INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

CSN-101 (Introduction to Computer Science and Engineering)

Lecture 20: Problem Solving using Computers, Binary Number System

Dr. Sudip Roy

Assistant Professor

Department of Computer Science and Engineering

Piazza Class Room: https://piazza.com/iitr.ac.in/fall2019/csn101

[Access Code: csn101@2019]

Moodle Submission Site: https://moodle.iitr.ac.in/course/view.php?id=45

[Enrollment Key: csn101@2019]

Plan for Lecture Classes in CSN-101 (Autumn, 2019-2020)

Week	Lecture 1 (Monday 4-5 PM)	Lecture 2 (Friday 5-6 PM)
1	Evolution of Computer Hardware and Moore's Law, Software and Hardware in a Computer	Computer Structure and Components, Operating Systems
2	Computer Hardware: Block Diagrams, List of Components	Computer Hardware: List of Components, Working Principles in Brief, Organization of a Computer System
3	Linux OS	Linux OS
4	Writing Pseudo-codes for Algorithms to Solve Computational Problems	Writing Pseudo-codes for Algorithms to Solve Computational Problems
5	Sorting Algorithms – Bubble sort, selection sort, and Search Algorithms	Sorting Algorithms – Bubble sort, selection sort, and Search Algorithms
6	C Programming	C Programming
7	Number Systems: Binary, Octal, Hexadecimal, Conversions among them	Number Systems: Binary, Octal, Hexadecimal, Conversions among them
8	Number Systems: Negative number representation, Fractional (Real) number representation	Boolean Logic: Boolean Logic Basics, De Morgan's Theorem, Logic Gates: AND, OR, NOT, NOR, NAND, XOR, XNOR, Truth-tables
9	Computer Networking and Web Technologies: Basic concepts of networking, bandwidth, throughput	Computer Networking and Web Technologies: Basic concepts of networking, bandwidth, throughput
10	Different layers of networking, Network components, Type of networks	Network topologies, MAC, IP Addresses, DNS, URL
11	Different fields of CSE: Computer Architecture and Chip Design	Different fields of CSE: Data Structures, Algorithms and Programming Languages
12	Different fields of CSE: Database management	Different fields of CSE: Operating systems and System softwares
13	Different fields of CSE: Computer Networking, HPCs, Web technologies	Different Applications of CSE: Image Processing, CV, ML, DL
14	Different Applications of CSE: Data mining, Computaional Geometry, Cryptography, Information Security	Different Applications of CSE: Cyber-physical systems and IoTs

