|  |  |  |  |
| --- | --- | --- | --- |
| Instructor | ***Prof. Luke Papademas*** | Due Date |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part | **1** | **2** | **3** | **4** | Total |
| *Maximum Points* | **25** points | **25** points | **25** points | **25** points | **100**G101010 pointsG |
| ***Your Score*** |  |  |  |  |  |

**Textbook Reading Assignment**

Thoroughly read Chapter(s) 8 in your Computer Architecture and Organization textbook.

**Part 1 Glossary Terms - System Software**

Define, in detail, each of these glossary terms from the realm of computer architecture and computer topics, in general. If applicable, use examples to support your definitions. Consult your notes

or course textbook(s) as references or the Internet by visiting Web sites such as:

[**http://www.bing.com**](http://www.bing.com) or [**http://www.webopedia.com**](http://www.webopedia.com/)

**(a) 3 - Tiered Architecture**

|  |
| --- |
| 3-tiered architecture refers to the “well-established” pattern in software in which the presentation, application, and data layers are separated. All requests from the presentation layer for data must first pass through the application layer, and all responses from the data layer must also pass through the application layer. |

**(b) Parse Tree**

|  |
| --- |
| A parse (or syntax) tree is used by language compilers to analyze and order words from the token stream that particular to a programming language into a data structure that can be used to create a “pseudo-assembly” code that can then be used to run machine instructions. |

**(c) Resident Monitor**

|  |
| --- |
| A “resident monitor” was the precursor to modern Operating Systems, and was used in early computers to control computer hardware and manage system resources. |

**(d) Tightly Coupled Multiprocessors**

|  |
| --- |
| Tightly coupled multiprocessors share a single centralized memory which requires an OS to synchronize processes carefully to ensure protection. This is typically used in systems with 16 or fewer processors (Null et al. 487). |

**(e) Transaction Processing Monitor**

|  |
| --- |
| A TPM is a middleware that is used for monitoring transactions between system components to help in system resource management. |

**Part 2 Exercises - System Software**

For each of the following, enter True or False.

**TRUE (1)** Spooling is the simplest form of multiprogramming.

**FALSE (2)** A virtual machine is the real hardware of the real computer that is controlling the program.

**TRUE (3)** Absolute code is non - executable binary code that must always be loaded at a particular location in memory.

**TRUE (4)** Interpreted languages also have a many - to - many relationship between the source code statements and executable machine instructions.

**TRUE (5)** In Java, deallocation of heap memory is referred to as garbage collection, which is done by the JVM automatically.

**TRUE (6)** To speed up the performance of the java software, Java’s Just - In - Time ( JIT ) compiler is used.

**TRUE (7)** The goal of database management systems is to provide timely and easy access to a large volume of data efficiently.

**TRUE (8)** Real - time systems are used for process control in manufacturing plants, assembly lines, robotics and complex physical systems.

**FALSE (9)** Java programs are stored in an intermediate code called bytecode.

**FALSE (10)** A timesharing system does not permit the systems to be accessed by multiple concurrent users.

**Part 3 Exercises - System Software**

**(1)** **( Database Processing )**

Answer the following with respect to database processing:

(a) What is a race condition? Give an example.

(b) How can race conditions be prevented?

(c) What are the risks in race condition prevention?

|  |
| --- |
| 1. **A race condition is a situation where one process begins before another has resolved on a shared resource, leading to conflicts in resource allocation or an inaccurate representation of state. An example I’m very recently familiar with involves a message queue service (AWS SQS) within a microservice architecture. I was working on some parallel processing application logic to fetch and process messages from the message queue in the cloud, and had to use a “visibility timeout” mechanism to prevent other processes from pulling messages that were currently being processed. I ran into a race condition when I underestimated how long a process would take to finish, and so a message was pulled twice from the queue, therefore I had two processes running the same job. One process finished, removed the message from the queue, and then the other process finished, and attempted to remove the message from the queue, however the message no longer existed (an inaccurate state), which threw an exception in my application.** 2. **A race condition can be prevented by blocking other processes from resources currently being used until the current process resolves.** 3. **Preventing race conditions can lead to idle processes, unutilized resources, etc., while we wait for availability. Using various methods (e.g., polling, interrupts) can help optimize performance while preventing race conditions. These concepts at a lower level apply to higher level languages, e.g, await/async and promises in JavaScript.** |

**(2)** **( Deadlocks )**

We said that the risk of deadlock is always present anytime a system resource is locked.

