DIGITAL REPRESENTASJON OG ASSEMBLERKODE

BEGREPER / FORKLARINGER

Binære tall: tall med grunntall 2, notasjon 1001_2

Oktale tall: tall med grunntall 8

Desimale tall: tall med grunntall 10

Hextall: tall med grunntall 16 (sifre fra 0 – 9, og A – F)

Bit: verdi med 0 eller 1

- Representerer «noe» som har to tilstander
- Alt er bit i en datamaskin; tekst, bilder og lyd

Fortegnsbit: øverste bit som angir pluss eller minus

Byte: samling av 8 bits

Short: 2 byte, 16 bit

Int: 4 byte, 32 bit

Long: 8 byte, 64 bit

ASCII: 7-bits tegnsett (ett tegn per byte)

ISO Latin-1: 8-bits tegnsett (ett tegn per byte)

Unicode: universelt tegnsett (4 byte per tegn), dominerer for koding av tegn

UTF-8: lagring av Unicode med 1 – 4 byte per tegn, dominerer for koding av tegn

PNG: bildeformat for rasterbilder

JPEG: bildeformat for rasterbilder

MP2: lydformat (gammel DAB)

MP3: lydformat (iPod, MP3-spiller)

AAC: lydformat (DAB+)

Lydkoding med MP2, MP3 og AAC er blitt standard, men man bør velge rett kvalitet

H264: Advanced Video Coding, utbredt standard for videokomprimering

GIF: en liten sekvens (video) av noen få, små bilder som sendes over nettet

PostScript: datamaskinspråk som regulerer beskrivelsen av en side og fungerer som et bindeledd mellom datamaskinen (tekstbehandler eller grafikkproduserende program) og utrskriftsenheten

Run length – koding: lagrer fargen og hvor mange slike piksler det er ved siden av hverandre, brukes av PNG og GIF

Maskinkode: den koden prosessoren utfører

Assemblerkode: den tekstlige formen av maskinkode

Registre: lagerlokasjoner helt tett på prosessoren (de er på 64 bit, altså 8 byte)

Flagg: 1-bits lagre som settes automatisk av prosessoren. Flagg er 0 eller 1 avhengig av hva som skjer i prosessoren

CISC (Complex Instruction Set Computing)

- Bruker omfattende samling av instruksjoner som kan utføre komplekse oppgaver med én enkelt instruksjon
- Målet er å redusere antall instruksjoner per program ved å bruke kraftigere individuelle instruksjoner, noe som kan gjøre programmeringen enklere -> men går utover maskinvarekompleksiteten og det blir lengre utførelsestid for visse instruksjoner

RISC (Reduced Instruction Set Computing)

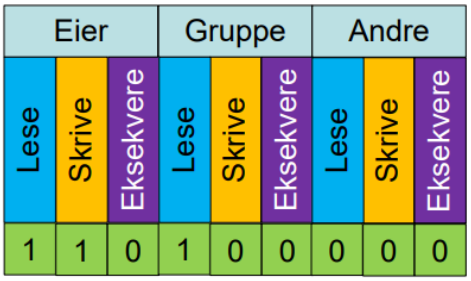
- Fokuserer på en mer begrenset og enkel samling av instruksjoner som kan utføres veldig raskt, vanligvis i en enkelt klokkesyklus
- Det fremmer høy ytelse ved å gjøre det mulig å utføre flere instruksjoner per sekund

Rasterbilde

- Type digitalt bilde som er bygget opp av et rutenett av individuelle piksler
- Hver piksel har en bestemt farge og plassering i rutenettet
- Til sammen danner disse pikslene bildet

Filtillatelser

- Tilgangene representeres med tre oktale tall
  - Disse gjøres om til binære tall
- Bit 1 = … bruker har tillatelse til å *plassering*
- Bit 0 = … bruker har ikke tillatelse til å *plassering
- Eksempel ved tallet 774_8:
  - 7 (111) = eier, tillatelse til alt
  - 7 (111) = gruppe, tillatelse til alt
  - 4 (100) = andre, tillatelse til å lese



HVA er en datamaskin?

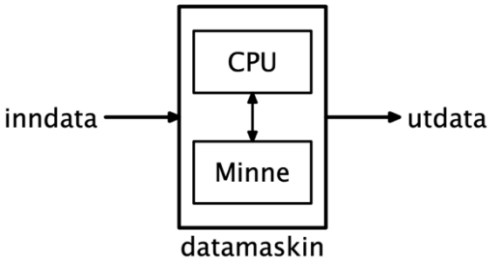
- Generell (kan løse alle typer problemer vi ønsker at den skal løse)
- Programmerbar (vi må kunne programmere den i minnet på maskinen)
- Ferdig, og tatt i bruk (ikke bare en idé på papiret: den skal eksistere og ha gjort nytten sin)
- Bør også være: binær og elektronisk

