

Basic Computer

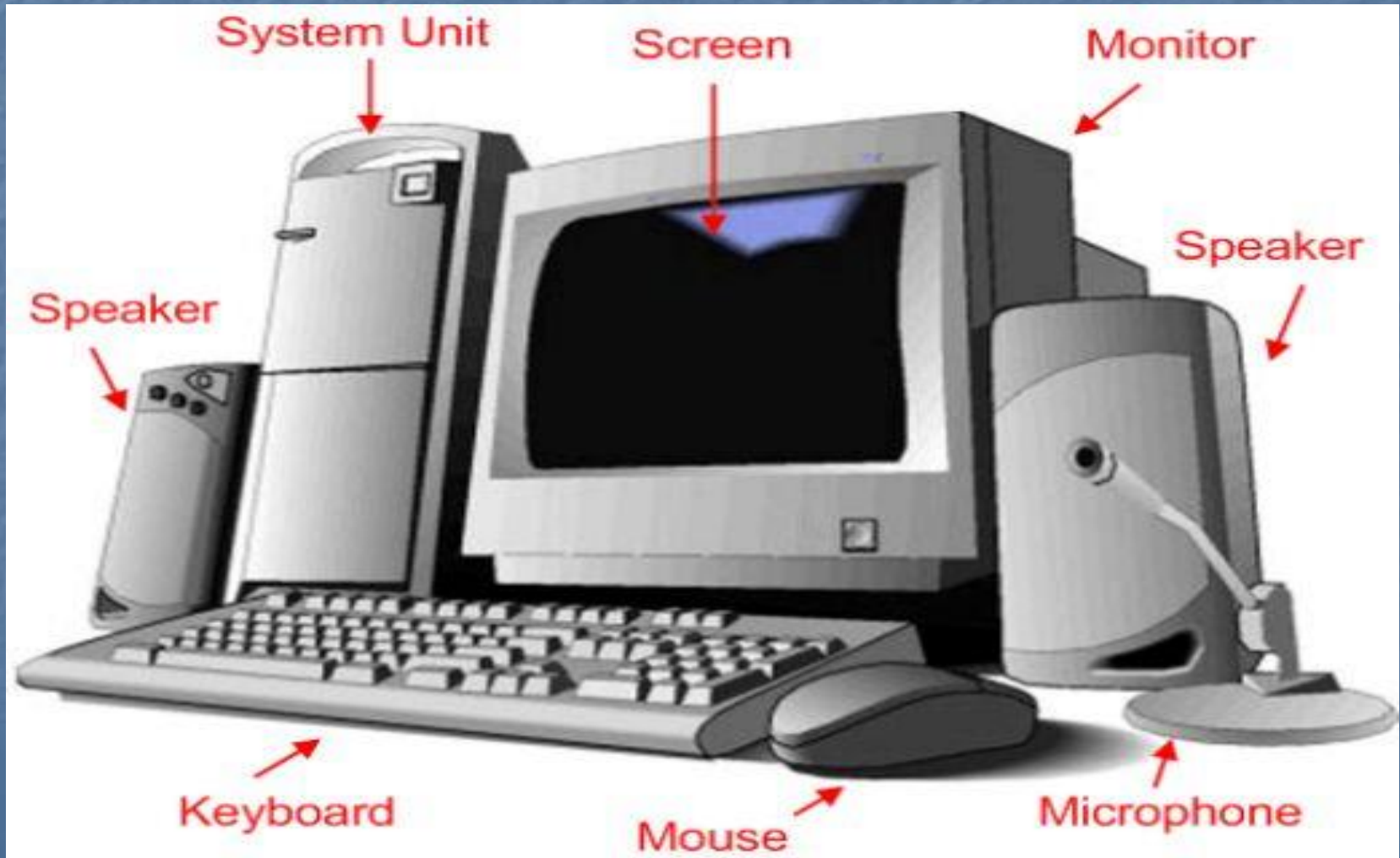


Table of Contents

- What is a Computer
- Computer Hardware
- Examples of Computer Hardware
- Computer Software
- Computer Input Devices
- Computer Output Devices
- The Central Processing Unit

Definition of Computer

- A device that computes, especially a programmable electronic machine that performs high-speed mathematical or logical operations or that assembles, stores, correlates, or otherwise processes information.



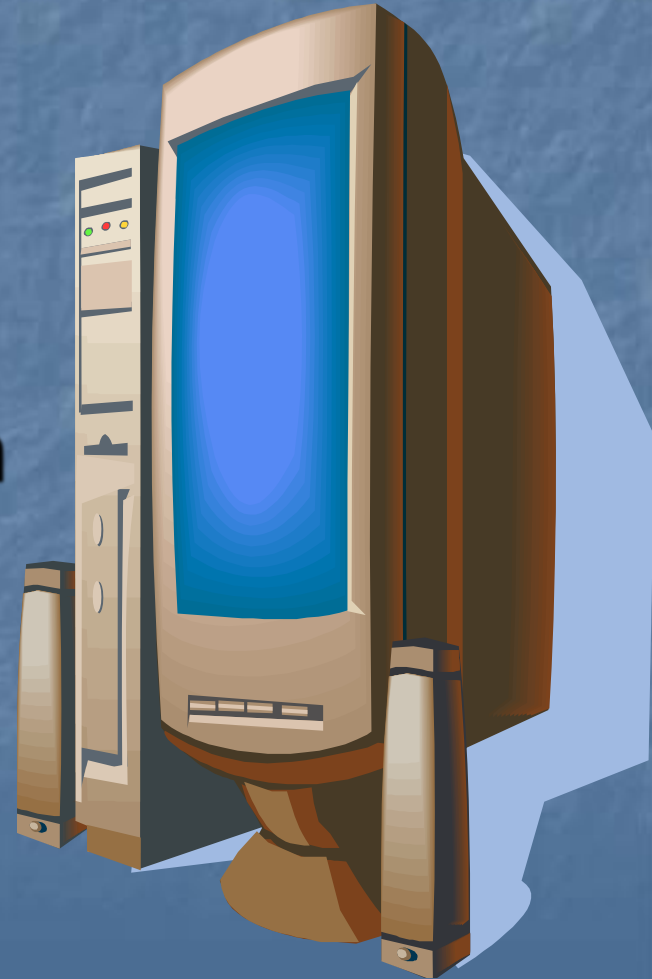
Computers Have Two Main Parts

1. Computer Hardware
2. Computer Software



What is Computer Hardware?

- Computer Hardware is the physical part of the computer system, the machinery and equipment.
- Parts of the computer “you can see”



Examples of Computer Hardware



Monitor: T.V. like screen used to show pictures and words



CPU: Central Processing Unit this is where most of the computer's calculations take place. In terms of computing power, the CPU is the most important element of a computing system.



Keyboard: This device is used to type information into the computer and contains the numbers 0-9.

More Computer Hardware



Mouse: a small device, which you move across the top of the desk to move the pointer or cursor on the screen.



Printer: used to make a paper copy of the information into the computer.



Image Scanner: an electronic device that generates a digital representation of an image for data input to a computer

What is Computer Software?

- Computer Software are programs that tell the computer what to do.

Examples

- Microsoft Word-word processing program
- Microsoft PowerPoint-presentation program
- Microsoft Excel-work book program used to track, calculate, and analyze numeric data

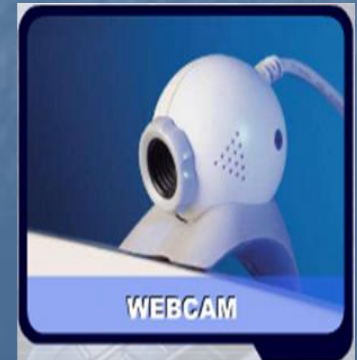


Computer Input Devices

- Computer Devices that input information in the computer

Examples

- Key Board
- Mouse
- Scanner
- Digital Camera



Computer Output Devices

- Computer Devices that output information from the computer.

Examples

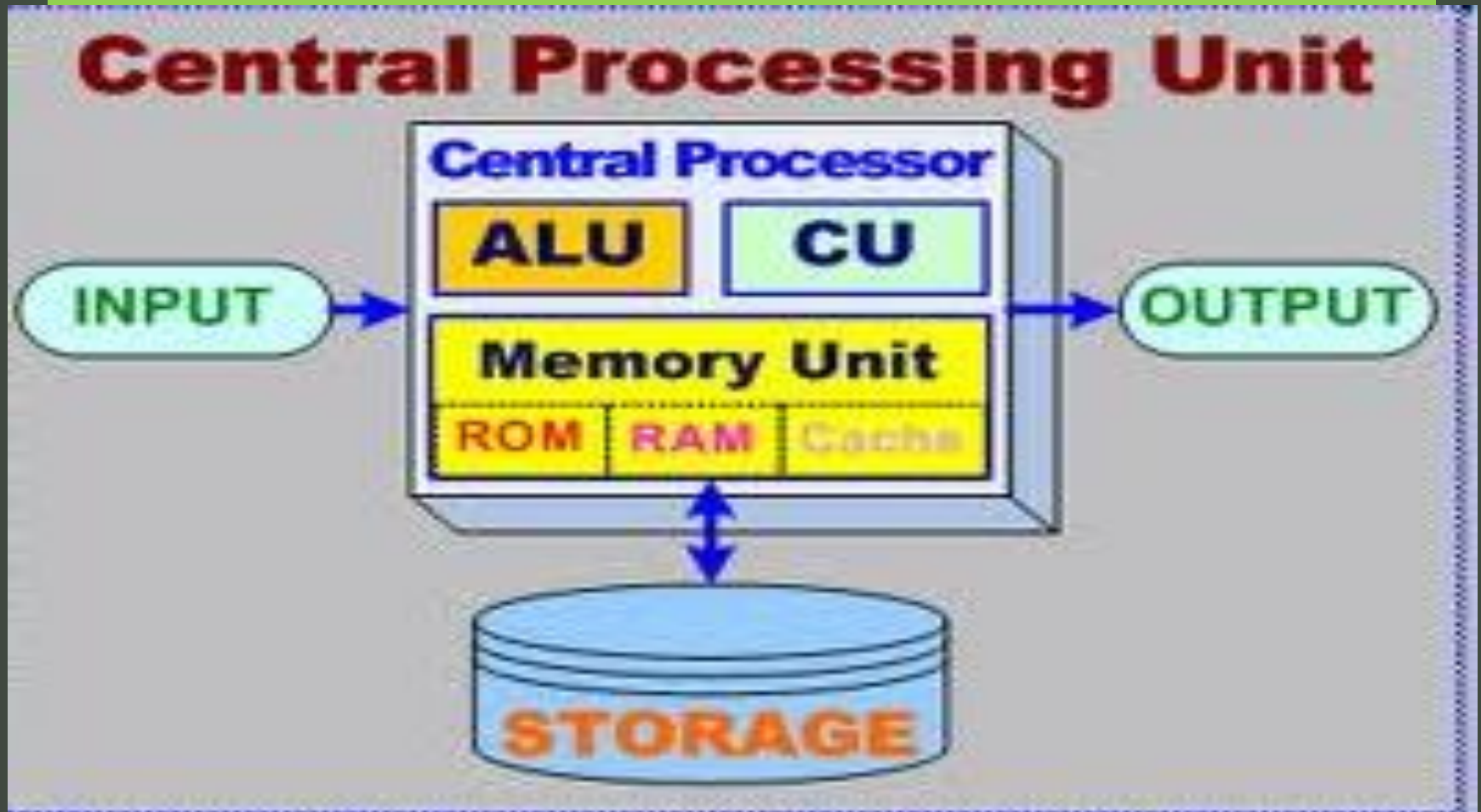
- Monitor
- Printer
- Speaker
- Headphones





The Central Processing Unit:
What Goes on Inside the Computer

The CPU



The CPU

- The central processing unit (CPU) is the portion of a computer system that carries out the instructions of a computer program, to perform the basic arithmetical, logical, and input and output operations of the system. It acts as the BRAIN OF COMPUTER



The CPU

- Converts data into information
- Control center
- Set of electronic circuitry that executes stored program instructions
- Two parts
 - Control Unit (CU)
 - Arithmetic Logic Unit (ALU)



Control Unit

CU

- Part of the hardware that is in-charge
- Directs the computer system to execute stored program instructions
- Communicates with other parts of the hardware



Arithmetic / Logic Unit

ALU

Performs arithmetic operations

Performs logical operations



Arithmetic Operations



Addition



Subtraction

Multiplication

Division



Logical Operations

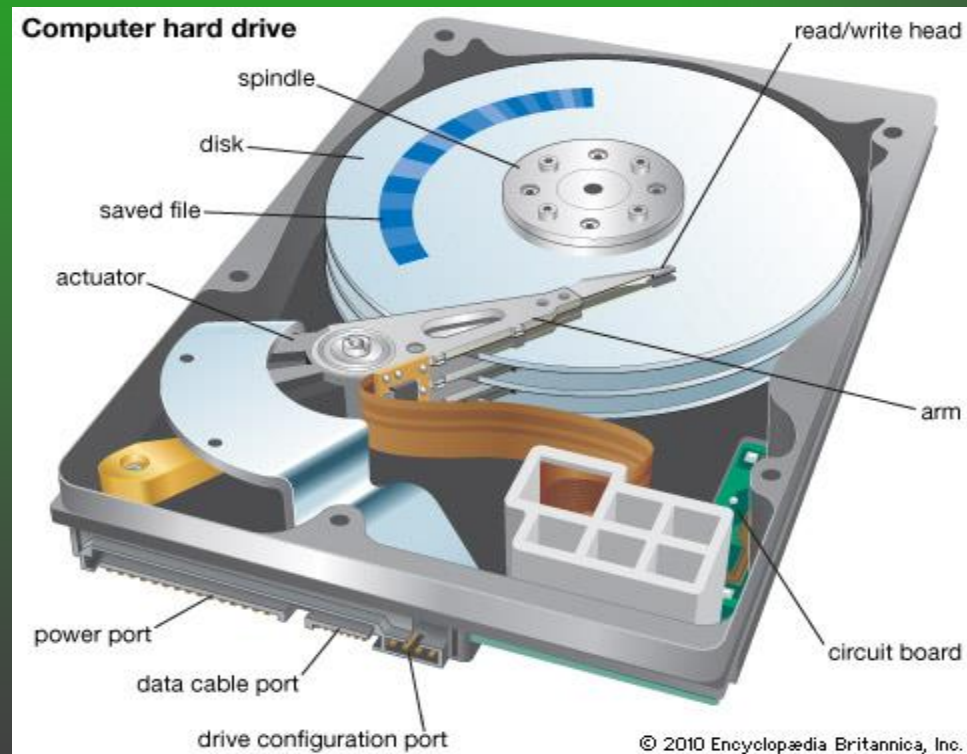
- Evaluates conditions
- Makes comparisons
- Can compare
 - Numbers
 - Letters
 - Special characters

AND
NOT
OR



Types of Storage

- Secondary
 - Data that will eventually be used
 - Long-term



Types of Storage

- Memory
 - Data that will be used in the near future
 - Temporary
 - Faster access than storage



Types of Storage

- Registers
 - Data immediately related to the operation being executed
 - Faster access than memory



Main Types of Memory

RAM

Random Access Memory

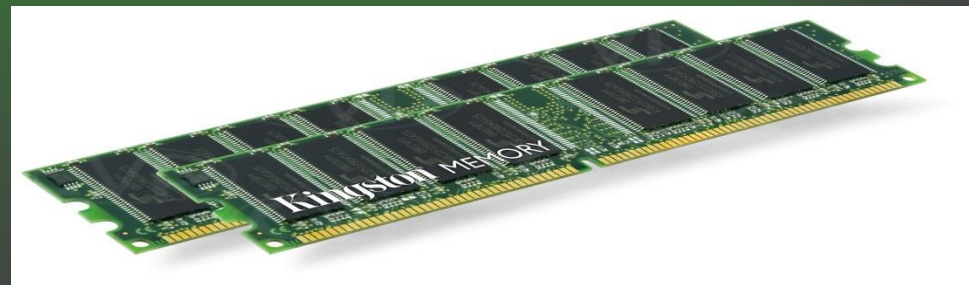
ROM

Read Only Memory



RAM Random Access Memory

- Random-access memory (RAM) is a form of computer data storage. A random-access memory device allows data items to be read and written in approximately the same amount of time, regardless of the order in which data items are accessed.



