COMPUTER SYSTEMS AND ORGANIZATION

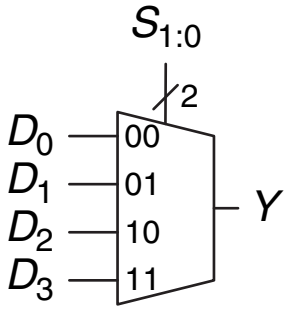
Part 1

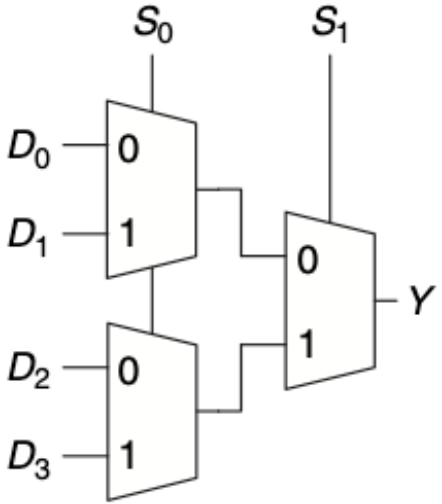
Daniel Graham

UNIVERSITY of VIRGINIA | ENGINEERING

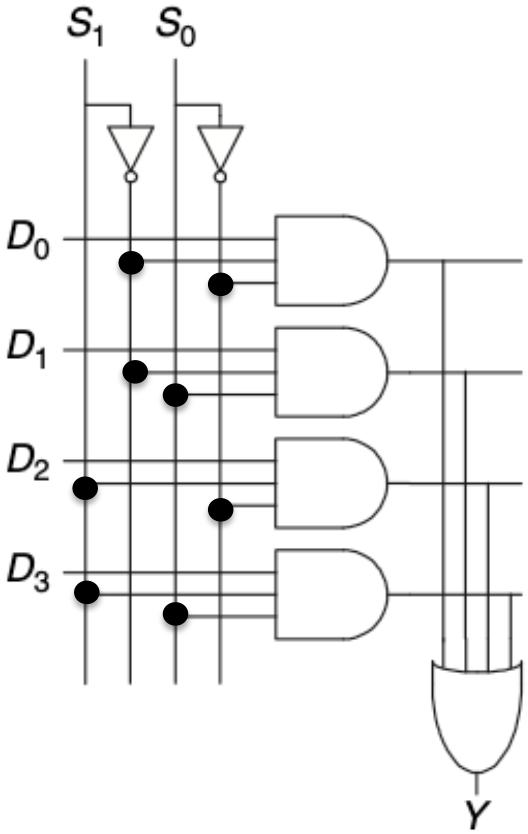
REVIEW

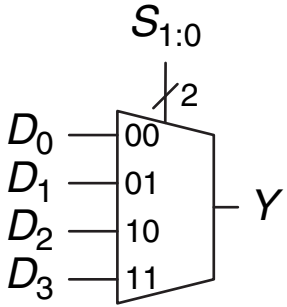
2 BIT MUX





2 BIT MUX



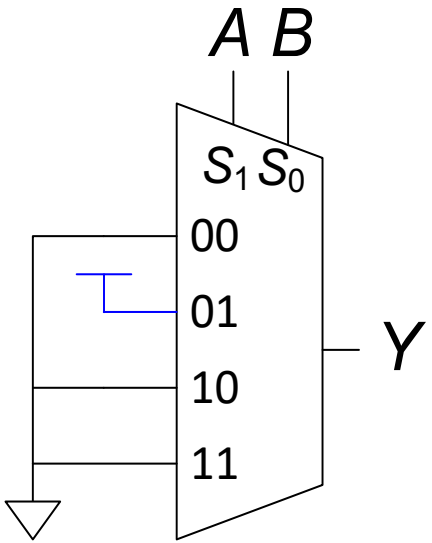


MUX AS A LOOK UP TABLE

Using mux as a **lookup table**

A	B	Y
0	0	0
0	1	1
1	0	0
1	1	0

$Y = AB$

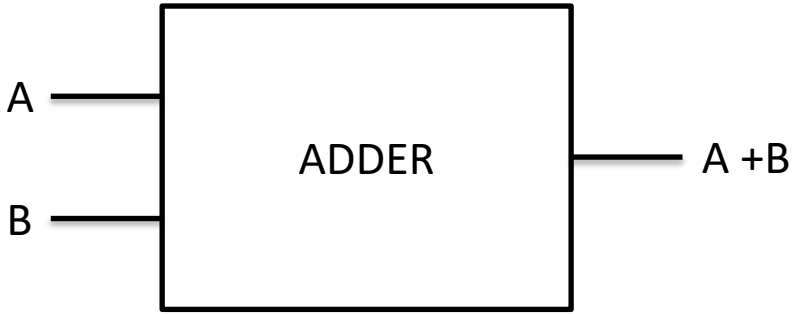


GREAT WE HAVE GATES

NOW LET'S BUILD SOMETHING.

HOW ABOUT A MACHINE THAT ADDS NUMBERS?

THE IDEA



THE CHALLENGE

Our gates only support 0 and 1s.

How can we represent other decimal numbers?

How can we present negative numbers?

What about fractions ☺?

DECIMAL

- Decimal numbers

1000's column
100's column
10's column
1's column

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

five thousands
three hundreds
seven tens
four ones

BINARY

8's column
4's column
2's column
1's column

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

one eight
one four
no two
one one

BINARY CONVERSION EXAMPLES

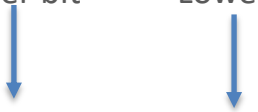
Convert 74 to binary = 110110

74 ÷ 2 = 37 remainder 0
37 ÷ 2 = 13 remainder 1
13 ÷ 2 = 6 remainder 1
6 ÷ 2 = 3 remainder 0
3 ÷ 2 = 1 remainder 1
1 ÷ 2 = 0 remainder 1

