KANAZAWA UNIVERSITY | Emerging Media Initiative

GSCI1801A

Information Science

Lecture 1: Computer and Architecture, and Basic Operations in Computer Science

Asst. Prof. Chawanat NAKASAN | 2021-10-05

Agenda

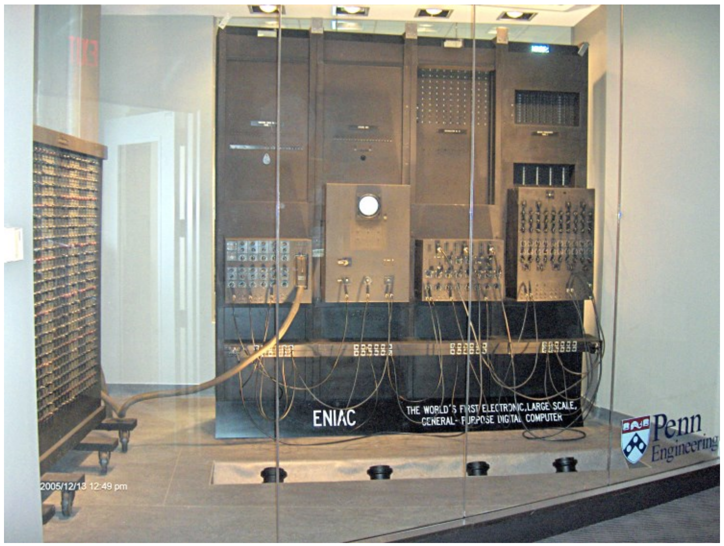
- What’s a computer?
- What’s inside a computer?
  - Explanation of various components
- Computer Architecture
- Binary Number System
- Arithmetic Operations
- Logical Operations

What's a computer?

- a machine that can be programmed to manipulate symbols. Its principal characteristics are:
  - It responds to a specific set of instructions in a well-defined manner.
  - It can execute a prerecorded list of instructions (a program).
  - It can quickly store and retrieve large amounts of data.

https://www.cs.cmu.edu/~fgandon/lecture/uk1999/computers_types/

How fast is a computer?

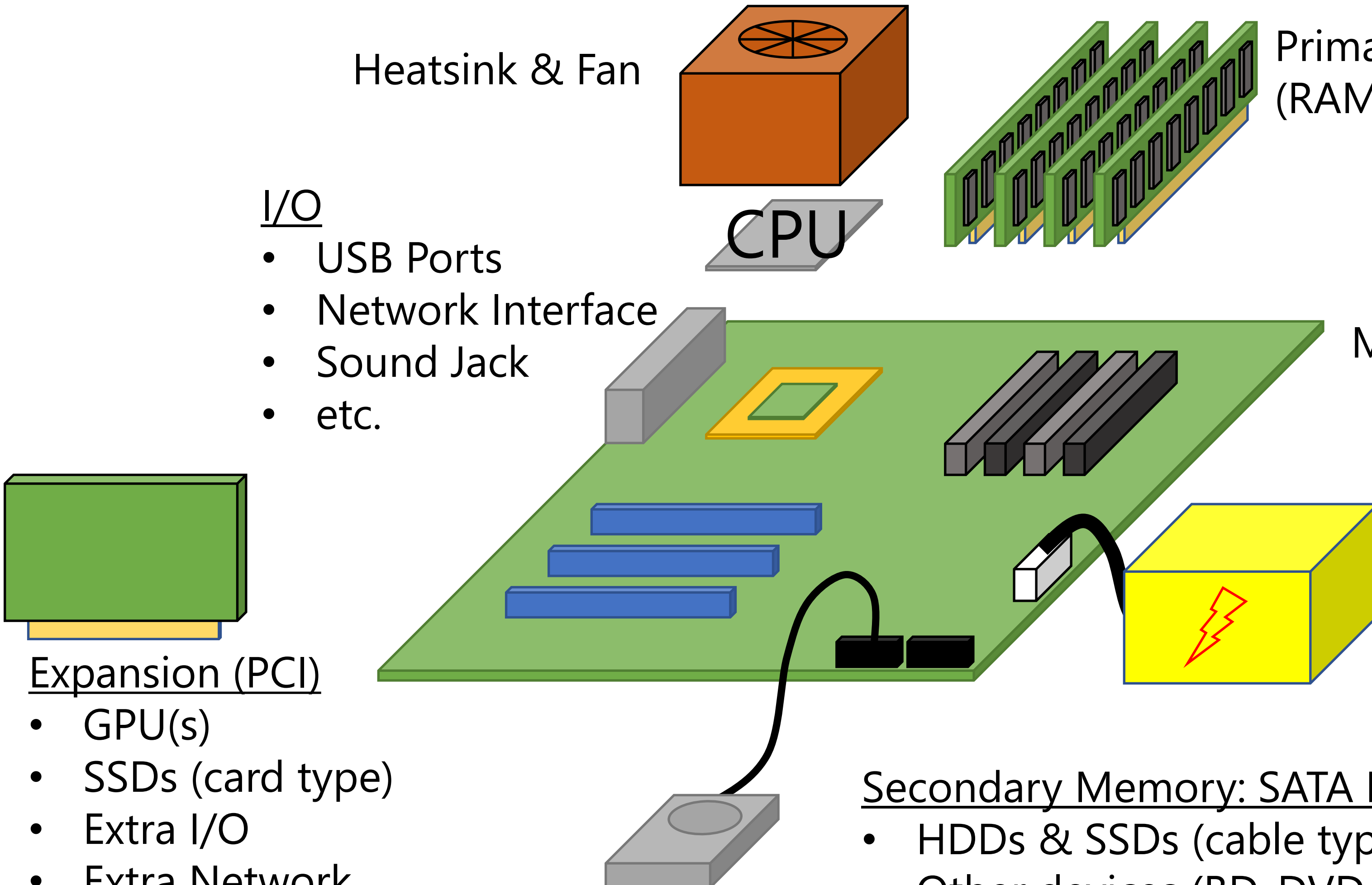


[ENIAC, 1945](#) (USA)
Speed: 5,000 cycles per second

[ABCI, 2018](#) (Japan)
Speed: 37 PFLOPS (high precision real numbers)

In less than a century, world-leading computers got faster by at least 7 billion-fold.

What's in a (typical) computer?



<u>I/O</u>
- USB Ports
- Network Interface
- Sound Jack
- etc.

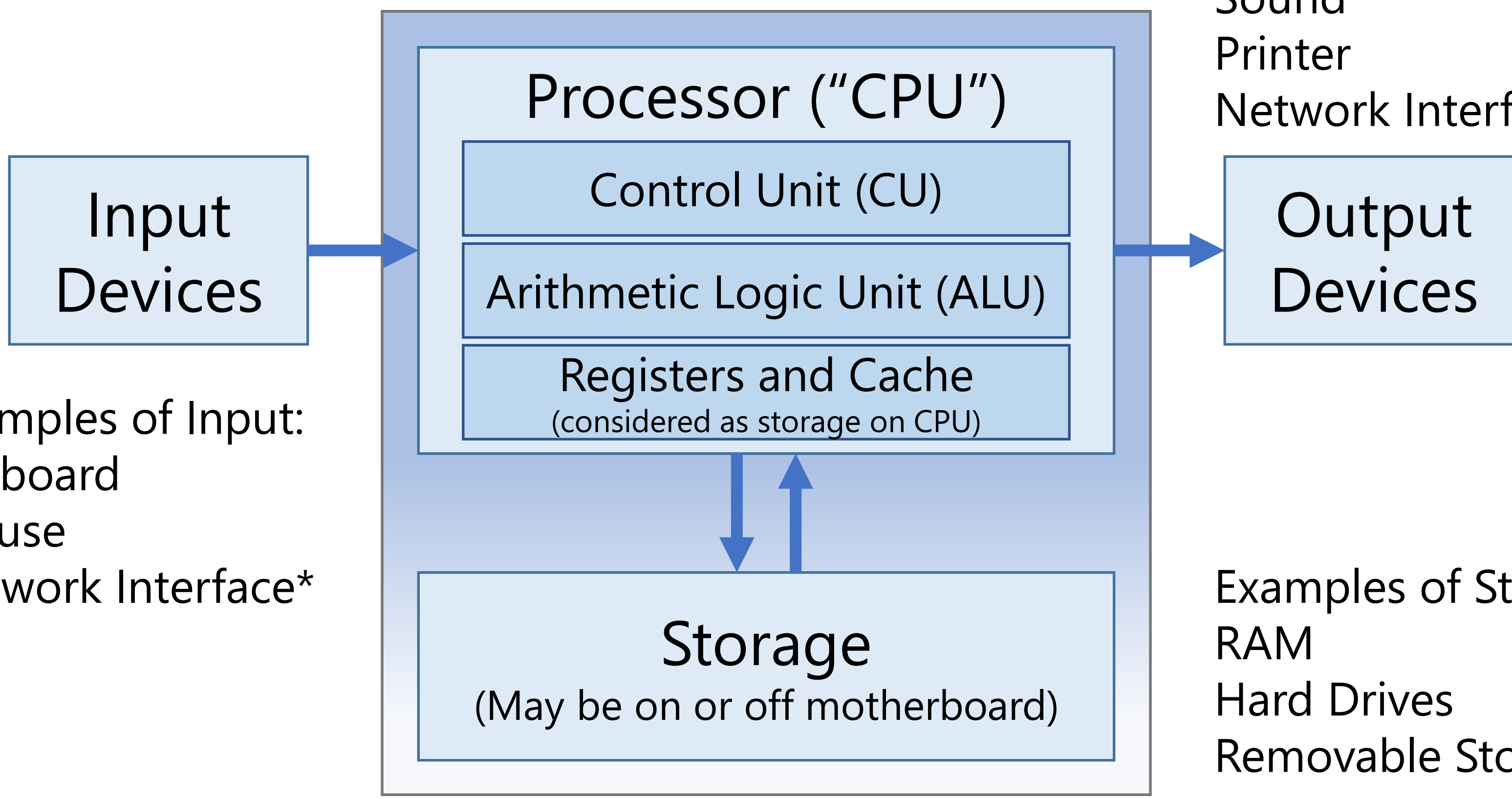
<u>Expansion (PCI)</u>
- GPU(s)
- SSDs (card type)
- Extra I/O
- Extra Network

