Init Assembly!

```
    init:

    .equ F_osc = 4000000 #klokkefrekvens
    .equ tick = F_osc/1024 #klokkefrekvens med prescalar

#init stack
    .def tmp = R16 #definer tmp til register 16

    ldi tmp, low(ramend)
    out spl, tmp
    ldi tmp, high(ramend)
    out sph, tmp

#init port
    ldi tmp, 0xFF #alle bit settes til 1, PORTA som output (high, alle bit til 1)
    out DDRA, tmp
    ldi tmp, 0x0 #alle bit settes til 0, PORTB som input (low, alle bit til 0)
    out DDRB, tmp
    ldi tmp, 0xFF
    ldi tmp, &(0x2) #bit 1 settes til 0 (low), kontroll bit satt til input
    #(rest av bits er output)
    out DDRC, tmp

#init timer

#timer compare
    ldi tmp,low(tick)
    out OCR1Bl, tmp
    ldi tmp, high(tick)
    out OCR1BH, tmp

#timer counter
    ldi tmp, (1 << WGM12) | (1 << CS12) | (1 << CS10) #WGM12 = Clear on timer (CTC)
    #alternativ: ldi tmp, (1 << ICES1) #ICES1 = Input capture. Trengs | (OR) med prescaler
    #CS12 + CS10 = 1024 prescaler bits
    #CS11 + CS10 = 64 prescaler bits
    out TCCRIB, tmp

#clear timer counter
    clr tmp
    out TCNT1L,tmp
    out TCNT1H,tmp
```

```
#enable interrupts on compare
    ldi tmp, (1 << OCIE1B) #Output compare enable
    #alternativ: ldi tmp, (1 << TICIE1) #Timer input capture enable
    out TIMSK, tmp

#enable interrupts
    sei

    .org OCR1Baddr #adresse 18 - hopp til ISR
    jmp ISR

    ISR:
    push tmp #push tmp til stack
    in tmp,sreg #hent fra sreg til tmp
    push tmp #push tmp til stack (sreg)
    ...
    ...
    pop tmp #hent tmp fra stack
    out sreg, tmp #send tmp til sreg
    pop tmp #hent tmp fra stack
    reti
```