Computer Systems 2

National Workshop 4 – Session 3

By the end of this session participants will have:

- gained a deeper understanding of computer systems concepts and terminology including logic gates and circuits, truth tables, basic electronics, computer components, von Neumann architecture and the fetch-execute cycle
- experienced inquiry-based learning (IBL) and in doing so appreciate the benefits of IBL
- taken part in and reflected upon an half-adder activity
- acquired additional knowledge and ideas on how they will facilitate the learning of computer systems in their own classrooms

Context

Strand 1: Practices and principles	Strand 2: Core concepts	Strand 3: Computer science in practice
Computers and society	Abstraction	Applied learning task 1 - Interactive information systems
Computational thinking	Algorithms	Applied learning task 2 - Analytics
Design and development	Computer systems	Applied learning task 3 - Modelling and simulation
	Data	Applied learning task 4 - Embedded systems
	Evaluation/Testing	

"The core concepts are developed theoretically and applied practically. In this way, conceptual classroom-based learning is intertwined with experimental computer lab-based learning throughout the two years of the course."

NCCA Curriculum specification, Page 20

LCCS Learning Outcomes

S2: Computer systems	
CPU: ALU, Registers, Program counter, Memory	2.11 describe the different components within a computer and the function of those components
Basic electronics: voltage, current, resistors, capacitors, transistors	2.12 describe the different types of logic gates **and explain how they can be arranged into larger units to perform more complex tasks**
Operating system layers: Hardware, OS, Application, User	2.13 describe the rationale for using the binary number system in digital computing and how to convert between binary, hexadecimal and decimal
	2.14 describe the difference between digital and analogue input
Web infrastructure - Computer Network Protocols: HTTP, **TCP, IP, VOIP**	2.15 explain what is meant by the World Wide Web (WWW) and the Internet, including the client server model, hardware components **and communication protocols**

Layers of a Computing System

End-user

Application Software

System Software

Computer Components

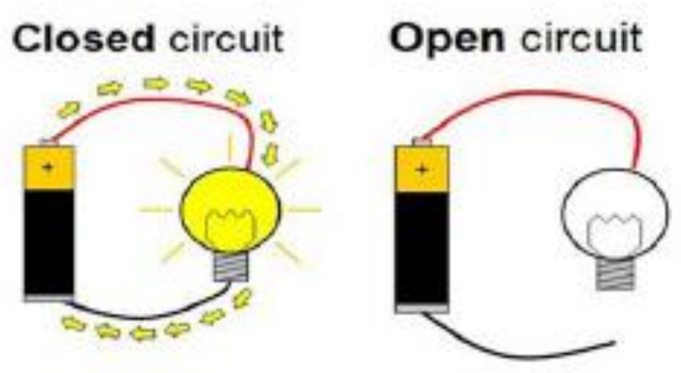
Electronics, Gates and Circuits

Data Representation

Logic Gates

Logic Gates

A gate is a device that performs a logical operation on electrical signals



These electrical signals are represented by bits (BInary digiTS) → 0 (0V) or 1 (5V)

The logical operations were defined by the mathematician George Boole (1815-64)

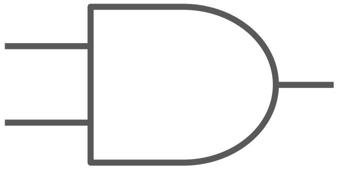
The most common logic (Boolean) operations are:

NOT	XOR
AND	NAND
OR	NOR

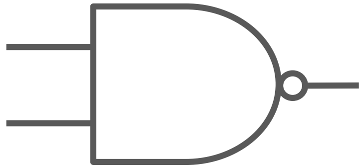
Logic Gates Symbols

Logic gates have one or more inputs and a single output

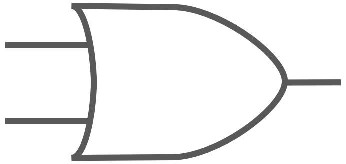
Each gate has its own logic symbol which allows circuits to be represented by a logic diagram



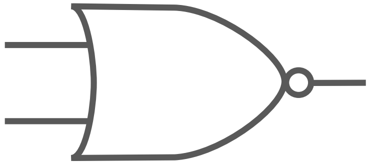
AND



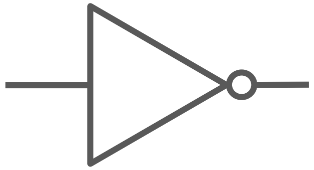
NAND



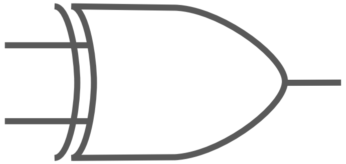
OR



NOR



NOT



Exclusive OR (XOR)

Logic Gates

The behaviour of gates (and circuits) are commonly represented in any of the following ways:

Operator	Example	Meaning
NOT	$\overline{A}$	NOT A
AND	A.B	A AND B
OR	A+B	A OR B

Boolean Expressions

Uses Boolean algebra, a mathematical notation for expressing two-valued logic

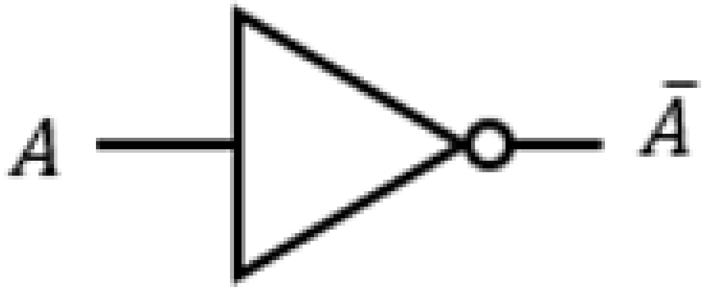
Truth Tables

A table showing all possible input values and the associated output values

Logic Diagrams

A graphical representation of a circuit; each gate has its own symbol

The NOT operation



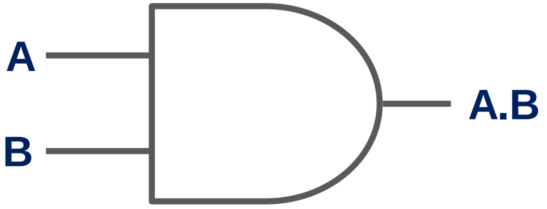
Logic Gate Symbol

A	$\bar{A}$
0	1
1	0

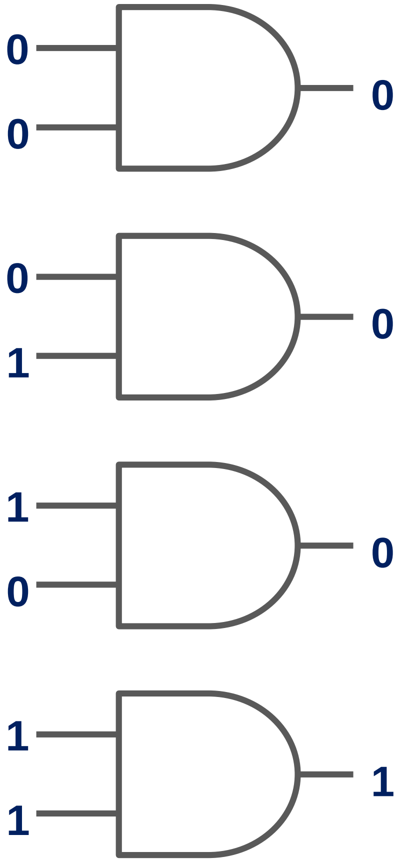
Truth Table

Inverts a single input. Also called an *inverter*.

The AND operation



Logic Gate Symbol

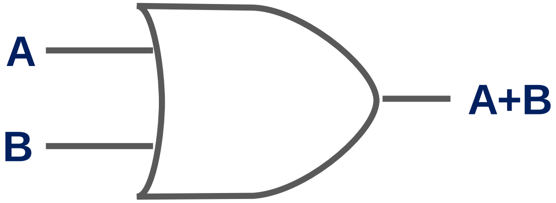


A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

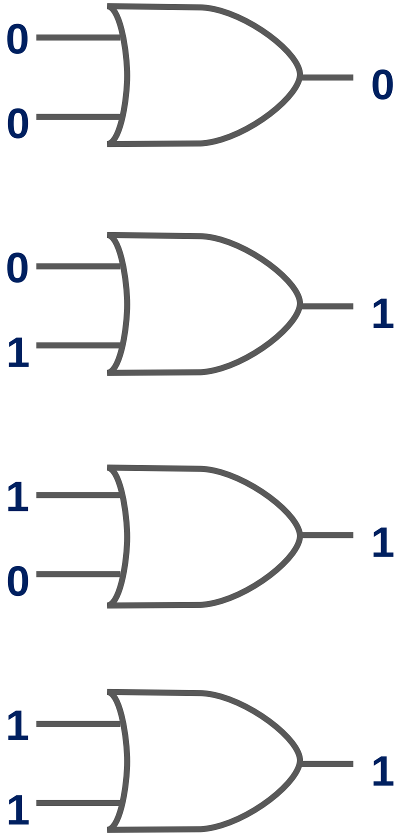
Truth Table

In order for the output to be 1 both inputs must be 1

The OR operation



Logic Gate Symbol

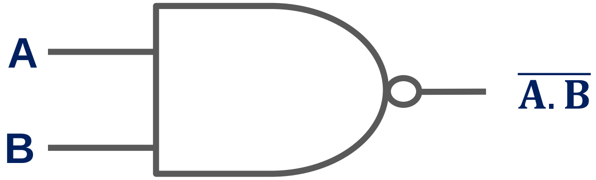


A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

In order for the output to be 1 either input must be 1

The NAND operation



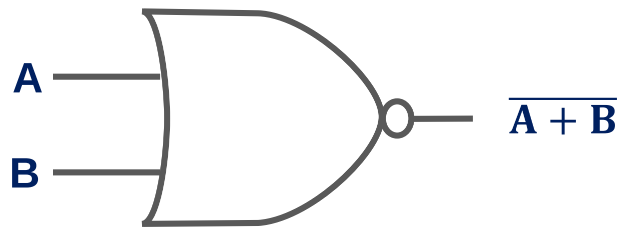
Logic Gate Symbol

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

The output is 1 if either input is 0

The NOR operation



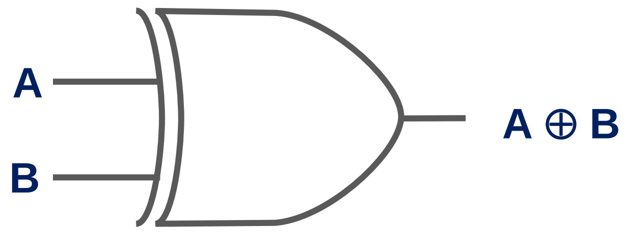
Logic Gate Symbol

A	B	$\overline{A + B}$
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table

