Unit-I: Logic Gates and Boolean Algebra
Short Answer Questions:

Q. What is POS? Explain. (Nov 22)
Ans. **POS (Product of Sums)** is a Boolean expression format where variables are grouped in OR terms and then ANDed together. It is used to simplify logic functions in digital circuits.
POS ਇੱਕ Boolean ਅਭਿਵੈਕਤੀ ਰੂਪ ਹੈ ਜਿਸ ਵਿੱਚ ਚੇਤਰਾਂ ਨੂੰ OR ਕੀਤਾ ਜਾਂਦਾ ਹੈ ਅਤੇ ਫਿਰ AND ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਡਿਜ਼ੀਟਲ ਸਰਕਟਾਂ ਵਿੱਚ ਲਾਜਿਕ ਸਧਾਰਨ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

Q. Draw the circuit diagram and truth table of a XOR gate. (Nov 22)
Ans. **XOR Gate** outputs HIGH (1) when the number of HIGH inputs is odd. It is used in parity checkers and adders.
XOR ਗੇਟ ਤਦੋਂ HIGH (1) ਆਉਟਪੁੱਟ ਦਿੰਦਾ ਹੈ ਜਦੋਂ HIGH ਇਨਪੁੱਟ ਦੀ ਗਿਣਤੀ ਵੀਰ ਹੋਵੇ। ਇਹ parity checkers ਅਤੇ adders ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

Q. What is SOP form? (Nov 23)
Ans. **SOP (Sum of Products)** is a form of expressing Boolean logic where multiple AND terms are ORed together. It helps in designing simplified logic circuits.
SOP ਇੱਕ Boolean ਲਾਜਿਕ ਰੂਪ ਹੈ ਜਿਸ ਵਿੱਚ AND ਟਰਮਾਂ ਨੂੰ OR ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਸਰਲ ਲਾਜਿਕ ਸਰਕਟ ਬਣਾਉਣ ਵਿੱਚ ਮਦਦ ਕਰਦਾ ਹੈ।

Q. Write at least two applications of Logic Gates. (Nov 24)
Ans. **Logic Gates** are used in digital circuits for decision making; applications include arithmetic logic units and signal processing.
ਲਾਜਿਕ ਗੇਟ ਡਿਜ਼ੀਟਲ ਸਰਕਟਾਂ ਵਿੱਚ ਫੈਸਲਾ ਲੈਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਇਹ arithmetic logic units ਅਤੇ signal processing ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

Q. Simplify the following Boolean expression XY+X(Y+Z) +Y(Y+Z). (Nov 24)
Ans. The Boolean expression **XY + X(Y+Z) + Y(Y+Z)** simplifies to **X + Y**, showing redundancy elimination using Boolean laws.
Boolean ਅਭਿਵੈਕਤੀ **XY + X(Y+Z) + Y(Y+Z)** ਨੂੰ ਸਧਾਰਨ ਕਰਕੇ **X + Y** ਮਿਲਦਾ ਹੈ। ਇਹ Boolean laws ਦੀ ਵਰਤੋਂ ਨਾਲ redundancy ਹਟਾਉਣ ਦਿਖਾਉਂਦਾ ਹੈ।

Q. What is K-Map? (Nov 24)
Ans. **K-Map (Karnaugh Map)** is a graphical tool used to simplify Boolean expressions by grouping adjacent 1's. It minimizes the number of logic gates in digital circuits.
K-Map ਇੱਕ ਗ੍ਰਾਫੀਕਲ ਟੂਲ ਹੈ ਜੋ Boolean ਅਭਿਵੈਕਤੀਆਂ ਨੂੰ ਸਧਾਰਨ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਲਾਜਿਕ ਗੇਟਾਂ ਦੀ ਗਿਣਤੀ ਘਟਾਉਂਦਾ ਹੈ।

Long Answer Questions:

Q. a) What is Boolean Algebra? Explain the role of De Morgan's theorem in Boolean Algebra.
b) Explain three input NOR gate with truth table and logic circuit. (Nov 22)
Ans. **a) Boolean Algebra and De Morgan's Theorem:**
Boolean Algebra is a mathematical framework used to analyze and simplify digital (binary) logic circuits. It involves variables that take values of 0 (false) or 1 (true) and uses logical operations such as AND (·), OR (+), and NOT (¬ or ‘). Boolean algebra helps design and optimize logic gates and digital systems.

De Morgan's Theorems are essential rules in Boolean algebra:

1. ¬(A + B) = ¬A · ¬B
2. ¬(A · B) = ¬A + ¬B

These theorems help in simplifying complex logic expressions and converting between AND-OR and NAND-NOR logic, making them critical for circuit minimization and implementation using universal gates.

a) ਬੂਲੀਅਨ ਬੀਜਗਣਿਤ ਅਤੇ ਡੀ ਮੋਰਗਨ ਦੇ ਸੁਤਰ:

Boolean Algebra ਇੱਕ ਗਣਿਤੀਕ ਢਾਂਚਾ ਹੈ ਜੋ ਡਿਜ਼ੀਟਲ (ਬਾਈਨਰੀ) ਲਾਜਿਕ ਸਰਕਟਾਂ ਨੂੰ ਵਿਸ਼ਲੇਸ਼ਣ ਅਤੇ ਸਰਲ ਬਣਾਉਣ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

ਇਸ ਵਿੱਚ ਵੇਰੀਏਬਲ 0 (ਝੂਠ) ਜਾਂ 1 (ਸੱਚ) ਮੁੱਲ ਲੈਂਦੇ ਹਨ ਅਤੇ AND (·), OR (+), ਅਤੇ NOT (¬ ਜਾਂ ') ਵਰਗੀਆਂ ਲਾਜਿਕ ਓਪਰੇਸ਼ਨਾਂ ਦੀ ਵਰਤੋਂ ਹੁੰਦੀ ਹੈ।

Boolean Algebra ਡਿਜ਼ੀਟਲ ਲਾਜਿਕ ਸਰਕਟਾਂ ਅਤੇ ਲਾਜਿਕ ਗੇਟਾਂ ਦੇ ਡਿਜ਼ਾਈਨ ਅਤੇ ਆਪਟੀਮਾਈਜ਼ੇਸ਼ਨ ਲਈ ਬਹੁਤ ਲਾਭਕਾਰੀ ਹੁੰਦੀ ਹੈ।

De Morgan ਦੇ ਸੁਤਰ (Theorems):

1. ¬(A + B) = ¬A · ¬B
2. ¬(A · B) = ¬A + ¬B

ਇਹ ਸੁਤਰ ਲਾਜਿਕ ਅਭਿਵੈਕਤੀਆਂ ਨੂੰ ਸਧਾਰਨ ਕਰਨ ਅਤੇ AND-OR ਨੂੰ NAND-NOR ਵਿੱਚ ਬਦਲਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

ਇਹ ਸਰਕਟਾਂ ਨੂੰ ਯੂਨੀਵਰਸਲ ਗੇਟਾਂ ਨਾਲ ਲਾਗੂ ਕਰਨ ਲਈ ਬਹੁਤ ਉਪਯੋਗੀ ਹਨ।

b) Three-input NOR Gate:

A **NOR gate** outputs true (1) only when all its inputs are false (0). It is the combination of an OR gate followed by a NOT gate.

Truth Table:

A B C Output (Y = ¬(A + B + C))

0 0 0 1
0 0 1 0
0 1 0 0
0 1 1 0
1 0 0 0
1 0 1 0
1 1 0 0
1 1 1 0

Logic Circuit: The inputs A, B, and C connect to a 3-input OR gate, and the output of the OR gate goes to a NOT gate, forming the NOR logic.

b) ਤਿੰਨ ਇਨਪੁੱਟ ਵਾਲਾ NOR ਗੇਟ (Three-Input NOR Gate):

NOR Gate ਤਦੋਂ ਹੀ ਆਉਟਪੁੱਟ 1 ਦਿੰਦਾ ਹੈ ਜਦੋਂ ਤਿੰਨੇ ਇਨਪੁੱਟ 0 ਹੋਣ।

ਇਹ OR ਗੇਟ ਦੇ ਬਾਅਦ NOT ਲਗਾ ਕੇ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ।

Truth Table:

A B C Y = ¬(A + B + C)

0 0 0 1
0 0 1 0

A B C Y = ¬(A + B + C)
0 1 0 0
0 1 1 0
1 0 0 0
1 0 1 0
1 1 0 0
1 1 1 0

ਲਾਜਿਕ ਸਰਕਟ (Logic Circuit):

ਇਨਪੁੱਟ A, B, ਅਤੇ C ਨੂੰ 3-Input OR Gate ਨਾਲ ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ।

ਫਿਰ OR Gate ਦਾ ਆਉਟਪੁੱਟ NOT Gate ਨੂੰ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ।

ਇਸ ਤਰੀਕੇ ਨਾਲ NOR Gate ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ।

Q. What is a Logic gate? Which gates are called as the universal gates? What are its advantages? Implement the basic gates using Universal gates. (Nov 23),(Nov 24)

Ans. A **Logic gate** is a fundamental building block of digital electronics. It performs a basic logical function on one or more binary inputs to produce a single binary output. The common logic gates are AND, OR, NOT, NAND, NOR, XOR, and XNOR. They are used in designing digital circuits such as computers, calculators, and memory devices.

Universal gates are NAND and NOR gates. They are termed "universal" because any other logic gate (AND, OR, NOT) can be constructed using only NAND or only NOR gates. This simplifies circuit design and manufacturing, as one type of gate can be used to implement complex logic functions.

Advantages of Universal Gates:

- Cost-effective fabrication
- Simpler and more flexible circuit design
- Greater fault tolerance and easier replacement in IC design

Implementation of Basic Gates Using NAND:

- **NOT Gate**: Connect both inputs of a NAND gate to a single input A → Output: A'
- **AND Gate**: Use two NAND gates. First NAND gate gives A·B', second NAND of the first output gives (A·B)''
- **OR Gate**: Apply De Morgan's Law: A + B = (A'·B')'. So, first invert A and B using NAND as NOT, then apply NAND to outputs.

Similar logic applies to building gates using NOR.

ਲਾਜਿਕ ਗੇਟ ਡਿਜ਼ੀਟਲ ਇਲੈਕਟ੍ਰੌਨਿਕਸ ਦਾ ਮੂਲ ਭਾਗ ਹੈ। ਇਹ ਇੱਕ ਜਾਂ ਵਧੇਰੇ **ਬਾਈਨਰੀ ਇਨਪੁੱਟ** 'ਤੇ ਆਧਾਰਤ **ਇੱਕ ਬਾਈਨਰੀ ਆਉਟਪੁੱਟ** ਦਿੰਦਾ ਹੈ। ਸਧਾਰਨ ਲਾਜਿਕ ਗੇਟਾਂ ਵਿੱਚ AND, OR, NOT, NAND, NOR, XOR ਅਤੇ XNOR ਸ਼ਾਮਲ ਹਨ। ਇਹ ਕੰਪਿਊਟਰ, ਕੈਲਕੂਲੇਟਰ ਅਤੇ ਮੈਮੋਰੀ ਜਿਹੇ ਡਿਜ਼ੀਟਲ ਸਰਕਟਾਂ ਦੇ ਡਿਜ਼ਾਈਨ ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

ਯੂਨੀਵਰਸਲ ਗੇਟ (Universal Gates): NAND ਅਤੇ NOR ਗੇਟ ਨੂੰ ਯੂਨੀਵਰਸਲ ਗੇਟ ਕਿਹਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਇਸ ਲਈ ਕਹੇ ਜਾਂਦੇ ਹਨ ਕਿਉਂਕਿ **ਹਰ ਹੋਰ ਲਾਜਿਕ ਗੇਟ (AND, OR, NOT)** ਨੂੰ ਸਿਰਫ਼ NAND ਜਾਂ ਸਿਰਫ਼ NOR ਨਾਲ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ। ਇਹ ਸਰਕਟ ਡਿਜ਼ਾਈਨ ਅਤੇ ਉਤਪਾਦਨ ਨੂੰ ਆਸਾਨ ਅਤੇ ਘੱਟ-ਖਰਚੀਲਾ ਬਣਾਉਂਦੇ ਹਨ।

ਯੂਨੀਵਰਸਲ ਗੇਟਾਂ ਦੇ ਲਾਭ (Advantages):

- ਘੱਟ ਲਾਗਤ ਵਾਲਾ ਚਿਪ ਨਿਰਮਾਣ
- ਸਧਾਰਨ ਅਤੇ ਲਚਕੀਲਾ ਸਰਕਟ ਡਿਜ਼ਾਈਨ
- ਵਧੀਆ ਫਾਲਟ ਟੋਲਰੇਂਸ ਅਤੇ ਆਸਾਨ ਬਦਲੀ (IC ਡਿਜ਼ਾਈਨ ਵਿੱਚ)

NAND Gate ਨਾਲ ਮੁਢਲੇ ਗੇਟਾਂ ਦਾ Implementation:

1. NOT Gate:

- ਇੱਕ ਹੀ ਇਨਪੁੱਟ A ਨੂੰ ਦੋਹਾਂ ਇਨਪੁੱਟਾਂ ਤੇ ਲਗਾਓ → ਆਉਟਪੁੱਟ = A' **(A NAND A = A')**

2. AND Gate:

- ਪਹਿਲਾਂ A ਅਤੇ B ਨੂੰ ਇੱਕ NAND Gate ਵਿੱਚ ਦਿਓ → ਆਉਟਪੁੱਟ = (A·B)'
- ਫਿਰ ਇਹ ਆਉਟਪੁੱਟ ਨੂੰ ਫਿਰ ਇੱਕ NAND Gate ਵਿੱਚ ਆਪਣੇ ਨਾਲ → ਆਉਟਪੁੱਟ = ((A·B)')' = A·B

3. OR Gate (De Morgan's Law ਤੋਂ):

- A + B = (A'·B')'
- ਪਹਿਲਾਂ A ਅਤੇ B ਨੂੰ ਇੱਕ-ਇੱਕ NAND Gate ਦੇ ਨਾਲ NOT ਕਰੋ
- ਫਿਰ ਉਨ੍ਹਾਂ ਦੇ ਆਉਟਪੁੱਟ ਨੂੰ ਤੀਜੇ NAND Gate ਵਿੱਚ ਦਿਓ → ਆਉਟਪੁੱਟ = OR

NOTE:
ਇਸੇ ਤਰ੍ਹਾਂ ਦੇ ਲਾਜਿਕ ਨਾਲ, **NOR Gate** ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਵੀ AND, OR, NOT ਗੇਟ ਬਣਾਏ ਜਾ ਸਕਦੇ ਹਨ।

Q. Write a detailed note on Simplification of Boolean Expression using K-Maps. (Nov 24)
Ans. **Simplification of Boolean Expressions Using Karnaugh Maps (K-Maps)**
Karnaugh Maps (K-Maps) are a graphical method used to simplify Boolean expressions without using Boolean algebra theorems extensively. They help in minimizing logic functions by reducing the number of logic gates needed in digital circuits, making them more efficient and cost-effective.
A K-Map is a grid-like structure where each cell represents a minterm of the function. The number of cells in a K-Map depends on the number of variables: for 2 variables, it has 4 cells; for 3 variables, 8 cells; and for 4 variables, 16 cells. The values from the truth table or minterm list are plotted into the corresponding cells of the K-Map.
The simplification process involves grouping adjacent 1's in powers of two (1, 2, 4, 8, etc.). These groups may be horizontal or vertical and should be as large as possible to reduce the expression to its minimal form. Each group results in a simplified product term, and the final simplified expression is the logical OR of all the product terms derived from the groups.
K-Maps eliminate the risk of errors in manual simplification and are widely used for designing combinational logic circuits such as adders, multiplexers, and encoders.

