Billy Koech and Nicolas Weinger

CS 141: Computing Hardware

Lab 4 Part 3 : MIPS Datapath (J Types)

Testing methodology

The MIPS data path now supports J types in addition to R types and J types. The following blocks are implemented as modules and instantiated in the MIPS core module:

1. Control
2. ALU control
3. ALU
4. PC Register
5. Register File
6. IorD mux
7. IR register
8. MDR register
9. RegDst mux
10. MemtoReg mux
11. A register
12. B register
13. ALUSrcA 3 to 1 mux
14. ALUSrcB 5 to 1 mux
15. PCSource 3 to 1 mux
16. ALUout register

The Control FSM module is tested by creating a testbench that resets the FSM and runs it for about 100 clock cycles. The waveform of the enable pins (*IRWrite, MemWrite RegWrite, PCWriteCond*) and select pins (*IorD, ALUSrcA, ALUSrcB, ALUOp, PCSource, RegDst,*

*MemtoReg*) are then observed in each state and checked for consistency with the MIPS R-Type FSM diagram.

In order to test for functionality of the entire data path for J-types we convert assembly code to machine code and load it into the instruction memory.

* j

START: nor $t1, $t1, $t1 # load 1s into t1

j START # keep FSM looped in nor states

nor $t2, $t2, $t2 # this should not run

* jr

For jr we load an instruction into the data memory (address 0x0) and use the zero value stored in the $zero register to jump to the zero address.

Test instruction:

jr $zero # jump to the 0x0 address

Instruction stored in 0x0:

nor $t1, $t1, $t1 # load 1s into $t1

We then look at the value of $t1 in the Register File to confirm that it contains all 1s.

* jal

For jal we run the following assembly code then check the value of the $ra register in the Register File to confirm that it has stored the address of the third line of code. (PC+8)

START: nor $t1, $t1, $t1 # load 1s into t1

jal START # loop back, link next line to $ra

nor $t2, $t2, $t2 # this instr. addr should be in $ra

* beq

Test 1:

START: nor $t1, $t1, $t1 # loads 1s into $t1

beq $s0, $t1, START # skip and run next line

nor $t3, $t3, $t3 # load 1s into $t3

Test 2:

START: nor $t1, $t1, $t1 # loads 1s into $t1

beq $t1, $t1, START # loop back

nor $t3, $t3, $t3 # this should not run

* bne

Test 1:

START: nor $t1, $t1, $t1 # loads 1s into $t1

bne $s0, $t1, START # loop back

nor $t3, $t3, $t3 # this should not run  
Test 2:

START: nor $t1, $t1, $t1 # loads 1s into $t1

bne $t1, $t1, START # skip and run next line nor $t3, $t3, $t3 # load 1s into $t3

We then view the waveform of the 32 registers in the Register File and check whether they hold the values given in the comments in the above code. We found that they did.