System Programming: Introduction

System Programming

- **Unit 1: Introduction to System Programming**

  Introduction, Assemblers , Macro Processors, RISC machines,

- **Unit 2: Macro Processor and Assembler**
- **Unit 3: Compilers, Loaders and Linkers**
- **Unit 4: Essential concepts of Systems programming for Linux as Open Source OS.**
- **Unit 5: Encoding, Decoding and Device drivers**
- **Unit 6: TSR Programming**

Hardware vs. Software

- **Hardware**

All physical contents of computer are hardware. This form is given to all electrical and mechanical devices attached to the computer for the purpose of input, process, and storage and output operations

- **Software**

Software is a general term used for computer Programs. A computer program is a planned, step by step set of instructions that directs the computer what to do and how to do.

Introduction to System Programming

Software

- *Software* is the name given to the computer programs that instruct the hardware how to work.
- *Software* are the instructions in the form of programs which control the operation of a computer, together with the associated documentation.
- Without software, the computer would

Types Of Software

System Software

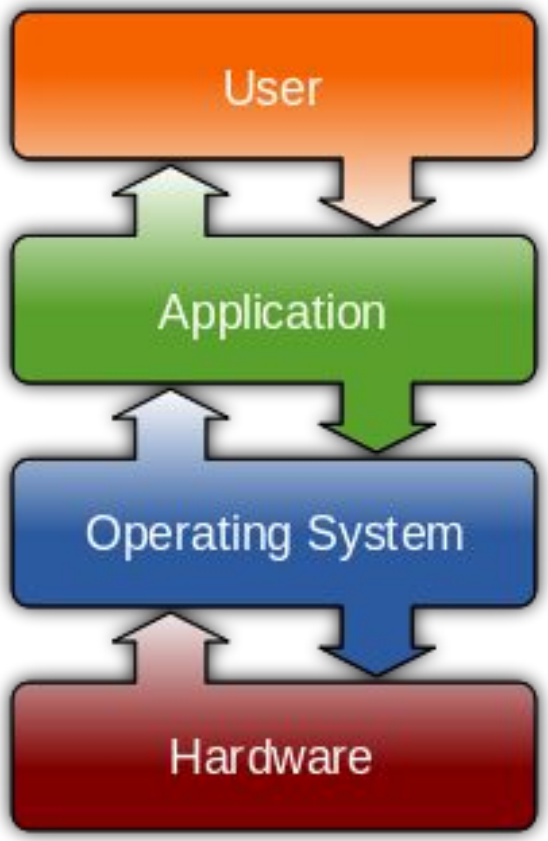
The programs directly related to the computer hardware and perform tasks associated with controlling and utilizing computer hardware

