Score: _____ Name: _____

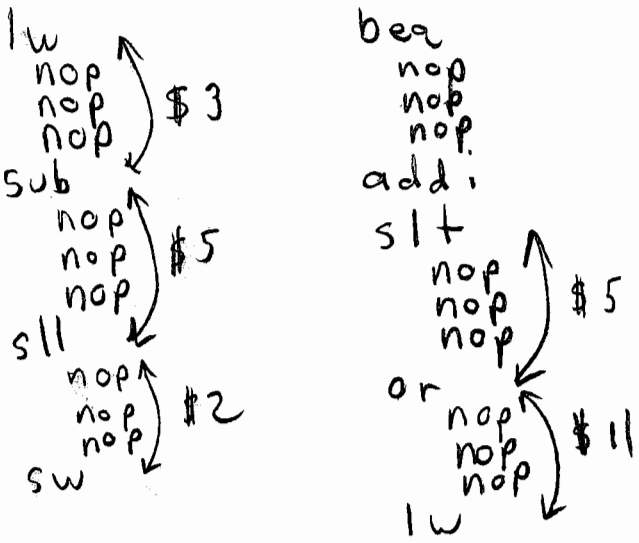
ECE 3055 Quiz 5 Wednesday, February 24

The program below is executed on the 5 stage pipelined MIPS processor design described in chapter 4. Answer the following questions about this program.

```
loop_top:   lw    $3,100($5)
            sub   $5,$5,$3
            sll   $2,$5,3
            sw    $3,200($2)
            beq   $2,$3,foobar
            addi  $6,$6,-2
            slt   $5,$5,$11
foobar:     or    $11,$5,$9
            lw    $9,100($11)
            beq   $8,$11,loop_top
```

Assume the control unit **does not have** any hazard detection, forwarding, a new branch compare circuit, or automatic branch flushing, and that the register file **will not** write and then read a new register value in one clock cycle. Rewrite the code sequence by only adding the minimum number of NOP instructions (*do not reorder or change instructions*) to eliminate all potential data and branch hazards. Assume other non-NOP instructions follow the last branch in the original code sequence above.

Total number of NOPs required 21



Next, assume the control unit is fully improved as outlined in the text by adding the hazard and forwarding unit, adding automatic branch flushing with a new compare unit to the decode stage along with forwarding muxes, and the register file writes and then reads a new value in a single clock cycle. Determine the number of clock cycles required to complete the first loop execution (i.e. executes code in loop and branches back to top of loop and is just ready to fetch lw again) of the original code sequence. Assume the inner branch (i.e., beq) **is taken**.

If there were no hazards or branch flushing, the original program would only require 8 clock cycles for a single loop execution. (***do not include the time to initially fill the pipeline at power up***). lw +2 beq taken

But this program will need to stall and/or flush the pipeline an additional 3 clock cycles so

a total of 11 **clock cycles** is required for execution (***do not include the time to initially fill the pipeline***).

This program achieves a CPI (clocks per instruction) of 1.375 .(*do not include time to fill the pipeline here*)

11/8