Litt om maskinvare

Hvorfor lære om maskinvare* ?

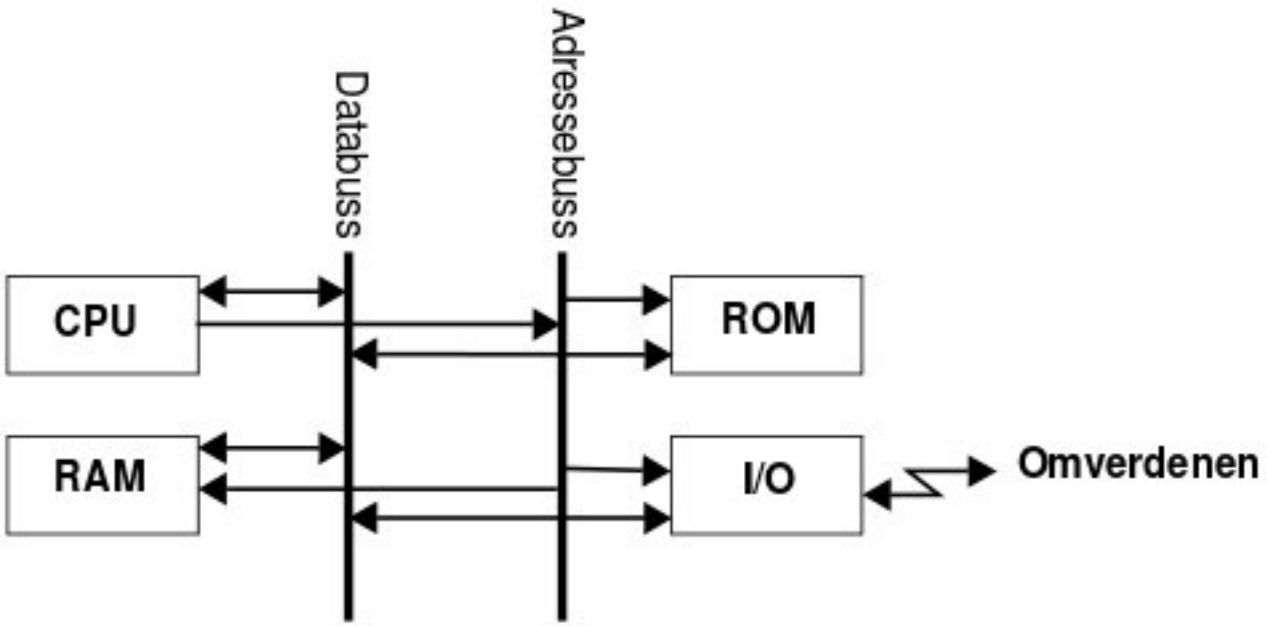
- Hovedoppgaven til et OS er å styre maskinvare
- Må ha grunnleggende kjennskap til maskinvare for å forstå hvordan OS fungerer
- Skal bare se på grunnleggende prinsipper:
  - Fokuserer på det som er relevant i forhold til OS
  - Går litt mer i dybden enn læreboken
  - En del av dette er kjent stoff for de fleste

* aka “hardware”, “elektronikken”, “leamikk”

Hovedkomponenter i en datamaskin*

- Mikroprosessor – Central Processing Unit
- Minne – Random Access Memory
- Read-Only Memory
- Input/Output enheter – I/O-celler
- Busser

