Gruppe 7 og 18 - IN1020
Uke 38: LMC

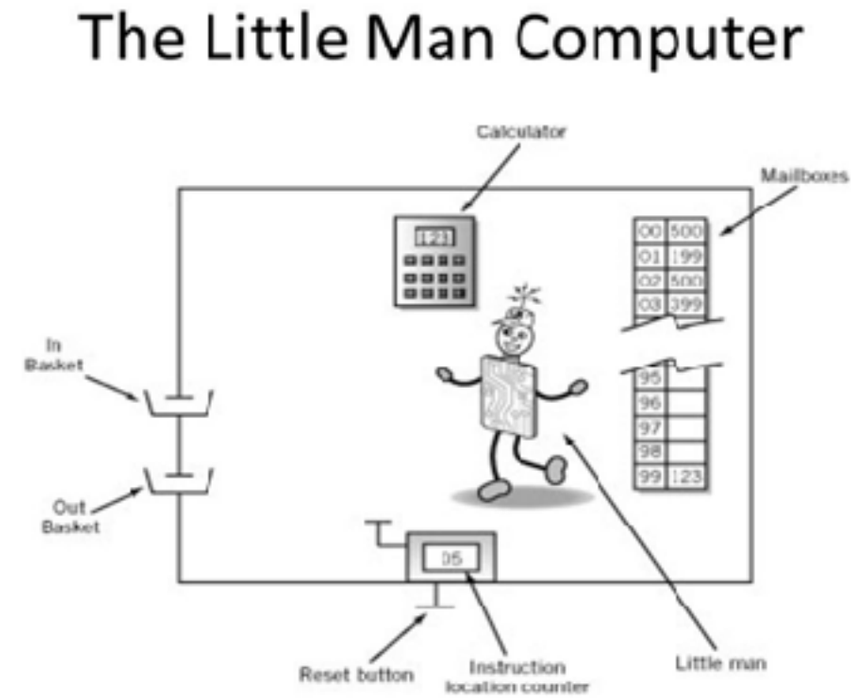
Jehan Khodabocus
jehank@uio.no

PLAN FOR TIMEN

- Info om oblig 2
- Kort repetisjon av pensum
- Eksempler av eksamensoppgave
- Ukeoppgave

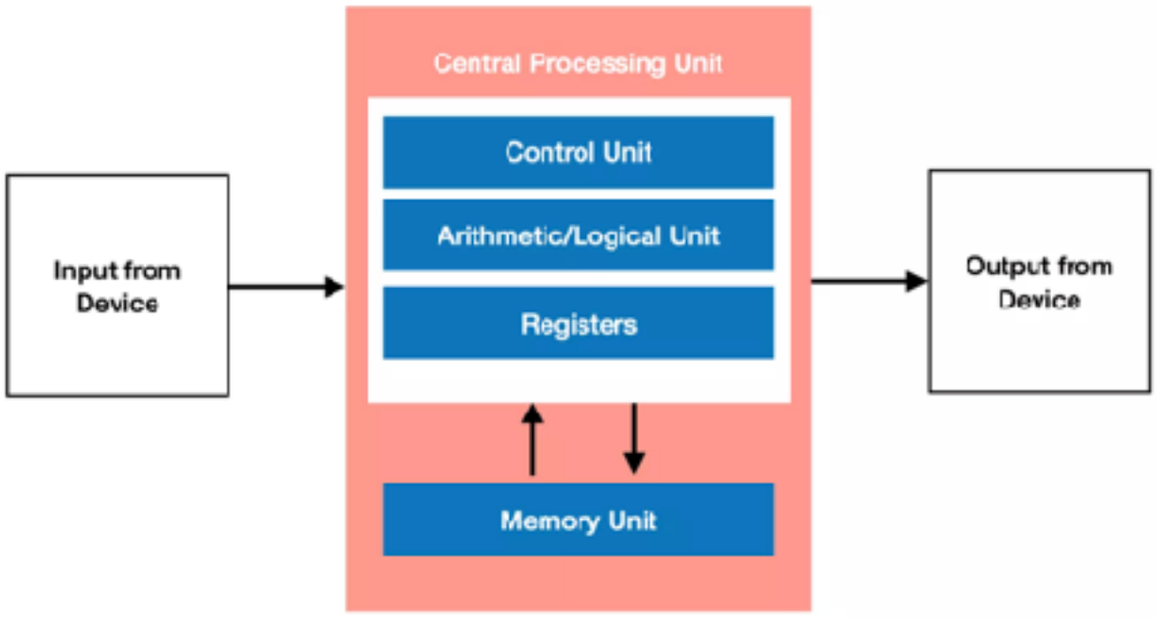
OBLIG 2

- Innleveringsfrist **4. oktober kl. 23:53**
- Skriv **navnet** og **brukernavnet** ditt i dokumentet du leverer.
- Oblig 2 består av **to** del:
  - 1) LMC : Lever som en **.txt** fil
  - 2) Tallsystem med base 30: Lever som en **.pdf** fil



