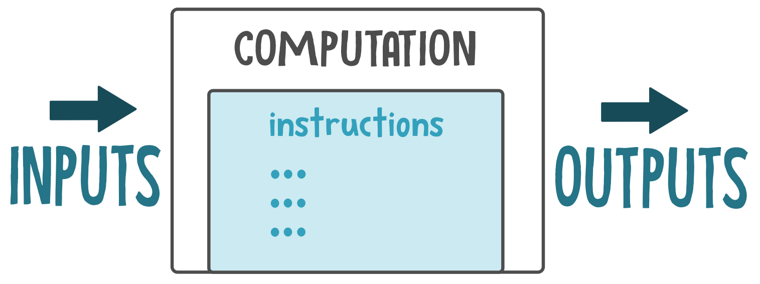
QC CLASSICAL COMPUTING LECTURE 1

HISTORY OF CLASSICAL COMPUTING

What is COMPUTATION

A mathematical calulation that maps inputs to an output based on a set of instructions

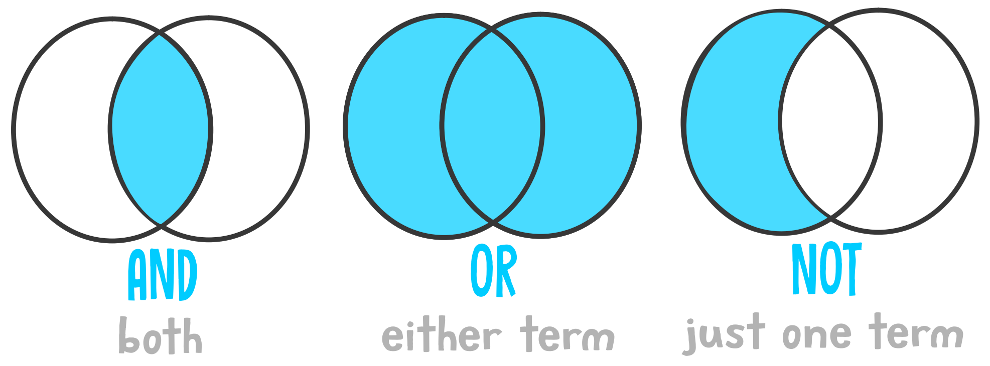


BIT-SIZED history of computing

- ★1939 Turing machine
- ★1946 the Eniac
- ★1949 the Modem
- ★1957 Fortran, Hard-drive, Ramac
- ★1961 the Mouse
- ★1968 RAM
- ★1969 the Arpanet
- ★1970 Mp 941
- ★1971 Intel 1001, Floppy
- ★1972 Pong, {C}
- ★1973 Ethernet cable
- ★1975 8800 Altair
- ★1977 the Apple 2
- ★1979 C++

BOOLEAN LOGIC

Boolean Logic: maps input bit(s) to output bit(s)



Logic gates + Truth tables

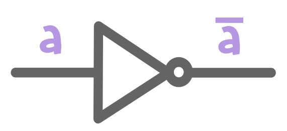
Logic: maps input bit(s) to output bit(s)

Tables: tells us the output of a logical operation

GATES 1 BIT

GATES: NOT

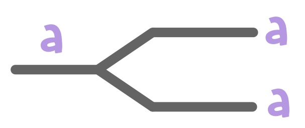
Flips the bit



INPUT	OUTPUT
0	1
1	0

GATES: FANOUT

Copies the bit

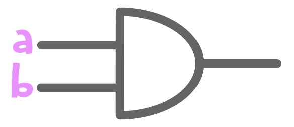


a	OUTPUT
0	00
1	11

GATES 2 BIT

GATES: AND

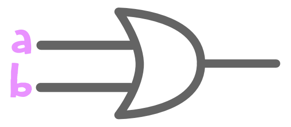
Outputs 1 if both are 1, outputs 0 otherwise