ROM Read Only Memory

- Read-only memory (ROM) is a class of storage medium used in computers and other electronic devices. Data stored in ROM can only be modified slowly, with difficulty, or not at all, so it is mainly used to distribute firmware (software that is very closely tied to specific hardware, and unlikely to need frequent updates).



Computer Software

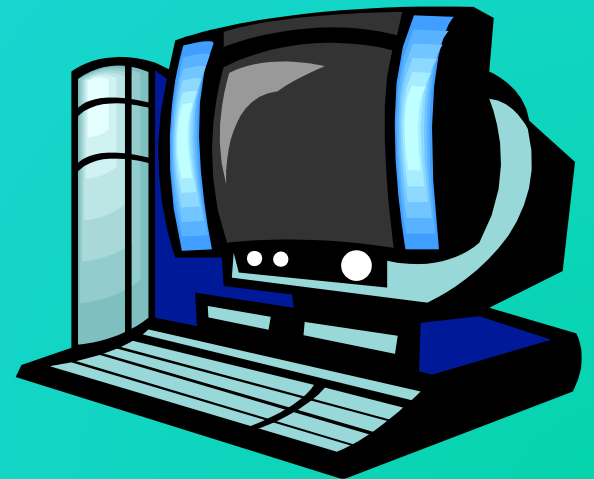
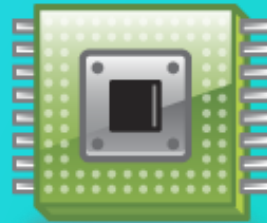
In this Lesson, You Will Learn:

- ❖ To distinguish between hardware and software
- ❖ To understand the role of software to the computer
- ❖ To identify main software categories
- ❖ Distinguish among different software programs

What is Hardware?

- ❖ Hardware-The physical devices that make up a computer system.
- ❖ Tangible components that can be seen and touched.

- Keyboard
- Printer
- Mouse
- chips



What is Software?

- ❖ **Software** - A program that tells a computer how to perform tasks.
- ❖ **Program** – aka application - a series of step by step instructions that tell the computer precisely what actions to perform.

What does Software do?

Software translates your commands into the language that computers understand.



Who Writes the Software?

Programmers – write instructions, or programs, to the computer so it is able to execute a task or operate properly.



Categories of Software

❖ **Operating System (OS) Software -**

Main program that makes your computer work.

❖ **Application Software -**

lets you do different tasks on your computer, such as writing reports and sending email

❖ **Utility Software –**

helps you control your computer and keep it in good running condition

Operating System (OS) Software

- ❖ **Operating System** - Software that controls all the other software programs and allows a computer to perform basic tasks.
- ❖ Every computerized device needs an operating system (OS) in order to work.



Graphic User Interface

- ❖ **GUI – Graphic User Interface** - uses graphics or pictures to help the user navigate within the computer system



Types of Operating Systems

- ❖ **Microsoft Windows** – Most popular operating system for PC's
- ❖ **Mac OS** – used by Apple computers
- ❖ **Linux** – powerful OS often used in large networks and business environments
- ❖ **Handheld Operating System** – used in tablets and cell phones

Windows OS

- ❖ Windows 95
- ❖ Windows 98
- ❖ Windows NT
- ❖ Windows 2000
- ❖ Windows XP
- ❖ Windows Vista
- ❖ Windows 7
- ❖ Windows 8

Application Software

Application Software - A software program that lets you perform specific tasks, like organizing info, creating reports or printing a picture.



Types of Application Software

Type of Software	What It Lets You Do	Examples
Web Browsers	Visit Web sites on the Internet	Microsoft Internet Explorer®, Apple Safari®, Netscape®
E-mail	Exchange messages and files with other computer users	Microsoft Outlook®, Eudora®
Word Processing	Create letters, term papers, reports, newsletters, etc.	Microsoft Word®, Corel WordPerfect®
Spreadsheet	Work with numbers and calculations to create tables, charts, and graphs	Microsoft Excel®, Lotus 1-2-3®
Database	Organize and retrieve large amounts of information	Microsoft Access®, FileMaker Pro®
Presentation	Create a slide presentation to show a group of people	Microsoft PowerPoint®, Apple Keynote®

Utility Software

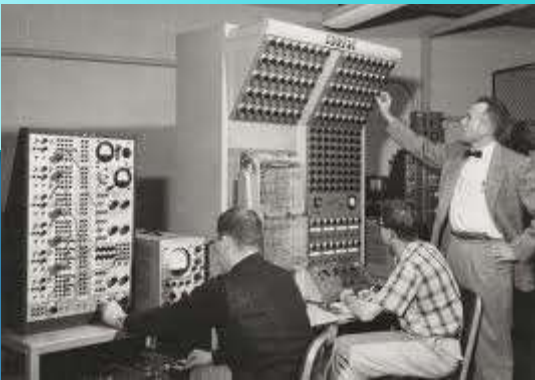
❖ **Utility Software** - is system software designed to help analyze, configure, optimize or maintain a computer.

❖ **Types of Utility Software:**

- Anti Virus Programs
- File managers
- Disk Cleaners



The Five Generations of Computers



First generation computers (1940-1956)

- The first computers used vacuum tubes for circuitry and magnetic drums for memory.
- They were often enormous and taking up entire room.
- First generation computers relied on machine language.
- . They were very expensive to operate and in addition to using a great deal of electricity, generated a lot of heat, which was often the cause of malfunctions.
- The UNIVAC and ENIAC computers are examples of first-generation computing devices.

First generation computers



Second generation computers (1956-1963)

- Transistors replaced vacuum tubes and ushered in the second generation of computers.
- Second-generation computers moved from cryptic binary machine language to symbolic.
- High-level programming languages were also being developed at this time, such as early versions of COBOL and FORTRAN.
- These were also the first computers that stored their instructions in their memory.

Second generation computers



Third generation computers (1964-1971)

- The development of the integrated circuit was the hallmark of the third generation of computers.
- Transistors were miniaturized and placed on siliconchips, called semiconductors.
- Instead of punched cards and printouts, users interacted with third generation computers through keyboards and monitors and interfaced with an operating system.
- Allowed the device to run many different applications at one time.

Third generation computers



Fourth generation computers (1971-1980)

- The microprocessor brought the fourth generation of computers, as thousands of integrated circuits were built onto a single silicon chip.
- The Intel 4004 chip, developed in 1971, located all the components of the computer.
- From the central processing unit and memory to input/output controls—on a single chip.
- . Fourth generation computers also saw the development of GUIs, the mouse and handheld devices.

Fourth generation computers



Fifth generation computers (1980 - until now)

- Fifth generation computing devices, based on artificial intelligence.
- Are still in development, though there are some applications, such as voice recognition.
- The use of parallel processing and superconductors is helping to make artificial intelligence a reality.
- The goal of fifth-generation computing is to develop devices that respond to natural language input and are capable of learning and self-organization.

Fifth generation computers



Algorithms

OBJECTIVES

After reading this chapter, the reader should be able to:

- Understand the concept of an algorithm.
- Define and use the three constructs for developing algorithms: sequence, decision, and repetition.
- Understand and use three tools to represent algorithms: flowchart, pseudocode, and structure chart.

Definition of Algorithm

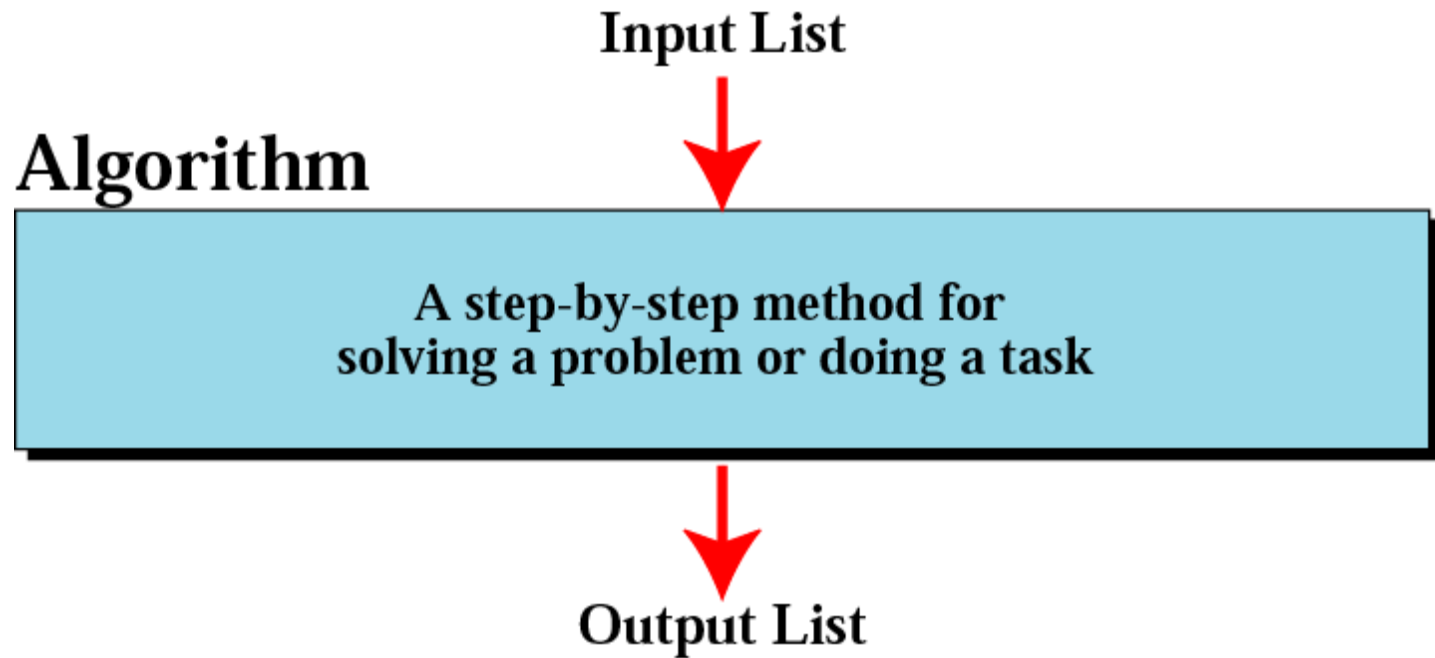
Step by step procedure designed to perform an operation, will lead to the sought result if followed correctly. Algorithms have a definite beginning and a definite end, and a finite number of steps. An algorithm produces the same output information given the same input information, and several short algorithms can be combined to perform complex tasks such as writing a computer program. a diagnosis, a problem solving routine, are some common examples of simple algorithms.

Characteristics Of An Algorithm

1. Algorithms always have a definite starting point and an end point. These points are generally marked with the words like Start, Begin, End, Stop etc.
2. They consist of finite number of steps.
3. They always relate to a specific problem or you can say that they are written for a given problem.
4. They serve as *foundation stone for programming*.
5. They are written in easy language.

Figure 8-1

Definition of an algorithm used in a computer



Three constructs

1 - Sequence: the algorithm is a set of sequential instructions, these instructions may be either simple or of the following two types.

2 - Selection: Some problems can not be solved by a simple sequence of instructions, you may need to test some of the conditions and look at the result of the test, if the result is correct track contains a sequential instructions, and if the wrong track the last track of the various instructions. This method is called the decision or choice.

3 - Repetition: When you solve some of the problems it is necessary to reproduce the same sequence of steps a number of times. This so-called

Three constructs

```
do action 1  
do action 2  
...  
...  
do action n
```

a. Sequence

```
if a condition is true,  
then  
    do a series of actions  
else  
    do another series of actions
```

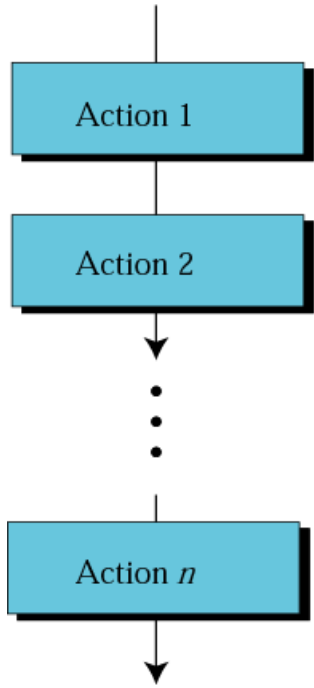
b. Decision

```
while a condition is true,  
    do action 1  
    do action 2  
    ...  
    ...  
    do action n
```

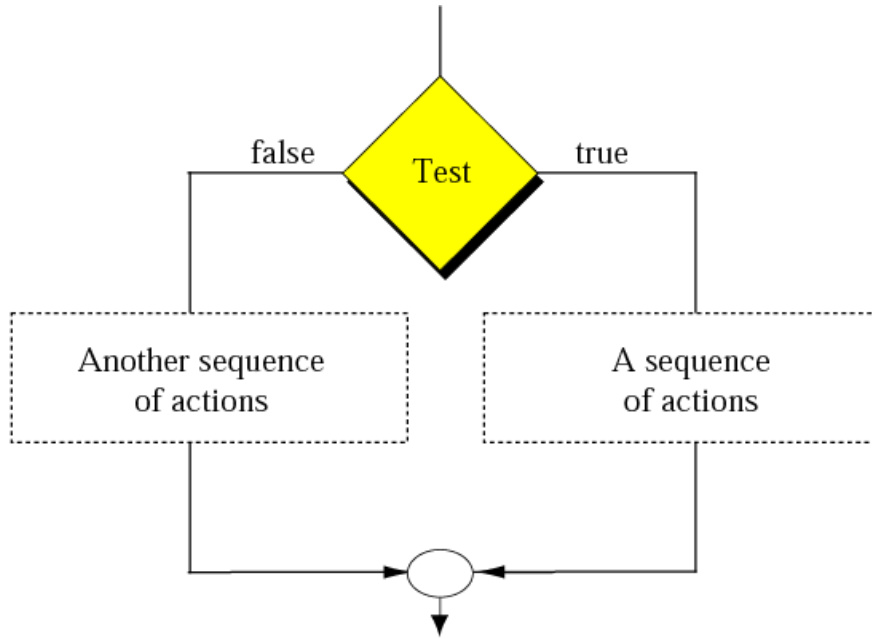
c. Repetition

Figure 8-7

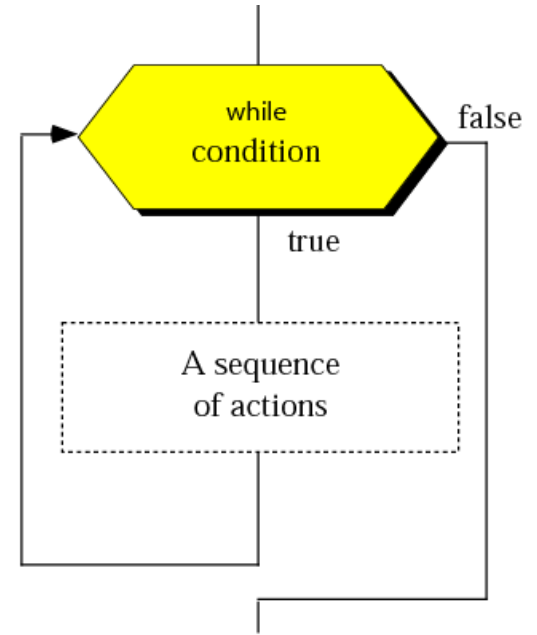
Flowcharts for three constructs



a. Sequence



b. Decision



c. Repetition

Pseudocode for three constructs

```
action 1  
action 2  
⋮  
action n
```

a. Sequence

```
if (condition)  
  then  
    action  
    action  
    ...  
  else  
    action  
    action  
    ...  
End if
```

b. Decision

```
while (condition)  
  action  
  action  
  ...  
End while
```

c. Repetition

Example 1

write an algorithm to calculate the size of the ball, knowing that the radius = 15 cm?

