Bhushan Pisal

Sub :- LP-I
SPOS

Page No.
Date: / /

Practical NO:02

* Title :- Implement pass-II of two pass assemble for pseudo machine in Java using object oriented features The output of assignment-1 (intermediate-file and symbol table) should be input for this assignment.

objective :-

- To understand Data structures of pass-I pass-2 assembler.
- To understand pass-I & pass-2 assembler concept
- To understand Advanced Assembler directive

Outcome :-

- After completion of this assignment implemented pass-2 assembler
- Implemented machine code from intermediate code.
- understand Concept of pass-2 Assembler.

Software Requirement :- Java JDK, notepad, Ubuntu OS.

Hardware :- intel i5.

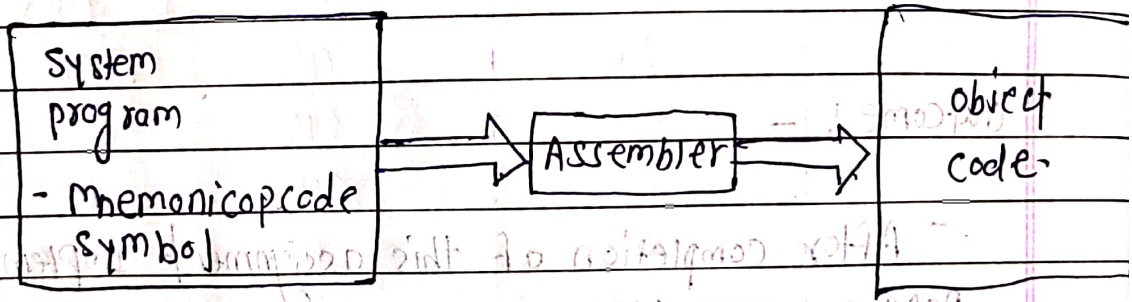
Teacher's Signature

Page No.
Date: / /

* Theory :-

- Introduction :- There are two main classes of programming language high level (e.g. c passes) and low level. Assembly language is a low level programming language programmers code symbolic instruction each of which generates machine instruction

An assembler is a program that accepts as input an assembly language program (source) and produces its machine language equivalent (object code) along with the information for the loader.



* Design of Two pass Assembler :-

Pass - 1 (Analysis of source program)

1) Separate the symbol, mnemonic opcode and operand fields
2) Build the symbol table
3) Perform LC processing
4) Construct intermediate representation.

Teacher's Signature

Pass -2

Process the intermediate representation (IR) to synthesize the target program.

