Lecture Notes
Subject: Introduction to Programming Methodology
Broad Classification of Computer Languages

- Machine language
- Assembly language
- High-level language
Machine Language

- Only language of a computer understood by it without using a translation program
- Normally written as strings of binary 1s and 0s
- Written using decimal digits if the circuitry of the computer being used permits this
A Typical Machine Language Instruction Format

<table>
<thead>
<tr>
<th>OPCODE</th>
<th>OPERAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>(operation code)</td>
<td>(Address/Location)</td>
</tr>
</tbody>
</table>

- OPCODE tells the computer which operation to perform from the instruction set of the computer.
- OPERAND tells the address of the data on which the operation is to be performed.
<table>
<thead>
<tr>
<th>In Binary</th>
<th>In Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>0010000000000001100111001</td>
<td>10001471</td>
</tr>
<tr>
<td>0011000000000010000100001</td>
<td>14002041</td>
</tr>
<tr>
<td>0110000000000011100101110</td>
<td>30003456</td>
</tr>
<tr>
<td>101000111111011100101110</td>
<td>50773456</td>
</tr>
<tr>
<td>0000000000000000000000000000000000000000000</td>
<td>0000000000</td>
</tr>
</tbody>
</table>
Advantages & Limitations of Machine Language

**Advantage**

- Can be executed very fast

**Limitations**

- Machine Dependent
- Difficult to program
- Error prone
- Difficult to modify
Programming language that overcomes the limitations of machine language programming by:

- Using alphanumeric mnemonic codes instead of numeric codes for the instructions in the instruction set
e.g. using ADD instead of 1110 (binary) or 14 (decimal) for instruction to add

- Allowing storage locations to be represented in form of alphanumeric addresses instead of numeric addresses
e.g. representing memory locations 1000, 1001, and 1002 as FRST, SCND, and ANSR respectively

- Providing pseudo-instructions that are used for instructing the system how we want the program to be assembled inside the computer’s memory
e.g. START PROGRAM AT 0000; SET ASIDE AN ADDRESS FOR FRST
Assembler

• Software that translates as assembly language program into an equivalent machine language program of a computer.

Assemble language program → Input → Assembler → Output → Machine language program

(Source Program) → One-to-one correspondence → (Object Program)
### An Example of Assembly Language Program

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Opcode</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLT</td>
<td>00</td>
<td>Halt, used at the end of program to stop</td>
</tr>
<tr>
<td>CLA</td>
<td>10</td>
<td>Clear and add into A register</td>
</tr>
<tr>
<td>ADD</td>
<td>14</td>
<td>Add to the contents of A register</td>
</tr>
<tr>
<td>SUB</td>
<td>15</td>
<td>Subtract from the contents of A register</td>
</tr>
<tr>
<td>STA</td>
<td>30</td>
<td>Store A register</td>
</tr>
</tbody>
</table>

A subset of the set of instructions supported by a computer
Advantages of Assembly Language Over Machine Language

- Easier to understand and use
- Easier to locate and correct errors
- Easier to modify
- No worry about addresses
- Easily relocatable
- Efficiency of machine language
Limitations of Assembly Language

- Machine dependent
- Knowledge of hardware required
- Machine level coding

Typical Uses of Assembly Language

- Mainly used today to fine-tune important parts of programs written in a high-level language to improve the program’s execution efficiency
High-Level Languages

- Machine independent
- Do not require programmers to know anything about the internal structure of computer on which high-level language programs will be executed
- Deal with high-level coding, enabling the programmers to write instructions using English words and familiar mathematical symbols and expressions
• Translator program (software) that translates a high-level language program into its equivalent machine language program

• Compiles a set of machine language instructions for every program instruction in a high-level language
Compiler

High-level language program → Input → Compiler → Output → Machine language program

(Source Program) → One-to-many correspondence → (Object Program)
A computer supporting languages L1 and L2

Illustrating the requirement of a separate compiler for each high-level language supported by a computer
Program P1 in high-level language L1

Compiler for language L1 on computer A -> Machine code for P1 that will run on computer A -> Executed on computer A

Compiler for language L1 on computer B -> Machine code for P1 that will run on computer B -> Executed on computer B