The output is 1 if both inputs are 0

The XOR operation



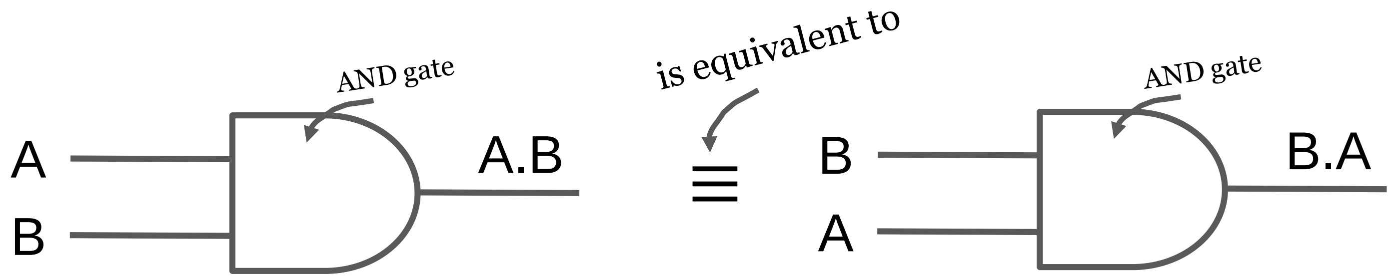
Logic Gate Symbol

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table

The output is 1 if both inputs are different

Boolean Algebra

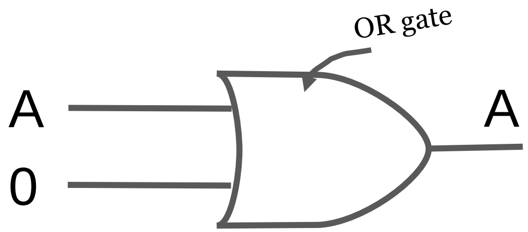


The logical AND operation is **commutitive**

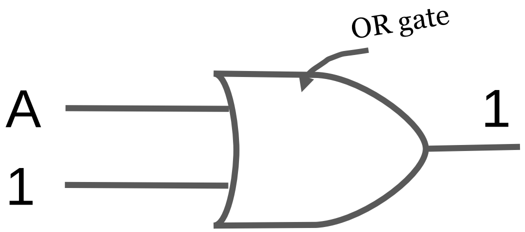
Law	AND	OR
Commutative	$A.B = B.A$	$A + B = B + A$
Associative	$A.(B.C) = (A.B).C$	$A + (B + C) = (A + B) + C$
Absorption	$A.(A + B) = A$	$A + (A.B) = A$
Distributive	$A.(B + C) = (A.B) + (B.C)$	$A + (B.C) = (A + B).(B + C)$
De Morgan's Law	$\overline{A.B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}.\bar{B}$

Boolean Algebra

PDST
Professional Development Service for Teachers | An tSeirbhís um Fhorbairt Ghairmiúil do Mhúinteoirí

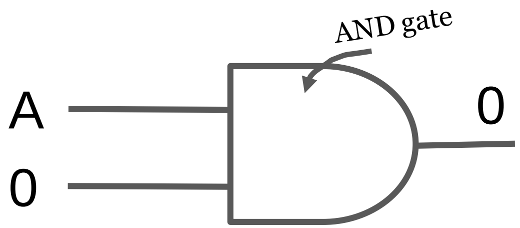


A variable ORed with 0 always gives the variable

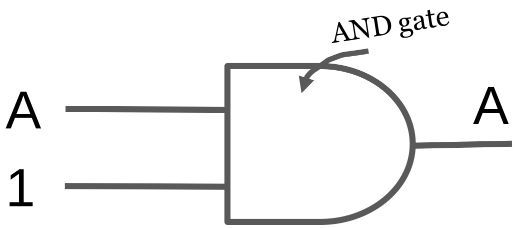


When a variable is ORed with 1 the output is always 1

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = A$
5. $A + A = A$
6. $A + \overline{A} = 1$
7. $A \cdot A = A$
8. $A \cdot \overline{A} = 0$
9. $\overline{\overline{A}} = A$
10. $A + AB = A$
11. $A + \overline{A}B = A + B$
12. $(A + B)(A + C) = A + BC$



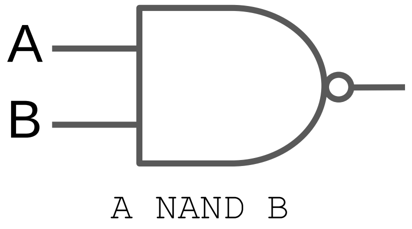
When a variable is ANDed with 0 the output is always 0.



