Toets Parel 000 der Informatica (201300070)

5 september 2014, 10:45–11:45

- Je mag 1 zelfgemaakt A4'tje met aantekeningen bij dit tentamen gebruiken, en een *simpele* rekenmachine.
- Wetenschappelijke of grafische rekenmachines, laptops, mobiele telefoons, boeken e.d. zijn niet toegestaan. ***Stop deze nu in je tas!***
- Het aantal te behalen punten per opgave staat in de marge.

---

1. Binaire getallen

(a) Reken het 2-complements binaire getal 11001 om naar decimaal. Laat zien hoe je dit berekent. [7]

(b) Reken het hexadecimale getal 3F om naar decimaal. Laat zien hoe je dit berekent. [6]

(c) Stel je hebt een 8-bits unsigned binair getal dat kleiner is dan 2^6 en je wilt dit met 4 vermenigvuldigen. Dit kan door de bits te verschuiven: in welke richting, over hoeveel posities, en wat moet er op de vrijgekomen plek(ken) worden ingeschoven? Leg uit. [6]

(d) Verder over de vorige deelvraag: stel het was bij nader inzien een negatief, **1**-complements getal, tussen 0 en -2^5. Werkt diezelfde bitschuifoperatie dan ook goed? Geef een voorbeeld waaruit blijkt dat het dan niet correct werkt, of leg uit waarom het wel werkt. [6]

---

2. Booleaanse logica

(a) Geef de waarheidstabel van een "full-adder": een opteller die 3 getallen van elk 1 bit bij elkaar optelt, en als resultaat dus 2 bits produceert. [7]

(b) Stel je hebt een simpele opteller (2 bits in, 2 bits uit) en zo'n full-adder uit de vorige vraag (3 bits in, 2 bits uit). Schets hoe je deze samen zou kunnen gebruiken om een opteller voor twee 2-bits-getallen te maken (alles unsigned). [5]

(c) Vereenvoudig de volgende Booleaanse formule en geef daarbij aan welke Booleaanse rekenregel(s) je gebruikt; begin met het wegwerken van het '·'-teken. [7]

$A \cdot \left(\overline{A + B}\right)$

(d) Schets hoe je met alleen NAND-poorten de volgende formule kunt realiseren: [6]

$A \cdot \overline{B \cdot \overline{C}}$

Zie volgende bladzijde...

Bij onderstaande opgaven wordt de instructieset van de AVR-processor (bekend van de Arduino) gebruikt. Ter herinnering geven we de betekenis van veel gebruikte mnemonics:

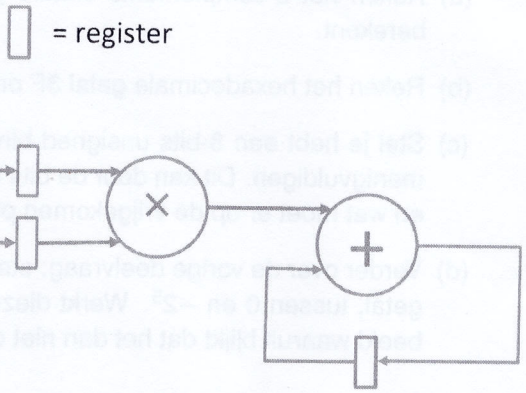
DEC betekent "DECrement (verminderen met 1)"
INC betekent "INCrement (vermeerderen met 1)"
SUB betekent "Subtract"
MUL betekent "MULtiply"
BREQ betekent "BRanch if EQual"
BRNE betekent "BRanch if Not Equal"

---

3. Opgave 3

15

In veel algoritmen vinden veelvuldig 'multiply-add' operaties plaats. Deze operatie is schematisch weergegeven in de figuur.



Uitleg: de waarden die in twee registers staan worden met elkaar vermenigvuldigd. Het product wordt opgeteld bij een waarde die in een ander register staat en het resultaat wordt teruggeplaatst in het laatstgenoemde register.

Geef AVR-code die deze operatie uitvoert. Kies hierbij zelf de registers.

---

4. Opgave 4

Gegeven het volgende AVR-programma. Alle instructies kosten 1 klokcyclus behalve BRNE. Als naar een ander adres dan het volgende wordt gesprongen, kost de BRNE-instructie 2 klokcycli, anders 1.

```
LDI R17, $03
LDI R18, $02
LDI R19, $01
ADD R18,R18
DEC R17
MOV R20, R17
SUB R20, R19
BRNE  -5
```

(a) Wat staat er na afloop in registers R17, R18 en R19? Laat zien hoe je hierop komt. 15

(b) Hoeveel klokcycli duurt dit programma? Leg uit. 10

(c) Stel dat de eerste instructie wordt veranderd in `LDI R17, 103` (merk op dat die 103 decimaal is), hoeveel klokcycli gaat het programma dan duren? 10

Einde van deze toets.