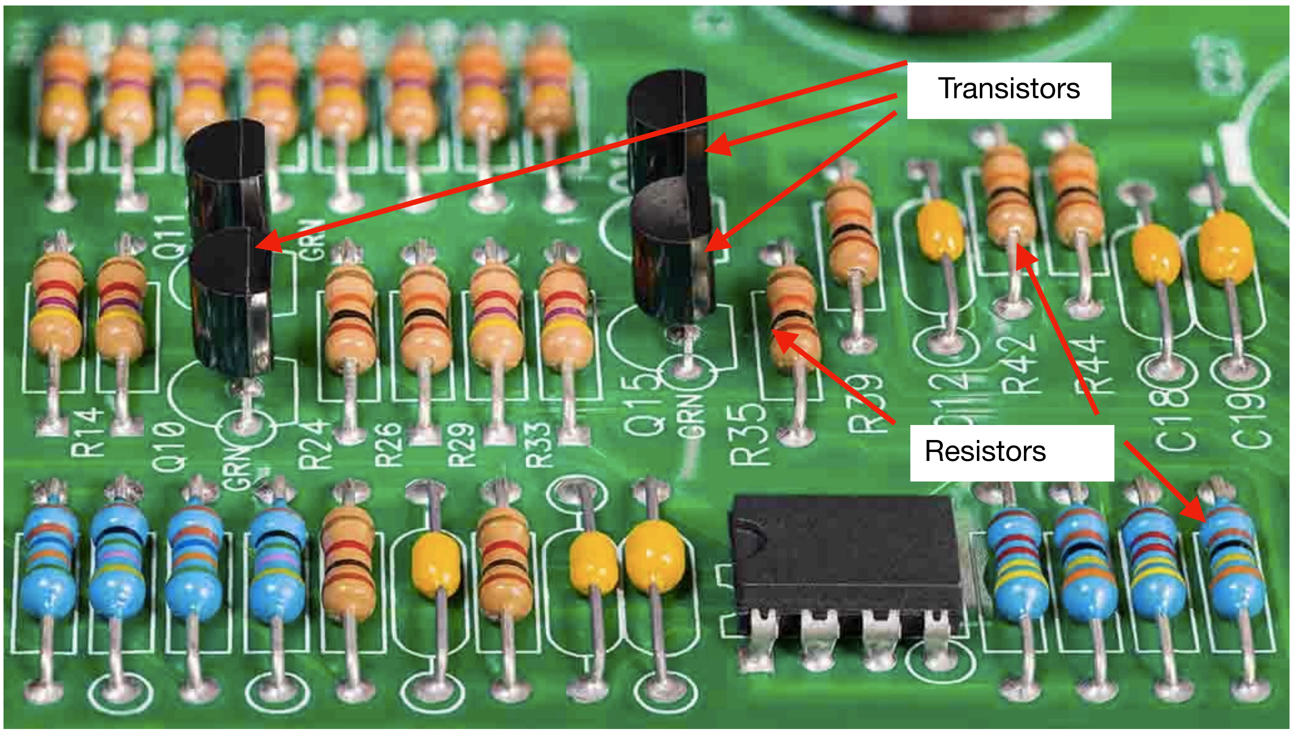
Logic Gates and Binary Conversion Revision

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

Voltage is the electrical force that causes electrons to move around a circuit. The movement of these electrons is called **current**. The source of voltage in a computer is from the mains electricity where it is plugged into the wall. Electrical devices such as computers only recognise the presence or absence of current or voltage. For example, in a light switch when it is off there is no current flowing to the bulb so it is off, if the light switch is switched on then current can flow and the bulb turns on. This is the exact same process in a computer, they are composed of billions of 'switches' called **transistors** and by manipulating the sequence with which these transistors turn on or off the computer can carry out different functions. The flow of current around a computer, or any electronic device, is controlled by devices called **resistors**. Resistors limit the flow of current and voltage to electronic circuits within a computer as different circuits may need a different amount of current or voltage. For example, USB ports may need 5 volts, 3.3 Volts is used to power the CPU and you may need 12 Volts for the motherboard or most new-gen graphics cards. Another device widely used in electronic circuits are **capacitors,** they are used for a number of reasons such as in signal processing to create a constant steady stream of electricity needed to power your components.



