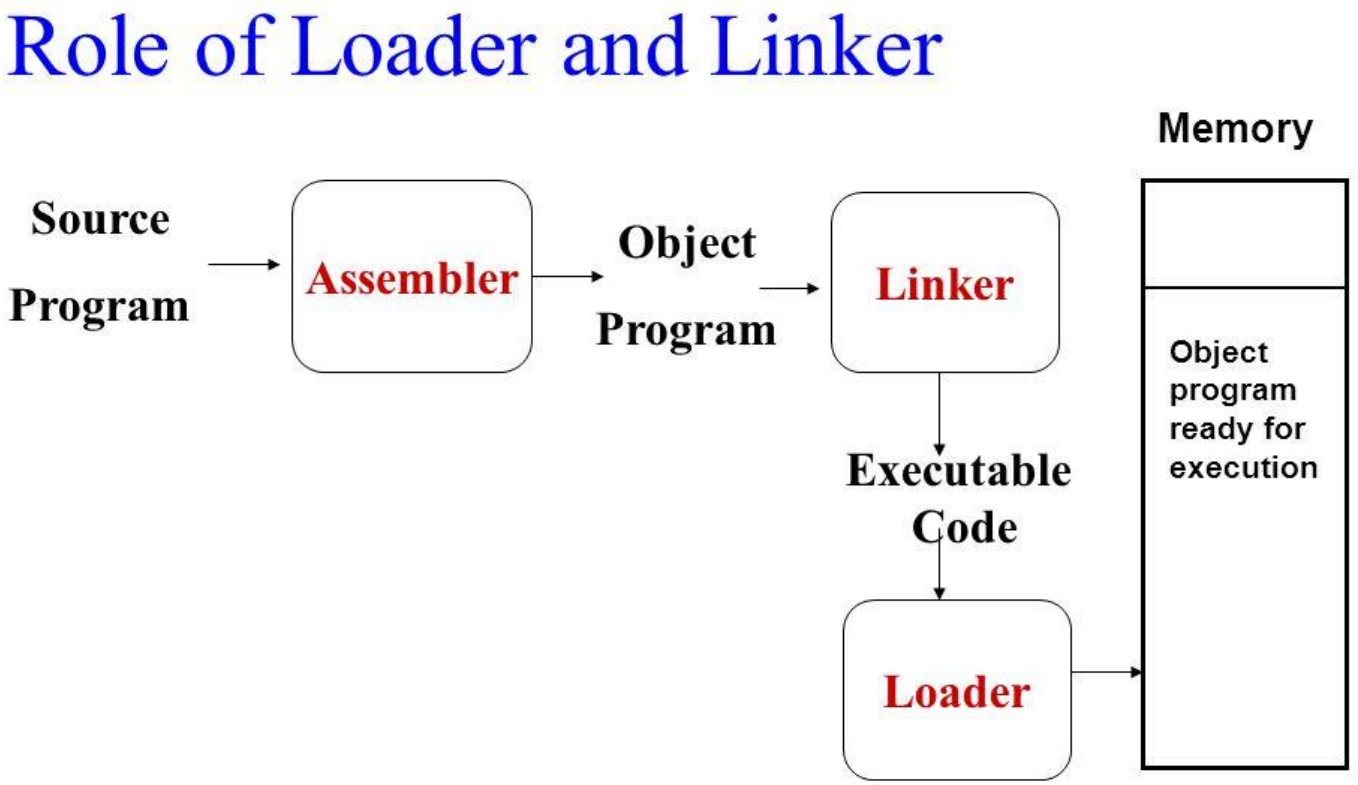
INTRODUCTION TO LOADERS

Loader definition

A loader is a system software program that performs loading functions.

