TK1100

FORELESNING 0x00 INTRO

Noen kjappe tips til oppgavene:

- Skriv med dine egne ord, oppsøk gjerne andre kilder enn forelesningen som kan hjelpe forståelsen din.
- Hold det til maks. 5 – 6 setninger på spørsmålene der det står «Forklar kort». Andre spørsmål kan det være fint å utdype grundigere.
- Enkelte av oppgavene kan gå litt utenom det som er presentert i forelesningene, bruk gjerne veilederene til å peke deg i riktig retning!

1) Datamaskinens bruksområder

a) Forklar kort hva et PC-system er.

Et PC-system er et system der ytelsen er å transportere data og instruksjoner mellom komponentene som finnes i PC'n og få de til at bli prosessert raskt. Her er det viktig at komponentene er i balanse med hverandre så at de ikke holder hverandre tilbake, de burde med andre ord være på samme nivå.

b) Gjør din egen definisjon av hva «informasjon» er i datasammenheng.

Her finnes det flere svar, det viktigste er å skjønne at «informasjon» kan bety flere ulike ting i ulike sammenhenger.

c) Nevn noen bruksområder for datamaskiner. Bruk gjerne eksempler fra din egen hverdag. Hva er de viktigste hovedmomentene ved datamaskinene du har i din egen hverdag?

Datamaskiner finnes i dag overalt, her vil vi at du reflekterer over hvilke datamaskiner som finnes i din hverdag og hva bruksområdene deres er.

Eksempel på hovedmomenter med datamaskiner:

- Hastighet
- Pålitelighet
- Lagringsevne
- Pris
- Størrelse

Eksempel på bruksområder for datamaskiner:

- Embedded systems (Trafikklys, kaffetrakter etc.)
- Mobiltelefon
- Datalagring
- System til banker, kommune etc.
- Forskning

d) Forklar med egne ord forskjellene mellom det binære tallsystemet og titall-systemet. Bruk gjerne tegninger for å vise hva du mener.

Her bør svaret ditt ha vært innom:

- Titalls-systemet er det vi bruker i hverdagen i Norge, og det vi lærer mest om på skolen.
- Det binære tallsystemet bruker det samme konseptet med «plasser» som titallssystemet, men representerer andre verdier ved hver plass:

  Når vi skriver tallet 10 i titallsystemet, betyr dette egentlig at vi har:
  1 x tierplassen og 0 x enerplassen som gir oss tallet ti.

  Når vi skriver tallet 10 i det binære tallsystemet, betyr dette egentlig at vi har:
  1 x toerplassen og 0 x enerplassen som gir oss tallet to.

- Base betyr hvilket tallsystem vi oppgir et tall på.
  Base 10 refererer til titallsystemet og *base 2* det binære tallsystemet, men vi er ikke begrenset til å oppgi tall på bare disse to basene.

- Totalls-system brukes for å «snakke» med datamaskiner.
- Et binært siffer kalles en bit
- En gruppe med 8 bit kalles for en **byte**

e) Skriv ned tallene 1 – 15 på det binære tallsystemet.

0	01	10	11
100	101	110	111
1000	1001	1010	1011
1100	1101	1110	1111

f) Hvorfor bruker datamaskiner det binære tallsystemet og ikke titalls-systemet?

Forenklet forklart fungerer en datamaskin ved at strøm sendes rundt omkring som signaler i maskinen med beskjed om hva som skal utføres. De ulike komponentene i maskinen tolker hva de skal gjøre basert på signalet de får inn. Det kan være forvirrende å skjønne hvordan dette knyttes opp mot det binære tallsystemet, men det kan være lurt å se på hva som ville skjedd hvis datamaskiner hadde brukt titallsystemet i stedet.

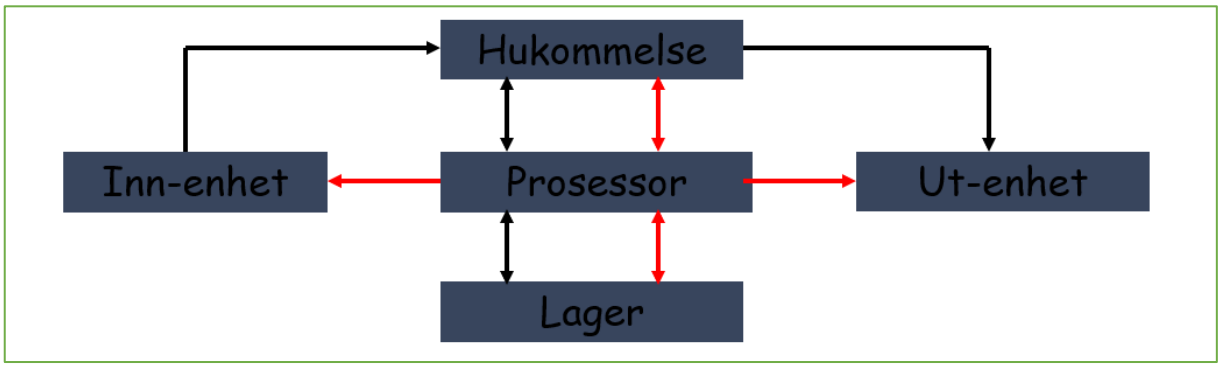
La oss se for oss et scenario der datamaskiner bruker titallssystemet og med det har ***10*** mulige ting som kan skje basert på et innkommende signal. Vi måler signalet i maskinen til å være 2,5 på en 0-9 skala. Skal vi tolke den målingen som 2 eller 3? Her blir det vanskelig for maskinen å *tolke* signalet, og skjønne hva som skal gjøres videre.

Gitt det samme scenarioet der datamaskiner i steden bruker *det binære tallsystemet*. Hvis vi bare har ***to*** mulige ting som kan skje kan vi jo si at alle målinger mellom 0 – 4 er ***0*** og alle målinger mellom 5 – 9 er ***1***. Med den samme målingen på 2,5 kan vi være helt sikre på at signalet egentlig skal tolkes som en 0.

Vi bruker altså det binære tallsystemet med datamaskiner blant annet fordi det eliminerer usikkerheter. Datamaskinen må vite nøyaktig hva den skal gjøre når den får et innkommende signal.

2) Datamaskinens bestanddeler

a) Tegn opp Von-Neumann modellen og delene den består av.



b) Hvor mange lag har en funksjonsorientert modell mot en maskinvareorientert modell?

En funksjonsorientert modell har totalt 6 lag og disse er:

- Lag 5: Brukerprogramnivå
- Lag 4: Kompilatornivå/interpreter
- Lag 3: Operativsystemnivå
- Lag 2: Instruksjonsnivå
- Lag 1: Mikroinstruksjonsnivå
- Lag 0: Digital kretsnivå

En maskinvareorientert modell har totalt 4 lag og disse er:

- Portnivå
- Registernivå
- Prosessornivå

c) Gi et par eksempel på hardware.

Hardware/maskinvare i datasammenheng er fysiske komponenter som håndterer prosessering, datalagring og inn-/ut-operasjoner.

Eksempel:

- Harddisk
- RAM
- CPU/prosessor
- GPU
- CD-ROM
- Diskett
- Mus
- Skriver

d) Gi et par eksempel på software.

Software/programvare er en «abstrakte» eller ikke-fysiske applikasjoner som kjøres på en datamaskin, disse er oppbygd med et programmeringsspråk som datamaskinen skjønner.

Eksempel:

- Skype, Zoom, Slack
- Word, Excel, PowerPoint
- Chrome, Safari, Edge

e) Hvorfor kalles «hjernen» i en datamaskin CPU eller «Central Processing Unit», og hvordan gir man den instruksjoner?

Det er CPU-en som utfører alle instruksjoner vi gir til datamaskinen. Når vi vil at datamaskinen skal gjøre noe må det alltid «gjennom» CPU-en. Den kan derfor tenkes på som en hovedmotor. Programmeringspråk gir oss mulighet til å sende instruksjoner til en prosessoren.

f) Skriv om tre valgfrie typer av minne. Bruk gjerne andre kilder enn forelesningene for å finne informasjon.

Eksempel på minne som kan skrives om:

- RAM
- SRAM
- DRAM
- ROM
- FLASH
- Virtuelt minne
- CACHE

g) Hva er forskjell på minne og masselager?

Et masselager er der data er lagret permanent, masselager brukes til alt fra f.eks. dine personlige bilder, til filer datamaskinen trenger for å fungere på lang sikt. Minne er der data ligger tilfeldig for at den skal brukes av prosessoren i datamaskinen. For eksempel så er et program lagret på din harddisk(masselager), men når du kjører programmet så er deler av programmet i RAM(minne) hvor det tar mye kortere tid for prosessoren å hente informasjon fra. For å overføre data til prosessoren fra masselager så brukes det en kontroller, en kontroller kan være IDE/EIDE, SATA eller SCSI.

Eksempel på masselager:

- CD-ROM
- USB
- Hullkort
- Harddisk
- Diskett
- Magnetbånd

h) Prøv å finne info om komponentene på din egen datamaskin. Eks. Hvor mye plass har du på din harddisk eller hvilken type CPU har du?

Dette er en praktisk oppgave som har ulike svar basert på hvilken datamaskin du har, for å finne ut av hva svaret er så er det enkleste å google hvordan du skal gå frem.

i) Forklar hva et abstraksjonsnivå er, og hensikten med det.
Et abstraksjonsnivå er et begrep på at man gjemmer bort kompleksitet for å forenkle hvordan man forstår og forholder seg til maskin- eller programvare. Jo *høyere* abstraksjonsnivå, jo *færre* detaljer og kompleksitet forholder man seg til. Jo *lavere* abstraksjonsnivå, jo *mer* detaljer og kompleksitet forholder man seg til.

Om du for eksempel skal utvikle en nettside er det sjeldent du trenger å ha et forhold til hvordan en prosessor fungerer ned på et fysisk nivå (f.eks. hvordan strømmen beveger seg i datamaskinen). Men det kan likevel være veldig fint å ha en konseptuell forståelse av hva en prosessors rolle er i en datamaskin og hvordan den oppnår noe av det. Da kan man si at denne personen forstår en prosessor på et *høyt* abstraksjonsnivå.

j) Forklar kort hva et operativsystem er.
Operativsystemet er programvare som gir et brukergrensesnitt, sikkerhet og pålitelighet. OS-et gjør det mulig å kjøre de fleste applikasjoner og administrerer ressursene som finnes i datamaskinen gjennom å tildele disse til andre programmer. En viktig funksjon operativsystemet har er at det ser til at flere programmer ikke kan sende data til samme ressurs samtidig, men legger ting i venteliste. Det kan også være verdt å nevne at alle datamaskiner er ikke *må* ha operativsystem.

3) Passord

a) Forklar kort hva bitstyrke er og hvorfor det er viktig i passordsammenheng

Bitstyrke sier noe om hvor sterkt et passord er. For å måle dette ser man på hvor mange forsøk en tilfeldig angriper maksimalt trenger for å gjette passordet. Formelen for å regne ut bitstyrken er:

lg_2(Antall *mulige* karakter) * antall karakterer

Gitt et passord på **4** karakterer der det kun er lov med norske bokstaver, altså **29** *mulige* karakterer, vil utregningen se slik ut:

$lg_2(29) * 4 =$

$4{,}86 * 4 =$

$\underline{\underline{19{,}44}}$

Anbefalt bitstyrke er på ca. 80. Dette kan da være en hvilken som helst kombinasjon av karakterer og mulige karakterer.

b) Hva er brute-force?

Brute-force er en metode for å finne/gjette riktig passord gjennom å prøve alle mulige kombinasjoner som finnes.

c) Er et passord som inneholder åtte bokstaver nødvendigvis sikkert?
Nei det er ikke nødvendigvis sikkert.

Når vi regner ut bitstyrken til et passord ser vi på antall mulige karakterer i tillegg til lengden på passordet. I denne oppgaven står det bare «bokstaver», men *hvilke* bokstaver snakker vi om?

Er det bare lov med bokstaver fra det norske alfabetet i passordet vil det være 29 mulige karakterer. Det vil være sterkere enn f.eks. det engelske alfabetet som bare har 26 bokstaver og dermed 26 mulige karakterer. For å avgjøre om et passord er sikkert er det derfor viktig at vi vet omstendighetene rundt passordet.

En annen faktor vi kan trekke inn er om de 8 bokstavene er del av et vanlig passord. Brute forcing er sjeldent tilfeldig, og man prøver gjerne de mest åpenbare mulighetene først. Gitt at de 8 bokstavene i passordet er «Password» kan man derfor regne med at passordet ikke er særlig trygt.

d) Hva er tiltak man kan gjøre for at et passord blir sterkere?
- Bruke mer enn bokstaver og tall, eksempel spesialtegn som: @£$€{[]}
- Ha blanding mellom store og små bokstaver
- Ikke bruke samme passord på flere ulike tjenester
- Lagre passordet sikkert

e) Sorter passordene under etter styrke, start med den som er minst sikker. Bruk f.eks. Microsoft sin passordsjekker til å teste svarene dine
- iliketoast
- qwerty
- password123
- jegelskerågåturiskogenpålørdager

Med utgangspunkt i at alle passordene kan ha tall fra 0-9 og norske karakterer vil styrken rangert fra svak til høy være følgende:

qwerty

iliketoast

password123

jegelskerågåturiskogenpålørdager

4) Historie

a) Skriv om de tre historiske problemstillingene den moderne datamaskinen løser.
Den moderne datamaskinen løser hovedsaklig tre ulike problemstillinger:

- Beregningsproblemet. Altså hvordan kan vi utføre kompliserte beregninger på en rask og sikker måte. Her kan man se for seg att man tidligere satt og gjorde komplekse beregninger med kalkulator og papir og penn. Tenk på alle feilkildene som kan oppstå når et menneske skal utføre det mot en dagens datamaskin.
- Massedataproblemet. Altså hvordan man kan lagre store mengder data på en pålitelig måte og behandle den. Tidligere brukte man kanskje papirarkiv som både er upålitelig og vanskelig å behandle effektivt.
- Reguleringsproblemet. Altså hvordan kan man styre og automatisere industrielle prosesser? Før hadde man mange fler industrielle arbeidere som styrte ulike deler av f.eks. en produksjon. I dag er mye helautomatisert, eller styres fra en eller flere datamaskiner.

b) Hva er Moore's lov og hva beskriver den? Vil den for alltid være sann?

Moore's lov er en observasjon (og teknisk sett ikke en lov), der man har sett at antall transistorer på et areal vil dobles hver 24. måned. Transistorer kan beskrives som det som får en datamaskin til å fungere, og man kan forstå det som at *flere* transistorer vil gi *høyere* ytelse i en datamaskin. Moore's lov beskriver derfor farten på den teknologiske utviklingen vi har sett siden midten av forrige århundre.

Det er ingen enighet om når Moore's lov kommer til «å dø», eller med andre ord slutte å være sann. Men det kommer jevnlig uttalelser fra ulike folk i industrien som konkluderer med at den allerede er død og at vi dermed kommer til å se en stagnert utvikling i antall transistorer/areal i fremtiden.

c) Hva er Intel 4004, og hvilken generasjon hørte den til?
Intel 4004 er en 4. generasjons 4 bit CPU produsert av Intel, lansert i 1971. Den var både den første CPU-en produsert av Intel og den første *mikroprosessoren* som ble kommersiellt tilgjengelig. En mikroprosessor mer spesifikt er en prosessor som kun består av én eller veldig få integrerte kretser.

d) Hva beskriver Wirth's lov?
Wirth's lov er som Moore's lov altså heller ikke teknisk sett en lov, men et slags ordtak eller en «saying» som konstaterer at programvare blir tregere fortere enn maskinvare blir raskere.

5) Binær koding

a) Trening til neste økt, fortsett å skrive ned tall på det binære tallsystemet opp til 128: 1, 2, 4, 8…

TK1100

FORELESNING 0x01 BINHEX

1) Binær koding

a) Hvilke ulike måter er det vist i forelesning at man kan representere karakterene i ASCII tabellen?

Svar: Desimaltall (base 10), Hexadesimal (base 16), Oktaltall (base 8) og Html-enkoding.

b) Hvilke hovedtyper data, eller informasjon, behandler en datamaskin?

- Numerisk
- Karakterbasert(alfanumerisk)
- Visuell
- Audio
- Instruksjoner
-

c) Fyll inn tabellen under.

2^0 =	1
2^1 =	2
2^2 =	4
2^3 =	8
2^4 =	16
2^5 =	32
2^6 =	64
2^7 =	128
2^8 =	256
2^9 =	512
2^{10} =	1024

d) Hvor mange bitmønstre (forskjellige kominasjoner av 0 og 1) kan du lage med 8 bits?

- 8, fordi det er åtte bits
- 16, fordi det er åtte bits som hver kan ha verdien 0 eller 1, som gir 8*2=16 kombinasjoner
- 64, fordi 8*8 = 64
- **256, fordi hver bit kan være 1 eller 0, da er det med 8 bit mulig å lage 2*2*2*2*2*2*2*2 = 28 = 256 kombinasjoner**
- 1024, fordi det er en Ki

e) Hva har det binære tallsystemet som base?

Svar: Base 2

f) Hvorfor bruker vi presisjon? Nevn fire vanlige presisjoner.

Datamaskiner har begrensninger på hvor mange bit som kan lagres i f.eks. minnet. Når vi gjør utregninger i binære tall må vi derfor ha et konsept av hvor mange bit vi må oppgi. Dette kan også føre til at når vi gjør utregninger vil enkelte bit havne utenfor kapasiteten i minnet. Disse såkalte «overflow»-bitene kan man ignorere.

Noen vanlige presisjoner er:
Byte – 8 bit
Word – 16 bit
Dword – 32 bit
Qword – 64 bit

g) Fyll inn tallene som skal erstatte x i utrekningene under. Skriv det ned på papir.

0001 = $0 + 0 + 0 + 2^0$ = $0 + 0 + 0 + 1$ = **1**

0010 = $0 + 0 + 2^1 + 0$ = $0 + 0 + 2 + 0$ = **2**

0011 = $0 + 0 + 2^1 + 2^0$ = $0 + 0 + 2 + 1$ = **3**

0100 = $0 + 2^2 + 0 + 0$ = $0 + 4 + 0 + 0$ = **4**

0101 = $0 + 2^2 + 0 + 2^0$ = $0 + 4 + 0 + 1$ = **5**

0110 = $0 + 2^2 + 2^1 + 0$ = $0 + 4 + 2 + 0$ = **6**

0111 = $0 + 2^2 + 2^1 + 2^0$ = $0 + 4 + 2 + 1$ = **7**

1000 = $2^3 + 0 + 0 + 0$ = $8 + 0 + 0 + 0$ = **8**

1001 = $2^3 + 0 + 0 + 2^0$ = $8 + 0 + 0 + 1$ = **9**

1010 = $2^3 + 0 + 2^1 + 0$ = $8 + 0 + 2 + 0$ = **10**

h) Fyll inn tabellen under, skriv det ned på papir.

$4^0 = 1$
$4^1 = 4$
$4^2 = 16$
$4^3 = 64$
$4^4 = 256$
$4^5 = 1\,024$
$4^6 = 4\,096$
$4^7 = 16\,384$
$4^8 = 65\,536$
$4^9 = 262\,144$
$4^{10} = 1\,048\,576$

2) Konvertering

a) Konverter disse tallene fra binærtall (base 2) til desimaltall (base 10)

- 1111 1111 = **255**
- 0000 0000 = **0**
- 1001 1001 = **153**
- 1100 0011 = **195**
- 1010 1010 = **170**
- 0100 0101 = **69**
- 0010 1101 = **45**
- 1011 0010 = **178**

b) Konverter disse tallene fra desimaltall (base 10) til binærtall (base 2)

- 21 = **0001 0101**
- 9 = **0000 1001**
- 16 = **0001 0000**
- 196 = **1100 0100**
- 232 = **1110 1000**
- 72 = **0100 1000**
- 32 = **0010 0000**
- 256 = **1 0000 0000**

3) Addisjon binært

Adder disse tallene, bruk en byte (8 bit) presisjon. Bruk penn og papir.

a) 0000 0000 + 0010 0010 **= 0010 0010**

b) 1001 1001 + 0110 0110 **= 1111 1111**

c) 0001 1010 + 0000 0101 **= 0001 1111**

d) 1111 1111 + 0001 0000 **= 0000 1111**

e) 1111 1010 + 0101 0110 = **0101 0000**

f) 1010 1010 + 0001 0101 = **1011 1111**

g) 0110 0110 + 1100 1100 = **0011 0010**

h) 1011 1100 + 0001 0011 = **1100 1111**

i) 1100 1001 + 1111 1001 = **1100 0010**

j) 1010 1000 + 1110 1010 = **1001 0010**

Adder disse tallene. Oppgi svaret på en word (16 bit) presisjon. Bruk penn og papir.

a) 0000 0000 0000 0000 + 0101 1110 1100 1000 = **0101 1110 1100 1000**

b) 1001 1001 1001 1001 + 0110 0110 0110 0110 = **1111 1111 1111 1111**

c) 0001 0111 0110 1001 + 0010 1010 1001 = **0001 1010 0001 0010**

d) 1111 0101 0001 + 0100 0110 1101 0011 = **0101 0110 0010 0100**

e) 1010 0110 1101 1010 + 0101 1010 0110 1111 = **0000 0001 0100 1001**

4) Enerkomplement og toerkomplement

a) Hva er enerkomplement og hva er toerkomplement?

