CS 330 Homework 4-C

Questions 1 to 3 concern the following code segment.

```
i0: or $t2, $t0, $t1
i1: or $t3, $t1, $t2
i2: or $t4, $t2, $t3
```

We want to execute this stream of instructions on a 5-stage pipelined MIPS processor.

1. Assume that the processor does not implement forwarding. Draw a stage diagram[1] showing the execution of each stage of the three instructions, assuming the processor fetches the next instruction on each clock cycle. Identify all the data hazards.

2. Again, assume that the processor does not implement forwarding. Draw another stage diagram, only this time insert `noop`s between regular instructions, as necessary, in order to stall the pipeline long enough to ensure that the instructions execute correctly.

3. In ALU-to-ALU forwarding, the output of the ALU for a particular instruction is available as an input of the ALU of the following instruction at the beginning of the next clock cycle. Draw another diagram like you did in question 1 showing the execution of each stage of the pipeline, this time assuming that the processor uses ALU-to-ALU forwarding. Show the forwarding by drawing a line from the ALU output of one instruction to the ALU input of the instruction which needs it.

Questions 4 to 7 concern the following segment of MIPS code.

```
        andi $t0, $t0, 0x0
label:  add $s0, $s0, $s1
        addi $t0, $t0, 1
        bne $t0, $t7, label
        ori  $s3, $s3, 0xA6A6
```

Assume the 5-stage pipeline discussed in lecture, that `$t7` initially contains 5, and that data hazards are properly dealt with using forwarding. Calculate the number of clock cycles required to completely execute this code under the following assumptions.

4. The processor inserts a `nop` after each branch. (Hint: draw a diagram like figure 4.32 of the text.)

[1] You can find an example of a stage diagram in figure 4.28 on page 279 of the text.

5. The processor uses branch prediction and assumes that a backwards branch will never be taken. When it guesses incorrectly, the processor inserts a `nop` into the pipeline.

6. The processor uses branch prediction and assumes that a backwards branch will always be taken. When it guesses incorrectly, the processor inserts a `nop` into the pipeline.

7. The processor uses branch prediction and the branch is always correctly predicted.