Algorithm:

1- start

2- let $R = 15$

3- let $\pi = 3.14$

4- $S = \frac{4}{3} * \pi * R^2$

5- print S

6- End

Example 2

Write an algorithm to find the largest of given three numbers.

Algorithm:

1. Start
2. Read three numbers A, B and C
3. Let Big=0
4. IF $A > B$ Then Big=A Else Big=B
5. IF $C > \text{Big}$ Then Big=C
6. Print Big
7. End

Example 3

write algorithm to find the result of equation

Algorithm:

$$f(x) = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

Step1: Start

Step2: Read/input x

Step3: If X Less than zero then F=-X

Step4: if X greater than or equal zero then F=X

Step5: Print F

Step6: End

Example 4

write algorithm to calculate even numbers between 0 and 99

Algorithm:

1. Start
2. $I = 0$
3. Write I in standard output
4. $I = I + 2$
5. If ($I \leq 98$) then go to line 3
6. End

Example 5

Design an algorithm with a natural number, n , as its input which calculates the following formula and writes the result in the standard output:

$$S = \frac{1}{2} + \frac{1}{4} + \dots + \frac{1}{n}$$

Algorithm:

1. Start
2. Read n
3. $I = 2$ and $S = 0$
4. $S = S + 1/I$
5. $I = I + 2$
6. If ($I \leq n$) then go to line 4
else write S in standard output
7. End

Example 6

Design an algorithm which gets a natural value, n , as its input and calculates odd numbers equal or less than n . Then write them in the standard output:

Algorithm:

1. Start
2. Read n
3. $I = 1$
4. Write I
5. $I = I + 2$
6. If ($I \leq n$) then go to line 4
7. End

Example 7

Write an algorithm for calculating and printing factorial (!) of a given number.

Algorithm:

1. Start
2. Read a number, A
3. Let $I=1$
4. Let $Sum=1$
5. $Sum=Sum*I$
6. $I=I+1$
7. If I is not $> A$ perform step 5, 6 and 7
8. Print Sum
9. End

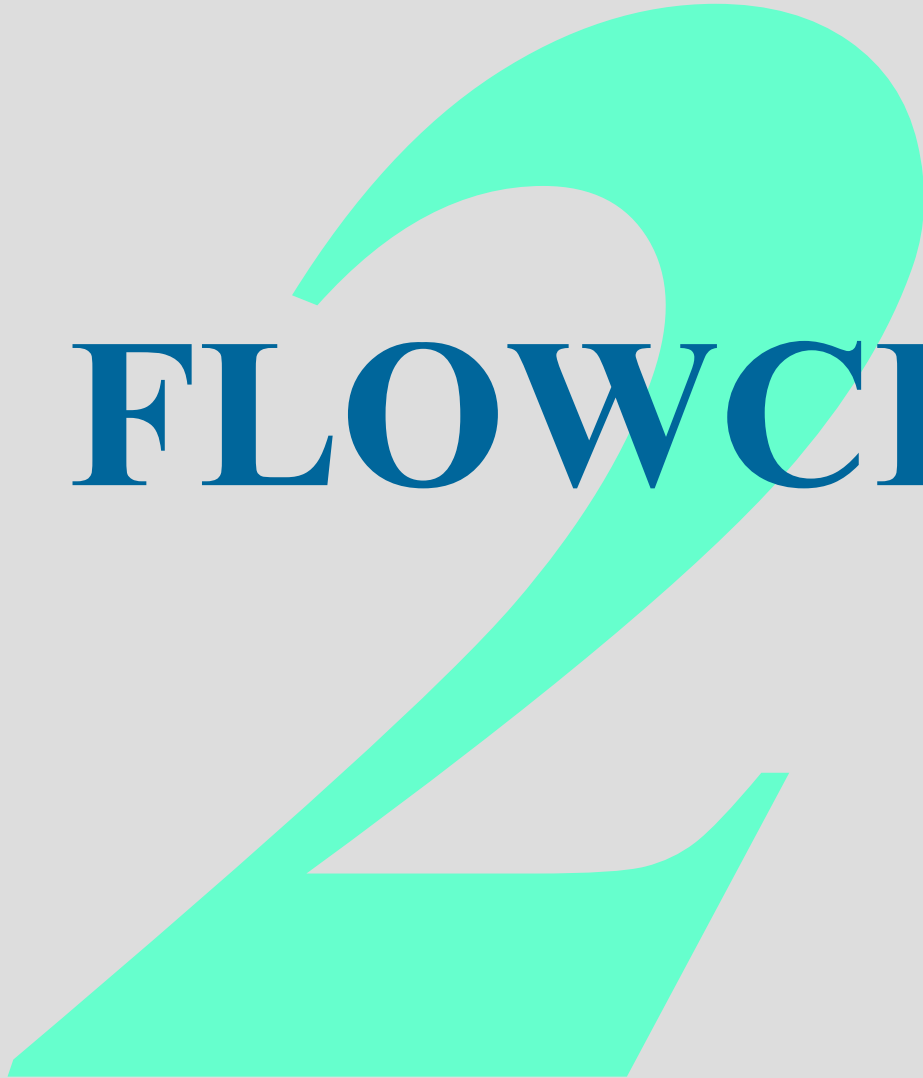
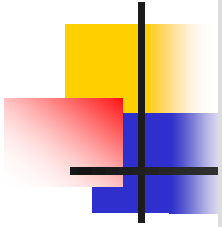
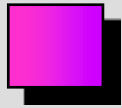
Example 8

Write an algorithm to find the largest of 1000 numbers.

Algorithm:

Input: 1000 positive integers

1. Set Largest to 0
 2. Set Counter to 0
 3. while (Counter less than 1000)
 - 3.1 if (the integer is greater than Largest)
 - then
 - 3.1.1 Set Largest to the value of the integer
 - End if
 - 3.2 Increment Counter
 - End while
 4. Return Largest
- End**



FLOWCHART



FLOWCHARTS

- A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows.



FLOWCHARTS

This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields



Best Practices in Flowcharting

- **1. Proper Form is Essential:** In drawing a proper flowchart, all necessary requirements should be listed out in logical order.
- **2. Clarity is Paramount:** The flowchart should be clear, neat and easy to follow. There should not be any room for ambiguity in understanding the flowchart.
- **3. Stick to the Right Direction:** The usual direction of the flow of a procedure or system is from left to right or top to bottom.

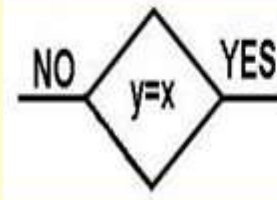
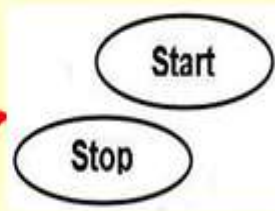


Best Practices in Flowcharting

- **4. Standard for Flow Lines:** Ideally just one flow line should come out from a process symbol. While only one flow line should enter a decision symbol, around three flow lines (depending on the answer) should leave the decision symbol. Additionally, only one flow line is utilized together with a terminal symbol.
- **5. Be Concise, not Copious:** Write within standard symbols briefly.
- **6. Logic precedes Everything:** If you are dealing with a complex flowchart then use connector symbols to minimize the number of flow lines. Ditch the intersection of flow lines to ensure effectiveness and better communication. It is imperative that your flowchart has a logical start and finish.

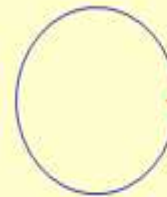
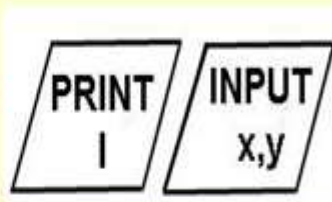
Flowchart Building Blocks

Terminal



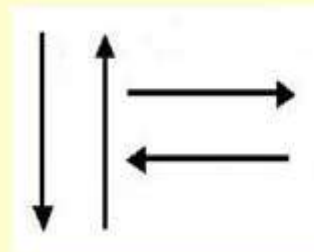
Decision

Input/Output
Operation



Connector

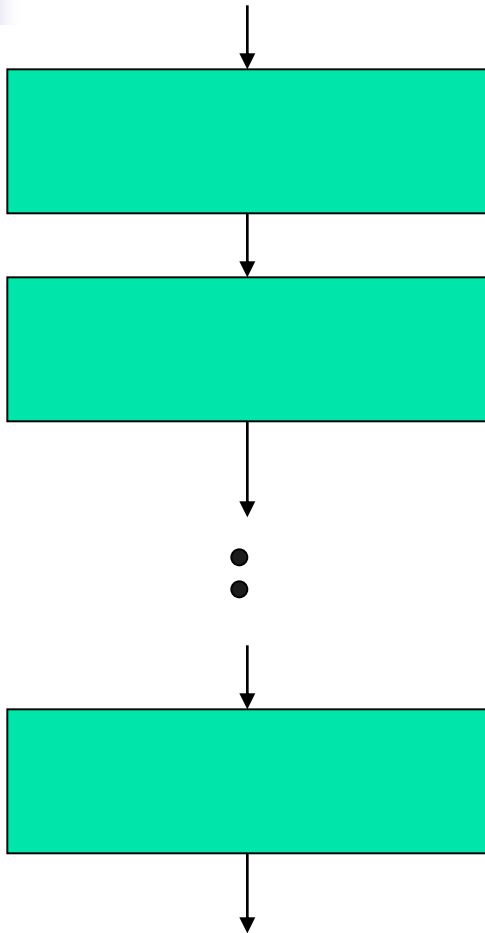
Process



Flow Line



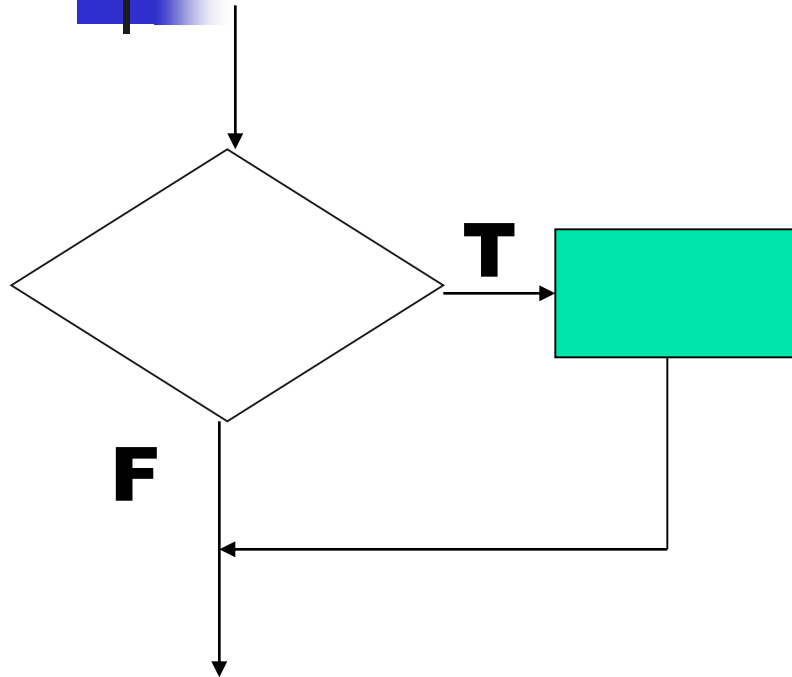
1) Sequence :



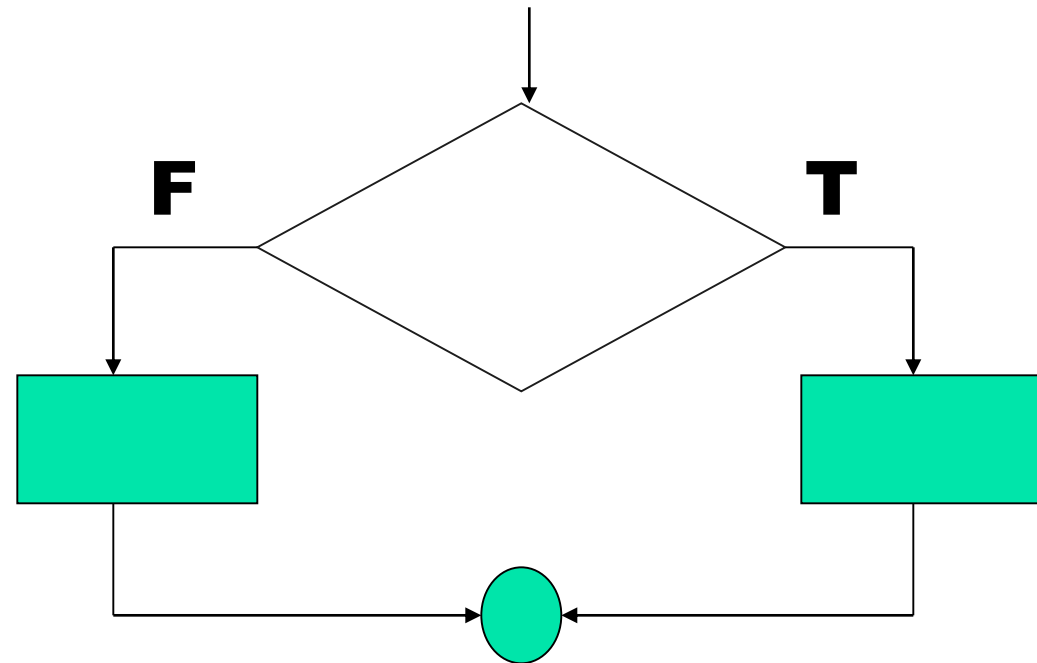
Example:

$$C=A+B$$

2) Selection: if & if/else

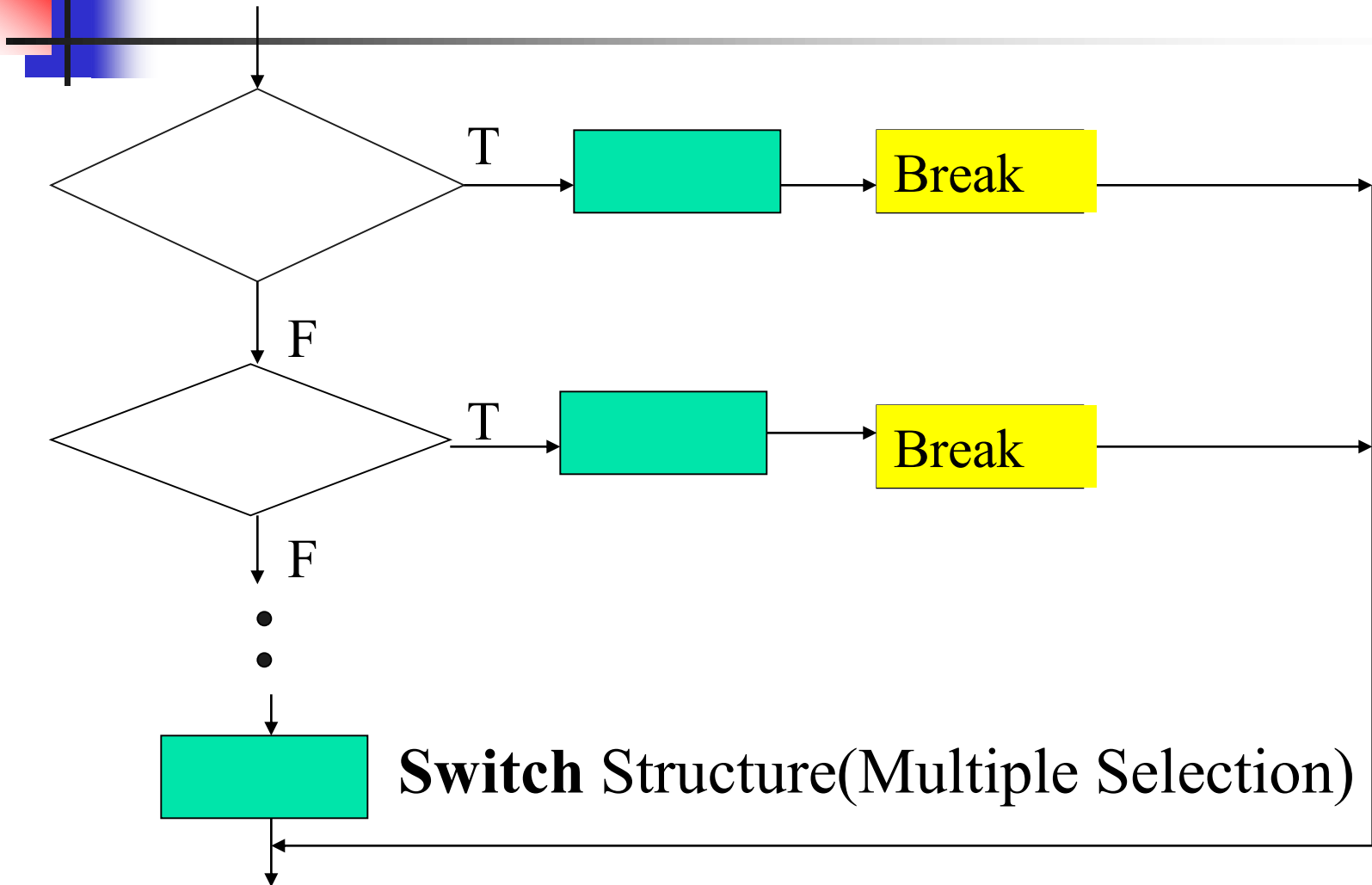


if Structure (Single Selection)

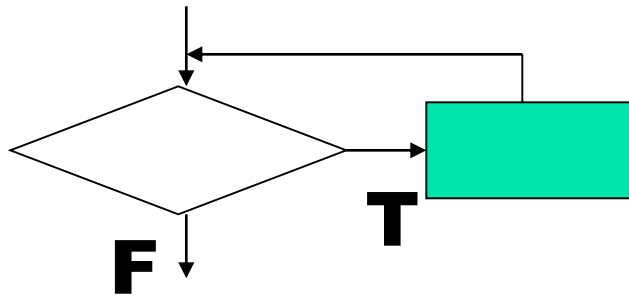


if/else Structure (Double Selection)

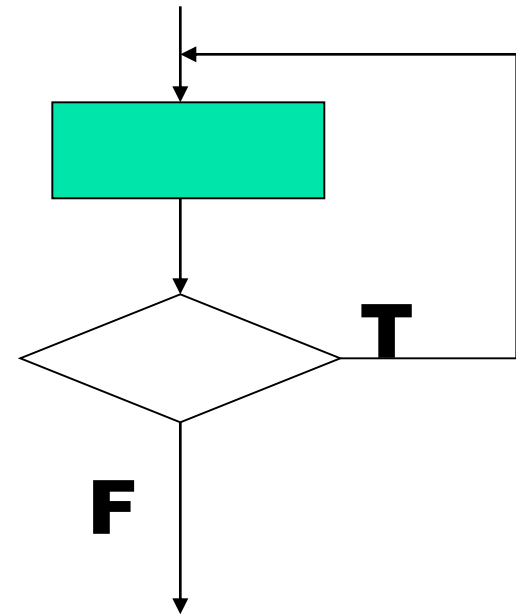
2) Selection: switch



3) Repetition: while & do/while

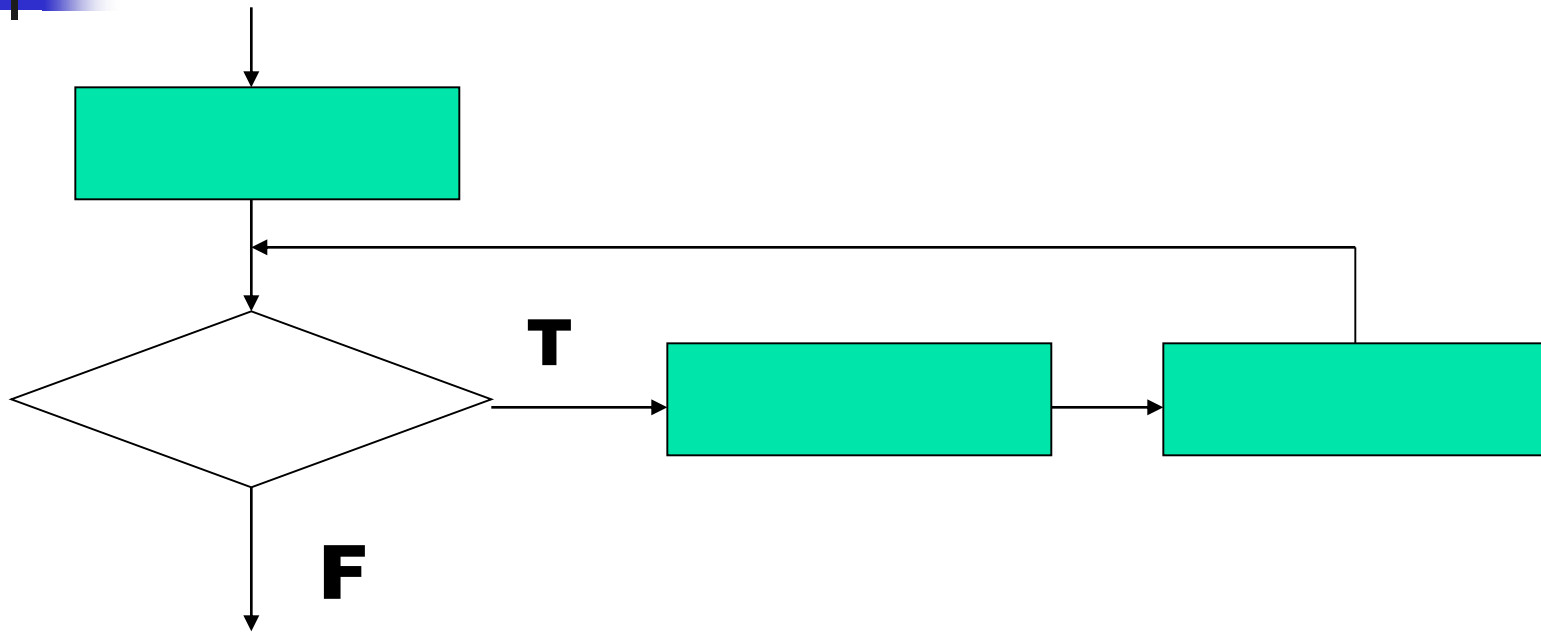


while Structure



do/while structure

3) Repetition: for loop

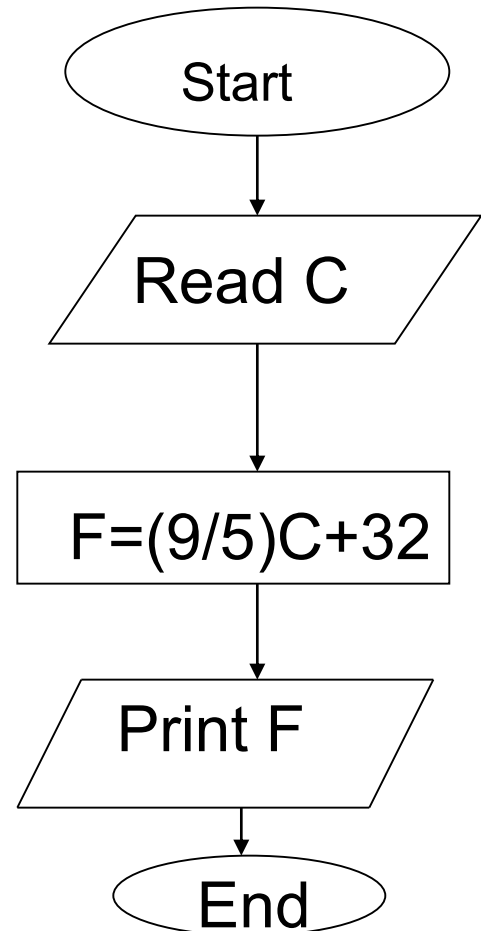


for Structure

Example 1

While the flowchart used is shown bellow (the used control structure is **Sequence**):

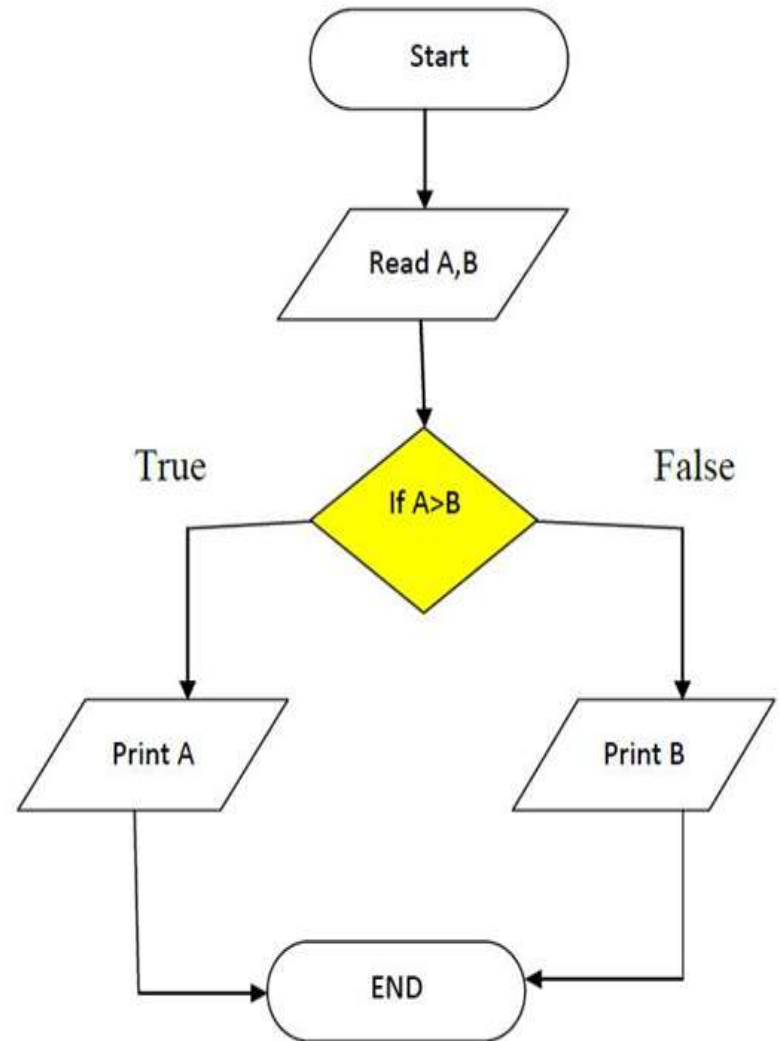
- Step1: Start
- Step2: Read/input C
- Step3: $F=(9/5)C+32$
- Step4: Print F
- Step5: End



Example 2

Algorithm for find the greater number between two numbers

- Step1: Start
- Step2: Read/input A and B
- Step3: If A greater than B then Print A
- Step4: if B greater than A then Print B
- Step5: End



Example 3:

Write an algorithm and draw a flowchart to read the grades of five students , find the average of these grades and print it

Algorithm:

Let counter to 1

Let total to zero

Let average to zero

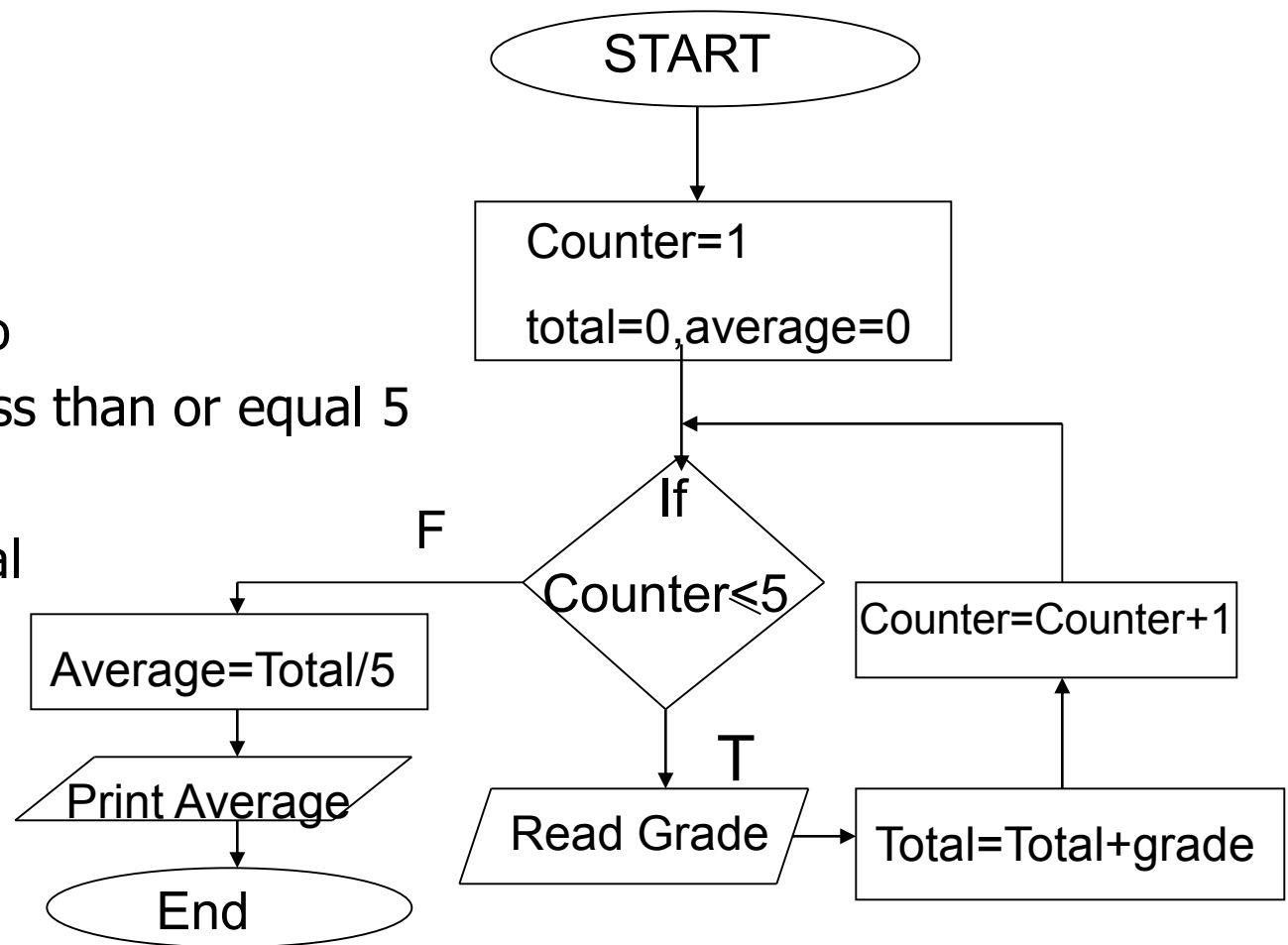
While counter is less than or equal 5

Input the grade

Add grade into total

average=total/5

Print average



Example4:

A student took final exams of 10 courses. Write an algorithm and draw a flowchart to read the grades and find their sum and print it.

