IN1020 Sammendrag

Datasikkerhet 2

1. **Informasjonssikkerhet** **2**
2. **Sikkerhetsmål** **4**
3. **Sikkerhetstiltak** **9**
4. **Sikkerhetstrusler** **10**
5. **Kryptering og nøkler** **15**
6. **Personvern** **16**
7. **Prosess** **17**

Nettverksdelen 19

1. **Nettverkskomponenter** **19**
2. **Aksessmodeller** **21**
3. **Lagene i Internett (TCP/IP referansemodellen)** **22**
4. **IP-adresser og CIDR-notasjon** **27**
5. **HTTP-protokollen** **33**
6. **Kryptering** **36**

LMC og representasjon 38

1. **Prosessoren** **38**
2. **LMC** **39**
3. **Assemblerkode** **41**
4. **Tallsystemer** **42**
5. **Bilder** **45**
6. **Lyd** **46**

Maskinvare 48

1. **Abstraksjonsnivå** **48**
2. **Porter og funksjoner** **49**
3. **Klokke og synkronisering** **52**
4. **Minnehirearki** **53**
5. **Datamaskinarkitektur** **55**

Datasikkerhet

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

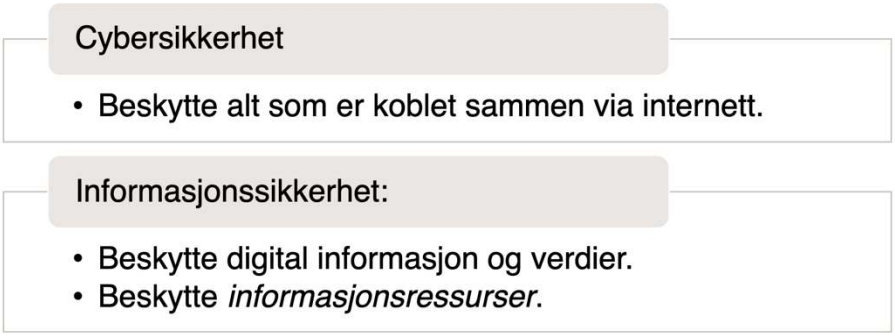
1. Informasjonssikkerhet

Hva er informasjonssikkerhet?

- Handler om å gjøre de systemene som brukes til å behandle all informasjon sikre, slik at informasjonen ikke blir synlig for uvedkommende, den ikke blir endret eller slettet av uvedkommende, og at den skal være tilgjengelig for de rette personene til rett tid.
- Informasjonssikkerhet er en kontinuerlig prosess for å oppdage og hindre trusler og å fjerne sårbarheter.

Datateknologi og sikkerhet

- Både i hjemmet og i virksomheter



Begreper:

Sikkerhet	handler om å beskytte verdier mot skade og ødeleggelse Lan defineres som en tilstand; fravær av uønskede hendelser eller frihet fra fare og frykt. Denne tilstanden er imidlertid ikke statisk, men påvirkes av endringer i faktorer som trussel og farer, sårbarhet og verdi
Verdi (vid betydning)	helse, miljø, informasjon, sensitiv informasjon, penger, gjenstander, omdømme osv. Hva som er en verdi handler om hva som er viktig for noen i et gitt tilfelle. Hva er viktig for deg? Hva er viktig for en virksomhet? Hva er viktig for samfunnet?
Skadelige hendelser (tilsiktet eller utilsiktet)	naturkatastrofer, krig, ulykker, skade påført med hensikt osv.
Informasjonssikkerhet	sikre informasjon, beskytte informasjonsressurser (IT-utstyr, infrastruktur, datafiler, programvare etc.).
Cybersikkerhet	sikre ting som er sårbare via IKT (f.eks. informasjon, selvkjørende biler, pacemaker osv.).
Personvern	handler om retten til et privatliv og retten til å bestemme over egne personopplysninger
Innebygget personvern	tar hensyn til personvern i alle utviklingsfaser av et system eller løsning. Setter en standard for hva som skal ligge til grunn for et system eller program.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Sikkerhetsmål	uavhengig av spesifikk implementering og kan ivaretas med ulikE sikkerhetstiltak
Sikkerhetstiltak	er tiltak for å oppdage, forebygge, ivareta eller gjenopprette sikkerhetsmål. De er basert på spesifikk implementering, ofte bundet til produkter. Fysiske, tekonologiske og administrative tiltak Ulike tilstander: når den lagres, under overføring og når den er i bruk (feks kryptering) Ulike tidsfaser: preventide, detekterende og korrigerende tiltak
DoS-angrep (Denial of Service-angrep)	hindrer tilgang til noe man vil ha tilgang til.
Sikkerhetsstyring	er systematiske aktiviteter for å oppnå og opprettholde et sikkerhetsnivå i overensstemmelse med de mål og krav en organisasjon har satt seg.
Risiko	sannsynlighet for angrep * kostnad eller tap i kroner
Trussel	uønsket hendelse som kan inntreffe. På datatrafikk: Avlytting: Uvedkommende lytter til nettverkstrafikk og får tak i konfidensiell informasjon. Passivt angrep. Data modifiseres (Man in the Middle-angrep, MitM): Uvedkommende griper inn i kommunikasjonen og modifiserer data. Aktivt angrep. Forfalskning: Uvedkommende sender ikke-autentiske meldinger eller later som de er en annen. Aktivt angrep. Tjenestenektangrep: Utilgjengelige ressurser. Bombadere en organisasjon slik at de ansatte ikke får tilgang til informasjon Uautorisert bruk: Misbruk av ressurser
Angrep	noen forårsaker med hensikt en uønsket hendelse.
Informasjonssikkerhet	er en kontinuerlig prosess for å oppdage og hindre trusler og å fjerne sårbarheter.
Sikkerhetsarbeid	Arbeid man gjør for ivareta sikkerheten (se def sikkerhet)
Nettverkssikkerhet	To hovedområder: Kommunikasjonssikkerhet: Beskytte data i transporten mellom virksomheter/endenoder Skallforsvar: Beskytte en virksomhets dataressurser mot uautorisert tilgang fra omverdenen
Trygg lagring	Lagring av informasjon, beskytte data mot innsyn og endring: kryptering, ta vare på data: sikkerhetskopi, kvitte seg med data og klassifisering av informasjon Åpen informasjon - for alle Begrenset informasjon - ikke for alle Konfidensiell informasjon - hemmelig og sensitiv Svart informasjon - superhemmelig
Trusselaktører	En trusselaktør er: Den som kan tenkes å utføre en potensielt skadelig handling Hvilke trusselaktører man bekymrer seg for er situasjonsbetinget.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

	Hvem er trusselaktørene? Sporadiske «hackere», Aktivister, Kriminelle, Virksomheter (fra konkurrenter, Nasjonalstater (spionasje)

2. Sikkerhetsmål

Sikkerhetsmål:

- CIA (KIT): Confidentiality, Integrity, Availability
- Stikkord som inngår i informasjonssikkerhet
- Det er 6 mål som vi har innen informasjonssikkerhet slik at vi ønsker at alle sikkerhetsaspektet av alle tjenester skal oppnå disse målene.

Konfidensialitet	Sikre at kun de med rettmessige behov har tilgang til enressurs. Tiltak: kryptering, tilgangskontroll, perimetersikring, gode policy for hvordan data skal håndteres og hvordan kommunikasjon skal foregå. Trussel: tyveri, lekkasje Konfidensialitet: • Handler om at kun de som skal ha rett til å se informasjonen gitt i en tjeneste skal ha tilgang til å se denne informasjonen • Det vil si, ingen uvedkommende skal kunne se informasjon som skal være skjult for allmenheten • **Viktig mål fordi:** Vi vil at kun de som har rett skal kunne få tak i informasjon, siden sensitiv informasjon kan fort misbrukes om den faller i feil hender. • **Sikkerhetstiltak:** kryptering, tilgangskontroll (hvem har tilgang til hva), skallsikring (feks brannmur men også fysiske tiltak).
Integritet	Sikre at en ressurs ikke endres eller slettes utilsiktet. Tiltak: tilgangskontroll, endringskontroll, kryptografisk sjekksumalgoritme, perimetersikring (f.eks. brannmur), digital signering også av programvare (code signing). Trussel: korrupsjon av data og systemer Integritet: • Informasjon skal ikke endres eller slettes av uvedkommende. • Det vil si at kun de som har rettigheter skal kunne endre og slette informasjon, og ingen andre. • Skal stole på at dataen som er tilgjengelig er korrekt. • **Viktig fordi:** Vi vil vite at den informasjonen vi har er korrekt og at ingen med onde hensikter eller noen med et uhell endrer eller sletter viktig data. • **Sikkerhetstiltak:** tilgangskontroll, endringskontroll, skallsikring, kryptering
Tilgjengelighet	**Tilgjengelighet**: Sikre at en ressurs er tilgjengelig for rett person til rett tid.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

	At informasjonen er tilgjengelig for autoriserte ved behov. Dreier seg om at informasjon til enhver tid skal være tilgjengelig for den som skal ha tilgang til informasjonen. Mobilnett, nødnett, pasientsystemer, banksystemer, betalingssystemer er viktige systemer vi i dag er forholdsvis avhengige av i hverdagen, og som dermed bør ha sterkt fokus på tilgjengelighet. Samtidig vil det for en nettbutikk føre til svikt i omsetningen hvis deres nettbutikk gjøres utilgjengelig for en periode, selv om det kanskje ikke berører oss forbrukere på samme måte (vi er ikke så veldig lojale, og velger raskt en annen butikk...). DDOS-angrep eller andre angrep som tar ned et system/en tjeneste er en trussel, men det samme er f.eks. systemfeil, feilkonfigurasjon (en ansatt har ikke fått riktige tilganger pga. manuell feilregistrering), naturkatastrofer eller strømbrudd. Tiltak: sikkerhetskopier, redundante tjenester, gode rutiner for hendelseshåndtering og gjenoppretting. Trussel: (D)DoS-angrep, systemfeiler også en trussel, og forekommer nok oftere enn DDoS-angrep **Tilgangskontroll**: Innebærer å sjekke om et subjekt har tilgang til å utføre en ønsket handling på et objekt, etter forhåndsdefinerte regler. Tilgangskontroll = autentisering (bekrefte identitet til subjektet) + kontrollere autorisasjon (bekrefte at subjektet har tilgang til å utføre den ønskede handlingen) Tilgangskontroll defineres som det å håndheve disse tilgangsrettighetene i et datasystem hver gang et subjekt (bruker, prosess, maskin, ol.) ønsker å utføre en handling (lese, endre, slette, etc) på et objekt (en ressurss, f.eks. datafil eller et dataprogram/tjeneste). Å skulle utføre tilgangskontroll for kombinasjonen subjekt, handling, ressurs vil dermed kreve at subjektet er autentisert (vi vet hvem det er), og at forespørselen kan kontrolleres mot et forhåndsbestemt regelverk (autorisasjonen, policyen) og enten godkjennes eller avvises. Tilgangskontroll = autentisering og autorisasjon Tilgangskontroll innebærer å sjekke om et subjekt har tilgang til å utføre en ønsket handling på et objekt, etter forhåndsdefinerte regler Håndhever og/eller overholder policyene som er satt Tilgjengelighet: • Informasjon skal alltid være tilgjengelig for de som har rett til å få tak i den • I praksis betyr det at informasjonen skal være «brukbar» for de som har rett til den uavhengig av feil, angrep, vedlikehold etc • **Viktig fordi:** I noen situasjoner trenger vi at informasjon er tilgjengelig til alle tider. Som et eksempel er det viktig å ha tilgang til pasientjournaler (?) til enhver tid, slik at denne informasjonen er tilgjengelig i tilfelle medisinsk personell trenger den. • **Sikkerhetstiltak:** sikkerhetskopier, rutinger for håndtering av data, gjenoppretting

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Autentisering	Skape tillit til ekteheten av en oppgitt identitet. Identifikasjon (hvem er du), autentifikasjon (legitimere at du er den du sier du er) Autentifikasjon kan skje gjennom noe du har (kodebrikke), noe du kjenner (passord, pin) eller noe du er (biometri - fingeravtrykk, ansiktsgjenkjenning) Utfordringer: lett å glemme, blir kanskje lagret i minne eller cache på maskinen. God passordhygiene er viktig. Kan være behov for kombinasjon av disse slik som bankinnlogging Kryptografiske autentiseringsprotokoller (f.eks. systemsertifikater som verifiseres av tredjepart) som TLS, VPN og IPSec (BankID) Kryptografiske mekanismer, PKI, digital signatur, elektronisk signatur **Autentisitet**: betegnelsen på egenskapen som forteller noe om ektehet, altså å vite og væreviss på at alle parter er de/den som de utgir seg for å være. De fleste kan se for seg nødvendigheten av å kunne stole på at en bruker i et datasystem er autentisk, og vet hva det er,men autentisitet er mer enn det. F.eks. at nettsiden du besøker faktisk er den nettsiden du mente å besøke, og ikke en identisk kopi noen har opprettet for å lure deg. Hovedtrusselen er at noe eller noen klarer å utgi seg for å være noen/noe de ikke er. Passord er f.eks. enkeltå stjele/gjette/knekke, og falske nettsider ganske enkelt å lage. **Autensitering**: Bekrefte identitet - er handlingen å bekrefte noe eller noen som autentisk, altså ekte eller sann. Autentisering og identifisering. Tekniske tiltak som bidrar til å skape autentisitet i en digital setting. Å bekrefte en hevet identitet. Ulike former for autentisering: organisasjonsautentisering, systemautentisering, brukautentisering **Autorisering**: Gi noen tilgang - *Autorisering* vil innenfor IT-området si å gi en gitt bruker tilgang til et veldefinert sett med ressurser. På forhold bestemmer seg for hvem som skal ha tilgang til hva. Å autorisere er å godkjenne eller gi løyve. Å autorierere er å spesifisere en tilgangs-policy. Spesifisere tilgangsrettighter til ressurser. Ansatt i lønnsavdelingen skal ha tilgang til lønnsystemet Autentisering: • Sjekke at noen er de de utgir seg for å være • Det vil si at man skal kunne verifisere at en person er den de sier de er, for eksempel i en innloggingssituasjon. • **<u>Viktig fordi:</u>** for eksempel når man vil logge inn i nettbanken, så er det viktig at kun den rette personen får tilgang før man får tilgang til sensitiv data. Det er (relativt) lett å få tak i brukernavn og passord for kontoer, så derfor er det viktig av vi setter inn mer avanserte mekanismer for å beskytte informasjonen vår • Det gjelder også at for eksempel nettsider er de de utgir seg for å være, og ikke kopier styrt av personer med onde

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

	hensikter. • **<u>Sikkerhetstiltak:</u>** 2-faktorautentisering, sertifikat på nettsider (?)
Uavviselighet	Sikre at mottaker eller avsender ikke kan bestride å ha sendt eller mottatt en melding. Bevis for at en melding er mottatt. Digitale signaturer kan løse dette. Det skal ikke være mulig å påstå at man ikke har gjort noe man har gjort, og at man har gjort noe man ikke har gjort. Dette gjelder f.eks om man har undertegnet en kontrakt (f.eks for boliglån), eller glemmer å overføre penger i nettbanken. Uavviselighet er altså å bekrefte at en handling eller et informasjonselement er uendret (integritet) og at det kan knyttes til en bestemt identitet. Uavviselighet er i mange sammenhenger også omtalt om ikke-benekting. En løsning for uavviselighet er en løsning som gjør det mulig for også en tredjepart å innhente tilstrekkelig dokumentasjon for at en annen part ikke kan nekte for å ha gjennomført en handlig eller ha valgt å bekrefte/vedgå seg etinformasjonselement. En løsning for uavviselighet er bygget opp på samme måte som løsninger som sikrer autentisitet. Forskjellen er at brukeren skal være i stand til ikke bare å bekrefte hvem man er overnettet, men også å legge igjen dokumentasjon som entydig knytter personen til en handlingeller et uendret informasjonselement. Uavviselighet: • Kommer fra u- avvise, eller ikke-avvise. • Man skal ikke kunne fornekte en handling. • Fra IN1020 ukesoppgaver 2018: Det skal ikke være mulig å påstå at man ikke har gjort noe man har gjort, og at man har gjort noe man ikke har gjort. • **<u>Viktig fordi:</u>** For eksempel om du har signert en kontrakt, skal ikke du eller den andre parten kunne nekte for at du har signert kontrakten. • **<u>Sikkerhetstiltak:</u>** Digitale signaturer (skal se på hvordan de fungerer senere)
Sporbarhet	Sikre at en gitt hendelse kan spores til en gitt identitet. Logging av alle hendelser Identifisere alle identiteter Perimeterforsvar, ikke gi uvedkommende tilgang Også omtalt som etterprøvbarhet, er egenskapen som sikrer at alle handlinger kanspores tilbake til en entydig entitet. Sporbarhet garanterer at alle operasjoner som utføres av individer (brukere, personer, systemer eller prosesser som kjøres kan identifiseres, og at sporene bevares (logges) for senere bruk. Sporbarhet:

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

	• Man vil kunne knytte en identitet til en gitt hendelse. • Poenget er at alle handlinger skal kunne spores tilbake til en identitet, og at sporene skal bevares til senere. • **Viktig fordi:** vi skal kunne spore tilbake til en kriminell person, om en kriminell handling skulle være utført, og denne personen kan bli holdt ansvarlig. • **Sikkerhetstiltak:** autentisering (slik at man skal finne den ansvarlige personen), logging av hendelser, etterforskning (korrigerende).
Personvern	Personvern stiller bl.a. krav til bl.a. hjemmel av behandling/oppbevaring av personopplysninger, samt skal sørge for at den registrertes rettigheter oppfylles. Personvern dreier seg også om å hindre uautorisert innsamling av data, at data som er samlet inn ikke benyttes til andre formål enn det den registrerte har samtykket til, det skal ikke lagres lenger enn nødvendig, samt at det gir den registrerte rett til innsyn i egne opplysninger, rett til å bli glemt, osv Personvern: • Ikke egentlig et sikkerhetsmål, men fortsatt så viktig at vi tar det med. • Handler i hovedsak om personopplysninger. • Det stilles altså krav til behandling og oppbevaring av personopplysninger som alle må følge. • **Viktig fordi:** vi ikke vil at personopplysninger om oss skal komme på avveie og at uvedkommende skal ha tilgang til å se slik informasjon • **Sikkerhetstiltak:** lovverket som tjenester må følge når det gjelder behandling og lagring av data.

