76_ Adnan Shaikh

Page No.: 1
Date. / /

Subject :- SPCC

Sem :- VI

Assignment No :- 1

Q.1) What is system programming? List some system programs and write their functions.

→ System Software or System program :-

System software is the software which is required to run the hardware parts of computer and other application software.

System program is a set of program which are developed to operate, control and extend processing capabilities of computer itself.

- Types of system software :-

1. Standard System program
   - It contains assembler, processor, linker, compiler, debugger, loaders, editors etc

2. Operating System Software
   - It include all operating system software which can be used to create interface between hardware and user application.
   - Eg :- Windows, DOS, UNIX, Linux etc.

(1) Assembler :-

- Assembler is a system software which convert programmes written in Assembly language into Machine language.

(2) Macro processor :-
- A Macro processor is a program which copes a bunch of text from one place to another. It replaces the macro cell into macro definition.

(3) Loader :-
- Loader performs the function of placing object code into main memory for execution purpose.
- To do this loader translate object code into executable form.

(4) Linker :-
- Linker is a program in a system which helps to link a object module of program into a single object file.

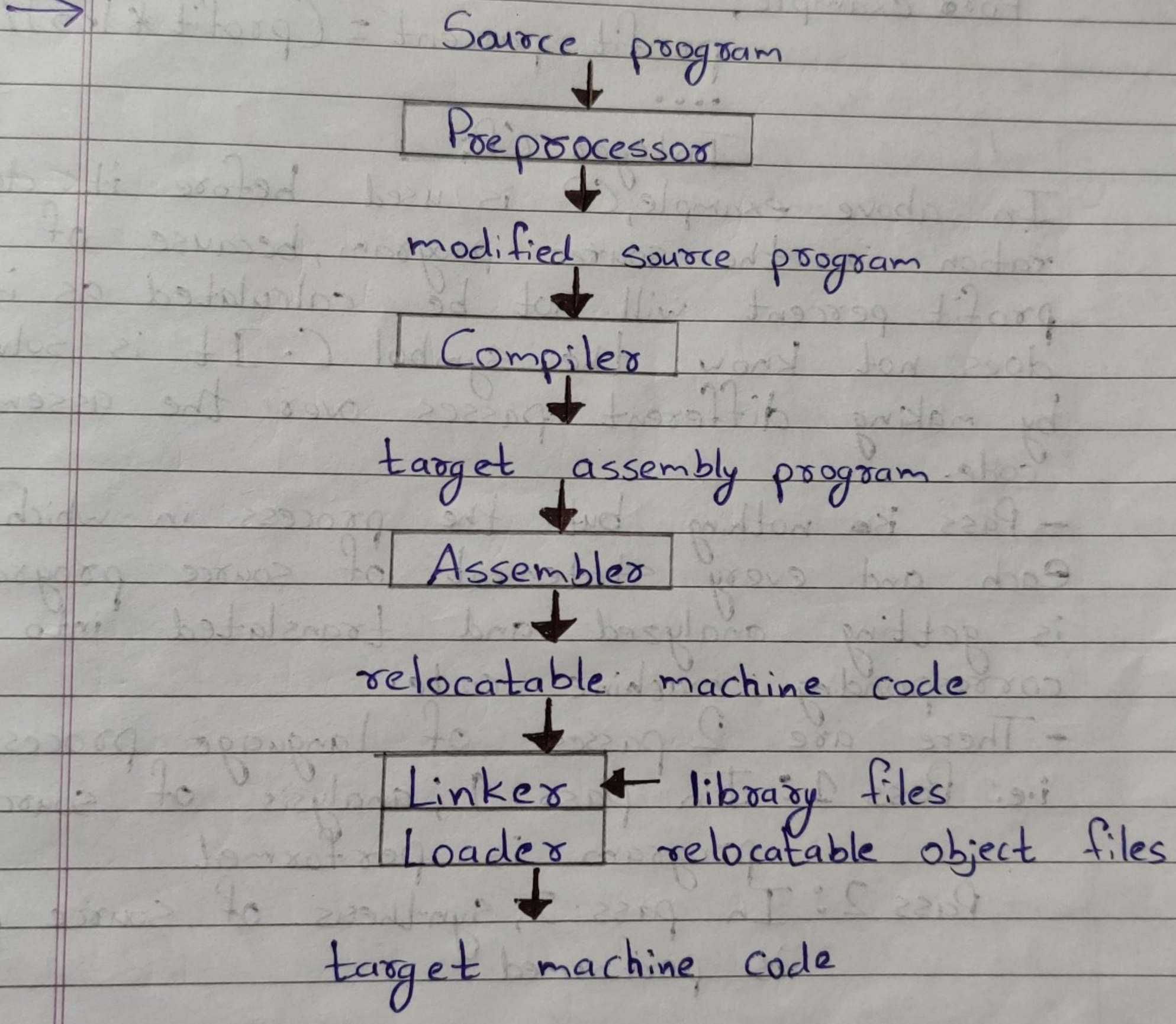
(5) Compiler :-
- Compiler is a system program that translates high level language into low level language.