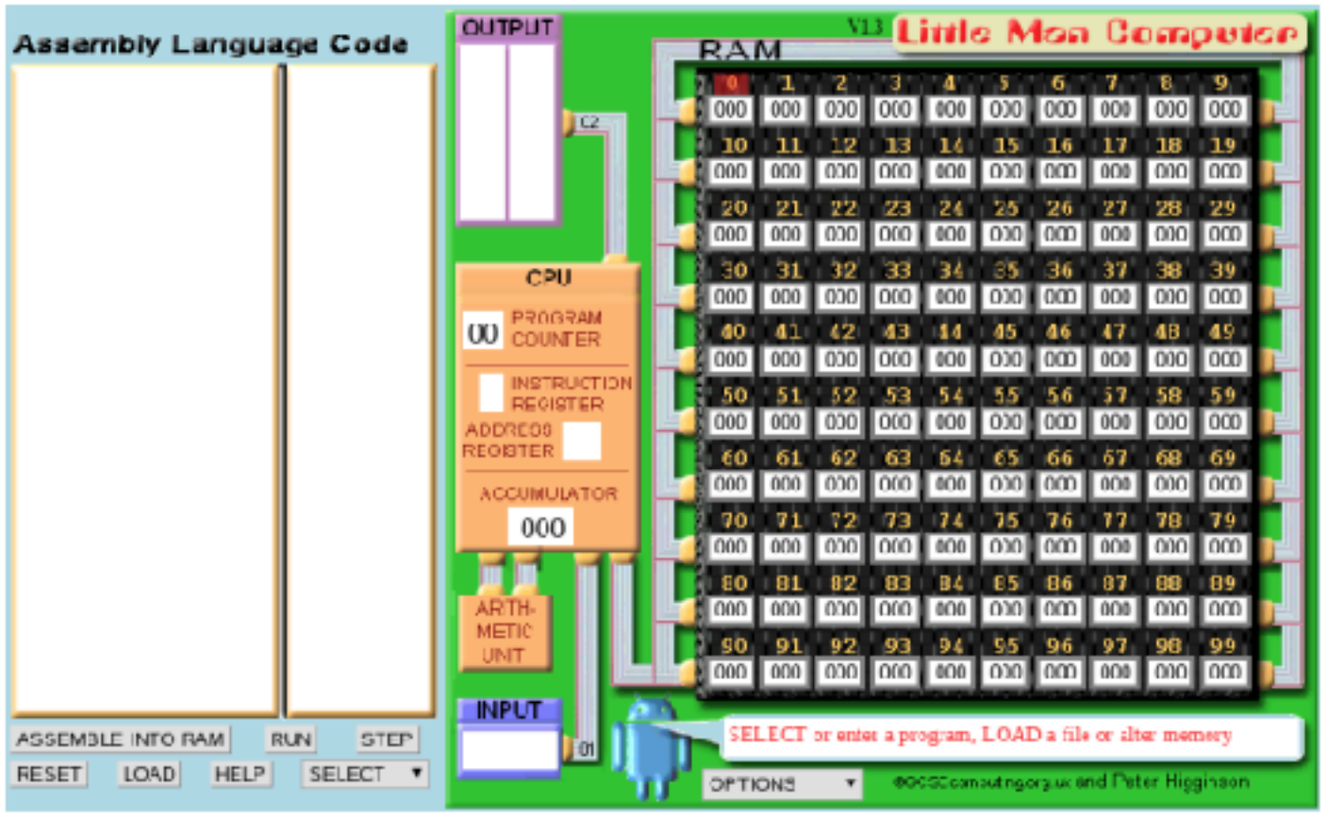
VON-NEUMANN ARKITEKTUR

- Von-Neumann arkitektur ble designet av John von Neumann i 1945.
- Det er en måte å designe datamaskiner hvor både data og instruksjoner ligger i samme minneenhet.
- Prosessoren er separat fra minneenheten.
- Arkitekturen skulle bestå av følgende deler:
  - **Minneenheten**: Består av RAM som er hovedminne som brukes til å lagre data og instruksjoner.
  - **Prosessor (CPU)**: En elektronisk krets som utfører instruksjoner på en datamaskin. Det er hjernen av datamaskinen som består av tre forskjellige komponenter.
