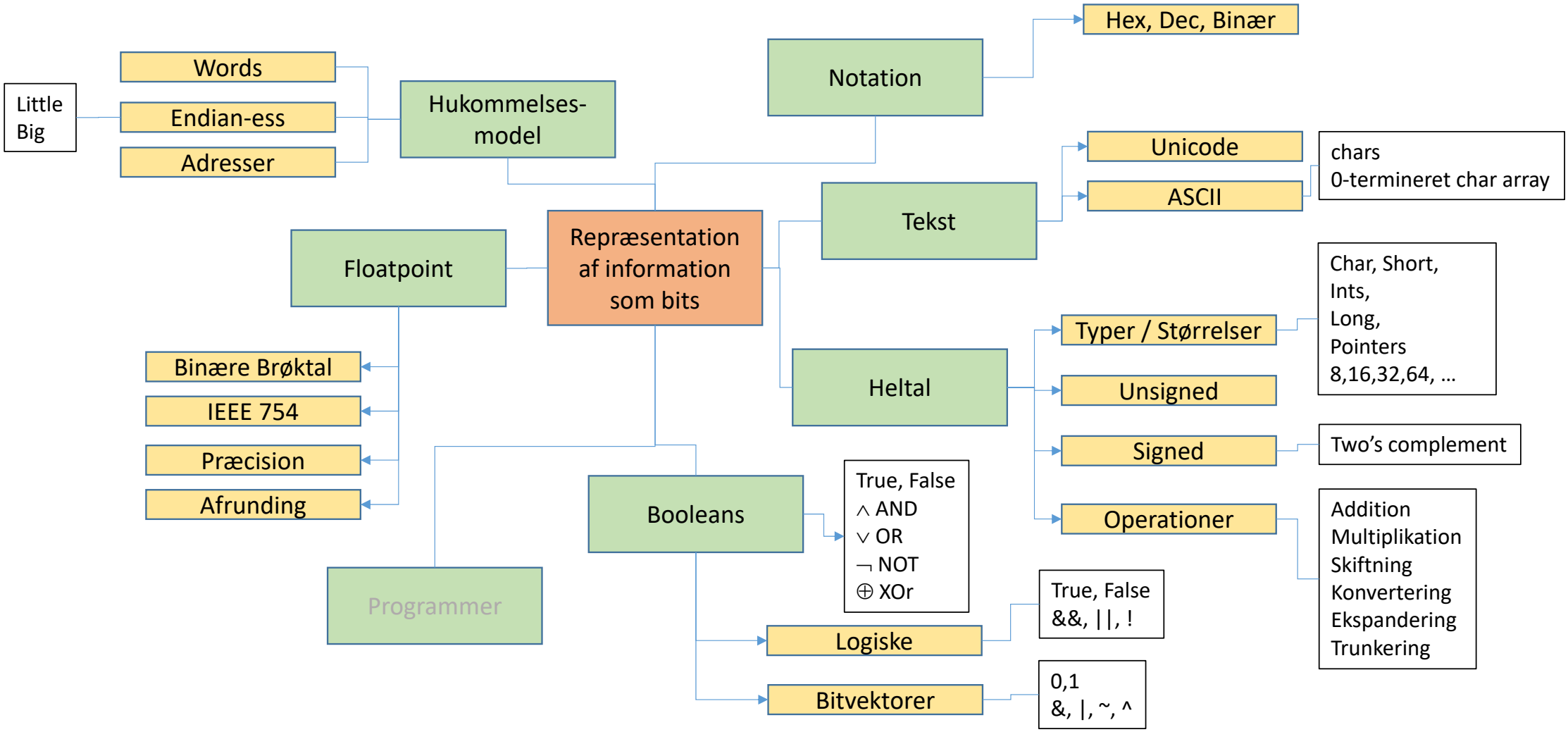
Computer Arkitektur og Operativ Systemer
Repræsentation af tal

Forelæsning 2
Brian Nielsen

Credits to
Randy Bryant & Dave O'Hallaron (CMU)