Enerkomplement er en måte å representere negative tall binært der man «flipper» alle bitene i tallet. Den negative representasjonen av et tall vil altså være den inverse versjonen av det «vanlige binærtallet».
Toerkomplement er en måte å blant annet representere negative tall binært. Den første biten i binærtallet på toerkomplement form er såkalt «signed» og angir om tallet er positivt(0) eller negativt(1). For å gjøre om et vanlig binærtall til toerkomplement gjør man det først om til enerkomplement, deretter legger man til 1.

```
Eksempel: Vi vil finne hvordan man representerer -7 binært
1. Finn det binære tallet til den positive versjonen av
   tallet, altså 7: 0110
2. Gjør det om til enerkomplement (flipper bitene): 1001
3. Legger til 1: 1001 + 0001 = 1011

Svar: -7 er 1011 på toerkomplement binærtall
```

b) Bruk enerkomplement på disse(5) tallene:

- 0000 = **1111**
- 0101 = **1010**
- 1100 1110 = **0011 0001**
- 0011 0001 1111 0000 = **1100 1110 0000 1111**
- 1010 0011 = **0101 1100**

c) Hvorfor trenger man presisjon i toerkomplement?

For at den signede biten skal bli riktig, altså det som angir om noe er et positivt eller negativt tall, må vi vite presisjonen. Om vi ikke hadde hatt presisjon og beholdt evt. overflow ville tall som egentlig er positive tall kunne blitt negative og vice versa.

d) I 8 bit presisjon, hva er det største og det minste tallet man kan representere binært uten toerkompliment?

Det største tallet man kan representere er 255 eller 1111 1111.
Det minste tallet man kan representere er 0 eller 0000 0000.

e) I 8 bit presisjon, hva er det største og det minste tallet man kan representere med toerkomplement?

Det største tallet man kan representere er 127 eller 0111 1111.
Det minste tallet man kan representere er -128 eller 1000 0000.

Vi kan ikke representere tallet 128 med 8 bits presisjon i toerkomplement fordi den første biten angir om tallet er positivt (0) eller negativt (1). For å representere tallet 128 trenger vi minst 9 bits presisjon (noe vi i dette tilfellet ikke har). Binærtallet 1111 1111 representerer i toerkomplement tallet -1.

f) Hvilken bit er det som gjør at et tall er negativt i toerkomplement?

Biten lengst til venstre, også kjent som den mest signifikante biten (MSB).

g) Sant / Usant: Man kan ikke se av seg selv om et binærtall er på toerkomplement form, det er noe man må få oppgitt.

Svar: Sant.

h) Hva er «the silly number»? Også kjent som «tulletallet».

Med 8 bits presisjon på toerkomplement er «the silly number» -128 da den positive versjonen av tallet ikke kan representeres med den presisjonen som er oppgitt. I et eksempel med 16 bits presisjon på toerkomplement vil -1024 være «the silly number».

i) Forklar og vis med eksempler hva overflow er, i 8 bits presisjon.

Si at vi skal regne `1111 0011 + 1111 1100 = ?` og oppgi svaret i 8-bit presisjon, i utregningen under ser vi at svaret kommer bli 9 bit men på grund av at svaret skal oppgis i 8-bit presisjon så blir den niende biten «overflow» og vil med det ikke være med i svaret.

Mente som blir «Overflow»

```
  1  111
    1111 0011
+   1111 1100
    1110 1111
```

5) Konvertering toerkomplement

Konverter binærtallene på toerkomplement med 8 bits presisjon under til desimaltall. Bruk penn og papir.

Når man skal konvertere et binærtall til desimal, og det er oppgitt at det er brukt toerkomplement på det, vil man først se til den mest signifikante biten, og avgjøre om den er positiv eller negativ. Om den er 0 er tallet positivt, og om den er 1 er tallet negativt. (Her må man også passe på at man vet presisjonen)

Eksempel:

Si at vi har tallet 1010 0101 med 8 bits presisjon på toerkomplement. Da må vi se på tallet 8 plasser fra høyre, og avgjøre om tallet er negativt eller positivt. Vi ser at 'Sign'-biten er et 1-tall, som vil si at tallet er negativt. Den første biten tilsvarer tallet –128.

Alle de andre bitene kan man anse som positive, og dermed kan vi addere dem med –128 slik:

`1010 0101 = (-128) + 32 + 4 + 1 = -91`

Om den mest signifikante biten hadde vært **0** ville vi fått et helt annet resultat:

`0010 0101 = (0) + 32 + 4 + 1 = 37`

Og om vi skulle kjørt regnestykket **uten** toerkompliment er resultatet helt annerledes:

`1010 0101 = (128) + 32 + 4 + 1 = 165`

a) 1111 1111 **= -1**

b) 0000 1000 = **8**

c) 1000 0000 = **-128**

d) 1110 0000 = **-32**

e) 0010 0000 = **32**

f) 0110 0010 = **98**

g) 1100 1000 = **-56**

h) 1111 0111 = **-9**

Konverter desimaltallene under til binærtall på toerkomplement med 8 bits presisjon. Bruk penn og papir.

Når man skal konvertere et desimaltall til binært, og det er oppgitt at det er brukt toerkomplement på det, vil man først se om det er et positivt eller negativt tall, det vil avgjøre om den mest signifikante biten er 0 eller 1. Om den er 0 er tallet positivt, og om den er 1 er tallet negativt. (Her må man også passe på at man vet presisjonen).

Om tallet er negativt vet vi altså att MSB er 1, det betyr at vi nå må finne ut av hvilke fler at bitene som skal være 1 før å frem rett svar, «Hva må vi plusse -128 med for å få rett svar?».

Eksempel:

Hva er -58 i binært med 8-bit presisjon og toerkomplement?

(-128) + 58 = 70

70 – 64 = 6

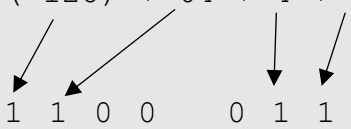
6 – 4 = 2

2 – 2 = 0

For å finne ut av hvilke tall som skal være 1 må vi følge det binære tallsystemet og se hvilket tall som er det nærmeste tallet under det opprinnelige tallet, eller samme tall som det opprinnelige tallet.

Nå har vi funnet ut av hvilke bit som skal være positive (1 og ikke 0), disse er 64, 4 og 2.

(-128) + 64 + 4 + 2 = -58



1 1 0 0 0 1 1 0

a) 10 = **0000 1010**

b) -8 = **1111 1000**

c) -77 = **1011 0011**

d) 69 = **0100 0101**

e) -42 = **1101 0110**

f) -153 = **Ingen løsning, tallet -153 er utenfor presisjonen vi har fått oppgitt.**

g) -12 = **1111 0100**

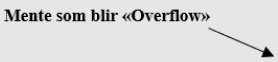
6) Subtraksjon og toerkomplement

Subtraher tallene under ved hjelp av toerkomplement med 8 bits presisjon. Oppgi svaret i binærtall (base 2) og vis utregning. Bruk penn og papir.

Eksempel:

Si at vi skal regne 23 – 4, det som er viktig å huske er at dette er de samme som 23 + (- 4).

1. Konverter til binært → 0001 0111 + (- 0000 0100) = ?
2. Bruk enerkomplement (flippe bitene) på det negative tallet.
3. (- 0000 0100) → 1111 1011
4. Bruk deretter toerkomplement (+ 1) på det flippede tallet.
5.
```
 1111 1011
+         1
 1111 1100
```
6. Plusse deretter det nye tallet med det opprinnelige positive tallet.



7.
```
    1  1111  1
     0001 0111
+    1111 1100
-------------
     0001 0011
```
8. Svar: 0001 0011
9. Siden vi får en overflow og MSB er 0 blir svaret positivt.

a) 16 – 8 = ?
0001 0000
+1111 1000
0000 1000

b) 24 – 9 = ?
0001 1000
+1111 0111
0000 1111

c) 127 – 128 = ?
0111 1111
+1000 0000
1111 1111

d) 98 – 100 = ?
0110 0010
+1001 1100
1111 1110

e) 87 – 19 = ?
0101 0111
+1110 1101
0100 0100

f) 78 – 12 = ?
0100 1110
+1111 0100
0100 0010

g) 82 – 69 = ?
0101 0010
+1011 1011
0000 1101

h) 42 – 42 = ?
0010 1010
+1101 0110
0000 0000

m) (10 – 12) – 2 = ?
0000 1010
+1111 0100
1111 1110

1111 1110
1111 1110
0000 0000

n) 127 – 119 – 7 – 1 = ?
0111 1111
+1000 1001
0000 1000

0000 1000
+1111 1001
0000 0001

0000 0001
+1111 1111
0000 0000

o) 50 – 10 = ?
0011 0010
+1111 0110
0010 1000

p) 77 – 99 = ?
0100 1101
+1001 1101
1110 1010

i) 12 – 20 = ?
0000 1100
+1110 1100
1111 1000

j) 15 – 24 = ?
0000 1111
+1110 1000
1111 0111

k) (10 – 5) – 9 = ?
0000 1010
+1111 1011
0000 0101

0000 0101
+1111 0111
1111 1100

l) (20 – 15) – 2 = ?
0001 0100
+1111 0001
0000 0101

0000 0101
+1111 1110
0000 0011

a) Hvorfor bruker vi heksadesimale tall?

   Med det binære tallsystemet blir det ofte mange bit å holde styr på, spesielt jo høyere tall man skal representere. Med det blir det også vanskeligere for oss mennesker å lese tallene. Vi bruker det heksadesimale tallsystemet fordi det kan forenkle 4 bit til kun ett siffer/karakter. Høye tall i det heksadesimale tallsystemet vil også være kortere enn i vårt «vanlige» desimale tallsystem.

b) Hva er prefix-en for heksadesimale tall?
   Svar: 0x

c) Hvor stor er en nibble?
   Svar: 4 bit

d) Skriv alle tall sifrene i det heksadesimale tallsystemet
   Svar: 0123456789ABCDEF

e) Hvor mange heksadesimale siffer trenger man for å representere 16 bit?

   To siffer som representerer 8 bit hver. Husk at 8 bit kan representere tall fra 0 – 15 som er tallene man kan representere på én plass i det heksadesimale tallsystemet.

8) Konvertering heksadesimal

Konverter tallene under fra det binære tallsystemet (base 2) til det heksadesimale tallsystemet (base 16).

a) 1000 = **0x8**

b) 0010 1101 = **0x2D**

c) 0101 1100 = **0x5C**

d) 1110 1110 = **0xEE**

Konverter tallene under fra det heksadesimale tallsystemet (base 16) til det binære tallsystemet (base 2).

a) 0x2 = **0010**

b) 0xFF = **1111 1111**

c) 0xD3 = **1101 0011**

d) 0x5F = **0101 1111**

Konverter tallene under fra det heksadesimale tallsystemet (base 16) til det desimale tallsystemet (base 10).

a) 0x3 = ***3***

b) 0xAB = ***171***

c) 0xFF = ***255***

d) 0x01 = **1**

Konverter tallene under fra det desimale tallsystemet (base 10) til det heksadesimale tallsystemet (base 16).

a) 7 = **0x7**

b) 130 = ***0x82***

c) 54 = ***0x36***

d) 23 457 = **0x5BA1**

9) Addisjon heksadesimal

Legg sammen tallene, og oppgi svaret på base 16 (heksadesimalt), bruk penn og papir. (Flere konverteringer kan være nødvendig underveis)

a) 0x00 + 0xFF = **0xFF**

b) 0x0F + 0xF0 = **0xFF**

c) 0xA5 + 0xC4 = **0x169**

d) 0x1D + 0xF4 = **0x111**

e) 0xA3 + 0x37 = **0xDA**

f) 0xDCBA + 0x1234 = **0xEEEE**

g) 0x2121 + 0x0F0F = **0x3030**

h) 0xAC5F + 0x0001 = **0xAC60**

i) 0xFFFF + 0x0 = **0xFFFF**

j) 0xBB8 + 0xB2D2 = **0xBE8A**

10) Subtraksjon heksadesimal

Subtraher tallene, og oppgi svaret på base 16 (heksadesimalt), bruk penn og papir. (Flere konverteringer kan være nødvendig underveis)

a) 0x14 – 0x03 **= 0x17**

b) 0xA6 – 0xA1 = **0x05**

c) 0xFF – 0xBC = **0x43**

d) 0xBB – 0xB5 = **0x06**

e) 0x7C – 0x33 = **0x49**

f) 0xF09D – 0xF00A = **0x0093**

g) 0xEDB – 0x8DA = **0x601**

h) 0xFFFF – 0xABCD = **0x5432**

i) 0x600F – 0x0A01 = **0x560E**

j) 0xFFFF – 0xFFFF = **0x0000**

11) ASCII

a) Bruk ASCII-tabellen og finn disse tegnene:
   - G – **0x47**
   - % **- 0x25**
   - = **- 0x3D**
   - } – **0x7D**
   - h – **0x68**
   - q – **0x71**

   Hva er heksadesimale tallet til hvert tegn?

b) Skriv ditt eget navn i heksadesimal fra ASCII tabellen på både store og små bokstaver.

   Eksempel:
   ole = 0x6F 0x6C 0x65
   OLE = 0x4F 0x4C 0x45

c) Hva er forskjellen på 0x41 og 0x61?

   0x41 = A og 0x61 = a. Dette er altså store og lille «A», hvis man ser på disse talene i binært ser vi at det er kun 1 bits forskjell mellom de.

   0x41 = 01**0**0 0001
   0x61 = 01**1**0 0001

d) Hva har binærtallet 0000 1000 som ASCII kode, binært?

   0000 1000 = 8, da må vi finne tallet «8» i ASCII tabellen hvilket er 0x38. 0x38 = 0011 1000, det betyr at svaret er **0011 1000**.

TK1100
FORELESNING 0x02 TXT MEDIA

1. Tegnsett og kodetabeller

a) Hva er en glyf?

En glyf er et tegn, i datasammenheng så er dette tegnet vi ser på skjermen som har blitt bestemt av en kodetabell, hvordan dette tegnet skal se ut bestemmes av font-tabellen.

b) Hva er et charset/tegnsett?

Det er det datamaskiner bruker før å finne ut hvilket tegn det er som skal vises på skjermen av et gitt heltall, eksempler på charset er UTF-8 og UTF 16.

c) Skriv litt om kodetabellene ASCII, ISO-8859-1, Windows 1252 og forskjellen mellom de.

I ditt svar burde du minimum vært innom:

- ASCII kom først og er 7-bit, og dermed er hvert tegn alltid 1 byte
- ISO-8859-1 og Windows 1252 er basert på ASCII
- Windows 1252 bygger også på ISO-8859-1 da denne kom sist av disse tre kodetabellene
- De første tegnene er kontrolltegn
- ASCII har kun det «amerikansk» alfabetet
- ISO-8859-1 har «undefined» tegn i tabellen
- Windows 1252 inneholder flere tegn som i dag ikke brukes ofte
- ISO-8859-1 og Windows 1252 inneholder «åæø» og andre nordiske bokstaver

d) Hva er ulempen med ASCII-kodetabellen?

Det støtter kun «amerikansk» alfabet og trenger dermed alternative kodetabeller og fonter for å fungere med andre språk med spesialtegn.

e) Slå opp i ASCII-tabellen og fyll inn riktig svar.

0x21 - !
0x44 - D
0x4F - O
0x43 - C
0x54 - T
0x45 - E
0x20 - Space

f) Hvorfor bruker ASCII kun syv bit?

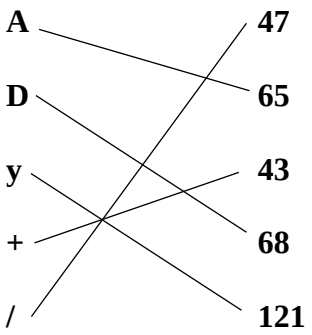
Med 7 bit vil den åttende biten fungere som en paritets bit, se på dette som en sjekk-bit, disse brukes som en error-sjekker og er lagd for å finne ut av om noe har gått galt. Fordi ASCII tabellen kun har 7 bit til å bruke til koding av tegnet/symbolet så finnes det 128 ulike tegn/symboler (0-127) istedenfor 256 som det er i for eksempel Windows 1252.

g) Bruk ASCII-tabellen til å sortere tegnene under i riktig rekkefølge. Det med minst tallverdi i tabellen til venstre, høyest verdi til høyre.

A
2
;
>
R
Y
V
!
$

Svar: ! $ 2 ; > A R V Y

h) Parr tegnene til venstre med sine respektive ASCII-koder(Desimal).



2. UNICODE

a) Skriv kort om UTF-8 og UTF-16, og forklar forskjellen mellom de.
I ditt svar burde du minimum vært innom:

- Sier hvordan kodepunktene skal kodes
- Er de mest brukte char-settene/tegnsettene i dag
- UTF-8 = minimum 1 byte
- UTF-16 = minimum 2 byte
- UTF-8 koder på samme måte som ASCII med 8 bit for U+0000 – U+007F, over U+007F bruker UTF-8 to eller flere byte for å kode et tegn
- Primært bruker UTF-16 to byte for å kode et tegn, her kommer UTF-8 å bruke tre eller mer byte for tegn der UTF-16 bruker kun to
- Å bruke UTF-8 når ASCII tegn er en stor del av de som skal kodes så er det en fordel, hvis ikke så er UTF-16 passende å bruke

b) Hva er prefix-en til Unicode?
Svar: U+

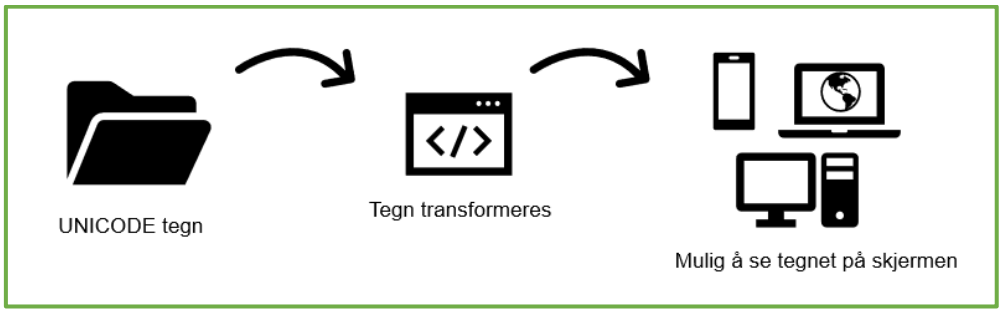
c) Hva er spesielt med kodene U+0000 til U+007F i Unicode?

Det er tegnene som er i ASCII tabellen, med andre ord så er de første tegnene i Unicode samme som i ASCII.

d) Se videoen: The Unicode Consortium Overview (5 min) og skriv kort om hva Unicode er og hva det brukes til.
https://www.youtube.com/watch?v=-n2nlPHEMG8&feature=youtu.be

I ditt svar burde du minimum vært innom:

- En struktur med kodepunkter delt opp i 17 plan, er ikke et charset/tegnset
- Unicode sier IKKE hvordan tegnene skal kodes binært
- Mulig å sende informasjon mellom alle, uavhengig av språk
- Alla typer av tegn og emojis
- Ulike charset/tegnsett, for eksempel UTF-32, UTF-16, UTF-8, Windows-1252, ISO-8859-1
- Charsetet/tegnsettet sier hvordan tegnene skal kodes binært



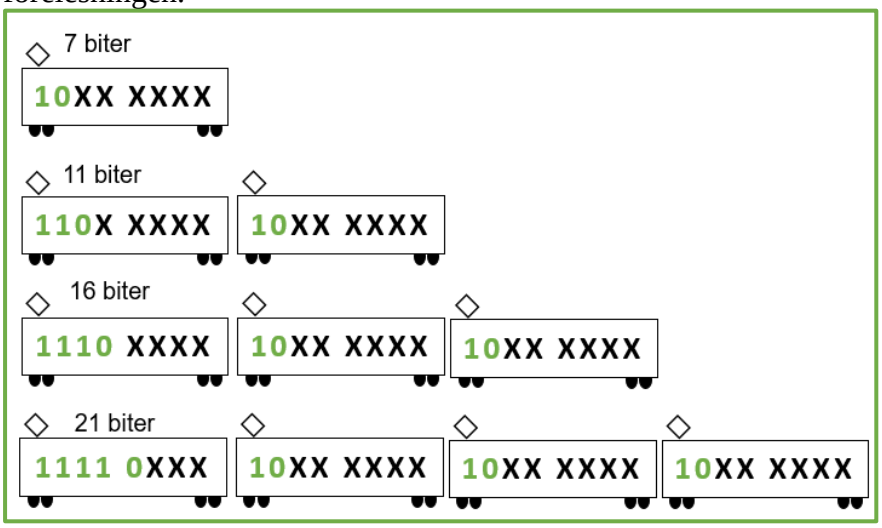
e) Hva er strukturen til Unicode?

17 plan hvor et plan inneholder 65536 kodepunkter, plan 0 (Basic Multilingual Plane) er det planet vi bruker mest, dette planet inneholder tegn for de vanligste språkene. Resterende plan inneholder, emojis, noter, uvanlige tegn, spesial tegn osv.

f) Hva brukes for å transformere Unicode-kodepunkter til binære tall, så datamaskiner kan lese de og vise tegn på skjermen?

Det er charset/tegnsett som brukes for å transformere disse til binære tall, de vi i dag bruker mest er UTF-8 og UTF-16.

g) Tegn opp «vognene» i Unicode UTF-8 (7, 11, 16 og 21 bit). Hint: se sildesettet fra forelesningen.



h) Forklar hva padding er i UTF-8 transformering.