    - **Aritmetisk logisk enhet(ALU)**: Delen av CPUen som utfører enkle aritmetiske og logiske operasjoner på data. Kan addere og subtrahere tall.
    - **Kontrollenheten (CU)**: Tolker instruksjonene i minnet og sørger for at de blir utført.
    - **Registre**: Små minneceller som CPUen trenger hurtig tilgang til. Registerene er i utgangspunktet midlertidige lagringssteder som lagrer adresser til instruksjonene som må utføres.
  - **Input-output (I/O) grensesnitt**: Lar brukerne kommunisere med maskinen.



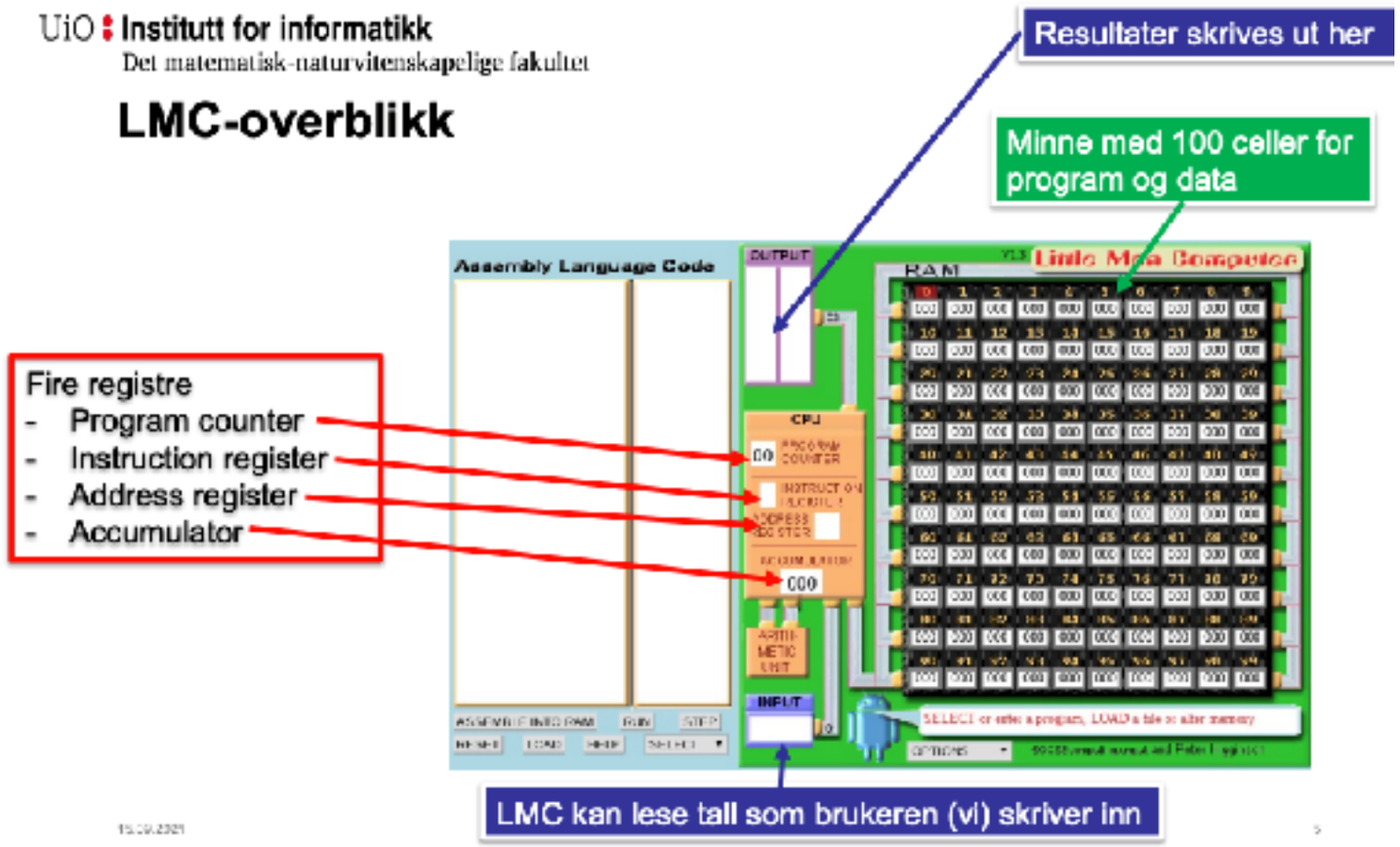
LITTLE MAN COMPUTER (LMC)

