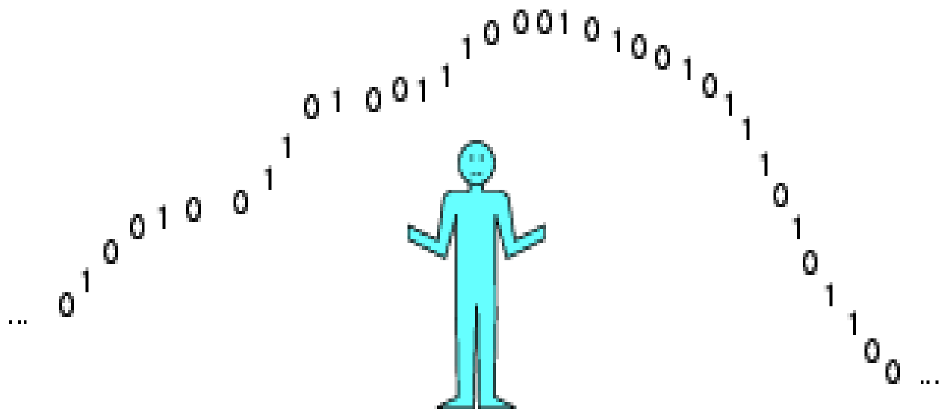
UNIVERSITETET I OSLO

INF1000: Forelesning 12

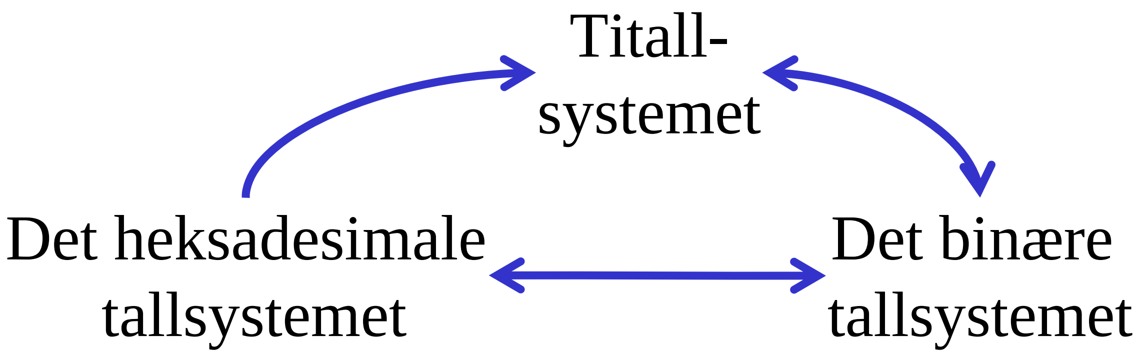
Digital representasjon av tall og tekst



Læringsmål

UNIVERSITETET I OSLO

- Kunne binærtall og heksadesimale tall og konvertering mellom ulike tallsystemer:



- Forstå prinsippene for hvordan tegn og tekst kan representeres ved hjelp av bits og bytes, og kjenne til en del sentrale standarder:
  - ASCII og ISO 8859 ”alfabetsuppen”
  - Unicode
- Forstå ulike prinsipper for representasjon av
  - negative tall
  - reelle tall

