DIGTEC/2021-2022

Jesse op den Brouw

DIGTEC

Two’s complement

DE HAAGSE HOGESCHOOL

Negatieve getallen

- Tot nu toe zijn alleen positieve getallen (en nul) behandeld.
- Getallen kunnen echter ook negatief zijn, bijvoorbeeld om een tekort aan te geven (banksaldo) of een richting (negatieve stroom).
- In het alledaags gebruik wordt de teken-en-grootte-representatie (*signed magnitude*) gebruikt.
- Er is een extra teken (het ‘–’-teken) nodig in het bekende decimale talstelsel.

Signed magnitude

- In de signed magnitude representatie bestaat een getal uit een *grootte* en een *teken* dat aangeeft of het een positief of negatief getal is.
- Dus +13 en –13 zijn even groot (magnitude) maar tegengesteld.
- Er zijn twee weergaven van nul, +0 en –0, beide met dezelfde waarde.
- Een getal zonder ‘+’ wordt als positief beschouwd.

Signed magnitude

- Het *optellen* van twee signed magnitude getallen kost enige moeite:
- Als de twee getallen hetzelfde teken hebben, moeten de grootten worden opgeteld en krijgt het antwoord hetzelfde teken.
- Als de twee getallen verschillende tekens hebben, moet de kleinste grootte worden afgetrokken van de grootste grootte, en het antwoord krijgt het teken van de grootste grootte.
- Al met al een hoop ‘als’, ‘optellen’, ‘aftrekken’, ‘vergelijken’.

Negatieve getallen

- Hieronder vier voorbeelden:

+321	–456	+321	+231
+244	–123	–132	–342
+565	–579	+189	↓
			+342
			–231
			+111
			↓
			–111

- Er zijn nog meer voorbeelden te bedenken.

Negatieve getallen

- Al dit ‘als’, ‘optellen’, ‘aftrekken’, ‘vergelijken’ levert een grote hoeveelheid digitale schakelingen op.

- Eén pluspunt: als bekend is hoe een signed magnitude opteller gebouwd moet worden, dan is een signed magnitude aftrekker erg simpel.

- Slimmer is om gebruik te maken van een *complement* (of *radix-complement)* representatie.

- Complement representaties zijn gebaseerd op *modulair rekenen.*

Modulair rekenen

- In digitale systemen worden getallen opgeslagen in een eindig aantal bits. Er zijn dus maar een eindig aantal getallen weer te geven.
- Zo zijn met 4 bits de getallen 0 t/m $2^4 - 1$ (0 t/m 15) weer te geven. Er zijn 2^4 (16) combinaties (verschillende getallen) mogelijk.
- Een optelling van twee getallen van 4 bits levert een resultaat van 5 bits, maar het 5^e bit wordt genegeerd.
- Alle berekeningen zijn *modulo* 2^4. Er zijn 2^4 combinaties mogelijk.

Hoe bepalen we de waarde -1

Bekijk de volgende binaire optelling:

```
  0001
  ???? +
  0000
```

Welk getal moet op de vraagtekens staan om 0000_2 te krijgen?

```
  0001
  1111 +
1 0000
```

Merk op: 1111_2 is $16 - 1$ ($= 2^4 - 1$)

Hoe bepalen we de waarde -2

Bekijk de volgende binaire optelling:

```
  0010
  ???? +
  0000
```

Welk getal moet op de vraagtekens staan om 0000_2 te krijgen?

```
  0010
  1110 +
1 0000
```

Merk op: 1110_2 is $16 - 2$ ($= 2^4 - 2$)

Hoe bepalen we de waarde -3

Bekijk de volgende binaire optelling:

```
  0011
  ???? +
  0000
```

Welk getal moet op de vraagtekens staan om 0000_2 te krijgen?

```
  0011
  1101 +
1 0000
```

Merk op: 1101_2 is 16 – 3 (= 2^4 – 3)

Modulair rekenen

- We zien een patroon ontstaan.
- Om bij een positief getal m de negatieve representant te krijgen, berekenen we $2^4 - m$.
- Dus $m + (2^4 - m) = 0$ (?)
- Het resultaat is 2^4 te groot. Dat moet er nog afgetrokken worden.
- Het *aantal* getallen is begrensd.

