**Assembler - Documentation**

# Assumptions & Syntax

The assembler is based on a 12-bit accumulator structure. It uses the two-pass algorithm to convert the assembly code to object code.

The following are the specifics and assumptions made in the Assembler

1. Only Single Line Comments are supported.
   * These comments can be in separate lines and can either begin by “//” or if they begin by “/\*” then they must end by “\*/”. Comments are not processed by the assembler
   * Same Line Comments Begin by ‘#’. The ‘#’ Character should have a single whitespace between the Code and the Comment.
2. Instructions need to be stored in a separate text file with the name “input.txt” in the same folder.
   * If the file is not found the Assembler throws an Error.
3. As it is based on a 12-bit Accumulator Architecture it can store a maximum value of 4095 in the accumulator.
4. Word length is taken to be 24-bit (Including the OPCODE Bits). Addresses 0-4095 have been reserved for variables and Literals
5. Addresses are only taken in numbers with base 10.
6. All labels used in the accumulator must start with an Uppercase ‘L’ followed by a number of choice.
7. All variables name must contain only alphabets.
8. Assembler **is** Case – Sensitive and takes into account the number of whitespaces in the instruction. Any deviation from the specified input format would lead to Errors.
9. As soon as an Error is encountered, the Assembler reports the error and exits the execution.
10. If Addresses are entered as Variables, then they need to be Declared.
11. Variable Declarations are done at the end of the Instruction Codes.

# Working

The Assembler is coded in Python 3.7. The code contains 3 Dictionaries one each for the Instruction Table, Symbol Table and Literal Table. A set has also been used to keep track of variables.

There are two major methods for the first and second pass respectively. The first pass uses many functions which have been implemented for checking and verifying Opcodes, Labels & Variables etc. The first pass makes use of Location Counter Variables to assign addresses to the Instructions

The Second Pass then reads the values from the created tables converts them into Object Codes and stores them in a text document named ‘Result.txt’

Forward Referencing is handled by checking the set Variable Logs i.e. if the set is not empty at the end of the first pass, then it would mean that, that Variable hasn’t been assigned an address.