Hvis «vognene» i transformeringen ikke blir fylt opp (se svar fra forrige spørsmål) vil plassene i vognene som er tomme fylles opp med padding, padding består alltid av '0'.

i) Regn disse punktene i UNICODE (se eksempel med karakteren «Å» i slidesettet fra forelesningen):

U+0042 - B
U+0003 - 3
U+00F1 - ñ
U+0067 - g
U+0021 - !

j) Hvor mange adresselokasjoner kan adresseres dersom start- og stoppadressene er som angitt: 0x3CA8 og 0x3CFF?

Svar: 88 (Desimal).
Hvis man konverterer de heksadesimale tallene til desimal, får man 15 528 og 15 615. Differansen mellom disse er 87, men siden det første tallet er start, og det siste er stopp, teller begge tallene, og dermed blir svaret 1 høyere.

3. Filformater

a) Hva er data og metadata i filformater?

Data ligger i Body og dette er data som skal brukes til noe, metadata forteller hvordan filen skal behandles av programmet som skal lese filen.

b) Hva er et «magic number», hvorfor trenger man det?

«Magic number» er starten av en fil fungerer som en ID, det vill si at dette nummert sier til noe om hva det er for typ av filformat.

c) Hva er «magic-number»-ene til filtypene .java, .exe og .pdf?

.java - CAFE BABE
.exe - MZ
.pdf - %PDF

d) Nevn noen typer tekst-filer, og noen typer binære filer.

Eksempel på svar:
Tekst-filer: HTML, CSS, SVG…
Binære filer: .class, .exe, .pdf…

4. Bilder, lyd og komprimering

a) Gi eksempel på når man bruker CMYK og når man bruker RGB. Forklar også kort hva additiv og subtraktiv er og om CMYK og RGB er additive eller subtraktive.

Eksempel på svar:
Subtraktiv: "Blekker" reflekterer den fargen vi vil se og absorberer de andre
Additiv: Skjermen stråler ut den fargen vi vil se
CMYK: Printere (Subtraktiv)
RGB: Skjerm (Additiv)

b) Hva er bruksområdene for vektorgrafikk mot Bitmap?

Vektorgrafikk som kan «blåses opp» i hvilken som helst fysisk størrelse, eksempel .svg filer, her bruker man seg av en matematisk formel for å lage formen du vil ha og då lagrer den formelen. Bitmap koder og lagrer hver enkelt piksel slik de skal vises på skjermen.

c) Hvorfor komprimerer man filer, og hvilke typer fil komprimerer man ofte?

Gjennom å komprimere filer vill man ha mulighet til å eksempel sende store filer over internett fortere, typiske filer som blir komprimert er media og bilder.

d) Gitt setningen: «Min a✪en år gamle ka✪ var beta✪ av en ha✪ så han må✪e på aku✪sykehus»

Finn ut hva ✪ er.

Svar: ✪ = tt

5. Repetisjon addisjon binært

Regn alle oppgaver med penn og papir, viktig at du skriver ned hele utregningen for hver oppgave.

a) 1001 0001 + 1111 0000 = ? (8 bit presisjon)
Svar: 1000 0001

b) 1111 0111 + 0111 1111 = ? (8 bit presisjon)
Svar: 0111 0110

c) 7 + 123 = ? (Hvis utregning i binært og gi svar i 16 bit presisjon)
Svar: 0000 0000 1000 0010

d) 0000 0100 + 0000 1111 = ? (16 bit presisjon)
Svar: 0000 0000 0001 0011

e) 37 + 23 = ? (Hvis utregning i binært og gi svar i 8 bit presisjon)
Svar: 0011 1100

f) 1000 1111 0011 0001 + 1010 0001 = ? (16 bit presisjon)
Svar: 1000 1111 1101 0010

g) 0001 0110 0001 1010 + 0111 1111 0001 0101 = ? (16 bit presisjon)
Svar: 1001 0101 0010 1111

6. Repetisjon Toerkomplement

Regn alle oppgaver med penn og papir, viktig at du skriver ned hele utregningen for hver oppgave.

a) 1001 1101 - 1111 0100 = ? (Gi svar i desimal)
Svar: -87 (1010 1001 i binært, MSB indikerer på at dette er et negativt tall)

b) 15 - 121 = ? (Hvis utregning i binært og gi svar i 8 bit presisjon)
Svar: 1001 0110

c) 45 - 14 = ? (Hvis utregning i binært og gi svar i 8 bit presisjon)
Svar: 0001 1111

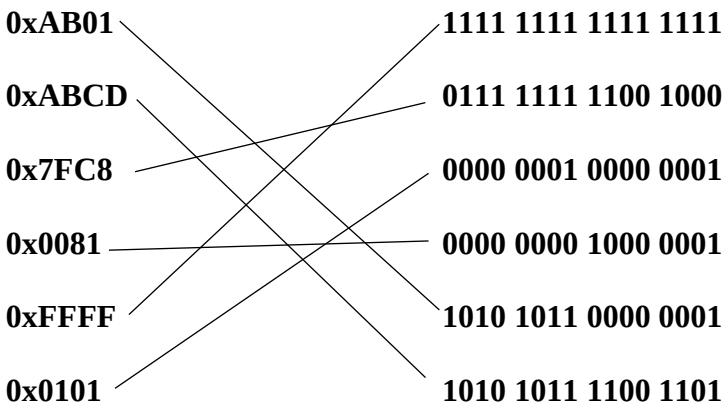
d) 1110 1101 0000 0001 - 1000 0001 = ? (16 bit presisjon)
Svar: 1110 1100 1000 0000

e) 0011 0010 1101 1110 - 0101 1101 1001 1111 = ? (16 bit presisjon)
Svar: 1110 1100 1000 0000

7. Repetisjon Heksadesimal

Regn alle oppgaver med penn og papir, viktig at du skriver ned hele utregningen for hver oppgave.

a) Par i hop heksadesimal-tall med binær-tall:



b) 0x007B + 0x7B06 = ? (Gi svar i heksadesimal, 16 bit presisjon binært og desimal)
Svar: 0x7B81, 0111 1011 1000 0001 og 31 617

c) 0x000F + 0x08C5 = ? (Gi svar i heksadesimal, 8 bit presisjon binært og desimal)
Svar: 0x8D4, 1101 0100 og 2 260

d) 0x001A – 0x0023 = ? (Gi svar i 8 bit presisjon binært, desimal og bruk toerkomplement)
Svar: 1111 0111 og -9

e) 0011 0010 1101 0110 = ? (Gi svar i heksadesimal og desimal)
Svar: 0x32D6 og 13 014

f) 0x0006 – 0x000E = ? (Bruk toerkomplement og gi svar i desimal)
Svar: -8

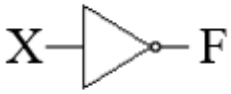
TK1100
FORELESNING 0x03 ARKITEKTUR

1) Boolsk Algebra

a) Tegn opp grunnoperasjonene(porter/gates) med tilhørende sannhetstabell

- NOT
- AND
- NAND
- OR
- NOR
- XOR
- XNOR

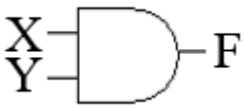
NOT



Sannhetstabell

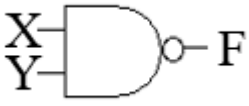
X	F
0	1
1	0

AND



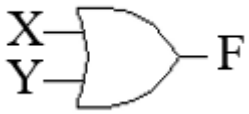
X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

NAND



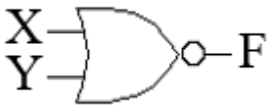
X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

OR



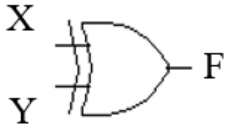
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

NOR



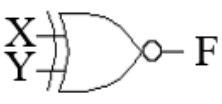
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

XOR



X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

XNOR



X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

b) Regneoppgaver, bruk sannhetstabellene til grunnoperasjonene. Bruk penn og papir.

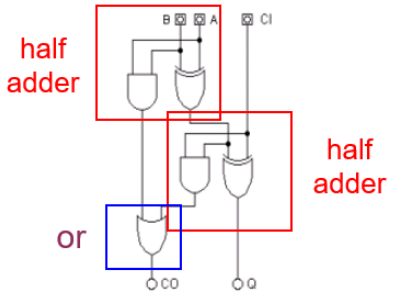
- 1100 0010 **AND** 1100 0000 = **1100 0000**
- 0000 1100 **OR** 0000 1111 = **0000 1111**
- 0101 0010 **XOR** 1110 1101 = **1011 1111**
- **NOT** 0110 1111 = **1001 0000**
- 0000 1000 **NOR** 1111 0000 = **0000 0111**
- 1111 1111 **AND** 1111 1111 = **1111 1111**
- 1000 1001 **NAND** 0001 1001 = **1111 0110**
- 1010 1010 **AND** 0101 0101 = **0000 0000**

c) Hva brukes en 8-bits full adder til? Forklar med egne ord hvordan den fungerer.

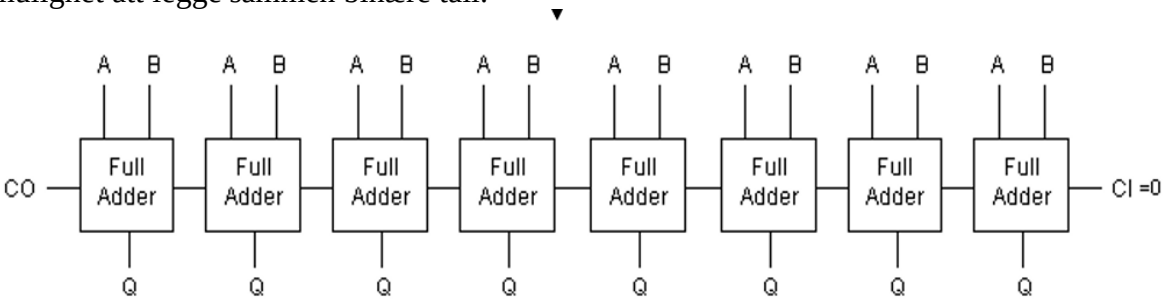
En 8-bits full adder brukes for å legge sammen tall. Når det er 8 bit betyr det at det er åtte fulle addere koplet sammen som har mulighet å legge sammen 8 bits tall. For å forklare å skjønne hvordan en 8-bits full adder fungerer så er det viktig å skjønne hvordan en full adder som kun legger sammen en bit fungerer.

Tenk litt som at du skulle regne selv, her skal vi legge sammen A og B (Se bilde under) og eventuelt de som kommer fra CI. CI er her «mente» som da kan være noe som er igjen fra forrige utregning.

A og B blir kjørt gjennom både en AND gate og en XOR gate, dette er noe som kalles for en «half adder». Svaret fra XOR kjøres gjennom en ny half adder sammen med CI (mente), og svaret fra de to AND-ene fra begge half adderene kjøres gjennom en OR port og dette blir den nye menten til neste adder, se CO i bilde under. Q er det sluttelige svaret fra en full adder.



I bilde under så vises det en 8-bit full adder, her ser vi at den har åtte «Full addere» og at CI (mente) sendes til neste adder i køen. Disse brukes altså i maskiner for å ha mulighet att legge sammen binære tall.



d) ASCII-koden for "F" er 0x46. Hvilket tegn/glyf får vi fra ASCII tabellen dersom vi utfører: 0x46 **OR** 0x20? Bruk penn og papir for utregning.

Tegnet «f» som er 0x66 i ASCII tabellen, **OR** operasjonen her endrer kun 1 bit:
0x46 = 01**0**0 0110
0x66 = 01**1**0 0110

0x20 kan faktisk 'OR-es' med alle store bokstaver i ASCII tabellen, for å få små bokstaver. Prøv det gjerne ut med andre bokstaver for å se resultatet!

e) Parr riktig logisk operator med bitwise operator.

2) Regneoppgaver

Alle oppgavene skal gjøres med penn og papir, vis utregning.

a) Følg stegene under:

1. Finn den heksadesimale verdien til tegnet ^ i ASCII-tabellen.
2. Gjør om verdien til det binære tallsystemet.
3. Bruk AND-operasjonen med verdien du har funnet og 0110 0001.
4. Gjør om det binære tallet du står igjen med til heksadesimal.
5. Finn tegnet til den heksadesimale verdien, dette er svaret på oppgaven.

Svaret er: @

b) (0x37 XOR 0xF3) XOR 0x37 = ? (Oppgi svar i heksadesimal)

```
(0011 0111 XOR 1111 0011) XOR 0011 0111 =
1100 0100 XOR 0011 0111 =
1111 0011

Konverter 1111 0011 til heksadesimal = 0xF3
```

3) Arkitektur

a) Nevn noen av de viktigste elementene på ett hovedkort.

Svaret burde være innom disse punktene:

- CPU - Prosessor
- North bridge – kobler sammen prosessor, minne og GPU
- South bridge – direkte koblet til North bridge og resterende komponenter på hovedkortet, eks. I/O enheter og BIOS
- BIOS – Inneholder alt for oppstart av maskinen
- Ulike slotts for å koble RAM, periferiutstyr, harddisk etc., eksempel PCI, SATA, IDE.

b) Gi eksempler på hva som kan kobles til et hovedkort.

Her finnes det mange riktige svar, eksempel:

- GPU (Graphics Processing Unit), grafikkort
- Harddisk
- Minne, eks. RAM
- Lydkort
- Nettverkskort
- Mus
- Tastatur
- Printer
- Ekstern skjerm
- Ekstern harddisk
- Etc.

c) Forklar hva en GPU er.

En GPU er enten direkte tilkoblet til hovedkortet eller på et grafikkort som senere blir koblet til hovedkortet, det er GPU-en som prosesserer signaler fra maskinen til skjermen. Dette er altså en mikroprosessor som er designet for å utføre grafikkrelaterende beregninger, dette er altså ikke noe som CPU-en utfører.

4) CPU

a) Hva er en instruksjon?

Det er instruksjonen til CPU-en i en maskin som forteller hvilke oppgaver den skal og hva slags data den skal utføre det med. Man sier altså til maskinen hva den skal gjøre med hjelp av instruksjoner.

Hvilke typer av instruksjoner en prosessor har mulighet til å utføre er klarlagt i et så kallet instruksjonssett, eksempel x86.

b) Forklar forskjellene mellom en CISC-prosessor og en RISC-prosessor.

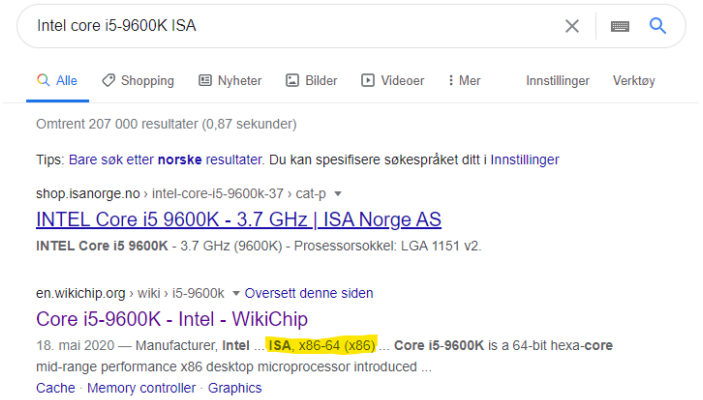
CISC betyr Complex Instruction Set Computer og RISC betyr Reduced Instruction Set Computer. Gjennom navnene kan man skjønne at CISC er for mer komplekse system (desktop og server) og RISC er for mindre system som er enklere og trenger en veldig rask utførelse fra prosessoren (embedded og mobil). CISC og RISC er to forskjellige arkitekturer av en prosessor, her finnes det i dag også CPU-er som bruker seg av begge arkitekturene, eksempel moderne Intel/AMD CPU-er hvor CISC er eksternt og RISC er på selve prosessoren i maskinen. Andre produsenter av blant annet prosessorer er MIPS og ARM, de fokuserer på RISC arkitektur.

Maskiner som bruker RISC utfører en instruksjon per klokkesyklus og CISC maskiner kan ha instruksjoner som bruker mer enn en syklus for å bli utført. Når man bruker RISC trengs det mer RAM minne enn når man bruker CISC. Det vil si at CISC bruker RAM mer effektivt. En instruksjon på en CISC maskin kan trenge flere instruksjoner på en maskin med RISC arkitektur da denne kun utfører en instruksjon per klokkesyklus.

c) Finn ut av hva slags type prosessor du har i din PC og telefon, og hva slags ISA den prosessoren har.

Her er noen guider på hvordan finne prosessor i PC-en:
Windows:
https://www.howtogeek.com/413942/how-to-see-what-cpu-is-in-your-pc-and-how-fast-it-is/
MacOS:
https://www.techjunkie.com/find-mac-cpu-model/
Linux:
https://www.cyberciti.biz/faq/check-how-many-cpus-are-there-in-linux-system/

Etter du har funnet ut navnet på cpuen skal det være lett å google seg frem til hvilken ISA den har. Typ:



d) Hva er en mikrokontroller og hvor brukes det?

Typisk eksempel på hvor man bruker mikrokontroller er embedded systems, her er det mikrokontrolleren som styrer systemet hvor den ofte kun kjører et program, som gjerne kan være enkelt. Det finnes mange ulike embedded system, de vi kjenner godt til er vaskemaskiner, klokkeradioer, hjemme-routere etc. En mikrokontroller er altså ett alternativ til CPU. Her er den store forskjellen at en CPU er kun en prosessor og bruker seg av operativsystem, men mikrokontroller har også minne, I/O, lagring og bruker sjeldent operativsystem.

e) Skriv kort om x86 instruksjonssett.

Svaret burde være innom disse punktene:

- Eksempel på hvor det brukes, for eksempel prosessorer med CISC arkitektur
- Hvorfor det finnes instruksjonssett, for at forstå at en prosessor trenger retningslinjer på hvordan den skal agere.
- Hvilke deler x86 inneholder (Data flytting, Kontrolloverføring, Aritmetikk/Logisk, Input/Output, Debug og Interrupt håndtering)

5) Teknikker og begreper

a) Hva er forskjellen på en system-klokke og en vanlig klokke.

En system-klokke viser ikke tiden, den holder takten i et system for at komponentene på hovedkortet skal være synkronisert og sikkert, hvilket betyr at de utfører sin oppgave når system-klokken er på «high» og ikke når den er på «low».

b) Når du kjører et program på en datamaskin, hvor ligger det fulle programmet?

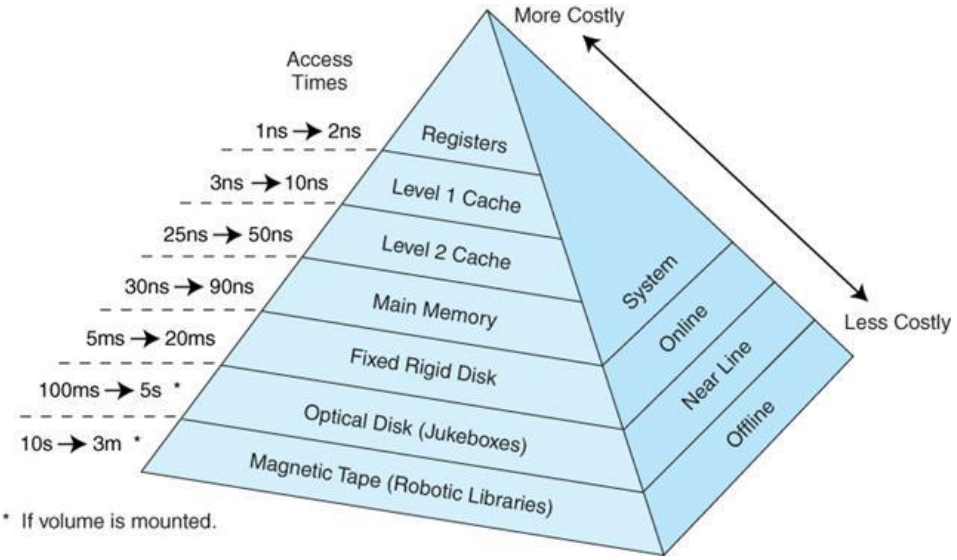
Når du kjører et program så vil det fullstendige programmet ligge i RAM-en, her er det kun enkeltinstruksjoner og enkeltdata som ligger på CPU-en når programmet er i gang. CPU-en får altså instruksjoner og data fra RAM-en og disse går via North bridge for å komme frem til CPU-en.

c) Hvorfor har en CPU flere kjerner?

For å gjøre det mulig å kjøre flere program samtidig og at det skal gå raskt. Se for eksempel hvor mange program du har oppe og kjører akkurat nå, dette hadde ikke CPU-en klart hvis den ikke hadde hatt flere kjerner.

d) Hva representer minnehierarkiet?

I minnehierarkiet er det mulighet til å få forståelse og oversikt av hvilken type av minnelagring som er enkel å aksessere, mest sparsom og mest effektiv.



e) Ut ifra minnehierarkiet hva slags type lagring ville du brukt om du skulle lagret noe du bare trengte tilgang til hvert 5. år.

Her kan svar på oppgaven variere, det som er viktig er å forstå forskjellene mellom de ulike typer av lagring og hva som er det positive/negative med de. På grunn av at denne informasjonen kun skal hentes ut hvert 5. år så er en type offline løsning et bra alternativ, dette kan for eksempel være en ekstern harddisk.

f) Hva brukes cache-minnet til?

