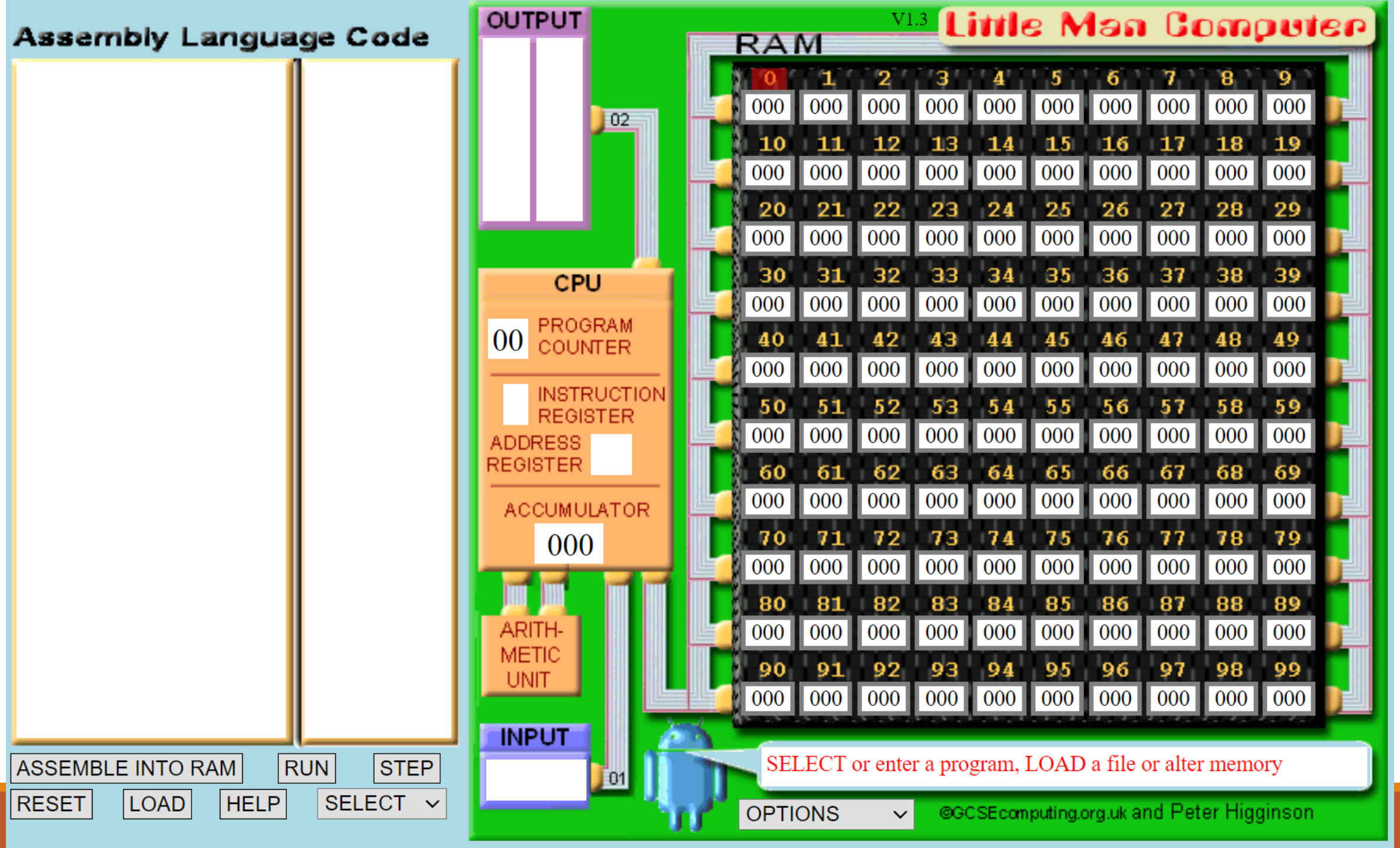
IN1020 uke3

GRUPPE 6

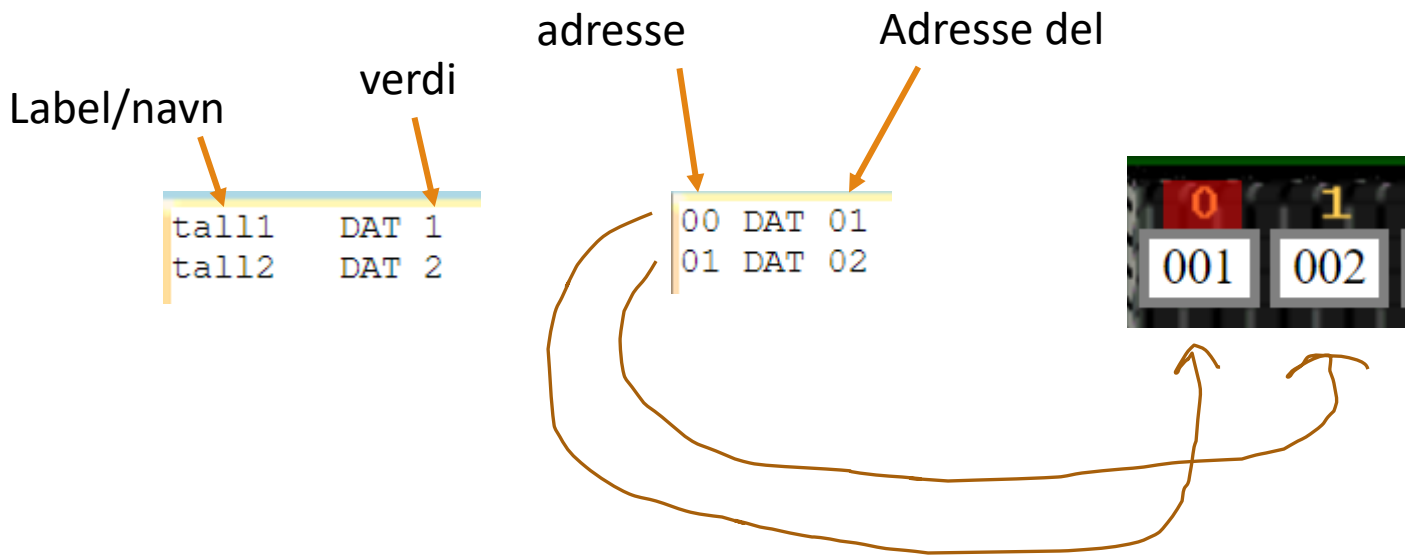
Plan for dagen

- Menti
- Repetisjon av LMC
- Live koding oppgaver
- Tips til oblig1
- Litt OS – operativsystemer
- Jobbe med ukesoppgaver eller oblig1


Assembly Language Code
ASSEMBLE INTO RAM
RUN
STEP
RESET
LOAD
HELP
SELECT
OUTPUT
02
CPU
00
PROGRAM COUNTER
INSTRUCTION REGISTER
ADDRESS REGISTER
ACCUMULATOR
000
ARITH-METIC UNIT
INPUT
01
V1.3
Little Man Computer
RAM
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99
000
SELECT or enter a program, LOAD a file or alter memory
OPTIONS
