

INF105 - 02

David Grellscheid

Bekymringer

- Kan PCen min kjøre det vi skal
- Hvordan eksamen blir
- Ingen forkunnskaper / vet ingenting om informatikk / Bekymret jeg skal ikke forstå alt
- Redd for at faget ikke blir avansert nok
- Forventning: Å være komplett fullstackutvikler etter kurset

Q&A

- Litteratur til de ulike temaene?

Q&A

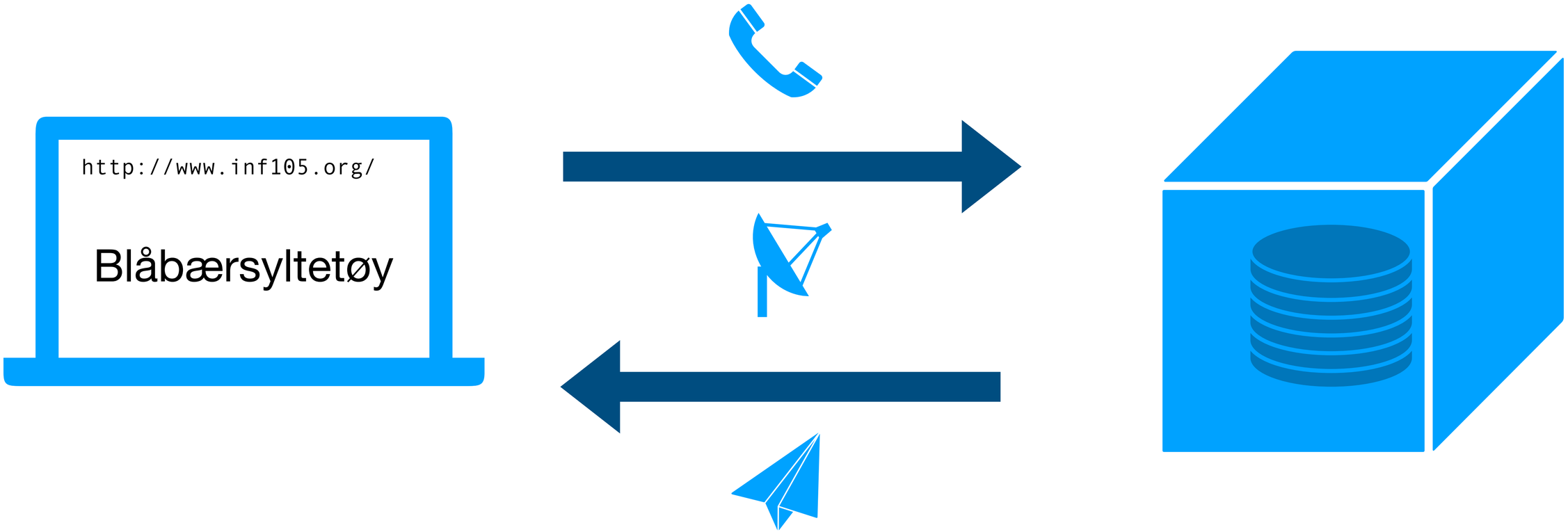
- Lenken i dag:



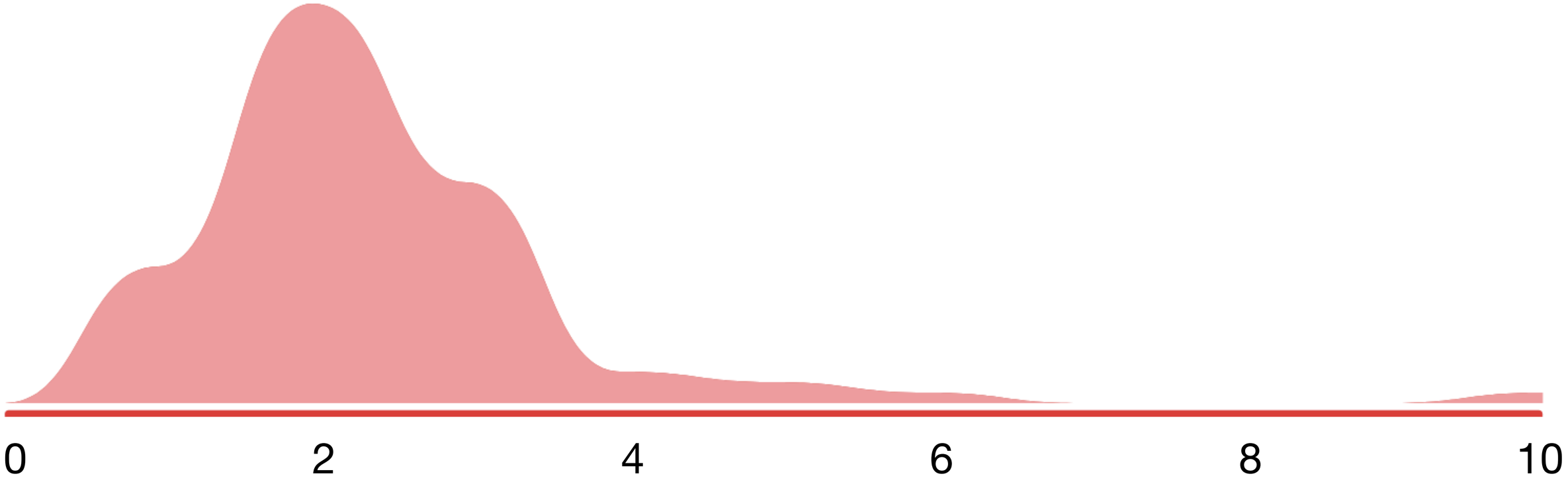
Oversikt

- Historie
- Hardware
- Operativsystem
- Kommandolinje / Terminal

Rød tråd



Hvor mange datamaskiner har du med deg nå?

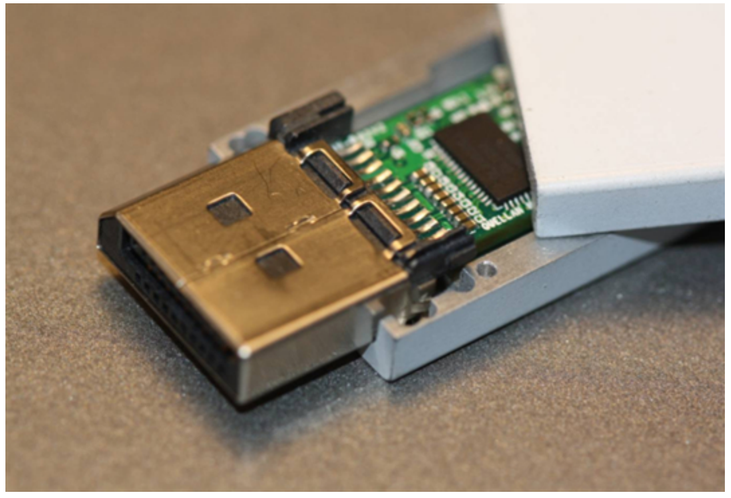








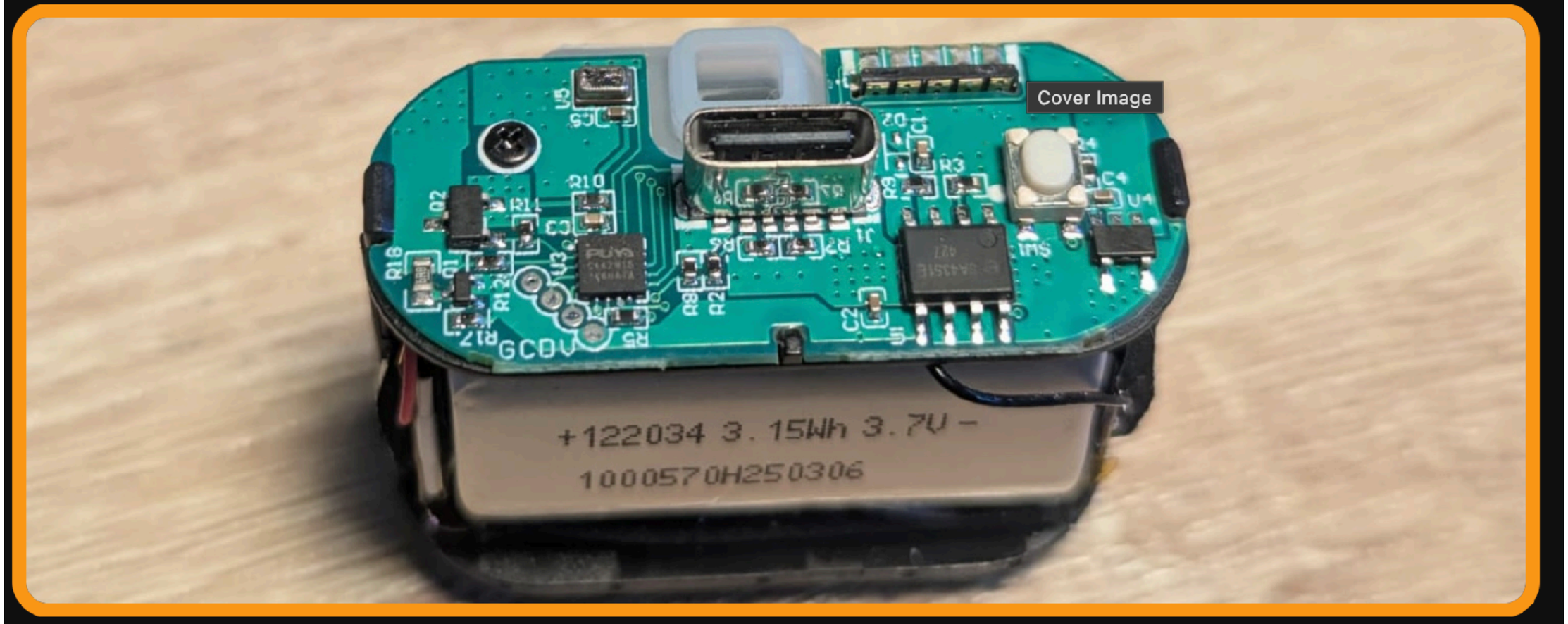

GAME BOY COLOR



Hosting a WebSite on a Disposable Vape

2025-09-13 :: Bogdan Ionescu :: 6 min read (1266 words) :: source :: report issue

#programming #arm #tools #electronics



https://bogdanthegeek.github.io/blog/projects/vapeserver/

Hva er en datamaskin?

Credits: NASA/JPL-Caltech

Hva er en datamaskin?

- 1950:
  What is a computer?



Credits: NASA/JPL-Caltech

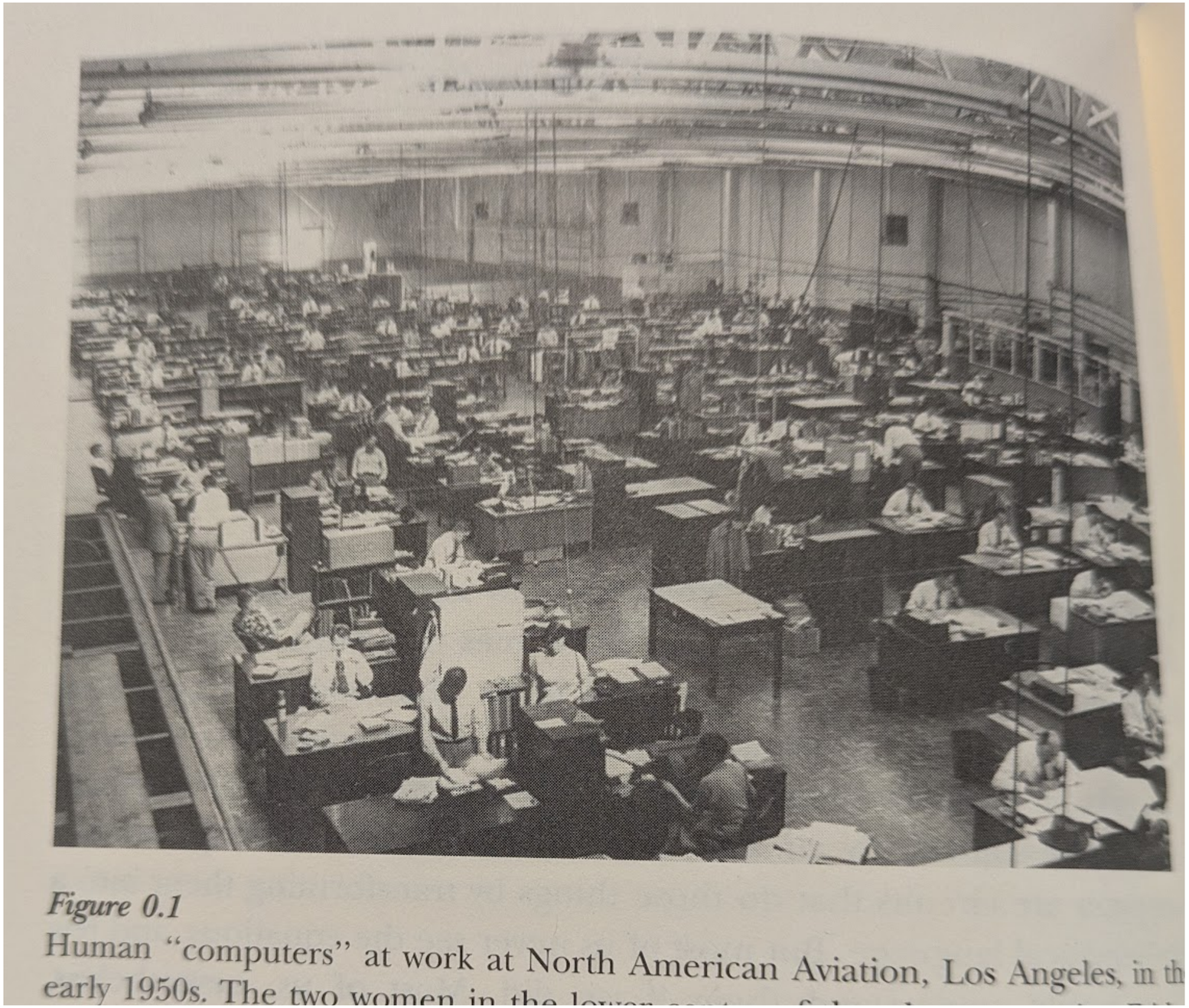


Figure 0.1
Human "computers" at work at North American Aviation, Los Angeles, in th
early 1950s. The two women in the lower

