FUNDAMENTAL PROCESS OF LOADERS

- **Allocation :** the space for program is allocated in the main memory, by calculating the size of the program.
- **Loading** – brings the object program into memory for execution.
- **Relocation** – modifies the object program so that it can be loaded at an address different from the location originally specified.
- **Linking**, which combines two or more separate object programs and supplies the necessary information.

How does loader gets loaded ?

ANSWER: **BOOTSTRAP LOADER**

A **boot**strap loader is a computer program that loads the main operating system or runtime environment for the computer after completion of self-tests.

Types of Loaders

- Compile and go loader
- Absolute Loader
- Relocating Loader (Relative Loader)
- Direct Linking Loader

COMPILE-AND-GO LOADER

- In **compile and go loader** is a link editor/program loader in which the assembler itself places the assembled instruction directly into the designated memory locations for execution.
- The instruction are read line by line, its machine code is obtained and it is directly put in the main memory at some known address.
- After completion of assembly process, it assigns the starting address of the program to the location counter.

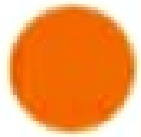
- The assembler is first executed and it, when it is finished, causes a branch straight to the first instruction of the program.
- There is no stop between the compilation, link editing, loading, and execution of the program.
- It is also called an **assemble-and-go** or a **load-and-go** system.

ADVANTAGES OF COMPILE-AND-GO LOADERS

- They are simple and easier to implement.
- No additional routines are required to load the compiled code into the memory.

DISADVANTAGES OF COMPILE-AND-GO LOADERS

- There is wastage in memory space due to the presence of the assembler.
- 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.



General Loader Scheme

- In "Compile-and-Go" the outputting instruction and data are assembled. In which assembler is placed in main memory that results in wastage of memory.
- To overcome that we requires the addition of the new program of the system, a loader.
- Generally the size of loader is less than that of assembler.

- The loader accepts the assembled machine instructions, data and other information present in the object format and places machine instructions and data in core in an executable computer form.

- The reassembly is no longer necessary to run the program at a later date.