* Foruten tastatur, skjerm, disk og andre nødvendigheter

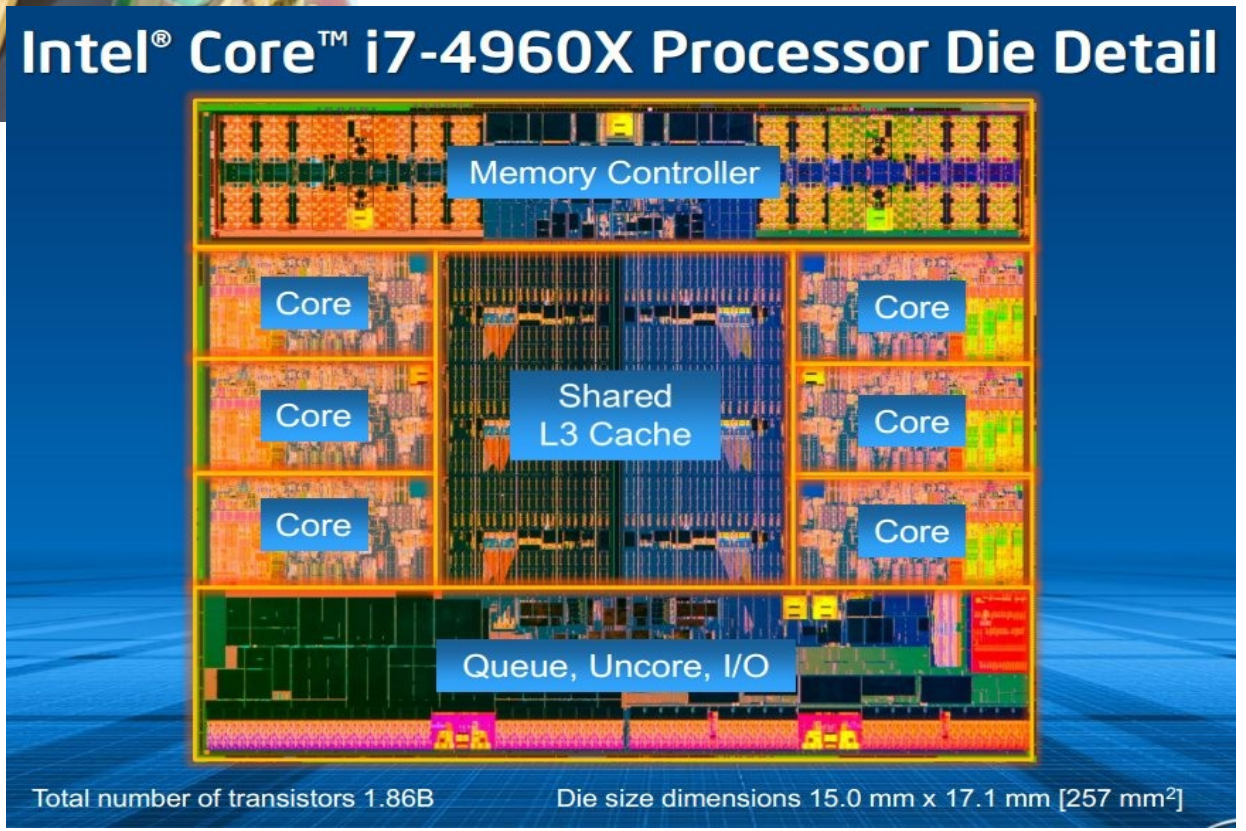


Hovedkomponentene i en datamaskin

Central Processing Unit (CPU)

- “Hjernen” som utfører det meste av beregningene som får maskinen til å fungere
- Inneholder ofte flere mikroprosessorer (kjerner/ cores) som kan utføre beregninger parallellt
- Er ofte koblet med en grafikkprosessor (GPU)
- Krever kjøling

DIGI+ DRAM
Fan Xpert 2
DIGI+ VRM


Intel® Core™ i7-4960X Processor Die Detail
Memory Controller
Core
Core
Core
Core
Core
Core
Shared
L3 Cache
Queue, Uncore, I/O
Total number of transistors 1.86B
Die size dimensions 15.0 mm x 17.1 mm [257 mm²]

Komponenter i en CPU (1)

- Maskinens instruksjonssett
  - De grunnleggende operasjoner som kan utføres (ADD, SUB, MOV, JMP, JEQ...)
  - Hardkodet i elektronikken
- Registre
  - Svært raske minneceller som kan leses og skrives
  - Lite, begrenset antall registre
  - Brukes til regneoperasjoner og lagring av kontrolldata (f.eks. RAM-adressen til neste instruksjon)

Komponenter i en CPU (2)

- Arithmetic Logic Unit (ALU)
  - Utfører alle regneoperasjoner og logiske tester
  - Bruker registrene både som *operander* og for å *lagre* resultater
- Minnekontrollenhet og instruksjonsdekoder
  - Henter inn neste instruksjon fra minnet
  - “Dekoder”, setter opp og kjører instruksjonen i ALU
- Og en del mer...

CPU-ens virkemåte

- Utfører et (kompilert) program en og en maskin-instruksjon om gangen:
  - Neste instruksjon i RAM hentes over bussen(e)
  - Instruksjonen dekodes og data legges i registrene
  - ALU utfører selve instruksjonen
- Programutførelsen kan avbrytes midlertidig:
  - Avbrudd / “interrupts” skjer svært ofte
  - Utføres av OS
  - Ved f.eks. behov for I/O eller feilsituasjoner

Minne

- Kalles også:
  - RAM – Random Access Memory
  - Hukommelse
  - Arbeidslager
  - Hurtiglager
- Lagrer både programinstruksjoner og data
- Ikke permanent, data forsvinner når “strømmen går”

elixir
M2U51264DS8HC3G-5T
512MB DDR-400MHz-CL3
PC3200U-30331
Warranty Void If Removed
0542.MN05A1107A.5YM0410N08.XX.CN
elixir
M2U51264DS8HC
512MB DDR-40
PC3200U-303
Warranty Void
0542.MN05A11

Minne: Adresserbare lagerceller

- Lagrer data i (tradisjonelt 8-bits) *minneceller*
- Hver celle har en (heltalls)adresse som kan brukes for å lese verdien eller lagre en verdi
- Lesing og skriving skjer ved å overføre data på databussen, mellom cellen og et register i CPU
- Cellene ligger oftest i “banker” med minnebrikker på hovedkortet