Here we can see that when a variable is ANDed with 1 the output is always the variable

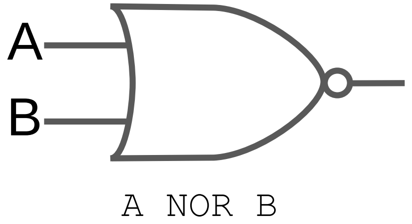
Using truth tables to verify identities

$\overline{A.B} = \bar{A} + \bar{B}$



A	B	$A.B$	$\overline{A.B}$	$\bar{A}$	$\bar{B}$	$\bar{A}+\bar{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

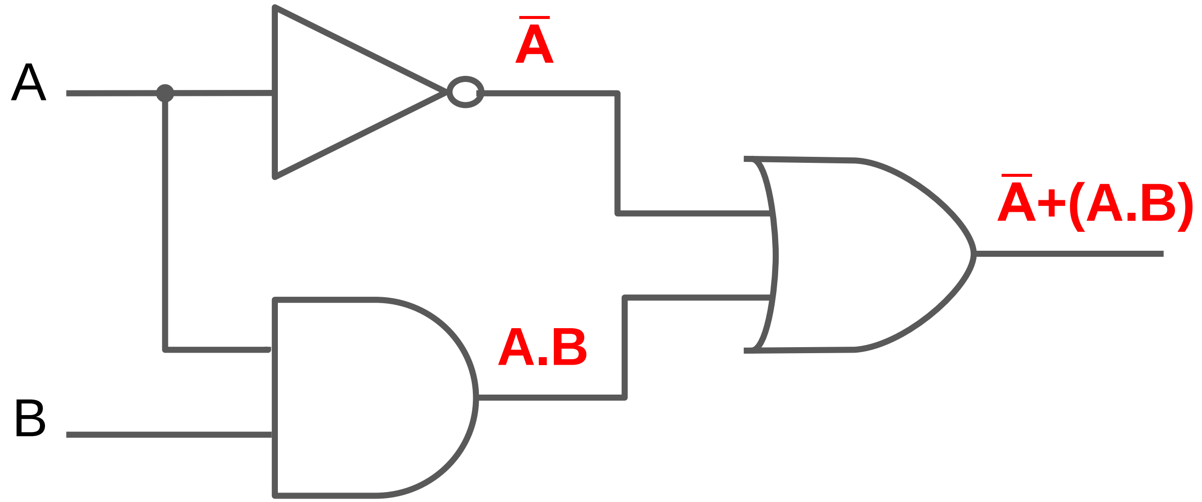
$\overline{A+B} = \bar{A}.\bar{B}$



A	B	$A+B$	$\overline{A+B}$	$\bar{A}$	$\bar{B}$	$\bar{A}.\bar{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Connect Logic Gates (to create circuits)

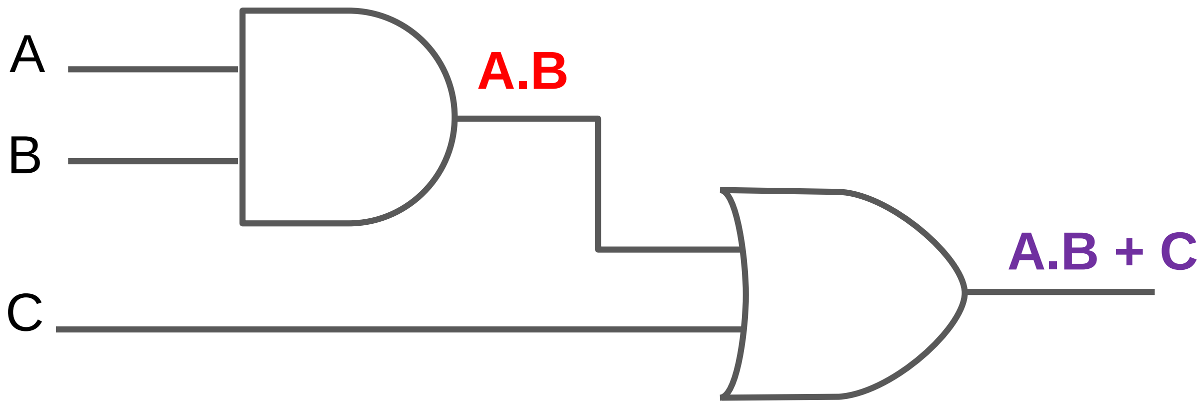
Logic gates may be combined by using the output of one gate as the input to another.



Work progressively from the inputs to the output adding logic expressions to the output of each gate in turn

Connect Logic Gates (to create circuits)

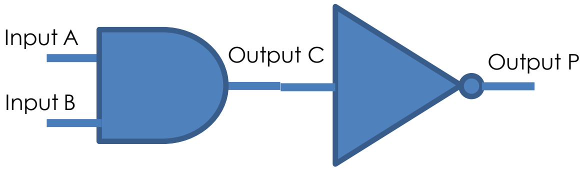
Consider this circuit used to model a smart light. The light comes if it is dark (input A) and it detects motion (input B) or it is switched on manually (C).



