Chapter one

Introduction to System programming

What is system programming?

- SP is a program that used to design and write a system software.
- i.e. It provides a platform for other software's to ***be built upon.***
- ***System Programming involves the study of***
  - ✓ ***Process Management***
  - ✓ ***Memory Management***
  - ✓ ***Filesystem***
  - ✓ ***Input/Output***

SP involves the development of ***the individual pieces of software*** that allow the ***entire system to function*** as a single unit.

What is system software?

❑ ***System software*** is a type of computer program that is **designed to run** a computer hardware and application programs.

- ✓ Examples of system software include Os, game engines, industrial automation, and a like.
- ✓ Such software is not considered as a system software ***when it can be uninstalled*** usually **without affecting the functioning of other software**.

Cont…

- The primary distinguishing characteristic of systems software to application software is that:-
  - Application software:-is a software which ***provides services to the user*** (e.g. word processor),
    - ✓***It can be uninstalled*** usually **without affecting the functioning of other software**.
  - Systems software is a software which ***provides services to the computer hardware and application program*** (e.g. Os).
    - ✓***It can't be uninstalled*** usually **without affecting the functioning of other software**.

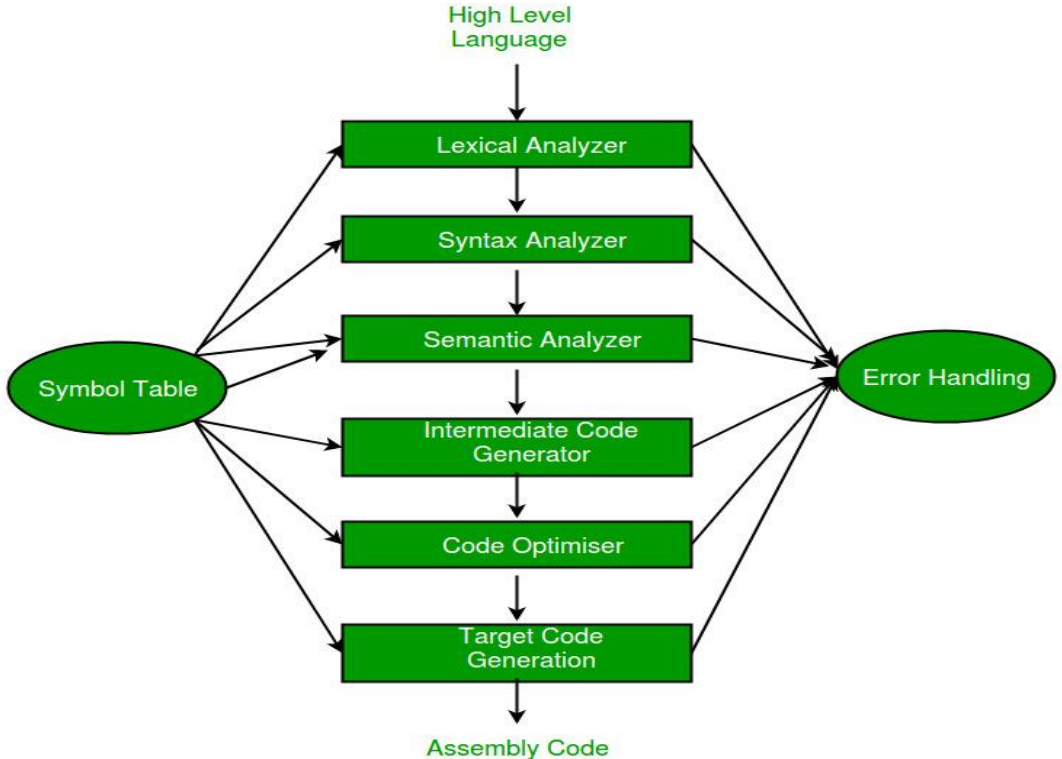
What is language processor?

