# 2

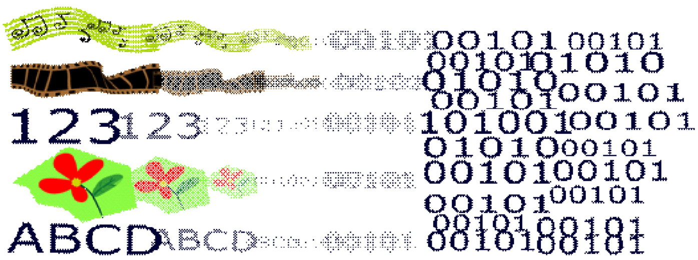
# ഡാറ്റയുടെ പ്രതിനിധാനവും ബൂളിയൻ ബീജഗണിതവും

## പ്രധാന ആശയങ്ങൾ

- **സംഖ്യാ സമ്പ്രദായം**
  - ദശസംഖ്യ (Decimal Number), ദ്വയസംഖ്യ (Binary Number), അഷ്ടസംഖ്യ (Octal Number), ഷോഡശസംഖ്യ (Hexadecimal Number) ഹെക്സാഡെസിമൽ സംഖ്യ (Hexadecimal Number)
- **സംഖ്യാ പരിവർത്തനങ്ങൾ**
- **ഡാറ്റ പ്രതിനിധീകരണം**
  - പൂർണസംഖ്യകളുടെയും ഫ്ളോട്ടിങ് പോയിന്റ് സംഖ്യകളുടെയും പ്രതിനിധാനം
  - അക്ഷരങ്ങളുടെ പ്രതിനിധാനം ആസ്കി, എബ്സിഡിക്, ഇസ്കി, യൂണികോഡ്
  - ശബ്ദം, ചിത്രം, വീഡിയോ ഡാറ്റ പ്രതിനിധാനം
- **ബൈനറി ഗണിതം**
  - സങ്കലനം, വ്യവകലനം, പൂരകങ്ങൾ
- **ബൂളിയൻ ബീജഗണിതം ഒരു ആമുഖം**
  - ലോജിക് ഓപറേറ്ററുകളും ഗേറ്റുകളും
- **ബൂളിയൻ അടിസ്ഥാന അംഗീകൃത തത്വം**
- **ലളിതമായ ബൂളിയൻ പദപ്രയോഗങ്ങളും സർക്യൂട്ടുകളുടെ രൂപകൽപനയും**
- **യൂനിവേഴ്സൽ ഗേറ്റുകൾ**

R8L9A6

വ്യത്യസ്തതരം ഡാറ്റകൾ കൈകാര്യം ചെയ്യാൻ ശേഷിയുള്ള ഒരു ഉപകരണമാണ് കമ്പ്യൂട്ടർ. ഡാറ്റ പ്രോസസ്സിംഗ് നടത്തുന്നതിന് വേണ്ടി സംഖ്യകൾ, അക്ഷരങ്ങൾ, ചിത്രങ്ങൾ, വീഡിയോകൾ, ശബ്ദങ്ങൾ തുടങ്ങിയ ഡാറ്റരൂപങ്ങൾ നമ്മൾ കമ്പ്യൂട്ടറിലേക്ക് നൽകാറുണ്ട്. വൈദ്യുതിയുടെ രണ്ട് അവസ്ഥകളായ ഓൺ (ON), ഓഫ് (OFF) എന്നിവയെ അടിസ്ഥാനമാക്കി പ്രവർത്തിക്കുന്ന ഒരു ഇലക്ട്രോണിക് ഉപകരണമാണ് കമ്പ്യൂട്ടർ എന്നത് നമുക്ക് അറിയാം. അതുപോലെ എല്ലാ ഇലക്ട്രോണിക് സർക്യൂട്ടുകൾക്കും തുറന്നിരിക്കുന്നതും അടഞ്ഞിരിക്കുന്നതും ആയ രണ്ടവസ്ഥകളാണുള്ളത്. തുറന്നിരിക്കുന്ന അവസ്ഥയെ സൂചിപ്പിക്കാനായി ഓഫ് (OFF) അല്ലെങ്കിൽ പൂജ്യവും അടഞ്ഞിരിക്കുന്ന അവസ്ഥയെ സൂചിപ്പിക്കാനായി ഓൺ (ON) അല്ലെങ്കിൽ ഒന്നും ഉപയോഗിക്കുന്നു. ഈ രണ്ടവസ്ഥയിലുള്ള പ്രവർത്തനത്തെ ബൈനറി ഓപ്പറേഷൻ (Binary Operation) എന്ന് പറയുന്നു. അതുകൊണ്ട് കമ്പ്യൂട്ടറിലേക്ക് നൽകുന്ന ഡാറ്റയും ബൈനറി രൂപത്തിലായിരിക്കണം. സംഖ്യകൾ, അക്ഷരങ്ങൾ, ചിത്രങ്ങൾ, വീഡിയോകൾ, ശബ്ദങ്ങൾ എന്നിവയെ പ്രതിനിധാനം ചെയ്യുന്നതിനുള്ള വിവിധ രീതികളാണ് ഈ അധ്യായത്തിൽ ചർച്ച ചെയ്യുന്നത്.



*ചിത്രം 2.1 ഡാറ്റയുടെ ബാഹ്യവും ആന്തരികവുമായ രൂപങ്ങൾ*

ഒരു കമ്പ്യൂട്ടറിൽ ഡാറ്റയെ ആന്തരികമായി പ്രതിനിധീകരിക്കുന്നതിന് ഉപയോഗിക്കുന്ന രീതിയാണ് ഡാറ്റയുടെ പ്രതിനിധാനം (Data representation). സംഖ്യകളുടെ പ്രതിനിധാനം ചർച്ച ചെയ്യുന്നതിന് മുമ്പായി സംഖ്യാ സമ്പ്രദായം (Number System) എന്താണെന്നു നമുക്ക് നോക്കാം.

## 2.1 സംഖ്യാ സമ്പ്രദായം (Number systems)

എണ്ണുന്നതിനും, അടയാളപ്പെടുത്തുന്നതിനും, അളക്കുന്നതിനും ഉള്ള ഗണിതശാസ്ത്രപരമായ ഉപാധിയാണ് സംഖ്യ. ചിട്ടയോടെ സംഖ്യകളെ പ്രതിനിധാനം ചെയ്യുന്ന രീതിയാണ് സംഖ്യാന സമ്പ്രദായം. പത്ത് അക്കങ്ങൾ ഉപയോഗിച്ച് കൊണ്ടുള്ള ദശസംഖ്യാ സമ്പ്രദായമാണ് (Decimal Number System) നമ്മൾ നിത്യജീവിതത്തിൽ ഉപയോഗിച്ച് വരുന്നത്. 289 എന്ന സംഖ്യയെ ഇരുന്നൂറ്റി എൺപത്തി ഒൻപത് എന്നാണ് ഉച്ചരിക്കുന്നത്. ഇതിൽ 2, 8, 9 എന്നീ അക്കങ്ങൾ അടങ്ങിയിട്ടുണ്ട്. അതുപോലെ മറ്റ് സംഖ്യാ സമ്പ്രദായങ്ങളും നിലവിലുണ്ട്. ഓരോന്നിനും അതിന്റേതായ ചിഹ്നങ്ങളും രീതികളുമാണ് അവയിലെ സംഖ്യ രൂപകൽപന ചെയ്യുന്നതിന് ഉപയോഗിക്കുന്നത്. ഓരോ സംഖ്യാ സമ്പ്രദായത്തിനും തനതായ ആധാരം ഉണ്ട്. ഇത് ആ സംഖ്യാ സമ്പ്രദായത്തിലെ ചിഹ്നങ്ങളുടെ എണ്ണത്തെ ആശ്രയിച്ചിരിക്കുന്നു. ഒരു സംഖ്യാ സമ്പ്രദായത്തിൽ ഉപയോഗിക്കുന്ന അക്കങ്ങളുടെ അല്ലെങ്കിൽ ചിഹ്നങ്ങളുടെ എണ്ണത്തെ ആ സംഖ്യാ സമ്പ്രദായത്തിലെ ആധാരം (Base) അല്ലെങ്കിൽ മൂലസംഖ്യ (Radix ) എന്ന് പറയുന്നു.

ചില സംഖ്യാ സമ്പ്രദായങ്ങളെ കുറിച്ച് നമുക്ക് ചർച്ച ചെയ്യാം.

### 2.1.1 ദശസംഖ്യാ സമ്പ്രദായം (Decimal number system)

ദശസംഖ്യാ സമ്പ്രദായത്തിൽ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 തുടങ്ങിയ പത്ത് അക്കങ്ങളാണ് സംഖ്യാ രൂപീകരണത്തിന് ഉപയോഗിക്കുന്നത്. ദശസംഖ്യാ സമ്പ്രദായത്തിൽ പത്ത് അക്കങ്ങൾ ഉപയോഗിക്കുന്നതുകൊണ്ട് അതിന്റെ ആധാരം (Base) 10 ആകുന്നു. അതുകൊണ്ടു ദശസംഖ്യാ സമ്പ്രദായത്തെ 10 ആധാരമാക്കിയ സംഖ്യാ സമ്പ്രദായം എന്നു കൂടി വിളിക്കുന്നു.

743, 347 എന്നീ രണ്ട് ദശസംഖ്യകൾ പരിഗണിക്കുക.

743 = ഏഴ് നൂറുകൾ + നാലു പത്തുകൾ + മൂന്ന് ഒന്നുകൾ ( $7\times10^2 + 4\times10^1 + 3\times10^0$)

347 = മൂന്ന് നൂറുകൾ + നാലു പത്തുകൾ + ഏഴ് ഒന്നുകൾ ( $3\times10^2 + 4 \times10^1 + 7\times10^0$)

ഇവിടെ ഒന്നാമത്തെ സംഖ്യയായ 743 ൽ 7 ന്റെ സ്ഥാനവില (Weight) $10^2 = 100$ ആകുന്നു. എന്നാൽ രണ്ടാമത്തെ സംഖ്യയായ 347 ൽ 7 ന്റെ സ്ഥാനവില $10^0 = 1$ ആകുന്നു. ഒരു സംഖ്യയുടെ സ്ഥാനവില അതിന്റെ ആപേക്ഷിക സ്ഥാനത്തെ ആശ്രയിച്ചിരിക്കുന്നു. അത്തരം സംഖ്യാ സമ്പ്രദായത്തെ സ്ഥാനീയ സംഖ്യാ സമ്പ്രദായം (Positional Number System) എന്നു പറയുന്നു. എല്ലാ സ്ഥാനീയ സംഖ്യാ സമ്പ്രദായത്തിനും ഒരു ആധാരം (Base) ഉണ്ടായിരിക്കും. ഒരു അക്കത്തിന്റെ സ്ഥാനവില ആധാരത്തിന്റെ ചില കൃത്യങ്കം (Power) ആയിരിക്കും. ഓരോ ദശസംഖ്യ അക്കത്തിന്റെ സ്ഥാന വില 10 ന്റെ കൃത്യങ്കം ആയിരിക്കും ($10^0$, $10^1$, $10^2$, ……). 5876 എന്ന ദശസംഖ്യ പരിഗണിക്കുക. ഈ സംഖ്യ താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ വിപുലീകരിച്ചു എഴുതാം.

| സ്ഥാനവില (Weight) | $10^3$ | $10^2$ | $10^1$ | $10^0$ |
|---|---|---|---|---|
| ദശസംഖ്യ | 5 | 8 | 7 | 6 |

$= 5\times10^3 + 8\times10^2 + 7\times10^1 + 6\times10^0$

$= 5\times1000 + 8\times100 + 7\times10 + 6\times1$

$= 5000 + 800 + 70 + 6$

$= 5876$

മുകളിൽ കൊടുത്തിരിക്കുന്ന ഉദാഹരണത്തിൽ 5 എന്ന അക്കത്തിന് ഏറ്റവും കൂടിയ സ്ഥാനവിലയായ $10^3$=1000 ഉം 6 എന്ന അക്കത്തന് ഏറ്റവും കുറഞ്ഞ സ്ഥാനവിലയായ $10^0$=1 ഉം ആണ്. ഏറ്റവും കൂടിയ സ്ഥാനവിലയുള്ള അക്കത്തെ കൂടിയ പ്രബലതയുള്ള അക്കം (Most Significant Digit - MSD) എന്നും ഏറ്റവും കുറഞ്ഞ സ്ഥാനവിലയുള്ള അക്കത്തെ കുറഞ്ഞ പ്രബലതയുള്ള അക്കം (Least Significant Digit - LSD) എന്നും വിളിക്കുന്നു. അതിനാൽ മുകളിൽ കൊടുത്തിരിക്കുന്ന സംഖ്യയിൽ MSD എന്നത് 5 ഉം LSD എന്നത് 6 ഉം ആകുന്നു.

***ഒരു സംഖ്യയുടെ ഏറ്റവും ഇടതു വശത്തുള്ള അക്കം MSD ഉം ഏറ്റവും വലതു വശത്തുള്ള അക്കം LSD ഉം ആകുന്നു.***

ദശാംശ സംഖ്യകളിൽ ദശാംശ ബിന്ദുവിന് വലുത് ഭാഗത്തുള്ള സംഖ്യകളുടെ സ്ഥാനവില 10 ന്റെ നെഗറ്റിവ് കൃത്യങ്കങ്ങൾ ആണ് ($10^{-1}$, $10^{-2}$, $10^{-3}$, .......). 249.367 എന്ന സംഖ്യ ഉദാഹരണമായി എടുക്കാം.

| സ്ഥാനവില (Weight) | $10^2$ | $10^1$ | $10^0$ | $10^{-1}$ | $10^{-2}$ | $10^{-3}$ |
|---|---|---|---|---|---|---|
| ദശസംഖ്യ | 2 | 4 | 9 | 3 | 6 | 7 |
| | MSD | | (.) | | | LSD |

$= 2\times10^2 + 4\times10^1 + 9\times10^0 + 3\times10^{-1} + 6\times10^{-2} + 7\times10^{-3}$

$= 2\times100 + 4\times10 + 9\times1 + 3\times0.1 + 6\times0.01 + 7\times0.001$

$= 200 + 40 + 9 + 0.3 + 0.06 + 0.007$

$= 249.367$

ഇതുവരെ 10 അക്കങ്ങൾ ഉപയോഗിച്ചുകൊണ്ടുള്ള ഒരു സംഖ്യാ സമ്പ്രദായത്തെക്കുറിച്ചാണ് നമ്മൾ ചർച്ചചെയ്തത്. ഇനി നമുക്ക് വ്യത്യസ്ത ആധാരങ്ങളിലുള്ള മറ്റ് സംഖ്യാ സമ്പ്രദായങ്ങൾ രൂപീകരിക്കുന്നതെങ്ങനെ എന്ന് പഠിക്കാം.

### 2.1.2 ദ്വയസംഖ്യാ സമ്പ്രദായം (Binary number system)

ഒരു സംഖ്യ രൂപീകരിക്കാൻ 0, 1 എന്നീ രണ്ടക്കങ്ങൾ മാത്രം ഉപയോഗിക്കുന്ന സംഖ്യാ സമ്പ്രദായത്തെയാണ് ദ്വയസംഖ്യാ സമ്പ്രദായം (Binary Number System) എന്ന് പറയുന്നത്. ഇംഗ്ലീഷിൽ bi

(ബൈ) എന്ന വാക്ക് കൊണ്ട് അർഥമാക്കുന്നത് 2 എന്നാണ്. അതിനാൽ ഈ സംഖ്യാ സമ്പ്രദായത്തിന്റെ ആധാരം 2 ആകുന്നു. അതുകൊണ്ട് ഇതിനെ 2 ആധാരമാക്കിയുള്ള സംഖ്യാ സമ്പ്രദായം എന്ന് കൂടി വിളിക്കുന്നു. ഒരു സംഖ്യ ദ്വയസംഖ്യയാണെന്ന് സൂചിപ്പിക്കാൻ ആ സംഖ്യയോടു കൂടി 2 കീഴ്ക്കുറിപ്പ് (Subscript) ആയി ഉപയോഗിക്കുന്നു.

ഉദാഹരണങ്ങൾ $(1101)_2$, $(101010)_2$, $(1101.11)_2$

ഒരു ദ്വയസംഖ്യയിലെ ഓരോ അക്കത്തെയും ബിറ്റ് (bit) എന്നാണ് വിളിക്കുന്നത്. ഇംഗ്ലീഷിൽ bit ന്റെ പൂർണരൂപം binary digit എന്നാകുന്നു. ദ്വയസംഖ്യാന സമ്പ്രദായവും ഒരു സ്ഥാനീയ സംഖ്യാന സമ്പ്രദായമാണ്. ഓരോ ദ്വയസംഖ്യ അക്കത്തിന്റെയും സ്ഥാന വില 2 ന്റെ കൃത്യങ്കം (Power) ആണ്. $(1101)_2$ എന്ന ദ്വയസംഖ്യ ഉദാഹരണമായി പരിഗണിക്കുക. ഈ ദ്വയസംഖ്യ താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ വിപുലീകരിച്ച് എഴുതാം.

| സ്ഥാനവില (Weight) | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|
| Binary Number | 1 | 1 | 0 | 1 |
| | MSB | | | LSB |

$$\begin{aligned} &= 1\times2^3 + 1\times 2^2 + 0\times2^1 + 1\times2^0 \\ &= 1\times8 + 1\times4 + 0\times2 + 1\times1 \\ &= 8 + 4 + 0 + 1 \\ &= 13 \end{aligned}$$

ദ്വയസംഖ്യയിലെ ഏറ്റവും വലതുവശത്തു നിൽക്കുന്ന അക്കത്തിനെ കുറഞ്ഞ പ്രബലതയുള്ള ബിറ്റ് (Least Significant Bit - LSB) എന്നും ഏറ്റവും ഇടതുവശത്തു നിൽക്കുന്ന അക്കത്തിനെ കൂടുതൽ പ്രബലതയുള്ള ബിറ്റ് (Most Significant Bit - MSB) എന്നും വിളിക്കുന്നു.

1101 എന്ന ദ്വയസംഖ്യ 13 എന്ന ദശസംഖ്യയ്ക്ക് തുല്യമാണ്. എന്നാൽ 1101 എന്ന സംഖ്യ ദശസംഖ്യാ സമ്പ്രദായത്തിലും ഉണ്ട്. പക്ഷെ അതിനെ വ്യാഖ്യാനിക്കുന്നത് ആയിരത്തി ഒരുന്നൂറ്റി ഒന്ന് എന്നാണ്. ഈ ആശയക്കുഴപ്പം ഒഴിവാക്കുവാൻ വേണ്ടി ദശസംഖ്യാ സമ്പ്രദായം ഒഴികെയുള്ള എല്ലാ സംഖ്യാ സമ്പ്രദായങ്ങളിലും ആധാരം വ്യക്തമായി സൂചിപ്പിക്കണം. അതിന്റെ പൊതുവായ ഘടന താഴെ കൊടുത്തിരിക്കുന്നു.

**$(\text{സംഖ്യ})_{\text{ആധാരം}}$**

വ്യത്യസ്ത ആധാരത്തിലുള്ള സംഖ്യകളെ തരംതിരിച്ചറിയുവാൻ ഈ അടയാളപ്പെടുത്തൽ സഹായിക്കുന്നു. ഉദാഹരണമായി 1101 എന്ന ദ്വയസംഖ്യയെ $(1101)_2$ എന്ന് എഴുതുകയും അതിനെ 'ഒന്ന് ഒന്ന് പൂജ്യം ഒന്ന് ആധാരം രണ്ട് ' എന്ന് വായിക്കുകയും ചെയ്യണം. ഒരു സംഖ്യയ്ക്ക് ആധാരം നൽകിയിട്ടില്ലെങ്കിൽ അതിനെ ദശസംഖ്യയായി പരിഗണിക്കണം. അതായത് ദശസംഖ്യക്ക് ആധാരം സൂചിപ്പിക്കണമെന്ന് നിർബന്ധമില്ല.

ഭിന്നകമായ ഒരു ദ്വയസംഖ്യയുടെ അംശബിന്ദുവിന് (Binary Point) വലതുഭാഗത്തുള്ള അക്കങ്ങളുടെ സ്ഥാനവില 2 ന്റെ നെഗറ്റിവ് കൃത്യങ്കം ആയിരിക്കും. ($2^{-1}$, $2^{-2}$, $2^{-3}$,....). $(111.011)_2$ എന്ന സംഖ്യ ഉദാഹരണമായി എടുക്കാം.

| സ്ഥാനവില (Weight) | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ |
|---|---|---|---|---|---|---|
| ദ്വയസംഖ്യ (Binary Number) | 1 | 1 | 1 | 0 | 1 | 1 |

MSB (.) LSB

$$= 1\times2^2 + 1\times2^1 + 1\times2^0 + 0\times2^{-1} + 1\times2^{-2} + 1\times2^{-3}$$

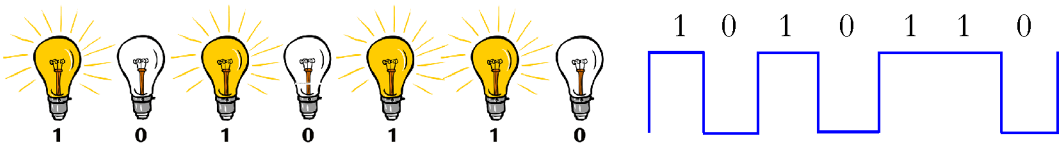
$$= 1\times4 + 1\times2 + 1\times1 + 0\times\frac{1}{2} + 1\times\frac{1}{4} + 1\times\frac{1}{8}$$

$$= 4 + 2 + 1 + 0 + 0.25 + 0.125$$

$$= 7.375$$

**കമ്പ്യൂട്ടറിൽ ദ്വയസംഖ്യയുടെ പ്രധാന്യം**

ദ്വയസംഖ്യാ സമ്പ്രദായം 1, 0 എന്നീ അക്കങ്ങൾ അടിസ്ഥാനമാക്കിയാണെന്നു നമ്മൾ കണ്ടല്ലോ. ചിത്രം 2.2 ൽ വൈദ്യുതിയുടെ ഓൺ (ON) ആയിരിക്കുന്ന അവസ്ഥ 1 കൊണ്ടും ഓഫ് (OFF) ആയിരിക്കുന്ന അവസ്ഥ 0 കൊണ്ടും സൂചിപ്പിക്കുന്നു. ഇക്കാരണത്താൽ, കമ്പ്യൂട്ടറിൽ ഡാറ്റയെ പ്രതിനിധാനം ചെയ്യുന്നതിന് അടിസ്ഥാന സംഖ്യാ സമ്പ്രദായമായി ദ്വയസംഖ്യാന സമ്പ്രദായം ഉപയോഗിക്കുന്നു.




*ചിത്രം 2.2 ON ഉം OFF ന്റെയും ഡിജിറ്റൽ രൂപത്തിന്റെ പ്രതിനിധാനം*

### 2.1.3 അഷ്ടസംഖ്യാ സമ്പ്രദായം (Octal number system)

എട്ട് അക്കങ്ങളായ 0, 1, 2, 3, 4, 5, 6, 7 എന്നിവ ഉപയോഗിച്ചുണ്ടാക്കുന്ന സംഖ്യാ സമ്പ്രദായത്തെ അഷ്ടസംഖ്യാ സമ്പ്രദായം (Octal Number System) എന്ന് പറയുന്നു. ഇംഗ്ലീഷിൽ Octa (ഒക്ട) എന്ന വാക്ക് കൊണ്ട് അർഥമാക്കുന്നത് 8 എന്നാണ്. അതിനാലാണ് ഈ സംഖ്യാ സമ്പ്രദായത്തെ ഒക്ടൽ സംഖ്യാ സമ്പ്രദായം എന്ന് പറയുന്നത്. ഈ സംഖ്യാ സമ്പ്രദായത്തിന്റെ ആധാരം 8 ആകുന്നു. അതുകൊണ്ട് ഇതിനെ 8 ആധാരമായ സംഖ്യാ സമ്പ്രദായം എന്നും വിളിക്കുന്നു. ഉദാഹരണമായി $(236)_8$ പരിഗണിക്കുക. ഓരോ ഒക്ടൽ അക്കത്തിന്റെയും സ്ഥാനവില 8 ന്റെ കൃത്യങ്കം (Power) ആയിരിക്കും ($8^0, 8^1, 8^2, 8^3$, …..). $(236)_8$ എന്ന സംഖ്യ താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ വിപുലീകരിച്ച് എഴുതാം.

| സ്ഥാനവില (Weight) | $8^2$ | $8^1$ | $8^0$ |
|---|---|---|---|
| ഒക്ടൽ സംഖ്യ | 2 | 3 | 6 |

$$= 2\times8^2 + 3\times8^1 + 6\times8^0$$

$$= 2\times64 + 3\times8 + 6\times1$$

$$= 128 + 24 + 6$$

$$= 158$$

ഭിന്നകമായ ഒരു അഷ്ടസംഖ്യയുടെ അംശബിന്ദുവിന് വലതുഭാഗത്തുള്ള അക്കങ്ങളുടെ സ്ഥാനവില 8 ന്റെ നെഗറ്റിവ് കൃത്യങ്കം ആയിരിക്കും ($8^{-1}, 8^{-2}, 8^{-3}, \ldots\ldots$). $(172.4)_8$ എന്ന സംഖ്യ ഉദാഹരണമായി എടുക്കാം.

| സ്ഥാനവില (Weight) | $8^2$ | $8^1$ | $8^0$ | $8^{-1}$ |
|---|---|---|---|---|
| ഒക്ടൽ സംഖ്യ | 1 | 7 | 2 | 4 |

$$\begin{aligned} &= 1\times8^2 + 7\times8^1 + 2\times8^0 + 4\times8^{-1} \\ &= 64 + 56 + 2 + 4\times\frac{1}{8} \\ &= 122 + 0.5 \\ &= 122.5 \end{aligned}$$

## 2.1.4 ഷോഡശ (ഹെക്സാഡെസിമൽ) സംഖ്യാ സമ്പ്രദായം (Hexadecimal number system)

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F എന്നീ 16 ചിഹ്നങ്ങൾ ഉപയോഗിച്ചുണ്ടാക്കുന്ന സംഖ്യാ സമ്പ്രദായത്തെ ഹെക്സാഡെസിമൽ സംഖ്യാ സമ്പ്രദായം എന്ന് പറയുന്നു. ഈ സംഖ്യാ സമ്പ്രദായത്തിൽ 16 ചിഹ്നങ്ങൾ ഉപയോഗിക്കുന്നതുകൊണ്ട് ഇതിന്റെ ആധാരം 16 ആകുന്നു. ആയതിനാൽ ഇതിനെ 16 ആധാരമായ സംഖ്യാ സമ്പ്രദായം എന്നും വിളിക്കുന്നു. ഈ സംഖ്യാ സമ്പ്രദായത്തിലെ A, B, C, D, E, F എന്നീ ചിഹ്നങ്ങൾ ഉപയോഗിക്കുന്നത് യഥാക്രമം ദശസംഖ്യാ സമ്പ്രദായത്തിലെ 10, 11, 12, 13, 14, 15 എന്ന സംഖ്യകളെ സൂചിപ്പിക്കുന്നതിനാണ്. ഹെക്സാഡെസിമൽ അക്കങ്ങളും അവയ്ക്ക് തുല്യമായ ദശസംഖ്യകളും ചുവടെ കാണിച്ചിരിക്കുന്നു.

| ഹെക്സാഡെസിമൽ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ദശസംഖ്യ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

ഉദാഹരണമായി $(12AF)_{16}$ എന്ന ഹെക്സാഡെസിമൽ സംഖ്യ പരിഗണിക്കുക. ഓരോ ഹെക്സാഡെസിമൽ അക്കത്തിന്റെയും സ്ഥാനവില 16 ന്റെ കൃത്യങ്കം (Power) ആയിരിക്കും ($16^0, 16^1, 16^2, 16^3, \ldots\ldots$). ഈ സംഖ്യയെ താഴെ കാണിച്ചിരിക്കുന്നത് പോലെ വിപുലീകരിച്ചു എഴുതാം.

| സ്ഥാനവില (Weight) | $16^3$ | $16^2$ | $16^1$ | $16^0$ |
|---|---|---|---|---|
| ഹെക്സാഡെസിമൽ അക്കം | 1 | 2 | A | F |

$$\begin{aligned} &= 1\times16^3 + 2\times16^2 + 10\times16^1 + 15\times16^0 \\ &= 1\times4096 + 2\times256 + 10\times16 + 15\times1 \\ &= 4096 + 512 + 160 + 15 \\ &= 4783 \end{aligned}$$

ഭിന്നകമായ ഒരു ഹെക്സാഡെസിമൽ സംഖ്യയുടെ അംശബിന്ദുവിന് വലതുഭാഗത്തുള്ള അക്കങ്ങളുടെ സ്ഥാനവില 16 ന്റെ നെഗറ്റിവ് കൃത്യങ്കം ആയിരിക്കും ($16^{-1}, 16^{-2}, 16^{-3}, \ldots$) $(2D.4)_{16}$ എന്ന സംഖ്യാ ഉദാഹരണമായി എടുക്കാം.

| | | | |
|---|---|---|---|
| സ്ഥാനവില (Weight) | $16^1$ | $16^0$ | $16^{-1}$ |
| ഹെക്സാഡെസിമൽ അക്കം | 2 | D | 4 |

$$= 2\times16^1 + 13\times16^0 + 4\times\frac{1}{16}$$

$$= 32 + 13 + 0.25$$

$$= 45.25$$

പട്ടിക 2.1 ൽ വിവിധ സംഖ്യാന സമ്പ്രദായത്തിൽ ഉപയോഗിക്കുന്ന ആധാരവും ചിഹ്നങ്ങളും കാണിച്ചിരിക്കുന്നു.

| സംഖ്യാന സമ്പ്രദായം | ആധാരം | ഉപയോഗിക്കുന്ന ചിഹ്നങ്ങൾ |
|---|---|---|
| ബൈനറി | 2 | 0, 1 |
| ഒക്ടൽ | 8 | 0, 1, 2, 3, 4, 5, 6, 7 |
| ഡെസിമൽ | 10 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 |
| ഹെക്സാ ഡെസിമൽ | 16 | 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F |

*പട്ടിക 2.1 വിവിധ സംഖ്യാ സമ്പ്രദായത്തിലുമുള്ള ആധാരവും ചിഹ്നങ്ങളും*

**ഒക്ടൽ, ഹെക്സാഡെസിമൽ സംഖ്യാ സമ്പ്രദായങ്ങളുടെ പ്രാധാന്യം**

കമ്പ്യൂട്ടറിൽ ഡാറ്റ പ്രോസസ്സ് ചെയ്യുന്നതിനും അതിനെ പ്രതിനിധാനം ചെയ്യുന്നതിനും ബൈനറി സംഖ്യാ സമ്പ്രദായമാണ് ഉപയോഗിക്കുന്നത് എന്ന് നമ്മൾ മനസ്സിലാക്കിക്കഴിഞ്ഞു. ബൈനറി സംവിധാനത്തിൽ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യുന്നതിനും അവയുടെ പ്രവർത്തനങ്ങൾക്കും കൂടുതൽ ബിറ്റുകളും പ്രയത്നങ്ങളും ആവശ്യമാണ്. മൂന്നു ബിറ്റുകളുടെ ഗ്രൂപ്പിനെ ഒരു ഒക്ടൽ അക്കമായും (കാരണം $2^3 = 8$) നാലു ബിറ്റുകളുടെ ഗ്രൂപ്പിനെ ഒരു ഹെക്സാഡെസിമൽ അക്കമായും (കാരണം $2^4 = 16$) മാറ്റാവുന്നതും ഇത്തരം ഗ്രൂപ്പുകളെ അവയുടെ തത്തുല്യമായ ഒക്ടൽ, ഹെക്സാഡെസിമൽ ചിഹ്നങ്ങളിലേക്കു മാറ്റാവുന്നതാണ്. ബൈനറി സംഖ്യകളുടെ ഒക്ടൽ, ഹെക്സാ ഡെസിമൽ സംഖ്യാ സമ്പ്രദായത്തിലേക്കുള്ള ഇത്തരം മാറ്റവും തിരിച്ചുള്ള മാറ്റവും വളരെ എളുപ്പമാണ്. ഇലക്ട്രോണിക് സർക്യൂട്ടുകളുടെ രൂപകൽപ്പനയിലും പ്രവർത്തനത്തിലും ഈ പരിവർത്തന പ്രക്രിയ വലിയ തോതിൽ ഉപയോഗിക്കുന്നു.

