Description of circuit files for Lab 7 – Single Cycle Data Path

The CPUComponents.circ file contains the following incomplete subcircuits which you will need to complete:

BranchAddress – Computation of the address to branch to. Given the branch offset, this component does the shift by two (addresses are always on words), then adds the current PC+4. The input pins on the left, from the top:

- PC+4 – 32 bits (PC for current instruction plus 4)
- Branch offset – 32 bits

The output pin on the right is the branch address

JumpAddress – Computes the address to jump to for an unconditional jump. The inputs on the left from the top:

- PC+4 – 32 bits (PC for current instruction plus 4)
- Jump address – 26 bits

The output pin on the right is the jump address

The CPUComponents.circ file contains the following completed subcircuits:

Inst Mem Interface – accesses the instruction memory. The instruction memory is shown below it in the diagram. Connections, each 32 bits:

- Instruction Address – on the left
- Instruction – on the right

InstructionDecode – splits out the instruction fields from the instruction. The field connections on the right of this component are as follows from top to bottom:

- Opcode – 6 bitd
- Rs – 5 bits
- Rt – 5 bits
- Rd – 5 bits
- ShAmt – 5 bits
- Function code – 6 bits
- Immediate – 16 bits
- Jump Address – 26 bits

Registers – a full bank of 32 32-bit registers. The pins on the left side are, from the top:

- Select Read Register 1 – 5 bits
- Select Read Register 2 – 5 bits
- Select Write Register – 5 bits
- Data to Write – 32 bits

The control pins on the bottom are one bits each, from left to right:

- Write Enable
- Clock
- Reset – sets all registers to zero

The output pins on the right side, each 32 bits, top to bottom:

- Read Register 1
- Read Register 2

ALU – creates a MIPS ALU circuit (somewhat different from the one defined in the text). The pins on the left, each 32 bits:

Input A
Input B

The control pin on the bottom takes the 4-bit ALU control signal.
The output pins on the right, from top to bottom:

Zero – 1 bit
Result – 32 bits
Overflow – 1 bit

Data Mem Interface – provides an interface for accessing the data memory. The pins on the left, from the top:

Data Address – 32 bits, for either write to memory or reds from memory
Data to Memory – 32 bits, data to be written to the memory

The output pin on the right side of this component is the 32 bit data value read from memory
The leftmost control pin on the bottom of this component is the 1 bit clock signal. The other pins on the bottom connect to the Logi-Sim memory component, shown below the component in the circuit.

Sign Extend – Extends a 16-bit twos-complement value to a 32 bit value. The 16 bits come in on the left and the 32 bits go out on the right.

For the ALU the operation controls are as follows:

and	0000
or	0001
add	0010
sub	0110
slt	0111
nor	1100

Note: the corresponding immediate operations would have the same controls here, but the inputs to the ALU would be different. For example, the addi, lw and sw instructions would use the add control here, with immediate input and the result being used as the memory address for lw and sw.

The MiscComponents.circ file contains the following subcircuits that are used to build the CPUComponents.circ subcircuits:

GetSign – send the sign bit of a 32-bit input to the down output and passes the 32-bit value to the right output.
SetLSB – takes a 32-bit input and sets its least significant bit to 1, leaving the other bits as is.
Zero – takes a 32-bit input and outputs a single bit as 1 if the value is zero, 0 otherwise.
Overflow – Takes the a, b and sum high order bits and sets the output bit to 1 if there is overflow from an arithmetic operation in the ALU (two operation bits at 10 indicating an add op)
Registers4 – creates 4 32-bit registers with 2-bit control for selecting read-1, read-2 and write.
Registers4Z – same as above except the zero register is hard-wired to be zero, cannot be written.
WordAddress – Pulls an 8-bit word address from a 32-bit byte address – our data path uses memory addressed with 8-bits for simplicity.

The control.circ file supplies the control and the ALUcontrol circuits to the datapath.circ file.

Subcircuits:

Control – input is the six-bit opcode from the instruction and output are the following control lines:

- Branch – 1 for branch instruction, 0 otherwise
- Jump – 1 for jump instruction, 0 otherwise
- MemToReg – 1 if write value for register is from memory, 0 otherwise
- MemRead – 1 for a memory read (for lw), 0 otherwise
- MemWrite – 1 if if memory is to be written, 0 otherwise
- ALUSrc – indicates source for ALU B-operand, 0 for the rt register and 1 for the sign-shifted immediate
- ALUOp – 2 bits indicate the ALU operation for ALUControl
  - 00 for addi, lw or sw instructions
  - 01 for branch instructions
  - 10 for R-type instructions
  - 11 for ori
- RegWrite – 1 if a register is to be written, 0 otherwise
- RegDst – determines whether destination register is from rt (0) or rd (1)

control_0 and control_1 – subcircuits for building the control circuit

ALUcontrol – input is the six-bit function code and the two-bit ALUOp from control, output is the 4-bit control for the ALU from the cpu32.circ file.

ALUcontrol_0 – subcircuit for implementing ALUcontrol circuit

Fake_Control and Fake_ALUcontrol – subcircuits that can be used in place of the control and ALUcontrol circuits for testing. If these are placed in the locations for control and ALUcontrol in the datapath circuit, then a ten-bit input can be placed on the top of the Fake_Control and a four-bit input on the top of the Fake_ALUcontrol, so the control values can be set by the user for testing purposes.

The SingleCycleCPU.circ file contains an incomplete implementation of the simplified MIPS single cycle processor described in the book.