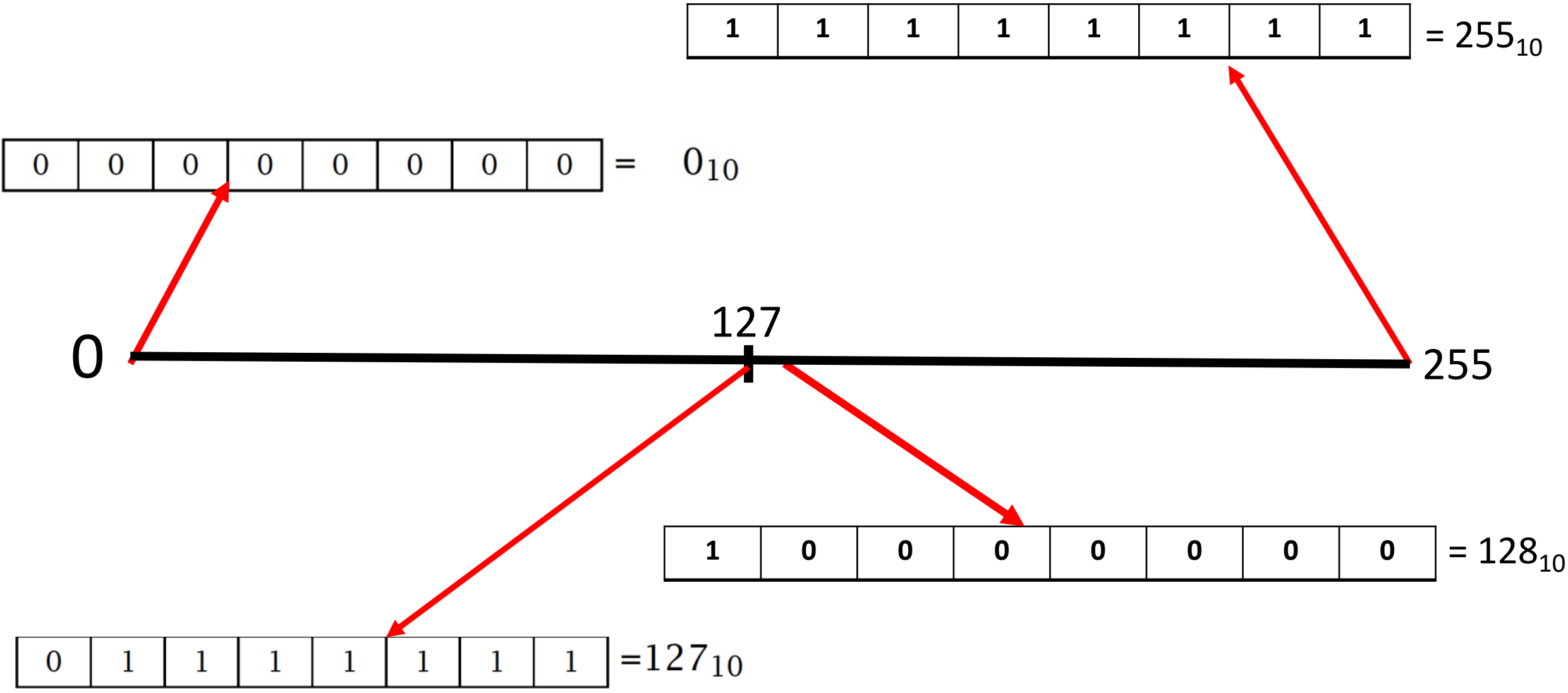
IN1020 uke 2

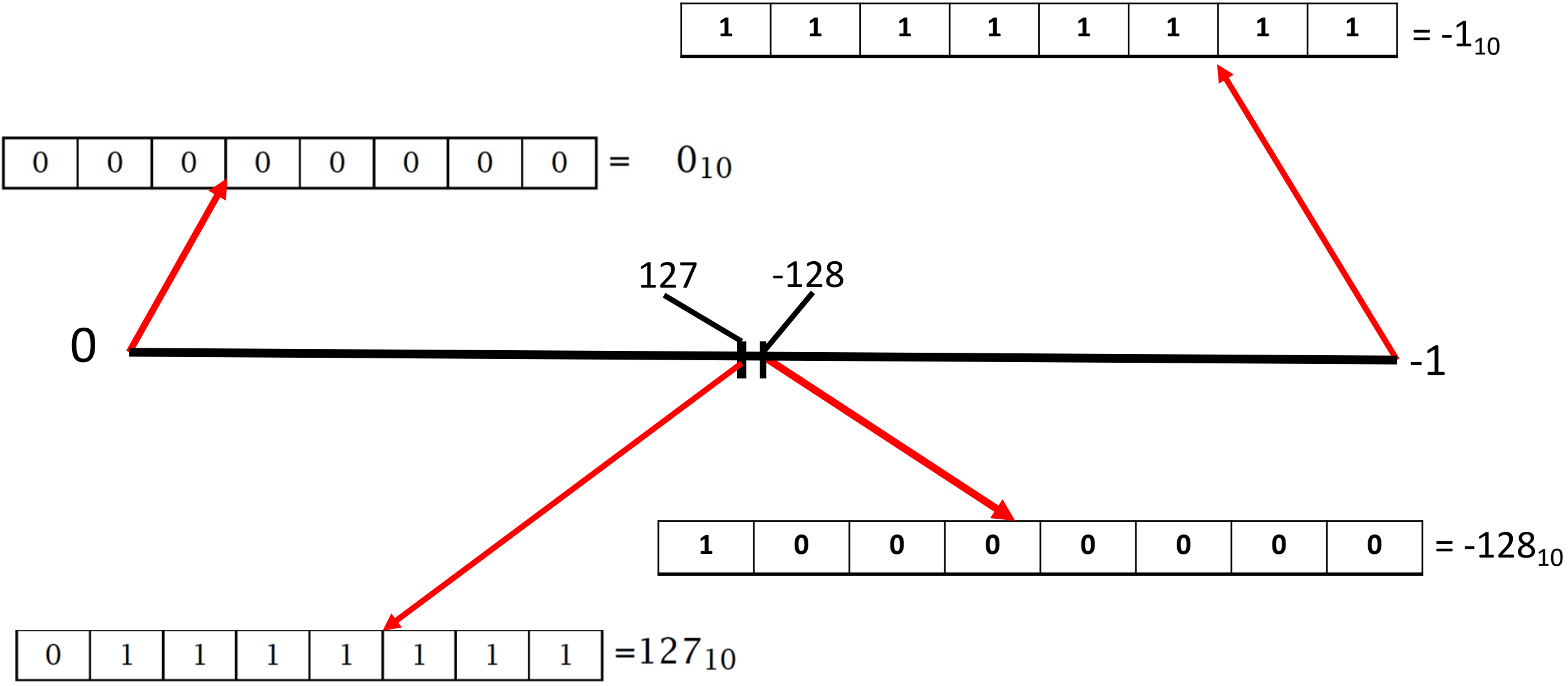
Plan for dagen

- Byte
- 2-er-komplement
- LMC
- Oblig jobbing eller ukesoppgaver

1 Byte = 8 bit


1 1 1 1 1 1 1 1 = 255₁₀
0 0 0 0 0 0 0 0 = 0₁₀
0
127
255
1 0 0 0 0 0 0 0 = 128₁₀
0 1 1 1 1 1 1 1 =127₁₀

2-er-komplement



Regning med to-er-komplement

- Fra binær til desimal

1	0	1	0	1	0	1	0

$-2^7 + 2^5 + 2^3 + 2^1$

$-128 + 32 + 8 + 2 = -86$

1	1	1	0	1	0	0	1

$-2^7 + 2^6 + 2^5 + 2^3 + 2^0$

$-128 + 64 + 32 + 8 + 1 = -23$

Fra decimal til binær

- F.eks gjør om –50 til binær

1. Bytte fortegn slik at -50 blir til 50
2. Finne binære tallet for 50
3. Inversere det binære tallet
4. Legge til 1 på slutten av binære tallet

