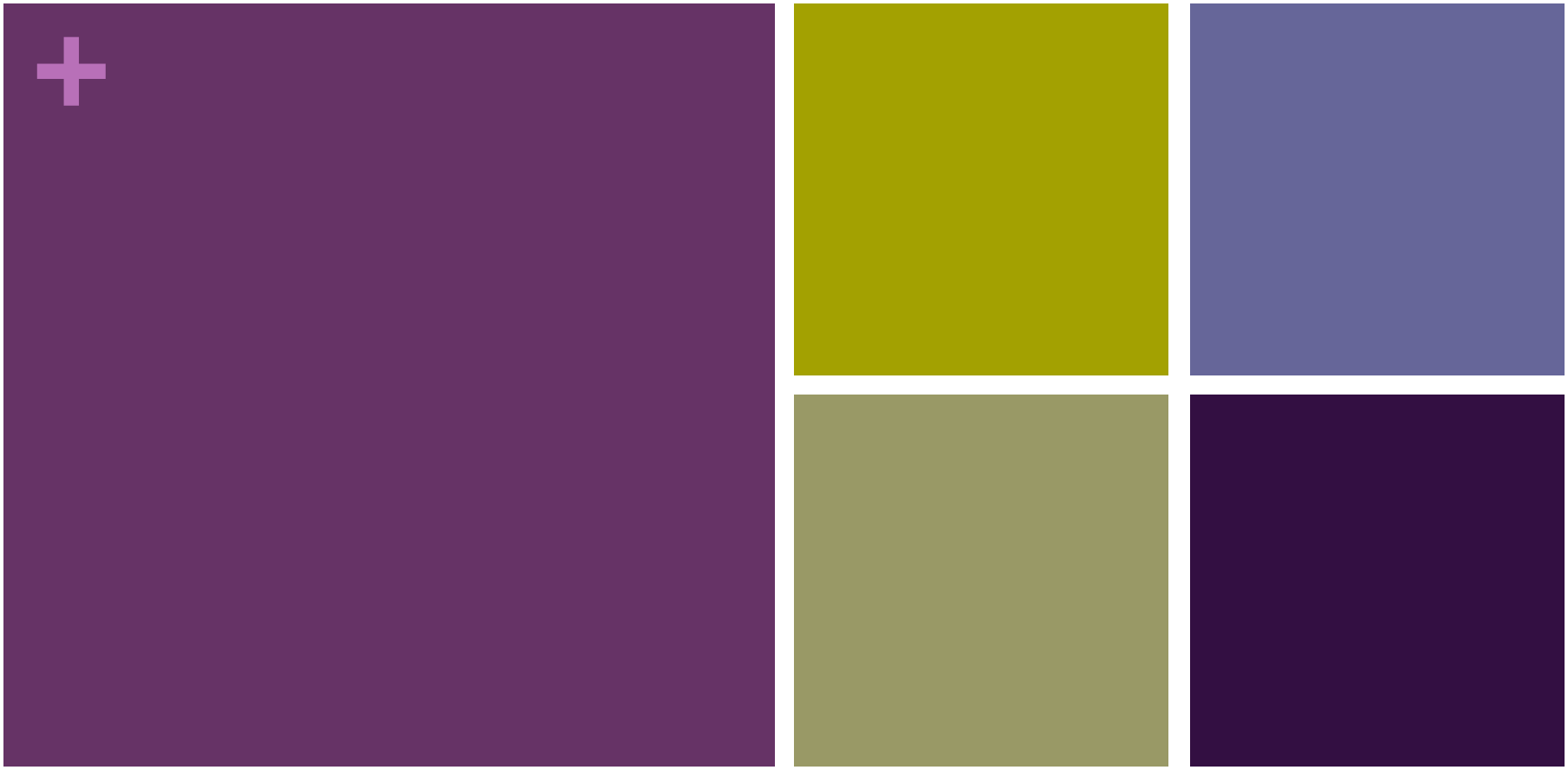


Computers, Programming Languages and How We Got Here

CSCI-UA.0002 - 002

Hi, I'm Julie Lizardo (she/hers)

- Adjunct Professor
- MPS from NYU Interactive Telecommunications Program
- Interested in exploring the intersection of art and technology

itp
NYU | TISCH

NYU | COLLEGE OF ARTS & SCIENCE

General Safety

- Keep your masks on at all times (covering your nose and mouth)
- No eating or drinking in the classroom (water is okay)
- Be respectful of one another

Who are you?

www.pollev.com/python002

Course Site

https://cs.nyu.edu/courses/summer22/CSCI-UA.0002-002/

+

Let's get started!

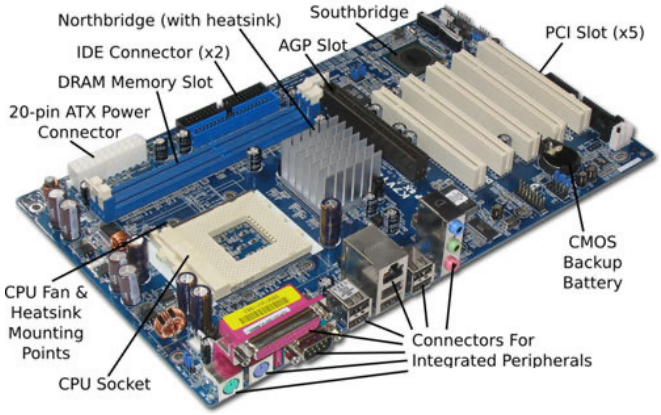
What is a Computer?

- A machine that processes information based on a program

Are laptops and desktops the only devices that qualify as computers?

Computers are Everywhere!

What makes a computer?



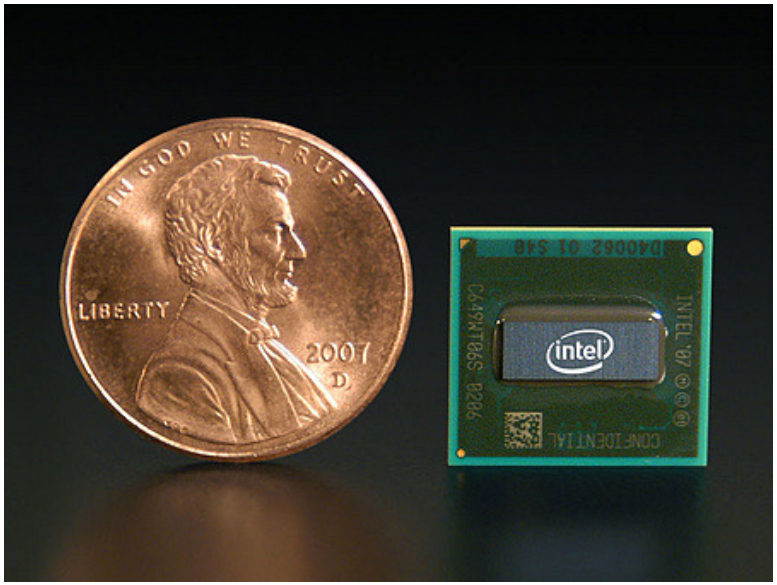

```
particle [] p = new particle [10];
void setup(){
  for(int i = 0; i < p.length; i++){
    p[i] = new particle(random(400),random(400),random(10));
  }
  size (400,400);
  smooth();
}
void draw(){
  background (255);
  for(int i = 0; i<p.length; i++){
    p[i].gmove();
    p[i].draw();
  }
  line(mouseX,0,mouseX,height);
}
```

