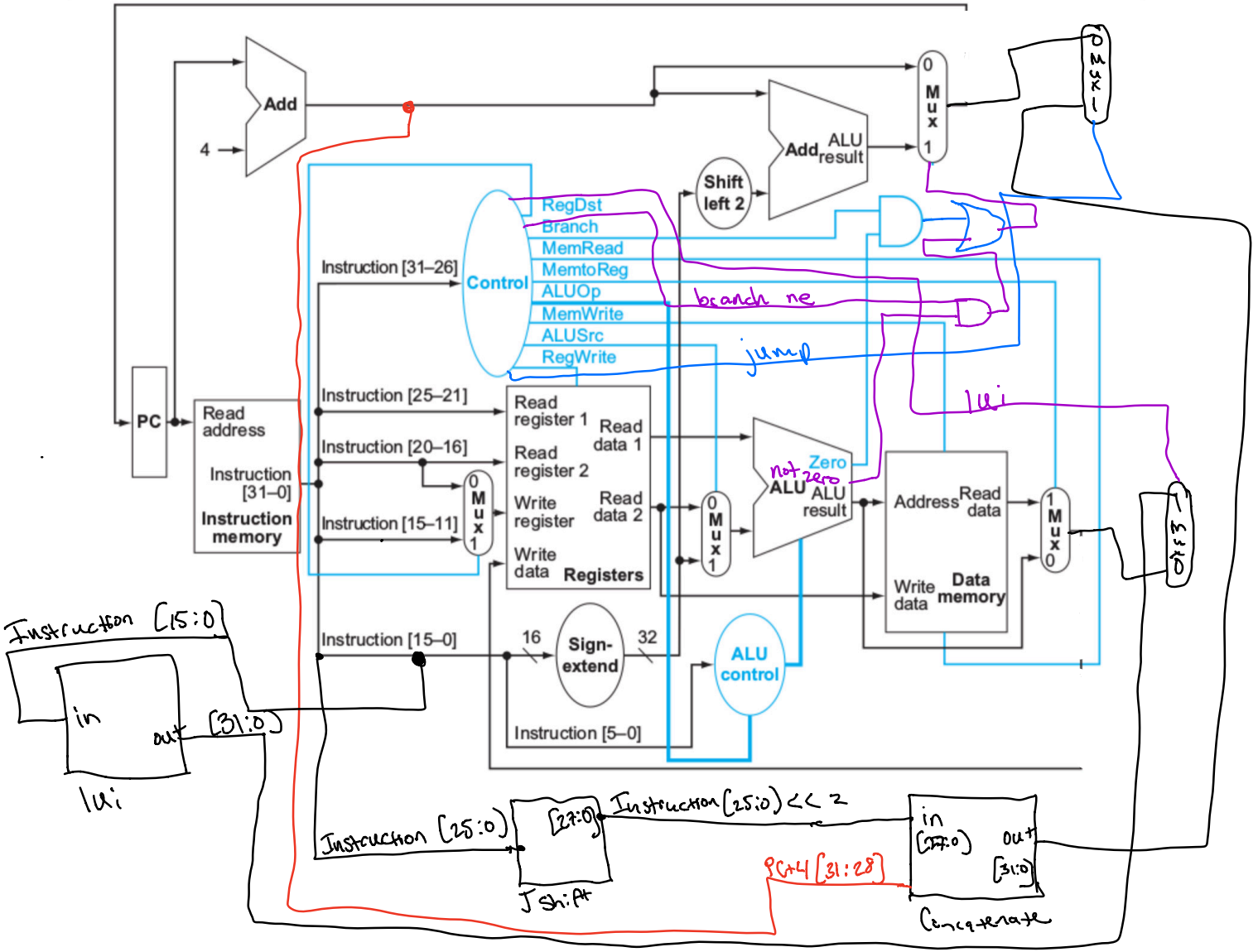
Peter Wallace
Derek Barbosa
Lab 6

- **SLT**: for SLT all we did was modify ALU and ALU Control: in ALU control we added a case where the op was 6'b000000 (for R type instruction), and set the ALUop equal to 3'd5. In ALU, we added a case where instruction = 0 and ALUop = 3'd5 and did (a < b)? 1:0 to implement the SLT.
- **ADDI**: for ADDI we added an instance to the Control unit. In this instance we set RegDst = to 0 since in I type the register destination is the second register in the instruction, we set ALUSrc = 1'b1 to force the ALU to take the immediate as input, and we set RegWrite = 1 to write back to the register.
- **J**: for the J instruction we added a module to shift the lower 26 bits of the instruction to the left by to bits, and then we added a module to concatenate the upper 4 bits of the output of the PC adder (to get PC + 4) with the now 28 bits from the instruction to get the new address of the next instruction. This was then mux'd with the output of the PC mux (taking a new control signal "jump" as select) to get the input to the program counter module.
- **BNE**: for BNE, we added a "not zero" output to the ALU and a branch_ne signal to the control, and then we put those two through an AND gate (when branch_ne and not_zero both are 1, the CPU branches), and then OR'd that output with the output of the AND gate for BEQ that is already implemented (if 1 of the outputs is 1 then branch). That output was then fed into the branch mux as the select.
- **LUI**: for LUI, we added a module that takes the lower 16 bits of the input (for I type instructions the lower 16 bits are the immediate), and puts those 16 bits into the upper 16 bits of a 32 bit wire. The lower 16 bits of the wire are given 0. Then this output goes through a mux, with the other input of the mux being the write_data from the data memory mux, and a new lui signal from control being the select. This output is then fed into the write data port of the register file.


Add
4
0
M
u
x
1
Add
ALU result
Shift left 2
RegDst
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
branch ne
jump
lui
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign-extend
32
ALU control
not Zero
ALU
ALU result
Address
Read data
Write data
Data memory
Instruction [15:0]
in
out
[31:0]
lui
Instruction [25:0]
[27:0]
Instruction [25:0] << 2
PC+4 [31:28]
in
[27:0]
out
[31:0]
Concatenate
Jshift