<u>Secondary Memory: SATA Ports</u>
- HDDs & SSDs (cable type)
- Other devices (BD, DVD, …)

Basic Computer Architecture

[[von Neumann Architecture]]

Examples of Output:
Display
Sound
Printer
Network Interface*

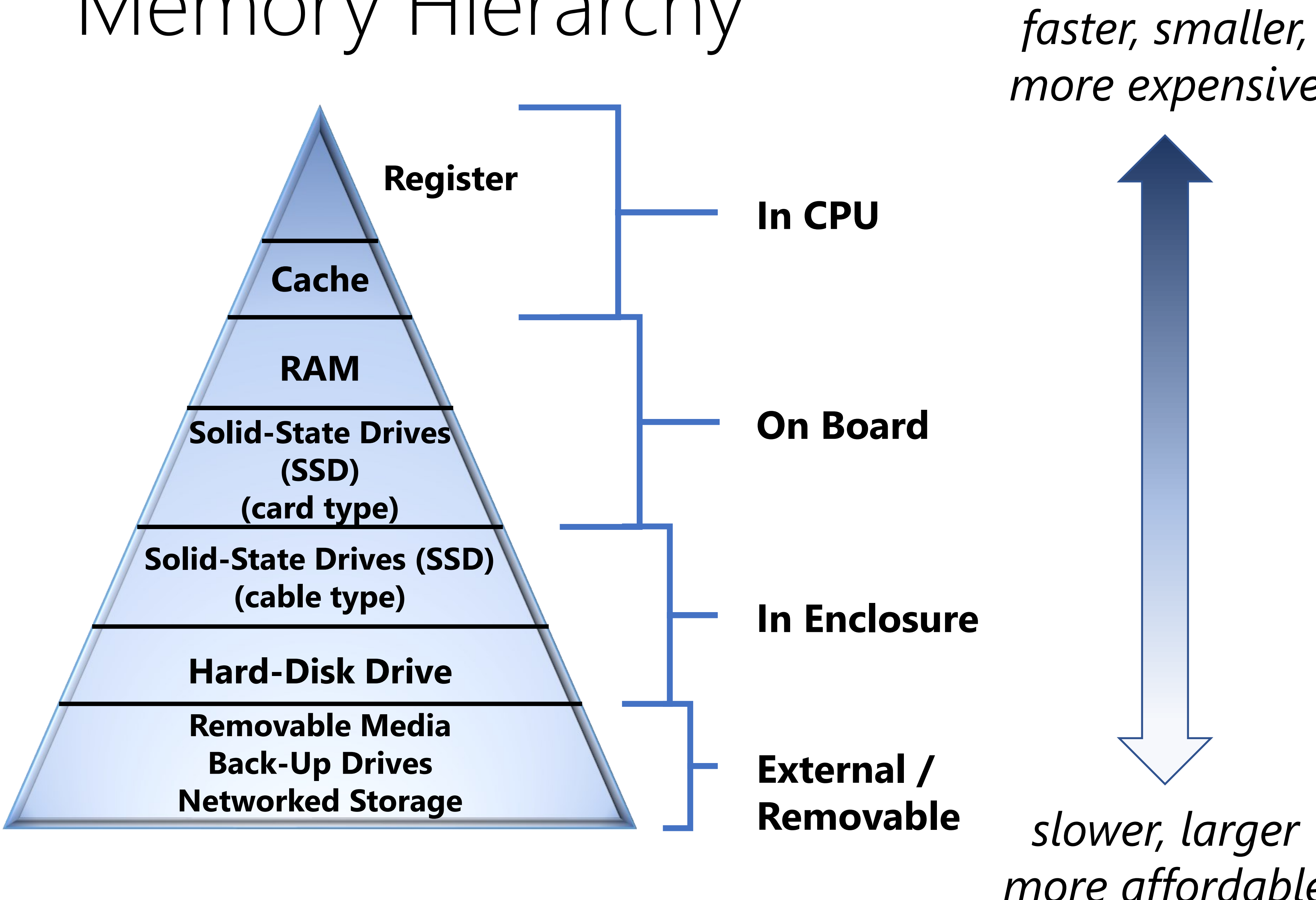


Examples of Input:
Keyboard
Mouse
Network Interface*

Examples of Storage:
RAM
Hard Drives
Removable Storage

*Network Interface is a funny oddball and will be *greatly* explained later!

Memory Hierarchy



Binary number system

Binary Number System

- Conversion from/to Decimal
- Representing negative numbers
- Why it’s easier than decimal?
- Binary systems in the nature

What is decimal number?

4,482,692,653

4,000,000,000	4×10^9
400,000,000	4×10^8
80,000,000	8×10^7
2,000,000	2×10^6
600,000	6×10^5
90,000	9×10^4
2,000	2×10^3
600	6×10^2
50	5×10^1
3	3×10^0

Now, what is binary number?

11101010

>> This part is decimal

1 0 0 0 0 0 0 0	$1 \times 2^7 =$	128
1 0 0 0 0 0 0	$1 \times 2^6 =$	64
1 0 0 0 0 0	$1 \times 2^5 =$	32
0 0 0 0 0	$0 \times 2^4 =$	0
1 0 0 0	$1 \times 2^3 =$	8
0 0 0	$0 \times 2^2 =$	0
1 0	$1 \times 2^1 =$	2
0	$0 \times 2^0 =$	0

234

Indicating Numbers by Bases

Using Subscripts

10001101_2 | 10001101_{10}

0b10001101 | 10001101

Programming Style

Binary Numbers ("bin") | Decimal Numbers ("dec")

A Bit and a Byte

Bit is the smallest digital information possible.

This is a bit.

10001101_2

8 bits make a byte.

A lot more meaningful data in computers are in bytes, such as characters (1 or 2 bytes) and numbers (1-8 bytes).
Files are almost never stored by individual bits.

Conversion Methods

Binary to Decimal

- Position value method (earlier)

Decimal to Binary

- Position value method
- Short division method

Decimal to Binary by Short Division

$$
\begin{array}{r|r l}
 & & \text{Remainder} \\
2 & 223 & \downarrow \\
\hline
 & 112 & 1
\end{array}
$$

$223 = ?_2$

Decimal to Binary by Short Division

		Remainder
2	223	
2	112	1
	56	0

$$223 = ?_2$$

Please continue dividing until you reach 1.

The first student to show their work gets a **participation point**.

In-class: Raise your hand.
Online: Use “raise hand” function in WebEx.

Decimal to Binary by Short Division

$223 = ?_2$

10100001_2

		Remainder
2	223	
2	112	1
2	56	0
2	28	0
2	14	0
2	7	0
2	4	1
2	2	0
	1	

Read back upwards

Quick Detour
Binary's brothers: the oct and the hex

Binary Numbers ("bin")	Octal ("oct")	Hexadecimal ("hex")
10001101_2	215_8	$8C_{16}$
`0b10001101`	`0215`	`0x8C`

Quick Detour
Binary's brothers: the oct and the hex

- bin ⇔ hex and bin ⇔ oct conversion is very easy!

