Parul® University

System Programming

Prof. Uma Bhatt, Assistant Professor
Information Technology Engineering

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-1

Overview of System Software

Parul® University

DIGITAL LEARNING CONTENT

Topics to be covered

- Introduction
- Software
- Software Hierarchy
- Systems Programming
- Machine Structure
- Interfaces
- Address Space
- Computer Languages
- Life Cycle of a Source Program
- System Software Development
- Recent Trends in Software Development
- Levels of System Software

Parul® University

DIGITAL LEARNING CONTENT

Introduction

- A Set of instructions to perform specific tasks is called a program.

- The Collection of one or many programs for a specific purpose is termed as Computer Software or Simply Software.

- In general , **Software** is a set of computer programs which are designed and developed to perform specific task desired by the user or by the computer itself.

Parul® University

DIGITAL LEARNING CONTENT

Software Characteristics

Operational Characteristics

Includes characteristics such as correctness, usability/learnability, integrity , reliability, efficiency , security and safety.

Transitional Characteristics

Includes interoperability, reusability and portability.

Revision Characteristics

These are the characteristics related to ‘ interior quality ’ of software such as efficiency , documentation , and structure. Various revision characteristics of software are maintainability, flexibility , extensibility, scalability, Testability and modularity.

Types of Software

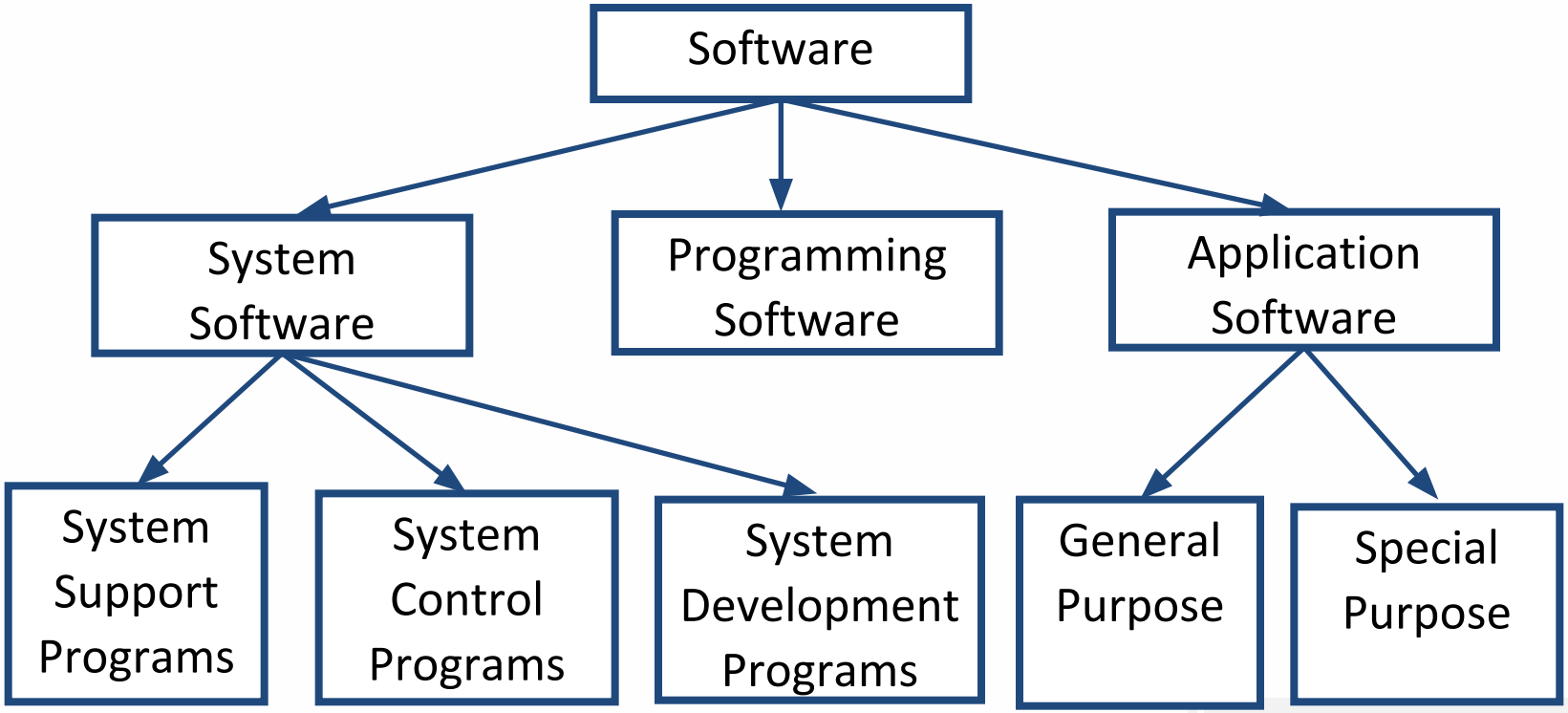


Figure 1.1 : Types of Software[1]

Parul®
University

DIGITAL LEARNING CONTENT

System Software

- The system software is collection of programs designed to operate, control and extend the processing capabilities of the computer itself.
- These are generally prepared by computer manufacturers.
- These software perform a variety of functions like file editing, storage management, resource accounting, I/O management, etc.

Parul® University

DIGITAL LEARNING CONTENT

Types of System Software

- **System Control Programs** : They control the execution of programs, manage the storage and processing resources of the computer and perform other management and monitoring functions. e.g., OS

- **System Support Programs** : They provide routine service functions to other computer programs and computer users. e.g., Utility Programs

- **System Development Programs** : They assist in the creation of publication programs. e.g., Language translators like interpreters, compilers and assemblers

System Control Programs - OS

- An operating system is an integrated set of specialized programs that are used to manage overall resources of and operations of the computer.

