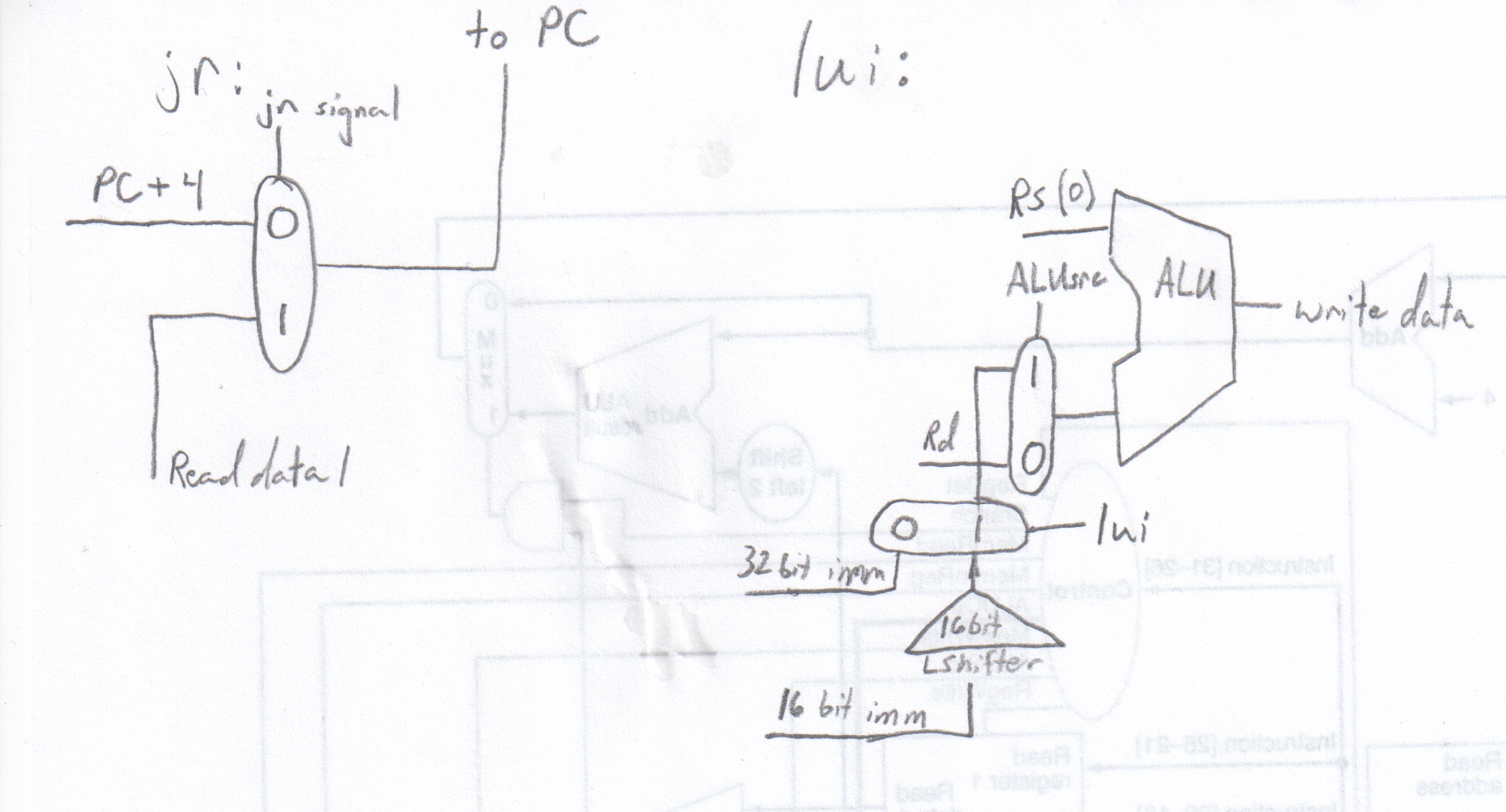
**Homework 6**

Trevor Lund, CprE 381

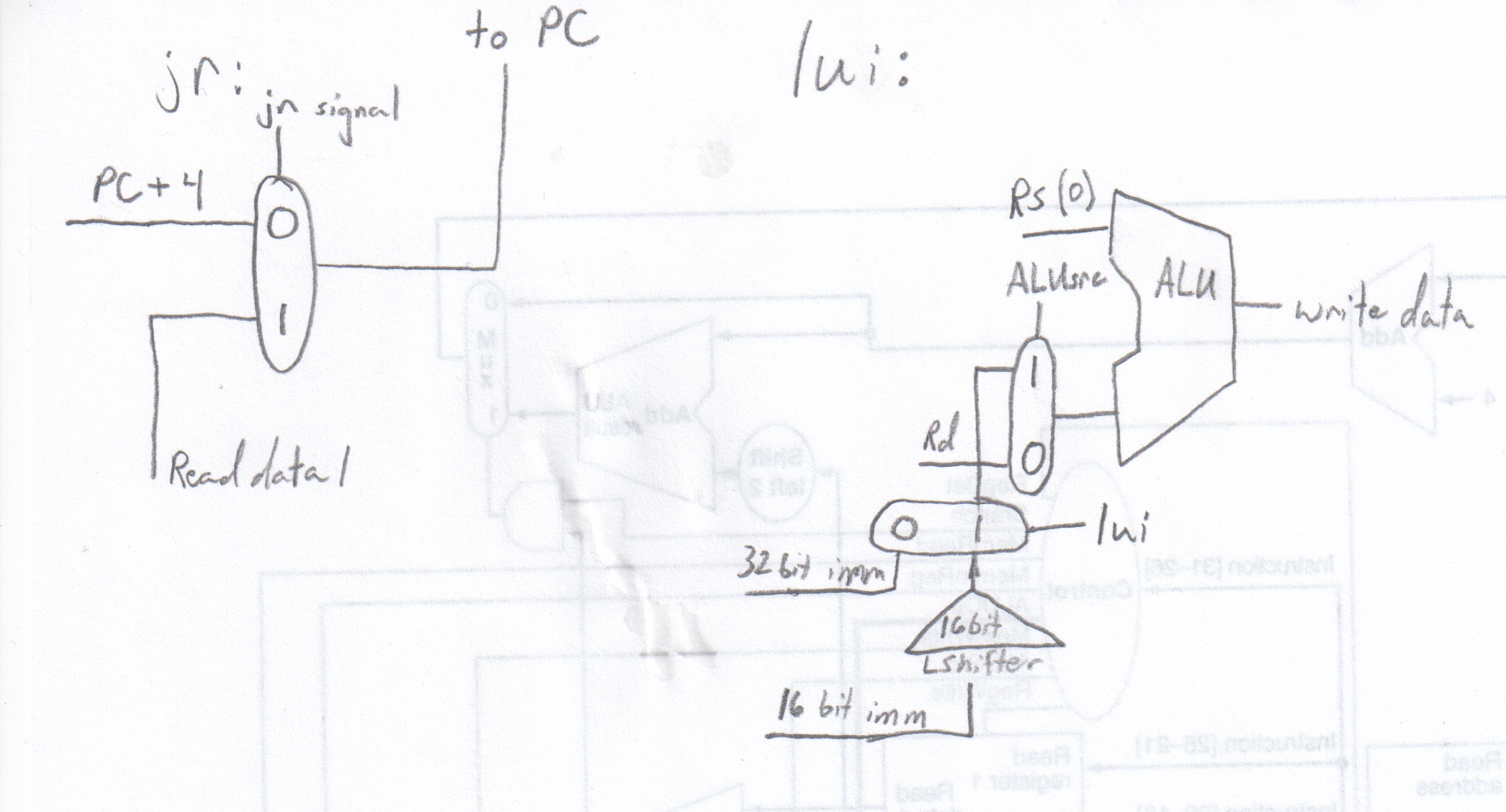
1. Single-Cycle MIPS Enhancements
   1. Figure 4.22 improvements for jr:

|  |  |
| --- | --- |
| **Signal Name** | **Jr** |
| RegDst | X |
| ALUSrc | X |
| MemtoReg | X |
| RegWrite | 0 |
| MemRead | 0 |
| MemWrite | 0 |
| Branch | 0 |
| ALUOp1 | X |
| ALUOp2 | X |
| Jr | 1 |

Picture:

* 1. Figure 4.22 improvements for lui:

|  |  |
| --- | --- |
| **Signal Name** | **lui** |
| RegDst | 0 |
| ALUSrc | 1 |
| MemtoReg | 0 |
| RegWrite | 1 |
| MemRead | 0 |
| MemWrite | 0 |
| Branch | 0 |
| ALUOp1 | 0 |
| ALUOp2 | 0 |
| lui | 1 |

Picture: 

* 1. No, the MemRead control is not necessary because if you don’t need the data from memory, the MemtoReg control will take care of it, making sure it doesn’t get read into the register file.
  2. My friend’s modification will only work if he uses MemRead as the control signal to replace MemtoReg. Addi uses the immediate source for the ALU, but doesn’t read from memory, while lw also uses the immediate source, but does read from memory.
  3. I would remove the datapath from the ALU to the memory’s address port, and I would instead directly connect read data port 1 to that address port.
