Συστήματα Μικροϋπολογιστών 2018-2019
1η Ομάδα Ασκήσεων

1η Άσκηση

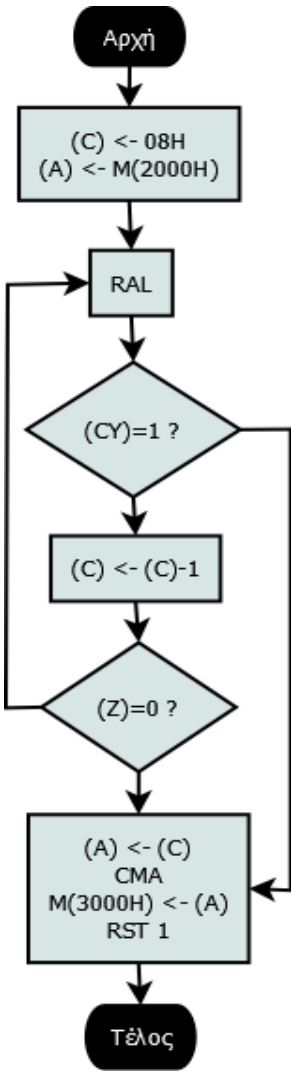
Το πρόγραμμα που δίνεται διαβάζει τον αριθμό που αντιστοιχεί στο MSB των διακοπτών εισόδου σε δεκαδική μορφή και τον εμφανίζει στις λυχνίες εξόδου σε δυαδική μορφή. Για να εκτελείται συνεχώς, θα πρέπει να προστεθεί μία εντολή άλματος στο τέλος του προγράμματος που να μεταφέρει την εκτέλεση στην αρχή του προγράμματος.

Στον παρακάτω πίνακα παρουσιάζεται σε assembly το αρχικό πρόγραμμα και το τροποποιημένο πρόγραμμα με την εντολή άλματος `JMP` για συνεχή λειτουργία και τις απαραίτητες ετικέτες για μετάφραση στον προσομοιωτή TSIK:

Αρχικό πρόγραμμα εκφώνησης	**Επαναλαμβανόμενη εκτέλεση στον προσομοιωτή**

`MVI C,08H`
`LDA 2000H`
`RAL`
`JC 080DH`
`DCR C`
`JNZ 0805H`
`MOV A,C`
`CMA`
`STA 3000H`
`RST 1`
	`START:`
	`  MVI C,08H`
	`  LDA 2000H`
	`GOTO1:`
	`  RAL`
	`  JC GOTO2`
	`  DCR C`
	`  JNZ GOTO1`
	`GOTO2:`
	`  MOV A,C`
	`  CMA`
	`  STA 3000H`
	`  JMP START`
	
	`  END`

Παρακάτω δίνεται ένα διάγραμμα ροής για το πρόγραμμα



2η Άσκηση

Δίνεται το πρόγραμμα σε assembly

```
    IN 10H
    LXI B,01F4H    ;Καθυστέρηση 500ms = 0x1F4
    MVI E,01H      ;Αρχικό LED το LSB
START:
    LDA 2000H
    MOV D,A
    RRC            ;Ολίσθηση δεξιά
    JC START       ;Έλεγχος του LSB
    CALL DELB      ;Καθυστέρηση 0,5s
    MOV A,D
    RLC            ;Ολίσθηση αριστερά
    JC GORIGHT     ;Έλεγχος του MSB
GOLEFT:            ;Κίνηση αριστερά
    MOV A,E        ;Προηγούμενο LED
    CMA            ;Αντίστροφη λογική στα LEDs
    STA 3000H
    CMA
    RLC
    MOV E,A        ;Επόμενο LED
    JMP START
GORIGHT:           ;Κίνηση δεξιά
    MOV A,E
    CMA
    STA 3000H
    CMA
    RRC
    MOV E,A
    JMP START

    END
```

3η Άσκηση

Δίνεται το πρόγραμμα σε assembly (σημειώνεται ότι έχει τροποποιηθεί έτσι ώστε να εμφανίζεται και το δεκαδικό 99)

