UNIT 3 LINKER AND LOADER

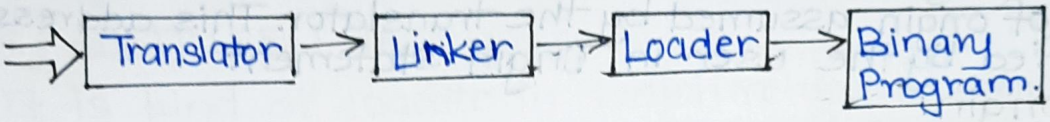
Execution of any program involves following steps :

Translation : of the program.

Linking : of the program with other program needed for its execution.

Relocation : of the program to execute from the specific memory area allocated to it.

Loading : of the program in the memory for the purpose of execution.



LOADER & LINKER

The loader is a program which accepts object codes and prepares them for execution and initiates execution.

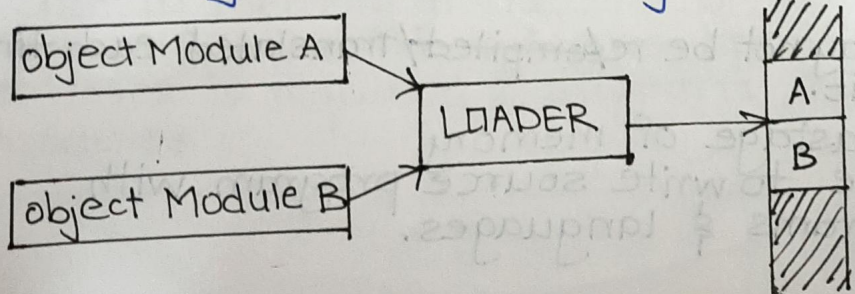
The loader is responsible for the activities such as allocation, linking, loading, relocation.

It allows allocation of the space for programs in the memory, by calculating the size of program. The activity is called allocation.

It resolvs the symbolic references (code/data) between the object modules by assigning all the user subroutines and library subroutines addresses. This activity is called linking

There are some address dependant locations in the program, such address constant must be adjusted according to allocated spaces, such activity done by loader is called relocation.

Finally it places all the machine Instruction and data of corresponding programs and subroutines into memory Thus program now become ready for execution. This activity is called loading.



The following terminology is used to refer to the address to the address of the program entity at different times

* Translation time address : address assigned by the translator.
* Linked address : address by the linker.
* Load time address: address assigned by the Loader.

The same prefixes above are used with the origin and execution start address of a program thus :-

Translated origin :
Address of origin assumed by the translator. This address is specified by the user in Origin statement

Linked origin:
Address of origin created by Linker while producing binary program.

Load origin:
Address of origin assigned by Loader while loading program for execution.

TYPES OF LOADER

1. General Loader
2. Absolute Loader
3. Subroutine Linkage
4. Relocating Loaders.
   - 4.1> Relocation bit
   - 4.2> Relocation map.
5. Direct Linking Loader.

1] GENERAL LOADER:

- In this scheme, the source program is converted to object program by translator.
- The loader accepts these modules and puts machine instruction and data in an executable form at their assigned memory.

Advantage:

- The program may not be ~~re-compited~~ translated each time while running it.
- There is no wastage of memory.
- It is possible to write source program with multiple programs & languages.

2] ASSEMBLE & GO LOADER

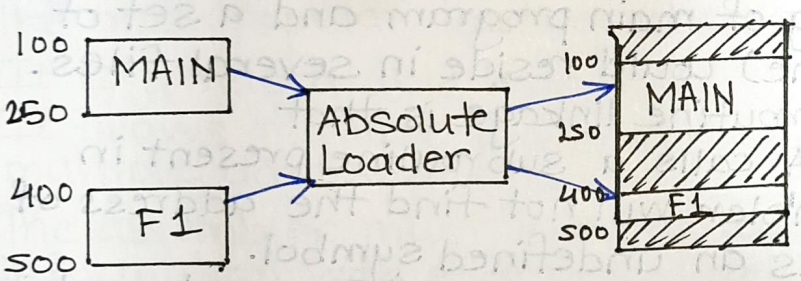
- The object code is stored in Memory after Assembly
- Single Jump Instruction.

Advantages:
Simple, developing environment.

Disadvantage:
- Whereever the Assembler Program is to be executed. it has to be assembled again.
- Programs have to be coded in the same language.