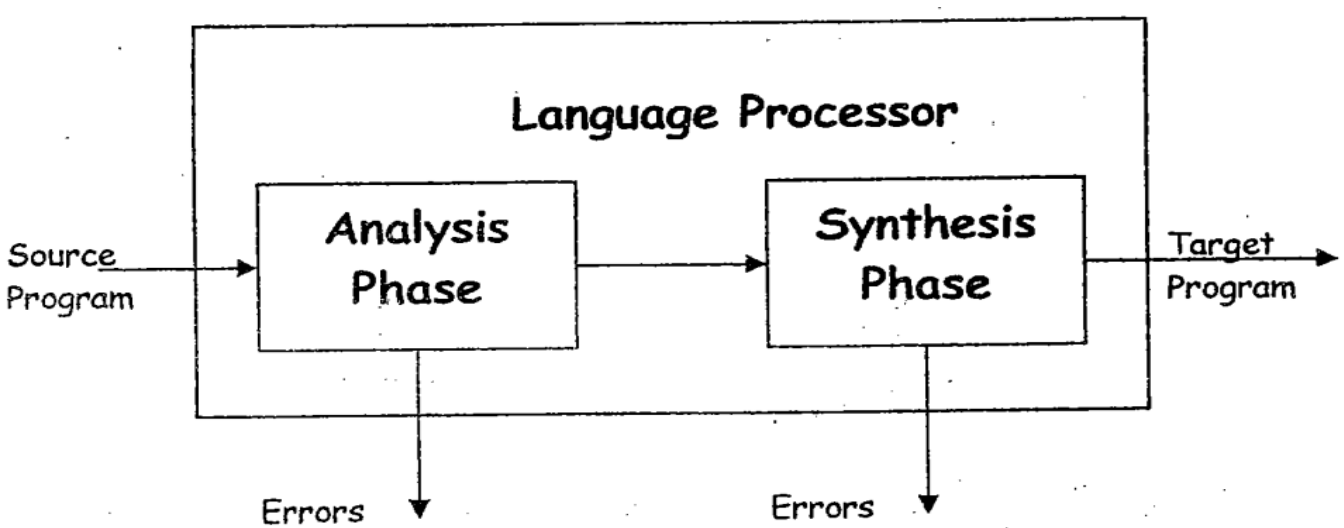
- A **language processor** is a software program designed or used to perform tasks, such as processing/translating a source program code in to machine code.

- ***Language Processor = Analysis of Source Program + Synthesis of Target Program.***

  ✓ The **analysis phase** creates an **intermediate representation** from the given source code.

  ✓ The **synthesis phase** creates an **equivalent target program** from the intermediate representation.





Cont….

- **Lexical Analyzer/scanner**: It reads the characters from the source program and groups them into lexemes (sequence of characters that “go together”).
- **Syntax Analyzer**– It is sometimes called a parser. It takes all the tokens one by one and uses Context-Free Grammar to construct the parse tree.
- **Semantic Analyzer** – It verifies the parse tree, whether it’s meaningful or not.
- **Intermediate Code Generator**– It generates intermediate code, which is a form that can be readily executed by a machine
- **Code Optimizer**– It transforms the code so that it consumes fewer resources and produces more speed.
- **Target Code Generator** – The main purpose of the Target Code generator is to write a code that the machine can understand and also register allocation, instruction selection, etc.

