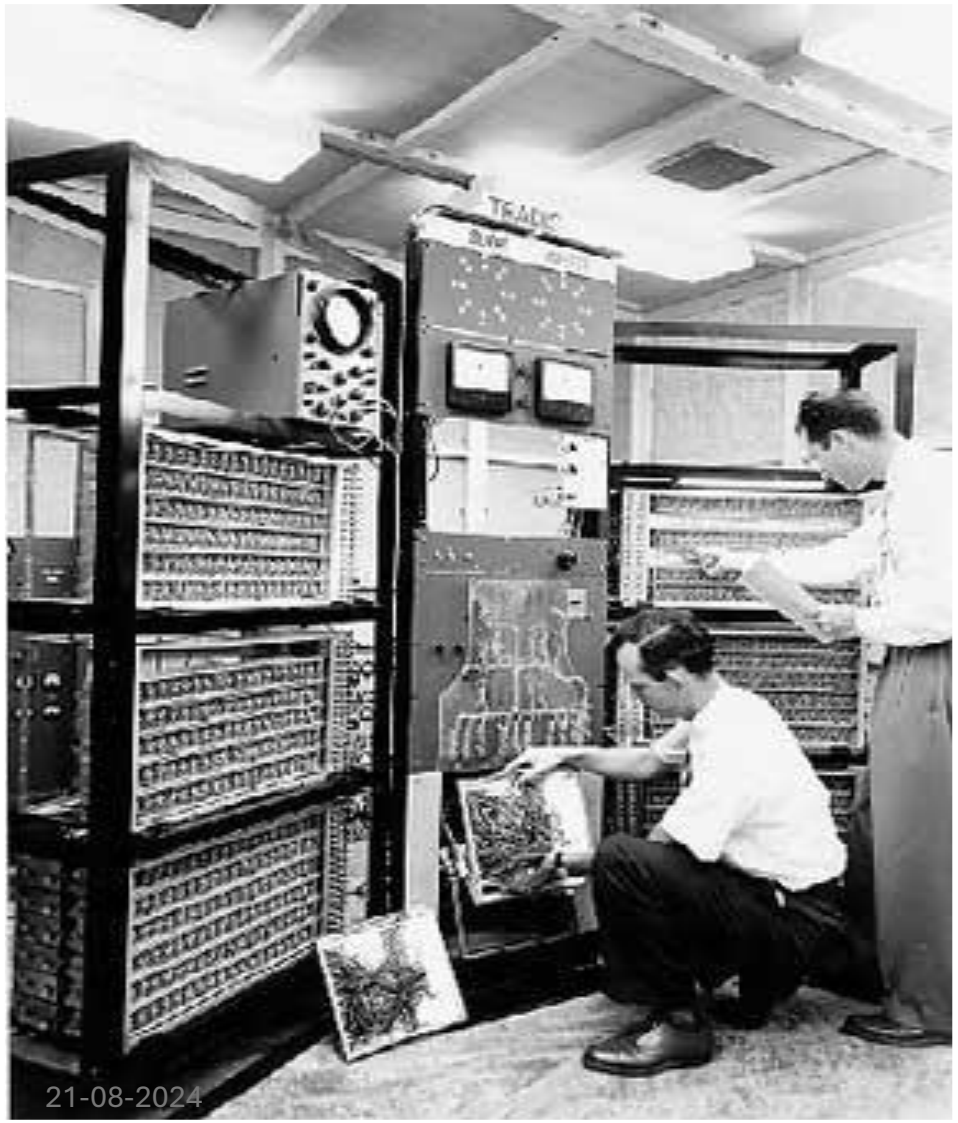
UNIT-1

- Historical Background
- Functional Units of a computer
- Basic Operational Concepts
- Computer Arithmetic (ADD, SUB, MUL, DIV)
- Combination Circuit
- Sequential circuit

GENERATIONS OF A COMPUTER

S.N.	Generation & Description
1	**First Generation** The period of first generation: 1946-1959. Vacuum tube-based.
2	**Second Generation** The period of second generation: 1959-1965. Transistor based.
3	**Third Generation** The period of third generation: 1965-1971. Integrated Circuit based.
4	**Fourth Generation** The period of fourth generation: 1971-1980. VLSI microprocessor based.
5	**Fifth Generation** The period of fifth generation: 1980-onwards. ULSI microprocessor based

First generation



- The computers of first generation used vacuum tubes as the basic components for memory and circuitry for CPU (Central Processing Unit).
- These tubes, like electric bulbs, produced a lot of heat and were prone to frequent fusing of the installations, therefore, were very expensive and could be afforded only by very large organizations.
- The computers in this generation used machine code as programming language.

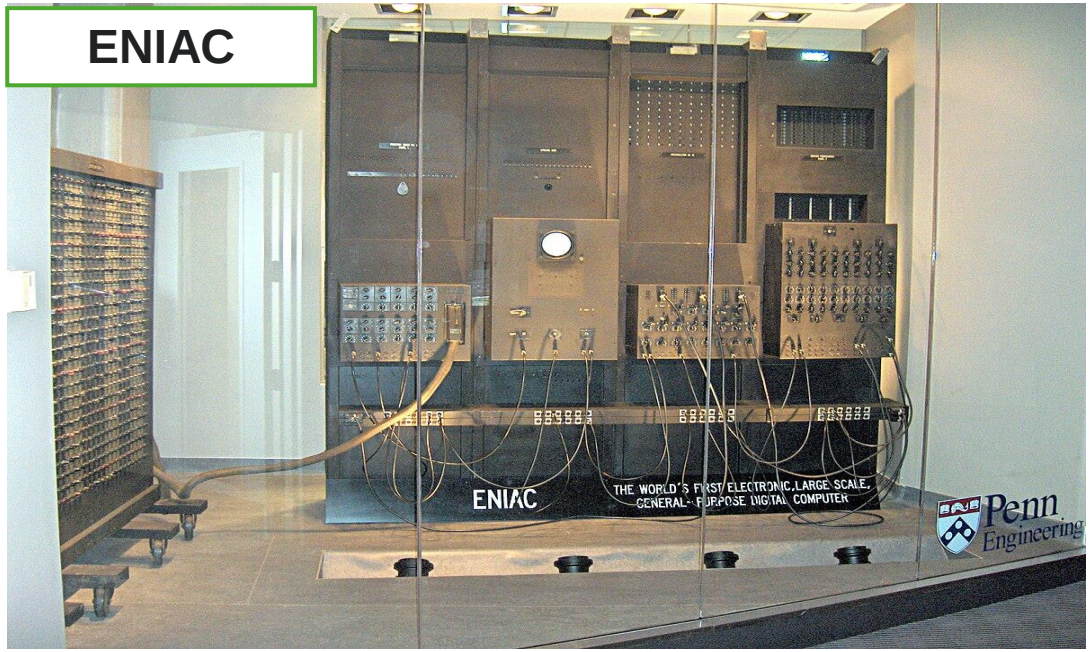
The main features of first generation are:

- Vacuum tube technology
- Unreliable
- Supported machine language only
- Very costly
- Generated lot of heat
- Slow input and output devices
- Huge size
- Non-portable
- Consumed lot of electricity

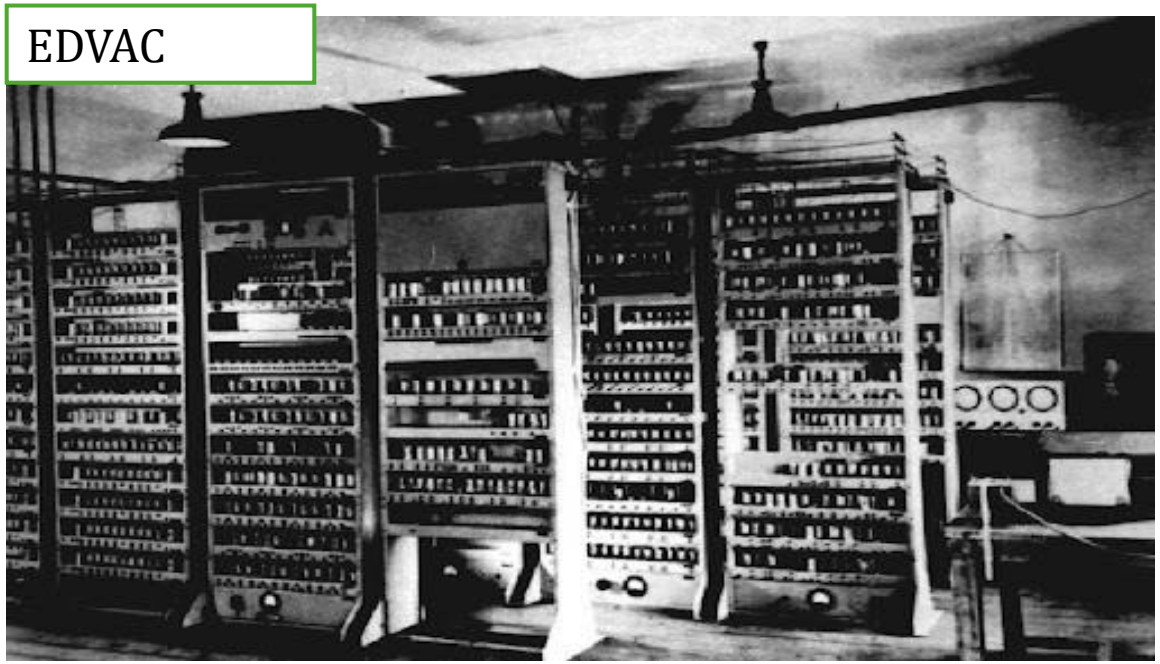
Some computers of this generation were:

- ENIAC
- EDVAC
- UNIVAC
- IBM-701

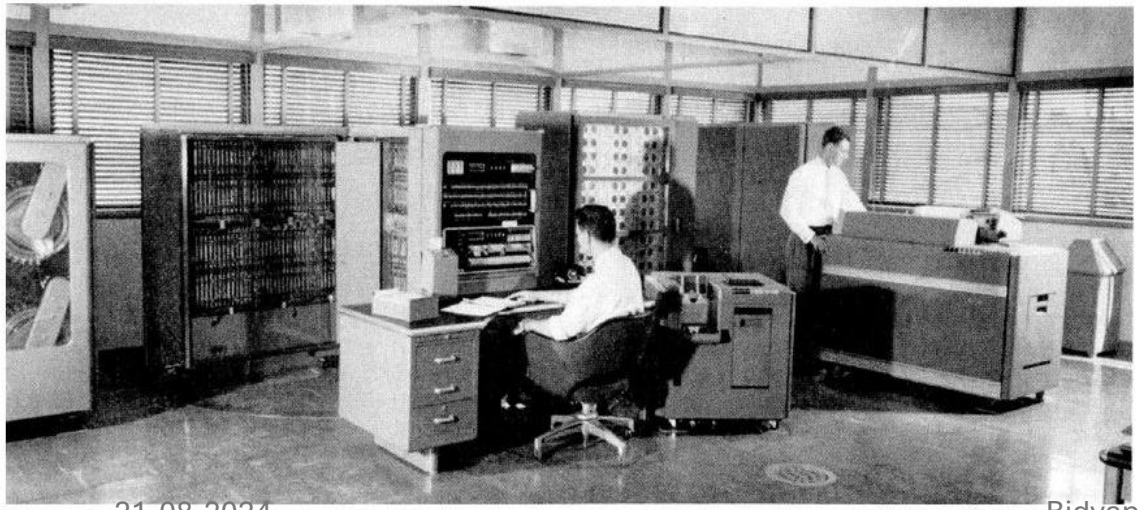
ENIAC



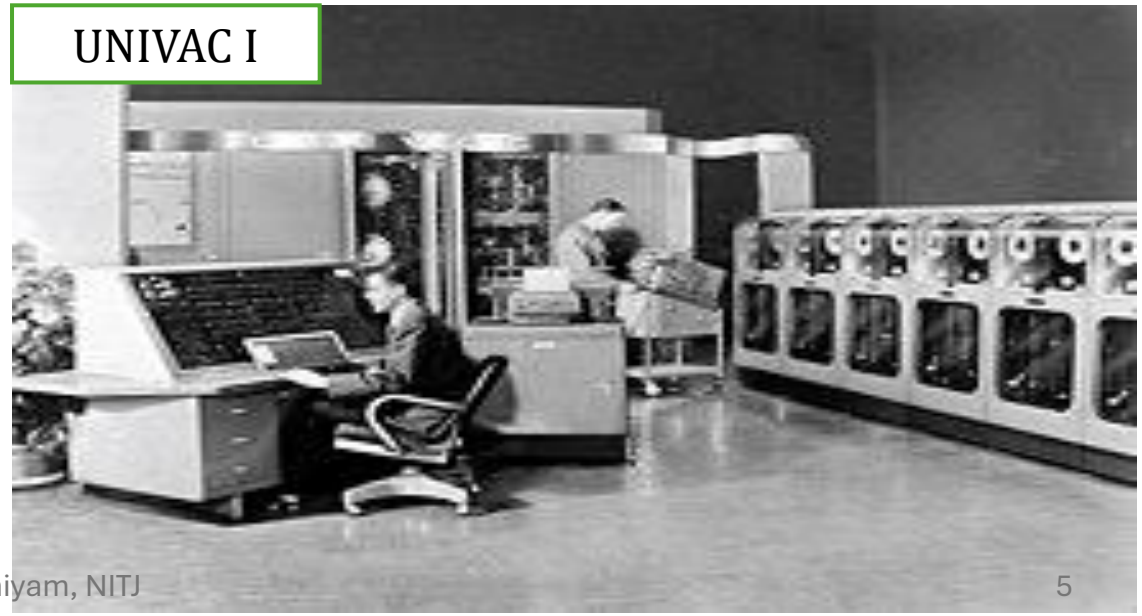
EDVAC



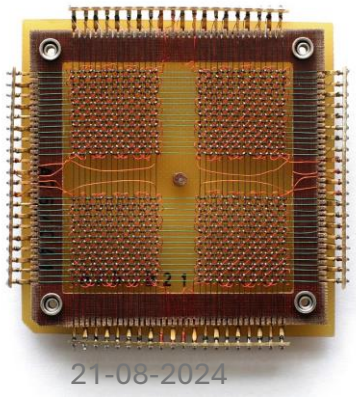
IBM 701 COMPUTER

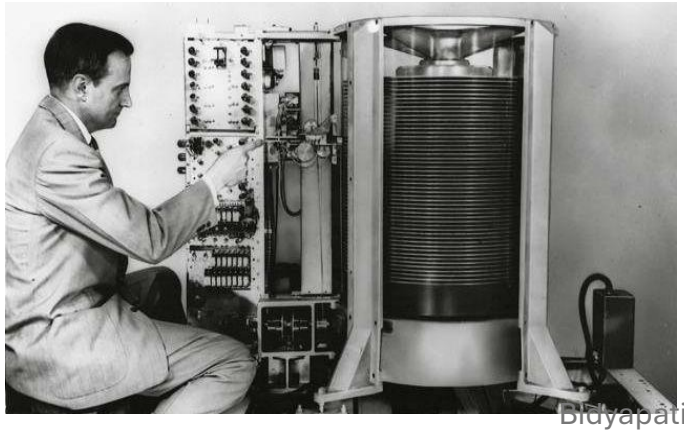


UNIVAC I



Second generation





- In this generation transistors were used that were cheaper, consumed less power, more compact in size, more reliable and faster than the first generation machines made of vacuum tubes.
- magnetic cores were used as primary memory and magnetic tape and magnetic disks as secondary storage devices
- In this generation assembly language and high-level programming languages like FORTRAN, COBOL were used.
- The computers used batch processing and multiprogramming operating system

