Lecture 19: Binary Numbers and Logic

binary numbers, addition, 2's complement, subtraction, utility of binary numbers

Binary System

Decimal numbers are based on power of 10 while binary numbers are based on powers of 2

$(562)_2=5*10^2+6*10^1+2*10^0$

$(\ 1\ 1\ 0)_2=1*2^2+1*2^1+0*2^0=6$

0000-> 0

1000->8

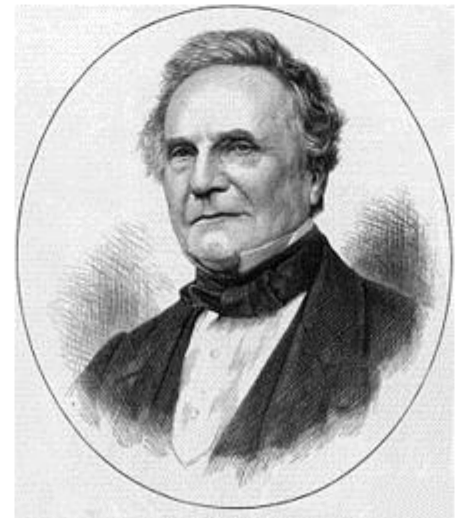
1100->10

1111->15

n bits can represent positive decimal numbers from 0 up to 2^n-1

Digital Systems and Binary Numbers

the first computer was invented in 1822 by Charles Babbage, who was a mechanical engineer, and used decimal numbers and mechanical cranks



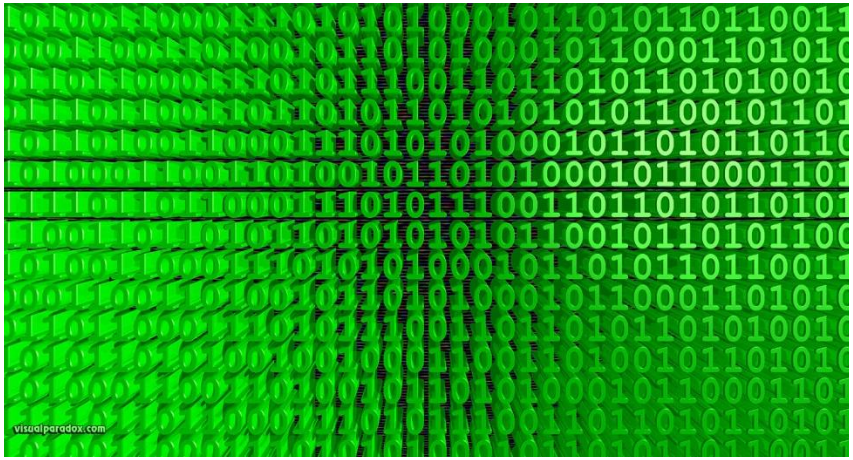
Wikipedia

modern computers use binary numbers for signal processing

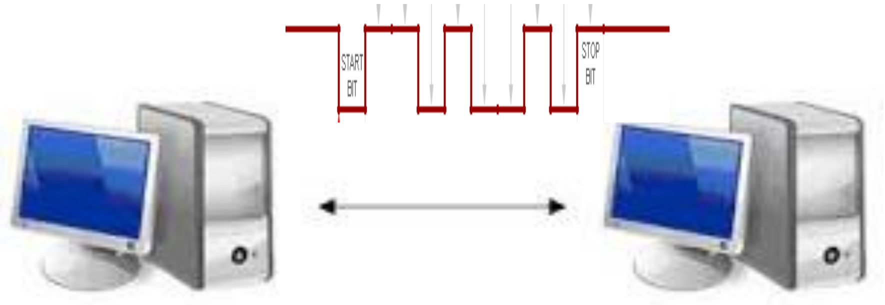
these binary numbers are comprised of bits with every bit assuming a logic “0” or logic “1” value

Digital Systems (Cont’d)

Data is stored inside computer memory in the form of billions of capacitors (bits) where every capacitor is either charged (logic “1”) or discharged (logic “0”)



Computers communicate over the internet or wireless through sequences of binary bits



Examples with Binary Numbers

$(1011)_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = (11)$

$(1.1011)_2 = 1x2^0 + 1x2^{-1} + 0x2^{-2} + 1x2^{-3} + 1x2^{-4}$

$= 1+(1/2)+(0/4)+(1/8)+(1/16) = (1.6875)$

$(-100.1)2$ = (sign?) $1x2^2 + 0x2^1 + 0x2^0 + 0x2^{-1}$

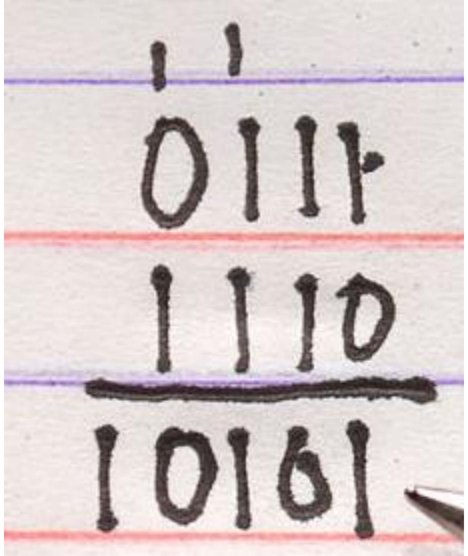
$= (-4.5)$

Addition of Binary Numbers

remember that when adding decimal number we can have a carry over between digits of 0-9

similarly, when adding binary numbers, we can have a carry over between of either 0 or 1

overflow happens when the number of binary digits can not represent the summation