Resistors and transistors in a circuit

Two different types of capacitors

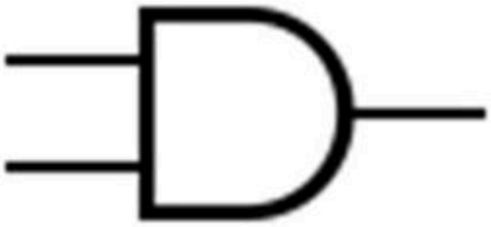
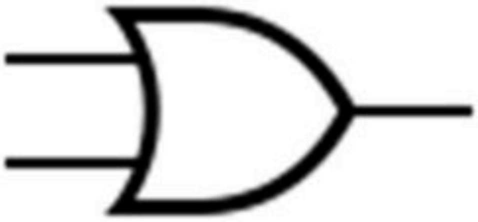
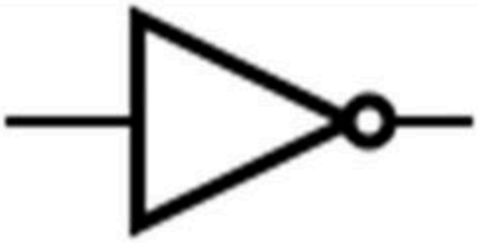
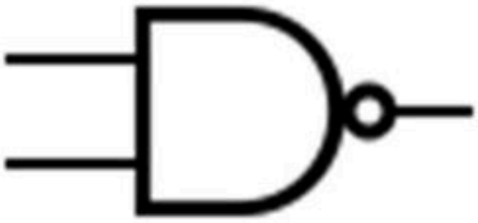
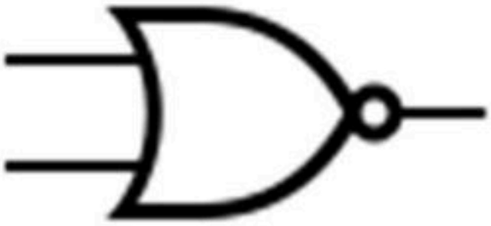
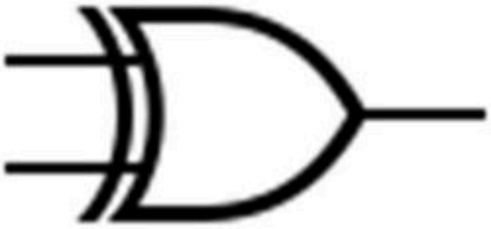
2.12 Describe the different types of logic gates and explain how they can be arranged into larger units to perform more complex tasks

Logic Gates

One of the most fundamental uses of components such as transistors is in the building of logic gates. A logic gate is a device that **acts as a building block for digital circuits**. They are used to make decisions so that electrical outputs only turn on/off when the correct logic sequence has been applied.

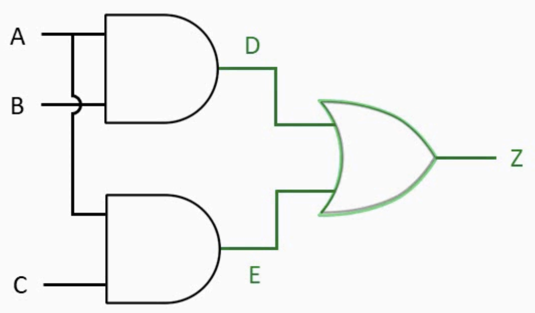
Electronic circuits in computers, solid state drives and controlling devices are made up of thousands of **logic gates**. Logic gates take binary inputs and produce a binary output. Several logic gates combined together form a **logic circuit** and these circuits are designed to carry out a specific function. The checking of the output from a logic gate or logic circuit is done using a **truth table**.

