STRUCTURED COMPUTER ORGANIZATION

FIFTH EDITION

PROBLEM SOLUTIONS

ANDREW S. TANENBAUM

Vrije Universiteit
Amsterdam, The Netherlands

PRENTICE HALL

UPPER SADDLE RIVER, NJ 07458

SOLUTIONS TO CHAPTER 1 PROBLEMS

1. a. A translator converts programs in one language to another.
b. An interpreter carries out a program instruction by instruction.
c. A virtual machine is a conceptual machine, one that does not exist.

2. An interpreter executes a program by fetching the first instruction, carrying it out, then fetching the next one, and so on. A translator first converts the original program into an equivalent one in another language and then runs the new program.

3. It is possible, but there are problems. One difficulty is the large amount of code produced. Since one ISA instruction does the work of many microinstructions, the resulting program will be much bigger. Another problem is that the compiler will have to deal with a more primitive output language, hence it, itself, will become more complex. Also, on many machines, the microprogram is in ROM. Making it user-changeable would require putting it in RAM, which is much slower than ROM. On the positive side, the resulting program might well be much faster, since the overhead of one level of interpretation would be eliminated.

4. During the detailed design of a new computer, the device and digital logic levels of the new machine may well be simulated on an old machine, which puts them around level 5 or 6.

5. Each level of interpretation slows down the machine by a factor of n/m. Thus the execution times for levels 2, 3, and 4 are kn/m, kn^2/m^2, and kn^3/m^3, respectively.

6. Each additional level of interpretation costs something in time. If it is not needed, it should be avoided.

7. You lose a factor of n at each level, so instruction execution times at levels 2, 3, and 4 are kn, kn^2, and kn^3, respectively.

8. Hardware and software are functionally equivalent. Any function done by one can, in principle, be done by the other. They are not equivalent in the sense that to make the machine really run, the bottom level must be hardware, not software. They also differ in performance.

9. Not at all. If you wanted to change the program the difference engine ran, you had to throw the whole computer out and build a new one. A modern computer does not have to be replaced because you want to change the program. It can read many programs from many CD-ROMs.

10. A typical example is a program that computes the inner product of two arrays, *A* and *B*. The first two instructions might fetch *A*[0] and *B*[0], respectively. At the end of the iteration, these instructions could be incremented to point to *A*[1] and *B*[1], respectively. Before indexing and indirect addressing were invented, this was done.

11. Raw cycle time is not the only factor. The number of bytes fetched per cycle is also a major factor, this increasing with the larger models. Memory speed and wait states play a role, as does the presence of caching. A better I/O architecture causes fewer cycles to be stolen, and so on.

12. The design of Figure 1-5 does I/O one character at a time by explicit program command. The design of Figure 1-6 can use DMA to have the controller do all the work, relieving the CPU of the burden, and thus making it potentially better.

13. Each person consumes 730 tags per nonleap year. Multiply by 300 million and you get 219 billion tags a year. At a penny a tag, they cost $2.19 billion dollars a year. With GDP exceeding $10 trillion, the tags add up to 0.02% of GDP, not a huge obstacle.

14. The following appliances are normally controlled by embedded systems these days: alarm-clock radios, microwave ovens, television sets, cordless telephones, washing machines, sewing machines, and burglar alarms.

15. According to Moore’s law, next year the same chip will have 1.6 times as many transistors. This means that the area of each transistor will be 1/1.6 or 0.625 times the size of this year’s transistors. Since the area goes like the square of the diameter, the diameter of next year’s transistors must be 0.079 microns.

SOLUTIONS TO CHAPTER 2 PROBLEMS

1. The data path cycle is 20 nsec. The maximum number of data path cycles/sec is thus 50 million. The best the machine could do is thus 50 MIPS.

2. The program counter must be incremented to point to the next instruction. If this step were omitted, the computer would execute the initial instruction forever.