Hvordan ivareta sikkerhetsmål? Innføre sikkerhetstiltak

- Ikke alltid nødvendig å ivareta alle målene
- For alle datatilstander
  - Lagring, Overføring (over nett eller fysisk bærer på minnepinne), Bruk
- I ulike faser av et IT-system
  - Forebyggende
  - Detekterende
  - Korrigerende
- Sørger vi for god sikkerhet i alle punktene vil vi få et forholdsvis sikkert system

3. Sikkerhetstiltak

Sikkerhetstiltak

- Tiltak vi gjør for å nå sikkerhetsmålene. Vi kan dele sikkerhetstiltak i flere kategorier:
- Fysiske, teknologiske og administrative tiltak er en måte å gruppere på.
- Forebyggende, detekterende, korrigerende tiltak er en annen måte å gruppere på.

Fysiske	**Tekniske**	**Administrative**
Eksempelvis: Låser	Eksempelvis: Kryptering/kryptografi	Eksempelvis: Opplæring, bevissthet

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Alarmer Vektere Kameraovervåkning	Skallforsvar Innbruddsdeteksjon Tilgangskontroll Sikkerhets-oppdateringer (OS, firmware, software) Sikkerhetskopiering Gjenoppretting Redundans på tjenester	Rutiner for øvelse og hendelses-håndtering Sikker systemutvikling (programmering) Definere retningslinjer, policyer, standarder Internkontroll og styringssystemer

Sikkerhetstiltak:

- Opplæring
- Kryptering (asymmetrisk, symmetrisk, sjekksum, kryptering av trafikk)
    - Sikre trygg lagring av data i utrygge lagringsenheter, sikre trygg overføring av data i f.eks åpne nett.
- Perimeterforsvar
- Tilgangskontroll
- Sikkerhetsoppdateringer
- Sikkerhetskopiering
- Gjenoppretting
- Redundans av tjenester
- Autentisitet
    - Autentisitet til ulike entiteter, med ulike autentiserings-former (se begrepsliste)
    - Brukerautentisering, med autentiseringsfaktorer:
        - «Noe du husker»
        - «Noe du har»
        - «Noe du er»
        - To -faktor/multi-faktor
    - System- og organisasjons-autentisering
    - Autentisering av dataopprinnelse
- Sikkerhetstiltak i ulike datatilstander:
    - Lagring
    - Overføring
    - Bruk
- Sikkerhetstiltak i ulike faser:
    - Preventive (forebyggende) tiltak, f.eks. Kryptering
    - Detekterende tiltak, varsle angrep som forsøkes eller skjedd.
        - Eksempel: inntrengingsdeteksjon(IDS)
    - Korrigerende tiltak: gjenopprette skade på dataressurser etter angrep.
        - Eksempel: hente sikkerhetskopi av programmer og data

Autentifiseringsfaktor:

Noe du vet	Noe du har	Noe du er
Fordel: • Du har det alltid med deg Ulempe: • Lett å glemme • Lett å gjette for	Fordel: • Du slipper å huske Ulempe: • Lett å miste • Kan stjeles/lånes Noe du har	Fordel: • Du slipper å huske • Du har det alltid med deg Ulempe: • Unikt nok? • Permanent nok?

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

uvedkommende • Lagres på datamaskin? Nedskrevet? • Lagres hos tjenesteleverandør Noe du har: • Passord til IT-system • Kode til elektronisk dørlås	• Nøkkel til lås • Engangspassord fra kodegenerator • SIM-kort • Kontaktløse brikker (RFID-kort)	• Sikkert nok? • Krever ekstrautstyr (sensorer, lesere, ol) • Ressurskrevende å sammenligne Noe du er • Ansiktsgjenkjenning • Fingeravtrykk • Scanning av øyet • Signatur • Ganglag

4. Sikkerhetstrusler

Sikkerhetstrusler

- Trusselscenarioer: Hvem, hvorfor, hvordan?
  - På datatrafikk:
    - Avlytting: Uvedkommende lytter til nettverkstrafikk og får tak i konfidensiell informasjon. Passivt angrep.
    - Data modifiseres (Man in the Middle-angrep, MitM): Uvedkommende griper inn i kommunikasjonen og modifiserer data. Aktivt angrep.
    - Forfalskning: Uvedkommende sender ikke-autentiske meldinger eller later som de er en annen. Aktivt angrep.
  - Tjenestenektangrep: Utilgjengelige ressurser
    - Bombadere en organisasjon slik at de ansatte ikke får tilgang til informasjon
  - Uautorisert bruk: Misbruk av ressurser
- Berørte sikkerhetsmål:
  - Konfidensialitet
  - Integritet
  - Autentisitet for opphav til data/melding.
  - Tilgjengelighet

Sårbarheter knyttet til:

- Sosial manipulering
- Menneskelig svakheter
- Nettverkssikkerhet/Nettverkskommunikasjon
- Applikasjonssikkerhet
- Sikkerhet i datamaskinen
- Lagring og avhending av informasjon
- Bruk av skytjenester

Dataangrep: Hvem?

En trusselaktør er	Hvem er trusselaktørene?
• Den som kan tenkes å utføre en potensielt skadelig handling • Hvilke trusselaktører	• Sporadiske «hackere» • Aktivister • Kriminelle

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

man bekymrer seg for er situasjonsbetinget.	• Virksomheter (fra konkurrenter) • Nasjonalstater (spionasje)

Skadelig programvare: Typer og kjennetegn

- SQL-injisering:
  - Hvis en applikasjon forventer et input fra brukeren på 16 byte, men du sender inn 20 byte, da kan det hende du overskriver noe annet av data og kan kjøre egen kode med de 4 siste bytene.
  - En velger altså å sende inn for mye data i håp om at man kan overskrive annen data og få kjørt egen kode.
- XSS: Cross site scripting:
  - år en angriper plasserer kode, som brukerinput på en side, som lagres hos web-tjeneren.
  - En annen bruker kan bli rammet av denne koden, som betyr at angriperen nå har kontroll over brukerens maskin eller data. Dette kan for eksempel skje via en gjestebok eller et kommentarfelt på en nettside.
- Phishing:
  - En svindler utgir seg for at de er noen de ikke er, for å så få deg til å logge inn på en side som ser ut som en annen side.
  - Dette gjøres i håp om at du skal skrive inn passord og annen vikting informasjon på deres side.
- DNS-sikkerhet:
  - En DNS-server kan bli utnyttet ved at en uvedkommende legger inn feil IP knyttet til et domenenavn.
  - I tillegg har vi man in the middle-angrep som er når en hacker plasserer seg imellom to endepunkter i en dataoverføring og tyvlytter på all data som sendes fra ende til ende.
  - For de to som kommuniserer virker alt helt normalt, som om de kommuniserer direkte med hverandre, men i virkeligheten strømmer all data gjennom hackeren først.
  - Hackeren kan også sende data, som for de to partene vil virke som legit, ufarlig data, men som egentlig er skadevare.
- Hardware:
  - Det kan finnes sårbarheter i hardware.
  - Et eksempel på dette er spectre som ble oppdaget for 1-2 år siden.
  - Den gikk ut på at intel prosessorer forutså hvilke kommandoer som skulle bli kjørt før de skulle, (for å ligge i forkant og spare tid). Dette kunne bli utnyttet for å kjøre kode som ikke skulle kjøres.
- Digital sårbarhet:
  - En feil i et system. Det kan være en svakhet i systemet eller en feil som er lett å utnytte.
  - Digitale sårbarheter kan være kjente av leverandørene, eller ukjente, som gjør at utnyttelse av disse sårbarhetene kommer som en overaskelse.
- Nulldags-sårbarhet:
  - En Nulldags-sårbarhet er en sårbarhet som ennå ikke er kjent for leverandøren av systemet.
  - Den kan altså være kjent av angripere, som gjør at man helst burde rette opp i den fortest mulig.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

        - En angrepsvektor er den konkrete metoden en angriper velger for å utføre et angrep på en datamaskin, et nettverk eller et system, mens en exploit er en konkret utnyttelse av en sårbarhet
- Skadevare:
    - Programvare som har som mål å ødelegge for eller utnytte personen som kjører den.
    - Skadevare kan være et eget program eller en bestanddel av et program.
    - Skadevare vil ofte prøve å spre seg til andre maskiner.
    - Skadevare består som oftest av et stridshode, som er måten den infiserer et datasystem, en spredning, som er måten den sprer seg på og nyttelast, som er det konkrete den gjør.

Ulike typer skadevare:
- Virus: spres gjennom åpning av fil av bruker eller systemet
- Ormer: sprer seg og finner nye ofre på egenhånd
- Trojaner: skadevare som utgir seg for å være et nyttig program
- Bakdør: en ubeskyttet og ukjent åpning inn i et system
- Spionvare: programvare for å spionere på en bruker for å få tak i info av nytte
- Logisk bombe: skadevare som kjøres ved et gitt tidspunkt eller ved gitt handling eller hendelse
- Tastelogger/keylogger: enhet/programvare som er koblet til tastatur eller maskin som lagrer alle tastetrykk
- Rootkit: skadevare som endrer funksjoner i OS eller applikasjoner, kan ta kontrollover maskin/system

Nettverkssikkerhet:
- Tradisjonell sikkerhetsmodell: stoler på endesystemet, nettverket er det som er upålitelig om noe skjer
- Tradisjonelle trusler: avlytting (passivt), forfalskning (aktivt), modifisering av data(aktivt)
- To hovedområder: kommunikasjonssikkerhet og perimeterforsvar
    - Kommunikasjonssikkerhet:
        - Skal beskytte data i transporten mellom virksomheter.
        - TLS (Transport layer security) passer på sikkerhet i transportlaget og er basert på PKI (Public key infrastructure).
        - IP security står for sikkerhet i nettverkslaget som baserer seg på kryptering, autentisering og nøkkelhåndtering.
    - Perimeterforsvar:
        - Sikkerhet ved hjelp av brannmur og innbruddsdeteksjon.
        - Brannmur - førstelinjeforsvar: Avgjør hva som slippes inn og ut fra et lokalt nettverk. Fungerer som en slags tilgangskontroll i nettverket og avverger uautorisert tilgang til/fra et privat nettverk. Implementeres i programvare eller maskinvare.
        - Innbruddsdeteksjon (IDS): System som detekterer mistenkelig aktivitet gjennom nettverksanalyse. Systemet kan detektere misbruk, skadevare og (D)DoS-angrep.

DNS-sikkerhet: (domain name system)
- DNS-forfalskning: uvedkommende introduserer uriktige opplysninger i navnetjener
- DNS-modifisering: Man in the middle-angrep, forfalsker meldinger
- DNS-kapring: endre hvilke navnetjener offerets maskin benytter (hacke maskin)

Tilgangskontroll:
- Et subjekt vil utføre handling på et objekt, for eksempel endre et dokument.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Tilgangskontroll = autentisering + autorisasjon.
- Autorisasjon = sjekke at subjektet har tilgang til å utføre den ønskede handlingen på objektet etter forhåndsdefinerte regler.

Tilgangskontroll i UNIX:
- Brukere (UID) og grupper (GID) er grunnleggende elementer, begge er permanente identiteter.
- Brukere har brukerkonto og tilhører en eller flere grupper, en av gruppene er primærgruppe
- Subjekter i UNIX: en prosess
- Objekter i UNIX: filer (programmer), mapper og devicer (maskinvareenheter,representert som filer)

Tilgangskontroll på flere nivåer
- Brukere og passord kan kontrolleres i applikasjonslaget
- Tilgang til ulike nettverksporter kan kontrolleres i transportlaget
- Tilgang basert på IP-adresser kan kontrolleres i nettverkslaget

Sikkerhet i operativsystemet:
- Kjernen i operativsystemet styrer alt.
- Kjernen kommuniserer med periferenhetene i datamaskinen gjennom drivere.
- Brukerprosessene må kommunisere via kjernen for å kunne benytte periferenhetene.
- Minnet i en datamaskin benyttes til mellomlagring.
    - Kjerner og drivere er dataprogrammer som kan inneholder sårbarheter. Kjernen har tilgang til alt (supervisor mode) og blir dermed en sårbarhet som er svært kritisk
    - Viktig å holde operativsystem og drivere oppdatert for å unngå at sårbarheter skal utnyttes.

Lagring av data
- Data kategoriseres etter hvilket beskyttelsesbehov det har og hvilke tiltak som iverksettes avhenger av beskyttelsesbehovet.
- Tre ulike behov:
    - Åpen informasjon
    - Informasjon med begrenset beskyttelsesbehov
    - Informasjon med sterkt beskyttelsesbehov

Sikkerhetskopiering
- Sikkerhetstiltak hvor man kan gjenopprette data ved uønsket hendelse som sletter/endre data.
- Husk: også sikkerhetskopien må sikres.

Applikasjonssikkerhet
- Applikasjoner må designes og utvikles med tanke på å kunne brukes trygt
- I designfasen: Trusselmodellering. Hvilke trusler ser vi? Hvilke tiltak kan iverksettes for å motvirke truslene?
- I programmeringsfasen: Unngå å skape/tilrettelegge sårbarheter

Buffer overflow
- En sammenhengende seksjon i minnet i datamaskinen, allokert til å inneholde alt fraen streng til en array med heltall, kalles en buffer
- En sårbarhet i et program som gjør at programmet overskrider bufferets grenser i minnet kalles buffer overflow.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

TSD - Tjenester for sensitive data

- Sensitive data:
  - Helsedata, data om etnisitet og rase, politiske meninger, religiøse eller filosofiske standpunkt, fagforeningsmedlemskap, sexliv
- Virtuelle maskiner:
  - betyr at man har en fysisk enhet i en laptop eller server også i stedet for å si at en server tilsvarer en maskin så tilsvarer den mange maskiner.
  - Man simulerer altså at man har flere fysiske maskiner.

Skallforsvar

- Brannmur:
  - Slipper inn eller avvise trafikk inn i og/eller ut fra et nettverk basert på forhåndsdefinerte regelsett.
  - Brannmur fungerer som en tilgangskontroll for nettverket.
  - Kan være et dataprogram eller maskinvare laget for akkurat dette formålet.
- Innbruddsdeteksjon
  - Overvåker nettverkstrafikk, og kan detektere:
    - Både forsøk på og suksessfulle innbrudd
    - Datavirus og annen ondsinnet programvare
    - Tjenestenektangrep
- Behov for teknologi og algoritmer som kan sikre dette som kan avsløre mistenkelig oppførsel

Nytteverdi av digitale sertifikater i https

- Digitale sertifikater hindrer:
  - Falske nøkler og falske tjenester. Den offentlige nøkkelen til en entitet inngår i en ubrutt kjede av sertifikater (med nøkler) som går god for hverandre, helt opp til det øverste nivået, rot-sertifikatet.
  - Vanskelig for svindlere å generere og spre falske nøkkelpar til allerede etablert tjeneste.
- Digitale sertifikater hindrer ikke:
  - Svindler i å opprette et gyldig domene, f.eks. sparebnak1.com, generere privat/offentlig nøkkelpar, og få dette signert av tiltrodd sertifikatutsteder. Fremgangsmåten benyttes ofte i svindel-angrep
- HTTPS bidrar kun til å krypterer kommunikasjonen og verifiserer motparten.
- Tillitt: Stoler dere på oss når vi ber dere gå til uio.inspera.no for å gjøre prøveeksamen?
- Tips:Sjekk sertifikatet i nettleseren.

5. Kryptering og nøkler

Kryptering

- Hovedmål:
  - Sikre konfidensialitet og/eller integritet
  - Brukes også til autentisering av dataopprinnelse og for å oppnå uavviselighet.
- Krypteringsformer:

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

    - Hash-algoritmer (enveis-kryptering)
    - Symmetrisk (én hemmelig delt nøkkel)
    - Asymmetrisk (nøkkelpar: offentlig + privat nøkkel)
- Nytten av kryptografi innen datasikkerhet:
    - Sikre konfidensialitet
    - Ivareta integritet
    - Bidra til autentisitet
    - Oppnå uavviselighet

Krypteringsnøkler

- Styrken i kryptografisk sikkerhet avhenger av:
    - Størrelsen/lengden på nøkkelen.
    - Håndtering og beskyttelse av nøklene.
    - Styrken i den kryptografiske algoritmen.
- God nøkkelhåndtering er basisen for trygg generering, lagring, distribusjon og destruksjon av nøkler.

Symmetrisk kryptering:

- En felles nøkkel brukes til kryptering og dekryptering.
- Man må ha delt denne nøkkelen på forhånd slik at både sender og mottaker har nøkkelen.
- Senderen bruker nøkkelen til å kryptere meldingen.
- Meldingen blir sendt.
- Mottakeren bruker den samme nøkkelen til å dekryptere meldingen.

Asymmetrisk kryptering:

- En nøkkel brukes til kryptering og en annen nøkkel brukes til dekryptering.
- Disse to nøklene opptrer i par.
- Når to parter skal kommunisere så har senderen og mottakeren 2 nøkler hver– en offentlig og en privat. Disse nøklene opptrer altså i par.
- Det betyr at en melding som krypteres med offentlige nøkkelen kan KUN bli dekryptert med den tilhørende private nøkkelen (og omvendt).
- I kommunikasjon mellom to parter vil asymmetrisk kryptering fungere slik:
    - Senderen vil sende en kryptert melding til en mottaker.
    - Da bruker senderen mottakeren sin offentlige nøkkel til å kryptere.
    - Når mottakeren skal dekryptere meldingen så bruker mottakeren sin egen private nøkkel.
    - I digitale signaturer så bruker jeg min egen private nøkkel for å kryptere.
    - Når den digitale signaturen skal bli verifisert, så brukes min offentlige nøkkel til dette.

