

COMPUTER ORGANIZATION & ARCHITECTURE

Course objective

This course will help the learner to understand the basic digital logic circuits, computer architecture, the memory system and Input / Output organization of a computer system

Syllabus

Course Code: CSE212

Semester: III

UNIT - I

15 Periods

Basics in Boolean logic and Combinational/Sequential Circuits: Digital Computers – Logic Gates - Boolean Algebra - Map Simplification - Combinational Circuits - Flip-Flop – Sequential Circuits.

Functional blocks of a computer: CPU, memory, input-output subsystems, control unit.

Data representation: Signed number representation, fixed and floating point representations, character representation.

Computer arithmetic: Addition and Subtraction of Signed Numbers - Design of Fast Adders - Multiplication of Unsigned Numbers – Multiplication of Signed Numbers - Fast Multiplication - Integer Division - Floating-Point Numbers and Operations, IEEE 754 format.

UNIT – II

15 Periods

Instruction set architecture of a CPU: Memory Locations and Addresses – Memory Operations - Instructions and Instruction Sequencing - Addressing Modes – Assembly Language - Stacks - Subroutines - Additional Instructions - CISC Instruction Sets.

Introduction to x86 architecture: The Intel IA-32 Architecture: Memory Organization - Register Structure - Addressing Modes – Instruction Set – Interrupts and Exceptions.

UNIT – III

15 Periods

CPU control unit design: Instruction Codes - Computer Registers - Computer Instructions - Timing and Control - Instruction Cycle - Memory-Reference Instructions - Input-Output and Interrupt - Design of Basic Computer - Design of Accumulator Logic.

Microprogrammed Control: Control Memory - Address Sequencing-Microprogram Example - Design of Control Unit.

Pipelining: Basic Concept - Pipeline Organization - Pipelining Issues – Data Dependencies - Memory Delays- Branch Delays - Superscalar Operation - Pipelining in CISC Processors.

Parallel Processors: Hardware Multithreading - Vector (SIMD) Processing -Cache Coherence

UNIT – IV

15 Periods

Memory system: Basic Concepts - Semiconductor RAM Memories - Read-only Memories - Direct Memory Access - Memory Hierarchy - Cache Memories - Performance Considerations - Virtual Memory - Memory Management Requirements - Secondary Storage.

Basic Input/Output: Accessing I/O Devices - Interrupts.

Input / Output Organization: Bus Structure - Bus Operation – Arbitration - Interface Circuits - Interconnection Standards

TEXTBOOKS

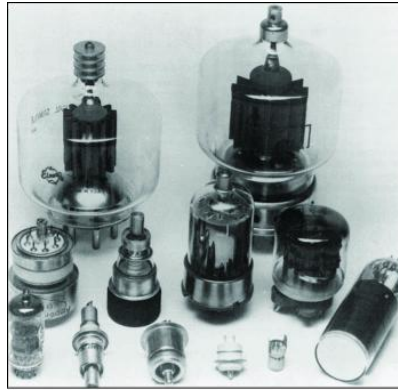
1. M. M. Mano, Computer System Architecture, Prentice Hall of India, 3rd Edition, 2007.
2. David A. Patterson and John L. Hennessy, Computer Organization and Design: The Hardware/Software Interface, Morgan Kaufmann publishers, 5th Edition, 2014.
3. Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Naraig Manjikian, Computer Organization and Embedded Systems, McGraw Hill, 5th Edition, 2012.

REFERENCES

1. John P. Hayes, Computer Architecture and Organization, McGraw-Hill, 2nd Edition, 1998.
2. William Stallings, Computer Organization and Architecture: Designing for Performance, Pearson India, 11th Edition, 2019.

Evolution of Electronic Devices

Vacuum
Tubes



(a)

Discrete
Transistors



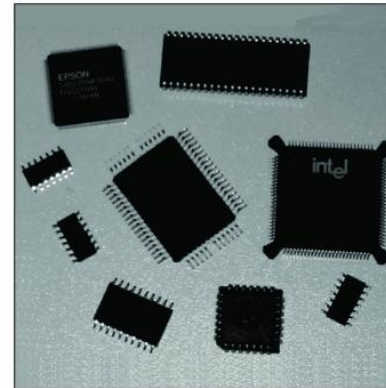
(b)

SSI and MSI
Integrated
Circuits



(c)

VLSI
Surface-Mount
Circuits



(d)

System Details

The screenshot displays the Windows Settings application, specifically the 'About' section. The left sidebar shows the 'Settings' menu with 'Home' at the top and a search bar. Below the search bar, the 'System' category is selected, with a list of sub-options: Display, Sound, Notifications & actions, Focus assist, Power & sleep, Battery, Storage, Tablet, Multitasking, Projecting to this PC, Shared experiences, Clipboard, and Remote Desktop. The main content area is titled 'About' and features a message: 'Your PC is monitored and protected. See details in Windows Security'. Below this, the 'Device specifications' section lists hardware details for 'G3 3579': Device name (DESKTOP-7LTOMIN), Processor (Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz), Installed RAM (8.00 GB), Device ID (74BBB383-8398-4607-BF93-66A682DFDC76), Product ID (00325-81129-40223-AAOEM), System type (64-bit operating system, x64-based processor), and Pen and touch (No pen or touch input is available for this display). Buttons for 'Copy' and 'Rename this PC' are provided. The 'Windows specifications' section lists: Edition (Windows 10 Home), Version (21H2), Installed on (24-Mar-21), OS build (19044.1889), and Experience (Windows Feature Experience Pack 120.2212.4180.0). On the right, a note states: 'This page has a few new settings. Some settings from Control Panel have moved here, and you can copy your PC info so it's easier to share.' Below this are links for 'Related settings': BitLocker settings, Device Manager, Remote desktop, System protection, Advanced system settings, and Rename this PC (advanced). At the bottom, a 'Help from the web' section includes links for 'Finding out how many cores my processor has', 'Checking multiple Languages support', 'Get help', and 'Give feedback'. The taskbar at the bottom shows the Start button, search bar, task view button, and several pinned applications (Chrome, File Explorer, PowerPoint, etc.). The system tray on the right shows the weather (34°C Haze), time (11:56), and date (22-Aug-22).

Settings

Home

Find a setting

System

- Display
- Sound
- Notifications & actions
- Focus assist
- Power & sleep
- Battery
- Storage
- Tablet
- Multitasking
- Projecting to this PC
- Shared experiences
- Clipboard
- Remote Desktop

About

Your PC is monitored and protected.

[See details in Windows Security](#)

Device specifications

G3 3579

Device name	DESKTOP-7LTOMIN
Processor	Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz 2.21 GHz
Installed RAM	8.00 GB (7.86 GB usable)
Device ID	74BBB383-8398-4607-BF93-66A682DFDC76
Product ID	00325-81129-40223-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Copy

Rename this PC

Windows specifications

Edition	Windows 10 Home
Version	21H2
Installed on	24-Mar-21
OS build	19044.1889
Experience	Windows Feature Experience Pack 120.2212.4180.0

This page has a few new settings

Some settings from Control Panel have moved here, and you can copy your PC info so it's easier to share.

Related settings

- [BitLocker settings](#)
- [Device Manager](#)
- [Remote desktop](#)
- [System protection](#)
- [Advanced system settings](#)
- [Rename this PC \(advanced\)](#)

Help from the web

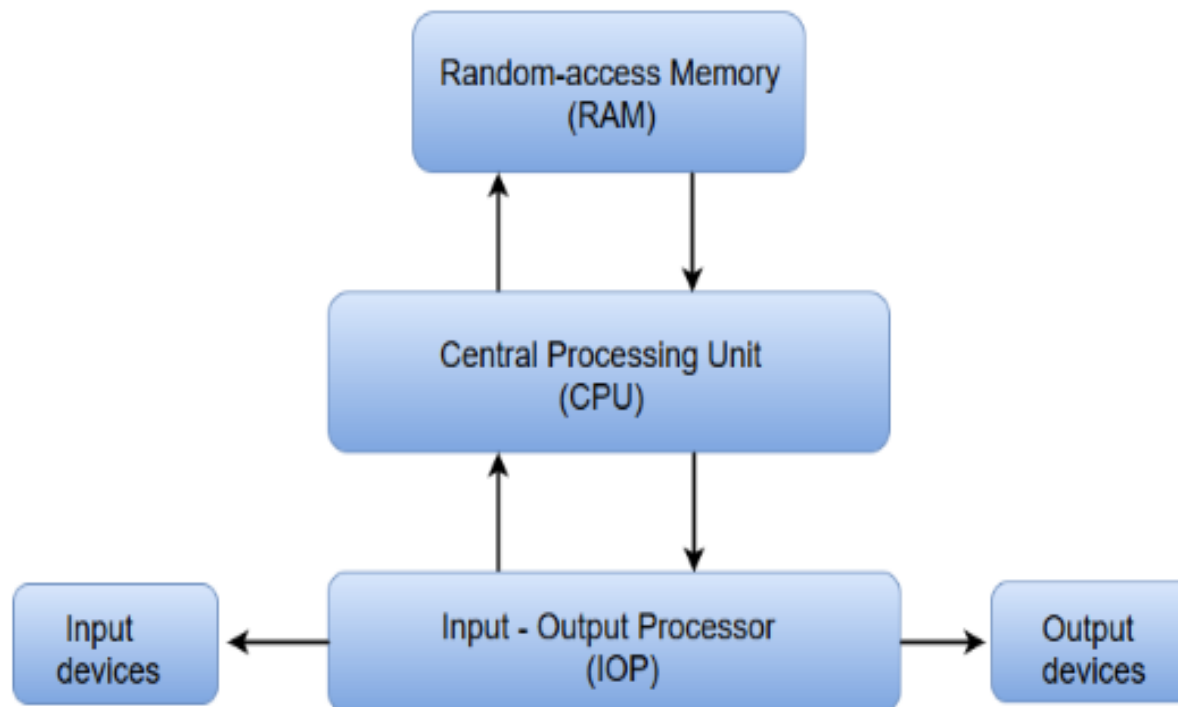
- [Finding out how many cores my processor has](#)
- [Checking multiple Languages support](#)
- [Get help](#)
- [Give feedback](#)

Type here to search

34°C Haze 11:56 22-Aug-22

UNIT - I




Block diagram of Digital Computers






Logic gates

- Binary information is represented in digital computers by physical quantities called signals .
- Electrical signals such as voltages exist throughout the computer in either one of two recognizable states.
- The two states represent a binary variable that can be equal to 1 or 0.
- The manipulation of binary information is done by logic circuits called gates .

- Gates are blocks of hardware that produce signals of binary 1 or 0 when input logic requirements are satisfied.
- A variety of logic gates are commonly used in digital computer systems.
- Each gate has a distinct graphic symbol and its operation can be described by means of an algebraic expression.
- The input-output relationship of the binary variables for each gate can be represented in tabular form by a truth table.

Name	Graphic symbol	Algebraic function	Truth table															
AND		$x = A \cdot B$ or $x = AB$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	x	0	0	0	0	1	0	1	0	0	1	1	1
A	B	x																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$x = A + B$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	x	0	0	0	0	1	1	1	0	1	1	1	1
A	B	x																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
Inverter		$x = A'$	<table><tr><th>A</th><th>x</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	x	0	1	1	0									
A	x																	
0	1																	
1	0																	

Buffer	 $A \rightarrow x \quad x = A$	<table><tr><th>A</th><th>x</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	x	0	0	1	1									
A	x																
0	0																
1	1																
NAND	 $A \quad B \rightarrow x \quad x = (AB)'$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	x	0	0	1	0	1	1	1	0	1	1	1	0
A	B	x															
0	0	1															
0	1	1															
1	0	1															
1	1	0															
NOR	 $A \quad B \rightarrow x \quad x = (A + B)'$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	x	0	0	1	0	1	0	1	0	0	1	1	0
A	B	x															
0	0	1															
0	1	0															
1	0	0															
1	1	0															

**Exclusive-OR
(XOR)**



$$x = A \oplus B$$

or

$$x = A'B + AB'$$

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

**Exclusive-NOR
or equivalence**



$$x = (A \oplus B)'$$

or

$$x = A'B' + AB$$

A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Boolean algebra:

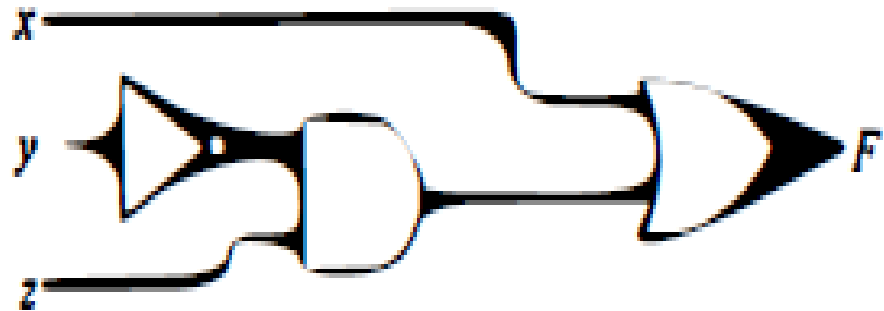
The purpose of Boolean algebra is to facilitate the analysis and design of digital circuits. It provides a convenient tool to:

1. Express in algebraic form a truth table relationship between binary variables.
2. Express in algebraic form the input-output relationship of logic diagrams.
3. Find simpler circuits for the same function.

Truth table and logic diagram for $F = x + y'z$.

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(a) Truth table



(b) Logic diagram

Basic Identities of Boolean Algebra

$$(1) \ x + 0 = x$$

$$(3) \ x + 1 = 1$$

$$(5) \ x + x = x$$

$$(7) \ x + x' = 1$$

$$(9) \ x + y = y + x$$

$$(11) \ x + (y + z) = (x + y) + z$$

$$(13) \ x(y + z) = xy + xz$$

$$(15) \ (x + y)' = x'y'$$

$$(17) \ (x')' = x$$

$$(2) \ x \cdot 0 = 0$$

$$(4) \ x \cdot 1 = x$$

$$(6) \ x \cdot x = x$$

$$(8) \ x \cdot x' = 0$$

$$(10) \ xy = yx$$

$$(12) \ x(yz) = (xy)z$$

$$(14) \ x + yx = (x + y)(x + z)$$

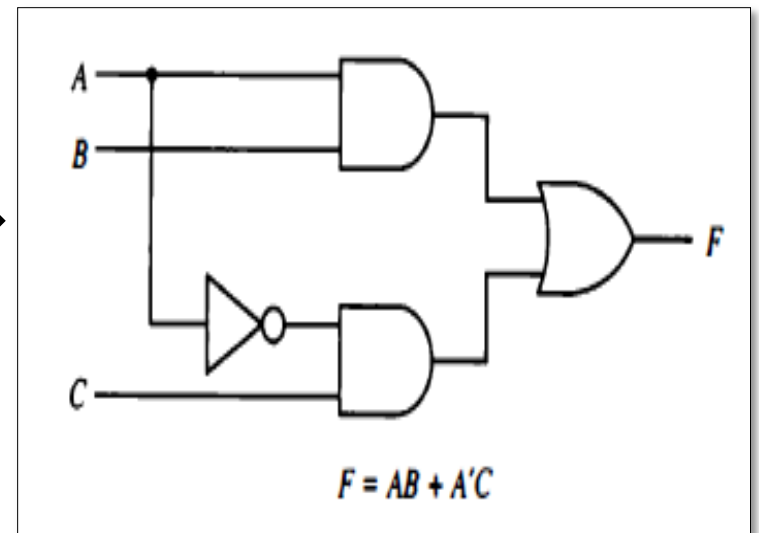
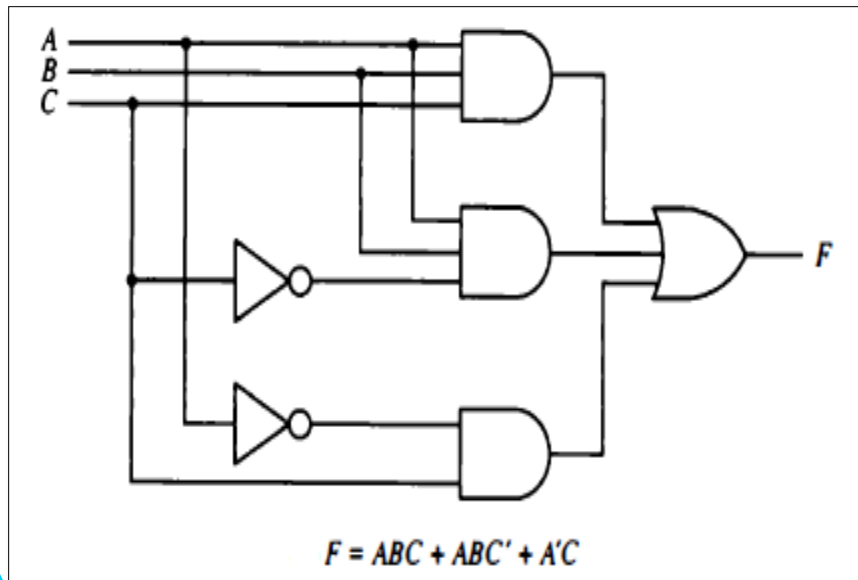
$$(16) \ (xy)' = x' + y'$$

Boolean algebra manipulation is used to simplify digital circuits.

Example

$$F = ABC + A BC' + A 'C = AB (C + C ') + A 'C$$

$$= AB + A 'C$$



Simplify the following expressions using Boolean algebra.

$$\mathbf{AB + A(CD + CD')}$$

$$\begin{aligned} AB + A(CD + CD') &= AB + AC(D + D') \\ &= AB + AC(1) && [D + D' = 1] \\ &= A(B + C) \\ &= A \end{aligned}$$

Using DeMorgan's theorem , show that

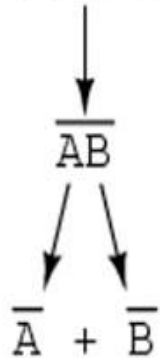
$$\mathbf{(A + B)'(A' + B')' = 0}$$

$$\begin{aligned} (A + B)'(A' + B')' &= (A' B')(AB) && [AA' = 0] \\ &= 0 \end{aligned}$$

DeMorgan's Theorems

$$\overline{A B} = \overline{A} + \overline{B}$$

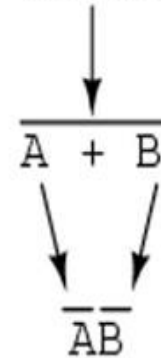
break!



NAND to Negative-OR

$$\overline{A + B} = \overline{A} \overline{B}$$

break!



NOR to Negative-AND

Karnaugh map (k-map)

➤ Complexity of digital logic is directly related to the complexity of the algebraic expression. So, we need to simplify the boolean function

1. Boolean laws

2. K-Map

➤ It is a digram made up of squares, with each square representing one minterm of the function that is to be minimized.

➤ A pictorial arrangement of the truth table which allows an easy interpretation for choosing the minimum number of terms needed to express the function algebraically.

➤ Each combination of the variables in a truth table is called a minterm.