3. You cannot say anything for sure. If computer 1 has a five-stage pipeline, it can issue up to 500 million instructions/second. If computer 2 is not pipelined, it cannot do any better than 200 million instructions/sec. Thus without more information, you cannot say which is faster.

4. On-chip memory does not affect the first three principles. Having only LOADs and STOREs touch memory is no longer required. There is no particular reason not to have a memory-to-memory architecture if memory references are as fast as register references. Likewise, the need for many registers becomes less in this environment.

5. A pipeline processor is better. Array processors are useful only if the problem contains inherent parallelism.

6. The monastery resembles Figure 2-7, with one master and many slaves.

7. The access time for registers is a few nanoseconds. For optical disk it is a few hundred milliseconds. The ratio here is about 10^8.

8. Sixty-four 6-bit numbers exist, so 4 trits are needed. In general, the number of trits, k, needed to hold n bits is the smallest value of k such that $3^k \geq 2^n$.

9. A pixel requires 6 + 6 + 6 = 18 bits, so a single visual frame is 1.8×10^7 bits. With 10 frames a second, the gross data rate is 180 Mbps. Unfortunately, the brain's processing rate is many orders of magnitude less than this. As an experiment, try watching the random noise on a color television when no station is broadcasting and see if you can memorize the color bit pattern in the noise for a few minutes.

10. With 44,000 samples per second of 16 bits each, we have a data rate of 704 kbps.

11. There are 2 bits per nucleotide, so the information capacity of the human genome is about 6 gigabits. Dividing this number by 30,000, we get about 200,000 bits per gene. Just think of a gene as an 25-KB ROM. This estimate is an upper bound, because many of the nucleotides are used for purposes other than coding genes.

12. For efficiency with binary computers, it is best to have the number of cells be a power of 2. Since 268,435,456 is 2^{28}, it is reasonable, whereas 250,000,000 is not.

13. From 0 to 9 the codes are: 0000000, 1101001, 0101010, 1000011, 1001100, 0100101, 1100110, 0001111, 1110000, and 0011001.

14. Just add a parity bit: 00000, 00011, 00101, 00110, 01001, 01010, 01100, 01111, 10001, and 10010.

15. If the total length is $2^n - 1$ bits, there are n check bits. Consequently, the percentage of wasted bits is $n/(2^n - 1) \times 100\%$. Numerically for n from 3 to 10 we get: 42.9%, 26.7%, 16.1%, 9.5%, 5.5%, 3.1%, 1.8%, and 1.0%.

16. With 4096 bits/sector and 1024 sectors/track, each track holds 4,194,304 bits At 7200 RPM, each rotation takes 1/120 sec. In 1 sec it can read 120 tracks for a rate of 503,316,480 bits/sec or 62,914,560 bytes/sec.

17. At 160 Mbytes/sec and 4 bytes/word, the disk transfer rate is 40 million words/sec. Of the 200 million bus cycles/sec, the disk takes 1/5 of them. Thus the CPU will be slowed down by 20 percent.

18. Logically it does not matter, but the performance is better if you allocate from the outside in. One rotation of the outermost track takes as long as one rotation of the innermost track (because hard disks rotate with constant angular velocity), but there are more sectors on the outermost track, so the transfer rate is higher. It is smarter to use the high-performance sectors first. Maybe the disk will never fill up and you will never have to use the lowest-performance sectors.

19. A cylinder can be read in four rotations. During the fifth rotation, a seek is done to the next cylinder. Because the track-to-track seek time is less than the rotation time, the program must wait until sector zero comes around again. Therefore, it takes five full rotations to read a cylinder and be positioned to start reading the next one. Reading the first 9999 cylinders thus takes 49,995 rotations. Reading the last cylinder requires four rotations, because no final seek is needed. The 49,999 rotations at 10 msec/rotation take 499.99 sec. If the sectors are skewed, however, it may be possible to avoid the fifth rotation per track.