Hardware

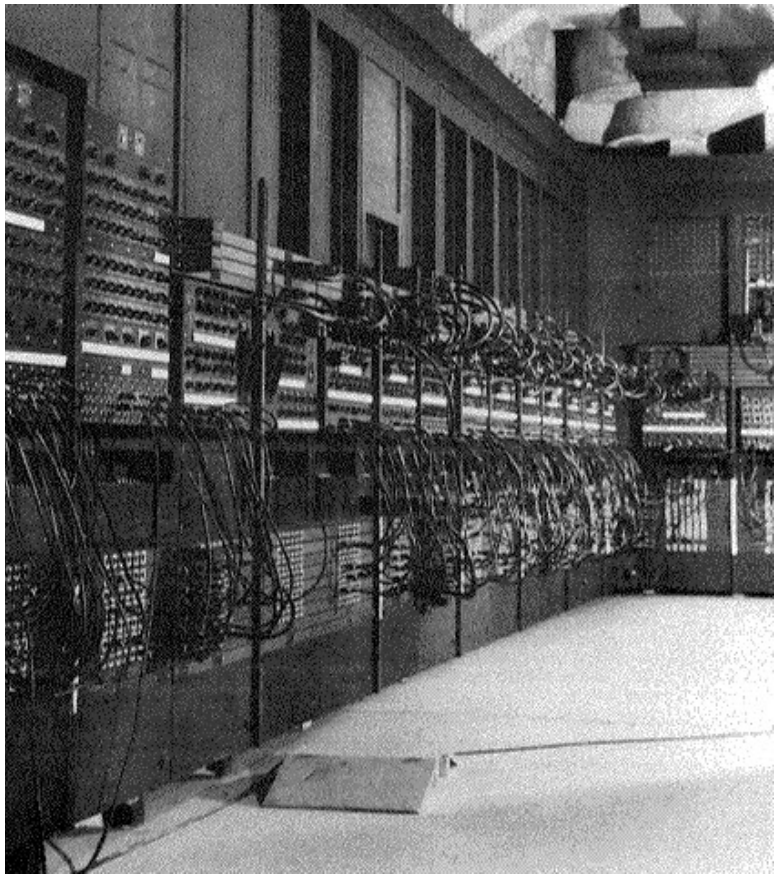
"The Guts"

The Processor (CPU)

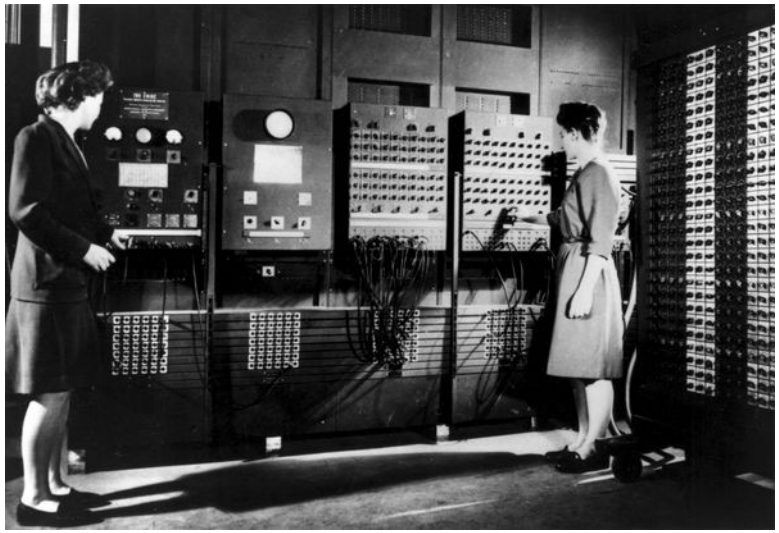
- Every computer has at least one Processor
- The processor acts as the computer's "brain" and coordinates all activity
- As far as a brain goes, the processor isn't very bright – it can only perform four distinct actions:
  - Receive a new instruction (Fetch)
  - Make sense of this instruction (Decode)
  - Perform the action defined by this instruction (Execute)
  - Store the result of the action (Store)
- These processors, sometimes called a CPU or "Central Processing Unit," are made up of miniaturized transistors and circuits on a semiconductor.



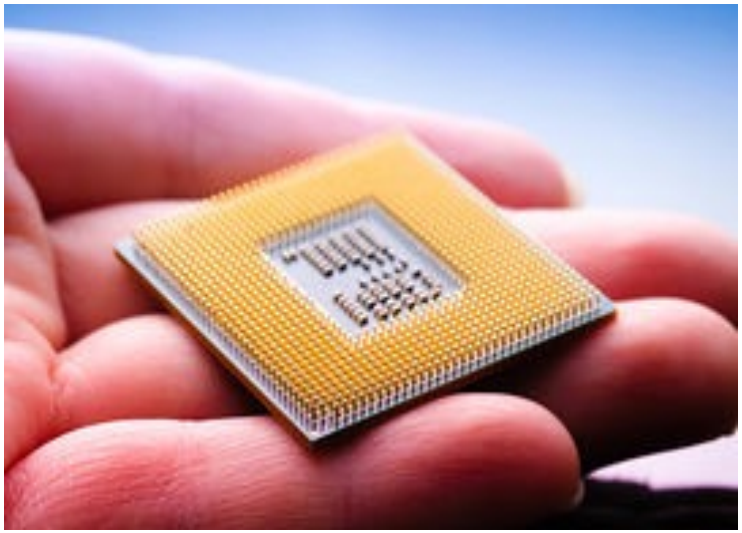
The ENIAC (1945)



