Assignment Details

Project 6: Stack-Based Expression Evaluation

A detailed writeup can be found here

You will convert a C# program that evaluates mathematical expressions using a stack (postfix notation) into MASM assembly code. This project includes implementing stack operations and parsing strings

Instructions

- Write a C# program that evaluates a postfix expression using a stack.
- Convert the program to MASM, implementing push and pop stack operations and parsing input strings.
- Evaluate the expression and display the result.

MASM Implementation Notes

- Implement a stack in memory, managing pushes and pops with stack pointer manipulations. - Parse the postfix expression and execute operations based on stack state.

<—- Running source code —->

Running C# source code

- The easiest way to run the c# source code is to use the dotnet cli
  - execute `dotnet run` in the directory with the cs file (postfix-evaluation-cs)

Running ASM source code

- There are 2 asm files… One contains the minimal source with no dependencies, the other contains extras for printing outputs and timing for measuring latency and throughput.

- The easiest way to run the main.asm (minimal) source code is to setup visual studio 2022 in a windows enviornment (I use a virtual machine since my laptop runs linux)

  - set breakpoints at points of interest and step through the code using the built in debugger in the env.
  - This should be nice for grading source code

- There is no easy way to run extra.asm (extras) source code. There is an including powershell script that will assemble and link the source code. However this makes the assumption that you have the masm-32 sdk installed which can be gotten from here. This makes a directory in 'C:/masm32' that contains an executable assembler in '/bin' (ml.exe) an executable linker in /bin (link.exe) as well as libraries like 'kernal.inc'. I can't figure out how to get Visual Studio 2022 to use these executables so the script assembles and links the source file and necissary libraries.

Team Members Tim Price