Karnaugh Maps (K-Maps) ਦੀ ਵਰਤੋਂ ਨਾਲ ਬੂਲੀਅਨ ਅਭਿਵਕਤੀਆਂ ਦੀ ਸਧਾਰਤਾ: Karnaugh Maps (K-Maps) ਇੱਕ ਗ੍ਰਾਫਿਕਲ ਤਰੀਕਾ ਹੈ ਜੋ Boolean ਅਭਿਵਕਤੀਆਂ ਨੂੰ ਸਧਾਰਨ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ, ਜਿਸ ਵਿੱਚ Boolean Algebra ਦੇ ਸੁਤਰਾਂ ਦੀ ਵਧੇਰੇ ਲੋੜ ਨਹੀਂ ਪੈਂਦੀ। ਇਹ ਲਾਜਿਕ ਫੰਕਸ਼ਨ ਨੂੰ ਘਟਾ ਕੇ ਡਿਜ਼ੀਟਲ ਸਰਕਟ ਵਿੱਚ ਲਾਜਿਕ ਗੇਟਾਂ ਦੀ ਗਿਣਤੀ ਘਟਾਉਂਦੇ ਹਨ, ਜਿਸ ਨਾਲ ਸਰਕਟ ਜ਼ਿਆਦਾ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਅਤੇ ਘੱਟ ਖਰਚੀਲਾ ਬਣਦਾ ਹੈ।

K-Map ਇੱਕ ਗਰਿੱਡ-ਆਕਾਰ ਦਾ ਸਰੂਪ ਹੁੰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਹਰ ਸੈੱਲ ਕਿਸੇ ਮਿੰਟਰਮ (minterm) ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ। K-Map ਵਿੱਚ ਸੈੱਲਾਂ ਦੀ ਗਿਣਤੀ ਵੇਰੀਏਬਲਾਂ ਦੀ ਗਿਣਤੀ 'ਤੇ ਨਿਰਭਰ ਕਰਦੀ ਹੈ:

- 2 ਵੇਰੀਏਬਲ ਲਈ – 4 ਸੈੱਲ
- 3 ਵੇਰੀਏਬਲ ਲਈ – 8 ਸੈੱਲ
- 4 ਵੇਰੀਏਬਲ ਲਈ – 16 ਸੈੱਲ Truth Table ਜਾਂ ਮਿੰਟਰਮ ਲਿਸਟ ਤੋਂ ਲਏ ਗਏ ਮੁੱਲ K-Map ਦੇ ਅਨੁਕੂਲ ਸੈੱਲਾਂ ਵਿੱਚ ਭਰੇ ਜਾਂਦੇ ਹਨ।

ਸਧਾਰਤਾ ਦੀ ਪ੍ਰਕਿਰਿਆ ਵਿੱਚ ਕਾਢੇ ਹੋਏ 1's ਨੂੰ 2 ਦੀ ਘਾਤਾਂ (1, 2, 4, 8 ਆਦਿ) ਦੇ ਸਮੂਹਾਂ ਵਿੱਚ ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਸਮੂਹ ਹੌਰਿਜ਼ਾਂਟਲ ਜਾਂ ਵਰਟੀਕਲ ਹੋ ਸਕਦੇ ਹਨ ਅਤੇ ਜਿੰਨੇ ਵੱਡੇ ਹੋਣ, ਉਨ੍ਹਾਂ ਤੋਂ ਮਿਲਣ ਵਾਲੀ ਅਭਿਵਕਤੀ ਉੱਨੀ ਹੀ ਛੋਟੀ ਹੋਵੇਗੀ।

ਹਰ ਸਮੂਹ ਇੱਕ ਸਧਾਰਿਤ product term ਦਿੰਦਾ ਹੈ ਅਤੇ ਆਖਰੀ Boolean ਅਭਿਵਕਤੀ ਉਹਨਾਂ ਸਾਰੇ product terms ਦਾ OR ਹੁੰਦੀ ਹੈ।

K-Maps ਹੱਥੀਂ ਕੀਤੀ ਸਧਾਰਤਾ ਵਿੱਚ ਗਲਤੀ ਦੇ ਜੋਖਮ ਨੂੰ ਘਟਾਉਂਦੇ ਹਨ ਅਤੇ ਇਹ adders, multiplexers, ਅਤੇ encoders ਵਰਗੇ combinational logic circuits ਦੇ ਡਿਜ਼ਾਈਨ ਵਿੱਚ ਵਿਅਪਕ ਤੌਰ 'ਤੇ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

Q. Find the minimum SOP expression for F(a, b, c, d) = Σm (1,3,5,8,9,10,11,12,13,15) using K map. (Nov 23)

Ans. To find the **minimum SOP (Sum of Products)** expression for the Boolean function **F(a, b, c, d) = Σm(1, 3, 5, 8, 9, 10, 11, 12, 13, 15)** using a **4-variable K-map**, follow these steps:

Step 1: Setup the K-map

The K-map for 4 variables (a, b, c, d) has 16 cells representing minterms from 0 to 15.

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

Step 2: Fill 1s for given minterms

Fill **1** in the cells for minterms: 1, 3, 5, 8, 9, 10, 11, 12, 13, 15.

Step 3: Grouping

Make largest possible adjacent groups of 1s in powers of 2:

- Group1 (4-cell): m(8,9,10,11) → a'b
- Group2 (4-cell): m(12,13,15,11) → ab
- Group3 (2-cell): m(1,3) → a'b'd
- Group4 (2-cell): m(5,13) → bd'c

Step 4: Write simplified SOP expression

The minimized SOP expression is:

F(a, b, c, d) = a'b + ab + a'b'd + bd'c

This expression uses fewer terms and variables, making it optimal for implementation in digital logic circuits.

F(a, b, c, d) = Σm(1, 3, 5, 8, 9, 10, 11, 12, 13, 15) ਦੀ ਘੱਟੋ-ਘੱਟ SOP ਅਭਿਵਕਤੀ ਕੱਢਣ ਲਈ K-Map ਦੀ ਵਰਤੋਂ:

ਕਦਮ 1: K-Map ਬਣਾਓ

4 ਵੇਰੀਏਬਲ (a, b, c, d) ਵਾਲੀ K-Map ਵਿੱਚ ਕੁੱਲ **16 ਸੈੱਲ** ਹੁੰਦੇ ਹਨ (ਮਿੰਟਰਮ 0 ਤੋਂ 15 ਤੱਕ)। K-Map ਦਾ ਫਾਰਮੈਟ:

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

ਕਦਮ 2: ਦਿੱਤੇ ਮਿੰਟਰਮਾਂ ਵਿੱਚ 1 ਭਰੋ

ਮਿੰਟਰਮ: **1, 3, 5, 8, 9, 10, 11, 12, 13, 15** — ਇਨ੍ਹਾਂ ਸੈੱਲਾਂ ਵਿੱਚ '1' ਭਰੋ।

ਕਦਮ 3: 1's ਦੇ ਵੱਡੇ ਸਮੂਹ ਬਣਾਓ (ਪਾਵਰ ਆਫ਼ 2 ਵਿਚ: 2, 4, 8...)

Group 1 (4 ਸੈੱਲ): $m(8, 9, 10, 11) \rightarrow$ **a'b**

Group 2 (4 ਸੈੱਲ): $m(12, 13, 15, 11) \rightarrow$ **ab**

Group 3 (2 ਸੈੱਲ): $m(1, 3) \rightarrow$ **a'b'd**

Group 4 (2 ਸੈੱਲ): $m(5, 13) \rightarrow$ **bd'c**

ਕਦਮ 4: ਘੱਟੋ-ਘੱਟ SOP ਅਭਿਵਕਤੀ ਲਿਖੋ

F(a, b, c, d) = a'b + ab + a'b'd + bd'c

ਨੋਟ:

- ਇਹ ਅਭਿਵਕਤੀ ਘੱਟ ਟਰਮਾਂ ਅਤੇ ਵੇਰੀਏਬਲਾਂ ਨਾਲ ਹੈ, ਜਿਸ ਕਰਕੇ ਇਹ ਡਿਜ਼ੀਟਲ ਲਾਜਿਕ ਸਰਕਟ ਲਈ ਸਭ ਤੋਂ **ਚੁਸਤ (Optimal)** ਹੈ।
- ਇਹ ਸਧਾਰਤਾ K-Map ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਹੱਥੀਂ ਬਿਨਾ ਗਲਤੀ ਦੇ ਕੀਤੀ ਜਾ ਸਕਦੀ ਹੈ।

Unit-II: Combinational Logic Circuits
Short Answer Questions:

Q. What is a Decoder? (Nov 22),(Nov 24)
Ans. **Decoder** is a combinational circuit that converts binary input into a unique active output line. It is used in memory address decoding and display systems.
ਡੀਕੋਡਰ ਇੱਕ ਸੰਯੁਕਤ ਲਾਜਿਕ ਸਰਕਟ ਹੈ ਜੋ ਬਾਈਨਰੀ ਇਨਪੁੱਟ ਨੂੰ ਇੱਕ ਵਿਲੱਖਣ ਐਕਟਿਵ ਆਉਟਪੁੱਟ ਲਾਈਨ ਵਿੱਚ ਬਦਲਦਾ ਹੈ। ਇਹ ਮੈਮੋਰੀ ਐਡਰੈੱਸ ਡੀਕੋਡਿੰਗ ਅਤੇ ਡਿਸਪਲੇ ਸਿਸਟਮ ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।

Q. How do multiplexer differ from demultiplexer? (Nov 23)
Ans. A **Multiplexer** selects one input from many and sends it to a single output line, while a **Demultiplexer** takes one input and distributes it to multiple outputs.
ਮਲਟੀਪਲੈਕਸਰ ਬਹੁਤ ਸਾਰੇ ਇਨਪੁੱਟ ਵਿੱਚੋਂ ਇੱਕ ਚੁਣਦਾ ਹੈ ਅਤੇ ਇੱਕ ਆਉਟਪੁੱਟ ਰਾਹੀਂ ਭੇਜਦਾ ਹੈ। ਡਿਮਲਟੀਪਲੈਕਸਰ ਇੱਕ ਇਨਪੁੱਟ ਲੈਂਦਾ ਹੈ ਅਤੇ ਇਸਨੂੰ ਕਈ ਆਉਟਪੁੱਟਾਂ ਵਿੱਚ ਵੰਡਦਾ ਹੈ।

Q. What is the truth table of Half Subtracter? (Nov 24)
Ans. The **truth table of a Half Subtracter** shows two inputs (A and B) and outputs difference (D = A⊕B) and borrow (B = A'B).
ਇਸ ਵਿੱਚ ਦੋ ਇਨਪੁੱਟ (A ਅਤੇ B) ਹੁੰਦੇ ਹਨ ਅਤੇ ਆਉਟਪੁੱਟ ਵਿੱਚ Difference (D = A⊕B) ਅਤੇ Borrow (B = A'B) ਮਿਲਦੇ ਹਨ।

Q. Differentiate between encoder and decoder. (Nov 24)
Ans. An **Encoder** converts active input signals into coded binary output, whereas a **Decoder** performs the reverse by decoding binary input into distinct outputs
Encoder ਐਕਟਿਵ ਇਨਪੁੱਟ ਨੂੰ ਕੋਡਿਤ ਬਾਈਨਰੀ ਆਉਟਪੁੱਟ ਵਿੱਚ ਬਦਲਦਾ ਹੈ। Decoder ਇਨ੍ਹਾਂ ਕੋਡਿਤ ਇਨਪੁੱਟਾਂ ਨੂੰ ਵੱਖ-ਵੱਖ ਆਉਟਪੁੱਟ ਲਾਈਨਾਂ ਵਿੱਚ ਪਰਿਵਰਤਿਤ ਕਰਦਾ ਹੈ।

Q. Name two uses of multiplexers. (Nov 24)
Ans. **Multiplexers** are used in data routing and communication systems to manage multiple data lines and reduce hardware complexity.
ਮਲਟੀਪਲੈਕਸਰ ਡਾਟਾ ਰਾਊਟਿੰਗ ਅਤੇ ਸੰਚਾਰ ਪ੍ਰਣਾਲੀਆਂ ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਕਈ ਡਾਟਾ ਲਾਈਨਾਂ ਨੂੰ ਸੰਭਾਲ ਕੇ ਹਾਰਡਵੇਅਰ ਦੀ ਜਟਿਲਤਾ ਘਟਾਉਂਦਾ ਹੈ।

Long Answer Questions:

Q. What is a Multiplexer? Explain with a truth table and logic circuit the design of an 8 to 1 line multiplexer. (Nov 23), (Nov 22)
Ans. A **Multiplexer (MUX)** is a combinational logic circuit that selects one of several input signals and forwards the selected input to a single output line. It acts like a digital switch, allowing the transmission of data from multiple sources to one destination.
An **8-to-1 multiplexer** has **8 input lines (I_0 to I_7)**, **3 select lines (S_2, S_1, S_0)**, and **1 output line (Y)**. The select lines determine which one of the 8 inputs is connected to the output.
Truth Table:

S_2	S_1	S_0	Output (Y)
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4

S_2 S_1 S_0 Output (Y)

1 0 1 I_5

1 1 0 I_6

1 1 1 I_7

Logic Circuit:

The output Y is expressed using the select lines as:

$$\mathbf{Y = \bar{S}_2\bar{S}_1\bar{S}_0 \cdot I_0 + \bar{S}_2\bar{S}_1S_0 \cdot I_1 + \bar{S}_2S_1\bar{S}_0 \cdot I_2 + \bar{S}_2S_1S_0 \cdot I_3 + S_2\bar{S}_1\bar{S}_0 \cdot I_4 + S_2\bar{S}_1S_0 \cdot I_5 + S_2S_1\bar{S}_0 \cdot I_6 + S_2S_1S_0 \cdot I_7}$$

This implementation uses AND, OR, and NOT gates. Each input is ANDed with the corresponding combination of select lines, and the results are ORed together to form the output. Multiplexers are widely used in communication systems, data routing, and ALUs.

