ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ
Εργαστήριο Μικροϋπολογιστών

6η Εργαστηριακή Άσκηση
Ακ. έτος 2011-2012

Ομάδα C07:

Ελένη Ευαγγελάτου Α.Μ.: 03108050
Γρηγόρης Λύρας Α.Μ.: 03109687
Βασιλεία Φραγκιαδάκη Α.Μ.: 03108026

5 Φεβρουαρίου 2012

Άσκηση (i)

Στην άσκηση αυτή ζητείται στο χρονόμετρο που απεικονίζεται στην PORTA και υλοποιείται με ταχύτητα 100 msec ανά μέτρηση, όταν ενεργοποιείται εξωτερική διακοπή INT0, αν το PD0 είναι μηδέν, να απαριθμεί τις διακοπές και να εμφανίζει το πλήθος τους στην PORTB. Γι' αυτό το σκοπό ενεργοποιούμε με τις κατάλληλες εντολές τις διακοπές στο κυρίως πρόγραμμα και αρχικά βάζουμε το .org 0x002 που επιτρέπει την εξυπηρέτηση της διακοπής INT0 στον Atmega16.
Κυρίως κώδικας:

```
 ;
 ;part1.asm
 ;
 ; Created: 29/1/2012 11:50:22 ??
 ;  Author: Valia
 ;

 ;.include "m16def.inc"
 .def temp1 = r24
 .def temp2 = r25
 .def metrhths = r26
 .def check_PD0 = r22
 .def state = r27
 .def metr_diakopwn = r23

 .org 0x000
 rjmp main
.org 0x002
    rjmp ISR0
    reti

main:
    ldi temp1,high(RAMEND)
    out SPH,temp1          ; arxikopoihsh ths stoivas
    ldi temp1,low(RAMEND)
    out SPL,temp1

    ldi temp1,(1<<ISC01) | (1 << ISC00) ;energopoihsh ths diakophs INT0
    out MCUCR,temp1
    ldi temp1,(1 <<INT0)
    out GICR ,temp1
    sei
    clr temp1

    ser metrhths
    out DDRA,metrhths    ;PORTA exodos

    clr metrhths
    out DDRD,metrhths    ;PORTD eisodos

loop:
    out PORTA,metrhths

    ldi temp1,low(100)
    ldi temp2,high(100)  ;ka8ysterhsh 100 msec
    rcall wait_msec
    inc metrhths
    rjmp loop

ISR0:
    push metrhths
    in state,SREG    ;swzw thn katastash:D
    push state
    push check_PD0

    in check_PD0,PIND  ;elegxos PD0
    ror check_PD0
    brcs exit       ;an PRD0 = 0 metraei th diakoph

    ser metrhths
    out DDRB,metrhths    ;exodos h PORTB

```

```
63      inc metr_diakopwn    ;auxanoume t metrhth
64
65      out PORTB, metr_diakopwn        ;emfanish twn diakopwn
66      ldi temp1,low(1000)
67      ldi temp2,high(1000)
68      ;rcall wait_msec    ;ka8ysterhsh 1 sec
69
70  exit:
71      out SREG,state
72      pop check_PD0
73      pop state
74      pop metrhths
75      reti
76
77  ;routines pou pragmatopoioun ka8ysterhsh tosa msec, oso to periexomeno twn r25-r24
78  wait_msec:
79      push temp1
80      push temp2
81      ldi temp1, low(998)
82      ldi temp2, high(998)
83      rcall wait_usec
84      pop temp2
85      pop temp1
86      sbiw temp1, 1
87      brne wait_msec
88
89      ret
90
91  wait_usec:
92      sbiw temp1,1
93      nop
94      nop
95      nop
96      nop
97      brne wait_usec
98
99      ret
```

Άσκηση (ii)

