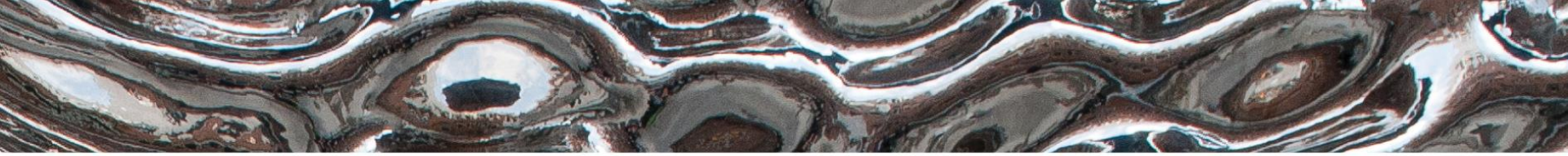


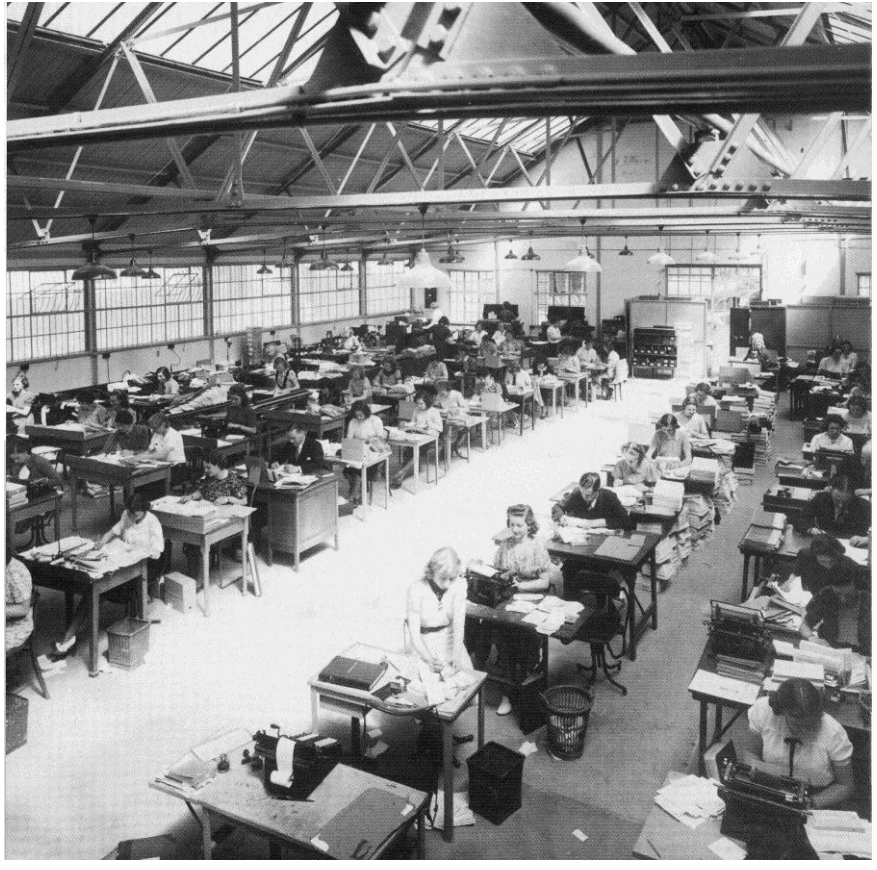
UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

CPU-en, den sentrale prosesseringsenheten

Kristoffer Robin Stokke (krisrst@ifi.uio.no)
Foiler er basert på tidligere materiale laget av Dag Langmyhr