Multiplexer (MUX):

ਮਲਟੀਪਲੇਕਸਰ ਇੱਕ ਕੰਬੀਨੇਸ਼ਨਲ ਲਾਜਿਕ ਸਰਕਟ ਹੁੰਦਾ ਹੈ ਜੋ ਕਈ ਇਨਪੁੱਟ ਸਿਗਨਲਾਂ ਵਿਚੋਂ ਇੱਕ ਚੁਣਦਾ ਹੈ ਅਤੇ ਉਸ ਨੂੰ ਇੱਕ ਸਿੰਗਲ ਆਉਟਪੁੱਟ ਲਾਈਨ 'ਤੇ ਭੇਜਦਾ ਹੈ। ਇਹ ਇੱਕ ਡਿਜ਼ੀਟਲ ਸਵਿੱਚ ਵਾਂਗ ਕੰਮ ਕਰਦਾ ਹੈ ਜੋ ਕਈ ਸਰੋਤਾਂ ਤੋਂ ਇੱਕ ਮੰਜ਼ਿਲ ਵੱਲ ਡਾਟਾ ਭੇਜਣ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ।

8-to-1 Multiplexer:

- **8 ਇਨਪੁੱਟ ਲਾਈਨਾਂ:** I_0 ਤੋਂ I_7
- **3 ਸਿਲੈਕਟ ਲਾਈਨਾਂ:** S_2, S_1, S_0
- **1 ਆਉਟਪੁੱਟ ਲਾਈਨ:** Y

ਸਿਲੈਕਟ ਲਾਈਨਾਂ ਇਹ ਤੈਅ ਕਰਦੀਆਂ ਹਨ ਕਿ 8 ਇਨਪੁੱਟਾਂ ਵਿੱਚੋਂ ਕਿਹੜੀ ਇੱਕ Y ਨਾਲ ਜੁੜੇਗੀ।

Truth Table:

S_2 S_1 S_0 Y = Output

0 0 0 I_0

0 0 1 I_1

0 1 0 I_2

0 1 1 I_3

1 0 0 I_4

1 0 1 I_5

1 1 0 I_6

1 1 1 I_7

ਲਾਜਿਕ ਸਰਕਟ (Logic Expression):

$$Y = \bar{S}_2\bar{S}_1\bar{S}_0 \cdot I_0 + \bar{S}_2\bar{S}_1S_0 \cdot I_1 + \bar{S}_2S_1\bar{S}_0 \cdot I_2 + \bar{S}_2S_1S_0 \cdot I_3 + S_2\bar{S}_1\bar{S}_0 \cdot I_4 + S_2\bar{S}_1S_0 \cdot I_5 + S_2S_1\bar{S}_0 \cdot I_6 + S_2S_1S_0 \cdot I_7$$

- ਹਰ ਇਨਪੁੱਟ ਨੂੰ ਇੱਕ AND ਗੇਟ ਰਾਹੀਂ ਉਨ੍ਹਾਂ ਦੇ ਸਿਲੈਕਟ ਲਾਈਨਾਂ ਨਾਲ ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ।
- ਸਭ AND ਆਉਟਪੁੱਟ ਨੂੰ OR ਗੇਟ ਰਾਹੀਂ ਮਿਲਾ ਕੇ ਆਉਟਪੁੱਟ Y ਬਣਾਈ ਜਾਂਦੀ ਹੈ।

ਮਲਟੀਪਲੇਕਸਰ ਦੀ ਵਰਤੋਂ: ਮਲਟੀਪਲੇਕਸਰ **ਕਮਿਊਨਿਕੇਸ਼ਨ ਸਿਸਟਮ**, **ਡਾਟਾ ਰਾਊਟਿੰਗ**, ਅਤੇ **ਅਰੀਥਮੈਟਿਕ ਲਾਜਿਕ ਯੂਨਿਟ (ALU)** ਵਿੱਚ ਵਿਆਪਕ ਤੌਰ 'ਤੇ ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

Q. Explain Binary Adder/Subtractor with the help of an example. (Nov 22)

Ans. A **Binary Adder/Subtractor** is a combinational circuit that performs both binary addition and subtraction using logic gates. It uses **full adders** and an **XOR gate** to control whether the operation is addition or subtraction.

Working Principle:

- **Addition**: When the control signal **M = 0**, the circuit performs binary addition.

- **Subtraction**: When **M = 1**, it performs binary subtraction using 2's complement (invert the second number and add 1).

Circuit Components:

- XOR gates to invert B inputs during subtraction.
- Full Adders to perform bit-wise addition.
- A control signal (M) to choose between addition and subtraction.

Example:

Let's take two 4-bit binary numbers:

A = 0101 (5)
B = 0011 (3)

- **Addition (M = 0):**
  - B remains 0011
  - A + B = 0101 + 0011 = **1000 (8)**
- **Subtraction (M = 1):**
  - B becomes 1100 (1's complement of 0011)
  - Add 1 to get 2's complement: 1101
  - A + 2's complement of B: 0101 + 1101 = **0010 (2)**

Conclusion: A binary adder/subtractor efficiently handles both operations using the same hardware. This dual functionality is essential in arithmetic logic units (ALUs) of processors.

ਬਾਈਨਰੀ ਐਡਰ/ਸਬਟ੍ਰੈਕਟਰ (Binary Adder/Subtractor):

ਬਾਈਨਰੀ ਐਡਰ/ਸਬਟ੍ਰੈਕਟਰ ਇੱਕ **ਕੰਬੀਨੇਸ਼ਨਲ ਲਾਜਿਕ ਸਰਕਟ** ਹੈ ਜੋ **ਬਾਈਨਰੀ ਜੋੜ ਅਤੇ ਘਟਾਊ** ਦੋਵੇਂ ਕਾਰਜ ਕਰ ਸਕਦਾ ਹੈ। ਇਹ **ਫੁੱਲ ਐਡਰ (Full Adders)** ਅਤੇ **XOR ਗੇਟ** ਦੀ ਵਰਤੋਂ ਕਰਦਾ ਹੈ ਜੋ ਇਹ ਨਿਰਧਾਰਤ ਕਰਦੇ ਹਨ ਕਿ ਸਰਕਟ ਐਡ ਕਰੇਗਾ ਜਾਂ ਸਬਟ੍ਰੈਕਟ।

ਕਾਰਜ ਕਰਨ ਦਾ ਸਿਧਾਂਤ (Working Principle):

- **ਜੋੜ (Addition):** ਜਦੋਂ ਕੰਟਰੋਲ ਸਿਗਨਲ **M = 0** ਹੋਵੇ, ਤਾਂ ਸਰਕਟ ਸਿੱਧਾ **ਬਾਈਨਰੀ ਜੋੜ** ਕਰਦਾ ਹੈ।
- **ਘਟਾਊ (Subtraction):** ਜਦੋਂ **M = 1** ਹੋਵੇ, ਤਾਂ ਇਹ **2's ਕਮਪਲੀਮੈਂਟ** ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਘਟਾਊ ਕਰਦਾ ਹੈ (B ਨੂੰ ਇਨਵਰਟ ਕਰਕੇ 1 ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ)।

ਸਰਕਟ ਦੇ ਮੁੱਖ ਹਿੱਸੇ (Circuit Components):

- **XOR ਗੇਟਸ:** Subtraction ਦੌਰਾਨ B ਦੇ ਇਨਪੁੱਟਸ ਨੂੰ ਇਨਵਰਟ ਕਰਨ ਲਈ।
- **Full Adders:** ਹਰ ਬਿੱਟ ਦਾ ਜੋੜ ਕਰਨ ਲਈ।
- **Control Signal (M):** ਇਹ ਨਿਰਧਾਰਤ ਕਰਦਾ ਹੈ ਕਿ ਐਡ ਕਰਨਾ ਹੈ ਜਾਂ ਸਬਟ੍ਰੈਕਟ।

ਉਦਾਹਰਨ (Example):

ਚਲੋ ਦੋ 4-ਬਿੱਟ ਬਾਈਨਰੀ ਨੰਬਰ ਲਈਏ:

A = 0101 (5)
B = 0011 (3)

Addition (M = 0):

- B ਅਜਿਹਾ ਹੀ ਰਹੇਗਾ → 0011
- A + B = 0101 + 0011 = **1000 (8)**

Subtraction (M = 1):

- B ਦਾ 1's ਕਮਪਲੀਮੈਂਟ → 1100
- 2's ਕਮਪਲੀਮੈਂਟ ਲਈ 1 ਜੋੜੋ → 1101

- A + 2's ਕਮਪਲੀਮੈਂਟ of B = 0101 + 1101 = **0010 (2)**

ਨਤੀਜਾ (Conclusion): Binary Adder/Subtractor ਇੱਕ ਹੀ ਹਾਰਡਵੇਅਰ ਨਾਲ ਦੋਵੇਂ ਜੋੜ ਅਤੇ ਘਟਾਊ ਕਰ ਸਕਦਾ ਹੈ। ਇਹ **ਡੁਅਲ ਫੰਕਸ਼ਨਾਲਟੀ** ਪ੍ਰੋਸੈਸਰ ਦੇ **Arithmetic Logic Units (ALUs)** ਵਿੱਚ ਬਹੁਤ ਮਹੱਤਵਪੂਰਨ ਹੁੰਦੀ ਹੈ।

Q. Explain the following terms in detail: (Nov 24)
a) Half Adder
b) Full Subtractor
Ans. **a) Half Adder:** A **Half Adder** is a basic combinational logic circuit used to perform the addition of two single-bit binary numbers. It has **two inputs** (A and B) and **two outputs**: **Sum (S)** and **Carry (C)**.

- The **Sum** is given by: **S = A ⊕ B** (XOR gate)
- The **Carry** is given by: **C = A · B** (AND gate)
  The Half Adder cannot account for a carry-in from a previous stage, so it is mainly used in the first stage of binary addition. It is simple and forms the basis for building Full Adders.

a) ਹਾਫ ਐਡਰ (Half Adder): ਹਾਫ ਐਡਰ ਇੱਕ ਮੂਲ **ਕੰਬੀਨੇਸ਼ਨਲ ਲਾਜਿਕ ਸਰਕਟ** ਹੈ ਜੋ ਦੋ ਇੱਕ-ਬਿੱਟ ਬਾਈਨਰੀ ਨੰਬਰਾਂ ਦਾ ਜੋੜ ਕਰਨ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਦੇ **ਦੋ ਇਨਪੁੱਟ** ਹੁੰਦੇ ਹਨ: **A ਅਤੇ B**, ਅਤੇ **ਦੋ ਆਉਟਪੁੱਟ**: **Sum (S)** ਅਤੇ **Carry (C)**।

- **Sum:** S = A ⊕ B *(XOR ਗੇਟ ਦੁਆਰਾ)*
- **Carry:** C = A · B *(AND ਗੇਟ ਦੁਆਰਾ)*

ਹਾਫ ਐਡਰ **ਕਿਸੇ ਪਿਛਲੇ ਕੈਰੀ (Carry-in)** ਨੂੰ ਧਿਆਨ ਵਿੱਚ ਨਹੀਂ ਰੱਖਦਾ, ਇਸ ਲਈ ਇਹ ਆਮ ਤੌਰ 'ਤੇ **ਬਾਈਨਰੀ ਜੋੜ ਦੀ ਪਹਿਲੀ ਸਟੇਜ** ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਸਰਲ ਹੁੰਦਾ ਹੈ ਅਤੇ **ਫੁੱਲ ਐਡਰ** ਬਣਾਉਣ ਦਾ ਆਧਾਰ ਹੈ।

b) Full Subtractor: A **Full Subtractor** is a combinational circuit that performs binary subtraction of three bits: **minuend (A), subtrahend (B),** and **borrow-in (Bin)**. It has **two outputs**: **Difference (D)** and **Borrow-out (Bout)**.

- The **Difference** is given by: **D = A ⊕ B ⊕ Bin**
- The **Borrow-out** is: **Bout = B·Bin + B·Ā + Bin·Ā**
  The Full Subtractor handles subtraction with a previous borrow, unlike a Half Subtractor. It is essential in multi-bit binary subtraction operations in ALUs and digital processors.

b) ਫੁੱਲ ਸਬਟ੍ਰੈਕਟਰ (Full Subtractor):

ਫੁੱਲ ਸਬਟ੍ਰੈਕਟਰ ਇੱਕ ਕੰਬੀਨੇਸ਼ਨਲ ਲਾਜਿਕ ਸਰਕਟ ਹੈ ਜੋ **ਤਿੰਨ ਬਿੱਟਾਂ** ਦਾ ਬਾਈਨਰੀ ਘਟਾਊ ਕਰਦਾ ਹੈ:

- **Minuend (A)**
- **Subtrahend (B)**
- **Borrow-in (Bin)**

ਇਸ ਦੇ **ਦੋ ਆਉਟਪੁੱਟ** ਹੁੰਦੇ ਹਨ: **Difference (D)** ਅਤੇ **Borrow-out (Bout)**

- **Difference:** D = A ⊕ B ⊕ Bin
- **Borrow-out:** Bout = B·Bin + B·Ā + Bin·Ā

ਫੁੱਲ ਸਬਟ੍ਰੈਕਟਰ **ਪਿਛਲੇ ਬੋਰੋ (Borrow-in)** ਨੂੰ ਵੀ ਸੰਭਾਲਦਾ ਹੈ, ਜੋ ਕਿ ਹਾਫ ਸਬਟ੍ਰੈਕਟਰ ਨਹੀਂ ਕਰ ਸਕਦਾ। ਇਹ **multi-bit ਬਾਈਨਰੀ ਘਟਾਊ** ਅਤੇ **ALU/Processor** ਸਰਕਟਾਂ ਵਿੱਚ ਬਹੁਤ ਜ਼ਰੂਰੀ ਹੁੰਦਾ ਹੈ।

Q. Write a detailed note on following: (Nov 24)

a) Demultiplexer
b) Decoders

Ans. **a) Demultiplexer:** A **Demultiplexer (DEMUX)** is a combinational logic circuit that takes a **single input** and routes it to **one of several output lines**, depending on the values of **select lines**. It functions as a reverse of a multiplexer. An *n-to-2^n* demultiplexer has 1 data input, *n* select lines, and 2^n output lines. For example, a 1-to-4 DEMUX uses 2 select lines to direct input to one of four outputs. It is widely used in data distribution, memory addressing, and digital signal routing, where a single data source needs to be sent to multiple destinations based on control signals.

a) ਡੀਮਲਟੀਪਲੇਕਸਰ (Demultiplexer - DEMUX): ਡੀਮਲਟੀਪਲੇਕਸਰ ਇੱਕ **ਕੰਬੀਨੇਸ਼ਨਲ ਲਾਜਿਕ ਸਰਕਟ** ਹੁੰਦਾ ਹੈ ਜੋ ਇੱਕ ਸਿੰਗਲ ਇਨਪੁੱਟ ਨੂੰ ਲੈਂਦਾ ਹੈ ਅਤੇ **ਸਿਲੈਕਟ ਲਾਈਨਾਂ** ਦੇ ਮੁੱਲ ਦੇ ਆਧਾਰ 'ਤੇ ਇਸਨੂੰ ਕਈ ਆਉਟਪੁੱਟ ਲਾਈਨਾਂ ਵਿੱਚੋਂ ਕਿਸੇ ਇੱਕ ਵੱਲ ਭੇਜਦਾ ਹੈ। ਇਹ **ਮਲਟੀਪਲੇਕਸਰ ਦਾ ਉਲਟਾ ਕੰਮ** ਕਰਦਾ ਹੈ।

- ਇੱਕ **n-to-2^n DEMUX** ਵਿੱਚ 1 ਇਨਪੁੱਟ, **n ਸਿਲੈਕਟ ਲਾਈਨਾਂ**, ਅਤੇ **2^n ਆਉਟਪੁੱਟ ਲਾਈਨਾਂ** ਹੁੰਦੀਆਂ ਹਨ।
- ਉਦਾਹਰਨ ਵਜੋਂ, ਇੱਕ **1-to-4 DEMUX** 2 ਸਿਲੈਕਟ ਲਾਈਨਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਇਨਪੁੱਟ ਨੂੰ 4 ਆਉਟਪੁੱਟਾਂ ਵਿੱਚੋਂ ਕਿਸੇ ਇੱਕ ਵੱਲ ਭੇਜਦਾ ਹੈ।

ਵਰਤੋਂ: DEMUX ਦਾ ਵਰਤਾਉ **ਡਾਟਾ ਵੰਡਣ**, **ਮੈਮੋਰੀ ਐਡਰੈਸਿੰਗ**, ਅਤੇ **ਡਿਜੀਟਲ ਸਿਗਨਲ ਰਾਊਟਿੰਗ** ਵਿੱਚ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਜਿੱਥੇ ਇੱਕ ਸਿੰਗਲ ਡਾਟਾ ਸਰੋਤ ਨੂੰ ਵੱਖ-ਵੱਖ ਮੰਜ਼ਿਲਾਂ 'ਤੇ ਭੇਜਣਾ ਹੋਵੇ।

b) Decoders: A **Decoder** is a digital logic circuit that converts **binary input codes** into **a unique output line**. It has *n* input lines and 2^n output lines, with only one output active at a time, corresponding to the binary value of the input. For example, a **3-to-8 decoder** activates one of 8 outputs based on the 3-bit binary input. Decoders are used in memory address decoding, instruction decoding in CPUs, and enabling devices in microprocessors. They are essential for translating encoded data into a human- or machine-readable form, ensuring proper routing and control within digital systems.

