Brandon Cortez

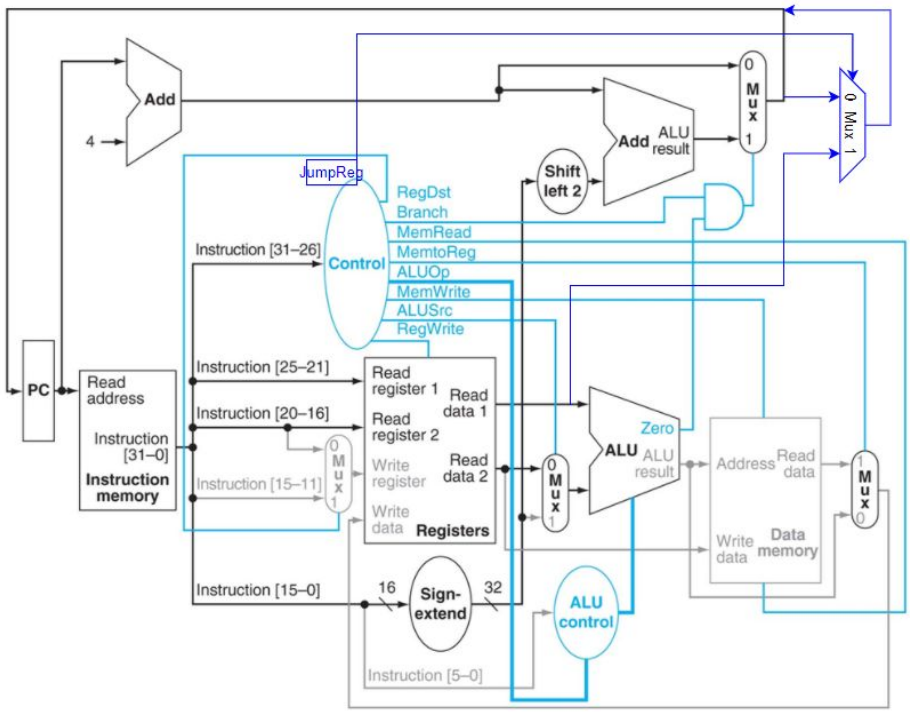
CprE 381

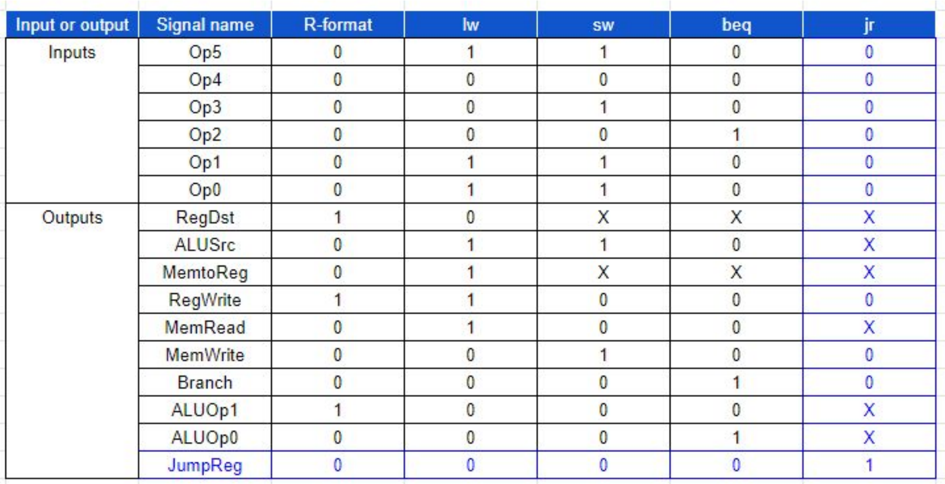
Homework 7

10/16/2020

1. Single-Cycle MIPS Enhancements

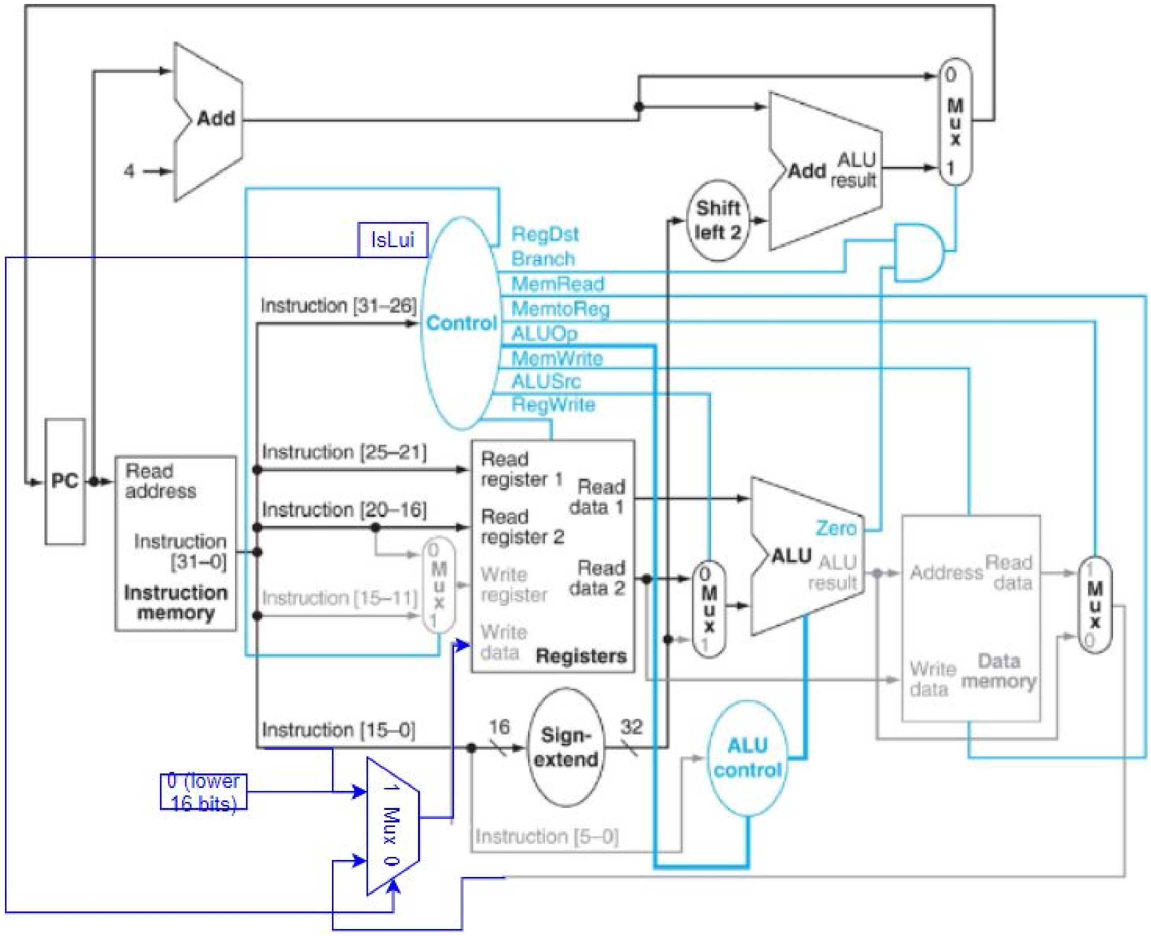
   a. jr instruction





| Input or output | Signal name | R-format | lw | sw | beq | jr |
|---|---|---|---|---|---|---|
| Inputs | Op5 | 0 | 1 | 1 | 0 | 0 |
| | Op4 | 0 | 0 | 0 | 0 | 0 |
| | Op3 | 0 | 0 | 1 | 0 | 0 |
| | Op2 | 0 | 0 | 0 | 1 | 0 |
| | Op1 | 0 | 1 | 1 | 0 | 0 |
| | Op0 | 0 | 1 | 1 | 0 | 0 |
| Outputs | RegDst | 1 | 0 | X | X | X |
| | ALUSrc | 0 | 1 | 1 | 0 | X |
| | MemtoReg | 0 | 1 | X | X | X |
| | RegWrite | 1 | 1 | 0 | 0 | 0 |
| | MemRead | 0 | 1 | 0 | 0 | X |
| | MemWrite | 0 | 0 | 1 | 0 | 0 |
| | Branch | 0 | 0 | 0 | 1 | 0 |
| | ALUOp1 | 1 | 0 | 0 | 0 | X |
| | ALUOp0 | 0 | 0 | 0 | 1 | X |
| | JumpReg | 0 | 0 | 0 | 0 | 1 |