Hashing:

- Brukes til å kryptere en melding.
- Man trenger bare en sjekksumalgoritme som tar inn teksten som skal krypteres og genererer en sjekksum for denne.
- Sjekksummen inneholder helt tilfeldige strenger av tall og bokstaver.
- Når vi har en sjekksum, kan vi ikke rekonstruere den originale meldinga.
- Vi kan ikke endre en melding uten at sjekksummen blir endret.
- Ingen beskjeder har ikke samme sjekksum.

Hva er et sertifikat?

- Et sertifikat består av den offentlige nøkkelen, samt en rekke attributer knyttet til nøkkelen:
    - Eier, gyldighetsperiode, utsteder, kryptoalgoritme brukt, nøkkelstørrelse, etc.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Attributtene pakkes sammen og signeres av sertifikatutstederen = vi har et sertifikat
  (En standard for sertifikater er såkalt X.509)

Hvordan distribuere offentlige nøkler?

- Man lager en «Offentlig-nøkkel infrastruktur» (PKI)
  - *Public key infrastructure*
    - Rammeverk for utstedelse, administrasjon og bruk av digitale sertifikater over datanettverk.
    - Anvendelsesområder for PKI: kryptering, autentisering og signering av dokumenter eller programvare.
    - Signering har to hovedfunksjoner: Verifisering av dataintegritet og ikke-avviselighet.

Public Key Infrastructure (PKI):

- Sikre autentiske offentlige nøkler.
- Binder en offentlig nøkkel til en navngitt enhet, og bindingen kan bekreftes av en betrodd "myndighet" som utsteder et sertifikat.
- Et sertifikat forteller følgende: "Offentlig nøkkel K eies av X".
- Rammeverk for kryptering/autentisering og signering av dokumenter og programvare
- Asymmetrisk kryptering, med en offentlig og en privat nøkkel
  - Eks: signere lån fra lånekassen: bruker Bank ID
- Hensikt med digitale sertifikater: Å knytte sammen en offentlig nøkkel og et individ/identitet.
- PKI er et rammeverk for utstedelse, administrasjon og bruk av digitale sertifikater med offentlige nøkler
- **Hovedformål:** Sikre ektheten av offentlige nøkler, samt forenkle nøkkel-distribusjonen.
- PKI må inneholde:
  - En policy for sertifikat-håndtering
  - Teknologi for å implementere policyen
  - Prosedyrer for hvordan håndtere og forvalte nøklene
  - Tillittsmodell for sertifikatene med offentlige nøkler

6. Personvern

Personvern:

- Ikke egentlig et sikkerhetsmål, men fortsatt så viktig at vi tar det med.
- Handler i hovedsak om personopplysninger.
- Det stilles altså krav til behandling og oppbevaring av personopplysninger som alle må følge.
- Personvern stiller bl.a. krav til bl.a. hjemmel av behandling/oppbevaring av personopplysninger, samt skal sørge for at den registrertes rettigheter oppfylles.
- Personvern dreier seg også om å hindre uautorisert innsamling av data, at data som er samlet inn ikke benyttes til andre formål enn det den registrerte har samtykket til, det skal ikke lagres lenger enn nødvendig, samt at det gir den registrerte rett til innsyn i egne opplysninger, rett til å bli glemt, osv
- **Viktig fordi:** vi ikke vil at personopplysninger om oss skal komme på avveie og at uvedkommende skal ha tilgang til å se slik informasjon
- **Sikkerhetstiltak:** lovverket som tjenester må følge når det gjelder behandling og lagring av data.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Personopplysningsvern

- Personvern - loven, vit at den finnes
- GDPR er en stor innstramming i personvernsregelverket i hele EU/EØS og andreland som opererer innenfor EU/EØS som gjelder fra mai 2018.

Hva er en personopplysning:	Viktig i Personvernforordningen:
• Særlig kategorier Personopplysninger Ordinære personopplysninger	• Registrertes rett til privatliv • Lovlighet av behandlingen (hjemmel/samtykke/hvilke data) • Åpenhet om behandlingen (til hva) • Hvor lenge man kan lagre informasjon • Krav til informasjonssikkerhet

7. Prosess

Helhetlig prosess

- Må ta hensyn til dette hele veien, fra A til Å



Tjenesteutsetting og sky

- Tjenesteutsetting (Gir andre tilgang til «dine» data)
- Skytjenester (Gir andre tilgang til «dine» data)
- Sikkerhetsmessige utfordringer knyttet til behandling av informasjon, og spesielt personopplysninger, når data behandles hos en 3.-part.

Skytjenester

- Fordelene med skytjenester er blant annet redusert infrastrukturkostnader, besparelser i form av betaling for faktisk bruk og lokasjonsuavhengig tilgang.
- Ulemper er at dersom en virksomhet ønsker å benytte skytjenester til behandling av personopplysninger, er virksomheten juridisk ansvarlig for at personopplysningene prosesseres og lagres i samsvar med personvernregelverket.

Innebygget sikkerhet oppnås når informasjonssikkerhet er:

1. innarbeidet i virksomhetsstyringen og understøtter virksomhetens mål
2. innarbeidet i virksomhetens prosesser og prosjekter fra starten av
3. tatt hensyn til i hele livssyklusen til IKT-løsninger
4. et tema alle ansatte i offentlig sektor kjenner til og vet hva innebærer for sinearbeidsoppgaver

Trusselmodellering

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Trusselmodellering er en prosess der man systematisk gjennomgår sikkerheten for et IT-system:
    - Identifisere potensielle trusler
    - Evaluerer konsekvensen assosiert med hvert område
- Mål:
    - Tidlig håndtering av sikkerhetsproblemer
    - Gjøre bedre sikkerhetsvurderinger
    - Mer effektiv sikkerhetstesting
- Flere innfallsvinkler:
    - Hva er verdifullt i systemet, hvordan kan det gå tapt?
    - Hva er motivasjonen for et angrep?
    - Hva har gått galt tidligere?
- Husk: det å beskytte ressurser skaper verdi ;)

Vurdering av trussel og risiko

- Kunne anvende de hittil nevnte punktene til å gjøre en enkel trusselmodellering (sikkerhetsvurdering) av et gitt scenario.
- Risikoanalyse innebefatter også vurdering av potensielle kostnader/tap, og ender opp i en prioritering av nødvendige sikkerhetstiltak.

Risikoanalyse

- Risiko involverer konsekvens, gjerne målt i kroner og øre.
- Flere definisjoner, men:
  «Risiko er et produkt av sannsynlighet for hendelse og kostnad/tap knyttet til hendelsen»
- Hvor skal innsatsen legges?
    - Usannsynlig hendelse som medfører store tap?
    - Sannsynlige hendelser som har lavere kostnad?

Kontinuerlig prosess

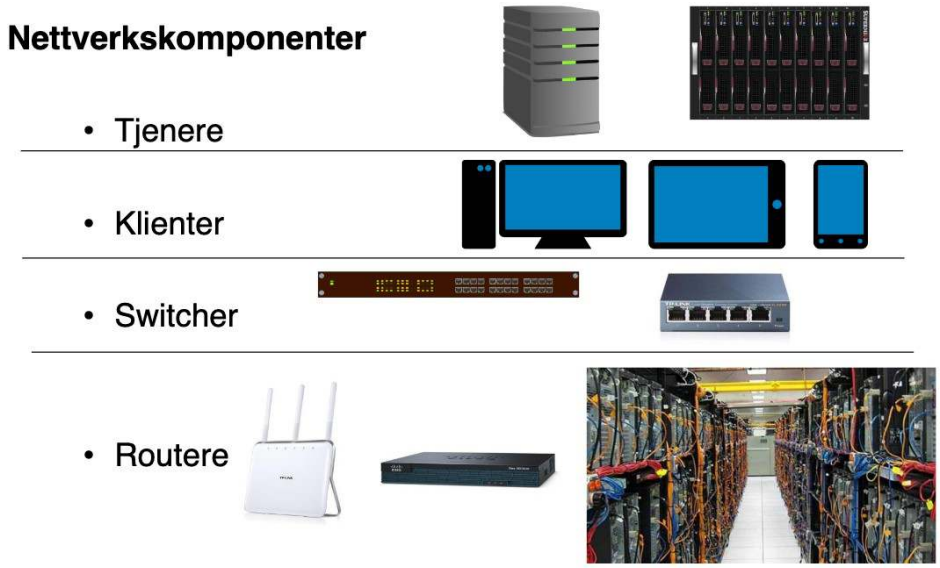
- Må innarbeides som en viktig del av virksomhetens prosesser (blir aldri ferdig)
- Selges inn til en virksomhets ledelse
- Sikre datasystemer = å skape verdi
- Sikkerhet koster
- Manglende sikkerhet koster (mer?)

Nettverksdelen

1. Nettverkskomponenter

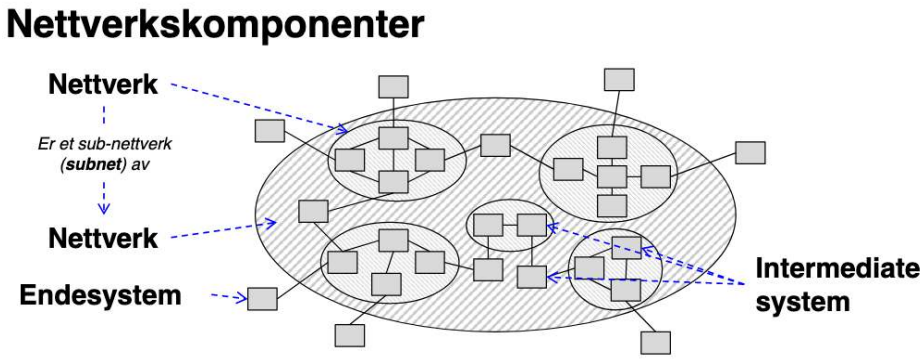
Nettverkskomponenter i datanettverk

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)



- Tjener/Server om hverandre
- Klienter - endebrukersystem feks. laptop, mobil
- Switcher - bokser med kontaktuttak
    - Hjemme eller på datasenter
    - Ikke smart, sender data videre til neste steg
    - TP-kabelene(twisted-pair), eternettkabel
    - Ikke treigere, men en forsinkelse som en skjøteledning
- Ruter smartere enn switcher
    - Vet hvor beskjeden skal
    - Har en forståelse om dataen skal videre eller være hjemme
    - Noen rutere har innebygd switch

Nettverkskomponenter



- Endesystemer
    - Finnes helt ytterst i nettverket
    - Eks. Datamaskin, sensor, mobil, skriver
- Intermediate system
    - Eks. ruter, wwitch, gateway, repeater, bridge
- Internett - mange datanettverk koblet sammen

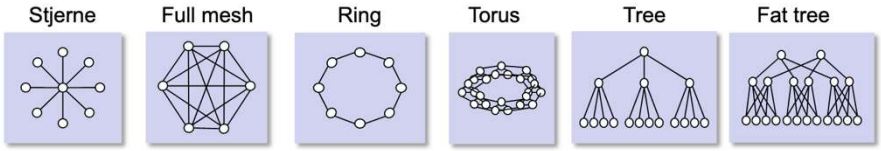
<u>Nettverksstrukturer:</u>

- Punkt-til-punkt nettverk:
    - Flere forskjellige kabler, kabeltyper eller radiolinker som kommuniserer fra punkt til punkt.
    - Kabel eller radiolink kobler alltid sammen to noder.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- En-til-en overføring.
- Topologieksempler:



- Broadcast-nettverk:
  - Nettverk som deler kommunikasjonsmedium.
  - En sender, alle lytter (en-til-mange).
  - Bruk:
    - Trådløs: Eneste mulighet (mobiltelefoner, satellitter, radio, NFC, ...)
    - Kablet: Gamle nettverk (Coax, Token ring)

Eksempler

-



Endesystem:

- Enheter som ligger helt ytterst i et nettverk, der kommunikasjonen over nettverket skjer primært mellom disse to enhetene (som vanligvis er styrt av mennesker eller prosesser på en datamaskin).
- Vi kan og dele inn endesystem inn i to kategorier: klienter og tjenere/servere.
- En server programvare/enhet som tilbyr en tjeneste til klientene
- den SERVERER en tjeneste.
  - Eksempler: datamaskiner, mobil, printer, kjøleskap, tv – disse vil være klienter.
  - Eksempler: web server - dette omtales som server.

Intermediate system:

- Enheter som ligger mellom endepunktene i et nettverk.
- Slike enheter brukes ikke direkte av brukere, men istedenfor mottar pakker med informasjon og sender disse videre i riktig «retning» på nettverket til det spesifiserte endesystemet.

- Switcher:
  - En nettverkskomponent som kobler sammen flere enheter på et datanettverk. Switchen lar disse enhetene som er på det samme nettverket (subnettet) kommunisere med hverandre.
  - Dvs. at switcher kan koble sammen mobiler, dataamaskiner, printere, eller kjøleskap sammen og lar de kommunisere med hverandre.
- Routere:
  - Nettverkskomponent som lar ulike nettverk kommunisere sammen.
  - Routeren videresender pakker/rette pakker mot mottakernoden på nettet, og er ansvarlig for å velge den beste ruten for datapakkene med informasjon skal ta gjennom nettverket slik at kommunikasjonen skal skje så effektivt som mulig.
- Gateway:
  - En enhet som brukes for kommunikasjon mellom nettverk med ulike protokoller.
  - Gatewayen vil på en måte «oversette» fra det «språket» det ene nettverket bruker til dent «språket» det andre nettverket bruker for at kommunikasjon mellom de skal være mulig. Gatewayen er «plassert» på kanten av nettverket som betyr at all

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

kommunikasjon fra utsiden og inn til nettverket (og omvendt) vil skje gjennom gatewayen – derfor det heter en «gate».

- Mange eksempler på gateways.
- En gateway kan være mye, som for eksempel router eller brannmur etc.
- Et gøyt eksempler er «Internet-To-Orbit Gateway» - «En internet-to-orbit-gateway (I2O) er en maskin som fungerer som en kontakt mellom datamaskiner eller enheter som er koblet til internett og pc-systemer i bane rundt Jorden, for eksempel satellitter eller bemannede romfartøy.»

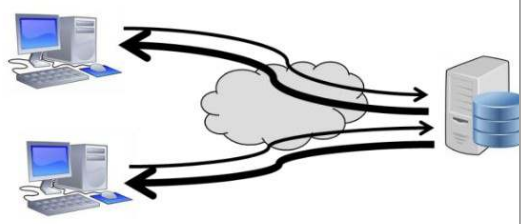
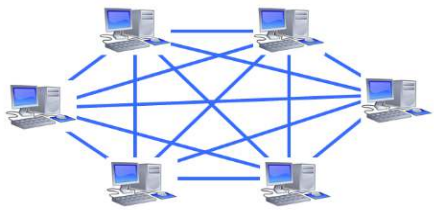
2. Aksessmodeller

Forbindelsesstrategier:

- Pull: klienten ber tjeneren om en tjeneste, og er den vanligste metoden
- Push: tjeneren dytter en tjeneste til en klient, dette krever at det er en forbindelse fra før eller at klienten lytter (Eks. push varsler).
- Publish-subscribe: variant av push der tjeneren dytter ut beskjeder til en gruppe av abonnenter.

Aksessmodeller:

- Klient-tjener
  - Tradisjonell kommunikasjonsmodell: Klient ber om en tjeneste (oppretter en forbindelse), tjenere leverer tjenesten (svarer på forespørselen)
  - Eks. Webklient (nettleser), mail
- Peer-to-peer (P2P)
  - Første velkjente programmet: Napster
  - Fildeling (brukt til f.eks. musikk og film. Eks: Popcorn time)
  - Hvis en maskin har tilgang til en fil blir dette delt i et felles nettverk
  - Alle noder er likeverdige - Alle noder kan nå hverandre
  - Eierskapet er distribuert

Klient-tjener	Peer-to-Peer (P2P)
Klienter ber om en tjeneste (oppretter en forbindelse) Tjenere leverer tjenesten (svarer på forespørselen) 	Alle noder er likeverdige Alle noder kan nå hverandre Eierskapet er distribuert 

Fysisk plassering av innholdet

- Man kan utvide cachehierarkiet for å gjelde også utenfor datamaskinen. Da kan man mellomlagre data på ulike avstander fra brukeren også på internett.