ETE

ETE

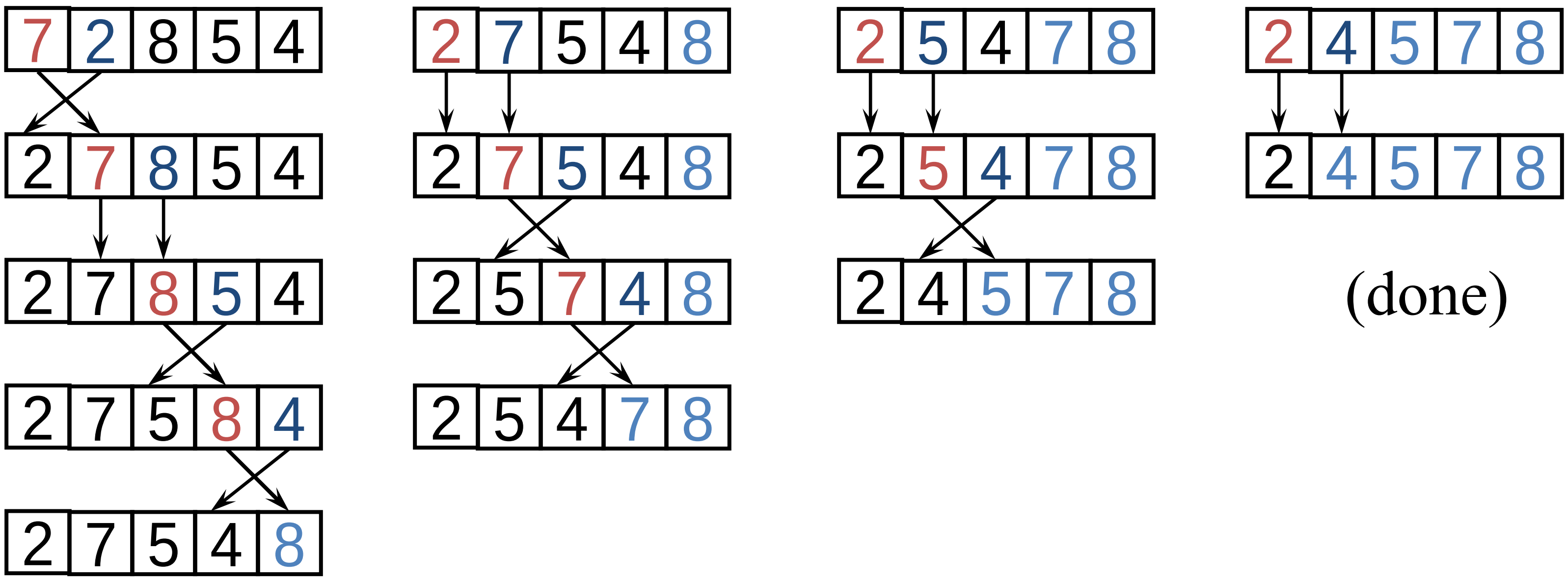
Term Project

Some Examples: Problem Solving with Computers

Bubble sort

- Compare each element (except the last one) with its neighbor to the right
  - If they are out of order, swap them
  - This puts the largest element at the very end
  - The last element is now in the correct and final place
- Compare each element (except the last *two*) with its neighbor to the right
  - If they are out of order, swap them
  - This puts the second largest element next to last
  - The last two elements are now in their correct and final places
- Compare each element (except the last *three*) with its neighbor to the right
  - Continue as above until you have no unsorted elements on the left

Example of bubble sort



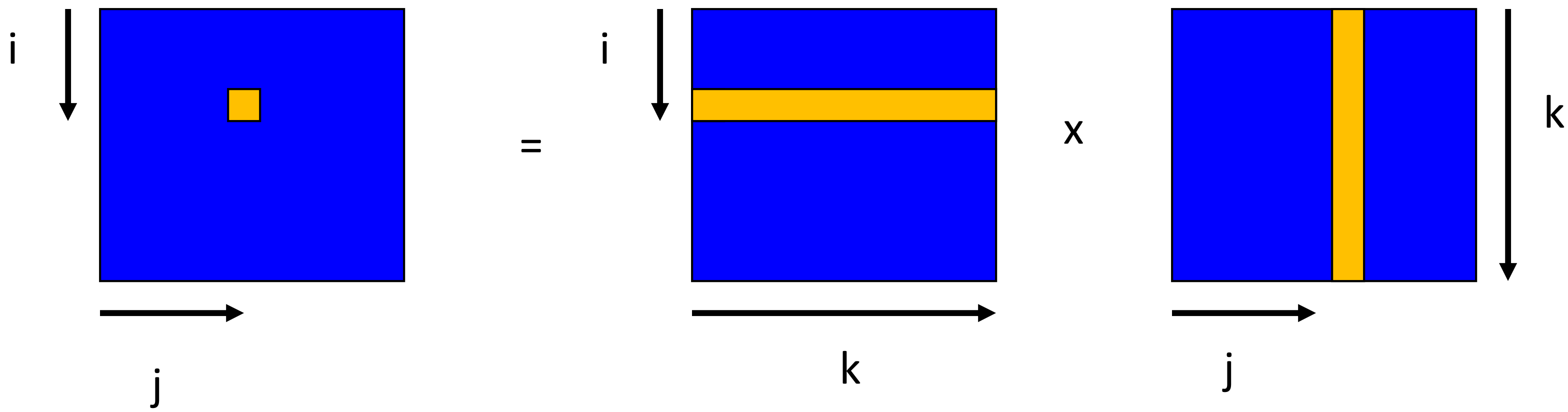
Code for bubble sort

```
public static void bubbleSort(int[] a) {
    int outer, inner;
    for (outer = a.length - 1; outer > 0; outer--) {  // counting down
        for (inner = 0; inner < outer; inner++) {       // bubbling up
            if (a[inner] > a[inner + 1]) {  // if out of order...
                int temp = a[inner];          // ...then swap
                a[inner] = a[inner + 1];
                a[inner + 1] = temp;
            }
        }
    }
}
```