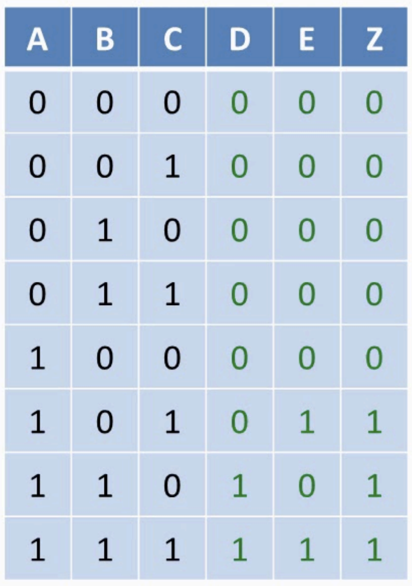
Digital Logic Gate Symbols

Gate	Symbol	Notation	Truth Table
AND		$A \cdot B$	INPUT OUTPUT A B A AND B 0 0 0 0 1 0 1 0 0 1 1 1
OR		$A + B$	INPUT OUTPUT A B A OR B 0 0 0 0 1 1 1 0 1 1 1 1
NOT		$\overline{A}$	INPUT OUTPUT A NOT A 0 1 1 0
NAND		$\overline{A \cdot B}$	INPUT OUTPUT A B A NAND B 0 0 1 0 1 1 1 0 1 1 1 0
NOR		$\overline{A + B}$	INPUT OUTPUT A B A NOR B 0 0 1 0 1 0 1 0 0 1 1 0
XOR		$A \oplus B$	INPUT OUTPUT A B A XOR B 0 0 0 0 1 1 1 0 1 1 1 0

Logic Gates and their truth table

You need to be able to combine different logic gates and create the relevant truth table.





A	B	C	D	E	Z
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Combined AND and OR gates and the relevant truth table

Binary, Decimal and Hex Conversion

2.13 Describe the rationale for using the binary number system in digital computing and how to convert between binary, hexadecimal and decimal

Computer systems use a binary system or base-2 system. This means it uses just two values 0 and 1. Binary systems can be represented by just two electrical signals, on and off. Values are stored in binary using switches (transistors) by setting them on (1) or off (0). One switch is equivalent to one bit, so a bit represent the smaller amount of information it is possible to configure. Eight switches, i.e 8 bits, make up a byte which can represent any value between 0 and 256. Computers rely on binary numbers and binary math because **it greatly simplifies their tasks.** Since there are only two possibilities (0 and 1) for each digit rather than 10 (0-9), it is easier to store or manipulate the numbers. However, when we interact with computer systems we don't use binary values, for example I am using an English keyboard to type this revision sheet, so the computer must understand how to convert each input to a binary value it understands.

Decimal to Binary

1. Draw your binary place value table.
2. Identify your decimal value you want to convert, in this case we will use 149.
3. Starting on the left, find the first place value that is smaller than 149. In this instance it is 128 or $2^{7.}$
4. Place a 1 in this position.

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1
	1							

Binary place values

5. Subtract 149 - 128 to get 21, repeat step 3.

6. 16 is the first place value that is smaller than 21. So put a 1 in this place value and 0 in each place value between the 1’s.

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1
	1	0	0	16				

Binary place values

7. Repeat step 5 and 6 until your table looks like the following.

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1
	1	0	0	1	0	1	0	1

Binary place values

8. 149 in binary is 10010101.

Binary to Decimal

We will now convert 10010101 back into a decimal number.

1. Draw your binary place value table.

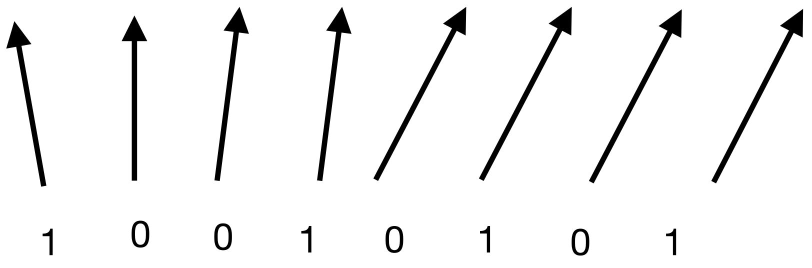
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

Binary place values

2. The place value on the furthest to the right , 2^0 ,is the least significant bit and value on the left is the most significant value.
3. In the binary value, the value furthest to the right is the least significant bit and value on the left is the most significant value.
4. Starting with the least significant bit place the binary number into the place value table.

