

Lab 11: Writeback

Jiasen Zhou, Jon Johnston

April 14, 2019

1 Executive Summary

This lab had three parts: creating the Write Back stage, integrating it into the datapath, and then testing the datapath with code that performs a simple division. For the Write Back stage, a new module did not have to be created, a mux was simply added to the current datapath. However, an iWriteBack module was created to nest the mux for ease of reading. The division test did not require a new module to be created either, machine code for the division simply had to be created and implemented in the instruction memory file. The reg data and memory files also had to be edited to ensure correct operation. The division of 21 by 7 correctly produced the result of 3, demonstrating that all three parts of the lab were successful.

2 Test Report

To verify operation of these modules, this lab required two test benches.

1. Datapath Test Bench
2. Division Test Bench

2.1 Datapath Test Bench

The Datapath Test Bench contains:

1. Inputs
 - (a) reset - reset button that sets PC source to 0
2. Outputs
 - (a) read_data1 - data stored in read register 1
 - (b) read_data2 - data stored in read register 2
 - (c) write_data - the output of the Write Back stage

2.2 Division Test Bench

The Division Test Bench contains:

1. Inputs
 - (a) reset - reset button that sets PC source to 0
2. Outputs
 - (a) read_data1 - data stored in read register 1
 - (b) read_data2 - data stored in read register 2
 - (c) write_data - the output of the Write Back stage

The screenshot displays a logic analyzer interface with two main sections: a memory dump and a list of CPU registers.

