Logical and Bitwise Expressions

The truth value will set you free.

iClicker Attendance

Please click on A if you are here:

A. I am here today.

Exercise Review: Ball Motion

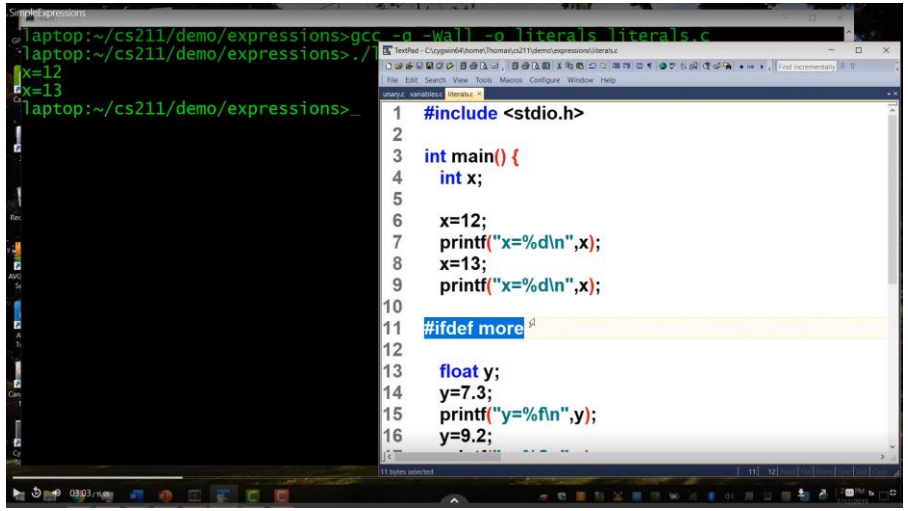
Assume you throw a (frictionless) ball straight up in the air at some initial velocity. With gravity acting at 9.81 m/s^2, after time t, what is the height and velocity of the ball?