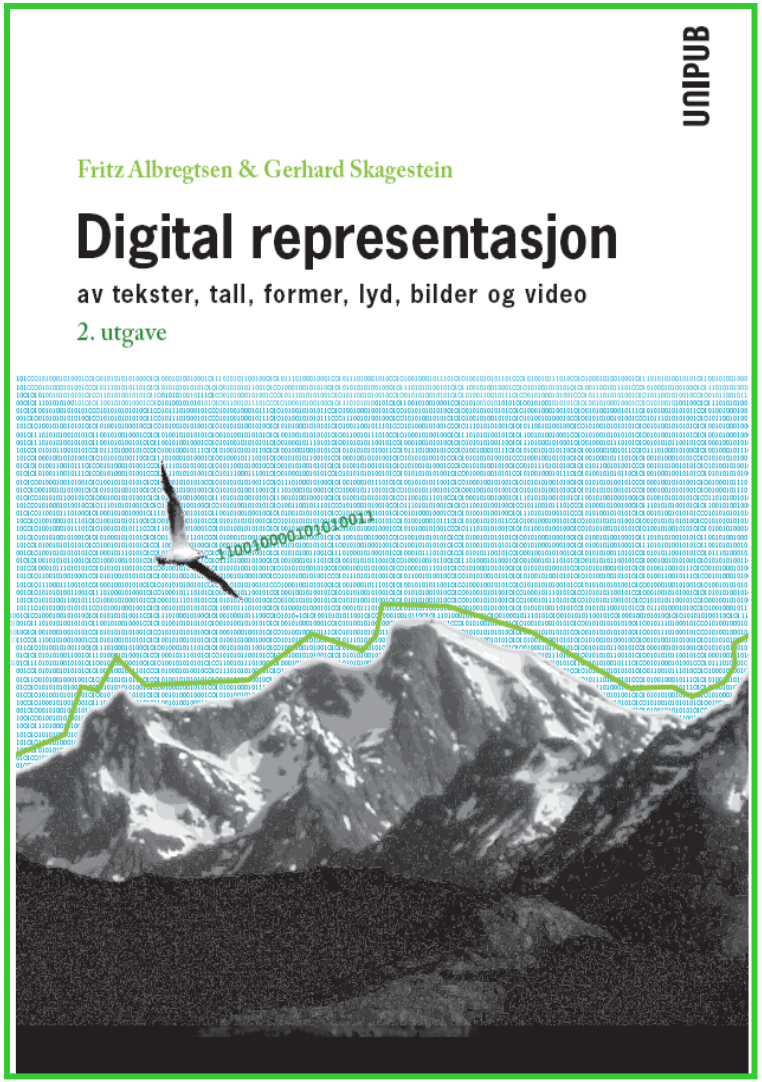
UNIVERSITETET I OSLO

Hovedkilde

Fritz Albregtsen &
Gerhard Skagestein:
Digital representasjon
av tekster, tall, former,
lyd, bilder og video

Kapittel 1,2,6 og 7.

(Lærebok i tidligere INF1040)



UNIVERSITETET I OSLO

BITS OG BYTES

Datamaskinverdenen er binær digital

- 2 diskrete verdier, 0 og 1
- 0 og 1 kalles binære sifre – binary digits – bits
- Alt i datamaskinen er representert ved sekvenser av bits – bitmønstre
- Moderne datamaskiner arbeider gjerne med grupper på 8 bits
  - En slik gruppe på 8 bits kalles en byte.

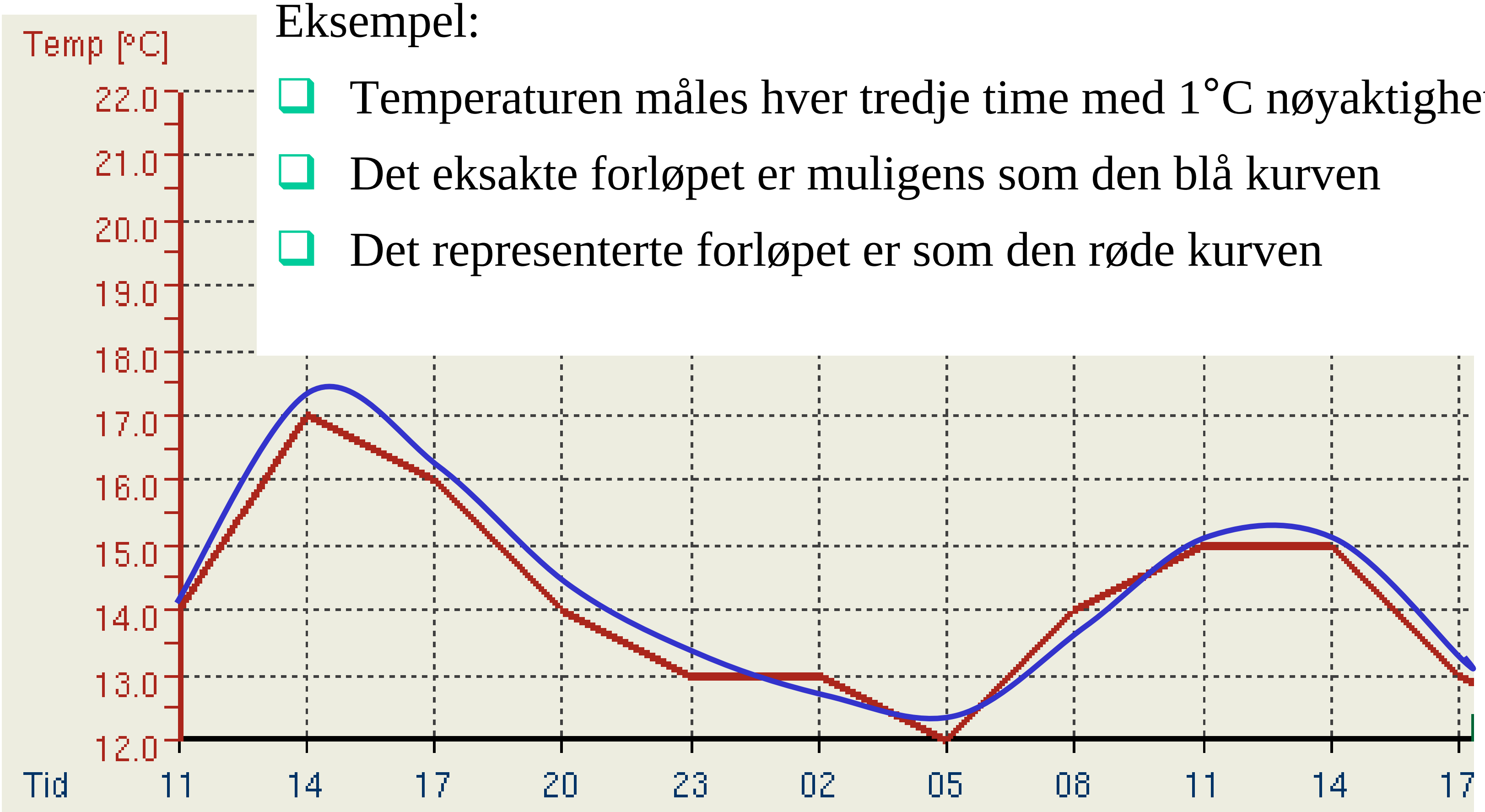
Hvorfor bare to verdier?