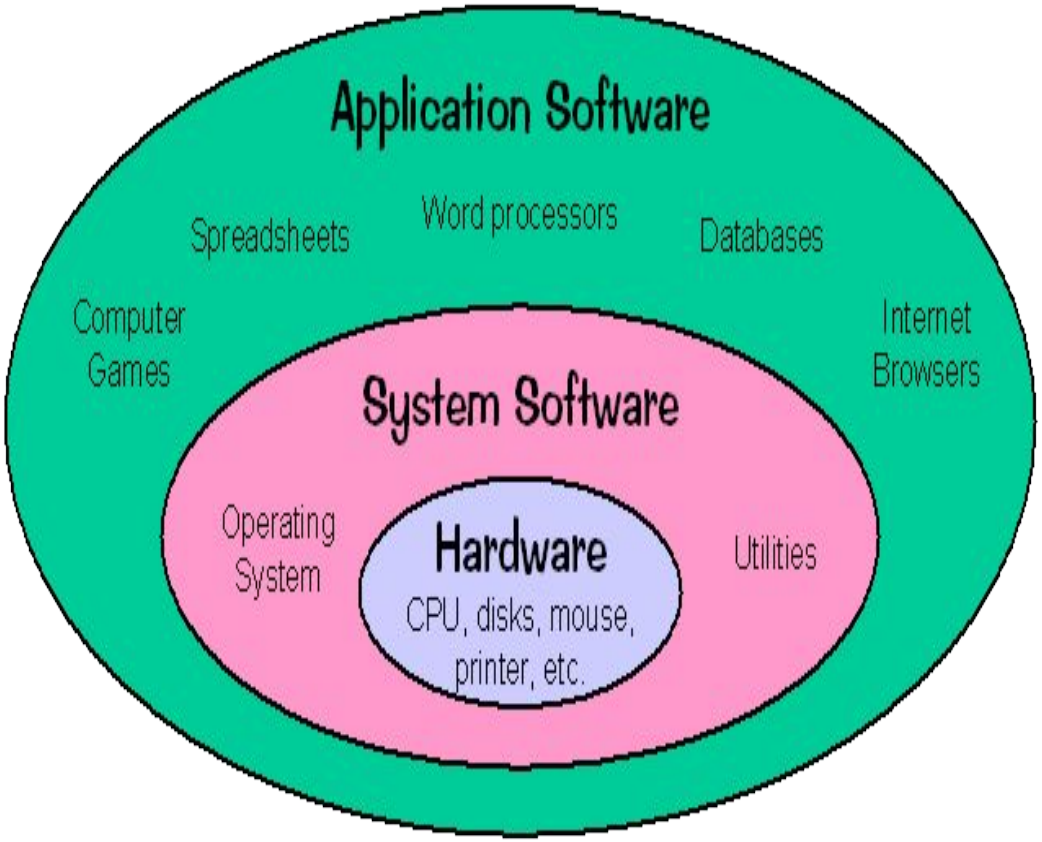
Application Software

An application is a job or task a user wants to accomplish through a computer.

Application software are programs that help a user perform a specific job.

How it is?


User
Application
Operating System
Hardware


Application Software
Spreadsheets
Word processors
Databases
Computer Games
Internet Browsers
System Software
Operating System
Hardware
CPU, disks, mouse, printer, etc.
Utilities

System Software

- System software is computer software designed to operate the computer hardware, to provide basic functionality, and to provide a platform for running application software.
- System software includes device drivers, operating systems, servers, utilities, and window systems.
- System software is responsible for managing a variety of independent hardware components, so that they can work together harmoniously.
- System programs manages internal operation of Computer also manages peripherals like storage Devices, Monitor, Printer.

System Software

Machine dependency of system software

–System programs are intended to support the operation and use of the computer.

–Machine architecture differs in:

- Machine code
- Instruction formats
- Addressing mode
- Registers

Machine independency of system software

–General design and logic is basically the same:

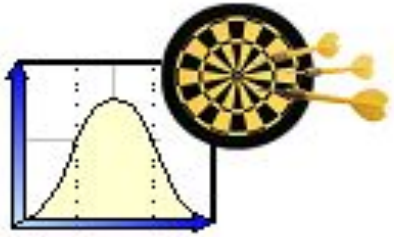
- Code optimization
- Subprogram linking

System Software

System software helps run the computer hardware and computer system. It includes combination of the following:

- **Device drivers**
- **Operating systems**
- **Servers**
- **Utilities**
- **Compiler**
- **Assembler**

SYSTEM PROGRAMMING



System performance and workload management

Security, Availability and Integrity

System parameters and system libraries management

Controlling operating activities and functions

z/OS new features implementation and z/OS system maintenance

Hardware I/O configuration

Need Of System Software

- **System control programs**
  - controls the execution of programs, manage the storage & processing resources of the computer & perform other management & monitoring function.
  - The most important of these programs is the operating system.
- **System support programs**
  - provide routine service functions to the other computer programs & computer users: E.g. Utilities, libraries,
  - Ex:Text editors, language translators such as BASIC interpreter
- **System Development Programs**
  - They assists in the creation of computer programs. Examples of system development are –
  - Language translations.

Application software

- Application software are the software that are designed to satisfy a particular need of a particular environment.
- Guides the computer to carry out instructions provided by the user.
- Application Software can't work without System Software.
- Examples of application software are-student record software, railway reservation software, income tax software, word processors etc.

