CPU, LMC og abstraksjonsnivåer

Velkommen til ny gruppetime

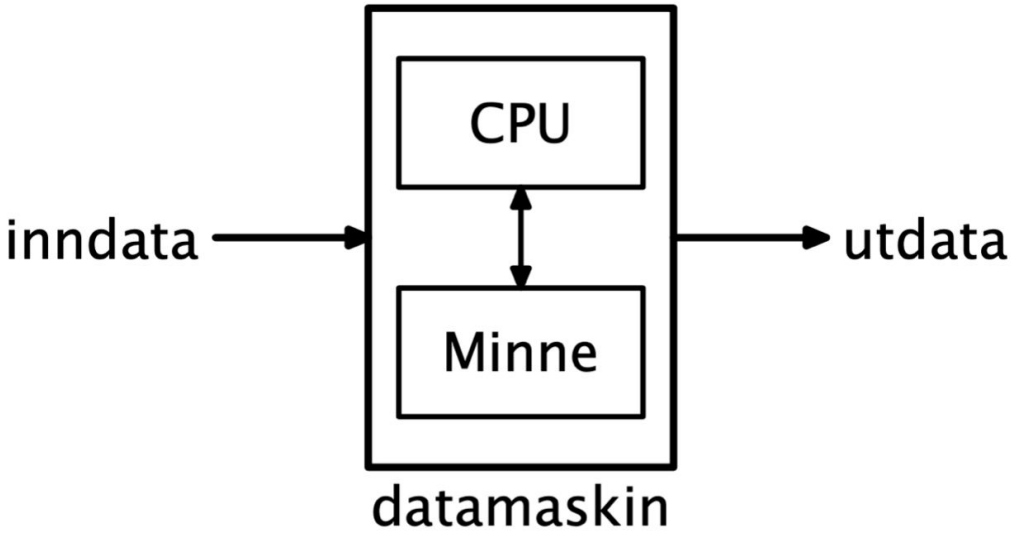
Hva dere får med på eksamen:

- Enkel kalkulator
- ASCII tabell (Hvis dere trenger det)
- Får **IKKE** LMC instruksjoner, må huske assemblerkoden, ikke numerisk kode

CPU og LMC - kort rep. og litt nytt

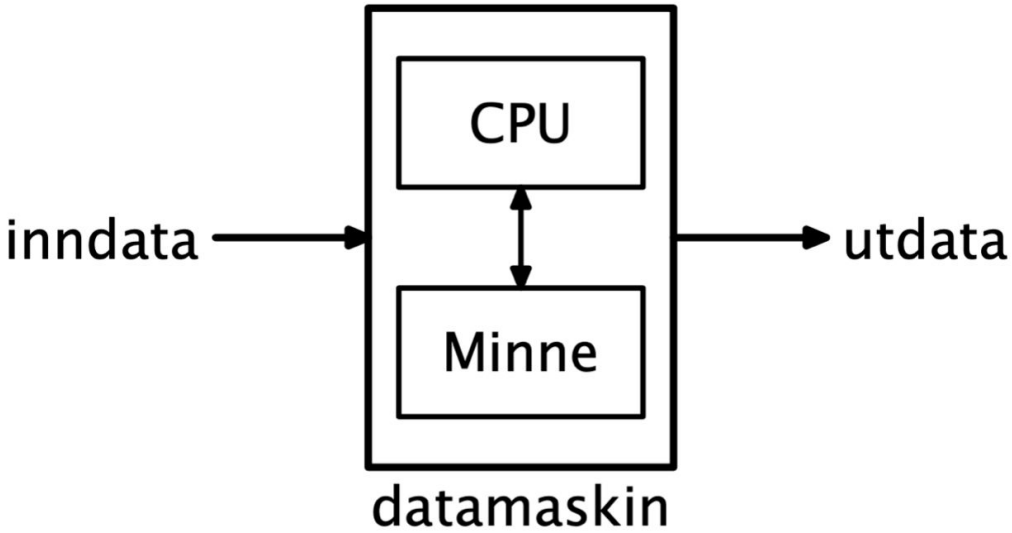
Hva er en prosessor (CPU)?

