Velkommen!

Digital representasjon og datahistorie

14. september 2021

(Hanne-)Kristin S. Leiros

hkleiros@uio.no

Plan for dagen

- Kort repetisjon av noe datahistorie
- Litt mer om digital representasjon
- Spørsmål om oblig
- Jobbe med oppgaver

Åpnet for bytting av grupper

- Ta kontakt med gruppelærer av gruppen du vil bytte til
- Vent på bekreftelse om at det er ledig kapasitet
- Gi beskjed til opprinnelig gruppelærer om gruppebytte etter mottatt bekreftelse

- Vil ikke oppdateres i StudentWeb

Oblig 1 frist i morgen

Kl. 23:59

Send melding til gruppelæreren dere er satt opp hos for utsettelse

Læringsmål denne uka

- Forklare hvordan tall, tekst, bilder og annen data kan representeres via binære tall
- Kunne konvertere mellom tallsystemer
- Kunne forklare hva overflow (overflyt) er og drøfte designvalg som unngår det
- Gjøre rede for int, float, double, og andre primitive datatyper
- Ha grunnleggende kunnskap på raster vs vektorbilder, lydkoding og tekstkoding

Analoge og digitale datamaskiner

- Analoge maskiner
  - Verdier representeres av fysiske størrelser
    - Lengde, rotasjon, elektrisk spenning etc.
- Digitale maskiner
  - Verdier representeres symbolsk

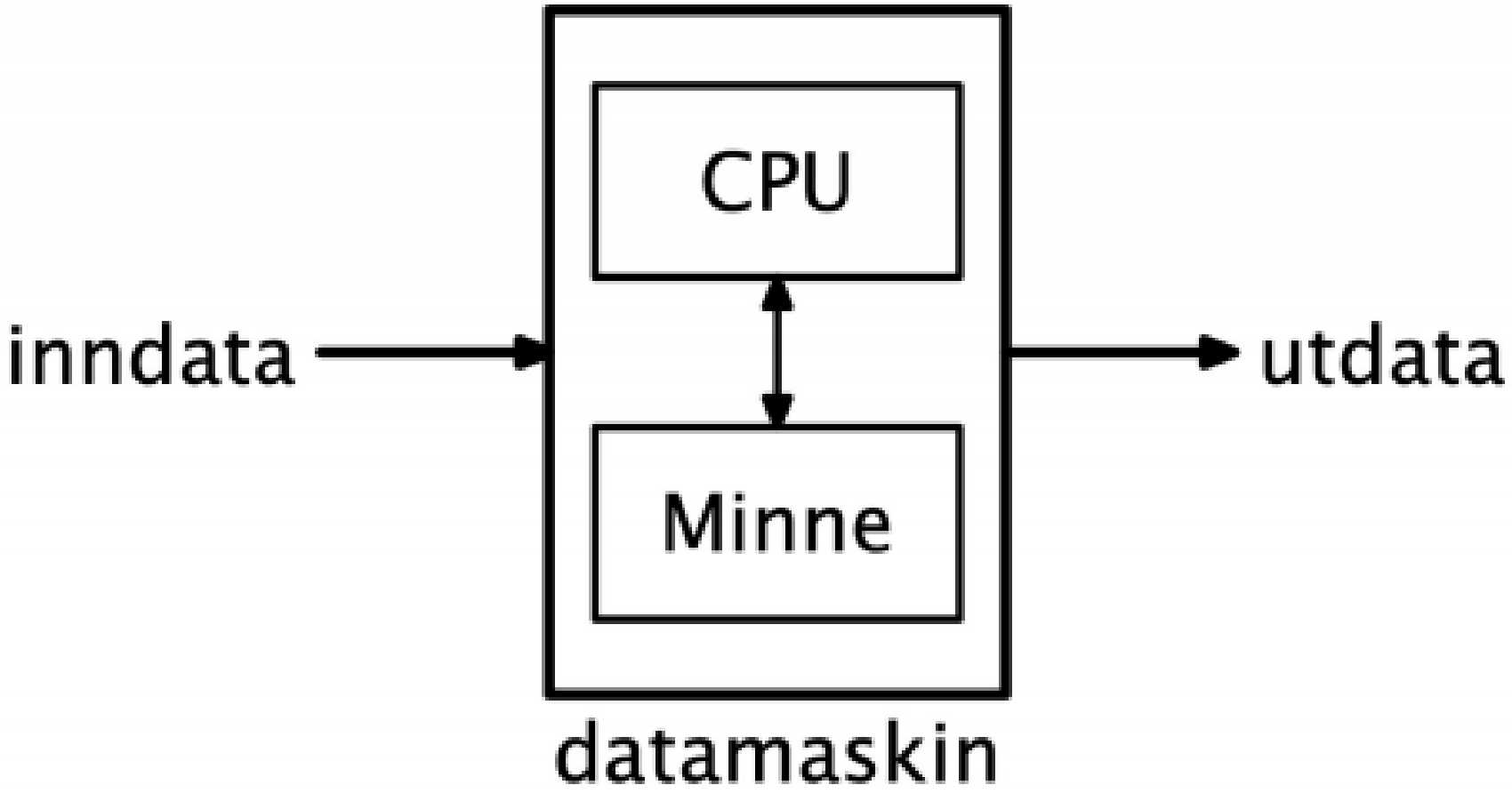
- Fram til 2. verdenskrig var det analoge maskiner som dominerte

