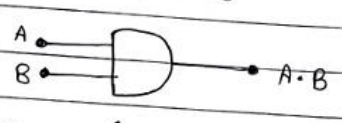
> Basic gates

1) AND gate

o/p is high only if all inputs are high

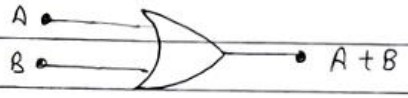


Boolean eq^ : $A \cdot B$

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

2) OR gate

o/p is high ~~on~~ if atleast one i/p is high

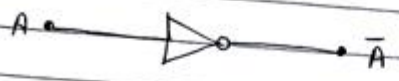


Boolean eq^ : $A + B$

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

3) NOT gate

it inverts the binary i/p i.e o/p is the opposite of the i/p.



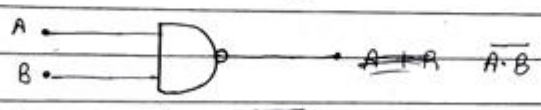
Boolean eq^ : $\bar{A}$

A	$\bar{A}$
0	1
1	0

> Universal gates

1) NAND gate

o/p is ~~#~~ low only when all inputs are high

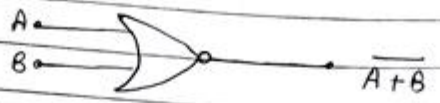


Boolean eq^ : $\overline{A \cdot B}$

A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

2) NOR gate

o/p is high only when all inputs are low

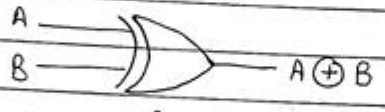


Boolean eq^n : $\overline{A+B}$

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

> Ex-OR gate (Exclusive OR) / XOR

o/p is high only when inputs are different.

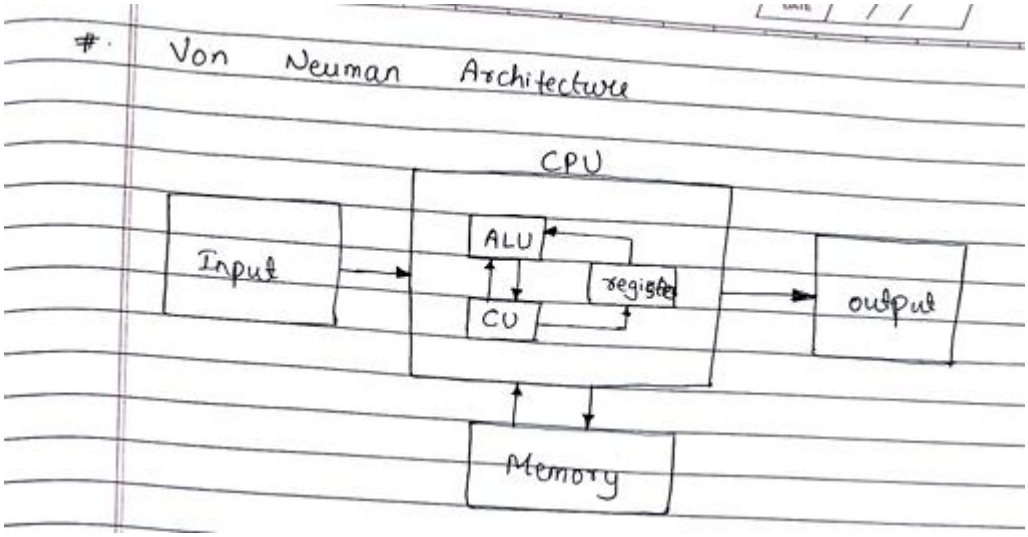


Boolean eq^n : $A \oplus B = A\bar{B} + \bar{A}B$

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

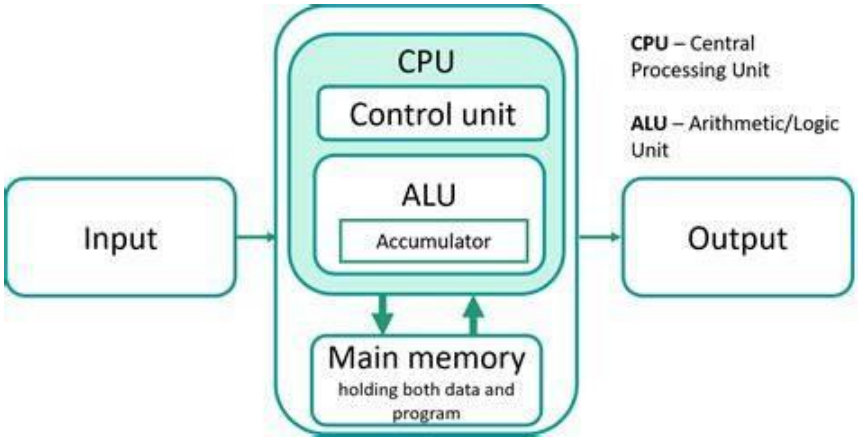
if $A = B \Rightarrow$ o/p low

if $A \neq B \Rightarrow$ o/p high



Difference between Restoring and Non-Restoring Division Algorithms

Point	Restoring Division	Non-Restoring Division
1. Procedure	If subtraction gives a negative result, the divisor is **restored** (added back) in the next step.	If subtraction gives a negative result, no restoring is done — instead, correction happens in the next cycle.
2. Complexity	Needs **extra step** for restoring the remainder, so more operations.	Faster since it avoids restoring steps.
3. Hardware	Simpler control logic but requires **more steps**.	Slightly more complex control logic but **fewer steps overall**.
4. Speed	**Slower** due to restoring after negative remainders.	**Faster** as it directly adjusts in the next iteration.
5. Usage	Easier to understand and implement, mostly used for teaching and simple hardware.	Preferred in modern processors due to efficiency.



It is a computer architecture where **program instructions and data are stored in the same memory** and accessed through a single bus.

🔷 Key points:

- Uses **stored-program concept**.
- Single memory for **data + instructions**.
- Consists of **CPU (ALU + Control Unit), Memory, and I/O**.
- Executes instructions **sequentially (fetch → decode → execute)**.