Supplement to

Logic and Computer Design Fundamentals
3rd Edition[1]

UNICODE

Selected topics not covered in the third edition of *Logic and Computer Design Fundamentals* are provided here for optional coverage and for self-study if desired. This material fits well with the desired coverage in some programs but not may not fit within others due to time constraints or local preferences. This supplement is referenced in Chapter 1 as a part of the coverage of Alphanumeric Codes.

Unicode is a standard for 16-bit alphanumeric codes. It is incorporated in the standard ISO/IEC (International Organization for Standardization/International Electrotechnical Commission) 10646 and is sometimes referred to as Unicode/10646. Since 16 bits provide 65,536 code words, Unicode has the capacity to represent the symbols and ideographs of the world's languages. A character code in Unicode is represented by four hexadecimal digits. Standard ASCII codes with $(00)_{16}$ appended to the left (excluding the control codes) constitute the first 95 characters in Unicode. The codes used are from $(0000)_{16}$ through $(007F)_{16}$ (excluding the control codes). Table 1 gives the first 191 character codes in Unicode. The codes are in hexadecimal, with the three most significant digits determining the table column and the least significant digit the table row. Note that different graphic characters may be associated with a given character code, such as for the dollar sign ($) and tilde (~). The codes from $(00A0)_{16}$ through $(00FF)_{16}$ are referred to as Latin 1. These codes provide additional letters used in the major languages of Europe, based on the Latin alphabet. Carrying on the tradition of ASCII, a miscellaneous set of mathematical signs and punctuation is included. Latin 1 codes are based on standard ISO 8859-1. In Unicode, the mathematical symbols, signs, and punctuation not included in ASCII and Latin 1 are included at a higher range.

Figure 1 gives the four major zones for the assignment of Unicode code words. The hexadecimal digit pairs shown are the leftmost two digits of the code

[1] © Pearson Education 2004. All rights reserved.

TABLE 1
First 256 Codes for Unicode[a]

	Control		ASCII						Control		Latin 1					
	000	001	002	003	004	005	006	007	008	009	00A	00B	00C	00D	00E	00F
0	CTRL	CTRL	SPACE	0	@	P	`	p	CTRL	CTRL	NB SP	°	À	Ð	à	D
1	CTRL	CTRL	!	1	A	Q	a	q	CTRL	CTRL	¡	±	Á	Ñ	á	ñ
2	CTRL	CTRL	"	2	B	R	b	r	CTRL	CTRL	¢	2	Â	Ò	â	ò
3	CTRL	CTRL	#	3	C	S	c	s	CTRL	CTRL	£	3	Ã	Ó	ã	ó
4	CTRL	CTRL	$\|$	4	D	T	d	t	CTRL	CTRL	¤	´	Ä	Ô	ä	ô
5	CTRL	CTRL	%	5	E	U	e	u	CTRL	CTRL	¥|¥	µ	Å	Õ	å	õ
6	CTRL	CTRL	&	6	F	V	f	v	CTRL	CTRL	¦	¶	Æ	Ö	æ	ö
7	CTRL	CTRL	'	7	G	W	g	w	CTRL	CTRL	§	·	Ç	×	ç	÷
8	CTRL	CTRL	(	8	H	X	h	x	CTRL	CTRL	¨	¸	È	Ø	è	ø
9	CTRL	CTRL	)	9	I	Y	i	y	CTRL	CTRL	©		É	Ù	é	ù
A	CTRL	CTRL	*	:	J	Z	j	z	CTRL	CTRL	ª	º	Ê	Ú	ê	ú
B	CTRL	CTRL	+	;	K	[	k	{	CTRL	CTRL	«	»	Ë	Û	ë	û
C	CTRL	CTRL	,	<	L	\	l	|	CTRL	CTRL	¬	¼|1/4	Ì	Ü	ì	ü
D	CTRL	CTRL	-	=	M	]	m	}	CTRL	CTRL	-	½|1/2	Í	Ý	í	ý
E	CTRL	CTRL	.	>	N	^	n	~|¯	CTRL	CTRL	®	¾|3/4	Î	'b	î	'b
F	CTRL	CTRL	/	?	O	_	o	CTRL	CTRL	CTRL	¯	¿	Ï	ß	ï	ÿ

[a] Unicode, Inc., The Unicode Standard: Worldwide Character Encoding, Version 1.0, Volume 1, © 1990, 1991 by Unicode, Inc. Reprinted by permission of Addison-Wesley Publishing Company, Inc.