Cont…

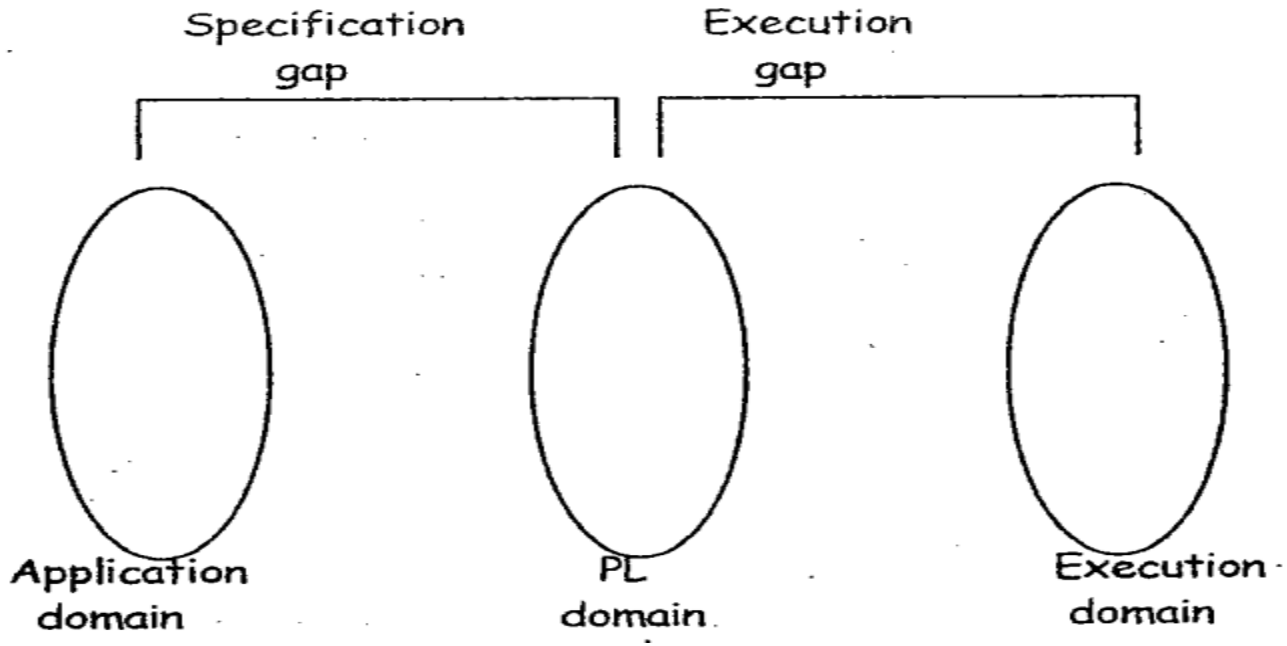
- ***Some of the activates of LP:***
  - Bridging the gap between specification and execution gap
  - Translating one level of language to another
  - detect error in source during analysis and synthesis.
  - Program generation and execution

Cont…

One of the important of LP is bridge the semantic, specification and execution gap between application domain, PL domain and execution domain.

Consequence of development gaps:

- Large development time
- high development effort
- Poor quality of software
- Late delivery of product

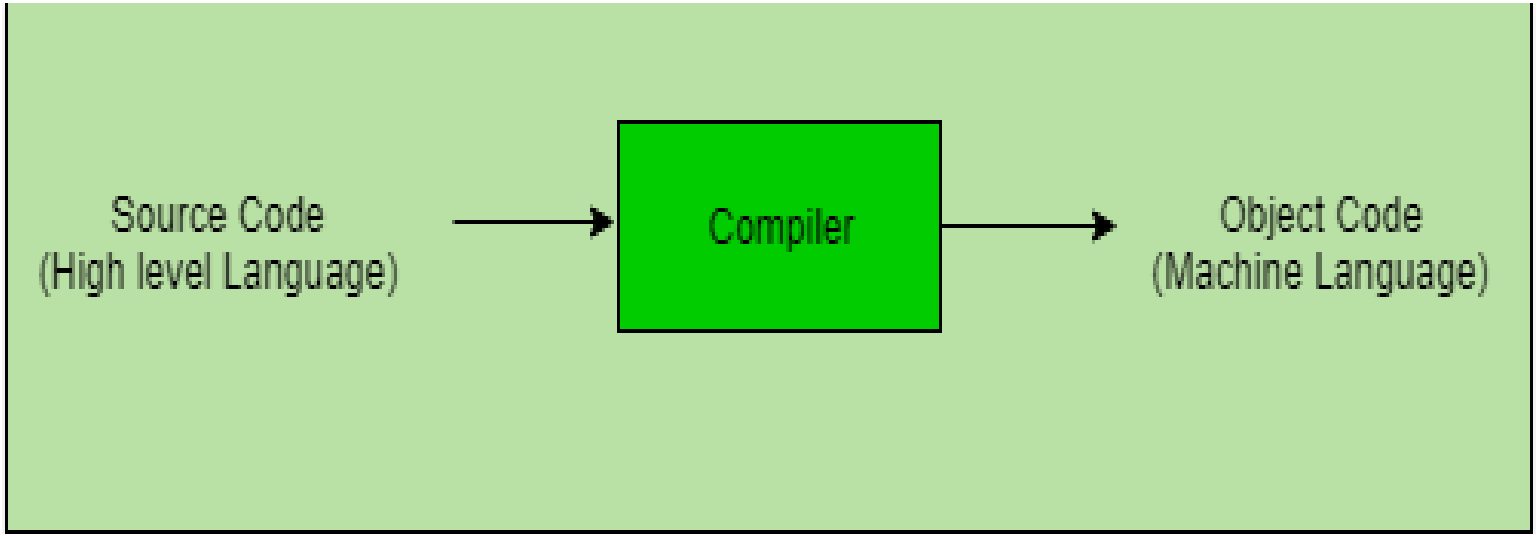


- The gap b/n application domain and PL domain is called specification gap and it bridged by ***SW development teams.***
- The gap b/n PL and execution domain is called execution gap and it bridged by ***designer of programming language processor***.

