

Logical and Bitwise Expressions

The truth value will set you free.

**Can You
Handle
the Truth?**

iClicker Attendance

Please click on A if you are here:

A. I am here today.

Exercise Review: Ball Motion

Assume you throw a (frictionless) ball straight up in the air at some initial velocity. With gravity acting at 9.81 m/s^2 , after time t , what is the height and velocity of the ball?

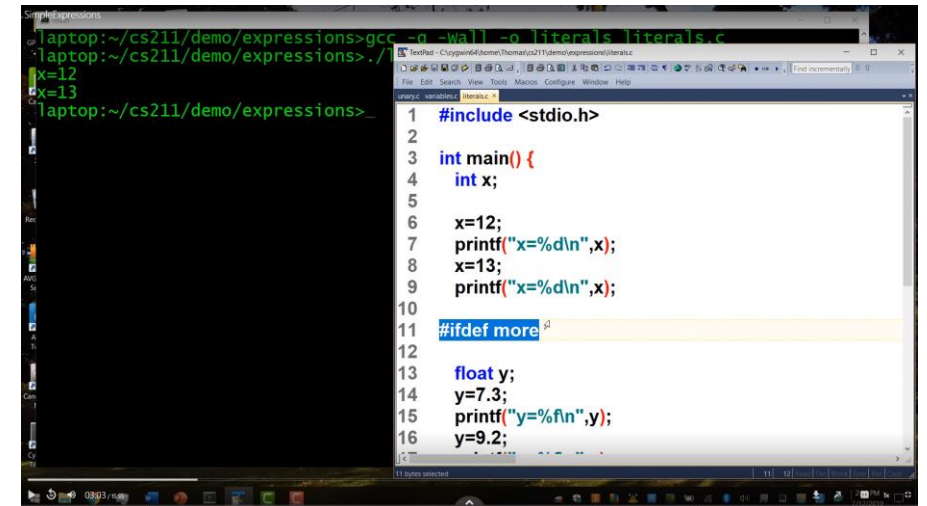
Physics: $v(t) = v_0 - gt$ and $h(t) = v_0 t - \frac{gt^2}{2}$

Q: What do you need to know? v_0 and t



Video Review: Simple Expressions

- Assignment Statements (lhs=rhs)
- Expressions:
 - Literal Expressions
 - Variable Expressions
 - Unary Operator Expressions
 - Unary Operators with side effects
 - Pitfall: Mixing assignment and increment/decrement



The image shows a terminal window on the left and a code editor on the right. The terminal window displays the following commands and output:

```
laptop:~/cs211/demo/expressions>gcc -g -Wall -o literals literals.c
laptop:~/cs211/demo/expressions>./literals
x=12
x=13
laptop:~/cs211/demo/expressions>
```

The code editor shows the source code for `literals.c`:

```
1 #include <stdio.h>
2
3 int main() {
4     int x;
5
6     x=12;
7     printf("x=%d\n",x);
8     x=13;
9     printf("x=%d\n",x);
10
11 #ifdef more
12
13     float y;
14     y=7.3;
15     printf("y=%f\n",y);
16     y=9.2;
```