Inspirasjonen

	0	1	2	3	4	5	6	7	8	9	1 2 3	4 5 6	7 8 9
1·0	·0000	0043	0086	0128	0170	0212	0253	0294	0334	0374	4 8 12	17 21 25	29 33 37
1·1	·0414	0453	0492	0531	0569	0607	0645	0682	0719	0755	4 8 11	15 19 23	26 30 34
1·2	·0792	0828	0864	0899	0934	0969	1004	1038	1072	1106	3 7 10	14 17 21	24 28 31
1·3	·1139	1173	1206	1239	1271	1303	1335	1367	1399	1430	3 6 10	13 16 19	23 26 29
1·4	·1461	1492	1523	1553	1584	1614	1644	1673	1703	1732	3 6 9	12 15 18	21 24 27
1·5	·1761	1790	1818	1847	1875	1903	1931	1959	1987	2014	3 6 8	11 14 17	20 22 25
1·6	·2041	2068	2095	2122	2148	2175	2201	2227	2253	2279	3 5 8	11 13 16	18 21 24
1·7	·2304	2330	2355	2380	2405	2430	2455	2480	2504	2529	2 5 7	10 12 15	17 20 22
1·8	·2553	2577	2601	2625	2648	2672	2695	2718	2742	2765	2 5 7	9 12 14	16 19 21
1·9	·2788	2810	2833	2856	2878	2900	2923	2945	2967	2989	2 4 7	9 11 13	16 18 20
2·0	·3010	3032	3054	3075	3096	3118	3139	3160	3181	3201	2 4 6	8 11 13	15 17 19
2·1	·3222	3243	3263	3284	3304	3324	3345	3365	3385	3404	2 4 6	8 10 12	14 16 18
2·2	·3424	3444	3464	3483	3502	3522	3541	3560	3579	3598	2 4 6	8 10 12	14 15 17
2·3	·3617	3636	3655	3674	3692	3711	3729	3747	3766	3784	2 4 6	7 9 11	13 15 17
2·4	·3802	3820	3838	3856	3874	3892	3909	3927	3945	3962	2 4 5	7 9 11	12 14 16
2·5	·3979	3997	4014	4031	4048	4065	4082	4099	4116	4133	2 3 5	7 9 10	12 14 15
2·6	·4150	4166	4183	4200	4216	4232	4249	4265	4281	4298	2 3 5	7 8 10	11 13 15