It brings object programs into memory and start its execution



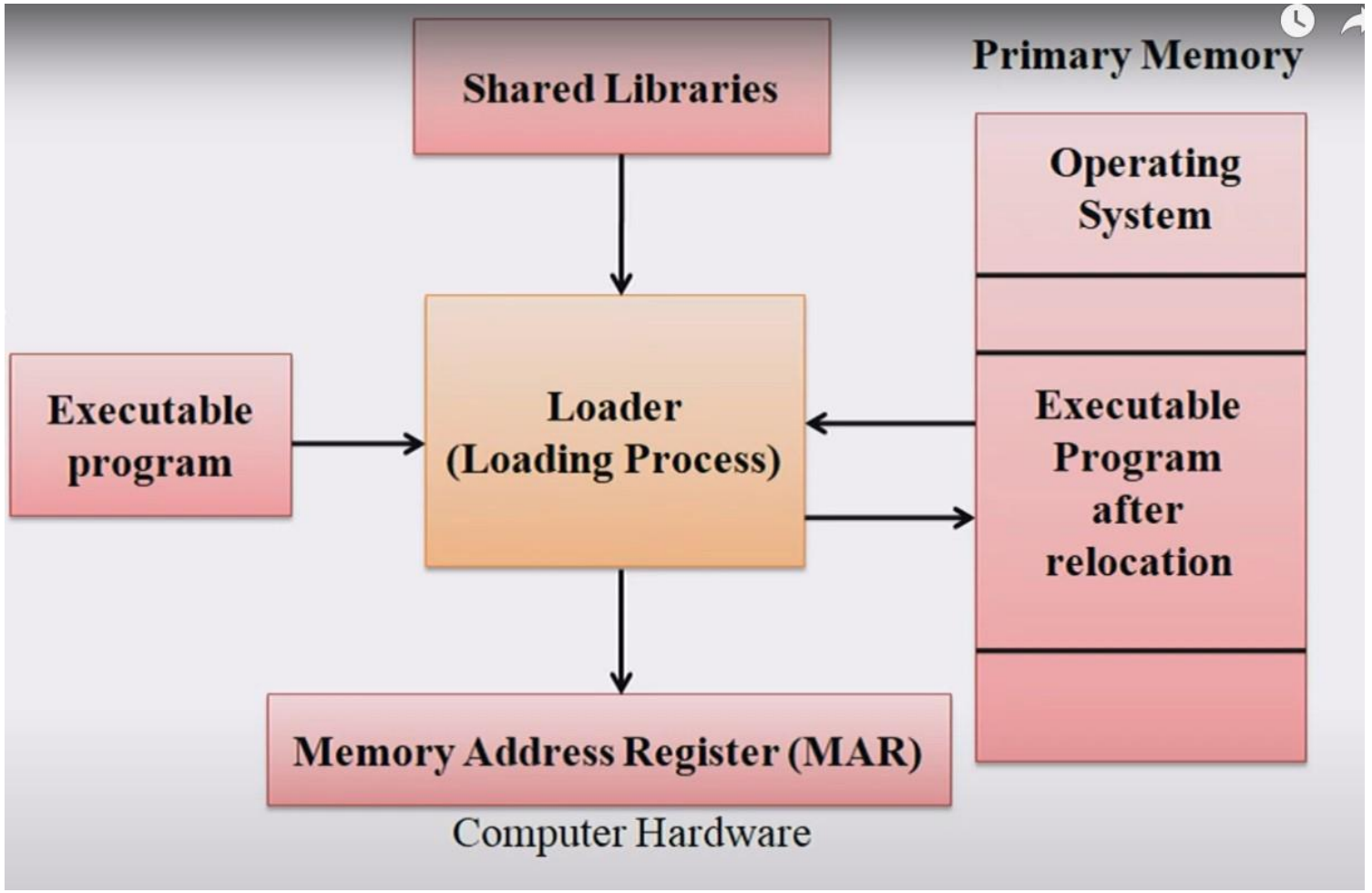
Basic Loading Principles

Loading Operations

- Loading the program uses program load address (PLA) as pointer.
- Stores the content of the machine operation code in memory at address specified by PLA.

- There are two basic principles of loading operations.
  - Make the program ready for execution.
  - Start execution

- Loader reads size of executable program
- Requests to memory manager of OS for requisite space to load the executable program in the primary memory
- It loads the executable program into primary memory
- Loader sends the starting address the Memory address register or Program Counter so computer can start execution of loaded program.