b) ਡੀਕੋਡਰ (Decoder): ਡੀਕੋਡਰ ਇੱਕ **ਡਿਜੀਟਲ ਲਾਜਿਕ ਸਰਕਟ** ਹੈ ਜੋ **ਬਾਈਨਰੀ ਇਨਪੁੱਟ ਕੋਡ** ਨੂੰ ਇੱਕ **ਵਿਲੱਖਣ (unique) ਆਉਟਪੁੱਟ ਲਾਈਨ** ਵਿੱਚ ਤਬਦੀਲ ਕਰਦਾ ਹੈ।

- ਇਸ ਵਿੱਚ **n ਇਨਪੁੱਟ ਲਾਈਨਾਂ** ਅਤੇ **2^n ਆਉਟਪੁੱਟ ਲਾਈਨਾਂ** ਹੁੰਦੀਆਂ ਹਨ।
- ਹਰ ਸਮੇਂ ਸਿਰਫ **ਇੱਕ ਆਉਟਪੁੱਟ ਐਕਟਿਵ** ਹੁੰਦੀ ਹੈ ਜੋ ਇਨਪੁੱਟ ਦੇ ਬਾਈਨਰੀ ਮੁੱਲ ਦੇ ਅਨੁਸਾਰ ਚੁਣੀ ਜਾਂਦੀ ਹੈ।

ਉਦਾਹਰਨ ਵਜੋਂ: ਇੱਕ **3-to-8 ਡੀਕੋਡਰ** 3-ਬਿੱਟ ਇਨਪੁੱਟ ਦੇ ਆਧਾਰ 'ਤੇ 8 ਵਿੱਚੋਂ ਇੱਕ ਆਉਟਪੁੱਟ ਲਾਈਨ ਨੂੰ ਐਕਟਿਵ ਕਰਦਾ ਹੈ।

ਵਰਤੋਂ: ਡੀਕੋਡਰ ਨੂੰ **ਮੈਮੋਰੀ ਐਡਰੈਸ ਡੀਕੋਡਿੰਗ**, **CPU ਵਿੱਚ ਇੰਸਟਰੱਕਸ਼ਨ ਡੀਕੋਡਿੰਗ**, ਅਤੇ **ਮਾਈਕਰੋਪ੍ਰੋਸੈਸਰ ਵਿੱਚ ਡਿਵਾਈਸ ਐਨੇਬਲ ਕਰਨ** ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ **ਕੋਡਿਤ ਡਾਟਾ ਨੂੰ ਪੜ੍ਹਨਯੋਗ ਰੂਪ ਵਿੱਚ ਬਦਲਣ** ਅਤੇ **ਸਹੀ ਨਿਯੰਤਰਣ ਅਤੇ ਰਾਊਟਿੰਗ** ਲਈ ਬਹੁਤ ਮਹੱਤਵਪੂਰਨ ਹਨ।

Q. Design a full-adder with the help of logic gates. (Nov 23)

Ans. A **Full Adder** is a digital combinational circuit that adds **three input bits**: **A, B, and Carry-in (Cin)**, and produces two outputs: **Sum (S)** and **Carry-out (Cout)**. It is used in binary addition to handle carry from previous stages, unlike a Half Adder.

Logic Expression:

- **Sum (S) = A ⊕ B ⊕ Cin**
- **Carry-out (Cout) = (A · B) + (B · Cin) + (A · Cin)**

Logic Gate Implementation:

1. First XOR gate computes: **X1 = A ⊕ B**
2. Second XOR gate computes the final sum: **Sum = X1 ⊕ Cin**

3. AND gates compute intermediate carries:
   C1 = A · B
   C2 = B · Cin
   C3 = A · Cin
4. OR gate combines the intermediate carries: **Cout = C1 + C2 + C3**

Circuit Description:

- The full adder requires **two XOR gates**, **three AND gates**, and **one OR gate**.
- It forms the basic building block of multi-bit binary adders like Ripple Carry Adders.
- Full Adders are essential in Arithmetic Logic Units (ALUs), digital processors, and calculators.

This gate-level design ensures accurate bit-wise addition with proper carry handling.

ਫੁੱਲ ਐਡਰ (Full Adder): ਫੁੱਲ ਐਡਰ ਇੱਕ **ਡਿਜੀਟਲ ਕੰਬੀਨੇਸ਼ਨਲ ਸਰਕਟ** ਹੈ ਜੋ **ਤਿੰਨ ਇਨਪੁੱਟ ਬਿੱਟਾਂ — A, B ਅਤੇ Carry-in (Cin)** — ਦਾ ਜੋੜ ਕਰਦਾ ਹੈ ਅਤੇ **ਦੋ ਆਉਟਪੁੱਟ** ਦਿੰਦਾ ਹੈ: **Sum (S)** ਅਤੇ **Carry-out (Cout)**। ਇਹ **Half Adder** ਨਾਲੋਂ ਵੱਧ ਤਰੱਕੀਸ਼ੀਲ ਹੈ ਕਿਉਂਕਿ ਇਹ ਪਿਛਲੇ ਕੈਰੀ ਨੂੰ ਵੀ ਸੰਭਾਲ ਸਕਦਾ ਹੈ।

ਲਾਜਿਕ ਐਕਸਪ੍ਰੈਸ਼ਨ (Logic Expression):

- **Sum (S):** A ⊕ B ⊕ Cin
- **Carry-out (Cout):** (A · B) + (B · Cin) + (A · Cin)

ਲਾਜਿਕ ਗੇਟ ਅਨੁਸਾਰ ਇੰਪਲੀਮੈਂਟੇਸ਼ਨ:

- **ਪਹਿਲਾ XOR ਗੇਟ:** **X1 = A ⊕ B**
- **ਦੂਜਾ XOR ਗੇਟ:** **Sum = X1 ⊕ Cin**
- **AND ਗੇਟਸ ਦੁਆਰਾ Intermediate Carries:**
  C1 = A · B
  C2 = B · Cin
  C3 = A · Cin
- **OR ਗੇਟ ਦੁਆਰਾ Carry-out:** **Cout = C1 + C2 + C3**

ਸਰਕਟ ਦਾ ਵਰਣਨ (Circuit Description):

ਫੁੱਲ ਐਡਰ ਬਣਾਉਣ ਲਈ ਲੋੜੀਂਦੇ ਹਨ:

- **2 XOR ਗੇਟ**,
- **3 AND ਗੇਟ**,
- **1 OR ਗੇਟ**

ਇਹ **Ripple Carry Adders** ਵਰਗੇ **ਮਲਟੀ-ਬਿਟ ਐਡਰ** ਬਣਾਉਣ ਲਈ ਬੇਸਿਕ ਬਿਲਡਿੰਗ ਬਲੌਕ ਹੈ। ਫੁੱਲ ਐਡਰ **ALU (Arithmetic Logic Units)**, **ਡਿਜੀਟਲ ਪ੍ਰੋਸੈਸਰ**, ਅਤੇ **ਕੈਲਕੂਲੇਟਰਸ** ਵਿੱਚ ਬਹੁਤ ਜ਼ਰੂਰੀ ਹਿੱਸਾ ਨਿਭਾਉਂਦਾ ਹੈ। ਇਹ **ਗੇਟ-ਲੈਵਲ ਡਿਜ਼ਾਈਨ** ਸਹੀ **ਬਿੱਟ-ਵਾਇਜ਼ ਜੋੜ** ਅਤੇ **ਕੈਰੀ ਹੈਂਡਲਿੰਗ** ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ।

Unit-III: Sequential Logic Circuits
Short Answer Questions:

Q. What is Race around condition in a Flip flop? (Nov 22)
Ans. **Race around condition** in a flip-flop occurs when the output changes continuously due to continuous input while the clock is high, mainly in JK flip-flops.
Race-around condition ਉਹ ਸਮਾਂ ਹੁੰਦਾ ਹੈ ਜਦੋਂ JK ਫਲਿਪ-ਫਲੌਪ ਵਿੱਚ ਕਲਾਕ HIGH ਹੋਣ ਦੌਰਾਨ ਆਉਟਪੁੱਟ ਲਗਾਤਾਰ ਬਦਲਦਾ ਰਹਿੰਦਾ ਹੈ। ਇਹ ਜ਼ਿਆਦਾਤਰ ਤਦੋਂ ਹੁੰਦਾ ਹੈ ਜਦੋਂ ਇਨਪੁੱਟ ਲਗਾਤਾਰ ਐਕਟਿਵ ਰਹੇ।

Q. What are the different applications of Flip flops? (Nov 22)
Ans. **Flip-flops** are used in registers, counters, memory units, and digital clocks for storing binary data and state transition.
Flip-flops ਰਜਿਸਟਰ, ਕਾਊਂਟਰ, ਮੈਮੋਰੀ ਯੂਨਿਟ ਅਤੇ ਡਿਜ਼ੀਟਲ ਘੜੀਆਂ ਵਿੱਚ ਬਾਈਨਰੀ ਡਾਟਾ ਸੰਭਾਲਣ ਲਈ ਵਰਤੇ ਜਾਂਦੇ ਹਨ। ਇਹ ਸਟੇਟ ਟਰਾਂਜ਼ੀਸ਼ਨ ਲਈ ਵੀ ਮਹੱਤਵਪੂਰਨ ਹਨ।

Q. What is a Latch? (Nov 22)
Ans. A **Latch** is a basic memory device that holds the binary state as long as power is supplied, and is level-triggered.
ਲੈਚ ਇੱਕ ਬੁਨਿਆਦੀ ਮੈਮੋਰੀ ਡਿਵਾਈਸ ਹੈ ਜੋ ਬਾਈਨਰੀ ਸਥਿਤੀ ਨੂੰ ਤਕ ਰੱਖਦਾ ਹੈ ਜਦ ਤੱਕ ਪਾਵਰ ਮਿਲਦੀ ਰਹੇ। ਇਹ **ਲੈਵਲ-ਟ੍ਰਿੱਗਰਡ** ਹੁੰਦਾ ਹੈ।

Q. What can be done to avoid racing problem in JK flip flop? (Nov 23)
Ans. To **avoid race-around condition** in JK flip-flop, master-slave configuration or edge-triggered flip-flops are used.
JK ਫਲਿਪ-ਫਲੌਪ ਵਿੱਚ ਰੇਸ-ਅਰਾਊਂਡ ਤੋਂ ਬਚਣ ਲਈ **ਮਾਸਟਰ-ਸਲੇਵ ਕਨਫਿਗਰੇਸ਼ਨ** ਜਾਂ **ਐਜ-ਟ੍ਰਿੱਗਰਡ ਫਲਿਪ-ਫਲੌਪ** ਵਰਤੇ ਜਾਂਦੇ ਹਨ।

Q. What is a sequential circuit? Give an example. (Nov 24)
Ans. A **Sequential circuit** is a digital circuit where output depends on current inputs and previous states; e.g., flip-flops and counters.
Sequential ਸਰਕਟ ਉਹ ਡਿਜ਼ੀਟਲ ਸਰਕਟ ਹੁੰਦਾ ਹੈ ਜਿਸ ਵਿੱਚ ਆਉਟਪੁੱਟ ਮੌਜੂਦਾ ਇਨਪੁੱਟ ਅਤੇ ਪਿਛਲੇ ਸਟੇਟ 'ਤੇ ਨਿਰਭਰ ਕਰਦਾ ਹੈ। ਉਦਾਹਰਣ: ਫਲਿਪ-ਫਲੌਪ ਅਤੇ ਕਾਊਂਟਰ।

Q. How a D Flip flop is converted in to T flip flop? (Nov 24)
Ans. A **D flip-flop** can be converted into a **T flip-flop** by connecting the Q' output to the D input, allowing toggling behavior.
D ਫਲਿਪ-ਫਲੌਪ ਨੂੰ T ਫਲਿਪ-ਫਲੌਪ ਵਿੱਚ ਬਦਲਣ ਲਈ Q' ਆਉਟਪੁੱਟ ਨੂੰ D ਇਨਪੁੱਟ ਨਾਲ ਜੋੜਿਆ ਜਾਂਦਾ ਹੈ। ਇਸ ਨਾਲ ਟੌਗਲਿੰਗ ਵਿਵਹਾਰ ਮਿਲਦਾ ਹੈ।

Long Answer Questions:
Q. Explain the working of a JK flip-flop with a truth table and logic circuit. (Nov 22)
Ans. A **JK Flip-Flop** is a **bistable sequential circuit** that operates with two inputs, **J** and **K**, a **clock signal (CLK)**, and two outputs: **Q** and $\bar{Q}$. It is an improved version of the SR flip-flop, overcoming the undefined state when both inputs are high.
Working Principle:
- The output changes only on the **rising or falling edge of the clock**, depending on the design.
- It uses feedback to control toggling and ensures stability.

Truth Table:

J	K	CLK	Q (Next State)	Description
0	0	↑	Q (No change)	Memory state
0	1	↑	0	Reset
1	0	↑	1	Set
1	1	↑	$\bar{Q}$ (Toggle)	Toggle output

(↑ represents a rising clock edge)

Logic Circuit:

The JK flip-flop can be built using **AND, OR, NOT, and NAND gates** or using two SR flip-flops with additional logic. Inputs J and K are ANDed with the clock and feedback from outputs Q and $\bar{Q}$ to generate the set/reset conditions.

Applications:

JK flip-flops are widely used in **counters**, **shift registers**, and **memory storage**, where controlled toggling or state change is essential.