Det tar tid før CPU-en å få informasjon fra RAM-en, så for at det skal gå kjappere å få frem informasjonen så finns cache-minne som ofte er lokalisert direkte på CPU-en. Når CPU-en kjører et program å skal hente informasjon så sjekker den først cachen for å se om de den søker finnes der, om ikke så sjekker den i RAM-en. I dag er det vanlig med flere cache-minner, disse kan ha ulike spesifikasjoner og effektivitet.

g) Hva går en CPU gjennom for å hente data fra RAM-en.

For å hente informasjon fram RAM-en trenger CPU-en å gå gjennom North bridge, her er det North bridge som er direkte koblet til både CPU-en og RAM-en.

6) Repetisjonsspørsmål

a) Vi arbeider med 8 bits presisjon og toerkomplement. Utfør binærsubtraksjonen: 1000 0111 - 0010 0001 = ?

Svar: 0110 0110

b) Foreta en logisk OR mellom 1001 1111 og 0101 1011

Svar: 1001 1111 OR 0101 1011 = **1101 1111**

c) 1001 1111 & 0101 1011 **= ?**

Svar: 0001 1011

d) Hva blir resultatet dersom man subtraherer («trekker») ASCII-koden for tegnet «3» fra ASCII-koden for tegnet «0»? Oppgi svaret i heksadesimal.

Svar: 0xFD

e) Hva er UTF-8 kodingen av Unicode-tegnet U+00BD (1⁄2)?

- 0xBD
- **0xC2BD**
- 0x00BD
- 0xBD00
- Ingen av alternativene over

f) Hva blir resultatet av den binære addisjonen 0100 1001 + 0101 0101?

- **1001 1110**
- 1100 0000
- 1100 1111
- 0100 0001
- 0111 1111

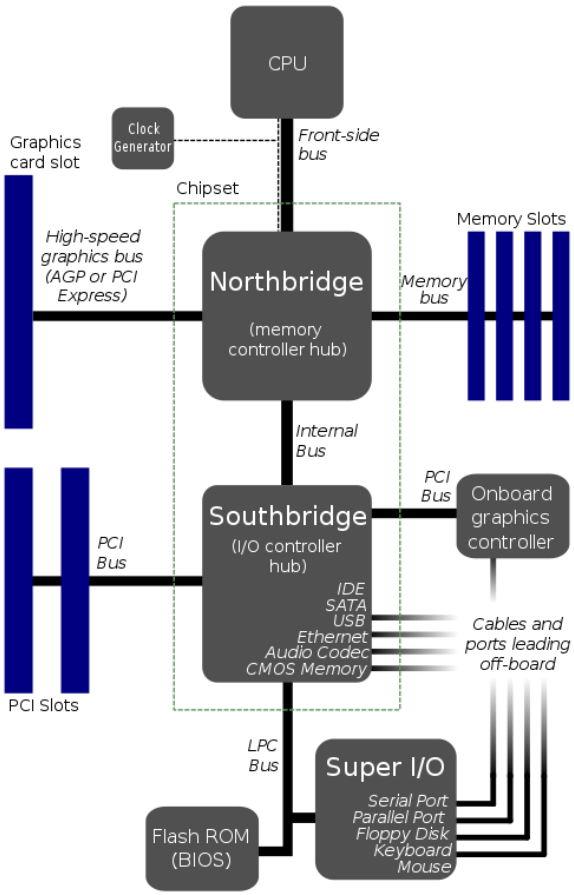
g) Hvordan vil det negative tallet 53 (-53) bli lagret som en byte?

- 0101 0011
- 1101 0101
- 1100 1010
- **1100 1011**
- 1101 0110

TK1100
FORELESNING 0x04 ORGANISERING

1. Hovedkortet

a) Tegn et enkelt hovedkort med de viktigste komponentene/elementene.



b) Hva er forskjellen mellom integrert og ikke-integrerte hovedkort?

Et integrert hovedkort er det vanligste i dag og her er flere av komponentene allerede ferdig innebygd i selve kortet, for eksempel CPU, skjermkort, GPU, nettverkskort etc. De fleste Laptoper har integrerte hovedkort, da dette tar mye mindre plass i maskinen.

Et ikke-integrert hovedkort er helt enkelt tvertom, her er flere komponenter ikke ferdig innebygd og de monteres som et tillegg. Dette er det vanligste blant stasjonære datamaskiner. Likevel er det nokså vanlig i dag at hovedkort i stasjonære maskiner har integrert lydkort eller andre enkle komponenter, og kan sees på som et "delvis integrert" hovedkort.

c) Sant/usant: Det er Southbridge som støtter og kontrollerer kommunikasjon mellom CPU og primærminne (RAM).

Usant. Det er North Bridge som har denne oppgaven.

d) Hvilket chipset kontrollerer hvilke data, instruksjoner og kontroll-signaler som flyter mellom prosessor (CPU) og system-minne (RAM)?

North Bridge, dette er på grunn av at det er North Bridge som kobler CPU og RAM sammen og her må data, instruksjoner og kontroll-signaler gå via North Bridge for å komme frem og tilbake til CPU-en.

2. CPU og busser

a) Hva er en CPU-socket?

CPU-socketen er den som kobler CPU-en sammen med hovedkortet, her finnes det mange ulike sockets avhengig på hvilken CPU som skal kobles til hovedkortet.

b) Hvorfor trenger en CPU nedkjøling, og hvilke ulike måter finnes det for å kjøle ned en CPU?

CPU-en bruker mye strøm for å kunne fungere, energien fra strømmen gir ifra seg varme og for at CPU-en ikke skal bli for varm når den jobber så har vi CPU-nedkjøling. De ulike måtene for å kjøle ned en CPU er ved luft (via vifter) eller vann, her er det luft som er den mest vanlige måten og det er den som brukes i de fleste maskinene i dag. De som bruker vannkjøling, gjør oftest dette i kombinasjon med vifter.

c) Hva er de typiske problemene med en CPU?

Det som er de typiske problemene er enten kjøling eller strøm, her er de vanlige symptomene at maskinen ikke booter, ikke laster in OS, krasjer ved kjøring av applikasjoner eller plutselige POST-feil.

d) Hva er et chipset?

Dette er kontroller som styrer all flyt av data, instruksjoner og kontroll-signaler mellom CPU-en og annet utstyr som er tilkoblet til hovedkortet. I starten av PC-en sin historie fantes det kun et chipset, i dag er det typisk to stykk chipset som fungerer som hovedkontrollere (North bridge og South bridge).

e) Hvilken del har vanligvis størst betydning for PC-en sin samlede ytelse?
- South bridge
- **North bridge**
- L3 Cache
- Super I/O kontrolleren
- Ingen av alternativene over

f) Hvorfor finnes det «busser» i en pc?

Bussene i en maskin er koblingen mellom komponentene, det er via bussene som data, adresse, kontroll-signaler og strøm transporteres gjennom. Det vil si at bussene i en maskin har flere oppgaver og disse er:

- Adressering av data
- Frakt av data
- Synkronisering av maskinens komponenter
- Flytkontroll mellom komponentene
- Signalering mellom komponentene
- Strømførsel til noen komponenter

Det finnes også to hovedbusser i en PC og disse er:

- System/FSB: Forbinder CPU med North bridge og RAM
- Ekspansjon: Forbinder chipset med ekspansjons-slotts

g) Forklar kort forskjellen mellom synkrone og asynkrone komponenter.

Synkronisering er veldig viktig for at en maskin skal fungere, her er det viktig at komponentene utfører sin oppgave i riktig takt så at det blir riktig med komponentene den skal kommunisere med eller er avhengig av. For å synkronisere komponentene i en maskin så brukes det klokker som fungerer som taktgivere, synkrone komponenter holder samme eller tilsvarende takt til en klokke i maskinen og asynkrone komponenter har en egen takt de holder seg til.

h) Hva står IRQ for og hva er det?

IRQ = Interrupt Request. Dette er signaler som sendes til CPU-en for å gi beskjed at CPU-en skal avbryte pågående bearbeiding for å ta imot et såkalt «Interrupt». (For eksempel når du trykker på en knapp, tastatur eller mus, og trenger umiddelbar respons.) Disse «Interrupt-ene» har et nummer som indikerer på hvor de kommer ifra, for eksempel hvis det kommer en IRQ med nummer 1 så betyr dette at det kommer fra tastaturet.

3. Oppstart av maskinen

a) Hvor på PC-en lagres grunnleggende innstillinger slik som dagens dato og hvordan harddisker er konfigurert?

De lagres i CMOS. Her bruker for eksempel BIOS disse innstillingene som ligger lagret i CMOS for å utføre sine oppgaver, det er altså i CMOS maskinvare innstilningene ligger lagret når maskinen er slått av. Tenk at når du starter PC-en så trenger du ikke å sette opp dato/tid hver gang uten dette er lagret i CMOS og gjør det mulig at datoen og tiden stemmer til hver tid.

b) Forklar kort hvordan en boot-sekvens foregår.

Forklaringen burde inneholde disse delene:

1. Strømforsyningen
2. BIOS lastes in i RAM
3. «boot» bekreftes
4. POST starter
5. Programvaren for video adapter lastes
6. BIOS for annet utstyr lastes
7. CMOS konfigurering testes
8. Porter blir tildelt og utstyr blir konfigurert
9. Hardware konfigurasjon blir bekreftet
10. OS blir lokalisert og lastes inn

c) Hvis du får en feilmelding med 2xx, hvilken del av din maskin er det feil på da? Fra hvilken test får du denne feilmeldingen ifra?

Denne feilmeldingen blir vist på skjerm når POST (Power-On Self-Test) blir kjørt og betyr at det er noe feil med RAM-en til maskinen. Merk at dette finnes flere ulike feilmeldinger som indikerer at de er noe feil med RAM-en, men alle starter med «2».

d) Dersom en PC-en booter uten problemer, men alltid "fryser"/stopper etter noen få minutter, så er den mest sannsynlige årsaken at

- CPU-en er i gang med å gå i stykker
- BIOS er i gang med å slutte å fungere
- CPU-en får ikke nok strøm
- **CPU-en blir for varm**

e) Forklar kort hva BIOS er og hva som er forskjellene på BIOS og UEFI.

BIOS (Basic Input/Output System) er for 16 bit maskiner og UEFI støtter 32 og 64 bit maskiner, det er også forskjell på hvor stort minne de begge støtter hvor UEFI har større kapasitet enn BIOS. UEFI har også et grensesnitt som er mer morderne og utviklet, i grensesnittet til BIOS er det kun mulig at bruke tastaturet for å gi kommandoen. Det finnes flere detaljerte forskjeller, les denne artikkelen for mer informasjon: https://www.freecodecamp.org/news/uefi-vs-bios/

4. RAM

a) Hva er forskjellen mellom «BIT», «BYTE» og «WORD»?

BIT: 1 eller 0 / av eller på
BYTE: 8 bit, minste adresserbare enhet i RAM
WORD: Ulike betydninger: størrelsen på et register i maskinen (antall bit en CPU kan prosessere som en enhet) eller adresseringsenhet til RAM

b) Hva brukes RAM til?

RAM (Random Access Memory) gir applikasjonene/programmene som aktivt brukes på maskinen et sted å lagres under en kortere tid, dette gir da muligheten til CPU-en å hente data fra RAM-en, hvilket går mye raskere enn å hente det i masselageret (for eksempel. harddisk). Dette betyr at desto flere applikasjoner/programmer du kjører på din maskin, desto mer RAM-minne trenger den.

c) Forklar kort hva DRAM, SDRAM og SRAM er.

Alle disse er ulike typer av RAM, kort forklart hva de innebærer:

DRAM: er dynamisk og sender ikke data i takt med CPU-en sin klokke, det betyr at den er asynkron og sender instruksjoner så fort den får input fra brukeren istedenfor for å vente på klokken til CPU-en. DRAM er også et minne som må «friskes opp», det betyr at denne typen av RAM må bli supplert av strøm for å beholde minne.

SDRAM: Synchronous DRAM, dette er altså samme som DRAM, men den går i takt med CPU-en og kommer derfor å sende instruksjoner i samme takt som CPU-en sin klokke.

SRAM: Dette er et RAM som IKKE må «friskes opp» for å beholde det som er lagret, dette er en type RAM som er raskere enn DRAM men det er en mer komplisert konstruksjon og er dermed dyrere å produsere.

d) Hva er typiske RAM problemer?

Eksempel:

- RAM-et er satt inn feil i maskinen
- Konfigurert feil
- Minne-Modul har løsnet eller er av feil type
- Datamaskinen støtter ikke den typen RAM du har kjøpt, og ergo passer den ikke i RAM-sloten.

e) Finn ut hvor mye RAM-minne du har på din maskin.

Forslag på lenker for å finne ut av dette:

Windows: https://uk.pcmag.com/gallery/120326/how-much-ram-do-i-have-in-my-pc

OSX: https://www.macworld.co.uk/how-to/much-ram-need-more-3793418/

OSX/LINUX: https://www.howtogeek.com/howto/28475/how-to-tell-what-type-of-memory-your-linux-pc-has-installed/

5. Periferiutstyr

a) Hva står I/O for?

Svar: Input/Output, dette er en forkortelse som beskriver dataflyt eller kommunikasjon mellom to parter. Disse to kan for eksempel være en PC og et menneske eller et hovedkort og et nettverkskort.

b) Hvorfor har I/O-enheter «Device controller»?

Det er som sagt vanlig at en I/O-enhet består av «selve enheten» og en elektronisk komponent som kalles for «Device Controller». Her trenger enheten denne kontrolleren for å fungere for sitt bruksområde. For eksempel er det kontrolleren som tar imot instruksjoner og ser til at data er tilgjengelig i RAM-en.

c) «Device controller» i en I/O-enhet har registre, hvorfor har de det?

Dette er på grunn av at kontrollerne skal ha mulighet til å kommunisere med CPU-en, hvor det er mulig å skrive til registret for å kontrollere enheten, og for å lagre status som da blir mulig å lese for å finne ut status til enheten.

d) Hva er PCI, AGP, ISA og PCIe eksempel på?

Svar: Ekspansjonsbusser. Disse er altså ulike eksempler på busser som transporterer informasjon mellom kortkontrolleren og chipset-et.

e) Hva er prinsippet bak LCD-skjermer?

LCD står for Liquid Crystal Display og er en type «Flat-panel» skjerm. En LCD-skjerm består av pixler og her er det kun pixlene som blir endret når bildet oppdateres, dette er mulig da hver pixel har en egen adresse.

6. Masselager

a) Hva er forskjellen mellom masselager og minne?

I masselageret lagres data permanent, mens i minnet lagres data midlertidig.
Eksempel på masselager er: harddisk, CD og DVD.
Eksempel på minne er: RAM, ROM og Flash.

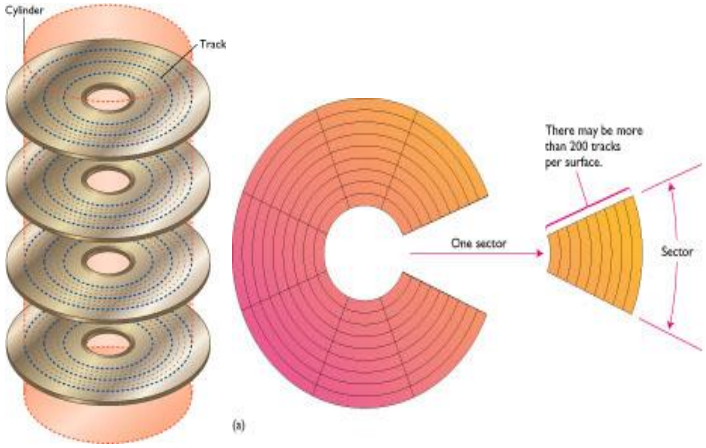
b) Gi eksempel på en kontroller som brukes for overføring av data til/fra masselager.

Eksempel: IDE/EIDE, SATA og SCSI.

c) Hvilke er de tre delene i diskens organisering? Og hva er diskens logiske organisering for noe?

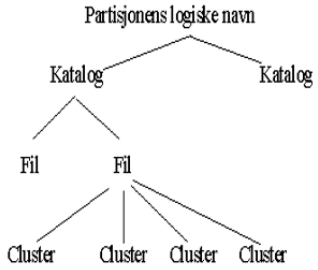
Diskens organisering:

- *Spor* – området lesehodet dekker på en rotasjon
- *Sektor* – minste delen av et spor som kan leses
- *Sylinder* – alle spor som er i samme posisjon (under hverandre) på disken (en disk består av flere disker)



Diskens logiske organisering:
Dette er en organisering for å vise hvordan hierarkiet er oppbygget i disken, her er det totalt fire deler i hierarkiet: Cluster, fil, katalog og partisjon.

- En *Partisjon* får ofte navnet C: D: eller E: på maskinen.
- En *katalog* kan også kalles en mappe/folder.
- En *fil* kan inneholde flere *clustere*, som er det minste OS kan lese.



d) Hvorfor er det bra med sikkerhetskopi? Nevn også noen muligheter for å sikkerhetskopiere data på din maskin.

For at du skal være sikker på at du ikke tapper noe data så er det veldig viktig med sikkerhetskopiering. Her finnes det mange anledninger til at data blir tapt og det er viktigst å bruke sikkerhetskopiering for den mest sensitive/viktige dataen.

Eksempel på hvordan du kan sikkerhetskopiere data på din maskin er å ta bruk av skylagring eller eksternt masselager, for eksempel ekstern harddisk.

e) Se denne videoen: https://www.youtube.com/watch?v=U-OCdTeZLac og forklar hvordan RAID fungerer og hvorfor det finnes.

Du burde vært innom disse delene i ditt svar:

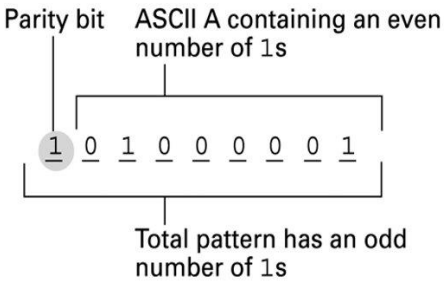
- De ulike funksjonene for RAID 0, RAID 1, RAID 5, RAID 10
- Aksessering av data med de ulike RAID-en
- Pålitelighet av de ulike RAID-en
- Hva som brukes mest i dag

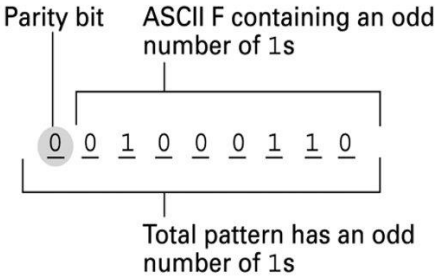
f) Kan en fil bli deles opp i flere biter på disken og fortsatt være intakt, og i så fall hva kalles denne prosessen?

Ja, dette er fullt mulig og veldig nødvendig for å kunne bruke resterende plass på disk etter at programmer for eksempel blir slettet, og små tomrom ligger igjen. Denne prosessen kalles *fragmentering*.

g) Hvilke av følgende påstander er riktig om paritetsbit i denne ASCII-blokken:
1000 0110

- Det heter 7-bit ASCII, og dermed har denne blokken 1 bit for mye.
- Det er et partall av 1-ere ekskludert paritetsbit, ergo er det feil i denne blokken.
- **Det er et oddetall av totale 1-ere (inkludert paritetsbit), ergo er det ingen feil i denne blokken.**
- Det siste sifferet er 0, altså ikke det samme som paritetsbiten, ergo er det feil.



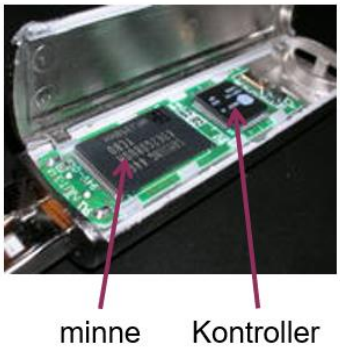


7. USB og flash-minne

a) Hva er Flash Memory? Hvilke typer operasjoner brukes i Flash Memory?

Flash Memory er et minne som brukes for å lagre og transportere data mellom en PC og en enhet, et vanlig eksempel på dette er en USB flash driver. Operasjoner som brukes i Flash Memory er NOR og NAND.

Eksempel på hvordan en USB flash drive ser ut:



b) Hvilke typer av USB brukes mest i dag?

USB-A, USB-B, Mini-USB, Micro-USB og USB-C (husk at USB finnes i ulike «Speed Standards» også).

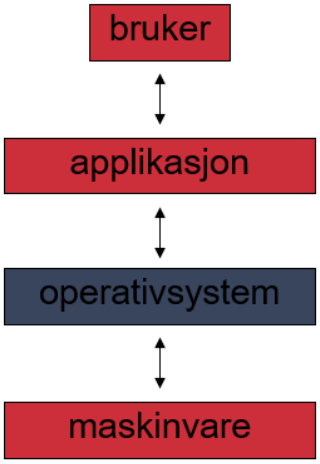
TK1100
FORELESNING 0x05 OS

1. OS

a) Forklar med egne ord hva et operativsystem er og hvorfor man trenger/ikke trenger et OS.

Her finnes det forskjellige svar da oppgaven sier at man skal forklare med egne ord. Det er viktig å få med at operativsystemer ligger mellom applikasjonen og maskinvaren, her fungerer operativsystemet som grensesnitt mellom applikasjonen og maskinvaren, for å at de skal klara av å kommunisere med hverandre. Operativsystemet tar «vekk» informasjonen som finnes i maskinvaren/hardwaren som brukeren ikke trenger eller skjønner noe av, og tilbyr derfor brukeren og applikasjonene en virtuell maskin som er enkel å bruke.

