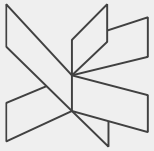
Gør tanke til handling
VIA University College



Lecture 1

Boolean Logic and Boolean arithmetic and breadboard

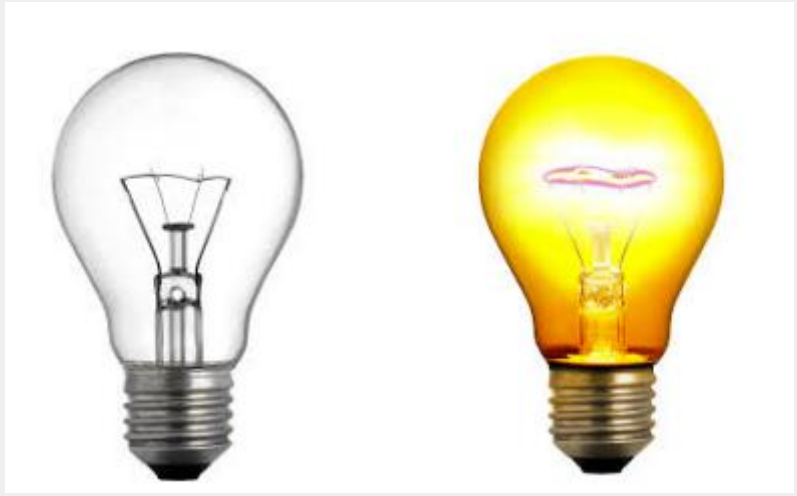
But first an introduction to this course

- How a computer works
- A plan of all the lectures

Todays lecture – MSE review

- Boolean values
- Boolean operations
- Truth table and Boolean functions
- Boolean algebra
- Binary numbers
- Breadboard
- Exercise

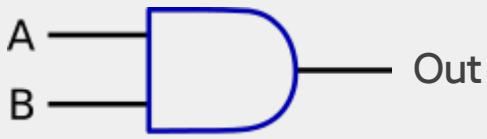
Boolean values



False	True
No	Yes
0	1
Ground	Voltage
0 V	5V/ 3.3V

Boolean operations

AND



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

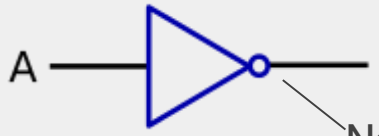
$A * B$

OR

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

$A + B$

NOT



Notice the little circle

A	Out
0	1
1	0

$\overline{A}$

Boolean operations

NAND (Not And)

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

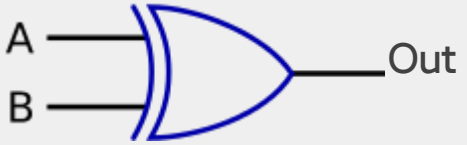
$$\overline{A * B}$$

NOR (Not OR)

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

$$\overline{A + B}$$

XOR



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

$$\overline{A} * B + \overline{B} * A$$

Boolean algebra – laws

Commutative laws

$$X * Y = Y * X$$
$$X + Y = Y + X$$

Associative laws

$$(X * Y) * Z = X * (Y * Z)$$
$$(X + Y) + Z = X + (Y + Z)$$

Distributive laws

$$X * (Y + Z) = X * Y + X * Z$$
$$X + (Y * Z) = (X + Y) * (X + Z)$$

De Morgan laws

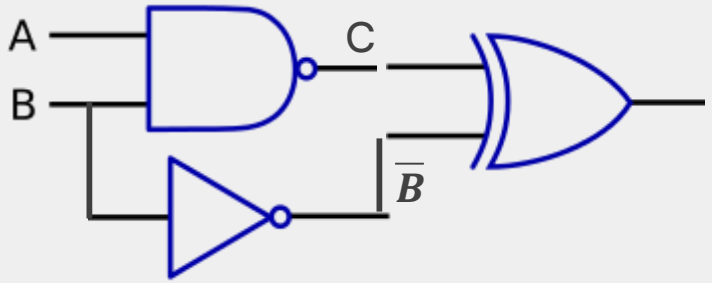
$$\overline{X * Y} = \overline{X} + \overline{Y}$$
$$\overline{X + Y} = \overline{X} * \overline{Y}$$

De Morgan's Laws

Not (**A** *and* **B**) is the same as *Not* **A** *or Not* **B**.
Not (**A** *or* **B**) is the same as *Not* **A** *and Not* **B**.

Associative law, in mathematics, either of two laws relating to number operations of addition and multiplication, stated symbolically: $a + (b + c) = (a + b) + c$, and $a(bc) = (ab)c$; that is, the terms or factors may be associated in any way desired.

Example – what's the truth-table?

