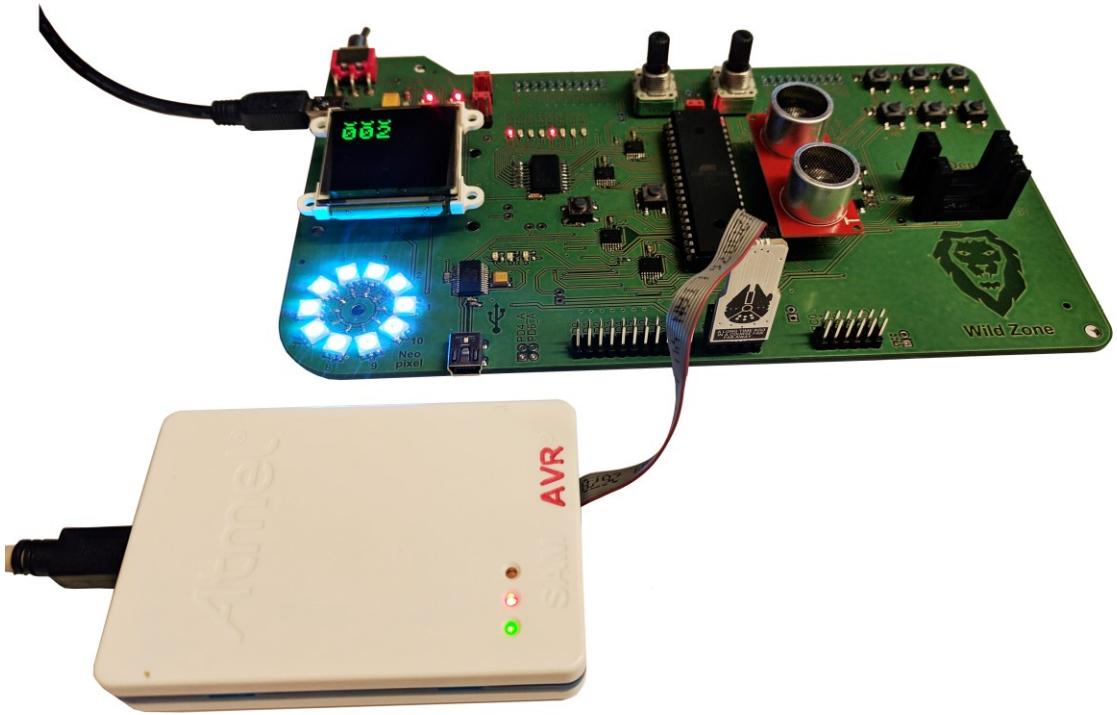
Digitala system EITA15

Elektro- och informationsteknik

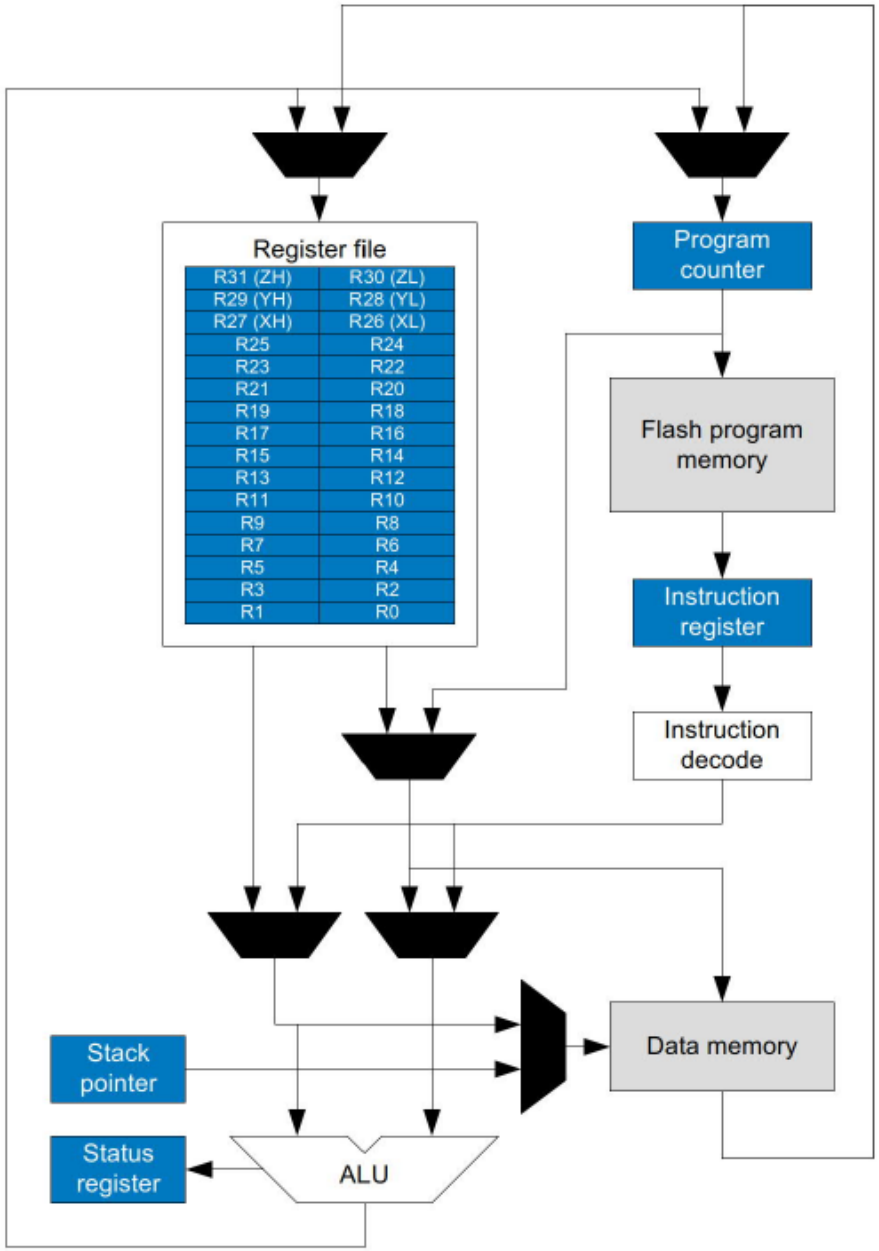
Laboration 3

Labbkort med JTAG debugger



AVR CPU core

R0 –r31 8 bit reg.
R26 –R31 16-bit indirect adr.
SP 16 bit



Assemblykod

```
start:
        ldi r16 , 12     ; load 12 into register 16
        ldi r17 , 13     ; load 13 into register 17
        add r16 , r17 ; add r16 and r17 , save result in r16
        cpi r16 , 26   ; compare r16 with the values 26
        breq end       ; jump to end if true
        add r16 , 1    ; add 1 to r16 , r16 now holds the value 26
                       ; after this instruction , we will execute the rjmp end
end:
rjmp end               ; infinite loop
```

https://www.microchip.com/webdoc/avrassembler/avrassembler.wb_instruction_list.html

I/O Assembly

Register	Address
PINA	0x00
DDRA	0x01
PORTA	0x02
PINB	0x03
DDRB	0x04
PORTB	0x05
PINC	0x06
DDRC	0x07
PORTC	0x08
PIND	0x09
DDRD	0x0A
PORTD	0x0B

I/O

```
#define DDRB 0x04
#define PORTB 0x05

start:
        ldi r18 , 0x01
        out DDRB , r18      ; set LED0 as output
        out PORTB , r18     ; turn on LED0. r18 already contains 0x01 ;)
end:
        rjmp end            ; infinite loop
```

Exekveringstid

- Varje instruktion tar en eller ett antal klockpulser