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



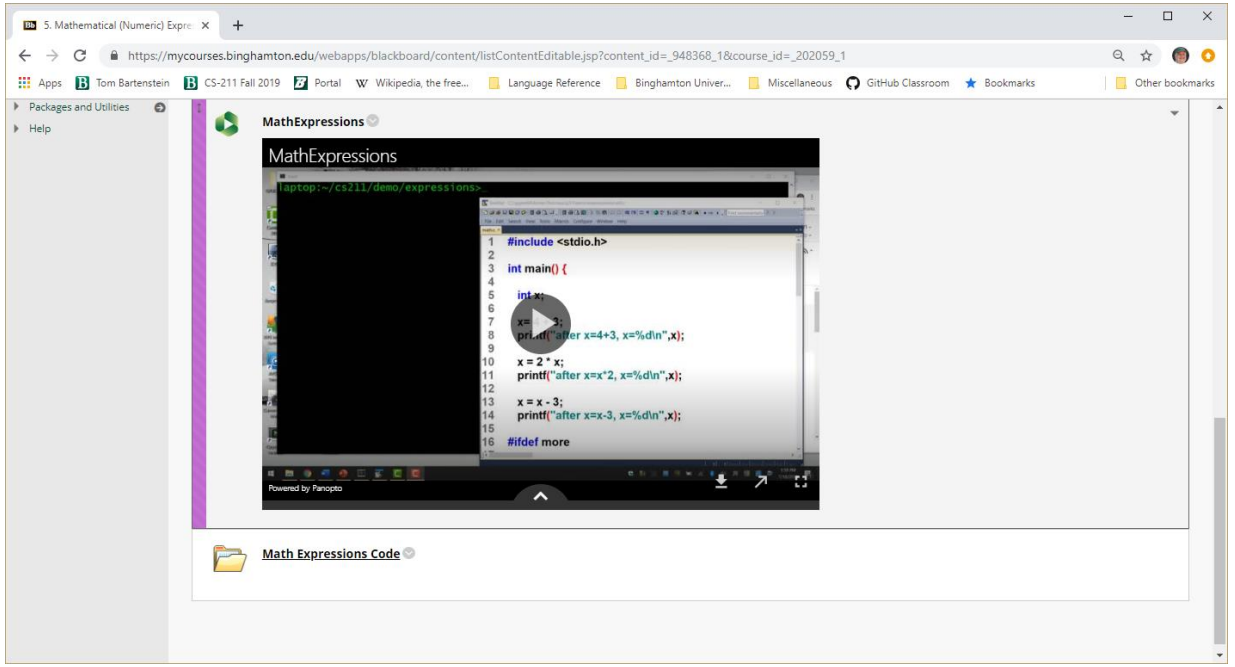
Class Exercise: Assignments

- 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 $h(t) = v_0 t - \frac{gt^2}{2}$

g=9.81 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

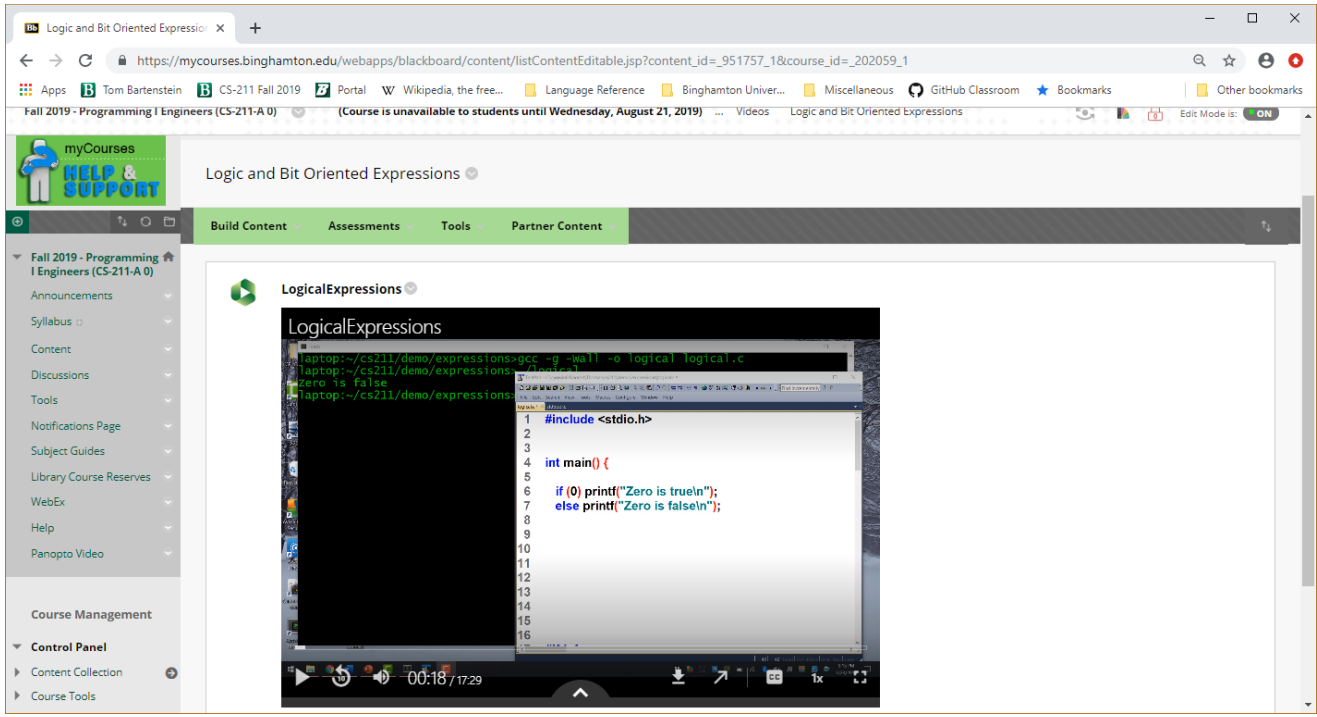
- Expression interpreted as a logical value (true or false)

```
int x=9;

x>10
x==6
(x-9)
x && (113/x < 12)
```