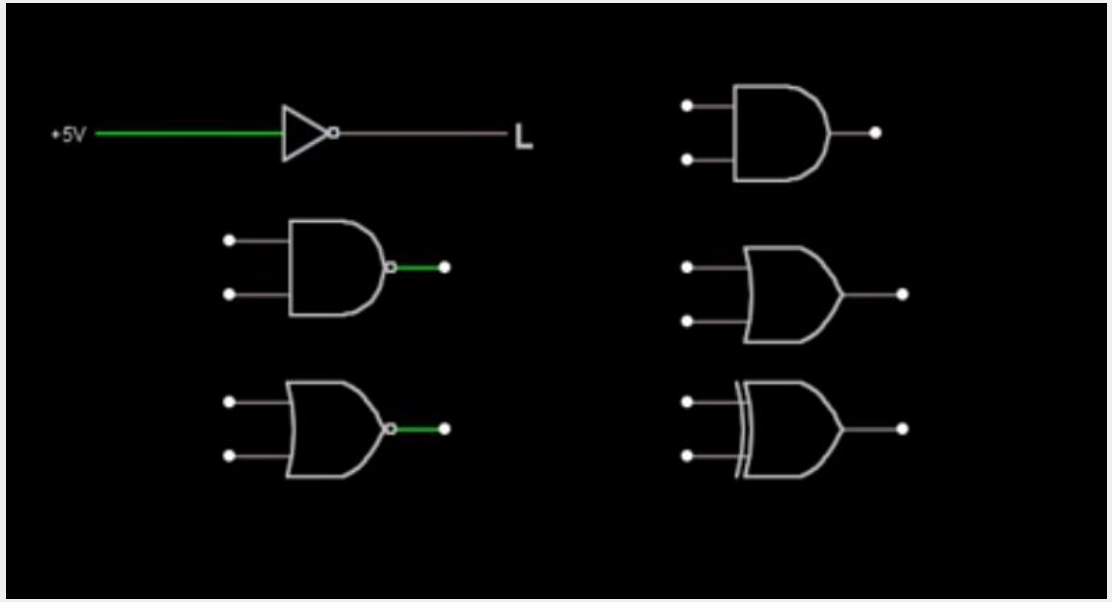


$B * \overline{A}$

A	B	Out
0	0	0
0	1	1
1	0	0
1	1	0

Boolean algebra simulator

Falstad boolean digital logic gates simulator
https://www.falstad.com/circuit/



Fast demo track: https://www.youtube.com/watch?v=aFJ6yZQ7myQ
(Watch youtube video and have special focus on video's time stamp from 14:15 to 16:30)

Binary numbers

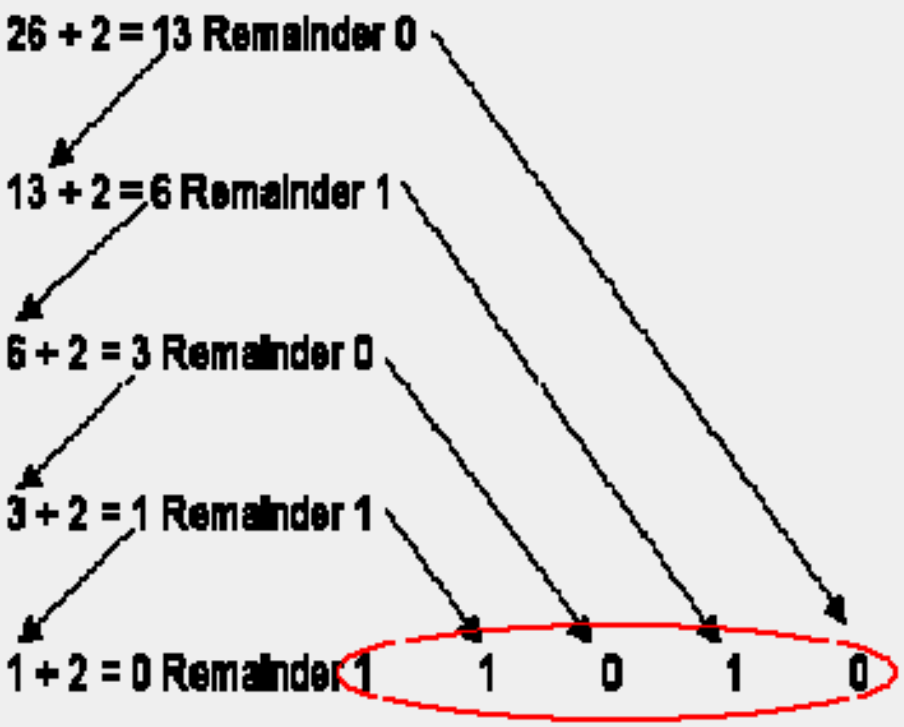
- Decimal numbers have the base 10
- Decimal number is a sequence of numbers that are between 0 and 9, (position based)
- Binary numbers have the base is 2
- A binary number is a sequence of numbers that are between 0 and 1 (position based)

