CS1760: Multiprocessor Synchronization Due 6 October 2022

Homework 2

Professor: Maurice Herlihy *TA: James Scherick*

Problem 1. Consider a variant of Peterson's algorithm, where we change the `unlock` method as shown in Figure 1. Does the modified algorithm satisfy deadlock-freedom? What about starvation-freedom? Sketch a proof showing why it satisfies both properties, or display an execution where it fails.

```
public void unlock() {
  int i = ThreadID.get();
  flag[i] = false;
  int j = 1 - i;
  while (flag[j] == true) {}
}
```

Figure 1: The revised unlock method for Peterson's algorithm used in Assignment 3

Problem 2. There is a theorem that any deadlock-free mutual exclusion algorithm for n processes must use at least n shared registers. For this exercise, we examine a new algorithm that shows that the theorem's space lower bound is tight. Specifically,

> **Theorem**: There is a deadlock-free mutual exclusion algorithm for n processes that uses exactly n shared bits.

To prove this theorem, we study the One-Bit algorithm, which uses exactly n bits to achieve mutual exclusion. (This algorithm was developed independently by J.E. Burns and L. Lamport and is interesting in that it uses the minimum possible shared space.) The pseudocode is shown in Figure 2.

```

Initially : all entries in Flag are false
repeat
   Flag[i] := true; j := 1
   while (Flag[i] = true and j < i) do
       if Flag[j] = true then
           Flag[i] := false; await Flag[j] = false;
       j := j+1;
 until Flag[i] = true;
for j:=i+1 to n do await Flag[j] = false;

{enter critical section}

Flag[i] := false;
```

Figure 2: Pseudocode for the One-bit algorithm

The way the One-Bit algorithm works as follows: First, a thread must indicate that it is contending for the critical section by setting its bit to true. Then, it loops and reads the bits of all threads with smaller identifiers than its own. If all of these bits are false (while its own bit is true), then the thread exits the loop. Otherwise, the thread sets its bit to false, waits until the bit it is waiting on becomes false, and starts all over again. Afterwards, the process reads the bits of all threads that have identifiers greater than its own, and waits until they are all false. Once this check passes, the thread can safely enter the critical section.

- Prove that the One-Bit algorithm satisfies mutual exclusion.
- Prove that the One-Bit algorithm is deadlock-free.

Problem 3. Use Amdahl's Law to resolve the following questions:

- Suppose a computer program has a method M that cannot be parallelized, and that this method accounts for 40% of the program's execution time. What is the limit for the overall speedup that can be achieved by running the program on an n-processor multiprocessor machine?
- Suppose the method M accounts for 30% of the program's computation time. What should be the speedup of M, so that the overall execution time improves by a factor of 2?
- Suppose the method M can be sped up three-fold. What fraction of the overall execution time must M account for in order to double the overall speedup of the program?

Problem 4. A *regular* register has the property that if a `read()` call overlaps a `write()` call, then the value returned can be either the new value or the old. A `read()` that does not overlap a `write()` returns the last value written. (Note that a regular register is not linearizable.)

An *safe* register has the property that if a `read()` call overlaps a `write()` call, then the value returned can be arbitrary. A `read()` that does not overlap a `write()` returns the last value written.

An *wraparound* register has the property that there is a value v such that adding 1 to v yields 0, not $v + 1$.

If we replace the Bakery algorithm's shared variables with either regular, safe, or wrap-around registers, then does it still satisfy (1) mutual exclusion, (2) first-come-first-served ordering?

You should provide six answers (some may imply others). Justify each claim.