WEEK 1

ONDERWERPEN

- de structuur van AVR-assembly
- AVR instructies
- AVR registers en I/O
- ATmega memory map
- Atmel Studio
- AVR expressies en directives
- AVR addressing modes

GROEN GEMARKEERD = AFTEKEN OPDRACHT

OPDRACHT 1 : AVR INSTRUCTIESET

a) Met welke instructie kan je een (heel) register met nullen vullen, en met welk instructie een register met enen vullen ?
b) Wat doet de instructie NEG ? Als operatie staat hier : Rd <- $00 – Rd. Klopt dit wel met wat er in het college behandeld is over two's complement ?
c) In de "instruction set summary" staat een overzicht van alle branch instructies. Hoe kun je direct zien of het om een conditionele sprong gaat of niet ? (Tip : kijk eens in de kolom operation).
d) Waarvoor dienen de instructies IN en OUT ?
e) Hoelang duurt de CALL instructie (bij de ATmega32) ? Waarom duurt deze instructie zo lang ? (Tip : zie DIjkstra paragraaf 7.2.1).
f) Wat is het verschil tussen RJMP en JMP ? Wat gebeurt er als je die instructie "RJMP 2049" zou geven ?
g) Wat is het verschil tussen ADC en ADD ?
h) Stel dat je 2 registers bij elkaar op wil tellen die allemaal '1'-nen bevatten, is dan één carry bit wel voldoende ?

OPDRACHT 2 : WELKE INSTRUCTIE ?

In het code segment van de ATmega328P ziet iemand de volgende instructies:

0000 1100 0101 0101

Geef de AVR instructies en operand(en) welke horen bij deze machinecode. Interessant in dit geval is dat er twee antwoorden mogelijk zijn. Hiervoor moet je kijken naar de opcodes van de instructies. Bedenk dat de CPU alleen maar nullen en enen 'ziet', waarbij (in dit geval) de eerste zes bits van de instructie uniek moet zijn voor de instructie.

Om het zoeken in de instructieset te beperken is gegeven dat de instructie begint met een 'a' en/of een 'l'.

Kun je uitleggen waarom er in dit geval twee antwoorden mogelijk zijn ?

OPDRACHT 3 : HET BESTAND M328PDEF.INC

De AVR assembly code voor de ATmega328P begint altijd met de directive **`.include "m328Pdef.inc".`**

a) Waar staat dit bestand in je file systeem ?
b) Met welke adressen corresponderen de symbolen PORTB, DDRB en PINB ? (Zoek in het tekstbestand m328Pdef.inc).
c) Waarvoor staan de symbolen XH en XL ?
d) Wat is het startadres voor het RAM ? Wat is de grootte van het RAM ? Wat is het hoogste RAM adres ? (Data Memory Declarations)

OPDRACHT 4 : AVR EXPRESSIES

Geef aan wat het binaire resultaat is na de volgende stukjes code.

Voorbeeld : `ldi r16,0x00` heeft als binair resultaat `r16=s00000000`.

De identifiers `PB3`, `INT1` enz. kun je opzoeken in het bestand m32def.inc

a.

```
ldi r16,~0xf0
```

b.

```
.equ a = 5
.equ b = 0xff16
ldi r18,low(a|b)
```

c.

```
.equ a = 5
.equ b = 0xff
ldi r18,a^b
```

d.

```
.include "m32def.inc"
ldi r16, (1<<5)|(1<<7)
out PORTB, r16
```

e.

```
.include "m32def.inc"
ldi r16, ~(1<<PB3)
```

f.

```
.include "m32def.inc"
ldi r16,low(RAMEND)
out SPL,r16
ldi r16,high(RAMEND)
out SPH,r16
```

OPDRACHT 5 : RESULTAAT INSTRUCTIES

Neem aan dat voor een instructie wordt uitgevoerd steeds geldt : r16 = 0xAA en r17 = 0x12. Beantwoord onderstaande vragen met behulp van de instructieset.

a) Wat is na elke instructie de waarden van 16 en r17 ? NB. voor elke instructie worden r16 en r17 weer op bovenstaande waarden gezet.

- `neg r16`
- `swap r17`
- `sbr r16,3`
- `dec r17`
- `ori r16, 0xF0`

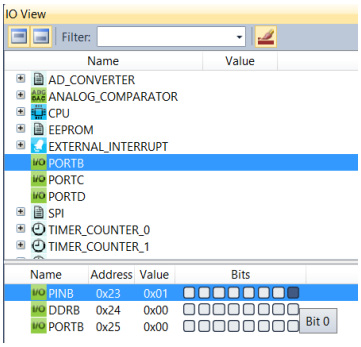
b) Wat is de waarden van r0 en r1 na : `mul r16, r17` ?

