Bináris műveletek gyakorlati alkalmazása: AND, OR, XOR

A bináris logikai műveletek (**AND**, **OR**, **XOR**) fontos szerepet játszanak a bitek kezelésében, mint például a bitek beállítása, kikapcsolása, illetve megfordítása. Az alábbiakban bemutatjuk, hogyan lehet ezeket a műveleteket felhasználni gyakorlati feladatok megoldására.

1. AND művelet

Az **AND** művelet csak akkor ad 1-es eredményt, ha mindkét bemenet 1. A bitek "kikapcsolására" használhatjuk, mert a 0 hatására minden egyes bitet 0-ra állít.

Igazságtábla:

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Példa: Adott egy 8 bites számunk: `10101101`. Ha csak az alsó négy bitet (jobbról négyet) akarjuk megtartani, akkor használjuk az AND műveletet:

```
  10101101
& 00001111
  --------
  00001101
```

Ez hasznos, ha egy adott bitcsoportot akarunk *maszkolni*, azaz megtartani a kívánt biteket, és lenullázni a többit.

2. OR (vagy) művelet

Az **OR** művelet akkor ad 1-es eredményt, ha legalább az egyik bemenet 1. Ezt a bitek "bekapcsolására" használhatjuk, mert a 0 nem változtatja meg az eredményt, de a 1-es beállítja az adott bitet 1-re.

Igazságtábla:

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Példa: Ha egy bitek közül egyet be akarunk állítani (pl. jobbról a negyediket), és a többit érintetlenül hagyni, akkor használjuk az OR-t:

```
  10101101
| 00001000
----------
  10101101
```

Ez hasznos, ha egy konkrét bitet szeretnénk 1-re állítani anélkül, hogy a többi bitet megváltoztatnánk.

3. XOR művelet

Az **XOR** (kizáró vagy) művelet akkor ad 1-et, ha a bemenetek eltérnek. Ezzel biteket kapcsolgathatunk, azaz ha egy bit értéke 1, akkor 0-ra vált, és fordítva.

- Példa: Ha az alsó 4 bit értékét akarjuk megváltoztatni:
    1. `10101101 ^ 00001111 = 10100010`.

Ez a művelet különösen hasznos, ha egy adott bitet meg akarunk “fordítani”.

Igazságtábla:

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Gyakorlati példa: Bitek beállítása és törlése

Tegyük fel, hogy van egy 8 bites regiszterünk: `R = 10101101`.

1. **Egy bit bekapcsolása (pl. 2. bit):**
    - Maszk: `00000100` (1 « 2)
    - Művelet: `R | Maszk`
    - Eredmény: `10101101 | 00000100 = 10101101` (a 2. bit már 1, nincs változás)

1. **Egy bit kikapcsolása (pl. 3. bit):**

- Maszk: `11110111` ~(1 « 3)
- Művelet: `R & Maszk`
- Eredmény: `10101101 & 11110111 = 10100101` (a 3. bit 0 lett)

1. **Egy bit átkapcsolása (pl. 0. bit):**

- Maszk: `00000001` (1 « 0)
- Művelet: `R ^ Maszk`
- Eredmény: `10101101 ^ 00000001 = 10101100` (a 0. bit megfordult)

Az igazságtáblák és a fenti példák segítségével megérthető, hogyan manipulálhatók a bitek a logikai műveletek segítségével. Ez alapvető fontosságú az alacsony szintű programozásban és a digitális áramkörök tervezésében.