Same results obtained

Illustrating the machine independence characteristic of a high-level language. Separate compilers are required for the same language on different computers.
Interpreter

- Interpreter is a high-level language translator
- Takes one statement of a high-level language program, translates it into machine language instructions
- Immediately executes the resulting machine language instructions
- Compiler simply translates the entire source program into an object program and is not involved in its execution
Advantages of High-Level Languages

- Machine independent
- Easier to learn and use
- Fewer errors during program development
- Lower program preparation cost
- Better documentation
- Easier to maintain

Limitations of High-Level Languages

- Lower execution efficiency
- Less flexibility to control the computer’s CPU, memory and registers
Object-Oriented Programming Languages

- Programming languages are used for simulating real-world problems on computers
- Much of the real world is made up of objects
- Essence of OOP is to solve a problem by:
  - Identifying the real-world objects of the problem
  - Identifying processing required of them
  - Creating simulations of objects, processes, and their communications
FORTRAN

- Stands for FORMULA TRANslating
- Originally developed by John Backus and his team at IBM followed by several revisions
- Standardized by ANSI as FORTRAN-77 and FORTRAN-90
- Designed for solving scientific & engineering problems
- Oriented towards solving problems of a mathematical nature
- Popular language amongst scientists and engineers
- Stands for **CO**mmonly **B**usiness **O**riented **L**anguage
- Originally developed started under Grace Hopper followed by C**ON**ference on D**A**ta S**Y**stems L**a**nguages (CODASYL)
- Standardized by ANSI as COBOL-74, COBOL-85, and COBOL-2002
- Designed for programming business data processing applications
- Designed to have the appearance and structure of a business report written in English, hence often referred to as a self-documenting language
BASIC

- Stands for Beginners All-purpose Symbolic Instruction Code
- Developed by Professor John Kemeny and Thomas Kurtz at Dartmouth College in the United States
- Standardized by ANSI as BASIC-78
- Designed to be an interactive language and to use an interpreter instead of a compiler
- Simple to implement, learn and use language. Hence, it is a widely used language on personal computers
- Flexible and reasonably powerful language and can be used for both business and scientific applications
Pascal

- Named after the famous seventeenth-century French mathematician Blaise Pascal
- Developed by Professor Nicklaus Wirth of Federal Institute of Technology in Zurich
- Encourages programmers to write well-structured, modular programs, instills good program practices
- Recognized as an educational language and is used to teach programming to beginners
- Suitable for both scientific & business applications
- Has features to manipulate numbers, vectors, matrices, strings, sets, records, files, and lists
• Developed in 1972 at AT&T’s Bell laboratories, USA by Dennis Ritchie and Brian Kernighan
• Standardized by ANSI and ISO as C89, C90, C99
• High-level programming languages (mainly machine independence) with the efficiency of an assembly language
• Language of choice of programmers for portable systems software and commercial software packages like OS, compiler, spreadsheet, word processor, and database management systems
A Sample C Program

/* PROGRAM TO COMPUTE THE SUM OF 10 NUMBERS */
/* Directives to include standard library and header */
#include <stdlib.h>
#include <stdio.h>
/* Main function starts here */
void main ( )
{
    /* Declaration of variables */
    float Sum = 0.0, N = 0.0;
    int Count = 0;
    for (Count = 0; Count < 10; Count++)
    {
        printf("Give a number:");
        scanf("%f", N);
        Sum += N;
    }
    printf("THE SUM OF GIVEN NUMBERS = %f", &Sum);
}
Java

- Development started at Sun Microsystems in 1991 by a team led by James Gosling
- Developed to be similar to C++ with fewer features to keep it simple and easy to use
- Compiled code is machine-independent and developed programs are simple to implement and use
- Uses *just-in-time* compilation
- Used in embedded systems such as hand-held devices, telephones and VCRs
- Comes in two variants – Java Runtime Engine (JRE) and Java Software Development Kit (SDK)
SNOBOL

- Stands for String Oriented Symbolic Language
- Used for non-numeric applications
- Powerful string manipulation features
- Widely used for applications in the area of text processing
Characteristics of a Good Programming Language

- Simplicity
- Naturalness
- Abstraction
- Efficiency
- Structured Programming Support
- Compactness
- Locality
- Extensibility
- Suitability to its environment
Computer Software
Software

- **Hardware** refers to the physical devices of a computer system.
- **Software** refers to a collection of programs.
- **Program** is a sequence of instructions written in a language that can be understood by a computer.
- **Software package** is a group of programs that solve a specific problem or perform a specific type of job.
Both hardware and software are necessary for a computer to do useful job. They are complementary to each other.

