# Velkommen!

CPU-en den sentrale prosesseringsenheten

Gruppetime 4

(Hanne-)Kristin Schrøder Leiros

hkleiros@uio.no

# Plan for dagen

- Info om oblig 2
- Repetisjon
- Jobbing med ukesoppgaver eller oblig

- Slutt 15:50 ☺

# Oblig 2 publisert

- Innleveringsfrist 4. oktober kl. 23:59

- A) LMC program som tar inn n-antall tall,
  - Beregne og skrive ut summen av tallene,
  - Beregne rest av deling på 2, for å kunne skrive ut om tallet er et heltall eller oddetall
  - En .txt fil som kan lastes opp til https://peterhigginson.co.uk/lmc/

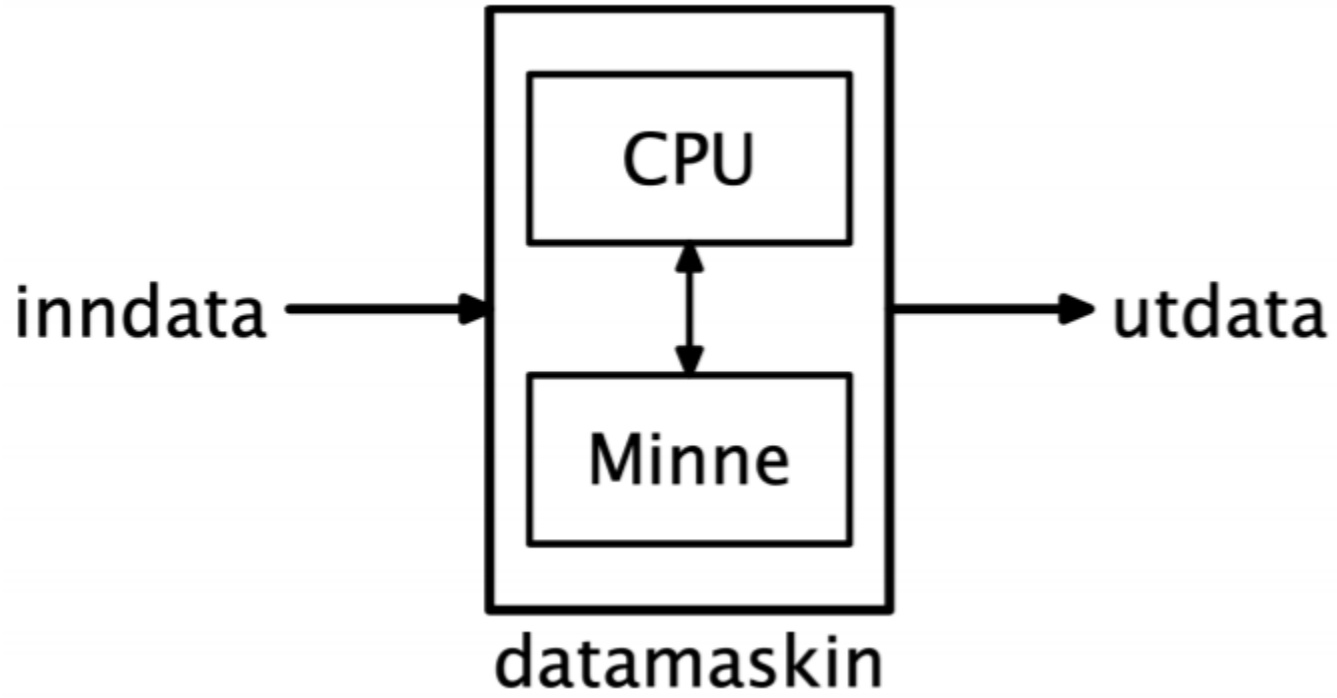
- B) Lage ditt eget tallsystem med base 30
  - Symbolvalg og begrunnelse
  - Konvertering
  - Utregning
  - En .pdf fil

# Læringsmål

- Forklare Von-Neumann arkitekturen
- Forklare hva en CPU er og hvordan en forenklet modell er bygget opp
- Kunne programmere enkle programmer ved bruk av LMC simulatoren