Binary numbers – position based

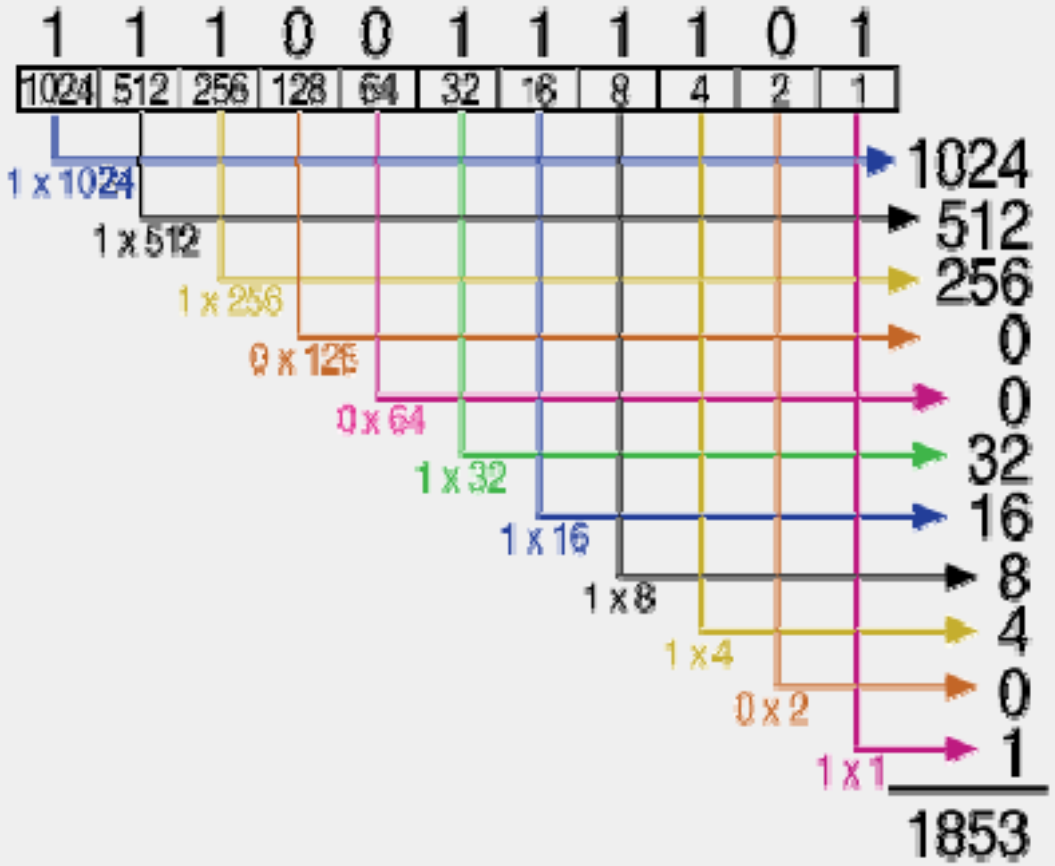
Position	7	6	5	4	3	2	1	0	
Position value	$2^7 = 128$	$2^6 = 64$	$2^5 = 32$	$2^4 = 16$	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	
Example	0	0	0	1	0	1	1	0	
Value	0	0	0	16	0	4	2	0	= 22
Example	1	0	0	1	0	0	1	1	
	128	0	0	16	0	0	2	1	= 147

Convert from decimal to binary

1. Use quotient and remainder
2. Continuously divide the number by 2
3. Process stops when quotient is 0
4. First remainder = LSB
5. Last remainder = MSB



Convert from binary to decimal



Decimal, Binary, and Hex

- Table shows conversion between the first 16 numbers (0 … 15)
- Hex(adecimal) system has base 16 (0..,, A..F)
- One 8-bit binary number (a byte) corresponds to a two-digit HEX number (two nibbles)
- With the AVR assembler code, the decimal number 130 is written as follows:
- Hex it is written0x82
- Binary it is written0b10000010

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

Adding binary numbers

It's exactly like adding decimal number, but…

Stay within the base, and remember that:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 and carry 1

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Carry flag affected by – unsigned arithmetic overflow (ADD, ADC, SUB, …) and logic operations (LSL, LSR, ROL, …)

Subtracting binary numbers

If A and B are binary numbers

Compute A – B as A + (- B)

(- B) is two's complement of B:
Invert all bits of B

Add 1

Compute 0b01001010 – 0b00101111

0b01001010 + (COM(0b00101111) + 1) =

0b01001010 + (0b11010000 + 1) =

0b01001010 + 0b11010001 = 0b100011011

???????

2. compliment – technique to subtract

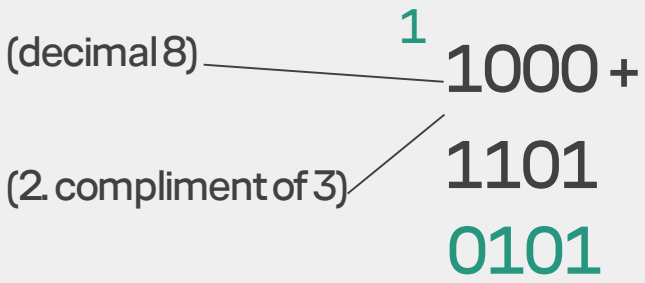
- Invert the bits (1. compliment)
- Add 1

- Add with this number instead of substracting

Example: 8 – 3 = 5

Example: 8 + second_compliment_of_3 = 5

0011
1100 First compliment
1101 Second compliment



Breadboard