

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

CPU-en, den sentrale prosesseringsenheten

Eyvind W. Axelsen (eyvinda@ifi.uio.no)
Foiler er basert på tidligere materiale laget av Dag Langmyhr

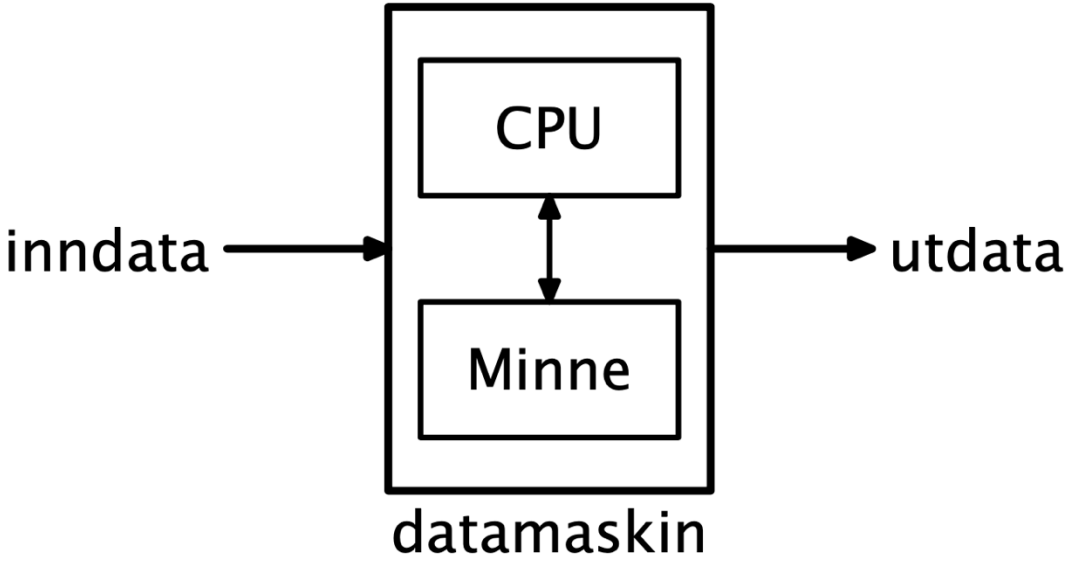




I dag:

- En første titt på LMC, og simulatoren for den
- De første instruksjonene for LMC
- Programmering med instruksjoner
- Sannsynligvis en litt kort forelesning – bruk resten av tiden til å kose deg med LMC på egen hånd ☺