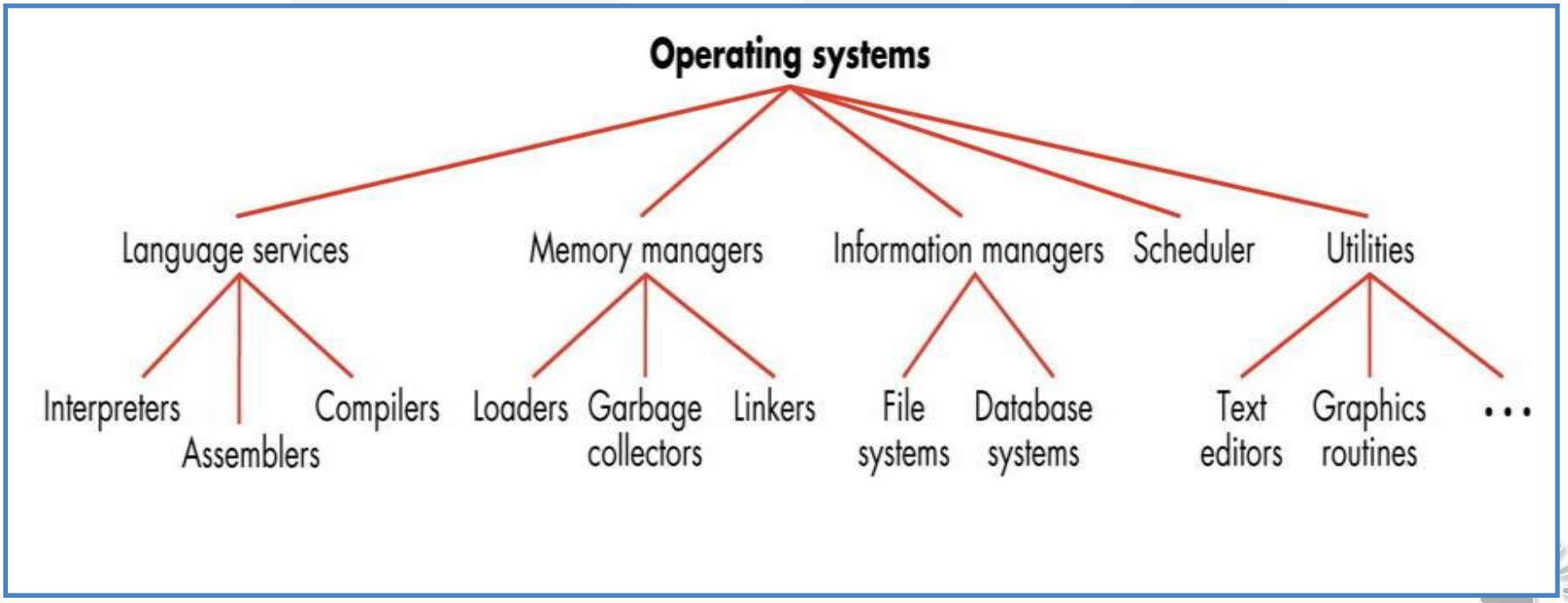


Figure 1.2 : Operating System Services[2]

Parul® University

DIGITAL LEARNING CONTENT

System Support Programs – Utility Programs

- Utility programs/Service programs are provided by the computer manufacturers to perform tasks that are common to all data processing installations. The tasks performed are :

- Sorting data
- Editing the output
- Dumping of data to disc/tape
- Tracing the operation of program
- Copying data from one recording medium to another

System Development Programs – Language Translator

- Language translators are also called language processors.

- **Main functions** are :

  Translate high level language to low level language.

  Check for and identity syntax errors that may be present in the program being installed.

- There are **3 types of translator programs-**

1. Assembler
2. Interpreter
3. Compiler

Assembler

- Translates a source program into a corresponding object program.
- Convert symbolic op codes to binary
- Convert symbolic addresses to binary
- Perform assembler services requested by the pseudo-ops
- Put translated instructions into a file for future use

Interpreter

- A language translator that translates one program statement at a time into machine code.

- Interpreters usually take less amount of time to analyse the source code. However, the overall execution time is comparatively slower than compilers.

- Programming languages like JavaScript, Python, Ruby use interpreters.

Compiler

- A language translator that converts a complete program into machine language to produce a program that the computer can process in its entirely.
- Scans the entire program and translates it as a whole into machine code.
- Compilers usually take a large amount of time to analyse the source code. However, the overall execution time is comparatively faster than interpreters.
- Programming languages like C, C++, Java use compilers.

Parul® University

DIGITAL LEARNING CONTENT

Programming and Application Software

Programming Software :

- Tools provided to the programmer.
- E.g compilers, Debuggers, Interpreters, Linkers, Text editors

Application Software :

- End user oriented
- Provide specific function
- E.g microsoft office
- Computer games
- Media players

System Software VS Application Software

Table 1.1 : System Software Vs Application Software

System Software	Application Software
It is the type of software which is the interface between application software and system.	It is the type of software which runs as per user request. It runs on the platform which is provide by system software.
It is used for operating computer hardware.	It is used by user to perform specific task.
System software are installed on the computer when operating system is installed.	Application software are installed according to user's requirements.
It can run independently. It provides platform for running application software.	It can't run independently. They can't run without the presence of system software..
Each computer must have a system software to function.	Without an application software the computer is still able to function.
Examples of system software's are compiler, assembler, debugger, driver, etc.	Examples of application software's are word processor, web browser, media player, etc.

