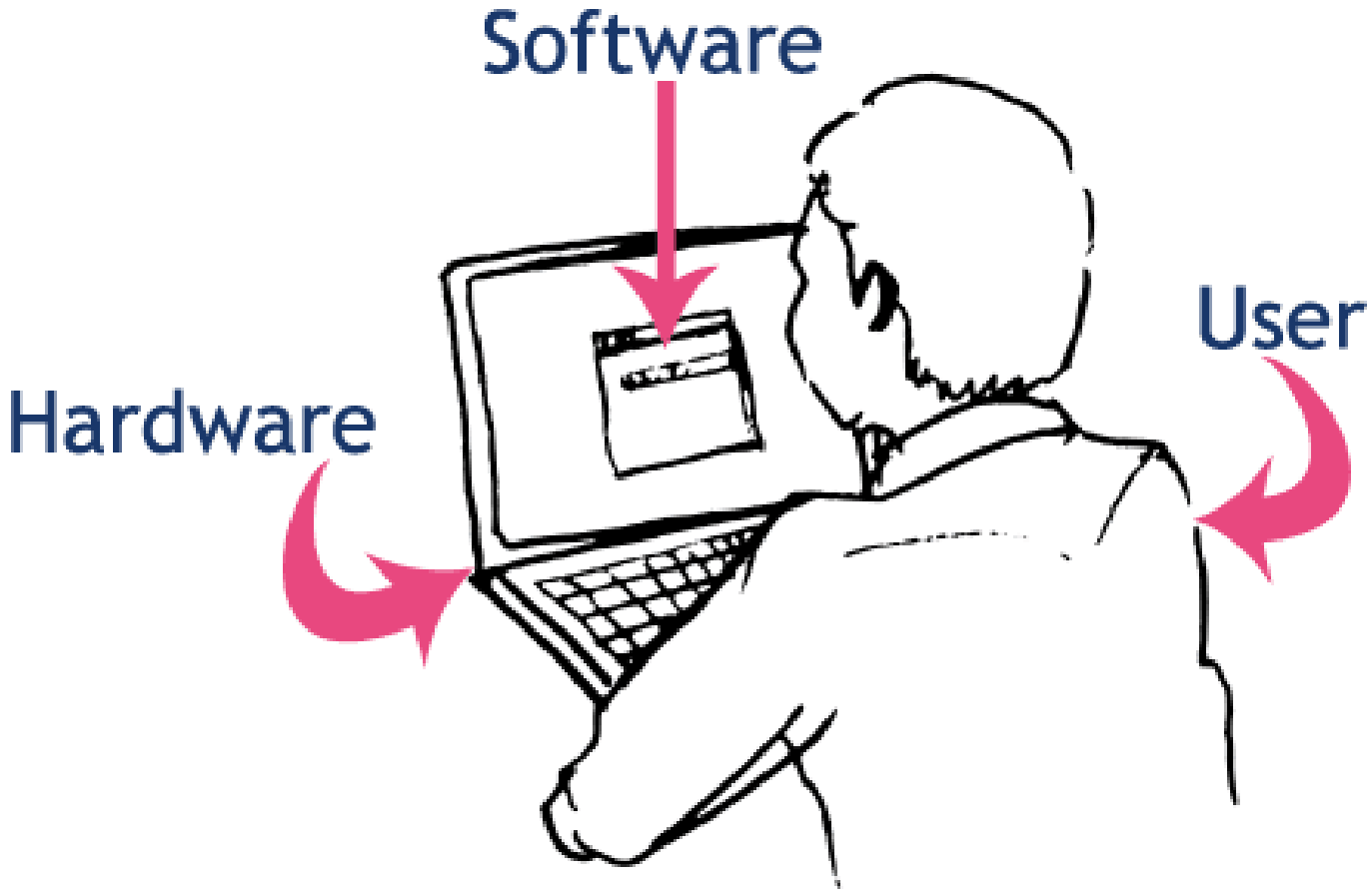
Datasystemer

VG1 IM

Introduksjon til datasystemer

- **Hva er et datasystem?**
- Består av **maskinvare** (CPU, minne, lagring, enheter)
- **Programvare** (operativsystem, apper, drivere)
- **Bruker** som samhandler med systemet
- Samarbeid = gjør komplekse oppgaver mulig



Hva består en PC av?

- Maskinvare og Programvare
- Prosessor - M
- Minne - M
- Lagring - M
- Operativsystem – P
- Apper - P

Aktivitet

- **Hva bruker dere PC/mobil til i løpet av en dag?**
- Kommunikasjon
- Underholdning
- Skole
- Praktiske ting
- Word

Mobil

- Sende meldinger (Messenger, Snapchat, WhatsApp)
- Sosiale medier (Instagram, TikTok, Facebook)
- Ta bilder / video
- Spille spill
- Bruke bank-app / Vipps
- Se på YouTube eller Netflix

PC

- Gjøre skolearbeid (Word, PowerPoint, Teams, nettleser)
- Skrive oppgaver / søke informasjon
- Spille dataspill
- Laste ned / installere programmer
- Redigere bilder eller video
- Surfe på nettet

- 📡 **Kommunikasjon**
- Meldinger (Messenger, WhatsApp, Snapchat)
- Samtaler / videosamtaler (Teams, Zoom, Facetime)
- Sosiale medier (Instagram, TikTok, Facebook)

- 🎮 **Underholdning**
- YouTube, Netflix, Spotify
- Gaming (PC- og mobilspill)
- TikTok-scrolling
- Musikk og podkast

- 📚 **Skole / arbeid**
- Teams (innleveringer, møter)
- Word, PowerPoint, Excel
- Research på nett (Google, NDLA)
- Programmering / praktiske oppgaver i fag

- 💳 **Praktiske ting**
- Vipps / nettbank
- Bestille mat / billetter / klær på nett
- Helsesjekk (Helsenorge, apotek-apper)
- Kart og kollektivtransport (Google Maps, RuterBillett)

