# Repræsentation af tal

DM573

Rolf Fagerberg

# Mål

Målet for disse slides er at beskrive, hvordan tal repræsenteres som bitmønstre i computere.

# Bitmønstre

Information = valg mellem forskellig muligheder.

Simpleste situation: valg mellem to muligheder. Kald dem 0 og 1. Denne valgmulighed kaldes en bit.

# Bitmønstre

Information = valg mellem forskellig muligheder.

Simpleste situation: valg mellem to muligheder. Kald dem 0 og 1. Denne valgmulighed kaldes en bit.

Relevante for computere fordi to-delte valg er nemmest at repræsentere rent fysisk (1 = strøm, 0 = ikke strøm).

# Bitmønstre

Information = valg mellem forskellig muligheder.

Simpleste situation: valg mellem to muligheder. Kald dem 0 og 1. Denne valgmulighed kaldes en bit.

Relevante for computere fordi to-delte valg er nemmest at repræsentere rent fysisk (1 = strøm, 0 = ikke strøm).

Større samling information: brug flere bits:

$$\boxed{01101011}\,000110010101011011\ldots$$

F.eks. 8 bits (= 1 byte): valg mellem $2^8 = 256$ muligheder.

# Bitmønstre

Bitmønstre skal *fortolkes* for at have en betydning.

$$01101011 = ?$$

Der er brug for et system, som angiver, hvilke mening de forskellige bitmønstre skal tillægges.

# Bitmønstre

Bitmønstre skal *fortolkes* for at have en betydning.

$$01101011 = ?$$

Der er brug for et system, som angiver, hvilke mening de forskellige bitmønstre skal tillægges.

Der er lavet sådanne systemer for f.eks.:

- Tal (heltal, kommatal)
- Bogstaver
- Pixels (billedfil)
- Amplitude (lydfil)
- Computerinstruktion (program)
- $\vdots$

# Bitmønstre

Bitmønstre skal *fortolkes* for at have en betydning.

$$01101011 = ?$$

Der er brug for et system, som angiver, hvilke mening de forskellige bitmønstre skal tillægges.

Der er lavet sådanne systemer for f.eks.:

- Tal (heltal, kommatal)
- Bogstaver
- Pixels (billedfil)
- Amplitude (lydfil)
- Computerinstruktion (program)
- $\vdots$

Fokus i dag: systemer for heltal og kommatal.

# Talsystemer

Tital-systemet:

4532

# Talsystemer

Tital-systemet:

$$4532 \quad = \quad 4 \cdot 1000 \quad + \quad 5 \cdot 100 \quad + \quad 3 \cdot 10 \quad + \quad 2 \cdot 1$$

# Talsystemer

Tital-systemet:

$$\begin{aligned} 4532 &= 4 \cdot 1000 + 5 \cdot 100 + 3 \cdot 10 + 2 \cdot 1 \\ &= 4 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0 \end{aligned}$$

# Talsystemer

Tital-systemet:

$$\begin{aligned} 4532 &= 4 \cdot 1000 + 5 \cdot 100 + 3 \cdot 10 + 2 \cdot 1 \\ &= 4 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0 \end{aligned}$$

Grundtal: 10
Cifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (fordi $10 \cdot 10^i = 10^{i+1}$)

# Talsystemer

Tital-systemet:

$$
\begin{aligned}
4532 &= 4 \cdot 1000 + 5 \cdot 100 + 3 \cdot 10 + 2 \cdot 1 \\
&= 4 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0
\end{aligned}
$$

Grundtal: 10

Cifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (fordi $10 \cdot 10^i = 10^{i+1}$)

Syvtal-systemet:

$$
\begin{aligned}
4532_7 &= 4 \cdot 7^3 + 5 \cdot 7^2 + 3 \cdot 7^1 + 2 \cdot 7^0 \\
&= 4 \cdot 343 + 5 \cdot 49 + 3 \cdot 7 + 2 \cdot 1 \\
&= 1640
\end{aligned}
$$

# Talsystemer

Tital-systemet:

$$
\begin{aligned}
4532 &= 4 \cdot 1000 + 5 \cdot 100 + 3 \cdot 10 + 2 \cdot 1 \\
&= 4 \cdot 10^3 + 5 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0
\end{aligned}
$$

Grundtal: 10
Cifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 (fordi $10 \cdot 10^i = 10^{i+1}$)