Software Hierarchy

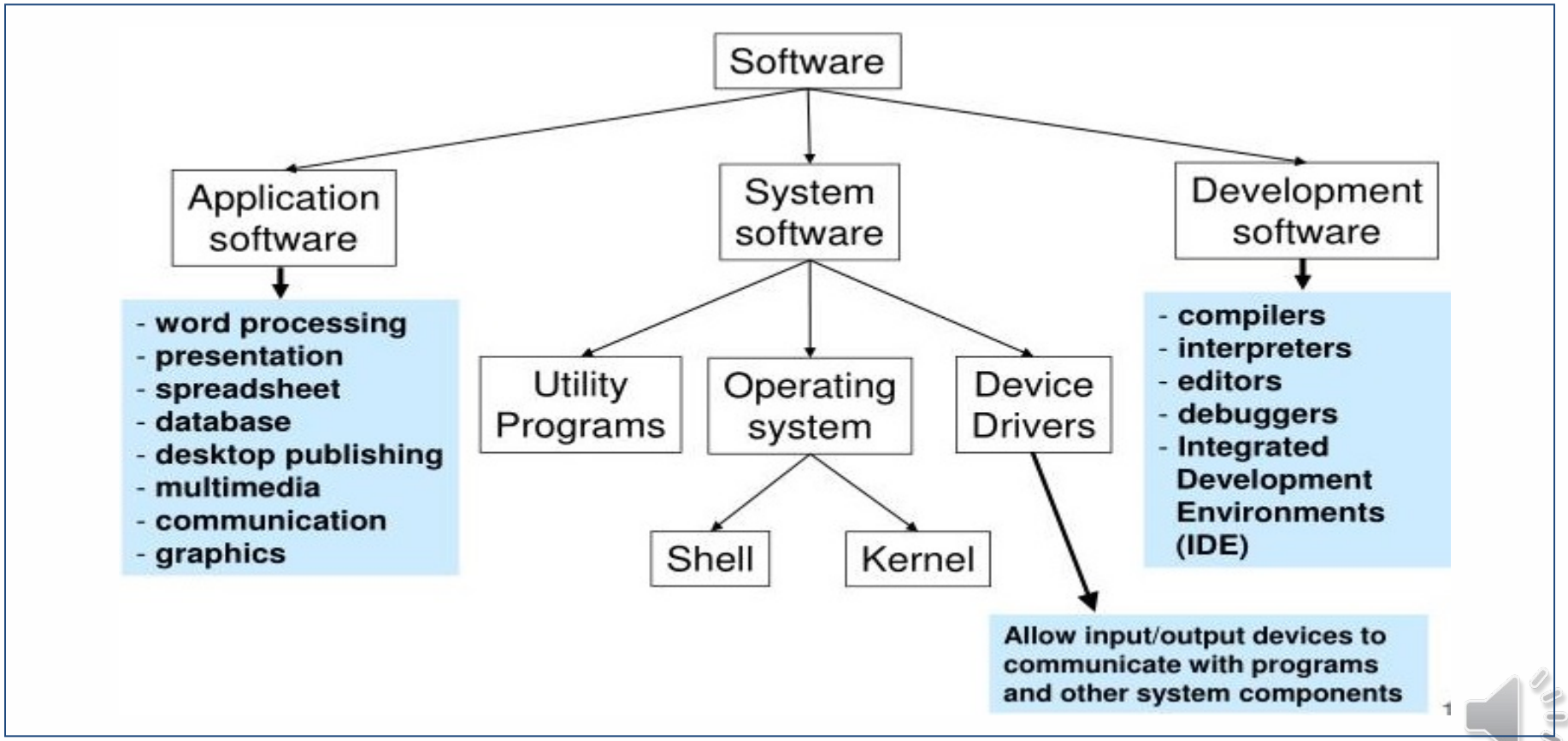


Figure 1.3 :Software Hierarchy[1]

Systems Programming

- Systems programming involves the development of the individual pieces of software that allow the entire system to function as a single unit.
- System programming involves designing and writing computer programs that allow the computer hardware to interface with the programmer and the user, leading to the effective execution of application software on the computer system.
- Typical system programs include the operating system and firmware, programming tools such as compilers, assemblers, I/O routines, interpreters, scheduler, loaders and linkers as well as the runtime libraries of the computer programming languages.
- Systems programming deals with “low-level” programming.

Essential Characteristics of Systems Programming

- Programmers are expected to know the hardware and internal behavior of the computer system on which the program will run.
- Uses a Low Level Programming Language.
- Requires little run time overhead and can execute in a resource constrained environment.
- These are very efficient programs with a small or no runtime library requirements.
- Has access to system resource including Memory.
- Can be written in Assembly Language.

Limiting Factors of Systems Programming

- Limited Programming facility is available , which requires high skills for the system Programmer.
- Less Powerful runtime Library (if available at all), with less error checking capability.

