Μικροϋπολογιστές

1η σειρά ασκήσεων

Λιαροκάπης Αλέξανδρος (03114860)
Μάνος Απόστολος (03114855)

Άσκηση 1

Σε ένα μΥ-Σ 8085 να γραφεί σε assembly το παρακάτω πρόγραμμα που δίνεται σε γλώσσα μηχανής και να εξηγηθεί η λειτουργία του:

0E 08 **3A** 00 20 **17 DA** 0D 08 **0D C2** 05 08 **79 2F 32** 00 30 **CF**

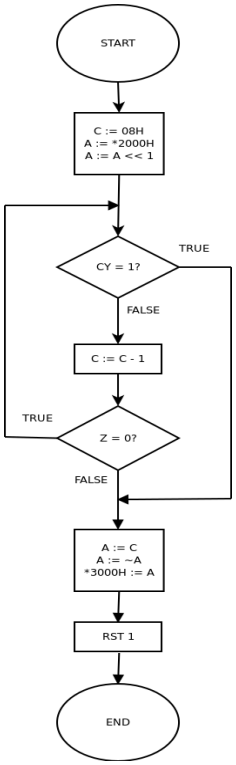
Το πρόγραμμα υποθέτουμε ότι είναι φορτωμένο στη μνήμη με αρχή τη διεύθυνση 0800 και δίνεται για διευκόλυνσή σας ότι οι bold κωδικοί είναι εντολές.

Λύση

Παρατίθεται το πρόγραμμα τροποποιημένο για να λειτουργεί με τον simulator:

```
          IN 10H
          MVI C,08H ; we loop at most 8 times - once for each bit
          LDA 2000H
ASK1_LOOP: RAL ; we left shift the value
          JC ASK1_END ; if the uppermost bit was 1 we stop the loop
          DCR C ; otherwise we decrease the counter
          JNZ ASK1_LOOP ; and retry
ASK1_END:  MOV A,C
          CMA
          STA 3000H
          RST 1
          END
```

Καθώς επίσης και το flow graph:



Το παραπάνω πρόγραμμα παίρνει την είσοδο μέσω των dip switch, υπολογίζει τη θέση του πρώτου απο τα αριστερά bit που είναι 1 και μας τον δίνει σε δυαδική μορφή μέσω των led.

Άσκηση 2

Να γραφεί σε assembly πρόγραμμα που να απεικονίζει ένα αναμμένο led το οποίο να κινείται αριστερά (από το LSB προς το MSB) και να συνεχίζει να κινείται κυκλικά (θέσεις led 0123456701... κ.λπ.) όταν το MSB της θύρας των dip switch είναι OFF. Αλλιώς, όταν το MSB των dip switch γίνεται ON να αναστρέφεται η κατεύθυνση και το αναμμένο led να κινείται δεξιά (από το MSB προς το LSB). Τέλος, όταν το LSB της θύρας των dip switch γίνεται ON, το led να σταματάει εκεί που βρίσκεται. Στη συνέχεια, όταν ξαναγίνει OFF να συνεχίζεται η κίνησή του σύμφωνα με το MSB των dip switch. Να γίνει χρήση της θύρας εισόδου dip switch (θέση μνήμης 2000 Hex) και της θύρας εξόδου των LED (που αντιστοιχεί στη θέση μνήμης 3000 Hex – προσοχή στην αντίστροφη λογική απεικόνισης). Στη συνέχεια, όταν ξαναγίνει ON να συνεχίζεται η κίνηση του αναμμένου led. (Διάρκεια ανάμματος $\sim 1/2$ sec).

Λύση

```
             IN 10H
             MVI A,7FH ; 01111111 bit pattern - leds would output 10000000
             STA 3000H ;
             MOV D,A ; we store A to D since we will have to reuse A.
ASK2_LOOP:   LDA 2000H
             MOV C,A ; we store A to C since we will follow up with branching tests
             ANI 01H
             JNZ ASK2_LOOP ; if the LSB is 1 we restart without shifting anything.
             MOV A,C
             ANI 80H
             JNZ ASK2_RIGHT ; otherwise if the MSB is 1 we shift right and retry.
             MOV A,D ; otherwise we shift left.
             RLC
             MOV D,A
             STA 3000H
             JMP ASK2_NEXT
ASK2_RIGHT:  MOV A,D
             RRC
             MOV D,A
             STA 3000H
ASK2_NEXT:   LXI B, 01F4H ; called when shifting to create a delay of approximately 500ms.
             CALL DELB
             JMP ASK2_LOOP
             END
```