➤ Function of n variables will have 2^n minterms

Sum-of-Products Simplification:

- The map is a diagram made up of squares, with each square representing one minterm.
- The squares corresponding to minterms that produce 1 for the function are marked by a 1 and the others are marked by a 0 or are left empty.

		B	
		0	1
A	0	0	1
	1	2	3

Two-variable map

		B			
		00	01	11	10
A	0	0	1	3	2
	1	4	5	7	6

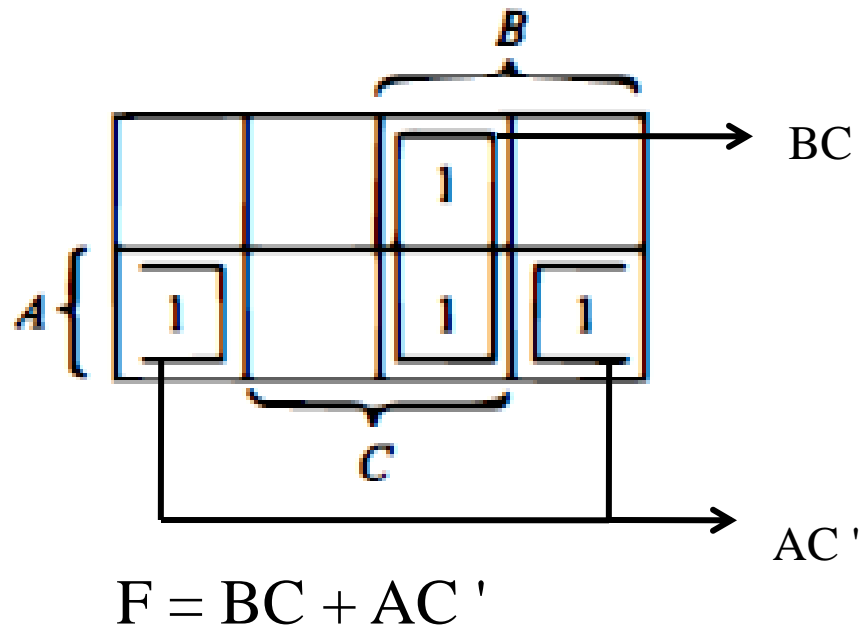
Three-variable map

		CD			
		00	01	11	10
AB	00	0	1	3	2
	01	4	5	7	6
	11	12	13	15	14
	10	8	9	11	10

Four-variable map

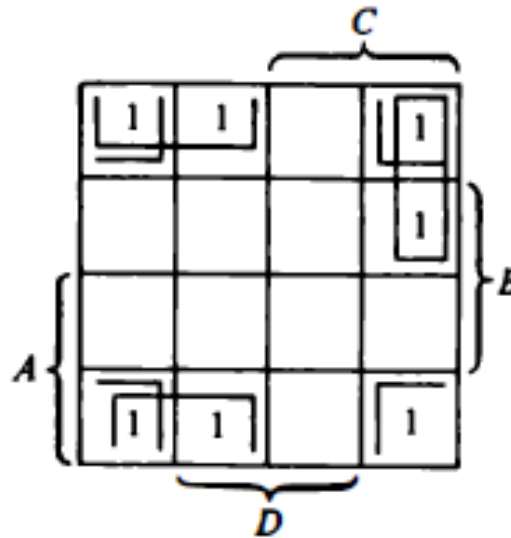
Example: 1

$$F(A, B, C) = \Sigma (3, 4, 6, 7)$$



Example: 2

$$F(A, B, C, D) = \Sigma (0, 1, 2, 6, 8, 9, 10)$$



$$F = B' D' + B' C' + A' C D'$$

Product-of-Sums Simplification:

- The Boolean expressions derived from the maps in the preceding examples were expressed in sum-of-products form.
- The product terms are AND terms and the sum denotes the ORing of these terms.
- It is sometimes convenient to obtain the algebraic expression for the function in a product-of-sums form.

The procedure for obtaining a product-of-sums expression follows from the basic properties of Boolean algebra.

- The 1's in the map represent the minterms that produce 1 for the function.
- The squares not marked by 1 represent the min terms that produce 0 for the function.
- If we mark the empty squares with 0' s and combine them into groups of adjacent squares, we obtain the complement of the function, f' .
- Taking the complement of F' produces an expression for F in product-of-sums form.

$$F(A, B, C, D) = \Sigma (0, 1, 2, 5, 8, 9, 10)$$

		C		
	1	1	0	1
	0	1	0	0
A	0	0	0	0
	1	1	0	1
		D		

$$F' = AB + CD + BD'$$

$$F = (A' + B')(C' + D')(B' + D)$$

Don't-care Conditions:

Min terms that may produce either 0 or 1 for the function are said to be don't-care conditions and are marked with an x in the map.

$$F(A, B, C) = \Sigma (0, 2, 6)$$

$$d(A, B, C) = \Sigma (1, 3, 5)$$

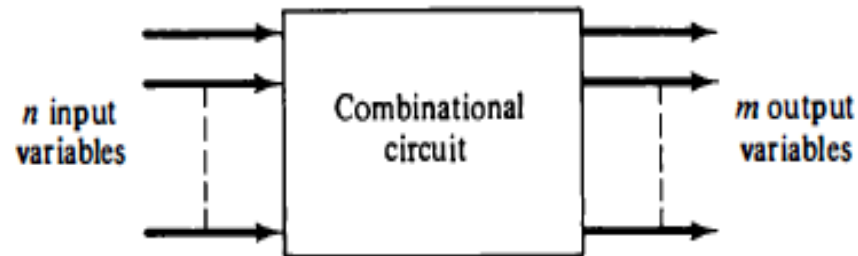
A 2x4 Karnaugh map for variables A, B, and C. The vertical axis is labeled A with a bracket on the left, and the horizontal axis is labeled B with a bracket on top. The columns are labeled C with a bracket below. The cells contain the following values: Row A=0: (B=0, C=0) is 0, (B=0, C=1) is x, (B=1, C=0) is 0, (B=1, C=1) is 1. Row A=1: (B=0, C=0) is 1, (B=0, C=1) is x, (B=1, C=0) is x, (B=1, C=1) is 1. A group of four cells (A=0, B=1, C=0, C=1) is enclosed in a box.

A {		B			
		C			
0		0	1	0	1
1		0	1	x	x

$$F = A' + B C'$$

Combinational Circuits:

- A combinational circuit is a connected arrangement of logic gates with a set of inputs and outputs.



Block diagram of a combinational circuit.

- The n binary input variables come from an external source, the m binary output variables go to an external destination, and in between there is an interconnection of logic gates

The design of combinational circuits involves the following steps:

- The problem is stated.
- The input and output variables are assigned letter symbols.
- The truth table that defines the relationship between inputs and outputs is derived.
- The simplified Boolean functions for each output are obtained.
- The logic diagram is drawn.

Half adder:

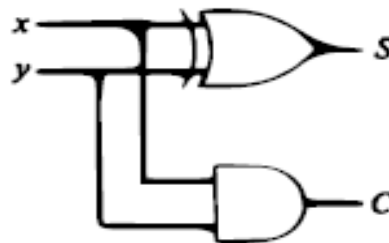
A combinational circuit that performs the arithmetic addition of two bits is called a half-adder.

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth table

$$S = x'y + xy' = x \oplus y$$

$$C = xy$$



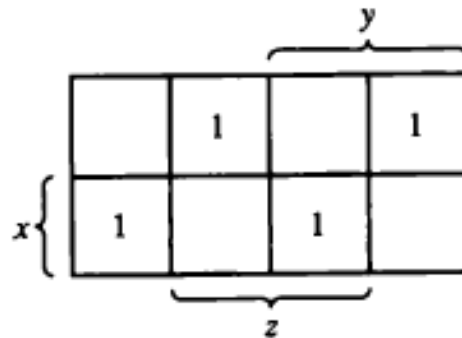
Logic diagram

Full-Adder:

A full-adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs.

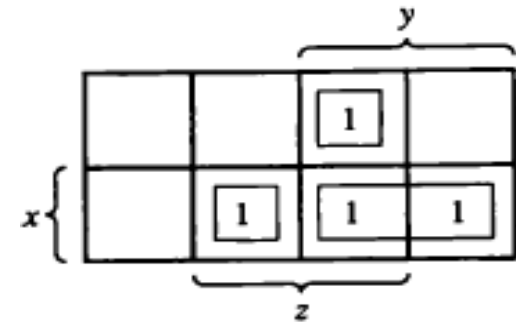
Truth Table for Full-Adder

Inputs			Outputs	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$= x \oplus y \oplus z$$

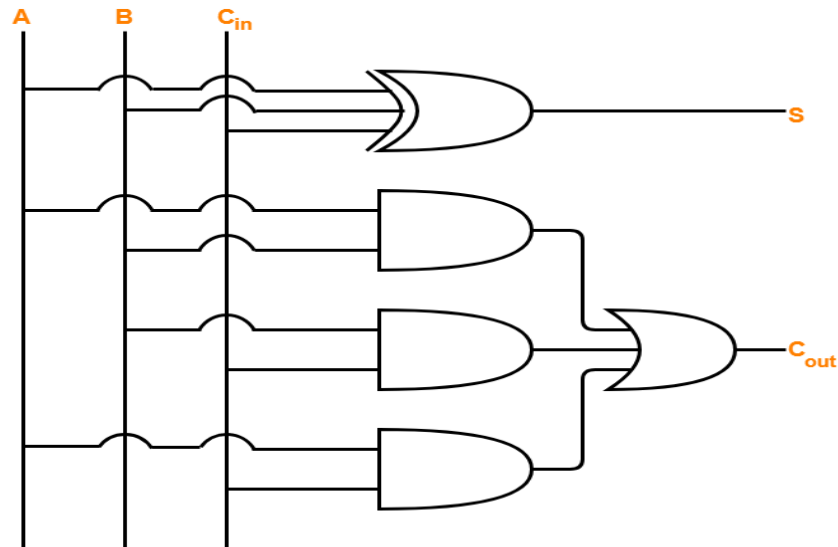


$$C = xy + xz + yz$$

$$= xy + (x'y + xy')z$$



Block diagram



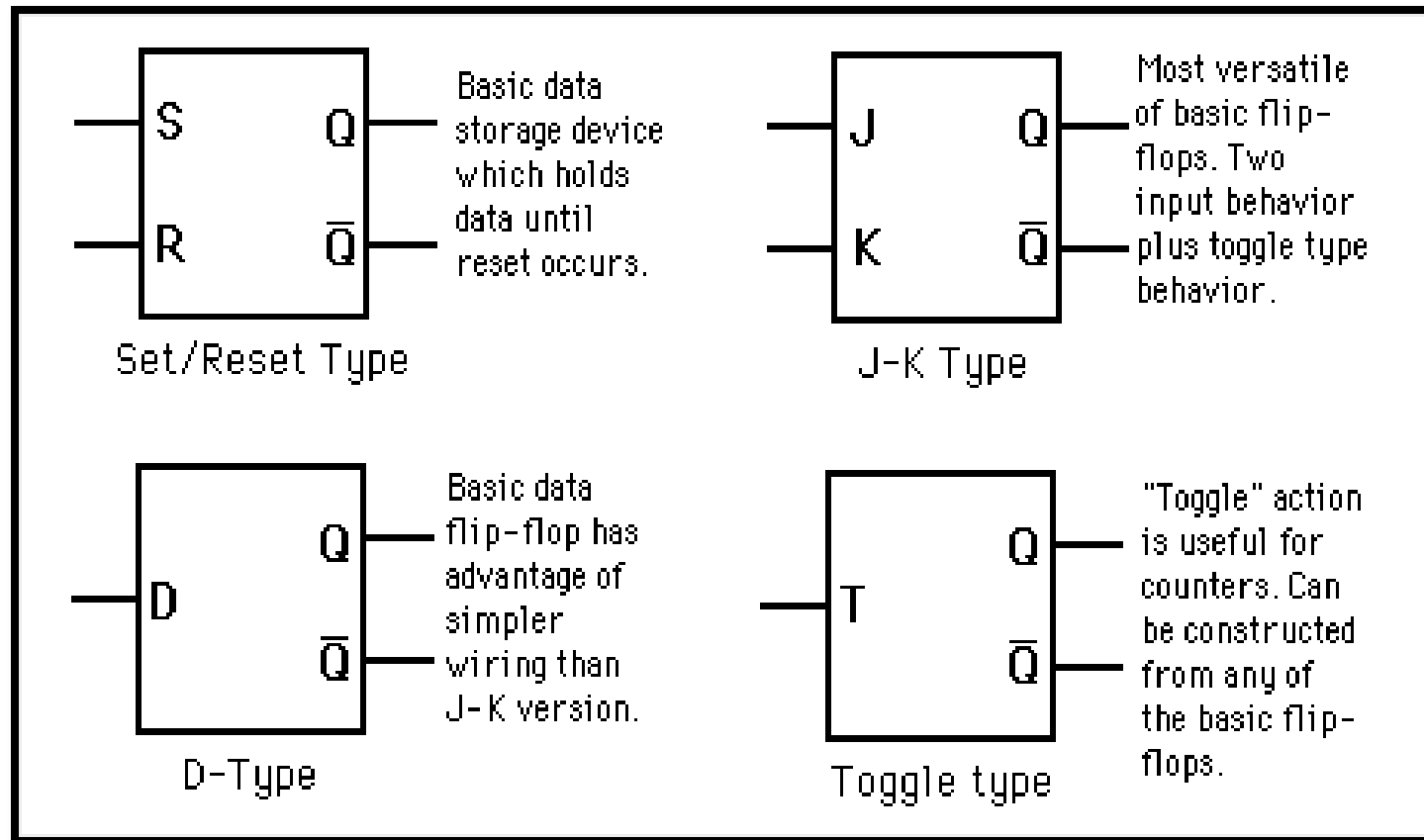
Full Adder Logic Diagram

Logic diagram

Flip-Flops:

- The storage elements employed in clocked sequential circuits are called flip-flops.
- A flip-flop is a binary cell capable of storing one bit of information.
- It has two outputs, one for the normal value and one for the complement value of the bit stored in it.
- A flip-flop maintains a binary state until directed by a clock pulse to switch states.

The types of flip-flops are



Truth Table : Defines input and output relation

Characteristics Table: It defines the next state of flip-flop in terms of flip-flop input and current state.

Excitation Table: It defines the flip-flop input variable as function of the current state and next state.

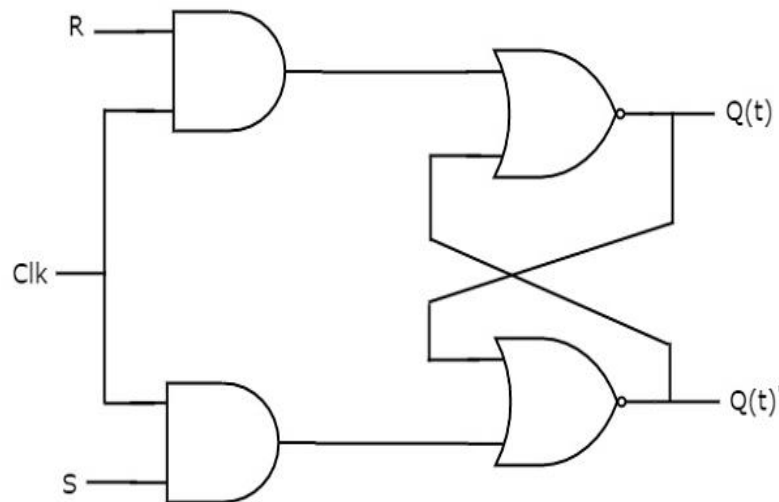
SR Flip-Flop:

State Table : Defines input and output relation

Characteristics Table: It defines the next state of flip-flop in terms of flip-flop input and current state.

Excitation Table: It defines the flip-flop input variable as function of the current state and next state.

circuit diagram



State table

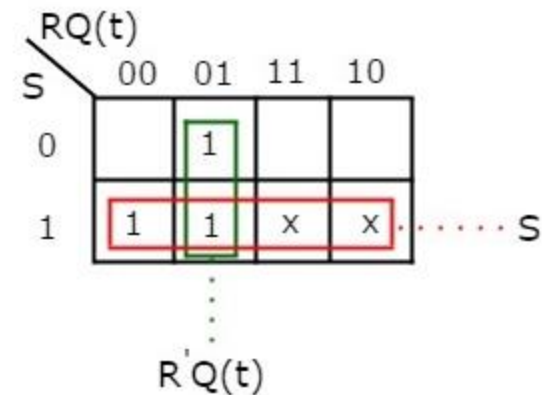
<i>S</i>	<i>R</i>	<i>Q</i> (<i>t</i> + 1)	
0	0	<i>Q</i> (<i>t</i>)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	?	Indeterminate

characteristic table

Present Inputs		Present State	Next State
S	R	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	x
1	1	1	x

Excitation table

$Q(t)$	$Q(t+1)$	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

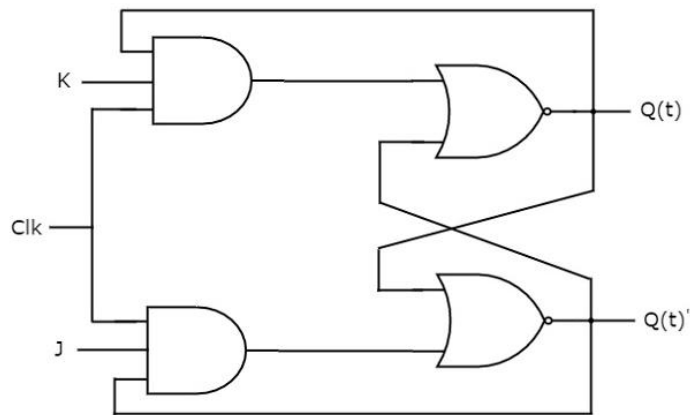


Characteristic equation

$$Q(t+1) = S + R'Q(t)$$

JK Flip-Flop:

circuit diagram



State table

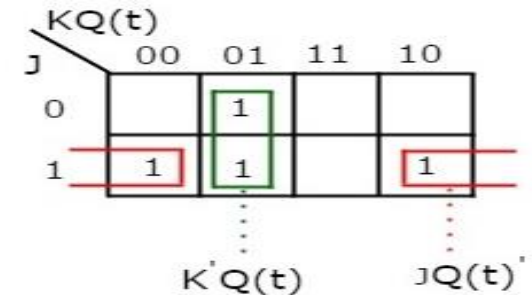
<i>J</i>	<i>K</i>	<i>Q</i> (<i>t</i> + 1)	
0	0	<i>Q</i> (<i>t</i>)	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	<i>Q'</i> (<i>t</i>)	Complement

characteristic table

Present Inputs		Present State	Next State
J	K	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Excitation table

