	Department of Computer and Mathematical Sciences	**CS 1408 Intro to Computer Science with Visual Basic .NET**	**2**

Lab 2: Information Representations – Characters, Pictures, Sounds and instructions

Objectives:

- Learn how characters are represented in computer systems.
- Learn how graphics and images are represented in computer systems.
- Learn how Sounds are represented in computer systems.
- Learn how instructions or programs are represented in computer systems.

Introduction

From Module 1, we know computers store information only as binary numbers, that is, a sequence of 0s and 1s. Information can roughly be categorized into 5 groups. They are

1. Numbers
2. Characters
3. Pictures
4. Sounds
5. Instructions

We have discussed how numbers can be represented using binary numbers in Module 1. We can see in Module 1 we established ways of converting any decimal numbers to binary numbers. So, intuitively, if other types of information such as characters, pictures, sounds, and instructions can be represented as decimal numbers, then we already have a mechanism to convert these types of information into binary representation. Let us explore how such representations can be developed.

Task 1: Character representations

ASCII Code

As previously mentioned, if we can represent each character using a decimal number, then we can convert that particular decimal numbers to binary. Hence we can communicate with a computer. In May 1961, Robert W. Bemer submitted a proposal for a common computer code called **ASCII** (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) to the ANSI (American National Standards Institute) and two years later ANSI agreed upon a common code similar to Bob Bemer's original proposal. Bob Bemer headed the team that created most of the ASCII code. In 1962, IBM wrote and promoted, a coding standard known as **Extended Binary-Coded-Decimal Interchange Code**, or **EBCDIC**, an eight-bit code that was a direct competitor to ASCII. However, ASCII won the standards race.

ASCII (pronounced ask-ee) is a code for representing English characters as decimal numbers, with each letter assigned a number from 0 to 127. For example, the ASCII code for uppercase M is 77 and will be represented in a computer as binary 01001101. Most computers use ASCII codes to represent text, which makes it possible to store and transfer data from one computer to another.

Text files stored in ASCII format are sometimes called ***ASCII files***. Text editors and word processors are usually capable of storing data in ASCII format, although ASCII format is not always the default storage format. Most data files, particularly if they contain numeric data, are not stored in ASCII format.

The standard ASCII character set uses just 7 bits for each character. There are several larger character sets that use 8 bits, which gives them 128 additional characters. The extra characters are used to represent non-English characters, graphics symbols, and mathematical symbols. Several companies and organizations have proposed extensions for these 128 characters. The DOS operating system uses a superset of ASCII called extended ASCII or high ASCII. A more universal standard is the International Standards Organization (ISO) Latin 1 set of characters, which is used by many operating systems, as well as Web browsers. The following table is an ASCII table that shows the ASCII values in HEX as well as decimal. Keep in mind that HEX representation is a binary representation in a compact form.

HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII	HEX	DEC	ASCII
00	0	NULL	20	32	(SP)	40	64	@	60	96	`
01	1	SOH	21	33	!	41	65	A	61	97	a
02	2	STX	22	34	"	42	66	B	62	98	b
03	3	ETX	23	35	#	43	67	C	63	99	c
04	4	EOT	24	36	$	44	68	D	64	100	d
05	5	ENQ	25	37	%	45	69	E	65	101	e
06	6	ACK	26	38	&	46	70	F	66	102	f
07	7	BEL	27	39	'	47	71	G	67	103	g
08	8	BS	28	40	(	48	72	H	68	104	h
09	9	HT	29	41	)	49	73	I	69	105	i
0A	10	LF	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	30	48	0	50	80	P	70	112	p
11	17	DC1	31	49	1	51	81	Q	71	113	q
12	18	DC2	32	50	2	52	82	R	72	114	r
13	19	DC3	33	51	3	53	83	S	73	115	s
14	20	DC4	34	52	4	54	84	T	74	116	t
15	21	NAK	35	53	5	55	85	U	75	117	u
16	22	SYN	36	54	6	56	86	V	76	118	v
17	23	ETB	37	55	7	57	87	W	77	119	w
18	24	CAN	38	56	8	58	88	X	78	120	x
19	25	EM	39	57	9	59	89	Y	79	121	y
1A	26	SUB	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	3C	60	<	5C	92	\	7C	124	|
1D	29	GS	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	3E	62	>	5E	94	^	7E	126	~
1F	31	US	3F	63	?	5F	95	_	7F	127	(sp)

Table 1: ASCII Table

So, from ASCII table, the ASCII value of character **4** is 52_{10} in decimal and 34_{16} in HEX. That means the representation of character **4** in binary is $34_{16} = 0011\ 0100_2$

Another example, the ASCII value of character **s** is 115_{10} in decimal and 73_{16} in HEX. That means the representation of character s in binary is $73_{16} = 0111\ 0011_2$

Note that an ASCII character can be represented in binary with **8 bits**. It is easier to see what binary representation for a character is from the ASCII value in HEX.