Datahistorie

- **Analoge maskiner**
  - Verdier representeres av fysiske størrelser: lengde, rotasjon, elektrisk spenning etc.
  - Dominerte frem til rundt andre verdenskrig
- **Digitale maskiner**
  - Verdier representeres symbolsk
- **Før datamaskiner ble vanlige**
  - Regnet på papir
  - Mekaniske regnemaskiner
  - Tabeller (eks. logaritmetabeller, tidevannstabeller)
- **Antikythera-maskinen**
  - Trolig laget mellom år 200 – 100 fvt.
  - Analog maskin
  - Besto av minst 30 tannhjul
- **Difference Engine**
  - 1820-årene
  - Planlagt av Charles Babbage
  - For å lage tabeller automatisk
- **«Oslo-analysatoren»**
  - 1930-årene
  - Verdens mest avanserte analoge maskin: bygget på UiO
  - Professor Svein Rosseland
  - Løse avanserte differensialligninger, og konfigureres ved å flytte rundt på de forskjellige delene
- **Eniac – Electronic Numerical Integrator and Computer**
  - 1939 – 1946
  - Laget for å beregne ballistiske tabeller
  - I drift frem til 1955
- **EDSAC – Electronic Delay Storage Automatic Calculator**
  - 1949
  - Første som kjørte
- **NUSSE – Norsk Universell Siffermaskin, Selvstyrt, Elektronisk**
  - Bygget 1950 – 1954
  - Første norske maskin

Von Neumann-arkitektur

- Funnet opp av John von Neumann og Alan Turing



TEKST

ASCII-tegn

- Enkelt å sjekke om det er et siffer eller en bokstav
- Enkelt å konvertere fra en liten bokstav til en stor bokstav (ligger akkurat 32 bit unna hverandre)
- Trenger 7 bit for å lagre alle verdiene, under 128 ulike verdier
- Inneholder kontrolltegn/styretegn, brukt til å jobbe med fjernskrivere
- MINUS: mangler en del bokstaver og tegn (ÆØÅ)

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

ISO 8859-1

- Halvparten av tabellen (128 verdier) er ASCII

Unicode

- Tegnkoding som omfatter alle skriftspråk i verden
- Plass til rundt 1 000 000 tegn: foreløpig er rundt 143 000 tegn definert
- Det meste av nyere programvare støtter Unicode

BILDER

- Hvert bildepunkt (piksel) består av tre farger: rød, grønn og blå, disse kan lyse sterkt eller svakt
- For å lagre et bilde må vi dele det inn i passelige biter som kan representeres binært
- Hvitt – 1, svart – 0
- Ofte en byte til hver farge, dvs. 3 byte per piksel
- Kvalitet kontra plass, for å redusere plass kan man lage en fargetabell for å se hva fargene inneholder og hvilke piksler som inneholder hvilke farger
- **Run length-koding**: lagrer fargen og hvor mange slike piksler det er ved siden av hverandre, brukes av PNG og GIF

Fotografi

- Ikke brå overganger
- Ekte rasterbilde tar stor plass
- JPEG komprimerer bildet slik at det tar mindre plass, men taper informasjon (altså fjerner unødvendig informasjon)
- Mennesker kan bare skjelne et begrenset antall nyanser, dette utnyttes

Vektorgrafikk

- Lagrer bilde som geometriske elementer (rette linjer, buet linjer etc.)
- Koder i det grafiske språket, det er matematisk definert

- Kan forstørre så mye man vil uten å tape info (geometri fungerer i alle størrelser)

LYD

- Trykkbølger i luft som f.eks. går inn i en mikrofon og sender elektrisk strøm som gjenspeiler lyden: høyttaler mottar dette og kan gjenskape lyden gjennom membraner i høyttaleren
- Vi kan lagre lyden ved å måle strømmen med jevne mellomrom, men for å lagre digitalt må vi må måle styrken i faste intervaller
- Med disse målingene kan lyden gjenskapes (men ikke helt nøyaktig)
- **Lydkvalitet**: kvalitet av lyd representeres digitalt avhengig av hvor ofte vi måler, og hvor mange trinn vi har
- **For å spare plass**: lagre bare én kanal og litt informasjon om forskjellen mellom høyre og venstre kanal, i stedet for å lagre all informasjon. Ikke lagre hver måling, men lagre forskjellen

Bit og byte

- 1 byte = 8 bit
- Med n bit kan vi representere 2^n unike tall:
  - Vi kan telle fra 0 til 255 med én byte
- Jobber vanligvis med bytes og ikke bits
- Bit er den minste enheten av data i en datamaskin

2'er komplement

- For å kunne lagre både negative og positive tall, må vi bestemme oss for en representasjon av dette
- Representere negative tall i binær form
- Med én byte på toerkomplement kan vi lagre heltall mellom -128 til 127
- Øverste bit (fortegnsbit) tolkes som det negative av den verdien det ellers ville hatt

Instruksjonssykelen i LMC:

1. Bruker verdien i programtelleren som adresse til minnet og henter neste instruksjon der
2. Splitt (dekod) instruksjonen i instruksjonskode og adressedel, og legg disse i instruksjonsregisteret og adresseregisteret
3. Utfør det som registeret angir
4. Øk programtelleren med én
5. Gjenta fra punkt 1

FORKORTELSER

PNG: Portable Network Graphic

JPEG: Joint Photographiv Experts Group

GIF: Graphics Interchange Format

AAC: Augmentative and Alternative Communication

SVG: Scalable Vector Graphics

PDF: Portable Document Format

CD: Compact Disc

CISC: Complex Instruction Set Computing

RISC: Reduced Instruction Set Computing

UTFORMING

GENERELL OMREGNING FRA ANDRE BASER TIL DESIMAL

'tall som skal omgjøres' * 'grunntall'$^{\text{'vekt/posisjon'}}$

Eksempel:

44_3 til desimal:

$4 * 3^0 + 4 * 3^1 = 16_{10}$

GENERELL OMREGNING FRA DESIMAL TIL ANDRE BASER

Dele det desimale taller på grunntallet, og finne rest

VERDI	REST
765 – (47 * 16)	13 (D)
47 – (2 * 16)	15 (F)
2	2
0	

765_{10} = 0x2FD

2-er komplement av binært tall

1. Inverser alle bitene: bytt alle 0-er til 1-er og alle 1-er til 0-er
2. Legg til 1 til resultatet av trinn 1

Eksempel:

- Vi har det binære tallet: 0101
- Inversere alle bitene: 1010
- Legg til 1: 1010 + 1 = 1011

FARGEKODER I HEKSADESIMAL TIL DESIMALTALL:

En fargekode er f. eks: 0x4A412A

Da har vi de heksadesimale tallene:

4A 41 2A

Gjøre om hvert av disse tallene til binær:

01001010 01000001 00101010

Så gjøre om de tre binære tallene til desimaltall:

74 65 42

TABELLER/BILDER

1*2^7	1*2^6	1*2^5	1*2^4	1*2^3	1*2^2	1*2^1	1*2^0
128	64	32	16	8	4	2	1

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

LMC - Instruksjoner

0 HLT	Stopper
1 ADD	
2 SUB	
3 STA	Lagrer i minnelokasjon
5 LDA	Henter verdi i minnelokasjon
6 BRA	Hopper til angitt adresse
7 BRZ	Hopper hvis akkumulatoren er 0
8 BRP	Hopper hvis akkumulatoren er >= 0
901 INP	Leser verdi fra inp, legger svaret i akk
902 OUT	Skriver ut verdi som ligger i akk
922 OTC	Skriver ut verdien I ASCII-tegn

PRØVEEKSAMEN

2'er Komplement

Byten: 10101100 kan representere verdiene -84 og 172

Filtillatelser

Tillatelsen representeres ved det oktale tallet 774_8

User			Group			Other		
Read	Write	Execute	Read	Write	Execute	Read	Write	Execute

Hvilke utsagn er riktige:

- Andre brukere kan kun eksekvere filen.
- Gruppe-eierne av filen kan lese og eksekvere filen.
- Eier av filen kan lese og skrive, men ikke eksekvere filen.

Konvertering

765_{10} -> 0x2FD

765 – (16*47) = 13 (D)

47 – (16*2) = 15 (F)

2 i rest = 2

Tallsortering

$10000000_2 < 10010000_2 < 00000000_2 < 00010110_2$

LMC

Hva skriver LMC ut dersom bruker gir input **5** og så **1**?

```
          INP
          STA a
          INP
          ADD a
          SUB v
          OUT
          HLT
a         DAT
v         DAT 2
```

Velg ett alternativ:

- 4

Du får oppgitt følgende program for LMC:

```
          LDA 90
          OTC
          LDA 91
          OTC
          LDA 90
          OTC
          LDA 92
          OTC
          HLT
```

Initialisèr minnet slik at "**IFI2**" skrives ut når koden kjøres.

90	91	92
73	70	50

```
        INP
        BRP fin
        STA v
        SUB v
        SUB v
fin     OUT
        HLT
v       DAT
```

Programmet ovenfor skriver ut **2**. Hva kan bruker ha gitt som **input** til LMC?
Velg ett eller flere alternativer

- [ ] -1
- [x] -2
- [ ] 5
- [x] 2

Du får oppgitt følgende kode for LMC:

```
start   INP
        ADD jump
        STA jump
jump    DAT 600
        LDA a
        OTC
        BRA start
        LDA b
        OTC
        BRA start
a       DAT 35 // '#'
b       DAT 94 // '^'
```

```
00 INP
01 ADD 03
02 STA 03
03 DAT 600
04 LDA 10
05 OTC
06 BRA 00
07 LDA 11
08 OTC
09 BRA 00
10 DAT 35
11 DAT 94
```

Hva må bruker gi som input for å få skrevet ut "^^" og så stanse?

Første input: 7

Andre input: 0

Tredje input: -607