Student Name:..……………………KEY…………………………………. UIN:………………………………..

**Student Score / 55**

True/False Questions [20 pts] **I highlighted only the false answers**

1. The executable image of a program must be loaded into the main memory first before executing
2. An Operating System (OS) does not trust application programs because they can be either buggy or malicious
3. There was no concept of OS in first generation computers
4. The PC register of a CPU points to the next instruction to execute in the main memory
5. Second generation computers still executed programs in a sequential/batch manner
6. Time sharing computers gave a fixed time quantum to each program
7. An OS resides in-between the hardware and application programs
8. The primary goal of OS is to make application programming convenient
9. Context switching does not contribute much to the OS overhead
10. Main Memory access is slower than register/cache access because it is physically outside the CPU
11. Multiprogramming cannot work without Direct Memory Access (DMA) mechanism
12. Interrupts are necessary for asynchronous event handling in a CPU
13. A program can be kicked out of a CPU when it requests I/O operation, or when another Interrupt occurs
14. A program error can kick a program out of CPU
15. Interrupts are necessary to bring a program back to CPU if it was previously kicked out
16. The “Illusionist” role of the CPU allows a programmer write programs that are agnostic of other programs running in the system
17. Modern operating systems come with many utility services that are analogous to the “Glue” role of the OS
18. Networking service is not a core OS part, rather a common service included with most OS
19. Resource allocation and Isolation are not part of the core OS, rather common services included with OS
20. Efficiency is the secondary goal of an OS

Short Questions

1. [5 pts] Define multiprogramming. How is this better than sequential program execution?

*It is a technique to keep the CPU utilized when the current program requests I/O operation. For this method to work, multiple programs must stay loaded in the memory so that they can brought in quickly. Another important factor to avoid CPU involvement in every I/O step is the DMA controller, which takes care of the back-and-forth with the I/O device for sending commands.*

*It is better than sequential execution because the CPU and I/O device utilization is higher in this model resulting in higher throughput (i.e., more things getting done in the same time period).*

***Note to grader:*** *3 factors have equal weight here: 1) Overlapping CPU and I/O operations, 2) The memory must be shared between the monitor and several programs. If the programs are not preloaded multiprogramming cannot be efficient, 3) DMA must be used to minimize/remove I/O interference with the CPU. Assign equal points (3.33) for each of these criteria*

1. [10 pts] Define time-sharing. Can you combine time-sharing with multiprogramming?

***Definition (5 pts):*** *Every program is allowed a maximum time called time quantum before being kicked out. This lets the CPU juggle many programs and prevents one program from monopolizing the CPU. The hardware timer is needed that is reset after each quantum. The CPU time shared by n-programs resulting in n-fold slowdown for each. However, most programs do not just do I/O operations, making the situation much better. In addition, human’s ability to notice this slowdown is also limited.*

***Note to grader:*** *students must mention a) time quantum, b) hardware timer, c) juggling ability*

***Can you ..? (5 pts):*** *Yes. A program will then be then be kicked out either by I/O even or because of the hardware timer fired, whichever happens first. This is how all modern OS’s are.*

1. [10 pts] Say you are running a program along with many other programs in a modern computer. For some reason, your program runs into a deadlock and never comes out of that. How does the OS deal with such deadlock? How about infinite loop? How does the OS detect, if at all, such cases?

*The OS does not need to do anything other than running time sharing. The deadlocked program periodically comes back to the CPU and spends it share of time. However, the timer kicks it out and brings other programs back in.*

1. [20 pts] In a single CPU single core system, schedule the following jobs to take the full advantage of multiprogramming. The following table shows how the jobs would look like if they ran in isolation.

|  |  |  |  |
| --- | --- | --- | --- |
|  | JOB1 | JOB2 | JOB3 |
| Type of job | Full CPU | Only I/O | Only I/O |
| Duration | 5 min | 15 min | 10 min |
| Memory required | 50MB | 100MB | 75MB |
| Needs disk? | No | No | Yes |
| Needs terminal? | No | Yes | No |

