COEN 122L: Computer Architecture Lab
Final Project Report

Stephen Tambussi and Kevin Molumphy

Spring 2022 - 2:15pm Thursday

Abstract

The purpose of this project was to design a structural model of a pipelined CPU with 13 instructions using Verilog HDL. We were given a unique instruction set architecture and created a datapath specific to those instructions. The datapath's purpose is to show how the data flowed and was manipulated in each module, and how it changed between cycles. This was a very difficult task as each individual bit of thousands had to be handled perfectly or the output would likely be incorrect. We did this by breaking down each step into separate modules that could be individually tested and verified. Then, we were able to complete the datapath by combining all of the modules together and making sure that each bit went into the correct place. This was done by sequentially combining and testing modules until we had the entire model working as intended.

Project Design

Every cycle in the datapath is controlled by the positive and negative edges of the Clock. On the positive edge each buffer sends the values through the output wires, muxes, and modules into the next buffer. The system starts with the PC that counts which programs are running and about to run. Every cycle the PC address is incremented by an adder but this can change if the Jump or Branch functions are called. The PC then sends the address of the next instruction to the Instruction Memory module. The PC address and instruction from the Instruction Memory are stored in the IF/ID Buffer. The IF/ID sends the correct bits to the Control module which uses wire signals to determine the current operation the datapath is executing by changing mux outputs. The two register address inputs are read from the instruction by the Register File which then reads from that register and sends the output to the ID/EX buffer. The Immediate Generator generates the correct immediate based on specific bits from the instruction and sends that to the ID/EX buffer. In the execution stage, the Data Memory is written to or the ALU is used based on the control bits. The inputs to the ALU are also controlled by muxes determined by control bits. There are Negative and Zero flags as outputs from the ALU control unit for the Branch instructions which are compared to their respective control bits to determine if the PC needs to jump to a different instruction. The outputs of the Data Memory and ALU are passed into the EX/WB buffer. One last mux is used to determine which of the outputs in the EX/WB buffer are sent to the Register File. This completes the cycle and the next instruction begins. For the non-jump instructions, they can work in sync since the buffers are pipelined as long as there are no data dependencies.