Operativsystemet fungerer også som en ressursadministrator, her er det operativsystemet som gir hvert program tid og plass på ressursene som finnes i maskinen, eks. CPU, minne etc.



b) Hva betyr abstraksjon i OS sammenheng?

Å abstrahere er å ta vekk detaljer som ikke er nødvendige å se for den aktuelle personen, i OS sammenheng så abstraherer operativsystemet detaljer fra maskinvaren for både brukeren og applikasjonene som kjører på maskinen. Det betyr at operativsystemet viser kun den informasjonen eller detaljene som brukeren/applikasjonen trenger for å kunne fungere eller bruke maskinen. For brukeren oppfattes abstraksjoner i form av symboler som er mulig å klikke på, for en applikasjon tilbyr operativsystemet systemkall som gjør det mulig for applikasjonen at kjøre det den er egnet for.

c) Hva er forskjellen på Monolithic kernel og Micro kernel?

Det er forskjellige måter en kjerne, og operativsystemet i seg selv, kan struktureres på. De to arkitekturene man ofte deler operativsystemer inn i er *Monolithic Kernel* og *Micro Kernel.*

En mikrokernel er en kjerne med minimal funksjonalitet. Den har kun det den trenger, som ofte vil det si at den kan administrere mine, prosessorer og interrupts, og evt andre små oppgaver som er nødvendig for systemet. En mikrokernel er ofte veldig lett å bygge på, utvide og endre. Samtidig har den noen faktorer som gjør den mer inefektiv enn andre kjerner.

En monolithic kernel fungerer derimot som en motpart til mikrokernelen. Den er en stor samling av masse funksjoner, der alt er linket sammen til et objekt. Den monolitiske kjernen kan ofte være veldig effektiv og kompleks, da alt er optimalisert på de beste måter, men når det er sagt er de ofte så store og komplekse at det er vanskelig å gjøre endringer. Det finnes mye kode som ingen har oversikt over, og det krasjer veldig lett.

Ofte sier man at de fleste operativsystemer starter som en mikrokernel, men ettersom man legger til mer funksjonalitet, likner den mer og mer på en monolotisk kjerne.

d) Hva er de viktigste funksjonene i operativsystemer?

- Brukergrensesnitt
- Applikasjons-kjøring
- Håndtering av ressurser
- Håndtering av maskinvare
- Håndtering av nettverk
- Sikkerhet

e) Forklar kort hva en ressurs er i sammenheng med et OS.

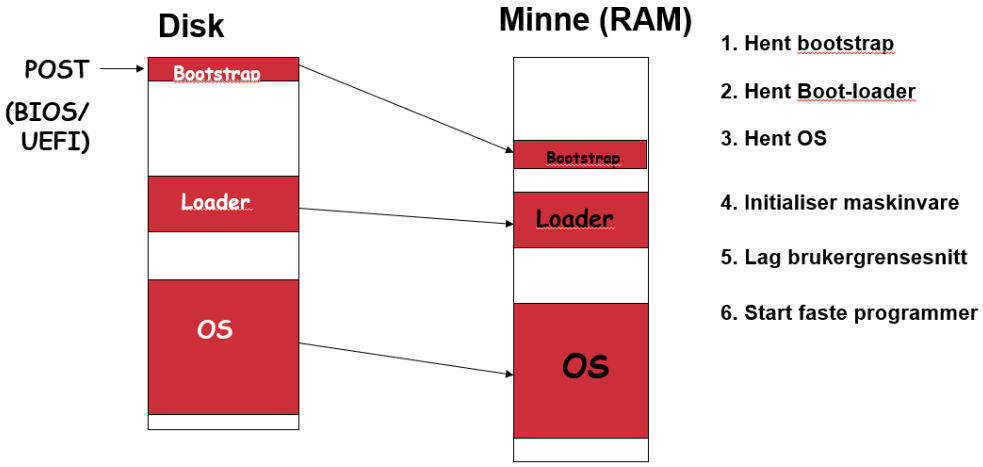
Operativsystemet fungerer som en ressursadministrator, det betyr at det er operativsystemet som sier når et program får tilgang til en ressurs og gir også plass til programmet på ressursen. En ressurs er et element i maskinen som programmet trenger å ta i bruk for å kunne bli kjørt, eks. prosessorer, minne, eksternt lager og I/O enheter.

2. Kjernen

a) Forklar hva som skjer i en oppstart av en maskin med operativsystem.

En såkalt bootstrapping er prosessen for å laste ned de grunnleggende delen i et program inn i minnet under en oppstart eller omstart av maskinen.

Se for deg at du skal starte din pc, når du trykker på knappen så hentes da bootstapen, boot-loadern og operativsystemet in til pc sin RAM. Etter det så sjekkes maskinvaren for å se at det ikke er noe feil med den, når det er gjort så lages brukergrensesnittet som er det vi ser på skjermen og til sist så starter de faste programmene seg (de som starter ved en oppstart uten at du som bruker fysisk setter i gang de).



b) Hva er en kjerne/kernel og hva har den for oppgaver?

Kjernen i en pc er et program, som ligger i sentrum av operativsystemet, og er et av de viktigste elementene for at en datamaskin skal fungere. Den starter opp maskinen, konfigurerer alt av hardware, og administrerer alle prosesser, tråder og ressurser som kjøres i pc-en. Uten kjernen ville ikke pc-en klart å skru seg på ordentlig, og den er kritisk for at du skal kunne bruke maskinen.

c) Hva er risikoene ved å kjøre på kjernemodus?

Når du kjører en maskin i *kjernemodus*, vil det si at du har de samme rettighetene som kjernen din har rettigheter til. Dette inkluderer tilgang til hele minnet, og rettigheter til å kjøre absolutt alle instruksjoner som er på maskinen. Dette vil si at det du velger å kjøre, likesågodt kan krasje hele pc-en din, bruke opp alt av minne, eller ødelegge hardware. Maskinen stopper deg ikke fra noen av disse tingene, siden du har fått rettighetene. Kjernemodus har de høyeste rettighetene man kan få på en pc, og er også kjent som “ring 0”.

d) Når skal man kjøre på bruker- vs kjernemodus i et OS?

De fleste vanlige applikasjoner kjører i brukermodus, og dette fordi det er mye tryggere å begrense rettighetene til en applikasjon fra å kunne tukle med hardware. Eksempler på slike applikasjoner: Word, Calculator, Chrome, World of Warcraft. Noen prosesser må likevel kjøre i kjernemodus, fordi det trenger å aksessere ressurser direkte. Eksempler på disse er: Operativsystem, BIOS, Drivere.

e) Hva er en prosess og en tråd, og hva er forskjellen mellom de?

En prosess er selve programmet som kjører med sine ressurser. En tråd er en del av programmet, og styrer de spesifikke instruksjonene. En prosess kan inneholde alt mellom en til mange tråder som gjør mange små oppgaver samtidig.

f) Hvilket/hvilke program på din datamaskin bruker mest av maskinen sin CPU akkurat nå?

Windows: Bruk taskmanager
OSX: Bruk Activity Monitor
LINUX/OSX: Bruk terminal med kommando fra slideserie

g) Hvorfor bruker operativsystemet Multitasking og hvordan fungerer kommunikasjonen her mellom OS og CPU? Hvordan hadde datamaskinen vært hvis det ikke fantes Multitasking?

Multitasking er når en maskin kan utnytte flere prosesser samtidig. Vi nevnte at en prosess kan ha flere tråder som kjører samtidig, men en datamaskin kan også ha flere prosesser som kjører samtidig. Hvis du vil kjøre flere prosesser samtidig, da trenger du multitasking. Multitasking oppnås ved at Operativsystemet sender en instruksjon til CPU-en. CPU-en kan bare behandle en instruksjon fra en prosess av gangen, per kjerne den har, så d

h) Hvilke tre ulike states finnes i “Context switching”?

- Running: prosessen som kjører akkurat nå
- Ready: Aktivisert prosess og venter i kø
- Blocked: prosess som venter på å bli aktivisert

i) Hva skjer hvis RAM-minnet ikke er stort nok?

I RAM-minnet så ligger alla program som kjøres, hvis det er så at RAM-minnet er for lite kommer noe av innholdet midlertidig legges på harddisken. Dette sier seg selv at nå kommer program kjøres veldig mye langsommere når de ligger data på harddisken også, det tar ekstrem mye lengre tid å hente data fra harddisken en fra RAM-minnet da RAM-en er koblet til North bridge som også er koblet direkte til CPU-en.

j) Hvordan vet et Operativsystem hvilken tråd som skal kjøres når?

Operativsystemet bruker seg av en type av kjøreplan for å vite når og hva som skal kjøres, her er det altså operativsystemet som bestemmer om hvilken tråd som får kjøre på prosessoren til hvilken tid.

k) Hvilke ulike deler har prosessen (tråden) i sitt minne?

- Code segment
- Data segment
- Stack segment
- System data segment (PCB)
- Evt. Flere stacker for trådene

l) Sant/usant: heapen i processen sitt minne er dynamisk.

Sant: at heapen er dynamisk betyr at den kan endre størrelse begge veier.

m) Nevn noen forskjeller mellom HEAP og STACK i en prosess sitt minne.

HEAP er dynamisk og vokser oppover, STACK er statisk og vokser nedover.

n) Hvorfor trenger hardware drivere og hva har det med OS å gjøre? Nevn også noen eksempel på drivere.

Drivere brukes som et grensesnitt mellom hardware og software. Operativsystemet trenger drivere for å kunne «forstå» hvordan det skal bruke hardwaret, hvar slags instrksjuner den kan sende til det osv. Uten drivere ville ikke operativsystemet klart å benytte hardware som blir plugget inn i PC-en (Grafikk-kort, mus, skjerm, vifter, CPU, etc.). Det finnes drivere for alle disse komponentene, men mange er «Plug'n'play» som vil si at du slipper å laste ned noe programvare for å bruke det. Det blir automatsikt lastet ned når du plugger det inn.

3. Skallet

a) «Signaler», «Flag» og «Redirects» bruker man i skjellet, forklar kort hva disse er.

Signaler: Når du kjører i et command line interface så er det mulig å gi signaler til operativsystemet, se dette som typer av kommando med hurtigtaster, for eksempel så kan du trykke `Ctrl-C` for å si «Aborter kjørende prosess».
Flag: Når du skal skrive in en kommando i command line interface har du mulighet å modifisere kommandot med hjelp av ulike flagg. Teste for eksempel `dir /?` (Windows) på din pc og se hva du får opp.
Redirects: Bruker ofte disse for å omdirigere output, for eksempel (finnes flere eksempel enn disse):

```
> skriver output til en fil
>> skriver output in på slutten av en fil
< tar innholdet i en fil
| tar output fra kommando og leverer den som input til
neste kommando
```

b) Ut ifra hvilket OS du har på din maskin, velg fremgangsmåte for WINDOIWS eller OSX/LINUX.

WINDOWS - *Utfør disse oppgavene i CMD (Command promt)*

1. Naviger til Dokumenter-mappen på maskinen din
2. List alle filene som finnes i dokument-mappen
3. Lag et directory(mappe)
4. Gå inn i mappen
5. Skriv in dette for å lage en fil: **notepad** kommandofil.txt
6. Nå skal et nytt notepad-vindu åpnes, skriv valgfri tekst i filen, lagre og lukk notepad-vinduet
7. Skriv ut innholdet i filen
8. Gå tilbake til CMD og kopier .txt-filen, du skal nå ha 2 filer
9. List innholdet i mappen på nytt, nå skal du ha **to filer i mappen**
10. Bytt navn på én av filene
11. Ta nå én av filene og flytt den ut til Dokumenter-mappen
12. Til slutt prøv deg frem med help kommando i CMD

OSX/LINUX - *Utfør disse oppgavene i Terminal.*

1. Naviger til Dokumenter-mappen på maskinen din
2. List alle filene som finnes i dokument-mappen
3. Lag et directory(mappe)
4. Gå inn i mappen
5. Skriv inn dette for å lage en ny fil: echo TK1104 er det beste faget > kommandofil.txt
6. Skriv ut innholdet i filen
7. Kopier kommandofil.txt
8. List innholdet i mappen på nytt, nå er det to filer i mappen
9. Bytt navn på én av filene
10. Flytt en av filene til Dokumenter-mappen
11. Til slutt prøv deg frem med man kommando i terminalen

WINDOWS	OSX/LINUX
dir	ls
type	cat
copy	cp
ren	mv
del	rm
rd	rmdir
find	grep
md	mkdir
cd	cd
help	man

4. Historikk

https://www.youtube.com/watch?v=26QPDBe-NB8
Se YouTube videoen, hent informasjon fra den og stoffet på slideserien, skriv siden 500 ord om OS og historien/utviklingen til OS. Bruk gjerne andre kilder til dette også.

TK1100
FORELESNING 0x06 WWW

1. WWW og historie

a) Forklar forskjellen mellom Internett og WWW (World Wide Web)

Internettet er et stort nettverk av maskiner som er koblet sammen og kommuniserer med hverandre. Internettet kjører på TCP/IP suiten, og bruker mange protokoller.
World Wide Web er en applikasjon som kjører på internettet, og er ansvarlig for du aksesserer via en browser/nettleser. Det finnes også andre typer Web, som holder på annen type informasjon og nettsider, eller har andre sikkerhetsnivåer. Eksempler på dette er Deep Web og Dark Web.

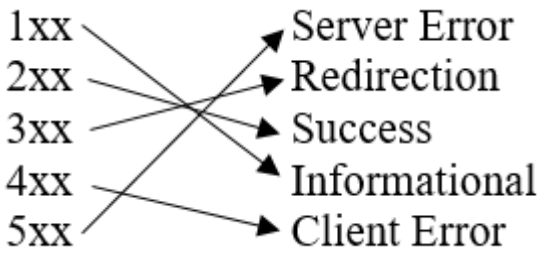
b) Gi noen eksempler på når du ville brukt utrykket «De Facto».

- «Man skal *de facto* alltid levere dokumenter i PDF format, om annet ikke er spesifisert»
- «Å bruke QWERTY oppsettet på keyboardet er en *de facto* standard. (Nesten alle gjør det)»
- «Når man skal koble til en skjerm (i nyere tid), vil man *de facto* bruke HDMI, Display Port eller VGA kabel.»
- «Når man får en presang, skal man *de facto* alltid si takk»

c) Hva står RFC for?

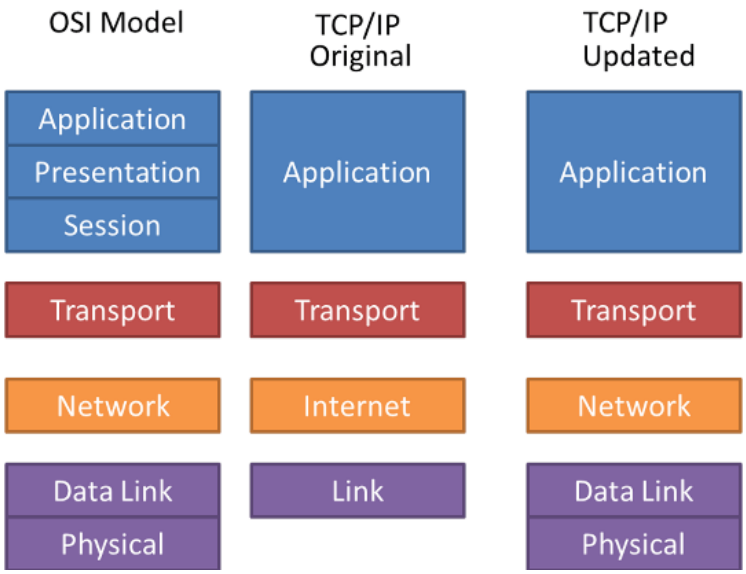
Svar: Request for Comment

d) Sett strek mellom de riktige statuskodene.



e) Nevn noen forskjeller mellom OSI og TCP/IP modellene.

OSI modellen har 7 lag (Application, Presentation, Session, Transport, Network, Data Link og Physical), mens TCP/IP bare (tidligere) hadde 4 (Application, Transport, Internet og Network Interface).
Likevel er det viktig å huske på at når man snakker om TCP/IP i dag, vil det være en kombinasjon av de to tidligere modellene, med 5 forskjellige lag. (Applikasjon, Transport, Nettverk, Datalink og fysisk)



Likevel er det forskjeller i hvordan disse lagene blir utført og brukt. TCP/IP headeren er blant annet 20bytes stor, mens OSI headeren bare er på 5bytes.

f) List opp de viktigste hendelsene i internettets historikk.

- Kleinrocks **Pakkeswitching**-prinsipp i 1961.
- Første **ARPAnet** node operativ i 1969.
- **Norge** blir en del av ARPAnet i 1972.
- **SMTP** og begynnelsen på email i 1982.
- **TCP** som standardprotokoll for ARPAnet i 1983.
- TCP **metningskontroll**, **IPv4** og **UDP** i 1988.
- Berner-Lee oppfant **HTML** og **HTTP** i 1991.

2. Protokoller og andre begreper

a) Sant/Usant: Protokoller definerer hvordan kommunikasjon over internett skal fungere.

Sant. Dette kommer vi til å gå grundigere gjennom i senere forelesninger.

b) Hva er forskjellen mellom en Klient-tjener-modell og en Peer-peer modell, tegn og vis.

En Klient Tjener modell er en modell der to parter kommuniserer med hverandre. Den ene av disse partene er en Tjener, ofte en server, som sitter på informasjon om for eksempel en nettside og dens innhold. Den andre parten er en klient, ofte en bruker av nettsiden, som vil sende en forespørsel til tjeneren om å for eksempel få informasjon, sende filer, logge inn eller se på en video.

Peer to Peer modellen har i motsetning ikke noe forskjell på medlemmene av den. Alle brukere kan fungere som både «klienter» og «tjenere», og den baserer seg på at noen brukere laster opp innhold, mens andre laster ned. Dette brukes ofte når man skal dele store filer over store mengder folk. For eksempel er det enkelte spillselskaper som velger å bruke dette, blant annet Blizzard, som har det i StarCraft 2, World of Warcraft m.m.

c) Hvilken port kobler en TCP-forbindelse seg mot for en hjemmeside.

Hvis nettsiden benytter http, går TCP over port 80, men hvis siden benytter https, går TCP over port 443.

d) Bruk tracert (Windows) eller traceroute (OSX/LINUX) i terminalen mot www.vg.no , hvor mange stop gjør ruten?

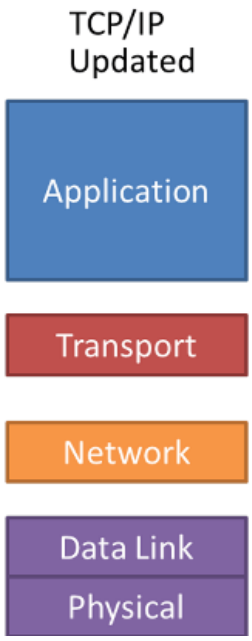
Eksempel på output, i dette tilfelle er det 6 stop fra maskinen frem til vg.no.

```
C:\Users>tracert vg.no

Tracing route to vg.no [195.88.55.16]
over a maximum of 30 hops:

  1     3 ms     2 ms     3 ms  172.26.0.1
  2     5 ms     2 ms     3 ms  cB152584D.static.as2116.net [77.88.82.177]
  3     6 ms     2 ms     3 ms  193.90.113.51
  4     3 ms     9 ms    10 ms  ae21.cr2.oslosda310.as2116.net [193.75.1.76]
  5     4 ms     4 ms     3 ms  he5-2-0.ar2.ulv89.as2116.net [193.90.113.91]
  6     3 ms     2 ms     3 ms  www.vg.no [195.88.55.16]

Trace complete.
```

e) Hvorfor deler man opp internett i ulike lag?

Dette er av flere grunner, der en av dem er struktur. Siden internettet er så stort, og det er så mange protokoller, har man funnet en måte å strukturere dette på. En annen av dem er abstraksjon og sikkerhet. For å sende en pakke over internett, må den først pakkes ned i mange ulike lag med kode, før den blir sendt, og senere pakket opp igjen gjennom alle lagene. De forskjellige lagene fikser på forskjellige problemer som kan ha oppstått i prosessen med de andre lagene. På et vis, kan man nesten også si at grunnen for at vi har så mange lag rett og slett er fordi internettet ikke var bygget for den mengden brukere vi er i dag, og dermed har man måttet fikse problemer underveis, ergo ble alle protokollene og lagene opprettet.

f) I forelesning snakket vi om organisering av internett: Tegn opp den oppdaterte TCP/IP modellen.



g) Hvorfor er internett usikkert oppbygget?

Den største grunnen er fordi at når internett ble laget var det ikke tenkt at det skulle bli så svært som det er i dag og derfor lagdes det ikke med innebygd sikkerhet. Man har med tiden skjønt at det finnes mange sikkerhetsfeil i internett og når man først har oppdaget en feil, har man kun løst *det* problemet, ofte med å lage en protokoll. Derfor er internettet som vi kjenner i dag en stor pakke med mange ulike protokoller.

h) Hva kan gå galt når data overføres over internett?

De tre hovedtingene som kan gå galt når informasjon transporteres over internett er tap av data, båndbredde og timing:

Tap av data: Avhengig av hva som brukes for å transportere informasjonen, er det mulig med tap av data, her er det noen applikasjoner som tåler litt tap av data (Feks. Streamingtjenester (Youtube) og enkelte spill), mens noe andre må ha 100% pålitelig transport av data (Email, dokument-deling).

