COMPUTER SYSTEMS AND ORGANIZATION

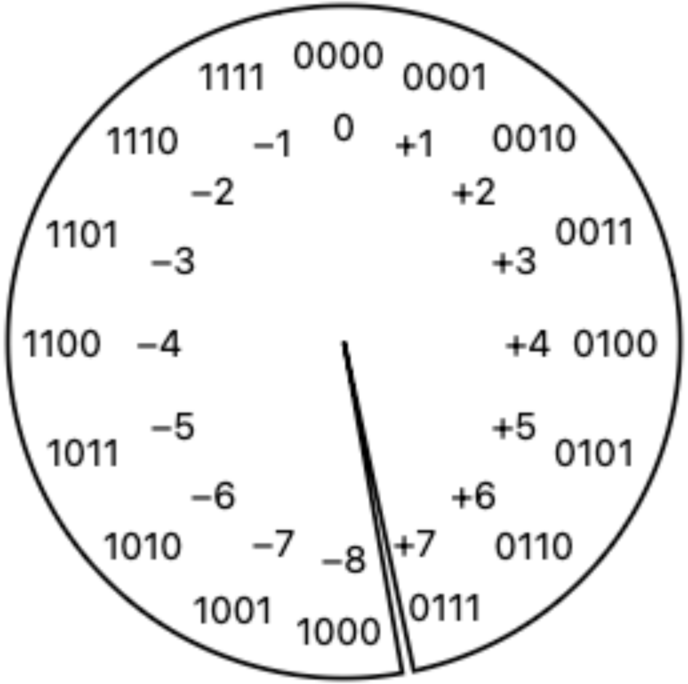
Bitwise Operations

Daniel Graham

University of Virginia | Engineering

REVIEW

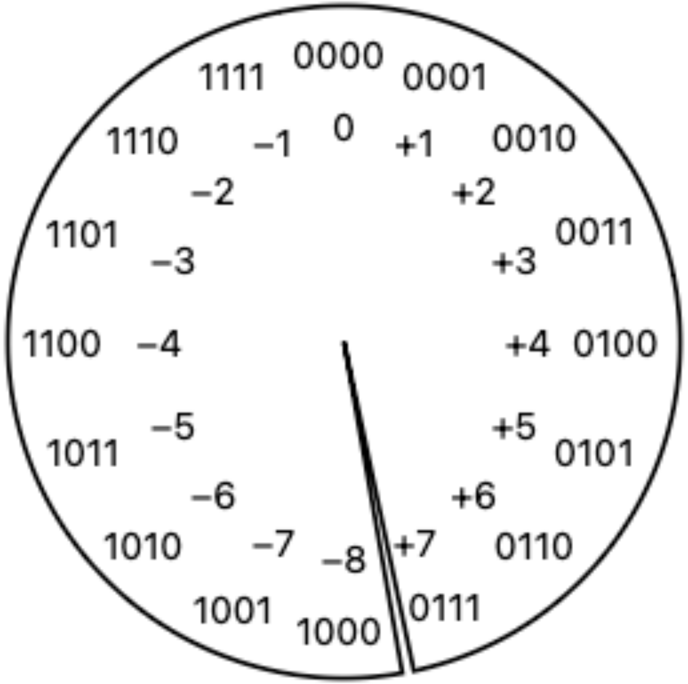
TWO COMPLEMENT



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 and including it are negative

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 ☺ ?

DEFINING OVERFLOW

If the sum of **two positive numbers** yields a **negative result**, the sum has **overflowed.**

If the sum of **two negative numbers** yields **a positive result**, the sum has **overflowed**.

Otherwise, the sum has not overflowed.

Over only exist for operation on signed numbers.

$$
\begin{array}{rl}
0111_2 & +7 \\
\&\ 0001_2 & +1 \\
\hline
1000_2 & -8
\end{array}
$$

NOT OVERFLOW

If the sum of **two positive numbers** yields a **negative result**, the sum has **overflowed.**

If the sum of **two negative numbers** yields **a positive result**, the sum has **overflowed**.

