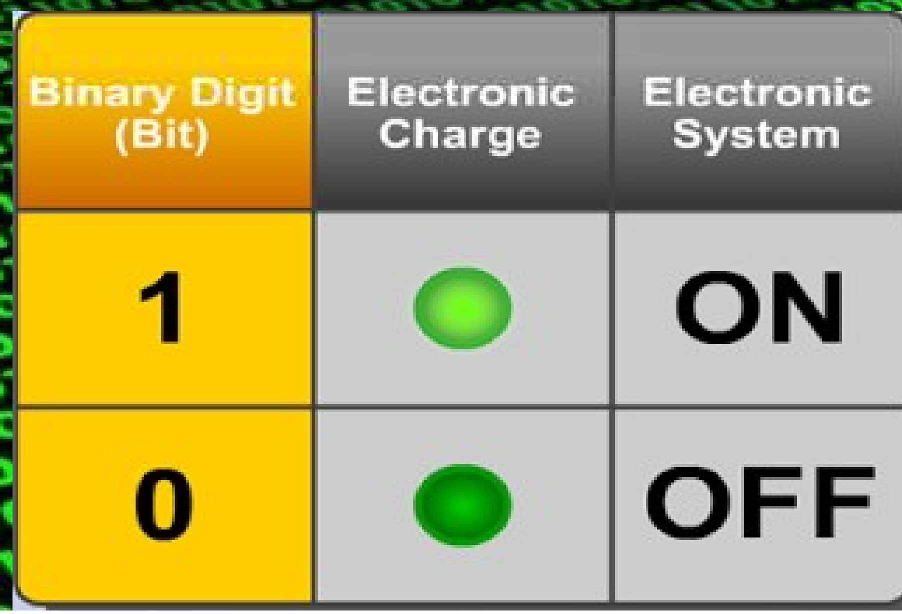
# CHAPTER-07

## (Data Representation and Number System)

## डेटा प्रतिनिधित्व और संख्या प्रणाली



# Number Systems
# संख्या पद्धतियाँ या संख्या प्रणालियाँ

**(Decimal Number System) - 0,1,2,3,4,5,6,7,8,9**

2 .द्विआधारी संख्या पद्धति

**(Binary Number System) - 0,1**

3 .आक्टल संख्या पद्धति

**(Octal Number System) - 0,1,2,3,4,5,6,7**

4 .हेक्साडेसिमल संख्या पद्धति

**(Hexadecimal Number System)**

**0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**

# दशमलव प्रणाली -Decimal Number System

- **दशमलव प्रणाली (Decimal Number System) में अंकों का आधार 10** है तथा इसमें प्रत्येक अंक का अलग स्थान है।
- उसे हम उस अंक का स्थानीय मान कहते हैं।

0,1,2,3,4,5,6,7,8,9

# DATA REPRESENTATION

- डेटा प्रतिनिधित्व कम्प्यूटर के अन्दर डेटा को संग्रह करने की विधि है।
- कम्प्यूटर विभिन्न प्रकार के डेटा का संग्रह करता है; जैसे– संख्या, टेक्सट, ग्राफिक्स, ध्वनि इत्यादि।
- ये सारे डेटा हमें अलग–अलग लगते हैं परन्तु कम्प्यूटर में ये सारे डेटा को एक ही स्वरूप **(Type)** में **store**किया जाता है।
- यह स्वरूप है 0 तथा 1 के क्रम में (अर्थात् डिजिटल रूप में) कम्प्यूटर ये सारी संग्रहित जानकारी का प्रतिनिधित्व करने के लिए सांख्यिक कोड**(numeric code)** का उपयोग करते हैं।

- सामान्यतः हम लोग 0 से 9 का उपयोग कर कोई भी संख्या लिखते हैं; इसका आधार 10 **(Decimal Number System)** होता है।
- परन्तु कम्प्यूटर में कोई भी पूरी संख्या 0 और 1 के द्वारा प्रतिनिनिधत्व किया जाता है।