Functions of Loaders

- Allocation:
  - Allocates the space for program in the memory, by calculating the size of the program.
- Linking:
  - Resolves the symbolic references (code/data) between the object modules by assigning all the user subroutines and library subroutines addresses.
- Relocation:
  - There are some address dependent locations in the program, such address constants must be adjusted according to the allocated spaces.

Functions of Loaders

- Loading:
  - Places all the machine instructions and data of corresponding program and subroutines into the memory.
  - Program becomes ready for execution.

Loader procedure

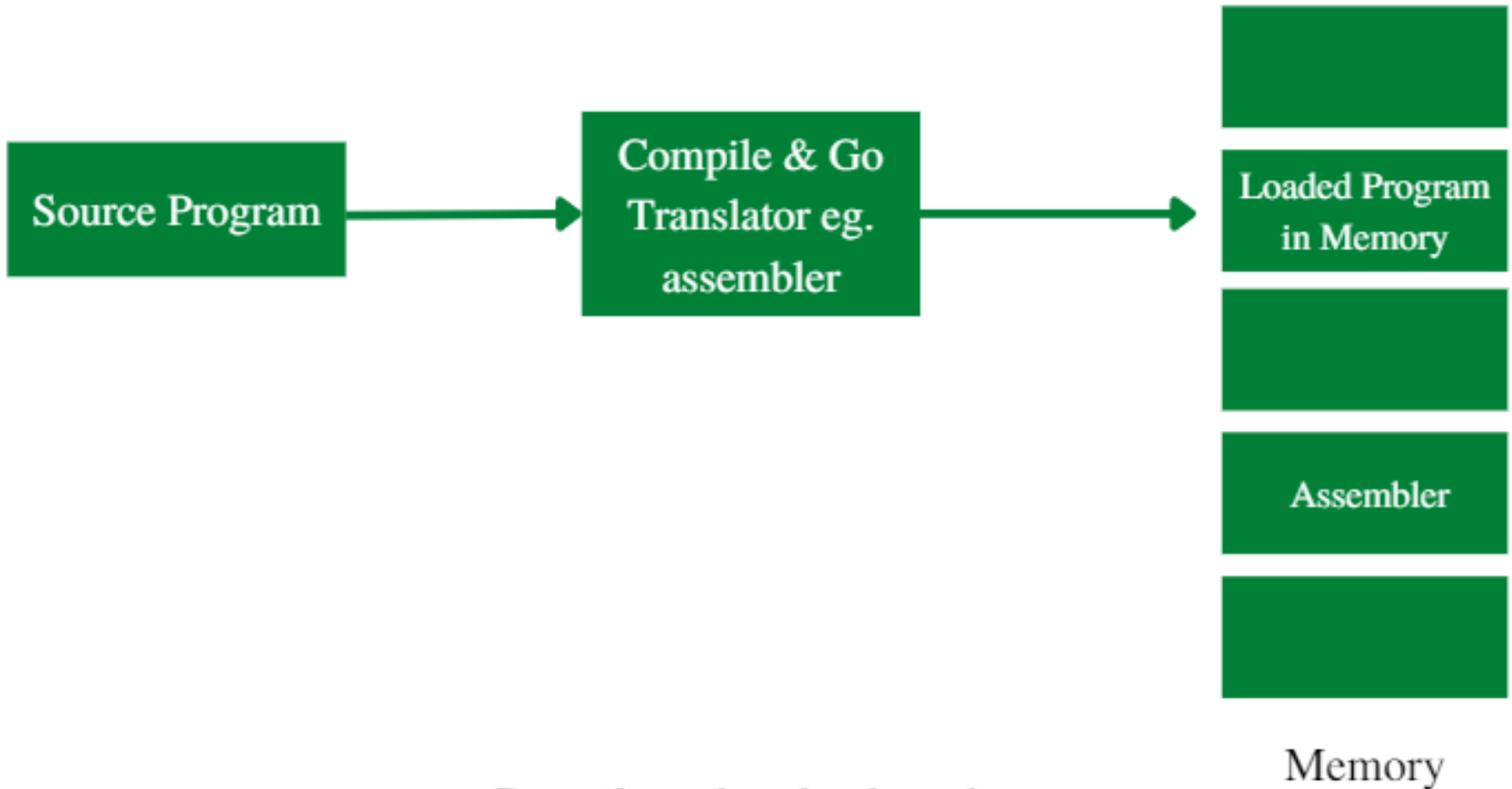
- Step 1: Reads the header of the executable file to find out the size of the text and data.
- Step 2: Creates an address space which is large enough.
- Step 3: Copies the instructions and data into primary memory.
- Step 4: Copies parameters to the main program onto the stack.
- Step 5: Initialize the machine registers and sets the stack pointer.
- Step 6: Jumps to a start-up procedure that copies the parameters into the argument registers and calls the main procedure.