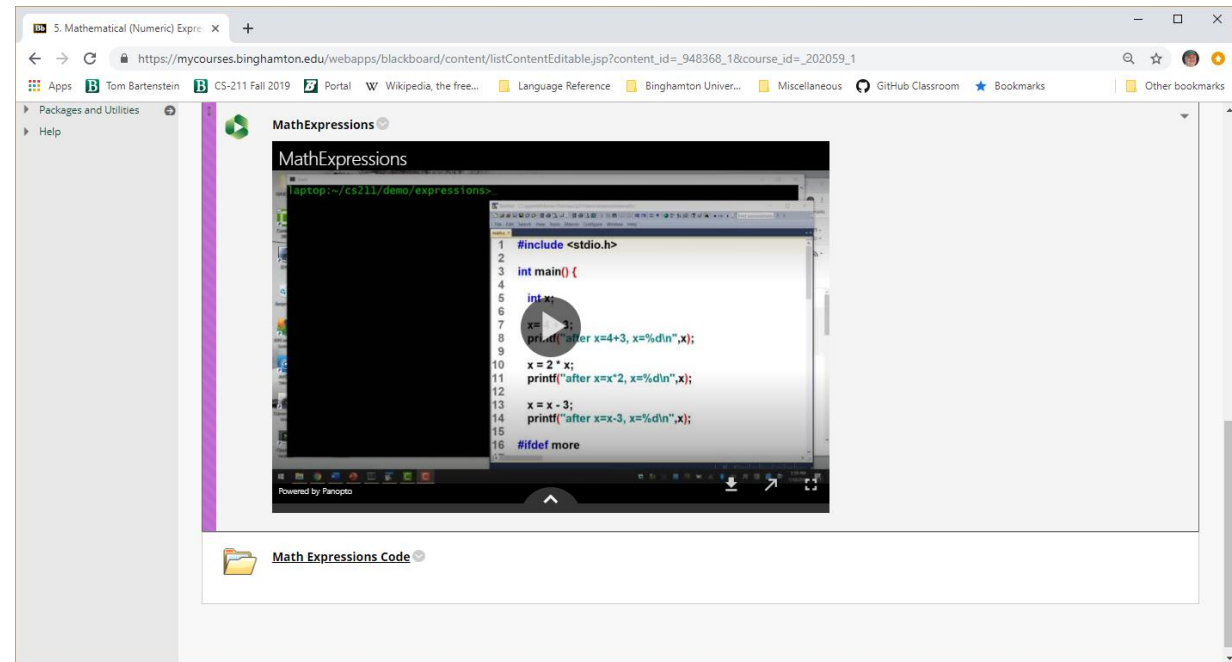
Class Exercise: Assignments

- What does the following C code print?

```
int x=83; int y=x+1;  
printf("First y is %c\n",y);  
y = x++;  
printf("Second x=%d y=%d\n",x,y);  
x = -++y;  
printf("Third x=%d y=%d\n",x,y);
```

Video Review: Math Expressions

- Calculator functions: +, -, *, and / (integer and float)
- Remainder (%)
- Operator Precedence
- Assignment
- Op/Assignment



Class Exercise: Ball Motion / Height

Given a height, “h”, what initial velocity is required when you throw a ball in the air to reach that height, but no higher?

Physics: $v(t) = v_0 - gt$ and $h(t) = v_0t - \frac{gt^2}{2}$

$g=9.81 \text{ m/s}^2$

Note: There is a “sqrt()” built-in function in library “math.h”

Class Exercise: Batting Average

- Prompt the user for the number of at bats
- Prompt the user for the number of times on base
- Compute the batting average using the formula:

$$avg = \frac{onBase}{atBats} \times 1000$$

- Print the result, rounded to the nearest integer.

Condition

- Expression interpreted as a logical value (true or false)

```
int x=9;
```

```
x > 10
```

```
x == 6
```

```
(x - 9)
```

```
x && (113 / x < 12)
```



- Watch the video, available on myCourses
 - Content
 - Videos
 - 6. Logic and Bit Oriented Expressions

The screenshot shows a web browser window with the URL https://mycourses.binghamton.edu/webapps/blackboard/content/listContentEditable.jsp?content_id=_951757_1&course_id=_202059_1. The page is titled "Logic and Bit Oriented Expressions". The left sidebar contains a navigation menu for the course "Fall 2019 - Programming I Engineers (CS-211-A 0)". The main content area features a video player titled "LogicalExpressions" which displays a terminal window with the following C code:

```

1 #include <stdio.h>
2
3
4 int main() {
5
6     if (0) printf("Zero is true\n");
7     else printf("Zero is false\n");
8
9
10
11
12
13
14
15
16

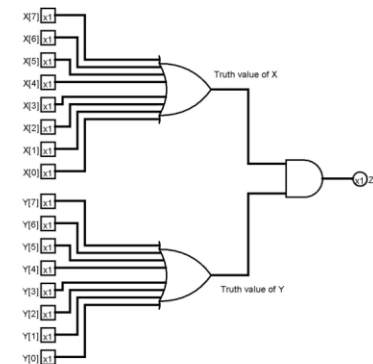
```

Bit Oriented Expressions

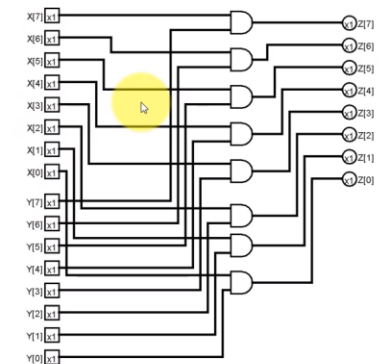
- Watch the video, available on myCourses
 - Content
 - Videos
 - 6. Logic and Bit Oriented Expressions

