**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**

**ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**



**Μάθημα: Εργαστήριο Μικροϋπολογιστών**

**Αναφορά 6ης Σειράς Εργαστηριακών Ασκήσεων**

**7ο Εξάμηνο**

|  |  |
| --- | --- |
| **Ομάδα Δ15** | |
| **Ονοματεπώνυμο** | **Αριθμός Μυτρώου** |
| **Ιωάννης Μύτης** | **031 13133** |
| **Νικόλαος Μαρίτσας** | **031 14934** |
| **Μιχάλης Παπαδόπουλλος** | **031 14702** |

* **Άσκηση 7.1(assembly)**

.include "m16def.inc"

.org 0x00

.CSEG

Message1:

.db "Hello World!" , '\0' ; Write a string in ram

ldi ZL, low(Message1<<1) ; Initialize Z pointer in order to point to

ldi ZH, high(Message1<<1) ; the first element of the string

; which is basically an array of characters

set\_stack: ; Initialize Stack

ldi r24, low(RAMEND)

out SPL, r24

ldi r24, high(RAMEND)

out SPH, r24

print\_strint\_to\_uart:

rcall usart\_init ; Initialize usart

loop\_:

lpm r24, Z+ ; Load the context of where Z points to r24

cpi r24, '\0' ; If you reach the end of string go to next\_ branch

breq next\_

rcall usart\_transmit ; Transmit to usart the context of r24

push r24 ; Save the value of r24 on stack

ldi r24,low(100) ; Wait 100 msec for the results

ldi r25,high(100) ; to be presented more clearly

rcall wait\_msec

pop r24 ; Restore the value of r24

rjmp loop\_

next\_:

ldi r24 , '\n' ; transmit the new\_line character to usart

rcall usart\_transmit

ret

; ------------------------ ;

; ------- ROUTINES ------- ;

; ------------------------ ;

; Routine: usart\_init

; Description:

; This routine initializes the

; usart as shown below.

; ------- INITIALIZATIONS -------

;

; Baud rate: 9600 (Fck = 8MHz)

; Asynchronous mode

; Transmitter on

; Reciever on

; Communication parameters: 8 Data ,1 Stop , no Parity

; --------------------------------

; parameters: None.

; return value: None.

; registers affected: r24

; routines called: None

usart\_init:

clr r24 ; initialize UCSRA to zero

out UCSRA ,r24

ldi r24 ,(1<<RXEN) | (1<<TXEN) ; activate transmitter/receiver

out UCSRB ,r24

ldi r24 ,0 ; baud rate = 9600

out UBRRH ,r24

ldi r24 ,51

out UBRRL ,r24

ldi r24 ,(1 << URSEL) | (3 << UCSZ0) ; 8-bit character size,

out UCSRC ,r24 ; 1 stop bit

ret

; Routine: usart\_transmit

; Description:

; This routine sends a byte of data

; using usart.

; parameters:

; r24: the byte to be transmitted

; must be stored here.

; return value: None.

; registers affected: r24

; routines called: None.

usart\_transmit:

sbis UCSRA ,UDRE ; check if usart is ready to transmit

rjmp usart\_transmit ; if no check again, else transmit

out UDR ,r24 ; content of r24

ret

; Routine: usart\_receive

; Description:

; This routine receives a byte of data

; from usart.

; parameters: None.

; return value: the received byte is

; returned in r24.

; registers affected: r24

; routines called: None.

wait\_usec:

sbiw r24 ,1 ; 2 cycles (0.250 micro sec)

nop ; 1 cycles (0.125 micro sec)

nop ; 1 cycles (0.125 micro sec)

nop ; 1 cycles (0.125 micro sec)

nop ; 1 cycles (0.125 micro sec)

brne wait\_usec ; 1 or 2 cycles (0.125 or 0.250 micro sec)

ret ; 4 cycles (0.500 micro sec)

wait\_msec:

push r24 ; 2 cycles (0.250 micro sec)

push r25 ; 2 cycles

ldi r24 , 0xe6 ; load register r25:r24 with 998 (1 cycles - 0.125 micro sec)

ldi r25 , 0x03 ; 1 cycles (0.125 micro sec)

rcall wait\_usec ; 3 cycles (0.375 micro sec), total delay 998.375 micro sec

pop r25 ; 2 cycles (0.250 micro sec)

pop r24 ; 2 cycles

sbiw r24 , 1 ; 2 cycles

brne wait\_msec ; 1 or 2 cycles (0.125 or 0.250 micro sec)

ret

* **Άσκηση 7.1(c)**

#include <avr/io.h>

#include <stdio.h>

//Declaration of functions

void USART\_init(void);

unsigned char USART\_receive(void);

void USART\_transmit(char);

void USART\_putstring(char\*);

int main(void){

//Declaration of variables

char check,output;

char msg1[] = "InvalidNumber\n" ;

char msg2[] = "Read" ;

char msg3[] = "\n" ;