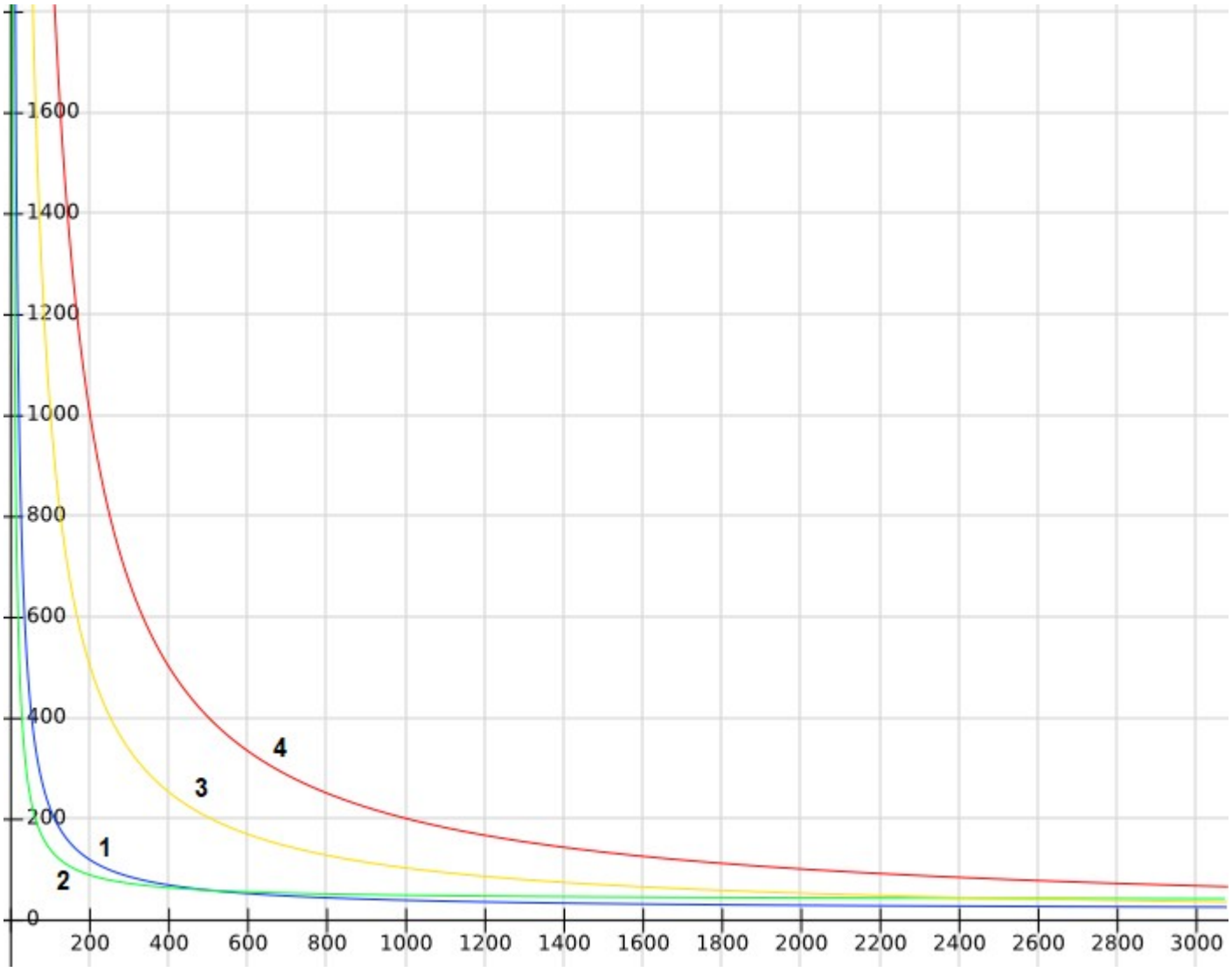
```
    LXI B,01F4H    ;Καθυστέρηση 500ms = 0x1F4
START:
    LDA 2000H
    CPI 64H        ;Σύγκριση με το δεκαδικό 100
    JNC TOOBIG     ;Έλεγχος σχετικά με το δεκαδικό 100
    MVI D,FFH
DECA:
    INR D
    SUI 0AH
    JNC DECA
    ADI 0AH
    MOV E,A
    MOV A,D        ;Δεκάδες
    RLC            ;Ολίσθηση 4 φορές αριστερά
    RLC
    RLC
    RLC
    ADD E          ;Μονάδες
    CMA
    STA 3000H
    JMP START
TOOBIG:            ;Είσοδος μεγαλύτερη του δεκαδικού 100
    MVI A,0FH      ;4 MSB
    STA 3000H
    CALL DELB      ;Καθυστέρηση 0,5s
    CMA            ;4 LSB
    STA 3000H
    CALL DELB
    JMP START

    END
```