Syvtal-systemet:

$$
\begin{aligned}
4532_7 &= 4 \cdot 7^3 + 5 \cdot 7^2 + 3 \cdot 7^1 + 2 \cdot 7^0 \\
&= 4 \cdot 343 + 5 \cdot 49 + 3 \cdot 7 + 2 \cdot 1 \\
&= 1640
\end{aligned}
$$

Grundtal: 7
Cifre: 0, 1, 2, 3, 4, 5, 6 (fordi $7 \cdot 7^i = 7^{i+1}$)

# Total-systemet

$$\begin{aligned}
1011_2 &= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\
&= 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 \\
&= 11
\end{aligned}$$

Grundtal: 2

Cifre: 0, 1 (fordi $2 \cdot 2^i = 2^{i+1}$)

# Total-systemet

$$\begin{aligned} 1011_2 &= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 \\ &= 11 \end{aligned}$$

Grundtal: 2
Cifre: 0, 1 (fordi $2 \cdot 2^i = 2^{i+1}$)

Relevante for computere fordi to-delte valg er nemmest at repræsentere rent fysisk (1 = strøm, 0 = ikke strøm).

Total-systemet kaldes også det *binære talsystem*.

Det giver en naturlig fortolkning af bitmønstre som ikke-negative hele tal.

# Hexadecimalt talsystem

Også brugt i datalogi er 16-tal-systemet:

$$\begin{aligned}
4A3F_{16} &= 4 \cdot 16^3 + 10 \cdot 16^2 + 3 \cdot 16^1 + 15 \cdot 16^0 \\
&= 4 \cdot 4096 + 10 \cdot 256 + 3 \cdot 16 + 15 \cdot 1 \\
&= 19007
\end{aligned}$$

Grundtal: 16

Cifre: 0, 1, 2, 3, ..., 9, A (=10), B (=11), ..., F (=15)

(fordi $16 \cdot 16^i = 16^{i+1}$)

# Hexadecimal notation

16-tals systemet kan også bruges som en simpel/kort måde at beskrive bitstrenge. Gruppér bits i grupper af 4 (dvs. 16 forskellige muligheder):

$\boxed{0110}\,\boxed{1010}\,\boxed{1110}\,01\ldots$

# Hexadecimal notation

16-tals systemet kan også bruges som en simpel/kort måde at beskrive bitstrenge. Gruppér bits i grupper af 4 (dvs. 16 forskellige muligheder):

$\boxed{0110}\boxed{1010}\boxed{1110}01\ldots$

Brug de 16 cifre til at beskrive disse muligheder:

| | | | |
|---|---|---|---|
| 0111 | 7 | 1111 | F |
| 0110 | 6 | 1110 | E |
| 0101 | 5 | 1101 | D |
| 0100 | 4 | 1100 | C |
| 0011 | 3 | 1011 | B |
| 0010 | 2 | 1010 | A |
| 0001 | 1 | 1001 | 9 |
| 0000 | 0 | 1000 | 8 |

$\boxed{0110}\boxed{1010}\boxed{1110}01\ldots = \texttt{6AE}\ldots$

# Addition

Addition fungerer ens i alle talsystemer, blot med grundtal udskiftet.

Tital-systemet:

$$
\begin{array}{r}
{\scriptstyle 1\,1\,1\,1\phantom{0\,0}} \\
5432 \\
+96781 \\
\hline
=102213
\end{array}
$$

# Addition

Addition fungerer ens i alle talsystemer, blot med grundtal udskiftet.

Tital-systemet:

$$\begin{array}{r} {\scriptstyle 1\,1\,1\,1\phantom{0}} \\ 5432 \\ +96781 \\ \hline =102213 \end{array}$$

Total-systemet:

$$\begin{array}{r} {\scriptstyle 1\,1\,1\phantom{00}} \\ 1110_2 \\ +11100_2 \\ \hline =101010_2 \end{array}$$

# Addition

Addition fungerer ens i alle talsystemer, blot med grundtal udskiftet.

