WEEK 2

ONDERWERPEN

- flow of control
- spring instructies, control structuren
- AVR studio & Arduino UNO
- stack & subroutines
- interrupts
- timer/counters
- switch bounce

GROEN GEMARKEERD = AFTEKEN OPDRACHT

OPDRACHT 1 : EEN C PROGRAMMA

Gegeven onderstaand C programma.

```
int a = 3;
int b = 7;
int c;
do {
    a = a-1;
    b = b-1;
    c = a;
    if (b == 6) {
        b = b-a;
    } else {
        if (b == 3) {
            a = a-1;
        }
    }
} while (a > 0);
```

a) Welke waarde hebben a, b en c na afloop van het programma ?
b) Herschrijf het bovenstaande C programma in AVR assembly.

OPDRACHT 2 : SUBROUTINE FIB

De rij van Fibonacci is de reeks 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... Zie voor meer uitleg Wiki Fibonacci.

a) Schrijf in Python (of een andere hogere programmeertaal) een functie die de rij van Fibonacci afdrukt.
b) "Vertaal" dit programma naar assembly. Je hoeft geen getallen groter dan 255 te genereren. Hoevaak moet je dan de lus doorlopen ?
c) Hoeveel klok-cycles doet je programma erover om de eindwaarde te berekenen ? (De debugger houdt dit ook voor je bij. Je kan dit zien in Debug > Windows > Processor view "cycle counter").

OPDRACHT 3 : EEN REEKS

Schrijf Schrijf in Python (of een andere hogere programmeertaal) een programma dat de reeks getallen 1, 3, 2, 6, 5, 15, 14, … afdrukt, waarbij de laatste waarde <= 255 moet zijn.

Vertaal dit programma naar een assembly programma dat deze reeks getallen plaatst in het datasegment vanaf adres 0x100.

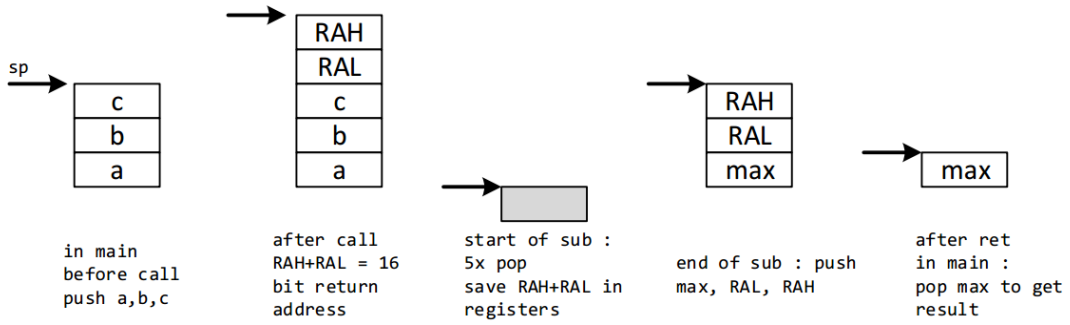
OPDRACHT 4 : MAX(A,B,C)

a) Schrijf in Python (of een andere hogere programmeertaal) een programma dat het maximum van 5 getallen bepaalt. Vertaal dit programma naar AVR assembly. Schrijf een subroutine max(a,b,c,d,e) die het maximum van 5 getallen a t/m e bepaalt. Roep de subroutine aan vanuit het hoofdprogramma met waarden voor a t/m e. Geef de waarden van a t/m e door via registers.

b) Laat het Z register wijzen naar data en gebruik de lpm-instructie om de getallen één voor één naar r0 te halen. De 5 getallen worden eerst in het code segment opgeslagen als volgt :

```
data:
   .db 5,7,2,3,9 ; getallen a t/m e
```

c) Zelfde als (a), maar nu worden parameters a t/m e en het resultaat doorgegeven via de *stack*. Het lastige hierbij is dat de call-instructie het return adres op de stack zet, en de ret-instructie het return adres van de stack haalt. Onderstaande figuur geeft weer hoe je dit kan oplossen (alleen a,b, en c zijn getekend) :



OPDRACHT 5 : REKENEN MET 16-BIT GETALLEN

a) Beschrijf hoe je met AVR instructies 2 positieve 16-bit getallen kunt optellen. Kijk eens goed naar de instrcties ADD en ADC. Hoe groot is het resultaat maximaal (in bits) ?

b) Schrijf een programma dat deze optelling demonstreert.