OPDRACHT 6 : BEGINNEN MET AVR STUDIO

Download Atmel Studio van Blackboard (onder “Atmel” staat een installer) of via internet (www.atmel.com, zie links op Blackboard). Eventueel kun je via het help menu de “AVR Studio User Guide” bekijken.

Op Blackboard is een eenvoudig assembly programma `assembly-intro-1.asm` te vinden. Maak een assembler project aan, kies als device ATmega328P, kopieer en “plak” de asm-file in de lege asm-file in je nieuwe project, compileer het project en kies als debugger “**AVR Simulator**”. Stap met de debugger door de source code.

a) Wat is het resultaat van dit programma ?
b) Met Debug > Windows > Processor-view kun je de waarden van registers bekijken. Maak een printscreen op het moment dat de variabele count gelijk aan 5 is. Neem een “printscreen” op in je uitwerkingen.
c) Met Debug > Windows > IO view kun je ook de waarde van PORTB bekijken. Neem een “printscreen” op in je uitwerkingen.



OPDRACHT 7 : BEGINNEN MET AVR STUDIO

Op Blackboard is een eenvoudig assembly programma assembly-intro-2.asm te vinden. Stap met de debugger door de source code.

a) Wat is het resultaat van dit programma ?
b) Maak een “printscreen” van de processor view waarin je laat zien dat variabele c de waarde 21 heeft gekregen.

OPDRACHT 8 : AVR DIRECTIVES

Gegeven de volgende declaraties in C. Vertaal deze declaraties naar assembly, waarbij je uitsluitend AVR **directives** gebruikt om geheugen te reserveren. Voorbeelden hiervan kun je vinden in de AVR Assembly User Guide (.byte, .db, enz.). Neem aan dat een int een storage size van 2 bytes heeft.

Een voorbeeld : “int x” wordt in assembly

```
.DSEG
x: .BYTE 2
```

a) int a;

b) char c;

c) const int c=9032;

d) const char b[ ]=“COMP9032”; (dit is een string)

OPDRACHT 9 : ADRESSING MODES

Gegeven het onderstaande programma.

```
.include "m32def.inc"
.org 0

    ldi zh, high(ASCII_TABLE << 1) ;define Z-pointer
    ldi zl, low(ASCII_TABLE << 1)

    ldi r16, 0x0
    out DDRC, r16 ;PORTC input
    ldi r16, 0xff
    out DDRD, r16 ;PORTD output
BEGIN:
    in r16, PINC
    andi r16, 0b00000111 ;mask upper 5 bits
    add zl, r16
    lpm r17, z
    out PORTD, r17
    rjmp BEGIN

.org 20
ASCII_TABLE:
```

```
  .DB '0','1','2','3','4','5','6','7'
```

a) Leg uit wat de instructie 'lpm' doet.
b) Geef voor elke instructie aan welke addressing mode wordt gebruikt en wat het type geheugen is (register, i/o, SRAM, flash).
c) Leg uit wat het programma doet (dus : wat is de input en wat is de output).

OPDRACHT 10 : AVR DIRECTIVES

a) Gegeven onderstaand programma. Zet bij elke regel passend commentaar.

```
.dseg
.org SRAM_START
dest: .byte 20

.cseg
     rjmp start

src: .db "hello world !"
.equ length=13
.def temp = r0
.def counter = r17

start:
     ldi zh,high(2*src) ; Z-pointer wijst naar ?
     ldi zl,low(2*src)
     ldi xh,high(dest)  ; X-pointer wijst naar ?
     ldi xl,low(dest)
     clr counter
loop:
     lpm
     inc zl
     st x+,temp
     inc counter
     cpi counter, length
     brlt loop
end:
     rjmp end
```

b) Wat is het resultaat van dit programma ?
c) Waarom wordt de waarde van src verdubbeld in `ldi zh,high(2*src)` ?
d) Het resultaat van dit programma is te zien in het geheugen. Maak een “printscreen” van de AVR Studio memory view waarin dit resultaat zichtbaar is.
e) Waarom kun je niet eenvoudig het volgende doen ?

```
.dseg
.db "hello world !"
```