Logical vs. Bitwise Hardware (8 bit)

Logical $Z = X \& Y$

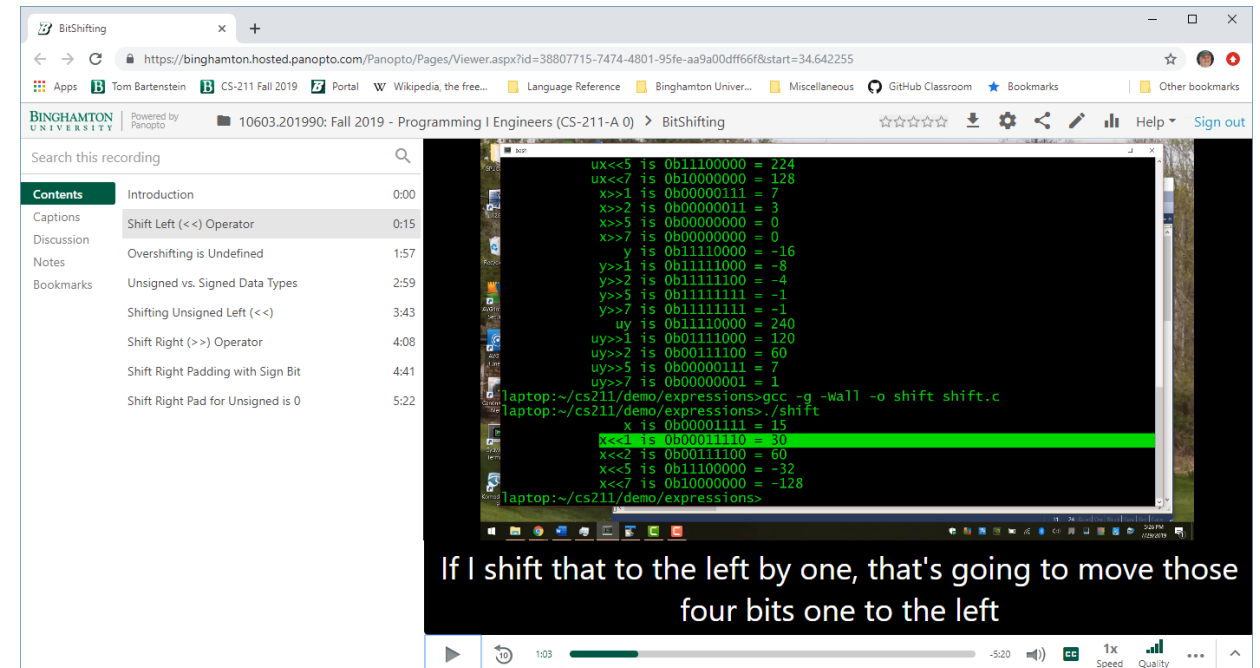


Bitwise $Z = X \& Z$



Bit Shifting

- Watch the video, available on myCourses
 - Content
 - Videos
 - 6. Logic and Bit Oriented Expressions



The screenshot shows a video player interface for a recording titled "BitShifting". The browser address bar shows the URL: <https://binghamton.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=38807715-7474-4801-95fe-aa9a00dff6f8&start=34.642255>. The video player includes a search bar and a table of contents on the left side.

Contents	Introduction	0:00
Captions	Shift Left (<<) Operator	0:15
Discussion	Overshifting is Undefined	1:57
Notes	Unsigned vs. Signed Data Types	2:59
Bookmarks	Shifting Unsigned Left (<<)	3:43
	Shift Right (>>) Operator	4:08
	Shift Right Padding with Sign Bit	4:41
	Shift Right Pad for Unsigned is 0	5:22