Matrix Multiplication (i,j,k)

```
for I = 1 to n do
      for j = 1 to n do
            for k = 1 to n do
                  C[i,j] = C[i,j] + A[i,k] x B[k,j]
            endfor
      endfor
endfor
```

(i,j,k) Memory Map

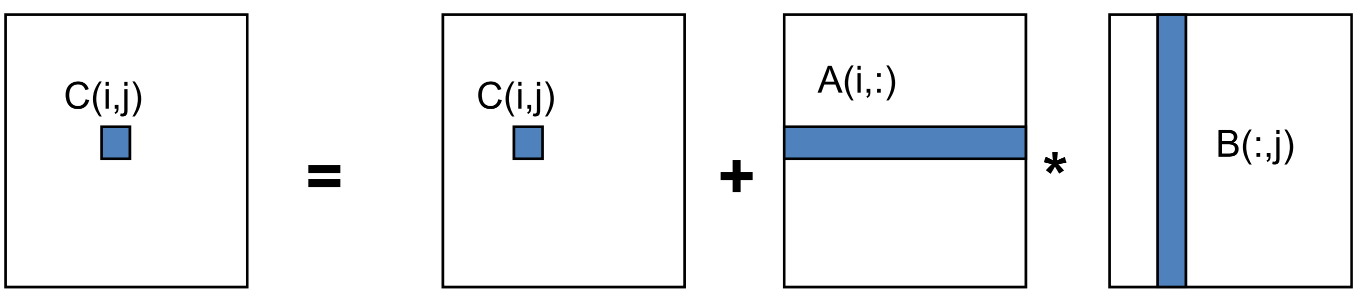


"Naïve" Matrix Multiply

{implements C = C + A*B}

```
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            C(i,j) = C(i,j) + A(i,k) * B(k,j)
```

Algorithm has $2*n^3 = O(n^3)$ Flops and operates on $3*n^2$ words of memory



Binary Number System

Negative Numbers?

- Digital electronics requires frequent addition and subtraction of numbers. You know how to design an adder, but what about a subtract-er?
- A subtract-er is not needed with the 2’s complement process. The 2’s complement process allows you to easily convert a positive number into its negative equivalent.
- Since subtracting one number from another is the same as making one number negative and adding, the need for a subtract-er circuit has been eliminated.

How To Create A Negative Number

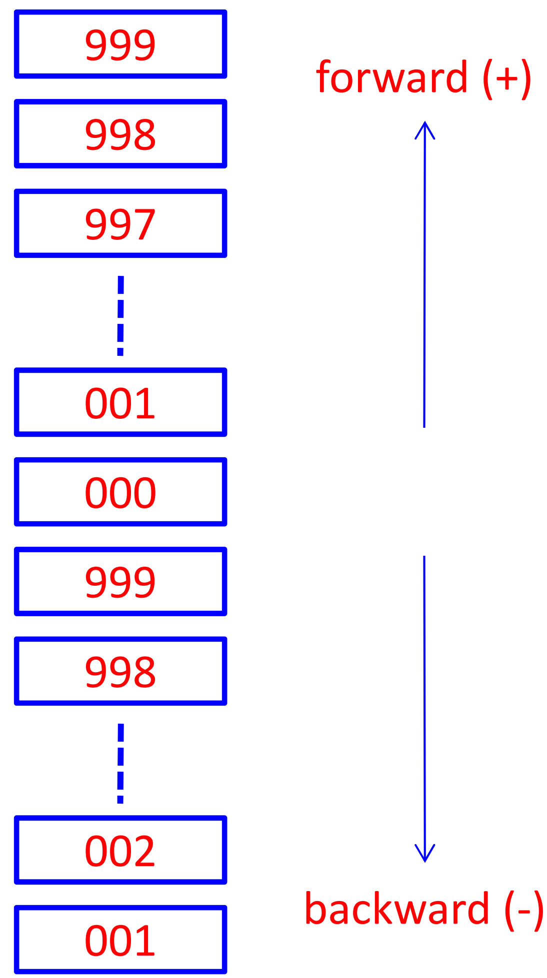
- In digital electronics you cannot simply put a minus sign in front of a number to make it negative.
- You must represent a negative number in a *fixed-length* binary number system. All signed arithmetic must be performed in a *fixed-length* number system.
- A physical *fixed-length* device (usually memory) contains a fixed number of bits (usually 4-bits, 8-bits, 16-bits) to hold the number.

