

# Intro to Computers

**UT LAW 379M**

**Spring 2021**

Lecture Notes

# Three Major Computer Parts



Central Processing Unit (CPU)



Hard Drive



Random Access Memory (RAM)



[This Photo](#) by Unknown Author is licensed under [CC BY-NC](#)

# First, a Word About Data

- Computers store and process all data as **BINARY NUMBERS**
- Everything is a binary number. Games, programs, music, etc
- The binary numbers are **interpreted** as letters, music, etc

# Quick Binary Lesson

Base 10 Numbers

$10^3 = 1000$	$10^2 = 100$	$10^1 = 10$	$10^0 = 1$
3	0	9	5
(3X1000) +	(0X100) +	(9X10) +	(5X1)
3000 +	0 +	90 +	5
3095			

Base 2 Numbers

$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$
1	0	1	0
(1X8) +	(0X4) +	(1X2) +	(0X1)
8 +	0 +	2 +	0
10			

# What are These Binary Numbers?

- Binary 1000?
- Binary 0100?
- Binary 0010?
- Binary 0001?
- Binary 0101?
- Binary 1010?
- Binary 1111?

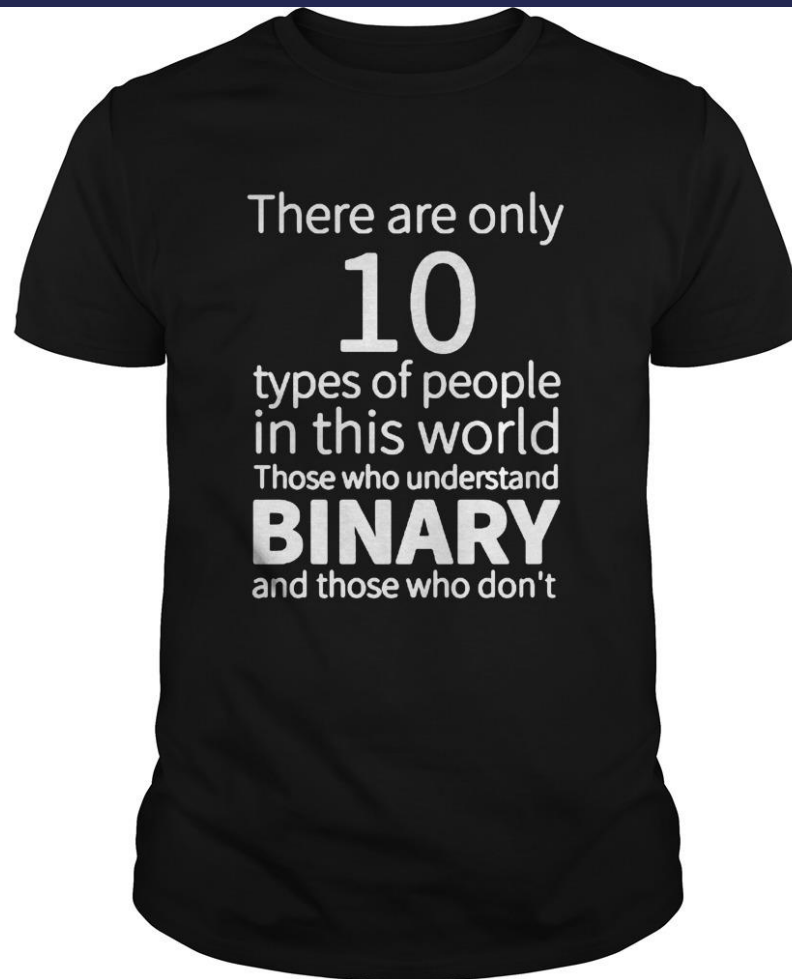
Remember 8, 4, 2, 1 columns.

