Score: _____ Name: _____

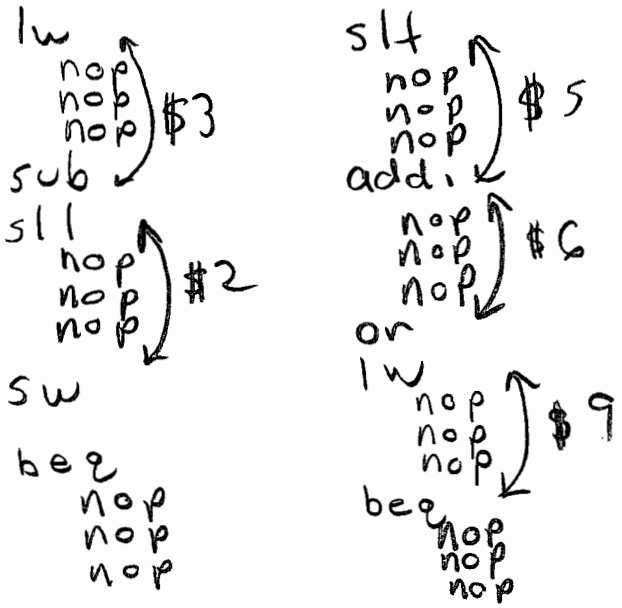
ECE 3055 Quiz 5 February 23

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,$3,$4
            sll     $2,$3,3
            sw      $3,200($2)
            beq     $2,$8,foobar
            slt     $5,$5,$6
            addi    $6,$5,-2
foobar:     or      $7,$6,$9
            lw      $9,100($6)
            beq     $9,$6,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 (with forwarding) to the decode stage along with the original register 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 ten executions of the loop code before it exits at the bottom of the loop. Assume the inner branch (i.e., beq) **is always taken**.

If there were no hazards or branch flushing, the original program would only require 80 clock cycles for ten loop executions. (***do not include the time to initially fill the pipeline at power up***). 21w + 2beq

But this program will need to stall and/or flush the pipeline an additional 39 clock cycles so −1beq at end

a total of 119 **clock cycles** is required for ten loop executions (***do not include the time to initially fill the pipeline***). 119/80

This program achieves a CPI (clocks per instruction) of 1.48 .(*do not include time to fill the pipeline here*)