Analog virkelighet – digital representasjon

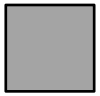
- Digital – ”som gjengir fysiske størrelser med diskrete tegn”
- Forutsetter diskretisering og kvantisering

Eksempel:

- Temperaturen måles hver tredje time med 1°C nøyaktighet
- Det eksakte forløpet er muligens som den blå kurven
- Det representerte forløpet er som den røde kurven



Hva betyr bitmønsteret?

- Bitmønsteret 10100100 kan bl.a. være:
  - 164 (tolket som binærtall)
  - -36 (negativt binærtall med fortegnsbit)
  - -90 (negativt tall på toerkomplementsform)
  - € (ISO 8859-15)
  - ¤ (Windows Codepage 1252)
  -  (som gråtone)
  - …

BINÆRTALL

Potensregning – en kort repetisjon

UNIVERSITETET I OSLO

- $\text{grunntall}^{\text{eksponent}} = \underbrace{\text{grunntall} * \text{grunntall} * \ldots * \text{grunntall}}_{\text{eksponent antall ganger}}$
- Spesialregel: $\text{grunntall}^0 = 1$

Titallsystemet – et posisjonssystem

- I titallsystemet (desimalsystemet) har vi 10 sifre:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Større tall konstrueres ved hjelp av et posisjonssystem:

10^6 (1 000 000)	10^5 (100 000)	10^4 (10 000)	10^3 (1 000)	10^2 (100)	10^1 (10)	10^0 (1)

Det binære tallsystemet

UNIVERSITETET I OSLO

- Også det binære tallsystemet (totallsystemet) er et posisjonssystem, denne gangen med
  - grunntall 2
  - 2 sifre: 0 og 1

2^7 (128)	2^6 (64)	2^5 (32)	2^4 (16)	2^3 (8)	2^2 (4)	2^1 (2)	2^0 (1)

Titallsystemet → binærtall

Gitt et tall x i titallsystemet.

1. Start på posisjon 0.
2. Hvis x er oddetall, sett 1 i denne posisjonen (ellers 0).
3. Sett x lik x heltallsdividert med 2.
4. Fortsett med neste posisjon til venstre, så lenge x ≠ 0.

```
int[] tilBintall(int x) {
    int[] bintall = new int[8];
    int pos = 0;

    do {
        if (x % 2 == 1) {
            bintall[pos] = 1;
        } else {
            bintall[pos] = 0;
        }
        x = x/2;
        pos++;
    } while (x != 0);

    return bintall;
}
```

Bitposisjoner og bitmønstre

- For et tall x i titallsystemet, hvor mange sifre (bits) trenger det tilsvarende binærtallet?
- Det største tallet som kan representeres med b bits er $2^b - 1$.
- Vi må altså finne den minste b'en slik at $x \leq 2^b - 1$.

