Introduction To Computing

Second Week

CLASS ASSIGNMENT

Verbal Presentation:

Topic: Ethical/Social/ Legal Repercussions

Course Description

- This course assumes that students coming from different academic backgrounds do not possess prior knowledge and understanding of computer science. It will provide basic knowledge about computers and processing information through computers.
- It includes topics related to computer hardware and software. Major computer science subjects will be delivered to the students which they will study throughout their degree program. These major subjects include computer architecture, software engineering, data structures, operating system, computer networks and databases.

Course Learning Outcomes (CLOs)

- CLO1: Solve the problem algorithmically for implementation on computers.
- CLO2: Describe the Number Systems.
- CLO3: Explain the concepts of computer architecture, software engineering, data structures, operating systems, computer networks and databases, at an introductory level.

Program Learning Outcomes (PLOs)

What you need to understand?

- CLOs are mapped to topics, chapters and questions of your course.

- CLOs are also mapped to PLOs of your Degree program.

Lecture Plan

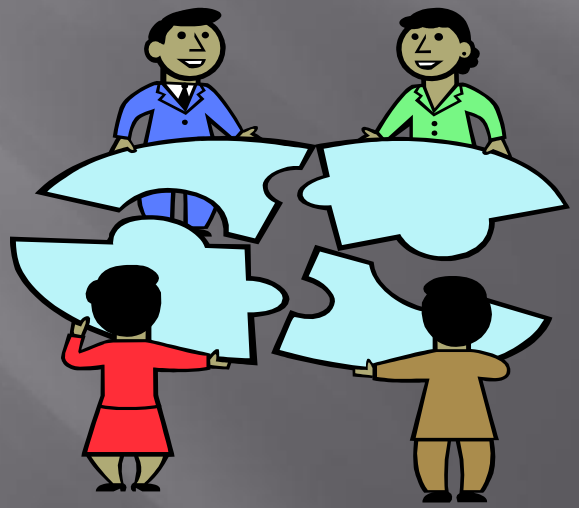
Lecture Plan

Analytical skill/reasoning

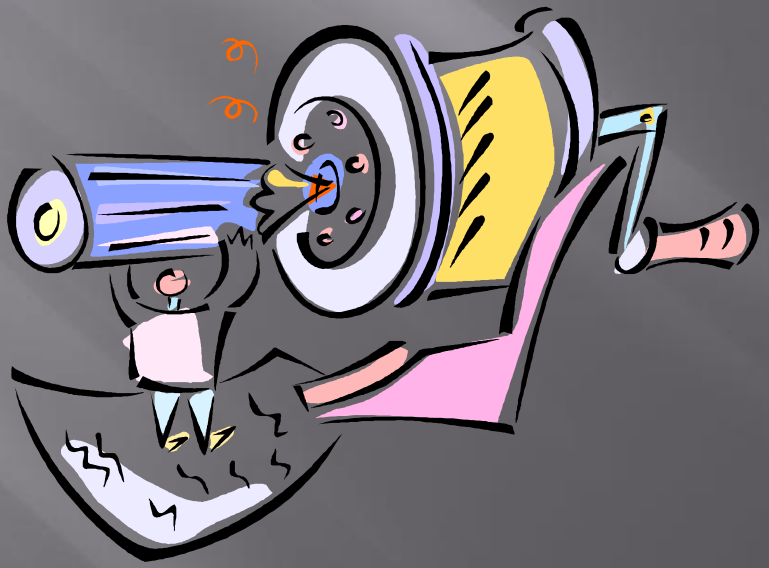
Analytical skill/ reasoning is the ability to visualize, articulate, and solve both complex and uncomplicated problems and concepts and make decisions that are based on available information.

Analytical thought:

Analytical thought is breaking down the pieces of something before reaching a conclusion, and not always the conclusion you expected to reach.



How to Develop Analytical skill?



Daily Routine Example: Buying TV

To be truly analytical you need to think about all things involved in a situation, take buying something for instance. You want to buy a TV.

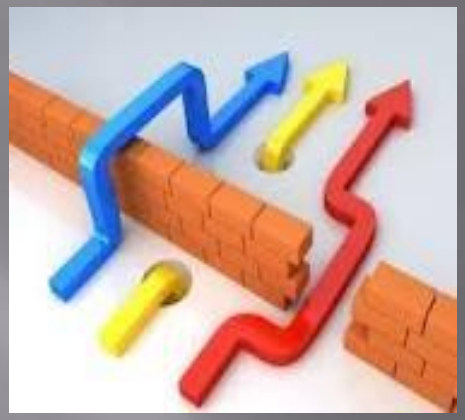
- You analyze the TV itself, is it a good brand, is it a current technology, how does it compare to other current technologies (LED vs LCD for instance) how is the resolution.
- Then you analyze if you have need for it, do you have a place to put it, is it really important? and
- You analyze the financial impact, do you have cash for it, are you using credit, how long will it take to pay it off, things of that nature.
- So you look at all angles of the same situation. To be analytical you have to weigh out your options, think about them before you act, and not act on first impulse about much of anything.

Simple Practice

It depends on what situations you want to apply your analytical skill.

Trial and errors. One of the easiest ways to practice is following the news. Read any articles of your interest, do some research on that particular event and guess what it going to happen because of that particular event. Compare yours with the "correct" answer the next day. Should you gave the wrong guess, read article that explains why it is so and ask yourself how to improve. Why couldn't you apply your analytical model in this case? Did you make certain false assumptions? Did you miss considering some aspects of the problem? Through the process, you will develop and perfect your own approach to analyze those kind of problems.

Imitating the best. Try to analyze a problem with your friends. Find out how their analytical process work and learn from them.



Analytical skill is a Problem solving skill

End of Lecture

Algorithms

Introduction

- The methods of algorithm design form one of the core practical technologies of computer science.
- The main aim of this lecture is to familiarize the student with the framework we shall use through the course about the design and analysis of algorithms.
- We start with a discussion of the algorithms needed to solve computational problems. The problem of *sorting* is used as a running example.
- We introduce a *pseudocode* to show how we shall specify the algorithms.

Algorithms

- The word *algorithm* comes from the name of a Persian mathematician Abu Ja'far Mohammed ibn-i Musa al Khowarizmi.

- In computer science, this word refers to a special method useable by a computer for solution of a problem. The statement of the problem specifies in general terms the desired input/output relationship.

- For example, sorting a given sequence of numbers into nondecreasing order provides fertile ground for introducing many standard design techniques and analysis tools.

The problem of sorting

Input: sequence $\langle a_1, a_2, \ldots, a_n \rangle$ of numbers.

Output: permutation $\langle a'_1, a'_2, \ldots, a'_n \rangle$ such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

Example:

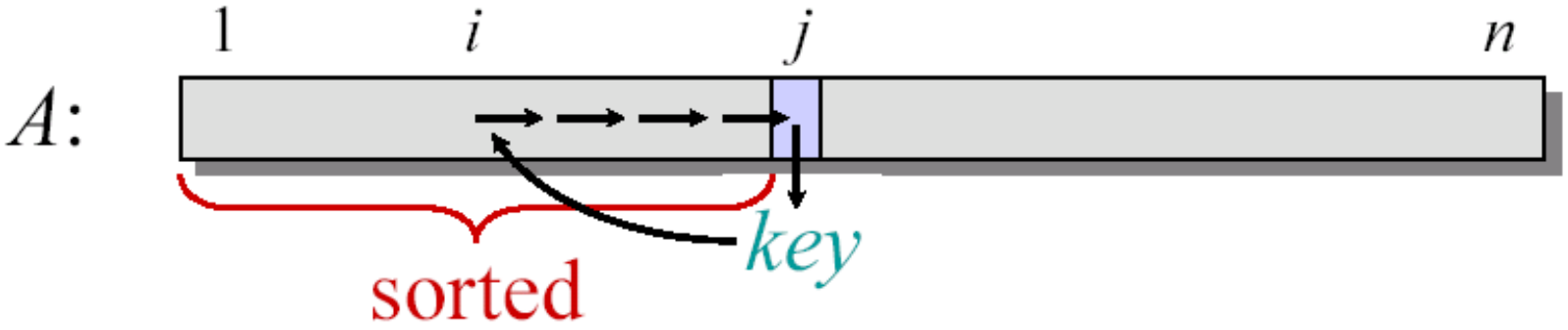
Input: 8 2 4 9 3 6

Output: 2 3 4 6 8 9

Insertion Sort

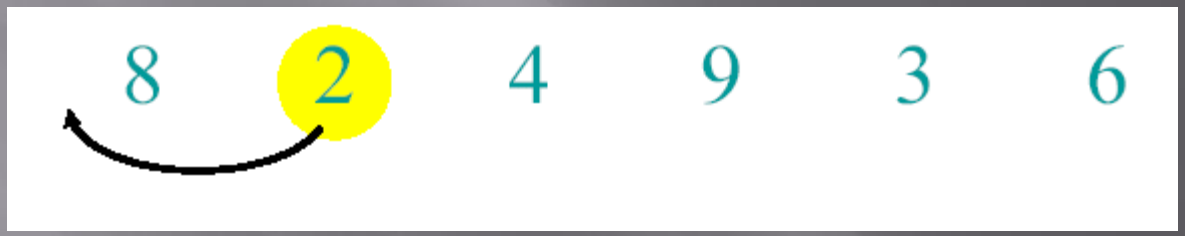
"pseudocode"

```
INSERTION-SORT (A, n)      ▷ A[1 . . n]
    for j ← 2 to n
        do key ← A[ j]
           i ← j – 1
           while i > 0 and A[i] > key
               do A[i+1] ← A[i]
                  i ← i – 1
           A[i+1] = key
```

