CSCE 313 QUIZ 1 Spring 2021

Student Name:..... Chaoyang Zhu ..... UIN:...930001840.....

Student Score / 100

True/False Questions **[25 pts]**

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. Multiprogramming cannot work without Direct Memory Access (DMA) mechanism
11. Interrupts are necessary for asynchronous event handling in a CPU
12. A program can be kicked out of a CPU when it requests I/O operation, or when another Interrupt occurs
13. A program error can kick a program out of CPU
14. Interrupts are necessary to bring a program back to CPU if it was previously kicked out
15. The "Illusionist" role of the OS allows a programmer write programs that are agnostic of other programs running in the system
16. Modern operating systems come with many utility services that are analogous to the "Glue" role of the OS
17. Networking service is not a core OS part, rather a common service included with most OS
18. Resource allocation and Isolation are not part of the core OS, rather common services included with OS
19. Efficiency is the secondary goal of an OS
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

Answer:

1	2	3	4	5	6	7	8	9	10
T	T	T	T	T	T	T	T	F	T
11	12	13	14	15	16	17	18	19	20
T	T	T	T	T	T	T	F	T	T

CSCE 313 QUIZ 1 Spring 2021

21	22	23	24	25
T	T	T	F	T

Short Questions

26. **[5 pts]** Define multiprogramming. How is this better than sequential program execution?

Answer:

It is a technique to ensure CPU and I/O can work at the same time. To realize that goal, we need to load multiple programs in the memory and have the DMA controller to drive I/O operations without involving CPU.

It is better than sequential program execution since the utilization of CPU is higher.

27. **[5 pts]** Define time-sharing. Can you combine time-sharing with multiprogramming?

Answer:

Each program is allocated a time slot to use the CPU, which is called time quantum. When time is up, the program will be kicked out and another program will get loaded. The program will resume when it gets turn to use the CPU again.

28. **[5 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 loops? How does the OS detect, if at all, such cases?

Answer:

Faced with deadlock and infinite loops, what the OS does is just running the time sharing. The deadlocked program will come back to the CPU periodically. When the time slot allocated to the program is up, the program will be kicked out by the timer. Then other program will be loaded into the CPU.

29. **[5 pts]** Which of the following are privileged operations allowed only in Kernel mode?

a) Modifying the page table entries
b) Disabling and Enabling Interrupts
c) Using the "trap" instruction
d) Handling an Interrupt
e) Clearing the Interrupt Flag

Answer:

a, b, d, e

30. **[25 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. [Use the attached pages from W. Stallings book to solve this problem]

	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

a. What is the total time of completion for all jobs in a sequential and multi-programmed model?
b. 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%	5/15=33.33%
Memory	32.55%	65.10%
Disk	33.33%	66.67%
Terminal	50%	100%

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:

Timeline	0----------->5	6----------->10	11----------->15
CPU	Job1		
Memory	225MB	175MB	100MB
Disk	Job3		
Terminal	Job2		

Processor:
5/15=33.33%
Memory:
0->5: 50+100+75 = 225MB
6->10: 100+75 = 175MB
11->15: 100MB
(225*5+175*5+100*5)/256*15=65.10%
Disk:
10/15=66.67%
Terminal:
15/15=100%

31. **[15 pts]** The following are steps in a "sequential" Interrupt handling. These steps are in the user-to-kernel direction, while the steps in the opposite direction are simply reversed.

Hardware does the following:

---

1. Mask further interrupts
2. Change mode to Kernel
3. Copy PC, SP, EFLAGS to the Kernel Interrupt Stack (KIS)
4. Change SP: to the KIS (above the stored PC, SP, EFLAGS)
5. Change PC: Invoke the interrupt handler

Software (i.e., the handler code) does the following:

---

1. Stores the rest of the general-purpose registers being used by the interrupted process

2. Performs the rest of the interrupt handling operation

Now, answer the following questions:

(a) [7 pts] What changes would you make in the steps below so that "nested" Interrupts can be handled?

(b) [3 pts] For sequential interrupt handling, can you interchange steps 2 and 3? Explain.

(c) [2 pts] For sequential interrupt handling, can we interchange step 1 and 2? Explain.

(d) [3 pts] For sequential interrupt handling, can we interchange step 1 and 3? Explain.

Answer:

(a) Before we enter the software handling period, we should reenable Interrupts. So that we can handle other "nested" Interrupts.

(b) Yes, because copying the value of registers into the KIS does not have relationship with the mode.

(c) Yes, we can interchange step1 and step2. Masking further interrupts does not affect changing mode and changing mode does not affect masking further interrupts.

(d) No, because if we do not mask further interrupt, other interrupts will interfere with the current interrupt state. Part of the value in this state may be overwritten.

32. [15 pts] Assuming each page is 4KB, how many page faults or faults of any kind the following program will generate? Explain your answer.

```
int size = 4 * 1024 * 1024; // size = 4 megabytes
char * memory = new char [size]; // allocate 4 mega bytes
__int64_t * a = (__int64_t *) memory;//__int64_t is 64-bit or 8-byte integer
for (int i=0; i< (100 * 1024) ; i++)
      a [i] = 0;
```

Answer:

The memory allocated is used to store char type data, while the for loop stores the int type data into this memory.

The for loop will use the memory of 100*1024*8 Bytes.

Since each page is 4K Bytes.

So, the number of fault pages is 100*1024*8/(4*1024) = 200.