Tar input, gjør noe med det, sender et output



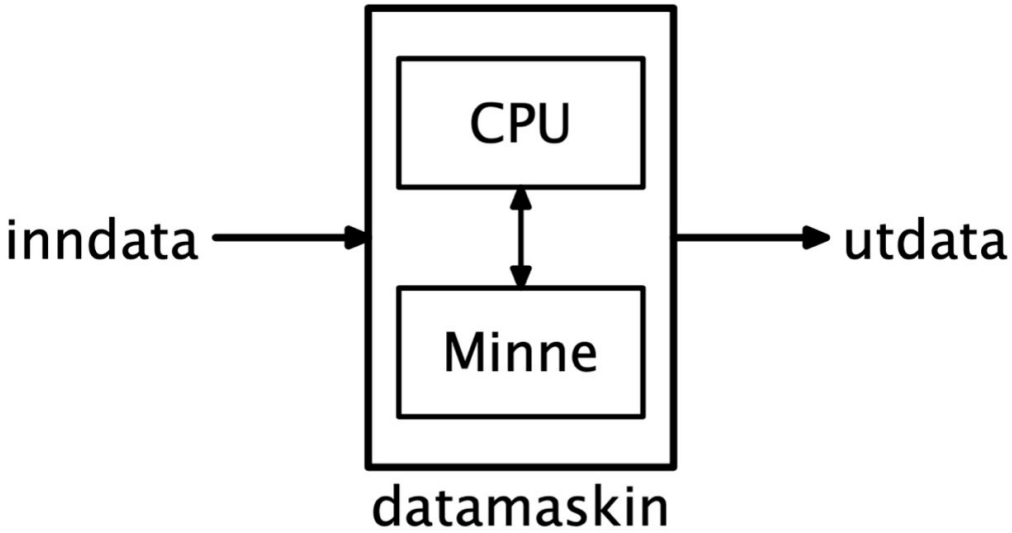
Hva er en prosessor (CPU)?

Tar input, **gjør noe med det,** sender et output



Hva er en prosessor (CPU)?

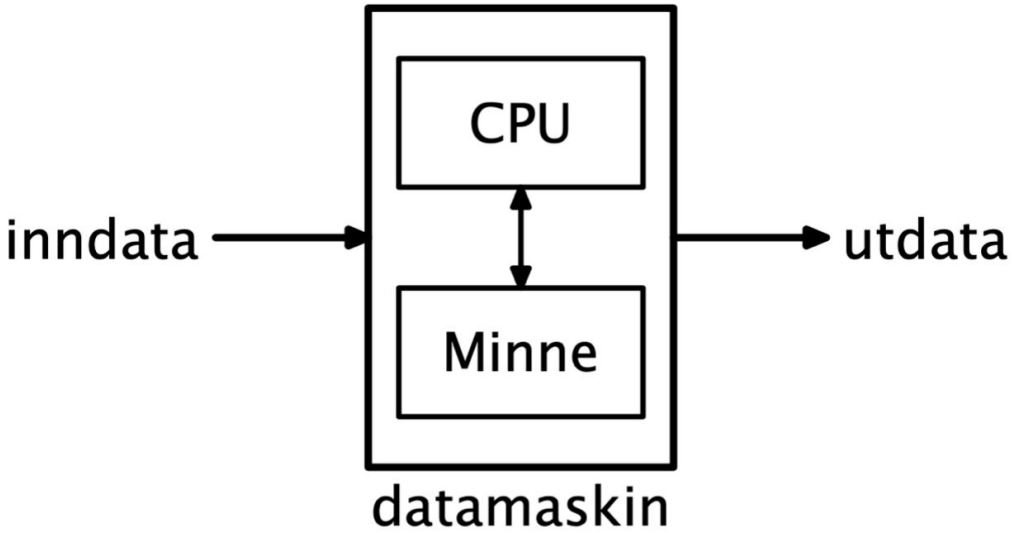
Tar input, gjør noe med det,
sender et output



Hva er en prosessor (CPU)?

Tar input, gjør noe med det, sender et output

Hjernen i datamaskinen!