$Q(t)$	$Q(t+1)$	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

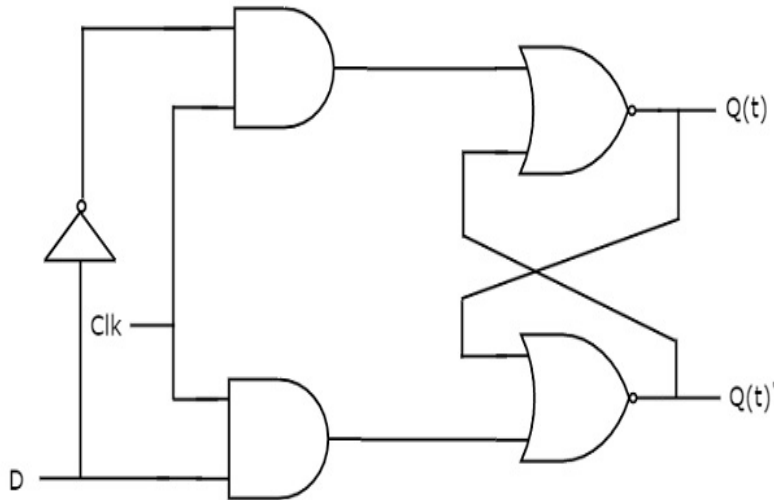


Characteristic equation

$$Q(t+1) = JQ(t)' + K'Q(t)$$

D Flip-Flop:

circuit diagram



State table

D	$Q(t+1)$
0	0
1	1

Clear to 0
Set to 1

Excitation table

$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

Characteristic table

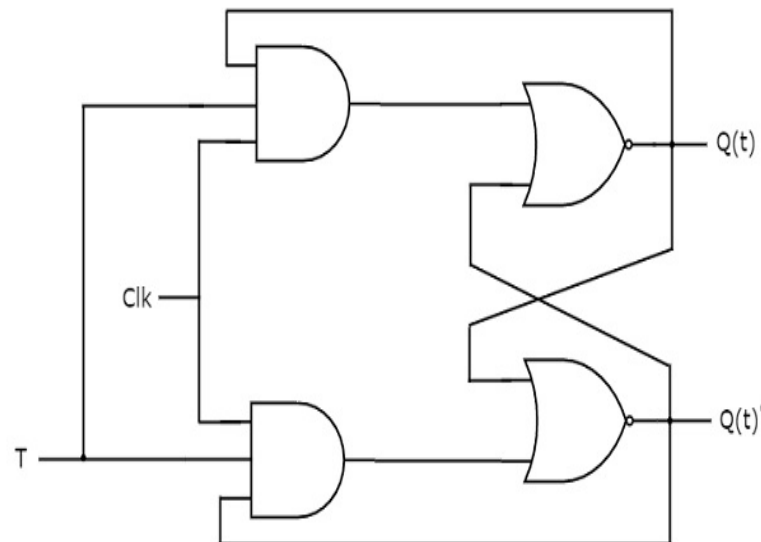
D	$Q(t)$	$Q(t+1)$
0	0	0
0	1	0
1	0	1
1	1	1

Characteristic equation

$$Q(t+1) = D$$

T Flip-Flop:

circuit diagram



State table

T	$Q(t+1)$
0	$Q(t)$ No change
1	$Q'(t)$ Complement

characteristic table

Inputs	Present State	Next State
T	Q t	Q t + 1
0	0	0
0	1	1
1	0	1
1	1	0

Excitation table

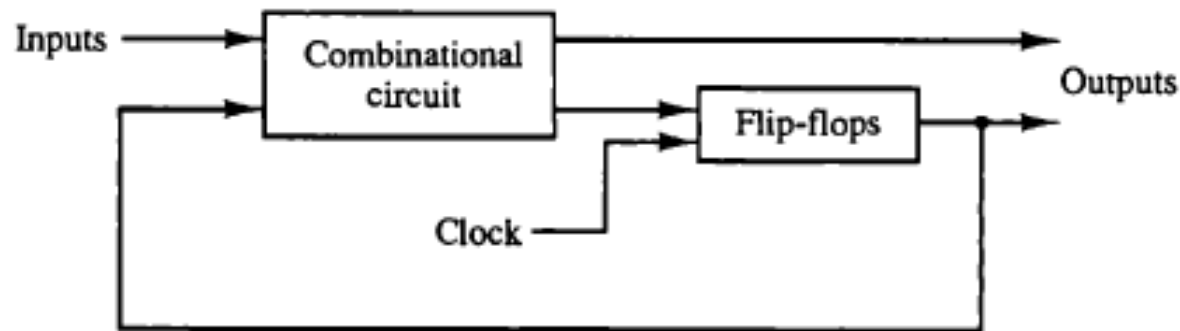
Q(t)	Q(t + 1)	T
0	0	0
0	1	1
1	0	1
1	1	0

Characteristic equation

$$Q(t + 1) = T'Q(t) + TQ(t)'$$

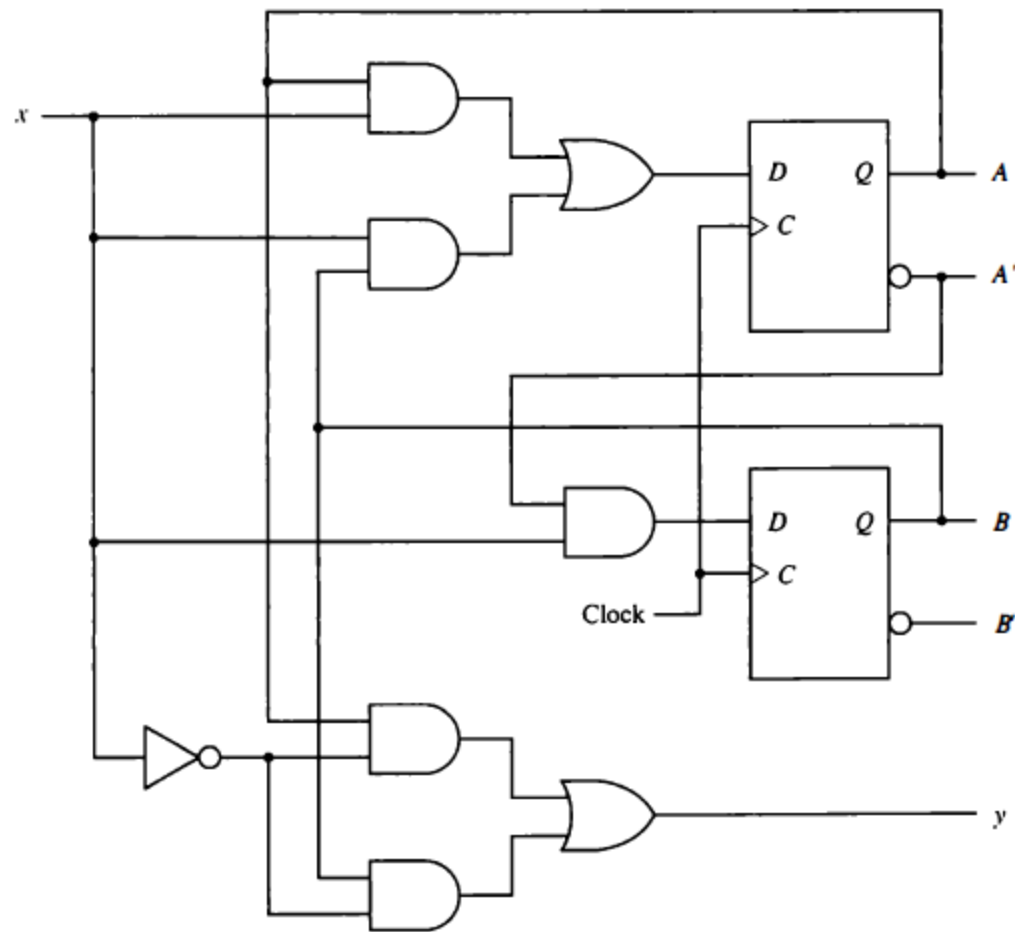
Sequential Circuits:

- A sequential circuit is an interconnection of flip-flops and gates.
- The gates by themselves constitute a combinational circuit, but when included with the flip-flops, the overall circuit is classified as a sequential circuit.



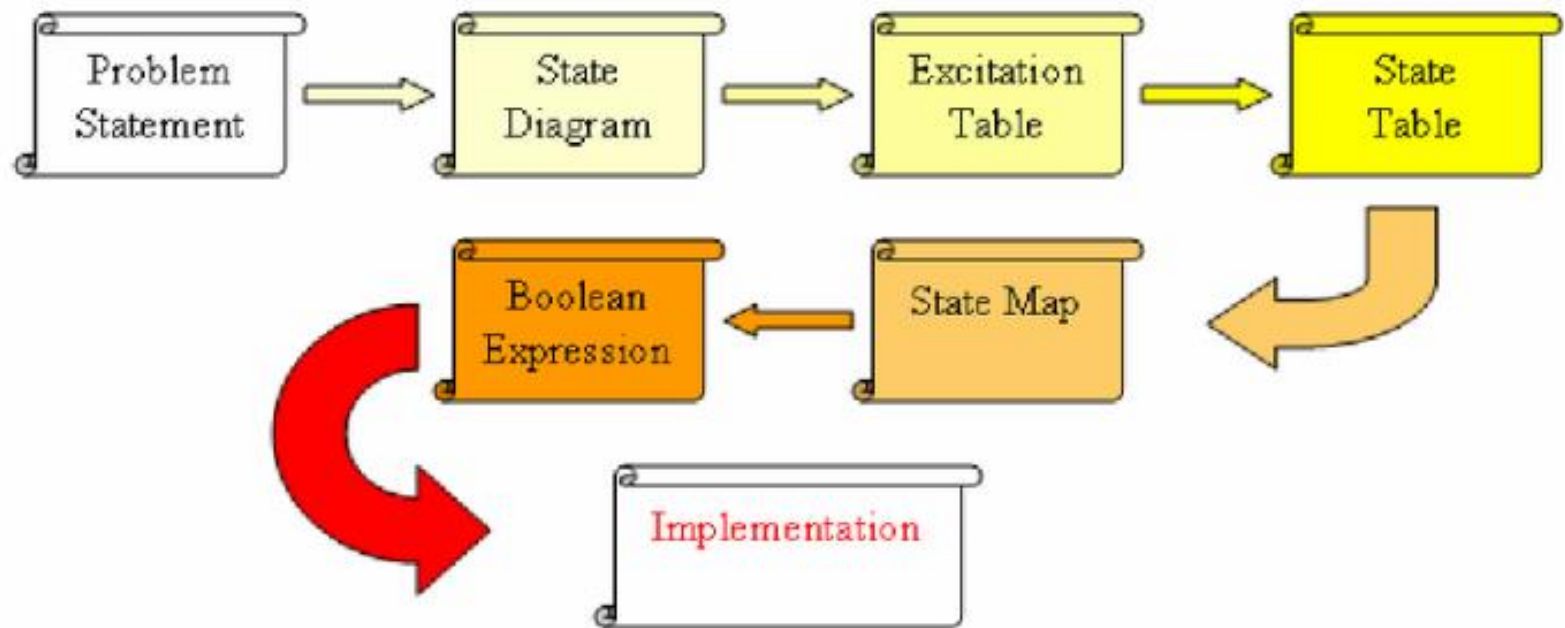
Block diagram of a clocked synchronous sequential circuit.

Example of a sequential circuit.



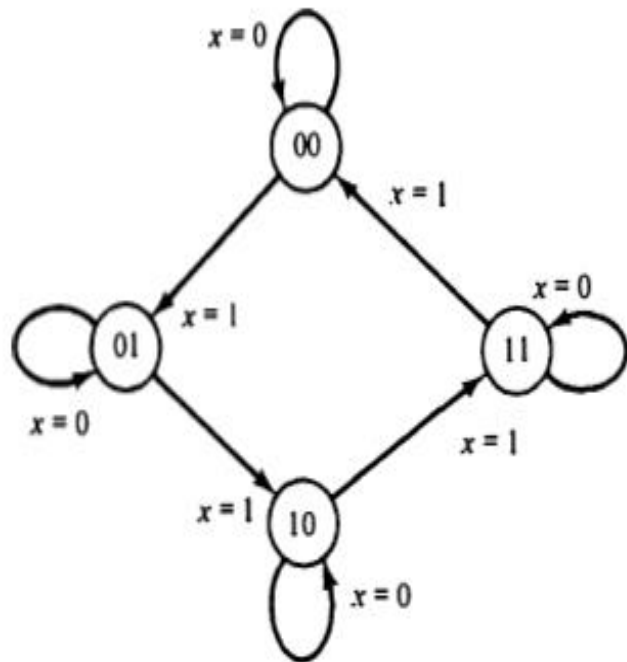
Key	Synchronous Sequential Circuits	Asynchronous Sequential Circuits
Definition	Synchronous sequential circuits are digital sequential circuits in which the feedback to the input for next output generation is governed by clock signals.	On other hand Asynchronous sequential circuits are digital sequential circuits in which the feedback to the input for next output generation is not governed by clock signals.
Memory Unit	In Synchronous sequential circuits, the memory unit which is being get used for governance is clocked flip flop.	On other hand unclocked flip flop or time delay is used as memory element in case of Asynchronous sequential circuits.
State	The states of Synchronous sequential circuits are always predictable and thus reliable.	On other hand there are chances for the Asynchronous circuits to enter into a wrong state because of the time difference between the arrivals of inputs. This is called as race condition.
Complexity	It is easy to design Synchronous sequential circuits	However on other hand the presence of feedback among logic gates causes instability issues making the design of Asynchronous sequential circuits difficult.
Performance	Due to the propagation delay of clock signal in reaching all elements of the circuit the Synchronous sequential circuits are slower in its operation speed	Since there is no clock signal delay, these are fast compared to the Synchronous Sequential Circuits
Example	Synchronous circuits are used in counters, shift registers, memory units.	On other hand Asynchronous circuits are used in low power and high speed operations such as simple microprocessors, digital signal processing units and in communication systems for email applications, internet access and networking.

Procedure for designing sequential circuits:



Design Example

State diagram for binary counter.



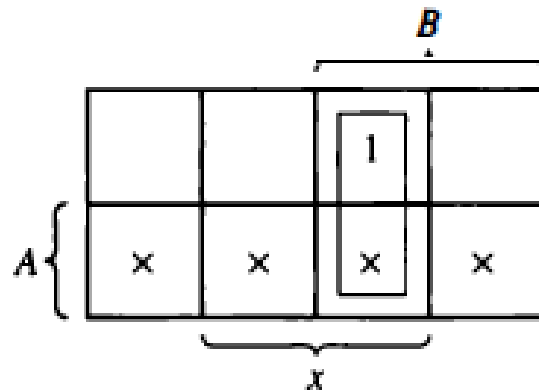
Excitation table

$Q(t)$	$Q(t + 1)$	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

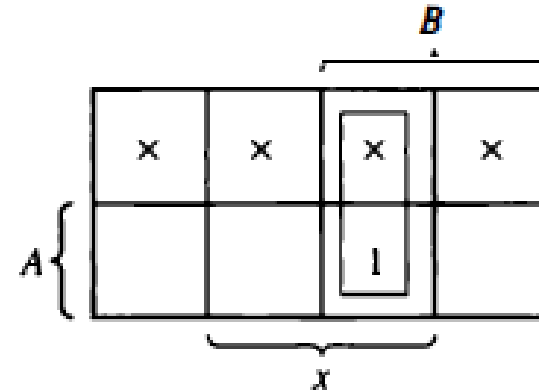
Excitation Table for Binary Counter

Present state		Input	Next state		Flip-flop inputs			
A	B		A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	×	0	×
0	0	1	0	1	0	×	1	×
0	1	0	0	1	0	×	×	0
0	1	1	1	0	1	×	×	1
1	0	0	1	0	×	0	0	×
1	0	1	1	1	×	0	1	×
1	1	0	1	1	×	0	×	0
1	1	1	0	0	×	1	×	1

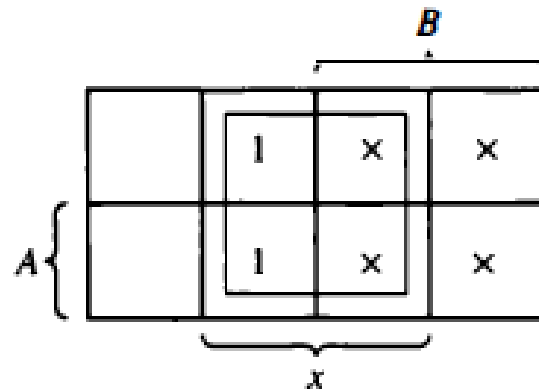
Maps for combinational circuit of counter.



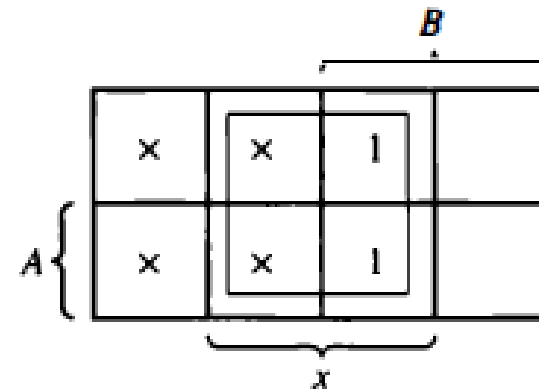
$$J_A = Bx$$



$$K_A = Bx$$

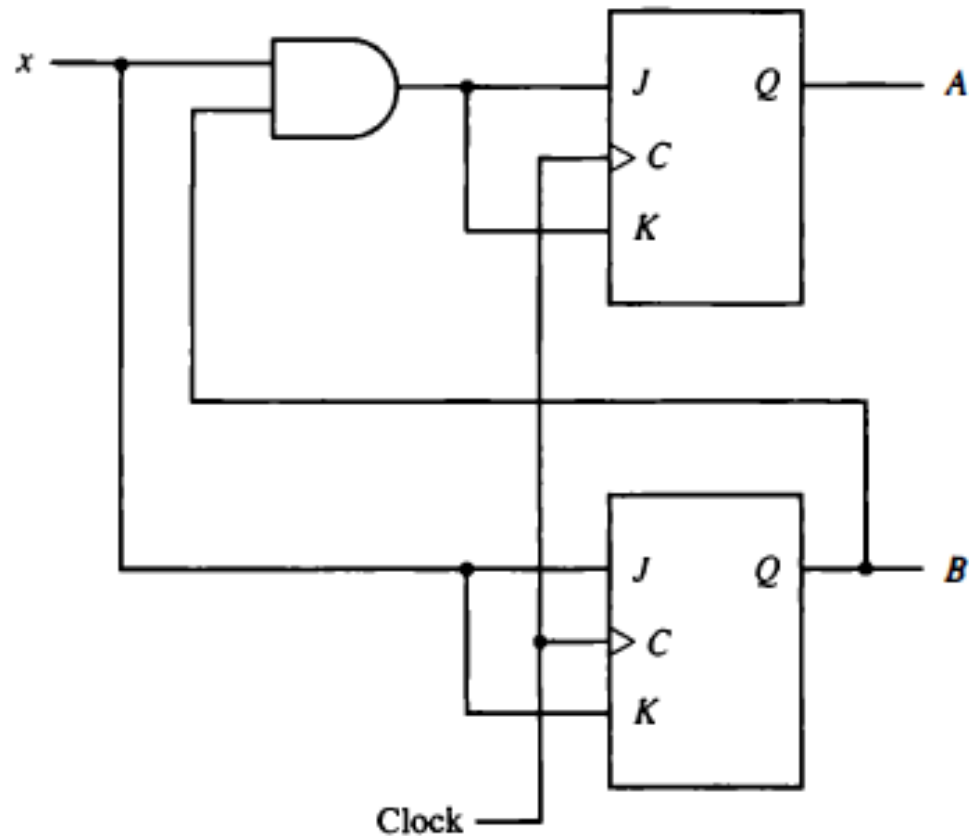


$$J_B = x$$



$$K_B = x$$

Logic diagram of a 2-bit binary counter.



