

CS

WEEK 1-1

AGENDA

week	onderwerp	P&H	AT	Dijkstra
1	coderingen en talstelsels representatie van getallen optellen en aftrekken vermenigvuldigen en delen logische poorten schakelingen met poorten geheugen-elementen systeemklok & timers	App. B2, B3, B7, B8, B9 2.4 3.2 t/m 3.5 app B	App. A, B 3.1, 3.2, 3.3	H1 H2
2	typen computers 8 great ideas organisatie van de computer CPU intern, instructies uitvoeren geheugen systeem adres- en databus byte ordering pipelining de AVR MCU	1.1 t/m 1.4 2.12 4.1 t/m 4.5	1.3 2.1, 2.2 3.7	H3 6.1 en 6.2 7.1 en 7.2
3	typen geheugen caching opslag (ssd, harddisk) translating and starting a program parallele architecturen - h/w multi-threading - multicore - GPU	5.2, 5.3 6.4 t/m 6.6	2.2, 2.3 7.3, 7.4 H8	4.1 7.3

AGENDA

- **intro**
- representatie van gegevens in de computer
- talstelsels
- rekenen met binaire getallen
- two's complement formaat
- floating point formaat

BELANG VAN DIT VAK

CS is **voorwaardelijk** voor:

- Thema 2.2 NSE/SE Netwerken
- Thema 2.2 NSE/SE Operating Systems

Zie [OER](#)

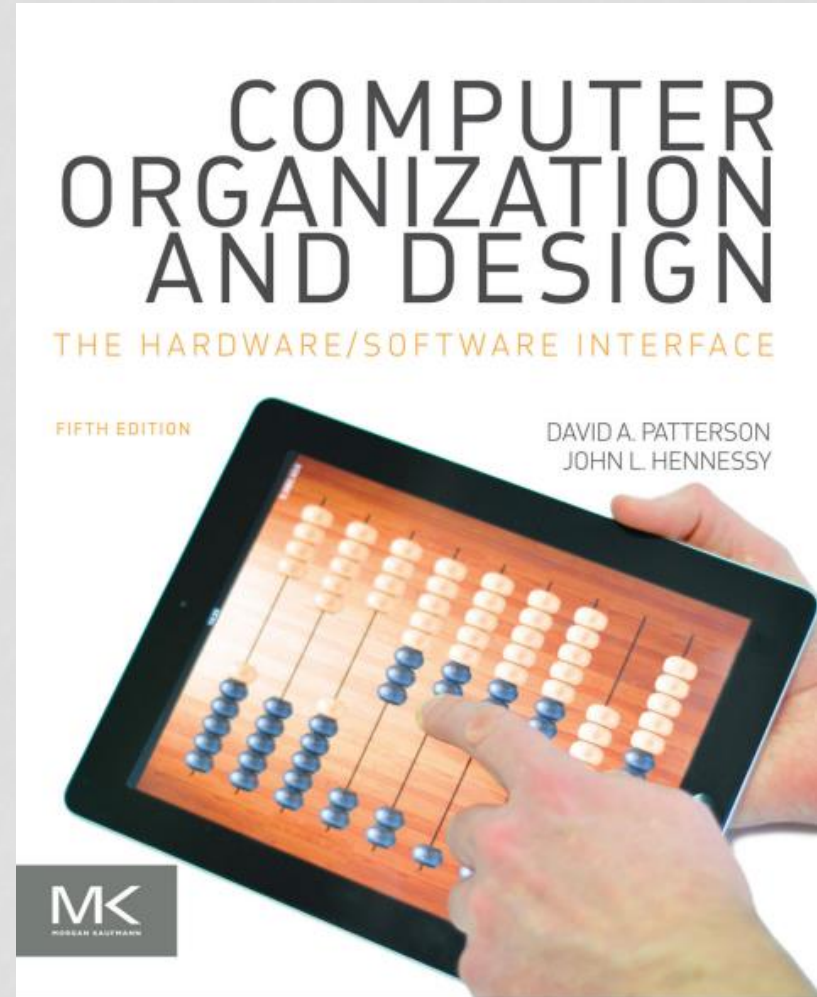
HUISHOUDELIJK

- Dit college:
 - niet verplicht
 - op tijd
 - mobiele telefoon uit/stil
 - aantekeningen: papier
- Laptop multitasking hinders classroom learning for both users and nearby peers,
<http://www.sciencedirect.com/science/article/pii/S0360131512002254?np=y>
- actieve deelname
- sheets: binnen 1 week op BlackBoard
- indeling : 45/10/45