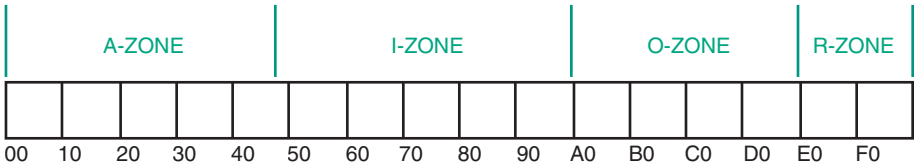

FIGURE 1
Major Zones for Unicode Code Assignment

words. Each block in the figure represents 4,096 codes. The A-Zone contains codes for alphabets, syllables, and symbols. The I-Zone contains codes for ideographs. An ideographic character stands for a word or inseparable grammatical unit, rather than a sound. Chinese script used in various languages is an example of ideographic characters. Since the ideographs represent words rather than characters, there are large numbers of them. The O-Zone is currently vacant, but is likely to be used for ideographs in the future.

The R-Zone is for restricted use. It is further broken down into the Private Use Area, the Compatibility Zone, and Special Codes. FFFE and FFFF are not character codes and are specifically excluded from Unicode. The Private Use Area is available to those needing special characters for their application programs. For example, icons used in menus could be specified by character codes in this range. The Compatibility Zone contains characters that are mapped to other areas in the overall code space. The characters available in this special area are in widespread use, but are not directly compatible with the way in which Unicode encoding deals with character representation, so they could not be included directly in other areas.

Although Unicode characters are defined as 16 bits, they can be implemented in computers by two bytes. A computer *word* is composed of multiple bytes. Suppose that we consider a 16-bit word consisting of two bytes. If byte 0 is on the right of the word (the little or least significant end) and byte 1 on the left, the computer is said to be *little-endian*. If byte 0 is on the left of the word (the big or most significant end) and byte 1 on the right, the computer is said to be *big-endian*. Now suppose that a string of $2n$ bytes representing n Unicode characters is transferred from a little-endian computer to a big-endian computer. Then the byte order in a 16-bit word is interchanged, thereby garbling the characters. This byte-ordering problem for Unicode is illustrated in Figure 2. If the code $(FEFF)_{16,}$ the nonprinting Unicode value for BYTE ORDER MARK, is included at the beginning of the original string, then it will appear as the invalid code

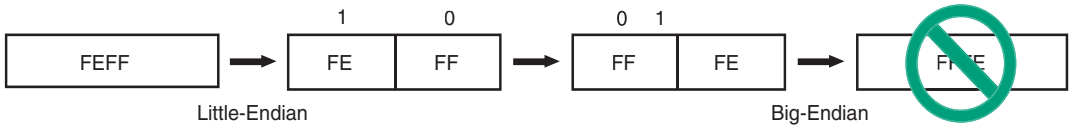

FIGURE 2
Byte-Ordering Problem for Unicode Codes

$(FFFE)_{16}$, due to the pairs of bytes being reversed. This signals that all pairs are in the wrong order and must be swapped before the 16-bit words are interpreted as characters. Thus, by placing Unicode $(FEFF)_{16}$ at the beginning of a character string, an application can determine whether the bytes need to be swapped before interpretation. A similar situation occurs in going from a big-endian to a little-endian computer.

• EXAMPLE 1 Unicode Application

Find the word represented by the Unicode string FFFE, 4300, 6F00, 6400, 6500.

Since the string begins with FFFE, the bytes are in the wrong order and need to be swapped before being interpreted as Unicode characters. Swapping the bytes gives the Unicode string

FEFF, 0043, 006F, 0064, 0065

which can be decoded as the word "Code" by using Table 1. •

References

1. The Unicode Consortium. *The Unicode Standard: Worldwide Character Encoding*, version 1.0, vol. 1. Reading, MA: Addison-Wesley, 1991.
2. Bettels, J., and Bishop, E. A. "Unicode: A Universal Character Code," *Digital Technical Journal*, vol. 5, no. 3 (Summer 1993), pp. 21-31.
3. Williams, M. R. *A History of Computing Technology.* Englewood Cliffs, NJ: Prentice Hall, 1985.

Problems

The plus (+) indicates a more advanced problem.

1. Write your full name in Unicode. Include a space between names and a period after the middle initial.

2. Decode the following Unicode given in hexadecimal:
   FEFF 0031 00F7 0034 003D 00BC 0020 0069 0073 0020 0061 0020 0046 0052 0041 0043 0054 0049 004F 004E 002E

3. +Decode the following Unicode given in binary. The result is a question in Spanish.
   1111111111111110 1011111100000000 0100001100000000
   1111001100000000 0110110100000000 0110111100000000
   0011111100000000