Convert 10101 to Decimal

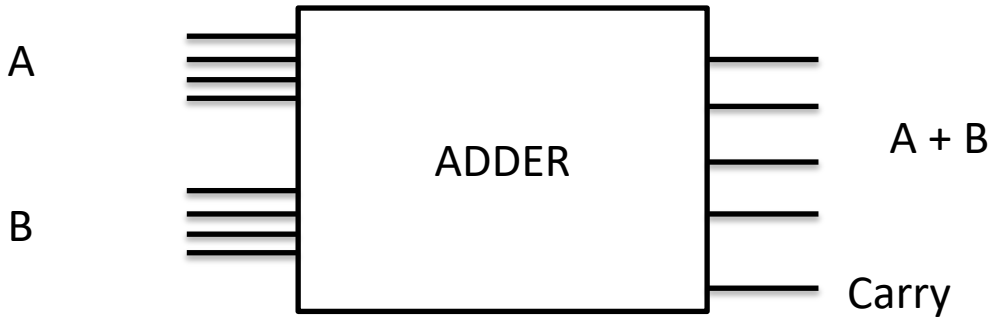
Highest order bit

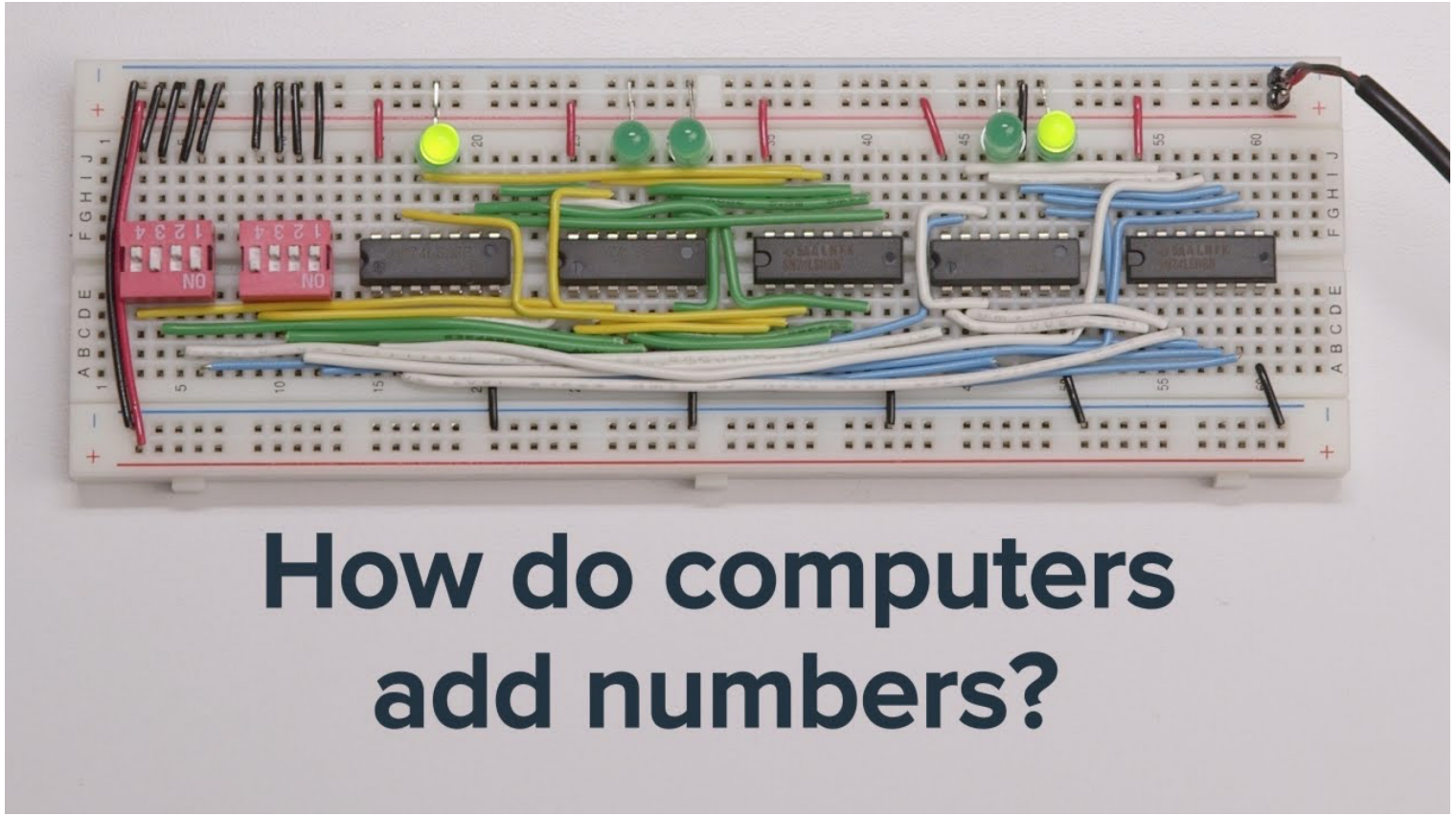
Lowest order bit



10101

4-BIT ADDER



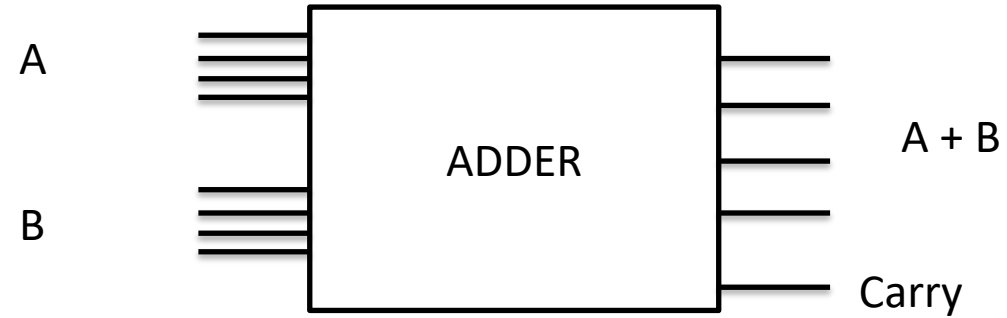

How do computers
add numbers?