कम्प्यूटर में प्रयोग होने वाली संख्या पद्धतियाँ

# कम्प्यूटर में प्रयोग होने वाली संख्या पद्धतियाँ

1 .द्विआधारी संख्या पद्धति

**(Binary Number System) - 0,1**

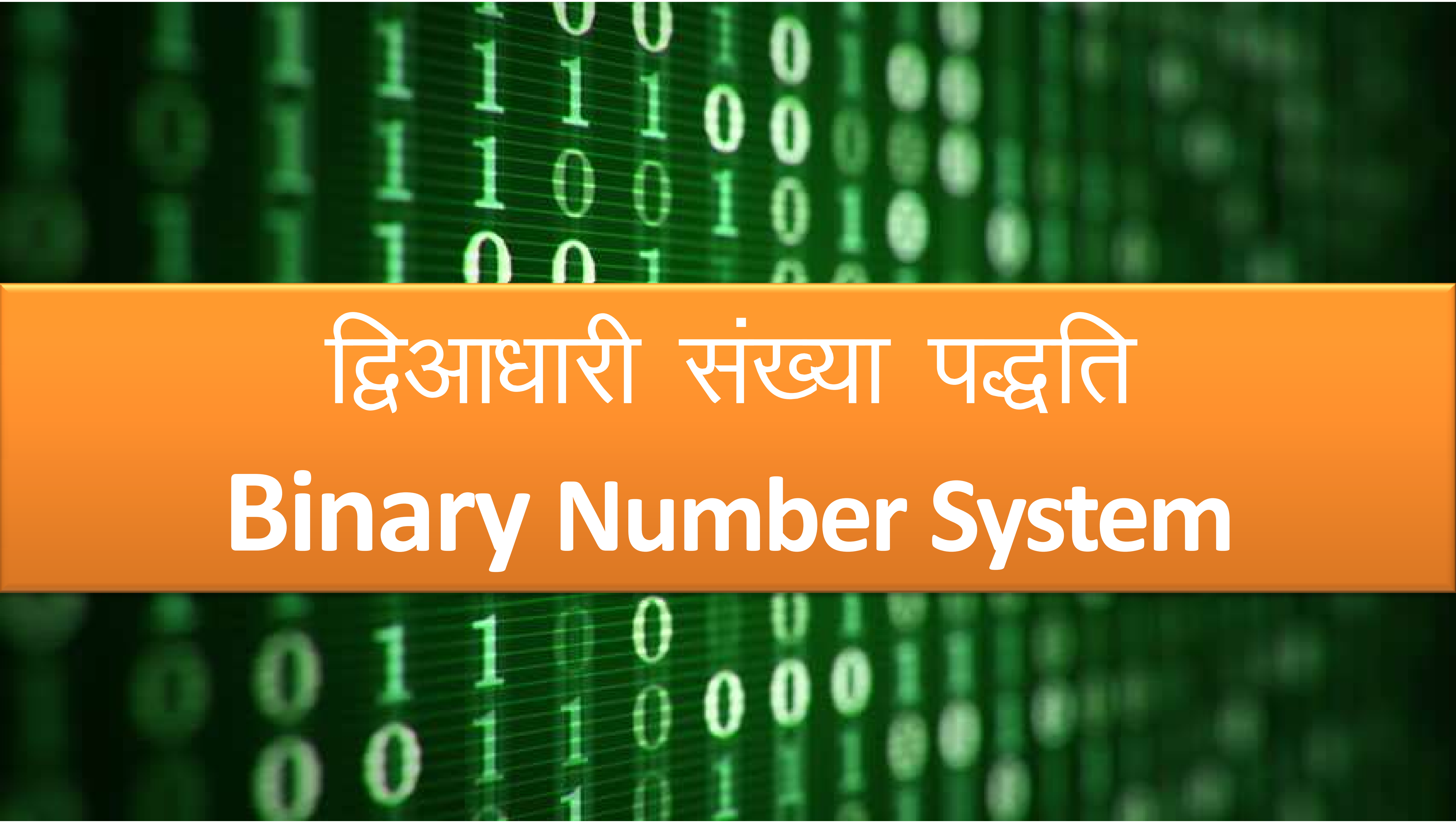
2 .आक्टल संख्या पद्धति

**(Octal Number System) - 0,1,2,3,4,5,6,7**

3 .हेक्साडेसिमल संख्या पद्धति

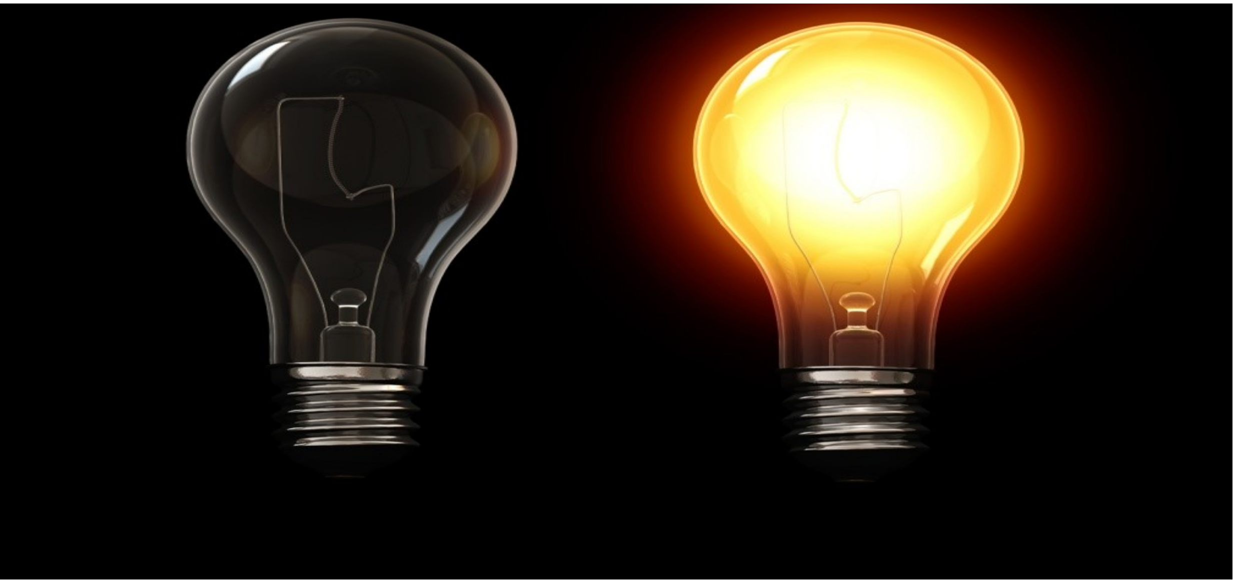
**(Hexadecimal Number System)**

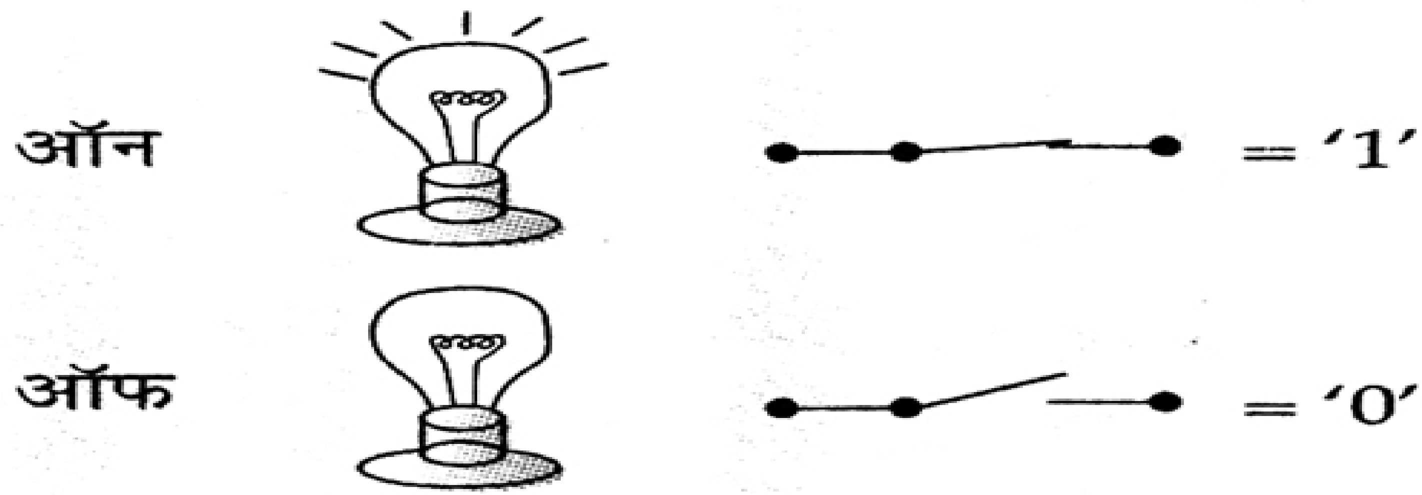
**0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**