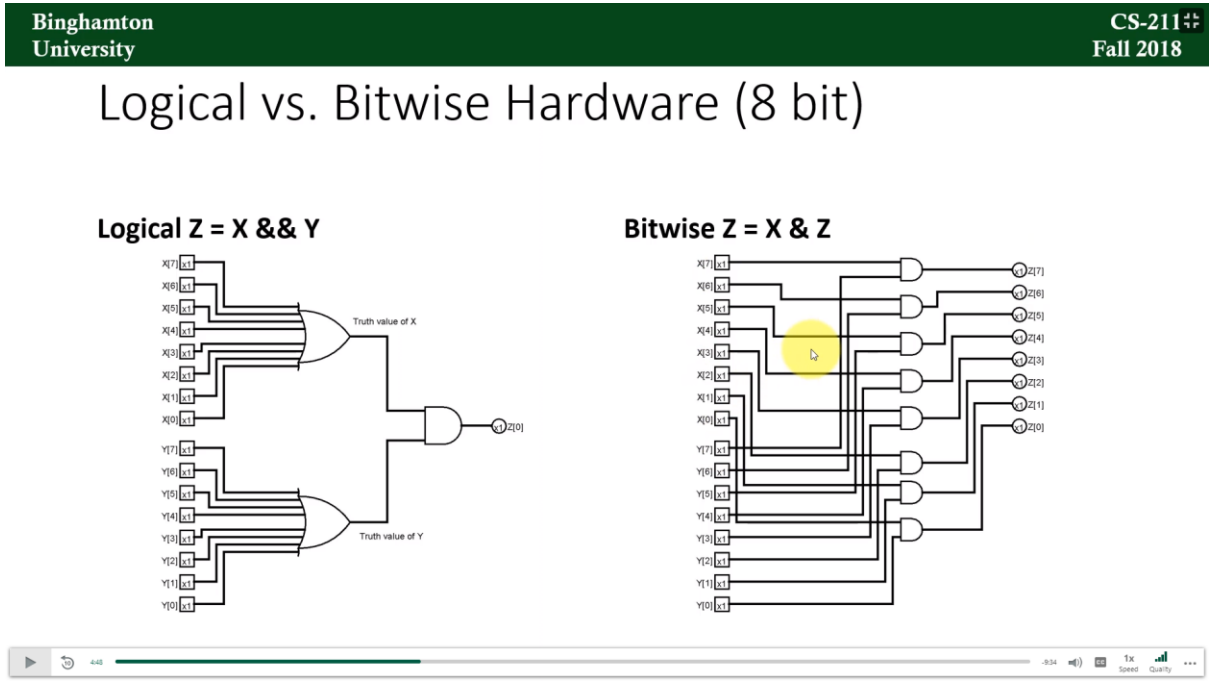
Logical Expressions

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



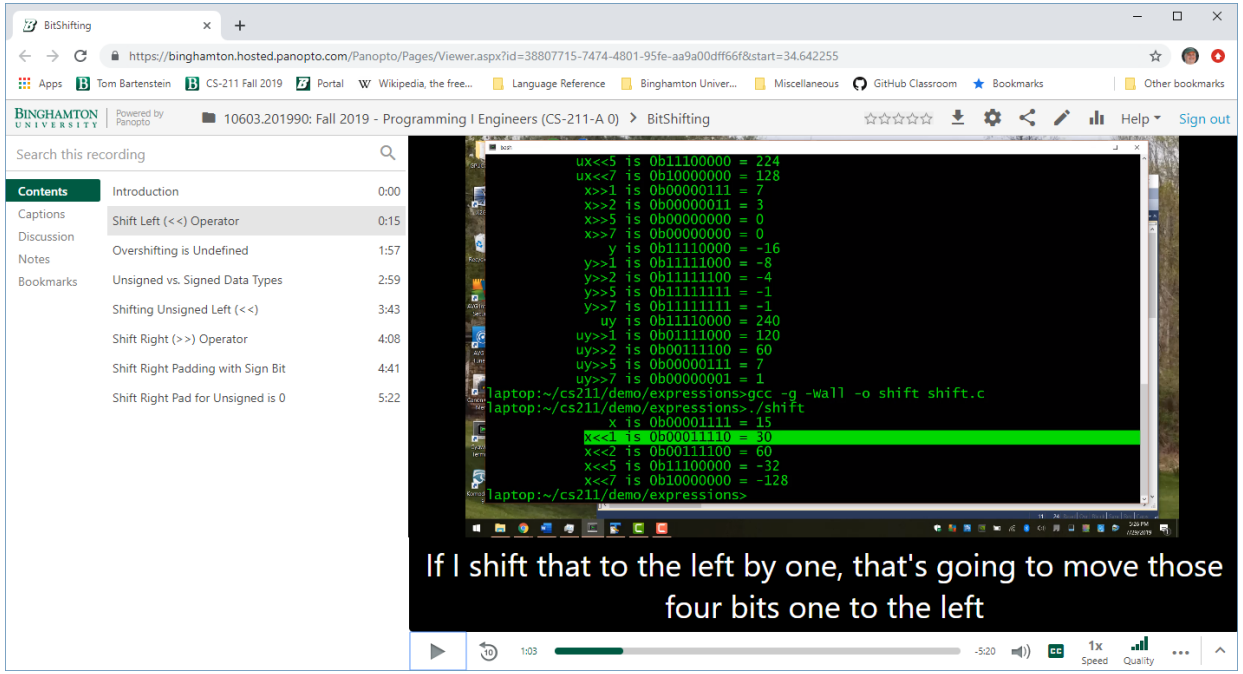
Bit Oriented Expressions