16 = 2^4 so one digit in oct = 4 digits in bin.

$$8C_{16}$$

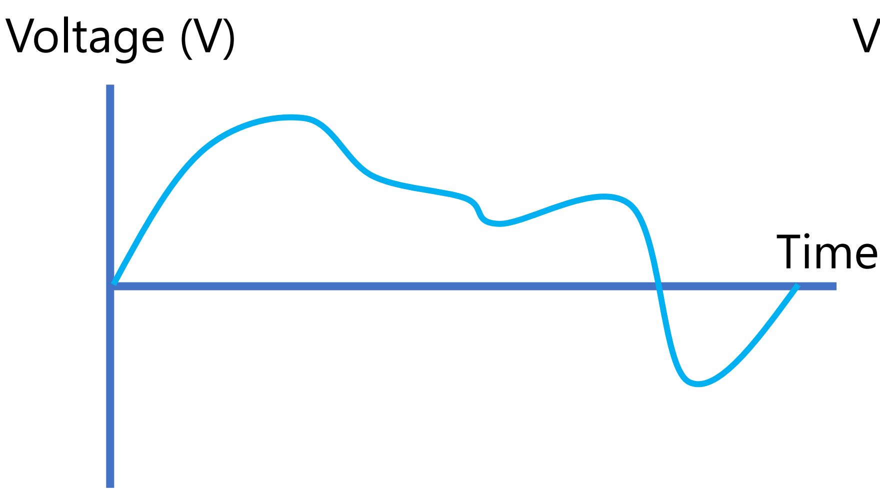
$$10001101_2$$

$$215_8$$

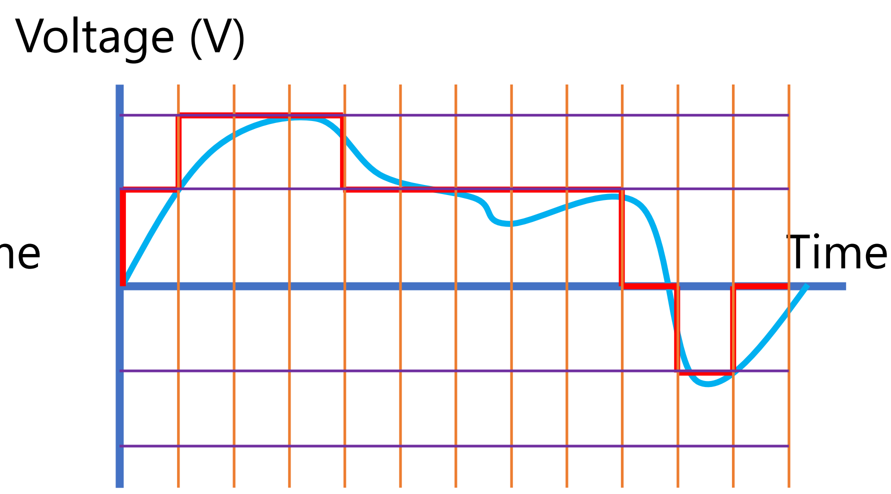
8 = 2^3 so one digit in oct = 3 digits in bin.

Analog and Digital …
where binary is used, a lot.

- Computers are actually trillions of *electronic switches* and *signals*.
- Electronic systems can work on either:



Analog System



Digital System

Conversion from A=>D involves *sampling* and *quantization*.

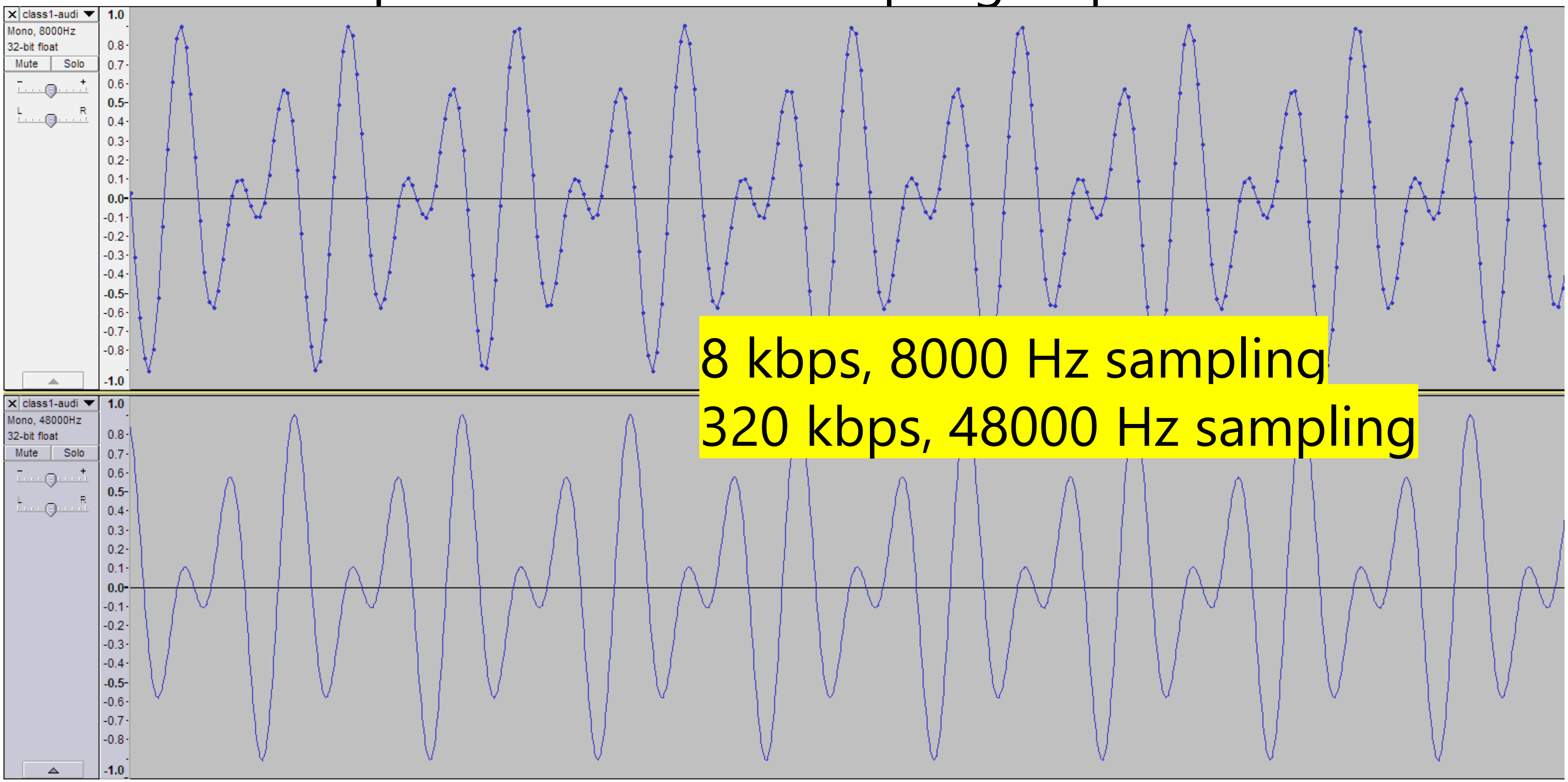
Discussion

Are digital computers “accurate”?

How do they work with natural information like noise and light?

Sampling Example

- Accurate quantization and sampling requires *more bits*.



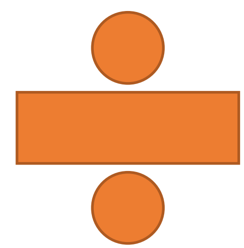
Software used: Audacity 2.1.2 © Audacity Team; Data: generated using this software.

Arithmetic Operations

- Addition
- Subtraction
- Multiplication
- Division and Modulo (same operation!)

Note: We'll discuss addition and multiplication today.



It doesn't end like you think …

Addition

$$36 + 29 = 100110_2 + 011101_2$$

$$\begin{array}{r} 100110_2 \\ +\,011101_2 \\ \hline _2 \\ \hline\hline \end{array}$$

Learn More:
https://www.sciencedirect.com/topics/computer-science/binary-addition

Two kinds of operations in adding

Carrying

$$
\begin{array}{r}
{}^{1}\phantom{{}_2} \\
1\,1_2 \\
+\,0\,1_2 \\
\hline
0_2 \\
\hline\hline
\end{array}
$$

Adding

Discussion

Is binary arithmetic easier than decimal? Why?