The main features of second generation are:

- Use of transistors
- Reliable in comparison to first generation computers
- Smaller size as compared to first generation computers
- Generated less heat as compared to first generation computers
- Consumed less electricity as compared to first generation computers
- Faster than first generation computers
- Still very costly
- Supported machine and assembly languages

Some computers of this generation were:

- IBM 1620
- IBM 7094
- CDC 1604
- CDC 3600

IBM 1620

IBM 7094

CDC 1604

CDC 3600

Third generation

- The computers of third generation used integrated circuits (IC's) in place of transistors.
- A single IC has many transistors, resistors and capacitors along with the associated circuitry.
- The IC was invented by Jack Kilby.
- This development made computers smaller in size, reliable and efficient.
- In this generation remote processing, time-sharing, multi-programming operating system were used.
- High-level languages (FORTRAN-II TO IV, COBOL, PASCAL PL/1, BASIC, ALGOL-68 etc.) were used during this generation.

The main features of third generation are:

- IC used
- More reliable in comparison to previous two generations
- Smaller size
- Generated less heat
- Faster
- Lesser maintenance
- Still costly
- Consumed lesser electricity
- Supported high-level language

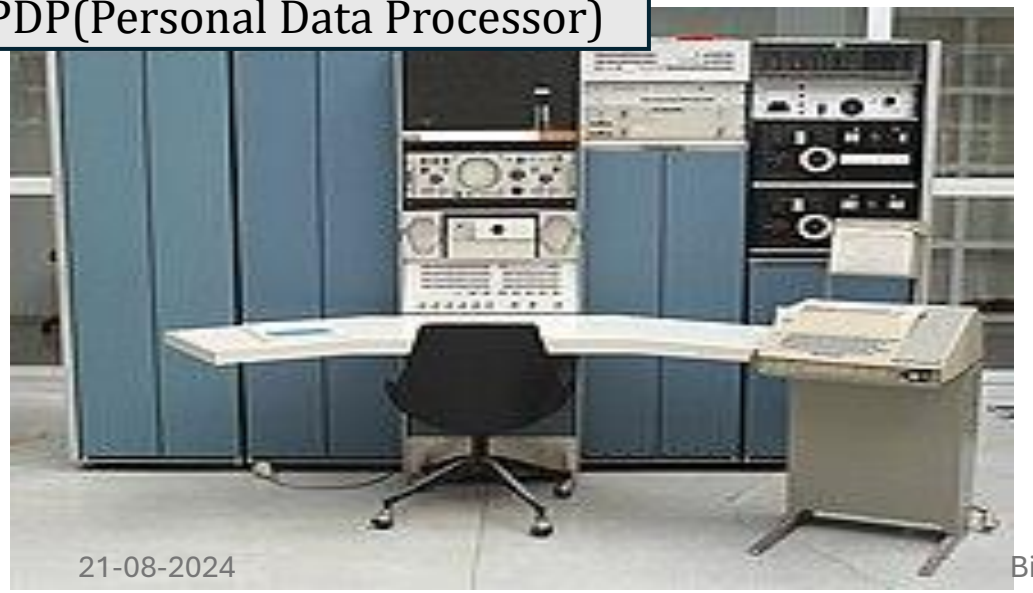
Some computers of this generation were:

- IBM-360 series
- Honeywell-6000 series
- PDP(Personal Data Processor)
- IBM-370/168

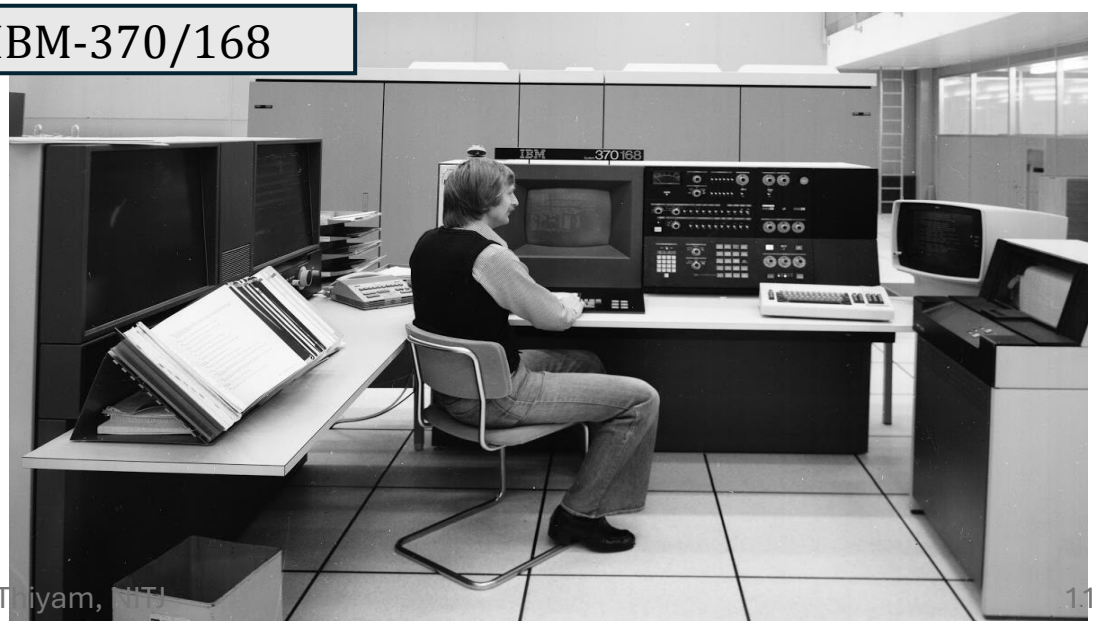
IBM-360 series

Honeywell-6000 series

PDP(Personal Data Processor)



IBM-370/168



Fourth generation

- The computers of fourth generation used Very Large Scale Integrated (VLSI) circuits.
- VLSI circuits having about 5000 transistors and other circuit elements and their associated circuits on a single chip
- Fourth generation computers became more powerful, compact, reliable, and affordable. As a result, it gave rise to personal computer (PC) revolution .
- In this generation time sharing, real time, networks, distributed operating system were used.
- All the high-level languages like C, C++, DBASE etc., were used in this generation.

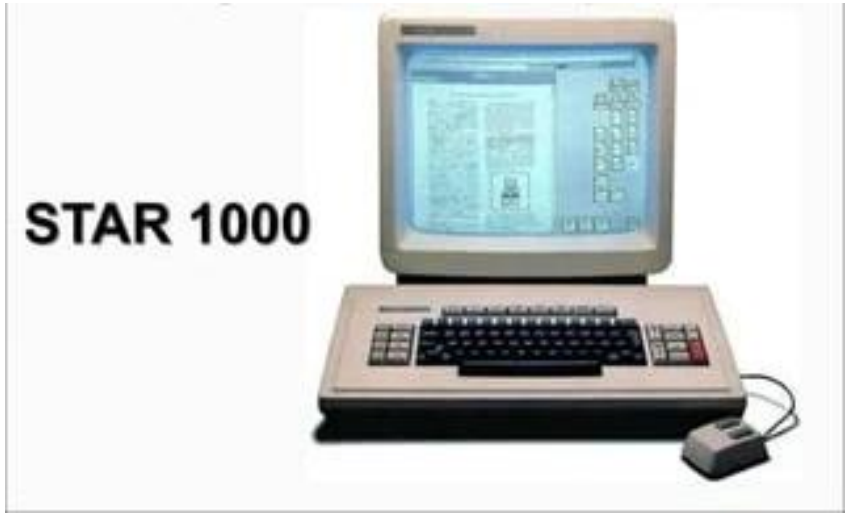
The main features of fourth generation are:

- VLSI technology used
- Very cheap
- Portable and reliable
- Use of PC's
- Very small size
- Pipeline processing
- Concept of internet was introduced
- Great developments in the fields of networks
- Computers became easily available

Some computers of this generation were:

- DEC 10
- STAR 1000
- PDP 11
- CRAY-1(Super Computer)

DEC 10



PDP 11

CRAY-1(Super Computer)

Fifth generation

- The period of fifth generation is 1980-till date.
- In the fifth generation, the VLSI technology became ULSI (Ultra Large Scale Integration) technology, resulting in the production of microprocessor chips having ten million electronic components.
- This generation is based on parallel processing hardware and AI (Artificial Intelligence) software.
- AI is an emerging branch in computer science, which interprets means and method of making computers think like human beings.
- All the high-level languages like C and C++, Java, .Net etc., are used in this generation.

The main features of fifth generation are:

- ULSI technology
- Development of true artificial intelligence
- Advancement in Superconductor technology
- More user-friendly interfaces with multimedia features
- Availability of very powerful and compact computers at cheaper rates

Some computer types of this generation are:

- Desktop
- Laptop
- NoteBook
- UltraBook

Fundamental units

A computer consists of five functionally independent main parts input, memory, arithmetic logic unit (ALU), output and control unit.

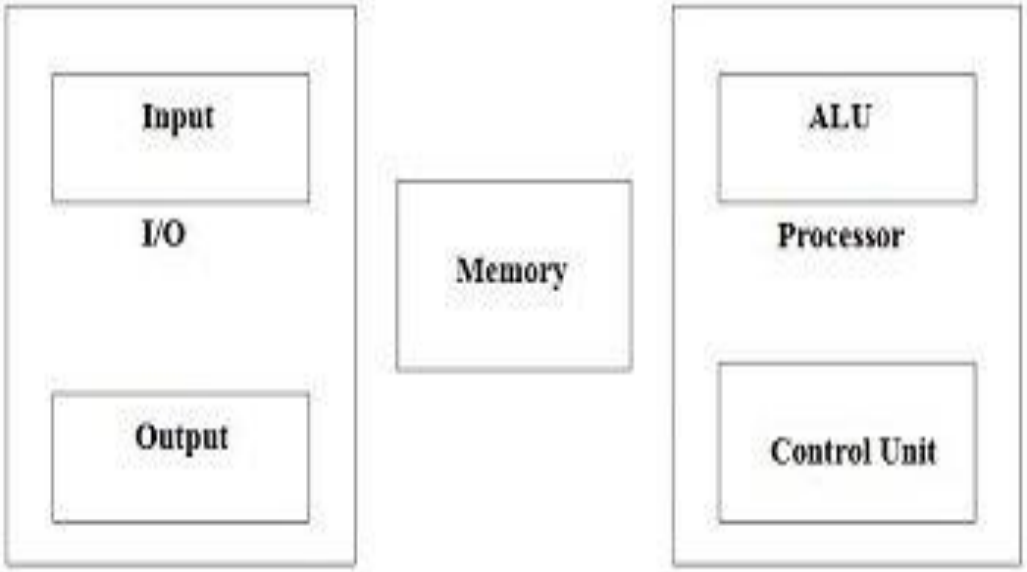


Figure1.1: Basic Functional units of computer

1. Input unit:

- The source program/high level language program/coded information/simply data is fed to a computer through input devices (keyboard is a most common type).
- Whenever a key is pressed, one corresponding word or number is translated into its equivalent binary code over a cable & fed either to memory or processor.
- Eg : Joysticks, trackballs, mouse, scanners etc are other input devices.

2. Memory:

- Its function into store programs and data. It is basically to two types
- 1. Primary memory :
  - fast memory that operates as electronic speeds.

    - Unit of word (n bits) is organized

    - Address to access the word easily

- Word length of the computer = The number of bits in each word

- Memory in which any location can be reached in a short and fixed amount of time after specifying its address is called random-access memory (RAM)

- The time required to access one word in called memory access time.

- Memory which is only readable by the user and contents of which can't be altered is called read only memory (ROM) it contains operating system

*Cache- small and fast, tightly coupled with processor and contained on the same integrated circuit chip to achieve high performance.

2. Secondary memory:

- Is used where large amounts of data & programs have to be stored, particularly information that is accessed infrequently.
- Examples: - Magnetic disks & tapes, optical disks (ie CD-ROM's), floppies etc.

