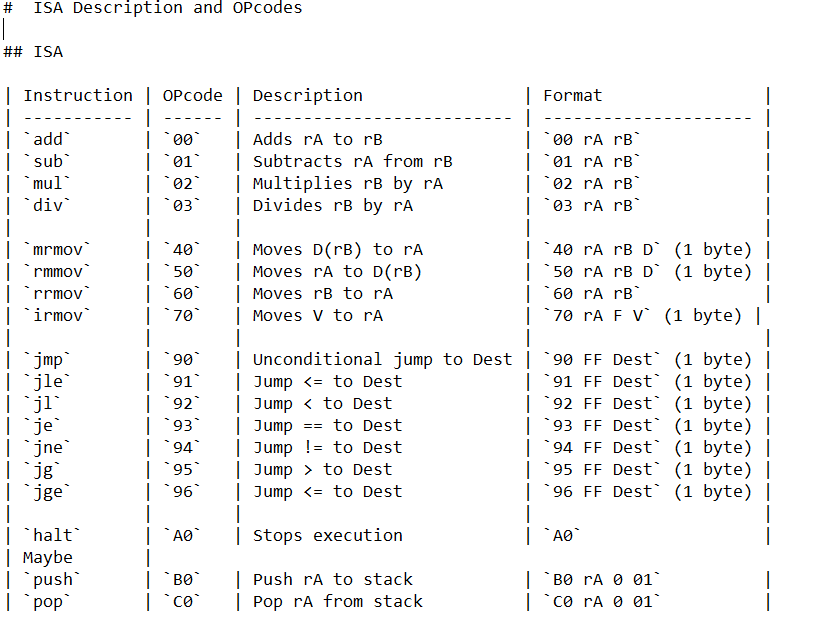
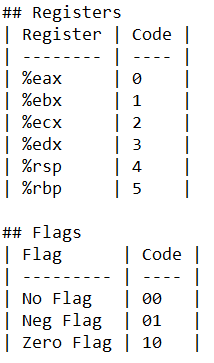
Group 19

Final Project Report

Problem 1)





Problem 2)

The assumptions for Matrix A, B, and C are that they are all 2x2 matrices. The corresponding values from the first value in matrix A and B should be added to produce the first value in matrix C. Continue this for the other 3 values. This produces matrix C. Eax holds the value of matrix A and is added to ebx which holds the matrix B value. The resulting ebx value is pushed for the corresponding matrix C value.

Main:

#\*\*\*\*\*\*\*Setting up base and stack pointer, filling stack

# with matrix values \*\*\*\*\*\*\*\*\*\*\*

irmovl $0xFC, %esp irmovl $0xFE, %ebp irmovl $0x05, %eax pushl %eax

irmovl $0x02, %eax pushl %eax

irmovl $0x04, %eax pushl %eax

irmovl $0x03, %eax pushl %eax

irmovl $0x01, %eax pushl %eax

irmovl $0x07, %eax pushl %eax

irmovl $0x09, %eax pushl %eax

irmovl $0x02, %eax pushl %eax

#Add first element and push to stack

irmovl $0xF8, %ecx mrmovl (%ecx), %eax irmovl $0xe8, %ecx mrmovl (%ecx), %ebx addl %eax, %ebx pushl %ebx

#Add second element and push to stack

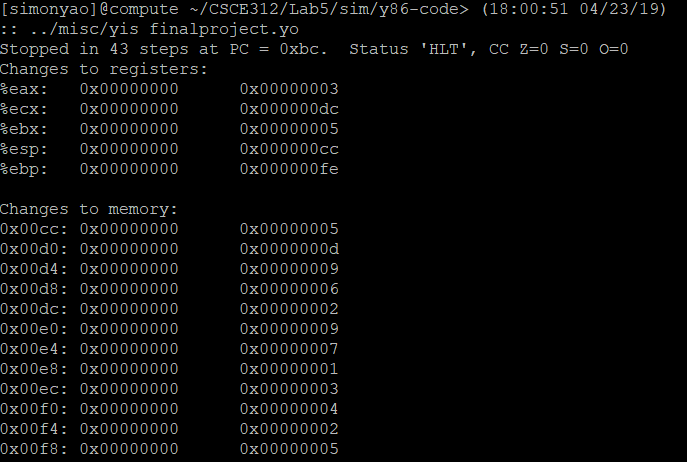
irmovl $0xF4, %ecx mrmovl (%ecx), %eax irmovl $0xe4, %ecx mrmovl (%ecx), %ebx addl %eax, %ebx pushl %ebx