Memory Dump: The top section shows a sequence of instructions. The first instruction at address 00000000 is a jump instruction: `00000000: 00000000 00000000 00000000 00000000`. The second instruction at address 00000004 is: `00000004: 00000000 00000000 00000000 00000000`. The third instruction at address 00000008 is: `00000008: 00000000 00000000 00000000 00000000`. The fourth instruction at address 0000000C is: `0000000C: 00000000 00000000 00000000 00000000`. The fifth instruction at address 00000010 is: `00000010: 00000000 00000000 00000000 00000000`. The sixth instruction at address 00000014 is: `00000014: 00000000 00000000 00000000 00000000`. The seventh instruction at address 00000018 is: `00000018: 00000000 00000000 00000000 00000000`. The eighth instruction at address 0000001C is: `0000001C: 00000000 00000000 00000000 00000000`. The ninth instruction at address 00000020 is: `00000020: 00000000 00000000 00000000 00000000`. The tenth instruction at address 00000024 is: `00000024: 00000000 00000000 00000000 00000000`. The eleventh instruction at address 00000028 is: `00000028: 00000000 00000000 00000000 00000000`. The twelfth instruction at address 0000002C is: `0000002C: 00000000 00000000 00000000 00000000`. The thirteenth instruction at address 00000030 is: `00000030: 00000000 00000000 00000000 00000000`. The fourteenth instruction at address 00000034 is: `00000034: 00000000 00000000 00000000 00000000`. The fifteenth instruction at address 00000038 is: `00000038: 00000000 00000000 00000000 00000000`. The sixteenth instruction at address 0000003C is: `0000003C: 00000000 00000000 00000000 00000000`. The seventeenth instruction at address 00000040 is: `00000040: 00000000 00000000 00000000 00000000`. The eighteenth instruction at address 00000044 is: `00000044: 00000000 00000000 00000000 00000000`. The nineteenth instruction at address 00000048 is: `00000048: 00000000 00000000 00000000 00000000`. The twentieth instruction at address 0000004C is: `0000004C: 00000000 00000000 00000000 00000000`. The twenty-first instruction at address 00000050 is: `00000050: 00000000 00000000 00000000 00000000`. The twenty-second instruction at address 00000054 is: `00000054: 00000000 00000000 00000000 00000000`. The twenty-third instruction at address 00000058 is: `00000058: 00000000 00000000 00000000 00000000`. The twenty-fourth instruction at address 0000005C is: `0000005C: 00000000 00000000 00000000 00000000`. The twenty-fifth instruction at address 00000060 is: `00000060: 00000000 00000000 00000000 00000000`. The twenty-sixth instruction at address 00000064 is: `00000064: 00000000 00000000 00000000 00000000`. The twenty-seventh instruction at address 00000068 is: `00000068: 00000000 00000000 00000000 00000000`. The twenty-eighth instruction at address 0000006C is: `0000006C: 00000000 00000000 00000000 00000000`. The twenty-ninth instruction at address 00000070 is: `00000070: 00000000 00000000 00000000 00000000`. The thirtieth instruction at address 00000074 is: `00000074: 00000000 00000000 00000000 00000000`. The thirty-first instruction at address 00000078 is: `00000078: 00000000 00000000 00000000 00000000`. The thirty-second instruction at address 0000007C is: `0000007C: 00000000 00000000 00000000 00000000`. The thirty-third instruction at address 00000080 is: `00000080: 00000000 00000000 00000000 00000000`. The thirty-fourth instruction at address 00000084 is: `00000084: 00000000 00000000 00000000 00000000`. The thirty-fifth instruction at address 00000088 is: `00000088: 00000000 00000000 00000000 00000000`. The thirty-sixth instruction at address 0000008C is: `0000008C: 00000000 00000000 00000000 00000000`. The thirty-seventh instruction at address 00000090 is: `00000090: 00000000 00000000 00000000 00000000`. The thirty-eighth instruction at address 00000094 is: `00000094: 00000000 00000000 00000000 00000000`. The thirty-ninth instruction at address 00000098 is: `00000098: 00000000 00000000 00000000 00000000`. The fortieth instruction at address 0000009C is: `0000009C: 00000000 00000000 00000000 00000000`. The forty-first instruction at address 000000A0 is: `000000A0: 00000000 00000000 00000000 00000000`. The forty-second instruction at address 000000A4 is: `000000A4: 00000000 00000000 00000000 00000000`. The forty-third instruction at address 000000A8 is: `000000A8: 00000000 00000000 00000000 00000000`. The forty-fourth instruction at address 000000AC is: `000000AC: 00000000 00000000 00000000 00000000`. The forty-fifth instruction at address 000000B0 is: `000000B0: 00000000 00000000 00000000 00000000`. The forty-sixth instruction at address 000000B4 is: `000000B4: 00000000 00000000 00000000 00000000`. The forty-seventh instruction at address 000000B8 is: `000000B8: 00000000 00000000 00000000 00000000`. The forty-eighth instruction at address 000000BC is: `000000BC: 00000000 00000000 00000000 00000000`. The forty-ninth instruction at address 000000C0 is: `000000C0: 00000000 00000000 00000000 00000000`. The fiftieth instruction at address 000000C4 is: `000000C4: 00000000 00000000 00000000 00000000`. The fifty-first instruction at address 000000C8 is: `000000C8: 00000000 00000000 00000000 00000000`. The fifty-second instruction at address 000000CC is: `000000CC: 00000000 00000000 00000000 00000000`. The fifty-third instruction at address 000000D0 is: `000000D0: 00000000 00000000 00000000 00000000`. The fifty-fourth instruction at address 000000D4 is: `000000D4: 00000000 00000000 00000000 00000000`. The fifty-fifth instruction at address 000000D8 is: `000000D8: 00000000 00000000 00000000 00000000`. The fifty-sixth instruction at address 000000DC is: `000000DC: 00000000 00000000 00000000 00000000`. The fifty-seventh instruction at address 000000E0 is: `000000E0: 00000000 00000000 00000000 00000000`. The fifty-eighth instruction at address 000000E4 is: `000000E4: 00000000`

