

IN1020 uke 4

GRUPPE 6

Plan for dagen

- Menti
- Oppsummere alt det viktigste vi har vært gjennom
- Eksamensoppgaver
- Ukesoppgaver og obligjobbing

Digital representasjon

- Hvordan lagrer vi data?
 - Tall
 - Tekst
 - Bilder
 - Lyd
- Hva er et bit?
 - 0 eller 1
 - False eller True
 - Rød eller grønn
 - Lys av eller på
 - osv
- Binære tall (binary digit, bit)
 - Bruker 0 og 1 for å representere andre tall
 - Telle binært
- Fremgangsmåte for å telle binært:
 - Øk siste siffer i tallet med 1
 - Hvis det ikke er flere sifre, sett siste til 0, og gjenta for sifferet til venstre
- Hvis på tavlen My!

Notasjon

- Vi bruker notasjon for å vite hvilken base tallet er i

- 1001 – er det et binærtall eller et desimaltall?

- 1001_2 og 1001_{10} gjør skillet tydelig

- $1001_2 = 9_{10}$

OBS: Selve basen skrives alltid i det desimale systemet

- $1001_{10} = 3E9_{16}$

- OBS: $1001_{10} = 0x3E9$

25_{10} til binært?

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

Fremgangsmåte:

- Dele tallet på 2 og se hva vi får i rest:
 - Kan få 0 i rest eller 1
 - Ettersom det er et odde eller partall

Svaret Leser vi nedenfra: $25_{10} = 11001_2$

11001₂ til desimal?

Posisjon:

4	3	2	1	0
1	1	0	0	1

$$1*2^4 + 1*2^3 + 0*2^2 + 0*2^1 + 0*2^0$$

$$= 16 + 8 + 0 + 0 + 1$$

$$= 25$$

Heksadesimal notasjon

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Rad 1: Desimal (base 10)

$$1\ 000\ 000_{10} =$$

Rad 2: Binær (base 2)

$$1111\ 0100\ 0010\ 0100\ 0000_2 =$$

0x F 4 2 4 0

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

$$\begin{array}{r} = 00110010 \\ \text{inverserer} \downarrow \\ = 11001101 \\ + \quad \quad \quad 1 \\ \hline = 11001110_2 = -50_{10} \end{array}$$

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

Legger til 2 nuller foran for å få en byte

= 110010

= 00110010

LMC – Little man computer

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

Assembly Language Code

OUTPUT

CPU

00 PROGRAM COUNTER

INSTRUCTION REGISTER

ADDRESS REGISTER

ACCUMULATOR 000

ARITH-METIC UNIT

INPUT

RAM

0	1	2	3	4	5	6	7	8	9
000	000	000	000	000	000	000	000	000	000
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	000	000

