

System Programming and Compiler Construction

-Ms.Nabanita Mandal

Syllabus

Module		Content	Hrs
1		Introduction to System Software	2
	1.1	Concept of System Software, Goals of system software, system program and system programming, Introduction to various system programs such as Assembler, Macro processor, Loader, Linker, Compiler, Interpreter, Device Drivers, Operating system, Editors, Debuggers.	
2		Assemblers	7
	2.1	Elements of Assembly Language programming, Assembly scheme, pass structure of assembler, Assembler Design: Two pass assembler Design and single pass Assembler Design for X86 processor, data structures used.	
3		Macros and Macro Processor	6
	3.1	Introduction, Macro definition and call, Features of Macro facility: Simple, parameterized, conditional and nested. Design of Two pass macro processor, data structures used.	
4		Loaders and Linkers	6
	4.1	Introduction, functions of loaders, Relocation and Linking concept, Different loading schemes: Relocating loader, Direct Linking Loader, Dynamic linking and loading.	

Syllabus

5		Compilers: Analysis Phase	10
	5.1	Introduction to compilers, Phases of compilers: Lexical Analysis - Role of Finite State Automata in Lexical Analysis, Design of Lexical analyzer, data structures used.	

		Syntax Analysis - Role of Context Free Grammar in Syntax analysis, Types of Parsers: Top down parser- LL(1), Bottom up parser- SR Parser, Operator precedence parser, SLR. Semantic Analysis , Syntax directed definitions.	
6		Compilers: Synthesis phase	8
	6.1	Intermediate Code Generation : Types of Intermediate codes: Syntax tree, Postfix notation, three address codes: Triples and Quadruples, indirect triple. Code Optimization : Need and sources of optimization, Code optimization techniques: Machine Dependent and Machine Independent. Code Generation : Issues in the design of code generator, code generation algorithm. Basic block and flow graph.	

Books and E-Resources

Textbooks:

- | | |
|---|--|
| 1 | D. M Dhamdhere: <i>Systems programming and Operating Systems</i> , Tata McGraw Hill, Revised Second Edition |
| 2 | A. V. Aho, R. Shethi, Monica Lam, J.D. Ulman: <i>Compilers Principles, Techniques and Tools</i> , Pearson Education, Second Edition. |
| 3 | J. J. Donovan: <i>Systems Programming</i> Tata McGraw Hill, Edition 1991 |

References:

- | | |
|---|--|
| 1 | John R. Levine, Tony Mason & Doug Brown, <i>Lex & YACC</i> , O 'Reilly publication, second Edition |
| 2 | D, M .Dhamdhere , <i>Compiler construction</i> 2e, Macmillan publication, second edition . |
| 3 | Kenneth C. Loudon , <i>Compiler construction: principles and practices</i> , Cengage Learning |
| 4 | Leland L. Beck, <i>System software: An introduction to system programming</i> , Pearson publication, Third Edition |

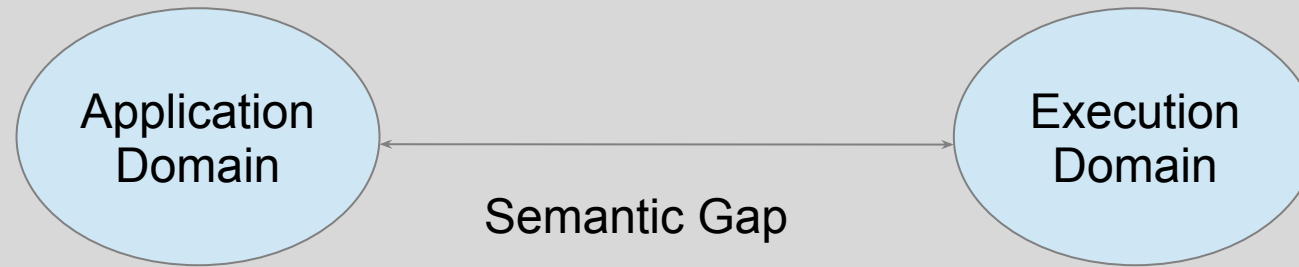
Useful Links for E-resources:

- | | |
|---|---|
| 1 | http://www.nptelvideos.in/2012/11/compiler-design.html |
| 2 | https://www.coursera.org/lecture/nand2tetris2/unit-4-1-syntax-analysis-5pC2Z |

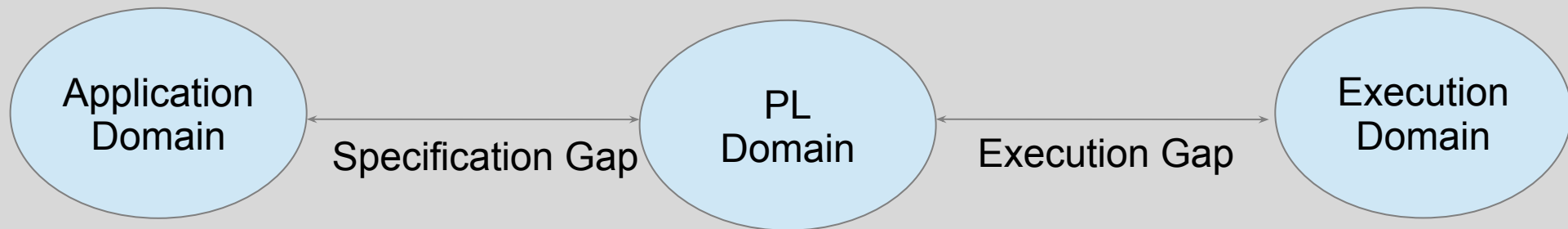
Module-1: Introduction to System Software

- ✓ Concept of System Software
- ✓ Goals of system software
- ✓ System program and system programming
- ✓ Introduction to various system programs such as: Assembler, Macro processor, Loader, Linker, Compiler, Interpreter, Device Drivers, Operating system, Editors, Debuggers

Introduction



Introducing Programming Language (PL) Domain



Programming Languages

Generation of Languages:

- First Generation Language (Machine Language)
- Second Generation Language (Assembly Language)
- Third Generation Language (High Level Language)

First Generation Language

- ◆ Also known as Machine Language
- ◆ Binary Language – Uses '0' and '1'
- ◆ Easy Understandable Instructions to Computers
- ◆ Used for Internal Structure of Program

- ◆ Advantages
 - ◆ Increase speed of processing
 - ◆ No Need of Translation
- ◆ Drawback
 - ◆ Difficult to Learn
 - ◆ Error prone

Second Generation Language

- ◆ Also known as Assembly Language
- ◆ Specific Meaningful Keywords: Mnemonics
- ◆ Disadvantage:
 - ◆ Not easily understood by machine
 - ◆ Program need to be translated to Machine Language. The translator is known as Assembler

