KCG College of Technology,Chennai-96
Computer Science and Engineering

CS 2304 SYSTEM SOFTWARE **III Sem CSE**

QUESTION BANK - UNIT-III LOADERS & LINKERS

Definitions

Loading

– Copies a object program into memory for execution.

Relocation

– Modifies an program so it can be loaded at a location different from the one specified at assembly or compilation.

Linking

– Combines two or more object files and the references between them.

Loader

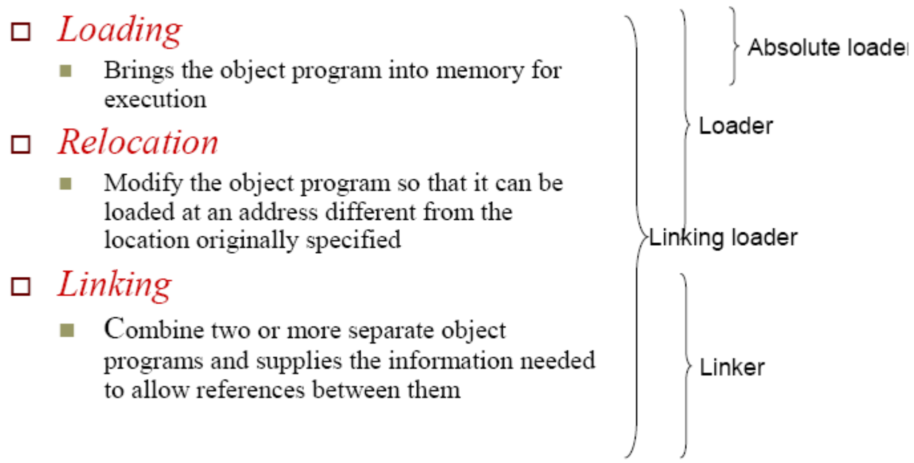
– System program that performs the loading function.

Linker or Linkage Editor

– Performs the linking function.

- Compilers & assemblers for a given machine produce files in the same format.

1) Define Loading,Relocation and Linking.



- Loading
  - Brings the object program into memory for execution
- Relocation
  - Modify the object program so that it can be loaded at an address different from the location originally specified
- Linking
  - Combine two or more separate object programs and supplies the information needed to allow references between them

2) What is the basic function of a loader?

The most fundamental functions of a loader:

Bringing an object program into memory and starting its execution

3) What are the characteristics of an absolute loader?

- No linking and relocation needed
- **Records in object program perform**
- **Header record**
  - Check the Header record for program name, starting address, and length (available memory)

- **Text record**
  - Bring the object program contained in the Text record to the indicated address
- **End record**
  - Transfer control to the address specified in the End record

4) Write an algorithm for an absolute loader.
Algorithm for an absolute loader

```
begin
    read Header record
    verify program name and length
    read first Text record
    while record type ≠ '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
```

E.g., convert the pair of characters "14" (two bytes) in the object program to a single byte with hexadecimal value 14

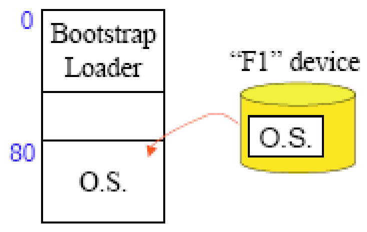
Figure 3.2 Algorithm for an absolute loader.

5) What is a bootstrap loader?

When a computer is first tuned on or restarted, a special type of absolute loader, the bootstrap loader loads the first program (usually O.S.) to be run into memory

6) Explain SIC bootstrap loader.

- ➢ The bootstrap itself begins at address 0
- ➢ It loads the OS starting address 0x80
- ➢ No header record or control information, the object code is consecutive bytes of memory
- ➢ After load the OS, the control is transferred to the instruction at address 80.



