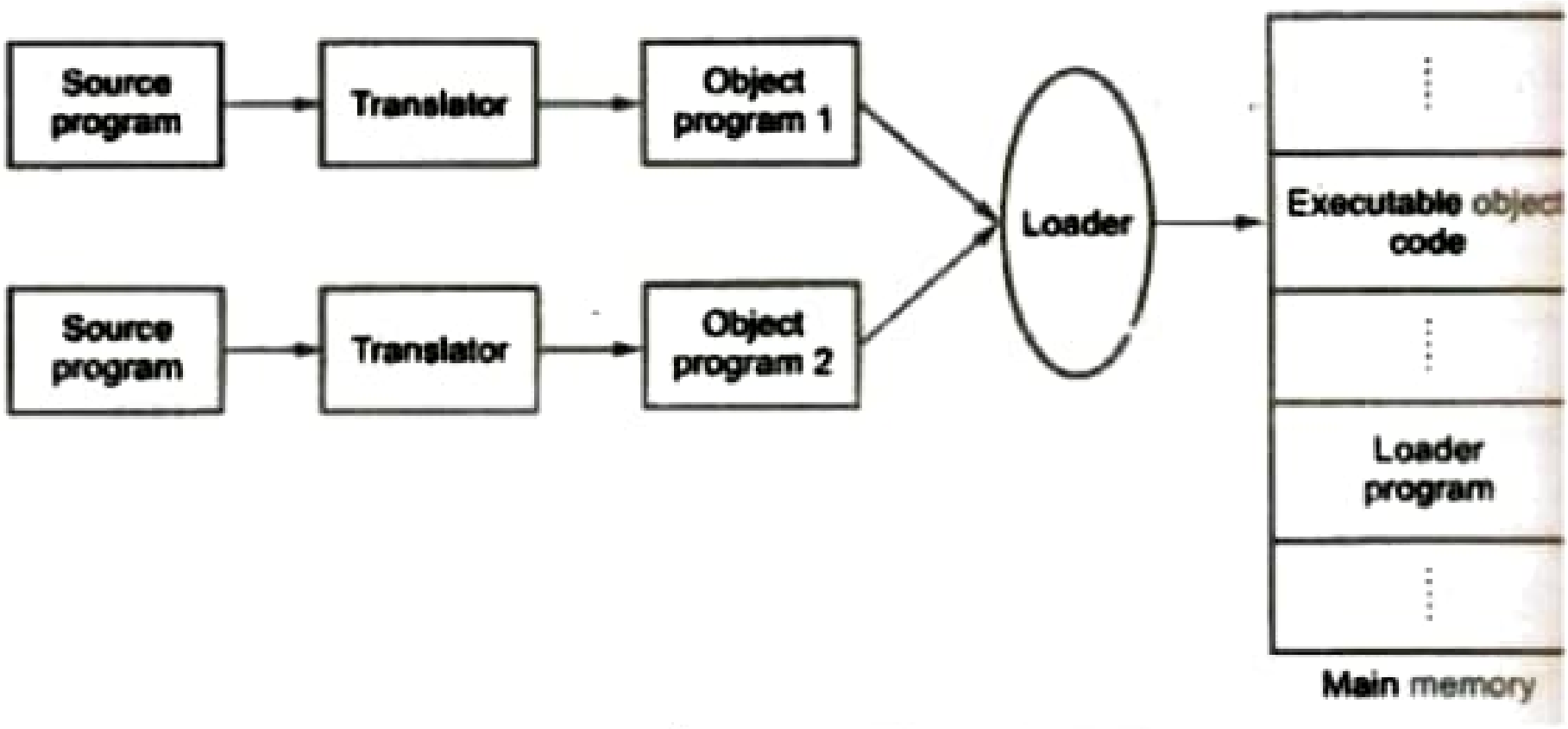