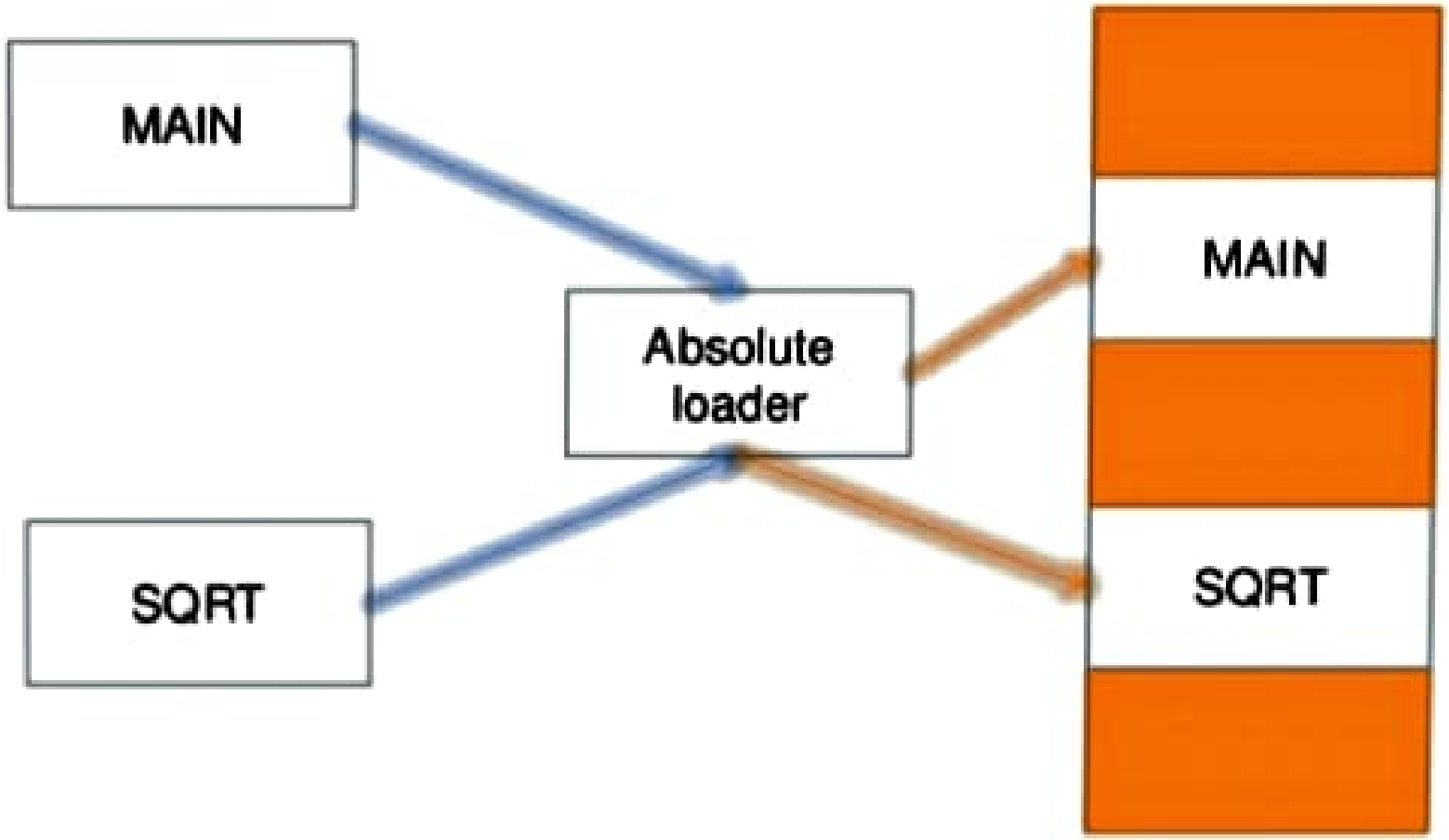
Advantage of General Loader Scheme

- In this scheme the source program translators produce compatible object program deck formats and it is possible to write subroutines in several different languages since the object decks to be processed by the loader will all be in the same "language" that is in "machine language".

Absolute Loaders

- In this scheme the assembler outputs the machine language translation of the source program in almost the same form as in the "Compile and go" , except that the data is punched on cards. Here it will directly placed in memory .

- The loader in turn simply accepts the machine language text and places it into core at the location prescribed by the assembler.


MAIN
SQRT
Absolute loader
MAIN
SQRT

DISADVANTAGES

- The programmer must specify to the assembler the address where the program is to be loaded.
- If there are multiple subroutines , the programmer must remember the address of each.
- The MAIN program is assigned to locations 100-247 and the SQRT subroutine is assigned locations 400-477. if changes were made to MAIN that increasing length to more than 300 bytes.

the end of MAIN is overlap with the start of SQRT. It would then necessary to assign SQRT to a new location by changing its START.

There are four functions involving in the Absolute loading

- Allocation
- Linking
- Relocation
- Loading

RELOCATING LOADERS

- To avoid possible reassembling of all subroutines when a single subroutine is changed and to perform the tasks of allocation and linking for the programmer the **relocating loaders** is introduced.

- BINARY SYMBOLIC SUBROUTINE(BSS)loader such as used in the IBM 7094,IBM 1130,GE 635.

- The BSS loader allows many procedure segments but only one data (common)segment.

- The assembler assembles each procedures segment independently and passes to loader the text and information as to relocation and intersegment references.

- The output of the relocating assembler using a BSS scheme is the Object program and information about all other program its references.

- For each program the assembler outputs a text(machine translation of the program)

- Prefixed by **Transfer Vector** that consists of address containing names of the subroutines referenced by the source program.

- For ex., if a Square Root Routine(SQRT) was referenced and was the first subroutine called, the first location of transfer vector contain the symbolic name SQRT.

- The statement calling SQRT would be translated into transfer instruction.

- The assembler would also provide the loader with additional information, such as the length of the entire program and the length of the transfer vector.
- After loading the text and the transfer vector into core, the loader would load each subroutine identified in the transfer vector.
- Thus the execution of the call SORT statement would result in branch to the first location in the transfer vector.
- The BSS loader scheme is often used on computers with a fixed-length direct address instruction format.

- For ex, if format of 360 RX instruction were:

- A2-16 bit direct address
- Works for 2^{16} memory
- If more than that then problem occurs that time the assembler associates a bit with instruction or address field that called "relocation bit."
- If it is one , 16 bit address is relocated .
- If not, then it is not relocated.

Source program

```
Main  START
      EXTRN SQRT
      EXTRN ERR
      ST          14,SAVE   save
          return address
      L           1,F'9'    load test
  value
      BAL         14,SQRT
  call sort
       C          1,F'3'
      BNE         ERR
      L            14,SAVE
      BR          14
SAVE  DS          F
          temp.loc.
          END
```

Rel.addr.	relocation	Rel.addr.
0	00	'SORT'
4	00	'ERRb'
8	01	ST 14,36
12	01	L 1,40
16	01	BAL 14,0
20	01	C 1,44
24	01	BC 7,4
28	01	L 14,36
32	0	BCR 15,14
34	0(skipped for alignment)	
36	00(Temp location)	
40	00	9
44	00	3