ASCII – American Standard Code for Information Interchange.

Since computers can do amazing things it is hard to realize that at its heart, a computer can only understand basic numbers. Most programs have many layers that take normal human readable text with colors, carriage returns, tabs, etc. and convert them into a simple string of ascii numbers. Think of it as a big decoder ring.

After the table below is a brief discussion on encoding as well as byte ordering.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	000	NUL	32	20	040	Space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	“	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	‘	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(	72	48	110	H	104	68	150	H
9	9	011	TAB	41	29	051	)	73	49	111	I	105	69	151	I
10	A	012	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	B	013	VT	43	2B	053	+	75	4B	113	K	107	6B	153	K
12	C	014	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	D	015	CR	45	2D	055	-	77	4D	115	M	109	6D	155	M
14	E	016	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	F	017	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC	59	3B	073	;	91	5B	133	[	123	7B	173	{
29	1C	034	FS	60	3C	074	<	92	5C	134	\	124	7C	174	|
29	1D	035	GS	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	US	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

Fun with Notepad

Open Notepad.exe on your computer reproduce the ASCII codes above by holding ALT+### in which ### is the Dec (decimal) number above. For example, in notepad, if you press and hold the ALT key and then type in 065 (i.e, ALT+065) you will see A.

Enter the following codes to reveal the secret message from a popular movie (do not type the spaces or + signs; you also cannot cut and paste—no pain no gain, you have to type them in):

ALT+66 ALT+101 ALT+32 ALT+115 ALT+117 ALT+114 ALT+101 ALT+32 ALT+116 ALT+111 ALT+32

ALT+100 ALT+114 ALT+105 ALT+110 ALT+107 ALT+32 ALT+121 ALT+111 ALT+117 ALT+114 ALT+32

ALT+79 ALT+118 ALT+97 ALT+108 ALT+116 ALT+105 ALT+110 ALT+101 ALT+33

Same Message Different Code

You can also write messages using Decimal (Dec), Hexa-decimal (Hex), or Octal (Oct). Let's look at a message in each of the codes:

Char: We Focus On The Root Cause

Dec: 87 101 32 70 111 99 117 115 32 79 110 32 84 104 101 32 82 111 111 116 32 67 97 117 115 101

Hex: 57 65 20 46 6F 63 75 73 20 4F 6E 20 54 68 65 20 52 6F 6F 74 20 43 61 75 73 65

Oct: 127 145 040 106 157 143 165 163 040 117 156 040 124 150 145 040 122 157 157 164 040 103 141 165 163 145

Aren't you glad we can type characters and do not have to type just numbers to create messages in a computer!

Peek at Saved File

Let's look at a saved file and verify what we have learned. Open Notepad.exe and cut/paste the following text into a file and save it to your desktop. Make sure you get the entire area that is shaded.

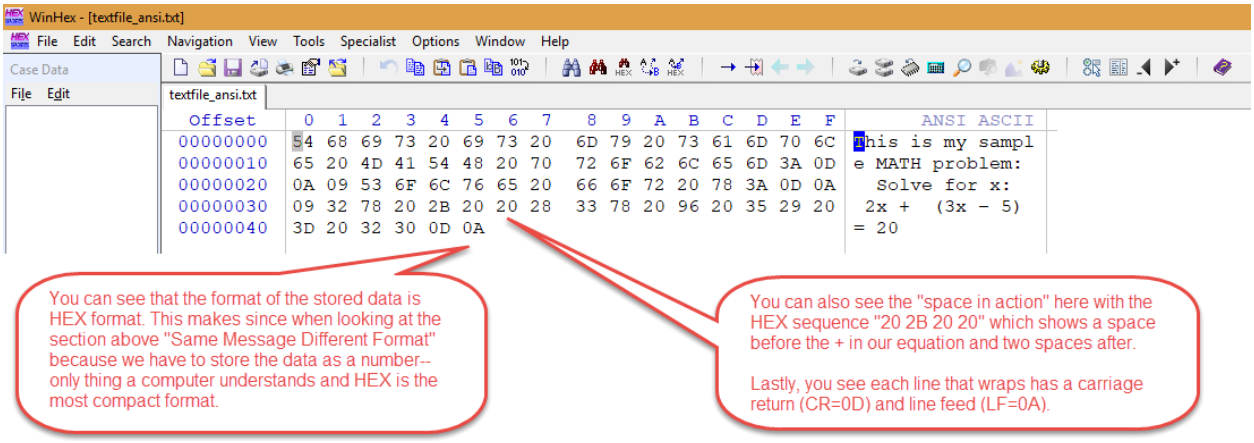
This is my sample MATH problem:

Solve for x:

2x + (3x – 5) = 20

Then open with your favorite HEX editor such as WinHex and you will see the following:

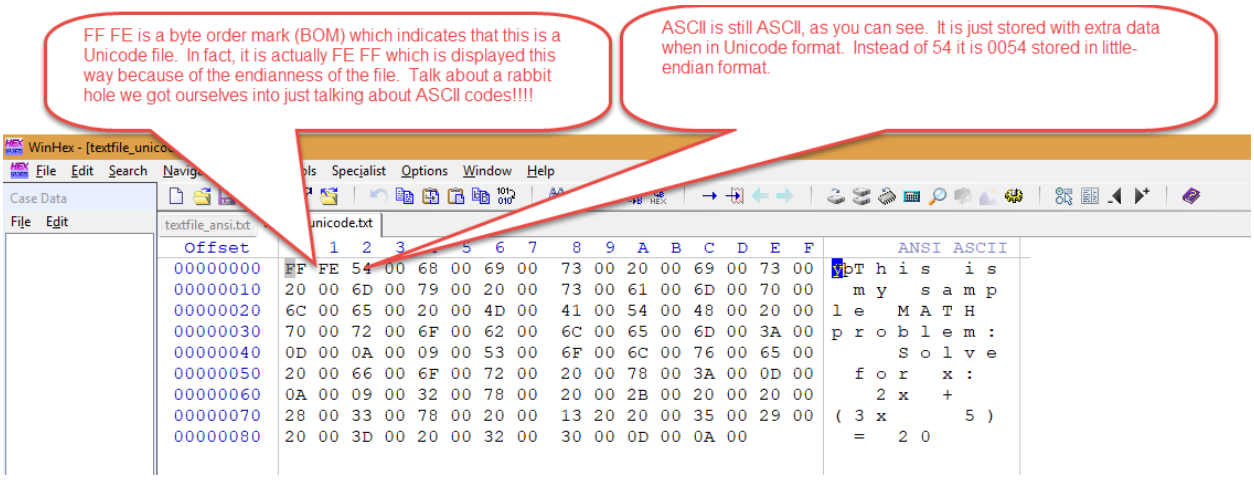
ANSI Encoding



What about Unicode?

Well, you will notice that the ASCII table only shows the English alphabet. When you saved the file with ANSI encoding you got the file above. ANSI (American National Standards Institute) is a format good for America but not for the rest of the world. This is nice for those in America, however, it presents limitations when dealing with other languages. Unicode was developed to give more bytes of data to describe each character in order to handle all of the letters of each language as well as punctuation markings. Let's take a peek at the file saved in Unicode format instead of ANSI.

Unicode Format (default is little-endian byte order)



Endianness

The Endianness referred to in the previous picture is a directive that specifies how multi-byte characters are transmitted or read. What?

Using our ASCII table we can see that a capital T in ANSI format is 54. A capital T in Unicode format is 0054. There is an extra byte (00) which is used to store the character. The problem is that not all

hardware and software reads data the same. Some read the Unicode characters in big-endian format which means the most-significant byte first.

