



NTNU

Kunnskap for en bedre verden

Teori – oppsummering (HW & DR)

Rune Sætre (satre@idi.ntnu.no)

Tema

- **Datamaskinens oppbygging**
- **Digital representasjon**
- Nettverk
- Algoritmer
- Systemutvikling

Datamaskinens oppbygging



Hovedenhet

CPU/RAM/Disk

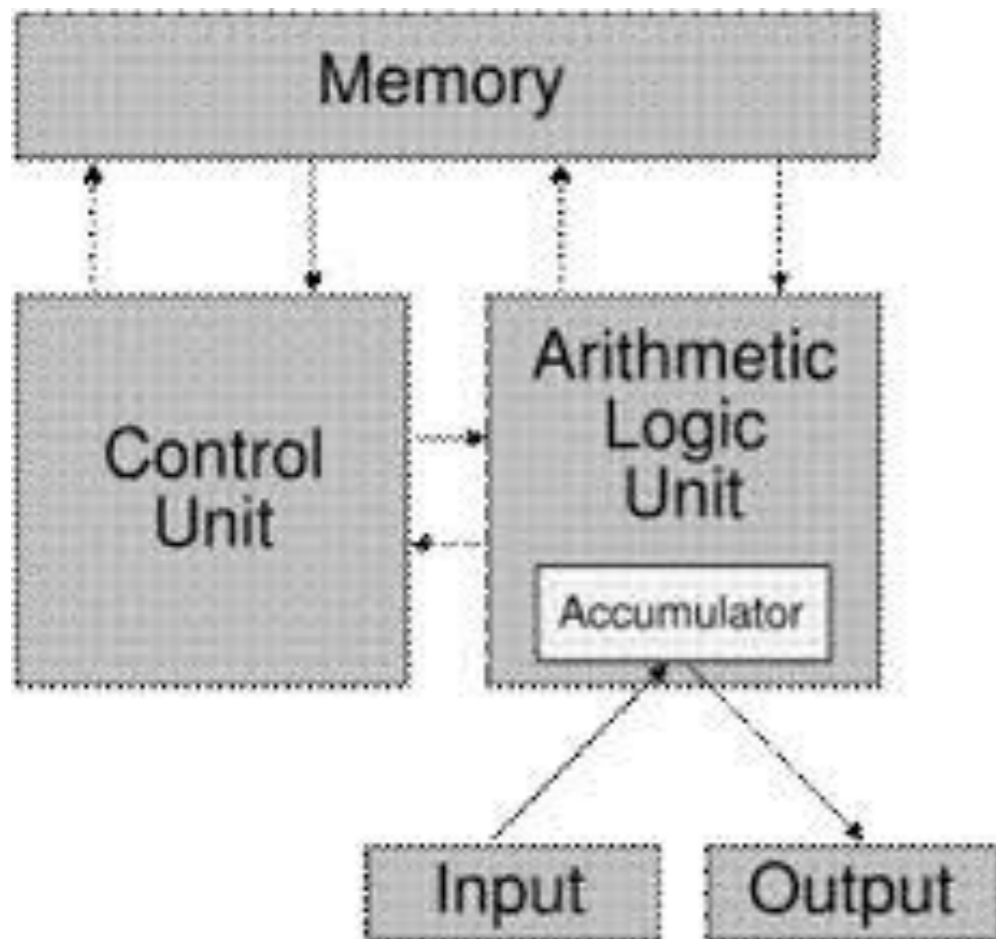
Inputenheter

Tastatur, Mus, Penn,
Finger, Scanner

Outputenheter

Skjerm
Printer
Høytalere

CPU oppbygging



- Aritmetisk-logisk enhet
 - Beregninger
 - Mellomlagring
 - Registre (Akkumulator, PC)
- Kontrollenhet
 - Kontroll av utførelse
- Hukommelse
 - Lagring av resultater
- Buss
 - Frakting av data
 - Kommunikasjon mellom enheter