General Machine Structure

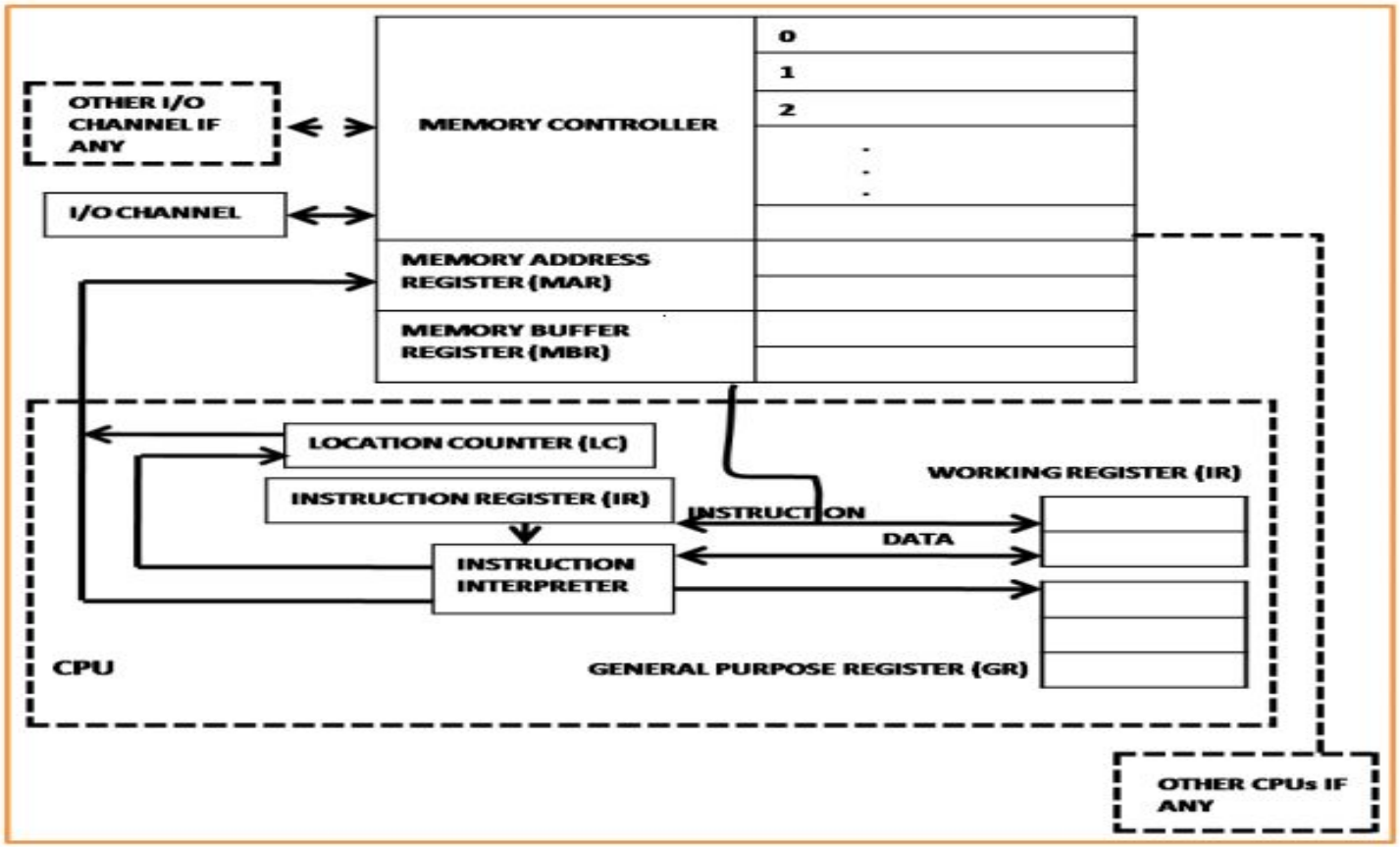


Figure 1.4 : General Machine Structure [2]

Contd..

The components of a general machine are as follows:

1. **Instruction interpreter**: The Instruction Interpreter Hardware is basically a group of circuits that perform the operation specified by the instructions fetched from the memory.

2. **Location counter**: LC or instruction counter IC, is a hardware memory device which denotes the location of the current instruction being executed.

3. **Instruction register**: A copy of the content of the LC is stored in IR.

4. **Working register**: The working registers are often called as the "scratch pads" because they are used to store temporary values while calculation is in progress.

Contd..

5. General register: are used by programmers as storage locations and for special functions.

6. Memory address registers (MAR): contains the address of the memory location that is to read from or stored into.

7. Memory buffer register (MBR): contains copy of address specified by MAR

8. Memory controller: It is used to transfer data between MBR & the memory location specified by MAR.

9. I/O channels: The role of I/O Channels is to input or output information from memory.

Interfaces

- The connection and interaction between hardware, software and the user.
- Thin layer between two components of computer which exchange information.
- Interface has been defined as a border or an entry point across which different components of a digital computing system interchange data and information.
- Users "talk to" the software. The software "talks to" the hardware and other software. Hardware "talks to" other hardware. All this is interfacing.
- It has to be designed, developed, tested and redesigned.

Contd..

Hardware Interfaces : When referring to hardware, an **interface** is a physical device, port, or connection that interacts with the computer or other hardware device.

Hardware interfaces are the plugs, sockets, cables and electrical signals traveling through them. Examples are USB, FireWire, Ethernet, ATA/IDE, SCSI and PCI.

Software/Programming Interfaces : Software interfaces (programming interfaces) are the languages, codes and messages that programs use to communicate with each other and to the hardware.

Examples are the Windows, Mac and Linux operating systems, SMTP email, IP network protocols and the software drivers that activate the peripheral devices.

Contd..

User Interfaces :

- User interfaces are the keyboards, mice, commands and menus used for communication between you and the computer.

- Examples are the command lines in DOS and Unix, and the graphical interfaces in Windows, Mac and Linux.

Contd..

User Interfaces :

- User interfaces are the keyboards, mice, commands and menus used for communication between you and the computer.

- Examples are the command lines in DOS and Unix, and the graphical interfaces in Windows, Mac and Linux.

Address Space

- **Address space** is the amount of memory allocated for all possible **addresses** .
- **Address space** may refer to a range of either physical or virtual **addresses** accessible to a processor or reserved for a process.
- The computational entities such as a device, file , server or networked computer all are addressed within Space.

