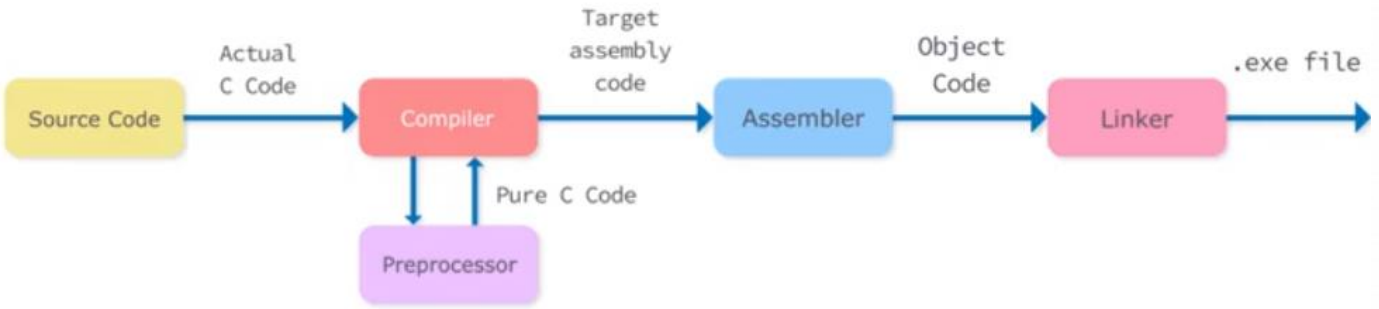
Linker

- In execution of the program, major role is played by two programs known as Linker and Loader.
- A linker is special program that combines the object files, generated by compiler/assembler, and other pieces of codes to originate an executable file have. exe extension. Linker are also called link editors.
- In the object file, linker searches and append all libraries needed for execution of file.
- It regulates memory space that code from each module will hold. It also merges two or more separate object programs and establishes link among them.



- Linking is performed at both **compile time**, when the source code is translated into machine code and **load time**, when the program is loaded into memory by the loader
- Generally, linkers are of two types :

  1. Static Linker

  2. Dynamic Linker

1. Static Linking –

- It is performed during the compilation of source program. Linking is performed before execution in static linking.
- It takes collection of relocatable object file and command-line argument and generate fully linked object file that can be loaded and run.

- Static linker perform two major task:

1) **Symbol resolution –** It associates each symbol reference with exactly one symbol definition .Every symbol have predefined task.