INPUTS AND OUTPUT OF OUR ADDER

What would the input be if wanted to add 5, and 9?

Notice we need to pick and order for the wires. More on this later ☺

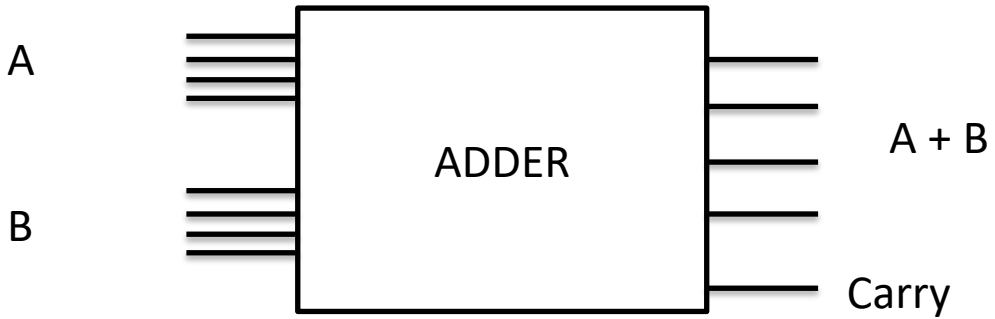
Which output lights would we want to light Up?



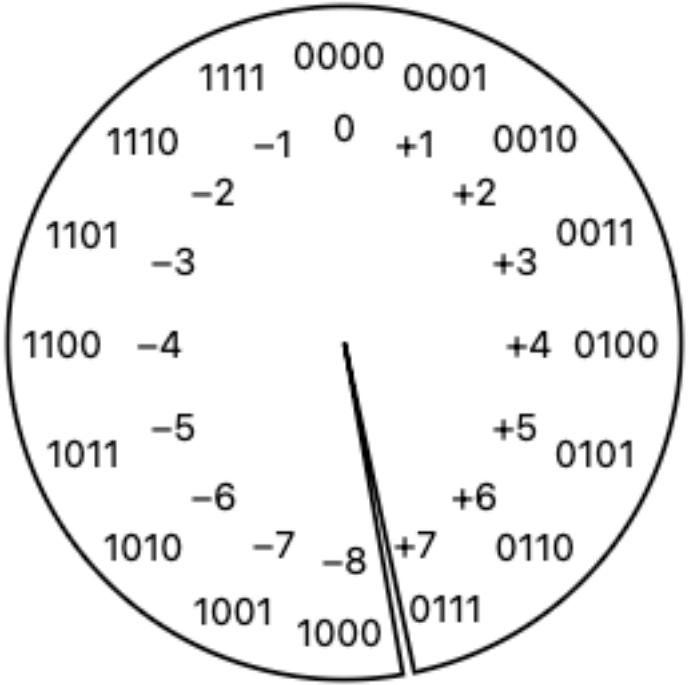
INPUTS AND OUTPUT OF OUR ADDER

What if we now added 7 and 9?

What would our inputs be and which lights do we expect to light up?