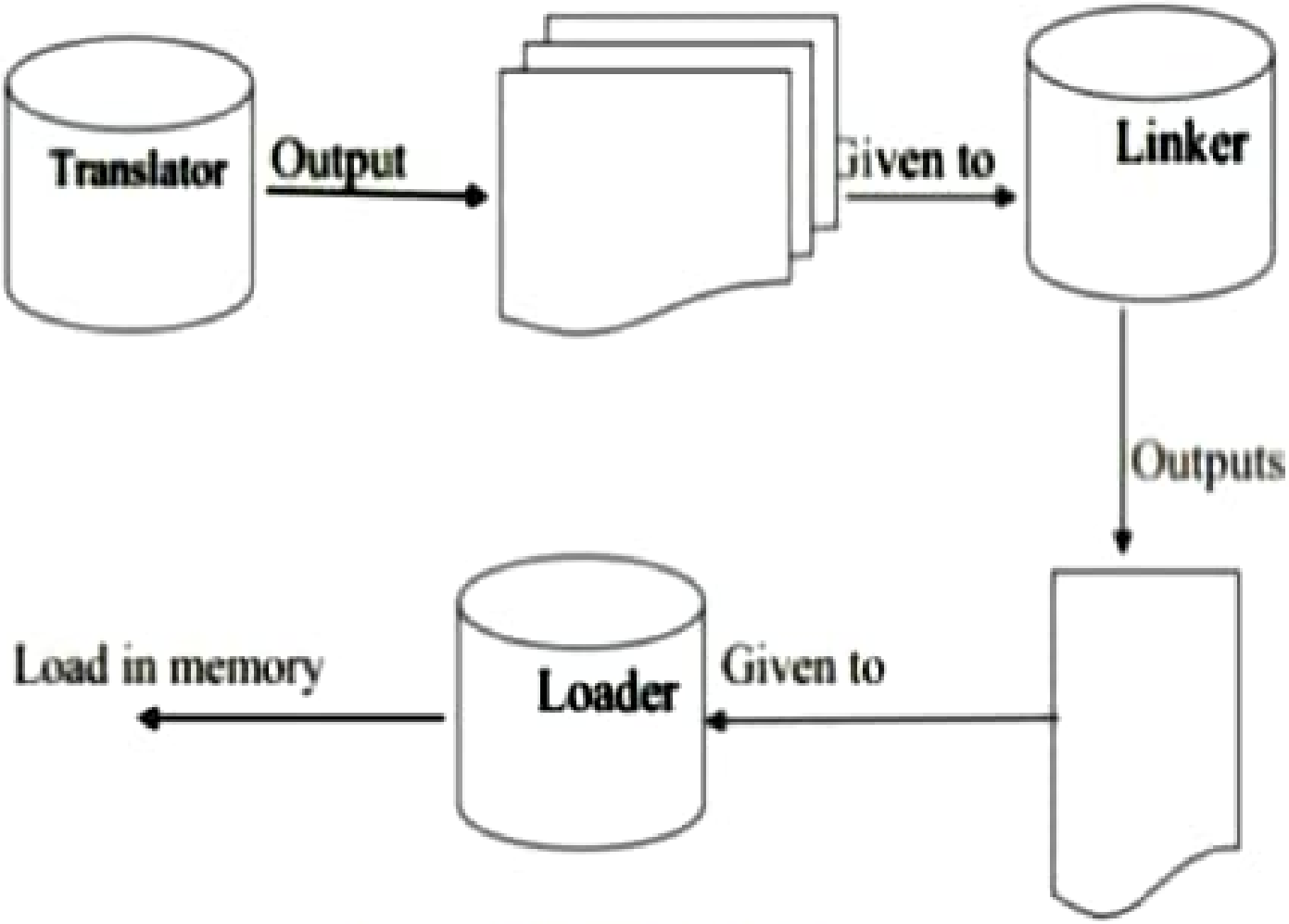
DEFINE LOADER:-

