CONTENTS

- Components Of System Software – A walkthrough.
- Introducing loader.
- Basic Loader functions.
- Steps involved in loading.
- Loader Schemes.
- Types of loaders.

COMPONENTS OF SYSTEM SOFTWARE

COMPILER

- Translates high-level language programs into assembly language programs.

ASSEMBLER

- Converts assembly language programs into object files.
- Object files contain a combination of machine instructions, data, and information needed to place instructions properly in memory.

Contd...

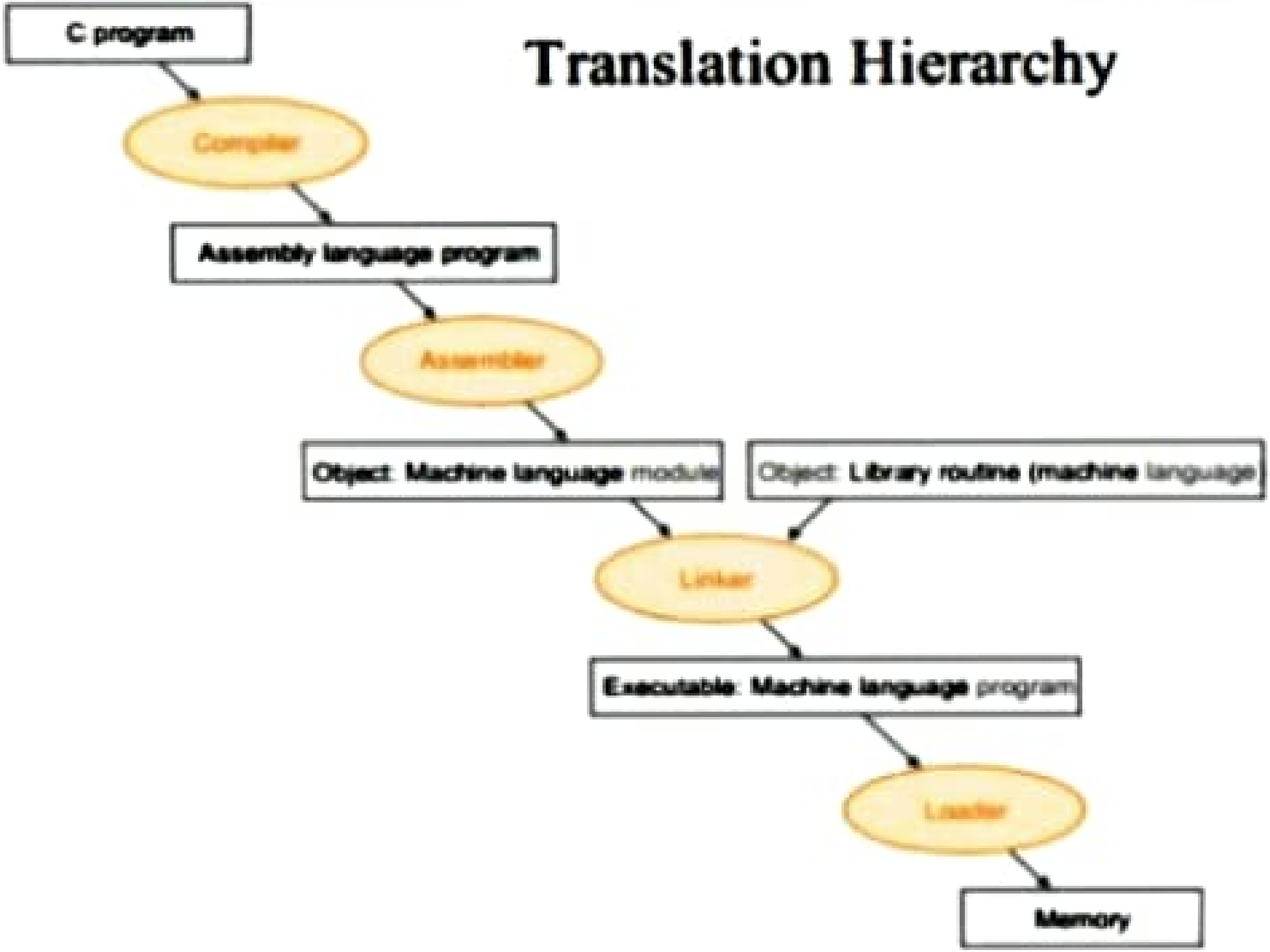
LINKER

- Tool that merges the object files produced by separate compilation or assembly and creates an executable.

