Unit 1

Computer Science as Applied Mathematics

Different Number Systems

The number system that we are used to is the decimal system. We are taught the decimal system from elementary school, but did you know that there are other number systems as well? In computer science, the three most important number systems are *decimal*, *binary*, and *hexadecimal*.

This section will be using the term **digit** to mean a single character (number, letter, symbol, etc.) that represents a certain quantity.

Example: The digit '3' may represent this quantity of dots: * * *.

The digit '§' may represent this quantity of dots: * * * *.

The digit 'F' may represent this quantity of dots: * * * * * * * * * * * * * * *.

Decimal Number System:	**Binary Number System**:	**Hexadecimal Number System**:
A number system of **base 10**, meaning that there are 10 individual **digits, 0-9**, that can be used to represent any number.	A number system of **base 2**, meaning that there are 2 individual **digits, 0 and 1**, that can be used to represent any number.	A number system of **base 16**, meaning that there are 16 individual **digits, 0-9 and A-F**, that can be used to represent any number.
Example: The number thirteen in decimal is represented using a 1 and a 3 (13).	*Example*: The number thirteen in binary is represented using only 1s and 0s (1101).	*Example*: The number thirteen in hexadecimal is represented by the digit D. (D)

In binary, a digit is called a **bit**. When there are four bits together, it is called a **nibble** or a **half-byte**. Since a 4-bit sequence is called a half-byte, it may be obvious that and 8-bit sequence is a **byte**.

Representing Numbers in Different Number Systems

When binary numbers get large, it can be hard to read. It is common practice to separate them into nibbles for readability. Start from the right and move left. Leading zeros may be added to the leftmost group to make it 4 bits long.

It isn't always clear what number system was used to write a number.

Example: "10" represents the quantity:

* * in binary (base 2)

* * * * * * * * * * in decimal (base 10)

* * * * * * * * * * * * * * * * in hexadecimal (base 16)

For this reason, we use a subscript to indicate which base a number is written in.

Example: We know that 10_2 is a binary number because of the subscript 2 following the number. We know that 10_{16} is a hexadecimal number because of the subscript 16. We know that 10_{10} is a decimal number because of the subscript 10.

When learning a new number system, it is helpful to start with learning to count in that system. We already know how to count in decimal, but what about binary and hexadecimal?

Fig. 1 shows how to represent the numbers (or how to count from) one through thirty in decimal, binary, and hexadecimal.

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	0001 0000	10
17	0001 0001	11
18	0001 0010	12
19	0001 0011	13
20	0001 0100	14
21	0001 0101	15
22	0001 0110	16
23	0001 0111	17
24	0001 1000	18
25	0001 1001	19
26	0001 1010	1A
27	0001 1011	1B
28	0001 1100	1C
29	0001 1101	1D
30	0001 1110	1E

Fig 1. A table of the numbers one through thirty represented in decimal, binary, and hexadecimal

Converting Base 2 numbers to Base 10 numbers

Take the rightmost digit of a binary number. Multiply it by 2^0. Take the next digit to the left and multiply it by 2^1. Continue with all of the digits, incrementing the power of 2 for each one. Now, add all of your answers together. That is the decimal representation of the binary number you started with.

Example: 00110101_2 to decimal

	0	0	1	1	0	1	0	1
=	$0(2^7)$ +	$0(2^6)$ +	$1(2^5)$ +	$1(2^4)$ +	$0(2^3)$ +	$1(2^2)$ +	$0(2^1)$ +	$1(2^0)$
=	0 +	0 +	32 +	16 +	0 +	4 +	0 +	1
=	**53_{10}**							

Converting Base 16 numbers to Base 10

Take the rightmost numeral of a hexadecimal number. Multiply it by 16^0. Take the next numeral to the left and multiply it by 16^1. Continue with all of the digits, incrementing the power of 16 for each one. For any letter numerals, rewrite them in their decimal form. A = 10, B = 11, C = 12, D = 13, E = 14, and F = 15. Now, add all of your answers together to get the decimal representation of the hexadecimal number you started with.

Example: $E4D1_{16}$ to decimal

	E		4		D		1
=	$E(16^3)$	+	$4(16^2)$	+	$D(16^1)$	+	$1(16^0)$
=	$14(16^3)$	+	$4(16^2)$	+	$13(16^1)$	+	$1(16^0)$
=	57344	+	1024	+	208	+	1
=	**58577_{10}**						

Modulus

Now we can represent binary and hexadecimal numbers in a number system that we understand more easily. It is also important to know how to represent our decimal numbers as binary and hexadecimal. To do this, we need to know the modulus operator. The modulus operator is an operator just like multiplication, division, addition, and subtraction.

The modulus operator is defined: x mod y = the remainder of x / y

Example:

4 mod 6	→	4 / 6 = 0 with **remainder 4**	→	4 mod 6 = 4
18 mod 6	→	18 / 6 = 3 with **remainder 0**	→	18 mod 6 = 0
9 mod 2	→	9 / 2 = 4 with **remainder 1**	→	9 mod 2 = 1

Converting Base 10 numbers to Base 2

Divide the starting number by 2 to get the quotient. Write the remainder is the rightmost digit of your answer. Next, divide the quotient you got by 2. Write the remainder to the left of the last remainder. When you get 0 as your quotient, stop here. Write the remainder as the leftmost digit in your answer.