= 00110010

inverserer ↓

= 11001101

= 11001101

\+ 1

= 11001110_2 = -50_{10}

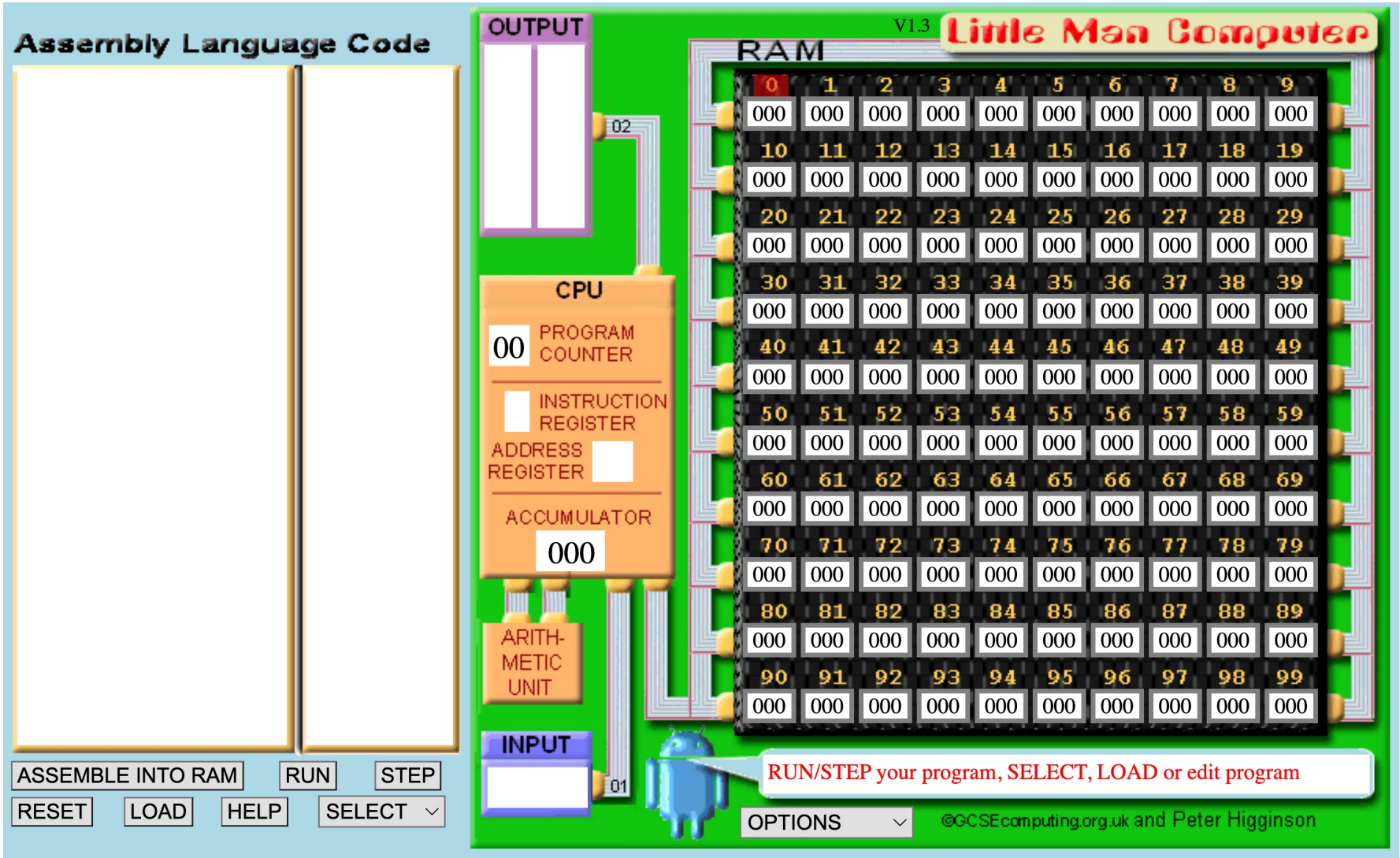
Verdi	Rest
50	0
25	1
12	0
6	0
3	1
1	1
0	

= 110010