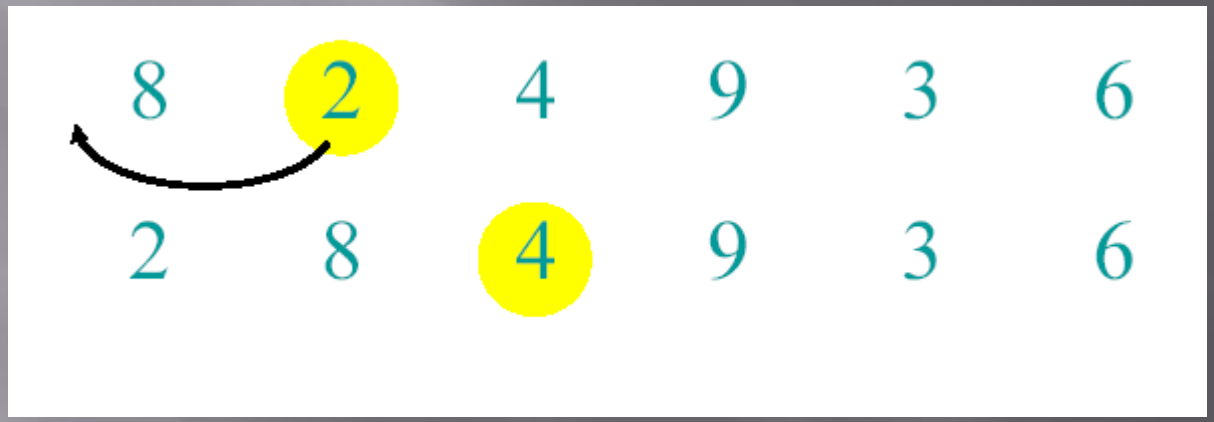


Example of Insertion Sort

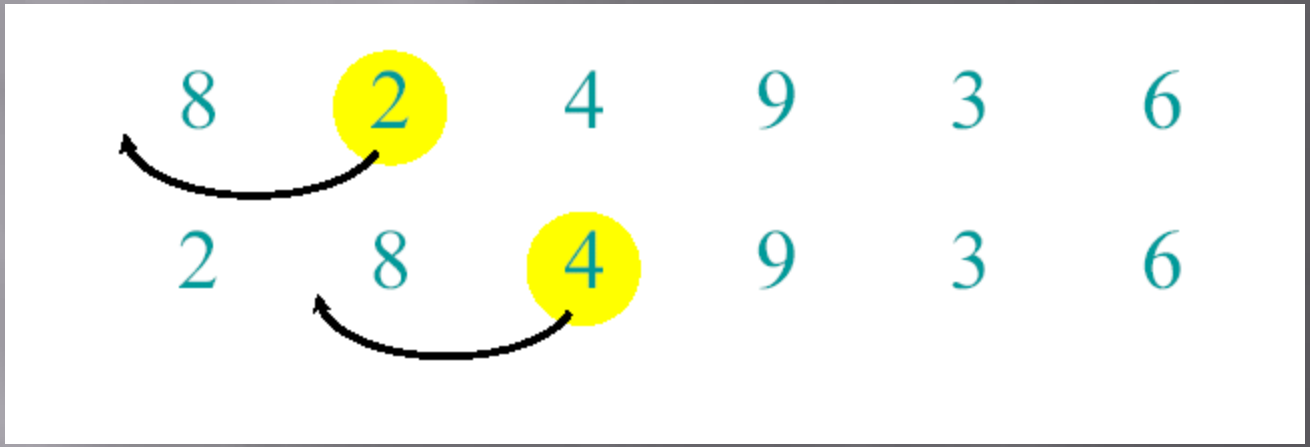
Example of Insertion Sort



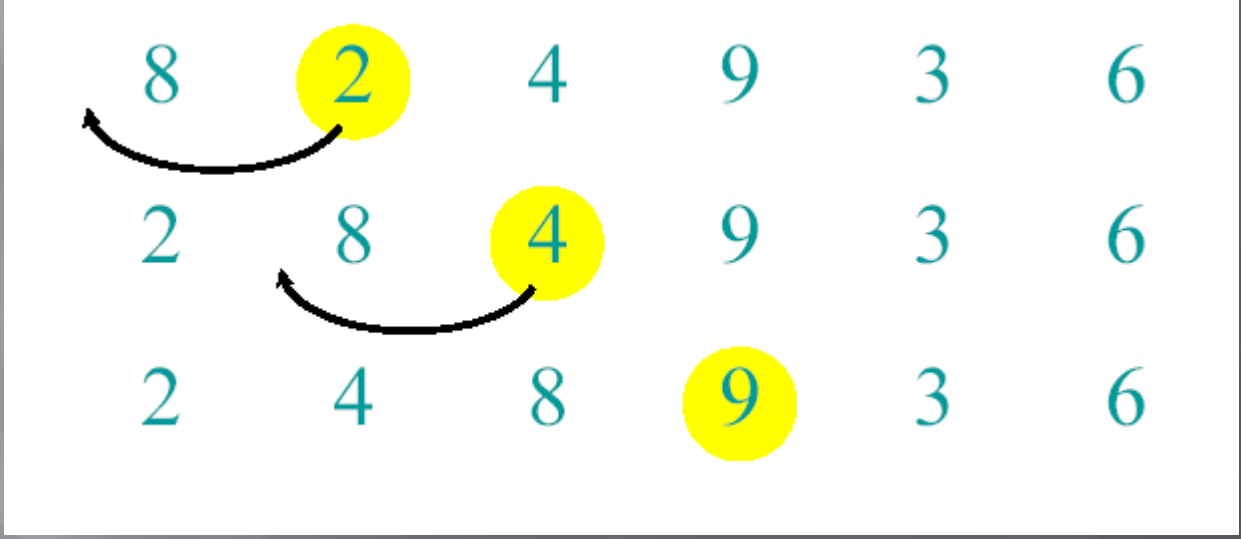
Example of Insertion Sort



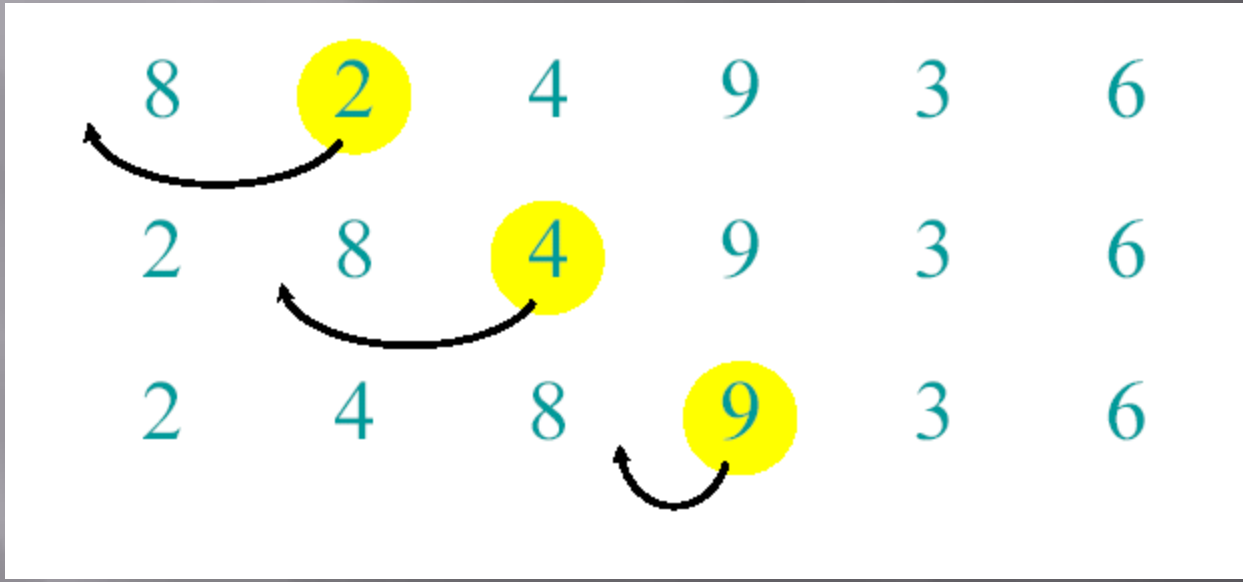
Example of Insertion Sort



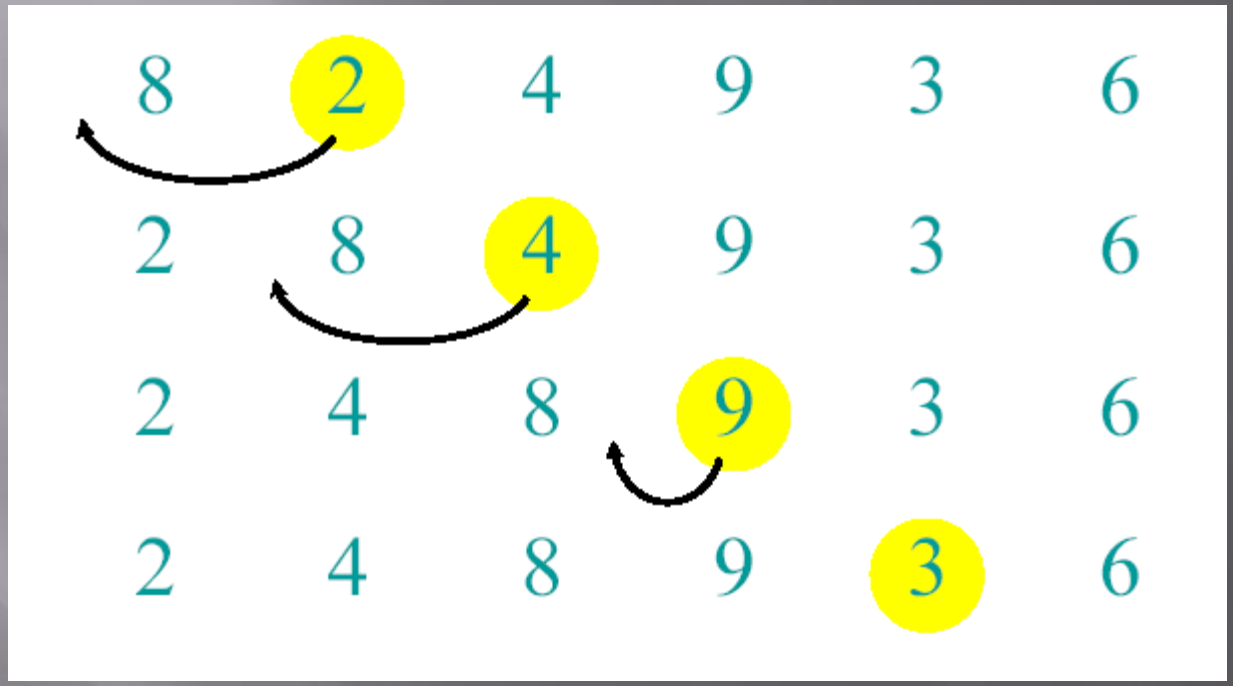
Example of Insertion Sort



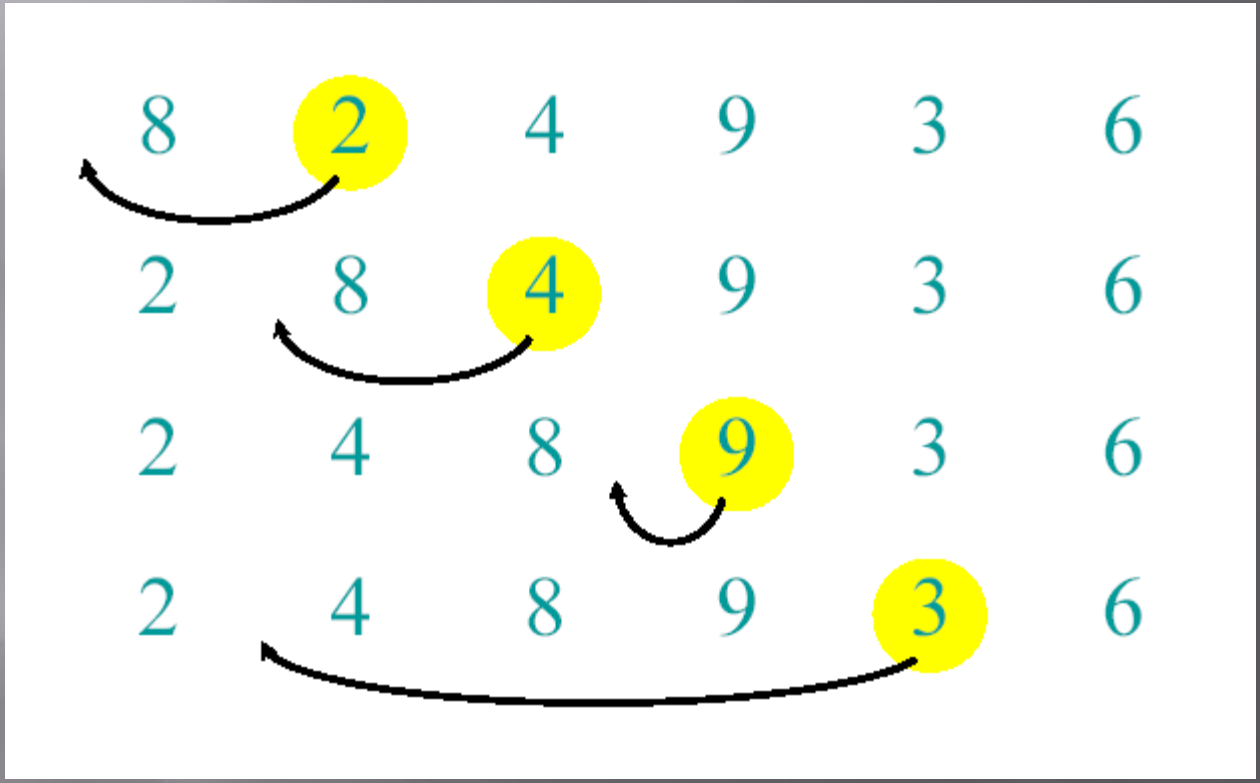
Example of Insertion Sort



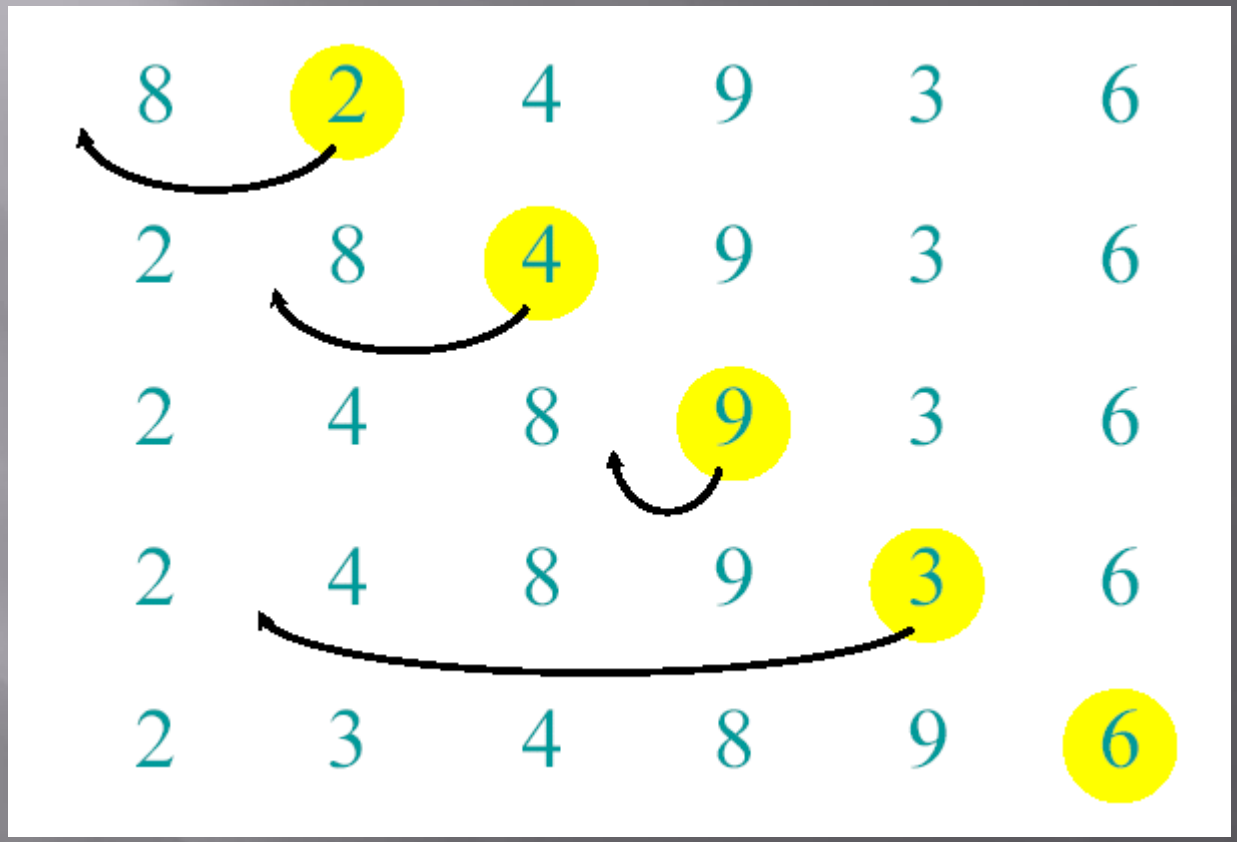