Then vs Now

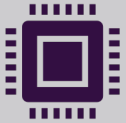


**ENAIC,
1945**



**Microprocessor,
Today**

Moore's Law



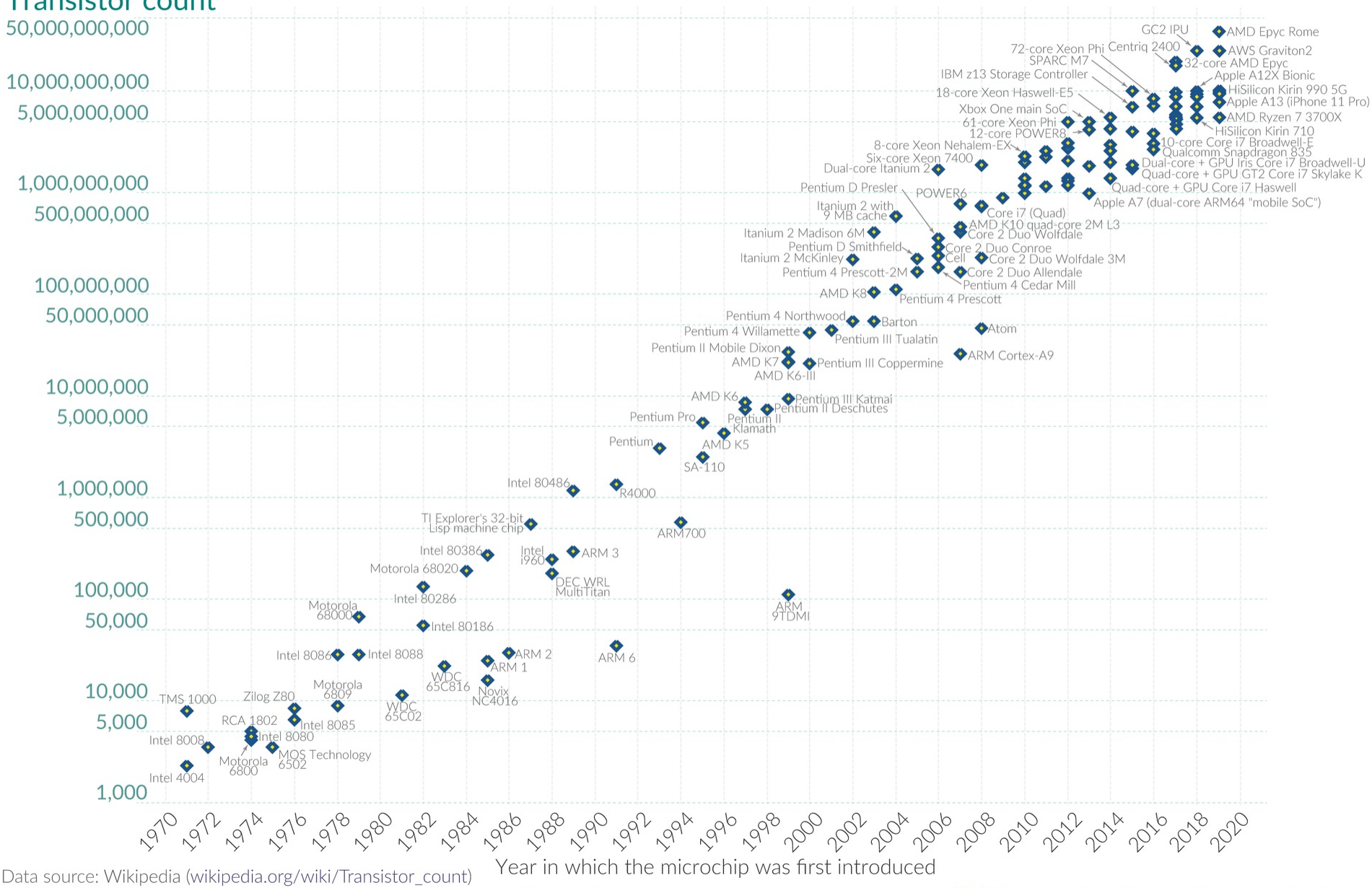
Advances in chip design have facilitated these technologies thanks to Moore's Law.

The empirical observation that at our rate of technological development, the complexity of an integrated circuit, with respect to minimum component cost will double in about 24 months.

Moore's Law: The number of transistors on microchips doubles every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)

Memory

- Computers, like us, need short term memory in order to function properly
- We call this RAM or "Random Access Memory"
- It's very fast!
- We call it "Random" since the computer can selectively read and write to it's RAM at will – (think of accessing any song on a CD versus accessing that same song on a cassette tape)
- When a computer turns off, it loses its short term memory (we call this volatility)
- In general, the more RAM a system has, the better it performs.

Storage

- Temporary storage is accomplished in memory (RAM)
- We use RAM for short term memory because it is very fast, but it is also volatile and loses information if it is denied of electricity
- Long term storage is, by definition, non-volatile – it has the ability to persist even if the power is off

Input Devices

- Input devices allow us to communicate "real world" information into a format that the computer can understand

Output Devices

- Output devices translate the internal workings of your computer into "real world" stimulus

+

Software

"The Ideas"

What is Software?

- Software is, in essence, a set of instructions that tell our computers how to behave.
- Software is fluid, and can easily be changed up updated
- Hardware is rigid, and cannot perform that were not originally planned for during the device's design

The Operating System

- Specialized software that coordinate all activities among hardware
- Contains instructions for running application software
- Also know as a "platform" or "software platform"
- programs that run on different operating systems are know as "cross platform" applications.
- The OS is the internal "Traffic Cop" of your computer

The User Interface

- Portion of System Software that allows you to interact with data
- Two types
  - Graphical (GUI)
  - Command Line
- GUI is more user-friendly, but command line is faster.

Application Software

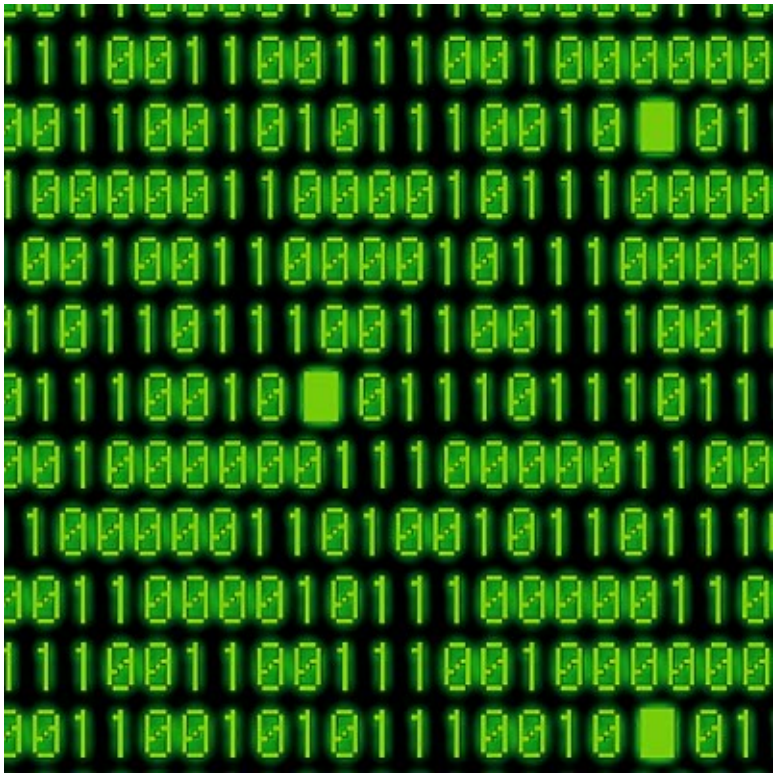
- Software that serves to help "users more productive and/or assist them with personal tasks"
- Application Software is described as a set of programs that are designed to perform specific tasks for the user.
- Categories (very loose grouping)
  - Productivity
  - Graphic Design / Multimedia
  - Home / Personal / Educational
  - Communications
  - … + many others

