This lab investigates using predication (executing code along more than one possible path of a branch) as opposed to using speculation (executing along the predicted path of a branch).

1. In number theory it is often necessary to find large primes. It is actually easier to find pseudoprimes and just check if they are prime. An integer $p$ greater than two which satisfies $k^{p-1} \equiv 1 \pmod{p}$ for $k$ in a small carefully chosen subset of $\{2, 3, 5, \ldots\}$ is called a pseudoprime (with respect to that subset). Fermat’s little theorem ensures that real primes are always pseudoprime. The reserve is true with a very high probability. For example, the first pseudoprime (for any subset) which is not prime is $1729 = 7 \cdot 13 \cdot 19$.

2. Consider the following algorithm to find $k^{p-1} \pmod{p}$. First $p-1$ can be written in binary as

$$p - 1 = b_{n-1}b_{n-2} \ldots b_3b_2b_1b_0$$

so that

$$k^{p-1} = \prod_{b_j=1} k^{2^j}$$

In order to compute $k^{p-1}$ we just have to start with one and multiply by $k^{2^j}$ if the j-th bit is one. Also note that $k^{2^{j+1}} = (k^{2^j})^2$.

3. Show that the following MIPs assembly would implement the computation (ignoring the problem of overflow which could be alleviated — how?). We assume that $k$ is in t1, $p$ is in t2. We put $p - 1$ in t3 and shift it to the right on each iteration before checking the least significant bit. We initialize t4 to contain $k$ and square it on each iteration. We keep the final result in t6:

```
addi $t3, $t2, -1 ; initialize p-1
mov $t4, $t1 ; initialize k
addi $t6, $zero, 1 ; initialize result to 1

loop:
    andi $t0, $t3, 1 ; t0 will hold j-th bit: 0 or 1
    beq $t0, $zero, bitzero
    mult $t6, $t4 ; result in hi:lo
    mflo $t6 ; update product

bitzero:
    srl $t3, $t3, 1 ; shift p-1 right one bit
    mult $t4, $t4 ; result in hi:lo
    mflo $t4 ; t4 has been squared
    bne $t3, $zero, loop
```

**Assignment** Discuss the branch prediction for the code above. Then, suppose you have a processor (such as the Itanium) which will accept *predicated* code. Can the algorithm above be improved upon? Why or why not?