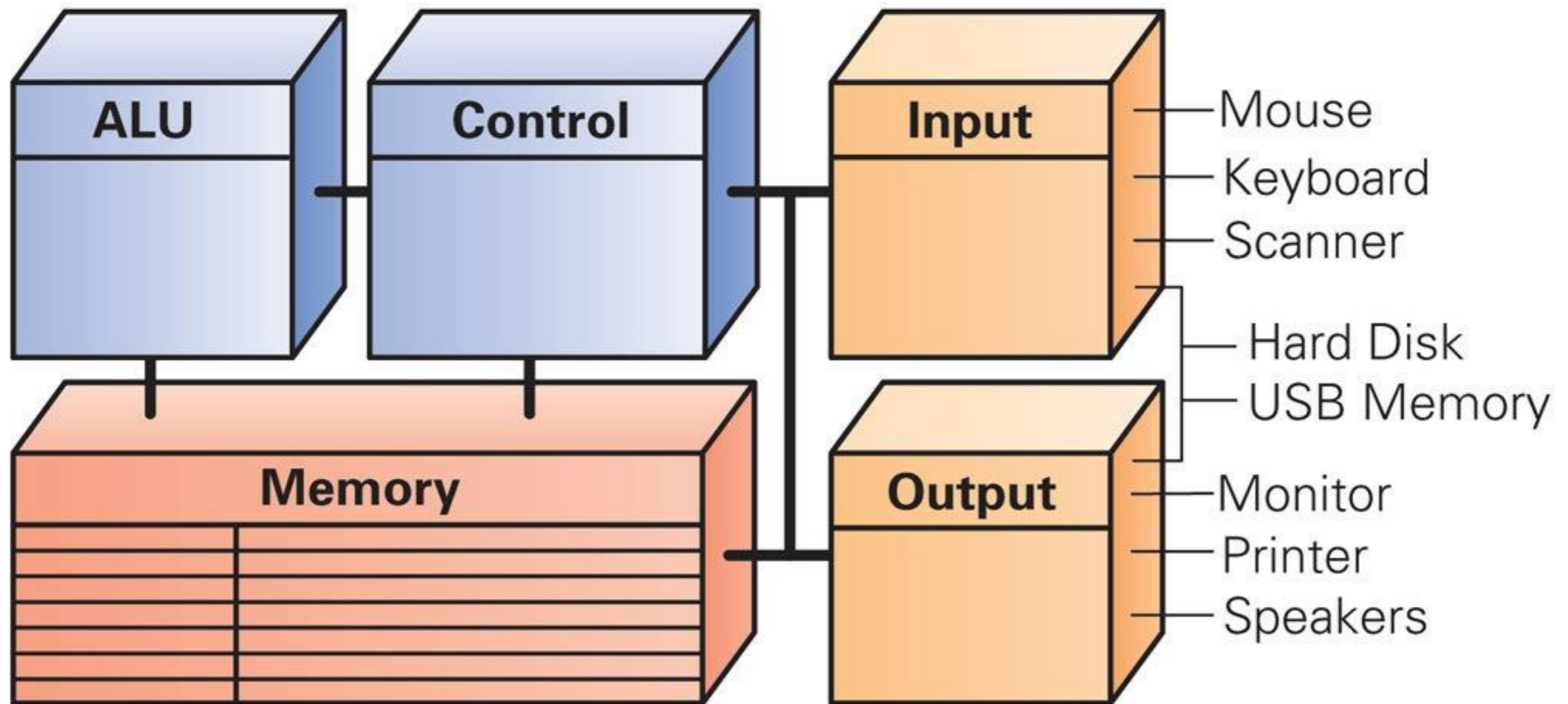
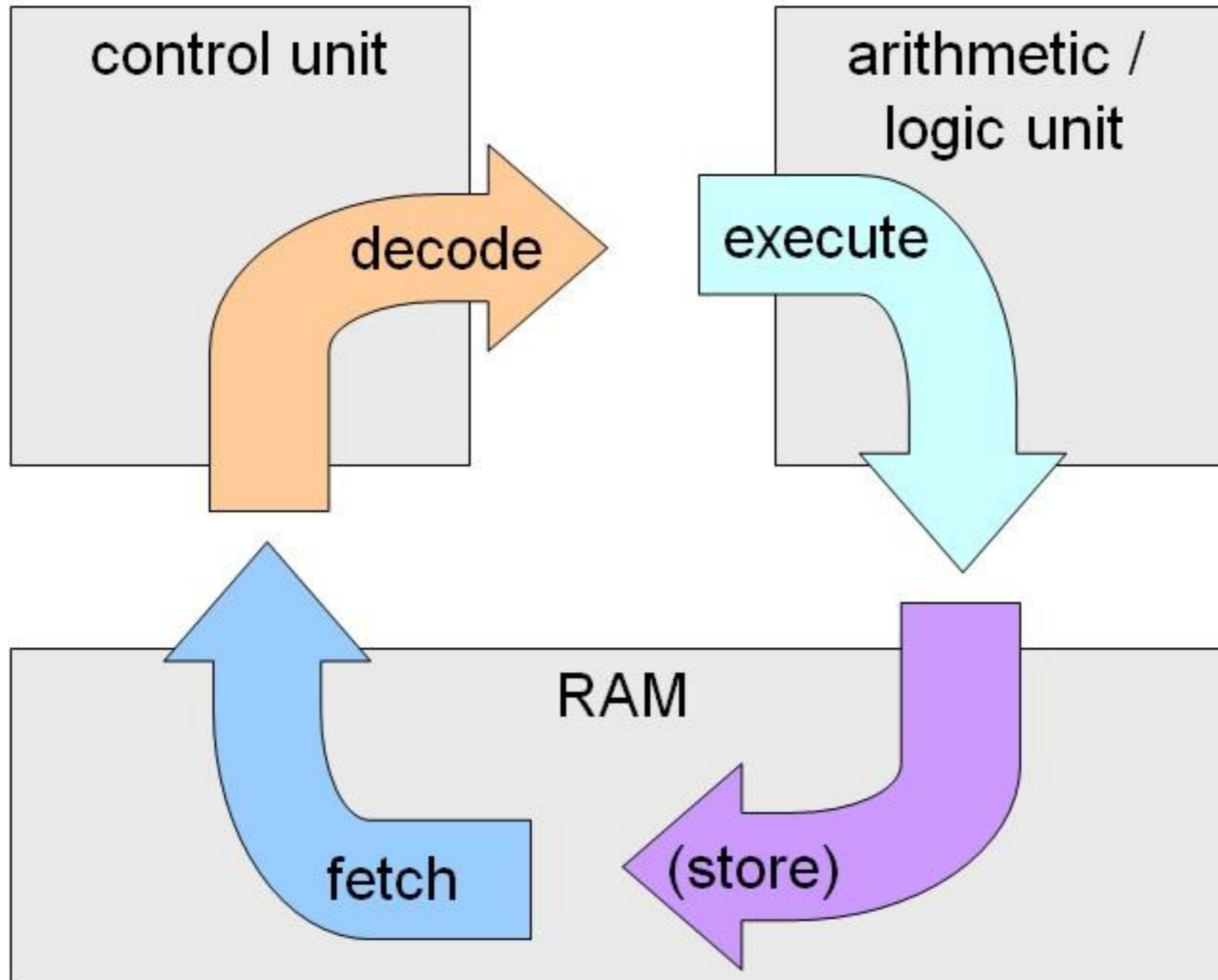


Figure 9.2. The principal subsystems of a computer.

Fetch-decode-execute



ALU

- Utfører regneoperasjonene
- En krets i ALU kan legge sammen to tall
- Det finnes også kretser for å multiplisere, sammenligne, osv.
- Gjør vanligvis jobben i Instruksjonsutførings-steget (EX) i syklusen, men instruksjoner som bare flytter på data bruker **ikke** ALU
- Data/Operand Hente-steget (DF) i syklusen henter de verdiene som ALUen trenger å jobbe med (operandene til operatoren)
- Når ALUen fullfører operasjonen sørger Returnerings/Lagrings-steget for å flytte svaret fra ALUen til den minnelokasjonen som er spesifisert i instruksjonen

PC (Program Counter)

- Hvordan vet maskinen hvilket steg som er det neste som skal utføres?
- Adressen til den neste instruksjonen lagres i kontroll-delen til maskinen. Den kalles programteller (PC)
- Fordi instruksjoner bruker 4 byte minne, må neste instruksjon være på lokasjon $PC + 4$, 4 bytes lenger avgårde i sekvensen/programmet (generelt sett)
- Maskinen plusser på fire i PC, så når F/E-syklusen kommer tilbake til InstruksjonsHente-steget neste gang, så «peker» PC på den neste instruksjonen