Άσκηση 3

Να επεκταθεί το 4ο παράδειγμα που αφορά στη μετατροπή δυαδικού αριθμού των 8 bits σε δεκαδική μορφή 2 ψηφίων (σελ. 84 του βιβλίου) χωρίς τον περιορισμό να είναι μικρότεροι του 100_10 . Τα 8 bit του δυαδικού αριθμού υποθέτουμε δίνονται από τα dip switches της πόρτας εισόδου (θέση μνήμης 2000 Hex). Το αποτέλεσμα να εμφανισθεί στη ν πόρτα εξόδου των LED (που αντιστοιχεί στη θέση 3000 Hex) ως εξής: οι μονάδες στα 4 LSB και οι δεκάδες 4 MSB. Στην περίπτωση που ο αριθμός είναι μεγαλύτερος του 99 , να αναβοσβήνουν συνεχώς και εναλλασσόμενα τα 4 LSB και τα 4 MSB των LED (όταν ανάβουν τα LSB να σβήνουν τα MSB και αντίστροφα - επιλέξτε ένα ρυθμό που να είναι ορατός) . Το πρόγραμμα να είναι συνεχούς λειτουργίας.

Λύση

```
        IN 10H
START:  LDA 2000H
        CPI 64H
        JNC BLINK ; if > 99 blink leds and restart
        MVI B,0FFH

DECA:   INR B
        SUI 0AH
        JNC DECA ; if positive keep substracting
        ADI 0AH ; otherwise we correct the remainder
                ; at this point A contains the digit in the units positions
                ; while B contains the digit in the tens position

        MOV C,A
        MOV A,B
        RLC
        RLC
        RLC
        RLC ; A now contains the digit in the tens position shifted to the upper 4 bits
        ORA C ; we OR A with C so that the digit in the unit position is
              ; now contained in the lower 4 bits.
        CMA
        STA 3000H ; we now output A to the leds
        JMP START ; and restart

BLINK:  MVI A,0FH
        STA 3000H
        LXI B,01F4H
        CALL DELB
        MVI A,0F0H
        STA 3000H
        CALL DELB
        JMP START
        END
```

Άσκηση 5

i. Από το πρόβλημα 3 - 31 τα σχήματα: 3.20α , 3.21β, 3.24, 3.25 (σε επίπεδο πυλών).

ii. Από το πρόβλημα 3 - 32 τα σχήματα: 3.20β, 3.21α, 3.24, 3.25 (μοντελοποίηση ροής δεδομένων με εντολές συνεχούς ανάθεσης).

Λύση

i.
```
module sxhma320a(A, B, C, D, F);
    input A, B, C, D;
    output F;
    wire w1, w2, w3, w4;
    and (w1, C, D);
    and (w2, B, ~C);
    or (w3, w1, B);
    and (w4, w3, A);
    or (F, w4, w2);
endmodule

module sxhma321b(A, B, C, D, F);
    input A, B, C, D;
    output F;
    wire w1, w2, w3, w4, w5;
    nand (w1, A, ~B);
    nand (w2, ~A, B);
    or (w3, C, ~D);
    or (w4, ~w1, ~w2);
    and (w5, w4, w3);
    not (F, w5);
endmodule

module sxhma324(A, B, C, D, E, F);
    input A, B, C, D, E;
    output F;
    wire w1, w2;
    or (w1, A, B);
    or (w2, C, D);
    and (F, w1, w2, ~E);
endmodule

module sxhma325(A, B, C, D, F);
    input A, B, C, D;
    output F;
    wire w1, w2, w3, w4;
    and (w1, A, ~B);
    and (w2, ~A, B);
    or (w3, C, ~D);
    or (w4, w1, w2);
    and (F, w4, w3);
endmodule
```

ii.

```
module sxhma320a(A, B, C, D, F);
    input A, B, C, D;
    output F;
    wire w1, w2, w3, w4;
    and (w1, C, D);
    and (w2, B, ~C);
    or (w3, w1, B);
    and (w4, w3, A);
    or (F, w4, w2);
endmodule

module sxhma321b(A, B, C, D, F);
    input A, B, C, D;
    output F;
    wire w1, w2, w3, w4, w5;
    nand (w1, A, ~B);
    nand (w2, ~A, B);
    or (w3, C, ~D);
    or (w4, ~w1, ~w2);
    and (w5, w4, w3);
    not (F, w5);
endmodule

module sxhma324(A, B, C, D, E, F);
    input A, B, C, D, E;
    output F;
    wire w1, w2;
    or (w1, A, B);
    or (w2, C, D);
    and (F, w1, w2, ~E);
endmodule

module sxhma325(A, B, C, D, F);
    input A, B, C, D;
    output F;
    wire w1, w2, w3, w4;
    and (w1, A, ~B);
    and (w2, ~A, B);
    or (w3, C, ~D);
    or (w4, w1, w2);
    and (F, w4, w3);
endmodule
```

Άσκηση 6

i. Το πρόβλημα 3 - 33 χωρίς την προσομοίωση.

ii. Το πρόβλημα 3 - 34 χωρίς το πρόγραμμα δοκιμαστικής εισόδου.

iii. Το πρόβλημα 3 - 36.

