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

The **CPUComponents.circ** file contains the following incomplete subcircuits:

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

BranchControl – Determines whether a branch is taken

The input pins on the left, from the top:

Unconditional branch control — 1 bit
Conditional branch control — 1 bit
Zero (from ALU) — 1 bit

Output pin on the right:

control to MUX that determines next value to PC — 1 bit
0 — PC gets PC+4
1 — PC gets branch address

The CPUComponents.circ file contains the following completed subcircuits:

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 – 11 bits
Rn – 5 bits
Rm – 5 bits
Rd/t – 5 bits
ShAmt – 6 bits (Unused in our implementation)
full instruction to immediate extend unit – 32 bits
opcode to ALU control unit — 11 bits

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

Select Read Register A – 5 bits
Select Read Register B – 5 bits
Select Write Register – 5 bits
Data to Write – 32 bits

The control pins on the top, one bit:

Write Enable

The control pins on the bottom, one bit each:

Clock
Reset – sets all registers to zero

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

Read Register A
Read Register B

Note: Register 31 is hardwired to be zero and writing to it has no effect.

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

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

Control pins at the top, one bit each:

MemWrite

MemRead

Pin on the right:

Data value read from memory — 32 bits

Pins on the bottom:

Clock signal in — 1 bit

Other pins on the bottom connect to the Logi-Sim memory component

Sign Extend – Takes full 32-bit instruction and determines which bits to sign extend, using bits 26 and 31 to differentiate between a load/store, a conditional branch, or an unconditional branch. Selects the appropriate field and sign-extends to 32 bits.

Left input pin:

full instruction — 32 bits

Right output pin:

sign-extended immediate — 32 bits (indeterminate on some instructions)

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

Sign — 1 bit

Overflow – 1 bit

Carry out — 1 Bit

Note: The four 1 bit outputs are just those needed to set the condition register (not implemented)

For the ALU the operation controls are as follows:

and	0000	
or	0001	
add	0010	
sub	0110	
nor	1100	(not implemented)
pass B	0011	(for CBZ, note error in the text)

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.

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/underflow from an arithmetic operation in the ALU (two operation bits at 10 indicating an add/subtract 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 three register is hard-wired to be zero, cannot be written.

WordAddress – Pulls an 8-bit word address from a 32-bit byte address – the logi-sim memory uses 8-bit addresses for 32-bit words.

MemoryData – Uses tri-state controls to create memory interface. Makes our data memry interface simpler.

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

Subcircuits:

Control – input is the eleven-bit opcode from the instruction and output are the following control lines:
- Unconditional Branch – 1 for branch, 0 otherwise
- Conditional Branch – 1 for conditional branch, 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 lw or sw instructions (specifies add for address calculation)
  - 01 for conditional branch instructions
  - 10 for R-type instructions (ALUControl determines R-type: add, sub, and, or)
  - 11 not specified for this simplified implementation
- RegWrite – 1 if a register is to be written, 0 otherwise
- Reg2Loc – determines whether second (B) read register is from rm (0) or rt (1)

control_0 and control_1 – subcircuits for building the control circuit

ALUcontrol – input is the eleven-bit opcode and the two-bit ALUOp from control, output is the 4-bit control for the ALU.

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 output values can be set by the user for testing purposes.

The **SingleCycleCPU.circ** file contains an incomplete implementation of the simplified ARM (LEGv8) single cycle processor described in the book.