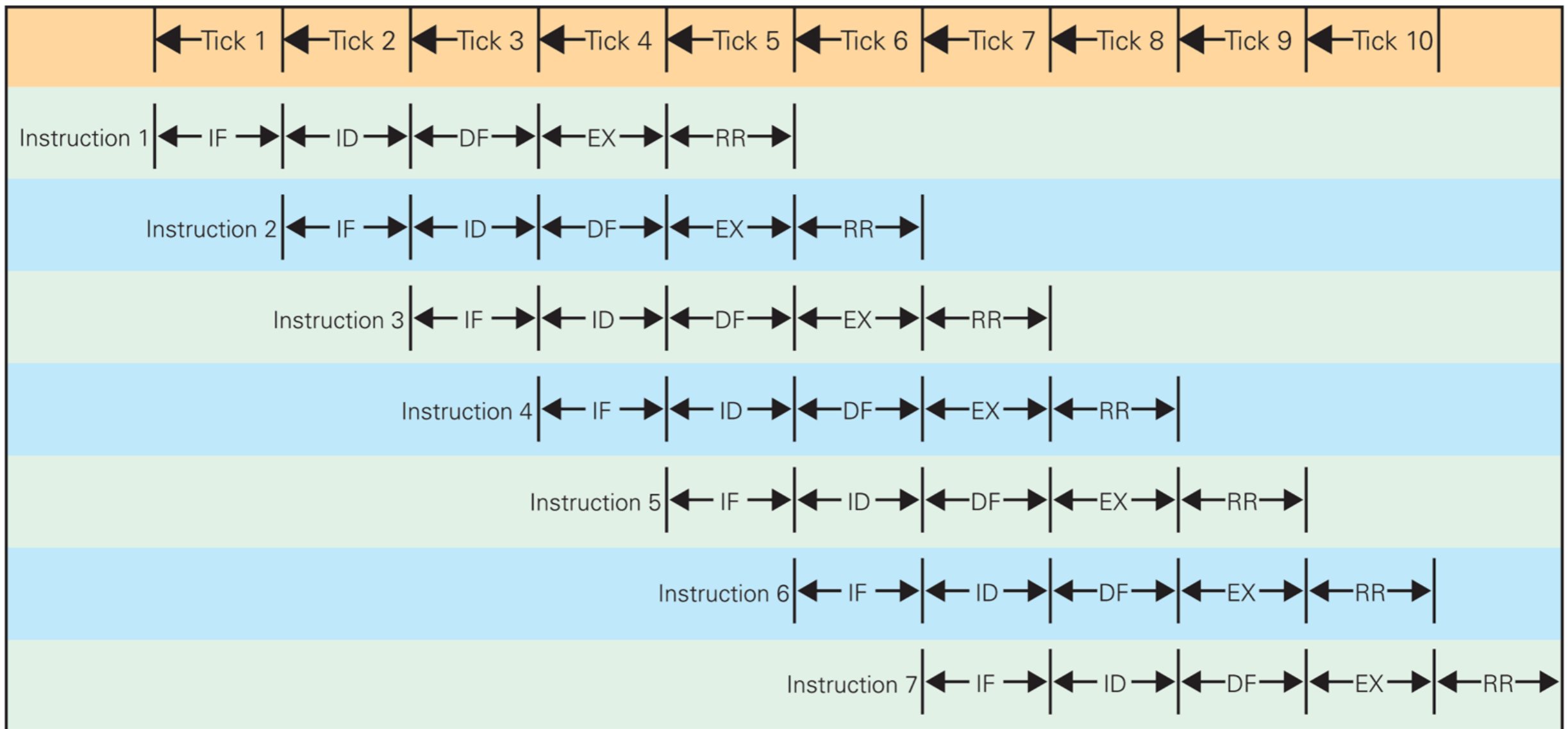
Minne

- Minnet lagrer programmet som kjører og dataene som programmet behandler
- Egenskapene til minnet
 - Atskilte lagerplasser/lokasjoner. Hver plass består av 1 byte
 - Adresser. Hver minnelokasjon (byte) har en adresse (hele tall som starter på null).
 - Verdier. Minnelokasjoner tar opp eller lagrer verdier
 - Begrenset kapasitet. En gitt størrelse—programmerere må huske at dataene kanskje ikke «får plass» i minnelokasjonen
- 1-bytes minnelokasjoner kan lagre ett ASCII-tegn, eller ett nummer <256 (0-255)
- Programmerere bruker ofte en sammenhengende rekke av minnelokasjoner, uten å bry seg om at hver enkeltlokasjon har en egen adresse
 - Blokker på fire bytes blir så ofte brukt som en enhet at de kalles for (minne-) ord («Words»)
- Stort minne fører til mindre I/O

Hvordan forbedre klokke-hastigheten?

- Moderne maskiner prøver å starte en instruksjon på hvert klokke-tikk
- Overlater avslutning av instruksjonen til andre kretser (dette kalles «pipelining», som er en slags parallell-utføring. Forskjellig fra flerkjerne parallell-prosessering)
 - Fem instruksjoner kan behandles samtidig
- Kan en datamaskin med 1 GHz klokke virkelig utføre en milliard instruksjoner per sekund?
 - Ikke helt presist mål. Datamaskinen klarer ikke alltid å starte en ny instruksjon på hvert tikk, men klarer noen ganger å starte mer enn en instruksjon samtidig





Figur 9.12 Skjematisk diagram for en parallell-prosessert «pipeline» Hente/Eksekvare «F/E»-syklus. På hvert tikk starter IF en ny instruksjon, og sender den deretter videre til ID (Instruksjons-Dekoding) enheten; ID-enheten arbeider med instruksjonen den fikk, og når den er ferdig sendes instruksjonen videre til DF (Datahenting «Fetch»)-kretsen, og så videre. Mens «røret» er fullt utføres fem instruksjoner samtidig, og en instruksjon blir ferdig på hvert klokke-tikk, som gjør at det ser ut som om maskinen kjører en instruksjon per tikk.

computer appear to be running at one instruction per tick.

Digital representasjon

- Læringsmål
 - Digital representasjon av informasjon
 - Binære og heksadesimale tallsystemer
 - Koding av tekst, heltall og flyttall

Representasjon av informasjon

- Bits: 0/1 (Bit = Binary digiTs)
- Bytes: 8 bits, $2^8 = 256$ symboler (mønster)
 - 0000 0000, 0000 0001, 0000 0010, ... , 1111 1110, 1111 1111
- Digitalisering: Representere informasjon ved symboler
 - Diskret vs. analog: distinkt, skillbart vs. kontinuerlig
- P and A-koding (Present and Absent)
 - Presence / absence av fysisk fenomen
 - Strøm, spenning, magnetisme, lys/ikke-lys, ...
- Datamaskinens minne
 - Sekvens av bits gruppert i bytes
 - Hver byte har en unik adresse