B o o t s t r a p   l o a d e r   f o r   S I C / X E

```
begin
X=0x80 ; the address of the next memory location to be loaded
Loop
A¬GETC ; read one char. From device F1 and convert it from the
; ASCII character code to the value of the hex digit
save the value in the high-order 4 bits of S
A¬GETC
A¬ (A+S) ; combine the value to form one byte
store the value (in A) to the address represented in register X
X¬X+1
    End
```

7) **What are the disadvantages of absolute loaders?**

Drawback of absolute loaders

- o Programmer needs to specify the actual address at
- o which it will be loaded into memory.
- o Difficult to run several programs concurrently,
- o sharing memory between them.
- o Difficult to use subroutine libraries.
- o **Solution**: a more complex loader that provides
- o *Program relocation*
    - *Program linking*

8) **What are relocatable loaders? What are different types of relocatable loaders?**

Loaders that allow for program relocation are called
relocating loaders or ***relative loaders***.

o Two methods for specifying relocation as part of the
object program

Modification records

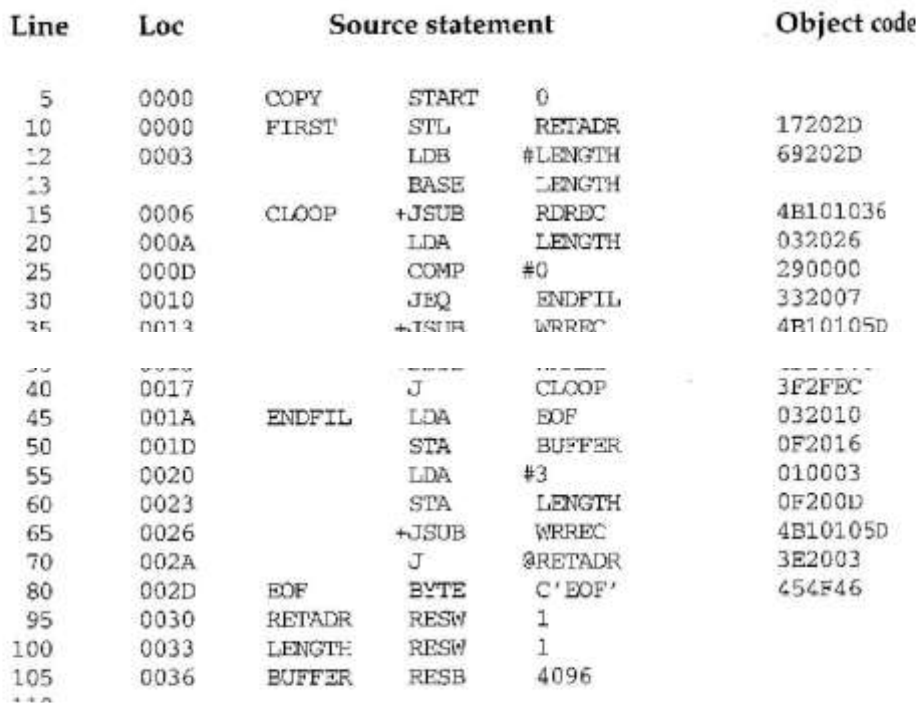
o Suitable for a ***small*** number of relocations required when relative
or immediate addressing modes are extensively used

Relocation bits

o Suitable for a ***large*** number of relocations required when only
direct addressing mode can be used in a machine with fixed
instruction format (e.g., the standard SIC machine)

9) **Explain with an example ,the relocatable loader.**

Example of a SIC/XE Program



Line	Loc	Source statement			Object code
5	0000	COPY	START	0	
10	0000	FIRST	STL	RETADR	17202D
12	0003		LDB	#LENGTH	69202D
13			BASE	LENGTH	
15	0006	CLOOP	+JSUB	RDREC	4B101036
20	000A		LDA	LENGTH	032026
25	000D		COMP	#0	290000
30	0010		JEQ	ENDFIL	332007
35	0013		+JSUB	WRREC	4B10105D
40	0017		J	CLOOP	3F2FEC
45	001A	ENDFIL	LDA	EOF	032010
50	001D		STA	BUFFER	0F2016
55	0020		LDA	#3	010003
60	0023		STA	LENGTH	0F200D
65	0026		+JSUB	WRREC	4B10105D
70	002A		J	@RETADR	3E2003
80	002D	EOF	BYTE	C'EOF'	454F46
95	0030	RETADR	RESW	1	
100	0033	LENGTH	RESW	1	
105	0036	BUFFER	RESB	4096	

Relocatable Program

o *Modification record*

Col 1 M

Col 2-7 Starting location of the address field to be modified, relative to the beginning of the program (hex)