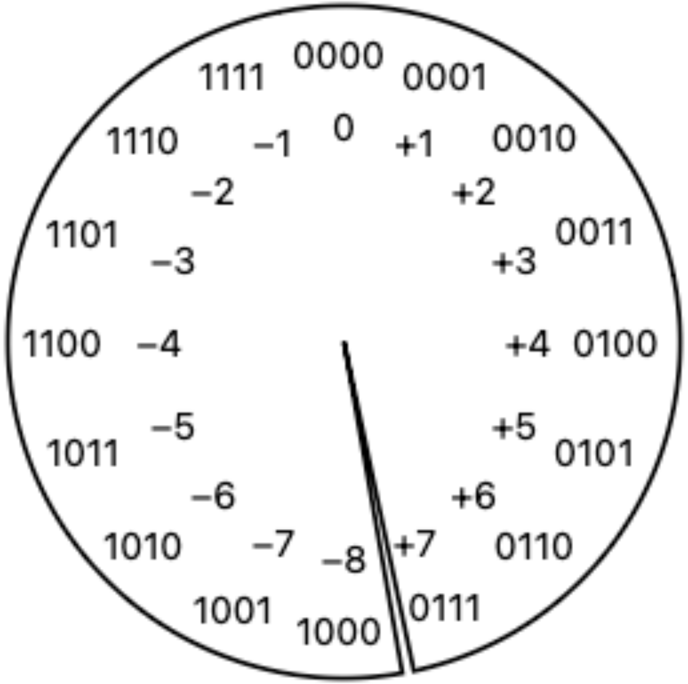
Otherwise, the sum has not overflowed.

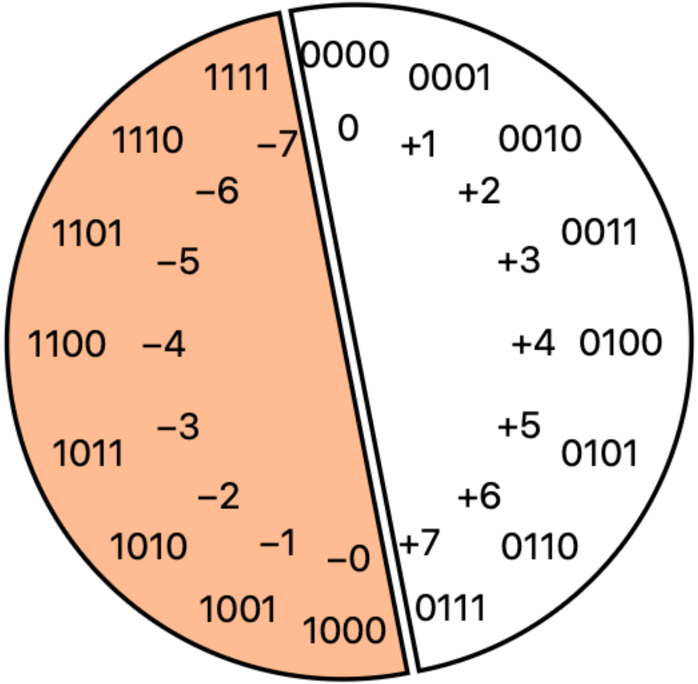
Over only exist for operation on signed numbers.

$$
\begin{array}{rl}
1111_2 & -1 \\
\&\ 0001_2 & +1 \\
\hline
10000_2 &
\end{array}
$$

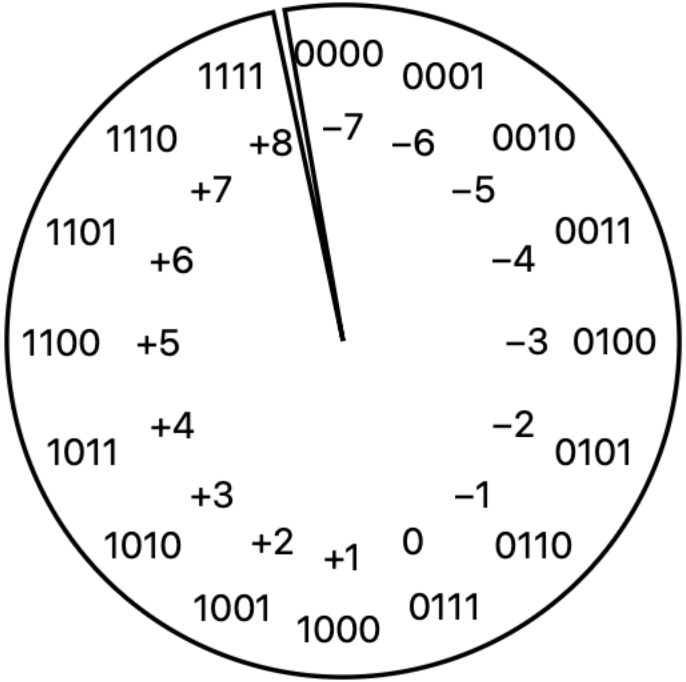
Carry Ignored, But NOT over considered overflow. The answer is correctly zero

TWOS COMPLEMENT VS SIGN BIT





BIAS



From original number to BIAS

BIAS = FLOOR (MAX_NUM/2)