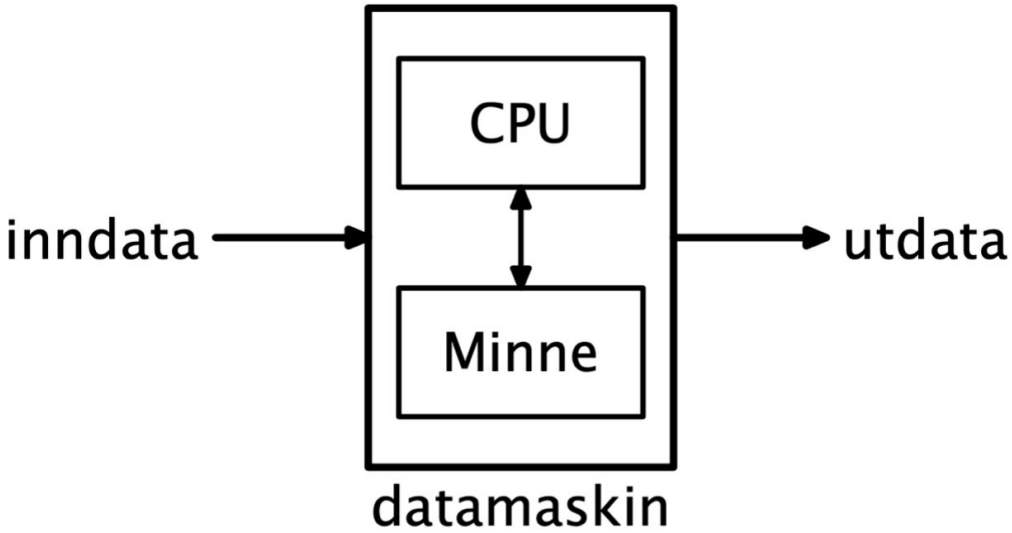


Hva er en prosessor (CPU)?

Tar input, gjør noe med det, sender et output

Hjernen i datamaskinen!

Den har ansvar for å utføre instruksjonene i programmer — det vil si å regne, flytte data og styre resten av systemet