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. In mode 2, the data streams at 175,200 bytes/sec. In a 80-min time span, the number of seconds is 5920, so the size of a 80-min mode 2 CD-ROM is 840,960,000 bytes or 802 MB. Of course, in mode 2 there is no error correction, which is fine for music but not for data. In mode 1, only 2048/2336 of the bits are available for data, reducing the payload to 737,280,000 bytes or 703 MB.

22. The mode does not matter, since the laser has to pulse for preamble bits, data bits, ECC bits, and all the overhead bits as well. The gross data rate at 1x is 75 sectors/sec, each sector consisting of $98 \times 588 = 57,624$ bits. Thus 4,321,800 bits/sec fly by the head at 1x. At 10x, this is 43,218,000 bits/sec. Thus each pulse must last no more than 23.14 nsec (actually slightly less, since there is a blank interval between pulses).

23. Each frame contains 345,600 pixels or 1,036,800 bytes of information. At 30 fps, the rate per second is 31,104,000 bytes/sec. In 133 minutes this amounts to 2.482×10^{11} bytes. The disk capacity is 3.5×2^{30} which is about

3.758×10^9 bytes. We are off by a factor of 66, so the compression has to be 66x.

24. The read time is the size divided by the speed: $25 \times 2^{30} / 4.5 \times 10^{20}$ or about 5688.9 sec. This is almost an hour and 35 minutes.

25. The CPU wastes millions of instructions while waiting for mechanical I/O (e.g., a printer) to finish. The problem can be handled using DMA and causing an interrupt after a block has been transferred. In some cases, the CPU can copy a block of data to the device, which then handles it on its own and causes an interrupt when it is ready for more.

26. The usual way to handle this would be to have a bank of 256 24-bit mapping registers in the hardware. Whenever a byte was fetched from the video RAM, the 8-bit number would be used as an index into the mapping registers. The register selected would deliver the 24 bits to drive the display (typically 8 bits each for the red, green, and blue electron guns). Thus indeed 2^{24} colors are available, but at any instant, only 256 are available. Changing colors means reloading the mapping registers.

27. The display must paint $1600 \times 1200 \times 75$ pixels/sec. This is a total of 144,000,000 pixels Thus the pixel time is 6.944 nsec.

28. A page has 4000 characters. Each character uses 25% of 4 mm^2, or 1 mm^2. Thus a page has 4000 mm^2 of toner. With a thickness of 25 microns (0.025 mm), the volume of toner on a page is 100 mm^3. The capacity of the toner cartridge is 400 cm^3 or 400,000 mm^3. A cartridge is good for 4000 pages.

29. At any speed above 110 bps, there is one start bit, one stop bit, and one parity bit, so 7 of the 10 bits are data. Thus 70 percent of the bits are data. Since $0.7 \times 5600 = 39,200$, the true data rate is 39,200 bps.

30. Each interval can transmit 6 bits, so the data rate is 6*n* bps.

31. A 12 MHz cable with QAM-64 has a data rate of 72 Mbps. With *nf* computers sharing the bandwidth, each user gets $72/nf$-Mbps. Thus the cable user gets better service if $72/nf > 2$. An alternative way to write this is $nf < 36$. In other words, if the 72 Mbps bandwidth is being shared by 36 active users, it is the same as 2-Mbps ADSL; with fewer users, cable wins; with more users, ADSL wins.

32. Each uncompressed image file is 18 million bytes. After 5x compression, it is 3.6 million bytes. To write this in 2 sec requires a data rate of 1.8 MB/sec.

33. The uncompressed image is 48 million bytes. The compressed image is 9.6 million bytes. The number of images stored is thus $2^{30}/9.6$ million or 111 with a few megabytes left over.

34. A typical computer science textbook has about a million characters, so it needs about 1 MB. Ten thousand books require 10^{10} bytes. A CD-ROM holds 650 MB, so you need 16 CD-ROMs. A dual-layer, double-sided DVD holds 17 GB, so the whole library fits on one DVD.

