Chapter One
DATA STORAGE

Chapter Summary

This chapter presents the rudiments of data storage within digital computers. It introduces the basics of digital circuitry and how a simple flip-flop can be used to store a single bit. It then discusses addressable memory cells and mass storage systems (magnetic disk, compact disks, and flash memory). Having established this background, the chapter discusses how information (text, numeric values, images, and sound) are encoded as bit patterns. There are two optional sections, first in a brief introduction to Python highlighting the means by which a programming languages hides the details of memory represention. The second delves more deeply into data storage topics by presenting the problems of overflow errors, truncation errors, error detection and correction techniques, and data compression.

Comments

1. Perhaps the most important comment I can make about this chapter (and the next one as well) is to explain its role in the chapters that follow. This involves the distinction between exposing students to a subject and requiring them to master the material—a distinction that is at the heart of the spirit in which the entire text was written. The intention of this chapter is to provide a realistic exposure to a very important area of computer science. It is not necessary for the students to master the material. All that is needed from this chapter in the remaining part of the book are the remnants that remain from a brief exposure to the issues of data storage. Even if the course you teach requires a mastery of these details or the development of manipulation skills, I encourage you to avoid emphasizing bit manipulations and representation conversions. In particular, I urge you to avoid becoming bogged down in the details of converting between base ten and binary notation. I can't think of anything that would be more boring for the students. (I apologize for stating my opinion.)

2. The "required" sections in this chapter cover the composition of main memory (as a background for machine architecture in chapter 2 and data structures in chapter 8), the physical issues of external data storage systems (in preparation for the subjects of file and database systems in chapter 9), and the rudiments of data encoding (that serves as a background for the subject of data types and high-level language declaration statements in chapter 6). The optional sections explore the issues of error handling, including transmission error detection and correction as well as the problem of truncation and overflow errors resulting from numeric coding systems.

3. As mentioned in the preface of the text, there are several themes that run throughout the text, one of which is the role of abstraction. I like to include this theme in my lecture in which I introduce flip-flops. I end up with both flip-flop diagrams from the text on the board, and I emphasize that they represent two ways of accomplishing the same task. I then draw a rectangle around each diagram and erase the circuits within the rectangles leaving only the inputs, outputs, and rectangles showing. At this point the two look identical. I think that this creates a strong visual image that drives home the distinction between an abstract tool's interface with the outside world and the internal details of the tool.

This is a specific example of teaching several topics at the same time—in this case, the concepts of abstraction and encapsulation are taught in the context of teaching digital circuits.

4. Don't forget about the circuits in Appendix B. I used to have students who continued to record an extra bit in the answer to a two's complement addition problem when a carry occurred—even though I had explained that all values in a two's complement system were represented with the same number of bits. Once I started presenting the addition circuit in Appendix B, this problem disappeared. It gave the students a concrete understanding that the carry is thrown away. (Of course, in a later course computing students will learn that it really isn't thrown away but saved as the "carry bit" for potential use in the future, but for now I ignore this.) I have also found that a good exercise is to ask students to extend the circuit in Figure B.3 so that it produces an additional output that indicates whether an overflow has occurred. For example, the output could be 1 in the case of an overflow and 0 otherwise.

5. For most students, seeing the reality of the things they are told is a meaningful experience. For this reason I often find it advantageous to demonstrate the distinction between numeric and character data using a spreadsheet. I like to show them how the manipulation of large numbers can lead to errors.

7. I have found that students respond well to hearing about CD and DVD storage systems, how sound is encoded, and image representation systems such as GIF and JPEG. I have often used these topics as a way of getting non-majors interested in technical issues.

8. For students not majoring in computer science, topics such as two's complement and floating-point notation can get a bit dry. The main point for them to understand is that when information is encoded, some information usually gets lost. This point can be made just as well using audio and video, which are contexts that seem to be more interesting to the non-majors.

Answers to Chapter Review Problems

1. With a 1 on the upper input and a 0 on the lower input, all circuits will produce an output 0. If instead a 0 is on the upper input and 1 is on the lower input, circuits b and c will produce an output 1, and circuit a will still produce a 0.

2. a. The entire circuit is equivalent to a single AND gate.

   b. The entire circuit is equivalent to an Exclusive OR gate.