Cache-minne *

- Overføring av data mellom RAM og CPU over bussene tar “lang tid”
- Cache-minne er mindre samlinger av raske minneceller som enten ligger internt i CPU eller nærmere CPU enn vanlig RAM
- Cache er mye raskere å bruke enn RAM fordi vi slipper å sende data ut på bussene

* *cache*: depot, forråd

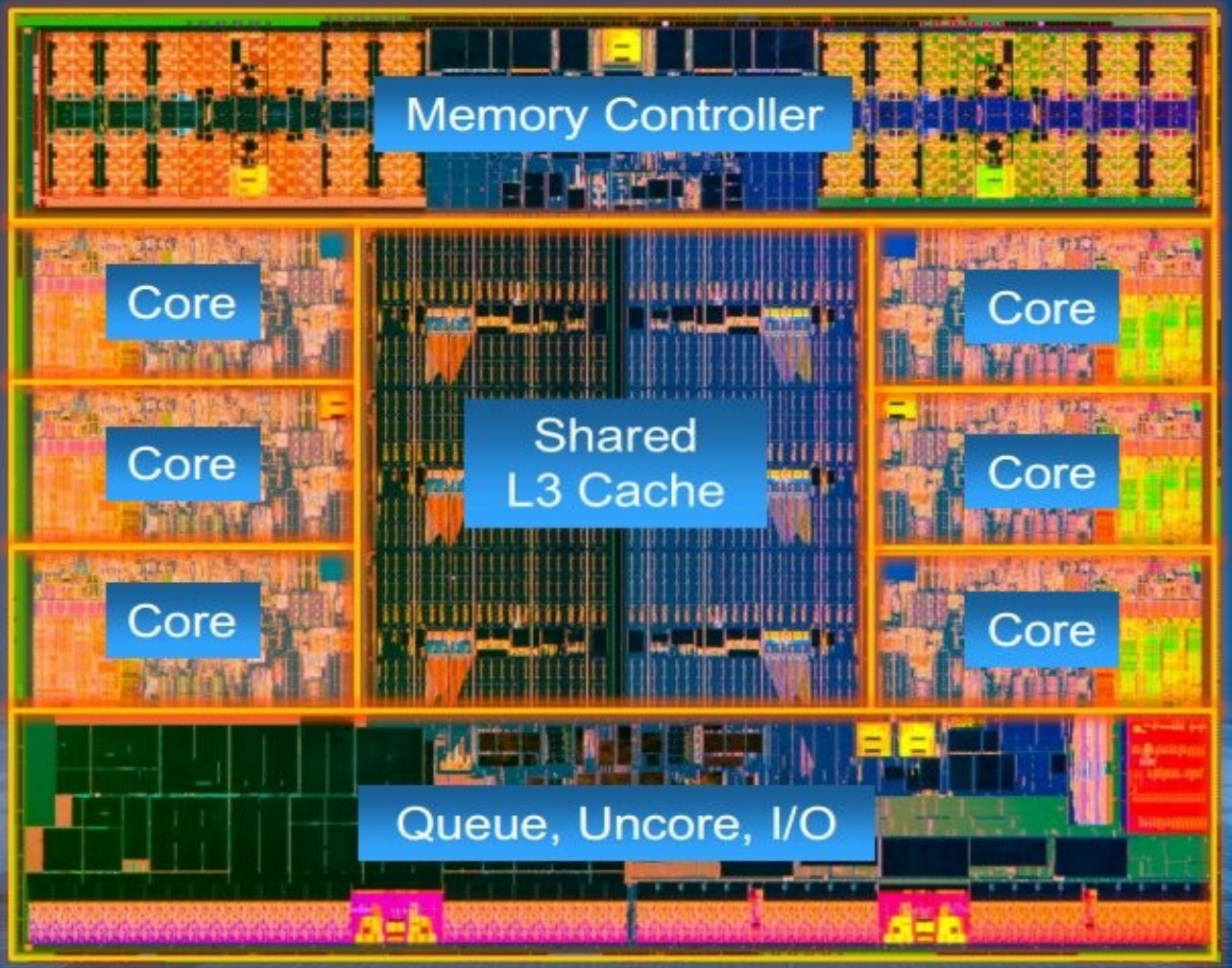
Styring av cache

- Håndteres av egen cache-kontroller internt i CPU
- Er transparent for programkoden som utføres
- Hvis vi klarer å lagre de mest brukte dataene i cache, går programutførelsen mye raskere
- Caching-algoritmer er kompliserte, og krever mye “bokholderi” og testing for å sikre korrekthet og konsistens

Ulike typer cache

- Nivå 1 cache:
  - Eget lite minneområde internt i en CPU-kjerne
- Nivå 2 cache:
  - Større ekstern eller intern cache-brikke med egen rask buss
  - Ofte én egen level 2 cache for hver CPU-kjerne
- Nivå 3 cache:
  - “Mater” level 2 cache med data fra RAM
  - Ofte level 3 cache som deles av alle kjernene (og av GPU)

Intel® Core™ i7-4960X Processor Die Detail



Total number of transistors 1.86B

Die size dimensions 15.0 mm x 17.1 mm [257 mm^2]