Activity 1.1: Use ASCII table to determine how a letter "**a**" is coded or stored in a computer.

Activity 1.2: Use ASCII table to determine how a letter "**K**" is coded or stored in a computer?

Activity 1.3: Use ASCII table to determine how the word "**Computer**" is stored in a computer?

Activity 1.4: Use ASCII table to determine how the sentence "**How are you?**" is stored in a computer?

Activity 1.5: How many bytes does a computer use to store a character?

Activity 1.6: Assume that a text file contains the following information: **The universe has fascinated mankind for many, many years, dating back to the very earliest episodes of "Star Trek" when the brave crew of the starship Enterprise set out, wearing pajamas, to explore the boundless voids of space.**

What is, at least, the size in bytes of the text file containing the above information? Assume that there are no other special characters. Note that space is also a character. Hint: Count characters including spaces.

Unicode

Have you ever tried to include a passage in a different alphabet in one of your documents, for example a quotation in Russian in an English document, only to find that you have no Cyrillic characters available? Or sent a Spanish document in electronic form to someone in Greece, only to be told that the accented Latin characters have been replaced by Greek characters? Or produced a Web page that includes technical symbols and found that it works with Windows but not with Mac OS or Unix? Problems like these arise with non-Latin alphabets and Symbol fonts because until recently most computers used fonts that contain a maximum of 256 characters. The first 128 characters (the ASCII characters) of most fonts include punctuation marks, numbers and the letters a-z and A-Z, and are not a problem. In the USA, Canada, the United Kingdom, the rest of the English-speaking world and much of Western Europe, the second set of 128 characters comprises more punctuation marks, some currency symbols (such as £ and ¥) and a lot of accented letters (such as á, ç, è, ñ, ô and ü). Older English versions of Microsoft Windows, and several other language editions, used this set of 256 characters, which is known as the ANSI character set.

If you live in a country such as China, Egypt, Greece, Israel, Japan, Korea, Russia or Thailand that uses a different alphabet, then your version of Windows may use a different character set. The first 128 characters will be the same as in ANSI, but many of the places in the second set of 128 will be taken by characters from the Arabic, Greek, Hebrew, Cyrillic or Thai alphabets. Now that documents are often transferred electronically as e-mail messages, e-mail attachments or Web pages, instead of on paper, reading documents from another country, particularly a country with a different alphabet, is becoming more of a problem. There are similar problems when moving documents between operating systems such as DOS, Windows, Mac OS and UNIX.

The solution is to leave behind the assortment of 8-bit fonts with their limit of 256 characters, where the same character number can represent a different character in different alphabets, and move to a system that assigns a unique number to each character in each of the major languages of the world. Such a system has been developed and is known as ***Unicode***. It is intended for use on all computer systems, not just Windows, and covers Chinese, Japanese and Korean as well as the alphabets for many other languages and scripts, plus a large number of special characters. Some Unicode support has been included in Microsoft Windows since Windows 95, and Windows NT 4, Windows 2000 and Windows XP are based on Unicode instead of the ANSI or Windows Glyph List 4 (WGL4) character sets. WGL4 contains characters that are required for Western, Central and Eastern European languages, and includes Cyrillic and Greek alphabets and many characters for which Monotype's Symbol font was previously required. The WGL4 standard incorporates codepages 1250 (Eastern Europe), 1251 (Cyrillic), 1252 (US English = ANSI), 1253 (Greek) and 1254 (Turkish). WGL4 contains 652 characters (compared with 256 in the old codepages), and uses Unicode numbering for the characters. Some Unicode support has

been included in Mac OS since Mac OS 8.5, but prior to Mac OS X 10 only limited use has been made of it by applications. Unicode is often referred to as a 16-bit system, which would allow for only 65,536 characters, but this is not correct, and Unicode has the potential to cope with over one million unique characters