b. lui instruction



| Input or output | Signal name | R-format | lw | sw | beq | lui |
|---|---|---|---|---|---|---|
| Inputs | Op5 | 0 | 1 | 1 | 0 | 0 |
| | Op4 | 0 | 0 | 0 | 0 | 0 |
| | Op3 | 0 | 0 | 1 | 0 | 1 |
| | Op2 | 0 | 0 | 0 | 1 | 1 |
| | Op1 | 0 | 1 | 1 | 0 | 1 |
| | Op0 | 0 | 1 | 1 | 0 | 1 |
| Outputs | RegDst | 1 | 0 | X | X | 0 |
| | ALUSrc | 0 | 1 | 1 | 0 | X |
| | MemtoReg | 0 | 1 | X | X | X |
| | RegWrite | 1 | 1 | 0 | 0 | 1 |
| | MemRead | 0 | 1 | 0 | 0 | X |
| | MemWrite | 0 | 0 | 1 | 0 | 0 |
| | Branch | 0 | 0 | 0 | 1 | 0 |
| | ALUOp1 | 1 | 0 | 0 | 0 | X |
| | ALUOp0 | 0 | 0 | 0 | 1 | X |
| | IsLui | 0 | 0 | 0 | 0 | 1 |

c. Since ALUSrc and MemRead are identical to MemtoReg except for the don't care values, yes either signal could be substituted in for MemtoReg. However, since ALUSrc and MemRead are not identical to each other one of them can not be substituted for the other.

d. I would not have to change the control layout although some control signals during those instructions would now be don't cares. I could add in the data path a way to bypass the ALU altogether to allow the value of readData1 to be read by the memory using a mux.

2. Architecture and Performance

   a. P1 seems most reasonable as latency refers to the length of time required between input and output. Since our single-cycle processor is not carrying out any extremely complex actions it would seem reasonable that the latency would be the lower of the two options.

   b. ALU : I-Mem + Regs + Mux + ALU + Mux

      i. P1: 400ps + 200ps + 30ps + 120ps + 30ps = 780ps

      ii. P2: 500ps + 220ps + 100ps + 180ps + 100ps = 1100ps

   c. lw : I-Mem + Regs + Mux + ALU + D-Mem + Mux

      i. P1: 400ps + 200ps + 30ps + 120ps + 350ps + 30ps = 1130ps

      ii. P2: 500ps + 220ps + 100ps + 180ps + 1000ps + 100ps = 2100ps

   d. lw has the longest path of all four of the instructions, therefore the clock cycle would be the same as part c.

      i. P1: 400ps + 200ps + 30ps + 120ps + 350ps + 30ps = 1130ps

      ii. P2: 500ps + 220ps + 100ps + 180ps + 1000ps + 100ps = 2100ps

3. Fault Tolerance

    a. Stuck-at-0 fault test for write register bit 0

        i. PC: 100 (any # divisible by 4)
        ii. Set all registers to 0
        iii. Instruction will be addi (any instruction that writes to a reg)
        iv. Read register 1 will be the zero register
        v. The Immediate value will be 7 ( any value will do, we are testing where it is written)
        vi. Memories can be any values for this test, we do not care
        vii. Destination reg will be 3 (any register that has a 1 in bit 0 will do)
        viii. If there is no fault on bit 0 the value of 7 will be written to register 3, however, if there is a stuck-at-0 fault on bit 0 then it will be written to register 2

    b. Stuck-at-1 fault test for write register bit 0

        i. PC: 100 (any # divisible by 4)
        ii. Set all registers to 0
        iii. Instruction will be addi (any instruction that writes to a reg)
        iv. Read register 1 will be the zero register
        v. The Immediate value will be 7 ( any value will do, we are testing where it is written)
        vi. Memories can be any values for this test, we do not care
        vii. Destination reg will be 2 (any register that has a 0 in bit 0 will do)

viii. If there is no fault on bit 0 the value of 7 will be written to register 2, however, if there is a stuck-at-1 fault on bit 0 then it will be written to register 3

ix. No, one test can not test for both faults. Since only one instruction can be run per test you can only write to a single register per test which means you cannot know for sure if the registers ending in the opposite bit than the one you chose behave as expected or if there is a fault that is present that forced the value to the one you had already chosen

c. Yes, if the processor has a stuck-at-1 fault on bit 0 of the write register signal the processor is still usable. The stuck-at-1 fault limits the reg file to only be able to write to odd registers (1 in bit 0) however those registers that it can write to should behave normally and be able to carry out any required functions. This fault essentially only limits the processor to be able to use half the number of registers it normally can.