Computer Science

COMPSCI 111 SC - Lecture 2

July 2003

The Digital World

(Representing Information)

Analog Systems

Analog information is that which exists in a continuous domain. The strength of the signal in an analog system varies constantly, albeit by minor amounts some of the time. Both light and sound fall into this category, being able to vary to any degree. When we measure analog information, we quantize it by assigning a discrete value to the information. Example: Measuring a length (analog) to the nearest millimetre (digital).

Digital Systems

A digital system is simply a system which uses discrete numbers to represent information. In order to achieve this, we must have a method of converting the information into numbers. Example: We could decide that the letter 'A' will be represented by the number '1', 'B' will be '2', 'C' will be '3' and so on. It would then be possible to write a sentence using numbers instead of letters (e.g. "CAB" could be represented as 3, 1, 2). You will note that it is very important that we know what process has been used to encode the information. If one method of encoding has been used to convert letters into numbers, and a different method is used to convert from numbers back to letters, the end result will be gibberish.

Standards

A definition or format that has been approved by a recognised standards organisation or is accepted as a de facto standard by the industry.

In the computer market, there is often a need for information to be exchanged. Information is often shared between different programs on the same computer (ie; picture being placed into a word processing document), between different people (who may use different applications) or between people who use different operating systems.

If information is to be shared between two systems, then they need to be able to understand the same information. That means that the format, or the way that the information is represented, needs to be understood by both systems. In order for technology to advance globally, everyone needs to use a standard accepted method of representing information. Sometimes these standards are developed by committees of industry experts, in which case they are referred to as official standards. Sometimes a product becomes so common and so widely used that it becomes a de facto standard (examples include VHS video and Postscript).

Standards are extremely important for users, because they allow products from a variety of sources to be combined. Without standards, only products from the same source would be able to be used.

These standards exist in all aspects of computer technology, from hardware to software, operating systems to applications, text formatting to picture and sound representation. Anytime that information is stored or displayed on a computer, standards are being used at some level.

Low Level Representation

Text, Pictures, Sounds, Video and all the other information stored in a computer is actually numbers. You can see how text is represented by looking at the ASCII table below. Storing pictures, sounds and video requires different methods of encoding, but eventually, it all turns into numbers. But computers are electronic devices, so how do they store these numbers. The answer is (as you may have guessed) by using binary.

Signals inside a computer are sent along thin wires. The signal is either a high voltage, or a low voltage. That means that each signal has 2 possibilities, which is exactly what a binary system is.

ASCII (pronounced ask-ee)

Acronym for American Standard Code for Information Interchange. ASCII is a code which is used to represent English characters as numbers. These codes are recognised by most computer systems, and provide a method of transferring text from one machine to another.

The full table of ASCII codes follows. These are the only characters which may be represented using ASCII. Other encoding systems, such as Extended ASCII, EDBIC, and Unicode are commonly supported by many computer systems