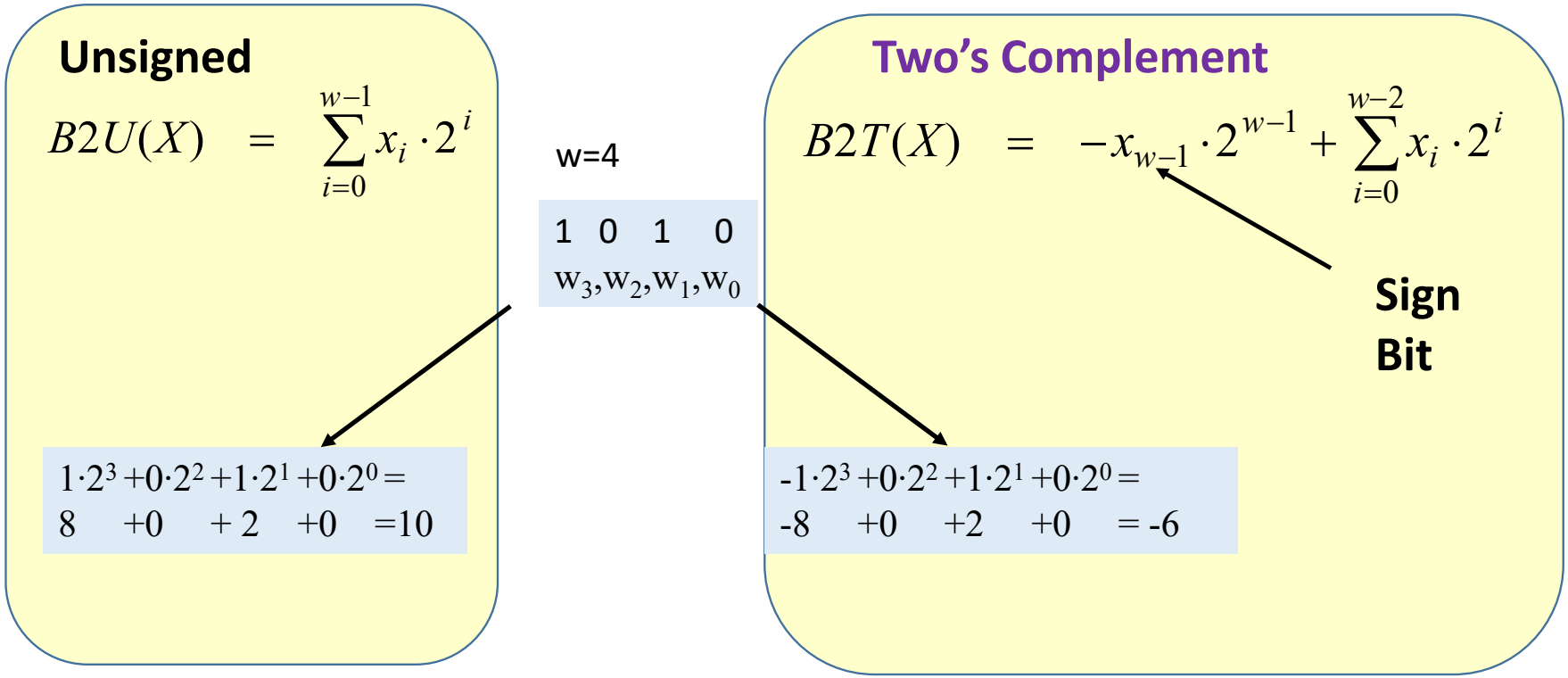
Plenum (ca 8.15-9.30)

Overblik over emner: forelæsning 1+2



Hvordan repræsenteres heltal (signed/unsigned)?

- Med *w*-bits kan vi indkode 2^w forskellige værdier



Hvilke heltal kan repræsenteres med w-bits?

