COSC 122
Computer Fluency

Information Representation

Dr. Ramon Lawrence
University of British Columbia Okanagan
ramon.lawrence@ubc.ca

Survey
Reading the Notes

Question: HONESTLY, how often do you read the notes before class?

A) never
B) up to 25% of the time
C) up to 50% of the time
D) all the time
E) This class has notes?

Survey
Class Still Easy?

Question: HONESTLY, rate the course difficulty so far from 1 (easy) to 5 (difficult).

A) easy
B) below normal
C) normal
D) above normal
E) difficult

Key Points

1) Representing data digitally means to represent it using discrete units.

2) The lowest level of data representation on a computer is a single bit that represents either 0 or 1.

3) Bits are combined to allow more information to be represented including characters and numbers.

4) More complex information like documents, spreadsheets, and databases (all of which we will see later) are simply compositions and higher-level abstractions of bits.

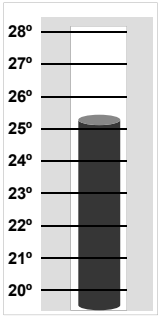
Everything is digital - Is that good?

Almost all of our music, movies, data, and pictures are digital.
- Most people believe digital is better. What does digital mean?

Representing something ***digitally*** means to store the data in discrete units. A unit is ***discrete*** if it is distinct or separate from other units. The smallest unit of data depends on what we are representing.

Digital differs from ***analog*** where the information is encoded on a continuous signal (spectrum of values).
- Note that sound and images are analog by nature.

Analog versus Digital
Thermometer Example



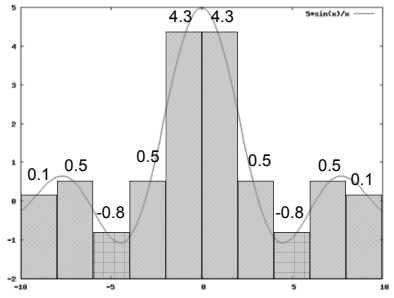
A thermometer contains mercury which expands and contracts in response to temperature changes.

The mercury level is analog, and its expansion continuous over the temperature range.

By adding marks and units to the thermometer, we are digitizing the information.

Conversion from Analog to Digital

How would you digitize this analog data into 10 discrete points?



Why are electronics digital?

1) Computers are digital and many home electronics are interfacing with computers.

2) Analog signals are more susceptible to noise that degrades the quality of the signal (sound, picture, etc.). The effect of noise also makes it difficult to preserve the quality of analog signals across long distances.

3) Reading data stored in analog format is susceptible to data loss and noise. Copying analog data leads to declining quality.

Digitizing Discrete Information Phone Numbers

A simple example of digital data is a phone number. A phone number consists of multiple units of information called digits (the numbers 0 through 9).

Although numbers are used to represent the values of different digits, it is possible to use any collection of 10 distinct symbols to represent the 10 possible different values.

However, using numbers is nice because they have a natural ordering (0 < 1 < 2 < 3 < … < 9).

Digitizing Discrete Information Phone Numbers

Standard Phone Keys	Phone Keys (with shift)	Phone Keys as Musical Symbols
1 2 3	! @ #	▶ ▼ ◀
4 5 6	$ % ^	▶▶ ‖ ◀◀
7 8 9	& * (	♫ ▲ ♯
0	)	■

Question: Represent the phone number 254-123-6789 using both alternative digitization methods.

Digital Phone Numbers

Question: Using the symbol encoding for phone numbers, what is this number: `$#% - **()`

A) 615 - 8809

B) 435 - 8800

C) 453 - 8899

D) 435 - 8890

1 2 3	! @ #	
4 5 6	$ % ^	
7 8 9	& * (	
0	)	

Encoding Information with Dice

We will see how much information we can encode using six-sided dice.

Quick question: If a dice has six unique sides, how many different values/states can it encode?

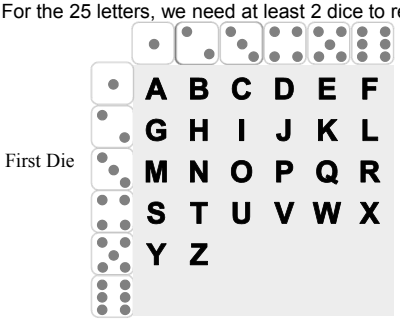
Answer: 6

By using more dice, we can encode more data:

1 die = 6 states
2 dice = 6 x 6 = 36 states
3 dice = 6 x 6 x 6 = 216 states
N dice = 6^N states

Encoding Information with Dice (2)

For the 25 letters, we need at least 2 dice to represent a symbol:



Question: Spell your name using our dice representation.

