Introduction to Computer Programming

Introduction

- A programming language can be defined as –
  - "The language used for expressing a set of instructions that can be executed by the computer".
- Programming languages can be divided into two major categories: **low level** and **high level** languages.
- Low level languages can be further divided into machine and assembly languages.
- The high level languages can be, however, categorized into three many types, that is, Procedure, Object and Problem oriented

Low-level languages

- Computer hardware is a digital machine
- It works on binary electronic pulses – 0s and 1s.
- Every operation has to be expressed as a binary instruction.
- The binary instructions are also called machine instructions.
- A machine instruction consists of two parts: op-code and operand.

Low-level languages (contd.)

- Machine instruction is simply a string of binary digits.
- With a view to increase readability, programmers assigned appropriate symbols to the op-codes on the basis of the operation performed by the instructions.
- The symbols of a hypothetical machine are given in Table.

Low-level languages (contd.)

Symbol (mnemonic)	Binary code
Load	0000 0000
Add	0000 0001
Sub	0000 0010
:	:
Stop	1111 1111

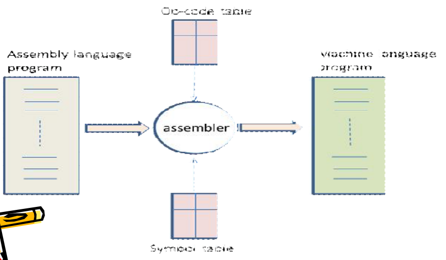
- This symbolic language for writing programs was termed assembly language.

Assembly Language

- English-like abbreviations representing elementary computer operations.
- The computer cannot understand assembly language –
- a program called assembler is used to convert assembly language programs into machine code

The Assembler

- An assembler converts the assembly language program into an equivalent machine executable program, as shown in Fig.



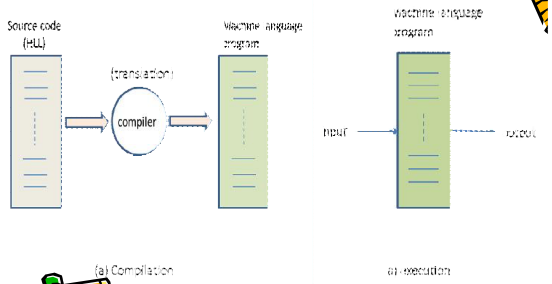
7

High-level Languages

- HLL is English-like,
- a statement written in this language is not understandable to the machine.
- A high-level language translator known as a compiler is required.
- A **compiler** can be precisely defined as *a program that translates a program written in high-level language into machine language.*

8

High-level Languages



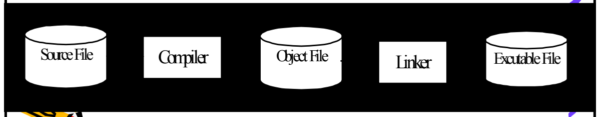
9

Compiling Source Code

•A program written in a high-level language is called a *source program (or source code).*

•Program called a *compiler* is used to translate the source program into a machine language program called an *object program.*

•The object program is often then linked with other supporting library code before the object can be executed on the machine.



10

Source program

- The form in which a computer program, written in some formal programming language, is written by the programmer.
- Can be compiled automatically into machine code or executed by an interpreter.

A Linker

- A program that pulls other programs together so that they can run.
- Most programs are very large and consist of several modules.
- Even small programs use existing code provided by the programming environment called libraries.
- The linker pulls everything together, makes sure that references to other parts of the program (code) are resolved.

Compilers & Programs

- **Object program**
  - Output from the compiler
  - Equivalent machine language translation of the source program
- **Executable program**
  - Output from linker/loader
  - Machine language program linked with necessary libraries & other files
  - Files usually have extension '.exe'

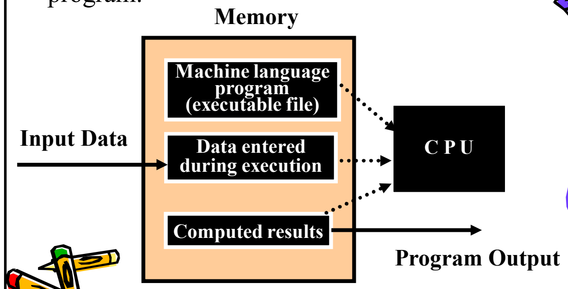
The Concept of Interpretation

- An **interpreter** is a simple program.
- It does not translate the source code into machine code.
- it reads the source code program line by line and executes it.
- Therefore, an **interpreter is also called a program execution environment.**

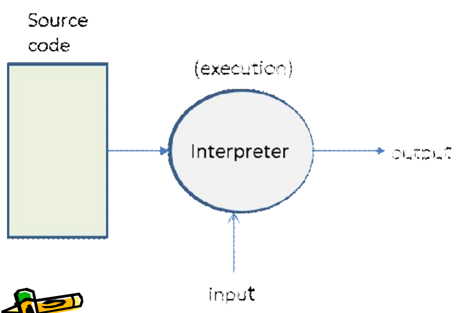
16

Running Programs

- Steps that the computer goes through to run a program:


Memory
Machine language program (executable file)
Data entered during execution
Computed results
Input Data
C P U
Program Output

The Concept of Interpretation



17

Program Execution

- Steps taken by the CPU to run a program (instructions are in machine language):
  1. Fetch an instruction
  2. Decode (interpret) the instruction
  3. Retrieve data, if needed
  4. Execute (perform) actual processing
  5. Store the results, if needed

Compilers Vs Interpreters

- Compilers
  - Translate the program before it's executed.
  - When programs are compiled, they are translated all at once.
  - Compiled programs typically execute more quickly than interpreted programs, but have a slower translation speed

Compilers Vs Interpreters

- Interpreters
- Translate programs line-by-line instead of all at once (like compiled programs).
- Interpreted programs generally translate quicker than compiled programs, but have a slower execution speed.