Aktivitet

- Eksempel kobling
  - Mus
  - Tastatur
  - USB

Operativsystem

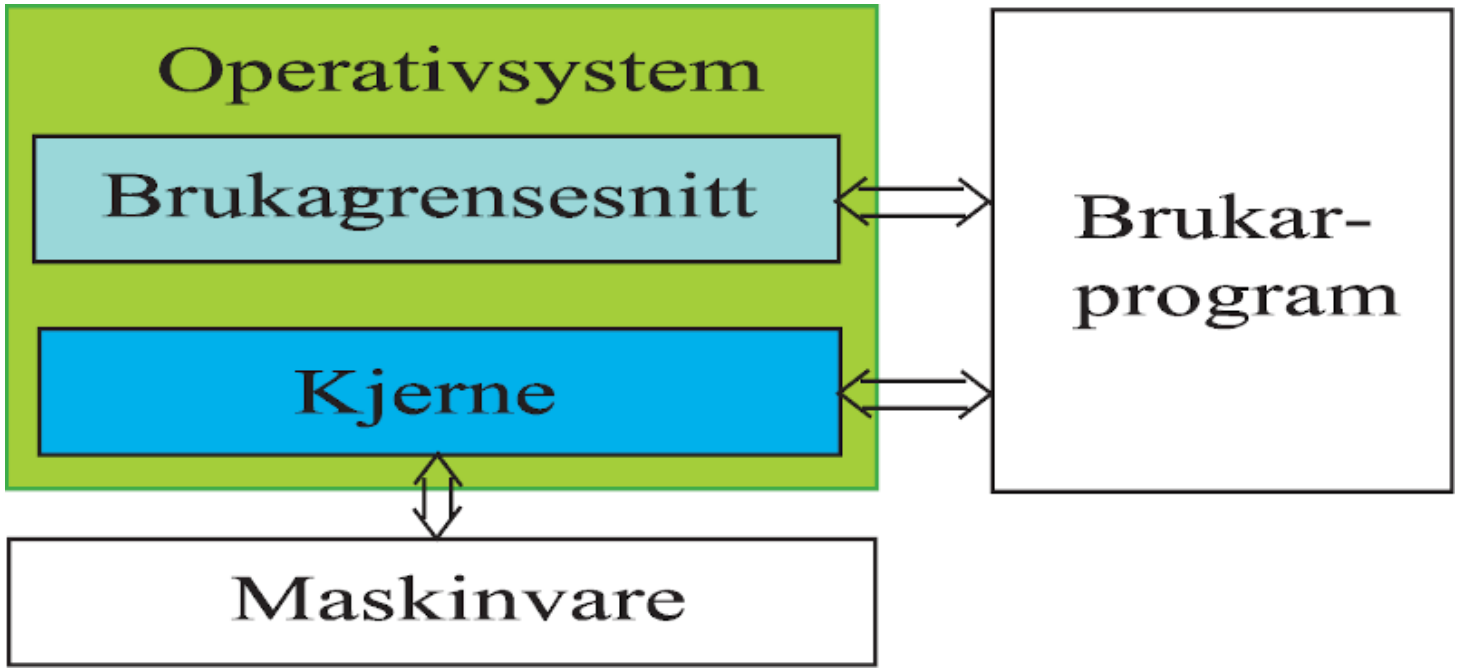
- Hjernen?

Diskusjon

- Hva tror dere “operativsystem” betyr?
- Word

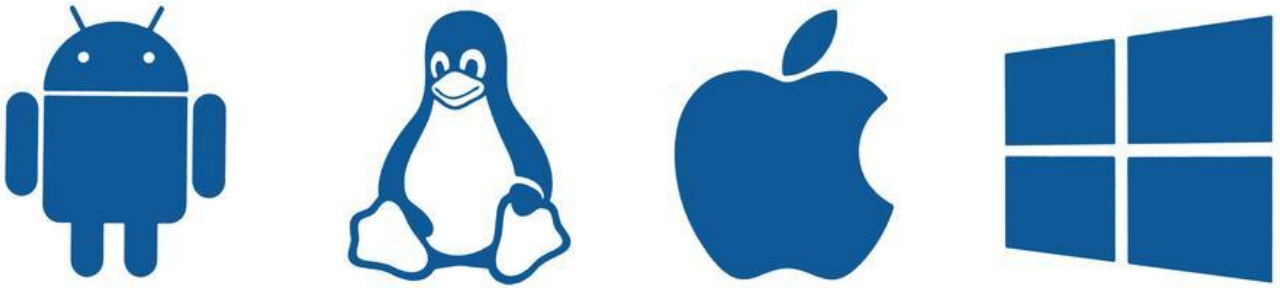
Hva er et operativsystem?

- OS = programvare som styrer og fordeler ressursene i en datamaskin.
- Ligger mellom maskinvaren og programmene.
- Gir oss brukergrensesnitt (tekst + grafikk).



Eksempler

- Windows
- MacOS
- Linux
- Android
- iOS



Første datamaskiner (ingen OS)

- Tidlige maskiner → enorme, kun 1 program av gangen.
- Programmer ble skrevet direkte inn i maskinen.
- Stor ulempe: programmer måtte tilpasses hver maskin.

Behovet for OS

- Batch processing → kjøre flere programmer etter hverandre.
- Multitasking → flere programmer samtidig.
- Multi-user → flere brukere koblet på samme maskin.

Dagens operativsystemer

- PC: Windows, macOS, ChromeOS.
- Mobil/nettbrett: Android, iOS.
- Servere: Linux, Windows Server.
- «Usynlige OS»: minibank, biler, togskjermer.
- **Oppgave**: Finn noe dere har hjemme som dere tror har et OS.

Brukergrensesnitt: CLI

- Tekstbasert (kommandoer).
- Gir mer detaljstyring, brukes fortsatt på servere.
- Lite krevende for maskinen.

