Velkommen!

Mer om prosessoren, sikker programmering + Introduksjon til operativsystemer

Gruppe 27

(Hanne-)Kristin Schrøder Leiros

hkleiros@uio.no

Plan for dagen

- Repetisjon
  - Prosessoren, sikker programmering
  - Intro til OS
- Ukesoppgaver i plenum?
  - LMC del 2
  - Hovedsakelig tidligere prøveeksamensoppgaver
- *Gjerne avbryt med spørsmål eller innspill!*

Rett under 1 uke til Oblig 2

- Frist 4. oktober kl. 23:59
- Del 1: LMC programmering
- Del 2: Tallsystemer
- Gjerne still spørsmål på mail eller mattermost: hkleiros@uio.no

- De fleste har fått tilbake oblig 1
  - Sjekk studentmail / sett opp forwarding så dere får beskjeder om tilbakemelding og nye forsøk!

Læringsmål for uka

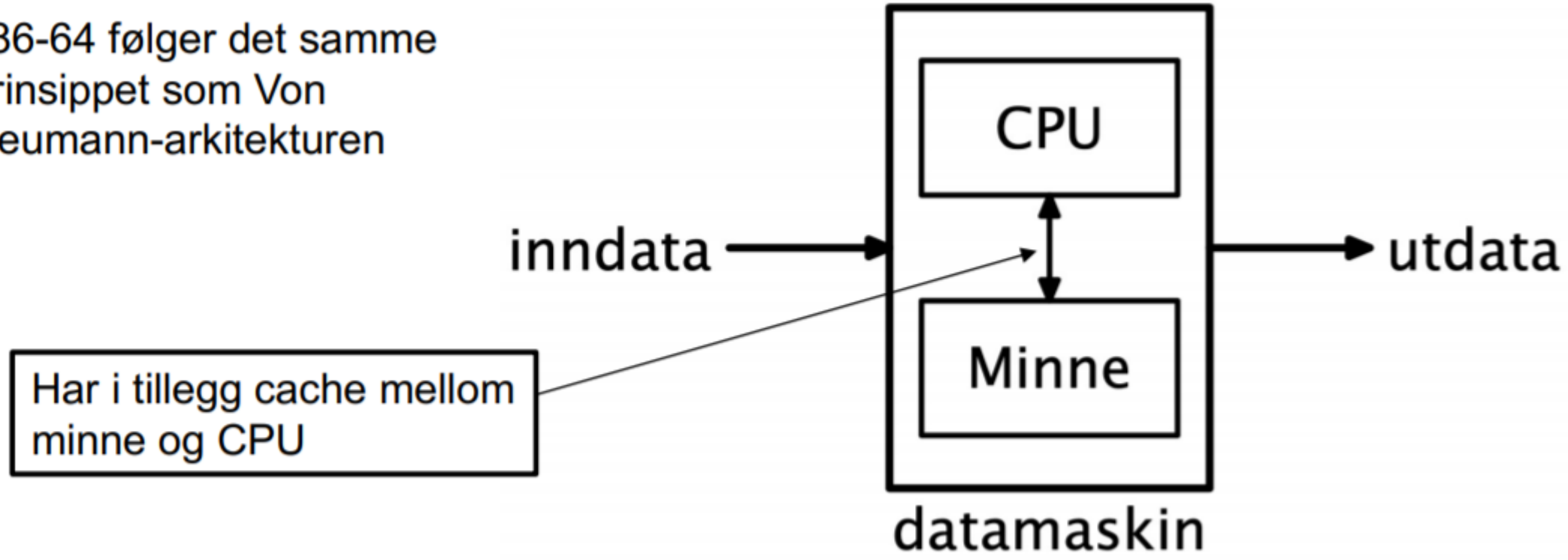
- Ha grunnleggende kunnskap over operativsystemer , hvordan de implementeres og hvorfor vi har dem

- Kunne grunnleggende om datahistorie og utviklingen av datateknologi

Prosessoren x86-64

- Den vanligste prosessoren
- Laget av Intel og AMD

x86-64 følger det samme prinsippet som Von Neumann-arkitekturen



Tilsvarende ekseveringsløkke som LMC

1. Hent instruksjonen i minnet;
   Gitt av programtelleren
2. Dekod instruksjonen i operasjon og adresse;
   legges i instrksjonsregisteret og adresseregisteret
3. Utfør instruksjonen
4. Om aktuelt;
   skriv svaret i minnet / register

X86-64 har i tillegg **pipeline** for å kunne jobbe raskere

Basic five-stage pipeline

Instr. No. \ Clock cycle	1	2	3	4	5	6	7
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX

(IF = Instruction Fetch, ID = Instruction Decode, EX = Execute, MEM = Memory access, WB = Register write back).

