Linnéuniversitetet
Kalmar Växjö

Linnaeus University

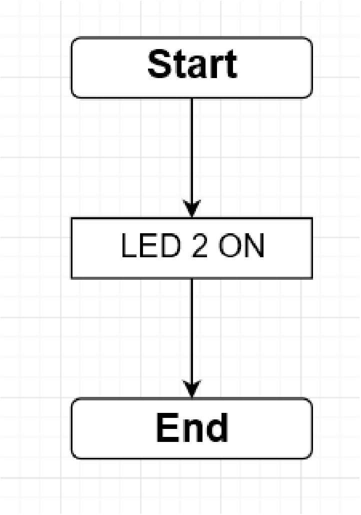
1DT301 - Computer Technology
Laboration 1

Students:
Alexander Risteski - ar222yu@student.lnu.se
Dimitrios Argyriou - da222kf@student.lnu.se

Task 1

Write a program in Assembly language to light LED 2. You can use any of the four ports, but start with PORTB.
The program should be very short! How many instructions is minimum number?

Starting with a simple task to light up a specific LED on the board by loading the register 16 with the specific bit for LED2 and assigning port B as output. The flow chart and the code are shown below.

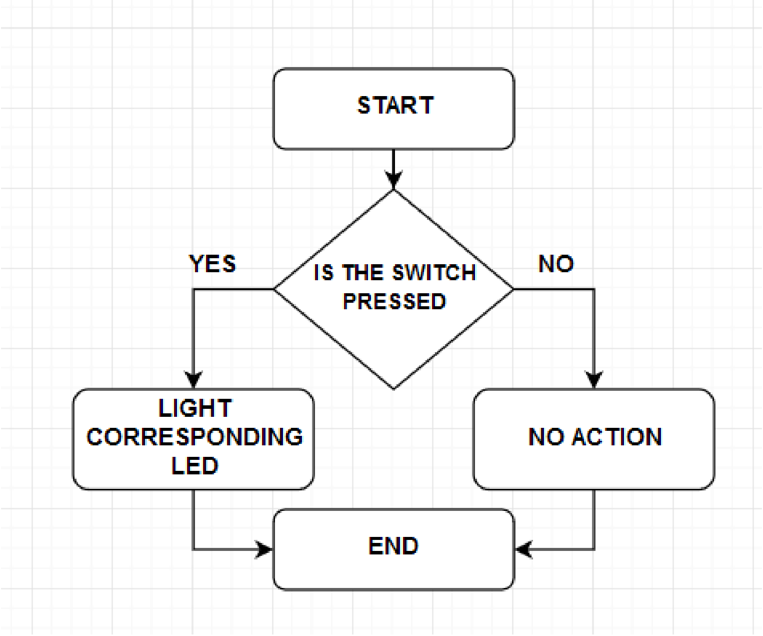


```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;               1DT301, Computer Technology I
;               Date: 08/09/2017
;               Authors:
;                                               Alexander Risteski
;                                               Dimitrios Argyriou
;
;               Hardware: STK600, CPU ATmega2560
;
;               Function: This program lights LED 2
.
;               Output ports: On-board LEDs connected to PORTB.
;
;               Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"
 .DEF mr = R16                                  ;defining register r16 as mr.

ldi r16,0b00000100                              ;load 1111 1011 to register r16
out DDRB, r16                                   ;write 1111 1011 to port B output resister
```

Task 2

Write a program in Assembly language to read the switches and light the corresponding LED. Example: When you press SW5, LED5 so should light. Make an initialization part of the program and after that an infinite loop.

Second task, it required a user input, pressing specific switch and LED begin activated to corresponding switch. As shown in the flow chart below we decided to use CPI instruction to compare input values then activate (in case of input value) corresponding LED and deactivating upon release.