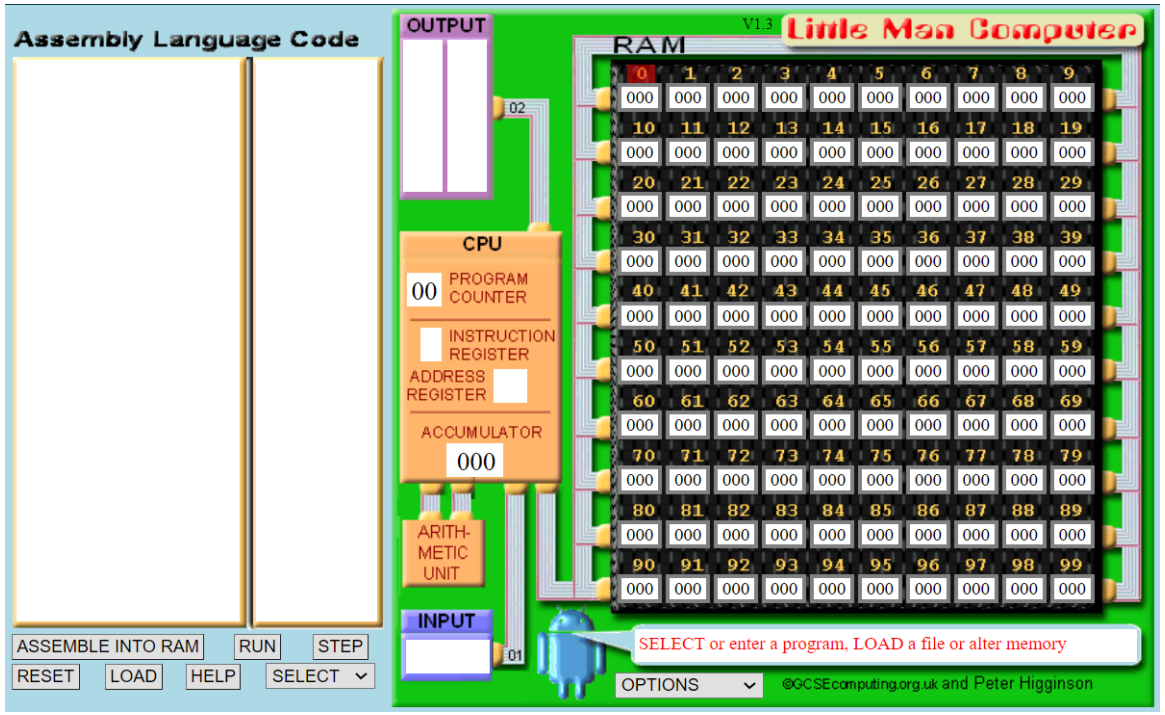
# Von-Neuman arkitekturen

- Minnet inneholder både data og instruksjoner til CPU-en



# Little Man Computer

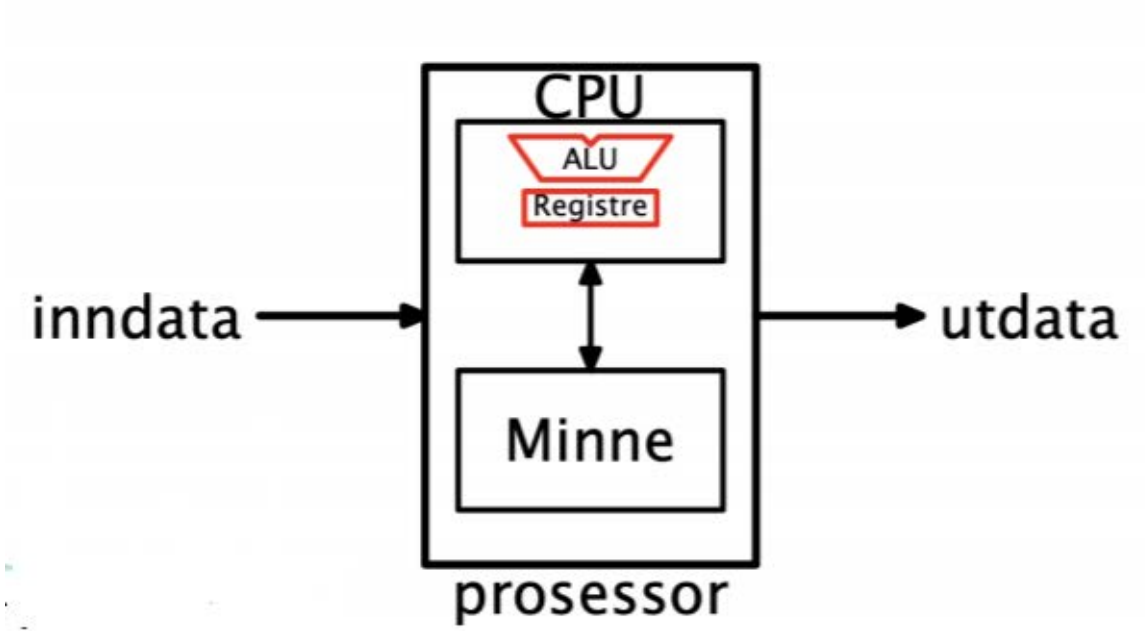
- Meget enkel, men teoretisk korrekt datamaskin
- Lære om hvordan prosessorer fungerer
- Simulator som bruker desimaltall
- Minnet og instruksjoner samlet
  - Posthylle nummerert fra 0-99 og hver hylle kan lagre ett tall fra 0 til 999
- Hvis man vil skrive ut tegn bruker man ASCII-verdien til tegnene