ASSEMBLE INTO RAM RUN STEP

RESET LOAD HELP SELECT

INPUT

02

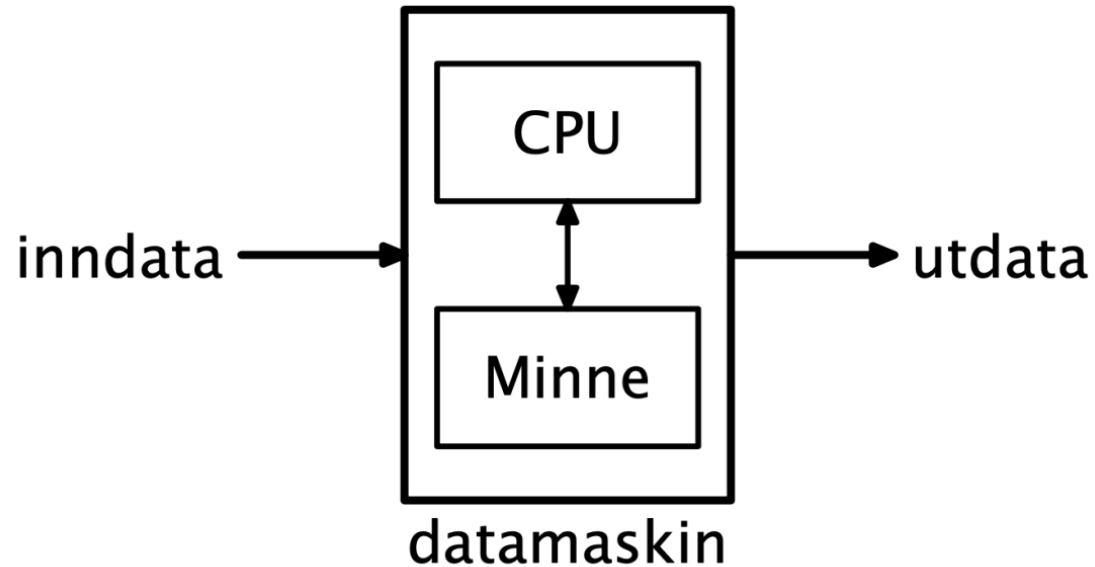
01

RUN/STEP your program, SELECT, LOAD or edit program

OPTIONS

@GCSEcomputing.org.uk and Peter Higginson

Von Neumann-arkitekturer



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

Assembly Language Code

OUTPUT

CPU

00 PROGRAM COUNTER

INSTRUCTION REGISTER

ADDRESS REGISTER

ACCUMULATOR 000

ARITH-METIC UNIT

INPUT

RAM

0	1	2	3	4	5	6	7	8	9
000	000	000	000	000	000	000	000	000	000
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	000	000

ASSEMBLE INTO RAM RUN STEP

RESET LOAD HELP SELECT

INPUT

OUTPUT

RAM

02

01

RUN/STEP your program, SELECT, LOAD or edit

OPTIONS

@GCSEcomputing.org.uk and Peter

Minne

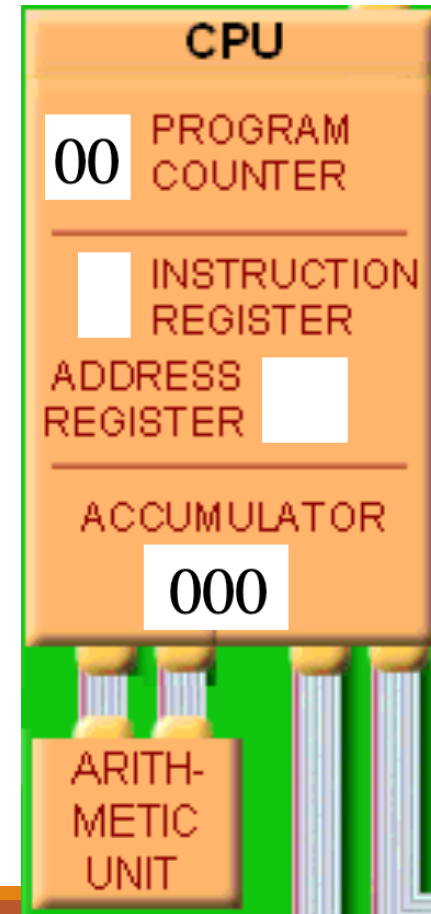
- 99 hyller
- Hylle inneholder:
 - Instruks
 - Data
- Minne = Oppskrift
- CPU = Den som bruker oppskriften

The screenshot displays the Little Man Computer V1.3 interface. On the left, there is a large empty box for 'Assembly Language Code'. To its right is the 'CPU' section, which includes a 'PROGRAM COUNTER' set to 00, an 'INSTRUCTION REGISTER', an 'ADDRESS REGISTER', and an 'ACCUMULATOR' set to 000. Below the CPU is the 'ARITH-METIC UNIT'. At the top right, there is an 'OUTPUT' box and an 'INPUT' box. The central part of the interface is a grid representing 'RAM' with 100 cells (addresses 000 to 999). Each cell contains the value '000'. At the bottom, there are control buttons: 'ASSEMBLE INTO RAM', 'RUN', 'STEP', 'RESET', 'LOAD', 'HELP', and 'SELECT'. A small blue robot character is visible at the bottom center. The title 'Little Man Computer V1.3' is at the top right, and the copyright notice '@GCSEcomputing.org.uk and Peter Higginson' is at the bottom right.

