να κατανοήσουμε πλήρως την λειτουργία του προγράμματος. Πιο συγκεκριμένα, το πρόγραμμα δέχεται ως είσοδο έναν αριθμό των 8 bit και τυπώνει έναν άλλο αριθμό των 8 bit, που αντιστοιχεί στη θέση του δεξιότερου 1 άμα ο αριθμός εισόδου θεωρηθεί πως είναι σε δυαδική αναπαράσταση. Τυπώνει, δηλαδή, τον αριθμό που αντιστοιχεί στη θέση του δεξιότερου αναμμένου led εισόδου σε δυαδική αναπαράσταση. Επίσης, στην περίπτωση που κανένα led δεν είναι αναμμένο, τυπώνει 0.

Το διάγραμμα βρίσκεται στην επόμενη σελίδα.

➔ Continuous operation

Τέλος, προκειμένου να πετύχουμε τη συνεχή λειτουργία του προγράμματος πρέπει να υλοποιήσουμε τις εξής αλλαγές:

- προσθήκη ετικέτας START στην αρχή του προγράμματος
- προσθήκη εντολής JMP START στον τέλος του προγράμματος ακριβώς πριν την εντολή RST 1

Ο αντίστοιχος κώδικας φαίνεται παρακάτω.

```

askisi1 - TSIK
START:
    MVI B,01H
    LDA 2000H
    CPI 00H
    JZ END

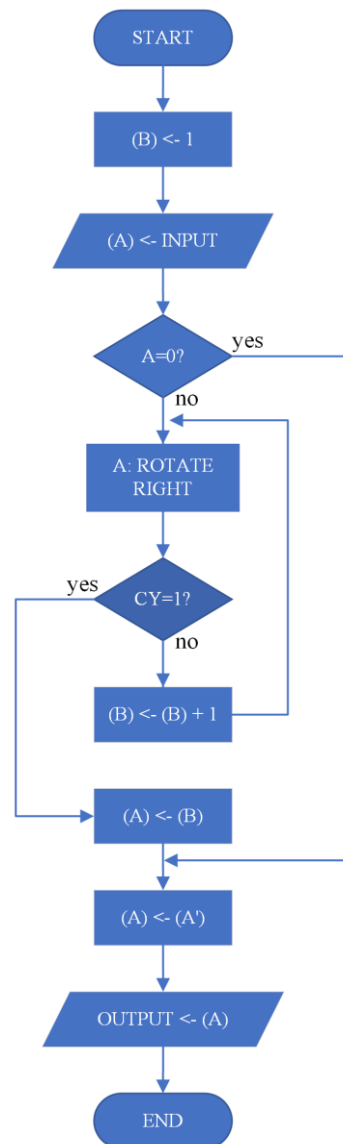
FOR:
    RAR
    JC DONE
    INR B
    JNZ FOR

DONE:
    MOV A,B

END:
    CMA
    STA 3000H
    JMP START
    RST 1

END

```



Άσκηση 2

Ο κώδικας υλοποιήθηκε με βάση τις προδιαγραφές της εκφώνησης, δίνοντας έμφαση στα εξής σημεία:

Στην αρχή χρησιμοποιούμε την εντολή `IN 10H`, που αίρει την προστασία της μνήμης, επιτρέποντας έτσι πρόσβαση για αποθήκευση μεταβλητών και δεδομένων οπουδήποτε στην διαθέσιμη μνήμη RAM του συστήματος.

Επιπλέον, εισάγουμε την χρονοκαθυστέρηση μέσω της ρουτίνα `DELB`, η οποία προκαλεί μεταβλητή καθυστέρηση ίση με την τιμή του ζεύγους BC επί 1 ms, οπότε αρχικοποιούμε το ζεύγος με την τιμή `01FEH = 510` που οδηγεί σε χρονοκαθυστέρηση $\approx 0.5\text{sec}$.