0	00	000	NUL (Null)	46	2E	056	.	92	5C	134	\
1	01	001	^A SOH (Start of heading)	47	2F	057	/	93	5D	135	]
2	02	002	^B STX (Start of text)	48	30	060	0	94	5E	136	^
3	03	003	^C ETX (End of text)	49	31	061	1	95	5F	137	_
4	04	004	^D EOT (End of transmission)	50	32	062	2	96	60	140	`
5	05	005	^E ENQ (Enquiry)	51	33	063	3	97	61	141	a
6	06	006	^F ACK (Acknowledge)	52	34	064	4	98	62	142	b
7	07	007	^G BEL (Ring bell)	53	35	065	5	99	63	143	c
8	08	010	^H BS (Backspace)	54	36	066	6	100	64	144	d
9	09	011	^I HT (Horizontal tab)	55	37	067	7	101	65	145	e
10	0A	012	^J LF (Line feed)	56	38	070	8	102	66	146	f
11	0B	013	^K VT (Vertical tab)	57	39	071	9	103	67	147	g
12	0C	014	^L FF (Form feed)	58	3A	072	:	104	68	150	h
13	0D	015	^M CR (Carriage return)	59	3B	073	;	105	69	151	i
14	0E	016	^N SO (Shift out)	60	3C	074	<	106	6A	152	j
15	0F	017	^O SI (Shift in)	61	3D	075	=	107	6B	153	k
16	10	020	^P DLE (Data link escape)	62	3E	076	>	108	6C	154	l
17	11	021	^Q DC1 (Device control 1)	63	3F	077	?	109	6D	155	m
18	12	022	^R DC2 (Device control 2)	64	40	100	@	110	6E	156	n
19	13	023	^S DC3 (Device control 3)	65	41	101	A	111	6F	157	o
20	14	024	^T DC4 (Device control 4)	66	42	102	B	112	70	160	p
21	15	025	^U NAK (Negative acknowledge)	67	43	103	C	113	71	161	q
22	16	026	^V SYN (Synchronous idle)	68	44	104	D	114	72	162	r
23	17	027	^W ETB (End transmission blk.)	69	45	105	E	115	73	163	s
24	18	030	^X CAN (Cancel)	70	46	106	F	116	74	164	t
25	19	031	^Y EM (End of medium)	71	47	107	G	117	75	165	u
26	1A	032	^Z SUB (Substitute)	72	48	110	H	118	76	166	v
27	1B	033	ESC (Escape)	73	49	111	I	119	77	167	w
28	1C	034	FS (File separator)	74	4A	112	J	120	78	170	x
29	1D	035	GS (Group separator)	75	4B	113	K	121	79	171	y
30	1E	036	RS (Record separator)	76	4C	114	L	122	7A	172	z
31	1F	037	US (Unit separator)	77	4D	115	M	123	7B	173	{
32	20	040	SP (Space)	78	4E	116	N	124	7C	174	|
33	21	041	!	79	4F	117	O	125	7D	175	}
34	22	042	"	80	50	120	P	126	7E	176	~
35	23	043	#	81	51	121	Q	127	7F	177	DEL
36	24	044	$	82	52	122	R				
37	25	045	%	83	53	123	S				
38	26	046	&	84	54	124	T				
39	27	047	'	85	55	125	U				
40	28	050	(	86	56	126	V				
41	29	051	)	87	57	127	W				
42	2A	052	*	88	58	130	X				
43	2B	053	+	89	59	131	Y				
44	2C	054	,	90	5A	132	Z				
45	2D	055	-	91	5B	133	[				

Figure 1: Table of ASCII codes

Binary and Decimal Number Systems

All numbers are merely a symbolic representation of quantitative value. Just as words are symbols which have a particular meaning, so too are numbers. The meaning of numbers is to represent a particular quantity. These quantities can be represented by different systems.

We are all familiar with the idea that words which have a particular meaning may have different symbolic representations in different languages (eg; water = aqua = eau). Similarly, numbers with a particular quantitative value may have different representations in different systems (eg; $12_{(decimal)} = C_{(hexadecimal)} = 1100_{(binary)}$)

Decimal Numbers

It is natural for humans to use decimal, or base 10 for numbers because we each have 5 digits on each hand = 10 digits (8 fingers and 2 thumbs). The numerical representation of these numbers is: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. You can see there are only 10 different digits. This number system is Arabic in origin, and represents a major advance over the previously used Roman numerical system.

Counting in Decimal

We are all familiar with counting in the decimal system. We can count to 10 using our fingers, but then what? We need to make a note that we have reached 10 once, and we can continue counting on our fingers. If we were only counting on fingers, we would need one person to keep track of the individual digits (the number of single units), one person to keep track of how many times we reached 10, another to keep track of the number of times 100 was reached, and so on.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
...									
90	91	92	93	94	95	96	97	98	99
100									

Decimal Powers

All decimal numbers can be broken into individual digits, each of which belongs in a particular column. Each new column has a value = previous column x 10. For this reason, the decimal number system is also known as the base-10 system.