द्विआधारी संख्या पद्धति
Binary Number System

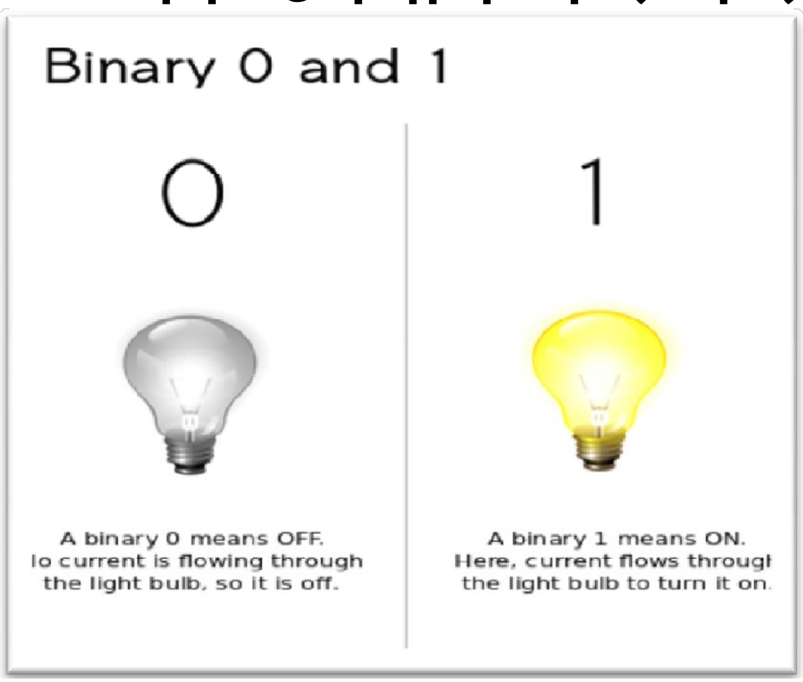
# द्विआधारी संख्या पद्धति – Binary Number System

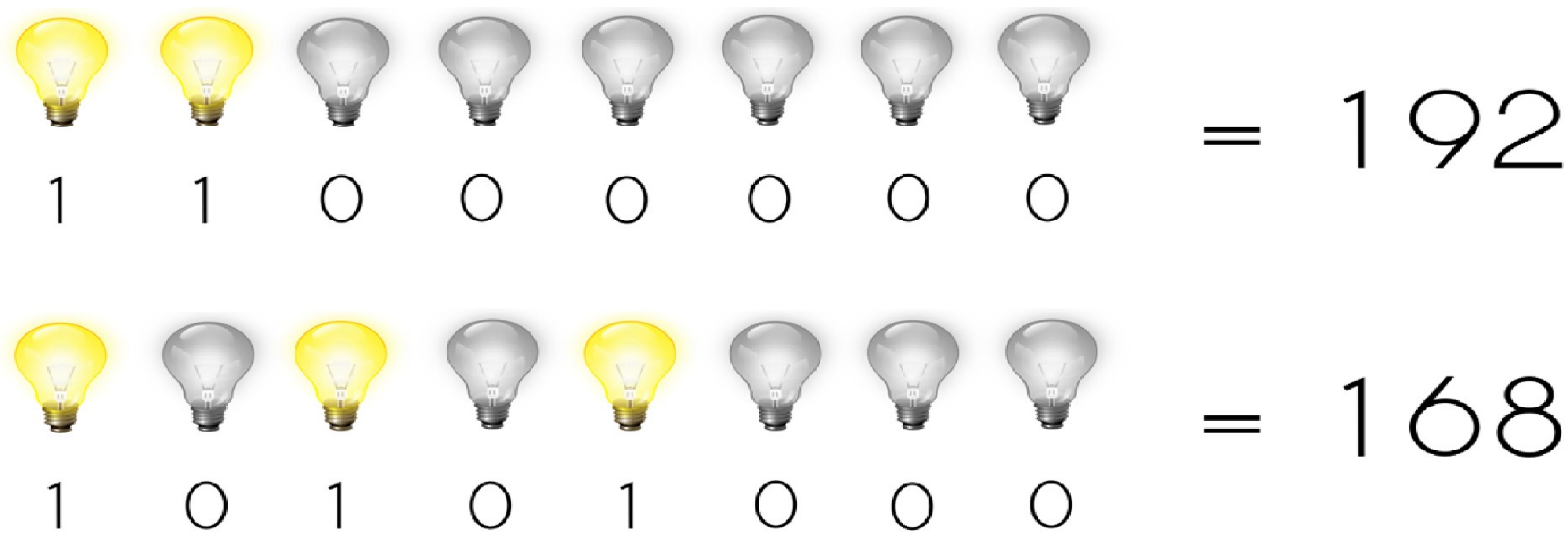
- कम्प्यूटर गणना के लिए केवल दो संख्या 0 तथा 1 का उपयोग करता है।
- जो कम्प्यूटर के परिपथ में प्रवाहित हो रही विद्युत धारा की 'ऑन' तथा 'ऑफ' दो स्थिति का संकेतात्मक मान है।
- जब परिपथ में धारा प्रवाहित हो रही है तब उसका संकेतात्मक मान 1 होता है अन्यथा 0 होता है।
- इन 0 तथा 1 का बाइनरी संख्या कहते हैं एक बाइनरी संख्या को बिट **(Bit)** कहते हैं।
- इस संख्या पद्धति का उपयोग कम्प्यूटर द्वारा डेटा संग्रह या गणना करने के लिए होता है।