Cont…

➢ ***Some of the language processors are:-***

❑ **Compiler:-** The language processor that reads the complete source program written in high level language as a whole in one go and translates it into an equivalent program in machine language is called Compiler. **Example:** C, C++, C#, Java



Cont…

- **Interpreter -** The translation of single statement of source program into machine code is done by language processor and executes it immediately before moving on to the next line is called an **interpreter**. Example:-Python and Matlab, R studio.
  - ✓ ***Interpreter takes a source program and runs it line by line, translating each line as it comes to it.***

Compiler vs Interpreter

Compiler	**Interpreter**
Compiler convert entire	Interpreter translate code one line at a time
It produce optimized code.	Code is not too much optimized
Program execution is faster	Program execution is relatively slow.
Program analysis time is more	Program analysis time is less
Execution gap is present	Execution gap is absent

What is Operating System?

❖Os is a program that designed to control the operation of a computer system.

❖ Os act as an interface between ***computer and the users/outside*** world.

❖ Os is the layer between users and computer /hard ware parts.

❖ What are the faction of OS?

- ✓Process Management
- ✓Memory management
- ✓File Management
- ✓Device Management

Cont……

- **Process Management**:
  - ✓ Is a process of assignment/allocation of **processor** to different tasks/processes being performed by the computer system.
  - ✓ It allows two or more processes to be executed at a time, using multitasking concept.
- **Memory management**
  - ✓ **Allocating:** Assign main memory and other storage areas to the system programs as well as user programs.
  - ✓ **Swapping:** exchange the content of memory to the disk storage.
  - ✓ **Paging:** means the ***virtual address space is divided*** into units. It helps the program data is loaded into pages of memory.
- **Device Management (Input/output management)**
  - ✓A coordination and assignment of different input and output device to a given process, while one or more programs are being executed.
- **File Management**
  - ✓It allows all files to be easily created, deleted and changed or modified using text editors or any other files manipulation.

Types of Operating Systems

- There are several types of operating systems. The most widely used
  - ✓**Windows:** is the popular Microsoft brand preferred by most personal users. It is commercial and less secured compared with Unix/Linux Os.
  - ✓**Unix/Linux**
    - UNIX: It is well known for its ***stability*** and often used more in a server side rather than a workstation.
    - Linux: was designed based on the UNIX system, with the source code being a part of GNU open-source project.
  - ✓**Macintosh:** Recent versions OS, and it is efficient & easy to use, but can only function on ***Apple branded*** hardware.

❑ Type of Os based on the:-

- ➢Number of users they can serve at one or different stations and
- ➢Number of programs the operating system handle/ support on a time,
- ➢Times required to response the given request/input.

❑**Real-time operating system** (RTOS):

- ✓This kind of Os is used when there are time requirement are ***very strict*** like missile system, robots, scientific instruments and other industrial systems.
- ✓This Os very small response time. i.e. The time interval required to process and response to input is very small.

Advantage

- ✓ Error free
- ✓ Effective results

Disadvantage

- ✓Limited tasks run at the same time
- ✓Complex algorithm for designer to write on.
- ✓RTOS has very little user-interface capability, and no end-user utilities.

Cont….

❑**Single-user, single task:**

- This kind of OS allows only one user to access the system at a time. i.e. It runs only one program at a time.
- Examples: MS-DOS (Micro Soft Disk Operating System) produced by Microsoft.

Advantage

- Execution speed is good b/c only one process run a time.

Disadvantage

- A low degree of resource utilization b/c a number of resources are present in a computer system and only one of these resources being utilized by the process

❑ **Single-user, multi-tasking:**

- In this operating system, all resources are dedicated to only one user. But the user can use more than one program (resource).
- It overcomes a single user-single tasking operating system limitation.
- Single user have a permission to run multiple programs at the same time.
- Example Microsoft's Windows (98, 2000, XP, Vista etc) and Apple's MacOS X.