Primary memory	Secondary memory
Primary memory is temporary.	Secondary memory is permanent.
Primary memory is directly accessible by Processor/CPU.	Secondary memory is not directly accessible by the CPU
RAM- volatile in nature.	It’s always Non-volatile in nature.
Primary memory devices are more expensive	Secondary memory devices are less expensive
The memory devices used for primary memory are semiconductor memories.	The secondary memory devices are magnetic and optical memories.
Primary memory is also known as Main memory or Internal memory.	Secondary memory is also known as External memory or Auxiliary memory
Examples: RAM, ROM, Cache memory, PROM, EPROM, Registers, etc.	Examples: Hard Disk, Floppy Disk, Magnetic Tapes, etc.

3. Arithmetic and Logic Unit (ALU):

- ALU->performs operations like addition, subtraction, division, multiplication, etc.
- Data (operands) are brought into the ALU from memory and stored in high speed storage elements called register.
- Each register can store one word of data. Its speed is faster than cache
- The control and the ALU are may times faster than other devices connected to a computer system
- This enables a single processor to control a number of external devices such as key boards, displays, magnetic and optical disks, sensors and other mechanical controllers

4. Output unit:

- Its basic function is to send the processed results to the outside world.
- Examples:- Printer, speakers, monitor etc.

5. Control unit:

- It effectively is the nerve center that sends signals to other units and senses their states.
- The actual timing signals that govern the transfer of data between input unit, processor, memory and output unit are generated by the control unit.

Basic operational concepts

- An instruction consists of two parts
    1. operation code (OPCODE)
    2. operands

- Reserved symbols
  Eg. ADD, SUB, LD
- It is always 8 bit

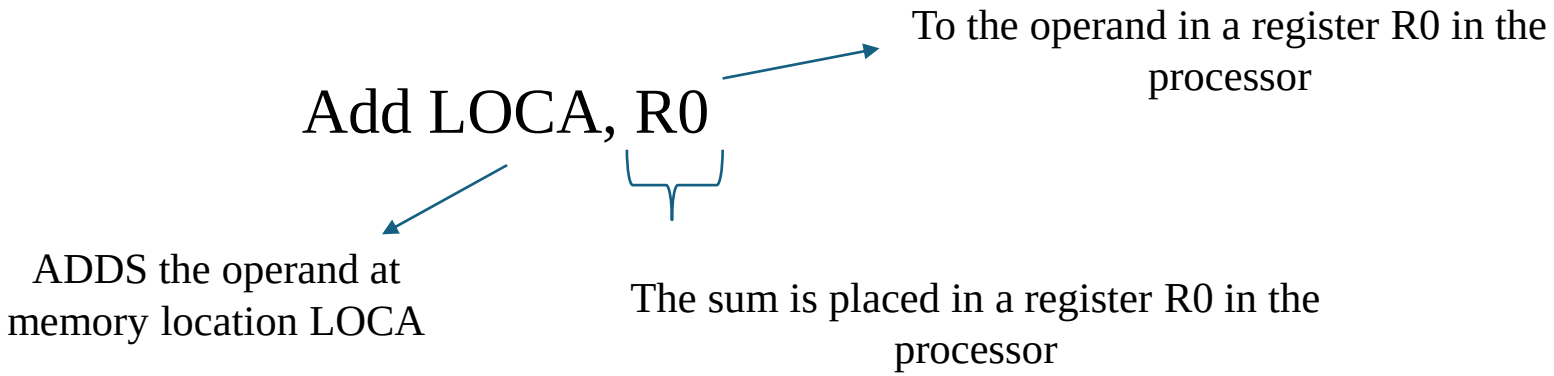
- Consists of Register, Number, Label that are separated by comma
  Eg. ADD R1, R2, R3
  LD R6, NUMBER
- It can be 8 bit or 16 bit

- Data/ operands are stored in memory
- Individual instructions are brought from memory to the processor.

Add LOCA, R0

- This instruction requires the performance of several steps:

1. First the instruction is fetched from the memory into the processor.
2. The operand at LOCA is fetched and added to the contents of R0
3. Finally the resulting sum is stored in the register R0



*operand at LOCA is preserved. However, the operand is register R0 is overwritten.

Addition and Subtraction

Basic Idea

- Hardware can only deal with binary digits, 0 and 1.
- Must represent all numbers, integers or floating point, positive or negative, by binary digits, called *bits*.
- Devise electronic circuits to perform arithmetic operations (add, subtract, multiply and divide) on binary numbers.

Positive Integers

- Decimal system: made of 10 digits, {0,1,2, . . . , 9}

$$41 = 4\times10^1 + 1\times10^0$$

$$255 = 2\times10^2 + 5\times10^1 + 5\times10^0$$

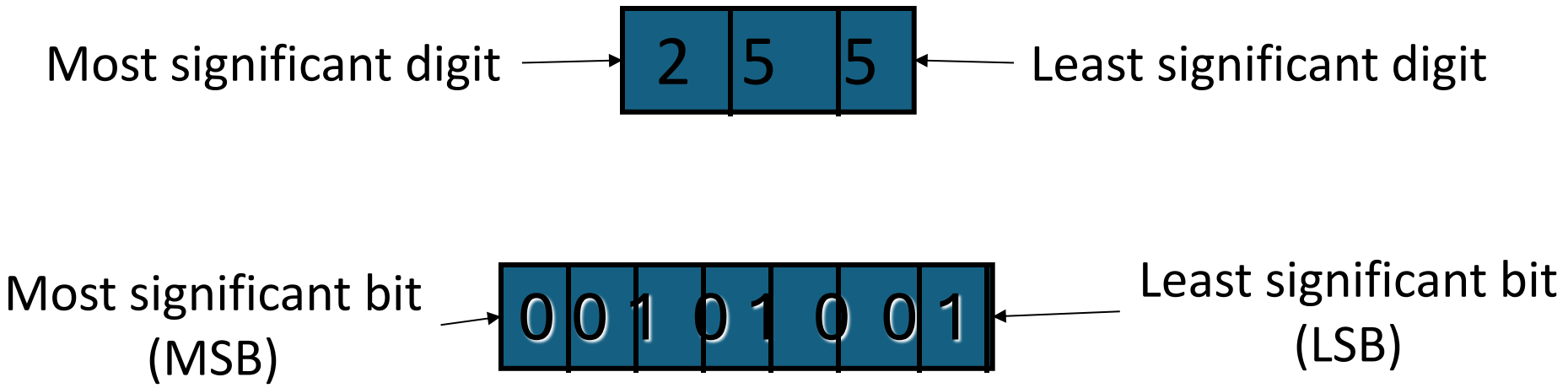
- Binary system: made of two digits, {0,1}

$$00101001 = 0\times2^7 + 0\times2^6 + 1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$$

$$= 32 + 8 + 1 = 41$$

$11111111 = 255$, largest number with 8 binary digits, 2^8-1

Writing Numbers



Base or Radix

- For decimal system, 10 is called the base or radix.
- Decimal 41 is also written as 41_{10} or 41_{ten}
- Base (radix) for binary system is 2.
- Thus, $41_{ten} = 101001_2$ or 101001_{two}
  $111_{two} = 7_{ten}$

Signed Integers

- Use left-most bit (called most significant bit or MSB) for sign:

  0 for positive

  1 for negative

- Example: $+18_{ten} = 00010010_{two}$

  $-18_{ten} = 10010010_{two}$

Why Not Use Sign Bit

- Sign and magnitude bits should be differently treated in arithmetic operations.
- Addition and subtraction require different logic circuits.
- Overflow is difficult to detect.
- "Zero" has two representations:

$$+0_{ten} = 00000000_{two}$$
$$-0_{ten} = 10000000_{two}$$

- Signed-integers are not used in modern computers.

Integers With Sign – Other Ways

- Use fixed-length representation, but no sign bit
  - 1's complement: To form a negative number, complement each bit in the given number.
  - 2's complement: To form a negative number, start with the given number, subtract one, and then complement each bit, *or*

    first complement each bit, and then add 1.
- 2's complement is the most preferred representation.

1’s-Complement

- To change the sign of a binary integer simply complement (invert) each bit.
- Example: 3 = 0011, -3 = 1100
- n-bit representation: Negation is equivalent to subtraction from $2^n – 1$

2’s-Complement

- Add 1 to 1’s-complement representation.
- Some properties:
    - **Only one representation for 0**
    - **Exactly as many positive numbers as negative numbers**
    - **Slight asymmetry – there is one negative number with no positive counterpart**

Three Representations

Sign-magnitude	1's complement	2's complement
000 = +0	000 = +0	000 = +0
001 = +1	001 = +1	001 = +1
010 = +2	010 = +2	010 = +2
011 = +3	011 = +3	011 = +3
100 = - 0	100 = - 3	100 = - 4
101 = - 1	101 = - 2	101 = - 3
110 = - 2	110 = - 1	110 = - 2
111 = - 3	111 = - 0	111 = - 1
		(Preferred)

Binary Multiplication

- Unsigned Integer Multiplication
- Signed Integer Multiplication
- Faster Integer Multiplication

Unsigned Integer Multiplication

- Paper and Pencil Example:

Multiplicand 1 1 0 0 = 12

Multiplier × 1 1 0 1 = 13

```
        1  1 0  0
+     0  0  0  0
    1  1  0  0
  1  1  0  0
-------------------
```

Product 1 0 0 1 1 1 0 0 = 156

Binary multiplication is easy

0 × multiplicand = 0

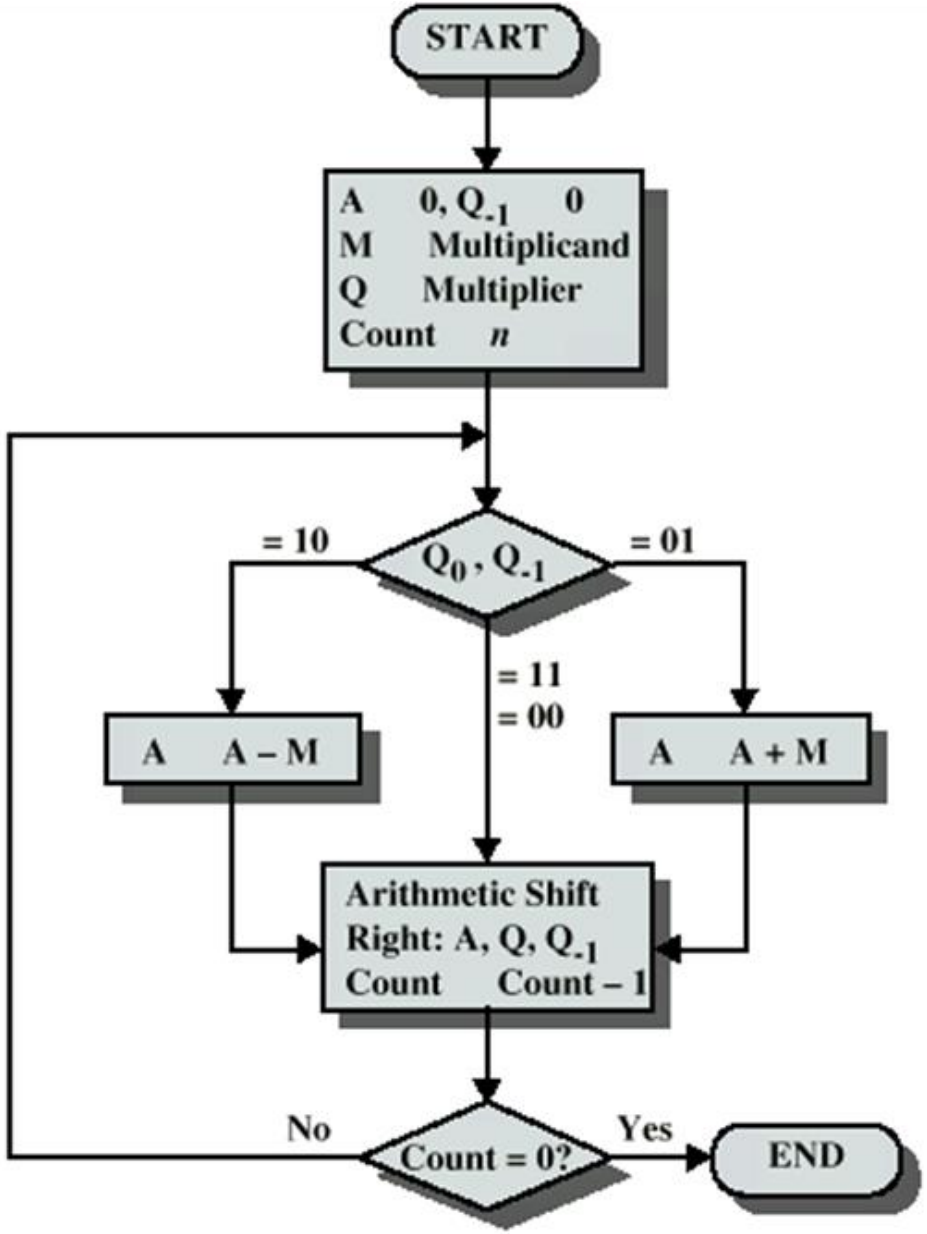
1 × multiplicand = multiplicand

- m-bit multiplicand × n-bit multiplier = (m+n)-bit product
- Accomplished via shifting and addition
- Consumes more time and more chip area than addition

Signed Integer Multiplication

- Booth's algorithm is a method for multiplying two signed binary numbers. It uses a technique called "bit shifting" to quickly multiply the numbers while taking into account their signs.
- The algorithm works by repeatedly shifting the multiplicand and adding or subtracting it from the partial product based on the value of the multiplier's bits.
- This process continues until all bits of the multiplier have been processed, resulting in the final product.
- Booth's algorithm is faster than traditional multiplication methods and is often used in computer hardware and software.