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



Bit Shifting

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



Exercise: Convert to Binary & Hex

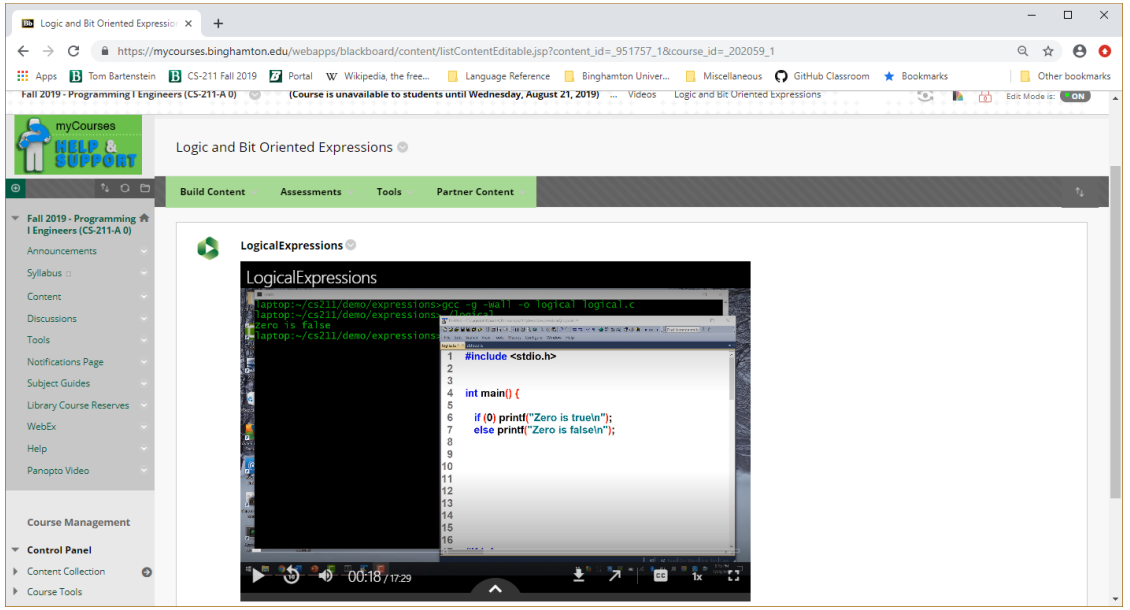
Get an integer number from the user.