Paul E. Ceruzzi, "A History of Modern Computing"

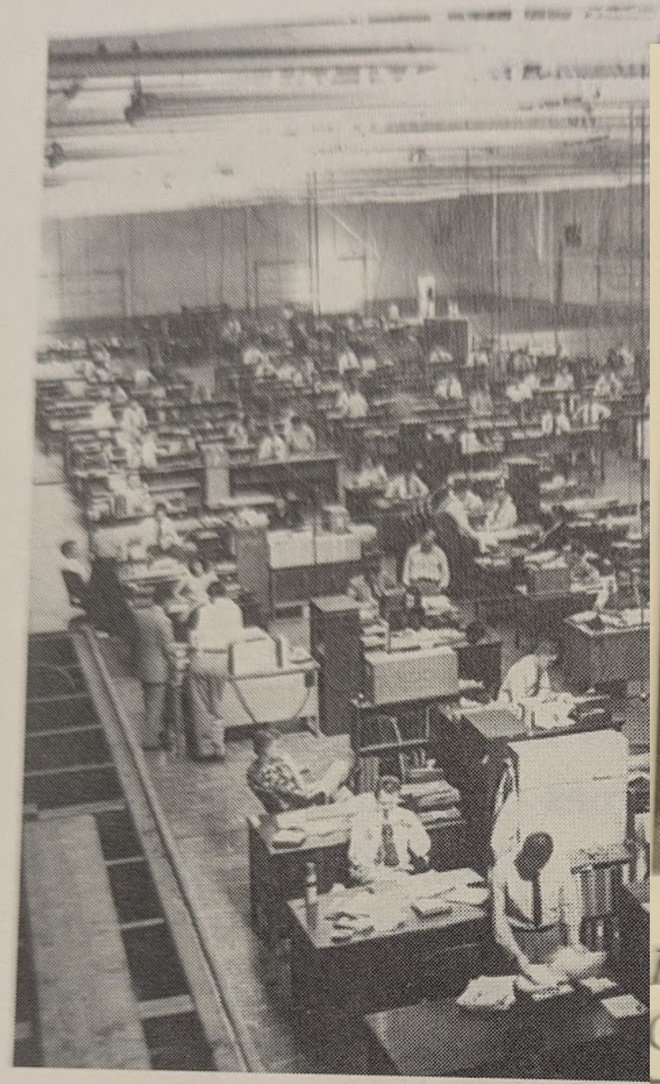


Figure 0.1
Human "computers" at work at North American Aviation, Los Angeles, in th
early 1950s. The two women in the lower

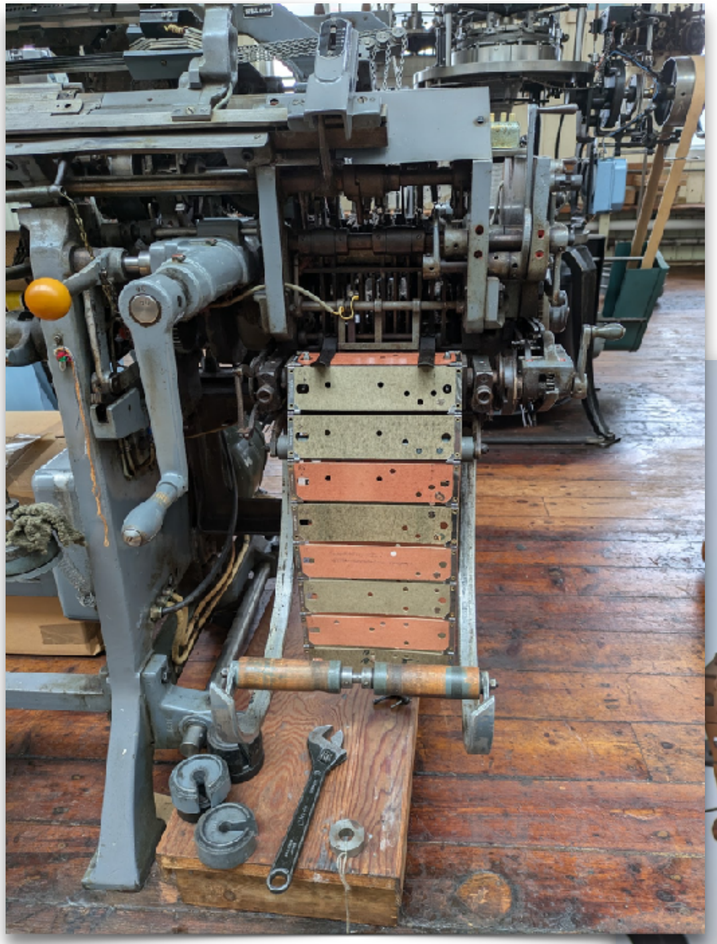
than

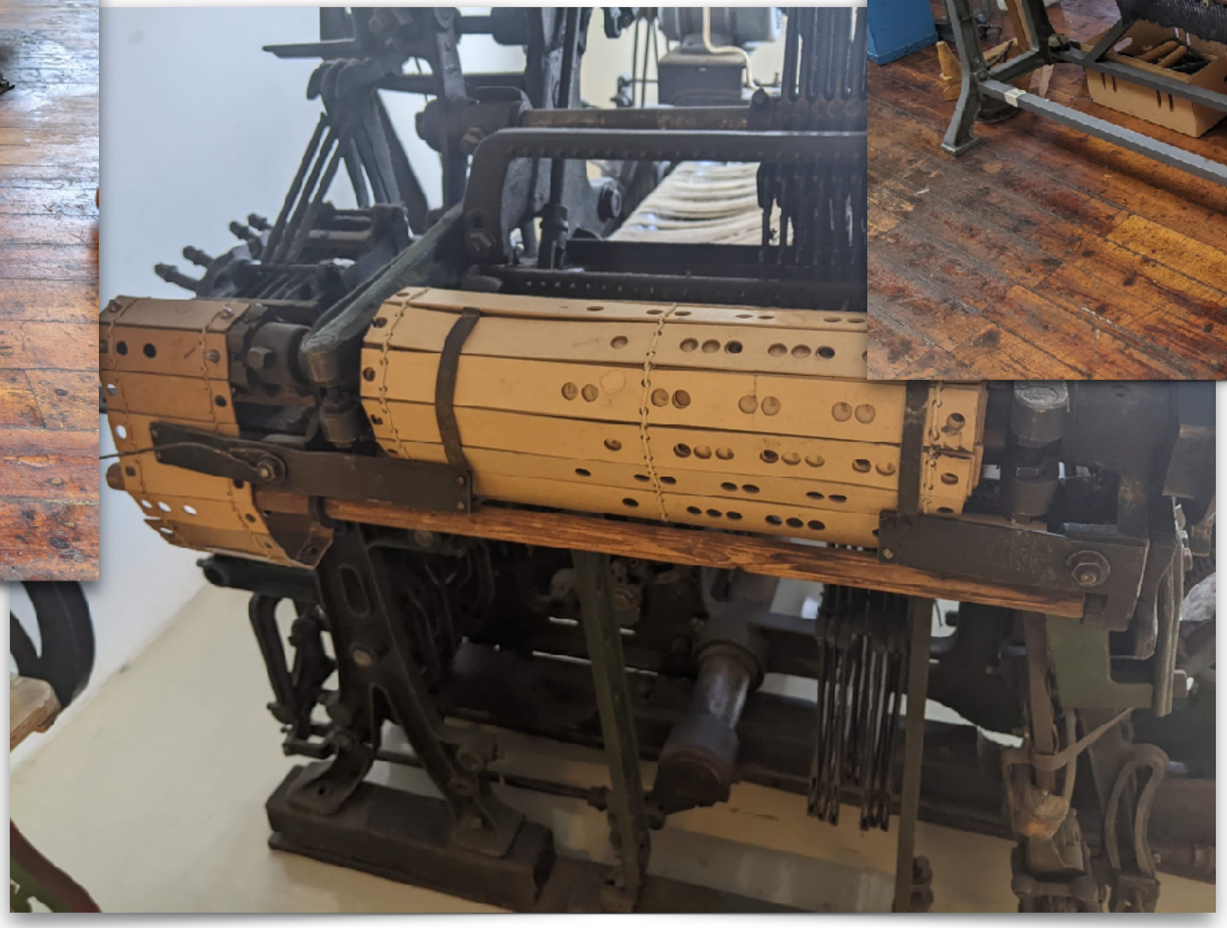


Figure 0.2
Computing with machines at the same company ten years later. A pair of IBM
7090 computers assist in the design and testing of the rocket engines that will

Paul E. Ceruzzi, "A History of Modern Computing"

- Fleksibel maskin som er *programmerbart*
- resultatet er ikke forhåndsbestemt i oppbyggingen





