**CS 3343 Operating Systems Exam 1 100 points**

**Due February 19 at 5pm.**

**No late exams. Exams must be received by February 19 at 5pm.**

**Write all code in the Java programming language to create the following software. Show all screenshots of running programs. Include all source code. Be sure to use the objects using the notation from the book and in class. Do not check for file existence or open errors. No AI or GPT use. Cite references. Microsoft Word only. Include all .java files as attachments as well.**

**Email your completed exam to me at** [**harringp@nsuok.edu**](mailto:harringp@nsuok.edu)

**Send email exclusively via** [**https://mail.google.com/**](https://mail.google.com/)

**Sign in with your NSU email and password**

**One exam submission per student.**

**13 questions**

**Part 1: Answer the following questions:**

A black screen with white text

Description automatically generated

1. There are multiple models of the M model CPU from Apple now, but to answer this specific question, I am going to reference the M1 CPU. When I searched around online, I couldn’t find specific information regarding the CPU, maybe I just couldn’t find the right link. However, I did find one of the official Apple docs that talks about the CPU, and I learned that the CPU is a fully housed chip with a CPU, GPU, and other things on board the single chip.

With that information, I did find out that the GPU can handle up to 25,000 threads at a time. When it comes to video processing and things that involve using the GPU. I also learned more about the unified memory architecture that the new M model CPUs take advantage of. I will talk about that in more detail in later questions though.

Lastly, I did find info on the core structure of the CPU. I learned that the M model CPU uses what is called performance cores and efficiency cores. The M1 has 8 of these cores, with 4 being performance and 4 being high efficiency. These cores are built to improve performance in single threaded tasks, and the 4 of them can be used in multithreaded scenarios as well. **(Link:** [**https://www.apple.com/ua/business/mac/pdf/Apple-at-Work-M1-Overview.pdf**](https://www.apple.com/ua/business/mac/pdf/Apple-at-Work-M1-Overview.pdf)**)**

1. I found a ton of interesting information for this question in the Apple documentation. From what I can tell, Mac uses this system called Mach. Mach 3.0 according to the webpage was originally conceived as a simple, extensible, communications microkernel. However, it is capable of running as a stand-alone kernel. Mach provides object-based APIs, highly parallel execution, among other things. Most importantly for this question though, Mach provides a complete set of IPC primitives, including messaging, RPC, synchronization, and notification.

Scrolling further down the documentation, we can see a section dedicated to IPC. It says that Mach supports a client/server system structure in which tasks access services by making requests of other tasks via messages sent over a communication channel. The endpoints of these channels are called ports. The forms of IPC provided by Mach include message queues, semaphores, notifications, lock sets, and remote procedure calls (RPCs).

**Main Link:** [**https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/Mach/Mach.html**](https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/Mach/Mach.html)

**Extra Links: (I found some interesting information at these places that is both related to the question, but unrelated to what was specifically asked if you wanted to check them out)**

* [**https://developer.apple.com/documentation/uikit/interprocess\_communication**](https://developer.apple.com/documentation/uikit/interprocess_communication)
* [**https://developer.apple.com/library/archive/documentation/mac/pdf/Interapplication\_Communication/Intro\_to\_IAC.pdf**](https://developer.apple.com/library/archive/documentation/mac/pdf/Interapplication_Communication/Intro_to_IAC.pdf)

1. According to the book, a socket is defined as an endpoint for communication. For example, a pair of processes communicating over a network employs a pair of sockets. **(Book - 146)**
2. According to GeeksForGeeks, there are two types of network operating systems. These include Peer-to-Peer and Client/Server. The function of a Network OS is to create and manage user accounts on the network, control access to resources, provide communication services between devices, monitor and troubleshoot, and configure and manage the resources on the network. **(Link:** [**https://www.geeksforgeeks.org/what-is-a-network-operating-system/**](https://www.geeksforgeeks.org/what-is-a-network-operating-system/)**)**
3. According to the book, symmetric multiprocessing is one of the most common multiprocessor systems. In a SMP, each peer CPU processor performs all tasks, including operating-system functions and user processes. **(Book – 16)**.