**സ്വയം വിലയിരുത്താം**

1. ഒരു സംഖ്യാ സമ്പ്രദായത്തിൽ ഉപയോഗിച്ചിരിക്കുന്ന ചിഹ്നങ്ങളുടെ എണ്ണത്തെ ................. എന്ന് വിളിക്കുന്നു.
2. താഴെ കൊടുത്തിരിക്കുന്നതിൽ നിന്ന് അസാധുവായ സംഖ്യകൾ തിരഞ്ഞെടുക്കുക.
   i) $(10101)_8$ ii) $(123)_4$ iii) $(768)_8$ iv) $(ABC)_{16}$
3. ബിറ്റ് എന്ന പദം നിർവചിക്കുക.
4. 7854.25. എന്ന ദശസംഖ്യയുടെ എം.എസ്.ഡി (MSD) കണ്ടുപിടിക്കുക.
5. ഹെക്സാഡെസിമൽ സംഖ്യാ സമ്പ്രദായത്തിന്റെ ആധാരം ........................ ആകുന്നു.

## 2.2 സംഖ്യകളുടെ പരിവർത്തനങ്ങൾ (Number conversions)

വിവിധ സംഖ്യാ സമ്പ്രദായങ്ങളെക്കുറിച്ച് നമ്മൾ പഠിച്ചു കഴിഞ്ഞു, ഒരാധാരത്തിലുള്ള സംഖ്യകളെ മറ്റൊരാധാരത്തിലുള്ള തത്തുല്യ സംഖ്യകളാക്കി പരിവർത്തനം ചെയ്യുന്നതെങ്ങനെയാണെന്നു നമുക്ക് ചർച്ച ചെയ്യാം. ദശസംഖ്യയിൽ നിന്ന് ബൈനറി, ബൈനറിയിൽ നിന്ന് ദശസംഖ്യ, ദശസംഖ്യയിൽ നിന്ന് ഒക്ടൽ എന്നിങ്ങനെ പല വിധത്തിലുള്ള സംഖ്യാ സമ്പ്രദായത്തിലേക്കു പരിവർത്തനം ചെയ്യാം. ഒരു സംഖ്യാ സമ്പ്രദായത്തിൽ നിന്ന് മറ്റൊരു സംഖ്യാ സമ്പ്രദായത്തിലേക്ക് എങ്ങനെ പരിവർത്തനം ചെയ്യാമെന്ന് നമുക്ക് നോക്കാം.

### 2.2.1 ദശസംഖ്യയിൽ നിന്ന് ബൈനറിസംഖ്യയിലേക്കുള്ള പരിവർത്തനം (Decimal to binary conversion)

ആവർത്തിച്ചുള്ള ഹരണം വഴിയാണ് ദശസംഖ്യയെ ബൈനറി സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യുന്നത്. ഈ രീതിയിൽ ദശസംഖ്യയെ 2 കൊണ്ട് തുടർച്ചയായി ഹരിക്കുകയും (സംഖ്യ 0 ആകുന്നത് വരെ), അതിന്റെ ശിഷ്ടങ്ങൾ രേഖപ്പെടുത്തുകയും ചെയ്യുന്നു. MSB അവസാന ശിഷ്ടമായും LSB ആദ്യത്തെ ശിഷ്ടമായും എടുത്ത് ശിഷ്ടങ്ങളെ കൂട്ടമായി എഴുതിയാൽ ദശസംഖ്യക്ക് തുല്യമായ സംഖ്യ ലഭിക്കുന്നു. ഓരോ ഘട്ടത്തിലും ഹരിക്കുമ്പോൾ കിട്ടുന്ന ശിഷ്ടങ്ങൾ ഒന്നുകിൽ 0 അല്ലെങ്കിൽ 1 എന്നീ ബൈനറി അക്കങ്ങൾ ആയിരിക്കും.

**ഉദാഹരണങ്ങൾ**

25 എന്ന ദശസംഖ്യയുടെ ബൈനറിക്ക് തുല്യമായ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ | |
|---|---|---|---|
| 2 | 25 | | |
| 2 | 12 | 1 | LSB |
| 2 | 6 | 0 | |
| 2 | 3 | 0 | |
| 2 | 1 | 1 | |
| | 0 | 1 | MSB |

$(25)_{10} = (11001)_2$

$(80)_{10}$ ന് തുല്യമായ ബൈനറി സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ | |
|---|---|---|---|
| 2 | 80 | | |
| 2 | 40 | 0 | LSB |
| 2 | 20 | 0 | |
| 2 | 10 | 0 | |
| 2 | 5 | 0 | |
| 2 | 2 | 1 | |
| 2 | 1 | 0 | |
| | 0 | 1 | MSB |

$(80)_{10} = (1010000)_2$

**സൂചന:** ഒറ്റ സംഖ്യയായ ദശസംഖ്യയ്ക്ക് തുല്യമായ ബൈനറി സംഖ്യ 1 ൽ അവസാനിക്കുകയും ഇരട്ട സംഖ്യയായ ദശസംഖ്യക്ക് തുല്യമായ ബൈനറി സംഖ്യ 0 ൽ അവസാനിക്കുകയും ചെയ്യുന്നു.

### ദശാംശ സംഖ്യകൾ ബൈനറിയിലേക്ക് പരിവർത്തനം ചെയ്യൽ

ദശാംശ സംഖ്യകൾ ബൈനറിയിലേക്ക് മാറ്റാൻ അതിനെ തുടർച്ചയായി 2 കൊണ്ട് ഗുണിക്കുന്ന രീതിയാണ് നാം ഉപയോഗിക്കുന്നത്. ഉത്തരത്തിന്റെ പൂർണസംഖ്യഭാഗം ബൈനറി ഭിന്നകത്തിലെ MSB ആയിരിക്കും. അടുത്ത ബൈനറി ഭിന്നകത്തിന്റെ പ്രബലതയുള്ള ബിറ്റ് കിട്ടുന്നതിന് വീണ്ടും ഭിന്നക ഭാഗത്തിന്റെ ഉത്തരത്തെ 2 കൊണ്ട് ഗുണിക്കുന്നു. ഭിന്നക ഭാഗം പൂജ്യം ആകുന്നതു വരെയോ അല്ലെങ്കിൽ ആവശ്യമുള്ളത്ര കൃത്യത (Precision) ലഭിക്കുന്നത് വരെയോ ഈ നടപടിക്രമം തുടരുന്നു.

**ഉദാഹരണങ്ങൾ:**

0.75 നെ ബൈനറിയിലേക്ക് മാറ്റുക.

| | |
|---|---|
| | $0.75 \times 2 = 1.50$ |
| 1 | $.50 \times 2 = 1.00$ |
| 1 | .00 |

$(0.75)_{10} = (0.11)_2$

0.625 നെ ബൈനറിയിലേക്ക് മാറ്റുക.

| | |
|---|---|
| | $0.625 \times 2 = 1.25$ |
| 1 | $.25 \times 2 = 0.50$ |
| 0 | $.50 \times 2 = 1.00$ |
| 1 | .00 |

$(0.625)_{10} = (0.101)_2$

15.25 നെ ബൈനറിയിലേക്ക് മാറ്റുക.

15 നെ ബൈനറിയിലേക്കു മാറ്റുക.

| | | ശിഷ്ടങ്ങൾ |
|---|---|---|
| 2 | 15 | |
| 2 | 7 | 1 |
| 2 | 3 | 1 |
| 2 | 1 | 1 |
| | 0 | 1 |

0.25നെ ബൈനറിയിലേക്കു മാറ്റുക

| | |
|---|---|
| | $0.25 \times 2 = 0.50$ |
| 0 | $.50 \times 2 = 1.00$ |
| 1 | .00 |

$(15.25)_{10} = (1111.01)_2$

## 2.2.2 ദശസംഖ്യയിൽ നിന്ന് ഒക്ടലിലേക്കുള്ള പരിവർത്തനം (Decimal to Octal conversion)

ആവർത്തിച്ചുള്ള ഹരണം വഴിയാണ് ദശസംഖ്യയെ ഒക്ടൽ സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യുന്നത്. ദശസംഖ്യയെ 8 കൊണ്ട് തുടർച്ചയായി ഹരിക്കുകയും (സംഖ്യ 0 ആകുന്നത് വരെ), അതിന്റെ ശിഷ്ടങ്ങൾ രേഖപ്പെടുത്തുകയും ചെയ്യുന്നു. MSD അവസാന ശിഷ്ടമായും LSD ആദ്യത്തെ

ശിഷ്ടമായും എടുത്ത് ശിഷ്ടങ്ങളെ കൂട്ടമായി എഴുതിയാൽ ഒക്ടൽസംഖ്യക്ക് തുല്യമായ സംഖ്യ ലഭിക്കുന്നു. ഓരോ ഘട്ടത്തിലും ഹരിക്കുമ്പോൾ കിട്ടുന്ന ശിഷ്ടങ്ങൾ 0, 1, 2, 3, 4, 5, 6 , 7.എന്നിവയിൽ ഏതെങ്കിലും ആയിരിക്കും.

**ഉദാഹരണങ്ങൾ:**

125 എന്ന ദശസംഖ്യക്ക് തുല്യമായ ഒക്ടൽ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ | |
|---|---|---|---|
| 8 | 125 | | |
| 8 | 15 | 5 | ↑ LSD |
| 8 | 1 | 7 | |
| | 0 | 1 | MSD |

$(125)_{10} = (175)_8$

$(400)_{10}$ ന് തുല്യമായ ഒക്ടൽ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ |
|---|---|---|
| 8 | 400 | |
| 8 | 50 | 0 ↑ |
| 8 | 6 | 2 |
| | 0 | 6 |

$(400)_{10} = (620)_8$

## 2.2.3 ദശസംഖ്യയിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്കുള്ള പരിവർത്തനം (Decimal to hexadecimal conversion)

ന ആവർത്തിച്ചുള്ള ഹരണം വഴിയാണ് ദശസംഖ്യയെ ഹെക്സാഡെസിമൽ സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യുന്നത്. ദശസംഖ്യയെ 16 കൊണ്ട് തുടർച്ചയായി ഹരിക്കുകയും (സംഖ്യ 0 ആകുന്നത് വരെ), അതിന്റെ ശിഷ്ടങ്ങൾ രേഖപ്പെടുത്തുകയും ചെയ്യുന്നു. MSD അവസാന ശിഷ്ടമായും LSD ആദ്യ ശിഷ്ടമായും എടുത്ത് ശിഷ്ടങ്ങളെ കൂട്ടമായി എഴുതിയാൽ ഹെക്സാഡെസിമൽ സംഖ്യക്ക് തുല്യമായ സംഖ്യ ലഭിക്കുന്നു. ഓരോ ഘട്ടത്തിലും ഹരിക്കുമ്പോൾ കിട്ടുന്ന ശിഷ്ടങ്ങൾ 0, 1, 2, 3, 4, 5, 6,7 , 8, 9, 10, 11, 12, 13, 14, 15 എന്നിവയിൽ ഏതെങ്കിലും ആയിരിക്കും. കിട്ടുന്ന ശിഷ്ടങ്ങൾ 10, 11, 12, 13, 14, 15 ആണെങ്കിൽ അതിനെ യഥാക്രമം A, B, C, D, E, F എന്നിങ്ങനെ രേഖപ്പെടുത്തണം.

**ഉദാഹരണങ്ങൾ:**

155 എന്ന ദശസംഖ്യക്ക് തുല്യമായ ഹെക്സാഡെസിമൽ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ | |
|---|---|---|---|
| 16 | 155 | | |
| 16 | 9 | 11 (B) ↑ | → LSD |
| | 0 | 9 | → MSD |

$(155)_{10} = (9B)_{16}$

**ഉദാഹരണങ്ങൾ:** $380_{10}$ തുല്യമായ ഹെക്സാഡെസിമൽ സംഖ്യ കണ്ടുപിടിക്കുക.

| | | ശിഷ്ടങ്ങൾ |
|---|---|---|
| 16 | 380 | |
| 16 | 23 | 12 (C) |
| 16 | 1 | 7 |
| | 0 | 1 |

$$(380)_{10} = (17C)_{16}$$

## 2.2.4 ബൈനറിയിൽ നിന്ന് ദശസംഖ്യയിലേക്കുള്ള പരിവർത്തനം (Binary to decimal conversion)

ബൈനറി സംഖ്യകൾക്ക് തുല്യമായ ദശസംഖ്യ കാണുന്നതിന്, ബൈനറി സംഖ്യയിലെ ഓരോ അക്കത്തിനെയും, അതിന്റെ സ്ഥാനവില കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കണ്ടാൽ മതി. സ്ഥാനവില 2 ന്റെ കൃത്യങ്കം ആയിരിക്കും ($2^0, 2^1, 2^2, 2^3, \ldots$)..

**ഉദാഹരണങ്ങൾ:**

$(10110)_2$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 0 | 1 | 1 | 0 |

$$\begin{aligned}(10110)_2 &= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0\\ &= 16 + 0 + 4 + 2 + 0\\ &= 22\end{aligned}$$

$$(10110)_2 = (22)_{10}$$

$(11011)_2$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 1 | 0 | 1 | 1 |

$$\begin{aligned}(11011)_2 &= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0\\ &= 16 + 8 + 2 + 1\\ &= 27\end{aligned}$$

$$(11011)_2 = (27)_{10}$$

$(1100010)_2$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 1 | 0 | 0 | 0 | 1 | 0 |

$(1100010)_2 = 1\times2^6 + 1\times2^5 + 0\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 64 + 32 + 2$

$= 98$

$(1100010)_2 = (98)_{10}$

പട്ടിക 2.2 ൽ രണ്ടിന്റെ 10 വരെയുള്ള കൃത്യങ്കങ്ങൾ കൊടുത്തിരിക്കുന്നു.

| $2^{10}$ | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1024 | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

പട്ടിക 2.2 രണ്ടിന്റെ കൃത്യങ്കങ്ങൾ

## ബൈനറി ഭിന്നകങ്ങൾ ദശസംഖ്യയിലേക്ക് പരിവർത്തനം ചെയ്യൽ

ഒരു ബൈനറി ഭിന്നസംഖ്യ ദശസംഖ്യയിലേക്ക് മാറ്റുന്നതിന്, ഓരോ അക്കത്തിനെയും അതിന്റെ സ്ഥാനവില കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കണ്ടാൽ മതി. ബൈനറി അംശബിന്ദുവിന് ശേഷമുള്ള അക്കത്തിന്റെ സ്ഥാനവില 2 ന്റെ നെഗറ്റീവ് കൃത്യങ്കം ആയിരിക്കും $(2^{-1}, 2^{-2}, 2^{-3}, \ldots)$

**ഉദാഹരണങ്ങൾ:**

$(0.1011)_2$ നെ ദശസംഖ്യയിലേക്ക് മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ |
|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 0 | 1 | 1 |

$(0.1011)_2 = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$

$= 0.5 + 0 + 0.125 + 0.0625$

$= 0.6875$

$(0.1011)_2 = (0.6875)_{10}$

$(0.101)_2$ നെ ദശസംഖ്യയിലേക്ക് മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ |
|---|---|---|---|
| ബൈനറി അക്കം | 1 | 0 | 1 |

$(0.101)_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 0.5 + 0 + 0.125$

$= 0.625$

$(0.101)_2 = (0.625)_{10}$

$(1010.11)_2$ നെ ദശസംഖ്യയിലേക്ക് മാറ്റുക.

| സ്ഥാനവില (Weight) | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|
| ബൈനറി അക്കം | 1 | 0 | 1 | 1 |

$(1010)_2 \quad = 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 8 + 0+2+0$

$= 10 \qquad (1010)_2 = (10)_{10}$

$(0.11)_2 \quad = 1\times2^{-1} + 1\times2^{-2}$

$= 0.5 + 0.25$

$= 0.75 \qquad (0.11)_2 = (0.75)_{10}$

| സ്ഥാനവില (Weight) | $2^{-1}$ | $2^{-2}$ |
|---|---|---|
| ബൈനറി അക്കം | 1 | 1 |

$(1010.11)_2 = (10.75)_{10}$

പട്ടിക 2.3 ൽ രണ്ടിന്റെ നെഗറ്റീവ് കൃത്യങ്കങ്ങൾ കാണിച്ചിരിക്കുന്നു.

| $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ | $2^{-5}$ |
|---|---|---|---|---|
| 0.5 | 0.25 | 0.125 | 0.0625 | 0.03125 |

*പട്ടിക 2.3 രണ്ടിന്റെ നെഗറ്റീവ് കൃത്യങ്കങ്ങൾ*

### 2.2.5 ഒക്ടൽ സംഖ്യയിൽ നിന്ന് ദശസംഖ്യയിലേക്കുള്ള പരിവർത്തനം (Octal to decimal conversion)

ഒക്ടൽ സംഖ്യയെ ദശസംഖ്യയിലേക്കു മാറ്റുന്നതിന്, ഒക്ടൽ സംഖ്യയിലെ ഓരോ അക്കത്തിനെയും, അതിന്റെ സ്ഥാനവില കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കണ്ടാൽ മതി. സ്ഥാനവില 8 ന്റെ കൃത്യങ്കം ആയിരിക്കും $(8^0, 8^1, 8^2, 8^3, ...)$.

**ഉദാഹരണങ്ങൾ:**

$(257)_8$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $8^2$ | $8^1$ | $8^0$ |
|---|---|---|---|
| ഒക്ടൽ അക്കം | 2 | 5 | 7 |

$(257)_8 \quad = 2\times8^2 + 5\times8^1 + 7\times8^0$

$= 128 + 40 + 7$

$= 175$

$(257)_8 = (175)_{10}$

$(157)_8$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില (Weight) | $8^2$ | $8^1$ | $8^0$ |
|---|---|---|---|
| ഒക്ടൽ അക്കം | 1 | 5 | 7 |

$(157)_8 = 1\times8^2 + 5\times8^1 + 7\times8^0$

$= 64 + 40 + 7$

$= 111$

$(157)_8 = (111)_{10}$

$(1005)_8$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാനവില Weight | $8^3$ | $8^2$ | $8^1$ | $8^0$ |
|---|---|---|---|---|
| ഒക്റ്റൽ അക്കം | 1 | 0 | 0 | 5 |

$(1005)_8 = 1\times8^3 + 0\times8^2 + 0\times8^1 + 5\times8^0$

$= 512+5$

$= 517$

$(1005)_8 = (517)_{10}$

## 2.2.6 ഹെക്സാഡെസിമൽ സംഖ്യയിൽ നിന്ന് ദശസംഖ്യയിലേക്കുള്ള പരിവർത്തനം (Hexadecimal to decimal conversion)

ഹെക്സാഡെസിമൽ സംഖ്യയെ ദശസംഖ്യയിലേക്കു മാറ്റുന്നതിന്, ഹെക്സാഡെസിമൽ സംഖ്യയിലെ ഓരോ അക്കത്തിനെയും, അതിന്റെ സ്ഥാനവില കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കണ്ടാൽ മതി. സ്ഥാനവില 16 ന്റെ കൃത്യങ്കം ആയിരിക്കും ($16^0$, $16^1$, $16^2$, ...). ഹെക്സാഡെസിമൽ അക്കങ്ങൾ A, B, C, D, E, F ആണെങ്കിൽ അത് യഥാക്രമം 10, 11, 12, 13, 14, 15 എന്നിങ്ങനെ മാറ്റി എഴുതണം.

**ഉദാഹരണങ്ങൾ:**

$(AB)_{16}$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാന വില (Weight) | $16^1$ | $16^0$ |
|---|---|---|
| ഹെക്സാഡെസിമൽ അക്കം | A | B |

A = 10 B = 11

$(AB)_{16} = 10\times16^1+11\times16^0$

$= 160 + 11$

$= 171$

$(AB)_{16} = (171)_{10}$

**ഉദാഹരണങ്ങൾ:** $(2D5)_{16}$ നെ ദശസംഖ്യയിലേക്കു മാറ്റുക.

| സ്ഥാന വില (Weight) | $16^1$ | $16^1$ | $16^0$ |
|---|---|---|---|
| ഹെക്സാഡെസിമൽ അക്കം | 2 | D | 5 |

D = 13

$(2D5)_{16} = 2\times16^2+13\times16^1+5\times16^0$

$= 512+208+5$

$= 725$

$\boxed{(AB)_{16} = (171)_{10}}$

## 2.2.7 ഒക്ടലിൽ നിന്ന് ബൈനറിയിലേക്കുള്ള പരിവർത്തനം (Octal to binary conversion)

ഓരോ ഒക്ടൽ അക്കവും തത്തുല്യമായ 3 ബിറ്റ് ബൈനറി അക്കത്തിലേക്ക് മാറ്റി എഴുതിയാൽ ഒക്ടൽ സംഖ്യ ബൈനറി സംഖ്യയായി പരിവർത്തനം ചെയ്യാനാകും. സാധ്യമായ എട്ട് ഒക്ടൽ അക്കങ്ങളും അവയുടെ തത്തുല്യ ബൈനറി അക്കങ്ങളും പട്ടിക 2.4 ൽ നൽകിയിരിക്കുന്നു.

| ഒക്ടൽ അക്കം | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| തുല്യമായ ബൈനറി | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |

*പട്ടിക 2.4 ഒക്ടൽ അക്കങ്ങളുടെ തത്തുല്യമായ ബൈനറി സംഖ്യകൾ.*

**ഉദാഹരണങ്ങൾ:**

$(437)_8$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

ഓരോ ഒക്ടൽ അക്കത്തിനും തുല്യമായ 3 ബിറ്റ് ബൈനറി അക്കങ്ങൾ താഴെക്കൊടുത്തിരിക്കുന്നു.

$$\begin{array}{ccc} 4 & 3 & 7 \\ \downarrow & \downarrow & \downarrow \\ 100 & 011 & 111 \end{array}$$

$\boxed{(437)_8=(100011111)_2}$

$(7201)_8$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

ഓരോ ഒക്ടൽ അക്കത്തിനും തുല്യമായ 3 ബിറ്റ് ബൈനറി അക്കങ്ങൾ താഴെക്കൊടുത്തിരിക്കുന്നു.

$$\begin{array}{cccc} 7 & 2 & 0 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ 111 & 010 & 000 & 001 \end{array}$$

$\boxed{(7201)_8 = (111010000001)_2}$

## 2.2.8 ഹെക്സാഡെസിമലിൽ നിന്ന് ബൈനറിയിലേക്കുള്ള പരിവർത്തനം (Hexadecimal to binary conversion)

ഓരോ ഹെക്സാഡെസിമൽ അക്കവും തത്തുല്യമായ 4 ബിറ്റ് ബൈനറി ആക്കി മാറ്റി എഴുതിയാൽ ഹെക്സാഡെസിമൽ സംഖ്യ ബൈനറിയായി പരിവർത്തനം ചെയ്യാനാകും. ഹെക്സാഡെസിമൽ അക്കങ്ങളും അവയ്ക്കു തുല്യമായ ബൈനറി അക്കങ്ങളും പട്ടിക 2.5 ൽ നൽകിയിരിക്കുന്നു.

**ഉദാഹരണങ്ങൾ:**

$(AB)_{16}$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

ഓരോ ഹെക്സാഡെസിമൽ അക്കത്തിനും തുല്യമായ 4 ബിറ്റ് ബൈനറി അക്കങ്ങൾ താഴെക്കൊടുത്തിരിക്കുന്നു.

| A | B |
|---|---|
| ↓ | ↓ |
| 1010 | 1011 |

$$(AB)_{16} = (10101011)_2$$

$(2F15)_{16}$ നെ ബൈനറിയിലേക്കു മാറ്റുക

ഓരോ ഹെക്സാഡെസിമൽ അക്കത്തിനും തുല്യമായ 4 ബിറ്റ് ബൈനറി അക്കങ്ങൾ താഴെക്കൊടുത്തിരിക്കുന്നു.

| 2 | F | 1 | 5 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 0010 | 1111 | 0001 | 0101 |

$$(2F15)_{16} = (10111100010101)_2$$

| ഹെക്സാ ഡെസിമൽ അക്കം | തുല്യമായ ബൈനറി |
|---|---|
| 0 | 0000 |
| 1 | 0001 |
| 2 | 0010 |
| 3 | 0011 |
| 4 | 0100 |
| 5 | 0101 |
| 6 | 0110 |
| 7 | 0111 |
| 8 | 1000 |
| 9 | 1001 |
| A | 1010 |
| B | 1011 |
| C | 1100 |
| D | 1101 |
| E | 1110 |
| F | 1111 |

*പട്ടിക 2.5 ഹെക്സാഡെസിമൽ അക്കങ്ങളുടെ തത്തുല്യമായ ബൈനറി അക്കങ്ങൾ.*

### 2.2.9 ബൈനറിയിൽ നിന്നും ഒക്ടലിലേക്കുള്ള പരിവർത്തനം (Binary to octal conversion)

തന്നിരിക്കുന്ന ബൈനറി സംഖ്യ വലത്തു നിന്ന് ഇടത്തേക്ക് 3 ബൈനറി ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി അതിന്റെ തത്തുല്യമായ ഒക്ടൽ അക്കം എഴുതിയാൽ ഒരു ബൈനറി സംഖ്യ ഒക്ടൽ സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യാം. മൂന്നിന്റെ കൂട്ടങ്ങൾ ആക്കുമ്പോൾ ഏറ്റവും ഇടത് വശത്തെ കൂട്ടത്തിൽ 3 ബിറ്റുകൾ തികയുന്നില്ലെങ്കിൽ ഇടത് വശത്ത് ആവശ്യമായ പൂജ്യങ്ങൾ കൊടുത്ത് 3 ബിറ്റ് രൂപത്തിൽ ആക്കണം.

**ഉദാഹരണങ്ങൾ:**

$(101100111)_2$ നെ ഒക്ടലിലേക്കു മാറ്റുക.

ബൈനറി സംഖ്യ 101100111 ന്റെ വലതുഭാഗത്ത് നിന്ന് ചുവടെ കാണിച്ചിരിക്കുന്നതുപോലെ കൂട്ടങ്ങളാക്കാം.

| 101 | 100 | 111 |
|---|---|---|
| ↓ | ↓ | ↓ |
| 5 | 4 | 7 |

$$(101100111)_2 = (547)_8$$

$(10011000011)_2$ നെ ഒക്ടലിലേക്കു മാറ്റുക.

ബൈനറി സംഖ്യ 10011000011 ന്റെ വലതുഭാഗത്ത് നിന്ന് ചുവടെ കാണിച്ചിരിക്കുന്നതു പോലെ കൂട്ടങ്ങളാക്കാം.