Hva er en datamaskin?

- Kriterier som brukes i dette faget:
  - Generell
    - Kan løse alle typer beregnbare problemer
  - Programmerbar
    - Med programmet i minnet på maskinen
  - Ferdig og tatt i bruk
    - Ikke bare en idé på papiret

- Bør også være:
  - Binær
  - Elektronisk
    - Ikke mekaniske elementer / fysiske bevegelige deler

Von Neumann arkitektur



Hva er en bit?

- Bit
  - Binary digit, to tilstander f.eks. 0 og 1
  - Alt på datamaskinen er lagret som bit
    - Tall
    - Tekst
    - Bilder
    - Lyd

Binære tall

- For å kunne bruke bit som tall, må vi telle binært
- Ganske likt som når vi teller desimalt:
  - Øk siste siffer i tallet med 1
  - Hvis det ikke er flere sifre, sett siste til 0 og gjenta for sifferet til venstre
- Notasjon
  - 1001
    - 9 eller tusen og én?
    - 1001_2 og 1001_{10} tydeliggjør det

Desimal	0	1	2	3	4	5	6	7	8	9	10
Binær	0	1	10	11	100	101	110	111	1000	1001	1010

Hvordan fungerer tallsystemer egentlig?

- Posisjonsbaserte, sifrene har vekt i henhold til posisjonen

- I vårt vanlige desimale tallsystem har posisjonene vekt
  1 10 100 1000 osv.
  Dette tilsvarer
  10^0 10^1 10^2 10^3 osv.
- I det binære tallsystemet har posisjonen tilsvarende vekt
  1 2 4 8 osv.
  Dette tilsvarer
  2^0 2^1 2^2 2^3 osv.