Tital-systemet:

$$\begin{array}{r} {\scriptstyle 1\,1\,1\,1\phantom{0}} \\ 5432 \\ +96781 \\ \hline =102213 \end{array}$$

Total-systemet:

$$\begin{array}{r} {\scriptstyle 1\,1\,1\phantom{00}} \\ 1110_2 \\ +11100_2 \\ \hline =101010_2 \end{array}$$

Subtraktion, multiplikation, division fungerer også ens. F.eks.

$$\begin{array}{rl} 1010_2 \cdot 1110_2 = 10001100_2 & \text{(Check: } 10 \cdot 14 = 140\text{)} \\ 1101011_2 : 101_2 = 10101_2, \text{ rest } 10_2 & \text{(Check: } 107 : 5 = 21, \text{ rest } 2\text{)} \end{array}$$

# Konvertering mellem talsystemer

*Fra* andre grundtal: brug definitionen af talsystemer.

$$\begin{aligned}
1011_2 &= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\
&= 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 \\
&= 11
\end{aligned}$$

$$\begin{aligned}
4532_7 &= 4 \cdot 7^3 + 5 \cdot 7^2 + 3 \cdot 7^1 + 2 \cdot 7^0 \\
&= 4 \cdot 343 + 5 \cdot 49 + 3 \cdot 7 + 2 \cdot 1 \\
&= 1640
\end{aligned}$$

# Konvertering mellem talsystemer

*Fra* andre grundtal: brug definitionen af talsystemer.

$$\begin{aligned} 1011_2 &= 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 1 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1 \\ &= 11 \end{aligned}$$

$$\begin{aligned} 4532_7 &= 4 \cdot 7^3 + 5 \cdot 7^2 + 3 \cdot 7^1 + 2 \cdot 7^0 \\ &= 4 \cdot 343 + 5 \cdot 49 + 3 \cdot 7 + 2 \cdot 1 \\ &= 1640 \end{aligned}$$

*Til* andre grundtal: brug gentagen heltalsdivision. Husk hvordan heltalsdivision fungerer:

| Heltalsdivision | Kvotient | Rest | Som ligning |
|---|---|---|---|
| $31:7$ | 4 | 3 | $31 = 7 \cdot 4 + 3$ |
| $25:2$ | 12 | 1 | $25 = 2 \cdot 12 + 1$ |

# Konvertering mellem talsystemer

*Fra* andre grundtal: brug definitionen af talsystemer.

$$\begin{aligned}
1011_2 &= 1\cdot 2^3 + 0\cdot 2^2 + 1\cdot 2^1 + 1\cdot 2^0\\
&= 1\cdot 8 + 0\cdot 4 + 1\cdot 2 + 1\cdot 1\\
&= 11
\end{aligned}$$

$$\begin{aligned}
4532_7 &= 4\cdot 7^3 + 5\cdot 7^2 + 3\cdot 7^1 + 2\cdot 7^0\\
&= 4\cdot 343 + 5\cdot 49 + 3\cdot 7 + 2\cdot 1\\
&= 1640
\end{aligned}$$

*Til* andre grundtal: brug gentagen heltalsdivision. Husk hvordan heltalsdivision fungerer:

| Heltalsdivision | Kvotient | Rest | Som ligning |
|---|---|---|---|
| $31:7$ | 4 | 3 | $31 = 7\cdot 4 + 3$ |
| $25:2$ | 12 | 1 | $25 = 2\cdot 12 + 1$ |

Detaljer for grundtal to: næste side.

# Konvertering til binært talsystem

Følgende algoritme finder cifrene *fra højre til venstre* i den binære representation af et positivt heltal $N$:

$X = N$
Sålænge $X > 0$ gentag:
  Næste ciffer $=$ rest ved heltalsdivision $X:2$
  $X =$ kvotient ved heltalsdivision $X:2$

# Konvertering til binært talsystem

Følgende algoritme finder cifrene *fra højre til venstre* i den binære representation af et positivt heltal $N$:

$X = N$
Sålænge $X > 0$ gentag:
  Næste ciffer $=$ rest ved heltalsdivision $X : 2$
  $X =$ kvotient ved heltalsdivision $X : 2$

Eksempel: $N = 25$:

| Heltalsdivision | Kvotient | Rest |
|---|---|---|
| $25 : 2$ | 12 | 1 |
| $12 : 2$ | 6 | 0 |
| $6 : 2$ | 3 | 0 |
| $3 : 2$ | 1 | 1 |
| $1 : 2$ | 0 | 1 |