©GCSEcomputing.org.uk and Peter Higginson

DAT

- Brukes til å deklarere en verdi i minnet
- Initialiseres med en verdi
- Skriver (DAT <verdi>) eller (<label> DAT <verdi>) for å deklarere en variabel med en verdi



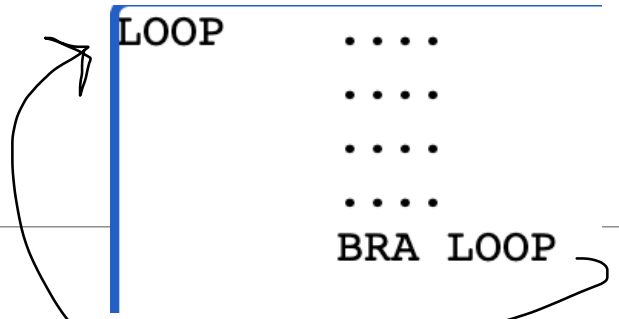
Alle instruksjonene i LMC

Kode	Navn	Beskrivelse
0xx	HLT	Stopper eksekveringen
1xx	ADD	Adderer verdien i angitt minnelokasjon med akkumulatoren
2xx	SUB	Subtraherer verdien i angitt minnelokasjon med akkumulatoren
3xx	STA	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 hvis akkumulatoren er 0
8xx	BRP	Hopper 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)

