* 1. **Aim:**

LAB Manual PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.02**

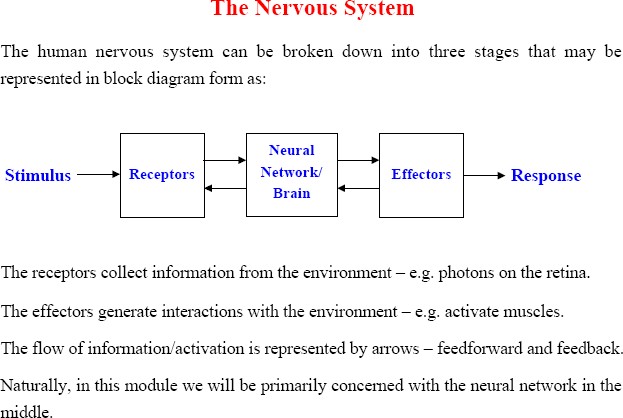
Implementation of logic gate (AND, OR, NOT, NAND, NOR ) using Mc-Culloch Pitts (MCP) model.

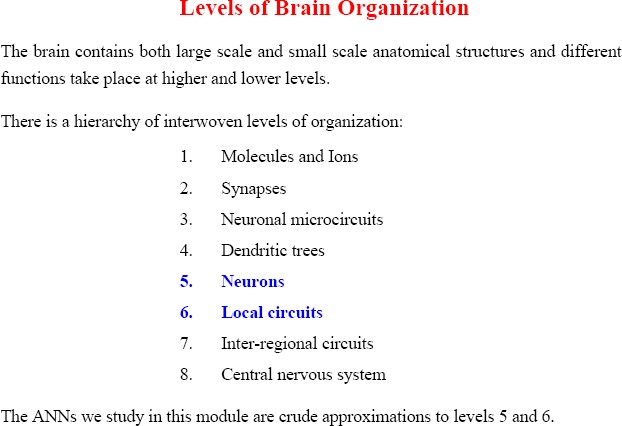
* 1. **Prerequisite:**
     1. Theoretical knowledge of Mc-Culloch Pitts model of neural network.
     2. Knowledge of logic gates.
     3. Different programming language structure overview.
  2. **Outcome:**

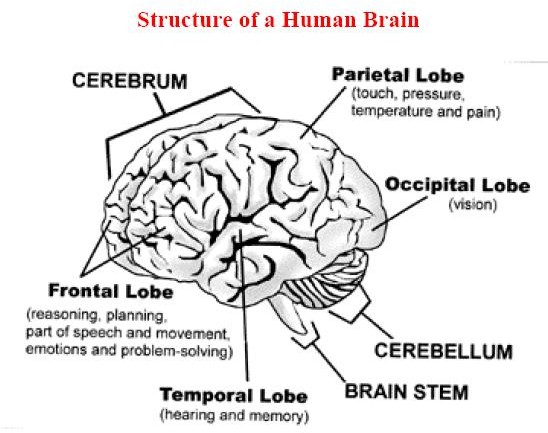
**After successful completion of this experiment students will be able to**

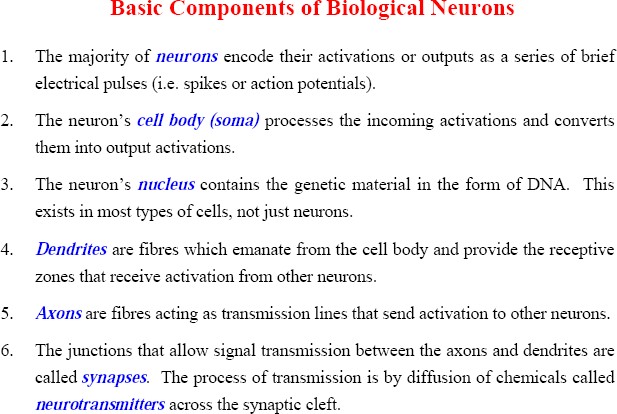
* + 1. Apply MCP Neuron Model to solve simple logic examples.
    2. Design neural network by making use of MCP neuron model.
  1. **Theory:**

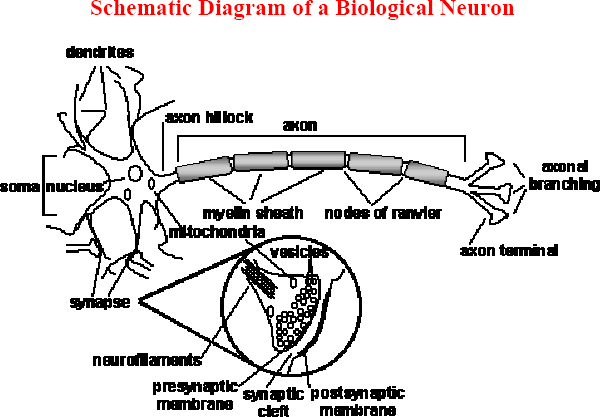
**A.4.1. Biological Neuron**:

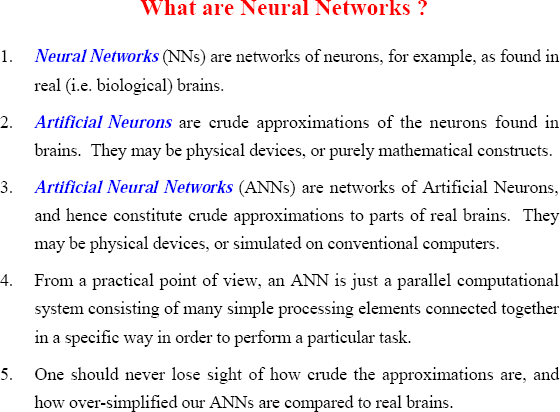


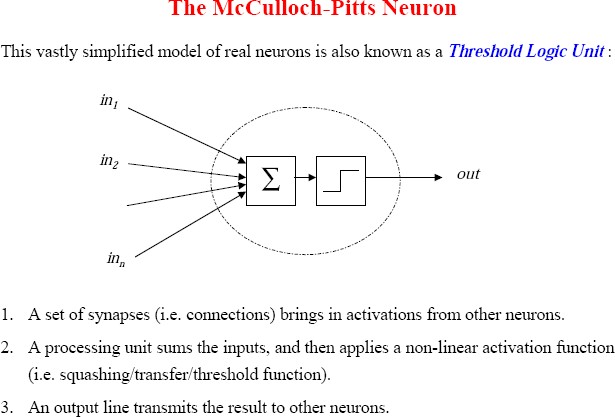


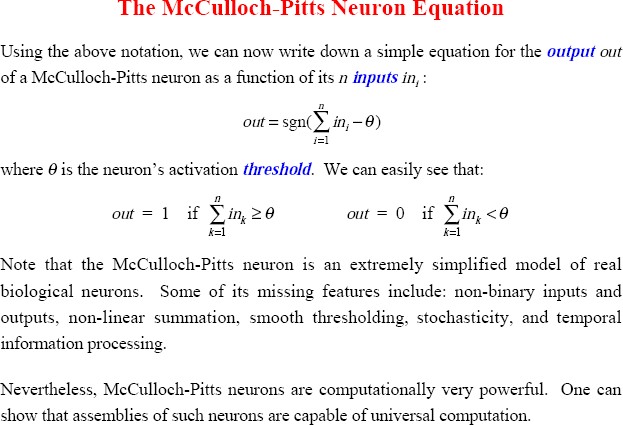


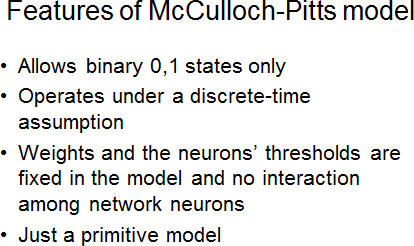


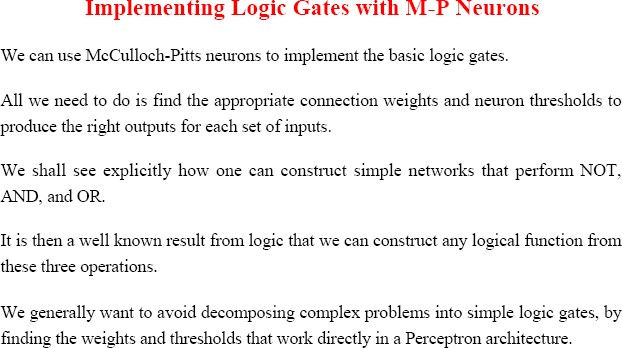


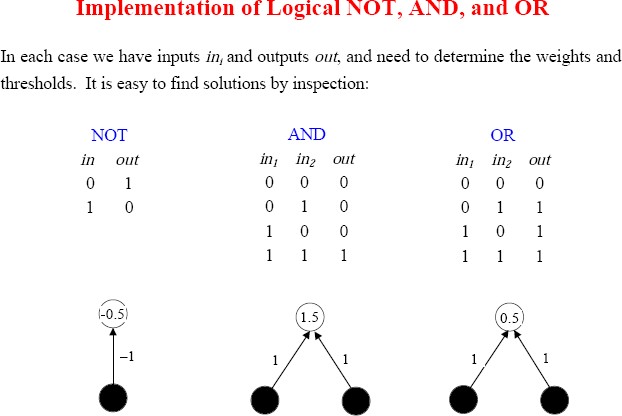


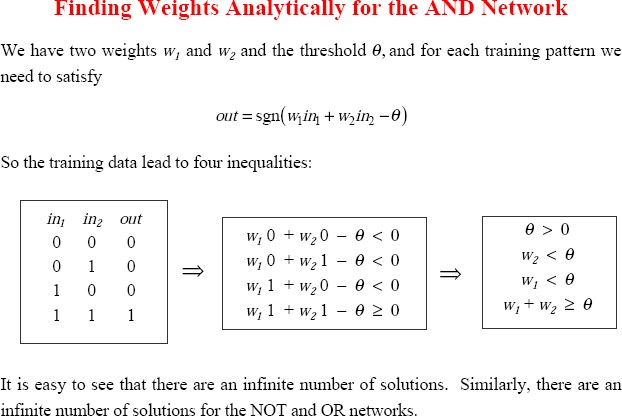












# Procedure/Algorithm:

* + 1. **:**
       1. **Read the inputs for a logic gate**
       2. **Read weights for each input**
       3. **Read threshold value for the logic gate**
       4. **Determine Yin; total input to the logical gate.**
       5. **Compare Yin with threshold value.**
       6. **Generate the output as 1 if Yin ≥ threshold value; else 0.**
       7. **Repeat the same procedure (Step 1 to 6) for all logic gates.**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.: C026 | Name: Anirbaan Ghatak |
| Class : B | Batch : B1 |
| Date of Experiment: 09/08/2023 | Date of Submission: 14/08/2023 |
| Grade : | Time of Submission: |
| Date of Grading: |  |

# Software Code written by student:

def threshold\_neuron(weights, inputs, threshold):

    weighted\_sum = sum(w \* x for w, x in zip(weights, inputs))

    if weighted\_sum >= threshold:

        return 1

    else:

        return 0

def and\_neuron(x, y):

    weights = [1, 1]

    threshold = 2

    return threshold\_neuron(weights, [x, y], threshold)

def or\_neuron(x, y):

    weights = [1, 1]

    threshold = 1

    return threshold\_neuron(weights, [x, y], threshold)

def not\_neuron(x):

    weights = [1]

    threshold = 1

    return threshold\_neuron(weights, [x], threshold)

def nand\_neuron(x, y):

    weights = [-1, -1]

    threshold = -2

    return threshold\_neuron(weights, [x, y], threshold)

def nor\_neuron(x, y):

    weights = [-1, -1]

    threshold = 0

    return threshold\_neuron(weights, [x, y], threshold)

input\_x = 0

input\_y = 1

print(and\_neuron(input\_x, input\_y))