If there's a one, add the column:

$$0110 = 4 + 2 = 6$$

$$1001 = 8 + 1 = 9$$

# Decode This Shirt



# Binary Sizes

```
09/09/2020 10:50 PM <DIR> Links
09/09/2020 10:50 PM <DIR> Music
10/04/2020 09:41 AM 397,307 npm-debug.log
09/11/2020 11:28 AM <DIR> OneDrive
06/29/2020 01:51 PM 26 osync.log
10/05/2020 11:51 AM <DIR> Pictures
08/26/2019 11:10 PM 13 query
09/09/2020 10:50 PM <DIR> Saved Games
09/09/2020 10:50 PM <DIR> Searches
03/11/2020 08:05 AM 3,298,911 snap_beans1.tgz
08/26/2019 11:10 PM 13 start
05/19/2019 09:57 PM 0 Sti_Trace.log
08/26/2019 11:10 PM 13 stop
10/06/2020 08:53 AM <DIR> Videos
02/10/2020 07:48 PM <DIR> VirtualBox VMs
      8 File(s)      61,789,284 bytes
     22 Dir(s)  12,737,953,792 bytes free

C:\Users\seth_>
```

- A one or a zero is a “bit”
- 8 bits is a byte
- Let's use our command line to see how big files are:
  - Open a terminal on Mac or the cmd shell on Windows
  - Type “ls -l” on Mac or “dir” on Windows

```
sethjn@Paladin:~$ ls -l
total 69160
-rwxrwxrwx 1 sethjn sethjn 737282 Jul 27 13:01 chaum_001.pdf
lrwxrwxrwx 1 sethjn sethjn 30 Aug 26 2019 dev -> /mnt/d/seth_/f
v/
-rw-rw-rw- 1 sethjn sethjn 2488927 Mar 30 2020 files.txt
-rwxrwxrwx 1 sethjn sethjn 32655988 Jul 27 09:58 iso11770_1.pdf
-rwxrwxrwx 1 sethjn sethjn 779655 Jul 27 09:59 iso11770_3.pdf
-rwxrwxrwx 1 sethjn sethjn 186903 Jul 27 22:20 nplaces_caswell.pdf
```

# Binary is Often Represented as Hex

- Binary is Base 2
- For “short hand”, binary is often written in Base 16
- This is because there is a 1:1 mapping between 4 bits and 1 Hex number:

0000 – 0	0100 – 4	1000 – 8	1100 – C (hex 12)
0001 – 1	0101 – 5	1001 – 9	1101 – D (hex 13)
0010 – 2	0110 – 6	1010 – A (hex 10)	1110 – E (hex 14)
0011 – 3	0111 – 7	1011 – B (hex 11)	1111 – F (hex 15)



# Interpretation - Letters

- ASCII – The original English language character mapping
- Each letter, symbol, and control character had a code
- 'a' – 97 (decimal), 0x61 (hex), 0110 0001 (bin)
- 'A' – 65 (decimal), 0x41 (hex), 0100 0001 (bin)
- Newline – 10 (decimal), 0x0a (hex), 0000 1010 (bin)

# Interpretation - Images

- Uncompressed pictures can be 1 number per pixel
- Each number is an index into a table of colors
- 1 bit-per-pixel can do two colors (black and white)
- 8 bpp can do 256 colors
- 32 bpp can do 4,294,967,296 colors

# (Side Note, Compression)

- Images can be compressed with special techniques
- Groups of same-color pixels stored together
- Videos go even further
  - I frame, full picture
  - B frame/p frame, changes from previous picture

# Interpretation - Documents

- Multi-level encoding
- Codes indicate whether text is regular, bold, etc
- All the data is **rendered** by the display program

# HTML Example

- The entire document, except for media, is text
  - So, there is “interpretation” from binary to text
  - And more “interpretation” from text to codes
- Example:
  - `<B>This is bold</B>`
  - The `<B>` says “all text until next `</B>`” is bold

# Processors and Binary



Computer “Instruction” is just a binary number. In older computers, like the 486 shown here, it was 32-bit. In more modern processors it is 64-bit.

1	0	0	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	1	1	1	1	0	1	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

# CPU Instructions



The instruction is broken down into pieces. One part is called the “op code” or operation code, it tells the computer what to do. The other parts are parameters. Each instruction “add”, “subtract,” etc. has its own op code.