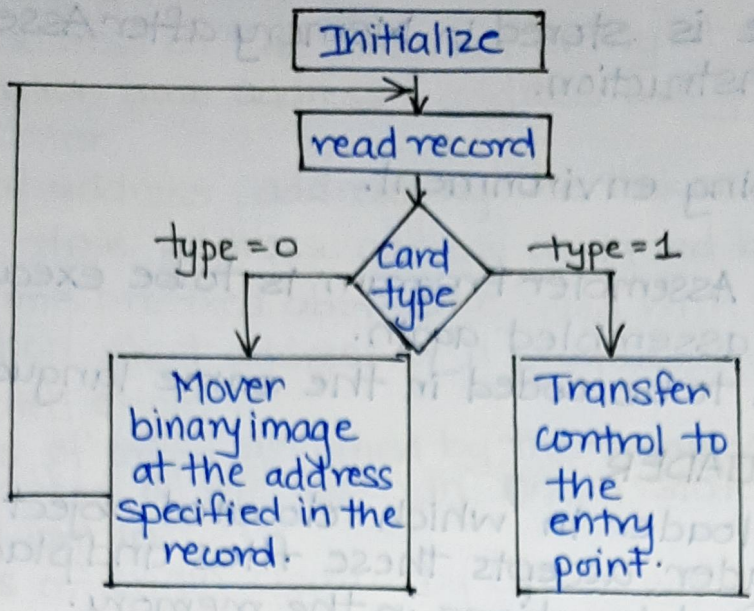
3] ABSOLUTE LOADER

- It is kind of loader in which relocated object file are created loader accepts these files and places them at specified locations in the memory.
- This type of loader is called absolute because no relocation information is needed; rather it is obtained from programmer or assembler.
- The task is virtually trivial. This simply accepts the Machine language code produced by an assembler and places it into main memory at the location specified by the assembler.



Advantages:
- Simple & Efficient
- No relocation info required. so size is comparatively small
- It make memory available for loading, since assembler is not in memory at the time of loading.
- No modification of address specific/sensitive entities is required & supports multiple object modules.

Algorithm / Flow chart.



Types of record :

TEXT RECORD :

Record type	No. of bytes of info	Memory address	Binary image of data or inst

TRANSFER RECORD :

Record type	No of bytes of info	Address of entry points.

SUBROUTINE LINKAGE

- A program consisting of main program and a set of function (sub routine) could reside in several files.
- The problem of subroutine linkage is that when a main file A calls a subroutine present in file B. The assembler will not find the address of B and declare it as an undefined symbol.
- To realize this interaction. A and B must contain public definition and external references.

EXTERN STATEMENT :

The Extern statement lists the symbols to which external references are made in the current program unit. These symbols are defined in other program units.

ENTRY STATEMENT :

The Entry statement lists the public definitions of a program unit i.e, it lists those symbols defined in the program unit which may be referenced in other program units.

RELOCATING LOADERS

When a single subroutine is changed then all the subroutines need to be reassembled. The task of allocation and linking must be carried out once again. To avoid this rework a new class of loaders is introduced which is called as relocating of loaders

Two methods for specifying relocation.

- Modifying record.
- Relocation bit.

- Each instruction is associated with one relocation bit
- These relocation bits in a text record is gathered into bit mask.
- An absolute loader loads a program at specific memory location sharing different resources.
- This would be impractical.
- Thus a loader able to load program into memory wherever there is a room is called a relocation Loaders.

DIRECT LINKING LOADER

It is a general relocated loader and is perhaps the most popular loading scheme presently used.

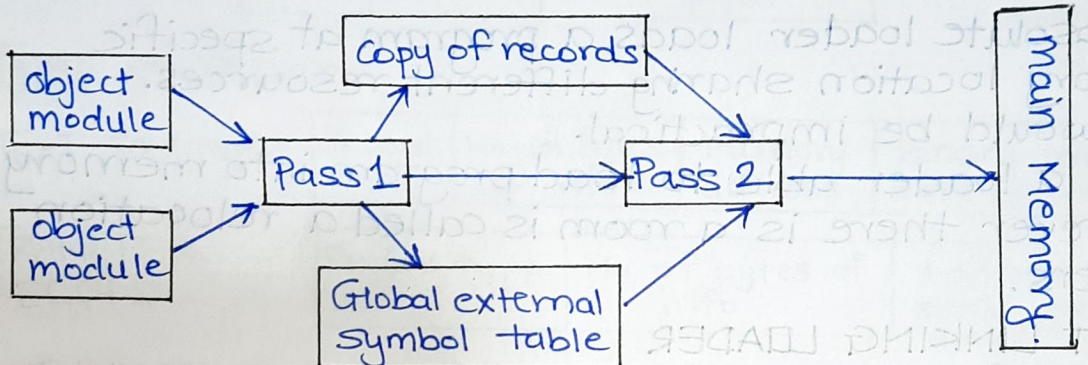
- It is a relocatable loader.
- It allows multiple procedure segments and multiple data segments.

The assembler must give the loader the following information with each procedure

1) length of segment
2) A list of symbol that may be used in other segments
3) A list of symbols not in current segment but referenced externally.
4) Information about address constraint.
5) The machine code translation of source program. & relative assigned address.

Each object module generated by the assembler is divided into four sections:

External Symbol Directory (ESD)	Actual Assembled program (TXT)	Relocation Directory (RLD)	End of obj module. (END)
SD: Symbol definition LD: can be referenced by other program ER: External reference symbol.	Text portion of obj module which contain relocatable machine language instruction and data that were produced.	1) Addr of each operand 2) By what it is to be changed 3) The operation to be performed	It indicates the end of the object module.



	STATIC	DYNAMIC
Definition.	The process of combining all necessary library routines and external references into a single executable file at compile time.	The process of loading linking external libraries and references at runtime. when the programs is loaded or executed.
Linking	at compile time	at run time
File size	greater as all lib and references in one file.	smaller as libraries and references are linked at run time
Flexibility	less flexible	more flexible
Performance	Faster program execution	Slower execution
Example	File extentions like .lib, .exe, .a, .elf, etc	File extentions like .dll, .so, .dylib, etc