# Grunnleggende datateknikk

## Datahistorikk

## Maskinklasser

Introduksjon

* Stormaskin
  + Moderne arvtageren etter den første generasjonen datamaskiner. De er voluminøse, kostbare som blir brukt til å håndtere store menger data og kjempeoppgaver.
  + Ex: på dette er databaser, arkiver, store regnskapssystemer osv.
  + Det meste framtredende trekk ved denne klassen er uovertruffent I/O-kapasitet, både med hensyn til interaktiv I/O-kilder, nettverk, disker og andre typer arkivmedia.
* Signalprosessorer
  + Prosessorer som er små, billige prosessorer som utfører spesialiserte, avgrensede oppgaver i behandling av digitaliserte fysiske data. (Dataanalyse)
  + Har logikk for å raskt utføre matematiske operasjonen knyttet til f.ex. fourier-transformasjon og digital filtrering.
  + Ex: Behandling av lyden i en mobiltelefon, ekkoet fra et seismisk «skudd», analyse fra gps-satelitter osv.
* Kontrollere
  + Ligner på signalprosessorer, men brukes til styringsoppgaver. Kontrollere er generelle prosessorer som primært genererer eksterne styresignaler.
  + I dag brukes ofte spesialutgaver av standard mikroprosessorer til kontrollere; i tillegg til CPU er det lagt inn I/O-elektronikk på samme brikke.
  + Ex: fjernkontroller, ekeltrisk utstyr, leker osv
* Minimaskiner
  + En nedskalert stormaskin, som vanligvis har mellom 3 og 300 interaktive brukere, diverse muligheter for ulikt I/O-utstyr, fleksibilitet med hensyn til utbygging og relativt stor regnekapasitet.
  + Ex: DECs VAX-serie fra Norsk Data
  + Blir ikke brukt eller produsert lenger, ble utkonkurrert av PC og andre mikromaskiner i nettverk.
* Personlig datamaskiner
  + Begynte som hobbymaskin på 80-tallet. De fikk etter hvert virtuelt minne, disklager på hundrevis av megabytes, høyoppløselige skjermer og flerprosess-OS.
  + Ex: PC og Mac
* Servere (tjenermaskiner)
  + En generell maskin med spesielt utstyr og/eller spesiell *konfigurering* og *bruk*. Det kan være en ordinær PC strippet for det meste interaktive I/O-utstyret, men istedenfor utstyrt for spesielle oppgaver.
  + Ex:
    - Diskserver: utstyrt med stor disk-kapasitet med raske disk-kontrollere, gjerne flere SCSI I/O-busser, rask systembuss for å kunne overføre data til minne, og en eller flere tilkoblinger til lokalnett. Utstyrt med utstyr og programvare for pålitelig, automatisk sikkerhetskopiering.
    - Kommunikasjons-server: Kan ha kobling til Internett eller andre typer nettverk, og kan stå som en diskserver for kommunikasjons-relaterte data.
  + Servere utgjør en viktig del an klient-tjener-løsninger, som blant annet hele internett er bygget opp av.
    - Baserer seg på at en maskin (server) utfører oppgaver på forespørsel fra en annen maskin (klient). En servermaskin kan utføre tjenester for mange klienter samtidig.
* Supermaskin for tungregning
  + Maskiner lagd med vekt på ekstremt hurtig regnehastighet. Dette er maskiner for såkalt tungregning; store, tunge og dyre installasjoner.
  + Forskjellen på en stormaskin og en supermaskin, er stormaskin har bra prosessorkapasitet, men størst fokus på stor I/O-kapasitet. Derimot er supermaskiner laget for særdeles stor prosessorkapasitet, og mye mindre kapasitet for I/O