LOADER

- Part of the OS that brings an executable file residing on disk into memory and starts it running.

Translation Hierarchy



WHAT IS A LOADER ?

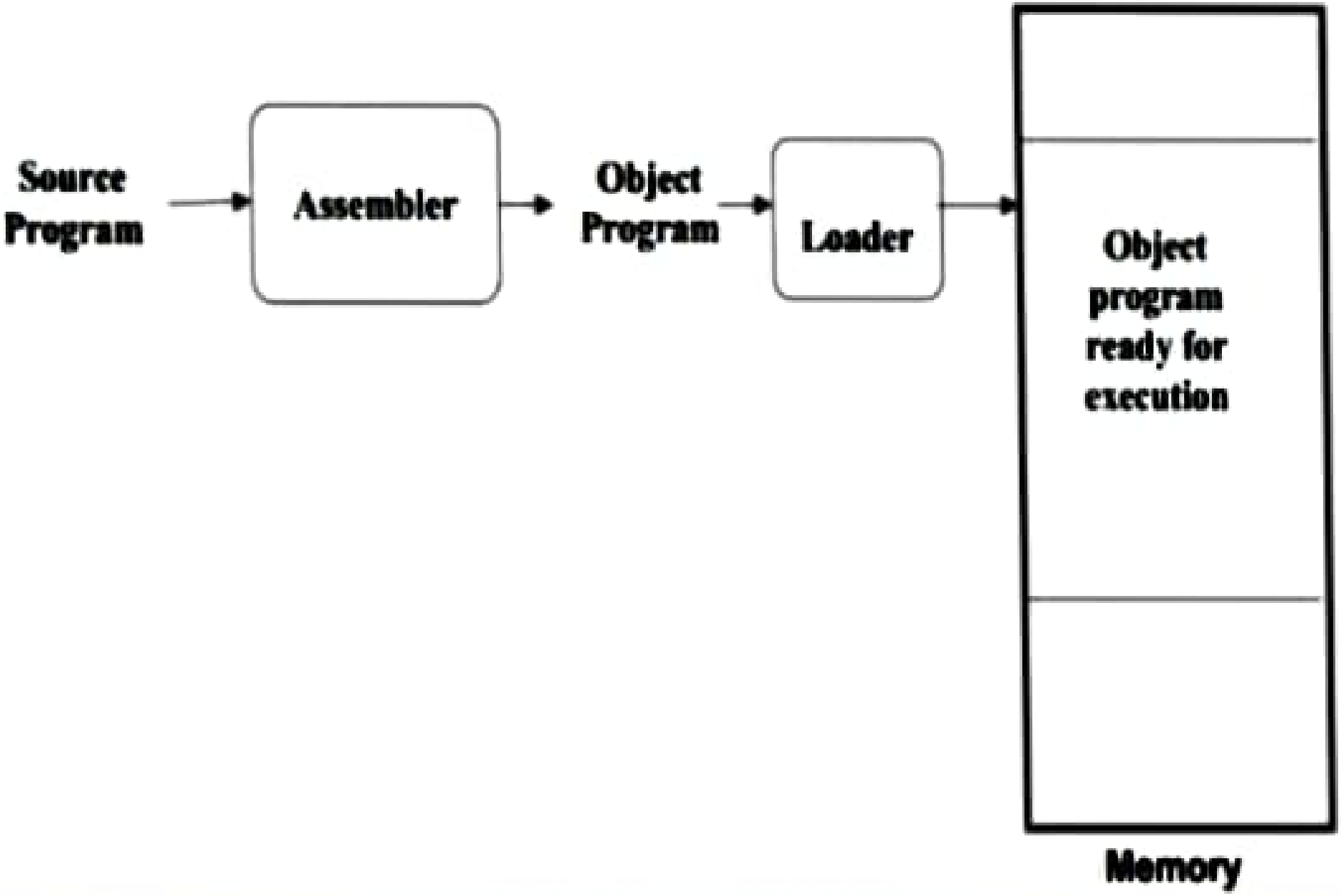
- Loader is a system program that loads machine codes of a program into the system memory.
- Loading a program involves reading the contents of executable file into memory.
- Once loading is complete, the operating system starts the program by passing control to the loaded program