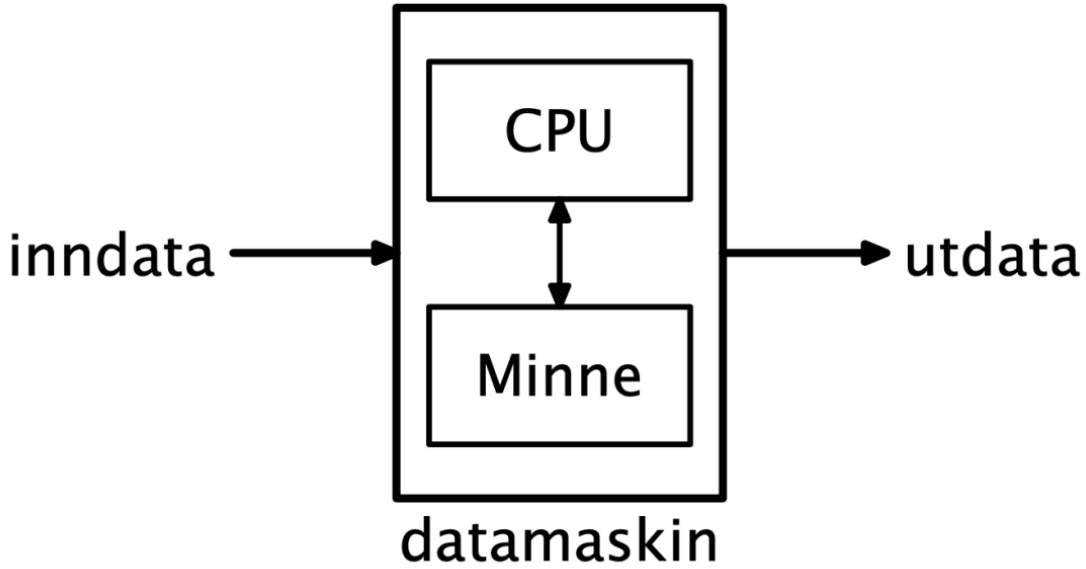
= 00110010

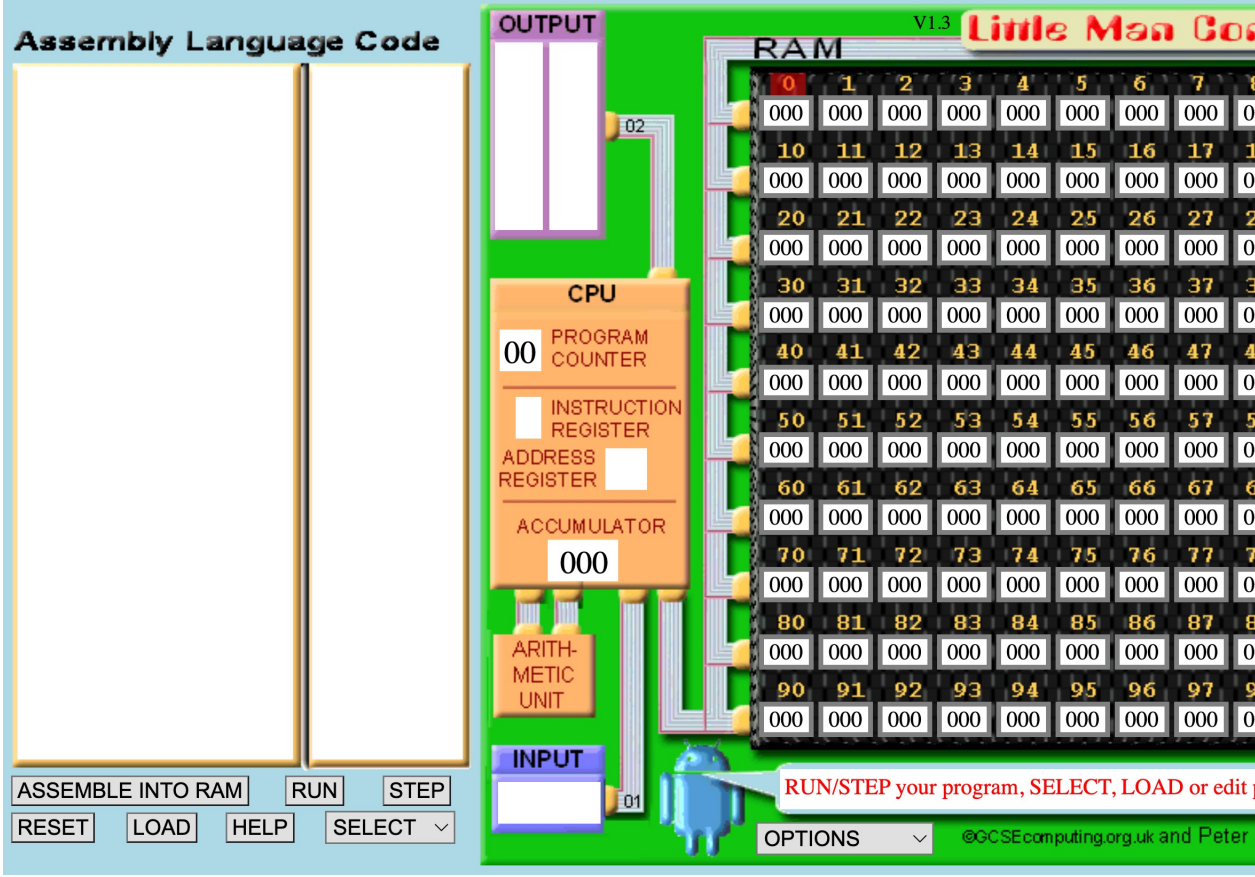
LMC – Little man computer

- Minne (RAM)
- OUTPUT
- INPUT
- CPU
- ALU (Arithmetic Logic Unit)
- Assembly code



