Gruppe 7 og 18 - IN1020
Uke 37: Datahistorie og Digital representasjon

Jehan Khodabocus
jehank@uio.no

# PLAN FOR TIMEN

- Kort repetisjon av pensum
- Ukeoppgave

# DIGITAL REPRESENTASJON – ALT ER BIT!

- Alt som lagres og gjøres i en datamaskin er lagret eller gjøres som bit.
    - tekst, tall, bilder og lyd.
- Bit er en slags forkortelse for binary digit, og representerer noe som har to tilstander.
- Eks: 0 eller 1, Sant eller Usant, Rød eller Grønn, Strøm av eller Strøm på og 0 volt eller 5 volt



# BINÆR TALL

- Å telle med binære tall er som å late som at man bare har to fingre.
- Vi gjør som i 10-tallssystemet.
  - Vi øker alltid siste tall med 1, hvis vi ikke har flere siffer, så setter vi tallet til 0 og gjentar for sifferet til venstre.
- Notasjon: For å vite hvilken base som benyttes skriver vi binær tall som $1001_2$ og desimale tall som $1001_{10}$

| Decimal | Binary |
|---|---|
| 1 | 0 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |

# LITT MATTE

- Moderne tallsystemer er posisjonsbaserte, og sifrene har vekt i henhold til posisjonen.
- I det vanlige desimale tallsystemet har posisjonene vekt.

| 1 | 10 | 100 | 1000 | 10000 | 100000 |
|---|---|---|---|---|---|
| $10^0$ | $10^1$ | $10^2$ | $10^3$ | $10^4$ | $10^5$ |

- Også i det binære tallsystemet har posisjonen tilsvarende vekt

| 1 | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|
| $2^0$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ | $2^5$ |

# DESIMAL TIL BINÆR

- Hvordan regner man et desimal tall (for eks. $37_{10}$) til binært?
  - å dele på 2 og se på resten

| | *Verdi* | *Rest* |
|---|---|---|
| 2 | 37 | 1 |
| 2 | 18 | 0 |
| 2 | 9 | 1 |
| 2 | 4 | 0 |
| 2 | 2 | 0 |
| | 1 | |

*Svaret leser vi nedenfra:* $37_{10} = 100101_2$

*Tilbake igjen:* $1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 37$

# HEKSADESIMAL NOTASJON

- Binære tall blir fort ekstremt lange/store og det er lett å gjøre feil når man jobber med binære tall
- For eks. $1\ 000\ 000_{10}$ = $11110100001001000000_2$
- Et tallsystem med 16 forskjellige siffer dvs. 0-9 og A-F representerer verdiene $0\text{-}15_{10}$
- Da blir $1\ 000\ 000_{10}$ = $F4240_{16}$

| Binary | Decimal | Hexadecimal |
|---|---|---|
| 0001 | 1 | 1 |
| 0010 | 2 | 2 |
| 0011 | 3 | 3 |
| 0100 | 4 | 4 |
| 0101 | 5 | 5 |
| 0110 | 6 | 6 |
| 0111 | 7 | 7 |
| 1000 | 8 | 8 |
| 1001 | 9 | 9 |
| 1010 | 10 | A |
| 1011 | 11 | B |
| 1100 | 12 | C |
| 1101 | 13 | D |
| 1110 | 14 | E |
| 1111 | 15 | F |

$$\underbrace{0000}_{0x0}\ \underbrace{0001}_{0x1}\ \underbrace{0010}_{0x2}\ \underbrace{0011}_{0x3}\ \underbrace{0100}_{0x4}\ \underbrace{0101}_{0x5}\ \underbrace{0110}_{0x6}\ \underbrace{0111}_{0x7}$$
$$\underbrace{1000}_{0x8}\ \underbrace{1001}_{0x9}\ \underbrace{1010}_{0xA}\ \underbrace{1011}_{0xB}\ \underbrace{1100}_{0xC}\ \underbrace{1101}_{0xD}\ \underbrace{1110}_{0xE}\ \underbrace{1111}_{0xF}$$

$$1000\ 000_{10} = 1111|0100|0010|0100|0000_2$$
$$= 0xF4240_{16} \qquad\qquad 4 \quad 2 \quad 4 \quad 0$$

