F2
Datarepresentation – talbaser, dataformat och teckenkodning

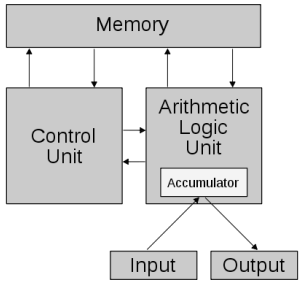
EDAA05 Datorer i system

Roger Henriksson
Jonas Wisbrant

Von Neumann-arkitekturen

Gemensamt minne för programinstruktioner och data.

Sekvensiell exekvering av instruktionerna.



Datarepresentation

I en dator lagras och behandlas all information i form av binära tal – ettor och nollor.

En binär siffra kallas för en *bit* – BInary digiT.
Ett antal bitar kombineras ihop till större enheter:

- Byte – 8 bitar (talområde: 0-255)
- Word – 16, 32 eller 64 bitar

Bitarnas innebörd beror på hur vi väljer att tolka dem – tal, text, ljud, bilder, programinstruktioner, etc.

Talsystem

Vi använder talsystem för att representera numeriska värden. Vårt vanliga, *decimala*, talsystem är ett så kallat *positionssystem*.

Exempel:

$$827 = 8 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0$$
$$827 = 8 \cdot 100 + 2 \cdot 10 + 7 \cdot 1$$

I detta talsystem utgör talet 10 systemets *bas*.
Siffrornas position anger hur många gånger motsvarande potens av basen behöver adderas till.

Binära tal

Det binära talsystemet är ett talsystem med basen 2.
Exempel:[1]

$$1001_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$
$$1001_2 = 1 \cdot 8 + 1 \cdot 1 = 9$$

$$111000_2 = 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$$
$$111000_2 = 1 \cdot 32 + 1 \cdot 16 + 1 \cdot 8 = 56$$

Praktisk sekvens att memorera: 1,2,4,8,16,32,64,128,…

[1] Exemplet helt oförskämt stulet från Per Holm.

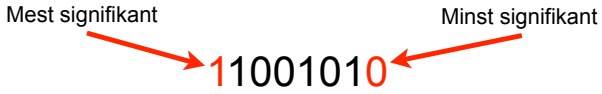
Mest och minst signifikant siffra

Mest signifikant siffra

Den siffra i ett tal som motsvarar störst värde är *mest signifikant*.
Normalt siffran längst till vänster.

Minst signifikant siffra

Den siffra i ett tal som motsvarar minst värde är *minst signifikant*,
dvs entalssiffran (siffran längst till höger).



Addition

Additionstabell:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

Addition görs på liknande sätt som i det vanliga decimala talsystemet:

```
  1 1 1 1 1     (minnessiffror)
    0 1 1 0 1   (13)
+   1 0 1 1 1   (23)
-------------
= 1 0 0 1 0 0   (36)
```


Subtraktion

Analogt med vanlig subtraktion i det decimala talsystemet, men:

När vi "lånar" lånar vi inte 10 utan 2!

Exempel:

```
    *   * * *    (kolumner från vilket lån skett)
  1 1 0 1 1 1 0  (110)
-     1 0 1 1 1  (23)
---------------
= 1 0 1 0 1 1 1  (87)
```

Negativa tal

Naturligtvis kan vi skriva ett negativt binärt tal som -10011, men datorn saknar förståelse för minustecknet.

Därför måste vi representera negativa tal på ett annat sätt i datorn.

Vi vill t.ex. att:

- en addition med ett ska alltid ge ett resultat som är ett större än ursprungstalet *oavsett om vi tolkar talet som positivt eller negativt*.
- att datorns hårdvara inte ska påverkas av hur vi tolkar talet.
- varje bitmönster ska motsvara ett, och endast ett, tal.

Naiv teckenrepresentation

Låt den mest signifikanta biten i en byte eller ett ord tala om vilket tecken talet har.

Lagra talet -5 i en byte (8 bitar): 10000101

Vi kan nu representera talen -127…127 med en byte.