+

Data Storage

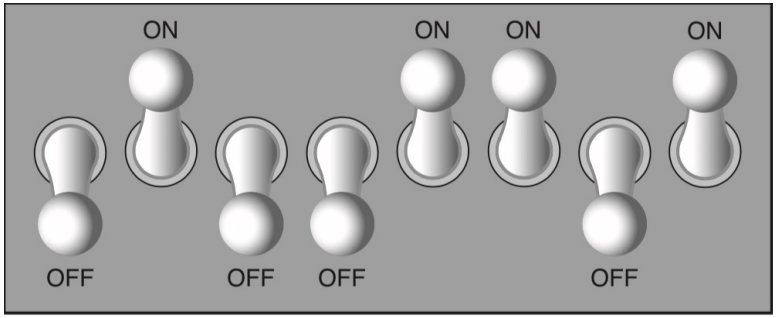
It's all Zeros and Ones …

- Everything that communicates with a computer "speaks" the same language (binary)
- Only two "letters" in this language – "0" and "1" which really correspond to electrical impulses (+5v / -5v)

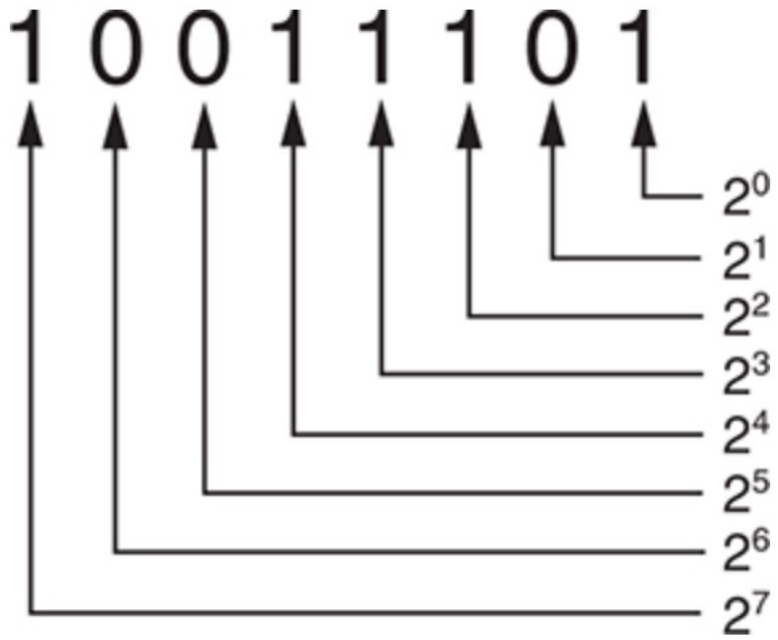


Binary

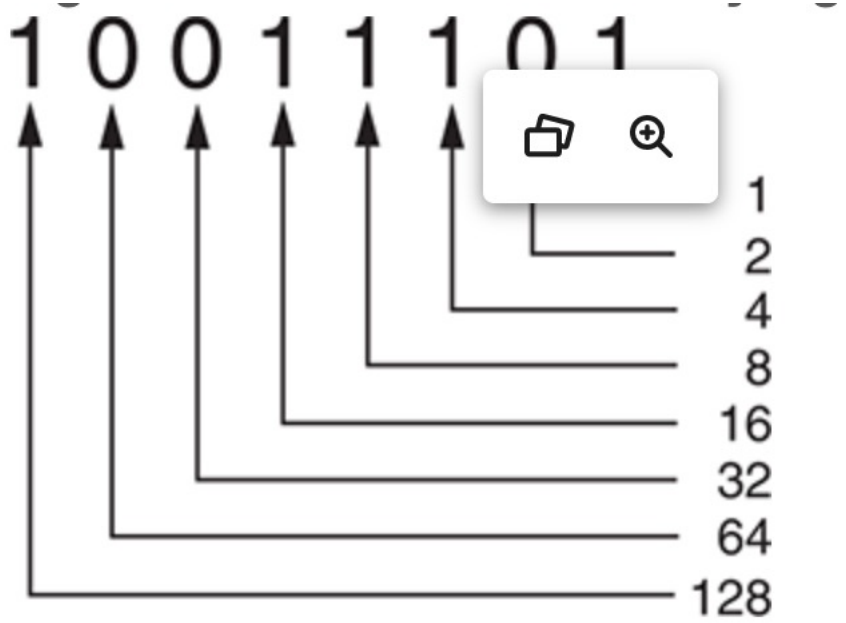
- Only 2 "letters" in the entire language (0 and 1)
- A single 0 or 1 is referred to as a "bit"
  - bit: 1
- Groupings of 8 bits are referred to as a "byte"
  - byte: 01001011
- 1 byte has the possibility of having 256 unique "states"