# DESIMAL TIL HEKSADESIMAL

- Hvordan regner man direkte et desimal tall (for eks. $201_{10}$ eller $1128_{10}$) til heksadesimal?
  - å dele på 16 og se på resten
  - Vi ganger verdien som er etter desimalen med 16

| | *Verdi* | *Resultat* | *Rest* |
|---|---|---|---|
| 16 | 1128 | 70.5 | 8 |
| 16 | 70 | 4.37 | 6 |
| 16 | 4 | 0.25 | 4 |
| | | 0 | |

| | *Verdi* | *Resultat* | *Rest* |
|---|---|---|---|
| 16 | 201 | 12.5625 | 9 |
| 16 | 12 | 0.75 | 12 = C |
| | | 0 | |

*Svaret leser vi nedenfra:* $1128_{10} = 468_{16}$

*Tilbake igjen:* $4 * 16^2 + 6 * 16^1 + 8 * 16^0 = 1128$

*Svaret leser vi nedenfra:* $201_{10} = C9_{16}$

*Tilbake igjen:* $12 * 16^1 + 9 * 16^0 = 201$

# 1 BYTE = 8 BIT

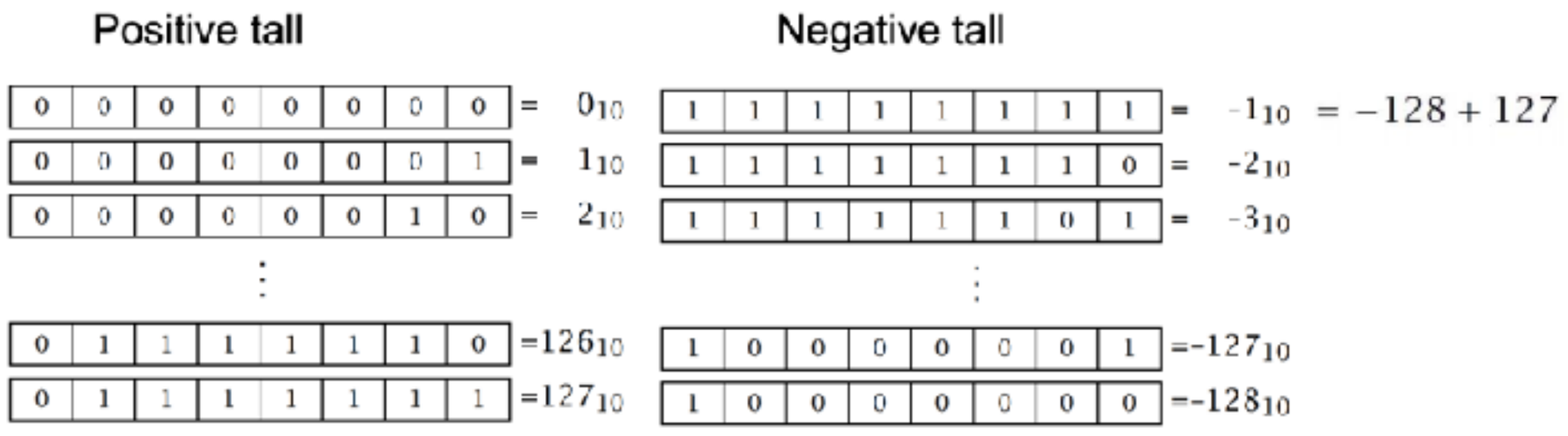
- I en datamaskin kan vi ikke hente ut 1 og 1 bit.
- I stedet lagrer vi dem som byte (dvs grupper av 8 bit):

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | = | $0_{10}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | = | $1_{10}$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | = | $2_{10}$ |
| ⋮ | | | | | | | | | |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | = | $254_{10}$ |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | = | $255_{10} = 2^8 - 1$ |

Vi kan også lagre tall i 2, 4 eller 8 byte satt sammen.

