Daniel Sciortino
Evan Weiss
ECE 443 Project 3: 16 – Bit MIPS Processor

Introduction

For this final project, we were required to make a 16 – bit single cycle MIPS processor. We were to take the ALU that we created structurally from project 2 and use it at the center of this new design. In addition, we needed to implement a Register file, RAM, ALU control, and program counter. Once that was all complete connect all the components into a single file and verify the processor works correctly.

Components

ALU

The Arithmetic Logic Unit is the main component of a processor. It is responsible for the mathematical calculations as well as being used to pass through the correct information from the registers to the memory. Because our processor did not have any branch or jump capabilities the ALU is used on every instruction. It has 5 capabilities; addition, multiplication, subtraction, pass-through argument A, and pass-through argument B. The functionalities of these can be seen in the following test.

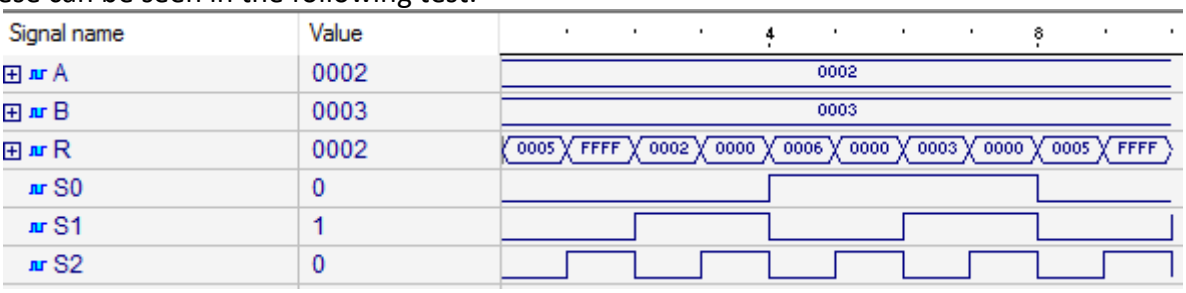

Figure 1: Test Results of the ALU

In order to test the functionality, the inputs for the ALU are given as 2 and 3. All the possible ALU controls are passed displaying each functionality of the ALU. Note that the subtraction yields 0xffff which is the 2's compliment of negative one

Instruction Decode Control Unit

The instruction decode control unit is what takes the opcode from the instruction and sets various control signals based on the operation indicated by the op code. The entity takes just the opcode as an input and from it generates the signals to indicate reading of the memory, the multiplexer to choose between memory output and the ALU output to be sent to the register file, the write memory control, the multiplexer choosing between a register or immediate value for the ALU, and the register write command. This is all performed with combinational logic. The test results can be seen in Figure 2.