3-Digit Decimal Number System

A bicycle odometer with only three digits is an example of a fixed-length decimal number system.

The problem is that without a negative sign, you cannot tell a +998 from a -2 (also a 998). Did you ride forward for 998 miles or backward for 2 miles?

Note: Car odometers do not work this way.



Negative Decimal

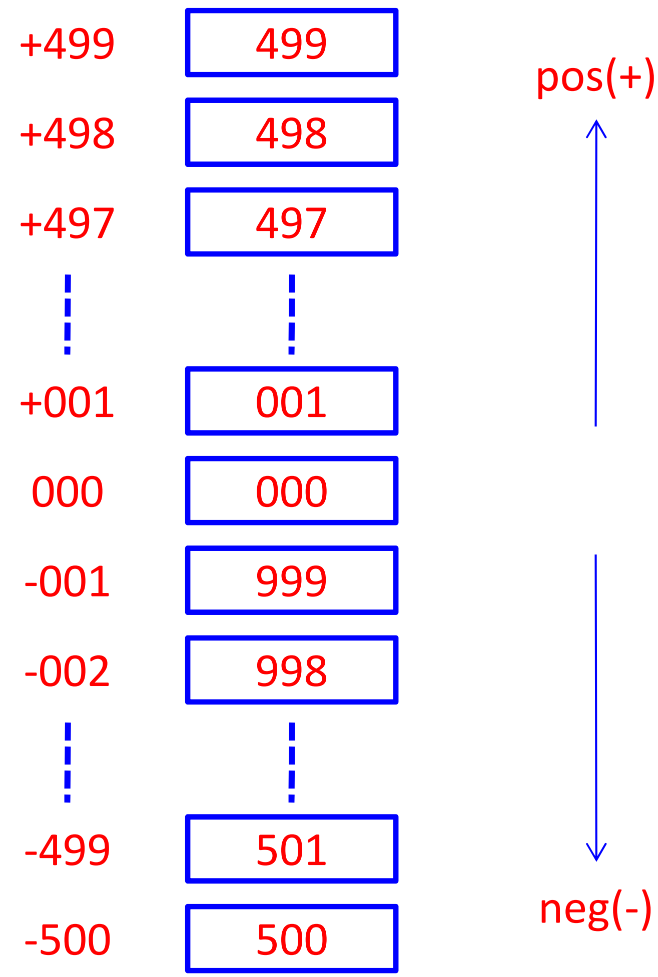
How do we represent negative numbers in this 3-digit decimal number system without using a sign?

→Cut the number system in half.

→Use 001 – 499 to indicate positive numbers.

→Use 500 – 999 to indicate negative numbers.

→Notice that 000 is not positive or negative.



“Odometer” Math Examples

$$\begin{array}{r} 3 \\ +\;\;2 \\ \hline 5 \end{array} \qquad \begin{array}{r} 003 \\ +\;\;002 \\ \hline 005 \end{array}$$

$$\begin{array}{r} 6 \\ +\;(-3) \\ \hline 3 \end{array} \qquad \begin{array}{r} 006 \\ +\;\;997 \\ \hline 1]003 \end{array}$$

Disregard Overflow

$$\begin{array}{r} (-5) \\ +\;\;2 \\ \hline (-3) \end{array} \qquad \begin{array}{r} 995 \\ +\;\;002 \\ \hline 997 \end{array}$$

$$\begin{array}{r} (-2) \\ +\;(-3) \\ \hline (-5) \end{array} \qquad \begin{array}{r} 998 \\ +\;\;997 \\ \hline 1]995 \end{array}$$

Disregard Overflow

It Works!