Konvertering mellom binære tall og desimaltall

- Hver posisjon har en vekt: $2^n, 2^{n-1}, \dots, 2^1, 2^0$
- Konvertering binært til desimalt er enkelt og rett fram
 - $101 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 4 + 0 + 1 = 5$
 - $11001010 = 2^7 + 2^6 + 2^3 + 2^1 = 128 + 64 + 8 + 2 = 202$
- Konvertering desimalt til binært er litt mer komplisert

Tall som konverteres	202	202	74	10	10	10	2	2	0
Posisjonsverdi	256	128	64	32	16	8	4	2	1
Subtraherer		128	64			8		2	
Binærtall	0	1	1	0	0	1	0	1	0

Addisjon og subtraksjon

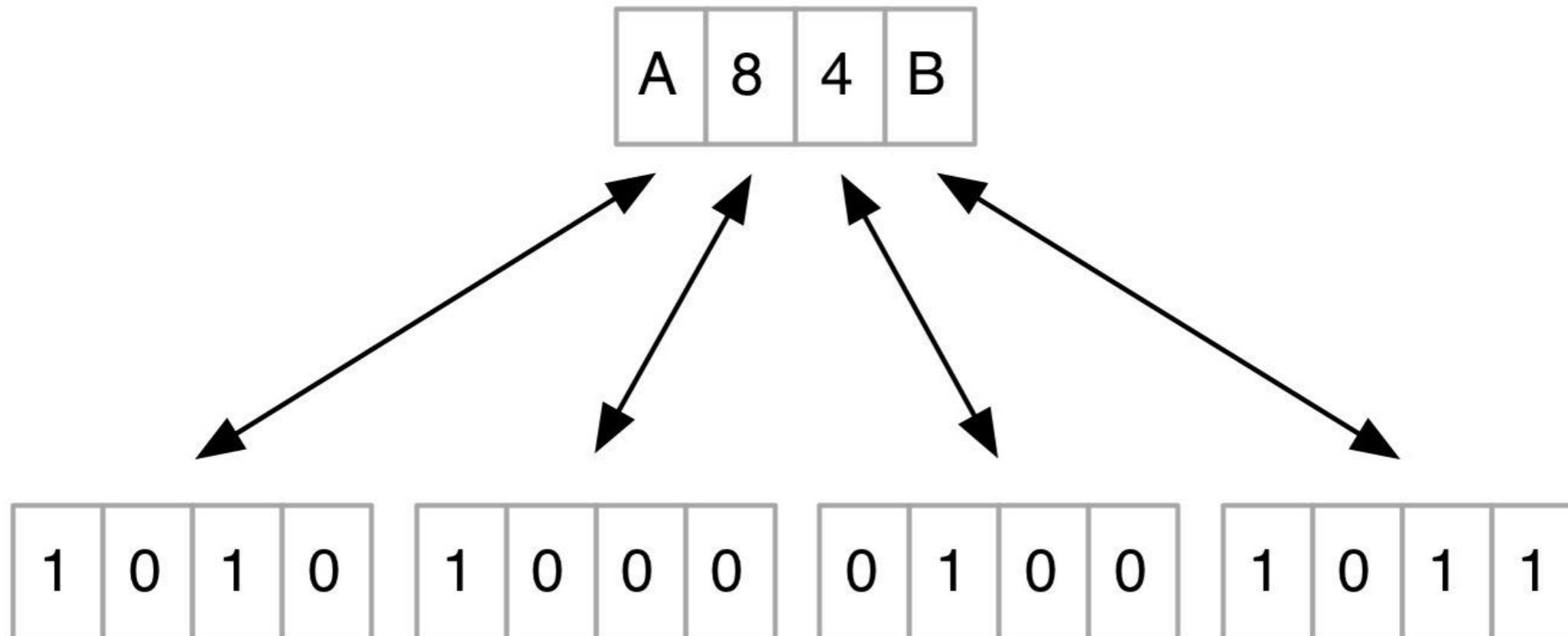
- Samme prinsipper som for desimaltall

		1	1				
		1	1	0	0	(12)	
+		1	1	1	0	(14)	
		1	1	0	1	0	(26)

		2	2				
		1	1	0	1	0	(26)
		1	1	1	1	0	(14)
-		0	1	1	0	0	(12)

Hex til/fra binært

- $10 \cdot 16^3 + 8 \cdot 16^2 + 4 \cdot 16 + 11 = 43083$
- $1 \cdot 2^{15} + 1 \cdot 2^{13} + 1 \cdot 2^{11} + 1 \cdot 2^6 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 = 43083$
- Bruker ofte hex der vi skal gi inn binære tall
 - #FFFFFF = 1111 1111 1111 1111 1111 1111
 - Spesifiserer helt hvitt i RGB-fargekoding på WWW



Representasjon av heltall

- Fast størrelse i antall byte
- int8 (1 byte)
 - $2^8 = 256$ symboler
 - Bare positive tall (unsigned): 0 - 255
 - Positive og negative (signed): Fra -128 til 127
- int32 (4 byte)
 - $2^{32} = 4\,294\,967\,296$ symboler
 - Bare positive tall: 0 – $(2^{32}-1)$
 - Positive og negative tall: Fra -2 147 483 648 til 2 147 483 647
- Eksakte verdier
- Kan få overflyt
 - Int8: 255 + 1 kan ikke representeres

Positive og negative tall

- Fortegnsbit (0 = positivt, 1 = negativt)
- Case: 3 bits heltall (8 symboler)
- Enkleste representasjon vist til høyre
- To nuller!
- Vanlig å bruke 2-komplement for negative tall

Fortegnsbit
↓

0	1	1	3
0	1	0	2
0	0	1	1
0	0	0	0
1	0	0	0
1	0	1	-1
1	1	0	-2
1	1	1	-3

Vanlig å bruke 2-komplement for negative tall

- Ta positiv koding, bytt alle 0 og 1, legg til 1 (ignorerer overflyt)
- Eks:
 - 001 -> 110 -> 111 (-1)
 - 010 -> 101 -> 110 (-2)
 - 011 -> 100 -> 101 (-3)
- Bonus: Subtraksjon == Addisjon