Algorithm:

Let Count to 1

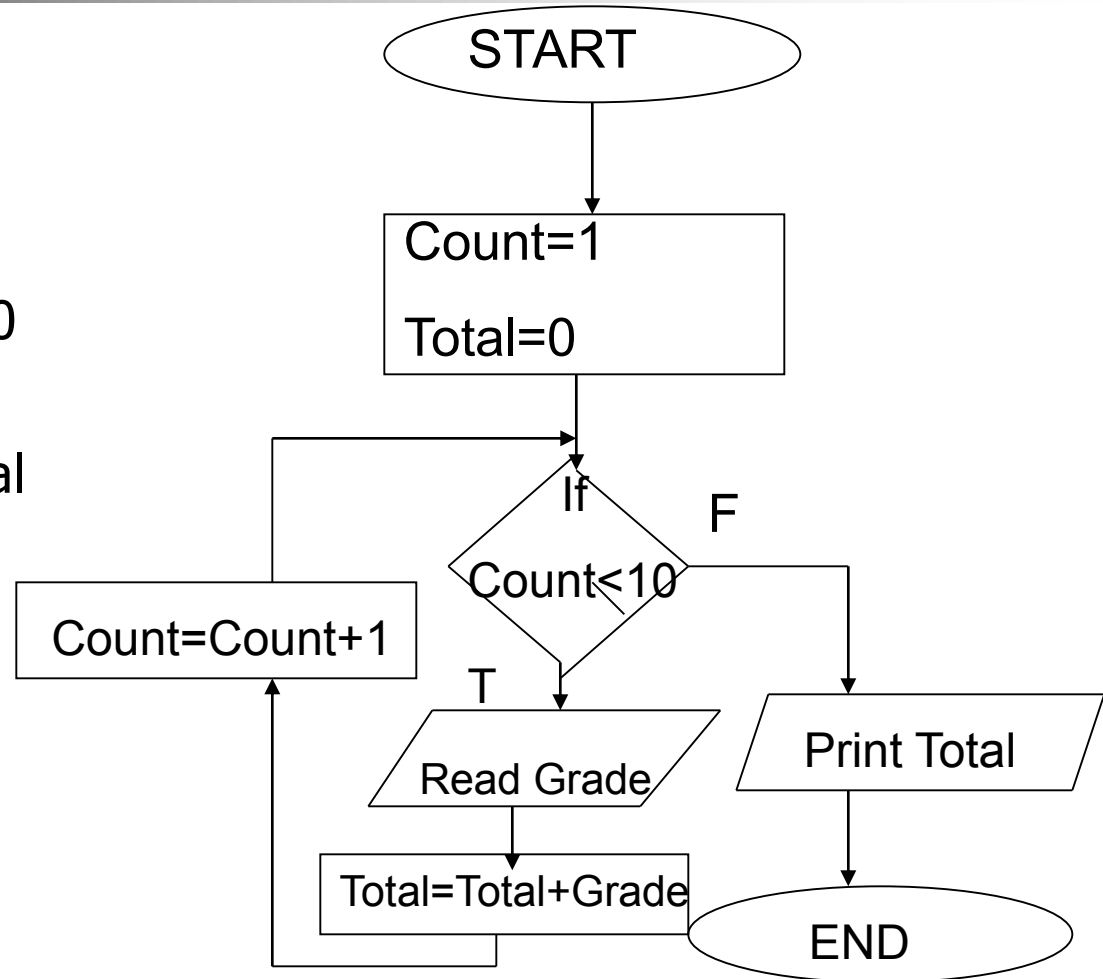
Let Total to Zero

For Count=1 to 10

Read a Grade

Add Grade to Total

Print Total



Example 5:

Write an algorithm and draw a flowchart to read a grade, test it if PASS (more than or equal 50) or FAIL, and print the result .

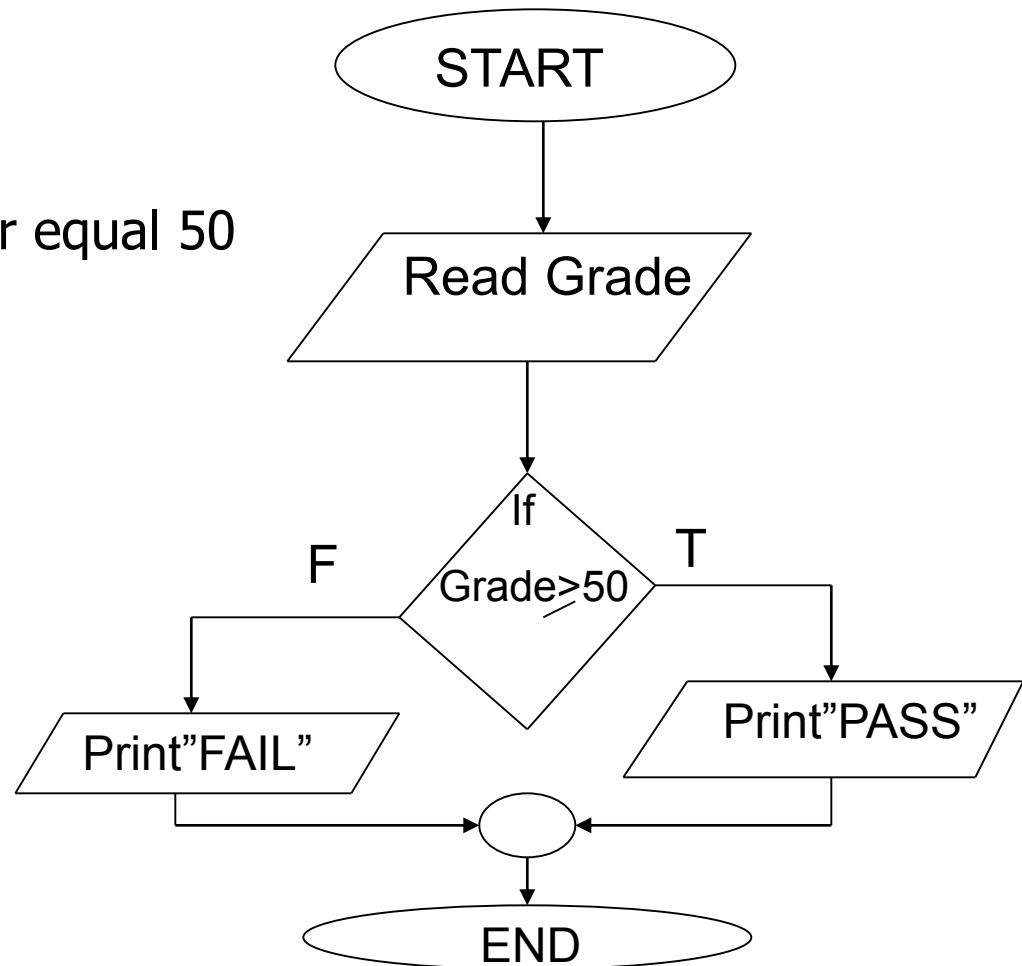
Algorithm :

Read a Grade

If Grade more than or equal 50

 Print PASS

Else Print FAIL

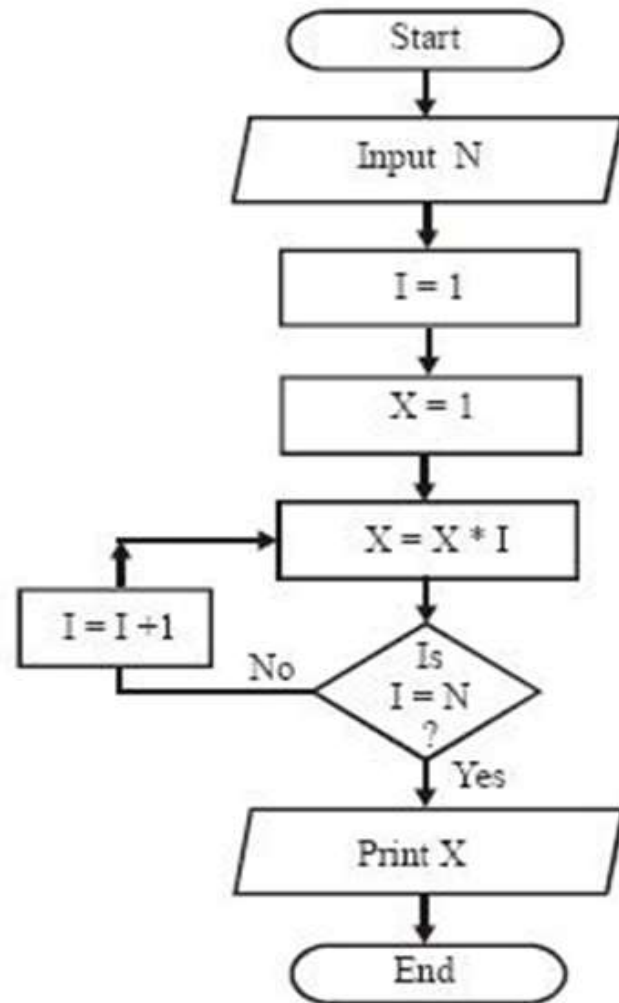


Example 6 :

Write an algorithm and draw a flowchart calculating and printing factorial (!) of a given number.

Algorithm:

1. Start
2. Read a number , N
3. Let I=1
4. Let X=1
5. $X=X*I$
6. $I=I+1$
7. If $I = N$ perform step 5, 6 and 7
8. Print Sum
9. End

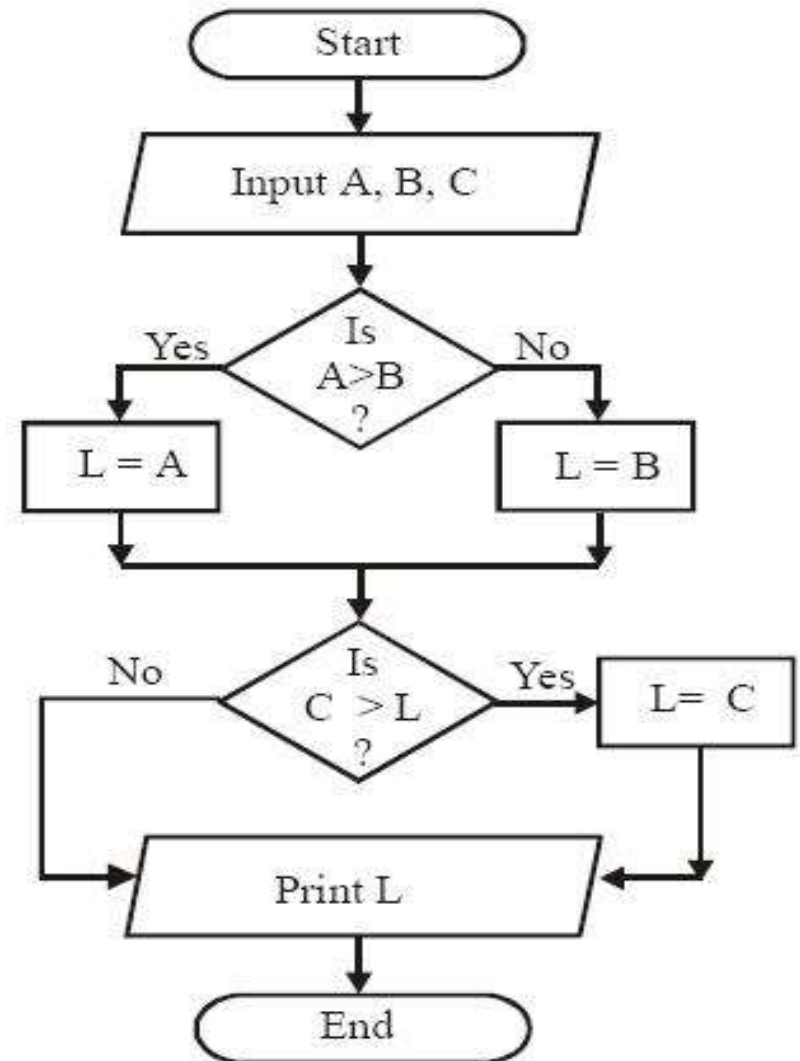


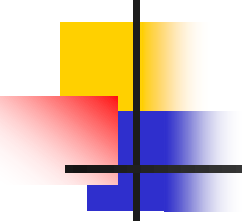
Example 7 :

Write an algorithm and Draw a flowchart to find the larger of the three given numbers.

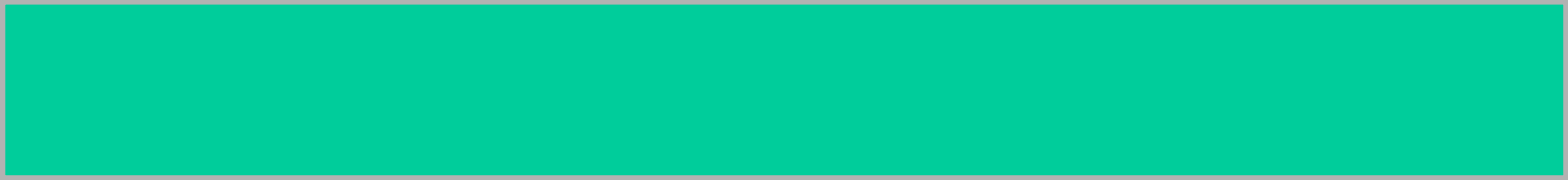
Algorithm:

1. Start
2. Read three numbers A B and C
3. Let Big=0
4. IF $A > B$ Then Big=A
Else Big=B
5. IF $C > \text{Big}$ Then Big=C
6. Print Big
7. End



- 
-
- Write an algorithm and draw a flowchart calculating and printing factorial (!) of a given number.

Number Systems



Objectives

After studying this chapter, the student should be able to:

- Understand the concept of number systems.
- Distinguish between non-positional and positional number systems.
- Describe the decimal, binary, hexadecimal and octal system.
- Convert a number in binary, octal or hexadecimal to a number in the decimal system.
- Convert a number in the decimal system to a number in binary, octal and hexadecimal.
- Convert a number in binary to octal and vice versa.
- Convert a number in binary to hexadecimal and vice versa.

2-1 INTRODUCTION

A **number system** defines how a number can be represented using distinct symbols. A number can be represented differently in different systems. For example, the two numbers $(2A)_{16}$ and $(52)_8$ both refer to the same quantity, $(42)_{10}$, but their representations are different.

$$(2A)_{16} = (52)_8 = (42)_{10}$$

Several number systems have been used in the past and can be categorized into two groups: **positional** and **non-positional** systems. Our main goal is to discuss the positional number systems, but we also give examples of non-positional systems.