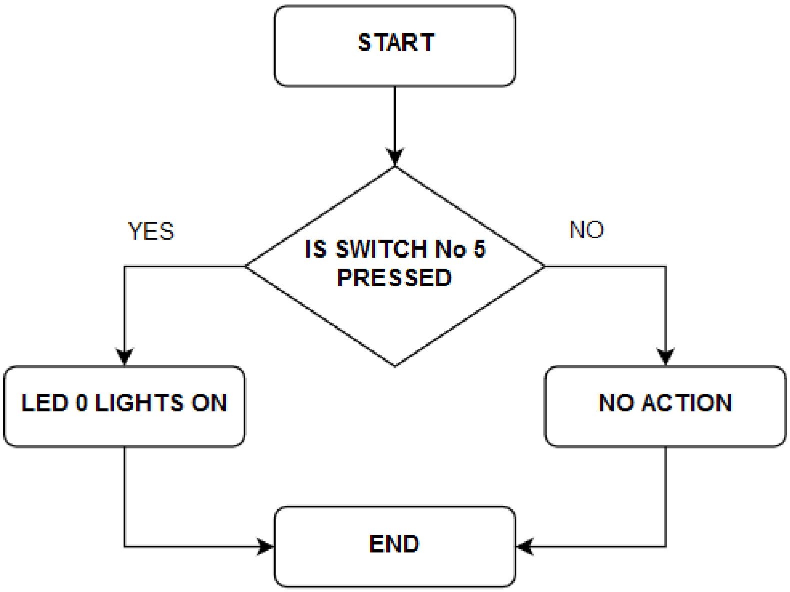
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;               1DT301, Computer Technology I
;               Date: 08/09/2017
;               Authors:
;                                       Alexander Risteski
;                                       Dimitrios Argyriou
;
;               Hardware: STK600, CPU ATmega2560
;
;               Function: Reads the switches and lights the corresponding LED.
;
;               Input ports: On-board switches connected to PORTA.
;               Output ports: On-board LEDs connected to PORTB.
;
;               Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

 .include "m2560def.inc"
 my_loop:
 .DEF mr = r16                          ;defining register r16 as "mr"

                ldi mr, 0xff            ;Set Data Direction Registers
                out DDRB,mr             ;port B as outputs
```

```
ldi mr,0x00            ;Load r16 with hex 00.
out DDRA,mr            ;port A as outputs.

in mr, PINA            ;read portA as input.
out PORTB,mr           ;output r16 to port B.

rjmp my_loop           ;jump to my_loop.
```

Task 3

Write a program in Assembly language to read the switches and light LED0 when you press SW5. For all other switches, there should be no activity.

Third task requirement was pressing switch 5 and activating LED0. The approach was similar to the one of the second task. CPI instruction was used to compare the input value to SW05 and branch to subroutine that activates the LED0 if the input value is equal to SW05, and branch to the beginning if not equal (which gives an instruction to jump back to the beginning of the loop, “my_loop”).



```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;              1DT301, Computer Technology I
;              Date: 08/09/2017
;              Authors:
;                                  Alexander Risteski
;                                  Dimitrios Argyriou
;
;              Hardware: STK600, CPU ATmega2560
;
;              Function: Reads the switches and when we press SW5, LED0 lights up.
;
;              Input ports: On-board switches connected to PORTA.
;              Output ports: On-board LEDs connected to PORTB.
;
;              Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
.include "m2560def.inc"

 .DEF mr = r16                               ; Assigning register r16 as mr.
 .DEF mr2 = r17                              ; Assigning register r17 as mr2.

              ldi mr,0xff                    ; Load register r16 with 0xff hex.
```

```
            out DDRB,mr                     ; Sets the direction of port b according to the
                                            ;            r16 value.



my_loop:
            in mr2, PINA                    ; Read port a as input.
            out PORTB,mr                    ; Output r16 to port b

            cpi mr2,0b11011111              ; Compare register r17 with 0b11011111
            breq equal                      ; Branch if registers equal
            brne notEqual                   ; Branch if registers not equal jump to
                                            ;            notEqual

            rjmp my_loop                    ; Relative jump to my_loop
equal:

            ldi mr,0b11111110               ; Make the lower 1 bit output
            out PORTB,mr                    ;            for port b.
            rjmp my_loop                    ; Relative jump to my_loop