Number	*10000*	*1000*	*100*	*10*	*1*
305			3	0	5
83526	8	3	5	2	6

The *nth* column can be represented by 10 x 10 x ... x 10 x 10, where there are *n* lots of 10 multiplied together. This is the same as 10^n. So the table can be rewritten as:

Number	10^4	10^3	10^2	10^1	10^0
305			3	0	5
83526	8	3	5	2	6

Binary Numbers

The *numbers* represented by the binary system are the same as those represented by decimal. Only the symbolic representation changes. The binary system uses only 2 digits: 0, or 1. This system is ideal for computers, since a digit can easily be represented by an electrical signal, or switch (0 = OFF, 1 = ON).

Counting in Binary

The process of counting in binary works in a similar way to decimal, except that only 2 digits are available. This means that when we run out of digits, the new digit we create records how many times we have counted from 0 to 1. This digit records how many groups of two we have counted. Again the process continues for each new number.

eg:

0	1
10	11
100	101
110	111
1000	1001
...	

Binary Powers

Just like decimal numbers, all binary numbers can be broken into individual digits. Each of these digits belong in a particular column. Each new column has a value = previous column x 2. For this reason, the binary number system is also known as the base-2 system.

	16	*8*	*4*	*2*	*1*
101			1	0	1
11101	1	1	1	0	1

The *nth* column can be represented by 2 x 2 x ... x 2 x 2, where there are *n* lots of 2 multiplied together. This is the same as 2^n. So the table can be rewritten as:

	2^4	2^3	2^2	2^1	2^0
101			1	0	1
11101	1	1	1	0	1

Converting from Binary to Decimal.

Since we know that each column in a binary number has twice the value of the previous column, it is easy to convert from one to the other.

1110101 = (1 x 1) + (0 x 2) + (1 x 4) + (0 x 8) + (1 x 16) + (1 x 32) + (1 x 64)
= 1 + 4 + 16 + 32 + 64
= 117

Converting from Decimal to Binary.

It is a little more difficult to convert from decimal to binary. You must split the number up, so that it can be written using powers of 2. In order to do this, you need to know your powers of 2 quite well. Write them down if you forget them, starting with one, and doubling each time (1, 2, 4, 8, 16, 32, 64, 128, 256, 512 …).

A number such as 7 can be written as 4 + 2 + 1, which is 111 in binary.

To convert from decimal to binary, write down all the powers of 2 until you reach a number bigger than the decimal one. Use these as the headings of columns to make sure that you put the digits in the correct place. The biggest column which is *less* than the decimal number you are converting must have a one in it. You can then subtract that number from the number to convert, and see what is left. Repeat the process with the remaining number until the entire number is accounted for. Example: Convert 235 to binary.

	256	128	64	32	16	8	4	2	1
235		1							
235-128 = 107			1						
107 - 64 = 43				1					
43 - 32 = 11						1			
11 - 8 = 3								1	
3 - 2 = 1									1
235 = 128+64+32+8+2+1		1	1	1	0	1	0	1	1

So 235 in decimal = 11101011 in binary

There are other methods to convert decimal to binary. Use any method which provides the correct result.

Adding Decimal numbers

With decimal, adding is done by columns, using a carry into the next column if the digits in one column add up to more than 10.

```
  1 1 1          <- Carry
  1 2 3 4 5
+ 6 7 8 9 0
  8 0 2 3 5
```

Adding Binary numbers

The same method is used with binary, but remember that there are only 2 digits to use.

```
    1 1 1        <- Carry
  1 0 1 1 1 0
+ 0 0 1 0 1 1
  1 1 1 0 0 1
```

Conventions

For ease of recognition, large decimal numbers are usually grouped into threes, with commas seperating each component. For example, 1324557232 is written 1,324,557,232. A similar convention applies with binary numbers, except they are grouped into fours, and spaces are used to separate the components. For example, 1010010100101 is written as 1 0100 1010 00101.