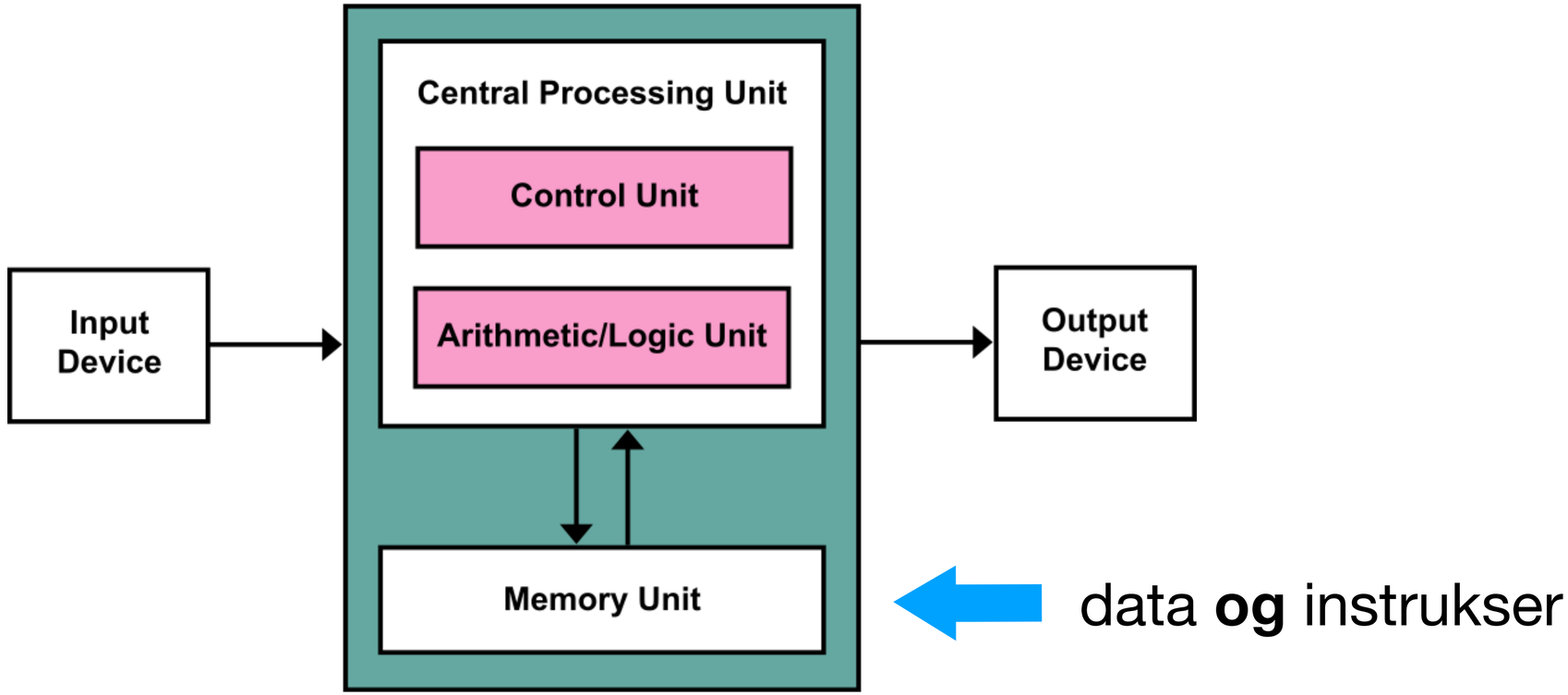


- S1E4 36:55

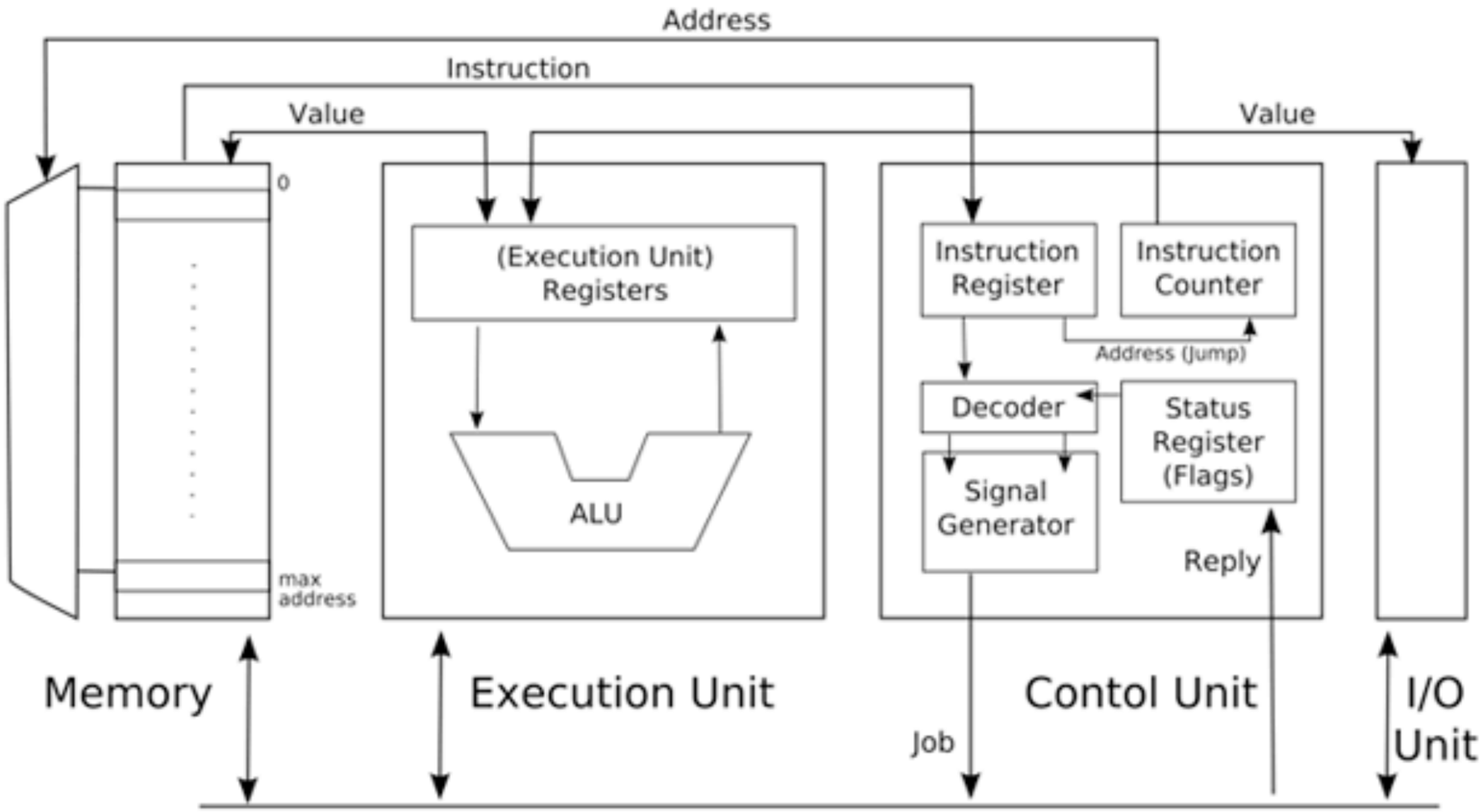
https://en.wikipedia.org/wiki/Connections_(British_TV_series)

... hopp over 150 år ...

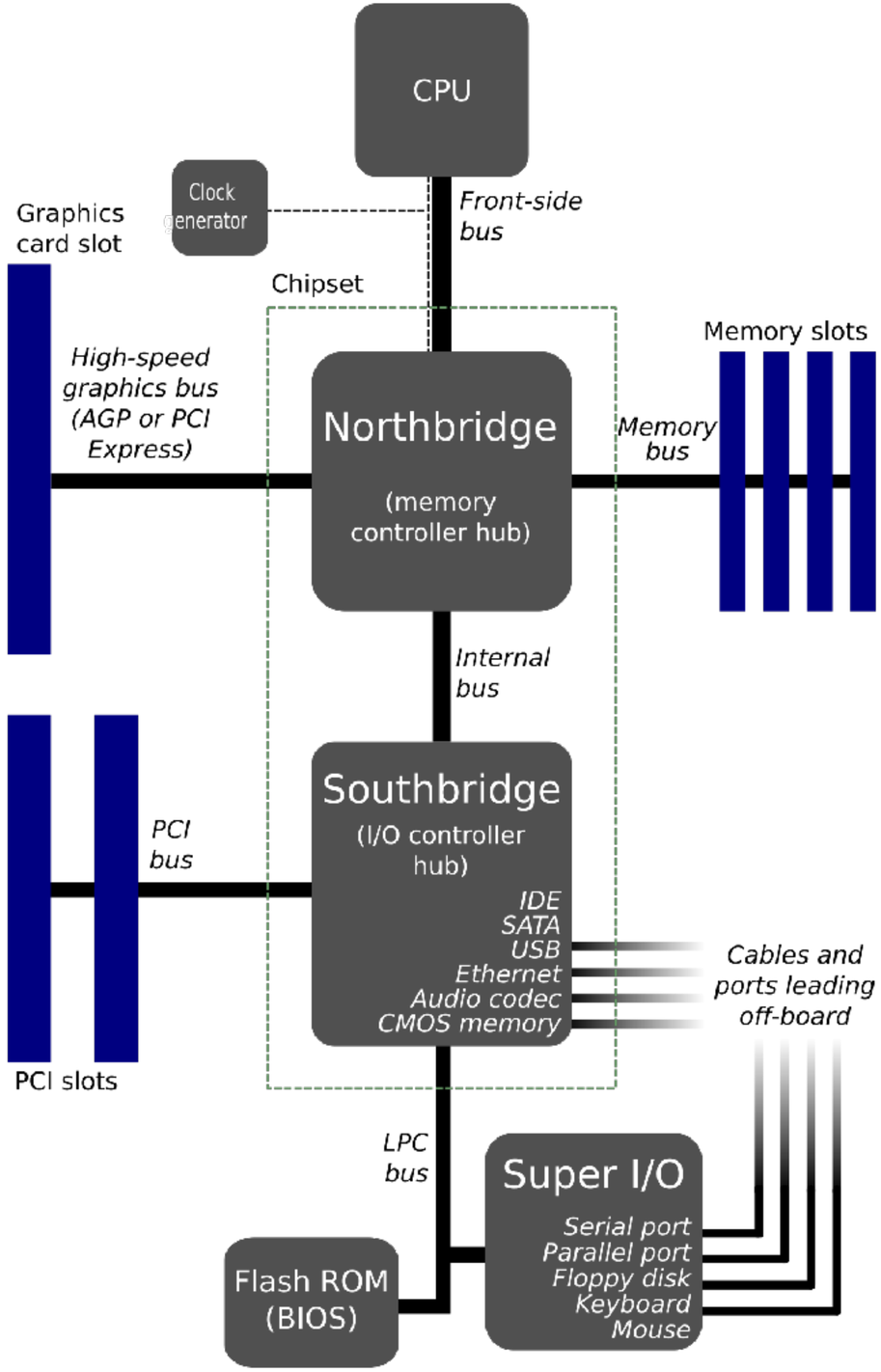
- Den moderne datamaskinarkitekturen kalles etter John von Neumann
- https://en.wikipedia.org/wiki/First_Draft_of_a_Report_on_the_EDVAC (1945)



Von-Neumann arkitektur



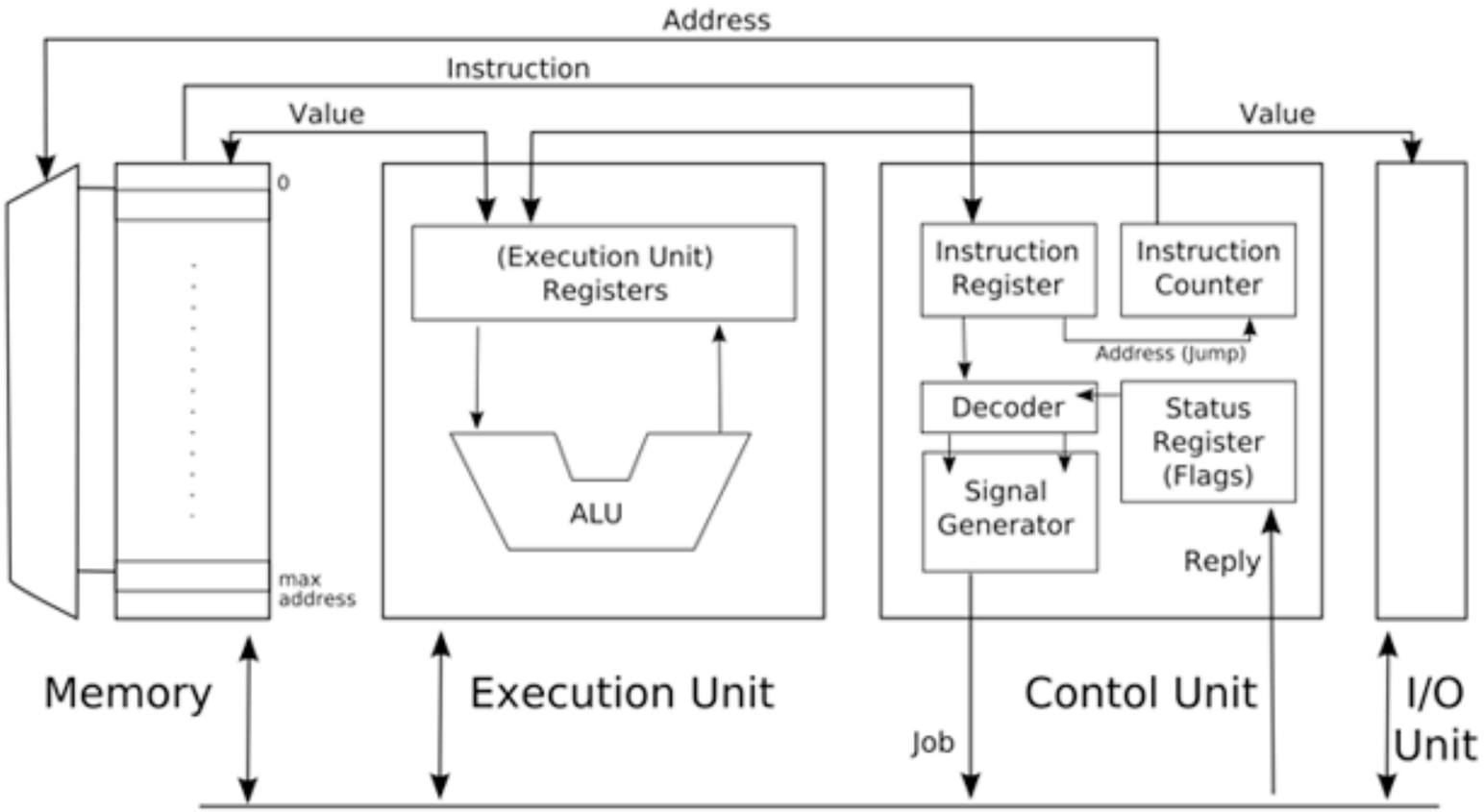
Både instrukser og data lagres sammen i minne.
De fleste moderne CPU har denne strukturen



https://en.wikipedia.org/wiki/Motherboard

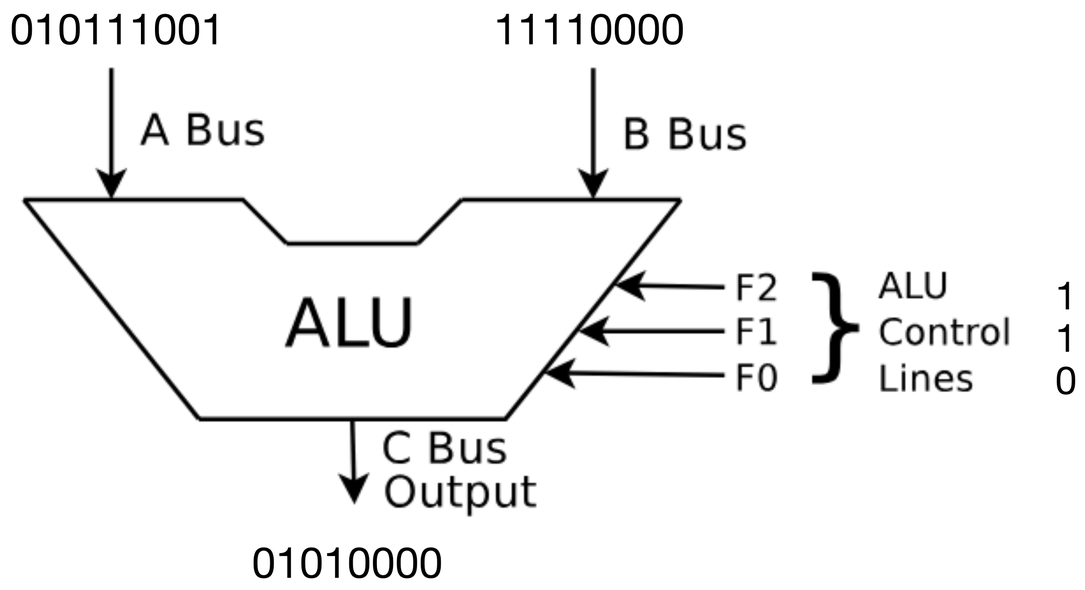
https://en.wikipedia.org/wiki/System_on_a_chip

Von-Neumann arkitektur



Både instrukser og data lagres sammen i minne.
De fleste moderne CPU har denne strukturen

Arithmetic Logic Unit



F2	F1	F0	Output
0	0	0	A
0	0	1	B
0	1	0	A+1
0	1	1	B+1
1	0	0	A+B
1	0	1	A-B
1	1	0	A AND B
1	1	1	A OR B

https://archive.org/download/computer-organization-and-design-fourth-edition-the-hardware-software-interface-/Step-by-Step%20Design%20and%20Simulation%20of%20a%20Simple%20CPU.2445296.pdf

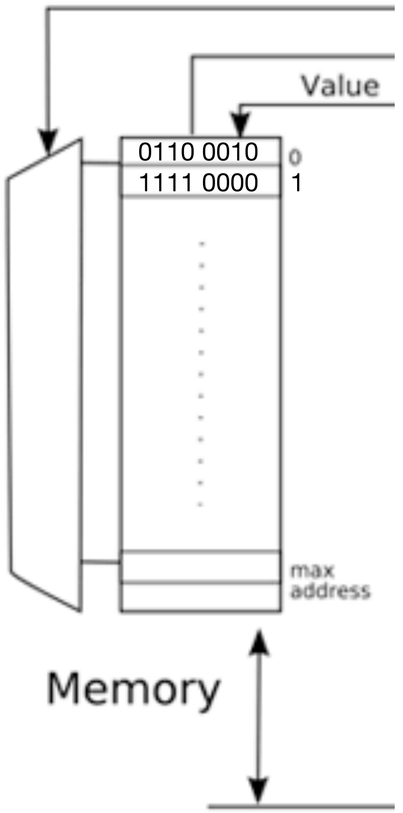
Minne

ROM - Read Only Mem.