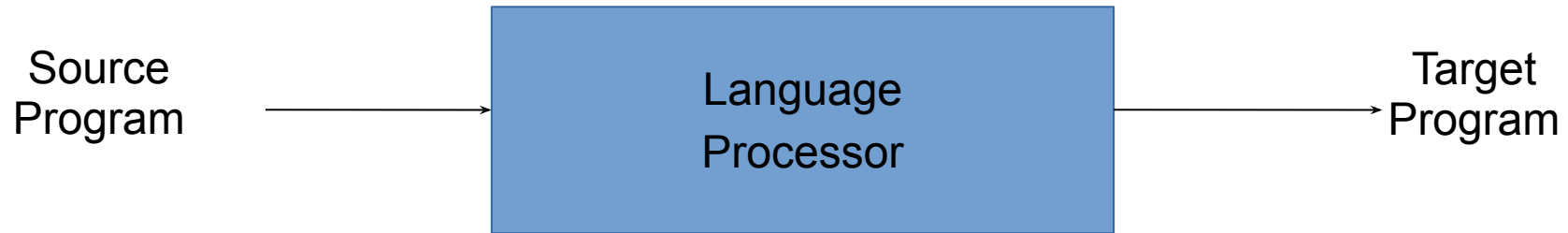
Third Generation Language

- ◆ Also known as High Level Language
- ◆ Advantage:
 - ◆ Syntax uses English keywords which are easy to understand
 - ◆ Languages are not machine oriented.
- ◆ Disadvantage
 - ◆ Program need to be translated to Machine Language. The translator known as Compiler
 - ◆ Slow processing in comparison with 1st and 2nd Generation Languages.

Example: C, C++, Java

Language Processor

- ◆ A language processor is software which bridges a specification or execution gap.



- ◆ A spectrum of language processors is designed to meet practical requirements.

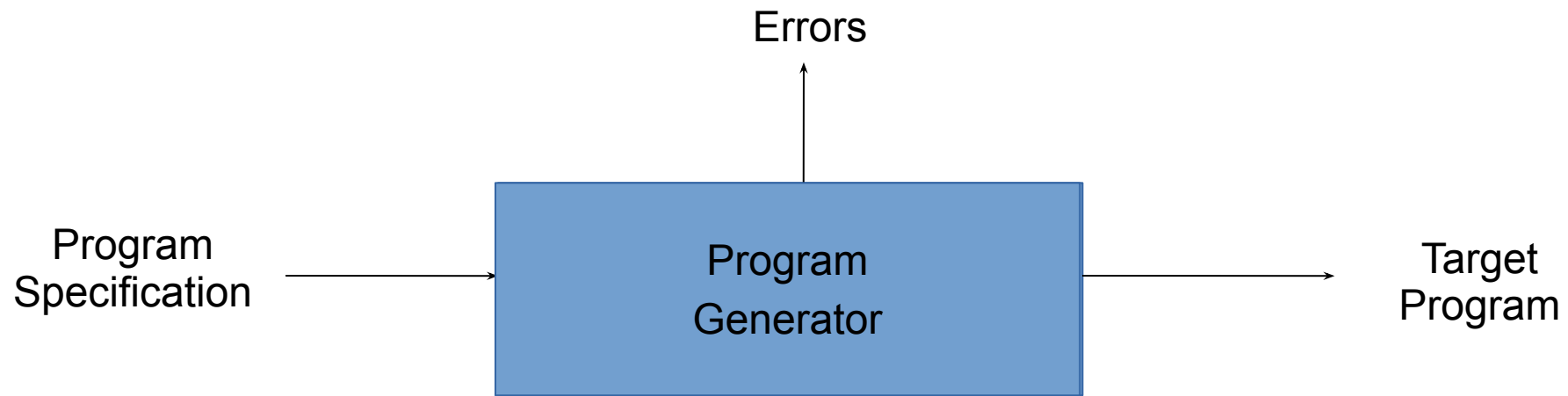
Language Processing Activities

◆**Program Generation Activities:** Bridges Specification (Design) Gap

◆**Program Execution Activities:** Bridges Execution Gap

Program Generation

- ❖ The program generator is system software which accepts the specification of a program to be generated and generates a program in the target language.



Program Execution

There are two modes of Program Execution

- ✓ **Program Translation**
- ✓ **Program Interpretation**

System Software

- It is a collection of program that facilitates execution of programs and use of resources in a computer system
- It contains a hierarchical arrangement of layers in which programs in upper layers use facilities provided by the layer below it

Goals of System Software

- **User Convenience:-** Provide convenient methods of using a computer system
- **Efficient Use:-** Ensure efficient use of a computer resources
- **Non-Interference:-** Prevent interference in the activities of its users

Goals of System Software

➤ User Convenience:-

Table 1.1 Facets of user convenience

Facet	Examples
Fulfillment of necessity	Ability to execute programs, use the file system
Good Service	Speedy response to computational requests
User friendly interfaces	Easy-to-use commands, Graphical user interface (GUI)
New programming model	Concurrent programming
Web-oriented features	Means to set up web enabled servers
Evolution	Add new features, use new computer technologies

Goals of System Software

➤ Efficient Use:-

- ✓ System software must efficiently use fundamental computer resources like CPU, Memory, Disks and other I/O Devices.
- ✓ Poor efficiency can occur if a program does not use the resource allocated to it which further results in snowballing effect.

Snowballing Effect:- If the resource is allocated to a user, it is denied to other programs that need it. These programs can't execute and hence the resources allocated to them also remains idle

- ✓ To achieve efficiency, the system software must minimize the waste of resources by programs and its own overhead.

Goals of System Software

➤ Non Interference:-

- The system software must ensure that no person can illegally use programs and resources in the system or interfere with them

System Programs and System Programming

- **System Software**:- It is a collection of programs
- **System Program**:- Each Program in the collection of system software

Design Goals

- The program should function correctly under all conditions
- The program should be effective in its computing environment
- The program should be portable
- The program should be able to evolve to provide new functionalities and adapt to new technologies

- **System Programming**:- It is the set of techniques used to realize the design goals of system program

System Software

Coordinates the functions and activities of hardware and software

Controls operations of computer hardware

Supports variety of programs that enables smooth execution of computer

Provides infrastructure over which programs can operate

User can focus on application rather than internal mechanism of Machine

Application Software

Any tool that functions and is operated by means of computer and allow end user to accomplish specific task

