Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Εργαστήριο Μικροϋπολογιστών 2021-2022

2η Εργασία (AVR)

Κωνσταντίνος Σιδέρης, Ομάδα 48

Α.Μ.: 03118134

Άσκηση 1

Ακολουθεί ο κώδικας της άσκησης 1 (assembly) με σχόλια:

```
            .DEF i = r15
            .DEF a = r16
            .DEF b = r17
            .DEF c = r18
            .DEF d = r19
            .DEF na = r20
            .DEF nb = r21
            .DEF o1 = r22
            .DEF o2 = r23
            .DEF o3 = r24
            .DEF o4 = r25

            clr r31
            out DDRC, r31      ;Αρχικοποίηση PORTC ως input
            ser r31
            out DDRB, r31      ;Αρχικοποίηση PORTB ως output

start:      in i, PINC         ;Λήψη εισόδου από τη θύρα PINC

            ldi r30, 0x01
            mov a, i
            and a, r30         ;Απομόνωση του A και αποθήκευση στο register a
            ror i              ;Περιστροφή ώστε το B να έρθει στην θέση LSB
            mov b, i
            and b, r30         ;Απομόνωση του B και αποθήκευση στο register b
            ror i              ;Περιστροφή ώστε το C να έρθει στην θέση LSB
            mov c, i
            and c, r30         ;Απομόνωση του C και αποθήκευση στο register c
            ror i              ;Περιστροφή ώστε το D να έρθει στην θέση LSB
            mov d, i
            and d, r30         ;Απομόνωση του D και αποθήκευση στο register d
            mov na, a
            com na             ;Συμπλήρωμα ως προς 1 του A και αποθήκευση στο
register na
            mov nb, b
            com nb             ;Συμπλήρωμα ως προς 1 του B και αποθήκευση στο
register nb

            mov o1, na
            and o1, b          ;Δημιουργία και αποθήκευση της λογικής πράξης (A'B)
στο register o1
            mov o2, nb
            and o2, c
            and o2, d          ;Δημιουργία και αποθήκευση της λογικής πράξης
(B'CD) στο register o2
```

```
            or o1, o2          ;Δημιουργία και αποθήκευση της λογικής πράξης (A'B
+ B'CD) στο register o1
            com o1             ;Συμπλήρωμα ως προς 1 στο register o1 και
δημιουργία της λογικής πράξης F0=(A'B + B'CD)'
            andi o1, 0x01      ;Απομόνωση του αποτελέσματος F0 στο register o1

            mov o3, a
            and o3, c          ;Δημιουργία και αποθήκευση της λογικής πράξης (AC)
στο register o3
            mov o4, b
            or o4, d           ;Δημιουργία και αποθήκευση της λογικής πράξης (B+D)
στο register o4
            and o3, o4         ;Δημιουργία και αποθήκευση της λογικής πράξης
F1=(AC)(B+D) στο register o3

            rol o3             ;Περιστροφή του αποτελέσματος F0 ώστε να έρθει στην
θέση του 2ου LSB και απομόνωση
            andi o3, 0x02
            or o1, o3          ;Συνδυασμός των F0, F1 στο register o1

            out PORTB, o1      ;Εμφάνιση αποτελέσματος στην θύρα PORTB
            rjmp start
```

Ακολουθεί ο κώδικας της άσκησης 1 (C) με σχόλια:

```
#include <avr/io.h>

char i, a, b, c, d, o;

int main(void)
{
   DDRB=0xFF;                        //Αρχικοποίηση PORTB ως output
   DDRC=0x00;                        //Αρχικοποίηση PORTC ως input
      while(1)
      {
            i = PINC & 0x0F;  //Λήψη εισόδου και απομόνωση των 4ων LSB
            a = i & 0x01;             //Απομόνωση και αποθήκευση του A στον
register a
            b = i & 0x02;
            b = b >> 1;               //Απομόνωση και αποθήκευση του B στον
register b
            c = i & 0x04;
            c = c >> 2;               //Απομόνωση και αποθήκευση του C στον
register c
            d = i & 0x08;
            d = d >> 3;               //Απομόνωση και αποθήκευση του D στον
```

```
register d
            o = (a&c)&(b|d);  //Αποθήκευση του F1 στον register o και
περιστροφή κατά μία θέση ώστε να έρθει στην θέση του 2ου LSB
            o = o << 1;
            PORTB = o | ((~((((~b)&c&d)|((~a)&b))))&0x01);  //Συνδυασμός
των F0, F1 και εμφάνιση αποτελέσματος στην θύρα PORTB
      }
      return 0;
}
```