Mer om enkelte klasser

* Stormaskin
  + En stormaskin håndterer «alt» av I/O utstyr, men med vekt på lagringsmedier og kommunikasjon, mot terminaler, printere og datanett. RAM-minne på 100-talls Gbyte og er betydelige buffere plassert ute på kanal-enhetene i tillegg til CPUens egen RAM.
  + Stormaskiner benyttes primært i administrative og økonomiske systemer (de gamle etablerte datamiljøene)
* Supermaskiner til tungregning
  + Hovedforskjellen mellom super- og stormaskin er forholdet deres mellom CUP-hastighet og I/O kapasitet. Super har mer fokus på ekstrem rask CPU og stor nok I/O kapasitet. Stor har mer langsommere CPU, men ekstrem stor I/O-kapasitet.
  + Tungregning er brukt av maskiner med svært stor regnekraft til å utføre kompliserte beregninger.
    - **Def: «***Et tungregneanlegg er en datamaskin som har en spissytelse som er minst 20 ganger spissytelsen til den til enhver tid raskeste arbeidsstasjon.»*
      * Siden arbeidsstasjoner øker hurtig, øker tungregning like raskt.
  + Hva trenger vi egentlig tungregneanlegg til?
    - Innen teknikk og naturvitenskap brukes matematiske modeller til å beskrive store og komplekse systemer. For å gjøre beregninger på slike modeller ender vi ofte opp med å måtte løse problemer som krever så mye hukommelse og/eller regnetid. Slike beregninger er ikke mulig på konvensjonelle datamaskiner.
    - Ex: Værvarsling, beregning innen kvantefysikk og kvantekjemi, simulering av vann osv
  + Tungmaskin blir utviklet i tre retninger:
    - Supermaskin med få, meget raske prosessorer. Vektorprosessering (dvs. samme instruksjon på mange data samtidig. (i parallell)
    - Parallelle datamaskiner med et stort antall mindre kraftige prosessorer
    - Cluster (klynger). Samling av billige arbeidsstasjoner som sammarbeider om å utføre beregninger.
  + Parallell tungregning
    - En vektormaskin har èn prosessor, men kan utføre samme instruksjon så mange ulike data samtidig. (Utfører same instruksjon på mange data samtidig)
    - Parallelle datamaskiner med flere prosessorer som jobber sammen om å utføre beregningene. Disse prosessorene jobber mot samme minne. Prosessorene kan enten være standart prosessorer eller spesiallagede prosessorer.
    - En «klynge» maskiner som kommuniserer over et hurtig nettverk, for å utføre hver sin deloppgave. Fra brukerens synspunkt er det som en stor maskin. Mange dyre, hurtige arbeidsstasjoner er fortsatt billigere enn en massiv parallellmaskin
* Mer om signal-prosessorer
  + En-chips spesialisert mikroprosessor som er særlig tilpasset sum-av-produkter type operasjoner. (a = b+c\*d+e). De kan utføre en multiplikasjon og to addisjoner / to multiplikasjoner og fire addisjoner pr.klokkepuls. Pipeline.tenikker og hardware-dekoding og eksekvering.
  + Prosessorene er enten heltallsaritmetikk eller flyttallsaritmetikk. Flyttall er egnet til rekkeutvikling, der det legger sammen stort antall små ledd og et mindre antall store ledd.
    - Flyttall presisjon er 32 bits, Heltall er 16bits.
  + **Signalprosessorene kan betraktes som generelle mikroprosessorer der en del funksjoner er fjernet, multiplikasjons – og addisjonsinstruksjoner er særlig hurtige, og en rekke tilleggsfunksjoner er trukket inn på CPU-brikken**

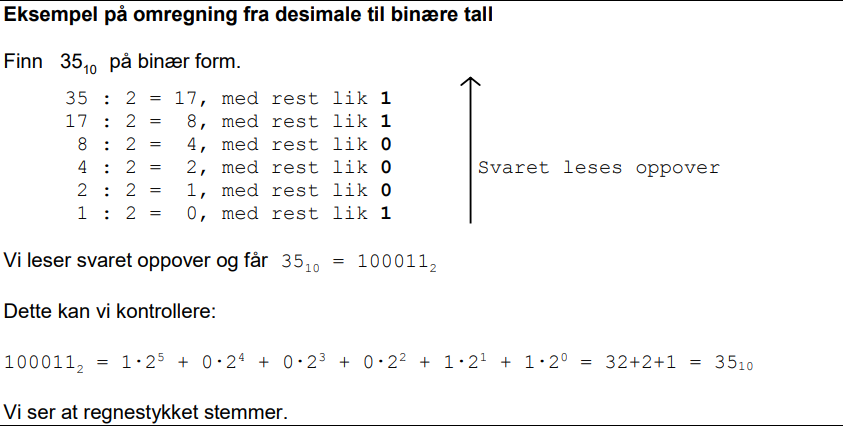
## Tallsystemer

* I vestlig kultur er det vanlig å bruke ti-tallssystemet, men å få til ti ulike spenningsnivå er vanskelig, pga støy, spenningsfall osv. Derfor bruker moderne datamaskiner binært tallsystem (to-talls). Nemlig av og på.
* **10-tallsystem:** (desimale tallsystem). Har grunntall 10:
  + ->
* **2-tallsystem:** (binært tallstem). Grunntall 2:
  + ->
  + Notasjon:
    - – fotskriften angir grunntall
  + Hvert siffer i et binært tall kalles en **bit**. Bit = Binary digit. En samling på 8 bit er en **byte**. Samling på 4 bit kan kalles en **nibble**.
  + Omregning fra 10-tall til 2-tall (Desimal -> Binært).
  + Metode 1: Prøv deg frem
    - Metoden egner seg for verdier som ikke er alt for store

