

UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

Oppsummering, repetisjon

Eyvind W. Axelsen (eyvinda@ifi.uio.no)

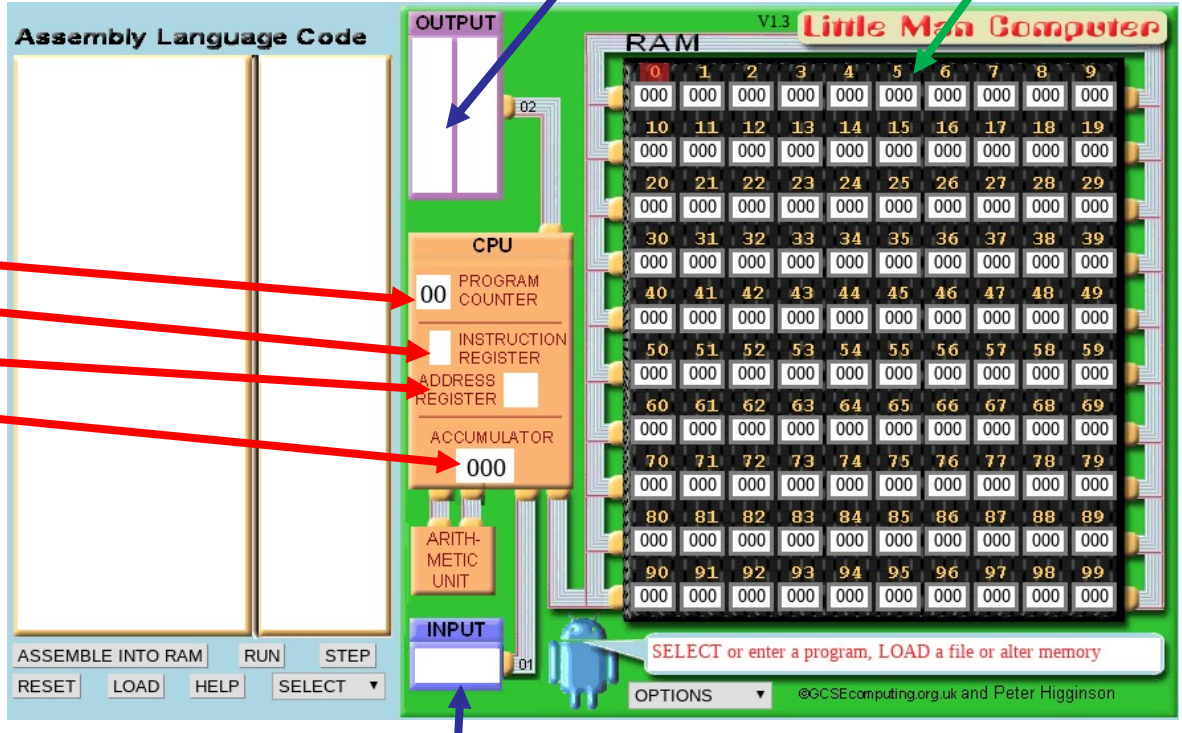
- LMC
- Binær representasjon

LMC-overblikk

Resultater skrives ut her

Minne med 100 celler for program og data

Fire registre
- Programteller
- Instruksjonsregister
- Addresseregister
- Akkumulator



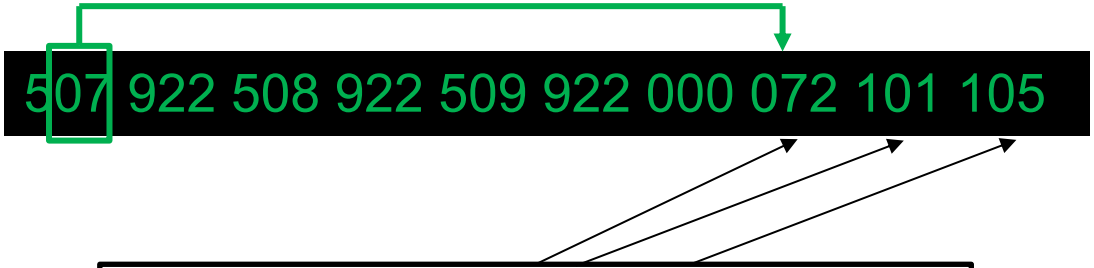
LMC kan lese tall som brukeren (vi) skriver inn