Two’s complement

- Deze techniek wordt in hardware gebruikt (en dus ook in een computer).
- Deze techniek wordt de two’s complement representatie genoemd.
- Dus $m + (2^4 - m) = 0 \mod 2^4$

  two’s complement representatie van $-m$
- Dus het resultaat is de *rest na deling* door 2^4.

De waarde -5 in two's complement

Bekijk de volgende binaire optelling:

```
  0101
  ???? +
  0000
```

Welk getal moet op de vraagtekens staan om 0000_2 te krijgen?

```
  0101
  1011 +
1 0000
```

Dus: 1011_2 is de *two's complement representatie* van –5.

Two's complement

- Een positief getal wordt negatief gemaakt door het positieve getal af te trekken van een macht van 2, bijvoorbeeld 2^4 (16, 4-bits getal).
- Als voorbeeld +5 → – 5

$$\begin{array}{r} 10000_2 \\ 0101_2 \\ \hline 1011_2 \end{array} \quad - \qquad \begin{array}{r} 2^4 \\ +5 \\ \hline 2^4 - 5 \end{array} \quad -$$

- Het binaire getal 1011_2 is de *representatie* van -5_{10}. Merk op dat de getallen *modulo* 2^4 zijn!

Two's complement

- We kunnen nu de getallen gebruiken.
- Als voorbeeld: +5 + (–5) = 0

0101		+5
1011	+	$2^4 - 5$
1) 0000		$2^4 + 0$
↓		
0000		0

- De uitgaande carry moet geschrapt worden, die is het resultaat van de wijze waarop de two's complement representatie werkt.

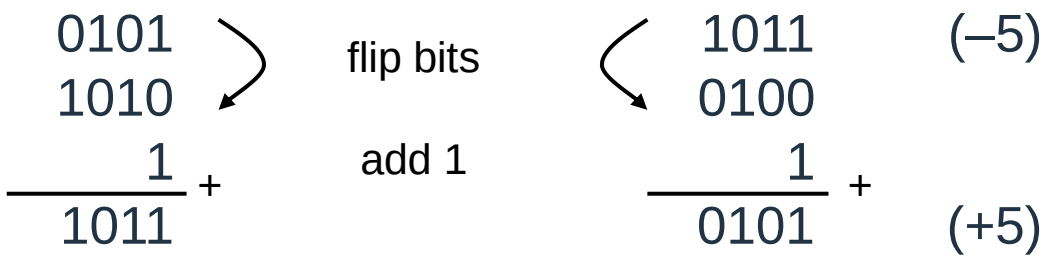
Two's complement

- Bij het omkeren van het teken is het handiger om uit te gaan van $(2^4 – 1) + 1$, want $(2^4 – 1)$ levert allemaal 1-en op!
- Er moet dan nog wel 1 bij opgeteld worden.
- Dus +5 → –5:

1111	-	
0101		+5
1010		
1	+	
1011		–5

Two's complement

- Een binair getal aftrekken van 1...1 is heel gemakkelijk.
- Het antwoord is namelijk de inverse van de afzonderlijke bits van dat getal (*flip bits*). Daarna moet er 1 bij opgeteld worden (*add 1*). Dit werkt ook van negatief naar positief.
- Als voorbeeld +5 → –5 en –5 → +5



DE HAAGSE HOGESCHOOL

Two's complement

- We willen graag een evenwichtige verdeling van de positieve en negatieve representaties. De helft van de combinaties representeren positieve getallen (inclusief 0) en de andere helft representeert de negatieve getallen.