Example of Insertion Sort



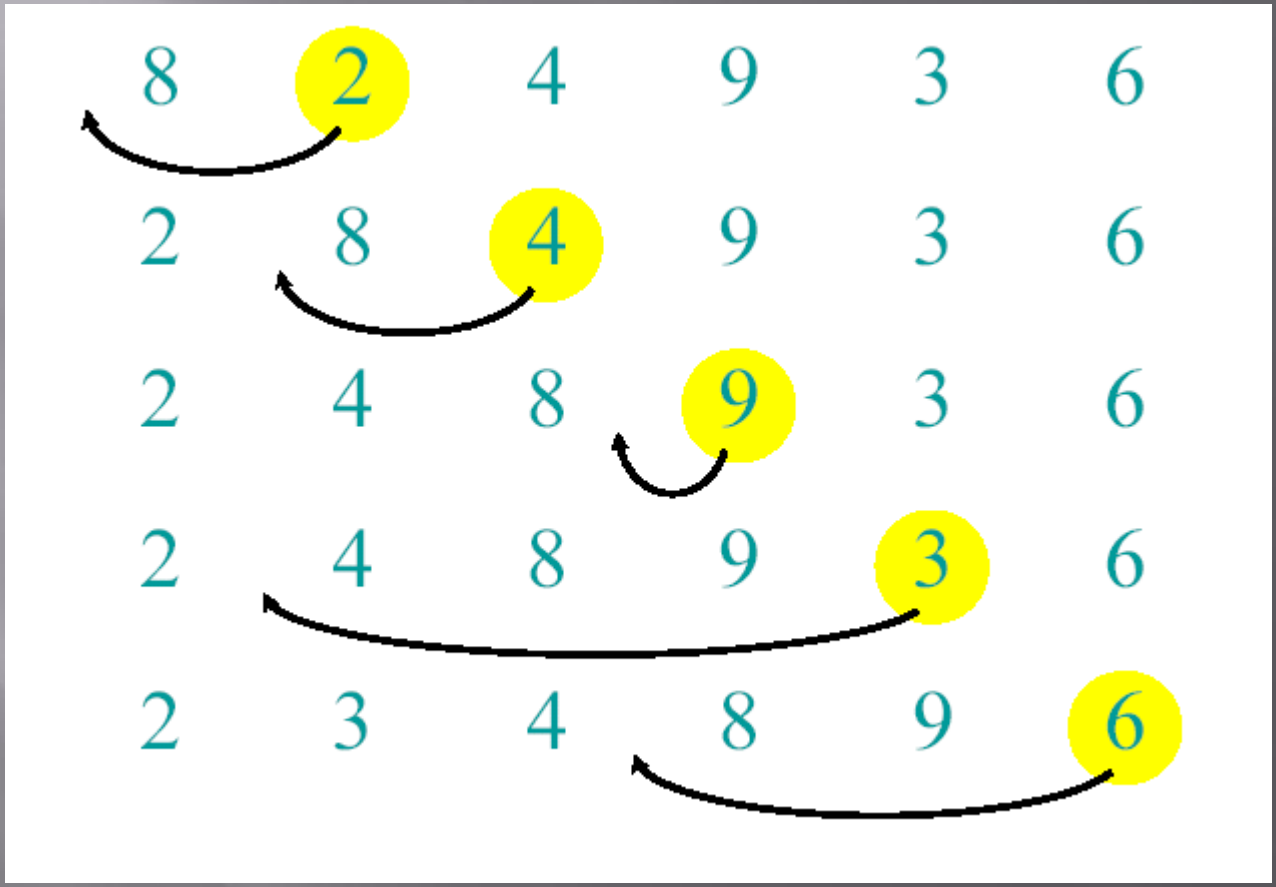
Example of Insertion Sort



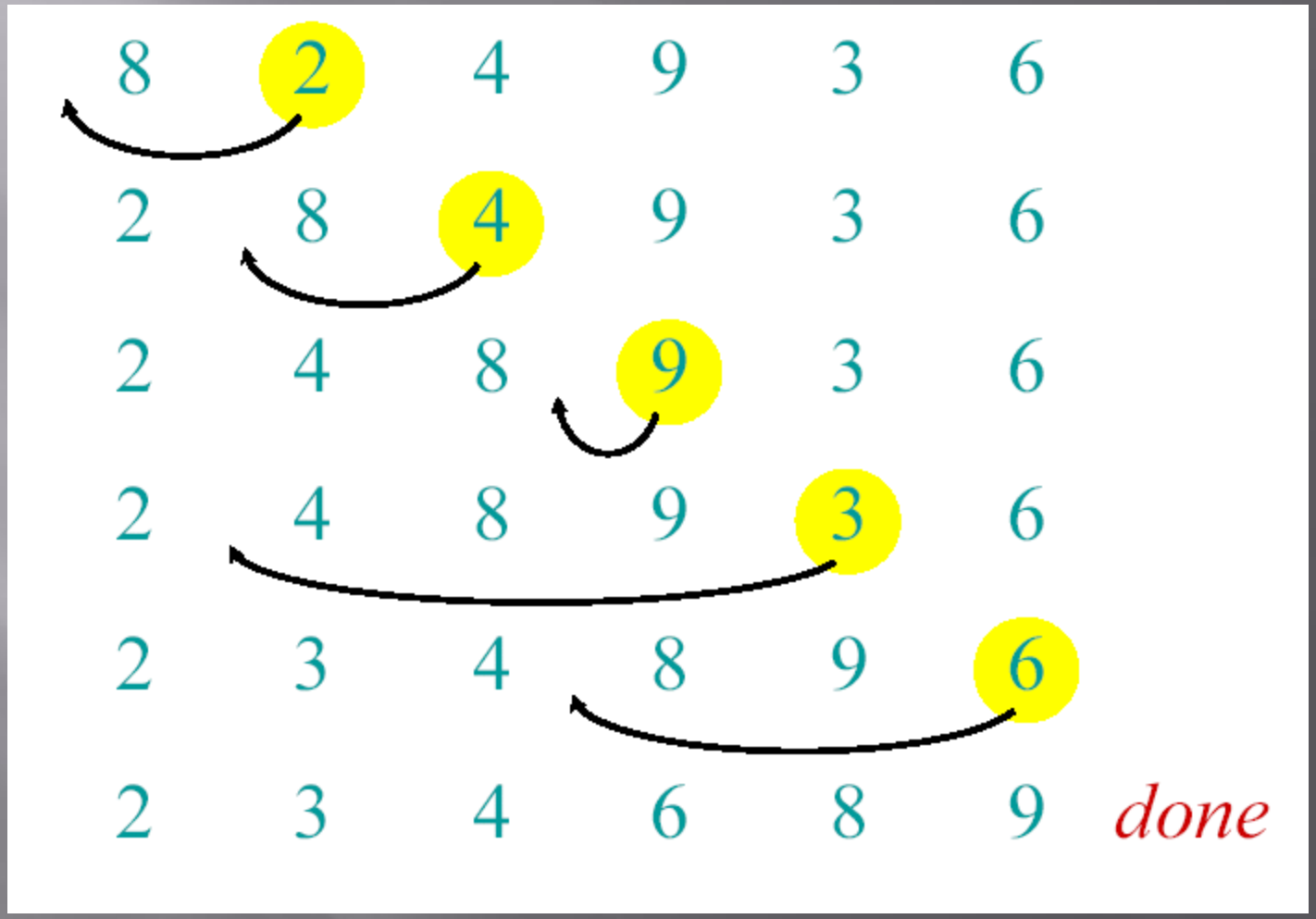
Example of Insertion Sort



Example of Insertion Sort

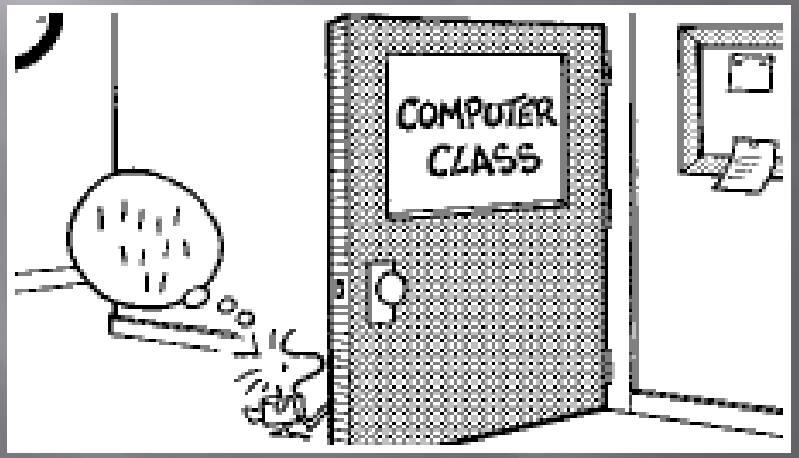


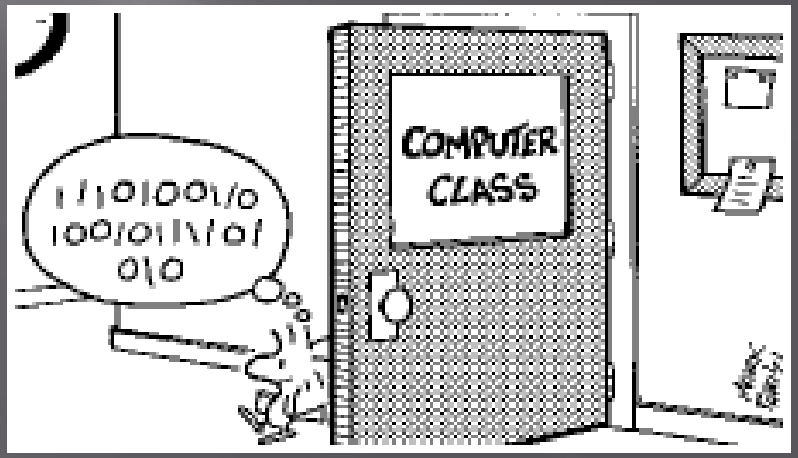
Example of Insertion Sort



CHAPTER 1

Data Storage (& Representation)



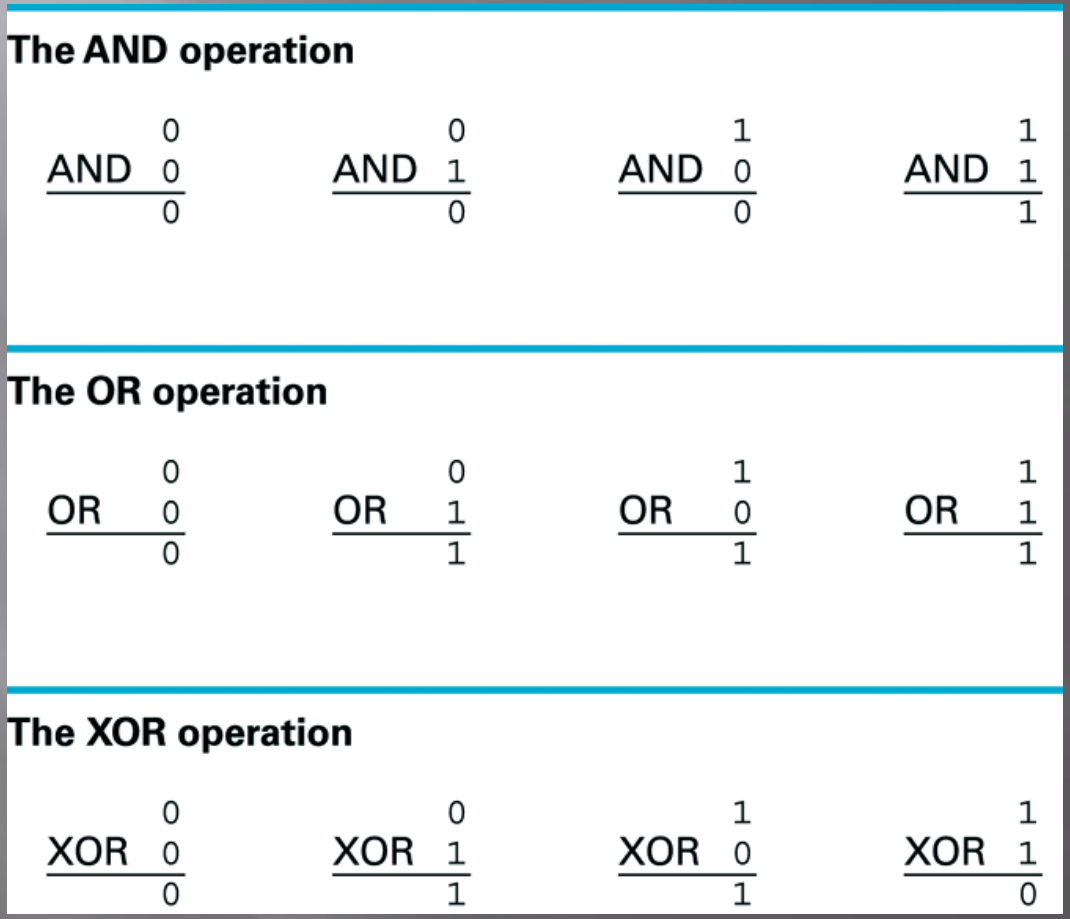


1.1 Bits and Their Storage