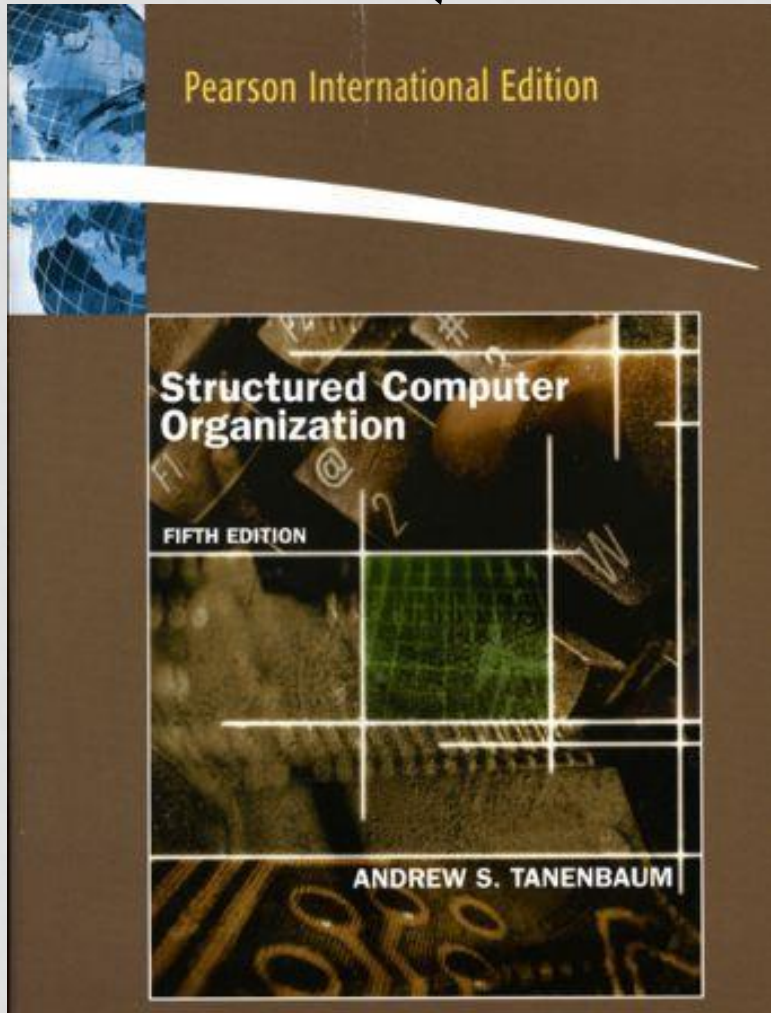
meest toegankelijk & eenvoudig, NL



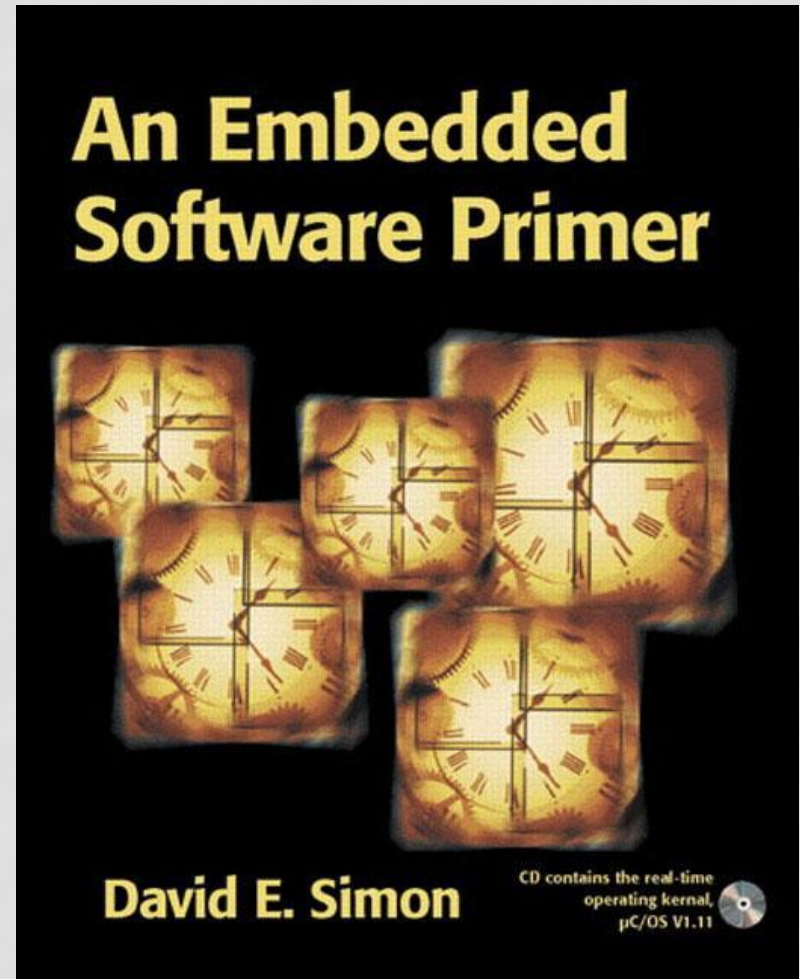
Stanford&Berkely
meest up-to-date en compleet
Hennessy ontwierp de MIPS CPU