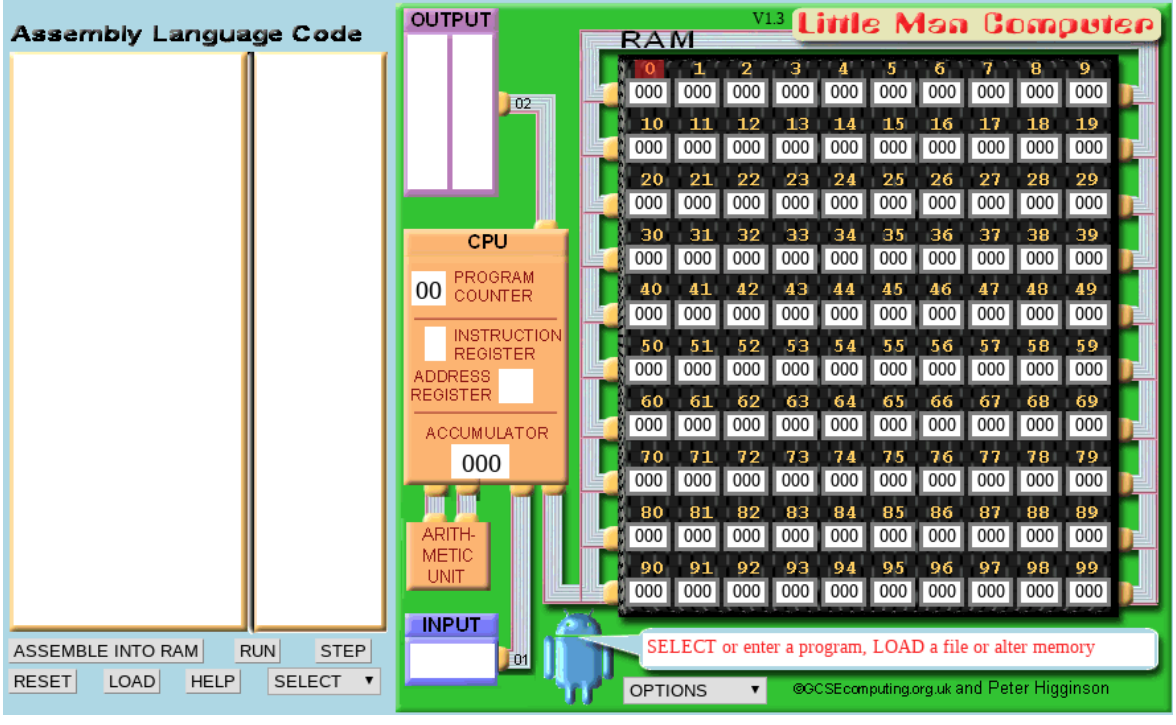
Von Neumann-arkitekturen



Den store erkjennelsen er at minnet holder på både *data* og *instruksjoner* til CPU-en (altså *programmet*)

LMC – Little Man Computer

En meget enkel, men teoretisk korrekt, datamaskin



Vi skal bruke en simulator laget av Peter Higginson,
https://peterhigginson.co.uk/lmc/

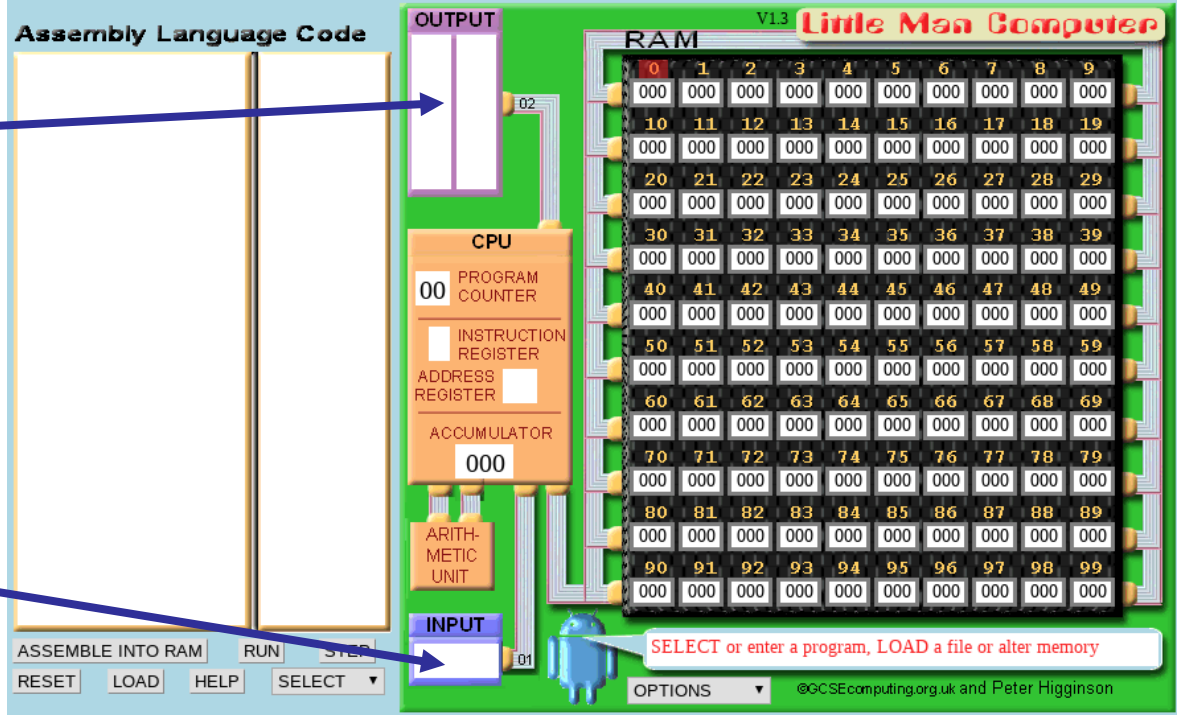
NB: LMC-simulatoren bruker desimaltall, og ikke binær/heksadesimal