Det heksadesimale tallsystemet

- I det heksadesimale tallsystemet har vi
  - grunntall 16
  - 16 sifre: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

16^5 (1 048 576)	16^4 (65 536)	16^3 (4 096)	16^2 (256)	16^1 (16)	16^0 (1)

Konvertering binært ↔ heksadesimalt

UNIVERSITETET I OSLO

Binært	Heksadesimalt
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

Binært	Heksadesimalt
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

- For å angi at noe er skrevet på heksadesimal form kan vi enten
  - føye til 16 som subskript, f.eks. $A1_{16}$, eller
  - føye til 0x i forkant, f.eks. 0xA1

TEKST

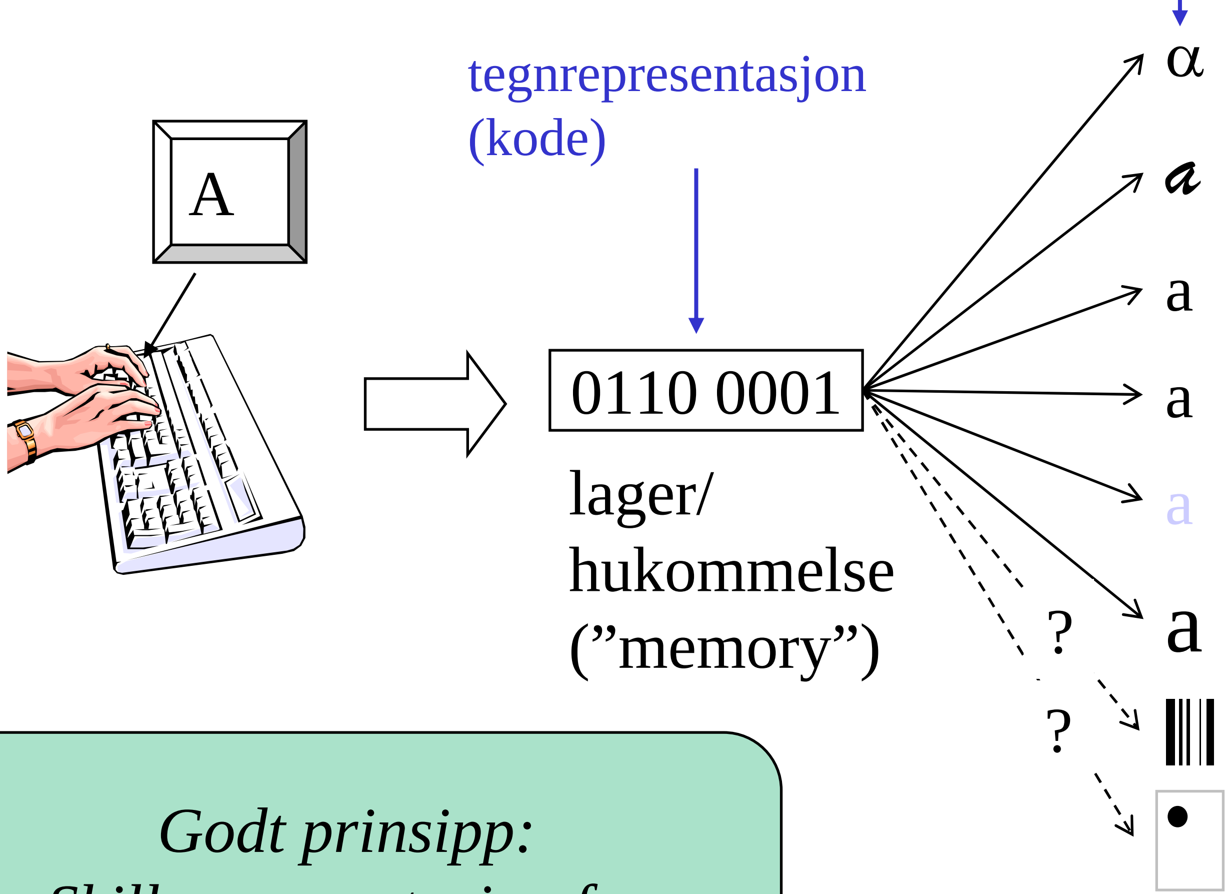
Problemstilling

- Utgangspunkt:
  Hvert tegn i teksten representeres av et unikt bitmønster.
- Eksempel:
  - E = 01000101
  - H= 01001000
  - I = 01001001
  - HEI = 01001000 01000101 01001001
- Sender (skriver) og mottaker (leser) må være enige om kodingen.
  - Vi trenger standarder!

