EXPERIMENT NO. 1 DATE:

COMPONENTS OF A COMPUTER

AIM : To assemble a computer system.

1. **POWER SUPPLY**

A power supply converts Ac power from an outlet into Dc power for the computer. Most modern desktop personal computer power supplies conform to the ATX specification which includes form factor and voltage tolerances. While an ATX power supply is connected to the mains supply, it always provides a 5-volt standby (5VSB) power so that the standby functions on the computer and certain peripherals are powered.

2. **HEAT SINK FAN**

Forces air over the heat sink. A heat sink and fan (HSF) is an active cooling solution used to cool down integrated circuits in computer systems, commonly the central processing unit (CPU). As the name suggests, it is composed of a passive cooling unit (the heat sink) and a fan.

3. **VIDEO ADAPTER CARD**

A graphics card (also called a video card, display card, graphics adapter, GPU, VGA card/VGA, video adapter, or display adapter) is an expansion card which generates a feed of output images to a display device, such as a computer monitor.

4. **NIC**

The Full Form of NIC is a Network Interface Card. NIC stands for Network Interface Card: It is a card that holds an Ethernet adapter for connecting the computer to a wired network. It is also called Ethernet or physical or network card.

5. **WIRELESS NIC**

A wireless network interface controller (WNIC) is a network interface controller which connects to a wireless network, such as Wi-Fi or Bluetooth, rather than a wired network, such as a Token Ring or Ethernet. A WNIC, just like other NICs, works on the layers 1 and 2 of the OSI model and uses an antenna to communicate via radio waves**.**

6. **OPTICAL DRIVE**

In computing, an optical disc drive is a disc drive that uses laser light or electromagnetic waves within or near the visible light spectrum as part of the process of reading or writing data to or from optical discs. Some drives can only read from certain discs, but recent drives can both read and record, also called burners or writers (since they physically burn the organic dye on write-once CD-R, DVD-R and BD-R LTH discs). Compact discs, DVDs, and Blu-ray discs are common types of optical media which can be read and recorded by such drives.

7. **FLOPPY DRIVE**

A floppy disk or floppy diskette (casually referred to as a floppy, or a diskette) is an obsolescent type of disk storage composed of a thin and flexible disk of a magnetic storage medium in a square or nearly square plastic enclosure lined with a fabric that removes dust particles from the spinning disk. Floppy disks store digital data which can be read and written when the disk is inserted into a floppy disk drive (FDD) connected to or inside a computer or other device.

8. HARD DISK DRIVE

A hard disk drive (HDD), hard disk, hard drive, or fixed disk[b] is an electro-mechanical data storage device that stores and retrieves digital data using magnetic storage with one or more rigid rapidly rotating platters coated with magnetic material. The platters are paired with magnetic heads, usually arranged on a moving actuator arm, which read and write data to the platter surfaces.[2] Data is accessed in a random-access manner, meaning that individual blocks of data can be stored and retrieved in any order.

9. SATA CABLE

SATA (also referred to as Serial ATA) stands for Serial Advanced Technology Attachment, an industry-standard bus interface for connecting a computer's host bus adapter to storage devices such as hard disk drives (HDD), optical drives and solid-state drives (SSD).

10. MOTHERBOARD

A motherboard (also called mainboard, main circuit board,[1] mb, mboard, backplane board, base board, system board, logic board (only in Apple computers) or mobo) is the main printed circuit board (PCB) in general-purpose computers and other expandable systems.

11. PATA CABLE

PATA, short for Parallel ATA, is an IDE standard for connecting storage devices like hard drives and optical drives to the motherboard. PATA generally refers to the types of cables and connections that follow this standard. It's important to note that the term Parallel ATA used to simply be called ATA. ATA was retroactively renamed to Parallel ATA when the newer Serial ATA (SATA) standard came into being.

12. RAM

Random-access memory (RAM) is a form of computer memory that can be read and changed in any order, typically used to store working data and machine code. A random-access memory device allows data items to be read or written in almost the same amount of time irrespective of the physical location of data inside the memory, in contrast with other direct-access data storage media (such as hard disks, CD-RWs, DVD-RWs and the older magnetic tapes and drum memory), where the time required to read and write data items varies significantly depending on their physical locations on the recording medium, due to mechanical limitations such as media rotation speeds and arm movement.

EXPERIMENT NO. 2 DATE:

ASSEMBLING OF A COMPUTER SYSTEM

AIM : To assemble a computer system.

PROCEDURE :

- The assembling of the computer system is exactly the opposite of disassembling operation. Before starting assembling the computer system, make sure you have the screws and a screwdriver for those.
- The first step for assembling the computer system starts with mounting the processor on the processor socket of the motherboard. To mount the process, you don't need to apply any force. The special ZIF (zero insertion force) sockets are usually used to prevent any damage to the processor pins. Once the processor is mounted, the heat sink will be attached on top of the processor. The CPU fan is also attached on top of the heat sink.
- Now the motherboard is to be fixed vertically in the tower case and the screws are fixed from behind of the motherboard.
- Now line up the power supply at the top back end of the cabinet and screw it. The power connectors for motherboard power supply and CPU fan power supply are to be connected. If the cabinet cooling FAN is required then it is to be screwed at the back end grill of the cabinet and its power connector is to be connected from SMPS.
- Install the CD/DVD drives at the top front end of the cabinet and screw it. Install the Hard disk drive and floppy disk drive below CD/DVD drive and screw it. Make sure once screwed there is no vibration in either of the CD/DVD, Hard disk or Floppy disk drives.
- Now select the appropriate data cable and connect one end of the cable to its drive socket and another end at its appropriate connector on the motherboard. For SATA hard disk drive or CD/DVD drives use SATA cable and its power cable, else use IDE data cable. Do the proper jumper settings as per the usage requirement.
- It is time now to mount the memory modules on the motherboard by aligning the RAM to its socket on the motherboard and press it downward. Make sure the side tab are fixed into the RAM notch. If not, you may still have to press a bit.
- Install the internal cards to its socket and attach the cables or power cable to it. The selection of right socket or slot is required as per the type of socket.
- Cover the tower by placing it and pressing towards front side and screw it.
- Connect the external devices with CPU at its appropriate socket. It includes

mouse and keyboard at PS2 or USB connectors. Monitor at the video output socket. Connect the power cable to the back of tower in SMPS. Plug in the power cable to the electric board.