$25 = 11001_2$

# Hvorfor virker det?

| Heltalsdivision | Kvotient | Rest |
|---|---|---|
| $25:2$ | 12 | 1 |
| $12:2$ | 6 | 0 |
| $6:2$ | 3 | 0 |
| $3:2$ | 1 | 1 |
| $1:2$ | 0 | 1 |

$$25 = 11001_2$$

# Hvorfor virker det?

| Heltalsdivision | Kvotient | Rest |
|---|---|---|
| $25:2$ | 12 | 1 |
| $12:2$ | 6 | 0 |
| $6:2$ | 3 | 0 |
| $3:2$ | 1 | 1 |
| $1:2$ | 0 | 1 |

$$25 = 11001_2$$

$$25 = 2 \cdot 12 + 1$$

# Hvorfor virker det?

| Heltalsdivision | Kvotient | Rest |
|---|---|---|
| $25 : 2$ | 12 | 1 |
| $12 : 2$ | 6 | 0 |
| $6 : 2$ | 3 | 0 |
| $3 : 2$ | 1 | 1 |
| $1 : 2$ | 0 | 1 |

$$25 = 11001_2$$

$$\begin{aligned} 25 &= 2 \cdot 12 + 1 \\ &= 2(2 \cdot 6 + 0) + 1 \end{aligned}$$

# Hvorfor virker det?

| Heltalsdivision | Kvotient | Rest |
| --- | --- | --- |
| $25:2$ | 12 | 1 |
| $12:2$ | 6 | 0 |
| $6:2$ | 3 | 0 |
| $3:2$ | 1 | 1 |
| $1:2$ | 0 | 1 |

$$25 = 11001_2$$

$$\begin{aligned} 25 &= 2 \cdot 12 + 1 \\ &= 2(2 \cdot 6 + 0) + 1 \\ &= 2(2(2 \cdot 3 + 0) + 0) + 1 \end{aligned}$$

# Hvorfor virker det?

| Heltalsdivision | Kvotient | Rest |
|---|---|---|
| $25 : 2$ | 12 | 1 |
| $12 : 2$ | 6 | 0 |
| $6 : 2$ | 3 | 0 |
| $3 : 2$ | 1 | 1 |
| $1 : 2$ | 0 | 1 |

$$25 = 11001_2$$

$$\begin{aligned} 25 &= 2 \cdot 12 + 1 \\ &= 2(2 \cdot 6 + 0) + 1 \\ &= 2(2(2 \cdot 3 + 0) + 0) + 1 \\ &= 2(2(2(2 \cdot 1 + 1) + 0) + 0) + 1 \end{aligned}$$

# Hvorfor virker det?

| Heltalsdivision | Kvotient | Rest |
| --- | --- | --- |
| $25 : 2$ | 12 | 1 |
| $12 : 2$ | 6 | 0 |
| $6 : 2$ | 3 | 0 |
| $3 : 2$ | 1 | 1 |
| $1 : 2$ | 0 | 1 |

$$25 = 11001_2$$

$$\begin{aligned}
25 &= 2 \cdot 12 + 1 \\
&= 2(2 \cdot 6 + 0) + 1 \\
&= 2(2(2 \cdot 3 + 0) + 0) + 1 \\
&= 2(2(2(2 \cdot 1 + 1) + 0) + 0) + 1 \\
&= 2(2(2(2(2 \cdot 0 + 1) + 1) + 0) + 0) + 1
\end{aligned}$$

# Hvorfor virker det?

| Heltalsdivision | Kvotient | Rest |
|---|---|---|
| $25:2$ | 12 | 1 |
| $12:2$ | 6 | 0 |
| $6:2$ | 3 | 0 |
| $3:2$ | 1 | 1 |
| $1:2$ | 0 | 1 |

$$25 = 11001_2$$

$$\begin{aligned}
25 &= 2\cdot 12 + 1 \\
&= 2(2\cdot 6 + 0) + 1 \\
&= 2(2(2\cdot 3 + 0) + 0) + 1 \\
&= 2(2(2(2\cdot 1 + 1) + 0) + 0) + 1 \\
&= 2(2(2(2(2\cdot 0 + 1) + 1) + 0) + 0) + 1 \\
&= 2^5\cdot 0 + 2^4\cdot 1 + 2^3\cdot 1 + 2^2\cdot 0 + 2^1\cdot 0 + 2^0\cdot 1
\end{aligned}$$