Read-Only Memory (ROM)

- Permanent minne, innholdet er “brent” i elektronikken
- Lagrer BIOS (Basic Input Output System)
- BIOS inneholder instruksjonene som kjøres når maskinen startes (booting / bootstrapping)
- ROM er nødvendig for å kunne få systemet til å starte
- Se også:
  - Firmware
  - UEFI

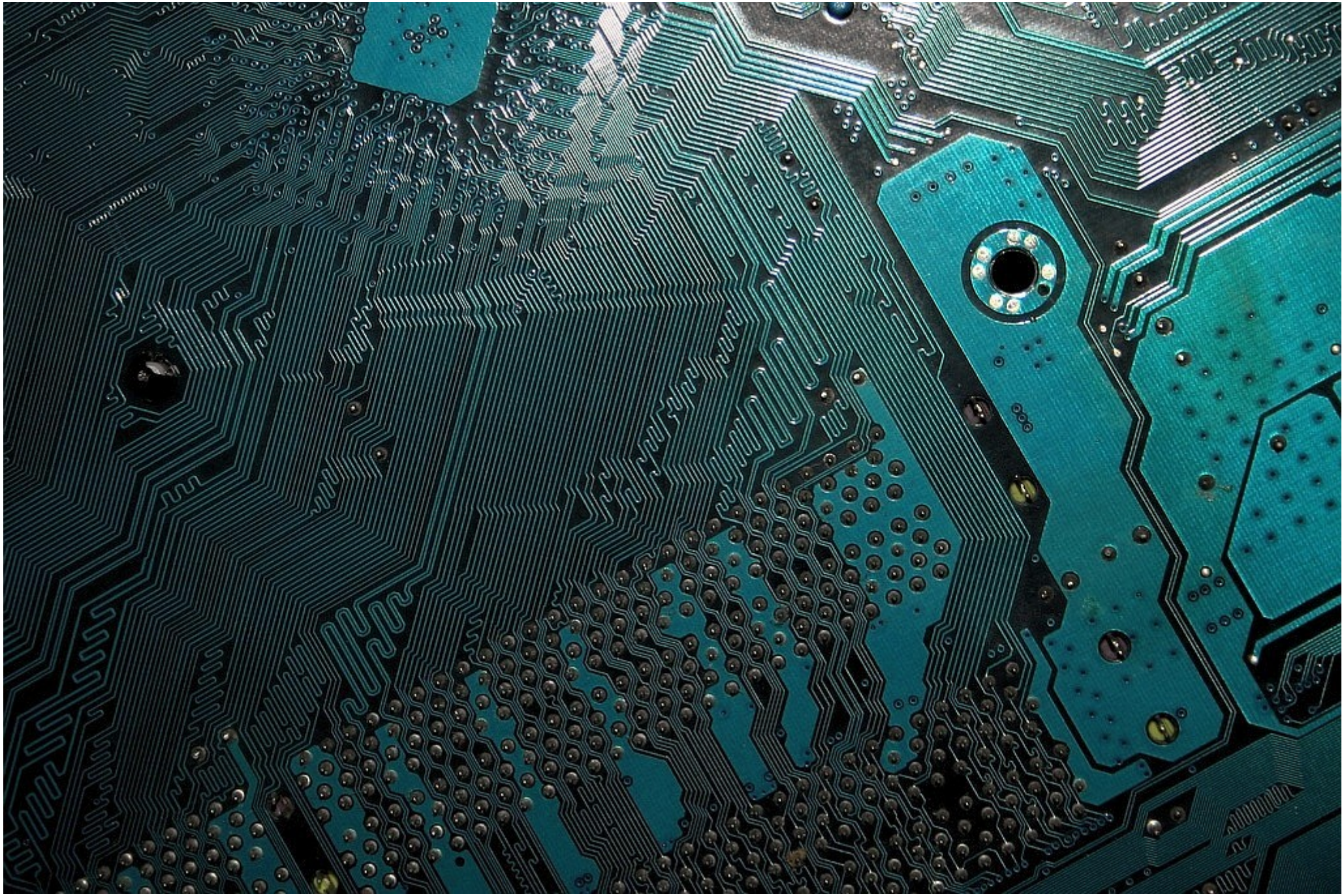
I/O-celler: Spesialisert minne

- CPU kommuniserer bare gjennom å lese og skrive til/fra minnet
- Input/output-enheter har derfor et grensesnitt mot CPU som *ser ut som* minne: I/O-celler.
- I/O-celler brukes til kontroll og dataflyt fra/til disk, nettverk, tastatur, skjerm etc.