notEqual:
            ldi mr,0xff                     ; Load register r16 with 0xff hex.
            out PORTB, mr                   ; Output r16 to port b
            rjmp my_loop                    ; Relative jump to my_loop
```

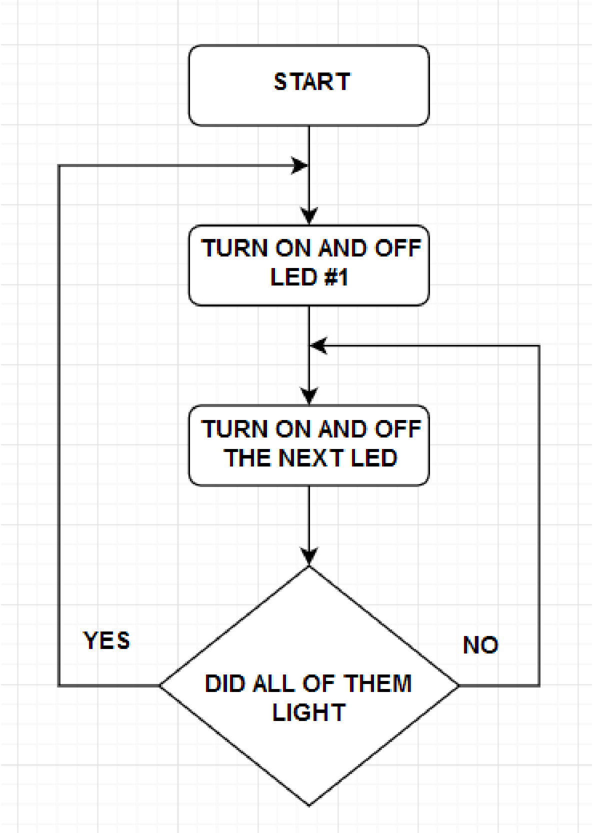
Task 5

Write a program in Assembly language
that creates a Ring Counter. The values should be displayed with the LEDs. Use shift instructions, LSL or LSR. Make a delay of approximately 0.5 sec in between each count. Write the delay as a subroutine. For using the subroutine, you must initialize the Stack Pointer, SP. Include the following instructions in beginning of your program:

```
; Initialize SP, Stack Pointer
ldi r20, HIGH(RAMEND) ; R20 = high part of RAMEND address
out SPH,R20           ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)  ; R20 = low part of RAMEND address
out SPL,R20           ; SPL= low part of RAMEND address
```

Function, the 8 LEDs:(0000 000X, 0000 00X0, 0000 0X00, 0000 X000, 000X0000, 00X0 0000, 0X00 0000, X000 0000)

Following task was to create a Ring Counter using instructions such as LSL and LSR. Starting with activating LED0, followed by a delay of 0,5 seconds and LSL instruction is instructed after the first delay to shift the bit one step left and activating the LED 1 (LED0 have been deactivated). The loop continues till all the bits have been shifted and the process is restarted by the CPI instruction that branches to subroutine “start” (creating an infinite loop).



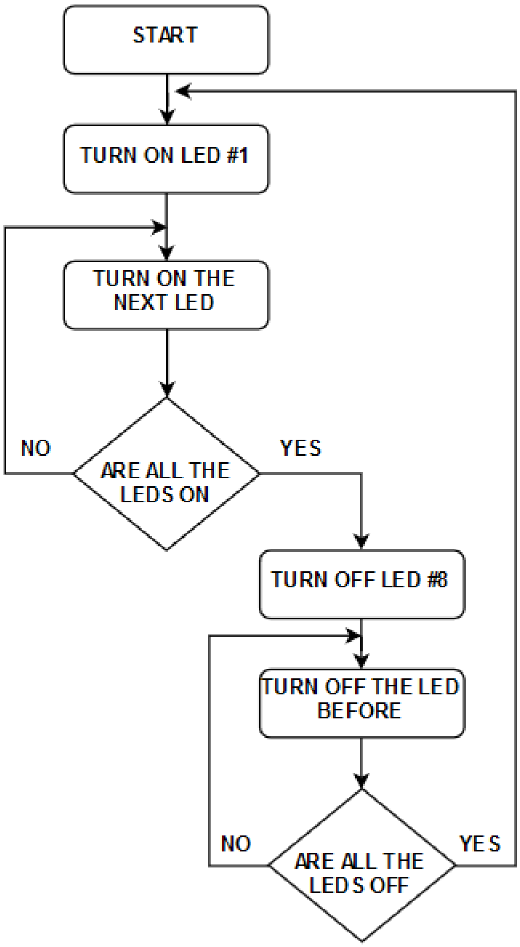
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;                    1DT301, Computer Technology I
;                    Date: 08/09/2017
;                    Authors:
;                                        Alexander Risteski
;                                        Dimitrios Argyriou
;
;                    Hardware: STK600, CPU ATmega2560
;
;                    Function: This program creates a Ring Counter and displays the values
;                    with the LEDs by using Shift instructions(lsl/lsr) with a delay of approx-
;                    imately 0.5 sec in between each count.
;
;                    Output ports: On-board LEDs connected to PORTB.
;
;                    Subroutines: A delay of 0.5 seconds at 1 MHz clock frequency.
;
;                    Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"
 ; Initialize SP, Stack Pointer
ldi r20, HIGH(RAMEND)           ; R20 = high part of RAMEND address
out SPH,R20                     ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)            ; R20 = low part of RAMEND address
out SPL,R20                     ; SPL = low part of RAMEND address

; Delay NaN cycles
; at 8.0 MHz
.DEF mr = r16                   ; Assigning register r16 onto mr

start:
ldi mr, 0b00000001              ; Make the lower 1 bit output
out DDRB, mr                    ;           for Port B.
rcall delay                     ; Relative call on delay subroutine

myloop:
lsl mr                          ; Logical shift of register r16 left.
out DDRB, mr                    ; Make the lower 1 bit output for Port B.

cpi mr, 0b00000000              ; Compare register r16 to 0
breq equal                      ; and if equal, branch to equal
rcall delay                     ; Relative call to delay subroutine
rjmp myloop                     ;Relative jump to start

equal:
rjmp start                      ; Relative jump to start

delay:

   ldi  r18, 5
   ldi  r19, 15
   ldi  r20, 242
```

```
L1: dec  r20
    brne L1
    dec  r19
    brne L1
    dec  r18
    brne L1
    ret
```