Negative Numbers in Digital Logic: 2's Complement

- There are no minus signs in digital logic, so John von Neumann invented something cool: Two's Complement.

$$81 = 01010001_2$$

Flip all bits

$$10101110_2$$

Then add 1

$$10101111_2 \quad = -81 \ldots ?$$

For this to work, we need another zero bit up front.

2's Complement (cont.)
Now try it.

$$
\begin{array}{r}
01010001_2 \\
+\,10101111_2 \\
\hline
\cancel{1}00000000_2 \\
\hline\hline
\end{array}
$$

Participation Point! Can someone help me add this?

Disregard this bit. If we are working with 8-bit computer, it has no place.

Why computers do not really have subtraction?

- Addition = Add and Carry operations
- Subtraction (arithmetic) = Subtract and Borrow operations

- Subtraction (computer) =
  2's Complement and Addition {Add & Carry}!

Multiplication: Decimal

$$
\begin{array}{r l}
22 & \\
\times\ 34 & \\
\hline
88 & 22 \times 4 \\
66 & 22 \times 3 \\
\hline
748 & \\
\hline\hline
\end{array}
$$

Multiplication: Binary

Filtering Operation

Add Operation Σ

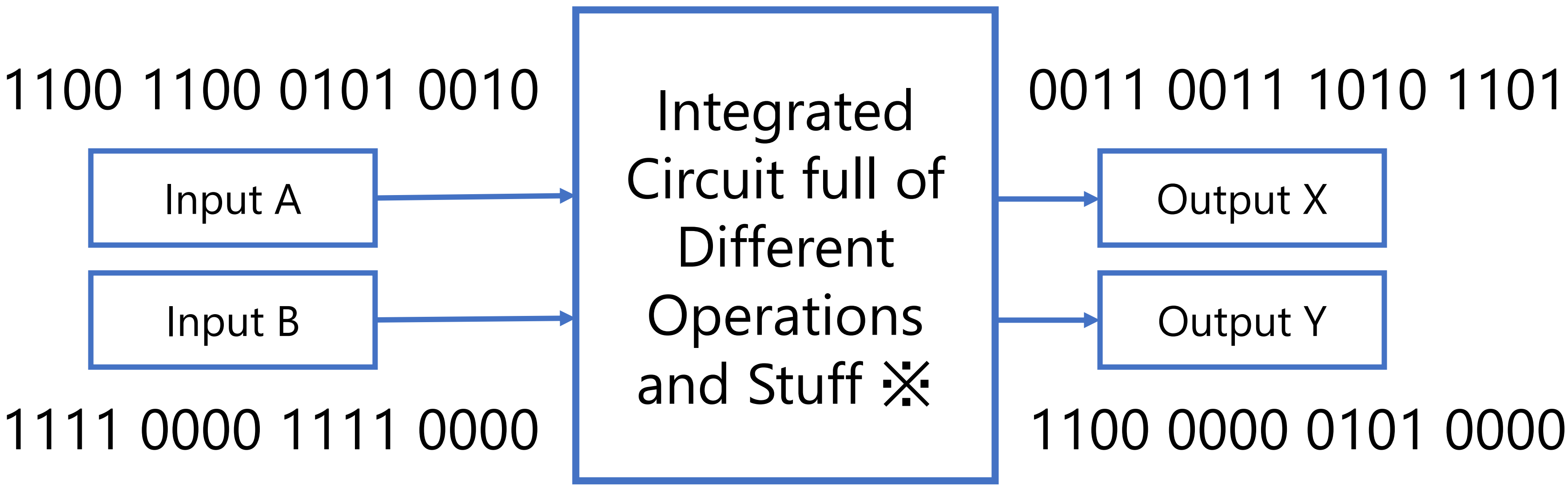
$$\begin{array}{r} 1001_2 \\ \times\ 1011_2 \\ \hline 1001 \\ 1001 \\ 0000 \\ 1001 \\ \hline 1100011_2 \\ \hline\hline \end{array}$$

Shifting Operation

EASIER than decimal multiplication!

Logical Operations

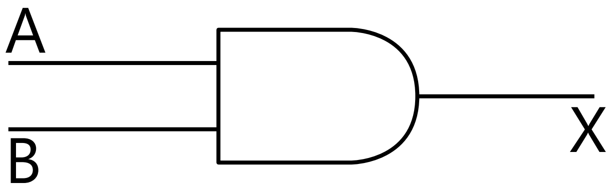
- Work on each bit at a time.
- Good for signal control and manipulation.



Bonus point!: What are Output X and Y, logically?

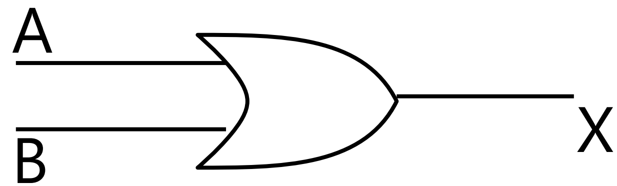
※ This is traditionally called a black box. Black in this case means “cannot see the inside.”

Logical Operators and Logic Gates



AND Gate

A	B	X = A ∧ B
0	0	0
0	1	0
1	0	0
1	1	1

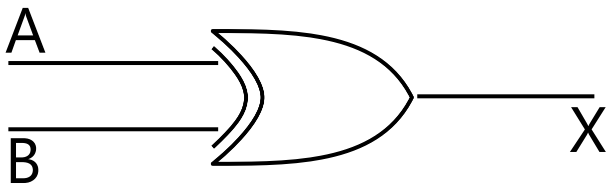


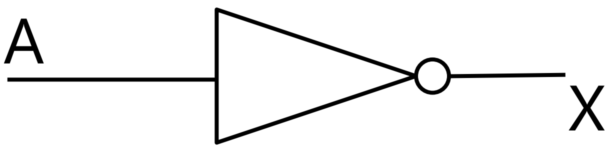
OR Gate

A	B	X = A ∨ B
0	0	0
0	1	1
1	0	1
1	1	1

※ You can also use "X = A AND B" and "X = A OR B" respectively.

Logical Operators and Logic Gates





XOR Gate

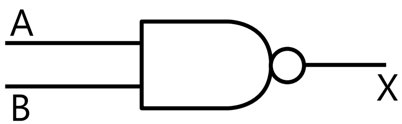
A	B	X = A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Inverter (“NOT”)

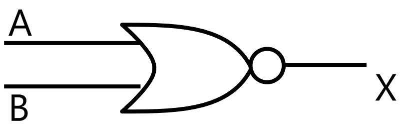
A	X = ¬A
0	1
1	0

NAND, NOR, and XNOR

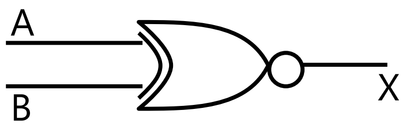
- They provide opposite results from AND, OR, XOR respectively.



NAND Gate



NOR Gate



XNOR Gate

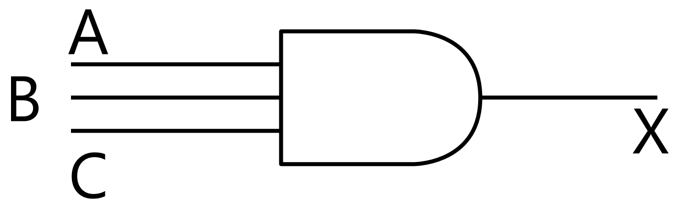
A	B	X = ¬(A ∧ B)
0	0	1
0	1	1
1	0	1
1	1	0