Adding and Subtracting Decimal Numbers

316 + 59 =

316 – 59 =

316 – 905 =

this subtraction approach is not suitable for computers

the subtraction is converted into addition by using complements

Example: 355-316=39. We can get the same answer by first finding the 10's complement of 316 is 1000-316 = 684 and then adding it to 355 to get 355+684= 1039

Subtracting Binary Numbers

we can do the same thing with binary numbers!

2’s complement (2^n – number)

1001 -> 10000 – 1001 -> 0111

100100 -> 1000000 – 100100 -> 011100

notice that the 2’s complement is equivalent to toggling the bits of the original number and adding 1

Subtracting Binary Numbers (Cont'd)

316-59 =100111100 - 000111011

000111011 -> 111000101

100111100 + 111000101 = 1 100000001

= 257 (what about the carry?)

316-905 =0100111100 -1110001001

1110001001 -> 0001110111

0100111100 + 0001110111 = 0110110011

(no carry) 0110110011 -> - 1001001101 = -589

Utility of Binary Numbers

ASCII	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0 0 0 0	NU	SH	SX	EX	ET	EQ	AK	BL	BS	HT	LF	YT	FF	CR	SO	SI
0 0 0 1	DL	D1	D2	D3	D4	NK	SY	EΣ	CN	EM	SB	EC	FS	GS	RS	US
0 0 1 0		!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
0 0 1 1	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0 1 0 0	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0 1 0 1	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0 1 1 0	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0 1 1 1	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
1 0 0 0	Ä	Å	Ç	É	Ñ	Ö	Ü	á	à	â	ä	ã	å	ç	é	è
1 0 0 1	ê	ë	í	ì	î	ï	ñ	ó	ò	ô	ö	õ	ú	ù	û	ü
1 0 1 0	†	°	¢	£	§	•	¶	ß	®	©	™	´	¨	≠	Æ	Ø
1 0 1 1	∞	±	≤	≥	¥	µ	∂	∑	∏	π	∫	ª	º	Ω	æ	ø
1 1 0 0	¿	¡	¬	√	ƒ	≈	∆	«	»	…		À	Ã	Õ	Œ	œ
1 1 0 1	–	—	“	”	‘	’	÷	◊	ÿ	Ÿ	⁄	€	‹	›	ﬁ	ﬂ
1 1 1 0	‡	·	‚	„	‰	Â	Ê	Á	Ë	È	Í	Î	Ï	Ì	Ó	Ô
1 1 1 1		Ò	Ú	Û	Ù	ı	ˆ	˜	¯	˘	˙	˚	¸	˝	˛	ˇ

Utility of Binary Numbers (Cont'd)

Powers of Two

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

Binary Logic and Operators

logic variables that take two values TRUE or FALSE

AND operator (represented with a dot (.))

$x.y$ is TRUE if and only if x is TRUE and y is TRUE

OR operator (represented with a plus (+))

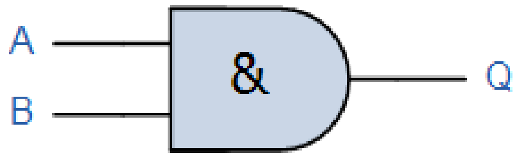
$x + y$ is TRUE if and only if x is TRUE or y is TRUE or both are TRUE

NOT operator (represented with a prime (') or bar ( ))

x' ($\overline{x}$) is TRUE if and only if x is FALSE

Truth Tables

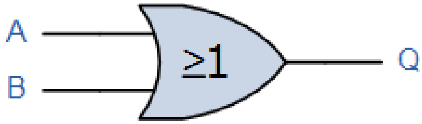
A Truth Table enumerates all combinations of inputs and the output of logical operation

<table>
<tr><th>Symbol</th><th colspan="3">Truth Table</th></tr>
<tr><td rowspan="5">

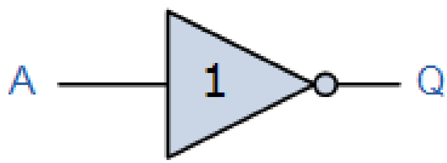
2-input AND Gate</td><td>A</td><td>B</td><td>Q</td></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
<tr><td>Boolean Expression Q = A.B</td><td colspan="3">Read as A AND B gives Q</td></tr>
</table>

http://www.electronics-tutorials.ws/boolean/bool_7.html

Truth Tables (Cont'd)



Symbol	Truth Table		
2-input OR Gate	A	B	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	1
Boolean Expression Q = A+B	Read as A OR B gives Q		



Symbol	Truth Table	
Inverter or NOT Gate	A	Q
	0	1
	1	0
Boolean Expression Q = NOT A or $\overline{A}$	Read as inversion of A gives Q	

http://www.electronics-tutorials.ws/boolean/bool_7.html