Booth Algorithm:



Booth's algorithm for two complements multiplication:

1. Multiplier and multiplicand are placed in the Q and M register respectively.
2. Result for this will be stored in the AC and Q registers.
3. Initially, AC and Q_{-1} register will be 0.
4. Multiplication of a number is done in a cycle.
5. A 1-bit register Q_{-1} is placed right of the least significant bit Q_0 of the register Q.
6. In each of the cycle, Q_0 and Q_{-1} bits will be checked.
    i. If Q_0 and Q_{-1} are 11 or 00 then the bits of AC, Q and Q_{-1} are shifted to the right by 1 bit.
    ii. If the value is shown 01 then multiplicand is added to AC. After addition, AC, Q_0, Q_{-1} register are shifted to the right by 1 bit.
    iii. If the value is shown 10 then multiplicand is subtracted from AC. After subtraction AC, Q_0, Q_{-1} register is shifted to the right by 1 bit.

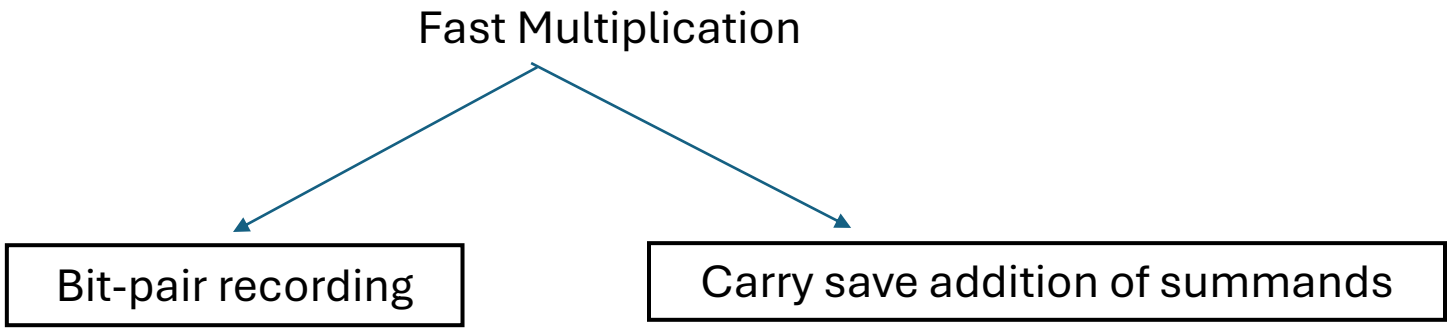
	7	(0 1 1 1)				
	x 3	(0 0 1 1)				
	A	**Q**	**Q_{-1}**	**M**		
Initial values	**0000**	**0011**	**0**	**0111**		
	1001	**0011**	**0**	**0111**	A = A - M	} 1
	1100	**1001**	**1**	**0111**	Shift	
	1110	**0100**	**1**	**0111**	Shift	} 2
	0101	**0100**	**1**	**0111**	A = A + M	} 3
	0010	**1010**	**0**	**0111**	Shift	
	0001	**0101**	**0**	**0111**	Shift	} 4

Exercise: Multiply the (17) with (13) using Booth's algorithm. Give each step.

Exercise: Multiply the (15) with (-13) using Booth's algorithm. Give each step.

Multiplier		Version of multiplicand selected by bit i
Bit i	Bit $i-1$	
0	0	$0 \times M$
0	1	$+1 \times M$
1	0	$-1 \times M$
1	1	$0 \times M$

Figure 6.12 Booth multiplier recoding table.



Bit-pair recording

Multiplier bit-pair		Multiplier bit on the right	Multiplicand selected at position i
$i+1$	i	$i-1$	
0	0	0	$0 \times M$
0	0	1	$+1 \times M$
0	1	0	$+1 \times M$
0	1	1	$+2 \times M$
1	0	0	$-2 \times M$
1	0	1	$-1 \times M$
1	1	0	$-1 \times M$
1	1	1	$0 \times M$

Example

$$\begin{array}{rrrrrl} & 0 & 1 & 1 & 0 & 1 \quad (+13) \\ \times & 1 & 1 & 0 & 1 & 0 \quad (-6) \end{array}$$

$$\begin{array}{rrrrrrrrrrl} & & & & & 0 & 1 & 1 & 0 & 1 & \\ & & & & & 0 & -1 & +1 & -1 & 0 & \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & & \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & & & \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & & & & & \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (-78) \end{array}$$

Multiplication Requiring only n/2 Summands

$$\begin{array}{rrrrrrrrrrl} & & & & & 0 & 1 & 1 & 0 & 1 & \\ & & & & & 0 & & -1 & & -2 & \\ \hline 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & & & & & \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & \end{array}$$

Carry-Save Addition of Summands

- Consider the addition of many summands, we can:

➢ Group the summands in threes and perform carry-save addition on each of these groups in parallel to generate a set of S and C vectors in one full-adder delay

➢ Group all of the S and C vectors into threes, and perform carry-save addition on them, generating a further set of S and C vectors in one more full-adder delay

➢ Continue with this process until there are only two vectors remaining

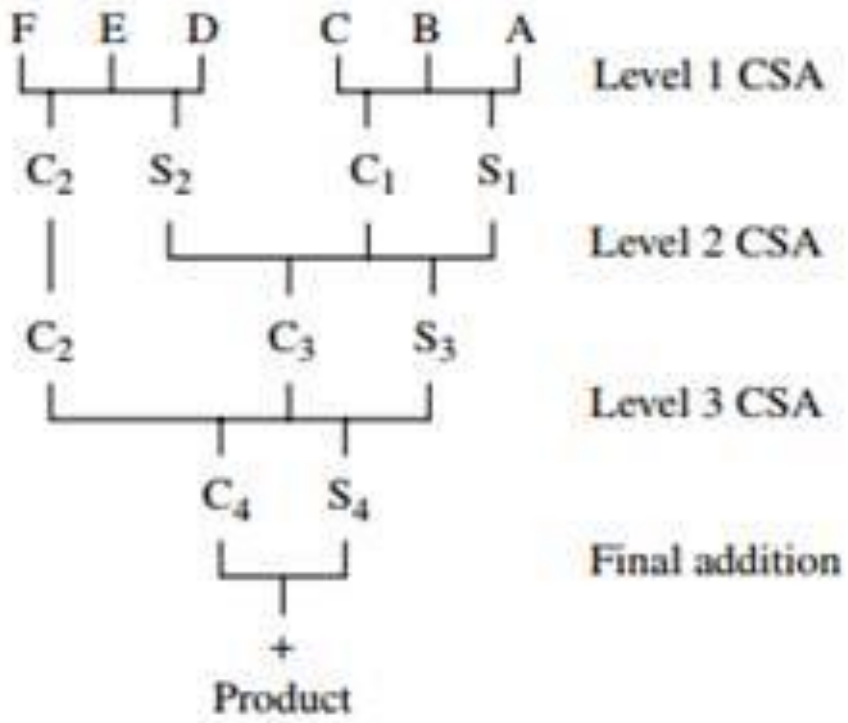


Figure 9.19 Schematic representation of the carry-save addition operations in Figure 9.18.

Carry-Save Addition of Summands

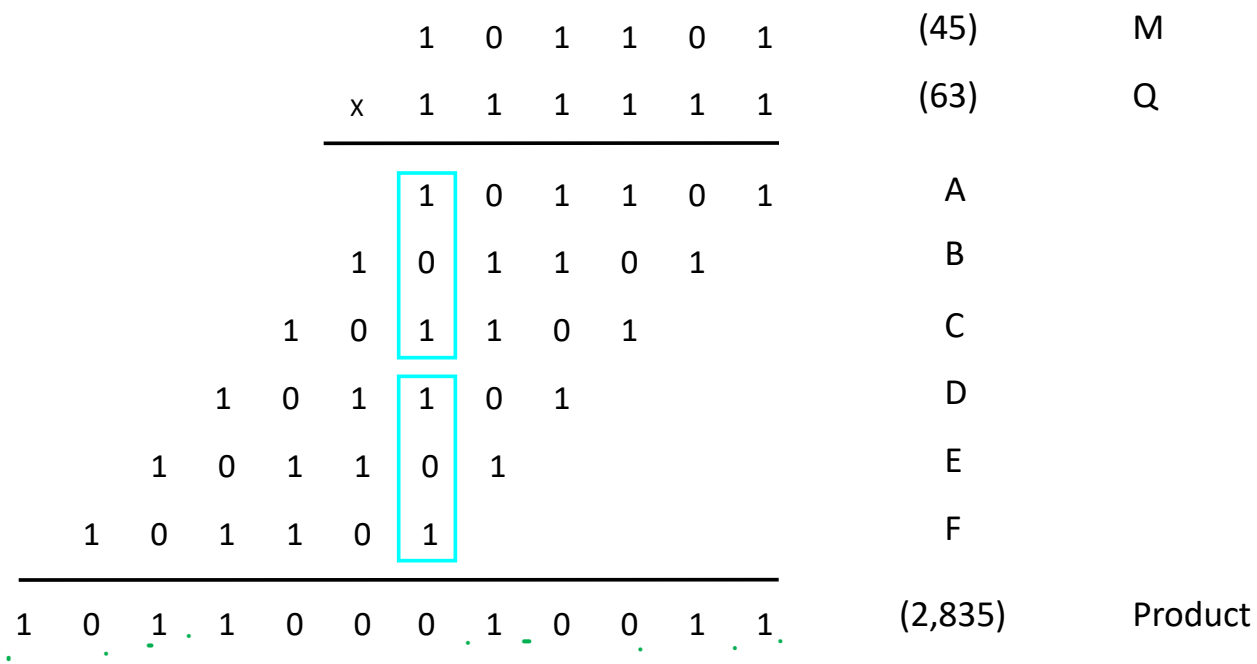


Figure 6.17. A multiplication example used to illustrate carry-save addition as shown in Figure 6.18.

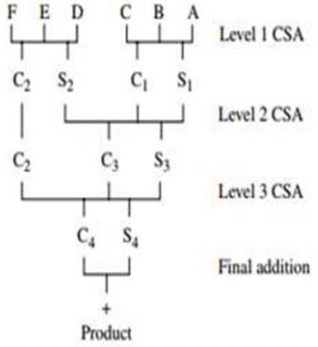


Figure 9.19 Schematic representation of the carry-save addition operations in Figure 9.18.

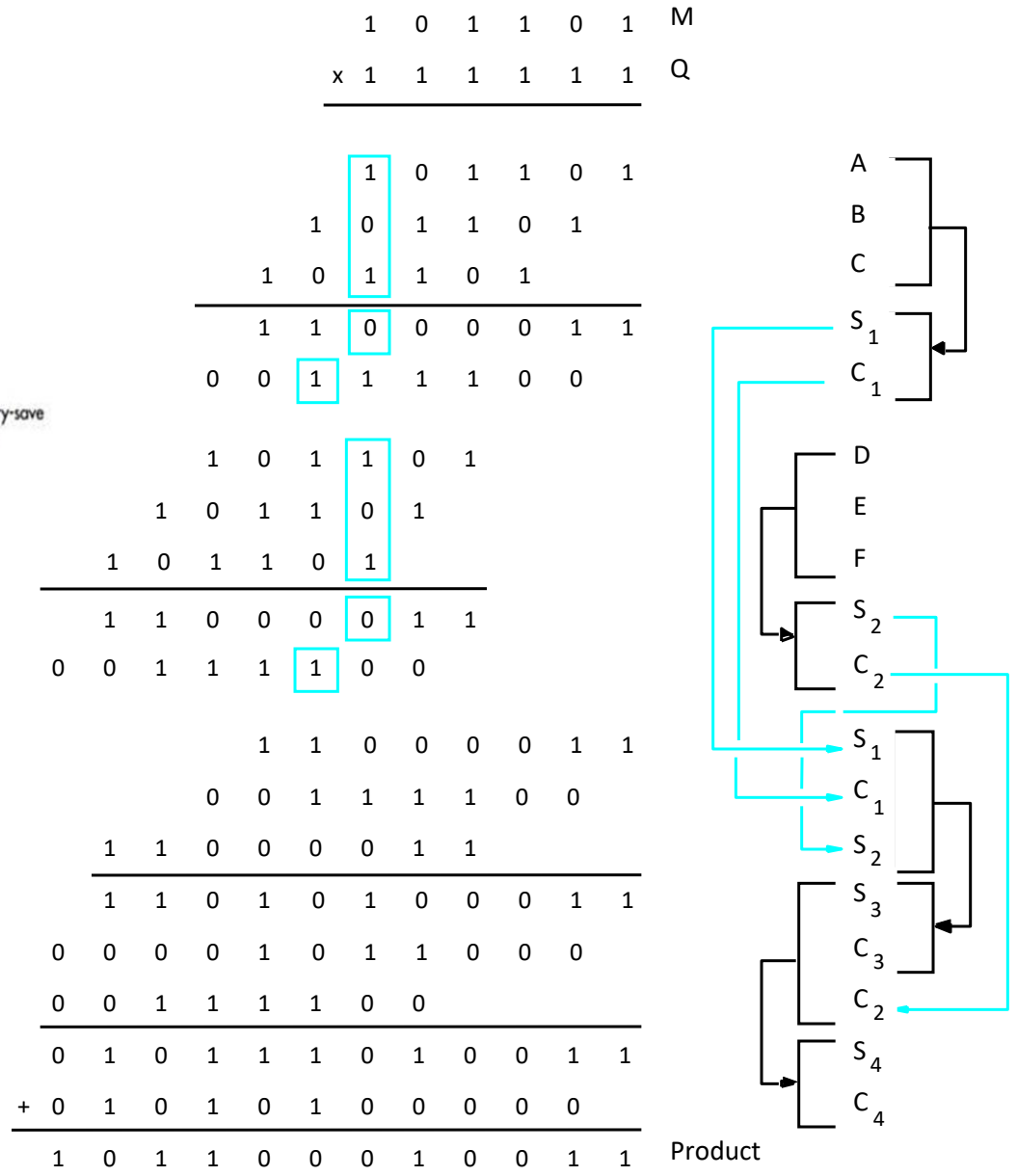
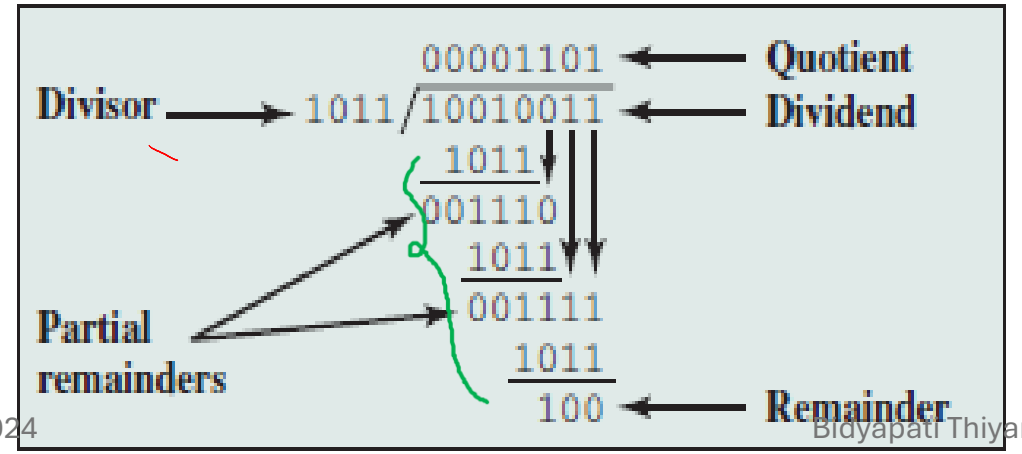


Figure 6.18. The multiplication example from Figure 6.17 performed using carry-save addition.