(6) ~~Inter~~ Interpreter :-
- It is also a translator like a compiler but interpreter takes one line at a time and translate it into object code, then go for next line and interprets. This process continue until entire source code is interpreted.

(7) Operating System :-
- It acts as a interface between computer user and computer hardware.
- Eg :- Windows, IOS, Linux etc.

Q.2) Indicate the order in which following System program are used from developing program upto its execution:- Assembler, Loader, Linker, Macro processor, Compiler, Editor

→



Q.3) Explain forward reference problem and how it is handled in assemble design.

→ - When statement is processing, some of the symbol are used but they are not declared anywhere in the source program.
- Due to this the processor does not find its corresponding memory address and gets failed to generate target program.

Page No.: 4
Date. / /

- If the symbol is used before its declaration then the synthesis cannot continue its task. Such problems is called as forward reference problem.

For example,

```
profit percent = (profit * 100)/CP
....
Integer CP;
```

In above example, CP is used before its declaration in the source program, because of this profit percent will not be calculated as it does not know the symbol C. It is solved by making different passes over the assembly code.

- Pass is nothing but the process in which each and every statement of source program is getting analyzed and translated into its corresponding machine code.
- There are 2 passes of language processor i.e. Pass 1: In pass 1, analysis of source program is performed

Pass 2: In pass 2, synthesis of source program is performed

Q.4) Explain with neat flowchat and database working of two ~~part~~ pass assembler.

→ In two pass assembler the forward reference problem can be resolved easily because in first pass all the LC processing is done and all the symbols that are used in the source program has the ~~valued~~ valid memory address associated with them.

- So the second pass only use the symbol and generate the target code by using the address found in symbol table.

- Pass 1 :-

1) It seperates the labels mnemonic opcode and operand fields of a statement.
2) Validate the mnemonic opcode through opcode table.
3) Build symbol and literal table.
4) Perform the LC processing.
5) Generate the intermediate code.

- Pass 2 :-

1) Take the address of symbols from symbol table addresses of literals from literal table and address of opcode from opcode table.
2) Synthesize the target program.

