Sub :- L P.

Page No.
Date: / /

Practical No I

Title :- Design Suitable data structure and implement pass-I of a pass-II assembler for Pseudo machine in Java or C++ using object oriented feature.

Objective :-
1) To learn System programming tools
2) Implement language translator
3) To learn the data structure used in assembler.

problem statement :- Design Suitable data structure and implement pass 1 and pass 2 of two pass assembler for pseudo-machine implementation should consist of a few instruction from each category and few assembler directives The output of pass 1 (intermediate code file & symbol table) should be input for pass 2.

* Software and Hardware Requirement :-

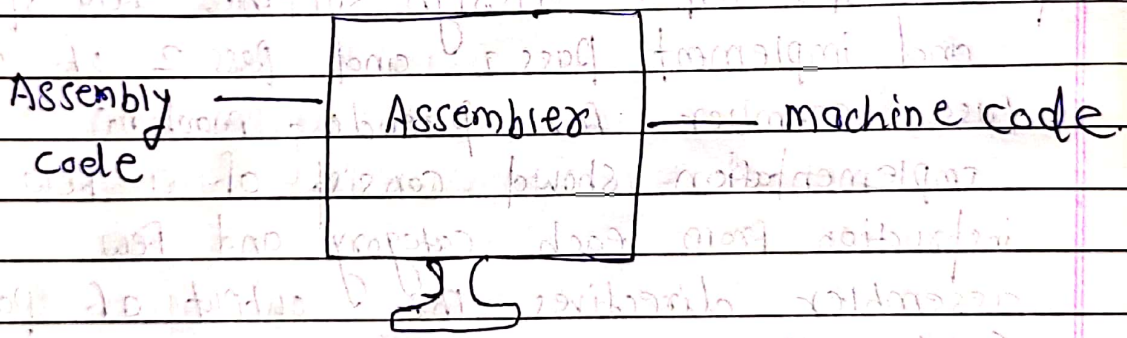
1) Software :- Ubuntu operating System and Notepad

2) Hardware :- IS machine.

Teacher's Signature

Theory :—

* Assembler :— Assemblers are used to translate a program written in a low-level assembly language into machine code (object code) file so it can be used and executed by the computer once assembled the program file can be used again and again without reassembly.



Infruction :—

1) An instruction is a statement that becomes executables when a program is assembled.

2) Instruction are translated by the assembler into machine language byte which are loaded and executed by the CPU at run time.

Teacher's Signature

Page No.
Date: / /

Instruction Format :—

- When the assembler processes an instruction it converts the instruction from it mnemonics form to standard machine language format called the "Instruction Format".

Syntax :— [Label] : mnemonic operand1 , operand2 ;

Where

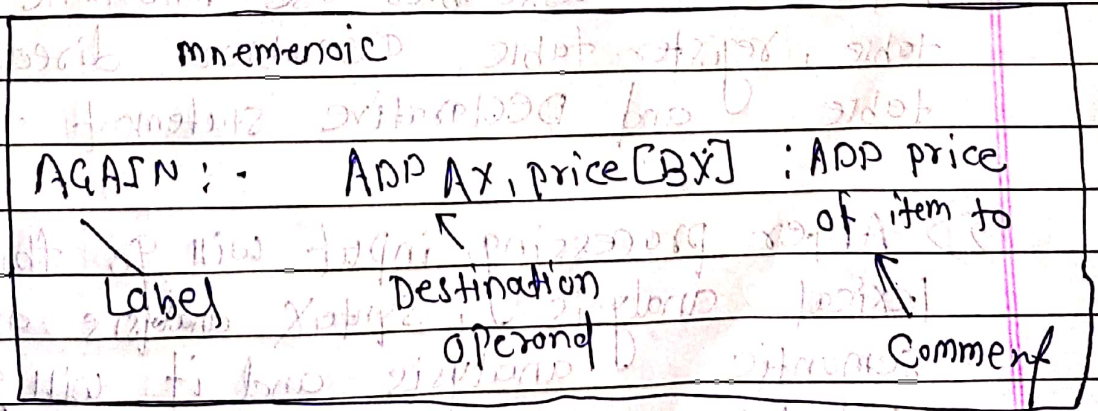
Label : optical

Instruction mnemonic required such as mov, Add, SUB, NULL

operands :- Usually required

Comment optional

Example :-



Teacher's Signature

* Data Definition statement :-

Syntax :-

[name] directive initializer [, initializer]

where

Name :- optional

Directive :- It can be BYTE, WORD, DWORD or Any others types.

Initializer :- At least one initializer is required, even if it is zero initial such as 00110010b, 32h and 50d

* The design of pass-I assembler with the help of flowchart and example.

* Design of pass-I

1) Pass 1 of assembler use mnemanic op-code table, register table, assembler directive table and Declarative statement table.

2) After processing input will go through lexical analysis, syntax analysis and semantic analysis and it will generate symbol table operation code table literal table and pool table (SYMTAB, OPTAB LITTAB, POOLTAB)

Teacher's Signature

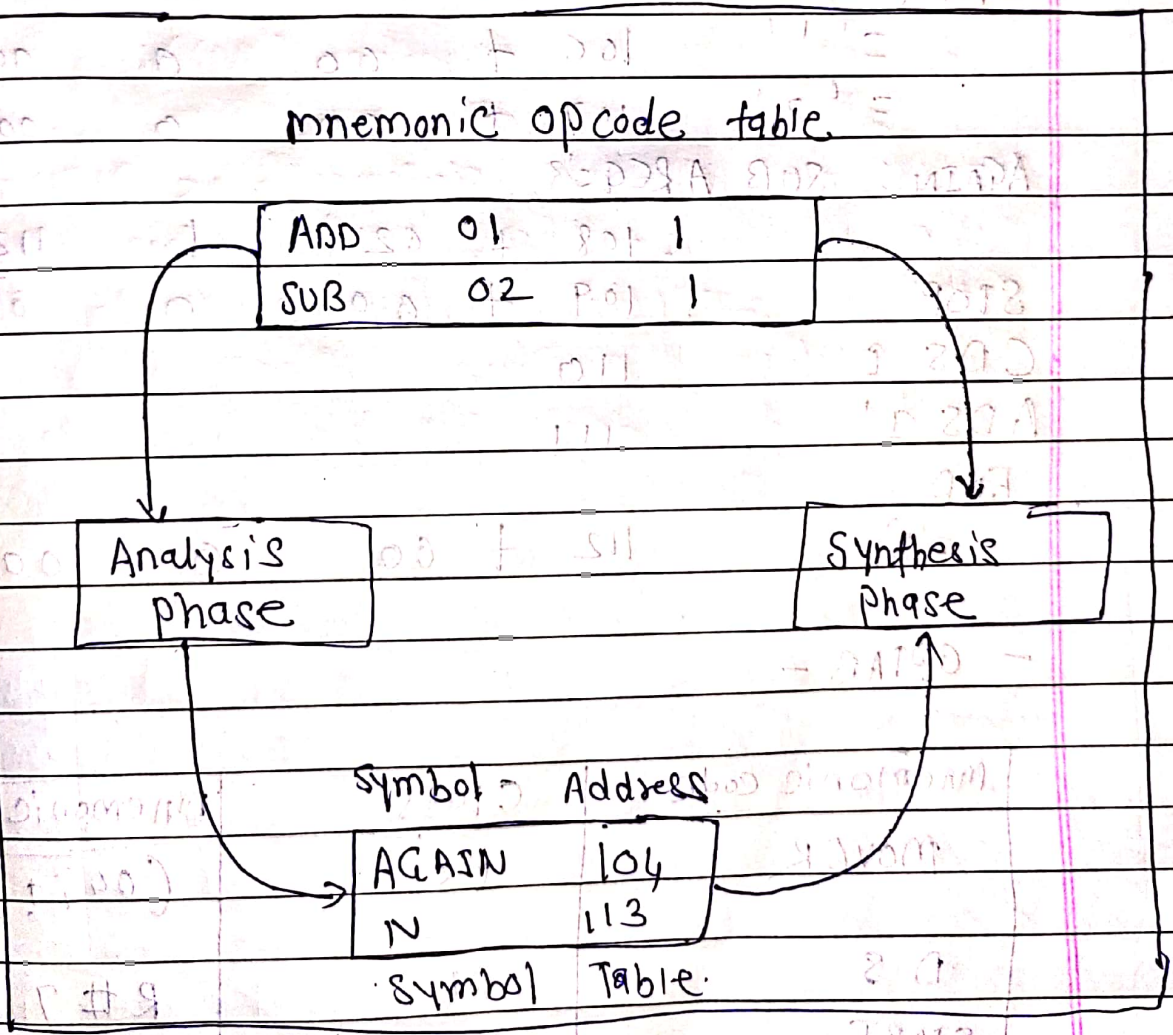
3.7 steps required to design on pass1 or 2 pass assembler.

a) seprate the symbol, mnemonic opcode and operand fields.

b) Build the symbol table.

c) perform LC processing.

d) construct intermediate representation



Teacher's Signature

Example :-

START 100	LC	machine	Instruction
MOVER CREG='1'	100 +	04 3	106
MOVEM AREG,C	101 +	05 1	110
MOVER BREG,A	102 +	04 1	110
MOVER BREG,A	103 +	04 2	111
ADD CREG='2'	104 +	01 3	107
BC ANY,AGIN	105 +	07 6	108
LTORG			
='1'	106 +	00 0	001
='2'	107 +	00 0	002
AGAIN: SUB AREG='3'	108 +	02 1	112
STOP	109 +	00 0	000
CDS 1	110		
ADS 1'	111		
END			
='3'	112 +	00 0	003

- OPTAB -

Mnemonic Code	Class	Mnemonic Info
MOVER	IS	(04,1) → Mnemonic code value
DS	DL	R#7
START	AD	R#11
'	'	'

Teacher's Signature

Page No.
Date: / /

SymTAB :-

Symbol	Address	Length
C	110	1
A	111	1
AGAIN	108	1

LITTAB :-

SR. NO	Literal	Address
1	='1'	106
2	='2'	107
3	='3'	112

- POOLTAB

Starting address of first Literal pool in Serial number in LITTAB	#1
	#2

→ starting address of second Literal pool in Serial in POOLTAB

Teacher's Signature

* Various data structure used in pass-I along with its formed and significence of each field.

1) OPTAB :-
- Contain the fields mnemonic opcode class and mnemonic information.

- The 'class' field indicates whether opcode corresponds to an imperative statement (IS) a declaration statement (DL) or an assembler directive (AD).

- If an imperative statement is present then the mnemonic info field contain the pair (machine → opcode, instruction length else if contain the pair id of a routine to handle the declaration or directive statement.

2) SYMTAB :-
- Contain the fields Address and length.
- The processing of an assembly statement begins with the processing of its label field.

- If it contains a symbol, the symbol and the value in L.C is copied into a new entry of SYMTAB.

Teacher's Signature

- If it is impressive statement, then length is also entered into the symbol table.

3) LITTAB :-

- It used to collect all literals used in the program.

- The awareness of different literals Pools in maintained by an auxiliary table POOLTAB.

4) POOLTAB :-

- This table contain the literal no of starting literal of each literal pool.

mnemonic opcode	class	mnemonic info
MOVER	IS	(04,1)
DS	DL	R#7
START	AP	R#11

Symbol	address	length
Loop	202	1
NEXT	214	1
LAST	216	1
A	217	1
BACK	202	1
B	218	1

SYMTAB

Teacher's Signature

	Literal	Addr
1	='5'	
2	='1'	
3	='1'	

literal Address

1	='5'
2	='1'
3	='1'

LITTAB

literal no

#1
#3
—

POOLTAB

Algorithm :—

1) Start
2) location counter initialize first entries of all tables of zero
3) Read statement from input file one by one
4) while next statement is not END statement
5) Tokensize or seperate out input statement as label, numenomic, operand, operation
6) If label is present insert label into symbol table

Teacher's Signature

7) If the statement is ITORG statement processes it by making its entry into literal table pool table and allocate memory.

8) If statement is START or ORIGEN process location

9) If an all statement assign value to symbol by correcting entry in symbol table.

10) For declaritive statement update code size and location counter.

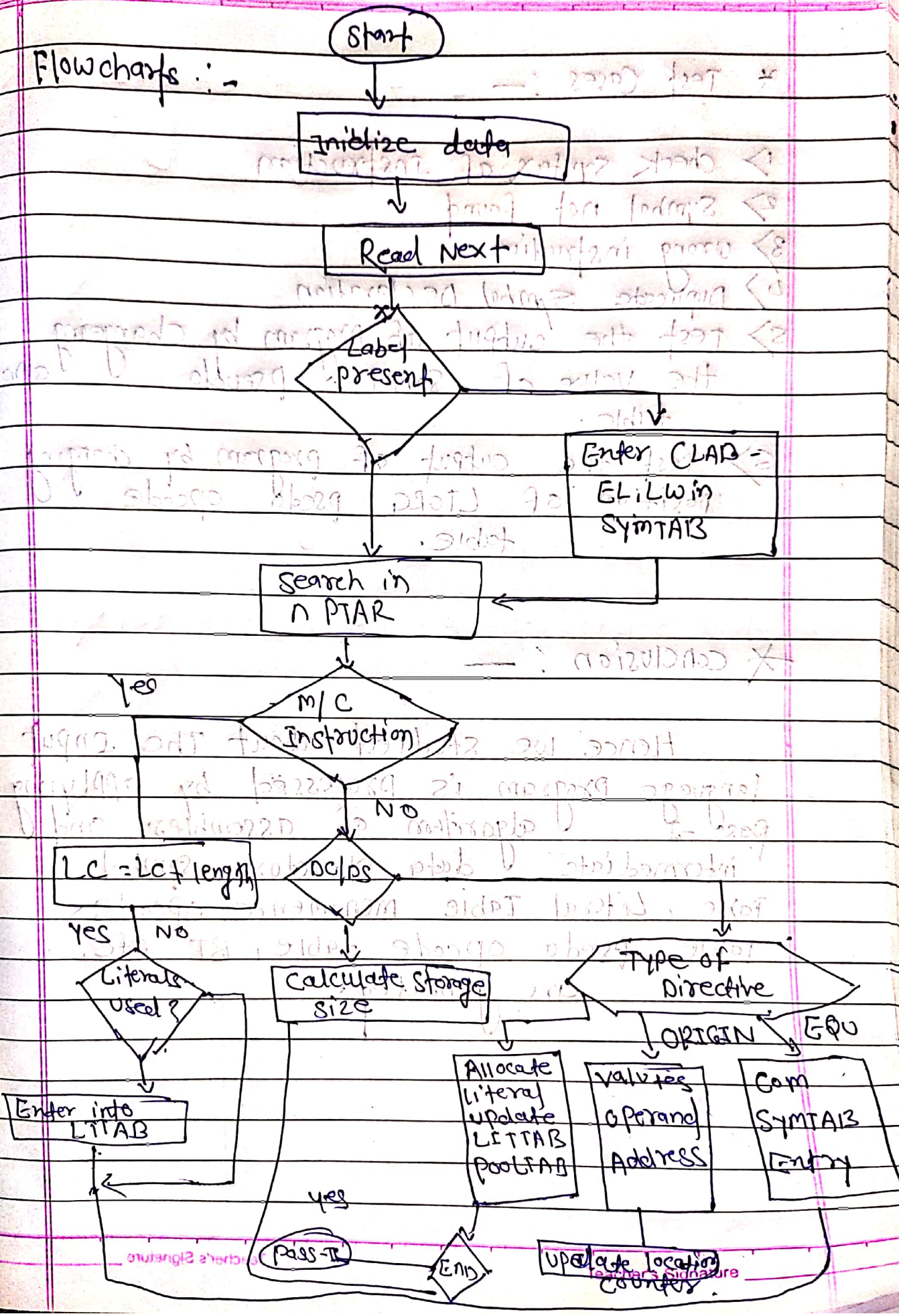
11) Generate intermediate code.

12) Pass this intermediate code to pass-2

13) END.

Page No.
Date: / /

Flowcharts :-



Scanned with CamScanner

Page No.
Date: / /

* Test Cases :—

1) Check syntax of Instruction
2) Symbol not found
3) Wrong instruction
4) Duplicate Symbol Declaration
5) Test the output of program by changing the value of START pseudo opcode table.
6) Test the output of program by changing position of LTORG pseudo opcode table.

* Conclusion :—

Hence we studied about The input language program is processed by applying pass-I algorithm of assembler and intermediate data structure, symbol Table, Literal Table menomenic opcode Table pseudo opcode table, BT etc. are genrated.

Teacher's Signature