DDRC = 0xFF; // set PORTC as output

USART\_init(); // initialize usart

while (1){

check = USART\_receive(); // receive a byte from usart

if ( (check < '0') || (check > '8') ) // if the byte is not a number { // between 0 and 8

USART\_putstring(msg1); // print InvalidNumber, change line

continue; // and go through while loop from the top

}

USART\_putstring(msg2); // else print ReadX ,

USART\_transmit(check); // which X is the number we received

USART\_putstring(msg3); // and change line

output = check - '0'; // transform ascii to value(HEX)

output = (1 << output)>>1; // convert the value in order to

PORTC = output; // light the led suited to input number

}

return 0;

}

void USART\_init(void){

UCSRA = 0; // initialize UCSRA to zero

UCSRB = (1<<RXEN) | (1<<TXEN); // activate transmitter/receiver

UBRRH = 0; // baud rate = 9600

UBRRL = 51;

UCSRC = (1 << URSEL) | (3 << UCSZ0); // 8-bit character size, 1 stop bit

}

unsigned char USART\_receive(void){

// check if usart received a byte.

while((UCSRA & 0x80) == 0){} // if no check again

return UDR; // else return the received byte

}

void USART\_transmit(char input){

// check if usart is ready to transmit.

while((UCSRA & 0x20) == 0){} // if no check again

UDR = input; // else transmit the input byte to usart

}

void USART\_putstring(char\* StringPtr){ // this function is used to print

// a string to usart

while(\*StringPtr != '\0'){ // if you haven't reached the end of string

USART\_transmit(\*StringPtr); // transit the current character of the string

StringPtr++; // and then point to the next one

}

}

* **Άσκηση 7.2(assembly)**
* **Άσκηση 7.2(c)**

/\*

; Author: Michalis Papadopoullos

; Exercise: 7.2 (ii)

\*/

#define F\_CPU 8000000UL

#include <avr/io.h>

#include <util/delay.h>

#define USART\_BAUDRATE 9600

#define BAUD\_PRESCALE (((F\_CPU / (USART\_BAUDRATE \* 16UL))) - 1)

void USART\_init(void)

{

    UCSRA = 0;

    UCSRB = (1<<RXEN) | (1<<TXEN);

    UBRRH = 0;

    UBRRL = 51;

    UCSRC = (1 << URSEL) | (3 << UCSZ0);

}

unsigned char USART\_receive(void)

{

    while((UCSRA & 0x80) == 0 ){}

    return UDR;

}

void USART\_transmit(char input)

{

    while((UCSRA & 0x20) == 0){}

    UDR = input;

}

void ADC\_init ()

{

    ADMUX = (1 << REFS0);

    ADCSRA = (1 << ADEN) | (1 << ADPS0) | (1 << ADPS1) | (1 << ADPS2);

}

int main (void)

{

    /\*

     ADCSRA [ADEN|ADSC|ADATE|ADIF|ADIE|\*|\*|\*]

    \*/

    // Left Adjust Result: No

    // ADLAR = 0;

    // Initialize USART

    USART\_init();

    // Initialize ADC

    ADC\_init();

    while (1) {

// To start ADC conversion ADSC=1

        ADCSRA = ADCSRA | 0x40;

        while ( !(ADCSRA & 0x40) ) { /\* WAIT UNTIL CONVERSION IS DONE \*/ }

        \_delay\_ms(100);

        int res = ADC;

        USART\_transmit ( (res \* 5) / 1024 + '0' );

        int dig = (res % 10) + '0';

        USART\_transmit ( ',' );

        USART\_transmit ( ( (res \* 50) / 1024) % 10 + '0' );

        USART\_transmit('\n');

        \_delay\_ms(100);

}

}