TWO COMPLEMENT

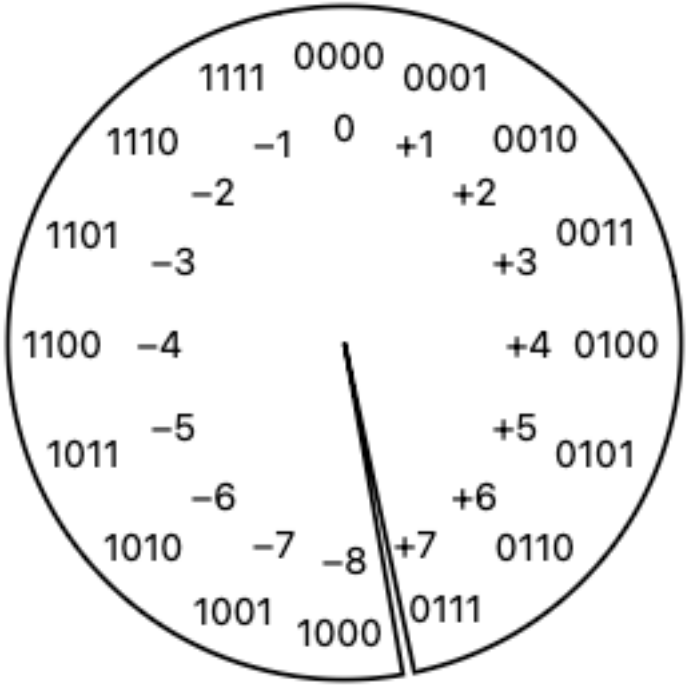


Two's complement picks a number (typically half of the maximum number we can write, rounded up) and decides that that number and all numbers bigger than it are negative

Two's complement is nice because the three most common mathematical operators (addition, subtraction, and multiplication) work the same for signed and unsigned values. Division is messier, but division is always messy

Flip the bits and Add on

TWO COMPLEMENT



Flip the bits and add on trick for converting between positive and negative numbers?

EXAM REVIEW FALL 2018

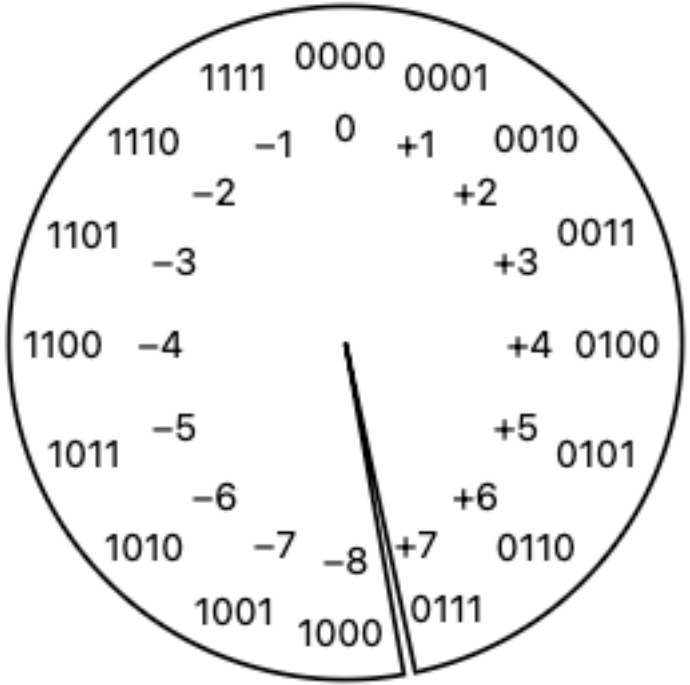
The following assume 8-bit 2's-complement numbers. For each number, bit 0 is the low-order bit, bit 7 is the high-order bit.

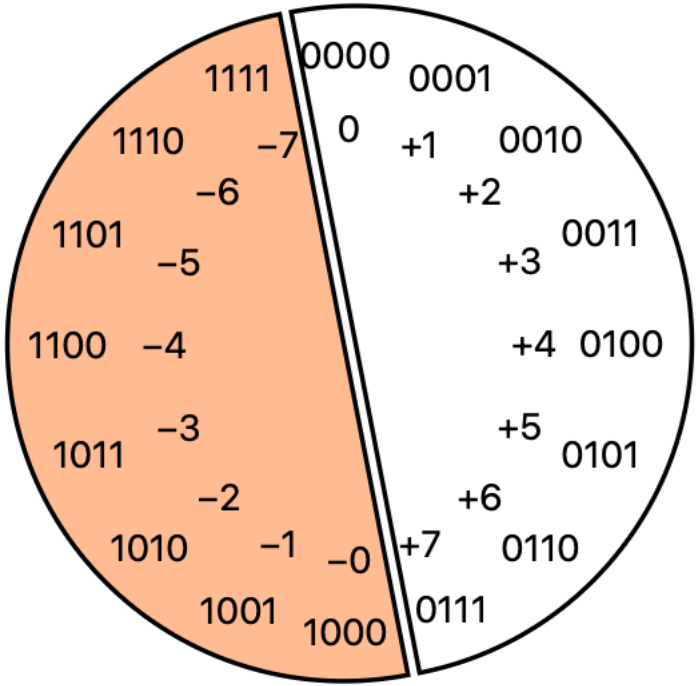
Question 2 [2 pt]: (see above) Complete the following sum, showing your work (carry bits, etc)

```
  0 0 1 1 0 0 1 1
+ 0 1 1 0 0 1 1 0
-----------------
```