RESULT : The computer system was successfully assembled.

EXPERIMENT NO. 3 DATE:

DISASSEMBLING OF A COMPUTER SYSTEM

AIM : To disassemble a computer system.

PROCEDURE :

- **Detach the power cable:**

The disassembling of the computer system starts with externally connected device detachment. Make sure the computer system is turned off, if not then successfully shut down the system and then start detaching the external devices from the computer system. It includes removing the power cable from electricity switchboard, then remove the cable from SMPS (switch mode power supply) from the back of the CPU Cabinet. Do not start the disassembling without detaching the power cable from the computer system. Now remove the remaining external devices like keyboard, mouse, monitor, printer or scanner from the back of CPU cabinet.

- **Remove the Cover:**

The standard way of removing tower cases used to be to undo the screws on the back of the case, slide the cover back about an inch and lift it off. The screwdrivers as per the type of screw are required to do the task.

- **Remove the adapter cards:**

Make sure if the card has any cables or wires that might be attached and decide if it would be easier to remove them before or after you remove the card. Remove the screw if any, that holds the card in place. Grab the card by its edges, front and back, and gently rock it lengthwise to release it.

- **Remove the drives:**

Removing drives is easier. There can be possibly three types of drives present in your computer system, Hard disk drive, CD/DVD/Blue-ray drives, floppy disk drives (almost absolute now a day). They usually have a power connector and a data cable attached from the device to a controller card or a connector on the motherboard. CD/DVD/Blue Ray drive may have an analog cable connected to the sound card for direct audio output.

The power may be attached using one of two connectors, a Molex connector or a Berg connector for the drive. The Molex connector may require to be wiggled

slightly from side to side and apply gentle pressure outwards. The Berg connector may just pull out or it may have a small tab which has to be lifted with a screwdriver.

Now Pull data cables off from the drive as well as motherboard connector. The hard disk drive and CD/DVD drives have two types of data cables. IDE and SATA cables. The IDE cables need better care while being removed as it may cause the damage to drive connector pins. Gently wiggle the cable sideways and remove it. The SATA cables can be removed easily by pressing the tab and pulling the connector straight back.

Now remove the screws and slide the drive out the back of the bay.

- **Remove the memory module:**

Memory modules are mounted on the motherboard as the chips that can be damaged by manual force if applied improperly. Be careful and handle the chip only by the edges.

- **Remove the power supply:**

The power supply is attached into tower cabinet at the top back end of the tower. Make sure the power connector is detached from the switchboard. Start removing the power connector connected to motherboard including CPU fan power connector, cabinet fan, the front panel of cabinet power buttons and all the remaining drives if not detached yet.

Now remove the screws of SMPS from the back of the cabinet and the SMPS can be detached from the tower cabinet.

- **Remove the motherboard:**

Before removing all the connectors from the motherboard, make sure u memorize the connectors for assembling the computer if required, as that may require connecting the connectors at its place. Remove the screws from the back of the motherboard and you will be able to detach it from the cabinet. Now remove the CPU fan from the motherboard. The heat sink will be visible now which can be removed by the pulling the tab upward. Finally, the processor is visible now, which can be removed by the plastic tab which can be pulled back one stretching it side way.

RESULT : The computer system was successfully disassembled.

EXPERIMENT NO. 4a DATE:

ADDITION OF TWO 8-BIT NUMBERS

AIM: Adding two 8 Bit Numbers.

SOFTWARE USED: emu 8086

CODE:

```
data segment
a db 15h
b db 12h
data ends
code segment
assume cs: code, ds: data
start:
mov ax, data
mov ds, ax
mov al, a
mov bl, b
add al, bl
mov c, ax
int 3
code ends
end start
```

ALGORITHM:

Load data from offset 500 to register AL (first number)
Load data from offset 501 to register BL (second number)
Add these two numbers (contents of register AL and register BL)
Apply DAA instruction (decimal adjust)
Store the result (content of register AL) to offset 600
Set register AL to 00
Add contents of register AL to itself with carry
Store the result (content of register AL) to offset 601
Stop

OUTPUT :
(Snapshot)

RESULT : Addition of two 8 bit numbers was successfully done.

EXPERIMENT NO. 4b DATE:

SUBTRACTION OF TWO 8-BIT NUMBERS

AIM : Subtracting two 8 Bit Numbers.

SOFTWARE USED : emu 8086 CODE

:

```
data segment
a db 15h
b db 12h
data ends
code segment
assume cs: code, ds: data
start:
mov ax, data
mov ds, ax
mov al, a
mov bl, b
sub al, bl
mov c, ax
int 3
code ends
end start
```

ALGORITHM:

Load data from offset 500 to register AL (first number)
Load data from offset 501 to register BL (second number)
Subtract these two numbers (contents of register AL and register BL)
Apply DAS instruction (decimal adjust)
Store the result (content of register AL) to offset 600
Set register AL to 00
Add contents of register AL to itself with carry (borrow)
Store the result (content of register AL) to offset 601
Stop

OUTPUT:

(Snapshot)

RESULT : Subtraction of two 8 bit numbers was successfully done.

EXPERIMENT NO. 5a DATE:

ADDITION OF TWO 16-BIT NUMBERS

AIM : Adding two 16 Bit Numbers.

SOFTWARE USED : emu 8086 CODE

:

```
data segment
N1 dw 4004h
N2 dw 1001h
data ends
code segment
assume cs: code, ds: data
start:
mov ax, data
mov ds, ax
mov ax, N1
mov bx, N2
add ax, bx
mov Res, ax
int 3
code ends
end start
```

ALGORITHM:

Load 0000H into CX register (for carry)

Load the data into AX(accumulator) from memory 3000

Load the data into BX register from memory 3002

Add BX with Accumulator AX

Jump if no carry

Increment CX by 1

Move data from AX(accumulator) to memory 3004

Move data from CX register to memory 3006

Stop

OUTPUT:

(Snapshot)

RESULT : Addition of two 16 bit numbers was successfully done.

EXPERIMENT NO. 5b DATE:

SUBTRACTION OF TWO 16-BIT NUMBERS

AIM : Subtracting two 16 Bit Numbers.