A	B	X = ¬(A ∨ B)
0	0	1
0	1	0
1	0	0
1	1	0

A	B	X = ¬(A ⊕ B)
0	0	1
0	1	0
1	0	0
1	1	1

Multiple-Input Gates

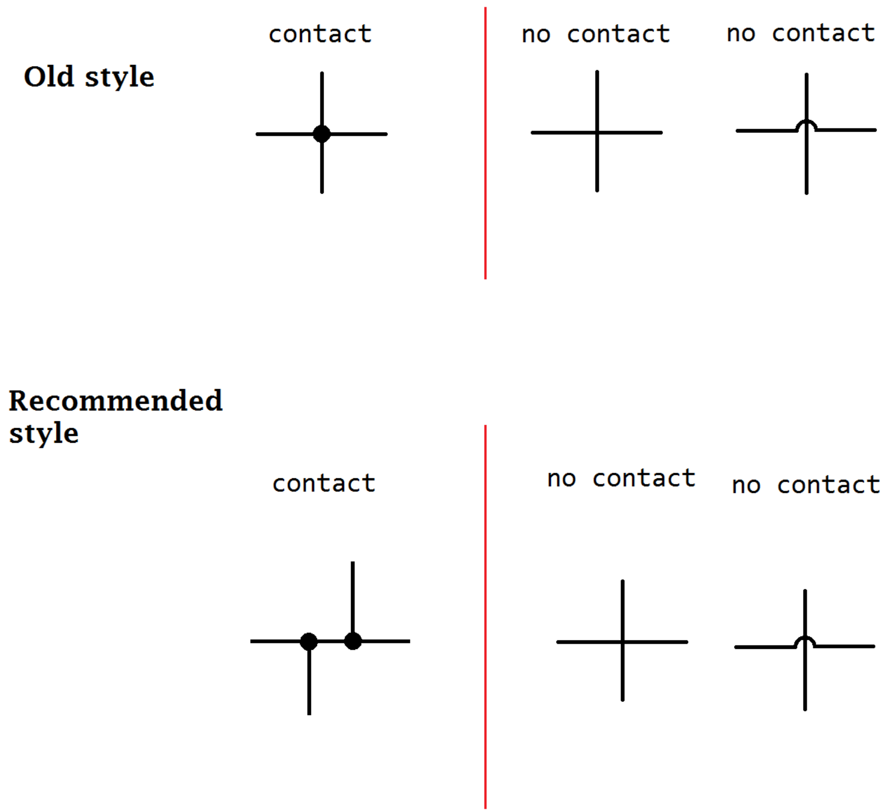
- Both logical math and logic gates can extend to more than two operands (where it makes sense).



A	B	C	X = A ∧ B ∧ C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

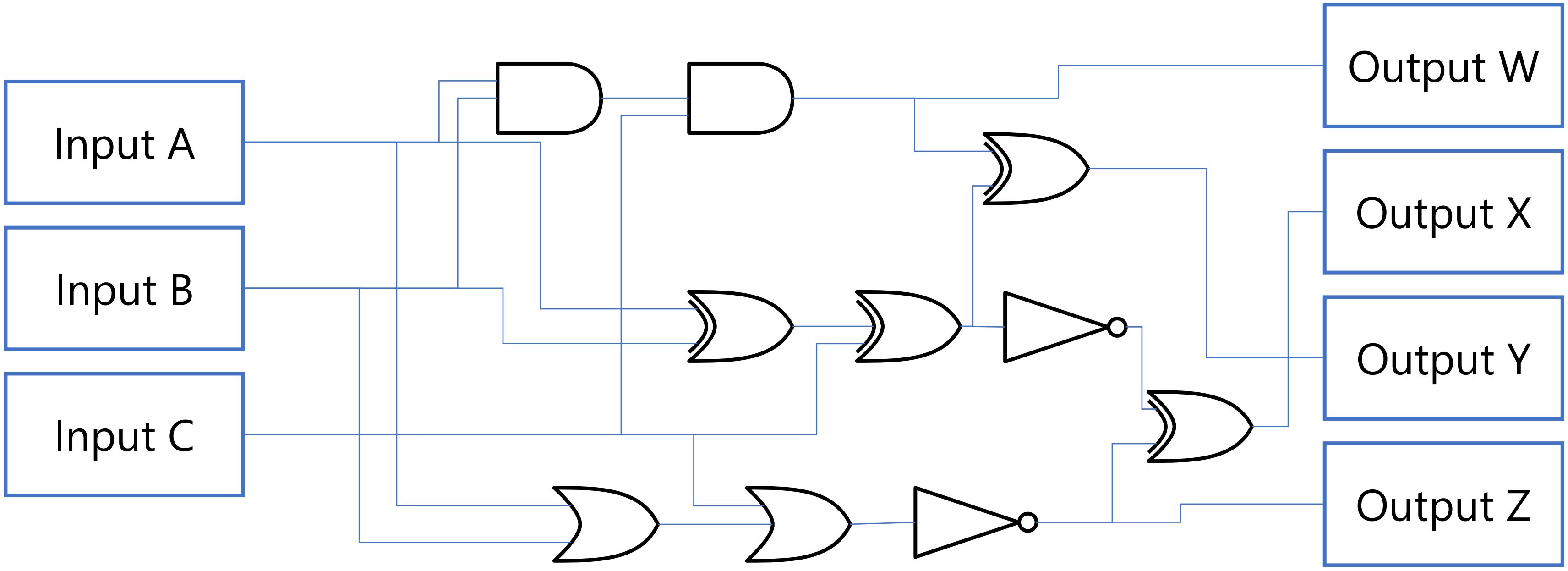
4-way crossovers in Electrical Diagrams

- This is how to correctly draw 4-way crossovers in electrical diagrams (and logic gate diagrams).
- This class follows either old or recommended style.
- Jumping is always better than no-dot crossing, but sometimes it is inconvenient in most drawing tools.
- (Three-way junctions should also have dots, but PPT can't draw them automatically.)



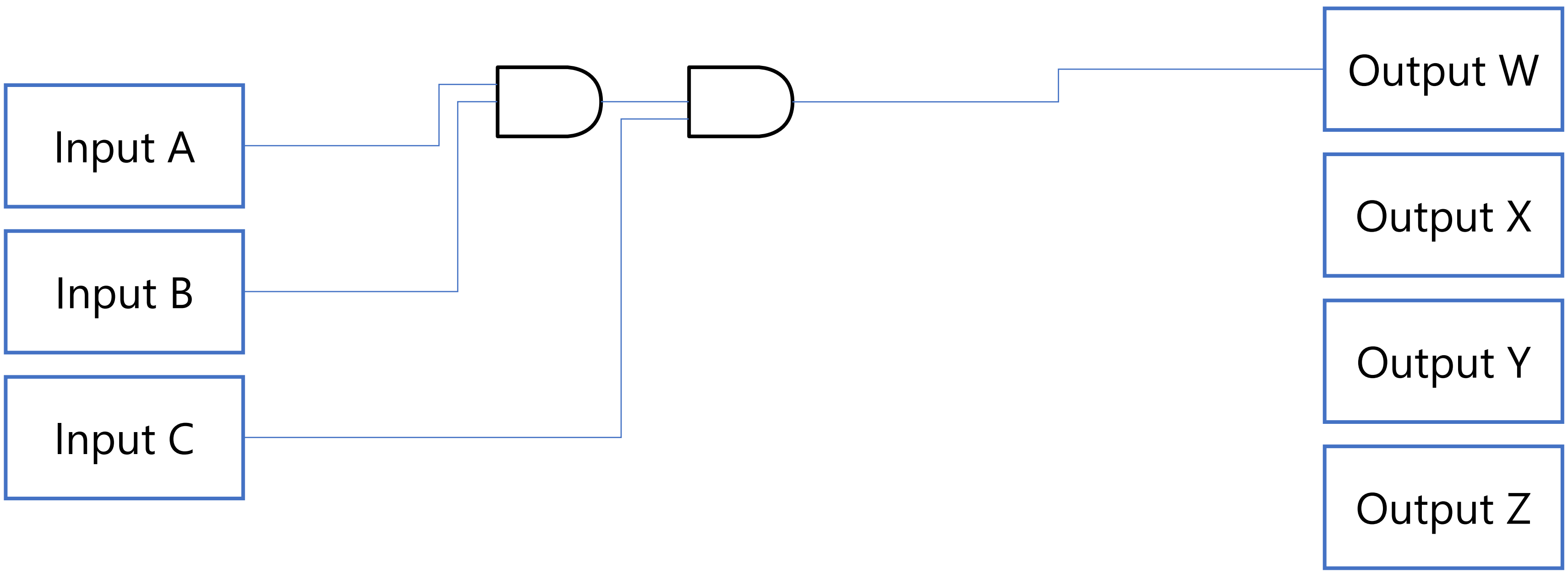
https://en.wikipedia.org/wiki/Circuit_diagram#/media/File:Wire_Crossover_Symbols_for_Circuit_Diagrams.png

Combining them together, we can achieve a lot of results.