- Information represented as patterns of *bits* (binary digits)

- A bit is either 0 or 1 (*true* or *false*)
- Meaning of bit(-stream)s varies
  - numeric values, characters, images, sounds…
- Requires a device that can be in one of two states (& remain in that state as long as needed)
  - *Flip-flop* circuits

1.1 The Boolean Operations AND, OR, and XOR



- Note: AND and OR exist in natural language!

Example:

Boolean Operations and natural language

1. Khalid has gone to cafe.
2. **AND** Umer has gone to School.
3. **THEN** their mother can do house chores.

Result:

- Both has to go to school for their mother to do chores of the house.
- Check with OR operation.

End of Lecture

1.1 AND and OR Gates

- '0' and '1' Digits are representing "Voltage levels" ?

AND

Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

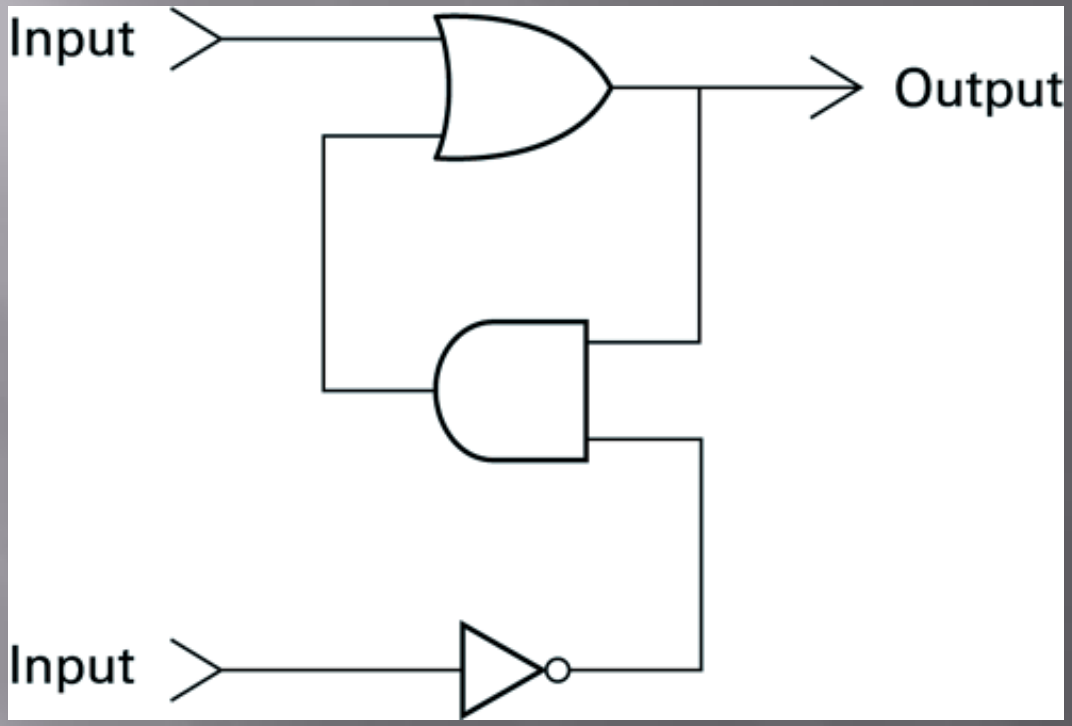
OR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

