Lecture 4: Bits, Bytes, Binary

continued

Using bits to represent information

- AB / BSE
  - 1 bit
- Fr / So / Jr / Sr
  - 2 bits
- grads, auditors, faculty as well
  - 3 bits
- a unique number for each person in 109
  - 6 bits
- a unique number for each freshman at PU
  - 11 bits
- a unique number for each PU undergrad
  - 13 bits

Powers of two, powers of ten

1 bit = 2 possibilities

2 bits = 4 possibilities

3 bits = 8 possibilities

...

n bits = 2^n possibilities

2^{10} = 1,024 is about 1,000 or 1K or 10^3

2^{20} = 1,048,576 is about 1,000,000 or 1M or 10^6

2^{30} = 1,073,741,824 is about 1,000,000,000 or 1G or 10^9

the approximation is becoming less good

but it's still good enough for estimation

- terminology is often imprecise:
  - " 1K " might mean 1000 or 1024 (10^3 or 2^{10})
  - " 1M " might mean 1000000 or 1048576 (10^6 or 2^{20})

Bytes

- **"byte" = a group of 8 bits treated as a unit**
  - on modern machines, the fundamental unit of processing and memory addressing
  - can encode any of 2^8 = 256 different values, e.g., numbers 0 .. 255 or a single letter like A or digit like 7 or punctuation like $
    - ASCII character set defines values for letters, digits, punctuation, etc.
- **group 2 bytes together to hold larger entities**
  - two bytes (16 bits) holds 2^{16} = 65,536 values
  - a bigger integer, a character in a larger character set
    - Unicode character set defines values for almost all characters anywhere
- **group 4 bytes together to hold even larger entities**
  - four bytes (32 bits) holds 2^{32} = 4,294,967,296 values
  - an even bigger integer, a number with a fractional part (floating point), a memory address
  - current machines use 64-bit integers and addresses (8 bytes)
    - 2^{64} = 18,446,744,073,709,551,616

- **no fractional bytes: the number of bytes is always an integer**

Interpretation of bits and bytes depends on context

- meaning of a group of bits depends on how they are interpreted
- 1 byte could be
  - 1 bit in use, 7 wasted bits (e.g., M/F in a database)
  - 8 bits representing a number between 0 and 255
  - an alphabetic character like W or + or 7
  - part of a character in another alphabet or writing system (2+ bytes)
  - part of a larger number (2 or 4 or 8 bytes, usually)
  - part of a picture or sound
  - part of an instruction for a computer to execute
    - instructions are just bits, stored in the same memory as data
    - different kinds of computers use different bit patterns for their instructions
      - laptop, cellphone, game machine, etc., all potentially different
  - part of the location or address of something in memory
  - ...
- one program's instructions are another program's data
  - when you download a new program from the net, it's data
  - when you run it, it's instructions

Some things are intrinsically discrete / digital

- another kind of conversion
  - letters are converted into numbers when you type on a keyboard
  - the letters are stored (a Word document), retrieved (File/Open...), processed (paper is revised), transmitted (submitted by email), printed on paper
- letters and other symbols are inherently discrete
- encoding them as numbers is just assigning a numeric value to each one, without any intrinsic meaning
- what letters and other symbols are included?
- how many digits/letter?
  - determined by how many symbols there are
  - how do we disambiguate if symbols have different lengths?
- how do we decide whose encoding to use?
- the representation is arbitrary
- but everyone has to agree on it
  - if they want to work together

ASCII: American Standard Code for Information Interchange

- an arbitrary but agreed-upon representation for USA
- widely used everywhere

32	space	33	!	34	"	35	#	36	$	37	%	38	&	39	'
40	(	41	)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[	92	\	93	]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124	|	125	}	126	~	127	del

00010000 space 00010001 ! 00010010 " 00010101 # ...

Hexadecimal notation

- binary numbers are bulky
- hexadecimal notation is a shorthand
- it combines 4 bits into a single digit, written in base 16
  - a more compact representation of the same information
- hex uses the symbols A B C D E F for the digits 10 .. 15

0 1 2 3 4 5 6 7 8 9 A B C D E F

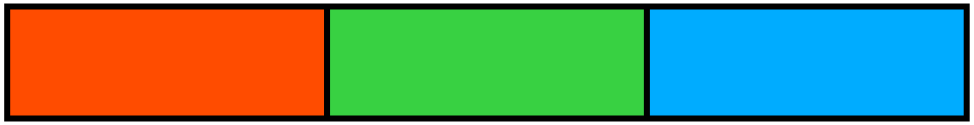
0	0000	1	0001	2	0010	3	0011
4	0100	5	0101	6	0110	7	0111
8	1000	9	1001	A	1010	B	1011
C	1100	D	1101	E	1110	F	1111

ASCII, using hexadecimal numbers

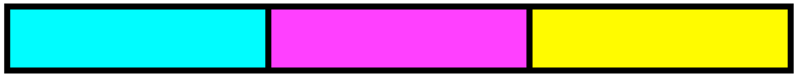
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Color

- TV & computer screens use Red-Green-Blue (RGB) model



- each color is a combination of red, green, blue components
  - R+G = yellow, R+B = magenta, B+G = cyan, R+G+B = white
- for computers, color of a pixel is usually specified by three numbers giving amount of each color, on a scale of 0 to 255
- this is often expressed in hexadecimal so the three components can be specified separately (in effect, as bit patterns)
  - 000000 is black, FFFFFF is white
- printers, etc., use cyan-magenta-yellow[-black] (CMY[K])



A very important idea

- **number of items and number of digits are tightly related:**
  - one determines the other
  - maximum number of different items = base $^{\text{number of digits}}$
  - e.g., 9-digit SSN: 10^9 = 1 billion possible numbers

  - e.g., to represent up to 100 “characters”: 2 digits is enough
  - but for 1000 characters, we need 3 digits

  - the same for bits: 9 bits can represent up to 2^9 = 512 items

- **interpretation depends on context**
  - without knowing that, we can only guess what numbers mean

Things to remember

- **digital devices represent everything as numbers**
  - discrete values, not continuous or infinitely precise
- **all modern digital devices use binary numbers (base 2)**
  - instead of decimal (base 10)
- **it's all bits at the bottom**
  - a bit is a "binary digit", that is, a number that is either 0 or 1
  - computers ultimately represent and process <u>everything</u> as bits
- **groups of bits represent larger things**
  - numbers, letters, words, names, pictures, sounds, instructions, ...
  - the interpretation of a group of bits depends on their context
  - the representation is arbitrary; standards (often) define what it is
- **the number of digits used in the representation determines how many different things can be represented**
  - number of values = base $^{\text{number of digits}}$
  - e.g., 10^2, 2^{10}