Xerox Corporation v. Google Inc. et al Doc. 163 Att. 10

EXHIBIT 10

Dockets.Justia.com

EDITION 10

computer science

AN OVERVIEW

J. Glenn Brookshear

Publisher	Greg Tobin
Executive Editor	Michael Hirsch
Editorial Assistant	Stephanie Sellinger
Associate Managing Editor	Jeffrey Holcomb
Cover Design	Joyce Cosentino Wells
Photo Research	Beth Anderson
Digital Assets Manager	Marianne Groth
Senior Media Producer	Bethany Tidd
Marketing Manager	Erin Davis
Senior Author Support/Technology Specialist	Joe Vetere
Senior Manufacturing Buyer	Carol Melville
Production Coordination	Shelley Creager, Aptara, Inc.
Text Design, Composition, and Illustrations	Aptara, Inc.

About the Cover: The three-dimensional image of an airplane on the cover is not a photograph, but rather a computer-generated image, created using technology similar to that used for the animated movies *Toy Story* and *Shrek 2* referenced in Chapter 10.

Photo Credits

Cover Images: 3D airplane © Shutterstock/Computer Earth; Landscape photograph © Shutterstock/Elena Elisseeva

Figure 0.3, "An abacus (photography by Wayne Chandler)" © Glenn Brookshear
Figure 0.4, "The Mark I computer" © Addison-Wesley
Figure 10.1, "A 'photograph' of a virtual world produced using 3D graphics (from *Toy Story* by Walt Disney Pictures/Pixar Animation Studios)" © Corbis/Sygma.
Figure 10.6, "A scene from *Shrek 2* by Dreamworks SKG" © Dreamworks/The Kobal Collection.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed in initial caps or all caps.

Library of Congress Cataloging-in-Publication Data

Brookshear, J. Glenn.
Computer science : an overview / J. Glenn Brookshear.—10th ed.
p. cm.
Includes index.
ISBN-13: 978-0-321-52403-4
ISBN-10: 0-321-52403-9
1. Computer science. I. Title.
QA76.B743 2007
004—dc22

2007046365

Copyright © 2009 Pearson Education, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America. For information on obtaining permission for use of material in this work, please submit a written request to Pearson Education, Inc., Rights and Contracts Department, 501 Boylston Street, Suite 900, Boston, MA 02116, fax (617) 671-3447, or online at http://www.pearsoned.com/legal/permissions.htm.

ISBN-13: 978-0-321-52403-4
ISBN-10: 0-321-52403-9
3 4 5 6 7 8 9 10—CW—12 11 10

1.4 Representing Information as Bit Patterns

Having considered techniques for storing bits, we now consider how information can be encoded as bit patterns. Our study focuses on popular methods for encoding text, numerical data, images, and sound. Each of these systems has repercussions that are often visible to a typical computer user. Our goal is to understand enough about these techniques so that we can recognize their consequences for what they are.

Representing Text

Information in the form of text is normally represented by means of a code in which each of the different symbols in the text (such as the letters of the alphabet and punctuation marks) is assigned a unique bit pattern. The text is then represented as a long string of bits in which the successive patterns represent the successive symbols in the original text.

In the 1940s and 1950s, many such codes were designed and used in connection with different pieces of equipment, producing a corresponding proliferation of communication problems. To alleviate this situation, the **American National Standards Institute (ANSI,** pronounced "AN-see") adopted the **American Standard Code for Information Interchange (ASCII,** pronounced "AS-kee"). This code uses bit patterns of length seven to represent the upper- and lowercase letters of the English alphabet, punctuation symbols, the digits 0 through 9, and certain control information such as line feeds, carriage returns, and tabs. Today, ASCII is often extended to an eight-bit-per-symbol format by adding a 0 at the most significant end of each of the seven-bit patterns. This technique not only produces a code in which each pattern fits conveniently into a typical byte-size memory cell but also provides 128 additional bit patterns (those obtained by assigning the extra bit the value 1) that can represent symbols excluded in the original ASCII. Unfortunately, because vendors tend to use their own interpretations for these extra patterns, data in which these patterns appear often are not easily transported from one vendor's application to another.

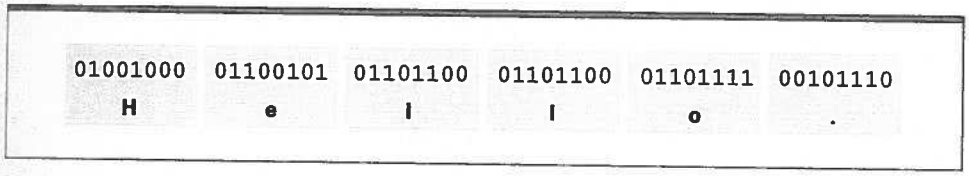
A portion of ASCII in its eight-bit-per-symbol format is shown in Appendix A. By referring to this appendix, we can decode the bit pattern

```
01001000  01100101  01101100  01101100  01101111
00101110
```

as the message "Hello." as demonstrated in Figure 1.13.

Although ACSII has been the dominant code for many years, other more extensive codes, capable of representing documents in a variety of languages, are now competing for popularity. One of these, **Unicode,** was developed through the cooperation of several of the leading manufacturers of hardware and software and is rapidly gaining support in the computing community. This code uses a unique pattern of 16 bits to represent each symbol. As a result, Unicode

Figure 1.13 The message "Hello." in ASCII



consists of 65,536 different bit patterns—enough to allow text written in such languages as Chinese, Japanese, and Hebrew to be represented.