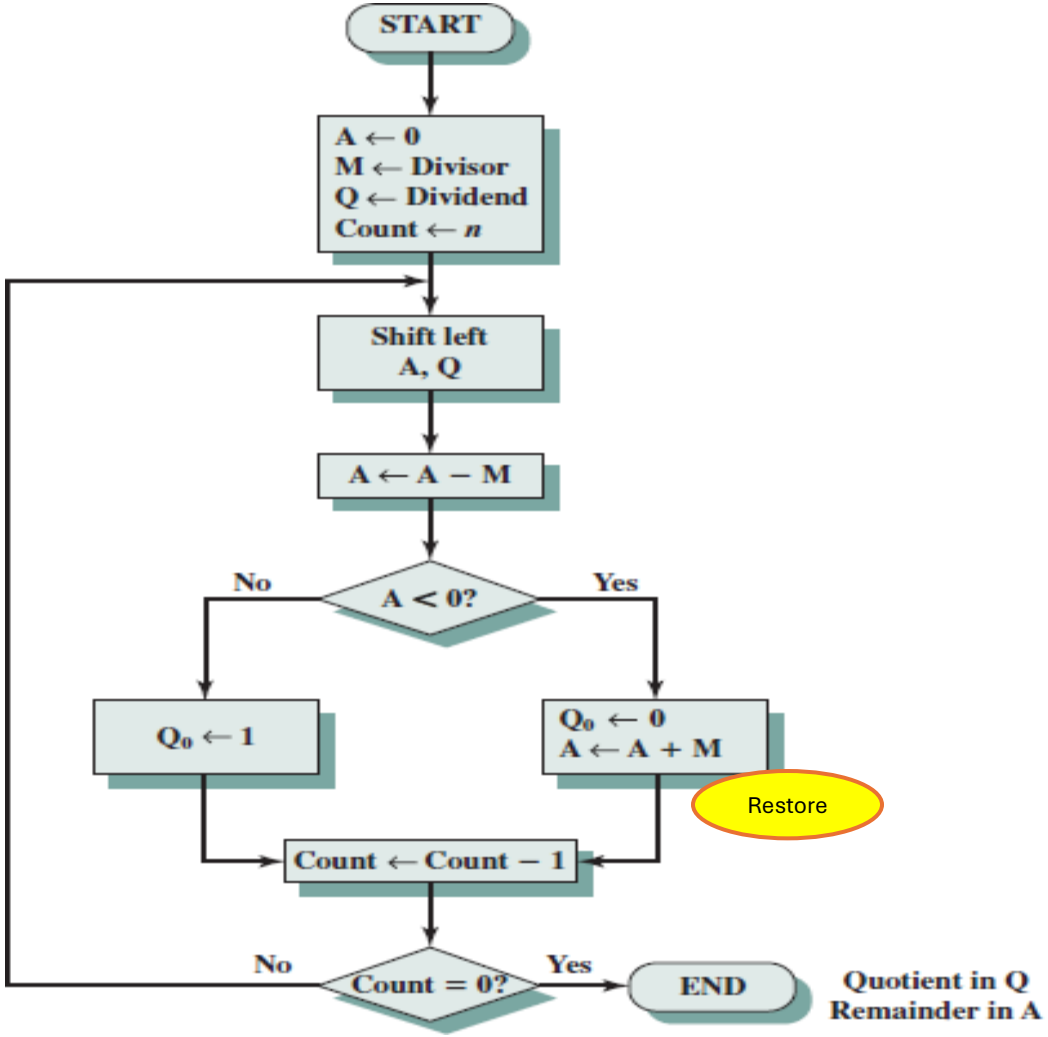
Binary Division

```
        21
  13 ) 274
       26
      ---
       14
       13
      ---
        1
```

```
          10101
       ----------
1101 ) 100010010
       1101
       ----
        10000
        1101
        ----
         1110
         1101
         ----
            1
```



Restoring Division



Step Involved

Step-1: First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)

Step-2: Then the content of register A and Q is shifted left as if they are a single unit

Step-3: Then content of register M is subtracted from A and result is stored in A

Step-4: Then the most significant bit of the A is checked if it is 0 the least significant bit of Q is set to 1 otherwise if it is 1 the least significant bit of Q is set to 0 and value of register A is restored i.e the value of A before the subtraction with M

Step-5: The value of counter n is decremented

Step-6: If the value of n becomes zero we get of the loop otherwise we repeat from step 2

Step-7: Finally, the register Q contains the quotient and A contain remainder

1. M=3, Q=8

A	Q	M	n	Steps
0000	1000	0011	4	Initial steps
0001	000 ■			Shift Left A,Q
1110	000 ■			A=A-M
1110	0000			Set Qo= 0
0001				A=A+M
0001	0000	0011	3	
0010	000 ■			Shift Left A,Q
1111	000 ■			A=A-M
1111	0000			Set Qo= 0
0010				A=A+M

A=0001
M=0011
1's C of M= 1100 2's C of M= 1101
A=1110

A=1110
M=0011
A=0001

A	Q	M	n	Steps
0010	0000	0011	2	Initial steps
0100	000 ■			Shift Left A,Q
0001	000 ■			A=A-M
0001	0001			Set Qo= 1
0001	0001	0011	1	
0010	001 ■			Shift Left A,Q
1111	001 ■			A=A-M
1111	0010			Set Qo= 0
0010	0010			A=A+M

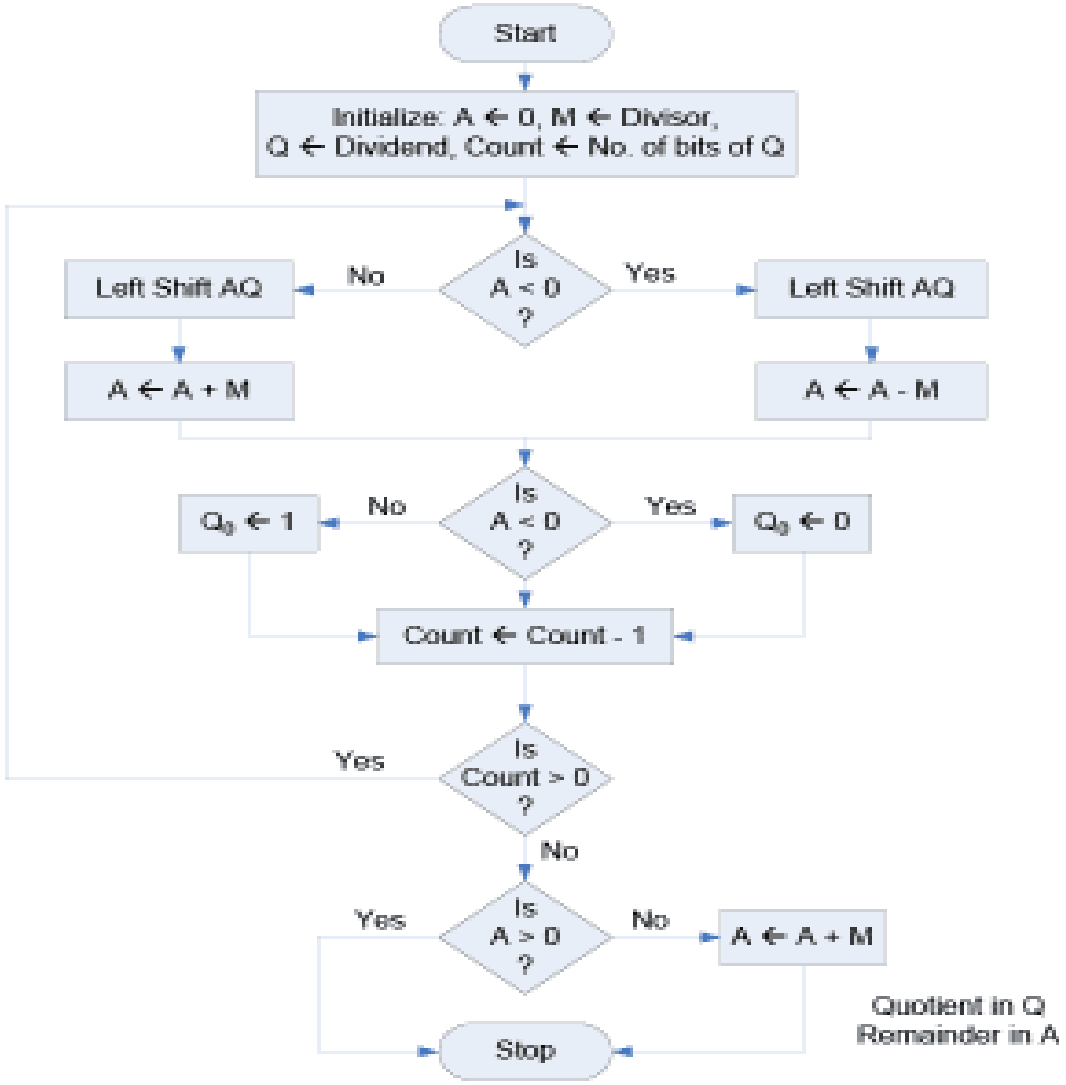
A=0100
M=0011
1's C of M= 1100 2's C of M= 1101
A=0001

A=0010
M=1101
A=1111

Remainder

Quotient

Non-restoring



Non-restoring algorithm

Step-1: First the registers are initialized with corresponding values (Q = Dividend, M = Divisor, A = 0, n = number of bits in dividend)

Step-2: Check the sign bit of register A

Step-3: If it is 1 shift left content of AQ and perform A = A+M, otherwise shift left AQ and perform A = A-M (means add 2's complement of M to A and store it to A)

Step-4: Again the sign bit of register A

Step-5: If sign bit is 1 Q[0] become 0 otherwise Q[0] become 1 (Q[0] means least significant bit of register Q)

Step-6: Decrements value of N by 1

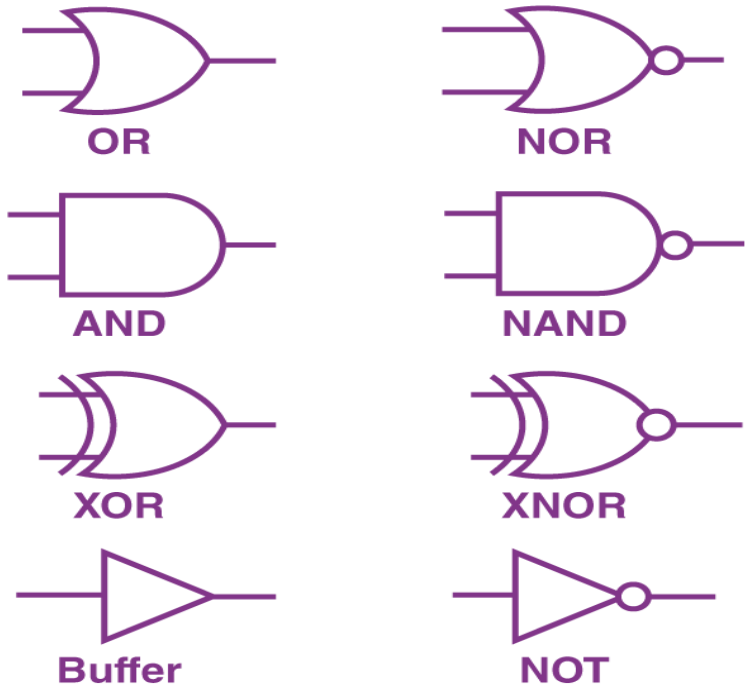
Step-7: If N is not equal to zero go to Step 2 otherwise go to next step

Step-8: If sign bit of A is 1 then perform A = A+M

Step-9: Register Q contains quotient and A contains remainder.

Digital Logic Circuit

A digital logic circuit, also known as a logic circuit, is an electronic circuit that performs a logical operation. These circuits are the foundation of all computers and computer systems.



- Combinational Circuit
- Sequential Circuit

Combinational Circuit

- Combinational circuit is a circuit in which we combine the different gates in the circuit Means this is the combination of different logic circuits.
- Characteristics of combinational circuits are following –
  - The output of combinational circuit at any instant of time, depends only on the levels present at input terminals.
  - The combinational circuit do not use any memory. The previous state of input does not have any effect on the present state of the circuit.
  - A combinational circuit can have an n number of inputs and m number of outputs.