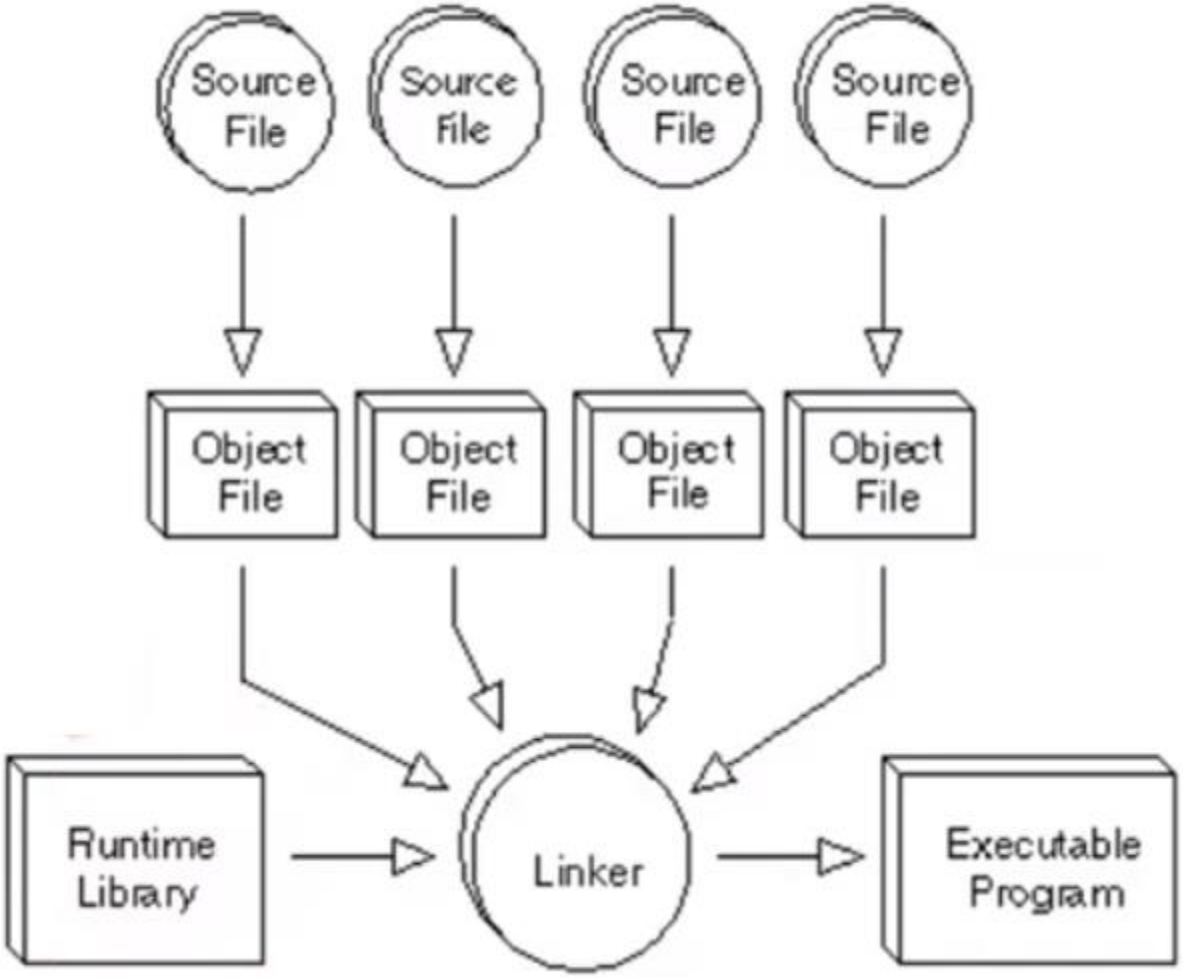
2) **Relocation –** It relocate code and data section and modify symbol references to the relocated memory location.

- The linker copy all library routines used in the program into executable image. As a result, it require **more memory space.**
- As it does not require the presence of library on the system when it is run . so, it is **faster and more portable.**
- **No failure** chance and less error chance.

2. Dynamic linking – Dynamic linking is performed during the run time. This linking is accomplished by placing the **name of a shareable library** in the executable image.

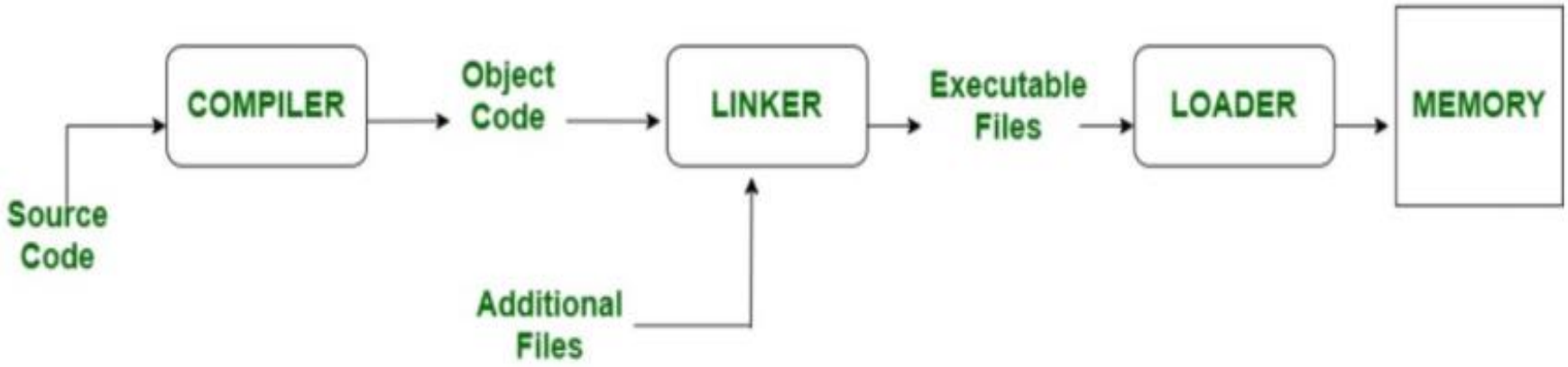
- There is more chances of error and failure.
- It require less memory space as multiple program can share a single copy of the library.