• चित्र की सहायता से अब हम देख सकते हैं कि कम्प्यूटर इन दोनों संख्या का उपयोग कर कैसे कार्य करता है।



# Conversion from Binary to Decimal

# द्विआधारी का दशमलव में रूपान्तरण

# Conversion from Binary to Decimal

- द्विआधारी को दशमलव में रूपान्तरित करने के लिए द्विआधारी संख्या के अंकों को उसके स्थानीय मान से गुणा कर प्राप्त सभी गुणनफल को जोड़ दिया जाता है।
- उदाहरण– $100011_2$ को दशमलव में बदलें।

| | | | | | | |
|---|---|---|---|---|---|---|
| • द्विआधारी संख्या | 1 | 0 | 0 | 0 | 1 | 1 |
| • स्थान | 5 | 4 | 3 | 2 | 1 | 0 |
| • मान | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

1 0 0 0 1 1 – **Binary**

= 32 16 8 4 2 1

$1 \times 1 = 1$

$1 \times 2 = 2$

$0 \times 4 = 0$

$0 \times 8 = 0$

$0 \times 16 = 0$

$1 \times 32 = 32$

$35$

अत: $100011_2$ $35_{10}$

# Conversion from Decimal to Binary
# दशमलव का द्विआधारी में रूपान्तरण

# Conversion from Decimal to Binary

- दशमलव को द्विआधारी संख्या में रूपान्तरित करने के लिए दशमलव संख्या को दो (2) से विभाजित करते हैं,
- फिर इसके भागफल को दो से पुनः तब तक विभाजित करते हैं, जब तक कि भागफल एक (1) या शून्य (0) न हो जाये।
- उदाहरण– (1) $35_{10}$ द्विआधारी पद्धति में बदलें।

$\therefore \quad 35_{10} = 100011_2$

| | | |
|---|---|---|
| 2 | 35 | शेष |
| 2 | 17 | 1 |
| 2 | 8 | 1 |
| 2 | 4 | 0 |
| 2 | 2 | 0 |
| | 1 | 0 |

↑ नीचे से पढ़ते या लिखते हैं।

- दशमलव के बाद वाले संख्या को द्विआधारी में बदलने के लिए 2 से लगातार गुणा करते हैं
- तथा इसके पूर्णांक **(Integer)** भाग को अलग लिखते जाते हैं।
- उदाहरण–
- $0.6875_{10}$ को **(Binary)** में बदलें।

ऊपर से पढ़ते हैं

| | | |
|---|---|---|
| | | 0.6875 |
| | | × 2 |
| 1 | ← | .3750 |
| | | × 2 |
| 0 | ← | .7500 |
| | | × 2 |
| 1 | ← | .5000 |
| | | × 2 |
| 1 | ← | .0000 |

$$\therefore \quad 0.6875_{10} = 0.1011_{2}$$

# Octal Number System
# ऑक्टल संख्या पद्धति

# Octal Number System- ऑक्टल संख्या पद्धति

- ऑक्टल संख्या पद्धति में 0 से 7 अर्थात् 8 अंकों के द्वारा संख्या का प्रतिनिधित्व किया जाता है।
- दशमलव संख्या पद्धति में हर स्थानीय मान 10 के गुणक में बढ़ता है,
- तथा द्विआधारी संख्या पद्धति में 2 के गुणक में बढ़ता है,
- परन्तु ऑक्टल संख्या पद्धति में 8 के गुणक में बढ़ता है। इनमें आधार 8 होता है।

# Conversion of Decimal to Octal
# दशमलव का ऑक्टल में रूपान्तरण

# Conversion of Decimal to Octal

- दशमलव को ऑक्टल में रूपान्तरित करने के लिए दशमलव को 8 के द्वारा तब तक विभाजित किया जाता है जबतक कि भागफल 8 से कम न हो जाये।
- उदाहरण– 385 को ऑक्टल में बदलें।

| | | |
|---|---|---|
| 8 | 385 | शेष |
| 8 | 48 | 1 |
| | 6 | 0 |

↑ नीचे से पढ़ते हैं।

# Conversion from Octal to Decimal

# ऑक्टल का दशमलव में रूपान्तरण

# Conversion from Octal to Decimal

- ऑक्टल को दशमलव में रूपान्तरित करने के लिए ऑक्टल संख्या के अंको को उसके स्थानीय मान से गुणा कर फिर सभी को जोड़ दिया जाता है।

उदाहरण– $175_8$ को दशमलव में बदलें।

$$175_8 = (1 \times 8^2) \quad (7 \times 8^1) \quad (5 \times 8^0)$$

$$64 + 56 + 5 = 125_{10}$$

अतः $175_8 = 125_{10}$

# Conversion from Octal to Binary
# ऑक्टल का द्विआधारी में रूपान्तरण

# Conversion from Octal to Binary

**Octal को Binary में रूपान्तरित करने की दो विधियाँ हैं।**

1 .ऑक्टल संख्या को द्विआधारी में रूपान्तरित करने के लिए पहले उसे दशमलव में रूपान्तरित करते हैं फिर दशमलव को बाइनरी में रूपान्तरित करते हैं।

2. ऑक्टल संख्या को द्विआधारी संख्या के तीन अक्षरों के समूह द्वारा निरूपित किया जाता है।

**1.** ऑक्टल संख्या को द्विआधारी में रूपान्तरित करने के लिए पहले उसे दशमलव में रूपान्तरित करते हैं फिर दशमलव को बाइनरी में रूपान्तरित करते हैं।

$725_8$ को **Binary** में बदलें।

$$725_8 \quad (7 \times 8^2) + (2 \times 8^1) + (5 \times 8^0)$$

$$448 + 16 + 5 = 469_{10}$$

फिर, $469_{10}$ को **Binary** में बदलते हैं।

| | | |
|---|---|---|
| 2 | 469 | शेष |
| 2 | 234 | 1 |
| 2 | 117 | 0 |
| 2 | 58 | 1 |
| 2 | 29 | 0 |
| 2 | 14 | 1 |
| 2 | 7 | 0 |
| 2 | 3 | 1 |
| | 1 | 1 |