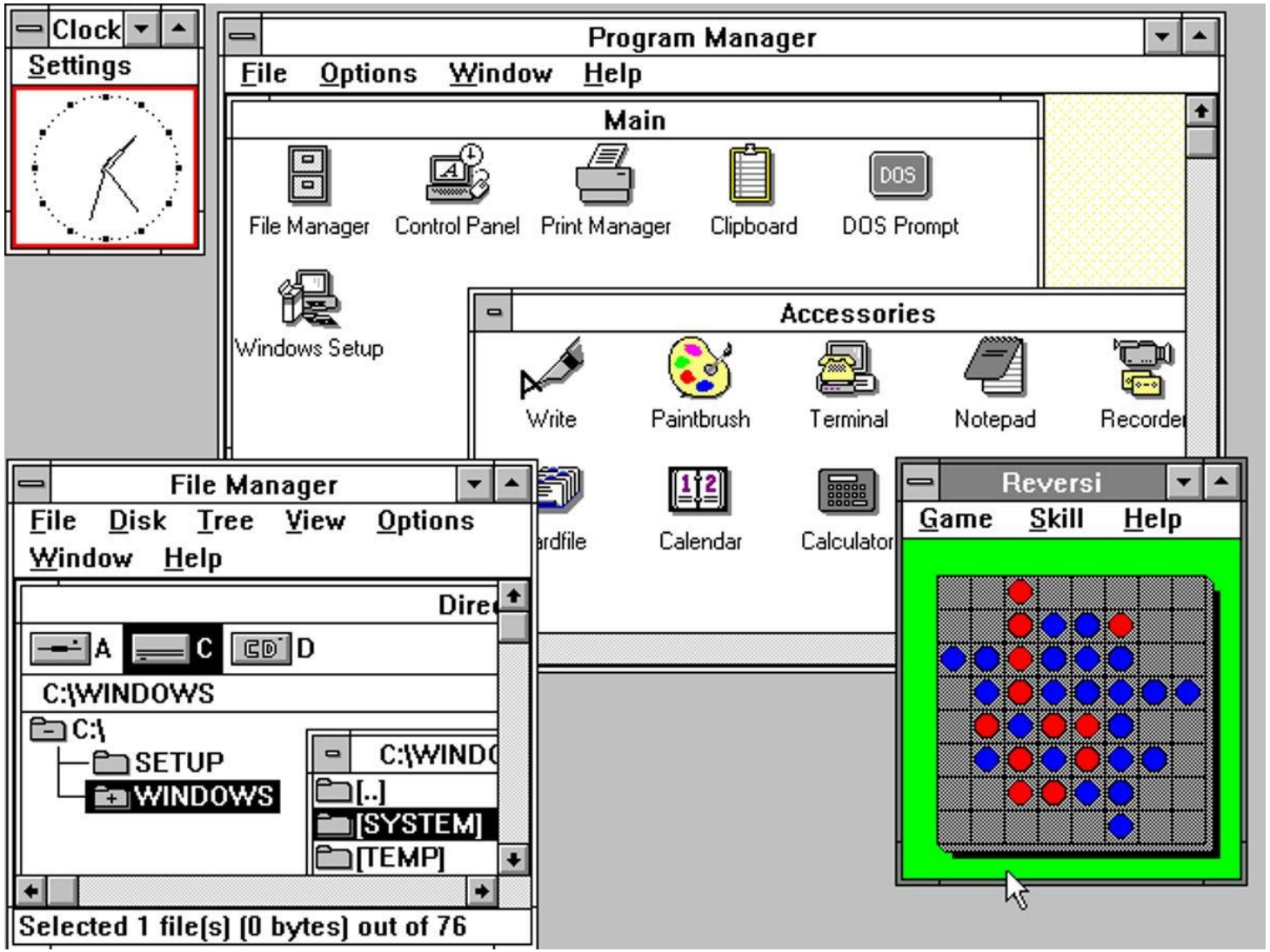
- **Eksempel**: ipconfig i Windows.

- **Miniaktivitet**:
  - Åpne terminal og skriv cmd på PC
  - test en enkel kommando

Brukergrensesnitt: GUI

- Grafisk brukergrensesnitt → ikoner, vinduer, mus/touch.
- Vanlig for PC/mobil.
- Historie: først 1980-tallet (Apple, Xerox, Windows 1.0).
  📌 Bilde: Windows/macOS skrivebord.

- **Spørsmål**: Hva er fordelen med GUI sammenlignet med CLI?



Brukergrensesnitt: Web GUI

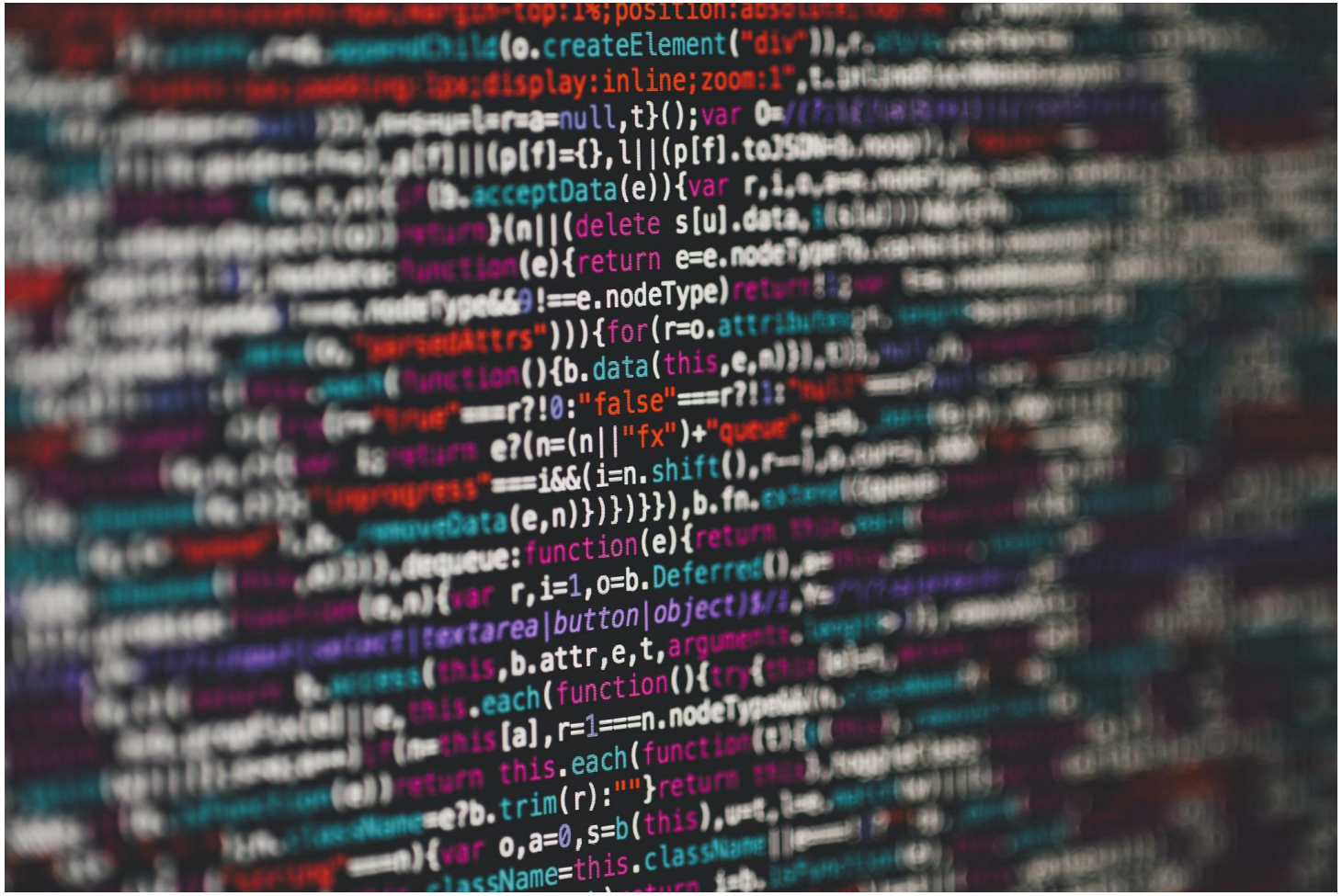
- Mange tjenester styres i nettleser (router, skyløsninger).
- Enkelt å bruke, men gir ikke alltid full tilgang.
- Ofte kombinert med CLI for avanserte brukere.
- **Eksempel**: Prøv hjemme: Logge inn på «192.168.1.1».
- **Diskusjon**: Hvorfor tror dere man ikke gir full tilgang via web?