Assemble and Go Loader

- One of the simplest loaders.
- The assembler itself does the process of assembling and also places the assembled instructions in the designated memory locations.
- In this type of Loader,
  - The instruction is read line by line, its machine code is obtained.
  - 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

Assemble and Go Loader

- After completion of the assembly/ translation process, the assembler passes the starting address of the program to the computer hardware (e.g. Memory address register or program counter) i.e., the first instruction is sent to the hardware for execution.
- The typical example is WATFOR-77, a FORTRAN compiler which uses “load and go” scheme.



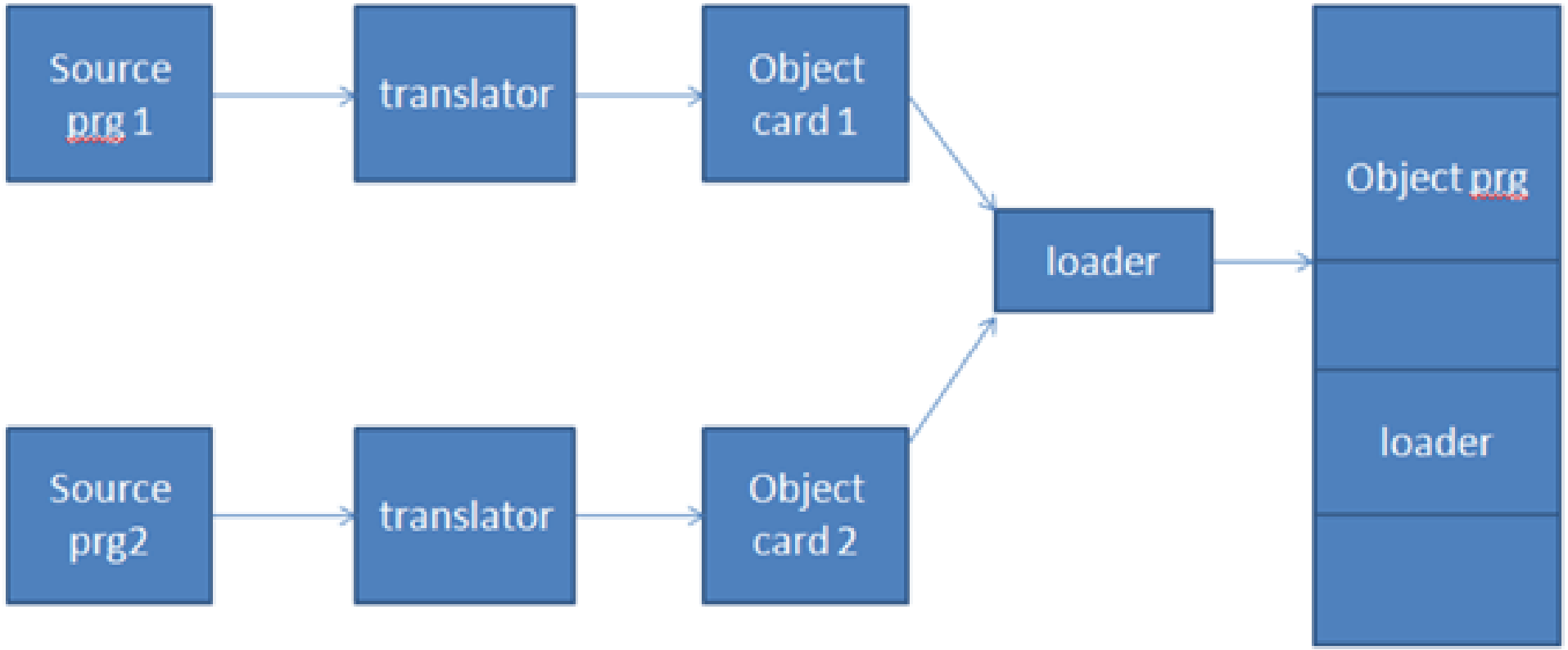
Compile and go loader scheme

Assemble and Go Loader

- Advantages:
  - Simple and easy to implement.
  - No additional routines are required to load the assembled code into the memory.
- Disadvantages:
  - Wastage of memory space due to presence of an assembler.
  - For every execution, the translator translates the source program into executable program.
  - Difficult to handle multiple source program segments in different languages.
  - Execution time required is more

General Loader Scheme