Types of Combinational circuit

Combinational Circuit are categorized into three categories:

- Arithmetic and Logical Function
- Data Transmission
- Code Convertor

- Arithmetic & Logical Function
  - Adders
  - Subtractors
  - Comparators
  - PLDs
- Data Transmission
  - Multiplexer - DeMultiplexer
  - Encoder – Decoder
- Code Convertor
  - Binary
  - BCD

Block diagram of Combinational circuit

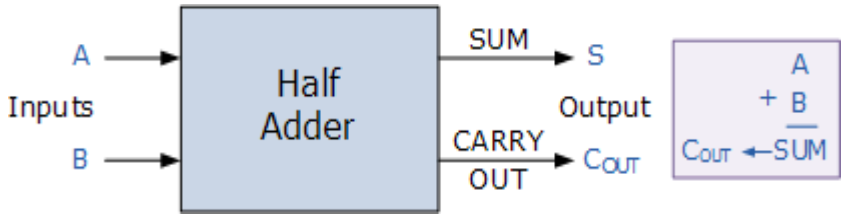
ADDER:

- An adder is a digital logic circuit in electronics that is extensively used for the addition of numbers.
- Adders are basically classified into two types: Half Adder and Full Adder
  - Half adder:
    - A typical **combinational** circuit
    - Adding two binary digits together.
  - How to construct a half-adder?

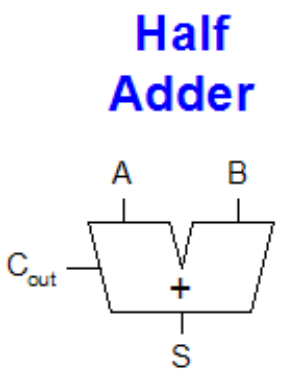
    The truth table reveals that
    - Sum is actually an XOR.
    - Carry is equivalent to an AND gate.
  - Combining an XOR gate and an AND gate results in the logic diagram for a half-adder.

1-bit Half Adder

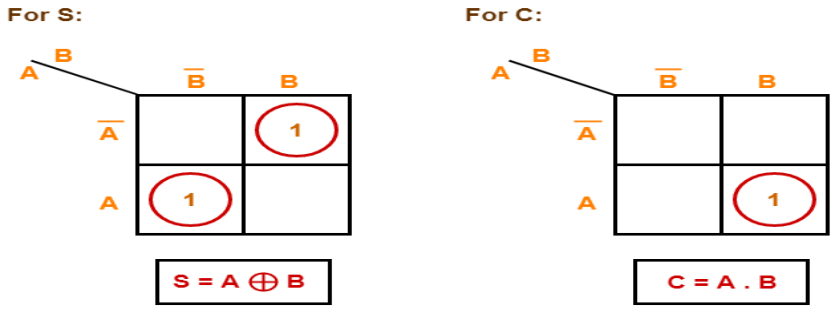
- Let's first consider how to implement an 1-bit adder

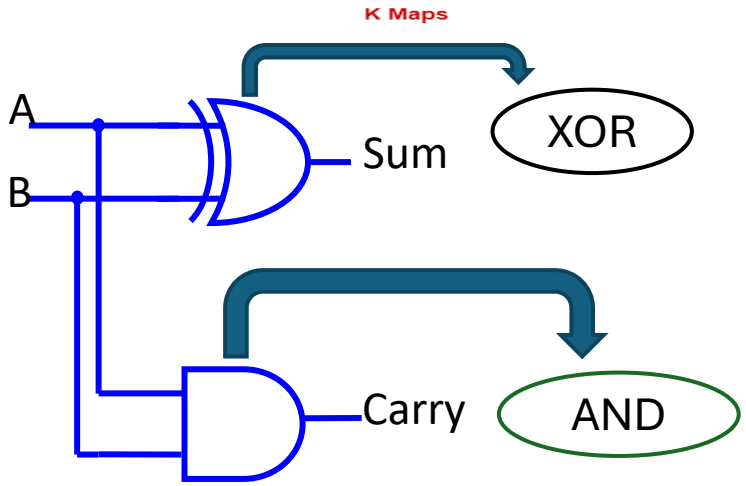


- Half adder
  - 2 inputs: A and B
  - 2 outputs: S (Sum) and C_{out} (Carry)



A	B	S(um)	C(arry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1





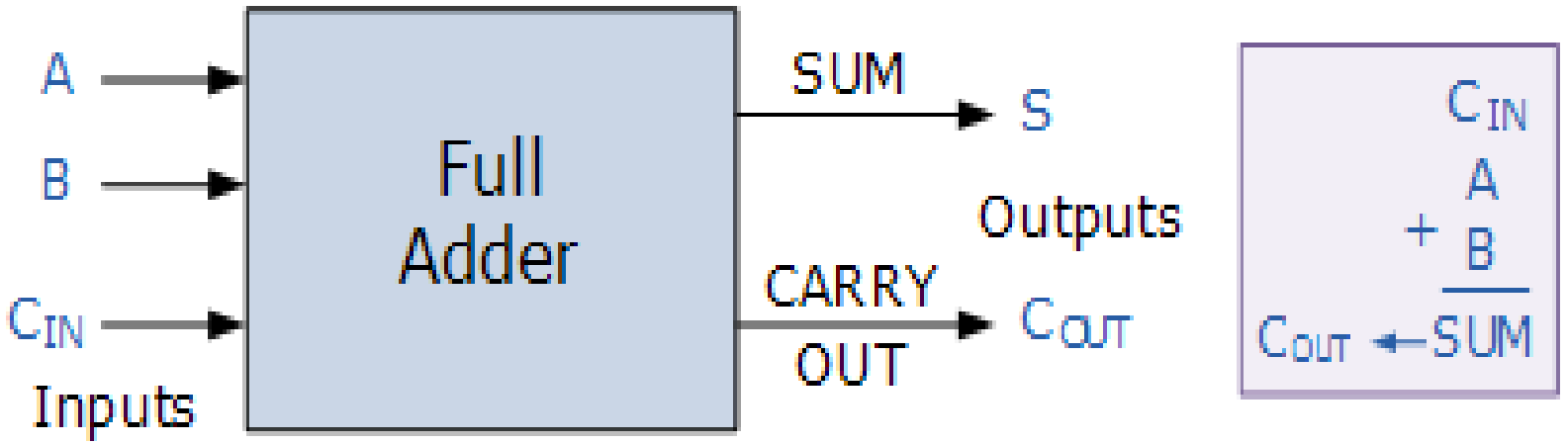
$$S = \overline{A}B + A\overline{B} = A \oplus B$$

$$C = AB$$

1-bit Full Adder

- Half adder lacks a C_{in} input to accept C_{out} of the previous column

- Full adder
  - 3 inputs: A, B, C_{in}
  - 2 outputs: S, C_{out}



1-bit Full Adder

Cin	A	B	S(um)	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum

Cin \ AB	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = Cin\overline{A}\,\overline{B} + \overline{Cin}\,\overline{A}B + CinAB + \overline{Cin}A\overline{B}$$
$$= Cin(\overline{A}\,\overline{B} + AB) + \overline{Cin}(\overline{A}B + A\overline{B})$$
$$= Cin(\overline{A \oplus B}) + \overline{Cin}(A \oplus B)$$
$$= Cin \oplus A \oplus B$$

C_{out}

Cin \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$Cout = CinB + CinA + AB = AB + Cin(A + B)$$

or

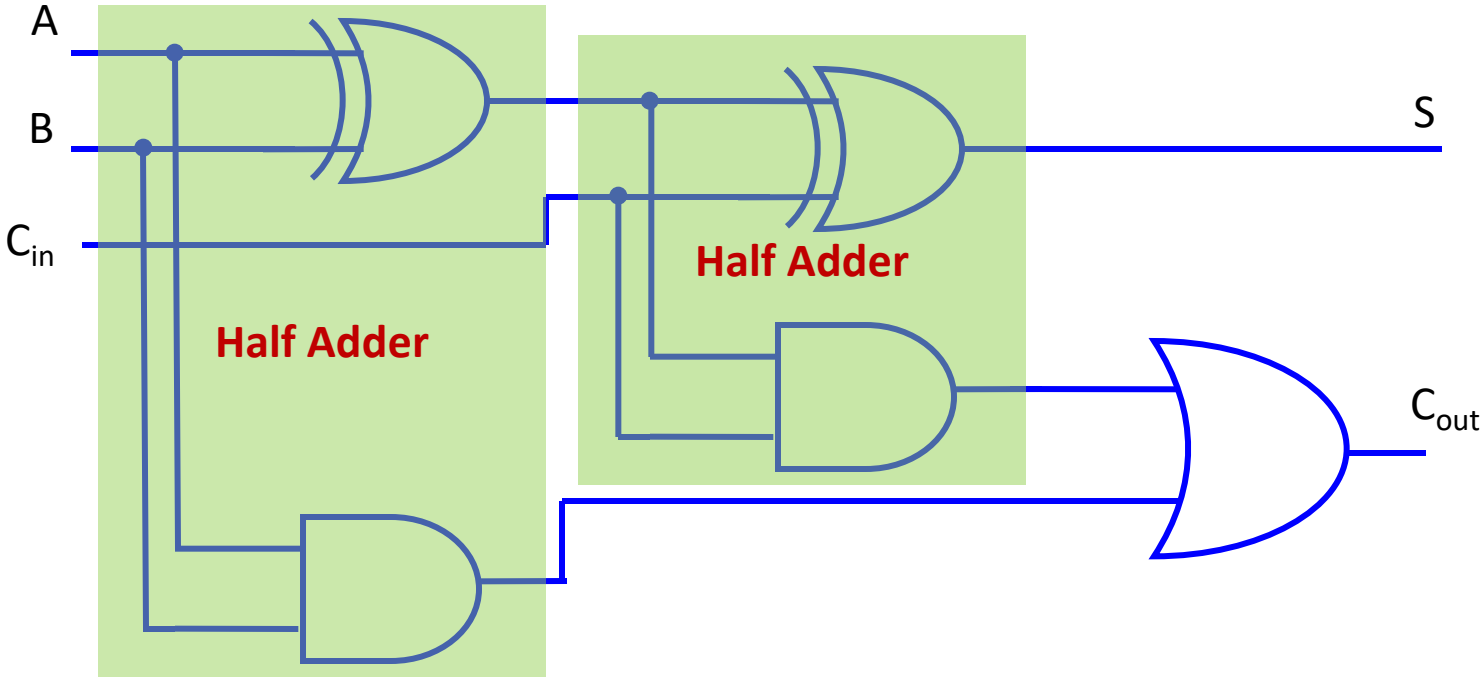
Cin \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$Cout = AB + Cin(\overline{A}B + A\overline{B}) = AB + Cin(A \oplus B)$$

1-bit Full Adder Schematic

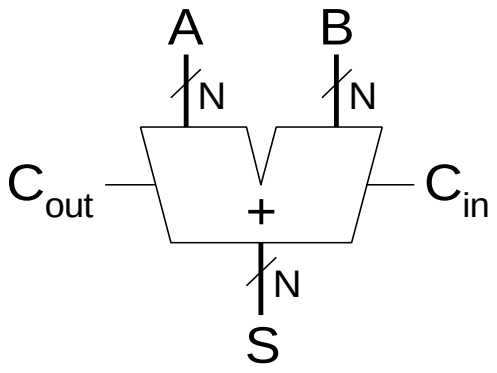
$$S = Cin \oplus A \oplus B$$

$$Cout = AB + Cin(A \oplus B)$$



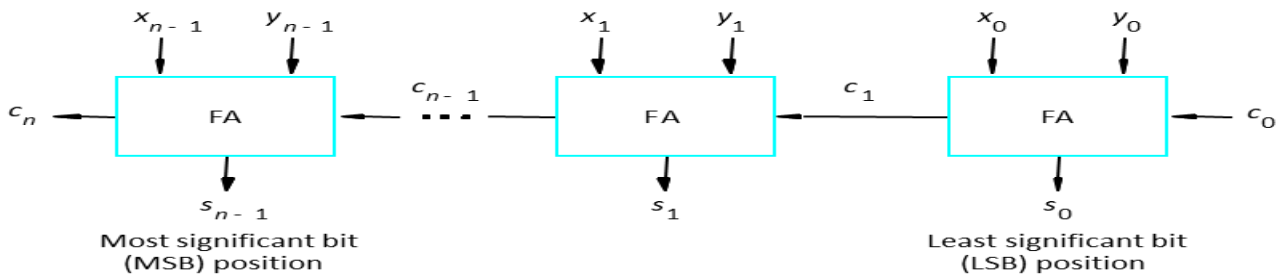
Multi-bit Adder

- It seems that an 1-bit adder is doing not much of work
- Three common CPA implementations
  - Ripple-carry adders (slow)
  - Carry-lookahead adders (fast)
  - Prefix adders (faster)
- It is commonly called carry propagate adders (CPAs) because the carry-out of one bit propagates into the next bit