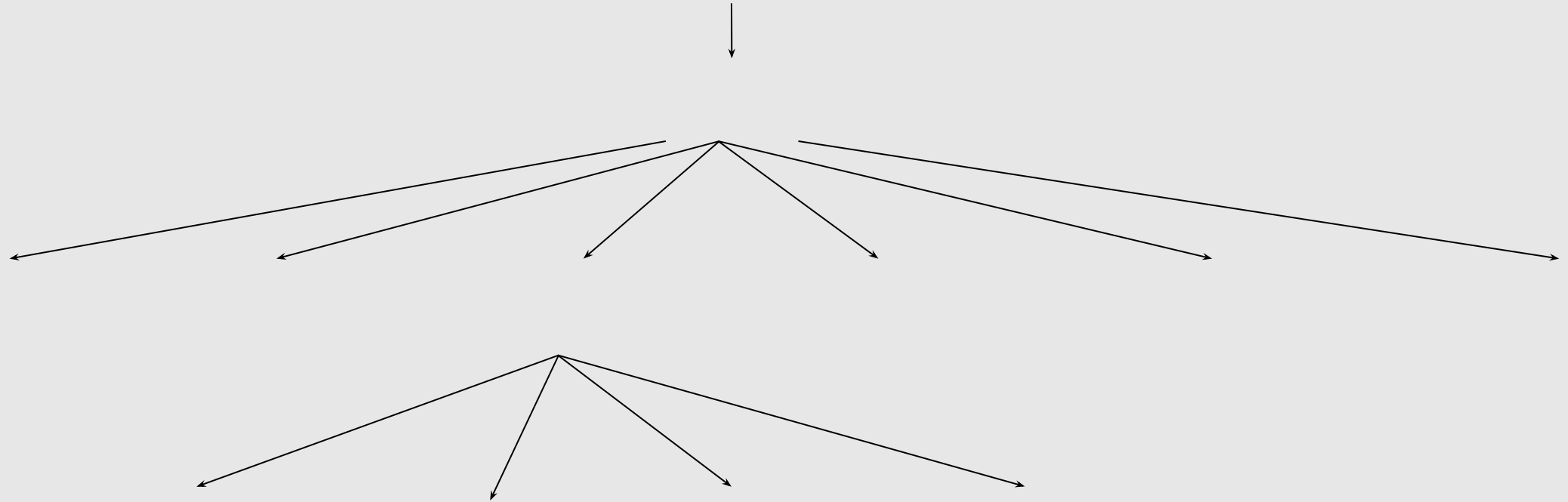
Most common programs run in foreground of computer

Tend to perform useful task which are not related with hardware communication

Dependent on system software to communicate to physical components

Examples: Industrial Automation, Office Suite, Media Players, Computer Games

System Software



System Software

➤ **Macro Processor**

➤ **Assembler**

➤ **Compiler**

➤ **Interpreter**

➤ **Loader**

➤ **Linker**

➤ **Device Drivers**

➤ **Operating System**

➤ **Editor**

➤ **Debugger**

Macroprocessor



Macro: Abbreviation for small code



Macro Definition: Sequence of code that has name



Macro Processor: Program that substitutes and specializes macro definitions and Macro calls

Assembler



It is a language processor



It converts an assembly language code to machine code

Compiler



It is a language processor



It translates a source program
written in some
high-level programming
language into machine code

Interpreter



It is a language processor



An interpreter translates high-level instructions into an intermediate form, which it then executes.

Loader

**Executable file
made by loader
has relative
memory
addresses
(Relocatable Code)**

**Loader will resolve
all relocatable
addresses by
considering base
Address**

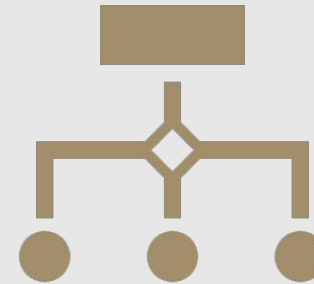
**It places program
in memory and
prepare for its
execution**

**Schemes:
Relocating,
Absolute and
direct linking**

Linker



It collects the code which are separately compiled or assembled and put into object file which is direct executable.

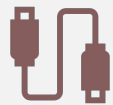


It connects an object program to the code for standard library and resource supplied by Operating system.

Device Drivers



➤ A computer program that controls a particular device connected to computer.



➤ Devices includes Input Output and Storage Devices

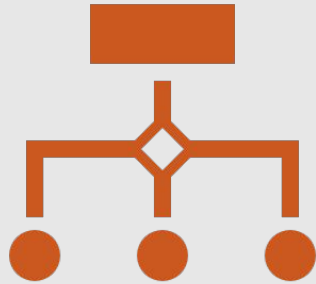
Operating System

➤ Mediator between User Programs and Hardware

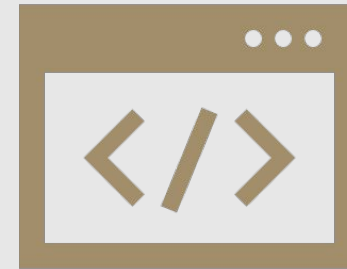
➤ Allocation of Resources and Services

➤ It includes: Process Management, Memory Management, File System Management, Secondary Storage Management, Program to manage these resources: Scheduler, Traffic Controller

Editor



Compilers usually accept source programs written using an editor that will produce a standard file such as ASCII file.



Compilers normally are bundled together with the editors and other programs into an Interactive Development Environment or IDE.

Debugger

- Debuggers are program that can be used to determine execution errors in a compiled program.
- It is also packaged with a compiler in an IDE.
- The debugger keeps track of most or all source code information, such as line numbers, names of variable and procedures.
- It can also halt execution at pre-specified locations called breakpoints as well as provide information on what functions have been called and what the current values of the variables are.

A decorative graphic on the left side of the slide, consisting of a network of light blue lines and small circles, resembling a circuit board or a neural network, extending vertically from the top to the bottom.

Introduction To System Software

MODULE 1

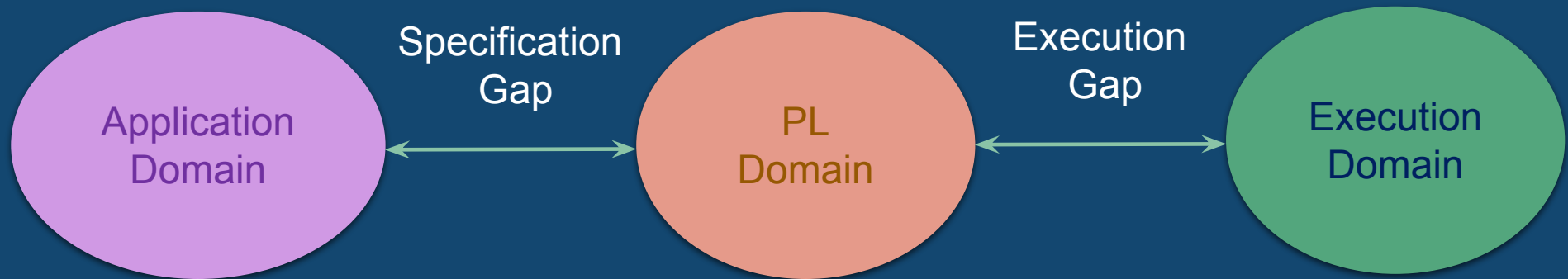
CONTENT

- Concept
- Programming Language Domain
- Generation of Languages
- System Program and Programming
- System Software and Application Software
- Examples of System Software

CONCEPT



CONCEPT





PROGRAMMING LANGUAGE DOMAIN

- Languages are used for communication
- Natural Mode of Communication: **Human Languages**
- To complete task from Machine: Use program or Request to machine
- Programs are written in: **Programming Languages**
- Generation of Languages:
 - First Generation Language (Machine Language)
 - Second Generation Language (Assembly Language)
 - Third Generation Language (High Level Language)