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!

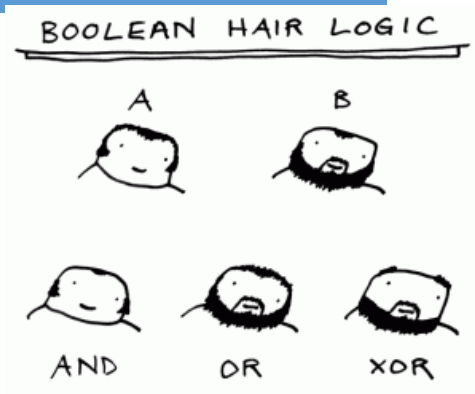
```
int a=16;
if (a=3) printf("The value of a is %d\n",a);

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

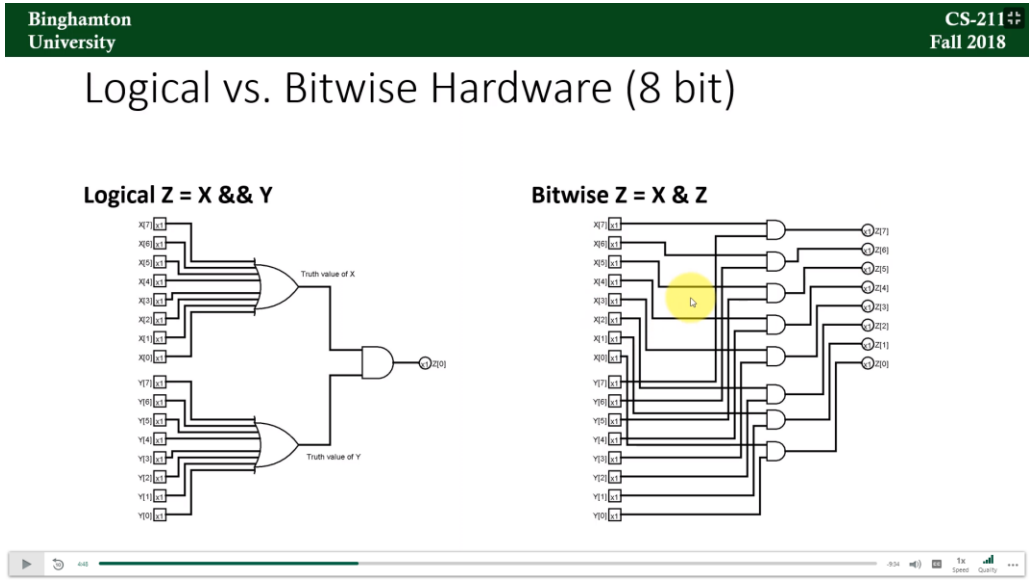
condition ? *true_expr* : *false_expr*

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;
```