Heksadesimal

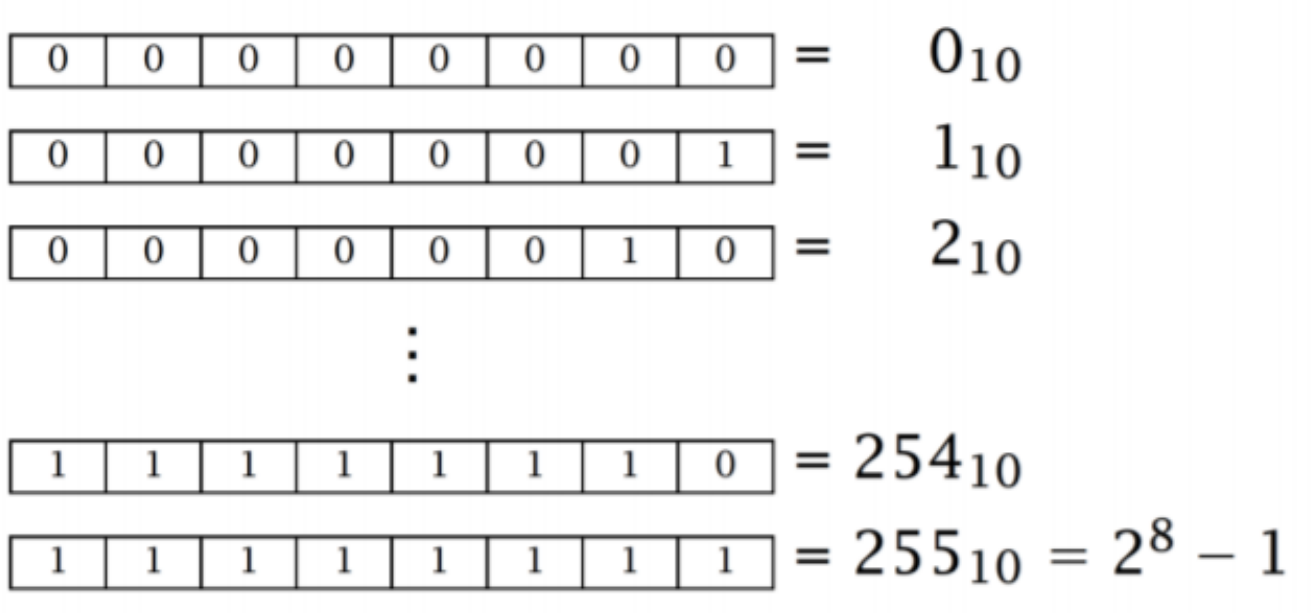
- 16 forskjellige siffer
  - 0-9 og A-F
- Notasjon:

$$1\ 000\ 000_{10} = F4240_{16} = 0xF4240$$

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

1 byte = 8 bit

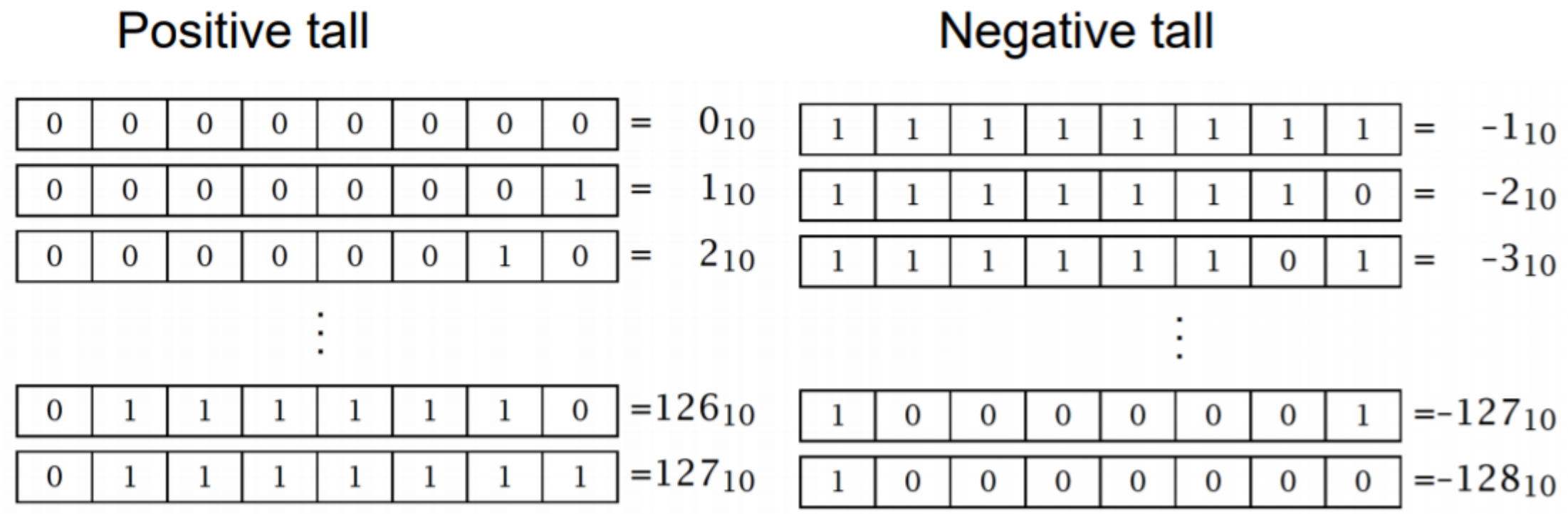
I en datamaskin kan vi ikke hente ut 1 og 1 bit; i stedet lagrer vi dem som byte (dvs grupper av 8 bit):