Complex Problems

- The previous examples demonstrate that this process works, but how do we *easily* convert a number into its negative equivalent?
- In the examples, converting the negative numbers into the 3-digit decimal number system was fairly easy. To convert the (-3), you simply counted backward from 1000 (i.e., 999, 998, 997).
- This process is not as easy for large numbers (e.g., -214 is 786). How did we determine this?
- To convert a large negative number, you can use the 10's Complement Process.

10’s Complement Process

The **10’s Complement** process uses base-10 (decimal) numbers. Later, when we’re working with base-2 (binary) numbers, you will see that the **2’s Complement** process works in the same way.

First, complement all of the digits in a number.

– A digit’s complement is the number you add to the digit to make it equal to the largest digit in the base (i.e., 9 for decimal). The complement of 0 is 9, 1 is 8, 2 is 7, etc.

Second, add 1.

– Without this step, our number system would have two zeroes (+0 & -0), which no number system has.

10's Complement Examples

Example #1

$$\left.\begin{array}{r} -003 \\ \downarrow\downarrow\downarrow \\ 996 \end{array}\right\}\ \text{Complement Digits}$$

$$\left.\begin{array}{r} 996 \\ +1 \\ \hline 997 \end{array}\right\}\ \text{Add 1}$$

Example #2

$$\left.\begin{array}{r} -214 \\ \downarrow\downarrow\downarrow \\ 785 \end{array}\right\}\ \text{Complement Digits}$$

$$\left.\begin{array}{r} 785 \\ +1 \\ \hline 786 \end{array}\right\}\ \text{Add 1}$$

8-Bit Binary Number System

Apply what you have learned to the binary number systems. How do you represent negative numbers in this 8-bit binary system?

→Cut the number system in half.

→Use 00000001 – 01111111 to indicate positive numbers.

→Use 10000000 – 11111111 to indicate negative numbers.

→Notice that 00000000 is not positive or negative.

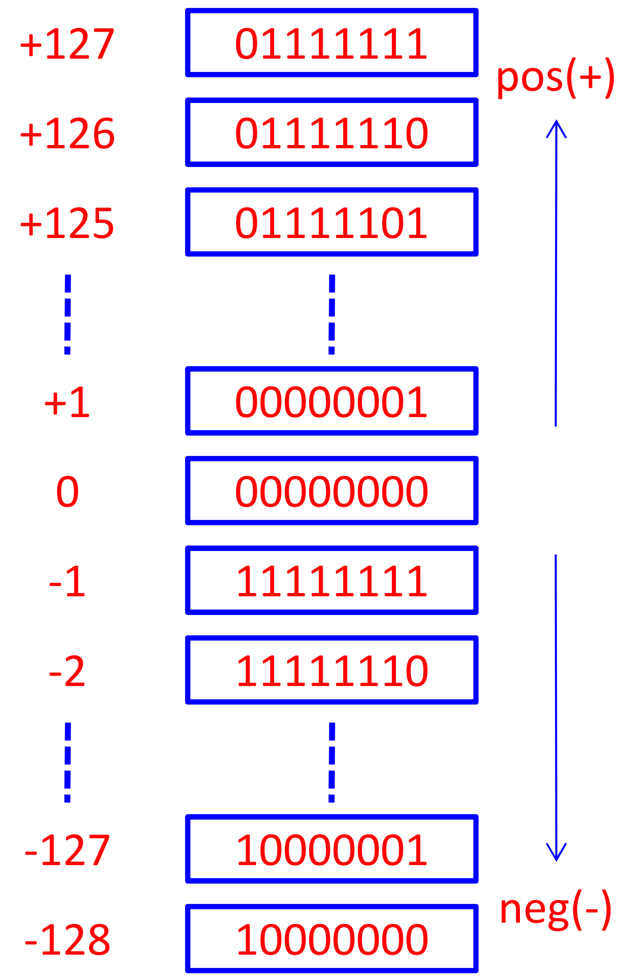
pos(+)

+127	01111111
+126	01111110
+125	01111101
⋮	⋮
+1	00000001
0	00000000
-1	11111111
-2	11111110
⋮	⋮
-127	10000001
-128	10000000

neg(-)

Sign Bit

- What did do you notice about the most significant bit of the binary numbers?
- The MSB is (0) for all positive numbers.
- The MSB is (1) for all negative numbers.
- The MSB is called the sign bit.
- In a signed number system, this allows you to instantly determine whether a number is positive or negative.