- Het bereik van een getal *M* gerepresenteerd met 4 bits ligt als volgt:

- $-8 \leq M \leq +7$ (0 wordt positief als beschouwd)

- We zullen later zien dat niet alle resultaten van rekenkundige bewerkingen in het bereik van de representatie van 4 bits (of algemeen: *n* bits) liggen. Er is dan sprake van *overflow*.

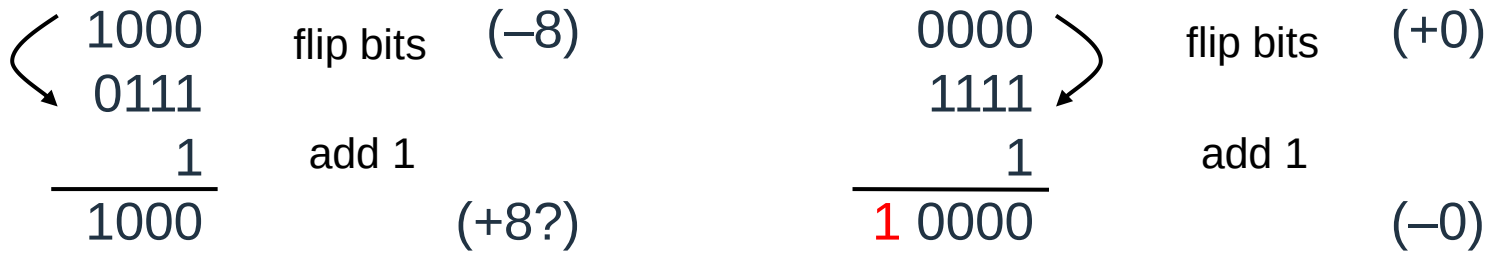
Two’s complement

Hieronder een lijst van de getallen –8 t/m +7.

dec	bits	two's com	
+7	0111	7	
+6	0110	6	
+5	0101	5	
+4	0100	4	
+3	0011	3	
+2	0010	2	
+1	0001	1	
0	0000	0	
–1	1111	15	$= 2^4-1$
–2	1110	14	$= 2^4-2$
–3	1101	13	$= 2^4-3$
–4	1100	12	$= 2^4-4$
–5	1011	11	$= 2^4-5$
–6	1010	10	$= 2^4-6$
–7	1001	9	$= 2^4-7$
–8	1000	8	$= 2^4-8$

Two’s complement

- Er is geen tegengesteld getal voor –8. Er is slechts één representatie van 0.



- Two’s complement heeft een asymmetrisch bereik.

DE HAAGSE HOGESCHOOL

Two’s complement

- Het bereik van two's complement getallen is als volgt:

4 bits	–8	≤ N ≤	+7
8 bits	–128	≤ N ≤	+127
16 bits	–32768	≤ N ≤	+32767
32 bits	–2147483648	≤ N ≤	+2147483647
n bits	-2^{n-1}	≤ N ≤	$+2^{n-1} - 1$

- Er is geen tegengesteld getal voor -2^{n-1}. Het bereik van two's complement getallen is asymmetrisch.

Two’s complement

- Een two's complement getal is positief als het meest significante bit 0 is.
- Een two's complement getal is negatief als het meest significante bit 1 is.
- Het getal 0 wordt dus als positief gezien.
- Het meest significante bit wordt het *tekenbit* genoemd.
- Merk op dat het tekenbit ook een *gewicht* heeft, zie verderop.

Two’s complement

- Een two's complement getal kan met meer bits geschreven worden dan oorspronkelijk door middel van *tekenuitbreiding* (sign extension).
- Het tekenbit moet gekopieerd worden naar de nieuw toe te voegen bits.
- Positieve getallen worden aangevuld met 0-en.
- Negatieve getallen worden aangevuld met 1-en.