- Source program is converted to object program by some translator.
- The loader accepts these object modules and puts the machine instructions and data in an executable form at their assigned memory.
- The loader occupies some portion of the main memory.


Source prg 1
translator
Object card 1
Source prg2
translator
Object card 2
loader
Object prg
loader

General Loader Scheme

- Advantages:
  - Program need not be retranslated each time while running it
  - No wastage of memory because the assembler is not placed in the memory instead of it, loader occupies some portion of the memory.
  - Possible to write source program with multiple programs and multiple languages.
- Disadvantages:
  - Loader cannot handle different object modulus obtained from different kinds of computer hardwares.

Absolute Loader

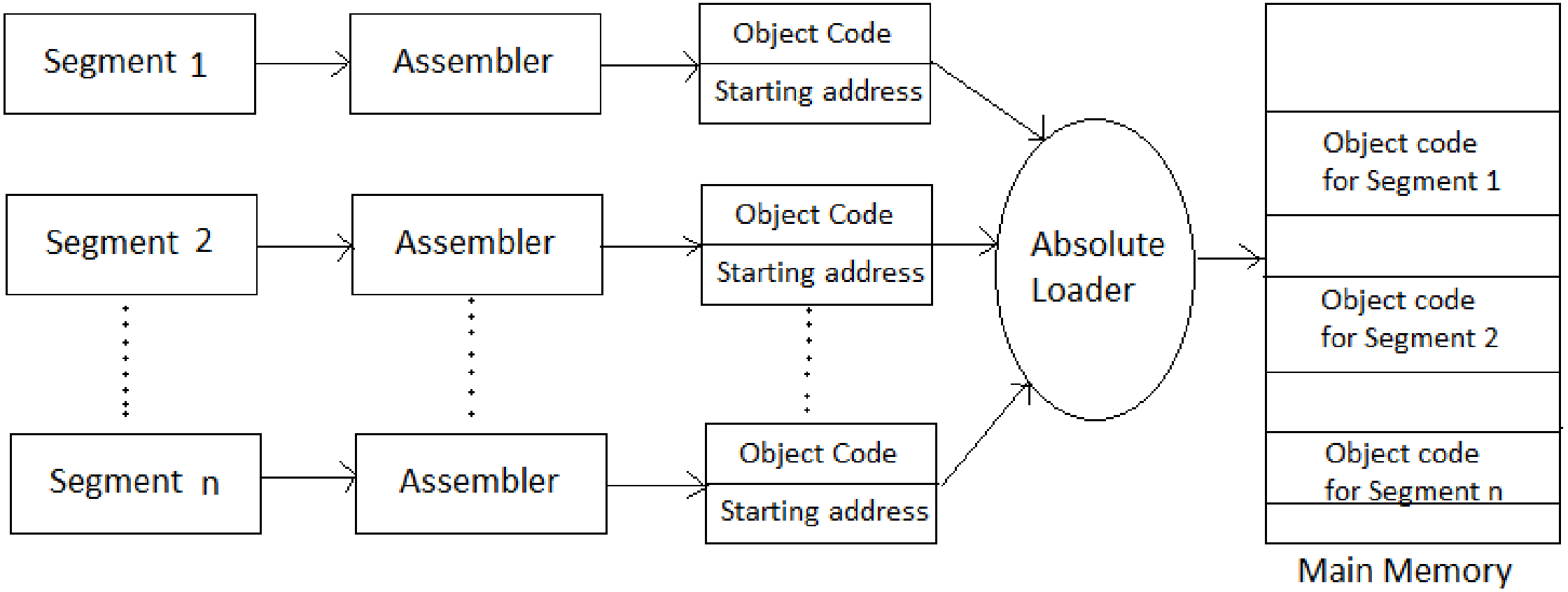
- Relocated object files are created.
- Loader accepts these files and places them at a specified location in the memory.
- Called absolute loader because no relocating information is needed, rather obtained from the programmer or the assembler.
- The translator outputs the machine language.
- It is very similar to assemble and go loader.
- Absolute loader accepts this translated output in the form of records