4η Άσκηση

Οι συναρτήσεις κόστους ανά τεμάχιο για κάθε τεχνολογία φαίνονται στον παρακάτω πίνακα

1η	2η	3η	4η
$\frac{20000+20x}{x}$	$\frac{10000+40x}{x}$	$\frac{100000+4x}{x}$	$\frac{200000+2x}{x}$

ενώ οι αντίστοιχες γραφικές παραστάσεις φαίνονται στο επόμενο διάγραμμα



Εξισώνοντας τις εκφράσεις των καμπυλών ανά 2, βρίσκουμε τα σημεία τομής των καμπυλών μεταξύ τους

1η-2η	2η-3η	1η-3η	2η-4η	1η-4η	3η-4η
$x=500$	$x=2500$	$x=5000$	$x=5000$	$x=10000$	$x=50000$

Μελετώντας τον προηγούμενο πίνακα και το διάγραμμα των γραφικών παραστάσεων, εξάγουμε τα διαστήματα τιμών του αριθμού τεμαχίων που ελαχιστοποιούν το κόστος κατασκευής για κάθε τεχνολογία

$$0<x<500 \quad : \textbf{2η}$$
$$500<x<5000 \quad : \textbf{1η}$$
$$5000<x<50000 \quad : \textbf{3η}$$
$$x>50000 \quad : \textbf{4η}$$

Παρατηρούμε ότι οι τεχνολογίες με υψηλό κόστος σχεδίασης γίνονται συμφέρουσες μόνο σε υψηλούς αριθμούς τεμαχίων.

Αν z το κόστος ανά IC για την τεχνολογία των FPGAs, τότε το συνολικό κόστος κατασκευής για τη 2η τεχνολογία θα είναι μικρότερο αυτού της 1ης όταν

$$\frac{10000+(z+10)x}{x}<\frac{20000+20x}{x} \quad \Rightarrow \quad z<\frac{10000}{x}+10$$

Για $z\le 10$ το συνολικό κόστος κατασκευής της 2ης τεχνολογίας είναι πάντα μικρότερο από αυτό της 1ης, άρα για να αποκλειστεί η 1η τεχνολογία πρέπει το κόστος ανά IC της 2ης να είναι το πολύ **10€** ανά τεμάχιο.

5η Άσκηση

(i)

```
// Περιγραφή Verilog με μοντελοποίηση επιπέδου πυλών
// για το πρόβλημα 3-31, σχήμα 3.20α
module Circuit_3_20a_gates (A, B, C, D, F);
    output F;
    input A, B, C, D;
    wire w1, w2, w3, w4, w5;

    not G1 (w1, C);
    and G2 (w2, B, w1);
    and G3 (w3, C, D);
    or G4 (w4, w3, B);
    and G5 (w5, w4, A);
    or G6 (F, w5, w2);
endmodule
```

```
// Περιγραφή Verilog με μοντελοποίηση επιπέδου πυλών
// για το πρόβλημα 3-31, σχήμα 3.21β
module Circuit_3_21b_gates (A, B, C, D, F);
    output F;
    input A, B, C, D;
    wire w1, w2, w3, w4, w5, w6, w7, w8;

    not G1(w1, A), G2(w2, B), G3(w3, C), G9(F, w8);
    nand G4(w4, A, w2), G5(w5, w1, B), G8(w8, w6, w7);
    nor G6(w7, w3, D), G7(w6, w4, w5);
endmodule
```

```
// Περιγραφή Verilog με μοντελοποίηση επιπέδου πυλών
// για το πρόβλημα 3-31, σχήμα 3.24
module Circuit_3_24_gates (A, B, C, D, E, F);
    output F;
    input A, B, C, D, E;
    wire w1, w2, w3;

    not G1(w1, E);
    nor G2(w2, A, B), G3(w3, C, D);
    nand G4(F, w2, w3);
endmodule
```

```
// Περιγραφή Verilog με μοντελοποίηση επιπέδου πυλών
// για το πρόβλημα 3-31, σχήμα 3.25
module Circuit_3_25_gates (A, B, C, D, F);
    output F;
    input A, B, C, D;
    wire w1, w2, w3, w4, w5, w6, w7;

    not G1(w1, A), G2(w2, B), G3(w3, D);
    nand G4(w4, w1, B), G5(w5, A, w2), G8(F, w7, w6);
    nor G6(w6, C, w3), G7(w7, w4, w5);
endmodule
```