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

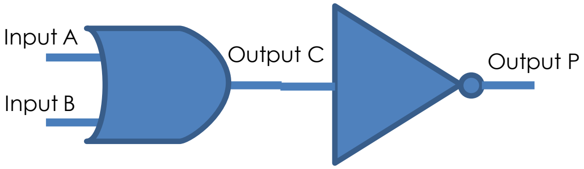
Logic Gates Truth Table Challenge

Challenge 1



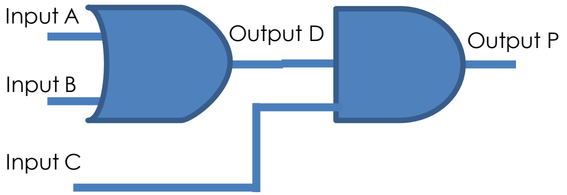
A	B	C	P

Challenge 2



A	B	C	P

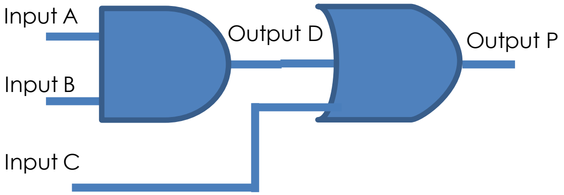
Challenge 3



For this challenge you need to work out output D first then work out what the output P is when D and C are inputted.

A	B	D	C	P
			0	
			0	
			0	
			0	
			1	
			1	
			1	
			1	

Challenge 4



A	B	D	C	P

Breakout Activity

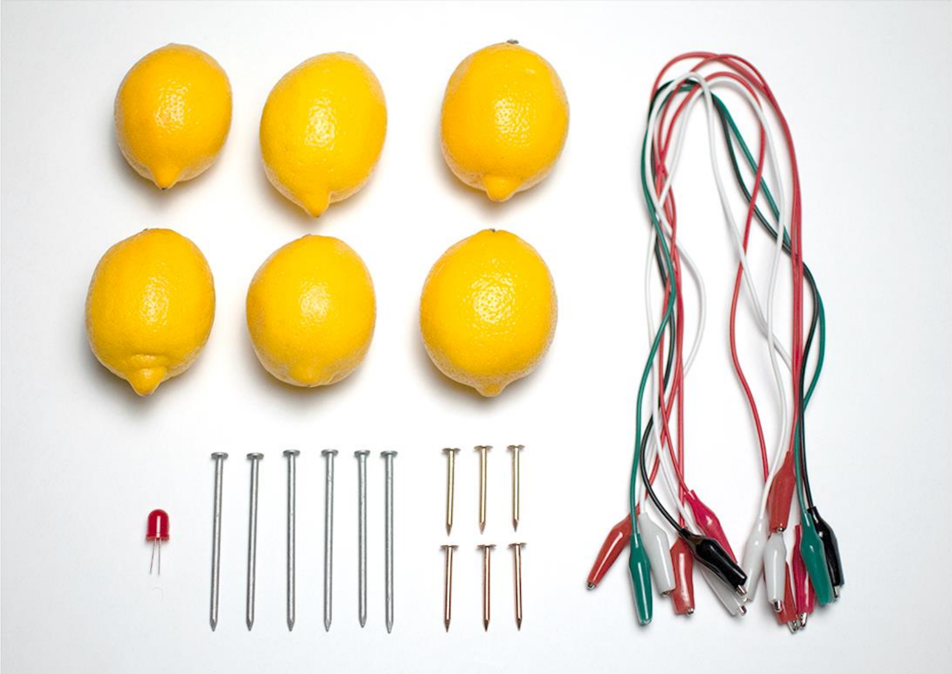
Groups and Tasks

Task 1: Design a Half-Adder (NCCA booklet Pg.23)

Design your own half-adder in some language you have learned, for example Python or Scratch

INPUTS		OUTPUT	
A	B	Sum (S)	Carry (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Task 2: Lemon Battery (Microsoft Hacking STEM)