Col 8-9 length of the address field to be modified, in half-bytes

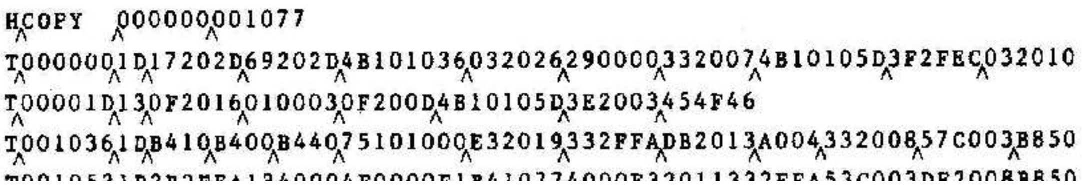
E.g M^000007^05

Pass the address – modification information to the **relocatable loader**

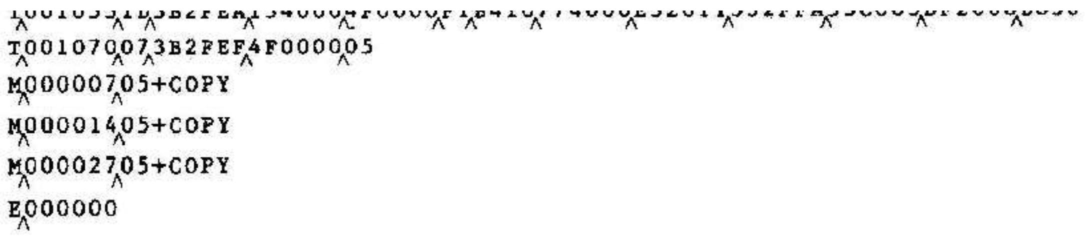
Beginning address of the program is to be added to a field that begins at addr ox000007 and is 5 bytes in length.

Object Program with Relocation by Modification Records

Chapter 3 Loaders and Linkers



```
H^COPY  ^000000^001077
T^000000^1D^17202D^69202D^4B101036^032026^290000^332007^4B10105D^3F2FEC^032010
T^00001D^13^0F2016^010003^0F200D^4B10105D^3E2003^454F46
T^001036^1D^B410^B400^B440^75101000^E32019^332FFA^DB2013^A004^332008^57C003^B850
[illegible]
```



```
T^001053^1D^3B2FEA^134000^4F0000^F1^B410^774000^E32011^332FFA^53C003^DF2008^B850
T^001070^07^3B2FEF^4F0000^05
M^000007^05+COPY
M^000014^05+COPY
M^000027^05+COPY
E^000000
```

Figure 3.5 Object program with relocation by Modification records.

10) What is linking?

Linking, which combines two or more separate object programs and supplies the information needed to allow references between them .

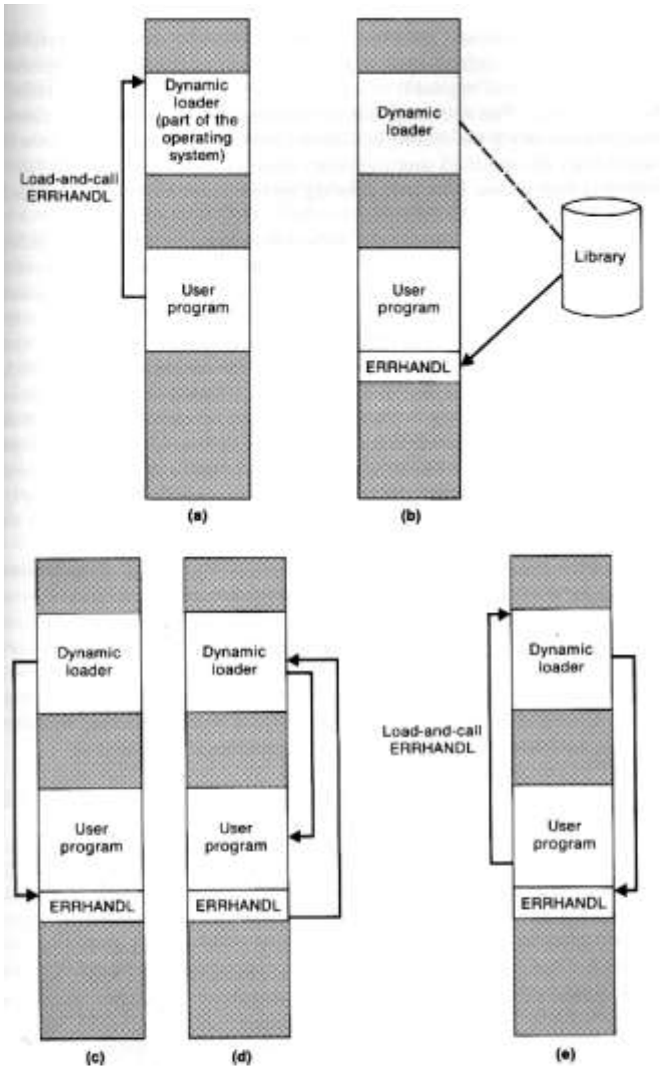
11) What are loader design options?