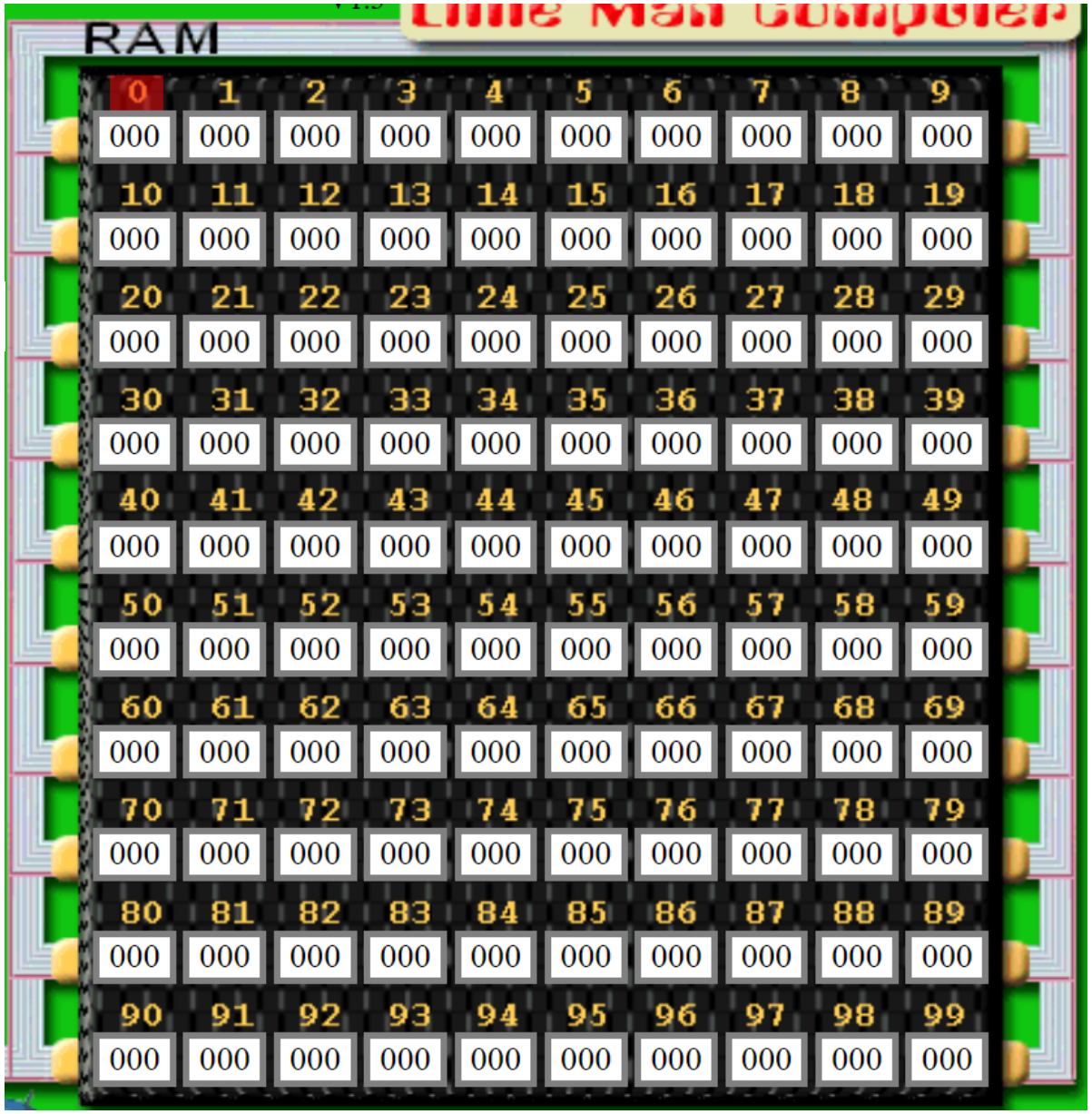
In the fourth clock cycle (the green column), the earliest instruction is in MEM stage, and the latest instruction has not yet entered the pipeline.

