Last class we talked

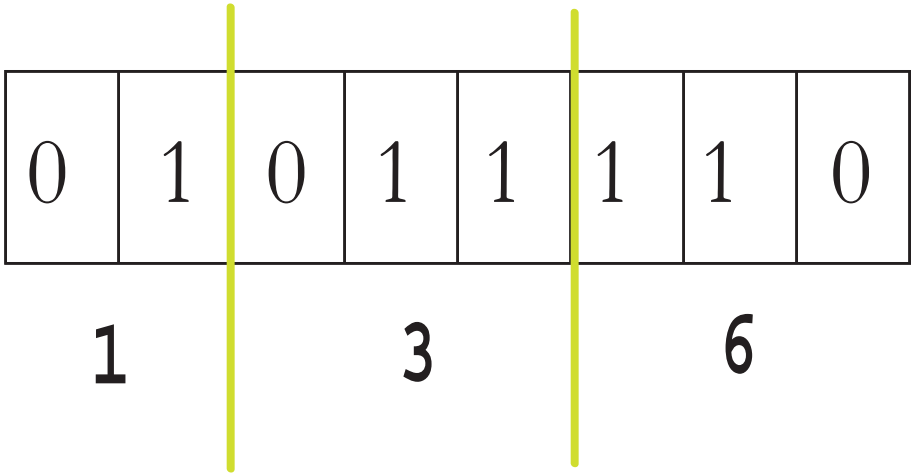
Encoding numbers

Integers:

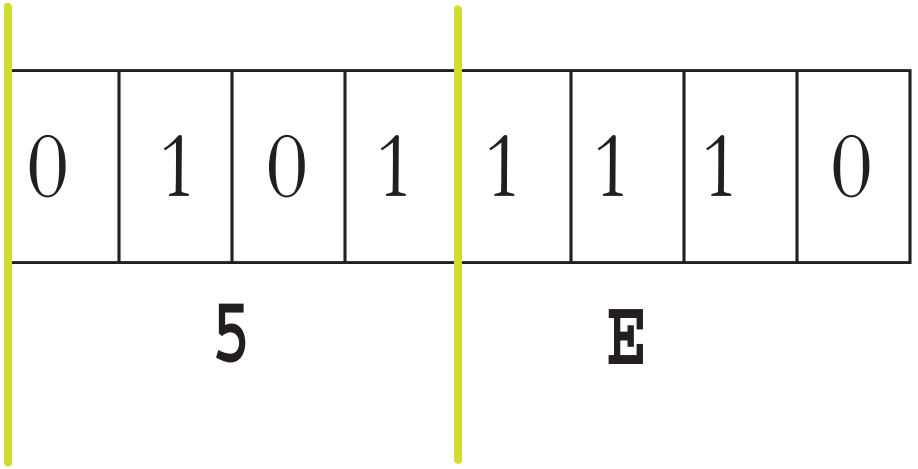
$$V = \sum_{i=0}^{N-1} 2^i b_i$$

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
0	1	0	1	1	1	1	0	$= 94_{10}$

$$0 \bullet 2^7 + 1 \bullet 2^6 + 0 \bullet 2^5 + 1 \bullet 2^4 + 1 \bullet 2^3 + 1 \bullet 2^2 + 1 \bullet 2^1 + 0 \bullet 2^0$$



Octal base-8: `0136`



Hexadecimal base-16: `0X5E`

Encoding Information

8-bit ASCII code

8-BIT ASCII Code represents characters: {A-Z},{a-z},{1,2,3,4,5,6,7,8,9,0},{symbols, signs}

```
0011 0101 ~ '5';     0011 1001 ~ '9'
```

See Figure 2.21 ASCII representation of characters (page 91)

UNICODE uses 16-bit to encode international characters

See Figure 2.22 Examu652ia(x)T □ 3.23976 0 Td□ (a)T □ (m)T □ (u652ia(x)T □ 3.23976 0 V023976 0.91

Two's comple
Representat

- Subtract large number from s

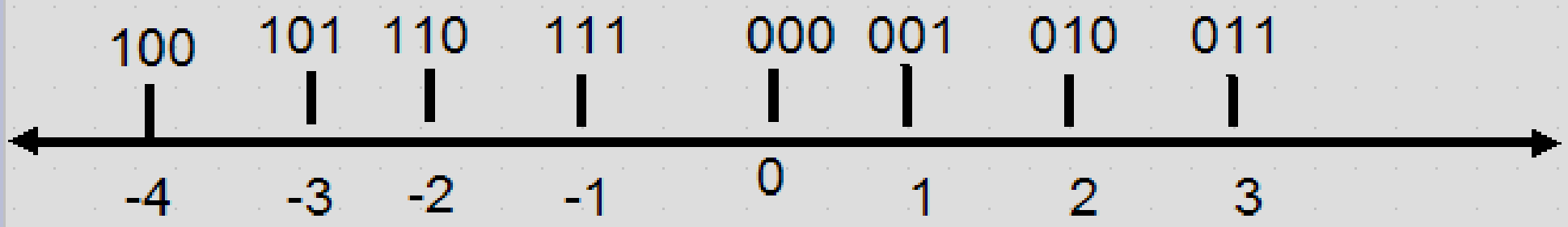
Borrow from 0s:

0	0	0	0	0	1	1	1

0	0	0	0	1	0	0	0

1	1	1	1	1	1	1	1

- Adding 1 moves to the right one position, -1 moves to the left!
- For fixed N it is a closed loop
- Adding 1 to the largest positive number results in the largest negative