a	b	OUTPUT
0	0	0
0	1	0
1	0	0
1	1	1

GATES: OR

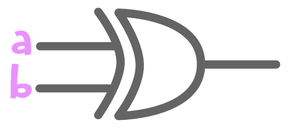
Outputs 1 if either of the inputs bits is 1 outputs 0 if neither of the inputs bits is 1



a	b	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	1

GATES: XOR

Outputs 1 if either input bits are 1, but not both outputs 0 if neither or both bits are 1



a	b	OUTPUT
0	0	0
0	1	1
1	0	1
1	1	0

BASE-REPRESENTATIONS

"Learning to think like a computer"

DECIMALS

- ★Decimal number system is based on numerical digits 0-9
- ★Base determines how numbers get represented and how we perform arithmetic operations

Example:

$$6 = 6 = (6\cdot10^0)$$

$$36 = 30+6 = (3\cdot10^1)+(6\cdot10^0)$$

$$536 = 500+30+6 = (5\cdot10^2)+(3\cdot10^1)+(6\cdot10^0)$$

BINARY

- ★We can describe any number with BITS

- ★Base-2 is one of the most important bases for performing computation
- ★It is binary, only 0 and 1
- ★Also reffered to as a BIT

- ★We can still do operations; all of the operations in a classical computer happen by manipulating BITS

Converting:

1010 → decimal

$$= (1\cdot10^3)+(0\cdot10^2)+(1\cdot10^1)+(0\cdot10^0)$$

1010 → binary

$$= (1\cdot2^3)+(0\cdot2^2)+(1\cdot2^1)+(0\cdot2^0)=8\ 2=10$$

BITS: ARITHMETIC OPERATIONS

"How computers compute"

BINARY ADDITION

- ★Similar to the decimal we are used to
- ★BITS carry over when the sum becomes larger than 2

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

MULTIPLYING BITS

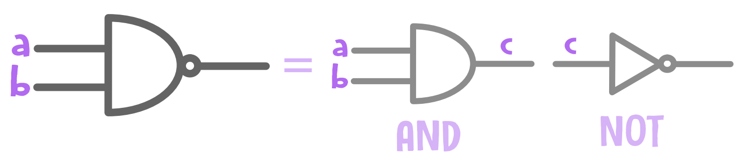
- ★It is the same as binary multiplication
- ★It's like "normal" (the decimal one)

$$0\cdot0=0$$
$$0\cdot1=0$$
$$1\cdot0=0$$
$$1\cdot1=1$$

UNIVERSALITY

Any computation operation can be made by using a combination of {NOT, AND, OR, FANOUT}

GATES: NAND (NOT + AND)



a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

c	OUT
0	1
1	0

a	b	OUT
0	0	1
0	1	1
1	0	1
1	1	0

"Let's make a binary adder with the gates we have" a+b=

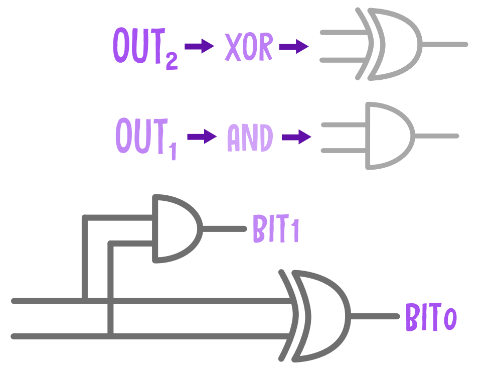
a	b	OUT_1	OUT_2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

a	b	BIT_0
0	0	0
0	1	1
1	0	1
1	1	0

OUT_2

a	b	BIT_1
0	0	0
0	1	0
1	0	0
1	1	1

OUT_1



REVERSIBILITY

Given the output of a gate, we can determine what the inputs are

REVERSIBLE GATE preserves all the information

NON-REVERSIBLE GATE loses some information

QUBIT X QUBIT