The video content shows a terminal window with the following output:

```

ux<<5 1s 0b11100000 = 224
ux<<7 1s 0b10000000 = 128
x>>1 1s 0b00000111 = 7
x>>2 1s 0b00000011 = 3
x>>5 1s 0b00000000 = 0
x>>7 1s 0b00000000 = 0
y 1s 0b11110000 = -16
y>>1 1s 0b11111000 = -8
y>>2 1s 0b11111100 = -4
y>>5 1s 0b11111111 = -1
y>>7 1s 0b11111111 = -1
uy 1s 0b11110000 = 240
uy>>1 1s 0b01111000 = 120
uy>>2 1s 0b00111100 = 60
uy>>5 1s 0b00000111 = 7
uy>>7 1s 0b00000001 = 1
x 1s 0b00001111 = 15
x<<1 1s 0b00011110 = 30
x<<2 1s 0b00111100 = 60
x<<5 1s 0b11100000 = -32
x<<7 1s 0b10000000 = -128
laptop:~/cs211/demo/expressions>gcc -g -Wall -o shift shift.c
laptop:~/cs211/demo/expressions>./shift

```

Below the terminal window, a subtitle reads: "If I shift that to the left by one, that's going to move those four bits one to the left".

Exercise: Convert to Binary & Hex

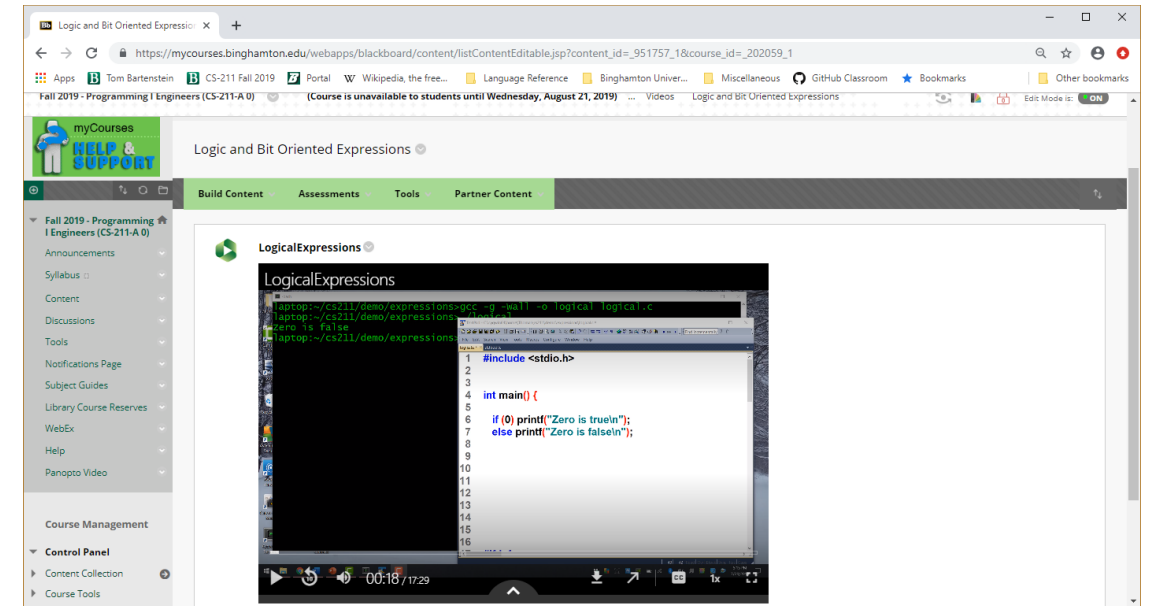
Get an integer number from the user.

Write out the binary representation of that number.

Write out the hexadecimal representation of that number (without using printf %x format).

Resources

- Programming in C, Chapter 3
- Wikipedia: Operators in C and C++
(https://en.wikipedia.org/wiki/Operators_in_C_and_C%2B%2B)
- Wikipedia: Short Circuit Evaluation
(https://en.wikipedia.org/wiki/Short-circuit_evaluation)
- Wikipedia: Bit manipulation
(https://en.wikipedia.org/wiki/Bit_manipulation)

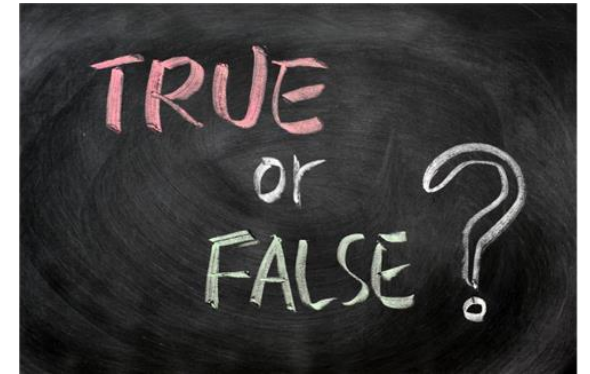


Logical Expressions

Summary Notes

C Numbers and Logic

- Logic deals with two values: True and False
- Numbers have many values
- In C, zero is interpreted as logically “False”
 - Expressions which evaluate to “False” have a value of 0
- In C, every number OTHER than zero is interpreted as logically “True”
 - Both positive and negative numbers are “True”
 - Logically, -32,451 and 1 and 345 all are “True”
 - Expressions which evaluate to “True” have a value of 1