# Hvorfor virker det?

| Heltalsdivision | Kvotient | Rest |
|---|---|---|
| $25:2$ | 12 | 1 |
| $12:2$ | 6 | 0 |
| $6:2$ | 3 | 0 |
| $3:2$ | 1 | 1 |
| $1:2$ | 0 | 1 |

$$25 = 11001_2$$

$$\begin{aligned}
25 &= 2 \cdot 12 + 1 \\
&= 2(2 \cdot 6 + 0) + 1 \\
&= 2(2(2 \cdot 3 + 0) + 0) + 1 \\
&= 2(2(2(2 \cdot 1 + 1) + 0) + 0) + 1 \\
&= 2(2(2(2(2 \cdot 0 + 1) + 1) + 0) + 0) + 1 \\
&= 2^5 \cdot 0 + 2^4 \cdot 1 + 2^3 \cdot 1 + 2^2 \cdot 0 + 2^1 \cdot 0 + 2^0 \cdot 1
\end{aligned}$$

Bemærk at sidste division altid er $1:2$ (med kvotient $0$ og rest $1$). Fordi $X$ bliver $1$ på et tidspunkt, da man ved en heltalsdivision med 2 hele tiden gør $X$ mindre, men ikke kan komme fra heltal $\geq 2$ til heltal $\leq 0$.

# Repræsentation af heltal

Talrepræsentationer bruger (næsten altid) et fast antal bits (så operationer kan implementeres effektivt).

$$\boxed{k \text{ bits} = 2^k \text{ forskellige bitmønstre}}$$

# Repræsentation af heltal

Talrepræsentationer bruger (næsten altid) et fast antal bits (så operationer kan implementeres effektivt).

$$\boxed{k \text{ bits} = 2^k \text{ forskellige bitmønstre}}$$

Positive heltal: det binære talsystem giver en naturlig repræsentation.

$k = 4$ :

| | | | |
|---|---|---|---|
| 0111 | 7 | 1111 | 15 |
| 0110 | 6 | 1110 | 14 |
| 0101 | 5 | 1101 | 13 |
| 0100 | 4 | 1100 | 12 |
| 0011 | 3 | 1011 | 11 |
| 0010 | 2 | 1010 | 10 |
| 0001 | 1 | 1001 | 9 |
| 0000 | 0 | 1000 | 8 |

# Repræsentation af heltal

Talrepræsentationer bruger (næsten altid) et fast antal bits (så operationer kan implementeres effektivt).

$$\boxed{k \text{ bits} = 2^k \text{ forskellige bitmønstre}}$$

Positive heltal: det binære talsystem giver en naturlig repræsentation.

$k = 4$ :

| | | | |
|---|---|---|---|
| 0111 | 7 | 1111 | 15 |
| 0110 | 6 | 1110 | 14 |
| 0101 | 5 | 1101 | 13 |
| 0100 | 4 | 1100 | 12 |
| 0011 | 3 | 1011 | 11 |
| 0010 | 2 | 1010 | 10 |
| 0001 | 1 | 1001 | 9 |
| 0000 | 0 | 1000 | 8 |

Hvordan skal disse $2^k$ bitmønstre fordeles, hvis vi både vil repræsentere negative og positive heltal?

# Two's complement

En mulig repræsentation af både negative og positive heltal er følgende:

$k = 4:$

| | | | |
|---|---|---|---|
| 0111 | 7 | 1111 | -1 |
| 0110 | 6 | 1110 | -2 |
| 0101 | 5 | 1101 | -3 |
| 0100 | 4 | 1100 | -4 |
| 0011 | 3 | 1011 | -5 |
| 0010 | 2 | 1010 | -6 |
| 0001 | 1 | 1001 | -7 |
| 0000 | 0 | 1000 | -8 |

Dette kaldes "two's complement" (af grunde, som ikke er relevante her).

# Two's complement

En mulig repræsentation af både negative og positive heltal er følgende:

$k = 4$ :

| | | | |
|---|---|---|---|
| 0111 | 7 | 1111 | -1 |
| 0110 | 6 | 1110 | -2 |
| 0101 | 5 | 1101 | -3 |
| 0100 | 4 | 1100 | -4 |
| 0011 | 3 | 1011 | -5 |
| 0010 | 2 | 1010 | -6 |
| 0001 | 1 | 1001 | -7 |
| 0000 | 0 | 1000 | -8 |

Dette kaldes "two's complement" (af grunde, som ikke er relevante her).