Men:

- Både 00000000 och 10000000 motsvarar talet 0 (+0,-0).
- Addition till negativa bitmönster blir fel:
  10000101 + 00000001 = 10000110 (-5 + 1 = -6 !!!)

Tvåkomplement

Aven i tvåkomplementsform anger mest signifikanta biten talets tecken.

N-bitars tvåkomplementsform representerar ett tal -X som:

$$2^N\text{-}X$$

Exempel, -7 i 8-bitars tvåkomplentsform:

100000000 - 111 = 11111001

Talområde, 8-bitars tvåkomplementstal: -128…127

Tvåkomplement

Det är enkelt att omvandla ett binärt tal till dess negativa tvåkomplementsform:

1. Invertera talet (byt ettor mot nollor och tvärtom).
2. Addera 1 till talet.

Exempel, -7 i 8-bitars tvåkomplementsform:

1. 7 = 00000111
2. Invertera: 11111000
3. Addera 1: 11111001, klart!

Tvåkomplement

Omvandlingen fungerar också åt andra hållet:

1. -7 = 11111001
2. Invertera: 00000110
3. Addera 1: 00000111, klart!

Tvåkomplement

Varför fungerar den enkla omvandlingsformen?

Jo, om vi t.ex. ska ta reda på hur -X representeras i 8-bitars tvåkomplementsform utgår vi från definitionen:

100000000 - X

Detta är samma sak som:

1 + 11111111 - X

Men att subtrahera något från ett binärt tal som bara består av ettor är samma sak som att invertera talet!

Alltså är uttrycket ovan samma sak som att invertera X och sedan addera 1!

Aritmetik med tvåkomplement

Vi kan hantera addition precis som vanligt utan att egentligen behöva veta om talen ska betraktas som negativa i tvåkomplementsform eller som stora positiva heltal:

Exempel, 00000111 + 11111100 (8-bitarstal):

```
             tvåk.  pos.
  00000111     7      7
 +11111100    -4    252
 =========
(1)00000011     3      3 (259 - overflow)
```


Subtraktion med tvåkomplement

Räkna ut 00000111 - 00000100!

Börja med att negera andra termen och byt till addition. Utför därefter additionen:

00000111 - 00000100 = 00000111 + 11111100 = 00000011 (efter att ha slängt bort "overflow-ettan")

Hexadecimala tal

Talsystem med 16 som bas.
Praktiskt eftersom $16 = 2^4$, vilket innebär att fyra binära siffror direkt kan översättas till en hexadecimal.
Siffror: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Konvertera mellan binärt och hexadecimalt:

0101 1010 1001 1111 = 5A9F

Gruppera de binära siffrorna i grupper om fyra!

Hexadecimalt till decimalt

Konvertera 5A9F till decimalt:

$$5A9F_{16} = 5 \cdot 16^3 + 10 \cdot 16^2 + 9 \cdot 16^1 + 15 \cdot 16^0 =$$
$$20480 + 2560 + 144 + 15 = 23209$$

Oktala tal

Talsystem med 8 som bas.
$8 = 2^3$, så det är enkelt att konvertera grupper om tre binära siffror till oktal form och vice versa.

Exempel:
Unix-kommandot "chmod" som ändrar accessrättigheterna för en fil:

```
chmod 755 filnamn
```

$755_8 = 111\ 101\ 101_2$,
vilket motsvarar rättigheterna rwxr-xr-x

Flyttal

Approximativ representation av reella tal.

Ett flyttal består av tecken (t, -1/1), mantissa (m), exponent (e) och en bas (b, normalt 2 i datorer):

$$tal = t \cdot m \cdot b^e$$

Jämför med hur ni ibland skriver tal i matematiken:
Exempel:

$$2306000 = 2{,}306 \cdot 10^6$$

Exponenten avgör decimalkommats plats, som kan sägas "flyta", därav namnet flyttal.

IEEE754

Standard för hur man representerar flyttal i en dator.

Definierar flera format med olika noggrannhet och talomfång.

