ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ 2

Ομάδα 7
Χατζηθωμά Αντρέας 8026
Νήρας Δημήτρης 8057

<u>ΤΜΗΜΑ 1</u>

Αρχικά αποθηκεύσαμε στην μνήμη προγράμματος τους βαθμούς για τον καθένα μας από το 5ο εξάμηνο και έπειτα κάναμε definition τους καταχωρητές R17 και R25 ούτως ώστε να αποθηκεύσουμε το άθροισμα των βαθμών και τον μέσο όρο τους αντίστοιχα. Στη συνέχεια ορίσαμε δυο μακροεντολές, μία για την εμφάνιση των κωδικών-βαθμών όπως ορίζεται στην εκφώνηση, και μία για το άθροισμα των βαθμών. Έπειτα αρχικοποιούμε τον stack pointer και ορίζουμε σαν έξοδο των LEDs το PORTB. Εμφανίζουμε διαδοχικά τους κωδικούς-βαθμούς για 5 δευτερόλεπτα όπως αναφέρεται στην εκφώνηση, μεσολαβώντας ανάμεσα στην εμφάνιση τους 4 δευτερόλεπτα αναβοσβήμματος τους. Αφού εμφανιστούν οι βαθμοί και των δυο μας, εμφανίζουμε τους μέσους όρους αντίστοιχα για τον καθένα μας με ακρίβεια μισού βαθμού. Για τον υπολογισμό του μέσου όρου χρησιμοποιείται η μακροεντολή addNum για την πρόσθεση των βαθμών και η διαδικάσια division για την διαίρεση τους με το 6 (η μεθοδολογία της διαδικασίας της διαίρεσης έχει βρεθεί απο το διαδίκτυο). Οι διαδικασίες delay για 5, 4, 2, 0.5 δευτερόλεπτα καθυστέρηση αποτελούνται από επαναληπτικές διαδικασίες (for loops), και των οποίων η διάρκεια υπολογίστηκε όπως και στην προηγούμενη εργασία.

Το κομμάτι που δυσκολευτήκαμε περισσότερο στο τμήμα 1 αυτής της άσκησης ήταν κυρίως το κομμάτι της διαίρεσης με το 6, καθώς δεν μπορούσε να υλοποιηθεί με ολισθήσεις και χρειάστηκε η συγγραφή επιπλέον κώδικα, ηγ οποία βρέθηκε από το διαδίκτυο.

