Y86 Assembler and Processor

OCS35 Assignment, Trisha Vidyanand and Jake North

About the Assembler

Running

- Main program code is in assembler.js
    - The Assembler class takes a code string in its constructor, and calling the assemble method will return a ByteList of the memory output. To see this in hex form, call the toHex method of the ByteList.
        - Example: (new Assembler("irmovq 5, %rax")).assemble().toHex()
    - If the assembler encounters a syntax error, it will throw a JS error with a message describing the problem and line number.
- The web interface is in index.html.

Assumptions

- All ints are 8 bytes
- Align on 8-byte boundaries
- All instructions padded to 16 bytes
- Numbers are little-endian
- since registers are 8 bytes, rrmovq instead of rrmovl etc
    - and registars are %rax instead of %eax
    - and addq instead of addl, etc

About the Processor

Running

- Main program code is in processor.js

Assumptions

- Assembly takes the form of padded output by our Part 1 assembler

Conventions

- All processor state values are in decimal
- 2's compliment overflow cannot occur, since JS BigInts are used internally
- Addresses over 10,000 are illegal, since seeking addresses that large is usually the result of an error and may crash the proccessor. While our ByteArray class is designed to scale automatically, scaling it to an absurd value would reqire enough actual memory to reach the maximum pointer.

Citations

- Syntax highlighting in the web interface is done using CodeMirror.
- CSS styles for the web interface are derived from a stylesheet that I use for personal projects