Program Errors

- Syntax Errors:
  - Errors in grammar of the language
- Runtime error:
  - When there are no syntax errors, but the program can't complete execution
    - Divide by zero
    - Invalid input data
- Logical errors:
  - The program completes execution, but delivers incorrect results
  - Incorrect usage of parentheses

Syntax & Semantics

- Syntax:
  - The structure of strings in some language. A language's syntax is described by a grammar.
  - Examples:
    - Binary number

```
<binary_number>        = <bit> | <bit>
  <binary_number>
<bit>                  = 0 | 1
```

    - Identifier

```
<identifier> = <letter> {<letter> | <digit> }
<letter>               = a | b | . . . | z
<digit                 = 0 | 1 | . . . | 9
```

- Semantics:
  - The meaning of the language

Programming Domains

- **Business applications**:
  - These languages offer features which are suitable for business application development.
  - For instance, COBOL (common business-oriented language**)** was very widely used for developing business applications.
  - It has a strong support for file handling, a much needed feature by application programmers.
  - It also supports long variable and file names resulting in an easily readable program.
  - RPG is another language used for business application development.

23

Syntax & Grammars

- Syntax descriptions for a PL are themselves written in a formal language.
- The formal language is not a PL but it can be implemented by a compiler to enforce grammar restrictions.
- Some PLs look more like grammar descriptions than like instructions.

Programming Domains

- **Scientific applications**:
  - These languages support a wide variety of data types i.e., very large and very small numbers can be easily represented.
  - Example: ALGOL 60 and FORTRAN programming languages are very widely used to develop scientific applications.
  - Even 'C' and C++ can be used for this purpose.

Programming Domains (contd.)

- **Programming languages for artificial intelligence**:
  - These languages are used to represent logical statements and their manipulation to arrive at a particular inference.
  - LISP and PROLOG languages are widely used languages for this purpose.

25

Programming Domains (contd.)

- **Parallel Programming languages**:
  - These languages are designed to program algorithms and applications that can run on parallel computers.
  - The program is written as a set of concurrent tasks.
  - These languages are also called concurrent programming languages. Examples: Concurrent Pascal, Occam, Parallel Fortran, and Hope

Generations of Programming Languages

- The programming languages can be classified into five generations of programming languages.

  1. First generation Languages (1GL)
  2. Second generation Languages(2GL)
  3. Third generation Languages (3GL)
  4. Fourth generation Languages (4GL)
  5. Fifth generation languages (5GL)

27

First Generation Languages (1GL)

- The machine language comprising 0s and 1s is termed a 1st generation computer language.
- It is a machine-dependent language designed to program as per the architecture of the processor of the machine
- Example machine instruction: 11010111 10011011
- Since it is a machine language, no translation is required and the program runs very fast and efficiently on the machine

28

Second Generation Languages (2GL)

- Assembly languages are called 2nd generation programming languages.
- An assembly language is a sugared form of the machine language.
- It uses symbols, called **mnemonics**, to write the programming code.
- The symbolic codes are easier to comprehend as compared to the code written using binary language.

29

Third Generation Languages (3GL)

- These languages are also called high-level languages.
- The architecture of a 3GL is different from that of the underlying machine.
- As discussed earlier, 3GL (high-level languages) are further divided into procedural, object oriented, and problem oriented languages.
- Examples of 3GL are: 'C', 'C++', Java, Python, etc.

30

Fourth Generation Languages (4GL)

- 4GL have been developed keeping in view the following observation:
  - 4GL offer non-procedural constructs such as 'sort', 'index', 'search', 'create table' etc.
  - A non-procedural construct performs a task for which the programmer need not provide the programming logic.
  - The language has built-in logic for the construct.
  - Thus, the programming effort is drastically reduced.

31

The timeline of the development of some popular programming languages

	Language	Remarks
1950-56	Assembly languages	Low level languages
1957	FORTRAN	High level language for Sc. Applications
1958	ALGOL	Algorithmic language supporting block structures
1960	COBOL, LISP	Business programming and List processing
1962	Simula	Simulation programming, first OOP language
1964	BASIC	General purpose, easy to program language
1970	PROLOG, Hope	Logic programming language suitable for AI applications Hope was a functional programming language
1972	'C'	High level language suitable for system programming
1973	PASCAL	Block structured language

34

Fourth Generation Languages (4GL)

- For instance, a programmer can issue a statement such as that given below:
  - 'Search all records where Marks > 70'
- It may be noted that the programmer has not supplied the procedure for searching into the file or data base.
- In an equivalent 3GL program, the programmer would have had to provide some lines of code for this task.
- Examples of 4GL are: FoxPro, SQL, MATLAB, Oracle Reports etc.

The timeline of the development of some popular programming languages

	Language	Remarks
1983	Smalltalk, Ada	OOP languages
1984	ML	Functional programming
1986	C++, Eiffel	OOP languages
1990	Haskell	Functional programming
1990-1995	Perl, Python, JavaScript, PHP	Scripting languages
1995	Java	OOP language suitable for Internet programming
2000	C#	A multi-paradigm programming language
2005-2006	Ruby on Rails	Web application framework
2010	(Standard) PHP	Scripting language
2014	iOS/swift	Programming language for iOS and OS X developers

Programming in C

Fifth Generation Languages (5GL)

- designed to develop machines that behave like humans.
- these machines are capable of learning and self-organization.
- Therefore, the machines are also called artificially intelligent machines.
- Artificial Intelligence is a branch of computer science concerned with making computers behave like humans.
  - Examples of 5GL are: LISP and Prolog.

33

Questions