**POC Planning**

Overall Workflow

The compiler will get a txt file with a single basic calculator entry (Example: 5+5).

# Tokenization:

The compiler would first of all **Tokenize** the string file into these tokens (Which would be held in an enum called TOKENS)

* INT
* LEPREN
* RPREN
* ADDITION SIGN
* MULTIPLICATION SIGN
* DIVISION SIGN
* SUBSTRECTION SIGN

The output of this process is this Data structure:

Vector < Pair <Literal (String), TOKENS >>

# Syntax check

In this process we need to check the following conditions:

* For every LPREN there is a fitting RPREN
* There is only one binary operation between two numbers
* There could only be – or + before a number

# Optimization

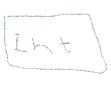
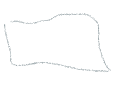
++++---++ -> -

# AST

In this process we would build a tree that is made from numbers and an operation. Since we are dealing with binary operation, the root of each expression would be its operation, and the nodes of that root would be the operands of that expression.

Every tree would look like this:

First operation



# Converting AST to ASM :

We first of all put 0 in ax

And now based on the sign expression put together the tree in a recursive way