  4. I’m not sure if we could modify the MIPS ISA. I don’t fully understand word and byte addresses.

1. Architecture and Performance
   1. P2 is more reasonable, because while a mux doesn’t seem to take 100ps, a shifter would have to take 20ps instead of 0.
   2. P1: 400 + 100 + 30 + 120 + 200 + 20 = 870ps  
      P2: 450 + 150 + 100 + 180 + 220 + 90 = 1190ps
   3. P1: 400 + 100 + 200 + 350 + 20 = 1070ps  
      P2: 450 + 150 + 220 + 900 + 90 = 1810ps
   4. P1: 400 + 100 + 30 + 120 + 200 + 350 + 20 + 0 = 1220ps  
      P2: 450 + 150 + 100 + 180 + 220 + 900 + 90 + 20 = 2010ps
2. Fault Tolerance
   1. PC: 0, registers: 0, memory: 0  
      Ori $s0, $zero, 65535 # Result should be 65535, but won’t if there’s a problem with bit 7 of the output Instruction
   2. Test the stuck-at-1 case with ori $s0, $zero, $zero.  
      You can’t test both cases with the same code because the way I tested them requires all the bits to be ones or zeroes.
   3. I can’t see this being a fault, nor do I see a need to provide a test for it, because any MIPS OpCode that uses all zeroes (R-type instructions) won’t use the memory module anyway, resulting in a don’t care signal for MemRead.