best gestructureerd, beetje verouderd



inleiding : embedded systemen, hardware, scheduling, interrupts, real-time OS



AGENDA

- intro
- **representatie van gegevens in de computer**
- talstelsels
- rekenen met binaire getallen
- two's complement formaat
- floating point formaat



REPRESENTATIE

- binair = 2 waarden
 - betrouwbare opslag en communicatie
- rijtjes van '1'-nen en '0'-en
 - hoge of lage spanning
 - wel of geen putje in CD/DVD
 - magnetische oriëntatie (links of rechts)
- hoe gegevens (getallen, tekst, muziek, plaatje) opslaan in en verwerken met computer ?

BIT, BYTE EN WOORD

- bit : 0 of 1 ("binary digit")
 - eenheid van informatie
- byte = 8 bit
- nibble = 4 bit
- rijtje '1'-nen en '0'-en opslaan in "register"
 - breedte register = woord (vaak 32 of 64 bit)

HOEVEEL BITS ?

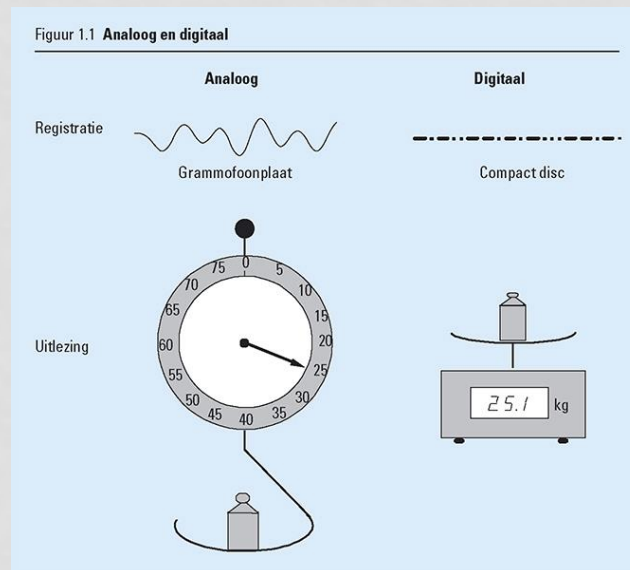
- hoeveel rijtjes '0' en '1' zijn er mogelijk met 2 bits ?
- hoeveel rijtjes '0' en '1' zijn er mogelijk met 5 bits ?
- hoeveel waarden kan ik coderen met een 5-bits register ?
- en met een 8-bits register ?
- en met een 16-bits register ?
- en met een n-bits register ?