#Add third element and push to stack

irmovl $0xF0, %ecx mrmovl (%ecx), %eax irmovl $0xe0, %ecx mrmovl (%ecx), %ebx addl %eax, %ebx pushl %ebx

#Add fourth element and push to stack

irmovl $0xec, %ecx mrmovl (%ecx), %eax irmovl $0xdc, %ecx mrmovl (%ecx), %ebx addl %eax, %ebx pushl %ebx



Our ISA program:

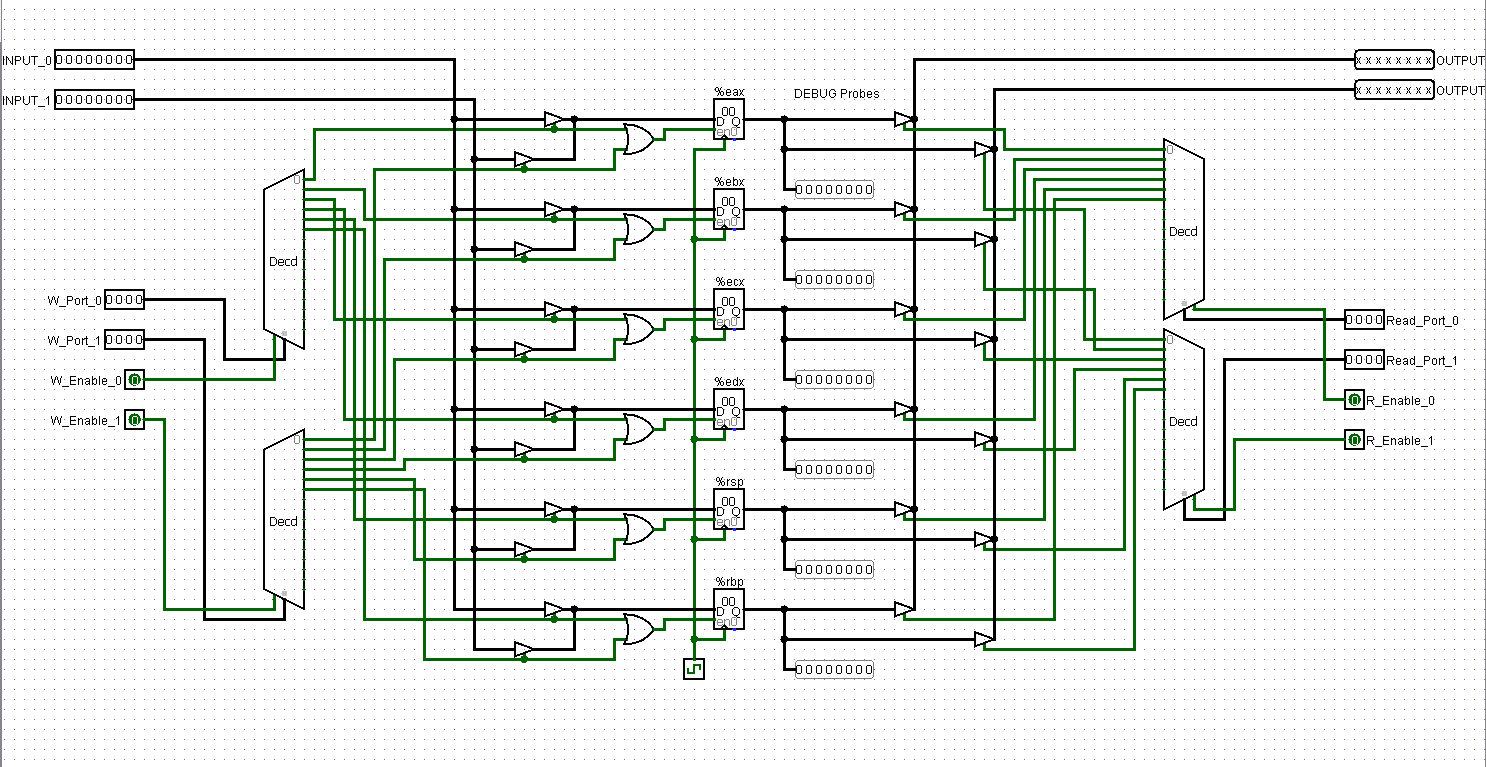
irmov FF -> %ebp 70 5 0 FE irmov F7 -> %esp 70 4 0 F7 irmov FE -> %ecx 70 2 0 FE mrmov %ecx -> %eax 40 0 2 00 irmov FA -> %ecx 70 2 0 FA mrmov %ecx -> %ebx 40 1 2 00 add %eax + %ebx 00 0 1 00 push %ebx B1 1 0 01