Registre

- Fra forrige uke:
  - Lagerceller for data som CPU-en trenger hurtig tilgang til
- I LMC har vi ett register som kan programmeres akkumulatorregistrert
- X86-64 har 16 registre
  - De kan brukes som 1-, 2-, 4-, og 8-bytes registre

RAM, minne

- LMC har vi 100 celler i minnet
- X86-64 kan ha intill $2^{64} \approx 1,8 \cdot 10^{19}$ celler
  - Ca 17 000 000 TB
  - Hver celle inneholder én byte, 8 bit
- X86-64 har også støtte for virtuelt minne



CPU-arkitektur

- CISC: Complex instruction set computer
  - Har noen enkle og noen ganske kompliserte instruksjoner
    - REP MOVSB: flytter et vilkårlig antall byte fra et sted i minnet til et annet
- RISC: Reduced instruction set computer
  - Begrenset antall enkle insturksjoner
    - Kan utføres raskt og effektivt
  - LMC bruker denne arkitekturen

Sikker programmering

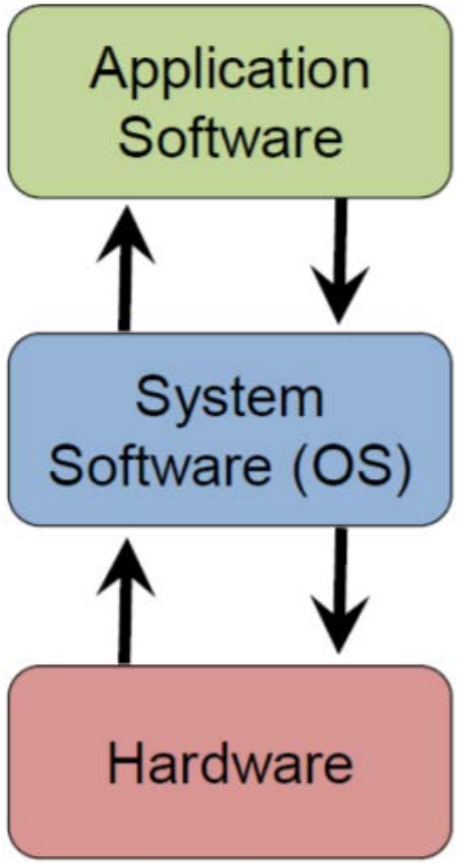
- Som programmerere er det viktig å skrive programmer som er
  - Trygge (safe)
    - At det oppfører seg fornuftig
  - Sikre (secure)
    - At de tåler bevisste angrep

4 generasjoner

- 1. Generasjon, maskinkode
  - Effektivt
  - Veldig lett å gjøre ulike feil
- 2. generasjon, assemblerkode
  - Slippe å huske numeriske koder
  - Adresser håndteres automatisk
  - Ingen feilskekking og ingen begrensinger
- 3. generasjon Cobol, Frotran, etc.
  - Mer strukturert programmering
  - Ingen sjekk av grenser
  - Ingen krav om å deklarere varaibler
- 4. generasjon
  - Mer fleksible
  - God sjekking av ulovligheter
  - Tregere eksekvering

Hva er et operativsystem?

- Dirigenten i datamaskinen
  - Søster for at flere brukere kan dele en datamaskin
  - Sørge for at flere programmer kan kjøre samtidig
  - Fungere som et «mellomnivå» mellom maskinvare og programvare



Hvordan fungerer OS?