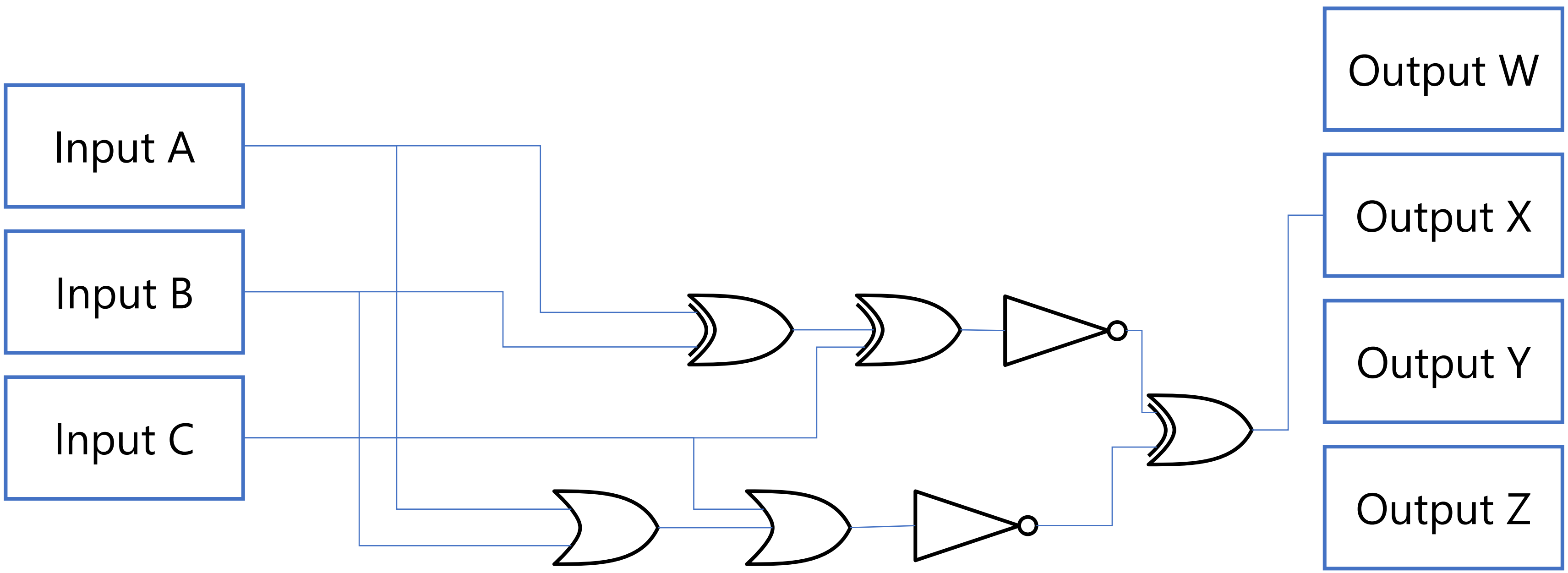
Bonus Point: What is this circuit?
Explain in human language!

Hint: Trace one output at a time.



$$W = (A \wedge B) \wedge C$$

Hint: Trace one output at a time.



Download the slide from LMS
and try later from your home ☺

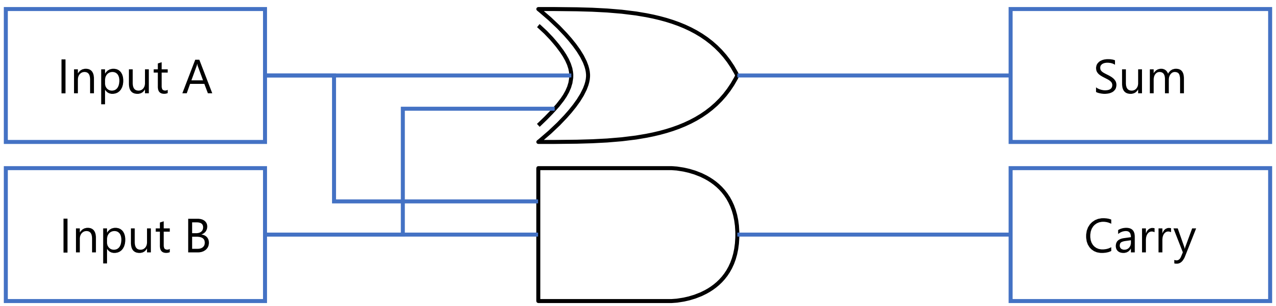
Logic Gate Simulation

- You can try various tools for logic gate simulation if you want to understand them better.

- We'll be using many kinds of simulators throughout the class, and you can also use them for your reports too. Get used to them ☺

- https://circuitverse.org/simulator
- https://academo.org/demos/logic-gate-simulator/

Logic to Arithmetic: Half Adders



A	B	C = A ∧ B	S = A ⊕ B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

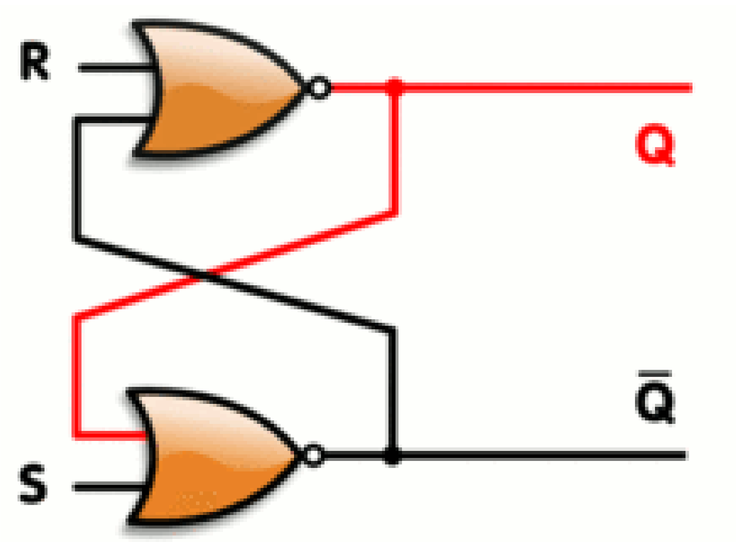
Carrying

$$\begin{array}{r} {}^{1} \\ 11_2 \\ +\,01_2 \\ \hline 0_2 \\ \hline\hline \end{array}$$

Adding

Circuit contains both adding and carrying!

Logic to Operations: SR Latch



S (Set)	R (Reset)	Q	$\bar{Q}$
0	0	Keep State	
1	0	1	0
0	1	0	1
1	1	Invalid	

※ The official KU textbook uses the term "RS Flip-Flop" which is the same.
The difference is that the textbook illustrates a NAND SR Latch. Here we use NOR SR Latch.
They both achieve same results.

Wait! Is that memory?

A Little Bit!

We just (re)discovered 1 bit of memory!

- It took humanity about 300 years from the first definition of electricity by William Gilbert to creating the first flip-flop.
- (A modern smartphone with 256 GB memory contains about 2,048,000,000,000 bits.)

In-Class Activity

Create a circuit diagram and a truth table for the following expression:

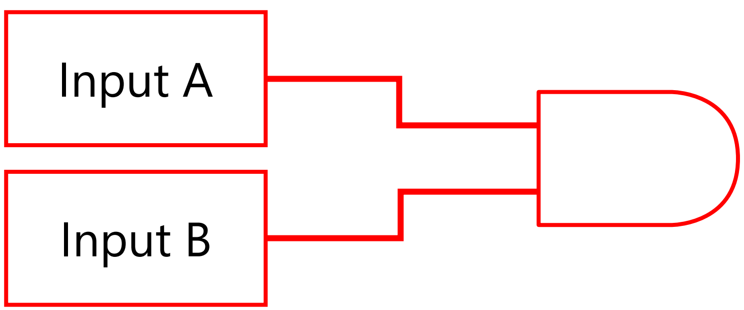
$$X = (A \wedge B) \vee C$$

In-Class Activity

$$X = (A \land B) \lor C$$

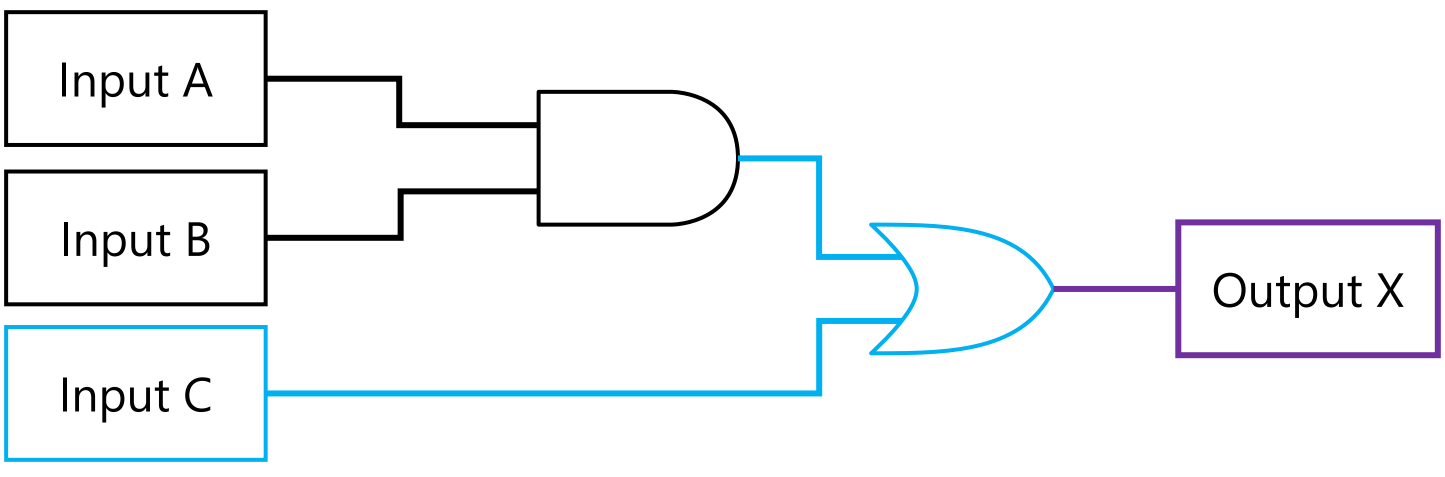
In-Class Activity

$$X = (A \wedge B) \vee C$$



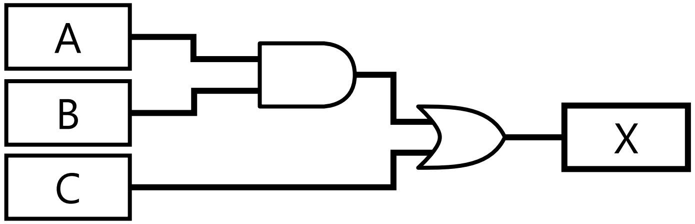
In-Class Activity

$$X = (A \wedge B) \vee C$$



In-Class Activity

$$X = (A \wedge B) \vee C$$



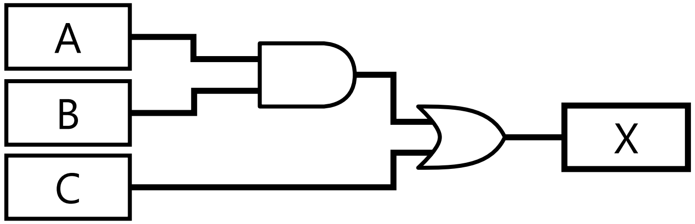
A	B	C	A ∧ B	(A ∧ B) ∨ C
0				
0				
0				
0				
1				
1				
1				
1				

Copy Here

Paste Here

In-Class Activity

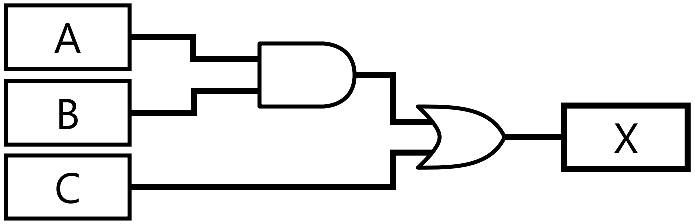
$$X = (A \wedge B) \vee C$$



A	B	C	A ∧ B	(A ∧ B) ∨ C
0	0			
0	0			
0	1			
0	1			
1	0			
1	0			
1	1			
1	1			

In-Class Activity

$$X = (A \wedge B) \vee C$$

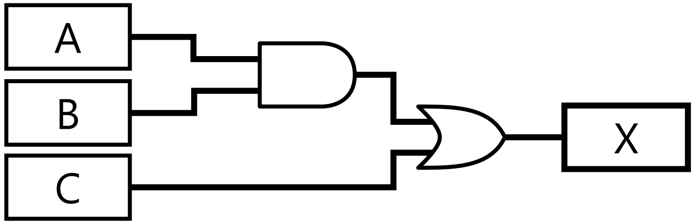


A	B	C	A ∧ B	(A ∧ B) ∨ C
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

In-Class Activity

If C is 1, then X is immediately 1.

$$X = (A \land B) \lor C$$

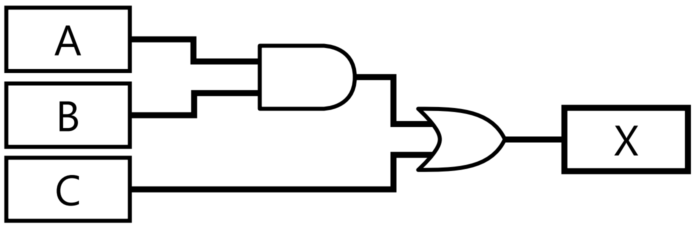


A	B	C	A ∧ B	(A ∧ B) ∨ C
0	0	0		
0	0	1		1
0	1	0		
0	1	1		1
1	0	0		
1	0	1		1
1	1	0		
1	1	1		1

In-Class Activity

Fill in the logic
for A AND B.

$$X = (A \land B) \lor C$$

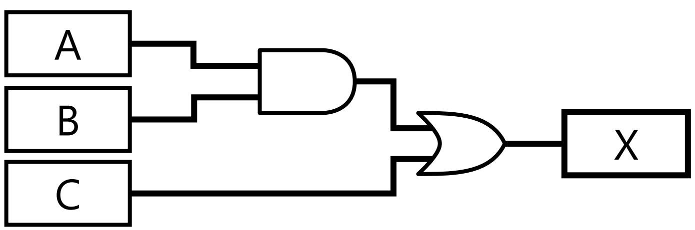


A	B	C	A ∧ B	(A ∧ B) ∨ C
0	0	0	0	
0	0	1	0	1
0	1	0	0	
0	1	1	0	1
1	0	0	0	
1	0	1	0	1
1	1	0	1	
1	1	1	1	1

In-Class Activity

Compute the rest of the logic.

$$X = (A \wedge B) \vee C$$

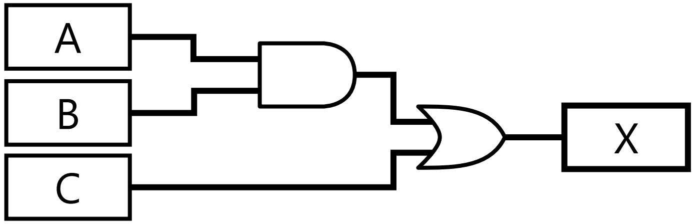


A	B	C	A ∧ B	(A ∧ B) ∨ C
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

In-Class Activity

Done.

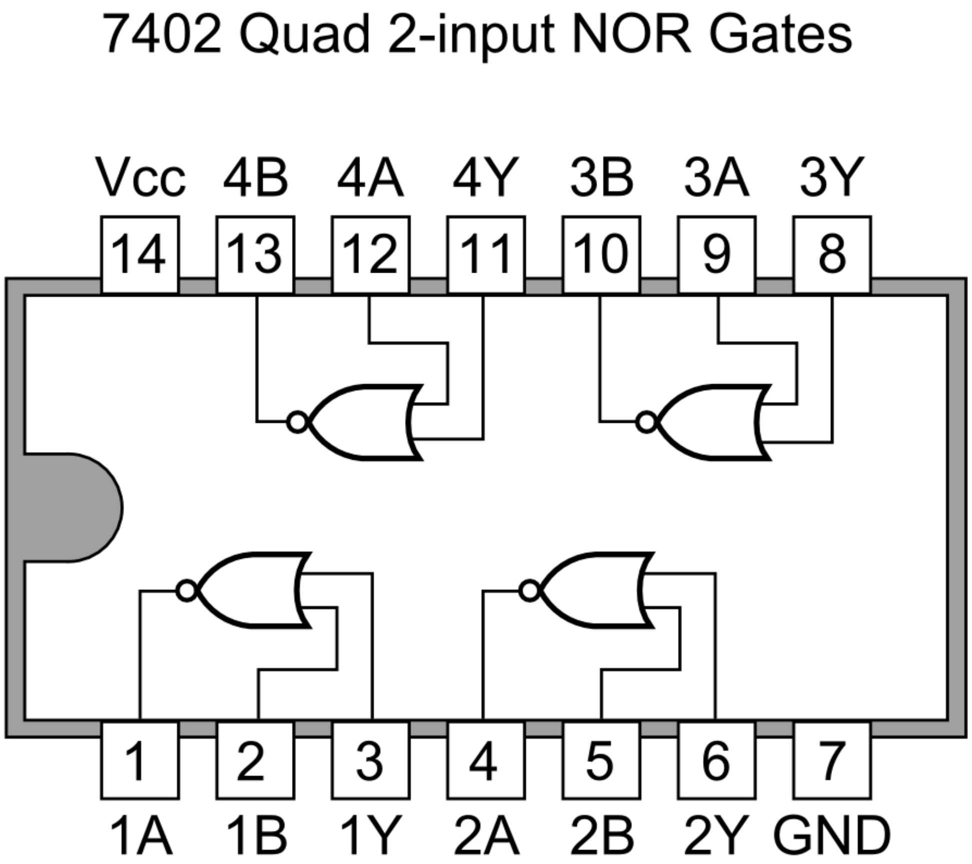
$$X = (A \wedge B) \vee C$$



A	B	C	A ∧ B	(A ∧ B) ∨ C
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

Logic gates in ICs

- This is the 7400 IC.
- You may notice the dot at the end of OR. This is the NOR gate.
- There are hundreds of millions of these gates in a CPU.