STA og LDA

- STA tar verdien som er i akkumulatoren og laster det opp i en address
- LDA laster inn tallet fra en adresse inn til akkumulatoren
- Det går alltid an å overskrive verdier ved å bruke STA flere ganger
- LDA brukes ofte sammen med STA, i praksis så gjør vi følgende:
  - Bruker LDA til å laste inn tall i akkumulatoren
  - Deretter gjøre beregninger, enten addere eller subtrahere
  - Bruker STA til å lagre den nye verdien i en ny adresse eller samme adresse

Løkker og hopping



BRA (Branch):

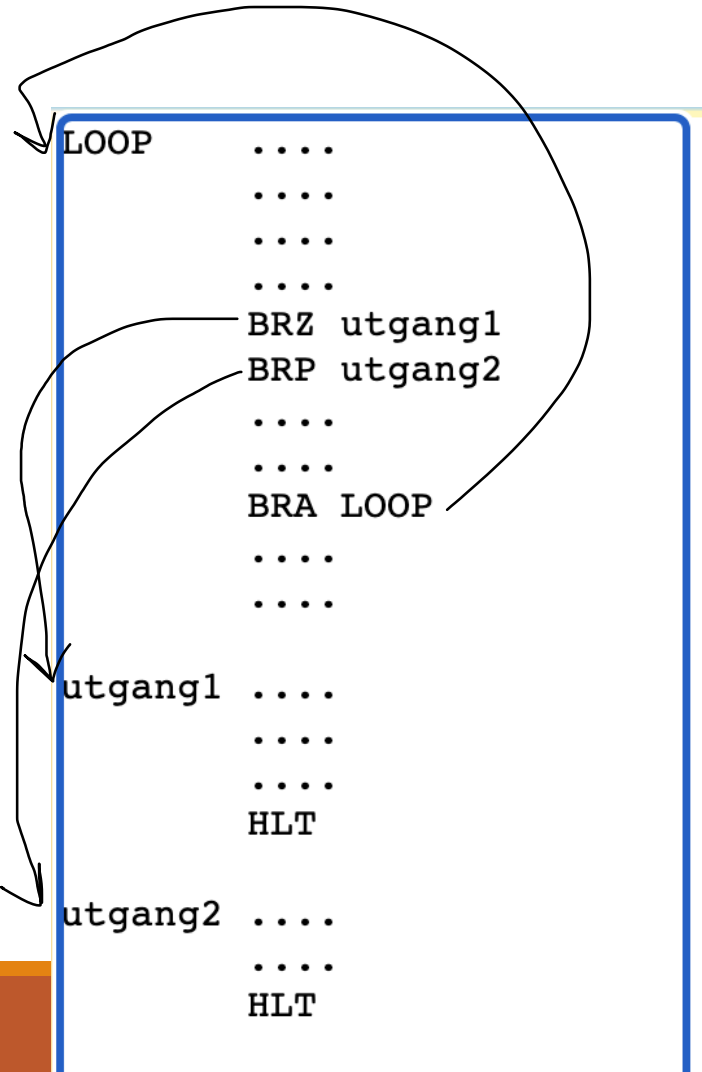
- Hopper ALLTID til label

BRZ (Branch if zero):

- Hopper til label HVIS akkumulatoren er null

BRP (Branch if positive):