- Oversetter fra applikasjoner til forskjellig maskinvare
- Utføre parallelle oppgaver
  - Bedre utnyttelse av maskinen
  - Kunne bruke flere programmer samtidig
  - Utfordringer:
    - «Samtidig» tilgang til informasjon
    - Sikkerhet
    - Rettferdighet av ressurser
- Enkel maskin (ovenfra og ned)
  - Skjuler kompliserte detaljer, presenterer noe som er enkelt å bruke
- Resurshånderer (nedenfra og opp)
  - Hvert program får sin tid/plass i resursene

Forskjellige typer OS

- Enkel-bruker, enkel-oppgave:
  Historisk og sjelden (kun noen gamle PDA 'er brukte dette)
- Enkel-bruker, multi-oppgaver:
  PCer og arbeidsstasjoner kan være konfigurert slik (typisk gamle mobiltelefoner)
- Multi-bruker, multi-oppgaver:
  brukes på servere, PCer, arbeidsstasjoner, bærbare PCer og de fleste moderne maskiner.

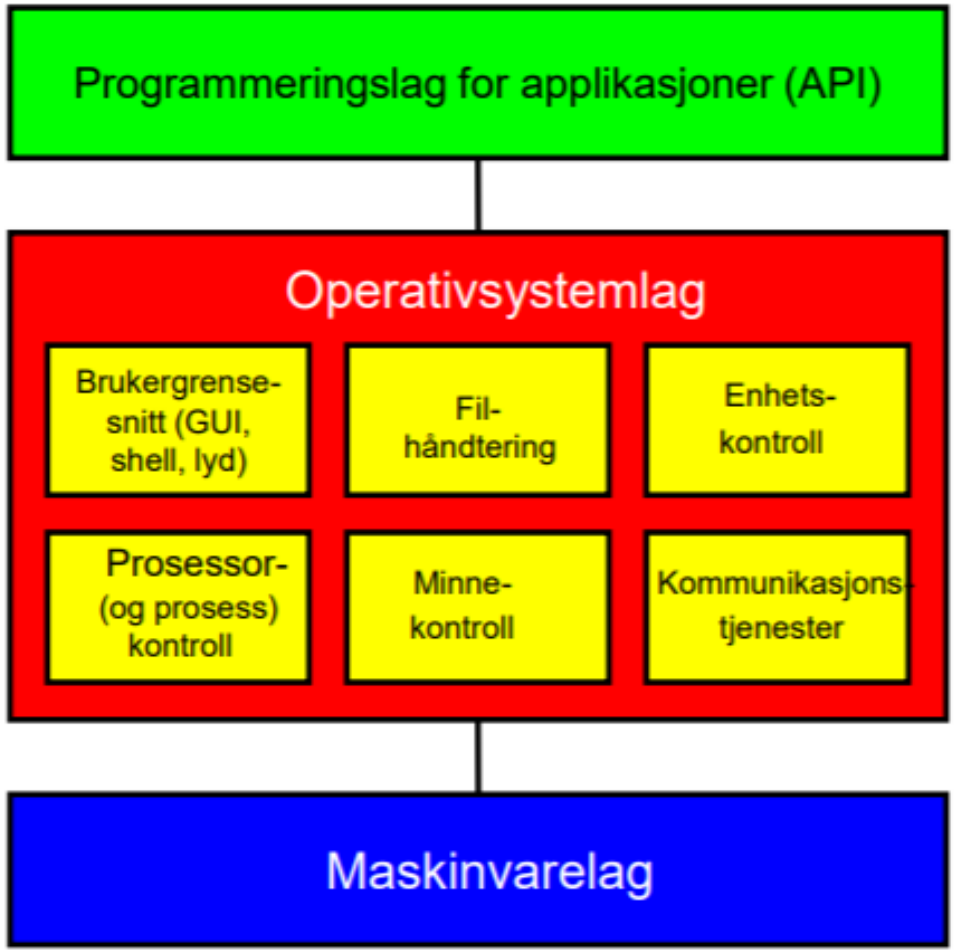
- Distribuert OS:
  Støtte for å kontrollere resurser på flere datamaskiner. Ikke så vanlig.

- Real-time OS:
  Støtte for systemer hvor sanntid er viktig som biler, robotter, fly, instrumenter, atomreaktorer, osv.