Same hardware can be loaded with different software to make a computer system perform different types of jobs.

Except for *upgrades*, hardware is normally a one-time expense, whereas software is a continuing expense.

Upgrades refer to renewing or changing components like increasing the main memory, or hard disk capacities, or adding speakers, modems, etc.
Types of Software

Most software can be divided into two major categories:

- **System software** are designed to control the operation and extend the processing capability of a computer system.

- **Application software** are designed to solve a specific problem or to do a specific task.
System Software

- Make the operation of a computer system more effective and efficient
- Help hardware components work together and provide support for the development and execution of application software
- Programs included in a system software package are called **system programs** and programmers who prepare them are called **system programmers**
- Examples of system software are operating systems, programming language translators, utility programs, and communications software
Application Software

- Solve a specific problem or do a specific task
- Programs included in an application software package are called *application programs* and the programmers who prepare them are called *application programmers*
- Examples of application software are word processing, inventory management, preparation of tax returns, banking, etc.
Logical System Architecture

Relationship among hardware, system software, application software, and users of a computer system.
Developing a software and putting it to use is a complex process and involves following steps:

- Analyzing the problem at hand and planning the program(s) to solve the problem
- Coding the program(s)
- Testing, debugging, and documenting the program(s)
- Implementing the program(s)
- Evaluating and maintaining the program(s)
Firmware

- Firmware is software substituted for hardware and stored in read-only memory
- Firmware technology has enabled production of various types of smart machines having microprocessor chips with embedded software
Basic idea is to have a separate software layer to:

- Act as “glue” between client and server parts of application
- Provide programming abstraction
- Mask heterogeneity of underlying network, hardware, and OS
- Encourages three-tier software architecture against two-tier popularized by Server-Client architecture
Planning the Computer Program
Algorithm

- Refers to the logic of a program and a step-by-step description of how to arrive at the solution of a given problem

- In order to qualify as an algorithm, a sequence of instructions must have following characteristics:
  - Each and every instruction should be precise and unambiguous
  - Each instruction should be such that it can be performed in a finite time
  - One or more instructions should not be repeated infinitely. This ensures that the algorithm will ultimately terminate
  - After performing the instructions, that is after the algorithm terminates, the desired results must be obtained
There are 50 students in a class who appeared in their final examination. Their mark sheets have been given to you.

The division column of the mark sheet contains the division (FIRST, SECOND, THIRD or FAIL) obtained by the student.

Write an algorithm to calculate and print the total number of students who passed in FIRST division.
Sample Algorithm (Example 1)

Step 1: Initialize Total_First_Division and Total_Marksheets_Checked to zero.

Step 2: Take the mark sheet of the next student.

Step 3: Check the division column of the mark sheet to see if it is FIRST, if no, go to Step 5.

Step 4: Add 1 to Total_First_Division.

Step 5: Add 1 to Total_Marksheets_Checked.

Step 6: Is Total_Marksheets_Checked = 50, if no, go to Step 2.

Step 7: Print Total_First_Division.

Step 8: Stop.
Flowchart is a pictorial representation of an algorithm.

- Uses symbols (boxes of different shapes) that have standardized meanings to denote different types of instructions.
- Actual instructions are written within the boxes.
- Boxes are connected by solid lines having arrow marks to indicate the exact sequence in which the instructions are to be executed.
- Process of drawing a flowchart for an algorithm is called flowcharting.
Basic Flowchart Symbols

- Terminal
- Input/Output
- Processing
- Decision
- Flow lines
- Connectors
Examples of Decision Symbol

(a) A two-way branch decision.

(b) A three-way branch decision.
50 students of a class appear in the examination of Example 3.

Draw a flowchart for the algorithm to calculate and print the percentage marks obtained by each student along with his/her roll number and name.
Flowchart for the solution of Example 4 with an infinite (endless) process loop.