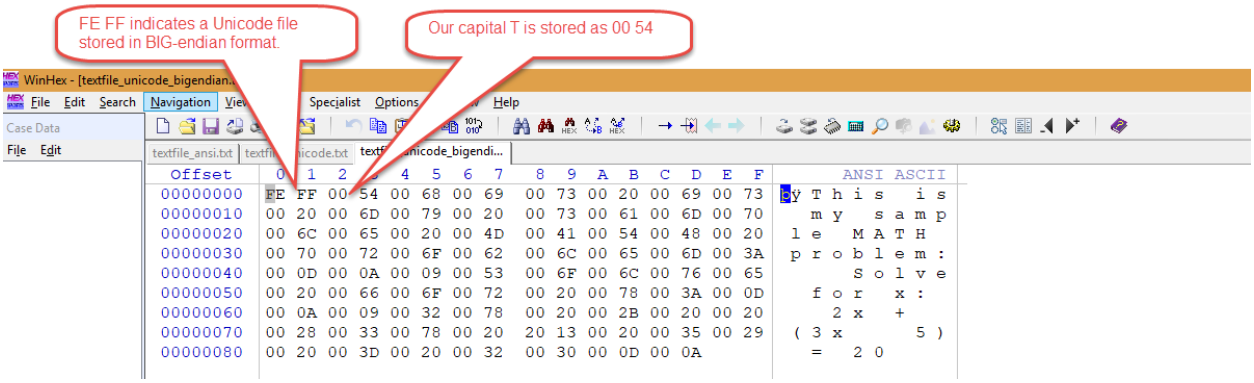
This means it expects it will see 00 then 54. If it is reading in little-endian format, it expects the least significant byte first. It expects 54 then 00.

Little-endian: least significant byte first in a multi-byte character.

Big-endian: most significant byte first in a multi-byte character.

Unicode Format – Big Endian Byte Order

Let's peek at our exact same file saved as Unicode Big Endian format:



Notice in the previous picture that the capital T is now stored as 0054 and not 5400.

How does a program figure this out?

When notepad opens and starts to read the file it looks at the first bytes. When it sees FE or FF then it knows it is not an ANSI encoded file. How? FE and FF are not in the ASCII table. So then it reads the next bytes which is either FE or FF. If notepad reads FEFF then it jumps to the function to read a Unicode Big-endian file. If it reads FFFE then it jumps to the function to read a Unicode Little-endian file.

Unicode Transformation Format (UTF)

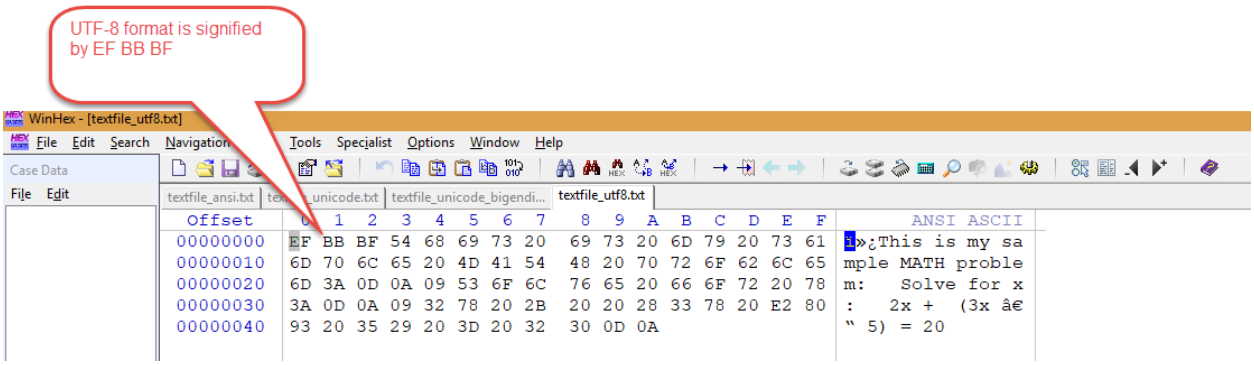
In notepad you can save a file in UTF-8 format. Although UTF starts to get more complicated, in a nutshell, the number after UTF indicates the number of bytes used to store one character.

- UTF-8 uses 1 byte
- UTF-16 uses 2-4 bytes
- UTF-32 uses 4 bytes.

The variable nature of UTF-16 has to do mainly with if the character is in the range 0-127 or 128-255. They are still all Unicode.

UTF-8 Format

Let's look at one last picture of a UTF-8 version of our same file.



In this version of the file, we see the start of the file is EF BB BF which indicates UTF-8 format.

Incidentally, the default format of Unicode is UTF-16 in little endian format.

Simple Unicode Formats

Encoding	Byte Mark Representation
UTF-8	EF BB BF
UTF-16 little endian	FF FE
UTF-16 big endian	FE FF
UTF-32 little endian	FF FE 00 00
UTF-32 big endian	00 00 FE FF

Who would have thought this would be so complicated!

Do I really have to memorize all this stuff? Nope. Just bookmark this page.