Prepared by: Mr. Vaibhav Ambhire

GENERATION OF LANGUAGES

1ST GENERATION

- Also known as Machine Language
- Binary Language – Uses '0' and '1'
- Easy Understandable instructions to computers
- Used for internal structure of program
- Advantage
 - Increase speed of Processing
 - No need of translation
- Disadvantage
 - Difficult to Learn
 - Error Prone

2ND GENERATION

- Also known as Assembly Language
- Specific Meaningful words Mnemonics
- Not easily understood by machine
- Program need to be translated into machine language
- The translator is known as 'Assembler'

3RD GENERATION

- Also known as High Level Language
- Syntax uses English keywords which are easy to understand
- Languages are not machine oriented
- Program need to be translated into machine language
- Slow processing in comparison with 1st and 2nd Generation Languages
- The translator is known as 'Compiler'
- Example: C, C++, Java

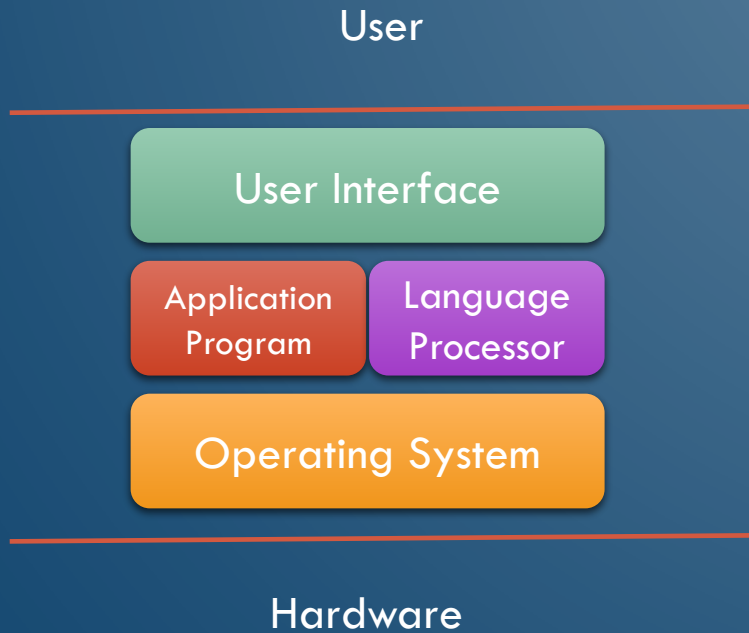
Prepared by: Mr. Vaibhav Ambhire

LANGUAGE PROCESSOR

- A language processor is software which bridges a specification or execution gap.
- A spectrum of language processors is designed to meet practical requirements
- Language Processing Activities
 - Program Generation: Bridges Specification Gap
 - Program Execution: Bridges Execution Gap

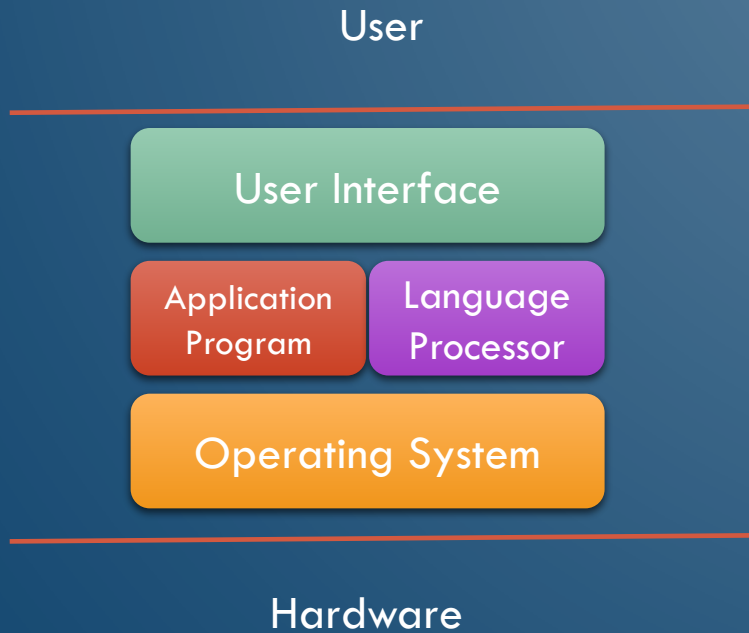


Representation Views of Computer System



- User Interface:
 - Accepts users commands for using services provided by the operating system
 - initiates execution of one or more programs for fulfilment of command
- Application Program or Language Processors:
 - Implements the user's application (Student's View)
 - assists in development of a program (Programmer's View)
- Operating System:
 - controls operation of computer
 - provides a set of services for executing programs and using resources of the computer

System Software



- It is a collection of program that facilitates execution of programs and use of resources in a computer system
- It contains an hierarchical arrangement of layers in which programs in upper layers use facilities provided by the layer below it
- Goals of System Software
 1. User Convenience
 2. Efficient Use
 3. Non Interference



GOALS OF SYSTEM SOFTWARE

1. User Convenience

Facet	Example
Fulfilment of Necessity	Ability to execute the program Use the File System
Good Service	Speedy response to Computational Request
User friendly Interfaces	Easy-to-use Command Graphical User Interfaces (GUI)
New Programming Model	Concurrent Programming
Web – oriented Features	Means to set up web enabled servers
Evolution	Add New Features Use new Computer Technologies



GOALS OF SYSTEM SOFTWARE

2. Efficient Use


- System software must efficiently use fundamental computer resources like CPU, Memory, Disks and other I/O Devices
- Poor efficiency can occur if a program does not use the resource allocated to it which further results in snowballing effect.
- *Snowballing Effect*: If the resource is allocated to a user, it is denied to other programs that need it. These programs can't execute and hence the resources allocated to them also remains idle
- To achieve efficiency, the system software must minimize the waste of resources by programs and its own overhead.