Aktuelle spørsmål

- Hvilke tegn skal representeres?
- Hvor mange bits per tegn?
- Fast eller variabelt antall bits per tegn?
- Bør det være noen form for systematikk i bitmønstrene?

Om tegn og glyfer



Godt prinsipp:
Skill representasjon fra fremvisning (”rendering”)

ASCII kodetabell

- American Standard Code for Information Interchange
- 1963 →

	00	10	20	30	40	50	60	70
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	”	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

Linjeskift: LF og CR

- LF (Line Feed):
  - 0x0A
  - “Indicates movement of the printing mechanism or display cursor to the next line.“
- CR (Carriage Return):
  - 0x0D
  - “Indicates movement of the printing mechanism or display cursor to the starting position of the same line.”
- Linjeskift representeres i dag på ulike måter:
  - PC: CR + LF
  - Mac: CR
  - UNIX: LF

ISO 646-60 kodetabell

- Bygger på ASCII.
- [\ ] { | } er ofret til fordel for Æ Ø Å æ ø å
- Lignende tilpasninger er gjort i tilsvarende standarder for andre språkmiljøer.

	00	10	20	30	40	50	60	70
0	NUL	DLE	space	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	”	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	Æ	k	æ
C	FF	FS	,	<	L	Ø	l	ø
D	CR	GS	-	=	M	Å	m	å
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O			DEL

ISO 8859-1 (Latin-1) kodetabell

	00	10	20	30	40	50	60	70	80	90	A0	B0	C0	D0	E0	F0
0	NUL	DLE	space	0	@	P	`	p			no break space	°	À	Ð	à	ð
1	SOH	DC1	!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
2	STX	DC2	”	2	B	R	b	r			¢	²	Â	Ò	â	ò
3	ETX	DC3	#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
4	EOT	DC4	$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
5	ENQ	NAK	%	5	E	U	e	u	un-		¥	µ	Å	Õ	å	õ
6	ACK	SYN	&	6	F	V	f	v	defined		¦	¶	Æ	Ö	æ	ö
7	BEL	ETB	'	7	G	W	g	w			§	·	Ç	×	ç	÷
8	BS	CAN	(	8	H	X	h	x			¨	¸	È	Ø	è	ø
9	HT	EM	)	9	I	Y	i	y			©	¹	É	Ù	é	ù
A	LF	SUB	*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
B	VT	ESC	+	;	K	[	k	{			«	»	Ë	Û	ë	û
C	FF	FS	,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü
D	CR	GS	-	=	M	]	m	}			-	½	Í	Ý	í	ý
E	SO	RS	.	>	N	^	n	~			®	¾	Î	Þ	î	Þ
F	SI	US	/	?	O	_	o	DEL			¯	¿	Ï	ß	ï	Ÿ

ETSI GSM 03.38 kodetabell for SMS

UNIVERSITETET I OSLO

	00	10	20	30	40	50	60	70
0	@	Δ	space	0	¡	P	¿	p
1	£	_	!	1	A	Q	a	q
2	$	Φ	"	2	B	R	b	r
3	¥	Γ	#	3	C	S	c	s
4	è	Λ	¤	4	D	T	d	t
5	é	Ω	%	5	E	U	e	u
6	ù	Π	&	6	F	V	f	v
7	ì	Ψ	'	7	G	W	g	w
8	ò	Σ	(	8	H	X	h	x
9	Ç	Θ	)	9	I	Y	i	y
A	LF	Ξ	*	:	J	Z	j	z
B	Ø	ESC	+	;	K	Ä	k	ä
C	ø	Æ	,	<	L	Ö	l	ö
D	CR	æ	-	=	M	Ñ	m	ñ
E	Å	undef	.	>	N	Ü	n	ü

