SOLUTIONS TO CHAPTER 5 PROBLEMS

1. In the figure, we see controllers and devices as separate units. The reason is to allow a controller to handle multiple devices, and thus eliminate the need for having a controller per device. If controllers become almost free, then it will be simpler just to build the controller into the device itself. This design will also allow multiple transfers in parallel and thus give better performance.

2. Easy. The scanner puts out 400 KB/sec maximum. The wireless network runs at 6.75 MB/sec, so there is no problem at all.

3. It is not a good idea. The memory bus is surely faster than the I/O bus, otherwise why bother with it? Consider what happens with a normal memory request. The memory bus finishes first, but the I/O bus is still busy. If the CPU waits until the I/O bus finishes, it has reduced memory performance to that of the I/O bus. If it just tries the memory bus for the second reference, it will fail if this one is an I/O device reference. If there were some way to instantaneously abort the previous I/O bus reference to try the second one, the improvement might work, but there is never such an option. All in all, it is a bad idea.

4. An advantage of precise interrupts is simplicity of code in the operating system since the machine state is well defined. On the other hand, in imprecise interrupts, OS writers have to figure out what instructions have been partially executed and up to what point. However, precise interrupts increase complexity of chip design and chip area, which may result in slower CPU.

5. Each bus transaction has a request and a response, each taking 50 nsec, or 100 nsec per bus transaction. This gives 10 million bus transactions/sec. If each one is good for 4 bytes, the bus has to handle 40 MB/sec. The fact that these transactions may be sprayed over five I/O devices in round-robin fashion is irrelevant. A bus transaction takes 100 nsec, regardless of whether consecutive requests are to the same device or different devices, so the number of channels the DMA controller has does not matter. The bus does not know or care.

6. (a) Word-at-a-time mode: $1000 \times [(t_1 + t_2) + (t_1 + t_2) + (t_1 + t_2)]$
Where the first term is for acquiring the bus and sending the command to the disk controller, the second term is for transferring the word, and the third term is for the acknowledgement. All in all, a total of $3000 \times (t_1 + t_2)$ nsec.

(b) Burst mode: $(t_1 + t_2) + t_1 + 1000$ times $t_2 + (t_1 + t_2)$
where the first term is for acquiring the bus and sending the command to the disk controller, the second term is for the disk controller to acquire the bus, the third term is for the burst transfer, and the fourth term is for acquiring the bus and doing the acknowledgement. All in all, a total of $3t_1 + 1002t_2$.

7. Memory to memory copy can be performed by first issuing a read command that will transfer the word from memory to DMA controller and then issuing a write to memory to transfer the word from the DMA controller to a different address in memory. This method has the advantage that the CPU can do other useful work in parallel. The disadvantage is that this memory to memory copy is likely to be slow since DMA controller is much slower than CPU and the data transfer takes place over system bus as opposed to the dedicated CPU-memory bus.

8. An interrupt requires pushing 34 words onto the stack. Returning from the interrupt requires fetching 34 words from the stack. This overhead alone is 340 nsec. Thus the maximum number of interrupts per second is no more than about 2.94 million, assuming no work for each interrupt.

9. The execution rate of a modern CPU is determined by the number of instructions that finish per second and has little to do with how long an instruction takes. If a CPU can finish 1 billion instructions/sec it is a 1000 MIPS machine, even if an instruction takes 30 nsec. Thus there is generally little attempt to make instructions finish quickly. Holding the interrupt until the last instruction currently executing finishes may increase the latency of interrupts appreciably. Furthermore, some administration is required to get this right.

10. It could have been done at the start. A reason for doing it at the end is that the code of the interrupt service procedure is very short. By first outputting another character and then acknowledging the interrupt, if another interrupt happens immediately, the printer will be working during the interrupt, making it print slightly faster. A disadvantage of this approach is slightly longer dead time when other interrupts may be disabled.

11. Yes. The stacked PC points to the first instruction not fetched. All instructions before that have been executed and the instruction pointed to and its successors have not been executed. This is the condition for precise interrupts. Precise interrupts are not hard to achieve on machine with a single pipeline. The trouble comes in when instructions are executed out of order, which is not the case here.

12. The printer prints $50 \times 80 \times 6 = 24,000$ characters/min, which is 400 characters/sec. Each character uses 50 μsec of CPU time for the interrupt, so collectively in each second the interrupt overhead is 20 msec. Using interrupt-driven I/O, the remaining 980 msec of time is available for other work. In other words, the interrupt overhead costs only 2% of the CPU, which will hardly affect the running program at all.