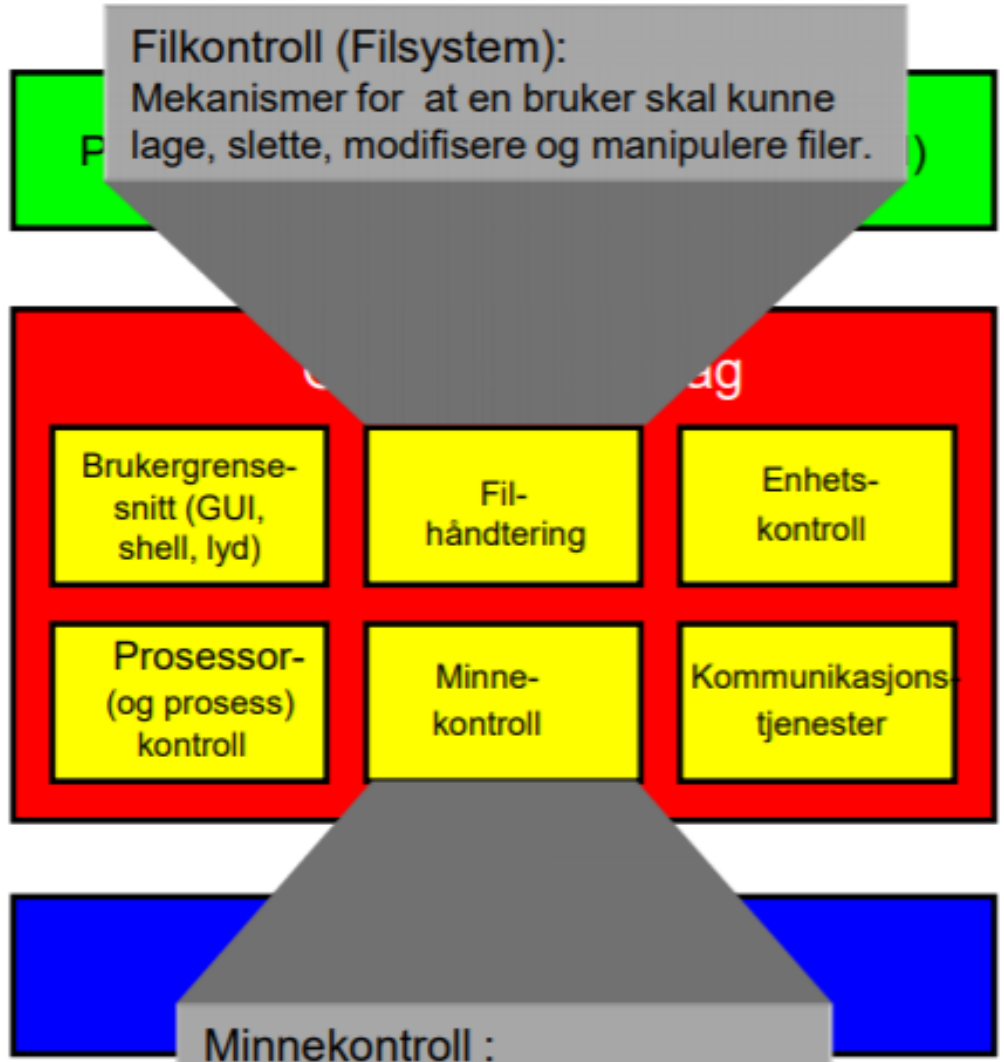
- Embedded OS:
  Bygd inn i enheter for å løse et konkret problem, som for eksempel enkle mobiltelefoner, mikrobølgeovner, vaskemaskiner, dørlås, osv.

Hovedkomponeneter i et OS

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



Hovedkomponenter - Detaljert



Filkontroll (Filsystem):
Mekanismer for at en bruker skal kunne lage, slette, modifisere og manipulere filer.

Brukergrensesnitt :
Mekanismer for at bruker og programmer skal kommunisere med OS
og bruke maskinen sine resurser.

Prosessorkontroll :
Tilby mekanismer for at systemet effektivt og rettferdig skal kunne dele CPU for programmene som kjører