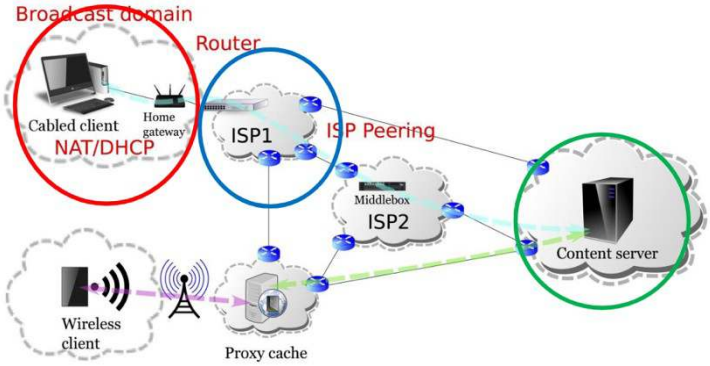
This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Eksempelvis ha en lokal proxy på et lokalt internett, hvor den har en cache som kan levere data med kort responstid til brukerne av det lokale nettet.
- Internettleverandører har ofte ting lagret hos den for store bedrifter som trenger data istore deler av verden.
- Man kan også gå til den originale datakilden selv om dette tar mye lengre tid

Fysisk plassering av innholdet

Cachenivå	Fysisk beliggenhet	Est. RTT
Lokal Proxy Content Delivery Network Original datakilde	Organisasjon / LAN ISP Internett	< 10ms < 50ms ~10ms - ~250ms



3. Lagene i Internett (TCP/IP referansemodellen)

TCP/IP-modellen:

- Forenkling av OSI
- Applikasjon: der programmene i nettet kjører
- Transport: Kobler sammen ende-til-ende (TCP/UDP). Finner porten dataen skal sendes til (f.eks. nettleser, iMessage osv.)
- Nettverk: Rute data fra ende-til-ende systemer (IP) - vanskeligst å gjøre endringer. Finner riktig mottaker
- Link: Pålitelig overføring mellom to noder (MAC-adresse, Ethernet, LAN), at alt ersendt og riktig rekkefølge
- Fysisk: Sender bit ut på mediet (kablet eller trådløst)

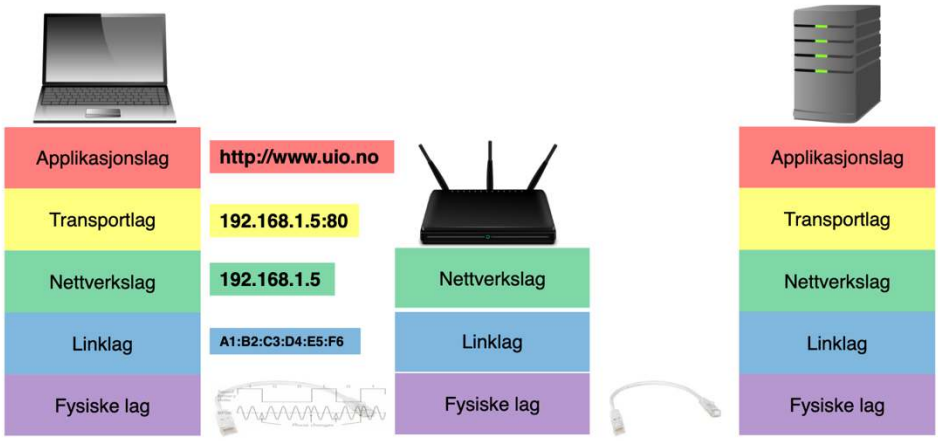
Hva er en protokoll?

- En protokoll i IT-verden er et standardisert sett med regler som bestemmer tilkobling, kommunikasjon og dataoverføring mellom to endepunkter.
- Protokoller kan implementeres i b̊ade hardware og software, eller begge samtidig.
- N̊ar det kommer til datakommunikasjon skal vi n̊a se nærmere p̊a TCP/IP-modellen.
- Dette er et sett med kommunikasjonsprotokoller som brukes p̊a Internett og datanettverk.
- TCP/IP-modellen spesifiserer hvordan data bør pakkes, adresseres, sendes og mottas.
- Modellen brukes s̊a og si overalt, s̊a det er lurt̊a gjøre seg litt kjent med den.
- Vi gjør som en god diktator og splitter modellen i to og ”hersker” over hvert av begrepene som bygger opp TCP/IP.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Hva er poenget med lagdelingen?

- Lagdeling = standardiserte abstraksjonsnivåer.
- (Lagdeling av en datamaskin (fra transistorer til internett): er en forutsetning for å lage en datamaskin. Takket være lagdeling er det mulig å håndtere kompliserte konstruksjoner.)
- Lagdeling i internettarkitekturen: utfordringene utgjør at det kan være veldig komplisert å kommunisere med fremmede maskiner på nettet og interaksjon kan påvirkes av forskjellige typer system/nettverk (f eks PC/CPU vs MobilTlf/CPU), man må sørge for at maskinene snakker samme språk.
- Løsningen på dette er standardiserte abstraksjonsnivåer/moduler/lag. Data som skal sendes over nettet går altså gjennom flere lag og oversettes til et språk som mottakermaskinen til slutt kan forstå.
- Lagene kan byttes ut/endres for å tilpasse ulike oppgaver og språk/kommunikasjonsmåter (hvilket er enda en motivasjon for å bruke flere lag).
- Lagdelingen gjør det f.eks. mulig for nettverkskomponenter (som swithcher og routere) å implementere bare et delsett av lag, for å støtte bare det som trengs for å sende en pakke videre.



Applikasjonslaget **HTTP-protokollen**	Lag med tjenester for applikasjoner, som nettleser, e-post ol. • Kommunikasjon mellom applikasjoner • Vår maskin kontakter web-serveren og sende HTTP request • Vi forbreder forespørselen til web-serveren • Legger til nødvendig metadata i dette laget og sender pakken videre • Hypertext Transfer Protocol
Transportlaget **TCP/UDP-protokollene**	I transportlaget er det to viktige protokoller, TCP og UDP. • Transmission Control Protocol / User Datagram Protocol • TCP har et oppsett av forbindelse med 3-veis håndtrykk. • UDP (User datagram protocol) er en tilkoblingsløs protokoll og har ingen garantier. • Kobler sammen systemene ende-til-ende • UDP: Tilkoblingsløs • TCP: Forbindelsesorientert, pålitelighet, sikrer flytkontroll, sjekksum, metningskontroll, garanterer at pakkenes sendes i riktig rekkefølge
Nettverkslaget	Kobler sammen ende-til-ende systemer. • Ansvarlig for ruting - hvordan pakken skal flyte gjennom nettet.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

IP-protokoll	• IP (Internet protocol) er den mest brukte nettverkslag protokollen. En IP må være unik siden det er ende-til-ende. • Dette gjør at vi begynner å få for få IP-adresser og vi går nå fra en 32 bit adresse (IPv4) til en 128-bit adresse (IPv6). • Sende data fra ende-til-ende systemer • Legger på en IP-header
Linklaget **MAC/WiFi-protokoll**	Har som oppgave å sikre pålitelig overføring mellom to enheter, passe på enkel flytkontroll og feildeteksjon. • Medium Acess Controll Protocoll / Wireless Network protocols • Hver endenode får en MAC-adresse (Medium Access Controll) på 6 byte som ofte erlagret i nettverkskortet. • Dette gjør at ingen snakker i munnen på hverandre. • Det vanligste linklaget er Ethernet og Wi-Fi. • Pålitelig overføring mellom to noder • Mac-adressen til ruter (avsender) • Mac-adressen til server (mottaker)
Det fysiske laget **Kablet eller trådløst**	Består av signalrepresentasjon av bits. Sørger for at 1-bit blir mottatt som 1-bit. • Dette gjennom kabler eller andre medium. • Står for formelle regler om kommunikasjon. • Beskjeden klar til å sendes • Vet hvor informasjonen skal og hvordan informasjonen skal sendes • Legger til mer og mer informasjon nedover • Når den sendes oppover dekodes den til mottaker

	Lag	Funksjon
5	Applikasjon	Applikasjonsrelaterte tjenester (HTTP, Mail)
4	Transport	Kobler sammen systemene ende-til-ende (TCP/UDP)
3	Nettverk	Sende data fra ende-til-ende systemer (IP)
2	Link	Pålitelig overføring mellom to noder (LAN/WiFi)
1	Fysisk	Sender bit ut på mediet (kablet eller trådløst)

Transmission Control Protocol (TCP)

- Forbindelsesorientert
  - Settes opp ved et 3-veis-håndtrykk
  - SYN-SYN+ACK-ACK (se figur)
- Flytkontroll
  - Sender ikke fortere enn mottageren kan ta imot
- Metningskontroll
- Byte-strøm og levering i rekkefølge
- Pålitelighet
  - Implementert ved at bekreftelser på hver pakke sendes tilbake fra mottakeren

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Feilsjekking av nyttelasten (sjekksum)
- Mest brukte, web, epost, filnedlasting osv

User Datagram Protocol (UDP)

- Motsatt av TCP
- Forbindelsesløst
  - Ikke oppsett av forbindelse på forhånd
- Ingen flytkontroll eller metningskontroll
  - Pakker sendes ut så fort som mulig
- Ingen garanti for rekkefølgen
- Ingen garanti for pålitelighet
- Feilsjekking av nyttelasten (sjekksum)
- Brukes ofte i videostreaming

TCP Segment Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset	Res	Flags	Window Size
128	Header and Data Checksum		Urgent Pointer	
160...	Options			

UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

Lagdeling i internettprotokollen

- En protokoll definerer strukturen på beskjeder sendt over et nettverk.
- En protokoll må adressere mange kompleksiteter:
  - Hvordan skal maskinvaren oppføre seg?
  - Hvordan skal beskjeden finne frem?
  - Er det noen garantier for levering?
  - Hvordan håndtere kø, tap og andre problemer?
- En protokoll handler om kommunikasjon mellom samme lag. Protokollen definerer formatet på beskjeden og en header.

Nettverkskomponenter:

- Endesystemer(C) finnes ytterst i nettverket(datamaskiner, mobiler, servere osv.)
- Intermediate systemer (D) finnes inne i et nettverk (ruter, switch, gateway osv.) - Subnet (A), en gruppe av enheter innenfor et nettverk (B)

Nettverksstrukturer:

- Punkt-til-punkt: forskjellige typer kabler eller radiolinker som kommuniserer fra punkt til punkt (en-til-en)
- Topologi: f.eks. ring eller stjerne
- Broadcast-systemer: nettverk som deler kommunikasjonsmedium. Én sender, alle lytter (én-til-mange), eks. trådløst internett. En melding som sendes ut på en spesiell adresse og leveres til alle enheter på samme LAN. Problem: hvordan takle kø, alle skal ikke få "høre" alle sitt.

Protokoller og lag

- Utfordring: komplisert å kommunisere med fremmede maskiner på nettet og interaksjon mellom forskjellige system og/eller nettverk

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Forenkling: introduserte standardiserte abstraksjonsnivåer (lag)

Lag i nettverket (OSI)

- (N)-lag er et bestemt abstraksjonsnivå
- (N)-entity er oppgaven til et lag (typisk en prosess)
- (N)-Service Access Point er tjenesteidentifikasjon. Beskriver hvordan lag N tilbyr entjeneste til lag N+1
- (N)-protokoll er et sett med regler for hvordan data skal overføres mellom entity på samme nivå

Dataenheter i OSI-modellen

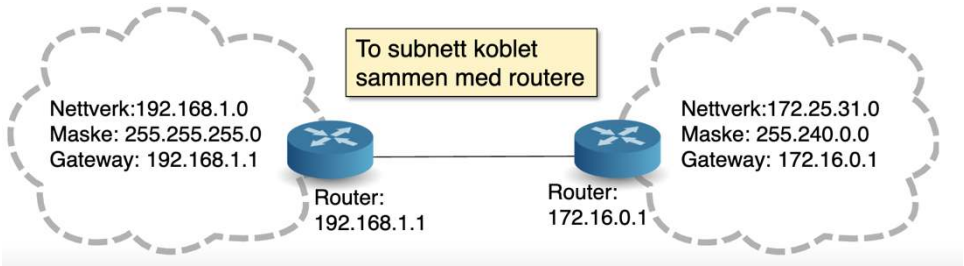
- Data units har ulike navn ut i fra hvilket lag vi befinner oss i
- Transportlaget: packets
- Nettverkslaget: datagram
- Linklaget: Frame

Lagene spiller sammen

- Internett er en sammenkobling av mindre, separate nettverk. På lokale nettverk kan man koble sammen enheter ved hjelp av en switch eller en HUB, dette fungerer på nettverkslaget.
- For å sende en pakke til en maskin utenfor ditt lokale nettverk må den sendes til en router som vet hvor en skal videresendes.

Lokalnettverk (LAN), subnett og broadcast

- Internett er en sammenkobling av mindre, separate nettverk.
- Koblet sammen med switcher og/eller HUBer.
    - Kunne regne seg frem til nettmaske og broadcast adresser.



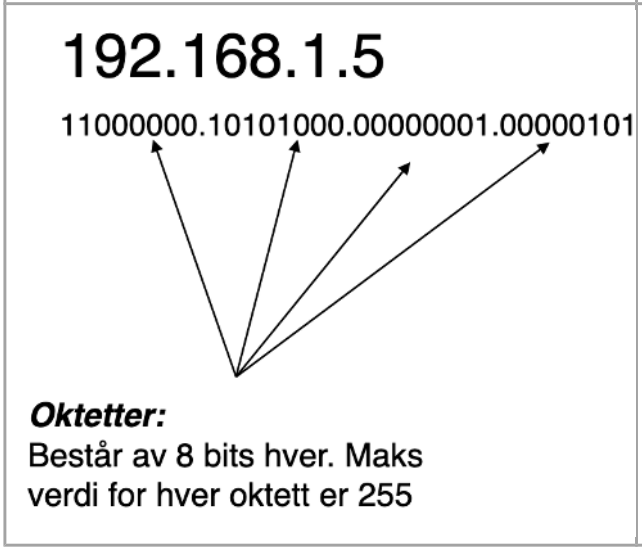
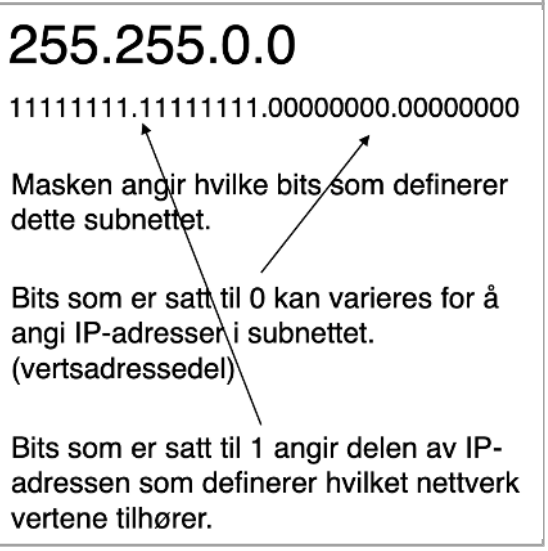
Hovedkomponenter i et operativsystem

- "Synlige" for bruker:
    - Brukergrensesnitt
    - Filsystem
    - Enhetskontroll
- "(Relativt)Transparent":
    - Prosessorkontroll
    - Minnekontroll
    - Kommunikasjonstjenester

4. IP-adresser og CIDR-notasjon

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

IP-adresser (IPv4)

IP-adresse	Nettverksmaske
	

Hvordan fungerer DNS/Domain Name System?

- DNS synker domenenavn med IP-adresser hvilket gir brukere domenenavn/adresser som er enkle å huske, mens maskinene på nettet bruker IP-adressene.
- DNS identifiserer og lokaliserer datasystem og andre ressurser på nettet; når man skriver inn en webadresse/URL matcher DNS denne adresse med tilsvarende IP-adresse og kobler en til den nettsiden.
- DNS er en database som forvarer alle domenenavn og tilsvarende IP-adresser for en spesifikk top-level domain (TLD), f eks .com, .net etc.
- DNS databaser er lagret på fysiske servere i en distribuert database, dvs. Mange maskiner på en måte som gjør at man kan få tak i den informasjonen man trenger hvor enn man befinner i verden.
- Hierarkisk navnetilordning:
  .com > google.com > mail.google.com
- Ulike maskiner har ansvar for forskjellige deler av navneoppslagene.
- DNS har en enkel klient/tjener arkitektur (den mest brukte tilkoblingsmåten på nettet), man setter opp en maskin/tjener som oppretter en tjeneste/lager en port f eks TCP. Alle maskiner vet da at hvis de kobler seg til akkurat den porten - kobles de til en DNS-tjeneste. UDP (vanligst frem til nå, brukes fortsatt på de fleste klientmaskiner) eller TCP port 53, fra nylig må tjener bruke TCP. Dette grunnes overgang til internett protokoll 6 hvilket medfører veldig lange IP-adresser, som ikke får plass i en DNS-pakke. Informasjonen som da skal sendes med UDP, må deles opp i to pakker hvilket medfører større risiko for å miste informasjon på veien. Klienter som bruker TCP avvises ofte.
- Klassisk metode/rekursivt oppslag: forespørselen går igjennom mange maskiner før den returneres til maskinen som først spurte. Dataflyten konsentreres rundt de sentrale tjenerne der også mange tilstander lagres. En del ressurser blir dårlig brukt.
- Ny metode/iterert oppslag: man delegerer til den første gyldige DNS-tjeneren i stedet for å gå igjennom mange maskiner på veien. Tilstanden lagres bare av den lokale tjeneren inntil svaret er levert.

Navnehierarki og rottjenere.

- Rottjenere (et set med maskiner) har lister på TLDs samt adresse til hvem det er som er ansvarlig for disse domenene. Siste leddet i en nettadresse. F eks .no, .com

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Disse TLD' forgrener seg nedover, f eks. .no > uio.no > ifi.uio.no > www.ifi.uio.no
- Rottjenerne administreres av et foretak, ICANN (frem til nylig eid/administrert i USA, er nå mer globalisert). I Norge er det UNINETT-NOR, disse godkjenner ansøkninger om bruk av nye domenenavn. Deretter er UIO administrator.
- Hver DNS-tjener har autoritet over en del av hierarkiet, de skal ha en liste over alle som sorteres under sitt sub-tre. Må kunne replikeres, derfor er det minst to tjenere på alle nivåer. Alle må kjenne til adressene til rottjenerne.
- Rottjenerne er ansvarlige for "Root Zone File"/liste over TLDer og hvem somkontrollerer dem. Det finnes 13 rottjenere hver av 6 er replikert globalt med en teknikk som heter "anycast".
- Rottjeneren kontaktes når man mislykkes med navneoppslag. I praksis mellomlagres denne informasjon i cache på de fleste system.
- DDoS-angrep: denial of service-angrep, man prøver å sende så mange forespørsler at serveren går ned (har vært nær, men ikke skjedd).
- Rottjenerne ligger fremst i USA.