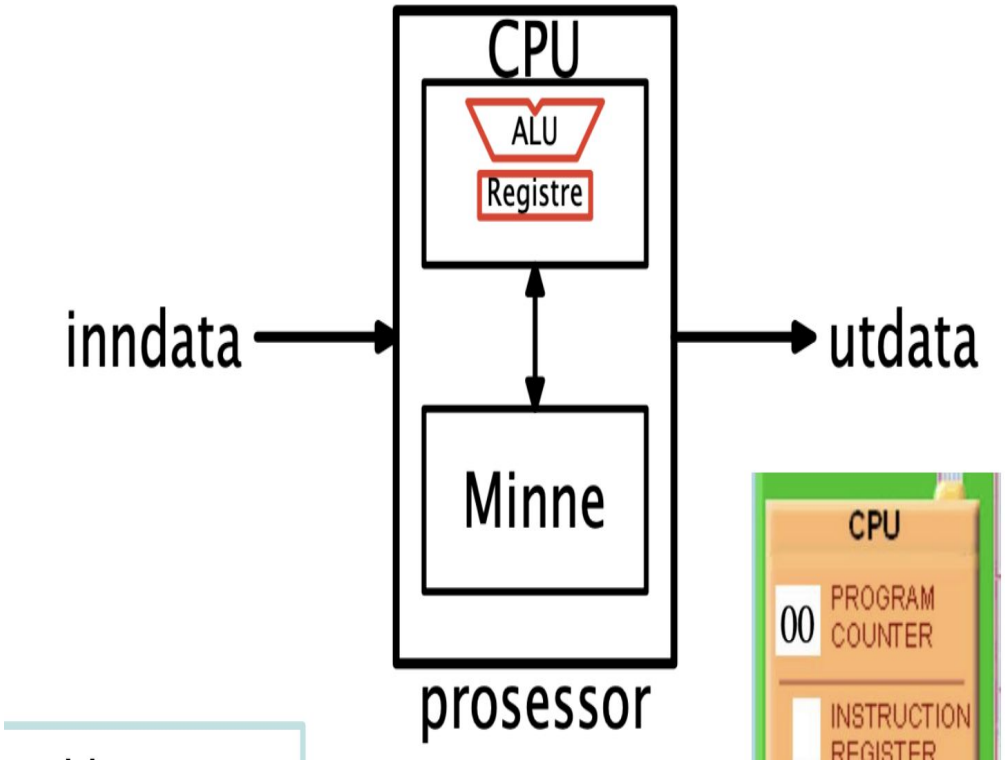
Den styrer alt

Prosessoren er hjernen

Inneholder ALU og registre

ALU: Gjør matematiske beregninger - kalkulatoren i CPU'en

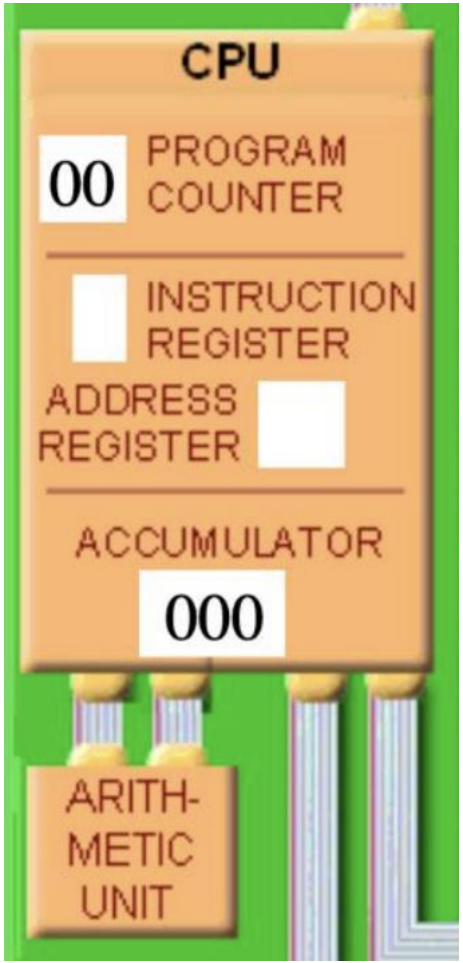
Register: Minne / Lagring i CPU'en



Registre

Minneceller for data CPU’en trenger

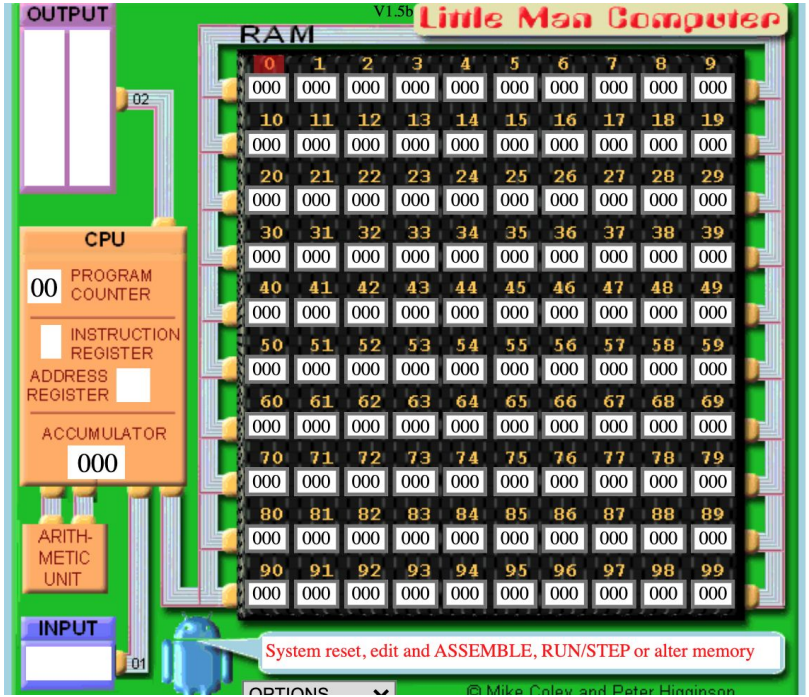
I LMC betyr dette akkumulator, program counter, instruksjonsregister og adresseregister