Von Neumann-arkitekturen



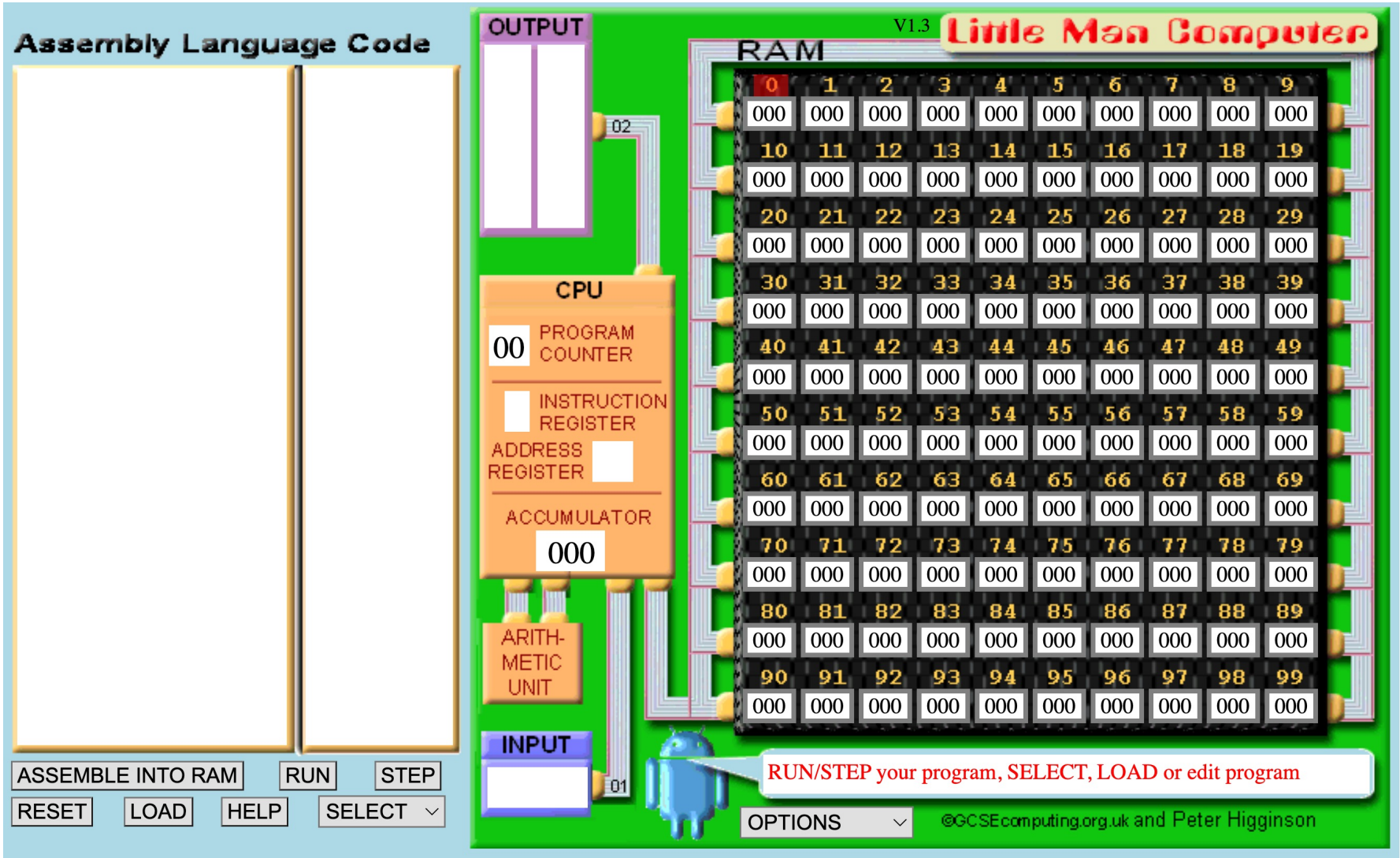


Den store erkjennelsen er at minnet holder på både *data* og *instruksjoner* til CPU-en (altså *programmet*)

Minne

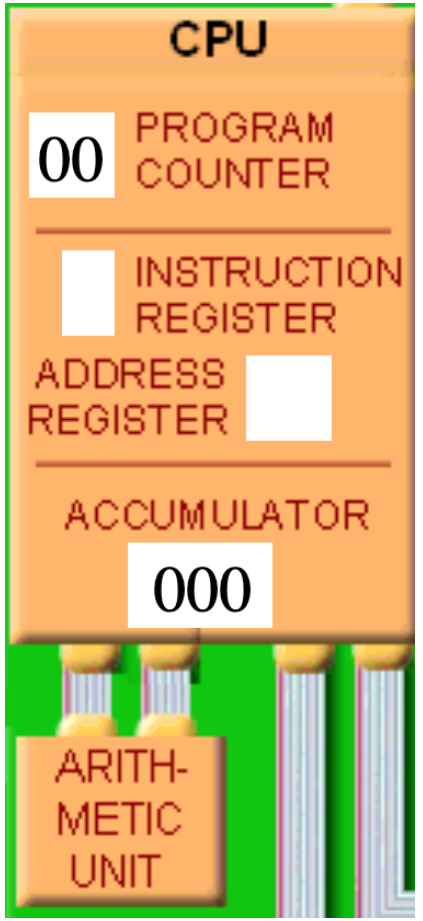
- 99 hyller
- Hylle inneholder:
  - Instruks
  - Data

- Minne = Oppskrift
- CPU = Den som bruker oppskriften



CPU – Central processing Unit

- Inneholder ALU (Arithmetic Logic Unit):
  - ALU kan utføre enkle operasjoner som pluss og minus
- Register:
  - Program Counter:
  - Instruction Register
  - Address Register
  - Accumulator
- Kontrollogikk til å utføre ting i riktig ordre



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)

Instruksjon: DAT

Pseudoinstruksjonen DAT

- Hvordan sier vi:
  - «Adresse 6 brukes til DATA»
- **DAT <verdi>**
  - Deklarerer en variabel i minnet
  - Initialiseres med **verdi**

ADD

Addresse	Instruksjon	Adresse del
0	INP	
1	STA	6
2	INP	
3	ADD	6
4	OUT	
5	HLT	
6	DAT	0

Addresse	Instruksjon	Adresse del
0	INP	
1	STA	7
2	INP	
3	ADD	7
4	ADD	7
5	OUT	
6	HLT	
7	DAT	0

Labels

Bruk av etiketter «labels»

- En **etikett** «holder» adressen til en assembler instruksjon
  - Feks. **tall** til høyre ☺
- Bruk etiketten i adressedelen
- Assembleren finner ut hvor i minnet «ting» havner

Etikett ☺ ☺	Instruksjon	Adresse del
	INP	
	STA	tall
	INP	
	ADD	tall
	ADD	tall
	OUT	
	HLT	
tall	DAT	0