- Here we can perform code sharing. it means we are using a same object a number of times in the program.
- Instead of linking same object again and again into the library, each module share information of a object with other module having same object.
- The shared library needed in the linking is stored in virtual memory to save RAM.

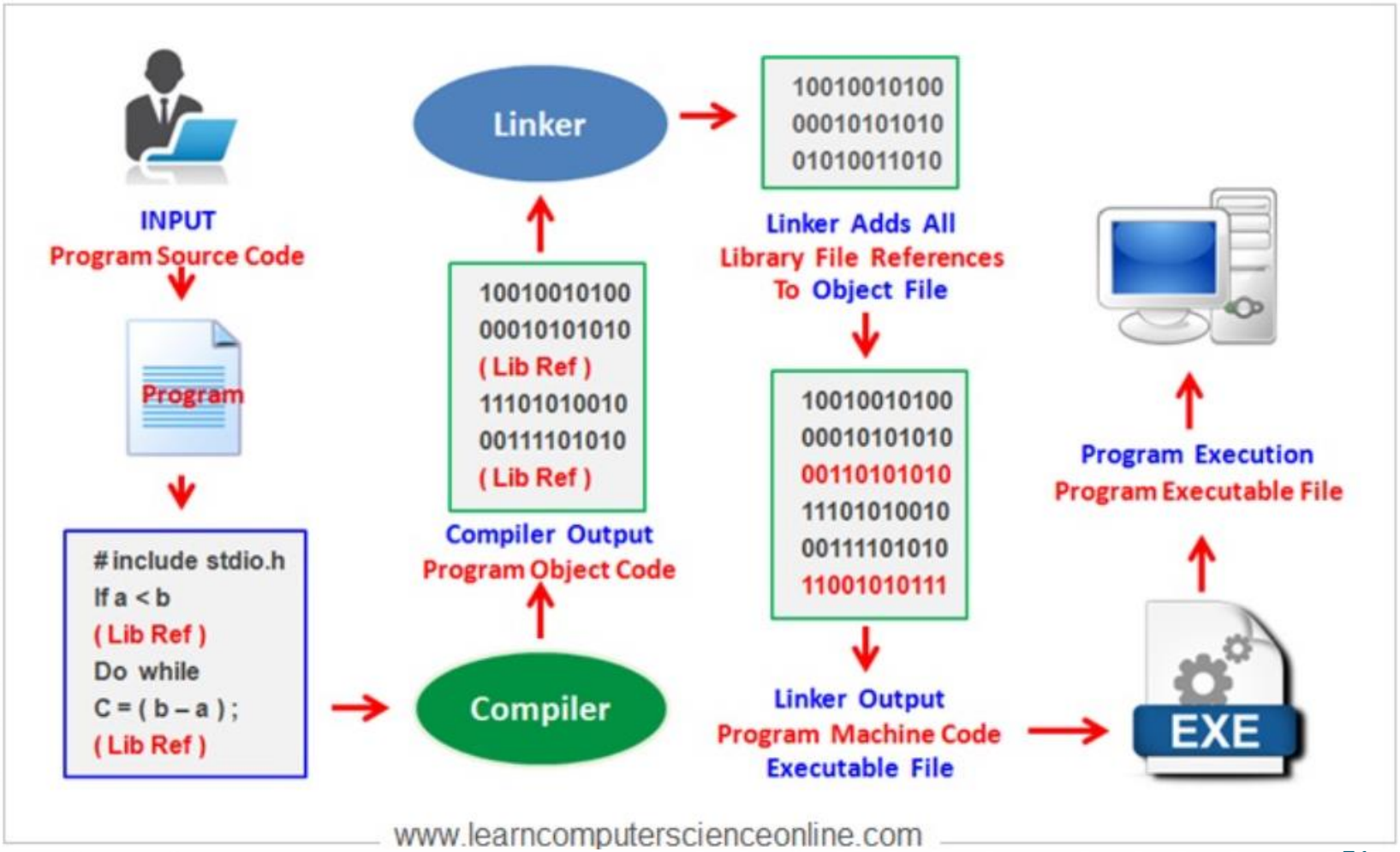

Source File
Source File
Source File
Source File
Object File
Object File
Object File
Object File
Runtime Library
Linker
Executable Program

Loader

- A loader is a piece of software that chooses exactly where to put object code in RAM, ready for it to be run. It also adjusts the memory references in programs.
- The loader is special program that takes input of object code from linker, loads it to main memory, and prepares this code for execution by computer. Loader allocates memory space to program.