PC	PC+5	PC+10	PC+15	PC+20	PC+25	PC+30	PC+35	PC+40	PC+45	PC+50	PC+55	PC+60	PC+65	PC+70	PC+75	PC+80	PC+85	PC+90	PC+95	PC+100	PC+105	PC+110	PC+115	PC+120	PC+125	PC+130	PC+135	PC+140	PC+145	PC+150	PC+155	PC+160	PC+165	PC+170	PC+175	PC+180	PC+185	PC+190	PC+195	PC+200	PC+205	PC+210	PC+215	PC+220	PC+225	PC+230	PC+235	PC+240	PC+245	PC+250	PC+255	PC+260	PC+265	PC+270	PC+275	PC+280	PC+285	PC+290	PC+295	PC+300	PC+305	PC+310	PC+315	PC+320	PC+325	PC+330	PC+335	PC+340	PC+345	PC+350	PC+355	PC+360	PC+365	PC+370	PC+375	PC+380	PC+385	PC+390	PC+395	PC+400	PC+405	PC+410	PC+415	PC+420	PC+425	PC+430	PC+435	PC+440	PC+445	PC+450	PC+455	PC+460	PC+465	PC+470	PC+475	PC+480	PC+485	PC+490	PC+495	PC+500	PC+505	PC+510	PC+515	PC+520	PC+525	PC+530	PC+535	PC+540	PC+545	PC+550	PC+555	PC+560	PC+565	PC+570	PC+575	PC+580	PC+585	PC+590	PC+595	PC+600	PC+605	PC+610	PC+615	PC+620	PC+625	PC+630	PC+635	PC+640	PC+645	PC+650	PC+655	PC+660	PC+665	PC+670	PC+675	PC+680	PC+685	PC+690	PC+695	PC+700	PC+705	PC+710	PC+715	PC+720	PC+725	PC+730	PC+735	PC+740	PC+745	PC+750	PC+755	PC+760	PC+765	PC+770	PC+775	PC+780	PC+785	PC+790	PC+795	PC+800	PC+805	PC+810	PC+815	PC+820	PC+825	PC+830	PC+835	PC+840	PC+845	PC+850	PC+855	PC+860	PC+865	PC+870	PC+875	PC+880	PC+885	PC+890	PC+895	PC+900	PC+905	PC+910	PC+915	PC+920	PC+925	PC+930	PC+935	PC+940	PC+945	PC+950	PC+955	PC+960	PC+965	PC+970	PC+975	PC+980	PC+985	PC+990	PC+995	PC+1000	PC+1005	PC+1010	PC+1015	PC+1020	PC+1025	PC+1030	PC+1035	PC+1040	PC+1045	PC+1050	PC+1055	PC+1060	PC+1065	PC+1070	PC+1075	PC+1080	PC+1085	PC+1090	PC+1095	PC+1100	PC+1105	PC+1110	PC+1115	PC+1120	PC+1125	PC+1130	PC+1135	PC+1140	PC+1145	PC+1150	PC+1155	PC+1160	PC+1165	PC+1170	PC+1175	PC+1180	PC+1185	PC+1190	PC+1195	PC+1200	PC+1205	PC+1210	PC+1215	PC+1220	PC+1225	PC+1230	PC+1235	PC+1240	PC+1245	PC+1250	PC+1255	PC+1260	PC+1265	PC+1270	PC+1275	PC+1280	PC+1285	PC+1290	PC+1295	PC+1300	PC+1305	PC+1310	PC+1315	PC+1320	PC+1325	PC+1330	PC+1335	PC+1340	PC+1345	PC+1350	PC+1355	PC+1360	PC+1365	PC+1370	PC+1375	PC+1380	PC+1385	PC+1390	PC+1395	PC+1400	PC+1405	PC+1410	PC+1415	PC+1420	PC+1425	PC+1430	PC+1435	PC+1440	PC+1445	PC+1450	PC+1455	PC+1460	PC+1465	PC+1470	PC+1475	PC+1480	PC+1485	PC+1490	PC+1495	PC+1500	PC+1505	PC+1510	PC+1515	PC+1520	PC+1525	PC+1530	PC+1535	PC+1540	PC+1545	PC+1550	PC+1555	PC+1560	PC+1565	PC+1570	PC+1575	PC+1580	PC+1585	PC+1590	PC+1595	PC+1600	PC+1605	PC+1610	PC+1615	PC+1620	PC+1625	PC+1630	PC+1635	PC+1640	PC+1645	PC+1650	PC+1655	PC+1660	PC+1665	PC+1670	PC+1675	PC+1680	PC+1685	PC+1690	PC+1695	PC+1700	PC+1705	PC+1710	PC+1715	PC+1720	PC+1725	PC+1730	PC+1735	PC+1740	PC+1745	PC+1750	PC+1755	PC+1760	PC+1765	PC+1770	PC+1775	PC+1780	PC+1785	PC+1790
----	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------	---------

```

reg reset;
wire [‘INSTR_LEN-1:0] Instruction;
wire uncond_branch, branch,
      mem_read, mem_to_reg,
      mem_write, ALU_src, clk,
      fetch_clk, idecode_clk,
      im_clk, wb_clk;
wire [1:0] ALU_op;