Det kan også beskrives som at højeste ciffer tæller $-(2^{k-1})$ i stedet for $2^{k-1}$:

$$\begin{aligned} 1101_2 &= 1 \cdot (-(2^3)) + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= 1 \cdot (-8) + 1 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 \\ &= -3 \end{aligned}$$

# Two's complement

Repræsentationen two's complement har mange gode egenskaber og vælges ofte.

# Two's complement

Repræsentationen two's complement har mange gode egenskaber og vælges ofte.

- Fortegn kan ses af første bit.

# Two's complement

Repræsentationen two's complement har mange gode egenskaber og vælges ofte.

- Fortegn kan ses af første bit.
- Simpel metode til at skifte fortegn findes:

  Kopier bits fra højre til venstre, til og med første 1-bit.
  Resten af bits inverteres.

  (Eksempel: $6 = 0110 \rightarrow 1010 =$ -6)

# Two's complement

Repræsentationen two's complement har mange gode egenskaber og vælges ofte.

- Fortegn kan ses af første bit.
- Simpel metode til at skifte fortegn findes:

  Kopier bits fra højre til venstre, til og med første 1-bit.
  Resten af bits inverteres.

  (Eksempel: $6 = 0110 \rightarrow 1010 =$ -6)

- Den almindelige metode til addition virker også for negative tal. Ingen ekstra logiske kredsløb for disse (sparer transistorer på CPU).

# Two's complement

Repræsentationen two's complement har mange gode egenskaber og vælges ofte.

- Fortegn kan ses af første bit.
- Simpel metode til at skifte fortegn findes:

  Kopier bits fra højre til venstre, til og med første 1-bit.
  Resten af bits inverteres.

  (Eksempel: $6 = 0110 \rightarrow 1010 =$ -6)

- Den almindelige metode til addition virker også for negative tal. Ingen ekstra logiske kredsløb for disse (sparer transistorer på CPU).
- Subtraktion kan laves ved at vende fortegn og addere. Ingen logiske kredsløb for subtraktion (sparer transistorer på CPU).

# Two's complement

Repræsentationen two's complement har mange gode egenskaber og vælges ofte.

- Fortegn kan ses af første bit.
- Simpel metode til at skifte fortegn findes:

  Kopier bits fra højre til venstre, til og med første 1-bit.
  Resten af bits inverteres.

  (Eksempel: $6 = 0110 \rightarrow 1010 =$ -6)

- Den almindelige metode til addition virker også for negative tal. Ingen ekstra logiske kredsløb for disse (sparer transistorer på CPU).
- Subtraktion kan laves ved at vende fortegn og addere. Ingen logiske kredsløb for subtraktion (sparer transistorer på CPU).

I Java er f.eks. typen `int` heltal i two's complement ($k = 32$). I Python er dette også grundtypen for heltal.

# Repræsentationer af kommatal

Talrepræsentationer bruger (næsten altid) et fast antal bits.

$$\boxed{k \text{ bits} = 2^k \text{ forskellige bitmønstre}}$$

Hvordan bruge $k$ bits til at beskrive kommatal?

# Repræsentationer af kommatal

Talrepræsentationer bruger (næsten altid) et fast antal bits.

$$\boxed{k \text{ bits} = 2^k \text{ forskellige bitmønstre}}$$

Hvordan bruge $k$ bits til at beskrive kommatal?

Fra tital-systemet kendes

- Fast decimalpunkt ($45.32$)
- Flydende decimalpunkt ($-6.87 \cdot 10^{-6}$)

Disse kan nemt gentages i total-systemet (grundtal 2). Se næste sider.

# Repræsentationer af kommatal

Talrepræsentationer bruger (næsten altid) et fast antal bits.

$$\boxed{k \text{ bits} = 2^k \text{ forskellige bitmønstre}}$$

Hvordan bruge $k$ bits til at beskrive kommatal?

Fra tital-systemet kendes

- Fast decimalpunkt ($45.32$)
- Flydende decimalpunkt ($-6.87 \cdot 10^{-6}$)

Disse kan nemt gentages i total-systemet (grundtal 2). Se næste sider.

I computere bruges oftest flydende decimalpunkt (med grundtal 2). For at forstå disse skal man forstå fast decimalpunkt (med grundtal 2) først.