കൂട്ടങ്ങളാക്കിയശേഷം ഏറ്റവും ഇടത് ഭാഗത്തെ കൂട്ടത്തിൽ 3 ബിറ്റുകൾ ഇല്ലെങ്കിൽ ആവശ്യമായ 0 ചേർത്ത് 3 ബിറ്റുകൾ ആക്കുക.

| 010 | 011 | 000 | 011 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 2 | 3 | 0 | 3 |

$(10011000011)_2 = (2303)_8$

## 2.2.10 ബൈനറിയിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്കുള്ള പരിവർത്തനം (Binary to hexadecimal conversion)

തന്നിരിക്കുന്ന ബൈനറി സംഖ്യ വലത്തു നിന്ന് ഇടത്തേക്ക് 4 ബൈനറി ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി അതിന്റെ തത്തുല്യമായ ഹെക്സാഡെസിമൽ അക്കം എഴുതിയാൽ ഒരു ബൈനറി സംഖ്യയെ ഹെക്സാഡെസിമൽ സംഖ്യയിലേക്കു പരിവർത്തനം ചെയ്യാം. നാലിന്റെ കൂട്ടങ്ങൾ ആക്കുമ്പോൾ ഏറ്റവും ഇടത് വശത്തെ കൂട്ടത്തിൽ 4 ബിറ്റുകൾ തികയുന്നില്ലെങ്കിൽ ഇടത് വശത്ത് ആവശ്യമായ പൂജ്യങ്ങൾ കൊടുത്ത് 4 ബിറ്റ് രൂപത്തിൽ ആക്കണം.

**ഉദാഹരണങ്ങൾ:**

$(101100111010)_2$ നെ ഹെക്സാഡെസിമലിലേക്കു മാറ്റുക.

ബൈനറി സംഖ്യ 101100111010 ന്റെ വലതുഭാഗത്ത് നിന്ന് ചുവടെ കാണിച്ചിരിക്കുന്നതുപപോലെ കൂട്ടങ്ങളാക്കാം.

| 1011 | 0011 | 1010 |
|---|---|---|
| ↓ | ↓ | ↓ |
| B | 3 | A |

$(101100111010)_2 = (B3A)_{16}$

$(110111100001100)_2$ നെ ഹെക്സാഡെസിമലിലേക്കു മാറ്റുക.

ബൈനറി സംഖ്യ 110111100001100 ന്റെ വലതുഭാഗത്ത് നിന്ന് ചുവടെ കാണിച്ചിരിക്കുന്നതു പോലെ കൂട്ടങ്ങളാക്കാം.

കൂട്ടങ്ങളാക്കിയശേഷം ഏറ്റവും ഇടത് ഭാഗത്തെ കൂട്ടത്തിൽ 4 ബിറ്റുകൾ ഇല്ലെങ്കിൽ ആവശ്യമായ 0 ചേർത്ത് 4 ബിറ്റുകൾ ആക്കുക.

| 0110 | 1111 | 0000 | 1100 |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| 6 | F | 0 | C |

$(110111100001100)_2 = (6F0C)_{16}$

## 2.2.11 ഒക്ടലിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്കുള്ള പരിവർത്തനം. (Octal to Hexadecimal conversion)

ഒക്ടൽ സംഖ്യയിൽ നിന്ന് ഹെക്സാഡെസിമൽ സംഖ്യയിലേക്ക് മാറ്റുന്നതിന് രണ്ട് ഘട്ടങ്ങൾ ഉണ്ട്. ആദ്യം ഒക്ടൽ സംഖ്യ ബൈനറിയായി പരിവർത്തനം ചെയ്യുക. ഈ ബൈനറി സംഖ്യ തത്തുല്യമായ ഹെക്സഡെസിമൽ സംഖ്യയിലേക്കു മാറ്റുക.

**ഉദാഹരണം:**

$(457)_8$ നെ ഹെക്സാഡെസിമലിലേക്കു മാറ്റുക.

ഘട്ടം 1. $(457)_8$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

$$(457)_8 = \underset{100}{\overset{4}{\downarrow}} \quad \underset{101}{\overset{5}{\downarrow}} \quad \underset{111}{\overset{7}{\downarrow}}$$

$$= (100101111)_2$$

ഘട്ടം 2. $(100101111)_2$ നെ ഹെക്സാഡെസിമലിലേക്കു മാറ്റുക.

$(100101111)_2$ നെ 4 ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി മാറ്റുക.

$$(100101111)_2 = \underset{1}{\overset{0001}{\downarrow}} \quad \underset{2}{\overset{0010}{\downarrow}} \quad \underset{F}{\overset{1111}{\downarrow}}$$

$$= (12F)_{16}$$

$$\boxed{(457)_8 = (12F)_{16}}$$

## 2.2.12 ഹെക്സാഡെസിമലിൽ നിന്ന് ഒക്ടലിലേക്കുള്ള പരിവർത്തനം (Hexadecimal to Octal conversion)

ഹെക്സാഡെസിമൽ സംഖ്യയിൽ നിന്ന് ഒക്ടൽ സംഖ്യയിലേക്ക് മാറ്റുന്നതിന് രണ്ട് ഘട്ടങ്ങൾ ഉണ്ട്. ആദ്യം ഹെക്സാഡെസിമൽ സംഖ്യ ബൈനറിയായി പരിവർത്തനം ചെയ്യുക. ഈ ബൈനറി സംഖ്യ തത്തുല്യമായ ഒക്ടൽ സംഖ്യയിലേക്കു മാറ്റുക..

**ഉദാഹരണം:**

$(A2D)_{16}$ നെ ഒക്ടലിലേക്കു മാറ്റുക.

ഘട്ടം 1. $(A2D)_{16}$ നെ ബൈനറിയിലേക്കു മാറ്റുക.

$$(A2D)_{16} = \underset{1010}{\overset{A}{\downarrow}} \quad \underset{0010}{\overset{2}{\downarrow}} \quad \underset{1101}{\overset{D}{\downarrow}}$$

$$= (101000101101)_2$$

ഘട്ടം 2. $(101000101101)_2$ നെ ഒക്ടലിലേക്കു മാറ്റുക.

$(101000101101)_2$ നെ 3 ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി മാറ്റുക.

$$(101000101101)_2 = \underset{5}{\overset{101}{\downarrow}} \quad \underset{0}{\overset{000}{\downarrow}} \quad \underset{5}{\overset{101}{\downarrow}} \quad \underset{5}{\overset{101}{\downarrow}}$$

$$= (5055)_8$$

$$\boxed{(A2D)_{16} = (5055)_8}$$

**പട്ടിക 2.6 ൽ വിവിധ സംഖ്യാ പരിവർത്തനങ്ങളുടെ നടപടിക്രമങ്ങൾ കാണിച്ചിക്കുന്നു.**

| സംഖ്യ പരിവർത്തനം | നടപടിക്രമം |
|---|---|
| ദശസംഖ്യയിൽ നിന്ന് ബൈനറിയിലേക്ക് | തുടർച്ചയായി 2 കൊണ്ട് ഹരിച്ച് ശിഷ്ടങ്ങൾ കൂട്ടങ്ങളാക്കുക. |
| ദശസംഖ്യയിൽ നിന്ന് | തുടർച്ചയായി 8 കൊണ്ട് ഹരിച്ച് ശിഷ്ടങ്ങൾ കൂട്ടങ്ങളാക്കുക. |
| ഒക്ടലിലേക്ക് | |
| ദശസംഖ്യയിൽ നിന്ന് | തുടർച്ചയായി 16 കൊണ്ട് ഹരിച്ച് ശിഷ്ടങ്ങൾ കൂട്ടങ്ങളാക്കുക. |
| ഹെക്സാഡെസിമലിലേക്ക് | |
| ബൈനറിയിൽ നിന്ന് ദശസംഖ്യയിലേക്ക് | ബൈനറി സംഖ്യയിലെ ഓരോ അക്കത്തിന്റെയും സ്ഥാനവില (2 ന്റെ കൃത്യങ്കം) കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കാണുക. |
| ഒക്ടലിൽ നിന്ന് ദശസംഖ്യയിൽലേക്ക് | ഒക്ടൽ സംഖ്യയിലെ ഓരോ അക്കത്തിന്റെയും സ്ഥാനവില (8 ന്റെ കൃത്യങ്കം) കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കാണുക. |
| ഹെക്സാഡെസിമലിൽ നിന്ന് ദശസംഖ്യയിലേക്ക് | ഹെക്സാഡെസിമൽ സംഖ്യയിലെ ഓരോ അക്കത്തിന്റെയും സ്ഥാനവില (16 ന്റെ കൃത്യങ്കം) കൊണ്ടു ക്രമമായി ഗുണിച്ച് തുക കാണുക. |
| ഒക്ടലിൽ നിന്ന് ബൈനറിയിലേക്ക് | ഓരോ ഒക്ടൽ അക്കവും 3 ബിറ്റ് ബൈനറി സംഖ്യ ആയി പരിവർത്തനം ചെയ്യുക. |
| ഹെക്സാഡെസിമലിൽ നിന്ന് ബൈനറിയിലേക്ക് | ഓരോ ഹെക്സാഡെസിമൽ അക്കവും 4 ബിറ്റ് ബൈനറി സംഖ്യ ആയി പരിവർത്തനം ചെയ്യുക. |
| ബൈനറിയിൽ നിന്ന് | ബൈനറി സംഖ്യ വലത്തു നിന്ന് ഇടത്തേക്ക് 3 ബൈനറി ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി |
| ഒക്ടലിലേക്ക് | അതിന്റെ തുല്യമായ ഒക്ടൽ അക്കം എഴുതുക. |
| ബൈനറിയിൽ നിന്ന് | ബൈനറി സംഖ്യ വലത്തു നിന്ന് ഇടത്തേക്ക് 4 ബൈനറി ബിറ്റുകളുടെ കൂട്ടങ്ങളാക്കി |
| ഹെക്സാഡെസിമലിലേക്ക് | അതിന്റെ തുല്യമായ ഹെക്സാഡെസിമൽ അക്കം എഴുതുക. |
| ഒക്ടലിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്ക് | ഒക്ടലിനെ ബൈനറിയിലേക്കും തുടർന്ന് ബൈനറിയിൽ നിന്ന് ഹെക്സാഡെസിമലിലേക്കും മാറ്റുക. |
| ഹെക്സാഡെസിമലിൽ നിന്ന് ഒക്ടലിലേക്ക് | ഹെക്സാഡെസിമലിനെ ബൈനറിയിലേക്കും തുടർന്ന് ബൈനറിയിൽ നിന്ന് ഒക്ടലിലേക്കും മാറ്റുക. |

*പട്ടിക 2.6 വിവിധ സംഖ്യാ പരിവർത്തനങ്ങളുടെ നടപടിക്രമങ്ങൾ*

## സ്വയം വിലയിരുത്താം

1 31 എന്ന ദശസംഖ്യ ബൈനറിയിലേക്കു മാറ്റുക.

2 $(10001)_2$ നു തത്തുല്യമായ ദശസംഖ്യ കണ്ടുപിടിക്കുക.

3 $(x)_8 = (101011)_2$, ആയാൽ x ന്റെ വില കാണുക.

4 വിട്ട ഭാഗം പൂരിപ്പിക്കുക.

a) $(\_\_\_\_\_\_\_)_2 = (AB)_{16}$

b) $(\_\_D\_\_)_{16} = (1010\_\_\_\_1000)_2$

c) $0.25_{10} = (\_\_\_\_\_\_)_2$

5 താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകളിൽ ഏറ്റവും വലിയ സംഖ്യ കണ്ടുപിടിക്കുക.

(i) $(1001)_2$  (ii) $(A)_{16}$  (iii) $(10)_8$ (iv) $(11)_{10}$

## 2.3 ബൈനറി അരിത്തമറ്റിക്

ദശസംഖ്യാ സമ്പ്രദായത്തിലുള്ളത് പോലെ ദ്വയസംഖ്യാ സമ്പ്രദായത്തിലും ഗണിത ക്രിയകൾ ചെയ്യാം. നമ്മൾ രണ്ട് ദശസംഖ്യകൾ കമ്പ്യൂട്ടറിൽ സങ്കലനം (addition) ചെയ്യാൻ നിർദേശം നൽകുമ്പോൾ, കമ്പ്യൂട്ടർ അതിന്റെ തുല്യമായ ബൈനറി സംഖ്യകൾ ആണ് കൂട്ടുന്നത്. ബൈനറി സംഖ്യകളുടെ സങ്കലനവും (addition) വ്യവകലനവും (subtraction) എങ്ങനെയാണ് ചെയ്യുന്നത് എന്ന് നമുക്ക് നോക്കാം.

### 2.3.1 ബൈനറി സംഖ്യകളുടെ സങ്കലനം (Binary addition)

രണ്ട് ബിറ്റുകൾ കൂട്ടുവാനുള്ള നിയമങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നു.

| A | B | തുക | ശിഷ്ടം |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

ഒന്നും ഒന്നും കൂട്ടുമ്പോൾ മാത്രമാണ് ശിഷ്ട (ക്യാരി) ബിറ്റ് 1 ഉണ്ടാകുന്നത് എന്ന് ശ്രദ്ധിക്കുക. മൂന്നു ഒന്നുകൾ കൂട്ടുമ്പോൾ (1+1+1) തുക 1 ഉം ശിഷ്ടം (ക്യാരി) ബിറ്റ് 1 ഉം കിട്ടുന്നു.

**ഉദാഹരണങ്ങൾ:**

ബൈനറി സംഖ്യകളായ 1011 ന്റെയും 1001 ന്റെയും തുക കണ്ടുപിടിക്കുക.

```
 1 0 1 1 +
 1 0 0 1
10 1 0 0
```

ബൈനറി സംഖ്യകളായ 110111 ന്റെയും 100110 ന്റെയും തുക കണ്ടുപിടിക്കുക.

```
  1 1 0 1 1 1 +
  1 0 0 1 1 0
1 0 1 1 1 0 1
```

### 2.3.2 ബൈനറി സംഖ്യകളുടെ വ്യവകലനം (Binary subtraction)

ഒരു ബൈനറി ബിറ്റിൽ നിന്ന് മറ്റൊരു ബൈനറി ബിറ്റ് കുറയ്ക്കുവാനുള്ള നിയമം താഴെ കൊടുത്തിരിക്കുന്നു.

| A | B | വ്യത്യാസം | കടം |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

0 ൽ നിന്ന് 1 കുറച്ചാൽ വ്യത്യാസം 1 ആണ്, എന്നാൽ ആദ്യത്തെ ബിറ്റിന്റെ ഇടതു ഭാഗത്ത് തൊട്ടടുത്തുള്ള ബിറ്റിൽ നിന്ന് 1 കടമെടുക്കുന്നു. വലിയ ബൈനറി സംഖ്യയിൽ നിന്ന് ഒരു ചെറിയ ബൈനറി സംഖ്യ കുറയ്ക്കുവാൻ മാത്രമേ മേൽപ്പറഞ്ഞ നിയമങ്ങൾ ഉപയോഗിക്കുവാൻ സാധിക്കുകയുള്ളൂ.

**ഉദാഹരണങ്ങൾ:**

$(10101)_2$ ഇൽ നിന്നും $(11111)_2$.

```
11111 -
10101
01010
```

$(10111)_2$ ഇൽ നിന്നും $(101000)_2$.

```
101000 -
 10111
 10001
```

## 2.4 ഡാറ്റയുടെ പ്രതിനിധാനം (Data representation)

സംഖ്യകൾ, അക്ഷരങ്ങൾ, ചിത്രങ്ങൾ, ശബ്ദങ്ങൾ, വീഡിയോകൾ തുടങ്ങിയ ഡാറ്റയെ കമ്പ്യൂട്ടറുകളിൽ നിശ്ചിത ബിറ്റുകളുടെ കൂട്ടങ്ങളായിട്ടാണ് പ്രതിനിധാനം ചെയ്യുന്നത്. എന്നിരുന്നാലും എല്ലാതരം ഡാറ്റകളെയും കമ്പ്യൂട്ടർ പ്രതിനിധാനം ചെയ്യുന്നതും പ്രോസസ്സ് ചെയ്യുന്നതും നിശ്ചിത എണ്ണം ബിറ്റുകളായിട്ടാണ്. ഒരു കമ്പ്യൂട്ടറിൽ ആന്തരികമായി ഡാറ്റയെ പ്രതിനിധീകരിക്കുന്നതിന് ഉപയോഗിക്കുന്ന രീതിയാണ് ഡാറ്റ പ്രതിനിധാനം. കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയിൽ വ്യത്യസ്ത തരത്തിലുള്ള ഡാറ്റ എങ്ങനെ പ്രതിനിധീകരിക്കുന്നു എന്ന് നമുക്ക് നോക്കാം.

### 2.4.1 സംഖ്യകളുടെ പ്രതിനിധാനം (Representation of numbres)

സംഖ്യകളെ പൂർണസംഖ്യകൾ, ദശാംശസംഖ്യകൾ എന്നിങ്ങനെ രണ്ടായി തിരിക്കാം. പൂർണസംഖ്യകൾ ഭിന്നസംഖ്യാ ഭാഗം ഇല്ലാത്ത സംഖ്യകൾ ആകുന്നു. ദശാംശസംഖ്യ (Floating Point Number) അല്ലെങ്കിൽ രേഖീയസംഖ്യ ഭിന്നകഭാഗത്തോട് കൂടിയ സംഖ്യ ആകുന്നു. ഈ രണ്ടു സംഖ്യകളെയും കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയിൽ വ്യത്യസ്തമായിട്ടാണ് കൈകാര്യം ചെയ്യുന്നത്. പൂർണസംഖ്യകൾ എങ്ങനെയാണ് മെമ്മറിയിൽ പ്രതിനിധാനം ചെയ്യുന്നത് എന്ന് നമുക്ക് നോക്കാം.

### എ. പൂർണസംഖ്യകളുടെ പ്രതിനിധാനം (Representation of integers)

ഒരു പൂർണ സംഖ്യ കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയിൽ പ്രതിനിധീകരിക്കുന്നത് മൂന്ന് രീതിയിലാണ്.

i) ചിഹ്നവും മൂല്യവും കൊണ്ടുള്ള പ്രതിനിധാനം (Sign and magnitude representation)

ii) 1 ന്റെ പൂരകം കൊണ്ടുള്ള പ്രതിനിധാനം (1's complement representation)

iii) 2 ന്റെ പൂരകം കൊണ്ടുള്ള പ്രതിനിധാനം (2's complement representation)

കമ്പ്യൂട്ടർ പ്രോസസ്സർ ഒരു യൂണിറ്റായി കൈകാര്യം ചെയ്യുന്ന നിശ്ചിത വ്യാപ്തിയിലുള്ള ഒരു കൂട്ടം ബിറ്റുകളെയാണ് പദം (Word) എന്ന് പറയുന്നത്. ഒരു പദത്തിലെ ബിറ്റുകളുടെ എണ്ണത്തെ പദദൈർഘ്യം (Word length) എന്ന് പറയുന്നു. കമ്പ്യൂട്ടർ രൂപകൽപ്പന ചെയ്യുന്ന വിദഗ്ധരാണ് അതിന്റെ പദദൈർഘ്യം തീരുമാനിക്കുന്നത്. 8, 16, 32, 64 എന്നിവ സാധാരണയായി നിലവിലുള്ള

ചില പദദൈർഘ്യങ്ങളാണ്. പദങ്ങൾ ബിറ്റുകളുടെ കൂട്ടമായതുകൊണ്ട് പദദൈർഘ്യം രണ്ടിന്റെ കൃത്യങ്കങ്ങൾ ആയിരിക്കും.

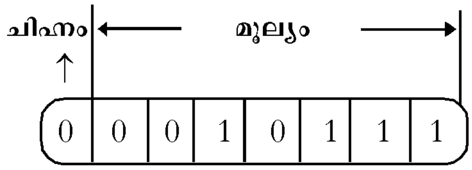
ഇനി ഡാറ്റയെ പ്രതിനിധാനം ചെയ്യുന്ന രീതികൾ (8 ബിറ്റ് പദദൈർഘ്യത്തെ അടിസ്ഥാനമാക്കി) വിശദമായി പരിശോധിക്കാം.

### i. ചിഹ്നവും മൂല്യവും കൊണ്ടുള്ള പ്രതിനിധാനം (Sign and magnitude representation)

ഈ രീതിയിൽ, ഇടതുഭാഗത്തെ ആദ്യത്തെ ബിറ്റ് (MSB) പൂർണസംഖ്യയുടെ ചിഹ്നത്തെയും ബാക്കിയുള്ള 7 ബിറ്റുകൾ സംഖ്യയുടെ മൂല്യത്തെയും പ്രതിനിധാനം ചെയ്യുന്നു. ചിഹ്നത്തെ പ്രതിനിധാനം ചെയ്യുന്ന ബിറ്റ് 1 ആണെങ്കിൽ അത് നെഗറ്റീവ് പൂർണസംഖ്യയും 0 ആണെങ്കിൽ പോസിറ്റീവ് പൂർണസംഖ്യയുമായിരിയ്ക്കും.

**ഉദാഹരണങ്ങൾ:**

+ 23 നെ ചിഹ്നവും മൂല്യവും ഉപയോഗിച്ച് പ്രതിനിധാനം ചെയ്യുക.

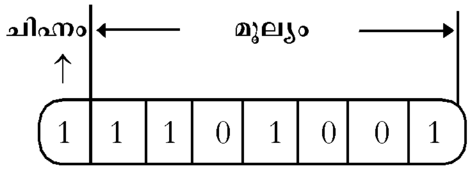



സംഖ്യ പോസിറ്റീവ് ആയതിനാൽ ഒന്നാമത്തെ ബിറ്റ് (MSB) 0 ആകുന്നു.

23 ന് തുല്യമായ 7 ബിറ്റ് ബൈനറി സംഖ്യ= $(0010111)_2$

അതുകൊണ്ട് +23 നെ $(00010111)_2$ കൊണ്ട് പ്രതിനിധീകരിക്കാം.

-105 നെ ചിഹ്നവും മൂല്യവും രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക




സംഖ്യ നെഗറ്റീവ് ആയതിനാൽ ഒന്നാമത്തെ ബിറ്റ് (MSB) 1 ആകുന്നു.

7 ബിറ്റ് ബൈനറി സംഖ്യ 105 = $(1101001)_2$

-105 ന് തുല്യമായ 7 ബിറ്റ് ബൈനറി സംഖ്യ = $(11101001)_2$

അതിനാൽ -105 നെ $(11101001)_2$ കൊണ്ട് പ്രതിനിധീകരിക്കാം

**കുറിപ്പ്** : ഈ രീതിയിൽ 8 ബിറ്റ് പദം കൊണ്ട് $2^8$-1 =255 സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ കഴിയുന്നു. സംഖ്യകൾ -($2^7$ - 1) മുതൽ +($2^7$ - 1) വരെ ആയിരിക്കും. (അതായത് -127 മുതൽ +127 വരെ). അതുപോലെ 16 ബിറ്റ് പദം കൊണ്ട് $2^{16}$ - 1 = 65535 സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാനും കഴിയുന്നു (അതായത് -32767 മുതൽ +32767 വരെ). പൊതുവായി, *n* ബിറ്റ് പദം കൊണ്ട് $2^n$-1 സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ കഴിയും (അതായത് $-(2^{n-1}-1)$ മുതൽ $+(2^{n-1}-1)$. വരെ). പൂർണസംഖ്യയായ പൂജ്യത്തെ +0 = 00000000 എന്നും 0 = 10000000 എന്നും രണ്ട് രീതിയിൽ പ്രതിനിധാനം ചെയ്യാം.

### ii. 1 ന്റെ പൂരകം കൊണ്ടുള്ള പ്രതിനിധാനം (1's complement representation)

ഈ രീതിയിൽ, പൂർണസംഖ്യയുടെ കേവല വിലയ്ക്ക് തത്തുല്യമായ 8 ബിറ്റ് ബൈനറി സംഖ്യ കണ്ടുപിടിക്കുന്നു. ബൈനറി സംഖ്യയ്ക്ക് 8 ബിറ്റുകൾ ഇല്ലെങ്കിൽ ഇടതുവശത്ത് ആവശ്യമായ പൂജ്യം ചേർത്ത് 8 ബിറ്റ് സംഖ്യ ആക്കുക. സംഖ്യയിലെ ഓരോ പൂജ്യത്തിനും പകരം ഒന്ന് എന്നും ഓരോ

ഒന്നിന് പകരം പൂജ്യം എന്നും മാറ്റി എഴുതിയാൽ ആ സംഖ്യയുടെ 1 ന്റെ പൂരകം ലഭിക്കും. ചില ബൈനറി സംഖ്യകളും അവയുടെ 1ന്റെ പൂരക പ്രതിനിധാനങ്ങളും താഴെ കൊടുത്തിരിക്കുന്നു.

| പൂർണസംഖ്യ | ബൈനറി സംഖ്യ | 1 ന്റെ പൂരക പ്രതിനിധാനം |
|---|---|---|
| +25 | 00011001 | 00011001 |
| - 25 | 00011001 | 11100110 |

സംഖ്യ നെഗറ്റീവ് ആണെങ്കിൽ അതിന്റെ തത്തുല്യമായ 8 ബിറ്റ് ബൈനറി രൂപത്തെ 1 ന്റെ പൂരകമായി പ്രതിനിധീകരിക്കുന്നു. എന്നാൽ സംഖ്യ പോസിറ്റീവ് ആണെങ്കിൽ സംഖ്യയുടെ 8 ബിറ്റ് പ്രതിനിധാനവും 1ന്റെ പൂരക പ്രതിനിധാനവും ഒരു പോലെയായിരിക്കും.

**ഉദാഹരണങ്ങൾ: -**

119 നെ 1ന്റെ പൂരക രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക.

119 ന്റെ 8 ബിറ്റ് ബൈനറി രൂപം = $(01110111)_2$

-119 ന്റെ 1ന്റെ പൂരക പ്രതിനിധാന രൂപം = $(10001000)_2$

+119 നെ 1ന്റെ പൂരക രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക

119 ന്റെ 8 ബിറ്റിൽ ഉള്ള ബൈനറി രൂപം = $(01110111)_2$

+119 ന്റെ 1ന്റെ പൂരക പ്രതിനിധാന രൂപം = $(01110111)_2$

*(സംഖ്യ പോസിറ്റീവ് ആയതിനാൽ 1 ന്റെ പൂരക പ്രതിനിധാനം കണ്ടുപിടിക്കേണ്ടതില്ല)*

**കുറിപ്പ്** : ഇത്തരം പ്രതിനിധീകരണത്തിൽ ഒന്നാമത്തെ ബിറ്റ് (MSB) 0 ആണെങ്കിൽ സംഖ്യ പോസിറ്റീവും MSB 1 ആണെങ്കിൽ സംഖ്യ നെഗറ്റീവും ആയിരിക്കും. 8 ബിറ്റ് പദദൈർഘ്യം കൊണ്ട് -127 (10000000) മുതൽ +127 (01111111) വരെ പ്രതിനിധാനം ചെയ്യാൻ കഴിയുന്നു. ഈ സംവിധാനത്തിലൂടെ പൂജ്യത്തിനെ +0 = 00000000 എന്നും -0 = 11111111 എന്നും രണ്ട് രീതിയിൽ പ്രതിനിധാനം ചെയ്യാം. പൊതുവായി, *n* ബിറ്റ് പദം കൊണ്ട് $2^n$-1 സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ കഴിയും (അതായത് - $(2^{n-1} - 1)$ മുതൽ $+(2^{n-1} - 1)$ വരെ.

### iii. 2 ന്റെ പൂരകം കൊണ്ടുള്ള പ്രതിനിധാനം (2's complement representation)

ഈ രീതിയിൽ, പൂർണസംഖ്യയുടെ കേവലവിലയ്ക്ക് തത്തുല്യമായ 8 ബിറ്റ് ബൈനറി സംഖ്യ കണ്ടുപിടിക്കുന്നു. സംഖ്യ നെഗറ്റീവ് ആണെങ്കിൽ 8 ബിറ്റ് ബൈനറിയുടെ 2 ന്റെ പൂരകരൂപത്തിൽ അതിനെ പ്രതിനിധാനം ചെയ്യുന്നു. എന്നാൽ സംഖ്യ പോസിറ്റീവ് ആണെങ്കിൽ 8 ബിറ്റ് ബൈനറി സംഖ്യ തന്നെയാണ് അതിന്റെ 2 ന്റെ പൂരക പ്രതിനിധാനം. ഒരു ബൈനറി സംഖ്യയുടെ 1 ന്റെ പൂരകത്തോട് 1 കൂട്ടിയാൽ അതിന്റെ 2 ന്റെ പൂരകം കിട്ടുന്നു.

ഉദാഹരണമായി നമുക്ക് $(10101)_2$ ന്റെ 2 ന്റെ പൂരകം കണ്ടുപിടിക്കാം.

$(00010101)_2$ ന്റെ 1 ന്റെ പൂരകം = $(11101010)_2$

$(10101)_2$ ന്റെ 2 ന്റെ പൂരകം = 11101010 + 1

= $(11011010)_2$

**ഉദാഹരണങ്ങൾ:**

-38 നെ 2ന്റെ പൂരക രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക.

38 ന്റെ 8 ബിറ്റിലുള്ള ബൈനറി രൂപം = $(00100110)_2$

-38 ന്റെ 2 ന്റെ പൂരക പ്രതിനിധാനം = 11011001+ 1

= $(11011010)_2$

+38 നെ 2ന്റെ പൂരക രൂപത്തിൽ പ്രതിനിധാനം ചെയ്യുക..