(ii)

```
// Περιγραφή Verilog με μοντελοποίηση ροής δεδομένων
// για το πρόβλημα 3-32, σχήμα 3.20β
module Circuit_3_20b_dataflow (F, A, B, C, D);
    output F;
    input A, B, C, D;

    assign F=(!(!((((!(!(C&&D)))||(!(!B)))&&A)))||(!(!(B&&(!C)))));
endmodule

// Περιγραφή Verilog με μοντελοποίηση ροής δεδομένων
// για το πρόβλημα 3-32, σχήμα 3.21α
module Circuit_3_21a_dataflow (F, A, B, C, D);
    output F;
    input A, B, C, D;

    assign F=((A&&(!B))||((!A)&&B))&&(C||(!D));
endmodule

// Περιγραφή Verilog με μοντελοποίηση ροής δεδομένων
// για το πρόβλημα 3-32, σχήμα 3.24
module Circuit_3_24_dataflow (F, A, B, C, D, E);
    output F;
    input A, B, C, D, E;

    assign F=(!(!(A||B)))&&(!(!(C||D)))&&(!(!E));
endmodule

// Περιγραφή Verilog με μοντελοποίηση ροής δεδομένων
// για το πρόβλημα 3-32, σχήμα 3.25
module Circuit_3_25_dataflow (F, A, B, C, D);
    output F;
    input A, B, C, D;

    assign F=(!(!((((!(!A))&&(!B))||((!A)&&(!(!B))))))&&(!(!(C||(!D)))));
endmodule
```

6η Άσκηση

(i)

```
// Περιγραφή Verilog με μοντελοποίηση επιπέδου πυλών
// για το πρόβλημα 4-36 (σχήμα 4.23)
module Circuit_4_36_gates (x, y, V, D);
    output x, y, V;
    input [3:0]D;
    wire w1, w2;

    not
        G1(w1, D[2]);
    and
        G2(w2, w1, D[1]);
    or
        G3(y, D[3], w2),
        G4(x, D[3], D[2]),
        G5(V, x, D[1], D[0]);
endmodule
```

(ii)

```
// Περιγραφή Verilog με μοντελοποίηση συμπεριφοράς
// για το πρόβλημα 4-45 (σχήμα 4.23)
module Circuit_4_45_behavioral (x, y, V, D);
    output reg x, y, V;
    input [3:0]D;

    always @(D)
        if (D[3]) begin y=1; x=1; V=1; end
        if (D[2]) begin y=0; x=1; V=1; end
        if (D[1]) begin y=1; x=0; V=1; end
        if (D[0]) begin y=0; x=0; V=1; end
        else V=0;
endmodule
```

Σημείωση: Η επεξεργασία του κώδικα των προγραμμάτων έγινε στο **Notepad++** 7.6.6 portable, το διάγραμμα ροής σχεδιάστηκε στο **Dia** 0.97.2 Rev 2 portable, οι γραφικές παραστάσεις σχεδιάστηκαν στο **FooPlot** και η συγγραφή της παρούσας αναφοράς έγινε στο **LibreOffice** 6.3.1 portable.