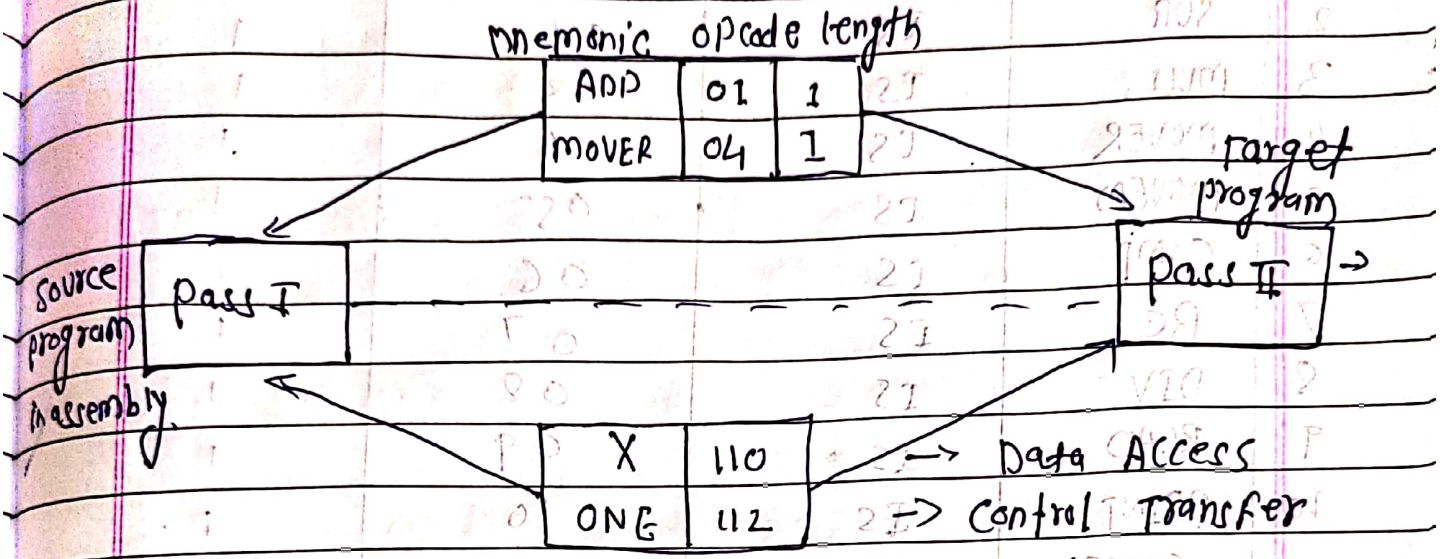


Fig. Data structure for assembler

IS :- Imperative Statement
AD :- Assembler Directive
DL :- Declaration Statement
RG :- Register
CC :- Comparison Condition

Teacher's Signature

	mnemonic op	Class	opcode	length
0	STOP	IS	00	1
1	ADD	IS	01	1
2	SUB	IS	02	1
3	MULT	IS	03	1
4	MOVER	IS	04	1
5	MOVEM	IS	05	1
6	COMP	IS	06	1
7	BC	IS	07	1
8	DIV	IS	08	1
9	READ	IS	09	1
10	PRINT	IS	10	1
11	START	AD	01	-
12	END	AD	02	-
13	ORIGIN	AD	03	-
14	EQU	AD	04	-
15	LTORG	AD	05	-
16	DS	DL	01	-
17	DC	DL	02	-
18	AREG	RG	01	-
19	BREG	RG	02	-
20	CREG	RG	03	-
21	DREG	CC	01	-
22	LT	CC	02	-
23	GT	CC	03	-
24	LE	CC	04	-
25	GE	CC	05	-
26	NE	CC	06	-
27	ANY	CC	07	-

Teacher's Signature

Page No.
Date: / /

Algorithm :—

1) Code area address = address of code area
   loop tab - ptr = 1;
   LOC - cntr = 0;

2) While next statement is not an END statement

a) Clear the machine code - buffer

b) If an LTORG statement

I) Process literals in LITTAB [POOLTAB) (Pooltab PTR)]

LITTAB [POOLTAB [Pooltab - ptr+1]] - 1

Similar to processing to constants in a dc statement

II) Size = size of memory area required for literals

III) Pooltab PTR = Pooltab - ptr+1

c) If a start or ORIGIN Statement then

I) LOC cntr = value specified in operand field

II) size = 0;

d) If a declaration statement

I) If a DC statement then assemble the

Teacher's Signature

Scanned with CamScanner

constant in machine code-buffer.

II) side = size of memory area required by DC or CS

e) If an imperative statement then

i) get operand address from SYMTAB or LITTAB

ii) Assemble instruction in machine code buffer

iii) size = size of instruction

f) If size ≠ 0 then

i) move contents of machine code buffer to the address code-area + address LOC cntr.

ii) LOC cntr = LOC cntr + size.

* Test case :-

1) Check syntax of instruction (Correct or wrong)
2) Symbol not found
3) wrong instruction
4) Duplicate symbol declaration

Teacher's Signature

5) Test the output of program by changing value of start & using pesudo opcode.

Conlcusion :— The intermediat data structure generated in pass - I of assembler are given as input to the pass - II of a assembler, processed by applying pass - II algoroithm of assembler and machine code is generated.

Teacher's Signature