Address Space

Table : 1.2 Physical Vs Logical Address Space

Physical Address Space	Logical Address Space
Physical address space is the collection of all physical addresses produced by a computer program and provided by the hardware.	Logical Address Space is generated by the CPU. It is also sometimes called Virtual Address Space.
Every machine has its own physical address space with its valid address range between 0 and some maximum limits supported by the machine.	In the virtual address Space, there is one address space per process , which may or may not start at 0 and extend to the highest address.

Computer Languages

- A Computer language includes various languages that are used to communicate with a Computer machine.
- Computer languages are used to write computer programs.
- Every computer language contains:
  Syntax: possible combination of symbols.
  Semantics: gives meaning to language.

•Basically there are two types of programming language: –

Low-level language

High-level language

Computer Language types

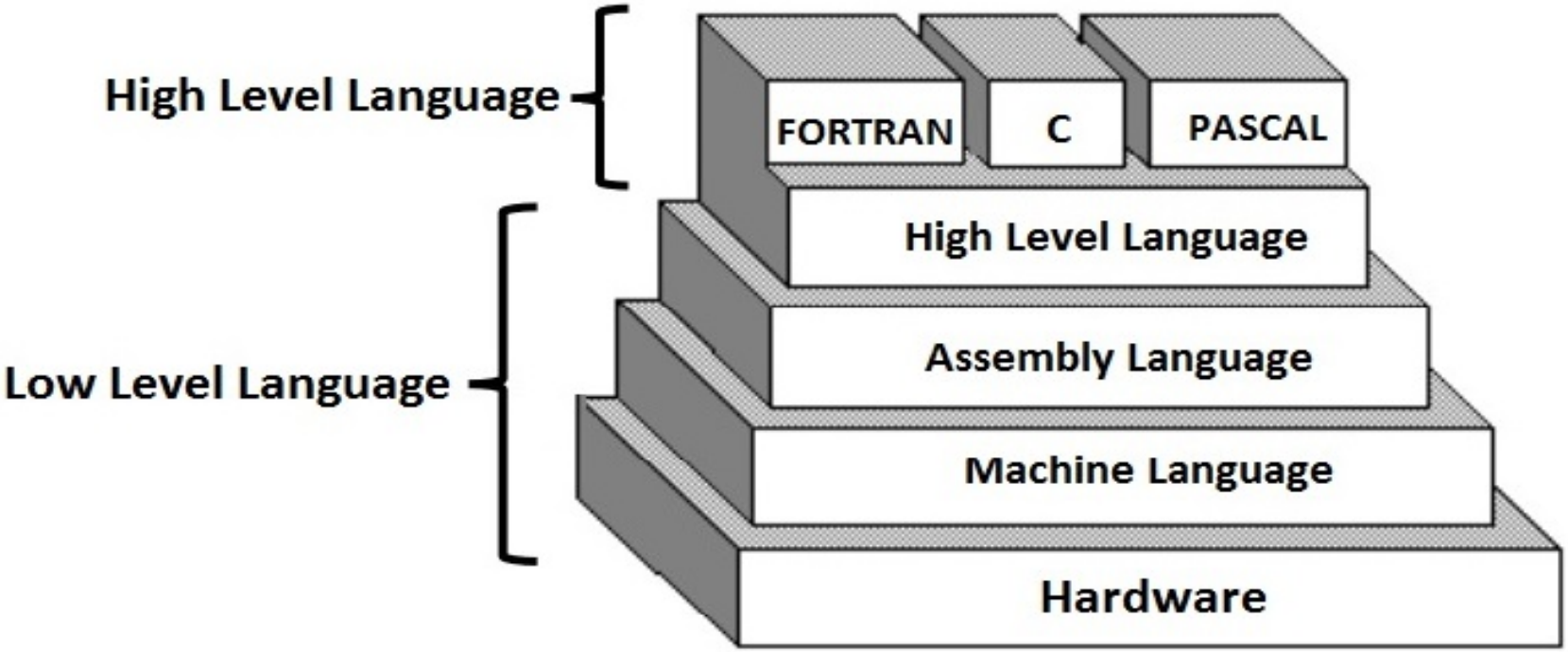


Figure 1.5 :Computer Languages[2]

Parul® University

Low Level Languages

Low Level Languages :

- Low level languages are the basic computer instructions or better known as machine codes.
- The main function of low level languages is to interact with the hardware of the computer.
- The languages that come under this category are the Machine level language and Assembly language.

Parul® University

Machine Level Languages

Machine Level Languages :

- Machine Language is truly a Computer Dependent Programming Language , which is written using binary codes.
- Machine code encompasses a function(opcode) and Operand(Address) Part.
- Programs written by using machine Language can be executed immediately without the requirement of any Language Translator.

Contd..

Disadvantage of Machine Level Languages :

- Programs based on machine Language are difficult to understand as well as develop.
- A pure machine-oriented language is difficult to remember and recall: All the instructions and data to the computer are fed in numerical form.
- Knowledge of computer internal architecture and codes is must for programming.
- Writing Code using machine language is time consuming ,cumbersome and complicated.
- Debugging of Programs is difficult.

Assembly Language

- It is a kind of Low-Level Programing Language which uses symbolic codes or mnemonics as instruction.
- The Processing of an assembly Language Program is done by using a Language Translator called Assembler that translates assembly Language code into Machine code.
- Assembly Language program needs to be translated into equivalent machine language code(binary code) before execution.

Parul® University

Advantages of Assembly Language

Advantages of Assembly Language :

- Due to use of Symbolic codes(mnemonics) , an assembly language programs can be written faster.
- It makes the programmer free from the burden of remembering the operation codes and addresses of memory location.
- It is easier to debug.

Disadvantages of Assembly Language :

- It takes a lot of time and effort to write the code for the same.
- It is very complex and difficult to understand.
- The syntax is difficult to remember.
- It has a lack of portability of program between different computer architectures.

