Assembly languages were soon developed that let the programmer specify instruction in a text format (e..g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses.  
  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
Ideally, the programming language best suited for the task at hand will be selected.  
Scripting and breakpointing is also part of this process.  
Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute.  
 Allen Downey, in his book How To Think Like A Computer Scientist, writes:  
 Many computer languages provide a mechanism to call functions provided by shared libraries.  
However, readability is more than just programming style.  
 Programmable devices have existed for centuries.  
 The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems.  
The following properties are among the most important:  
  
 In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
Integrated development environments (IDEs) aim to integrate all such help.