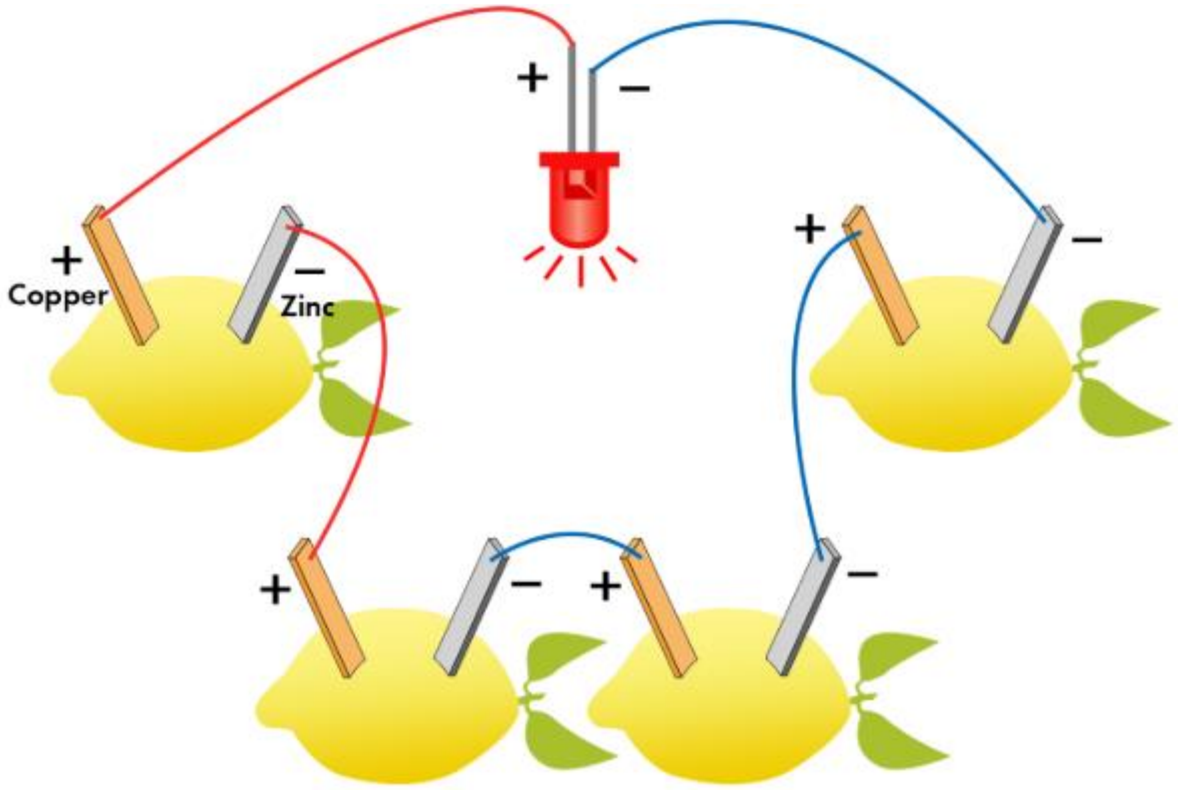


Microsoft

As you complete the task you should prepare feedback that addresses each of the following points:

1. A statement of the problem.
2. A summary of how you approached the problem.
3. Did you get a working solution? How might the task be simplified and/or extended?
4. What prior knowledge would it be useful for students to have before attempting a task such as this?
5. Describe any challenges you encountered and how you attempted to overcome these challenges. What were the outcomes?
6. What challenges are there to implementing this lesson in your classroom?
7. What scaffolding would be needed for your students with special educational needs?
8. Discuss alternative approaches that could be taken
9. Where and how could you make links with other parts of the course?
10. What 'soft skills' did the task elicit?

One Possible Solution



For more infor see: https://www.explainthatstuff.com/electricity.html

Presentation & Debrief

Explain your activity

Simplifications?
Extensions?

What problem?

Prior Knowledge?

Challenges?

Alternative solutions?

Scaffoldings for students with SEN?

Where and how could you make links with other parts of the course?