…pluss disse 10 ”escape”-sekvensene

€	ESC e
FF	ESC LF
[	ESC <
\	ESC /
]	ESC >
^	ESC Λ
{	ESC (
|	ESC @
}	ESC)
~	ESC =

Den endelige løsning? Unicode og ISO 10646

UNIVERSITETET I OSLO

- 21 bits, med mulighet for 1 114 112 kodepunkter
  - Ca 130 000 private
  - Ca 870 000 ennå ikke brukt
- Første 256 tegn identisk med ISO 8859-1
- For hvert tegn finnes
  - en representativ glyf
  - kodepunktet
  - et navn
  - klassifisering
  - skriveretning
- Vedtatte tegn med kodepunkter skal aldri endres

se http://www.unicode.org/charts/

String-metoden compareTo i Java

- Basert på Unicode-verdiene til hvert enkelt tegn i de to tekstene:

```
void sammenlign(String s1, String s2) {
    if (s1.compareTo(s2) < 0) {
        System.out.println(s1 + " " + s2);
    } else if (s1.compareTo(s2) = 0) {
        System.out.println("Like!");
    } else { // s1.compareTo(s2) > 0
        System.out.println(s2 + " " + s1);
    }
}
```

- NB: Hva skjer hvis tekstene de norske tegnene æøå?

Kodepunkt vs representasjon

- Kodepunkt:
  - Tegnets ”numeriske” verdi.
  - Det som står i kodetabellen.
- Representasjon:
  - Hvordan tegnet representeres som et bitmønster.

Unicode Transformation Formats

- UTF-32 (lite brukt):
  - Bruker 32 biter for alle tegn.
- UTF-16:
  - Tegn i BMP: 16 biter.
  - Tegn utenfor BMP: surrogatpar (32 biter).
- UTF-8:
  - Bruker 8, 16, 24, eller 32 biter avhengig av tegnet.
  - Tegnene i ASCII: 8 biter med ledende 0.

Unicode UTF-8

- Koding med variabel lengde
  - 8, 16, 24 eller 32 biter avhengig av kodepunktet

Big endian vs. Little endian

- I representasjoner som krever mer enn én byte, finnes det to mulige rekkefølger av bytene:
  - Starte med den mest signifikante (”Big endian”)
  - Starte med den minst signifikante (”Little/small endian”)
- Eksempel:
  UTF-16 Big endian for A er 0x 00 41
  UTF-16 Little endian for A er 0x 41 00
- Begge muligheter blir brukt i praksis, og dette kan gi problemer når data overføres fra et maskinmiljø til et annet!

Byte order mark (BOM)

- Et ”Byte order mark” (BOM) er tegnet ”Zero width no-break space” med kodepunkt U+FEFF i begynnelsen av en Unicode-fil.
- Siden det ikke finnes noe tegn med kodepunkt FFFE, kan BOM brukes til å finne filformatet (UTF-32, UTF-16, UTF-8 og Big eller Small endian):

Koding	BOM-bitmønster
UTF-32, big-endian	0x 00 00 FE FF
UTF-32, little-endian	0x FF FE 00 00
UTF-16, big-endian	0x FE FF
UTF-16, little-endian	0x FF FE
UTF-8	0x EF BB BF