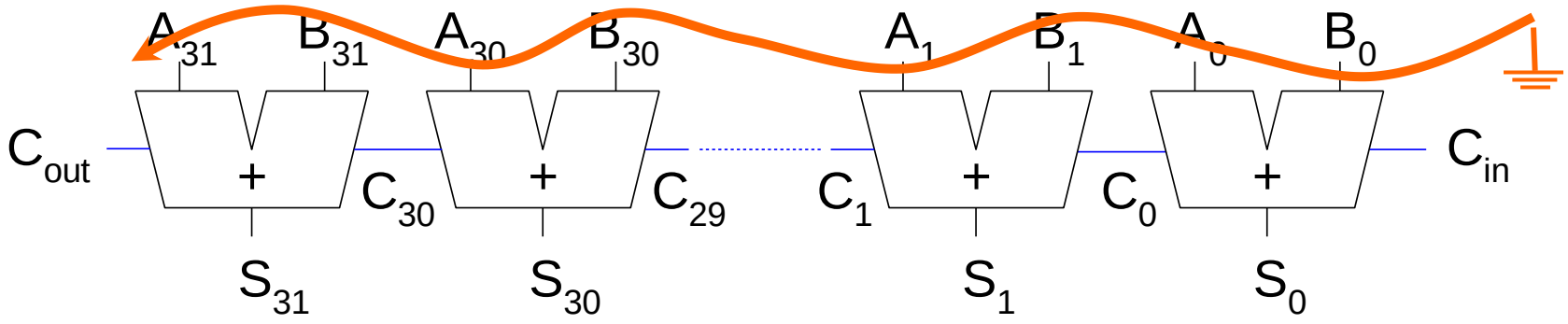
Ripple-Carry Adder

- The simplest way to build an N-bit CPA is to chain 1-bit adders together
  - Carry ripples through entire chain

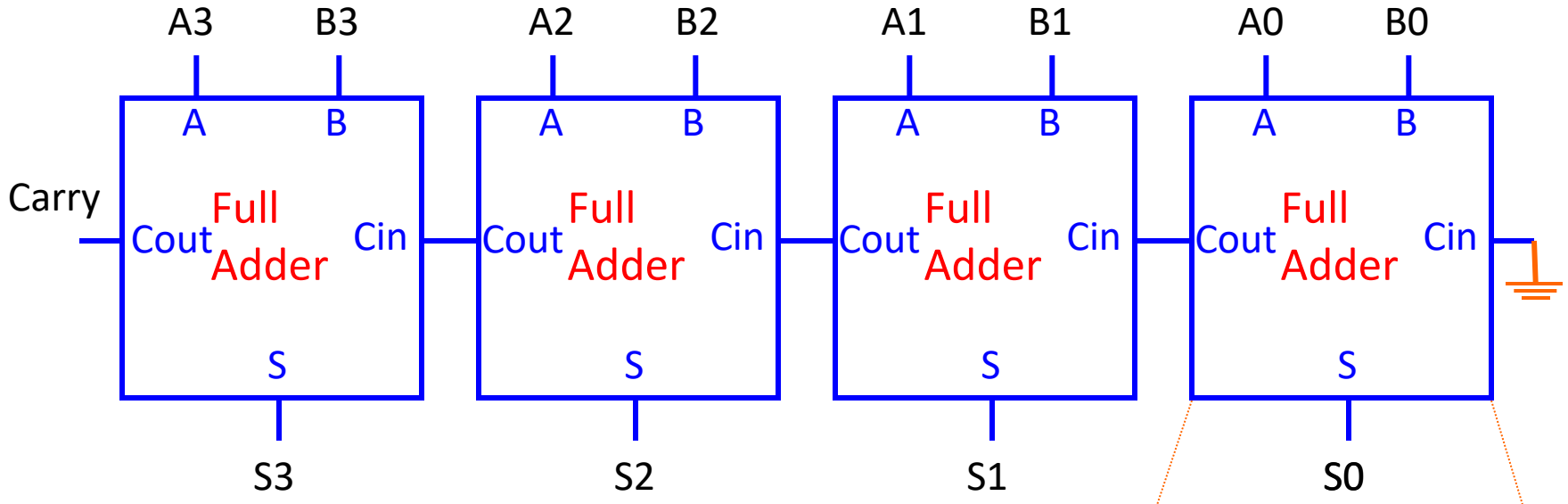


(b) An n-bit ripple-carry adder

Example: 32-bit Ripple Carry Adder

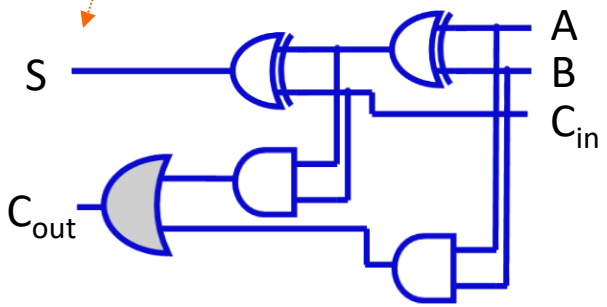


4-bit Ripple-Carry Adder

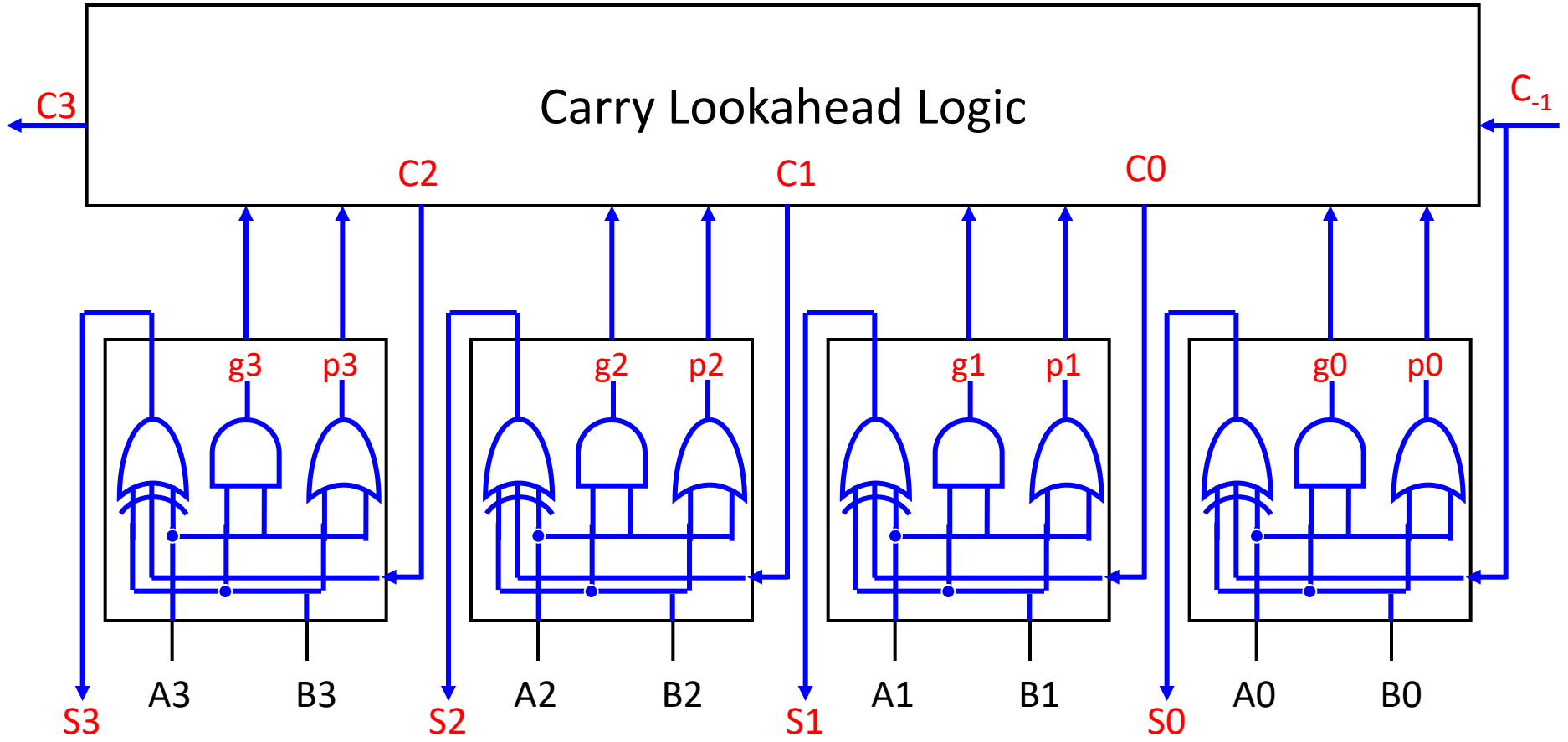


$$S = Cin \oplus A \oplus B$$

$$Cout = AB + Cin(A \oplus B)$$



4-bit CLA

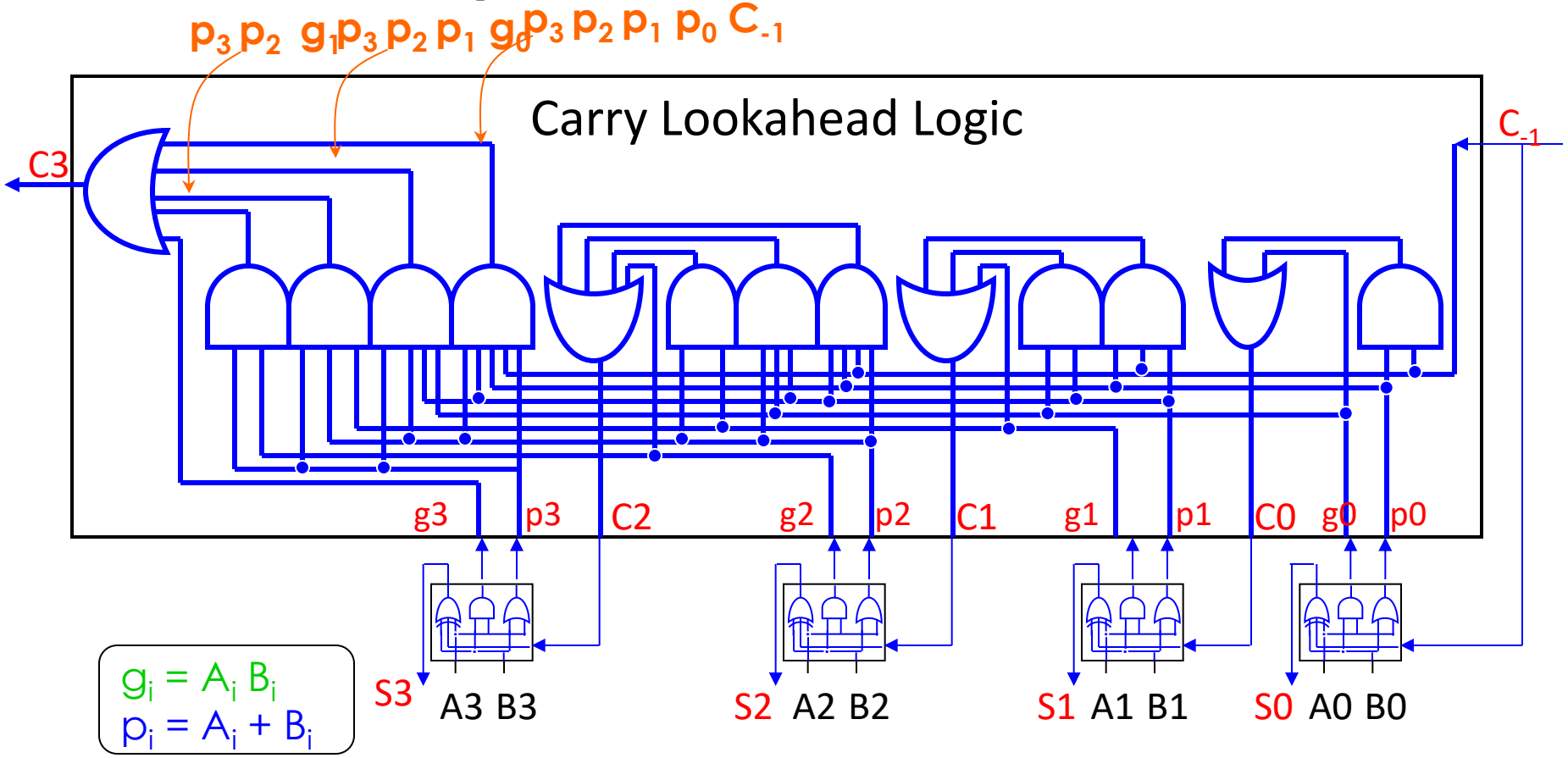


$$S_i = C_i \oplus A_i \oplus B_i$$

$$g_i = A_i B_i \quad \text{(generate)}$$

$$p_i = A_i + B_i \quad \text{(propagate)}$$

A CLA Implementation



$C_0 = g_0 + p_0\,C_{-1}$

$C_1 = g_1 + p_1\,C_0 = g_1 + p_1\,g_0 + p_1\,p_0\,C_{-1}$

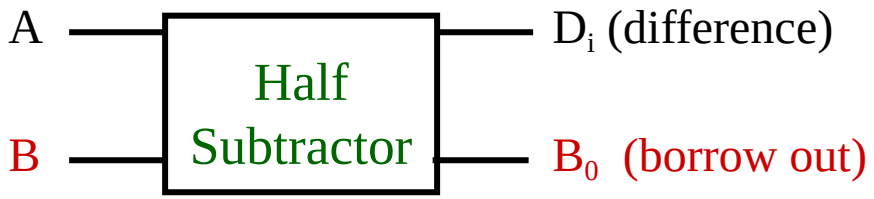
$C_2 = g_2 + p_2\,C_1 = g_2 + p_2\,g_1 + p_2\,p_1\,g_0 + p_2\,p_1\,p_0\,C_{-1}$

$C_3 = g_3 + p_3\,C_2 = g_3 + p_3\,g_2 + p_3\,p_2\,g_1 + p_3\,p_2\,p_1\,g_0 + p_3\,p_2\,p_1\,p_0\,C_{-1}$