Two’s complement

- Voorbeeld van een positief en negatief getal:

+6	0110	00000110
–6	1010	11111010

- Omrekenen:

+6	00000110	
	11111001	flip bits
	+1	add 1
–6	11111010	

Two's complement

- Het omzetten van een two's complement naar decimaal kan eenvoudig door het tekenbit als *negatief* gewicht te gebruiken. De overige bits hebben hetzelfde gewicht als bij unsigned.

- Voorbeeld: $1011 \rightarrow -1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = -8 + 3 = -5$

  $0110 \rightarrow 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 0 + 6 = 6$

Two’s complement

- Decimaal naar two’s complement:

- Positief getal: omzetten als bij unsigned (herhaald delen door 2)

- Negatief getal:
  - Bepaal aantal bits, bv. 8
  - Trek getal af van 2^8
  - Zet uitkomst om alsof het unsigned is.

- Dus -57 met 8 bits: $2^8 – 57 = 199$
  - Nu 199 omzetten als unsigned (met 8 bits). Levert mogelijk leidende 1-en.

Unsigned versus two's complement

- Hoe de getallen gebruikt/gelezen moeten worden, hangt af van de *interpretatie* van de gebruiker. Voor de hardware is er geen verschil*.

unsigned	bits	two's compl.	dec
+15	1111	$2^4 - 1$	–1
+15 +	1111 +	$2^4 - 1$ +	–1 +
+16 +14	1 1110	$2^4 + 2^4 - 2$	–2
↓	↓	↓	↓
+14	1110	$2^4 - 2$	–2

* En dat was ook de bedoeling!

Aftrekken

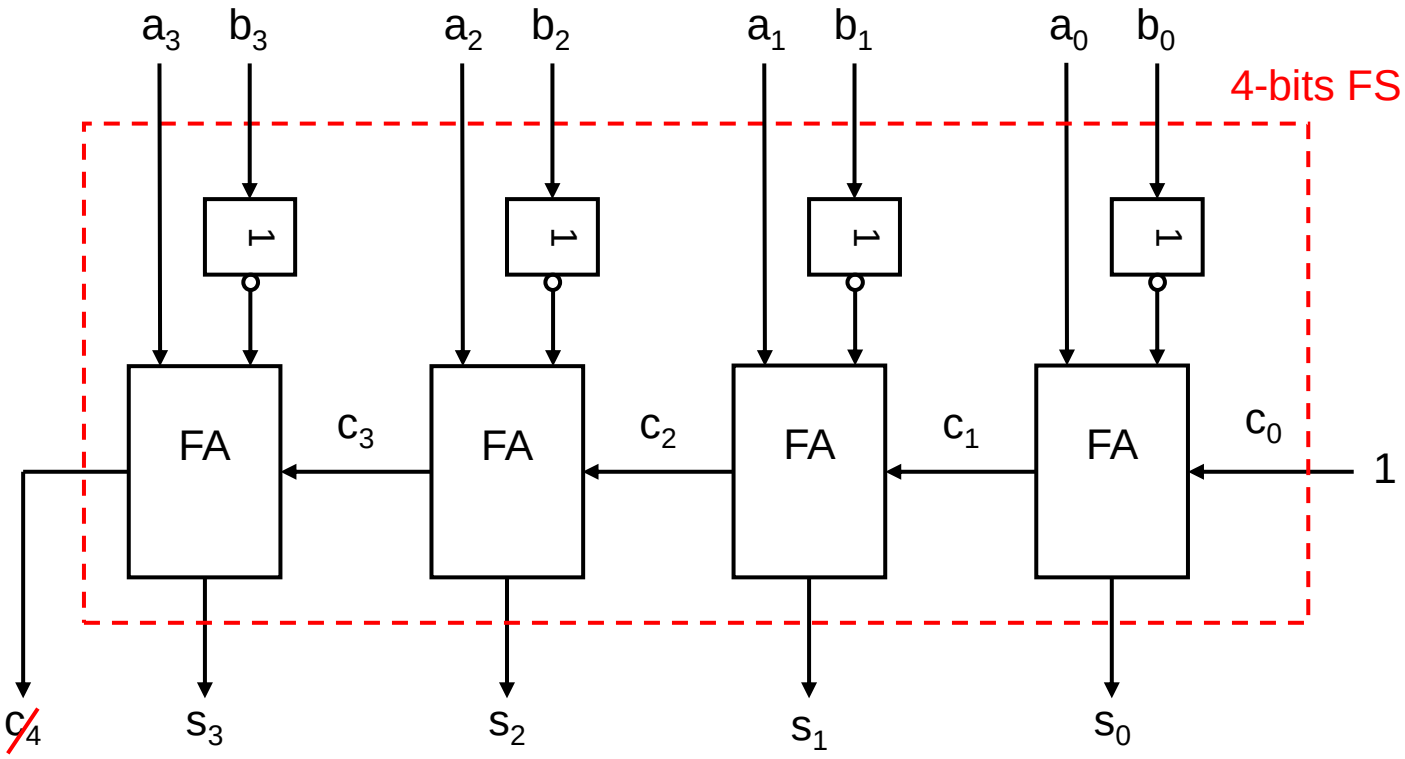
- Het ontwerpen van een aftrekschakeling kan op vergelijkbare wijze als een optelschakeling.
- Het is echter veel handiger om gebruik te maken van de gelijkheid:

$$A - B = A + (-B)$$

- Nu kan een normale full adder gebruikt worden en is alleen een *negator* (schakeling die het getal negatief maakt) nodig.
- Het negatief maken van een two's complement getal gebeurt door de afzonderlijke bits te inverteren en daarna het getal 1 er bij op te tellen.

Two's complement

- Het werking van de negator kan worden opgenomen in een gewone full adder. Dit levert een aftrekschakeling (full subtractor).



DE HAAGSE HOGESCHOOL

Optel/aftrekschakeling

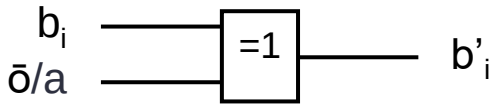
- Een optelschakeling kan gecombineerd worden met een aftrekschakeling.
- Een (nieuw) besturingssignaal ō/a geeft aan op welke operatie uitgevoerd moet worden.
- Als ō/a = 0 moet er worden opgeteld.
- Als ō/a = 1 moet er worden afgetrokken.

Optel/aftrekschakeling

- De operatie A – B moet worden omgezet in A + (–B) want dan kan een gewone full adder gebruikt worden.
- Voor aftrekken is dus –B nodig. Het negatief maken van B gebeurt door alle bits te inverteren en er dan 1 bij optellen (two's complement).
- Voor optellen is B nodig. Er hoeft niets te worden aangepast.
- Het (al dan niet) inverteren van de afzonderlijke bits van B kan gedaan worden door de omschakelbare buffer/NOT-poort.

Optel/aftrekschakeling

- De buffer/NOT-poort is de bekende EXOR!



