SSC LAB ASSIGNMENT NO.2
STUDY ASSIGNMENT

NAME : KETAKI PATIL
ROLL NO : PA-17
BATCH : A1

Assignment Title: Study design of Pass 2 of Two Pass Assembler.

Aim: Study design of suitable data structure and algorithm of pass 2 of Two Pass Assembler pseudo machine.

Objective: Study suitable data structure and algorithm of pass 2 of Two Pass Assembler pseudo machine. Subset should consist of a few instructions from each category & few assembler directive.

Theory:

Assembler :

An assembler is a translator, that translates an assembler program into a conventional machine language program. Basically, the assembler goes through the program one line at a time and generates machine code for that instruction.

Why we need a Two Pass Assembler?

The one-pass assembler cannot resolve forward references of data symbols. It requires all data symbols to be defined prior to being used. A two-pass assembler solves this dilemma by devoting one pass to exclusively resolve all (data/label) forward references and then generate object code with no hassles in the next pass. If a data symbol depends on another and this another depends on yet another, the assembler resolved this recursively.

Advantages of Two Pass Assembler :

- One of the main advantages of Two-Pass Assembler is that many times the first pass of an extreme Two-pass assembler generates the output file which is then read by the second pass.

- The advantage of this is that first pass can record each line of input, along with that the next position of some or all lexemes of that line and some of the results of parsing.

Design of a Two Pass Assembler:

Assembler divide these tasks in two passes:

- Pass-1:

1. Define symbols and literals and remember them in symbol table and literal table respectively.
2. Keep track of location counter
3. Process pseudo-operations

- Pass-2:

1. Generate object code by converting symbolic op-code into respective numeric op-code
2. Generate data for literals and look for values of symbols

Working of Pass 2:

Pass-2 of assembler generates machine code by converting symbolic machine-opcodes into their respective bit configuration(machine understandable form). It stores all machine-opcodes in MOT table (op-code table) with symbolic code, their length and their bit configuration. It will also process pseudo-ops and will store them in POT table(pseudo-op table).

Various Data bases required by pass-2:

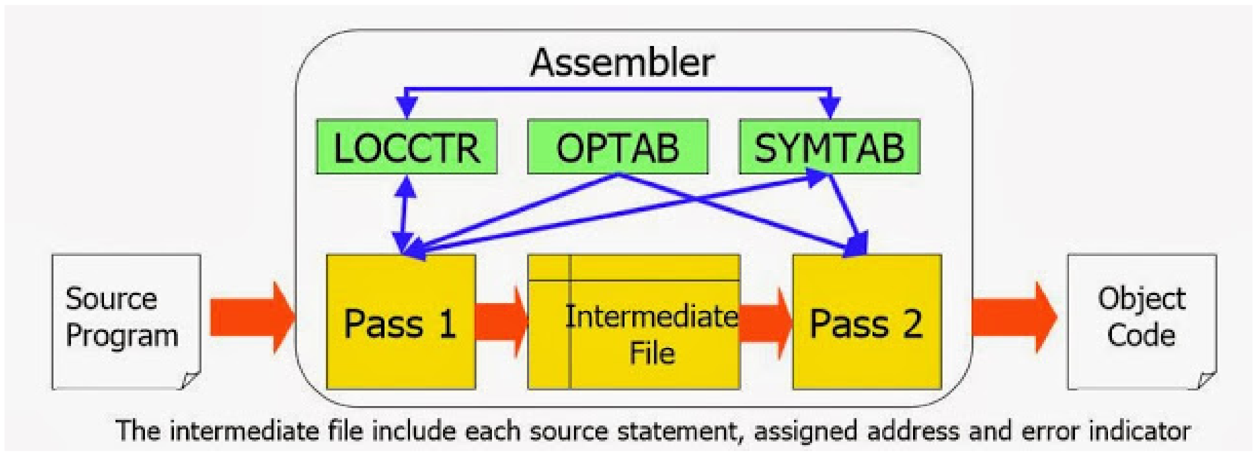
1. MOT table(machine opcode table)

2. POT table(pseudo opcode table)

3. Base table(storing value of base register)

4. LC (location counter)

TWO PASS ASSEMBLER FLOW :


Assembler
LOCCTR
OPTAB
SYMTAB
Source Program
Pass 1
Intermediate File
Pass 2
Object Code
The intermediate file include each source statement, assigned address and error indicator

Algorithm for Pass II :

```
1.  Code_area_address: = address of code area;
    loccntr: = 0;

2.  While next statement is not an END statement
    (a) Clear machine_code_buffer;
    (b) If a START or ORIGIN statement then
        (i) loccntr: = value specified in operand field;
        (ii) size: = 0;
    (c) If a declaration statement
        (i) If a DC statement then
            Assemble the constant in machine_code_buffer.
        (ii) size: = size of memory area required by DC/DS;
    (d) If an imperative statement
        (i) Get operand address from SYMTAB or LITTAB.
        (ii) Assemble instruction in machine_code_buffer.
        (iii) size: = size of instruction;
    (f) If size i= 0 then
        (i) Move contents of machine_code_buffer to the address code_area_address+loccntr;
        (ii) loccntr: = loccntr + size;
```

Write *code area* into output file.

OUTPUT :

I/C CODE :			
LC	OPCODE	OPERAND 1	OPERAND 2
-	(AD,01)	-	
100	(IS,04)	1	(S,1)
101	(IS,01)	2	(S,1)
102	(IS,04)	2	(S,3)
-	(AD,03)	-	
101	(IS,04)	2	(S,1)
102	(DL,02)	-	(C,5)
107	(DL,01)	-	(C,5)
-	(AD,02)	-	

M/C CODE :	
LC	(CLASS,OPCODE)
-	(AD,01)
100	(IS,04)
101	(IS,01)
102	(IS,04)
-	(AD,03)
101	(IS,04)
102	(DL,02)
107	(DL,01)
-	(AD,02)

SYMBOL TABLE :

SYMBOL TABLE:			
SYMBOL ID	SYM NAME	ADDRESS	LENGTH
1	A	102	5
2	L1	101	1
3	B	107	1

INPUT :

```
START 100
MOVER AREG A
L1 ADD BREG A
MOVER BREG B
ORIGIN L1
MOVER BREG A
A DS 5
B DC 5
END
```

Conclusion: The function of Pass II in an assembler are studied.