- Arbeidsbeskrivelse

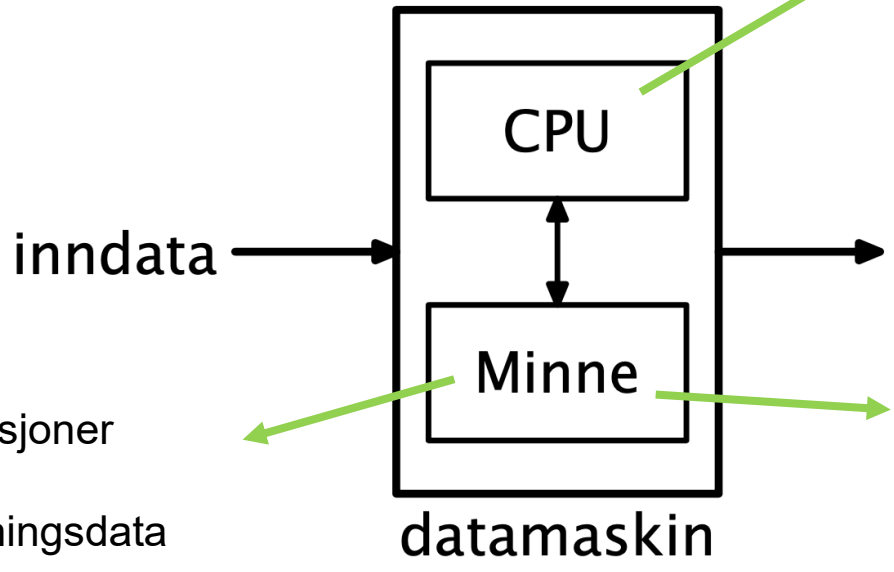
- Inndata og resultater

- Regnemaskin

- Ark til mellomresultater

Von Neumann-arkitekturen

I dag:
Sentral prosesseringsenhet



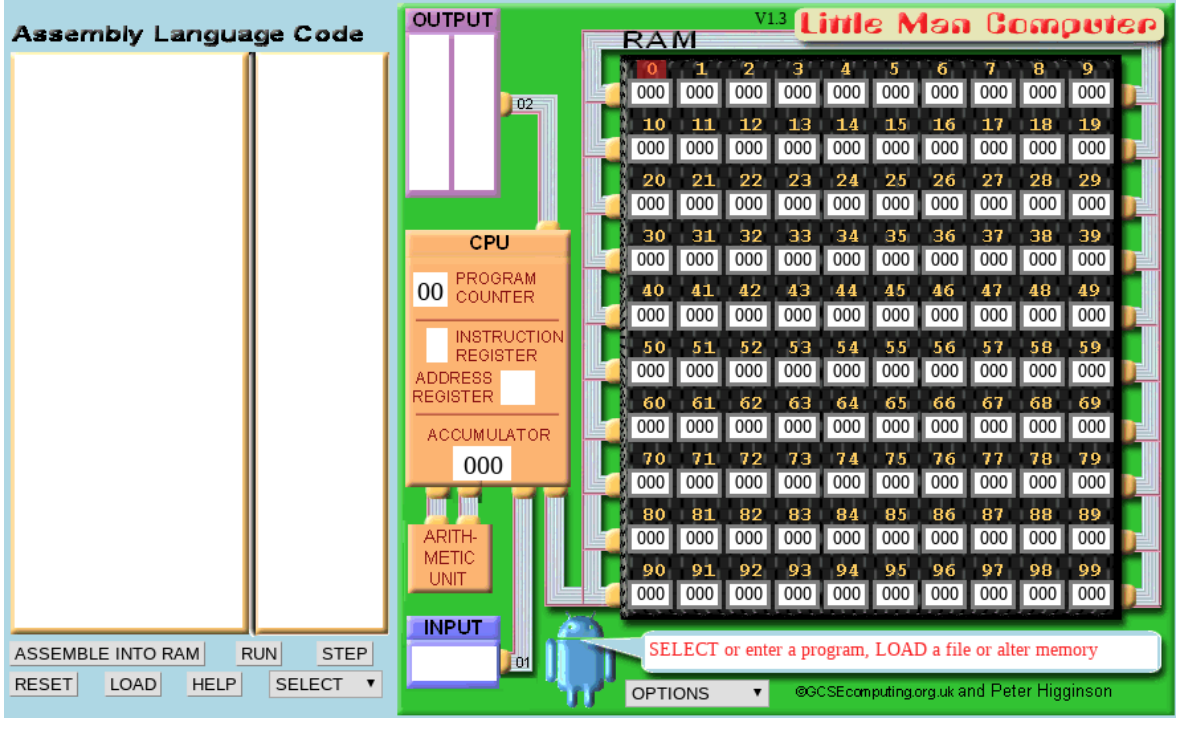
- Instruksjoner
  OG
- Beregningsdata

011001010101100....