Flip Flop circuit:

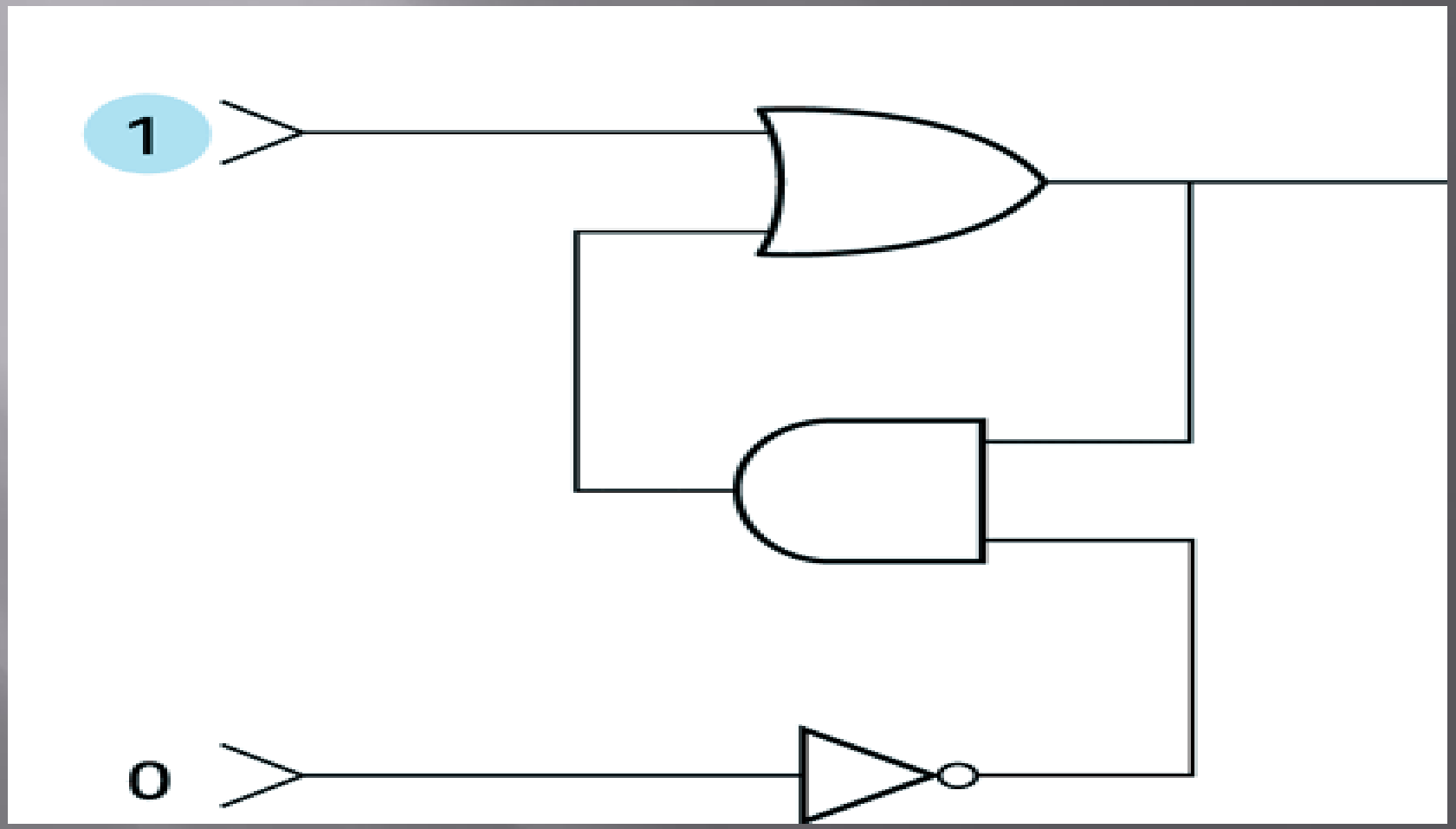
- A flip flop circuit that produces an output value of ‘0’ or ‘1’ that remains constant until a temporary pulse from another circuit causes it to shift to the other value.

1.1 A Simple Flip-flop Circuit

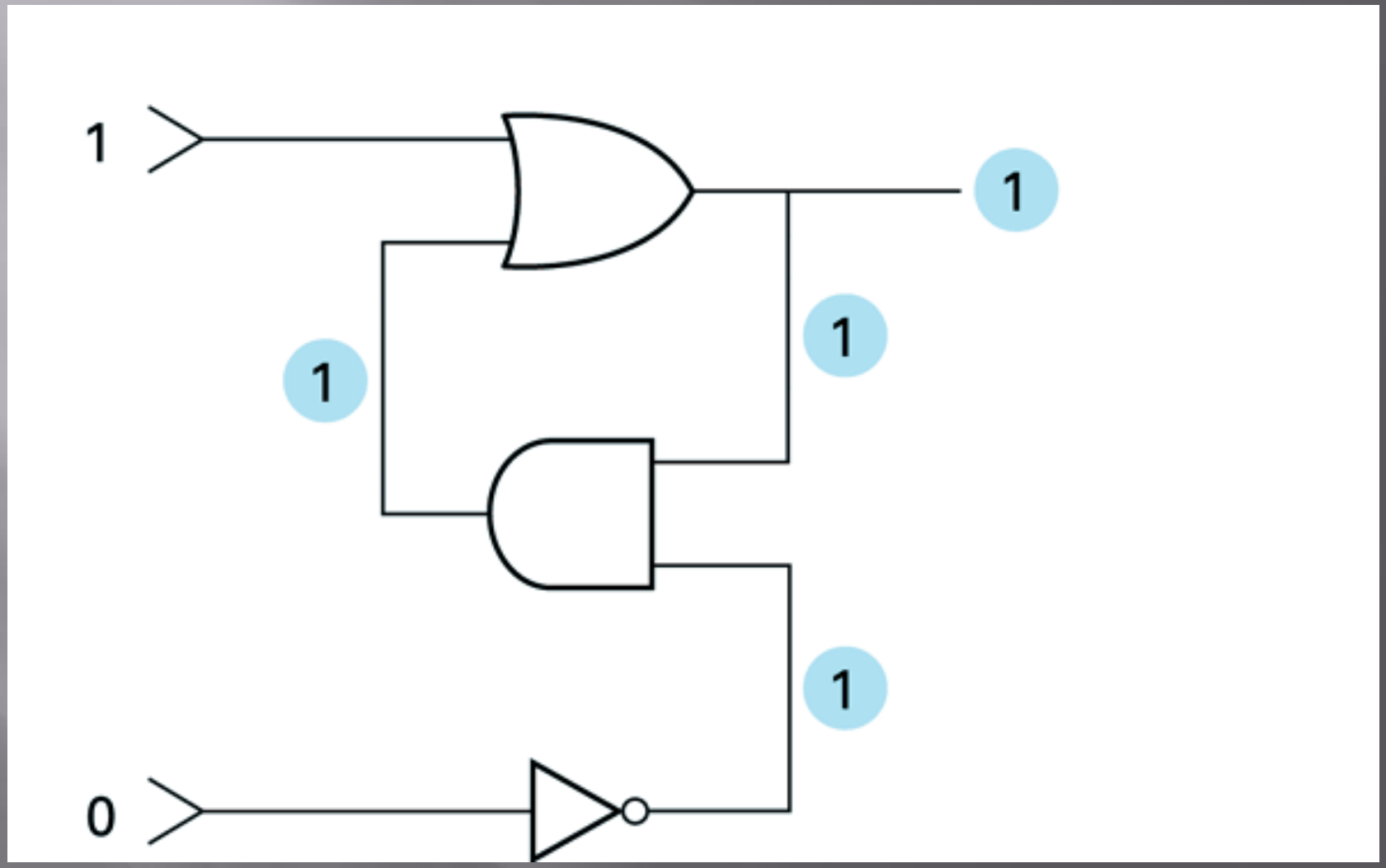


- As long as both inputs remain 0: output does not change
- Temporarily placing 1 on upper input => output = 1
- Temporarily placing 1 on lower input => output = 0
- So: output flip-flops between 2 values under external control