Loader & its Functions

- A loader is a system program, which takes the object code of a program as input and prepares it for execution.

- **Loader Function** : The loader performs the following functions :
  - *Allocation* - The loader determines and allocates the required memory space for the program to execute properly.
  - *Linking* -- The loader analyses and resolve the symbolic references made in the object modules.
  - *Relocation* - The loader maps and relocates the address references to correspond to the newly allocated memory space during execution.
  - *Loading* - The loader actually loads the machine code corresponding to the object modules into the allocated memory space and makes the program ready to execute.

ROLE OF LOADER



STEPS INVOLVED IN LOADING

- Executable file' s header is read to determine the size of text and data segments.
- Instructions and data are copied into address space.
- Arguments passed to the program are copied on the stack

Contd...

- Machine registers including the stack pointer are initialized.
- The control is transferred to a start-up routine that copies the program' s arguments from the stack to registers and calls the program' s main routine.

LOADER SCHEMES

- "Compile-and-Go" Loaders
- General Loader Scheme

"COMPILE-AND-GO" LOADERS:

- Assembler will be running in one part of memory.
- The assembled data & machine instructions are directly assigned into the memory locations.
- Example : FORTRAN

Contd...

- When the assembly is over, the control transfers to the starting instruction of the program.
- The loader consists of an instruction to make this control transfer.

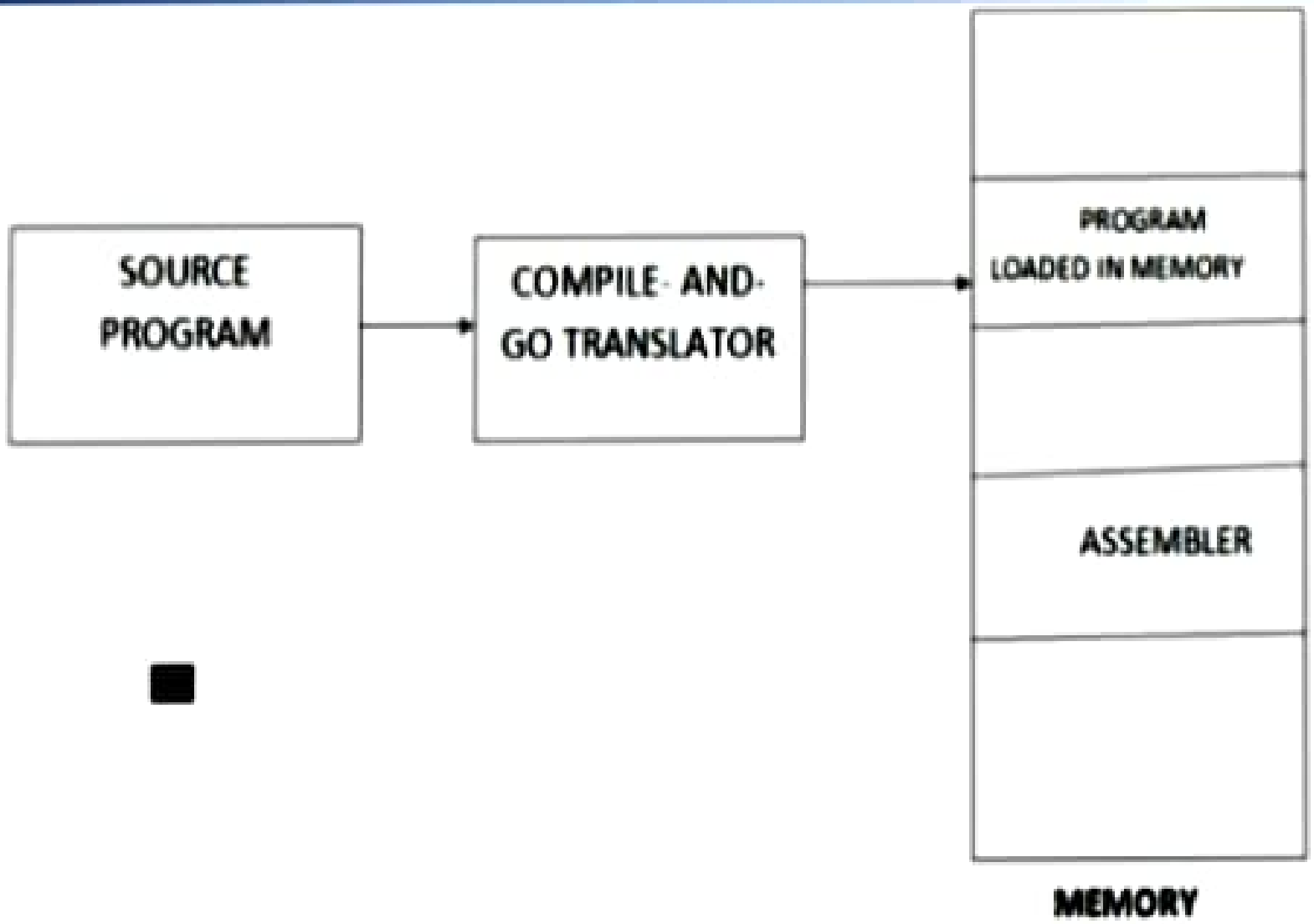
DISADVANTAGES :