Start

Read input data

Add marks of all subjects giving Total

Percentage = Total / 10

Write output data
Flowcharting Rules

- First chart the main line of logic, then incorporate detail.
- Maintain a consistent level of detail for a given flowchart.
- Do not chart every detail of the program. A reader who is interested in greater details can refer to the program itself.
- Words in the flowchart symbols should be common statements and easy to understand.
Flowcharting Rules

- Be consistent in using names and variables in the flowchart
- Go from left to right and top to bottom in constructing flowcharts
- Keep the flowchart as simple as possible. Crossing of flow lines should be avoided as far as practicable
- If a new flowcharting page is needed, it is recommended that the flowchart be broken at an input or output point.
- Properly labeled connectors should be used to link the portions of the flowchart on different pages
Advantages of Flowchart

- Better Communication
- Proper program documentation
- Efficient coding
- Systematic debugging
- Systematic testing
Limitations of Flowchart

- Flowcharts are very time consuming and laborious to draw (especially for large complex programs)
- Redrawing a flowchart for incorporating changes/modifications is a tedious task
- There are no standards determining the amount of detail that should be included in a flowchart
Pseudocode

- A program planning tool where program logic is written in an ordinary natural language using a structure that resembles computer instructions.

- "Pseudo" means imitation or false and "Code" refers to the instructions written in a programming language. Hence, pseudocode is an imitation of actual computer instructions.

- Because it emphasizes the design of the program, pseudocode is also called Program Design Language (PDL).
Any program logic can be expressed by using only following three simple logic structures:

1. Sequence logic,
2. Selection logic, and
3. Iteration (or looping) logic

Programs structured by using only these three logic structures are called \textit{structured programs}, and the technique of writing such programs is known as \textit{structured programming}.
It is used for performing instructions one after another in sequence.

(a) Flowchart

(b) Pseudocode
Also known as decision logic, it is used for making decisions.

Three popularly used selection logic structures are:
1. IF...THEN...ELSE
2. IF...THEN
3. CASE
Selection Logic (IF...THEN...ELSE Structure)

(a) Flowchart

- IF (condition)
  - THEN Process 1
  - ELSE Process 2

(b) Pseudocode

- IF Condition
  - THEN Process 1
  - ELSE Process 2

ENDIF
Iteration (or Looping) Logic

- Used to produce loops in program logic when one or more instructions may be executed several times depending on some conditions.

- Two popularly used iteration logic structures are:
  1. DO...WHILE
  2. REPEAT...UNTIL
Iteration (or Looping) Logic
(DO...WHILE Structure)

(a) Flowchart

(b) Pseudocode

```
DO WHILE Condition
    Process 1
    Process n
ENDDO
```
Iteration (or Looping) Logic
(REPEAT...UNTIL Structure)

(a) Flowchart

(b) Pseudocode
Advantages of Pseudocode

- Converting a pseudocode to a programming language is much more easier than converting a flowchart to a programming language.

- As compared to a flowchart, it is easier to modify the pseudocode of a program logic when program modifications are necessary.

- Writing of pseudocode involves much less time and effort than drawing an equivalent flowchart as it has only a few rules to follow.
Limitations of Pseudocode

- In case of pseudocode, a graphic representation of program logic is not available.
- There are no standard rules to follow in using pseudocode.
- Different programmers use their own style of writing pseudocode and hence communication problem occurs due to lack of standardization.
- For a beginner, it is more difficult to follow the logic of or write pseudocode, as compared to flowcharting.
Operating Systems
Definition and Need for OS

- Integrated set of programs that controls the resources (the CPU, memory, I/O devices, etc.) of a computer system
- Provides its users with an interface or virtual machine that is more convenient to use than the bare machine
- Two primary objectives of an OS are:
  - Making a computer system convenient to use
  - Managing the resources of a computer system
The operating system layer hides the details of the hardware from the programmer and provides the programmer with convenient interface for using the system.
Main Functions of an OS

- Process management
- Memory management
- File management
- Security
- Command interpretation
Parameters for Measuring System Performance

- **Throughput**: Amount of work that the system is able to do per unit time

- **Turnaround time**: Interval from the time of submission of a job to the system for processing to the time of completion of the job

- **Response time**: Interval from the time of submission of a job to the system for processing to the time the first response for the job is produced by the system
Process Management

- A **process** (also called **job**) is a program in execution.
- **Process management** manages the processes submitted to a system in a manner to minimize *idle time* of processors (CPUs, I/O processors, etc.) of the system.
Process States in Multiprogramming

- New job → Ready
  - Job is allocated the CPU for execution
  - I/O completed → Blocked
    - Job must wait for I/O completion
  - Blocked → Running
    - Job processing completed
Memory is important resource of a computer system that must be properly managed for the overall system performance.

