Stephen Chambers
Professor Varki
CS 620
September 11, 2013

Assignment 1

1. (3) At any given time, the CPU executes a single process, even in a multiprogramming environment. In the light of this, how does multiprogramming improve CPU utilization?
A single program in memory can't keep either the CPU or the I/O devices busy at all times. If it was allowed to do this, the CPU's immense processing power would be wasted. Additionally, most users frequently have multiple programs running. In order to utilize the CPU in the most efficient way possible, multiprogramming organizes jobs, which can be defined as code or data, so that the CPU always has one to execute.

2. (3) Differentiate between multiprocessor systems and multiprogramming systems.
A multiprocessor system has multiple CPU's in close relation to each other. This allows the computers to share hardware resources on the device and help each other in the execution of jobs on the machine. In contrast, multiprogramming is the process of organizing code and data so that the CPU always has something to execute. In essence, a multiprocessor system does not necessarily guarantee that the CPUs always have something to execute, however, the goal is certainly to have multiprogramming in a multiprocessor environment. Both multiprogramming and multiprocessing are not independent of one another. A system can be both of them, only either of them, or neither of the two.

3. (2) Why are disk I/O operations DMA driven, not interrupt driven I/O.
Traditionally, all disk operations were interrupt driven I/O. This caused the CPU to be interrupted after every character was read or written. This was terribly inefficient, and the concept of a DMA (Direct Memory Access) controller was developed. The DMA allows data to be read and written in blocks, interrupting the CPU at the end. This drastically shortens the amount of time that I/O takes, and it frees the CPU to execute other processes without being constantly interrupted.

4. (3) In virtually all computers, DMA access to main memory is given higher priority than processor access to main memory. Why?
The DMA controller has a higher priority than the CPU when accessing main memory because the DMA is used less often. If the CPU had priority, the DMA would not be able to access main memory efficiently, if at all, because the CPU would constantly intervene. Additionally, the DMA operates by writing and reading blocks of data. If the CPU constantly interrupted the DMA in the middle of the transfer of data to memory, it may result in memory loss and would defeat the purpose of the DMA controller itself.

5. (6) A main memory system consists of a number of memory modules attached to the system bus. When a write request is made, the bus is occupied for 50 nanoseconds (ns) by the data, address, and control signals. During the same 50 ns, and for 200 ns thereafter, the addressed memory module executes one cycle accepting and storing the data. The operation of the memory modules may overlap, but only one request can be on the bus at any time. Assume that there are eight such modules connected to the bus. What is the maximum possible rate (in requests per second) at which data can be stored? Explain you answer.

It only takes 50 nanoseconds when a write request is made through the bus, and 200 nanoseconds when the memory module accepts and stores the data. Therefore, after five write requests, the first memory module is done storing the data and is ready to process again. Since this problem has eight memory modules, the amount of time it takes for the memory modules to process and store data is insignificant, as a memory module will always be available when a write request is made. Therefore, the maximum possible rate in requests per second is as follows:

Write Request Time: 50ns
X: Number of Write Requests

50(X)ns = 1 second
50(X)ns = 1,000,000,000 nanoseconds
X = 1,000,000,000ns /50ns
X = 20,000,000

The maximum possible rate at which data can be stored is 20,000,000 write requests per second.

6. (1) Differentiate between a trap and an interrupt.

An interrupt is performed by a hardware controller; letting the CPU that something else needs its attention. On the contrary, a trap is a *software* interrupt, letting the user know that something was done that is not possible to execute (trying to read/write outside of the program domain, dividing by 0, etc.).

7. (3) What are the actions taken by the CPU when a trap/interrupt occurs.

When an interrupt occurs, the device controller sets its corresponding bit in the interrupt register. During the fetch execute cycle, the CPU checks the interrupt register. If a bit is set, the CPU calls the Operating System and transfers execution by setting the Program Counter to a fixed location in memory, which is usually the address of the service routine where the interrupt is located. Upon the service routine's completion, the CPU resumes the interrupted process.

8. (3) Why is an interrupt given higher priority than currently executing program?
If interrupts had a lower priority than the currently executing program, they would never be serviced. It is in the benefit of efficiency to have a CPU handle an interrupt, as other processors or resources would be able to continue executing the program that is no longer interrupted. Interrupts are the driving force behind an Operating System; it is the reason that multiple programs can be executed in a timely, efficient manner. Without them, an I/O instruction would kill the processing time of a particular program, as the CPU would always have other "lower-priority" things to execute with its resources. Additionally, interrupts are the force behind timers, another integral part of Operating Systems. If a program fell into an infinite loop, a timer would fire and send an interrupt to the CPU, allowing it to handle the faulty program and then move on to other tasks. If this interrupt had a lower priority, there would be no way to stop the program with the infinite loop from executing.