The current version (5.0.0) of the Unicode Standard, developed by the Unicode Consortium, assigns a unique identifier to each of 99,024 graphical and formatting characters, covering the scripts of the world's principal written languages and many mathematical and other symbols. A previous version (2.1) of the Unicode Standard encompassed 38,887 characters and was adopted as part of the recommendations for HTML 4.0. The following is an example taken from the Web site http://en.wikipedia.org/wiki/Unicode_and_HTML

Character	HTML char ref	Unicode name	What your browser displays
U+0041	&#65;	Latin capital letter A	A
U+00DF	&#223;	Latin small letter Sharp S	ß
U+00FE	&#254;	Latin small letter Thorn	þ
U+0394	&#916;	Greek capital letter Delta	Δ
U+0419	&#1049;	Cyrillic capital letter Short I	Й
U+05E7	&#1511;	Hebrew letter Qof	ק
U+0645	&#1605;	Arabic letter Meem	م
U+0E57	&#3671;	Thai digit 7	๗
U+1250	&#4688;	Ethiopic syllable Qha	
U+3042	&#12354;	Hiragana letter small A (Japanese)	あ
U+53F6	&#21494;	CJK Unified Ideograph-53F6 (Simplified Chinese "Leaf")	叶
U+8449	&#33865;	CJK Unified Ideograph-8449 (Traditional Chinese "Leaf")	葉
U+B0FB	&#45307;	Hangul syllable Nyaelh (Korean "Nieun Yae Rieulhieuh")	냻
U+10346	&#66374;	Gothic letter Faihu	𐍆
To display all of the characters above, you may need to install one or more large multilingual fonts, like Code2000 (and Code2001 for some extinct languages, for example Gothic).			

Table 2: Unicode Table

So, the Unicode value of character **A** is 65_{10} in decimal and 0041_{16} in HEX. That means the representation of character **A** in binary is $0041_{16} = 0000\ 0000\ 0100\ 0001_2$

Another example, the Unicode value of character the Greek capital letter delta Δ is 916_{10} in decimal and 0394_{16} in HEX. That means the representation of character delta Δ in binary is $0394_{16} = 0000\ 0011\ 1001\ 0100_2$

Activity 1.7: How is a letter "**a**" coded in a computer using Unicode?

Activity 1.8: How is a letter "**K**" coded in a computer using Unicode?

Activity 1.9: How is a Greek capital letter Delta "**Δ**" coded in a computer using Unicode?

Activity 1.10: How is a Thai digit 7 " " coded in a computer using Unicode?

Task 2: Picture representations

An ever-increasing application of computers is the processing of VISUAL or GRAPHICAL information. There is a wide variety of data that is classified as graphical, from print graphics like character typefaces and simple hand-drawn images, to still photographs, moving pictures and video, to sophisticated two- and three-dimensional animations generated entirely by a computer. No matter how complicated the information, however, if it can be represented in digital (binary) form, then it can be processed by a computer. You may have heard about digital TVs. These are televisions that will receive broadcast video images in binary form, and then decode these images for display on the screen. TV images that are digitally encoded can be saved permanently, like files on a diskette, and are not subject to degeneration like images recorded on video tape. There are many types of binary codes for representing graphical information. One of the easiest to understand is based on the notion of a ***PIXEL***, or "picture element".

A PIXEL is a tiny black, white, or colored rectangle. A graphics image, such as a drawing or photograph, can be "*pixelated*" by dividing the image into an array of these tiny solid rectangles. Pixelated images are often called BITMAPS.

Figure 1: Original Black and White Rose

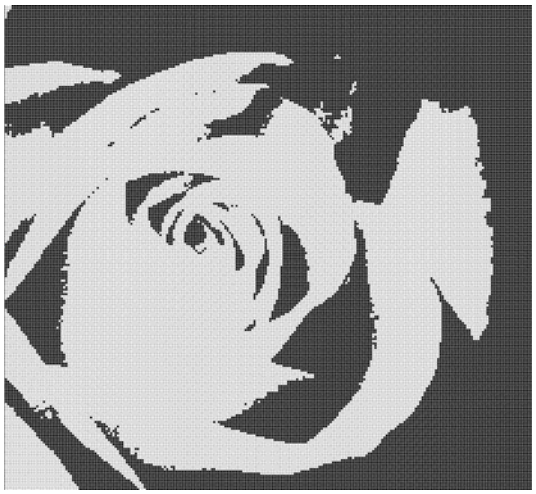


Figure 2: Black and White Rose Pixilation

The black-and-white picture that you see in Figure 1 is enlarged using Microsoft Paint application software and displayed with **Show Grid** option in **Zoom** sub menu under **View** menu to illustrate pixelation effect. This image can be coded in binary form by converting each pixel to

a bit - black pixels can be represented by 1's and white pixels can be represented by 0's, or vice versa.