STORAGE SIZES (IN BYTES)

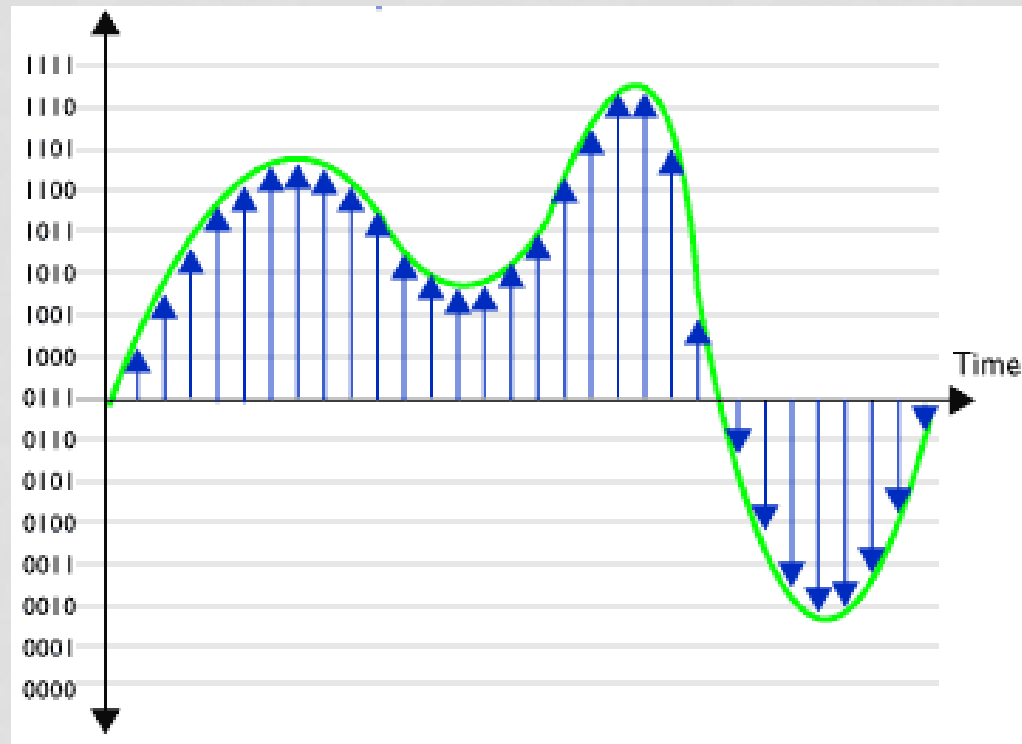
C Data Type	Typical 32-bit	Intel IA32	x86-64
char	1	1	1
short	2	2	2
int	4	4	4
long	4	4	8
long long	8	8	8
float	4	4	4
double	8	8	8
long double	8	10/12	10/16
pointer	4	4	8

ANALOOG EN DIGITAAL

- wereld om ons heen is analoog : licht, geluid, temperatuur, afstand, snelheid, druk, enz.
- wanneer meetwaarden (van sensor) als *getallen* worden weergegeven dan kan een computer ze verwerken
- ADC - Analog-to-Digital Converter
- voorbeeld : geluid opnemen, een plaatje scannen



ADC : SAMPLING (BEMONSTEREN)



DIGITAAL – ANALOOG?

- DAC = Digital-to-Analog Converter
- VOORBEELDEN?
 - www.dutchaudioclassics.nl/the_complete_d_a_dac_converter_list/

Part #	Manufacturer	Description	Stock	Price
PCM54HP	BB	Replaced by DAC7742 : 16-Bit Monolithic Digital-to-Analog Converter 28-PDIP	73072	1: \$3.55 10: \$3.02 100: \$2.84

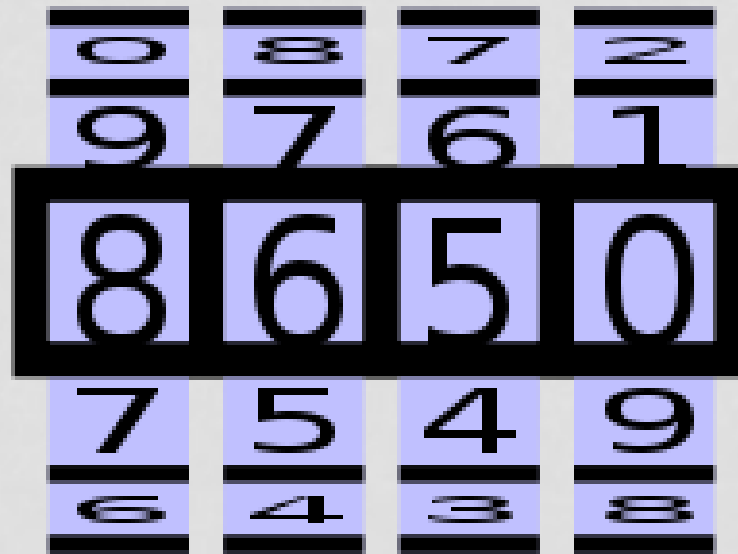
AGENDA

- intro
- representatie van gegevens in de computer
- **talstelsels**
- rekenen met binaire getallen
- two's complement formaat
- floating point formaat