	0	1	1	3
+	1	1	0	-2
=	0	0	1	1

	0	1	0	2
+	1	0	1	-3
=	1	1	1	-1

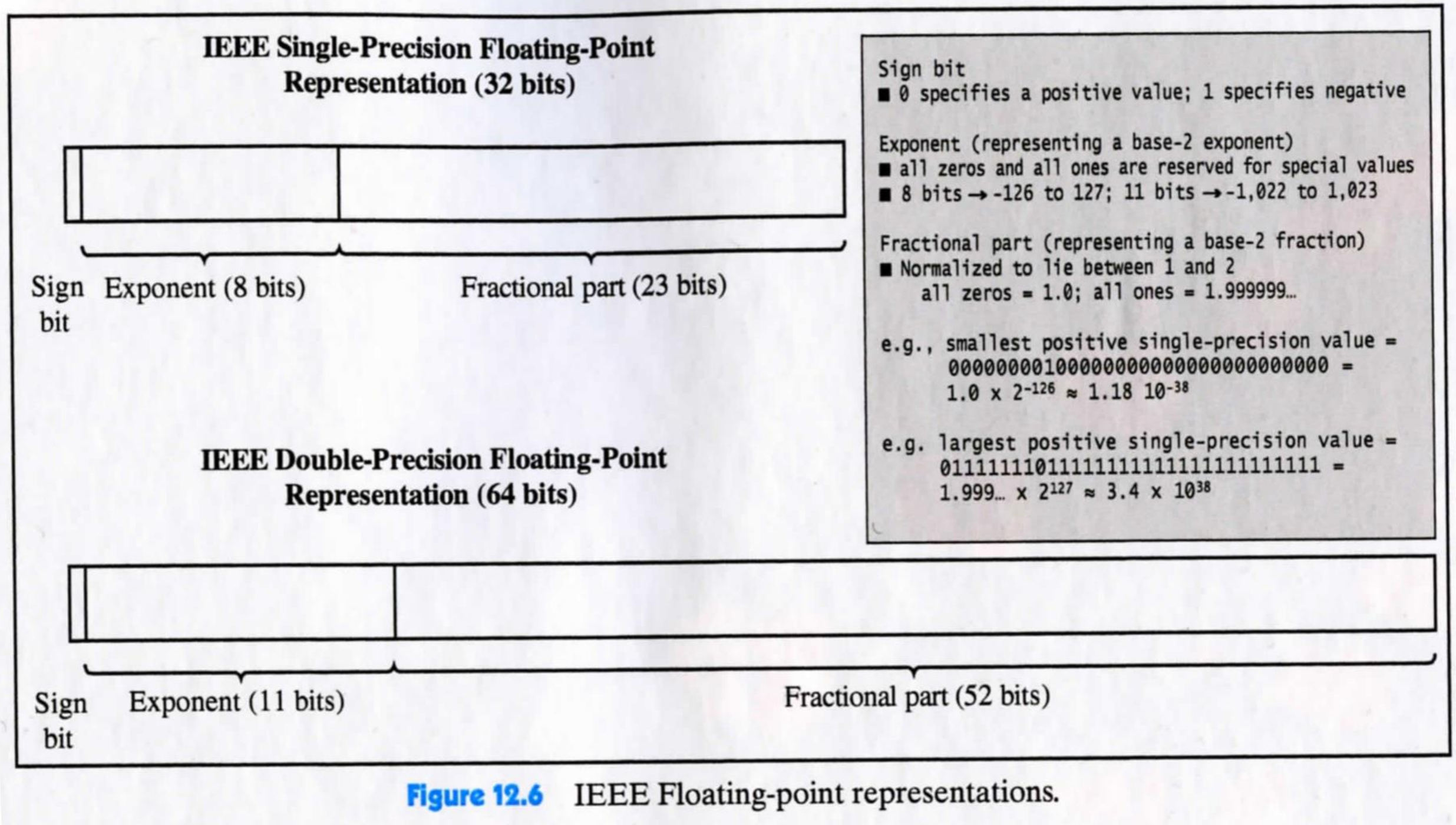
(ignorerer overflyt ut av 3 bit)

011	3
010	2
001	1
000	0
111	-1
110	-2
101	-3
100	-4

Representasjon av reelle tall

- Tallene representeres i scientific notation
 - $1234,56 = 1,23456 \times 10^3$
 - $0,0011 = 1,1 \times 10^{-3}$
 - Normalisert med ett siffer foran komma
- Kan representere tall med:
 - Mantisse (fractional part) inkludert fortegnstegn
 - Eksponent (exponent)
- Komma alltid etter første siffer i mantissen (floating point)
- Eksempler:
 - 1234,56 har 0 (tegn), 123456 (mantisse), 3 (eksponent)

IEEE flyttallstandard



Representerer lyd/grafikk

Lyd

- Et objekt lager lyd ved å vibrere et medium
- Kraften eller intensiteten av bølgene avgjør volumet
- Frekvensene (antall bølger per sek) avgjør tonehøyden (pitch)
- "Sample" eller ta målinger på bestemte intervall
- Antall samples per sekund kalles samplingsfrekvens (rate)
- Jo raskere frekvens, jo mer nøyaktig opptak av lyden