Activity 2.1: For black and white pictures, each pixel represents either white or black color which computers will store as 0 or 1. How many bits does it take for a pixel to store color information as black or white?

Activity 2.2: Assume that the black and white Rose picture in Figure 1 composed of 32,800 pixels, each either solid white or solid black (there are 200 pixels across and 164 pixels down). How many bytes (at least) will it take to store the picture?

Now let us look at pictures that have more than two colors, black and white. Consider the following the picture in Figure 3, you can see the picture of a tree has not only black and white colors but also different shades or levels of gray color. In Figure 4, the same picture is display with pixilation. You can notice that some pixels are white, some are black, some are gray and some are grayer.

Figure 3: Original Tree Picture

Figure 4: Tree Picture with Pixalation

Activity 2.3: Suppose the tree picture in Figure 3 contains only 4 colors: black, white, 25% gray and 50% gray. How many bits does it take for a pixel to store color information?

Activity 2.4: Assume that the tree picture in Figure 3 composed of 32,800 pixels, each either solid white, solid black, 25% gray, and 50% gray (there are 200 pixels across and 164 pixels down). How many bytes (at least) will it take to store the tree picture?

Activity 2.5: Suppose that a picture contains 8 colors. How many bits does it take for a pixel to store color information of 8 colors.

Activity 2.6: Assume that a picture composed of 32,800 pixels, each can be either one of the 8 colors (there are 200 pixels across and 164 pixels down). How many bytes (at least) will it take to store the picture?

Activity 2.7: Suppose that a picture contains 256 colors. How many bits does it take for a pixel to store color information of 256 colors.

Activity 2.8: Assume that a picture composed of 32,800 pixels, each can be either one of the 256 colors (there are 200 pixels across and 164 pixels down). How many bytes (at least) will it take to store the picture?

In general, pictures contain many colors. Computer systems use RGB (**R**ed **G**reen **B**lue) RGB colors. RGB is an additive color system, which means that color is added to a black background. Black is the absence of light and therefore the absence of color. Secondary colors, such as cyan, magenta and yellow, are created by combining the primary colors. The color white is achieved by adding the three primary colors together in equal amounts. In grade school we learn that there are three primary colors - red, yellow and blue - that when combined produce secondary colors such as green, purple and orange, and all the various shades and variations on the primary and secondary colors. However, these are the primary colors of pigment and different systems apply to the colors that we see on computer screens and the colors that we see in printed media and photographs.

Most computers are capable of displaying RGB colors. These RGB colors are obtained from mixing different shades or values of red, green, and blue where each color has 256 shades or values. So, an RGB color is a combination of red, green and blue colors represented by values of each color, red, green and blue, as shown in the following examples.

Color	RGB
red	255.0.0
green	0.255.0
blue	0.0.255
lavender	230.230.250

For lavender color is a mixture of red (230), green (230), and blue (250). Since each color has 256 shades or values, there are a total of 256×256×256 = 16,777,216 combinations or colors.

Activity 2.9: Suppose that a picture contains RGB colors. How many bits does it take for a pixel to store color information of RGB colors.

Activity 2.10: Assume that a picture composed of 20,000 pixels, each can be either one of the RGB colors. How many bytes (at least) will it take to store the picture?

Activity 2.11: What is the RGB value for Black color?

Activity 2.12: What is the RGB value for Pink color? Use the Internet to search for the value.

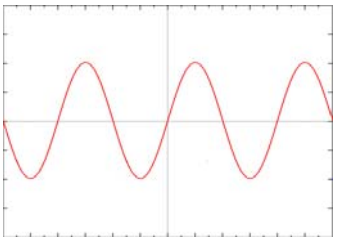
Task 3: Sound representations

As previously mentioned, if we can represent information as numbers, then we can represent that information in binary just like ASCII characters and pictures. This is true with sound information or data.

What is a sound?