- Memory wastage due to the unavailability of assembler to the object program.

Contd…

- Need to assemble the program every time it's run.
- Difficulty in handling if the source programs are in different languages.

"COMPILE-AND-GO" LOADER SCHEME



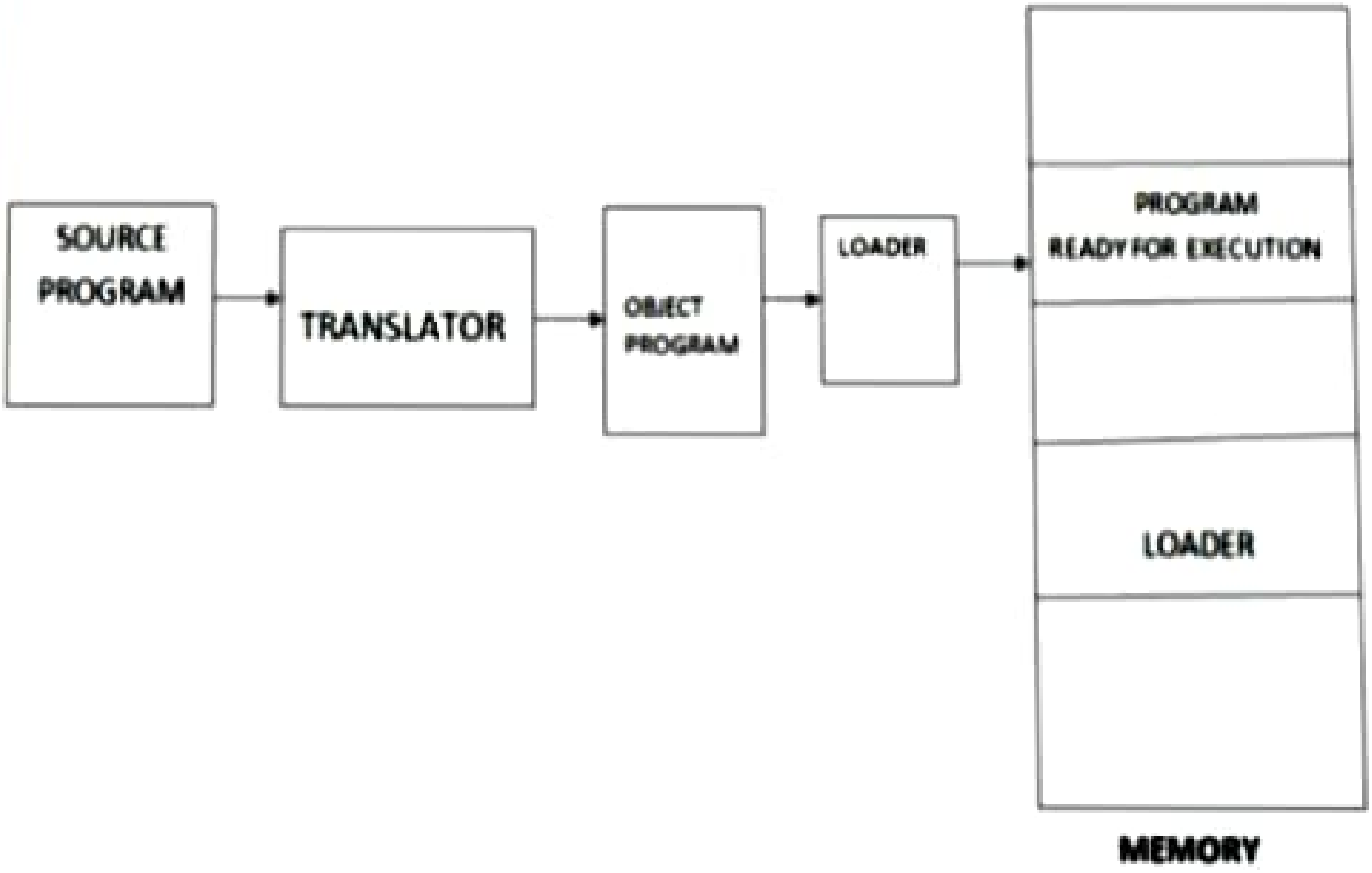
GENERAL LOADER SCHEME

- Uses an object deck as intermediate data.
- Here the loader accepts the assembled machine instructions and data in the object format and places it in the memory in an executable form.

ADVANTAGES :

- Smaller than assembler.
- No reassembly is needed.
- Possible to write subroutines in different languages.