- En meget enkel, men teoretisk korrekt datamaskin
- **Input**: LMC kan lese tall som brukeren skriver inn
- **Output**: Resultatet som LMC skriver ut
- **Minnet** i LMC kan sees på om en posthylle fra 0 til 99 hvor hver "hylle" kan lagre ett tall.
- **Simulator**: http://peterhigginson.co.uk/LMC/



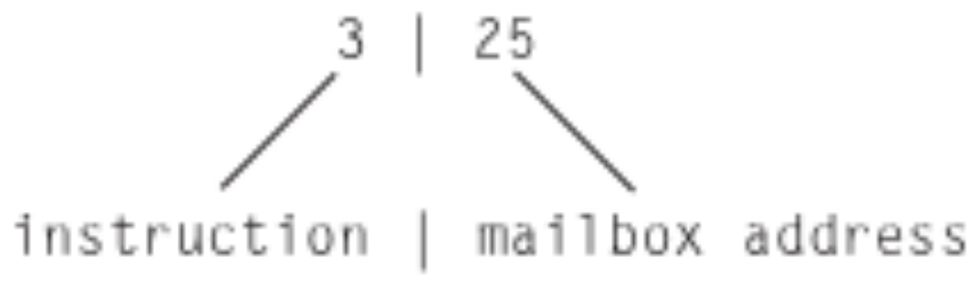
LMC HAR 4 REGISTRE

- **Program counter**: angir hvilken intruksjon 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. Henter data fra et område i minnet.
- **Accumulator**: til beregninger/operasjoner. Svaret kommer hit.



LMC INSTRUKSJONER

- Programmet lagres på numerisk form i minnet som en sekvens av instruksjoner.
- Hver instruksjon er et tall mellom 0 og 999 der
  - Første siffer angir instruksjonskoden
  - Andre og tredje siffer angir en adresse i minnet



- Hver instruksjon har en assembly mnemonic som tilsvarer en tresifret (desimal) instruksjon.

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)

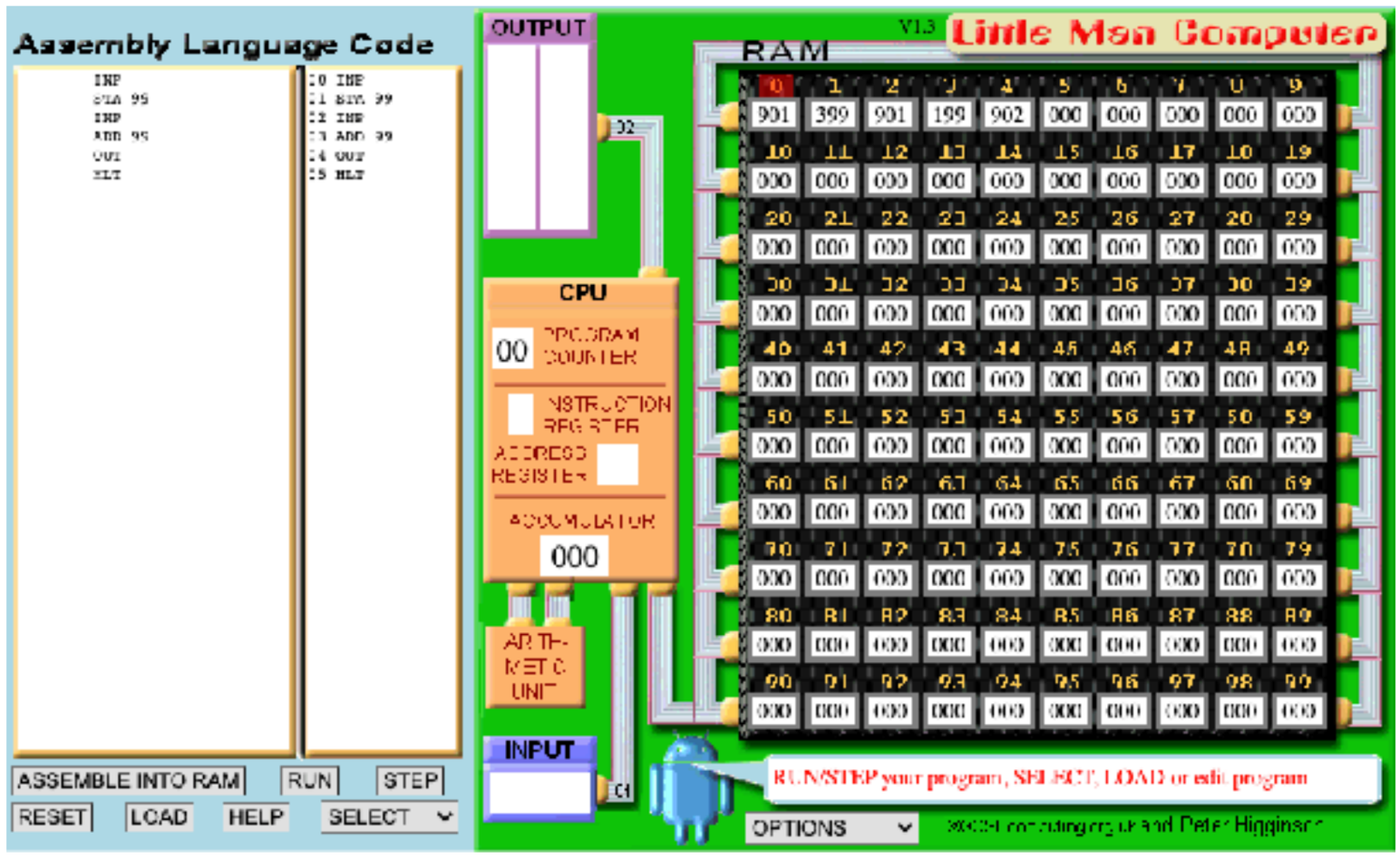
ASSEMBLER VS NUMERISK

```
00  INP         // Les et tall
01  STA 99     // Legg det i celle 99
02  INP         // Les et tall til
03  ADD 99   // Legg til tallet i celle 99
04  OUT        // Skriv ut svaret
05  HLT        // Stopp eksekvering
```