38 ന്റെ 8 ബിറ്റിലുള്ള ബൈനറി രൂപം = $(00100110)_2$

+38 ന്റെ 2 ന്റെ പൂരക പ്രതിനിധാനം = $(00100110)_2$ (സംഖ്യ പോസിറ്റീവ് ആയതിനാൽ 2ന്റെ പൂരക പ്രതിനിധാനം കണ്ടുപിടിക്കേണ്ടതില്ല)

**കുറിപ്പ്** : ഈ രീതിയിൽ ഒന്നാമത്തെ ബിറ്റ് (MSB) 0 ആണെങ്കിൽ സംഖ്യ പോസിറ്റീവും MSB 1 ആണെങ്കിൽ സംഖ്യ നെഗറ്റീവും ആയിരിക്കും. ഇവിടെ പൂജ്യം എന്ന പൂർണസംഖ്യ 00000000 എന്ന രീതിയിൽ മാത്രമേ പ്രതിനിധീകരിക്കുവാൻ കഴിയൂ. 8 ബിറ്റ് പദം കൊണ്ട് -128 (10000000) മുതൽ +127 (01111111) വരെ പ്രതിനിധാനം ചെയ്യാൻ കഴിയുന്നു. പൊതുവായി, *n* ബിറ്റ് പദം കൊണ്ട് $2^n$ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യുവാൻ കഴിയും. സംഖ്യകൾ $-(2^{n-1})$ മുതൽ $+(2^{n-1}-1)$. വരെ ആകുന്നു. ഈ രീതിയാണ് പൂർണസംഖ്യ പ്രതിനിധീകരിക്കുന്നതിന് സർവസാധാരണമായി ഉപയോഗിക്കുന്നത്. പട്ടിക 2.7 ൽ പൂർണസംഖ്യകളെ 8 ബിറ്റ് പദദൈർഘ്യത്തിൽ പ്രതിനിധീകരിക്കുന്നതിനുള്ള വിവിധ രീതികൾ താരതമ്യം ചെയ്യുന്നു.

| സവിശേഷത | ചിഹ്നവും മൂല്യവും | 1 ന്റെ പൂരകം | 2 ന്റെ പൂരകം | കുറിപ്പ് |
|---|---|---|---|---|
| പരിധി | -127 മുതൽ +127 വരെ | -127 മുതൽ +127 വരെ | -128 മുതൽ +127 വരെ | 2 ന്റെ പൂരകത്തിൽ പരിധി കൂടുതലാണ് |
| ആകെ സംഖ്യകൾ | 255 | 255 | 256 | |
| 0 ന്റെ പ്രതിനിധാനം | 2 രീതിയിലുള്ള പ്രതിനിധാനം | 2 രീതിയിലുള്ള പ്രതിനിധാനം | ഒരേയൊരു രീതിയിലുള്ള പ്രതിനിധാനം | പൂജ്യത്തെ 2 ന്റെ പൂരകത്തിൽ പ്രതി നിധീകരിക്കുന്നതിന് ഒരു അവ്യക്തതയും ഇല്ല. |
| പോസിറ്റീവ് സംഖ്യകളുടെ പ്രതിനിധാനം | സംഖ്യയ്ക്ക് തുല്യമായ 8 ബിറ്റ് ബൈനറി രൂപം | സംഖ്യയ്ക്ക് തുല്യമായ 8 ബിറ്റ് ബൈനറി രൂപം | സംഖ്യയ്ക്ക് തുല്യമായ 8 ബിറ്റ് ബൈനറി രൂപം | മൂന്നു രീതിയിലും ഒരേ പോലെയാണ് |
| നെഗറ്റീവ് സംഖ്യകളുടെ പ്രതിനിധാനം | ചിഹ്നം 1 ബിറ്റിലും മൂല്യം 7 ബിറ്റ് ബൈനറി രൂപത്തിലും പ്രതിനിധീകരി ക്കുന്നു | 8 ബിറ്റ് ബൈനറി രൂപത്തിലാക്കിയ ശേഷം അതിന്റെ 1 ന്റെ പൂരകം കാണുന്നു. | 8 ബിറ്റ് ബൈനറി രൂപത്തിലാക്കിയശേഷം അതിന്റെ 2 ന്റെ പൂരകം കാണുന്നു. | എല്ലാ നെഗറ്റീവ് സംഖ്യകളുടെയും MSB 1 ആകുന്നു |

*പട്ടിക 2.7 ൽ പൂർണ സംഖ്യകളുടെ 8 ബിറ്റ് പദദൈർഘ്യത്തിലുള്ള വിവിധ പ്രതിനിധാനങ്ങളുടെ താരതമ്യം*

*താഴെ കൊടുത്തിരിക്കുന്ന പട്ടികയിൽ 4 ബിറ്റ് പദദൈർഘ്യം ഉപയോഗിച്ച് പൂർണ സംഖ്യകളുടെ 3 രീതിയിലുള്ള പ്രതിനിധാനങ്ങൾ വിശദീകരിച്ചിരിക്കുന്നു.*

| സംഖ്യ | ചിഹ്നവും മൂല്യവും | 1 ന്റെ പൂരകം | 2 ന്റെ പൂരകം |
|---|---|---|---|
| -8 | സാധ്യമല്ല | സാധ്യമല്ല | 1000 |
| -7 | 1111 | 1000 | 1001 |
| -6 | 1110 | 1001 | 1010 |
| -5 | 1101 | 1010 | 1011 |
| -4 | 1100 | 1011 | 1100 |
| -3 | 1011 | 1100 | 1101 |
| -2 | 1010 | 1101 | 1110 |
| -1 | 1001 | 1110 | 1111 |
| 0 | 1000 അല്ലെങ്കിൽ 0000 | 0000 അല്ലെങ്കിൽ 1111 | 0000 |
| 1 | 0001 | 0001 | 0001 |
| 2 | 0010 | 0010 | 0010 |
| 3 | 0011 | 0011 | 0011 |
| 4 | 0100 | 0100 | 0100 |
| 5 | 0101 | 0101 | 0101 |
| 6 | 0110 | 0110 | 0110 |
| 7 | 0111 | 0111 | 0111 |

*പൂർണ സംഖ്യകളുടെ മൂന്നു രീതിയിലുള്ള പ്രതിനിധാനത്തിലും MSB സംഖ്യയുടെ ചിഹ്നം സൂചിപ്പിക്കുന്നു. ബിറ്റ് 1 ആണെങ്കിൽ സംഖ്യ നെഗറ്റീവും ബിറ്റ് 0 ആണെങ്കിൽ സംഖ്യ പോസിറ്റീവും ആണ്. തന്നിരിക്കുന്ന പദദൈർഘ്യം കൊണ്ട് സംഖ്യകളെ ഏറ്റവും കൂടുതൽ പ്രതിനിധീകരിക്കുവാൻ സാധിക്കുന്നത് 2 ന്റെ പൂരക രീതിയിലാണെന്ന് പട്ടികയിൽ കാണുന്നു. 4 പദദൈർഘ്യം ഉപയോഗിച്ചാൽ 7 നെക്കാൾ ചെറുതും +7 നെക്കാൾ വലുതും ആയ സംഖ്യകൾ പ്രതിനിധീകരിക്കാൻ ചിഹ്നവും മൂല്യവും രീതിയിലും 1 ന്റെ പൂരക രീതിയിലും സാധ്യമല്ല. അതുകൊണ്ട് 8 ബിറ്റ് പദദൈർഘ്യമുള്ള പ്രതിനിധാനം ഉപയോഗിക്കുന്നു. അതുപോലെ 2 ന്റെ പൂരക പ്രതിനിധാന രീതിയിൽ −8 മുതൽ +7 പരിധിക്ക് പുറത്തുള്ള സംഖ്യകൾ കൈകാര്യം ചെയ്യുന്നതിനായി 8 ബിറ്റ് ആവശ്യമാണ്.*

*8 ബിറ്റ് പദദൈർഘ്യം ഉപയോഗരീതിയിൽ −128 മുതൽ +127 വരെയുള്ള സംഖ്യകൾ 2 ന്റെ പൂരക രീതിയിൽ പ്രതിനിധീകരിക്കാം. എന്നാൽ മറ്റു രണ്ടു രീതികളായ 1 ന്റെ പൂരകത്തിലും ചിഹ്നവും മൂല്യത്തിലും −127 മുതൽ +127 വരെ പരിധിയുള്ള സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ സാധിക്കുകയുള്ളൂ. മേൽപ്പറഞ്ഞ പരിധിക്ക് പുറത്തുള്ള സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യാൻ നമ്മൾ 16 ബിറ്റ് ഉപയോഗിക്കുന്നു.*

## പൂരകം ഉപയോഗിച്ചുള്ള വ്യവകലനം (Subtraction using complements)

ഒരു ബൈനറി സംഖ്യയിൽ നിന്ന് മറ്റൊരു ബൈനറി സംഖ്യ വ്യവകലനം ചെയ്യുന്ന രീതി നമ്മൾ ചർച്ച ചെയ്തു. പക്ഷേ, ഈ രീതിയിലുള്ള ബൈനറി വ്യവകലനം, ഒരു ഇലക്ട്രോണിക്ക് സർക്യൂട്ട് രൂപകൽപ്പന ചെയ്ത് പ്രാവർത്തികമാക്കുക എന്നത് വളരെ സങ്കീർണവും പ്രയാസമുള്ളതുമാണ്. എന്നാൽ ബൈനറി സങ്കലനത്തിന്റെ ഇലക്ട്രോണിക് സർക്യൂട്ട് വളരെ ലളിതമാണ്. അതിനാൽ വ്യവകലനം സങ്കലനക്രിയ വഴി ചെയ്യുന്നതാകും നല്ലത്. വ്യവകലനം ബൈനറി സംഖ്യയുടെ പൂരകം എന്ന ആശയം ഉപയോഗിച്ച് സങ്കലന ക്രിയയിലൂടെയാണ് ചെയ്യുന്നത്. ഇതിന് രണ്ടു രീതികൾ ഉപയോഗിക്കുന്നു.

1 ന്റെ പൂരകം ഉപയോഗിച്ചുള്ള വ്യവകലനം

ഒരു വലിയ ബൈനറി സംഖ്യയിൽ നിന്ന് ഒരു ചെറിയ ബൈനറി സംഖ്യ കുറയ്ക്കുന്നതിനുള്ള ഘട്ടങ്ങൾ.

**ഘട്ടം 1:** ചെറിയ ബൈനറി സംഖ്യയുടെ ഇടതുവശത്ത് ആവശ്യമായ 0 ചേർത്ത് വലിയ ബൈനറി സംഖ്യയുടെ ബിറ്റുകളുടെ എണ്ണത്തിന് തുല്യമാക്കുക.

**ഘട്ടം 2:** ഏതു സംഖ്യകൊണ്ടാണോ കുറയ്ക്കേണ്ടത് അതിന്റെ 1 ന്റെ പൂരകം കാണുക. (ഇവിടെ ചെറിയ ബൈനറി സംഖ്യ)

**ഘട്ടം 3:** ഏതു സംഖ്യയിൽ നിന്നാണോ കുറക്കേണ്ടത് അതും, 1 ന്റെ പൂരകവും തമ്മിൽ കൂട്ടുക. (ഇവിടെ വലിയ ബൈനറി സംഖ്യ )

**ഘട്ടം 4:** തുകയോട് ശിഷ്ടം വരുന്ന ബിറ്റ് (ക്യാരി) കൂട്ടക്കിട്ടുന്നതാണ് ഉത്തരം.

**ഉദാഹരണം:** 1 ന്റെ പൂരക രീതി ഉപയോഗിച്ച് $(1010)_2$ ൽ നിന്നും $(100)_2$ കുറയ്ക്കുക.

ആദ്യമായി $(100)_2$ നെ നാല് ബിറ്റ് രൂപത്തിലേക്ക് മാറ്റുക $=(0100)_2$

ഏതു സംഖ്യയിൽ നിന്നാണോ കുറക്കേണ്ടത് അതും
$(0100)_2$ ന്റെ പൂരക സംഖ്യയും തമ്മിൽ കൂട്ടുക      1010 +

1011

MSB — 1 0101

0101 +

ക്യാരി കൂട്ടുക      1

ഉത്തരം      0110

MSB ഒഴിവാക്കി കൊണ്ട് ഉത്തരത്തോട് കൂട്ടുക.

**2 ന്റെ പൂരകം ഉപയോഗിച്ചുള്ള വ്യവകലനം**

ഒരു വലിയ ബൈനറി സംഖ്യയിൽ നിന്ന് ഒരു ചെറിയ ബൈനറി സംഖ്യ കുറയ്ക്കുന്നതിനുള്ള ഘട്ടങ്ങൾ.

**ഘട്ടം 1:** ചെറിയ ബൈനറി സംഖ്യയുടെ ഇടതുവശത്ത് ആവശ്യമായ 0 ചേർത്ത് വലിയ ബൈനറി സംഖ്യയുടെ ബിറ്റുകളുടെ എണ്ണത്തിന് തുല്യമാക്കുക.

**ഘട്ടം2:** ഏതു സംഖ്യകൊണ്ടാണോ കുറയ്ക്കേണ്ടത് അതിന്റെ 2 ന്റെ പൂരകം കാണുക. (ഇവിടെ ചെറിയ ബൈനറി സംഖ്യ )

**ഘട്ടം3:** ഏതു സംഖ്യയിൽ നിന്നാണോ കുറയ്ക്കേണ്ടത് അതും, 2 ന്റെ പൂരകവും തമ്മിൽ കൂട്ടുക. (ഇവിടെ വലിയ ബൈനറി സംഖ്യ )

**ഘട്ടം 4:** തുകയിൽ ശിഷ്ടം വരുന്ന ബിറ്റ് (ക്യാരി) ഒഴിവാക്കി കിട്ടുന്നതാണ് ഉത്തരം.

**ഉദാഹരണം:** 2 ന്റെ പൂരക രീതി ഉപയോഗിച്ച് $(1010)_2$ ൽ നിന്നും $(100)_2$ കുറയ്ക്കുക.

$(100)_2$ നെ നാല് ബിറ്റ് രൂപത്തിലേക്ക് മാറ്റുക 0100

$(0100)_2$ ന്റെ 2 ന്റെ പൂരകം കണ്ടുപിടിക്കുക 1011 +

1

1100

ഏതു സംഖ്യയിൽ നിന്നാണോ കുറക്കേണ്ടത് അതും 2 ന്റെ പൂരക സംഖ്യയും തമ്മിൽ കൂട്ടുക 1010 +

1100

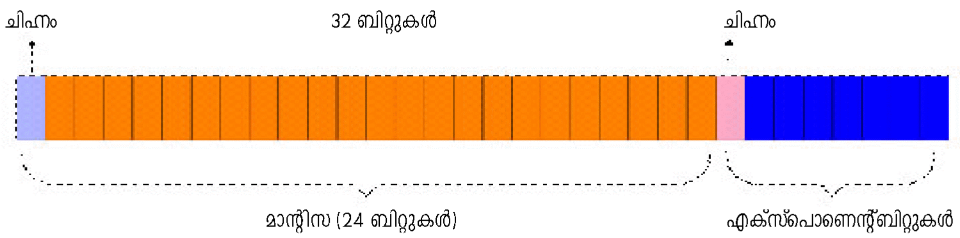
10110

ശിഷ്ടം ഒഴിവാക്കി കിട്ടുന്ന താണ് ഉത്തരം

ഉത്തരം 0110

## ബി. ഫ്ളോട്ടിംഗ് പോയിന്റ് സംഖ്യകളുടെ പ്രതിനിധാനം (Representation of floating point numbers)

ഒരു ഫ്ളോട്ടിംഗ് പോയിന്റ് സംഖ്യ അല്ലെങ്കിൽ രേഖീയ സംഖ്യയിൽ പൂർണസംഖ്യാഭാഗവും ഭിന്നക ഭാഗവും അടങ്ങിയിട്ടുണ്ട്. ഒരു രേഖീയ സംഖ്യയെ ഫ്ളോട്ടിംഗ് പോയിന്റ് എന്ന സവിശേഷമായ ചിഹ്നസമ്പ്രദായം ഉപയോഗിച്ച് എഴുതാവുന്നതാണ്. ഈ ചിഹ്നസമ്പ്രദായം ഉപയോഗിച്ച് എഴുതുമ്പോൾ ഏത് സംഖ്യയ്ക്കും മാന്റിസ, എക്സ്പൊണെന്റ് എന്നീ രണ്ട് ഭാഗങ്ങൾ ഉണ്ടാകും. ഉദാഹരണമായി 25.45 നെ $0.2545\text{x}10^2$ എന്നെഴുതാം. ഇതിൽ 0.2545 എന്നത് മാന്റിസയും കൃത്യങ്കം 2 എന്നത് എക്സ്പൊണെന്റുമാണ്. (ക്രമാനുസൃതമായ (Normalised) ഫ്ളോട്ടിംഗ് പോയിന്റ് പ്രതിനിധാനത്തിൽ മാന്റിസ 0.1നും 1നും ഇടയിലായിരിക്കും). അതുപോലെ 0.0035 എന്ന സംഖ്യ $-0.35\text{x}10^{-2}$ എന്ന് എഴുതാം. ഇവിടെ -0.35 എന്നത് മാന്റിസയും കൃത്യങ്കം -2 എന്നത് എക്സ്പൊണെന്റുമാണ്.




*ചിത്രം 2.3 ഫ്ളോട്ടിങ് പോയിന്റ് സംഖ്യകളുടെ പ്രതിനിധാനം*

32 ബിറ്റ് പദദൈർഘ്യമുള്ള കമ്പ്യൂട്ടറിൽ ഒരു രേഖീയ സംഖ്യ എങ്ങനെയാണ് പ്രതിനിധാനം ചെയ്യുന്നതെന്ന് നോക്കാം. ചിത്രം 2.3 ൽ കാണുന്നതുപോലെ, ഇതിൽ 24 ബിറ്റുകൾ മാന്റിസ

*രേഖീയസംഖ്യകളിൽ ബൈനറി അംശബിന്ദു മാന്റിസ, എക്സ്പൊണെന്റ് ഭാഗങ്ങളുടെ വിവരങ്ങൾ സൂക്ഷിക്കുന്നു. ബൈനറി അംശബിന്ദുവിന്റെ സ്ഥാനം സ്ഥിരമല്ലാത്തതിനാൽ മാന്റിസ എക്സ്പൊണെന്റ് എന്നിവയുടെ വിലകൾ സംഖ്യകൾ തോറും മാറുന്നു. മറ്റൊരു വിധത്തിൽപ്പറഞ്ഞാൽ അത് ഫ്ളോട്ട് ചെയ്യുകയാണ് (വെള്ളത്തിൽ പൊങ്ങിക്കിടക്കുന്നതുപോലെ) അതിനാൽ ഈ പ്രതിനിധാനത്തെ ഫ്ളോട്ടിംഗ് പോയിന്റ് പ്രതിനിധാനം എന്നറിയപ്പെടുന്നു.*

രേഖപ്പെടുത്താനും (അതിൽ ആദ്യത്തെ ബിറ്റ് മാന്റിസയുടെ ചിഹ്നത്തിനുവേണ്ടിയാണ്), 8 ബിറ്റുകൾ എക്സ്പൊണെന്റ് രേഖപ്പെടുത്താനും (അതിൽ ആദ്യത്തെ ബിറ്റ് എക്സ്പൊണെന്റിന്റെ ചിഹ്നത്തിനുവേണ്ടി) ഉപയോഗിക്കുന്നു. ദശാംശബിന്ദു മാന്റിസയുടെ ചിഹ്നം സൂചിപ്പിക്കുന്ന ബിറ്റിന്റെ വലത് ഭാഗത്താണെന്ന് അനുമാനിക്കുക. ദശാംശസ്ഥാനം സാങ്കല്പികമായതിനാൽ അത് രേഖപ്പെടുത്താൻ പ്രത്യേകമായി ബിറ്റുകൾ ആവശ്യമില്ല.

ഉദാഹരണമായി 25.45 എന്ന രേഖീയ സംഖ്യ മാന്റിസ എക്സ്പൊണെന്റ് രീതിയിൽ $0.2545 \times 10^2$ എന്ന് എഴുതാം. ഇവിടെ മാന്റിസയായ 0.2545 നെയും എക്സ്പൊണെന്റായ 2 നെയും ബൈനറി രൂപത്തിലേക്കു മാറ്റി അവയെ അതാതു സ്ഥാനങ്ങളിൽ രേഖപ്പെടുത്തുന്നു. മാന്റിസയും എക്സ്പൊണെന്റും രേഖപ്പെടുത്താൻ വ്യത്യസ്തങ്ങളായ മാനദണ്ഡങ്ങൾ ഉപയോഗിക്കുന്നു. പദദൈർഘ്യം മാറുന്നതിനനുസരിച്ച് മാന്റിസയും എക്സ്പൊണെന്റും രേഖപ്പെടുത്താൻ ഉപയോഗിക്കുന്ന ബിറ്റുകളുടെ എണ്ണത്തിലും മാറ്റം ഉണ്ടാകും.

## 2.4.2 അക്ഷരങ്ങളുടെ പ്രതിനിധാനം (Representation of characters)

കംപ്യൂട്ടറിന്റെ മെമ്മറിയിൽ സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യുന്നത് എങ്ങനെയെന്ന് നമ്മൾ കണ്ടു. അതുപോലെ അക്ഷരങ്ങളെ (Characters) പ്രതിനിധാനം ചെയ്യുന്നതിന് വ്യത്യസ്തങ്ങളായ സമ്പ്രദായങ്ങളുണ്ട്. അവയിൽ ചിലതിനെക്കുറിച്ച് ചുവടെ പ്രതിപാദിക്കുന്നു.

### എ. ആസ്കി (ASCII)

കമ്പ്യൂട്ടറിന്റെ മെമ്മറിയിൽ 7 ബിറ്റുകൾ ഉപയോഗിച്ച് ഓരോ അക്ഷരവും പ്രതിനിധാനം ചെയ്യാൻ ഉപയോഗിക്കുന്ന ASCII (ആസ്കി) കോഡ് American Standard Code for Information Interchange (അമേരിക്കൻ സ്റ്റാൻഡേർഡ് കോഡ് ഫോർ ഇൻഫർമേഷൻ ഇന്റർചേഞ്ച്) എന്നതിന്റെ ചുരുക്ക രൂപമാണ്. അമേരിക്കൻ സർക്കാർ അംഗീകരിച്ച ആസ്കികോഡ് വ്യാപകമായി സ്വീകരിക്കപ്പെട്ടു കഴിഞ്ഞു. ഇതിൽ ഓരോ അക്ഷരത്തിനും വത്യസ്ത പൂർണസംഖ്യ നിശ്ചയിച്ചിരിക്കുന്നു. ആസ്കി കോഡ് എന്ന് വിളിക്കുന്ന ഈ പൂർണസംഖ്യ മെമ്മറിയിൽ സൂക്ഷിക്കുന്നതിനായി ബൈനറി സംഖ്യയിലേക്ക് പരിവർത്തനം ചെയ്യുന്നു. ഉദാഹരണത്തിന് A എന്ന അക്ഷരത്തിന്റെ ആസ്കി കോഡ് 65 ആകുന്നു. ഇതിന് തുല്യമായ 7 ബിറ്റ് ബൈനറി 1000001 ആണ്. 7 ബിറ്റുകൾ കൊണ്ട് വത്യസ്തങ്ങളായ 128 സംയോഗങ്ങൾ (Unique combination) സൃഷ്ടിക്കാനാകും. ആയതിനാൽ 7 ബിറ്റ് ആസ്കി ഉപയോഗിച്ച് 128 അക്ഷരങ്ങളുടെ കോഡുകൾ ഉണ്ടാക്കാം.

ഓരോ അക്ഷരത്തിനും 8 ബിറ്റ് ഉപയോഗിക്കുന്ന ഇതിന്റെ മറ്റൊരു പതിപ്പിനെ ആസ്കി 8 അഥവാ എക്സ്റ്റൻഡ് ആസ്കി (Extended ASCII) എന്ന് വിളിക്കുന്നു. 8 ബിറ്റ് ആസ്കി കൊണ്ട് 256 വ്യത്യസ്താക്ഷരങ്ങളുടെ കോഡുകൾ ഉണ്ടാക്കാം. ഉദാഹരണമായി A എന്ന അക്ഷരത്തെ 01000001 എന്നും B എന്ന അക്ഷരത്തെ 01000010 എന്നും കോഡ് ചെയ്യപ്പെടുന്നു. സാധാരണ കീബോർഡിലെ മുഴുവൻ അക്ഷരങ്ങൾക്കും കോഡ് നൽകുവാൻ ആസ്കി 8 ന് കഴിയുന്നു.

### ബി. എബ്സിഡിക് (EBCDIC)

എക്സ്റ്റൻഡ് ബൈനറി കോഡഡ് ഡെസിമൽ ഇന്റർചേഞ്ച് കോഡ് (Extended Binary Coded Decimal Interchange Code) എന്നതിന്റെ ചുരുക്ക രൂപമാണിത്. ഇന്റർനാഷണൽ ബിസിനസ് മെഷീൻ (ഐ.ബി.എം) നിർമ്മിക്കുന്ന കമ്പ്യൂട്ടറുകളിൽ, ആസ്കിയെ പോലെ ഇതിലും 8 ബിറ്റ് കോഡ് ഉപയോഗിക്കുന്നു. ഇതുപയോഗിച്ച് 256 അക്ഷരങ്ങൾക്ക് കോഡ് നൽകാനാവും. ആസ്കിയിൽ കോഡ് ചെയ്യപ്പെട്ട ഡാറ്റ എബ്സിഡിക് കോഡ് ഉപയോഗിക്കുന്ന കമ്പ്യൂട്ടറിൽ ഉപയോഗിക്കണമെങ്കിൽ ആസ്കി കോഡിൽ നിന്ന് എബ്സിഡിക് കോഡിലേക്ക് മാറ്റേണ്ടതുണ്ട്. അതുപോലെ, എബ്സിഡിക് കോഡ് ഉപയോഗിച്ചുണ്ടാക്കിയ ഡാറ്റ ആസ്കി കമ്പ്യൂട്ടറിൽ ഉപയോഗിക്കണമെങ്കിൽ, ആസ്കിയിലേക്കും മാറ്റേണ്ടതുണ്ട്.

### സി. ഇസ്കി (ISCII)

ഇന്ത്യൻ സ്റ്റാൻഡേർഡ് കോഡ് ഫോർ ഇൻഫർമേഷൻ ഇന്റർചേഞ്ച് (Indian Standard Code for Information Interchage) അല്ലെങ്കിൽ ഇന്ത്യൻ സ്ക്രിപ്റ്റ് കോഡ് ഫോർ ഇൻഫർമേഷൻ ഇന്റർചേഞ്ച് (Indian Script Code for Information Interchage) എന്നതിന്റെ ചുരുക്കരൂപമാണിത്. വിവിധ ഇന്ത്യൻഭാഷകളിലെ അക്ഷരങ്ങളുടെ എൻകോഡിംഗ് (Encoding) വ്യവസ്ഥയാണിത്. 8 ബിറ്റ് ഉപയോഗിച്ചാണ് ഇസ്കി ഡാറ്റ പ്രതിനിധാനം ചെയ്യുന്നത്. 1986 ൽ ഇലക്ട്രോണിക് വകുപ്പിന് കീഴിലുള്ള നിലവാരം നിശ്ചയിക്കൽ സമിതി ചിട്ടപ്പെടുത്തിയ ഈ വ്യവസ്ഥ ബ്യൂറോ ഓഫ് ഇന്ത്യൻ സ്റ്റാൻഡേർഡ്സ് (BIS) അംഗീകരിച്ചതാണ്. ഇസ്കിക്ക് പകരം യൂനിക്കോഡാണ് ഇപ്പോൾ ഉപയോഗിക്കുന്നത്.

### ഡി. യൂണികോഡ് (Unicode)

8 ബിറ്റുകൾ ഉപയോഗിക്കുന്ന ആസ്കിക്ക് 256 അക്ഷരങ്ങൾ മാത്രമേ പ്രതിനിധാനം ചെയ്യാനാകൂ. ലോകം മുഴുവനുമുള്ള ലിഖിതഭാഷകളിലെ അക്ഷരങ്ങളെയും ചിഹ്നങ്ങളേയും പ്രതിനിധാനം ചെയ്യാൻ ഇത് മതിയാകില്ല. ഈ പ്രശ്നം പരിഹരിക്കാനാണ് യൂണിക്കോഡ് വികസിപ്പിച്ചെടുത്തത്. ആഗോളവും കാര്യക്ഷമവും നിലവാരമുള്ളതും ആയ അക്ഷരങ്ങളുടെ എൻകോഡിങ് രീതിയാണ് ഇതിന്റെ ലക്ഷ്യം. ഏത് ഭാഷയായാലും ഏത് പ്ലാറ്റ്ഫോമായാലും (Platform) അവയ്ക്കെല്ലാം വ്യത്യസ്തമായ ഒരക്കം ഇത് നൽകുന്നു.

യൂണിക്കോഡിൽ മൗലികമായി 16 ബിറ്റുകളാണ് ഉപയോഗിക്കുന്നത്. അതിന് 65,536 അക്ഷരങ്ങൾ പ്രതിനിധീകരിക്കാൻ കഴിയും. യൂണിക്കോഡ് കൺസോർഷ്യം എന്ന ലാഭേച്ഛയില്ലാത്ത സംഘടനയാണ് ഇത് ചിട്ടപ്പെടുത്തുന്നത്. കൺസോർഷ്യം 1991 ൽ ആദ്യപതിപ്പായ 1.0.0 പ്രസിദ്ധീകരിച്ചു. അതിനെ അടിസ്ഥാനമാക്കി നിലവാരം മെച്ചപ്പെടുത്തുന്നതിനുള്ള ശ്രമം തുടരുകയാണ്. ഈ കാലയളവിൽ യൂണികോഡ് ഉപയോഗിക്കുന്നത് 16ൽ അധികം ബിറ്റുകളാണ്. അതിനാൽ ധാരാളം അക്ഷരങ്ങളെ പ്രതിനിധാനം ചെയ്യാൻ അതിന് സാധിക്കും. ലോകത്തിലെ എല്ലാ ലിഖിത ഭാഷകളുടെയും അക്ഷരങ്ങളെ പ്രതിനിധാനം ചെയ്യുവാൻ യൂണിക്കോഡിന് സാധിക്കുന്നു.