For «ordentlige» datamaskiner kan input komme fra mus, tastatur, pen, berøring, sensorer, etc; og output kan gå til skjerm, høyttaler, skriver, etc

Data inn og ut

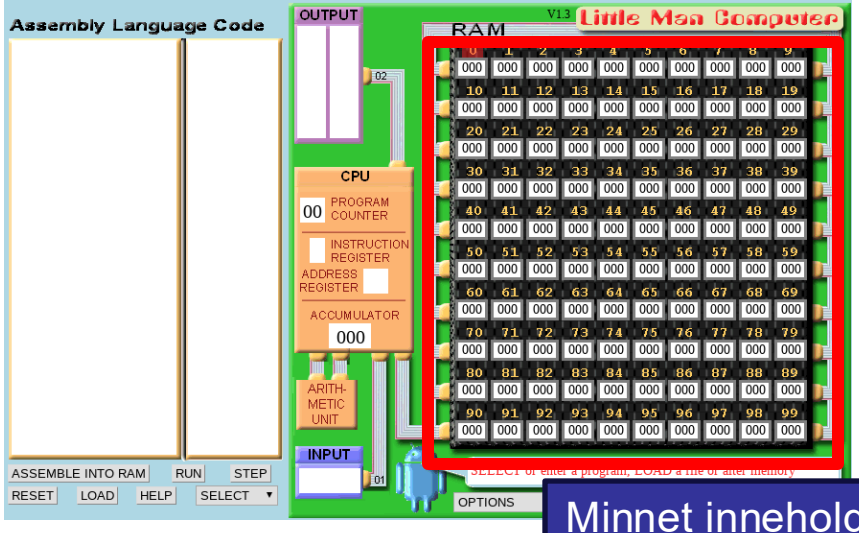
Resultatet kommer ut som tall som LMC skriver ut