High Level Language

- High Level Language were developed to overcome the large time consumption and cost in developing machine and assembly languages.
- It is much closure to English like Language.
- A Separate Language translator is required to translate high level language programs into machine readable object code.
- Some of the distinguished features of High level Language include interactive, support variety of data types, a rich set of operators, flexible control structures ,
  readability ,modularity, file handling and memory management.

Contd..

Advantages of High Level Language:

- It is machine independent.
- It can be used both as Problem and Procedure oriented.
- It follows simple English-like structure for program coding.
- It does not necessitate extensive knowledge of computer architecture.
- Writing code using high level language consumes less time.
- Debugging and program maintenance are easy.

Disadvantages of High Level Language:

- It requires language translators for converting instructions from high level language into low level object code.

High Level Language Vs. Low Level Language

Table 1.3 : High Level Vs Low Level Language

High Level Language	Low Level Language
High-Level Languages are easy to learn and understand.	Low-Level Languages are challenging to learn and understand.
They are executed slower than lower level languages because they require a translator program.	They execute with high speed.
They allow much more abstraction.	They allow little or no abstraction.
They do not provide many facilities at the hardware level.	They are very close to the hardware and help to write a program at the hardware level.
For writing programs, hardware knowledge is not required.	For writing programs, hardware knowledge is a must.

Contd…

High Level Language	Low Level Language
The programs are easy to modify.	Modifying programs is difficult.
A single statement may execute several instructions.	The statements can be directly mapped to processor instructions.
BASIC, Perl, Pascal, COBOL, Ruby etc are examples of High-Level Languages.	Machine language and Assembly language are Low-Level Languages.

Parul®
University

Life cycle of source Program

- The life cycle of a source program defines the program behavior and extends through execution stage, which exhibits the behavior specified in the program.
- Every source program goes through a life cycle of several stages.

- **Edit time**: It is the phase where editing of the program code takes place and is also known as design time. At this stage, the code is in its raw form and may not be in a consistent state.
- **Compile time**: At the compile time stage, the source code after editing is passed to a translator that translates it into machine code. One such translator is a compiler. This stage checks the program for inconsistencies and errors and produces an executable file.
- **Distribution time**: It is the stage that sends or distributes the program from the entity creating it to an entity invoking it. Mostly executable files are distributed.

Contd..

- **Installation time** : Typically, a program goes through the installation process, which makes it ready for execution within the system.
- **Link time** : At this stage, the specific implementation of the interface is linked and associated to the program invoking it. System libraries are linked by using the lookup of the name and the interface of the library needed during compile time or throughout the installation time, or invoked with the start or even during the execution process.
- **Load time** : This stage actively takes the executable image from its stored repositories and places them into active memory to initiate the execution. Load time activities are influenced by the underlying operating system.
- **Run time** : This is the final stage of the life cycle in which the programmed behavior of the source program is demonstrated

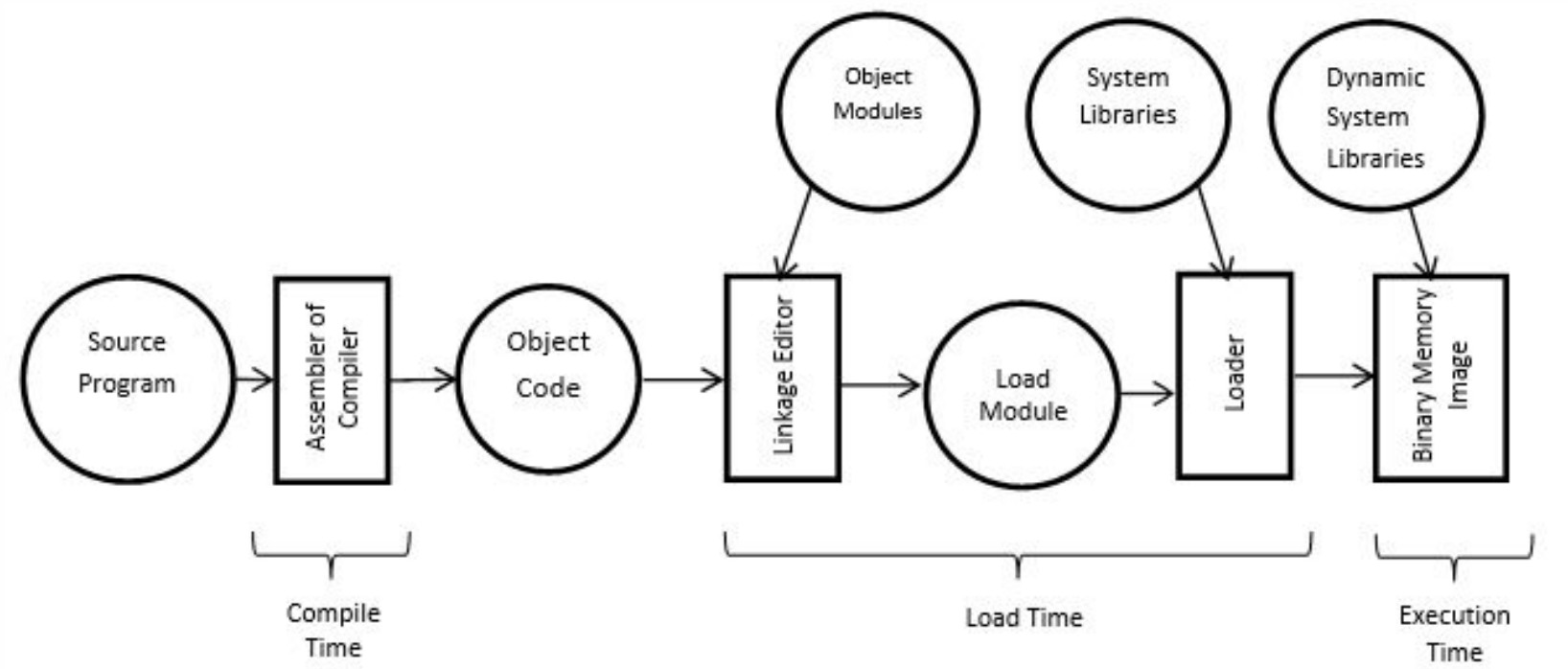


Figure 1.6 : Life Cycle of a Source Program[2]