b_i is het i^e bit van getal B

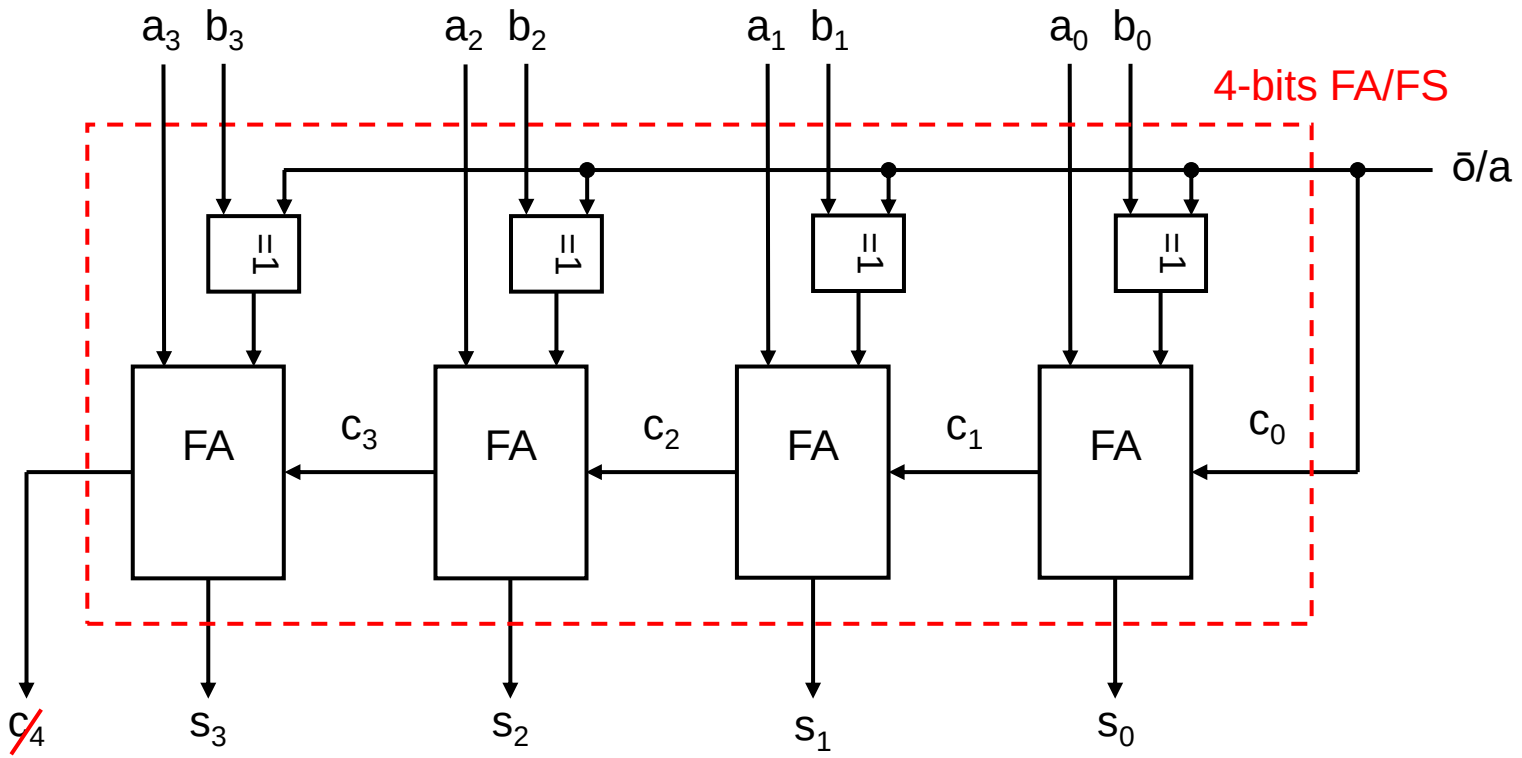
$$\left.\begin{array}{l} b'_i = b_i \quad \text{als} \quad \bar{o}/a = 0 \\ b'_i = \overline{b_i} \quad \text{als} \quad \bar{o}/a = 1 \end{array}\right\} \quad b'_i = \overline{\bar{o}/a} \cdot b_i + \bar{o}/a \cdot \overline{b_i}$$

- Bij optellen moet de c_0 aan een logische 0 verbonden zijn, dus als ō/a = 0.

- Bij aftrekken moet de c_0 aan een logische 1 verbonden zijn, dus als ō/a = 1.

Optel/aftrekschakeling

- Hieronder de complete optel/aftrekschakeling.



DE HAAGSE HOGESCHOOL

Overflow

- Bij het het optellen van twee two's complement getallen kan het zijn dat het antwoord buiten het bereik van de representatie komt te liggen.
- Er wordt dan gesproken van *overflow* (overstromen, overlopen).
- Overflow kan voorkomen als twee getallen van gelijk teken worden opgeteld.
- Getallen van verschillend teken kunnen nooit overflow opleveren.
- NB: overflow bij *unsigned getallen* is eenvoudig te zien aan de *uitgaande* carry.