Describe how a deadlock can occur such as in the situation given below.

Suppose T1 has X locked, and T2 has Y locked.

Now suppose T1 needs Y and T2 needs X .

|  |
| --- |
| **Because T1 has X locked and T2 has Y locked, if T1 needs Y and T2 needs X, neither process can resolve, leading to a deadlock.** |

**(3)** **( Operating Systems )**

What do you feel are the limitations of a computer that has no operating system? How

would a user load and execute a program?

|  |
| --- |
| **The limitations include inaccessibility to most users and ample opportunity for all users to shoot themselves in the foot due to poor resource management. I’m pretty sure a user could write a program that would set the computer on fire if there were no operating system to provide an interface between the users software and hardware.**  **To load and execute a program without the use of, a user would need to use a special tool to edit and load their program into the PROM before executing their program.** |

**(4)** **( Compilers )**

Match the analyzer relative to compilers:

**B** Semantic analyzer (a) the phase of a compiler that would give you a syntax error

**C** Lexical analyzer (b) the phase that complains about undefined variables

**A** Syntax analysis (c) the compiler phase that would emit an error message if you try to add an integer to a character string

**(5)** **( Java Virtual Machine )**

Why is the execution environment of a Java class called a virtual machine? How does this virtual machine compare to a real machine running code written in C ?

|  |
| --- |
| **The execution environment of a Java class is called a virtual machine (the JVM) because it manages its own subset of allocated memory and ISA like a real machine. The JVM compiles Java classes down to bytecode which is then converted to native machine code.  This virtual machine (which is a program) runs on the real machine and is allocated a fixed set of memory and resources from the real machine.** |

**Part 4 Exercises - System Software**

Write a complete answer for each of these.

**(1) ( Mobile Operating Systems )**

What do you feel are the limitations of a mobile telephone that has no operating system? How would a user load and execute an application program?

|  |
| --- |
| **The limitations on a mobile phone with no operating system are the same as any system; low accessibility, with plenty of room to wreck your system. I assume they would need to use a PROM editor to load and run their program.** |

**(2) ( Multiprogramming, Multiprocessing and Multithreading )**

What is the difference between multiprogramming and multiprocessing?

Multiprogramming and multithreading?

|  |
| --- |
| **The difference between multiprogramming and multiprocessing is that in systems that have multiple processors, multiple processes can be executed concurrently, while in multiprogramming, is the concurrent execution of more than one program in main memory.** |

**(3) ( Subsystems )**

Under what circumstances is it desirable to collect groups of processes and programs into

subsystems running on a large computer? What advantages would there be to creating

logical partitions on this system?

|  |
| --- |
| **Partitioning a system helps with resource and task management that can lead to optimal performance.** |

**(4) ( Dynamic Linking )**

Discuss the advantages and disadvantages of dynamic linking.

|  |
| --- |
| **Dynamic linking enables an operating system to load external routines that are needed when a program is loaded. A DLL (dynamic link library) file for a program is read by an OS, which then fetches the dependencies for the program before executing it.   The main disadvantage of dynamic linking is that if the DLL file is not present at program load time, or the expected dependencies do not exist in the system, the program will fail.** |

**(5) ( Assembly Language )**

Why should assembly language be avoided for general application development? Under

what circumstances is assembly language preferred or required?

|  |
| --- |
| **Assembly language is difficult to read and write, and even more difficult to troubleshoot or collaborate on. Assembly code is preferred in systems with very limited resources, and required in systems with no operating systems / higher language compilers (e.g., embedded systems), in which it is necessary to interact directly with a machine’s ISA.** |