Standards for a code that could compete with Unicode have been developed by the **International Organization for Standardization** (also known as **ISO,** in reference to the Greek word *isos,* meaning equal). Using patterns of 32 bits, this encoding system has the potential of representing billions of different symbols.

A file consisting of a long sequence of symbols encoded using ASCII or Unicode is often called a **text file.** It is important to distinguish between simple text files that are manipulated by utility programs called **text editors** (or often simply editors) and the more elaborate files produced by **word processors** such as Microsoft's Word. Both consist of textual material. However, a text file contains only a character-by-character encoding of the text, whereas a file produced by a word processor contains numerous proprietary codes representing changes in fonts, alignment information, etc. Moreover, word processors may even use proprietary codes rather than a standard such as ASCII or Unicode for representing the text itself.

Representing Numeric Values

Storing information in terms of encoded characters is inefficient when the information being recorded is purely numeric. To see why, consider the problem of storing the value 25. If we insist on storing it as encoded symbols in ASCII using one byte per symbol, we need a total of 16 bits. Moreover, the largest number we could store using 16 bits is 99. However, as we will shortly see, by using **binary notation** we can store any integer in the range from 0 to 65535 in these 16 bits. Thus, binary notation (or variations of it) is used extensively for encoded numeric data for computer storage.

Binary notation is a way of representing numeric values using only the digits 0 and 1 rather than the digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 as in the traditional decimal, or base ten, system. We will study the binary system more thoroughly in Section 1.5. For now, all we need is an elementary understanding of the system. For this purpose consider an old-fashioned car odometer whose display wheels contain only the digits 0 and 1 rather than the traditional digits 0 through 9. The odometer starts with a reading of all 0s, and as the car is driven for the first few miles, the rightmost wheel rotates from a 0 to a 1. Then, as that 1 rotates back to a 0, it causes a 1 to appear to its left, producing the pattern 10. The 0 on the right then rotates to a 1, producing 11. Now the rightmost wheel rotates from 1 back to

The American National Standards Institute

The American National Standards Institute (ANSI) was founded in 1918 by a small consortium of engineering societies and government agencies as a nonprofit federation to coordinate the development of voluntary standards in the private sector. Today, ANSI membership includes more than 1300 businesses, professional organizations, trade associations, and government agencies. ANSI is headquartered in New York and represents the United States as a member body in the ISO. The website for the American National Standards Institute is at http://www.ansi.org.

Similar organizations in other countries include Standards Australia (Australia), Standards Council of Canada (Canada), China State Bureau of Quality and Technical Supervision (China), Deutsches Institut für Normung (Germany), Japanese Industrial Standards Committee (Japan), Dirección General de Normas (Mexico), State Committee of the Russian Federation for Standardization and Metrology (Russia), Swiss Association for Standardization (Switzerland), and British Standards Institution (United Kingdom).

0, causing the 1 to its left to rotate to a 0 as well. This in turn causes another 1 to appear in the third column, producing the pattern 100. In short, as we drive the car we see the following sequence of odometer readings:

```
0000
0001
0010
0011
0100
0101
0110
0111
1000
```

This sequence consists of the binary representations of the integers zero through eight. Although tedious, we could extend this counting technique to discover that the bit pattern consisting of sixteen 1s represents the value 65535, which confirms our claim that any integer in the range from 0 to 65535 can be encoded using 16 bits.

Due to this efficiency, it is common to store numeric information in a form of binary notation rather than in encoded symbols. We say "a form of binary notation" because the straightforward binary system just described is only the basis for several numeric storage techniques used within machines. Some of these variations of the binary system are discussed later in this chapter. For now, we merely note that a system called **two's complement** notation (see Section 1.6) is common for storing whole numbers because it provides a convenient method for representing negative numbers as well as positive. For representing numbers with fractional parts such as 4½ or ¾, another technique, called **floating-point** notation (see Section 1.7), is used.

appendix

A

ASCII

The following is a partial listing of ASCII code, in which each bit pattern has been extended with a 0 on its left to produce the eight-bit pattern commonly used today.

Symbol	ASCII
line feed	00001010
carriage return	00001101
space	00100000
!	00100001
"	00100010
#	00100011
$	00100100
%	00100101
&	00100110
'	00100111
(	00101000
)	00101001
*	00101010
+	00101011
'	00101100
-	00101101
.	00101110
/	00101111
0	00110000
1	00110001
2	00110010
3	00110011
4	00110100
5	00110101
6	00110110
7	00110111
8	00111000
9	00111001
:	00111010
;	00111011
<	00111100
=	00111101

Symbol	ASCII
>	00111110
?	00111111
@	01000000
A	01000001
B	01000010
C	01000011
D	01000100
E	01000101
F	01000110
G	01000111
H	01001000
I	01001001
J	01001010
K	01001011
L	01001100
M	01001101
N	01001110
O	01001111
P	01010000
Q	01010001
R	01010010
S	01010011
T	01010100
U	01010101
V	01010110
W	01010111
X	01011000
Y	01011001
Z	01011010
[	01011011
\	01011100
]	01011101

Symbol	ASCII
^	01011110
_	01011111
a	01100001
b	01100010
c	01100011
d	01100100
e	01100101
f	01100110
g	01100111
h	01101000
i	01101001
j	01101010
k	01101011
l	01101100
m	01101101
n	01101110
o	01101111
p	01110000
q	01110001
r	01110010
s	01110011
t	01110100
u	01110101
v	01110110
w	01110111
x	01111000
y	01111001
z	01111010
{	01111011
}	01111101