Assignment 2: Loaders & Linkers

~~Linker.~~

Date:

Page:

Name: Rasad Regmi
Roll No: 221737
Program: Software
Subject: System Programming

1) What is a Loader? Differentiate Linking Loader from Linkage editor.

→ A Loader is a system software component that Loads an executable file into memory, preparing it for execution by setting up memory space, resolving symbolic references, and linking various program modules and libraries.

Difference:

	Linking Loader	Linkage Editor
a)	~~Combines and loads or program modules into memory, resolving add~~ Performs all Linking and relocation operations, including automatic library search, and Loads the Linked program into memory for execution.	Produces a Linked version of the program, which is normally written to a file or library for Later execution.
b)	Suitable when a program is reassembled for nearly every execution.	Suitable when a program is executed many times without being assembled

c)	Resolution of external reference and library searching is performed more than once.	Resolution of external reference and library searching is performed only once.
d)	Linking loaders perform Linking operations at Load time.	Linkage editors perform Linking operations before the program is loaded for execution.
e)	There is no need of relocating loader.	The relocating loader Loads the Load module into the memory.
f)	The Loading may require two passes.	The Loading can be accomplished in one pass.
g)	When program is in development stage then at that time the linking loader can be used.	When the program development is finished or when the library is built then linkage editor can be used.

2) Explain the working mechanism of absolute Loaders with the algorithm.

→ Absolute loaders is a Loader that ~~is used to~~ Loads the executable code into memory at

a specified address and then transfers control to that code.

Algorithm

```
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
```

a) The process starts by reading the header record of the object program.

b) This step involves reading the header record to gather basic information about the program

c) The loader verifies the program's name and its length to ensure that it matches the expected values.

d) The first text record of the object program is read. Text records contain the actual executable code.

e) This loop continues until the end record ('E') is encountered.

e.1) if the object to code is in character form, it is converted into an internal representation ~~schedut~~ suitable for memory ~~to~~ storage.

e.2) The object code is then moved to the specified location in memory.

e.3) The next object program record is read and processed.

f) Once the end record is reached, control is transferred to the ~~stra~~ starting address specified in the end record to begin execution of the loaded program.

g) The algorithm concludes its execution.

5) What is relocation? How is relocation carried out in a loader?

→ Relocation refers to the process of modifying a program so that it can execute correctly regardless of where it is loaded in memory.

Relocation in a loader is carried out by:

a) ~~Read Executable:~~

~~The~~ loader reads the executable file, ~~including~~ relocation information.

Date:
Page:

~~b) Determine Base Address:~~

a) Get Program Address:
Obtain the starting address where the program will be loaded.

b) Read Records:
Read each input record until the end of the input.

c) Process Text Records (T):
For text records, move the object code to the location specified by adding PROGADDR to the address in the record.

d) Process Modification Records (M):
For modification records, adjust addresses by adding program address to specified addresses according to the relocation bits.

e) End Record (E):
Stop processing when the end record is ~~en~~ encountered.

6) What are the main features of a machine dependent loader? When logically related parts of programming are linked then what is generated and why is it important?

→ Main features:

Date: .....
Page: .....

a) Relocation:
This involves adjusting the addresses used in the program so that it can execute correctly regardless of where in memory it is loaded.

b) Program Linking:
This involves resolving references between modules of a program.

c) Algorithm and Data Structures for a Linking Loader:

- Algorithm
  - > Pass 1: The loader scans the program to determine the size of the program and the addresses of external references
  - > Pass 2: The loader adjusts the addresses in the program and resolves external references, then loads the program into memory

- Data Structures
  - > Symbol Table: Stores the addresses of symbols that are defined in and referenced by the program.

Date:
Page:

> Relocation Table: Keeps track of which parts of the program need to be relocated
> External Reference Table: Keeps track of symbols that are used in one module but defined in another

When logically related parts of a program are linked, they generate modules or sections. These modules are crucial for several reasons:

a) Modularity:
Breaking down a program into smaller, manageable pieces makes it easier to develop, test, and maintain.

b) Reusability:
Modules can be reused across different programs, reducing redundancy and saving development time.

c) Encapsulation:
Encapsulating functionality within modules means changes in one module do not affect others, provided the interface remains the same.

d) Maintainability:
Isolating different parts of the program makes it easier to locate and fix bugs.

Date:
Page:

e) Parallel Development:
Multiple developers can work on different modules simultaneously, speeding up the development process.

7) Explain different loader design options.
→ Loaders are essential components that place programs into memory, prepare them for execution, and start their execution.

Diff. ~~Loader~~ loader design options are:

a) Linkage Editors:
Linkage editors are programs that links multiple object modules into a single executable, resolving external references.

Advantages: Supports modular programming and optimizes the final executable
Disadvantages: Can be complex and time-consuming with many modules.

b) Dynamic Linking:
Dynamic ~~L~~ Linking defers the linking of some modules until run-time, loading libraries as needed.

Advantages: Reduces memory footprint, allows easy updates without recompilation.

Date :
Page :

Disadvantages: Introduces runtime overhead and complexity in managing dependencies.

c) Bootstrap Loaders:
Bootstrap loaders small programs that initialize hardware and load the operating system at startup.

Advantages: Ensures a consistent system ~~set~~ startup, necessary for system initialization.
Disadvantages: Limited functionality, designed only ~~Limited functionalt~~ for initial system loading.

9) Refer the program on page 137 of the textbook and the solution on page 138. Explain why the modification records are added in such fashion.

→ The modification records in an object program for a SIC/XE system are used to adjust ~~adt~~ addresses that need to be modified during the loading process.

Explaination:

- Relocation Adjustment:
Modification records ensure that the loader adjusts the addresses in the object ~~code~~ code during the loading process. This is crucial because the program might be loaded at a different address than where it was originally assembled.

Date: .....
Page: .....

- Specific Address Adjustments:
For instance, M^000007^05 + COPY indicates that at address 000007, the loader needs to add the address of the COPY program's starting address. This ensures that the instruction references the correct memory location after relocation.

Similarly for M^000014^05 + COPY and M^000027^05 + COPY.

- Ensuring Correct Execution:
These adjustments ensure that instructions like jumps, subroutine calls, and data references point to the correct memory locations, maintaining the program's functionality after relocation.

10) Refer the program on page 139 and its object program on page 140 and explain why the text records are modified.

→ The text records are modified for relocation, which allows the program to run correctly ~~regardless~~ regardless of where it is loaded in memory.

Explanation:

Date: .....
Page: .....

- **Relocatable Addresses:**
  Addresses in the object code need to be adjustable. The modifications ensure that when the program is loaded, these addresses are updated to the correct memory locations.

- **Relocation Bit Mask:**
  The bit mask in the object program indicates which parts of the text records contain addresses needing relocation. A bit set to 1 means the corresponding address must be adjusted.

- **Modification Records:**
  M records specify locations and lengths of addresses that need to be relocated. The loader uses these to adjust the addresses during loading.

11) ~~Refer the pages 143 to 145, explain the objec program.~~

→

12) On page 146 (figure 3.12 (a)), explain the loading process.

→ Explaination

a) Assigning Memory Addresses:
Each program (PROGC, PROGB, PROGA) is assigned a starting memory address.
- PROGA starts at 4000.

- PROGB starts at 4040.
- PROGA starts at 4080.

b) Placing Program Contents in Memory:
The instructions and data for each program are placed into memory at their designated addresses.
For instance, PROGC's instructions and data are placed starting from 4000 to 403F, PROGB from 4040 to 407F, and PROGA from 4080 to 408F.

c) Updating Addresses:
The loader updates any relative addresses within the programs to reflect their actual memory locations.
This involves updating pointers, jump addresses and any other address dependent code.