CMSC 312 Project Part 2 Description 2022
Operating system simulator

Multi-threading and multi-CPU implementation
The last part of the project will take the simulation software developed in parts 1&2 and move it to a multi-threading and multi-CPU architecture. The multi-threading part must be implemented with the usage of a selected threading library. The multi-CPU part must be simulated, by spreading the available number of threads equally among the simulated CPU number.

Requirements:

1. Multi-threading implementation of process lifecycle. 4 threads should be used to schedule processes. This means that the scheduler must be modified to service 4 threads at once and PCB must hold information about to which thread a given process is assigned. Threads must be implemented using a selected threading library and use hardware threads on CPU, leading to a parallel implementation of the OS simulator.
2. As now 4 processes will be running at the same time, your critical section solution must be able to handle multiple processes trying to access their respective critical sections. Your mutex lock/semaphore/monitor must monitor all threads at once and offer mutual exclusion of competing processes.
3. Multi-thread implementation must be extended to a multi-CPU implementation. You are required to simulate 2 CPUs that each have their own schedulers and registers, but share main and virtual memory. Multiple CPUs should be simulated by dividing threads into 2 equal pools and using each of them to run separate instances of CPU (dispatchers, schedulers, etc), but using shared memory.
4. Upon creation the process is assigned to one of the processors for running. It is sufficient to implement hard affinity, but implementing soft affinity and process migration mechanisms will result in bonus points.

General rules for the project

- This is an open-ended project and the design skills, capability of adapting to specifications where necessary, as well as imagination and creativity of solving the required tasks play a crucial role.
- Project must be fully functional, and the code of project must compile and run.
- Project must be a compound system of modules working together, not a collection of non-integrated parts.
- Project must be done individually, not code sharing or copying / modifying is allowed. • Students are allowed to discuss and consult theoretical solutions and approaches among themselves.
- Source code must be delivered for grading, as well as an executable version of the project.
- Student is obliged to deliver a .pdf documentation of the project, discussing the implemented solutions, software, and hardware requirements for the project to run, as well as guidelines on how to run the project.