13. UNIX does it as follows. There is a table indexed by device number, with each table entry being a C struct containing pointers to the functions for opening, closing, reading, and writing and a few other things from the device. To install a new device, a new entry has to be made in this table and the pointers filled in, often to the newly loaded device driver.

14. (a) Device driver.
(b) Device driver.
(c) Device-independent software.
(d) User-level software.

15. A packet must be copied four times during this process, which takes 4.1 msec. There are also two interrupts, which account for 2 msec. Finally, the transmission time is 0.83 msec, for a total of 6.93 msec per 1024 bytes. The maximum data rate is thus 147,763 bytes/sec, or about 12% of the nominal 10 megabit/sec network capacity. (If we include protocol overhead, the figures get even worse.)

16. If the printer were assigned as soon as the output appeared, a process could tie up the printer by printing a few characters and then going to sleep for a week.

17. The disk rotates at 120 RPS, so 1 rotation takes 1000/120 msec. With 200 sectors per rotation, the sector time is 1/200 of this number or 5/120 = 1/24 msec. During the 1-msec seek, 24 sectors pass under the head. Thus the cylinder skew should be 24.

18. At 7200 RPM, there are 120 rotations per second, so I rotation takes about 8.33 msec. Dividing this by 500 we get a sector time of about 16.67 μsec.

19. There are 120 rotations in a second. During one of them, 500×512 bytes pass under the head. So the disk can read 256,000 bytes per rotation or 30,720,000 bytes/sec.

20. RAID level 2 can not only recover from crashed drives, but also from undetected transient errors. If one drive delivers a single bad bit, RAID level 2 will correct this, but RAID level 3 will not.

21. The probability of 0 failures, P_0, is $(1 - p)^k$. The probability of 1 failure, P_1, is $kp(1 - p)^{k-1}$. The probability of a RAID failure is then $1 - P_0 - P_1$. This is $1 - (1 - p)^k - kp(1 - p)^{k-1}$.

22. Read performance: RAID levels 0, 2, 3, 4, and 5 allow for parallel reads to service one read request. However, RAID level 1 further allows two read requests to simultaneously proceed. Write performance: All RAID levels provide similar write performance. Space overhead: There is no space overhead in level 0 and 100% overhead in level 1. With 32-bit data word and six parity drives, the space overhead is about 18.75% in level 2. For a 32-bit data word, the space overhead in level 3 is about 3.13%. Finally, assuming 33 drives in levels 4 and

5, the space overhead is 3.13% in them. Reliability: There is no reliability support in level 0. All other RAID levels can survive one disk crash. In addition, in levels 3, 4 and 5, a single random bit error in a word can be detected, while in level 2, a single random bit error in a word can be detected and corrected.

23. A zebibyte is 2^{70} bytes; a pebibyte is 2^{50}. 2^{20} pebibytes fit in a zebibyte.

24. A magnetic field is generated between two poles. Not only is it difficult to make the source of a magnetic field small, but also the field spreads rapidly, which leads to mechanical problems trying to keep the surface of a magnetic medium close to a magnetic source or sensor. A semiconductor laser generates light in a very small place, and the light can be optically manipulated to illuminate a very small spot at a relatively great distance from the source.

25. The main advantage of optical disks is that they have much higher recording densities than magnetic disks. The main advantage of magnetic disks is that they are an order of magnitude faster than the optical disks.

26. Possibly. If most files are stored in logically consecutive sectors, it might be worthwhile interleaving the sectors to give programs time to process the data just received, so that when the next request is issued, the disk would be in the right place. Whether this is worth the trouble depends strongly on the kind of programs run and how uniform their behavior is.

27. Maybe yes and maybe no. Double interleaving is effectively a cylinder skew of two sectors. If the head can make a track-to-track seek in fewer than two sector times, than no additional cylinder skew is needed. If it cannot, then additional cylinder skew is needed to avoid missing a sector after a seek.

28. Consider,

(a) The capacity of a zone is tracks × cylinders × sectors/cylinder × bytes/sect.