## 2.4.3 ശബ്ദ, ചിത്രം, വീഡിയോ എന്നിവയുടെ പ്രതിനിധാനം (Representation of audio, image and video)

ഇതിന് മുമ്പുള്ള ഭാഗത്തിൽ അക്കങ്ങളും അക്ഷരങ്ങളും ഉൾപ്പെട്ട വിവരങ്ങൾ കമ്പ്യൂട്ടറിൽ പ്രതിനിധാനം ചെയ്യുന്ന വിധവും അവയുടെ വ്യത്യസ്ത മാനദണ്ഡങ്ങളും നാം പരിചയപ്പെട്ടു. ഡിജിറ്റൽ കമ്പ്യൂട്ടറുകളുടെ സഹായത്തോടെ നിത്യജീവിതത്തിലെ പ്രശ്നങ്ങൾ കൈകാര്യം ചെയ്യുമ്പോൾ മിക്കപ്പോഴും അക്കങ്ങളോ അക്ഷരങ്ങളോ അല്ലാത്ത വിവരങ്ങൾ രേഖപ്പെടുത്തുകയോ പ്രോസസ്സ് ചെയ്യേണ്ടതായോ വരാം. അക്കങ്ങളെയും അക്ഷരങ്ങളെയും പോലെ ശബ്ദം, ചിത്രം, വീഡിയോ എന്നിവയിലും ധാരാളം വിവരങ്ങൾ അടങ്ങിയിട്ടുണ്ട്. ഇവ സംഭരിക്കുന്നതിനുള്ള വിവിധ ഫയൽ ഘടനകളെക്കുറിച്ച് നമുക്ക് ചർച്ച ചെയ്യാം.

### ഡിജിറ്റൽ ശബ്ദം, ചിത്രം, വീഡിയോ എന്നിവയുടെ ഫയൽ ഘടനകൾ (Digital audio, image and video file formats)

ശബ്ദം, ചിത്രം, വീഡിയോ എന്നിങ്ങനെയുള്ള മൾട്ടിമീഡിയ ഡാറ്റ വ്യത്യസ്ത ഫയൽ ഘടനകളിലാണ് സംഭരിക്കുന്നത്. ഡാറ്റയുടെ വലുപ്പം കുറയ്ക്കുന്നതിനും ചുരുക്കുന്നതിനും വിവിധ കെട്ടുകളാക്കുന്നതിനും വിവിധ സമീപനരീതികൾ ഉപയോഗിക്കുമ്പോൾ അവ വ്യത്യസ്ത ഫയൽഘടനയ്ക്ക് കാരണമാകുന്നു. ഉദാഹരണത്തിന് ഒരു ചിത്രം സാധാരണനിലയിൽ ജോയിന്റ് പിക്ചർ എക്സ്പേർട്ട്സ് ഗ്രൂപ്പ് (ജെപെഗ് - JPEG) ഫയൽ ഘടനയിലാണ് സംഭരിക്കുന്നത്. ഈ ചിത്രത്തിന്റെ ഫയലിൽ തലക്കെട്ട് (Header) വിവരങ്ങളും ചിത്രത്തിന്റെ (Image) ഡാറ്റയും അടങ്ങിയിരിക്കുന്നു. ഫയലിന്റെ പേര്, വലുപ്പം, പരിഷ്കരിച്ച ഡാറ്റ, ഫയൽഘടന മുതലായ വിവരങ്ങൾ തലക്കെട്ട് ഭാഗത്താണ് സംഭരിക്കുന്നത്. പിക്സലുകളുടെ തീവ്രതയെക്കുറിച്ചുള്ള വിവരങ്ങൾ ഡാറ്റ ഭാഗത്തും ശേഖരിക്കുന്നു.

ഫയലിന്റെ വലുപ്പം കുറയ്ക്കുന്നതിന് ഡാറ്റ ചുരുക്കിയോ അല്ലാതെയോ സംഭരിക്കാം. സാധാരണനിലയിൽ ചിത്രം ഡാറ്റയെ ചുരുക്കിയാണ് സംഭരിക്കുന്നത്. എന്താണ് ചുരുക്കൽ (Compression) എന്ന് നോക്കാം. 400x400 പിക്സൽ വലുപ്പമുള്ള, കറുപ്പ് നിറമുള്ള ഒരു ചിത്രം ഉദാഹരണമായി എടുക്കാം. 1,60,000 (400x400) പിക്സലിലും കറുപ്പ്, കറുപ്പ്, ......കറുപ്പ് എന്നിങ്ങനെ ആവർത്തിച്ച് സംഭരിക്കാം. ഇത് ചുരുക്കാതെയുള്ള രൂപമാണ്. അതേസമയം, കറുപ്പ് എന്ന് ഒരു തവണ രേഖപ്പെടുത്തുകയും 1,60,000 തവണ ആവർത്തനം എന്നും രേഖപ്പെടുത്തുന്നതാണ് ചുരുക്കി സംഭരിക്കൽ. ചുരുക്കലിനായി ഇത്തരം നിരവധി രീതികൾ ഉപയോഗിക്കാറുണ്ട്. ബിറ്റ്മാപ് (BMP), ടാഗ്ഡ് ഇമേജ് ഫയൽ ഫോർമാറ്റ് (TIFF), ഗ്രാഫിക്സ് ഇന്റർചേഞ്ച് ഫോർമാറ്റ് (GIF), പോർട്ടബിൾ പബ്ലിക് നെറ്റ്വർക്ക് ഗ്രാഫ് (PNG) തുടങ്ങി വിവിധ തരത്തിലുള്ള ഫയൽ ഘടനകളിൽ ചിത്രങ്ങൾ ഉപയോഗത്തിനനുസരിച്ച് സംഭരിക്കപ്പെടുന്നു.

ചിത്രത്തിന്റെ കാര്യത്തിൽപ്പറഞ്ഞ തലക്കെട്ട് ഫയൽ വിവരങ്ങൾ, ശബ്ദം, വീഡിയോ എന്നീ ഫയലുകൾക്കും ബാധകമാണ്. WAV, MP3, MIDI, AIFF മുതലായ വ്യത്യസ്ത ഫയൽ ഘടനകളിൽ ഡിജിറ്റൽ ശബ്ദ ഡാറ്റ സംഭരിക്കാൻ കഴിയും. ഡിജിറ്റൽ ശബ്ദഡാറ്റ സംഭരിക്കുന്നതിന് ഒരു ശബ്ദ ഫയൽ ഘടന വിവരിക്കുന്നുണ്ട്. ചില സമയങ്ങളിൽ ഇത് കണ്ടെയ്നർ ഫോർമാറ്റ് (Container Format) എന്ന് സൂചിപ്പിക്കാറുണ്ട്. ഉദാഹരണമായി WAV ചുരുക്കാത്ത ശബ്ദവും, MP3 ഫയലുകളിൽ ചുരുക്കിയ ശബ്ദവുമാണ് ഉൾക്കൊള്ളുക. സംശ്ലേഷണം (synchronous) ചെയ്ത സംഗീത

**സ്വയം വിലയിരുത്താം**

1. -80 നെ ചിഹ്നവും മൂല്യവും രൂപത്തിൽ പ്രതിനിധാനം ചെയ്താൽ അതിന്റെ MSB ഏതാണ്?
2. 28.756 നെ മാന്റിസ എക്സ്പൊണെന്റ് രൂപത്തിൽ എഴുതുക.
3. ASCII യുടെ പൂർണരൂപം എഴുതുക.
4. 60 നെ 1 ന്റെ പൂരകമായി പ്രതിനിധാനം ചെയ്യുക.
5. യൂണികോഡ് നിർവചിക്കുക.
6. ഏതെങ്കിലും രണ്ട് ചിത്രഫയൽ ഘടനകൾ എഴുതുക.

ഡാറ്റ ശേഖരിച്ചുവയ്ക്കുന്ന സംവിധാനമാണ് MIDI (Musical Instrument Digital Interface). അതുപോലെ AVI (Audio Video Interleave) എന്നത് വീഡിയോഫയൽ ശേഖരിച്ചുവയ്ക്കുന്ന മറ്റൊരു സംവിധാനമാണ്. MP3, JPEG-2, WMV എന്നീ ഫയൽ ഘടനകൾ ശബ്ദം, വീഡിയോ എന്നിവ സംഭരിച്ചുവയ്ക്കുന്നതിനും ഒരേ സമയം പ്രവർത്തിപ്പിക്കുന്നതിനും ഉപയോഗിക്കുന്നു.

## 2.5 ബൂളിയൻ ബീജ ഗണിതത്തിന് ഒരാമുഖം (Introduction to Boolean algebra)

ശരി അല്ലെങ്കിൽ തെറ്റ് എന്ന് ഉത്തരം നൽകേണ്ട നിരവധി സന്ദർഭങ്ങൾ നമ്മുടെ ജീവിതത്തിൽ ഉണ്ടാകാറുണ്ട്. അതുപോലെ നമ്മുടെ ചിന്തയുടെ ഒരു വലിയ ഭാഗം ശരി അല്ലെങ്കിൽ തെറ്റ് എന്ന് ഉത്തരം പറയേണ്ടവയാണ്. ഇപ്രകാരം സത്യം കണ്ടെത്തുന്നതിനെ വ്യക്തി വിചിന്തനം (ഹ്യൂമൺ റീസണിങ് - Human Reasoning) അഥവാ യുക്തി വിചിന്തനം ( Logical Reasoning) എന്ന് പറയുന്നു. ശരി അല്ലെങ്കിൽ തെറ്റ് എന്നത് സംഖ്യാപരമായി ഒന്നോ പൂജ്യമോ ആകാം. ഈ രണ്ട് മൂല്യങ്ങളെ ദ്വന്ദമൂല്യങ്ങൾ (Binary Values) അല്ലെങ്കിൽ ബൂളിയൻ മൂല്യങ്ങൾ (Boolean Values) എന്ന് പറയുന്നു. 1 അല്ലെങ്കിൽ 0 എന്നിവയാൽ പ്രതിനിധാനം ചെയ്യപ്പെടുന്ന ചരങ്ങളുമായി (variables) ബന്ധപ്പെട്ട ക്രിയകൾ ചെയ്യുന്ന യുക്തിപരമായ ബീജഗണിതശാസ്ത്രശാഖയാണ് ബൂളിയൻ ബീജഗണിതം (Boolean Algebra). യുക്തിയും ഗണിതശാസ്ത്രവും തമ്മിൽ ബന്ധം സ്ഥാപിച്ച ബ്രിട്ടീഷ് ഗണിതശാസ്ത്രജ്ഞനായ ജോർജ് ബൂളിനെ ആദരിച്ചു കൊണ്ടാണ് ഈ പേര് നൽകിയിരിക്കുന്നത്. അദ്ദേഹത്തിന്റെ An Investigation of the Law of Thought എന്ന വിപ്ലവകരമായ പ്രബന്ധമാണ് ബൂളിയൻ ആൾജിബ്രയുടെ ഉൽപ്പത്തിക്ക് വഴിയൊരുക്കിയത്.

*ചിത്രം 2.4: ജോർജ്ബൂൾ (1815 1864)*

### 2.5.1 ദ്വന്ദമൂല്യമുള്ള അളവുകൾ (Binary Valued Quantities)

താഴെ പറയുന്നവ നോക്കുക.

1. ഞാൻ ഒരു കുട എടുക്കുന്നതല്ലേ നല്ലത്?
2. താങ്കളുടെ പേന എനിക്ക് തരാൻ സമ്മതമാണല്ലോ?

3. ജോർജ് ബൂൾ ഒരു ബ്രിട്ടീഷ് ഗണിതശാസ്ത്രജ്ഞനാണ്.
4. കേരളം ഇന്ത്യയിലെ ഏറ്റവും വലിയ സംസ്ഥാനമാണ്.
5. ഇന്നലെ ഹാജരാകാതിരുന്നത് എന്തുകൊണ്ട് ?
6. ബൂളിയൻ ബീജഗണിതത്തെക്കുറിച്ച് നിങ്ങളുടെ അഭിപ്രായം എന്താണ്?

1 ഉം 2 ഉം വാക്യങ്ങൾ അതെ (YES) അല്ലെങ്കിൽ അല്ല (NO) എന്ന് ഉത്തരം പറയാവുന്ന ചോദ്യങ്ങളാണ്. ഇത്തരം സന്ദർഭങ്ങളെ ബൈനറി തീരുമാനങ്ങൾ എന്നും ഇതിന്റെ ഉത്തരങ്ങളെ ബൈനറി മൂല്യങ്ങൾ (Binary Values) എന്നും വിളിക്കുന്നു. മൂന്നാമത്തെ വാക്യത്തിന്റെ ഉത്തരം ശരി (TRUE) എന്നും നാലാമത്തേതിന്റെ തെറ്റ് (FALSE) എന്നുമാണ്. എന്നാൽ അഞ്ചും ആറും ചോദ്യങ്ങൾക്ക് ഇങ്ങനെ ഉത്തരം പറയാനാകില്ല. TRUE, FALSE എന്ന് ഉത്തരം നൽകാവുന്ന വാക്യങ്ങൾ യുക്തി പ്രസ്താവനകൾ (Logical Statements) അല്ലെങ്കിൽ ട്രൂത്ത് ഫങ്ക്ഷൻ (Truth function) എന്നും അതിന്റെ ഉത്തരമായി നൽകുന്ന TRUE, FALSE എന്നിവയെ ലോജിക്കൽ സ്ഥിരമൂല്യങ്ങൾ (Logical Constants) അല്ലെങ്കിൽ ബൈനറി മൂല്യങ്ങൾ (Binary Values) എന്നും പറയുന്നു. ശരി എന്നത് 1 ഉം തെറ്റ് എന്നത് 0 ഉം ആണ്. ഈ ലോജിക്കൽ കോൺസ്റ്റന്റുകൾ കൈകാര്യം ചെയ്യുന്ന വേരിയബിളിനെ ലോജിക്കൽ വേരിയബിൾ അല്ലെങ്കിൽ ബൂളിയൻ വേരിയബിൾ എന്ന് പറയുന്നു.

### 2.5.2 ബൂളിയൻ ഓപറേറ്ററുകളും ലോജിക്കൽ ഗേറ്റുകളും (Boolean operators and logic gates)

കമ്പ്യൂട്ടറിലേക്ക് നമ്മൾ നൽകുന്ന വിവരങ്ങൾ 1, 0 എന്നിവയുടെ ഗണങ്ങളായി മാറ്റേണ്ടതുണ്ട്. കമ്പ്യൂട്ടറിലെ എല്ലാ ഡാറ്റയും, വിവരങ്ങളും, ക്രിയകളും 0, 1 എന്നീ അക്കങ്ങളിലാണ് പ്രതിനിധാനം ചെയ്യുന്നത്. ബൂളിയൻ മൂല്യങ്ങൾ അടിസ്ഥാനമാക്കി നിർവഹിക്കപ്പെടുന്ന ഈ പ്രവർത്തങ്ങളെ ബൂളിയൻ പ്രവർത്തനങ്ങൾ (Boolean Operation) എന്ന് വിളിക്കുന്നു. നമുക്കറിയാവുന്നതുപോലെ, ഈ ക്രിയകൾ ചെയ്യുവാൻ ഓപ്പറേറ്ററുകൾ ആവശ്യമാണ്. ഇത്തരം ഓപ്പറേറ്ററുകളെ ബൂളിയൻ ഓപ്പറേറ്ററുകൾ അല്ലെങ്കിൽ ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ എന്ന് വിളിക്കുന്നു. ബൂളിയൻ ബീജഗണിതത്തിൽ മൂന്ന് അടിസ്ഥാന ഓപ്പറേറ്ററുകളാണ് ഉള്ളത്. ഈ ഓപ്പറേറ്ററുകളും ബീജഗണിതത്തിൽ അവയുടെ പ്രവർത്തനങ്ങളും ചുവടെ ചേർത്തിരിക്കുന്നു.

OR → യുക്തിപരമായ സങ്കലനം (Logical Addition)

AND → യുക്തിപരമായ വ്യവകലനം (Logical Multiplication)

NOT → ലോജിക്കൽ നിഷേധം (Logical Negation)

ആദ്യത്തെ രണ്ട് ഓപ്പറേറ്ററുകളോടൊപ്പം രണ്ട് ഓപ്പറന്റുകളും, മൂന്നാമത്തേതിനോടുകൂടി ഒരു ഓപ്പറന്റും ഉപയോഗിക്കുന്നു. ഇവിടെ ഓപ്പറന്റുകൾ എല്ലായ്പ്പോഴും ബൂളിയൻ വേരിയബിൾ അല്ലെങ്കിൽ കോൺസ്റ്റന്റുകൾ ആയിരിക്കും. ഇത്തരം ഓപ്പറേഷനുകളുടെ ഉത്തരം ഒന്നുകിൽ TRUE (1) അല്ലെങ്കിൽ FALSE (0) ആയിരിക്കും.

ഇലക്ട്രോണിക് സർക്യൂട്ടുകൾ ഉപയോഗിച്ചാണ് കമ്പ്യൂട്ടറുകൾ ഇത്തരം പ്രവർത്തങ്ങൾ നിർവ ഹിക്കുന്നത് അവയെ ലോജിക്കൽ സർക്യൂട്ട് (Logical circuite) എന്ന് വിളിക്കുന്നു. ഓരോ വ്യക്തിഗത യൂണിറ്റുകളാലും നിർമ്മിക്കപ്പെടുന്ന ലോജിക്കൽ സർക്യൂട്ടിനെ ഗേറ്റ് (Gate) എന്ന് പറയുന്നു. ഒരു

ഗേറ്റ് ഒരു ബൂളിയൻ പ്രവർത്തനത്തെ പ്രതിനിധാനം ചെയ്യുന്നു. ഒന്നോ അതിലധികമോ ലോജിക്കൽ ഇൻപുട്ടുകളിൽ ലോജിക്കൽ പ്രവർത്തനം നടത്താനും ഒരേ ഒരു ലോജിക്കൽ ഔട്ട്പുട്ട് നൽകാനും കഴിയുന്ന ഭൗതികസാമഗ്രിയാണ് ഒരു ലോജിക് ഗേറ്റ് (Logic Gate). ഇലക്ട്രോണിക് സ്വിച്ചുകളായി പ്രവർത്തിക്കുന്ന ഡയോഡുകളോ ട്രാൻസിസ്റ്ററുകളോ ഉപയോഗിച്ചാണ് ലോജിക്കൽ ഗേറ്റുകൾ നിർമ്മിക്കുന്നത്. ബൂളിയൻ ബീജഗണിതത്തിലെ അടിസ്ഥാനപരമായ മൂന്ന് പ്രവർത്തനങ്ങളായ OR, AND, NOT എന്നിവ നിർവഹിക്കുന്നത് യഥാക്രമം OR, AND, NOT എന്നീ ലോജിക്കൽ ഗേറ്റുകൾ ഉപയോഗിച്ചാണ്.

## a. OR ഓപ്പറേറ്ററും OR ഗേറ്റും

ജീവിതത്തിലെ ഒരു സന്ദർഭം നമുക്ക് നോക്കാം. എപ്പോഴാണ് നാം കുട ഉപയോഗിക്കുക? മഴ ഉള്ളപ്പോൾ മാത്രമാണോ? വെയിലുള്ളപ്പോഴും നാം കുട ഉപയോഗിക്കാറുണ്ടല്ലോ. ഈ രണ്ട് സാഹചര്യങ്ങളും ചേർത്ത് ഒരു സംയുക്ത പ്രസ്താവന ഇങ്ങനെ ഉണ്ടാക്കാം. 'മഴ ഉള്ളപ്പോഴോ വെയിലുള്ളപ്പോഴോ നാം കുട ഉപയോഗിക്കുന്നു.' (If it is raining or if it is sunny, we use an umbrella). ഈ പ്രസ്താവനയിൽ OR ന്റെ ഉപയോഗം ശ്രദ്ധിക്കുക. പട്ടിക 2.8 ൽ ഈ പ്രസ്താവനയുടെ വിശകലനം കാണാം. മേൽക്കൊടുത്തിരിക്കുന്ന ഉദാഹരണത്തിൽ 'കുടയുടെ ആവശ്യം' എന്ന യുക്തി വിചിന്തനത്തിന് ബൂളിയൻ OR പ്രവർത്തനവുമായി വളരെയധികം സാദൃശ്യമുണ്ട്.

| മഴ | വെയിൽ | കുടയുടെ ആവശ്യം |
|---|---|---|
| ഇല്ല | ഇല്ല | ഇല്ല |
| ഇല്ല | ഉണ്ട് | ഉണ്ട് |
| ഉണ്ട് | ഇല്ല | ഉണ്ട് |
| ഉണ്ട് | ഉണ്ട് | ഉണ്ട് |

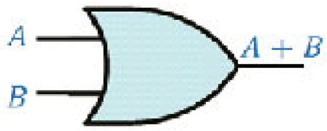
*പട്ടിക 2.8 ലോജിക്കൽ OR ന്റെ പ്രവർത്തനം*

OR ഓപ്പറേറ്റർ ലോജിക്കൽ സങ്കലനം നടത്തുന്നു. ഇതിനായി + ചിഹ്നം ഉപയോഗിക്കുന്നു. അതിനാൽ A+B എന്ന പദപ്രയോഗം വായിക്കേണ്ടത് A അല്ലെങ്കിൽ B (A OR B) എന്നാണ്. താഴെ കൊടുത്തിരിക്കുന്ന പട്ടിക 2.9 OR പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിളാണ്. A, B എന്നിവ ഇൻപുട്ടുകളും (ഓപ്പറന്റുകൾ), A+B ഔട്ട്പുട്ടും (ഉത്തരം) ആണെന്ന് അനുമാനിക്കുക. ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആണെങ്കിൽ ഔട്ട്പുട്ട് 1 ആയിരിക്കുമെന്ന് ട്രൂത്ത് ടേബിളിൽ വ്യക്തമാക്കാം.

| A | B | A + B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

*പട്ടിക 2.9 OR പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ*

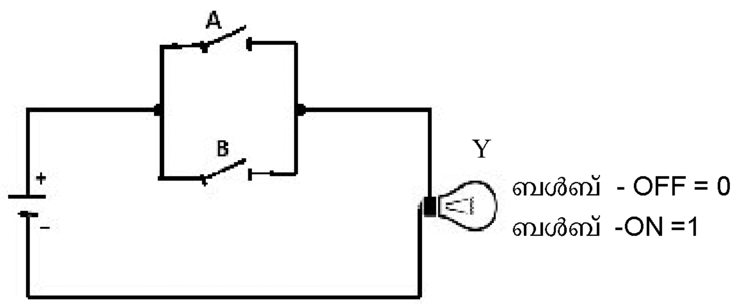
ബൂളിയൻ പ്രവർത്തനവും അതിന്റെ ഫലവും കാണിക്കുന്ന പട്ടികയാണ് ട്രൂത്ത് ടേബിൾ. ഒരു നിശ്ചിത പ്രവർത്തനത്തിന്റെ സാധ്യമായ എല്ലാ ഇൻപുട്ടുകളും അതനുസരിച്ചുള്ള ഫലവും ഈ പട്ടികയിൽ ഉണ്ടായിരിക്കും. സാധാരണ നിലയിൽ ഇത്തരം പ്രവർത്തനങ്ങളിൽ ഓപ്പറന്റ് വേരിയബിളുകളും, ഓപ്പറേറ്ററുകളും ഉണ്ടായിരിക്കും. ഓപ്പറന്റുകളും ഓപ്പറേറ്ററുകളും ചേർന്നതാണ് ബൂളിയൻ പദപ്രയോഗം (Boolean Expression). m ഓപ്പറന്റുകളും n ഓപ്പറേറ്ററുകളുള്ള ഒരു ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ട്രൂത്ത് ടേബിളിൽ $2^m$ നിരകളും m+n കോളങ്ങളും ഉണ്ടായിരിക്കും.




*ചിത്രം 2.5 ലോജിക്കൽ OR ഗേറ്റ്*

ലോജിക് സർക്യൂട്ടുകളുടെ രൂപകല്പനയിൽ ലോജിക്കൽ OR പ്രവർത്തനം നിർവഹിക്കുന്ന ഗേറ്റിനെ ലോജിക്കൽ OR ഗേറ്റ് എന്ന് വിളിക്കുന്നു. ചിത്രം 2.5 ബൂളിയൻ ബീജഗണിതത്തിലെ OR ഗേറ്റിന്റെ ലോജിക്കൽ ചിഹ്നം കാണിച്ചിരിക്കുന്നു.

ഒരു ഇലക്ട്രോണിക് സർക്യൂട്ടിന്റെ സഹായത്തോടെ ഈ ഗേറ്റിന്റെ പ്രവർത്തനം വിശദീകരിക്കാം. ചിത്രം 2.6 ൽ സമാന്തരമായി വിന്യസിച്ചിട്ടുള്ള രണ്ട് സ്വിച്ചുകൾ OR ഗേറ്റ് സർക്യൂട്ടിനെക്കുറിച്ചുള്ള സങ്കല്പം വ്യക്തമാക്കുന്നു. ഇതിൽ A യും B യും രണ്ട് സ്വിച്ചുകളും Y ഒരു ബൾബുമാണ്. ഇതിൽ സ്വിച്ചുകൾ ഒന്നുകിൽ അടഞ്ഞതോ (ON) അല്ലെങ്കിൽ തുറന്നതോ (OFF) ആയ അവസ്ഥയിലായിരിക്കും. സർക്യൂട്ടിലെ ഒരു സ്വിച്ചെങ്കിലും അടഞ്ഞിരുന്നാൽ ബൾബ് പ്രകാശിക്കും. രണ്ടു



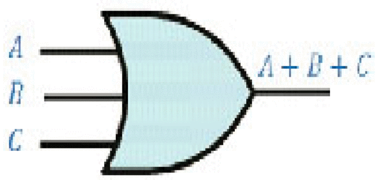

സ്വിച്ച് A - തുറന്നിരിക്കുന്നു = 0(OFF), അടഞ്ഞിരിക്കുന്നു = 1 (ON)
സ്വിച്ച് B - തുറന്നിരിക്കുന്നു = 0(OFF), അടഞ്ഞിരിക്കുന്നു = 1 (ON)

*ചിത്രം 2.6 രണ്ട് സ്വിച്ചും ഒരു ബൾബും സമാന്തരമായി ബന്ധിപ്പിച്ചിട്ടുള്ള സർക്യൂട്ട്*

സ്വിച്ചുകളും തുറന്നിരിക്കുമ്പോൾ മാത്രമാണ് ബൾബ് പ്രകാശിക്കാതിരിക്കുന്നത്. മേൽപ്പറഞ്ഞ സർക്യൂട്ടിന്റെ പ്രവർത്തനം OR ഗേറ്റിന്റെ പ്രവർത്തനവുമായി നമുക്ക് ബന്ധപ്പെടുത്താം. OFF പ്രതിനിധീകരിക്കുന്നത് ലോജിക്കൽ LOW (0) അവസ്ഥയെയും ON ലോജിക്കൽ HIGH (1) അവസ്ഥയെയുമാണ് A, B സ്വിച്ചുകൾ OR ഗേറ്റിന്റെ ഇൻപുട്ടും ബൾബിന്റെ അവസ്ഥ OR ഗേറ്റിന്റെ ഔട്ട്പുട്ടുമാണെന്ന് കരുതുക. ട്രൂത്ത് ടേബിൾ 2.9 ഇത്തരം ഒരു OR ഗേറ്റിന്റെ പ്രവർത്തനം വിശദീകരിക്കുന്നു. OR ഗേറ്റിന്റെ ബൂളിയൻ പദപ്രയോഗം Y=A+B എന്നാണ് എഴുതുന്നത്.

| A | B | C | A + B + C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.10 മൂന്ന് ഇൻപുട്ടുകളുള്ള OR ഗേറ്റിന്റെ ട്രൂത്ത് ടേബിൾ*




*ചിത്രം 2.7 മൂന്ന് ഇൻപുട്ടുകളുള്ള OR ഗേറ്റ്*

ഒരു OR ഗേറ്റിന് രണ്ടിലേറെ ഇൻപുട്ടുകൾ കൈകാര്യം ചെയ്യാൻ സാധിക്കും. മൂന്ന് ഇൻപുട്ടുകളുള്ള OR ഗേറ്റിന്റെ ബൂളിയൻ പ്രതിനിധാനവും ലോജിക്കൽ ചിഹ്നവും എങ്ങനെയായിരിക്കുമെന്ന്

നോക്കാം. മൂന്ന് ഇൻപുട്ടുകളുള്ള OR ഗേറ്റിന്റെ ബൂളിയൻ പ്രതിനിധാനം Y=A+B+C എന്നാണ്. ഇതിന്റെ ലോജിക്കൽ ചിഹ്നം ചിത്രം 2.7 ൽ കൊടുത്തിരിക്കുന്നു. OR ഗേറ്റിന്റെ ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആണെങ്കിൽ ഔട്ട്പുട്ടും 1 ആയിരിക്കുമെന്ന് പട്ടിക 2.9, 2.10 എന്നിവയിൽ നിന്ന് നമുക്ക് മനസിലാക്കാം. എല്ലാ ഇൻപുട്ടുകളും 0 ആണെങ്കിൽ മാത്രമേ OR ഗേറ്റിന്റെ ഔട്ട്പുട്ട് 0 ആകുന്നുള്ളൂ.

| റെസ്റ്റോറന്റ് | പണം | ഭക്ഷണം |
|---|---|---|
| ഇല്ല | ഇല്ല | ഇല്ല |
| ഇല്ല | ഉണ്ട് | ഇല്ല |
| ഉണ്ട് | ഇല്ല | ഇല്ല |
| ഉണ്ട് | ഉണ്ട് | ഉണ്ട് |

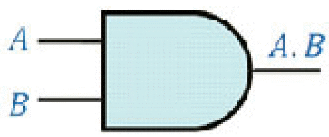
പട്ടിക 2.11 ലോജിക്കൽ AND ന്റെ പ്രവർത്തനം

## b. AND ഓപ്പറേറ്ററും AND ഗേറ്റും

AND എന്ന ബൂളിയൻ ക്രിയ മനസ്സിലാക്കുവാൻ നമ്മുക്ക് മറ്റൊരു ഉദാഹരണം നോക്കാം. നിങ്ങൾ വീട്ടിൽ നിന്നും അകലെയാണെന്നും ഇപ്പോൾ ഉച്ചഭക്ഷണത്തിനുള്ള സമയമായെന്നും വിചാരിക്കുക. ഉച്ച ഭക്ഷണം ലഭിക്കുന്നതിന് ഇവിടെ രണ്ടു കാര്യങ്ങൾ ആവശ്യമാണ്. 1. ഒരു റെസ്റ്റോറന്റ് ഉണ്ടായിരിക്കണം. 2. ഭക്ഷണം വാങ്ങുന്നതിനുള്ള പണം നിങ്ങളുടെ പക്കൽ ഉണ്ടായിരിക്കണം. ഇവ രണ്ടും ബന്ധിപ്പിച്ചുകൊണ്ട് നമുക്ക് ഒരു സംയുക്തപ്രസ്താവന ഉണ്ടാക്കാം. ഒരു റെസ്റ്റോറന്റും ഭക്ഷണം വാങ്ങുന്നതിനുള്ള പണവും ഉണ്ടെങ്കിൽ ഭക്ഷണം ലഭിക്കും. ഈ പ്രസ്താവനയിൽ 'ഉം' എന്നതിന്റെ ഉപയോഗം ശ്രദ്ധിക്കുക. പട്ടിക 2.11 ൽ ഭക്ഷണം ലഭിക്കുന്നതിനുള്ള യുക്തിപരമായ വിശകലനം കാണിക്കുന്നത് അതിന് ബൂളിയൻ AND ക്രിയയുമായുള്ള സാദൃശ്യമാണ്.

| A | B | A.B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

പട്ടിക 2.12 AND പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ

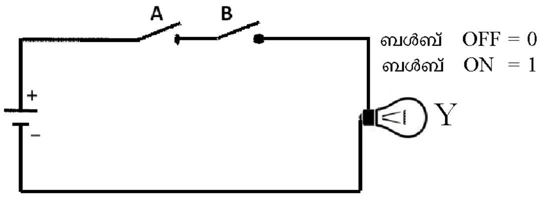



ചിത്രം 2.8 ലോജിക്കൽ AND ഗേറ്റ്

AND ഓപ്പറേറ്റർ ലോജിക്കൽ ഗുണിതം നിർവഹിക്കുന്നു. ഈ പ്രവർത്തനം സൂചിപ്പിക്കുന്നത് **(ഡോട്ട്)** ചിഹ്നം ഉപയോഗിച്ചാണ്. അതിനാൽ A.B എന്ന പദപ്രയോഗം A AND B എന്ന് വായിക്കാം. AND പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ പട്ടിക 2.12 ൽ കൊടുത്തിരിക്കുന്നു. A, B എന്നിവ ഇൻപുട്ടുകളും (ഓപ്പറന്റ്) A.B ഔട്ട്പുട്ടും (ഉത്തരം) ആണെന്ന് അനുമാനിക്കാം. എല്ലാ ഇൻപുട്ടുകളും 1 ആകുമ്പോൾ മാത്രമാണ് ഔട്ട്പുട്ട് 1 ലഭിക്കുന്നത്. ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 0 ആകുമ്പോൾ ഔട്ട്പുട്ടും 0 ആകുന്നു.

