# Εργαστήριο Μικροϋπολογιστών
# 6η Άσκηση

Ομάδα: Δ12
Βακαλόπουλος Θεόδωρος, ΑΜ: 03114013
Μαυρομμάτης Ιάσων, ΑΜ: 03114771
Νικητοπούλου Δήμητρα, ΑΜ: 03114954

## Άσκηση 1

Ο κώδικας της άσκησης φαίνεται παρακάτω:

```
.include "m16def.inc"
.def reg = r19
.def temp1 = r20
.def temp2 = r21
.def temp3 = r26
.def lastA = r22
.def lastC = r23
.def counter = r24
.def var = r25

main:
ldi reg, low(RAMEND)
out SPL, reg
ldi reg, high(RAMEND)
out SPH, reg
clr lastA
clr lastC
ser reg
out DDRB, reg   ;PortB as output
clr reg
out DDRA, reg   ;PortA as input
out DDRC, reg   ;PortC as input
out PORTB, reg  ;Initialise leds

repeat:
in lastA, PINA      ;Read push buttons PA0-PA7
clr reg         ;Create output in register reg
mov temp1, lastA
andi temp1, 0xc0    ;Isolate PA7-PA6
cpi temp1, 0x00
breq case1
cpi temp1, 0xc0
breq case1
next1:
mov temp1, lastA
andi temp1, 0x30    ;Isolate PA5-PA4
```

```
cpi temp1, 0x00
breq case2
next2:
mov temp1, lastA
andi temp1, 0x0c    ;Isolate PA3-PA2
cpi temp1, 0x00
brne case3
next3:
mov temp2, reg
andi temp2, 0x02
breq next4
mov temp2, lastA
andi temp2, 0x03    ;Isolate PA1-PA0
cpi temp2, 0x01
breq case4
cpi temp2, 0x02
breq case4
next4:
in temp1, PINC
ldi counter, 0x08
ldi var, 0x80
loop3:
rol temp1
brcc next_it
eor reg, var
next_it:
lsr var
dec counter
cpi counter, 0x00
brne loop3
out PORTB, reg      ;Display leds
rjmp repeat

case1:
ldi temp3, 0x08
add reg, temp3
rjmp next1
case2:
ldi temp3, 0x04
add reg, temp3
rjmp next2
case3:
ldi temp3, 0x02
add reg, temp3
rjmp next3
case4:
ldi temp3, 0x01
add reg, temp3
rjmp next4
```

## Σχόλια:

- Η υλοποίηση των διαφόρων πυλών της άσκησης γίνεται απομονώνοντας κάθε φορά τα αντίστοιχα bits που είναι είσοδοι στην τρέχουσα πύλη. Στη συνέχεια εξετάζουμε αν τα bits εισόδου ταυτίζονται με κάποιο από τους συνδυασμούς που δίνουν ως έξοδο της πύλης 1. Αν έχουμε κάποιο από αυτούς τους συνδυασμούς θέτουμε 1 στο αντίστοιχο bit της θύρας εξόδου όπου έχουμε θεωρήσει ότι βρίσκεται η έξοδος της πύλης διαφορετικά προχωράμε στην επόμενη πύλη.
- Ειδικότερα για την περίπτωση του LSB της θύρας εξόδου που αντιστοιχεί στο αποτέλεσμα μιας πύλης AND με εισόδους την έξοδο της προηγούμενης πύλης OR και μιας πύλης OR, ελέγχουμε αρχικά το αποτέλεσμα της προηγούμενης πύλης. Αν αυτό είναι 0 τότε δεν ελέγχουμε καθόλου την πύλη XOR και θέτουμε κατευθείαν LSB=0, διαφορετικά αναθέτουμε στο LSB εξόδου το αποτέλεσμα της πύλης XOR.
- Για την υλοποίηση της αντιστροφής των αποτελεσμάτων στην περίπτωση που πατάμε το αντίστοιχο bit της θύρας PORTC ελέγχουμε σειριακά μέσω του κρατουμένου όλα τα bits (PC7-PC0) εκτελώντας αριστερή ολίσθηση και στην περίπτωση που έχουμε 1 σε κάποιο bit εκτελούμε xor μεταξύ του καταχωρητή με το αποτέλεσμα και μιας μάσκας που έχει 1 στο bit υπό εξέταση, ώστε να αντιστραφεί τελικά το αποτέλεσμα.

