

Would you rather...

Skip spring?  
Skip fall?

↑  
half goes to  
winter, half to  
summer

# CS 340

---

Building Blocks 0b01  
(Gates and Binary)





[clicker.cs.illinois.edu](https://clicker.cs.illinois.edu)

Q1

~Code~  
340



# Updates

1. MP 0 - Setup due TODAY. 
2. MP 1 - debugger due TODAY. 
3. MP 2 - C from C++ out TODAY
4.  If you added late, let us know ASAP for possible extensions
5. HW 2 Due Thursday 1:59pm
  - a. Overview and C coding 

# Building Blocks 0b01

## Today's LGs:

- Be able to explain how logic, arithmetic calculations, and selection are built up using electrons
  - Knowing the terms, electricity, current, voltage, transistors, gates, binary
  - Be able to read a gate diagram and generate a corresponding truth table
  - Developing your intuition around the binary number system and skills around converting and using bits to do math
  - Be able to recognize how gates are used to do addition and selection

# Summary Slide/Agenda

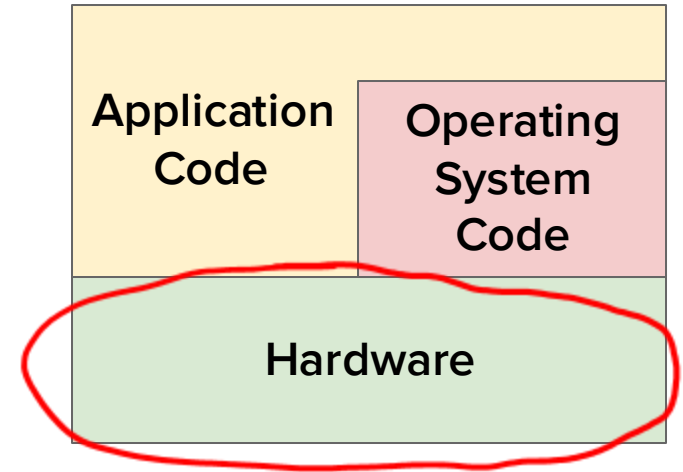
1. Circuit Basics

2. Gates

3. Binary

4. Arithmetic Computations

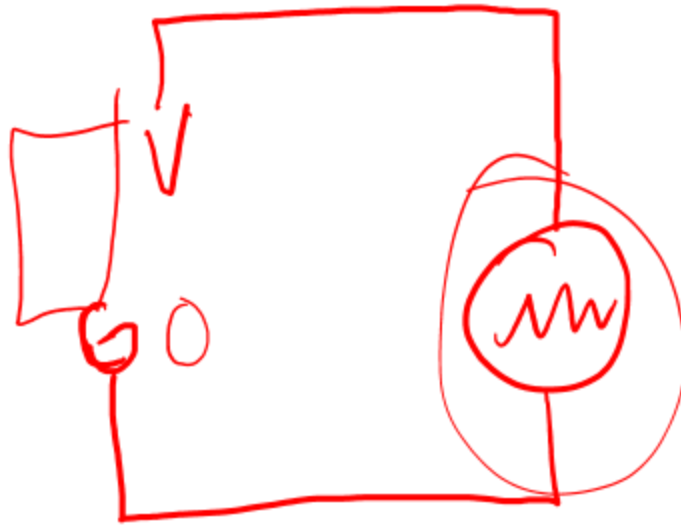
5. Selection