According to GeeksForGeeks, symmetric multiprocessing refers to the computer architecture where multiple identical processors are interconnected to a single shared main memory. **(Link:** [**https://www.geeksforgeeks.org/what-is-smp-symmetric-multi-processing/**](https://www.geeksforgeeks.org/what-is-smp-symmetric-multi-processing/)**)**.

Lastly, according to the book, an example of an OS that uses SMP is Windows. It says that Windows supports multiple operating environments and symmetric multiprocessing**. (Book – 55)**

1. According to GeeksForGeeks, asymmetric multiprocessing is a multiprocessor system where not all of the CPUs are treated equally. Only a master processor runs the tasks of the OS. This leads to a master-slave relationship. One processor serves as a master and the others as slaves to that master processor. **(Link:** [**https://www.geeksforgeeks.org/difference-between-asymmetric-and-symmetric-multiprocessing/**](https://www.geeksforgeeks.org/difference-between-asymmetric-and-symmetric-multiprocessing/)**)**.
2. Uniform Memory Access or UMA, according to the book, allows for access to all main memory by all the processors, without performance differences based on CPU or memory location. **(Book – G38 Glossary)**

I’m not sure if this supports uniform memory access or not, but according to the book definition and other information I have found, the new M1, M2, and M3 series chips from Apple follow a unified memory architecture system which is identical to the definition of UMA. Apple calls it Unified Memory Architecture, but it follows a UMA structure. **(Link:** [**https://developer.apple.com/videos/play/wwdc2020/10686/**](https://developer.apple.com/videos/play/wwdc2020/10686/)**)**

1. According to the book, cooperating processes are process that can affect or be affected by the other processes executing in the system. An OS provides cooperating processes to allow for information sharing, computation speedup, and modularity. **(Book: 123)** These are the three things the book lists. To add a fourth, tutorialspoint says that convenience is a reason for process cooperation on top of the three included in the book. **(Link:** [**https://www.tutorialspoint.com/cooperating-process**](https://www.tutorialspoint.com/cooperating-process)**)**
2. According to GeeksforGeeks, a time-sharing OS uses CPU scheduling and multi-programming to provide each user with a small portion of a shared PC at once. **(Link:** [**https://www.geeksforgeeks.org/time-sharing-operating-system/**](https://www.geeksforgeeks.org/time-sharing-operating-system/)**).** In the book, we can see also see that traditional time-sharing systems used a timer and scheduling algorithms to cycle processes rapidly through the CPU, giving each user a share of the resources. However, traditional time-sharing systems are rare today. **(Book: 41)**. An example of a time-sharing OS would be Windows according to javatpoint. **(Link:** [**https://www.javatpoint.com/time-sharing-vs-distributed-operating-system**](https://www.javatpoint.com/time-sharing-vs-distributed-operating-system)**)**
3. User View: The user view of an OS is designed mostly for ease of use. Things like the ability for I/O and a user interface to interact with. **(Book: 4-5)**

System View: The system view of an OS views the OS as a resource allocator with the computer. Things like CPU time, memory space, I/O device connectivity, etc. **(Book: 5)**