Computer Systems

Von Neumann Architecture

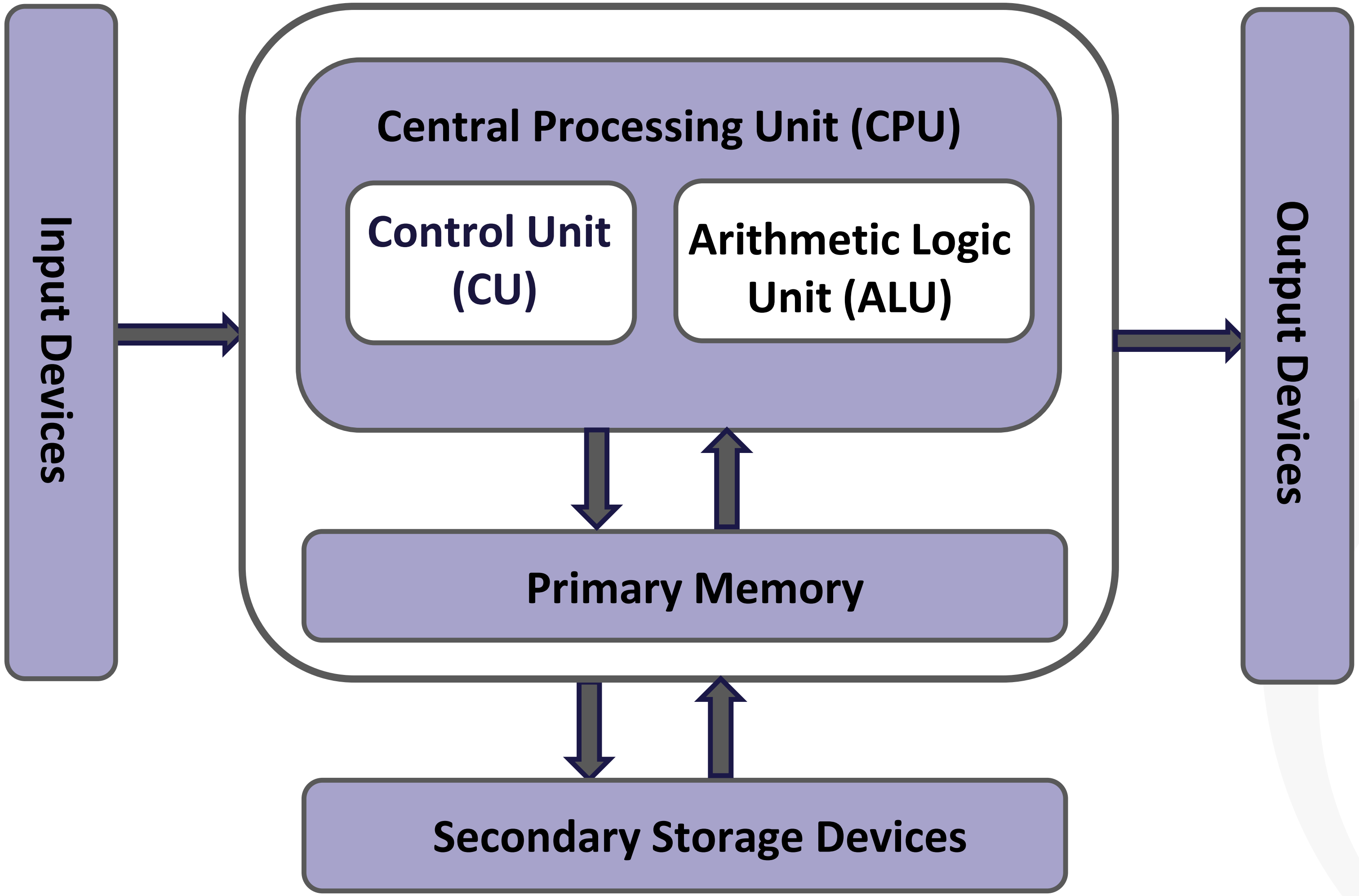
Von Neumann Architecture

Both program and data kept in storage (previously programs had been hard coded into the machine)

Simple structure

Basis upon which all modern computers are constructed

Von Neumann Architecture



2.11 describe the different components within a computer and the function of those components

The components are connected to one another by a collection of wires called a **bus**

Computer Systems

S2: Computer systems	
CPU: ALU, Registers, Program counter, Memory	2.11 describe the different components within a computer and the function of those components
Basic electronics: voltage, current, resistors, capacitors, transistors	2.12 describe the different types of logic gates **and explain how they can be arranged into larger units to perform more complex tasks**
Operating system layers: Hardware, OS, Application, User	2.13 describe the rationale for using the binary number system in digital computing and how to convert between binary, hexadecimal and decimal 2.14 describe the difference between digital and analogue input
Web infrastructure - Computer Network Protocols: HTTP, **TCP, IP, VOIP**	2.15 explain what is meant by the World Wide Web (WWW) and the Internet, including the client server model, hardware components **and communication protocols**