Unsigned Værdier

- *UMin*=0
  000…0
- *UMax*= $2^{w-1}+\ldots+2^1+2^0 = 2^w - 1$
  111…1

Værdier i Two's Complement

- *TMin* = -2^{w-1}
  100…0
- *TMax* = $2^{w-1} - 1$
  011…1
- Minus 1
  111…1

Værdier for *W* = 16

	Decimal	Hex	Binary
UMax	65535	FF FF	11111111 11111111
TMax	32767	7F FF	01111111 11111111
TMin	-32768	80 00	10000000 00000000
-1	-1	FF FF	11111111 11111111
0	0	00 00	00000000 00000000

char: w=8
short int: w=16
int: w=32
long int: w=64

Quizz: Heltal Repræssentation

Betragt 8-bit ordet w=10101010.

1. Hvilken værdi har w fortolket som unsigned (decimal)?
2. Hvilken værdi har w fortolket som signed two's complement (decimal)?
3. Hvilken two's complement bit-repræsentation skal w have for at gemme decimal værdien -8?

Consider a (hypothetical) CAOSv1 machine that uses **6-bit words** to represent integer numbers. Give answers as decimal numbers.

1. What is the smallest **unsigned** integer that the machine can represent as a single word
2. What is the largest **unsigned** integer that the machine can represent as a single word?
3. What is the smallest **signed** integer that the machine can represent as a single word?
4. What is the largest **signed** integer that the machine can represent as a single word?

Hvad sker der ved addition?

- **Overløb,** men veldefineret
- Unsigned

$$UAdd_w(u , v)= \begin{cases} u+v, & ,u+v \leq UMax \\ u+w- 2^w & ,u+v > UMax \end{cases}$$

$$=(u + v) \mod 2^w$$

- Signed

$$TAdd_w(u , v) = \begin{cases} (u + v) - 2^w & , TMax < u+v \text{ (pos overløb)} \\ (u + v) & , TMin \leq u+v \leq TMax \text{ (normalt)} \\ (u + v) + 2^w & , u+v < TMin \text{ (neg overløb)} \end{cases}$$

- Bevarer normale regne-regler for addition af heltal (”Abelsk gruppe”)!

```
Fx, w=4, 8+11=19
  1000
+ 1011
------
 ~~1~~0011 //19 mod 2^4 = 3
```

SIGNED Negativ overløb

```
Fx, w=4,-7+-5=-12
  1001
+ 1011
------
 ~~1~~0100 =4
//NB: -12+2^4 = 4
```

Bogen viser at man kan ræsonnere formelt om, hvad der sker på laveste maskin niveau!
I eksamineres ikke i ***selve*** beviserne – men sætningerne og en vis forståelse af baggrunden derfor