Computer Program Compilation Process

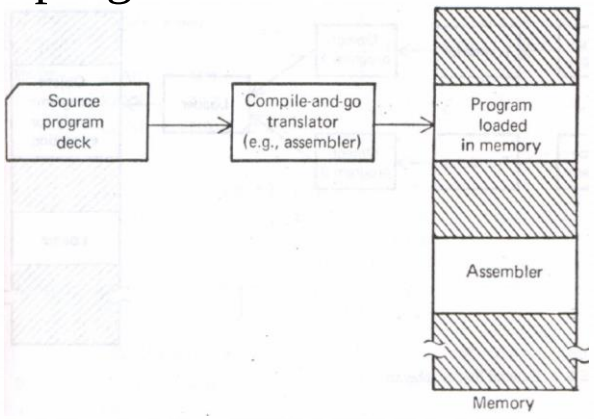


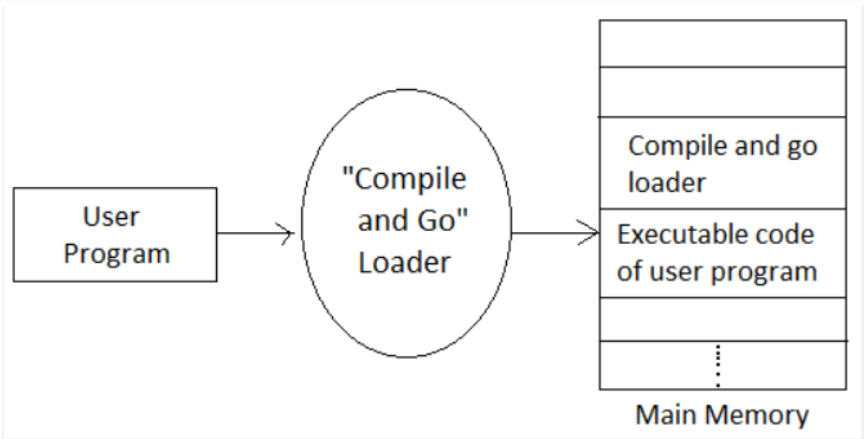
Types of Loaders

1) Absolute Loader
2) Boot Strap Loader
3) Relocating Loader
4) Direct Linking Loader
5) Assemble – and – go or Compile – and – go loader
6) Dynamic Loader

Assemble – and – go or Compile – and – go loader

- In this type of loader, the instruction is read line by line, its machine code is obtained and it is directly put in the main memory at some known address.
- Assembler runs in one part of memory and the assembled machine instructions and data is directly put into their assigned memory locations.
- After completion, the assembly process assigns the starting address of the program to the location counter.





Advantages

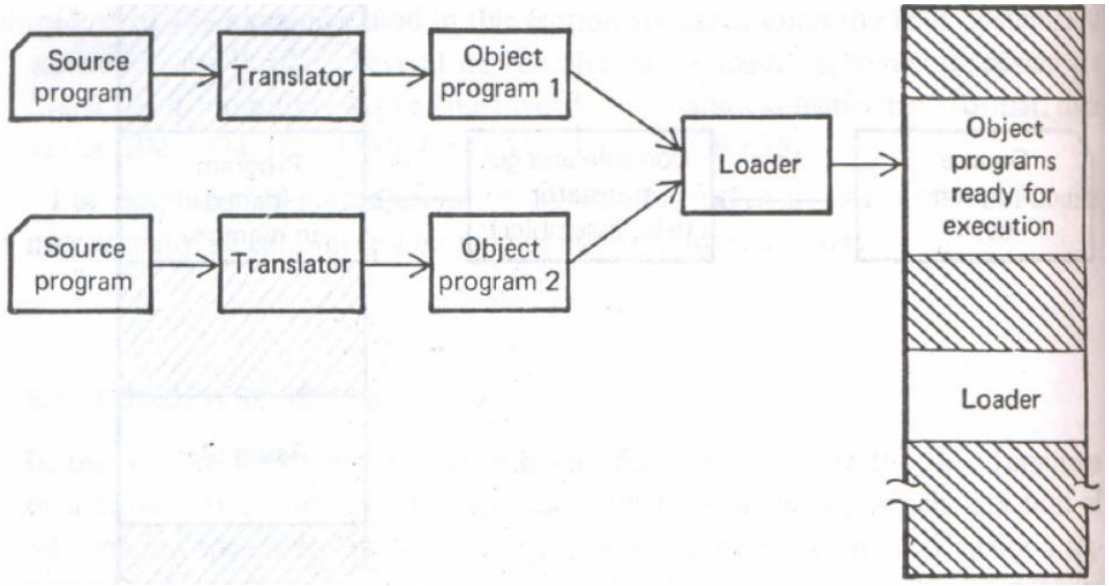
- This scheme is simple to implement because assembler is placed at one part of the memory and loader simply loads assembled machine instructions into the memory.