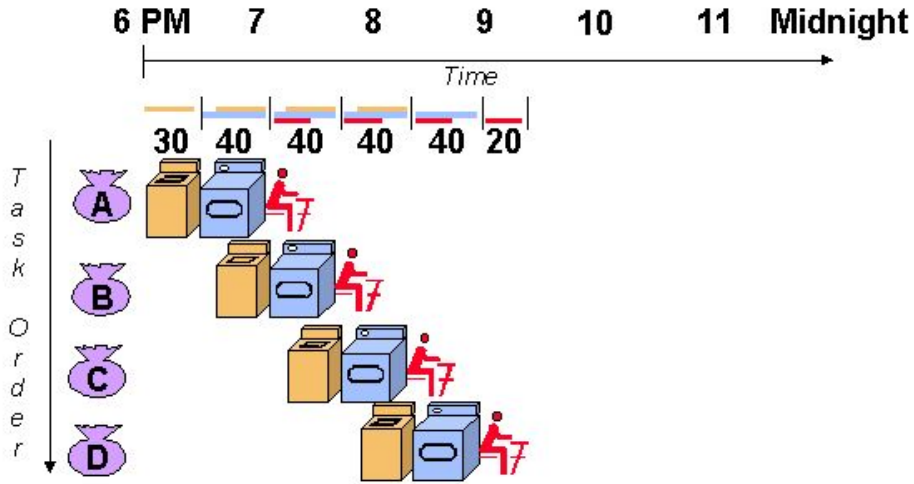
Hvordan henter CPU’en instruksjoner?

Alle CPU-er (fra Little Man Computer til Intel Core i9) følger samme prinsipp:

1. **Hent (Fetch) instruksjon fra minnet (RAM).**
2. **Dekod (Decode) – finn ut hva som skal gjøres.**
3. **Utfør (Execute) operasjonen.**
4. **Skriv tilbake (Write back) hvis nødvendig.**
5. **Gå til neste instruksjon → gjenta.**



Forskjell på x86 og LMC - Hvorfor er x86 bedre?

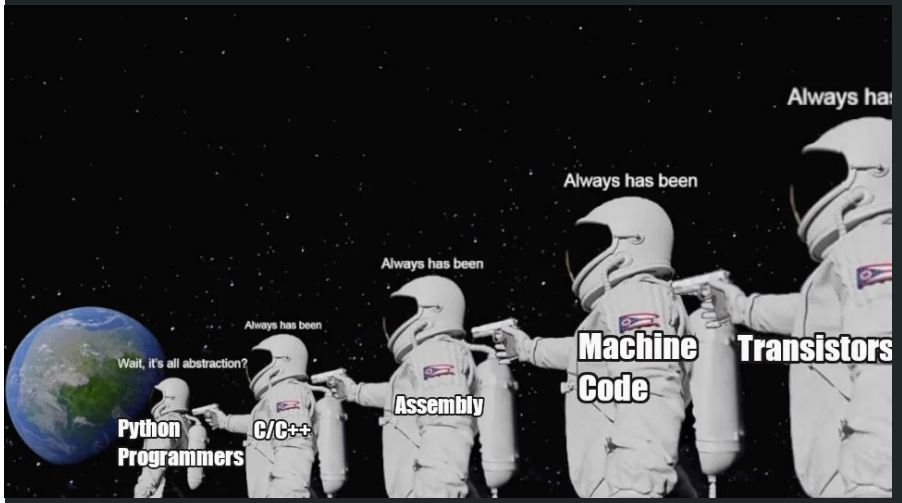


1. Fetch
2. Decode
3. Execute
4. Write back
5. Gå til neste instruksjon ➜ gjenta.

Abstraksjonsnivåer

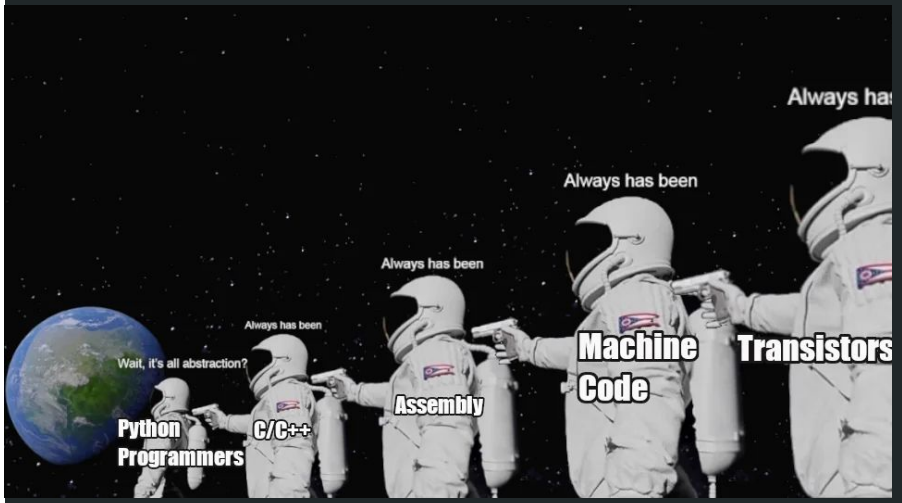
Hva er det?