Σε αυτό το μέρος πραγματοποιήσαμε ένα πρόγραμμα μέτρησης το οποίο διακόπτεται μόνο από την διακοπή INT1 (push button button PD3). Το πρόγραμμα μέτρησης εμφανίζεται στην σειρά led PA. Όταν έχουμε διακοπή πρέπει να διαβάζουμε από τα dip switches B και να μετράμε πόσα είναι ON. Το πόσα είναι ON φαίνεται στα leds PC. Μετά την διακοπή συνεχίζεται κανονικά η μέτρηση στα leds PA. Το πρόγραμμα μέτρησης είναι ακριβώς όπως το δεθέν της εκφώνησης. Αυτή την φορά απλά αναγνωρίζονται οι διακοπές INT1. Ο κώδικας εξυπηρέτησης της διακοπής βρίσκεται στην label ISR1. Το πόσα dip switches είναι ON υπολογίζεται ως εξής: διαβάζουμε από τον PINB, και κάνουμε shift δεξιά. Αν έχουμε carry, τότε αυξάνουμε κατά 1 το άθροισμα των switches που είναι on. Διαφορετικά ελέγχουμε το επόμενο ψηφίο. Αυτό γίνεται και για τα 8 bits. Στο τέλος, για ασφάλεια ώστε να έχουμε το αποτέλεσμα στα 4 least significant bits, κάνουμε μια bitwise λογική πράξη and με τον αριθμό 00001111.

Έπειτα δείχνουμε το αποτέλεσμα στα leds PC. Ο κώδικάς μας:
Κυρίως κώδικας:

```
1   ;
2   ;  part2.asm
3   ;
4   ;  Created: 31/1/2012 7:58:21 ??
5   ;  Author: Eleni
6   ;
7
8   .org 0x00
9       rjmp main
10  .org 0x04
11      rjmp ISR1
12      reti
13
14
15
16  main:
17      ldi r30,low(RAMEND)
18      out SPL, r30
19      ldi r30, high(RAMEND)
20      out SPH, r30
```

```
21
22      ;programma metrhshs opws dinetai
23      ser r26
24      out DDRA, r26
25      clr r26
26  ;=====gia energopoihsh ths INT1 =======
27
28      ldi r24, (1<< ISC11)|(1 << ISC10) ;na prokaleitai me shma 8etikhs akmhs
29      out MCUCR, r24
30      ldi r24, (1<< INT1)
31      out GICR, r24
32      sei
33
34
35  loop:
36      out PORTA, r26
37      ldi r24, low(100)
38      ldi r25, high(100)
39
40      rcall wait_msec
41      inc r26
42      rjmp loop
43
44  ;====routina eksuphrethshs diakophs=====
45
46  ISR1:
47      push r26 ;swzei to periexomeno tou r26
48      in r26, SREG ;kai SREG
49      push r26
50
51      clr r27 ;clear ton r27 giati ekei 8a mazepsw to sum mou
52
53      clr r25
54      out DDRB, r25 ;den eimai sigourh gia ton r25
55      in r29, PINB
56
57      mov r28, r29 ;mhpws tuxon xreiastei
58
59  bit0:
60      lsr r28
61      brcs plus_one_0 ;an exw carry dhlash asso tote pros8etw ena sto sum dhladh ston register r27
62
63  bit1:
64      lsr r28
65      brcs plus_one_1
66  bit2:
67      lsr r28
68      brcs plus_one_2
69  bit3:
70      lsr r28
71      brcs plus_one_3
72  bit4:
73      lsr r28
74      brcs plus_one_4
75  bit5:
76      lsr r28
77      brcs plus_one_5
78  bit6:
79      lsr r28
80      brcs plus_one_6
81  bit7:
82      lsr r28
83      brcs plus_one_7
84      rjmp go_on
85
86  ;auksanei kata 1
87  plus_one_0:
88      inc r27
89      rjmp bit1
90
91  plus_one_1:
92      inc r27
93      rjmp bit2
94
95  plus_one_2:
```

```
    inc r27
    rjmp bit3

plus_one_3:
    inc r27
    rjmp bit4

plus_one_4:
    inc r27
    rjmp bit4

plus_one_5:
    inc r27
    rjmp bit6

plus_one_6:
    inc r27
    rjmp bit7

plus_one_7:
    inc r27



;telos plus_one

;kanw AND me 0000 1111
;set bit mask ston r16 gia na exw ari8mo mono sta 4 lsb, gia asfaleia , mporei k na mhn xreiazetai
go_on:
    ldi r16, 0x0F
    and r27, r16 ;bitwise and. To apotelesma apo8hkeuetai ston r27

    ser r29
    out PORTC, r29
    out DDRC, r27 ;ta deixnw sta leds

    reti

;=========wait_msec===============
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
```