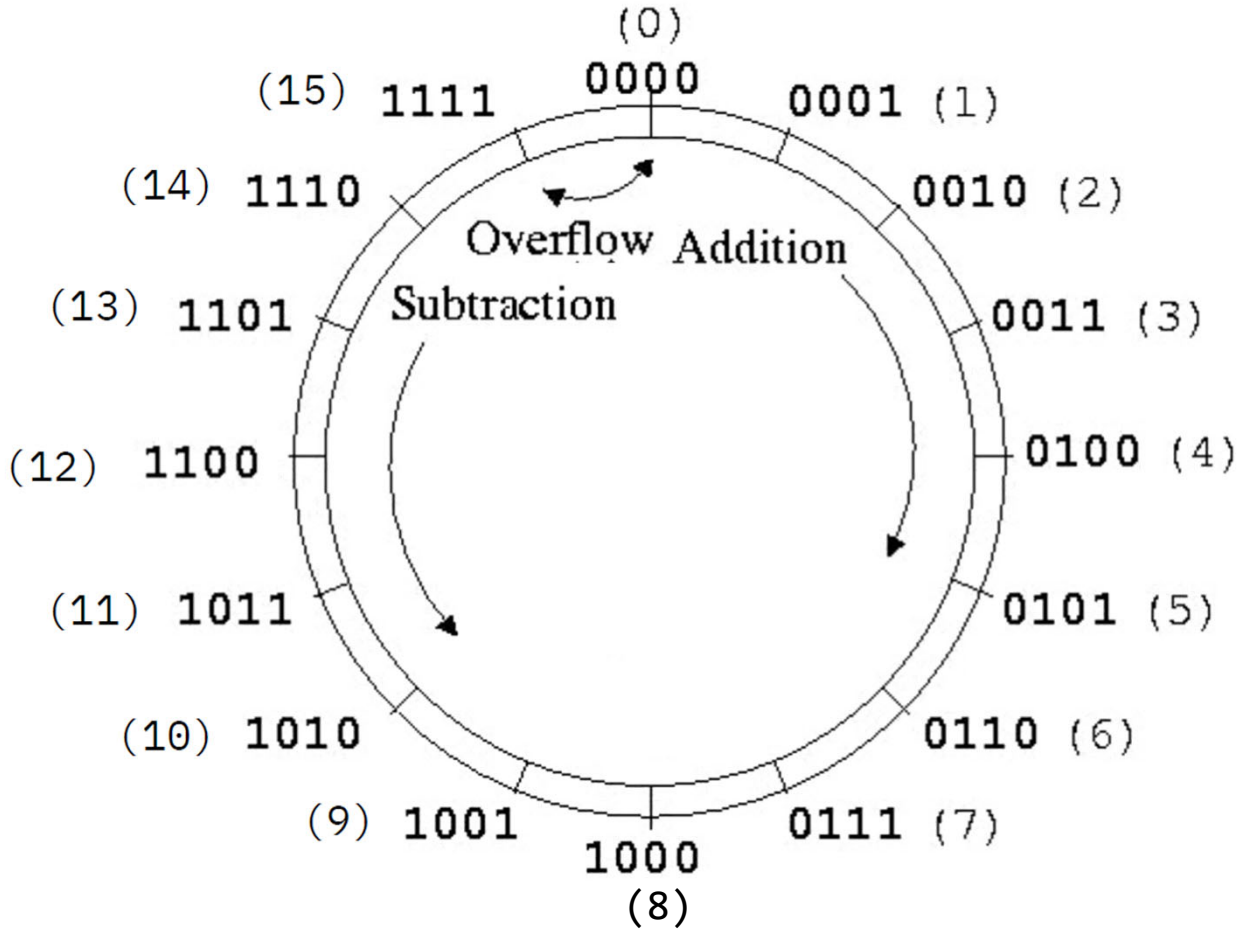
Anden måde at forstå circular arithmetik, unsigned

w=4

- Binær addition /subtraction
- Værdi i () fortolket som unsigned
- Ved overflow er værdi off med $16=2^4$ ifht. den rigtigt beregnede matematiske værdi
- Ex: 15+2=17 %16 giver 1

```
   1111  +
   0010  =
1  0001
```