$1111\ ...\ 1111\ \ 1111\ \ 1111\ \ 1111_{two} = -1_{ten}$

$0000\ ...\ 0000\ \ 0000\ \ 0000\ \ 0000_{two} = 0_{ten}$

$0000\ ...\ 0000\ \ 0000\ \ 0000\ \ 0010_{two} = 2_{ten}$

Largest positive number:

$0111\ ...\ 1111\ \ 1111\ \ 1111\ \ 1111_{two} = 2{,}147{,}483{,}647_{ten}$

Largest negative number:

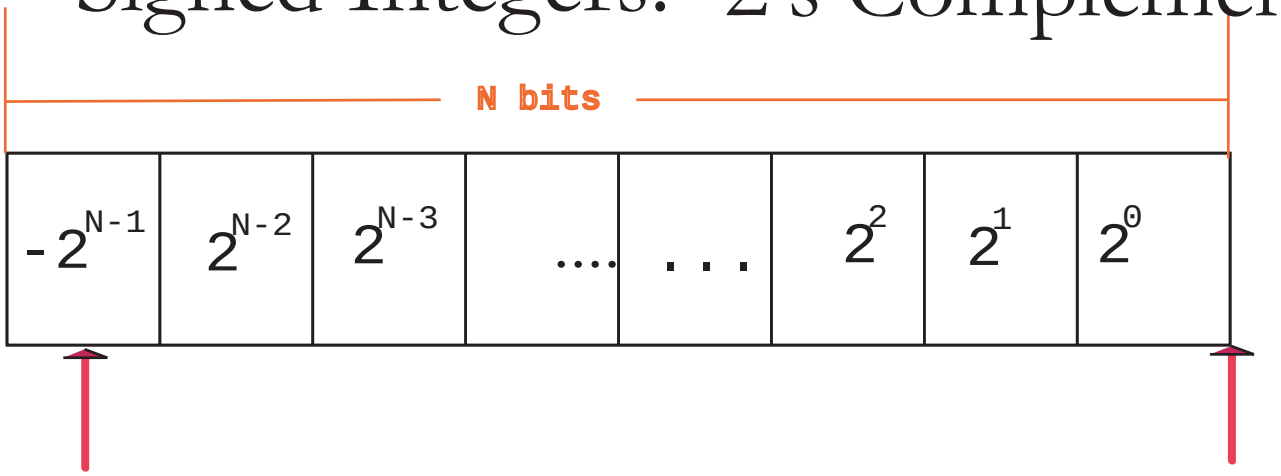
$1000\ ...\ 0000\ \ 0000\ \ 0000\ \ 0010_{two} = -2{,}147{,}483{,}646_{ten}$

Please read
CH.: 3.1, 3.2,3.3 .
Exercises p.229, 3.1-3.6,

Encoding Information

Encoding numbers

Signed Integers: 2's Complement representation



$11000101 = -2^{7} + 2^{6} + 2^{2} + 2^{0} = -128+64+4+1 = -59$

00111010
+1
00111011 = -59

The same binary addition procedure will work with both positive and negative numbers

There is no need for special subtraction operation

Range: -2^{N-1} to $+2^{N-1} - 1$

SIGN EXTENSION

X = 1111 1111 1111 1111 1111 1111 1111 1111$_{two}$

Y = 1111 1111 1111 1111$_{two}$

X>Y How to compare? X and Y have different number of bits.

copy thew most significant bit of the smaller number to fill missing bits

Y = 1111 1111 1111 1111 1111 1111 1111 1111$_{two}$

If a positive number:

Z = 0111 1111 1111 1111$_{two}$

Z = 0000 0000 0000 0000 0111 1111 1111 1111$_{two}$

QUESTION

X = 1111 1111 1111 1111 1111 1111 1111 1111_{two}

Y = 0111 1111 1111 1111 1111 1111 1111 1110_{two}

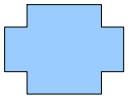
IF(X>Y) x,y 2's complement signed representeation.

IF(X>Y) x,y unsigned representeation.

OVERFLOW

Given 32 bit word

$0111 \ldots 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$



$0000 \ldots 0000\ 0000\ 0000\ 0001_{two} = 1_{ten}$

?

HW 2.Please write a program to detect overflow (in any language).

Largest positive number:

$0111 \ldots 1111\ 1111\ 1111\ 1111_{two} = 2{,}147{,}483{,}647_{ten}$

Largest negative number:

$1000 \ldots 0000\ 0000\ 0000\ 0010_{two} = -2{,}147{,}483{,}646_{ten}$

Encoding Information

ASCII versus Binary numbers

int k = 2147483647;

This is a **32** bit signed integer in C

long long k = 18446744073709551615;

This is a **64** bit signed integer in C

How do you store integer 18446744073709551619 ? in C

You can store it as a string "18446744073709551619" ? in C

Integer myint = new Integer("18446744073709551619"); in JAVA

Lets compare the number of bits in each case.

Encoding Information

Binary Coded Decimal (BCD)

Fixed-length codes

4-BIT Binary Code Decimal(BCD)- represents decimal digits{0,1,2,3,4,5,6,7,8,9,0}

```
0101 ~ 5;     1001 ~ 9
```

1237 -> 0001 0010 0011 0111

16 bits are used to represent 1237