नीचे

अत: $725_8 \quad 469_{10} \quad 111010101_2$

2. **Octal** संख्या को **Binary** संख्या के तीन **Digits** के समूह द्वारा **Convert** किया जाता है।

जैसे– **Octal** संख्या 1 को **Binary** संख्या में 001 लिखते हैं।

- अतः ऑक्टल संख्या को **Binary** में रूपान्तरित करने के लिए उसे उनके तीन अंको के द्विआधारी समूह से परिवर्तित कर देते हैं।

| Octal | Binary | Octal | Binary |
|---|---|---|---|
| 0 | 000 | 4 | 100 |
| 1 | 001 | 5 | 101 |
| 2 | 010 | 6 | 110 |
| 3 | 011 | 7 | 111 |

# Conversion from Binary to Octal
# द्विआधारी का ऑक्टल में रूपान्तरण

# Conversion from Binary to Octal

- द्विआधारी को ऑक्टल में रूपान्तरित करने की दो विधियाँ हैं।
- प्रथम विधि मे द्विआधारी संख्या को दायें से आरम्भ कर तीन–तीन द्विआधारी अंकों का समूह बना कर उसका ऑक्टल मान प्राप्त किया जाता है।

उदाहरण–

$$111010101_2 = \underbrace{111}\ \underbrace{010}\ \underbrace{101}$$

$$= 7 \quad 2 \quad 5 = 725_8$$

दूसरी विधि से द्विआधारी को ऑक्टल में रूपान्तरित करने के लिए द्विआधारी संख्या को पहले दशमलव में रूपान्तरित करते हैं, फिर उसे ऑक्टल में रूपान्तरित किया जाता है।

उदाहरण– $111010101_2$ को ऑक्टल में बदलें।

1 1 1 0 1 0 1 0 1

$(1 \times 2^8) + (1 \times 2^7) + (1 \times 2^6) + (0 \times 2^5) + (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$

$256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1$

$= 469_{10}$ फिर इसे ऑक्टल में बदलते हैं।

| 8 | 469 | शेष |
|---|---|---|
| 8 | 58 | 5 |
| | 7 | 2 |

↑ नीचे से

अतः $111010101_2 = 725_8$

# हेक्साडेसीमल संख्या पद्धति

# Hexadecimal Number System

# Hexadecimal Number System

- हेक्साडेसिमल संख्या पद्धति में 0 से 15 अर्थात् 16 अंकों का उपयोग किया जाता है।
- इनमें 0 से 9 के अंक तथा 10 से 15 को **A to F** अल्फाबेट से निरूपित किया जाता है।
- हेक्साडेसिमल संख्या को बाइनरी के चार अंकों के समूह से दर्शाया जाता है; जैसे– हेक्साडेसिमल 1 संख्या का बाइनरी मान 0001 होता है।
- इसमें आधार 16 होता है, तथा हेक्साडेसिमल संख्या पद्धति में स्थानीय मान 16 के गुणक में बढ़ता है।

| हेक्साडेसिमल | दशमलव | चार अंकों का द्विआधारी समूह |
| --- | --- | --- |
| 0 | 0 | 0000 |
| 1 | 1 | 0001 |
| 2 | 2 | 0010 |
| 3 | 3 | 0011 |
| 4 | 4 | 0100 |
| 5 | 5 | 0101 |
| 6 | 6 | 0110 |
| 7 | 7 | 0111 |

| | | |
|---|---|---|
| 8 | 8 | 1000 |
| 9 | 9 | 1001 |
| **A** | 10 | 1010 |
| **B** | 11 | 1011 |
| **C** | 12 | 1100 |
| **D** | 13 | 1101 |
| **E** | 14 | 1110 |
| **F** | 15 | 1111 |

# हेक्साडेसिमल का दशमलव में रूपान्तरण
# Conversion of Hexadecimal to Decimal

# Conversion of Hexadecimal to Decimal

1 .हेक्साडेसिमल को दशमलव में रूपान्तरित करने के लिए उसके अंकों को उसके स्थानीय मान से गुणा कर फिर सभी गुणनफल को जोड़ देते हैं।

उदाहरण– $10_{16}$ को दशमलव में बदलें।

$$10_{16} = 1 \times 16^1 + 0 \times 16^0$$

$$16 + 0 = 16_{10}$$

# Conversion of Decimal to Hexadecimal

दशमलव को हेक्साडेसिमल में रूपान्तरित करने के लिए दशमलव आधारी संख्या को 16 के द्वारा तब तक विभाजित करते हैं जब तक कि भागफल 16 से कम न हो जाये।

उदाहरण– $382_{10}$ को हेक्साडेसिमल में बदलें।

| | | |
|---|---|---|
| 16 | 382 | शेष |
| 16 | 23 | 14 = E |
| | 1 | 7 |

**अतः $382_{10} = 17E_{16}$**

हेक्साडेसिमल में 14 को **E** लिखते हैं।

# Conversion of Hexadecimal to Binary

हेक्साडेसिमल को द्विआधारी में रूपांतरित करने की दो विधियां हैं–

1. हेक्साडेसिमल को द्विआधारी में रूपान्तरित करने के लिए पहले दशमलव में बदलकर फिर उसे द्विआधारी में रूपान्तरित करते हैं।

**उदाहरण– $B6A_{16}$ को द्विआधारी में बदलें।**

$$B6A_{16} = (B \times 16^{2}) + (6 \times 16^{1}) + (A \times 16^{0})$$

$$(11 \times 256) + (6 \times 16) + (10 \times 1)$$

$$2816 + 96 + 10$$

# अब इसको द्विआधारी में रूपान्तरित करते हैं।

| | | |
|---|---|---|
| 2 | 2922 | शेष |
| 2 | 1461 | 0 |
| 2 | 730 | 1 |
| 2 | 365 | 0 |
| 2 | 182 | 1 |
| 2 | 91 | 0 |
| 2 | 45 | 1 |
| 2 | 22 | 1 |
| 2 | 11 | 0 |
| 2 | 5 | 1 |
| 2 | 2 | 1 |
| | 1 | 0 |

