1.2.4 High vs. Low Level Languages

Comparing High and Low Level Languages

High Level Languages	Low Level Languages	
Examples. Python, C#, Visual Basic, Java, C++	**Assembly Language**	**Machine Code**
Independent of hardware, therefore portable between machines.	Hardware specific. Non-portable.	
Translated using either a compiler or an interpreter.	Translated using an assembler.	Executable binary code.
One statement translates to many machine code instructions.	One to one relationship between assembly and machine code with mnemonics representing machine code instructions.	
No direct management of memory.	Direct control over how memory is used via addressing modes. More memory efficient.	
Programs tend to run slower as not coded specific to the hardware. Optimisers in compilers attempt to increase performance.	Programmers can choose exact instructions so can write code that is highly efficient.	
More intuitive and easier to read, write, debug and understand code. Easier to build programs in a team.	Difficult to read, write, debug and understand code. Takes longer to write programs then high level languages.	
Better in large projects which don't have many constraints	Better used in projects where highest performance is critical, where memory is very limited, where there doesn't exist a compiler or interpreter for the target CPU such as an embedded system.	
Examples		
payrate = 7.38 hours = 37.5 salary = payrate * hours	LDA 181 ADD 98 STO 194	1001110001101010111001 0001010110