Ακόμα, ο καταχωρητής E περιέχει πάντα την προηγούμενη κατάσταση των LED, οπότε κάθε φορά που επιθυμούμε να τυπώσουμε μία κατάσταση ακολουθούμε τα εξής βήματα:

Assembly	Επεξήγηση
MOV A,E	Ο Α περιέχει την προηγούμενη κατάσταση των LED
...	Ενέργεια ώστε να μεταβούμε στην τρέχουσα κατάσταση
STA 3000H	Προώθηση Α στην έξοδο
MOV E,A	Ενημέρωση του Ε, ώστε να περιέχει ξανά την προηγούμενη κατάσταση

Επίσης, προσέχουμε την αντίστροφη λογική απεικόνισης, οπότε αρχικοποιούμε τον καταχωρητή E με την τιμή FEH= 1111110, ώστε να είναι αρχικά αναμμένο το LSB και στα σημεία που επιθυμούμε να ελέγξουμε αν είναι αναμμένο το LSB ή MSB πραγματοποιούμε σύγκριση με τους αριθμούς FEH= 1111110 και 7EH= 0111111, αντίστοιχα.

```

askisi2 - TSIK
IN 10H
LXI B,01F4H      ; delay of 500ms
MVI E,FEH        ; starting point (LSB is on)

START:
    CALL DELB     ; delay
    LDA 2000H     ; load input
    ANI 03H       ; nullifies all bits except the 2 LSBs
    CPI 01H       ; (1stLSB==1 && 2ndLSB==0)?
    JZ LEFT       ; if yes, move left

RETURN:
    CPI 00H       ; (1stLSB==0 && 2ndLSB==0)?
    JZ CYCLE_L    ; if yes, do left cycle
    MOV A,E       ; the 2ndLSB is on, so stop
    STA 3000H     ; print
    JMP START

LEFT:
    MOV A,E
    CPI 7FH       ; (MSB is on)? 7EH= 01111111
    JZ RIGHT      ; if yes, move right
    RLC           ; rotate left
    STA 3000H     ; print
    MOV E,A
    JMP START

CYCLE_L:
    MOV A,E
    RLC           ; rotate left
    STA 3000H     ; print
    MOV E,A
    JMP START

RIGHT:
    MOV A,E
    CPI FEH       ; (LSB is on)? FEH= 1111110
    JZ LEFT       ; if yes, move left
    RRC           ; rotate right
    STA 3000H     ; print
    MOV E,A
    CALL DELB     ; delay
    LDA 2000H     ; load input
    ANI 03H       ; nullifies all bits except the 2 LSBs
    CPI 01H       ; (1stLSB==1 && 2ndLSB==0)?
    JZ RIGHT      ; if yes, move left
    JMP RETURN    ; else return

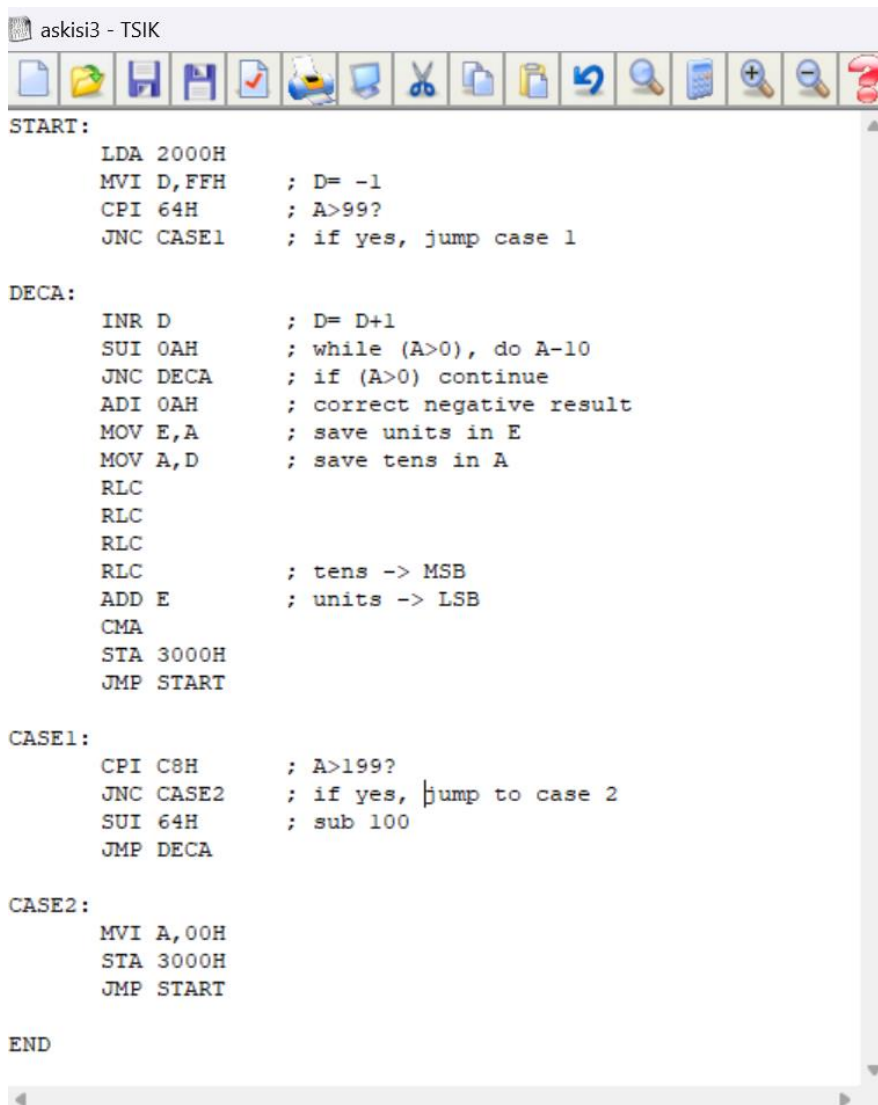
END