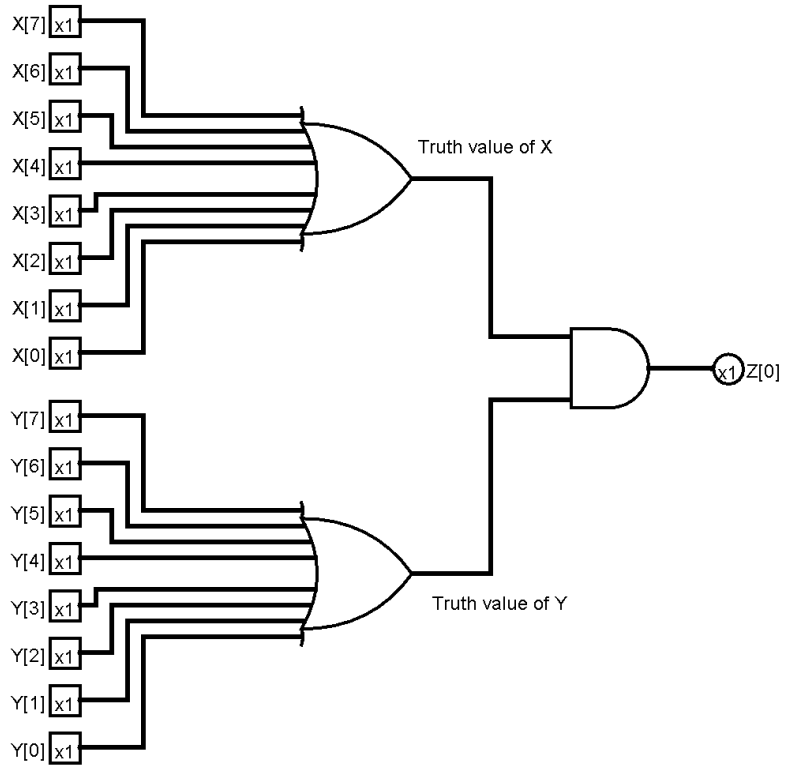


Bit Oriented Expressions

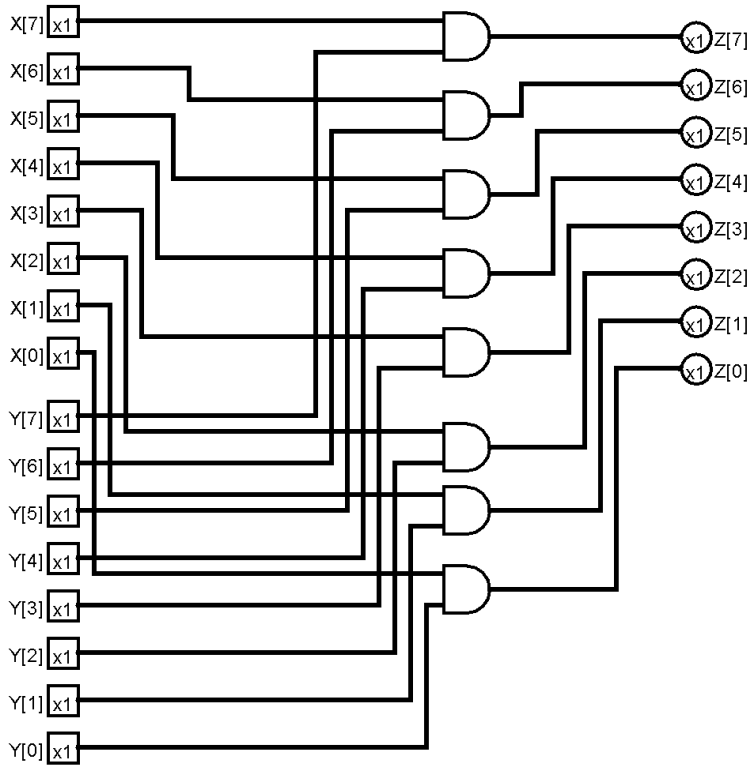
Summary Notes

Logical vs. Bitwise Hardware (8 bit)