1. What is the total time for completion for all jobs in sequential and multi-programmed model?
2. Fill out the multiprogramming column in the following table (i.e., when the jobs are scheduled in multiprogramming). Assume that the system’s physical memory is 256MB.

|  |  |  |
| --- | --- | --- |
| Average Resource Use | Sequential | Multiprogramming |
| Processor | 5/30 = 16.67% |  |
| Memory | 32.55% |  |
| Disk | 33.33% |  |
| Terminal | 50% |  |

Memory usage is computed as follows: (5minx50MB + 15minx100MB + 10minx75MB) / (30minx256MB) = 32.55%

Other resources are fully utilized during the time they are utilized. So, you compute utilization only based on the duration they are used.

**Answer:**

*The following is the schedule of the above 3 jobs in a multi-programmed system:*

*So, the completion time is 15 mins as follows:*

|  |  |  |  |
| --- | --- | --- | --- |
| *Time (min) 🡪* | *0 5* | *6 10* | *11 15* |
| *CPU* | *Job1* |  |  |
| *Memory(total 256MB)* | *225 MB* | *175 MB* | *100 MB* |
| *Terminal* | *Job2* | | |
| *Disk* | *Job3* | |  |

*The following is utilization of different resources:*

|  |  |  |
| --- | --- | --- |
| *Average Resource Use* | *Sequential* | *Multiprogramming* |
| *Processor* | *5/30 = 16.67%* | *5/15 = 33.33%* |
| *Memory* | *32.55%* | *65.10% \** |
| *Disk* | *33.33%* | *66.67%* |
| *Terminal* | *50%* | *100%* |

*\*Memory Utilization = (5minx225MB + 5minx175MB + 5minx100MB) / (15minx256MB) = 65.10%*

*Alternate Answer:*

*If you assume that since the CPU is fully utilized for the first 5 mins and you cannot even start Job2 or Job3 within that time period, you would be correct as well. The completion time is 20 min in that case where the schedule is as follows:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Time (min) 🡪* | *0 5* | *6 10* | *11 15* | *16 20* |
| *CPU* | *Job1* |  |  |  |
| *Memory* | *50 MB* | *175 MB* | | *100 MB* |
| *Terminal* |  | *Job2* | | |
| *Disk* |  | *Job3* | |  |

*Using the above, the utilizations now look like the following:*

|  |  |  |
| --- | --- | --- |
| *Average Resource Use* | *Sequential* | *Multiprogramming* |
| *Processor* | *5/30 = 16.67%* | *5/20 = 25%* |
| *Memory* | *32.55%* | *48.82% +* |
| *Disk* | *33.33%* | *50%* |
| *Terminal* | *50%* | *75%* |

*+Memory Utilization = (5minx50MB + 10minx175MB + 5minx100MB) / (20minx256MB) = 48.82%*

1. [25 pts] Consider the following program and provide explanations where asked in the code comment after running the program in your system. Note that there are 8 places where explanation is needed. For explanation 1 (i.e., the first commented line), does the Header size equal the sum of individual data types (i.e., a char, an int and a pointer)? Try to explain this with something called “packing”.

***Answer:***

*The size of a class is not always summation of the fields’s sizes because of something called “padding”/”packing” in most Oss for efficiency. The idea is that each field, even smaller, is promoted to a default (either 4 or 8) when part of a class. For that reason, you will see 1) and 2) printing wither 16 or 24 depending on which system. Note that only the char\*’s size (i.e.,, 8 bytes) is added to the computation. The payload size is not added because that is allocated separately in a different place in the heap.*

*4) h3 is just a pointer, so its size is 8 bytes. The object size is irrelevant for pointer size – all pointers are 8 bytes in a 64-bit system*

*5-rest) see the attached code q2*

*Note to grader:*

*Points distribution Parts 1-4: 5 pts, Part 5, 6, 7 and 8 are each 5 pts.*