State the % performance drop versus an ideal pipeline with CPI of 1.0 for the two questions. Briefly describe the mathematical derivations used to arrive at these answers.

$$CPI = 1 + \frac{1 * N_{\text{RAW,1 cycle}} + 2 * N_{\text{RAW,2 cycles}}}{N_{\text{insn}}}$$
$$\text{slowdown} = \frac{CPI - 1}{1}$$

```
sim_num_raw_q1_1_cycle      9206516
sim_num_raw_q1_2_cycle     88182314
sim_num_raw_q2_1_cycle     68796288
sim_num_raw_q2_2_cycle     20126394
sim_num_insn              279373007
```

$$CPI_{\text{q1}} = 1 + \frac{1{*}9206516 + 2{*}88182314}{279373007} = 1.6642$$
$$slowdown_{\text{q1}} = \frac{CPI - 1}{1} = 66.42\%$$
$$CPI_{\text{q2}} = 1 + \frac{1{*}68796288 + 2{*}20126394}{279373007} = 1.3903$$
$$slowdown_{\text{q2}} = \frac{CPI - 1}{1} = 39.03\%$$

Briefly explain how your microbenchmark collected statistics validate the correctness of your code for the first problem statement. Feel free to refer to comments within the `mbq1.c` file, as needed. Specify which compilation flags you used.

This microbenchmark is used to test the RAW hazard of processor 1 in the lab handout. It's built and tested with `-O2` optimization.

```
int main(void) {
    int a = 0;
    int b = 0;
    while (a < 1000000) {
        a++;
        asm("nop"); // By including or commenting out this nop, we can make this a 1-
cycle stall or 2-cycle stall
        b = b + a;
    }
    return a + b;
}
```

We can see the difference when including and excluding the `nop` and can verify the correctness of `sim-safe`, calculation details see the comment in `mbq1.c`.

```
$L4:
    addu        $3,$3,1
#APP
    nop
#NO_APP
    addu        $4,$4,$3
    slt $2,$5,$3
    beq     $2,$0,$L4
```

```
$L4:
    addu        $3,$3,1
    addu        $4,$4,$3
    slt $2,$5,$3
    beq $2,$0,$L4
```