SOLUTIONS TO CHAPTER 3 PROBLEMS

1. The order is

hamburger or hot dog and french fries

which can be parsed as

hamburger or (hot dog and french fries)

or as

(hamburger or hot dog) and french fries

The first parse leads to alternatives a and d. The second parse leads to alternatives e and d. Thus a, d, and e are all possible. Choice h is ruled out on educational grounds: the cook is too stupid to get the joke.

2. He should point to one of the roads and ask: "If I were to ask the other gang if this is the road to Disneyland, what would they say?" If the answer is no, he should take the road; if the answer is yes, he should not take it. The validity of this solution can easily be seen by trying all four combinations of right/wrong road and liars/truth-tellers.

3. The truth table required is as follows:

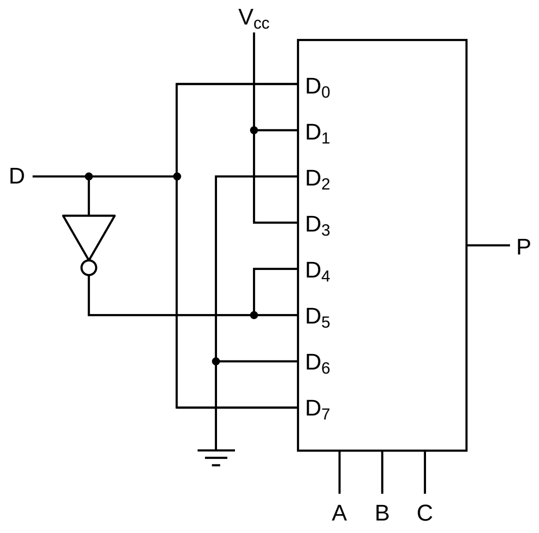
P	Q	P AND Q	P AND NOT Q	(P AND Q) OR (P AND NOT Q)
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

4. With three variables, the truth table has eight rows, so a function can be described by an 8-bit number. Thus 256 functions exist. With n variables, the truth table has $k = 2^n$ rows and there are 2^k functions.

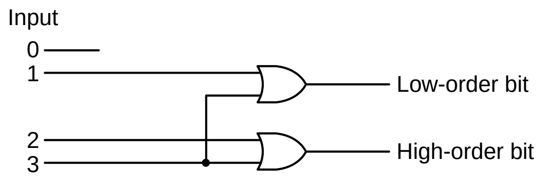
5. Call the two variables A and B. Connect the inputs to the first NAND gate to A and B. Take the output and feed it into both inputs of the second NAND gate and presto, AND.

6. Inputs D_1, D_2, D_4, and D_7 are all connected to ground The other four inputs are tied to V_{cc}.

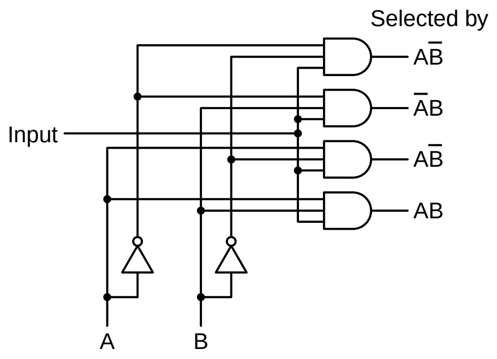
7. Input line D_0 supplies the output for truth table rows 0000 and 0001. Input line D_1 supplies the output for truth table rows 0010 and 0011, and so on. For each case, the function values for the two rows can be 00, 01, 10, and 11. If they are 00, just wire the input to ground; if they are 11, just wire it to V_{cc}. If they are 01, notice that they are just the same as the fourth input variable, D, so wire it to D. If they are 10, wire it to $\overline{D}$. The truth table values for the example, from 0000 to 1111, are 0111001110100001, so the circuit is as follows.



