Seminar 5 (LMC 2 + operativsystemer)

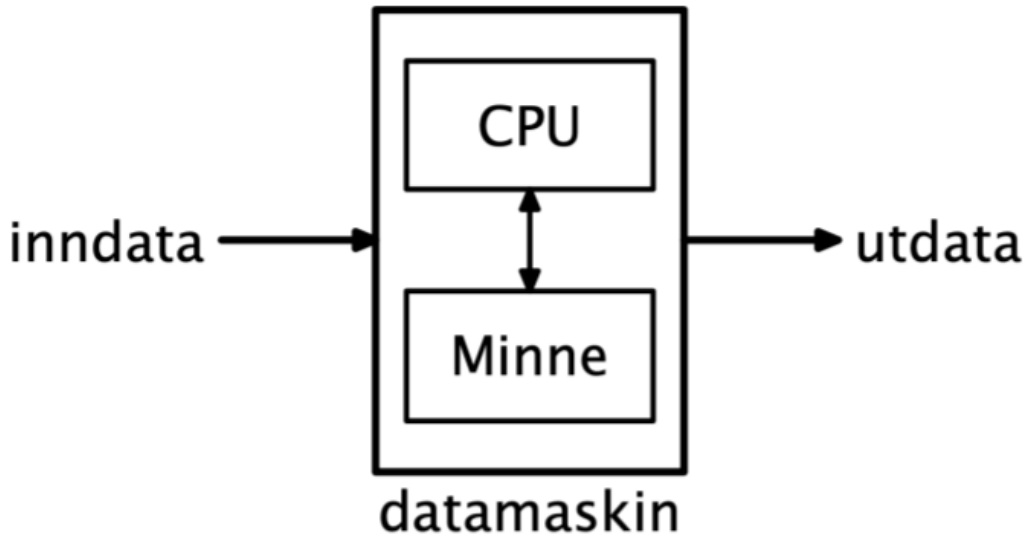
Ekte Prosessorer

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/96d8f86e-05c5-4fd1-9348-c58f7fca0398/uke-38-mer-om-prosessoren-og-sikker-programmering.pdf

En lenke til LMC

Prosessoren x86-64

- Bygget på samme Von Neumann-arkitekturen
- Har i tillegg cache mellom minne og CPU



Eksekveringsløkken

Samme som i **LMC**

1. Hent instruksjonen i minnet; programtelleren angir hvor.

2. Dekod instruksjonen i operasjon og adresse; de legges i instruksjonsregisteret og adresseregisteret.
3. Utfør instruksjonen.
4. Om aktuelt, skriv svaret i minnet/register.

x86-64 har pipeline for å kunne jobbe raskere. **LMC** har det ikke.

Dere kommer tilbake til pipeline senere i kurset.

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

LMC: 4, lagrer tall mellom -999 og +999

x86-64: 16, lagrer info i bytes (1, 2, 4, eller 8)

Minne

LMC: 100 celler, tall mellom -999 og +999

x86-64: potensielt svært mange (inntil 17 000 000 TB), inneholder en byte

CPU-arkitektur

LMC: RISC - kun enkle instruksjoner, men disse er utført veldig raskt og effektivt

x86-64: CISC - både enkle og kompliserte instruksjoner, men utført litt treigere

Operativsystemer

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/59a8e48d-af56-4e58-87dc-f6a3eda25241/in1020-uke38-operativsystemer.pdf