2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1

Binary place values



2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
256	128	64	32	16	8	4	2	1
	1	0	0	1	0	1	0	1

Binary place values

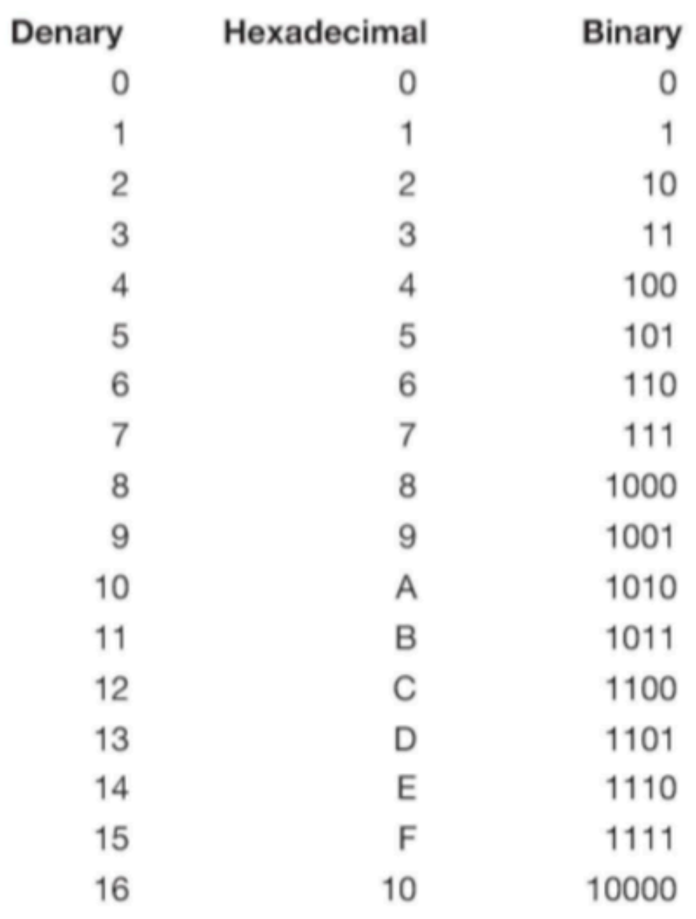
5. Each place value that has a 1 in the box, add those place values together.
6. 128 + 16 +4 +1 = 149

The **hexadecimal number system** is very closely related to the binary system. Hexadecimal (sometimes referred to as simply ‘hex’) is a base 16 system and therefore needs to use 16 different ‘digits’ to represent each value.

Because it is a system based on 16 different digits, the numbers 0 to 9 and the letters A to F are used to represent each hexadecimal (hex) digit. A in hex = 10, B = 11, C = 12, D = 13, E = 14 and F = 15.

Using the same method as for decimal and binary, this gives the place value table headings of 16^0, 16^1, 16^2, 16^3, and so on.

The hexadecimal system is used as a shorthand for binary since it is simple to represent a byte in just two digits, and fewer mistakes are likely to be made in writing a hex number than a string of binary digits. Hexadecimal is not used by computers it is only used by us to write long binary numbers in shorthand, for example 10111100011100 in binary can be represented by 2F1C in hex. It is easier for technicians and computer users to write or remember a hex number than a binary number. Colour codes in images often use hexadecimal to represent the RGB values, as they are much easier to remember than a 24-bit binary string.



Denary	Hexadecimal	Binary
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000

Comparing decimal, hex and binary numbers

The typical headings for a hexadecimal number with six digits would be:

16^5	16^4	16^3	16^2	16^1	16^0
1048576	65536	4096	256	16	1

Hexadecimal place value table

Hexadecimal to decimal

1. Draw the place value table
2. Place the hex value into the table, once again starting from the right hand side with the least significant figure. The hex value we will use is 36B, this means B is the least significant and 3 is the most significant figure.
3.