Vanligast är idag 64-bitars binära flyttal (8 bytes):

t = teckenbit

e = exponenten (-1022…1023)

m = mantissan:

$$m = 1 + d_1 \cdot 2^{-1} + d_2 \cdot 2^{-2} + \ldots + d_{52} \cdot 2^{-52}$$

Varning – flyttal

- Flyttal har en begränsad noggrannhet (ca 16 decimala siffror).
- Avrundningsfel kan göra att de vanliga matematiska lagarna inte alltid gäller:
  - $a + b - c \neq a - c + b$
  - $a/3 + a/3 + a/3 \neq a$
- Undvik att jämföra om två flyttal är exakt lika!

Teckenkodning

ETT STORT ELÄNDE!!!

Teckenkodning

Vi använder numeriska värden för att representera skrivtecken enligt en teckentabell.

Historiskt sett dåligt standardiserat. Olika standarder för vilket värde som representerar vilket tecken.

Olika standarder för hur man representerar radslut.

Teckenkodning – historik

Baudotkod – 5-bitars kod för fjärrskrivare (teletype – tty).

00	01	02	03	04	05	06	07
NUL	E 3	LF	A -	SP	S '	I 8	U 7
08	09	0A	0B	0C	0D	0E	0F
CR	D ENQ	R 4	J BEL	N ,	F !	C :	K (
10	11	12	13	14	15	16	17
T 5	Z +	L)	W 2	H £	Y 6	P 0	Q 1
18	19	1A	1B	1C	1D	1E	1F
O 9	B ?	G &	FIGS	M .	X /	V ;	LTRS
Letters			Figures		Control Chars.		

Baudot 1874, Murray 1901

EBCDIC

Extended Binary Coded Decimal Interchange Code

8-bitars IBM-standard som huvudsakligen användes i IBMs stordatorsystem.

Idag mest en historisk parantes.

ASCII

American Standard Code for Information Interchange (1963)
7-bitars teckenkod ursprungligen utvecklad för fjärrskrivare.

USASCII code chart

Row	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
10	LF	SUB	*	:	J	Z	j	z
11	VT	ESC	+	;	K	[	k	{
12	FF	FS	,	<	L	\	l	|
13	CR	GS	-	=	M	]	m	}
14	SO	RS	.	>	N	^	n	~
15	SI	US	/	?	O	_	o	DEL

Svensk ASCII

Olika modifierade 7- och 8-bitars ASCII-kodningar för att representera internationella alfabet.

ISO/IEC 646, svensk variant, endast 20-7F visade

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
2x	blank	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	Ä	Ö	Å	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	ä	ö	å	~	DEL

Observera ordningen på Å, Ä och Ö.

Radslut

Radslut representeras med ett (eller flera) specialtecken.
Tyvärr inte helt standardiserat.

Windows, och flera Internetprotokoll:
Carriage Return + Line Feed (ASCII 13 + ASCII 10)

Unix, Linux
Line Feed (ASCII 10)

Gamla MacOS (före Mac OSX)
Carriage Return (ASCII 13) – nu mest en historisk parantes.

ISO 8859

En serie 8-bitarsvarianter av ASCII.

- ISO 8859-1 – Latin-1, västeuropeiska
- ISO 8859-2 – Latin-2, östeuropeiska

Även för:
grönländska (3), kyrilliska (5), arabiska (6), grekiska (7), hebreiska (8), osv…

Inte heller här har Å,Ä,Ö hamnat rätt…

ISO/IEC 8859-1

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	reserverat för styrtecken (från ISO/IEC 6429)															
1x																
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8x	*reserverat för styrtecken (till exempel (!) från ISO/IEC 6429)*															
9x																
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Variant med extra tecken i stället för kontrolltecken 7F-AF:
Windows-1252.

Unicode/ISO 10646

Utvecklat för att klara världens alla skriftspråk.
Mer än 100000 möjliga skrivtecken.
Tar över allt mer…

UTF-8
Unicodetecken lagras i 1-4 bytes.
Tecken ur ASCII-koden representeras likadant i UTF-8.
Ovanligare tecken representeras som en sekvens av flera bytes.

Mer om Unicode/UTF-8 nästa vecka…