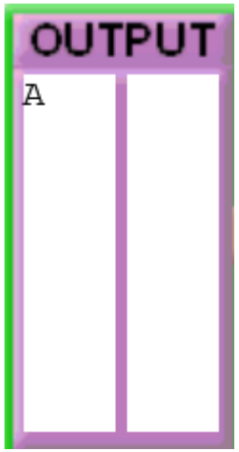
Hvordan printe streng (tekst)

- Bruker:

922	OTC	Skriver ut ASCII-tegn (ikke i boka)

  - **OTC** virker litt som OUT..
    - Men i stedet for å skrive ut akkumulatoren..
    - ..så skriver den ut akkumulatoren som et ASCII tegn!

  - Eksempel: Bokstaven A kodes som 65_{10}
    - **LDA** <adresse med tallet 65>
    - **OTC**
    - Gir da ----------→



ASCII

ASCII

H: $48_{16} = 4*16 + 8 = 72_{10}$

e: $65_{16} = 6*16 + 5 = 101_{10}$

i: $69_{16} = 6*16 + 9 = 105_{10}$

Første vellykkede forsøk på standardisering var ASCII (American Standard Code for Information Interchange) i 1963; den har 128 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

Binary/Decimal/Hexadecimal/ASCII Character Conversion Chart

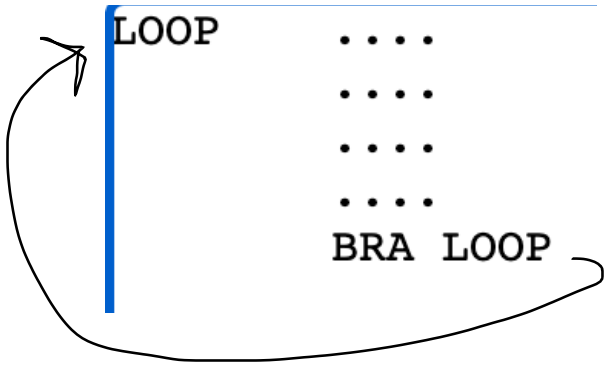
bin	dec	hex	char
0	0	0	NUL
1	1	1	STX
10	2	2	SOT
11	3	3	ETX
100	4	4	EOT
101	5	5	ENQ
110	6	6	ACK
111	7	7	BEL
1000	8	8	BS
1001	9	9	HT
1010	10	A	LF
1011	11	B	VT
1100	12	C	FF
1101	13	D	CR
1110	14	E	SO
1111	15	F	SI
10000	16	10	DLE
10001	17	11	DC1
10010	18	12	DC2
10011	19	13	DC3
10100	20	14	DC4
10101	21	15	NAK
10110	22	16	SYN
10111	23	17	ETB
11000	24	18	CAN
11001	25	19	EM
11010	26	1A	SUB
11011	27	1B	ESC
11100	28	1C	FS
11101	29	1D	GS
11110	30	1E	RS
11111	31	1F	US
100000	32	20	SP
100001	33	21	!
100010	34	22	"
100011	35	23	#
100100	36	24	$
100101	37	25	%
100110	38	26	&
100111	39	27	'
101000	40	28	(
101001	41	29	)
101010	42	2A	*

bin	dec	hex	char
101011	43	2B	+
101100	44	2C	,
101101	45	2D	-
101110	46	2E	.
101111	47	2F	/
110000	48	30	0
110001	49	31	1
110010	50	32	2
110011	51	33	3
110100	52	34	4
110101	53	35	5
110110	54	36	6
110111	55	37	7
111000	56	38	8
111001	57	39	9
111010	58	3A	:
111011	59	3B	;
111100	60	3C	<
111101	61	3D	=
111110	62	3E	>
111111	63	3F	?
1000000	64	40	@
1000001	65	41	A
1000010	66	42	B
1000011	67	43	C
1000100	68	44	D
1000101	69	45	E
1000110	70	46	F
1000111	71	47	G
1001000	72	48	H
1001001	73	49	I
1001010	74	4A	J
1001011	75	4B	K
1001100	76	4C	L
1001101	77	4D	M
1001110	78	4E	N
1001111	79	4F	O
1010000	80	50	P
1010001	81	51	Q
1010010	82	52	R
1010011	83	53	S
1010100	84	54	T
1010101	85	55	U