Overflow

Enige rekenvoorbeelden

+5		0101		–8		1000	
+5	+	0101	+	–8	+	1000	+
+10		1010	(= -6)	–16		1 0000	(= 0)

–7		1001		+4		0100	
–4	+	1100	+	–8	+	1000	+
–11		1 0101	(= +5)	–4		1100	

De uitgaande 1 (carry!) moet genegeerd worden.

Overflow

- Overflow is gemakkelijk te detecteren. Overflow kan alleen gebeuren als twee getallen met gelijk teken worden opgeteld.
- Als de tekens van de op te tellen getallen anders zijn dan dat van het antwoord is er overflow.
- Er zijn twee 4-bit getallen A = $a_3a_2a_1a_0$ en B = $b_3b_2b_1b_0$.
- Het antwoord is de 4-bit som S = $s_3s_2s_1s_0$.
- Er is overflow als $(a_3 = b_3) \neq s_3$.

Overflow

- De functie voor overflow van een 4-bit full adder is

$$ov_{op} = a_3 \cdot b_3 \cdot \overline{s_3} + \overline{a_3} \cdot \overline{b_3} \cdot s_3$$

- De functie voor overflow van een 4-bit full subtractor is (zonder aan te tonen)

$$ov_{af} = a_3 \cdot \overline{b_3} \cdot \overline{s_3} + \overline{a_3} \cdot b_3 \cdot s_3$$

- De functie voor overflow is te combineren

$$ov_{opaf} = \overline{\overline{o}\,/\,a} \cdot \left(a_3 \cdot b_3 \cdot \overline{s_3} + \overline{a_3} \cdot \overline{b_3} \cdot s_3\right) + \overline{o}\,/\,a \cdot \left(a_3 \cdot \overline{b_3} \cdot \overline{s_3} + \overline{a_3} \cdot b_3 \cdot s_3\right)$$

Overflow

- Het valt aan te tonen dat overflow kan worden gedetecteerd door:

$$V = C_4 \oplus C_3$$

- Er is dus overflow als de inkomende carry van de tekenbit en de uitgaande carry van de tekenbit ongelijk zijn.

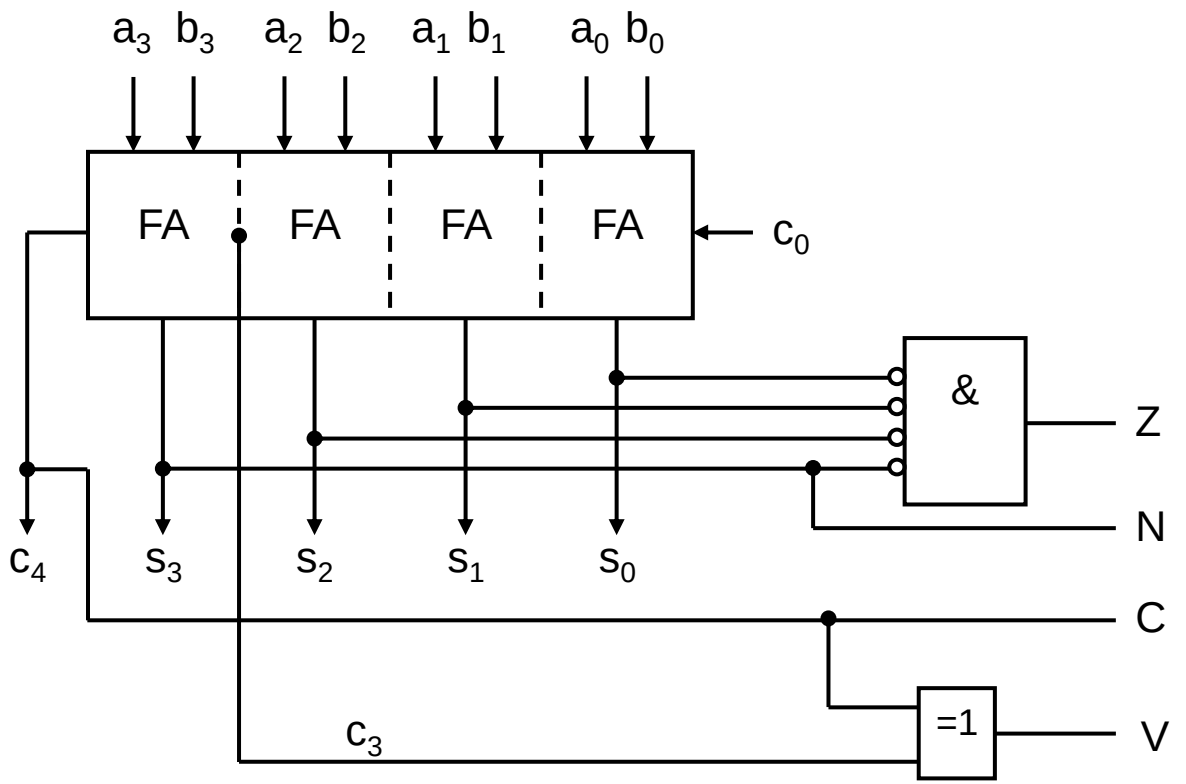
- Dit geldt zowel voor optellen als voor aftrekken.