Encoding Information with Dice (3)

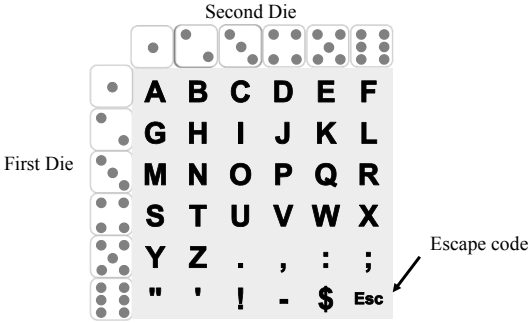
The extra 10 states could be used to encode numbers. However, what if we need to encode other symbols as well?

One solution is to use 3 dice per symbol which gives us 216 possible symbols.

Another way is to have one special symbol be an escape character. It does not match any legal character, so it will never be needed for normal text digitization. An escape character indicates that the digitization is "escaping from the basic representation" and applying a secondary representation.

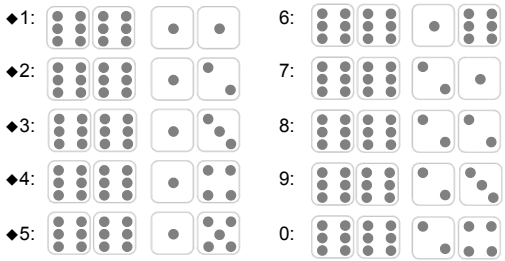
Question: What escape character have we already seen and in what context?

Encoding Information with Dice (4)



Encoding Information with Dice (6)

We will use the escape and two dice to represent numbers:



Question: How would we encode the number 198 in this notation?

Representing Data using Dice

Question: Using the dice encoding, what is this:

A) SIMYR

B) DICER

C) DICYR

D) SNMYO

Aside: The Time versus Space Tradeoff

A fundamental challenge in computer science is encoding information efficiently both in terms of space and time.

- We just saw an example where we could save space (only need 2 dice instead of 3) by using the escape symbol.

At all granularities (sizes) of data representation, we want to use as little space (memory) as possible. However, saving space often makes it harder to figure out what the data means (think of compression or abbreviations). In computer terms, the data takes longer to process.

The ***time versus space tradeoff*** implies that we can often get a faster execution time if we use more memory (space). Thus, we must strive for a balance between time and space.

Representing Binary Data

Data is information before it has been given any context, structure and meaning.

Binary data has two states and is represented in a computer using a bit. A ***bit*** can either be 0 or 1.

- The word bit is short for "binary digit".

A computer memory consists of billions of bits which allows for an almost limitless number of possible states.

What is a Byte?

A ***byte*** is a sequence of 8 bits.

Historical note: Byte is spelled with a "y" because engineers at IBM were looking for a word for a quantity of memory between a bit and a ***word*** (usually 32 bits). Bite seemed appropriate, but they changed the "i" to a "y", to minimize typing errors.

Converting Binary to Decimal

To convert a binary number *B* to a decimal number *D*:

Let *B* have *n* bits of the form $b_{n-1}b_{n-2}\ldots b_3b_2b_1b_0$ then

$D = b_{n-1}*2^{n-1} + b_{n-2}*2^{n-2} + \ldots + b_3*2^3 + b_2*2^2 + b_1*2^1 + b_0*2^0$

Base 10 (decimal) example:

- $765 = 7 * 10^2 + 6 * 10^1 + 5 * 10^0$

Example: binary value is 10010111

- $= 1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$
- = 151

Question:

1) Compute the decimal value of 1011.

2) Compute the decimal value of 00101010.