Daniel Sciortino
Evan Weiss
ECE 443 Project 3: 16 – Bit MIPS Processor

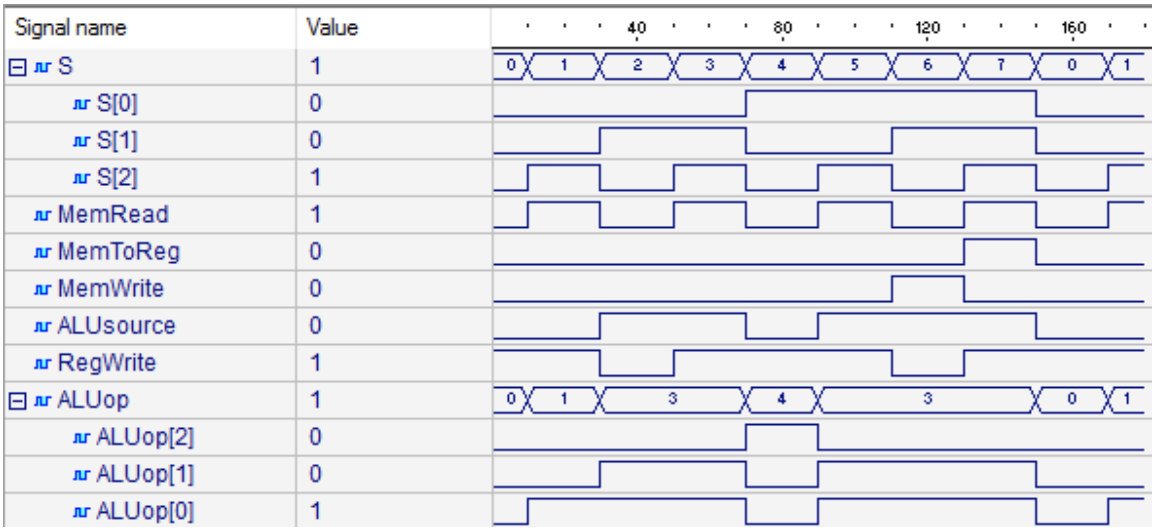

Figure 2: Test Results of the Control Unit

The test verified the combinational logic that was implemented in the control unit. It then drives all of the control signals to the multiplexers, ALU, and data memory.

Register File

The register file is the part of the process where values are stored before being used with the ALU. For this processor it is 16 bytes wide and has 16 registers. These registers can be accessed two at a time and written to one at a time if the write enable signal from the control unit is high. The functionality of the Register file is spot tested and shown in Figure 3.