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

Fortegnsbit – negative tall

- 2-er komplement
  - Første, eller øverste, bit tolkes som det negative av den verdien bitet ellers ville hatt



- Heltall på binærform trenger altså:
  - Hvor mange bit vi bruker
  - Om tallene kan være negative
    - Hvordan tallene representeres negativt

Datatyper

Java tilbyr følgende datatyper for å lagre heltall (hvor negative tall bruker 2-er-komplement):

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

- Float
- Double

Overflow

- Har i flere tilfeller skapt ulykker

```
int v = 1000, v2 = v*v, v4 = v2*v2;
```

En **int** i Java har plass til 4 byte (32 bit)

v:	1 000 =	00000000	00000000	00000011	11101000	
v2:	1 000 000 =	00000000	00001111	01000010	01000000	
v4:	1 000 000 000 000 =	11101000	11010100	10100101	00010000	00000000	= -727379968

Representasjon av tekst

- Tabell der hvert tegn har nummer og lagrer nummeret for hver bokstav

- Men alle alle må buke samme tabell
- Først hadde alle hver sin tabell
- Ble behov for at maskiner skal kunne kommunisere

1. ASCII
   - Ble brukt av de flreste
   - Lett å sjekke om et tegn er et siffer eller en boksav
   - Lett å konvertere bokstaver
   - Mangler ÆØÅ og mange andre tegn og bokstaver

ASCII

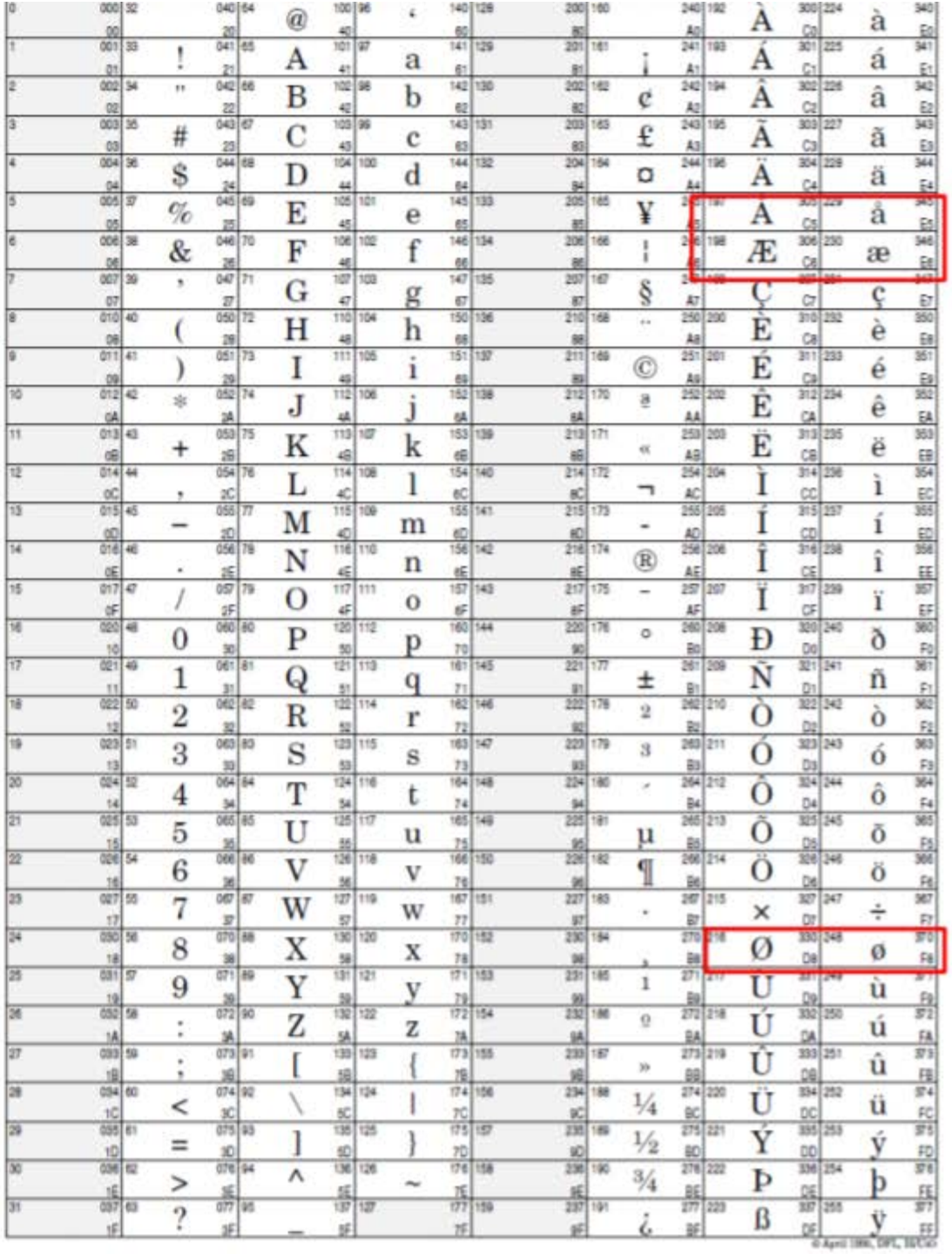
Første vellykkede forsøk på standardisering var ASCII (American Standard Code for Information Interchange) i 1963; den har 128 tegn:

ASCII Code Chart

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

Latin-1 / ISO 8859-1

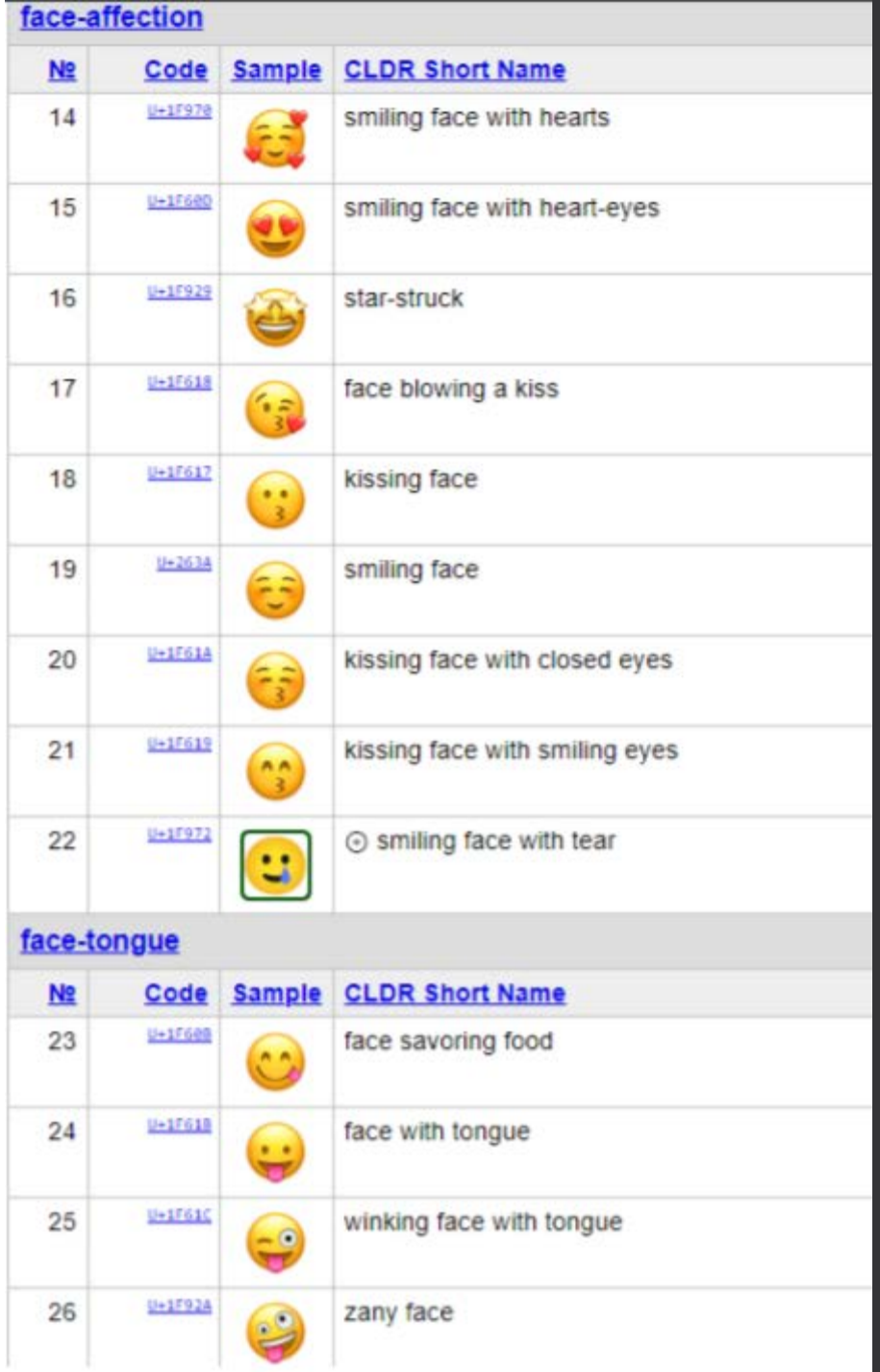
- En klart bedre løsning ble ISO 8859-1 (Latin-1) med 256 tegn.
- Den (og varianter) er ennå i bruk.
- Hvert tegn bruker 1 byte (8 bit)