DECIMALE STELSEL

- wat betekent eigenlijk 15671 ?
 - grondtal = 10
 - er zijn 10 symbolen
 - $1 \cdot 10^0 + 7 \cdot 10^1 + 6 \cdot 10^2 + 5 \cdot 10^3 + 1 \cdot 10^4$
 - als teller op **positie** $n > 9$ dan teller op positie $n+1$ met één verhogen
- heet : "positie-stelsel"
- hoe zou ik de cijfers 1, 7, 6, 5 en 1 kunnen *berekenen* ?
- wat betekent eigenlijk 15,671 ?

DECIMALE STELSEL



TALSTELSELS

- codering en notatie van getallen
- grondtal = 10 (decimaal)
- grondtal = 2 (binair)
- grondtal = 8 (octaal)
- grondtal = 16 (hexadecimaal)

BINAIRE STELSEL

- grondtal 2
- er zijn 2 symbolen
- wat betekent dan 100101 ?
- 100101 binair = ... decimaal ?
- 54 decimaal = ... binair ?

CONVERSIE VAN DECIMAAL NAAR BINAIR

Conversion of the decimal number 1492 to binary by *successive halving*, starting at the top and working downward. For example, 93 divided by 2 yields a quotient of 46 and a remainder of 1, written on the line below.

Quotients	Remainders
↓	↓
1492	
746	0
373	0
186	1
93	0
46	1
23	0
11	1
5	1
2	1
1	0
0	1

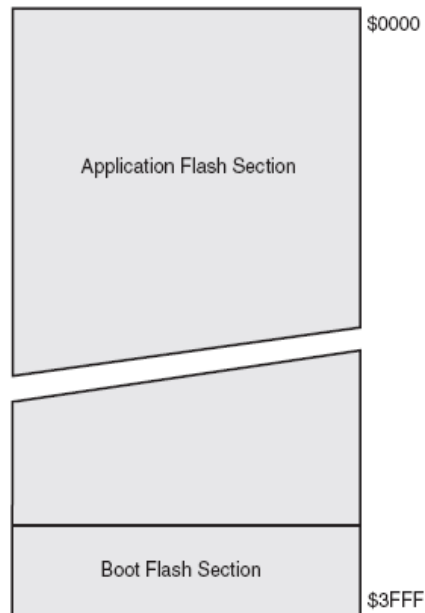
10111010100 = 1492₁₀

HEXADECIMALE STELSEL

- grondtal 16
- er zijn 16 symbolen : 0, 1, ..., 9, A, B, C, D, E, F
- wat betekent AEB7 ? (of : \$AEB7, 0xAEB7)
- AEB7 hex = ... decimaal ?
- 698 decimaal = ... hex ?
- van binair naar hexadecimaal en omgekeerd is eenvoudig (want $16 = 2^4$)
- 10110011 binair = ... hex ?

<i>decimaal</i>	<i>hex</i>	<i>binair</i>
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Figure 8. Program Memory Map



New Host [X]

Name (uses parent domain name if blank):

Fully qualified domain name (FQDN):

IP address:

☐ Create associated pointer (PTR) record

☐ Allow any authenticated user to update DNS records with the same owner name

OCTALE STELSEL

- grondtal 8
- er zijn 8 symbolen
- wat betekent 1356 ?
- 1356 octaal = ... decimaal ?
- 262 decimaal = ... octaal ?
- van binair naar octaal en omgekeerd is eenvoudig (want $8 = 2^3$)

Octal notation

Another common method for representing Unix permissions is *octal notation*. Octal notation consists of a three- or four-digit [base-8](#) value.

With three-digit octal notation, each numeral represents a different component of the permission set: user class, group class, and "others" class respectively.

Each of these digits is the sum of its component bits (see also [Binary numeral system](#)). As a result, specific bits add to the sum as it is represented by a numeral:

- The read bit adds 4 to its total (in binary 100),
- The write bit adds 2 to its total (in binary 010), and
- The execute bit adds 1 to its total (in binary 001).