2'S Complement Process

The steps in the **2's Complement** process are similar to the 10's Complement process. However, you will now use the base two.

First, complement all of the digits in a number.

– A digit's complement is the number you add to the digit to make it equal to the largest digit in the base (i.e., 1 for binary). In binary language, the complement of 0 is 1, and the complement of 1 is 0.

Second, add 1.

– Without this step, our number system would have two zeroes (+0 & -0), which no number system has.

2’s Complement Examples

Example #1

```
 5 = 00000101
     ↓↓↓↓↓↓↓↓      Complement Digits
     11111010
           +1      Add 1
     --------
-5 = 11111011
```

Example #2

```
-13 = 11110011
      ↓↓↓↓↓↓↓↓     Complement Digits
      00001100
            +1     Add 1
      --------
 13 = 00001101
```

Using The 2's Compliment Process

Use the 2's complement process to add together the following numbers.

$$\begin{array}{r} \text{POS} \\ +\ \ \text{POS} \\ \hline \text{POS} \end{array} \Rightarrow \begin{array}{r} 9 \\ +\ \ 5 \\ \hline 14 \end{array} \qquad \begin{array}{r} \text{NEG} \\ +\ \ \text{POS} \\ \hline \text{NEG} \end{array} \Rightarrow \begin{array}{r} (-9) \\ +\ \ 5 \\ \hline -\ 4 \end{array}$$

$$\begin{array}{r} \text{POS} \\ +\ \ \text{NEG} \\ \hline \text{POS} \end{array} \Rightarrow \begin{array}{r} 9 \\ +\ (-5) \\ \hline 4 \end{array} \qquad \begin{array}{r} \text{NEG} \\ +\ \ \text{NEG} \\ \hline \text{NEG} \end{array} \Rightarrow \begin{array}{r} (-9) \\ +\ (-5) \\ \hline -\ 14 \end{array}$$

POS + POS → POS Answer

If no 2's complement is needed, use regular binary addition.

$$
\begin{array}{rcr}
9 & \longrightarrow & 00001001 \\
+ \quad 5 & \longrightarrow & +\ 00000101 \\
\hline
14 & \longleftarrow & 00001110
\end{array}
$$

POS + NEG → POS Answer

Take the 2’s complement of the negative number and use regular binary addition.

$$
\begin{array}{r}
9 \\
+\ (-5) \\
\hline
4
\end{array}
\qquad
\begin{array}{r}
00001001 \\
+\ 11111011 \\
\hline
\cancel{1}]\,00000100
\end{array}
$$

8th Bit = 0: Answer is Positive
Disregard 9th Bit

$$
\left.
\begin{array}{r}
00000101 \\
\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow \\
11111010 \\
+1 \\
\hline
11111011
\end{array}
\right\}
\text{2's Complement Process}
$$

POS + NEG → NEG Answer

Take the 2's complement of the negative number and use regular binary addition.

```
  (-9)        11110111
+   5       + 00000101
------      ----------
  -4          11111100
```

8th Bit = 1: Answer is Negative

To Check: Perform 2's Complement On Answer

```
11111100
↓↓↓↓↓↓↓↓
00000011
      +1
--------
00000100
```

2's Complement Process

```
00001001
↓↓↓↓↓↓↓↓
11110110
      +1
--------
11110111
```

NEG + NEG → NEG Answer

Take the 2's complement of both negative numbers and use regular binary addition.

$$\begin{array}{rcr} (-9) & \longrightarrow & 11110111 \\ +\ (-5) & \longrightarrow & +\ 11111011 \\ \hline -\boxed{14} & \longleftarrow & \cancel{1}]\ 11110010 \end{array}$$

2's Complement Numbers, See Conversion Process In Previous Slides

8th Bit = 1: Answer is Negative
Disregard 9th Bit

To Check: Perform 2's Complement On Answer

$$\begin{array}{r} 11110010 \\ \downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow\downarrow \\ 00001101 \\ +1 \\ \hline 00001110 \end{array}$$

To be continued to next class…