Page No.: 6
Date. / /

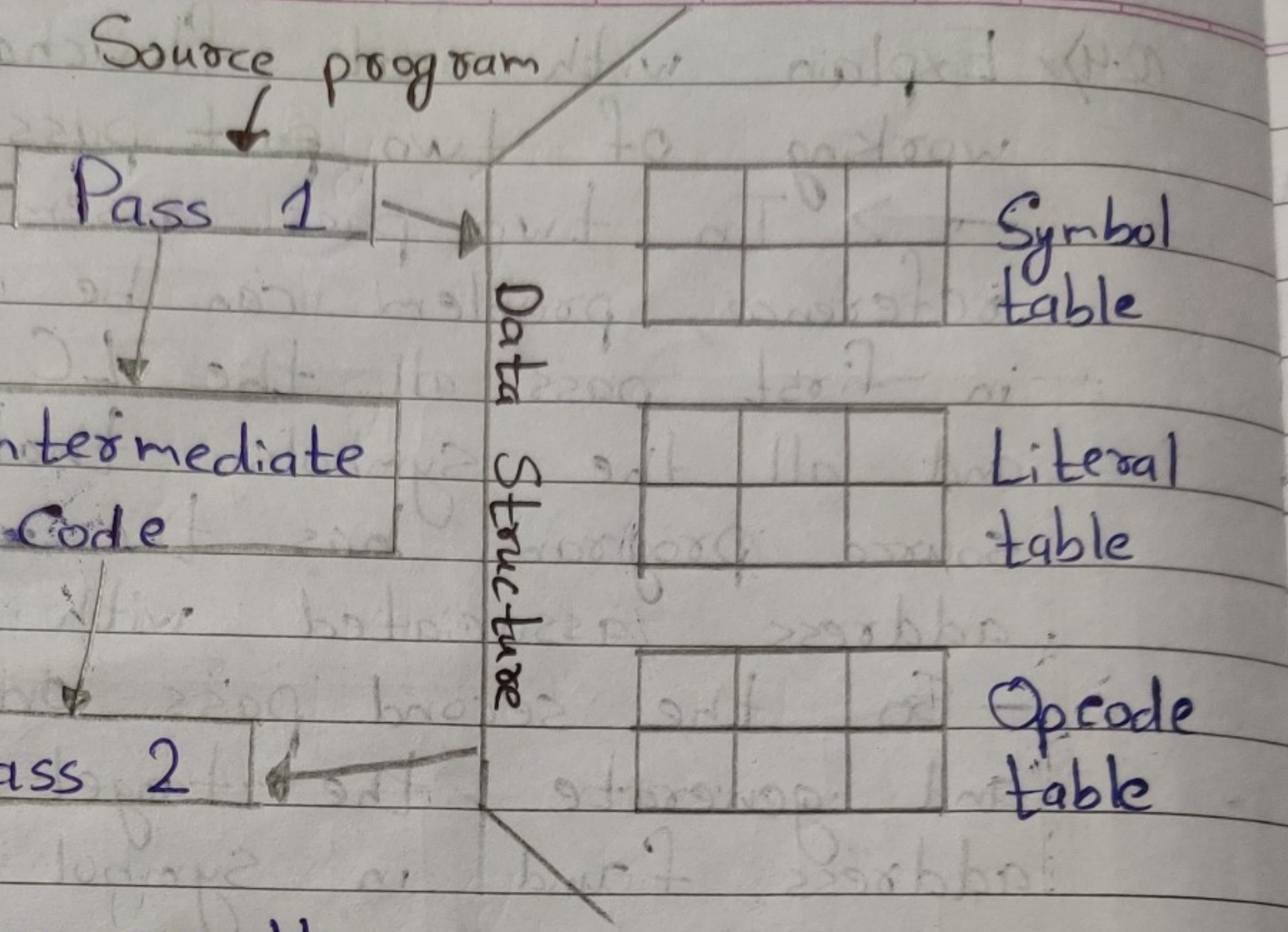


Fig - Two Pass assembler

Page No.: 7
Date. / /

Q.5) Explain Single pass assemblers with neat flowchart and databases.