Logical (Boolean) Variables

Option 1

- Use an integer variable
- Use '1' for true and '0' for false

```
int firstTime=1;  
myFunction(firstTime);  
firstTime=0;  
myFunction(firstTime);
```

Option 2

- Use library: <stdbool.h>
 - Includes data type "bool"
 - Includes values true/false

```
#include <stdbool.h>  
bool firstTime=true;  
myfucntion(firstTime);  
firstTime=false;  
myfunction(firstTime);
```

Unary Operator Logical Expression

- Only one unary logic operator... ! (not)
- !true => false !false=>true

```
int x=271;
```

```
x = !x;
```

```
int y = !x;
```

What is x? y?

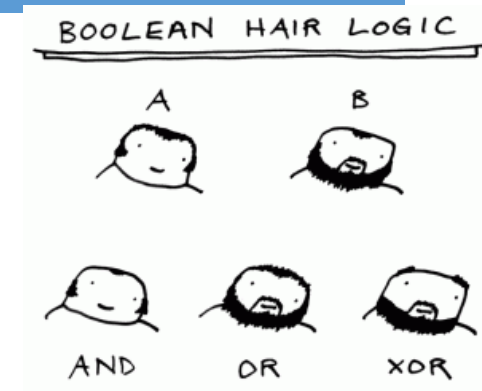
Binary Logic Operators

- Comparison: <, <=, ==, >, >=, !=
- Pitfall: Comparison is different than assignment!

```
int a=16;  
if (a=3) printf("The value of a is %d\n",a);  
  
if (3==a) printf("The value of a is 3\n");
```
- Also && and ||

Logical Truth Tables

A	B	A B	A&&B
F	F	F	F
F	T	T	F
T	F	T	F
T	T	T	T



Short Circuit Evaluation

- Given: *cond1* && *cond2*
 - If *cond1* evaluates to False, *cond2* is not evaluated!
- Given *cond1* || *cond2*
 - If *cond1* evaluates to True, *cond2* is not evaluated!



Example short circuit...

```
if ((x != 0) && (37/x > 3)) printf("x works\n");  
else {  
    printf("x is <=0 or too big to satisfy 37/x>3\n");  
    printf("Aborting");  
    exit(1);  
}
```

Ternary operator expression

condition ? true_expr : false_expr

Evaluate *condition*...

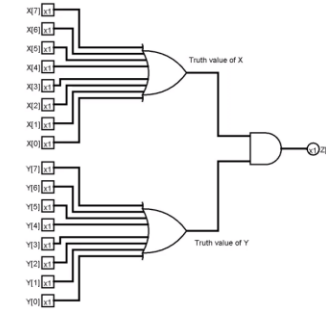
if true, evaluate *true_expr*

if false, evaluate *false_expr*

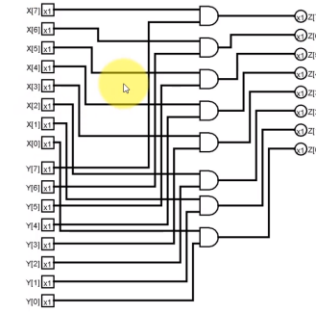
```
y=x ? 3/x :0; /* If x=7, y=2... if x=0, y=0 */  
printf("B variable is %s\n",B?"true":"false");  
hamlet=question?bb:!bb;
```

Logical vs. Bitwise Hardware (8 bit)