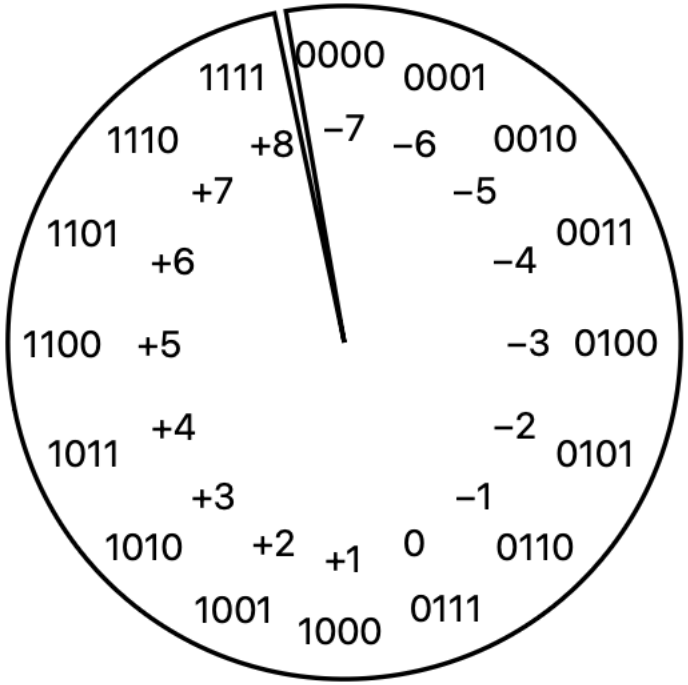
What is the result in base 10? Is it negative or positive? Would you get the same result in decimal if you had more bits ☺ ?

TWOS COMPLEMENT VS SIGN BIT





BIAS



From original number to BIAS

BIAS = FLOOR (MAX_NUM/2)

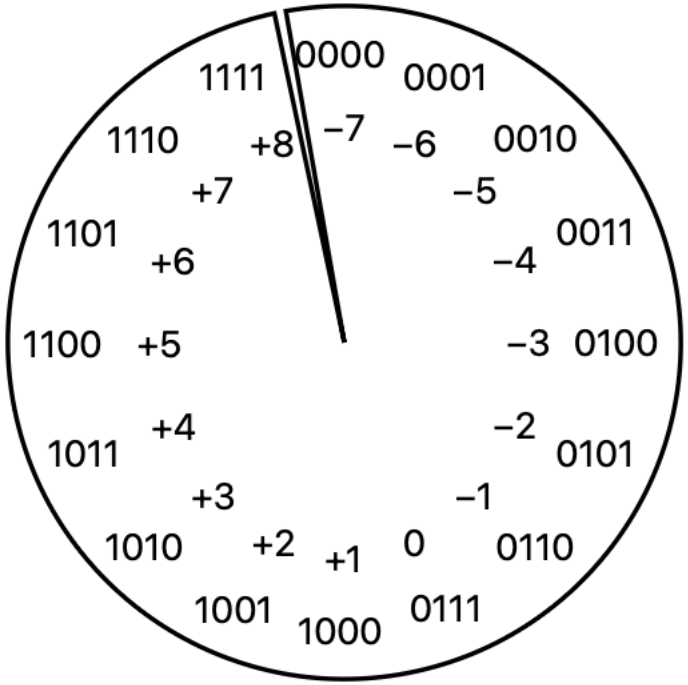
REPRESENTATION = ORGINAL_NUMBER + BIAS

From BIAS to Original

BIAS = FLOOR (MAX_NUM/2)

ORGINAL_NUMBER = REPRESENTATION - BIAS

BIAS EXAMPLE



From original number to BIAS

BIAS = FLOOR (MAX_NUM/2)

REPRESENTATION = ORGINAL_NUMBER + BIAS

Example

BIAS = FLOOR (15/2) = 7

REPRESENTATION = -2 + 7 = 5

WRITING LONG BINARY IS NO FUN.

