Grundlagen der Informatik

Angewandte Informatik

WS 20/21

Contents

1 Intro

1.1 Representation of numbers characters

Numbers and characters are saved in the memory and need a binary representation. There are different ways how one can represent numbers and charachters, depending on the needs the program has. Having a programm which needs a counter, only nedds positive integers so there is no need for saving decimals. Also the range is important. Is the programm counting to 100 or 100 milion. Different datatypes need less bytes to store data, but then the range or (Genauigketi) suffers.

Integers

With unsigned (only positive) integers only differ in how many bits they use. Typicall sizes are 8-bit (short), 16-bit (half word), 32-bit (word) and 64-bit (double word).
Signed integers need to save the minus symbol somewhere. There are several options to "save the minus". One is just saying if the MSB is 1, the number is negative. The problem is that 0000 and 1000 are both 0, but one is a positive and one is a negative 0 which isn't very effictive.
Another implementation is the one-complement. Here you just invert every bit to get the "negative version" of the number. Again the ± 0 is possible, but the one-complement creates a symmetrie with negative and ppostive numbers and is needed for the two-complement. The two-complement takes the result from the one complement and adds $+1$ to it. The symmetrie is gone but the ± 0 is gone (only positve 0) and an extra negative numebr is won.
One other way to create negative nummber is by using a bias/offset. One needs to define the offset first. Now every number in the memory will be read nad the offset will be subtracted from it. An offset of 128 means that the postive numbers will start at 1000.0000_b[1]. The offset is used in floating point numbers for the exponent.

Decimal numbers

Decimal numbers also have different possible representation. An easy with a fixed point. The number is treated as an integer but at a specific bit, the point is set. The position of the point needs to be definded first. If there are 8-bit to save the number and the point is defined at bit 3, there will be 5 bits for the integer and 3 bits for the mantissa[2]. The probelm is, that very big numbers or very small numbers aren't possible.
Floating point numbers fix this by introducing an exponent to the number. The exponetn has an offset, so it can be negativ. A neagtive exponent makes very small numbers possible, but because the exponent can be positive aswell big numbers are possible too. The formular fot calculating a normalized flaot is:

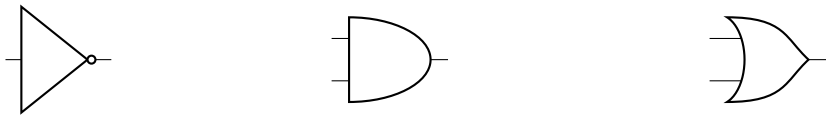
$$f = (-1)^{\text{sign}} \cdot 1.\text{mantissa} \cdot 2^{\text{exponent}}$$

[1] $1000.0000_b \equiv 0$
[2] Nachkommastellen

Depening on how many bits the float uses, different values need to be inserted into the formular.

	sign	mantissa	offset	exponent
32-bit	1-bit	23-bit	127-bit	8-bit
64-bit	1-bit	52-bit	1023-bit	11-bit

2 Boolean Algebra



3 Instruction Set Archetecture (ISA)

The ISA contatins definitions for numbers, adressing and instructions. Numbers were already done in the 1. chapter so they won't be done again.

3.1 Adressing

Adressing is needed for several operations. Mainly to write results(of an summation), read operands(like summands of a sum) and adresses can be a target of an (un-)conditional branch(if statement, return etc.).

4 miscelanious

4.1 ASCII tabel

Dez	Hex	Okt	Zeichen	Dez	Hex	Okt	Zeichen
0	0x00	000	NUL	32	0x20	040	SP
1	0x01	001	SOH	33	0x21	041	!
2	0x02	002	STX	34	0x22	042	"’
3	0x03	003	ETX	35	0x23	043	#
4	0x04	004	EOT	36	0x24	044	$
5	0x05	005	ENQ	37	0x25	045	%
6	0x06	006	ACK	38	0x26	046	&
7	0x07	007	BEL	39	0x27	047	’
8	0x08	010	BS	40	0x28	050	(
9	0x09	011	TAB	41	0x29	051	)
10	0x0A	012	LF	42	0x2A	052	*
11	0x0B	013	VT	43	0x2B	053	+
12	0x0C	014	FF	44	0x2C	054	,
13	0x0D	015	CR	45	0x2D	055	-

14	0x0E	016	SO	46	0x2E	056	.
15	0x0F	017	SI	47	0x2F	057	/
16	0x10	020	DLE	48	0x30	060	0
17	0x11	021	DC1	49	0x31	061	1
18	0x12	022	DC2	50	0x32	062	2
19	0x13	023	DC3	51	0x33	063	3
20	0x14	024	DC4	52	0x34	064	4
21	0x15	025	NAK	53	0x35	065	5
22	0x16	026	SYN	54	0x36	066	6
23	0x17	027	ETB	55	0x37	067	7
24	0x18	030	CAN	56	0x38	070	8
25	0x19	031	EM	57	0x39	071	9
26	0x1A	032	SUB	58	0x3A	072	:
27	0x1B	033	ESC	59	0x3B	073	;
28	0x1C	034	FS	60	0x3C	074	"<
29	0x1D	035	GS	61	0x3D	075	=
30	0x1E	036	RS	62	0x3E	076	">
31	0x1F	037	US	63	0x3F	077	?

Dez	Hex	Okt	Zeichen	Dez	Hex	Okt	Zeichen
64	0x40	100	@	96	0x60	140	‘
65	0x41	101	A	97	0x61	141	a
66	0x42	102	B	98	0x62	142	b
67	0x43	103	C	99	0x63	143	c
68	0x44	104	D	100	0x64	144	d
69	0x45	105	E	101	0x65	145	e
70	0x46	106	F	102	0x66	146	f
71	0x47	107	G	103	0x67	147	g
72	0x48	110	H	104	0x68	150	h
73	0x49	111	I	105	0x69	151	i
74	0x4A	112	J	106	0x6A	152	j
75	0x4B	113	K	107	0x6B	153	k
76	0x4C	114	L	108	0x6C	154	l
77	0x4D	115	M	109	0x6D	155	m
78	0x4E	116	N	110	0x6E	156	n
79	0x4F	117	O	111	0x6F	157	o
80	0x50	120	P	112	0x70	160	p
81	0x51	121	Q	113	0x71	161	q
82	0x52	122	R	114	0x72	162	r
83	0x53	123	S	115	0x73	163	s
84	0x54	124	T	116	0x74	164	t
85	0x55	125	U	117	0x75	165	u
86	0x56	126	V	118	0x76	166	v

87	0x57	127	W	119	0x77	167	w
88	0x58	130	X	120	0x78	170	x
89	0x59	131	Y	121	0x79	171	y
90	0x5A	132	Z	122	0x7A	172	z
91	0x5B	133	[	123	0x7B	173	{
92	0x5C	134	\	124	0x7C	174	|
93	0x5D	135	]	125	0x7D	175	}
94	0x5E	136	^	126	0x7E	176	"
95	0x5F	137	_	127	0x7F	177	DEL