Exercises

1. Convert the following numbers from binary to decimal

(a) 11
(b) 1010
(c) 10111
(d) 1000 0000
(e) 0111 1111
(f) 1110 1101

2. Convert the following numbers from decimal to binary

(a) 6
(b) 12
(c) 13
(d) 63
(e) 132
(f) 439

3. Complete the following binary additions, and convert the answer to decimal

(a)

```
  1 0 0 0  0 0 0 0
+ 0 0 0 0  0 0 1 0
```

(b)

```
  0 0 0 0  0 0 1 1
+ 0 0 0 0  1 0 0 1
```

(c)

```
  0 1 0 1  0 0 0 1
+ 1 0 1 0  1 1 1 0
```

(d)

```
  1 1 1 0  0 1 1 1
+ 1 1 0 1  1 1 0 1
```


Interfacing with the Machine

(Computer Software)

Programs

A program is a list of step-by-step instructions that tell the computer hardware what to do. Some programs are built into the hardware (ROM), others are stored in erasable programmable ROM chips (EPROM) for semi-permanent storage. These more permanent programs are known as “firmware” or "hard-wired", and are often used in home appliances (such as washing machines, videos and microwaves). The most common programs used by computers are known as software. They are typically stored on secondary storage and loaded into RAM before they are run (or executed). Software can be divided into two categories, applications software and system software.

Application Software

Application software is designed for an “end-user” to achieve a task. These programs act as tools for the user. The most common applications are word processors, spreadsheets, databases, graphics programs, and communications programs. There are also integrated packages of products which incorporate more than one kind of tool into a single integrated program. An example of this is ClarisWorks. Other companies produce several independent programs which are designed to work together by sharing data in a common format. These programs are often sold together as a software suite, and a common example is Microsoft Office.

Applications software can be described as public domain, shareware or proprietary. Public domain software is available free of charge to anyone who wants it. Programs of this type are called freeware and vary substantially in quality. Shareware is distributed freely (so users are able to test the product), but done so with the proviso that users who keep the product send some money to the author. Software of this type relies upon the honesty and goodwill of the computer community. Some authors release a shareware version of the software which is able to be evaluated for a number of days before it no longer works. Others produce shareware which does not have all the options enables (e.g. unable to save). On the other hand, proprietary software is owned by an individual or company that retains the rights. This software is usually sold commercially, and copying or distributing the program is illegal. It is possible to get demonstration versions of some commercial software so that you can try the software and evaluate before you buy.

System Software

System software is used to help manage the overall operation of the computer system. Examples of system software include compilers, interpreters, system administration software and utility programs. The most important example of system software is the *operating system* (OS). The operating system consists of a collection of programs which work together to manage the computer resources. Operating systems are designed to be either single-user systems, or multi-user systems. A single user system is perfect for a personal computer, where only a single user will be operating the computer at any one time. A multi-user system is important for larger organisations where a central computer is being used simultaneously by many people (each using a different terminal to communicate with the main computer).

Operating Systems

The operating system is the software which manages the system resources(including file management, allocation of storage, providing an interface to peripheral devices) and presents a default interface to the user when no application is running.

A computer is merely a device which follows instructions. It needs instructions to tell it how to run a program, how to access the information on the hard-drive, how to display text on the screen, where the keyboard is, and how to interact with the user. These fundamental instructions which allow the user to operate the computer for useful tasks are collected together and are known as the operating system. It is this operating system which runs when you switch the computer on and allows you to use the computer.

Single-user operating systems are designed for situations where a single person will be using the computer at any one time. Moreover, that person will use the computer exclusively (e.g. a personal computer in your office or home will typically use a single-user operating system). Examples include Windows 95, Mac OS, DOS.