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 fra akkumulatoren
3xx	STO	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 til angitt adresse hvis akkumulatoren er 0
8xx	BRP	Hopper til angitt adresse 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)

Eksempel: å skrive ut en streng

Dette programmet skriver **Hei**. Hvert av de tre tegnene hentes inn i akkumulatoren og skrives ut som tegn:



For å kunne angi tegnene i «Hei» må vi benytte ASCII-tabellen

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

$0x48 = 72_{10}$

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	STO	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 til angitt adresse hvis akkumulatoren er 0
8xx	BRP	Hopper til angitt adresse 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)

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.

Eksempel: Multiplikasjon

Vi ønsker å multiplisere to tall.

1. Har vi en instruksjon som gjør det?
2. Kan vi oppnå samme resultat med én eller flere andre instruksjoner?

Vi vet at

```
a * b = b + b + ... + b,
```

a ganger.

Da kan vi lage en *løkke* med addisjoner, siden LMC har instruksjoner for dette

Akk	w	y	x	z
0	0	1	3	100

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?

Akk	w	y	x	z
3	0	1	3	100

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
```

Vi laster x (3) inn i akkumulator-registeret

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

Akk	w	y	x	z
3	0	1	3	100

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
```

Hopp dersom akk er 0

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

Akk	w	y	x	z
2	0	1	3	100

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
```

Vi trekker y (1) fra akkumulatoren

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

Akk	w	y	x	z
2	0	1	2	100

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
```

Og lagrer resultatet i x

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

Akk	w	y	x	z
1	0	1	2	100

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
```

Leser input fra bruker (1)

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

Akk	w	y	x	z
1	1	1	2	100

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
```

Lagrer input (akkumulatoren) i w

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

Akk	w	y	x	z
100	1	1	2	100

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
```

Laster verdi av z inn i akkumulatoren

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

Akk	w	y	x	z
99	1	1	2	100

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
```

Trekker verdien av w (1) fra akkumulatoren

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

Akk	w	y	x	z
99	1	1	2	99

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
```

Lagrer akkumulatorens verdi i z

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

Akk	w	y	x	z
99	1	1	2	99

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
```

Hopper opp til start-merket igjen

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

Instruksjoner og data

Hva er problemet med dette programmet?

```
        INP             // Les inn
        STA     x       // verdien x.
        ADD     x       // Doble den.
        OUT             // Skriv den ut.

x       DAT     0       // Variabel x
```

Det mangler en HLT-instruksjon. Hva skjer da når vi leser inn verdien **600**? Hva med **101**?

Hvordan vet LMC om en celle inneholder en instruksjon eller en dataverdi?

Det vet den ikke!

Hvordan fungerer tallsystemer egentlig?

- Moderne tallsystemer er posisjonsbaserte, og sifrene har vekt i henhold til posisjonen.

- I vårt vanlige desimale tallsystem har posisjonene vekt
  1 10 100 1000 osv.
  Dette tilsvarer
  10^0 10^1 10^2 10^3 osv.

(Notasjonen a^n («a i n-te») betyr a * a * ... * a, n ganger.)

Hvordan fungerer tallsystemer egentlig?

- I det binære tallsystemet har posisjonen tilsvarende vekt
  1 2 4 8 osv.
  Dette tilsvarer
  2^0 2^1 2^2 2^3 osv.