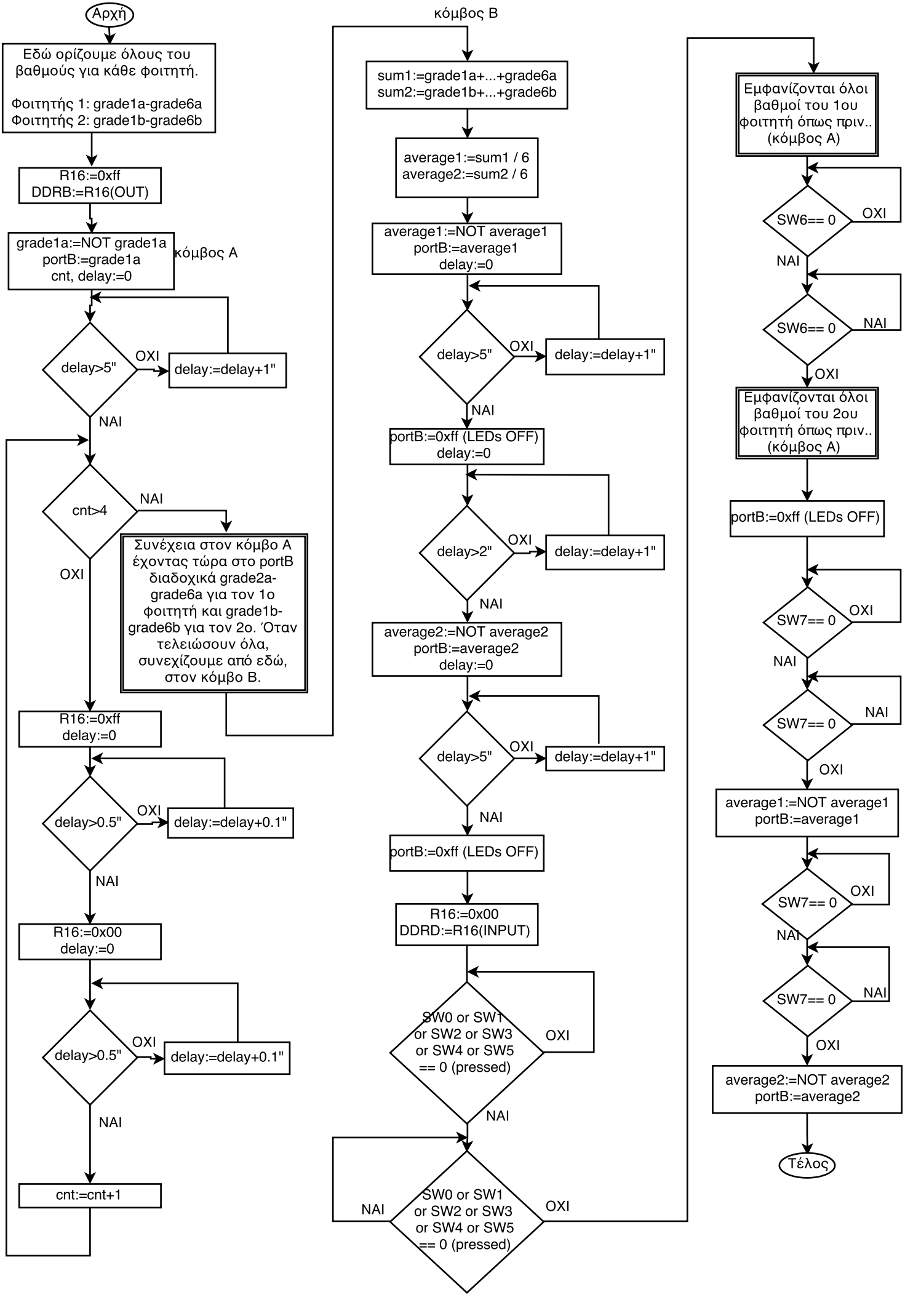
ΤΜΗΜΑ 2

Στο 2ο μέρος του προγράμματος ορίζουμε ως είσοδο το port D στο οποίο είναι συνδεδεμένοι οι διακόπτες, δίνοντας μηδενικά στο DDRD. Αφού το κάνουμε αυτό, σβήνουμε τα LEDs δίνοντας άσσους στα pins των LEDs (active low logic). Έπειτα, μέσω επαναληπτικών διαδικασιών (loops) ελέγχουμε αν κάποιος από τους διακόπτες SW1-SW5 έχει πατηθεί και ακολούθως πάλι μέσω επαναληπτικής διαδικασίας το πρόγραμμα περιμένει μέχρι να αφεθεί ο συγκεκριμένος διακόπτης οποίος πατήθηκε. Όταν γίνει αυτό (ενεργοποίηση – απενεργοποίηση διακόπτη), εμφανίζονται στα LEDs οι βαθμοί του 1ου φοιτητή όπως ακριβώς εμφανίζονται και στο τμήμα 1. Ακολούθως, με την ίδιο ακριβώς τρόπο με πριν, γίνεται έλεγχος για το αν ενεργοποιήθηκε-απενεργοποιήθηκε ο διακόπτης SW6 και ακολούθως εμφανίζονται οι βαθμοί του 2ου φοιτητή. Αφού γίνει και αυτό, όπως και πριν, ελέγχουμε για την ενεργοποίηση-απενεργοποίηση του διακόπτη SW7 και ακολούθως εμφανίζεται ο μέσος όρος του 1^{ου} φοιτητή όπως ακριβώς και στο τμήμα 1. Αφού εμφανιστεί, το πρόγραμμα περιμένει και πάλι την ενεργοποίηση-απενεργοποίηση του SW7 για την εμφάνιση του μέσου όρου του 2^{ου} φοιτητή. Οι διαδικασίες εμφάνισης των βαθμών στα LEDs, είναι οι ίδιες διαδικασίες που χρησιμοποιήθηκαν και στο τμήμα 1 του προγράμματος.
Τέλος, το πρόγραμμα εισέρχεται σε μια ατέρμονα επαναληπτική διαδικασία η οποία δηλώνει και το τέλος του προγράμματος.
Το τμήμα 2 δεν είχε ιδιαίτερες δυσκολίες καθώς οι διαδικασίες για την εμφάνιση των αποτελεσμάτων είχαν ήδη υλοποιηθεί στο τμήμα 1, όπως επίσης και όλες οι εντολές που χρησιμοποιήθηκαν, είχαν όλες χρησιμοποιηθεί ξανά και στην προηγούμενη εργαστηριακή άσκηση.

BREAKPOINTS-DEBUGGING

Για τον έλεγχο της σωστής λειτουργίας του προγράμματος επιλέξαμε να βάλουμε breakpoints στα σημεία της πρόσθεσης των βαθμών αλλά και στο σημείο του υπολογισμού του μέσου όρου (διαίρεση με το 6), ούτως ώστε να ελέγξουμε την ορθότητα των αποτελεσμάτων. Ακολούθως, για την εμφάνιση των αποτελεσμάτων στα LEDs, επιλέξαμε να μην βάλουμε breakpoints αλλά να χρησιμοποιήσουμε την εκτέλεση του προγράμματος βήμα-βήμα με την χρήση του κουμπιού F11.
Επίσης, κατά την εκτέλεση του προγράμματος χρησιμοποιήθηκαν μικρότερα delays για ευκολία κατά το debugging. Στο εργαστήριο χρησιμοποιήθηκαν τα σωστά delays με τους σωστούς χρόνους, οι οποίοι χρονομετρήθηκαν για τον έλεγχο ορθότητας τους.