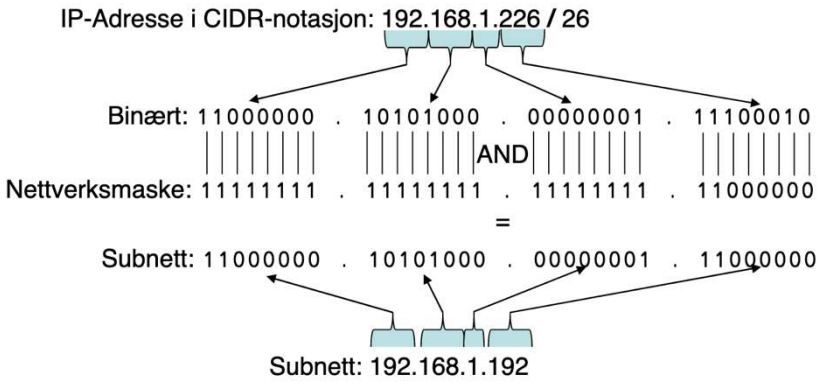
CIDR- og punktnotasjon av subnett (IP/INTERNETT)

- Nettverksmasken består alltid av en sammenhengende serie "1" deretter en sammenhengende serie "0"
  Eks: 255.255.255.0
  11111111.11111111.11111111.00000000
  nettverksdel vertsdel
- Det er to vanlige måter å notere omfanget av et subnett:
  - Punktnotasjon:
    - For eksempel: 192.168.1.0
    - Må da oppgi nettverksmaske: 255.255.255.0
  - CIDR (Classless Inter-Domain Routing) notasjon:
    - 192.168.1.0/24
    - Vanlig punktnotasjon først.
    - Tallet etter skråstreken angir hvor mange bits nettverksmasken består av

Regne ut subnettet fra en IP + nettverksmaske (IP/INTERNETT)

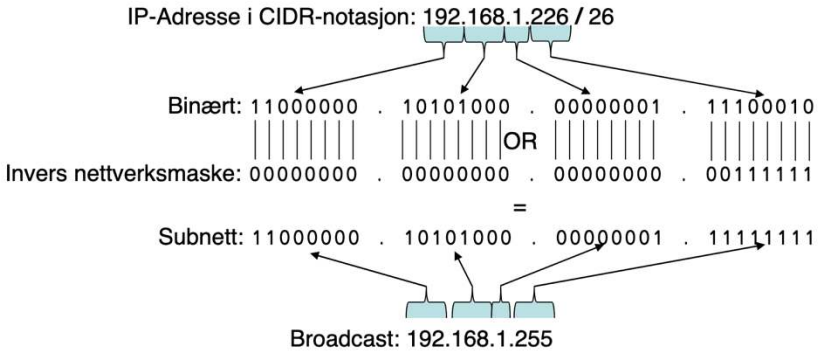
- En maskin i nettet har
  IP 192.168.1.5 = 11000000.10101000.00000001.00000101
- Nettverksmasken er
  255.255.255.0 =.11111111.11111111.11111111.00000000
- For å finne subnettadressen til maskinen må du gjøre en bitvis AND-operasjon mellom
- IP-adressen og nettverksmasken.
  11000000.10101000.00000001.00000000 = 192.168.1.0
- Dette er den første IP-adressen i subnettet og brukes til å identifisere subnettet.
- Eksempel: subnettadresse fra IP / nettverksmaske:

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)



Kringkasting (send til alle) (IP/INTERNETT) (LINKLAGET)

- En melding som sendes ut på en spesiell adresse.
- Leveres til alle enheter som er koblet på samme LAN (nettverk):
  - Linklaget (MAC): FF:FF:FF:FF:FF:FF
  - IP/Internett: 255.255.255.255
  - For en maskin på et subnett, finner du kringkastingsadressen ved å gjøre en bitvis OR-operasjon mellom maskinens IP-adresse og bit komplement (bitvis invers) av nettverksmasken.
    Eks: IP-adresse 192.168.1.5 nettverksmaske: 255.255.255.0
    (192.168.1.5) OR (0.0.0.255) = 192.168.1.255
- Eksempel: kringkastingsadresse fra IP / nettverksmaske
- Broadcast adressen og adressen til rutern er reservert (må trekke fra to)



Kunne regne seg frem til en nettmaske og broadcast adresser.

- Nettverksmaske:
  - er en 32-bit netmask/subnet mask som maskerer og separerer en IP-adresse inn i network-adressen og host-adressen (hver IP-adresse består av disse to komponentene).
  - Subnettmasken lages ved å sette alle network bits til "1" og alle host bits til "0".
  - Regne ut subnettet fra en IP + nettverksmaske: For å finne subnettadressen til maskinen må man gjøre en bitvis AND operasjon mellom IP-adressen og nettverksmasken.

Eks.

IP-adress til en maskin: 192.168.169.220
Binært: 11000000.10101000.10101001.11011100
Subnettmaske/CDIR: 255.255.240.0
Binært: 11111111.11111111.1111000000.00000000

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Vi kjører en AND-operasjon:

11000000.10101000.10101001.11011100
11111111.11111111.1111000000.00000000
= 11000000.10101000.10100000.00000000
= 192.168.160.0

Svar:

Subnettet til maskinen over i CIDR-notasjonen blir altså 192.168.160.0/20.
Tallet 20 på slutten her er CIDR verdien som er lik antallet positive bits i en 32 bit adresse (fra h til v) til en subnettmaske som i denne f eks.
11111111.11111111.1111000000.00000000.
Følgende adresse ville hatt en CIDR på 24: 11111111.11111111.11111111.00000000.

For utregning av broadcast-adressen (kringkastings-adressen) til et subnett må man gjøre en bitvis OR-operasjon mellom maskinens IP-adresse og bit komplement (bitvis invers) av nettverksmasken.

Eks.

Vi bruker samme adresser som over.
Subnett-adressen vi regnet ut (binary network):
11000000.10101000.10100000.00000000
Subnettmaske/CIDR Invertert: 00000000.00000000.00001111.11111111

Vi gjør en OR-operasjon:

11000000.10101000.10100000.00000000
/11000000.10101000.10101001.11011100??
00000000.00000000.00001111.11111111
= 11000000.10101000.10101111.11111111
= 192.168.175.255

Svar:

Broadcast/kringkastingsadressen til IP-adressen i CIDR-notasjon er 192.168.175.255/20.

Kringkastingsdomener

- Om det er for mange enheter så kan det bli problemer. ARP og DHCP-forespørsler går til alle, dette vil for et stort domene stjele kapasitet som burde brukes til å overføre data.
- Avanserte switcher kan fikse dette problemet. Burde unngå for store kringkastingsdomener, heller dele opp i subnett.
- Man kan skille mellom de ulike tjenestene man kan kjøre på en datamaskin ved bruk av porter. Notasjon IP : port

ARP-koblingen mellom nettverk og IP

- Nettverkskortene har en 6 byte lang MAC-adresse som brukes til å identifisere maskinen innenfor et kringkastingsdomene. For at IP skal fungere må avsenderen vite hvilken MAC-adresse pakken skal sendes til. ARP kobler IP (nettverkslaget) og MAC (linklaget). ARP kan også brukes til å finne duplikate IP-adresser, dersom to maskiner svarer.
- ARP-forespørsel:
  - Sender til alle: Hvem har denne adressen?
  - Den som har den svarer
  - Den som spurte lagrer info i en liten tabell, dette kalles en ARP-tabell/ARP-cache
- Hvis kringkastingsdomenet er for stort når ARP sender ut en melding kan det stjele kapasitet som ellers burde vært brukt til å overføre data (DHCP har samme problem og avanserte switcher kan filtrere ARP og DHCP for å beskytte mot overbelastning fra kringkastet trafikk).

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- ARP-tabellen lagrer informasjon om hvilke IP-adresser som assosieres med hvilke MAC-adresser. Når en pakke skal sendes til en IP-adresse sjekker systemet først om MAC-adressen finnes i tabellen; hvis ja er det ikke nødvendig og bruke ARP. Hvis nei sendes pakken til nettverket med ARP-protokollen som spør hvem som har den aktuelle adressen - maskinen med denne IP-adressen svarer med en ARP-pakke som også inneholder MAC-adressen som kan ta imot pakker for den IP'n.

Én IP-adresse – mange porter

Hvordan kan en IP adresse brukes til mange tjenester?
Adressering i transportlaget.



IP: 192.168.1.5

Port	Tjeneste
0	Reservert
1	tcpmux
…	
22	SSH
…	
25	SMTP
…	
1024-49151	Brukerporter
49152-65535	Dynamisk / privat

Transportprotokollene (**UDP, TCP**) implementerer "porter" som muliggjør totalt 65535 samtidige forbindelser på én IP-adresse

IP-adresse

- IPv4 deler opp en IP-adresse i fire oktetter. For å definere et subnett må man ha en nettverksmaske.
- I nettverksmasken kan bits satt til 0 varieres for å angi IP-adresser i subnettet, mens bits satt til 1 angir delen av IP-adressen som definerer hvilket nettverk vertene tilhører.
- Nettverksmasken består av en sammenhengende serie "1" deretter en sammenhengende serie "0". Det er to måter for å notere omfanget av et subnett.
- Ved punktnotasjon 192.168.1.0, da må man oppgi nettverksmaske. CIDR-notasjon (Classless Inter-Domain Routing) er eksempelvis 192.168.1.0/24 har vanlig punktnotasjon først og tallet etter skråstreken angir hvor mange bits nettverksmasken består av.

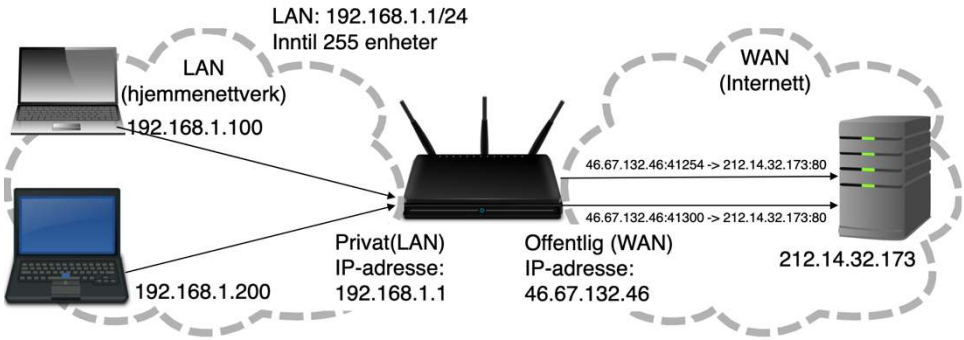
En IP-adresse - mange porter, hvordan fungerer dette?

- Transportprotokollene UDP og TCP implementerer 'porter' som muliggjør totalt 65535 (per UDP/TCP) samtidige forbindelser på en IP-adresse. Informasjon som sendes over nettet kan aksepteres av mottaker-maskinen ved å bruke TCP- eller UDP-porter.
- Når et program på en datamaskin sender/mottar data over nettet, sendes denne data til en IP-adresse og en spesifikk port på den "eksterne maskinen", og tar imot dataen på en (vanligvis random) port på lokal/mottakermaskinen.
- Ved bruk av TCP-protokollen tilkobles en TCP-port, og vice versa for UDP-protokollen
- Når en applikasjon kobler seg til en spesifikk port vil den ikke kunne brukes av andre applikasjoner. "First come, first served".

NAT – Network Address Translation

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)



- Problem: tom for IP-adresser
- Måte å oversette private adresser til offentlige IP-adresser på internett
- Bruker portnr for å definere en unik forbindelse med ruteren

Kilde IP	Mottaker	Oversatt adresse
192.168.1.100	212.14.32.173:80	46.67.132.46:41254
192.168.1.200	212.14.32.173:80	46.67.132.46:41300

NAT - Network Address Translation

- NAT-routeren oversetter datatrafikk inn/ut av det private nettverket.
- Det finnes 232 IP-adresser, og vi begynner å få for få. For å unngå problemet og fremme sikkerhet så kan vi bruke adressene flere ganger, dette gjennom NAT.
- NAT utnytter at transportlaget tilbyr porter som en funksjon. Routeren gir en maskin på det lokale nettet en bestemt port, og lagrer dette i en tabell som oversetter den lokale adressen til sin eksterne adresse. Når maskinen vil snakke med en annen maskin på internettet så vil den andre maskinen svare med IP + porten som er gitt for maskinen og routeren kan finne ut hvilken maskin som den skal sende svaret til.
- Dette innebærer at en unik IP-adresse er nok for flere datamaskiner.
- Ulemper med NAT: Gir en ekstra kompleksitet, og nye forbindelser må komme fra innsiden av NAT-nettverket.

DHCP - Automatisk tildeling av IP-adresser

Prosedyre:

1. Det kommer en ny maskin på nettverket. Maskinen sier til alle (sender en kringkastingsadresse): finnes det en maskin (DHCP-server) med myndighet til å dele ut IP-adresser på dette subnettet?
2. DHCP-tjener (hjemmeruter) sier til alle: 192.168.1.5 er tilgjengelig. Her har du også en liste over andre viktige adresser som gateway og DNS.
3. Ny maskin: Tar i mot adressen.
4. DHCP-tjener: skriver opp at 192.168.1.5 er i bruk av den nye maskinen i en periode på 24 timer.

Den nye maskinen må altså spørre igjen innen 24 timer om å få beholde IP-adressen hvis ikke blir den sett på som ledig.

Discovery > Offer > Request > Acknowledge

Ruting i Internett

- Mål: videresende en datapakke slik at den til slutt når måladressen sin.
- ISP (Internet Service Provider) er internettleverandørene som gir deg tilgang til internett.
- ISP-peering er når ISPer inngår avtaler om å videresende hverandres trafikk. Dette fører til at økonomiske prinsipper påvirker trafikkflyt. - BGP (Boarder gateway protocol) er protokollen som ISPene har for ruting mellom seg.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

5. HTTP-protokollen

World Wide Web (www): HTTP-protokollen



- HTTP: bruker TCP som transport:
  - Klienten oppretter en TCP-forbindelse (socket) til tjeneren, port 80
  - Tjeneren godtar TCP-forbindelsen fra klienten
  - HTTP-meldinger (protokollmeldinger på applikasjonslaget) utveksles mellom nettleseren (HTTP-klient) og Webtjeneren (HTTP-tjener)
  - TCP-forbindelsen lukkes
- HTTP er "stateless"
  - Tjeneren sparer ikke på tilstandsinformasjon om tidligere forespørsler
- Protokoller som sparer på "tilstand" er komplekse!
  - Tilstanden må vedlikeholdes
  - Om en tjener eller klient "kræsjer", kan tilstanden bli ulik mellom dem. Da må den gjennoprettes.

HTTP-streaming

- Dynamisk, adaptiv streaming over HTTP (DASH)
  - Bruker TCP som transportprotokoll.
  - Måten man gjør dette på er ved å dele videoenopp i biter på cirka 2-10 sek.
  - Hver av de bitene er en film som kan spilles av helt uavhengig.
  - Det finnes også flere versjoner av hver bit, en med god kvalitet, en med middels kvalitet og en med dårlig kvalitet.
  - Måten man sender dataen på er ved at videospilleren fungerer som en nettleser.
  - Den spør serveren om den kan levere et segment og serveren sender med lavest kvalitet og mottaker mottar.
  - Dersom dette går fint, ingen pakker blir mistet ol. så sender serveren gradvis med bedre og bedre kvalitet, helt til optimal kvalitet er oppnådd.
- HTTP-protokollen (hypertext transport protocol)
  - Er en applikasjonlagsprotokoll, som inneholder meldingskoder og metadata som trengs for å overføre websider oppå det som er av TCP-headere og IP-headere.
  - Har en klient-tjener modell
  - Bruker TCP som transport
  - HTTP har ingen tilstandsinformasjon, dette er en fordel for protokoller som trenger å være effektive

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Persistente og ikke-persistente forbindelser:

- Ikke-persistent:
  - Tjeneren leser forespørselen, svarer, lukker TCP-forbindelsen
  - Hver overføring lider av TCP sin gradvise økning i senderate (slow start)
  - Mange nettlesere åpner flere parallelle forbindelser for å prøve å ta mer kapasitet ogfå ting til å fortere, omgå et designproblem
- Persistent:
  - Standard for HTTP/1.1
  - Over samme TCP-forbindelse: tjeneren leser forespørselen, svarer, leser ny forespørsel
  - Klienten sender forespørsler for alle de objektene den trenger så fort den mottar hoveddokumentet (HTML)
  - Færre RTTer, mindre slow start - Persistent med pipelining: spør etter mange objekter på én gang (enda færre RTTer)

Cookies:

- Tar vare på tilstand
- Lager spesialtilpasset innhold for klienten, basert på lagrede HTTP-forespørsler
- Vise spesialtilpasset reklame

E-post

- Simple Mail Transfer Protocol (SMTP)
- Bruker TCP (alt må komme frem)
- 3 faser: - Håndtrykk
- Overføring av beskjeder
- Avslutning
- Ascii 7-bit

CDN - Content Delivery Network:

- Kopier av data som man ønsker å dele/levere til folk som er nærme der brukeren er.
- For eksempel hos internettleverandøren eller i sky-tjenester.
- Dette gir kortere RTT-tid og forhindrer også overbelastning på tjeneren.
- Minus: koster ekstra maskinvare og lagringsplass.

Proxy-cache:

- Er en forward-proxy som står nære klientene og vil da levere til klientene.
- En “Reverse proxy” står nær tjenerne og mellomlagrer data fra en eller flere tjenere slik at klienten slipper å gå helt til kilden.
- Fungerer som en front for klienten.
- Dette bruker mye på Wi-Fi på fly.

Head of line blocking

- Mottaker må vente på et forsinket segment før mer data kan leveres
- Tapte pakker må sendes på nytt
- Dette fører til en forsinkelse

Videostreaming - utfordringer:

- Konstant strøm av data (til avspilling slutter)
- Nedlastning: hele innholdet lastes ned til en lokal maskin
- Streaming: Ikke lokal lagring, sparer nettverksressurser
- UDP eller TCP?