Loader Design Options

- *Linking loaders*
  - Perform all linking and relocation at **load time**
- *Linking editors*
  - Perform linking **before the program is loaded for execution.**
- *Dynamic linking*
  - Perform linking at **execution time**

12) What is dynamic linking? Explain with an example.

- Dynamic Linking refers to postponing the linking function until execution time.
  - A subroutine is loaded and linked to the rest of the program when it is first called (Dynamic linking, dynamic loading, or load on call)
- Allow several executing programs to share one copy of a subroutine or library
- In object-oriented system, it allows the implementation of the object and its methods to be determined at the time the program is run
- Dynamic linking provides the ability to load the routines only when they are needed



- Dynamically loaded must be called via an operating system service request
- Load-and-call service
  - OS examines its internal tables to determine whether or not the routine is already loaded
  - Routine is loaded from library
  - Control is passed from OS to the called subroutine
  - Subroutine is finished
  - Calling to a subroutine which is already in memory
- ***Dynamic Linking :*** *Binding* of the name to an actual address is delayed from load time until execution time

13) What are the advantages of dynamic linking?

Dynamic Linking

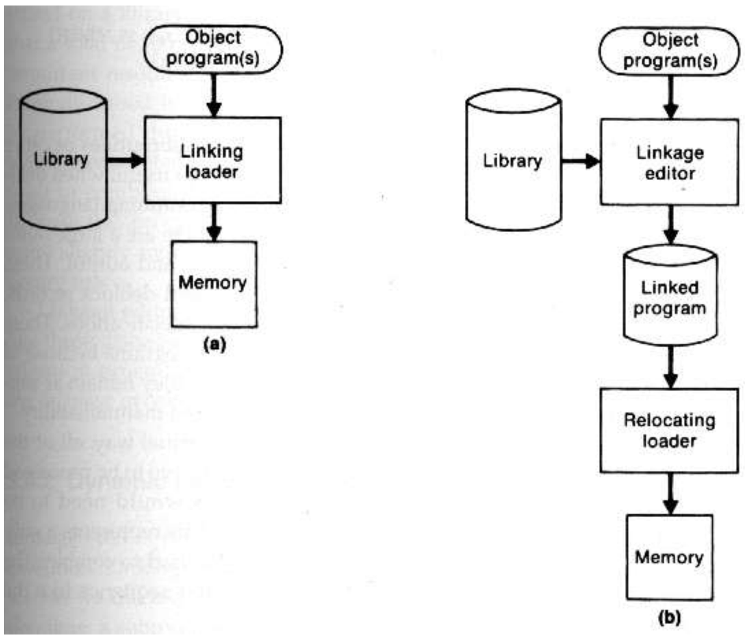
Advantages

- Load the routines when they are needed, the time
- and memory space will be saved.

- Avoid the necessity of loading the entire library for each execution
  - i.e. load the routines only when they are needed
- Allow several executing programs to share one copy of a subroutine or library (**Dynamic Link Library**, **DLL**)

11) What is the difference between linking loader and linkage editors?

- A **linking loader** performs
  - All linking and relocation operations
  - Automatic library search
  - Loads the linked program directly into memory for execution
- **A linkage editor**
  - Produces a linked version of program (often called a load module or an executable image), which is written to a file or library for later execution
  - A simple relocating loader can be used to load the linked version of program into memory
    - The loading can be accomplished in one pass with no external symbol table required



- **A linkage editor**
  - Resolution of external references and library searching are only performed once
  - In the linked version of programs
    - All external references are resolved, and relocation is indicated by some mechanism such as modification records or a bit mask
  - External references is often retained in the linked program
    - To allow subsequent relinking of the program to replace control sections, modify external references, etc.

Linking Loader vs. Linkage Editor

- *Linking loader*
  - *Searches libraries* and *resolves external references* **every time** the program is executed.
  - Avoid the writing and reading the linked program.
- *Linkage editor*
  - *Resolution of external reference* and *library searching* are only performed **once**

Linking loader

- Suitable when a program is reassembled for nearly every execution
- In a program development and testing environment
- When a program is used so infrequently that it is not
- worthwhile to store the assembled and linked version.