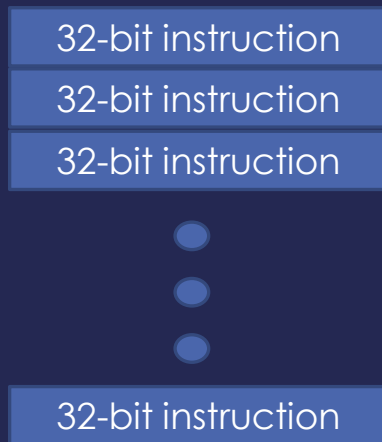
1	0	0	1	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	1	1	1	1	0	1	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

OP CODE

PARAMETER 1

PARAMETER 2

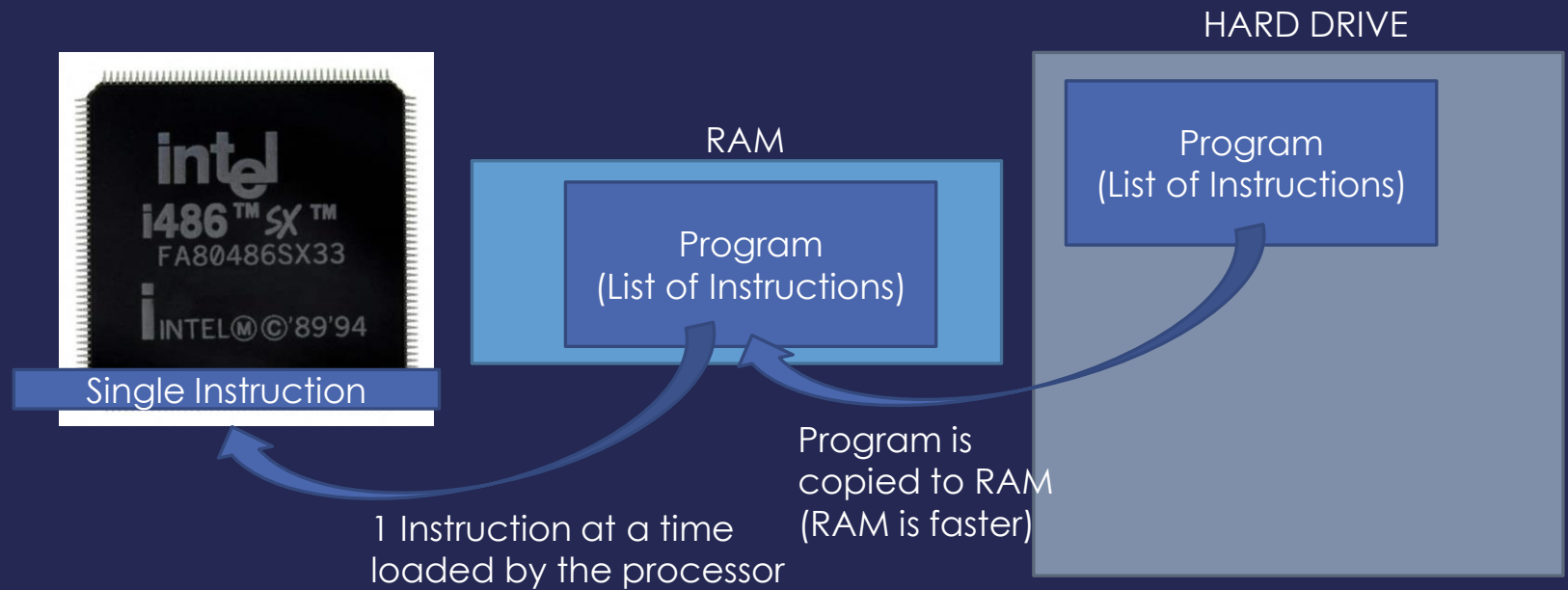
# Program Stored On Disk



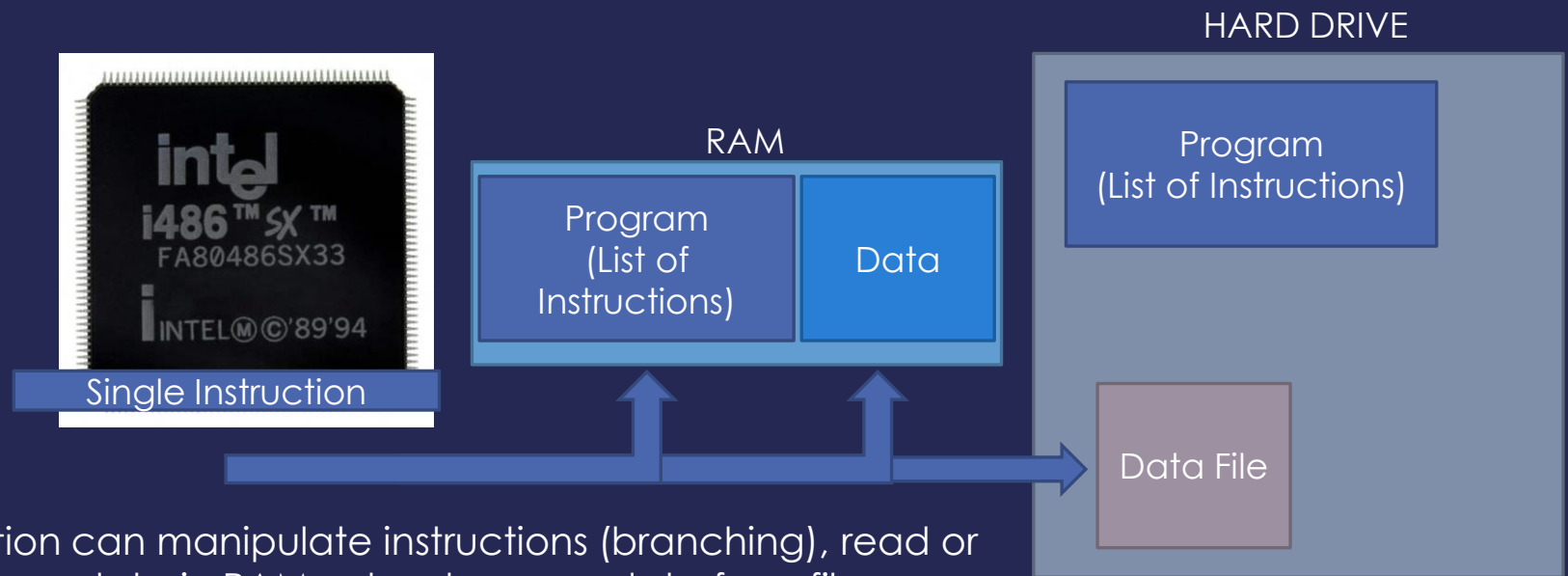
- Program is a list of instructions
- Processor executes one at a time.
- Some instructions are “branches”
- A branch jumps ahead or behind in the list
- Often “conditional” (e.g., if  $x > 0$  then jump)



# Running a Program



# Data Too



Instruction can manipulate instructions (branching), read or write from data in RAM or load or save data from files on disk. (All are just binary numbers)

# Creating a Program

- Programs written in Programming Language
- Programming Language is human readable
- However, programming language **NOT** machine readable
- Remember, processor **ONLY** speaks in basic instructions

# Compiling

- Some languages can be “compiled”
- This means the program is converted to machine readable
- Compiler knows how to translate and organize
- Quite complicated and still subject of research
- Example: C/C++

# C Example

```
#include <stdio>

int main(void) {
    printf("Hello World\n");
    return 0;
}
```



COMPILER



1100101001111010101  
1010010101001110101  
1010101111101110000  
1010101010001111000



# Interpreting

- Some languages can be “interpreted”
- This means the program is “executed” by interpreter
- Usually one instruction at a time
- Interpreter has to be a fully compiled program
- Interpretation is considerably slower
- Example: Python

# Python Example

```
if __name__ == "__main__":  
    print("Hello World")
```



```
print("Hello World")
```

**PYTHON INTERPRETER  
(Already Binary Program)**



1100101001111010101

# Virtual Machine

- An extreme form of “interpreting”
- Virtual processor with virtual instructions
- Programs are compiled to the virtual instructions
- Virtual instructions executed on the virtual processor
- Virtual processor translates instructions to host machine code
- Example: Java



# Java Example

```
import System;
```

```
class HelloWorld {  
    static int main(void) {  
        System.Out.println("Hello World!");  
        return 0;  
    }  
}
```

COMPILER

```
1100101001111010101  
1010010101001110101  
1010101111101110000  
1010101010001111000
```

Java Virtual  
Machine

```
1100101001111010101
```



# Why This is Useful

- Everything in a computer is a number. Programs, Data, etc
- This is great because programs and data can be stored the same way
- Everything can be copied over a network the same way