नीचे से लिखते हैं।

**$B6A_{16}$ $101101101010_2$**

1. हेक्साडेसिमल को द्विआधारी में रूपान्तरित करने के लिए उसके अंकों को द्विआधारी चार अंकों के समूह से बदल देते हैं।

उदाहरण— **$B6A_{16}$** को द्विआधारी में बदलें।

$$B6A_{16} = \underbrace{1011}_{B} \quad \underbrace{0110}_{6} \quad \underbrace{1010}_{A}$$

- **Conversion of Binary to Hexadecimal**

द्विआधारी को हेक्साडेसिमल में रूपान्तरित करने के लिए उसे दायें से चार–चार अंकों के समूह में विभाजित कर उसका हेक्साडेसिमल मान से रूपान्तरित कर देते हैं।

उदाहरण— $\underbrace{0001} \quad \underbrace{0110} \quad \underbrace{1111} = 16F_{16}$ ज में बदलें।

# BINARY ADDITION & SUBTRACTION

# BINARY ADDITION

- **Binary Addition** भी दशमलव जोड़ के तरह ही होता है परन्तु इसमें केवल दो अंकों 0 तथा 1 का उपयोग होता हैं।
- सबसे दायें वाले कॉलम से जोड़ना आरम्भ करते हैं।
- द्विआधारी जोड़ करने के लिए चार नियम हैं–

(1) 1 + 1 = 0 (हासिल

(2) 1 + 0 = 1

(3) 0 + 1 = 1

(4) 0 + 0 = 0

उदाहरण– $1010_2 + 0111_2$ को जोड़ें।

$$\begin{array}{r} 1010 \\ 0111 \\ \hline 10001_2 \end{array}$$

अत: $1010_2 + 0111_2 = 10001_2$

# BINARY SUBTRACTION

- यह भी केवल दो अंक 0 तथा 1 का घटाव है।
- सबसे दायें वाले कॉलम से घटाना आरम्भ करते हैं।
- द्विआधारी घटाव करने के लिए चार नियम निम्नलिखित हैं–
- (1) 0 – 0 = 0
- (2) 1 – 1 = 0
- (3) 1 – 0 = 1
- (4) 0 – 1 = 1 ( निकटतम बायीं तरफ से 1 उधार लेते हैं। )

उदाहरण– $11010_2 - 01001_2$ को घटायें।

$$\begin{array}{r} 11010 \\ 01001 \\ \hline 10011 \end{array}$$

अतः $11010_2 - 01001_2 = 10011_2$

# BINARY MULTIPLICATION

- यह सामान्य दशमलव गुणा की ही तरह किया जाता है।
- द्विआधारी अंकों का ही गुणनफल निकाला जाता है या द्विआधारी अंक को दशमलव में बदल कर इसका गुणनफल निकालते हैं और फिर उसे द्विआधारी में बदला जाता है।
- इ ा करें

**प्रथम विधि से,**

$$
\begin{array}{r}
1101_2 \\
\times\ 1100_2 \\
\hline
0000 \\
0000\phantom{0} \\
1101\phantom{00} \\
1101\phantom{000} \\
\hline
10011100_2
\end{array}
$$

**दूसरी विधि से,**

$$1101_2 = 13_{10}$$
$$1100_2 = 12_{10}$$

$$
\begin{array}{r}
13 \\
\times\ 12 \\
\hline
26 \\
13\phantom{0} \\
\hline
156
\end{array}
$$

$$156_{10} = 10011100_2$$

# BINARY DIVISON

- यह भी दशमलव भाग के तरह ही किया जाता है।
- द्विआधारी भाग के नियम निम्नलिखित हैं–

(i) $0 \div 1 = 0$

(ii) $1 \div 1 = 1$

उदाहरण– $100001_2 \div 110_2$

110 ) 100001 (101.1
110
× 1001
110
× 110
110
000

अत: $100001_2 \div 110_2 = 101.1_2$

**Subtraction**

$0 - 0 = 0$

$1 - 1 = 0$

$1 - 0 = 1$

# Binary Memory
# बाइनरी मेमोरी

# Binary Memory

- कम्प्यूटर में डेटा को संग्रहित करने के यंत्र को मेमोरी कहते हैं। मेमोरी में डेटा बाइनरी रूप (0 तथा 1) में संग्रहित होता है।
- बिट **(Bit)** : यह कम्प्यूटर मेमोरी की सबसे छोटी इकाई है। एक बिट का केवल एक ही मान होता है, 0 या 1
- निब्बल **(Nibble)** : यह चार बिट का समूह है।
- बाइट **(Byte)** : यह **8 Bits** का समूह हैं।
- डेटा को बाइट में मापा जाता है।

# Binary Memory

- किसी भी अक्षर, चिह्न या स्पेस के लिए कम से कम एक बाइट (Byte) स्थान की आवश्यकता होती है।
- शब्द (Word) : यह Bits का समूह है। कम्प्यूटर जिसे एक इकाई के तरह व्यवहार करता है।
- शब्द की लंबाई विभिन्न मषीनों के अनुसार बदलता रहता है।

# Memory Measurement

- 4 Bit = 1 Nibble
- 8 Bit = 1 Byte
- 1024 Byte = 1 KB (Kilobyte)
- 1024KB = 1 MB (Megabyte)
- 1024MB = 1GB (Gigabyte)
- 1024GB = 1 TB(Terabyte)
- 1024TB = 1 PB (Petabyte)
- 1024PB = 1 EB (Exabyte)
- 1024 EB = 1 ZB (Zettabyte)
- 1024 ZB = 1 YB (Yottabyte)

# COMPUTER CODES

# Computer Codes

- कम्प्यूटर में करेक्टर जैसे अल्फाबेट, संख्या या कोई चिह्न बिट्स के समूहों में स्टोर किया जाता है, जो किसी विशेष द्विआधारी पैटर्न पर आधारित होता है, ये विशेष द्विआधारी पैटर्न भिन्न–भिन्न प्रकार का होता है जिसे कम्प्यूटर कोड्स कहते हैं।