RAM - Random Access Mem.

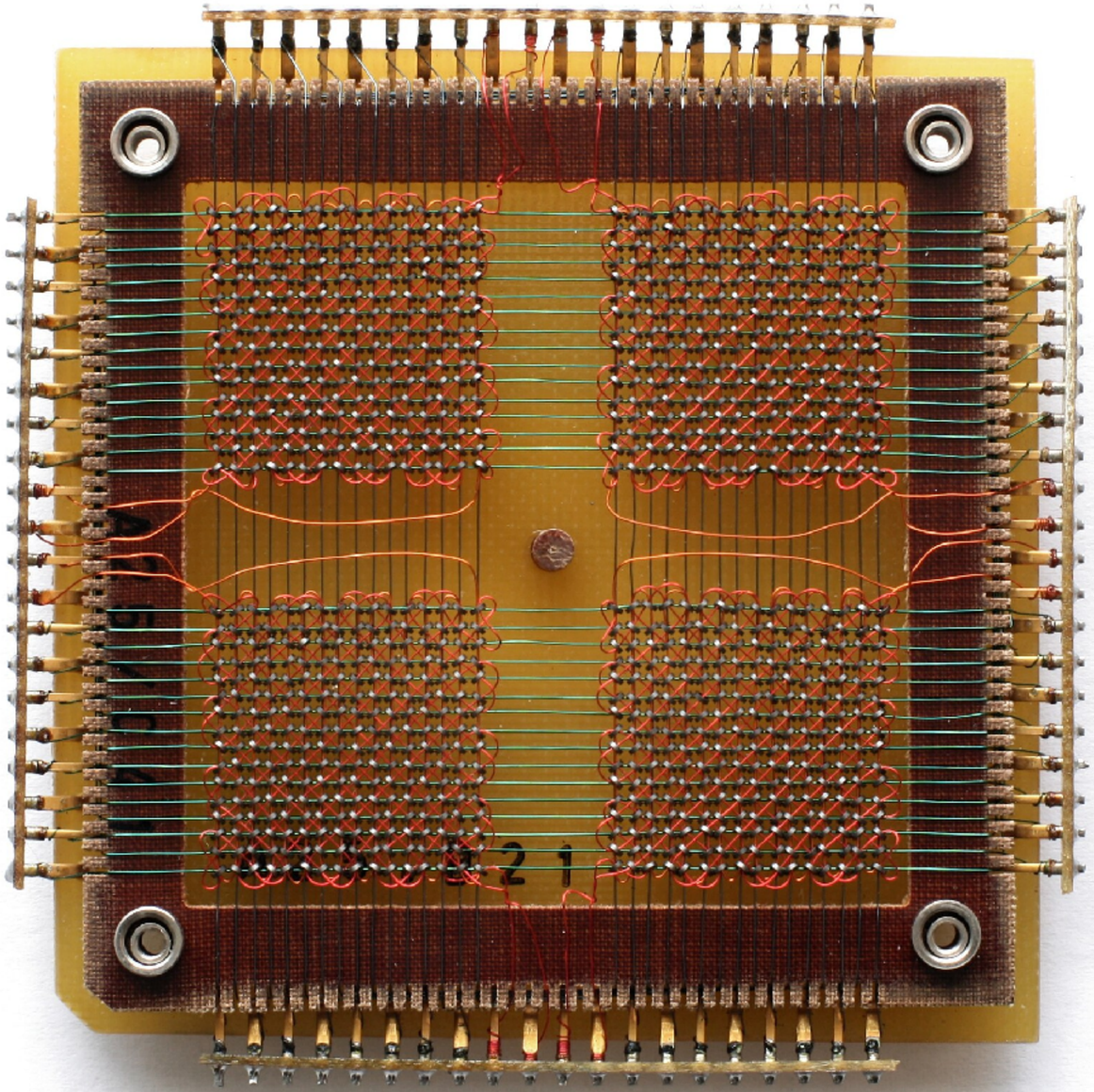
Adresse -> Innhold

0000 -> 0110 0010
0001 -> 1111 0000

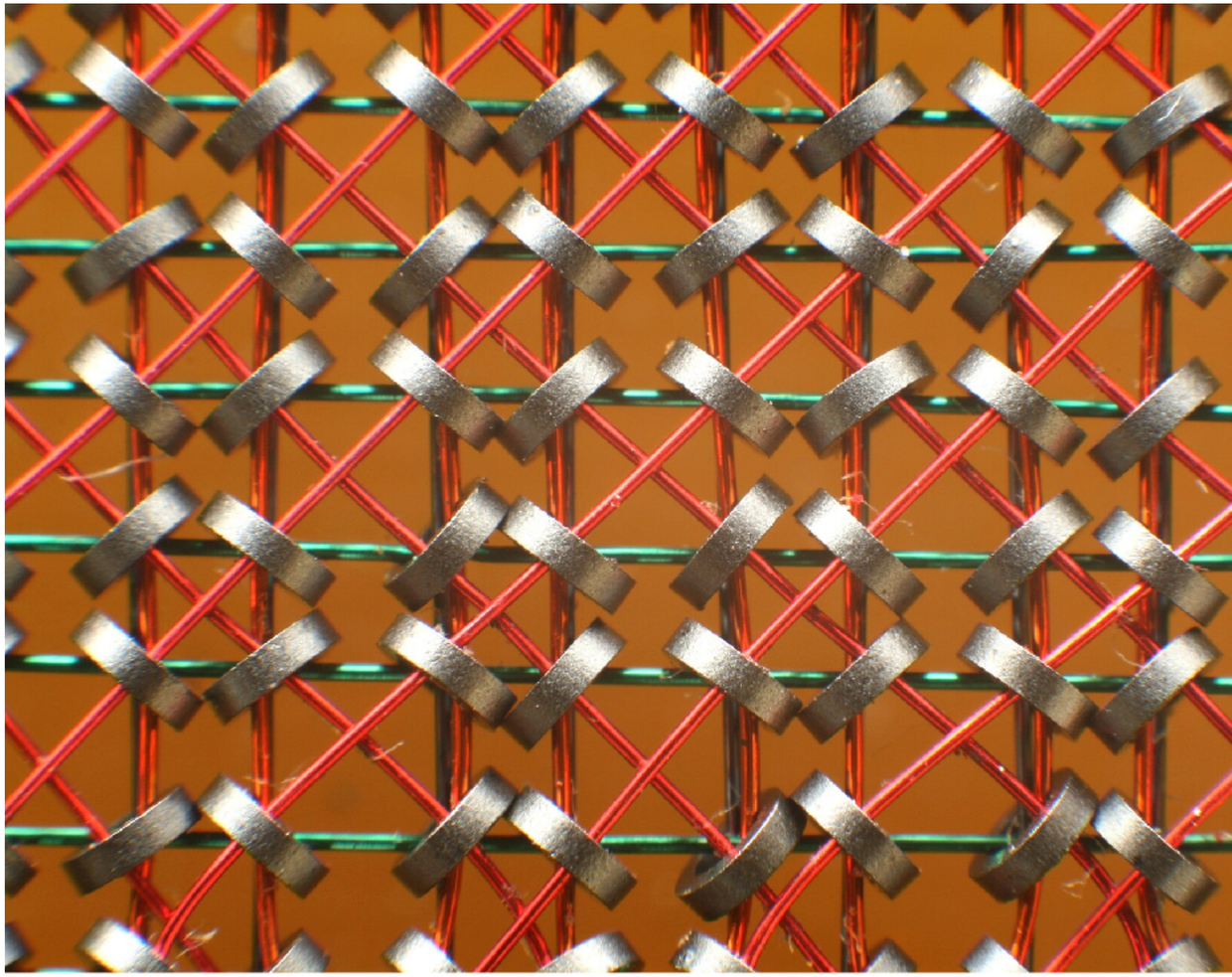


Innholdet 01100010 kan tolkes f.eks: som heltall 142, bokstav 'b', en instruks, en ny adresse, ...
Datatypen er ***ikke*** markert!

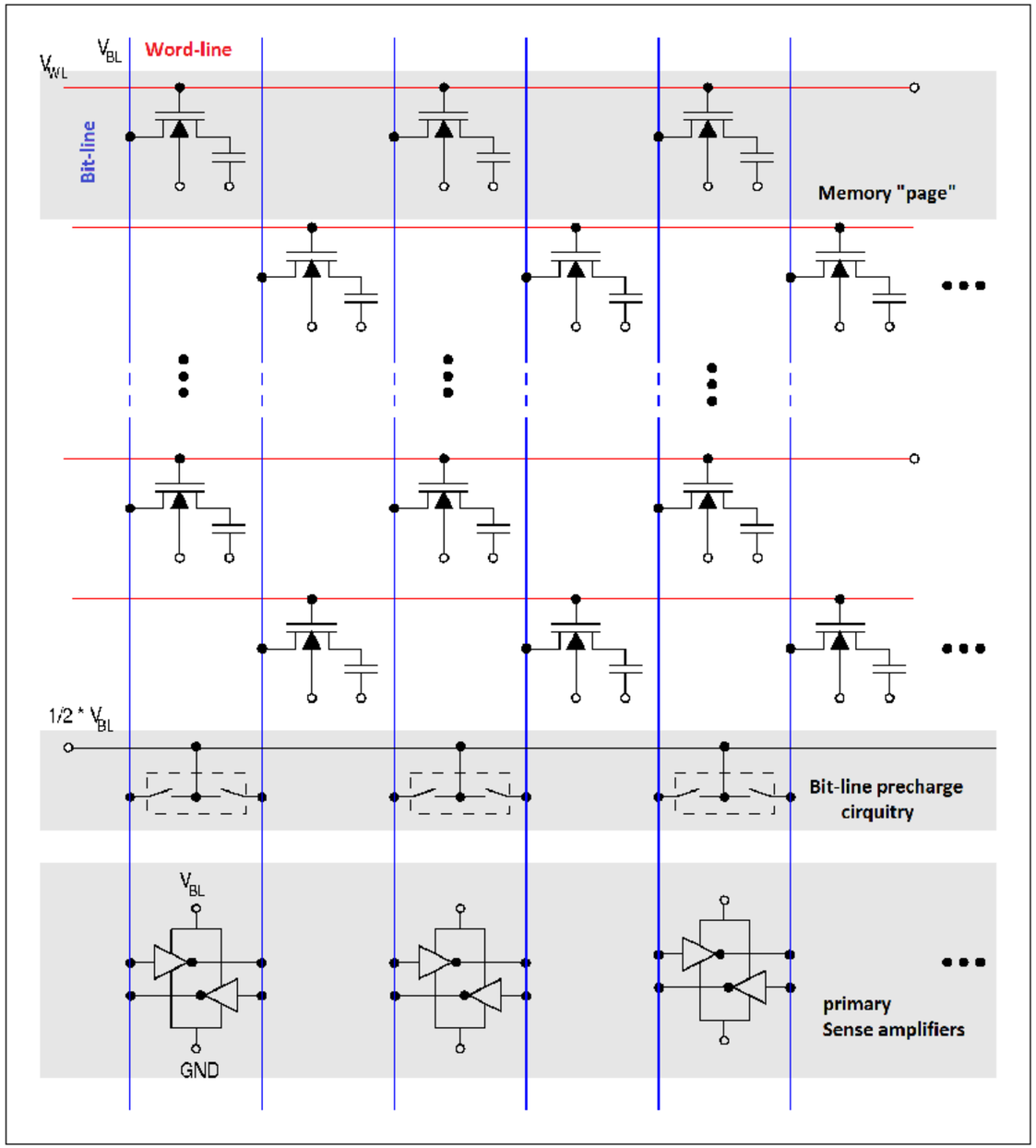
Core Memory

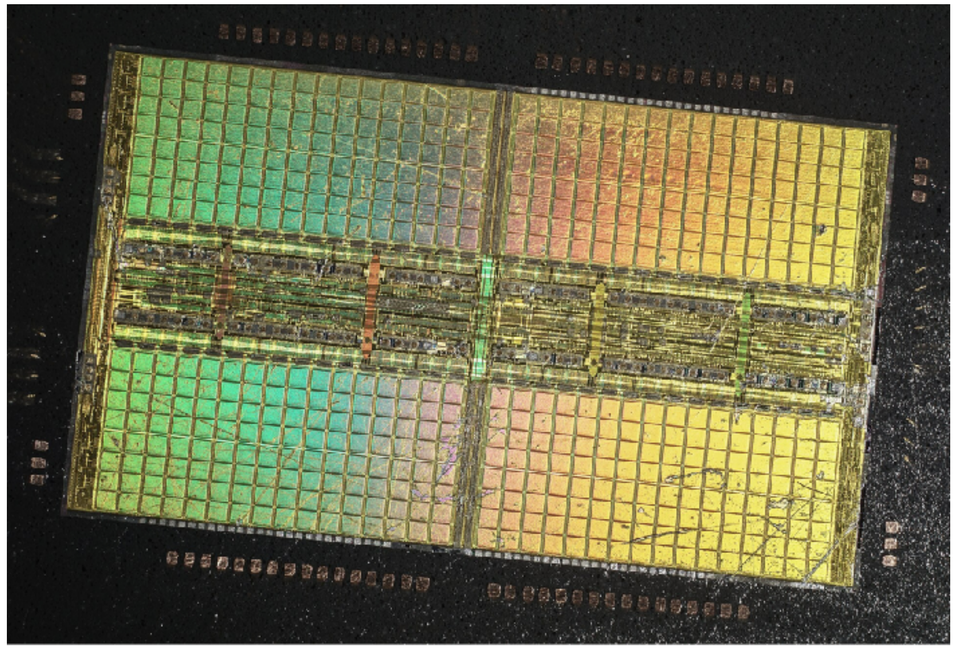


1024 bit



https://en.wikipedia.org/wiki/Magnetic-core_memory



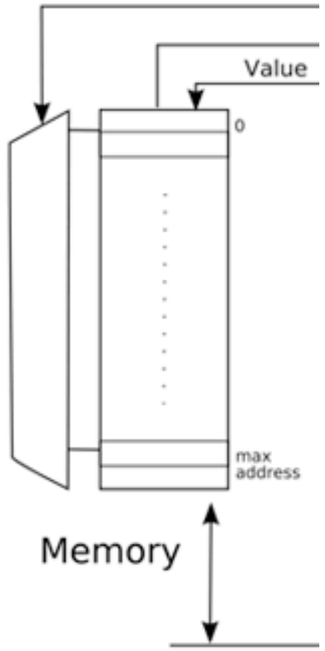


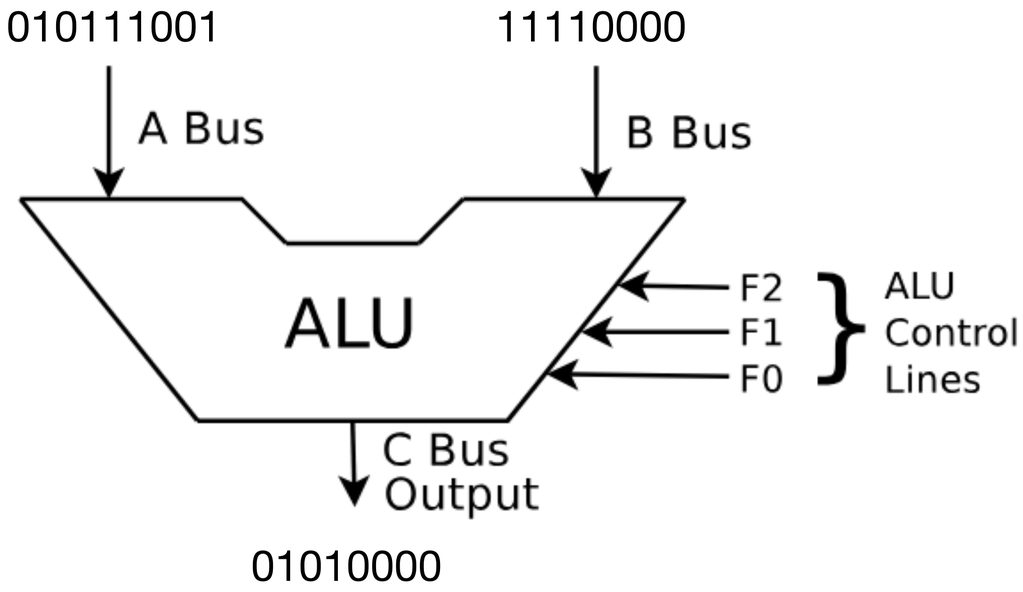
64 Mbit
67 108 864 bit

https://en.wikipedia.org/wiki/Dynamic_random-access_memory

Et mulig programm:

Legg verdien fra minneadresse ... på A-Bus
Legg verdien ... på B-Bus
Send 010 til control lines
Hent verdien fra C-Bus til adresse ....



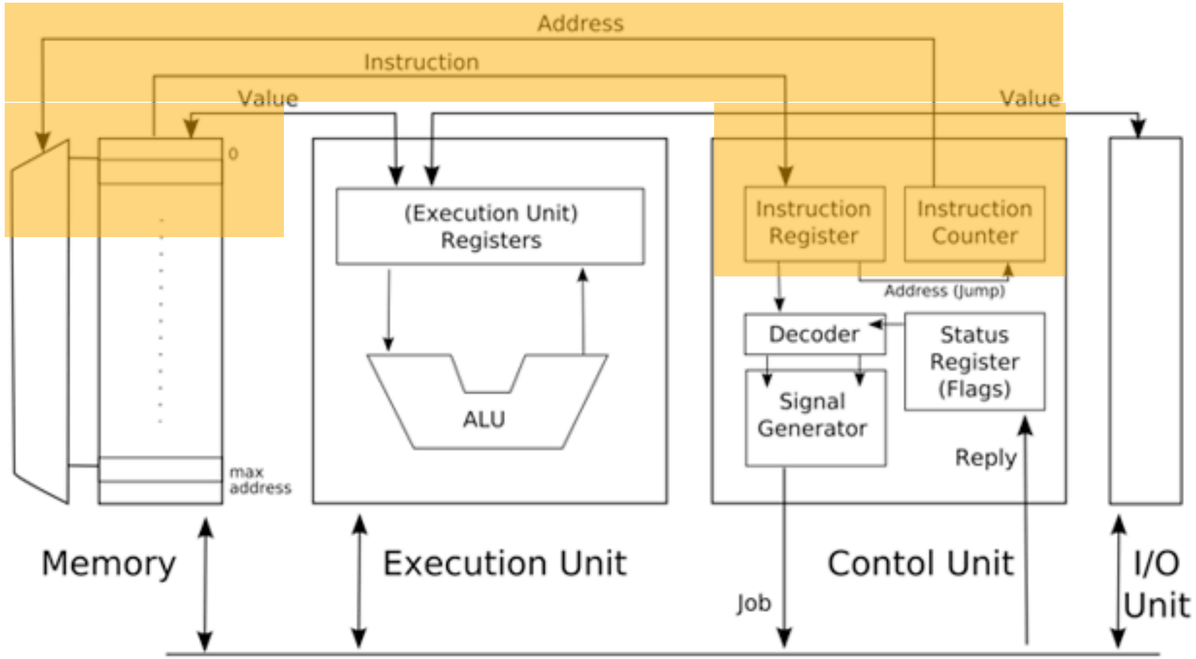


F2	F1	F0	Output
0	0	0	A
0	0	1	B
0	1	0	A+1
0	1	1	B+1
1	0	0	A+B
1	0	1	A-B
1	1	0	A AND B
1	1	1	A OR B

1
1
0

Arbeidssyklus

1. FETCH - last inn instruksen som IC peker på, øk IC
2. DECODE - tolke instruksen, beregn adresser som instr. trenger
3. EXECUTE - kjøre instr., f.eks
   move,
   add,
   compare,
   jump *(endrer IC)*
4. WRITE BACK
   - lagre resultat i registere

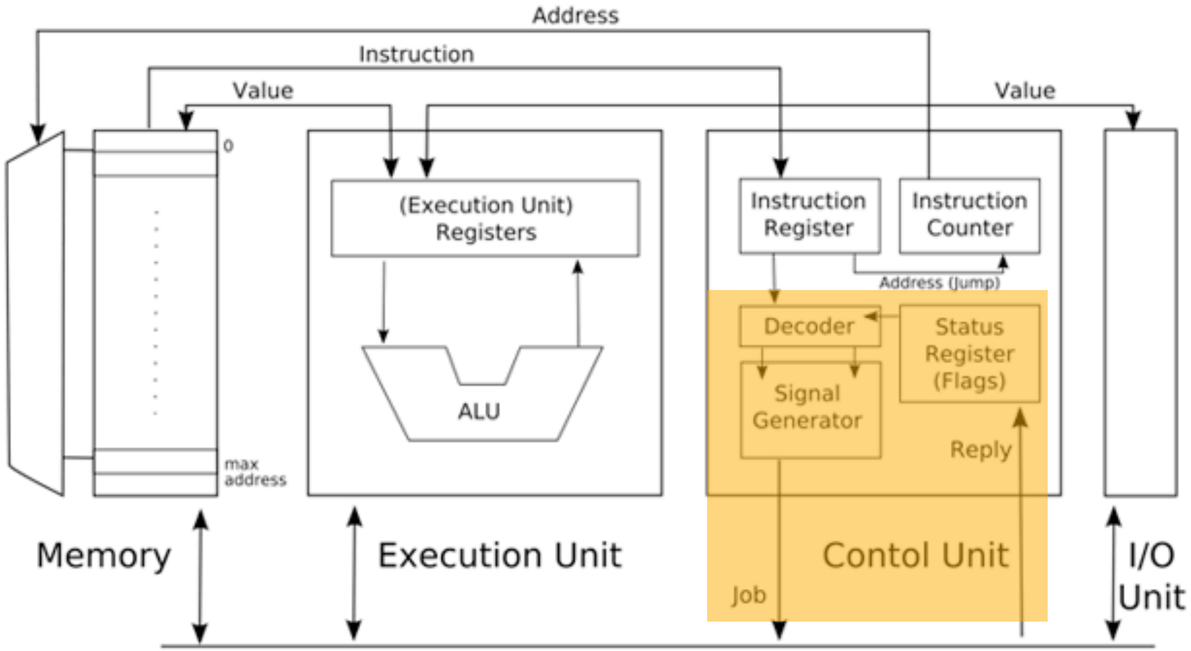


https://en.wikipedia.org/wiki/Instruction_cycle

https://en.wikipedia.org/wiki/Classic_RISC_pipeline

Arbeidssyklus

1. FETCH - last inn instruksen som IC peker på, øk IC
2. DECODE - tolke instruksen, beregn adresser som instr. trenger
3. EXECUTE - kjøre instr., f.eks
   move,
   add,
   compare,
   jump *(endrer IC)*
4. WRITE BACK
   - lagre resultat i registere



https://en.wikipedia.org/wiki/Instruction_cycle

https://en.wikipedia.org/wiki/Classic_RISC_pipeline

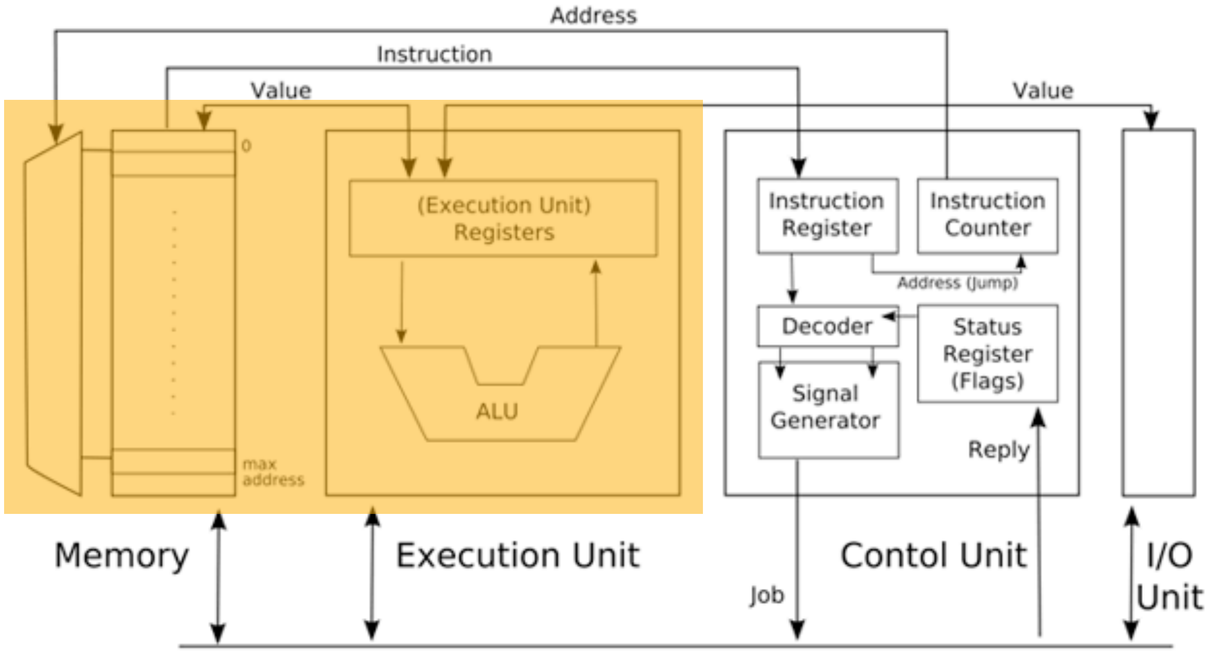
Arbeidssyklus

1. FETCH - last inn instruksen som IC peker på, øk IC

2. DECODE - tolke instruksen, beregn adresser som instr. trenger

3. EXECUTE - kjøre instr., f.eks
   move,
   add,
   compare,
   jump *(endrer IC)*

4. WRITE BACK
   - lagre resultat i registere

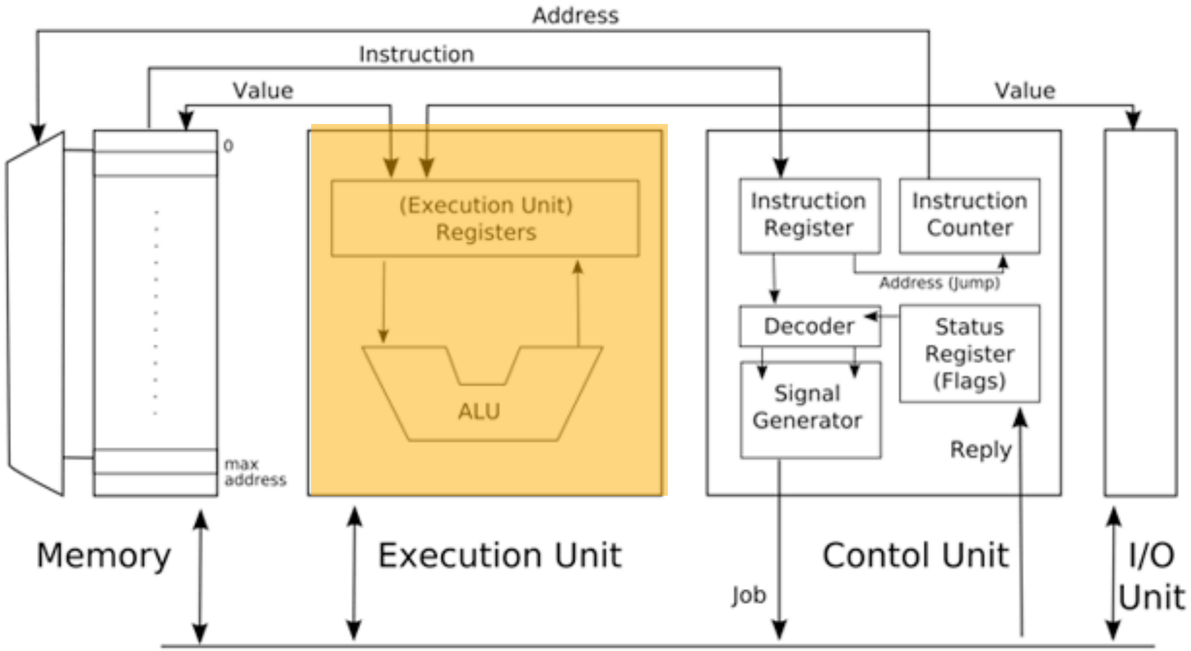


https://en.wikipedia.org/wiki/Instruction_cycle

https://en.wikipedia.org/wiki/Classic_RISC_pipeline

Arbeidssyklus

1. FETCH - last inn instruksen som IC peker på, øk IC
2. DECODE - tolke instruksen, beregn adresser som instr. trenger
3. EXECUTE - kjøre instr., f.eks
   move,
   add,
   compare,
   jump *(endrer IC)*
4. WRITE BACK
   - lagre resultat i registere



https://en.wikipedia.org/wiki/Instruction_cycle

https://en.wikipedia.org/wiki/Classic_RISC_pipeline

Interessert i mer dybde?