Linkage editor

- If a program is to be executed many times without being reassembled, the use of a linkage editor substantially reduces the overhead required.

14) What are machine independent loader features?

Machine-Independent Loader Features

- loading and linking are often thought of as operating system service functions.
- **Machine independent loader features:**
  - Automatic Library Search
  - Loader Options

Automatic Library Search for handling external references

- Allows programmers to use standard subroutines without explicitly including them in the program to be loaded.
- The routines are automatically retrieved from a library as they are needed during linking.

Linking loaders that support automatic library search:

- Enter the symbols from each ***Refer record*** into ***ESTAB***
- When the definition is encountered ***(Define record)***, the address is assigned
- At the end of Pass 1, the symbols in ESTAB that remain undefined represent ***unresolved external references***
- The ***loader*** searches the libraries specified for routines that contain the definitions of these symbols, and processes the subroutines found by this search exactly as if they had been part of the primary input stream
- Since the subroutines fetched from a library may themselves contain external references ,the library search process may be repeated.
- The programmers can override the standard subroutines in the library by supplying their own routines

15) **What are machine independent loader features?**

Common Loader Options – **Command Language**

- **Specifying alternative sources of input** :
- **INCLUDE** program-name(library-name)
    - Direct the loader to read the designed object program name
    - specified as a part of input program.
- **Changing or deleting external references**
- **DELETE** csect-name
    - Delete the named control section(s) from the program loaded
    - when not used
- **CHANGE** name1, name2
    - Change the external symbol name 1 to name 2 appeared in the
    - object program

Example

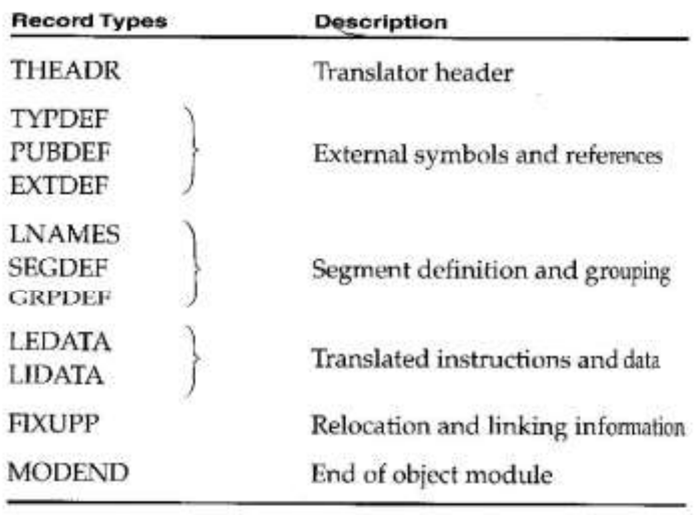
```
INCLUDE READ(UTLIB)
 INCLUDE WRITE(UTILB)
 DELETE RDREC, WRREC
 CHANGE RDREC, READ
 CHANGE WRREC, WRITE
```

16) **Write short notes on MSDOS - Linker**

MS-DOS Linker

- MS-DOS assembler (MASM) produce object modules (.OBJ)
- MS-DOS LINK is a linkage editor that combines one or more modules to produce a complete executable program (.EXE)
- **MS-DOS object module**
    - THEADER similar to Header record in SIC/XE
    - MODEND similar to End record in SIC/XE

MS-DOS Object Module



Record Types	Description
THEADR	Translator header
TYPDEF PUBDEF EXTDEF	External symbols and references
LNAMES SEGDEF GRPDEF	Segment definition and grouping
LEDATA LIDATA	Translated instructions and data
FIXUPP	Relocation and linking information
MODEND	End of object module

Figure 3.15 MS-DOS object module.

17) **Give an example of program using libraries.**

Example of Programs Using Libraries

main.c

```
include <stdio.h>
extern int a();
extern int b();
main()
{
int ret1, ret2;
ret1=a();
ret2=b();
printf("\n ret from a()= %d", ret1);
printf("\n ret from b()= %d", ret2);
}
```

a.c

```
int a()
{
return 5;
}i
nt a1()
{
return 8;
}
```

b.c

```
int b()
{
return 5;
}
```