# Two Pass Assembler

## Contents



**CREATED BY**

Manak Bisht 2018340

Manan Jain 2018294

# 1. Usage

As long as you have the JRE (Java Runtime Environment) installed on your system, you can run the assembler from the following command ~

```
$ java Main filename
```

Make sure that you have all the class files available, namely, '*AssemblerToOpcode*', '*Main*', '*Opcode*', '*Symbol*' in the same directory. Since the assembler is coded in java, it is cross-platform.

To invoke the assembler from a single command, anywhere in the filesystem, you could add it to the PATH environment variable. This will vary across platforms.

To compile multiple files at once,

```
$ java Main file1 file2 file2 *
```

In case you would like to compile the assembler to bytecode from source,

```
$ javac Main.java
```

Then you will have the class files. Run as explained above.

*Note:* You will need to have the JDK (Java Development Kit) installed and java & javac added to the PATH.

An installation guide for the JDK & JRE can be found here,

https://docs.oracle.com/en/java/javase/12/install/overview-jdk-installation.html#GUID-8677A77F-231A-40F7-98B9-1FD0B48C346A

# 2. Syntax & Assumptions

The assembler works like a typical two pass assembler. Building a symbol table in the first pass and generating the corresponding object code in the second pass.

The assembler ignores all whitespaces (spaces, tabs, etc). You are free to use any amount of spacing.

**An instruction has the word size of 12 bits, 4 bits for the opcode and 8 bits for the operand.** A typical instruction is of the format ~

```
Label(optional) : Opcode Operand(if any)
```

You can add comments using **double-slash (//)**. They are ignored by the assembler.

```
INP 11 //Assuming the number entered is positive
```

Variables can not be declared. However, input can be taken from the terminal using the 'INP' opcode and you can refer to memory addresses in your program. The assembler normalises their addresses and appends the variables at the end of the output file.

Registers R1 & R2 will be handled by the linker in the linking process. Referring to them directly or trying to access them is not allowed.

For information on particular opcodes, you may refer to the table provided at the end.

# 3. Errors Reported

## 1) File Not Found

The assembler throws this error when no file with the provided name exists. This may be due to a typo or the file having being moved.

If multiple files were supplied to the assembler and it cannot find a particular file, it prints the above error message and moves on to the next file.

## 2) No File Supplied

When the assembler is run without providing any files i.e. the command is called as is, it raises the above error. To replicate the error, run

```
$ java Main
```

## 3) Invalid Token

An invalid token error is raised when any of the unpermitted characters are used anywhere. Unpermitted characters include **all special characters except underscore(_)** which can be used in label names **and double-slash(//)** which is reserved for comments.

## 4) More Arguments Supplied

When an opcode is supplied with more arguments than it requires. For eg, if a unary operator is provided with more than one argument.

```
INP 42

ADD 42

Line 2, 'ADD 42', More arguments supplied
```

Or if an opcode which does not take any argument is supplied one,

```
CLA 67
```

```
Line 1, 'CLA 67', More arguments supplied
```

## 5) Less Arguments Supplied

Similar to the above, when an opcode is supplied with an insufficient number of arguments.

```
LAC 125
```

```
SUB
```

```
Line 2, 'SUB', Less arguments supplied
```

## 6) Invalid label

The following error is raised when an assembly opcode is used as a label. For example ~

```
CLA : BRN
```

```
Line 1, 'CLA : BRN', Invalid label
```

## 7) Invalid Opcode

An invalid opcode error is displayed for unrecognised opcodes.

```
CLA
JMP 99
Line 2, 'JMP 99', invalid opcode
```

## 8) Label Not Declared

The error is raised, if your code uses an undeclared label. For instance ~

```
CLA
INP 170
INP 171
L1: DSP 170
LAC 171
BRN L1
STP

Line 6, 'BRN L1', Label not declared
```

# 4. Instruction Set

| Opcode | Meaning | Assembly Opcode | Operand |
|---|---|---|---|
| 0000 | Clear accumulator | CLA | None |
| 0001 | Load into accumulator from address | LAC | Unary |
| 0010 | Store accumulator contents into address | SAC | Unary |
| 0011 | Add address contents to accumulator contents | ADD | Unary |
| 0100 | Subtract address contents from accumulator contents | SUB | Unary |
| 0101 | Branch to address if accumulator contains zero | BRZ | Unary |
| 0110 | Branch to address if accumulator contains negative value | BRN | Unary |
| 0111 | Branch to address if accumulator contains positive value | BRP | Unary |
| 1000 | Read from terminal and put in address | INP | Unary |
| 1001 | Display value in address on terminal | DSP | Unary |
| 1010 | Multiply accumulator and address contents | MUL | Unary |
| 1011 | Divide accumulator contents by address content. Quotient in R1 and remainder in R2 | DIV | Unary |
| 1100 | Stop execution | STP | None |