- Hopper til label HVIS akkumulatoren er null eller positiv



Live koding oppgaver

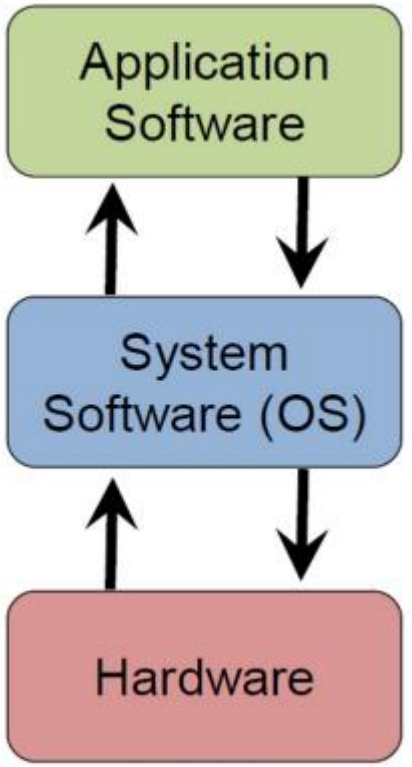
- Lage program som summerer og tall med hverandre
- Lage et program som tar 2 tall fra brukeren og plusser de med hverandre
- Lage program som teller fra 10 til 0
- Lage program som teller fra 0 til 10
- Lage et program som deler tall på 2
- Lage et program som skriver ut navnet mitt

Tips til oblig 1

- Løs oppgavene steg for steg:
  - Skriv litt kode først, så gjør den og test om den funker
  - Deretter så kan du utvide programmet mer
  - IKKE gjør hele obligen i en run også test på slutten
  - Hvis du sliter å anbefaler vi å gjøre innføring til LMC, link
- Skriv koden et sted der du kan lagre den f.eks. Atom, VS code, word, wordpad, eller på papir
- Kommentarer i LMC kan ha bugs, så pass på dette
- Pass alltid på det som står i akkumulatoren stemmer og at det som står i adressene stemmer:
  - Man sjekke om beregninger er feil ved å enten sjekke akkumulatoren eller adressene
- Ta en skikk på python programmet som er i oppgavesettet
- Hint: Se på gange eksempelet med Kristoffer Robin

OS - operativsystemer

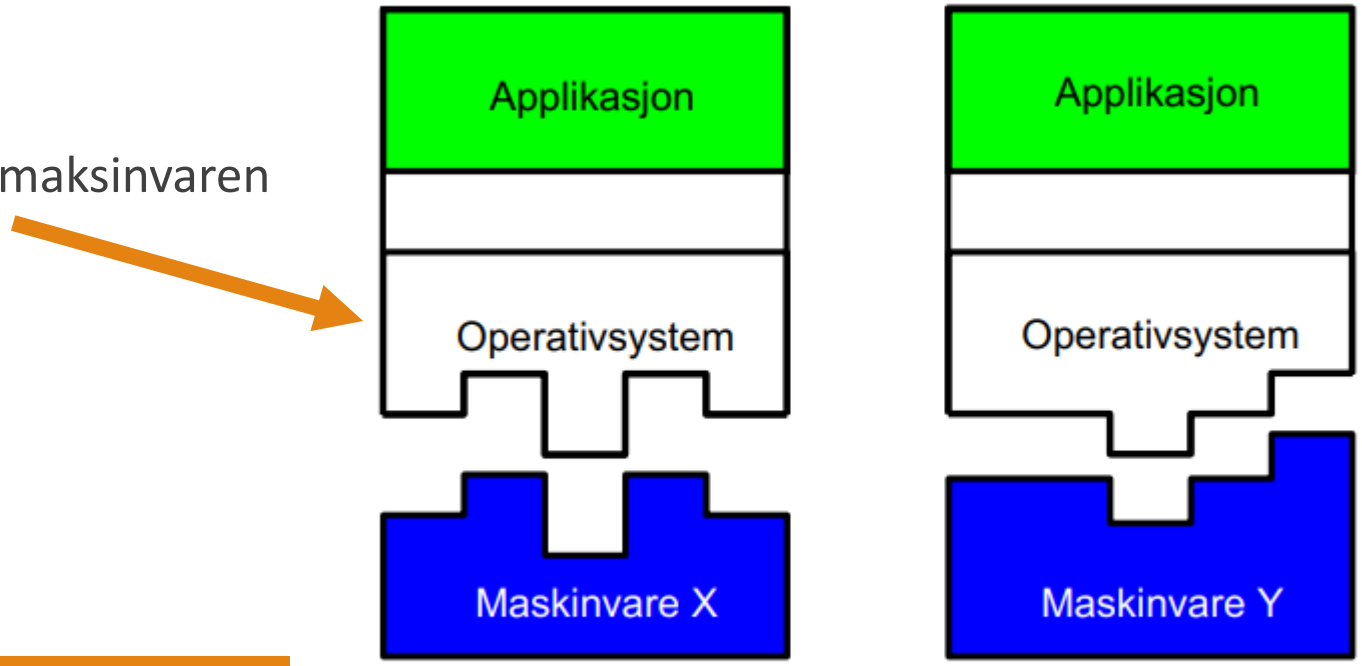
- Operativsystem = dirigenten
- Kontrollerer alt i en datamaskin
- OS er lag i software til datamaskinen:
  - Flere brukere samtidig
  - Flere programmer samtidig
  - Mellomlag som brukes til å kommunisere med programmer og software