irmov FE -> %ecx 70 2 0 FD mrmov %ecx -> %eax 40 0 2 00 irmov F9 -> %ecx 70 2 0 F9 mrmov %ecx -> %ebx 40 1 2 00 add %eax + %ebx 00 0 1 00 push %ebx B1 1 0 01

irmov FC -> %ecx 70 2 0 FC mrmov %ecx -> %eax 40 0 2 00 irmov F8 -> %ecx 70 2 0 F8 mrmov %ecx -> %ebx 40 1 2 00 add %eax + %ebx 00 0 1 00 push %ebx B1 1 0 01

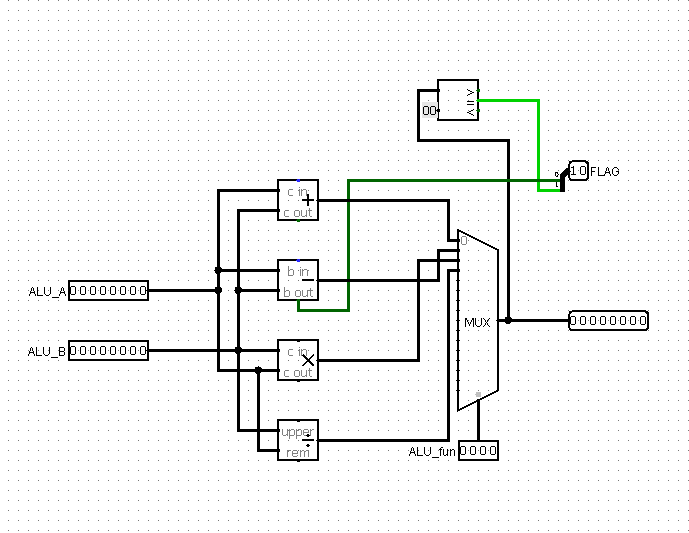
irmov FB -> %ecx 70 2 0 FB mrmov %ecx -> %eax 40 0 2 00 irmov F7 -> %ecx 70 2 0 F7 mrmov %ecx -> %ebx 40 1 2 00 add %eax + %ebx 00 0 1 00 push %ebx B1 1 0 01

Register File:



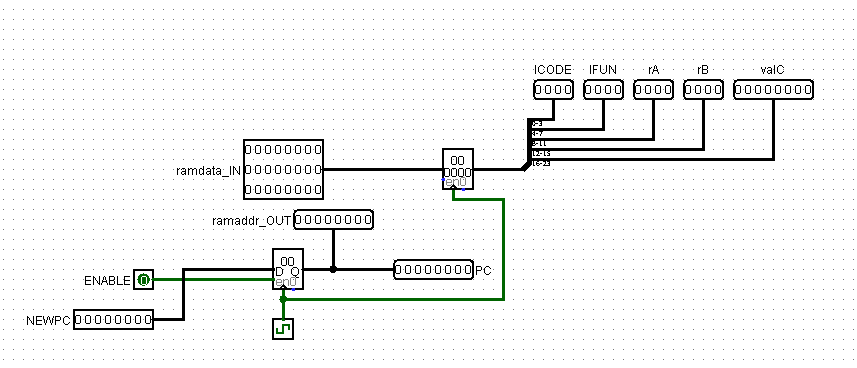
The register file is responsible for storing and updating register values. The 2 input and output values correspond to the 2 matrix values being operated on. The ports determine which register to use. The enable lines determine whether to read from or write to a register. The output will be the values read from the registers.

ALU:



The ALU is in for adding, subtracting, dividing, or multiplying the values inside the matrices corresponding to the ALU\_A and ALU\_B values. The result of the operation is the output. The mux enable determines which operation will be done. The flag is for checking whether or not a jump instruction will be completed. 01 represents negative or less than , 00 represents equal to, and 10 represents positive or greater than.

Instruction Memory:



Data memory:

