Ettor och nollor

11111... **00000...**

Ni vet nog att datorn egentligen bara kan förstå ettor och nollor

Men ni vet också att datorn kan hantera och lagra tal, bokstäver och andra tecken.

2014 | 15,78 | $6{,}6743 \cdot 10^{-11}$ | -86 | !##?VxQÖ

0703-123456

Bä bä vita lamm har du någon ull

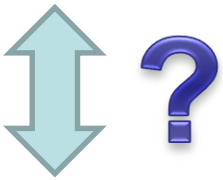
Dessutom kan datorn hantera bilder och ljud

Hur kan det fungera?

Detta dokument skall ge en förklaring till hur datorn hanterar tal och tecken.

2014 | 15,78 | $6{,}6743 \cdot 10^{-11}$ | -86 | !##?VxQÖ

0703-123456 | Bä bä vita lamm har du någon ull



Informationen i datorn

Exempel på information som lagras/hanteras i datorn

- tal
- text (tecken)
- bilder
- ljud
- program

Hur lagras informationen?

- I *datorn* används koder för att representera *all information*
- Koderna består av 1:or och 0:or
- Binära tal används (binära talsystemet)

- Historik: Info lagrades tidigare på hålkort och hålremsa (demo av hålkort…)

Vad är ett talsystem?

Ett talsystem har en bas och siffror (symboler)

Ex.decimala systemet har basen 10 och symbolerna 0-9:

$103_{10} = 1*10^2+0*10^1+3*10^0= 1*100 + 0*10 + 3*1$

Binära talsystemet

Basen 2 och symbolerna 0, 1.

$101_2 = 1*2^2+0*2^1+1*2^0= 4+0+1= 5_{10}$

Översätt från decimalt till binärt:

$27_{10} = 16 + 8 + 2 + 1 =$

$1*2^4+ 1*2^3+ 0*2^2 +1*2^1+ 1*2^0= 11011_2$

Binära talsystemet…

En *byte* = 8 bit, 0-255, dvs 256 st värden

```
00000000
00000001
00000010
…
11111111
```

Ex. en byte:

$00001111 = 0*2^7 + 0*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 15_{10}$

Binär aritmetik enkel…

Exempel addition:

1+0=1, 0+1=1, 1+1=10 **(2^1)**

```
 1001001
+0010010
 -------
 1011011
```

```
  11  1
 1001001
+0011001
 -------
 1100010
```

```
 1001001
+1000000
 -------
10001001
```

Exempel multiplikation:

1*1=1, 0*1=0, 1*0=0

```
 110   (6)
* 10   (2)
 ---
 000
110
----
1100   (12)
```

Aritmetik binärt är enkelt…

Exempel addition i decimala talsystemet: 18 + 15 = 33

```
  1
 18
+15
 --
 33
```

Talen 18 och 15 översätts till binära tal:

18_{10}=16+2 = $2^4 + 2^1$ =10010_2

15_{10}= 8+4+2+1 = $2^3 + 2^2 + 2^1 + 2^0 = 1111_2$

Additionen binärt blir:
10010 + 01111 = 100001

```
  111
 10010
+01111
------
100001
```

Binära talsystemet…

Bit : 0 eller 1

Byte: 8 bitar

Kilobyte (KB): 10^3 byte (tusen)

Megabyte (MB): 10^6 byte (miljon)

Gigabyte (GB): 10^9 byte (miljard)

Terabyte (TB): 10^{12} byte

En dators s.k. internminne kan vara 8 GB

Hexadecimala talsystemet

- Basen 16 och symbolerna 0-9,A,B,C,D,E,F

Ex:

$2F_{16} = 2*16^1 + F*16^0 = 2*16^1 + 15*16^0$

$= 32 + 15 = 47_{10}$

Omvandling hexadecimalt → binärt och tvärtom är relativt enkelt (dela upp I fyrgrupper):

$1111010 = 1111\ \ 0101 = 15*16^1 + 5*16^0 = F5_{16}$

- Hexadecimala tal är ett kompakt sätt att visa ett binärt tal
- Färgvärden kan uttryckas som hex. tal

Andra talsystem

Sexagesimala - basen 60

Användes av Babylonierna 2000 f.Kr.

Saknade nolla.

Härifrån kommer: 60 min = 1 timme, 360 grader = 1 varv.