# **Circuit and Gate Basics**

# Circuit Basics

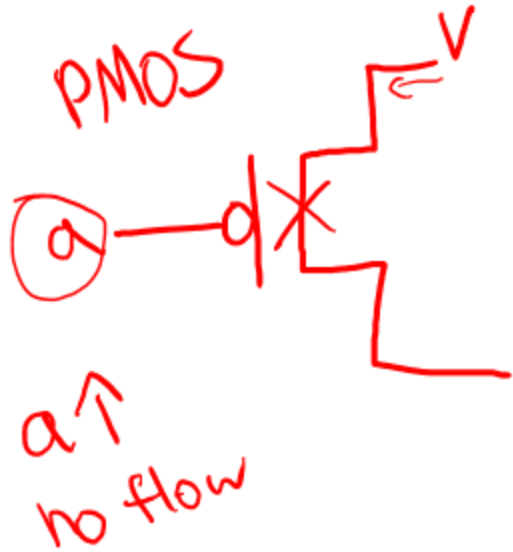
- Electricity
- Current
- Voltage



# Gates



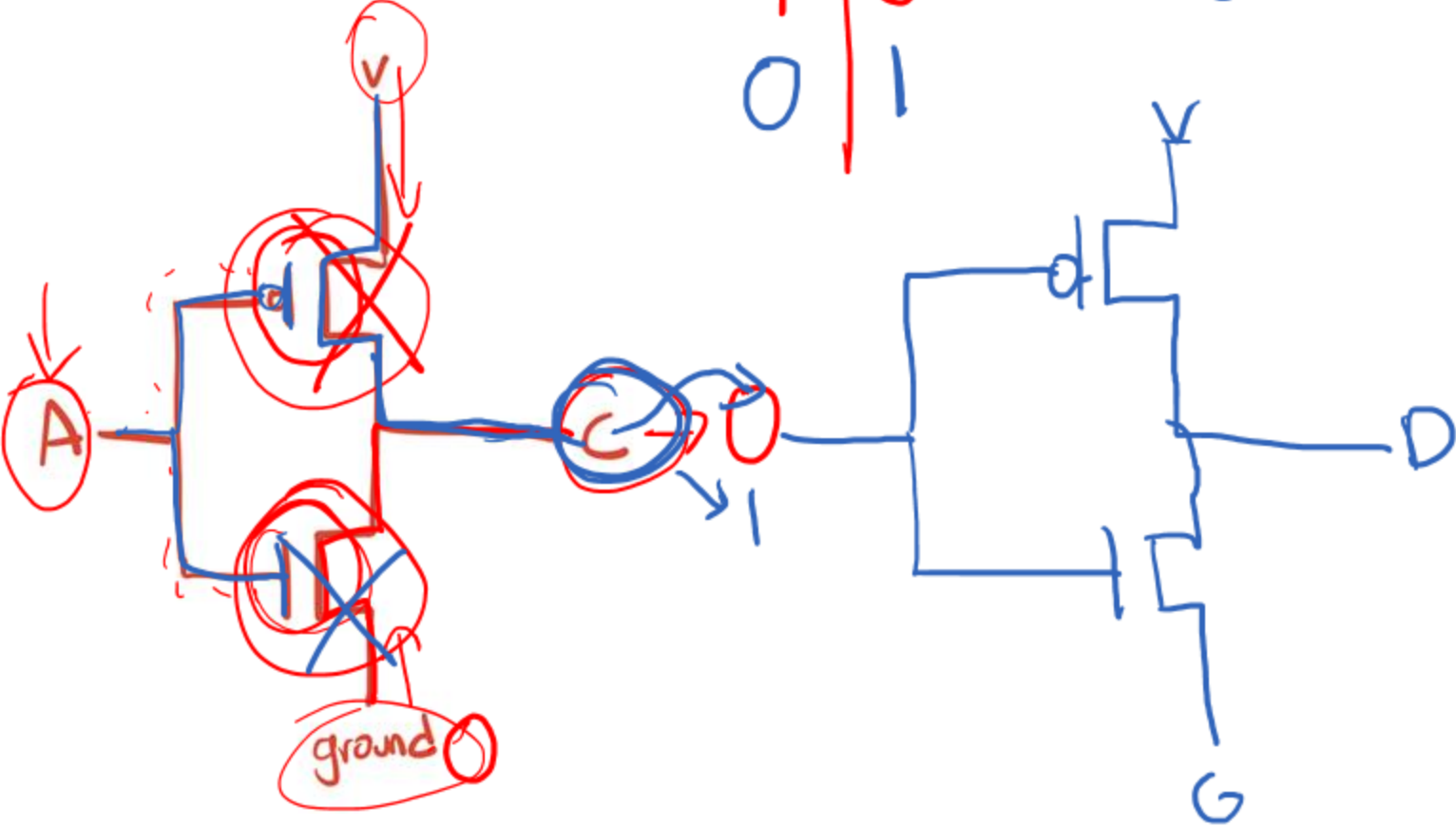
# Transistors



# Example Gate

A	
1	0
0	1

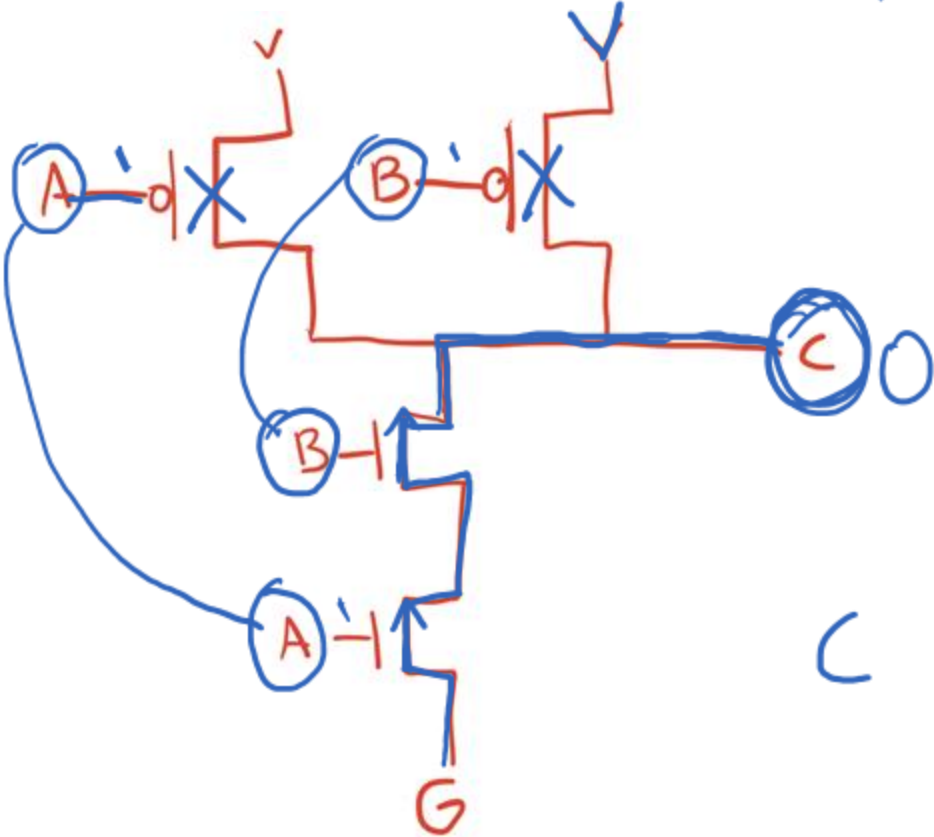
not





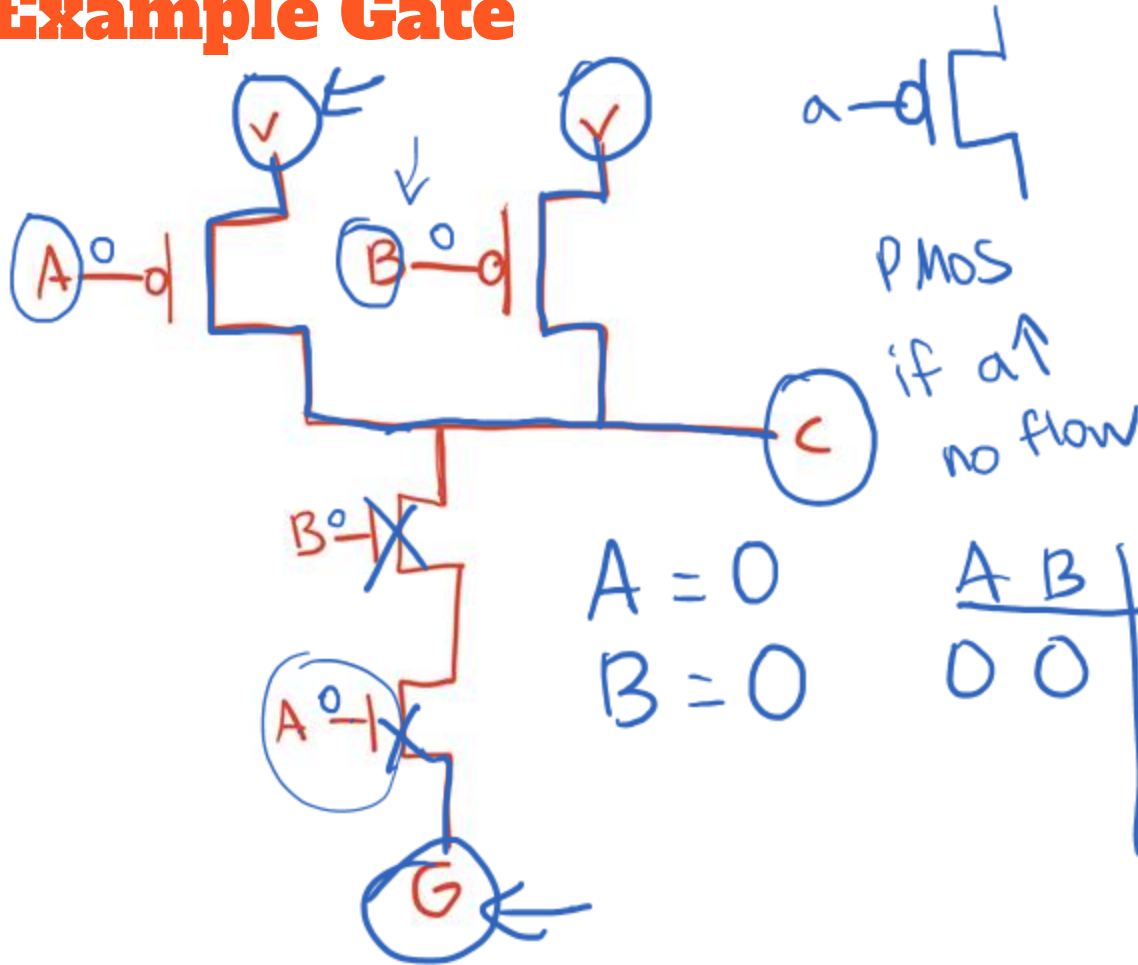
# Example Gate

A	B	C
1	1	0



C

# Example Gate



[clicker.cs.illinois.edu](http://clicker.cs.illinois.edu)