```

Άσκηση 3

Ο κώδικας υλοποιήθηκε με βάση τις προδιαγραφές της εκφώνησης και διαχειρίζεται τις διάφορες περιπτώσεις εισόδου με τον εξής τρόπο:

1. $\text{input} < 100$: το διαχειριζόμαστε στο DECA όπου μετατρέπουμε τη δυαδική αναπαράσταση σε δεκαδική.
2. $100 < \text{input} < 200$: το διαχειριζόμαστε στο CASE1, όπου αφαιρούμε 100 ώστε να το ανάγουμε στην πρώτη περίπτωση και έπειτα τυπώνουμε το σωστό output μέσω της DECA.
3. $\text{input} > 200$: το διαχειριζόμαστε στο CASE2, όπου ανάβουν όλα τα LED, με προσοχή στην αντίστροφη λογική απεικόνισης, μέχρι να έχουμε αποδεκτό input.



```
askisi3 - TSIK
START:
    LDA 2000H
    MVI D,FFH    ; D= -1
    CPI 64H      ; A>99?
    JNC CASE1    ; if yes, jump case 1

DECA:
    INR D        ; D= D+1
    SUI 0AH      ; while (A>0), do A-10
    JNC DECA     ; if (A>0) continue
    ADI 0AH      ; correct negative result
    MOV E,A      ; save units in E
    MOV A,D      ; save tens in A
    RLC
    RLC
    RLC
    RLC          ; tens -> MSB
    ADD E        ; units -> LSB
    CMA
    STA 3000H
    JMP START

CASE1:
    CPI C8H      ; A>199?
    JNC CASE2    ; if yes, jump to case 2
    SUI 64H      ; sub 100
    JMP DECA

CASE2:
    MVI A,00H
    STA 3000H
    JMP START

END
```

Άσκηση 4

Προκειμένου να μελετηθεί από τεχνικο-οικονομική άποψη η κατασκευή μια ηλεκτρονικής συσκευής με την χρήση τεσσάρων διαφορετικών τεχνολογιών, υπολογίζουμε τις σχέσεις και σχεδιάζουμε τις καμπύλες του κόστους ανά τεμάχιο.