GOALS OF SYSTEM SOFTWARE

3. Non-Interference

The system software must ensure that no person can illegally use programs and resources in the system or interfere with them



SYSTEM PROGRAM AND SYSTEM PROGRAMMING

- **System Software:** It is a collection of programs
- **System Program:** Each Program in the collection of system software
- **Design Goals of System Programs:**
 - ✓ The program should function correctly under all conditions
 - ✓ The program should be effective in its computing environment
 - ✓ The program should be portable
 - ✓ The program should be able to evolve to provide new functionalities and adapt to new technologies
- **System Programming:** It is the set of techniques used to realize the design goals of system

TYPES OF SOFTWARE

SYSTEM SOFTWARE

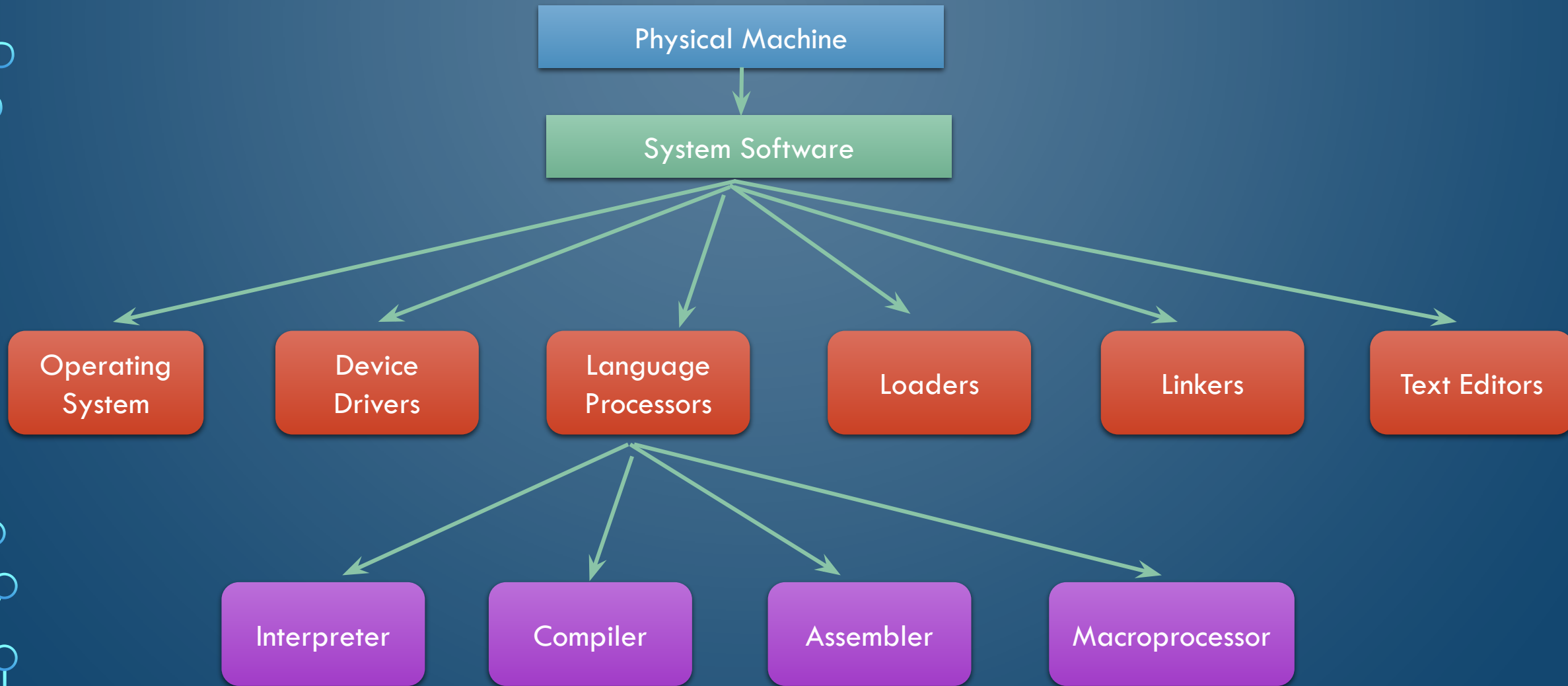
- Coordinates the functions and activities of hardware and software
- Controls operations of computer hardware
- Supports variety of programs that enables smooth execution of computer
- It provides infrastructure over which programs can operate
- User can focus on application rather than internal mechanism of Machine

APPLICATION SOFTWARE

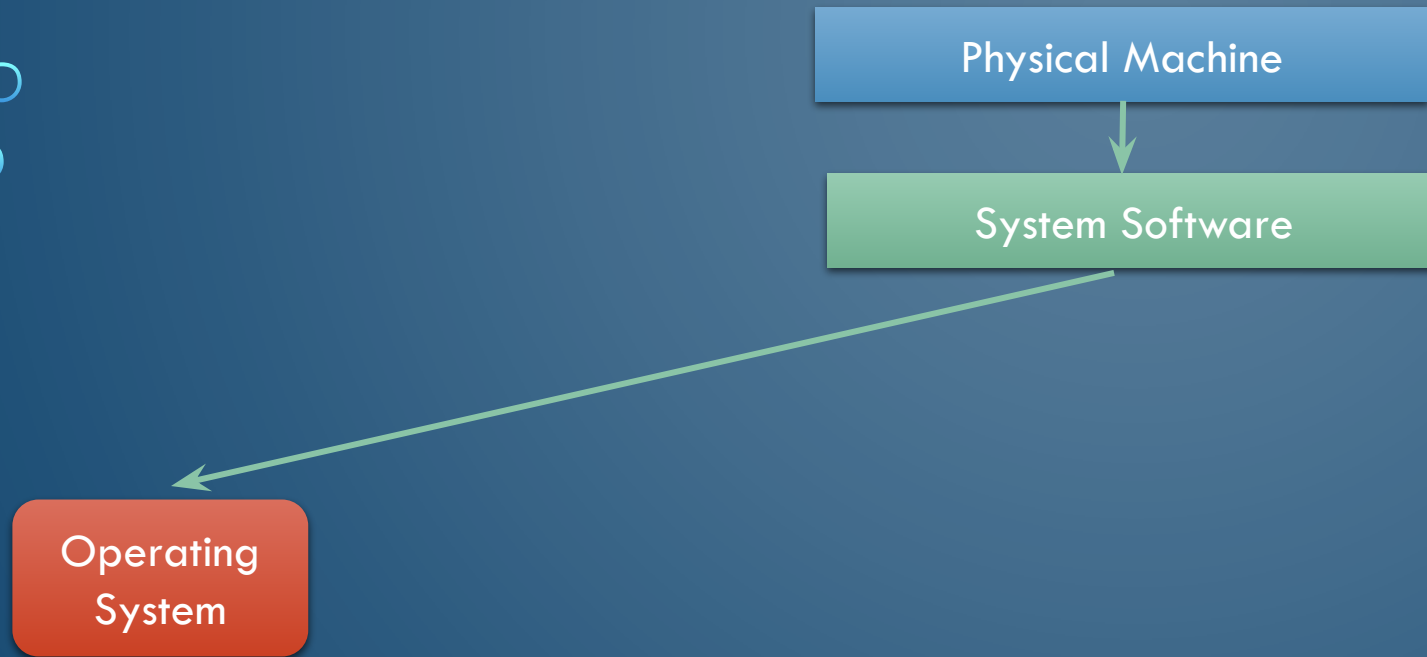
- Any tool that functions and is operated by means of computer and allow end user to accomplish specific task
- Most common programs run in foreground of computer
- Tend to perform useful task which are not related with hardware communication
- It provides infrastructure over which programs can operate
- Dependent on system software to communicate to physical components
- Examples: Industrial Automation, Office Suite, Media Players, Computer Games