Daniel Sciortino
Evan Weiss
ECE 443 Project 3: 16 – Bit MIPS Processor

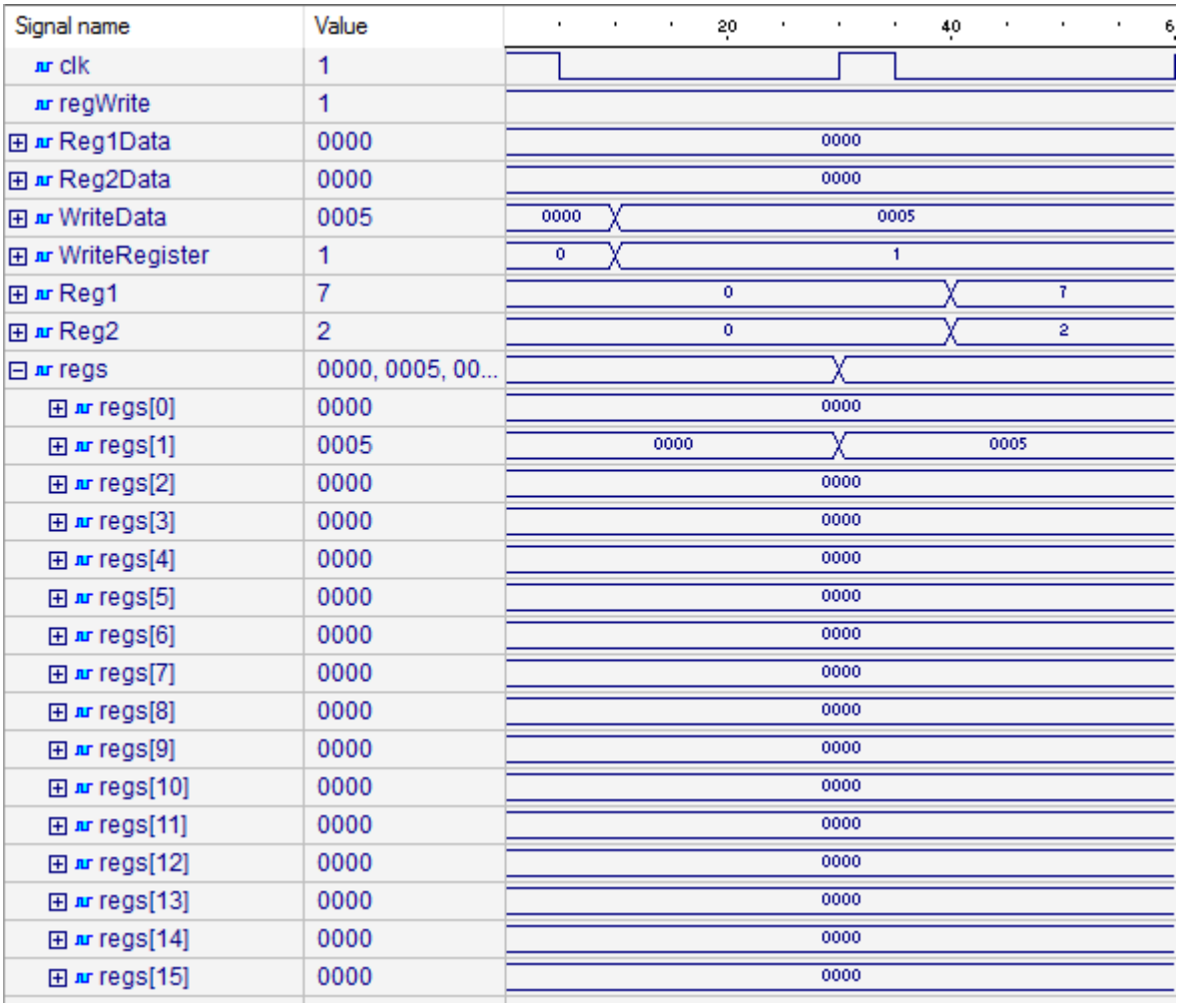

Figure 3: Functionality of the Register File

The registers are all initialized to zero. On each clock the registers are written and read. The test shows that data is correctly being written into $r1 and that $r7 and $r2 are read yielding all zeros.

Data Memory

The data memory allows for the processor to hold more than just the registers worth of data. The memory can be loaded and read from using the store half word and load half word instructions. The memory has 256 locations that are 16 bits wide each. To test them a few were written to and read from concurrently while the Write data signal was turned on and off. The results can be seen in Figure 4.

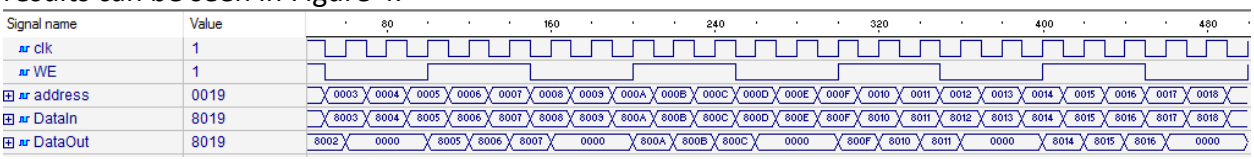

Figure 4: Test Results of the Data Memory

Daniel Sciortino
Evan Weiss
ECE 443 Project 3: 16 – Bit MIPS Processor

Instruction Memory

The instruction memory is where the program is loaded to allow the processor to fetch instructions. Each instruction is loaded into a 16-bit location that is indexed by its corresponding program counter number. This allows the program counter to increment the count each cycle so that the next instruction is ready for the next cycle. To test the instruction memory and verify that the correct program was loaded the test cycled through each possible program count and read the output. The results can be seen in Figure 5.

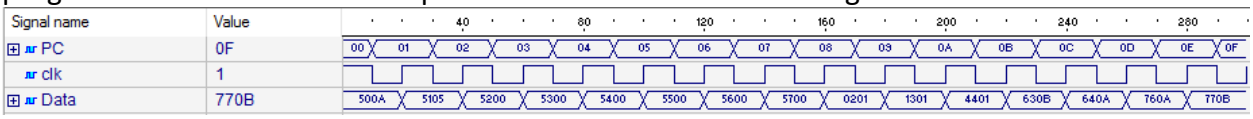

Figure 5: Test Results of the Instruction Memory

This test data compared to the compiled code shown in Table 1 verifies the functionality of the instruction memory.

`ldi $r0, 10`	500A
`ldi $r1, 5`	5105
`ldi $r2, 0`	5200
`ldi $r3, 0`	5300
`ldi $r4, 0`	5400
`ldi $r5, 0`	5500
`ldi $r6, 0`	5600
`ldi $r7, 0`	5700
`add $r2, $r0, $r1`	0201
`mult $r3, $r0, $r1`	1301
`sub $r4, $r0, $r1`	4401
`sh $r3, 0x0B`	630B
`sh $r4, 0x0A`	640A
`lh $r6, 0x0A`	760A
`lh $r7, 0x0B`	770B

Table 1: Code to be run and its hexadecimal conversion

Processor

The culmination of the project was the connection of all of the sub parts into a processor. The processor entity simply connects the lower entities together using port maps and signals to send information between them. The processor's slowest component was the ALU which restricted the processors clock speed to 10kHz. The only input needed for the processor is the clock which is run at this speed by the test bench. The program's execution is detailed in Figure 6.