- https://www.nand2tetris.org/
- http://www.simplecpudesign.com/

- Minecraft redstone computers
  - https://www.youtube.com/watch?v=ydd6l3iYOZE
  - https://minecraft.fandom.com/wiki/Tutorials/Redstone_computers

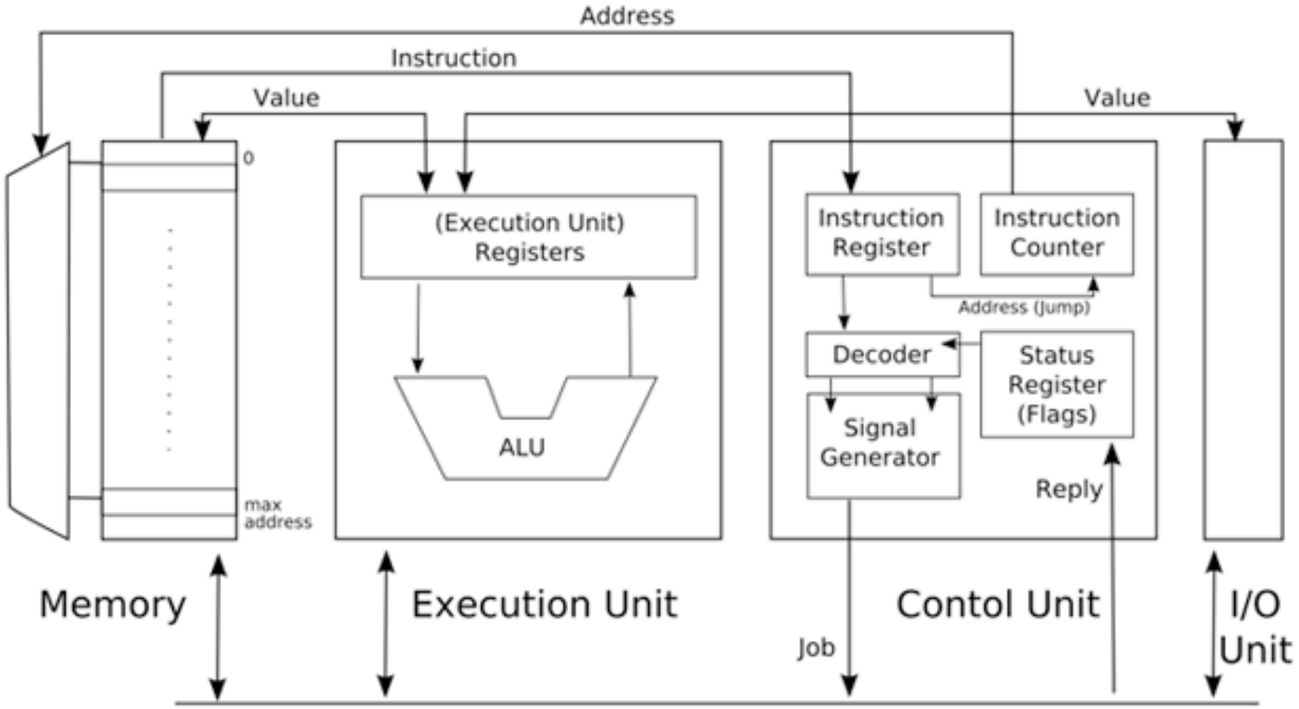
Arbeidssyklus

1. FETCH - last inn instruksen som IC peker på, øk IC

2. DECODE - tolke instruksen, beregn adresser som instr. trenger

3. EXECUTE - kjøre instr., f.eks
   move,
   add,
   compare,
   jump *(endrer IC)*

4. WRITE BACK
   - lagre resultat i registere



https://en.wikipedia.org/wiki/Instruction_cycle

https://en.wikipedia.org/wiki/Classic_RISC_pipeline

Hvor kommer den første instruksen fra?

- Video PDP11 - 3:39

https://web.archive.org/web/20221115033951/http://acad.depauw.edu/dharms_web/pdp11/

- ett programm om gangen
- må lastes inn manuelt og utføre

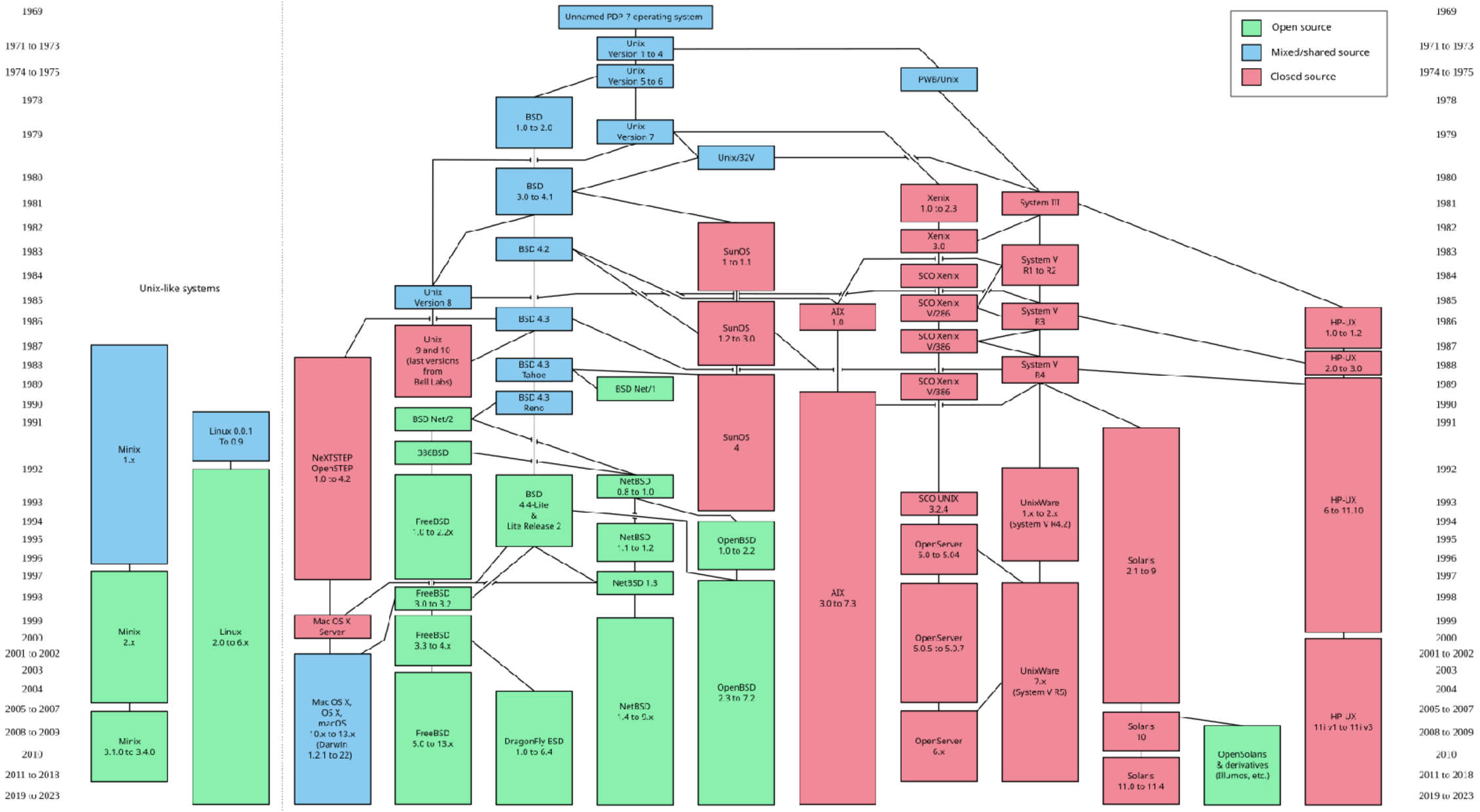
- hva med et "hoved"programm som er der permanent og kan utføre andre programmer? -> Operativsystem

- ett programm om gangen
- må lastes inn manuelt og utføre

- hva med et "hoved"programm som er der permanent og kan utføre andre programmer? -> Operativsystem
- BIOS - Basic I/O system på Hovedkortet, starter ("booter") og laster inn OS

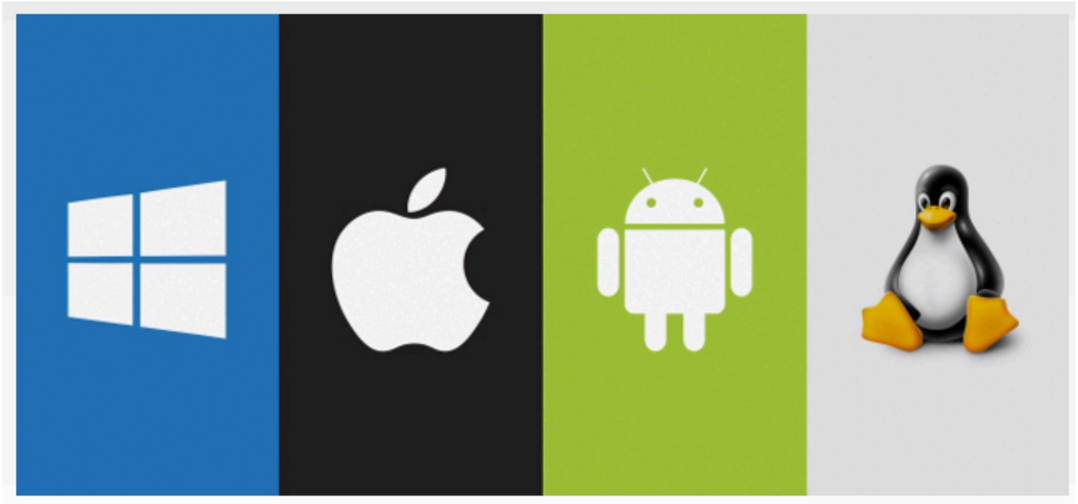
Et eksempel: Unix

https://en.wikipedia.org/wiki/History_of_Unix



Hva er et operativsystem?

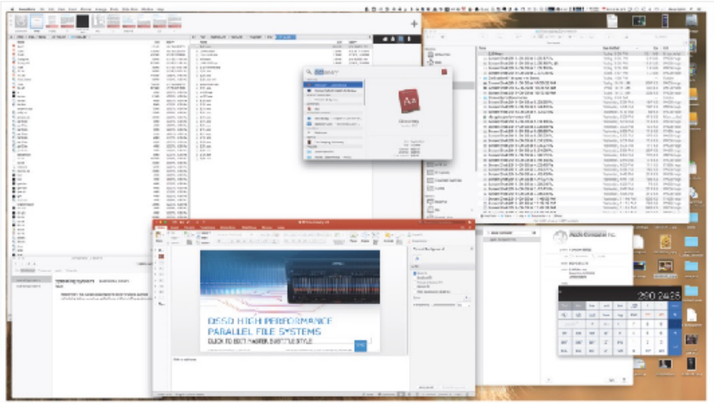
- Hverdagsbruk til ordet:



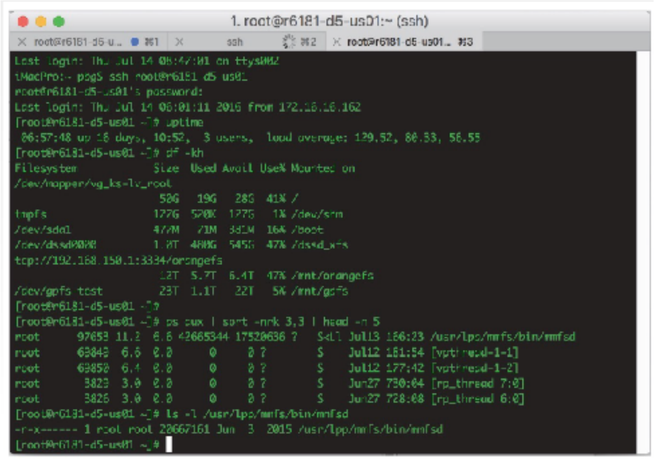
- må skille mellom brukergrensesnitt og "egentlig" OS
- vi snakker om det siste her

Ulike grensesnitt (user interfaces)

- Graphical UI (GUI)