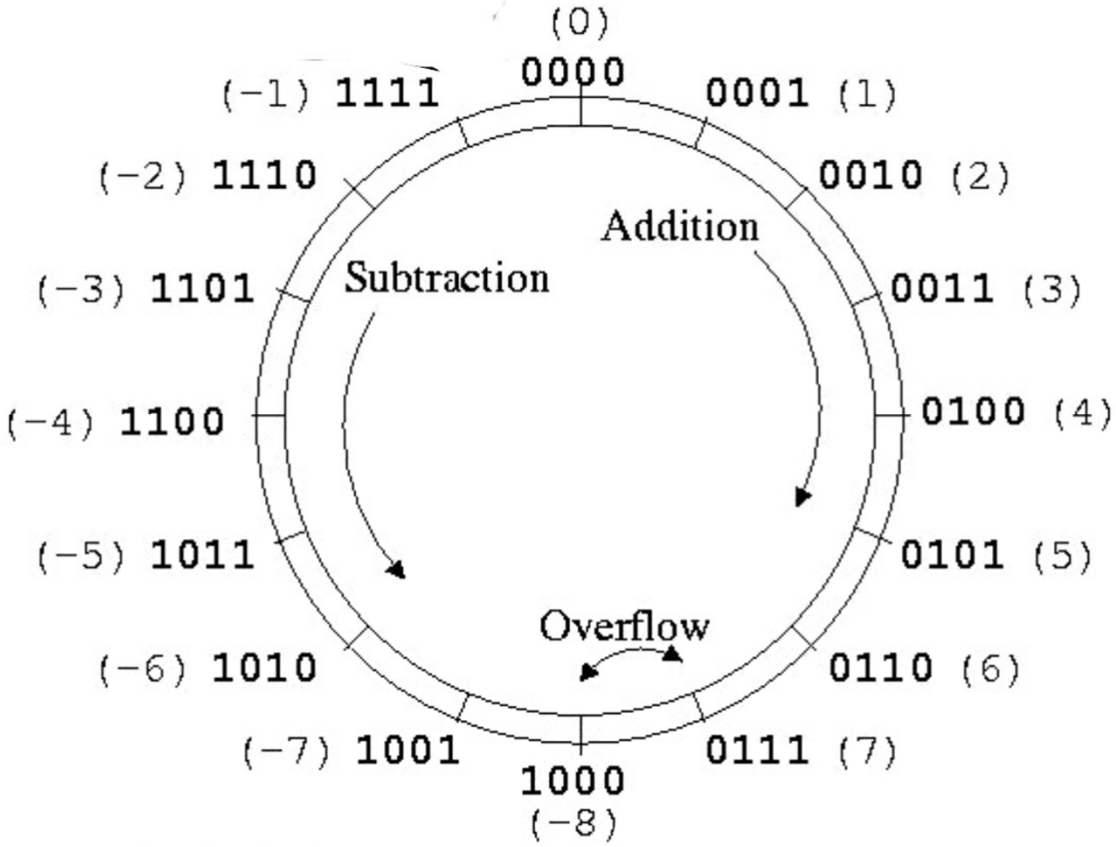
- Generelt modulo 2^w



Anden måde at forstå circular arithmetik, signed

w=4

- Binær addition /subtraction
- Værdi i () fortolket som signed two's complement
- Ved overflow er værdi off med +16 eller -16 (=2^4) ifht den rigtigt beregnede matematiske værdi
- Generelt 2^w
- Binær add omkring 0000 giver overflow, men korrekt værdi



Hvordan håndteres konvertering imellem signed & unsigned?

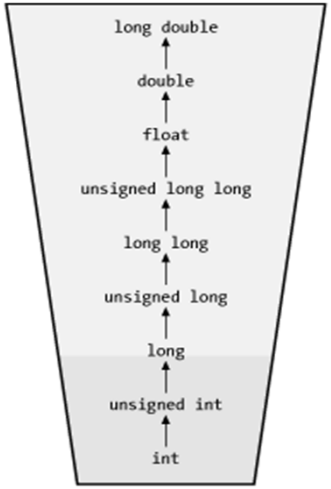
- Hvad sker der i C?
- Hvad sker der på maskin-niveau når man konverterer fra signed til unsigned?

```
int tx,ty;
unsigned ux,xy;

tx=ux;
uy=ty;
```

Mulig difference mellem på 2^w
Alvorlige overraskelser ved blandede udtryk

- Grundlæggende regler ved konvertering mellem Signed ↔ Unsigned
  - Bit mønstret bevares
  - Men genfortolkes som den nye type
  - Kan have overraskende effekter: addition eller subtraktion med 2^w



- Hvis udtryk indeholder både signed og unsigned integers
  - **`int`** konverteres til **`unsigned`**!!

Quizz: Heltal Overflow

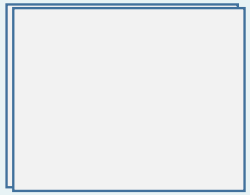
Betragt (den hypotetiske) CAOSv1 maskine som anvender **6 bits** til repræssentation af heltal (integers). Det er endvidere givet at den anvender **two's complement** til repræssentation af signed integers og two's complement addition på signed integers.