FIGURE 6.2

Program to Add Two Numbers

Mailbox code		Instruction description
00	901	INPUT
01	399	STORE DATA
02	901	INPUT 2ND #
03	199	ADD 1ST # TO IT
04	902	OUTPUT RESULT
05	000	STOP
99		DATA



ASCII

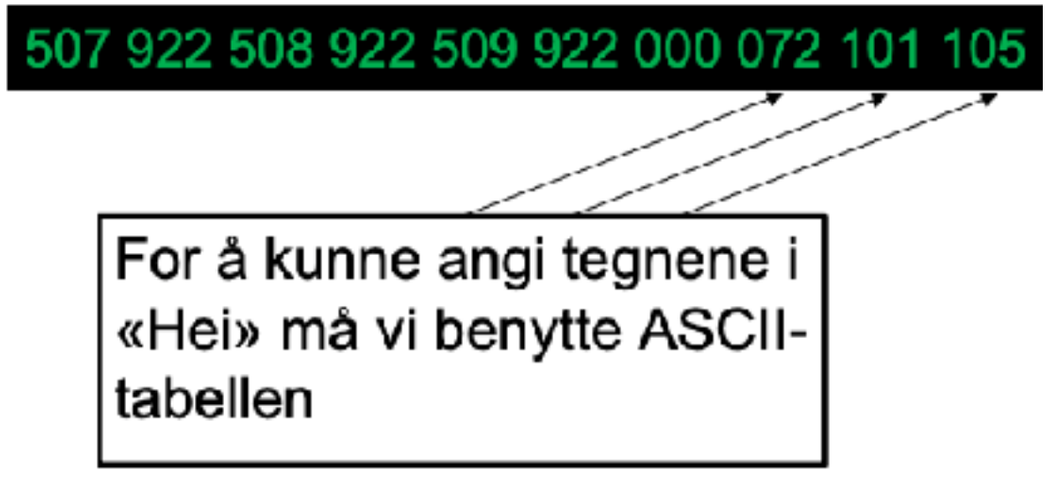
- ASCII (American Standard Code for Information Interchange): Første vellykkede forsøk på standardisering i 1963 med 128 tegn
- ASCII var vellykket
    - ASCII ble etter hvert brukt av de fleste
- Det er lett å sjekke om et tegn er et siffer eller en bokstav
- Det er lett å konvertere fra liten til stor bokstav
- Mangler Æ,Ø, Å og mange andre bokstaver og tegn