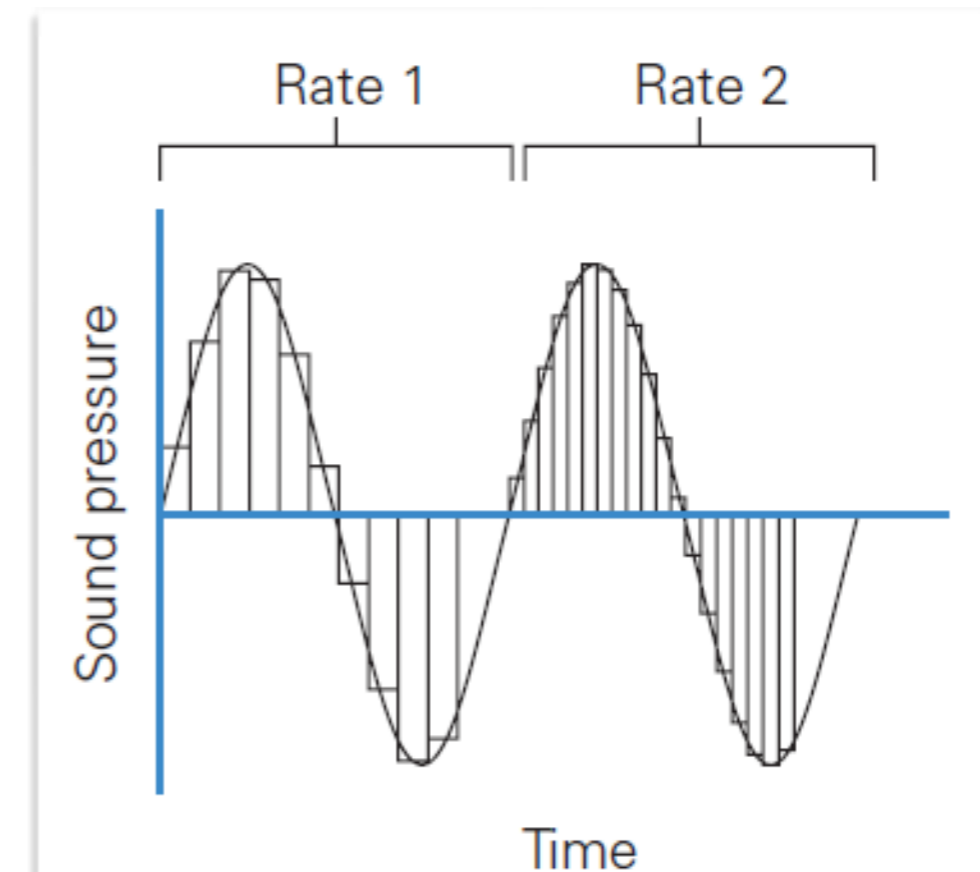
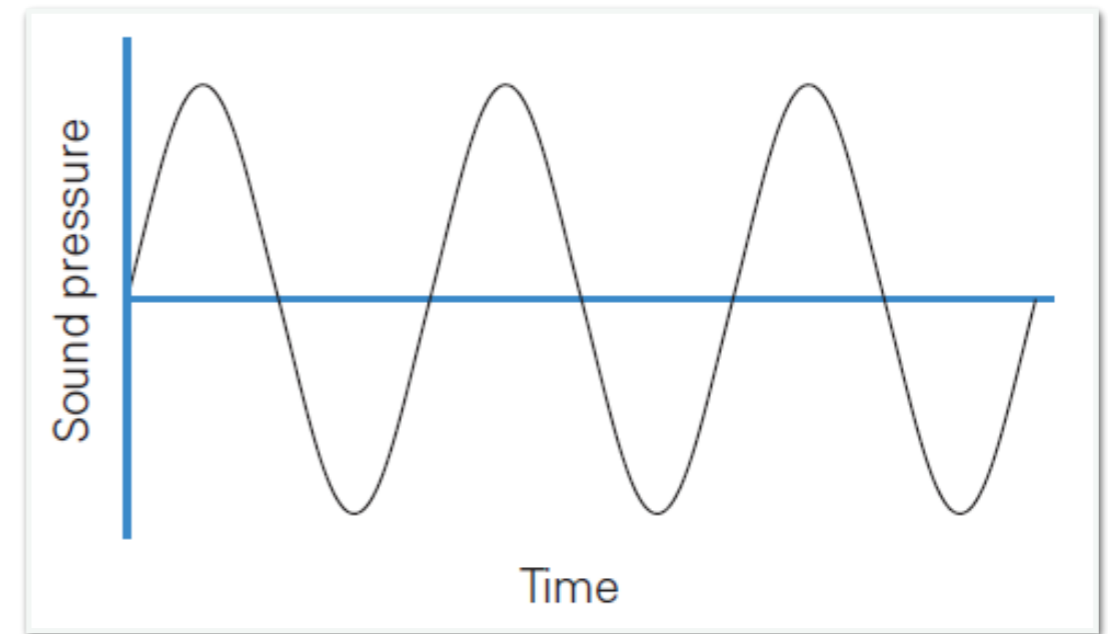
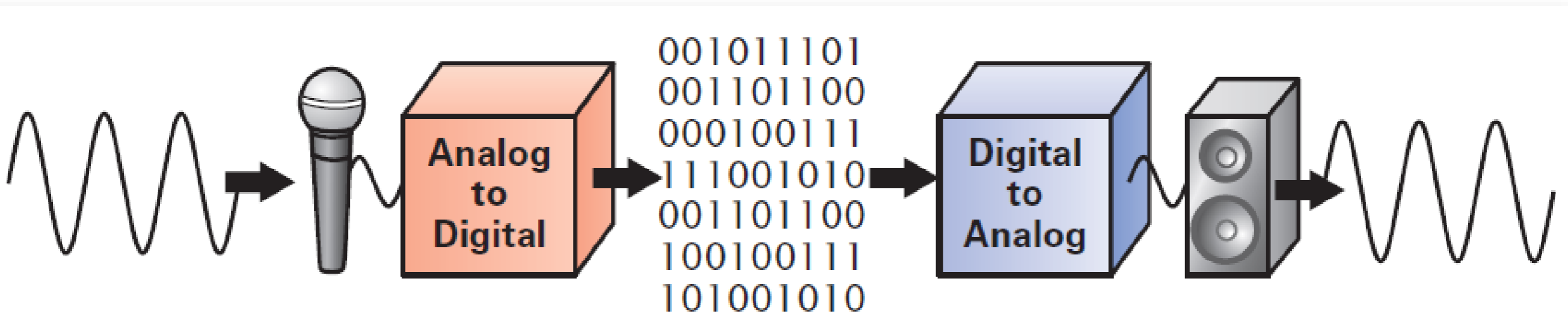


Figure 8.13 Two sampling rates; the rate on the right is twice as fast as that on the left.



- Vi kan kun få omtrentlige målinger av lyd
- Hvis et ekstra bit vil bli brukt, vil lydsamplet bli dobbelt så nøyaktig
- Flere bits oppløsning gir mer nøyaktig digitalisering
- Digital CD lyd representeres ved 16 bit, dvs. 65 536 forskjellige lydnivåer.