These values never produce ambiguous combinations; each sum represents a specific set of permissions.

These are the examples from the [Symbolic notation](#) section given in octal notation:

- "-rwxr-xr-x" would be represented as 755 in three-digit octal.
- "-rw-rw-r--" would be represented as 664 in three-digit octal.
- "-r-x-----" would be represented as 500 in three-digit octal.

Here is a summary of the meanings for individual octal digit values:

```
0 --- no permission
1 --x execute
2 -w- write
3 -wx write and execute
4 r-- read
5 r-x read and execute
6 rw- read and write
7 rwx read, write and execute
```

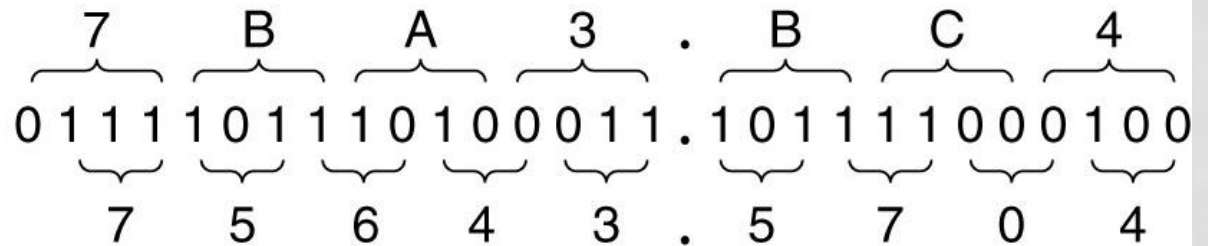
CONVERSION FROM ONE RADIX TO ANOTHER

Example 2

Hexadecimal

Binary

Octal

7	B	A	3	.	B	C	4
							
0111	1011	1010	0011	.	1011	1100	0100
7	5	6	4	.	5	7	04

AGENDA

- intro
- representatie van gegevens in de computer
- talstelsels
- **rekenen met binaire getallen**
- two's complement formaat
- floating point formaat

OPTELLEN

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 0 = 1$
- $1 + 1 = 0$ met carry

- $25 + 9 = ?$

$$\begin{array}{r}
 25 \\
 9 \\
 \hline
 34
 \end{array}
 +
 \begin{array}{r}
 11001 \\
 1001 \\
 \hline
 100010
 \end{array}
 +
 \begin{array}{r}
 111 \\
 \leftarrow \text{carry}
 \end{array}$$

OVERFLOW

- stel we hebben registers met een breedte van 4 bit
- $15 + 2 = 17$, dit past niet meer in het 4-bit register
- dit 'niet-meer-passen' noemen we **overflow**

VERMENIGVULDIGEN

- $0 \times 0 = 0$
- $0 \times 1 = 0$
- $1 \times 0 = 0$
- $1 \times 1 = 1$

DECIMAAL VERMENIGVULDIGEN

231 (<- B)
103 (A ->)
--- *

3 * 231

0 * 2310

1 * 23100

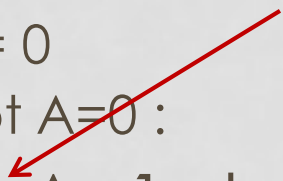
----- +

693 + 0 + 23100

25 X 13 BINAIR = ?

```
      11001      (<- B)
       1101      (A ->)
----- *
      1 * 11001
      0 * 110010
      1 * 1100100
      1 * 11001000
----- +
      101000101
```

BINAIR VERMENIGVULDIGEN

- binair vermenigvuldigen = **schuiven en optellen**
 - want het is 0x of 1x
 - $A \times B$:
 - resultaat = 0
 - herhaal tot $A=0$:
 - als **lsb** $A=1$ dan resultaat = resultaat + B
 - shift **left** B
 - shift **right** A
- lsb = least significant bit = meest rechtse bit*
- 

DECIMAAL DELEN

- voorbeeld : decimaal $1001010 : 1000 = 1001$ rest 10
- at every step
 - shift divisor right and compare it with current dividend
 - if divisor is **larger**, shift 0 as the next bit of the quotient
 - if divisor is **smaller**, subtract to get new dividend and shift 1 as the next bit of the quotient