Common Number Systems

Summary of the four positional number systems

<i>System</i>	<i>Base</i>	<i>Symbols</i>	<i>Examples</i>
Decimal	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	2345.56
Binary	2	0, 1	$(1001.11)_2$
Octal	8	0, 1, 2, 3, 4, 5, 6, 7	$(156.23)_8$
Hexadecimal	16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	$(A2C.A1)_{16}$

Quantities/Counting (1 of 3)

Decimal	Binary	Octal	Hexa- decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Quantities/Counting (2 of 3)

Decimal	Binary	Octal	Hexa- decimal
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

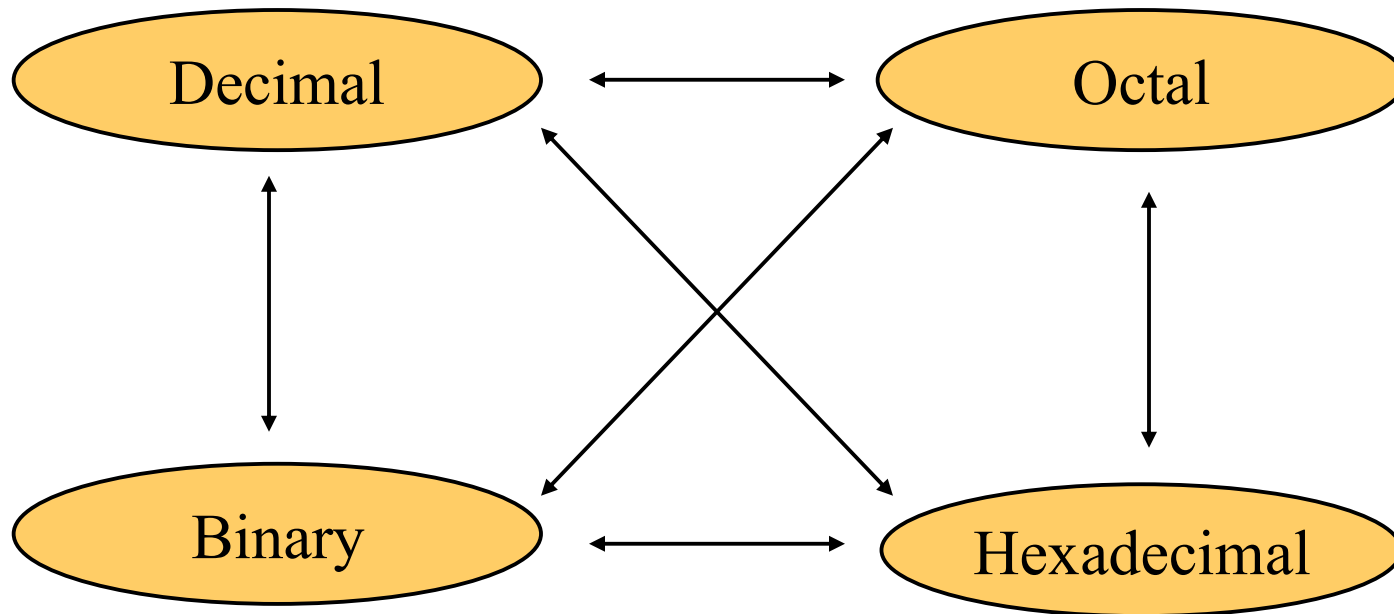
Quantities/Counting (3 of 3)

Decimal	Binary	Octal	Hexa- decimal
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17

Etc.

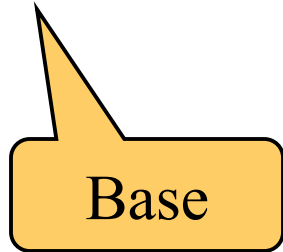
Conversion Among Bases

- The possibilities:

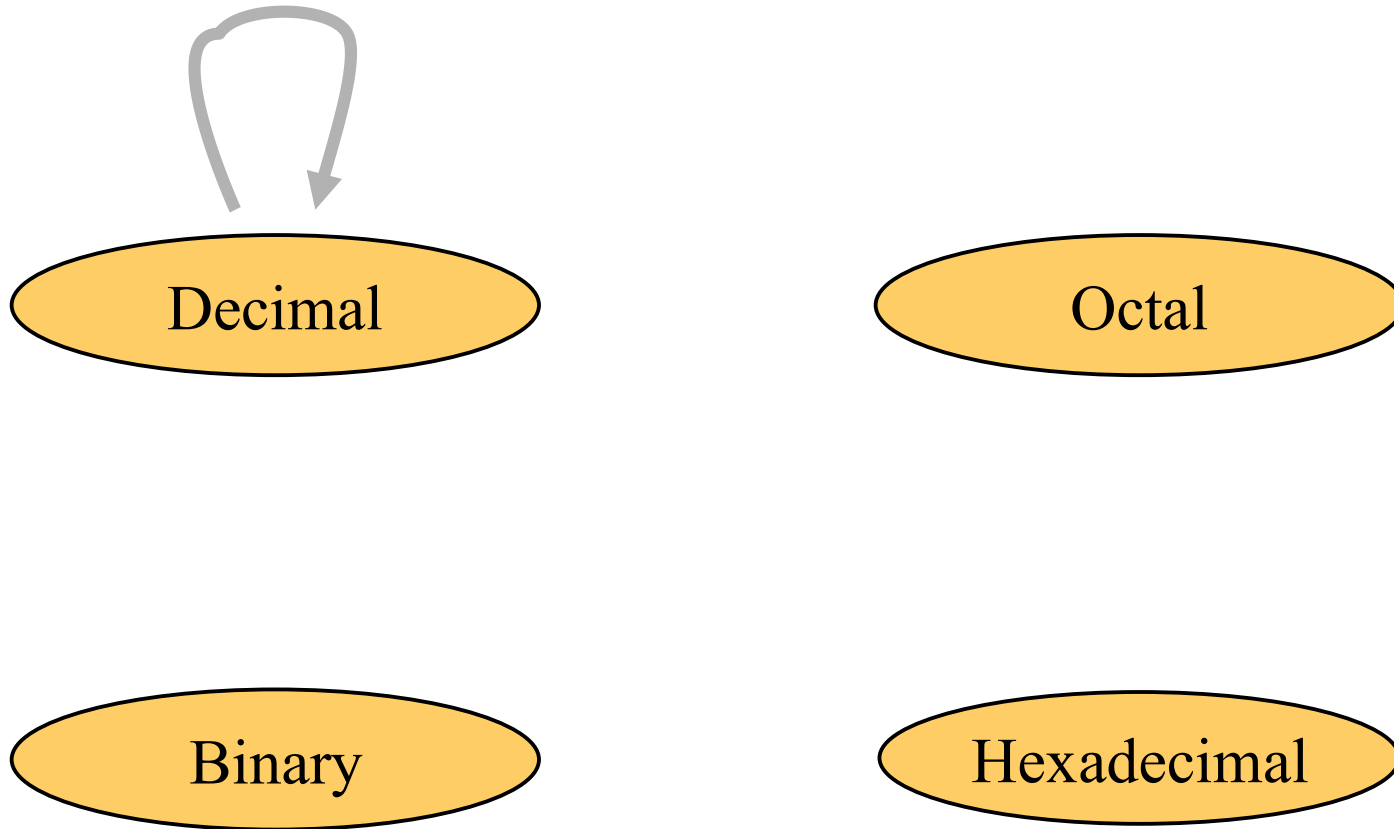


Quick Example

$$25_{10} = 11001_2 = 31_8 = 19_{16}$$



Decimal to Decimal (just for fun)



$125_{10} \Rightarrow$

5	\times	10^0	$=$	5
2	\times	10^1	$=$	20
1	\times	10^2	$=$	100
				<hr/>
				125

Weight

Base

Why Binary System?

- Computers are made of a series of switches
- Each switch has two states: ON or OFF
- Each state can be represented by a number
 - 1 for “ON” and 0 for “OFF”

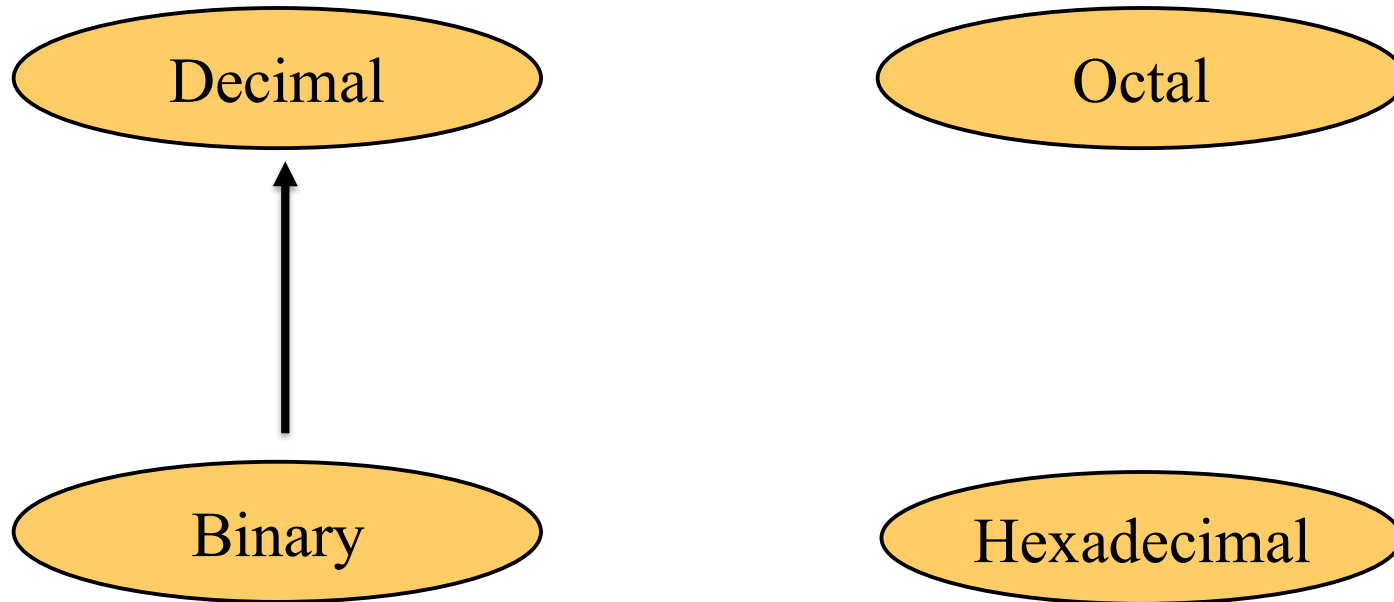
Example 2.17

An alternative method for converting a small decimal integer (usually less than 256) to binary is to break the number as the sum of numbers that are equivalent to the binary place values shown:

Place values	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Decimal equivalent	128	64	32	16	8	4	2	1

Decimal 165 =	128	+	0	+	32	+	0	+	0	+	4	+	0	+	1
Binary	1		0		1		0		0		1		0		1

Binary to Decimal



Binary to Decimal

- Technique
 - Multiply each bit by 2^n , where n is the “weight” of the bit
 - The weight is the position of the bit, starting from 0 on the right
 - Add the results

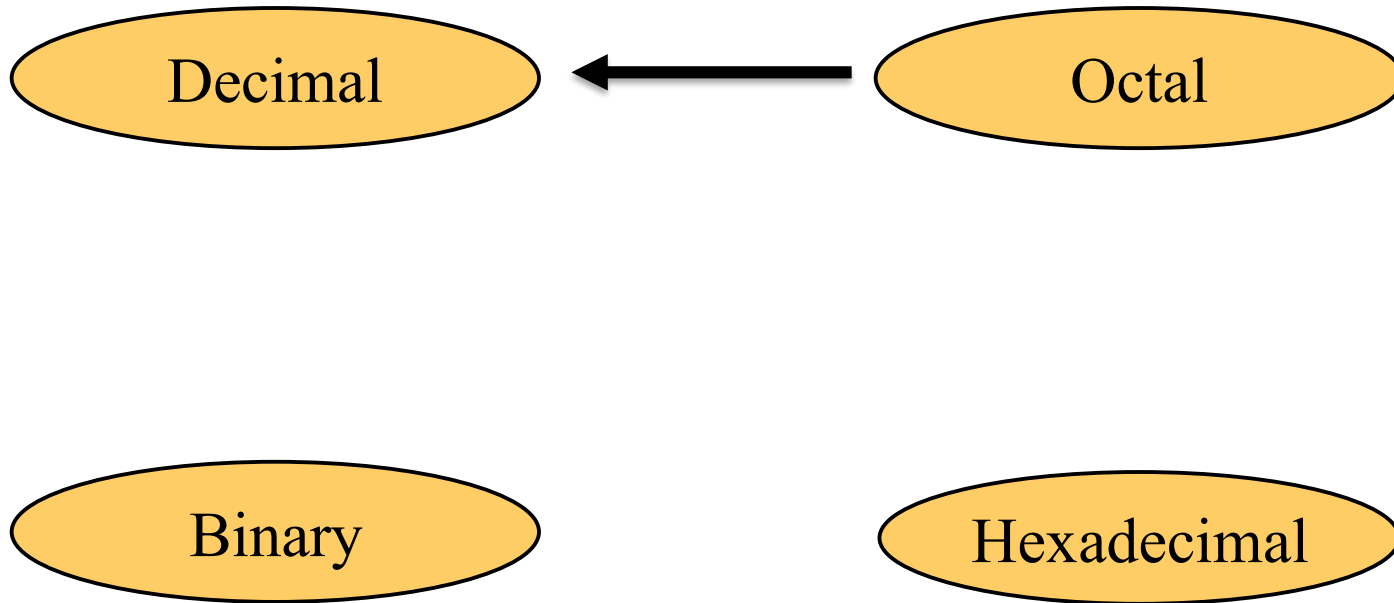
Example

Bit "0"

$101011_2 \Rightarrow$

1	x	2^0	=	1
1	x	2^1	=	2
0	x	2^2	=	0
1	x	2^3	=	8
0	x	2^4	=	0
1	x	2^5	=	32
				<hr/>
				43_{10}

Octal to Decimal



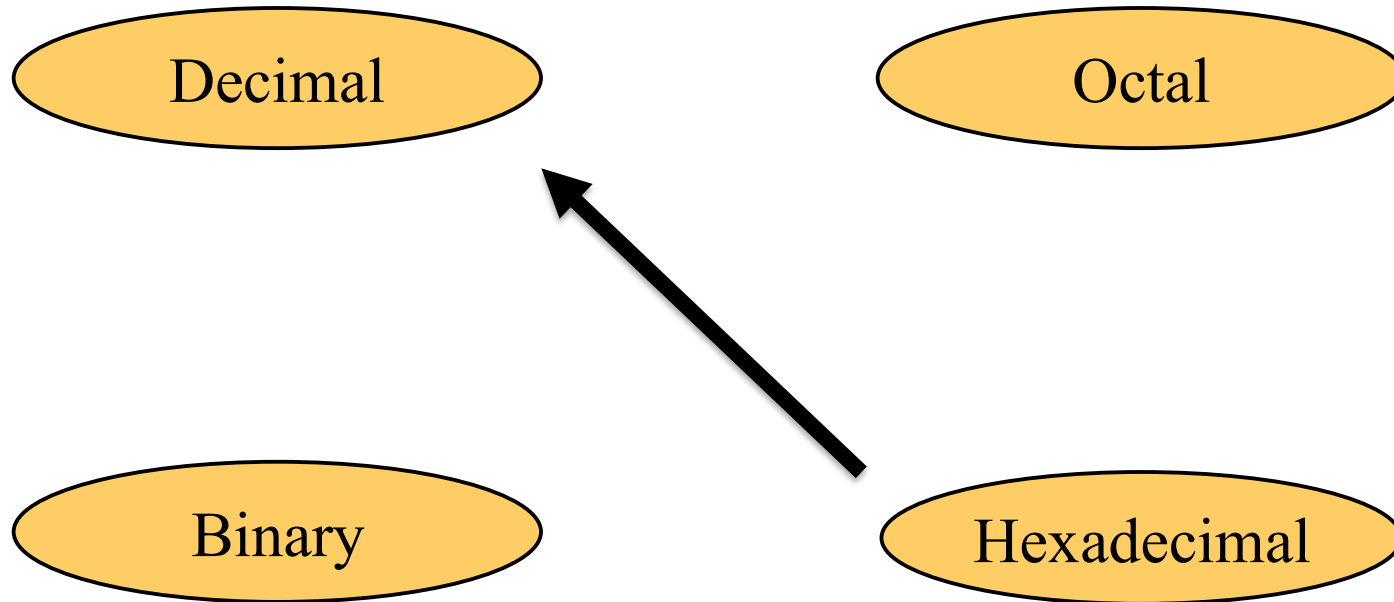
Octal to Decimal

- Technique
 - Multiply each bit by 8^n , where n is the “weight” of the bit
 - The weight is the position of the bit, starting from 0 on the right
 - Add the results