GENERAL LOADER SCHEME



TYPES OF LOADERS

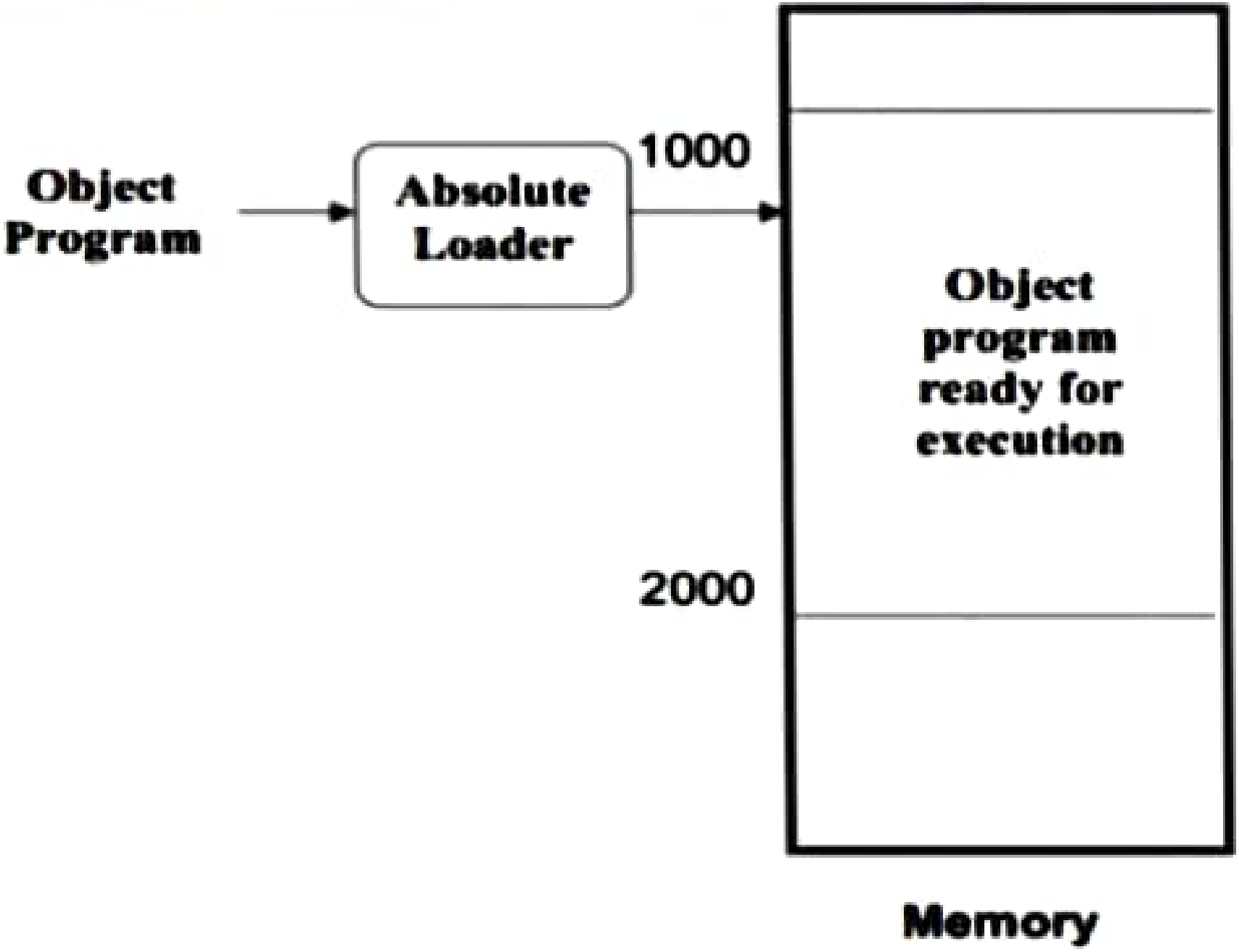
- Absolute Loader.
- Bootstrap Loader.
- Relocating Loader (Relative Loader)
- Linking Loader.

ABSOLUTE LOADER

- Absolute loader loads the object code to specified locations in the memory.
- At the end the loader jumps to the specified address to begin execution of the loaded program.
- ADVANTAGES : Simple and efficient.
- DISADVANTAGES :

- The need for programmer to specify the actual address.
- Difficulty in using subroutine libraries.

ROLE OF ABSOLUTE LOADER



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.
- The bootstrap itself begins at address 0.

RELOCATING LOADER

- Loaders that allow for program relocation are called Relocating loaders or Relative loaders.
- CONCEPT OF RELOCATION :

- The execution of the object program is done using any part of the available & sufficient memory.
- The object program is loaded into memory wherever there is room for it.