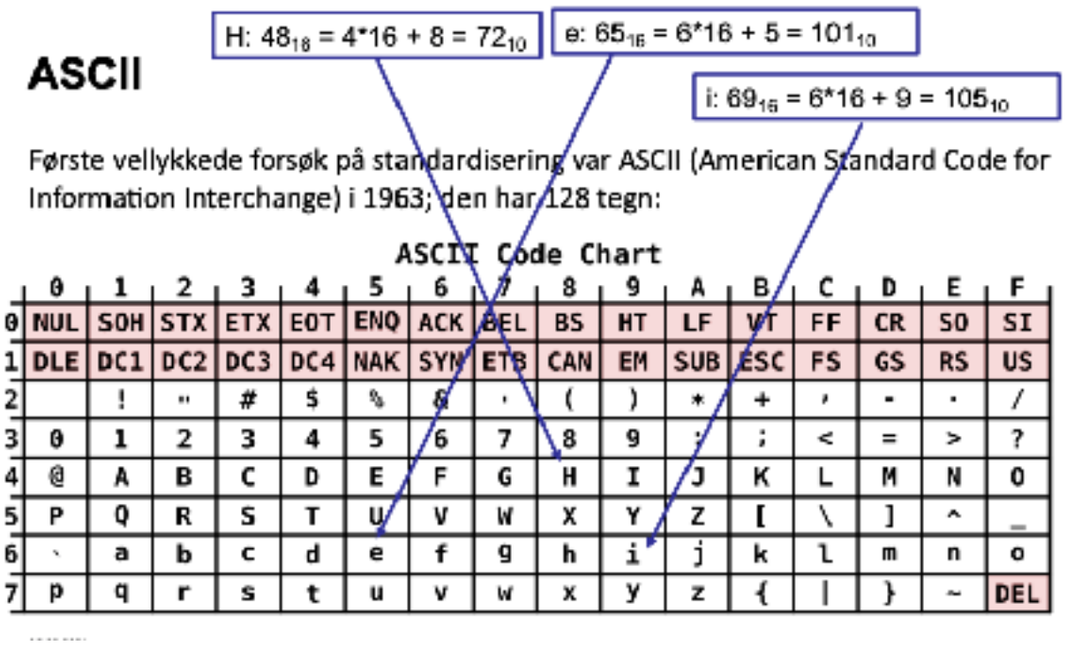
ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

NUMERISK KODE

- Her skriver programmet “Hei”. Hver av de tre tegnene hentes inn i akkumulatoren of skrives ut som tegn
- Alle programmer er lagret som tall når de kjøres, men:
  - Det er ikke så lett å huske alle tallkodene
  - Vanskelig både å skrive og lese programmer
  - Det er problematisk å holde oversikt hva som ligger i hvilke celler
  - Det er fort gjort å skrive et cellenummer feil
  - Om vi endrer programmet, må vi oftest endre veldig mange instruksjoner





	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

ASSEMBLERKODE

- Oversettes til numerisk kode før kjøring
- Mye enklere å lese og skrive for mennesker
- Kan også sette navn på celler i minnet, slik at det blir lettere å holde styr på data i minnet

Assemblerkode

Direkte oversettelse numerisk ←→ tekstlig representasjon

```
START   LDA   H
        OTC
        LDA   e
        OTC
        LDA   i
        OTC
END     HLT

H       DAT    72  // = 'H'
e       DAT   101  // = 'e'
i       DAT   105  // = 'i'
```

Spesifikasjonen **DAT** (= «data») er ingen instruksjon, men en angivelse av at det skal settes av en celle med en gitt startverdi.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	D	100	64	d
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	I	105	69	i
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	N	110	6E	n
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
19	13	*[DEVICE CONTROL 3]*	51	33	3	83	53	S	115	73	s
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[ENG OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
24	18	*[CANCEL]*	56	38	8	88	58	X	120	78	x
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

WHAT NEXT?

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

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)