Forskjellig maskinvare

- Operativsystemet gjør at selv om det er forskjellige hardware så kan man fortsatt levere samme applikasjoner for folk:
  - Gjør at man kan utvikle programvare for maskiner med forskjellige hardware
  - Genialt siden, det gir hardware-utviklerne samme muligheter

- OS-en endrer form for å passe inn med maksinvaren

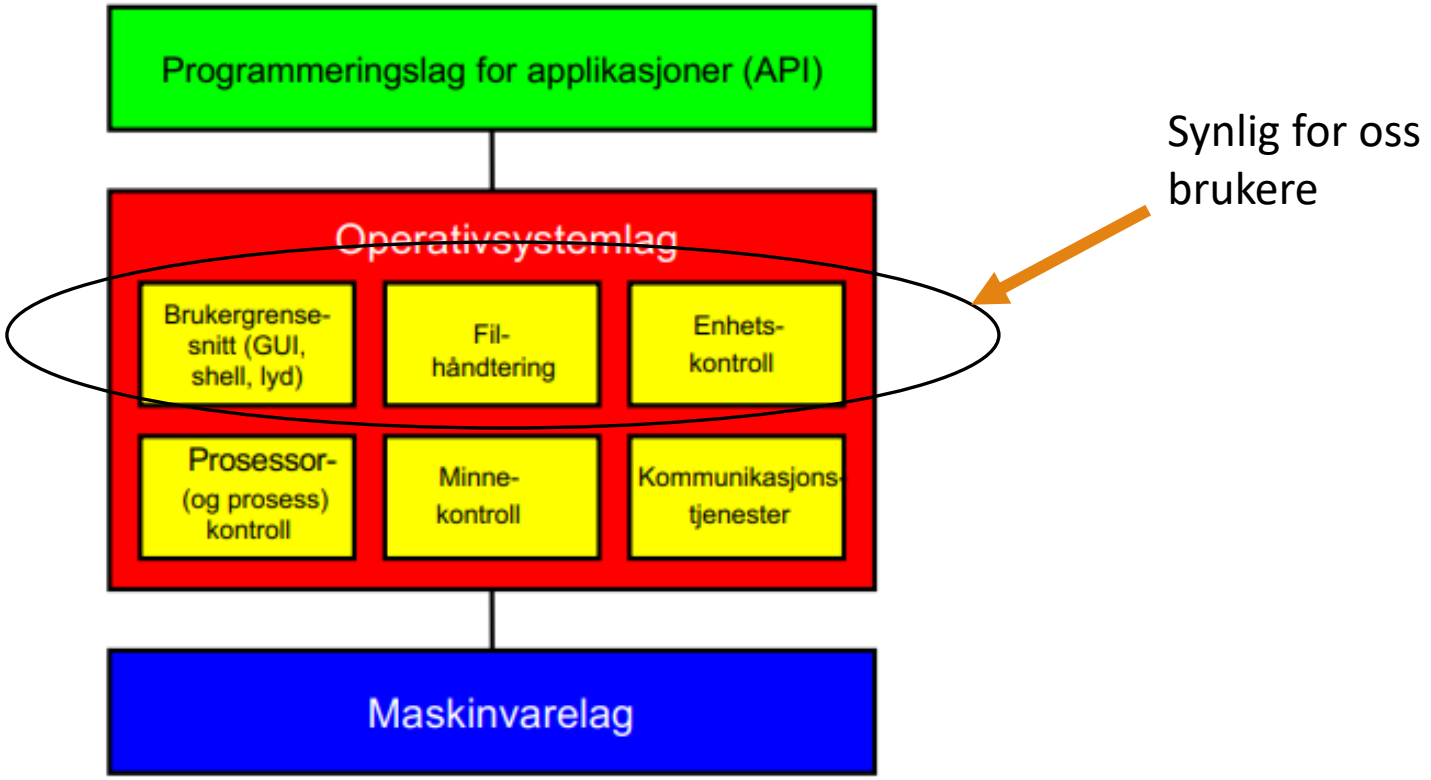


Forskjellige typer operativsystemer (OS)