Logical Z = X && Y



Bitwise Z = X & Z

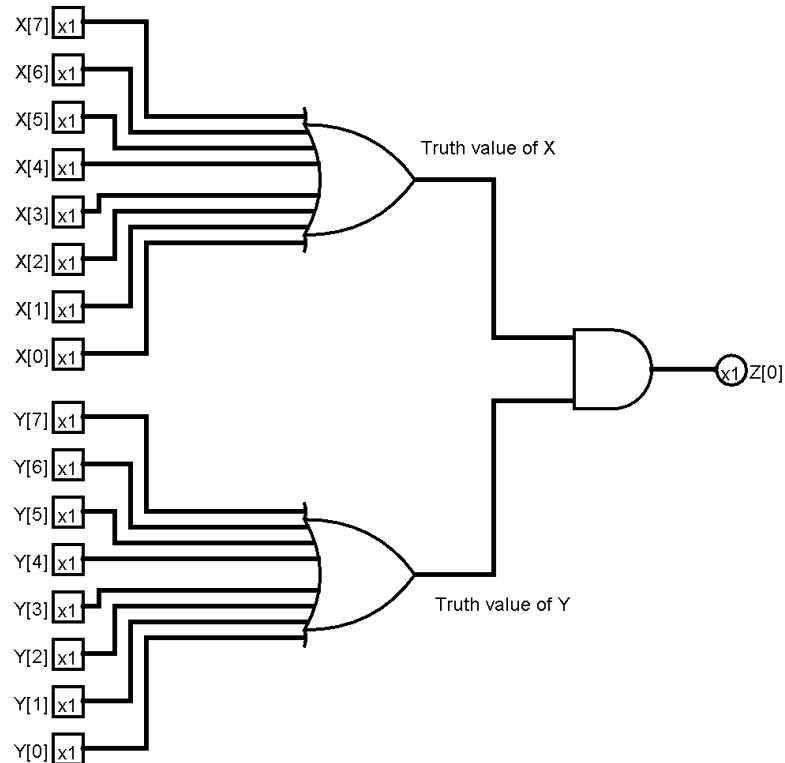


Bit Oriented Expressions

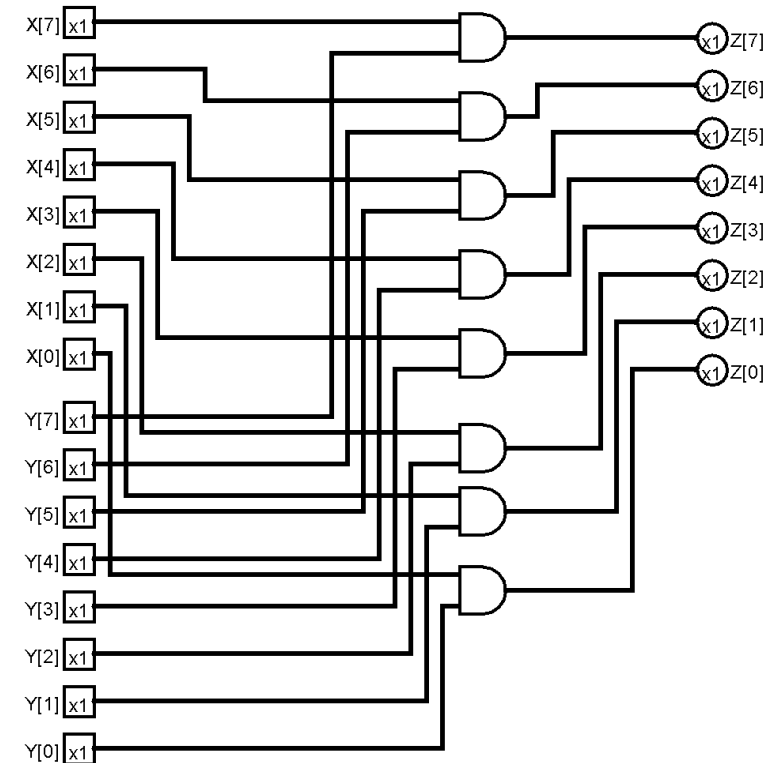
Summary Notes

Logical vs. Bitwise Hardware (8 bit)

Logical Z = X && Y



Bitwise Z = X & Y



Logical vs. Bitwise in Software

Logical

```
char x=0b00011111;
char y=0b11110011;
char z = x && y;
```

```
x!=0, so x is true
y!=0, so y is true
z = true && true
z = true = 0b0000 00001
```

Bitwise

```
char x=0b00011111;
char y=0b11110011;
char w = x & y;
```

```
x= 0b0001 1111 = 31
y= 0b1111 0011 = -13
&-----
w= 0b0001 0011 = 19
```