LITEN OPPGAVE FRA BOKA

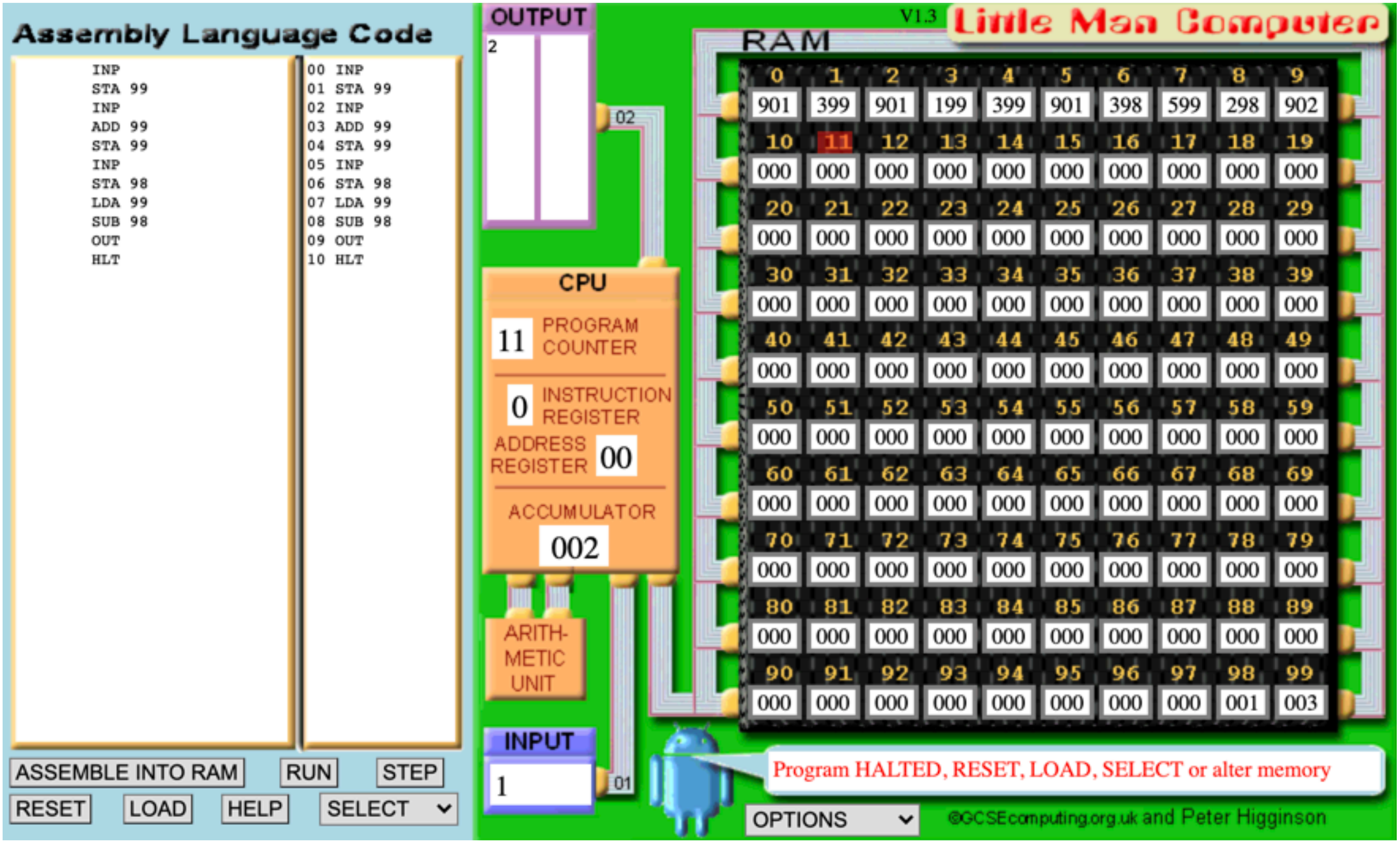
6.12 The following Little Man program is supposed to add two input numbers, subtract a third input number from the sum, and output the result, i.e.,

```
OUT = IN1 + IN2 -- IN3
```

mailbox	mnemonic code	numeric code
00	IN	901
01	STO 99	399
02	IN	901
03	ADD 99	199
04	STO 99	399
05	IN	901
06	SUB 99	299
07	OUT	902
08	COB	000

What is wrong with this program? Modify the program so that it produces the correct result.

SVAR


Assembly Language Code
INP
STA 99
INP
ADD 99
STA 99
INP
STA 98
LDA 99
SUB 98
OUT
HLT
00 INP
01 STA 99
02 INP
03 ADD 99
04 STA 99
05 INP
06 STA 98
07 LDA 99
08 SUB 98
09 OUT
10 HLT
ASSEMBLE INTO RAM
RUN
STEP
RESET
LOAD
HELP
SELECT
OUTPUT
2
02
CPU
11 PROGRAM COUNTER
0 INSTRUCTION REGISTER
ADDRESS REGISTER 00
ACCUMULATOR
002
ARITH-METIC UNIT
INPUT
1
01
V1.3
Little Man Computer
RAM
0 1 2 3 4 5 6 7 8 9
901 399 901 199 399 901 398 599 298 902
10 11 12 13 14 15 16 17 18 19
000 000 000 000 000 000 000 000 000 000
20 21 22 23 24 25 26 27 28 29
000 000 000 000 000 000 000 000 000 000
30 31 32 33 34 35 36 37 38 39
000 000 000 000 000 000 000 000 000 000
40 41 42 43 44 45 46 47 48 49
000 000 000 000 000 000 000 000 000 000
50 51 52 53 54 55 56 57 58 59
000 000 000 000 000 000 000 000 000 000
60 61 62 63 64 65 66 67 68 69
000 000 000 000 000 000 000 000 000 000
70 71 72 73 74 75 76 77 78 79
000 000 000 000 000 000 000 000 000 000
80 81 82 83 84 85 86 87 88 89
000 000 000 000 000 000 000 000 000 000
90 91 92 93 94 95 96 97 98 99
000 000 000 000 000 000 000 000 001 003
Program HALTED, RESET, LOAD, SELECT or alter memory
OPTIONS
©GCSEcomputing.org.uk and Peter Higginson

