# **Algorithm Design**

In our design of a Two-pass assembler, we are interested in generating machine codes from a given set of assembly language codes. Tasks performed by the assembler in the two passes are:

**Pass 1 – Define symbols and literals.**

* The length of machine instructions would be determined.
* The track of location counter (LC) would be kept.
* Symbol values must be remembered until pass 2.
* Some pseudo-ops, if present in the card, would be processed.
* The literals, if present in the card, would be remembered.

**Pass 2 – Generate object program**

The second pass reads the program again from the beginning. Each instruction is translated into the appropriate binary machine code. Translation includes the following operations:

* Read operation from the remembered symbol values.
* Generate Instructions.
* Data for DS, DC and literals would be generated.
* Remaining pseudo-ops which remained unprocessed from pass 1 would be processed.

After studying the tasks classification for the two passes, it is necessary to establish a database that will work with our assembler.

**Pass 1 Database**

* Input source program.
* A Location Counter for keeping the track of instruction’s location.
* A Machine Operation Table for indicating the symbolic mnemonic for each instruction and its length.
* A Pseudo-Operation table for indicating the symbolic mnemonic for each pseudo-op in Pass 1.
* A Symbol Table to store label and its value.
* A Literal Table to store literal and assigned location.
* A copy of input source file to be used in pass 2 can be stored as a File pointer.

**Pass 2 Database**

* Copy of Input Source Program in pass 1.
* Location counter.
* A Machine-Operation Table (MOT) that indicates Symbolic mnemonic, Length, Binary machine op-code & format (RS, RX, SI).
* A Pseudo-Operation Table (PoT) that indicates symbolic mnemonic and action to be taken for each pseudo-op.
* A Symbol Table (SYMTAB) which is prepared during pass 1 containing each label and its value.
* A Base Table indicates the registers which are currently specified as base register by USING pseudo-op and its contents.
* A workspace which is used to hold each instruction as its various parts (eg. Binary op-code, register fields, length fields, displacement fields) are being assembled.
* A workspace, PRINT, used to produce a printed listing.
* A workspace, PUNCH CARD, used prior to actual outputting for converting the assembled instruction into a format suitable for loader.
* An output program in a format suitable for the loader.

The design of the algorithm for the working of Two-pass Assembler is presented in the form of flowcharts.