Application software

1) Opera (Web Browser)
2) Microsoft Word (Word Processing)
3) Microsoft Excel (Spreadsheet software)
5) MySQL (Database Software)
6) Microsoft Powerpoint (Presentation Software)
7) iTunes (Music / Sound Software)
8) VLC Media Player (Audio / Video Software)
9) World of Warcraft (Game Software)
10) Adobe Photoshop (Graphics Software)

System Software vs. Application Software

System Software	Application Software
Provides an platform for an user to interact with hardware of computer	Runs on System software for serving specific purpose.
Run in background and act as platform	Run in foreground & interact with user
e.g Language Processor, Operating System, Disk Driver	e.g. Video games, Text Editor & Browser
Is often done in low level language & C where programs have to manage memory themselves.	Is ofetn done in languages Java C#, Pearl, Python, Ruby, Javascript that feature automatic garbage collection & free the programmer from low level worries.

Evolution of Programming

- **Machine Language**
  - Binary format

    ```
    11100101100111111000100000010000
    11100101100111111000000000001000
    11100000100000010101000000000000
    11100101100011110101000000001000
    ```

  - **Hexadecimal format**

    ```
    E59F1010
    E59F0008
    E0815000
    E58F5008
    ```

Evolution of Programming

- **Assembly Language**
  - Mnemonic codes

```
E59F1010   LDR   R1, num1
E59F0008   LDR   R0, num2
E0815000   ADD   R5, R1, R0
E58F5008   STR   R5, sum
```