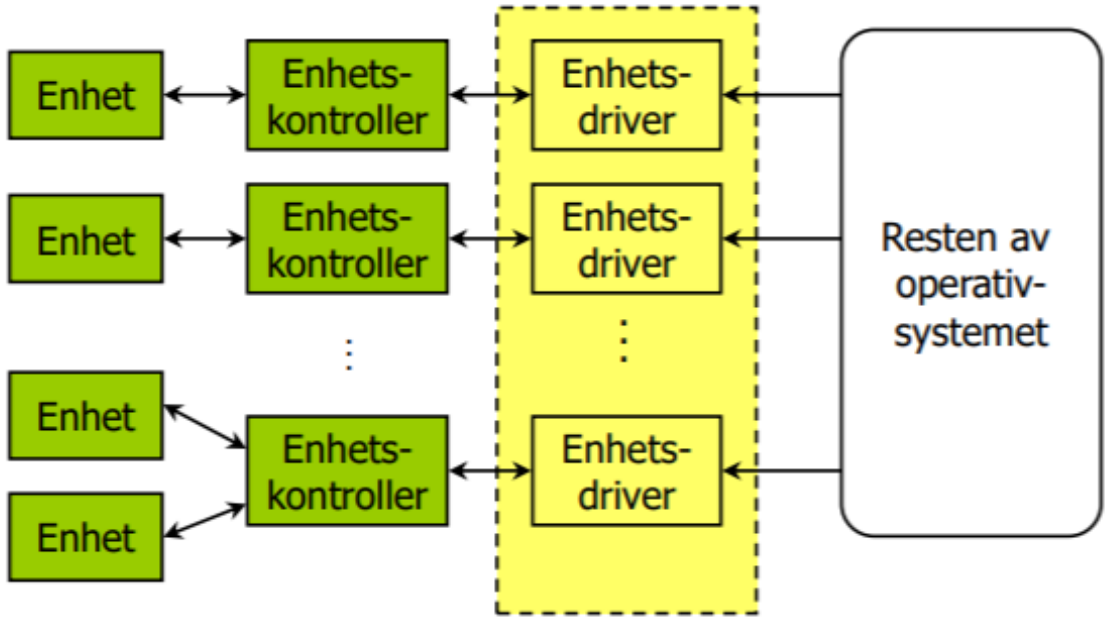
Enhetskontroll :
Gjøre at systemet skal kunne kontrollere enheter som er koblet til som for eksempel tastatur, mus, skjerm

Kommunikasjonstjenester :
Tilby tjenester for at systemet skal kunne med andre prosesser (på samme eller på andre maskiner)

Minnekontroll :
Tilby mekanismer for at systemet skal kunne fordele minneresurser – allokerer plass til programmer

Enhetskontroll

- Operativsystemet må kunne kontrollere enheter tilkoblet maskinen som for eksempel disker, tastatur, nettverkskort, skjermer, høyttalere, mus, USB-minne, kamera, DVD/Blu-ray, mikrofon, skrivere, joysticks, ...
  - Stor forskjell
  - Varierende hastighet
  - Forskjellige måter å hente data



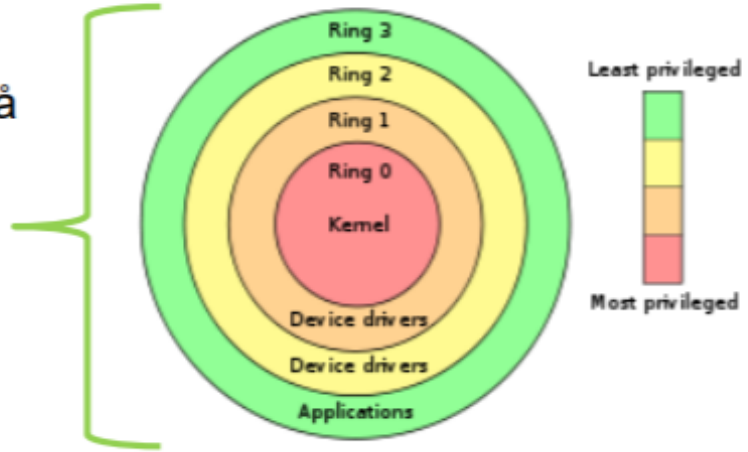
Brukergrensesnitt



- Koblingen mellom bruker og datamaskinen
- Operativsystemet har logikk som støtter kobling mellom maskinvare og programmer
  - Kommandolinje, for eksempel: en terminal
  - Grafisk brukergrensesnitt (GUI): Grensesnitt med vinduer, ikoner, menyer og peker
  - Selve grensesnittet ligger ikke i operativsystem**kjernen**
- Eksempel: ***X*** (i Linux se `man X`)
  - Vindushåndteringssystem som kjører på de fleste ANSI C og POSIX (Portable OS Interface for UNIX) kompatible systemer
  - Bruker kommunikasjon mellom prosesser for å få input fra, og sende output til flere programmer
- Andre: Desktop Window Manager (Windows), Wayland (Linux)