Vigesimala - basen 20

Användes av Mayafolket ca 500 e.Kr.

Duodecimala - basen 12

Förenklade "vardagsmatematik" som 1/4, 1/3 och 1/2.
Från basen 12 kommer begreppen

dussin (12) och gross (144).

Basen användes i England för bl.a. mynt

Att lagra/hantera tal i datorn

Tal kan lagras som

- Heltal
- Flyttal

Heltal

- Lagras ofta i 4 byte, dvs 32 bit.
- En bit används för +/-
  Vänstra biten=0 → positivt tal 31 bitar
  Vänstra biten=1 → negativt tal 31 bitar

- Värdemängden i 4 byte: $[-2^{31}, 2^{31}-1]$, d.v.s.

  [-2147483648, 2147483647]

Exempel:

$67_{10} = 1000011_2$ och lagras i 4 byte som

00000000 00000000 00000000 01000011

Negativa heltal (överkurs)

- Lagras enligt *2-komplementform*
- Exempel: Antag att värdet -1 skall lagras.
  Vi börjar med värdet +1 binärt som är:

```
00000000 00000000 00000000 00000001
```

  Gör alla 1:or till 0:or och tvärtom, dvs gör komplementet vilket ger:

```
11111111 11111111 11111111 11111110
```

  Addera ett, ger slutresultatet, dvs -1 lagras som:

```
11111111 11111111 11111111 11111111
```

  Exempel: -2 lagras som:

```
11111111 11111111 11111111 11111110
```

  Exempel: -3 lagras som:

```
11111111 11111111 11111111 11111101
```

- Det största talet negativa talet som kan lagras med 2- komplementform är

```
10000000 00000000 00000000 00000000
```

  Det motsvarar $-2^{31} = -2147483648$

Problem... (överkurs)

Det största positiva heltalet i fyra byte är 2147483647, motsvarande binära tal som ser ut så här:

```
01111111  11111111  11111111  11111111
```

Om vi nu adderar ett till det värdet kommer det bli ett negativt tal

```
-2147483648
```

Förklaring:

```
 01111111  11111111  11111111  11111111
+00000000  00000000  00000000  00000001
 ----------------------------------------
 10000000  00000000  00000000  00000000
```

Tecken

Lagras ofta i en byte (i java 2 byte)
Tecken kodas till heltal ,exvis ASCII-koden, som sedan lagras binärt.

Vi studerar www.asciitabell.se

(koderna visas där som decimala, oktala, hexadecimala, binära tal)

Övning:

Öppna Notepad och knappa in några tecken. Spara filen. Markera med högermusknapp på filens ikon och välj Properties så får du veta hur stor filen är uttryckt i antal bytes (*size* resp. *size on disk*)

Testa det även i Word!

Text

Text är en sekvens av tecken

Exempel, texten:

`Torsten`

Består av 7 tecken som kodas enligt ascii-koden (decimalt som):

T	o	r	s	t	e	n
84_{10}	111_{10}	114_{10}	115_{10}	116_{10}	101_{10}	110_{10}

Vilket kan lagras binärt i 7x1 byte enligt ascii-koden.

Flyttal

- Används för stora och små tal
- Decimaltal
- 10-potenstal
- Ett flyttal = $0.\text{mantissa} * 2^{\text{exponent}}$
- Lagras som två delar: Mantissa (siffrorna) och exponent
- Lagras vanligen i 4 eller 8 byte
- 4 byte: 3 byte mantissa och 1 byte exponent
- 8 byte: 6 byte mantissa och 2 byte exponent

Princip för lagring av flyttal…

Ex: $2,5_{10} = 2*(10)^0 + 5*(10)^{-1} = 0.25*(10)^1$

Men tal lagras binärt, gör om till binärt:

$2,5_{10} = 1*(2)^1 + 0*(2)^0 + 1*(2)^{-1}$

$= 10.1_2$ (obs binärpkt).

Flytta binärpkt: $10.1_2 = 0.101*2^2$

Mantissan är 101

Exponenten är **2** = 10 binärt.

