Σχολή Ηλεκτρολόγων Μηχανικών
και
Μηχανικών Υπολογιστών

Συστήματα Μικροϋπολογιστών [Ροή Υ]

Εργαστηριακή Αναφορά

Αμπατζή Ναυσικά <031 17 198>

Θεμελής Γιώργος <031 17 131>

Δήμος Δημήτρης <031 17 165>

6ο Εξάμηνο

Μάιος 2020

Ζήτημα 2.1

```
.include "m16def.inc"

stack:      ldi r24 , low(RAMEND)   ; initialize stack pointer
            out SPL , r24
            ldi r24 , high(RAMEND)
            out SPH , r24

IO_set:  ser r24                    ; initialize PORTB
            out DDRB, r24           ; for output
            clr r24                 ; initialize PORTC
            out DDRC, r24           ; for input

main:    ldi r26, 01                ; initialize r26
            rcall left
            nop
            rcall right
            rjmp main

left:    in r24, PINC               ; check input
            andi r24, 04            ; keep bit 2 == PC2
            cpi r24, 04             ; repeat till it's not 1
            brcc left
            out PORTB, r26
            cpi r26, 80
            brcc endl
            lsl r26
            rjmp left
endl:    ret

right:   in r24, PINC               ; check input
            andi r24, 04            ; keep bit 2 == PC2
            cpi r24, 04             ; repeat till it's not 1
            brcc right
            out PORTB, r26
            cpi r26, 01
            breq endr
            lsr r26
            rjmp right
endr:    ret
```

Ζήτημα 2.2

```
.include "m16def.inc"

.DEF A = r16
.DEF B = r17
.DEF C = r18
.DEF D = r19
.DEF F = r20
.DEF T = r21

stack:    ldi r24 , low(RAMEND)
          out SPL , r24
          ldi r24 , high(RAMEND)
          out SPH , r24

IO_set:   ser r24          ; initialize PORTA
          out DDRA, r24    ; for output
          clr r24          ; initialize PORTC
          out DDRB, r24    ; for input

main:     clr F                  ; ready F
          in T, PINB             ; T <-- input

          mov A, T               ; LSB(A) = A
          lsr T
          mov B, T               ; LSB(B) = B
          lsr T
          mov C, T               ; LSB(C) = C
          lsr T
          mov D, T               ; LSB(D) = D

          mov T, B               ; save B in T

          mov F, B
          com F                  ; LSB(F) = B'
          and F, A               ; LSB(F) = AB'
          com C                  ; LSB(C) = C'
          and B, C               ; LSB(B) = BC'
          and B, D               ; LSB(B) = BC'D
          or F, B                ; LSB(F) = (AB' + BC'D)
          com F                  ; LSB(F) = (AB' + BC'D)'  <---

          com C                  ; restore C: LSB(C) = C
          mov B, T               ; restore B: LSB(B) = B
          or A, C                ; LSB(A) = A + C
          or B, D                ; LSB(B) = B + D
          and A, B               ; LSB(A) = (A + C)(B + D) <---
          lsl A                  ; A1 = (A + C)(B + D)

          andi A, 2              ; A = F1
          andi F, 1              ; F = F0
```

```
                or F, A                 ; F = F + A = OUTPUT
                out PORTA, F
                rjmp main
```

Ζήτημα 2.3

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

char x = 1;

int main(void) {
        DDRA = 0x00;    // define input
        DDRB = 0xFF;    // define output
        PORTB = x;      // set output to x

   while (1) {
                if(PINA == 1)           // if 1st LSB is pressed
                        if (x == 1) x = 128;
                        else x = x >> 1;  // right slide
                else if(PINA == 2)      // if 2nd LSB is pressed
                        if (x == 128) x = 1;
                        else x = x << 1;  // left slide
                else if(PINA == 4)      // if 3rd LSB is pressed
                        x = 1;          // set output to 1st LSB
                else if(PINA == 8)      // if 4th LSB is pressed
                        x = 128;        // set output to MSB
                while(PINA != 0);       // wait till button is unpressed again, to apply output changes
                PORTB = x;              // show output
   }
}
```