Key	Combinational Circuit	Sequential Circuit
Definition	Combinational Circuit is the type of circuit in which output is independent of time and only relies on the input present at that particular instant.	On other hand Sequential circuit is the type of circuit where output not only relies on the current input but also depends on the previous output.
Feedback	In Combinational circuit as output does not depend on the time instant, no feedback is required for its next output generation.	On other hand in case of Sequential circuit output relies on its previous feedback so output of previous input is being transferred as feedback used with input for next output generation.
Performance	As the input of current instant is only required in case of Combinational circuit, it is faster and better in performance as compared to that of Sequential circuit.	On other hand Sequential circuit are comparatively slower and has low performance as compared to that of Combinational circuit.
Complexity	No implementation of feedback makes the combinational circuit less complex as compared to sequential circuit.	However on other hand implementation of feedback makes sequential circuit more complex as compared to combinational circuit.
Elementary Blocks	Elementary building blocks for combinational circuit are logic gates.	On other hand building blocks for sequential circuit are flip flops..
Operation	Combinational circuit are mainly used for arithmetic as well as Boolean operations.	On other hand Sequential circuit is mainly used for storing data.

DATA REPRESENTATION

Data Types

Data are numbers and other binary-coded information that are operated on to achieve required computational results.

The data types found in the registers of digital computers may be classified as being one of the following categories:

- (1) numbers used in arithmetic computations,
- (2) letters of the alphabet used in data processing. and
- (3) other discrete symbols used for specific purposes.

All types of data, except binary numbers, are represented in computer registers in binary -coded form. This is because registers are made up of flip-flops and flip-flops are two-state devices that can store only 1's and 0's.

A *bit* is the most basic unit of information in a computer.

- It is a state of “on” or “off” in a digital circuit.
- Sometimes these states are “high” or “low” voltage instead of “on” or “off..”

A *byte* is a group of eight bits.

- A byte is the smallest possible *addressable* unit of computer storage.
- The term, “addressable,” means that a particular byte can be retrieved according to its location in memory.

A *word* is a contiguous group of bytes.

- Words can be any number of bits or bytes.
- Word sizes of 16, 32, or 64 bits are most common.
- In a word-addressable system, a word is the smallest addressable unit of storage.

A group of four bits is called a *nibble*.

- Bytes, therefore, consist of two nibbles: a “high-order nibble,” and a “low-order” nibble.

Number Systems:

Number systems are very important to understand because the design and organization of a computer depends on the number systems.

The four kind of number system used by the digital computer –

1. Decimal number system
2. Binary number system
3. Octal number system
4. Hexadecimal number system

Decimal Number System

- The decimal number system consists of 10 digits namely 0 to 9.
- Since the decimal number system consists of 10 digits, the base or radix of this system is 10.

e.g $(405)_{10}$, $(145.25)_{10}$

Binary Number System

- The binary number system consists of 2 digits namely 0 and 1.
- Since the binary number system consists of 2 digits, the base or radix of this system is 2.

e.g $(101)_2$, $(1001.11)_2$

Octal Number System

- The octal number system consists of 8 digits namely 0 to 7.
- Since the Octal number system consists of 8 digits, the base or radix of this system is 8.

e.g $(76)_8$, $(55.25)_8$

Hexadecimal Number System

- The Hexadecimal number system, popularly known as Hex system has 16 symbols, therefore its base/radix is 16.
- The 16 symbols used in Hexadecimal system are 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

e.g $(45)_{16}$, $(11A)_{16}$

Decimal, Binary, Hexadecimal, Octal

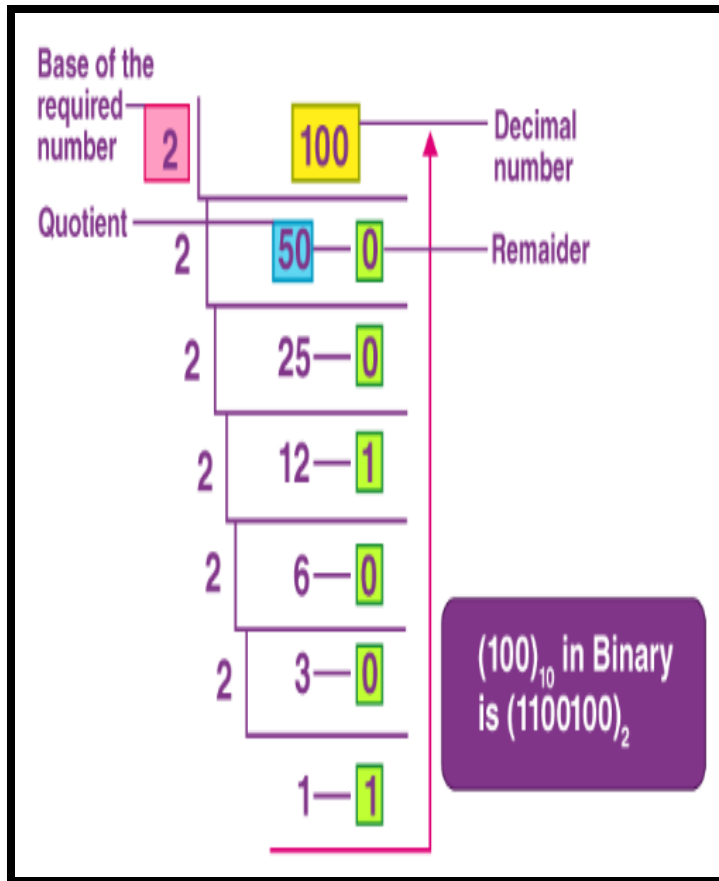
Decimal	Binary	Hexadecimal	Octal
0	00000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	10
9	1001	9	11
10	1010	A	12
11	1011	B	13
12	1100	C	14
13	1101	D	15
14	1110	E	16
15	1111	F	17

Conversion between Number Systems

Decimal real number into Binary, Octal and Hexadecimal

- **Step 1.** Multiply the fractional part by the base of the numbers system (2, 8 or 16).
- **Step 2.** Remove the whole number from the product (the result of the multiplication) and collect it separately.
- **Step 3.** Repeat the step 1 and 2 with the new fractional part till the fractional part becomes zero.

Decimal real number into Binary, Octal and Hexadecimal



$$\begin{array}{r} 8 \overline{) 158} \\ 8 \overline{) 19} - 6 \\ 8 \overline{) 2} - 3 \\ 8 \overline{) 0} - 2 \end{array}$$

$(158)_{10} = (236)_8$

$$\begin{array}{r} 16 \overline{) 450} \\ 16 \overline{) 28} - 2 \\ 16 \overline{) 1} - 12(C) \\ 16 \overline{) 0} - 1 \end{array}$$

$(450)_{10} = (1C2)_{16}$

Conversion of decimal 41.6875 into binary.

Integer = 41

41	
20	1
10	0
5	0
2	1
1	0
0	1

$$(41)_{10} = (101001)_2$$

Fraction = 0.6875

0.6875
<u>2</u>
1.3750
<u>x 2</u>
0.7500
<u>x 2</u>
1.5000
<u>x 2</u>
1.0000

$$(0.6875)_{10} = (0.1011)_2$$

$$(41.6875)_{10} = (101001.1011)_2$$

Binary to Decimal

Any binary number can be converted into decimal number using the weights assigned to each bit.

e.g. $(11011)_2$ Its decimal equivalent is

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = (27)_{10}$$

Binary to Octal

Indirect Method:

Binary \rightarrow Decimal \rightarrow Octal

e.g. $(11011)_2$

Its decimal equivalent is

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ = (27)_{10}$$

Octal equivalent is $(33)_8$

Direct Method

Binary \rightarrow Octal

➤ Step 1: Make the group of 3-bits from right to left for integer from left to right for fraction.

➤ Step 2: Find decimal equivalent of each group.

e.g. $(101111)_2 = (57)_8$

Binary to Hexadecimal

Indirect Method:

Binary \rightarrow Decimal \rightarrow Hexa

e.g. $(11011)_2$

Its decimal equivalent is

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ = (27)_{10}$$

Hexa equivalent is $(1B)_{16}$

(division method)

Direct Method:

Binary \rightarrow Hexa

Step 1: Make the group of 4-bits from right to left for integer from left to right for fraction.

Step 2: Find decimal equivalent of each group.

e.g. $(101111)_2 = (?)_{16}$

$$(\underline{0010} \ \underline{1111})_2 = (2 \ 15)_{16} = \\ (2F)_{16}$$

Octal to Hexadecimal

Octal → Binary → Hexa

Step 1:

Octal to Binary Conversion

7	5	2
111	101	010

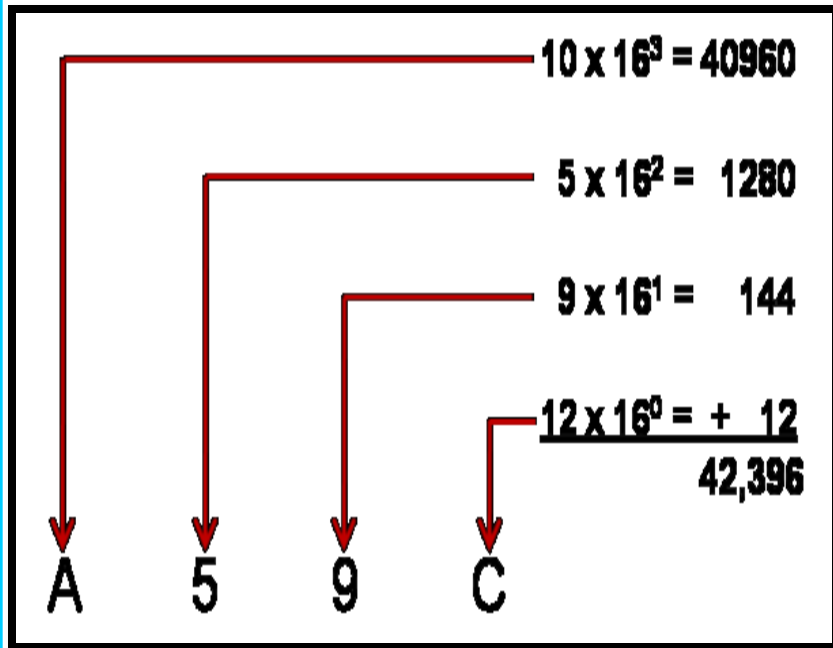
So the binary equivalent is 111101010

Step 2:

Binary to Hex Conversion

<u>0001</u>	<u>1110</u>	<u>1010</u>
1	E	9

Hexa to decimal



Hexa to binary

Binary to Hex Conversion

0001

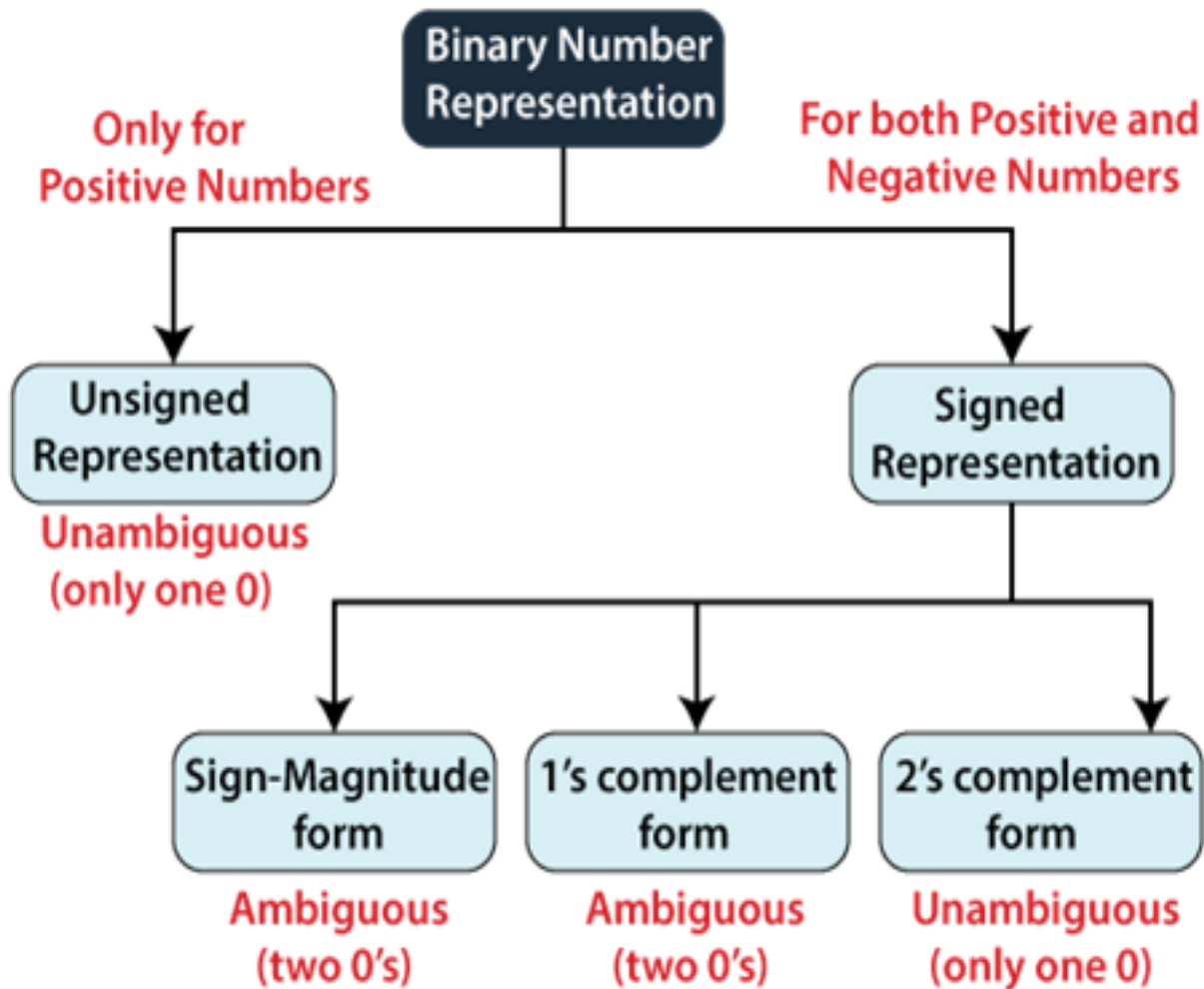
1110

1010

1

E

9



Fixed-Point Representation:

- Positive integers, including zero, can be represented as unsigned numbers.
- Negative integers, we need a notation for negative values.
- In ordinary arithmetic, a negative number is indicated by a minus sign and a positive number by a plus sign.
- Computers must represent everything with 1's and 0's, including the sign of a number.

- The convention is to make the sign bit equal to 0 for positive and to 1 for negative.
- To represent the sign with a bit placed in the leftmost position of the number.
- Representation of positive number is same in all the three representation

Positive number – MSB ‘0’

Negative number – MSB ‘1’

1. Sign magnitude representation

- In the sign magnitude representation, positive number have a additional bit (sign bit) 0, while the negative number has a sign bit 1, while the magnitude is a simple binary equivalent of the number.
- N – bit signed binary number, MSB is a sign bit and remaining N-1 bit represents magnitude

E.g. +5 and -5 can be representing in 6 bit register as:

+5 = 0 00101 and -5 = 1 00101

Signed Magnitude Form

Decimal	Signed Magnitude
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
-0	1000
-1	1001
-2	1010
-3	1011
-4	1100
-5	1101
-6	1110
-7	1111
-8	—

2. One's Complement representation

- In one's complement, positive numbers are represented as usual in signed magnitude.
- However, negative numbers are represented differently.
- To negate a number, replace all zeros with ones, and ones with zeros - flip the bits.

+12 = 0 0001100, and -12 = 1 1110011.

Signed – 1's Complement Form

Decimal	Signed-1's Complement
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
–0	1111
–1	1110
–2	1101
–3	1100
–4	1011
–5	1010
–6	1001
–7	1000
–8	—

3. Two's Complement representation

- In two's complement, positive numbers are represented as usual in signed magnitude.
- However, negative numbers are represented by adding 1 in magnitude part of one's complement.

$$+12 = 0\ 0001100$$

$$-12 = 1\ 1110011 \text{ (1's complement)}$$


$$-12 = 1\ 1110100 \text{ (2's complement)}$$

Signed – 2's Complement Form

Decimal	Signed-2's Complement
+7	0111
+6	0110
+5	0101
+4	0100
+3	0011
+2	0010
+1	0001
+0	0000
—0	—
—1	1111
—2	1110
—3	1101
—4	1100
—5	1011
—6	1010
—7	1001
—8	1000

- With one's complement addition, the carry bit is “carried around” and added to the sum.

– Example: Using one's complement binary arithmetic, find the sum of 48 and - 19



$$\begin{array}{r}
 1 \\
 00110000 \\
 11101100 \\
 \hline
 00011100 \\
 +1 \\
 \hline
 00011101
 \end{array}$$

We note that 19 in binary is
so -19 in one's complement is:

00010011,
11101100.

➤ Although the “end carry around” adds some complexity, one’s complement is simpler to implement than signed magnitude.

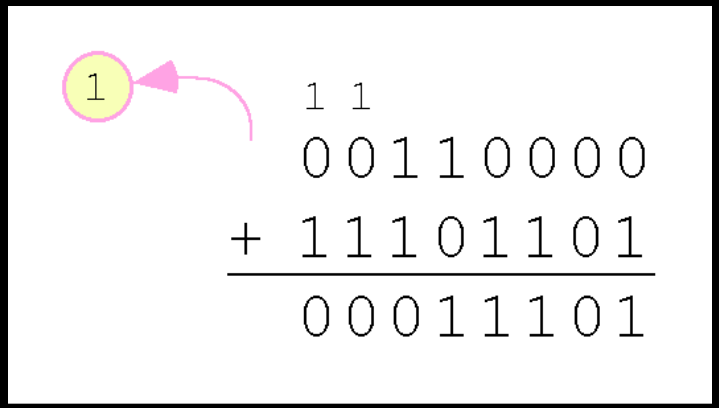
➤ But it still has the disadvantage of having two different representations for zero: positive zero and negative zero.

➤ Two’s complement solves this problem.

➤ Two’s complement is the radix complement of the binary numbering system; the *radix complement* of a non-zero number N in base r with d digits is $r^d - N$.

- With **two's complement arithmetic**, all we do is add our two binary numbers. Just discard any carries emitting from the high order bit.

