Gruppetime 3 i IN1020

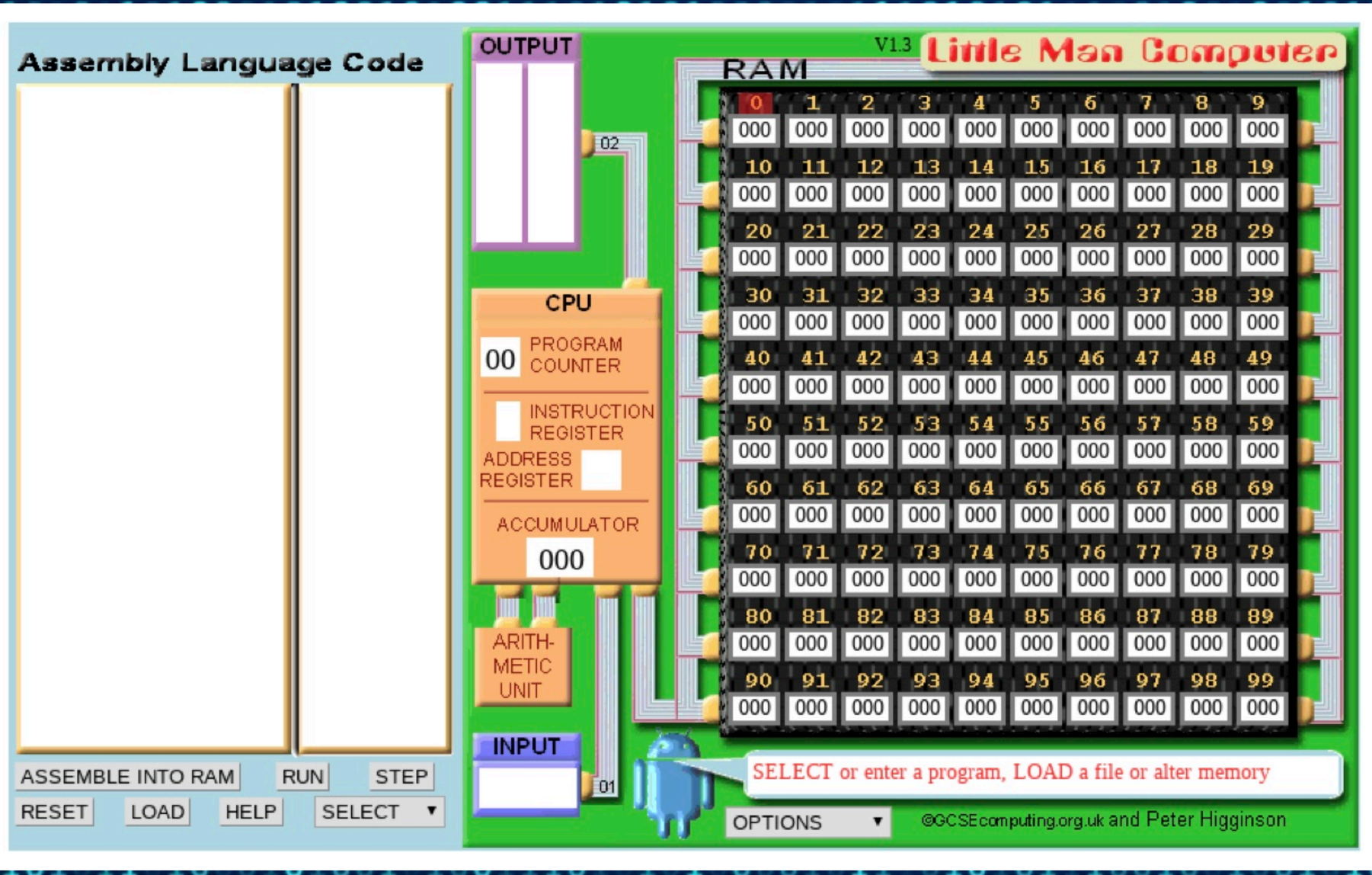
Cristina Tezec (cristite@ifi.uio.no)

Plan for dagens gruppetime

> LMC
> ukesoppgaver
> kort oppsummering av oblig 1
> Kahoot!

Little Man Computer

> CPU består av ALU og registre
>> ALU gjør aritmetiske beregninger (addere og subtrahere)
>> Registre :
>>> Accumulator (tar varet på svaret fra beregninger)
>>> Program counter (angir hvilken instruksjon i minne skal utføres)
>>> Instruction register (inneholder koden til den instruksjonen som utføres)
>>> Address register (inneholder addressedelen av den instruksjonen som utføres)



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)

Instruksjonen HLT (HALT eller COB for «Coffe Break») har kode 000 og stopper kjøringen.

Instruksjonen INP for input og OUT for output starter med siffer 9:

>> 901 for Input

>> 902 for Output

>> 905 for Subrutine

Instruksjonen STA («Store Accumulator») har kode 3xx, og den kopierer verdien i akkumulatoren til adresse xx i minnet:

>> eks: 374

>> eks: 374

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

>> eks: 532

>> eks: 235

Instruksjonen ADD har kode 1xx og brukes for addisjon.