print(or\_neuron(input\_x, input\_y))

print(not\_neuron(input\_x))

print(nand\_neuron(input\_x, input\_y))

print(nor\_neuron(input\_x, input\_y))

# Input and Output:

# 

# 

# 

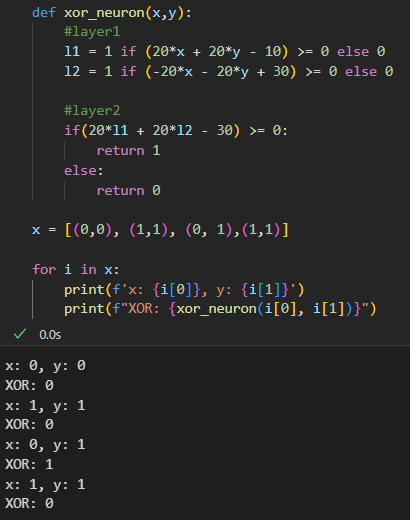
* 1. **Observations and learning:**

The code shows how logic gate simulations using threshold neurons, mirrors neural network principles. It employs modular functions for common computations, showcasing the weight-threshold for different gates. This implementation a fundamental grasp of neural network behavior in emulating logical operations.

# Conclusion:

It introduces us to basic decision-making units using virtual neurons, demonstrating their versatility in mimicking fundamental logic gates. Thus, diving deeper into the neural network and their application in modern computing

# Question of Curiosity

* + 1. **Implement ANDNOT Function using Mc-Culloch Pitts neuron model.**
       - **Provided in the above code**
    2. **Implemet XOR logic using Mc-Culloch Pitts neuron model.**
       - ****
    3. **What are the limitations of Mc-Culloch Pitts neuron model?**

The limitations are

* + **Binary Output**: It produces only binary (0 or 1) output, limiting its ability to represent complex information or continuous data.
  + **Fixed Weights and Thresholds**: It lacks the ability to adjust weights and thresholds during learning, which hinders its adaptability to changing patterns or tasks.
  + **Single Layer**: It can only represent linearly separable functions, limiting its capacity to model more complex relationships.
  + **No Learning Mechanism**: The model doesn't have a learning mechanism, so it can't improve its performance through experience or data.
  + **No Activation Functions**: It doesn't incorporate activation functions like those in modern neurons, limiting its ability to represent more nuanced behaviors.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*