Oppgave

- Hva er OS, og hvorfor trenger vi det?
- Eksempler på OS i dagliglivet.
- Forskjell mellom CLI, GUI, Web GUI.
- Sammenlign mobiler
  - Hvilken OS bruker dere?
  - Hva er forskjellen i hverdagen?
- Bruk google og finn svar.

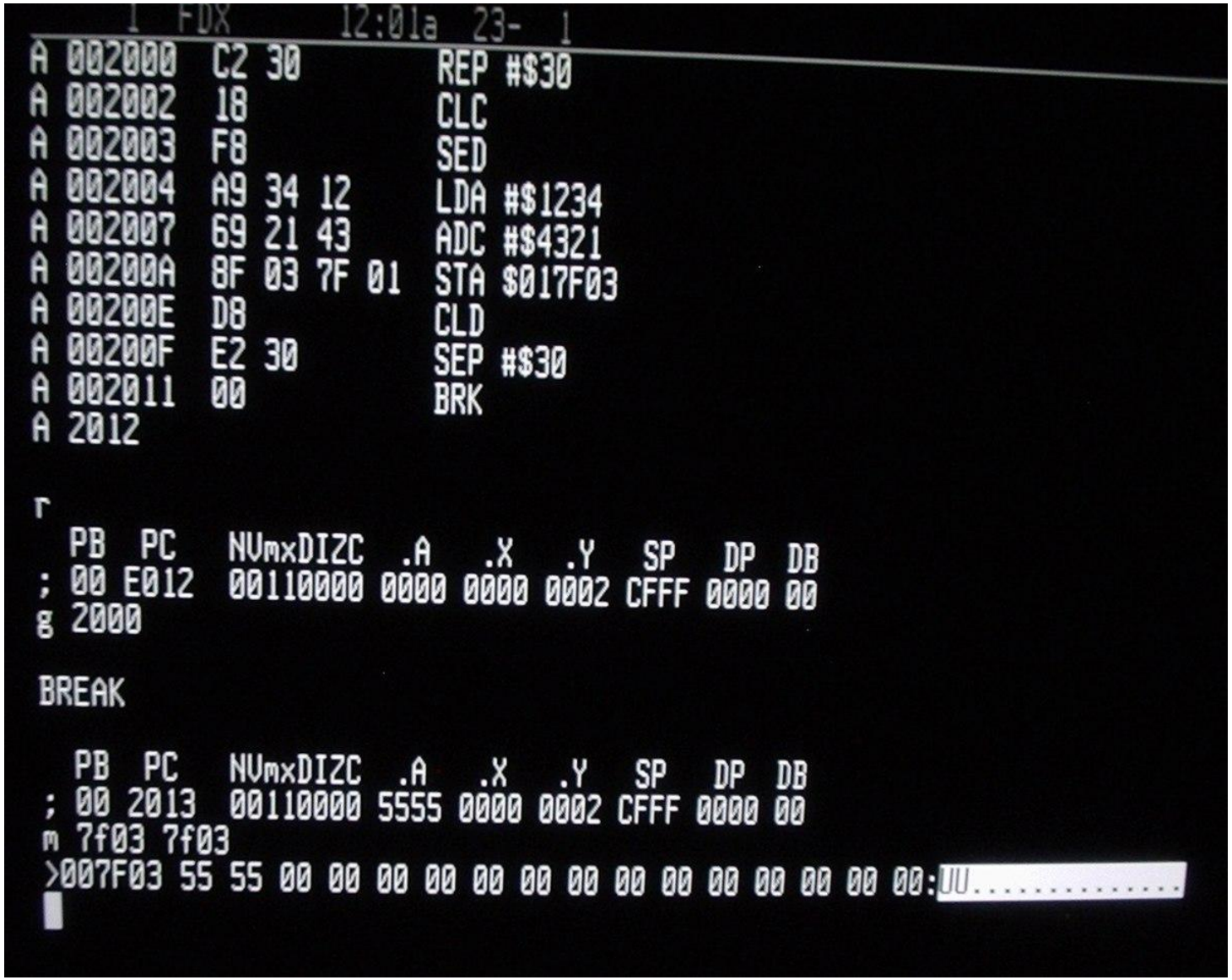
Drivere + maskinkode

- Driver = “tolk” mellom maskinvare og OS.



Fra kildekode til maskinkode

- Programmer skrives i språk vi mennesker forstår (Python, Java, C# ...).
- Datamaskiner forstår bare 0 og 1 → maskinkode.
- Derfor må koden «oversettes».
- Diskusjon: Hvilke programmeringsspråk har dere hørt om / brukt?

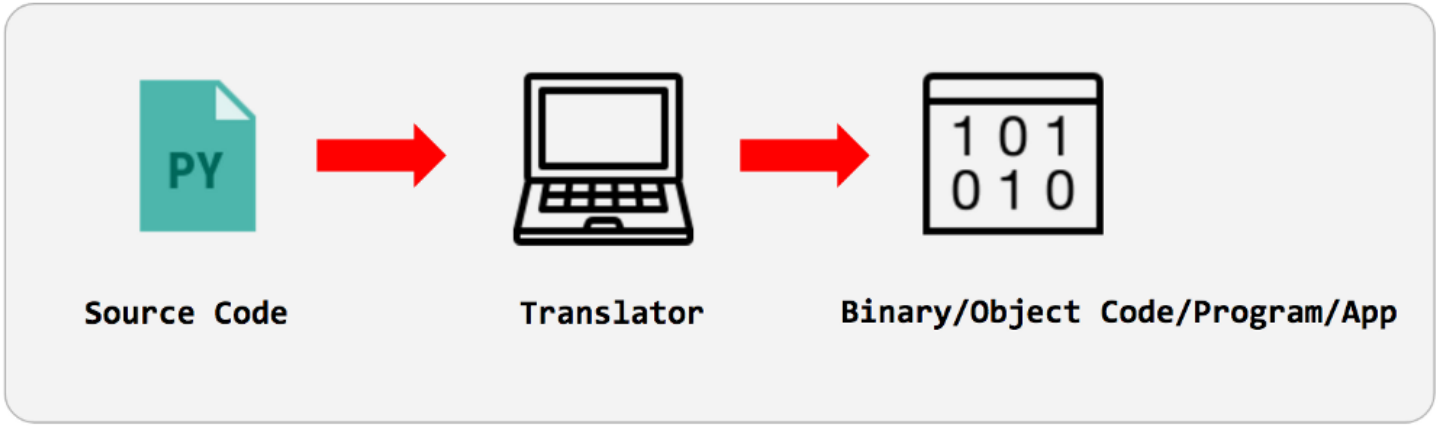


Hva er maskinkode?