- Tall
- Tekst
- Bilder
- Lyd

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/

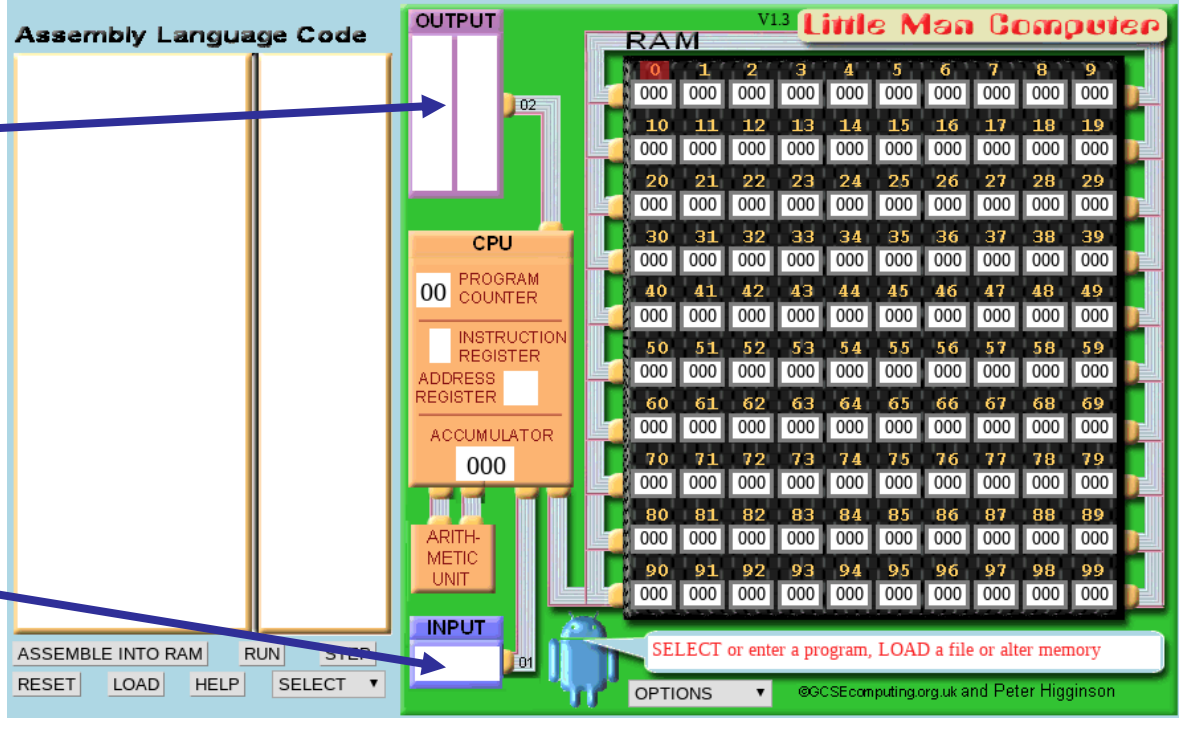
NB: LMC-simulatoren bruker desimaltall, og ikke binær/heksadesimal

