A&C week 2

Remi Reuvekamp - Timo Strating - ITV2E

1 oktober 2018, Groningen

Inhoudsopgave

1. *Opdracht 1*

```c
#include <stdio.h>

int main() {

    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);

    printf("%d", a);
    printf("%d", b);
    printf("%d", c);
}
```

1.1 A

Welke waarde hebben a, b en c na afloop van het programma ?

A: 0; B: 3; C: 1

(Tussenstappen:
A: 3; B: 7; C: 0
If
A: 2; B: 4; C: 2

A: 2; B: 4; C: 2
Elif
A: 0; B: 3; C: 1
)

1.2 B

Herschrijf het bovenstaande C programma in AVR assembly.

```
.def a=r16
.def b=r17
.def c=r18

start:                  ; Initialize de variabelen
  ldi a, 0x03
  ldi b, 0x07
  ldi c, 0x00

loop:                   ; De main loop
  dec a
  dec b
  mov c, a
  cpi b, 6
  breq ifbeq6
  cpi b, 3
  breq ifbeq3

ifbeq6:                 ; De eerste IF
  sub b, a
  rjmp loopend

ifbeq3:                 ; De tweede IF
  dec a
  rjmp loopend

loopend:                ; Het einde van de loop
  cpi a, 0
  breq end
  rjmp loop

end:                   ; Infinite loop
  rjmp end
```

2. *Opdracht 2*

2.1 Python

Schrijf in Python (of een andere hogereprogrammeertaal) een functie die de rij van Fibonacci afdrukt

```
def fib(i):
    n1, n2 = 1, 1; listje = [1, 1]
    while i > 0:
        n1, n2 = n2, n1 + n2; listje.append(n2); i -= 1
    return listje

print(fib(20))
```

```
>>> [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987,
    1597, 2584, 4181, 6765, 10946, 17711]
```

2.2 Assembly

"Vertaal"dit programma naar assembly. Je hoeft geen getallen groter dan 255 te genereren. Hoe vaak moet je dan de lus doorlopen?

```
.def n1=r16
.def n2=r17
.def temp=r18
.def counter=r19

start:            ; Initialize de variabelen
  ldi n1, 0x01
  ldi n2, 0x01

loop:            ; De main loop
  mov temp, n2
  add n2, n1
  brcs loopend    ; Als add een overflow veroorzaakt dan kunnen we niet verder
  mov n1, temp
  inc counter
  rjmp loop

loopend:    ; Onze laatste berekening veroorzaakte een overflow
            ; dus we moeten de oude waarde nog ff terug zetten
            ; om het juiste antwoord in het geheugen te laten staan
  mov n2, temp
    rjmp end
```

```
end:
  rjmp end
```

2.3 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").

Cycle counter: 83

3. *Opdracht 3*

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 kleiner dan of gelijk aan 255 moet zijn.

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

```
.def a=r16
.def temp=r17

ldi a, 0x01
ldi xl, low(0x100)
ldi xh, high(0x100)
st x+, a

loop:
  ldi temp, 0x03
  mul a, temp          ; * 3
  mov temp, r1
  mov a, r0
  cpi temp, 0x01
  brpl end             ; jump naar het einde als we niet verder kunnen
  st x+, a
  dec a                ; -1
  st x+, a
  rjmp loop

end:
  rjmp end
```

4. *Opdracht 7*

4.1 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.

```
start:
  ldi r16, low(RAMEND)
  out SPL, r16
  ldi r16, high(RAMEND)
  out SPH, r16

  ldi r16, 0xFF           ; Pins B to output
  out DDRB, r16

  ldi r16, 0x00           ; Pins D to input
  out DDRD, r16

loop:
  in r16, PinD
  out PortB, r16

  rjmp loop               ; jump to loop
```

4.2 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.

```
.def temp=r16
.def state=r17

start:
  ldi temp, low(RAMEND)
  out SPL, temp
  ldi temp, high(RAMEND)
  out SPH, temp

  ldi temp, 0xFF      ; Pins B to output
  out DDRB, temp
```

```
  ldi temp, 0x00      ; Pins D to input
  out DDRD, temp

  ldi state, 0xFF

disable:
  ldi temp, 0x00
  out PortB, temp

loopoff:
  sbic PinD, 2
  rjmp enable
  rjmp loopoff

enable:
  ldi temp, 0xFF
  out PortB, temp

loopon:
  sbic PinD, 3
  rjmp disable
  rjmp loopon
```

5. *Opdracht 8*

5.1 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      ; De Maximale waarde van de timer
TCCR1B = 0b00001010             ; Prescaler van 8 met de CTC Timer modus
                                ;  wat betekent Clear Timer on Compare
TIMSK1= 0b00000010              ; Compare A Math Interrupt Enable
```

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

0.5 sec = (256/16.000.000) * 31250

5.2 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 1uit (a). Genereer elke 0,5 seconde een interrupt door de teller te laten tellen totdat een öutput compare match"bereikt is. Geef in je antwoord aan hoe je de waarde van OCR1A hebt berekend.

```
.def var = r16
.def temp = r17
.def tmpSREG = r18
.org 0x0000

rjmp init

.org OC1Aaddr
    rjmp TIMER1_COMP_ISR        ; adres ISR (Timer1 Output Compare Match)

TIMER1_COMP_ISR:                ; ISR wordt elke seconde aangeroepen
    in tmpSREG, SREG
    in var, PORTB
    com var
    out PORTB, var              ; Schrijf de inverteerde waarde terug naar PORTB
    out SREG, tmpSREG           ; Haal SREG terug uit temp
    reti                        ; Return van de interrupt
```

```
init:
    ldi var, high(RAMEND)       ; init stack pointer
    out SPH, var
    ldi var, low(RAMEND)
    out SPL, var

    ; 0.5 sec = (256/16.000.000) * 31250
    ldi temp, high(31250)
    sts OCR1AH, temp
    ldi temp, low(31250)
    sts OCR1AL, temp

    ; Zet de prescaler op 256 en stel de timer in op de CTC-modes
    ldi temp, (1 << CS12) | (1 << WGM12)
    sts TCCR1B, temp

    ldi temp, (1 << OCIE1A)     ; Zet de interrupt aan
    sts TIMSK1, temp

    ser temp                    ; tmp = 0xFF
    out DDRB, temp              ; Port B is output port (via LEDs)
    out PORTB, temp             ; LEDs uitzetten

    sei                         ; Zet alle interrupts aan

loop:
    rjmp loop
```