JK ਫਲਿਪ-ਫਲੌਪ (JK Flip-Flop):

JK ਫਲਿਪ-ਫਲੌਪ ਇੱਕ **ਬਿਸਟੇਬਲ ਸੀਕਵੇਂਸ਼ਲ ਸਰਕਟ** ਹੈ ਜਿਸ ਵਿੱਚ **ਦੋ ਇਨਪੁੱਟ** ਹੁੰਦੇ ਹਨ: **J ਅਤੇ K**, ਇੱਕ **ਕਲੌਕ ਸਿਗਨਲ (CLK)** ਅਤੇ **ਦੋ ਆਉਟਪੁੱਟ**: **Q ਅਤੇ $\bar{Q}$ (Q bar)**। ਇਹ **SR ਫਲਿਪ-ਫਲੌਪ** ਦਾ ਸੁਧਾਰਿਤ ਰੂਪ ਹੈ ਜੋ ਦੋਵੇਂ ਇਨਪੁੱਟ ਹਾਈ ਹੋਣ ਦੀ ਅਣਨਿਰਧਾਰਿਤ ਅਵਸਥਾ ਨੂੰ ਦੂਰ ਕਰਦਾ ਹੈ।

ਕਾਰਜ ਸਿਧਾਂਤ (Working Principle):

- ਆਉਟਪੁੱਟ ਸਿਰਫ **ਕਲੌਕ ਦੇ rising ਜਾਂ falling edge** 'ਤੇ ਹੀ ਬਦਲਦਾ ਹੈ (ਡਿਜ਼ਾਈਨ ਤੇ ਨਿਰਭਰ ਕਰਦਾ ਹੈ)।
- ਇਹ **ਫੀਡਬੈਕ** ਵਰਤਦਾ ਹੈ ਜੋ toggling ਨੂੰ ਨਿਯੰਤਰਿਤ ਕਰਦਾ ਹੈ ਅਤੇ **ਸਥਿਰਤਾ (stability)** ਨੂੰ ਯਕੀਨੀ ਬਣਾਉਂਦਾ ਹੈ।

ਟ੍ਰੂਥ ਟੇਬਲ (Truth Table):

J	K	CLK	Q (ਅਗਲਾ ਸਥਿਤੀ)	ਵਿਵਰਣ
0	0	↑	Q (ਕੋਈ ਬਦਲਾਅ ਨਹੀਂ)	ਮੈਮੋਰੀ ਸਥਿਤੀ
0	1	↑	0	ਰੀਸੈੱਟ
1	0	↑	1	ਸੈੱਟ
1	1	↑	$\bar{Q}$ (ਟੌਗਲ)	ਆਉਟਪੁੱਟ ਟੌਗਲ ਹੋਵੇਗਾ

(↑ ਕਲੌਕ ਦੇ rising edge ਨੂੰ ਦਰਸਾਉਂਦਾ ਹੈ)

ਲਾਜਿਕ ਸਰਕਟ (Logic Circuit): JK ਫਲਿਪ-ਫਲੌਪ ਨੂੰ AND, OR, NOT ਅਤੇ NAND ਗੇਟਾਂ ਦੀ ਵਰਤੋਂ ਕਰਕੇ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ ਜਾਂ **ਦੇ SR ਫਲਿਪ-ਫਲੌਪ** ਨਾਲ ਵਾਧੂ ਲਾਜਿਕ ਜੋੜ ਕੇ। J ਅਤੇ K ਇਨਪੁੱਟ ਨੂੰ ਕਲੌਕ ਅਤੇ ਆਉਟਪੁੱਟ Q/$\bar{Q}$ ਨਾਲ **AND** ਕਰਕੇ **Set/Reset** ਅਵਸਥਾਵਾਂ ਬਣਾਈਆਂ ਜਾਂਦੀਆਂ ਹਨ।

ਵਰਤੋਂ (Applications): JK ਫਲਿਪ-ਫਲੌਪ **ਕਾਊਂਟਰਸ**, **ਸ਼ਿਫਟ ਰਜਿਸਟਰਸ**, ਅਤੇ **ਮੈਮੋਰੀ ਸਟੋਰੇਜ** ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਹਨ ਜਿੱਥੇ **ਨਿਯੰਤਰਿਤ ਟੌਗਲਿੰਗ** ਜਾਂ **ਸਥਿਤੀ ਬਦਲਾਅ** ਜ਼ਰੂਰੀ ਹੁੰਦੇ ਹਨ।

Q. Explain the working of following flip-flops with a truth table and logic circuit: (Nov 23)
a) R-S Flip Flop
b) Master-Slave J-K Flip Flop

Ans. **a) R-S (Reset-Set) Flip-Flop:** The **RS Flip-Flop** is a basic **bistable circuit** with two inputs: **R (Reset)** and **S (Set)**, and two outputs: **Q and $\bar{Q}$**. It stores a single bit of data and changes state based on the input values. It can be built using **NOR** or **NAND** gates.

Truth Table (NOR-based RS Flip-Flop):

S	R	Q (Next)	State
0	0	No change	Memory
0	1	0	Reset
1	0	1	Set
1	1	Invalid	Not allowed

- **Q = 1**, when **S = 1, R = 0** (Set)
- **Q = 0**, when **S = 0, R = 1** (Reset)

Logic Circuit:

Two cross-coupled NOR gates with S and R inputs.

b) Master-Slave J-K Flip-Flop: The **Master-Slave JK Flip-Flop** is a sequential circuit composed of two JK flip-flops connected in series. It eliminates race conditions by using a **master-slave configuration** controlled by the clock.

Truth Table:

J	K	CLK	Q (Next)	Description
0	0	↑	Q	No change
0	1	↑	0	Reset
1	0	↑	1	Set
1	1	↑	$\bar{Q}$	Toggle

Working:

- When **CLK = 1**, the master latches input and the slave is off.
- When **CLK = 0**, the slave updates its output based on the master.
- This ensures output changes only on the clock edge, preventing glitches

(a) R-S (Reset-Set) ਫਲਿਪ-ਫਲੌਪ: **RS ਫਲਿਪ-ਫਲੌਪ** ਇੱਕ ਬੁਨਿਆਦੀ **ਬਿਸਟੇਬਲ ਸਰਕਟ** ਹੈ ਜਿਸ ਵਿੱਚ **ਦੋ ਇਨਪੁੱਟ** ਹੁੰਦੇ ਹਨ: **R (ਰੀਸੈੱਟ)** ਅਤੇ **S (ਸੈੱਟ)**, ਅਤੇ **ਦੋ ਆਉਟਪੁੱਟ**: **Q ਅਤੇ $\bar{Q}$**। ਇਹ ਇੱਕ ਬਿੱਟ ਡਾਟਾ ਸੰਭਾਲਦਾ ਹੈ ਅਤੇ ਇਨਪੁੱਟਾਂ ਦੇ ਅਧਾਰ 'ਤੇ ਆਪਣੀ ਅਵਸਥਾ ਬਦਲਦਾ ਹੈ। ਇਸਨੂੰ **NOR ਜਾਂ NAND ਗੇਟਸ** ਦੀ ਵਰਤੋਂ ਨਾਲ ਬਣਾਇਆ ਜਾ ਸਕਦਾ ਹੈ।

NOR-ਅਧਾਰਤ Truth Table:

S	R	Q (ਅਗਲਾ ਸਥਿਤੀ)	ਸਥਿਤੀ
0	0	No Change	ਮੈਮੋਰੀ
0	1	0	ਰੀਸੈੱਟ
1	0	1	ਸੈੱਟ
1	1	Invalid	ਅਮਾਨਯ

- **Q = 1**, ਜਦੋਂ **S = 1, R = 0** (Set)
- **Q = 0**, ਜਦੋਂ **S = 0, R = 1** (Reset)

ਲਾਜਿਕ ਸਰਕਟ:

ਦੋ NOR ਗੇਟ ਜੋ ਆਪਸ ਵਿੱਚ **ਕਰਾਸ-ਕਪਲਡ** ਹਨ, ਇਨਪੁੱਟ ਹਨ **S ਅਤੇ R**।

(b) ਮਾਸਟਰ-ਸਲੇਵ J-K ਫਲਿਪ-ਫਲੈਪ: ਮਾਸਟਰ-ਸਲੇਵ JK ਫਲਿਪ-ਫਲੈਪ ਇੱਕ **ਸੀਕਵੇਂਸ਼ਲ ਸਰਕਟ** ਹੈ ਜੋ **ਦੋ JK ਫਲਿਪ-ਫਲੈਪਸ** ਨੂੰ ਲੜੀਵਾਰ ਜੋੜ ਕੇ ਬਣਾਇਆ ਜਾਂਦਾ ਹੈ। ਇਹ **ਰੇਸ ਅਵਸਥਾ (Race Condition)** ਨੂੰ ਦੂਰ ਕਰਦਾ ਹੈ ਕਿਉਂਕਿ ਇਹ **ਮਾਸਟਰ-ਸਲੇਵ ਵਿਧੀ** 'ਤੇ ਕੰਮ ਕਰਦਾ ਹੈ ਜੋ **ਕਲੈਕ (Clock)** ਦੁਆਰਾ ਨਿਯੰਤਰਿਤ ਹੁੰਦਾ ਹੈ।

Truth Table:

J	K	CLK	Q (ਅਗਲਾ ਸਥਿਤੀ)	ਵਿਵਰਣ
0	0	↑	Q	ਕੋਈ ਬਦਲਾਅ ਨਹੀਂ
0	1	↑	0	ਰੀਸੈੱਟ
1	0	↑	1	ਸੈੱਟ
1	1	↑	$\bar{Q}$	ਟੈਗਲ

ਕਾਰਜ ਵਿਧੀ (Working):

- ਜਦੋਂ **CLK = 1**, **ਮਾਸਟਰ** ਇਨਪੁੱਟ ਨੂੰ ਲੈ ਲੈਂਦਾ ਹੈ ਅਤੇ **ਸਲੇਵ** ਬੰਦ ਰਹਿੰਦਾ ਹੈ।
- ਜਦੋਂ **CLK = 0**, **ਸਲੇਵ** ਮਾਸਟਰ ਦੀ ਅਵਸਥਾ ਦੇ ਅਨੁਸਾਰ ਆਉਟਪੁੱਟ ਨੂੰ ਅਪਡੇਟ ਕਰਦਾ ਹੈ।

ਇਹ ਵਿਧੀ **ਕਲੈਕ ਐਜ** 'ਤੇ ਹੀ ਆਉਟਪੁੱਟ ਨੂੰ ਬਦਲਣ ਦੀ ਆਗਿਆ ਦਿੰਦੀ ਹੈ, ਜਿਸ ਨਾਲ **ਗਲਿਚ (glitch)** ਨਹੀਂ ਆਉਂਦੇ।

Q. a) What is meant by JK Flip Flop? Explain the race-around condition in detail.
b) Write a detailed note on Applications of Flip-Flops. (Nov 24)

Ans. **a) JK Flip-Flop and Race-Around Condition:** A **JK Flip-Flop** is a sequential logic circuit with inputs **J**, **K**, and **Clock (CLK)**, and outputs **Q** and $\bar{Q}$. It performs the functions of Set, Reset, and Toggle based on input combinations. When **J = K = 1**, it toggles the output. The **Race-Around Condition** occurs when **J = K = 1** and the clock pulse remains **high for a longer duration**. In this case, the output toggles repeatedly within a single clock cycle, leading to unpredictable behavior. To avoid this, techniques like **edge-triggering** or **master-slave JK flip-flops** are used for stable operation.

b) Applications of Flip-Flops: Flip-Flops are fundamental memory elements in digital electronics used to store a **single bit of data**. They are widely used in various applications:

- **Counters:** Used in digital clocks, timers, and frequency dividers.
- **Registers:** Store and shift data in CPUs and memory units.
- **Data Storage:** Temporary data holding in RAM and buffers.
- **Control Circuits:** Used in sequence detectors and state machines.
- **Frequency Division:** Divide input clock frequency in digital systems. Their ability to switch states predictably makes them essential in **synchronous digital systems**, forming the core of all digital computing hardware.

a) JK ਫਲਿਪ-ਫਲੈਪ ਅਤੇ ਰੇਸ ਅਰਾਊਂਡ ਸਥਿਤੀ (Race-Around Condition): JK ਫਲਿਪ-ਫਲੈਪ ਇੱਕ **ਸੀਕਵੇਂਸ਼ਲ ਲਾਜਿਕ ਸਰਕਟ** ਹੈ ਜਿਸ ਵਿੱਚ **J, K ਅਤੇ ਕਲੈਕ (CLK)** ਇਨਪੁੱਟ ਅਤੇ **Q ਅਤੇ $\bar{Q}$** ਆਉਟਪੁੱਟ ਹੁੰਦੇ ਹਨ। ਇਹ ਇਨਪੁੱਟ ਦੇ ਮਿਲਾਓ ਅਨੁਸਾਰ **ਸੈੱਟ (Set)**, **ਰੀਸੈੱਟ (Reset)** ਅਤੇ **ਟੈਗਲ (Toggle)** ਫੰਕਸ਼ਨ ਕਰਦਾ ਹੈ। ਜਦੋਂ **J = K = 1** ਹੋਵੇ, ਤਾਂ ਆਉਟਪੁੱਟ **toggle** ਕਰਦਾ ਹੈ (Q ਆਪਣੀ ਪਿਛਲੀ ਅਵਸਥਾ ਦੇ ਉਲਟ ਹੋ ਜਾਂਦੀ ਹੈ)। **ਰੇਸ-ਅਰਾਊਂਡ ਸਥਿਤੀ** ਤਾਂ ਹੁੰਦੀ ਹੈ ਜਦੋਂ **J = K = 1** ਹੋਣ ਤੇ **ਕਲੈਕ ਪਲਸ (Clock Pulse)** ਜ਼ਿਆਦਾ ਸਮੇਂ ਲਈ **HIGH** ਰਹਿੰਦੀ ਹੈ। ਇਸ ਦੌਰਾਨ, ਆਉਟਪੁੱਟ ਇੱਕੋ ਕਲੈਕ ਸਾਈਕਲ ਵਿੱਚ ਬਾਰ-ਬਾਰ toggle ਹੋਣ ਲੱਗਦੀ ਹੈ, ਜਿਸ ਨਾਲ ਅਣਅਨੁਮਾਨਤ ਵਿਹਾਰ (Unpredictable Behavior) ਪੈਦਾ ਹੁੰਦਾ ਹੈ। ਇਸ ਸਮੱਸਿਆ ਤੋਂ ਬਚਣ ਲਈ, ਹੇਠਾਂ ਦਿੱਤੀਆਂ ਤਕਨੀਕਾਂ ਵਰਤੀ ਜਾਂਦੀਆਂ ਹਨ:

- **Edge-Triggered JK Flip-Flop**
- **Master-Slave JK Flip-Flop**

ਇਹ ਤਕਨੀਕਾਂ ਆਉਟਪੁੱਟ ਨੂੰ ਸਿਰਫ਼ ਕਲੌਕ ਦੇ ਐਜ 'ਤੇ ਹੀ ਬਦਲਣ ਦੀ ਆਗਿਆ ਦਿੰਦੀਆਂ ਹਨ, ਜਿਸ ਨਾਲ ਰੇਸ ਅਰਾਊਂਡ ਨੂੰ ਰੋਕਿਆ ਜਾਂਦਾ ਹੈ।

b) ਫਲਿਪ-ਫਲੌਪਸ ਦੀਆਂ ਐਪਲੀਕੇਸ਼ਨਜ਼ (Applications of Flip-Flops): ਫਲਿਪ-ਫਲੌਪ ਡਿਜ਼ੀਟਲ ਇਲੈਕਟ੍ਰਾਨਿਕਸ ਵਿੱਚ ਇੱਕ **ਮੁੱਢਲਾ ਮੈਮੋਰੀ ਇਲੈਮੈਂਟ** ਹੈ ਜੋ **ਇੱਕ ਬਿੱਟ** ਡਾਟਾ ਸਟੋਰ ਕਰਦਾ ਹੈ। ਇਹ ਕਈ ਤਰ੍ਹਾਂ ਦੇ ਉਪਕਰਨਾਂ ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਹਨ:

- **ਕਾਊਂਟਰ (Counters):** ਡਿਜੀਟਲ ਘੜੀਆਂ, ਟਾਈਮਰ, ਅਤੇ ਫ੍ਰੀਕਵੈਂਸੀ ਡਿਵਾਈਡਰ ਵਿੱਚ।
- **ਰਜਿਸਟਰ (Registers):** ਡਾਟਾ ਨੂੰ ਸਟੋਰ ਅਤੇ ਸ਼ਿਫਟ ਕਰਨ ਲਈ CPUs ਅਤੇ ਮੈਮੋਰੀ ਯੂਨਿਟ ਵਿੱਚ।
- **ਡਾਟਾ ਸਟੋਰੇਜ:** RAM ਜਾਂ ਬਫਰ ਵਿੱਚ ਆਸਥਾਈ ਡਾਟਾ ਰੱਖਣ ਲਈ।
- **ਕੰਟਰੋਲ ਸਰਕਟਸ:** ਸੀਕਵੇਂਸ ਡਿਟੈਕਟਰ ਅਤੇ ਸਟੇਟ ਮਸ਼ੀਨ ਵਿੱਚ।
- **ਫ੍ਰੀਕਵੈਂਸੀ ਡਿਵਿਜ਼ਨ:** ਘੜੀ ਦੀ ਆਵ੍ਰਿਤੀ ਨੂੰ ਵੰਡਣ ਲਈ।