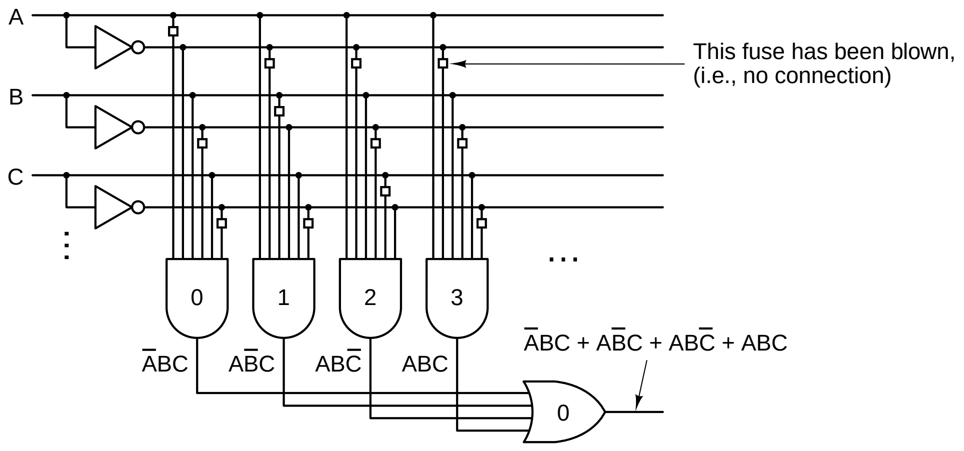
8. The encoder looks like this. Note that line 0 is not used.



9. The demultiplexer looks like this.



10. The relevant part of the PLA is as follows.


A
B
C
This fuse has been blown, (i.e., no connection)
0
1
2
3
$\overline{A}BC$
$A\overline{B}C$
$AB\overline{C}$
ABC
$\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$
0

11. It is a half adder with *C* as the sum and *D* as the carry.

12. The chip needs four pins for the first operand, four pins for the second operand, four pins for the result, one pin for carry in, and one pin for carry out (to make it cascadable), plus power and ground, for a total of 16 pins.

13. The carry into stage *i* can be written as $C_i = P_{i-1} + S_{i-1}C_{i-1}$, where P_{i-1} is the product term $A_{i-1}B_{i-1}$ and S_{i-1} is the sum term $A_{i-1} + B_{i-1}$. This result follows directly from the fact that a carry is generated from a stage if both operands are 1, or if one operand and the carry in are both 1. For example,

$$C_0 = 0$$

$$C_1 = P_0 + S_0C_0 = P_0$$

$$C_2 = P_1 + S_1C_1 = P_1 + P_0S_1$$

$$C_3 = P_2 + S_2C_2 = P_2 + P_1S_2 + P_0S_1S_2$$

$C_4 = P_3 + S_3C_3 = P_3 + P_2S_3 + P_1S_2S_3 + P_0S_1S_2S_3$

As soon as the inputs, *A* and *B*, are available, all the *P* and *S* terms can be generated simultaneously in one gate delay time. Then the various AND terms such as $P_0S_1S_2$ can be generated in a second gate delay. Finally, all the carries can be produced in a third gate delay. Thus all the carries are available after three gate delays, no matter how many stages the adder has. The price paid for this speedup is a considerable number of additional gates, of course.

14. The timing of the circuit can be found by writing a 0 on each of the input lines and then tracing them through the circuit, adding 1 at each gate. The *A* input takes 2 nsec to become available, so the output of the logic unit takes 4 nsec and the final output for a Boolean operation takes 5 nsec. The decode lines that drive the logic unit each have two gate delays, so the enable lines are set in plenty of time. The adder also takes 3 nsec to produce its contribution to the output gate after it has *A*. Worst case through the whole circuit is 6 nsec.

15. The ALU is already capable of subtraction. Note that in 2's complement, $B - A = B + (-A)$. To get $-A$, we can use $\overline{A} + 1$. Thus to subtract *A* from *B* we need to add B, $\overline{A}$ and then increment the sum. The circuit can already do this by asserting INVA and INC and then adding the two inputs.

16. A basic cycle is 11 nsec, including propagation. Sixteen cycles take 176 nsec. However, the last propagation is not needed, so the answer is available 175 nsec after the start.