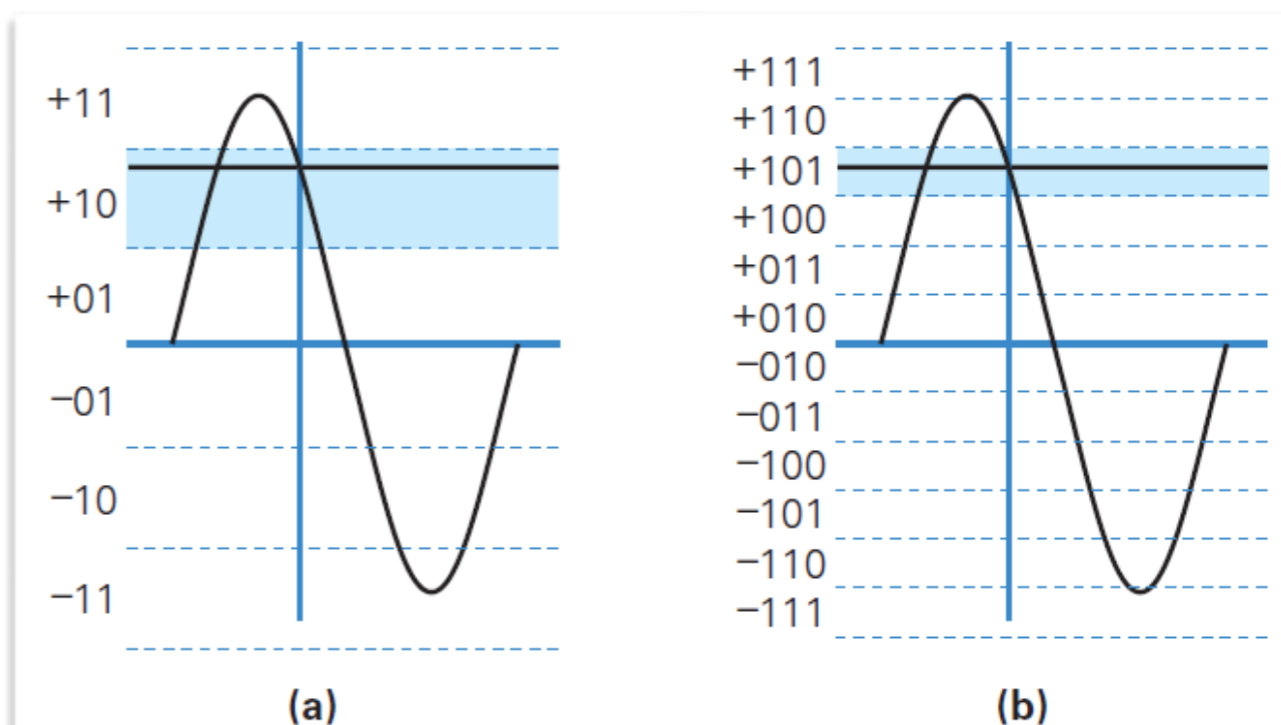


Figure 8.15 (a) Three-bit precision for samples requires that the indicated reading is approximated as +10. (b) Adding another bit makes the sample twice as accurate.

Digital lyd

- En viktig fordel med digital informasjon er muligheten til å kjøre databehandling på representasjonen.
 - En viktig databehandling er å komprimere digital lyd eller redusere antall bit som trengs.
- Hva med lyder som det menneskelige øre ikke kan høre fordi de er for høye eller lave?
- MP3 er representasjon av digital lyd som er databehandlet for å ta mindre plass (fjerner unødvendig informasjon).
 - Gir komprimering på mer enn 10:1 (mindre filer)
 - Reduserer datainnholdet i filene, og dermed lydkvaliteten.
- Digital data kan reproduseres nøyaktig.
 - Kan dette være et problem?

Komprimering av lyd

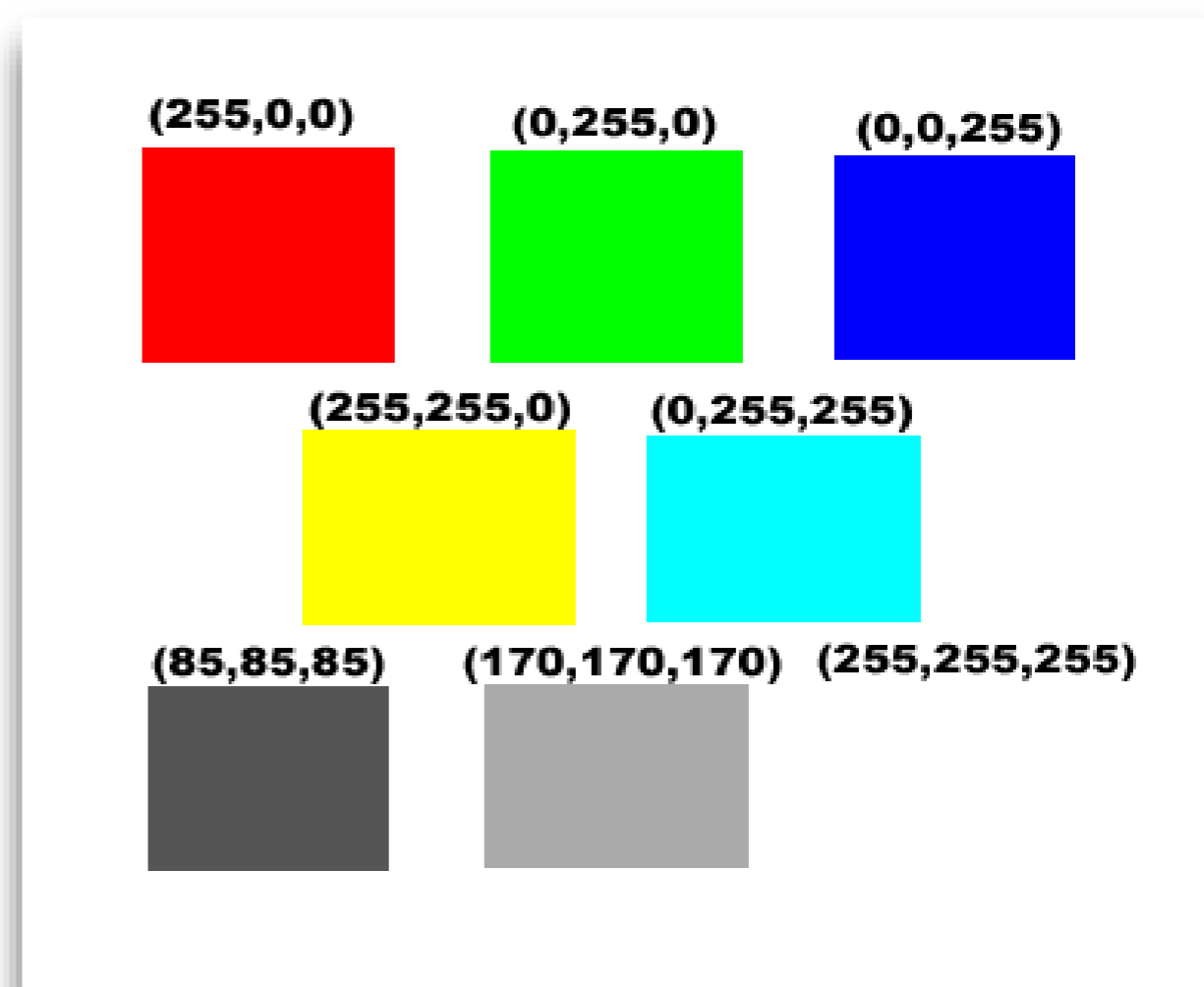
- Lossy - ca 90% reduksjon i filstørrelse
 - MP3
 - MPEG-1 og 2 Audio layer III - tatt i bruk i 1992
 - AAC - Advanced Audio Coding
 - MPEG-4 lydstandard - introdusert i 1997
- Lossless - ca 40% reduksjon i filstørrelse
 - FLAC - Free Lossless Audio Codec - introdusert i 2001
 - ALAC - Apple Lossless Audio Codec - introdusert i 2004
 - Begge er open source og gratis