Converting Decimal to Binary

To convert a decimal number *D* to a binary number *B*:

- Repeat until *D* = 0
  - ⇨IF D is odd THEN append a 1 bit to the front of B
  - ⇨ELSE IF D is even THEN append a 0 bit to the front of B
  - ⇨Set D equal to D / 2

Example: Decimal value of D = 19

- 19 is odd — B = 1
- 9 is odd — B = 11
- 4 is even — B = 011
- 2 is even — B = 0011
- 1 is odd — B = 10011

Question: Compute the binary value of 115.

Aside: Adding Binary Numbers

Just like regular addition, we can add binary numbers. The rules are the same:

- Work from right to left, adding corresponding digits in each place position.
- If adding the two digits is bigger than the maximum digit value (9 in base 10 and 1 in base 2), we carry to the next position.

Example:

```
          1 1      (carries)
  1 0 0 1 0 1 1 1
+ 0 1 1 0 0 1 1 0
-----------------
  1 1 1 1 1 1 0 1
```


Hex Explained

Previously we specified custom colors in HTML using hex digits

- e.g., <font color ="#FF8E2A">
- *Hex* is short for hexadecimal (base 16)

We use hex as it is easier than writing sequences of bits. Each hex digit corresponds to a 4-bit sequence.

- e.g. 1011 (binary) = 11 (decimal) = B (hexadecimal)

Question:

Convert this binary sequence to hexadecimal:

0000 0101 1000 0001 1111 1110

Decimal to Binary to Hex Conversion Table

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**

Review Binary to Decimal

Question: Convert this binary number to decimal: **01001111**.

A) 143

B) 78

C) 79

D) 47

Review Decimal to Binary

Question: Convert this decimal number to binary: **123**.

A) 1011011

B) 1111011

C) 11111011

D) 1110011

Review Binary to Hexadecimal

Question: Convert this binary number to hexadecimal:

0111 1000 1111 1110 1001

A) 78ACD

B) 58FED

C) 78FE9

D) 78FFD

Review Questions Decimal to Binary to Hexidecimal

1) Convert 163 (decimal) to binary and hexadecimal.

2) Covert 10101010 to decimal and hexadecimal.

3) Convert EF (hexadecimal) to binary and decimal.

Representing Characters using Bits

In total, there are 95 basic character symbols which would require 7 bits to encode.

- 26 uppercase and 26 lowercase Roman letters, 10 Arabic numerals, 10 arithmetic characters, 20 punctuation characters, and 3 non-printable characters (tab, backspace, new line).

The standard 7-bit code for characters is called ***ASCII*** *(American Standard Code for Information Interchange).*

- Later, the ASCII code was extended (*extended ASCII*) to 8 bits to handle additional characters.

Just like the dice encoding, each 8-bit sequence maps to a particular character. We use an ASCII table to determine what each bit sequence means.

ASCII Table

First 4 bits (most significant)	Next 4 bits (least significant) 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0000	␀	␁	␂	␃	␄	␅	␆	␇	␈	␉	␊	␋	␌	␍	␎	␏
1	0001	␐	␑	␒	␓	␔	␕	␖	␗	␘	␙	␚	␛	␜	␝	␞	␟
2	0010		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	0110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	0111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	␡
8	1000	␀	␁	␂	␃	␄	␅	␆	␇	␈	␉	␊	␋	␌	␍	␎	␏
9	1001	␐	␑	␒	␓	␔	␕	␖	␗	␘	␙	␚	␛	␜	␝	␞	␟
A	1010	␠	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	1011	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	1100	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	1101	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	1110	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	1111	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Question: Represent the phone #: 254-123-6789 using ASCII.

Representing Text Beyond ASCII - Unicode

Although ASCII is suitable for English text, many world languages, including Chinese, require a larger number of symbols to represent their basic alphabet.

The ***Unicode standard*** uses patterns of 16-bits (2 bytes) to represent the major symbols used in all languages.

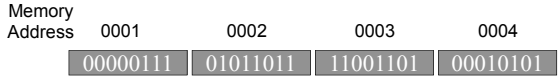
- First 256 characters exactly the same as ASCII.
- Maximum # of symbols: 65,536.

Representing Data in Memory Integers

A integer is a whole number. It is encoded in a computer using a fix sized number of bits (usually 32).

- The first bit is a sign bit (0=positive, 1=negative).
- Negative numbers are represented in *two's complement notation*. The "largest" bit pattern FFFFFFFF is -1.