I Java er typerne `float` ($k = 32$) og `double` ($k = 64$) kommatal i flydende decimalpunkt. I Python er typen `float` det samme ($k = 64$).

# Fast decimalpunkt

Tital-systemet:

$$\begin{aligned} 45.32 &= 4 \cdot 10 &+& \quad 5 \cdot 1 &+& \quad 3 \cdot 1/10 &+& \quad 2 \cdot 1/100 \\ &= 4 \cdot 10^1 &+& \quad 5 \cdot 10^0 &+& \quad 3 \cdot 10^{-1} &+& \quad 2 \cdot 10^{-2} \end{aligned}$$

# Fast decimalpunkt

Tital-systemet:

$$\begin{aligned} 45.32 &= 4 \cdot 10 &+\quad& 5 \cdot 1 &+\quad& 3 \cdot 1/10 &+\quad& 2 \cdot 1/100 \\ &= 4 \cdot 10^1 &+\quad& 5 \cdot 10^0 &+\quad& 3 \cdot 10^{-1} &+\quad& 2 \cdot 10^{-2} \end{aligned}$$

Det binære talsystem:

$$\begin{aligned} 10110.111_2 &= \phantom{+}\ 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 \\ &\quad + 0 \cdot 2^0 + 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} \\ &= \phantom{+}\ 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 \\ &\quad + 0 \cdot 1 + 1 \cdot 1/2 + 1 \cdot 1/4 + 1 \cdot 1/8 \\ &= 22\tfrac{7}{8} \\ &= 22.875 \end{aligned}$$

# Flydende decimalpunkt

Tital-systemet: få kommaet til at stå efter første ciffer $\neq 0$.

$2340000.0 = 2.34 \cdot 10^6 \qquad 0.000456 = 4.56 \cdot 10^{-4} \qquad -0.0987 = -9.87 \cdot 10^{-2}$

# Flydende decimalpunkt

Tital-systemet: få kommaet til at stå efter første ciffer $\neq 0$.

$2340000.0 = 2.34 \cdot 10^6$ $\quad$ $0.000456 = 4.56 \cdot 10^{-4}$ $\quad$ $-0.0987 = -9.87 \cdot 10^{-2}$

| | | | | | |
|---|---|---|---|---|---|
| Fortegn: | plus | Fortegn: | plus | Fortegn: | minus |
| Eksponent: | 6 | Eksponent: | $-4$ | Eksponent: | $-2$ |
| Mantisse: | 2.34 | Mantisse: | 4.56 | Mantisse: | 9.87 |

# Flydende decimalpunkt

Tital-systemet: få kommaet til at stå efter første ciffer $\neq 0$.

$2340000.0 = 2.34 \cdot 10^6$ $\quad 0.000456 = 4.56 \cdot 10^{-4}$ $\quad -0.0987 = -9.87 \cdot 10^{-2}$

| | | | | | |
|---|---|---|---|---|---|
| Fortegn: | plus | Fortegn: | plus | Fortegn: | minus |
| Eksponent: | 6 | Eksponent: | $-4$ | Eksponent: | $-2$ |
| Mantisse: | 2.34 | Mantisse: | 4.56 | Mantisse: | 9.87 |

Total-systemet: få kommaet til at stå efter første ciffer $\neq 0$ (er altid 1).

$101100.0_2 = 1.011_2 \cdot 2^5$ $\quad -0.01101_2 = -1.101_2 \cdot 2^{-2}$

# Flydende decimalpunkt

Tital-systemet: få kommaet til at stå efter første ciffer $\neq 0$.

$2340000.0 = 2.34 \cdot 10^6$ $\quad$ $0.000456 = 4.56 \cdot 10^{-4}$ $\quad$ $-0.0987 = -9.87 \cdot 10^{-2}$

| | | | | | |
|---|---|---|---|---|---|
| Fortegn: | plus | Fortegn: | plus | Fortegn: | minus |
| Eksponent: | 6 | Eksponent: | $-4$ | Eksponent: | $-2$ |
| Mantisse: | 2.34 | Mantisse: | 4.56 | Mantisse: | 9.87 |

Total-systemet: få kommaet til at stå efter første ciffer $\neq 0$ (er altid 1).

