

Instructions

In this video we will learn how instructions are coded as bit patterns and we will learn how the computer distinguishes between instructions and data.

Computer programs are made up of a series of instructions, a compiler running a high level program must first convert the code that has been written by a programmer into a binary representation of the instructions, this is known as machine code.

When we program we do not directly write in a language that the computer understands, a programming language is known as a 'higher level language', this is then converted into machine language known as binary instructions that a computer would understand.

Instructions are coded as binary patterns. Bit patterns are made up of three parts: the Op-code sets the instruction part, the Operand – this will contain either data or the address to be processed and the Number bit – this tells the computer if the information in the Operand is either a value or an address.

This here is an instruction set. A series of instructions are used by the CPU to perform a series of tasks.

The example here shows the Op-code as "011" which has the Mnemonic "ADD" and the function would be to ADD the value to the operand 011010.

I know that in binary 000011 is the number 3.

The number bit in this example is set as '1' so I know that in this instruction 3 is a value and not an address. So, if I put it all together, this instruction code would mean to ADD value 3.

As we have explained in the fetch execute cycle, instructions are collected, processed and run one by one. So here is a typical table that shows the start of an instruction's being followed by the CPU.

The first three digits in any line of the machine code will always be the Op-code instruction and these are written as Mnemonics to simplify programming (so we have LOAD, STORE, ADD), this is known as assembly language. Can you see it is much easier for us to read and follow the Assembly code instead of the machine code? An assembler is used to convert the instructions to the binary representation so that a CPU can understand the instruction.


It is a really important point here to remember that the computer will expect the first three digits to be the Op-Code instruction set, it has no way to understand and cannot distinguish between the instructions and the data that it receives. Therefore an error will occur if this is given incorrectly. The CPU will always attempt to process the first three digits as an O-code instruction.

Summary

To summarise we have learnt that the binary instructions have three main parts: the Op-Code, the Operand and the binary bit.

The Op-code sets the instruction part, the Operand – this contains either the data or an address to be processed and the number bit which sits in between the Op-code and the Operand and this tells the computer if the information in the Operand is either a value or an address.

Mnemonics are used to simplify the programming. An assembler is used to convert the instructions to machine language so that the CPU can understand the instruction. Machine language is when the



instructions are in binary code. This is how the computer understands instructions, but it is not so easy for us humans to read what is happening.

Lastly, it is important to note that the computer cannot tell the difference between the data that it receives. The computer expects the first part of the data to be the Op-Code, the instruction, and once it has fetched the instruction it will process it as an instruction. This can cause errors in the program if it does not fetch the instruction correctly.