Register and opcodes (part one)

Introduction:

The set of instructions a particular microprocessor can execute is called its instruction set. Instruction sets consist of binary codes but are usually described for human readers in a text-based assembly language. Binary instructions are given to the microprocessor with an operation code to represent the type of instruction followed by any parameters required by that instruction. An instruction, shown in assembly language, could look something like this:

ADD r1, r2

In most common instruction sets, this instruction would add the contents of register1 and register2, storing the result back into register1.

A little history about programming languages:

First Generation languages were written in binary (1s and 0s) and were written for a specific CPU. This meant that code needed to be rewritten when a program moved to another computer.

1GLs were not user friendly. Read more 8.1.2 1GL

2GLs made life a little easier. Read more 8.1.2 2GL

3GLs used more common language structures to make coding more natural. Python (and all C family languages) are mostly 3GLs Read more 8.1.2 3GL

4GLs are natural language models for code. The most common is SQL. Read more 8.1.2 4GL

Why is this important to us? All code made in any language or generation of programming language other than 1GL or binary code needs to be compiled. Read more 8.1.2 Translation, Compilation, Interpretation

a. Summarise these notes here:

b. Complete these questions

1. Describe key features of 1st, 2nd, 3rd and 4th generation programming languages.

2. Distinguish between 1st, 2nd, 3rd and 4th generation programming languages (that is to say show the key differences between each generation or programming language).

3. Complete this table:

	Compilers	Interpreters
Advantages		
Disadvantages		
Steps to go from coding language to machine language		
Example languages		

Binary, ASCII, number representation and how this all relates to Mechatronics

Data is processed in binary.

All data is processed as binary. Why? Read more here:
9.4.2 - 0 Digital Representation of Data and 9.4.2 - 1 ASCII and Unicode

Summarise your findings here

Number systems

In which we learn to convert between Binary, decimal and Hexadecimal. And how to render negative numbers.

And why it's important (hint: registers in microcontrollers need to know how its done)

9.4.2 - 2 Number systems and 9.4.2 - 3 Rendering negative numbers

Complete these exercises

Convert the following number between bases.

Base2 (Binary)	Base10 (Decimal)	Base16 (Hex)
11001010		
	452	
	368	
		6f1

Hint: Start with the binary number line. Remember when doing Hex and Oct conversions, start from the right and add leading zeros. Show your working below:

Render these numbers as their 2s Compliment version (convert dec to bin, invert 1s and 0s, then add 1)

Decimal	**Binary**	**1s Compliment**	**2s Compliment**
-23			
-67			
-95			
-234			

Floating Point Numbers are very accurate, very big, or very small numbers. Read more
9.4.2 - 5 Floating Point Numbers