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ



ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

```
;lab2
;Χατζηθωμά Αντρέας 8026
;Νήρας Δημήτρης 8057
;Στο τμήμα 1 αρχικοποιούμε τον stack pointer, αποθηκεύουμε τους βαθμούς μας στην μνήμη
;προγράμματος με την ζητούμενη μορφή και εμφανίζουμε τα δεδομένα στα LEDs διαδοχικά.
;Ακολούθως υπολογίζουμε τον μέσο όρο του καθενός μας  και εμφανίζουμε τα αποτελέσματα
;LEDs διαδοχικά.
;Στο τμήμα 2 εκτελείται η ίδια διαδικασία με το τμήμα 1 με την χρήση διακοπτών. Πιέζοντας ένα
;διακόπτη SW0-SW5 το πρόγραμμα μπαίνει σε διαδικασία αναμονής. Απελευθερώνοντας τον
;διακόπτη εμφανίζονται οι βαθμοί του 1ου φοιτητή. Για τους βαθμούς του 2ου φοιτητή πιέζουμε το ;SW6 και ακολούθως 2 φορές το SW7 για την εμφάνιση του 1ου μέσου όρου και του 2ου αντίστοιχα.

.include "m16def.inc"
.org 0x0
.cseg

;Andreas-8026
grade1a: .dw 0b0101000100001100 ;6
grade2a: .dw 0b0101001000010000 ;8
grade3a: .dw 0b0101001100001101 ;6.5
grade4a: .dw 0b0101010000010000 ;8
grade5a: .dw 0b0101010100010011 ;9.5
grade6a: .dw 0b0101011000001010 ;5

;Dimitris-8057
grade1b: .dw 0b0101000100001110 ;7
grade2b: .dw 0b0101001000010001 ;8.5
grade3b: .dw 0b0101001100001110 ;7
grade4b: .dw 0b0101010000010000 ;8
grade5b: .dw 0b0101010100010011 ;9.5
grade6b: .dw 0b0101011000100000 ;10

.def sum=r17
.def ans=r25

;function for adding two numbers
.macro addNum
        ldi ZL, low(@0<<1)
        ldi ZH, high(@0<<1)
        lpm r20, Z+       ;low

        sbrc r20, 5 ;skips next command if 5th bit is cleared (0)
        rjmp plus10
        rjmp done

;sets the register to the correct form of number 10 to correctly calculate the average later
plus10:
        andi r20, 0b11011111     ;logical AND to clear register
        ori r20, 0b00010100              ;logical OR to set the register to number 10(decimal)

;adds the 2 registers
done:
        add sum, r20
.endmacro


;function for led light up
```

```
.macro LEDS
        ldi ZL, low(@0<<1)
        ldi ZH, high(@0<<1)
        lpm r20, Z+ ;low
        lpm r21, Z      ;high

        ;Display Num for LEDs 7-4
        lsl r21
        lsl r21
        lsl r21
        lsl r21

        ;Display num for LEDs 3-0
        sbrc r20, 5
        rjmp grade10
        andi r20, 0b00011110
        lsr r20
        rjmp done

grade10:
        ldi r20, 0b00001010

done:
        or r20, r21
        com r20
        out PORTB, r20

        rcall delay5
        rcall blinking

.endmacro

;initialize stack pointer
sp_init:
ldi r16, low(RAMEND)
out spl, r16
ldi r16, high(RAMEND)
out sph, r16

main:
        ser r16
        out DDRB, r16  ;set the portB for output (LEDs)
        out PORTB, r16

        rcall led1              ;leds for student1
        rcall led2              ;leds for student2

        rcall sum1              ;average for student1
        rcall delay5

        ser r16
        out PORTB, r16
        rcall delay2

        rcall sum2              ;average for student2
        rcall delay5

        ser r16
        out PORTB, r16
        rjmp part2
```

```
led1:
        LEDs grade1a
        LEDs grade2a
        LEDs grade3a
        LEDs grade4a
        LEDs grade5a
        LEDs grade6a
        ret

led2:
        LEDs grade1b
        LEDs grade2b
        LEDs grade3b
        LEDs grade4b
        LEDs grade5b
        LEDs grade6b
        ret

sum1:
        clr sum

        addNum grade1a
        addNum grade2a
        addNum grade3a
        addNum grade4a
        addNum grade5a
        addNum grade6a

        rcall division      ;divide by 6

        ori ans, 0b01000000 ;logical OR to set the led6 (= student1)
        com ans                         ;NOT because of active low logic
        out PORTB, ans

        ret

sum2:
        clr sum

        addNum grade1b
        addNum grade2b
        addNum grade3b
        addNum grade4b
        addNum grade5b