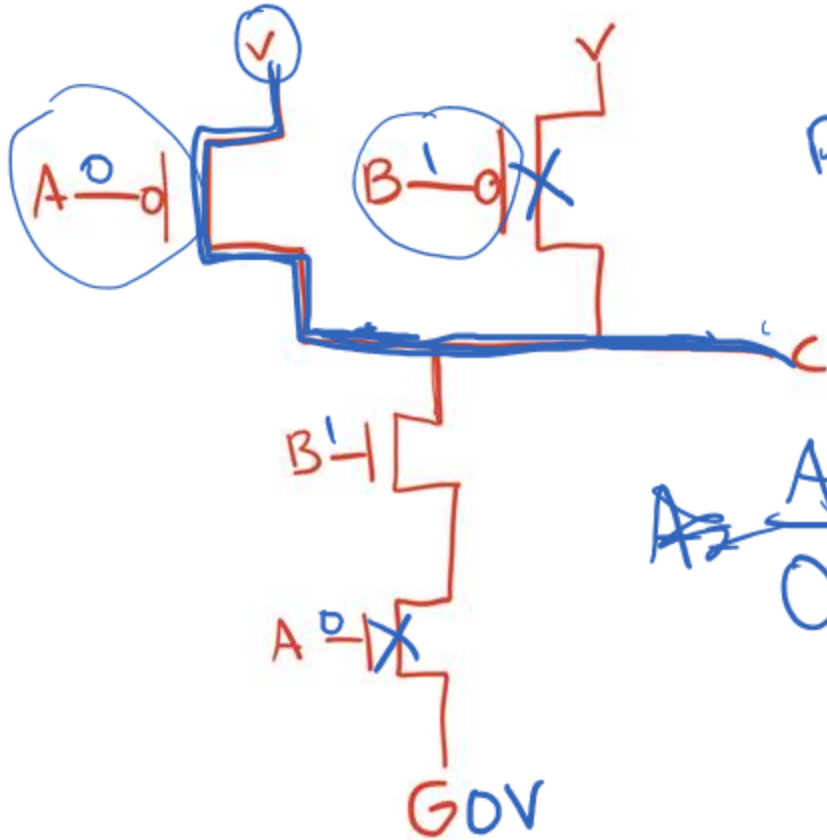
Q2

~Code~  
340



A	B	C
0	0	1

# Example Gate



a-d

PMOS  
if a ↑  
no flow

<del>A</del>	A	B	C
	0	1	1

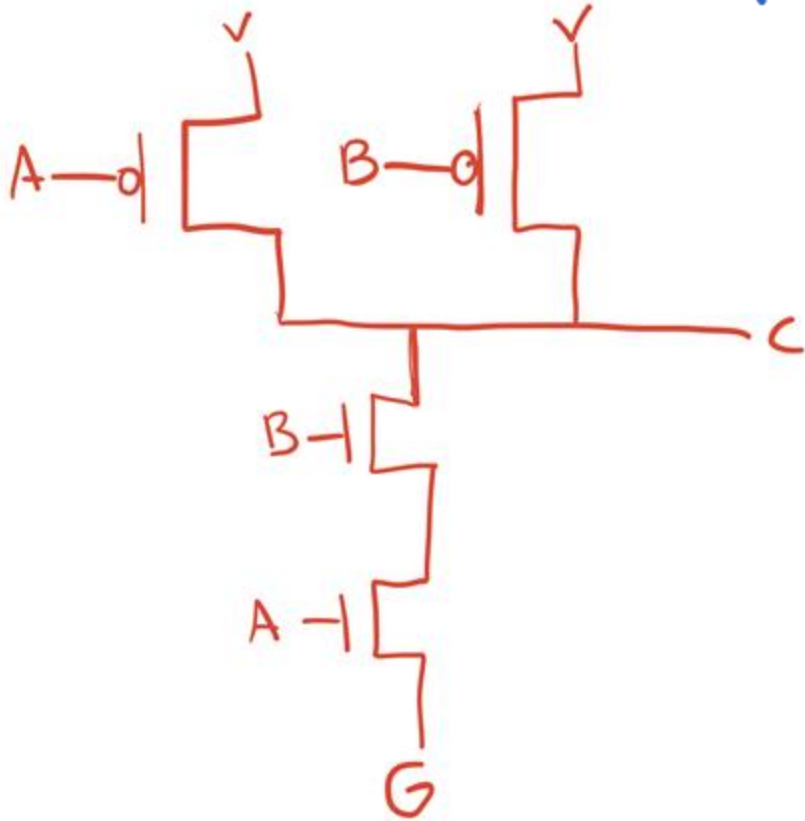
clicker.cs.illinois.edu

Q3

~Code~  
340



# Example Gate



Which  
gate?

