HW #7

CSEE W3827 - Fundamentals of Computer Systems Spring 2022

Prof. Rubenstein
Due 4/8/2022, 5pm

Topics: CPU Design Note that this homework has 5 problems on 2 pages.

1. Give the bit sequences (you can write them in decimal or hex per field for convenience if desired) for the following instructions (please make a clear partitioning of the different fields, see below). Slides 27-31 of lecture 8 should be helpful, and also slides 13-17 of lecture 7:

   (a) lw $t3, 40($s2)

   (b) add $t3, $s2, $s1

   (c) addi $t3, $s0, 37

   (d) beq $s1, $s3, LABEL (where LABEL is 7 instructions before **this** beq instruction)

   (e) j LABEL (where LABEL is the address 2088 in decimal).

2. Explain (in one sentence) why op-code and function fields are separate (i.e, why didn't the architecture design a combined op-code/function field)?

3. Consider the effect that a stuck-at-0 (or stuck-at-1) fault would have (i.e., regardless of what the signal should be for some selector/enabler, it remains stuck on 0 (or 1)) in the single cycle architecture depicted in slide 15 of lecture 11, for following nine signals. Which of the following instruction types, if any, will not work correctly, and what do they do wrong? Consider:

   - R-type
   - I-type
   - beq
   - lw
   - sw

   Information provided in slides 106-127 from Lecture 11 might be helpful. Consider each case below separately.

   (a) RegWrite= 0

   (b) ALUOp0= 0

   (c) ALUOp1= 1

   (d) Branch= 1

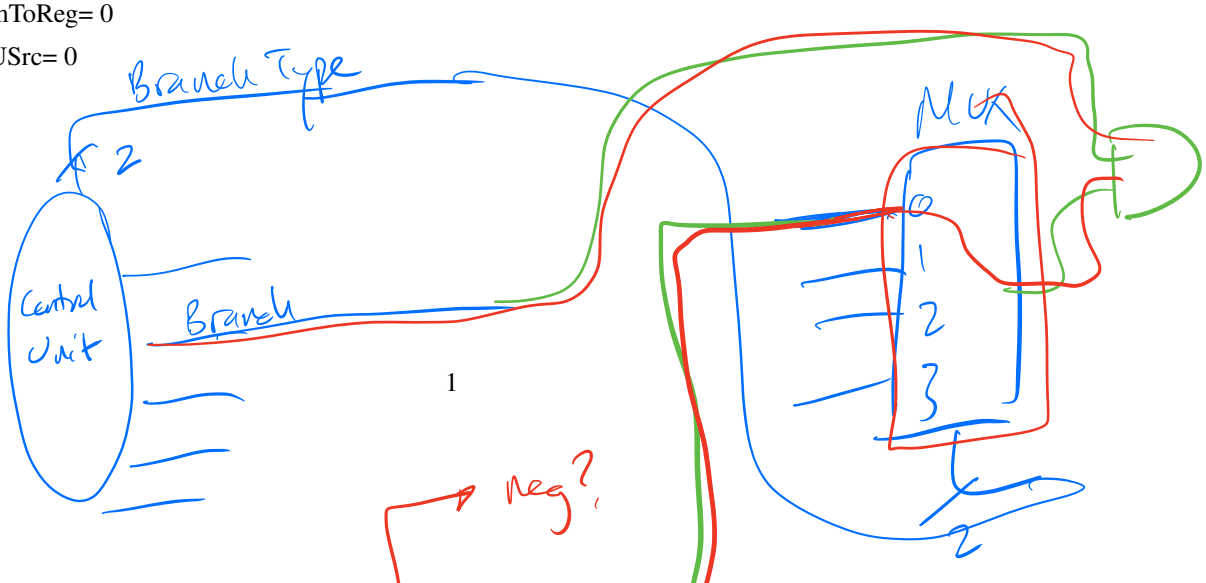
   (e) MemRead= 0

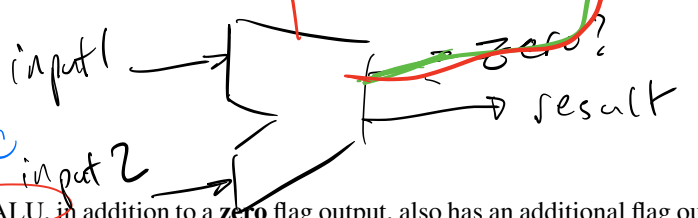
   (f) MemWrite= 0

   (g) RegDest= 1

   (h) MemToReg= 0

   (i) ALUSrc= 0





4. Suppose the MIPS ALU, in addition to a **zero** flag output, also has an additional flag output for "negative" which equals 1 whenever the computation result of the ALU is negative (i.e., has high-order bit set to 1). Show how to modify the data path to include, in addition to beq, the branch instructions bne (branch if not equal), bltz (branch if less than 0), and bgez (branch if greater than or equal to 0).

   You may assume the following:

   - Control produces an additional 2-bit output, **BranchType** that equals 00 for beq, 01 for bne, 10 for bltz and 11 for bgez.
   - The branch instructions bltz and bgez will make the second input register equal the $zero register.

   Hint: the circuitry that determines whether or not to take the branch (i.e., the conditional that feeds into the MUX making that determination) gets a bit more complicated

5. Suppose we wish to modify MIPS memory-access instructions so that the constant affects the value being stored / read from rather than offsetting the address in which the value is read from / stored. For instance, the lw command would have the form:

   ```
   lw $s0, $s1, 13
   ```

   For the above lw instruction, if the value in $s1 equaled A, then memory address A would be read, 13 would be added to this value, and the result of the sum would be stored in $s0.

   ```
   sw $s0, $s1, 13
   ```

   For the above sw instruction, if the value in $s1 equaled A, then memory address A would be written to with the value of 13 plus the value in $s0.

   (a) How should the MIPS architecture be modified to implement the above sw instruction (in place of the existing sw).

   (b) How about to implement lw?

   (c) Show how both be done simultaneously with a single ALU. The trick is to have MUX's which choose the order of ALU and memory access.