– Example: Using one's complement binary arithmetic, find the sum of 48 and -19.



$$\begin{array}{r}
 11 \\
 00110000 \\
 + 11101101 \\
 \hline
 00011101
 \end{array}$$

We note that 19 in binary is: **00010011**,
 so -19 using one's complement is: **11101100**,
 and -19 using two's complement is: **11101101**.

Floating-Point Representation:

- In the decimal system, very large and very small numbers are expressed in scientific notation as follows: 4.69×10^{23} and 1.601×10^{-19}
- Binary numbers can also be expressed by the floating point representation.
- The floating point representation of a number consists of two parts:
 - The first part represents a signed, fixed point number called mantissa(m)
 - The second part designates the position of the decimal point is called the exponent(e).

To represent the fractional values & values beyond 2^n-1 .

$$\begin{array}{lcl} +3207.23 & = & 3.20723 \times 10^3 \\ -0.000321 & = & -3.21 \times 10^{-4} \end{array}$$

Diagram illustrating the components of scientific notation for the second example:

- Sign:** Points to the negative sign ($-$) in -3.21×10^{-4} .
- Mantissa:** Points to the mantissa (3.21) in -3.21×10^{-4} .
- Radix (base):** Points to the base (10) in -3.21×10^{-4} .
- Exponent:** Points to the exponent (-4) in -3.21×10^{-4} .

1. Single precision

■ 32-bits

- 23-bit mantissa
- 8-bit exponent
- 1-bit sign



2. Double precision

■ 64-bits

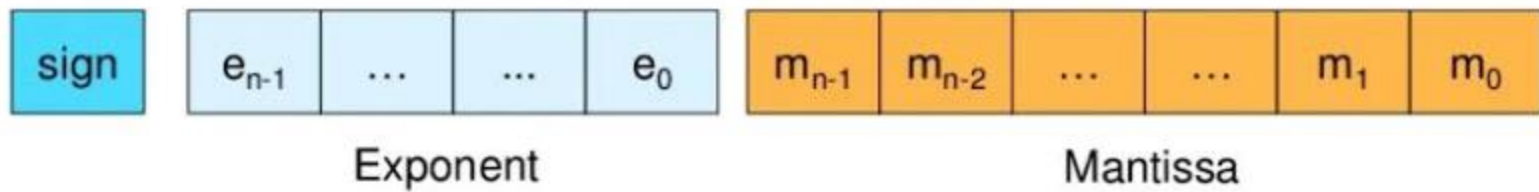
- 52-bit mantissa
- 11-bit exponent
- 1-bit sign



$$N = (-1)^s m \cdot 2^e$$

Diagram illustrating the IEEE 754 floating-point format components:

- Sign:** Points to the superscript s in $(-1)^s$.
- Mantissa:** Points to the mantissa m .
- Radix:** Points to the base 2 in 2^e .
- Exponent:** Points to the superscript e in 2^e .



Character Representation:

- Belong to category of qualitative data
- Represent quality or characteristics
- Includes

■ Letters	a-z, A-Z
■ Digits	0-9
■ Symbols	!, @ , *, /, &, #, \$
■ Control characters	<CR>, <BEL>, <ESC>, <LF>

- With a single byte (8-bits) 256 characters can be represented
- Standards
 - ASCII – American Standard Code for Information Interchange
 - EBCDIC – Extended Binary-Coded Decimal Interchange Code
 - Unicode

ASCII CODE:

- ▣ Used to represent
 - Upper & lower-case Latin letters
 - Numbers
 - Punctuations
 - Control characters
- ▣ There are 128 standard ASCII codes
 - Can be represented by a 7 digit binary number
 - ▣ 000 0000 through 111 1111
 - Plus parity bit

ASCII TABLE:

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
64	40	@	80	50	P	96	60	`
65	41	A	81	51	Q	97	61	a
66	42	B	82	52	R	98	62	b
67	43	C	83	53	S	99	63	c
68	44	D	84	54	T	100	64	d
69	45	E	85	55	U	101	65	e
70	46	F	86	56	V	102	66	f
71	47	G	87	57	W	103	67	g
72	48	H	88	58	X	104	68	h
73	49	I	89	59	Y	105	69	i
74	4A	J	90	5A	Z	106	6A	j
75	4B	K	91	5B	[107	6B	k
76	4C	L	92	5C	\	108	6C	l
77	4D	M	93	5D]	109	6D	m
78	4E	N	94	5E	^	110	6E	n
79	4F	O	95	5F	_	111	6F	o

ASCII	Hex	Symbol
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI

ASCII	Hex	Symbol
32	20	(space)
33	21	!
34	22	"
35	23	#
36	24	\$
37	25	%
38	26	&
39	27	'
40	28	(
41	29)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

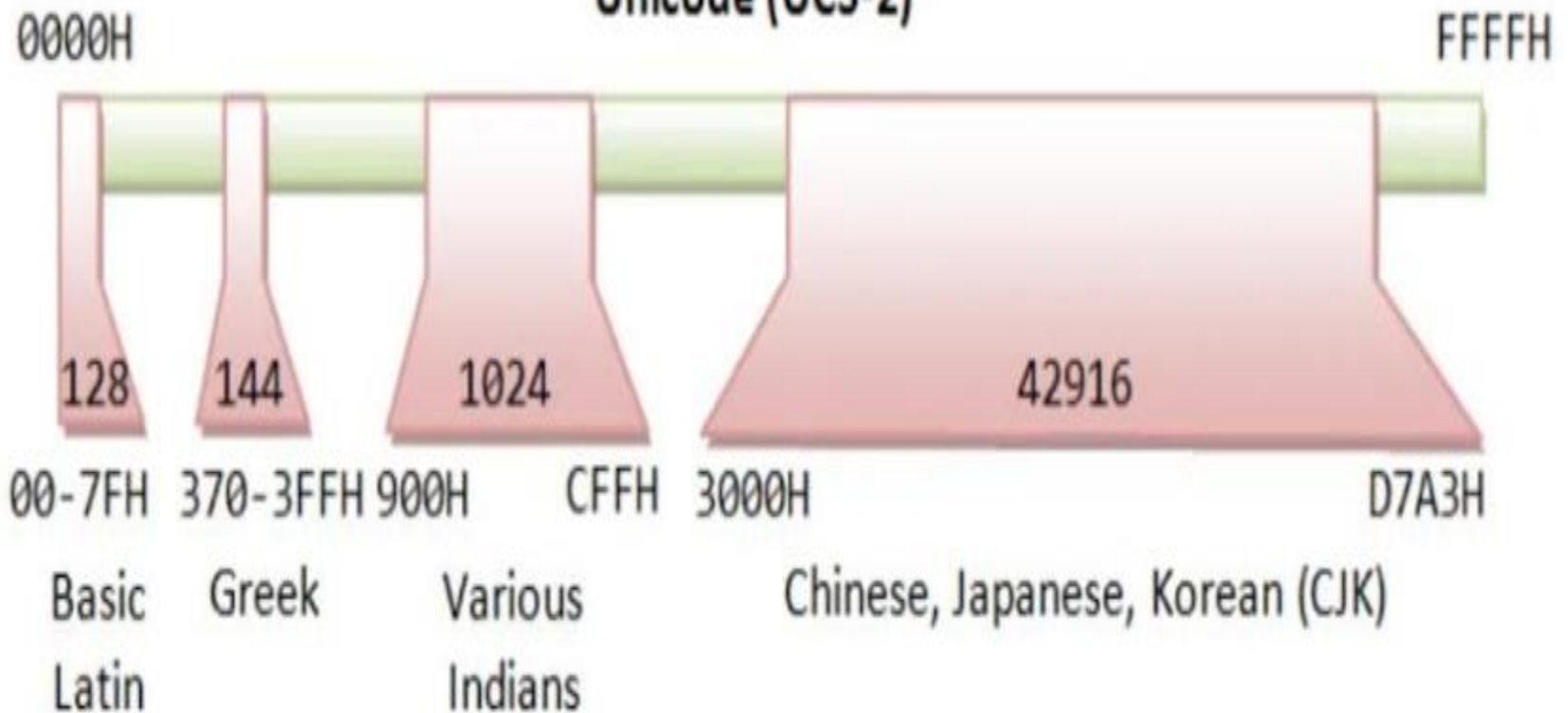
ASCII	Hex	Symbol
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

- ❑ ASCII codes for digits aren't equal to numeric value
- ❑ Uppercase & lowercase alphabetic codes differ by 0x20
 - Shift key clears bit 5
 - $0x20 = 32 = 0010\ 0000$
 - Example:
 - ❑ $A = 65 = 0100\ 0001$
 - ❑ $a = 97 = 0110\ 0001$
- ❑ Most languages need more than 128 characters

UNICODE:

- ❑ Designed to overcome limitation of number of characters
- ❑ Assigns unique character codes to characters in a wide range of languages
- ❑ A 16-bit character set
 - UCS-2
 - UCS-4 is 32-bit
- ❑ 65,536 (2^{16}) distinct Unicode characters

Unicode (UCS-2)



BINARY AND DECIMAL CODES:

4-Bit Gray Code

Binary code	Decimal equivalent	Binary code	Decimal equivalent
0000	0	1100	8
0001	1	1101	9
0011	2	1111	10
0010	3	1110	11
0110	4	1010	12
0111	5	1011	13
0101	6	1001	14
0100	7	1000	15

Four Different Binary Codes for the Decimal Digit

Decimal digit	BCD 8421	2421	Excess-3	Excess-3 gray
0	0000	0000	0011	0010
1	0001	0001	0100	0110
2	0010	0010	0101	0111
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1100
6	0110	1100	1001	1101
7	0111	1101	1010	1111
8	1000	1110	1011	1110
9	1001	1111	1100	1010
Unused bit combi- nations	1010	0101	0000	0000
	1011	0110	0001	0001
	1100	0111	0010	0011
	1101	1000	1101	1000
	1110	1001	1110	1001
	1111	1010	1111	1011

Computer Arithmetic

Introduction:

- Data is manipulated by using the arithmetic instructions in digital computers.
- Data is manipulated to produce results necessary to give solution for the computation problems.
- The Addition, subtraction, multiplication and division are the four basic arithmetic operations.
- An arithmetic instruction may specify binary or decimal data, and in each case the data may be in fixed-point or floating-point form.

- Fixed-point numbers may represent integers or fractions.
- Negative numbers may be in signed-magnitude or signed-complement representation.
- The arithmetic processor is very simple if only a binary fixed-point add instruction is included.
- It would be more complicated if it includes all four arithmetic operations for binary and decimal data in fixed-point and floating-point representation.

Addition and Subtraction:

➤ There are three ways of representing negative fixed-point binary numbers:

- Signed - magnitude,
- Signed - 1's complement,
- Signed - 2's complement.

➤ Most computers use the signed-2's complement representation when performing arithmetic operations with integers.

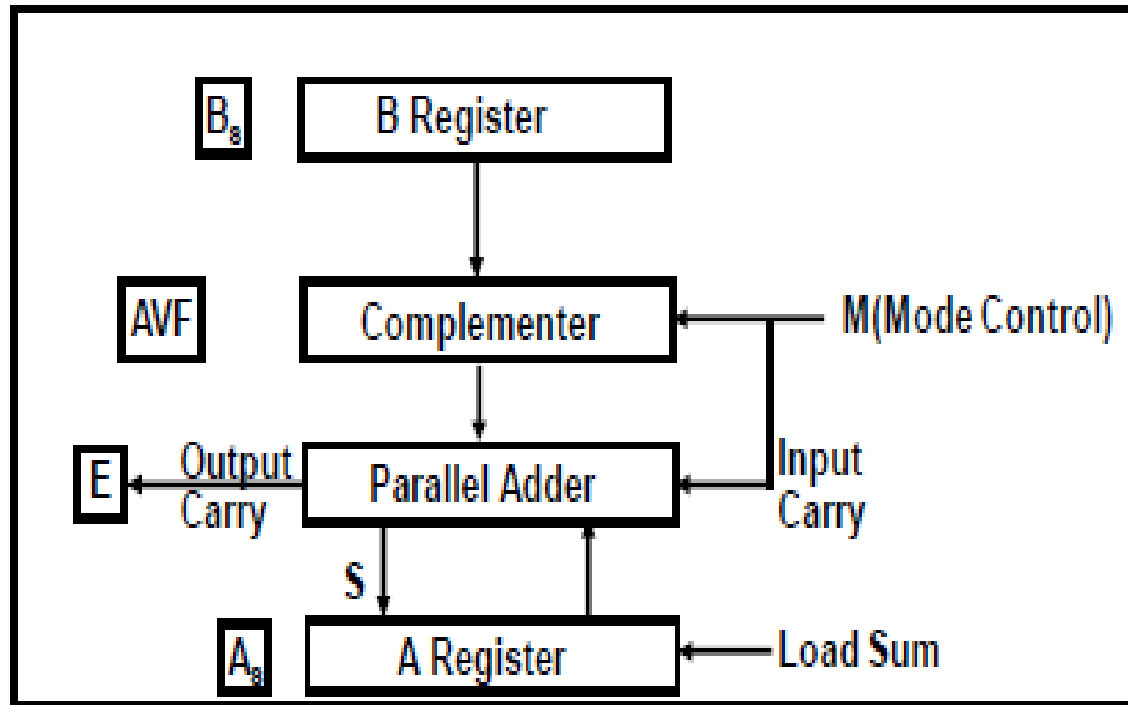
➤ For floating-point operations, most computers use the signed-magnitude representation for the mantissa.

Addition and Subtraction with Signed-Magnitude Data:

When the signed numbers are added or subtracted, we find that there are eight different conditions to consider, depending on the sign of the numbers and the operation performed.

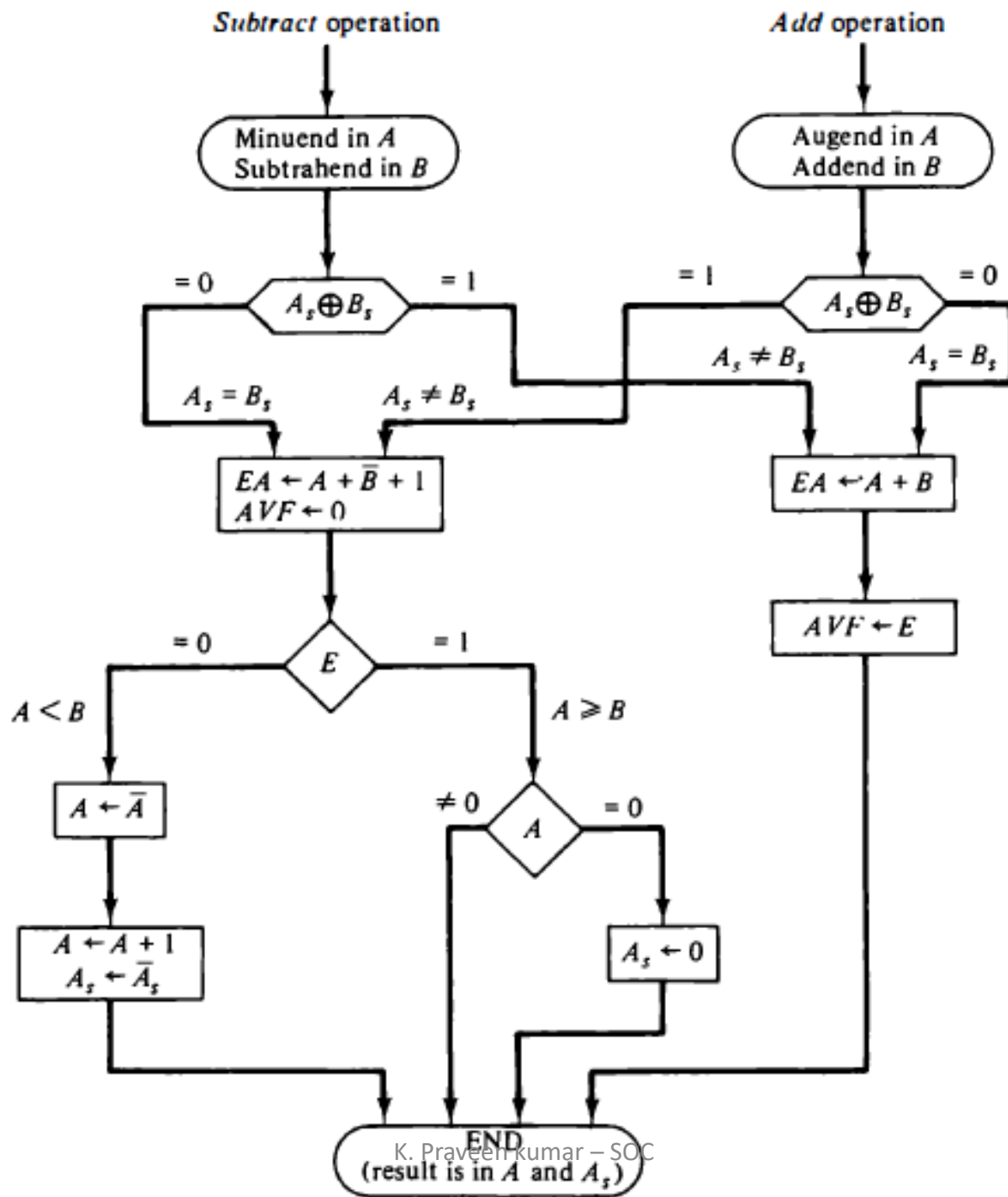
Operation	Add Magnitudes	Subtract Magnitudes		
		When $A > B$	When $A < B$	When $A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

Hardware Implementation:



- A_s -- Sign of A
- B_s -- Sign of B
- A & B -- Accumulator and b register
- AVB -- Add Overflow flip-flop bit for A+ B
- E -- Output carry bit for parallel adder
- $M = 0$ or 1 -- 0 for addition and 1 for subtraction