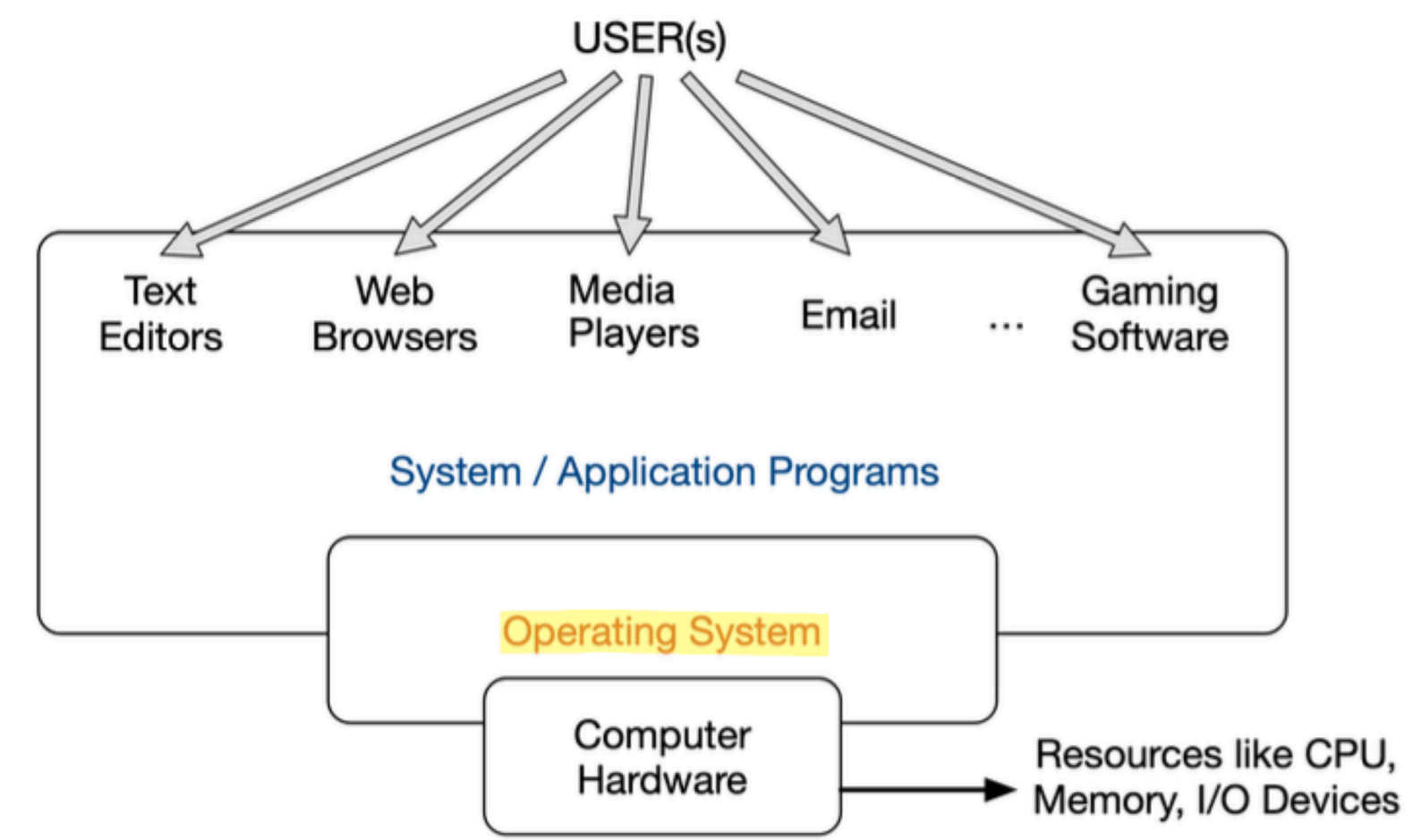
- Kommandolinje (CLI - command line interface)



- ingen egen UI
  (kanskje webserver)



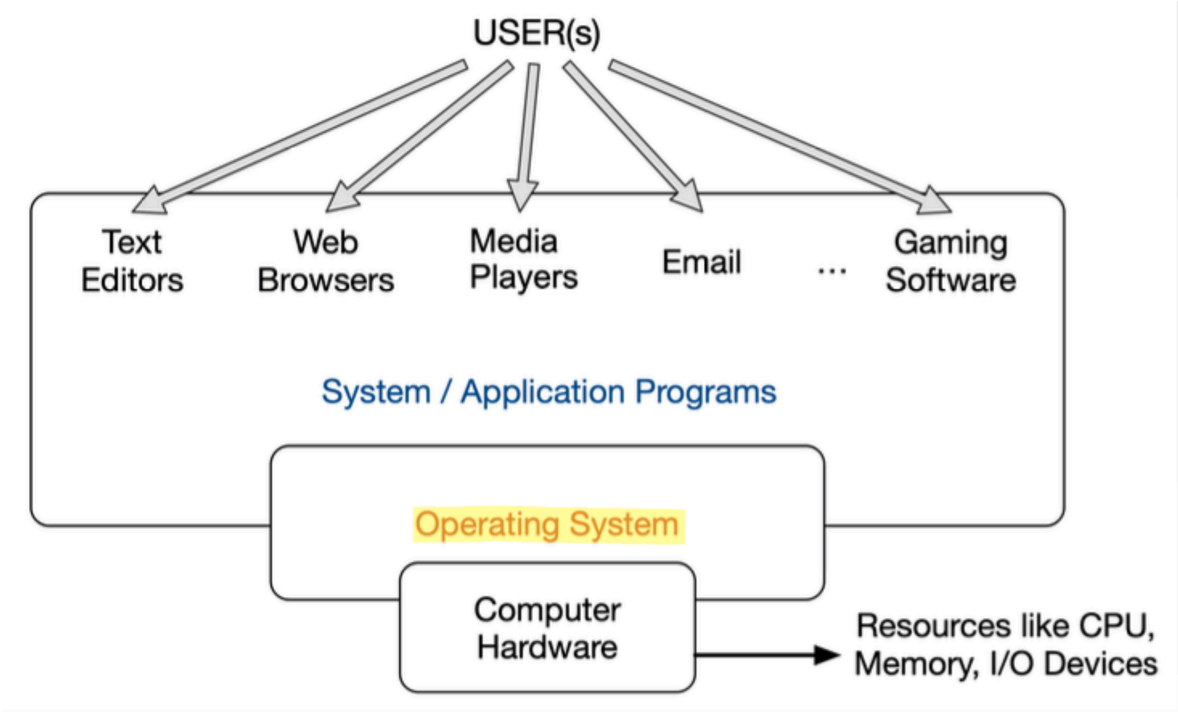
Hva er et operativsystem?



GUI / CLI sitter her

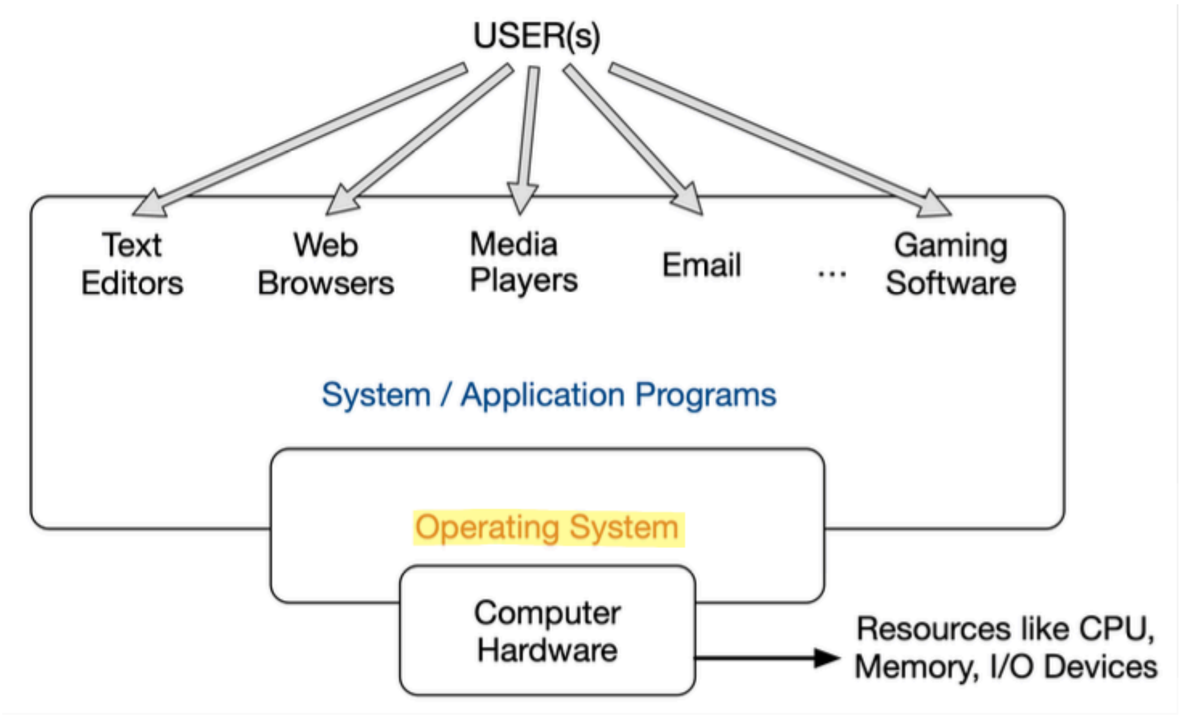
egentlig OS (kernel)

Hvorfor operativsystem?



- **Abstraksjon**: "later som..."
- felles, enhetlig tilgang til ulike hardware-enheter
- eksempel: programmene må ikke bry seg om hva slags lagring er koblet til maskinen (SSD / HDD / SD-card /...)
- eksempel: print / input til terminal-programmet

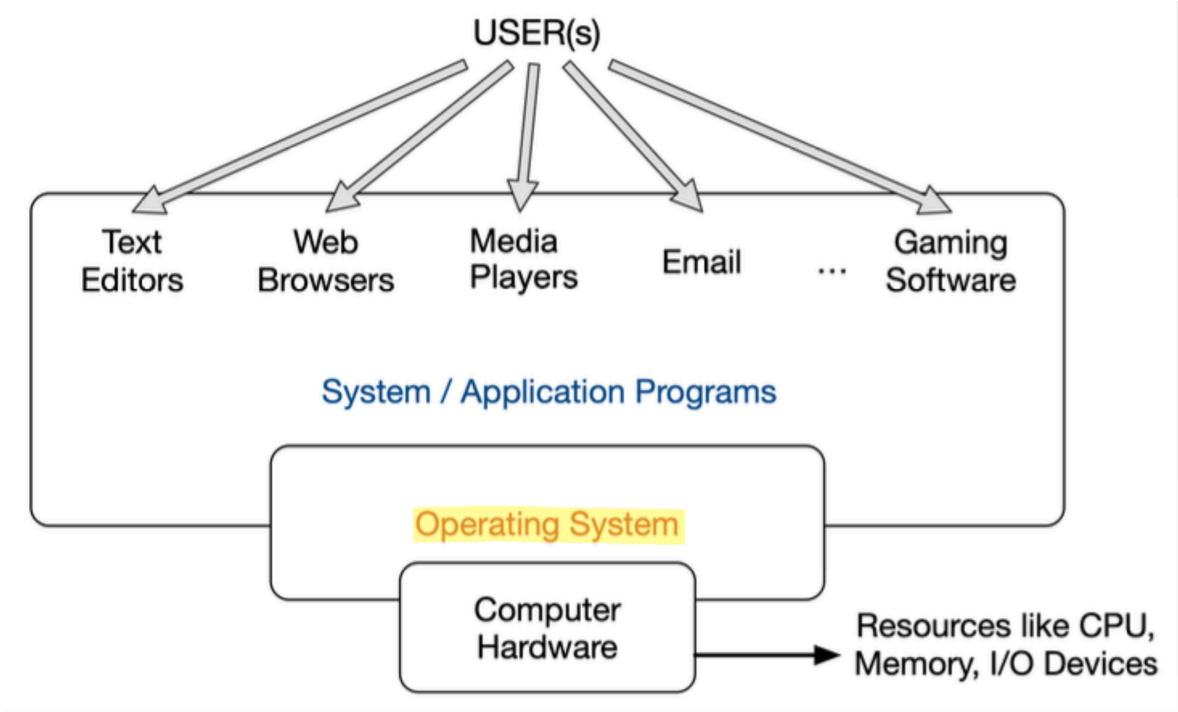
Hvorfor operativsystem?





Terminal

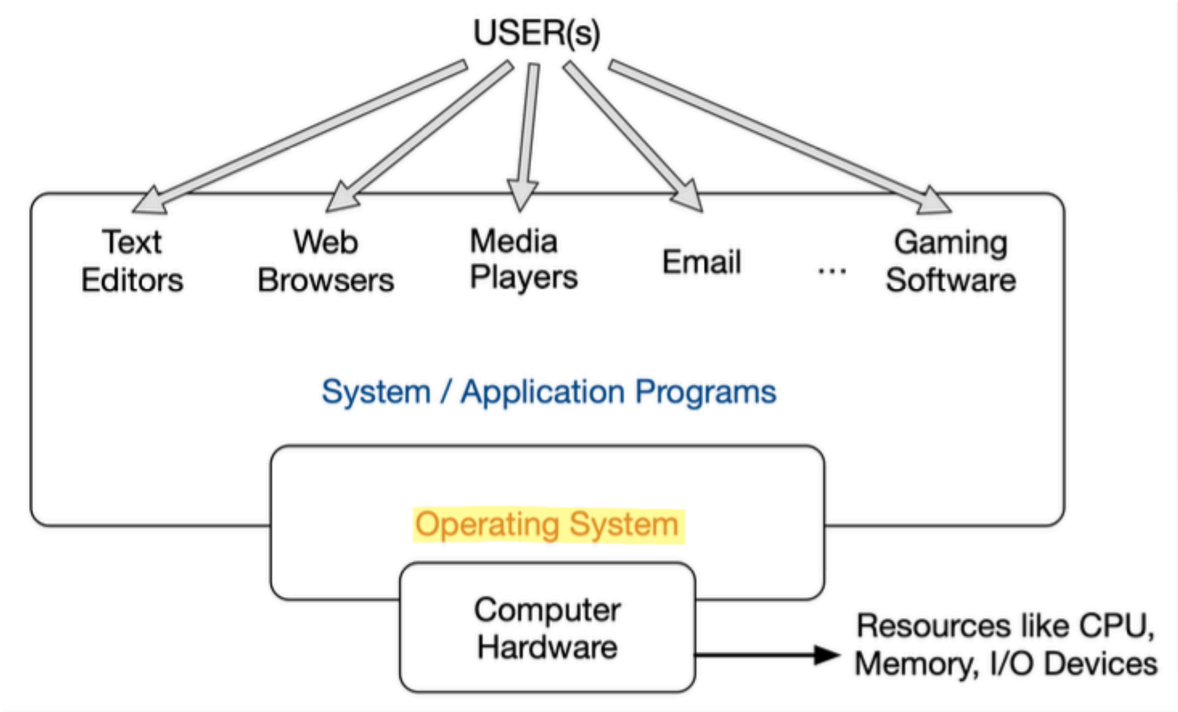
Hvorfor operativsystem?