Binary in the Nature

- There is one important that work “similar” to binary. It’s actually our DNA.

Standard genetic code

1st base	2nd base: T		2nd base: C		2nd base: A		2nd base: G		3rd base
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	T
	TTC		TCC		TAC		TGC		C
	TTA	(Leu/L) Leucine	TCA		TAA	Stop (*Ochre*)[B]	TGA	Stop (*Opal*)[B]	A
	TTG[A]		TCG		TAG	Stop (*Amber*)[B]	TGG	(Trp/W) Tryptophan	G
C	CTT		CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	T
	CTC		CCC		CAC		CGC		C
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CTG[A]		CCG		CAG		CGG		G
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	T
	ATC		ACC		AAC		AGC		C
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	T
	GTC		GCC		GAC		GGC		C
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GTG		GCG		GAG		GGG		G

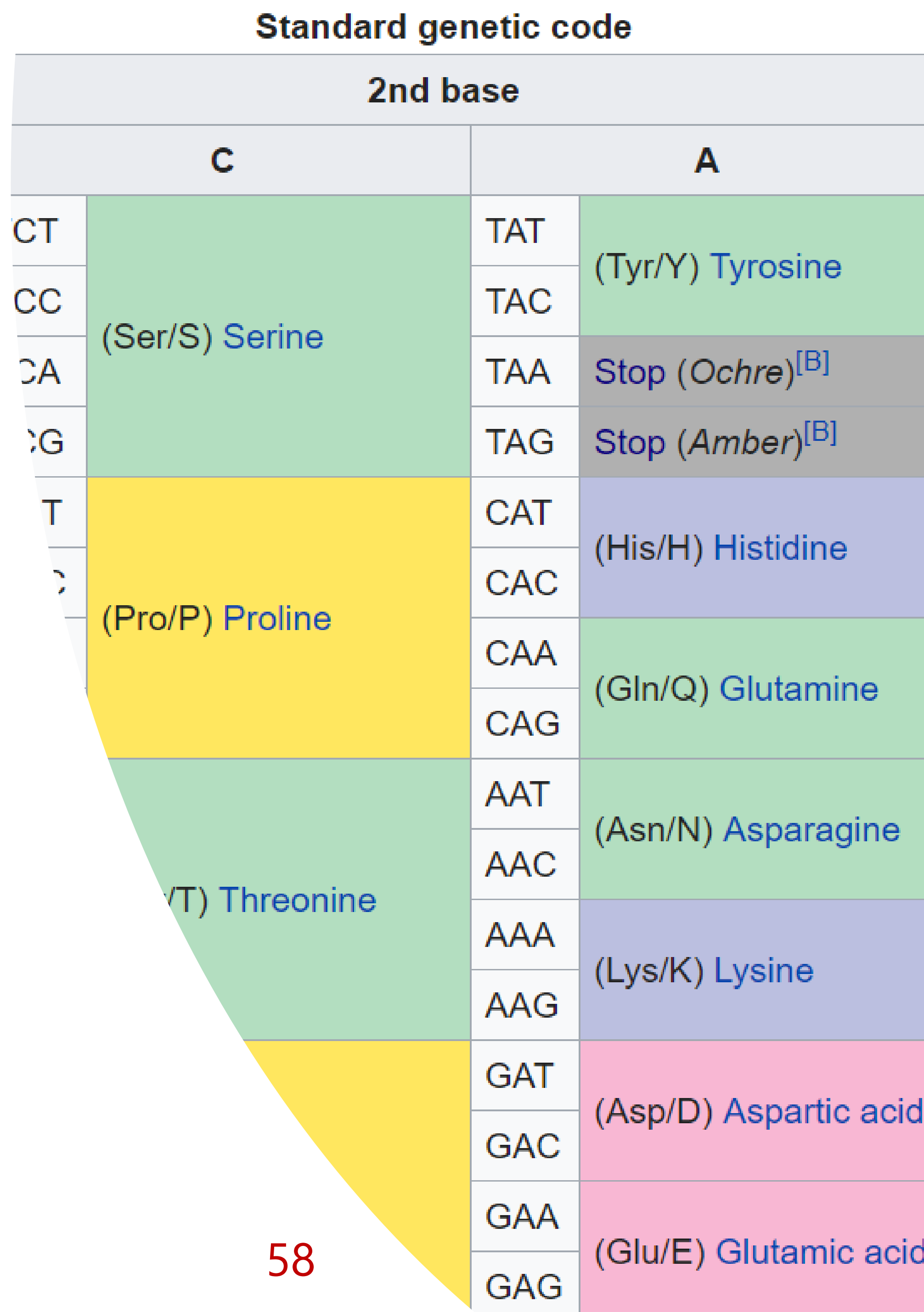
https://en.wikipedia.org/wiki/DNA_codon_table

Wait, what?

- There are 4 bases, right? We can try to represent:

DNA Base	Binary
T	00
C	01
A	10
G	11

Note: This is a simplification.



Store DNA information in binary!

※
atgagagcggtccatctccttgcggtgagc

30 bases

DNA Base	Binary
T	00
C	01
A	10
G	11

60 bits = ~ 8 bytes

100011 101110
110111 110001
011000 010001
010000 110111
110011 101101

Human Genome = 3 BILLION bases => 750 Megabytes PER PERSON AT LEAST. (Just the data)

NOTE AGAIN: This is just a sample hypothetical case!

Download human genome at https://www.gencodegenes.org/human/
(FASTA file for ALL regions = 800+ MB compressed)

And, store binary information in DNA …

A DNA-Based Archival Storage System

James Bornholt[†] Randolph Lopez[†] Douglas M. Carmean[‡]
Luis Ceze[†] Georg Seelig[†] Karin Strauss[‡]
† University of Washington ‡ Microsoft Research

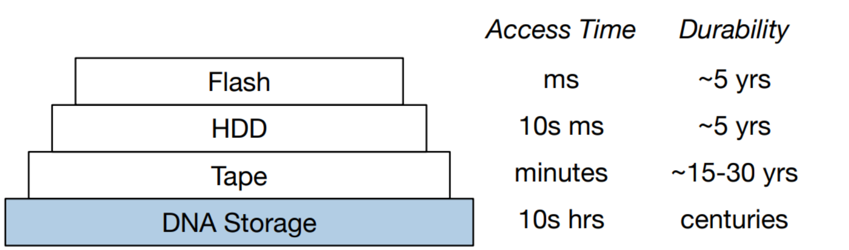


Figure 2. DNA storage as the bottom level of the storage hierarchy

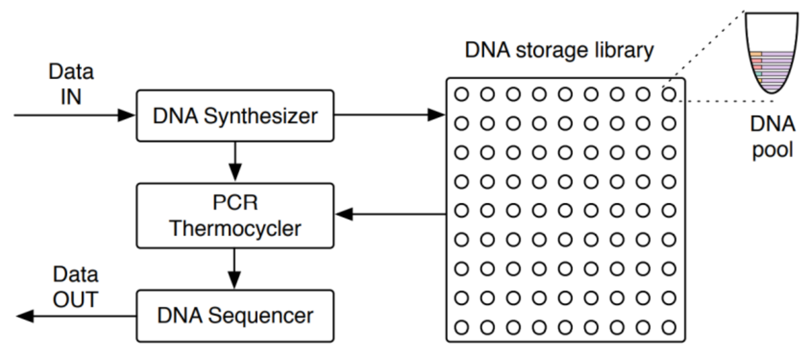


Figure 3. Overview of a DNA storage system.

Discussion

How can we represent other things in binary form?

What do you think about using DNA as data storage?