Båndbredde: noen applikasjoner krever et minimum båndbredde (dataoverførings-kapasitet) for å fungere, for eksempel audio og video, mens noe kan bruke båndbredden dynamisk og «rette seg etter kapasitet», for eksempel filoverføring.

Timing: Om det er tidsforsinkelser så kan opplevelsen av ytelsen minskes, dette er vanlig i spill eller sanntidsprosesser.

i) For å se på nettverkstrafikk i Chrome:

1. Åpne Chrome
2. Høyreklikk og trykk på «Inspiser»
3. Velg fanen «Network» lengt opp i inspiser-vindu
4. Skriv in valgfri URL i browsern og trykk enter
5. Orienter deg siden litt rundt i inspiser-vindu for å se hva som skjer når du åpner en hjemmeside, klikk også rundt litt på hjemmesiden og se hva som skjer.
6. Gjør siden det samme med www.ikke.no

3. HTTP

a) Sant/Usant: HTTP oppbevarer informasjon om tidligere forespørsler.

Usant, HTTP protokollen er det man kaller for «Stateless».

b) Sant/Usant: En URL er en type URI

Sant. URI (Uniform Resource Identifier) er en streng som inneholder karakterer for å identifisere en ressurs, URL (Uniform Resource Locator) er en type av URI og brukes for å spesifisere en lokasjon av en ressurs på et nettverk og har mulighet å hente in ressursen.

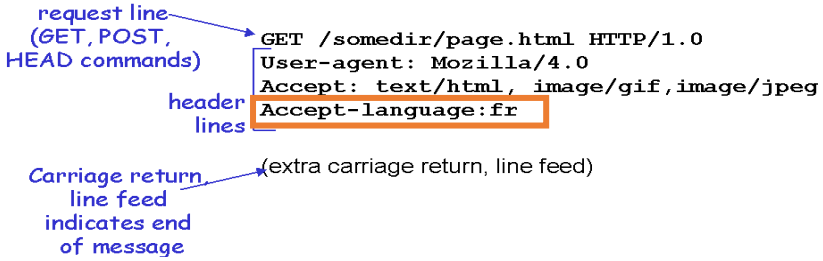
c) https://www.kristiania.no/sok?phrase=Digital%20Teknologi
Identifiser de ulike delene av HTTP url-en, host, port, path, query, fragment. Alle er ikke nødvendigvis tilstede. Vet du også hva %20 betyr her?

https://www.kristiania.no/sok**?**phrase=Digital%20Teknologi

Host : https://www.kristiania.no
Path : /sok
Query: phrase=Digital%20Teknologi

%20 henter ut Ascii tegnet som tilsvarer hexadecimal 20. Dette er tenget for «mellomrom» / [SPACE].

d) I HTTP 1.0 GET-request, hvordan spesifiseres språk?



e) Hvilken type metoder har HTTP 1.1 som ikke HTTP 1.0 har?

HTTP 1.0: GET, POST, HEAD
HTTP 1.1: GET, POST, HEAD, PUT, DELETE, OPTION, TRACE

f) Gi tre eksempler på statuskoder og hva de betyr.

200 OK
- spørring vellykket, objektet kommer senere i meldingen

301 Moved Permanently
- etterspurt objekt flyttet, ny adresse senere i meldingen

400 Bad Request
- spørring ikke forstått av tjeneren

404 Not Found
- etterspurt dokument/fil ikke funnet på denne tjeneren

505 HTTP Version Not Supported

4. DNS

a) Hva brukes DNS til?

Når du skal inn på en hjemmeside, skriver du inn navnet på siden i browsern, her er det ikke enkelt å finne hvilken hjemmeside som skal lastes inn i din browser hvis ikke man vet hvor informasjonen skal hentes, og det er det DNS brukes til. Se på det som et personnummer eller et ID-nummer, www.vg.no har en adresse og det er denne adressen som trengs for å finne ut hvor informasjonen skal hentes fra, og denne adressen kommer fra DNS. DNS-en inneholder altså en type register hvor en IP-adresse er knyttet til en hjemmeside, fungerer på samme måte som en telefonbok hvor det går an å søke opp på navn for å finne telefonnummeret.

b) Hvordan er navne-tjenere fordelt?

DNS er hele systemet hvor det finnes flere navne-tjenere, her er det oppdelt for at det blant annet skal bli raskere å få tak i informasjonen man ser etter, men også for at det skal bli mer sikkert. Fordelingen er gjort hierarkisk og når din maskin spør etter en adresse så spør den først den nærmeste navne-tjeneren, hvis den ikke har informasjonen du ser etter, spør den neste navne-tjener i leddet. Totalt finnes det 13 stk. hoved navne-tjenere i verden og disse blir da kontaktet av lokale navne-tjenere ved behov.

c) Bruk `nslookup` mot www.vg.no i terminalen, hvilken informasjon vises?

```
C:\Users>nslookup www.vg.no
Server:  OSLDC102.EGMS.no
Address:  172.25.4.11

Non-authoritative answer:
Name:    www.vg.no
Addresses:  2001:67c:21e0::16
          195.88.55.16
          195.88.54.16
```

d) Prøv deg frem og endre Resource Records når du skal kjøre `nslookup`, hvordan ser de ulike svarene ut? Er det annerledes fra når du kjørte det første gang?

Eksempel på hvordan det ser ut med type **A** og **NS**:

```
C:\Users>nslookup
Default Server:  OSLDC102.EGMS.no
Address:  172.25.4.11

> set type=A
> www.vg.no
Server:  OSLDC102.EGMS.no
Address:  172.25.4.11

Non-authoritative answer:
Name:    www.vg.no
Addresses:  195.88.54.16
          195.88.55.16
```

```
> set type=NS
> www.vg.no
Server:  OSLDC102.EGMS.no
Address:  172.25.4.11

vg.no
        primary name server = fw.hmg9.vgnett.no
        responsible mail addr = hostmaster.ns.vg.no
        serial  = 2021011402
        refresh = 300 (5 mins)
        retry   = 150 (2 mins 30 secs)
        expire  = 2419200 (28 days)
        default TTL = 300 (5 mins)
>
```

TK1100
FORELESNING 0x07 APPLIKASJONSLAGET

1. Applikasjonslaget

a) Hva er tjenerens oppgave i klient-tjener modellen?

Tjeneren kan for eksempel være en server som sitter på informasjon om en nettside og dess innhold, her skal da tjeneren gi ønsket data som er etterspurt av klienten. Det er altså tjenerens oppgave å tjene klienten.

b) Hva er det en del store spill bruker for type av oppsett i nettverket for å transportere informasjon? Og hva er det som er bra/dårlig med dette oppsettet?

Oppsettet som blir brukt i noen spill er Peer-to-Peer modellen, her er det mulighet for alle enheter i nettverket å fungere som en tjenere eller klient. Her kan det være vanskelig med sikkerhet og pålitelighet, for eksempel blir alle ressurser som er tilegnelige på nettverket delt mellom enheter/«peers» uten en sentral server som er innblandet.

c) Hvordan er det applikasjonen på maskinen (applikasjonslaget) kommuniserer med transportlaget?

Via «Sockets» eller også kalt internetts API (Application Programming Interface), her er det laget med sockets som brukes for å sende data inn i socket og lese data ut fra socket.

d) Praktisk oppgave med `curl`

- Åpne terminalen på maskinen din og skriv `curl -v http://www.vg.no`
- Hva er HTTP statuskoden du får, og hva må du eventuelt gjøre for å få http statuskoden 200 OK?

```
C:\Users>curl -v http://www.vg.no
* Rebuilt URL to: http://www.vg.no/
*   Trying 195.88.54.16...
* TCP_NODELAY set
* Connected to www.vg.no (195.88.54.16) port 80 (#0)
> GET / HTTP/1.1
> Host: www.vg.no
> User-Agent: curl/7.55.1
> Accept: */*
>
< HTTP/1.1 301 Moved
< Date: Fri, 05 Mar 2021 08:50:50 GMT
< Server: Varnish
< X-Varnish: 65781497
< location: https://www.vg.no/
< Content-Length: 0
< Connection: keep-alive
<
* Connection #0 to host www.vg.no left intact
```

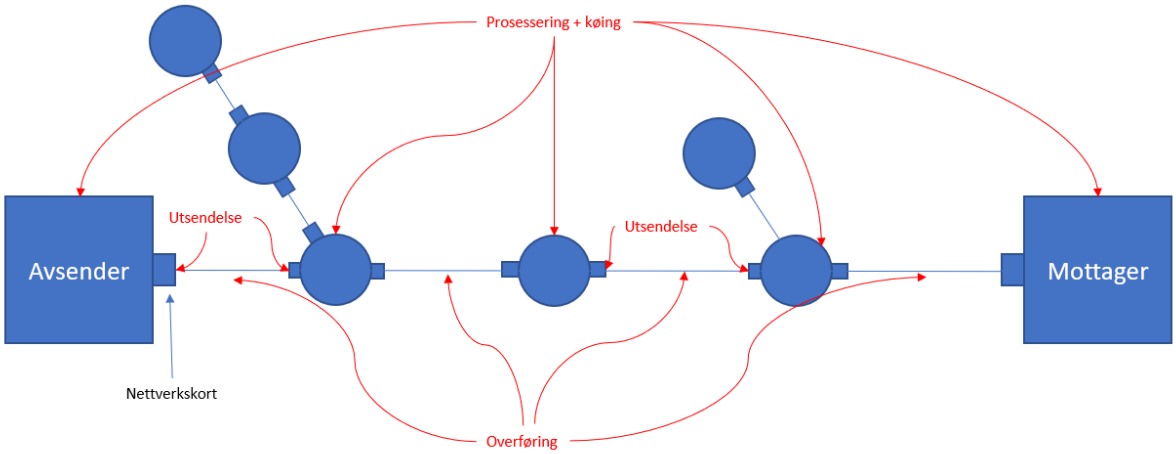
For at kommando skal fungere: bytt http med https

e) Hva står UDP og TCP for, og hva er disse for noe?

Disse er transport protokoller som brukes i internett, neste forelesning kommer vi se nærmere begge.

UDP: User Datagram Protocol
TCP: Transmission Control Protocol

f) Tegn opp og forklar de fire ulike forsinkelse typene.



2. DNS

a) Hva står DNS for?

Domain Name System.

b) Hva menes med en hoved navne-tjener og TLD?

Hoved navne-tjenere er de navne-tjenerne som blir kontaktet av lokale tjenere om de lokale tjenerne ikke har den informasjonen du spør etter, TLD (Top Level Domain) er toppnivådomenene og der er disse domenene hoved navne-tjenere har oversikt over.

c) Hvilken default name server bruker du akkurat nå, og hva er adressen til den? (Hint, ta i bruk terminalen)

Svar: skriv in nslookup i terminalen, mulig å skrive nslookup + adresse til en nettside.

d) Hvilke typer av spørringer finnes det når en maskin skal finne adressen gjennom en navnetjener?

Enten er spørringene rekursive eller iterative, for rekursiv spørring får du svaret fra samme server uansett om den ikke har informasjonen du spør etter og må spørre en annen server. Hvis det er iterative spørring får den som spør etter informasjon en adresse til neste navne-tjener og spør den direkte selv, enn å få svaret fra den første serven den spurte.

e) Hvorfor finnes DNS records?

Det finns for at vite hvordan informasjonen i DNS skal lagres, det er da enklere for både mennesker og maskiner å jobbe med informasjonen når det finnes en struktur. De vanligste er A, NS, CNAME og MX.

3. EPOST

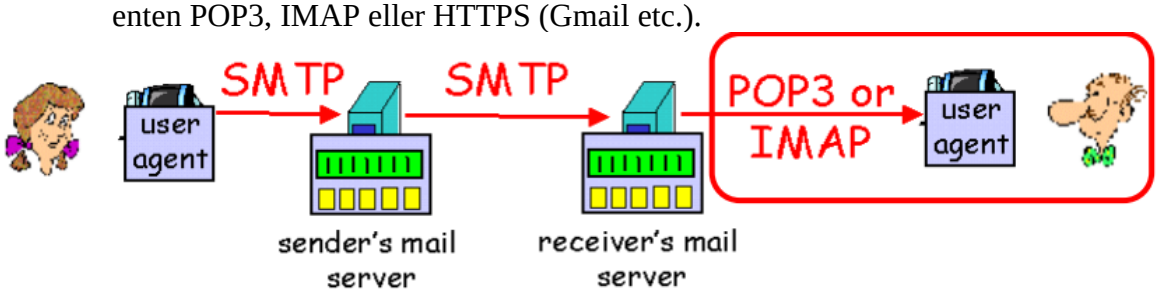
a) Hva er en protokoll og hva brukes SMTP protokollen til?

En protokoll er en standard som er satt for at to enheter skal ha mulighet å kommunisere med hverandre, her er det altså lagt opp på hvordan de skal kommunisere fast de ikke snakker samme «språk».

SMTP er en protokoll for å overføre post fra klient til tjener over internett, altså så brukes denne protokollen for å sende epost.

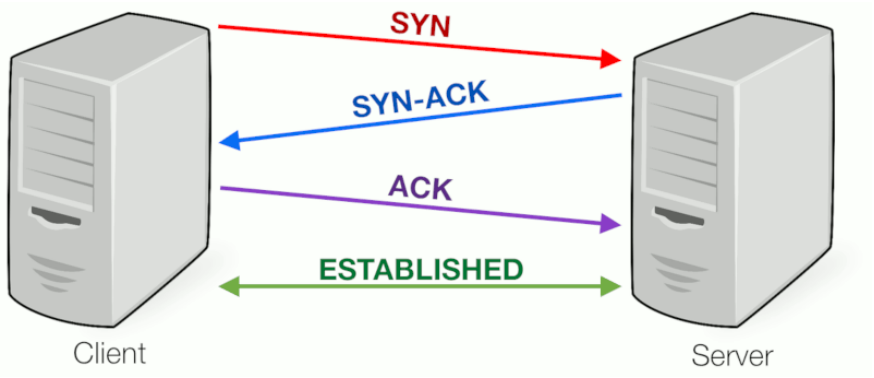
b) Hva brukes for å ta imot en epost?

Det er spørsmålet hvilken protokoll du bruker med din mailtjeneste, her er det enten POP3, IMAP eller HTTPS (Gmail etc.).



c) For å sende en epost trengs en TCP-kobling å etableres, hva inneholder en TCP-kobling mellom en klient og tjenere?

TCP er en kommunikasjon mellom en klient og tjener, her er det SYN (synkronisering), SYN + ACK (synkronisering + Accept) og ACK (Accept) som trengs for å etablere en TCP-kobling, også kallet for handshake.



d) Hvilken port brukes for TCP-kobling for å sende epost?

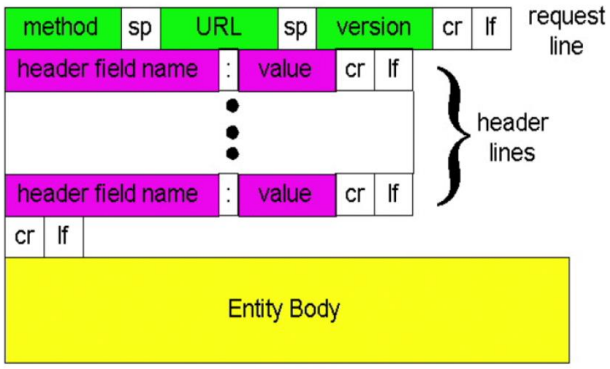
På port 25.

4. HTTP

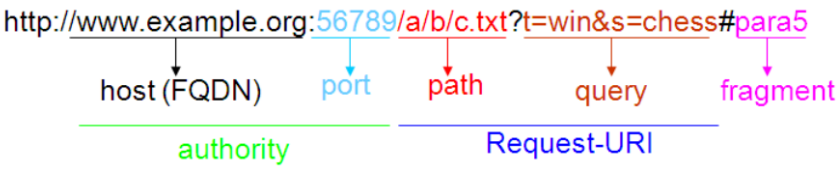
a) Sant/usant: HTTP er en filoverførningsprotokoll.

Sant, det er en enkel filoverførningsprotokoll.

b) Tegn opp HTTP-headern med penn og papir, og bytt ut spørsmåltegnene med riktig beskrivelse.



c) Hvilke deler finnes i en http URL?

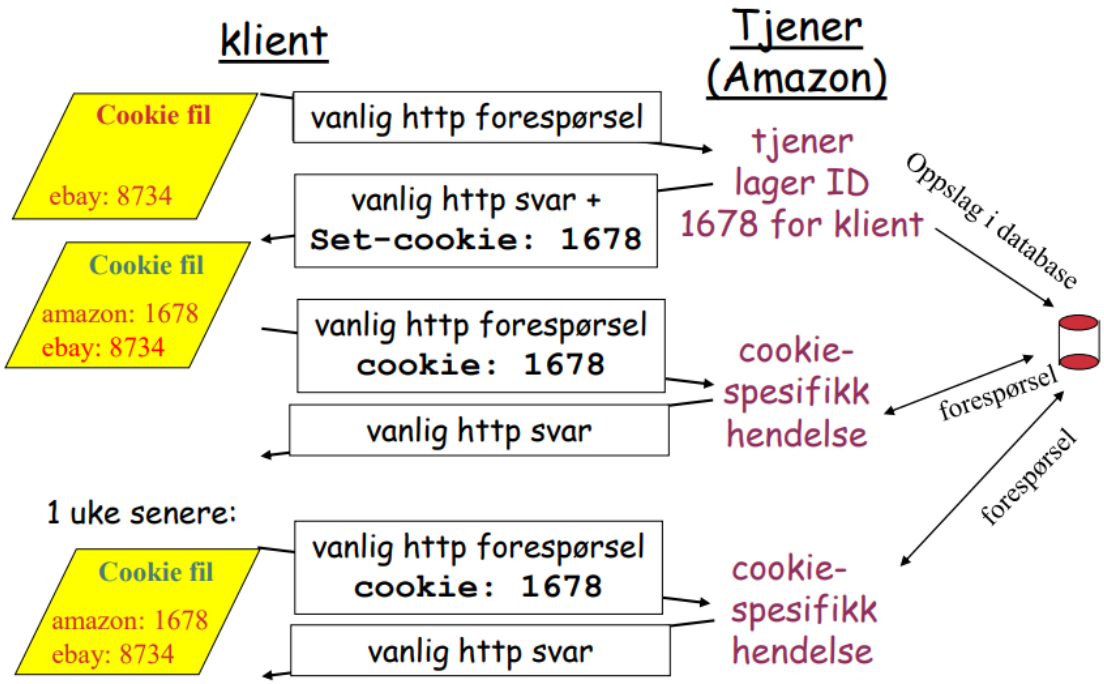


d) Hva er forskjellene mellom http 1.0 og http 1.1?

1.1 har flere kommando enn 1.0 og har da flere muligheter.

e) Hva er en «Cookie»?

Når man snakker om «cookie» er det hovedsakelig databasen, som ligger hos tjeneren, man refererer til. Det finns også en så kallet «cookie-fil», som oftest ligger hos klienten, denne inneholder cookies for alle de ulike nettsidene klienten har godkjent cookies for. Cookies gjør forbindelsene mellom klient og tjener lettere tilgjengelig (ved for eksempel et http-request) da tjener vet hvem klienten er og tvertom. Illustrasjonen under viser hvordan det er mulig å beholde denne tilstanden med hjelp av cookie. Det er altså dette som gjør at en nettside vet vem du er, hva du har lagt i handlekurven eller hvilke andre nettsider du har vært inne på og kan derfor gi deg reklame som er rettet mot deg.



f) Hva utvikledes for å få http mer sikkert?

HTTPS, som er HTTP sammen med SSL/TLS protokollen.

5. FTP

a) Sant/usant: FTP er ikke en protokoll.

Usant, FTP er en protokoll og står for File Transfer Protocol.

b) Hva brukes FTP til? Og hvilken port brukes med FTP?

FTP brukes for å overføre filer mellom to parter, den bruker klient/tjener modellen for å gjøre overføringen og gir en rask overføring. FTP bruker to porter når den skal overføre data, her er det port 21 som brukes for kommandoer og svar, port 20 brukes for selve overføringen av data. For at klient skal få kontakt med tjener brukes TCP, da på port 21.

c) Hva brukes disse for i FTP?

125, 331, 425, 252

Disse er FTP returkoder:

- 125 Data connection already open; transfer starting
- 331 Username OK, password required
- 425 Can't open data connection
- 452 Error writing file

d) Er det stor forskjell på FTP og SFTP?

SFTP er altså ikke en ny versjon av FTP men faktisk et helt nytt protokoll og den bruker oftest SSH for å opprette en kryptert forbindelse til en SFTP-server, SFTP bruker som standard port 22.

TK1100
FORELESNING 0x08 TRANSPORTLAGET

1. Transportlaget

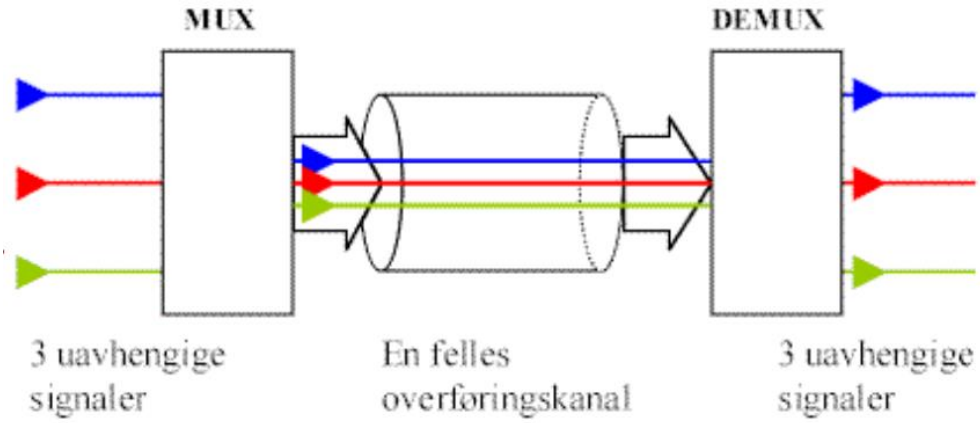
a) Kort forklart, hva er hensikten med transportlaget?