Example: 123,456,789 as a 32-bit integer:



Representing Data in Memory Doubles and Floats

A number with a decimal may be either stored as a *double* or *float* value. On 32-bit machines, a double is usually 8 bytes long.

- ⇨A float is normally half the size of a double value and has less precision.

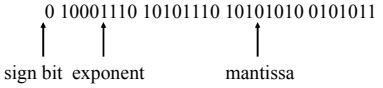
Double values are stored using a ***mantissa*** and an ***exponent***:

- Represent numbers in scientific format: $N = m * 2^e$
  - ⇨m - mantissa, e - exponent, 2 - radix
  - ⇨Note that converting from base 10 to base 2 is not always precise, since real numbers cannot be represented precisely in a fixed number of bits.
- There are many standards for representing numbers in a fixed number of bits. The most common is **IEEE 754 Format**:
  - ⇨32 bits - 1-bit sign; 8-bit exponent; 23-bit mantissa
  - ⇨64 bits - 1-bit sign; 11-bit exponent; 52-bit mantissa

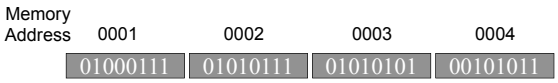
Representing Data in Memory Doubles (2)

The number 55,125.17 stored as 4 consecutive bytes is:

- Hexadecimal value is: 4757552B Stored value is: 55125.168



- Divided into bytes looks like this:



Aside: Can you really get rich by stealing fractions of a penny?

Have you ever seen a movie (e.g. Office Space) where the plot was to steal fractions of a penny lost due to rounding?

Can that really happen?

- Called ***salami slicing*** as stealing money in very small quantities by always rounding down fractions of a penny.

Consider the salary in the previous example: $55,125.17 that had an actual value of 55,125.16**8** where stored in the computer.

- That imprecision can be serious when we are talking about millions of numbers and operations.
- Idea: Round ***down*** to 55,125.16 and take the extra penny

Good code would not store monetary values as doubles because they are imprecise or make sure to round appropriately.

Representing Data in Memory Strings from Characters

A ***string*** is a sequence of characters allocated in consecutive memory bytes.

The first character of the string is at the first location of memory. The last character can be known by either:

- ***Null-terminated string*** - last byte value is 0 to indicate end of string.
- ***Byte-length string*** - length of string in bytes is specified (usually in the first few bytes before string starts).

Representing Data in Memory Dates

A ***date*** value can be represented in multiple ways:

- Integer representation - number of days past since a given date
  - ⇨Example: # days since Jan 1, 1900
- String representation - represent a date's components (year, month, day) as individual characters of a string
  - ⇨Example: YYYYMMDD or YYYYDDD
  - ⇨Please do not reinvent Y2K by using YYMMDD!!

A ***time*** value can also be represented in similar ways:

- Integer representation - number of seconds since a given time
  - ⇨Example: # of seconds since midnight
- String representation - hours, minutes, seconds, fractions
  - ⇨Example: HHMMSSFF

Encoding Higher-Level Information

We have seen how we can encode characters, numbers, and strings using only sequences of bits (and translation tables).

The documents, music, and videos that we commonly use are much more complex. However, the principle is exactly the same. We use sequences of bits and ***interpret*** them based on the ***context*** to represent information.

As we learn more about representing information, always remember that everything is stored as bits, it is by interpreting the context that we have information.

Encoding an HTML Document

Here is our first HTML document:

```
<HTML><HEAD><TITLE>Hello World using HTML</TITLE></HEAD>
<BODY>
<P>Hello world!</P>
</BODY></HTML>
```

Here is its hexadecimal encoding:

```
3C 48 54 4D 4C 3E 3C 48 45 41 44 3E 3C 54 49 54 4C 45 3E
48 65 6C 6C 6F 20 57 6F 72 6C 64 20 75 73 69 6E 67 20 48
54 4D 4C 3C 2F 54 49 54 4C 45 3E 3C 2F 48 45 41 44 3E 0A
3C 42 4F 44 59 3E 0A 3C 50 3E 48 65 6C 6C 6F 20 77 6F 72
6C 64 21 3C 2F 50 3E 0A 3C 2F 42 4F 44 59 3E 3C 2F 48 54
4D 4C 3E
```

Some key hex digits: 3C = "<" 3E = ">" 20 = space 2F = "/" 0A = new line

Encoding Higher-Level Information (2)

Note that the tag instructions to HTML are encoded in ASCII characters just like the text of the document. However, when the web browser processes the document they are treated as the special instructions that they are.

