### Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Δεκέμβριος 2016

**ΣΥΣΤΗΜΑΤΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ**

# 2η Εργαστηριακή Άσκηση

Ρόπουτης Δημήτρης 8233 Μπαλτζής Ευριπίδης 8196

Ομάδα 6 Group 3

## Περιγραφή της λειτουργίας του προγράμματος

Στη δέυτερη εργαστηριακή άσκηση, μας ζητήθηκε να υλοποιήσουμε ένα σύστημα ασφαλείας, η λειτουργία του οποίου βασιζόταν στην εισαγωγή ενός επιθυμητού 4-ψήφιου κωδικού ως κωδικό ασφαλείας. Στη συνέχεια, με την ενεργοοπίηση του συστήματος, ο χρήστης έπρεπε να εισάγει τον κωδικό αυτό μέσω των switches, ώστε να ξεκλειδώσει το σύστημα και να τεθεί σε ουσιαστική λειτουργία. Για την υλοποίηση αυτών ακολουθήσαμε την εξής διαδικασία:

Αρχικά, θέσαμε ως αρχικη θέση μνήμης τη θέση 100 και αρχικοποιήσαμε τον stack pointer, έτσι ώστε να έχουμε τη δυνατότητα χρήσης subroutines κατά τη διάρκεια εκτέλεσης του προγράμματος. Κατά την εκκίνηση του προγράμματος τέθηκαν όλα τα bit του καταχωρητή data direction register του PortD ίσα με 1 προκειμένου να ορισθεί το PortD ως input port (switches). Αντίστοιχα τα bit του data direction register του PortB τέθηκαν ίσα με 0 προκειμένου να ορισθεί το PortB ως output port (leds).

Αρχικά, πραγματοποιείται έλεγχος σχετικά με το ποια από τα SW0 – SW3 είναι πατημένα, και εφόσον αυτό επιβεβαιωθεί και όλα τα switches απελευθερωθούν, το πρόγραμμα συνεχίζει και περιμένει πλέον το πάτημα του SW7, ώστε ο επιθυμητός εισαχθής κωδικός να αποθηκευτεί.

Στη συνέχεια, με το πάτημα του SW0, το σύστημα ενεργοποιείται και αναμένει από το χρήστη το πάτημα του κωδικού που αποθηκεύτηκε παραπάνω. Η διαδικασία αναμονής έχει μέγιστη διάρκεια 5 δευτερολέπτων (λόγω προδιαγραφών) μέσω ενός counter και της κλήσης ενός delay διάρκειας 1 δευτερολέπτου. Αν η εισαγωγή του κωδικού είναι λανθασμένη (πρώτη αποτυχία) ή εάν ο κωδικός δεν εισαχθή εντός χρονικού ορίου, τότε ενεργοποιείται το LED0 και αναβοσβήνει με ρυθμό 1 δευτερολέπτου μέσω του ίδιου delay με παραπάνω (1 sec ON – 1 sec OFF) και το πρόγραμμα αναμένει από το χρήστη την επόμενη εισαγωγή τυ κωδικού. Αν και η δεύτερη εισαγωγή είναι λανθασμένη, τότε αναβοσβήνουν όλα τα LED με ρυθμό 1 sec, ενώ στη συνέχεια το πρόγραμμα τερματίζει. Ταυτόχρονα, αν κατά τη διάρκεια της εισαγωγής του κωδικού γίνει κάποια από τις παραβάσεις που περιγράφονται στις προδιαγραφές, το κατάλληλο LED ανάβει σε συνδυασμό με το LED2, που οφείλει να παραμένει αναμμένο σε κάθε παράβαση του συστήματος. Στην περίπτωση ορθής εισαγωγής του κωδικού πρόσβασης, το LED1 ανάβει και παραμένει αναμμένο μέχρι να πατηθεί το SW0, που οδηγεί σε επανεκκίνση του συστήματος.

**ΚΩΔΙΚΑΣ**

.include "m16def.inc"

.def temp=r16

.def code1=r17

.def code2=r18

.def code3=r19

.def code4=r20

.def flag=r21

.def counter=r22

.def temp2=r23

.def inner\_count\_L=R24

.def inner\_count\_H=R25

.def outer\_count\_L=R26 ; (XL)

.def outer\_count\_H=R27 ; (XH)

.org $100

reset:

ldi temp,LOW(RAMEND) ;arxikopoiisti stack pointer

out SPL,temp

ldi temp,HIGH(RAMEND)

out SPH,temp

ser temp

out DDRB,temp ;DDRB ws thyra eksodou

out PORTB,temp

clr temp

out DDRD,temp ;DDRD ws thyra eisodou

ch\_sw:

in temp,PIND

cpi temp, 0b11110000 ;if sw0-sw3=on

breq save\_code1

rjmp ch\_sw

save\_code1:

in temp,PIND

cpi temp, 0b11111111

breq save\_code1

;rcall DLY

;rcall check\_correct

mov code1,temp

save\_code2:

in temp,PIND

cpi temp, 0b11111111

breq save\_code2

;rcall DLY

;rcall check\_correct

mov code2,temp

save\_code3:

in temp,PIND

cpi temp, 0b11111111

breq save\_code3

;rcall DLY

;rcall check\_correct

mov code3,temp

save\_code4:

in temp,PIND

cpi temp, 0b11111111

breq save\_code4

;rcall DLY

;rcall check\_correct

mov code4,temp

rjmp ch\_sw7

check\_correct:

cpi temp,0b11101111

breq save\_code1

cpi temp,0b11011111

breq save\_code1

cpi temp,0b10111111

breq save\_code1

cpi temp,0b01111111

breq save\_code1

ret

DLY:

in temp2, PIND

cpi temp2,0b11111111

brne DLY

ret

ch\_sw7:

in temp,PIND

;rcall DLY

cpi temp,0b01111111

breq main

rjmp ch\_sw7

main:

in temp,PIND

;rcall DLY

cpi temp,0b1111110

breq main

rjmp led0\_on

led0\_on:

ldi temp, 0b1111110

out PORTB, temp

ldi counter,0

ldi flag,0

ch\_code1:

rcall inc\_counter

in temp,PIND

;rcall DLY

cp temp,code1

brne alarm

breq ch\_code2

cpi counter, 10

breq alarm

rjmp ch\_code1

ch\_code2:

rcall inc\_counter

in temp,PIND

;rcall DLY

cp temp,code2

brne alarm

breq ch\_code3

cpi counter, 10

breq alarm

rjmp ch\_code2

ch\_code3:

rcall inc\_counter

in temp,PIND

;rcall DLY

cp temp,code3

brne alarm

breq ch\_code4

cpi counter, 10

breq alarm

rjmp ch\_code3

ch\_code4:

rcall inc\_counter

in temp,PIND

;rcall DLY

cp temp,code4

brne alarm

breq finish

cpi counter, 10

breq alarm

rjmp ch\_code4

alarm:

inc flag

ldi counter, 0

cpi flag, 1

breq alarm\_1

cpi flag, 2

breq alarm\_2

alarm\_1:

ldi temp, 0b11111111

out PORTB, temp

rcall time\_delay\_1s

ldi temp, 0b11111110

out PORTB, temp

in temp, PIND

;rcall DLY

cpi temp,0b11111111

brne ch\_code1

rjmp alarm\_1

alarm\_2:

clr temp

out PORTB, temp

rcall time\_delay\_1s

ser temp

out PORTB, temp

rcall time\_delay\_1s

brne ch\_code1

rjmp alarm\_2

inc\_counter:

rcall time\_delay\_05s

inc counter

ret

time\_delay\_05s:

ldi outer\_count\_L, $78 ; 1 cycle

ldi outer\_count\_H, $02 ; 1 cycle

outer\_loop2:

ldi inner\_count\_L, $78 ; 1 cycle

ldi inner\_count\_H, $02 ; 1 cycle

inner\_loop2:

nop ; 1 cycle, possible use of more "nop" instructions for longer delays

sbiw inner\_count\_L, 1 ; 2 cycles

brne inner\_loop2 ; 2 cycles if true (Ni-1 times), 1 cycle if false

sbiw outer\_count\_L, 1 ; 2 cycles

brne outer\_loop2 ; 2 cycles if true (No-1 times), 1 cycle if false

ret ; 4 cycles (plus 3 cycles for rcall)

time\_delay\_1s:

ldi outer\_count\_L, $FE ; 1 cycle

ldi outer\_count\_H, $03 ; 1 cycle

outer\_loop3:

ldi inner\_count\_L, $FE ; 1 cycle

ldi inner\_count\_H, $03 ; 1 cycle

inner\_loop3:

nop ; 1 cycle, possible use of more "nop" instructions for longer delays

sbiw inner\_count\_L, 1 ; 2 cycles

brne inner\_loop3 ; 2 cycles if true (Ni-1 times), 1 cycle if false

sbiw outer\_count\_L, 1 ; 2 cycles

brne outer\_loop3 ; 2 cycles if true (No-1 times), 1 cycle if false

ret ; 4 cycles (c s 3 cycl

normal:

ldi temp, 0b11111100

out PORTB, temp

ch\_vio:

in temp,PIND

cpi temp, 0b11111101

;rcall DLY

breq led2on

in temp,PIND

cpi temp, 0b11111011

;rcall DLY

breq led3\_2on

in temp,PIND

cpi temp, 0b11100111

;rcall DLY

breq led4\_2on

in temp,PIND

cpi temp, 0b11011111

;rcall DLY

breq led5\_2on

in temp,PIND

cpi temp, 0b10111111

;rcall DLY

breq led6\_2on

in temp,PIND

cpi temp, 0b01111111

;rcall DLY

breq led7\_2on

rjmp ch\_vio

sw0\_normal:

in temp, PIND

cpi temp, 0b11111110

;rcall DLY

breq normal

rjmp sw0\_normal

led2on:

ldi temp, 0b11111011

out PORTB, temp

rjmp sw0\_normal

led3\_2on:

ldi temp, 0b11110011

out PORTB, temp

rjmp sw0\_normal

led4\_2on:

ldi temp, 0b11101011

out PORTB, temp

rjmp sw0\_normal

led5\_2on:

ldi temp, 0b11011011

out PORTB, temp

rjmp sw0\_normal

led6\_2on:

ldi temp, 0b10111011

out PORTB, temp

rjmp sw0\_normal

led7\_2on:

ldi temp, 0b01111011

out PORTB, temp

rjmp sw0\_normal

finish:

jmp end