LET'S EXPRESS IT IN ANOTHER BASE TO MAKE EASIER. DEFINITELY CHOOSE SOMETHING LARGER THAN BASE 10

HEXADECIMAL

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

BASE 8 OCTAL

Convert 67 to octal

67 ÷ 8 = 8 remainder 3
8 ÷ 8 = 1 remainder 0
1 ÷ 8 = 0 remainder 1

103 (octal) to decimal

27 (octal) to decimal

(23) Strange write haha

BITWISE AND &

$$\begin{array}{r} 1100_2 \\ \& \; 0110_2 \\ \hline 0100_2 \end{array}$$

```
#python example
x = 12
y = 6
z = x & y
print(z)
#prints 4
```

BITWISE OR |

$$\begin{array}{r} 1100_2 \\ \mid\ 0110_2 \\ \hline 1110_2 \end{array}$$

```
#python example
x = 12
y = 6
z = x | y
print(z)
#prints 14
```

BITWISE OR XOR ^

$$
\begin{array}{rr}
 & 1100_2 \\
\hat{} & 0110_2 \\
\hline
 & 1010_2
\end{array}
$$

```
#python example
x = 12
y = 6
z = x ^ y
print(z)
#prints 10
```

BIT-WISE RIGHT SHIFT

1101001_2

$>> 3$

1101_2

```
#python example
x = 105
y = x >> 3
print(y)
#prints 15
```

SIGN EXTENSIONS

$11000_2 >> 2 = 11110_2$ With Sign Extension. (The sign bit is copied)

$11000_2 >> 2 = 00110_2$ Without Sign Extension

LEFT SHIFT

$$1101_2 \ll 3$$

$$1101000_2$$

```
#python example
x = 13
y = x << 3
print(y)
#prints 104
```

SHIFTING MULTIPLYING AND DIVIDING BY 2

A left shift is equivalent of multiplying by 2

0001 << 1 = 0010 (2).

0001 << 2 = 0100 (4)

0001 << 3 = 1000 (8)

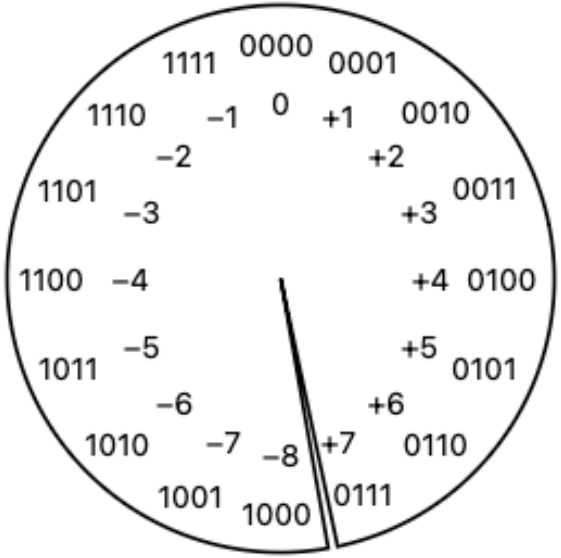
A right shift is equivalent to dividing by 2

01000 >> 1 = 0100 (4)

01000 >> 2 = 0010 (2)

01000 >> 3 = 0001 (2)