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

    - UDP kan fungere helt fint, fungerer raskt og merker det ikke om det mangler noen pakker.
    - Mer TCP i det siste
- Nettverksutfordringer: - Forsinkelser, tap, variasjoner i leveringstid (“jitter”)
    - jitter kompensasjon
    - tapskompensasjon

En pakkes vei gjennom nettet

1. Du kobler maskinen til ditt lokale nettverk.
2. Linklaget mottar bits fra det fysiske laget og ser om det kan snakke med bitsene som kommer.
3. Om den kan det så vil den få en IP-adresse fra en tjener. Så kan du skrive inn et domenenavn i nettleseren din.
4. Da vil applikasjonslaget lage en HTTP-header. Dette er det som trengs for at maskinen din skal kunne snakke med den andre maskinen. HTTP er en vanlig applikasjonsprotokoll.
5. Denne protokollen inneholder en get-forespørsel som sier lever f.eks. forsiden til facebook til meg. Så vil den spørre nedover i laget.
6. Det sendes ut en ARP-forespørsel og man kan lage en linklag-header. Du har en datapakke, som legges på en http-header, så legges det på en TCP-header som sier at den jeg ønsker å kontakte på andre siden er på port 80, i tillegg til en del annen informasjon. Utenpå TCP-headeren legges det på IP-header som inneholder adressen til mottakeren.
7. Kjører DNS og får IP-adressen
8. Utenpå IP-headeren får du et linklag som er fylt ut av ARP og som har en MAC-adresse til ruteren din.
9. Dette legges ned i det fysiske laget som kjører ut en serie med bits som kan forstås i den andre enden.
10. Datapakken går fra maskinen til routeren din, der blir ethernet headeren tatt av ogdet settes på en ny som sendes til ISP.
11. Denne sender videre ut i fra informasjonen som er med.
12. Til slutt fjernes alle headerne og det er bare nyttelasten som leveres til web-tjeneren som leverer nettsiden.

6. Kryptering

Kryptografi

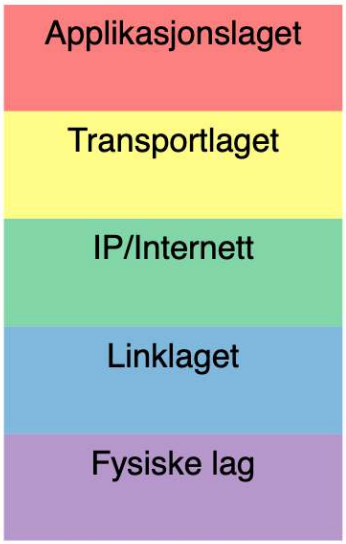
- Hovedideen med kryptering er at dataen som skal leses gjøres uleselig for datamaskiner og mennesker.
- Så om to parter skal kommunisere med hverandre uten at andre skal kunne avlytte og forstå dataen som blir sendt, kan kommunikasjonen krypteres.
- For at partene i kommunikasjonen skal forstå hverandre, så må man kunne dekryptere dataen også.
- Dette gjør vi ved å bruke nøkler på ulike måter.
- Forskjellige teknikker for kryptering:
    - Hemmelig nøkkel (symmetrisk) kryptering.
    - Offentlig nøkkel (asymmetrisk) kryptering.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Hash-algoritmer.

Kryptering / sikkerhet i nettverket

	Secure Sockets Layer – kryptering for ende-til-ende Applikasjoner – f.eks nettbank eller butikker. F.eks. tcpcrypt– har som mål at alle TCP-forbindelser som settes opp skal være kryptert VPN (IPSEC etc.) – kobler to subnett sammen så det fungerer som ett LAN selv om de er fysisk adskilt Kryptering på flere lag gjør det vanskeligere for uvedkomne å lytte til kommunikasjonen. Adressen (avsender / mottakeren) er vanskelig å kryptere, da routere må vite hvor pakken skal leveres.

Kryptering/sikkerhet i lagene

- Viktig å sikre informasjon som sendes i leddene på internett.
- Det er mulig å sikre informasjon i alle lagene.
  - VPN (IPSEC) kobler to subnett sammen så det fungerer som et LAN selv om de er fysisk adskilt (oppretter kryptert forbindelse og blir enige om en nøkkel som begge utveksler på en sikker måte)
  - TCPcrypt har som mål at alle TCP-forbindelser som settes opp skal være kryptert.
  - SSL (Secure sockets layer) er kryptering fra ende til ende

Symmetrisk kryptering:

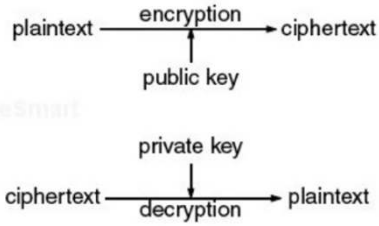
- En felles nøkkel brukes til kryptering og dekryptering.
- Man må ha delt denne nøkkelen på forhånd slik at både sender og mottaker har nøkkelen.
- Senderen bruker nøkkelen til å kryptere meldingen.
- Meldingen blir sendt.
- Mottakeren bruker den samme nøkkelen til å dekryptere meldingen.

Asymmetrisk kryptering:

- En nøkkel brukes til kryptering og en annen nøkkel brukes til dekryptering.
- Disse to nøklene opptrer i par.
- Når to parter skal kommunisere så har senderen og mottakeren 2 nøkler hver– en offentlig og en privat. Disse nøklene opptrer altså i par.
- Det betyr at en melding som krypteres med offentlige nøkkelen kan KUN bli dekryptert med den tilhørende private nøkkelen (og omvendt).
- I kommunikasjon mellom to parter vil asymmetrisk kryptering fungere slik:
  - Senderen vil sende en kryptert melding til en mottaker.
  - Da bruker senderen mottakeren sin offentlige nøkkel til å kryptere.
  - Når mottakeren skal dekryptere meldingen så bruker mottakeren sin egen private nøkkel.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- I digitale signaturer så bruker jeg min egen private nøkkel for å kryptere.
- Når den digitale signaturen skal bli verifisert, så brukes min offentlige nøkkel til dette.



Hashing:

- Brukes til å kryptere en melding.
- Man trenger bare en sjekksumalgoritme som tar inn teksten som skal krypteres og genererer en sjekksum for denne.
- Sjekksummen inneholder helt tilfeldige strenger av tall og bokstaver.
- Når vi har en sjekksum, kan vi ikke rekonstruere den originale meldinga.
- Vi kan ikke endre en melding uten at sjekksummen blir endret.
- Ingen beskjeder har ikke samme sjekksum.

Vi kan generere en sjekksum av en fil basert på SHA256 algoritmen og teste det ut i terminalen.

1. Vi lager en fil test.txt og skriver noe i den.
2. Vi genererer en sjekksum.
3. Vi gjør en endring i filen.
4. Vi genererer en ny sjekksum.
5. Den gamle sjekksummen og den nye sjekksummen er ikke like!!! Dette er den grunnleggende egenskapen ved sjekksumalgoritmer.
6. Vi genererer en ny sjekksum for denne filen og vi ser at sjekksummen er lik som før!

Hvordan kan sjekksumalgoritmer bidra til å sikre dataintegritet?

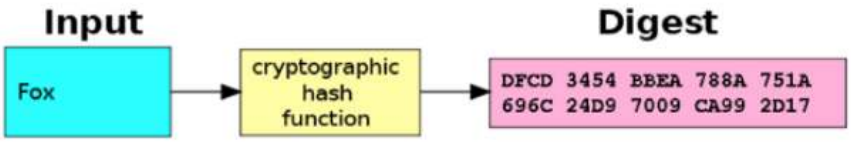
- Man kan sammenligne sjekksum av en datafil opp mot en kjent sjekksum av denne fila.
- Er den lik?
- Isåfall er fila ikke endret mellom første og andregangskjøring av sjekksum.

- Når kan sjekksum være nyttig å bruke?

- Sjekksumalgoritmer brukes ofte ved deling av programvare (f.eks. nedlasting fra nett).
- Til- byderen av programvare genererer en sjekksum av orignalen (ofte et filarkiv (f.eks. .zip-fil), som publiseres sammen med programvaren.
- Brukere som laster ned programvare kan enkelt selv kjøre en sjekksumalgoritme etter nedlasting, og kan med det finne ut om programvaren det den utgir seg for å være, eller er den kanskje endret etter at sjekksummen ble generert?
- Vil også kunne avdekke hvorvidt en nedlasting har gått greit eller ikke (pakketap?)

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)



LMC og Representasjon

1. Prosessoren

Oppbygging av vanlig datamaskin:

- Dette er en enkel modell for hvordan en datamaskin er bygd opp.
- Den består av følgende komponenter:
- Input: Det vi gjør på maskinen, f.eks. tastetrykk, museklikk, touch etc.
- Output: Maskinen «reagerer» på vår input og gir ut et resultat, f.eks. vise noe på skjermen.

CPU («Central Processing Unit»)

- Den sentrale delen i en datamaskin.
- Den inneholder:
  - ALU («Arithmetic Logic Unit»)
    - Til aritmetiske og logiske beregninger
    - ALU-en (regneenheten) i LMC kan addere og subtrahere heltall
    - ALU-en i en moderne datamaskin kan alle vanlige regnearter for heltall av ulik lengde, og for flyt-tall.
- Registre
  - Inne i CPU-en finnes noen få lagerceller for data som CPU-en trenger hurtig tilgang til.
- Disse kalles registre.
- LMC har fire registre:
  - Accumulatorer til beregninger; svaret kommer hit.
  - Program counter(alias location counter) angir hvilken instruksjon i minnet som neste gang skal utføres.
  - Instructionregister inneholder koden til den instruksjonen som utføres.
  - Addressregister inneholder adressedelen av den instruksjonen som utføres.
- Moderne prosessorer har fra 20 til et par hundre registre
- Kontrollogikk til å utføre instruksjonene

Minne (RAM)

- LMC har 100 celler i minnet
- Hver celle kan lagre verdier –999–+999
- x86-64 kan ha inntil $2^{64} \approx 1,8 \cdot 10^{19}$ celler; dvs ca 17 000 000 TB («terabyte») eller 16 EB («exabyte»)
- Ingen maskiner i dag har så mye installert

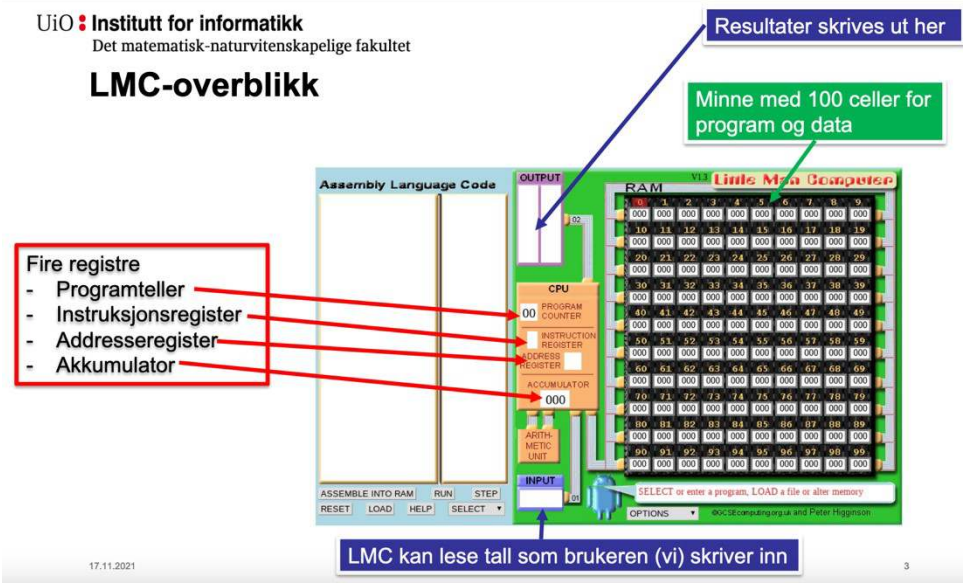
Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Hver celle inneholder én byte (= 8 bit)
- I tillegg har x86-64 støtte for virtuelt minne

2. LMC

Little Man Computer:

- LMC er en forenklet versjon av en datamaskin.
- Den inneholder de samme komponentene, men disse er svært enkle i forhold til en ekte datamaskin.
- Disse er:
- Input: Tall
- Output: Tall/bokstaver
- CPU:
    - ALU: Utfører beregninger på inputtallene, men kun + og – og ingen logikk.
    - Registre:
    - Har 4 registre, der hver av dem holder på en spesifikk verdi.
    - Program counter: hvilken instruksjon i minnet som skal utføres neste gang
    - Instruction register: koden til instruksjonen som skal utføres (5 – LDA)
    - Adress register: adressedelen til koden til instruksjonen (99 – minne 99)
    - Accumulator: svaret kommer hit



Tips og anna til LMC:

- Etter du har skrevet kode, se inn i RAM og se om minneadressene har blitt fylt ut.
- Hvis du ser at kun noen må minneadresser har blitt fylt ut og koden din er lang, så har du skrivefeil i koden.
- Husk å RESET etter hver RUN (etter du har kjørt programmet) slik at CPUen ikke husker verdien fra forrige kjøring og forstyrrer det nye resultatet.
- Husk å ha med HLT (stopp) i koden slik at vi signaliserer at koden skal være ferdig å kjøre.
- Dere skal både klare å kode med «tall» og «bokstaver» (sånn som i tabellen.)

Alle instruksjonene i LMC

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Kode	Navn	Beskrivelse
0xx	HLT	Stopper eksekveringen
1xx	ADD	Adderer verdien i angitt minnelokasjon med akkumulatoren
2xx	SUB	Subtraherer verdien i angitt minnelokasjon fra akkumulatoren
3xx	STO	Lagrer akkumulatoren i angitt minnelokasjon
4xx	-	Ikke i bruk
5xx	LDA	Henter verdi fra minnet til akkumulatoren
6xx	BRA	Hopper til angitt adresse
7xx	BRZ	Hopper til angitt adresse hvis akkumulatoren er 0
8xx	BRP	Hopper til angitt adresse hvis akkumulatoren er ≥ 0
901	INP	Leser verdi fra input, og legger svaret i akkumulatoren
902	OUT	Skriver ut verdien i akkumulatoren
922	OTC	Skriver ut ASCII-tegn (ikke i boka)

Ascii:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

3. Assemblerkode

Ord og begreper:

- Maskinkode er den koden prosessoren utfører
- Assemblerkode er en tekstlig form av maskinkoden
- Registre er lagerlokasjoner helt tett på prosessoren; de er på 64 bit (dvs 8 byte)
- Flagg er 1-bits lagre som settes automatisk av prosessoren. Flagg er 0 eller 1 avhengig av hva som skjer i prosessoren.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Assemblerkode

- Direkte oversettelse numerisk <---> tekstlig representasjon
- LDA = Load akkumulator
- DAT = Setter av plass i minne

START	LDA	H
	OTC	
	LDA	e
	OTC	
	LDA	i
	OTC	
END	HLT	
		72 // = 'H'
H	DAT	101 // = 'e'
e	DAT	105 // = 'i'
i	DAT	

- H og H kune stått hva som helst men måtte stått det samme, kunne stått x,y,z
- Spesifikasjonen DAT (= «data») er ingen instruksjon, men en angivelse av at det skal settes av en celle med en gitt startverdi

KODEEKSEMPEL:

Oppgave: Skal lese inn 2 tall og multiplisere dem sammen.

	INP		Leser input
	STA a		Lagrer inputen som a
	INP		Ny input
	STA b		Lagrer inputen i variabelen b – fortsatt i akkumulatoren
loekke	LDA	tot	Starter løkke og henter tallet i tot fra minnet
	ADD	a	Add a til a b ganger = multiplikasjon med addisjon
	STA	tot	Lagrer tot+a i totalen
a	LDA	b	Henter b
b	SUB	en	Har nå adda sammen a en gang
tot	STA	b	Lagrer b sin nye verdi til neste runde
en	BRP	loekke	Hvis b >= 0 så er vi fortsetter vi løkka
d			
o	LDA	tot	Henter ut totalen
n	SUB	a	Minus b fordi vi har gjort de en gang for mye ovenfor (b=0 nono)
e	STA	tot	Lagrer totalen
	OUT		Printer
	LDA	d	
	OTC		
	LDA	o	
	OTC		
	LDA	n	
	OTC		
	LDA	e	Stopper
	HLT		
			Huske å definere variabelen a med startverdi 0
	DAT	0	Definere variabel
	DAT	0	Definere konstant 1
	DAT	0	
	DAT	1	

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

	DAT DAT DAT DAT	68 111 110 101	

4. Tallsystemer

Digital representasjon – alt er bit!

- Definisjon: hvordan vi lagrer og representerer informasjon
- I dag: Hvordan representerer og lagrer vi
  - Tall, Tekst, Bilder, Lyd - som bit i datamaskinen?
  - Bit - er den grunnleggende enheten for digital informasjon, 0 og 1, av og på
  - Byte - er en enhet for mengde av elektronisk informasjon, 8 bits, blokker av bytes
  - Nibble - halv byte

Ord og begreper:

- Binærtall: tall med grunntall 2 - Notasjon: 1001_2
- Oktaltall: tall med grunntall 8
- Desimaltall: tall med grunntall 10
- Hextall: tall med grunntall 16 (og sifre 0-9 og A-F)
- Bit: verdi 0 eller 1 - Fortegnsbit: øverste bit som angir + eller -
- Byte: samling av 8 bit - ASCII: 7-bits tegnsett (ett tegn per byte)
- ISO Latin-1: 8-bits tegnsett (ett tegn per byte)
- Unicode: universelt tegnsett (4 byte per tegn)
- UTF-8: lagring av Unicode med 1-4 byte per tegn

Hvordan fungerer tallsystemer egentlig?

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

  1 10 100 1000 osv.

  Dette tilsvarer

  10^0 10^1 10^2 10^3 osv.

  Notasjonen an(«a i n-te») betyr a * a * ... * a, n ganger.
- I det binære tallsystemet har posisjonen tilsvarende vekt

  1 2 4 8 osv.

  Dette tilsvarer

  2^0 2^1 2^2 2^3 osv.