Example

$$724_8 \Rightarrow \begin{array}{r} 4 \times 8^0 = 4 \\ 2 \times 8^1 = 16 \\ 7 \times 8^2 = 448 \\ \hline 468_{10} \end{array}$$

Hexadecimal to Decimal



Hexadecimal to Decimal

- Technique
 - Multiply each bit by 16^n , where n is the “weight” of the bit
 - The weight is the position of the bit, starting from 0 on the right
 - Add the results

Example

$$\begin{array}{r} \mathbf{ABC}_{16} \Rightarrow \\ \mathbf{C} \times 16^0 = 12 \times 1 = 12 \\ \mathbf{B} \times 16^1 = 11 \times 16 = 176 \\ \mathbf{A} \times 16^2 = 10 \times 256 = 2560 \\ \hline \mathbf{2748}_{10} \end{array}$$

Decimal to Binary

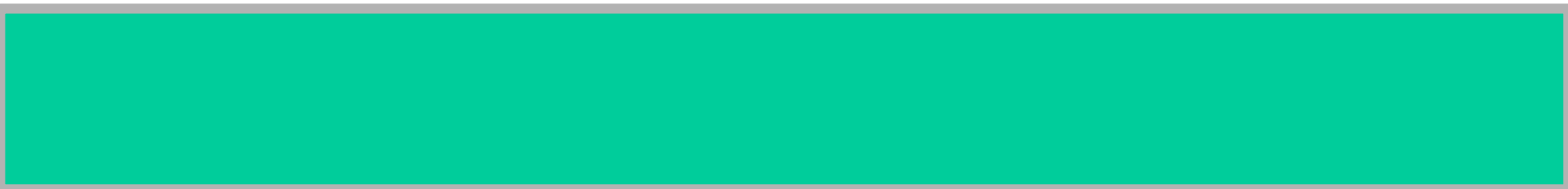
Decimal

Octal



Binary

Hexadecimal



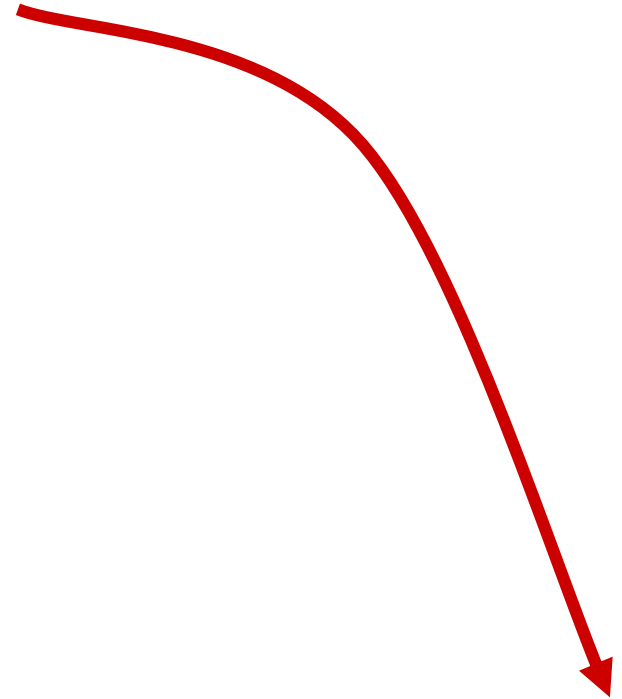
Decimal to Binary

- Technique
 - Divide by two, keep track of the remainder
 - First remainder is bit 0 (LSB, least-significant bit)
 - Second remainder is bit 1
 - Etc.

Example

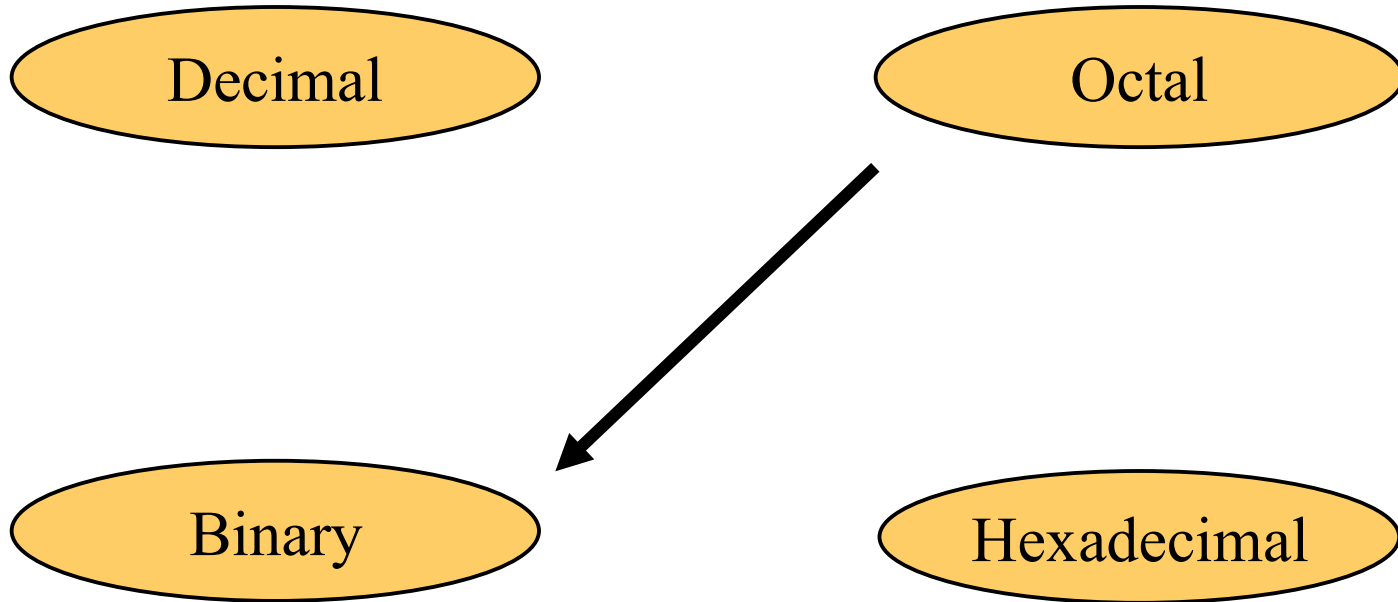
$$125_{10} = ?_2$$

2		125	
2		62	1
2		31	0
2		15	1
2		7	1
2		3	1
2		1	1
		0	1



$$125_{10} = 1111101_2$$

Octal to Binary

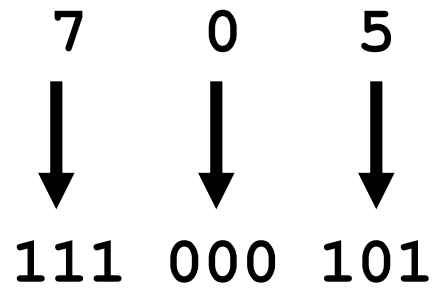


Octal to Binary

- Technique
 - Convert each octal digit to a 3-bit equivalent binary representation

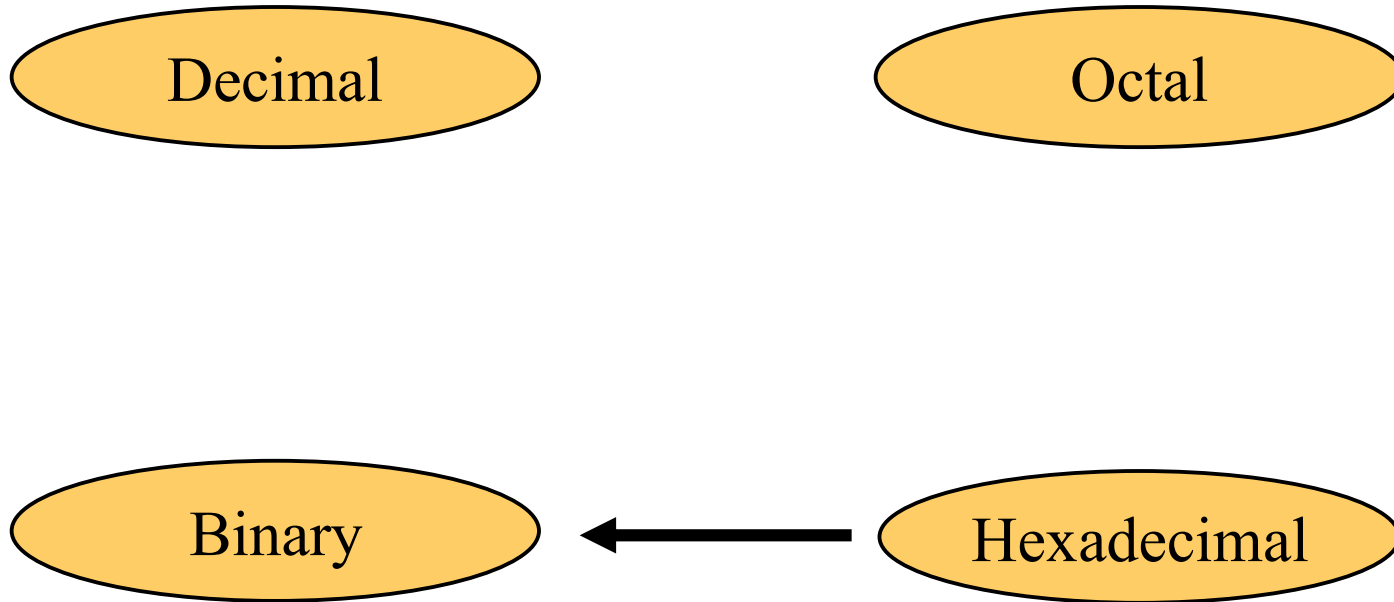
Example

$$705_8 = ?_2$$



$$705_8 = 111000101_2$$

Hexadecimal to Binary

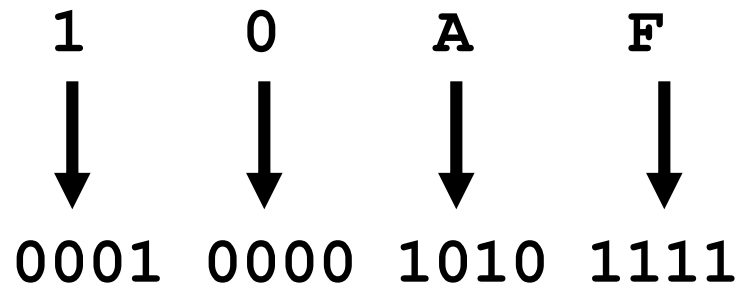


Hexadecimal to Binary

- Technique
 - Convert each hexadecimal digit to a 4-bit equivalent binary representation