Brukernivå vs. Kjernenivå (Beskyttelse)

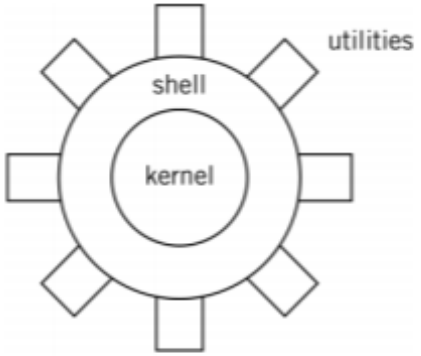
- Mange OS skiller mellom brukernivå og kjernenivå, grunnen til dette er sikkerhet og beskyttelse
- Operativsystemet kjører i kjernemodus
  - *Real mode*
  - Tilgang til hele minne
  - Alle instruksjoner kan bli kjørt
  - Går forbi all sikkerhet
  - Bytte til *Protected mode* eller *Long mode* snarest mulig
- Vanlige programmer og flere API'er kjører i brukernivå
  - *Protected mode*
  - Bruke privilegerte nivåer (x86: «ring», ARM: «mode»)
  - Ytre ringer
    - kan ikke kontakte maskinvare eller enheter direkte, kun gjennom et API
    - tilgang kun til et begrenset minneområde
    - begrenset tilgang til prosessoren



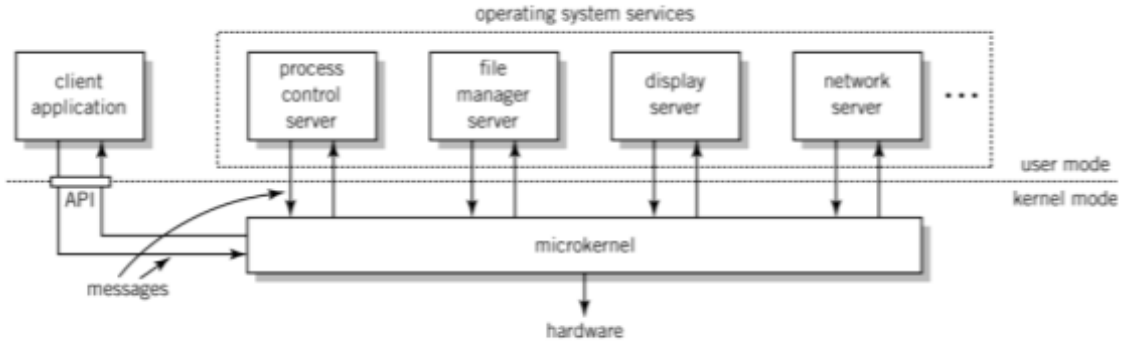
Organisering av operativsystemer

- Det er ikke noe bestemt standard for hvordan et operativsystem er bygd opp (i motsetning til nettverksprotokoller, etc.), men det er to hovedmodeller:

- Monolittisk kjerne ("the big mess"):
  - All beskyttet kode kjører i samme beskyttelsesområde
  - Effektivt: alle komponenter kan kommunisere direkte (funksjonskall, skrive i minne)
  - Feil kan få følger for andre komponenter enn der de oppstår
  - Store og komplekse
  - BSD UNIX, Linux, Windows 10 (++), …



- Mikro-kjerner
  - Små moduler med minimal funksjonalitet (avbrudd, minne, prossessor, osv.)
  - Så mange som mulig av tjenestene er skriver i brukermodus
  - Mye beskjeder frem og tilbake (ineffektivt)
  - Vanskelig å utføre ting i riktig rekkefølge
  - Vanskelig feilsøking pga grensene
  - Liten, modulær, utvidbar, portabel, …
  - MACH, Windows NT, …



Oppsummering

- Operativsystemer skjuler kompleksiteten i maskinvare
- Tillater flere brukere å bruke maskinen samtidig
- API for å tillate programmer å bruker resurser i maskinen
- Grunnleggende sikkerhetsmekanismer er innebygd