9. How are complement systems like the odometer on a bicycle?

10. Do you think that double-dabble is an easier method than the other binary-to-decimal conversion methods explained in this chapter? Why?

11. With reference to the previous question, what are the drawbacks of the other two conversion methods?

12. What is overflow and how can it be detected? How does overflow in unsigned numbers differ from overflow in signed numbers?

13. If a computer is capable only of manipulating and storing integers, what difficulties present themselves? How are these difficulties overcome?

14. What are the three component parts of a floating-point number?

15. What is a biased exponent, and what efficiencies can it provide?

16. What is normalization and why is it necessary?

17. Why is there always some degree of error in floating-point arithmetic when performed by a binary digital computer?

18. How many bits long is a double-precision number under the IEEE-754 floating-point standard?

19. What is EBCDIC, and how is it related to BCD?

20. What is ASCII and how did it originate?

21. How many bits does a Unicode character require?

22. Why was Unicode created?

23. Why is non-return-to-zero coding avoided as a method for writing data to a magnetic disk?

24. Why is Manchester coding not a good choice for writing data to a magnetic disk?

25. Explain how run-length-limited encoding works.

26. How do cyclic redundancy checks work?

27. What is systematic error detection?

28. What is a Hamming code?

29. What is meant by Hamming distance and why is it important? What is meant by minimum Hamming distance?

30. How is the number of redundant bits necessary for code related to the number of data bits?

31. What is a burst error?

32. Name an error detection method that can compensate for burst errors.

EXERCISES

◆**1.** Perform the following base conversions using subtraction or division-remainder:

◆ **a)** $458_{10} =$ _____ $_3$

◆ **b)** $677_{10} =$ _____ $_5$

- **c)** $1518_{10} =$ _____ $_7$
- **d)** $4401_{10} =$ _____ $_9$

2. Perform the following base conversions using subtraction or division-remainder:

a) $588_{10} =$ _____ $_3$

b) $2254_{10} =$ _____ $_5$

c) $652_{10} =$ _____ $_7$

d) $3104_{10} =$ _____ $_9$

3. Convert the following decimal fractions to binary with a maximum of six places to the right of the binary point:

- **a)** 26.78125
- **b)** 194.03125
- **c)** 298.796875
- **d)** 16.1240234375

4. Convert the following decimal fractions to binary with a maximum of six places to the right of the binary point:

a) 25.84375

b) 57.55

c) 80.90625

d) 84.874023

5. Represent the following decimal numbers in binary using 8-bit signed magnitude, one's complement, and two's complement:

- **a)** 77
- **b)** −42

c) 119

d) −107

6. Using a "word" of 3 bits, list all of the possible signed binary numbers and their decimal equivalents that are representable in:

a) Signed magnitude

b) One's complement

c) Two's complement

7. Using a "word" of 4 bits, list all of the possible signed binary numbers and their decimal equivalents that are representable in:

a) Signed magnitude

b) One's complement

c) Two's complement

8. From the results of the previous two questions, generalize the range of values (in decimal) that can be represented in any given x number of bits using:

a) Signed magnitude

b) One's complement

c) Two's complement

9. Given a (very) tiny computer that has a word size of 6 bits, what are the smallest negative numbers and the largest positive numbers that this computer can represent in each of the following representations?

◆ **a)** One's complement

b) Two's complement

10. You have stumbled on an unknown civilization while sailing around the world. The people, who call themselves Zebronians, do math using 40 separate characters (probably because there are 40 stripes on a zebra). They would very much like to use computers, but would need a computer to do Zebronian math, which would mean a computer that could represent all 40 characters. You are a computer designer and decide to help them. You decide the best thing is to use BCZ, Binary-Coded Zebronian (which is like BCD except it codes Zebronian, not Decimal). How many bits will you need to represent each character if you want to use the minimum number of bits?

11. Perform the following binary multiplications:

◆ **a)**
$$\begin{array}{r} 1100 \\ \times\ 101 \\ \hline \end{array}$$

b)
$$\begin{array}{r} 10101 \\ \times\ 111 \\ \hline \end{array}$$

c)
$$\begin{array}{r} 11010 \\ \times\ 1100 \\ \hline \end{array}$$

12. Perform the following binary multiplications:

a)
$$\begin{array}{r} 1011 \\ \times\ 101 \\ \hline \end{array}$$

b)
$$\begin{array}{r} 10011 \\ \times\ 1011 \\ \hline \end{array}$$

c)
$$\begin{array}{r} 11010 \\ \times\ 1011 \\ \hline \end{array}$$

13. Perform the following binary divisions:

◆ **a)** 101101 ÷ 101

b) 10000001 ÷ 101

c) 1001010010 ÷ 1011

14. Perform the following binary divisions:

a) 11111101 ÷ 1011

b) 110010101 ÷ 1001

c) 1001111100 ÷ 1100

◆ **15.** Use the double-dabble method to convert 10212_3 directly to decimal. (Hint: you have to change the multiplier.)

16. Using signed-magnitude representation, complete the following operations:

$$+0 + (-0) =$$
$$(-0) + 0 =$$
$$0 + 0 =$$
$$(-0) + (-0) =$$

◆ **17.** Suppose a computer uses 4-bit one's complement numbers. Ignoring overflows, what value will be stored in the variable *j* after the following pseudocode routine terminates?

```
0 → j    // Store 0 in j.
-3 → k   // Store -3 in k.
while k ≠ 0
  j = j + 1
  k = k - 1
end while
```

18. If the floating-point number storage on a certain system has a sign bit, a 3-bit exponent, and a 4-bit significand:

a) What is the largest positive and the smallest negative number that can be stored on this system if the storage is normalized? (Assume no bits are implied, there is no biasing, exponents use two's complement notation, and exponents of all zeros and all ones are allowed.)

b) What bias should be used in the exponent if we prefer all exponents to be nonnegative? Why would you choose this bias?