OPDRACHT 6 : ASSEMBLER OUTPUT FILES

Bouw de assembly code uit de vorige opgave. Zoek de map en list file (.lss) van het project op in je filesysteem.

a) Welke informatie geeft de map file ?
b) Bekijk de list file. Hoe groot zijn het data en het code segment ? Neem de relevante regels over in je uitwerking.
c) En hoeveel bytes gebruikt je programma van beide segmenten ?
d) Wat is de eerste instructie ? Op welk adres staat deze ? Wat is de opcode van deze instructie ? (Klopt dit met de opcode zoals vermeld in de AVR instructieset ?)
e) Open de executable (.hex) in een tekst editor. In het college is uitgelegd hoe de layout van deze file is. Komen opcodes in de hex file precies overeen met die in de list file ? Zo nee, wat is het verschil ?

OPDRACHT 7 : KNOPJES LEZEN

In deze opdracht gaan we beginnen met het Arduino bordje.

Voorbereiding

Installeer avrdude in (bijvoorbeeld) c:\avrdude. Het is handig om dan "c:\avrdude" toe te voegen aan je PATH-variabele. Dit kan via "this PC > properties > advanced system settings". Test of je avrdude kunt uitvoeren in de Windows shell.

Om te controleren of het allemaal werkt is het een goed idee om de executable uni_test.hex op het bordje te flashen. Verbind het bordje met je laptop. Controleer welke COM-poort je moet hebben via Device Manager. Je kan nu het hex-bestand gaan flashen met avrdude, zie de sheets van week 1. Verbind poorten PB0 en PB1 elk met een LED, en elke LED via een weerstand met de ground. Zet de LED's in de goede richting. Als het goed is knipperen de LED's één keer per seconde.

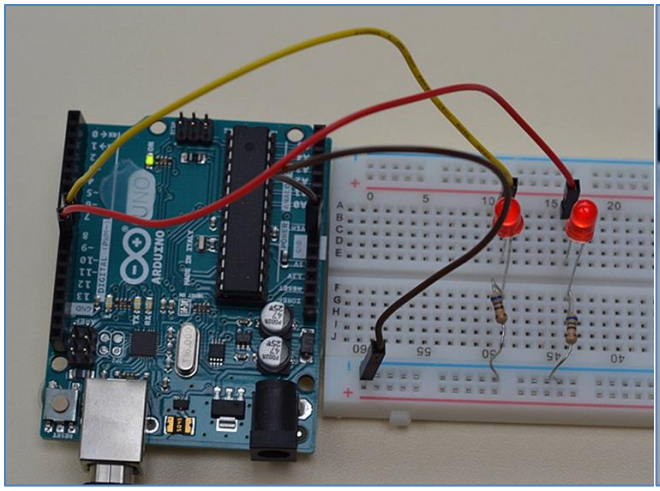
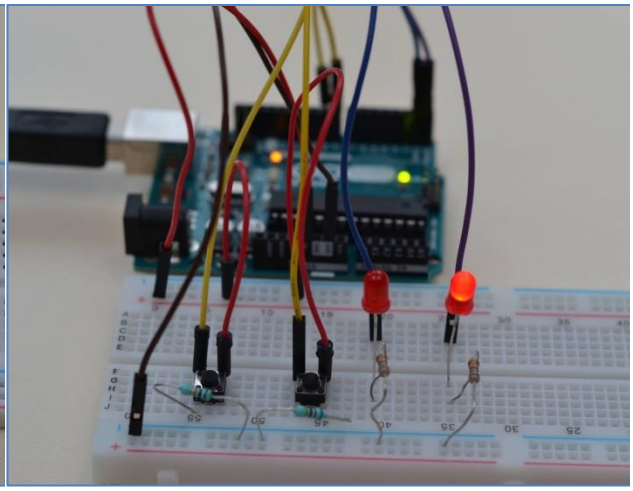
Opdracht

Maak een nieuw assembler project aan en kies als device ATmega328P.