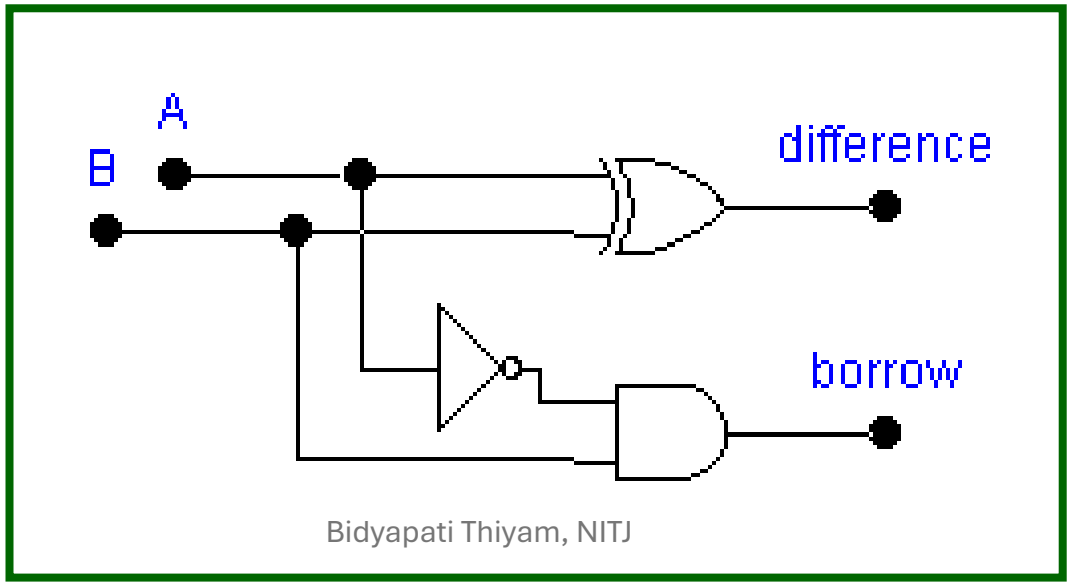
HALF SUBTRACTOR

Input Output

Logic Symbol:



Logic Diagram:

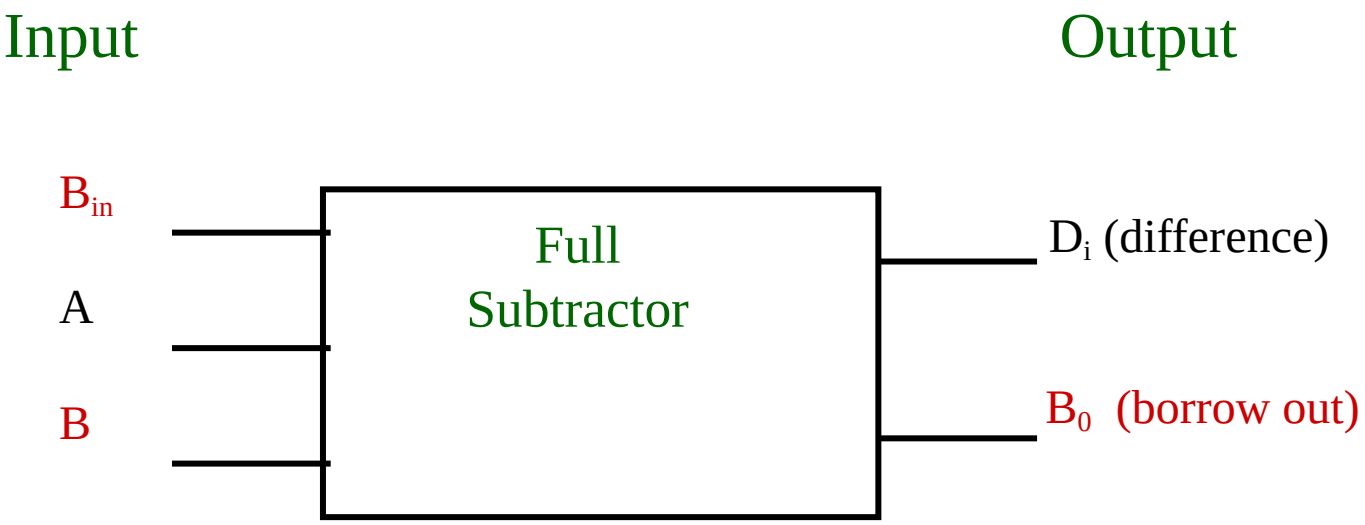


Half subtractor truth table :

Inputs		Outputs	
X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

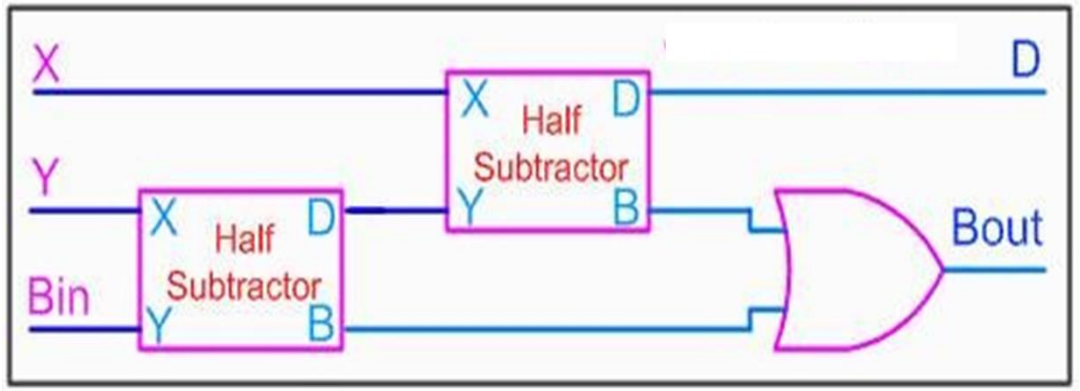
FULL SUBTRACTOR

Logic Symbol:

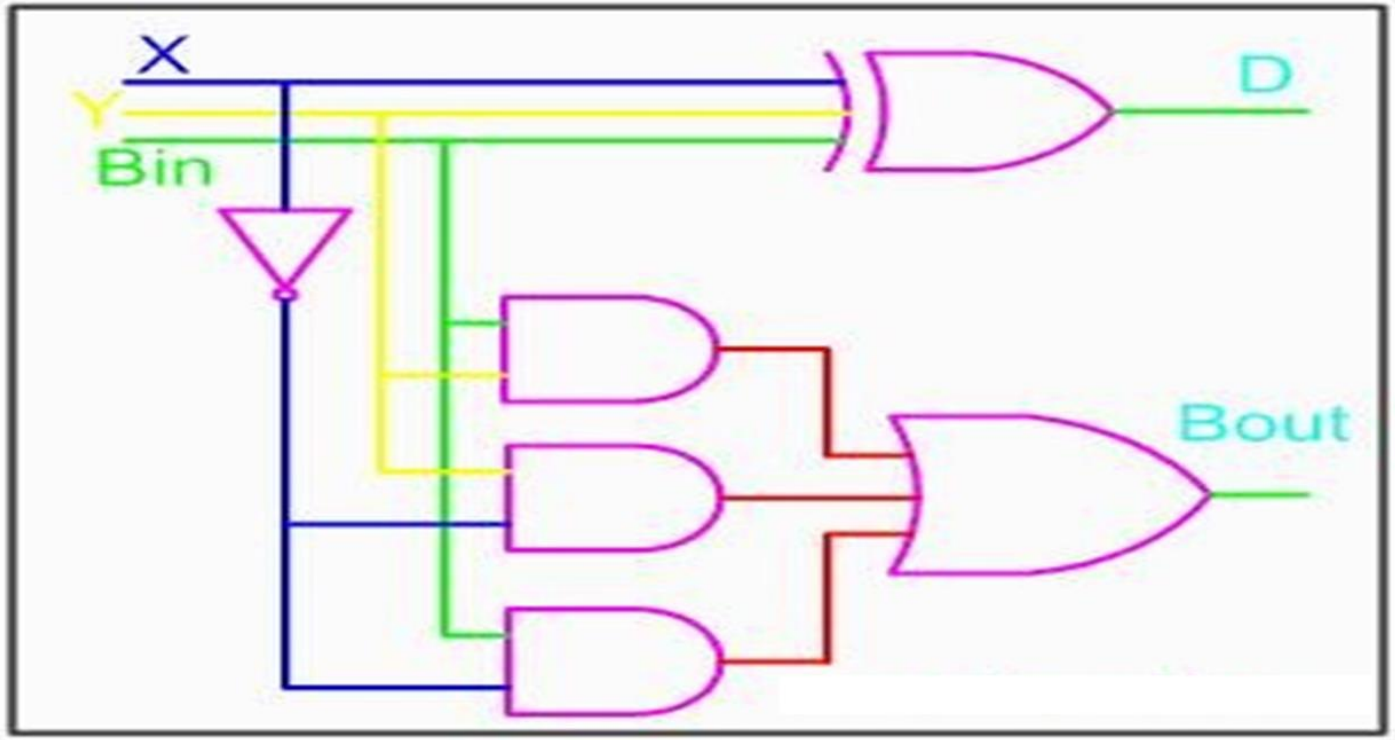


Full subtractor truth table

Inputs			Outputs	
X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

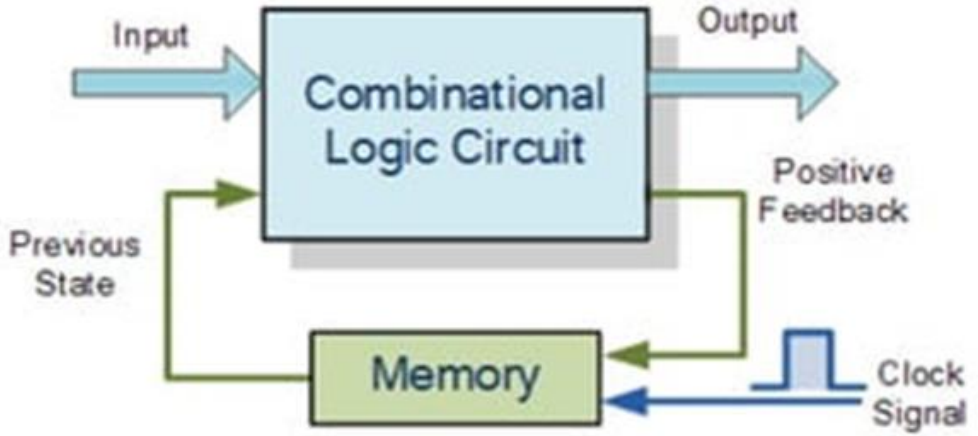


Full-subtractor circuit is more or less same as a full-adder with slight modification.



Sequential Circuit

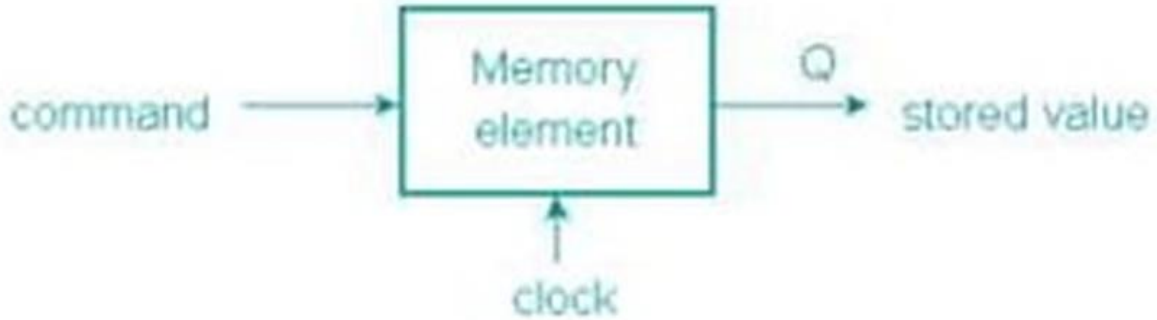
- Sequential circuit consists of feedback path and several memory elements



- Sequential circuit = Combinational Logic + Memory Elements

Memory Element

- Memory element device that can remember a value for a certain period, or change value based on the input instruction
- Example: Latch and flip-flop



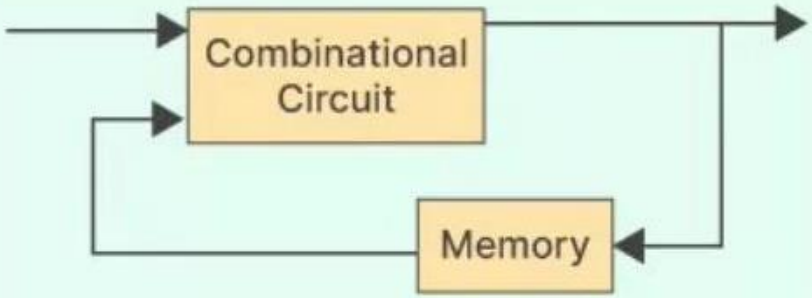
Commands for latches include set and reset commands

Types of Sequential Circuit:

- The sequential circuit can be classified depending on the timing of their signals :
  1. Synchronous sequential circuit and
  2. Asynchronous sequential circuit.

In synchronous sequential circuits, the output changes whenever a clock pulse is applied.
Ex: clocked flip-flops.
Asynchronous sequential circuits do not use clock pulses.
Ex: unclocked flip-flops (Latches) .

Combinational vs Sequential Circuits

Combinational Circuit	Sequential Circuit
Output only depends on the present input	Output depends on present input & past output
Memory element is absent	Memory element is present
No clock signal is applied	Clock signal is required
	
Example - Half Adder, Full Adder, Multiplexer	Example - Flipflop, Counters, Registers

ASSIGNMENT-1

1. Define adder. Explain the types of adders?
2. Explain bits, bytes, and words with example.
3. A half adder is a combinational logic circuit that has two inputs, x and y, and two outputs, s and c, that are the sum and carry-out, respectively, resulting from the binary addition of x and y.
   (a) Design a half adder as a two-level AND-OR circuit.
   (b) show how to implement a full adder, by using two half adders and external logic gates, as necessary.
4. Using manual methods, perform the operations A x B and A/B on the 5-bit unsigned numbers A= 55 and B= 5
5. Perform restoring and non-restoring division algorithm
   Given M =10101 and Q =00100

Submission date on or before – 4th September 2024