Unicode

- Det meste nyere programvare støtter unicode
- Den «ordentlige» løsningen på problemet med antall tegn
- Omfatter alle skriftspråk i verden både moderne og antikke, levende og døde

- Plass til drøyt 1 000 000 tegn
- Nesten 150 000 tegn i dag! https://home.unicode.org/basic-info/overview/



face-affection

№	Code	Sample	CLDR Short Name
14	U+1F970		smiling face with hearts
15	U+1F60D		smiling face with heart-eyes
16	U+1F929		star-struck
17	U+1F618		face blowing a kiss
18	U+1F617		kissing face
19	U+263A		smiling face
20	U+1F61A		kissing face with closed eyes
21	U+1F619		kissing face with smiling eyes
22	U+1F972		⊙ smiling face with tear

face-tongue

№	Code	Sample	CLDR Short Name
23	U+1F60B		face savoring food
24	U+1F61B		face with tongue
25	U+1F61C		winking face with tongue
26	U+1F92A		zany face

Unicode forts

- UTF-8 (8-bit Unicode Transformation Format) bruker 1 til 4 byte per tegn
- Bakover kompatibel med ASCII (de første 128 tegnene er samme)

A	U+0041	01000001
Å	U+00C5	11000011 10000101
€	U+20AC	11100010 10000010 10101100
𐎄	U+10384	11110000 10010000 10001110 10000100

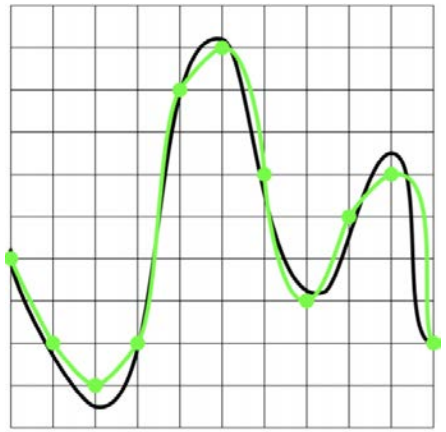
Digital representasjon: bilder

- Pixler
  - Bildepunkt som består av rød, grønn og blå farger kan lyste svakt eller sterkt

- Lagrer fargeverdiene også hvor mange pixler med samme farge som kommer etter hverandre
  - «run length-koding»
    - Brukt at PNG og GIF

Digital representasjon: Lyd

- Lagre styrken til lydbølgene med faste intervaller for å gjenskape lyden



- Kvalitetne på lyd avhenger av:
  - Hvor ofte vi måler lydbølgene
  - Hvo mange trinn vi har å plassere målingene på

Ukesoppgaver

Viktig å få mengdetrening i tallsystemer

Neste uke:

- LMC
- litt Assembler