Sound is a varying air pressure wave, produced by all manner of sources that is sensed by our ears. Sound occurs naturally as an analog wave. In an analog sound system, the pressure wave is captured by some kind of a device such as a microphone that produces an electrical voltage or current that varies proportionally to the sound pressure. This electrical signal can then be transmitted by telephone, or broadcast by radio, or preserved on magnetic tape such as audio cassettes or used in other ways. En route, the sound might be processed to change its character in some way, for example to reduce noise or squeeze out unwanted or unneeded frequencies. The electrical signal is used to recreate the sound by vibrating some mechanical surface in a loudspeaker or an earphone, reproducing the original pressure wave (with varying degrees of fidelity) so that we can hear the sound.

A simplest kind of sound wave is a sine wave as shown in the right figure. It has the two fundamental properties, *frequency* and *amplitude*.



- Frequency (pitch) of a sound is the number of times the pressure rises and falls, or oscillates, in a second and is

measured in hertz (Hz). A frequency of 100 Hz means 100 oscillations per second. A convenient abbreviation, kHz for kilohertz, is used to indicate thousands of oscillations per second: 1 kHz equals 1000 Hz. The frequency range of normal human hearing extends from around 20 Hz up to about 20 kHz.

– Amplitude is the level of loudness of a sound wave. The amplitude of a sound is the measure of the displacement of air pressure from its mean. The greater the amplitude is, the louder is the sound, usually measured in *decibels* (dB)

Before a sound can be analyzed or understood by a computer, we need to find a way to represent the two fundamental properties of sound, *frequency* and *amplitude*, *digitally*. The process of taking analog data, such as sound, and making it digital is called *analog to digital conversion*.

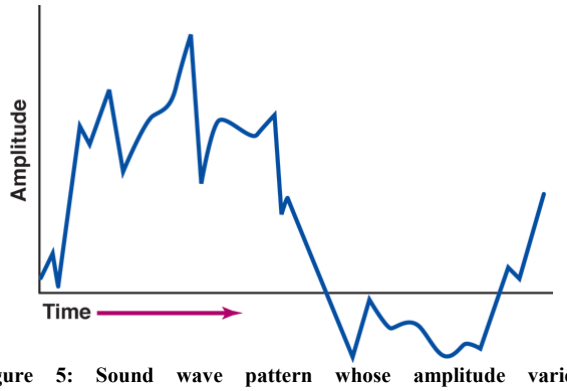


Figure 5: Sound wave pattern whose amplitude varies continuously over time.

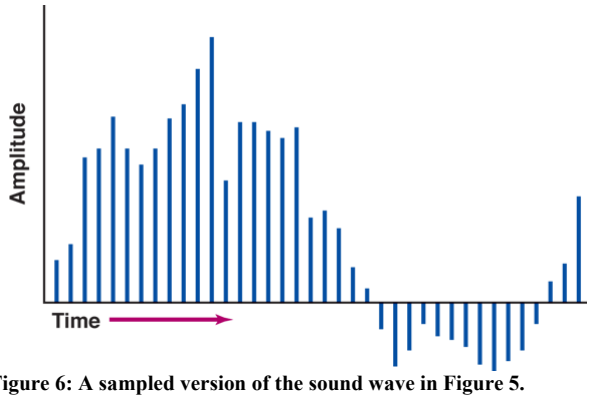


Figure 6: A sampled version of the sound wave in Figure 5.

To convert an analog wave into digital, converters use a process called *sampling*. A computer measures the amplitude of the waveform at regular time intervals to produce a series of numbers. Each of these measurements is called a *sample*. Figure 6 illustrates a digitally sampled waveform of the sound wave in Figure 5. Each vertical bar in Figure 6 represents a single sample. The height of a bar indicates the value of that sample. The rate at which a waveform is sampled is called the *sampling* rate.

Sampling is done at regular intervals of time, often small fractions of a second. Like frequencies, sampling rates are measured in hertz. The CD standard sampling rate of 44.1 kHz means that the waveform is sampled 44100 times per second. This may seem a bit excessive, considering that we can't hear frequencies above 20 kHz; however, the highest frequency that a digitally sampled signal can represent is equal to half the sampling rate. So a sampling rate of 44100 Hz can only represent frequencies up to 22050 Hz, a boundary much closer to that of human hearing. If one byte is used to hold a single sample of an analog wave, then the wave can be one of 256 different heights (0 being the lowest height and 255 being the highest). These heights represent the *decibel* level of the sound. Thus a spoken word might occupy several hundred bytes - each being a sample of the sound wave of the voice at a small fraction of a second. If these 100 bytes were sent to a computer's speaker, the spoken word would be reproduced.