❑**Multi-user multi tasking**:

- It is mostly used on the networked environment.
- More than one user can utilized the resources available on the computer simultaneously like hard wares and soft wares.
- Example: Linux, UNIX, etc

When OS start its work?

- **OS start its work, w**hen the computer is turned on, at that time the Os will be loaded from the ***hard drive into the computer's memory***.
- The process of loading the operating system into memory is called **bootstrapping**, or **booting** the system.

Cont….

What Is Linux?

- ✓ Linux is one of the type of OS (mostly coded in C)goes by the name *Linux kernel.*
- ✓ Linux is a *multiuser, multitasking OS.*
- ➢ It provides a number of facilities:
  - ✓ It used to mange hardware resources
  - ✓ It helps to mange directory and file
  - ✓ It used to loading a programs
  - ✓ It used to execution a program and
  - ✓ used to suspend a program if any interference.

Why Use Linux?

- ✓Multi-tasking / multi-user
- ✓Networking capability
- ✓Graphical (with command line)
- ✓Portable(PCs, mainframes, super-computers)
- ✓Free and Shareable! (LINUX, FreeBSD, GNU)
- ✓Security
- ✓Compatible with Older Hardware/ platform Independent

The Unix Philosophy

- ✓Powerful
- ✓Simple
- ✓General/not specific
- ✓Extensible

Cont.…

- ❑Activities that we should know during operating system installation are:-

✓Understanding partitioning techniques

✓Identify the difference b/n **window and other Os** file systems.

✓Check available space and if it is **inefficient *freeing up space***

Chapter three

Assemblers

- Program write in assembly language are compiled by an assembler.
- Every assembler has its own assembly language ,which is designed for one specific computer architecture/microprocessor.
- An assembler is a program that accepts an input in assembly language and produce a machine language equivalent to the source code.

❑ **The basic assembler functions are:**

- Translating symbolic/mnemonic language code to its equivalent object code.
- **Assigning** machine addresses to symbolic labels.

Cont…

- The design of assembler can be to perform the following activities:
  - ✓ **Scanning:-**tokenizing-
    - ✓ **Tokenization** is the act of **breaking up a sequence of strings into pieces** such as words, keywords, phrases, symbols and other elements called tokens.
  - ✓ **Parsing:-**validating the syntax and semantic validity of the given instructions.
  - ✓ **Creating the symbol table**:- for files and strings;
  - ✓ **Resolving the forward references:** symbol/variable is referenced before it is declared.
  - ✓ **object code conversion:-**Converting into the machine language

❑ Advantage of assembly languages

- ➢ It allows to get the **better performance**, b/c assembler language run faster than high level language.

❑ Disadvantage of assembly languages

- ✓ It requires the use of an **assembler:-**to translator a source program to machine code
- ✓ Assembly language **is not portable** b/c assembly language are specific to a given microprocessor.

Statements in assembly language program

❑ Imperative statement:-

- ✓ It indicates actions to be performed **during the execution of the assembled program**.
- ✓ It manly direct on what type of operations performed by the written program.
- ✓ E.g. Move, Read, write…etc.

❑ Declarative statement:-

- ✓ It used to declares constants or storage areas in a program.
- ✓ Declaration [Label/name] DC <values(how much word can contain)>
- ✓ The DS a keyword used for reserve/declare storage. e.g. A DS 1
- ✓ The above statement reserves a memory area of 1 word and associates the name A with it

❑ Directive statement:-

- ✓ It provide instructions to the assembler itself, that means . They are only used to instruct assembler to perform certain actions e.g. START,END..etc.
- ✓ It is not translated into machine operation codes