3. a. After the third pulse, this circuit will produce an output of 1 and 1. After the fourth pulse, both flip-flops are flipped back to a 0 state so the circuit will again produce an output of 0 and 0. It is interesting to note that this circuitry forms a binary counter that will repeatedly count from 00 to 11. Thus, this circuit forms an abstract tool that can be used as a building block in other circuits. Additional flip-flops can be added to count through a larger range of numbers.

   b. A 1 will be sent on Output B on the 2^{nd}, 6^{th}, 10^{th} … pulses of the clock. Likewise, a 1 will be sent to Output C on the 3^{rd}, 7^{th}, 11^{th} … pulses of the clock. A 1 will not be sent to any output on the 4^{th}, 8^{th}, 12^{th} … pulses of the clock. As we move forward into the next chapter, a circuit similar to this can be used to drive the machine cycle (composed of fetch, decode, and execute). Output A would be connected to the input that activates the fetch circuitry. Likewise Output B and Output C would be connected to the decode and execute circuits respectively.

4. This is a flip-flop that is triggered by 0s rather than 1s. That is, temporarily changing the upper input to 0 will establish an output of 1, whereas temporarily changing the lower input to 0 will establish an output of 0. To obtain an equivalent circuit using NAND gates, simply replace each AND-NOT gate pair with a NAND gate.

5.

Address	Contents
00	02
01	53
02	01
03	53

6. 256 using two hexadecimal digits (16 bits) , 65536 using four hexadecimal digits (32 bits).

7. a. 11001101 b. 01100111 c. 10011010 d. 11111111 e. 00010000

8. a. 1 b. 1 c. 0 d. 0

9. a. A0A b. C7B c. 0BE

10. The image consists of 1024 x 1024 = 1,048,576 pixels and therefore 3 x 1,048,576 = 3,145,728 bytes, or about 3MB. This means that about 86 images could be stored in the 256MB camera storage system. (By comparing this to actual camera storage capacities, students can gain an appreciation for the benefits of image compression techniques. Using them, a typical 256MB storage system can hold as many as 300 images.)

11. 786,432. (Each pixel would require one memory cell.)

12. Data retrieval from main memory is much faster than from disk storage. Also data in main memory can be referenced in byte-sized units rather than in large blocks. On the other hand, disk storage systems have a larger capacity than main memory and the data stored on disk is less volatile than that stored in main memory.

13. There are 70GB of material on the hard-disk drive. Each CD can hold no more than 700MB. Thus, it will require at least 100 CDs to store all the material. That does not seem practical to me. On the other hand, DVDs have capacities of about 4.7GB, meaning that only about 15 DVDs would be required. This may still be impractical, but its a big improvement over CDs. (The real point of this problem is to get students to think about storage capacities in a meaningful way.)

14. There would be about 5,000 characters on the page requiring two bytes for each two byte Unicode character. So the page would require about 10,000 bytes or 10 sectors of size 1024 bytes.

15. The novel would require about 1.4MB using ASCII and about 2.8MB if two byte Unicode characters were used.

16. The latency time of a disk spinning at 3600 revolutions per minute is only 0.00833 seconds.

17. About 11.4 milliseconds.

18. About 7 years!

19. What does it say?

20. hex notation

21. a.

D	o	e	s		1	0	0
01000010	01101111	01100101	01110011	00100000	00110001	00110000	00110000
	/		5		=		2
00100000	00101111	00100000	01101010	00100000	00111101	00100000	00110010
0	?						
00110000	00111111						

b.

T	h	e		t	o	t	a
01010100	01101000	01100101	00100000	01110100	01101111	01110100	01100001
l		c	o	s	t		i
01101100	00100000	01100011	01101111	01110011	01110100	00100000	01101001
s		$	7	.	2	5	.

01110011 00100000 00100100 00110111 00101110 00110010 00110101 00101110

22. a. 42 6F 65 73 20 31 30 30 20 2F 20 35 20 3D 20 32 30 3F

b. 54 68 65 20 74 6F 74 61 6C 20 63 6F 74 20 69 73 20 24 37 2E 32 35 2E

23. 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 10000, 10001, 10010

24. a. 00110010 00110011 b. 10111

25. They are the powers of two. 1 10 100 1000 10000 100000

26. a. 15 b. 1 c. 21 d. 16 e. 19 f. 0 g. 5 h. 9 i. 17 j. 25 k. 26 l. 27

27. a. 111 b. 1011 c. 10000 d. 10001 e. 11111

28. a. 1 b. 5 c. -3 d. -1 e. 15

29. a. 100 b. 111 c. 010 d. 011 e. 110

30. a. 15 b. -12 c. 12 d. -16 e. -10

31. a. 0001101 b. 1110011 c. 1111111 d. 0000000 e. 0010000

32. a. 01101 b. 00000 c. 10000 (incorrect) d. 10001 e. 11110

f. 10011 (incorrect) g. 11110 h. 01101 i. 10000 (incorrect) j. 11111

33. a.

```
  5              00101
+ 1   becomes  + 00001
                 00110  which represents 6
```