EKSAMENSOPPGAVE HØST 2018

2(c) **Maskinkode 1**

```
start   INP
        STA     50
        ADD     50
        ADD     50
        OUT
slutt   HLT
```

Hva skrives ut når denne koden kjøres og brukeren gir **7** som inndata?

Velg ett eller flere alternativer (men bare ett er korrekt):

- [ ] Det skrives ikke ut noe.
- [ ] 21
- [ ] 50
- [ ] 0
- [ ] 7

SVAR

2(c) **Maskinkode 1**

```
start   INP
        STA     50
        ADD     50
        ADD     50
        OUT
slutt   HLT
```

Hva skrives ut når denne koden kjøres og brukeren gir **7** som inndata?

Velg ett eller flere alternativer (men bare ett er korrekt):

- Det skrives ikke ut noe.
- 50
- 21 ✔
- 7
- 0

EKSAMENSOPPGAVE HØST 2018

2(d) **Maskinkode 2**

```
start   LDA     x
        BRZ     slutt
        SUB     y
        STA     x
        INP
        STA     w
        LDA     z
        SUB     w
        STA     z
        BRA     start
slutt   LDA     z
        OUT
        HLT

w       DAT     0
y       DAT     1
x       DAT     3
z       DAT     100
```

Hva skrives ut når denne koden kjøres og brukeren oppgir de tre verdiene **1**, **2** og **3** som inndata?

Velg ett eller flere alternativer (men bare ett er korrekt):

- [ ] 6
- [ ] Noe annet enn de andre alternativene
- [ ] 100
- [ ] 106
- [ ] 94

SVAR

2(d) **Maskinkode 2**

```
start   LDA     x
        BRZ     slutt
        SUB     y
        STA     x
        INP
        STA     w
        LDA     z
        SUB     w
        STA     z
        BRA     start
slutt   LDA     z
        OUT
        HLT

w       DAT     0
y       DAT     1
x       DAT     3
z       DAT     100
```

Hva skrives ut når denne koden kjøres og brukeren oppgir de tre verdiene **1**, **2** og **3** som inndata?

Velg ett eller flere alternativer (men bare ett er korrekt):

- 100
- 106
- 94 ✓
- 6
- Noe annet enn de andre alternativene

EKSAMENSOPPGAVE HØST 2018

2(e) Maskinkode 3

Anta at minnet inneholder disse verdiene:

0:	901
1:	805
2:	508
3:	902
4:	000
5:	108
6:	308
7:	600
8:	000

Anta også at *programtelleren* (også kalt «program counter» eller «location counter») inneholder **0**. Hva blir da skrevet ut når vi starter prosessoren vår og brukeren gir verdiene **10**, **1** og **-2** som inndata?

Velg ett eller flere alternativer (men **bare ett er korrekt**):

- [ ] 0
- [ ] -2
- [ ] Ingenting blir skrevet ut
- [ ] 11
- [ ] 2

Maks poeng: 4

SVAR

2(e) Maskinkode 3

Anta at minnet inneholder disse verdiene:

0:	901
1:	805
2:	508
3:	902
4:	000
5:	108
6:	308
7:	600
8:	000

Anta også at *programtelleren* (også kalt «program counter» eller «location counter») inneholder **0**. Hva blir da skrevet ut når vi starter prosessoren vår og brukeren gir verdiene **10**, **1** og **-2** som inndata?

Velg ett eller flere alternativer (men bare ett er korrekt):

- -2
- 0
- Ingenting blir skrevet ut
- 2
- 11 ✔

Maks poeng: 4

UKEOPPGAVE

- https://www.uio.no/studier/emner/matnat/ifi/IN1020/h21/ukeoppgaver/oppgaver-lmc-1.pdf

NESTE UKE

Programmering av prosessoren og Introduksjon til operativsystemer