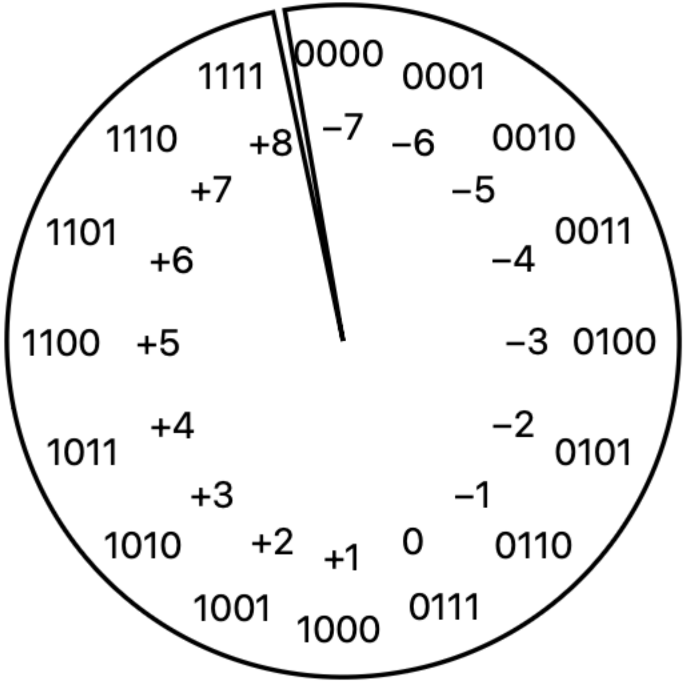
REPRESENTATION = ORGINAL_NUMBER + BIAS

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

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

HEXADECIMAL

Convert 00101110 to hexadecimal Answer: 2E

Group them

0010 = 2

1110 = E

Final 0x2E

- Some programming languages uses prefixes
  - Hex: 0x
    - 0x23AB = $23AB_{16}$
  - Binary: 0b
    - 0b1101 = 1101_2

UNIVERSITY of VIRGINIA | ENGINEERING

BASE 8 OCTAL

Convert 67 to octal

$67 \div 8 = 8$ remainder 3
$8 \div 8 = 1$ remainder 0
$1 \div 8 = 0$ remainder 1

103 (octal) to decimal

27 (octal) to decimal

(23) Strange write haha

Page 4: Binary

4. [6 points] Convert `219` into binary.

Answer

5. [6 points] What is `0b101100110111` in hexadecimal?

Answer

Page 4: Binary

4. [6 points] Convert `219` into binary.

128 64 32 16 8 4 2 1

$1\ 1\ 0\ 1\ 1\ 0\ 1\ 1_2$

Answer

11011011

5. [6 points] What is `0b101100110111` in hexadecimal?

$\boxed{1011}\boxed{0011}\boxed{0111}$

B 3 7

Answer

0xB37

TODAY'S LECTURE

BITWISE AND &

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

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

BITWISE OR |

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

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


UNIVERSITY of VIRGINIA | ENGINEERING

BITWISE OR XOR ^

$$
\begin{array}{r}
1100_2 \\
\wedge\ \ 0110_2 \\
\hline
1010_2
\end{array}
$$

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

BIT-WISE RIGHT SHIFT

$$\begin{array}{r} 1101001_2 \\ \gg 3 \\ \hline 1101_2 \end{array}$$

```
#python example
x = 105
y = x >> 3
print(y)
#prints 13
```