NEGATIVE OG REELLE TALL

Ulike klasser tall

- De naturlige tallene:
  $\mathbb{N} = \{ 1, 2, 3, \dots \}$

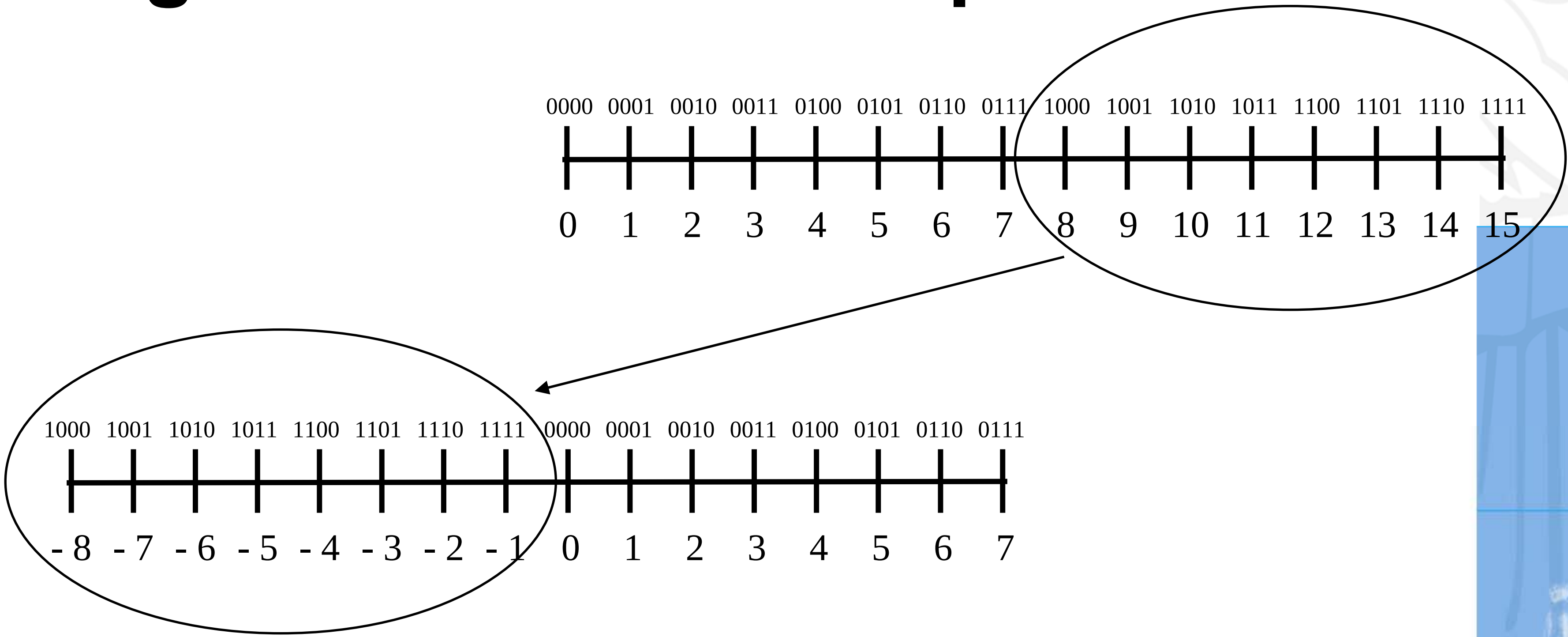
- De hele tallene:
  $\mathbb{Z} = \{ \dots, -2, -1, 0, 1, 2, \dots \}$

- De rasjonale tallene:
  $\mathbb{Q}$ = alle tall som kan skrives som en brøk

- De reelle tallene:
  $\mathbb{R}$ = alle tallene på tallinjen

UNIVERSITETET I OSLO

Negative tall: toer-komplement



- For å negere et tall:
  1. Snu alle bit'ene i tallet (0 ↔ 1).
  2. Legg til 1.

UNIVERSITETET I OSLO

Negative tall: Bias

- Et alternativ er å legge til en konstant bias til alle tallene.
- Med 8 bitposisjoner og bias 128 kan tallene fra -128 til 127 representeres ved hjelp av tallene fra 0 til 255.
- Eksempler:
  53: -21:

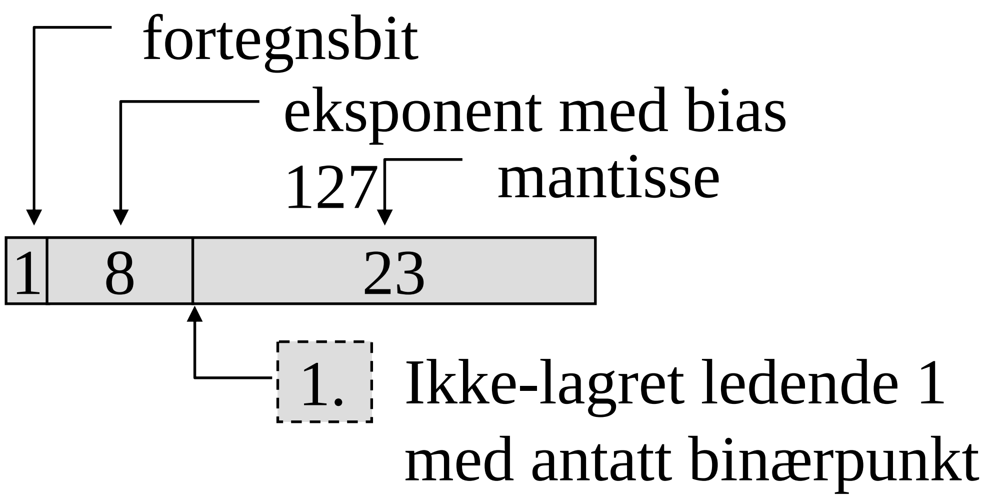
Flyttall

- I titallsystemet kan et tall skrives på formen
  mantisse * $10^{\text{eksponent}}$
- Eksempler:
  $0.5 * 10^2 = 0.5 * 100 = 50$
  $0.5 * 10^0 = 0.5 * 1 = 0.5$
  $0.5 * 10^{-1} = 0.5 * 0.1 = 0.05$
  $-5 * 10^{-1} = -0.5 * 0.1 = -0.05$