Άσκηση (iii)

Σ' αυτή την άσκηση ζητείται να ανάβει το led PA1 για 3 sec με τη βοήθεια χρονιστή, όταν ενεργοποιηθεί διακοπή INT0 ή είναι on το PA0. Γι' αυτό το σκοπό "σετάρουμε" το χρονιστή όταν ενεργοποιείται διακοπή INT0 ή το PA0 είναι ένα και όταν προκληθεί υπερχείλιση ενεργοποιείται η αντίστοιχη διακοπή που σβήνει το led. Αν ωστόσο ενεργοποιηθεί και άλλη διακοπή ή το PA0, σετάρεται και πάλι ο χρονιστής στην αρχική του τιμή ώστε να ανανωθεί η καθυστέρηση. Να σημειώσουμε ότι έχουμε βάλει τις απαράιτητες εντολές .org στην αρχή του προγράμματος ώστε να είναι δυνατή η εξυπηρέτηση των διακοπών.
Κυρίως κώδικας:

```
.org 0x000
    rjmp main
```

```
.org 0x002
    rjmp ISR0
.org 0x010
    rjmp TIM1_OVF
    reti

main:
reset:
    ldi r24,high(RAMEND)
    out SPH,r24
    ldi r24,low(RAMEND)
    out SPL,r24
    ldi r24,2
    ;#set A1 for output
    ;#and A0 for input
    out DDRA,r24
    ;activate timer1
    ;2 is TOIE1
    ;avrStudio doesn't know about it
    ldi r24,(1 << 2)
    out TIMSK,r24
    ldi r24,(1 << CS12)|(0<<CS11)|(1<<CS10)
    out TCCR1B,r24
    ;activate INT0
    ldi r24,(1 << ISC01) | (1 << ISC00)
    out MCUCR,r24
    ldi r24,(1 << INT0)
    out GICR,r24
    sei

fun:
    ;busy wait
    ;for interrupt
    in r26,PINA
    ror r26
    brcs ON
    rjmp fun
ON:
    rcall set_timer
    rcall led_on
    rjmp fun



ISR0:
    push r26 ;swzei to periexomeno tou r26
    in r26, SREG ;kai SREG
    push r26
    rcall debounce
    rcall set_timer
    rcall led_on
    out SREG,r26
    pop r26
    sei
    reti

TIM1_OVF:
    rcall led_off
    reti

debounce:
    push r26
    push r25
    push r24
deb_loop:
    ldi r26,(1<<INTF0)
    out GIFR,r26
    in  r26,GIFR
    rol r26
    rol r26
    ldi r24,0x05
    ldi r25,0x00
    rcall wait_msec
    brcs deb_loop
    pop r24
```

```
78      pop r25
79      pop r26
80      ret
81
82  set_timer:
83      push r24
84      ldi r24,0xa4
85      sts TCNT1H,r24
86      ldi r24,0x72
87      sts TCNT1L,r24
88      pop r24
89      ret
90
91  led_on:
92      push r24
93      ldi r24,2
94      out PORTA,r24
95      pop r24
96      ret
97
98  led_off:
99      push r24
100     ldi r24,0
101     out PORTA,r24
102     pop r24
103     ret
104
105 wait_usec:
106     sbiw r24,1
107     nop
108     brne wait_usec
109     ret
110
111 wait_msec:
112     push r24
113     push r25
114     ldi r24,low(998)
115     ldi r25,high(998)
116     rcall wait_usec
117
118     pop r25
119     pop r24
120     sbiw r24,1
121     brne wait_msec
122     ret
```