# Programming Assignment #2 - SPIN

ComS 412/512 - Due: Wednesday April 10th, 2024

## Problem Specification

In this assignment, you will model a simple RAM allocation protocol that could be used by an operating system. Your model has 3 processes ($P_0$, $P_1$, $P_2$), each requiring a different number of Mbytes of RAM to run ($N_0$, $N_1$, $N_2$), but the system has only $M$ Mbytes of RAM available for use. The system allocates a single Mbyte of RAM at a time. During each execution step, the operating system will non-deterministically chooses a process to activate. A process $P_i \in P$ activates according to the following rules:

- If $P_i$ is *idle* (not *requesting* or *running*), it can begin *requesting* RAM
- If $P_i$ is *requesting* RAM and there are available Mbytes, the process can access one available Mbyte:
  - Remove 1 Mbyte from available memory.
  - Add 1 Mbyte to the RAM being accessed by $P_i$.
  - If $P_i$ has *accessed* $N_i$ Mbytes of memory, it will immediately stop *requesting* and begin *running*.
- If $P_i$ is *running*, it can continue *running* OR it can complete and release all memory:
  - Release all $N_i$ Mbytes of memory *accessed* by $P_i$.
  - Add $N_i$ Mbytes of memory back to the available RAM.
  - $P_i$ becomes *idle*.

An example of a system execution path, starting from initial state $M = 2, N_0 = 2, N_1 = 2, N_2 = 1$, is as follows:

```
P0 requests RAM.
P0 accesses 1 Mbytes (1 Mbytes left available)
P0 accesses 2 Mbytes (0 Mbytes left available)
Process 0 starts running...
P1 requests RAM
Process 0 completes (2 Mbytes now available)
P0 requests RAM
P0 accesses 1 Mbytes (1 Mbytes available)
P1 accesses 1 Mbytes (0 Mbytes available)
P2 requesting RAM
```

A significant problem with this system model is deadlocks. It is possible that all (or a subset of) the processes are *requesting*, but there is no available RAM and no process is *running*. In this scenario, no progress will be made and none of the processes will ever run. For example, the previous execution path ends in a deadlock.

# Assignment

This assignment requires you to use SPIN. Instructions, guides, and examples can be found in the 'files' section of the Canvas page, and the SPIN Homepage is a good place to start. Make sure you can both run and understand the example programs before continuing.

## Part 1 - Progam Creation

In this programming assignment, you should use SPIN to model the RAM allocation protocol as specified above. A starting template has been provided, but you are welcome to write a program from scratch, if you prefer. **Your program should:**

- **Correctly model the problem for any initial $M$ and $(N_0, N_1, N_2)$ values.**
- **Evaluate whether or not there is a potential deadlock in the system.**

In your program, $M$, $N_0$, $N_1$, and $N_2$ should be only defined in the initialization function or with $\#define$. Additionally, an execution path should be printed (similar to the one provided in the above example) and an LTL safety property should be used to check for deadlocks, so that a finite path can be given as a counter-example. *Note: a deadlock is reachable iff there is a future state where all processes are requesting RAM but there is no RAM available.*

At the end of this part, you should have a SPIN program named **processRAM.txt** which models the RAM allocation protocol for any $M$ and $(N_0, N_1, N_2)$ values, and will return a trace to a deadlock if one exists (in a format similar to the example path given earlier).

## Part 2 - Deadlock Analysis

**Add comments at the top of your program** answering each of the following questions:

1. Determine the relation between $M$ and $(N_0, N_1, N_2)$ to determine whether there will be a deadlock.
2. Without removing the 1-Mbyte-at-a-time allocation rule, could you updated the model so that no deadlocks exist, no matter what values of $M$ and $(N_0, N_1, N_2)$ were provided, as long as $M \geq \max\{N_0, N_1, N_2\}$? How would you accomplish this in your program? Be specific; reference code lines or add comments if needed.

# Submission

You should submit a program .txt file called **processRAM.txt**. When run, the program should determine whether there is a deadlock given the specified $M$ and $(N_0, N_1, N_2)$ values. If a deadlock is found, it should write a trace to the deadlock. Otherwise, the program should run with no errors. Additionally, the program should have the answers to the questions from part two commented at the top. Make sure your file has good documentation (use "*" to start a comment), and your name at the top.