ਇਹਨਾਂ ਦੀ **ਪ੍ਰਡਿਕਟੇਬਲ ਸਟੇਟ-ਸਵਿੱਚਿੰਗ ਸਮਰੱਥਾ** ਕਰਕੇ ਇਹ **ਸਿੰਕ੍ਰੋਨਸ ਡਿਜ਼ੀਟਲ ਸਿਸਟਮਜ਼** ਦੇ ਮੁੱਖ ਹਿੱਸੇ ਵਜੋਂ ਕੰਮ ਕਰਦੇ ਹਨ, ਜੋ ਕਿ ਹਰ ਡਿਜ਼ੀਟਲ ਕੰਪਿਊਟਿੰਗ ਹਾਰਡਵੇਅਰ ਦੀ ਨਿਵ ਹੈ।

Unit-IV: Computer Organization
Short Answer Questions:

Q. What is an Address bus? (Nov 22)
Ans. An **Address Bus** carries the address of memory locations where data is to be read or written; it is unidirectional.
ਐਡਰੈੱਸ ਬੱਸ ਉਹ ਪਤਾ ਲੈ ਜਾਂਦੀ ਹੈ ਜਿਸ ਮੈਮੋਰੀ ਸਥਾਨ ਤੋਂ ਡਾਟਾ ਲੈਣਾ ਜਾਂ ਲਿਖਣਾ ਹੋਵੇ। ਇਹ **ਇੱਕ-ਦਿਸ਼ਾ (Unidirectional)** ਹੁੰਦੀ ਹੈ।

Q. Write a short note on Computer registers. (Nov 22)
Ans. **Computer Registers** are small, high-speed storage locations within the CPU used to hold data temporarily during processing.
ਕੰਪਿਊਟਰ ਰਜਿਸਟਰ CPU ਵਿੱਚ ਛੋਟੀਆਂ ਤੇਜ਼ ਸਟੋਰੇਜ ਥਾਵਾਂ ਹੁੰਦੀਆਂ ਹਨ। ਇਹ ਪ੍ਰੋਸੈਸ ਦੌਰਾਨ ਅਸਥਾਈ ਤੌਰ 'ਤੇ ਡਾਟਾ ਰੱਖਣ ਲਈ ਵਰਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।

Q. List the different types of Instruction Formats. (Nov 22)
Ans. **Instruction Formats** include zero-address, one-address, two-address, and three-address formats, based on the number of operands.
ਇਨਸਟਰੱਕਸ਼ਨ ਫਾਰਮੈਟ ਵਿੱਚ 0, 1, 2 ਜਾਂ 3 ਓਪਰੇਂਡ ਹੋ ਸਕਦੇ ਹਨ। ਇਹ ਉਨ੍ਹਾਂ ਦੀ ਗਿਣਤੀ ਦੇ ਆਧਾਰ 'ਤੇ ਵਰਗੀਕ੍ਰਿਤ ਹੁੰਦੇ ਹਨ।

Q. Differentiate between direct and indirect instruction modes. (Nov 23)
Ans. **Direct instruction mode** accesses memory directly, while **indirect mode** refers to a memory location that holds the actual address.
Direct mode ਵਿੱਚ ਮੈਮੋਰੀ ਦਾ ਪਤਾ ਸਿੱਧਾ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ। Indirect mode ਵਿੱਚ ਇੱਕ ਸਥਾਨ ਦਿੱਤਾ ਜਾਂਦਾ ਹੈ ਜੋ ਅਸਲੀ ਪਤਾ ਰੱਖਦਾ ਹੈ।

Q. Explain three address instruction format. (Nov 23)
Ans. In a **three-address instruction format**, the instruction contains three operands—typically two sources and one destination.
ਇਸ ਫਾਰਮੈਟ ਵਿੱਚ ਤਿੰਨ ਓਪਰੇਂਡ ਹੁੰਦੇ ਹਨ—ਆਮ ਤੌਰ 'ਤੇ ਦੋ ਸਰੋਤ ਅਤੇ ਇੱਕ ਗੰਤਵਯ।

Q. Define computer architecture. (Nov 23)
Ans. **Computer Architecture** defines the structure, behavior, and design of a computer system's components and their interconnection.
ਕੰਪਿਊਟਰ ਆਰਕੀਟੈਕਚਰ ਸਿਸਟਮ ਦੇ ਹਿੱਸਿਆਂ ਦੀ ਬਣਤਰ, ਵਿਹਾਰ ਅਤੇ ਡਿਜ਼ਾਈਨ ਨੂੰ ਦਰਸਾਉਂਦੀ ਹੈ।

Q. Write a note on Von Neumann architecture. (Nov 23)
Ans. The **Von Neumann architecture** is a computer design model where data and instructions share the same memory and pathways.
ਇਹ ਮਾਡਲ ਉਹ ਹੈ ਜਿਸ ਵਿੱਚ ਡਾਟਾ ਅਤੇ ਨਿਰਦੇਸ਼ ਇਕੋ ਮੈਮੋਰੀ ਅਤੇ ਰਸਤੇ ਸਾਂਝੇ ਕਰਦੇ ਹਨ। *((Nov 23))*

Q. Define Control Bus and Data Bus. (Nov 24)
Ans. The **Control Bus** carries control signals, while the **Data Bus** carries actual data between components in a computer.
ਕੰਟਰੋਲ ਬੱਸ ਕੰਪੋਨੈਂਟਾਂ ਵਿਚਕਾਰ ਕੰਟਰੋਲ ਸਿਗਨਲ ਲਿਜਾਂਦੀ ਹੈ। ਡਾਟਾ ਬੱਸ ਅਸਲੀ ਡਾਟਾ ਨੂੰ ਇਕ ਹਿੱਸੇ ਤੋਂ ਦੂਜੇ ਹਿੱਸੇ ਤੱਕ ਲਿਜਾਂਦੀ ਹੈ।

Long Answer Questions:

Q. Explain the 16-bit common bus system of the computer with the help of a neat diagram. (Nov 22)

Ans. The **16-bit Common Bus System** is a structural model used in computer architecture to allow multiple **registers**, **ALU**, and **memory units** to share a single data path for efficient data transfer. The bus is **16 bits wide**, meaning it can carry 16 bits of data simultaneously.

Components:

- **Registers:** Multiple registers like AR (Address Register), DR (Data Register), AC (Accumulator), PC (Program Counter), etc.
- **ALU (Arithmetic Logic Unit):** Performs arithmetic and logic operations.
- **Control Lines:** Direct data flow by enabling or disabling specific components.
- **Bus:** A set of 16 parallel lines used to transfer data between components.

Working:

- Only **one register at a time** can place data on the bus using a **bus enable line**.
- Other registers or the ALU can **read from the bus** when their **load signal** is active.
- The **control unit** manages timing and control signals for coordinated data flow.

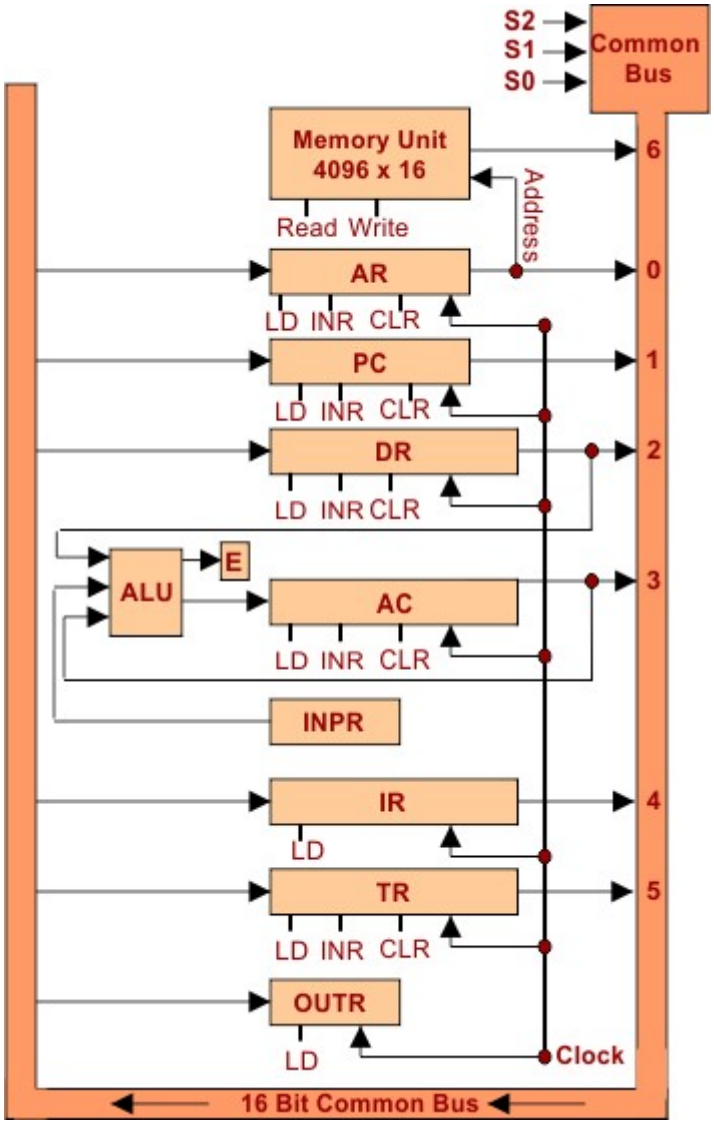
16-ਬਿਟ ਕਾਮਨ ਬੱਸ ਸਿਸਟਮ (Common Bus System) 16-ਬਿਟ ਕਾਮਨ ਬੱਸ ਸਿਸਟਮ ਇੱਕ ਸੰਰਚਨਾਤਮਕ ਮਾਡਲ ਹੈ ਜੋ ਕੰਪਿਊਟਰ ਆਰਕੀਟੈਕਚਰ ਵਿੱਚ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ। ਇਹ ਮਾਡਲ ਰਜਿਸਟਰਾਂ, ALU (ਐਰੀਥਮੈਟਿਕ ਲਾਜਿਕ ਯੂਨਿਟ), ਅਤੇ ਮੈਮੋਰੀ ਯੂਨਿਟ ਨੂੰ ਇੱਕੋ ਡਾਟਾ ਪਾਥ (ਬੱਸ) ਸਾਂਝਾ ਕਰਨ ਦੀ ਆਗਿਆ ਦਿੰਦਾ ਹੈ ਤਾਂ ਜੋ ਡਾਟਾ ਦਾ ਟ੍ਰਾਂਸਫਰ ਪ੍ਰਭਾਵਸ਼ਾਲੀ ਢੰਗ ਨਾਲ ਕੀਤਾ ਜਾ ਸਕੇ। ਇਹ **ਬੱਸ 16-ਬਿਟ ਚੌੜੀ ਹੁੰਦੀ ਹੈ**, ਜਿਸਦਾ ਅਰਥ ਹੈ ਕਿ ਇਹ ਇੱਕ ਸਮੇਂ ਵਿੱਚ 16 ਬਿਟ ਡਾਟਾ ਟ੍ਰਾਂਸਫਰ ਕਰ ਸਕਦੀ ਹੈ।

ਮੁੱਖ ਭਾਗ (Components):

- **ਰਜਿਸਟਰ (Registers):** ਕਈ ਤਰ੍ਹਾਂ ਦੇ ਰਜਿਸਟਰ ਜਿਵੇਂ AR (ਐਡਰੈੱਸ ਰਜਿਸਟਰ), DR (ਡਾਟਾ ਰਜਿਸਟਰ), AC (ਐਕਯੂਮੂਲੇਟਰ), PC (ਪਰੋਗ੍ਰਾਮ ਕਾਊਂਟਰ) ਆਦਿ।
- **ALU (Arithmetic Logic Unit):** ਗਣਿਤ ਅਤੇ ਲਾਜਿਕ ਓਪਰੇਸ਼ਨਾਂ ਨੂੰ ਕਰਦਾ ਹੈ।
- **ਕੰਟਰੋਲ ਲਾਈਨਾਂ (Control Lines):** ਡਾਟਾ ਦੇ ਫ਼ਲੋ ਨੂੰ ਨਿਯੰਤ੍ਰਿਤ ਕਰਦੀਆਂ ਹਨ। ਇਹ ਕੁਝ ਭਾਗਾਂ ਨੂੰ ਐਨੇਬਲ ਜਾਂ ਡਿਸਏਬਲ ਕਰਦੀਆਂ ਹਨ।
- **ਬੱਸ (Bus):** 16 ਸਮਾਂਤਰ ਲਾਈਨਾਂ ਦਾ ਇੱਕ ਸਮੂਹ ਜੋ ਡਾਟਾ ਨੂੰ ਰਜਿਸਟਰਾਂ, ALU ਅਤੇ ਮੈਮੋਰੀ ਵਿਚਕਾਰ ਟ੍ਰਾਂਸਫਰ ਕਰਨ ਲਈ ਵਰਤੀ ਜਾਂਦੀ ਹੈ।

ਕਾਰਜ ਕਰਨ ਦਾ ਢੰਗ (Working):

- ਇੱਕ ਸਮੇਂ ਵਿੱਚ **ਸਿਰਫ਼ ਇੱਕ ਰਜਿਸਟਰ** ਹੀ ਬੱਸ 'ਤੇ ਡਾਟਾ ਰੱਖ ਸਕਦਾ ਹੈ। ਇਹ **bus enable line** ਰਾਹੀਂ ਨਿਯੰਤ੍ਰਿਤ ਹੁੰਦਾ ਹੈ।
- ਹੋਰ ਰਜਿਸਟਰ ਜਾਂ ALU **ਬੱਸ ਤੋਂ ਡਾਟਾ ਲੋਡ** ਕਰ ਸਕਦੇ ਹਨ ਜਦੋਂ ਉਨ੍ਹਾਂ ਦੀ **load signal** ਐਕਟਿਵ ਹੋਵੇ।
- **ਕੰਟਰੋਲ ਯੂਨਿਟ** ਟਾਈਮਿੰਗ ਅਤੇ ਕੰਟਰੋਲ ਸਿਗਨਲਾਂ ਨੂੰ ਸੰਭਾਲਦੀ ਹੈ ਤਾਂ ਜੋ **ਸਮਨਵਿਤ ਡਾਟਾ ਪ੍ਰਵਾਹ (coordinated data flow)** ਹੋ ਸਕੇ।



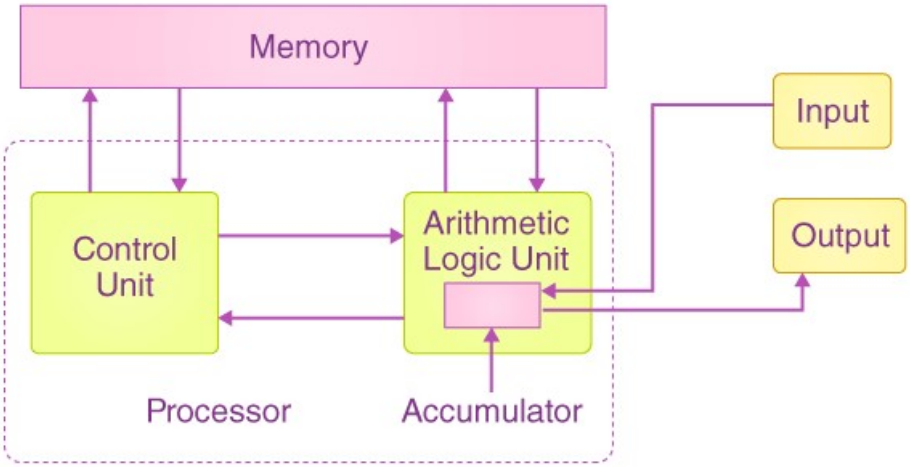
Q. Explain with diagrammatic illustration the Von Neumann architecture. (Nov 22)
Ans. The **Von Neumann Architecture** is a computer design model proposed by **John von Neumann** in 1945. It forms the basis of most modern computers. The architecture uses a **single memory** to store both **data and program instructions**, and processes them using a **Central Processing Unit (CPU)**.