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

$$\begin{array}{r} 1101_2 \\ \ll 3 \\ \hline 1101000_2 \end{array}$$

```
#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)

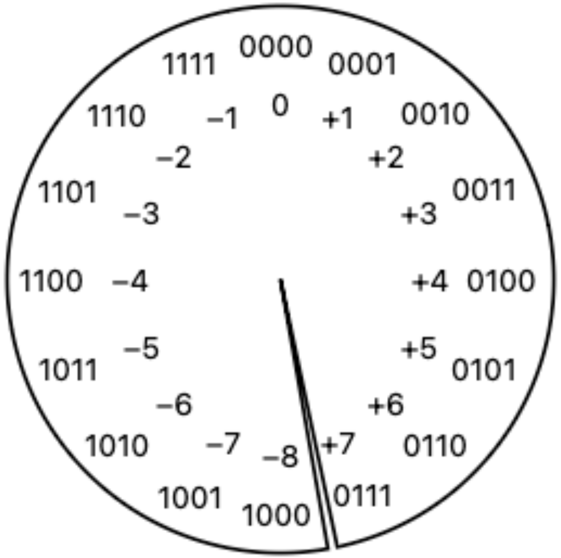
A right shift is equivalent to dividing by 2

01000 >> 1 = 0100 (4)

01000 >> 2 = 0010 (2)

01000 >> 3 = 0001 (2)

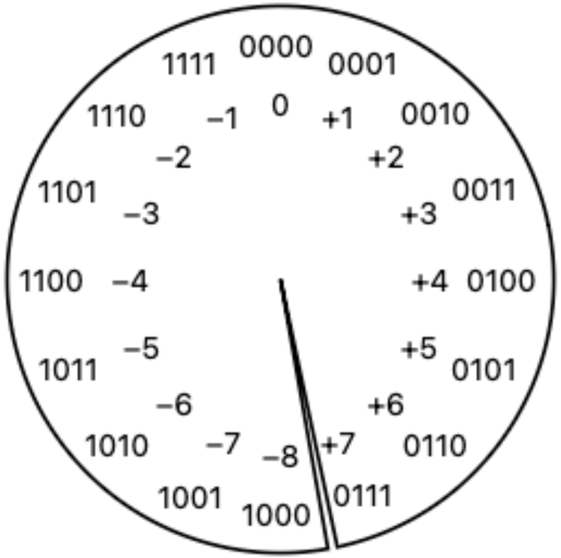
BITWIZE INVERT ~



$$\begin{array}{r} \sim\ 0000_2 \\ \hline 1111_2 \end{array}$$

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

BITWIZE INVERT ~



$$\begin{array}{r} \sim\ 0110_2 \\ \hline 1001_2 \end{array}$$

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

SETTING BITS TO 1

Set the last bit of this variable 1

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

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

SETTING BITS TO 1

Set the third bit of x to 1

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

```
#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

$$
\begin{array}{r}
1100_2 \\
\wedge \quad 0010_2 \\
\hline
1110_2
\end{array}
$$

What if the nth bit was 1 instead?

FLIPPING BITS

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

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

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


UNIVERSITY of VIRGINIA | ENGINEERING

MASKING (EXTRACTING BITS)

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

$$
\begin{array}{r}
11011100_2 \\
\&\ 11110000_2 \\
\hline
\boxed{1101}0000_2
\end{array}
$$

Upper 4 bits extracted

MASKING (EXTRACTING BITS)

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

$$
\begin{array}{r}
11011100_2 \\
\&\ 00001111_2 \\
\hline
0000\boxed{1100_2}
\end{array}
$$

```
#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.

$$
\begin{array}{r}
11011100_2 \\
\&\ 11110000_2 \\
\hline
\boxed{1101}0000_2
\end{array}
$$

Upper 4 bits extracted

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


UNIVERSITY of VIRGINIA ENGINEERING

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)`

COMBINING

We can also set multiple bits simultaneously

$$
\begin{array}{rr}
 & 10100000_2 \\
| & 00001111_2 \\
\hline
 & 10101100_2
\end{array}
$$

```python
#python example
b = 0x0F
a = 0xA0
x = a | b
print(hex(x))
#prints 0xAF
```

PARITY

Suppose you want to want to calculate the even parity of x.

If the number of one's bit is number is odd the parity value is 1, otherwise it is zero

0010 parity bit is 1

0110 parity bit is 0

```
parity = 0 repeat 32 times:
    parity ^= (x&1)
     x >>= 1
```

PARALLEL EVALUATION

Observe that xor is both transitive and associative; thus we can re-write

$x0 \oplus x1 \oplus x2 \oplus x3 \oplus x4 \oplus x5 \oplus x6 \oplus x7$

using transitivity as

$x0 \oplus x4 \oplus x1 \oplus x5 \oplus x2 \oplus x6 \oplus x3 \oplus x7$

and using associativity as

$(x0 \oplus x4) \oplus (x1 \oplus x5) \oplus (x2 \oplus x6) \oplus (x3 \oplus x7)$

and then compute the contents of all the parentheses at once via

`x ^ (x>>4)`.

PARALLEL EVALUATION

$x0 \oplus x1 \oplus x2 \oplus x3 \oplus x4 \oplus x5 \oplus x6 \oplus x7$

using transitivity as

$x0 \oplus x4 \oplus x1 \oplus x5 \oplus x2 \oplus x6 \oplus x3 \oplus x7$

and using associativity as

$(x0 \oplus x4) \oplus (x1 \oplus x5) \oplus (x2 \oplus x6) \oplus (x3 \oplus x7)$

and then compute all at once via

x ^ (x>>4).

```
x ^= (x>>16)
x ^= (x>>8)
x ^= (x>>4)
x ^= (x>>2)
x ^= (x>>1)
parity = (x & 1)
```

UNIVERSITY of VIRGINIA | ENGINEERING