Datamaskiner er bygd lag på lag — fra fysikk og elektroner (Hardware) helt opp til applikasjoner (Google, Word, Spotify etc.)



Hva er det?

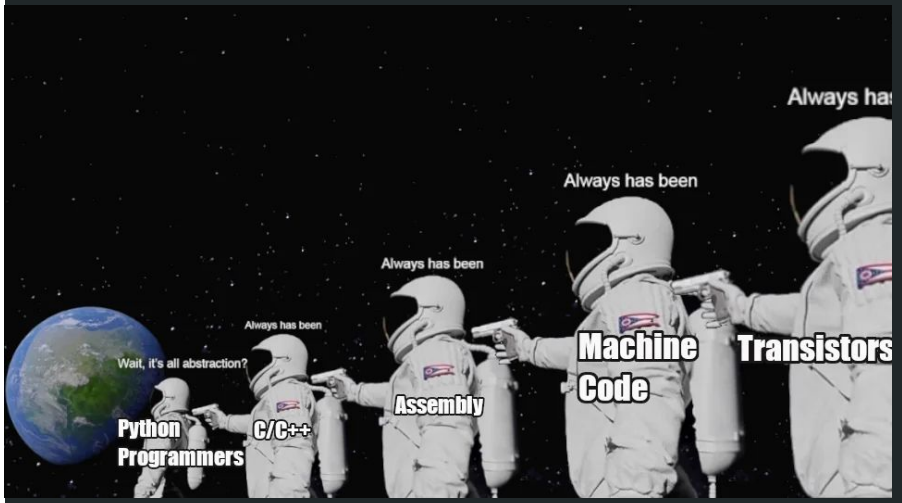
Lagene skjuler kompleksitet (vanskelige ting) under seg, slik at man ikke trenger å forholde seg til det



Hva er det?

Lagene skjuler kompleksitet (vanskelige ting) under seg, slik at man ikke trenger å forholde seg til det

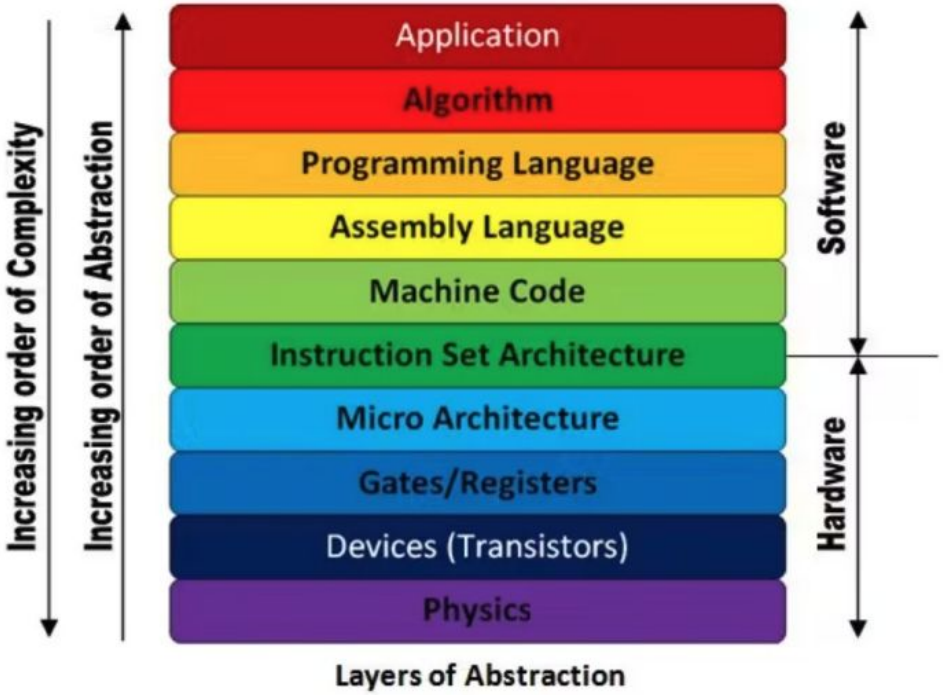
Du trenger ikke vite hvordan man bygger en bil for å kjøre en bil