        addNum grade6b

        rcall division      ;divide by 6

        ori ans, 0b10000000       ;logical OR to set the led7 (= student2)
        com ans                         ;NOT because of active low logic
        out PORTB, ans

        ret

;0.5 led blink for total time of 4 seconds
blinking:
        ldi r30, 4
b:
        ser r16
```

```
        out PORTB, r16
        rcall delay05

        clr r16
        out PORTB, r16
        rcall delay05

        dec r30
        brne b

        ret

;divide by 6 to calculate the average
division:
        ldi r22, 6          ;load divisor
        ldi r23, 9          ;load bit counter
        sub r24,r24   ;clear remainer
        mov ans, sum
loop:
        rol ans             ;shift left
        dec r23
        breq done
        rol r24
        sub r24, r22
        brcc skip
        add r24, r22
        clc
        rjmp loop
skip:
        sec
        rjmp loop
done:
        ret

;delay for 2 seconds
;255x255x40 loops for 2sec delay (cycle time = 0.25us)
delay2:
        ldi r17, 0xff
d2_3:
        ldi r18, 0xff
d2_2:
        ldi r19, 0x28
d2_1:
        dec r19
        brne d2_1
        dec r18
        brne d2_2
        dec r17
        brne d2_3
        ret

;delay for 5 seconds
;255x255x110 loops for 5sec delay (cycle time = 0.25us)
delay5:
        ldi r17, 0xff
d5_3:
        ldi r18, 0xff
d5_2:
        ldi r19, 0x6e
d5_1:
```

```
        dec r19
        brne d5_1
        dec r18
        brne d5_2
        dec r17
        brne d5_3
        ret

;delay for 0.5 seconds
;255x255x10 loops for 0.5sec delay (cycle time = 0.25us)
delay05:
        ldi r17, 0xff
d05_3:
        ldi r18, 0xff
d05_2:
        ldi r19, 0x0a
d05_1:
        dec r19
        brne d05_1
        dec r18
        brne d05_2
        dec r17
        brne d05_3
        ret



part2:
        clr r16
        out DDRD, r16  ;set portD for input (switches)

        ser r16
        out PIND, r16

;waits until a switch is pressed
unpressed:
        sbis PIND, 0x00 ;If (Port D, pin0 == 0) do
        rjmp press0

        sbis PIND, 0x01;If (Port D, pin1 == 0) do
        rjmp press1

        sbis PIND, 0x02;If (Port D, pin2 == 0) do
        rjmp press2

        sbis PIND, 0x03 ;If (Port D, pin3 == 0) do
        rjmp press3

        sbis PIND, 0x04 ;If (Port D, pin4 == 0) do
        rjmp press4

        sbis PIND, 0x05 ;If (Port D, pin5 == 0) do
        rjmp press5

        rjmp unpressed

;checks if a switch is still pressed
press0:
        sbic PIND, 0x00 ;Skip next instruction if (Port D, pin0 == 0)
        rjmp next
```

```
        rjmp press0
press1:
        sbic PIND, 0x01 ;Skip next instruction if (Port D, pin1 == 0)
        rjmp next
        rjmp press1
press2:
        sbic PIND, 0x02 ;Skip next instruction if (Port D, pin2 == 0)
        rjmp next
        rjmp press2
press3:
        sbic PIND, 0x03 ;Skip next instruction if (Port D, pin3 == 0)
        rjmp next
        rjmp press3
press4:
        sbic PIND, 0x04 ;Skip next instruction if (Port D, pin4 == 0)
        rjmp next
        rjmp press4
press5:
        sbic PIND, 0x05 ;Skip next instruction if (Port D, pin5 == 0)
        rjmp next
        rjmp press5


next:
        rcall led1

        ser r16
        out PORTB, r16

sw6_unpressed:
        sbic PIND, 0x06 ;Skip next instruction if (Port D, pin6 == 0)
        rjmp sw6_unpressed
sw6_pressed:
        sbis PIND, 0x06 ;If (Port D, pin6 == 0) do
        rjmp sw6_pressed

        rcall led2

        ser r16
        out PORTB, r16

        rcall press
        rcall sum1

        rcall press
        rcall sum2

;infinite loop
end_loop:
        rjmp end_loop

press:
sw7_unpressed:
        sbic PIND, 0x07 ;Skip next instruction if (Port D, pin7 == 0)
        rjmp sw7_unpressed
sw7_pressed:
        sbis PIND, 0x07 ;If (Port D, pin7 == 0) do
        rjmp sw7_pressed
        ret
```