സർക്യൂട്ടുകൾ രൂപകല്പന ചെയ്യുമ്പോൾ യുക്തിപരമായ AND പ്രവർത്തനം നിർവഹിക്കാൻ ഉപയോഗിക്കുന്ന ലോജിക്കൽ ഗേറ്റിനെ AND ഗേറ്റ് എന്ന് വിളിക്കുന്നു. ബൂളിയൻ ബീജഗണിതത്തിലെ AND ഗേറ്റിന്റെ ചിഹ്നം ചിത്രം 2.8 ൽ കാണിച്ചിരിക്കുന്നു. ചിത്രം 2.9 ൽ കൊടുത്തിരിക്കുന്ന ഇലക്ട്രോണിക് സർക്യൂട്ട് വഴി ഈ ഗേറ്റിന്റെ പ്രവർത്തനം വിശദമാക്കാം. ഈ സർക്യൂട്ടിൽ AND ഗേറ്റ് എന്ന സങ്കല്പം വിശദമാക്കുന്ന ശ്രേണിയായി രണ്ട് സ്വിച്ചുകളുണ്ട്. ഇതിൽ A, B എന്നിവ സ്വിച്ചുകളും Y ബൾബുമാണ്. സർക്യൂട്ടിലെ രണ്ട് സ്വിച്ചുകളും അടഞ്ഞിരിക്കുമ്പോൾ മാത്രമാണ് ബൾബ് പ്രകാശിക്കുന്നത് എന്ന് കാണാം. ഏതെങ്കിലും ഒരു സ്വിച്ച് തുറന്നിരുന്നാൽ ബൾബ്

പ്രകാശിക്കുന്നില്ല. മേൽപ്പറഞ്ഞ സർക്യൂട്ടിന്റെ പ്രവർത്തനം AND ഗേറ്റിന്റെ പ്രവർത്തനവുമായി നമുക്ക് ബന്ധപ്പെടുത്താം. OFF പ്രതിനിധീകരിക്കുന്നത് ലോജിക്കൽ LOW (0) അവസ്ഥയെയും ON ലോജിക്കൽ HIGH (1) അവസ്ഥയെയുമാണ്. A, B സ്വിച്ചുകൾ AND ഗേറ്റിന്റെ ഇൻപുട്ടും ബൾബിന്റെ അവസ്ഥ AND ഗേറ്റിന്റെ ഔട്ട്പുട്ടുമാണെന്ന് കരുതുക. ട്രൂത്ത് ടേബിൾ 2.9 ഇത്തരം ഒരു AND ഗേറ്റിന്റെ പ്രവർത്തനം വിശദീകരിക്കുന്നു. AND ഗേറ്റിന്റെ ബൂളിയൻ പദപ്രയോഗം Y=A.B എന്നാണ് എഴുതുന്നത്.




*സ്വിച്ച് A തുറന്നിരിക്കുന്നു = 0 (OFF),*
*അടഞ്ഞിരിക്കുന്നു = 1(ON)*
*സ്വിച്ച് B തുറന്നിരിക്കുന്നു = 0 (OFF),*
*അടഞ്ഞിരിക്കുന്നു = 1 (ON)*

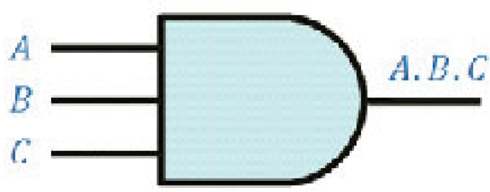
*ചിത്രം 2.9 രണ്ട് സ്വിച്ചും ഒരു ബൾബും ശ്രേണിയായി ബന്ധിപ്പിച്ചിട്ടുള്ള സർക്യൂട്ട്.*

AND ഗേറ്റിന് രണ്ടിലേറെ ഇൻപുട്ടുകൾ കൈകാര്യം ചെയ്യാനാകും. ഇതിൽ മൂന്ന് ഇൻപുട്ട് വരുമ്പോൾ ബൂളിയൻ പ്രതിനിധാനവും ചിഹ്നവും എങ്ങനെയാവും എന്ന് നോക്കാം. മൂന്ന് ഇൻപുട്ടുള്ള AND ഗേറ്റിന്റെ ബൂളിയൻ പദപ്രയോഗം Y=A.B.C എന്നാണ്.

| A | B | C | A.B.C |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 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 | 1 |

*പട്ടിക 2.13 : മൂന്ന് ഇൻപുട്ടുകളുള്ള AND ഗേറ്റിന്റെ ട്രൂത്ത് ടേബിൾ*

ചിത്രം 2.10 ൽ മൂന്ന് ഇൻപുട്ടുകളുള്ള AND ഗേറ്റ് കാണിച്ചിരിക്കുന്നു. AND ഗേറ്റിൽ ഏതെങ്കിലും ഇൻപുട്ട് 0 ആണെങ്കിൽ ഔട്ട്പുട്ടും 0 ആയിരിക്കും. എല്ലാ ഇൻപുട്ടുകളും 1 ആണെങ്കിൽ മാത്രമേ ഔട്ട്പുട്ടായി 1 ലഭിക്കുകയുള്ളൂ. 2.12, 2.13 എന്നീ ട്രൂത്ത് ടേബിളുകൾ ഇത് കാണിച്ചുതരുന്നു.




*ചിത്രം 2.10 : മൂന്ന് ഇൻപുട്ടുകളുള്ള AND ഗേറ്റ്*

## c. NOT ഓപ്പറേറ്ററും NOT ഗേറ്റും

ബൂളിയൻ NOT പ്രവർത്തനം മനസ്സിലാക്കുന്നതിന് നമുക്ക് മറ്റൊരു സാഹചര്യം ചർച്ച ചെയ്യാം. എന്നും രാവിലെ നിങ്ങൾ വ്യായാമത്തിനായി നടക്കാൻ പോകുന്നയാളാണെന്ന് കരുതുക. എല്ലാ ദിവസവും നിങ്ങൾക്ക് അതിന് കഴിയുമോ? മഴയുണ്ടെങ്കിൽ നിങ്ങൾക്ക് അതിന് സാധിക്കുമോ? പട്ടിക 2.14 ൽ ഇതുമായി ബന്ധപ്പെട്ട എല്ലാ സാധ്യതകളും കാണിച്ചിരിക്കുന്നു. ഇതിന് ബൂളിയൻ NOT പ്രവർത്തനവുമായി സാമ്യമുണ്ട്.

ഇത് ഒരു യുനറി (Unary) ഓപ്പറേറ്ററാണ്. അതിനാൽ ഇതിന് ഒരേ ഒരു ഓപ്പറന്റ് മാത്രമെ ആവശ്യമുള്ളൂ. NOT ഓപ്പറേറ്റർ യുക്തിപരമായ നിഷേധം (Logical Negation) നിർവഹിക്കുന്നു. അതിന്റെ ചിഹ്നം ¯ (over bar) മുകൾവര ആണ്.

| മഴ | വ്യായാമം |
|---|---|
| ഇല്ല | ഉണ്ട് |
| ഉണ്ട് | ഇല്ല |

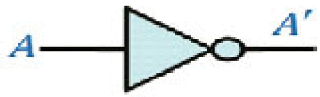
*പട്ടിക 2.14 ലോജിക്കൽ NOT*

NOT ന്റെ പദപ്രയോഗത്തെ $\bar{A}$ എന്നോ $A'$ എന്നോ അടയാളപ്പെടുത്താം. $\bar{A}$ എന്നതിനെ A ബാർ എന്നും $A'$ എന്നതിനെ A ഡാഷ് എന്നും ഉച്ചരിക്കുന്നു. ടേബിൾ 2.15 ൽ NOT പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ കൊടുത്തിരിക്കുന്നു.

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

*പട്ടിക 2.15 NOT പ്രവർത്തനത്തിന്റെ ട്രൂത്ത് ടേബിൾ*

NOT ഓപ്പറേറ്ററിൽ A എന്നത് ഇൻപുട്ടും (ഓപ്പറന്റും) $\bar{A}$ എന്നത് ഔട്ട്പുട്ടും (ഉത്തരം) ആണെന്ന് അനുമാനിക്കുക. ഇൻപുട്ടിന്റെ വിപരീത മൂല്യമാണ് ഔട്ട്പുട്ട് എന്ന് ട്രൂത്ത് ടേബിളിൽ നിന്ന് മനസ്സിലാക്കാം. NOT പ്രവർത്തനം നിർവഹിക്കുന്ന ലോജിക് ഗേറ്റിനെ NOT ഗേറ്റ് എന്ന് വിളിക്കുന്നു. ചിത്രം 2.11 ൽ NOT ഗേറ്റിന്റെ ചിഹ്നം കാണിച്ചിരിക്കുന്നു.




*ചിത്രം 2.11 NOT ഗേറ്റ്*

**സ്വയം വിലയിരുത്താം**

1. ബൂളിയൻ വേരിയബിൾ എന്ന പദം നിർവചിക്കുക.
2. ഓരോ വ്യക്തിഗത യൂണിറ്റിനാലും നിർമ്മിക്കപ്പെടുന്ന ലോജിക്കൽ സർക്യൂട്ടിനെ .................. എന്ന് വിളിക്കുന്നു.
3. എല്ലാ ഇൻപുട്ടുകളും ഉയർന്നതായിരിക്കുമ്പോൾ (High) മാത്രം ഉയർന്ന (High) ഔട്ട്പുട്ട് തരുന്ന ലോജിക്കൽ ഓപ്പറേറ്റർ/ഗേറ്റ് ന്റെ പേര് എഴുതുക.
4. ട്രൂത്ത് ടേബിൾ എന്ന പദം നിർവചിക്കുക.
5. AND ഓപ്പറേഷനിൽ ലോജിക്കൽ ........................ ഉം OR ഓപ്പറേറ്ററിൽ ലോജിക്കൽ ........................ ഉം ആണ് നടക്കുന്നത്.
6. OR ഗേറ്റിന്റെ ലോജിക് ചിഹ്നം വരക്കുക.

ഒരു NOT ഗേറ്റിനെ ഇൻവർട്ടർ (Inverter) എന്നും വിളിക്കുന്നു. ഇതിന് ഒരു ഇൻപുട്ടും ഒരു ഔട്ട്പുട്ടും മാത്രമേയുള്ളൂ. ഇൻപുട്ട് എല്ലായ്പോഴും വിപരീതാവസ്ഥയിലേക്ക് പരിവർത്തനം ചെയ്യുന്നു. ഇൻപുട്ട് 0 ആണെങ്കിൽ അതിന്റെ ഔട്ട്പുട്ട് വിപരീതം അല്ലെങ്കിൽ പൂരകം 1 ആയിരിക്കും. അതുപോലെ ഇൻപുട്ട് 1 ആണെങ്കിൽ ഔട്ട്പുട്ട് അതിന്റെ പൂരകമായ 0 ആയിരിക്കും.

## 2.6 ബൂളിയൻ ബീജഗണിതത്തിലെ അടിസ്ഥാന അംഗീകൃത തത്വങ്ങൾ (Basic postulates of Boolean algebra)

ചില അടിസ്ഥാന നിയമങ്ങളോടുകൂടിയ ഒരു ഗണിതശാസ്ത്ര ശാഖയാണ് ബൂളിയൻ ബീജഗണിതം. ഈ അടിസ്ഥാന നിയമങ്ങളെ അംഗീകൃത തത്വങ്ങൾ (Postulates) എന്ന് വിളിക്കുന്നു. ഇവയ്ക്ക് ശാസ്ത്രീയമായ അടിത്തറയില്ലെങ്കിലും കൃത്യമായ ഘടനയോടുകൂടിയ ശാസ്ത്ര തത്വങ്ങൾ നിർമ്മിക്കുവാൻ ഇവ ഉപയോഗിക്കുന്നു. മറ്റൊരു തരത്തിൽപ്പറഞ്ഞാൽ ഇത്തരം അംഗീകൃത തത്വങ്ങളും നിയമങ്ങളും ഉപയോഗിച്ച് തെളിയിക്കാൻ കഴിയുന്ന ചില സിദ്ധാന്തങ്ങൾ ബൂളിയൻ ബീജഗണിതത്തിൽ ഉണ്ട്.

**അംഗീകൃത തത്വം 1 : 0 ന്റെയും 1 ന്റെയും സിദ്ധാന്തങ്ങൾ**

$A \neq 0$, ആണെങ്കിൽ $A = 1$ ഉം $A \neq 1$ ആണെങ്കിൽ $A=0$ ആകുന്നു.

**അംഗീകൃത തത്വം 2 : OR ഓപ്പറേഷൻ (യുക്തിപരമായ സങ്കലനം)**

$0 + 0 = 0 \qquad 0 + 1 = 1 \qquad 1 + 0 = 1 \qquad 1 + 1 = 1$

**അംഗീകൃത തത്വം 3 : AND ഓപ്പറേഷൻ (യുക്തിപരമായ ഗുണനം)**

$0 . 0 = 0 \qquad 0 . 1 = 0 \qquad 1 . 0 = 0 \qquad 1 . 1 = 1$

**അംഗീകൃത തത്വം 4 : NOT ഓപ്പറേഷൻ (യുക്തിപരമായ നിഷേധം അല്ലെങ്കിൽ പൂരക നിയമം)**

$\bar{0} = 1 \qquad \bar{1} = 0$

### ദ്വൈതസിദ്ധാന്തം (Principle of Duality)

ബൂളിയൻ വേരിയബിളും, വിലകളും, ബൂളിയൻ ഓപ്പറേറ്റർ വഴി കൂട്ടിച്ചേർത്തു ബൂളിയൻ പദപ്രയോഗം നിർമ്മിക്കാം. $X+Y$, $A+1$ എന്നിവ ബൂളിയൻ പദപ്രയോഗങ്ങൾക്ക് ഉദാഹരണങ്ങളാണ്. അതുപോലെ ബൂളിയൻ ബീജഗണിതത്തിന്റെ അംഗീകൃത തത്വങ്ങളായ 2, 3, 4 എന്നിവയും ബൂളിയൻ പദപ്രയോഗങ്ങളാണ്. അംഗീകൃത തത്വം 2 ലെ പ്രസ്താവനകൾ നമുക്ക് നോക്കാം. ഇതിൽ പൂജ്യത്തിനെ ഒന്നായും ഒന്നിനെ പൂജ്യമായും അതുപോലെ ഓപ്പറേറ്ററുകളായ OR (+) നെ AND (.) ആയും മാറ്റംവരുത്തിയാൽ അംഗീകൃത തത്വം 3 ലഭിക്കുന്നു. അതുപോലെ അംഗീകൃത തത്വം 2 പ്രസ്താവനയിൽ പൂജ്യത്തിനെ ഒന്നായും ഒന്നിനെ പൂജ്യമായും അതുപോലെ ഓപ്പറേറ്ററുകളായ AND (.) നെ OR (+) ആയും മാറ്റം വരുത്തിയാൽ അംഗീകൃത തത്വം 3 ലഭിക്കുന്നു. ഈ ആശയത്തെയാണ് ദ്വൈതസിദ്ധാന്തം എന്ന് പറയുന്നത്.

ഏതൊരു ബൂളിയൻ പ്രസ്താവനയ്ക്കും ഒരു ദ്വൈത രൂപം ഉണ്ടായിരിക്കും. അത് താഴെ പറയുന്ന മൂന്ന് നിയമങ്ങൾ അനുസരിച്ചാണ് വികസിപ്പിക്കുന്നതെന്ന് ദ്വൈതസിദ്ധാന്തം പ്രസ്താവിക്കുന്നു.

(i) ഓരോ OR (+) ഉം AND (.) ആയി മാറ്റുക.

(ii) ഓരോ AND (.) ഉം OR (+) ആയി മാറ്റുക.

(iii) ഓരോ 0 ഉം 1 ആയും 1 നെ 0 ആയും മാറ്റുക.

## 2.7 ബൂളിയൻ ബീജഗണിതത്തിലെ അടിസ്ഥാന തത്വങ്ങൾ (Basic theorems of Boolean algebra)

ഓരോ സിദ്ധാന്തത്തിനും ചില അടിസ്ഥാനവും അംഗീകൃതവുമായ നിയമങ്ങളും ഉണ്ട്. സിദ്ധാന്തത്തിന്റെ നിയമാവലികൾ സ്വയം പ്രമാണം (Axioms) എന്നറിയപ്പെടുന്നു. മേൽപ്പറഞ്ഞ സ്വയം പ്രമാണങ്ങളോ അടിസ്ഥാന തത്വങ്ങളോ ഉപയോഗിച്ചുള്ള അനുമാനങ്ങളിൽ നിന്ന് ഒരു നിഗമനത്തിൽ എത്തിച്ചേരാവുന്നതാണ്. ഇത്തരം നിഗമനങ്ങളെ നിയമങ്ങൾ അല്ലെങ്കിൽ സിദ്ധാന്തങ്ങൾ എന്ന് വിളിക്കുന്നു. ബൂളിയൻ പദപ്രയോഗം ലഘൂകരിക്കുന്നതിനും കൈകാര്യം ചെയ്യുന്നതിനും ഉള്ള സാമഗ്രികൾ ബൂളിയൻ ആൾജിബ്രയുടെ സിദ്ധാന്തങ്ങൾ തരുന്നു. ഇവിടെ ചില ബൂളിയൻ നിയമങ്ങളെയും സിദ്ധാന്തങ്ങളെയെയും കുറിച്ച് ചർച്ച ചെയ്യാം. മുൻകൂട്ടി തെളിയിച്ചിട്ടുള്ള ബൂളിയൻ നിയമങ്ങളുടെയും ട്രൂത്ത് ടേബിളുകളുടെയും സഹായത്താൽ മേൽപ്പറഞ്ഞ സിദ്ധാന്തങ്ങളും നിയമങ്ങളും തെളിയിക്കാം.

### 2.7.1 അനന്യത നിയമം (Identity law)

X ഒരു ബൂളിയൻ വേരിയബിൾ ആണെങ്കിൽ, അനന്യത നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാണ്.

(i) $0 + X = X$ (ii) $1 + X = 1$

(iii) $0 . X = 0$ (iv) $1 . X = X$

പ്രസ്താവന (i) ഉം (ii) ഉം സങ്കലന അനന്യത എന്നും പ്രസ്താവന (iii) ഉം (iv) ഉം ഗുണന അനന്യത എന്നും അറിയപ്പെടുന്നു. പ്രസ്താവന (i), പ്രസ്താവന (iv) ന്റെ ദ്വൈത രൂപവും (Dual form), അതുപോലെ തിരിച്ചുമാകുന്നു. അതുപോലെ പ്രസ്താവന (ii), പ്രസ്താവന (iii) ന്റെ ദ്വൈത രൂപങ്ങളാകുന്നു. ട്രൂത്ത് ടേബിളുകൾ 2.16(a), 2.16(b), 2.17(a), 2.17(b) ഈ നിയമങ്ങൾ തെളിയിക്കുന്നു.

| 0 | X | 0 + X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |

*പട്ടിക 2.16(a) സങ്കലന അനന്യത നിയമം*

| 1 | X | 1 + X |
|---|---|---|
| 1 | 0 | 1 |
| 1 | 1 | 1 |

*പട്ടിക 2.16(b) സങ്കലന അനന്യത നിയമം*

പട്ടിക 2.16 (a) ലെ 2 ഉം 3 ഉം നിരകളിലെ വിലകൾ തുല്യമാണെന്നു കാണാം. അതിനാൽ $0 + X = X$ എന്ന് തെളിയിക്കപ്പെടുന്നു. അതുപോലെ പട്ടിക 2.16 (b) ലെ 1 ഉം 3 ഉം നിരകളിലെ വിലകൾ തുല്ല്യമായതിനാൽ $1+X=1$ എന്ന പ്രസ്താവനയും ശരിയാകുന്നു.

| 0 | X | 0 . X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |

*പട്ടിക 2.17(a) ഗുണന അനന്യത നിയമം*

| 1 | X | 1 . X |
|---|---|---|
| 1 | 0 | 0 |
| 1 | 1 | 1 |

*പട്ടിക 2.17(b) ഗുണന അനന്യത നിയമം*

പട്ടിക 2.17(a) ലെ 2 ഉം 3 ഉം നിരകളിലെ വിലകൾ തുല്യമാണെന്നു കാണാം. അതിനാൽ $0.X = 0$ എന്ന് തെളിയിക്കപ്പെടുന്നു. അതുപോലെ പട്ടിക 2.17(b) ലെ 2 ഉം 3 ഉം നിരകളിലെ വിലകൾ തുല്യമായതിനാൽ $1.X = X$ എന്ന പ്രസ്താവനയും ശരിയാകുന്നു.

### 2.7.2 വർഗസമ നിയമം (Idempotent law)

വർഗസമ നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു. (i) $X + X = X$

(ii) $X . X = X$

| X | X | X + X |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |

*പട്ടിക 2.18 (a) : വർഗ്ഗസമനിയമം*

| X | X | X . X |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 1 | 1 |

*പട്ടിക 2.18 (b) : വർഗ്ഗസമനിയമം*

X ന്റെ വില 0 ആയാൽ പ്രസ്താവന സത്യമാകുന്നു, എന്തുകൊണ്ടെന്നാൽ $0+0 = 0$ (അംഗീകൃത തത്വം 2) ഉം $0.0 = 0$ ഉം (അംഗീകൃത തത്വം 3). പ്രകാരം പ്രസ്താവന സത്യമാകുന്നു. അതുപോലെ X ന്റെ വില 1 ആയാലും പ്രസ്താവനകൾ ശരിയാകുന്നു. ട്രൂത്ത് ടേബിളുകൾ 2.18(a), 2.18(b) എന്നിവ ഈ നിയമങ്ങളുടെ തെളിവ് കാണിച്ച് തരുന്നു.

## 2.7.3 വർഗപൂരക നിയമം ( Involution law)

ഈ നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു. $\overline{\overline{X}} = X$
അംഗീകൃത തത്വം 4 പ്രകാരം $X = 0$ ആയാൽ $\overline{X} = 1$ ആകുന്നു. അതിന്റെ പൂരകം $\overline{\overline{X}} = \overline{1} = 0$ എന്നത് X നു തുല്യമാകുന്നു. X ന്റെ വില 1 ആയാലും പ്രസ്താവന സത്യമാകുന്നു.

ട്രൂത്ത് ടേബിൾ 2.19 ലെ 1 ഉം 3 ഉം നിരകൾ $\overline{\overline{X}} = X$ എന്നത് തെളിയിക്കുന്നു.

| $X$ | $\overline{X}$ | $\overline{\overline{X}}$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 1 |

*പട്ടിക 2.19 വർഗപൂരക നിയമം*

## 2.7.4 പൂരക നിയമം (Complementary law)

പൂരക നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു. (i) $X + \overline{X} = 1$

(ii) $X \cdot \overline{X} = 0$

X ന്റെ വില 0 ആയാൽ $\overline{X}$ എന്നത് 1 ആകുന്നു. ആയതിനാൽ $X + \overline{X}$ എന്നത് $0 + 1 = 1$ ആകുന്നു (അംഗീകൃത തത്വം 2). അതുപോലെ X ന്റെ വില 1 ആയാൽ $\overline{X} = 0$ ആകുന്നു. ട്രൂത്ത് ടേബിളുകൾ 2.20(a), 2.20(b) എന്നിവ സാധ്യമായ വിലകൾ ഉൾക്കൊള്ളിച്ചു കൊണ്ട് ഈ നിയമം തെളിയിക്കുന്നു. പ്രസ്താവനകൾ പരസ്പരം ദ്വന്ദ്വങ്ങൾ ആണെന്ന് ശ്രദ്ധിക്കുക.

| $X$ | $\overline{X}$ | $X + \overline{X}$ |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 0 | 1 |

*പട്ടിക 2.20(a) പൂരക നിയമം*

| $X$ | $\overline{X}$ | $X \cdot \overline{X}$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 0 | 0 |

*പട്ടിക 2.20(b) പൂരക നിയമം*

## 2.7.5 ക്രമനിയമം (Commutative law)

OR, AND എന്നീ ഓപ്പറേഷനുകളിൽ വേരിയബിളിന്റെ സ്ഥാനം മാറ്റുന്നതിന് ക്രമനിയമം അനുവദിക്കുന്നു. X ഉം Y ഉം രണ്ട് വേരിയബിളുകൾ ആണെങ്കിൽ, ഈ നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു.

(i) $X + Y = Y + X$

(ii) $X \cdot Y = Y \cdot X$

ട്രൂത്ത് ടേബിളുകൾ 2.21(a), 2.21(b) എന്നിവ ഈ നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | X + Y | Y + X |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.21(a) ക്രമനിയമം*

| X | Y | X.Y | Y.X |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.21(b) ക്രമനിയമം*

OR, AND എന്നിവയുടെ ഓരോ ഓപ്പറേഷനിലും ഓപ്പറന്ററുകളുടെ ക്രമം ഔട്ട്പുട്ടിനെ ബാധിക്കുകയില്ലെന്ന് നിയമം ഉറപ്പുവരുത്തുന്നു.

## 2.7.6 സംയോജന നിയമം (Associative law)

OR, AND എന്നീ ഓപ്പറേഷനുകളിൽ മൂന്നു ഓപ്പറന്റുകൾ വരുന്ന സന്ദർഭങ്ങളിൽ, ഓപ്പറന്റുകളെ വ്യത്യസ്ത രീതിയിൽ ഗ്രൂപ്പ് ചെയ്യുവാൻ സംയോജന നിയമം അനുവദിക്കുന്നു. X, Y, Z എന്നിവ മൂന്ന് വേരിയബിളുകൾ ആണെങ്കിൽ സംയോജന നിയമം ഇങ്ങനെയാകുന്നു.