Prepared by: Mr. Vaibhav Ambhire

SYSTEM SOFTWARE



SYSTEM SOFTWARE

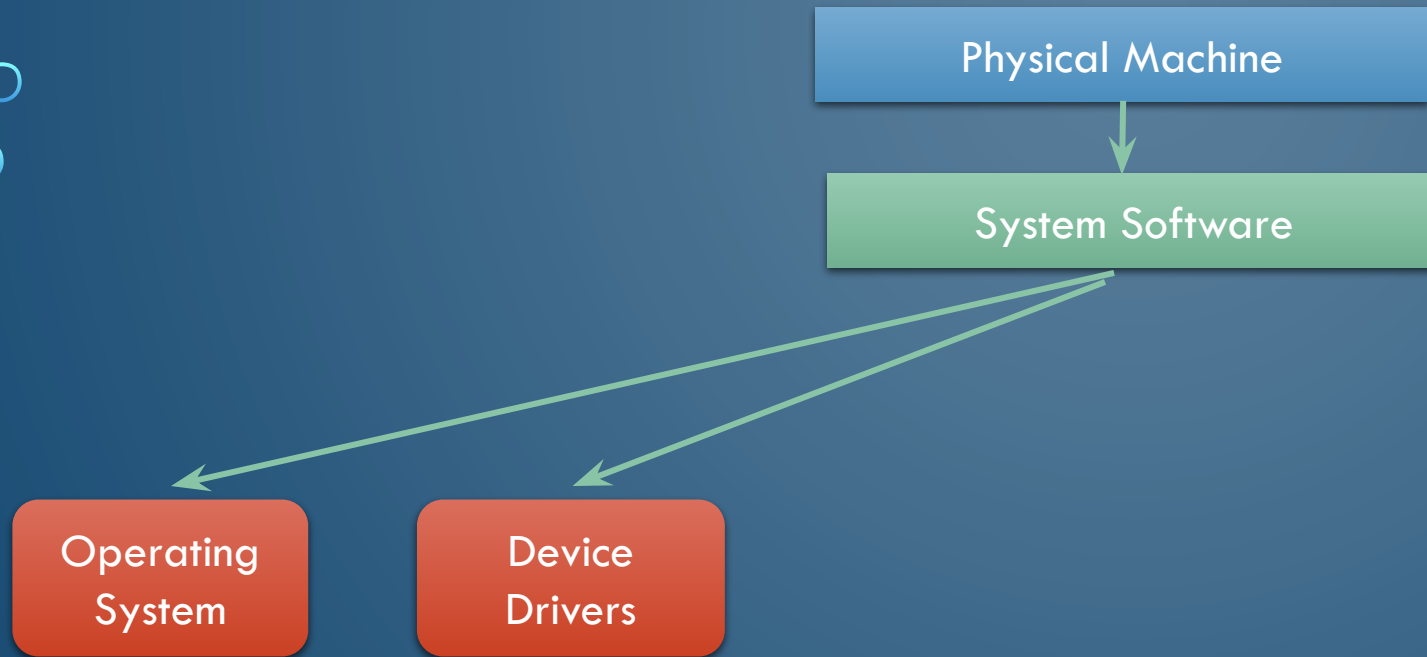




OPERATING SYSTEM

- Mediator between User Programs and Hardware
- Allocation of Resources and Services
- It includes
 - Process Management
 - Memory Management
 - File System Management
 - Secondary Storage Management
- Program to manage these resources:
Scheduler, Traffic Controller

SYSTEM SOFTWARE





UTILITIES AND DEVICE DRIVERS

Utilities

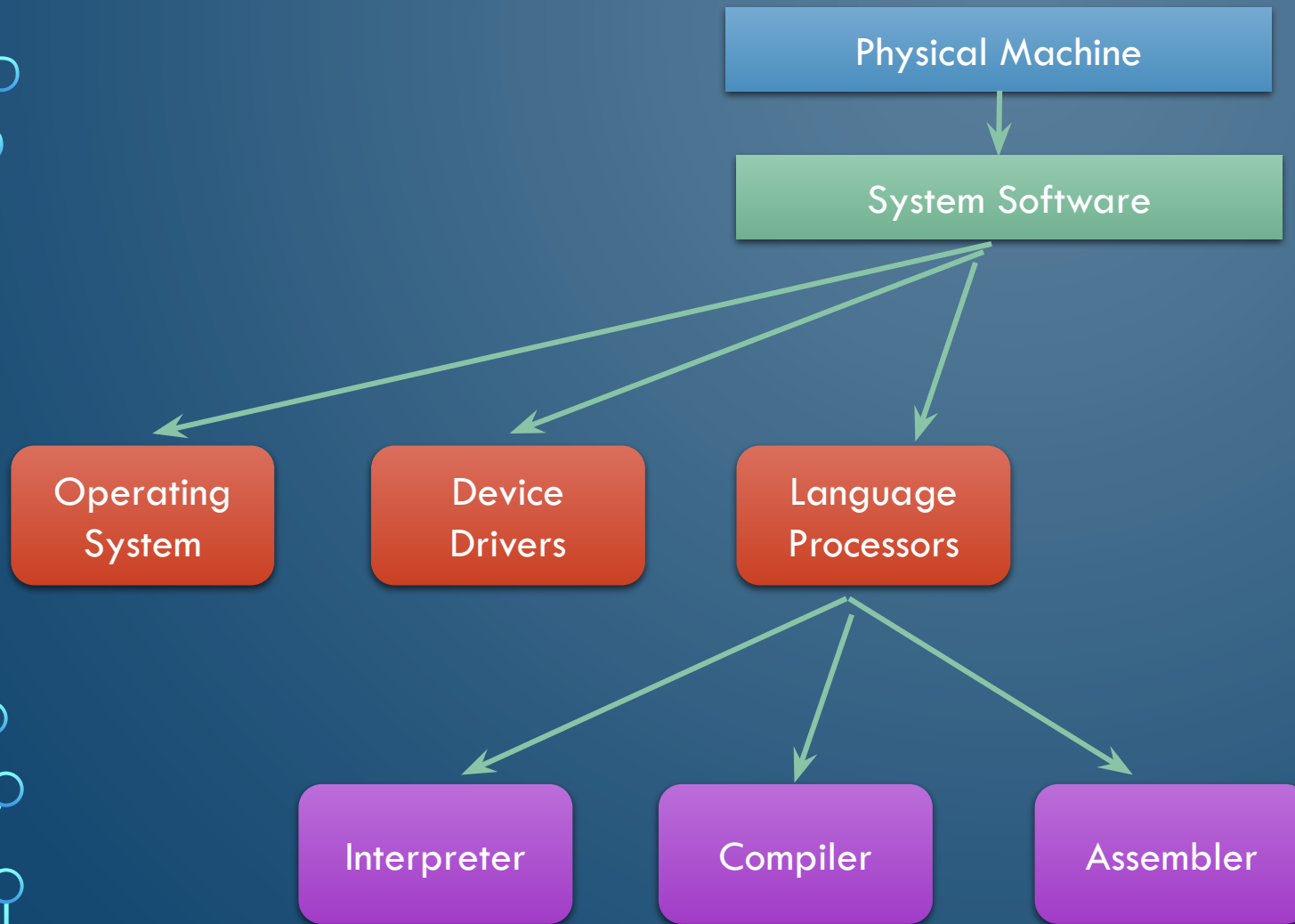
- small program that provides additional capabilities to operating system
- Special but non-essential part of operating system
- Performs functions related to computer system management and maintenance
- Example: Antivirus SW, Data Compression SW, Disk Optimization SW

