CSCE 312.506: Final Project

Dr. Yum

TA: Haopei Wang

Akash Kundu, Cole Boggus, Julian Tiu, Robert Paraliticci

Abstract: In this project, we were challenged to create a simplified processor that can carry out two by two matrix addition. To achieve this, we had to design an instruction set architecture in Logisim. The major components in our project were an instruction fetch mechanism, a decoder, controller, ALU, registers, and program counter.

Results and Discussions:

Program Counter: Our PC is composed of an enabler and a register. It is essentially a counter that stores the first memory location of the ROM because each location in ROM corresponds to a specific instruction. The way the PC is iterated is every 4 clock cycles the PC increments by 1 until the maximum value 0x17 is reached which the 24^{th} instruction which corresponds to the 24^{th} memory address in ROM.

Instruction Decoder: Our instruction decoder is comprised of 3 main parts. The decoder takes in an 8-bit input the first two of which are fed into the operation code decoder which takes the two bits and decides which operation will happen. There are four possible operations: read, load store, and add. The next three bits from the input are fed into the second part of the instruction decoder, which is a register decoder. The last three bits are fed into the last part which is another register decoder. These register decoders specify which registers are going to be used.

Register File: the register file is comprised of six registers. The first register holds the memory address of the first matrix from RAM. Every time the first register is accessed, there is a counter that is incremented so that the memory address that is outputted is incremented corresponding to the memory address of the next value of the first matrix. The second register functions the same way as register one, except the address it stores is the address of the second matrix instead of the first. Register one outputs a location and register three stores the value of whatever is at the memory location that register one pointed to. Register number four functions the same way register three does, however it uses the address from register two as opposed to register one. Once register three and four have the values from the locations from register one and two, the two values are outputted to the ALU and it writes back the result. This result is stored in register six and the address of the result where it is stored in RAM is stored in register five. Moreover, every time register five is accessed, it is also incremented by the counter.

Arithmetic Logic Unit: The ALU in our design is comprised of an 8-it adder and an enabler that is controlled by the operation code decoder from the instruction decoder. The ALU is passed the values from register three and four from the register file and adds them. The result is then written back to register six.

Conclusion: In our project, we combined aspects of previous labs to create an extremely simplified computer. Even though our computer is only capable of adding two two by two matrices, it is easy to imagine how we could scale this project to include other operations and features. We learned how to design an Instruction set Architecture with an 8-bit input and how to utilize each bit simultaneously by sending each bit to a separate and specific function in the design.