LMC kan lese tall som brukeren (vi) skriver inn



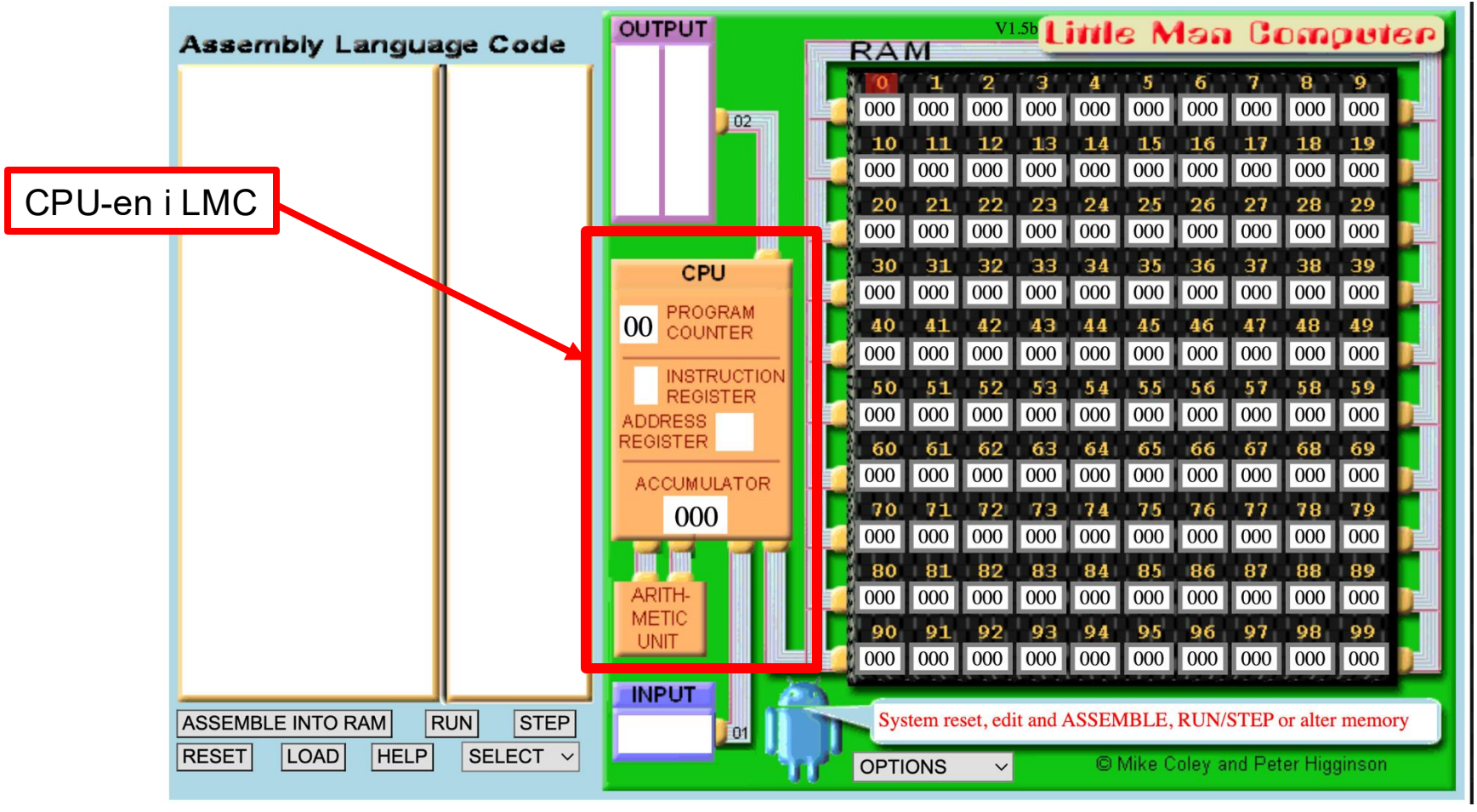
Minnet i LMC kan sees på som en posthylle nummerert fra 0-99. Hver «hylle» kan lagre ett tall.





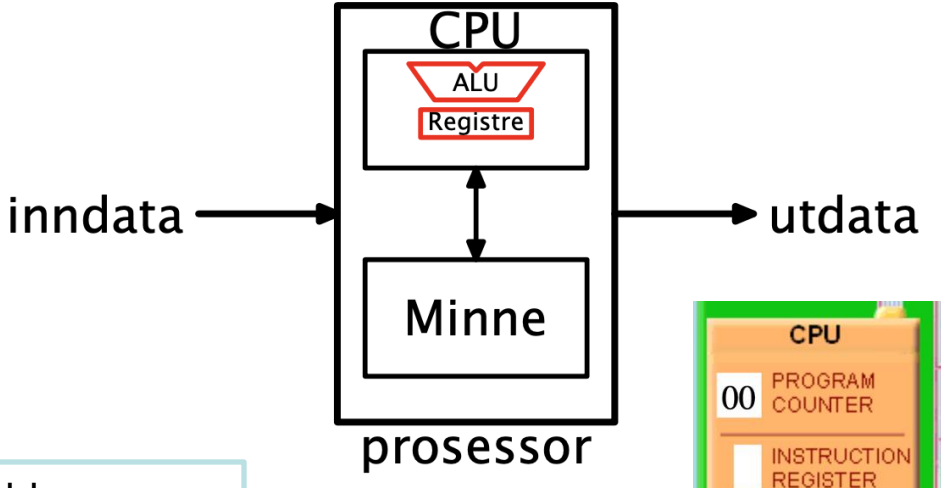
Minnet inneholder både program og data!

Ekte datamaskiner har typisk et minne på 4-32 GB (gigabyte)
(1 GB = 1 073 741 824 byte)


CPU-en i LMC
Assembly Language Code
OUTPUT
V1.5b
Little Man Computer
RAM
CPU
00
PROGRAM COUNTER
INSTRUCTION REGISTER
ADDRESS REGISTER
ACCUMULATOR
000
ARITH-METIC UNIT
INPUT
ASSEMBLE INTO RAM
RUN
STEP
RESET
LOAD
HELP
SELECT
System reset, edit and ASSEMBLE, RUN/STEP or alter memory
OPTIONS
© Mike Coley and Peter Higginson

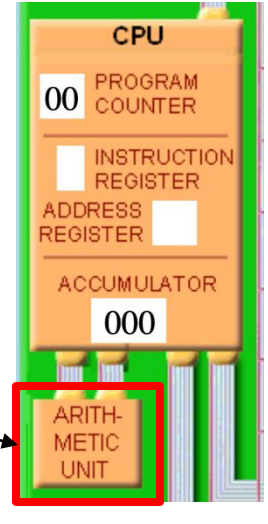
CPU («Central Processing Unit») er den sentrale «hjernen» i en datamaskin. Den inneholder:

- ALU («Arithmetic Logic Unit») til aritmetiske og logiske beregninger
- Registre
- Kontrollogikk til å utføre instruksjonene



ALU-en (regneenheten) i LMC kan addere og subtrahere heltall

ALU-en i en moderne datamaskin kan alle vanlige regnearter for heltall av ulik lengde, og for flyt-tall.

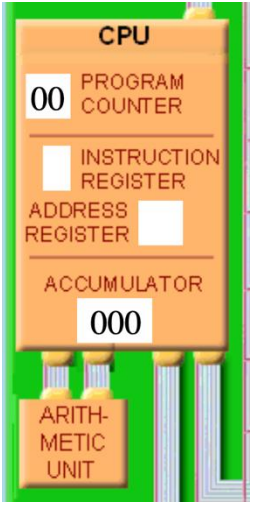


Registre

Inne i CPU-en finnes noen få lagerceller for data som CPU-en trenger hurtig tilgang til. Disse kalles **registre**.

LMC har fire registre:

- *Accumulator* er til beregninger; svaret kommer hit.
- *Program counter* (alias *location counter*) angir hvilken instruksjon i minnet som neste gang skal utføres.
- *Instruction register* inneholder koden til den instruksjonen som utføres.
- *Address register* inneholder adressedelen av den instruksjonen som utføres.



Moderne prosessorer har fra 20 til et par hundre registre.

Instruksjoner

Programmet lagres på numerisk form i minnet, som en sekvens av instruksjoner. Hver instruksjon er et tall 0–999 der

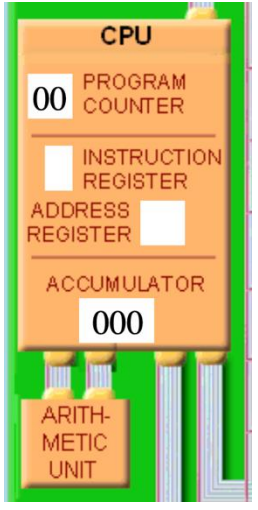
- 1. siffer angir instruksjonskoden og
- 2. og 3. siffer angir en adresse i minnet (om det er aktuelt).

- **Eksempel: 542** betyr *«Hent verdien i adresse **42** i minnet og legg den i akkumulatorregisteret.»*
  (Kode **5** angir at prosessoren skal hente noe i minnet.)

- LMC har 9 forskjellige instruksjoner

Eksekvering av programmer i LMC

1. Bruk verdien i **programtelleren** som adresse til minnet og hent neste instruksjon der.
   – Øk samtidig programtelleren med 1
2. Splitt instruksjonen og legg delene i **instruksjonsregisteret** og **adresseregisteret**.
3. Utfør det som instruksjonsregisteret angir.
4. Gjenta fra punkt 1



Instruksjonen HLT

Instruksjonen **HLT** («Halt», også kalt COB = «Coffee break» i boka) har kode **000**. Den stopper kjøringen.