Key Components:

1. **Central Processing Unit (CPU):** Contains the **Arithmetic Logic Unit (ALU)** and **Control Unit (CU)**.
2. **Memory Unit:** Stores data and instructions.
3. **Input/Output Devices:** Facilitate communication with the external world.
4. **Control Bus, Data Bus, Address Bus:** Used for transferring signals, data, and memory addresses.

Working Principle:

- Instructions and data are fetched from the **same memory**.
- The **Control Unit** fetches, decodes, and executes instructions sequentially.
- The **ALU** performs arithmetic and logical operations.
- Input and output units allow interaction with the user and external devices.



Von Neumann Basic Structure

ਵੋਨ ਨਿਊਮੈਨ ਆਰਕੀਟੈਕਚਰ (Von Neumann Architecture): ਵੋਨ ਨਿਊਮੈਨ ਆਰਕੀਟੈਕਚਰ ਇੱਕ ਕੰਪਿਊਟਰ ਡਿਜ਼ਾਇਨ ਮਾਡਲ ਹੈ ਜੋ **1945 ਵਿੱਚ John von Neumann** ਵਲੋਂ ਪ੍ਰਸਤਾਵਿਤ ਕੀਤਾ ਗਿਆ ਸੀ। ਇਹ ਆਰਕੀਟੈਕਚਰ ਜ਼ਿਆਦਾਤਰ ਆਧੁਨਿਕ ਕੰਪਿਊਟਰਾਂ ਦੀ ਨੀਂਹ ਰੱਖਦਾ ਹੈ। ਇਸ ਮਾਡਲ ਵਿੱਚ **ਇੱਕੋ ਮੈਮੋਰੀ** ਉਪਯੋਗ ਕੀਤੀ ਜਾਂਦੀ ਹੈ ਜਿਸ ਵਿੱਚ **ਡਾਟਾ ਅਤੇ ਪ੍ਰੋਗਰਾਮ ਨਿਰਦੇਸ਼ (instructions)** ਦੋਹਾਂ ਸੰਭਾਲੇ ਜਾਂਦੇ ਹਨ, ਅਤੇ ਇਹਨਾਂ ਨੂੰ **CPU ਰਾਹੀਂ ਪ੍ਰੋਸੈਸ** ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

ਮੁੱਖ ਭਾਗ (Key Components):

- **ਸੈਂਟਰਲ ਪ੍ਰੋਸੈਸਿੰਗ ਯੂਨਿਟ (CPU):** ਇਸ ਵਿੱਚ ਦੋ ਭਾਗ ਹੁੰਦੇ ਹਨ:
  - **ALU (Arithmetic Logic Unit):** ਗਣਿਤ ਅਤੇ ਲਾਜਿਕ ਓਪਰੇਸ਼ਨ ਕਰਦੀ ਹੈ।
  - **CU (Control Unit):** ਨਿਰਦੇਸ਼ਾਂ ਨੂੰ ਲੈਣ, ਡੀਕੋਡ ਕਰਨ ਅਤੇ ਚਲਾਉਣ ਦੀ ਜ਼ਿੰਮੇਵਾਰੀ ਲੈਂਦੀ ਹੈ।
- **ਮੈਮੋਰੀ ਯੂਨਿਟ (Memory Unit):** ਡਾਟਾ ਅਤੇ ਨਿਰਦੇਸ਼ਾਂ ਨੂੰ ਸੰਭਾਲਦੀ ਹੈ।
- **ਇਨਪੁਟ/ਆਉਟਪੁਟ ਡਿਵਾਈਸ (Input/Output Devices):** ਬਾਹਰੀ ਦੁਨੀਆਂ ਨਾਲ ਸੰਚਾਰ ਕਰਨ ਲਈ ਵਰਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।
- **ਬੱਸ ਸਿਸਟਮ (Bus System):**
  - **ਕੰਟਰੋਲ ਬੱਸ (Control Bus)** – ਨਿਯੰਤ੍ਰਣ ਸੰਕੇਤਾਂ ਲਈ।
  - **ਡਾਟਾ ਬੱਸ (Data Bus)** – ਡਾਟਾ ਦੇ ਟ੍ਰਾਂਸਫਰ ਲਈ।
  - **ਐਡਰੈੱਸ ਬੱਸ (Address Bus)** – ਮੈਮੋਰੀ ਐਡਰੈੱਸਾਂ ਦੇ ਟ੍ਰਾਂਸਫਰ ਲਈ।

ਕਾਰਜ ਕਰਨ ਦਾ ਤਰੀਕਾ (Working Principle):

- **ਇੰਸਟ੍ਰਕਸ਼ਨ ਅਤੇ ਡਾਟਾ** ਇੱਕੋ ਮੈਮੋਰੀ ਤੋਂ ਲਏ ਜਾਂਦੇ ਹਨ।
- **Control Unit** ਨਿਰਦੇਸ਼ ਨੂੰ **ਲੈਦੀ, ਡੀਕੋਡ ਕਰਦੀ ਅਤੇ ਲਾਗੂ** ਕਰਦੀ ਹੈ।
- **ALU** ਗਣਿਤ ਅਤੇ ਤਰਕਸੰਗਤ ਕਾਰਜ ਕਰਦੀ ਹੈ।
- **Input/Output ਯੂਨਿਟ** ਯੂਜ਼ਰ ਅਤੇ ਬਾਹਰੀ ਡਿਵਾਈਸਾਂ ਨਾਲ ਸੰਚਾਰ ਦੀ ਆਗਿਆ ਦਿੰਦੇ ਹਨ।

Q. Write a note on register transfer and micro operations. (Nov 23)

Ans. **Register Transfer** refers to the process of moving data from one register to another within a computer system. It is fundamental to the internal operation of digital systems, especially in CPUs, where data must frequently be moved, modified, or stored. Register transfer operations are controlled by signals generated by the **control unit**, and are represented using symbolic notations such as:
R1 ← R2, meaning data from register R2 is transferred to R1.
Micro-operations are low-level operations performed on the data stored in registers. These are the elementary operations that occur during the execution of machine instructions. Micro-operations are categorized into four main types:

1. **Register Transfer Micro-operations:** Move data between registers.
2. **Arithmetic Micro-operations:** Perform operations like addition, subtraction, increment, etc.
3. **Logic Micro-operations:** Perform bitwise operations like AND, OR, NOT, XOR.
4. **Shift Micro-operations:** Perform shift operations (left, right, logical, or arithmetic).

Each instruction in a computer is executed as a sequence of micro-operations. These operations are synchronized with clock cycles and managed by the **control logic** of the processor. Together, register transfers and micro-operations form the building blocks of instruction execution in computer architecture.

ਰਜਿਸਟਰ ਟ੍ਰਾਂਸਫਰ (Register Transfer):

ਰਜਿਸਟਰ ਟ੍ਰਾਂਸਫਰ ਦਾ ਅਰਥ ਹੈ **ਇੱਕ ਰਜਿਸਟਰ ਤੋਂ ਦੂਜੇ ਰਜਿਸਟਰ ਵਿਚ ਡਾਟਾ ਨੂੰ ਭੇਜਣ ਦੀ ਪ੍ਰਕਿਰਿਆ**, ਜੋ ਕਿ ਕੰਪਿਊਟਰ ਸਿਸਟਮ ਦੇ ਅੰਦਰੂਨੀ ਕਾਰਜ ਲਈ ਬਹੁਤ ਹੀ ਆਵਸ਼ਯਕ ਹੈ। ਖਾਸ ਕਰਕੇ **CPU** ਵਿੱਚ, ਜਿੱਥੇ ਡਾਟਾ ਨੂੰ ਲਗਾਤਾਰ ਮੂਵ, ਮੋਡੀਫਾਈ ਜਾਂ ਸਟੋਰ ਕੀਤਾ ਜਾਂਦਾ ਹੈ। ਇਹ ਟ੍ਰਾਂਸਫਰ ਕੰਟਰੋਲ ਯੂਨਿਟ ਵੱਲੋਂ ਉਤਪੰਨ **ਕੰਟਰੋਲ ਸਿਗਨਲਜ਼** ਰਾਹੀਂ ਹੋਦਾ ਹੈ। ਇਸਨੂੰ ਨਿਮਨ ਲਿਖਤ ਰੂਪ ਵਿੱਚ ਦਰਸਾਇਆ ਜਾਂਦਾ ਹੈ:

R1 ← R2
ਅਰਥ: **ਰਜਿਸਟਰ R2 ਤੋਂ ਡਾਟਾ ਨੂੰ R1 ਵਿੱਚ ਭੇਜਿਆ ਗਿਆ ਹੈ।**

ਮਾਈਕ੍ਰੋ-ਓਪਰੇਸ਼ਨ (Micro-operations): ਮਾਈਕ੍ਰੋ-ਓਪਰੇਸ਼ਨ ਉਹ ਨਿੱਕੇ-ਨਿੱਕੇ ਕਾਰਜ ਹਨ ਜੋ **ਰਜਿਸਟਰਾਂ ਵਿੱਚ ਮੌਜੂਦ ਡਾਟਾ** 'ਤੇ ਕੀਤੇ ਜਾਂਦੇ ਹਨ। ਇਹ ਕੰਪਿਊਟਰ ਦੀ ਹਰੇਕ **ਇੰਸਟ੍ਰਕਸ਼ਨ** ਨੂੰ ਚਲਾਉਣ ਸਮੇਂ ਆਮ ਤੌਰ 'ਤੇ ਕਈ **ਕਲੈਕ ਸਾਈਕਲਾਂ** ਵਿੱਚ ਕੀਤੇ ਜਾਂਦੇ ਹਨ।
ਮਾਈਕ੍ਰੋ-ਓਪਰੇਸ਼ਨ ਦੀਆਂ ਮੁੱਖ ਤਰ੍ਹਾਂ ਚਾਰ ਸ਼੍ਰੇਣੀਆਂ ਹੁੰਦੀਆਂ ਹਨ:

1. **Register Transfer Micro-operations:** ਰਜਿਸਟਰਾਂ ਵਿਚਕਾਰ ਡਾਟਾ ਮੂਵ ਕਰਦੇ ਹਨ।
2. **Arithmetic Micro-operations:** ਜਿਵੇਂ ਜੋੜ (Addition), ਘਟਾਉਣਾ (Subtraction), ਵਾਧਾ (Increment) ਆਦਿ।
3. **Logic Micro-operations:** ਬਿੱਟਵਾਈਜ਼ AND, OR, NOT, XOR ਵਰਗੀਆਂ ਲਾਜਿਕਲ ਕਾਰਵਾਈਆਂ।
4. **Shift Micro-operations:** ਡਾਟਾ ਨੂੰ ਖੱਬੇ ਜਾਂ ਸੱਜੇ ਸ਼ਿਫਟ ਕਰਨਾ (Logical ਜਾਂ Arithmetic Shift)।

ਸੰਖੇਪ ਵਿੱਚ: ਕਿਸੇ ਵੀ ਕੰਪਿਊਟਰ ਵਿੱਚ **ਹਰ ਇੰਸਟ੍ਰਕਸ਼ਨ ਦੀ ਪ੍ਰਕਿਰਿਆ** ਕਈ ਮਾਈਕ੍ਰੋ-ਓਪਰੇਸ਼ਨਾਂ ਦੀ ਲੜੀ ਰੂਪ ਵਿੱਚ ਹੁੰਦੀ ਹੈ। ਇਹ ਕਾਰਵਾਈਆਂ **ਕਲੈਕ ਸਾਈਕਲ** ਨਾਲ ਸਮਨ੍ਵਿਤ ਹੋਕੇ **ਕੰਟਰੋਲ ਯੂਨਿਟ** ਦੁਆਰਾ ਸੰਚਾਲਿਤ ਕੀਤੀਆਂ ਜਾਂਦੀਆਂ ਹਨ।
Register Transfer ਅਤੇ Micro-operations, ਕੰਪਿਊਟਰ ਆਰਕੀਟੈਕਚਰ ਵਿੱਚ **ਇੰਸਟ੍ਰਕਸ਼ਨ ਐਗਜ਼ੀਕਿਊਸ਼ਨ ਦੇ ਨਿਰੀ ਅਹੰਸ** ਹਨ।

Q. Explain data movement among registers using 16 bit common system. (Nov 23)
Ans. In a **16-bit common bus system**, data movement among registers is handled through a **shared communication pathway** called the **bus**. The bus is **16 bits wide**, allowing it to transfer 16 bits of data at a time. This system is used in computer architecture to reduce wiring

complexity and allow efficient data flow among multiple components like registers, ALU, and memory.

Components Involved:

- **Registers** (e.g., AR, DR, AC, PC): Hold temporary data or addresses.
- **Bus**: A set of 16 parallel lines for data transfer.
- **Control Lines**: Determine which register puts data onto the bus and which receives it.

Data Movement Process:

1. The **control unit** activates the output control signal (e.g., DRout) to place data from **Data Register (DR)** onto the bus.
2. The bus carries the 16-bit data.
3. Another control signal (e.g., ACin) is activated to load the data into the **Accumulator (AC)**.
4. All data transfers are synchronized with the **clock** to ensure proper timing.

Example:

To move data from DR to AC:

DRout → Bus → ACin

This controlled, step-by-step movement ensures data integrity and efficient internal communication in a CPU.