System Software Development

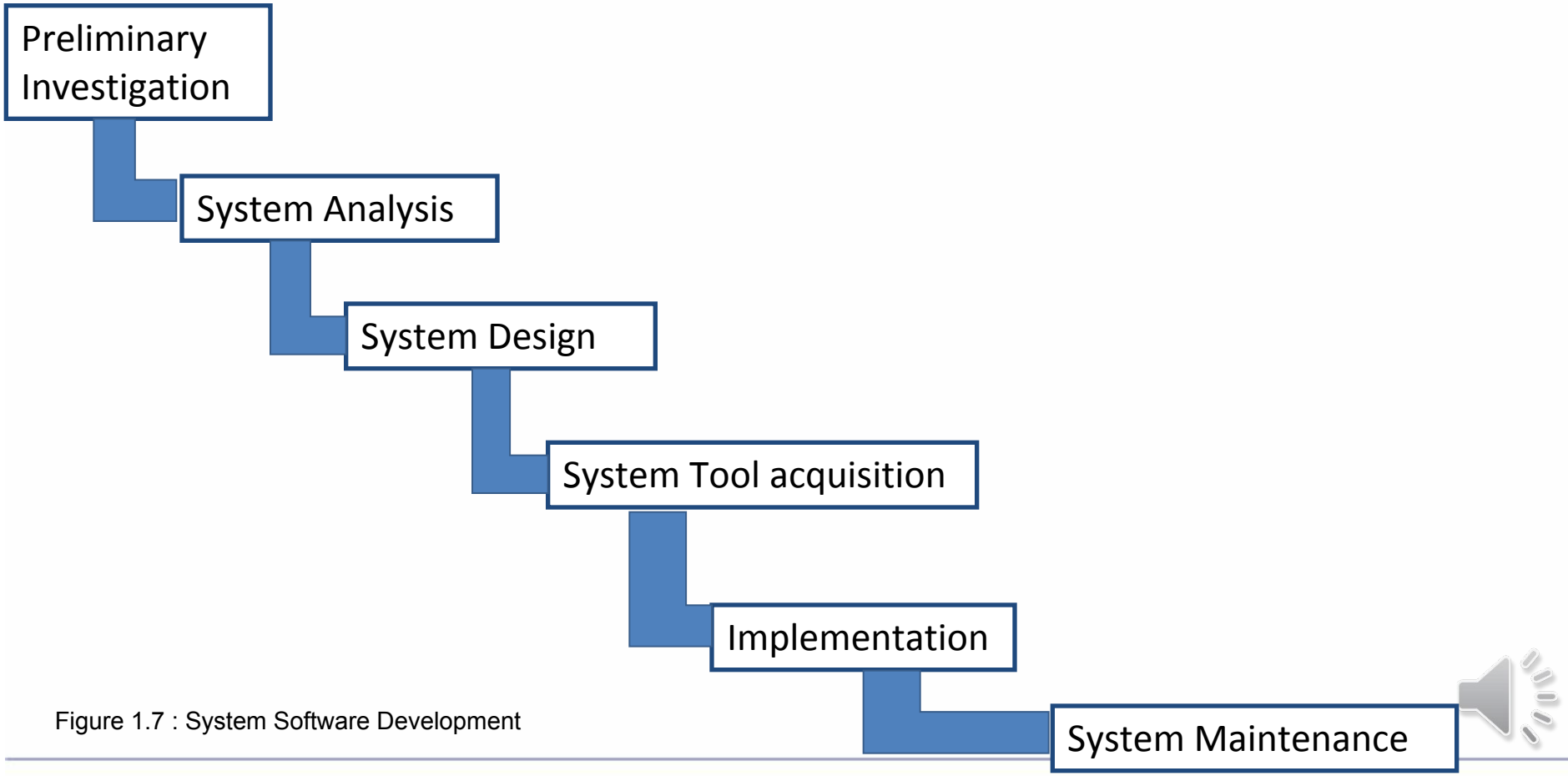


Figure 1.7 : System Software Development

Contd..

- Software development process follows the Software Development Life Cycle (SDLC), which has each step doing a specific activity till the final software is built.
- The system software development process also follows all the stages of SDLC, which are as follows:

- **Preliminary investigation**: It determines what problems need to be fixed by the system software being developed and what would be the better way of solving those problems.

- **System analysis:** It investigates the problem on a large scale and gathers all the information. It identifies the execution environment and interfaces required by the software to be built.

- **System design**: This is concerned with designing the blueprint of system software that specifies how the system software looks like and how it will perform.

Contd..

- **System tool acquisition:** It decides and works around the software tools to develop the functionalities of the system software.
- **Implementation:** It builds the software using the software tools with all the functionality, interfaces, and support for the execution. This may be very specific as the system software adheres to the architecture. Operating system support is sought for allocations and other related matters.
- **System maintenance:** Once the system software is ready, it is installed and used. The maintenance includes timely updating software what is already installed.