SOFTWARE USED : emu 8086 CODE

:

```
data segment
N1 dw 4004h
N2 dw 1001h
data ends
code segment
assume cs: code, ds: data
start:
mov ax, data
mov ds, ax
mov ax, N1
mov bx, N2
sub ax, bx
mov Res, ax
int 3
code ends
end start
```

ALGORITHM:

Load 0000H into CX register (for borrow)
Load the data into AX(accumulator) from memory 3000
Load the data into BX register from memory 3002
Subtract BX with Accumulator AX
Jump if no borrow
Increment CX by 1
Move data from AX(accumulator) to memory 3004
Move data from CX register to memory 3006
Stop

OUTPUT :

(Snapshot)

RESULT : Subtraction of two 16 bit numbers was successfully done.

EXPERIMENT NO. 6 DATE:

FACTORIAL OF A 8-BIT NUMBER

AIM : To find factorial of a 8 bit number.

SOFTWARE USED : emu 8086

CODE :

```
data segment
A db 5
data ends
code segment
assume cs: code, ds: data
start:
mov ax, data
mov ds, ax
mov ah, 00
mov al, a
L1 : Dec a
Mul A
mov cl, a
cmp cl, 01
Jnz L1
Mov ah, 4ch
int 21H
code ends
end start
```

ALGORITHM:

Input the Number whose factorial is to be find and Store that Number in CX Register (Condition for LOOP Instruction)

Insert 0001 in AX(Condition for MUL Instruction) and 0000 in DX

Multiply CX with AX until CX become Zero(0) using LOOP Instruction

Copy the content of AX to memory location 0600

Copy the content of DX to memory location 0601

Stop Execution

OUTPUT :

(Snapshot)

RESULT : Factorial of a 8 bit number has been found.

EXPERIMENT NO. 7 DATE:

MULTIPLICATION OF TWO 8-BIT NUMBERS

AIM : Multiplying two 8 bit numbers.

SOFTWARE USED : emu 8086 CODE

:

```
data segment
A db 09h
B db 02h
Res1 dw ?
data ends
code segment
assume cs: code, ds: data
start:
mov ax, data
mov ds, ax
mov ax, 0000h
mov bx, 0000h
Mul b
mov Res1, ax
int 3
code ends
end start
```

ALGORITHM:

Load data from offset 500 to register AL (first number)

Load data from offset 501 to register BL (second number)

Multiply them (AX=AL*BL)

Store the result (content of register AX) to offset 600

Stop

OUTPUT :

(Snapshot)

RESULT : Multiplication of two 8 bit numbers has been done.

EXPERIMENT NO. 8 DATE:

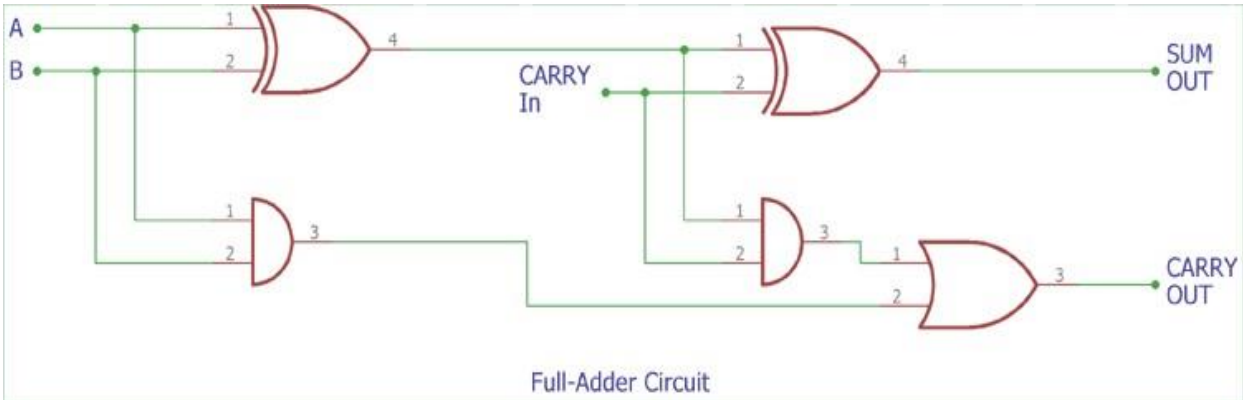
FULL ADDER CIRCUIT

AIM : To design and implement a full adder circuit.

SOFTWARE USED : Logic gate simulator TRUTH TABLE :

INPUTS			OUTPUT	
A	B	C-IN	C-OUT	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

CIRCUIT DIAGRAM :



Full-Adder Circuit

PROCEDURE:

Full Adder is the adder that adds three inputs and produces two outputs.
The first two inputs are A and B and the third input is an input carry as C-IN.
The output carry is designated as C-OUT and the normal output is designated as S which is SUM.
A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to another.
we use a full adder because when a carry-in bit is available, another 1-bit adder must be used since a 1-bit half-adder does not take a carry-in bit.
A 1-bit full adder adds three operands and generates 2-bit results.

OUTPUT :

(Snapshot)

RESULT : Here, we add three one bit binary numbers, two operands & a carry bit.

EXPERIMENT NO. 9 DATE:

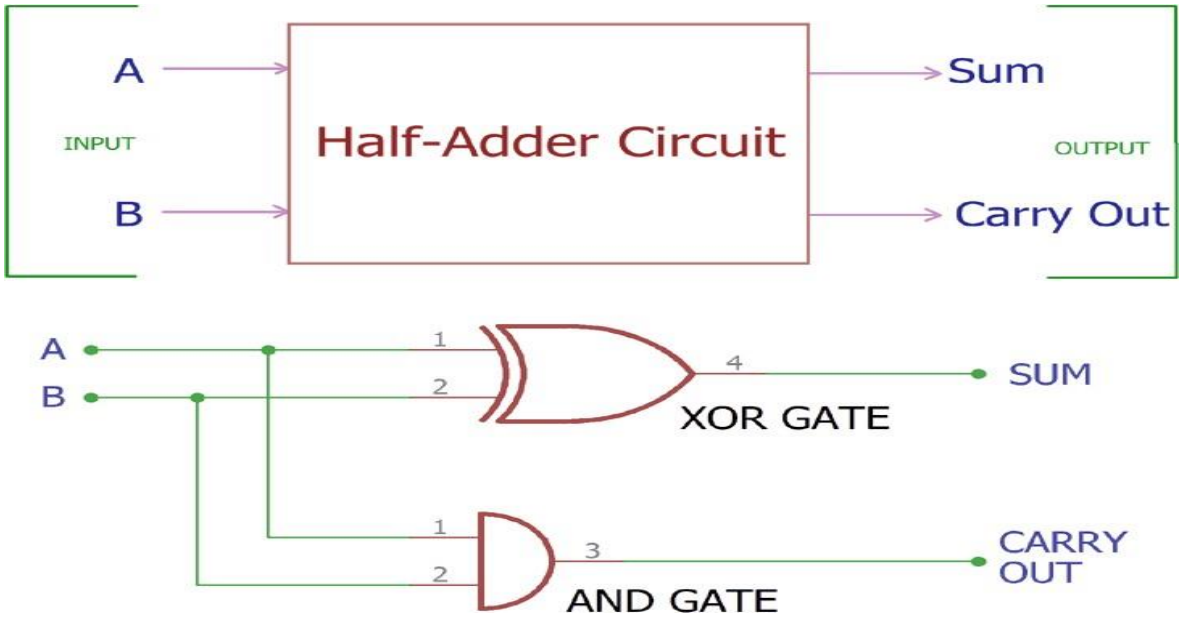
HALF ADDER CIRCUIT

AIM : To design and implement a half adder circuit.

SOFTWARE USED : Logic gate simulator TRUTH TABLE :

INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

CIRCUIT DIAGRAM :



PROCEDURE:

Half adder is the simplest of all adder circuits.

Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (s) and carry bit (c) both as output.

The addition of 2 bits is done using a combination circuit called a Half adder.

The input variables are augend and addend bits and output variables are sum & carry bits.

A and B are the two input bits.

OUTPUT:

(Snapshot)

RESULT : Here, we add two single digit binary numbers & results in two digit out.

EXPERIMENT NO. 10 DATE:

RIPPLE CARRY ADDER

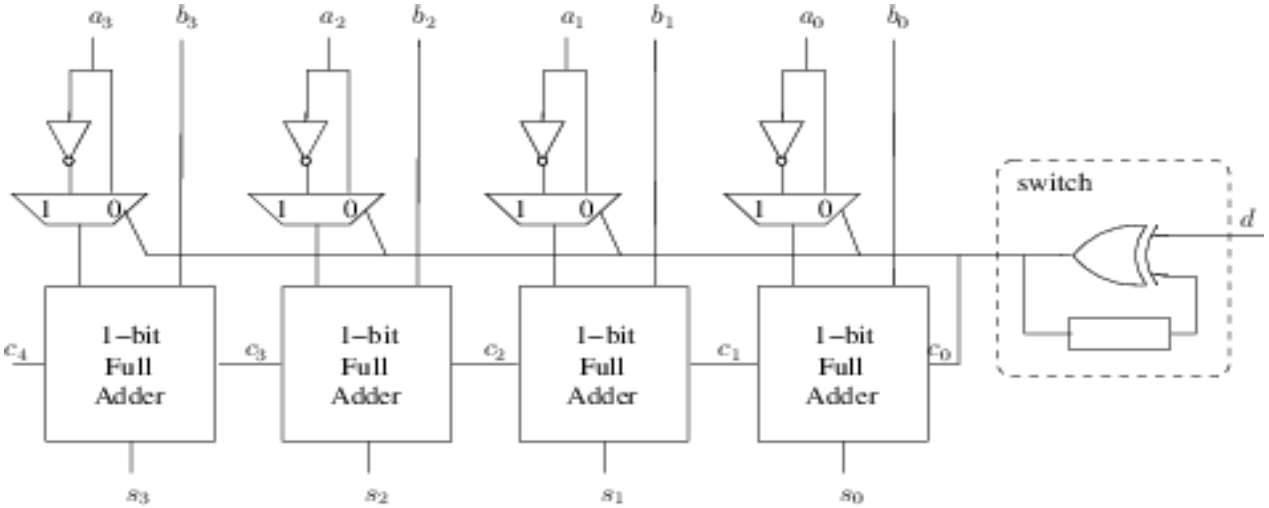
AIM : To design and implement a ripple carry adder.

SOFTWARE USED : Logic gate simulator TRUTH TABLE :

Truth table of ripple carry adder

A1	A2	A3	A4	B4	B3	B2	B1	S4	S3	S2	S1	carry
0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	1
1	0	1	0	1	0	1	0	0	1	0	0	1
1	1	0	0	1	1	0	0	1	0	0	0	1
1	1	1	0	1	1	1	0	1	1	0	0	1
1	1	1	1	1	1	1	1	1	1	1	0	1

CIRCUIT DIAGRAM :



PROCEDURE:

Ripple Carry Adder works in different stages.
Each full adder takes the carry-in as input and produces carry-out and sum bit as output.
The carry-out produced by a full adder serves as carry-in for its adjacent most significant full adder.
When carry-in becomes available to the full adder, it activates the full adder.
After full adder becomes activated, it comes into operation.

OUTPUT :

(Snapshot)

RESULT : Hence, ripple carry adder was constructed in the simulator & it's characteristics were studied.

EXPERIMENT NO. 11 DATE:

CARRY LOOK AHEAD ADDER

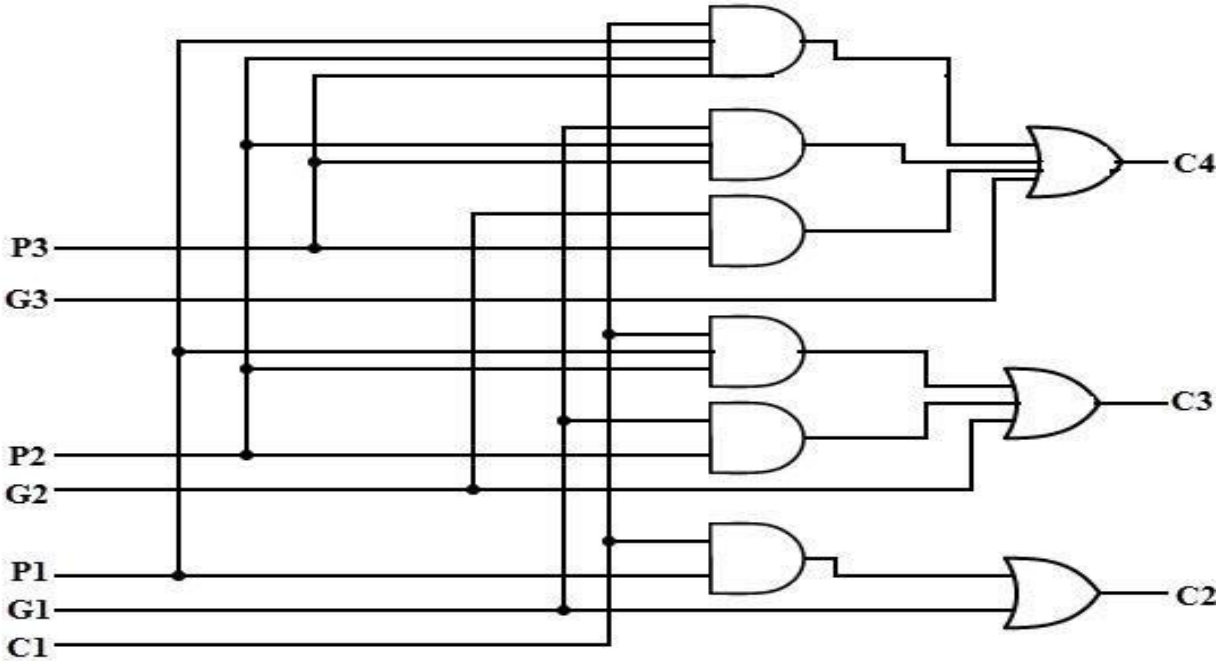
AIM : To design and implement a carry look ahead adder.

SOFTWARE USED : Logic gate simulator

TRUTH TABLE :

A	B	Ci	C i +1	Condition
0	0	0	0	No carry generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No carry propagate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry generate
1	1	1	1	

CIRCUIT DIAGRAM :



PROCEDURE:

In order to perform these repeated functions, adder circuits are required and those are half adder, full adder, carry lookahead adder.

Calculate every digit position to know whether that position is propagating a carry bit that comes from its right position.

Then combine the calculated values to produce the output for every set of digits where the group generates a propagation bit that comes from the right position.

OUTPUT:

(Snapshot)

RESULT : Hence, carry look ahead adder was constructed & char. were studied.

EXPERIMENT NO. 12 DATE:

2-BIT MULTIPLIER

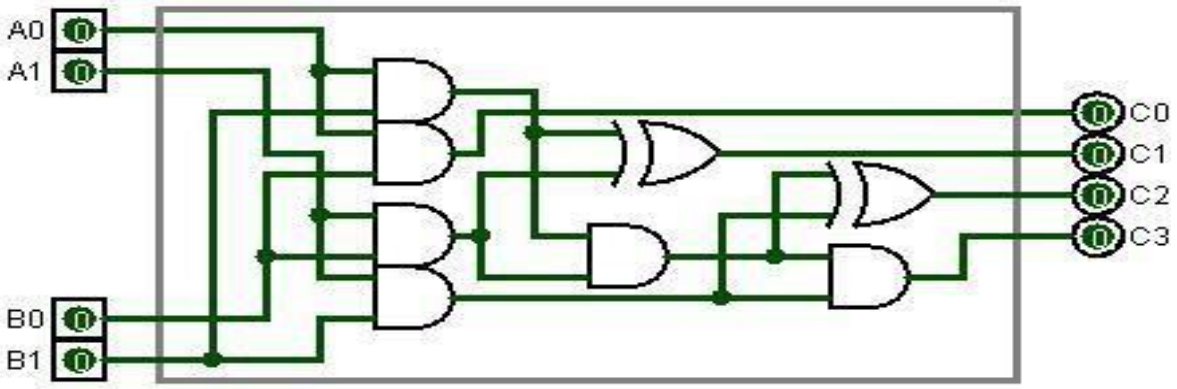
AIM : To design and implement a carry look ahead adder.

SOFTWARE USED : Logic gate simulator

TRUTH TABLE :

	Inputs				Outputs				
	A	B	C	D	W	X	Y	Z	
0×0	0	0	0	0	0	0	0	0	0
0×1	0	0	0	1	0	0	0	0	0
0×2	0	0	1	0	0	0	0	0	0
0×3	0	0	1	1	0	0	0	0	0
1×0	0	1	0	0	0	0	0	0	0
1×1	0	1	0	1	0	0	0	1	1
1×2	0	1	1	0	0	0	1	0	2
1×3	0	1	1	1	0	0	1	1	3
2×0	1	0	0	0	0	0	0	0	0
2×1	1	0	0	1	0	0	1	0	2
2×2	1	0	1	0	0	1	0	0	4
2×3	1	0	1	1	0	1	1	0	6
3×0	1	1	0	0	0	0	0	0	0
3×1	1	1	0	1	0	0	1	1	3
3×2	1	1	1	0	0	1	1	0	6
3x3	1	1	1	1	1	0	0	1	9

CIRCUIT DIAGRAM :



PROCEDURE:

To multiply two binary numbers, AND gates, shifters and adders are required.

Product of N*M bit binary numbers in of (N+M) bits.

N*M AND gates are required to generate partial products of two M*N bit binary numbers.

Number of adders required = N+M-2

Speed limiting factor here is to sum up partial products.

OUTPUT :

(Snapshot)

RESULT : 2- bit binary multiplier was constructed & studied.

EXPERIMENT NO. 13 DATE:

BINARY PARALLEL ADDER & SUBTRACTOR

AIM : To design and implement a binary parallel adder & subtractor.