Bussene

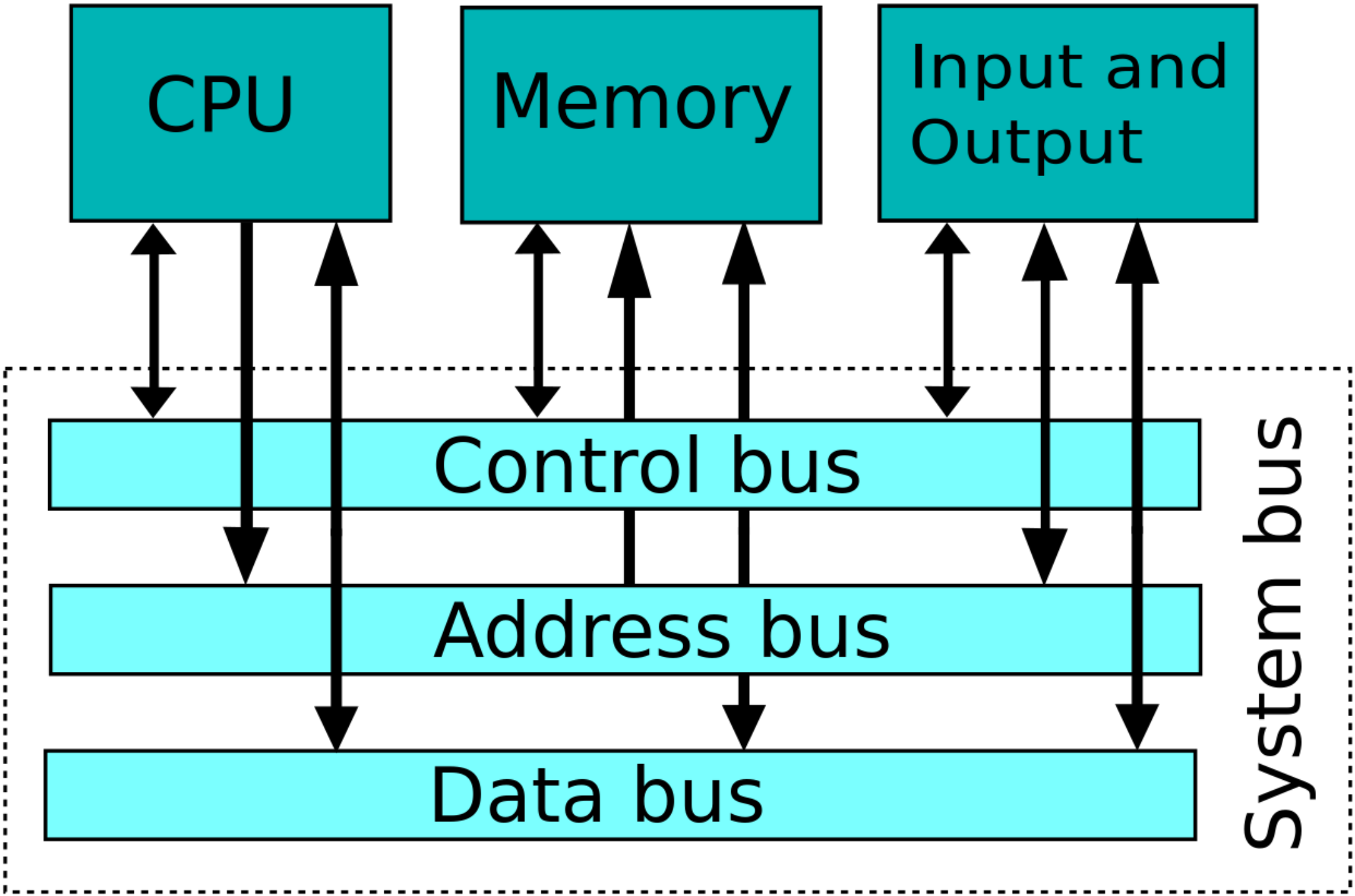
- “In computer architecture, a bus (from the Latin omnibus, meaning 'for all') is a communication system that transfers data between components inside a computer, or between computers. This expression covers all related hardware components (wire, optical fiber, etc.) and software, including communication protocols”

  – *Wikipedia*



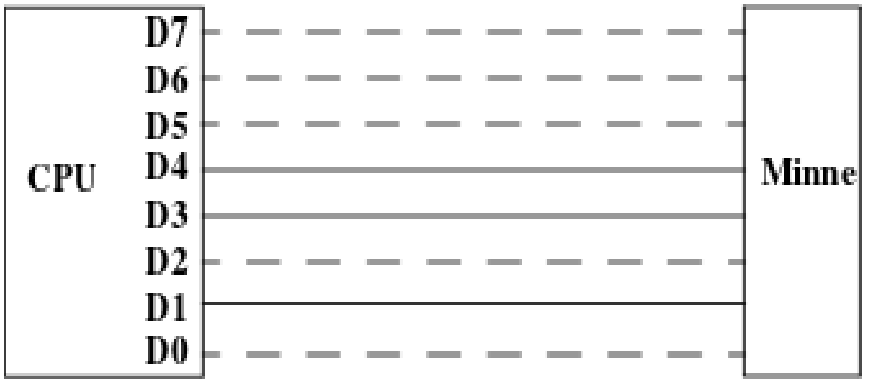
Systembussen

- Systembussen er et sett av *parallelle* signalveier som forbinder CPU med RAM, andre interne komponenter og I/O-celler
- Er oftest satt sammen av tre ulike busser som har hver sine oppgaver:
  - Databussen
  - Adressebussen
  - Kontrollbussen


CPU
Memory
Input and Output
Control bus
Address bus
Data bus
System bus

Databussen

- Overfører data mellom CPU og minne
- Overføres binært som elektriske signaler (0 eller 1) over minst 8 (evt. 16, 32) parallelle ledere



Tilstanden på databussen ved overføring av verdien 00011010.
Stiplet linje forestiller 0 volt spenning, heltrukken linje forestiller 5 volt.