Loader is utility program which takes object code as input prepares it for execution and loads the executable code into the memory. Thus loader is actually responsible for initiating the execution process.



General loader scheme



Process of linking a program

DIRECT-LINKING LOADERS

- A direct-linking loader is a relocatable loader.
- It has advantage of allowing programmer multiple procedure segments and multiple data segments.
- Complete freedom in referencing data or instructions contained in other segments, provides flexible intersegment referencing.

- The assembler should give the following information to the loader:

1. The length of the object code segment.
2. A list of external symbols (could be used by other segments).
3. List of External symbols(The segment is using).
4. Information about address constants.
5. Machine code translation of the source program.

EXAMPLE

Card no.	Program			Rel. loc.	Translation	
1.	JOHN	START				
2.		ENTRY	RESULT			
3.		EXTRN	SUM			
4.		BALR	12,0	0	BALR	12,0
5.		USING	*,12			
6.		ST	14,SAVE	2	ST	14,54(0,12)
7.		L	1,POINTER	6	L	1,48(0,12)
8.		L	15,ASUM	10	L	15,58(0,12)
9.		BALR	14,15	14	BALR	14,15
10		ST	1,RESULT	16	ST	1,50(0,12)
11.		L	14,SAVE	20	L	14,54(0,12)
12.		BR	14	24	BCR	15,14
				26	—	
13.	TABLE	DC	F'1,7,9,10,3'	28	1	
				32	7	
				36	9	
				40	10	
				44	3	
14.	POINTER	DC	A(TABLE)	48	28	
15.	RESULT	DS	F	52	—	
16.	SAVE	DS	F	56	—	
17.	ASUM	DC	A(SUM)	60	?	
18.		END		64		

FIGURE 6.7 Assembly source program and its translation

- The list of symbols not defined in the current segment but used in the current segment are stored in a data structure called USE table.

- The lists of symbols defined in the current segment and referred by the other segments are stored in a data structure called DEFINITION table.

- To place the object code in the memory there are two situations:
    1. Address of the object code could be absolute.
    2. The address of object code can be relative.

- The assembler generates following types of cards:
  - ESD
  - TXT
  - RLD
  - END

1. **ESD** -

   External symbol dictionary contains information about all symbols that are defined in the program but referenced somewhere. It contains

   - Reference no
   - Symbol Name
   - TYPE
   - Relative Location
   - Length

ESD cards

Reference no.	Symbol	Type	Relative location	Length
1	JOHN	SD	0	64
2	RESULT	LD	52	—
3	SUM	ER	—	—

- TYPE:
  - SD - Segment Definition.
  - LD - Local Definition.
  - ER - External Reference.

2. **TXT** –
   Text card contains actual object code.(translated source code).

3. **RLD** –
   Relocation and linkage directory contains information about locations in the program whose content depend on the address at which program is placed.

TXT cards

Reference no.	*Relative location*	*Object code*	
4	0	BALR	12,0
6	2	ST	14,54(0,12)
7	6	L	1,46(0,12)
8	10	L	15,58(0,12)
9	14	BALR	14,15
10	16	ST	1,50(0,12)
11	20	L	14,54(0,12)
12	24	BCR	15,14
13	28	1	
13	32	7	
13	36	9	
13	40	10	
13	44	3	
14	48	28	
17	60	0	

RLD cards

Reference no.	Symbol	Flag	Length	Relative location
14	JOHN	+	4	48
17	SUM	+	4	60

- The RLD cards contains information:
  - Location of the constant that need relocation
  - By what it has to be changed
  - The operation to be performed

- The Format of RLD
  - Reference No
  - Symbol
  - Flag
  - Length
  - Relative Location