- Klockpulsen är beroende på klockfrekvensen
- Ex: 16 Mhz
- 1klockpuls ∗1/16MHz =62.5 ns

Medelvärdesberäkning av loopar:

$$\frac{n-1}{n} * p + \frac{1}{n} * \quad q$$

Se sidan 7 i labbhandledning

stack

- På samma sätt kan en subrutin lägga upp data på stacken. När subrutinen (uppgiften) är klar tas data bort från stacken igen.
- För att krångla till det: Stacken i vår dator växer nedåt
- Det som motsvarar pilen (håller reda på stackens topp) är ett register: stackpekaren (sp).

Stack i AVR

- Ligger i RAM, används för lokala och temporära variabler samt återhoppsadresser från subrutiner och interrupt.
- Växer från högre adress till lägre
- Vill vi allokera minne, subtrahera stackpekaren
- Lämna tillbaka allokerat minne, addera stackpekaren
- SP är på 16 bitar, minnesmappad på adress 0x3D.

Subrutiner

- Call

  Återhoppsadressen sparas på stacken (push)
- Ret

  Återhoppsadressen hämtas från stacken (pop)

Viktigt att alla register som används i subrutinen sparas på stacken.

Blanda C och Assemby

- I C-program

  extern char myadd (char, char);

Myadd finns någon annanstans, exempelvis en assemblerfil

- I Assemblerfil

  .global myadd

Myadd ska vara åtkomlig utanför assemblermiljön (C-fil)

- Register r25 till r8 parvis, övrigt på stack
- Returnvärde i r24 och r25

C-> assembler

assembler -> C