Absolute Loader

- Records containing
  - The text (instructions, data or both) to be loaded.
  - The length of text
  - The starting address

Absolute Loader

- The task of the absolute loader is to place these records into the primary memory at the location specified by assembler during process of assembly.
- The resolution of external references or linking of different subroutines are the issues which need to be handled by the programmer.
- The programmer should have knowledge of memory management.
- The programmer should take care of:
  - Specification of starting address of each module to be used.
  - While branching from one segment to another, the absolute starting address of respective module is to be known by the programmer.


Segment 1
Assembler
Object Code
Starting address
Segment 2
Assembler
Object Code
Starting address
Segment n
Assembler
Object Code
Starting address
Absolute Loader
Object code for Segment 1
Object code for Segment 2
Object code for Segment n
Main Memory

Absolute Loader

- Advantages:
  - Simple to implement
  - Task of loader becomes simpler
  - Allows multiple programs or the source programs written in different languages.
  - Process of execution is efficient.
- Disadvantages:
  - Programmer must specify the loading address in the assembly program
  - The programmer must remember the addresses of each subroutine.

Relocating Loader

- To avoid possible reassembling of all subroutines when a single subroutine is changed and to perform the task of allocation and linking for the programmer, the relocating loaders are used.
- The execution of the object program is done using any part of the available and sufficient memory.
- The object program is loaded into memory wherever there is room for it.
- The assembler assembles each procedure segment independently and passes to loader the text and information as to relocation and intersegment references, also the length of program.

Relocating Loader

- Transfer Vector
  - It is used for linking
  - It is the beginning part of the program
  - It stores the names of the external subrotines
- Initially when we call the subroutine, it transfers to the corresponding location of transfer vector.
- Finally loader calculates absolute location (exact position).
- It is used on computers with a fixed length direct address instruction format.

Relocation Loader

Relocation bit:

- It is used for relocation, using a bit that is present in object code.
- This bit is associated with each instruction.
- If its 1, then relocation is done, else not done.

Relocating Loader

- Advantages:
  - Avoids possible reassembling of all subroutines when a singe subroutine is changed.
  - Perform the tasks of allocation and linking for the programmer.
- Disadvantages:
  - Difficult to implement.
  - Slower than absolute loaders.

Direct Linking Loader

- Direct Linking Loader is the most common type of loader
- It is a relocateable loader.
- It doesn’t have direct access to the source code.
- To place the object code in memory, there are two situations:
  - Either the address of object code could be Absolute which can be directly placed in memory
  - Address is relative, then the assembler informs the loader about the relative address.

Direct Linking Loader

- The assembler should give the following information to the loader:
  - The length of the program
  - A list of external symbols
  - Information about address constant
  - Machine code translation of the source program

Direct Linking Loader

- Advantages:
  - More flexible
- Disadvantages:
  - Necessary to allocate, relocate, link and load all subroutines each time in order to execute a program.
  - Even though loader program is smaller than the assembler, it absorbs considerable amount of memory.