A C-program has the following declarations:

```
unsigned int x1 = UINT_MIN ;
unsigned int x2 = UINT_MAX; //UMax_w
int y1 = INT_MIN ; //TMin_w
int y2 = INT_MAX; //TMax_w
int y3 = 6;
```

Beregn værdien af nedenstående C-udtryk (giv resultater som decimal værdier (10 talssystem):

Integer udtryk:



1. x2+1
2. x2+y3
3. x1-y3
4. y1-1
5. y1-y3
6. y2+y3
7. x2+y1

Extension

- Unsigned

```
unsigned long x
unsigned int y;
x = y; //fra 32-> 64 bit
```

Udfyldes med 0'er: "zero-extension"

Ex. med w=4 -> w= 8

```
y=Decimal 7:   0111 -> x= 0000 0111
y=Decimal 10:  1010 -> x= 0000 1010
```

- Signed

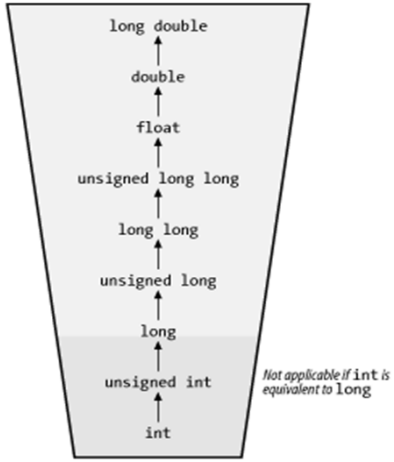
```
long x;
int y;
x= y; //fra 32 -> 64 bits
```

MSB (fortegnsbit gentages): "sign extension"

Ex med w=4 -> w=8

```
Decimal 7:  0111 -> 0000 0111
Decimal -6: 1010 -> 1111 1010
```

(-128+64+32+16+8+0+2+0)



Extension

```
int expand_u() {
 unsigned long x;
 unsigned int y;
 y=7;
 x=y;
 printf("x=%lu \n",x);
 printBits(sizeof(x), &x);

 y=10;
 x=y;
 printf("x=%lu \n",x);
 printBits(sizeof(x), &x);
}

int expand_s() {
 long x;
 int y;
 y=7;
 x=y;
 printf("x=%ld \n",x);
 printBits(sizeof(x), &x);

 y=-6;
 x=y;
 printf("x=%ld \n",x);
 printBits(sizeof(x), &x);
}
```

```
caos@caos-virtual-machine:~/Skrivebord$ gcc printBits.c
caos@caos-virtual-machine:~/Skrivebord$ ./a.out
x=7
00000000000000000000000000000000000000000000000000000000000000111
x=10
00000000000000000000000000000000000000000000000000000000000001010
x=7
00000000000000000000000000000000000000000000000000000000000000111
x=-6
11111111111111111111111111111111111111111111111111111111111111010
caos@caos-virtual-machine:~/Skrivebord$
```

Truncation: w->w' bits, w'<w

- Unsigned

```
unsigned short x;
unsigned char y;
y = x; //fra 16-> 8 bit
```

8 lsb (mindst betydende bits) bevares

Modulo $2^{w'}$-aritmetik

```
Ex. med w=8 -> k= 4
x=Decimal   7 (0x07): 0000 0111 -> y= 0111
x=Decimal 250 (0xFA): 1111 1010 -> y= 1010
250%2^4 = 250%16=10 =0b1010
```

