Ritom Gupta, PA-25, Panel 1, Batch B2

School of CET

System Software and Compiler lab

Assignment No.2

TY BTech CSE

Assignment Title: Design of Pass 2 of Two Pass Assembler.

Aim: Design suitable data structure & implement pass 2 of Two Pass Assembler pseudo machine.

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

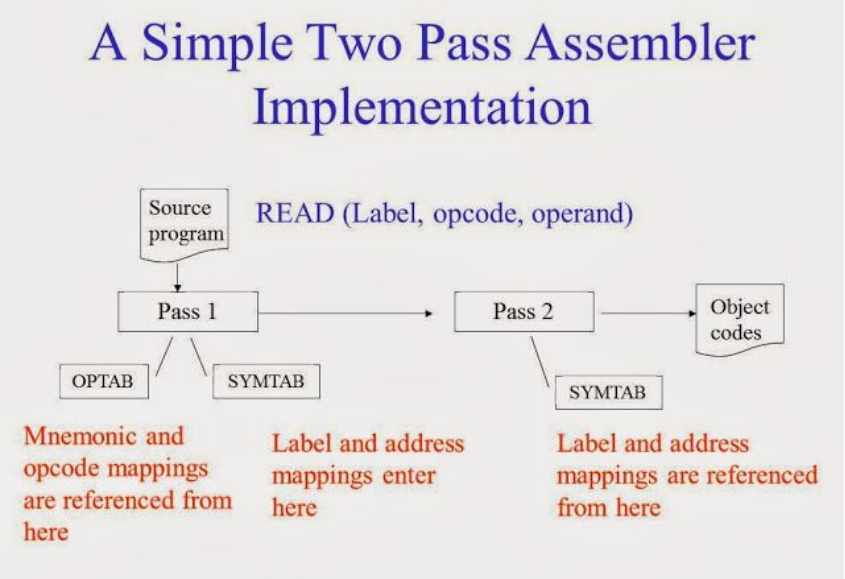
Theory:

Design of a Two Pass Assembler:

Two-pass assembler: Assemblers typically make two or more passes through a source program in order to resolve forward references in a program. A forward reference is defined as a type of instruction in the code segment that is referencing the label of an instruction, but the assembler has not yet encountered the definition of that instruction.

Pass 1: Assembler reads the entire source program and constructs a symbol table of names and labels used in the program, that is, name of data fields and programs labels and their relative location (offset) within the segment.
Pass 1 determines the amount of code to be generated for each instruction.

Pass 2: The assembler uses the symbol table that it constructed in Pass 1. Now it knows the length and relative of each data field and instruction, it can complete the object code for each instruction. It produces .OBJ (Object file), .LST (list file) and cross reference (.CRF) files.



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) s*ize:* = 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;*

3. Write *code area* into output file.

Ritom Gupta, PA-25, Panel 1, Batch B2

Input: Symbol table and Intermediate code generated by Pass I.

Output:

1. Final Output (After Pass II)

Address (LC value)	Op-code	Operand 1 (Value/Address)	Operand 2 (Value/Address)

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

Platform: Linux (JAVA)