# NEGATIVE TALL

- For å kunne lagre både negative og positive tall, må vi bestemme oss for en representasjon av dette.
- Dette gjør vi på en form kalt ***2-er-komplement***:
    - Øverste bit (gjerne kalt ***fortegnsbit***) tolkes som det ***negative*** av den verdien det ellers ville hatt.



# EKSAMENSSPØRSMÅL 2019

- Hvordan lagres verdien 57 i en byte (dvs 8 bit)? Skriv bit-ene i hver sin rute.
- Og hvordan lagres verdien -57 når vi lagrer tall som 2-er-komplement?

# SVAR

### 3.2 Binærtall

Hvordan lagres verdien **57** i én byte (dvs 8 bit)? Skriv bit-ene i hver sin rute.

[ ] (0) [ ] (0) [ ] (1) [ ] (1) [ ] (1) [ ] (0) [ ] (0) [ ] (1)

og hvordan lagres verdien **-57** når vi lagrer tall som **2-er-komplement**?

[ ] (1) [ ] (1) [ ] (0) [ ] (0) [ ] (0) [ ] (1) [ ] (1) [ ] (1)

Maks poeng: 4

# PRIMITIVE DATATYPER

- Java tilbyr følgende datatyper for å lagre heltall hvor negative tall bruker 2-er-komplement
- Hver gang vi definerer en variabel i Java, må vi spesifisere hvilken datatypen den er.
- Alle representerer et visst antall bit og har en maksimumverdi og minimumsverdi som må tas hensyn til.

| | | |
|---|---|---|
| **byte** | (1 byte, 8 bit) | -128 til 127 |
| **short** | (2 byte, 16 bit) | -32 768 til 32 767 |
| **int** | (4 byte, 32 bit) | -2 147 483 648 til 2 147 483 647 |
| **long** | (8 byte, 64 bit) | -9 223 372 036 854 775 808 til 9 223 372 036 854 775 807 |

- Det finnes også andre datatyper som **boolean** med verdier true eller false; **float** (4 byte, 32 bit) og **double** (8 byte, 64 bit).

| | | Minste | Største |
|---|---|---|---|
| **float** | 32 bit | $1{,}2 \cdot 10^{-38}$ | $3{,}4 \cdot 10^{38}$ |
| **double** | 64 bit | $2{,}2 \cdot 10^{-308}$ | $1{,}8 \cdot 10^{308}$ |

# OVERFLYT

- Cellene i minnet er av en fast størrelse og kan bare lagre et begrenset antall verdier.
- Hvis vi forsøker å lagre tall som er større enn cellen får vi et problem -> overflyt
- Overflyt er når vi har for store verdier til å sette inn i tilgjengelig lagerplass.
- En måte å håndtere overflyt er å bytte datatypen. For eks. ***long*** istedenfor av en ***int***

**Example: 8-bit overflow**

An example of an 8-bit overflow occurs in the binary sum 11111111 + 1 (denary: 255 + 1).

```
  1 1 1 1 1 1 1 1
+ 0 0 0 0 0 0 0 1
-----------------
1 0 0 0 0 0 0 0 0
```

The total is a number bigger than 8 digits, and when this happens the **CPU drops the overflow digit** because the computer cannot store it anywhere, and **the computer thinks 255 + 1 = 0**.




*Source: BBC (https://www.bbc.co.uk/bitesize/guides/zjfgjxs/revision/3)*

# EKSEMPEL AV EN OVERFLYT CASE

- The Explosion of the Ariane 5
- https://www-users.cse.umn.edu/~arnold/disasters/ariane.html#:~:text=On%204%20June%201996%2C%20the,path%2C%20broke%20up%20and%20exploded

# UKEOPPGAVE

- https://www.uio.no/studier/emner/matnat/ifi/IN1020/h21/ukeoppgaver/oppgaver-digrepr.pdf

# EKSAMENSSPØRSMÅL 2019

- Tall på ulik form
- Til venstre er vist fire ulike tall og de er gitt på ulik form. Sorter disse verdiene etter størrelse (størst, nest størst, nest minst, minst)
    - 2013 (oktalt)
    - 40E (hex)
    - 1031 (desimalt)
    - 100 0000 1001 (binært)