- Består av binære instruksjoner (0 og 1).
- Eksempler: regne, flytte data, sammenligne tall.
- Upraktisk å skrive for mennesker → men superrask for maskinen.

Aktivitet – bruk google

- Hva er lavnivåspråk? Eksempler
- Hva er høynivåspråk? Eksempler
- Hva betyr kompilering?
- Word



Assembly (lavnivåspråk)

- Nærmest maskinkoden, men med «ord» istedenfor 0/1.
- Eksempel: mov ecx, ebx (flytter data mellom registre).
- Effektivt, men avansert.

- Hvorfor tror dere Assembly fortsatt brukes i små ting som mikrokontrollere?

```
section .data
    hello_msg db "Hello, World!", 0xA  ; Message followed by newline (0xA is newli
    hello_len equ $ - hello_msg        ; Length of the message

section .text
    global _start                      ; Entry point for the program

_start:
    ; Write syscall (sys_write, syscall number 4)
    mov eax, 4                         ; System call number for sys_write
    mov ebx, 1                         ; File descriptor 1 - standard output (stdo
    mov ecx, hello_msg                 ; Address of the message
    mov edx, hello_len                 ; Length of the message
    int 0x80                           ; Interrupt to make the syscall

    ; Exit syscall (sys_exit, syscall number 1)
    mov eax, 1                         ; System call number for sys_exit
    xor ebx, ebx                       ; Return code 0 (success)
    int 0x80                           ; Interrupt to make the syscall
```

Høynivåspråk

- Språk som ligner mer på naturlig språk → Python, JavaScript, C#.
- Én linje kode kan tilsvare mange linjer maskinkode.
- Gjør programmering enklere for mennesker.

- Eksempel: "Hello World.

- Aktivitet: Skriv i VS Code.

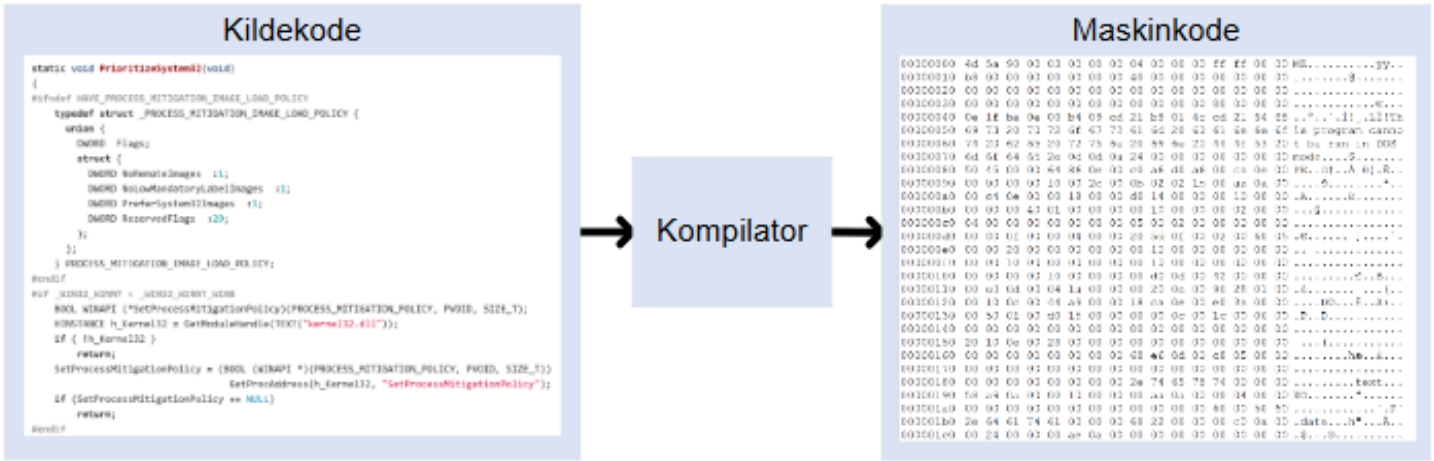
```
<script>
  console.log("Hello World");
  alert("Hei, verden!");
</script>
```

Kompilering

- Kompilator = oversetter kildekode → maskinkode.
- Samme kode kan kjøres på ulike prosessorer (x86, ARM).
- Viktig for at programmer skal virke på både PC og mobil.

- Grafikk: Kildekode → Kompilator → Maskinkode.

- Spørsmål: Hvorfor er det nyttig at koden kan kompileres til flere typer maskiner?



Vanlige feil

- **Syntaksfeil** = grammatikkfeil i koden (mangler semikolon, feil bokstav).
- **Logisk feil** = koden kjører, men gjør feil (f.eks. ganger i stedet for å legge til).
- Eksempel i Python:
  - print("Hei"
  - feil: mangler parentes

Oppsummering

- Kode → kompilering → maskinkode.
- Maskinkode = 0/1, Assembly = lavnivå, Python/Java = høynivå.
- Viktig å forstå prosessen, selv om vi sjelden jobber med 0/1.

Hva er en standard?

- En standard = regler/definisjoner for at ting skal passe sammen.
- Eksempel:
  - A4-papir
  - USB-lader

Hvorfor standarder i IT?