clicker.cs.illinois.edu

Q4

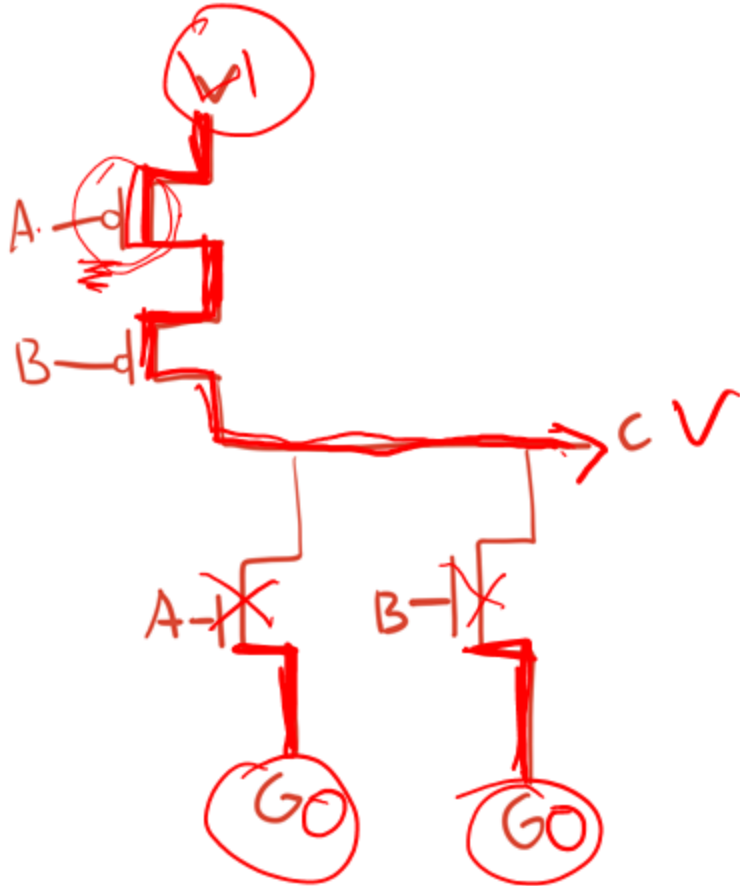
~Code~  
340



A	B	C
1	1	0
1	0	1
0	0	1

NAND

# Example Gate



A	B	C
0	0	1
1	1	
0	1	
1	0	

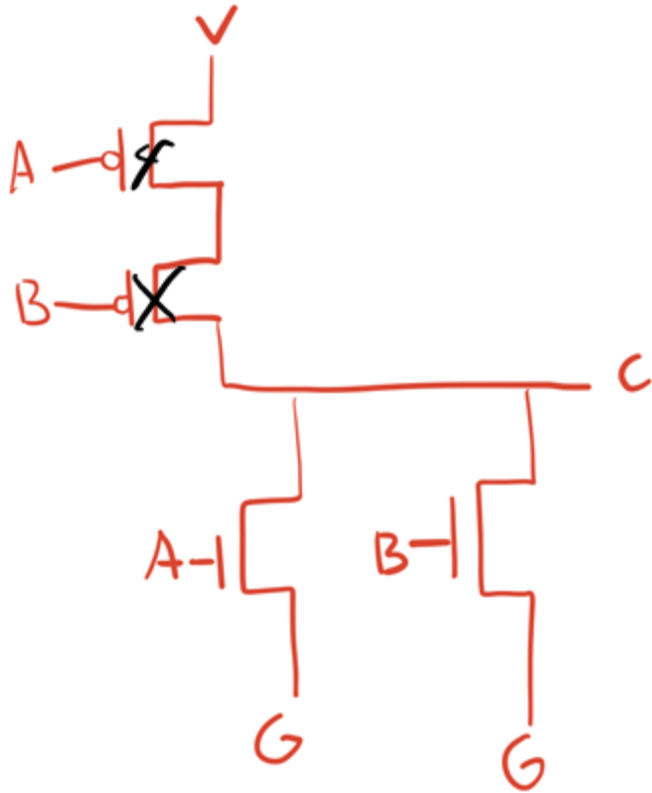
[clicker.cs.illinois.edu](http://clicker.cs.illinois.edu)

Q5

~Code~  
340



# Example Gate



[clicker.cs.illinois.edu](http://clicker.cs.illinois.edu)

Q6

~Code~  
340



# Big Takeaways

AND  
NAND  
OR  
XOR  
NOT



math  
selection

storage ← hardware

busses, clock

# **The Binary Number System**



# Binary Number System

0,1

0,1,2,3,4,5,6,7,8,9

203  
 $10^2$   $10^1$   $10^0$   
 $2 \times 10^2$   $0 \times 10^1$   $3 \times 10^0$