OS presenterer et virtuell bilde av hardware til programmene:

- CPU
- minne
- lagring

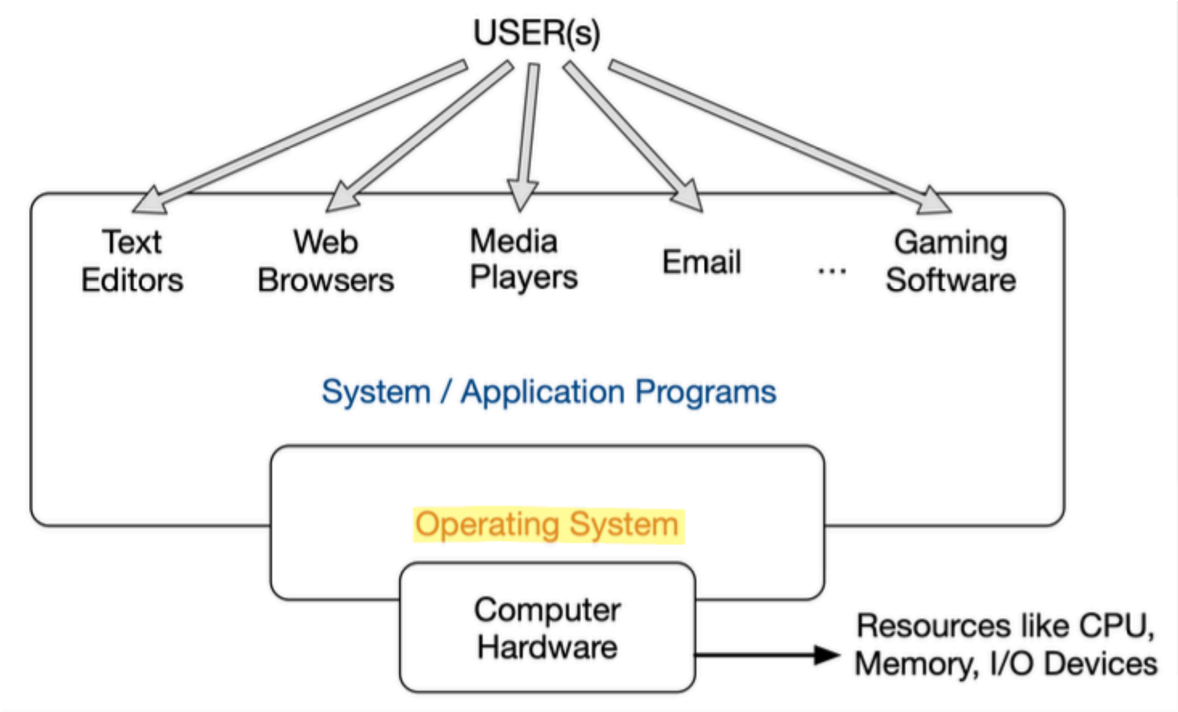
Hvorfor operativsystem?



- **Ressursfordeling**
- kjøre flere programm "samtidig" på CPU
- fordele tilgang til minne
- fordele tilgang til I/O

Må være effisient / lett å bruke av programmene

Hvorfor operativsystem?



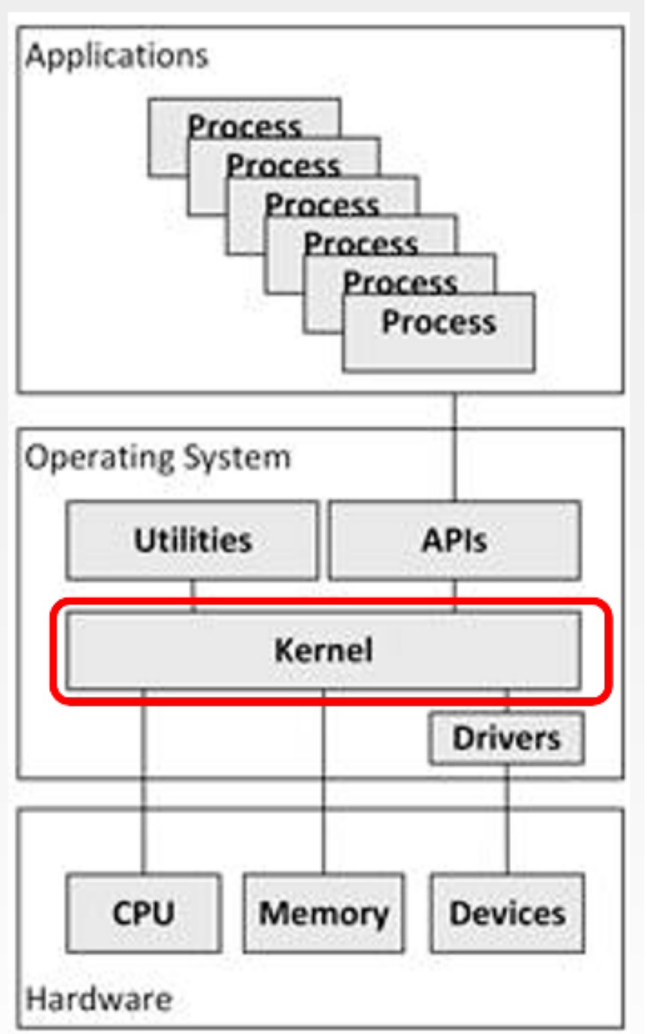
- **Sikkerhet**
- programmer må ikke påvirke andre urelaterte som kjører samtidig

Oppgavefelt til et OS

- **CPU**: hvilken prosess får kjøre? når? hvor lenge?
- **Minne**: hvilken prosess får tilgang? hvor mye?
- **I/O:** hvilken prosess får skrive/lese? når? hvor lenge?
- **Lagring**: forvalte filer (navn, størrelse, tilgang, ...)
- **Kommunikasjon**: mellom prosesser / mellom maskiner

Prøv https://plbrault.itch.io/youre-the-os :-)

Komponenter til et OS

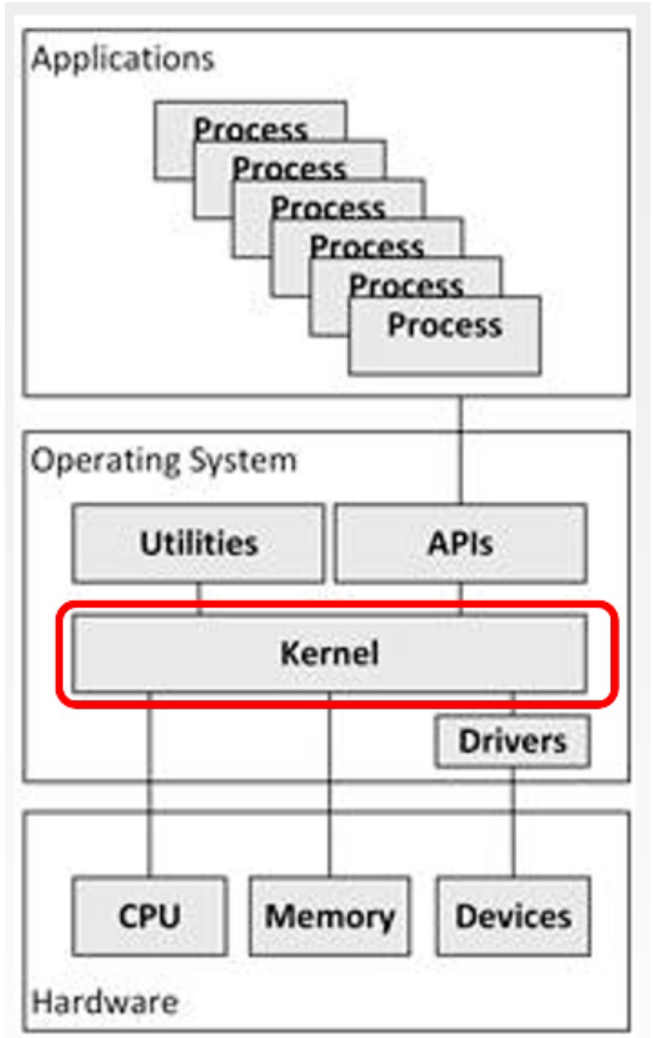


Linux

Linux-distribusjon

Alpine, Mint, Ubuntu, Debian, ...

Komponenter til et OS



Darwin

macOS

Gruppetime 1

- Bygg en PC
- Installer en minimal Alpine Linux der, uten GUI
- Installer en minimal Alpine i VM på egen PC, uten GUI
- Prøv ut kommandolinjen

Kommandolinjen:

Terminal:
- enhet

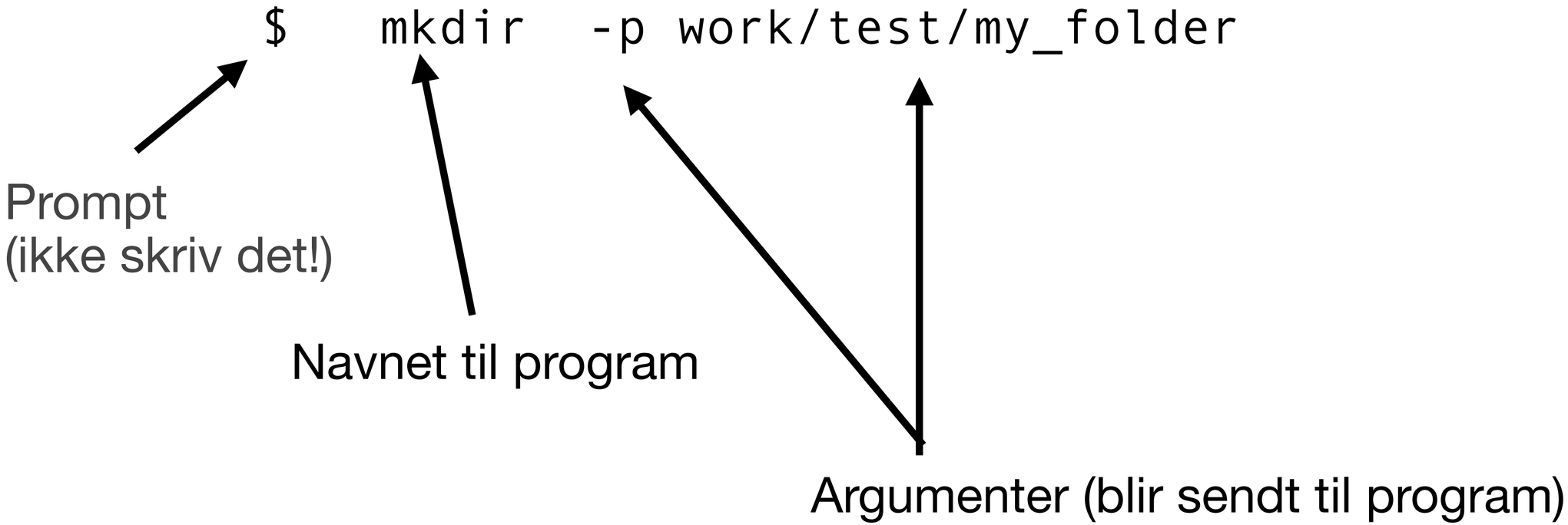
Shell
- programmet som tolker kommandoene



Kommandolinjen / CLI

- Programmene som tolker kommandolinjen kalles for "shell"
- Vi bruker *bash* i eksemplene
  (på windows: bruk vår virtuell maskin eller git-bash eller WSL)
- Direkt og fleksibel interaksjon med alt som OS tilbyr
- ikke limitert til det som GUI-apper tillater
- enkel å automatisere
  bytte navn på 300 filer / legge inn dato i hver jpeg filnavn / flytte alle filer større enn 5GB i en mappe -> vanskelig i GUI

Kommandolinjen / CLI

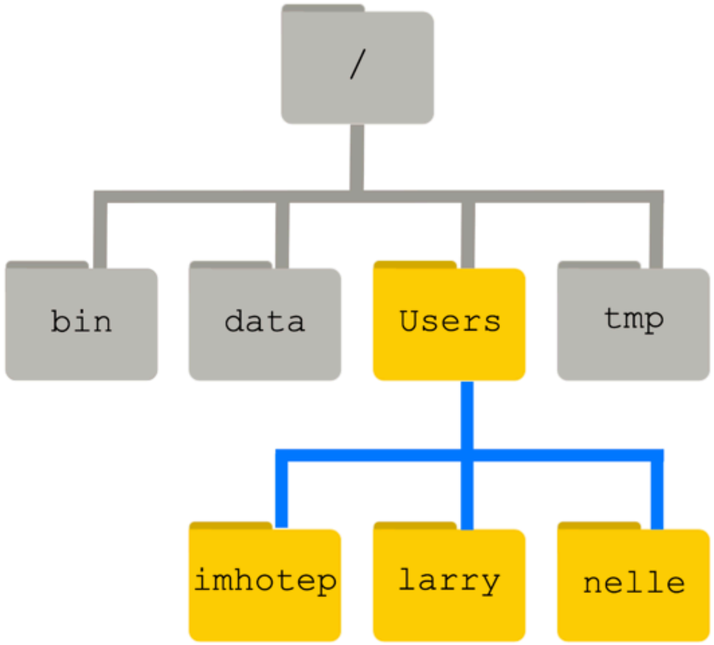


Ulike delene skilles med mellomrom. Mellomrom i filnavn er vanskelig i shell, bruk _

Bash introduksjon - les ila. neste ukene

https://swcarpentry.github.io/shell-novice/01-intro.html

til og med avsnitt "4. Pipes and filters..."



Alternativt intro til bash: https://ictp.grelli.org/bash/