TWO PASS ASSEMBLER

- **How to run on terminal?**
  1. Change the directory to the place where the code file is present.
  2. Enter command: `javac Code.java`
  3. The above command will compile the code. Then enter : `java Code`
  4. Your code will get executed.
  5. It will display the errors(if any) , the opcode table, literal table and the symbol table along with the memory addresses.

- **Assumptions-**
  1. The START statement should always be present and the value given by the START statement to initialize the location counter must be a valid integer.
  2. Length of operand is 1 , eg: A , M . Length of assembly opcode is 3. However, if the statement is declarative(DC), the length of assembly opcode is 2.
  3. We have used DC and not DW . DC is used to declare operands. The operand should be valid.The operand should be declared with a value that must be a valid integer. If the instruction is declared using opcode as it's first term (no label) then a space should be given and then the instruction should be written.
  4. Length of every instruction is set to 1. Each instruction is written with a single space gap between the contents of the instruction.
  5. Instructions are written in simultaneous lines without any gap in between them, except the comments.
  6. There is no special character used after using a label, just a space.
  7. Comment line is not considered in updation of the location counter. Comment is always written in a new line and is always preceded by //.
  8. Comment and Start Line are not included in the Machine Code.
  9. Division opcode assembly format: DIV operand,REG1,REG2
  10. Operand can be a variable or a literal. The format of using a literal is ='*value*' , where *value* is the numerical value of the literal.
  11. Register name should always begin with 'R'.
  12. If an operand is defined more than once , then its first declaration value will be taken into account and error will be given for the remaining ones.

- **First Pass-**
  The code is scanned for the first pass. Operands and Labels are put in the Symbol Table, opcodes in the Opcode table and Literals in the literal table along with their values (if any), location counter and memory address. Errors are printed on the console at the end of first pass along with the three tables mentioned above. An ArrayList named IC is used to store strings in the format {Opcode + Assembly Opcode + length + line} to help generate the machine code.

- **Second Pass-**
  The strings written in IC are sequentially read. The machine code gets generated in a 12-bit instruction format using the opcode in IC that is generated after the first pass and the memory address stored in symbol/literal table.

- **Assembler Output-**
  After the first pass, the following tables will be printed on the console depending on the assembly code text file given as an input-
  1. Opcode table - consisting of assembly opcode and opcode.
  2. Symbol table-
     - If the symbol is a constant- it's symbol, value and memory address will be printed.
     - If the symbol is a label- it's symbol and memory address will be printed.
  3. Literal table - consisting of the literal and it's memory address.

  After the first pass, an ErrorFile.txt file will be generated which will contain the errors(if any).

  After the second pass, MachineCode.txt file will be generated.

- **Files-**
  1. Code.java - Contains the well commented code of the two pass assembler.
  2. AssemblyCode.txt - It contains the assembly code to be given as an input to the assembler.

3. MachineCode.txt - It contains the machine code of the assembly language program given as an output. The instructions will be a 12 bit instruction with 4 bit opcode and 8 bit memory address.
4. ErrorFile.txt - It contains the errors caught(if any).

- **Errors Handled -**
    1. Opcode is invalid.
    2. End(STP) statement is missing.
    3. Symbol has been defined more than once.
    4. A symbol has been used but not defined.
    5. An opcode is not supplied with enough operands.
    6. An opcode is supplied with too many operands.

Note - All errors will be printed at the end of first pass if the code is run on the terminal and written in the ErrorFile.txt and no Machine code will be generated for them (as it is known to us that all the errors will be skipped for invalid statements and code will be generated only for the valid ones).