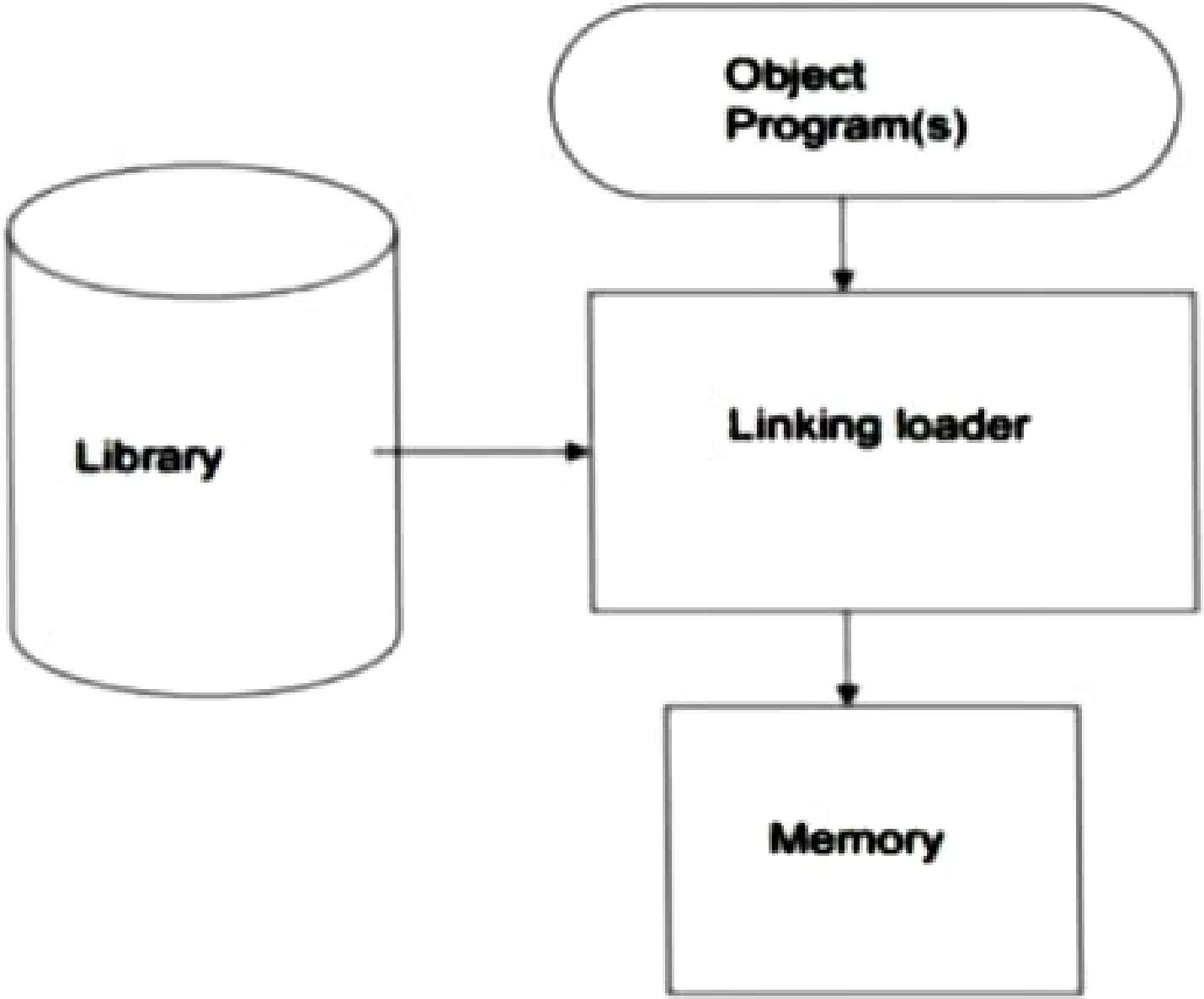
Contd...

- The actual starting address of the object program is not known until load time.
- Relocation provides the efficient sharing of the machine with larger memory and when several independent programs are to be run together.
- It also supports the use of subroutine libraries efficiently.

LINKING LOADERS

- Perform all linking and relocation at load time.
- The Other Alternatives :
  - Linkage editors – which performs linking prior to load time.
  - Dynamic linking – where linking function is performed at execution time.