- **High-Level Language**
  - C language

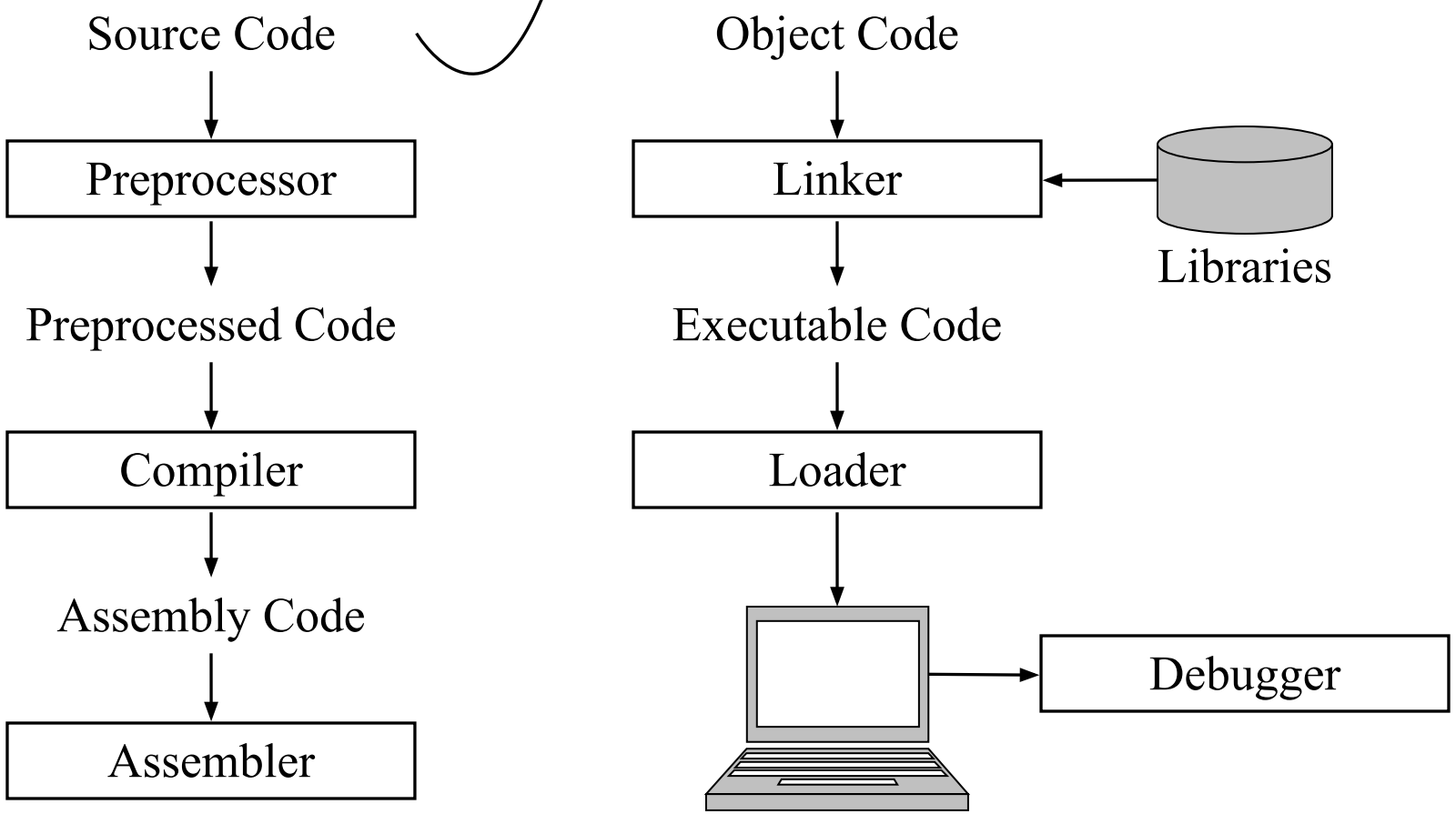
```
sum = num1 + num2;
```

Components of System Software:

The system software includes

- Assembler
- Linker
- Loader
- Macro processor
- Text editor
- Compiler
- Operating system
- Debugging system

From Source to Executable



Components of System Software:

Text editor (hello.c)

– To create and modify the program

– Editors rarely have the advanced formatting and other features of a regular word processor, but sometimes include special tools and features that are useful for programming.

Components of System Software:

Pre-processor (hello.c -> hello.i)

–Includes the Headers

–Expand the macros

–Augmentation

#define identifier replacement

e.g. #define TABLE_SIZE 100

int table1[TABLE_SIZE];

int table2[TABLE_SIZE];

- **Conditional inclusions (#ifdef, #ifndef, #if, #endif, #else and #elif)**

Compiler

Compilation(hello.i -> hello.s)

It is a program which translates a high level language program into a machine language program (**optionally with relocation)**

- A software program that converts source code that written in high level programming language into low level language.
- A ***Native-compiler*** runs on a computer platform and produces code for that same computer platform.
- A ***Cross-compiler*** runs on one computer

Assembler

- **Converts mnemonic codes into object file. (hello.s -> hello.o)**
- Assemblers are further divided into two types: One Pass Assembler and Two Pass Assembler.
- **Converts symbolic (e.g., jump labels, variable names)**
- **operands to their machine addresses**
- **Uses proper addressing modes and formats to build efficient machine instructions**
- **Outputs the object program and provide other information (e.g., for linker and loader)**

Linker

- A linker or link editor is a program that takes one or more objects generated by compilers and assembles them into a single executable program or a library .(**concatenation, hello.o -> hello.exe)**
- Link the object files and libraries to form an executable

LOADER

- Loader is a program that loads machine codes of a program into the system memory.
- it places programs into memory and prepares them for execution.
- Loading a program involves reading the contents of executable file into memory.

Debugger

- A **debugger** or **debugging tool** is a computer program that is used to test and debug other programs.
- The code to be examined might alternatively be running on an *instruction set simulator* .
- When the program crashes, the debugger shows the actual position(Segment) in the original code if it is a source-level debugger.
- If it is a low-level debugger or a machine-language debugger it shows that line

Emulator

- An emulator is a piece of Hardware/Software that enables one computer system to run programs that are written for another computer system.
- For example emulator 8086, 8086 microprocessor programs.
- An emulator is used on the target processor (the processor for which the program is being written).