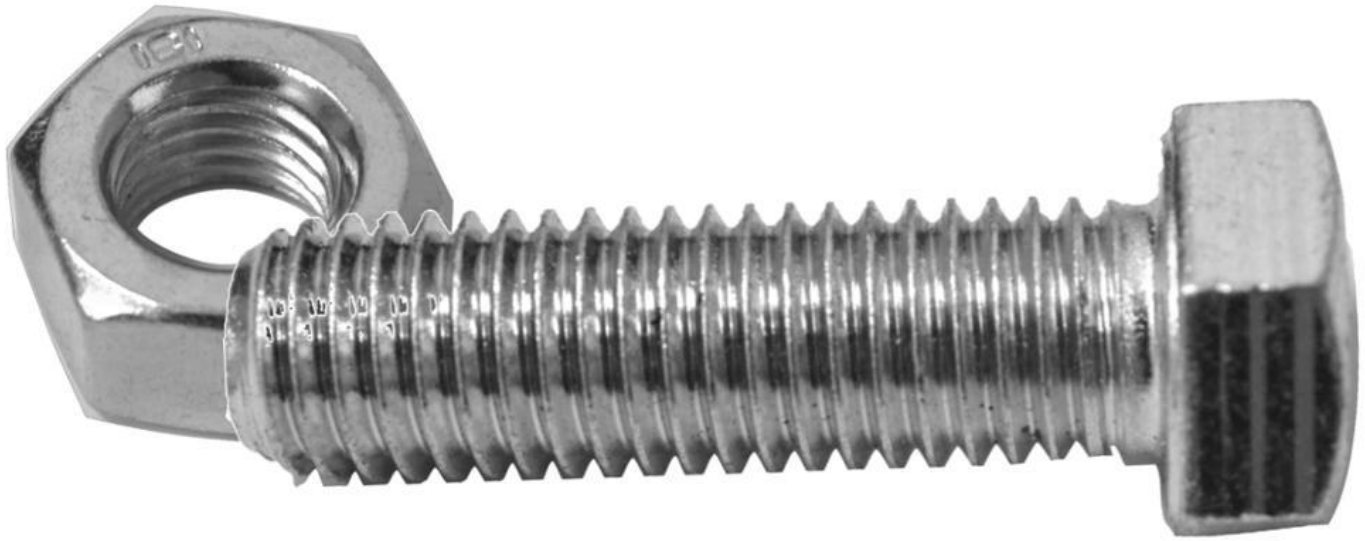
- Tidlige datamaskiner brukte egne (proprietære) løsninger.
- Problemer: utstyr passet ikke sammen.
- I dag: fellesstandarder (eks. USB-C).

Åpen standard

- Fri tilgjengelig dokumentasjon.
- Alle kan bruke gratis.
- Styres ofte av organisasjoner (f.eks. W3C → HTML, CSS, SVG).

Proprietær (lukket) standard

- Eies av en bedrift/organisasjon.
- Dokumentasjon ikke offentlig.
- Eksempel: NTFS (Microsoft).

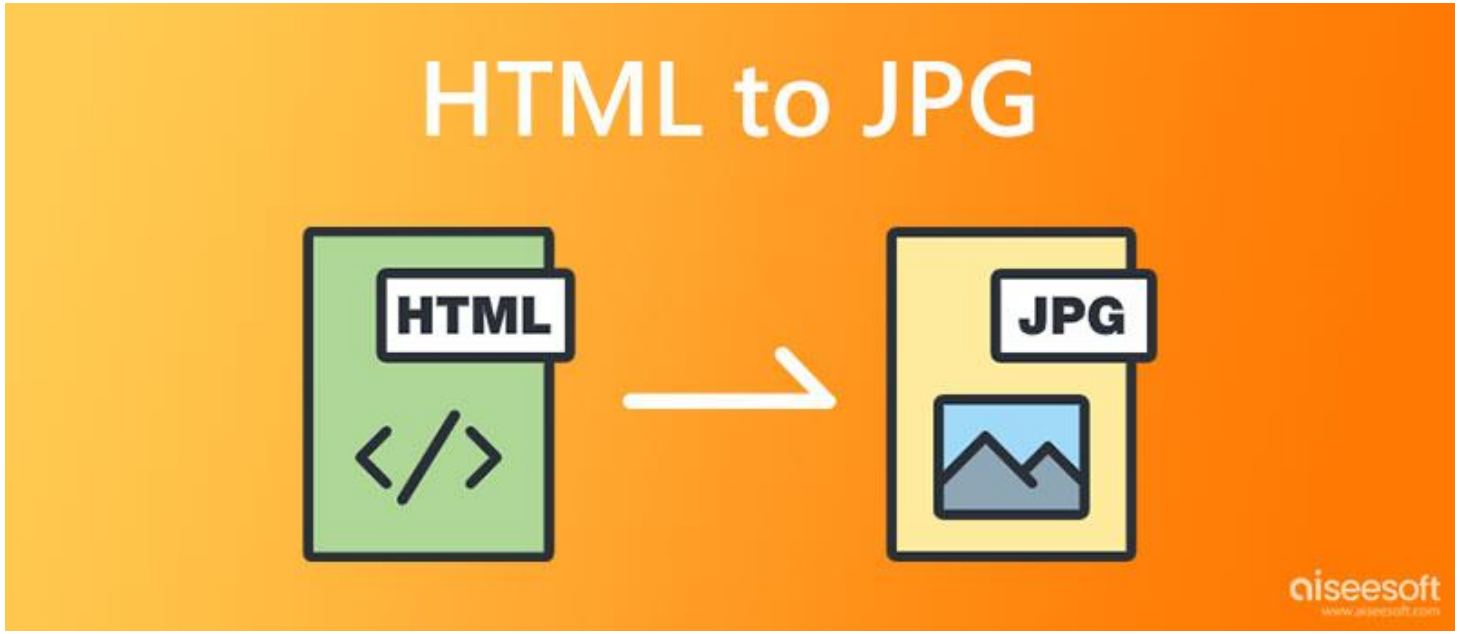


Åpne standarder med restriksjoner

- Mest åpne, men noen begrensninger.
- Eksempel: USB – gratis å bruke, men logo krever sertifisering (dyrt).

Eksempler fra hverdagen

- Åpen standard: HTML (alle nettsider), JPEG (bilder).
- Lukket standard: WhatsApp-protokollen, Apples Lightning-kontakt.