# SVAR

- 2013 (oktalt) =

$8^3 \times 2 + 8^2 \times 0 + 8^1 \times 1 + 8^0$
$\times 3 = 1024 + 8 + 3$
$= 1035_{10}$

- 40E (hex) =

$16^2 \times 4 + 16^1 \times 0 + 16^0 \times 14$
$= 1024 + 14 = 1038_{10}$

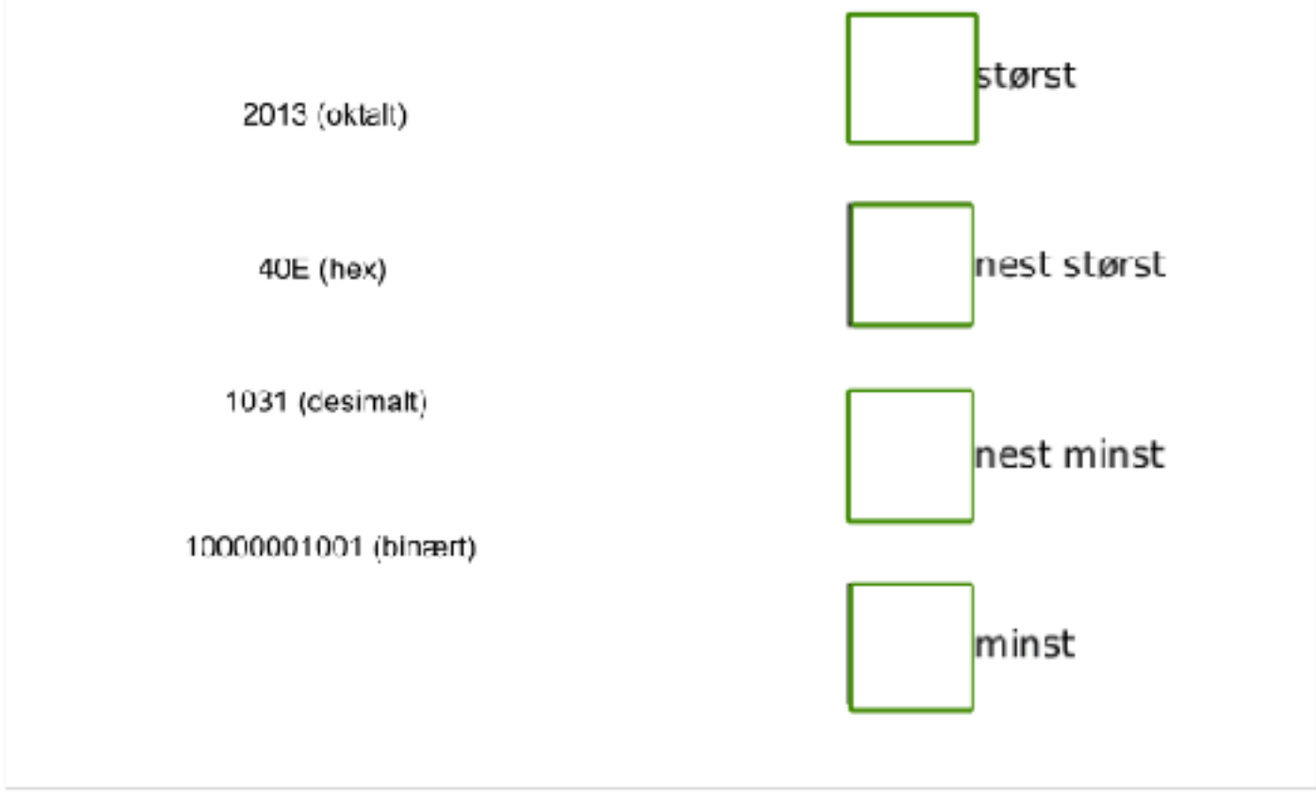
- 1031 (desimalt) = $1031_{10}$

- 10000001001 (binært) =

$2^{10} \times 1 + 2^3 \times 1 + 2^0 \times 1$
$= 1024 + 8 + 1 = 1033_{10}$

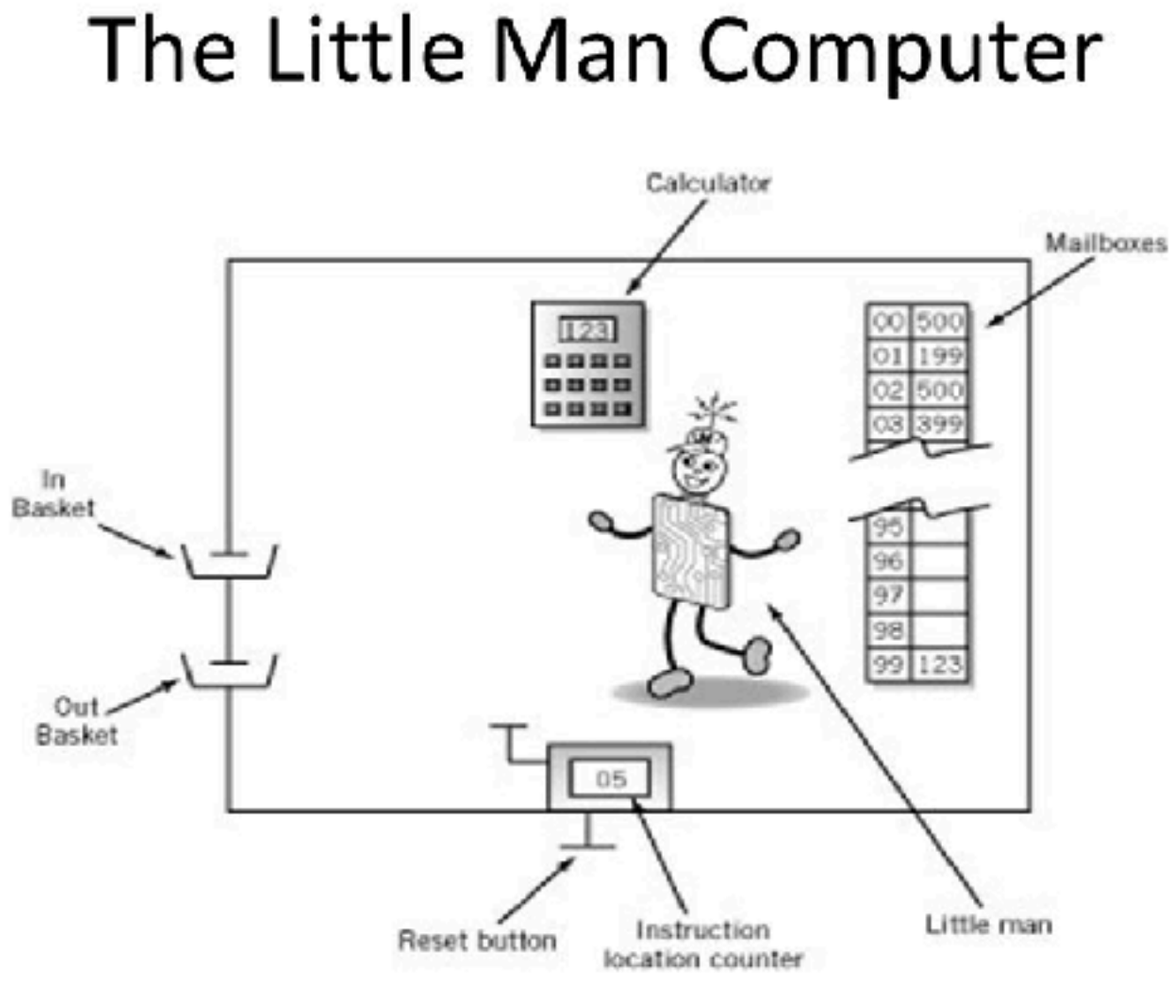
**3.1 Tall på ulik form**

Til venstre er vist fire ulike tall og de er gitt på ulik form. Sorter disse verdiene etter størrelse og bruk datamusen til å flytte dem til riktig plass til høyre.




Maks poeng: 4

# NESTE UKE


The Little Man Computer
Calculator
Mailboxes
123
00 500
01 199
02 500
03 399
95
96
97
98
99 123
In Basket
Out Basket
05
Reset button
Instruction location counter
Little man