~~101~~ 0101

0110

# Going from Binary to Decimal

0b100101  
32 16 8 4 2 1

$$32 + 4 + 1 = \underline{\underline{37}}$$

x x x x  
x x x  
x x x

# What is 0b10111 in Decimal?

0b10111  
16 8 4 2 1

$$1 + 2 + 4 + 16 = 23$$

clicker.cs.illinois.edu

Q7

~Code~  
340



# Going from Decimal to Binary

$$\begin{array}{cccccccccccc} & & \underline{1056} & & & & & & & & & & \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \uparrow & 1024 & 512 & 256 & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ \hline 2048 \end{array}$$

$$\begin{array}{r} 1056 \\ - 1024 \\ \hline 0032 \end{array}$$

# What is 14 in binary?

0 0 0 0 0 1 1 1 0  
16 8 4 2 1

00032

14  
- 8  
-----

6  
- 2  
-----

[clicker.cs.illinois.edu](https://clicker.cs.illinois.edu)

Q8

~Code~  
340



# Counting in Binary

0000

0001

0010

0011

0100

0101

0110

What is  $0b0001 + 0b1001$  in decimal?

$0b1010$

clicker.cs.illinois.edu

Q9

~Code~  
340



# What is 0b11111 in Decimal?

32 16 8 4 2 1  
+1  
1,00000

31

clicker.cs.illinois.edu

Q10

~Code~  
340





# Addition in Binary

$$\begin{array}{r} \phantom{+} 11011 \\ + 11001 \\ \hline 110100 \end{array}$$

Handwritten binary addition showing the sum of 11011 and 11001, resulting in 110100. The final result 110100 is circled.

**Bits take up space in hardware.**  
**How many bits of space would I**  
**need to fully represent 0b11011**  
**+ 0b11001?**



$$\begin{array}{r} \begin{array}{cccc} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ \hline \end{array} \\ \times 10100 \\ \hline \end{array}$$

6

clicker.cs.illinois.edu

Q11

~Code~  
340

# Subtraction in Binary

# Big Takeaways

current, voltage, transistors, gates, binary



# **An Implication when coding in C**

# Applying to C

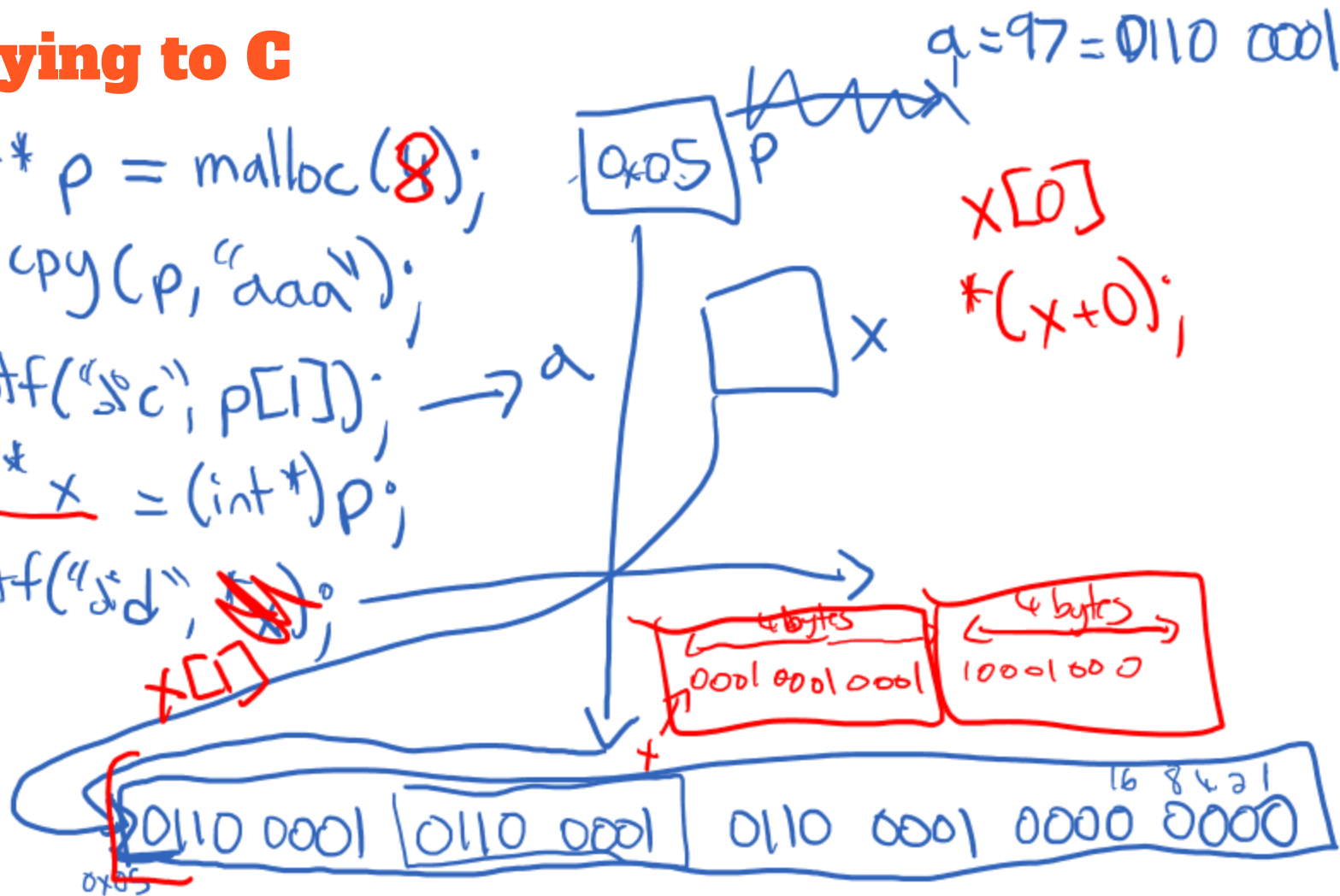
```
char* p = malloc(8);
```

```
mem strcpy(p, "aaa");
```

```
printf("%c", p[1]);
```