$101100.0_2 = 1.011_2 \cdot 2^5$ $\quad$ $-0.01101_2 = -1.101_2 \cdot 2^{-2}$

Der afsættes et fast antal bits til hver af: fortegn, eksponent, mantisse. For $k = 8$ kan vi f.eks. vælge: 1, 3 og 4 bits. Eksponent kan være positiv eller negativ, vi bruger two's complement til den. Mantisse fyldes om nødvendigt op med 0'er til højre. For $-0.01101_2$ fås:

| | | |
|---|---|---|
| Fortegn: | 1 | (1 for negativt tal, 0 for positivt) |
| Eksponent: | 110 | (-2 i two's complement (3 bits)) |
| Mantisse bits: | (1.)1010 | (første bit skrives ikke, da den altid er 1) |

# Flydende decimalpunkt

Tital-systemet: få kommaet til at stå efter første ciffer $\neq 0$.

$2340000.0 = 2.34 \cdot 10^6$ $\quad$ $0.000456 = 4.56 \cdot 10^{-4}$ $\quad$ $-0.0987 = -9.87 \cdot 10^{-2}$

| | | | | | |
|---|---|---|---|---|---|
| Fortegn: | plus | Fortegn: | plus | Fortegn: | minus |
| Eksponent: | 6 | Eksponent: | $-4$ | Eksponent: | $-2$ |
| Mantisse: | 2.34 | Mantisse: | 4.56 | Mantisse: | 9.87 |

Total-systemet: få kommaet til at stå efter første ciffer $\neq 0$ (er altid 1).

$101100.0_2 = 1.011_2 \cdot 2^5$ $\quad$ $-0.01101_2 = -1.101_2 \cdot 2^{-2}$

Der afsættes et fast antal bits til hver af: fortegn, eksponent, mantisse. For $k = 8$ kan vi f.eks. vælge: 1, 3 og 4 bits. Eksponent kan være positiv eller negativ, vi bruger two's complement til den. Mantisse fyldes om nødvendigt op med 0'er til højre. For $-0.01101_2$ fås:

| | | |
|---|---|---|
| Fortegn: | 1 | (1 for negativt tal, 0 for positivt) |
| Eksponent: | 110 | (-2 i two's complement (3 bits)) |
| Mantisse bits: | (1.)1010 | (første bit skrives ikke, da den altid er 1) |

Så $-0.01101_2$ repræsenteres som $\boxed{1}\boxed{110}\boxed{1010}$.

# Begrænsninger

Heltal og kommatal er uendelige talmængder. Hvis der afsættes et fast antal ($k$) bits fås et endeligt antal ($2^k$) forskellige bitmønstre.

# Begrænsninger

Heltal og kommatal er uendelige talmængder. Hvis der afsættes et fast antal ($k$) bits fås et endeligt antal ($2^k$) forskellige bitmønstre.

$\boxed{\text{Ikke alle tal kan repræsenteres!}}$

# Begrænsninger

Heltal og kommatal er uendelige talmængder. Hvis der afsættes et fast antal ($k$) bits fås et endeligt antal ($2^k$) forskellige bitmønstre.

Ikke alle tal kan repræsenteres!

Viser sig f.eks. ved

- Overflow
  - maxInt + maxInt = ?
- Rounding errors
  - Stort tal $x$ + meget lille tal $y$ = samme store tal $x$
  - $(x + y) + z \neq x + (y + z)$ hvis f.eks. $x + y$ ikke kan repræsenteres eksakt.

# Begrænsninger

Heltal og kommatal er uendelige talmængder. Hvis der afsættes et fast antal ($k$) bits fås et endeligt antal ($2^k$) forskellige bitmønstre.

$$\boxed{\text{Ikke alle tal kan repræsenteres!}}$$

Viser sig f.eks. ved

- Overflow
  - maxInt + maxInt = ?
- Rounding errors
  - Stort tal $x$ + meget lille tal $y$ = samme store tal $x$
  - $(x + y) + z \neq x + (y + z)$ hvis f.eks. $x + y$ ikke kan repræsenteres eksakt.

I praksis opleves sjældent problemer pga. et stort antal bits i talrepræsentationerne.

Alternativt findes programmeringsbiblioteker, der implementerer f.eks. vilkårligt store heltal (under brug af variabelt antal bits, samt tab af effektivitet). Dette sker automatisk i Python for typen `int`.