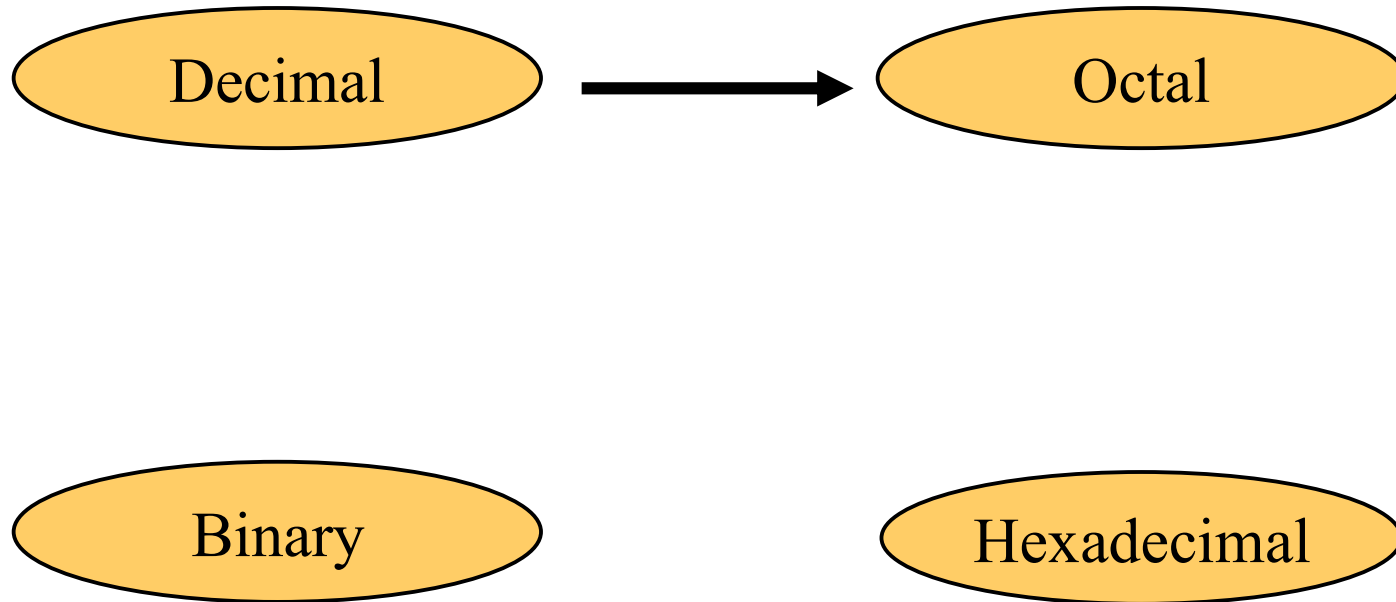
Example

$$10AF_{16} = ?_2$$



$$10AF_{16} = 0001000010101111_2$$

Decimal to Octal



Decimal to Octal

- Technique
 - Divide by 8
 - Keep track of the remainder

Example

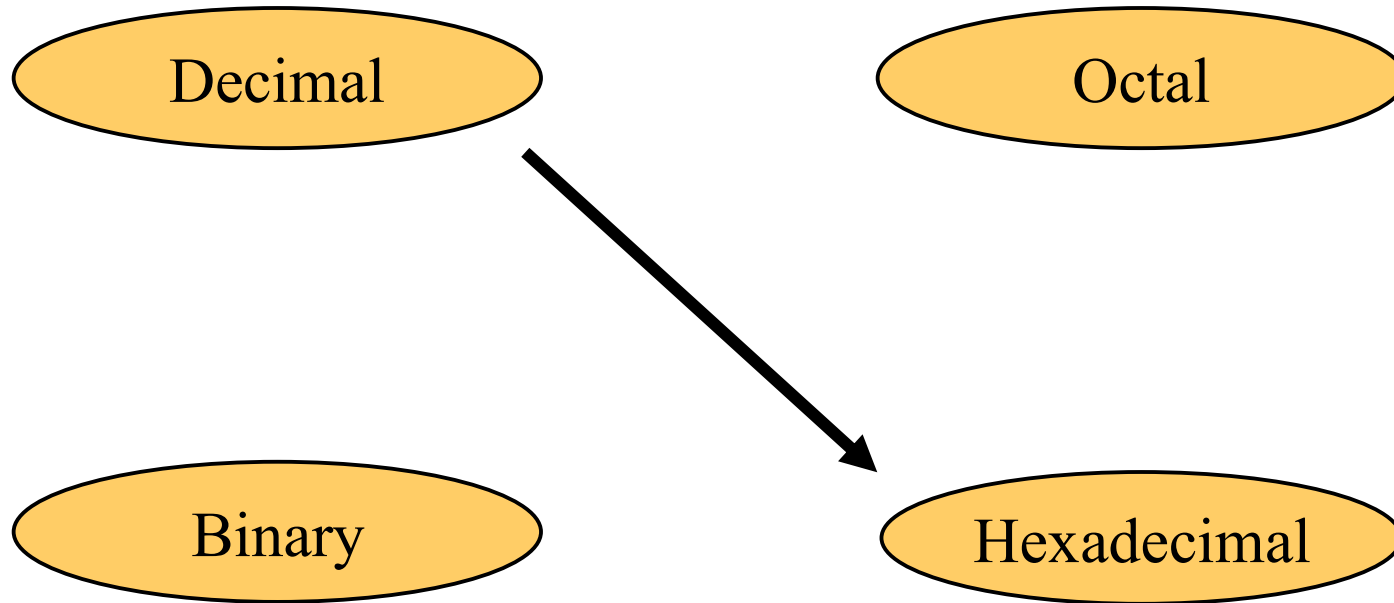
$$1234_{10} = ?_8$$

8		1234	
8		154	2
8		19	2
8		2	3
		0	2



$$1234_{10} = 2322_8$$

Decimal to Hexadecimal



Decimal to Hexadecimal

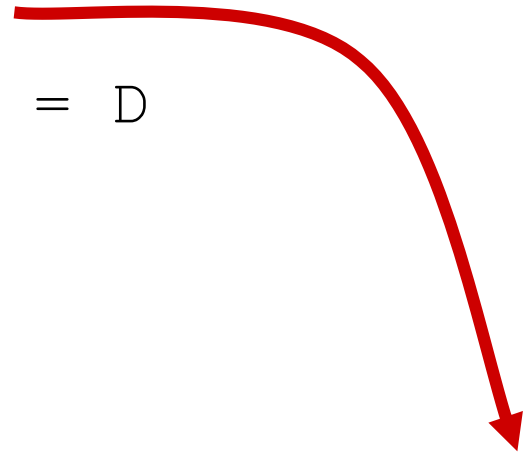
- Technique
 - Divide by 16
 - Keep track of the remainder

Example

$$1234_{10} = ?_{16}$$

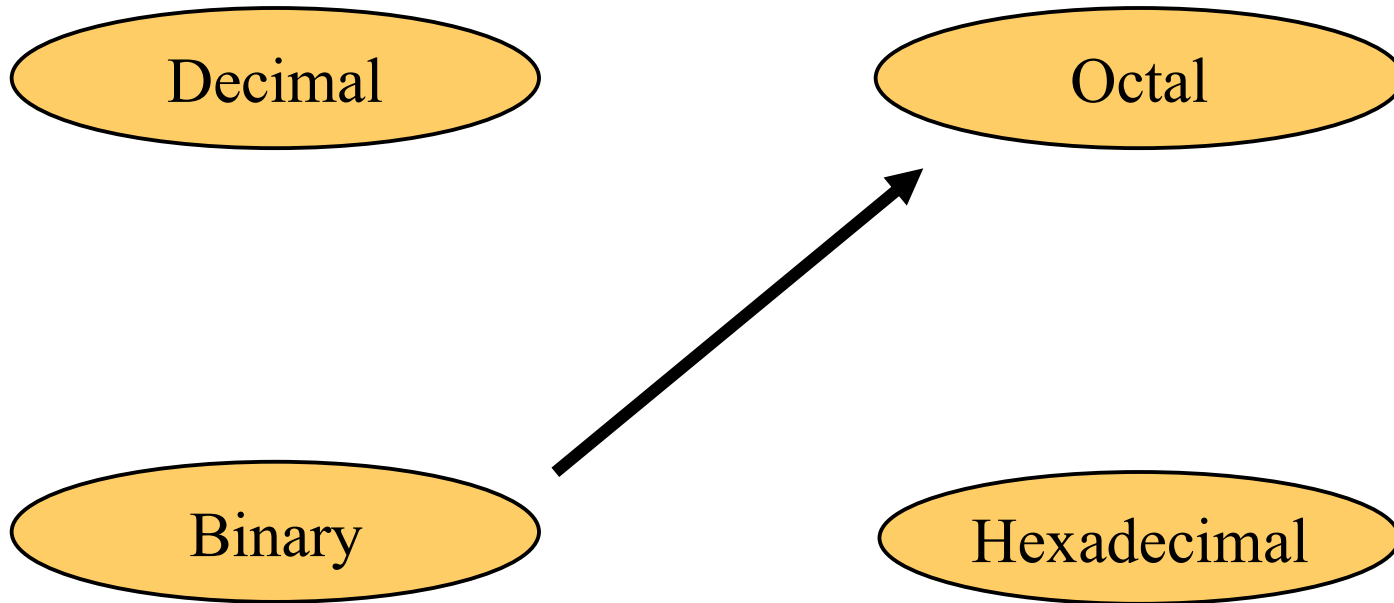
$$\begin{array}{r|l} 16 & \mathbf{1234} \\ \hline 16 & 77 \\ \hline 16 & 4 \\ \hline & 0 \end{array}$$

$$\begin{array}{l} 2 \\ 13 = \text{D} \\ 4 \end{array}$$



$$1234_{10} = 4\text{D}2_{16}$$

Binary to Octal

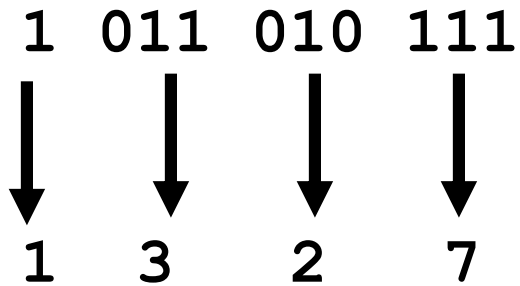


Binary to Octal

- Technique
 - Group bits in threes, starting on right
 - Convert to octal digits

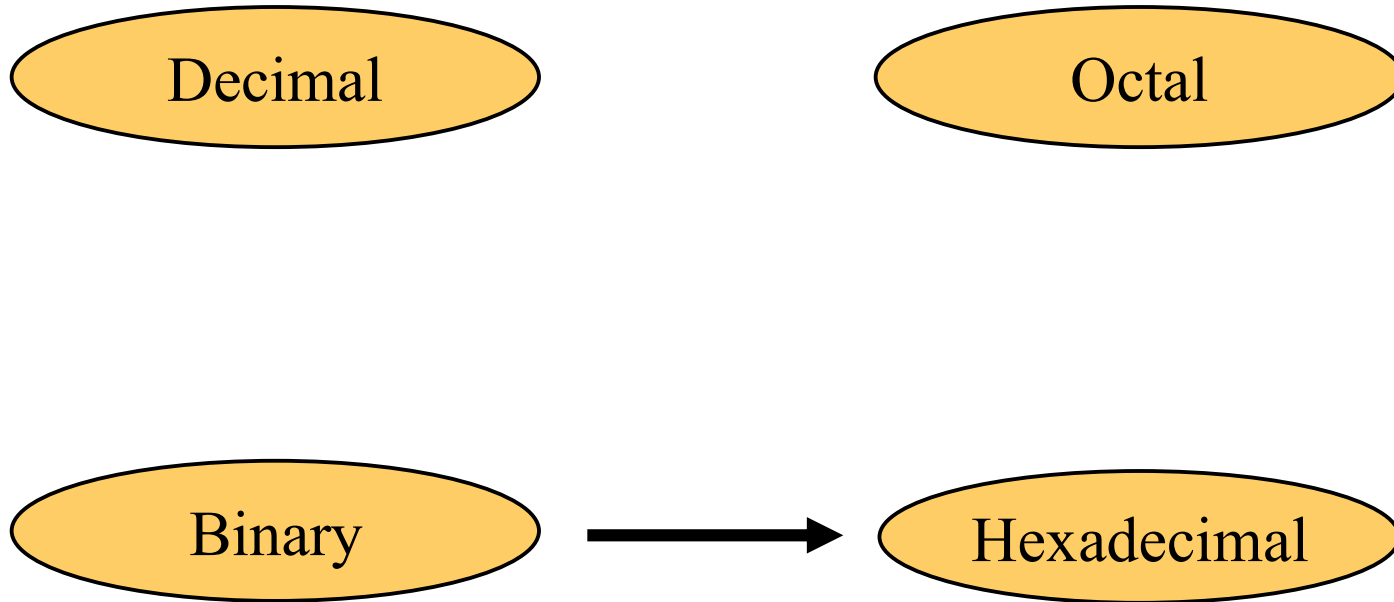
Example

$$1011010111_2 = ?_8$$



$$1011010111_2 = 1327_8$$

Binary to Hexadecimal

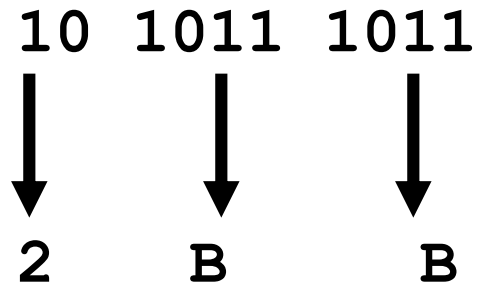


Binary to Hexadecimal

- Technique
 - Group bits in fours, starting on right
 - Convert to hexadecimal digits

Example

$$1010111011_2 = ?_{16}$$



$$1010111011_2 = 2BB_{16}$$

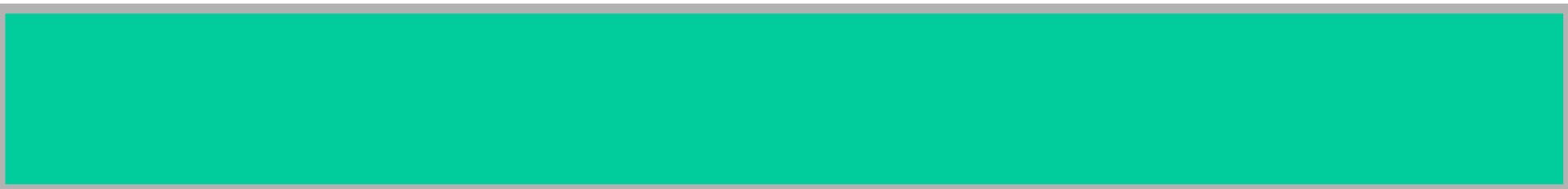
Octal to Hexadecimal

Decimal

Octal

Binary

Hexadecimal

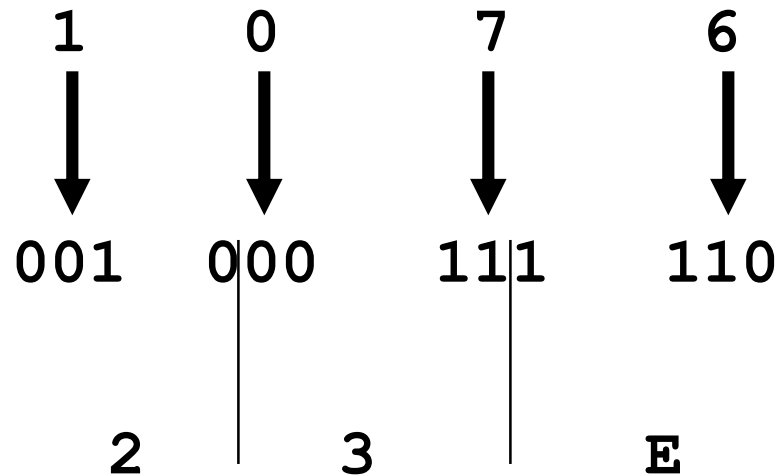


Octal to Hexadecimal

- Technique
 - Use binary as an intermediary

Example

$$1076_8 = ?_{16}$$



$$1076_8 = 23E_{16}$$

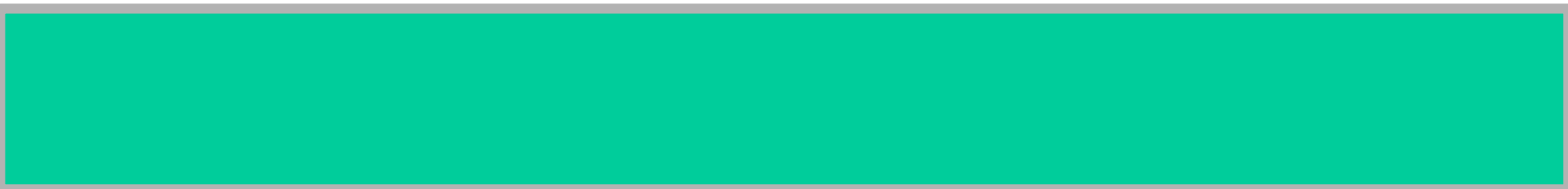
Hexadecimal to Octal

Decimal

Octal

Binary

Hexadecimal

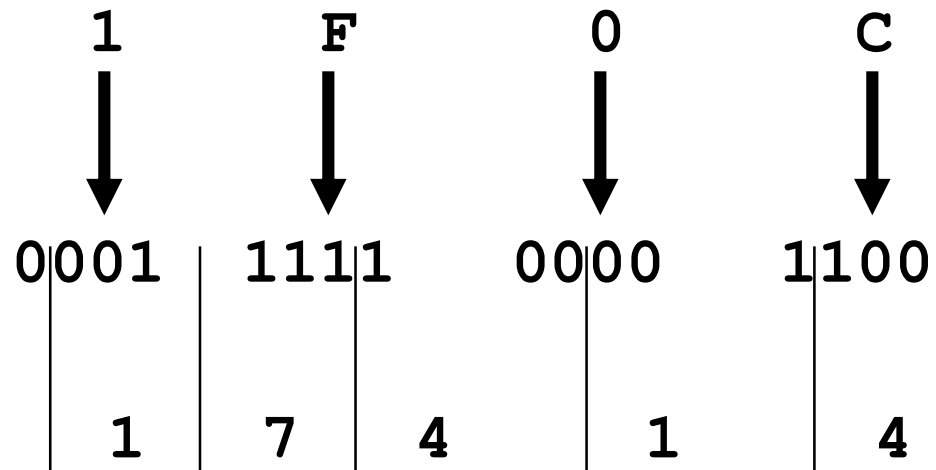


Hexadecimal to Octal

- Technique
 - Use binary as an intermediary

Example

$$1F0C_{16} = ?_8$$



$$1F0C_{16} = 17414_8$$

Exercise – Convert ...

Decimal	Binary	Octal	Hexa- decimal
33			
	1110101		
		703	
			1AF

Don't use a calculator!

Skip answer

Answer

Exercise – Convert ...

Answer

Decimal	Binary	Octal	Hexa- decimal
33	100001	41	21
117	1110101	165	75
451	111000011	703	1C3
431	110101111	657	1AF