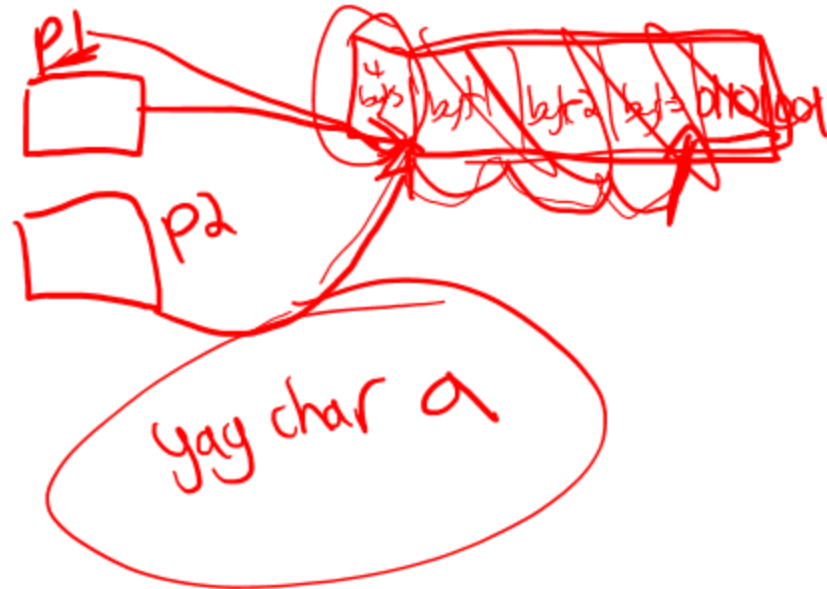
```
int* x = (int*)p;
```

```
printf("%d", x);
```



# What happens?

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() { malloc 4 bytes
5     int* p1 = malloc(sizeof(int));
6     → char* p2 = (char*)p1;
7     → p2[3] = 'a';
8     printf printf("yay char: %c", p2[3]);
9     free free(p1);
10 }
```



clicker.cs.illinois.edu

Q12

~Code~  
340



```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int* p1 = malloc(sizeof(int));
6     char* p2 = (char*)p1;
7     p2[3] = 'a';
8     printf("yay char: %c", p2[3]);
9     free(p1);
10 }
```