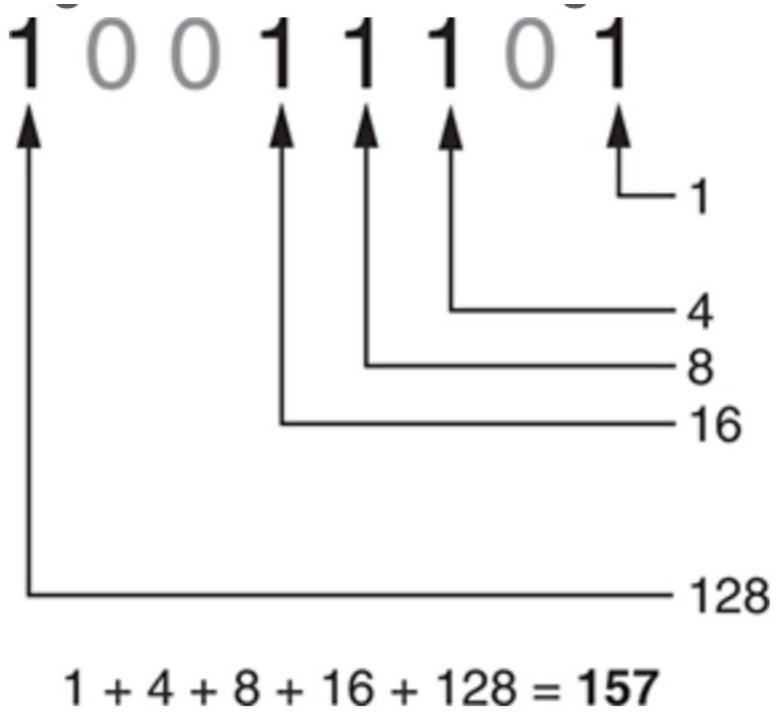
The Value of Binary Digits



The Value of Binary Digits

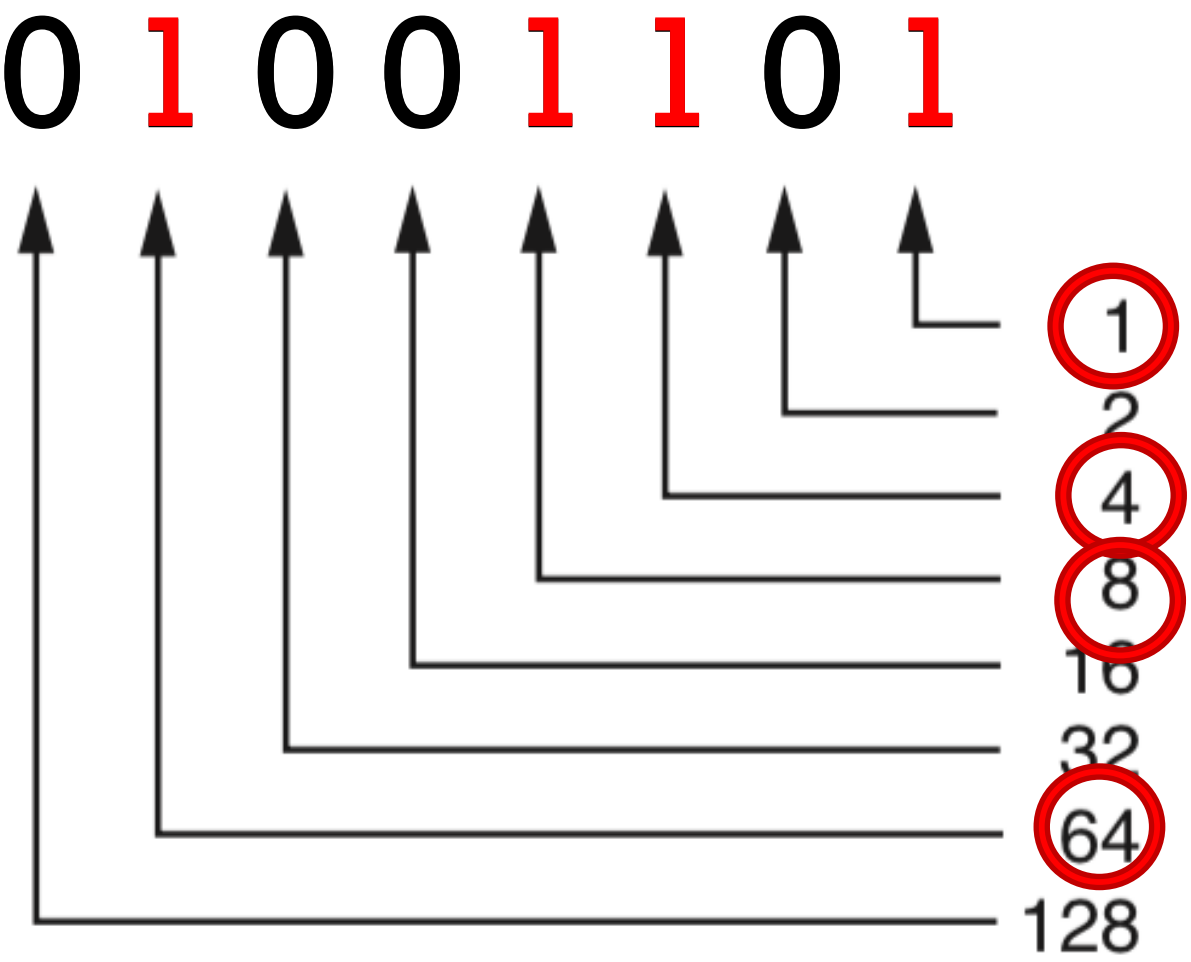


The Value of Binary Digits



1 + 4 + 8 + 16 + 128 = **157**

Try it out!



Counting in Binary (0 to 255)

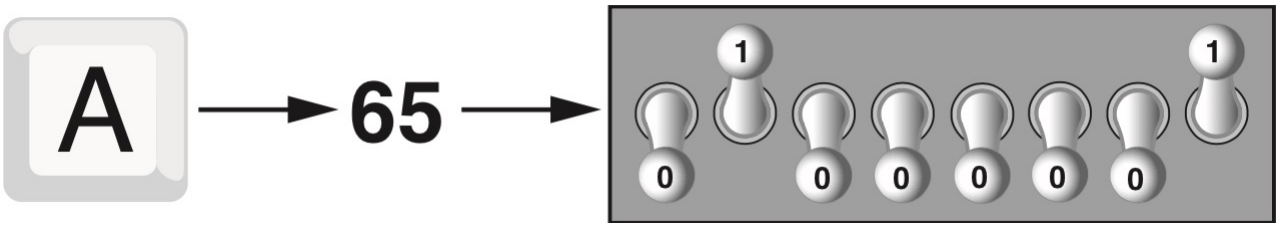
Decimal		128	64	32	16	8	4	2	1
0		0	0	0	0	0	0	0	0
1		0	0	0	0	0	0	0	1
2		0	0	0	0	0	0	1	0
3		0	0	0	0	0	0	1	1
12		0	0	0	0	1	1	0	0
200		1	1	0	0	1	0	0	0
255		1	1	1	1	1	1	1	1

Now you know binary!

+

Relax, you (usually) don't have to do this when programming

Encoding Characters



Let's Practice!

ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

01000111 01100101 01101110 01101001 01110101 01110011
00100000 00111010 01000100

Let's Practice!

