Evolution of programming languages

Machine language/Code

- The first generation of codes used to program a computer, was called machine language or machine code, it is the only language a computer really understands, a sequence of 0s and 1s that the computer's controls interprets as instructions, electrically

Symbolic

- Was developed by Grace Hopper, a mathematician and a US navy admiral.
- Developed the concept of a special computer program that would convert programs into machine language.
- Because computer do not understand symbolic language ,it must be translated to machine language
- A special program called assembler translates symbolic code into machine language
- Because symbolic languages had to be assembled into machine language, they soon became known as assembly language, this name is still used today for symbolic languages that closely represent the machine language of the compiler.

Assembly Code.

- The second generation of code was called assembly language, assembly language turns the sequences of 0s and 1s into human words like 'add'.
- Assembly language is always translated back into machine code by programs called assemblers

High level language.(HLL)

- Although assembly language greatly improved program efficiency, they still required programmers to concentrate on the hardware that they were using
- It was tedious working on symbolic languages because each machine instruction had to be individually coded
- The desire to improve programmer efficiency and to change the focus from the computer to the problem being solved led to the development of high level language.

- HLL are portable to many computers allowing the programmer to concentrate on application problem at hand rather than the intricacies of the computer.
- They are designed to relieve the programmer from the details of assembly language.
- They share one thing in common with assembly language/symbolic languages, they must be converted to machine language.

Natural Languages

- Ideally we could use our natural languages (such as English, French or Chinese) and the computer would understand it and execute our requests immediately.
- Though this sounds science fiction considerable work on natural language is being in labs today

History of C

The *milestones* in C's development as a language are listed below:

- UNIX developed c. 1969 - DEC PDP-7 Assembly Language
- BCPL - a user friendly OS providing powerful development tools developed from BCPL. Assembler tedious long and error prone.
- A new language ``B'' a second attempt. c. 1970.
- A totally new language ``C'' a successor to ``B''. c. 1971
- By 1973 UNIX OS almost totally written in ``C''.

Low level vs high level languages

Lecture Overview

- Definition Low & High Level Language
- Contrast Low & High Level Language

Definition

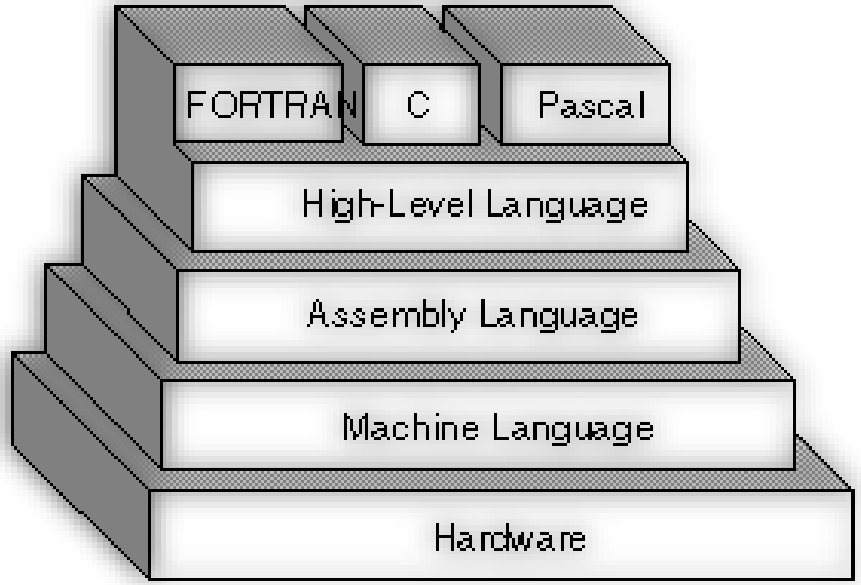
- Low level languages:
  - Computer language consisting of mnemonics that directly correspond to machine language instructions
- High Level Languages:
  - Basically symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes.

Contrast

- Low Level Languages
  - Very close to machine language
  - Concentrate on machine architecture
  - Doesn’t require translation
- High Level Languages
  - Machine-independent programming language
  - Concentrate on the logic of problem
  - Requires translation

Examples

- Low Level Language
  - Machine language
  - Assembly language
- High Level Language
  - C
  - C++
  - BASIC
  - Java



LOW vs. HIGH Level Languages

- Differences in terms of:
  i. Understandability
  ii. Ease of writing
  iii. Running speed
  iv. Writing format

Understandable

- Low Level Language:
  - Mnemonic, binary, hexadecimal

- High Level Language:
  - Simple English and mathematics symbols

Adds two numbers and stores the result

- Low Level Language:

```
.model small, C
.586

.data

mov eax,5
mov ebx,10

add eax,ebx

end
```

Adds two numbers and stores the result

- High Level Language:

```
int main()
{
   //assign to the variable result the value of 5 + 10
   int result = 5 + 10;

   return 0;
)
```

Ease of Writing

- Low Level Language:
  - Designed for the ease of the computer running the language.
  - Difficult for human to read and write

- High Level Language:
  - Designed for the ease of the person writing the language.
  - Using language that human can understand, English

Running Speed

- Low Level Language:
  - Faster
  - No need to compile

- High Level Language:
  - Need compiler or interpreter
  - Translate into machine code

Summary

- Low level languages:
  - Computer language consisting of mnemonics that directly correspond to machine language instructions
- High Level Languages:
  - Basically symbolic languages that use English words and/or mathematical symbols rather than mnemonic codes.
- Differences in terms of:
  i. Understandability
  ii. Ease of writing
  iii. Running speed
  iv. Writing format