Bilde

- Et bilde er en lang sekvens av RGB piksler
- Bildet er todimensjonalt, men tenk pikslene strekt ut rad etter rad i minne.
- Eksempel:
 - 8 x 10 tommer bilde skannet til 300 piksler per tomme (dpi)
 - Det er 80 tommer², hver trenger 300 x 300 = 90 000 piksler (7,2 megapiksler)
 - Representerer farger med 3 bytes per piksel, tar 21.6MB (3 * 7.2) i minnet til lagring av 8x10 fargebilde på 300 dpi (dots per pixels)
 - Sende dette bildet over en 56Kb/s linje vil ta minst $21\,600\,000 \times \frac{8}{56\,000} = 3085$ sekunder (mer enn 51 minutter)

Bildestørrelse

- Størrelse (dimention)
 - Bredder (cm/tomme) x høyde (cm/tomme)
- Oppløsning (resolution):
 - antall pixler pr cm/tomme (dpi = dot pr inch)
- Fargedybde
 - Antall fargealternativer for hver pixel:
 - 1 bit : 2 alternativ
 - 8 bit : $2^8 = 256$
 - 24 bit : $256 * 256 * 256 = 2^{24} = 16,7$ millioner
- På skjerm er oppløsning irrelevant
 - Bredder (antall pixler)
 - Høyde (antall pixler)
 - Fargedybde (antall byte pr pixel)
- Filstørrelse = Bredder * Høyde * Fargedybde
 - $600 * 400 * 3$ byte = $240.000 * 3 = 720.000$ byte
- 600x400, 256 farger (indeksert)
 - $600 * 400 * 1$ byte = 240.000 byte

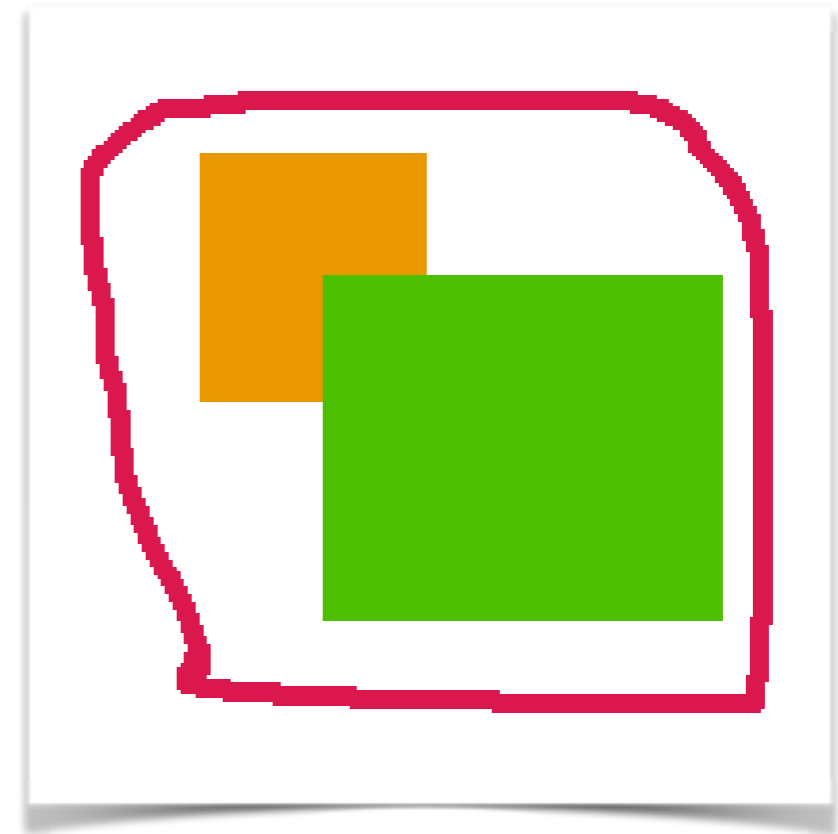


Filformater

- Regler for koding av bilder
- Fargedybde
 - 24 bit / indekserte farger
- Komprimering
 - Dataene kodes slik at de tar mindre plass
 - Komprimering: Bilde -> Kodet fil
 - Dekomprimering: Kodet fil -> Bilde
 - Tapsløs / tapskoding
 - Tapsløs: Halvering – tredeling
 - Tapskoding: Større effekt, avhengig av tap
- TIFF, PSD, BMP, PNG, GIF, JPEG/JPG

Formater (2)

- GIF
 - 8 bit fargetabell (= 256 farger)
 - Alfa-kanal for transparens
 - Tapsløs komprimering
 - Bra for "data grafikk", mindre bra for fotografier
- JPEG
 - 24 bits
 - Komprimering med tap, styrbar
 - Bra for fotografier (glatter skarpe kanter + artifakter)
 - Ikke bra for "datagrafikk"
 - Org 560 KB
 - 14 KB og 162 KB til høyre



Bildekomprimering

- Komprimering betyr å endre representasjon slik at færre bit trengs til å lagre eller overføre informasjon.
 - Eks: Fakser en sekvens av 0 og 1ere som koder en side av hvitt (0) og sort (1)
 - Bruker run-length-koding for å spesifisere hvor lang første sekvens av 0ere er, så hvor lang neste sekvens av 1ere er, osv...
- Run-length-koding er tapsløs (lossless) komprimering
 - Den originale representasjon av 0ere og 1ere kan bli rekonstruert perfekt fra den komprimerte versjonen.
- Motsatte er lossy- (taps)-komprimering
 - Originale representasjon kan ikke rekonstrueres eksakt fra den komprimerte versjonen.

Komprimering

- Lossy og lossless
 - Lossy fjerner informasjon man antar har liten effekt på opplevelsen
 - Lossless tar vare på all informasjon og fjerner ingenting
- MP-3 er sannsynligvis den mest kjente type komprimering
 - MP3 er lossy (mono i bass, fjerner lave lyder like etter kraftige etc...)
- JPG (or JPEG) er lossy komprimering av bilder
 - Utnytter karakteristikker av menneskelige oppfatningsevne for å lage forenklinger iht. lys og farge som ikke er så synlig.
 - Mennesker er temmelig sensitive til små endringer i lysstyrke (luminance)
 - Lysstyrkenivåer i et bilde må bevares mellom ukomprimert og komprimerte versjoner
 - Mennesker er ikke sensitive til små forskjeller i farger (chrominance)