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

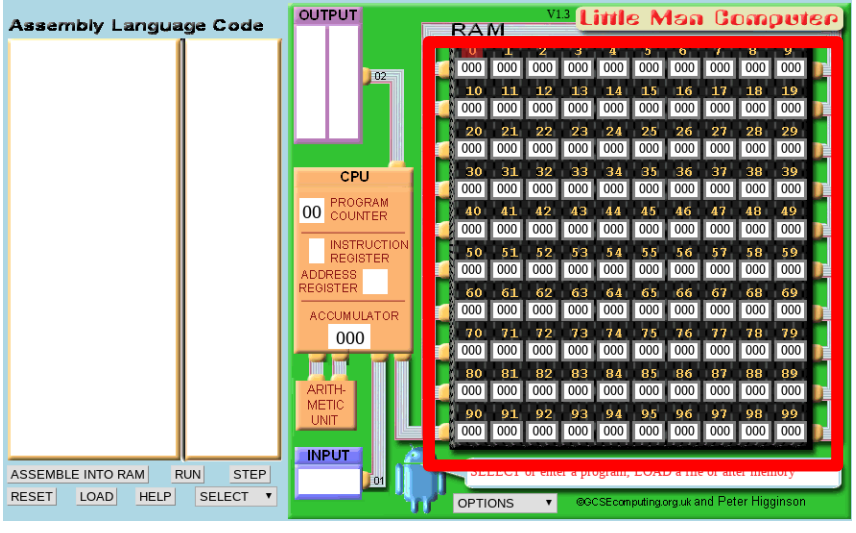
Data inn og ut

Resultatet kommer ut som tall som LMC skriver ut

LMC kan lese tall som brukeren (vi) skriver inn



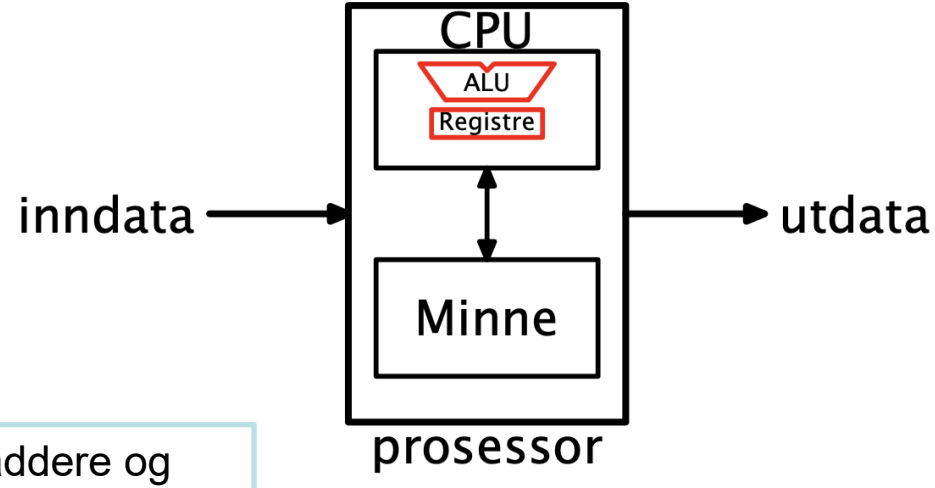
Minnet i LMC kan sees på som en posthylle nummerert fra 0-99. Hver «hylle» kan lagre ett tall.



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

CPU («Central Processing Unit») er den sentrale delen 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: 599** betyr *«Hent verdien i adresse **99** i minnet og legg den i akkumulatorregisteret.»*
  (Kode **5** angir at prosessoren skal hente noe i minnet.)

Eksekvering av programmer i LMC (Instruksjonssykelen)

Kontrollogikken i CPUen arbeider som følgende:

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 [står også i Englander-boka] vil

1. lese inn et tall og legge det i adresse 99 i minnet,
2. lese et tall til,
3. legge tallet i adresse 99 til 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 legg til tallet i celle 99.
OUT         // Skriv ut svaret.
HLT        // Stopp.
```

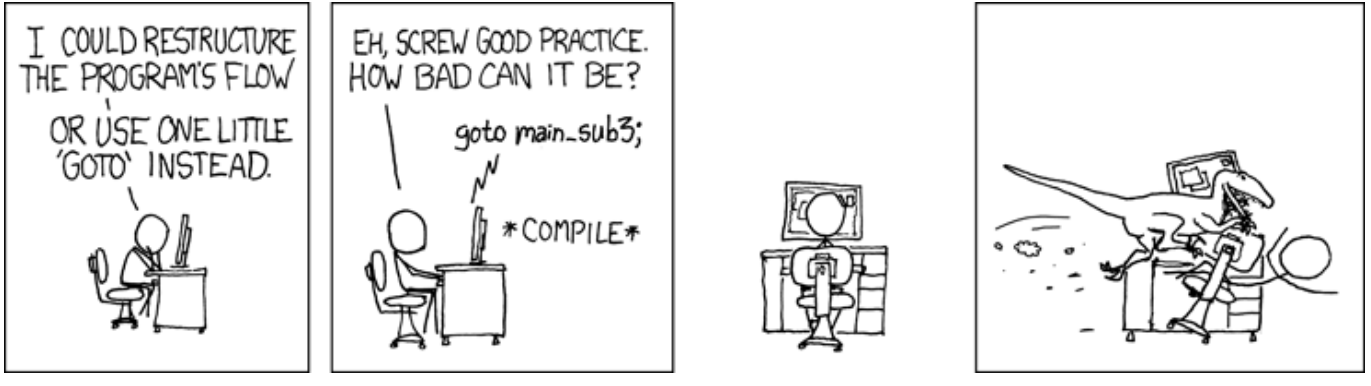
Denne typen kode kalles «assemblerkode».

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/