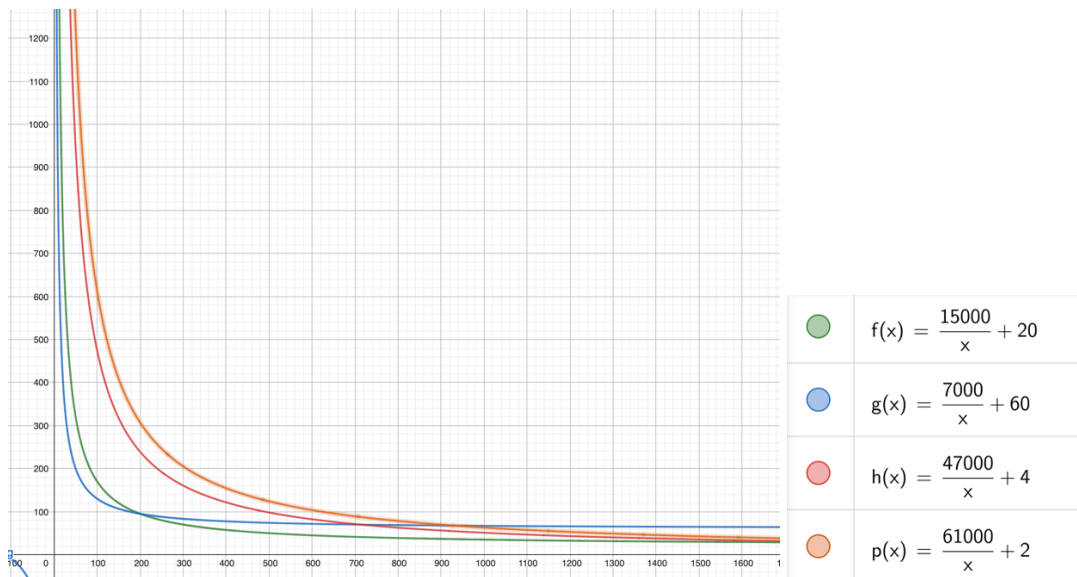
➔ Σχέσεις και καμπύλες κόστους ανά τεμάχιο

Η σχέση για την κάθε καμπύλη κόστους παραγωγής ανά τεμάχιο:

$$f(x) = \frac{\text{fixed cost}}{\text{\#units}} + \text{variable cost/unit}$$

	Technology	FC	VC/unit	Function
1.	IC	15.000	20	$f(x) = \frac{15.000}{x} + 20$
2.	FPGA	7.000	60	$g(x) = \frac{7.000}{x} + 60$
3.	SoC-1	47.000	4	$h(x) = \frac{47.000}{x} + 4$
4.	SoC-2	61.000	2	$p(x) = \frac{61.000}{x} + 2$

Οι αντίστοιχες καμπύλές φαίνονται παρακάτω.



➔ Περιοχές χαμηλότερου κόστους ανά τεχνολογία

Προκειμένου να βρούμε την περιοχή του πλήθος τεμαχίων για να έχουμε το χαμηλότερο κόστος τεχνολογίας πρέπει να βρούμε τα σημεία τομής των καμπυλών.

- $f(x)=g(x) \Rightarrow x=200$
- $f(x)=h(x) \Rightarrow x=2.000$
- $f(x)=p(x) \Rightarrow x=2.556$
- $g(x)=h(x) \Rightarrow x=715$
- $g(x)=p(x) \Rightarrow x=931$
- $h(x)=p(x) \Rightarrow x=7.000$

#units	0-200	200-2.000	2.000-7.000	7.000+
Best technology	FPGA	IC	SoC-1	SoC-2

➔ Εξάλειψη 1^{ης} επιλογής

Έστω y το κόστος ανά τεμάχιο των I.C. για την τεχνολογία FPGA. Τότε η νέα συνάρτηση κόστους ανά τεμάχιο είναι η εξής: $q(x) = \frac{7.000}{x} + 10 + y$

Η ελάχιστη τιμή κόστους y ώστε να εξαφανιστεί η 1^η επιλογή μπορεί να βρεθεί ως εξής:

$$q(x) > f(x) \Rightarrow \frac{7.000}{x} + 10 + y > \frac{15.000}{x} + 20 \Rightarrow y > \frac{8.000}{x} + 10$$

Αυτή η σχέση, λοιπόν, πρέπει να ισχύει για $0 < x < 2.000$, καθώς από εκείνο το σημείο και μετά η οικονομικότερη τεχνολογία είναι η SoC-1, δηλαδή το y πρέπει να είναι τουλάχιστον $y = \frac{8.000}{2.000} + 10 = 14$.

Συνεπώς, κόστος των I.C. για την τεχνολογία FPGA μπορεί να είναι τουλάχιστον 14€ ανά τεμάχιο.