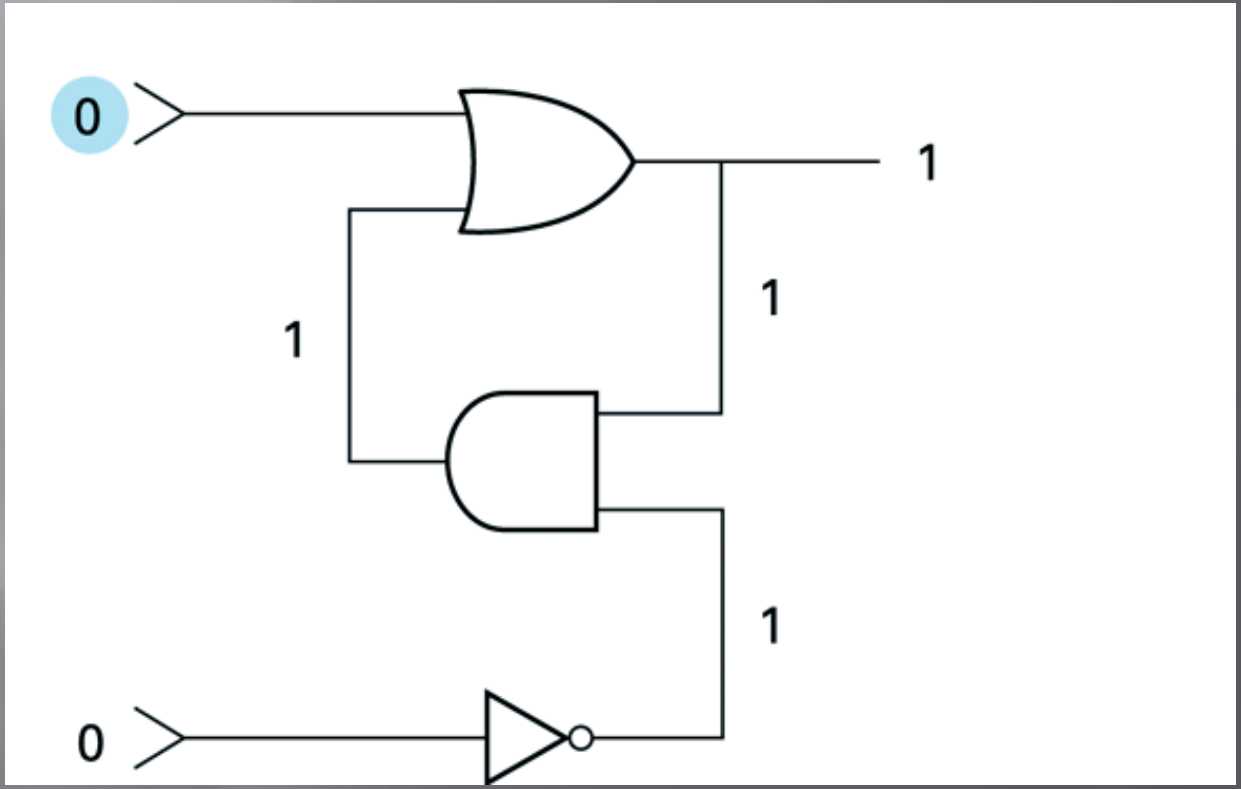
1.1 Setting the Output of a Flip-flop to 1



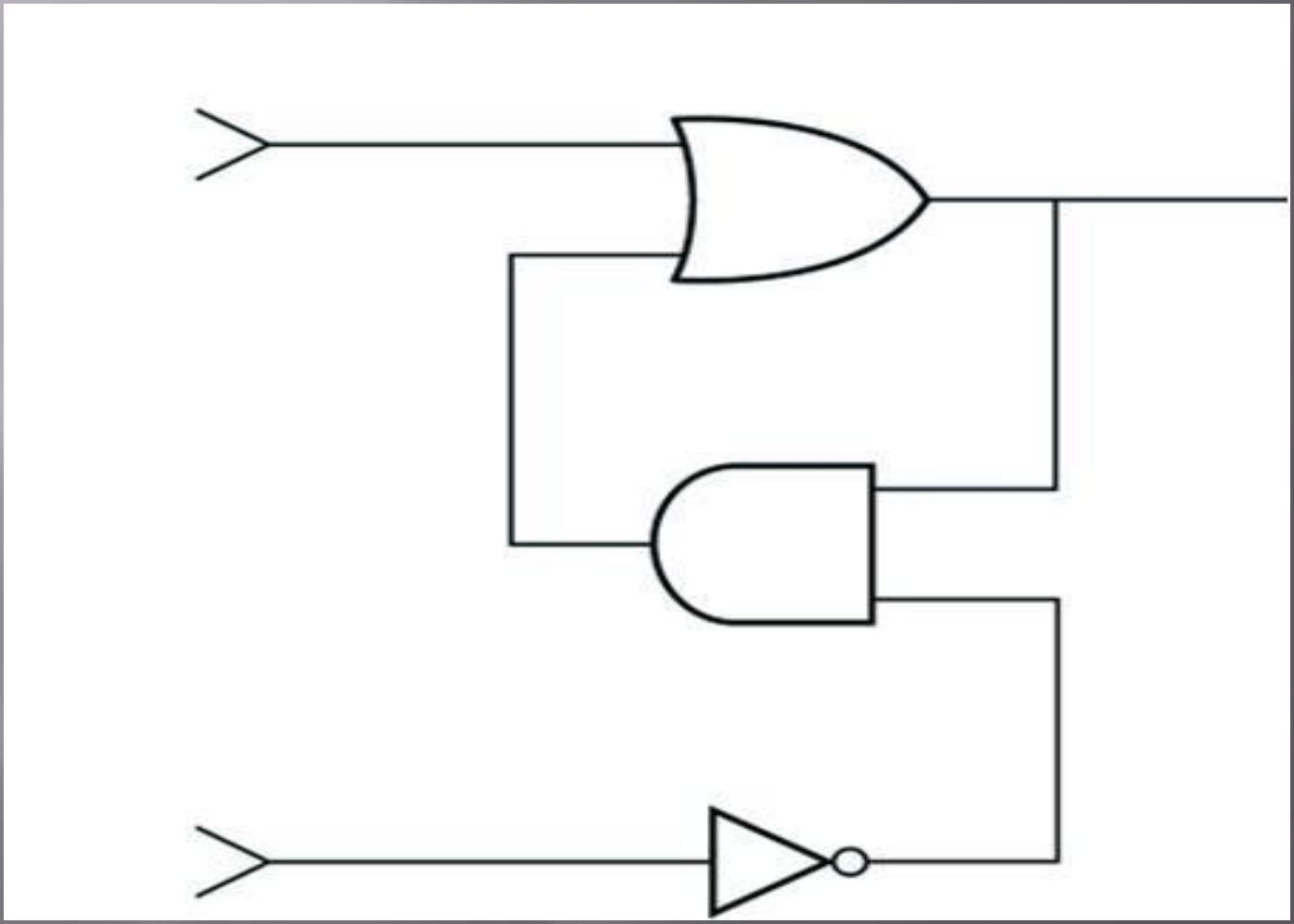
1.1 Setting the Output of a Flip-flop to 1(cont'd)



1.1 Setting the Output of a Flip-flop to 1(cont'd)



Temporarily placing ‘1’ on the lower end of the flip flop:

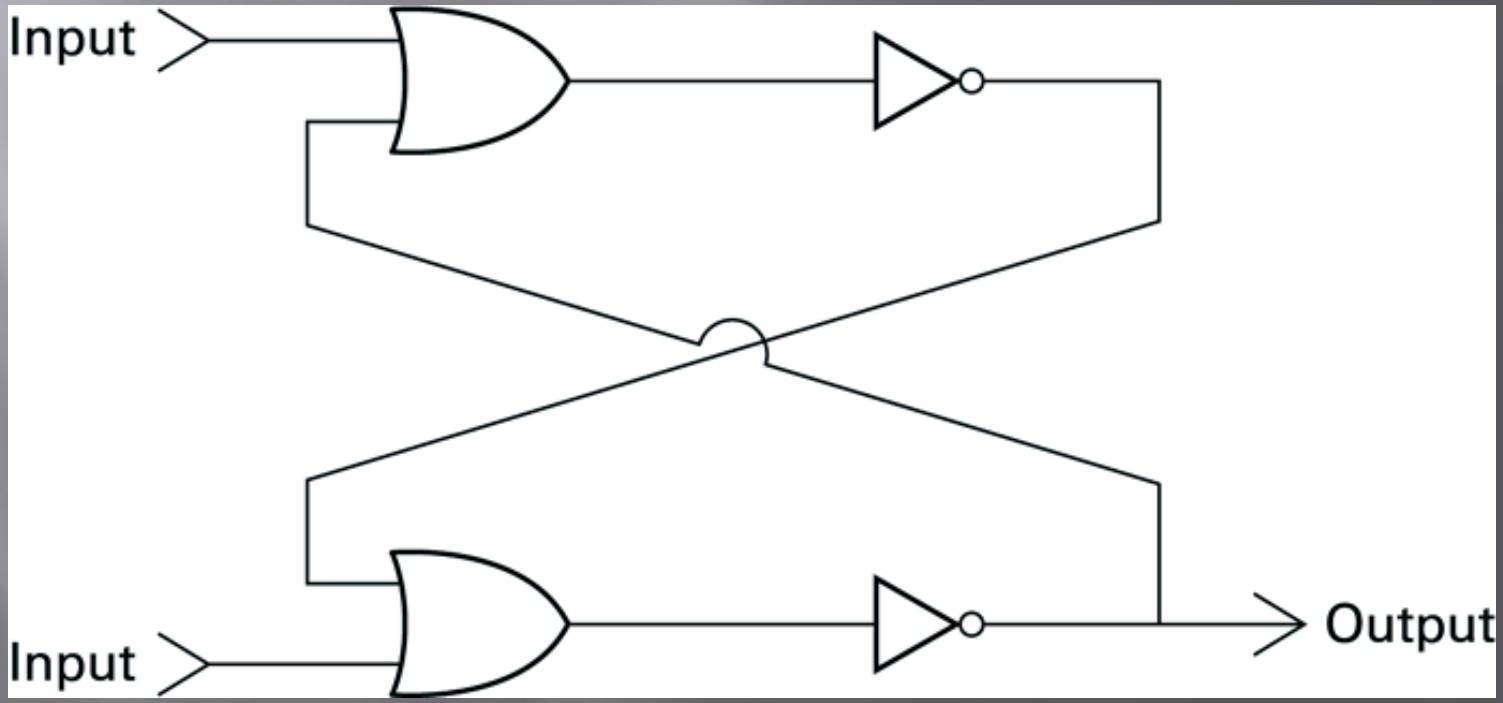


Another way of constructing Flip-Flop

Dead Line: 01-11-2016

Assignment: Write outputs & sequence of steps on the basis of following inputs?

- upper input = 1 and lower input = 0
- upper input = 0 and lower input = 0



1.1 The Hexadecimal Coding System

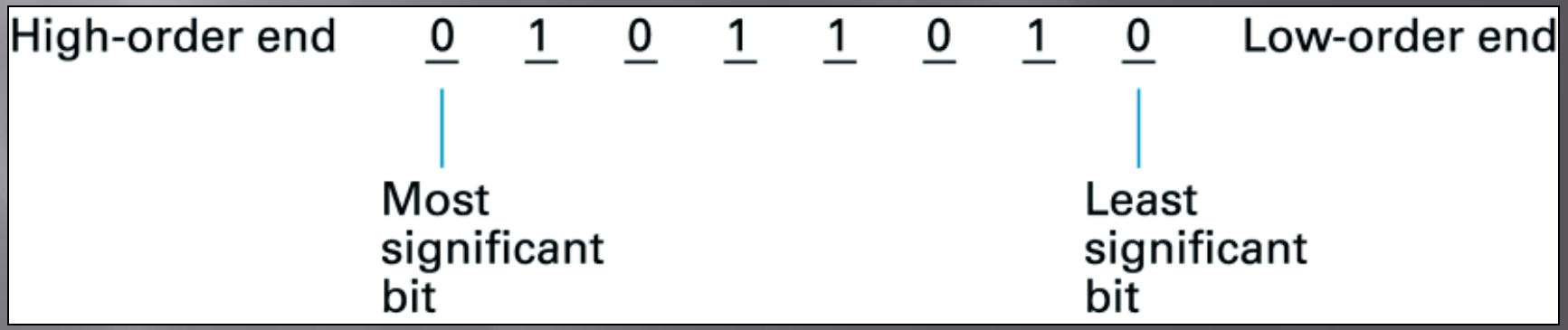
- Bit-streams often very long
- For simplicity of notation:
  - Hexadecimal system
- Reduces 4 bits to 1 symbol
- Especially important in assembly language programming

Bit pattern	Hexadecimal representation
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

1.2 Main Memory

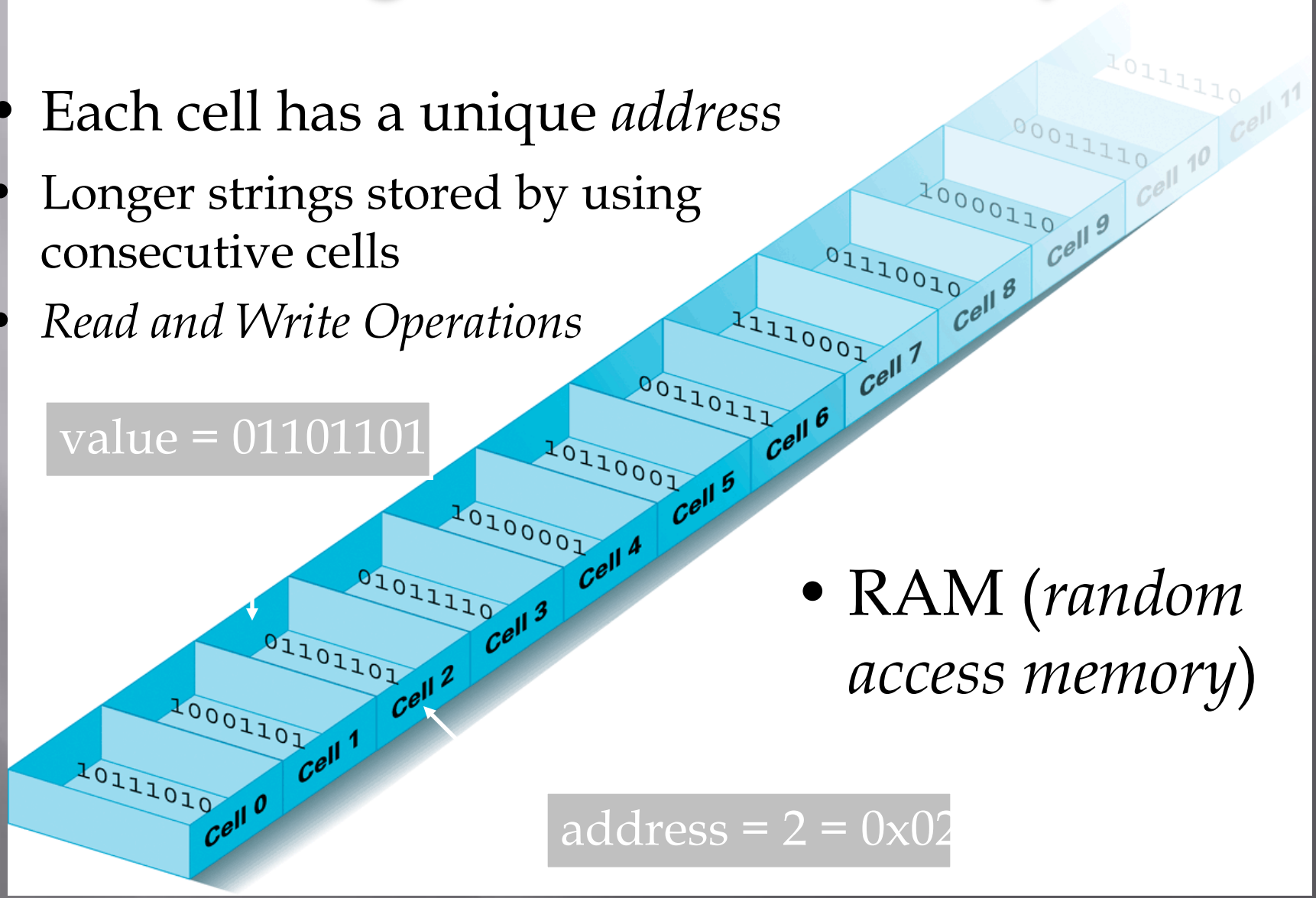
- Large collection of circuits, each capable of storing a single bit

- Arranged in small cells, typically of 8 bits each (a.k.a.: *byte*)



1.2 Arrangement of Memory Cells

- Each cell has a unique *address*
- Longer strings stored by using consecutive cells
- *Read and Write Operations*



- RAM (*random access memory*)

End of Lecture