→

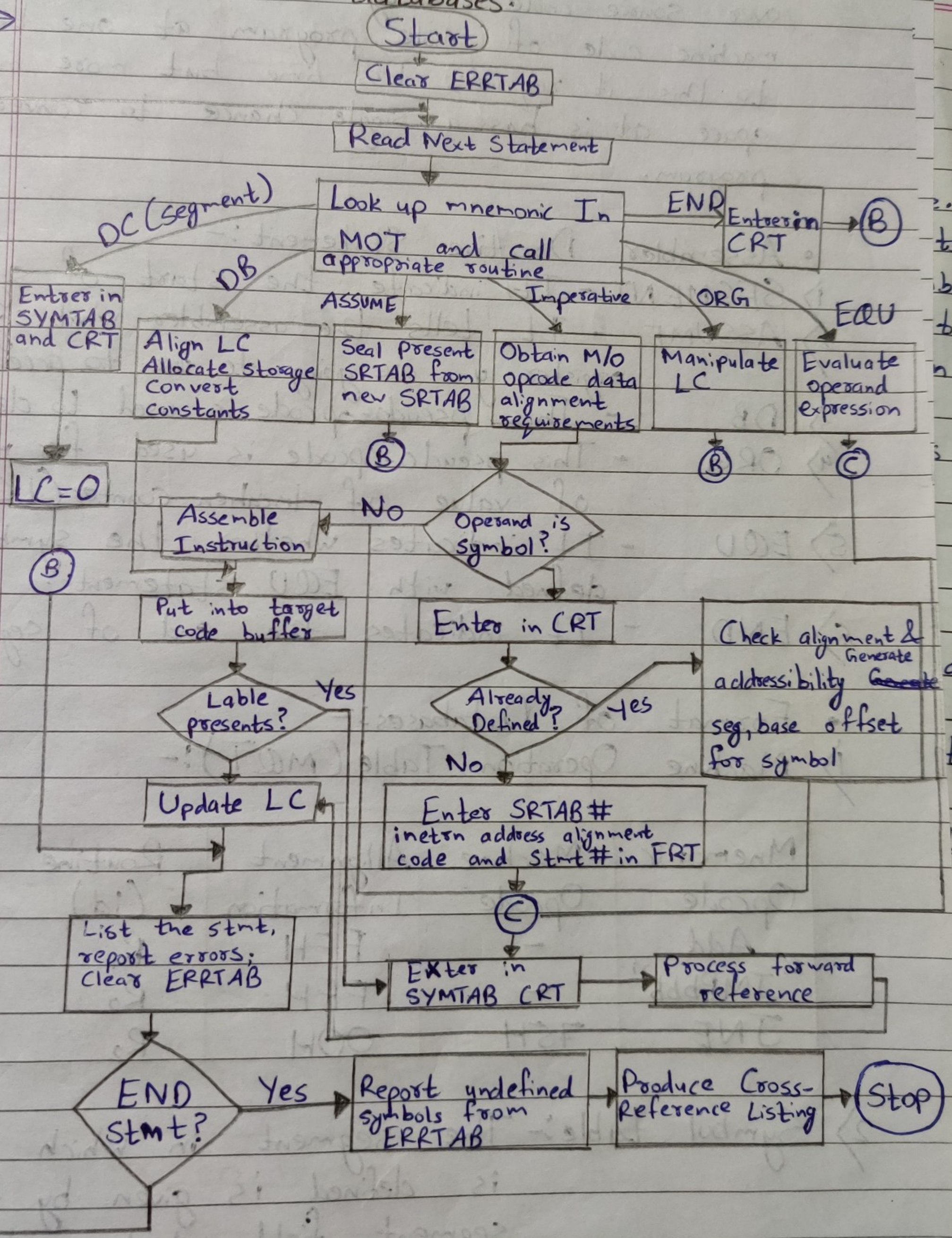


Fig :- Flowchart of the Single Pass Assembler

- In single pass assembler the assembler passes over source code exactly once and converts equivalent machine code of whole program at one time. Due to this it requires less time but more memory space at is has a single chance to convert the program.

- Assembler Directive Statement :-

1) SEGMENT - It indicate the start of segments.
2) ASSUME - It tells the assembler what segment register you are going to use to access a segment.
3) DB - This pseudo-opcode is used to define bytes.
4) ORG - This pseudo-opcode is used for manipulation of value of locahen counter (LC).
5) EQU - It indicates whether the symbol is defined with EQU statements.
6) END - It indicates the end of segment.

