# Week 2 Lab Handling data

## Objectives

Develop understanding and experience of:

1. Data representation – text and images
2. Data representation – signed integers
3. Building a simple ALU

### Part 1 Representing text and images

The tutor will give a short recap on binary and hexadecimal before introducing the concepts of storing text. You will use Visual Studio code for the practical work for this part (along with Windows File Explorer).

1. Looking at the ASCII table
2. Open Visual Studio Code (VS Code)
3. Choose file and new file
4. Ignore any message that comes up and type in a short message (more than one word) using standard English characters.
5. Save the file as plain text (file extension .txt)
6. Right-click on the filename at the top of the tab you are using in VS Code and Choose “Reopen Editor With…”
7. Choose Hex Editor. If hex editor does not come up on the list, start typing hex editor.
8. Look at the hex version of the message.
9. Use the ASCII table to change the case of the message by altering the hex digits. That is, change lower to upper case and vice-versa.
10. Comparing two text files
11. Download the files **bytes\_v1.txt** and **bytes\_v2.txt** from Moodle
12. Open both files in VS code (but do not look at the hex yet).
13. Compare the two files by reading them – they should look the same
14. Look at the number of bytes each file uses. In Windows File Explorer, you can right click and choose properties which will also give the size of the file.
15. The text looks the same but takes up a different number of bytes, use the hex editor in VS Code to work out what differences there are.

The tutor will lead a discussion into your findings from these exercises.

The tutor will explain some concepts relating to storing images and lead the following exercise on looking at the representation of a bitmap image.

1. Looking at images in bmp format
2. Download the file **tiny\_24\_bit.bmp** from Moodle and open in VS Code.
3. VS Code should open it as an image preview, but it will be tiny as it is a 4x4 pixel image. You should be able to zoom up to at least 1000%. In the bottom bar of VS Code you can click on the zoom level which might say “Whole Image”.
4. Right-click on the image name at the top of the tab and reopen with the Hex Editor
5. The first part of the file is metadata, the pixel data starts at position X’36’. We will look at the data together. You should be used to RGB colours with each having a value between 0 and 255. In bmp file encoding, the order of the colours is BGR, that is blue, green, red.
6. Your task is to change the colours in the image by changing the hex values only. You should change all the green parts of the rainbow to be blue instead. Save the file and open reopen with the Image preview to check your changes.

The tutor will lead a discussion into your findings.

### Part 2 Representing negative numbers

The tutor will talk through two’s complement as a way to store signed integers in binary (numbers that can be positive or negative).

Try the following exercises before the tutor looks at the solutions with the group. You should do rough work for the steps of these and to discuss your answers with other students. After working out the answers by hand, you might want to use a conversion website to check your answers.

1. The following numbers are all 8-bit two’s complement, which of them are negative? You do not need to convert them to decimal for this task.

00101011 10110010 00011111 10001001

1. Work out the two’s complement of the following binary numbers (they are 8-bits). You do not need to convert them to decimal for this task.

00011011

11110011

1. If we want to store the decimal number -5 in an 8-bit two’s complement format, what will it be?
2. Suppose you have the following binary value stored in 8-bits but you don’t know if it is a signed or unsigned integer, what are the two possible decimal values it could have?

10011001

1. What is the decimal equivalent of the following 8-bit binary number using 2’s complement representation?

01111111

What do we get if we add 1 to that binary number using normal binary arithmetic?

What is the decimal equivalent of the resulting binary number? Remember that we are using two’s complement.

1. If you need to subtract a number, it can be easier to add the negative instead. Using 8-bit two’s complement, use binary addition to do the equivalent calculation of 12 – 5, that is do 12 + (-5). Make sure that you check your answer.
2. What would the representation be if we store -5 using 16-bit two’s complement?

The tutor will discuss the answers with the group.

### Part 3 Combining logic gates to create components

The tutor will talk explain the basic ideas of an ALU.

1. Create an ALU in Logisim as follows:
   1. Have two 8-bit input pins (labelled A and B) for the data to operate on and a 2-bit input (labelled ALU OP) for the operation to carry out.
   2. Have one 8-bit output pin (labelled ALU OUT)
   3. The inputs should go into all of the required operations given in the table below. Use Logisim built-in components for all of the operations, adder and subtractor are in the arithmetic tools.
   4. Use a multiplexer to select the final output according to the value of the 2-bit ALU OP input according to the table below. Again, use a Logisim built-in component for the multiplexer (from the plexers tools).

|  |  |
| --- | --- |
| ALU OP value | Output |
| 0 | The result of adding A and B together |
| 1 | The result of a bitwise A AND B |
| 2 | The result of subtracting 1 from A |

Note that for operation 2 to subtract 1, you will need a constant (from the Logisim wiring tools) that has Data Bits set to 8 and the value set to 1.

The tutor will discuss the work. Remember to work on the reinforcement exercises if you have time during the lab.