	1001 _{ten}	Quotient
Divisor 1000 _{ten}	$ \begin{array}{r} \overline{1001010_{ten}} \\ -1000 \\ \hline 10 \\ 101 \\ 1010 \\ -1000 \\ \hline 10_{ten} \end{array} $	Dividend
		Remainder

AGENDA

- intro
- representatie van gegevens in de computer
- talstelsels
- rekenen met binaire getallen
- **two's complement formaat**
- floating point formaat

NEGATIEVE GETALLEN

- signed magnitude :
 - meest linkse bit is tekenbit
 - $+99 = 0110\ 0011$
 - $-99 = 1110\ 0011$
 - onhandig!
- one's complement :
 - negatieve waarde door alle bits te inverteren
 - $+99 = 0110\ 0011$
 - $-99 = 1001\ 1100$
 - 0000 0000 en 1111 1111 zijn **beide** 0
 - (vrijwel niet meer gebruikt)

NEGATIEVE GETALLEN

- two's complement :
 - negatieve waarde door alle bits te **inverteren** en er **1** bij op te tellen (*negeer overflow*)
 - **+99** = **0110 0011**
 - **-99** = **1001 1101**
 - 0 = 0000 0000; inverteren en 1 erbij op geeft zelfde representatie
 - voordeel : één representatie voor 0
 - voordeel : het optellen van negatieve getallen gaat net als optellen positieve getallen
 - nadeel : aantal negatieve getallen = aantal positieve getallen + 1

NEGATIEVE GETALLEN

- 8-bit two's complement van -60 ?
- $60_{10} = 0011\ 1100_2$
- reken uit in 4-bit : 3 - 5
- reken uit in 4-bit : -5 -5

- voorbeeld :
2's complement
met 4 bits

<i>decimaal</i>	<i>binair 2's c</i>
-8	1000
-7	1001
-6	1010
-5	1011
-4	1100
-3	1101
-2	1110
-1	1111
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Binary, inverse and complement codes



Number:

-11

Number of binary digits:

8

PLANET  CALC

Calculate

Range:	[-128,127]
Binary code:	00001011
Inverse code (one's complement):	11110100
Complement code (two's complement):	11110101

AGENDA

- intro
- representatie van gegevens in de computer
- talstelsels
- rekenen met binaire getallen
- two's complement formaat
- **floating point formaat**

FLOATING POINT FORMAT

```
float x = 0.4 - 0.3;  
float y = 0.3 - 0.2;  
if (x == y) {  
    printf ("gelijk");  
} else {  
    printf ("ongelijk");  
}
```

FLOATING POINT FORMAT

gebroken getallen :

$$\begin{aligned} + 0,000123 &= + 0,123 \times 10^{-3} \\ - 3450000 &= - 0,345 \times 10^{+7} \end{aligned}$$

mantisse *exponent*

Figuur 1.3 IEEE 32-bit floating-point formaat

Teken	Exponent	Mantisse	
1	8	23	bits

Teken : 0 is positief, 1 is negatief.

Exponent : loopt van -126 tot +127 (excess code).

Mantisse : is genormaliseerd tot het eerste bit een 1 is. Dit wordt niet genoteerd (dit heet een *hidden bit*). De complete mantisse is dus 24 bits groot.

FLOATING POINT FORMAT

- exponent kan positief of negatief zijn
- een bias (offset) van 127 wordt voor opslag opgeteld bij de exponent, dus :
 - -126 wordt genoteerd als $-126+127=1=0000\ 0001$
 - +127 wordt genoteerd als $127+127=254=1111\ 1110$
- Hoe 0?
 - Mantisse = 0 en exponent = 0
- Hoe oneindig?
 - Mantisse = 0 en exponent = 255

VOORBEELD DIJKSTRA OPG. 1.32A

Wat is 43,75 in floating point formaat ?

- 43 is binair 101011
- $0,75 = 3/4 = 1/2 + 1/4 = 0,11$
- samen geeft dit 101011,11
- **normaliseren** (komma opschuiven) geeft $1,0101111 * 2^5$
- het tekenbit = 0 (positief)
- exponent : bias 127 erbij = $5 + 127 = 132$ is binair 1000 0100
- mantisse : 0101111 (eerste bit weglaten)

teken	exponent	mantisse
0	1000 0100	0101 111 0.....0