Recent Trends in Software Development

Latest trends in program development from the coding perspective include the following:

- Use of preprocessors against full software stack
- JavaScript MV* frameworks rather than Java Script files
- CSS frameworks against generic cascading style sheets
- SVG with JavaScript on Canvas in competition with Flash
- Gaming frameworks against native game development
- Single-page Web apps against websites
- Mobile Web apps against native apps
- Android against iOS
- Moving towards GPU from CPU
- Renting against buying (Cloud Services)
- Web interfaces but not IDEs
- Agile development

Levels of System Software

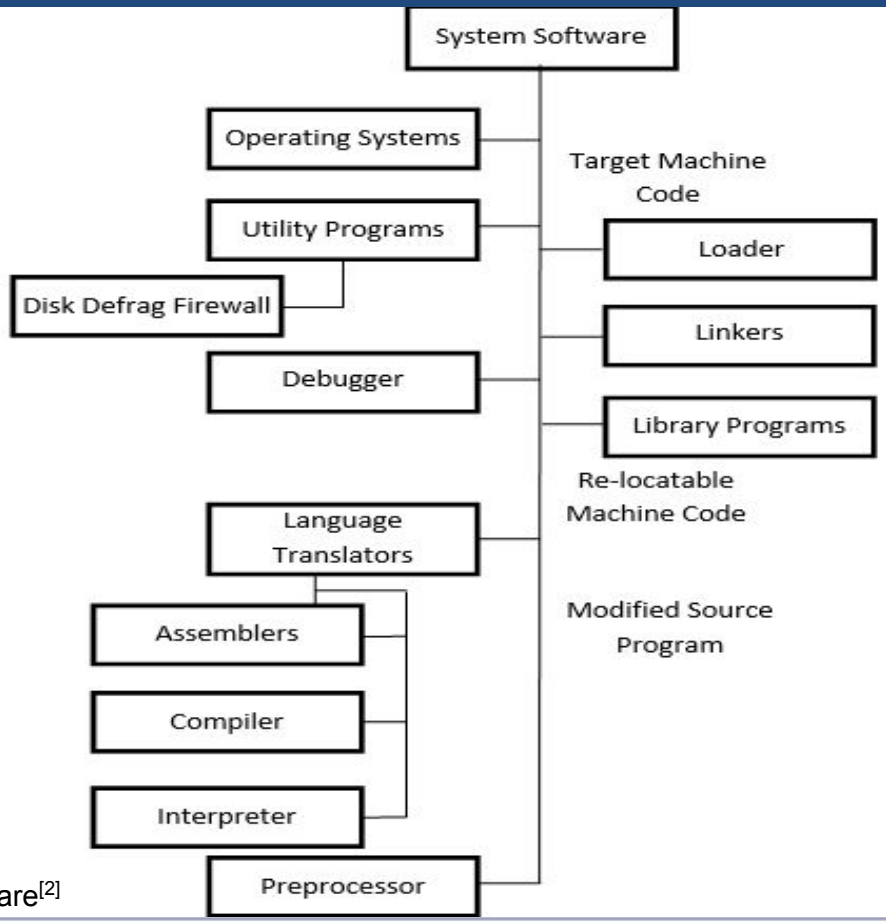


Figure 1.8: Levels of System Software[2]

Parul® University

References

[1] System Programming by D M Dhamdhere McGraw Hill Publication

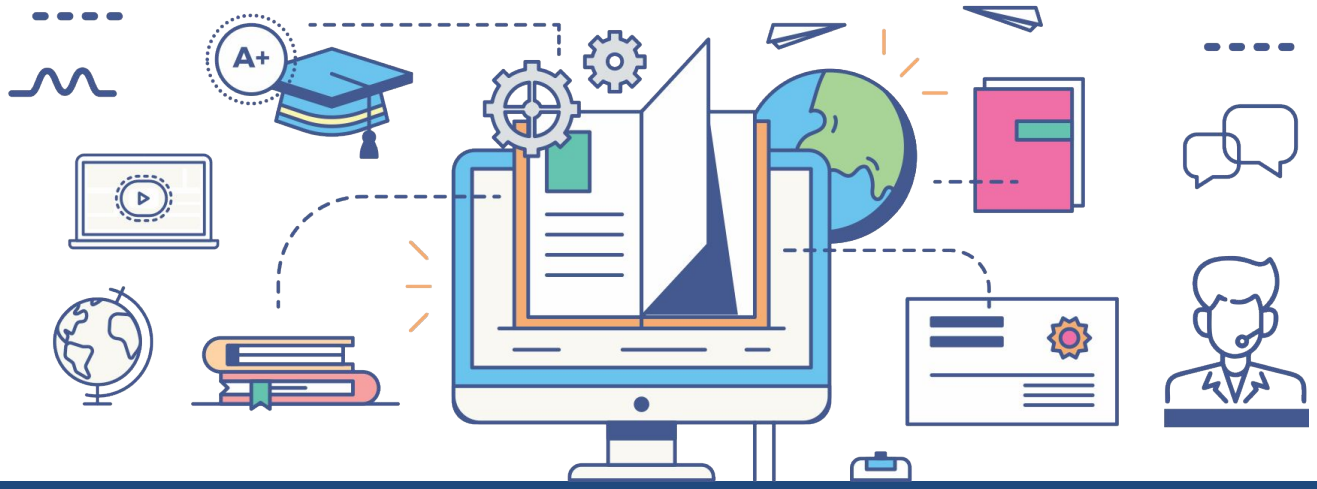
[2] System Programming by Srimanta Pal OXFORD Publication

[3] System Programming and Compiler Construction by R.K. Maurya& A. Godbole.

[4] System Software – An Introduction to Systems Programming by Leland L. Beck, 3rd Edition, Pearson Education Asia, 2000

[5] System Software by Santanu Chattopadhyay, Prentice-Hall India,2007

DIGITAL LEARNING CONTENT



Parul® University

www.paruluniversity.ac.in