- Signed

```
short x;
char y;
y = x ; //fra 16 -> 8 bits
```

8 lsb (mindst betydende bits) bevares

$y=B2T_{w'}(U2B_{w'}(x \bmod 2^{w'}))$ resten ved modulo kan blive $2^{w'}$ off

```
Ex. med w=8 -> w= 4
x=Decimal  7       :0000 0111 -> y= 0111 (7 dec)
x=Decimal  23      :0001 0111 -> y= 0111 (7 dec)
x=Decimal -6       :1111 1010 -> y= 1010 (-6 dec)
x=Decimal -70      :1011 1010 -> y= 1010 (-6 dec)
x=Decimal -128     :1000 0000 -> y= 0000 (0 dec)
```

```c
int truncate_u() {
  unsigned short x= 0xBCD;
  unsigned char y;
  y=x;
  printf("x=%d -> y=%d \n",x,y);
  printBits(sizeof(x), &x); printf("--->");
  printBits(sizeof(y), &y);
  //prints bit pattern for CD

  x=0xABCD;
  y=x;
  printf("x=%x -> y=%x \n",x,y);
  printBits(sizeof(x), &x); printf("--->");
  printBits(sizeof(y), &y);
   //prints bit pattern for CD
}
int truncate_s() {
  short x= 0xBCD;
  char y;
  y=x;
  printf("x=%d -> y=%d \n",x,y);
  printBits(sizeof(x), &x); printf("--->");
  printBits(sizeof(y), &y);
   //prints bit pattern for CD

  x=0xABCD;
  y=x;
  printf("x=%d -> y=%d \n",x,y);
  printBits(sizeof(x), &x); printf("--->");
  printBits(sizeof(y), &y); //prints bit pattern for CD

  x=-32768;
  y=x;
  printf("x=%d -> y=%d \n",x,y);
  printBits(sizeof(x), &x); printf("--->");
  printBits(sizeof(y), &y);
}
```

```
caos@caos-virtual-machine:~/Skrivebord$ gcc printBits.c
caos@caos-virtual-machine:~/Skrivebord$ ./a.out
x=3021 -> y=205
0000101111001101
--->11001101
x=abcd -> y=cd
1010101111001101
--->11001101
x=3021 -> y=-51
0000101111001101
--->11001101
x=-21555 -> y=-51
1010101111001101
--->11001101
x=-32768 -> y=0
1000000000000000
--->00000000
caos@caos-virtual-machine:~/Skrivebord$
```

Bit mønster for 0xCD

"Høk æ Hak" operationer (Shift-operations)

- Venstre skifte: x << y
- Højre skifte: x >> y
  - Logisk skift
    - Fyldes med 0 til venstre
  - Aritmetisk skift
    - Gentag msb til højre

Multiplikation med operand, som er en potens af 2:

`u << k` giver `u * 2`^{k}

Aritmetisk vs. logisk højreskift

- `int x;`
- `unsigned int x;`
- `x>>4 =?`
- `y>>4 =?`

```
17 int shift(){
18   char x=-128; //chars are signed!
19   unsigned char y =128;
20
21   printf("x=%d\t\ty=%u\n",x,y);
22
23   printf("x: "BYTE_TO_BINARY_PATTERN"\ty:"BYTE_TO_BINARY_PATTERN"\n",
   BYTE_TO_BINARY(x) ,BYTE_TO_BINARY(y));
24
25   x=x>>4;
26   y=y>>4;
27
28    printf("x=%d\t\ty=%u\n",x,y);
29  printf("x: "BYTE_TO_BINARY_PATTERN"\ty:"BYTE_TO_BINARY_PATTERN"\n",
   BYTE_TO_BINARY(x) ,BYTE_TO_BINARY(y));
30
31 }
```

```
caos@caos-virtual-machine: ~/Skrivebord
caos@caos-virtual-machine:~/Skrivebord$ ./hello

x=-128          y=128
x: 10000000     y:10000000
x=-8            y=8
x: 11111000     y:00001000
caos@caos-virtual-machine:~/Skrivebord$ '
```

QUIZZ: Binære Operationer

Beregn resultatet af følgende udtryk

Q1

Udtryk	Værdi
x	11001000
y	10101010
x | y	

Q2

Udtryk	Værdi
x	11001000
y	10101010
x ^ ~y	

Q3

Udtryk	Værdi
y	10101010
y << 4	

Q4

Udtryk	Værdi
x	11001000
x >>3 (logisk)	