Device Drivers:

- A computer program that controls a particular device connected to computer.
- Devices includes Input Output and Storage Devices

Prepared by: Mr. Vaibhav Ambhire

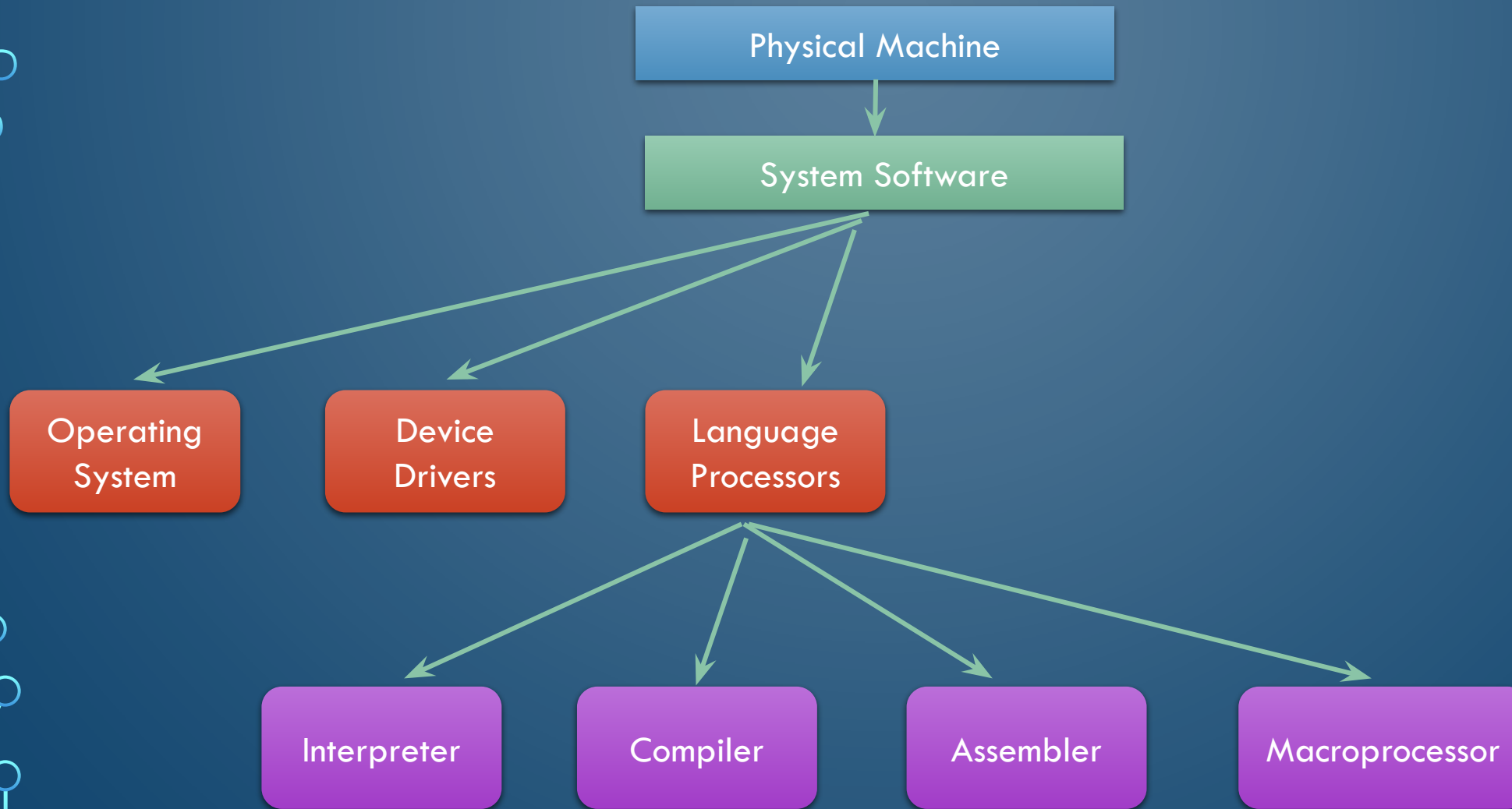
SYSTEM SOFTWARE



LANGUAGE PROCESSORS

Parameter	Assembler	Compiler	Interpreter
Conversion	Assembly Language into Machine Language	Source Program to Target Program	Line by Line Conversion from HLL to Machine Language and executes line by line
Speed of Execution	Fast	Fast	Slow
Translation of	Entire Program	Entire Program	Line by Line
Mechanism for Execution	Program need to be assembled	Once compiled can be executed multiple times	For every execution, each line must be interpreted
Creation of Object File	Yes	Yes	No
Example	MASM	C Compiler, Javac	Basic Interpreter, Python Interpreter

