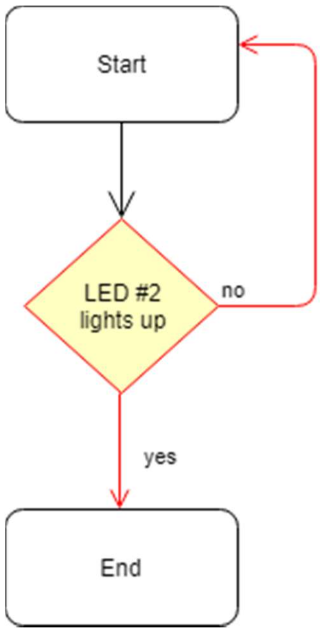
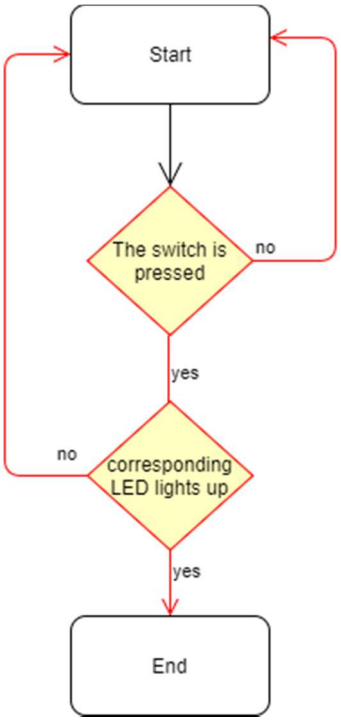
```
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
; 1DT301, Computer Technology I
; Date: 2016-09-11
; Author:
; Student name 1 Ruth Dirnfeld
; Student name 2 Alexandra Bjäremo
;
; Lab number: 1
; Title: How to use the PORTs. Digital input/output. Subroutine call.
;
; Hardware: STK600, CPU ATmega2560
;
; Function: Describe the function of the program, so that you can understand it,
; even if you're viewing this in a year from now!
;
; Input ports: Describe the function of used ports, for example on-board switches
; connected to PORTA.
;
; Output ports: Describe the function of used ports, for example on-board LEDs
; connected to PORTB.
;
; Subroutines: If applicable.
; Included files: m2560def.inc
;
; Other information:
;
; Changes in program: (Description and date)
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Task1

```
.include "m2560def.inc"      ;include file for Atmega 2560

ldi r16, 0b00000100
out DDRB, r16                ; One one to DDRB, input

/*
*Description:
*Using the only corresponding LED as input through
*Register 16
*Minimum instructions used in this case: 2
*/
```

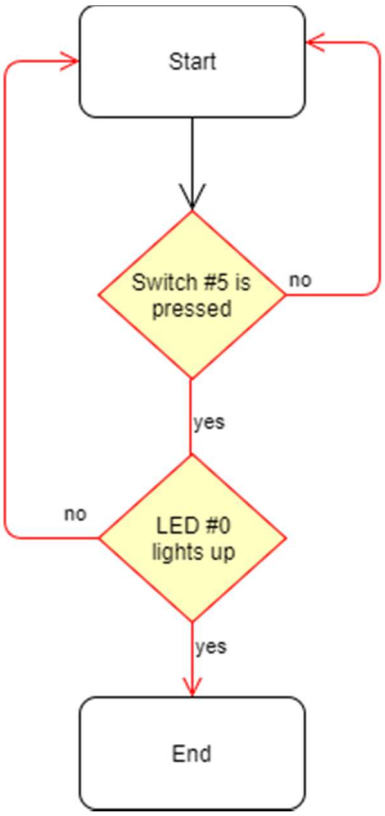


Task2




```
.include "m2560def.inc"

ldi r16, 0xFF
out DDRB, r16              ; All one's to DDRB, outputs

ldi r16, 0b00000000
out DDRA, r16              ; All zero's to DDRA, inputs

loop:
in r16, PINA               ; read PINA
out PORTB, r16             ; write in PORTB

rJmp loop

/*Description:
*Using DDRB as output and DDRA as input, so that when pressing one of the switches the
*corresponding LED will be lighted up
*/
```

Task3

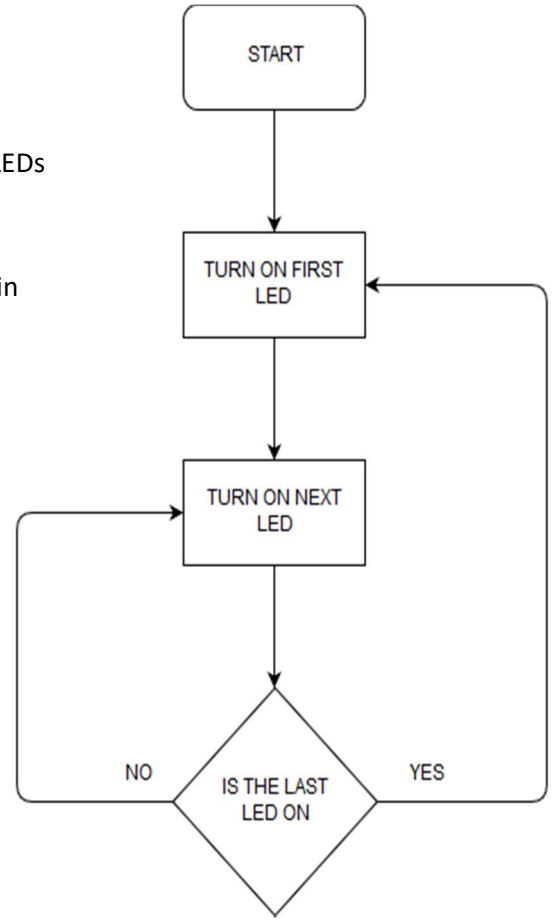