1. Secondary storage is needed in a computer to store large amounts of data that cannot be held on the primary storage. This is done through devices such as HDD (Hard Disk Drive) or SSD (Solid State Drives). Secondary storage works with the primary storage and holds programs for primary storage until the primary storage needs to access that data. So, the secondary storage holds large data, and also works with the primary storage to give the primary storage a database of sorts to access and pull from when needed. **(Book: 15)**
2. Windows has many forms of IPC, and in their official docs, they show things such as clipboard, COM, data copy, DDE, file mapping, mailslots, pipes, RPC, and Windows sockets. These things allow for the OS to work behind the scenes to allow for easier use of the system by the user. Things like the clipboard for example, when something is copied by one application, it is stored in the clipboard, this then allows for the ability to paste that data into another application through the clipboard. COM is another interesting one, it stands for Component Object Model which is the foundation of OLE. OLE allows for things such as, a word processor being able to embed a graph from a spreadsheet. There is a lot more data in the link, but I found these to be the most interesting to discuss and the easiest to connect to user scenarios. **(Link:** [**https://learn.microsoft.com/en-us/windows/win32/ipc/interprocess-communications**](https://learn.microsoft.com/en-us/windows/win32/ipc/interprocess-communications)**)**

**Part 2: Java Programming** *(40 points)***:**

A black screen with white text

Description automatically generated

**BufferImpl.java:**

**public class BufferImpl<E> implements Buffer<E>**

**{**

**private static final int BUFFER\_SIZE = 5;**

**private E[] elements;**

**private int in, out, count;**

**public BufferImpl() {**

**count = 0;**

**in = 0;**

**out = 0;**

**elements = (E[]) new Object[BUFFER\_SIZE];**

**}**

**// producers call this method**

**public void insert(E item) {**

**while (count == BUFFER\_SIZE)**

**; // do nothing -- no free space**

**// add an element to the buffer**

**elements[in] = item;**

**in = (in + 1) % BUFFER\_SIZE;**

**++count;**

**}**

**// consumers call this method**

**public E remove() {**

**E item;**

**while (count == 0)**

**; // do nothing - nothing to consume**

**// remove an item from the buffer**

**item = elements[out];**

**out = (out + 1) % BUFFER\_SIZE;**

**--count;**

**return item;**

**}**

**}**

**Buffer.java:**

**public interface Buffer <E>**

**{**

**public void insert(E item);**

**public E remove();**

**}**

import java.util.ArrayList;

public class BufferImpl

{

    private int bufferSize;

    private int in, out, count;

    //Array List for the buffer size

    //Setting the bufferSize as a variable allows for later entry of bufferSize at runtime

    ArrayList<Integer> elements = new ArrayList<Integer>(bufferSize);

    public BufferImpl() {

        count = 0;

        in = 0;

        out = 0;

        bufferSize = 0;

    }

    //Value-Pass Constructor. I took code from my Object-Oriented Programming course to set this up.

    public BufferImpl(int countVal, int inVal, int outVal, int bufferSizeVal)

    {

        count = countVal;

        in = inVal;

        out = outVal;

        bufferSize = bufferSizeVal;

    }

    // producers call this method

    public void insert(Integer item) {

        while (count == bufferSize)

            ; // do nothing -- no free space

        // add an element to the buffer

        elements.add(item);

        in = (in + 1) % bufferSize;

        ++count;

    }

    // consumers call this method

    public Integer remove() {

        Integer item = 0;

        while (count == 0)

            ; // do nothing - nothing to consume

        elements.remove(item);

        out = (out + 1) % bufferSize;

        --count;

        return item;

    }

}

/\*I'm not entirely sure if I got the right answer here or not. However, I changed as little as possible, while also keeping the integrity of the program intact.

I have a feeling my "insert and remove" functions are messed up now, but this is what I settled on. I tried to run it in a main function I threw together really quick, and I am able to enter values. However, I can't quite seem to tackle the functions. Somewhere along the line my modifications need feedback.

I do feel like I completed the ArrayList and the removal of the generics and interface properly though. I also added the value pass constructor.

Lastly, the instructions at the top of this exam say to show all code running and give screenshots of output. However, the instructions for the question said there was no main needed. So, I just modified the code like instructed.

Sources Used:

Oracle Docs for ArrayLists: https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Oracle Docs for Passing Values: https://docs.oracle.com/javase/tutorial/java/javaOO/arguments.html

Objected Oriented Programming in-class program: Book.java

\*/