1947
Solid State Transistors

Learning Outcomes addressed in this section are listed below.

1.16 compare two different user interfaces and identify different design decisions that shape the user experience

2.3 implement modular design to develop hardware or software modules that perform a specific function

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

2.12 describe the different types of logic gates **and explain how they can be arranged into larger units to perform more complex tasks**

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

When other Learning Outcomes are addressed, for instance in classroom activities or through related online resources, the LO is numbered.

The digital revolution, it can be argued, began in 1947 with the invention of the solid state transistor. John Bardeen, Walter Brattain, and William Shockley, working for Bell Labs in the USA, developed the revolutionary semiconductor device which can act as a switch, turning tiny electric currents on or off, and also as an amplifier of electric current, boosting electric currents.

The operation and function of modern solid state transistors (using n-type and p-type semiconductors) are explained in this series of videos on basic electronic components.

Learn about basic electronic components (resistors, diodes, transistors, &c) and an electronics timeline.

From explainthatstuff.com.

LO 2.14

Why Binary?

When transistors work as switches they can act as a computer's memory. When a switch is turned ON it is effectively storing the number 1 (high voltage). When a switch is turned OFF it is effectively storing the number 0 (low voltage). Transistors therefore operate in 2 distinct states and since there are only 2 digits it is called a binary system. **BI**nary digi**TS** are better known by their shortened name of **BITS**.

- **8 bits makes up 1 byte**
  8 bits can represent 256 (2^8) numbers and hence 256 characters on your keyboard. The old system for representing text and numbers was ASCII but the Unicode system is now the most widely used. Check out the Universal Transformation Format (UTF) codes for emojis.
- **1000 bytes is about 1 KiloByte**
  1 KB (kB also used) is exactly 1024 bytes. This is 2^10 bytes. The base of 2 represents the fact that a bit is either ON or OFF. 1 KB is a couple of short paragraphs of text.
- **1000KB is about 1 MegaByte**
  1 MB is 1024KB or 1,048,576 bytes. This is 2^20 bytes. MP3 audio is roughly 1MB of sound per minute.
- **1000MB is about 1 GigaByte**
  1 GB is 2^30 bytes or approximately 1 billion bytes. PC RAM is measured blocks of GB
- **1000 GB is about 1 TeraByte.**
  1 TB is a million million bytes or to be exact 2^40 bytes. Terabyte hard drives and the growth of cloud storage have made this unit of measurement part of common, everyday language.

Representing data using Braille Binary, Decimal and Hexadecimal.

From the CS Field Guide (Tim Bell)

Check out this interactive Stanford University site on measuring memory.

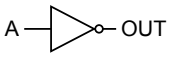
What do you think of its User Interface? Can you locate a better one? Why is it a better UI?

LO 1.16
LO 2.13

Logic Gates are the building blocks of computer systems. They are made up chiefly of transistors and enable computer systems to make decisions using Boolean Algebra. The functionality of 6 logic gates is described below (NOT, OR, AND, NAND, NOR and XOR.)

Logic Gate — **NOT**

Symbol

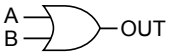


Functionality (Truth Table)

OUTPUT
A	not A
0	1
1	0

Logic Gate — **OR**

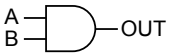
Symbol



INPUTS		OUTPUT
A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

Logic Gate — **AND**

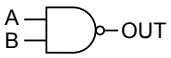
Symbol



INPUTS		OUTPUT
A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Logic Gate — **NAND (not AND)**

Symbol

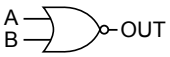


Truth Table

INPUTS		OUTPUT
A	B	not (A and B)
0	0	1
0	1	1
1	0	1
1	1	0

Logic Gate — **NOR (not OR)**

Symbol

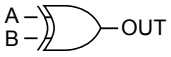


Truth Table

INPUTS		OUTPUT
A	B	not (A or B)
0	0	1
0	1	0
1	0	0
1	1	0

Logic Gate — **XOR (exclusive OR)**

Symbol



Truth Table

INPUTS		OUTPUT
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

From simple logic gates described below, the components of all computers can be constructed, such as flip-flops and memory components all the way up to ALUs and registers. In fact Arithmetic Logic Units (ALUs) comprise a vital part of the Central Processing Unit (CPU) of a computer. Logical and mathematical operations are performed here. The inputs to the ALU are controlled within the CPU by the control unit.

Analog v Digital is both very simple to understand at an abstract level and extremely complicated at the micro and quantum level. Explore the differences between analog and digital with this 5 minute Techquickie video. Microprocessors such as the Arduino and Microbit enable both digital and analog processing. The revolution in semiconductors, which began in 1947 with the invention at Bell Labs of the solid state transistor, was in many ways the birth of the digital era: of 1's and 0's. Before that, our world was driven by analog technology. Now we live in both an analog and digital world.

Take the Crash Course Video on Logic Gates

From the Crash Course Series on YouTube (Carri Anne Philbin)

Watch how to build logic gates and full adders using dominoes, plus an explanation of binary arithmetic.

LO 2.3

Interact with Logic Gates to explore how they work.

The NAND gate can be made up from the output of an AND gate fed into the input of a NOT gate. Use the simulator above to demonstrate its operation.

Use this or a similar interactive simulator to build a half-adder for the class activity.

An animated and comprehensive look at how computers add numbers.

LO 2.12

Watch the NCCA video on battery testing using a microbit, then implement in class, to explore Digital and Analog signals.

Visit ncca.ie (computer science) or compsci.ie to view similar videos.

LO 2.14
LO 3.11–3.13

Classroom Activity

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

What is a half-adder? The basic building blocks of an ALU are simple logic gates and one of the most fundamental elements of an ALU is the half-adder. The half adder takes two inputs in the form of bits, and outputs the sum and the carry-over. The addition and the Truth Table are shown below.

Addition of 2 bits
A + B = CS
0 + 0 = 00
0 + 1 = 01
1 + 0 = 01
1 + 1 = 10

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

The activity requires a minimum group of 3.

COMPONENT BUILDER

The role is to write functions in a programming language for the basic logic gates: AND, OR, XOR, NAND and NOR gates. The function should take 2 inputs (A,B) and output the result from the gate. The half-adder program will use these logic gate functions to simulate a half-adder.

HALF-ADDER DESIGNER

The role is to research half adder designs on the internet. With one of these designs, design an abstract model and algorithm to implement a half-adder design in your chosen programming language. The logic gate functions will be used within the overall model.

TESTER AND EVALUATOR

The responsibility is to ensure the half-adder outputs the correct values for all possible inputs. So, a program to perform such a test is to be designed. For example, a nested loop which generates every permutation of A and B in the Truth Table, which then feeds into the half-adder program. The program should also be tested for correct functionality and evaluated for efficiency and its Computational Thinking level. (See the html NCCA resource for evaluating CT criteria in block-based code.)

Half-Adder Activity

The activity addresses a vast number of LOs. In particular:

- Designing and Developing LOs in Strand 1 (**LO 1.19–1.23**)
- Abstraction LOs in Strand 2 (**LO 2.1–2.4**)
- ALT3 Modelling and Simulation LOs in Strand 3 (**LO 3.8–3.9**)