Du trenger ikke vite hvordan elektronikken i huset fungerer for å bruke en lysbryter, du forholder deg til sikringskapet (Abstraksjon)

Du trenger ikke vite hvordan elektronikken i huset fungerer for å bruke en lysbryter, du forholder deg til sikringskapet (Abstraksjon)

Når du er på restaurant trenger du ikke vite hvordan kokken lager mat, du vet bare at du får mat (Abstraksjon)


Application
Algorithm
Programming Language
Assembly Language
Machine Code
Instruction Set Architecture
Micro Architecture
Gates/Registers
Devices (Transistors)
Physics
Increasing order of Complexity
Increasing order of Abstraction
Software
Hardware
Layers of Abstraction

[Du spør ChatGPT] -> [ChatGPT forklarer Python-kode] -> [Du skriver: print("Hei")]

-> [Python tolker ➜ C-kode ➜ kompilert til maskinkode] -> [Maskinkode og OS ➜ CPU-instruksjoner]

-> [Maskinkode ➜ CPU-instruksjoner] -> [Bits strømmer i kretsene ⚡]

[Du spør ChatGPT] -> **[ChatGPT forklarer Python-kode]** -> [Du skriver: print("Hei")]

-> [Python tolker ➜ C-kode ➜ kompilert til maskinkode] -> [Maskinkode og OS ➜ CPU-instruksjoner]

-> [Maskinkode ➜ CPU-instruksjoner] -> [Bits strømmer i kretsene ⚡]

[Du spør ChatGPT] -> [ChatGPT forklarer Python-kode] **-> [Du skriver: print("Hei")]**

-> [Python tolker ➜ C-kode ➜ kompilert til maskinkode] -> [Maskinkode og OS ➜ CPU-instruksjoner]

-> [Maskinkode ➜ CPU-instruksjoner] -> [Bits strømmer i kretsene ⚡]

[Du spør ChatGPT] -> [ChatGPT forklarer Python-kode] -> [Du skriver: print("Hei")]

-> [Python tolker ➜ C-kode ➜ kompilert til maskinkode] -> [Maskinkode og OS ➜ CPU-instruksjoner]

-> [Maskinkode ➜ CPU-instruksjoner] -> [Bits strømmer i kretsene ⚡]

[Du spør ChatGPT] -> [ChatGPT forklarer Python-kode] -> [Du skriver: print("Hei")]

-> [Python tolker ➜ C-kode ➜ kompilert til maskinkode] -> **[Maskinkode og OS ➜ CPU-instruksjoner]**

-> [Maskinkode ➜ CPU-instruksjoner] -> [Bits strømmer i kretsene ⚡]

[Du spør ChatGPT] -> [ChatGPT forklarer Python-kode] -> [Du skriver: print("Hei")]

-> [Python tolker ➜ C-kode ➜ kompilert til maskinkode] -> [Maskinkode og OS➜ CPU-instruksjoner]

-> [Maskinkode ➜ CPU-instruksjoner] -> [Bits strømmer i kretsene ⚡]

[Du spør ChatGPT] -> [ChatGPT forklarer Python-kode] -> [Du skriver: print("Hei")]

-> [Python tolker ➜ C-kode ➜ kompilert til maskinkode] -> [Maskinkode og OS ➜ CPU-instruksjoner]

