Example program:

0-DEC X

1-DEC Y

2-DEC Z

3-DEC A

4-LDI 10

5-ST X

6-LDI 5

7-ST Y

8-LDI -1

9-ST Z

10-LDI 0

11-ST A

12-JZS 42

13-LDA A

14-LDB X

15-ADD

16-ST A

17-LDA Z

18-LDB Y

19-ADD

20-ST Y

21-JMP 24

22-HLT

This program performs multiplication of 10 with 5, to change these variables change in line numbers, 4 and respectively. I have entered line numbers to show how jump instructions work.

I have included another program in the input file in the submitted project

For memory initializaton i have used opcode for each instruction. The opcodes for instructions are

DEC - 0

LDA -1

LDB -2

LDI -3

ST -4

XCH - 5

JMP - 6

JZS - 7

JVS - 8

ADD - 9

HLT -10

So while reading from the editor, i first parsed the complete text. While parsing this if i encounter any instuction i store its opcode, if i encounter a value i store it or if i encounter a symbol i store its respective memory address.

The moery for my program would be:

0,-1,0,-1,0,-1,0,-1,3,10,4,1,3,20,4,3,3,-1,4,5,3,0,4,7,7,42,1,7,2,1,9,4,7,1,5,2,3,9,6,24,10

Now i read the memory and then depending on what opcode i encounter i call that class in the command pattern.

Important things to consider:

->**Jump instructons are followed by the memory location but not the line number in the code.**

The above is the basic implementation, depending on the various test cases i modeled the code to work correctly.

Environment used to run the code: Windows 10.