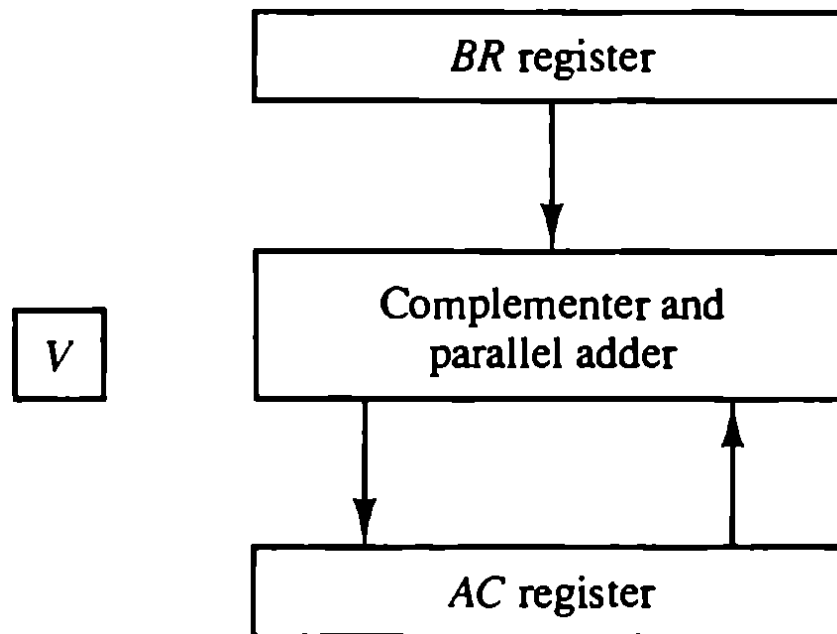
FLOWCHART FOR ADDITION AND SUBTRACT OPERATION



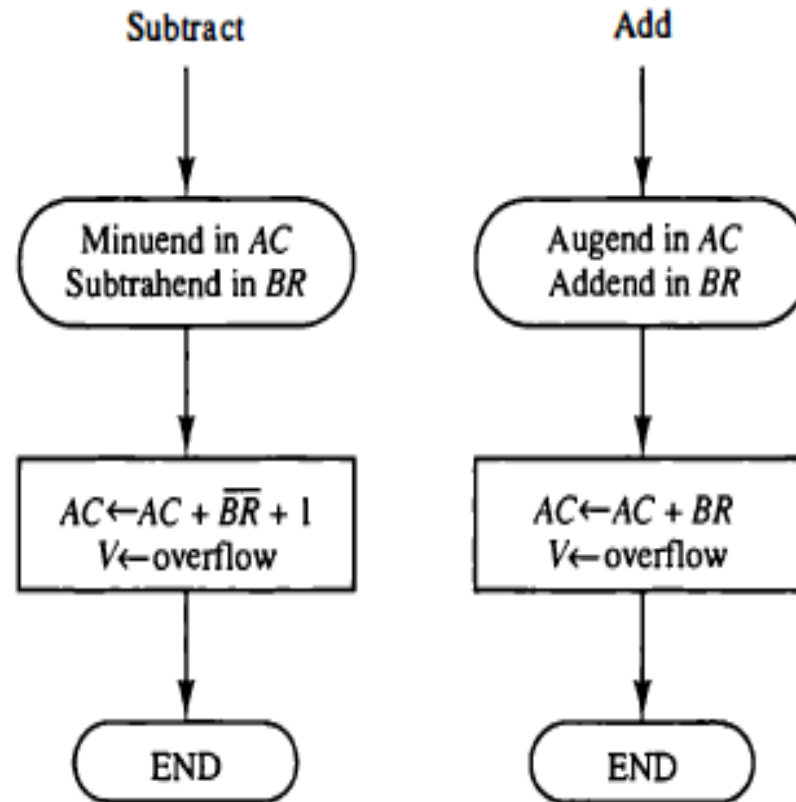
Addition and Subtraction with Signed-2's Complement Data:

- The addition of two numbers in signed-2's complement form consists of adding the numbers with the sign bits treated the same as the other bits of the number.
- A carry-out of the sign-bit position is discarded.
- The subtraction consists of first taking the 2's complement of the subtrahend and then adding it to the minuend.

Hardware Implementation



Algorithm for adding and subtracting numbers in signed-2's complement representation.



Design of fast adders

Binary addition would seem to be dramatically slower for large registers

- consider $0111 + 0011$
- carries propagate left-to-right
- So 64-bit addition would be 8 times slower than 8-bit addition

It is possible to build a circuit called a “carry look-ahead adder” that speeds up addition by eliminating the need to “ripple” carries through the word.

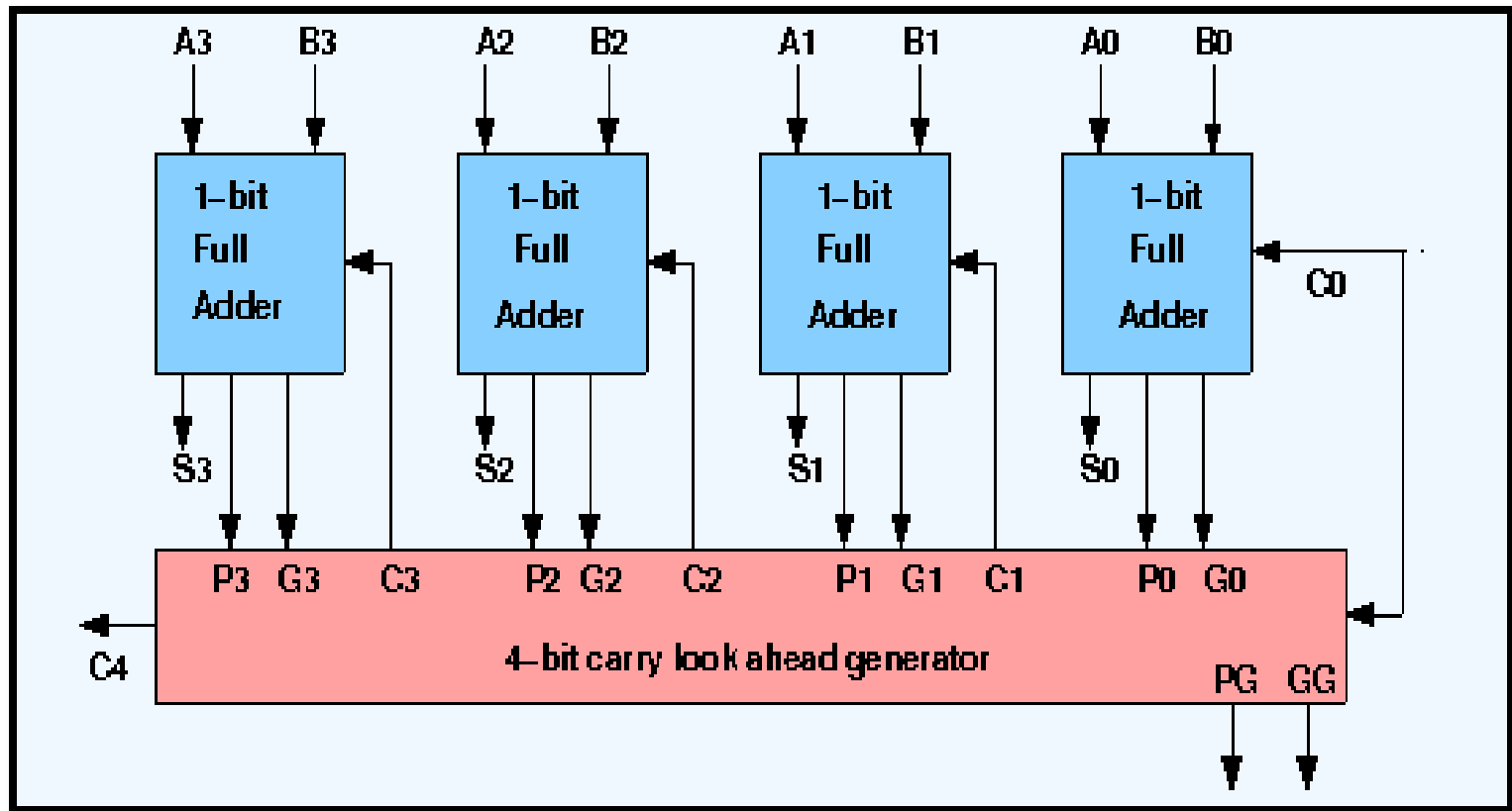
➤ To reduce the computation time, there are faster ways to add two binary numbers by using carry lookahead adders. They work by creating two signals P and G known to be **Carry Propagator** and **Carry Generator**.

➤ The carry propagator is propagated to the next level whereas the carry generator is used to generate the output carry, regardless of input carry.

➤ The number of gate levels for the carry propagation can be found from the circuit of full adder.

➤ The signal from input carry C_{in} to output carry C_{out} requires an AND gate and an OR gate, which constitutes two gate levels.

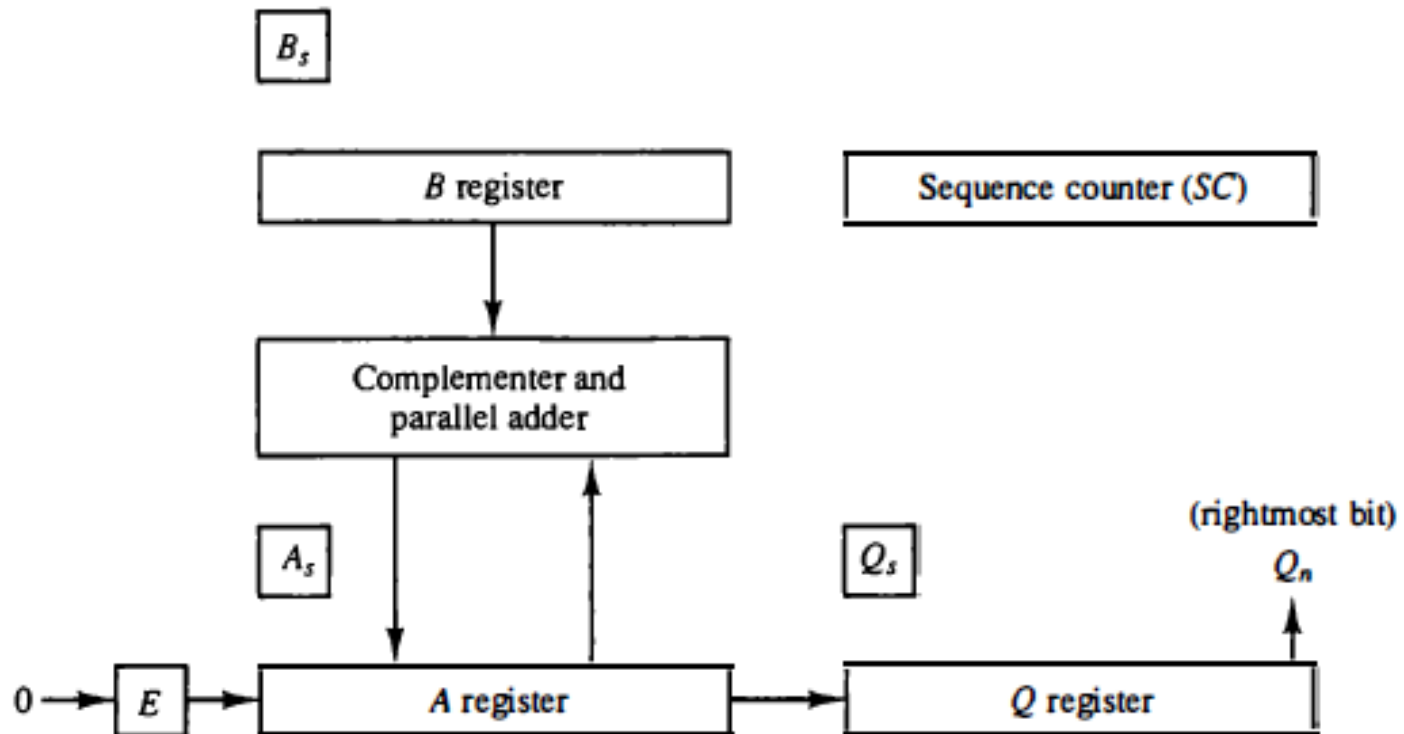
➤ So if there are four full adders in the parallel adder, the output carry C_5 would have $2 \times 4 = 8$ gate levels from C_1 to C_5 . For an n-bit parallel adder, there are $2n$ gate levels to propagate through.



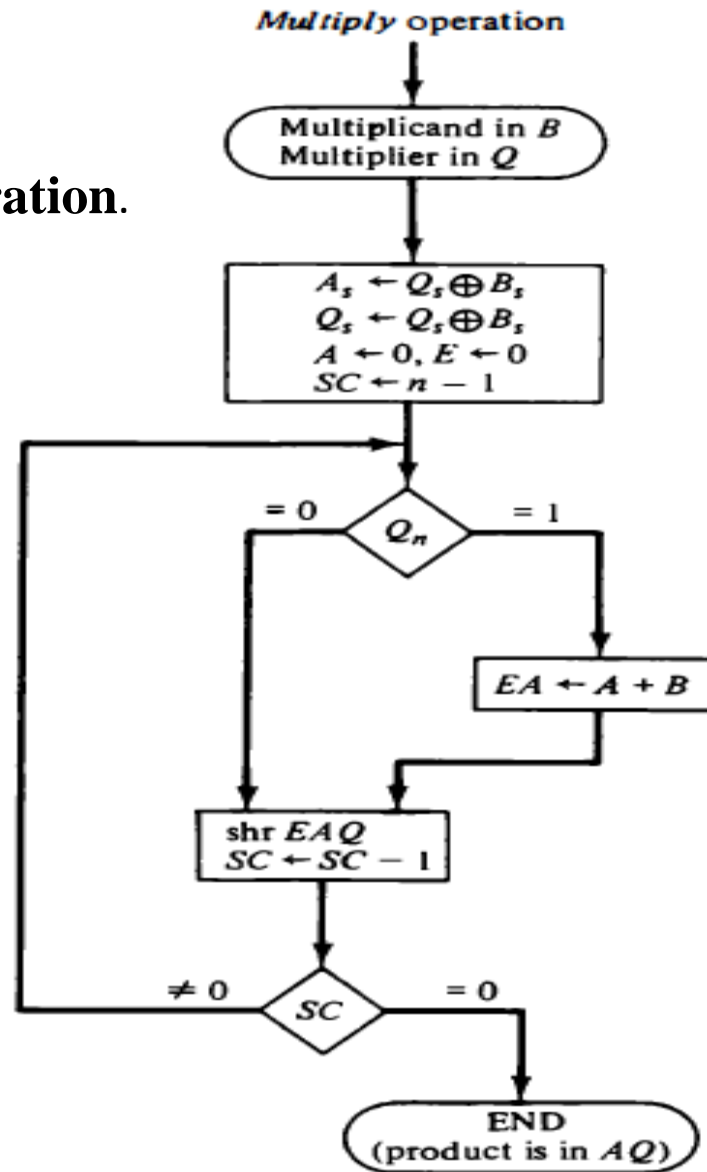
Multiplication Algorithms

23	10111	Multiplicand
<u>19</u>	<u>× 10011</u>	Multiplier
	10111	
	10111	
	00000	+
	00000	
	10111	
437	<u>110110101</u>	Product

Hardware for multiply operation.



Flowchart for unsigned Binary multiplication operation.



Numerical Example for Binary Multiplier

Multiplicand $B = 10111$	E	A	Q	SC
Multiplier in Q	0	00000	10011	101
$Q_n = 1$; add B		<u>10111</u>		
First partial product	0	10111		
Shift right EAQ	0	01011	11001	100
$Q_n = 1$; add B		<u>10111</u>		
Second partial product	1	00010		
Shift right EAQ	0	10001	01100	011
$Q_n = 0$; shift right EAQ	0	01000	10110	010
$Q_n = 0$; shift right EAQ	0	00100	01011	001
$Q_n = 1$; add B		<u>10111</u>		
Fifth partial product	0	11011		
Shift right EAQ	0	01101	10101	000
Final product in $AQ = 0110110101$				

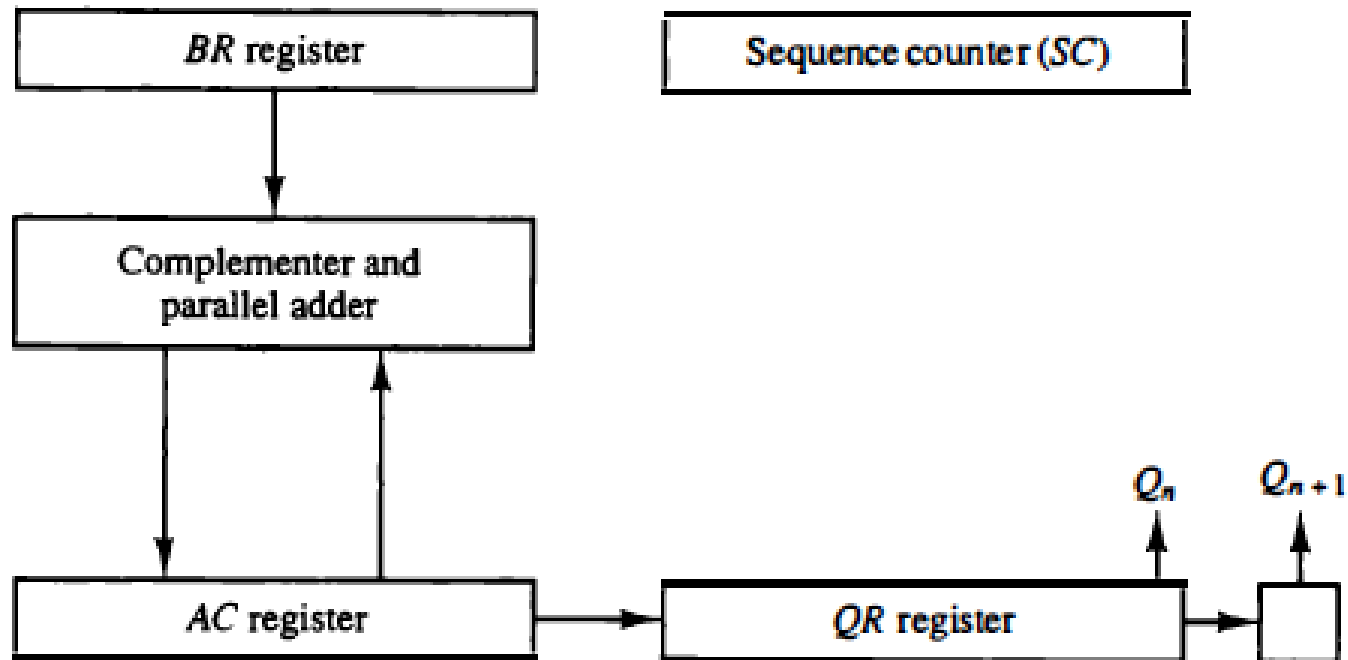
Booth Multiplication Algorithm

- Booth algorithm gives a procedure for multiplying binary integers in signed-2's complement representation.
- It operates on the fact that strings of 0's in the multiplier require no addition but just shifting and a string of 1's in the multiplier from bit weight 2^k to weight 2^m can be treated as $2^{(k+1)}$ to 2^m .
- Booth algorithm requires examination of the multiplier bits and shifting of the partial product.

