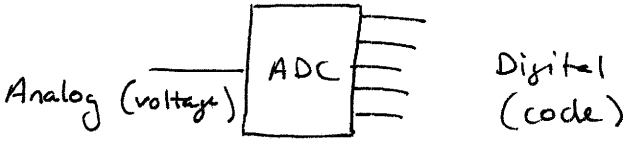
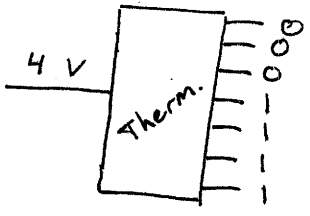
Lecture 2

Information encoded as electrical signals



Last time we used a "Thermometer Code"



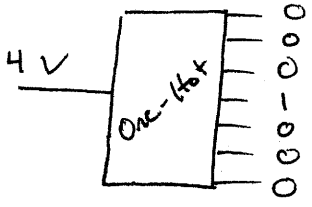
← First 4 are filled

OR: 0001111

$(0001111)_T$

(Note $0001111 = 1111$ in same way $0234 = 234$)

There are dozens of other kinds of codes, e.g. one-hot

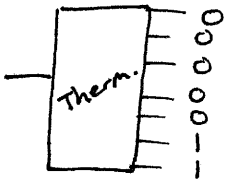


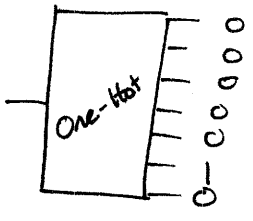
← Only 4th up is on

OR: 0001000

$(0001000)_{OH}$

What would thermometer and one-hot codes be for 2 V?





We will use one-hot today, as well as the most useful digital code - binary.

Decimal and Binary

We need symbols to count, but instead of having a unique symbol for <u>every</u> number, we use a logical pattern:

1, 2, 3, 4, 5, 6, 7, 8, 9

Then

"One group of ten" plus "no extras" = 1 0 $(10)_d$

Then "One group of ten" plus "one extra" = 1 1 $(11)_d$

etc. until

"One group of ten" plus "nine extras" = 1 9 $(19)_d$

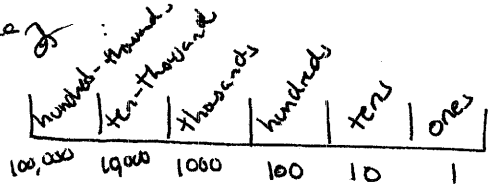
=> "Two groups of ten" plus "no extras" = 2 0 $(20)_d$.

The numbers 10, 11, 19, 20 are not unique symbols, but combinations of only ten symbols in a pattern to represent all possible numbers.

Why Ten? - Probably fingers, but no mathematical preference for ten.

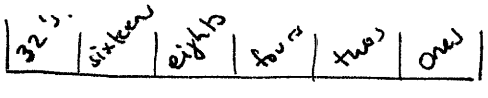
Base-2 or Binary:

Decimal:



10 symbols per box

Binary:



2 symbols per box

The rule for counting is the <u>same</u>:

Keep adding "one" to the lowest box; if it fills up (runs out of symbols), carry the group to the next box as a 1.

$(001)_d$	$(000)_b$
⋮	$(001)_b$
$(009)_d$	$(010)_b$ ↙ Out of symbols, must carry!
$(010)_d$	$(011)_b$
$(011)_d$	$(100)_b$ ↙ Must carry twice.

Converting Binary to Decimal

Just add up each place value you have

Ex. Convert $(100101)_b$ to decimal

32's	sixteen	eights	fours	twos	ones
1	0	0	1	0	1

One 32 + One 4 + one 1 = $(37)_d$

$$(100101)_b = (37)_d$$

Convert $(111011)_b$ to decimal

Solution: $64 + 32 + 16 + 2 + 1 = 115$

$$(111011)_b = (115)_d$$

Converting Decimal to Binary

- Find largest binary place value that fits into digital number
- Subtract, repeat

Ex. Convert $(73)_d$ to binary.

256	128	64	32	16	8	4	2	1
0	0	1	0	0	1	0	0	1

$73 - 64 = 9$
$9 - 8 = 1$
$1 - 1 = 0$

$$(73)_d = (1001001)_b$$

Convert $(121)_d$ to binary:

Solution:

$121 - 64 = 57$
$57 - 32 = 25$
$25 - 16 = 9$
$9 - 8 = 1$
$1 - 1 = 0$

128	64	32	16	8	4	2	1
0	1	1	1	1	0	0	1

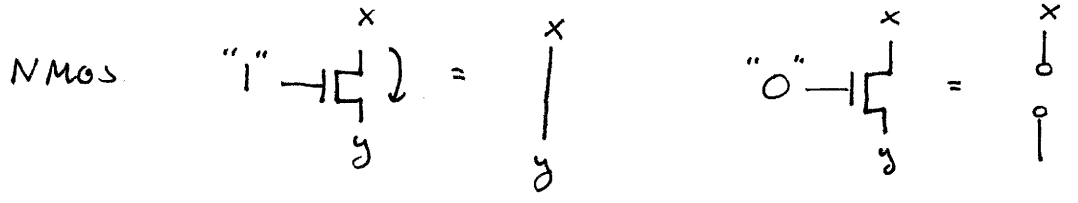
$$(121)_d = (1111001)_b.$$

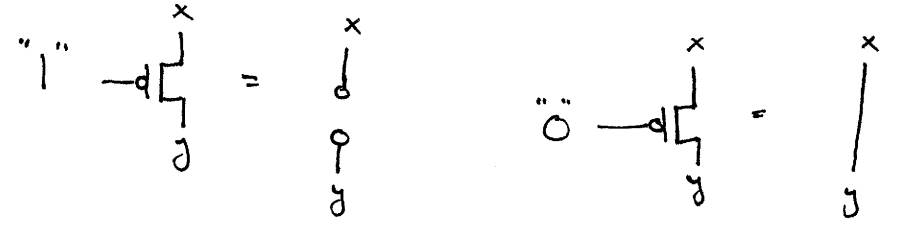
Digital Circuits

Truly "digital" circuits rely on Complimentary Metal Oxide Semiconductor or (CMOS) transistors, or MOS Field Effect Transistors (MOSFET)

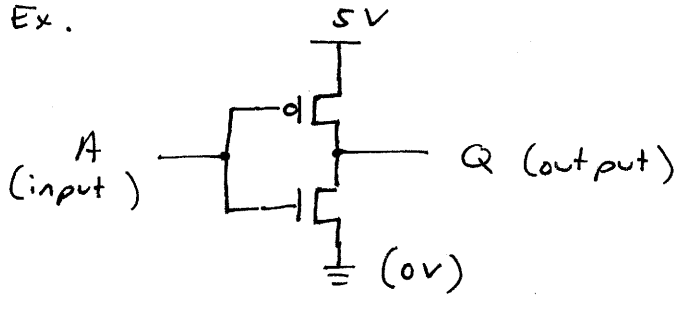
These devices are complicated, but in digital electronics we use them as switches - we slam them "on" or slam them "off".

Two "Complimentary Types"

NMOS 

PMOS 

We can list the output of a circuit for every possible input

Ex. 

A	Q
0	1 (5V)
1	0 (0V)

$Q = A'$

Some circuits are worth re-using over and over

"Inverter"

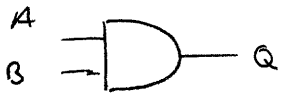
"NOT" Gate

Notes:  = copy or "buffer"

○ = invert

Power and ground are not drawn but still must be provided.

More useful are 2-input "gates" like the following:



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

"AND"

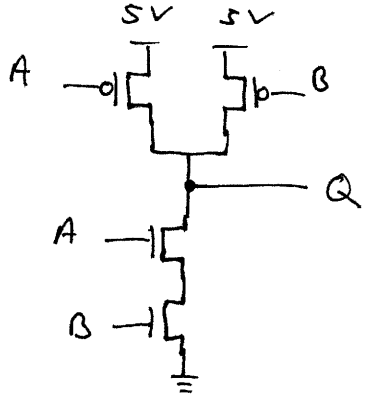
$Q = AB$
$= A \cdot B$
$= A$ AND B

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

"OR"

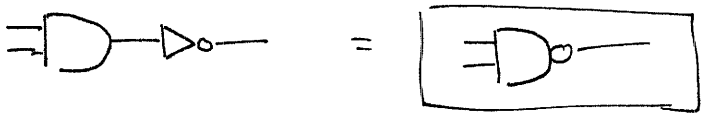
$Q = A + B$
$= A$ OR B

What is the behavior of the following circuit?



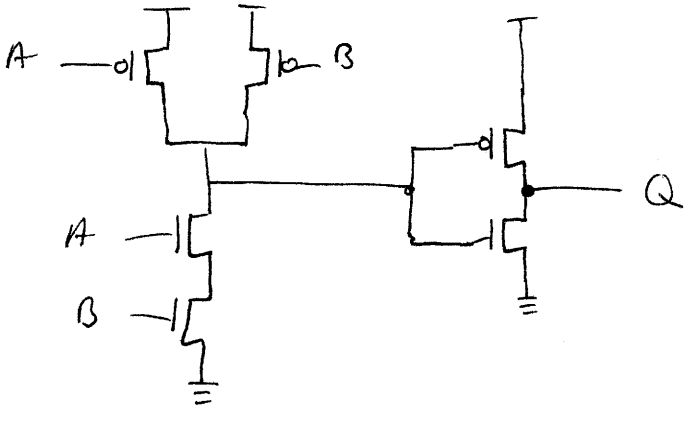
A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

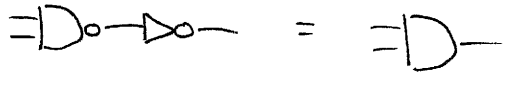
Note that this is just an inverted "AND", called "NAND"



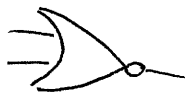
$Q = (AB)'$

It turns out NAND is easy to build, and AND is built as NAND followed by an inverter.





Bubbles cancel out

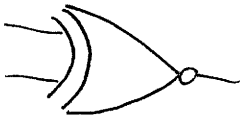


A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

"NOR" $Q = (A + B)'$

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0

"XOR" $Q = A \otimes B$



A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1

"XNOR" $Q = (A \otimes B)'$