Håndtering av transport for applikasjonsmeldinger mellom klient og tjener til en applikasjon, overføringsenheten er et segment. Det er altså protokollen i transportlaget som definerer forbindelser til de individuelle portene, når en skal sende/levere en pakke. Protokollene fungerer på toppen av IP-protokollene. IP-protokollene er i nettverkslaget og dette er tema for neste forelesning.

b) Hvilke typer protokoller brukes i transportlaget?

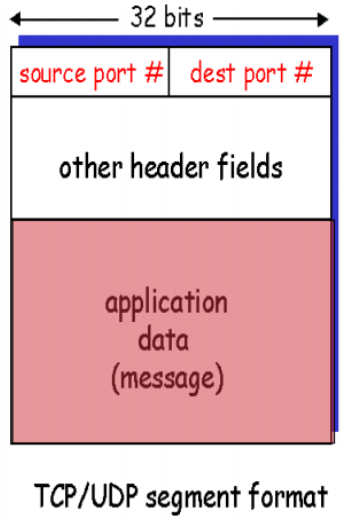
TCP og UDP.

c) Forklar med tegning hvordan multipleksing og demultipleksing fungerer.



d) Hva er et portnummer, og hvilken fordel gir det oss når vi sender data fra en maskin til en annen?

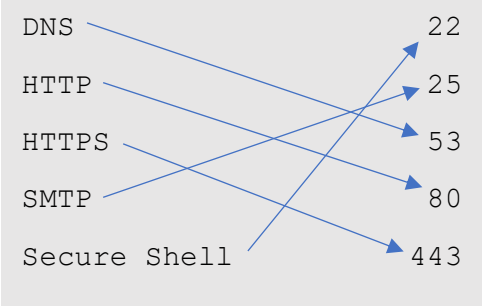
Segmentet til TCP eller UDP protokoller inneholder portnummer til sender og mottaker (som vist på bildet under).
Et portnummer er som en type ID, som brukes for at maskinen som får tilsendt data, skal vite hvilken applikasjon som skal motta den. Prosessene på maskinen må altså si hvilken port de vil forvente svar fra, *dest port.* Man kan også ved hjelp av *source port*, finne ut av hvor på senderens maskin dataen ble sendt fra.
Portnummeret hjelper oss altså å vite hvor på *maskinene* dataen blir sendt til og fra.



e) Hva menes med en «trafikk-kork»?

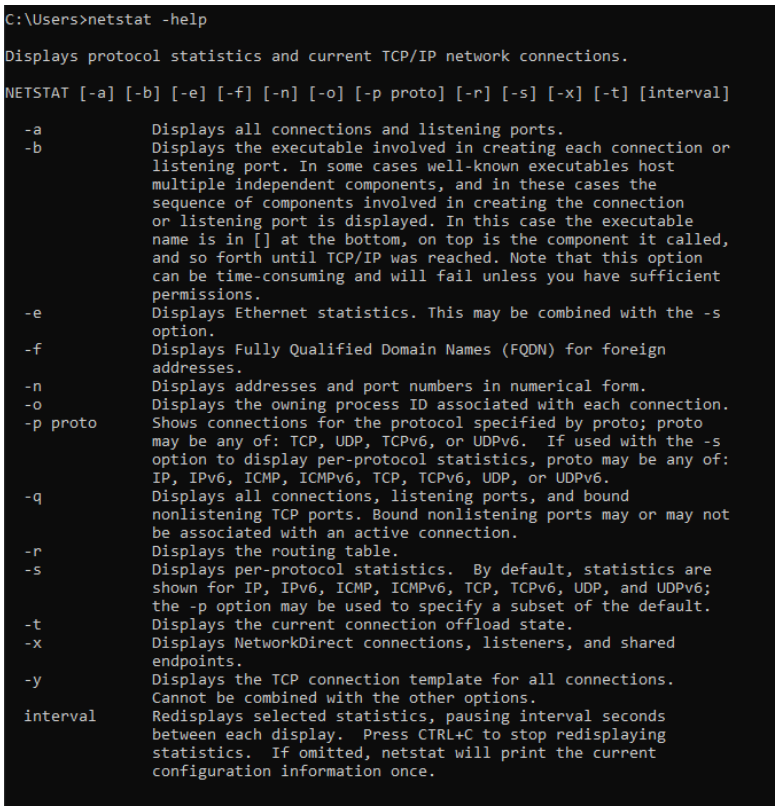
Dette skjer når det er for mange kilder som sender for mye data for fort til at de som skal transportere dataen ikke klarer å håndtere det. Disse trafikk-korkene skjer oftest i routerne som skaper nettverket dataen skal transporteres på. Når en trafikk-kork oppstår kan dette resultere i at pakker blir tapte eller at det blir lange forsinkelser fordi det er kø.

f) Sett samen protokoll med riktig port.



g) Hvilken type informasjon får du frem med å kjøre disse kommandoene i terminalen:

- `netstat -help`
  `(Se bilde under)`



- `netstat -a`
  `Displays all connections and listening ports.`

- `netstat -n`
  `Displays addresses and port numbers in numerical form`

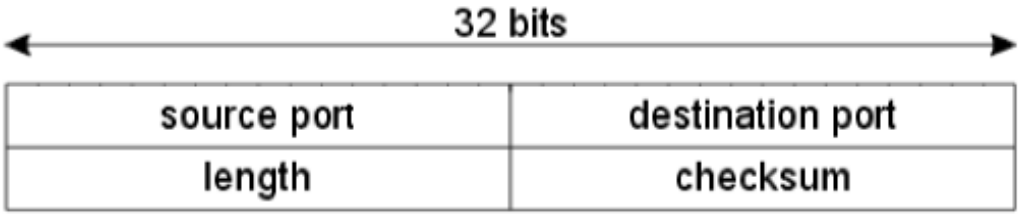
2. UDP

a) Skriv **150-250** ord om UDP, hva er det? Hva brukes det til? Hva er bra/dårlig med UDP? Hvordan oppdager man feil i UDP protokollen?

Punkter du burde ha vært innom:

- Tap av segmenter
- Rekkefølge av segmenter
- Handshake/etablering av forbindelse
- Ingen forsinkelse på grunn av ingen forbindelse
- Segment headern er liten
- Ingen kontroll av trafikk-kork
- Protokollen gir mulighet for bruk av broadcasting
- Feil-håndtering kan tas hånd om av mottakerens applikasjon
- UDP sin sjekksum og hvordan den fungerer

b) Fyll i UDP headern



c) Om vi regner på en UDP sjekksum, ville da disse pakkene inneholde feil? Hint: se slideserie for å finne ut av hvordan du skal regne dette, husk «wrap around»

Pakke 1 – utregning

Addere de første 16-bit dataene

```
  1010 0110 1010 1101
+ 1101 0010 0011 1010
1
  0111 1000 1110 0111
```

Wrap around gir svaret:
0111 1000 1110 1000

Ta svaret fra forrige og addere med neste 16-bit data

```
  0111 1000 1110 1000
+ 1000 1011 0100 1001
1
  0000 0100 0011 0001
```

Wrap around gir svaret:
0000 0100 0011 0010

Ta svaret fra forrige og sjekk det mot sjekksummen

```
  0000 0100 0011 0010
+ 1101 1011 1100 1011
  1101 1111 1111 1101
```

Svaret ovenfor er ikke kun 1-ere og derfor inneholder denne pakken en feil

Pakke 2 – utregning

Addere de første 16-bit dataene

```
  1010 1110 1010 1001
+ 0010 1011 1001 0101
  1101 1010 0011 1110
```

Ta svaret fra forrige og addere med neste 16-bit data

```
  1101 1010 0011 1110
+ 1100 1010 0110 0101
1
  1010 0100 1010 0011
```

Wrap around gir svaret:
1010 0100 1010 0100

Ta svaret fra forrige og sjekk det mot sjekksummen

```
  1010 0100 1010 0100
+ 0101 1011 0101 1011
  1111 1111 1111 1111
```

Svaret ovenfor er kun 1-ere og derfor er denne pakke riktig

3. TCP

a) Hvorfor brukes det mer TCP enn UDP i dag?

Med UDP som protokoll kan segmenter gå tapt eller leveres i feil rekkefølge. Det utføres heller ingen sikker forbindelse mellom avsender og mottaker gjennom for eksempel en handshake. Med andre ord er TCP generelt mer sikkert med overføringen av dataen og du kan være sikker på at all data kommer frem til mottaker. TCP er også en protokoll som gir mulighet for at både sender og mottaker kan sende data samtidig. Dette kalles for «full duplex».

Likevel må det nevnes at UDP har veldig rask overføring, og dermed egner seg bra til applikasjoner som vil være raske, ikke skades ved å miste litt data. Eks: Enkelte spill og videostreaming

b) På hvilken måte blir TCP pålitelig?

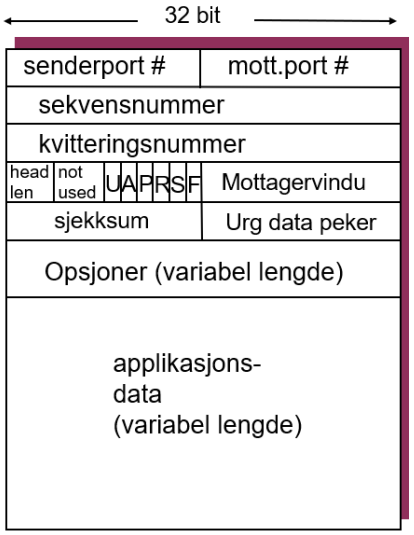
Fordi TCP bruker seg av punkt-til-punkt transformering av data. Her er det altså kun en avsender og en mottaker. For at avsender og mottaker skal kunne overføre data mellom hverandre, etablerer de en kobling gjennom en «handshake». Denne handshaken gjør at de får en sikker oppkobling helt til overføringen av dataen er ferdig. Men kanskje det viktigste med TCPs pålitelighet, er at til forskjell fra UDP, har TCP feil-håndtering, slik at applikasjonen selv slipper å ta hånd om dette.

c) Hvordan håndterer TCP problem som oppstår?

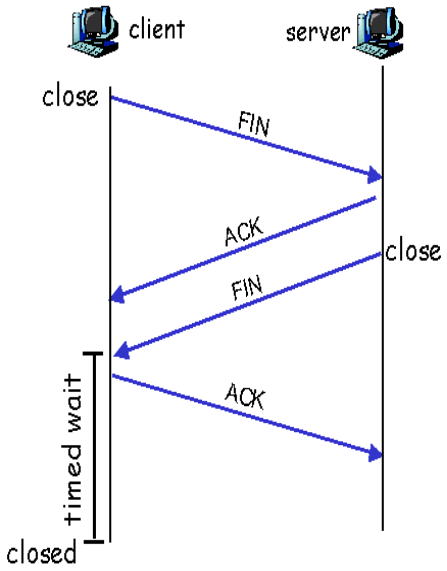
Ved hjelp av sjekksum, kvittering, sekvensnummer og timer. Under «For valgfritt egenstudier» i forelesningen er det beskrevet i detalj hvordan TCP bruker disse for å håndtere feilen som kan oppstå ved transport av data.

d) Er TCP sin header større eller mindre enn UDP sin header?

TCP sin header er større, for å få plass til alt av ekstra data for feilhåndtering. For eksempel har TCP headern et 32-bit kvitteringsnummer som UDP ikke har.



e) Tegn opp en nedkobling av en TCP-forbindelse.



f) Hva menes med TCP og rettferdighet?

Om du har to maskiner på samme nettverk og de skal bruke samme router for å transportere data ved hjelp av TCP, er det meningen at dataraten skal bli rettferdig. Med andre ord, skal de begge få lov til å bruke ressursene like mye. Det finnes forskjellige måter å regne denne rettferdigheten på, men det vanligste er å si: datarate / antallet TCP-sesjoner **eller** R/K

TK1100
FORELESNING 0x09 NETTVERKSLAGET

1. Nettverkslaget

a) Forklar kort hva hovedoppgaven til nettverkslaget er.

Nettverkslaget inneholder protokoller som gjør det mulig å overføre pakker fra avsender til mottaker via en rute i nettverket. Her brukes nettverksprotokollen hos avsender og mottaker, men også på hver mellomlanding i nettverkstjeneren. Det er nettverkslaget som «finner veien» som pakken skal transporteres via. Her er det også mulig å definere denne ruten før pakken sendes.

b) Test å skriv in `netstat -r` eller `route print` i terminalen på din maskin, hva får du for informasjon?

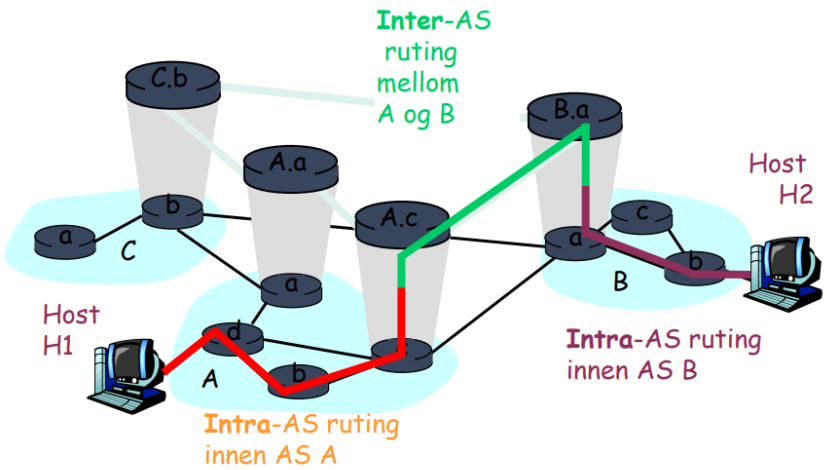
Interface list, IPv4/IPv6 Route table med aktive ruter, og eventuelle Persistent Routes.

c) Hva menes med «Routing av datagram»?

Routing er det som gjøres for å finne ut hvilken vei datagrammet skal bevege seg over nettverket, fra avsender til mottaker. Dette bestemmes før datagrammet sendes ut.

d) Hva er forskjellen mellom Intra-AS og Inter-AS ruting?

Intra-AS er for ruting lokalt på et nettverk og Inter-AS er for ruting mellom to nettverk, Intra-AS inneholder altså interne gateways/routere og Inter-AS inneholder eksterne gateways/routere.



e) Hva er adresse til localhost?

127.0.0.1

2. IPv4

a) Hvor lang er en IPv4 adresse og hvorfor kalles det for en adresse?

En IPv4 adresse er 32-bit lang og fungerer som en slags id på internett for maskiner. Adressen sier også hvilket nettverk maskinen er koblet til. Man kan tenke på det som en gateadresse, som posten bruker for å finne frem til riktig gate, og til slutt det spesifikke huset du sitter i.

b) Skriv in `ipconfig` (Windows) eller `ifconfig` (OSX/Linux) i terminalen, hvilken informasjon får du frem?

Ip adresse, subnet maske og defualt gateway for hver adpater koblet til TCP/IP.

c) Hva er din «Default Gateway» på din maskin akkurat nå?

Bruk ipconfig eller ifcongif i terminalen for å finne ut av dette. Dette vises også på slideserie fra dagens forelesning.

d) Hvordan får du tildelt din IPv4 adresse i et Local Area Network?

Det er mulig å sette en IPv4 adresse manuelt og få denne til å bli statisk i et LAN, men det som er vanligst er å ta i bruk DHCP (Dynamic Host Configuration Protocol) for å tildele IP-adresser i et LAN.

e) Forklar kort hvordan et datagram transporteres fra avsender til mottaker, de er på ulike nettverk.

1. Avsender finner nettverksadresse til mottaker
2. Avsender finner ut om mottaker er på samme nettverk eller ikke, for å se om de er direkte forbundet
3. Ruting-tabellen gir beskjed om hvor neste stop/hopp er på veien for avsender for å nå frem til mottaker
4. Link blir laget til de nødvendige stoppene/hoppene
5. Datagram sendes gjennom nettverket av stopp/hopp ved hjelp av linker og til den kommer frem til mottakers gateway og til slutt mottaker

f) Hva trengs for å finne nettverksprefixen?

Nettmaske og IP-adresse, med hjelp av disse er det mulig å finne ut av om to maskiner er på samme nettverk eller ikke.

g) Hva er en Nettmaske og hvorfor brukes den?

Det er nettmasken som bestemmer hvilken del av IP-adressen som er prefix og hvilken del som er host, det er altså på denne måten man finner ut av om maskiner er på samme IP-nettverk og om de har mulighet å sende data direkte mellom hverandre eller om de må utenfor nettverket.

h) Hvilken operasjon brukes for å regne ut adresse med nettmaske?

AND operasjonen

i) Kan 10.21.3.5 med nettmaske 255.255.254.0 sende direkte til 10.21.2.255 med nettmaske 255.255.254.0? Hint: Se slideserien fra forelesning for hvordan du skal regne ut dette.

Regn først ut hvilket nettverk de ulike maskinene tilhører med hjelp av nettmaske og AND operasjonen

```
10.21.3.5
255.255.254.0

    0000 1010 . 0001 0101 . 0000 0011 . 0000 0101
AND 1111 1111 . 1111 1111 . 1111 1110 . 0000 0000
    0000 1010 . 0001 0101 . 0000 0010 . 0000 0000
```

Konverter svaret til desimal og regn hvor mange bit som er prefix.

Det lokale nettverket er: 10.21.2.0/23

Når du siden gjør det samme med 10.21.2.255 og nettmaske 255.255.254.0 kommer du merke at du får samme svar hvilket betyr at de er i samme nettverk og kan sende direkte til hverandre.

j) Kan 10.21.3.5 med nettmaske 255.255.255.0 sende direkte til 10.21.2.255 med nettmaske 255.255.255.0? Hint: Se slideserien fra forelesning for hvordan du skal regne ut dette.

Regn først ut hvilket nettverk de ulike maskinene tilhører med hjelp av nettmaske og AND operasjonen

```
10.21.3.5
255.255.255.0

    0000 1010 . 0001 0101 . 0000 0011 . 0000 0101
AND 1111 1111 . 1111 1111 . 1111 1111 . 0000 0000
    0000 1010 . 0001 0101 . 0000 0011 . 0000 0000
```

Konverter svaret til desimal og regn hvor mange bit som er prefix.

Det lokale nettverket er: 10.21.3.0/24

Når du siden gjør det samme med 10.21.2.255 og nettmaske 255.255.255.0 kommer du merke at du ikke får samme svar, denne maskinen tilhører nettverket 10.21.2.0/23.

k) Hva menes med private adresser?

En privat IP-adresse er din adresse du har på det nettverket du er oppkoblet til og det er ikke den adressen som blir sendt ut til internett, her er det Internett-routere som automatisk dropper den private IP-adresse og gir pakken en IP-adresse som alle på ditt nettverk deler. At ta i bruk private IP-adresser gir en fleksibilitet for kommunikasjon internt i nettverk.

l) Hva er et subnet?

Subnet er et nettverk av maskiner som har samme prefix i sin IP-adresse og kan derfor kommunisere med hverandre uten å gå via en router. Ref. til oppgave i) og j) i dette oppgavesett, der fant vi ut av ved hjelp av nettmaske at maskinene hadde mulighet å kommunisere direkte med hverandre og da også om de var i samme subnet.

3. ICMP

a) Hvem er det som bruker ICMP?

Host, router og gateway, ICMP er da altså en protokoll som håndterer feil-rapportering.

b) Hva er en host, router og en gateway?

Host: En pc eller maskin som er koblet opp til internett.

Router: Dette er en enhet/tjeneste som har som funksjon å route IP-pakker mellom nettverk.

Gateway: Dette er en router som gir tilgang for IP-pakker å komme inn/ut mellom det lokale nettverket og resterende internett.

c) Hvorfor utnytter `tracert/traceroute` ICMP protokollen?

For at da er det enkelt å se hvor i ruten dataen skal transporteres kan ha forsinkelser eller problem, hvilket gjør at det er enklere å feil-søke.

