CSCE 313
Quiz 1
Jeffrey Xu
08/27/20

1. The executable image of a program must be loaded into the main memory first before executing

True. The PC register points to the next instruction in main memory and if the program isn't loaded into the main memory then the CPU doesn't know what instruction to run next.

2. An Operating System (OS) does not trust application programs because they can be either buggy or malicious

True. If the OS gave application programs access to priviledged instructions, then the programs could go and change the OS code and consequently crash the OS and lose data.

3. There was no concept of OS in first generation computers

True. The first generation of computers used card readers and programmers had to manually code and compile their programs.

4. The PC register of a CPU points to the next instruction to execute in the main memory

True. The PC register determines the program flow that the CPU uses. Without the PC, the CPU wouldn't know what instruction to execute.

5. Second generation computers still executed programs in a sequential/batch manner

True. The second generation computers automated the process of loading, translating and executing instructions and allowed for multiple programs to be run but it still performed these tasks in a sequential manner.

6. Time sharing computers gave a fixed time quantum to each program

True. Time sharing forced the CPU to rotate between programs (programs kicked out whenever the time quantum is up).

7. An OS resides in-between the hardware and application programs

True. The OS manages resources for the application layer and provides abstraction of hardware resources.

8. The primary goal of OS is to make application programming convenient

True. The goal of the OS is to provide virtualization to the programmer and program. This allows for each program to think it has all of the CPU resources and allows the program to not worry about physical memory management.

9. Context switching does not contribute much to the OS overhead

10. Multiprogramming cannot work without Direct Memory Access (DMA) mechanism

True. If multiprogramming didn't have DMA, then it couldn't load and kick out programs whenever a program required IO or CPU time.

11. Interrupts are necessary for asynchronous event handling in a CPU

True. The way asynchronous pipelining works is by kicking out programs and loading them back in. Without interrupts, we wouldn't be able to kick out programs from the CPU.

12. A program can be kicked out of a CPU when it requests I/O operation, or when another Interrupt occurs

True. In multiprogramming, whenever an interrupt occurs for a program, it can kick that program out of the CPU and handle the IO or another interrupt while another program uses the CPU.

13. A program error can kick a program out of CPU

True. If a program error causes a program to crash, then the program can be kicked out of the CPU and another program can use the CPU.

14. Interrupts are necessary to bring a program back to CPU if it was previously kicked out

True. Interrupts allow for programs to be kicked out of the CPU. If a program was previously kicked out and another program takes its place, then if the previously kicked out program needs the CPU, an interrupt must be sent to the program that is currently using the CPU.

15. The "Illusionist" role of the CPU allows a programmer write programs that are agnostic of other programs running in the system

True. Each program has its own program frame allowing it to run in the illusion that it has full access to the CPU.

16. Modern operating systems come with many utility services that are analogous to the "Glue" role of the OS

True. The OS connects the application layer and the hardware layer.

17. Networking service is not a core OS part, rather a common service included with most OS

True. The OS really only concerns itself with resource management and memory allocation, not network services but most OS do provide functionality for networks.

18. Resource allocation and Isolation are not part of the core OS, rather common services included with OS

False. The main purpose of the OS is to manage allocation and isolation of resources.

19. Efficiency is the secondary goal of an OS

False. Efficiency is the main goal of an OS. If an OS was slow, it would not be able to compete with other OS making it useless.

20. After handling a fault successfully, the CPU goes (when it does go back) to the instruction immediately after the faulting one

21. Interrupts are asynchronous events

22. Memory limit protection (within a private address space using base and bound) is implemented in the hardware instead of software

23. Memory limit protection checks are only performed in the User mode

24. Divide by 0 is an example of a fault

25. Multiprogramming can be effective even with one single-core CPU

26. [10 pts] Define multiprogramming. How is this better than sequential program execution?

Multiprogramming allows for asynchronous pipelining. This allows programs to be run out of order instead of the order than they come in (sequentially). Multiprogramming is much better than sequential program execution because it makes the CPU as busy as possible resulting in a higher throughput.

27. [10 pts] Define time-sharing. Can you combine time-sharing with multiprogramming?