Lesing og skriving på databussen

- CPU, minnet og I/O-celler er alle koblet til databussen
- Kun én enhet om gangen kan legge ut en dataverdi på bussen:
  - Minnet legger ut verdi hvis CPU vil *lese* en minnecelle
  - CPU legger ut en verdi hvis det skal *skrives* til minnet
- CPU styrer kommunikasjonen på databussen og adresseringen av minne, ved i tillegg å sende signaler på *adressebussen*

Adressebussen

- Bare CPU kan legge en verdi på adressebussen, “alle andre” bare lytter *
- *Før* hver overføring av verdi på databussen, legger CPU ut *adressen* som skal brukes på adressebussen
- Egne *dekodingskretser* leser denne adressen og sørger for at riktig minnebrikke kobles inn (“chip select”)
- For å angi om en verdi skal leses eller skrives og for å synkronisere, brukes enda en buss: Kontrollbussen

* Dette er ikke helt sant, I/O til disk bruker også adressebussen

Kontrollbussen

- Styrer samarbeidet mellom CPU og maskinvaren, f.eks:
  - Angir om verdi på databussen skal leses fra eller skrives til minneadressen på adressebussen (“write enable”)
  - Synkronisering av CPU og minneceller, slik at vi sikrer korrekt overføring av verdier på bussene
  - Overfører signaler fra I/O-enheter om at de har eller vil ha data, eller om at noe “akutt” har skjedd (interrupt)
  - Synkronisering mellom flere CPU’er på samme maskin
  - Synkronisering av I/O til/fra disk

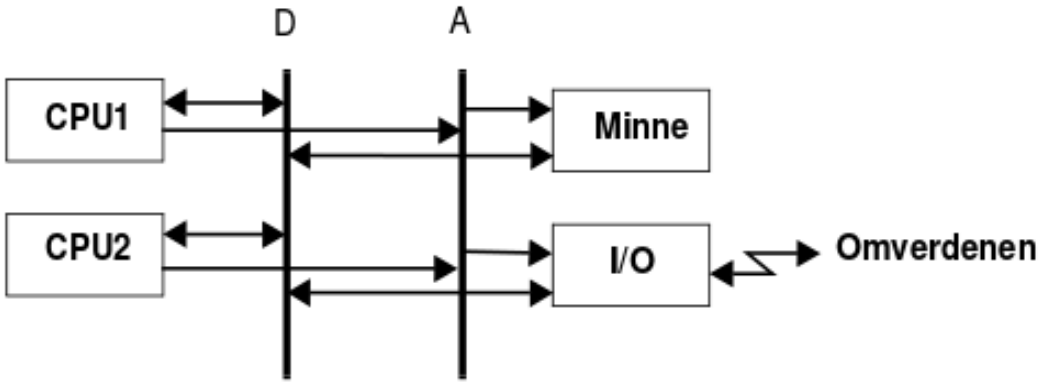
Buss-syklus

- En *buss-syklus* består (forenklet) av følgende operasjoner:
  - CPU legger en adresse på adressebussen
  - CPU legger styringsdata på kontrollbussen
  - Verdi overføres mellom to enheter på databussen
- En buss-syklus kan kan typisk ta et hundre-milliondels sekund, vi har da en 100-MHz systembuss