# Central Processing Unit

- Inneholder
  - ALU: Aritmetic Logic Unit
    - Brukes til aritmetiske og logiske beregninger
  - Registre
  - Kontrolllogikk til å utføre instruksjonene

- LMC har en aritmetisk unit, som kan addere og subtrahere heltall men ikke utføre logiske beregninger



# Registre

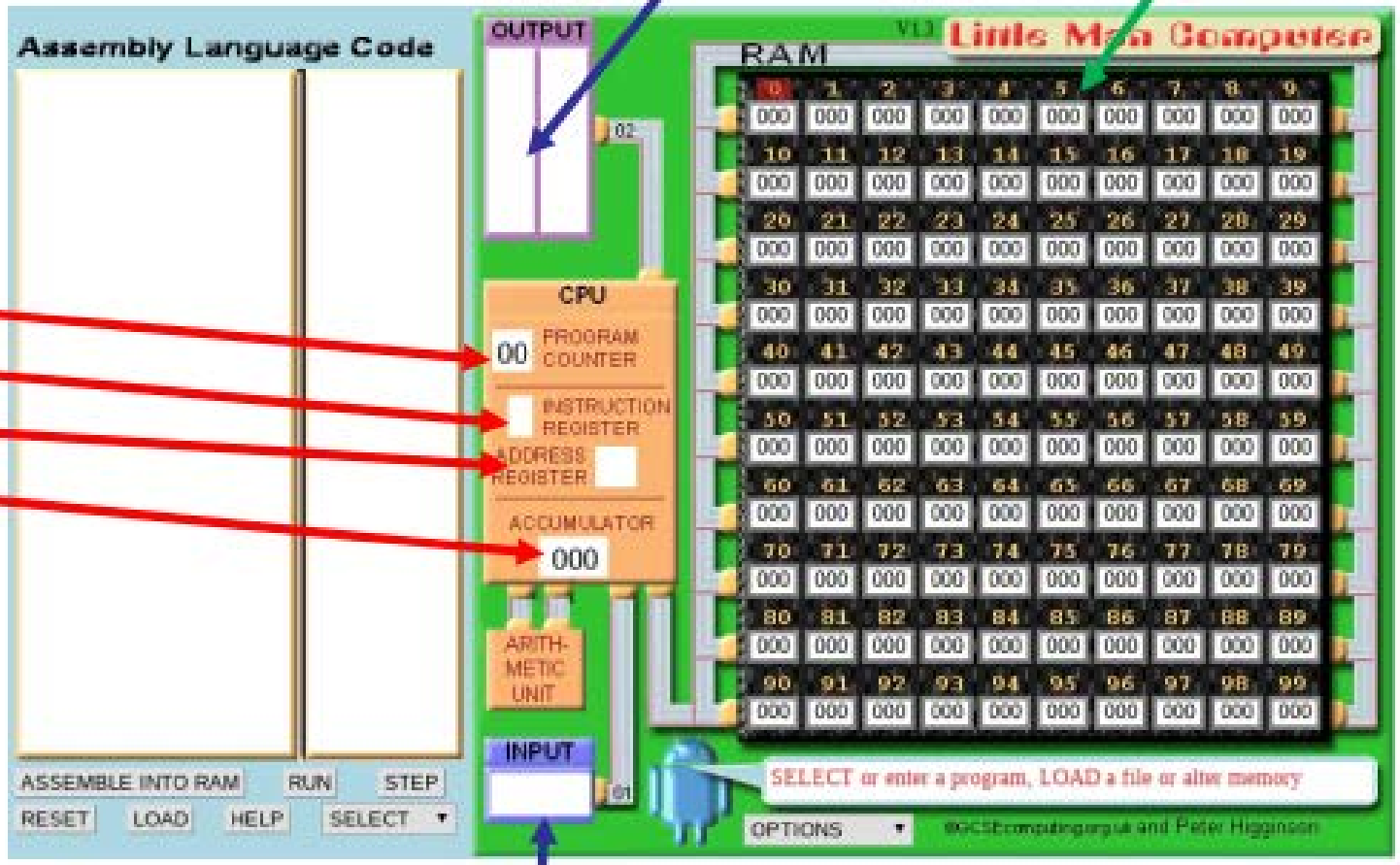
- Lagerceller for data som CPU-en trenger hurtig tilgang til
- I LMC:
  - **Accumulator** til beregninger
    - Svaret kommer hit
  - **Program counter** / location counter
    - Angir hvilken instruksjon i minnet som skal utføres neste gang
  - **Instruction register**
    - Koden til den instruksjonen som utføres
  - **Address register**
    - Inneholder adressedelen av den instruksjonen som utføres