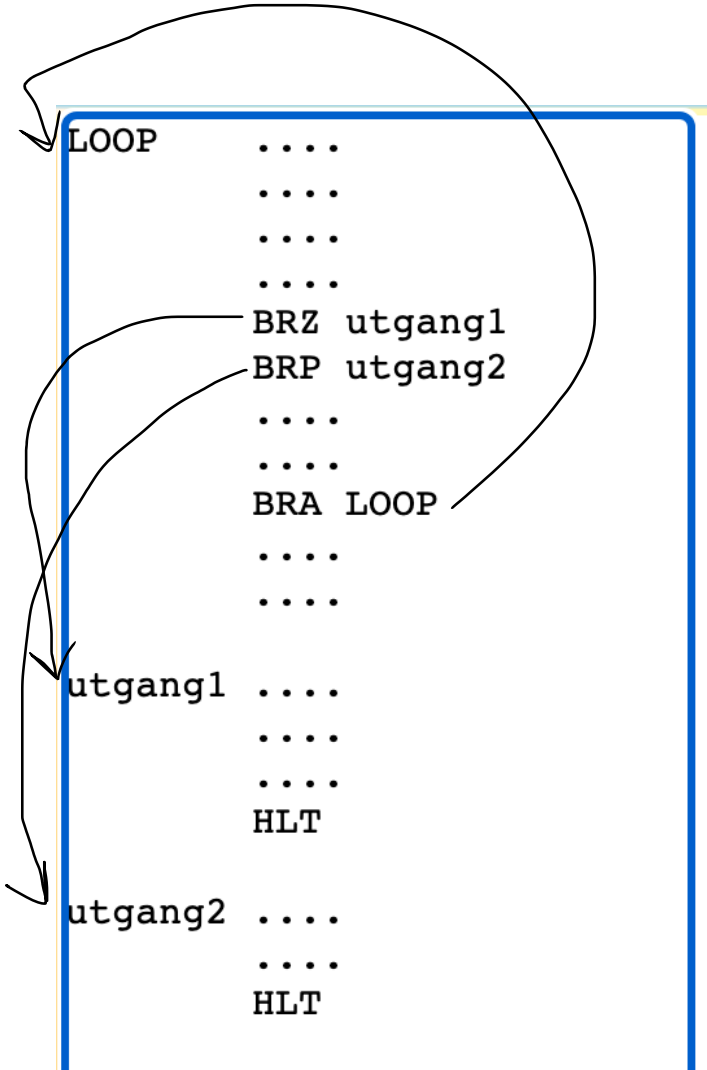
bin	dec	hex	char
1010110	86	56	V
1010111	87	57	W
1011000	88	58	X
1011001	89	59	Y
1011010	90	5A	Z
1011011	91	5B	[
1011100	92	5C	\
1011101	93	5D	]
1011110	94	5E	^
1011111	95	5F	_
1100000	96	60	`
1100001	97	61	a
1100010	98	62	b
1100011	99	63	c
1100100	100	64	d
1100101	101	65	e
1100110	102	66	f
1100111	103	67	g
1101000	104	68	h
1101001	105	69	i
1101010	106	6A	j
1101011	107	6B	k
1101100	108	6C	l
1101101	109	6D	m
1101110	110	6E	n
1101111	111	6F	o
1110000	112	70	p
1110001	113	71	q
1110010	114	72	r
1110011	115	73	s
1110100	116	74	t
1110101	117	75	u
1110110	118	76	v
1110111	119	77	w
1111000	120	78	x
1111001	121	79	y
1111010	122	7A	z
1111011	123	7B	{
1111100	124	7C	|
1111101	125	7D	}
1111110	126	7E	~
1111111	127	7F	DEL

[ASCII chart]

Løkker og hopping



- BRA (Branch):
  - Hopper ALLTID til label
- BRZ (Branch if zero):
  - Hopper til label HVIS akkumulatoren er null
- BRP (Branch if positive):
  - Hopper til label HVIS akkumulatoren er null eller positiv



Spørsmål

- Jobbe videre med oblig eller ukesoppgaver