Binær representasjon: 1 byte er 8 bit

I en datamaskin kan vi ikke hente ut 1 og 1 bit; i stedet lagrer vi dem som byte (dvs grupper av 8 bit):

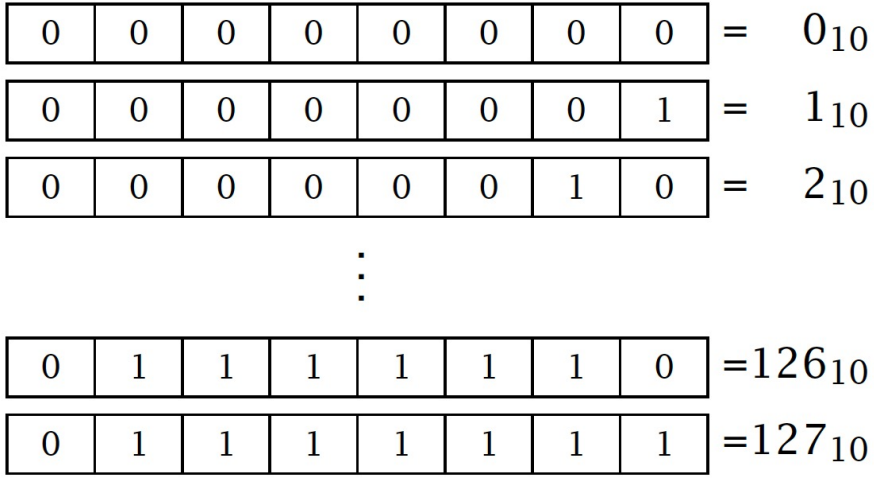
0	0	0	0	0	0	0	0	$= 0_{10}$
0	0	0	0	0	0	0	1	$= 1_{10}$
0	0	0	0	0	0	1	0	$= 2_{10}$
⋮								
1	1	1	1	1	1	1	0	$= 254_{10}$
1	1	1	1	1	1	1	1	$= 255_{10} = 2^8 - 1$

Vi kan også lagre tall i 2, 4 eller 8 byte satt sammen.

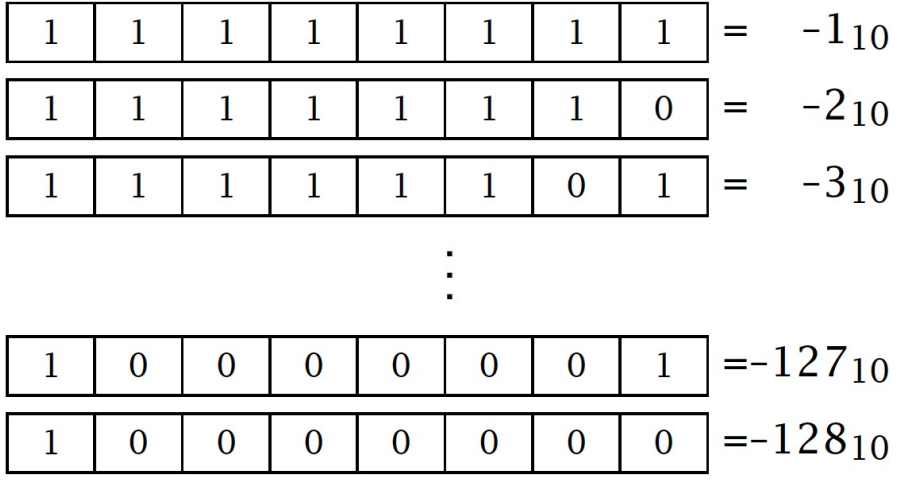
Tall med fortegn

Øverste bit (gjerne kalt fortegnsbit) tolkes som det negative av den verdien det ellers ville hatt.