(i) $X + (Y + Z) = (X + Y) + Z$

(ii) $X . (Y . Z) = (X . Y) . Z$

ട്രൂത്ത് ടേബിളുകൾ 2.22(a), 2.22(b) എന്നിവ ഈ നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | Z | X + Y | Y + X | X+(Y + Z) | (X+Y)+Z |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*പട്ടിക 2.22(a) സംയോജന നിയമം 1*

പട്ടിക 2.22(a) യിലെ 6 ഉം 7 ഉം നിരകൾ X + (Y + Z) = (X + Y) + Z എന്ന നിയമത്തെയും പട്ടിക 2.22(b) യിലെ 6 ഉം 7 ഉം നിരകൾ X . (Y . Z) = (X . Y) . Z എന്ന നിയമത്തെയും സാധൂകരിക്കുന്നു.

| X | Y | Z | X . Y | Y . Z | X . (Y . Z) | (X . Y) . Z |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*പട്ടിക 2.22(b) സംയോജന നിയമം 2*

OR (യുക്തിപരമായ സങ്കലനം) അല്ലെങ്കിൽ AND (യുക്തിപരമായ ഗുണിതം) പ്രവർത്തനങ്ങളിലെ വേരിയബിളുകളുടെ ക്രമവും കൂടിച്ചേരലും അന്തിമ ഔട്ട്പുട്ടിനെ ബാധിക്കുന്നില്ലെന്ന് സംയോജന നിയമം ഉറപ്പാക്കുന്നു.

## 2.7.7 വിതരണ നിയമം (Distributive law)

ബൂളിയൻ പദപ്രയോഗങ്ങൾ സാധാരണ ബീജഗണിതത്തിലെപ്പോലെ സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം വഴിയും അതുപോലെ ഗുണനത്തിനുമേലുള്ള സങ്കലന വിതരണം വഴിയും വിപുലീകരിക്കാൻ വിതരണ നിയമപ്രകാരം സാധിക്കുന്നു. X, Y, Z എന്നിവ വേരിയബിളുകൾ ആണെങ്കിൽ, ഈ നിയമം പ്രസ്താവിക്കുന്നത് ഇങ്ങനെയാകുന്നു.

(i) $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$

(ii) $X + Y \cdot Z = (X + Y) \cdot (X + Z)$

ചുവടെ ചേർത്തിരിക്കുന്ന ട്രൂത്ത് ടേബിളുകൾ ഈ നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | Z | Y + Z | X.(Y+Z) | X.Y | X.Z | X.Y + X.Z |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*പട്ടിക 2.23(a) സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം*

പട്ടിക 2.23(a) യിലെ 5 ഉം 8 ഉം നിരകൾ $X \cdot (Y + Z) = X \cdot Y + X \cdot Z$ എന്ന നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | Z | Y . Z | X + Y . Z | X+Y | X+Z | (X+Y) . (X+Z) |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

*പട്ടിക 2.23(b) ഗുണനത്തിനുമേലുള്ള സങ്കലന വിതരണം*

പട്ടിക 2.23(b) യിലെ 5 ഉം 8 ഉം നിരകൾ $X + Y . Z = (X + Y) . (X + Z)$ എന്ന നിയമത്തെ സാധൂകരിക്കുന്നു.

### 2.7.8 സ്വാംശീകരണ നിയമം (Absorption law)

രണ്ടു വേരിയബിളുകൾ ഉപയോഗിക്കുകയും അതിൽ ഒന്ന് ഉത്തരം ആകുകയും ചെയ്യുന്ന തരത്തിലുള്ള വിതരണ നിയമമാണ് സ്വാംശീകരണ നിയമം. X ഉം Y ഉം വേരിയബിളുകൾ ആണെങ്കിൽ, ഈ നിയമം ഇങ്ങനെ പ്രസ്താവിക്കുന്നു.

(i) $X + (X . Y) = X$

(ii) $X . (X + Y) = X$

ട്രൂത്ത് ടേബിളുകൾ 2.24(a), 2.24(b) എന്നിവ ഈ നിയമത്തെ സാധൂകരിക്കുന്നു.

| X | Y | X . Y | X+(X.Y) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.24(a) സ്വാംശീകരണ നിയമം 1*

| X | Y | X+Y | X.(X+Y) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 |

*പട്ടിക 2.24(b) സ്വാംശീകരണ നിയമം 2*

പട്ടിക 2.24(a) യിലെ 1 ഉം 4 ഉം നിരകളും പട്ടിക 2.24(b) യിലെ 1 ഉം 4 ഉം നിരകളും ഈ നിയമം ശരിയാണെന്നു തെളിയിച്ചിരിക്കുന്നു.

പട്ടിക 2.25 നമ്മൾ ചർച്ച ചെയ്ത എല്ലാ ബൂളിയൻ നിയമങ്ങളും സൂക്ഷ്മമായി ചിത്രീകരിക്കുന്നു.

| നം. | ബൂളിയൻ നിയമം | പ്രസ്താവന 1 | പ്രസ്താവന 2 |
|---|---|---|---|
| 1 | സങ്കലന അനന്യത (Additive Identity) | $0 + X = X$ | $1 + X = 1$ |
| 2 | ഗുണന അനന്യത (Multiplicative Identity) | $0 . X = 0$ | $1 . X = X$ |
| 3 | വർഗസമ നിയമം (Idempotent Law) | $X + X = X$ | $X . X = X$ |
| 4 | വർഗപൂരക നിയമം (Involution Law) | $\overline{\overline{X}} = X$ | |
| 5 | പൂരകനിയമം (Complimentary Law) | $X + \overline{X} = 1$ | $X . \overline{X} = 0$ |
| 6 | ക്രമനിയമം (Commutative Law) | $X + Y = Y + X$ | $X . Y = Y . X$ |
| 7 | സംയോജനനിയമം (Associative Law) | $X + (Y + Z) = (X + Y) + Z$ | $X . (Y . Z) = (X . Y) . Z$ |
| 8 | വിതരണനിയമം (Distributive Law) | $X . (Y + Z) = X . Y + X . Z$ | $X + (Y . Z) = (X+Y) . (X+Z)$ |
| 9 | സ്വാംശീകരണനിയമം (Absorption Law) | $X + (X . Y) = X$ | $X . (X + Y) = X$ |

*പട്ടിക 2.25 ബൂളിയൻ നിയമങ്ങൾ*

നമ്മൾ ഇതുവരെ ചർച്ച ചെയ്ത എല്ലാ നിയമങ്ങളും ട്രൂത്ത് ടേബിളിലൂടെയാണ് തെളിയിച്ചിട്ടുള്ളത്. അവയിൽ ചിലത് മറ്റ് ചില നിയമങ്ങൾ ഉപയോഗിച്ച് തെളിയിക്കാനാകും. ഇങ്ങനെ തെളിയിക്കുന്ന രീതിയെ ബീജഗണിത രീതിയിലുള്ള തെളിവ് (Algebraic proof) എന്ന് വിളിക്കുന്നു. അവയിൽ ചിലത് നമുക്ക് നോക്കാം.

**i. $X.(X+Y)=X$ എന്ന് തെളിയിക്കുക. (അബ്സോർപ്ഷൻ നിയമം).**

LHS $= X.(X+Y)$

$= X.X + X.Y$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X + X.Y$ *(വർഗസമനിയമം (Idempotent law))*

$= X.(1+Y)$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണനവിതരണം (Distribution of multiplication over addition))*

$= X.1$ *(സങ്കലന അനന്യത (Additive identity))*

$= X$ *(ഗുണന അനന്യത (Multiplicative identity))*

$=$ RHS

**ii. $X+(X.Y)=X$ എന്ന് തെളിയിക്കുക. (അബ്സോർപ്ഷൻ നിയമം).**

LHS $= X+(X.Y)$

$= X.1 + X.Y$ *(ഗുണന അനന്യത (Multiplicative identity))*

$= X.(1+Y)$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X.1$ *(സങ്കലന അനന്യത (Additive identity))*

$= X$ *(ഗുണന അനന്യത (Multiplicative identity))*

$=$ RHS

**iii. $X+(Y.Z)=(X+Y).(X+Z)$ എന്ന് തെളിയിക്കുക. (വിതരണ നിയമം).**

പ്രസ്താവനയിലെ RHS ലെ പദപ്രയോഗം നമുക്ക് എടുക്കാം.

$(X+Y).(X+Z)$

$= (X+Y).X + (X+Y).Z$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X.(X+Y) + Z.(X+Y)$ *(ക്രമനിയമം (Commutative law))*

$= X.X + X.Y + Z.X + Z.Y$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X + X.Y + Z.X + Z.Y$ *(വർഗസമ നിയമം (Idempotent law))*

$= X.1 + X.Y + Z.X + Z.Y$ *(ഗുണന അനന്യത (Multiplicative identity))*

$= X.(1+Y) + Z.X + Z.Y$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X.1 + Z.X + Z.Y$ *(സങ്കലന അനന്യത (Additive identity))*

$= X.(1+Z) + Z.Y$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$= X.1 + Z.Y$ *(സങ്കലന അനന്യത (Additive identity))*

$= X + Y.Z$ *(സങ്കലനത്തിനുമേലുള്ള ഗുണന വിതരണം (Distribution of multiplication over addition))*

$=$ LHS

കിട്ടിയിരിക്കുന്ന പദപ്രയോഗം തന്നിരിക്കുന്ന പ്രസ്താവനയുടെ LHS ആകുന്നു. ആയതിനാൽ സിദ്ധാന്തം തെളിയിക്കപ്പെട്ടിരിക്കുന്നു.

## 2.8 ഡി മോർഗൻസ് സിദ്ധാന്തം (De Morgan's theorems)

ലണ്ടൻ യൂണിവേഴ്സിറ്റി കോളേജിലെ തർക്കശാസ്ത്ര വിദഗ്ധനും ഗണിത ശാസ്ത്രജ്ഞനുമായിരുന്ന അഗസ്റ്റസ് ഡി മോർഗൻ (1806-1871) സങ്കീർണമായ ബൂളിയൻ പദപ്രയോഗങ്ങൾ ലളിതമാക്കാൻ രണ്ടു സിദ്ധാന്തങ്ങൾ നിർദേശിച്ചു. ഈ സിദ്ധാന്തങ്ങൾ ഡി മോർഗൻസ് സിദ്ധാന്തങ്ങൾ എന്നറിയപ്പെടുന്നു. ഇവ താഴെ കൊടുത്തിരിക്കുന്നു.

(i) $\overline{X+Y} = \overline{X} \cdot \overline{Y}$

(ii) $\overline{X.Y} = \overline{X} + \overline{Y}$

ഈ സിദ്ധാന്തങ്ങൾ ഇങ്ങനെ പ്രസ്താവിക്കാം.

(i) 'ബൂളിയൻ വേരിയബിളുകളുടെ തുകയുടെ പൂരകവും അവയുടെ ഓരോന്നിന്റെയും പൂരകങ്ങളുടെ ഗുണനഫലവും തുല്യമായിരിക്കും.'

(ii) 'ബൂളിയൻ വേരിയബിളുകളുടെ ഗുണനഫലത്തിന്റെ പൂരകവും അവയുടെ ഓരോന്നിന്റെയും പൂരകങ്ങളുടെ തുകയും തുല്യമായിരിക്കും.'

**ഒന്നാമത്തെ സിദ്ധാന്തത്തിന്റെ ബീജഗണിത രീതിയിലുള്ള തെളിവ്.**

നമുക്ക് തെളിയിക്കേണ്ടത് $\overline{X+Y} = \overline{X} \cdot \overline{Y}$ എന്നാണ്.

$Z = X + Y$ ______ (1) എന്ന് നമുക്ക് അനുമാനിക്കാം.

എങ്കിൽ $\overline{Z} = \overline{X+Y}$ ______ (2)

$Z + \overline{Z} = 1$ ______ (3)

$Z \cdot \overline{Z} = 0$ ______ (4)

പൂരകനിയമപ്രകാരം സമവാക്യം (3) ഉം (4) ഉം ശരിയാണെന്നു നമുക്കറിയാം

(1), (3) എന്നീ സമവാക്യങ്ങൾ (2), (4) എന്നീ സമവാക്യങ്ങളിൽ യഥാക്രമം പ്രയോഗിച്ചാൽ താഴെ കാണുന്ന (5), (6) എന്നീ സമവാക്യങ്ങൾ കിട്ടുന്നു.

$(X + Y) + (\overline{X+Y}) = 1$ ______ (5)

$(X + Y) \cdot (\overline{X+Y}) = 0$ ______ (6)

ഡി മോർഗന്റെ ഒന്നാമത്തെ സിദ്ധാന്തം ശരിയാണെന്നു നമ്മൾ അനുമാനിക്കുക. അങ്ങനെയാണെങ്കിൽ (5), (6) എന്നീ സമവാക്യങ്ങളിൽ $(\overline{X+Y})$ എന്നതിന് പകരം $(\overline{X} . \overline{Y})$ എന്ന് കൊടുക്കാമല്ലോ. എങ്കിൽ (5), (6) എന്നീ സമവാക്യങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നതുപോലെ മാറ്റിയെഴുതാം.

$(X+ Y) + (\overline{X} . \overline{Y}) = 1$ ______ (7)

$(X + Y) . (\overline{X} . \overline{Y}) = 0$ ______ (8)

(7), (8) എന്നീ സമവാക്യങ്ങൾ ഓരോന്നും പ്രത്യേകം തെളിയിച്ചാൽ, ആ സമവാക്യങ്ങൾ രൂപീകരിക്കാൻ നമ്മൾ നടത്തിയ അനുമാനവും ശരിയാണെന്ന് നമുക്ക് പരിഗണിക്കാം. അതായത് (7), (8) സമവാക്യങ്ങൾ ശരിയാണെങ്കിൽ ഡി മോർഗന്റെ സിദ്ധാന്തവും ശരിയാണ്.

സമവാക്യം (7) ന്റെ LHS പരിഗണിക്കുക.

| | | |
|---|---|---|
| $(X + Y) + (\overline{X} . \overline{Y})$ | $= (X + Y + \overline{X}) . (X + Y + \overline{Y})$ | *(വിതരണ നിയമം (Distributive Law))* |
| | $= (X + \overline{X} + Y) . (X + Y + \overline{Y})$ | *(സംയോജന നിയമം (Associative Law))* |
| | $= (1 + Y) . (X + 1)$ | *(പൂരക നിയമം (Complimentary Law))* |
| | $= 1 . 1$ | *(സങ്കലന അനന്യത (Additive Identity))* |
| | $= 1$ | |
| | $= \text{RHS}$ | |

സമവാക്യം (8) ന്റെ LHS പരിഗണിക്കുക.

| | | |
|---|---|---|
| $(X + Y) . (\overline{X} . \overline{Y})$ | $= (X . \overline{X} . \overline{Y}) + (Y . \overline{X} . \overline{Y})$ | *(വിതരണ നിയമം (Distributive Law))* |
| | $= (X . \overline{X} . \overline{Y}) + (Y . \overline{Y} . \overline{X})$ | *(സംയോജന നിയമം (Associative Law))* |
| | $= (0 . \overline{Y}) + (0 . \overline{X})$ | *(പൂരക നിയമം (Complimentary Law))* |
| | $= 0 + 0$ | *(സങ്കലന അനന്യത (Additive Identity))* |
| | $= 0$ | |
| | $= \text{RHS}$ | |

(7), (8) എന്നീ സമവാക്യങ്ങൾ ബീജഗണിതത്തിലൂടെ നമ്മൾ തെളിയിച്ചിരിക്കുന്നു. ഇത് അർഥമാക്കുന്നത് ഡി മോർഗന്റെ ഒന്നാമത്തെ സിദ്ധാന്തം തെളിയിച്ചിരിക്കുന്നു എന്നാണ്. ഈ സിദ്ധാന്തം ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചും തെളിയിക്കാവുന്നതാണ്. അത് നിങ്ങൾ ചെയ്ത് പരിശീലിക്കുക.

**രണ്ടാമത്തെ സിദ്ധാന്തത്തിന്റെ ബീജഗണിത രീതിയിലുള്ള തെളിവ്**

നമുക്ക് തെളിയിക്കേണ്ടത് $\overline{X . Y} = \overline{X} + \overline{Y}$ എന്നാണ്

എന്ന് നമുക്ക് അനുമാനിക്കാം. $Z = X . Y$ _______ (11)

എങ്കിൽ, $\overline{Z} = \overline{X . Y}$ _______ (12)

$Z + \overline{Z} = 1$ _______ (13)

$Z . \overline{Z} = 0$ _______ (14)

പൂരക നിയമ പ്രകാരം സമവാക്യം (13) ഉം (14) ഉം ശരിയാണെന്നു നമുക്കറിയാം.

$(X . Y) + (\overline{X . Y}) = 1$ _______ (15)

$(X . Y) . (\overline{X . Y}) = 0$ _______ (16)

ഡി മോർഗന്റെ ഒന്നാമത്തെ സിദ്ധാന്തം ശരിയാണെന്ന് നമ്മൾ അനുമാനിക്കുക. അങ്ങനെയാണെങ്കിൽ (15), (16) എന്നീ സമവാക്യങ്ങളിൽ $(\overline{X . Y})$എന്നതിന് പകരം $(\overline{X} + \overline{Y})$ എന്ന് കൊടുക്കാമല്ലോ. എങ്കിൽ (15), (16) എന്നീ സമവാക്യങ്ങൾ താഴെ കൊടുത്തിരിക്കുന്നതുപോലെ മാറ്റിയെഴുതാം.

$(X . Y) + (\overline{X} + \overline{Y}) = 1$ _______ (17)

$(X . Y) . (\overline{X} + \overline{Y}) = 0$ _______ (18)

(17), (18) എന്നീ സമവാക്യങ്ങൾ ഓരോന്നും പ്രത്യേകം തെളിയിച്ചാൽ, ആ സമവാക്യങ്ങൾ രൂപീകരിക്കാൻ നമ്മൾ നടത്തിയ അനുമാനവും ശരിയാണെന്ന് നമുക്ക് പരിഗണിക്കാം. അതായത് (17), (18) സമവാക്യങ്ങൾ ശരിയാണെങ്കിൽ ഡി മോർഗന്റെ സിദ്ധാന്തവും ശരിയാണ്.

സമവാക്യം (17) ന്റെ LHS പരിഗണിക്കുക.

| | | |
|---|---|---|
| $(X.Y)+(\overline{X}+\overline{Y})$ | $=(\overline{X}+\overline{Y})+(X.Y)$ | *(ക്രമനിയമം (Commutative law))* |
| | $=(\overline{X}+\overline{Y}+X).(\overline{X}+\overline{Y}+Y)$ | *(വിതരണ നിയമം (Distributive Law))* |
| | $=(\overline{X}+X+\overline{Y}).(\overline{X}+\overline{Y}+Y)$ | *(സംയോജന നിയമം (Associative Law))* |
| | $=(1+\overline{Y}).(\overline{X}+1)$ | *(പൂരക നിയമം (Complimentary Law))* |
| | $=1.1$ | *(സങ്കലന അനന്യത (Additive Identity))* |
| | $=1$ | |
| | $=$ RHS | |

സമവാക്യം (18) ന്റെ LHS പരിഗണിക്കുക.

| | | |
|---|---|---|
| $(X.Y).(\overline{X}+\overline{Y})$ | $=(X.Y.\overline{X})+(X.Y.\overline{Y})$ | *(വിതരണ നിയമം (Distributive Law))* |
| | $=(X.\overline{X}.Y)+(X.Y.\overline{Y})$ | *(സംയോജന നിയമം (Associative Law))* |
| | $=(0.Y)+(X.0)$ | *(പൂരക നിയമം (Complimentary Law))* |
| | $=0+0$ | *(സങ്കലന അനന്യത (Additive Identity))* |
| | $=0$ | |
| | $=$ RHS | |

(17), (18) എന്നീ സമവാക്യങ്ങൾ ബീജഗണിതത്തിലൂടെ നമ്മൾ തെളിയിച്ചിരിക്കുന്നു. ഇത് അർഥമാക്കുന്നത് ഡി മോർഗന്റെ രണ്ടാമത്തെ സിദ്ധാന്തം തെളിയിച്ചിരിക്കുന്നു എന്നാണ്. ഈ സിദ്ധാന്തം ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചും തെളിയിക്കാവുന്നതാണ്. അത് നിങ്ങൾ ചെയ്ത് പരിശീലിക്കുക.

ചുവടെ കാണിച്ചിരിക്കുന്നതുപോലെ ഡി മോർഗൻന്റെ സിദ്ധാന്തം ഉപയോഗിച്ച് എത്ര വേരിയബിളുകളെയും നമുക്ക് വിപുലീകരിച്ചെഴുതാം.

$$\overline{A+B+C+D+\ldots} = \overline{A}.\overline{B}.\overline{C}.\overline{D}.\ldots$$

$$\overline{A.B.C.D\ldots} = \overline{A}+\overline{B}+\overline{C}+\overline{D}+\ldots$$

മുകളിൽ കൊടുത്തിരിക്കുന്ന പദപ്രയോഗങ്ങൾ ഡി മോർഗൻസ് സിദ്ധാന്തം ഉപയോഗിച്ചുള്ള രൂപമാറ്റത്തെ പ്രതിനിധീകരിക്കുന്നുണ്ടെങ്കിലും ചുവടെ കൊടുത്തിരിക്കുന്ന ഘട്ടങ്ങളിലൂടെ ഈ പ്രക്രിയയെ കുറെക്കൂടി ലളിതമാക്കാം.

(i) ഫൻക്ഷനെ മുഴുവനായും പൂരകമാക്കുക.

(ii) എല്ലാ AND (.) കളെയും OR (+) ആയും എല്ലാ OR (+) കളെയും AND (.) ആയും മാറ്റുക.

(iii) ഓരോ വേരിയബിളിനെയും പ്രത്യേകം പൂരകമാക്കുക. ഈ പ്രക്രിയയെ ഡിമോർഗനൈസേഷൻ (Demorganisation) എന്ന് വിളിക്കുന്നു. ലളിതമായി പറഞ്ഞാൽ ഡിമോർഗനൈസേഷൻ എന്നത് 'ചിഹ്നങ്ങൾ മാറ്റിക്കൊണ്ട് വാക്യങ്ങളെ വിഘടിപ്പിക്കുക' (Break the line cahnge the sign) എന്നാകുന്നു.

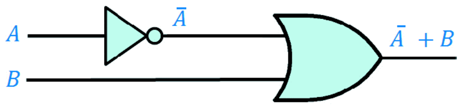
### സ്വയം വിലയിരുത്താം

1. A.B+B.C=1 എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ദ്വന്ദ്വങ്ങൾ കണ്ടുപിടിക്കുക.?
2. A+A=A എന്ന് പ്രസ്താവിക്കുന്ന ബൂളിയൻ നിയമത്തിന്റെ പേരെഴുതുക.
   (a) ക്രമ നിയമം (Commutative law) (b) വർഗസമ നിയമം (Idempotent Law) (c) സ്വാംശീകരണ നിയമം (Absorption Law)
3. ഡി മോർഗന്റെ സിദ്ധാന്തം പ്രസ്താവിക്കുക.

## 2.9 ലളിതമായ ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ സർക്യൂട്ട് രൂപകല്പന (Circuit designing for simple Boolean expression)

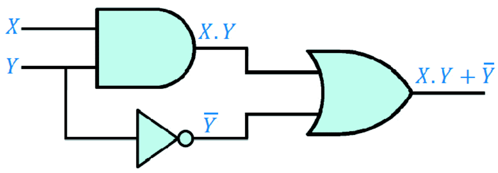
അടിസ്ഥാന ഗേറ്റുകൾ ഉപയോഗിച്ച് ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ സർക്യൂട്ട് രൂപകല്പന ചെയ്യാം. A.B എന്ന ബൂളിയൻ പദപ്രയോഗത്തെ പ്രതിനിധീകരിക്കുന്നതിന് AND ഗേറ്റും, A+B യെ പ്രതിനിധീകരിക്കുന്നതിന് OR ഗേറ്റും, $\overline{A}$ നെ പ്രതിനിധീകരിക്കുന്നതിന് NOT ഗേറ്റും ഉപയോഗിക്കാമെന്നു നമ്മൾ മനസ്സിലാക്കിക്കഴിഞ്ഞല്ലോ. അതുപോലെ മറ്റ് ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ സർക്യൂട്ട് രൂപകൽപ്പന ചെയ്യുന്നത് എങ്ങനെയാണെന്ന് നമുക്ക് നോക്കാം.

$\overline{A}+B$ എന്ന ബൂളിയൻ പദപ്രയോഗം പരിഗണിക്കുക. ഇതിൽ രണ്ട് ഇൻപുട്ടുകളുള്ള ഒരു ഓപ്പറേഷനും അതിൽ ഒരു ഇൻപുട്ട് വിപരീതമായതുമാണ് (Inverted). അതുകൊണ്ട് ഈ സർക്യൂട്ടിന്റെ രേഖാചിത്രം ചിത്രം 2.12 ൽ കാണിച്ചിരിക്കുന്നതുപോലെ വരയ്ക്കാം.



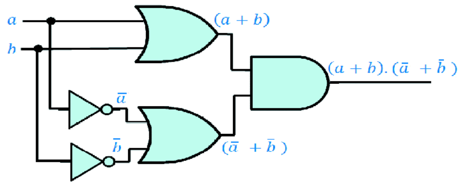

ചിത്രം 2.12 : $f(A, B) = \overline{A}+B$

**ഉദാഹരണം:** $f(X, Y) = X.Y + \overline{Y}$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ലോജിക്കൽ സർക്യൂട്ട് ഉണ്ടാക്കുക.



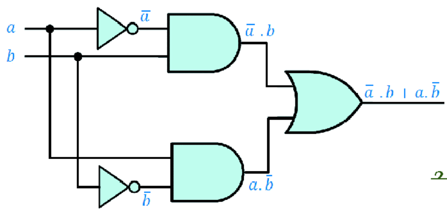

ചിത്രം 2.13 : $f(X, Y) = X.Y + \overline{Y}$

**ഉദാഹരണം:** $f(a, b) = (a + b) . (\overline{a} + \overline{b})$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ലോജിക്കൽ സർക്യൂട്ട് ഉണ്ടാക്കുക.




ചിത്രം 2.14 : $f(a, b) = (a + b) . (\overline{a} + \overline{b})$

**ഉദാഹരണം:** $\overline{a}.b + a.\overline{b}$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ലോജിക്കൽ സർക്യൂട്ട് ഉണ്ടാക്കുക.



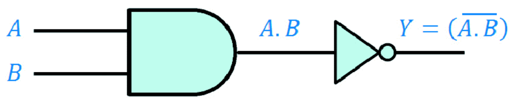

ചിത്രം 2.15 : $f(a, b) = \overline{a}.b + a.\overline{b}$

## 2.10 യൂണിവേഴ്സൽ ഗേറ്റ് (Universal gates)

NAND, NOR എന്നീ ഗേറ്റുകളാണ് യൂണിവേഴ്സൽ ഗേറ്റ് എന്ന് അറിയപ്പെടുന്നത്. മറ്റേതൊരു ഗേറ്റും ഉപയോഗിക്കാതെ ഏതൊരു ബൂളിയൻ ഫങ്ഷനും രൂപകല്പന ചെയ്യാൻ കഴിയുന്ന ഗേറ്റ് ആണ് യൂണിവേഴ്സൽ ഗേറ്റ്. ഭൂരിഭാഗം ഡിജിറ്റൽ ഇന്റഗ്രേറ്റഡ് ചിപ്പ് (IC) കളിലും അടിസ്ഥാന ഗേറ്റുകളായി ഉപയോഗിക്കുന്നത് NAND, NOR എന്നീ ഗേറ്റുകളാണ്, എന്തുകൊണ്ടെന്നാൽ ഈ ഗേറ്റുകൾ ചെലവുകുറഞ്ഞതും എളുപ്പത്തിൽ നിർമ്മിക്കാവുന്നതുമാണ്.

### 2.10.1 NAND ഗേറ്റ്

AND ഗേറ്റിന്റെ ഔട്ട്പുട്ട് NOT ഗേറ്റ് ഉപയോഗിച്ച് വിപരീതമാക്കുന്ന (Inverted) ഗേറ്റ് ആണ് NAND ഗേറ്റ്. ചിത്രം 2.16 NAND ഗേറ്റിന്റെ ലോജിക്കൽ സർക്യൂട്ട് സജ്ജീകരണം കാണിക്കുന്നു. A, B എന്നിവ AND ഗേറ്റിന്റെ ഇൻപുട്ടുകളും A.B എന്നത് അതിന്റെ ഔട്ട്പുട്ടുമായാൽ NAND ഗേറ്റിന്റെ ഔട്ട്പുട്ട് എന്നത് NOT ഗേറ്റ് ഉപയോഗിച്ച് വിപരീതമാക്കിയ $Y = (\overline{A.B})$ ആണ്. അതുകൊണ്ട് NAND ഗേറ്റിന്റെ ലോജിക്കൽ പദപ്രയോഗം $(\overline{A.B})$ എന്നാകുന്നു.

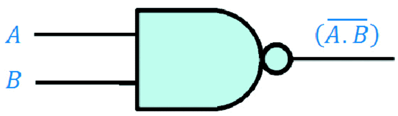



ചിത്രം 2.16 NAND ഗേറ്റിന്റെ ലോജിക്കൽ സർക്യൂട്ട്

| A | B | $Y = (\overline{A.B})$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

പട്ടിക 2.26 NAND ഗേറ്റിന്റെ ട്രൂത്ത് ടേബിൾ

NAND ഗേറ്റിന്റെ ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആയാൽ അതിന്റെ ഔട്ട്പുട്ടും 1 ആയിരിക്കുമെന്ന് ട്രൂത്ത് ടേബിളിലൂടെ (പട്ടിക 2.26) കാണിച്ചു തരുന്നു. എല്ലാ ഇൻപുട്ടുകളും 1 ആകുമ്പോൾ മാത്രമാണ് അതിന്റെ ഔട്ട്പുട്ട് 0 ആകുന്നത്. NAND ഗേറ്റ് എന്നത് AND ഗേറ്റിന്റെ വിപരീത പ്രവർത്തന മാണ് ചെയ്യുന്നത്. അതുകൊണ്ട് NAND ഗേറ്റ് ഇൻവെർട്ടഡ് (Inverted) AND ഗേറ്റ് എന്നറിയപ്പെടുന്നു.