Disadvantages:

- In this scheme, some portion of memory is occupied by assembler which is simply a **wastage of a memory**. As this scheme is a combination of assembler and loader activities this combination program occupies a large block of memory.
- There is no production of **.obj file**, the source code is directly converted to executable form. Hence even though there is no modification in the source program it needs to be assembled and **executed each time** which then become a time-consuming activity.
- It **cannot handle multiple source program** or multiple programs written in different languages. This is because assembler can translate one source language to another target language.
- The execution time will be **more.**

General Loader Scheme

- This scheme avoids the disadvantages of preceding “compile-and-go” scheme.
- As the size of the loader is assumed to be smaller than the assembler, more memory is available to the user

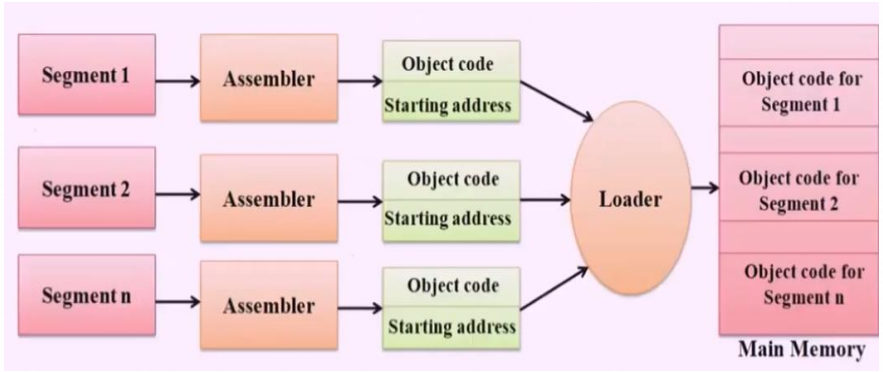


Absolute Loader

- The **absolute loader** is a kind of **loader** in which relocated object files are created, **loader** accepts these files and places them at a specified location in the memory.

- This type of **loader** is called **absolute loader** because no **relocating** information is needed, rather it is obtained from the programmer or assembler.

```
1          MAIN        START      1000
.
.
.
15                     JMP        5000
16                     STORE      ;instruction at location 2000
                       END
1          SUM         START      5000
2
.
.
.
20                     JMP        2000
21                     END
```



Program for Absolute Loader

```
Begin
read Header record
verify program name and length
read first Text record
while record type is <> 'E' do
        begin
        {if object code is in character form, convert into internal representation}
        move object code to specified location in memory
        read next object program record
        end
jump to address specified in End record
end
```

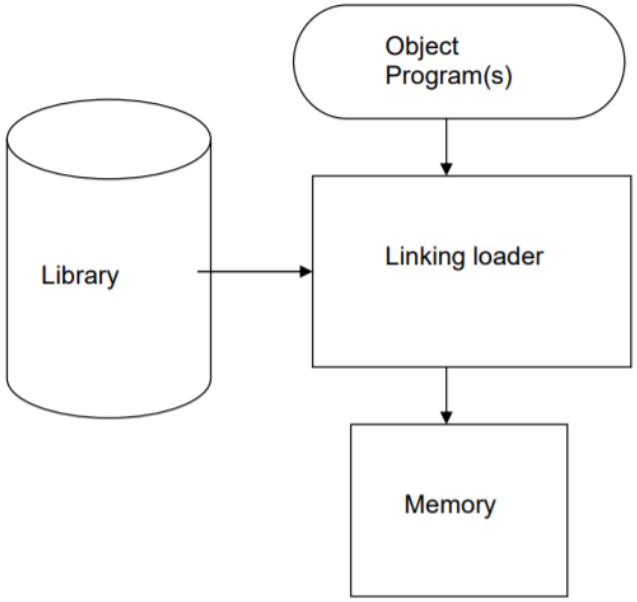
Simple Bootstrap Loader

- 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.
- The bootstrap itself begins at address 0. It loads the OS starting address 0x80.