Positive tall



Negative tall



Det største positive tallet blir da +127, og det største negative er -128 + 0
Det første negative tallet er -128 + 127 = -1

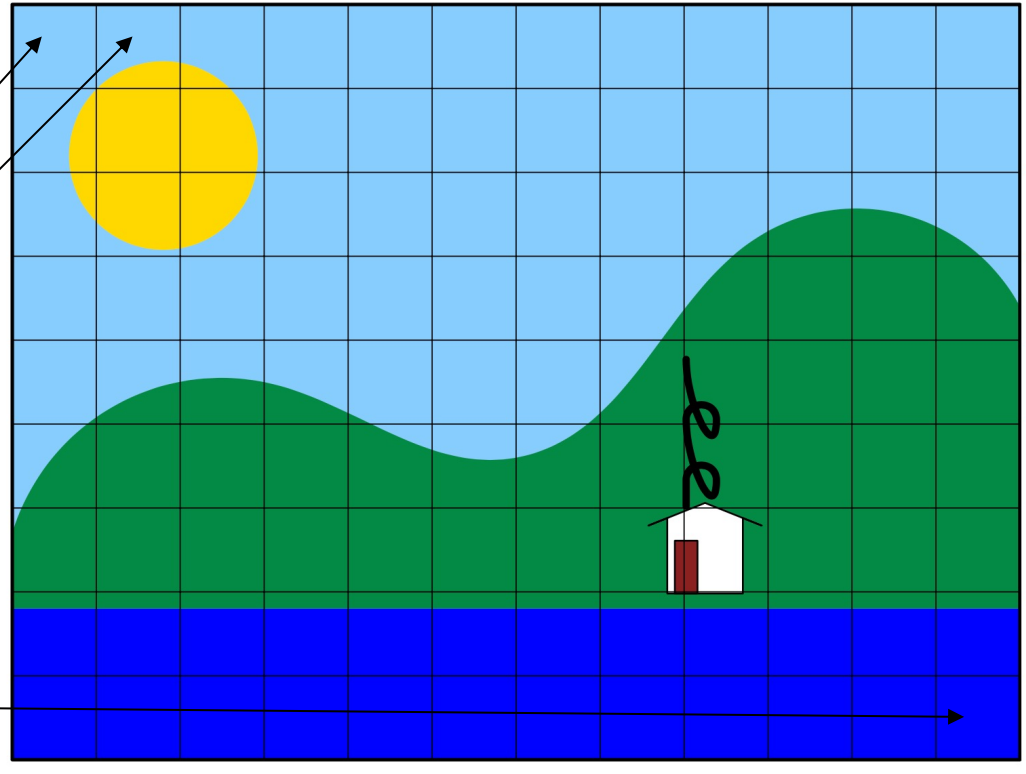
Binær representasjon

- I minnet er alt bare bit
- Det er opp til oss, som programmerere, hvordan disse bitene skal tolkes
  - Positive/negative tall
  - Flyttall
  - Tekst
  - Bilder
  - Lyd
  - Kode
  - Etc

For å lagre et bilde må vi dele det inn i passelige biter som kan representeres binært