Multi-user operating systems are designed for situations where many people are using the same computer at the same time. It is necessary in such cases to have a high level of security, so that files can be kept safe from other users. A more complex system of sharing resources is typically required for such systems, since jobs are required to be completed at the same time (e.g. many people may be running programs simultaneously, all of which require processing time). Multi-user operating systems usually solve this problem by time-sharing, or allocating a certain amount of time (usually a fraction of a second) to work on each job. The processor works for a short time on each program, before moving on the next one. When the last job is reached, the processor starts at the beginning again, and proceeds to cycle through the jobs until they are complete. Examples of multi-user operating systems include: Unix, Linux, Windows NT

Operating systems are all different, but they typically include the following components:
1. Supervisor (or Kernel). This component is responsible for managing the different programs or jobs which are requested. It starts programs, manages the memory and CPU time required to run the program, and oversees the other components. In a multi-user system, the Kernel is responsible for sharing resources between different users.
2. I/O Control Drivers (Device Drivers) These components provide the technical details required to operate the hardware devices, and handles the sending and receiving of information between the CPU and any input, output, communication or secondary storage hardware devices.
3. Memory Management. The memory management component is responsible for maintaining records of which memory is used and which is free. Since programs must be loaded into memory before they can be executed, it is necessary to allocate memory for each program and recover the memory when the program is finished.
4. File Management. A file is a collection of data. The file management system records where and how the data is stored (using secondary storage media). In a multi-user system, it must also ensure that files can only be manipulated by those users who are allowed.
5. User Interface. A user interface allows the user to communicate with a program. The operating system requires a user interface so that the user can tell the operating system what to do (e.g. start a program).

Flow of Information

The operating system acts as an interface between application software and the hardware. Programs are usually written to be compatible with a particular operating system, rather than specific hardware. A program written for one operating system will not run on a different operating system.

Since the operating system defines how information is to be stored on a computer, it is not surprising that information stored by one operating system cannot usually be understood by a different one. You need special conversion facilities, which tell one operating system how the others store information. Not all computers have these facilities available. For example; The Macintosh OS can read files stored in Windows 95 (or DOS) format, however, the PC cannot read files stored in the Macintosh OS format.

The operating system is just software, and so a version of the software can be written for any platform (ie; hardware). A version of Unix exists for both the IBM PC and the Macintosh. It would be possible to write a version of Windows 95 which would run on a Macintosh (in fact there are programs available which simulate Windows 95 on Macintosh, thus allowing Win 95 programs to be run by Macintosh users).

User Interface Definition

User Interface: The aspects of a computer system or program which can be seen (or heard or otherwise perceived) by the human user, and the commands and mechanisms the user uses to control its operation and input data.

*A **graphical user interface** emphasises the use of pictures for output and a pointing device such as a mouse for input and control whereas a **command line interface** requires the user to type textual commands and input at a keyboard and produces a single stream of text as output.*
FOLDOC (Free OnLine Dictionary Of Computing).

Explanation

Whenever you use a program on a computer, there is a particular way in which you (the *user*) interact with the program. An example can best illustrate this.

Consider a simple program which will prints out the times tables. The computer program may request information (input) by displaying a question on the screen.

```
Please enter the times table you wish printed out [1..9 ]:
```

The user may respond by typing the number 7 on the keyboard. The program should accept this input and provide the following response:

```
The 7 times tables are as follows:
1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
```

This process of passing information back and forth between user and computer is achieved through the *user interface*. The term user interface is used to refer to both hardware and software, although the hardware aspects (e.g. screen and keyboard) are so standard that they are taken for granted (except by researchers developing hardware). When we refer to the User Interface, we are generally referring to the way in which the software (or program) has been written. The program defines the way in which the user and the computer interact. These programs use one of two different approaches, *graphical user interfaces* or *command line interfaces*.

Graphical User Interfaces (GUI):

This approach to the computer/user interaction uses pictures rather than just words to represent the input and output of a program. A GUI usually consists of icons, buttons, dialogue boxes, windows, and pull-down menus which are typically controlled by moving a pointer on the screen (usually in correlation to the user moving a mouse), and selecting the objects pointed to (typically by clicking the mouse button). The GUI was designed to make commands easy to remember by representing them in a graphical way (e.g. to move a file from one location to another, you click on the file, then physically move the mouse to the new location and release).