4. **END** –

   Indicates the end of the program and specifies starting address for execution

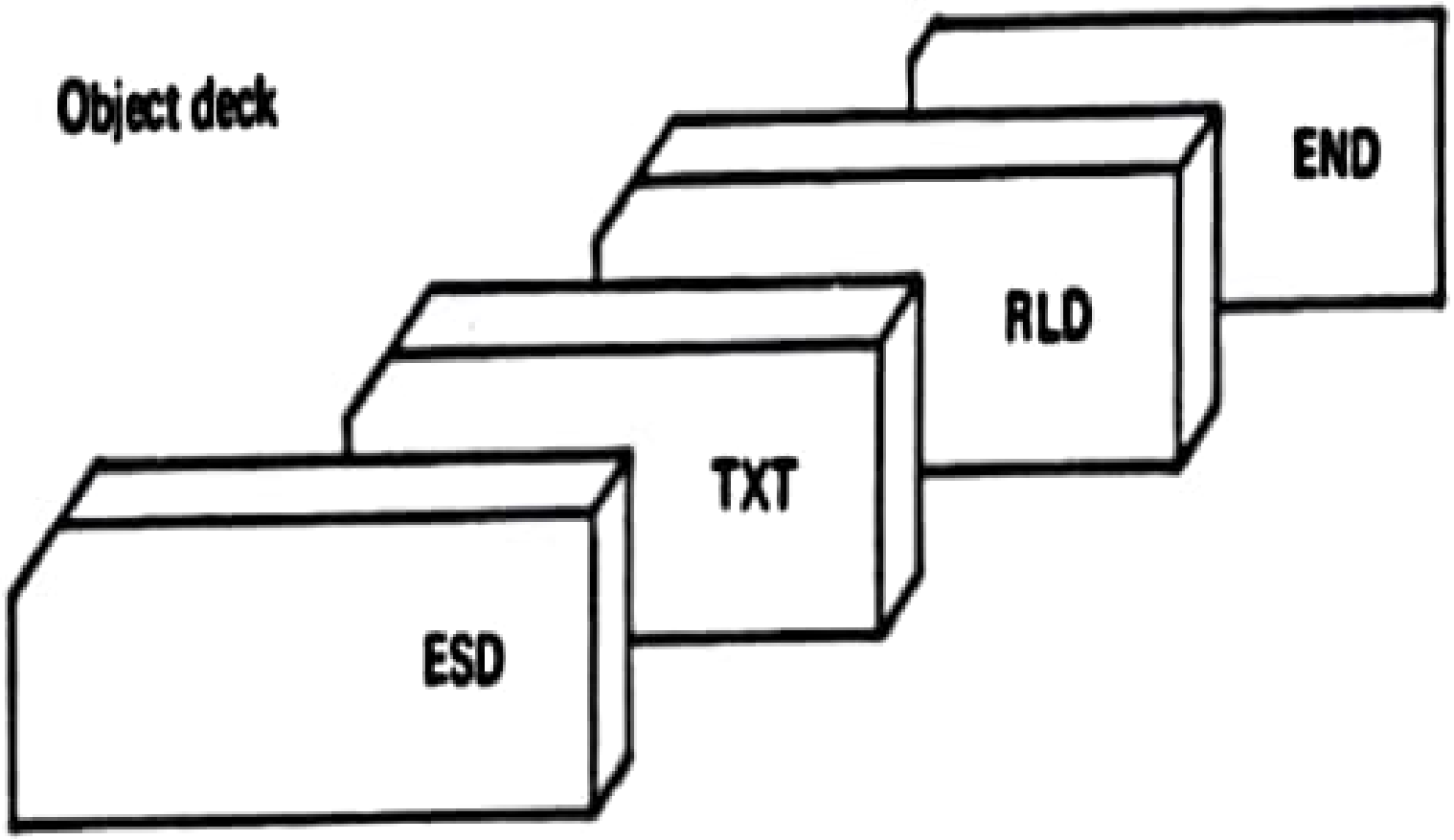


FIGURE 5.8 Example object deck for a direct-linking loader

Disadvantages of Direct Linking

- It is necessary to allocate, relocate, link, and load all of the subroutines each time in order to execute a program
  - loading process can be extremely time consuming.
- Though smaller than the assembler, the loader absorbs a considerable amount of space
  - Dividing the loading process into two separate programs a binder and a module loader can solve these problems.