Logical Z = X && Y



Bitwise Z = X & Z



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

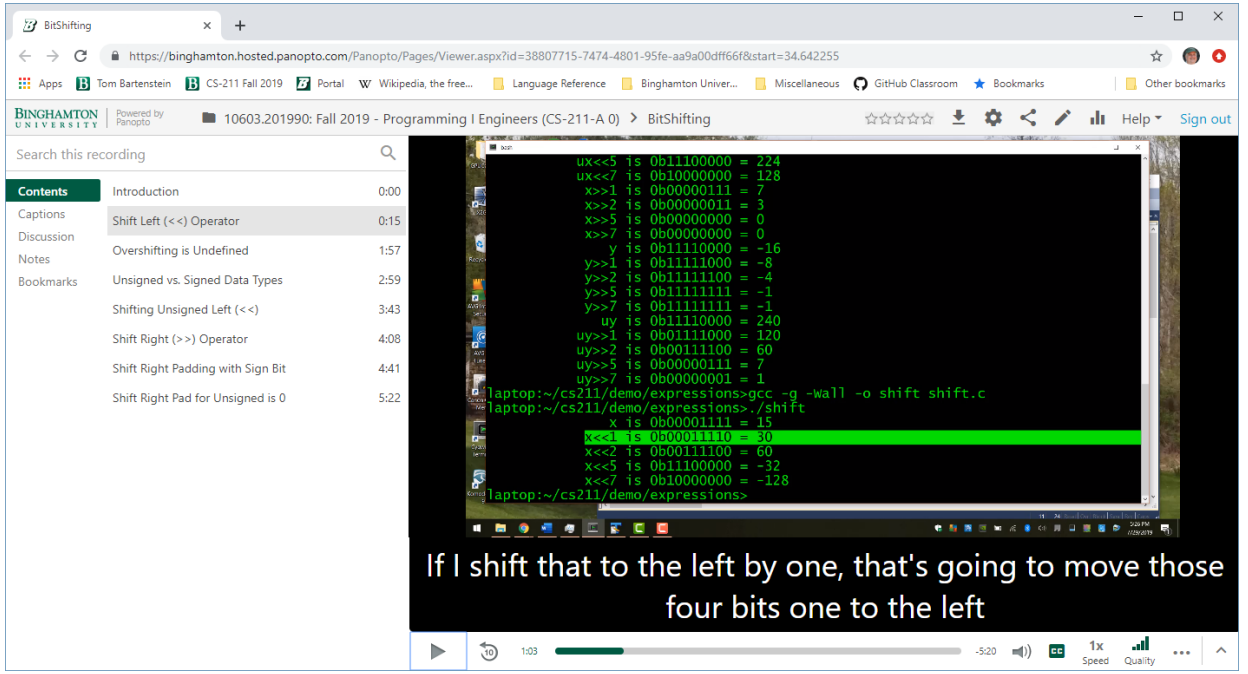
- Each bit in the variable has a single truth value
- BitWise operators work on columns of bits
- A bitwise operators performs multiple logical operations … one for each bit position in the arguments
- Use single operators & | ^

Bitwise Operators

- ~ - “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
- ^ - 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

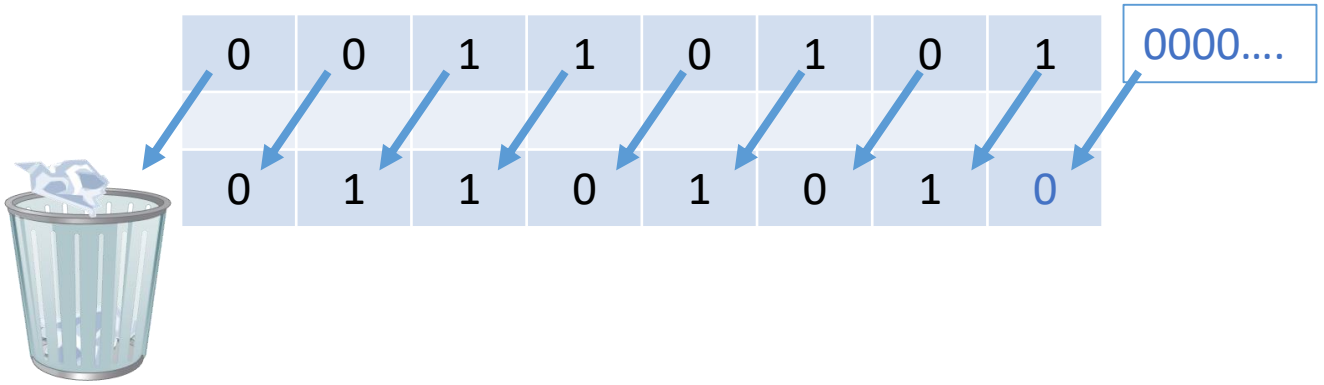
Summary Notes



Bit Shifting Expressions

- Shift Left: <<

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



- Shift Right: >>

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