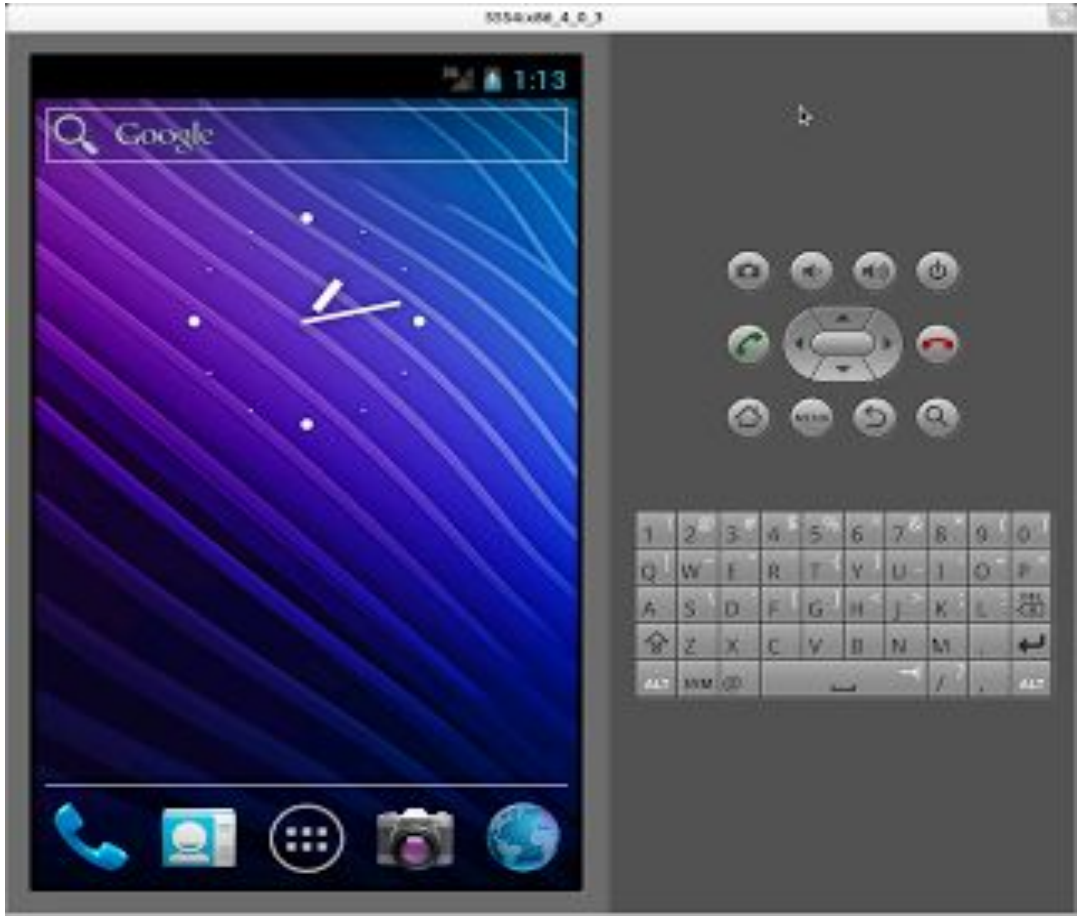
Interpreters

- convert each high level instruction into a series of machine instructions and then immediately run (or execute) those instructions.
- Allow a computer to interpret or understand, what the software program tells the computer to do, what task to perform.

DIFFERENCE BETWEEN

COMPILER	INTERPRETER
READS ENTIRE PROGRAM AND LISTS ALL ERRORS AFTERWARDS.	READS PROGRAM LINE BY LINE AND STOPS EXECUTION ON ENCOUNTERING ERROR
MEMORY REQUIRED IS MORE DUE TO INTERMEDIATE OBJECT CODE	MEMORY EFFICIENT AS NO INTERMEDIATE CODE IS GENERATED
OVERALL EXECUTION TIME IS FASTER	EXECUTION IS SLOWER AS AFTER EVERY STATEMENT THE INTERPRETER CHECKS FOR ERRORS
DEBUGGING IS DIFFICULT AS YOU HAVE TO COMPILE EVERYTIME YOU CORRECT AN ERROR	DEBUGGING IS EASY AS THE INTERPRETER IMMEDIATELY INDICATES THE ERRORS
EXAMPLE : C, C++	EXAMPLE: BASIC, PYTHON

Emulator Example



Android Virtual Machine(AVM)

Thank you!!!
Any Question..??

Language Processor

- *language processor, a program that processes programs written in a programming language (source language)*
- language processor is a *language translator, which translates the program from the source language into machine code, assembly language, or some other language.*
- The machine code can be for an actual computer or for a virtual (hypothetical) computer.

- Language processor
- 1. Analysis Phase
- 2. Synthesis Phase