Probeer netjes te werken op het breadboard. Verbind op het breadboard eerst een blauwe min-lijn met de ground en een rode plus-lijn met de 5V. Verbind daarna twee LED’s met port D (PD0 en PD1), elk via een weerstand naar de ground (zie foto). Plaats de LED's in de goede richting. Verbind dan twee schakelaars met port B (PB0 en PB1) elk via een weerstand met de ground, en via een schakelaar met de 5V.

Dus let er op :

- configureer PORT D als output via DDRD, en schrijf naar PORTD;
- configureer PORT B als input via DDRB, en lees op PINB.

a) Maak een programma dat een LED laat branden wanneer een knopje wordt ingedrukt. Wanneer knop 0 wordt ingedrukt, moet LED 0 gaan branden. Wanneer knop 1 wordt ingedrukt, moet LED 1 gaan branden.

b) Pas de vorige opdracht zodanig aan dat de LED blijft branden *ook wanneer je de knop loslaat*. De LED moet net zolang blijven branden tot de andere knop wordt ingedrukt.

Opmerking : deze opdracht kan je maken zonder gebruik te maken van externe interrupts, gewoon in een lus de waarde van de knopjes uitlezen.

OPDRACHT 8 : KNIPPERLICHT

a) In de Atmega datasheet is een hoofdstuk opgenomen over de 16-bit timer/counter1. De laatste paragraaf in dit hoofdstuk beschrijft de control registers van deze timer. Bij het debuggen van een programma leest iemand de volgende registers met bijbehorende waarden :

```
OCR1A  = 0b0000111110100000
TCCR1B = 0b00001010
TIMSK1 = 0b00000010
```

Bepaal wat deze instellingen betekenen, m.a.w. wat doet timer 1 en hoe vaak per seconde doet hij dit ?

b) Schrijf een programma dat alle LEDS laat knipperen met een periode van 0,5 seconde. Voor het bepalen van het interval gebruik je de 16-bit timer 1 uit (a). Genereer elke 0,5 seconde een interrupt door de teller te laten tellen totdat een "output compare match" bereikt is. Geef in je antwoord aan hoe je de waarde van OCR1A hebt berekend.

OPDRACHT 9 : KNOPAANSLAGEN TELLEN

a) Verbind een knopje met poort B0 en drie LED's met poort D0..2. Maak een programma dat telt hoe vaak de knop wordt ingedrukt. Het aantal wordt in binaire vorm weergeven via 3 LED's (dus 5 wordt bijvoorbeeld weergeven als 101).
b) Zeer waarschijnlijk klopt je teller niet. Wat is het probleem ? Hoe vaak, dus om de hoeveel tijd, “kijkt” de CPU naar het knopje ?
c) Verhelp het probleem door minder vaak naar het knopje te kijken. Om de hoeveel tijd ga je kijken ? Realiseer de oplossing met behulp van een interrupt van timer 1.
d) Pas het programma zodanig aan dat de knop verbonden wordt met externe interrupt1 (INT1). Hoofdstuk 13 uit de Atmega datasheet behandelt de externe interrupts. Zorg ervoor dat de interrupt wordt ge-triggerd door de opgaande flank (dus bij het indrukken van het knopje). In de ISR van INT1 hou je een teller bij die je laat zien via de LEDS, als in (a).
e) Hoe is het resultaat nu, is het probleem uit (b) verholpen ?