Single Cycle Data Path & Control (continued)

Purpose
Become more familiar with the MIPS datapath by producing a working implementation of a MIPS subset.

Method
Connect the datapath Control and ALU Control wires up to the MIPS register file, memory, and branch, and run a test program with no manual input.

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

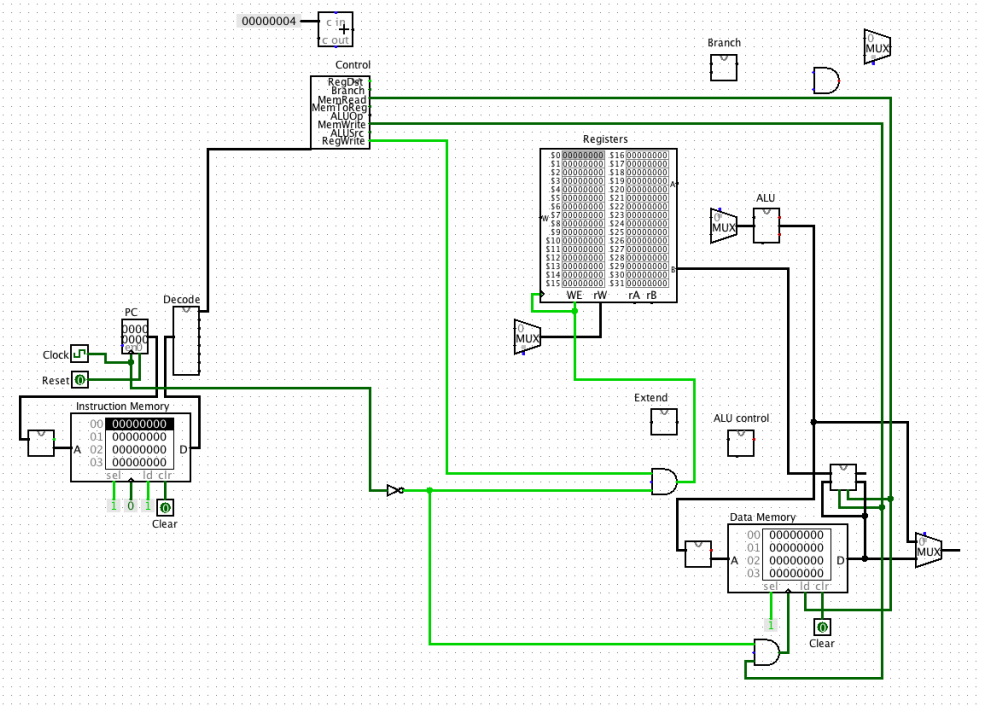
Acknowledgments: This assignment is an adopted version of an assignment constructed by Thomas M. Parks and Chris Nevison at Colgate University, and from Uppsala University and Cornell University.

Get Started

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

- datapath_with_control.circ, control.circ, cpu32.circ, misc32.circ, loop.mem, cs3410.jar

Open the datapath.circ project in Logisim. You should see a more complete data path than in the previous lab. Note that the register file has been expanded to look more like a MIPS register file:



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

Logisim Components

By the end of this lab, you should have a working MIPS datapath. To do this, you will need to connect and/or build the following components:

1. The Control input and all outputs must be connected properly.

2. The ALU control inputs and outputs must be connected.

3. The PC must be set to increment by 4 when not performing a branch instruction, and to jump to the proper address when performing a branch.

FIGURE 4.17 from COAD will guide you in connecting up the full datapath.

Notes and hints:

1. The register file does not have exactly the same connections as FIGURE 4.17. The translation is in the table below:

Figure 4.17	Logisim Register File
RegWrite	WE (already connected)
Read Register 1	rA
Read Register 2	rB
Write Register	rW
Write Data	W
Read Data 1	A
Read Data 2	B

2. The Decode block in Logisim is helpful so you don't have to worry about individual pins (e.g., [25-21]). The following table shows the translation:

Figure 4.17	Logisim Decode
Instruction [31-26] (already connected)	op
Instruction [25-21]	rs
Instruction [20-16]	rt
Instruction [15-11]	rd
Instruction [10-6] (not shown/used)	shamt

Figure 4.17	Logisim Decode
Instruction [5-0]	funct
Instruction [15-0]	immediate
Instruction [25-0] (not shown/used)	address

3. The branch control block has two inputs: PC+4 and a 32-bit unshifted immediate value. It takes care of shifting the immediate value and adding it to the PC+4 to produce a BranchAddress.

4. Suggested build order:

    a. PC (the input for the PC is at the middle-left of the PC block)

    b. Decode block to register file, sign extend

    c. Control block to MUXes, register file, ALU control, Data Memory, and Branch

    d. ALU control to ALU

    e. Register file connections A and B to ALU and MUX.

    f. Sign extend to branch

    g. PC+4 to branch

    h. Branch to MUX

    i. ALU zero to AND gate, then AND to MUX control

    j. Data Memory MUX to Register File

5. In order to test your full datapath, follow these steps:

    a. Go into test mode (click on the hand at the top left corner).

    b. Click the reset input so it reads “1”

    c. Click the clock so it reads “1”

    d. Click the reset so it reads “0”

    e. Click the clock to run the processor. The first instruction should load “11” into Register 8, the second instruction should load register 8 into Data Memory, and the third instruction should branch back to the first instruction.