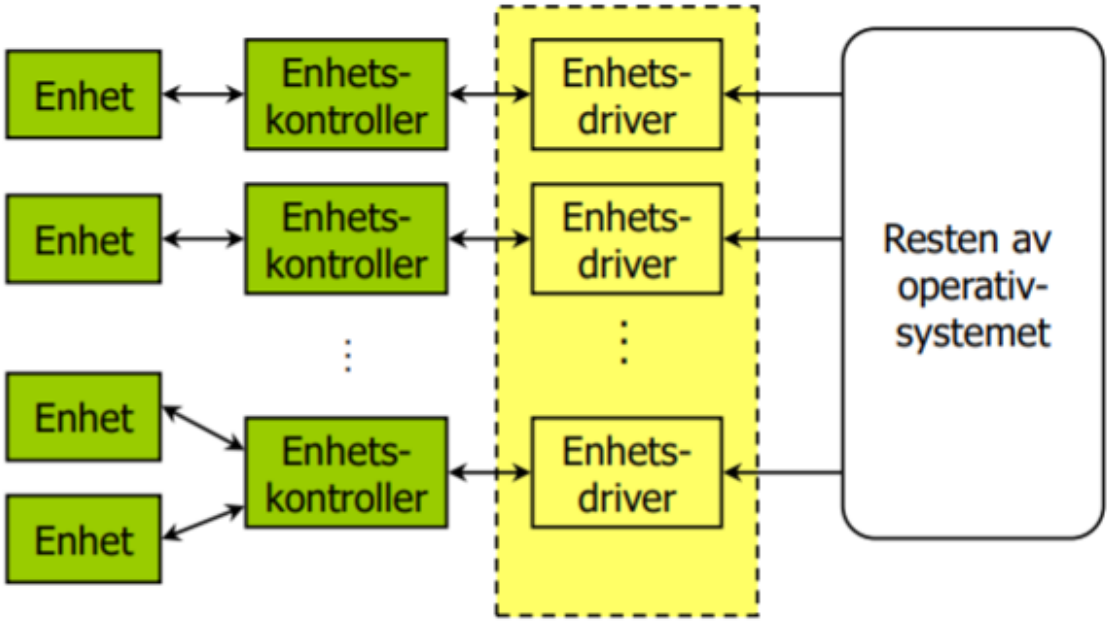
Hovedkomponenter i et operativsystem



"Synlige" for bruker

- Enhetskontroll

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



- Brukergrensesnitt
  - Koblingen mellom bruker og datamaskinen
  - Alt vi kan gjøre på en datamaskin: GUI (Graphic User Interface), terminal, etc.
- Filsystem (filkontroll, filhåndtering)
  - Mekanismer for at en bruker skal kunne lage, slette, modifisere og manipulere filer.

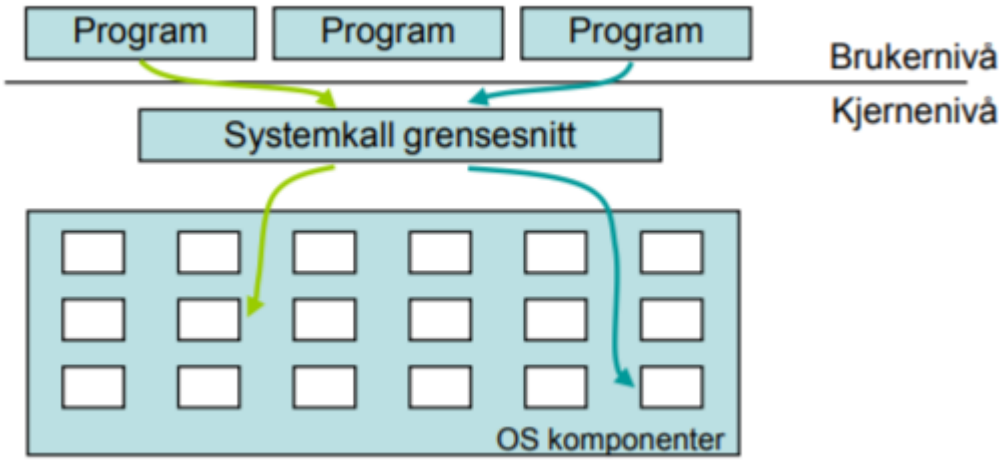
"(Relativt) Transparent"

- Prosessorkontroll (eller prosesskontroll)
  - Tilby mekanismer for at systemet effektivt og rettferdig skal kunne dele CPU for programmene som kjører
- Minnekontroll
  - Tilby mekanismer for at systemet skal kunne fordele minneresurser
    – allokerer plass til programmer

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

Systemkall

- Grensesnittet mellom operativsystemet og programmer (bruker) er definert med hva vi kaller et systemkall



Brukernivå vs. Kjernenivå (Beskyttelse)

- Mange OS skiller mellom brukernivå og kjernenivå, grunnen til dette er sikkerhet og beskyttelse
- Operativsystemet kjører på kjernenivå
    - *Real Mode* (ligner Superuser mode i mobiltelefon)
    - Tilgang til hele minne
    - Alle instruksjoner kan bli kjørt
    - Går forbi all sikkerhet
    - Bytte til *Protected mode* eller *Long mode* snarest mulig
- Vanlige programmer kjører i brukernivå
    - Protected mode
    - kan ikke kontakte maskinvare eller enheter direkte, kun gjennom et API (application programming interface)

- tilgang kun til et begrenset minneområde
- begrenset tilgang til prosessoren

Ukeoppgaver

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/98795ae1-294f-4e81-be8a-a1f40ffc296d/oppgaver-lmc-2.pdf

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/7f427dc5-c215-4eb9-aa91-4438f32bd55d/fasit-lmc-2.pdf

Oppgave 1

While-løkket som subtraherer b fra a inntil resten er et minimalt positiv tall mulig (eller 0).

Assemblerkode:

```
start   INP        // Les inn a
        STA a      // Lagre a
        INP        // Less inn b
        STA b      // Lagre b
        LDA a      // Hent a
subt    SUB b      // Subtraher b fra a
        BRP subt   // Gjenta så langt akkumulatorverdien er ≤0
        ADD b      // Hvis den er negativ, så legger vi til b en gang for å få et positivt tall (eller 0)
slutt   OUT        // Skrive ut resten.
        HLT        // "Halt" stopper alt.


// data-celler
a       DAT 0
```

b DAT 0

Oppgave 2

AAb

Oppgave 3

20

Oppgave 4

c)