- Format of Databases :-

1) Machine Operation Table (MOT) :-

Mnemonic Opcode	Machine Opcode	Allignment Information	Routine (id)
Add	-	FFH	R_1
"JMPbbb"	-	FFH	R_2
JNE	75H	OOH	R_2

2) Symbol table :- The segment in which the symbol is defined is given by the owner segment field which contain the SYMTAB entry.

3) SEGMENT REGISTER TABLE (SRTAB):-
Whenever an assembler encounter, ASSUME_Stmt it stores the previous SRTAB on the stack and new SRTAB is created.

Segment Reg	Segment Name

4) STORED SEGMENT REGISTER TABLE (STSRT):-

	Segment Register	Segment Name	
00	ES		#1 STSRT
01	CS (code segment)		
10	SS (Stack -11-)		
11	DS (Data -11-)		
00	ES		
01	CS		
10	SS		STSRT #2
11	DS		

5) Forward reference table (FRT):-

Pointer to Next Entry (2)	Entry # in STSRT (1)	Instrn Address	Usage Code (1)	Source Stmt (2)

6) Cross Reference Table (CRT):-

Pointer to next entry	Source stmt #

76_Adnan Shaikh

Q.6) Explain two pass macro processor with neat flowchart and database.

→ - It is used for, identifying the macro name and performing expansion.

- Features of Macroprocessor :-

i) Recognized the macro definition
ii) Save macro definition
iii) Recognized the macro call
iv) Perform macro expansion

- Forward reference problem :-

- The assembler specifies that the macro definition should occur anywhere in the program
- So there can be changes of macro call before its definition which gives rise to the forward reference problem and macro

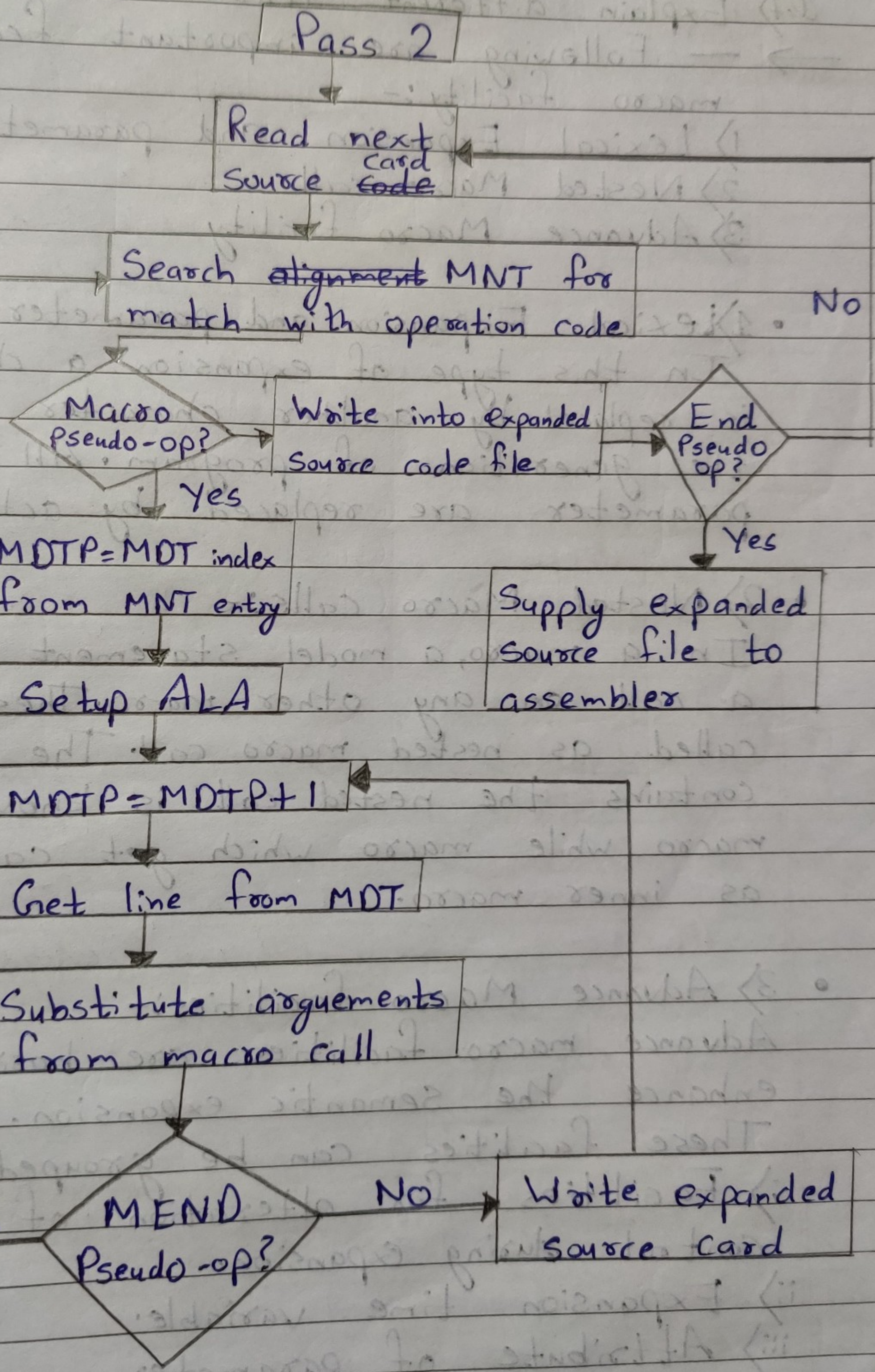
1. Pass 1 :- Recognize macro definition, save macro definition.
2. Pass 2 :- Recognize macro call, perform macro expansion.

- Database required for pass 2 :-

In Pass 2, we perform recognize remicro call and macro expansion.

i) Copy File :- It is a file which contain the output given from pass 1.
ii) MNT :- It is used for recognizing macro call and expansion
iii) MDTP :- It is used to point to index of MDT. The starting index is given by MNT.
iv) ALA :- It is used to replace the index notation by its actual value.
v) ESC :- It is used to contain the expanded macro call which is given to the assembler for further processing.

Pass 2


Read next source card
Search MNT for match with operation code
No
Macro Pseudo-op?
Write into expanded Source code file
End Pseudo op?
Yes
MDTP=MDT index from MNT entry
Yes
Supply expanded Source file to assembler
Setup ALA
MDTP=MDTP+1
Get line from MDT
Substitute arguements from macro call
Yes
MEND Pseudo-op?
No
Write expanded Source card

Q.7) Explain different features of macro facility

→ — Following are important features of macro facility :-

1) Lexical Expansion and parameter substitutioner?
2) Nested Macro call
3) Advance Macro facility

- 1) Lexical Expansion and parameter substitution :-
In this type of expansion a character string is replaced by another character string in the generation of program. All the formal parameter are replaced by actual parameter

- 2) Nested Macro calls :-
In a macro, a model statement can constitute a call on any other macro. These calls are called as nested macro call. The macro which contains the nested call is known as outer macro while macro which get called is known as inner macro.

- 3) Advance Macro facilities :-
Advance macro facilities are basically used to enhance the semantic expansion.
These facilities can be grouped into :-
i) Facilities for alteration of flow of control during expansion.
ii) Expansion time variable.
iii) Attribute of parameter.

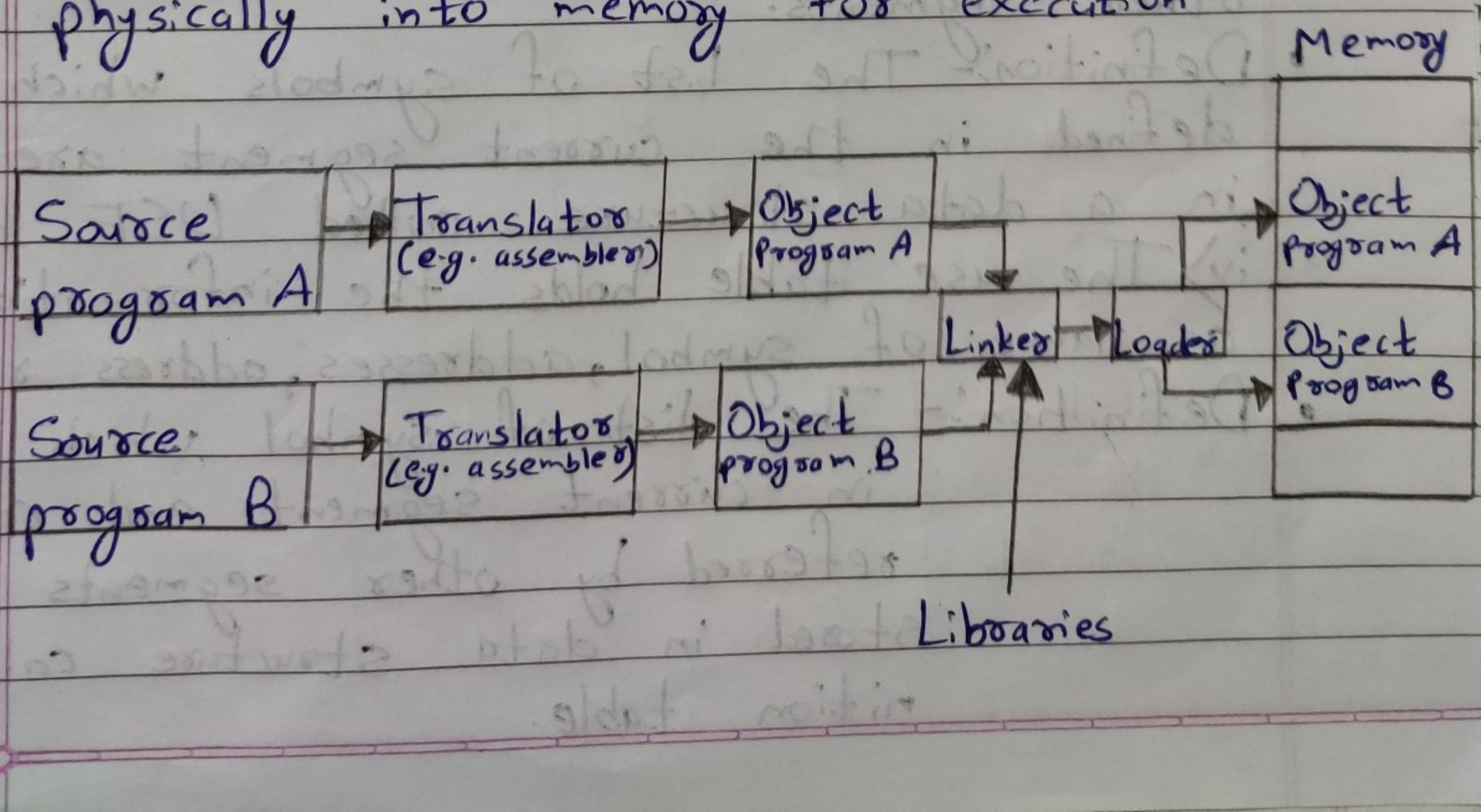
Page No.: 13
Date. / /

Q.8) What are different function of loader? Explain in brief.

→ Function of loader :-

i) Allocation
ii) Linking
iii) Relocation
iv) Loading

- i) Allocation :- Allocates memory space for program
- ii) Linking :- Binding of objects program code with necessary library routine or other object program codes to generate executable program.
- iii) Relocation :- It changes the memory address of address sensitive instruction of program so that it can be loaded at an address different from the location originally specified.
- iv) Loading :- Finally, executable file content (i.e. Machine instruction and data) will be placed physically into memory for execution



76. Ainan Shaikh

Q.9) Explain the working of a direct linking loader with proper example. Clearly show the entries in different database built by direct linking loader.

→ • It is a type of relocatable loader.
• The direct linking loader is most common type of loader.
• The loader cannot have direct access to the source code.
• And to place object code in the memory there are 2 situations:- Either distance of the object code could be absolute or can be relative addresses
• The assembler should give the following information to the loader:-
i) The length of the object code segment.
ii) The list of all symbol which are not defined in the current segment but can be used in current segment.
iii) The list of all symbol which are defined in current segment but can be referred by other segments.
• Definition:- The list of symbols which are not defined in the current segment are stored in a data structure called USE table.
iv) The use table holds the information such as name of symbol, addresses, address relativity.
• Definition:- The list of symbol which are defined in current segment and can be referred by other segments are stored in data structure called definition table.

v) The definition table holds information such as address, symbol.

* Object decks for direct linking loader:-
1) External symbol Dictionary (ESD) record:-
- Entries and Eternals
2) Text (TXT) Records:-
- Control the actual object code tognslated version of source program.
3) Relocation and Linkage Directory (RLD) Record:-
- Contains relocation information like location in the program whose contents depend on the address at which program is placed.
- These information are:-
- Location of each constant that needs to be changed due to relocation.
- By what it has to be changed.
4) END Record:-
- Specifies the starting address for execution

Page No.: 16
Date. / /

* Example:-

```
John    START
        ENTRY RESULT
        EXTERNAL                        SUM
LOAD    1,POINTER               0  LOAD  1,48
LD-ADWR 15,ASUM                 4  LD-ADWR 15,56
BALR    14 15                   8  BALR  14,15
STORE   1 ~~RESULT~~ RESULT     IC 4LT   0,0

TABLE   DC      1,7,9,10,13             28,0001
                                        2A,0007
                                        2C,0009
                                        30,000A
                                        34,000D
POINTER DATA    ADDR(TABLE)             48,0028
RESULT  NUM     0                       52,0000
ASUM    DATA    ADDR(SUM)               56, ???? External
        END
```

- SD - Segment Definition
- Local Definition
- External Reference

* RLD record:-

Symbol	Flag	Length	Relative Location
JOHN	+	4	48
SUM	+	4	56

* ESD record:-

Symbol	Type	Relative Location	Length
JOHN	SD	0	64
Result LD	52		
SUM	ER	-	-

Q.10) Explain different loader scheme

→ Types of loader :-

1) Absolute loader :-
It is a simplaced type of loader scheme. In this scheme the loader simply accepts the machine language code produced by assembler and place it into main memory at the location specified by the assembler. The task of an absolute loader is virtually trivial. Absolute loader is simply to implement but it has several disadvantages.

2) Compile and Go Loader :-
A compile and go loader is one of the loader in which assembler lies in memory and loader itself does the process of assembling and loading machine instructions and data at specified memory location. Instruction are read line by line. Machine code is generated and directly put in the memory at some known address. After completion of assembly process, assembler assign starting address of the program to locate counter.

3) General Loader :-
The source program is converted to object program by some translator. The loader accept these object modules and put machine instruction and data in an executable form at their assigned memory. The loader occupies some portion of main memory. The size of loader is smaller than assembler.

4) Subroutine Linkages / Program linking loader :-
In a given program, it is often needed to perform a particular subtask; Many times on different data values. Such a subtask is called as subroutine.

5) Relocating Loaders (BSS) :-
Loaders that allow for program relocation are called relocating / relative loader i.e. it loads a program in specific area of memory, rebcate it so that it can execute correctly.

6) Direct Linking Loaders :-
It is a type of relocatable loader. The direct linking loader is the most common type of loader. The loader cannot have direct access to the source code and to place the object code in the memory there are two situations :- Either address of subject could be 'absolute' or can be relative. If at all the address is relative then it is the assemblers who informs the loader about the relative adresses.

7) Dynamic Linking Loader :-
Dynamic loader is the loader that actually intersect the "calls" and loads the necessary procedure is called over super-visor or simply flipper. Thus over all scheme is called Dynamic loading or load on call.