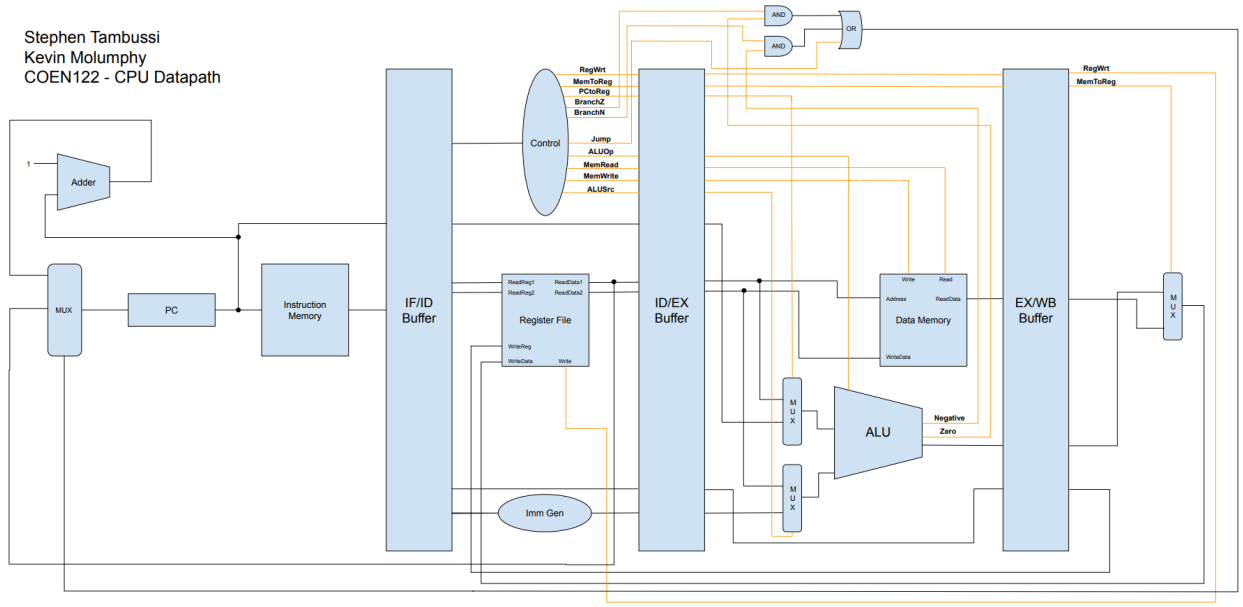


Figure 1: CPU Datapath

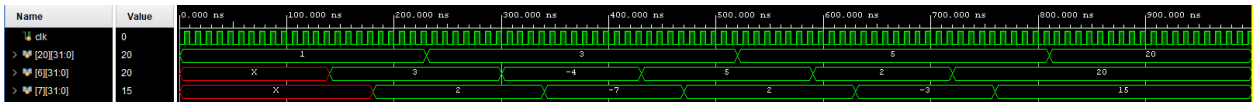
Kevin Molumphy & Stephen Tambussi - COEN122											
Instruction:	**Opcode:**	**regWrite:**	**memToReg:**	**PCtoReg:**	**branchN:**	**branchZ:**	**jump:**	**memRead:**	**memWrite:**	**aluOp:**	**aluSrc:**
NOP	0000:	0	0	0	0	0	0	0	0	000:	0
Save PC	1111:	1	0	1	0	0	0	0	0	100:	1
Load	1110:	1	1	0	0	0	0	1	0	111:	0
Store	0011:	0	0	0	0	0	0	0	1	111:	0
Add	0100:	1	0	0	0	0	0	0	0	100:	0
AddI	0101:	1	0	0	0	0	0	0	0	100:	1
Negate	0110:	1	0	0	0	0	0	0	0	010:	0
Subtract	0111:	1	0	0	0	0	0	0	0	001:	0
Jump	1000:	0	0	0	0	0	1	0	0	111:	0
Branch if zero	1001:	0	0	0	0	1	0	0	0	111:	0
Where an I	1010:	1	0	1	0	0	0	0	0	111:	0
Branch if negative	1011:	0	0	0	1	0	0	0	0	111:	0
MAX	0001:										

Figure 2: Truth Table of Control

Simulation Waveform

Figure 3: Entire Waveform

MAX Calculation

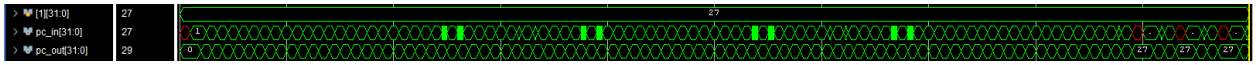


As shown above, the execution of the MAX program was successful. Register 20 holds the current maximum value as the program iterates through the array. The values in the array are [3, -4, 5, 2, 20] and so 20 is the maximum value.

Where am I Instruction

Shown above is the waveform for the *Where am I* instruction. At the time the instruction is called, the MAX program has finished execution and the current PC is 27.

Jump Instruction



The waveform above is for the *Jump* instruction. Register 1 is used as the argument to the instruction and it holds the value 27. This instruction also executes after the MAX program has finished execution.

Assembly Code for MAX Calculation

```
assign insts[0] = 32'b00000000000000000000000000000000; //NOP
assign insts[1] = 32'b00000000000000000000000000000000; //NOP
assign insts[2] = 32'b00000000000000000000000000000000; //NOP
assign insts[3] = 32'b00000000000000000000000000000000; //NOP
assign insts[4] = 32'b00000000000000000000000000000000; //NOP
assign insts[6] = 32'b0100_000101_000010_000011_0000000000; //ADD x5, x2, x3
assign insts[7] = 32'b1111_001001_000000_000000_0000000010; //SVPC x9, 2
assign insts[8] = 32'b1111_001010_000000_000000_0000001011; //SVPC x10, 11
assign insts[9] = 32'b1110_000110_000010_000000_0000000000; //LD x6, x2
assign insts[10] = 32'b00000000000000000000000000000000; //NOP
assign insts[11] = 32'b00000000000000000000000000000000; //NOP
assign insts[12] = 32'b00000000000000000000000000000000; //NOP
assign insts[13] = 32'b0111_000111_000110_010100_0000000000; //SUB x7, x6, x20
assign insts[14] = 32'b1011_000000_001010_000000_0000000000; //BRN x10
assign insts[15] = 32'b00000000000000000000000000000000; //NOP
assign insts[16] = 32'b00000000000000000000000000000000; //NOP
assign insts[17] = 32'b00000000000000000000000000000000; //NOP
assign insts[18] = 32'b0100_010100_000110_000000_0000000000; //ADD x20, x6, x0
assign insts[19] = 32'b0101_000010_000010_000000_0000000001; //ADDi x2, x2, 1
```

```
assign insts[20] = 32'b00000000000000000000000000000000; //NOP
assign insts[21] = 32'b00000000000000000000000000000000; //NOP
assign insts[22] = 32'b0111_001000_000010_000101_0000000000; //SUB x8, x2, x5
assign insts[23] = 32'b1011_000000_001001_000000_0000000000; //BRN x9
assign insts[24] = 32'b00000000000000000000000000000000; //NOP
assign insts[25] = 32'b00000000000000000000000000000000; //NOP
assign insts[26] = 32'b00000000000000000000000000000000; //NOP
```

Execution Time of MAX Calculation

The MAX program has 16 NOPs, each is 1 cycle for a total of 16 cycles. Every instruction executes in 4 cycles, but some instructions execute multiple times throughout the duration of the program. The instructions that execute once are 12 cycles total. There are 6 instructions that execute 5 times for 4 cycles that give a total of 120 cycles. Finally, there is one instruction that executes 3 times for 4 cycles and gives a total of 12 cycles. Adding these all up, the total number of cycles for the MAX program is 160. Considering that the clock period is 5ns, the execution time of the MAX program is 800 ns. This is mostly inline with the waveform where the last value is updated around the 800 ns mark.