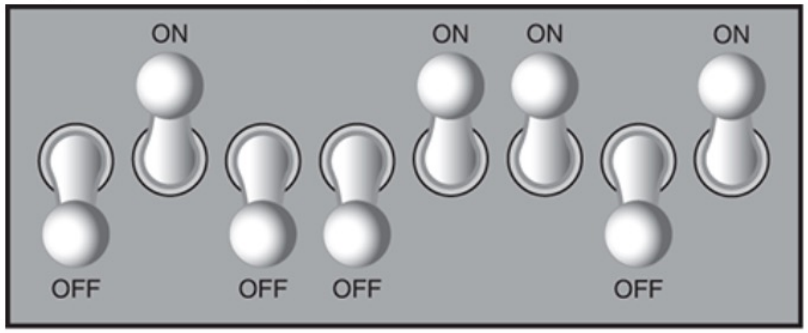
ASCII Code: Character to Binary

0	0011 0000	O	0100 1111	m	0110 1101
1	0011 0001	P	0101 0000	n	0110 1110
2	0011 0010	Q	0101 0001	o	0110 1111
3	0011 0011	R	0101 0010	p	0111 0000
4	0011 0100	S	0101 0011	q	0111 0001
5	0011 0101	T	0101 0100	r	0111 0010
6	0011 0110	U	0101 0101	s	0111 0011
7	0011 0111	V	0101 0110	t	0111 0100
8	0011 1000	W	0101 0111	u	0111 0101
9	0011 1001	X	0101 1000	v	0111 0110
A	0100 0001	Y	0101 1001	w	0111 0111
B	0100 0010	Z	0101 1010	x	0111 1000
C	0100 0011	a	0110 0001	y	0111 1001
D	0100 0100	b	0110 0010	z	0111 1010
E	0100 0101	c	0110 0011	.	0010 1110
F	0100 0110	d	0110 0100	,	0010 0111
G	0100 0111	e	0110 0101	:	0011 1010
H	0100 1000	f	0110 0110	;	0011 1011
I	0100 1001	g	0110 0111	?	0011 1111
J	0100 1010	h	0110 1000	!	0010 0001
K	0100 1011	I	0110 1001	'	0010 1100
L	0100 1100	j	0110 1010	"	0010 0010
M	0100 1101	k	0110 1011	(	0010 1000
N	0100 1110	l	0110 1100	)	0010 1001
				space	0010 0000

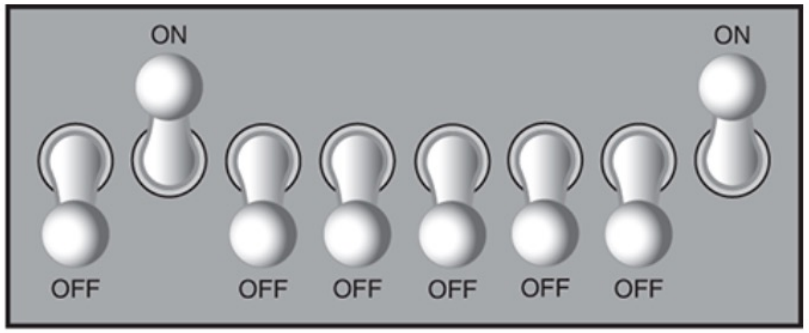
01011001 01101111 01110101 00100000 01100100 01101001
01100100 00100000 01101001 01110100 00100001

Differentiating Bit Patterns

Figure 1-7 Bit patterns for the number 77 and the letter A

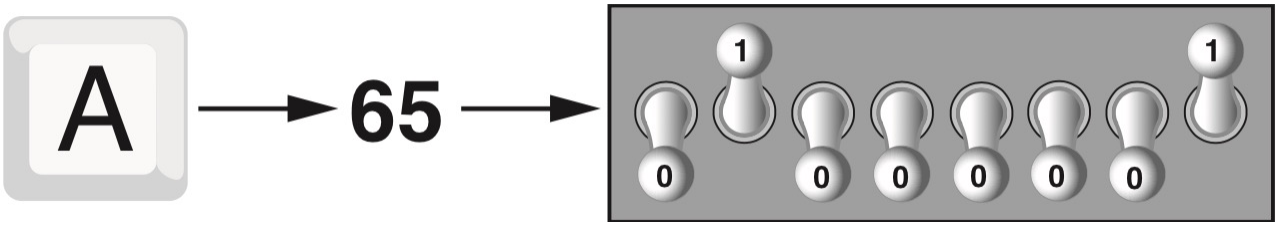


The number 77 stored in a byte.



The letter A stored in a byte.