Address	Value
000	000
001	000
002	000
003	000
004	000
005	000
006	000
007	000
008	000
009	000
010	000
011	000
012	000
013	000
014	000
015	000
016	000
017	000
018	000
019	000
020	000
021	000
022	000
023	000
024	000
025	000
026	000
027	000
028	000
029	000
030	000
031	000
032	000
033	000
034	000
035	000
036	000
037	000
038	000
039	000
040	000
041	000
042	000
043	000
044	000
045	000
046	000
047	000
048	000
049	000
050	000
051	000
052	000
053	000
054	000
055	000
056	000
057	000
058	000
059	000
060	000
061	000
062	000
063	000
064	000
065	000
066	000
067	000
068	000
069	000
070	000
071	000
072	000
073	000
074	000
075	000
076	000
077	000
078	000
079	000
080	000
081	000
082	000
083	000
084	000
085	000
086	000
087	000
088	000
089	000
090	000
091	000
092	000
093	000
094	000
095	000
096	000
097	000
098	000
099	000

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
- Kontrolllogikk til å utføre ting i riktig ordre



Alle instruksjonene i LMC

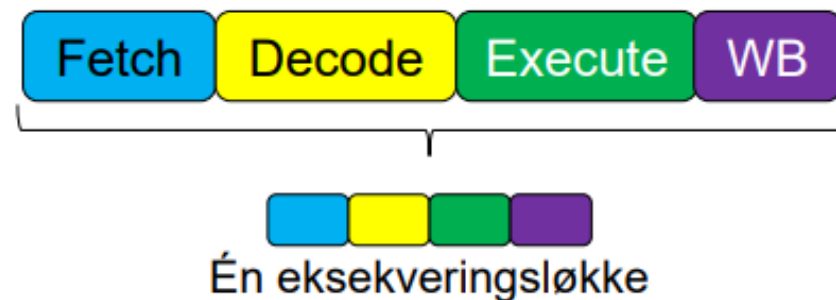
Kode	Navn	Beskrivelse
0xx	HLT	Stopper eksekveringen
1xx	ADD	Adderer verdien i angitt minnelokasjon med akkumulatoren
2xx	SUB	Subtraherer verdien i angitt minnelokasjon med akkumulatoren
3xx	STA	Lagrer akkumulatoren i angitt minnelokasjon
4xx	-	Ikke i bruk
5xx	LDA	Henter verdi fra minnet til akkumulatoren
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)

