Ömer Faruk Ünal
2019400048

Eren Dönmez
2019400021

Assembly Calculator

Usage

1. The code is compiled with the command `a86 main.asm`.
2. The executable is run with the command `main`.
3. The program then accepts input from the user.

Description

- Given hexadecimal quantities as an input, this program evaluates and calculates the result of the given mathematical operations in A86 Assembly language.
- Supports addition, multiplication, integer division, bitwise XOR, bitwise OR and bitwise AND operations.

Project Outline

This project was written in the A86 Assembly language and consists of one `.asm` file.

- `main.asm`: This is the main and only file in this project. Reads input, evaluates given postfix expression, and gives the result as an output.

Algorithm

There are two main containers of data: `curr_num`, which stores the value of the current number in the memory, and the stack, which is used while evaluating the results of the operations in the input. The code evaluates the input character by character. Thus, new characters are continuously read from the input.

- If the character read is a digit (including 0-9 and A-F), that digit is appended to the end of `curr_num`, by multiplying `curr_num` by 10h and adding the new digit to the result.
- If the character read is a plus sign (+), then the top two elements of the stack are popped, addition is applied, and the result is pushed to the stack.
- If the character read is an asterisk (*), then the top two elements of the stack are popped, multiplication is applied, and the result is pushed to the stack.
- If the character read is a division sign (/), then the top two elements of the stack are popped, integer division is applied, and the result is pushed to the stack.
- If the character read is an ampersand (&), then the top two elements of the stack are popped, bitwise AND operation is applied, and the result is pushed to the stack.
- If the character read is a vertical bar (|), then the top two elements of the stack are popped, bitwise OR operation is applied, and the result is pushed to the stack.
- If the character read is a caret (^), then the top two elements of the stack are popped, bitwise XOR operation is applied, and the result is pushed to the stack.
- If the character read is a space which signifies the end of a number, then that number is pushed to the stack.
- If the character read is a carriage return, it signifies the end of the input, thus the code moves on to printing the output. At this point, if no space character was encountered since the beginning of the input, this means that the input consists of a single number. If this is the case, that number is pushed to the stack. After that, the value at the top of the stack should be the output value. That value is

separated into its digits by continuously dividing by 10h and getting the remainder. Finally, the digit values are converted to their corresponding ASCII values and printed to the terminal.

Used Technologies and Structures

Since A86 assembly language works with Windows XP, both team members installed Windows XP on their devices via virtual machine. Apart from that, the Assembly A86 Manual, the D86 debugger (and its manual), and the ASCII table were used a lot in this project.