Aktivitet

- Lag liste over
  - Åpne standarder dere bruker daglig
  - Lukkede standarder dere bruker daglig
- Word

Oppsummering

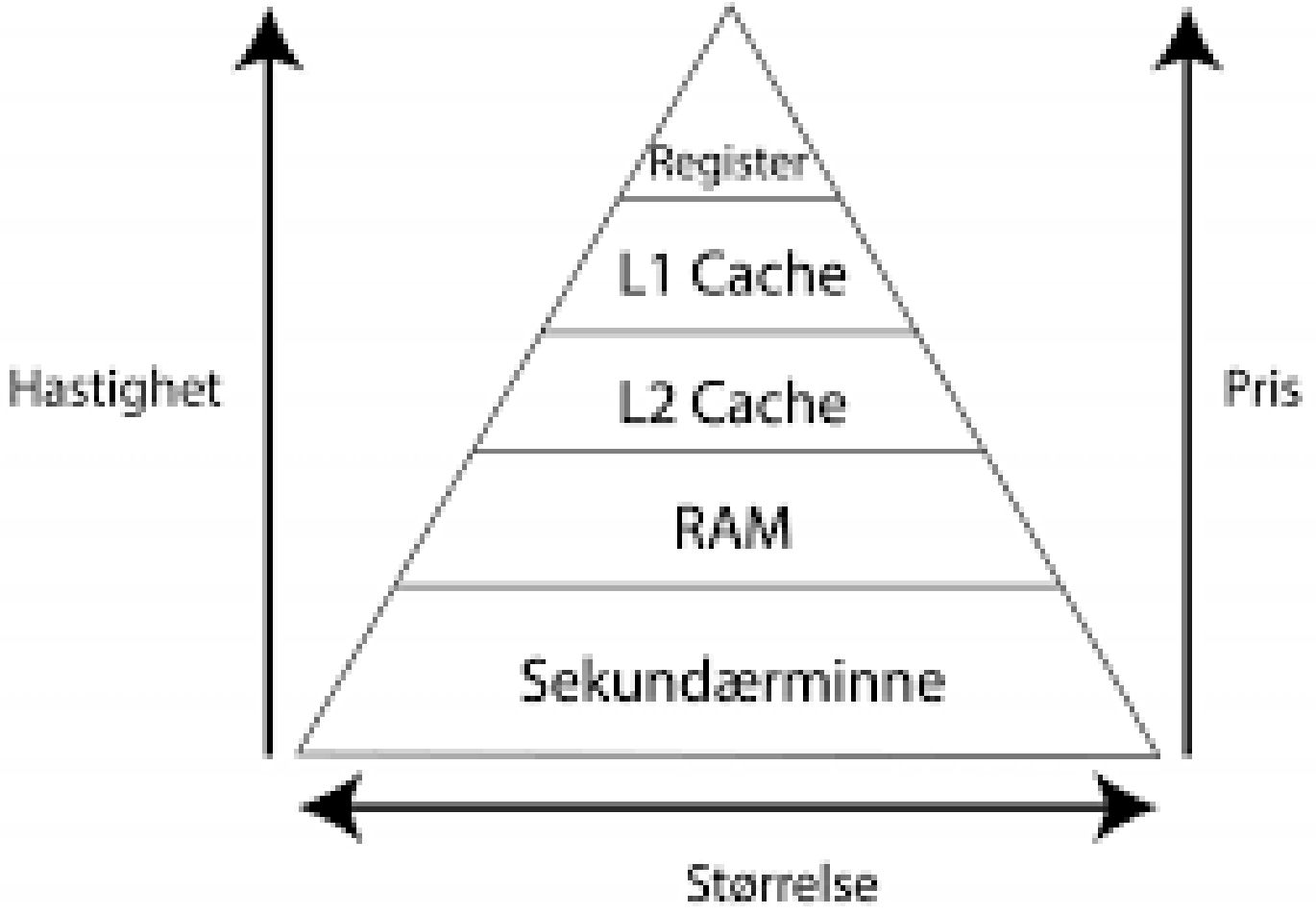
- Standarder = nødvendig for at systemer skal fungere sammen.
- Åpen standard → tilgjengelig for alle.
- Lukket standard → kontrollert av én aktør.

Innlevering på Teams

- Lag en kort tekst eller en punktliste (½–1 side) der du sammenligner **åpne og gratis formater** med **lukkede og kommersielle formater**.
  - Hva er fordelene med åpne formater?
  - Hva er ulempene?
  - Hva er fordelene med lukkede/kommersielle formater?
  - Hva er ulempene?
  - Nevn gjerne eksempler (for eksempel PDF, DOCX, MP3, OGG, PNG, JPG).

Hva er minnehierarki?

- Forskjellige typer minne: raske/dyre vs. trege/billige.
- PC kombinerer flere typer for å balansere **hastighet, pris, lagring**.