- 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.

Hovedkomponenter i et operativsystem

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

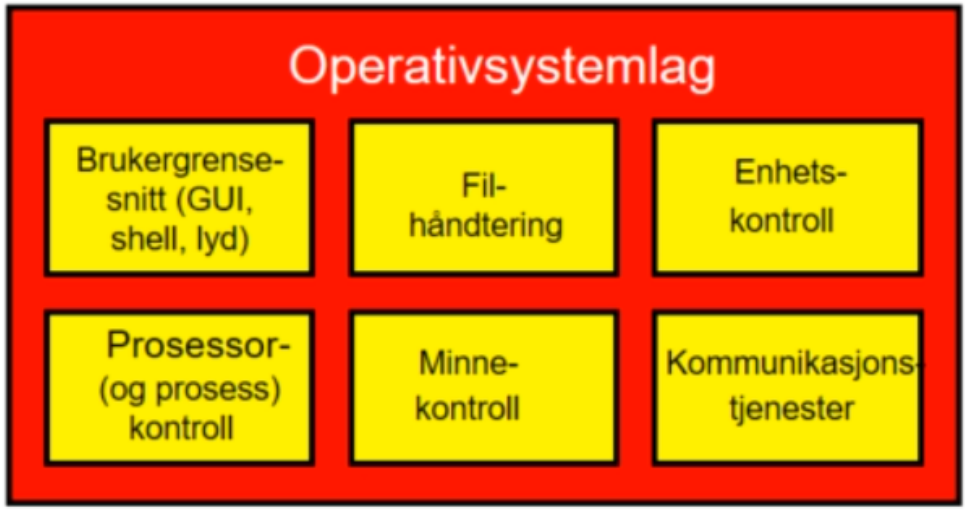


Hovedkomponenter - Detaljert

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



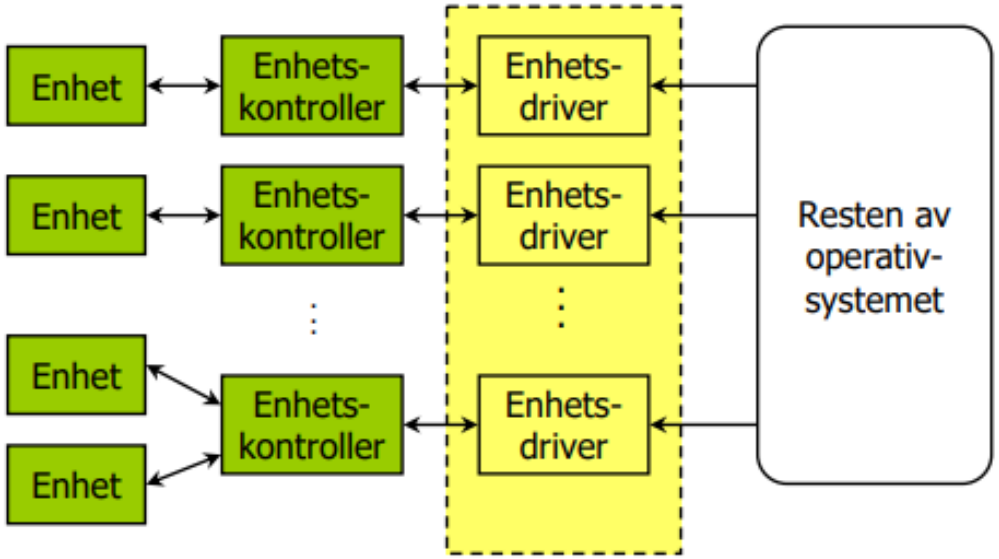
Enhetskontroll :
Gjøre at systemet skal kunne kontrollere enheter som er koblet til som for eksempel tastatur, mus, skjerm

