CS 307 Term Project

Implementation detail and Explanation of Threads

In total, you will have 5 threads:

Producer Thread for File Input

This thread will receive the file paths from a file called **inputSequence.tx**t. A line in inputSequence.txt will contain the file path of the assembly file and the time the thread has to wait before the next file path is received. After reading the file path, the thread will open the assembly file, convert it to binary and write it to a binary file with the same name as the binary file (ie. If the name of the assembly file is assemblyInput1.asm, the binary file will be named as assemblyInput1.bin). To store the file name of the binary file, you will create a new instance of ProcessImage and store it in File Input Queue. Then the thread will sleep for the amount of time given in inputSequence.txt and receive the next file path. The thread should stop receiving file paths if the File Input Queue is full and it should sleep until there is an empty place in the queue.

The reason this thread is called as a Producer is actually because it produces input for the File Input Queue. As the File Input Queue is a bounded buffer, we want you to implement the producer implementation of producer-consumer problem to make the sleep if the queue is full and wake up if there is an empty space.

Consumer Thread for File Input

This is the consumer thread for File Input Queue. This thread will remove an item from the File Input Queue and then will look at the memory if there is an available space for the process. While there is no available space, it will sleep for 2 seconds and try again. After the thread finds an available space, it will open the binary file stored in the ProcessImage instance and load the binary version of instructions into the memory. Then, you will set base and limit register accordingly and add it to the end of the Ready Queue

To keep track of available spaces in memory, you will use a bitmap. Each character in the memory will be represented by one bit. If an index in the memory is not free, the bitmap will store 1 and if the index is free, the bitmap will store 0. You will use first fit algorithm for memory allocation.

As this thread is the consumer part of the File Input Queue, the thread should sleep if there are no elements in the queue and wake up if there is an element. This should be done using the consumer implementation of the Producer-Consumer problem.

Producer Thread for Console Input

This thread will receive an unsigned integer as an input from console and store it in Console Input Queue. The thread should stop receiving input if the Console Input Queue is full and it should sleep until there is an empty place in the queue.

The reason this thread is called as a Producer is actually because it produces input for the Console Input Queue. As the Console Input Queue is a bounded buffer, we want you to implement the producer implementation of producer-consumer problem to make the sleep if the queue is full and wake up if there is an empty space.

Consumer Thread for Console Input

This is the consumer thread for Console Input Queue. This thread will remove an item from the Console Input Queue and then will look at the Blocked Queue. If there is an item at the Blocked Queue, then it will remove a process from the Blocked Queue, set the value of its register V to the item it removed from Console Input Queue and put the process to the end of the Ready Queue. While there are no processes in Blocked Queue, the thread will sleep for 2 seconds and try again.

As this thread is the consumer part of the Console Input Queue, the thread should sleep if there are no elements in the queue and wake up if there is an element. This should be done using the consumer implementation of the Producer-Consumer problem.

Operating System Thread

This is the thread that is already implemented in the first phase of the project. To recap, this is the thread which CPU and operating system related operations will execute. As there are multiple processes waiting to run, these should be a scheduling algorithm for the processes. You will use Round Robin Scheduling with Quantum = 5 for the scheduling.

This thread will remove the first process from the Ready Queue and it will execute the process in the fetch-decode-execute cycle until its quantum is finished or the process is finished or the process is blocked waiting for input.

If a process finishes its quantum time and it is not finished or blocked, then it is sent back to the end of the Ready Queue.

If a process finishes executing before its quantum if over, then the thread should empty the memory allocated by the process.

The I/O operations will be handled in the operating system thread. At the end of each fetch-decode-execute cycle, you should check whether the current instruction is an I/O operation. If the process should give an output, then it will give it using System.out.println(“ ”)function to the console. If the process should receive an input, the process should be sent to the end of the Blocked Queue and the operating system thread should receive the next waiting process.

While there are no processes waiting to be executed, the thread should sleep for 2 seconds and then check Ready Queue again.