1.Project Overview:

The assembler is designed to read an assembly language source file (input.asm), process its contents, and generate two output files: listing\_file and load\_file. The assembler is packaged as a JAR file (Computer\_Architecture.jar), which is executed via the command line or IDEA.

The assembler uses two passes:

The first pass constructs the symbol table by parsing labels and addresses.

The second pass generates machine code and produces output files.

2.Core Components:

The assembler consists of the following key components:

1)Main Class:

The entry point of the assembler.

Reads the input.asm file, which is hardcoded in the main method.

Coordinates the parsing and processing of the assembly file.

2)First Pass (firstPass):

Reads the assembly source file (input.asm) and builds the symbol table (labelTable).

Labels and addresses are stored in labelTable.

Handles LOC directive to modify the current location.

3)Second Pass (secondPass):

Reads each line of the assembly source file, processes instructions, and converts them to machine code using the opcode table (opcodes).

Machine code and listing data are stored in output and list, respectively.

4)Instruction Processing (processLine):

Identifies the type of instruction based on the mnemonic (e.g., LDR, STR, JCC, etc.).

Calculates the effective address (ea) using index registers, immediate addressing, or direct addressing.

Invokes appropriate helper functions to handle specific instructions like MLT, DVD, JMA, etc.

5)Register and Memory:

General registers (register\_general) and index registers (register\_index) are simulated as arrays.

Memory is simulated using the data array

6)Error Handling:

The assembler checks for several errors during instruction processing, such as invalid register usage or division by zero.

Errors are reported in the console and logged in the listing\_file