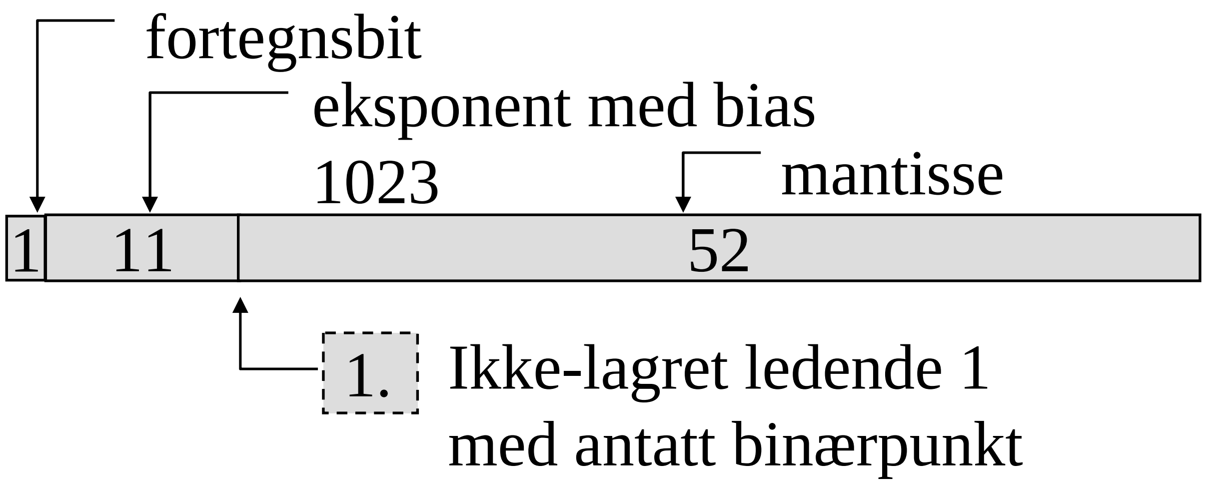
- Tilsvarende kan vi skrive binære flyttall på formen
  mantisse * $2^{\text{eksponent}}$
- For flyttall må vi altså representere både eksponent og mantisse.
  Begge må kunne være positive, negative og null.

UNIVERSITETET
I OSLO

Binære flyttall:
IEEE 754 single precision



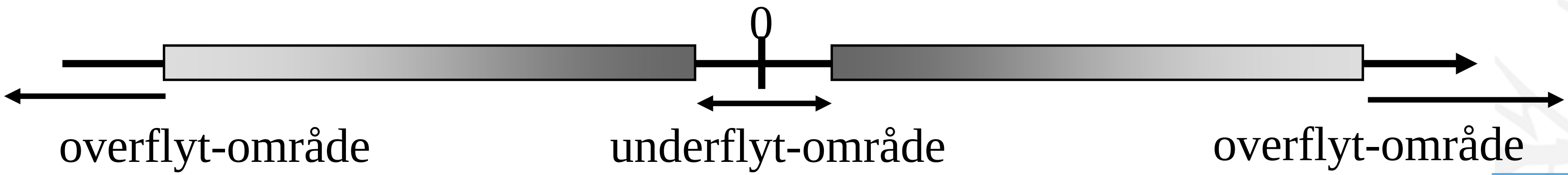
Binære flyttall:
IEEE 754 double precision



IEEE 754: Spesielle verdier

- Null: Både eksponent og mantisse er 0
- Uendelig: Eksponent med bare 1ere, mantisse med bare 0ere
- Not A Number: Eksponent med bare 1ere, mantisse ≠ 0
  - Mantisse som starter med 1 : Resultat av en udefinert operasjon (eksempel: 0/0)
  - Mantisse som starter med 0: Resultat av en ulovlig operasjon (eksempel: N/0)

Flyttallsområder i IEEE 754 og i Java



datatype	antall biter	minste positive tall	største positive tall
float	32	$2^{-126} \approx 10^{-44,85}$	$(2 - 2^{-23}) * 2^{127} \approx 10^{38,53}$
double	64	$2^{-1022} \approx 10^{-323,3}$	$(2 - 2^{-52}) * 2^{1023} \approx 10^{308.3}$