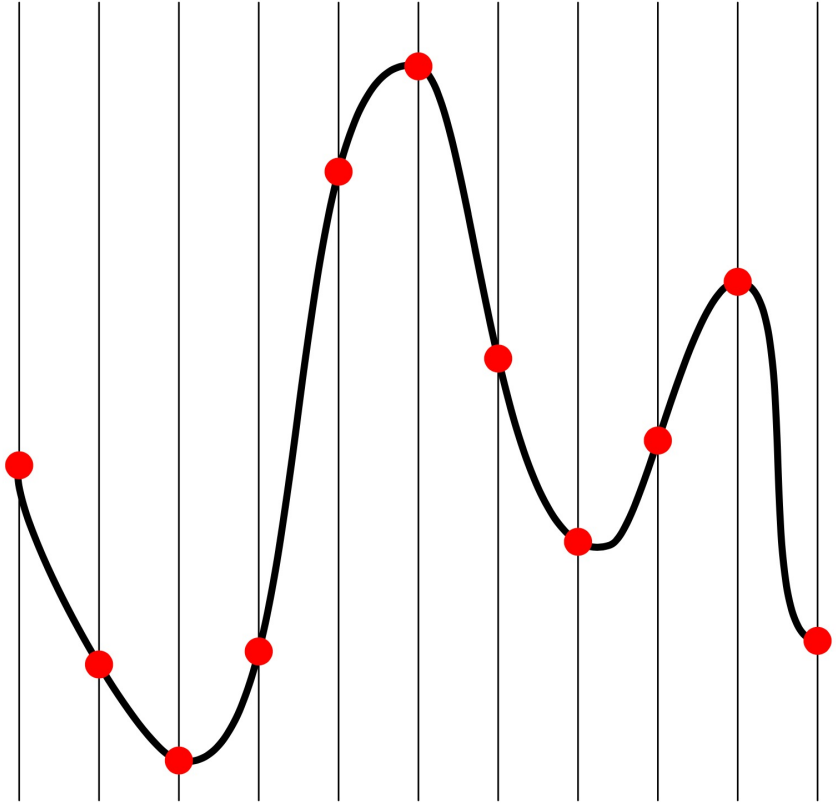
Vi legger et rutenett på bildet, og noterer mengden R, G og B i hver rute. Om vi bruker 1 byte til hver fargemengde, får vi:

R	G	B
135	206	255
135	206	255
135	206	255
...		
0	0	255

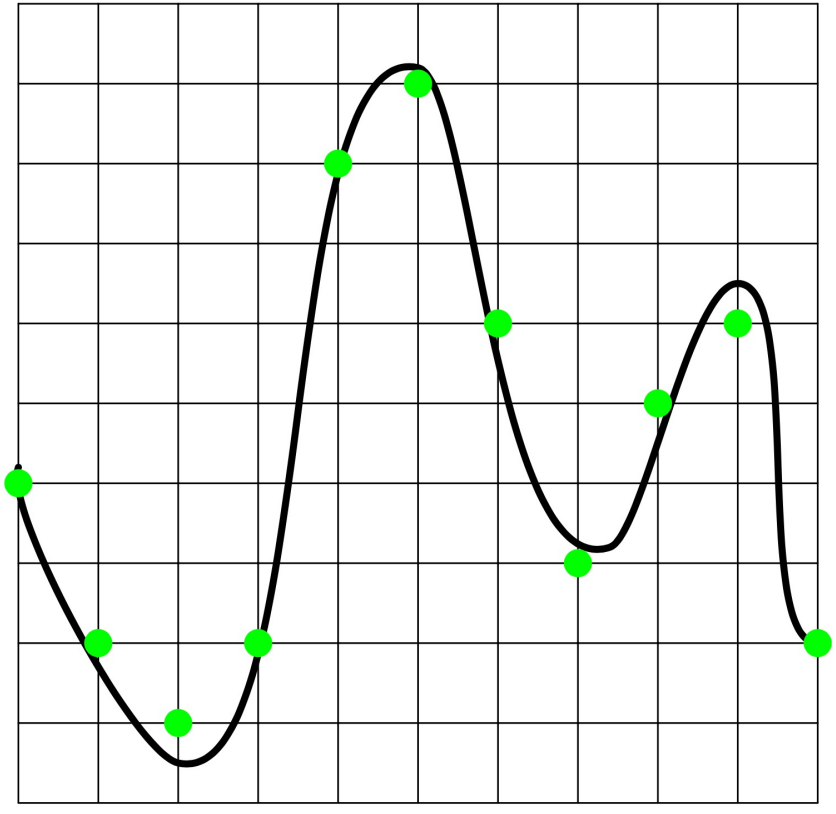


Slike rutenett-representasjoner kalles *rasterbilder*

Vi kan lagre lyden ved å måle strømmen med jevne mellomrom:



Men for å lagre digitalt må vi måle styrken i faste intervaller:



Lykke til! Spørsmål?