The GUI was invented with the work of Doug Engelbart in 1968, although the look and feel of the system we currently use was developed at Xerox's PARC (during 1970's). It was further refined and popularised by the Macintosh computer (in 1984), and finally brought to the masses using IBM PC's with Microsoft Windows 95 (in 1995).

Figure 8: A Typical Graphical User Interface

Command Line Interfaces (CLI):

A CLI provides the user with a means of communicating with the program solely by textual input. Commands are typically typed on a keyboard, and the results are displayed as text or graphics on the screen. This was the first common form of user interface. It generally provides greater flexibility than a GUI, but is more difficult to use. Examples of operating systems which use a CLI are: DOS, UNIX, Linux.

```
C:>_
```

Figure 9: An example of a Command Line Interface

Advantages and Disadvantages

The GUI is easier for a novice user. It is designed to be intuitive and natural to use. The visual information provides additional clues to enhance usability for a casual user. However, it is not as flexible as the CLI, which is usually faster and more powerful for an expert to use. Many people find the GUI allows them to quickly get started, but can often find it constrains them once they are familiar with the application. For this reason, experts are often frustrated with the limitations of the GUI.

Example:
Imagine that you are teaching a course at university (say, something like 415.111). You probably have to prepare handouts, overheads, labs and exams. It might be useful to include the paper number in the names of all the files, so that you remember what the documents are. You might end up with names like:

Lecture01_415111.doc
Lecture02_415.111.doc

and so on….

Now let us assume that you teach the same course, but in a different semester. It would be really useful to be able to rename all the files that end with 415111 to instead end with 415111FC so that they aren't confused with the second semester documents. Using Linux or DOS, this task could be achieved with a simple command. With a GUI OS like Windows 95/98/NT or MacOS, you would have to manually click on each filename and type a new name in. There is no way of automating even reasonably simple tasks in GUI operating systems. This is a huge limiting factor, which tends to make CLI based systems much more popular among experienced users.

File Management

You will need to use the operating system to manage your files. Every operating system handles files in a slightly different way, but the underlying principles remain the same. You must become proficient at manipulating files before you can use a computer successfully.

A File is a collection of data or information that has a name, called the filename. Almost all information stored in a computer must be in a file. There are many different types of files: data files, text files, program files, directory files, and so on. Different types of files store different types of information. For example, program files store programs, whereas text files store text.

PC Webopaedia (http://webopedia.internet.com)

Each file has a name (the filename) which is used to refer to that group of data. Most operating systems impose some restrictions on the characters which may be used in the name of a file (e.g. you are not allowed to use '/' or a space in an MS-DOS filename). In some systems, the length of the name is also limited.

Directories

Almost all operating systems today provide an abstract structure to help you manage your files. This structure is organised in a hierarchy, and is known as a hierarchical file structure (HFS). A special kind of file (known as a directory) is used to organise the various files stored in secondary storage devices. All files are arranged in a hierarchy, where the directories are able to "hold" or "contain" other files and directories. You can think of a directory as a container which is able to hold files and other directories. In practice, the directory stores a list of file names, along with additional information used by the operating system to locate and use the files. Filenames within a particular directory (or folder) must be unique, but files in different directories are permitted to have the same name.

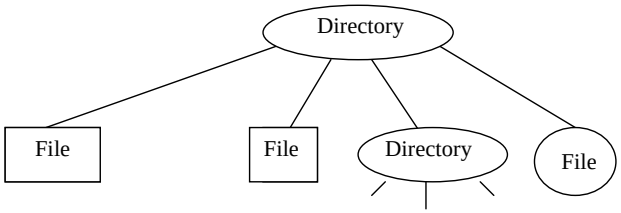

Figure 10: A Hierarchical File Structure