b.

```
  5              00101                      00101
- 1   becomes  - 00001  which converts to  + 11111
                                            00100   which represents 4
```

c.

```
 12              01100                      01100
- 5   becomes  - 00101  which converts to  + 11011
                                            00111   which represents 7
```

d.

```
  8              01000                      01000
- 7   becomes  - 00111  which converts to  + 11001
                                            00001  which represents 1
```

e.

```
 12             01100
+ 5  becomes  + 00101
               10001   which represents -15 (overflow)
```

f.

```
  5              00101                      00101
- 11  becomes  - 01011  which converts to  + 10101
                                            11010  which represents -6
```

34. a. 3 3/4 b. 4 5/16 c. 13/16 d. 1 e. 2 1/4

35. a. 101.11 b. 1111.1111 c. 101.011 d. 1.01 e. 110.101

36. a. 1 1/8 b. -1/2 c. -3/16 d. 9/32

37. a. 11111111 b. 01001000 c. 11101111

d. 00101110 e. 00011111 (truncation)

38. 00111100, 01000110, and 01010011

39. The best approximation of the square root of 2 is 1 3/8 represented as 01011011. The square of this value when represented in floating-point format is 01011111, which is the representation of 1 7/8.

40. The value one-eighth, which would be represented as 00101000.

41. Since the value one-tenth cannot be represented accurately, such recordings would suffer from truncation errors.

42. a. The value is either eleven or negative five.

b. A value represented in two's complement notation can be changed to excess notation by changing the high-order bit, and vice versa.

43. The value is two; the patterns are excess, floating-point , and two's complement, respectively.

44. b would require too many significant digits. c would require too large of an exponent. d would require too many significant digits.

45. When using binary notation, the largest value that could be represented would change from 15 to 63. When using two's complement notation the largest value that could be represented would change from 7 to 31.

46. 4FFFFF

47. 1123221343435

48. yyxy xx yyxy xyx xx xyx

49. Starting with the first entries, they would be *x*, *y*, space, *xxy*, *yyx*, and *xxyy*.

50. Not a chance. MPEG requires transfer rates of 40 Mbps.

51. a.

D	o	e	s		1	0	0
001000010	001101111	001100101	101110011	100100000	100110001	000110000	000110000
	/		5		=		2
100100000	100101111	100100000	000110101	100100000	100111101	100100000	100110010
0	?						
000110000	000111111						

b.

T	h	e		t	o	t	a
101010100	101101000	001100101	100100000	001110100	001101111	001110100	101100001
l		c	o	s	t		i
101101100	100100000	001100011	001101111	101110011	001110100	100100000	001101001
s		$	7	.	2	5	.
101110011	100100000	100100100	100110111	000101110	100110010	000110101	000101110

52. The underlined strings definitely contain errors.

11001 <u>11011</u> 10110 <u>00000</u> 11111 <u>10001</u> 10101 00100 01110

53. The code would have a Hamming distance of 3. Thus, by using it, one could detect up to 2 errors per character and correct up to 1 error per character.

54. a. HE b. FED c. DEAD d. CABBAGE e. CAFE
55. Answers will vary as the exchange rates change daily.
56. Answers will vary depending on the currency selected.
57. This is an interactive exercise, results will depend on the browser.
58. A change to the dollar amount would need to be made in each of the expressions.
59.

```
no_B = 123234234
no_KB = no_B / 1024
no_MB = no_KB / 1024
no_GB = no_MB / 1024
no_TB = no_TB / 1024
print(str(no_B) + ' B')
print(str(no_KB) + ' KB')
print(str(no_MB) + ' MB')
print(str(no_GB) + ' GB')
print(str(no_TB) + ' TB')
print(str(no_KB) + ' KB')
print(str(no_KB) + ' KB')


no_TB = 1.123
no_GB = no_TB * 1024
no_MB = no_GB * 1024
no_KB = no_MB * 1024
no_B  = no_KB * 1024
print(str(no_B) + ' B')
print(str(no_KB) + ' KB')
print(str(no_MB) + ' MB')
print(str(no_GB) + ' GB')
print(str(no_TB) + ' TB')
print(str(no_KB) + ' KB')
print(str(no_KB) + ' KB')
```

60.

```
mins = 50
secs = 23
tot_secs = mins * 60 + secs
samples_per_sec = 44100
bytes_per_sample = 2
tot_bytes = tot_secs * samples_per_sec * bytes_per_sample
print(tot_bytes)
```

61. '**' should be '*' and 'PRINT' should be 'print'