## Άσκηση 2

Ο κώδικας της άσκησης φαίνεται παρακάτω:

```
unsigned char input ,output;

void main(){
        DDRA = 0x00;          //Port A as input
        DDRC = 0xFF;          //Port C as output
        output = 0x00;
        PORTC = output;       //Initialise Port C
        unsigned char A, B, C, D, E, F0, F1, F2;
        unsigned char temp1, temp2, temp3;
        while(1){
                input = PINA;
                A = input & 0x01;
                B = (input & 0x02)>>1;
                C = (input & 0x04)>>2;
                D = (input & 0x08)>>3;
                E = (input & 0x10)>>4;
                temp1 = A & B & C;
                temp2 = C & D;
                temp3 = D & E;
                temp2 = temp1 | temp2 | temp3;
                F0 = ~temp2;
                temp2 = (~D) & (~E);
                F1 = temp1 | temp2;
                F2 = F0 | F1;
```

```
                F0 = F0 & 0x01;
                F1 = F1 & 0x01;
                F2 = F2 & 0x01;
                F0 = F0 << 5;
                F1 = F1 << 6;
                F2 = F2 << 7;
                output = F0 | F1 | F2;
                PORTC = output;
        }
}
```

## Σχόλια:

- Για την προσομοίωση των λογικών συναρτήσεων, αρχικά διαβάζουμε την πόρτα εισόδου και στη συνέχεια απομονώνουμε σε ξεχωριστές μεταβλητές τις λογικές μεταβλητές A, B, C, D και E αντίστοιχα. Έπειτα, με κατάλληλες ολισθήσεις ευθυγραμμίζουμε τα bit εισόδου ώστε να βρίσκονται όλα στο LSB της αντίστοιχης λογικής μεταβλητής και εκτελούμε τις πράξεις σε επίπεδο bit.
- Για την απεικόνιση των εξόδων αφού τις δημιουργήσουμε στις μεταβλητές F1, F1 και F2 εκτελούμε κατάλληλες ολισθήσεις ώστε να αντιστοιχούν στα bit 5, 6 και 7 αντίστοιχα της πόρτας εξόδου όπως ορίζεται στην εκφώνηση.