# LMC-overblikk

Resultater skrives ut her

Minne med 100 celler for program og data

Fire registre
- Program counter
- Instruction register
- Address register
- Accumulator




LMC kan lese tall som brukeren (vi) skriver inn

# LMC instruksjoner

- Hver instruksjon er et tall fra 0 til 999
- Første siffer angir instruksjonskoden
- Andre og tredje siffer angir en adresse i minnet

| 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) |

# ASCII tegn

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | LF | 39 | ' | 47 | / | 62 | > | 93 | ] | 124 | \| |
| 32 | | 40 | ( | 48 | 0 | 63 | ? | 94 | ^ | 125 | } |
| 33 | ! | 41 | ) | : | | 64 | @ | 95 | _ | 126 | ~ |
| 34 | " | 42 | * | 57 | 9 | 65 | A | 96 | ‘ | | |
| 35 | # | 43 | + | 58 | : | : | | 97 | a | | |
| 36 | $ | 44 | , | 59 | ; | 90 | Z | : | | | |
| 37 | % | 45 | - | 60 | < | 91 | [ | 122 | z | | |
| 38 | & | 46 | . | 61 | = | 92 | \ | 123 | { | | |

De mest nødvendige tegnene, i base 10.

# Instruksjoner – assemblerkode og tall

- HLT – 000
  - Stopper kjøringen
- INP – 901
  - Ber brukeren skrive et tall
  - Legger tallet i akkumulatorregisteret
- OUT – 902
  - Skriver ut verdien i akkumulatoren
- STA – 3xx
  - «Store accumulator»
  - Kopier verdien i akkumulatoren til adresse xx i minnet
- LDA – 5xx
  - «Load accumulator»
  - Kopierer verdien i lokasjon xx til akkumulatoren
- ADD – 1xx
  - Adderer verdien i adresse xx i minnet til akkumulatoren
- SUB – 2xx
  - Trekker verdien i adresse xx i minnet fra akkumulatoren
- BRA – 6xx
  - «Branch»
  - Hopper til adresse xx
- BRP – 8xx
  - «Branch if positive»
  - Hopper til adresse xx HVIS akkumulatoren inneholder en verdi som er >=0
- BRZ – 7xx
  - «Branch if zero»
  - Hopper til adresse xx HVIS akkumulatoren inneholder verdien 0,

# Nå har vi vært innom alle instruksjonene
# Hva kan vi nå programmere?

- Variabler (med DAT)
- Regning
  - Legge sammen (med ADD)
  - Trekke fra (med SUB)
  - Gange
  - Dele (med kode fra ukeoppgave)
- Tilordning (med STA)
- If-tester (med BRZ og BRP)
- While-løkker (med BRA, BRZ og BRP)

# Takk for i dag ☺

Neste uke: operativsystemer, programmering av prosessoren