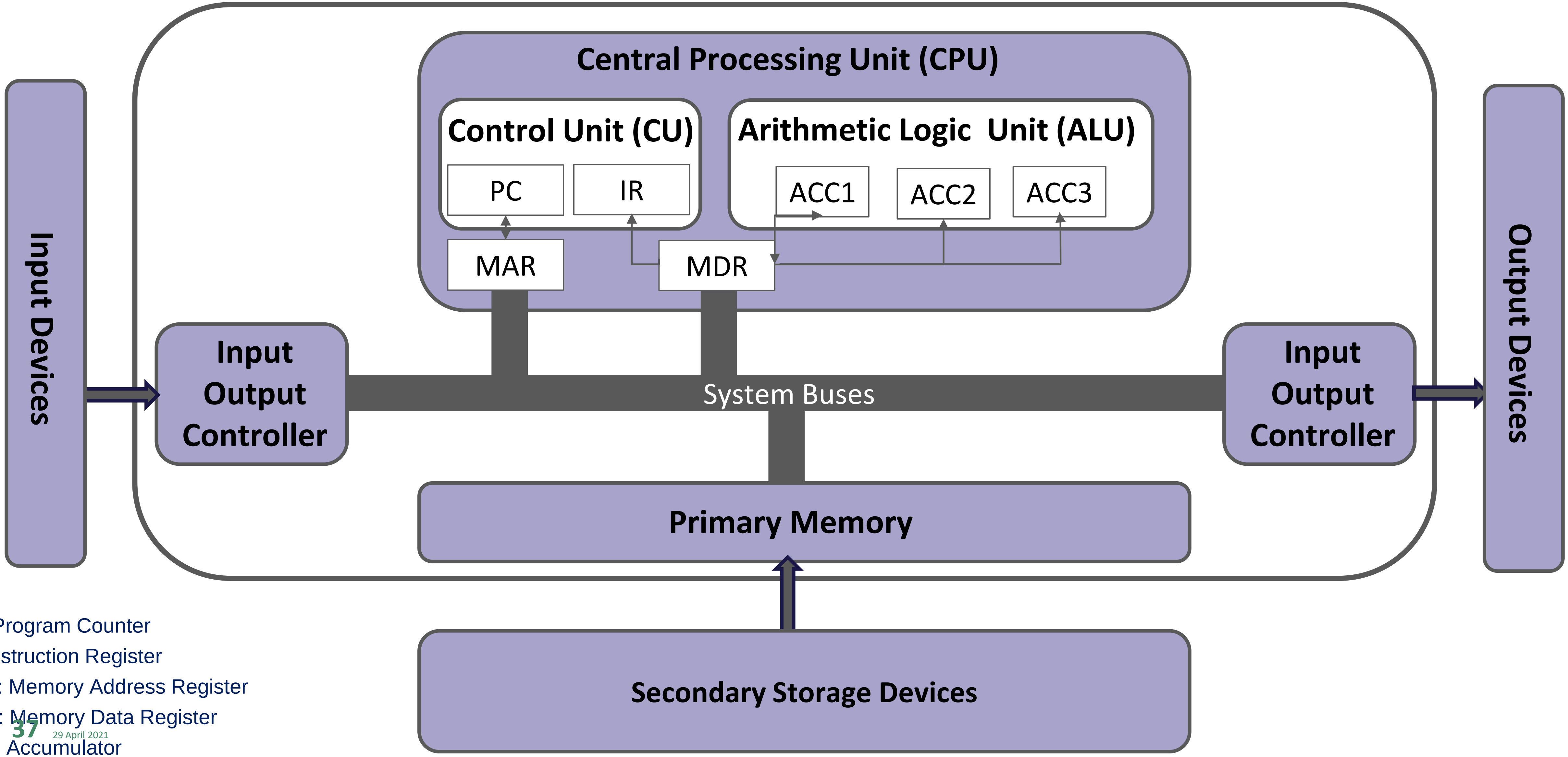
Registers

Storage locations internal to the CPU

Used as a scratchpad by the CPU to store data, addresses or instructions as it executes each program instruction

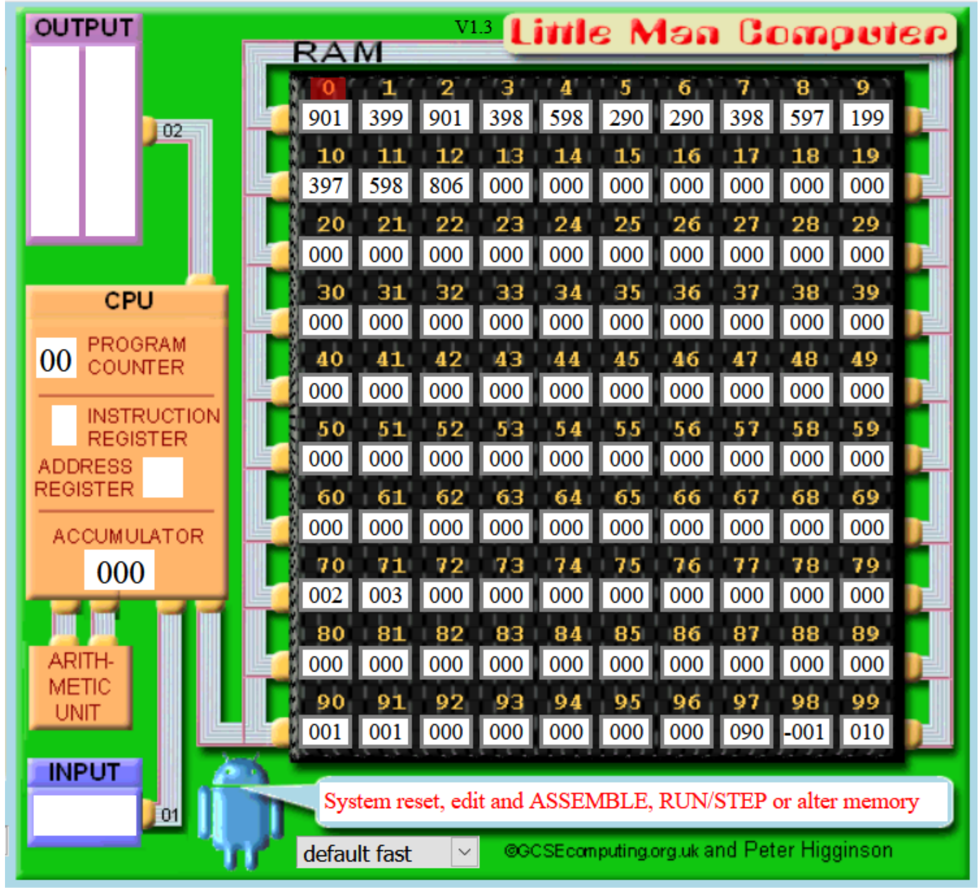
Data can be moved into and out of registers faster than from memory – dedicated pathways and hardware

The von Neumann Architecture of a Computer



PC: Program Counter
IR: Instruction Register
MAR: Memory Address Register
MDR: Memory Data Register
ACC: Accumulator

Little Man Computer



Little Man Computer Demo:
https://www.futurelearn.com/info/courses/how-computers-work/0/steps/49285

Little Man Computer Simulator:
https://peterhigginson.co.uk/LMC/

Little Man Computer Help:
https://peterhigginson.co.uk/LMC/help.html

Fetch Execute Decode Cycle:
https://www.futurelearn.com/info/courses/how-computers-work/0/steps/49284

(Another) Little Man Computer Simulator:
https://www.101computing.net/lmc-simulator/

An Roinn Oideachais
Department of Education