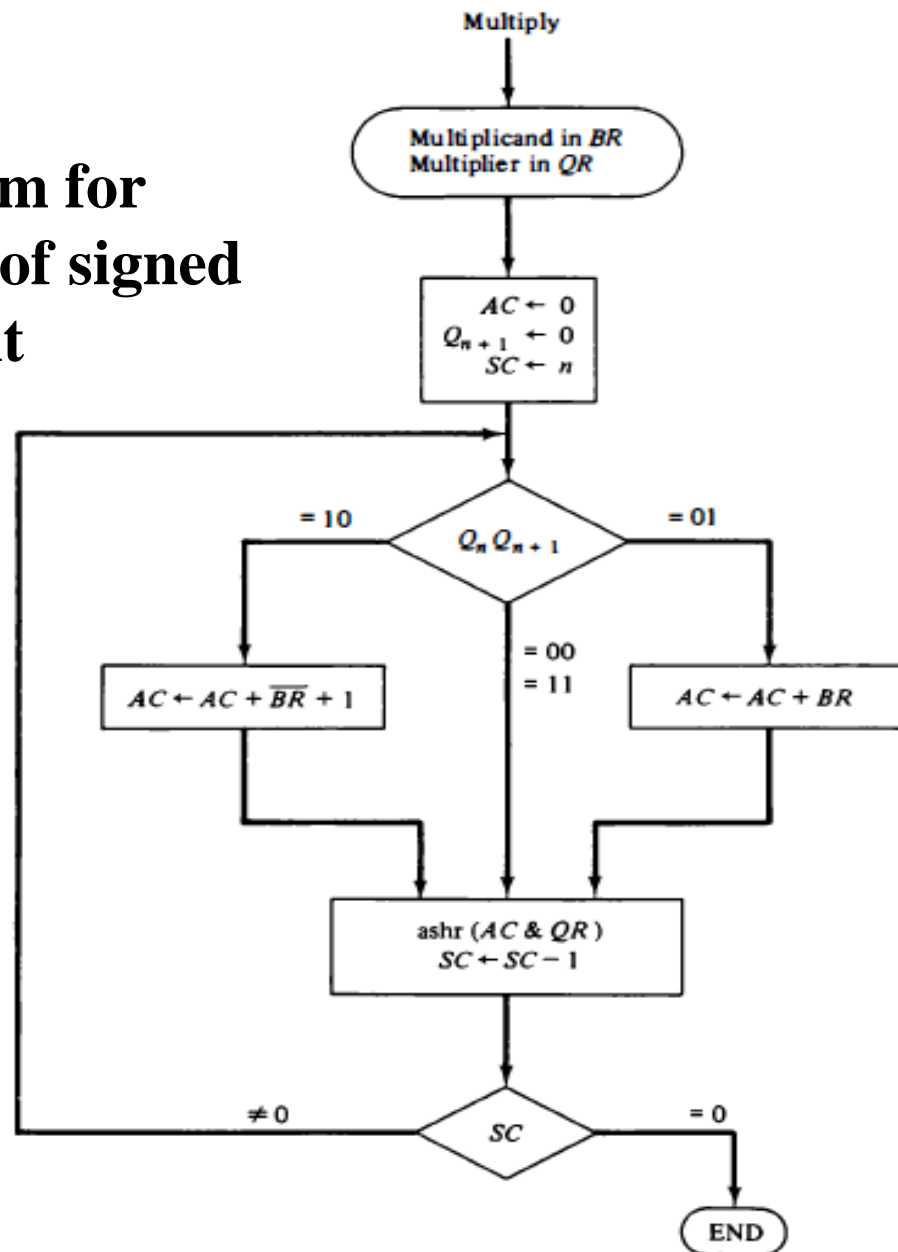
Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules:

- The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1's in the multiplier
- The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous '1') in a string of 0's in the multiplier.
- The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

Hardware Implementation



Booth algorithm for multiplication of signed 2's complement numbers.



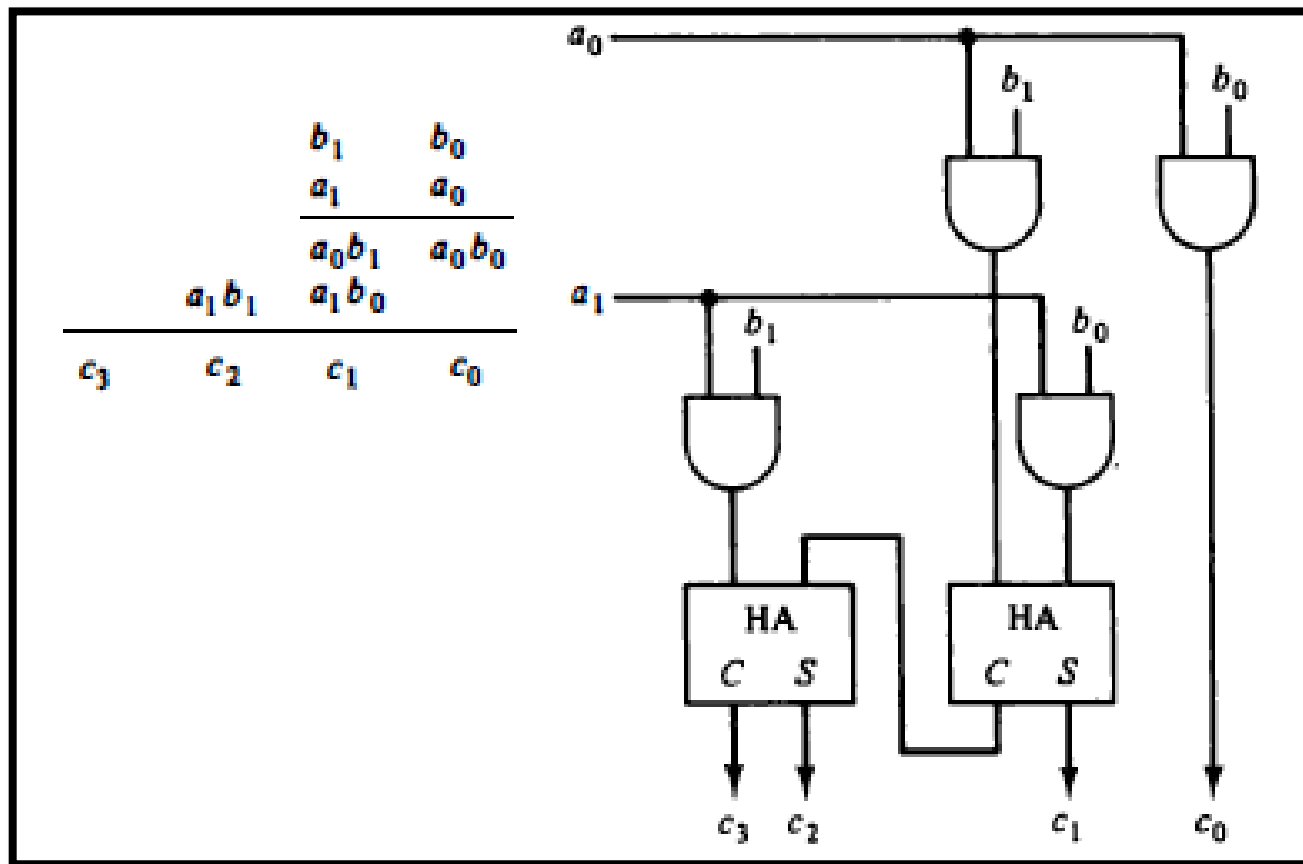
Example of Multiplication with Booth Algorithm

$Q_n Q_{n+1}$	$BR = 10111$ $\overline{BR} + 1 = 01001$	AC	QR	Q_{n+1}	SC
	Initial	00000	10011	0	101
1 0	Subtract BR	$\begin{array}{r} 01001 \\ \underline{01001} \end{array}$			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	Add BR	$\begin{array}{r} 10111 \\ \underline{10111} \end{array}$			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	Subtract BR	$\begin{array}{r} 01001 \\ \underline{01001} \end{array}$			
	ashr	00011	10101	1	000

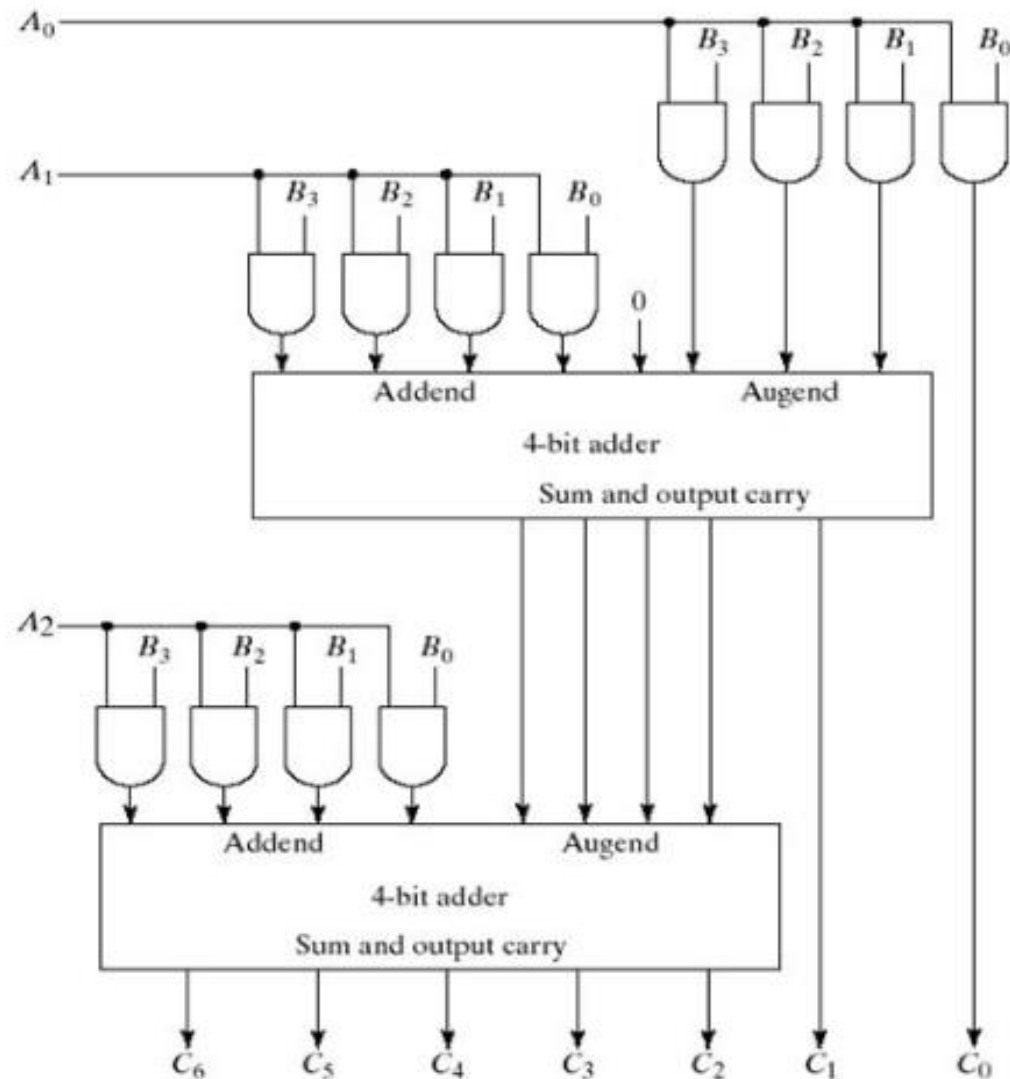
Array Multiplier

- An array multiplier is a digital combinational circuit used for multiplying two binary numbers by employing an array of full adders and half adders.
- Checking the bits of the multiplier one at a time and forming partial products is a sequential operation that requires a sequence of add and shift micro-operations.
- The multiplication of two binary numbers can be done with one micro-operation by means of a combinational circuit that forms the product bits all at once.
- This is a fast way of multiplying two numbers since all it takes is the time for the signals to propagate through the gates that form the multiplication array.

2-bit by 2-bit array multiplier.



4 bit by 3 bit array multiplier



Division algorithm

Binary division is simpler than decimal division because the quotient digits are either 0 or 1 and there is no need to estimate how many times the dividend or partial remainder fits into the divisor.

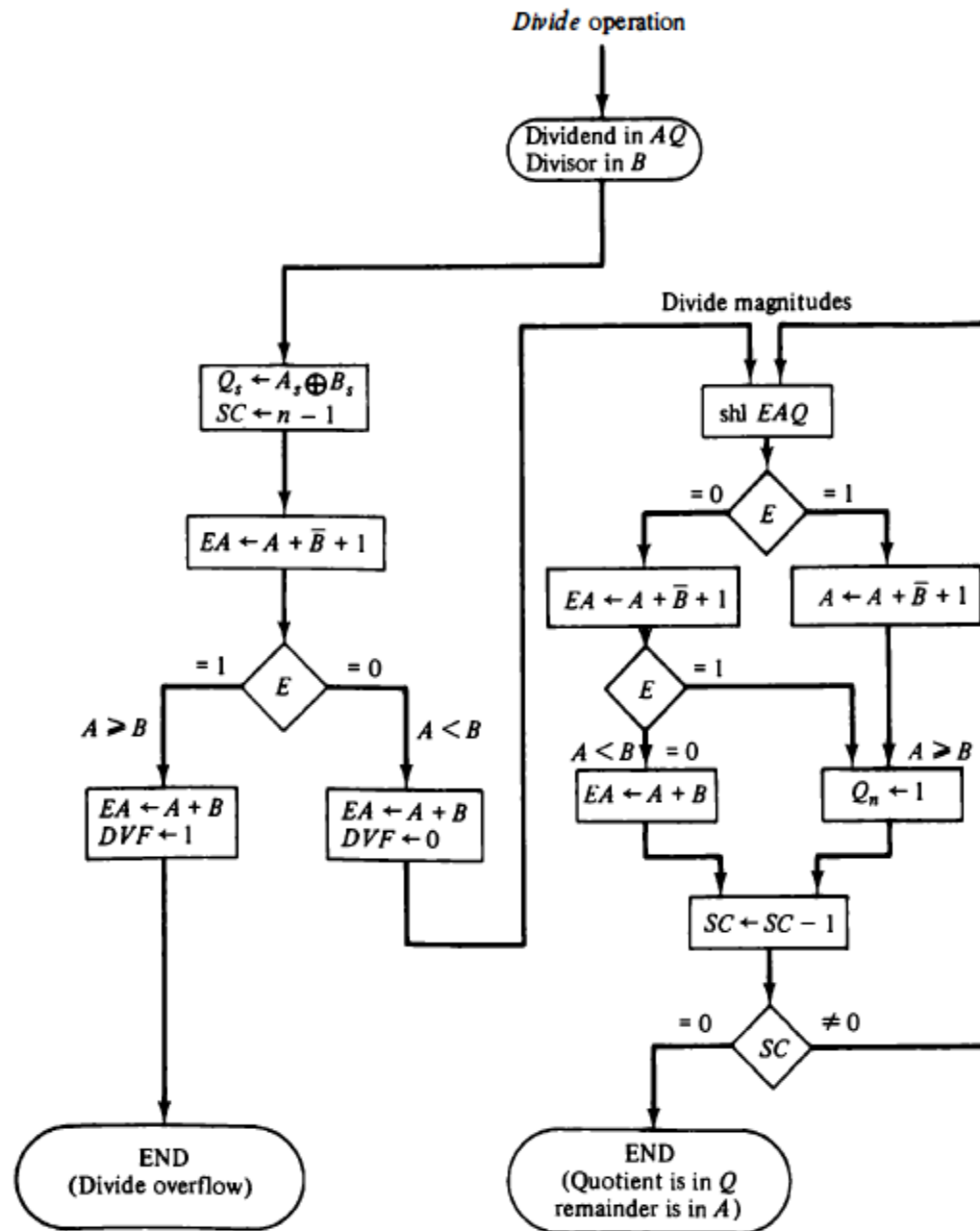
Divisor:	11010	Quotient = Q
$B = 10001$	$\overline{)0111000000}$	Dividend = A
	01110	5 bits of $A < B$, quotient has 5 bits
	011100	6 bits of $A \geq B$
	<u>-10001</u>	Shift right B and subtract; enter 1 in Q
	-010110	7 bits of remainder $\geq B$
	<u>--10001</u>	Shift right B and subtract; enter 1 in Q
	--001010	Remainder $< B$, enter 0 in Q , shift right B
	---010100	Remainder $\geq B$
	<u>----10001</u>	Shift right B and subtract; enter 1 in Q
	----000110	Remainder $< B$, enter 0 in Q
	-----00110	Final remainder

Example of binary division.

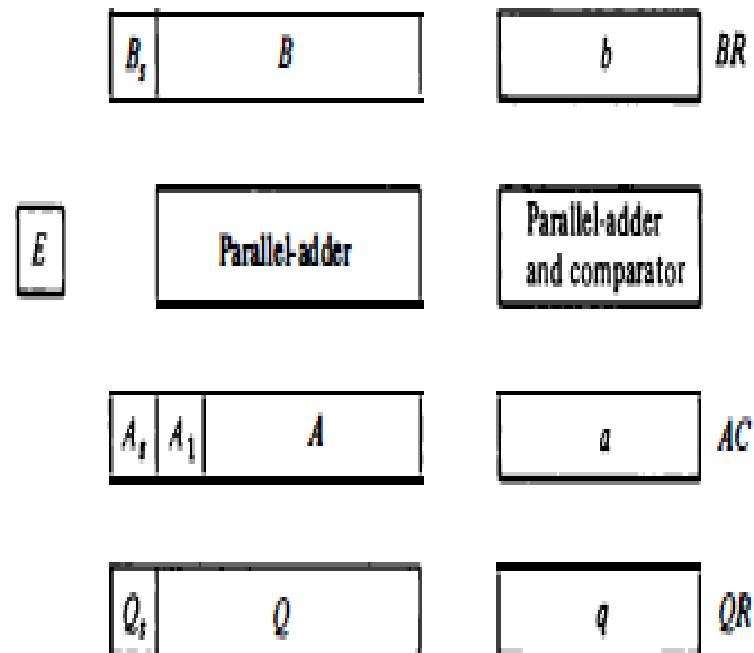
Divisor $B = 10001$,		$\overline{B} + 1 = 01111$		
	E	A	Q	SC
Dividend:		01110	00000	5
shl EAQ	0	11100	00000	
add $\overline{B} + 1$		<u>01111</u>		
$E = 1$	1	01011		
Set $Q_n = 1$	1	01011	00001	4
shl EAQ	0	10110	00010	
Add $\overline{B} + 1$		<u>01111</u>		
$E = 1$	1	00101		
Set $Q_n = 1$	1	00101	00011	3
shl EAQ	0	01010	00110	
Add $\overline{B} + 1$		<u>01111</u>		
$E = 0$; leave $Q_n = 0$	0	11001	00110	
Add B		<u>10001</u>		2
Restore remainder	1	01010		
shl EAQ	0	10100	01100	
Add $\overline{B} + 1$		<u>01111</u>		
$E = 1$	1	00011		
Set $Q_n = 1$	1	00011	01101	1
shl EAQ	0	00110	11010	
Add $\overline{B} + 1$		<u>01111</u>		
$E = 0$; leave $Q_n = 0$	0	10101	11010	
Add B		<u>10001</u>		
Restore remainder	1	00110	11010	0
Neglect E				
Remainder in A :		00110		
Quotient in Q :			11010	

Example of binary division with digital hardware.

Hardware Algorithm



Floating point number and operation



Registers for floating-point arithmetic operations.