-> [Maskinkode ➜ CPU-instruksjoner] **-> [Bits strømmer i kretsene ⚡]**

Dere har allerede vært borti abstraksjonsnivåer i IN1020

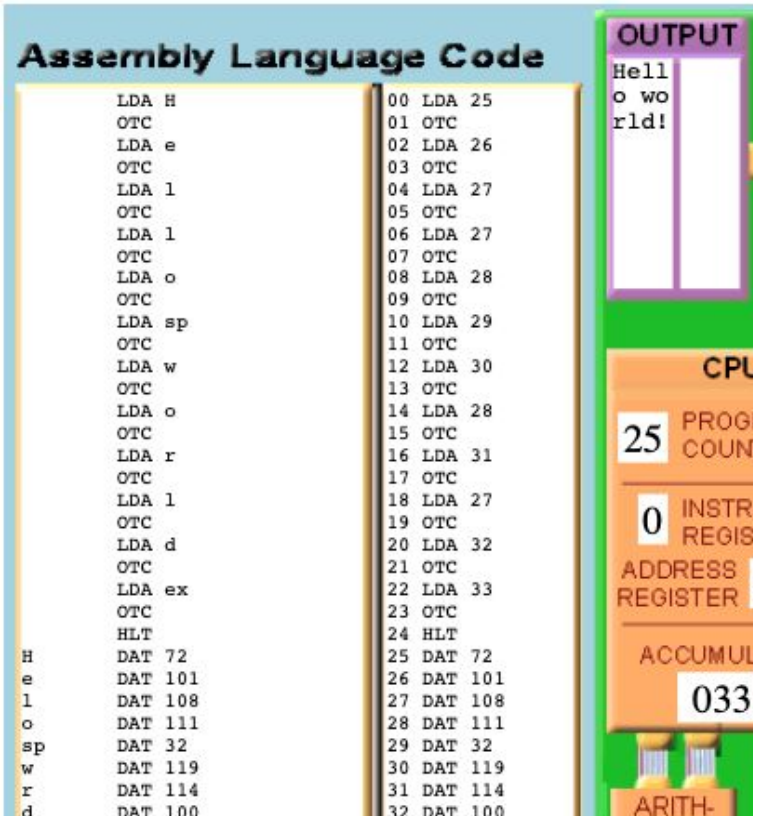
Lag		Funksjon
5	**Applikasjon**	Applikasjonsrelaterte tjenester (**HTTP**, **Mail**)
4	**Transport**	Kobler sammen systemene ende-til-ende (**TCP/UDP**)
3	**Nettverk**	Sende data fra ende-til-ende systemer (**IP**)
2	**Link**	Pålitelig overføring mellom to noder (**LAN/WiFi**)
1	**Fysisk**	Sender bit ut på mediet (kablet eller trådløst)

Noen andre eksempler fra IN1020 hvor dere har vært borti abstraksjonsnivåer?

Python

```
print("Hello, World!")
```

LMC



Operativsystemet abstraherer vekk detaljer

OS gjemmer vekk de tekniske detaljene i maskinvaren, og gir programmer og brukere et enklere og mer standardisert grensesnitt å jobbe med.



Ranger fra høyest til lavest

1. **LDA 99**
2. **def hei():**

   print(“hei”)

3. **“Hei Siri, sett sangen på pause”**
4. **CPU**
5. **TCP**
6. **En byte**

Ranger fra høyest til lavest

1. LDA 99
2. def hei():

   print(“hei”)

3. “Hei Siri, sett sangen på pause”
4. CPU
5. TCP
6. En byte

1. "Hei Siri, sett sangen på pause"
2. def hei():

   print("hei")

3. TCP
4. LDA 99
5. En byte
6. CPU

1.4 Godt og Blandet

Sett riktig eller galt for påstandene under.

	Riktig	Galt
En maskin som er bygget på von neumann arkitektur har både kode og data i samme minne.	●	○
DAT er en vanlig instruksjon som LMC forstår.	○	●
Én byte kan representere 512 unike verdier.	○	●
ASCII har plass til 128 unike kontroll- eller karakter-symboler.	●	○
ASCII oversettes enkelt til UTF8 ved å sette det øverste bitet til null.	●	○
Vektorgrafikk er et vel egnet format for å lagre bilder tatt med smart-telefon.	○	●
LMC har totalt tre interne registre: program-teller, instruksjonsregister, og akkumulator.	○	●
LMC forstår totalt 9 forskjellige instruksjonstyper.	●	○