**Compound assignment operators**

1. Bitwise **AND** operator (**&** Symbol)  
     
   **Use:**  
   Table of true: If any bits in the operation is equal to 0, the corresponding result of that bit will be 0. Useful when we need to disable bits in a register, force it to become zero or disable output register pins.   
     
   For example, if we need to filter or eliminate 4 Most significant bits of a 8 bit register, we can do it as follows:

8bit\_reg = 8bit\_reg & 0b00001111 Result: 8bit\_reg = 0b0000xxxx

Where “x” Could be any value from 0 to 1 depending of the previous value of the 8bit\_reg  
  
**Example:** Reg = 0b0011 & 0b0101  
**Result:** Reg = 0b0001  
  
In truth Table:

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Output (X)** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

1. Bitwise **OR** operator (**|** Symbol)

**Use:**  
Table of true: If any bits in the operation is equal to 1, the corresponding result of that bit will be 1. Useful when we need to enable bits in a register, force it to become one or enable output register pins.  
  
For example, if we need to set (Force to be logic one) the 4 Most significant bits of a 8 bit register, we can do it as follows: 8bit\_reg = 8bit\_reg | 0b00001111  
  
**Result:** 8bit\_reg = 0bxxxx1111  
  
Where “x” Could be any value from 0 to 1 depending of the previous value of the 8bit\_reg

**Example:** Reg = 0b0011 & 0b0101  
**Result:** Reg = 0b0111

In truth Table:

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Output (X)** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

1. Bitwise **XOR** operator (**^** Symbol)  
     
   **Use:**  
   Table of true: If any bits in the operation is equal to 1, the corresponding result of that bit will be inverted or switched. Useful when we need to switch an specific bit (from 0 to 1 or 1 to 0).

For example, if we need to switch 4 Least significant bits of a 8 bit register, we can do it as follows:   
8bit\_reg = 0b10101111  
8bit\_reg = 8bit\_reg ^ 0b00001111

Result: 8bit\_reg = 0b10100000  
  
**Example:** Reg = 0b0011 ^ 0b0101  
**Result:** Reg = 0b0110

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Output (X)** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

1. Bitwise **Left Shift** operator (**<<** Symbol)

**Use:**Useful when we need to move or displace a bit or specific bit to the left. For example, if we need to displace to the left 2 bits of a 8 bit register, we can do it as follows:  
  
8bit\_reg = 0b00100011

8bit\_reg = 8bit\_reg << 2  
  
**Result:** 8bit\_reg = 0b10001100

1. Bitwise **Right Shift** operator (**>>** Symbol)  
     
   **Use:**Works in the same way as the << symbol and its Useful when we need to move or displace a bit or specific bits to the right.  
     
   For example, if we need to displace to the left 2 bits of a 8 bit register, we can do it as follows:

8bit\_reg = 0b10001100

8bit\_reg = 8bit\_reg >> 2

**Result:** 8bit\_reg = 0b00100011