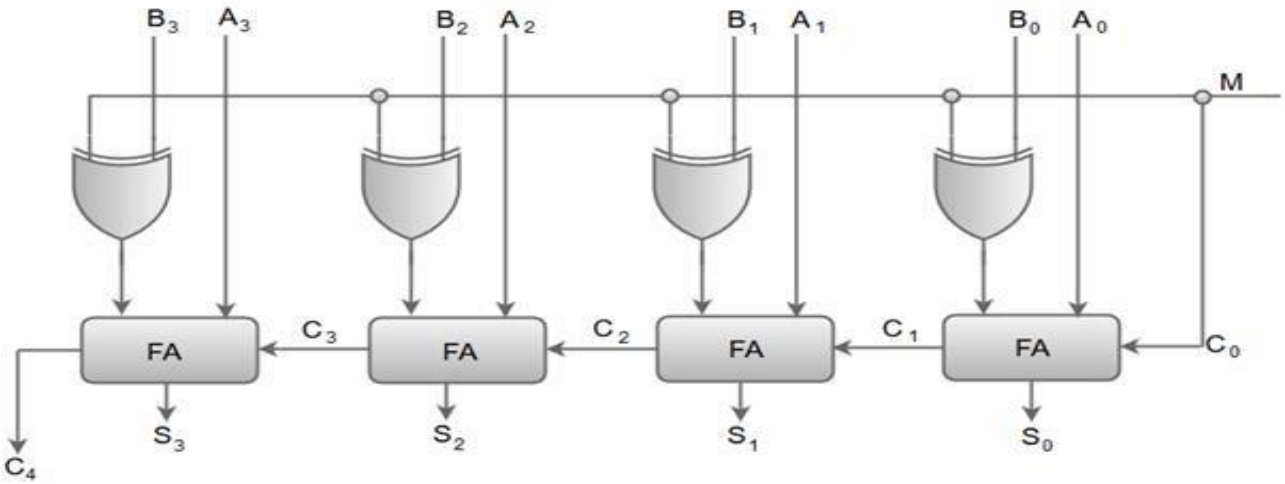
SOFTWARE USED : Logic gate simulator

TRUTH TABLE :

	B3	B2	B1	B0	M	X3	X2	X1	X0
Addition	0	0	0	0	**0**	0	0	0	0
	0	0	0	1	**0**	0	0	0	1
	0	0	1	0	**0**	0	0	1	0
	0	0	1	1	**0**	0	0	1	1
	0	1	0	0	**0**	0	1	0	0
	0	1	0	1	**0**	0	1	0	1
	0	1	1	0	**0**	0	1	1	0
	0	1	1	1	**0**	0	1	1	1
	1	0	0	0	**0**	1	0	0	0
	1	0	0	1	**0**	1	0	0	1
Subs traction	0	0	0	0	**1**	1	0	0	1
	0	0	0	1	**1**	1	0	0	0
	0	0	1	0	**1**	0	1	1	1
	0	0	1	1	**1**	0	1	1	0
	0	1	0	0	**1**	0	1	0	1
	0	1	0	1	**1**	0	1	0	0
	0	1	1	0	**1**	0	0	1	1
	0	1	1	1	**1**	0	0	1	0
	1	0	0	0	**1**	0	0	0	1
	1	0	0	1	**1**	0	0	0	0

CIRCUIT DIAGRAM :

4 bit adder-subtractor:



PROCEDURE:

the full adder FA1 adds A1 and B1 along with the carry C1 to generate the sum S1 (the first bit of the output sum) and the carry C2 which is connected to the next adder in chain.

Next, the full adder FA2 uses this carry bit C2 to add with the input bits A2 and B2 to generate the sum S2(the second bit of the output sum) and the carry C3 which is again further connected to the next adder in chain and so on.

The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and Bn to generate the last bit of the output along last carry bit Cout.

The parallel binary subtractor is formed by combination of all full adders with subtrahend complement input.

This operation considers that the addition of minuend along with the 2's complement of the subtrahend is equal to their subtraction.

Firstly the 1's complement of B is obtained by the NOT gate and 1 can be added through the carry to find out the 2's complement of B. This is further added to A to carry out the arithmetic subtraction.

The process continues till the last full adder FAn uses the carry bit Cn to add with its input An and 2's complement of Bn to generate the last bit of the output along last carry bit Cout.

OUTPUT :

(Snapshot)

RESULT : Binary parallel adder & subtractor was constructed & it's char. was studied.

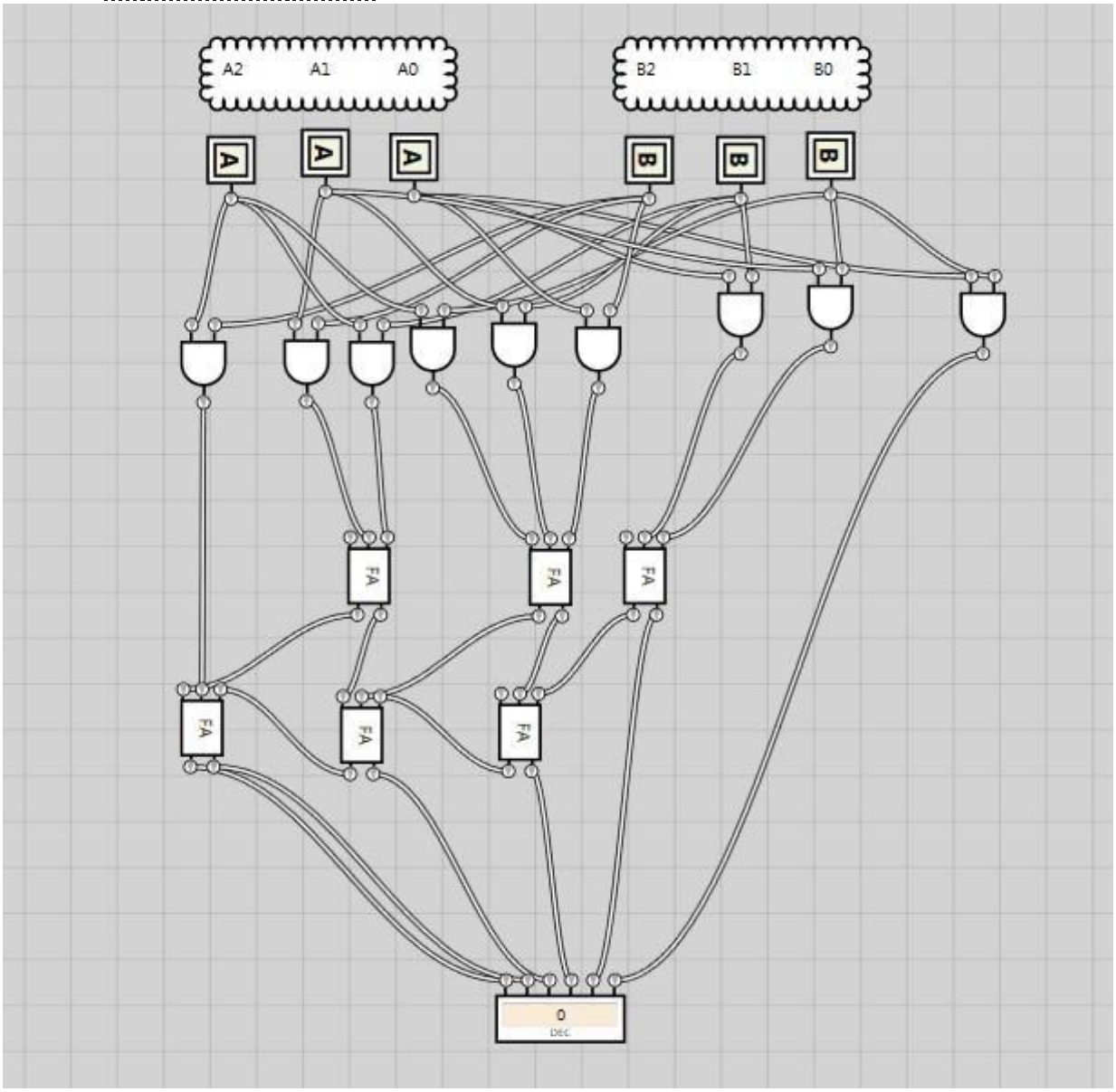
EXPERIMENT NO. 14 DATE:

CARRY SAVE MULTIPLIER

AIM : To design and implement a carry save multiplier.

SOFTWARE USED : Logic gate simulator

CIRCUIT DIAGRAM :



PROCEDURE:

A starting point for any consideration of machine processing of numerical data is a determination of the form in which the data will be presented. To begin, a numbering system must be chosen. Historically, machine processing of numbers has been related to numbers represented in base 2 (binary), 8 (octal), 10 (decimal) and 16 (hexadecimal). Because the binary numbering system has become the overwhelming choice for contemporary machine processing of numerical data, that system will be used herein to illustrate the methodology of the invention. It should, however, be apparent to those skilled in the art that the inventive method can readily be extended to other numbering systems.

OUTPUT:

(Snapshot)

RESULT : Carry save multiplier was constructed in simulator & char. were studied.

EXPERIMENT NO. 15 DATE:

MULTIPLEXER

AIM : To design and implement a Multiplexer.

SOFTWARE USED : Logic gate simulator TRUTH TABLE :

(4*1)

s_1	s_0	x_3	x_2	x_1	x_0	y
0	0	x	x	x	0	0
0	0	x	x	x	1	1
0	1	x	x	0	x	0
0	1	x	x	1	x	1
1	0	x	0	x	x	0
1	0	x	1	x	x	1
1	1	0	x	x	x	0
1	1	1	x	x	x	1

(8*1)

Select Data Inputs			Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

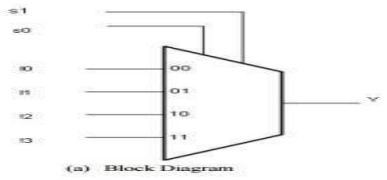
(16*1)

TRUTH TABLE

S0	S1	S2	S3	$\overline{E}$	SELECTED CHANNEL
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

H= High Level
L= Low Level
X= Don't Care

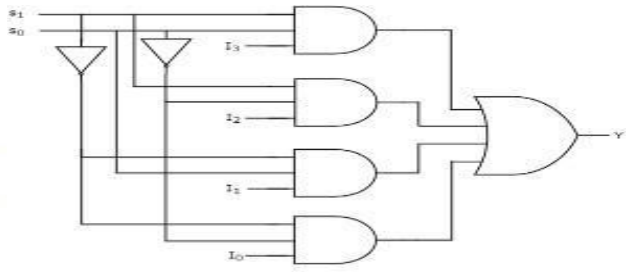
CIRCUIT DIAGRAM :

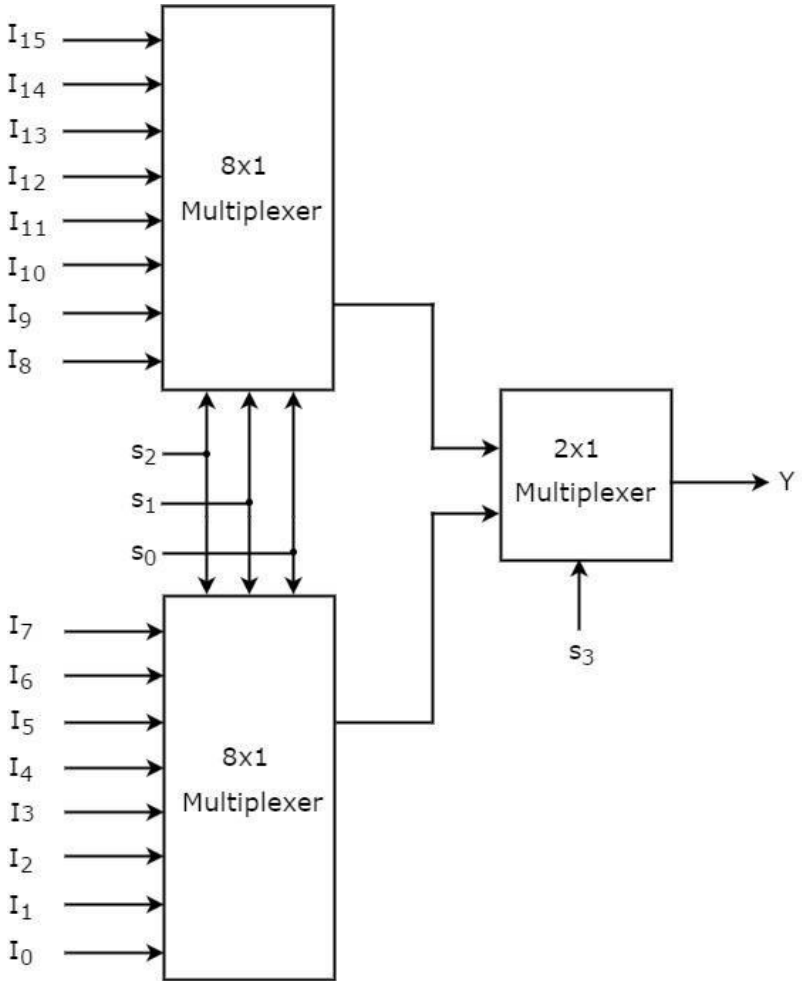


(a) Block Diagram

s1	s0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

(b) Function Table





PROCEDURE:

2×1 Multiplexer:

In 2×1 multiplexer, there are only two inputs, i.e., A0 and A1, 1 selection line, i.e., S0 and single outputs, i.e., Y. On the basis of the combination of inputs which are present at the selection line S0, one of these 2 inputs will be connected to the output.

