CompArch Lab 3

Ian Hill, Kathryn Hite, & William Lu

November 2016

1 Description and Block Diagram

We originally endeavored to craft a full pipeline processor implementing the ten MIPS ISA instructions in the project description; however, after a great deal of work, we decided to implement the pipeline processor with stalling logic after most instructions in order to test our design before complicating it further. In some ways, we've effectively implemented a fancy single instruction processor.

In the Instruction Fetch Phase, we retrieve a 32-bit MIPS instruction from the instruction memory and perform the appropriate changes to the program counter. Three muxes lead into the program counter which determines whether the PC (a) increments four, (b) jumps because of a branch instruction, (c) jumps because of a jump instruction, or (d) jumps because of a jump register instruction. The 32-bit address of the program counter determines the output of the instruction memory and therefore the instruction which is passed through the inter-phase register onto the Instruction Decode Phase.

In the Instruction Decode Phase, we sign-extend immediates, calculate control signals, and retrieve values from the register file. The MIPS instruction is broken apart such that the op code and funct code are fed into the controller, immediates are wired to the sign extender, and the appropriate register addresses are connected to the register file or passed onto the Execution Phase.

In the Execution Phase, we run computations on the ALU, decide the appropriate destination register for the computation, determine whether to branch, and shift left 2 bits 32-bit instruction addresses in order to jump the program counter.

In the Memory Phase, we store or load 32-bit numbers into data memory or allow computations to pass through the phase onto the Write Back Phase.

In the Write Back Phase, we allow the output, which has been passed through the Memory Phase from the Execution Phase, to write to the register file if appropriate.

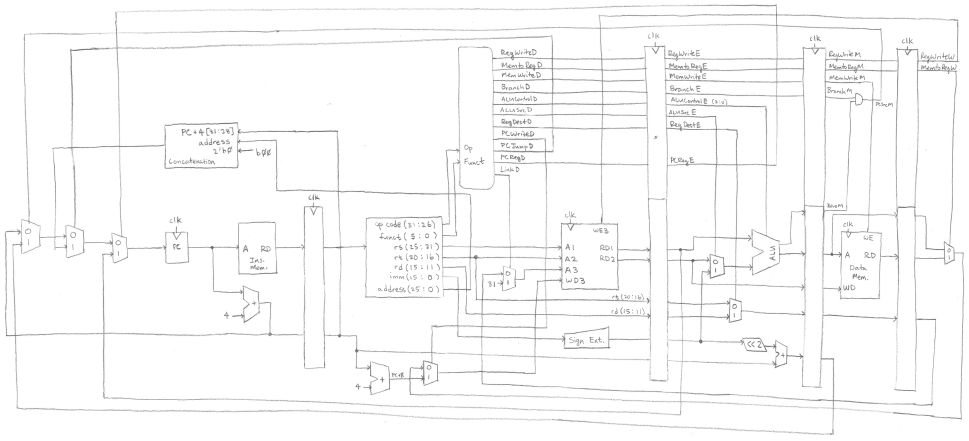

Figure 1: The full CPU layout

2 Test Strategy and Results

We tested each of the individual submodules including the new controller, data memory, and signextender that had not been used in previous labs with individual test files.

To test the functionality of the full processor, we use two assembly programs. The first runs through every one of the ten supported commands as well as computing the square of a number. The second computes prime numbers up to a given input and also tests using inputs to the assembly that are not preset.

3 Performance and Area Analysis

4 Work Plan Reflection

The hours that we put in prior to the midpoint check-in closely followed the work plan, and we achieved our goals leading up to that point. After that, we highly underestimated the amount of time that it would take to debug the individual submodules and design and assemble the final CPU structure.

```
-------------------------
CompArch Lab 3 Work Plan
ESTIMATED
-------------------------
```

```
(12) Build a Pipelined CPU
   (2) Full diagram of CPU
   (2) Build Verilog modules for CPU

(4) Write a Test Program
   (1) Fibonacci
   (2) Orignal test assembly program
   (1) Scripts to run test programs

----Midpoint Checkin----

(8) Build a Pipelined CPU
   (3) Write testbenches for Verilog modules
   (5) Assemble and Debug CPU

(4) Write Report
   (1) Written description and block diagram of your processor architecture w/
       selected RTL
   (1) Description of test plan and results
   (2) Performance/area analysis of your design
```

Implementing a pipeline CPU took us significantly longer than we estimated.

```
------------------------
CompArch Lab 3 Work Plan
ACTUAL
------------------------


(14) Build a Pipelined CPU
   (6) Full diagram of CPU
   (8) Build Verilog modules for CPU

(6) Write a Test Program
   (1) Fibonacci
   (2) Orignal test assembly program
   (2) Debugging the test assembly program
   (1) Scripts to run test programs

----Midpoint Checkin----

(15) Build a Pipelined CPU
   (3) Write testbenches for Verilog modules
   (12) Assemble and Debug CPU

(4) Write Report
```

```
  (1) Written description and block diagram of your processor architecture w/
      selected RTL
  (1) Description of test plan and results
  (2) Performance/area analysis of your design
```