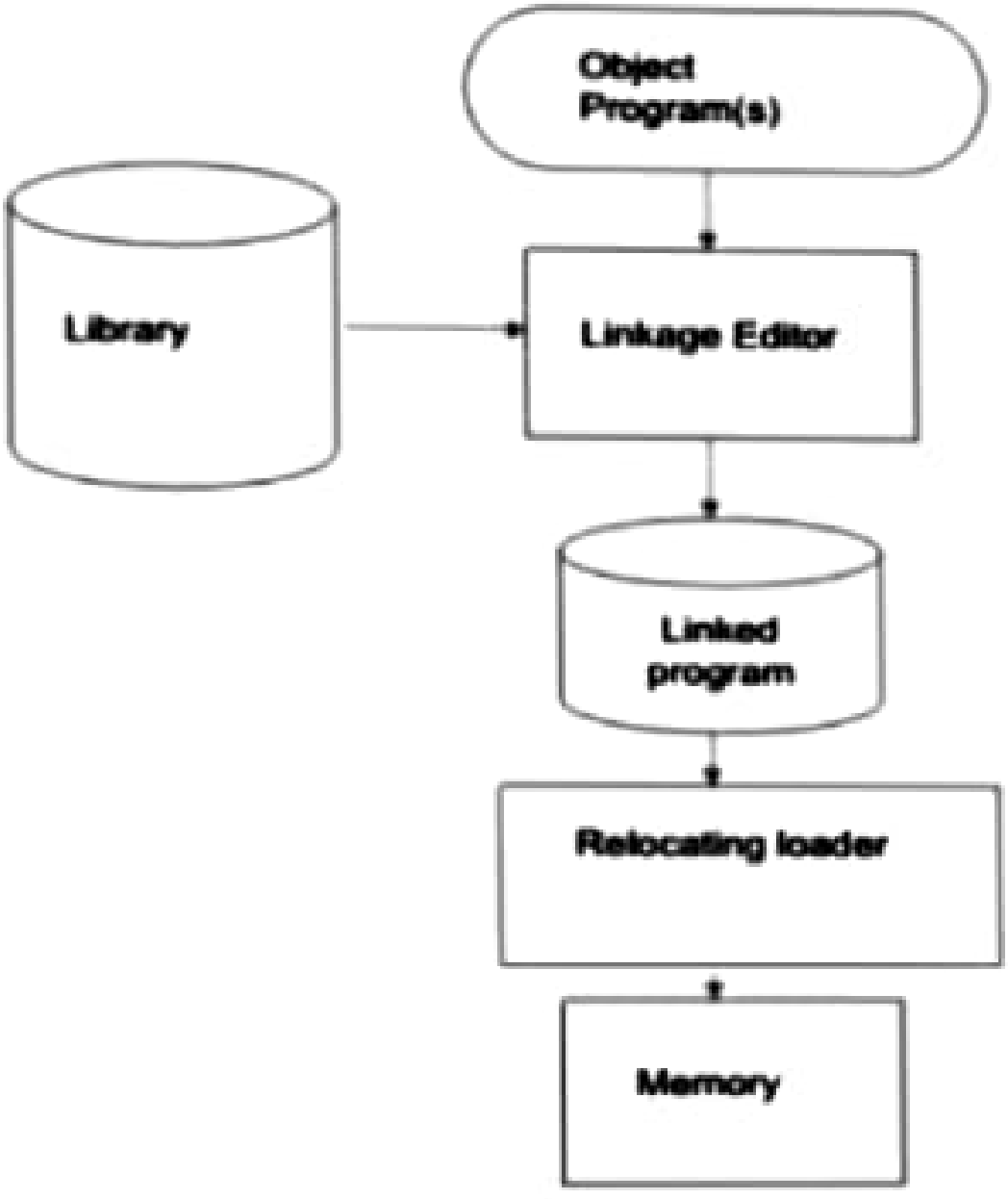
ROLE OF LINKING LOADERS



LINKAGE EDITORS

- Produces a linked version of the program – called a load module or an executable image.
- This load module is written to a file or library for later execution.
- The linked program produced is suitable for processing by a relocating loader.
- Using Linkage editor , an absolute object program can be created, if starting address is already known.

ROLE OF LINKAGE EDITOR



DYNAMIC LINKING

- This scheme postpones the linking functions until execution.
- A subroutine is loaded and linked to the rest of the program when it is first called.
- It is usually called dynamic linking, dynamic loading or load on call.
- Allows several executing programs to share one copy of a subroutine or library.

Contd...

- In an object oriented system, dynamic linking makes it possible for one object to be shared by several programs.
- Dynamic linking provides the ability to load the routines only when they are needed.
- Run-time linker uses dynamic linking approach which binds dynamic executables and shared objects at execution time.