The precision with which the digitized signal represents the continuous signal depends on two parameters of the digitizing process: the rate at which amplitude measurements are made (the *sampling rate* or *sampling frequency*), and the number of bits used to represent each amplitude measurement (the *sample size* or *bit depth*).

Commercial digital audio applications use sampling rates of 44.1 kHz (for audio compact discs, CD) or 48 kHz (for digital audio tape). Once a signal is digitized, its sampling rate is fixed. In

order to interpret a sequence of numbers as representing a time-varying signal, one needs to know the sampling rate. Thus, when a digitized signal is saved in a file format that is designed for saving sound information (such as AIFF or WAV), information about the sampling rate is saved along with the actual data points comprising the signal.

The more frequently a signal is sampled, the more precisely the digitized signal represents temporal changes in the amplitude of the original signal. The sampling rate that is required to make an acceptable representation of a waveform depends on the signal's frequency. More specifically, the sampling rate must be more than twice as high as the highest frequency contained in the signal. Otherwise, the digitized signal will have frequencies represented in it that were not actually present in the original at all.

The precision in which a sample represents the actual amplitude of the waveform at the instant the sample is taken depends on the sample size or number of bits (also called bit depth) used in the binary representation of the amplitude value. Some digitizers can take samples of one size only; others allow you to choose (usually through software) between two or more sample sizes. An 8-bit sample can resolve 256 ($=2^8$) different amplitude values; a 16-bit converter can resolve 65,536 ($=2^{16}$) values. Sound recorded on audio CDs is stored as 16-bit samples. When a sample is taken, the actual value is rounded to the nearest value that can be represented by the number of bits in a sample.

Since the actual analog value of signal amplitude at the time of a sample is usually not precisely equal to one of the discrete values that can be represented exactly by a sample, there is some digitizing error inherent in the acquisition process), which results in quantization noise in the digitized signal. The more bits used for each sample, the less quantization noise is contained in the digitized signal. If you listen to a signal digitized with 8-bit samples using high-quality headphones, you can hear the quantization noise as a low-amplitude broadband hiss throughout the recording. Signals digitized with 16-bit samples typically have no detectable hiss.

The ratio between the value of the highest amplitude sample that can be represented with a given sample size and the lowest non-zero amplitude is called the *dynamic range* of the signal, and is usually expressed in decibels (dB). The dynamic range corresponds to the ratio in amplitude between the loudest sound that can be recorded and the quantization noise. The dynamic range of a digitized sound is 6 dB/bit.

The increased frequency bandwidth obtainable with higher sampling rates and the increased dynamic range obtainable with larger samples both come at the expense of the amount of memory required to store a digitized signal. The minimum amount of storage (in bytes) required for a digitized signal is the product of the sample rate (in samples/sec), the sample size (in bytes; one byte equals 8 bits), and the signal duration (seconds). Thus, a 10-second signal sampled at 44.1 kHz with 16-bit (2-byte) precision requires 882,000 bytes (= 10 sec x 44,100 samples/sec x 2 bytes/sample), or about 861 Kbytes of storage (1 Kbyte = 1024 bytes). The actual amount of storage required for a signal may exceed this minimum, depending on the format in which the samples are stored.

One common digital process is compression: by taking advantage of how human hearing works, it is possible to compress sound data a great deal without having a perceptible effect on its quality. This fact is at the heart of music formats like MP3, which are typically 10 times smaller than the equivalent uncompressed sound.

What is WAV?

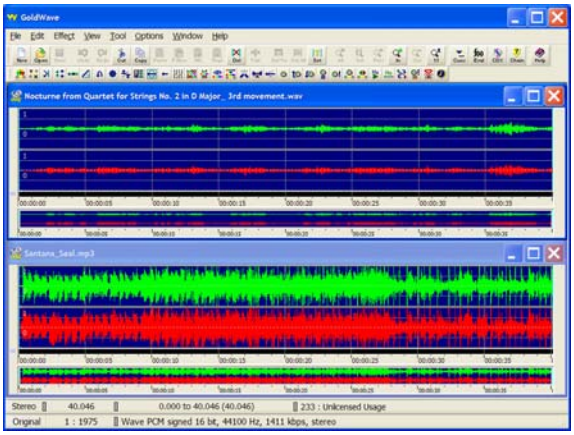
WAV is a format for storing sound in files. Microsoft and IBM introduced the wav file in 1991 for use on the Microsoft Windows 3.1 operation system. WAV sound files end with .wav extension and can be played by nearly all Windows applications that support sound. Long before digital audio became a staple, computer users were exposed to the wav file as an embedded sound file that played a chime-like sound at boot up of the Windows operating system.