◆ **19.** Using the model in the previous question, including your chosen bias, add the following floating-point numbers and express your answer using the same notation as the addend and augend:

0	1	1	1	1	0	0	0
0	1	0	1	1	0	0	1

Calculate the relative error, if any, in your answer to the previous question.

20. Assume we are using the simple model for floating-point representation as given in this book (the representation uses a 14-bit format, 5 bits for the exponent with a bias of 16, a normalized mantissa of 8 bits, and a single sign bit for the number):

a) Show how the computer would represent the numbers 100.0 and 0.25 using this floating-point format.

b) Show how the computer would add the two floating-point numbers in part a by changing one of the numbers so they are both expressed using the same power of 2.

c) Show how the computer would represent the sum in part b using the given floating-point representation. What decimal value for the sum is the computer actually storing? Explain.

21. What causes divide underflow and what can be done about it?

22. Why do we usually store floating-point numbers in normalized form? What is the advantage of using a bias as opposed to adding a sign bit to the exponent?

23. Let $a = 1.0 \times 2^9$, $b = -1.0 \times 2^9$ and $c = 1.0 \times 2^1$. Using the floating-point model described in the text (the representation uses a 14-bit format, 5 bits for the exponent with a bias of 16, a normalized mantissa of 8 bits, and a single sign bit for the number), perform the following calculations, paying close attention to the order of operations. What can you say about the algebraic properties of floating-point arithmetic in our finite model? Do you think this algebraic anomaly holds under multiplication as well as addition?

$$b + (a + c) =$$
$$(b + a) + c =$$

24. **a)** Given that the ASCII code for A is 1000001, what is the ASCII code for J?

b) Given that the EBCDIC code for A is 1100 0001, what is the EBCDIC code for J?

◆ **25.** Assume a 24-bit word on a computer. In these 24 bits, we wish to represent the value 295.

◆ **a)** If our computer uses even parity, how would the computer represent the decimal value 295?

◆ **b)** If our computer uses 8-bit ASCII and even parity, how would the computer represent the string 295?

◆ **c)** If our computer uses packed BCD, how would the computer represent the number +295?

26. Decode the following ASCII message, assuming 7-bit ASCII characters and no parity:

1001010 1001111 1001000 1001110 0100000 1000100 1000101

◆ **27.** Why would a system designer wish to make Unicode the default character set for their new system? What reason(s) could you give for not using Unicode as a default?

28. Write the 7-bit ASCII code for the character *4* using the following encoding:

a) Non-return-to-zero

b) Non-return-to-zero-invert

c) Manchester code

d) Frequency modulation

e) Modified frequency modulation

f) Run length limited

(Assume 1 is "high," and 0 is "low.")

29. Why is NRZ coding seldom used for recording data on magnetic media?

30. Assume we wish to create a code using 3 information bits, 1 parity bit (appended to the end of the information), and odd parity. List all legal code words in this code. What is the Hamming distance of your code?

31. Are the error-correcting Hamming codes systematic? Explain.

◆ **32.** Compute the Hamming distance of the following code:

0011010010111100

0000011110001111

0010010110101101

0001011010011110

33. Compute the Hamming distance of the following code:

0000000101111111

0000001010111111

0000100110111111

0000100011101111

0001000011110111

0010000011111011

0100000011111101

1000000011111110

34. Suppose we want an error-correcting code that will allow all single-bit errors to be corrected for memory words of length 10.

a) How many parity bits are necessary?

b) Assuming we are using the Hamming algorithm presented in this chapter to design our error-correcting code, find the code word to represent the 10-bit information word: 1001100110.

◆ **35.** Suppose we are working with an error-correcting code that will allow all single-bit errors to be corrected for memory words of length 7. We have already calculated that we need 4 check bits, and the length of all code words will be 11. Code words are created according to the Hamming algorithm presented in the text. We now receive the following code word:

1 0 1 0 1 0 1 1 1 1 0

Assuming even parity, is this a legal code word? If not, according to our error-correcting code, where is the error?

36. Repeat exercise 35 using the following code word:

0 1 1 1 1 0 1 0 1 0 1

37. Name two ways in which Reed-Soloman coding differs from Hamming coding.

38. When would you choose a CRC code over a Hamming code? A Hamming code over a CRC?

◆ **39.** Find the quotients and remainders for the following division problems modulo 2.

◆ **a)** $1010111_2 \div 1101_2$

◆ **b)** $1011111_2 \div 11101_2$

◆ **c)** $1011001101_2 \div 10101_2$

◆ **d)** $111010111_2 \div 10111_2$

40. Find the quotients and remainders for the following division problems modulo 2.

a) $1111010_2 \div 1011_2$

b) $1010101_2 \div 1100_2$

c) $1101101011_2 \div 10101_2$

d) $1111101011_2 \div 101101_2$

◆ **41.** Using the CRC polynomial 1011, compute the CRC code word for the information word, 1011001. Check the division performed at the receiver.

42. Using the CRC polynomial 1101, compute the CRC code word for the information word, 01001101. Check the division performed at the receiver.

***43.** Pick an architecture (such as 80486, Pentium, Pentium IV, SPARC, Alpha, or MIPS). Do research to find out how your architecture approaches the concepts introduced in this chapter. For example, what representation does it use for negative values? What character codes does it support?