Capacity of zone 1: $16 \times 100 \times 160 \times 512 = 131072000$ bytes
Capacity of zone 2: $16 \times 100 \times 200 \times 512 = 163840000$ bytes
Capacity of zone 3: $16 \times 100 \times 240 \times 512 = 196608000$ bytes
Capacity of zone 4: $16 \times 100 \times 280 \times 512 = 229376000$ bytes
Sum = 131072000 + 163840000 + 196608000 + 229376000 = 720896000

(b) A rotation rate of 7200 means there are 120 rotations/sec. In the 1 msec track-to-track seek time, 0.120 of the sectors are covered. In zone 1, the disk head will pass over 0.120×160 sectors in 1 msec, so, optimal track skew for zone 1 is 19.2 sectors. In zone 2, the disk head will pass over 0.120×200 sectors in 1 msec, so, optimal track skew for zone 2 is 24 sectors. In zone 3, the disk head will pass over 0.120×240 sectors in 1 msec, so, optimal track skew for zone 3 is 28.8 sectors. In zone 4, the disk head will pass over 0.120×280 sectors in 1 msec, so, optimal track skew for zone 3 is 33.6 sectors.

(c) The maximum data transfer rate will be when the cylinders in the outermost zone (zone 4) are being read/written. In that zone, in one second, 280 sectors are read 120 times. Thus the data rate is $280 \times 120 \times 512 = 17{,}203{,}200$ bytes/sec.

29. The drive capacity and transfer rates are doubled. The seek time and average rotational delay are the same. No properties are worse.

30. One fairly obvious consequence is that no existing operating system will work because they all look there to see where the disk partitions are. Changing the format of the partition table will cause all the operating systems to fail. The only way to change the partition table is to simultaneously change all the operating systems to use the new format.

31. (a) 10 + 12 + 2 + 18 + 38 + 34 + 32 = 146 cylinders = 876 msec.
(b) 0 + 2 + 12 + 4 + 4 + 36 +2 = 60 cylinders = 360 msec.
(c) 0 + 2 + 16 + 2 + 30 + 4 + 4 = 58 cylinders = 348 msec.

32. In the worst case, a read/write request is not serviced for almost two full disk scans in the elevator algorithm, while it is at most one full disk scan in the modified algorithm.

33. A disadvantage of one-shot mode is that the time consumed by interrupt handler is unaccounted for as the process of decrementing the counter is paused during this time. A Disadvantage of square-wave mode is that high clock frequencies may result in multiple interrupts being queued when new interrupts are raised before the previous ones have been serviced.

34. Not necessarily. A UNIX program that reads 10,000 blocks issues the requests one at a time, blocking after each one is issued until after it is completed. Thus the disk driver sees only one request at a time; it has no opportunity to do anything but process them in the order of arrival. Harry should have started up many processes at the same time to see if the elevator algorithm worked.

35. There is a race but it does not matter. Since the stable write itself has already completed, the fact that the nonvolatile RAM has not been updated just means that the recovery program will know which block was being written. It will read both copies. Finding them identical, it will change neither, which is the correct action. The effect of the crash just before the nonvolatile RAM was updated just means the recovery program will have to make two disk reads more than it should.

36. Yes the disk remains consistent even if the CPU crashes during a recovery procedure. Consider Fig. 5-31. There is no recovery involved in (a) or (e). Suppose that the CPU crashes during recovery in (b). If CPU crashes before the block from drive 2 has been completely copied to drive 1, the situation remains same as earlier. The subsequent recovery procedure will detect an

ECC error in drive 1 and again copy the block from drive 2 to drive 1. If CPU crashes after the block from drive 2 has been copied to drive 1, the situation is same as that in case (e). Suppose that the CPU crashes during recovery in (c). If CPU crashes before the block from drive 1 has been completely copied to drive 2, the situation is same as that in case (d). The subsequent recovery procedure will detect an ECC error in drive 2 and copy the block from drive 1 to drive 2. If CPU crashes after the block from drive 1 has been copied to drive 2, the situation is same as that in case (e). Finally, suppose the CPU crashes during recovery in (d). If CPU crashes before the block from drive 1 has been completely copied to drive 2, the situation remains same as earlier. The subsequent recovery procedure will detect an ECC error in drive 2 and again copy the block from drive 1 to drive 2. If CPU crashes after the block from drive 1 has been copied to drive 2, the situation is same as that in case (e).

37. Problems arise in scenarios shown in Figure 5-27 (b) and 5-27 (d), because they may look like scenario 5-27 (c), if the ECC of the corrupted block is correct. In this case, it is not possible to detect which disk contains the valid (old or new) lock, and a recovery is not possible.

38. Two msec 60 times a second is 120 msec/sec, or 12% of the CPU

39. With these parameters,

(a) Using a 500 MHz crystal, the counter can be decremented every 2 nsec. So, for a tick every millisecond, the register should be 1000000/2 = 500,000.

(b) To get a clock tick every 100 μsec, holding register value should be 50,000.