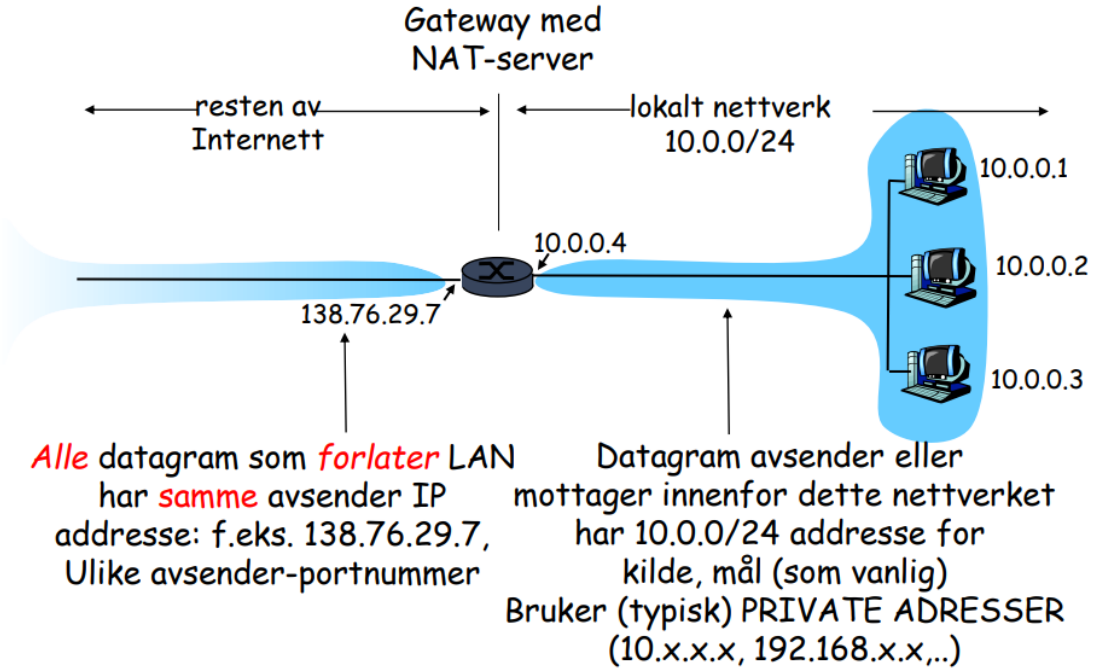
d) Test å «pinge» vg.no, hvilken informasjon får du av det?

```
C:\Users>ping vg.no

Pinging vg.no [195.88.55.16] with 32 bytes of data:
Reply from 195.88.55.16: bytes=32 time=5ms TTL=250
Reply from 195.88.55.16: bytes=32 time=4ms TTL=250
Reply from 195.88.55.16: bytes=32 time=3ms TTL=250
Reply from 195.88.55.16: bytes=32 time=4ms TTL=250

Ping statistics for 195.88.55.16:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 3ms, Maximum = 5ms, Average = 4ms
```

4. NAT

a) Hva er problemet/problemene som NAT løser?

Imellom det store internettet og det lokale nettverket sitter det en gateway. Tenk på det som en dør inn til huset (området med datamaskiner på samme nettverk). Når en pakke blir sendt over internett, og til slutt kommer frem til mottakerens gateway, må vi vite hvilke av maskinene på det lokale nettet som er den *faktiske* mottakeren.
(Eks: *En* datamaskin har åpnet vg.no. Da ville det vært dumt om alle datamaskiner i samme hus fikk masse VG-reklamer helt plutselig.)
Dette er her NAT kommer inn i bildet. NAT serveren ligger i gateway-en, og ved hjelp av en tabell holder den styr på hva som er source- og destinasjonsadressene på internettsiden og LAN-siden. I den tabellen kan det for eksempel stå: «Mia sin pc gikk inn på vg.no, og venter nå på svar fra VG», men formulert med private ip-adresser.



b) Hva er spesielt med 192.168.x.x og 10.x.x.x adresser?

Dette er starten på typiske private adresser, og disse brukes da i et lokalt nettverk. Med andre ord er det mest sannsynlig at din private adresse på nettverket du er koblet til nå starter på 192.168 eller 10, bruk ipconfig eller ifconfig i treminalen for å finne ut av dette.

c) Sant/Usant: Du og din venn er koblet på samme LAN og sender datagram ut til internettet, dere har da ulike IP-adresser for å kunne identifiser seg.

Usant, hvis dere bruker samme gateway for å sende ut data, vil den gi dere samme IP-adresse. NAT finner ut av hvem av dere svar-pakkene skal til, så dere trenger ikke være skillbare på internettet.

d) Forklar kort hva traversering-problemet med NAT betyr.

Traversering-problemet handler om at en avsender prøver å nå en mottaker inne på et lokalt nettverk, men har kun IP-adresse til den maskinen og vet da ikke at den skal gjennom NAT-en x.x.x.x, da det er IP-adressen til NAT-et som er synlig for avsenderen. Det finnes tre ulike løsninger til dette problemet, enten statisk konfigurering av NAT, bruk av UPnP og IGD, eller bruk av relaying.

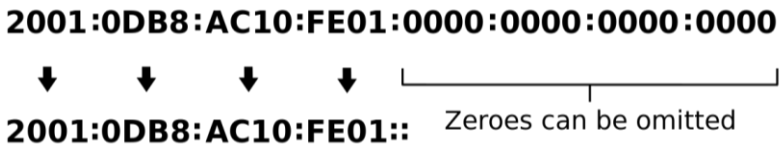
5. IPv6

a) Hva er de største forskjellene mellom IPv4 og IPv6?

IPv4	IPv6
32 bit Header inkluderer sjekksum Opsjoner i header opp til 40 bytes DNS A-records	128 bit Header inkluderer ikke sjekksum Utvidet header for opsjoner DNS AAAA-records

b) Hvordan ser IPv6 notasjonen ut?

Notasjonen er på totalt 128 bit og er skrevet i heksadesimalt i 8 grupper på 2 byte/ 16 bit. Om det er kun 0 i en gruppe så kan hele adressen forkortes gjennom å skrive to kolon (::).



c) Forklar kort hva «Unicast», «Anycast» og «Multicast» er.

Unicast: Brukes for å nå en enkelt IPv6 adresse

Anycast: Brukes for å nå «hvem som helst», men sender til den første som tar imot

Multicast: Brukes for å sende til en gruppe, dette er IPv6 sin versjon av broadcast.

d) Hva er forskjellen på Dual stack og Tunneling, og hva brukes det til?

Det brukes for å IPv4 og IPv6 skal kunne samarbeide og det skal være mulig å bruke begge, ved Dual stack forstår routeren begge typer av adresser og oversetter, men ved Tunneling pakkes IPv6 adresser inn i IPv4 så at det er mulig å lese adressen.

TK1100
FORELESNING 0x0A LINKLAGET

1. Linklaget

a) Hva er linklaget sitt mål og ansvar?

Linklaget har som ansvar å transportere data i en ramme (frame) fra node til node. Dette sendes over en link mellom for eksempel en switch eller router. Målet er å finne eventuelle feil, men også å rette opp de mulige feilene.

b) Hva er overføringsenheten for linklaget?

Ramme eller frame. Rammene innkapsler datagrammer. Husk at *datagram* er overføringsenheten fra Nettverkslaget. Det er dette datagrammet linklaget legger i en ramme for å håndtere det.

c) Hva er det linklaget skal transportere?

Datagram fra nettverkslaget.

d) Forklar kort disse bergreppene i sammenheng med linklaget:

Datagram:
Dataen som skal transporteres i linklaget.

Kommunikasjonslink:
Dette er transportetappen. Det kan finnes flere etapper for reisen til et datagram. Eksempel er fra en router til en annen. Denne etappen er en kommunikasjonslink mellom dem.

Linklagsprotokoll:
For å transportere dataen over kommunikasjonslinken trenger vi å vite hvordan den skal transporteres. Hvilken protokoll er det som skal brukes for å forflytte data på denne etappen?

Routing-algoritme:
Algoritmen brukes for å kalkulere hele reisen dataen skal bevege seg over linklaget, *før* dataen sendes ut. Her tar den i beregning hvilke noder den skal *routes* gjennom, før den sendes, og det kan ofte være flere veier som leder til samme sted. Da vil den finne ut den beste veien basert på ulike kriterier. Du kan sammenlikne det med et reisebyrå, som velger en reiserute for deg når du skal på ferie, før du faktisk setter deg på flyet.

e) Kort forklart, hva menes med «Framing»?

«Framing» eller *omramming* er prosessen for å legge datagram inn i rammer. Det legges også til header og trailer (les om forskjellen mellom header og trailer her: https://wikidiff.com/header/trailer).

2. MAC

a) Hvilke to hovedtyper av linker finnes det?

Punkt-til-punkt: fra en router til en annen.
Kringkasting: informasjon som deles sånn at fler enn en kan ta del av informasjonen eller lytte på. *(Det heter NRK, Norsk Riks-Kringkasting, fordi ALLE over hele landet kan se og lytte til det de sender ut)*

b) Forklart kort om hvordan Multippel aksess-protokoller fungerer.

MA (Multippel aksess-protokoll) bruker kringkasting, det betyr at det er en kanal som er delt med mange, der flere kan sende og lytte til. MA er ansvarlig for å avgjøre når en node kan sende data over kanalen. Denne kommunikasjonen ut til nodene skjer også via kanalen.

c) Hvilke tre ulike klassifikasjoner finnes det for kanaldeling?

- Kanalpartisjonering
- Random Access
- «Etter tur»

d) Hva er det en random access MAC protokoll gjør? Gi også noen eksempel på slik protokoller.

En random access MAC protokoll har ansvar for kollisjoner i linklaget. Det er disse protokollenes ansvar å dekke opp kollisjonene, men også håndtere dem når de skulle oppstå. Eksempel på random access MAC protokoll er: ALOHA, CSMA, CSMA/CD og CSMA/CA.

e) Hva gjør CSMA om kanalen den skal sende data gjennom er ledig, og hva gjør protokoller hvis kanalen er opptatt?

CSMA, Carrier Sense Multiple Access, som er en random access MAC protokoll, *lytter* før den skal sende data og hvis det er ledig på kanalen, *sender* den rammen via linken. Hvis kanalen er opptatt, avventer den og utsetter overføringen.

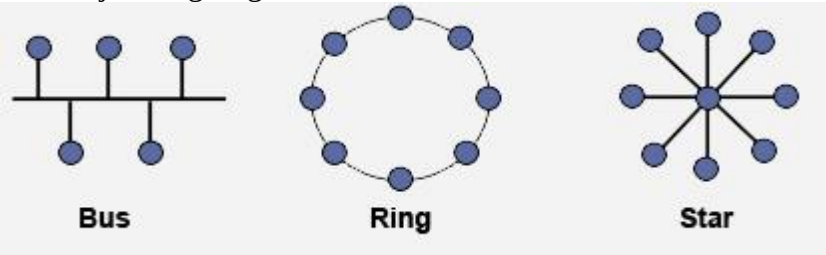
Som nevnt i forelesning, den menneskelige analogien på dette er: **før man begynner å snakke, hører man etter om andre snakker – ikke avbryt!**

f) Hva er forskjellen mellom CSMA og CSMA/CD?

Begge er random access MAC protokoller, men CSMA/CD har en såkalt *Collision Detection*. Her er den menneskelige analogien: **den høflige samtalepartneren.** Denne protokollen lytter samtidig som den sender data oven kanalen og ved kollisjon avbrytes sendingen av data med en gang.

3. LAN

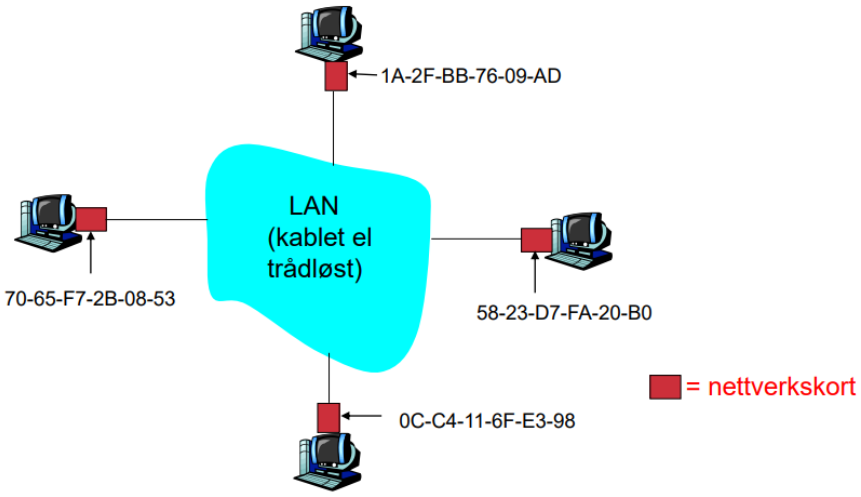
a) Hvilke ulike topologier finns i et lokalt nettverk?
Buss, stjerne og ring.



b) Hva brukes en MAC-adresse til?

MAC-adresser brukes for å overføre rammer i linklaget mellom to enheter på samme nettverk. Dette er altså adressen som brukes for å vite *hvor* dataen skal på linklaget. På nettverkslaget (som var tema for forrige forelesning) snakket vi om IP-adresse og dette er adressen som brukes for at dataen skal finne frem på nettverkslaget.

Du kan se på en MAC-adresse som den fysiske adressen til maskinen, denne adressen er også brent i nettverkskortets ROM.
Der IP-adressene brukes for å nå maskiner over det store internettet, brukes MAC-adressene over det lokale nettet, LAN (Local Area Network).



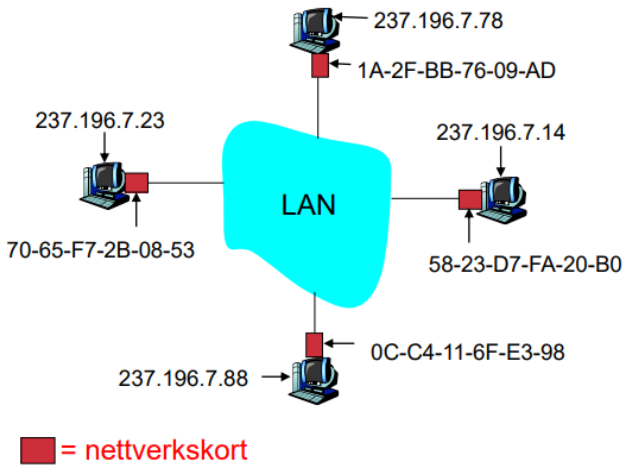
c) Finn ut av din MAC-adresse på din maskin.

Bruk denne linken for å finne ut av din MAC-adresse (ulike måter for ulike operativsystem): https://its.uiowa.edu/support/article/1618

d) Hvordan finner en maskin eller router på et LAN ut hvilken MAC-adresse som hører til hvilken IP-adresse?

Gjennom å benytte seg av protokollen ARP, Adress Resolution Protocol. I ett nettverk har hver maskin/router en ARP-tabell som inneholder en oversikt over hvilken IP-adresse som tilhører hvilken MAC-adresse. På denne måten kan maskinen som skal sende data koble sammen IP-adressen fra nettverkslaget og MAC-adressen fra linklaget og derfor enkelt finner frem hvor dataen skal sendes.

Om avsender ikke vet om MAC-adressen den skal sende til, altså at det ikke er lagret i sin lokale ARP-tabell, sender avsender ut en ARP forespørsel ut på LAN-et med hjelp av kringkasting. Denne forespørselen inneholder IP-adressen til mottaker og når mottaker med riktig IP-adresse får forespørselen, svarer han avsenderen med sin MAC-adresse. Nå kan avsender lagre MAC-adressen sammen med IP-adressen i sin ARP-tabell.



e) Ved hjelp av terminalen på din maskin, finn ut av hvordan ARP-cachen ser ut. Hint: `arp -help`

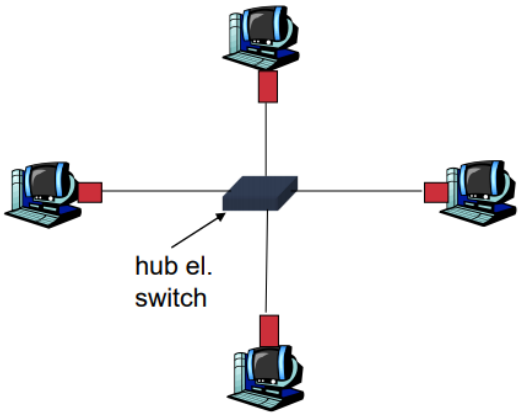
Eksempel på output:

```
C:\Users>arp -a

Interface: 192.168.228.1 --- 0xa
  Internet Address      Physical Address      Type
  192.168.228.255       ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
```

4. Ethernet

a) Hvilke type topologi bruker Ethernet og hva er det som har blitt brukt tidligere?

Tidligere var det busstopologi som var brukt, men nå er det stjernetopologi som er «de facto», midten av stjerna består da av en hub eller en switch.



b) Hvilken hardware brukes for å transportere data i et Ethernet?

Hub eller switch.

c) Hvorfor er Ethernet en upålitelig tjeneste?

Fordi den ikke bruker «hand shake» når det skal sendes data. Derfor er det mulig at det oppstår gap i strømmen med datatrafikk. Ethernet bruker seg også av CSMA/CD som er protokollen for å håndtere kollisjoner på kanalen.

5. HUB og SWITCH

a) Hva er forskjellen mellom en HUB og en SWITCH?

Det er flere forskjeller mellom en HUB og en SWITCH, men de viktigste handler om hvordan de sender ut data til maskinene via portene, og at HUB-en er på det fysiske laget, men SWITCH-en er på linklaget. Ved bruk av en HUB for å sende en ramme, sendes dette ved hjelp av en «broadcast» ut til alle porter. Det betyr at alle som er koblet til HUB-en får denne informasjonen. Dette skjer også om den rammen kun skulle til en port og ikke alle. En SWITCH har mulighet til å lagre MAC-adresser og det betyr at en SWITCH har mulighet til å sende data kun til den/de som skal ha den.

b) Hva brukes for å finne eventuelle kollisjoner på en HUB og hva brukes på en SWITCH?

HUB: bruker seg av NIC og har kun en kollisjonsdomene.
SWITCH: bruker seg av CSMA/CD og har en kollisjonsdomene per port.

c) Hva er det en SWITCH sjekker på en ethernetramme?

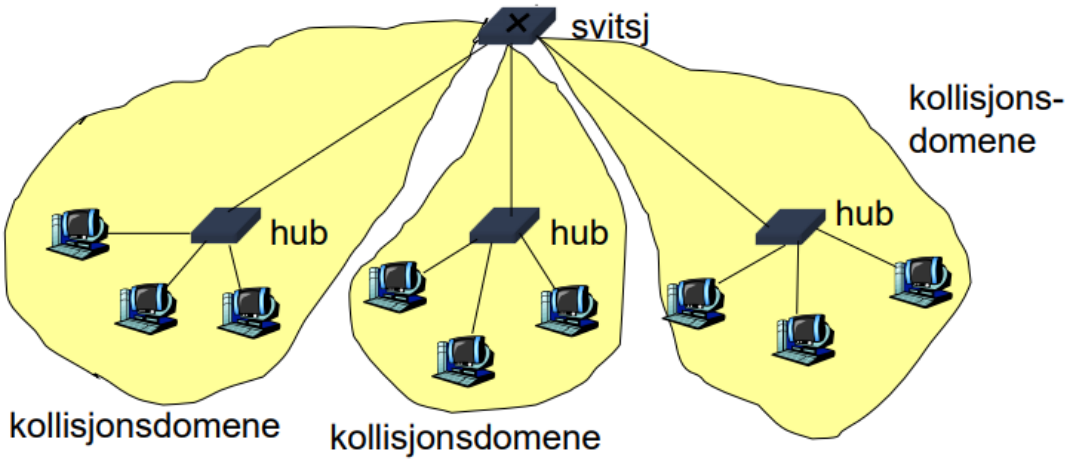
Når en ramme kommer frem til en SWTICH ser den på rammeheader-en for å finne ut av hvor rammen skal sendes videre til, og når det er behov for å sende rammen videre gjør SWITCH-en det gjennom å finne riktig MAC-adresse i nettverket.

d) Hva betyr at en SWTICH er selvlærende og hva gir dette for fordeler?

Det betyr at den husker og lagrer nødvendig informasjon fra sendinger av rammer. Her kan SWTICH-en huske hvilket segment rammen kommer fra og samtidig vite hvilken MAC-adresse den kom fra. Denne informasjonen lagres i en egen tabell som gjør det enklere og kjappere neste gang data skal overføres.

e) Hva menes med et kollisjonsdomene?

En SWITCH kan dele inn nettverket i segmenter for å strukturere nettverket. Hvert segment kalles for et kollisjonsdomene. På denne måten vil kollisjoner bare oppstå innad i et kollisjonsdomene, og ikke på tvers. Dette fører til at mindre data forstyrrer hverandre, da det er færre maskiner i samme segment.



f) Forklar hva forskjellen mellom en SWITCH og en ROUTER er.

Routere er for nettverkslaget og ser på nettverkslagsheadere, mens switchen er for linklaget og ser da på linkagsheadere. Begge bruker seg av tabeller, men ulike typer tabeller. En router bruker seg av routingtabeller og implementerer algoritmer for å finne fram riktig rute, mens en switch bruker seg av switchtabell og har selvlæring.

6. Wi-Fi

a) Hva er forskjellen mellom WLAN og LAN?

WLAN = Wireless Local Area Network
LAN = Local Area Network

Forskjellen handler om nettverket er trådløst eller kablet. WLAN er altså det vi kjenner som Wi-Fi.

b) Hva bruker WLAN for å finne kollisjoner?

WLAN bruker seg av CSMA/CA, her er da CSMA/CD ikke godt nok og man bruker /CA istedenfor. Når dataen blir sent blir det aktivert en timer, om avsender får en bekreftelse tilbake fra mottaker før tiden har gått ut så anses leveransen som vellykket, hvis ikke startes algoritmen på nytt.

c) Hva er IEEE 802.11?

Dette er en samling av ulike standarder som gjelder for WLAN, og de er tatt fram av organisasjonen IEEE. Det er altså disse standardene som brukes får å bygge opp et WLAN.

d) Skriv kort, utefra dine egne tanker, hva er usikkert/sikkert med WLAN?