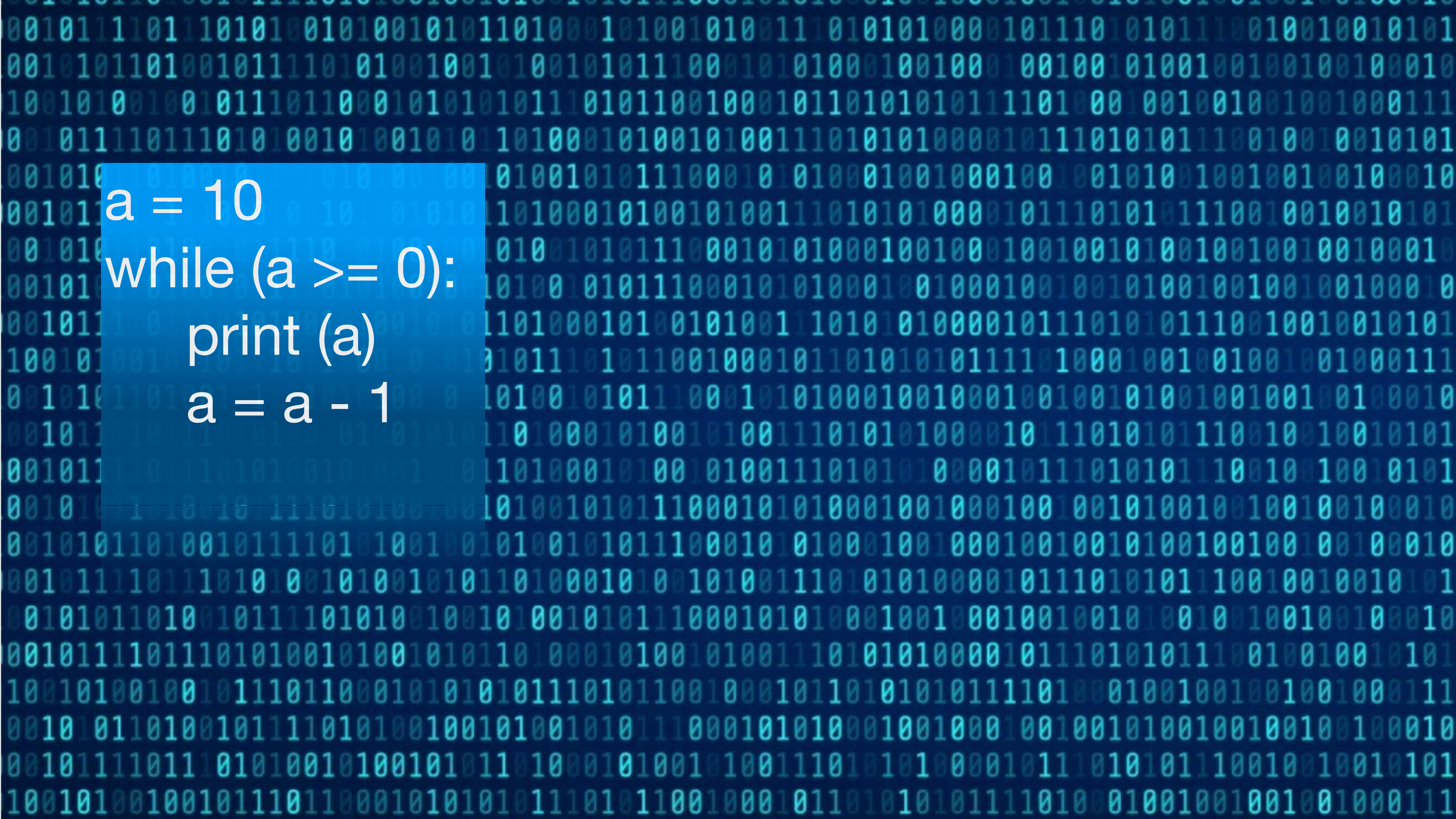
Instruksjonen SUB har kode 2xx og brukes for subtraksjon.

Hvordan lager vi løkker i LMC?

>> Instruksjonen BRA («Branch») har kode 6xx og brukes for å hoppe til angitt adresse.

>> Instruksjonen BRZ («Branch if zero») har kode 7xx - hopper til angitt adresse hvis akkumulatoren er 0.

>> Instruksjonen BRP («Branch if positive») har kode 8xx - hopper til angitt adresse hvis akkumulatoren er >= 0.


a = 10
while (a >= 0):
print (a)
a = a - 1

```
a = 10
while (a >= 0):
    print (a)
    a = a - 1
```

```
        LDA     v10     // Start med 10
TELL    OUT             // Skriv ut
        SUB     v1      // og tell ned.
        BRP     TELL    // Gjenta om akk. er >= 0
        HLT             // Stopp

v1      DAT     1       // Konstanten 1
v10     DAT     10      // Konstanten 10
```

For å skrive ord i LMC, bruker vi ASCII og kommando OTC (for «Output accumulator as a character»)

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

Pseudoinstruksjonen DAT deklarer en variabel i minnet. Bruk etiketter istedenfor tall, dvs variabelnavn:

>> eks. «tall | DAT» og ikke «7 | DAT»

Bruk gjerne kommentarer i koden din! Gjør det lettere for andre å forstå koden din.

```
>> // dette er en kommentar i LMC
```