Flags in een processor

- In elke microprocessor zit een rekeneenheid. Deze eenheid wordt doorgaans de ALU (*Arithmetic and Logic Unit*) genoemd, en kan naast optellen en aftrekken ook alle logische bewerkingen.
- Het detecteren van overflow is noodzakelijk om berekeningen betrouwbaar te laten verlopen.
- Daarnaast is het handig om te weten of een berekening 0 heeft opgeleverd of negatief is.
- Dit wordt weergegeven in de *flags*: oVerflow, Zero, Negative, Carry

Overflow

- Het detecteren van de zero-conditie is vrij simpel. Alle sombits moeten 0 zijn, dan is Z =1. De N is hetzelfde als het tekenbit. C levert 1 als er een carry uit de (totale) FA komt. N = 1 bij signed overflow.



DE HAAGSE HOGESCHOOL

Two's complement en hexadecimaal

- Hexadecimale getallen worden gebruikt om binaire getallen gecomprimeerd op te schrijven en kunnen dus ook negatief zijn.

- Hexadecimale getallen die beginnen met de cijfers 8 t/m F zijn negatief, want ze beginnen met een binaire 1.

  8 = 1000, 9 = 1001, A = 1010, B = 1011
  C = 1100, D = 1101, E = 1110, F = 1111

- Voorbeeld: $D0_{16} = 1101.0000_2 = -2^7 + 2^6 + 2^4 = -48_{10}$
  $FFFFFFFF_{16} = 11...11_2 = -2^{31} + 2^{30} + ... + 2^1 + 2^0 = -1_{10}$

Two’s complement en hexadecimaal

- We kunnen gebruik maken van two's complement representatie. Een hexadecimaal cijfer tussen 8 en F stelt een negatief getal voor. Dit geldt uiteraard alleen voor het meest significante cijfer.

- Dus: $8_{16} = -8_{10}$ $9_{16} = -7_{10}$ $A_{16} = -6_{10}$ $B_{16} = -5_{10}$
  $C_{16} = -4_{10}$ $D_{16} = -3_{10}$ $E_{16} = -2_{10}$ $F_{16} = -1_{10}$

- Voorbeeld:

$$C5_{16} = -4 \cdot 16 + 5 = -48 + 5 = -43_{10}$$
$$FFFF_{16} = -1 \cdot 16^3 + 15 \cdot 16^2 + 15 \cdot 16^1 + 15 \cdot 16^0 = -1_{10}$$

let’s change