17. One way is to negate ENB, forcing *B* to 0, then choosing function code 01 to select $\overline{B}$ as the ALU function. The 1's complement of 0 is −1 in 2's complement (all 1 bits). As long as INC is negated, the control lines for *A* do not matter. A second way is to negate and invert *A*, putting all 1s on the *A* input. Then negate ENB to force *B* to 0. With *A* equal to −1 and *B* equal to 0, we can either add or OR them together.

18. The NAND latch is wired the same way as the NOR latch. Normally, both inputs should be 1 to achieve consistency between input and output.

19. Use the same circuit, but replace the AND gate in the pulse generator by a NOR gate. The only time both inputs will be low is just after the falling edge.

20. The design uses two AND gates for chip enable logic plus two AND gates per word select line plus one AND gate per data bit. For a 256×8 memory this comes to 2 + 512 + 2048 = 2562 AND gates. The circuit also uses one OR gate for each bit in the word; hence eight of them would be needed.

21. It will not fly. Each of the four flip-flops needs three pins, for *D*, *Q*, and $\overline{Q}$. In addition, the chip needs clock, power, and ground. Unless you plan on putting out the world's first 15-pin chip, you are going to have to use a 16-pin package, thus wasting a pin. Thus the design is not very efficient. Since an extra pin is available, you might as well do something with it to make the chip more flexible. For example, you might use the sixteenth pin to reset all the flip-flops.

22. The pins can be multiplexed in time. For example, with $n/2$ pins, half the bits are presented in one cycle, and the other half in a succeeding cycle. Many dynamic RAMs work this way. Even more extreme is to feed the address serially into the chip a bit at a time using a single pin.

23. A 32-bit-wide data bus means that 32 chips must be used in parallel, each chip providing 1 bit. Thus the smallest memory consists of 32 chips, which is 32 megabits or 4 Mbytes.

24. With a 20-nsec clock period, $\overline{\text{MREQ}}$ might be asserted as late as 13 nsec into T_1. The data are required 2 nsec before the high-to-low transition in T_3, which occurs 10 nsec after the start of the cycle. From the midpoint of T_1 to the midpoint of T_3 is 40 nsec. Since the memory cannot start until 3 nsec after the transition in the first cycle and has to be done 2 nsec before the transition in the third cycle, in the worst case the memory has only 35 nsec in which to respond.

25. The memory would now have $20 - 3 - 4 = 13$ nsec to respond after $\overline{\text{MREQ}}$ and $\overline{\text{RD}}$ are asserted. There is not a lot of margin left, but if all the memory chips can always respond in 10 nsec, the system will still work.

26. In principle it could be negative, but it has to be asserted after the address is stable because on a write, the memory must not start changing the bytes addressed until the address is valid. On reads it does not matter.

27. In regular mode, it takes three cycles per transfer. In block mode, it takes one cycle per transfer once the transfer has started. Therefore block transfers have almost three times as much bandwidth. The bus width does not matter, so it is still three times more, even for 32-bit transfers.

28. The CPU cannot assert $\overline{\text{MSYN}}$ until it has provided an address and told what it wants, so we have

$T_{A1} < T_{MSYN1}$,
$T_{MREQ1} < T_{MSYN1}$, and
$T_{RD1} < T_{MSYN1}$.

The data cannot be asserted before $\overline{\text{MSYN}}$, so

$T_{MSYN1} < T_{D1}$ and
$T_{D1} < T_{SSYN1}$.

Once $\overline{\text{SSYN}}$ has been asserted, the master can clean up, so

$T_{SSYN1} < T_{MSYN2}$,
$T_{SSYN1} < T_{RD2}$,
$T_{SSYN1} < T_{MREQ2}$, and
$T_{SSYN1} < T_{A2}$.

Finally,

$T_{MSYN2} < T_{D2}$ and
$T_{MSYN2} < T_{SSYN2}$.