Eksempel

Dette minimale programmet gjør ingen ting, det bare stopper:

```
000
```

Instruksjonene INP og OUT

Instruksjonen **INP** («Input», kalt IN i boka) har kode **901**. Den ber brukeren skrive et tall, og dette tallet legges i akkumulatorregisteret.

Instruksjonen **OUT** («Output») har kode **902**. Den skriver ut verdien i akkumulatoren.

Eksempel

Dette lille programmet leser et tall og skriver det ut igjen før det stopper:

```
901
902
000
```

La oss prøve det ut i LMC →

Instruksjonene STA og LDA

Instruksjonen **STA** («Store accumulator», kalt STO i boka) har kode **3xx**, og den kopierer verdien i akkumulatoren til adresse xx i minnet.

Instruksjonen **LDA** («Load accumulator») har kode **5xx**, og den kopierer verdien i lokasjon xx til akkumulatoren.

Instruksjonene ADD og SUB

Instruksjonen **ADD** har kode **1xx** og adderer verdien i adresse xx i minnet til akkumulatoren.

Instruksjonen **SUB** («Subtract») har kode **2xx**, og den trekker verdien i adresse xx i minnet fra akkumulatoren.

Et eksempel

Dette lille programmet (som også står i Englander-boka) vil

1. lese inn et tall og legge det i adresse 99 i minnet,
2. lese et tall til,
3. addere tallet i adresse 99 med det sist innleste tallet og
4. skrive ut svaret.

```
901
399
901
199
902
000
```

La oss prøve det ut i LMC →

Assembler

Det er litt klønete å huske/skrive/lese numeriske koder. Det er bedre om vi kan navngi instruksjonene.

```
INP        // Les et tall
STA   99   // og legg det i celle 99.
INP        // Les et tall til
ADD   99   // og adder med tallet i celle 99.
OUT        // Skriv ut svaret.
HLT       // Stopp.
```

Denne typen kode kalles «assemblerkode».

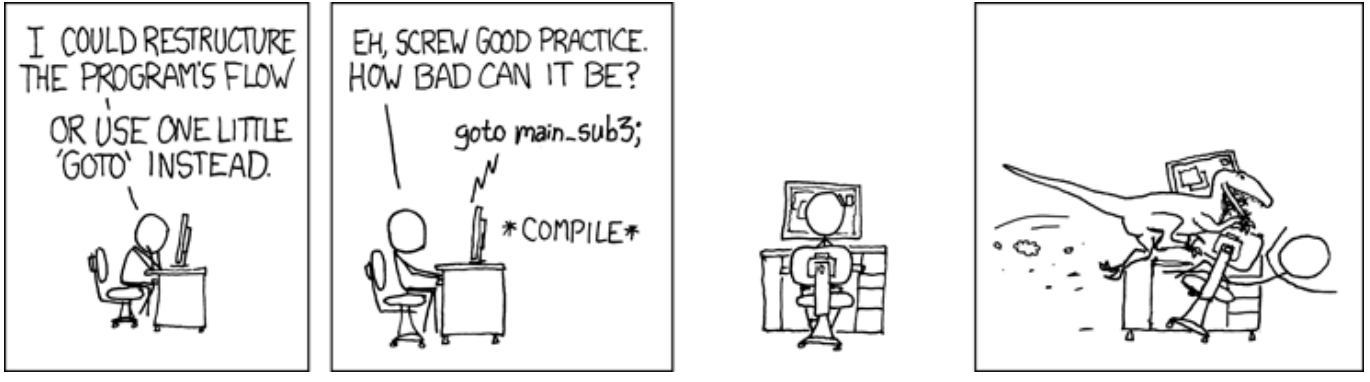
La oss prøve det ut i LMC →

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)

På torsdag

- Flere instruksjoner for LMC
- Mer programmering!



https://xkcd.com/292/