4×1 Multiplexer:

In the 4×1 multiplexer, there is a total of four inputs, i.e., A0, A1, A2, and A3, 2 selection lines, i.e., S0 and S1 and single output, i.e., Y. On the basis of the combination of inputs that are present at the selection lines S0 and S1, one of these 4 inputs are connected to the output.

8 to 1 Multiplexer

In the 8 to 1 multiplexer, there are total eight inputs, i.e., A0, A1, A2, A3, A4, A5, A6, and A7, 3 selection lines, i.e., S0, S1and S2 and single output, i.e., Y. On the basis of the combination of inputs that are present at the selection lines S0, S1, and S2, one of these 8 inputs are connected to the output.

16 to 1 Multiplexer:

In the 16 to 1 multiplexer, there are total of 16 inputs, i.e., A0, A1, …, A16, 4 selection lines, i.e., S0, S1, S2, and S3 and single output, i.e., Y. On the basis of the combination of inputs that are present at the selection lines S0, S1, and S2, one of these 16 inputs will be connected to the output.

OUTPUT :

(Snapshot)

RESULT : 4:1, 8:1, 16:1 Multiplexers were constructed & studied.

EXPERIMENT NO. 16 DATE:

ARITHMETIC LOGIC UNIT (ALU)

AIM : To implement Arithmetic and Logical Unit which perform **Addition and subtraction arithmetic operations** and **AND, OR, XOR logical operations**.

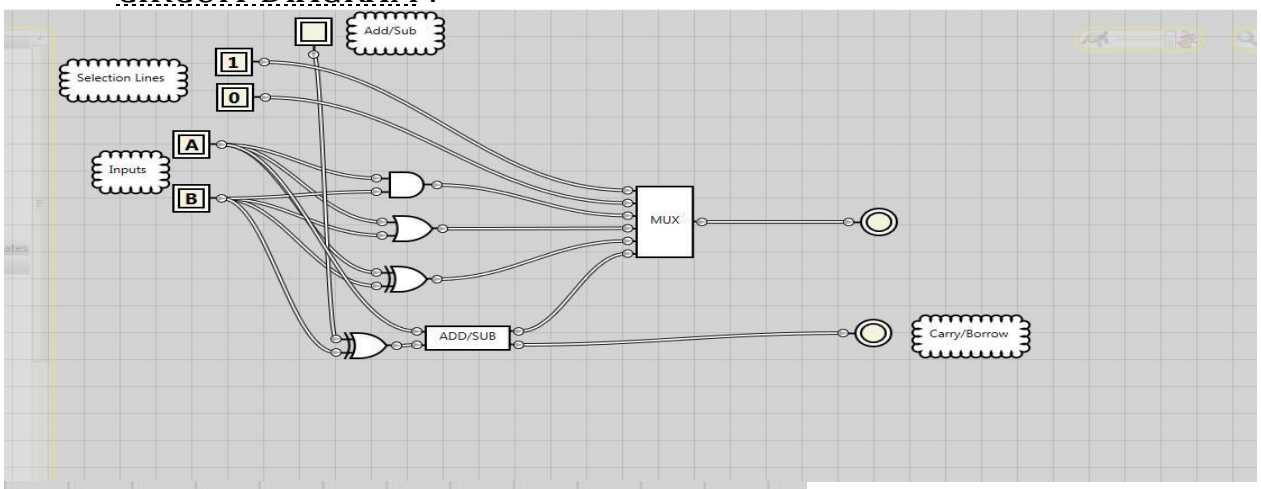
SOFTWARE USED : Logic gate simulator

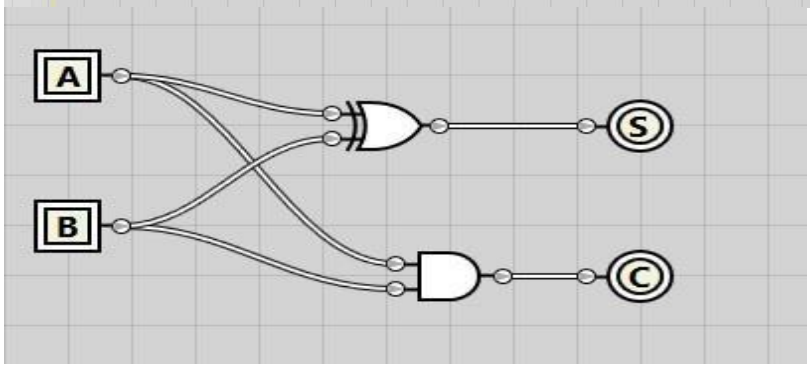
TRUTH TABLE :

TABLE 1. TRUTH TABLE OF ALU

S_2	S_1	S_0	**OPERATION**
0	0	0	OR
0	0	1	XNOR
0	1	0	XOR
0	1	1	AND
1	0	0	INCREMENT
1	0	1	ADDITION
1	1	0	DECREMENT
1	1	1	SUBTRACTION

CIRCUIT DIAGRAM :





PROCEDURE:

An ALU performs basic arithmetic and logic operations. Examples of arithmetic operations are addition, subtraction, multiplication, and division. Examples of logic operations are comparisons of values such as NOT, AND, and OR.

All information in a computer is stored and manipulated in the form of binary numbers, i.e. 0 and 1. Transistor switches are used to manipulate binary numbers since there are only two possible states of a switch: open or closed. An open transistor, through which there is no current, represents a 0. A closed transistor, through which there is a current, represents a 1. Operations can be accomplished by connecting multiple transistors. One transistor can be used to control a second one - in effect, turning the transistor switch on or off depending on the state of the second transistor. This is referred to as a gate because the arrangement can be used to allow or stop a current.

RESULT : Arithmetic logic unit was designed & studied.