Example: 38_{10} to binary

38 / 2 = **19** with **remainder 0**

19 / 2 = **9** with **remainder 1**

9 / 2 = **4** with **remainder 1**

4 / 2 = **2** with **remainder 0**

2 / 2 = **1** with **remainder 0**

1 / 2 = **0** with **remainder 1**

We got a quotient of 0 so we stop here. Starting from the top, we write the remainders from right to left. Our answer is **100110_2.**

Converting Base 10 numbers to Base 16

Divide the starting number by 16 to get the quotient. Make sure to note the remainder off to the side a little. Next, divide the quotient you got by 16 and note that remainder as well. Continue until you get a quotient of 0, and then note your last remainder. Rewrite each remainder that is more than 9 as it's hexadecimal form (e.q. rewrite 10 as A, 15 as F, etc.). Lastly, start from the bottom and go up writing the remainders from left to right to get your answer.

Example: 127_{10} to hexadecimal

127 / 16 = 7 with remainder 15 → remainder F

7 / 16 = 0 with remainder 7

We got a quotient of 0 so we stop here. We rewrite the remainders in hexadecimal if needed. Starting from the bottom line going up, we write each remainder from left to write. Our answer is **$7F_{16}$.**

How are Binary and Hexadecimal Related?

Each hexadecimal digit can be represented by a 4-bit binary sequence (e.g. $F_{16} = 1111_2$, $9_{16} = 1001_2$, etc.), and any half-byte can be represented by a single hexadecimal digit (e.g. $0001_2 = 1_{16}$, $1101_2 = D_{16}$, etc.).

One way to convert between binary and hexadecimal is to first convert the number to decimal and then convert the decimal number to the other number system. That can be time consuming so we have a few practical shortcuts that use the relation of binary and hexadecimal.

Converting Base 2 numbers to Base 16

Starting from the right side of the starting binary number, separate it into nibbles. Add leading zeros to the leftmost group if needed. Convert each nibble into decimal and then into hexadecimal. This will be much less complicated than what we saw earlier because the number can be no greater than fifteen.

Example: 10011011110010_2 to hexadecimal

0010 0110 1111 0010

First we separate the number into 4-bit groups. We added two leading zeros to the number to make every group 4 digits long.

2 6 15 2

We converted each binary number into its decimal equivalent.

2 6 F 2

We converted each decimal number into its hexadecimal equivalent.

Our answer is **$26F2_{16}$**.

Converting Base 16 numbers to Base 2

Separate the hexadecimal number into individual digits. Write each digit as its decimal equivalent and then write each decimal number as its binary equivalent. Add leading zeros to make each group 4-bits. Concatenate the numbers to get your answer.

Example: $AD4A_{16}$ to binary

A	D	4	A

First we separated the hexadecimal number into individual digits.

10	13	4	10

We converted each digit into its decimal form.

1010	1101	0100	1010

We converted each decimal number into its binary form and added leading zeros to each group if needed.

Our answer is **$1010\ 1101\ 0100\ 1010_2$**.

Why is Binary Used?

As previously mentioned, computers, are essentially millions of electrical circuits. These electrical circuits can exist in two states: powered and not powered. Since binary has two digits, it is suited to representing these two states very well.

Why is Hexadecimal Used?

Because hexadecimal codes are much simpler than binary, it is used by humans to shorten binary and make it more readable. Computers do not use hexadecimal; hexadecimal is translated into binary for computer use. Some common uses for hexadecimal are for colors, assembly language (we will learn about this later in Unit 3).

ASCII (American Standard Code for Information Interchange)

Computers see all data as 1's and 0's at the lowest level but is fundamentally necessary to represent text in computers. The problem then arises: how can text be represented with only numbers? The simplest method would be to just assign an arbitrary number to each letter.

In 1960, the American Standards Association began formal work on doing just that. In 1963, the first version of the ASCII character encoding scheme was published. It has gone through several revisions over the years but remains the base of most character encoding schemes today.

The original standard uses 7 bits to represent a character, giving a max of $2^7 = 128$ different characters under ASCII. Of the original 128 characters, only 94 are printable. 33 are used for control codes that are obsolete in today's computers.

Below is a table of the 128 ASCII characters.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

From this table, you can tell that the number that represents the character, 'A' is 65_{10} or 41_{16}.

Unicode

128 characters is grossly insufficient for languages like Chinese, which have 1000's of characters. The text-encoding standard that is most widely-used today is Unicode. ASCII is a subset of Unicode, meaning that all ASCII codes are also valid codes in Unicode.

Unicode specifies several different standards for encoding. The following are the most common:

- UTF-8
- UTF-16
- UTF-32

Each of these standards uses the respective number of bits to represent a single character, meaning UTF-32 has a theoretical maximum of $2^{32} = 4{,}294{,}967{,}296$ unique characters. In reality, the most recent version of Unicode as of 2017, Unicode 10.0, only specifies 136,755 unique characters.