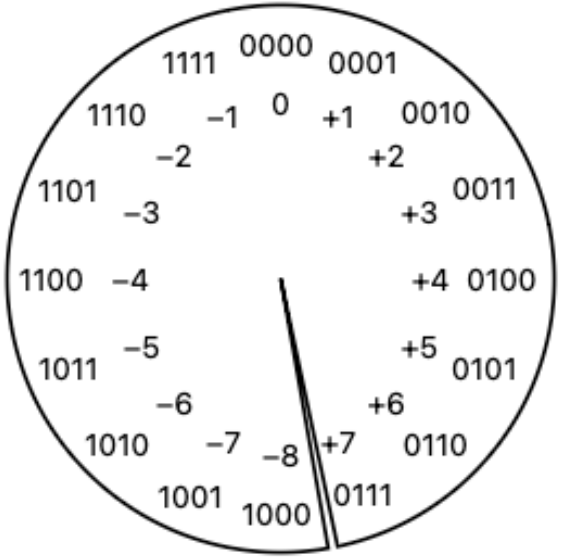
BITWIZE INVERT ~



$$\frac{\sim\ 0000_2}{1111_2}$$

```
#python example
x = 0
z = ~x
print(z)
#prints -1
```

BITWIZE INVERT ~



$$\sim 0110_2$$

$$1001_2$$

```
#python example
x = 6
z = ~x
print(z)
#prints -7
```

SETTING BITS TO 1

Set the last bit of this variable 1

$$\begin{array}{r} 0000_2 \\ | \quad 0001_2 \\ \hline 0001_2 \end{array}$$

```
#python example
x = 0
x = x | 0x01
print(x)
#prints 1
```

SETTING BITS TO 1

Set the third bit of x to 1

$$\begin{array}{r} 0000_2 \\ \mid\ 0100_2 \\ \hline 0100_2 \end{array}$$

```
#python example
x = 0
x = x | 0x04
print(x)
#prints 4
```

Question: What if it was already one?

SETTING BITS TO 1

Set the n bit of x to 1

$$\begin{array}{r} 0000_2 \\ \mid\ 0001_2 \ll 3 \\ \hline 1000_2 \end{array}$$

```
#python example
x = 0
n = 3
x = x | (0x01 << n)
print(x)
#prints 8
```

Question: What if it was already one?

FLIPPING BITS

File the second bit of x. 1 => 0 and 0 => 1

$$
\begin{array}{rr}
 & 1100_2 \\
\wedge & 0010_2 \\
\hline
 & 1110_2
\end{array}
$$

What if the nth bit was 1 instead?

FLIPPING BITS

File the **nth** bit of x. 1 => 0 and 0 => 1

$$\begin{array}{rr} & 1100_2 \\ \wedge & 0010_2 \\ \hline & 1110_2 \end{array}$$

```
#python example
x = 12
n = 1
x = x | (0x01 << n)
print(x)
#prints 14
```

MASKING (EXTRACTING BITS)

The Idea of masking with can extra a certain section of bits by anding.

$$
\begin{array}{rl}
 & 11011100_2 \\
\& & 11110000_2 \\
\hline
 & 11010000_2
\end{array}
$$

Upper 4 bits extracted

MASKING (EXTRACTING BITS)

The Idea of masking with can extra a certain section of bits by anding.

$$
\begin{array}{r}
11011100_2 \\
\& \; 00001111_2 \\
\hline
00001100_2
\end{array}
$$

```
#python example
x = 220
mask = 0x0F
x = x & mask
print(x)
#prints 12
```

Lower 4 bits extracted

MASKING (EXTRACTING BITS)

The Idea of masking with can extra a certain section of bits by anding.

$$
\begin{array}{r}
11011100_2 \\
\& \; 11110000_2 \\
\hline
11010000_2
\end{array}
$$

```
#python example
x = 220
mask = ~0x0F
x = x & mask
print(x)
#prints 208
```

Upper 4 bits extracted

EXAM QUESTION FALL 2018 EXAM 1

Information for questions 6–11

Each question gives two expressions of 32-bit two's-compliment integers `x` and `y`. If the two are equivalent for all `x` and `y`, write "same"; otherwise, write an example `x` (and `y` if used in the expressions) for which the two are different.

———— add example

Question 7 [2 pt]: (see above)

`(x<<2) + (x>>1)` and `((x<<3) + x)>>1`

Question 8 [2 pt]: (see above)

`x | (x>>1)` and `x ^ (x>>1)`

DIVIDE AND CONQUER

PARALLEL EVALUATION

COMBINING