40. At time 5000:
Current time = 5000; Next Signal = 8; Header $\rightarrow 8 \rightarrow 4 \rightarrow 3 \rightarrow 14 \rightarrow 8$.

At time 5005:
Current time = 5005; Next Signal = 3; Header $\rightarrow 3 \rightarrow 4 \rightarrow 3 \rightarrow 14 \rightarrow 8$.

At time 5013:
Current time = 5013; Next Signal = 2; Header $2 \rightarrow 14 \rightarrow 8$.

At time 5023:
Current time = 5023; Next Signal = 6; Header $\rightarrow 6 \rightarrow 4 \rightarrow 5$.

41. The number of seconds in a mean year is $365.25 \times 24 \times 3600$. This number is 31,557,600. The counter wraps around after 2^{32} seconds from 1 January 1970. The value of 2^{32}/31,557,600 is 136.1 years, so wrapping will happen at 2106.1, which is early February 2106. Of course, by then, all computers will be at least 64 bits, so it will not happen at all.

42. Scrolling the window requires copying 79 lines of 80 characters or 6320 characters. Copying 1 character (16 bytes) takes 800 nsec, so the whole window takes 5.056 msec. Writing 80 characters to the screen takes 400 nsec, so scrolling and displaying a new line take 5.456 msec. This gives about 183.2 lines/sec.

43. Suppose that the user inadvertently asked the editor to print thousands of lines. Then he hits DEL to stop it. If the driver did not discard output, output might continue for several seconds after the DEL, which would make the user hit DEL again and again and get frustrated when nothing happened.

44. It should move the cursor to line 5 position 7 and then delete 6 characters. The sequence is ESC [5 ; 7 H ESC [6 P

45. The maximum rate the mouse can move is 200 mm/sec, which is 2000 mickeys/sec. If each report is 3 byte, the output rate is 6000 bytes/sec.

46. With a 24-bit color system, only 2^{24} colors can be represented. This is not all of them. For example, suppose that a photographer takes pictures of 300 cans of pure blue paint, each with a slightly different amount of pigment. The first might be represented by the (R, G, B) value (0, 0, 1). The next one might be represented by (0, 0, 2), and so forth. Since the B coordinate is only 8 bits, there is no way to represent 300 different values of pure blue. Some of the photographs will have to be rendered as the wrong color. Another example is the color (120.24, 150.47, 135.89). It cannot be represented, only approximated by (120, 150, 136).

47.

(a) Each pixel takes 3 bytes in RGB, so the table space is $16 \times 24 \times 3$ bytes, which is 1152 bytes.

(b) At 100 nsec per byte, each character takes 115.2 μsec. This gives an output rate of about 8681 chars/sec.

48. Rewriting the text screen requires copying 2000 bytes, which can be done in 4 μseconds. Rewriting the graphics screen requires copying $1024 \times 768 \times 3 =$ 2,359,296 bytes, or about 4.72 msec.

49. In Windows, the OS calls the handler procedures itself. In X Windows, nothing like this happens. X just gets a message and processes it internally.

50. The first parameter is essential. First of all, the coordinates are relative to some window, so *hdc* is needed to specify the window and thus the origin. Second, the rectangle will be clipped if it falls outside the window, so the window coordinates are needed. Third, the color and other properties of the rectangle are taken from the context specified by *hdc*. It is quite essential.

51. The display size is $400 \times 160 \times 3$ bytes, which is 192,000 bytes. At 10 fps this is 1,920,000 bytes/sec or 15,360,000 bits/sec. This consumes 15% of the Fast Ethernet.

52. The bandwidth on a network segment is shared, so 100 users requesting different data simultaneously on a 1-Mbps network will each see a 10-Kbps effective speed. With a shared network, a TV program can be multicast, so the video packets are only broadcast once, no matter how many users there are and it should work well. With 100 users browsing the Web, each user will get 1/100 of the bandwidth, so performance may degrade very quickly.

53. Advantages of thin clients include low cost and no need for complex management for the clients. Disadvantages include (potentially) lower performance due to network latency and (potential) loss of privacy because the client's data/information is shared with the server.

54. If $n = 10$, the CPU can still get its work done on time, but the energy used drops appreciably. If the energy consumed in 1 sec at full speed is E, then running at full speed for 100 msec then going idle for 900 msec uses $E/10$. Running at 1/10 speed for a whole second uses $E/100$, a saving of $9E/100$. The percent savings by cutting the voltage is 90%.

55. The windowing system uses much more memory for its display and uses virtual memory more than the text mode. This makes it less likely that the hard disk will be inactive for a period long enough to cause it to be automatically powered down.