  Slik vi representerer tall i dataverden (0 og 1)

Binær representasjon:

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

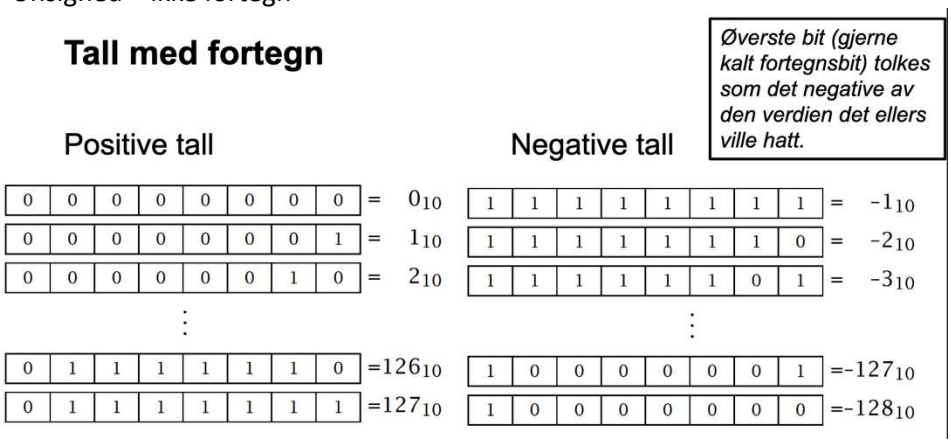
Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

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

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

Tall med fortegn

- Signed = har fortegn
- Unsigned = ikke fortegn



- Det største positive tallet blir da +127
- Det største negative er -128 + 0
- Det første negative tallet er -128 + 127 = -1

Desimalt til binært

- I minnet er alt bare bit
- Det er opp til oss, som programmerere, hvordan disse bitene skal tolkes
  - Positive/negative tall, Flyttall, Tekst, Bilder, Lyd, Kode, Etc
- Hvordan kan vi enkelt regne om 2510 til binært?

  Løsningen er å dele på 2 og se på resten:

Verdi	Rest
25	1
12	0
6	0
3	1
1	1

Svaret leser vi nedenfra: $25_{10} = 11001_2$

Tilbake igjen:

$1 * 2^4 + 1 * 2^3 + 0 * 2^2 + 0 * 2^1 + 1 * 2^0$
$= 16 + 8 + 0 + 0 + 1$
$= 25$

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Heksadesimal notasjon

- Hvor data er lagret i minne
- Det er lett å gjøre feil når man jobber med binære tall:
  $1\ 000\ 000_{10} = 11110100001001000000_2$
- Heksadesimal notasjon gjør det lettere!
- Et tallsystem med 16 forskjellige siffer; 0-9 og A-F representerer verdiene $0\text{-}15_{10}$:

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

- Da kan vi i stedet skrive $1\ 000\ 000_{10} = F4240_{16} =$ 0xF4240

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

Rad 1: Desimal (base 10) $1\ 000\ 000_{10}$ =
Rad 2: Binær (base 2) 11110100001001000000_2 =
Rad 3: Heksadesimal (base 16) 0xF 4 2 4 0 = $F4240_{16}$

Generell omregning mellom desimaltall og andre baser:

Et tall: 1 1 1 1 1
Posisjon: 4 3 2 1 0
Utregning: $1 \times \text{grunntall}^4 + 1 \times \text{grunntall}^3 + 1 \times \text{grunntall}^2 + 1 \times \text{grunntall}^1 + 1 \times \text{grunntall}^0$
Eks:
Base 4: 3 0 2 1 3
Posisjon: 4 3 2 1 0
Utregning: $3 \times 4^4 + 0 \times 4^3 + 2 \times 4^2 + 1 \times 4^1 + 3 \times 4^0$
$3 \times 256 + 0 \times 64 + 2 \times 16 + 1 \times 4 + 3 \times 1$
$768 + 0 + 32 + 4 + 3 = 807$
$302134 = 80710$

ASCII

- Enkelt å sjekke om det er et siffer eller en bokstav
- Enkelt å konvertere fra liten bokstav til stor bokstav, ligger akkurat 32 bit unna hverandre
- Trenger 7 bit for å lagre alle verdiene, under 128 ulike verdier
- Inneholder kontrolltegn/styretegn, brukt til å jobbe med fjernskrivere
- MINUS: mangler en del bokstaver og tegn, blant annet æøå

ISO 8859-1

- Halvparten av tabellen (128 verdier) er ASCII - UNICODE
- Tegnkoding som omfatter alle skriftspråk i verden
- Plass til drøyt 1 000 000 tegn
  - Foreløpig er 136 755 tegn definert
- Mer og mer programvare (som Java og Python) støtter UNICODE
- 1 000 000 tegn krever 24 bit = 4 byte på alle tegn
- UTF-8 er en måte å skrive UNICODE på som er komprimert (UNICODE eralle tegnene)
- Bruker fra 1 til 4 byte på å lagre et tegn

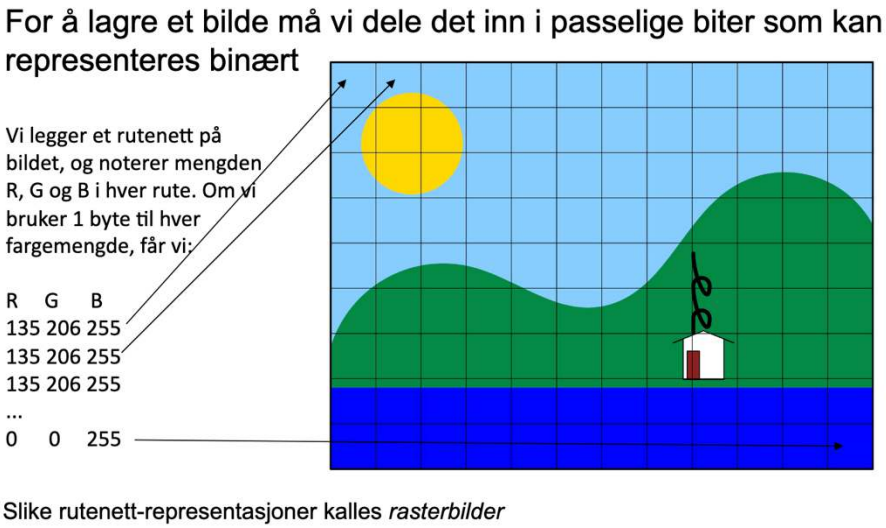
Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Trenger 2 byte til æ, ø, å
- Kommer an på tegnet hvor mange byte

5. Bilder

Bilder

- PNG: bildeformat for rasterbilder
- JPEG: bildeformat for fotografier
- Hvert bildepunkt(piksel) består av tre farger: rød, grønn, blå som kan lyse sterkt eller svakt
  - Hvitt -> 1
  - Svart -> 0
- (Utskrift på papir: CMYK = Cyan, Magenta, Yellow, blacK)
- Ofte en byte til hver farge, dvs. 3 byte per piksel
  - 135 206 255
  - 12 x 9 piksler = 324 byte
  - 348 x 257 piksler = 363 682 byte
- Kvalitet kontra plass



Slike rutenett-representasjoner kalles *rasterbilder*

Redusere plass:

- Lage en fargetabell for å se hva fargene inneholder, trenger kanskje ikke 3 byte per piksel
- Run lenght-koding
  - Lagrer fargen og hvor mange slike piksler det er på rad
  - PNG og GIF bruker denne teknikken

Fotografi litt annerledes:

- Ikke brå overganger
- Ekte rasterbilde tar stor plass
- JPEG komprimerer bildet slik at det tar mindre plass, men taper informasjon(fjerner unødvendig informasjon)
  - Mennesker kan bare skjelne et begrenset antall nyanser
  - Ser hovedsakelig på overganger på himmel og hvordan huden ser ut
  - Kan ikke få tilbake til det opprinnelige, info er tapt

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Vektorgrafikk:

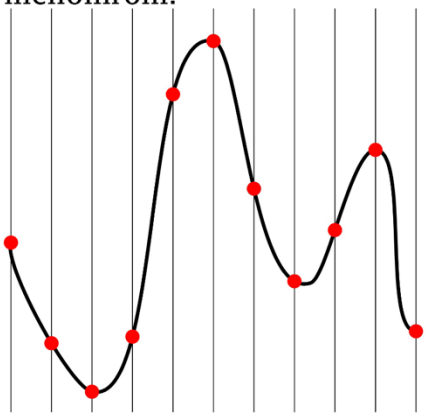
- Lagrer bilde som geometriske elementer (rette linjer, buet linje)
  - Koder i det grafiske språket, matematisk definert
  - Kan derfor forstørre så mye man vil uten å tape info (geometri fungerer i alle størrelser)
  - Bruke vektorgrafikk der man kan (SVG, ESP, PDF)

6. Lyd

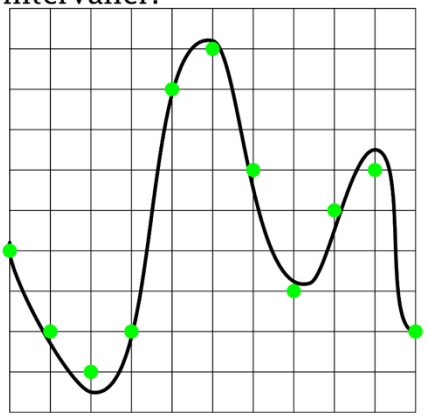
Lyd:

- Lyd er trykkbølger i luft som f.eks. går inn i en mikrofon som sender elektrisk strøm som gjenspeiler lyden.
- En høyttaler mottar dette og kan gjenskape lyden gjennom membranen i høyttaleren
- Måler strømmen med jevne tidsperioder, flerfoldige tusen ganger i sekundet
- Skjer digitalt, må måle noe som er i nærheten av de virkelige bølgene
  - Målingen er bra, men den gjenskapte lyden vil ikke være helt nøyaktig
- Kvalitet avhenger av:
  - Hvor ofte vi måler
  - Hvor mange trinn vi benytter til målingen (hvor tett rastermønster vi har over målingene)
- Dersom det komprimeres vil kvaliteten gå ned
- MP2: lydformat (gammel DAB)
- MP3: lydformat (MP3-spiller, iPod)
- AAC: lydformat (DAB+)

Vi kan lagre lyden ved å måle strømmen med jevne mellomrom:



Men for å lagre digitalt må vi måle styrken i faste intervaller:



CD:

- 74 minutter spilletid med god kvalitet
- 44 100 målinger per sekund
- 2^{16} ? 65 536 intervaller (2 byte)
- 2 kanaler

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- + 37 % feilkorreksjonsdata
- CD må ha plass til 783 MB

Spare plass:

- Lagre bare én kanal og litt informasjon om forskjellen mellom høyre og venstre kanal, i stedet for å lagre all informasjonen
    - Små forskjeller, og man reduserer nesten halvparten av dataen
- Ikke lagre hver måling, men lagre forskjellen
    - Må lagre hele verdien innimellom pga. feil eller spoling
- Mennesker hører ikke over 20 Hz eller under 20 000 Hz
- Hvis vi hører en kraftig lyd med én frekvens hører vi ikke en litt svakere lyd med høyere frekvens.
- Etter å ha hørt en kraftig lyd hører vi dårligere inntil 0,2 sekunder etterpå

Maskinvare

1. Abstraksjonsnivå

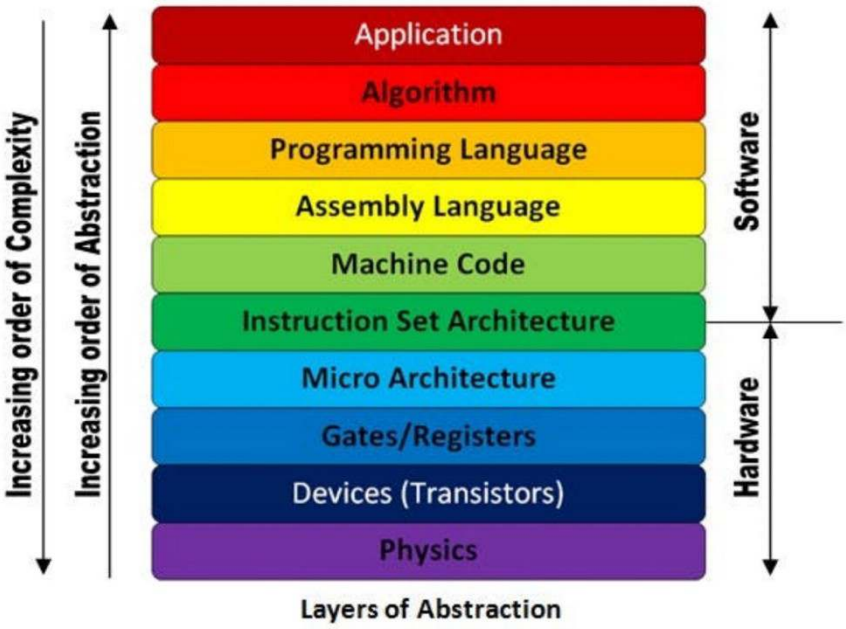
Eksempel: Abstraksjonsnivå: Computergames vs Cars

Høyeste nivå	Spill: Det du trenger å vite er hvordan du kan vinne Passasjer: Det du trenger å vite er at bilen tar deg dit du vil
Veldig høynivå	Game engine: tillater deg å kontrollere hva som er med i spillet Sjåfør: tillater deg å kontrollere hvor du skal
Høynivå	Høynivå programmeringspråk: overstyre kontoll/automasjon Dashbord: overstyre/autopilot av bilen
Middelnivå	Maskinkode: maskinens tolkning av det du programmerer (dekoding av instruksjon) Assemblerkode: en notasjon for et programmeringsspråk som er leselig for mennesker, brukt av en spesifikk maskinarkitektur Bilens chassis og driv: bilens respons til hvordan du bruker styringen
Lavnivå	CPU: kjernen og hjernen av maskinen (*central processing unit)* Under panseret: kjernen og hjernen av bilen

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

	ALU: hjernen av maskinen (*arithmetic logic unit)*
Lavere nivå	ALU: hjernen av maskinen (*arithmetic logic unit)* Porter: en del av maskinen (lavere nivå) Motor: hjernen av bilen
Enda lavere nivå	Logiske porter: utfører operasjoner på med bits. Stempel: bearbeider drivstoff og olje for å skape energy/bevegelse
Enda enda lavere nivå	Transistor: "en bryter" som styrer strømføringen omkring (0-1) Drivstoff: ?
Det laveste nivå	Strøm og spenning Hydrokarbon og oksygen



2. Porter og funksjoner

Porter:

- Et hvilken som helst boolsk funksjonsuttrykk kan implementeres med bare NAND
- Vi foretrekker NAND
- Et hvilket som helst boolsk funksjonsuttrykk kan implementeres med bare NOR

En port:

- En port har alltid en logisk funksjon
- Kan ha matematisk funksjon

En funksjon:

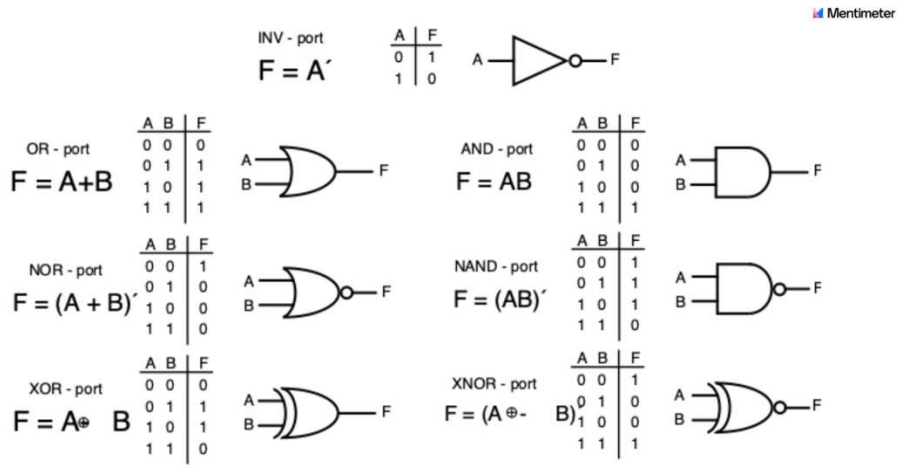
- Kan være en logisk funksjon
- Kan være en matematisk funksjon
- Inneholder en eller flere porter

Nyttige linker:

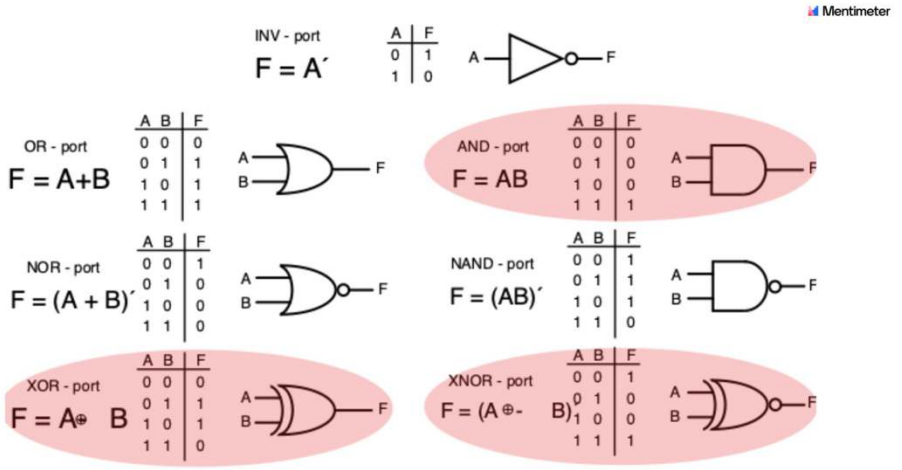
- Få tabell ut i fra utrykk: https://www.dcode.fr/boolean-truth-table
- Lage kretser: https://logic.ly/demo/

Alle logiske port-typer:

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)



Porter som har matematisk funksjon:

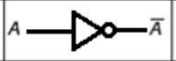


Sannhetsverditabeller

- NOT, OR, AND, XOR, NOR, NAND, and XNOR.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Gate Name	Symbol	Expression	TRUTH TABLE		
			INPUTS		OUTPUT
			B	A	Y
Inverter		$A = \overline{A}$		0	1
				1	0
AND		$A \cdot B = Y$	0	0	0
			0	1	0
			1	0	0
			1	1	1
OR		$A + B = Y$	0	0	0
			0	1	1
			1	0	1
			1	1	1
XOR		$A \oplus B = Y$	0	0	0
			0	1	1
			1	0	1
			1	1	0
NAND		$\overline{A \cdot B} = Y$	0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR		$\overline{A + B} = Y$	0	0	1
			0	1	0
			1	0	0
			1	1	0
XNOR		$\overline{A \oplus B} = Y$	0	0	1
			0	1	0
			1	0	0
			1	1	1

Minterm:

- En funksjon kan være gitt på ”sum av produkt” form
  Eksempel: F = xy + xy’ + x
- Hvert ”produktledd” som inneholder alle variablene kalles minterm
- For to variabler finnes det 22 forskjellige mintermer, for tre variabler finnes det 23 forskjellige mintermer
  xy + x’y + xy’ + x’y’ - Notasjon: m_x

Maksterm:

- En funksjon kan være gitt på ”produkt av sum” form
  Eksempel: F = (x+y)(x+y’)y
- Hvert ”summenledd” som inneholder alle variablene kalles maksterm
- For to variabler finnes det 22 forskjellige makstermer
  (x+y)(x’+y)(x+y’)(x’+y’)
- Notasjon: M_x

Binær adder:

- Vanlige anvendelser: mikroprosessor ALU, Xbox, mikserbord, digitalt kommunikasjonsutstyr, AD-DA omformere osv.
- Basis for addisjon, subtraksjon, multiplikasjon, divisjon og mange andre matematiske operasjoner
- All form for filtrering/signalbehandling

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

Halvadder:

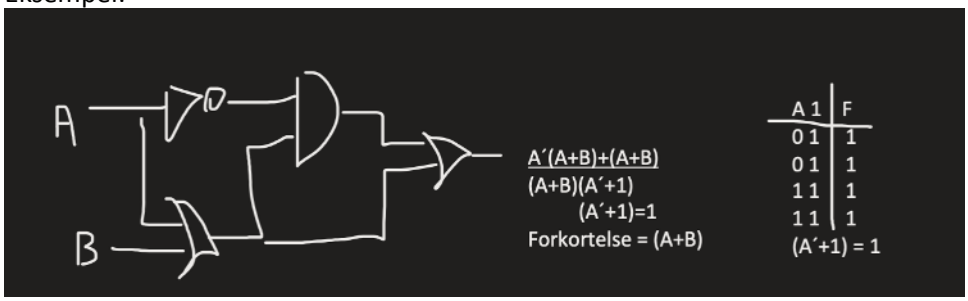
- Ingen mente inn
- To innganger, to utganger
- Adderer de to minste signifikante bittene

Fulladder:

- Adderer sammen bit An og Bn med evt. mente inn
- Tre innganger, to utganger
- Komparatorer:
  - nyttig krets for å sammenligne to bits, er den større eller mindre
  - brukes for å bestemme ting, skal vi gå den veien eller den veien

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\overline{A} = 0$	$A + \overline{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A}\,\overline{B}$

Eksempel:



3. Klokke og synkronisering

Synkronisering

This document is available free of charge on studocu

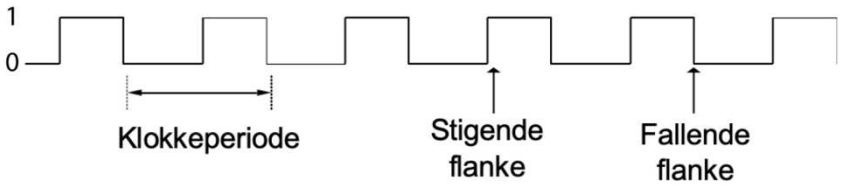
Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Det er en «flow» når instruksjoner kjøres for å kjøre en oppgave.
- Hver instruksjon kjøres akkurat når den skal og er gjort ferdig før neste instruksjon skal kjøres
- Dvs. at CPUen ikke skal vente på at instruksjoner skal kjøres, siden alt arbeidet skal være synkronisert.

Klokkesignal

- CPUen trenger klokkesignaler for å vite når og hvordan den skal kjøre de programmerte funksjonene.
- De vanligste instruksjonene i en klokkesykel er å hente data fra minnet, lagre data i minnet, hoppe til en angitt adresse eller hente en instruksjon.

Et klokkesignal er et digitalt signal som veksler mellom ´0´ og ´1´ med fast takt



Frekvens/klokkefrekvens:

- Antall sykler som CPUen klarer å utføre per sekund.
- Hver gang klokka slår, så henter CPUen en instruksjon og utfører den.
- Antall slike «fetch and execute»-sykler CPUen gjør på et sekund kalles frekvensen.
- Måles i Hz. 1 Hz er en klokkesykel per sekund.
- Jo større frekvens, dess fortere klarer CPUen å prosessere instruksjoner.
- Dette er fordi når frekvensen er større, så klarer den å utføre flere instruksjoner per sekund.
- Mål på hvor god pcen er.

$$frekvens = \frac{1}{klokkeperioden}$$

Eksempel:
Gitt en portforsinkelse på 5 ps for enhver port, hvilken frekvens vil en 1-bits halv-adder kunne operere på?

- En 1-bits halv-adder kan utføre XOR-porten og AND-porten samtidig og operere kretsen på 5ps
- Klokken må være på 5 ps og av 5 ps = klokkeperiode på 10 ps (forutsatt 50%-50% klokkesignal)
  F = 1/klokkeperiode(sekund) picosekund = s * 10^{-12}
  F = 1/10ps = 1/10^{-11} sekund = 10^{11} Hz
  100*10^{9} = 100 GHz i frekvens

En klokkeperiode på 10 ps = 100 GHz i frekvens
Porten kan operere på 100 GHz

4. Minnehirearki

Register:

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Bruksområder: Intern kladdeblokk for CPU'en med en rask aksess til innholdet
- Lagringskapasitet: Integrert i CPU'en, som en del av kretsene, lagret på chip, relativt få

Cache:
- Bruksområder:
  - Ligger i CPU
  - Hurtig mellomlager for både instruksjoner og data for å jevne ut hastighetsforskjellen mellom CPU'en og hurtigminnet
- Lagringskapasitet:
  - Mellomlagrer internt (L1) eller i nærheten (L2, L3) av CPU'en, typiske kapasiteter er fra 10 KiloByte (L1) til 1 MegaByte (L2) og flere MegaByte(L3)

RAM (random access memory):
- Bruksområder:
  - Utenfor CPU'en
  - Buffer mellom ekstern lagringsmedium og CPU'en med rask lese- og skriveaksess
- Lagringskapasitet:
  - Internt på hovedkortet i nærheten av CPU'en, størrelser opptil flere GigaByte
  - Hvor mange RAM man kan ha har en sammenheng med hvor mange bit CPU'en er.
    - 32 bit CPU = 232 adresselinjer i RAM

SSD(solide state-disker)/Flash/Harddisk:
- Bruksområder:
  - Høykapasitetsmedium for program/data. "langt" unna CPU
- Lagringskapasitet:
  - Ekstern eller intern lagringsenhet i maskinen med kapasitet opptil flere TeraByte

Pris, strøm, hastighet og plass
- Stor forskjell på register, cache og RAM i forhold til harddisk er at minnet slettes når strømmen skrus av!
- Antall kjerner vs antall cache (hastighet og lagring) i forhold til framtidige maskiner

Bus:
- Kommunikasjonskanal som brukes i datamaskinen for å frakte informasjon fra en komponent til en annen eller innenfor komponenter.
- For eksempel kan en bus frakte data mellom CPUen og minnet.
- Vi kan tenke på en bus som en vanlig buss.
- En bus frakter flere verdier for eksempel fra CPUen til minnet, på samme måte som bussen frakter flere mennesker fra en plass til en destinasjon.
- En bus består av en samling ledninger der hver ledning sender en bit med informasjon.
- Et stadig dilemma i dataverdenen er at vi vil gjøre ting raskere, og en av hjelpemidlene er altså busser.
- Med busser kan vi frakte flere verdier på samme tid istedenfor å måtte frakte dem en og en.
- Det vil si at vi kan bruke minnet mer effektivt når vi skal lagre eller hente verdier siden vi kan frakte flere verdier av gangen (spoiler alert: minnet er tregt!!!).

Pipeline:
- En metode for å få CPUen til å utføre instruksjoner «raskere».
- Når vi har en CPU som bruker pipeline, vil hver instruksjon deles opp i 4 deler: fetch, decode, execute og write-back.
  - Fetch: Henter instruksjonen som skal utføres.
  - Decode: Dekoder den – finner ut hva som faktisk skal gjøres.
  - Execute: Utfører instruksjonen.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- Write-back: Skriver resultatet til minne.

Når vi bruker pipeline, så vil ulike deler av CPUen gjør hvert av «stegene» over.

- Instruction memory: Henter fram instruksjonen (fetch)
- Control unit: Dekoder instruksjonen (og sender videre informasjon om hva instruksjonen sier (decode)
- ALU: Utfører instruksjonen (execute)
- Data memory: Skriver resultatet til minne (write-back)
- Siden det er ulike deler av CPUen som utfører en del av en instruksjon, vil de andre delene stå å vente mens en del utfører sin oppgave (dette er uten pipeline).
- Dermed sløser vi bort dyrbar tid og ressurser når vi lar kun en komponent jobbe av gangen.
- Om vi bruker pipeline kan vi la alle delene av CPUen jobbe samtidig på flere instruksjoner.
- Dvs. at flere ting skjer i løpet av samme klokkesykel.
- Hvis vi har 4 instruksjoner å utføre, så vil alle delene jobbe på sin oppgave på det mest «effektive» punktet.
- Altså: instruksjonsminne vil hente instruksjon 4, mens control uniten vil dekode instruksjon 3, samtidig som ALUen regne ut instruksjon 2 og dataminne vil lagre resultatet av instruksjon 1 i minnet.

Minne:

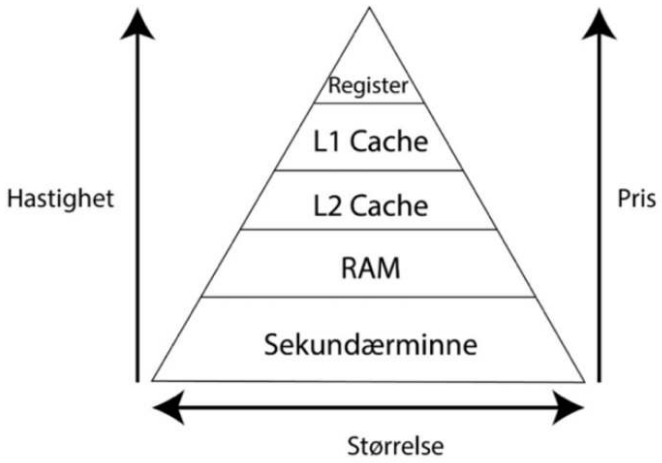
- En tredje måte å få CPUen til å gjøre instruksjoner raskere er å forbedre minnet.
- Vi har funnet ut av at det er lurt å ha flere nivåer av minnet slik at informasjonen som brukes mest hyppig av CPUen skal lagres nærmest, mens informasjonen som blir minst ofte brukt kan lagres lengst unna.
- Hvert nivå av minne består av en type minnekomponent som har sine fordeler, disse er:
- Flip flop:
    - Komponent som bare kan lagre 1 bit, altså 1 eller 0.
    - Har den enkleste strukturen av alle komponentene, så det er den raskeste.
- Register:
    - Ligger inn i CPUen og brukes for å lagre informasjonen vi bruker «nå».
    - Vi har ikke så mange registre, så det er lite kapasitet til lagring.
    - Vi bruker registre til å lagre den viktigste informasjonen i løpet av en begrenset tid der informasjonen trengs.
    - For eksempel i LMC hadde vi 4 registre som lagret informasjon: neste instruksjon som skal utføres (program counter), verdiene vi jobber med/resultatet (akkumulator) , instruksjonen som skal utføres (instruksjonsregister) og hvor i minnet vi skal aksessere/lagre en verdi (address register).
    - Registrene resettes hver gang maskinen blir slått av.
- Cache:
    - Minnelement som ligger rett ved CPUen, men ikke «inni inni».
    - Brukes til å hente og lagre data som trengs fra RAM. Så den fungerer som et mellomlager for å lagre data mellom CPU og RAM.
    - Har veldig rask aksessering og men lite kapasitet.
    - Cache henter en haug med informasjon fra RAM, så når CPUen trenger informasjon (om det neste steget) så vil denne mest sannsynlig allerede ligge i cache.
    - Poenget med cache er at CPUen skal slippe å aksessere RAM (som ligger utenfor CPUen) hver gang den trenger informasjon.
    - Cache resettes hver gang maskinen blir slått av.
- RAM:
    - Større minneelement som ligger utenfor CPUen, men den ligger på hovedkortet.
    - Den holder på informasjon som CPUen bruker mye og vil ha rask tilgang til.
    - Mer kapasitet til lagring, men også tregere.

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

    - Kjennetegnes ved at den lagrer data i en ubestemt rekkefølge og kan aksesseres i vilkårlig rekkefølge.
    - RAM resettes hver gang maskinen blir slått av.
- Sekundærminne/harddisk:
    - Det største minneelementet på maskinen.
    - Ligger utenfor CPUen og hovedkortet.
    - Her lagres alt dataen og brukes til langtidslagring.
    - Det som kjennetegner sekundærminne er at det har stor kapasitet, men tregt. I tillegg vil dataen som lagres her ikke bli mistet når man skrur av maskinen.

Hvordan bruker CPUen minne?

- CPUen vil først sjekke om den informasjonen den trenger ligger i cache.
- Cache miss:
    - Når informasjonen CPUen trenger ikke ligger tilgjengelig i cache.
    - Om vi får cache miss, så må CPUen lete videre i RAM.
    - Om vi får en cache miss, så vil det tar lenger tid for CPUen å gjøre oppgaven sin siden den må aksessere RAM også.
- Cache hit:
    - Når informasjonen CPUen trenger ligger i cache og kan brukes.
    - Om vi får cache hit så kan vi bare bruke dataen direkte.
- Om informasjonen vi trenger ikke ligger i RAM så må vi ut å lete i sekundærminnet.
- Dette vil ta enda lenger tid siden dette minnet er stort og tregt.



5. Datamaskinarkitektur

Aritmetisk logisk enhet (ALU):

- Den delen av CPU hvor logiske og aritmetiske beregninger/operasjoner utføres
    - Addisjon, subtraksjon, AND, OR osv.
- Vi trenger fulladder, multiplekser, AND, OR og NOT
- ALUer i CPU:
    - Moderne CPU kan inneholde flere ALUer
    - ALUer kan designes til å håndtere flere funksjoner per trinn og mer komplekse.
    - Alternativt kan man bruke software aktivt til å designe slik at en ALU kan utføre komplekse operasjoner over flere trinn.

This document is available free of charge on studocu

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)

- TRADE-OFF: hvor skal man plassere kompleksiteten, i hardware eller software
- Ulemper med å sette kompleksitet i hardware er høyere kostnader i fbm strømforbruk, areal og produksjonskost.
- ALU designet spiller en stor rolle med hensyn på CPU sin ytelse, da det mest komplekse operasjonen setter maksimal klokkefrekvens.

Assemblerkode

- Tolkes til et bitmønster, som har 64 0'ere eller 1'ere, som datamaskinenforstår. De ulike delene av remsen har en mening, og dette må dekodes og det gjøres av CPU
- Forsinkelse i CPU skapes av at klokken til CPU'en må ta hensyn til den tregeste prosessen.
- Problem: hvordan effektivisere? Løsning: Pipeline.

Pipeline

- Fungerer etter samlebåndsprinsippet, ulike prosesser kan skje samtidig. (Omid vasker klær.)
- Hvis man har flere subinstruksjoner som skal skje samtidig, vi kan utføre subinstruksjoner fra forskjellige instruksjoner samtidig dersom instruksjonene er ulike.
- Dette gjør at vi sparer tid, og at det blir mer effektivt. Det at man har en 4 trinns pipeline vil ikke si at man får 4 ganger raskere prosessering, da det alltid vil gå noe tid til administrering av instruksjoner. Må lagre verdiene/dataene til hver subinstruks, eller så kan dataen bli overskrevet avneste instruks og vil skape trøbbel. Klokkesignal styrer synkroniseringen

- Det kan oppstå tre hazarder ved pipelining, resource hazard, data hazard og control hazard.
    - Resource hazard:
        - Oppstår hvis to subinstruksjoner ønsker å aksessere samme ressurs.
        - Løsninger: design hvor dette ikke skjer, stoppe (STALL) pipelinen lenge nok til å aksessere sekvensielt, bruke lokale register som er organisert i en registerfil, bruke Harvard arkitekturen som har to separate minner for data og instruksjon (ingen er særlig gode løsninger)
    - Data hazard:
        - Oppstår hvis to forskjellige instruksjoner trenger å aksessere samme data samtidig, den ene instruksjonen vil da trenge et resultat fra forrige instruksjon før den har produsert gyldig svar.
        - God løsning: Ved hjelp av forwarding kan man lage en snarvei i pipelinen, i dette tilfellet en direkte datapath som aktiveres ved en hazard.
    - Control hazard:
        - Hvis vi har en pipeline som innhenter neste instruksjon fortløpende vil det oppstå problemer når vi får en JUMP instruksjon (js, jns osv).
        - Da må vi tømme pipelinen for andre instruksjoner og lese inn den nye.
        - Løsning: ikke hente noe annet før man vet om man skal hoppe eller ikke, forutsi om man skal hoppe, hvis vi dobler hardware så vil vi ha to parallelle løp hvor den ene ikke hopper og andre hopper og deretter bestemmes det til slutt hvilken som skal brukes.
- **Dette er koblingen mellom assemblerkode og maskinkode.**

Downloaded by _MeNeS03_ (przemolmroz@gmail.com)