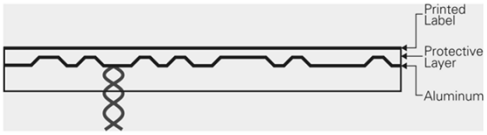
What we have is ***layers of abstraction*** or context to the bit sequence:

- Raw data – sequence of bits (or hexadecimal digits)
- Character level – Each 8 bit sequence represents a character encoded using ASCII.
- Document level – The document consists of text and tags. Tags are instructions to tell the browser how to display the document.

Aside: Encoding Data on CDs and DVDs

How the present and absent states of bits are encoded depends on the medium on which the information is stored.

A CD consists of several different material layers. In one of those layers, indentations (or ***pits***) are created. Areas between pits are called ***lands***. The transition between a pit to a land represents 1 and no change represents 0.



- DVDs store more information as they have smaller pit sizes and more tracks (smaller distance between tracks).

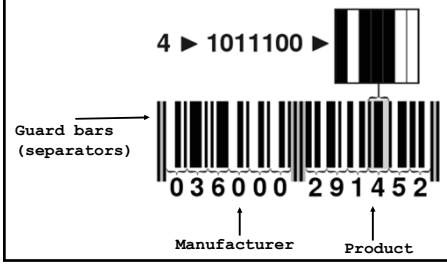
Aside: How do CD-R and CD-RW work?

The medium for encoding is different for CD-R and CD-RW.

- CD-R/DVD-R – use *photosensitive dye* and are initially "blank". The write-laser of a CD writer changes the color of the dye at desired locations to make the CD appear to have pits and lands.
  - ⇨ Note that the dye will fade over time causing read errors.
- CD-RW/DVD-RW – are re-recordable by using a *metallic alloy* that has its reflectivity changed by the heat of the write laser.
  - ⇨ There is not as great a difference in lands and pits with CD-RW, hence they sometimes are not readable by all players.

UPC Barcodes

Universal Product Codes (UPC) encode manufacturer on left side and product on right side. Each digit uses 7 bits with different bit combinations for each side (can tell if upside down).



QR Codes

A **QR** (**Q**uick **R**esponse) code is a 2D optical encoding developed in 1994 by Toyota with support for error correction.

Hello World!

First Page of Syllabus

Make your own codes at: www.qrstuff.com.

NATO Broadcast Alphabet

The code for broadcast communication is purposefully inefficient, to be distinctive when spoken amid noise.

A	Alpha	J	Juliet	S	Sierra
B	Bravo	K	Kilo	T	Tango
C	Charlie	L	Lima	U	Uniform
D	Delta	M	Mike	V	Victor
E	Echo	N	November	W	Whiskey
F	Foxtrot	O	Oscar	X	X-ray
G	Golf	P	Papa	Y	Yankee
H	Hotel	Q	Quebec	Z	Zulu
I	India	R	Romeo		

Question: Pick a partner. Pretend to be a pilot and broadcast your name to your partner using the NATO broadcast alphabet.

Conclusion

The ability to ***represent information*** is fundamental to the functions of a computer system.

There are multiple ways to represent information, the most basic of which is the presence and absence of information. A bit, which has the values 0 or 1, are used in computers.

Sequences of bits are combined to represent characters, numbers, and other data items. Larger data items are produced by combining these basic units.

Bits are just data until the necessary context is provided. There may be multiple levels of context (***abstraction***) needed to understand the meaning of a bit sequence.

Objectives

- Compare and contrast: digital versus analog
- Give one reason why electronics are increasing digital.
- Explain how we can encode states and characters using dice.
- Explain the usefulness of the escape symbol.
- Define: data, bit, byte, word
- Convert from decimal to binary and binary to decimal.
- Convert from binary to hexadecimal and hexadecimal to binary.
- Explain why ASCII table is required for character encoding.
- Convert characters to binary using ASCII table.
- Briefly explain how integers, doubles, and strings are encoded.
- Encode using the NATO broadcast alphabet.
- Explain why context and interpretation produces information from data.