Task 6

Write a program in Assembly language that creates a Johnson Counter in an infinite loop.
Function, the 8 LEDs:
(0000 000X, 0000 00XX, 0000 0XXX, 0000 XXXX, 000XXXXX, 00XX XXXX, 0XXXXXXX, XXXXXXXX, 0XXXXXXX, 00XXXXXX, 000XXXXX, 0000 XXXX, 0000 0XXX, 0000 00XX, 0000 000X, 0000 0000)

Final task was to create a Johnson counter. The idea and approach is similar to Ring Counter, however, in this case we let the first LED0 to stay active even after the logical shift to the left (same for the third LED3, LED1 and 0 are kept active). The code consists of three parts "forward" (led is activated from left to right LED0, 1,2 and so on), "backwards" (opposite of forward process) and Reset. Starting from subroutine "forward" followed by a CPI instruction to jump to "Reset" subroutine to initialize the register with opposite bit from the starting point and jump to subroutine "backwards" where it deactivates the LED's on opposite direction.
The entire process is trapped in infinite loop.



```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;               1DT301, Computer Technology I
;               Date: 08/09/2017
;               Authors:
;                                                       Alexander Risteski
;                                                       Dimitrios Argyriou
;
;               Hardware: STK600, CPU ATmega2560
;
;               Function: This program creates a Johnson Counter in an infinite loop.
;
;               Output ports: On-board LEDs connected to PORTB.
;
;               Included files: m2560def.inc
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

.include "m2560def.inc"
 ; Initialize SP, Stack Pointer
ldi r20, HIGH(RAMEND)                   ; R20 = high part of RAMEND address
out SPH,R20                             ; SPH = high part of RAMEND address
ldi R20, low(RAMEND)                    ; R20 = low part of RAMEND address
out SPL,R20                             ; SPL = low part of RAMEND address

.DEF mr = r16                           ; Assigning register r16 to mr.
.DEF mri = r17                          ; Assigning register r17 to mri.

ldi mr, 0xff                            ; Load 0b11111111 in R16.
out DDRB, mr                            ; Configure Port B as an Output port.

forward:
out PORTB, mr                           ; Write all 1's to the pins of PortA.
lsl mr                                            ; Apply logical shift left.
call delay                              ; Calls delay subroutine.
cpi mr, 0b00000000                      ; Compares register to 0.
brne forward                            ; If not equal branch to forward.
rjmp reset                              ; else relative jump to reset.

reset:
out PORTB,mr                            ; Outputs register r16 to Port B.

call delay                              ; Calls delay subroutine.

ldi mri,0b10000000                      ; Load 0b10000000 in R17.
rjmp backwards                          ; Relative jump to backwards.

 backwards:

 lsr mr                                 ; Logic shift of register r16 right.
 add mr, mri                            ; Adds registers r16 and r17 and puts the re-
sult in the destination register.
 out PORTB,mr                           ; Outputs register r16 to Port B.
 call delay                                  ; Calls delay subroutine.
 cpi mr, 0xFF                                ; Compares r16 to hex ff
 brne backwards                              ; Branches to backwards if not equal
rjmp forward                                 ; Relative jump to forward.

delay:

    ldi  r18, 3
    ldi  r19, 138
    ldi  r20, 86
L1: dec  r20
    brne L1
    dec  r19
    brne L1
    dec  r18
    brne L1
    rjmp PC+1

                  ret
```