29. Yes there is. When reading the lower half of a word, the lower half of the 32 data lines is used. However, when reading the upper half, the bus designers have a choice of using the upper half of the lines or shifting everything downward to the lower half. The former is an unjustified bus; the latter is a justified bus. Both types are in use.

30. The interrupt acknowledge cycle is needed so that the interrupting device (or the interrupt controller) can see that the interrupt has been accepted and the CPU wants the interrupt vector number. Since the CPU can disable interrupts for an arbitrarily long time, after asserting an interrupt, a device might have to wait a long time before the interrupt will be accepted, so it needs some way to see on which cycle it must output the interrupt vector number.

31. At 200 MHz, a cycle is 5 nsec. Four cycles take 20 nsec. A word is 8 bytes, so the machine needs 8 bytes every 20 nsec, for a data rate of 400 Mbytes/sec.

32. There are seven combinations: one for words, two for half-words (upper and lower), and four for bytes. Three pins are enough for seven combinations if complete encoding is used. This minimizes pins. On the other hand, it requires an external decoder, so there is something to be said for having four lines, one per byte, saying whether that byte is needed or not.

33. The Pentium 4 is capable of addressing 2^{36} bytes of memory. If they were addressed a word at time, there would be 2^{34} possible words to address. It would take a 34-bit address to address them all. However, the Pentium 4 has only 33 address lines. In theory, it could ask for a 64-bit word in two consecutive 32-bit cycles, but this change would mean redesigning the processor.

34. The average access time is $0.20 \times 1 + 0.60 \times 2 + 0.20 \times 10 = 3.4$ nsec.

35. Not very likely. The 8255A provides 24 I/O lines, but the 8051 already has 32 such lines built in. Only if more than 16 are needed would an 8255A make sense.

36. Each frame contains 921,600 bytes. In one second, 30 such frames are needed, or 27,648,000 bytes. If the data travel over the bus twice, the needed bandwidth is 55,296,000 bytes/sec or 52.734 MB/sec (1 MB is 2^{20} bytes.)

37. The FRAME# signal starts a PCI bus transaction. The Pentium 4 signal that says that it is ready to go is ADS#, so this should connect to FRAME#.

38. The DEVSEL# signal is not really essential. It is nice to know that somebody out there wants you, but the real information is conveyed by TRDY#.

39. For 8x operation, 8 lanes are needed each way for a gross capacity of 40 Gbps. The net capacity is then 32 Gbps.

40. With a 32-bit bus, the disk must make 262,144 transfers in 5 msec (one rotation) or about 52,429 transfers/msec. In this time, it uses up 52,429 of the 100,000 bus cycles available per millisecond, leaving only 47,571 for the CPU. The CPU is therefore slowed down to 47.571 percent of its full speed.

41. The frame rate is 1000 frames/sec. At 64 bytes/frame, the maximum data rate for a device is 1,023,000 bytes/sec or 999.023 MB/sec.

42. As the circuit now stands, the PIO is selected by all addresses beginning with 11, that is, any address above 48K. Adding a third line causes it to be selected only by addresses above 56K.

SOLUTIONS TO CHAPTER 4 PROBLEMS

1. Only one register may be gated out onto the B bus, so a single 4-bit field is sufficient to specify which one. The C bus may have many registers selected to be loaded, so a full bit map is required.

2. The Boolean function that is needed is

```
F = (JAMZ AND Z) OR (JAMN AND N) OR HIBIT
```

where HIBIT is the high-order bit of the NEXT_ADDRESS field. This can be implemented with two AND gates that feed an OR gate.

3. No. No matter what is in *MBR*, the next microinstruction will come from 0x1FF. The contents of *MBR* are completely irrelevant. There is no point ORing *MBR* with 0x1FF since the result will always be 0x1FF, no matter what is in *MBR*. The OR does nothing.

4. All compilers will add

```
BIPUSH 5
ISTORE k
```

at the label *L2*. However, an optimizing compiler will notice that *i* is never used after the if statement. Therefore the fourth and fifth instructions (*ISTORE* and *ILOAD*) are not needed. The code becomes