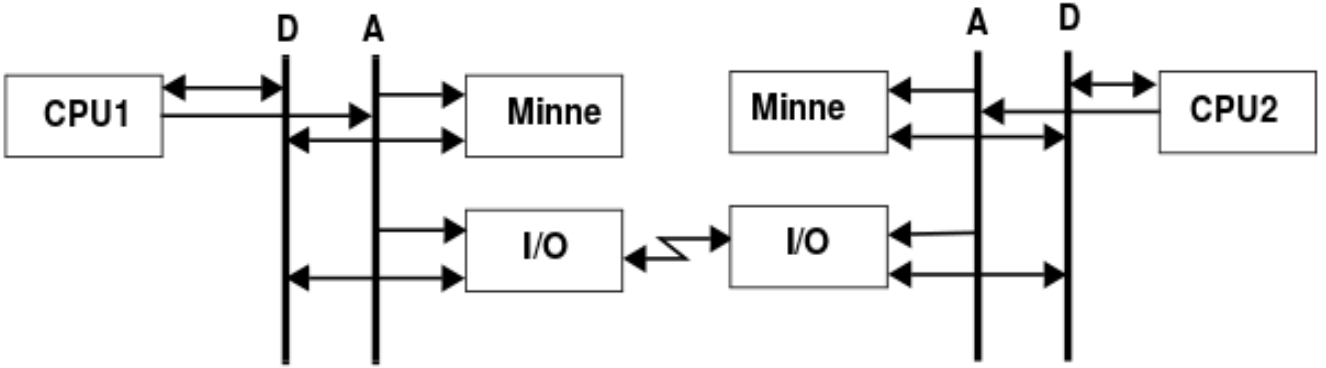
Buss- og CPU-hastighet

- Teoretisk ytelse for typisk systembuss:
  - 100-MHz, med 32-bits busser
  - 32 bits (4 bytes) med data kan sendes til/fra CPU 100 millioner ganger i sekundet
- CPU jobber mye raskere enn dette:
  - Prosessorkjernene er typisk i GHz-området
  - 1 GHz = 10^9 = 1000 millioner instruksjoner per sekund
- Systembussen blir derfor en “flaskehals” i maskinen
- Kan forbedres bl.a. ved å bruke cacheminne

Multiprosessorer



Tett koplet multiprosessor



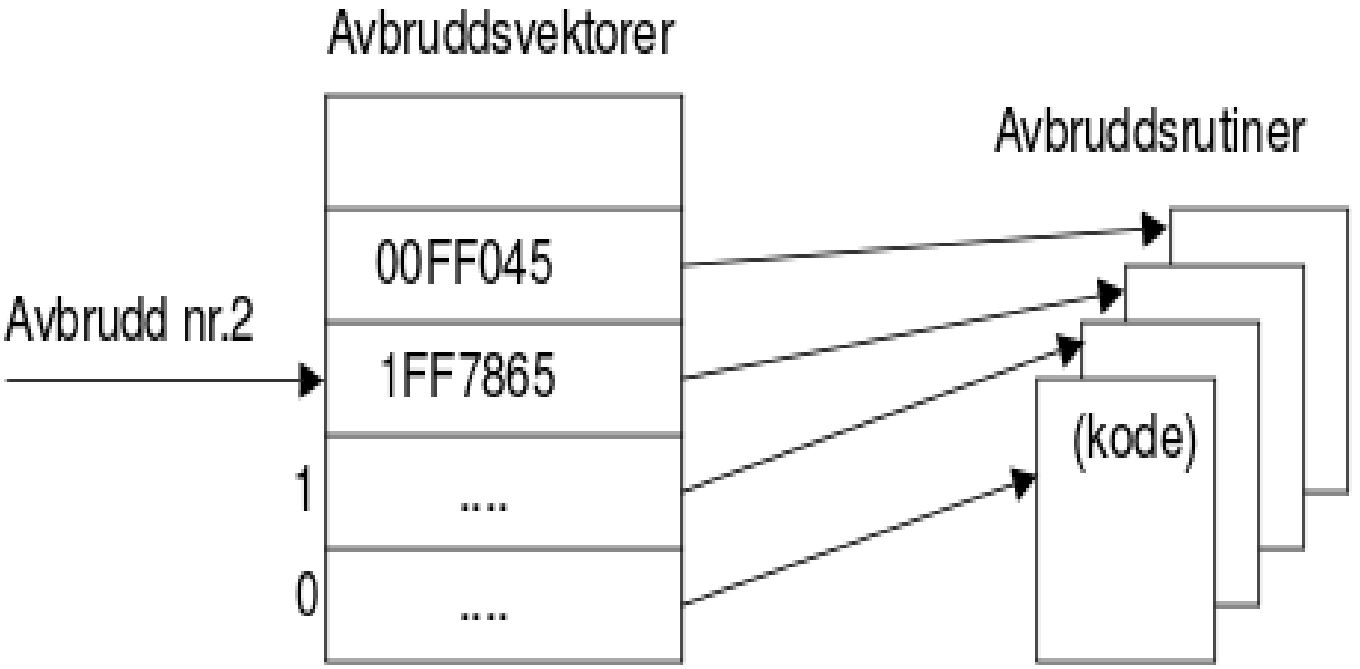
Løst koplet multiprosessor

Avbrudd / Interrupts

- I/O-enheter og komponenter kan sende *avbrudd-signaler* over kontrollbussen til CPU for å varsle om en hendelse som må håndteres, f.eks.:
  - Tastetrykk
  - Museklikk/bevegelse
  - Innkommende data over en nettverksport
  - Hardwarefeil
- Avbrudd skjer også planlagt av *timere*, flere ganger i sekundet, for å kunne gjøre *timesharing* i operativsystemet