The wav format is based on the Resource Interchange File Format (RIFF), which stores audio files in indexed "chunks" and "sub-chunks." RIFF is in turn based on the earlier Interchange File Format (IFF), established by Electronic Arts in 1985 for use in electronic gaming. Apple's version, known as Audio Interchange File Format (AIFF), was released in 1988 for Macintosh computers. Due to the common roots of these various audio formats, however, the audio files will play on any computer system, IBM or Apple.

The wav file could digitize sounds 100% faithful to the original source because it is a *lossless* format. "Lossless" means that the wav file format does not compromise audio quality even when it holds compressed data.

What is MP3?

MP3 (MPEG Audio Layer-III) is a standard format for compressing digital audio. MP3 squeezes (rips) audio files to about one tenth of their original size, while maintaining close to CD quality. Songs in MP3 format can be downloaded from the Internet, created from prerecorded music, or recorded from scratch. They can be played on personal computers, Walkman-style portable players or one of the new generations of dual-mode MP3/audio CD players.



The main benefit of MP3 is its high level of compression, which makes for smaller files and faster downloads. With MP3, you can store more than 12 hours of high-quality music on a single CD, versus the standard 74 minutes. With the matchbook-sized memory cards used by portable MP3 players, you can fit the equivalent of a dozen CDs in your wallet.

The figure on the right shows the spectrum of sound for classical and rock music using **GoldWave** share ware.

GoldWave is software that capable of converting a WAV file to an MP3 file and vice versa.

Activity 3.1: What are the two characteristic of sound wave?

Activity 3.2: What is the process of converting analog waveform to digital waveform?

Activity 3.3: What does amplitude of the sound wave measure?

Activity 3.4: Download the WAV files: Nocturne from Quartet for Strings No. 2 in D Major_ 3rd movement and Santana_Seal from my web page. What is the file size of these two WAV files? Use the file Properties in Windows Explorer to check the file size.

Activity 3.5: Use Software GoldWave to convert WAV format to MP3 format. What is the file size of the converted MP3 files?

Task 4: Instruction representations

When we refer to instructions, basically we mean an instruction set that use in a computer to perform a certain task. Some of operations that a computer performs are arithmetic operations such as addition, subtraction, multiplication, division. For the sake of simplicity, we will restrict ourselves to arithmetic operations. We will avoid actual details of how instructions are designed for a computer.

Suppose you want to design an instruction set for a computer that can perform only addition, subtraction, multiplication, and division. The computer is capable of processing 8 bits per instruction. Since there are 4 operations, you decide to use 2 bits for operations and 3 bits for each operand or number with a total of 8 bits. The operations can be coded with 2 bits as follows:

Operation	Code
Add	00
Subtract	01
Multiply	10
Divide	11

For example, the instruction **Add 2, 3** where we refer to **2** and **3** as operands can be represented in binary as **00010011**. **00** represents **Add**, **010** represents **2**, and **011** represents **3**.

Activity 4.1: How is the instruction **Add 5, 4** represented in binary (8 bits)?

Activity 4.2: How is the instruction **Subtract 5, 4** represented in binary (8 bits)?

Activity 4.3: How is the instruction **Multiply 5, 4** in binary (8 bits)?

Activity 4.4: How is the instruction **Divide 5, 4** represented in binary (8 bits)?

Activity 4.5: What is the maximum number (value) for each operand?

Activity 4.6: Suppose you want to design an instruction set for a computer that can process a 16-bit instruction and can perform 16 operations with 7 operations are addition, subtraction, multiplication, Integer division, division, modulus (remainder of division), negation (change negative number to positive number and vice versa), and the rest to be added later.
At least how many bits must you use to code 16 operations?

Activity 4.7: Design the code for operations described in Activity 4.6 as in Activity 4.1.

Operation	Code
Add	
Subtract	
Multiply	
Integer Divide	
Divide	
Modulus	
Negation	

Activity 4.8: How many bits are there in each operand?

Activity 4.9: What is the instruction **Modulus 5, 4** represented in binary?