9. (3) What is a real-time system?
A real-time system is a system in which the response time is critical. There are very strict time requirements set on the processors of these systems, and these processors are often dedicated to a single task. There are two types of real-time systems, hard real-time systems and soft real-time systems. Hard real-time systems have even stricter time requirements set on the processors, as lives or other immensely valuable things may be at stake. Some examples of hard real-time systems are hospital machines and air traffic controllers. Soft real-time systems do have stricter time requirements than other processes, but the response time is not as critical in comparison to hard real-time systems. An example of a soft real-time system would be a multimedia application, such as a movie. The movie should be given a higher priority response time than other processes, as the user's entertainment depends on a non-interrupted environment. However, the user's life is not at stake, so the response time is not as critical as a hard real-time system.

10. (3) Why do real-time systems use program controlled I/O and not interrupt driven?
In a program controlled I/O system, the CPU does not execute other processes when an I/O operation occurs. It waits until that I/O interrupt service routine is completed, and then immediately returns to executing the program. A real-time system uses this type of I/O because its response time is more critical than other processes. While this does not fully utilize the CPU efficiently, it allows for a faster response time for that particular program, which is crucial to real-time system applications.

11. (6) A DMA module is transferring characters to main memory from an external device transmitting at 1000 bits per second (bps). The processor can fetch instructions at the rate of 1 million instructions per second ($10^6 * 8$ bps). By what percentage would the processor be slowed down due to DMA activity.
Without interruption, the CPU can process instructions at 8,000,000 bits per second. When the DMA is transferring characters to main memory, the CPU cannot fetch those instructions. Therefore, the CPU would be slowed down by the CPUs maximum processing speed subtracting out the time the DMA takes to transfer characters to main memory as a percentage of the CPUs maximum processing speed. The math is shown below:

CPUs maximum processing speed = 8,000,000 bits/second
DMA transfer time: 1,000 bits/second
CPUs maximum processing speed less DMA transfer time = 8,000,000 – 1,000 = 7,999,000

Percentage slowdown = 100% - 100 * (CPUs maximum processing speed less DMA transfer time / CPUs maximum processing speed)

Percentage slowdown = 100% - 99.9875%
Percentage slowdown = 0.0125%

Therefore, the DMA transferring characters to main memory does not present a significant impact to CPU processing speed.

12. (2) What is the storage hierarchy model?
The storage hierarchy model contains many different ways of storing data into memory. At the top of the hierarchy, the memory storage has the most speed, is the most expensive, and has the least amount of space. On the contrary, at the bottom of the hierarchy, the memory storage has the least speed, is the least expensive, and has the most amount of space. The Storage Hierarchy model from top to bottom is as follows: Registers, Caches, Main Memory, Disk, and Cloud.

13. (3) Give one disadvantage of the storage hierarchy model.
One disadvantage of the storage hierarchy model is that memory needs to be moved up the hierarchy constantly. This is very complicated to handle, as the memory needs to be preserved as it moves up to more efficient hardware. There also is a sense of predictability that Operating System has to have with memory, because it needs to somehow know what memory will need to be accessed at different points in a programs execution. Needless to say, the storage hierarchy model adds a layer of complexity to the already complex job of the Operating System.

14. (3) Give one advantage of the storage hierarchy model.
One advantage of the storage hierarchy model is that there is almost an unlimited amount of space to store data at a very cheap price. Although the performance is slower, the introduction of cloud storage has given users a vast amount of data storage space at an extremely cheap price.

15. (5) A CPU always accesses data from the cache, never from memory. Suppose cache access takes 25 ns. If data are not found in the cache, then the data have to be loaded from main memory into cache taking 250 ns. The CPU then accesses the data from cache. Thus, a cache "hit" takes 25 ns and a cache "miss" takes 275 ns. Suppose 70% of a program hits in the cache. How long does it take the CPU to access the program.
Number of Memory Accesses: x
Hit Execution Time: 25ns * 0.70x
Miss Execution Time: 275ns * 0.30x
Total Execution Time: T

Number of Memory Accesses: Hit Execution Time + Total Execution Time

$$\text{Time}(T) = (25\text{ns} * 0.70x) + (275\text{ns} * 0.30x)$$
$$T = 17.5x + 82.5x$$
$$T = 100x$$

On average, the CPU will take 100ns for every memory access made the program. Therefore, the program will take 100x nanoseconds to execute, where x is the number of memory accesses by the program, assuming that 70% of the program hits in the cache.