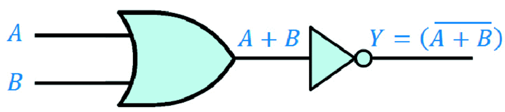



ചിത്രം 2.17 NAND ഗേറ്റ്

NAND ഗേറ്റിന്റെ പ്രതീകം ചിത്രം 2.17 ൽ കാണിച്ചിരിക്കുന്നു. AND ഗേറ്റിന്റെ പ്രതീകത്തിന്റെ ഔട്ട്പുട്ടിൽ ഒരു ചെറു വൃത്തം ചേർക്കുമ്പോൾ അത് NAND ഗേറ്റിന്റെ പ്രതീകം ആകും.

### 2.10.2 NOR ഗേറ്റ്

OR ഗേറ്റിന്റെ ഔട്ട്പുട്ട് NOT ഗേറ്റ് ഉപയോഗിച്ച് വിപരീതമാക്കുന്ന ഗേറ്റ് ആണ് NOR ഗേറ്റ്. ചിത്രം 2.18 ൽ NOR ഗേറ്റിന്റെ ലോജിക്കൽ സർക്യൂട്ട് സജ്ജീകരണം കാണിക്കുന്നു. A, B എന്നിവ OR ഗേറ്റിന്റെ ഇൻപുട്ടുകളും A+B എന്നത് അതിന്റെ ഔട്ട്പുട്ടുമായാൽ NOR ഗേറ്റിന്റെ ഔട്ട്പുട്ട് എന്നത് NOT ഗേറ്റ് ഉപയോഗിച്ച് വിപരീതമാക്കിയ $Y = (\overline{A+B})$ ആണ്. അതുകൊണ്ട് NOR ഗേറ്റിന്റെ ലോജിക്കൽ പദപ്രയോഗം $(\overline{A+B})$ എന്നാകുന്നു.



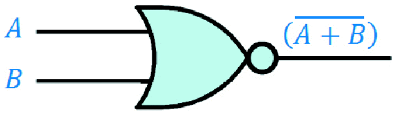

ചിത്രം 2.18 NOR ഗേറ്റിന്റെ ലോജിക്കൽ സർക്യൂട്ട്

NOR ഗേറ്റിൽ രണ്ട് ഇൻപുട്ടുകളും 0 ആകുമ്പോൾ മാത്രമാണ് ഇതിന്റെ ഔട്ട്പുട്ട് 1 ആകുന്നതെന്ന് ട്രൂത്ത് ടേബിളിലൂടെ (പട്ടിക 2.27) കാണിച്ചു തരുന്നു. ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആയാൽ അതിന്റെ ഔട്ട്പുട്ട് 0 ആയിരിക്കും. NOR ഗേറ്റ് എന്നത് OR ഗേറ്റിന്റെ വിപരീത പ്രവർത്തനമാണ് ചെയ്യുന്നത്. അതുകൊണ്ട് ഇതിനെ OR ന്റെ വിപരീതം (Inverted) എന്നുകൂടി പറയുന്നു.

| A | B | $Y = \overline{(A+B)}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

പട്ടിക 2.27 NOR ഗേറ്റിന്റെ ട്രൂത്ത് ടേബിൾ

NOR ഗേറ്റിന്റെ പ്രതീകം ചിത്രം 2.19 ൽ കാണിച്ചിരിക്കുന്നു. OR ഗേറ്റിന്റെ ഔട്ട്പുട്ടിൽ ചെറിയ വൃത്തത്തോടു കൂടിയ താണ് NOR ന്റെ പ്രതീകം.



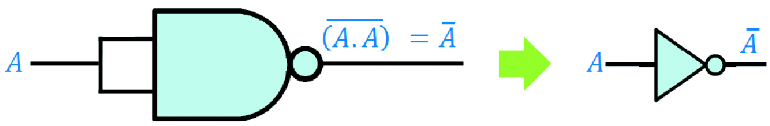

ചിത്രം 2.19 NOR ഗേറ്റ്

## 2.10.3 NAND ഗേറ്റും NOR ഗേറ്റും ഉപയോഗിച്ച് അടിസ്ഥാന ഗേറ്റുകളുടെ രൂപകല്പന (Implementation of basic gates using NAND and NOR)

NAND അല്ലെങ്കിൽ NOR ഗേറ്റ് ഉപയോഗിച്ച് എല്ലാ അടിസ്ഥാന ഗേറ്റുകളും നമുക്ക് രൂപകല്പന ചെയ്യാം. NAND ഗേറ്റ് ഉപയോഗിച്ച് അടിസ്ഥാന ഗേറ്റുകൾ രൂപകല്പന ചെയ്യുന്നത് എങ്ങനെയെന്ന് നമുക്ക് നോക്കാം.

### NAND ഗേറ്റ് ഉപയോഗിച്ച് NOT ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.20 ൽ കാണുന്നതുപോലെ ഒരു NAND ഗേറ്റിന്റെ എല്ലാ ഇൻപുട്ടുകളിലും ഒരേ വില നൽകികൊണ്ട് NOT ഗേറ്റിനെ പ്രതിനിധീകരിക്കാം.




ചിത്രം 2.20 NAND ഗേറ്റ് ഉപയോഗിച്ച് NOT ഗേറ്റിന്റെ പ്രതിനിധാനം

**തെളിവ്**

A NAND A = $\overline{(A.A)}$

എന്തുകൊണ്ടെന്നാൽ = $\overline{A}$ A.A = A

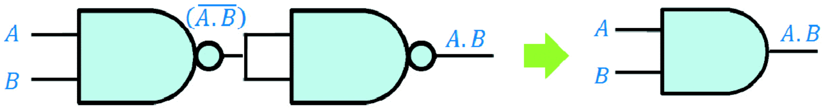
NAND ഗേറ്റ് കൊണ്ട് NOT ഗേറ്റ് ഉണ്ടാക്കുവാൻ കഴിയുമെന്ന് താഴെ കൊടുത്തിരിക്കുന്ന ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ (പട്ടിക 2.28) തെളിയിക്കുന്നു.

| A | AA | $\overline{(A.A)}$ | $\overline{A}$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

പട്ടിക 2.28 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്

## NAND ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.21 ൽ കാണുന്നതുപോലെ ഒരു NAND ഗേറ്റിനു തുടർച്ചയായി, ഔട്ട്പുട്ട് വിപരീതമാക്കുവാൻ മറ്റൊരു NAND ഗേറ്റ് ഉപയോഗിച്ച്, AND ഗേറ്റിനെ പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.21NAND ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

നമുക്കറിയാം A NAND B $= \overline{(A.B)}$

(A NAND B) NAND (A NAND B) $= \overline{(A.B)}$ NAND $\overline{(A.B)}$

$= \overline{(\overline{(AB)}\,.\,\overline{(AB)})}$

$= (\overline{\overline{(AB)}})$ എന്തുകൊണ്ടെന്നാൽ A.A = A

$=$ A.B എന്തുകൊണ്ടെന്നാൽ $\overline{\overline{(A)}} = A$

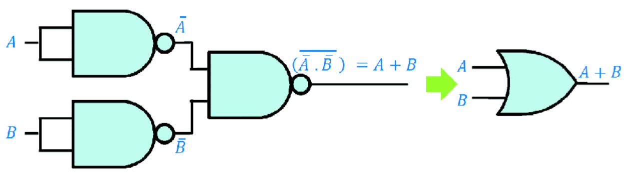
NAND ഗേറ്റ് കൊണ്ട് AND ഗേറ്റ് ഉണ്ടാക്കാൻ കഴിയുമെന്ന് ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ (പട്ടിക 2.29) തെളിയിക്കുന്നു.

| A | B | A.B | $\overline{(A.B)}$ | $\overline{(A.B)}\cdot\overline{(A.B)}$ | $\overline{(\overline{(AB)}\,.\,\overline{(AB)})}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |

*പട്ടിക 2.29 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## NAND ഗേറ്റ് ഉപയോഗിച്ച് OR ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.22 ൽ കാണുന്നതുപോലെ NAND ഗേറ്റിന്റെ എല്ലാ ഇൻപുട്ടുകളെയും പൂരകങ്ങൾ ആക്കി, OR ഗേറ്റിനെ NAND ഗേറ്റ് ഉപയോഗിച്ച് പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.22 NAND ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

$$A \text{ NAND } A = \overline{(A.A)}$$
$$= \overline{A}$$

അതുപോലെ, $B \text{ NAND } B = \overline{B}$

അതുകൊണ്ട്, $(A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B) = \overline{A} \text{ NAND } \overline{B}$

$$= \overline{(\overline{A}.\overline{B})}$$

$= \overline{\overline{A}} + \overline{\overline{B}}$ എന്തുകൊണ്ടെന്നാൽ $\overline{(A.B)} = \overline{A} + \overline{B}$

$= A + B$ എന്തുകൊണ്ടെന്നാൽ $\overline{\overline{(A)}} = A$

NAND ഗേറ്റ് കൊണ്ട് OR ഗേറ്റ് ഉണ്ടാക്കുവാൻ കഴിയുമെന്നതിന്റെ തെളിവ് ട്രൂത്ത് ടേബിളിൽ (പട്ടിക 2.30) കാണിച്ചിരിക്കുന്നു.

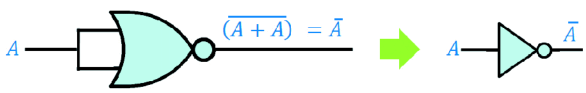
| A | B | $\overline{A}$ | $\overline{B}$ | $\overline{A}\cdot\overline{B}$ | $\overline{(\overline{A}.\overline{B})}$ | A + B |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |

*പട്ടിക 2.30 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

NAND ഗേറ്റ് ഒരു യൂണിവേഴ്സൽ ഗേറ്റ് ആകുന്നു. എന്തുകൊണ്ടെന്നാൽ ഇതുപയോഗിച്ച് AND, OR, NOT എന്നീ അടിസ്ഥാന ഗേറ്റുകൾ ഉണ്ടാക്കുവാൻ കഴിയുന്നു. NOR എന്ന മറ്റൊരു യൂണിവേഴ്സൽ ഗേറ്റ് ഉപയോഗിച്ച് അടിസ്ഥാന ഗേറ്റുകൾ എങ്ങനെ രൂപകല്പന ചെയ്യാം എന്ന് നോക്കാം.

## NOR ഗേറ്റ് ഉപയോഗിച്ച് NOT ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.23 ൽ കാണുന്നതുപോലെ ഒരു NOR ഗേറ്റിന്റെ രണ്ടു ഇൻപുട്ടുകളിലും ഒരേ വില നൽകികൊണ്ട്, NOT ഗേറ്റിനെ പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.23 NOR ഗേറ്റ് ഉപയോഗിച്ച് NOT ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

$A \text{ NOR } A = \overline{(A+A)}$

$= \overline{A}$ എന്തുകൊണ്ടെന്നാൽ

$A+A=A$

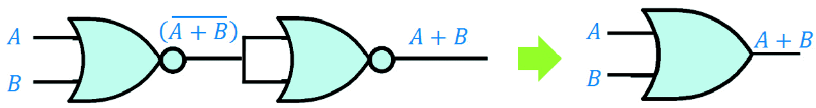
NOR ഗേറ്റ് കൊണ്ട് NOT ഗേറ്റ് ഉണ്ടാക്കാൻ കഴിയുന്നു എന്നതിന്റെ തെളിവ് ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ പട്ടിക 2.31 ൽ കൊടുത്തിരിക്കുന്നു.

| A | A+A | $\overline{(A+A)}$ | $\overline{A}$ |
|---|---|---|---|
| 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

*പട്ടിക 2.31 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## NOR ഗേറ്റ് ഉപയോഗിച്ച് OR ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.24 ൽ കാണുന്നതുപോലെ ഒരു NOR ഗേറ്റിനു തുടർച്ചയായി മറ്റൊരു NOR ഗേറ്റ് ഉപയോഗിച്ച്, OR ഗേറ്റിനെ പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.24 NOR ഗേറ്റ് ഉപയോഗിച്ച് OR ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

നമുക്കറിയാം $\text{A NOR B} = (\overline{A+B})$

അതുകൊണ്ട് $(\text{A NOR B}) \text{ NOR } (\text{A NOR B}) = (\overline{A+B}) \text{ NOR } (\overline{A+B})$

$$= \overline{((\overline{A+B})+(\overline{A+B}))}$$

$= ((\overline{\overline{A+B}}))$ എന്തുകൊണ്ടെന്നാൽ $A+A=A$

$= A+B$ എന്തുകൊണ്ടെന്നാൽ $(\overline{\overline{A}})=A$

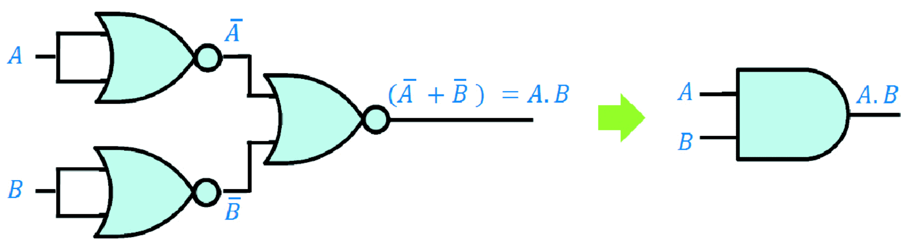
NOR ഗേറ്റ് കൊണ്ട് OR ഗേറ്റ് ഉണ്ടാക്കാൻ കഴിയുന്നു എന്നതിന്റെ തെളിവ് ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ പട്ടിക 2.32 ൽ കാണിച്ചിരിക്കുന്നു.

| A | B | A+B | $(\overline{A+B})$ | $(\overline{A+B}) + (\overline{A.B})$ | $\overline{((\overline{A+B}).(\overline{A+B}))}$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

*പട്ടിക 2.32 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## NOR ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം

ചിത്രം 2.25 ൽ കാണുന്നതുപോലെ NOR ഗേറ്റിന്റെ എല്ലാ ഇൻപുട്ടുകളും പൂരകങ്ങൾ ആക്കി, AND ഗേറ്റിനെ NOR ഗേറ്റ് ഉപയോഗിച്ച് പ്രതിനിധീകരിക്കാം.




*ചിത്രം 2.25 NOR ഗേറ്റ് ഉപയോഗിച്ച് AND ഗേറ്റിന്റെ പ്രതിനിധാനം*

**തെളിവ്**

$$A \text{ NOR } A = (\overline{A+A}) = \overline{A}$$

അതുപോലെ, $B \text{ NOR } B = (\overline{B+B}) = \overline{B}$

അതുകൊണ്ട്, $(A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B) = \overline{A} \text{ NOR } \overline{B}$

$$= \overline{(\overline{A}+\overline{B})}$$

$= (\overline{\overline{A}}).(\overline{\overline{B}})$ എന്തുകൊണ്ടെന്നാൽ $(\overline{A+B}) = \overline{A}.\overline{B}$

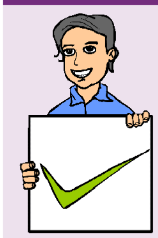
$= A.B$ എന്തുകൊണ്ടെന്നാൽ $\overline{\overline{A}} = A$

NOR ഗേറ്റ് ഒരു യൂണിവേഴ്സൽ ഗേറ്റ് ആകുന്നു എന്തുകൊണ്ടെന്നാൽ ഇതുപയോഗിച്ച് AND, OR, NOT എന്നീ അടിസ്ഥാന ഗേറ്റുകളുടെ പ്രവർത്തനങ്ങൾ നടപ്പിലാക്കുവാൻ കഴിയുന്നു. NOR ഗേറ്റ് കൊണ്ട് AND ഗേറ്റ് ഉണ്ടാക്കുവാൻ കഴിയുന്നു എന്നതിന്റെ തെളിവ് ട്രൂത്ത് ടേബിളിന്റെ സഹായത്തോടെ പട്ടിക 2.33 ൽ കാണിച്ചിരിക്കുന്നു.

| A | B | $\overline{A}$ | $\overline{B}$ | $\overline{A}+\overline{B}$ | $\overline{(\overline{A}+\overline{B})}$ | A.B |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |

*പട്ടിക 2.33 ട്രൂത്ത് ടേബിൾ ഉപയോഗിച്ചുള്ള തെളിവ്*

## നമുക്ക് പരിശോധിക്കാം



1. $X+\overline{Y}$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തിന്റെ ലോജിക്കൽ സർക്യൂട്ട് വരയ്ക്കുക.
2. യൂണിവേഴ്സൽ ഗേറ്റുകൾ ഏതൊക്കെയാണ്?.
3. ഏതെങ്കിലും ഒരു ഇൻപുട്ട് 1 ആകുമ്പോൾ ഔട്ട്പുട്ട് 0 തരുന്ന ഗേറ്റ് ______________ ആകുന്നു.
   (a) OR (b) AND (c) NAND (d) NOR
4. A NAND B = ______________.
   (a) A+B (b) A.B (c) $(\overline{A+B})$ (d) $(\overline{A.B})$

## നമുക്ക് സംഗ്രഹിക്കാം

വിവിധ രീതിയിലുള്ള ഡാറ്റ പ്രതിനിധീകരണത്തെ കുറിച്ച് ഈ അധ്യായത്തിൽ ചർച്ച ചെയ്തു. ഡാറ്റ പ്രതിനിധീകരണം ചർച്ച ചെയ്യുന്നതിനുമുമ്പ് വ്യത്യസ്ത സംഖ്യാ സമ്പ്രദായങ്ങളും അവയുടെ പരിവർത്തനങ്ങളും പരിചയപ്പെട്ടു. അതിനു ശേഷം പൂർണ സംഖ്യ, ഫ്ളോട്ടിങ് പോയിന്റ് സംഖ്യ, എന്നിവയുടെ പ്രതിനിധീകരണവും, ശബ്ദം, ചിത്രം, വീഡിയോ എന്നിവയുടെ പ്രതിനിധീകരണത്തിന്റെ വിവിധ രീതികളും നാം മനസ്സിലാക്കി. ബൂളിയൻ ബീജഗണിതത്തിലെ ആശയങ്ങൾക്കൊപ്പം ലോജിക്കൽ ഓപ്പറേറ്ററുകൾ, ഗേറ്റുകൾ, ബൂളിയൻ നിയമങ്ങൾ എന്നിവ നാം ചർച്ച ചെയ്തു. അടിസ്ഥാന ലോജിക് സർക്യൂട്ട് രൂപകല്പന ചെയ്യുന്ന രീതികൾ മനസ്സിലാക്കുന്നതോടൊപ്പം യൂണിവേഴ്സൽ ഗേറ്റിന്റെ പ്രാധാന്യവും ചർച്ച ചെയ്തുകൊണ്ടാണ് ഈ അധ്യായം അവസാനിപ്പിക്കുന്നത്.

## പഠനനേട്ടങ്ങൾ

ഈ അധ്യായം പൂർത്തിയാകുന്നതോടുകൂടി പഠിതാവിനു താഴെ പറയുന്നവ സാധ്യമാകും.

- വിവിധ സംഖ്യാ സമ്പ്രദായങ്ങളുടെ സവിശേഷതകൾ വിശദീകരിക്കാൻ.
- ഒരു സംഖ്യാ സമ്പ്രദായത്തിൽനിന്ന് മറ്റൊരു സംഖ്യാ സമ്പ്രദായത്തിലേക്ക് പരിവർത്തനം ചെയ്യാൻ.
- ബൂളിയൻ ഗണിതങ്ങൾ ചെയ്യാൻ.
- സംഖ്യകളും അക്ഷരങ്ങളും കമ്പ്യൂട്ടർ മെമ്മറിയിൽ പ്രതിനിധീകരിക്കാൻ.
- ശബ്ദം, ചിത്രം, വീഡിയോ എന്നീ ഫയലുകളുടെ ഘടനകൾ.
- ബൂളിയൻ ബീജഗണിതത്തിന്റെ ആശയങ്ങൾ.
- ലോജിക് ഓപ്പറേറ്ററുകളുടെയും ഗേറ്റുകളുടെയും പ്രവർത്തനങ്ങൾ ഉദാഹരണ സഹിതം വിശദീകരിക്കാൻ.
- ബൂളിയൻ ബീജഗണിതത്തിലെ അംഗീകൃതതത്വങ്ങൾ, സിദ്ധാന്തങ്ങൾ, നിയമങ്ങൾ എന്നിവ പ്രസ്താവിക്കാനും തെളിയിക്കാനും.
- ലളിതമായ അടിസ്ഥാന ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ സർക്യൂട്ട് രൂപകല്പന ചെയ്യാൻ.
- യൂനിവേഴ്സൽ ഗേറ്റുകൾ ഉപയോഗിച്ച് അടിസ്ഥാന ഗേറ്റുകളുടെ രൂപകല്പന.

## മാതൃകചോദ്യങ്ങൾ

**ഒറ്റ വാചകത്തിൽ ഉത്തരമെഴുതുക.**

1. $(296)_{10}$ എന്ന സംഖ്യയിൽ 9 ന്റെ സ്ഥാനവില എത്രയാണ്?
2. 55 എന്ന ദശസംഖ്യ (Decimal) ക്ക് തുല്യമായ അഷ്ടസംഖ്യ (Octal) കണ്ടുപിടിക്കുക.
3. താഴെ കൊടുത്തിരിക്കുന്ന ശ്രേണികളിൽ വിട്ടുപോയ സംഖ്യകൾ പൂരിപ്പിക്കുക.
   a) $101_2$, $1010_2$, $1111_2$, _______, ________
   b) $15_8$, $16_8$, $17_8$, _______, _______.
   c) $18_{16}$, $1A_{16}$, $1C_{16}$, _______, _______.
4. If $(X)_2 - (1010)_2 = (1000)_2$ ആയാൽ X ന്റെ വില കണ്ടുപിടിക്കുക.
5. ലോകത്തിലെ എല്ലാ ലിഖിതഭാഷകളിലെയും അക്ഷരങ്ങളെ പ്രതിനിധാനം ചെയ്യാൻ ഉപയോഗിക്കുന്ന സംവിധാനത്തിന്റെ പേരെഴുതുക.
6. താഴെ കൊടുത്തിരിക്കുന്നവയിൽ നിന്ന് ലോജിക്കൽ പ്രസ്താവനകൾ കണ്ടുപിടിക്കുക.
   a) നിങ്ങൾ എന്തുകൊണ്ടാണ് വൈകിയത്?
   b) നിങ്ങൾ എന്നോടൊപ്പം കമ്പോളത്തിൽ വരാൻ തയ്യാറാണോ?
   c) ഇന്ത്യ എന്റെ രാജ്യം ആകുന്നു.
   d) ക്ലാസ് മുറിയിലേക്ക് പോകുക.
7. മൂന്ന് അടിസ്ഥാന ഗേറ്റുകളുടെ പേരെഴുതുക.
8. ഇൻവെർട്ടർ എന്നറിയപ്പെടുന്ന ഗേറ്റ് ഏതാണ്?
9. രണ്ട് പൂരക നിയമങ്ങൾ എഴുതുക.
10. $\overline{(A+B)}$ എന്ന ബൂളിയൻ പദപ്രയോഗത്തെ ________ ഗേറ്റ് പ്രതിനിധാനം ചെയ്യുന്നു.
    a) AND b) NOR c) OR d) NAND

**ഒന്നോ രണ്ടോ വാക്യത്തിൽ ഉത്തരമെഴുതുക.**

1. ഡാറ്റ പ്രതിനിധാനം എന്ന പദം നിർവചിക്കുക.
2. സംഖ്യാ സമ്പ്രദായം എന്നത് കൊണ്ട് നിങ്ങൾ അർഥമാക്കുന്നത് എന്താണ്? ഏതെങ്കിലും നാല് സംഖ്യാ സമ്പ്രദായങ്ങൾ എഴുതുക.
3. താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകൾ മറ്റു മൂന്ന് സംഖ്യാ സമ്പ്രദായങ്ങളിലേക്കു മാറ്റിയെഴുതുക.
   a) $(125)_8$ b) 98 c) $(101110)_2$ d) $(A2B)_{16}$
4. താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകൾ മറ്റു മൂന്ന് സംഖ്യാ സംമ്പ്രദായങ്ങളിലേക്ക് പരിവർത്തനം ചെയ്യുക.
   a) $(7F.1)_{16}$ b) $(207.13)_8$ c) 93.25 d) $(10111011.1101)_2$

5. If $(X)_2 = (Y)_8 = (Z)_{16} = (28)_{10}$ ആയാൽ X, Y, Z എന്നിവയുടെ വില കണ്ടുപിടിക്കുക.

6. താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകൾ അവരോഹണക്രമത്തിലാക്കുക.

   a) $(101)_{16}$ b) $(110)_{10}$ c) $(111000)_2$ d) $(251)_8$

7. $(X)_2 = (10111)_2 + (11011)_2 - (11100)_2$ ആയാൽ X വില കണ്ടുപിടിക്കുക.

8. പൂർണസംഖ്യകൾ കമ്പ്യൂട്ടർ മെമ്മറിയിൽ പ്രതിനിധാനം ചെയ്യുവാൻ ഉപയോഗിക്കുന്ന രീതികൾ എന്തെല്ലാമാണ് ?

9. താഴെ കൊടുത്തിരിക്കുന്ന സംഖ്യകൾ ചിഹ്നവും മൂല്യവും, 1 ന്റെ പൂരകം, 2 ന്റെ പൂരകം എന്നീ രീതികളിൽ പ്രതിനിധീകരിക്കുക.

   a) -19 b) +49 c) -97 d) -127

10. ചിഹ്നവും മൂല്യവും രീതിയിൽ പ്രതിനിധാനം ചെയ്ത $(10011001)_2$ എന്ന സംഖ്യയുടെ പൂർണസംഖ്യ കണ്ടുപിടിക്കുക.

11. 32 ബിറ്റ് ഉപയോഗിക്കുന്ന കമ്പ്യൂട്ടറിൽ ഫ്ളോട്ടിങ് പോയിന്റ് സംഖ്യകൾ പ്രതിനിധാനം ചെയ്യുന്ന രീതി വിശദീകരിക്കുക.

12. കമ്പ്യൂട്ടർ മെമ്മറിയിൽ അക്ഷരങ്ങൾ പ്രതിനിധീകരിക്കുന്നതിനുള്ള രീതികൾ എന്തൊക്കെയാണ് ?

13. അക്ഷരങ്ങളുടെ പ്രതിനിധീകരണത്തിൽ യൂണികോഡിന്റെ പ്രാധാന്യം ചുരുക്കി വിവരിക്കുക.

14. ചേരുംപടി ചേർക്കുക:

| A | B |
|---|---|
| i) ഏതെങ്കിലും ഇൻപുട്ട് 1 ആയാൽ ഔട്ട്പുട്ട് 1 ആകുന്നു. | a) NAND |
| ii) ഏതെങ്കിലും ഇൻപുട്ട് 0 ആയാൽ ഔട്ട്പുട്ട് 0 ആകുന്നു. | b) OR |
| iii) ഏതെങ്കിലും ഇൻപുട്ട് 0 ആയാൽ ഔട്ട്പുട്ട് 1 ആകുന്നു. | c) NOR |
| iv) ഏതെങ്കിലും ഇൻപുട്ട് 1 ആയാൽ ഔട്ട്പുട്ട് 0 ആകുന്നു. | d) AND |

15. താഴെ കൊടുത്തിരിക്കുന്ന ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ ദ്വൈത (dual) രൂപങ്ങൾ എഴുതുക.

   a) $X.Y+Z$ b) $A.C+A.1+A.C$ c) $(A+0).(A.1.\overline{A})$

16. താഴെ കൊടുത്തിരിക്കുന്ന ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ പൂരകം (complement) എഴുതുക.

   a) $\overline{A}\,\overline{B}$ b) $\overline{A.B}+\overline{C.D}$

17. താഴെ കൊടുത്തിരിക്കുന്ന ബൂളിയൻ പദപ്രയോഗങ്ങളുടെ ലോജിക് സർക്യൂട്ട് നിർമിക്കുക.

   (i) $\overline{a}\,\overline{b}+c$ (ii) $ab+\overline{a}b+\overline{ab}$ (iii) $(a+\overline{b}).(\overline{a}+\overline{b})$

18. NAND, NOR എന്നീ ഗേറ്റുകളെ യൂണിവേഴ്സൽ ഗേറ്റുകൾ എന്ന് വിളിക്കുന്നത് എന്തുകൊണ്ടാണ്? ഉദാഹരണ സഹിതം സാധൂകരിക്കുക.

## ഉപന്യസിക്കുക

1. സംഖ്യകൾ കമ്പ്യൂട്ടർ മെമ്മറിയിൽ പ്രതിനിധീകരിക്കുന്നതിനുള്ള വിവിധ രീതികളെക്കുറിച്ച് വിശദീകരിക്കുക.
2. അക്ഷരങ്ങൾ കമ്പ്യൂട്ടർ മെമ്മറിയിൽ പ്രതിനിധീകരിക്കുന്നതിനുള്ള വിവിധ രീതികളെക്കുറിച്ച് വിശദീകരിക്കുക.
3. ശബ്ദം, ചിത്രം, വീഡിയോ എന്നീ ഫയലുകൾ കമ്പ്യൂട്ടറിൽ സംഭരിക്കുന്നതിന്റെ ഘടന വിവരിക്കുക.
4. മൂന്ന് ഇൻപുട്ടുകൾ ഉള്ള AND ഗേറ്റിന്റെ പ്രതീകം, ബൂളിയൻ പദപ്രയോഗം, ട്രൂത്ത് ടേബിൾ എന്നിവ എഴുതുക.
5. എല്ലാ അടിസ്ഥാന ഗേറ്റുകളും NOR ഗേറ്റ് ഉപയോഗിച്ച് നിർമ്മിച്ച് യൂനിവേഴ്സൽ ഗേറ്റ് ആണെന്ന് തെളിയിക്കുക.