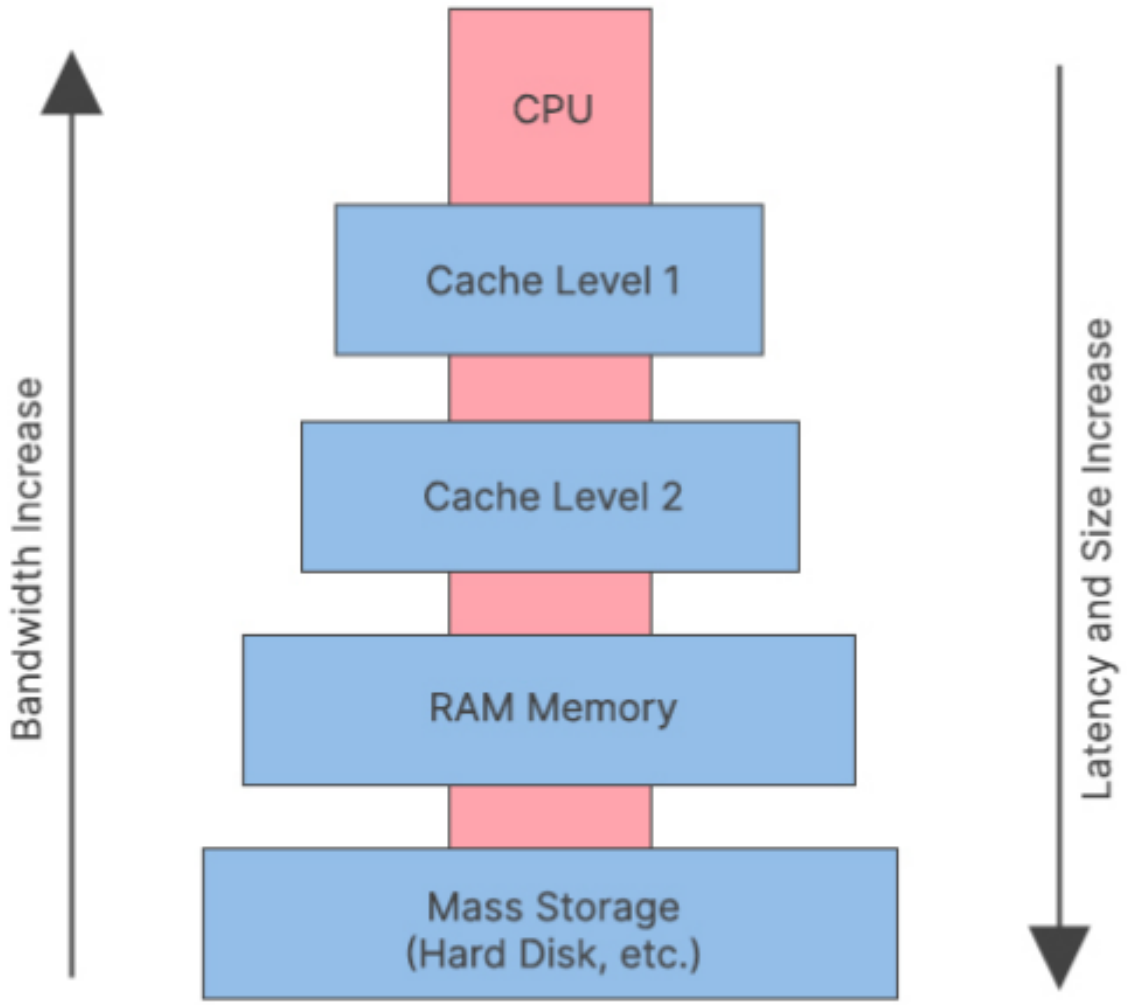


Minnehierarki

- Forskjellige typer minne: raske/dyre vs. trege/billige.
- PC kombinerer flere typer for å balansere **hastighet, pris, lagring**.
- Jo raskere → jo dyrere.
- Flyktig (RAM, cache) = mister data uten strøm.
- Ikke-flyktig (SSD, HDD, teip) = beholder data.
  Eksempel: Word-dokument i RAM → hvis strømmen kuttes, forsvinner det.

Pyramiden

- Øverst: CPU-register → superraskt, superlite.
- Cache (L1, L2, L3) → raskt, små mengder.
- RAM → større, men tregere.
- SSD/HDD → mye plass, men tregere.
- Teip/arkiv → billigst, tregest.
  💡 Aktivitet: Tegn pyramiden på tavla → elevene plasserer minnetypene i rekkefølge.

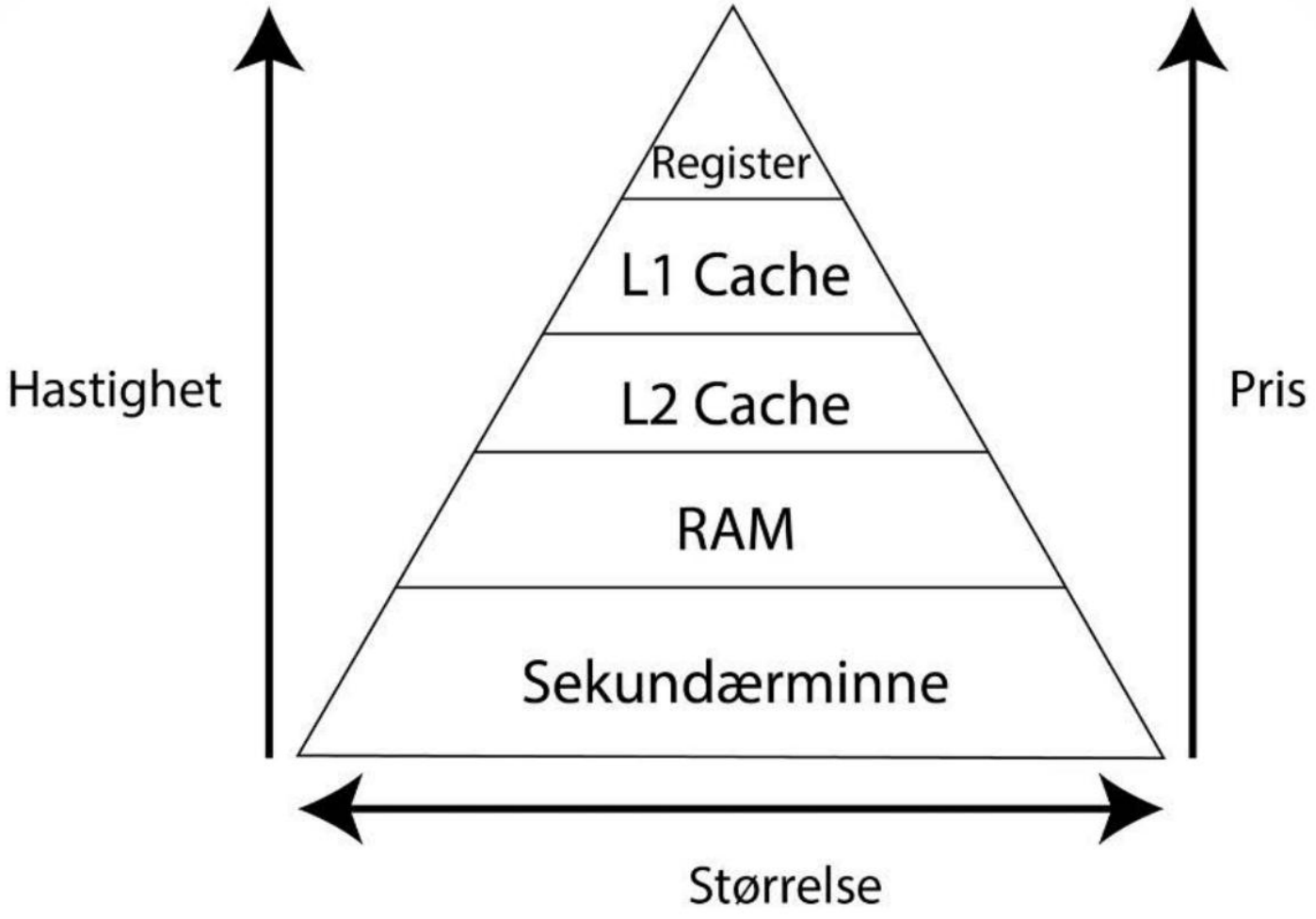


Minne i GPU

- Tidligere brukte GPU → PC-ens RAM.
- Nå: egne minnebanker (VRAM).
- Viktig for spill, KI, simuleringer.

Minnehierarki i hverdagen

- Åpne et program = data lastes fra HDD/SSD → RAM → cache → register.



Oppsummering

- Hierarki = kombinasjon av mange typer minne.
- Raskt og lite på toppen, stort og tregt nederst.
- Balansen gjør datamaskiner effektive og rimelige.