1 Data Representation

1.1 Number systems

1.1.1 Computers and binary

The number system consisting of digits 0 through 9 is called the denary or decimal system (10 digits). Computers use another number system called binary, consisting of digits 0 through 1. This is done such that computers are able to pass the data through logic gates and can be stored in registers.

1.1.2 The binary, denary and hexadecimal systems

The number of digits in a number system is the base of that system; denary is base 10; binary is base 2; hexadecimal is base 16.

Conversions

From positive denary to positive binary:

1. Perform short division on given denary number, taking note of the remainders.
2. Write the remainders from bottom to top, resulting in the binary number.

From positive binary to positive denary:

1. Write the binary number with their *place powers*.
2. Sum the products of each binary digit with the place power, resulting in the converted denary number.
3. Write the remainders from bottom to top, resulting in the binary number.

From positive denary to positive hexadecimal:

1. Convert given number to binary
2. Split resulting binary number to four-bit parts.
3. Convert the four-bit binaries to denary.
4. 1 = 1; 2 = 2; ... 10 = A; 11 = B; 12 = C; 13 = D; 14 = E; 15 = F.
5. Arrange resulting digits side-by-side.

From positive hexadecimal to positive denary:

1. Convert each given hex number to denary using the index in step 4 of denary-hex.
2. Convert each denary number to binary.
3. Arrange the resulting four-bit binary pieces, producing binary result.

From positive hexadecimal to positive binary:

1. Convert to denary.
2. Convert to binary.

From positive binary to positive hexadecimal:

1. Convert to binary.
2. Convert to denary.

1.1.3 Uses of the hexadecimal system

The hexadecimal number system is used to make life easier for humans dealing with bare low-level computer code. Hexadecimal requires less digits, and are easier to compare with the naked eye. Data errors can be easier to find when looking at this shortened form of binary, hexadecimal.

1.1.4 Binary addition

To add two binary numbers, refer to the following:

- $0 + 0 = 0$.
- $1 + 0 = 1$.
- $1 + 1 = 10$. (the one is carried on)

Sometimes, the addition results in an extra bit, which **overflows** off. This is because computers have predifined limits to which it can store its numbers (16, 32 bits) and when a value outside this limit is returned, it is not stored and an overflow error occurs.

1.1.5 Logical binary shifts

When performing logical shifts, we simply move the bits of a binary number to the right or left depending on what is required. We "delete" and hence lose the leftmost (most significant) or rightmost (least significant) bit depending on the direction of the shift performed.

Shifting right means dividing by two.

Shifting left means multiplying by two.

1.1.6 Two's complement

Two's complement is a method used to represent negative binary numbers. We simply convert given denary number to binary (if need be), invert all the bits and add $(1)_2$ to the result.

1.2 Text, sound and images

1.2.1 Text

Text is converted to binary so that a computer can process it. It does so by converting each character into an integer, as defined in the **ASCII** standard (American Standard Code For Information Interchange) and subsequently into a binary number.

Unicode is another such standard, which allows a greater range of characters, in various languages, as a result it also requires more bits per character.

1.2.2 Sound

Sounds are composed of waves. When we record values of the sound, we do so at set intervals, this process is called **sampling**. The more samples taken per unit time, the more accurate the sound recorded will be, i.e., higher the **sample rate** the greater the sound quality.

The sound values, which are usually denary numbers, can be converted to binary and stored into a computer.

The **sample resolution** is the number of bits allocated per sample value. So, the larger the sample resolution, the more the amount of digits that can be stored into a file. Thus, the higher the sample resolution, the higher the sound quality.

The file size of a sound file increases, with increased sample rate and sample resolution, that means storing high quality sound requires more space that low quality sound.

1.2.3 Images

An image is composed of **pixels**. The computer stores these pixels by processing them to binary, by assigning a binary number to a certain colour.

The **resolution** of an image is the number of pixels stored in it. Usually in the format: width $\times$ height.

The **colour depth** of an image is the number of bits allocated for each pixel of the image. Higher the colour depth, the more the number of colours that can be displayed.

Higher quality images result in larger file sizes as resolution and colour depth are large.

1.3 Data storage and compression

1.3.1 Measurement of data storage

- Bit: 1 or 0. Smallest possible data measurement.
- Nibble: 4 bits.
- Byte: 8 bits.
- Kibibyte (KiB): 1024 Bytes.
- Mebibyte (MiB): 1024 Kibibytes.
- Gibibyte (GiB): 1024 Mebibytes.
- Tebibyte (TiB): 1024 Gibibytes.
- Pebibyte (PiB): 1024 Tebibytes.
- Exbibyte: (EiB) 1024 Pebibytes.

1.3.2 File size calculation

To calculate the file size of an image, find the product of the image's width, height, colour depth

For the sound's file size, multiply the sample rate, sample resolution and soundtrack length.