The most popular version of Unicode is UTF-8, used on most webpages. It is a variable-length encoding scheme, meaning that it can use more 8 bits to represent a character when necessary.

Analyzing Data Collections

In the modern world, massive amounts of data are generated every second by users on the internet. It becomes of great interest for large companies to analyze this data to gain advantages over their competitors and improve their business' effectiveness.

Measuring Data

Before data can be analyzed, it is important to know how data is measured. Just like distance has inches, data has the bit. Below is a summary of the units of data.

Bit – a single 1 or 0

Nibble – 4 bits

Byte – 8 bits

Kilobyte – 1024 bytes (sometimes rounded to 1000) – 2^{10} bytes

Megabyte – 1024 Kilobytes (sometimes rounded to 1000) – 2^{20} bytes

Gigabyte – 1024 Megabytes (sometimes rounded to 1000) – 2^{30} bytes

Terabyte – 1024 Gigabytes (sometimes rounded to 1000) – 2^{40} bytes

Petabyte – 1024 Terabytes (sometimes rounded to 1000) – 2^{50} bytes

Exabyte – 1024 Petabytes (sometimes rounded to 1000) – 2^{60} bytes

Zettabyte – 1024 Exabytes (sometimes rounded to 1000) – 2^{70} bytes

Data Mining

As of the year 2018, your personal data is no longer your own. When you sign up to use services like Gmail, Facebook, Twitter, etc. you are agreeing to terms of services that often give Google, Apple, Microsoft, or other companies the right to collect your personal data and sell it to other companies. This process is known as data mining.

Most of the time, this process is benign and used for the purposes of advertising, but it does raise legitimate ethical questions about privacy that are still an area of political discussion.

Types of Data Collected

As of 5/19/2018, if you visit privacy.google.com/your-data.html, you can get a short summary of all the data google collects on you while you browse the web and use Google's services. The data mentioned on that page includes:

- Things you search for
- Websites you visit
- Videos you watch
- Ads you click on or tap
- Your location
- Device information
- IP address and cookie data
- Emails you send and receive on Gmail
- Contacts you add
- Calendar events
- Photos and videos you upload
- Docs, Sheets, and Slides on Drive
- Name
- Email address and password
- Birthday
- Gender
- Phone number
- Country

Each of these pieces of information collected are a unique type of data that can be analyzed in various ways.

Classification Analysis

Most email services have some sort of spam filter that allow users to mark an email as spam. The same spam email is typically sent to millions of users. Every time a user marks that email as spam, the spam filter takes notes unique patterns of text and other indicators that classify that email as spam. After a

few thousand users mark the email as spam, the spam filter program will have “learned” that any emails like it are more likely to be spam.

This process is not just an example of machine learning, but it is an example of how datamining emails that are sent and received through Gmail could potentially be used.

In general, classification analysis involves analyzing large sets of data in the hopes of classifying data points into different categories.

Outlier Analysis

If you have ever logged into your email in a different country or state that is geographically far away from where you usually log into it, you may have gotten a warning requiring you to enter a special code texted to you on your phone.

This is a result of outlier analysis. Google, for example, analyzes the IP Addresses that you typically log in from. Your IP Address typically can be used to get a rough estimate of the geographic region you are in. If Google notices an outlier in the data—that is an IP address that is associated with a geographic region that is different from the one you usually log in from, it will trigger an alarm. This could mean that someone in another country or state has your login credentials and is attempting to login. It could also mean you are traveling or on vacation.

In general, outlier analysis involves analyzing large sets of data for data points that are very far from the average. These data points can signal situations that require action—such as a person on the other side of the world logging into your email.

Other Forms of Data Analysis

There are other, more advanced types of data analysis, but classification and outlier analysis are two of the most common. Regression analysis is a type of statistical analysis that involves analyzing large sets of data to find rules that can be used to predict or forecast future trends. This type of analysis is used on everything from weather forecasting to stock-market speculations.

Data Analysis Techniques

When studying data analysis, there are a few ubiquitous techniques that are used to process massive sets of data.

Searching

Searching involves looking through a collection of data for a data point that matches a query.

Example: Find the user whose username is “jdoe.”

Searching is involved in nearly every data analyzing operation because to update a data point, you must first find where that data point is stored.

The study of how to efficiently search through massive collections of data is a classical problem in computer science. Some algorithms for searching are explored later in the course.

Sorting

Sorting is useful because it makes searching more efficient. Imagine how long it would take to look up a word in a dictionary if they were not sorted in alphabetical order!

The study of how to efficiently sort a list of items is also a classical problem in computer science.

Statistical Techniques

Statistical techniques like mean, median, standard deviation, variance, and others allow datasets to be mathematically analyzed and individual data points to be classified. Using statistical techniques can allow data points to be classified as outliers or classified into different groups.

Example: A company has been suffering from cyber attacks that attempt to access secure servers on its network. There is a good deal of legitimate traffic to the network that follows regular patterns (e.g. time of day, length of session, login attempts, etc.)

Wanting to tighten security, the company’s security administrator configures a program to run statistical analysis on the network traffic and alert him to any traffic that is statistically different than legitimate traffic.