CS 620: Homework 1

Carmen St. Jean

September 4, 2012

1. *The bootstrap code sets the dual mode bit to OS-mode before handing control to the OS. The OS sets the dual mode bit to user-mode before handing control to a user program.*

   (a) *(1) What is the function of the dual mode bit?*

   The function of the dual mode bit is to distinguish between user mode and operating system mode for an executing program. The bit indicates the mode of program is either user (0) or operating system (1). The dual mode ensures that a problematic executing program does not interfere with other executing programs or the operating system itself.

   (b) *(3) When does the dual mode bit change from user-mode back to OS-mode? Who (or what) changes the mode bit from user mode to OS mode? Explain.*

   When an executing program attempts to do an illegal instruction (e.g., divide by zero) or to go outside of OS-defined time or memory limits, then an interrupt occurs. The CPU stops executing the program and the mode bit is changed from user-mode to OS-mode.

   The mode bit is changed by the CPU. The interrupt register is checked by the CPU after each instruction. If the register indicates that an interupt should occur, then the CPU stops executing the program in question and switches the mode of that program to OS.

2. *(2) What are system calls? How do they relate to the concept of dual-mode operation?*

   A system call is when a program makes a request from the operating system. For example, a system call could be to request access to a

device or the disk. Essentially, it allows the executing program to interface with the operating system. When a user program makes a system call, the mode switches from user mode to operating system mode for that program until the request has been fulfilled.

3. *(2) Give one advantage that programmed I/O has over interrupt driven I/O and DMA I/O.*

   Although the CPU is often left waiting with program-controlled I/O, one advantage of it over interrupt-driven I/O and DMA I/O is that the program has complete control over all data transfer by communicating directly with the devices involved. Another advantage is that this method is easy to implement.

4. *(2) In virtually all computers, DMA access to main memory is given higher priority than processor access to main memory. Why?*

   The DMA interrupts the CPU after it has finished transferring data in order to perform a DMA operation. This is called cycle stealing. A result of cycle stealing is that the CPU is on hold while it waits for the DMA operation to finish. Granting DMA higher priority means that the DMA will complete its cycles sooner. From the processor's perspective, the sooner the DMA needs to stop stealing cycles the better. If the processor had higher priority, then DMA transfers might be waiting so long that time-outs could occur.

5. *(4) A DMA module is transferring characters to main memory from an external device transmitting at 9600 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?*

   Since the DMA module is transmitting at 9600 bits per second, this is equivalent to 1200 bytes (characters) per second:

$$9600 \ \frac{\text{bits}}{\text{sec}} \cdot \frac{1 \,\text{char}}{8 \,\text{bits}} = 1200 \ \frac{\text{char}}{\text{sec}} \tag{1}$$

   The DMA module will steal a cycle from the processor every time a character arrives, therefore 1200 bytes (characters) will be processed per second. This is to be compared with the processor's rate of 1 million instructions per second.

$$\frac{1,200 \text{ bytes/second}}{1,000,000 \text{ bytes/second}} = 0.0012 = 0.12\% \tag{2}$$

This means that the processor is running at only 0.12% of its original speed. Therefore, the processor would be slowed down 99.88% by the DMA module's transfer of characters to an external device.

6. *(4) A disk transmits at rate 10^6 bytes per second. Suppose interrupt driven I/O is used, and the CPU is interrupted after every byte is transferred. If the CPU executes a billion instructions per second, and it takes 100 instructions to service an interrupt, then by what percentage is the CPU slowed by the disk?*

   Since the CPU is interrupted after every byte, there will be 1,000,000 interrupts. The interrupts will take 100 instructions each to service, so this requires 100,000,000 instructions (because $100 \cdot 1,000,000$). Therefore:

$$\frac{100000000 \text{ instructions}}{1000000000 \text{ instructions}} = 0.1 = 10\% \tag{3}$$

   Therefore, the CPU is running at 10% of its normal speed and has been slowed down by 90%.

7. *(3) Reconsider the last problem: Suppose a DMA module handles all disk transmissions. Suppose the CPU is interrupted after the disk transmits 10^3 bytes. By what percentage is the CPU slowed by the disk?*

   Since the disk is interrupted every 10^3 bytes, then the disk will be interrupted a total of 10^3 times:

$$\frac{10^6 \text{ bytes}}{10^3 \text{ bytes/interrupt}} = 10^3 \text{ interrupts} \tag{4}$$

   It will require 10^5 instructions (since $10^3 \cdot 100$) to service these interrupts. Therefore:

$$\frac{100000 \text{ instructions}}{1000000000 \text{ instructions}} = 0.0001 = 0.01\% \tag{5}$$

   Thus, the CPU is running at 0.01% of its normal speed and has been slowed down by 99.99%.

8. *(4)* ***multiple interrupts****: Interrupts have priority - the highest priority interrupt is serviced before a lower priority interrupt. Suppose a higher priority interrupt is set while a lower priority interrupt is being serviced. How should the computer deal with this situation? Justify your answer.*

   The interrupt with the higher priority should interrupt the servicing of the interrupt with the lower priority so it can be serviced. There are interrupts which are much more "fatal" than others. For example, a low priority interrupt may be for the print screen function, while a timer reaching zero or a hardware failure should have higher priority. The print screen function probably should not be carried out (and maybe is impossible to carry out) if there is a hardware failure or the executing program's timer reaches zero. Otherwise, the high priority interrupts may be ignored for a while which could lead to a big problem in the meantime.