16^5	16^4	16^3	16^2	16^1	16^0
			3	6	B

4. Multiply each hex value by the place value and sum the values together.

5. (3 * 256) + (6 * 16) + (11 * 1) = 875

6. 36B is 875 in decimal form.

Binary addition and Overflow

During the fetch decode execute cycle the ALU may have to add or subtract binary values as part of an operation. Binary addition is similar to decimal addition apart from one rule. If you add two 1 bits, you don't get 2. You get a carry value instead, similar to decimal addition when you get add two values and get a number greater than 10.

Addition	Sum value	Carry value
0 + 0	0	
1 + 0	1	
0 + 1	1	
1 + 1	0	1

Rules of binary addition

Examples of addition of two binary values

Carry	1			1	1	1	
Binary Number	0	1	1	0	0	1	1
Binary Number	0	1	0	0	1	0	1
Sum	1	0	1	1	0	0	0

51 + 37 = 88

Carry	1		1				
Binary Number	0	1	0	1	0	0	1
Binary Number	0	1	0	1	1	0	0
Sum	1	0	1	0	1	0	1

41 + 44 = 85

Overflow occurs when you have a carry value when you get to the highest bit value. (furthest left). There is no more bits to continue the addition.

The result of the addition is greater than 255 and an overflow error occurs where a ninth bit is needed.

Most CPUs use a much bigger word size than 8 bits. Many have a 64-bit CPU. A 64-bit CPU can handle numbers larger than 18 quintillion (18,446,744,073,709,551,615 to be precise).

Carry		1	1				1		
Binary Numbe		1	1	1	0	0	0	1	0
Binary Numbe		1	0	1	1	1	0	1	0
Sum		?	0	0	1	1	1	0	0

Example of overflow

ASCII and Unicode

2.17 Use ASCII and Unicode character sets to encode/decode a message and consider the importance of having such standards

The **ASCII code** system (American Standard Code for Information Interchange) was set up in 1963 for use in communication systems and computer systems. A newer version of the code was published in 1986. The original ASCII code **character set** consists of 7-bit codes (0 to 127 in denary or 00 to 7F in hexadecimal) and the newer version in 1986 used an 8-bit veresion, that represent the letters, numbers and characters found on a standard keyboard, together with 32 control codes (that use codes 0 to 31 (denary) or 00 to 19 (hexadecimal)).

ASCII encoding is technically obsolete, having been replaced by Unicode. Yet, ASCII characters use the same encoding as the first 128 characters of the Unicode Transformation Format 8, so ASCII text is compatible with UTF-8.

Advantages

- **Universally accepted.** ASCII character encoding is universally understood. It is universally implemented in computing through the Unicode standard. Unicode character encoding replaces ASCII encoding, but it is backward-compatible with ASCII.
- **Compact character encoding.** Standard codes can be expressed in 7 bits. This means data that can be expressed in the standard ASCII character set requires only as many bytes to store or send as the number of characters in the data.
- **Efficient for programming.** The character codes for letters and numbers are well adapted to programming techniques for manipulating text and using numbers for calculations or storage as raw data.

Disadvantages

- **Limited character set.** Even with extended ASCII, only 255 distinct characters can be represented. The characters in a standard character set are enough for English language communications. But even with the diacritical marks and Greek letters supported in extended ASCII, it is difficult to accommodate languages that do not use the Latin alphabet.
- **Inefficient character encoding.** Standard ASCII encoding is efficient for English language and numerical data. Representing characters from other alphabets requires more overhead such as escape codes.

By the 1980s, several coding systems had been introduced all over the world that were all incompatible with one another. This created difficultly as multilingual data was being increasingly used and a new, unified format was sought. As a result, a new 16-bit cade called Unicode (UTF-16) was introduced. This allowed for 65,536 different combinations and could therefore represent alphabets from dozens of languages including Latin, Greek, Arabic and Cyrillic alphabets. The first 128 codes were the same as ASCII so compatibility was retained. A further version of Unicode called UTF-32 was also developed to include just over a mil ion characters, and this was more than enough to handle most of the characters Irom all languages, including Chinese and Japanese.