Logical vs. Bitwise Comparison

Logical

- The entire variable has a single truth value
- Logical operators work on truth values (each operand is T or F)
- A logical operator performs one logical operation
- Use double operators... `&&` `||`

BitWise

- Each bit in the variable has a single truth value
- BitWise operators work on columns of bits
- A bitwise operators performs multiple logical operations ... one for each bit position in the arguments
- Use single operators `&` `|` `^`

Bitwise Operators

- $\sim$ - “invert” operator... flips each bit in the argument
 - (different from !... logical “not”... inverts the logical value of a variable)
 - Unary... all the rest are binary
- $\&$ - and operator... 1 if both bits are 1. (used to turn OFF bits)
 - 0 bit forces 0 result
 - 1 bit – use other operand’s value
- $|$ - or operator... 1 if either bit is 1. (used to turn ON bits)
 - 0 bit – use other operand’s value
 - 1 bit – forces 1 result
- $\wedge$ - exclusive or operator... 1 if bits are different (selective invert)
 - 0 bit – use other operand’s value
 - 1 bit – use inverse of other operand’s value

Bit Shifting

Summary Notes

The video player shows a lecture titled "Bit Shifting". The main content is a terminal window displaying the following C code and its output:

```

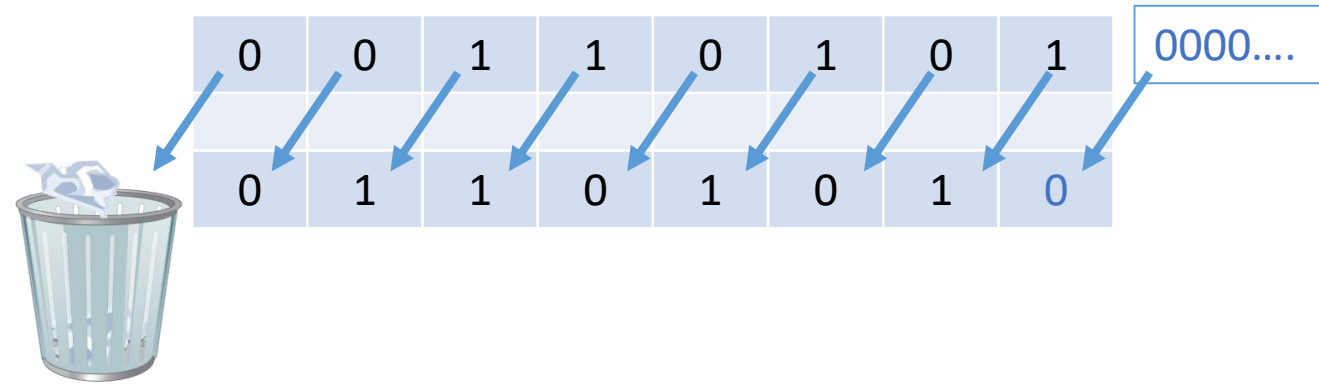
ux<<5 1s 0b11100000 = 224
ux<<7 1s 0b10000000 = 128
x>>1 1s 0b00000111 = 7
x>>2 1s 0b00000011 = 3
x>>5 1s 0b00000000 = 0
x>>7 1s 0b00000000 = 0
y 1s 0b11110000 = -16
y>>1 1s 0b11111000 = -8
y>>2 1s 0b11111100 = -4
y>>5 1s 0b11111111 = -1
y>>7 1s 0b11111111 = -1
uy 1s 0b11110000 = 240
uy>>1 1s 0b01111000 = 120
uy>>2 1s 0b00111100 = 60
uy>>5 1s 0b00000111 = 7
uy>>7 1s 0b00000001 = 1
laptop:~/cs211/demo/expressions>gcc -g -Wall -o shift shift.c
laptop:~/cs211/demo/expressions>./shift
x 1s 0b00001111 = 15
x<<1 1s 0b00011110 = 30
x<<2 1s 0b00111100 = 60
x<<5 1s 0b11100000 = -32
x<<7 1s 0b10000000 = -128
laptop:~/cs211/demo/expressions>
  
```

A subtitle at the bottom of the video frame reads: "If I shift that to the left by one, that's going to move those four bits one to the left".

Bit Shifting Expressions

- Shift Left: <<

```
char x=53;  
char y=x<<1;
```



- Shift Right: >>

```
char x=53;  
char y=x>>1;
```