Eksekveringsløkken

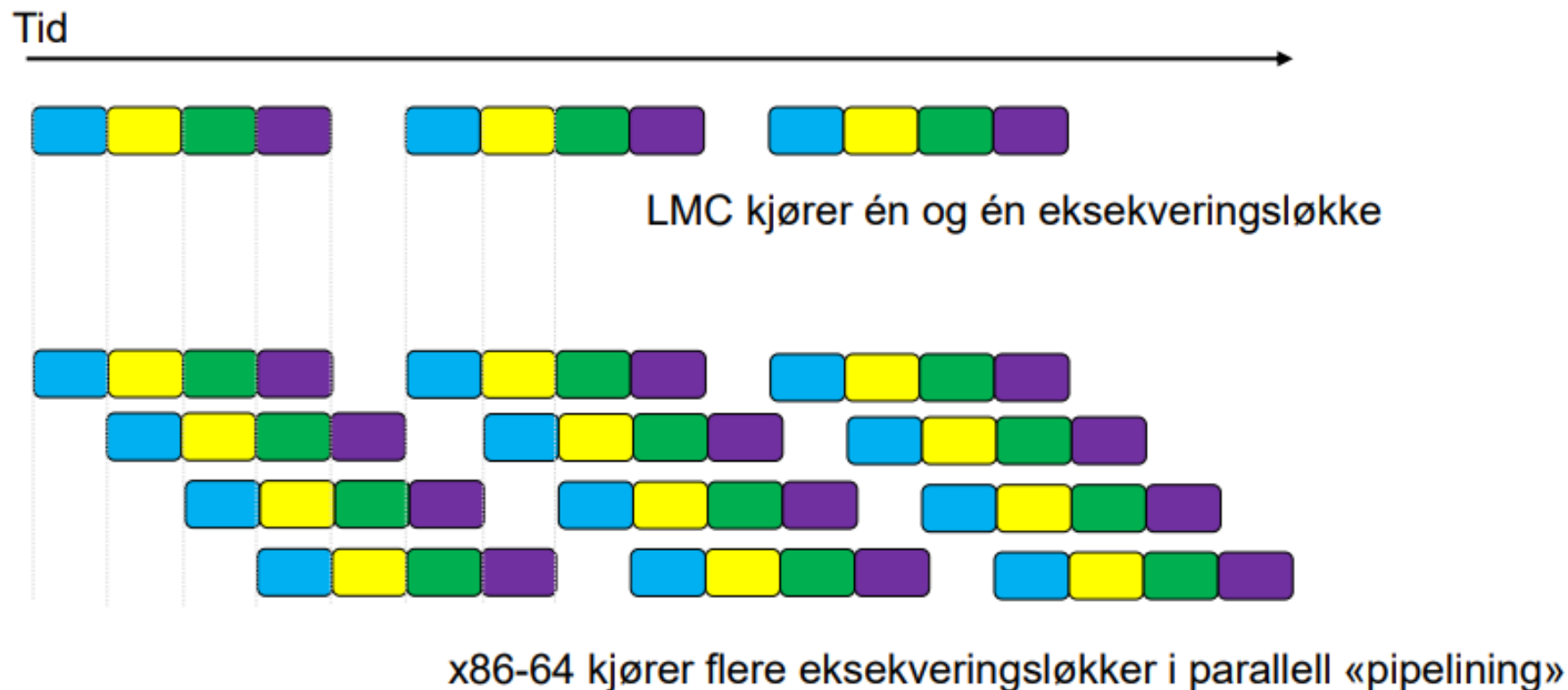
Både LMC og x86-64 har samme eksekveringsløkke:

1. Hent instruksjonen i minnet; programtelleren angir hvor.
2. Dekod instruksjonen i operasjon og adresse; de legges i instruksjonsregisteret og adresseregisteret.
3. Utfør instruksjonen.
4. Om aktuelt, skriv svaret i minnet/register.

x86-64 har i tillegg pipeline for å kunne jobbe raskere.



Pipelining



3(a) **Binære tall 1**

Verdien 27_{10} (dvs 27 i 10-tallsystemet) kan også representeres i andre tallsystemer. Hvilke av disse verdiene er lik 27_{10} ?

Velg ett eller flere alternativer:

- 102_5
- 123_4
- 1000_3
- 11001_2

Maks poeng: 3

3(c) **Bit og byte**

En byte inneholder disse bit-ene:

0	1	1	1	0	0	0	1
---	---	---	---	---	---	---	---

Hvilke verdier kan representeres av disse bit-ene?

Velg ett eller flere alternativer

- 97
- 113
- 143
- 65

Maks poeng: 3

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

Maks poeng: 4

Takk for i dag!

- Jobbe videre med oblig eller ukesoppgaver
- Send meg mail eller teamsmelding hvis dere lurer på noe eller trenger hjelp til oblig