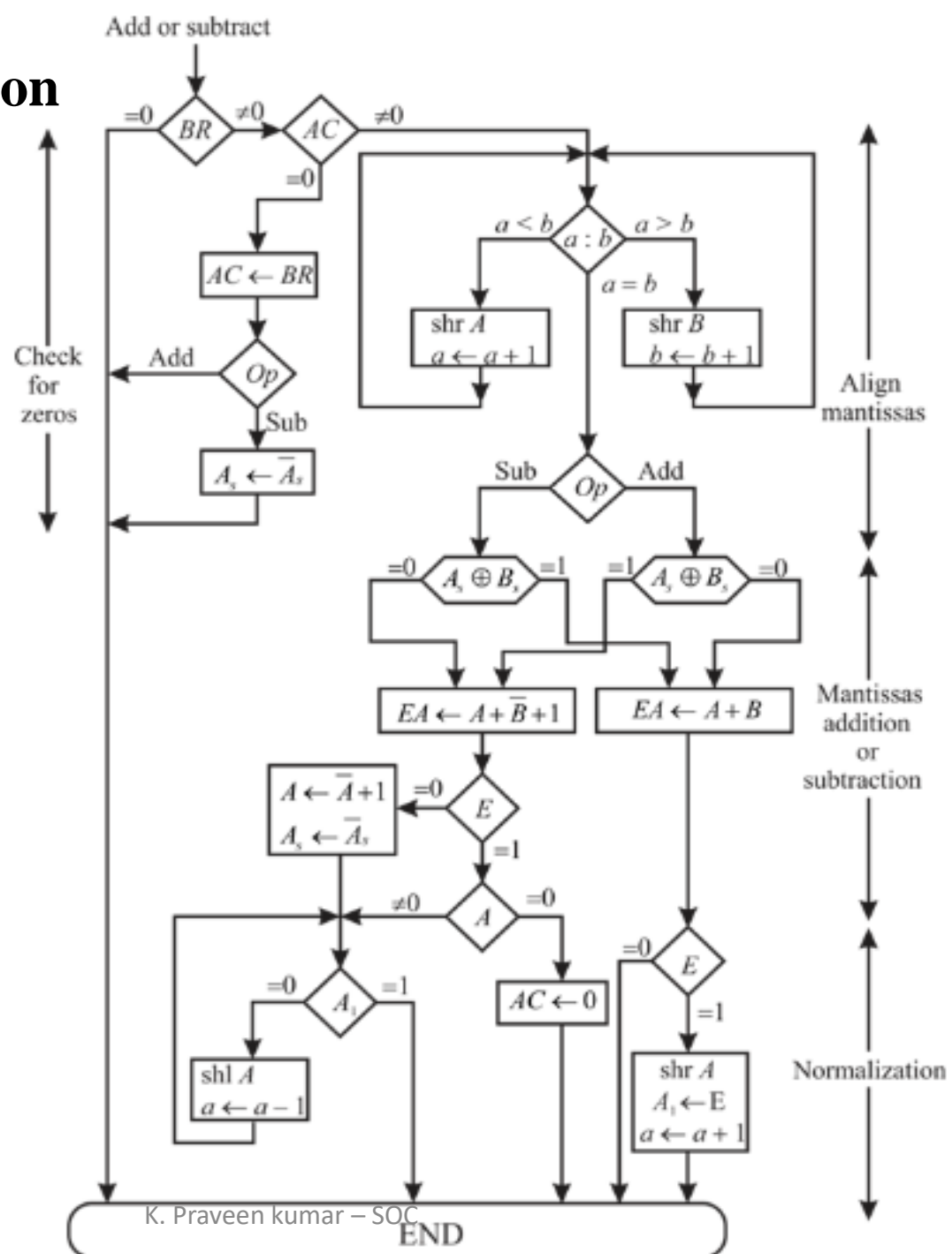
Addition and Subtraction

➤ During addition or subtraction, the two floating-point operands are in AC and BR .

➤ The sum or difference is formed in the AC . The algorithm can be divided into four consecutive parts:

1. Check for zeros.
2. Align the mantissas.
3. Add or subtract the mantissas.
4. Normalize the result.

Addition and subtraction of floating point numbers.



Multiplication

- The multiplication of two floating-point numbers requires that we multiply the mantissas and add the exponents.
- No comparison of exponents or alignment of mantissas is necessary.
- The multiplication of the mantissas is performed in the same way as in fixed-point to provide a double-precision product.
- The double-precision answer is used in fixed-point numbers to increase the accuracy of the product.

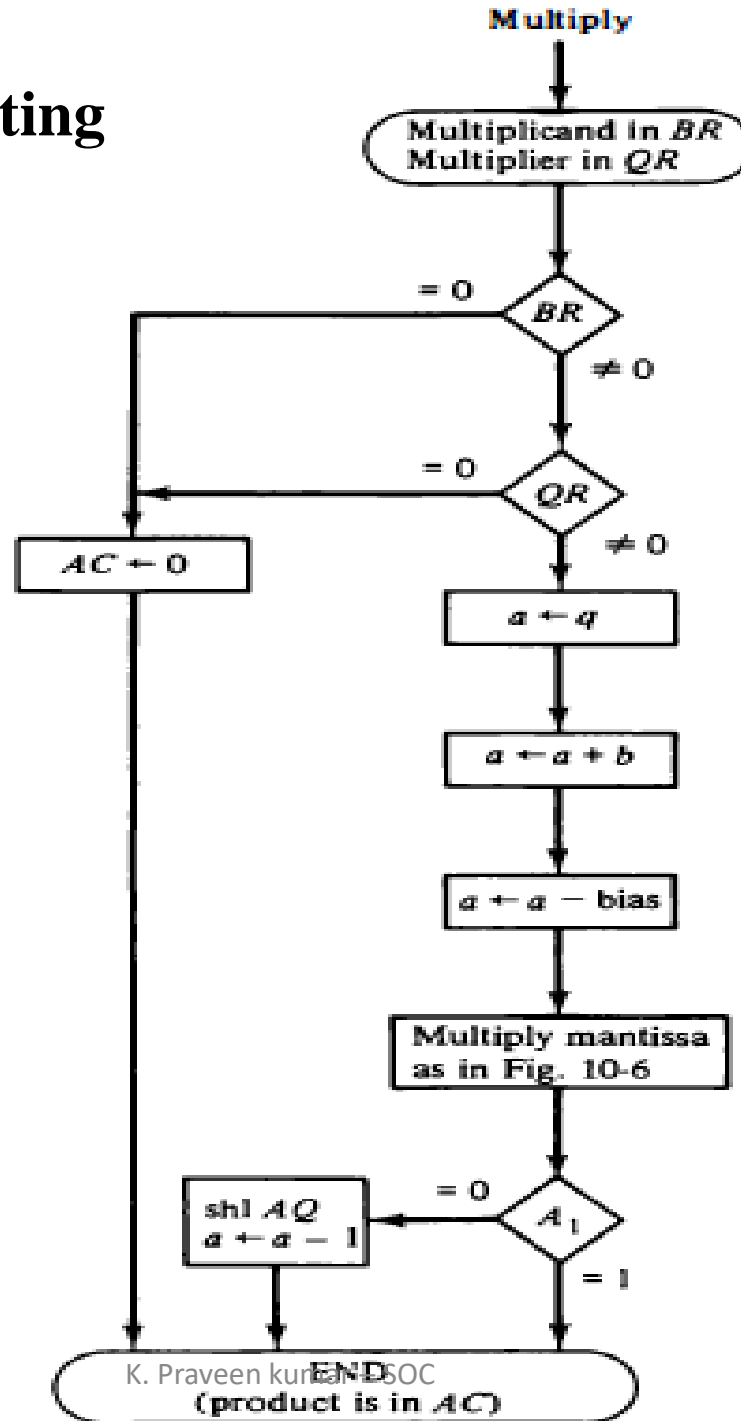
➤ In floating-point, the range of a single-precision mantissa combined with the exponent is usually accurate enough so that only single precision numbers are maintained.

➤ Thus the half most significant bits of the mantissa product and the exponent will be taken together to form a single precision floating-point product.

➤ The multiplication algorithm can be subdivided into four parts:

1. Check for zeros.
2. Add the exponents.
3. Multiply the mantissas.
4. Normalize the product.

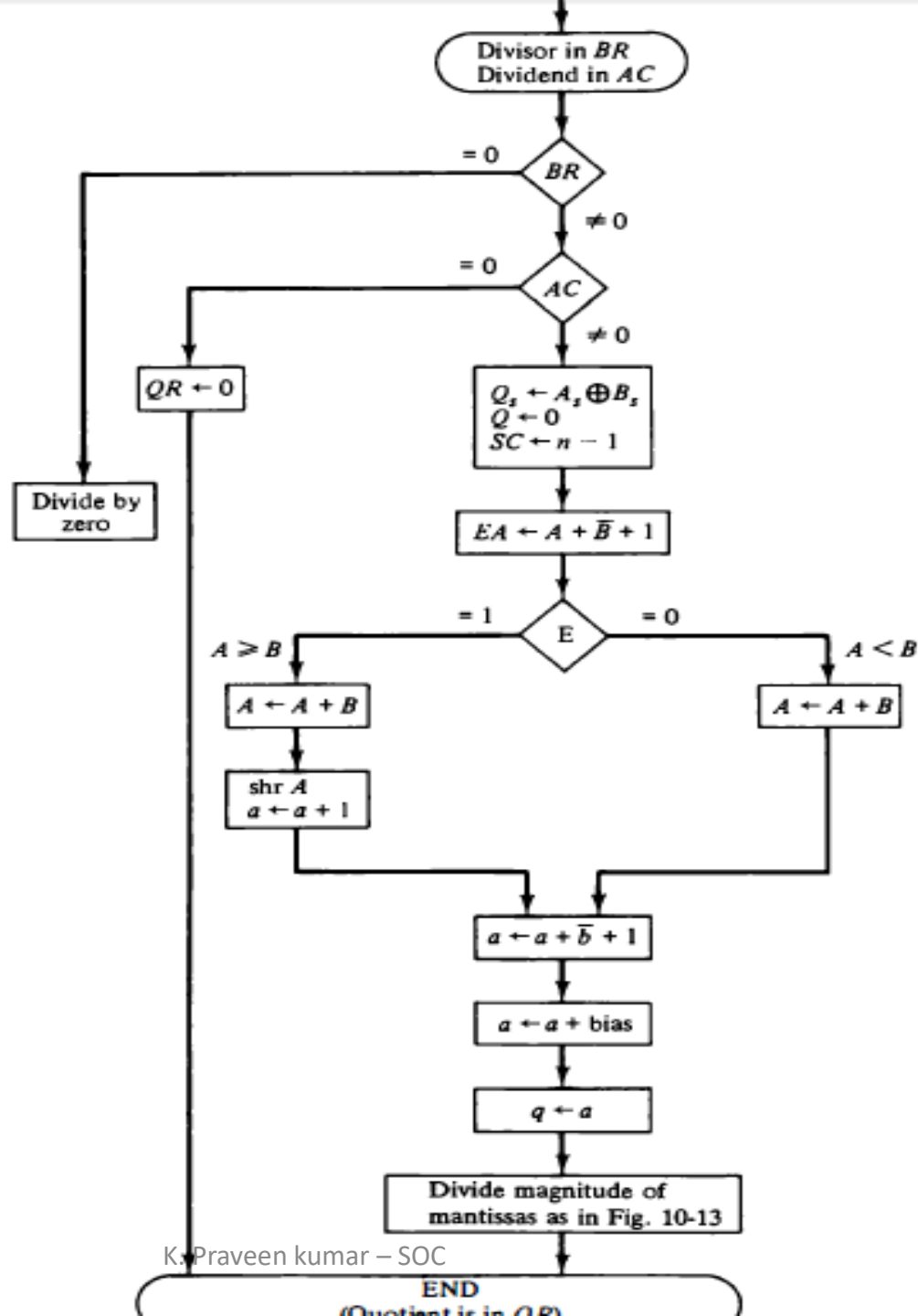
Multiplication of floating point numbers.



Division

- Floating-point division requires that the exponents be subtracted and the mantissas divided.
- The division algorithm can be subdivided into five parts:
 1. Check for zeros.
 2. Initialize registers and evaluate the sign.
 3. Align the dividend.
 4. Subtract the exponents.
 5. Divide the mantissas.

Division of floating point numbers.



IEEE 754 representations

➤ Normalized scientific notation: single non-zero digit to the left of the decimal (binary) point – example: 3.5×10^9

$$\text{➤ } 1.010001 \times 2^{-5}_{\text{two}} = (1 + 0 \times 2^{-1} + 1 \times 2^{-2} + \dots + 1 \times 2^{-6}) \times 2^{-5}_{\text{ten}}$$

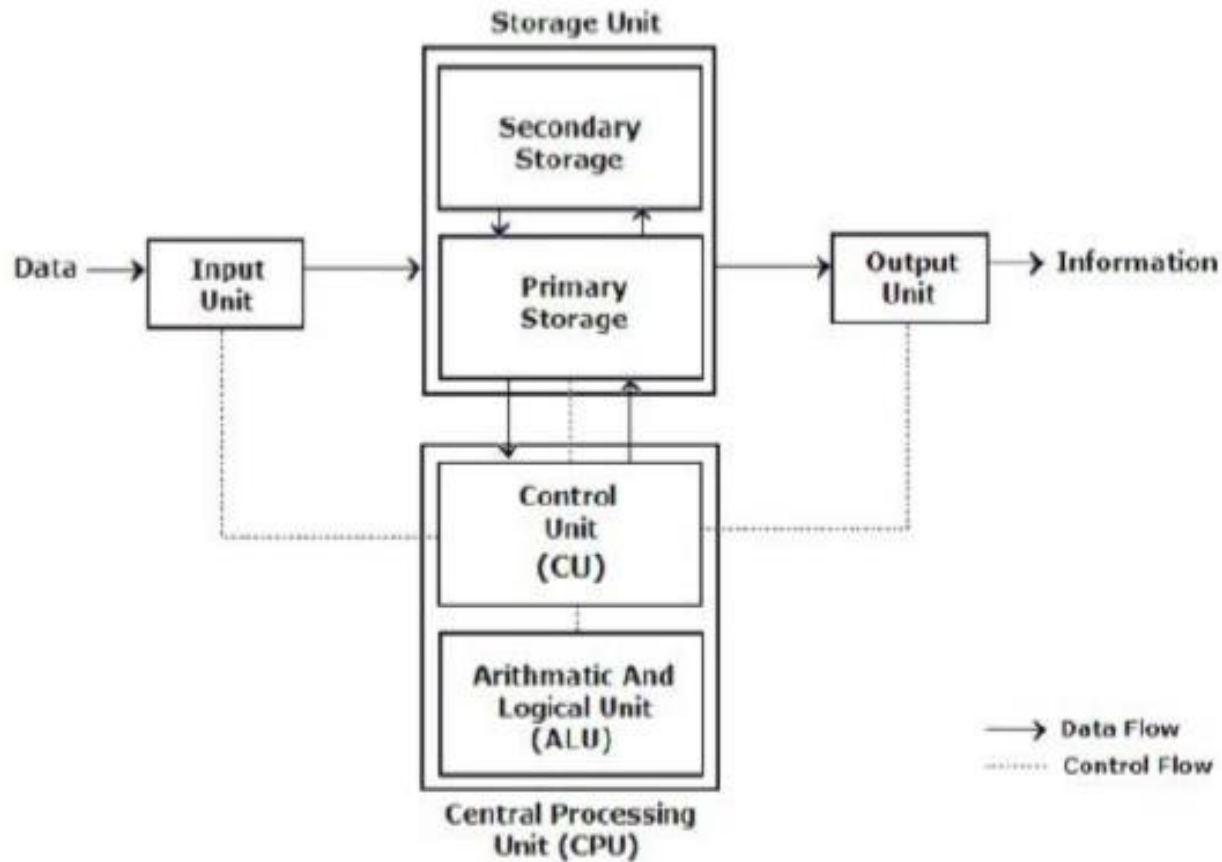
➤ A standard notation enables easy exchange of data between machines and simplifies hardware algorithms – the IEEE 754 standard defines how floating point numbers are represented



- More exponent bits → wider range of numbers (not necessarily more numbers – recall there are infinite real numbers)
- More fraction bits → higher precision
- Register value = $(-1)^S \times F \times 2^E$
- Since we are only representing normalized numbers, we are guaranteed that the number is of the form 1.xxxx.. Hence, in IEEE 754 standard, the 1 is implicit
Register value = $(-1)^S \times (1 + F) \times 2^E$

Functional blocks of a computer

Block diagram of computer



Input Unit:

- Computers need to receive data and instruction in order to solve any problem.
- The input unit consists of one or more input devices. Keyboard is the one of the most commonly used input device.
- Other commonly used input devices are the Mouse, Scanner, Microphone etc.
- All the input devices perform the following functions.
 - Accept the data and instructions from the outside world.
 - Convert it to a form that the computer can understand.
 - Supply the converted data to the computer system for further processing.

Storage Unit:

- The storage unit of the computer holds data and instructions that are entered through the input unit, before they are processed.
- It preserves the intermediate and final results before these are sent to the output devices.
- It also saves the data for the later use.
- The various storage devices of a computer system are divided into two categories.
 - a) **Primary storage**
 - b) **Secondary storage**

Primary Storage:

- Stores and provides very fast.
- This memory is generally used to hold the program being currently executed in the computer, the data being received from the input unit, the intermediate and final results of the program.
- The primary memory is temporary in nature.
- The data is lost, when the computer is switched off.
- In order to store the data permanently, the data has to be transferred to the secondary memory.
- The cost of the primary storage is more compared to the secondary storage. Therefore, most computers have limited primary storage capacity.

Secondary Storage:

- Secondary storage is used like an archive.
- It stores several programs, documents, data bases etc.
- The programs that you run on the computer are first transferred to the primary memory before it is actually run. Whenever the results are saved, again they get stored in the secondary memory.
- The secondary memory is slower and cheaper than the primary memory.
- Some of the commonly used secondary memory devices are Hard disk, CD, etc.,

Central Processing Unit:

➤ The Control Unit (CU) and Arithmetic Logic Unit (ALU) of the computer are together known as the Central Processing Unit (CPU).

➤ The CPU is like brain performs the following functions:

- It performs all calculations.
- It takes all decisions.
- It controls all units of the computer.

Arithmetic Logical Unit:

- All calculations are performed in the Arithmetic Logic Unit (ALU) of the computer.
- It also does comparison and takes decision.
- The ALU can perform basic operations such as addition, subtraction, multiplication, division, etc and does logic operations viz, $>$, $<$, $=$, 'etc.
- Whenever calculations are required, the control unit transfers the data from storage unit to ALU once the computations are done, the results are transferred to the storage unit by the control unit and then it is send to the output unit for displaying results.

Control Unit:

- It controls all other units in the computer.
- The control unit instructs the input unit, where to store the data after receiving it from the user.
- It controls the flow of data and instructions from the storage unit to ALU.
- It also controls the flow of results from the ALU to the storage unit.
- The control unit is generally referred as the central nervous system of the computer that control and synchronizes its working.

Output Unit:

➤ The output unit of a computer provides the information and results of a computation to outside world. Printers, Visual Display Unit (VDU) are the commonly used output devices.

➤ Other commonly used output devices are Speaker, Headphone, Projector etc.