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

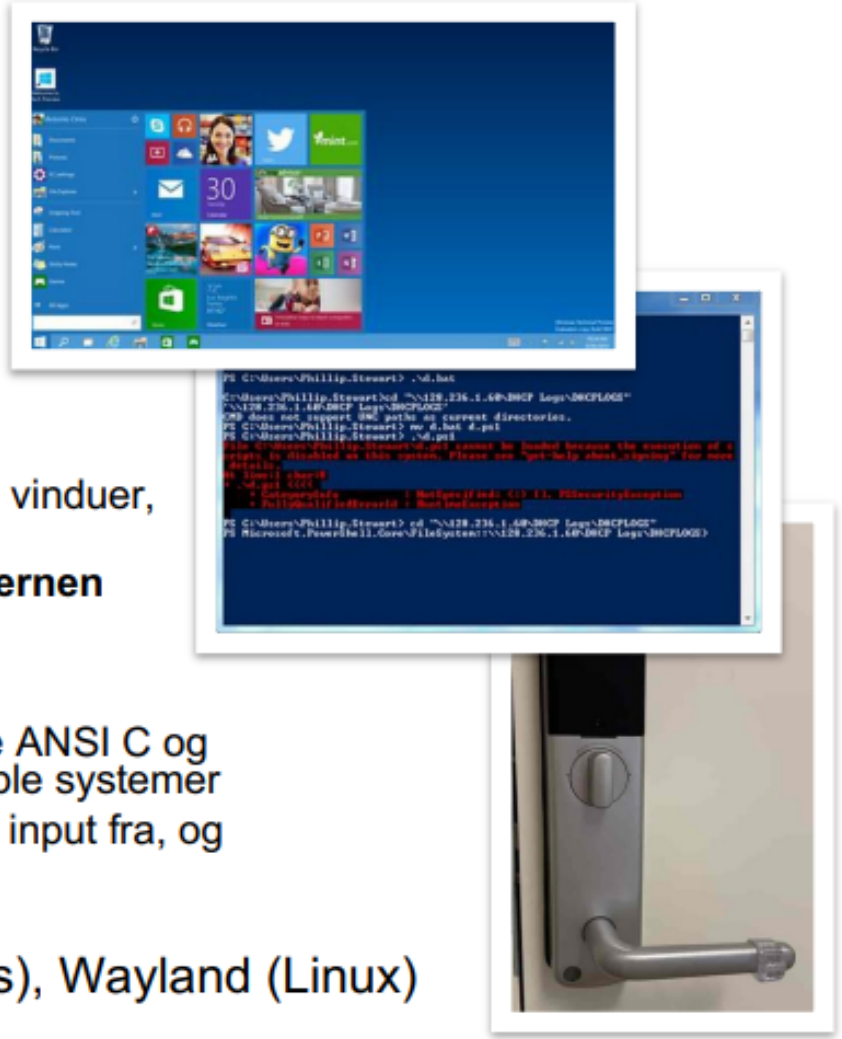
Minnekontroll :
Tilby mekanismer for at systemet skal kunne fordele minneresurser – allokerer plass til programmer

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)

Takk for i dag!

Spørsmål?

Send meg gjerne en mail til myhd@uio.no eller teams melding