Cont…

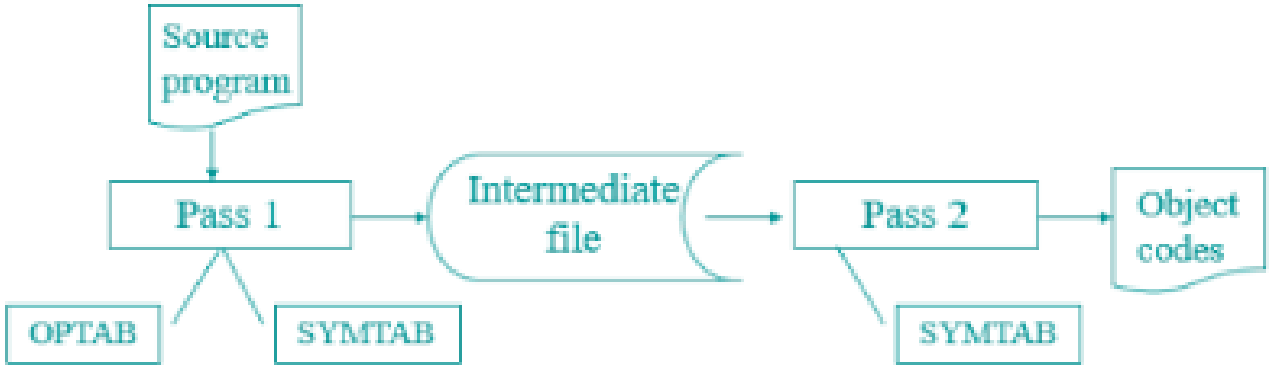
- **Basic assembler directives:-**
  - ✓ START: specify name & starting address.
  - ✓ END : specify the end of source program

Cont…

- **What is forward reference?**
  - The assembly language programming states that the ***symbol must be defined*** in the program.
  - So, there can be an instance in which ***a symbol/ variable is referred before it is defined*** such a reference is called **forward reference.**
- **What is Forward reference problem?**
  - The function of assembler is:- to **replace each symbol by its machine address** and
  - If we refer/use to that **symbol before it is defined its address is not known by the assembler** such a problem is called **forward reference problem**.

Cont…

- **What is the solution for forward reference problem?**
  - ✓ Forward reference problem in IBM 360 is solved by making **two passes over** the assembly code.
  - ✓ In two pass assembler, in the first pass Tokenization, parsing symbol table creation and intermediate code/file generation done. In the second pass code optimization and object code generation done and



- **Single-pass Assembler:**
  - ✓ In this case the whole process of **scanning, parsing, code optimazation and object code conversion** is done in single pass/simultaneously.

Two pass assembler

1. **Memory-**
   - ✓ A basic unit of a memory in the IBM 360 is **a byte**.
   - ✓ Each addressable position in a memory can contain eight bits of information.

Unit of Memory	Bytes	Length in bits
Byte	1	8
Halfword	2	16
Fullword	4	32
Doubleword	8	64

The size of the 360 memory is up to 2^{24} bytes (16 MB)

Cont…

2. Register:-

- The IBM 360 has 16 general purpose registers consisting of 32 bits each.
- General purpose registers are used to store temporary data within the microprocessor.
- In addition there are **floating point registers** consisting of 64 bits each.
- It has a 64 bit program status word(PSW)that contain program status information related to-
  - ✓ The value of the location counter
  - ✓ Protection information's
  - ✓ Interrupt status

Cont…

3. Instructions:-

- **In IBM 360 there are two type of operands such as:-**
  - **Register Operand:-** refers to a **data stored** on **one of the 16 general register**.
  - **Storage Operand:-** refers to **the data in** the **core memory**

❑ **Features of assembly language required two build a two pass assembler are?**

1. Mnemonic/ Symbolic opcode specification:-

✓Instead of using a numeric operation codes symbolic /mnemonic can be specified

2. Declaration of data or storage areas:-

✓Assembly language can be used to specify some part of the memory for storage purpose.

CHAPTER FOUR

LOADERS AND LINKERS

- The Source Program written in assembly language or high level language will be converted to object program.
- This conversion is done either using ***assembler or compiler*** which contains:-
  - ✓ **Translated instructions (obj prog)**
  - ✓ Data values from the **source program**
  - ✓ Specifies **addresses in primary memory** for execution.