16. (3) Differentiate between a parallel and a distributed operating system.
A distributed system is one in which the processors do not share resources between them. Each processor in this system has its own local resources, and it's commonly referred to as a loosely coupled system. On the contrary, a parallel system has resources that are shared between all processors. It is commonly referred to as a tightly coupled system. Based on these distinctions, the two systems can vary in many different ways. For example, a distributed system can have its processors placed far apart, while a parallel system would have to have its processers be placed close to each other because they share a common memory.

17. (3) List 2 differences between mobile OS and PC OS.
One difference between mobile OS and PC OS is that the mobile OS is much less complex and much more compact than a traditional PC operating system. A mobile device does not have as many parts as a PC, therefore the mobile OS does not have to handle many different parts of the PC OS. This leads into a second difference between the two Operating Systems; the PC OS has much more processing power than the mobile OS. These OS "deficiencies" in comparison to PC OS are not actually deficiencies, but are by design. A consumer wants their handheld devices to be small and compact to be able to be taken on the run, not large and obtrusive like a desktop tower.

18. (4) What is cache coherency? Why is this problem typically handled by HW and not by the OS?

In a multiprocessor system, each CPU usually maintains its own resources, such as internal registers. The CPUs also maintain their own local cache, and this creates the problem of cache coherency. A copy of data might exist at the same time throughout the many different caches. If that data is updated, the copies in the other caches must also be updated immediately. This is usually handled by the hardware because of two reasons, the first of which being that hardware is generally much faster than the OS itself. Imagine if the OS had to check whether memory content was being accurately maintained throughout the caches every time the CPU executed an instruction. This would immensely slow down the programs performance. Additionally, there is a higher likelihood that bugs could creep into the Operating System, corrupting the data that is trying to be preserved. Values such as the memory bounds of a program are much less likely to be affected if placed in the hardware of the computer, such as a register.

19. (3) What is a superuser? Why is it recommended that your user account be an ordinary account and not a superuser account?

A superuser is a type of account used for administering and maintaining a system. It can have many different names, for example, a Linux superuser is known as "root". Superusers have much more extensive privileges than a regular account, and can thus be very dangerous. Much of the restrictions on a regular account are put there for protection against ignorance; someone would not want to purposely harm their personal computer. Additionally, downloading software from the internet is also very dangerous in superuser mode. It is very common to get viruses when this happens, as running as a superuser is nearly equivalent to running as the Operating System. It is imperative that someone has extensive knowledge of what they are about to do when running a command as a superuser.

20. (4) What are virtual machines? Explain the relationship between the Java programming language and the virtual machine concept.

A virtual machine is the implementation of a physical machine in software. This machine can execute instructions like a physical machine. In Java, there is a specific VM called the Java Virtual Machine (JVM). A Java program is compiled into bytecode and this virtual machine provides an execution environment in which to interpret and execute that bytecode. The JVM is especially crucial because it allows Java to be compiled and run on any operating system. It is not the native OS which handles the code, but the JVM.

21. (2) Differentiate between device controllers and device drivers.

Every device connected to a computer has a controller. A controller is a piece of hardware that serves as that connected device's CPU. The controller has its own unique commands, or language. In order for a CPU to "talk" to the controllers, it uses a piece of software called a driver. The driver serves as an intermediary between the CPU and the controller, and allows for communication to occur between the two.

2. Navigating Linux:

1. (23) Submit linux.txt as per instructions.

2. (3) Some of the commands involve the "|" symbol representing pipes. What are pipes in Linux?

The pipe in Linux allows separate processes to communicate with one another when they were not necessarily designed to do so. The pipe allows for these processes to be combined in many different ways. When the pipe is used in a command, it redirects the output of one program into the input of another. For example, ls | grep cs would take the output of the ls command and use grep to search for the occurrences of the string "cs".

3. (3) What is the purpose of redirection symbols ">" and "<" in Linux.

These symbols allow for input and output to be redirected in various ways. For example, a C program could be given an input file to be treated as stdin and stdout (i.e. ./a.out < file.txt). An output of a program can also be redirected into a file (i.e. ./a.out < file.txt > outputfile.txt). These redirection symbols allow for ease of I/O, as a program may not necessarily have to wait for information to be inputted on a keyboard or a similarly slow I/O device. Additionally, these redirection symbols do not work on just files. They can be used to redirect input and output between programs and other processes.

4. (3) One of the commands uses the background symbol "&". How do background processes differ from foreground processes?

Background processes can be run in the "background" of a shell. In other words, once a background process is run, commands can still be entered into the shell and other programs can be started. This means that there can be multiple background processes running, but only one foreground process. The major difference between the two is that the shell does a wait() when executing a foreground command and does not when executing a background command. Many daemons or other processes that sit and wait for input are usually executed in the background. There is no fundamental difference with the way that the processor executes a foreground process in comparison to a background process.