```

```

wire [‘WORD-1:0] branch_target , cur_pc ,
        read_data , read_data1 , read_data2 ,
        write_data , sign_extended , alu_result ;
wire zero , pc_src ;

oscillator r_clk (. clk ( clk ));           //base clock
delay #(.DELAYAMT(1)) ClkPlus3 (. a ( clk ),
        . a_delayed ( fetch_clk ));        //fetch clock
delay #(.DELAYAMT(2)) ClkPlus2 (. a ( clk ),
        . a_delayed ( idecode_clk ));      //decode clock
delay #(.DELAYAMT(4)) ClkPlus6 (. a ( clk ),
        . a_delayed ( im_clk ));           //memory clock
delay #(.DELAYAMT(6)) ClkPlus8 (. a ( clk ),
        . a_delayed ( wb_clk ));           // writeback clock

fetch fetch_mod(
        . clk ( clk ),
        . fetch_clk ( fetch_clk ),
        . reset ( reset ),
        . branch_target ( branch_target ),
        . pc_src ( pc_src ),
        . instruction ( Instruction ),
        . cur_pc ( cur_pc ));

iDecode decode_mod(
        . write_data ( write_data ),
        . Instruction ( Instruction ),
        . uncond_branch ( uncond_branch ),
        . branch ( branch ),
        . mem_read ( mem_read ),
        . mem_to_reg ( mem_to_reg ),
        . mem_write ( mem_write ),
        . ALU_src ( ALU_src ),
        . read_clk ( idecode_clk ),
        . write_clk ( wb_clk ),
        . ALU_op ( ALU_op ),
        . read_data1 ( read_data1 ),
        . read_data2 ( read_data2 ),
        . sign_extended ( sign_extended ));

iExecute execute_mod(
        . pc_in ( cur_pc ),
        . read_data1 ( read_data1 ),
        . read_data2 ( read_data2 ),
        . sign_extend ( sign_extended ),

```

```

        .opcode(Instruction[31:21]),
        .alu_op(ALU_op),
        .alu_src(ALU_src),
        .alu_result(alu_result),
        .zero(zero),
        .branch_target(branch_target));

iMemory memory_mod(
    .im_clk(im_clk),
    .alu_result(alu_result),
    .read_data2(read_data2),
    .mem_read(mem_read),
    .mem_write(mem_write),
    .zero(zero),
    .branch(branch),
    .uncondbranch(uncond_branch),
    .read_data(read_data),
    .pc_src(pc_src));

iWrite_back writeback_m(
    .read_data(read_data),
    .alu_result(alu_result),
    .MementoReg(mem_to_reg),
    .write_data(write_data));

initial
begin
    reset = 1; #10
    reset = 0; #190
$finish;
end

endmodule

```