Encoding Characters

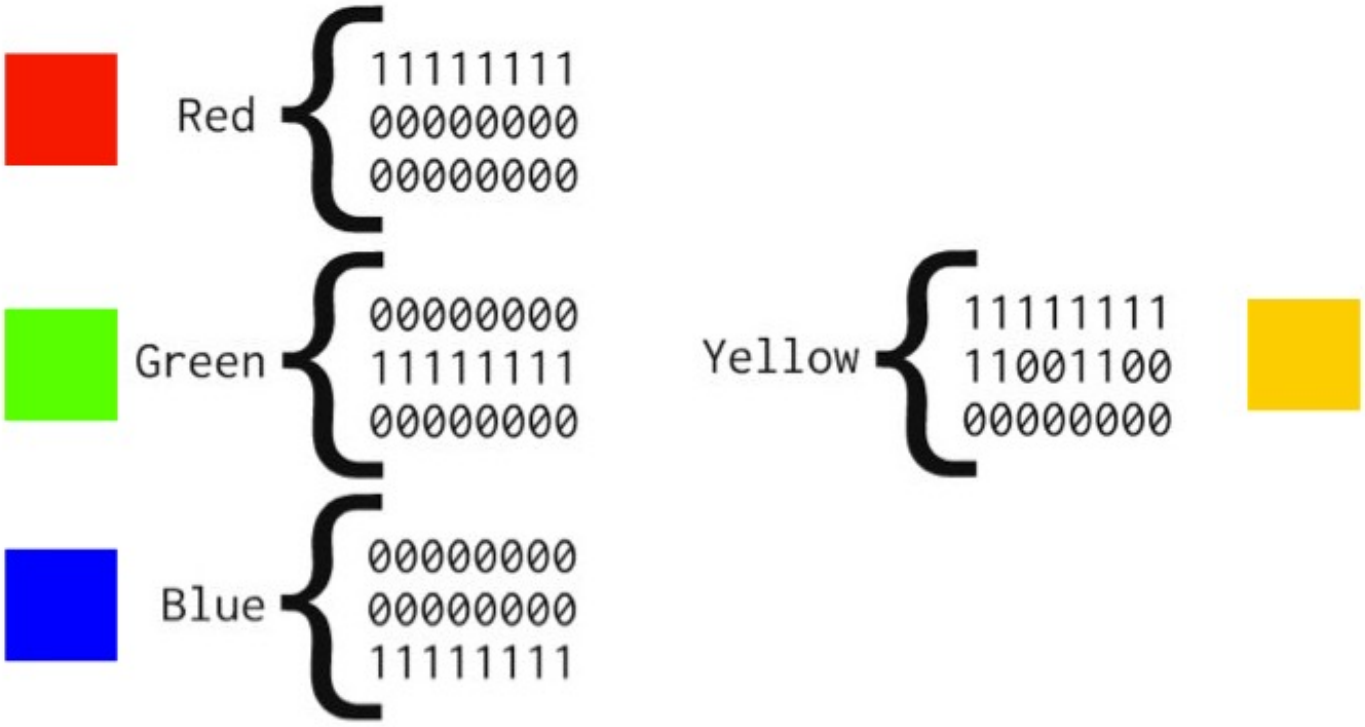


American Standard Code for Information Interchange Table

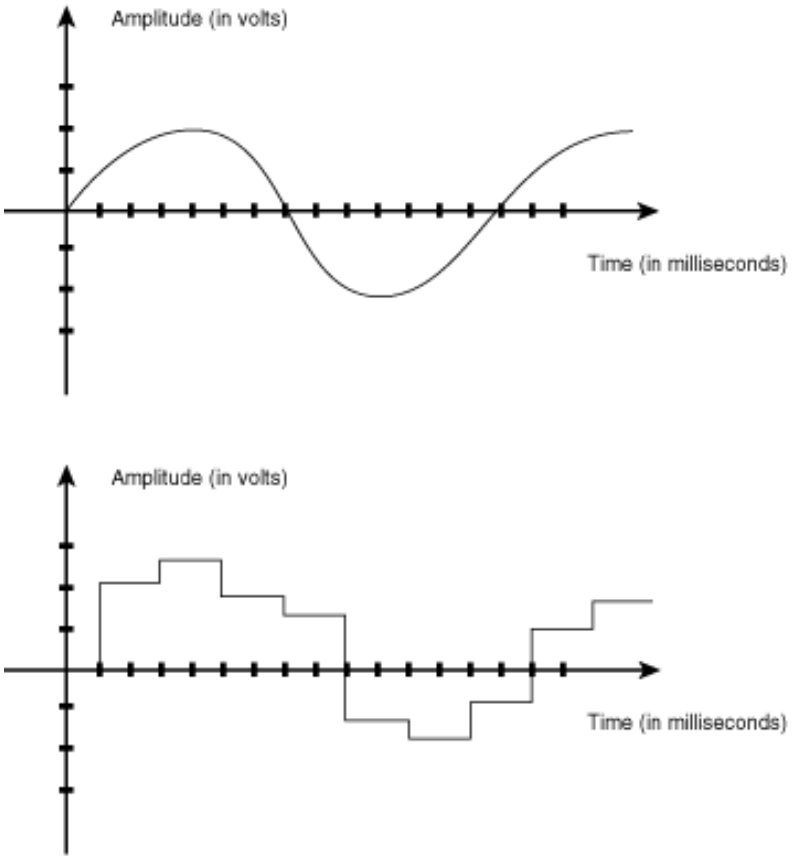
ASCII TABLE

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	*[NULL]*
1	1	1	1	*[START OF HEADING]*
2	2	10	2	*[START OF TEXT]*
3	3	11	3	*[END OF TEXT]*
4	4	100	4	*[END OF TRANSMISSION]*
5	5	101	5	*[ENQUIRY]*
6	6	110	6	*[ACKNOWLEDGE]*
7	7	111	7	*[BELL]*
8	8	1000	10	*[BACKSPACE]*
9	9	1001	11	*[HORIZONTAL TAB]*
10	A	1010	12	*[LINE FEED]*
11	B	1011	13	*[VERTICAL TAB]*
12	C	1100	14	*[FORM FEED]*
13	D	1101	15	*[CARRIAGE RETURN]*
14	E	1110	16	*[SHIFT OUT]*
15	F	1111	17	*[SHIFT IN]*
16	10	10000	20	*[DATA LINK ESCAPE]*
17	11	10001	21	*[DEVICE CONTROL 1]*
18	12	10010	22	*[DEVICE CONTROL 2]*
19	13	10011	23	*[DEVICE CONTROL 3]*
20	14	10100	24	*[DEVICE CONTROL 4]*
21	15	10101	25	*[NEGATIVE ACKNOWLEDGE]*
22	16	10110	26	*[SYNCHRONOUS IDLE]*
23	17	10111	27	*[ENG OF TRANS. BLOCK]*
24	18	11000	30	*[CANCEL]*
25	19	11001	31	*[END OF MEDIUM]*
26	1A	11010	32	*[SUBSTITUTE]*
27	1B	11011	33	*[ESCAPE]*
28	1C	11100	34	*[FILE SEPARATOR]*
29	1D	11101	35	*[GROUP SEPARATOR]*
30	1E	11110	36	*[RECORD SEPARATOR]*
31	1F	11111	37	*[UNIT SEPARATOR]*
32	20	100000	40	*[SPACE]*
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	*[DEL]*

Encoding Images



Encoding Audio



+

How a Program Works

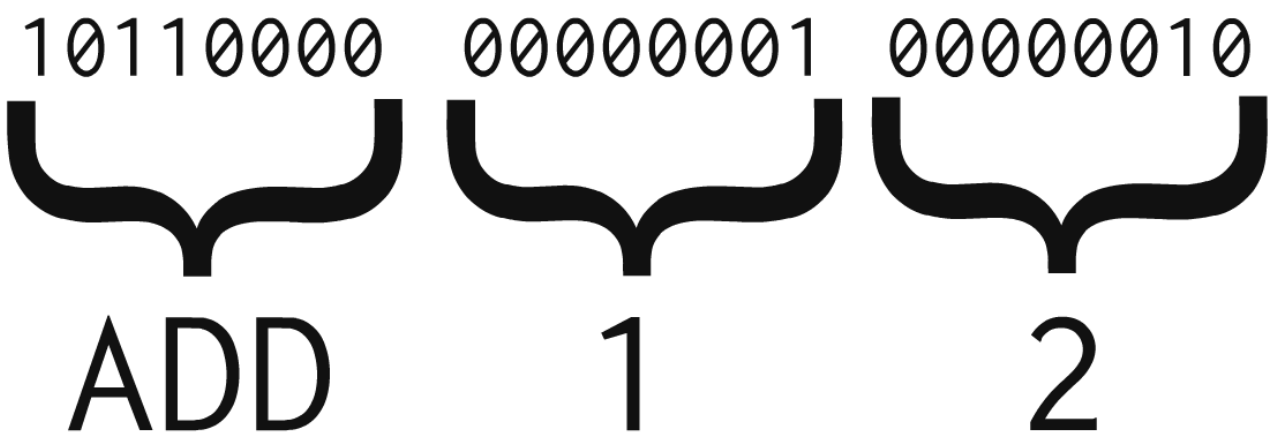
Computers aren't smart!

(they're just really, really, really, really, really, really fast!)

Most computers can only do a handful of things

- Read information from memory
- Add, subtract, multiply, divide numbers
- Move data to memory or to permanent storage
- Compare values

“Machine Language”



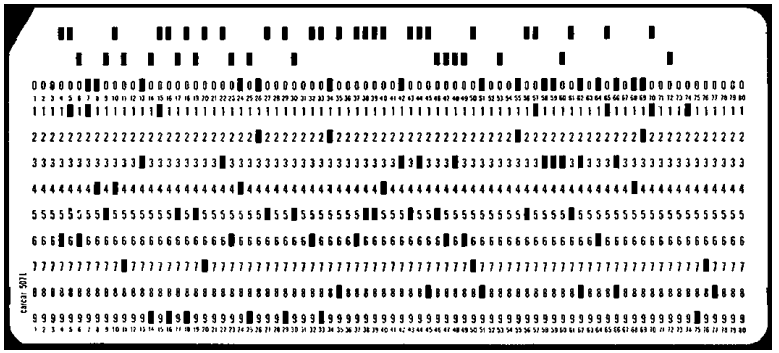
So what can a computer do?