Λύση

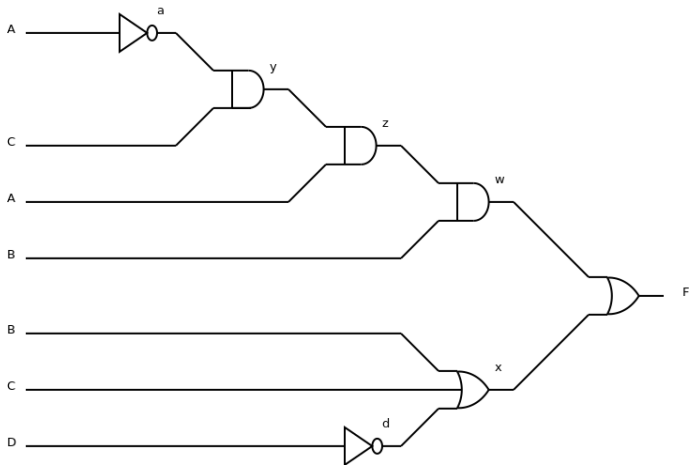
ia.

ns	x	y	x'	y'	x&y'	y&x'	xor
-1	0	0	1	1	0	0	0
0	0	1	1	1	0	0	0
2	0	1	1	1	0	0	0
4	0	1	1	0	0	0	0
6	0	1	1	0	0	0	0
8	0	1	1	0	0	1	0
10	0	1	1	0	0	1	0
12	0	1	1	0	0	1	0
14	0	1	1	0	0	1	0
16	0	1	1	0	0	1	0
18	0	1	1	0	0	1	1
20	0	1	1	0	0	1	1
22	0	1	1	0	0	1	1
50	0	1	1	0	0	1	1

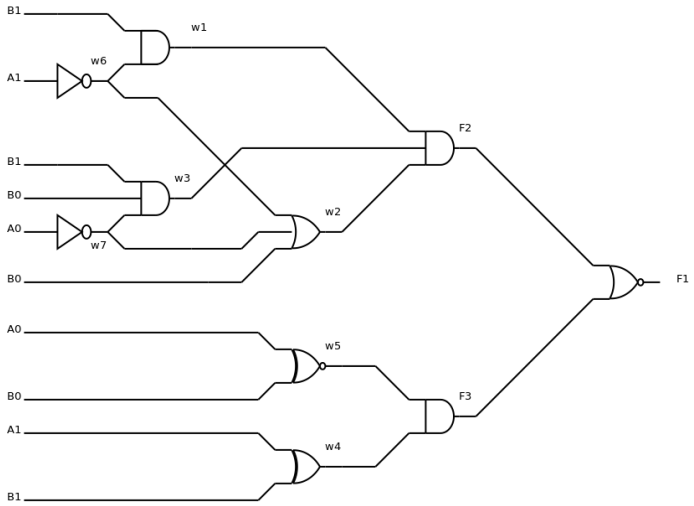
ib.
```
module askisi6i(X, Y, F);
    input X, Y;
    output F;
    wire w1, w2, w3, w4;
    not #4 (w1, X);
    not #4 (w2, Y);
    and #8 (w3, X, w2);
    and #8 (w4, Y, w1);
    or #10 (F, w3, w4);
endmodule
```

ii.
```
module askisi334(A, B, C, D, Out_1,Out_2, Out_3);
    input A,B,C,D;
    output Out_1, Out_2, Out_3;
    assign Out_1 = (A | ~B) & ~C & (C | D);
    assign Out_2 = ((~C & D) | (B & C & D) | (C & ~D)) & (~A | B);
    assign Out_3 = (A & B | C) & D | (~B & C);
endmodule
```

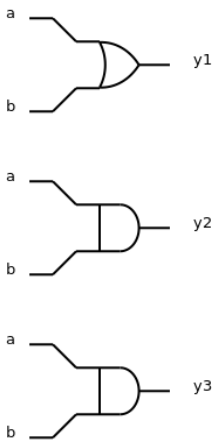
iiia.



iiib.



iiic.



Άσκηση 7

i. Το πρόβλημα 4 - 36.

ii. Το πρόβλημα 4 - 45.

Λύση

i.
```
module askisi7i(D0, D1, D2, D3, x, y, V);
    input D0, D1, D2, D3;
    output x, y, V;
    wire w1, w2;
    not (w1, D2);
    or (x, D2, D3);
    and (w2, w1, D1);
    or (V, x, D0, D1);
    or (y, w2, D3);
endmodule;
```

ii.
```
module askisi7ii(D,X,Y,V);
    input [0:3] D;
    output X,Y,V;
    reg X, Y, V;
    always @(D) begin
        casex (D)
            4'b0000: {X, Y, V} = 3'bxx0;
            4'b1000: {X, Y, V} = 3'b001;
            4'bx100: {X, Y, V} = 3'b011;
            4'bxx10: {X, Y, V} = 3'b101;
            4'bxxx1: {X, Y, V} = 3'b111;
            default: {X, Y, V} = 3'b000;
        endcase
    end
endmodule
```