Håndtering av avbrudd

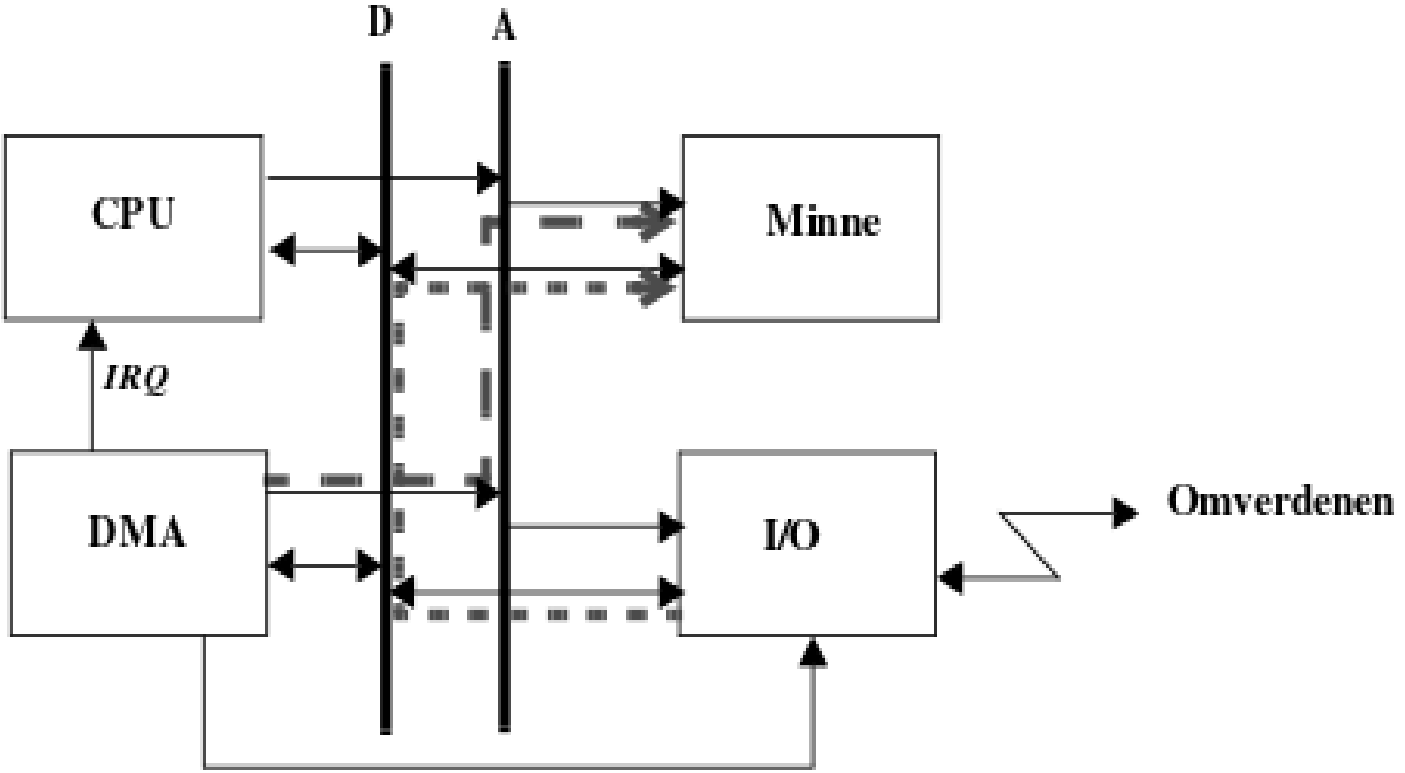
- Avbruddsignalene har en ID (et nummer) som forteller hva som har forårsaket avbruddet
- Når CPU mottar et avbruddsignal:
  - CPU fullfører nåværende instruksjon
  - Det startes deretter en *avbruddsrutine* som er et program som håndterer det angitte avbruddet
  - “Normal” kjøring gjenopptas deretter
  - Systemstatus må lagres før avbrudd håndteres



Eksempel på avbruddsvektorer. Når avbrudd nr.2 oppstår, finner CPUen adressen til vedkommende avbruddsrutine i element nr.2 i tabellen med avbruddsvektorer, i dette tilfellet 1FF7865. Tabellen med avbruddsvektorer ligger alltid på et kjent sted i minnet.

Direct Memory Access (DMA)

- Spesiell maskinvare – DMA-brikke – brukes for å flytte mye data raskere, typisk ved I/O mot disk
- Flytter data direkte mellom RAM og I/O-enhet, *uten* å gå innom CPU
- DMA-brikker legger adresser på adressebussen og slår av/på “write enable” i kontrollbussen
- Konflikter med CPU på bussene unngås ved:
  - Burst-mode: DMA stopper CPU med et avbruddsignal
  - Cycle stealing: DMA bruker bussen når CPU sier fra at den er ledig.



Signalveien for adresseinformasjon og data ved DMA-overføring

Linux og maskinvare

- Info. om maskinvare finnes i mange systemfiler, f.eks:

```
/proc/cpuinfo
/proc/meminfo
/proc/version
```

- Noen kommandoer for å se status for maskinvare:

```
arch
uname -a
lscpu
lspci
free -m
lsblk
df
```