Daniel Sciortino
Evan Weiss
ECE 443 Project 3: 16 – Bit MIPS Processor

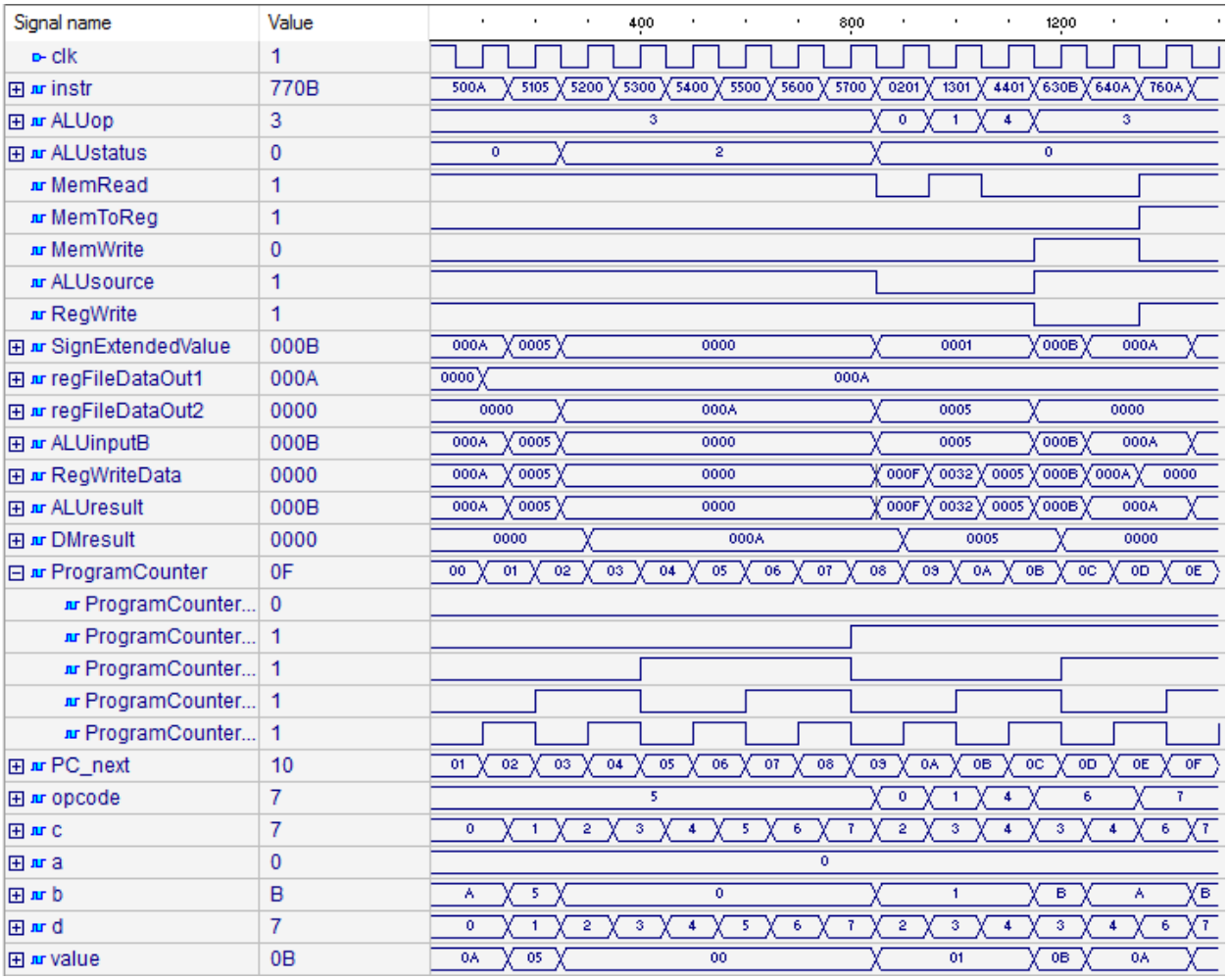

Figure 6: Program Execution

Conclusion

The processor functionality was confirmed at each stage by the lower level tests. After the components of the processor could be trusted they were compiled together into the 16-bit processor architecture. The functionality of the resulting architecture was then verified using a final test by running the given program.