-> 3510

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 |

->

* + Metode 2: Gjentatte heltallsdivisjoner hvor vi tar vare på restene
    - C:\Users\Zaim Imran\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image001.png
    - 
    - N siffer kan kombineres på 2n forskjellige måter, og gir 2n forskjellige binære tall. Siden vi begynner på 0 blir tallene mellom 0 og (2n-1)
* **Oktale tallsystem (8-tall):** 0-7
* **Hexadesimale (16-tall):** 0-9, A, B, C, D, E, F
  + Hexa til desimal:
  + Desimal til hexa:

|  |
| --- |
| 201410 : 16 = 125 (Runde av ned) | 125 \* 16 = 2000 -> 14 rest |
| 125:16 = 7 | 7 \* 16 = 112 -> 13 rest |
| 7:16 = 0 | rest 7 |
| (7 13 14)16 = 201410 | 7DE16 = 201410 |

* De første tallene i de viktigste tallsystemene

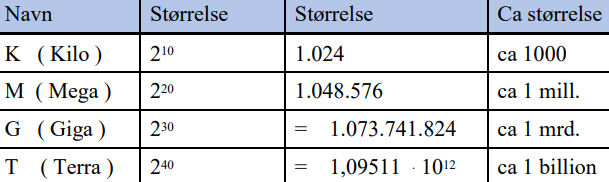
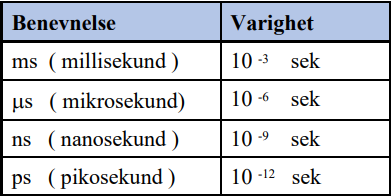
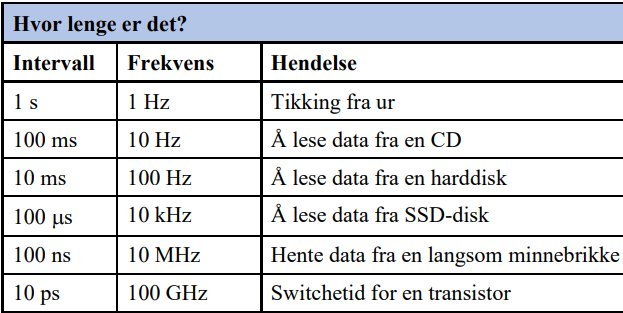
|  |  |  |  |
| --- | --- | --- | --- |
| **Desimal** | **Binær** | **Hexadesimalt** | **Oktalt** |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 2 | 10 | 2 | 2 |
| 3 | 11 | 3 | 3 |
| 4 | 100 | 4 | 4 |
| 5 | 101 | 5 | 5 |
| 6 | 110 | 6 | 6 |
| 7 | 111 | 7 | 7 |
| 8 | 1000 | 8 | 10 |
| 9 | 1001 | 9 | 11 |
| 10 | 1010 | A | 12 |
| 11 | 1011 | B | 13 |
| 12 | 1100 | C | 14 |
| 13 | 1101 | D | 15 |
| 14 | 1110 | E | 16 |
| 15 | 1111 | F | 17 |

* Regning mellom binær og hexa
  + Del opp de binære i byte og omgjør til hexa. LETT!!!!

|  |  |
| --- | --- |
| 1101 | 0110 |
| D | 6 |

* + 110101102 = D616

Andre benevninger

* 
* 
* 

## Datamaskinens virkemåte

* Grunnleggende virkemåte til en datamaskin

**CPU**

* Prosessoren er en elektronisk hjerne som er generell. Aka den kan utføre en del generelle enkle operasjoner; slik som pluss, sammenligne, sjekke om tallet er positivt osv.
* Bruker bitmønster som instruksjon til prosessoren, og den beskriver både instruksjon og hvilket data som skal behandles.
* Et dataprogram er en sekvens av instruksjoner, det vil si at instruksjonene som ligger i minnet overføres en for en til cpu og utføres

**Minnet**

* En enhet som kan lagre et antall *ord*. Et ord er en binær kode med n antall bit. Ofte er det 16, 32, 64, eller 128 bits *ordbredde.*
* Plass i minnet kalles *lokasjon*. Hver lokasjon nummereres med en adresse, som man kan lese eller skrive til.
* Data og instruksjon befinner seg i samme fysiske minne
* Lokasjonene i minnet kan adresseres uavhengig av om det er data eller instruksjoner som er lagret
* Det eneste som ligger i en minnelokasjon er et bitmønster. Det er opp til CPU å tolke dette mønsteret; enten instruks eller data.

**Buss**

* Et sett av parallelle ledere (Ledninger eller kobberbaner). Hver ledning overfører en bit, om i parallell kan bli et bitmønster.

## Busser

Introduksjon

* Parallell og seriell overføring
  + Parallell: overføring av bit over flere linjer.
  + Seriell: overføring av bit etter hverandre gjennom en ledning.
* Port
  + Når buss er koblet mellom nøyaktig 2 enheter
* Båndbredde
  + Overføringskapasitet på en buss. Hvor mye bit en buss kan sende i sekundet.
* Overføringsretning
  + **Simpleks:** En retnings informasjonssending; en sender og en mottar.
  + **Halv dupleks:** Begge enhetene kan lese og skrive, men ikke samtidig. Når den ene sender, den andre mottar.
  + **Full dupleks:** Begge kan lese og skrive samtidig, trafikk går i begge retninger.

Parallell buss

* Data buss – overfører binær data
* Adressebuss – overfører binære adresser
* Kontrollbuss – sendes signaler som styrer og synkroniserer overføringen.
* Synkrone busser – bruker klokkeslag
* Begrensning – klokkeslag – bits kommer ikke frem samtidig

Serielle busser

* Kan øke frekvens kraftig, fordi bits kommer frem i riktig rekkefølge uansett
* Fleksibelt
* Hot-swap: bytte ut usb uten å måtte stoppe maskinen
* Usb : 1.5 Mbits/s – 5Gbits/s

Svitsjede busser

* Som en telefon sentral, bytter mellom busser så de kan snakke sammen
* Dyrt

## Primærminnet

ROM

* Read Only Memory
* Ex: BIOS, mikroprogram
* EEPROM:
  + Elektrical (slette elektronisk)
  + Erasable (slette og skrive på nytt)
  + Programmerbare ( legge inn på et senere tidspunkt)
  + ROM
* Flash-memory

RAM

* Random Access Memory
* Kan vilkårlig lese fra hvor som helst til enhver tid
* DRAM
  + Dynamisk ram
  + Kondensator som husker på infomasjon i kort tid
  + Må stadig oppdateres
  + Aksesstid er 35-70 ns
  + Billigere enn SRAM, men litt tregere, men mye større lagringsplass
* SRAM
  + Hver celle en stabil transistorkreft (flipflop)
  + Informasjonen er lagret til drivspenningen forsvinner
  + Aksesstid er 1-10 ns

RAM vs ROM

* I dagens maskiner er ROM kun brukt til booting (oppstart av av datamaskin)
  + Grunnen er fordi RAM er tomt før strømmen skrus på
* BIOS (basic I/O system) – ligger systemrutiner som brukes av os.

Minnebrikkene

# f

## Cache

Prinsippet om lokalitet

* Dersom en minnelokasjon er benyttet, er det svært sannsynlig at neste vil ligge i nærheten
* I an cache blir minnelokasjonen og dens nabolokasjoner kopiert, fordi de mest sannsynlig vil bli brukt snart
* Sekvensielt
* While-løkker gjør at minne-lokasjonen refereres igjen
* Ikke lokalitet: subrutiner/prosedyrer – hopp i koden
* Sjelden funksjon i funksjon i funksjon
* Gjelder for data og instruksjoner
* Romlig og temporal lokalitet
  + Romlig: minnereferansene ligger nært hverandre
  + Temporal: minnereferansene brukes gjentatte ganger. (løkker)
* Cache-størrelse
  + En promille av primærminnet (4GB minne – 4MB cache)

Cachens virkemåte

* Kopierer minneblokk fra primærminnet
* Hvis prosessor ikke finner adressen den trenger, henter den minne fra primærminnet og bytter dne ut i cachen

Effektiv aksesstid ved bruk av cache

* Te = Tc + (1-H)T

Cache virkemåte

* Blokker kan enten være **store**, slipper den å hente så ofte fra primærminnet. Ved romlig lokalitet
* Ved bom bil det ta lenger tid å hente fra primærminnet
* Ved temporale lokalitet er det bedre med **mindre blokker**, siden da kan man ha flere blokker fra flere steder i minnet. Høyere sannsynlighet for at de inneholder blokken som trengs
* Moderne cacher ligger på 32/64-bits

Mapping

* Full-assosiativ cache
  + Blokker kan ligge hvor som helst i cachen
* Ikke-assosiatic cache
  + Blokker konkurrerer om en bestemt linje.
  + Trashing oppstår når to blokker bytter på å kaste hverandre ut
* Sett-assosiativ cache
  + Deler opp linjene i sett. Det hver blokk skal inn i et gitt sett, men her kan de ligge i den linjen de ønsker

Utskriftsalgoritmer

* LRU:
  + Least recently used – bytter ut den blokken det er lengst siden den ble aksessert
* FIFO:
  + First In First Out - Bytter ut den blokken som har vært lengt i cachen
* LFU:
  + Least Frequently Used – bytter ut blokken som har vært aksessert færrest antall ganger
* Random:
  + Tilfeldig utskriftning – tilfeldig hvem som blir kastet ut