कम्प्यूटर कोडींग सिस्टम विभिन्न प्रकार के होते हैं–

1. बी सी डी **(BCD-Binary Coded Decimal) :**

# BCD-Binary Coded Decimal Codes

- इसे पैकेट दशमलव **(Packet Decimal)** भी कहते हैं।
- इसमें दशमलव संख्या के प्रत्येक अंक को द्विआधारी के चार अंकों के समूह से परिवर्तित कर दिया जाता है।

## BCD TABLE

| दशमलव | BCD |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |

| दशमलव | BCD |
|---|---|
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |

- आस्की कोड (ASCII) : **American Standard Code for Information Interchange** का संक्षिप्त नाम है। **ASCII** सिस्टम में एक कैरेक्टर को **7 BITS** से **Convert** करते हैं व 256 कैरेक्टर

क

| | | | |
|---|---|---|---|
| 65 | 41 | 01000001 | A |
| 66 | 42 | 01000010 | B |
| 67 | 43 | 01000011 | C |
| 68 | 44 | 01000100 | D |
| 69 | 45 | 01000101 | E |
| 70 | 46 | 01000110 | F |
| 71 | 47 | 01000111 | G |
| 72 | 48 | 01001000 | H |
| 73 | 49 | 01001001 | I |
| 74 | 4A | 01001010 | J |
| 75 | 4B | 01001011 | K |
| 76 | 4C | 01001100 | L |
| 77 | 4D | 01001101 | M |

| | | | |
|---|---|---|---|
| 78 | 4E | 01001110 | N |
| 79 | 4F | 01001111 | O |
| 80 | 50 | 01010000 | P |
| 81 | 51 | 01010001 | Q |
| 82 | 52 | 01010010 | R |
| 83 | 53 | 01010011 | S |
| 84 | 54 | 01010100 | T |
| 85 | 55 | 01010101 | U |
| 86 | 56 | 01010110 | V |
| 87 | 57 | 01010111 | W |
| 88 | 58 | 01011000 | X |
| 89 | 59 | 01011001 | Y |
| 90 | 5A | 01011010 | Z |

| | | | |
|---|---|---|---|
| 97 | 61 | 01100001 | a |
| 98 | 62 | 01100010 | b |
| 99 | 63 | 01100011 | c |
| 100 | 64 | 01100100 | d |
| 101 | 65 | 01100101 | e |
| 102 | 66 | 01100110 | f |
| 103 | 67 | 01100111 | g |
| 104 | 68 | 01101000 | h |
| 105 | 69 | 01101001 | i |
| 106 | 6A | 01101010 | j |
| 107 | 6B | 01101011 | k |
| 108 | 6C | 01101100 | l |
| 109 | 6D | 01101101 | m |

# एब्सेडिक कोड (EBCDIC) : Extended Binary Coded Decimal Interchange Code:

यह एक्सटेन्डेड बाइनरी कोडेड डेसीमल इंटरचेंज कोड का संक्षिप्त नाम है। इस सिस्टम में एक कैरेक्टर को 8 बिटस से निरूपित करते हैं। आस्की तथा एब्सेडि कोडिंग सिस्टम हैं।

| Special characters | EBCDIC |
|---|---|
| < | 01001011 |
| ( | 01001100 |
| + | 01001101 |
| / | 01001110 |
| & | 01010000 |
| : | 01111011 |
| # | 01111011 |
| @ | 01111100 |
| ‘ | 01111101 |
| = | 01111110 |
| “ | 01111111 |
| , | 01101011 |
| % | 01101100 |
| - | 01101101 |
| > | 01101110 |

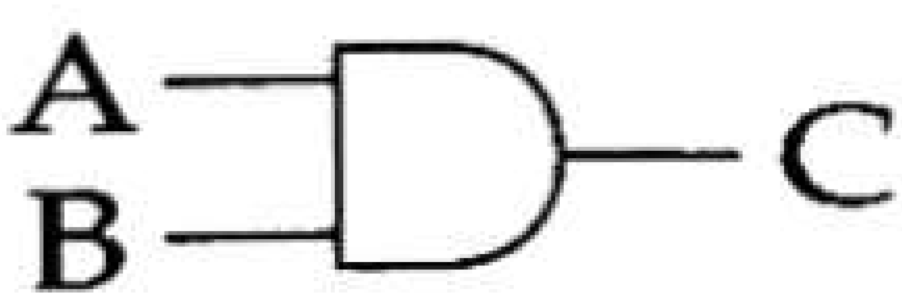
Contd..

| Alphabetic | EBCDIC |
|---|---|
| A | 11000001 |
| B | 11000010 |
| C | 11000011 |
| D | 11000100 |
| E | 11000101 |
| F | 11000110 |
| G | 11000111 |
| H | 11001000 |
| I | 11001001 |
| J | 11010001 |
| K | 11010010 |
| L | 11010011 |
| M | 11010100 |
| N | 11010101 |
| O | 11010110 |
| P | 11010111 |

# LOGIC GATES

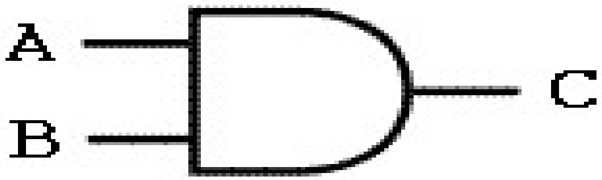
# LOGIC - GATES

- सभी डिजिटल सिस्टम केवल तीन आधार भूत गेट से बने होते हैं।
- ये तीनों क्रमषः **AND** गेट, **OR** गेट तथा **NOT** गेट हैं।
- गेट एक इलेक्ट्रॉनिक सर्किट है जहाँ 1 या 1 से अधिक इनपुट डाल कर



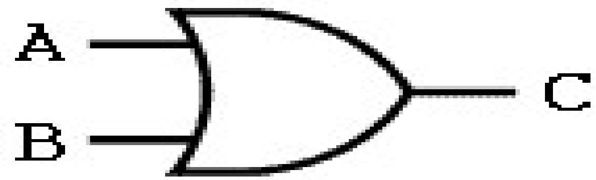

कते हैं।

AND



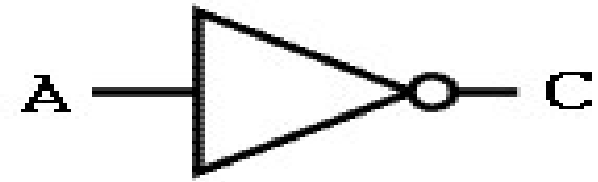

| Inputs | | Output |
|---|---|---|
| **A** | **B** | **C** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