Linking Loaders

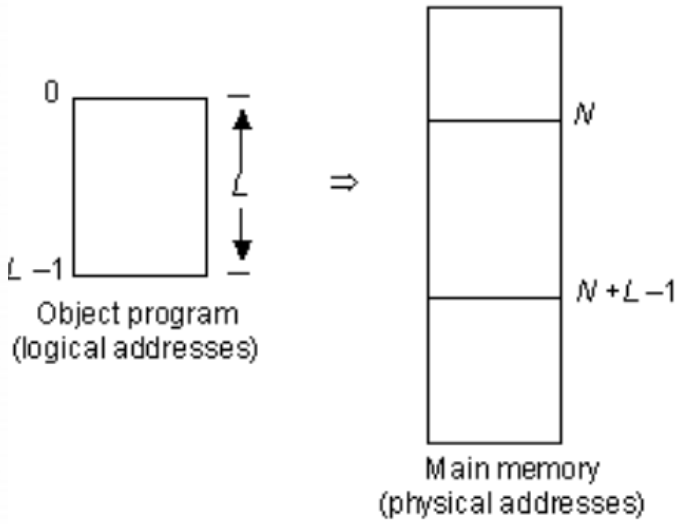
- Linking Loaders – Perform all linking and relocation at load time.

- The Other Alternatives are Linkage editors, which perform linking prior to load time and, Dynamic linking, in which linking function is performed at execution time



Relocating Loader

- In general, the user does not know *a priori* where the program will reside in memory. A *relocating loader* is capable of loading a program to begin anywhere in memory:
- The addresses produced by the compiler run from 0 to L–1. After the program has been loaded, the addresses must run from N to $N+L$–1.
- Therefore, **the relocating loader adjusts**, or *relocates*, each *address* in the program



Direct Linking Loader

- Direct Linking Loader is a general re-locatable loader.

- Allowing the programmer multiple procedure segments and multiple data segments and giving programmer complete freedom in referencing data or instruction contained in other segments.

- The assembler must give the loader the following information with each procedure or data segment.

1) Length and type of segment

1. A list of all symbols in the segment that may be referenced by other segments.
2. List of all symbols not defined in the segment but referenced in the segment.
3. Information where the address constant are loaded in the segment.

2) **Format of Databases:**

1) External Symbol Dictionary (ESD)
2) Text Cards (TXT)
3) Relocation and Linkage Directory (RLD)
4) END CARD

Dynamic Loading

- All the programs are loaded in the main memory for execution.

- Sometimes complete program is loaded into the memory, but some times a certain part or routine of the program is loaded into the main memory only when it is called by the program, this mechanism is called **Dynamic Loading**, this enhance the performance.

Functions of Loader

The loader performs the following functions:

1) Allocation

2) Linking

3) Relocation

4) Loading

Allocation:

- Allocates the space in the memory where the object program would be loaded for Execution.
- It allocates the space for program in the memory, by calculating the **size of the program.** This activity is called allocation.
- In **absolute loader** allocation is done by the programmer and hence it is the duty of the programmer to ensure that the programs do not get overlap.
- In **reloadable loader** allocation is done by the loader hence the assembler must supply the loader the size of the program.

Linking:

- It links two or more object codes and provides the information needed to allow references between them.
- It resolves the symbolic references (code/data) between the object modules by assigning all the user subroutine and library subroutine addresses. This activity is called linking.
- In **absolute loader** linking is done by the programmer as the programmer is aware about the runtime address of the symbols.
- In **relocatable loader**, linking is done by the loader and hence the assembler must supply to the loader, the locations at which the loading is to be done.

Relocation:

- It modifies the object program by changing the certain instructions so that it can be loaded at different address from location originally specified.
- There are some **address dependent locations** in the program, such address constants must be adjusted according to allocated space, such activity done by loader is called relocation.
- In **absolute loader** relocation is done by the assembler as the assembler is aware of the starting address of the program.
- In **relocatable loader**, relocation is done by the loader and hence assembler must supply to the loader the location at which relocation is to be done.