## Άσκηση 3

Ο κώδικας της άσκησης φαίνεται παρακάτω:

```
.include "m16def.inc"

 .def reg = r18
 .def flag = r19
 .def temp = r17

 .DSEG
 _tmp_:.byte 2

 .CSEG

 ldi reg, low(RAMEND)
 out SPL, reg
 ldi reg, high(RAMEND)
 out SPH, reg
 ser reg
 out DDRB, reg                  //PortB as output
 ldi r24 ,(1 << PC7) | (1 << PC6) | (1 << PC5) | (1 << PC4)
 out DDRC ,r24
 ldi r26,low(_tmp_) ;r26-r27 -> X
 ldi r27,high(_tmp_)
 clr reg
 st X+,reg
 st X,reg
 out PORTB,reg
```

```
reading:
ldi flag,0x01 ;0->wrong combination,1->right
ldi r24, 10      ;Initialise for 0.01s delay
rcall scan_keypad_rising_edge
rcall keypad_to_ascii
cpi r24,0x00
breq reading
cpi r24,'1'
breq read2nd
clr flag

read2nd:
ldi r24, 10      ;Initialise registers for 0.01s delay
rcall scan_keypad_rising_edge
rcall keypad_to_ascii
cpi r24,0x00
breq read2nd
cpi r24,'2'
breq leds
clr flag

leds:
cpi flag,0x00
breq wrong
ldi r25, HIGH(4000)
ldi r24, LOW(4000)   //Initialise registers for 4s delay
ser reg
out PORTB,reg
rcall wait_msec
clr reg
out PORTB,reg
jmp reading

wrong:
ldi temp,0x08

on_off:
ser reg
out PORTB,reg
ldi r25, HIGH(250)
ldi r24, LOW(250)    //Initialise registers for 0.25s delay
rcall wait_msec
ldi r25, HIGH(250)
ldi r24, LOW(250)
clr reg
out PORTB,reg
rcall wait_msec
dec temp
brne on_off
jmp reading
```

```
wait_msec:
push r24
push r25
ldi r24 , low(998)
ldi r25 , high(998)
rcall wait_usec
pop r25
pop r24
sbiw r24 , 1
brne wait_msec
ret

wait_usec:
sbiw r24 ,1
nop
nop
nop
nop
brne wait_usec
ret

scan_row:
ldi r25,0x08
back:
lsl r25
dec r24
brne back
out PORTC, r25
nop
nop
in r24, PINC
andi r24, 0x0f
ret

scan_keypad:
ldi r24,0x01
rcall scan_row
swap r24
mov r27,r24
ldi r24,0x02
rcall scan_row
add r27,r24
ldi r24,0x03
rcall scan_row
swap r24
mov r26,r24
ldi r24,0x04
rcall scan_row
add r26,r24
```

```
movw r24,r26
ret

scan_keypad_rising_edge:
mov r22, r24
rcall scan_keypad
push r24
push r25
mov r24, r22
clr r25
rcall wait_msec
rcall scan_keypad
pop r23
pop r22
and r24,r22
and r25,r23
ldi r26,low(_tmp_) ;r26-r27 -> X
ldi r27,high(_tmp_)
ld r23,X+
ld r22,X
st X,r24
st -X,r25
com r23
com r22
and r24,r22
and r25,r23
ret


keypad_to_ascii:
movw r26 ,r24
ldi r24 ,'*'
sbrc r26 ,0
ret
ldi r24 ,'0'
sbrc r26 ,1
ret
ldi r24 ,'#'
sbrc r26 ,2
ret
ldi r24 ,'D'
sbrc r26 ,3
ret
ldi r24 ,'7'
sbrc r26 ,4
ret
ldi r24 ,'8'
sbrc r26 ,5
ret
ldi r24 ,'9'
```

```
sbrc r26 ,6
ret
ldi r24 ,'C'
sbrc r26 ,7
ret
ldi r24 ,'4'
sbrc r27 ,0
ret
ldi r24 ,'5'
sbrc r27 ,1
ret
ldi r24 ,'6'
sbrc r27 ,2
ret
ldi r24 ,'B'
sbrc r27 ,3
ret
ldi r24 ,'1'
sbrc r27 ,4
ret
ldi r24 ,'2'
sbrc r27 ,5
ret
ldi r24 ,'3'
sbrc r27 ,6
ret
ldi r24 ,'A'
sbrc r27 ,7
ret
clr r24
ret
```

## Σχόλια:

- Οι ρουτίνες που χρησιμοποιήθηκαν για το διάβασμα του πληκτρολογίου έχουν παρθεί αυτούσιες από την εκφώνηση.
- Το flag ως καταχωρητής υποδεικνύει το κατά πόσο έχει δοθεί η σωστή είσοδος. Αρχικοποιείται στην τιμή 1 πριν το διάβασμα κάποιου ψηφίου από το πληκτρολόγιο και στη συνέχεια αν πατηθεί διαφορετικό ψηφίο από το 1 την πρώτη φορά ή διαφορετικό από 2 τη δεύτερη φορά (επιθυμητός συνδυασμός <12>) τον μηδενίζουμε, δηλαδή δηλώνουμε ότι δεν έχει πατηθεί ο επιθυμητός συνδυασμός. Ανάλογα με την τιμή του flag εκτελούμε την προκαθορισμένη ενέργεια και στη συνέχεια τον αρχικοποιούμε ξανά στην τιμή 1 για να διαβάσουμε πάλι από το πληκτρολόγιο.
- Αν η ρουτίνα keypad_to_ascii επιστρέψει μέσω του καταχωρητή r24 την τιμή 0 συνεπάγεται ότι δεν έχει διαβαστεί κανένας αριθμός οπότε συνεχίζουμε να διαβάζουμε μέχρι να δοθεί κάποιος αριθμός.
- Το αναβοσβήσιμο των leds στην περίπτωση λάθος συνδυασμού επιτυγχάνεται με μία επαναληπτική διαδικασία 8 επαναλήψεων όπου κάθε φορά ανάβουμε

τα leds για 250ms και στη συνέχεια τα σβήνουμε για 250ms. Έτσι επιτυγχάνουμε συνολική διάρκεια 4s όπως απαιτείται.

Το διάγραμμα ροής της άσκησης είναι το εξής: