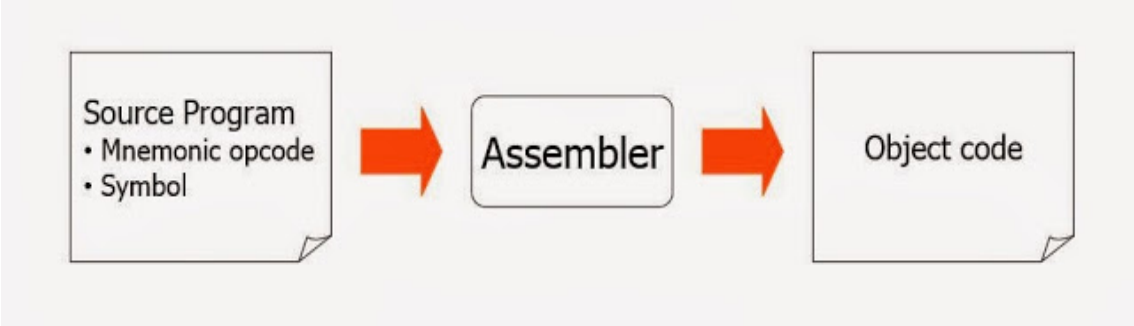
Course Code: CSL602	Course Name: SPCC LAB
Class : TE-CO B-3	Date: 18/02/2021
Roll no : 18CO63	Name : SHAIKH TAUSEEF MUSHTAQUE ALI

Experiment :02

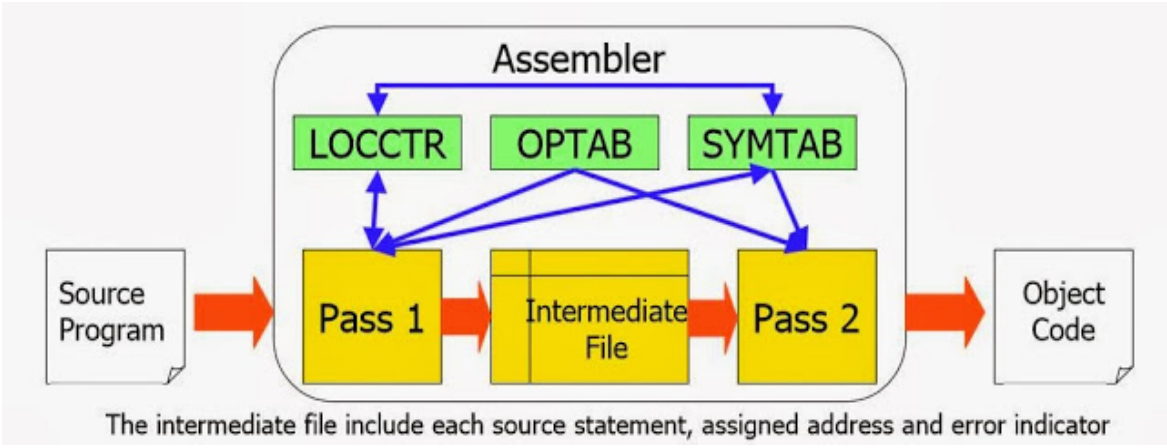
Aim: To study Two Pass Assembler.

Theory:

Assembly language is converted into executable machine code by a utility program referred to as an assembler; the conversion process is referred to as assembly, or assembling the code.



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. Then the assembler proceeds to the next instruction. In this way, the entire machine code program is created.



Assembler Directives

Assembler directives are pseudo instructions

They will not be translated into machine instructions.

They only provide instruction/direction/information to the assembler.

Basic assembler directives :

START : Specify name and starting address for the program

END : Indicate the end of the source program.

EQU : The EQU directive is used to replace a number by a symbol. For example: MAXIMUM EQU 99. After using this directive, every appearance of the label "MAXIMUM" in the program will be interpreted by the assembler as the number 99

Main Data Structures

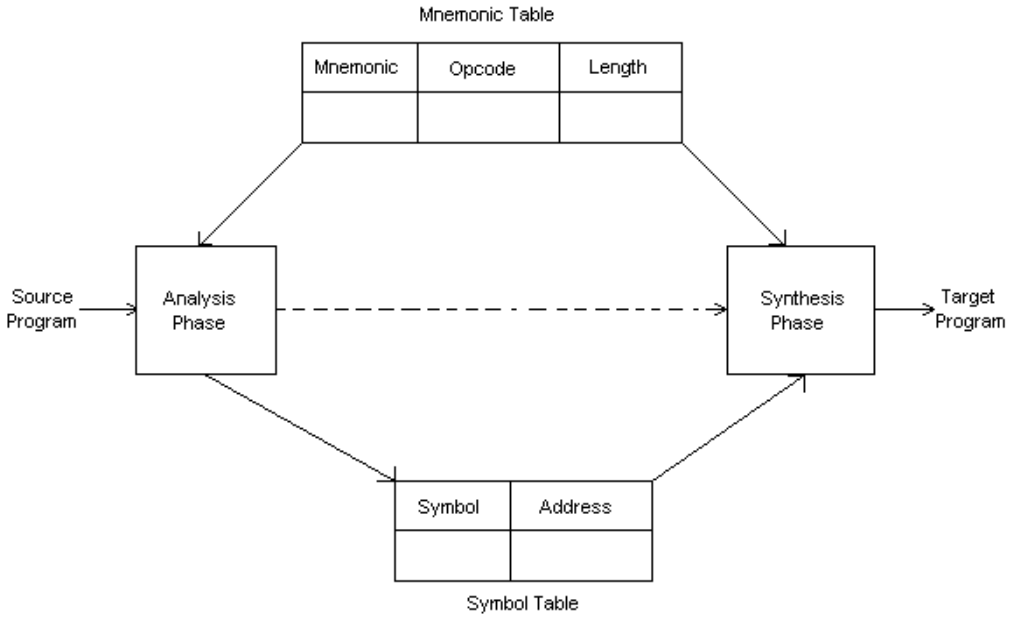
MOT (Machine Opcode Table) , ST (Symbol Table) , POT (Psudo Opcode table) , LC (Location Counter)

Instruction formats

Addressing modes · Direct addressing (address of operand is given in instruction itself) · Register addressing (one of the operand is general purpose register) · Register indirect addressing (address of operand is specified by register pair) · Immediate addressing (operand - data is specified in the instruction itself) · Implicit addressing (mostly the operation operates on the contents of accumulator)

One-pass assemblers

A one pass assembler passes over the source file exactly once, in the same pass collecting the labels, resolving future references and doing the actual assembly. The difficult part is to resolve future label references (the problem of forward referencing) and assemble code in one pass



Forward reference in one pass assembler

Omits the operand address if the symbol has not yet been defined Enters this undefined symbol into SYMTAB and indicates that it is undefined Adds the address of this operand address to a list of forward references associated with the SYMTAB entry When the definition for the symbol is encountered, scans the reference list and inserts the address.

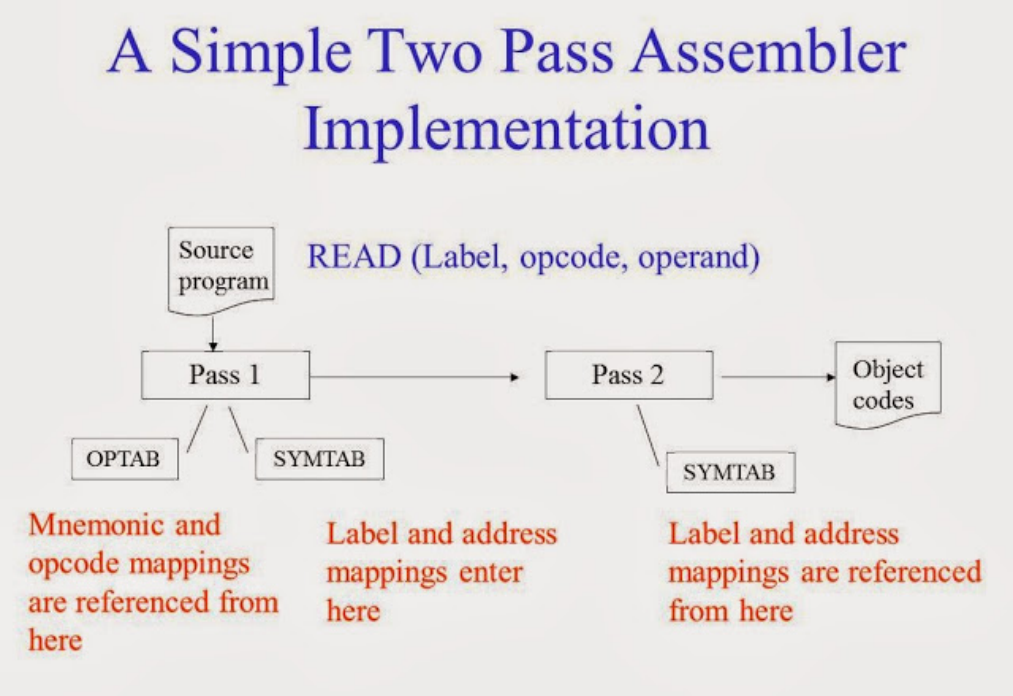
At the end of the program, reports the error if there are still SYMTAB entries indicated undefined symbols.

Two-pass assemblers

The two pass assembler performs two passes over the source program.

In the first pass, it reads the entire source program, looking only for label definitions. All the labels are collected, assigned address, and placed in the symbol table in this pass, To assign address to labels, the assembles maintains a Location Counter (LC).

In the second pass the instructions are again read and are assembled using the symbol table. Basically, the assembler goes through the program one line at a time, and generates machine code for that instruction.



Conclusion:

With the help of this assignment we get information about Two Pass Assembler.