Om talet lagras i 4 byte:

Mantissa (3 byte) = 101

Exponent (1 byte) = 10

Flyttal…

Problem med flyttal

- Begränsad mängd siffror:

  4 byte: 7 siffrors noggrannhet

  8 byte: 15 siffrors noggrannhet

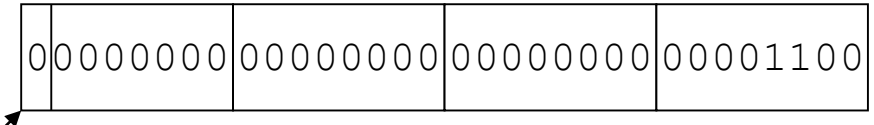
- Kan vanligen ej lagras exakt:

  Ex.flyttalet 2,6 = 2,5+0,1

$$2,6_{10} = 1*(2)^{1} + 0*(2)^{0} + 1*(2)^{-1} + 0*(2)^{-2} + 0*(2)^{-3} + 1*(2)^{-4} + 1*(2)^{-5} + \ldots$$

Princip för hur tal lagras i datorn

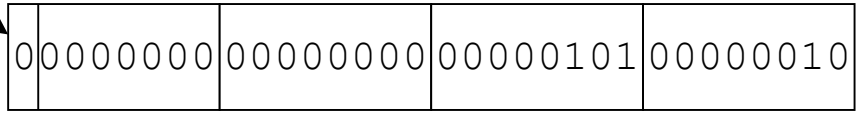
Heltalet 12 i fyra byte



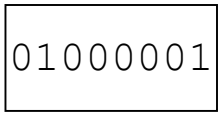
Flyttalet 2,5 i fyra byte

Teckenbit

Mantissan (3 byte) Exponent (1 byte)



Tecknet A i en byte (ASCII-kod)



Övningar

1. Vad är det decimala talet 37 uttryckt binärt?
2. Vad är det binära talet 10111 uttryckt decimalt?
3. Kan 120 vara ett binärt tal?
4. Vad är hexadecimala talet A4 uttryckt binärt resp. decimalt?
5. Vad är tecknet G uttryckt binärt (enligt ascii-koden)?
6. Hur lagras texten `Data` binärt?
7. Vad är heltalet 3 resp tecknet 3 uttryckt binärt?
8. Vad är flyttalet 4,5 uttryckt binärt?
9. Vad är flyttalet 4,6 uttryckt binärt?

Info = Bilder, ljud

Dessa kodas och lagras som filer.
Ex. JPEG-filer, MP3-filer…

Filernas innehåll är binärt.

Info = Program

Alla program (t.ex. exe-filer, class-filer) som körs i datorn är också uppbyggda av 0:or och 1:or, precis som vilken annan information som helst i datorn.

Är det ett program tolkas koderna som instruktioner så att datorn vet vad som ska utföras.

Informationen lagras i minnen

- Primärminne (arbetsminne)
  - Program som just nu körs
  - Data dvs värden (variabler) som programmen som just nu körs behöver
- Sekundärminne:
  För långtidslagring av dokument, program,bilder
  - Hårddisken
  - DVD
  - USB
  - …

Primärminnet

- Indelat i ett antal minnesceller där värden kan lagras
- Varje minnescell har en egen adress
- När värdet behövs använder datorn adressen för att komma åt det
- En minnescell består av en byte
- Innehållet i minnet är flyktigt

Primärminnet…

Värden lagras i flera minnesceller

efter varann.

Exempel, två värden: ett heltal och en text

Adress	Värde/byte	
4122	00010001	Ett heltal, 4 byte
4123	01000100	
4124	01111101	
4125	00001111	
4126	10010001	Ett char-värde (2 byte)
4127	00111100	

Sekundärminnet

- Data som ska sparas lägger man på en fil
- En fil kan t ex innehålla programkod eller data
- Filer som hör ihop brukar man lägga i en mapp

Vad är vad?

Om nu allting består av en sekvens av 0:or och 1:or, hur vet då datorn hur den ska tolka denna sekvens?

Jo…

Datorn vet vad det är för typ av information och hur den ska tolkas.

Informationen lagras i filer på sekundärminnet.

I början av varje fil finns det filinformation som anger filens typ, filens längd, etc. Denna information anger hur innehållet ska tolkas.

Ex.
Vi kör programmet Notepad (klickar på filen notepad.exe): innehållet I filen är programkod.

Om vi öppnar ett dokument i NotePad: innehållet i dokumentet tolkas som text (tecken).

Testa att öppna ett worddokument (som bara innehåller tecknen abcdef) i NotePad.