- Processors can only perform a few very simple operations
- Each processor has a fixed number of capabilities, called its "instruction set"
- Each manufacturer maintains its own instruction set

134. SUB - Subtract
135. TEST - Test For Bit Pattern
136. VERR - Verify Read (286+ protected)
137. VERW - Verify Write (286+ protected)
138. WAIT/FWAIT - Event Wait
139. WBINVD - Write-Back and Invalidate Cache (486+)
140. XCHG - Exchange
141. XLAT/XLATB - Translate
142. XOR - Exclusive OR

For a program to be meaningful we need lots of instructions!

- Usually on the order of millions or billions
- Programs are generally stored on external devices, but they must be copied into memory as needed
- Once in memory the CPU can begin to work its magic
  - Fetch
  - Decode
  - Execute
  - Store

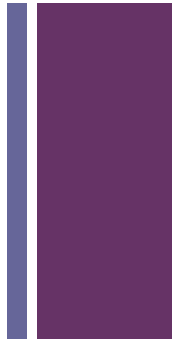


Are we ready to code in machine language?

High Level Languages

- Problem was solved in the 1950's when Grace Hopper, a captain in the US Navy, invented COBOL
- The big idea: Take English words and translate them into machine language in a way that was "device independent"
- Allowed programmers to concentrate on the tasks the needed doing, not on the mechanics of how a machine worked

High Level Languages

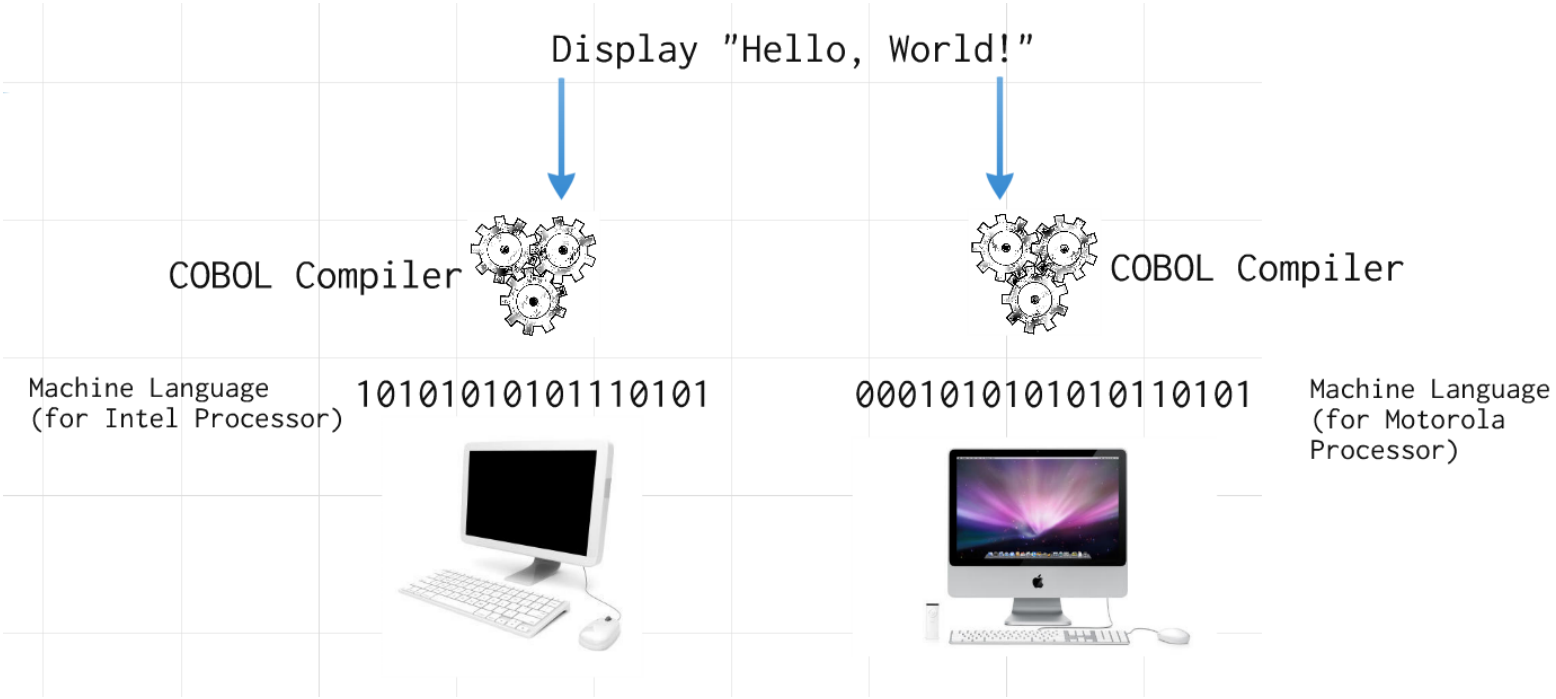


COBOL

```
DISPLAY "Hello, World!"
```

Python

```
print ('Hello, World!')
```


Display "Hello, World!"
COBOL Compiler
COBOL Compiler
Machine Language
(for Intel Processor)
10101010101110101
0001010101010110101
Machine Language
(for Motorola
Processor)

Many, many high level languages

High-Level Programming Language Structure

Programming Language Structure: Key Words

- Defined list of words that make up the language
- Sometimes called "Reserved Words"

```
and       del       from      not       while
as        elif      global    or        with
assert    else      if        pass      yield
break     except    import    print
class     exec      in        raise
continue  finally   is        return
def       for       lambda    try
```

Programming Language Structure: Operators

- Special symbols that perform certain actions on pieces of data

answer = 5 **+** 2

name = ‘Harry’ **+** ‘Potter’

average = 250 **/** 300

Programming Language Structure: Syntax

- Set of rules that must be followed when writing a program

```
if name == 'craig':
    print ('Hi there!')
else:
    print ('Who are you?')
```

Programming Language Structure: Statement

- Instructions that you write, consisting of keywords, operators, punctuation, etc

```
average = average * 2
```

High Level Languages: Code

- All statements you write while programming is referred to as "code" or "source code"

  (we don't say "source codes")

Python

- This semester we will be working with Python
- High level interpreted language
- Used extensively as both a production language as well as a teaching language
- Two modes
  - Interactive
  - Script
- IDLE
  - Integrated Development Environment

+

“Hello, World!”

For next time …

- Read the Syllabus … again
- Begin "Self-Paced Learning Module #1 + 2" on the class website
- Bring a laptop with you to class (if you have one)