```
.include "m2560def.inc"

ldi r16, 0b00000001      ; One one to DDRB, output
out DDRB, r16

ldi r16, 0b11011111      ; One zero to DDRA, input
out DDRA, r16

loop:
ldi r16, 0b11111111
out PORTB, r16             ; turn off all LEDs
in r16, PINA               ; read PINA
cpi r16, 0                 ; check if switch is pressed
brne loop                  ; restart loop
ldi r16, 0b11111110
out PORTB,r16              ; turn on LED0

rJmp loop

/*Description:
* Using only the first LED (LED0) as output and switch 5(SW5) as input.
```

* Checking if the correct switch is pressed and if it is, light up LED0, otherwise does nothing.
*/

Task 5

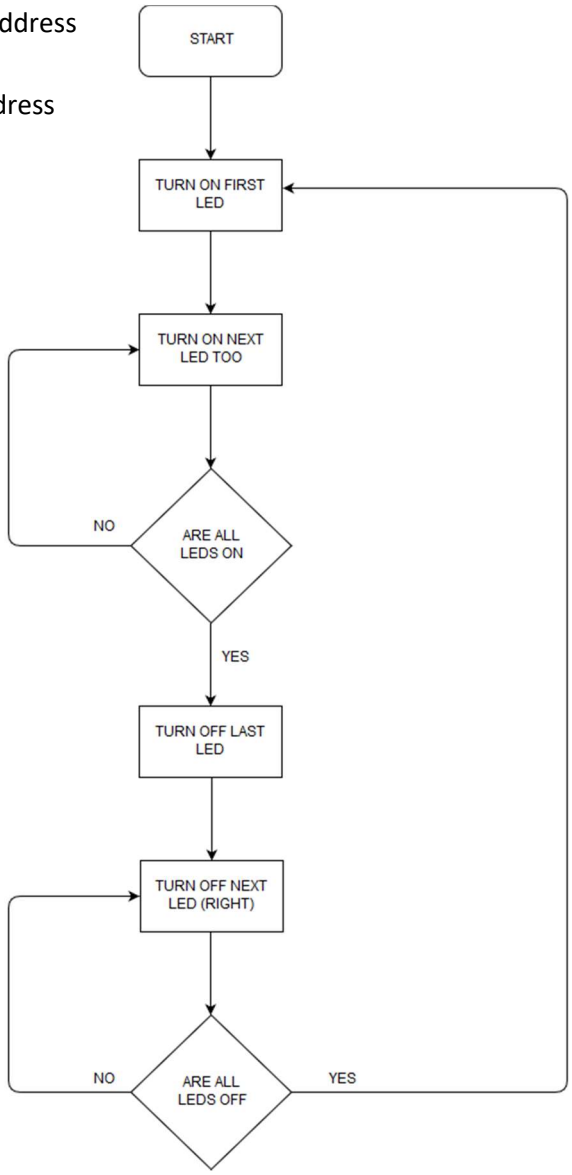
```
.include "m2560def.inc"

; 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

ldi r20, 0xFF
out DDRB, r20              ; All one's to DDRB, outputs

ldi r16, 0xFE              ; starting with LED0

floop:
cpi r16, 0xFF              ; checking if all LEDs are off
breq equal
out PORTB, r16             ; write in PORTB, turning on LEDs
com r16                    ; inverting the bits of r16
lsl r16                    ; pushing a 0 to the left
com r16                    ; inverting the bits of r16 again
rjmp delay
rjmp floop

equal:
ldi r16, 0xFE

rjmp floop

; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
;
; Delay 500 000 cycles
; 500ms at 1 MHz

delay:
    ldi  r18, 3
    ldi  r19, 138
    ldi  r21, 86
L1: dec  r21
    brne L1
```



```
    dec  r19
    brne L1
    dec  r18
    brne L1
    rjmp PC+1

rjmp floop

/*Description:
*  Using DDRB as output, we light up LED0. Checking if all LEDs are off we start writing in PORTB and
* turn the LEDs on by pushing a 0 with the Logical Shift Left to the left side.
*/
```

Task 6

```
.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

ldi r20, 0xFF
out DDRB, r20       ; All one's to DDRB, outputs

ldi r16, 0xFE       ; turn on LED0
ldi r17, 0x00       ; temp register to help with sloop

floop:
cpi r16, 0x00       ; check if all LEDs are on
breq sloop
out PORTB, r16      ; write to PORTB
lsl r16             ; pushing 0 to the left
rcall delay
rjmp floop

sloop:
out PORTB, r16      ; write to PORTB
cpi r16, 0xFF       ; check if all LEDs are off
breq floop
mov r17, r16        ; move r16's bits to r17
com r17             ; invert r17's bits
lsr r17             ; pushing 0 to the right
```



```
com r17             ; invert r17's bits again
mov r16, r17        ; move r17's bits to r16
rcall delay
rjmp sloop

; Generated by delay loop calculator
; at http://www.bretmulvey.com/avrdelay.html
;
; Delay 500 000 cycles
; 500ms at 1 MHz

delay:

    ldi  r18, 3
    ldi  r19, 138
    ldi  r21, 86
L1: dec  r21
    brne L1
    dec  r19
    brne L1
    dec  r18
    brne L1
    rjmp PC+1

ret

/*Description:
*  Using DDRB as output, we light up LED0. Checking if all LEDs are off we start writing in PORTB and
* turn the LEDs on by pushing a 0 with the Logical Shift Left to the left side. Writing to PORTB we
* check if all LEDs are off and move the register 16 bits to register 17 and invert it's bits and push 0's
* to the right. Inverting the r17's bits again and moving r17's bits to r16. At the end returning to
* subroutine.
*/
```