Memory management module:
- Keeps track of parts of memory in use and parts not in use.
- Allocates memory to processes as needed and deallocates when no longer needed.
A file is a collection of related information

Every file has a name, its data and attributes

File’s name uniquely identifies it in the system and is used by its users to access it

File’s data is its contents

File’s attributes contain information such as date & time of its creation, date & time of last access, date & time of last update, its current size, its protection features, etc.

File management module of an operating system takes care of file-related activities such as structuring, accessing, naming, sharing, and protection of files
File Operations

- Set of commands provided by an operating system to deal with files and their contents
- Typical file operations include create, delete, open, close, read, write, seek, get attributes, set attributes, rename, and copy
Deals with protecting the various resources and information of a computer system against destruction and unauthorized access.

**External security:** Deals with securing computer against external factors such as fires, floods, earthquakes, stolen disks/tapes, etc. by maintaining adequate backup, using security guards, allowing access to sensitive information to only trusted employees/users, etc.

**Internal security:** Deals with user authentication, access control, and cryptography mechanisms.
Security

- **User authentication:** Deals with the problem of verifying the identity of a user (person or program) before permitting access to the requested resource

- **Access Control:** Once authenticated, access control mechanisms prohibit a user/process from accessing those resources/information that he/she/it is not authorized to access

- **Cryptography:** Means of encrypting private information so that unauthorized access cannot use information
Command Interpretation

- Provides a set of commands using which the user can instruct the computer for getting some job done.
- Commands supported by the command interpretation module are known as **system calls**.
Two types of user interfaces supported by various operating systems are:

- **Command-line interface**: User gives instructions to the computer by typing the commands.

- **Graphical User Interface (GUI)**: User gives commands to the system by selecting icon or menu item displayed on the screen with the use of a point-and-draw device.
MS-DOS

- Stands for Microsoft Disk Operating System.
- Single-user OS for IBM and IBM-compatible personal computers (PC)
- Structured in three layers – BIOS (Basic Input Output System), kernel, and shell
- Very popular in the 1980s, now not in much use and development with the launch of Microsoft Windows OS in 1990s
Microsoft Windows

- Developed by Microsoft to overcome limitations of MS-DOS operating system
- Single-user, multitasking OS
- Native interface is a GUI
- Designed to be not just an OS but also a complete operating environment
- OS of choice for most PCs after 1990
File Concept
Data Storage Hierarchy

- **Level 0**: Bit
  - A single binary digit (0 or 1)

- **Level 1**: Character
  - Multiple related bits are combined to form a character (byte)

- **Level 2**: Field
  - Multiple related characters are combined to form a field

- **Level 3**: Record
  - Multiple related fields are combined to form a record

- **Level 4**: File
  - Multiple related records are combined to form a file

- **Level 5**: Database
  - Multiple related files are integrated to form a database
Relationship Among Character, Field, Record, and File

A record

Records of a file

A field having 4 characters
A file management system supports following file types:

- **Transaction file**: Stores input data until it can be processed
- **Master file**: Contains all current data relevant to an application
- **Output file**: Stores output produced by one program that is used as input to another program
- **Report file**: Holds a copy of a report generated by an application
- **Backup file**: Copy of a file, created as a safety precaution against loss of data
File Organizations

- File organization is the physical organization of the records of a file for convenience of storage and retrieval of data records.
- Three commonly used file organizations are:
  - **Sequential**: Records are stored one after another in ascending or descending order determined by the value of the key field of the records.
  - **Direct/random**: Desired record pertaining to current transaction can be directly located by its key field value without having to navigate through sequence of other records.
Indexed sequential: There are two files for every data file – the data file which contains the records stored in the file, and the smaller index file which contains the key and disk address of each record stored in the data file.