Q5

Udtryk	Værdi
x	11001000
x >> 3 (aritmetisk)	

Q6

Udtryk	Værdi
x	01001000
x >> 3 (aritmetisk)	

Reelle tal og Floats

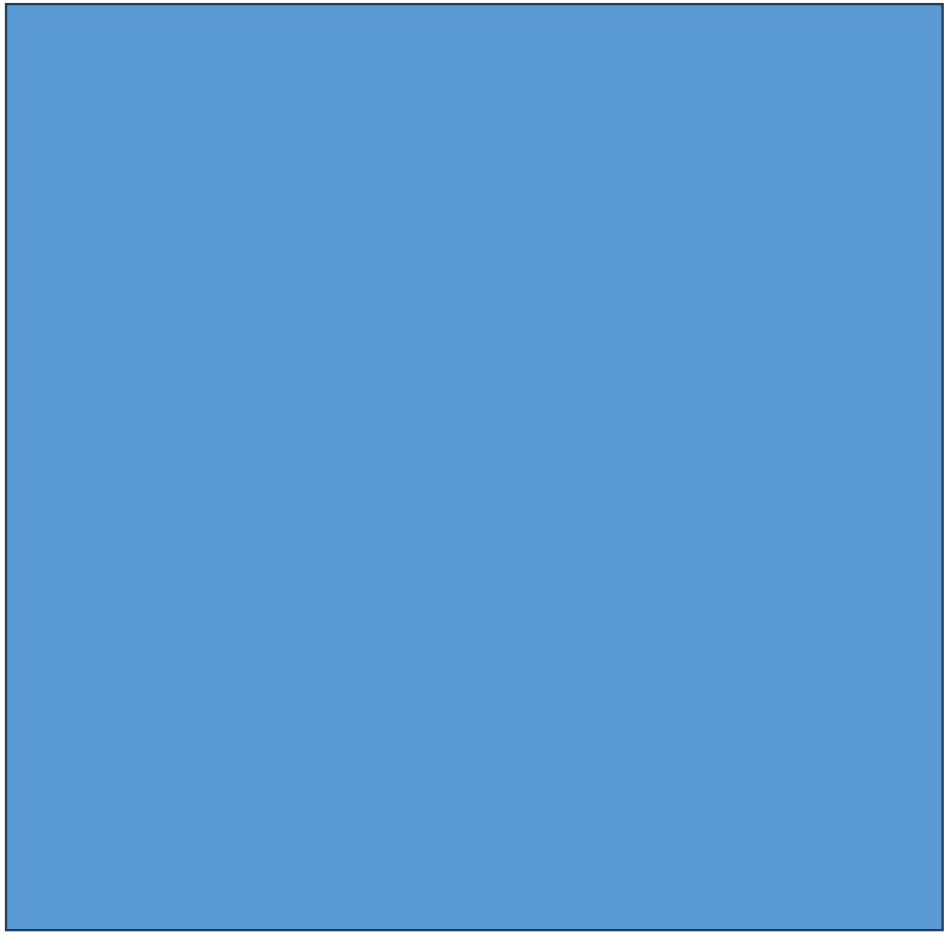
- Forstå grundideen i float-repræsentationen
- Forstå begrænsninger i repræsentationen
- Afrunding

Quizz: Floating Point

How many iterations will the loop in the following program execute?

```
#include<stdio.h>
int main () {
  for (double i = 10; i != 0; i = i - 0.1) {
     printf("%.15f\n",i);
   }
}
```

- a. 99
- b. 100
- c. 0
- d. A lot!
- e. 101



Øvelserne

- Talområder, twos complement
- Blanding af signed, unsigned
- Detektion af overløb
- Bitshift, aritmetik med bit-shift
- Challenge 0: hvordan bytes kan fortolkes som forskellig information!!
- Challenge 1: Røve en bank!

Forsøg løsning (med hjælpelærer) inden I går til løsningerne!

Start med øvelserne senest 10.15!