16-ਬਿਟ ਸਾਂਝੀ ਬੱਸ ਪ੍ਰਣਾਲੀ ਵਿੱਚ ਡਾਟਾ ਦੀ ਚਲਾਏਗਈ ਪ੍ਰਕਿਰਿਆ: 16-ਬਿਟ ਕਾਮਨ ਬੱਸ ਸਿਸਟਮ ਵਿੱਚ, ਰਜਿਸਟਰਾਂ ਵਿਚਕਾਰ ਡਾਟਾ ਦੀ ਅਦਾਨ-ਪ੍ਰਦਾਨ ਇੱਕ ਸਾਂਝੀ ਸੰਚਾਰ ਮਾਰਗ (Bus) ਰਾਹੀਂ ਹੁੰਦੀ ਹੈ। ਇਹ ਬੱਸ 16-ਬਿਟ ਚੌੜੀ ਹੁੰਦੀ ਹੈ, ਜਿਸ ਨਾਲ ਇਹ ਇੱਕ ਵਾਰ ਵਿੱਚ 16-ਬਿਟ ਡਾਟਾ ਟ੍ਰਾਂਸਫਰ ਕਰ ਸਕਦੀ ਹੈ। ਇਹ ਪ੍ਰਣਾਲੀ ਕੰਪਿਊਟਰ ਆਰਕੀਟੈਕਚਰ ਵਿੱਚ ਵਾਇਰਿੰਗ ਦੀ ਜਟਿਲਤਾ ਨੂੰ ਘਟਾਉਣ ਅਤੇ ਰਜਿਸਟਰਾਂ, ALU, ਅਤੇ ਮੈਮੋਰੀ ਵਰਗੀਆਂ ਇਕਾਈਆਂ ਵਿਚਕਾਰ ਸੁਚੱਜੀ ਡਾਟਾ ਚਲਾਏਗਈ ਯਕੀਨੀ ਬਣਾਉਣ ਲਈ ਵਰਤੀ ਜਾਂਦੀ ਹੈ।

ਮੁੱਖ ਹਿੱਸੇ:

- **ਰਜਿਸਟਰ (Registers):** ਜਿਵੇਂ AR (Address Register), DR (Data Register), AC (Accumulator), PC (Program Counter)। ਇਹ ਤਤਕਾਲੀ ਡਾਟਾ ਜਾਂ ਐਡਰੈੱਸ ਰੱਖਦੇ ਹਨ।
- **ਬੱਸ (Bus):** 16 ਸਮਾਂਤਰ ਲਾਈਨਾਂ ਦਾ ਸਮੂਹ ਜੋ ਡਾਟਾ ਟ੍ਰਾਂਸਫਰ ਲਈ ਵਰਤਿਆ ਜਾਂਦਾ ਹੈ।
- **ਕੰਟਰੋਲ ਲਾਈਨਾਂ (Control Lines):** ਇਹ ਨਿਰਧਾਰਤ ਕਰਦੀਆਂ ਹਨ ਕਿ ਕਿਹੜਾ ਰਜਿਸਟਰ ਬੱਸ ਉੱਤੇ ਡਾਟਾ ਭੇਜਦਾ ਹੈ ਅਤੇ ਕਿਹੜਾ ਰਜਿਸਟਰ ਡਾਟਾ ਪ੍ਰਾਪਤ ਕਰਦਾ ਹੈ।

ਡਾਟਾ ਟ੍ਰਾਂਸਫਰ ਦੀ ਪ੍ਰਕਿਰਿਆ:

1. **Control Unit** DR ਰਜਿਸਟਰ ਤੋਂ ਡਾਟਾ ਬੱਸ ਉੱਤੇ ਭੇਜਣ ਲਈ **DRout** ਸਿਗਨਲ ਐਕਟੀਵੇਟ ਕਰਦੀ ਹੈ।
2. ਬੱਸ 16-ਬਿਟ ਡਾਟਾ ਨੂੰ ਇੱਕ ਤੋਂ ਦੂਜੇ ਰਜਿਸਟਰ ਤੱਕ ਲਿਜਾਂਦੀ ਹੈ।
3. ਡਾਟਾ ਨੂੰ ਪ੍ਰਾਪਤ ਕਰਨ ਲਈ **ACin** ਕੰਟਰੋਲ ਸਿਗਨਲ ਐਕਟੀਵੇਟ ਕੀਤਾ ਜਾਂਦਾ ਹੈ, ਜਿਸ ਨਾਲ ਡਾਟਾ Accumulator ਵਿੱਚ ਲੋਡ ਹੋ ਜਾਂਦਾ ਹੈ।
4. ਇਹ ਸਾਰੇ ਸਟੈਪ **ਕਲੈਕ ਸਿਗਨਲ** ਨਾਲ ਸਿੰਕਰੋਨਾਈਜ਼ ਹੋਕੇ ਕੀਤੇ ਜਾਂਦੇ ਹਨ, ਤਾਂ ਜੋ ਟਾਈਮਿੰਗ ਠੀਕ ਰਹੇ।

ਉਦਾਹਰਨ:

ਜੇਕਰ DR ਤੋਂ AC ਵਿੱਚ ਡਾਟਾ ਮੂਵ ਕਰਨਾ ਹੋਵੇ ਤਾਂ:

👉 **DRout → Bus → ACin**

ਇਸ ਤਰੀਕੇ ਨਾਲ ਡਾਟਾ ਸਧਾਰਨ, ਕਦਮ ਦਰ ਕਦਮ ਅਤੇ ਨਿਰਧਾਰਤ ਕੰਟਰੋਲ ਨਾਲ CPU ਦੇ ਅੰਦਰ ਸੁਚੱਜੀ ਤਰ੍ਹਾਂ ਮੂਵ ਕੀਤਾ ਜਾਂਦਾ ਹੈ।

Q. Discuss any five memory reference instructions with the help of example. (Nov 22), (Nov 24),(Nov 23)

Ans. **Memory Reference Instructions** are instructions that involve memory access to either fetch, store, or manipulate data. These are commonly found in basic computer architectures like the **Basic Computer Model**, and typically use the memory to hold operands or results.

Here are five commonly used memory reference instructions:

1. **LDA (Load Accumulator):** Loads data from a memory location into the accumulator.
   Example: LDA 500 → Loads the content of memory location 500 into AC (Accumulator).
2. **STA (Store Accumulator):** Stores the content of the accumulator into a specified memory location.
   Example: STA 450 → Stores the content of AC into memory location 450.
3. **ADD (Add to Accumulator):** Adds the content of a memory location to the accumulator.
   Example: ADD 300 → AC = AC + M[300].
4. **SUB (Subtract from Accumulator):** Subtracts the content of a memory location from the accumulator.
   Example: SUB 250 → AC = AC - M[250].
5. **BUN (Branch Unconditionally):** Changes the Program Counter (PC) to the specified address.
   Example: BUN 600 → PC = 600; control jumps to address 600.

ਮੈਮੋਰੀ ਰੈਫਰੈਂਸ ਨਿਰਦੇਸ਼ (Memory Reference Instructions): ਮੈਮੋਰੀ ਰੈਫਰੈਂਸ ਨਿਰਦੇਸ਼ ਉਹ ਹੁੰਦੇ ਹਨ ਜੋ ਮੈਮੋਰੀ ਨੂੰ ਐਕਸੈੱਸ ਕਰਕੇ ਡਾਟਾ ਨੂੰ **ਫੈਚ**, **ਸਟੋਰ**, ਜਾਂ **ਮੈਨਿਪੂਲੇਟ** ਕਰਦੇ ਹਨ। ਇਹ ਨਿਰਦੇਸ਼ ਆਮ ਤੌਰ 'ਤੇ **ਬੇਸਿਕ ਕੰਪਿਊਟਰ ਮਾਡਲ** ਵਿੱਚ ਵਰਤੇ ਜਾਂਦੇ ਹਨ, ਜਿੱਥੇ ਓਪਰੇਂਡ ਜਾਂ ਨਤੀਜਾ ਮੈਮੋਰੀ ਵਿੱਚ ਰੱਖਿਆ ਜਾਂਦਾ ਹੈ।

ਪੰਜ ਆਮ ਮੈਮੋਰੀ ਰੈਫਰੈਂਸ ਨਿਰਦੇਸ਼:

1. **LDA (ਲੋਡ ਐਕਿਊਮੂਲੇਟਰ):** ਮੈਮੋਰੀ ਤੋਂ ਡਾਟਾ ਐਕਿਊਮੂਲੇਟਰ ਵਿੱਚ ਲੋਡ ਕਰਦਾ ਹੈ।
   ਉਦਾਹਰਨ: LDA 500 → ਮੈਮੋਰੀ ਅਡਰੈੱਸ **500** ਦੀ ਵੈਲਯੂ **AC** ਵਿੱਚ ਲੋਡ ਕਰਦਾ ਹੈ।
2. **STA (ਸਟੋਰ ਐਕਿਊਮੂਲੇਟਰ):** ਐਕਿਊਮੂਲੇਟਰ ਦੀ ਵੈਲਯੂ ਨਿਰਧਾਰਤ ਮੈਮੋਰੀ ਅਡਰੈੱਸ 'ਤੇ ਸਟੋਰ ਕਰਦਾ ਹੈ।
   ਉਦਾਹਰਨ: STA 450 → **AC** ਦੀ ਵੈਲਯੂ **450** ਅਡਰੈੱਸ 'ਤੇ ਲਿਖੀ ਜਾਂਦੀ ਹੈ।
3. **ADD (ਐਡ ਟੂ ਐਕਿਊਮੂਲੇਟਰ):** ਮੈਮੋਰੀ ਦੀ ਵੈਲਯੂ **AC** ਵਿੱਚ ਜੋੜੀ ਜਾਂਦੀ ਹੈ।
   ਉਦਾਹਰਨ: ADD 300 → **AC = AC + M[300]**
4. **SUB (ਸਬਟਰੈਕਟ ਫ੍ਰੋਮ ਐਕਿਊਮੂਲੇਟਰ):** ਮੈਮੋਰੀ ਦੀ ਵੈਲਯੂ **AC** ਵਿੱਚੋਂ ਘਟਾਈ ਜਾਂਦੀ ਹੈ।
   ਉਦਾਹਰਨ: SUB 250 → **AC = AC - M[250]**
5. **BUN (ਬ੍ਰਾਂਚ ਅਨਕੰਡੀਸ਼ਨਲੀ):** ਪਰੋਗ੍ਰਾਮ ਕਾਊਂਟਰ (PC) ਨੂੰ ਨਵੀਂ ਅਡਰੈੱਸ 'ਤੇ ਲੈ ਜਾਂਦਾ ਹੈ।
   ਉਦਾਹਰਨ: BUN 600 → **PC = 600**; ਕੰਟਰੋਲ **600** ਅਡਰੈੱਸ 'ਤੇ ਜੰਪ ਕਰ ਜਾਂਦਾ ਹੈ।

ਇਹ ਨਿਰਦੇਸ਼ ਮੂਲ ਕੰਪਿਊਟਰ ਆਰਕੀਟੈਕਚਰ ਵਿੱਚ **ਡਾਟਾ ਐਕਸੈੱਸ**, **ਕੰਟਰੋਲ ਫਲੋ**, ਅਤੇ **ਐਰਥਮੈਟਿਕ/ਲੋਜਿਕ ਪ੍ਰੋਸੈਸ਼ਨਾਂ** ਲਈ ਬਹੁਤ ਹੀ ਜ਼ਰੂਰੀ ਹੁੰਦੇ ਹਨ।

Q. Write a detailed note on RISC and CISC Architecture. (Nov 23), (Nov 24)

Ans. **RISC (Reduced Instruction Set Computer)** and **CISC (Complex Instruction Set Computer)** are two contrasting CPU design philosophies.

RISC Architecture: RISC focuses on a **small set of simple instructions** that can be executed in **one clock cycle**. It emphasizes **hardware simplicity and speed**. All instructions are of **fixed length**, and most operations are performed on **registers**, minimizing memory access. RISC uses techniques like **pipelining** to increase instruction throughput.

Features:
- Fewer instructions
- Simple addressing modes
- Faster execution
- Requires more instructions per program

Example: ARM, MIPS, SPARC

CISC Architecture: CISC provides a **large set of complex instructions**, some of which can execute multi-step operations like memory-to-memory data transfer or arithmetic directly. It reduces the number of instructions per program but can take **multiple clock cycles** to execute each instruction.

Features:
- Many specialized instructions
- Complex addressing modes
- Slower but fewer instructions needed
- Hardware is more complex

Example: Intel x86, VAX

Comparison:
- RISC: Faster, simpler, requires more code
- CISC: Slower per instruction but more efficient in code size

Modern CPUs often combine both approaches, using **RISC cores** with **CISC compatibility** for optimal performance.

RISC (Reduced Instruction Set Computer) ਅਤੇ CISC (Complex Instruction Set Computer)

RISC ਅਤੇ CISC ਦੋ ਵੱਖ-ਵੱਖ CPU ਡਿਜ਼ਾਇਨ ਦਰਸ਼ਨ ਹਨ, ਜੋ ਪ੍ਰੋਸੈਸਰ ਦੇ ਨਿਰਮਾਣ ਤੇ ਕੰਮ ਕਰਨ ਦੇ ਤਰੀਕਿਆਂ ਨੂੰ ਦਰਸਾਉਂਦੇ ਹਨ।

✅ RISC ਆਰਕੀਟੈਕਚਰ (ਸਿਮਪਲ ਅਤੇ ਤੇਜ਼)

RISC ਥੋੜ੍ਹੀਆਂ ਅਤੇ ਆਸਾਨ ਨਿਰਦੇਸ਼ਾਂ 'ਤੇ ਧਿਆਨ ਦੇਂਦਾ ਹੈ ਜੋ ਇੱਕ ਕਲਾਕ ਚੱਕਰ ਵਿੱਚ ਚਲ ਸਕਦੇ ਹਨ। ਇਹ ਆਮ ਤੌਰ ਤੇ **ਰਜਿਸਟਰ-ਅਧਾਰਤ** ਹੁੰਦੇ ਹਨ ਅਤੇ **ਮੈਮੋਰੀ ਐਕਸੈੱਸ ਘੱਟ** ਕਰਦੇ ਹਨ।

ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ:
- ਘੱਟ ਨਿਰਦੇਸ਼ (ਫੀਚਰ)
- ਆਸਾਨ ਐਡਰੈਸਿੰਗ ਮੋਡ
- ਤੇਜ਼ ਕਾਰਜਕੁਸ਼ਲਤਾ
- ਇੱਕ ਕੰਮ ਕਰਨ ਲਈ ਜ਼ਿਆਦਾ ਨਿਰਦੇਸ਼ ਲੋੜੀਂਦੇ ਹਨ

ਉਦਾਹਰਨ: ARM, MIPS, SPARC

✅ CISC ਆਰਕੀਟੈਕਚਰ (ਜਟਿਲ ਪਰ ਘੱਟ ਨਿਰਦੇਸ਼)

CISC ਵਿੱਚ ਬਹੁਤ ਸਾਰੇ ਜਟਿਲ ਨਿਰਦੇਸ਼ ਹੁੰਦੇ ਹਨ ਜੋ ਇੱਕ ਸਮੇਂ ਵਿੱਚ ਕਈ ਸਟੈਪ ਦੀ ਕਾਰਵਾਈ ਕਰ ਸਕਦੇ ਹਨ, ਜਿਵੇਂ ਕਿ **memory-to-memory** ਡਾਟਾ ਟ੍ਰਾਂਸਫਰ।

ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ:
- ਬਹੁਤ ਸਾਰੇ ਖਾਸ ਨਿਰਦੇਸ਼
- ਜਟਿਲ ਐਡਰੈਸਿੰਗ ਮੋਡ
- ਨਿਰਦੇਸ਼ ਘੱਟ, ਪਰ ਹਰੇਕ ਦੀ ਕਾਰਵਾਈ ਹੌਲੀ
- ਹਾਰਡਵੇਅਰ ਜ਼ਿਆਦਾ ਜਟਿਲ

ਉਦਾਹਰਨ: Intel x86, VAX

🔄 ਤੁਲਨਾ (Comparison):

ਵਿਸ਼ਾ	RISC	CISC
ਨਿਰਦੇਸ਼ਾਂ ਦੀ ਗਿਣਤੀ	ਘੱਟ ਅਤੇ ਆਸਾਨ	ਬਹੁਤ ਜ਼ਿਆਦਾ ਅਤੇ ਜਟਿਲ
ਕਾਰਗੁਜ਼ਾਰੀ	ਤੇਜ਼ ਅਤੇ ਕਮ ਸਾਈਕਲ	ਹੌਲੀ, ਪਰ ਨਿਰਦੇਸ਼ ਘੱਟ
ਕੋਡ ਦਾ ਆਕਾਰ	ਵੱਡਾ	ਛੋਟਾ
ਹਾਰਡਵੇਅਰ	ਆਸਾਨ	ਜਟਿਲ

🔄 ਮਾਡਰਨ CPU ਡਿਜ਼ਾਇਨ:

ਆਧੁਨਿਕ ਪ੍ਰੋਸੈਸਰ **RISC ਅਤੇ CISC** ਦੋਵਾਂ ਦੀਆਂ ਵਿਸ਼ੇਸ਼ਤਾਵਾਂ ਨੂੰ ਜੋੜਦੇ ਹਨ। ਉਦਾਹਰਨ ਵਜੋਂ, **Intel CPU** ਅੰਦਰ RISC ਕੋਰ ਵਰਤੇ ਜਾਂਦੇ ਹਨ ਪਰ **CISC-ਸਮਰਥਨ ਵਾਲੀ ਇੰਸਟ੍ਰਕਸ਼ਨ ਸੈੱਟ** ਵੀ ਹੁੰਦੀ ਹੈ।