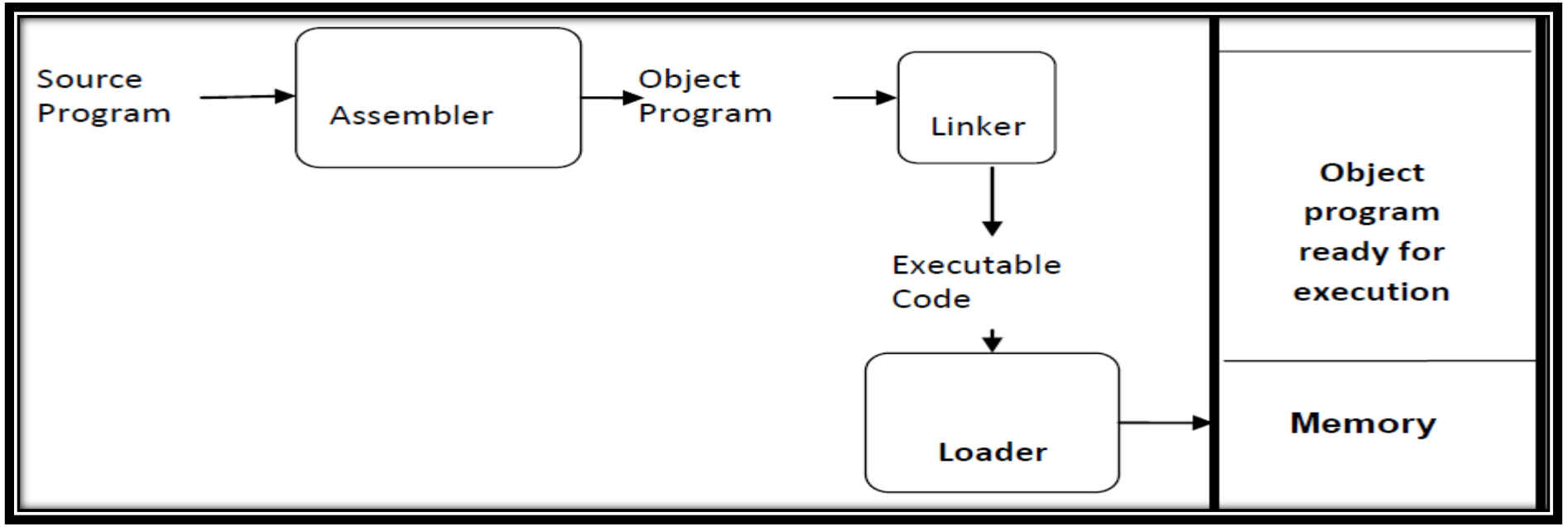
Cont...

Basic Functions of Loader

- ✓ **Allocation-**allocate a memory space for the object program.
- ✓ **Linking** – a process of combines ***two or more separate object programs*** and also supplies the information needed to reference them-(**Linker**).
- ✓ **Loading** – is a process of physically placing the machine instructions and data in to memory-**(loader).**
- ✓ **Relocation** – when links/modifies the object program it should be loaded at an address different from the location originally specified-**(Linking Loader).**

What is loader?

- It used to allocate ***memory location and brings the object program*** into memory for execution.
- The role of loader is as shown in the figure 1.
- In figure 1 assembler, which **generates the object program** and later loaded to the memory by the loader for execution.



Cont.…

Advantage

- Simple to implement

Disadvantage

- A portion of a memory is waste b/c the **space occupied by assembler** is unavailable to the object program.
- It is necessary to **retranslate a source programs** every time it is run.

Cont.…
Type of Loaders

- There are different types of loaders such as, absolute loader, bootstrap loader, and relocating loader (relative loader).
- **Absolute Loader**:-

✓ Is a loader, which is used only for loading activates.

✓ i.e. It doesn't need for **relocation and linking** activates rather than simple loading .

✓ b/c of this the operation of absolute loader is very simple ;

How absolute Loader is working?

✓ First the **object code** is loaded to **specified address** in the memory.

✓ At the end the loader jumps to the **specified** address to begin execution of the loaded program.

Cont.….

A Simple Bootstrap Loader

- When a computer is **first turned on or restarted**, a special type of absolute loader, called bootstrap loader is executed.
- This bootstrap loads the **first program** to be run by the computer -- usually an operating system.

Cont.…

Machine-Dependent Loader

- ❑ Absolute loader is **simple and efficient**, but the scheme has **potential disadvantages**
- ✓ The programmer has to specify the ***actual starting address***, from where the program to be loaded. This does not create difficulty, ***if one program to run***, but not for several programs.
- ❑ This motivate to design and implement **a more complex loader.**
- ❑ It helps the loader provide program **relocation and linking activities** , as well as simple loading functions.

Cont……

Relocating loaders or Relative loaders

- It is the execution of an **object program** using ***any part of the available and sufficient memory***.
- The **object program** is **loaded into memory** wherever there is space for it.
- Relocation provides the **efficient sharing of the machine** with larger memory, when ***several independent programs*** are to be run together.
- The loaders that allow for program relocation are called ***relocating loaders or relative loaders.***

Thank You!!!