The MIPS Pipeline

Purpose
Start implementing a pipelined MIPS processor.

Method
Build and connect the pipeline registers to create a MIPS pipelined processor that will work under non-hazard conditions.

Files to Use
datapath_with_pipelineregs.circ, control.circ, cpu32.circ, misc32.circ, loop.mem, and cs3410.jar.

Acknowledgments: This assignment is an adopted version of an assignment constructed by Thomas M. Parks and Chris Nevison at Colgate University, and from Uppsala University and Cornell University.

Get Started

Create a new folder (directory) named mips_datapath_with_pipelineregs. Download all the necessary files. Before you continue, make sure these files are in the mips_datapath_with_control folder:

- datapath_with_pipelineregs.circ, control.circ, cpu32.circ, misc32.circ, loopNoHazards.mem, cs3410.jar

Open the datapath.circ project in Logisim. The IF/ID register file has been built *and wired* for you. The ID/EX register file has been built *but not connected.* You will need to connect the ID/EX register inputs and outputs, and also build the EX/MEM and MEM/WB register files, and connect them. You should also modify the Load datapath to account for the Write Register.

Load the *loop.mem* test machine code file into Instruction Memory. To do this, right click on the instruction memory and click “Load Image.”

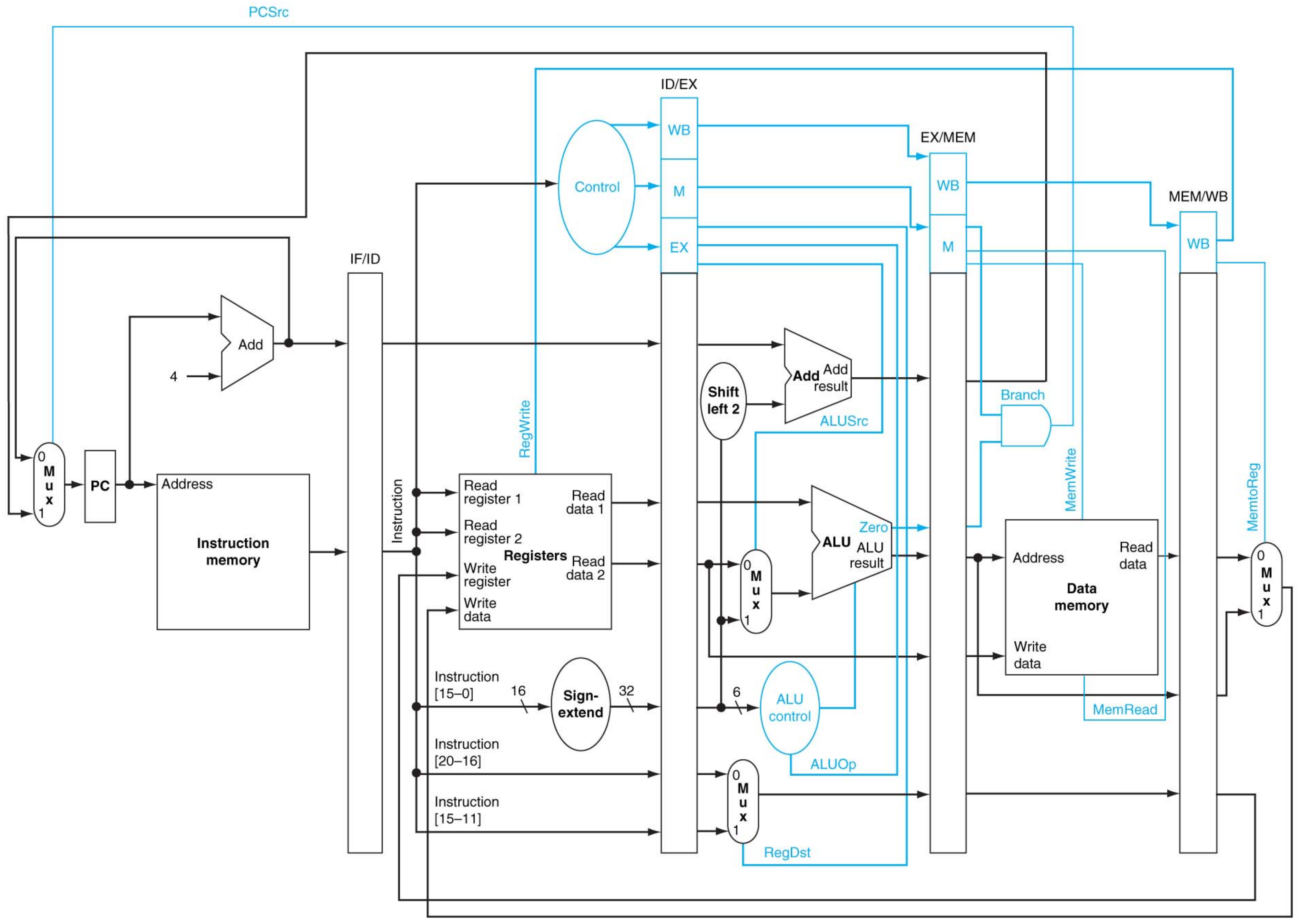
Testing Your Pipelined Processor

Once all of your pipeline registers are complete (don’t forget the reset and clock signals), you should be able to test our 3-instruction loop. The data hazard has been removed, but we now write register $9 into memory. In order to test this instruction, you should manually set register $9 in the register file to be equal to a value to be copied.

In order to start execution, make sure that you have reset the processor and left the clock in the “high” position, then set reset back to low.

Don’t forget: the clock cycle only covers *one stage* (but is shorter), so you need multiple clock cycles per instruction.

Figure 4.51 from COAD is copied below, and should be very helpful in determining which registers go where:



PCSrc
ID/EX
EX/MEM
MEM/WB
IF/ID
WB
M
EX
Control
Add
4
M u x
0
1
PC
Address
Instruction memory
Instruction
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
RegWrite
Instruction [15–0]
16
Sign-extend
32
Instruction [20–16]
Instruction [15–11]
Shift left 2
Add
Add result
ALUSrc
ALU
Zero
ALU result
6
ALU control
ALUOp
RegDst
Branch
MemWrite
Data memory
Read data
Write data
MemRead
MemtoReg