OR




| Inputs | | Output |
|---|---|---|
| **A** | **B** | **C** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

NOT

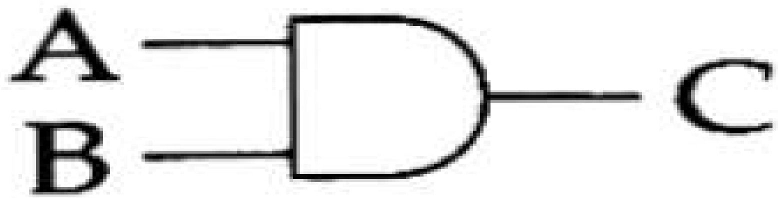



| Input | Output |
|---|---|
| **A** | **C** |
| 0 | 1 |
| 1 | 0 |

| A | B | A AND B | A OR B | NOT A |
|---|---|---|---|---|
| False | False | False | False | True |
| False | True | False | True | True |
| True | False | False | True | False |
| True | True | True | True | False |

# AND GATE

- **AND** गेट – **AND** गेट के सर्किट को इस प्रकार दर्शाया जाता है।

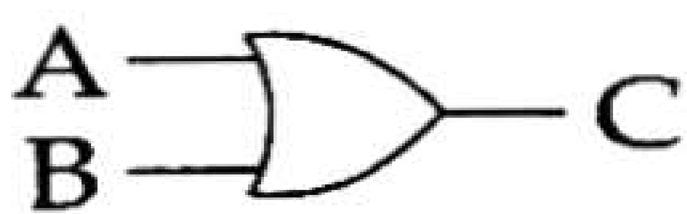



| स्विच A | स्विच B | स्विच C |
|---|---|---|
| ऑफ (0) | ऑफ (0) | ऑफ (0) |
| ऑफ (0) | ऑन (1) | ऑफ (0) |
| ऑन (1) | ऑफ (0) | ऑफ (0) |
| ऑन (1) | ऑन (1) | ऑन (1) |

अर्थात **C = A x B**

# OR GATE

**OR** गेट—**OR** गेट के सर्किट को इस प्रकार दर्शाया जाता है।

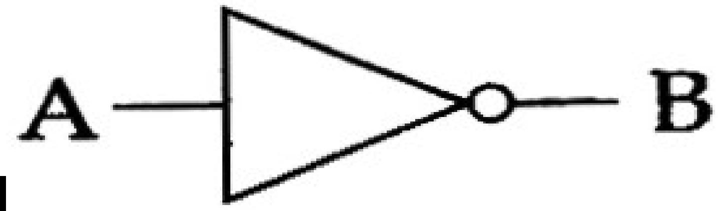



| स्विच A | स्विच B | स्विच C |
| --- | --- | --- |
| ऑफ (0) | ऑफ (0) | ऑफ (0) |
| ऑफ (0) | ऑन (1) | ऑन (1) |
| ऑन (1) | ऑफ (0) | ऑन (1) |
| ऑन (1) | ऑन (1) | ऑन (1) |

अर्थात **C = A + B**

# AND GATES

**NOT** गेट – **NOT** गेट के सर्किट को इस प्रकार दर्शाया जाता है।

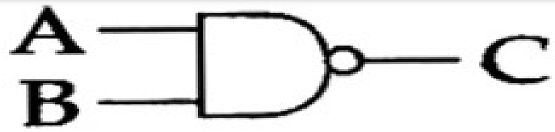



| स्विच A | स्विच B |
|---|---|
| ऑफ (0) | ऑन (1) |
| ऑन (1) | ऑफ (0) |

अर्थात **$B = \bar{A}$**

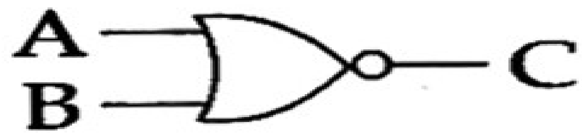
## इन तीनों गेटों की सहायता से अन्य गेट भी बनाये जाते हैं–

NAND गेट




अर्थात् $C = \overline{A \cdot B}$

NOR गेट



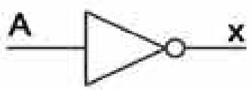
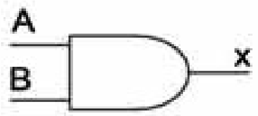
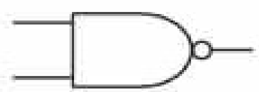
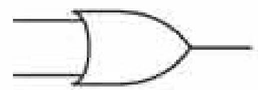
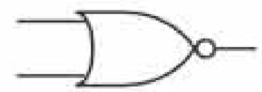
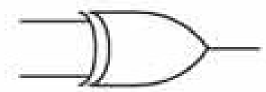

अर्थात् $C = \overline{A + B}$

XOR गेट

$C = A \oplus B$

- **NAND** गेट तथा **NOR** गेट यूनिवर्सल गेट कहलाते हैं, क्योंकि इनके द्वारा कोई भी डिजिटल सर्किट का निर्माण किया जा सकता है।

# Logic Gates

<table>
<tr><th>Name</th><th>NOT</th><th>AND</th><th>NAND</th><th>OR</th><th>NOR</th><th>XOR</th><th>XNOR</th></tr>
<tr><td>Alg. Expr.</td><td>$\overline{A}$</td><td>$AB$</td><td>$\overline{AB}$</td><td>$A + B$</td><td>$\overline{A + B}$</td><td>$A \oplus B$</td><td>$\overline{A \oplus B}$</td></tr>
<tr><td>Symbol</td><td>
</td><td>
</td><td></td><td></td><td></td><td></td><td> </td></tr>
<tr><td>Truth Table</td>
<td><table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table></td>
<td><table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td>
<td><table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td>
<td><table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td>
<td><table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td>
<td><table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table></td>
<td><table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table></td>
</tr>
</table>

# THANKS