Άσκηση 2

Κάνουμε την παραδοχή ότι μετράμε τον αριθμό των διακοπών ανεξάρτητα από την στάθμη των PA6, PA7, δηλαδή τα δύο MSB της θύρας A ελέγχουν μόνο το εάν θα εμφανιστεί το άθροισμα ή όχι στην έξοδο. Ακολουθεί ο κώδικας της άσκησης 2 με σχόλια:

```
            .DEF icnt = r15
            .DEF a = r16
            .DEF cnt = r17

            .org 0x0
            rjmp reset
            .org 0x3
            rjmp ISR1

ISR1:       inc icnt          ;Μέτρηση διακοπών και αποθήκευση στον icnt
            in a, PINA
            andi a, 0xC0      ;Λήψη εισόδου και απομόνωση των PA6, PA7
            cpi a, 0xC0       ;Αν τα PA6, PA7 είναι ON τότε εμφανίζουμε τον
αριθμό των διακοπών στην θύρα PORTB
            brne exit         ;Αλλιώς βγαίνουμε από την ρουτίνα διακοπής χωρίς
να κάνουμε κάτι
            out PORTB, icnt
exit:       clr r18
            out PORTB, r18
            reti

reset:      ldi   r18, LOW(RAMEND)  ;Αρχικοποίηση stack pointer
            out   spl, r18
            ldi   r18, HIGH(RAMEND)
            out   sph, r18

            ldi r18, 0x0C
            out MCUCR, r18    ;Ενεργοποίηση διακοπής INT1 με ανερχόμενη ακμή
            ldi r18, 0x80
            out GICR, r18     ;Επίτρεψη διακοπής INT1
```

```
            sei                 ;Ενεργοποίηση διακοπών

            clr r18
            out DDRA, r18       ;Αρχικοποίηση PORTA ως input
            ser r18
            out DDRB, r18       ;Αρχικοποίηση PORTB ως output
            out DDRC, r18       ;Αρχικοποίηση PORTC ως output

            ldi cnt, 0x00       ;Αρχικοποίηση μετρητή cnt
start:      out PORTC, cnt      ;Εμφάνιση μετρητή στην θύρα εξόδου PORTC
            inc cnt             ;Αύξηση του μετρητή και επανάληψη
            rjmp start
```

Άσκηση 3

Ακολουθεί ο κώδικας της άσκησης 3 με σχόλια:

```
#include <avr/io.h>
#include <avr/interrupt.h>

int i;
char out, pinb, pa2, cnt, y, w;

ISR(INT0_vect)                  //Ρουτίνα εξυπηρέτησης διακοπής INT0
{
      pa2 = PINA & 0x04;        //Λήψη εισόδου Α και απομόνωση του PA2
      pinb = PINB;              //Λήψη εισόδου από την θύρα PINB
      cnt = 0;

      for(i = 0; i < 8; i++)    //Μέτρηση αναμένων led της PINB και αποθήκευση
αριθμού στον cnt
      {
            y = pinb & 0x01;    //Περιστροφή εισόδου 8 φορές και έλεγχος του LSB
κάθε φορά
            pinb = pinb >> 1;
            if (y == 0x01)      //Αν το LSB είναι 1 αυξάνουμε τον καταχωρητή cnt
                  cnt = cnt + 1;
      }

      if (pa2 == 0x04)
      {
            PORTC = cnt;        //Αν το PA2 είναι ON εμφανίζουμε τον αριθμό
αναμένων led σε δυαδική μορφή
      }
      else                      //Αν το PA2 είναι OFF ανάβουμε ίσο αριθμό led
```

```
ξεκινώντας από το LSB
	{
		out = 0x00;
		for (i = 0; i < cnt; i++)
		{
			out = out << 1;   //Προσθέτουμε αναμμένο led στην έξοδο
(λογικό 1) μέχρι να έχουμε όσα led έχουμε μετρήσει στο cnt
			out = out | 0x01;
		}
		PORTC = out;			//Εμφάνιση εξόδου στην θύρα PORTC
	}

}

int main(void)
{
	DDRC = 0xFF;		// Αρχικοποίηση PORTB ως output
	DDRB = 0x00;		// Αρχικοποίηση PORTC ως input
	DDRA = 0x00;		// Αρχικοποίηση PORTA ως input
	GICR = 0x40;		//Επίτρεψη διακοπής INT0
	MCUCR = 0x03;		//Ενεργοποίηση διακοπής INT0 με ανερχόμενη ακμή
	sei();			//Ενεργοποίηση διακοπών

	while (1)   //Infinite loop που περιμένει την ενεργοποίηση διακοπών
    {
		w = w + 1;  //Εντολή χωρίς λειτουργία για την σωστή βηματική
εκτέλεση το προγράμματος
	}
}
```