SYSTEM SOFTWARE

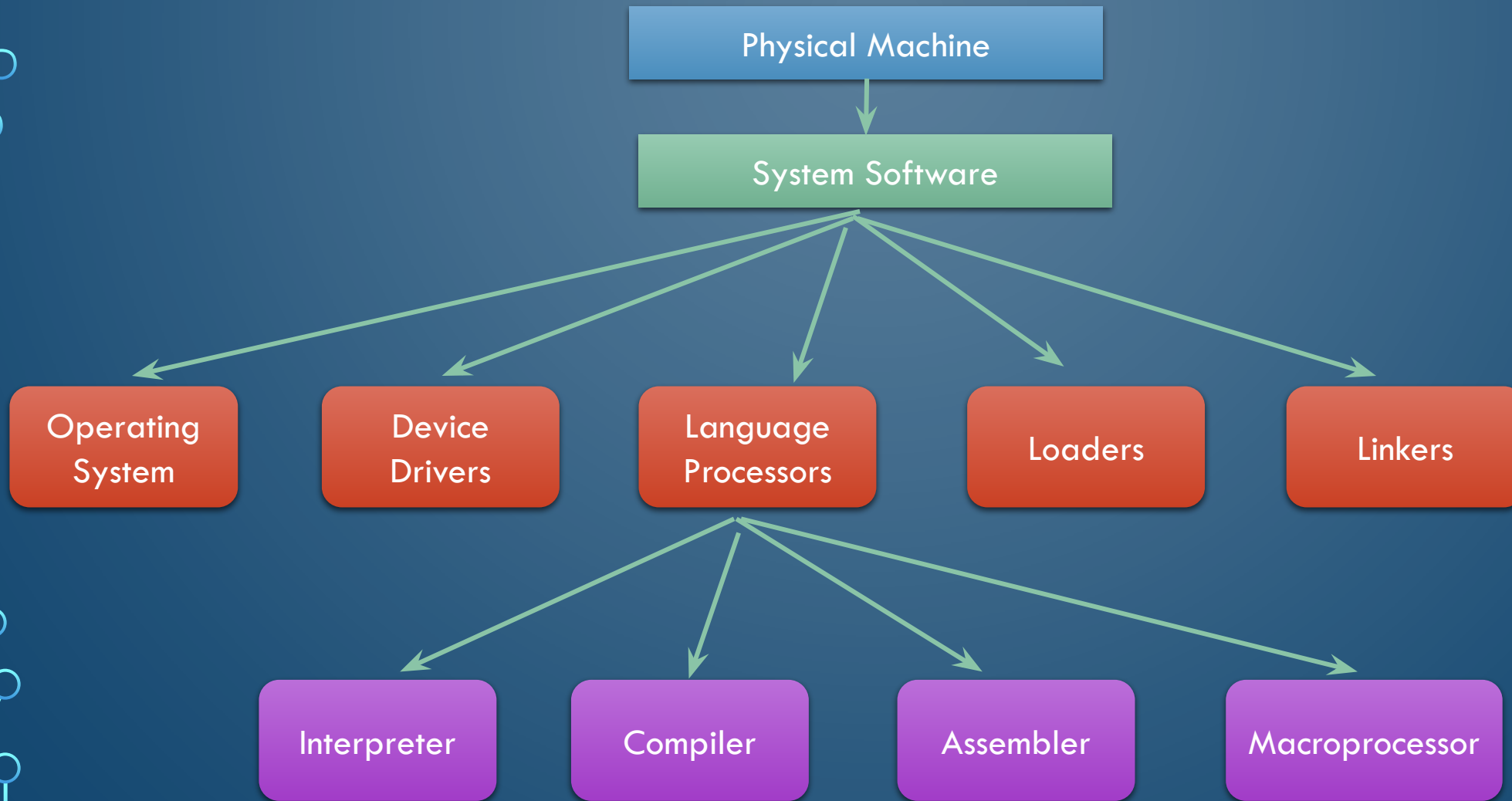




MACRO PROCESSOR

- **Macro:** Abbreviation for small code
- **Macro Definition:** Sequence of code that has name
- **Macro Processor:** Program that substitutes and specializes macro definitions and Macro calls

SYSTEM SOFTWARE





LINKERS AND LOADERS

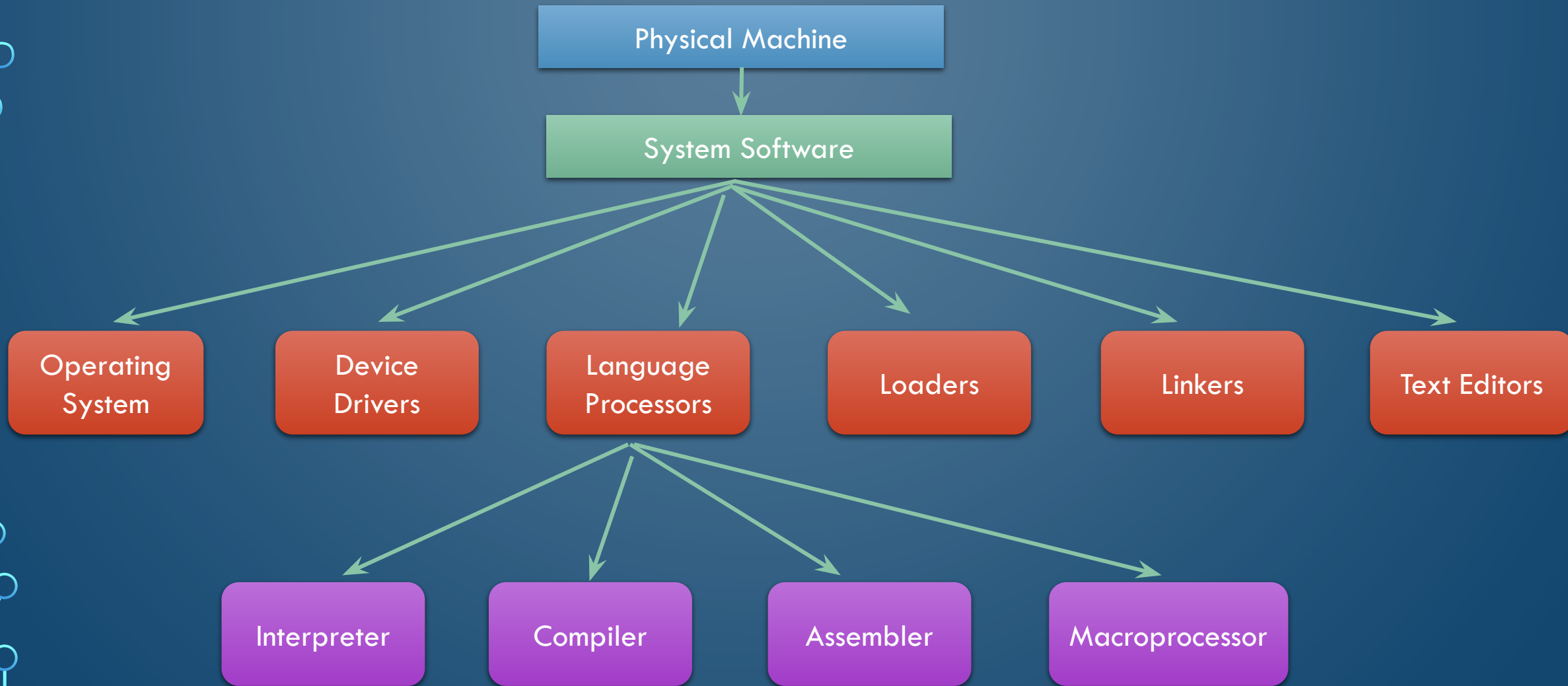
Linkers

- collects the code which are separately compiled or assembled and put into object file which is direct executable.
- Connects an object program to the code for standard library and resource supplied by Operating system

Loaders

- Executable file made by loader has relative memory addresses (Relocatable Code)
- Loader will resolve all relocatable addresses by considering base Address
- It places program in memory and prepare for its execution
- Schemes: Relocating, Absolute and direct linking

SYSTEM SOFTWARE





TEXT EDITORS AND DEBUGGER

Text Editor

- Compilers usually accept source programs written using an editor that will produce a standard file such as ASCII file.
- Compilers normally are bundled together with the editors and other programs into an Interactive Development Environment or IDE

Debugger

- Debuggers are program that can be used to determine execution errors in a compiled program.
- It is also packaged with a compiler in an IDE.
- Running a program with debugger differs from straight execution.
- The debugger keeps track of most or all source code information, such as line numbers names of variable and procedures.
- It can also halt execution at pre-specified locations called breakpoints
- At breakpoint it provide information on what functions have been called and what the current values of the variables are.

SYSTEM SOFTWARE

