

Repræsentation af tal

DM534

Rolf Fagerberg

Mål

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

Dette emne er et uddrag af kurset *DM548 Computerarkitektur og systemprogrammering* (3. semester).

Bitmønstre

011010110001100101011011...

Bitmønstre skal *fortolkes* for at have en betydning:

- ▶ Tal (heltal, kommatal)
- ▶ Bogstaver
- ▶ Computerinstruksjon (program)
- ▶ Pixels (billedfil)
- ▶ Amplitude (lydfil)
- ▶ ⋮

Bitmønstre

011010110001100101011011...

Bitmønstre skal *fortolkes* for at have en betydning:

- ▶ Tal (heltal, kommatal)
- ▶ Bogstaver
- ▶ Computerinstruksjon (program)
- ▶ Pixels (billedfil)
- ▶ Amplitude (lydfil)
- ▶ ⋮

Fokus her: heltal og kommatal.

Talsystemer

Tital-systemet:

4532

Talsystemer

Tital-systemet:

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

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 $10 \cdot 10^i = 10^{i+1}$)

Syvttal-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 $10 \cdot 10^i = 10^{i+1}$)

Syvttal-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*.

Addition

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

Tital-systemet:

$$\begin{array}{r} 1111 \\ 5432 \\ +96781 \\ \hline = 102213 \end{array}$$

Addition

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

Tital-systemet:

$$\begin{array}{r} 1111 \\ 5432 \\ +96781 \\ \hline = 102213 \end{array}$$

Total-systemet:

$$\begin{array}{r} 111 \\ 1110_2 \\ +11100_2 \\ \hline = 101010_2 \end{array}$$

Addition

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

Tital-systemet:

$$\begin{array}{r} 1111 \\ 5432 \\ +96781 \\ \hline = 102213 \end{array}$$

Total-systemet:

$$\begin{array}{r} 111 \\ 1110_2 \\ +11100_2 \\ \hline = 101010_2 \end{array}$$

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

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

Konvertering til binært talsystem

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

$$X = N$$

Så længe $X > 0$:

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 :

$$X = N$$

Så længe $X > 0$:

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	25 = 11001 ₂
3/2	1	1	
1/2	0	1	

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

Hvorfor virker det?

Heltalsdivision	Kvotient	Rest	
25/2	12	1	
12/2	6	0	
6/2	3	0	25 = 11001 ₂
3/2	1	1	
1/2	0	1	

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

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): X bliver 1 på et tidspunkt, da man ved en heltalsdivision med 2 hele tiden gør X mindre, men ikke kan komme fra heltal ≥ 2 til heltal ≤ 0 .

Repræsentationer af heltal

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

k bits = 2^k forskellige bitmønstre

Repræsentationer af heltal

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

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

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

Negative heltal

Forskellige forslag for k bit heltal:

- A Sign bit: Første bit = fortegn, resten af bits er binært talsystem.
- B Excess: tæl op fra -2^{k-1} med binært talsystem.
- C Two's complement: Første bit angiver $-(2^{k-1})$ i stedet for 2^{k-1} , ellers som binært talsystem.

Negative heltal

Forskellige forslag for k bit heltal:

- A Sign bit: Første bit = fortegn, resten af bits er binært talsystem.
- B Excess: tæl op fra -2^{k-1} med binært talsystem.
- C Two's complement: Første bit angiver $-(2^{k-1})$ i stedet for 2^{k-1} , ellers som binært talsystem.

Eksempel med $k = 4$:

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

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.
- ▶ 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.
- ▶ 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).

Metode til at skifte fortegn:

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

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

Decimaltal

Fast decimalpunkt:

Tital-systemet:

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

Decimaltal

Fast decimalpunkt:

Tital-systemet:

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

Det binære talsystem:

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

Flydende decimalpunkt

Flydende decimalpunkt (alias videnskabelig notation):

Tital-systemet:

$$-0.00000456 = (-1) \cdot 4.56 \cdot 10^{-6}$$

Fortegn: -1

Eksponent: -6

Mantisse: 4.56

Flydende decimalpunkt

Flydende decimalpunkt (alias videnskabelig notation):

Total-systemet:

$$-0.00000456 = (-1) \cdot 4.56 \cdot 10^{-6}$$

Fortegn: -1

Eksponent: -6

Mantisse: 4.56

Total-systemet:

$$-0.01101_2 = (-1) \cdot 1.101_2 \cdot 2^{-2}$$

Flydende decimalpunkt

Flydende decimalpunkt (alias videnskabelig notation):

Tital-systemet:

$$-0.00000456 = (-1) \cdot 4.56 \cdot 10^{-6}$$

Fortegn: -1

Eksponent: -6

Mantisse: 4.56

Total-systemet:

$$-0.01101_2 = (-1) \cdot 1.101_2 \cdot 2^{-2}$$

Der afsættes et fast antal bits til hver af de tre dele (fortegn, eksponent, mantisse). Vi bruger her: 1, 3 og 4 bits. Eksponent angives i excess notation. Mantisse fyldes om nødvendigt op med 0'er til højre.

Sign bit:	1	(sign bit 1 for negativt tal)
Eksponent:	010	(-2 i excess notation (3 bits))
Mantisse bits:	(1)101	(første bit underforstået)

Så resultatet er 1|010|1010.

Begrænsninger

Heltal (\mathbb{N} , \mathbb{Z}) og reelle tal (\mathbb{R}) er uendelige talmængder.

Hvis der afsættes et fast antal (k) bits fås et endeligt antal (2^k) forskellige bitmønstre.

Begrænsninger

Heltal (\mathbb{N} , \mathbb{Z}) og reelle tal (\mathbb{R}) er uendelige talmængder.

Hvis der afsættes et fast antal (k) bits fås et endeligt antal (2^k) forskellige bitmønstre.

Ikke alle tal kan repræsenteres!

Begrænsninger

Heltal (\mathbb{N} , \mathbb{Z}) og reelle tal (\mathbb{R}) 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
 - ▶ $\text{maxInt} + \text{maxInt} = ?$
- ▶ 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 (\mathbb{N} , \mathbb{Z}) og reelle tal (\mathbb{R}) 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
 - ▶ $\text{maxInt} + \text{maxInt} = ?$
- ▶ 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).

Hexadecimal notation

Gruppér bits i grupper af 4 (dvs. 16 forskellige muligheder):

0110 1010 1110 01 . . .

Hexadecimal notation

Gruppér bits i grupper af 4 (dvs. 16 forskellige muligheder):

01101010111001...

Vælg 16 tegn:

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

Hexadecimal notation

Gruppér bits i grupper af 4 (dvs. 16 forskellige muligheder):

`0110``1010``1110``01...`

Vælg 16 tegn:

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

Giver en simpel/kort måde at beskrive bitstrengene:

`0110``1010``1110``01...` = 6AE...

Hexadecimal notation

Gruppér bits i grupper af 4 (dvs. 16 forskellige muligheder):

`0110``1010``1110``01`...

Vælg 16 tegn:

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

Giver en simpel/kort måde at beskrive bitstreng:

`0110``1010``1110``01`... = 6AE...

NB: kan også bruges som cifre i et talsystem med grundtal 16.

Hexadecimalt talsystem

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 $10 \cdot 10^i = 10^{i+1}$)

Hexadecimalt talsystem

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 $10 \cdot 10^i = 10^{i+1}$)

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), ..., E(=15)
(fordi $16 \cdot 16^i = 16^{i+1}$)