Computer Architecture

The computer is six bits, meaning it can use 63 possible numbers for its operations. Click here if you do not know what binary is, as this is integral to programming the computer. The Computer has 4 bytes; while this is very small, it's also a question of performance. If you were to add more bytes, your *actual* computer would not be able to simulate the in-game computer. The clock speed is also 1hz, but can be increased, but not recommended. Also, do to the complexities of this computer, it's prone to failure, but it's most likely human error. If you have questions, please join the discord in the steam description of the contraption.

Bus

The Bus is the standard collection of data that all modules connect into; for example, the ALU could store the sum of registers A and B and then output it onto the RAM to be stored for further use.

Ram

The ram is 4 bytes, as mentioned before. The memory is divided into four segments, each with 8 bits that they can contain. To access registers, sending 00 to the input will access segment 1, sending 01 to the input will access segment 2, sending 10 to the input will access segment 3, and sending 11 to the input will access segment 4. Conveniently, I have neatly made each segment access respond to one of the assembly commands.

When accessing a specific segment, you can freely load data onto the ram segment and also export the data onto the bus.

Registers

The registers are essentially identical to the RAM, except they have the option to be loaded onto the ALU. Also, they don't have to be switched through and can be enabled or disabled by individual assembly commands.

Alu

The ALU adds the registers together, so if register A has 000001 stored and register B has 000001 stored it will output a 000010 onto the bus.

Screen

The screen works by individually turning it on and off each pixel; while slow and daunting, this is the easiest method I could find without breaking the activation signal limit. As

an example, if you wanted to turn pixel 1 on, you would write 000001 onto the bus, and that would turn pixel 1 on, and 111111 would turn pixel 63 on.

Assembly Language Guide

There are 19 commands in total.
Ram 1: Accesses the first ram segment (Remember to deactivate other segments before changing, also, if no ram is being accessed, it will access segment 0 of the RAM)
Ram 2: Accesses the first ram segment
Ram 3: Accesses the first ram segment
Load Screen: Loads the contents of the bus onto the screen
Load A: Loads the contents of the bus onto Register A
Load B: Loads the contents of the bus onto Register B
Ram Load: Loads the contents of the bus onto the currently selected RAM segment.
Enable A: Enables the output of Register A onto either the ALU or bus
Enable B: Enables the output of Register B onto either the ALU or bus
Ram Enable: Enables the output of the currently selected RAM onto the bus
Enable Alu: Enables the output of the ALU onto the bus
Switch A: Switches between register A outputting onto the bus or the ALU. (Enable must also be enabled.)
Switch B: Switches between register B outputting onto the bus or the ALU. (Enable must also be enabled.)
B1: Outputs 000001 on the bus
B2:Outputs 000010 on the bus
B3:Outputs 000100 on the bus
B4:Outputs 001000 on the bus
B5:Outputs 010001 on the bus
B6:Outputs 100000 on the bus
Make sure no more than one enable is activated at a time; otherwise, the bus will overwrite and break the computer. Also, make sure a enable command isn’t activated white one of the B commands is active. B commands can be combined to make any number on the bus from 63-0