ALGEBRA DE BOOLE

El procesador del PLC se encarga de realizar las **operaciones lógicas o cálculos** entre los bits de memoria que dispone el PLC. Sin embargo, hacer cálculos no es exactamente lo mismo a lo que nosotros conocemos como sumar o restar en nuestro sistema de numeración. Las personas para poder hacer cálculos utilizamos 10 unidades (sistema decimal) que van del 0 al 9, las cuales podemos sumar, restar, multiplicar, dividir, etc. La información que posee el PLC sólo está disponible en forma de bits que pueden estar o bien apagados o bien prendidos. Esto para el PLC representa un sistema del tipo **binario** en el cual se asume que el estado apagado de un bit adopta el valor 0 y el estado prendido de un bit adopta el valor 1. Con esa convención se pueden especificar las operaciones lógicas y programarlas en el sistema operativo para que el PLC pueda aprovecharlas y hacer los cálculos. El conjunto de operaciones lógicas referidas a los estados de los bits es lo que denominamos **álgebra de Boole**.

Veamos cuáles son estas operaciones lógicas.

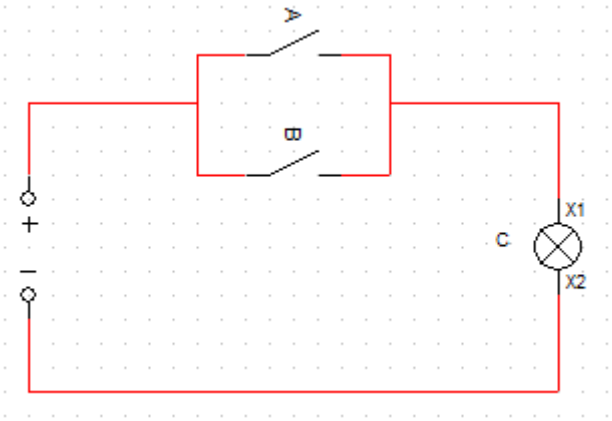
OPERACIÓN LÓGICA OR (SUMA DE BOOLE)

Si tomamos 3 bits, A, B y C y queremos realizar la operación lógica binaria del tipo OR (expresada como A + B = C) veremos que hay cuatro posibles resultados según los valores 0 y 1 que adopten A y B. A continuación, vemos detallados los resultados.

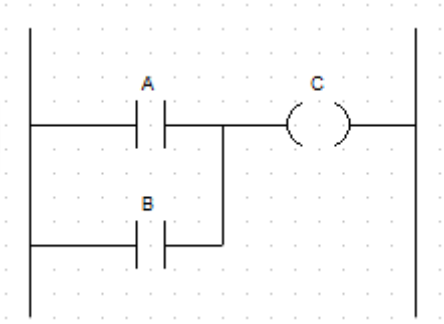
A + B = C
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1

Si comparamos con la suma como la conocemos en el sistema decimal, es prácticamente idéntica con excepción del resultado 1+1, recordemos que los sistemas de numeración son diferentes y las operaciones que estamos efectuando no son exactamente las mismas.

Estos resultados se pueden asimilar a dos contactos denominados A y B ubicados en **paralelo** que podemos abrir y cerrar, y una luz denominada C que prende si es que al menos uno de los dos contactos está cerrado. Para comprenderlo mejor lo podemos visualizar en el siguiente circuito.



Por ende decimos que la operación lógica OR arroja un resultado 1 (prendido) si es que hay al menos uno de los bits involucrados que tenga el valor 1 (prendido). Para el PLC sería como armar el siguiente programa en Ladder.



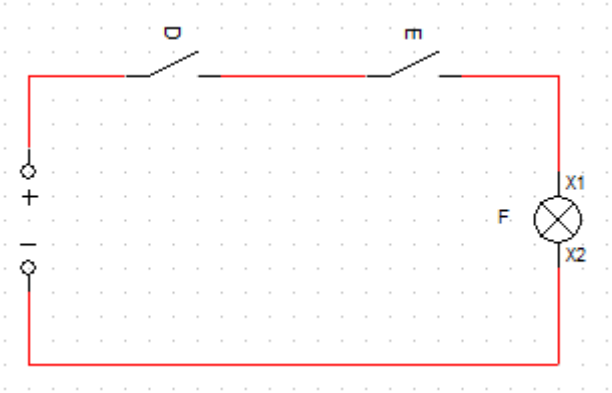
OPERACIÓN LÓGICA AND (PRODUCTO DE BOOLE)

Si tomamos 3 bits, D, E y F y queremos realizar la operación lógica binaria del tipo AND (expresada como D x E = F) veremos que hay cuatro posibles resultados según los valores 0 y 1 que adopten D y E. A continuación, vemos detallados los resultados.

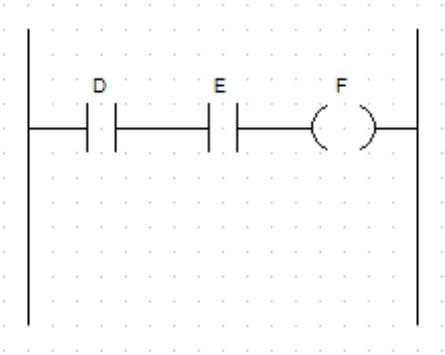
D x E = F
0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Si comparamos con el producto (o multiplicación) como la conocemos en el sistema decimal, es convenientemente idéntico, aún cuando los sistemas de numeración sean diferentes y las operaciones que estamos efectuando no son exactamente las mismas.

Estos resultados se pueden asimilar a dos contactos denominados D y E ubicados en **serie** que podemos abrir y cerrar, y una luz denominada F que prende si es que todos los contactos están cerrados. Para comprenderlo mejor lo podemos visualizar en el siguiente circuito.



Por ende decimos que la operación lógica AND arroja un resultado 1 (prendido) si es que todos los bits involucrados tienen el valor 1 (prendido). Para el PLC sería como armar el siguiente programa en Ladder.



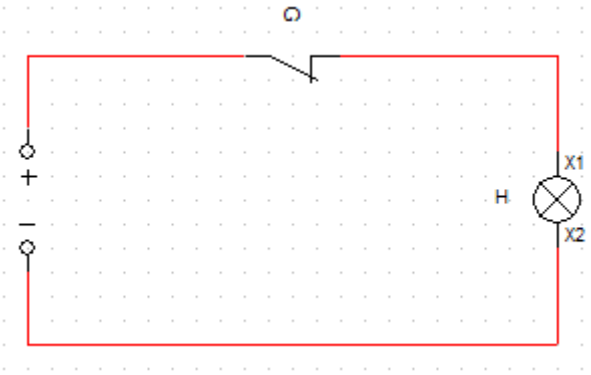
OPERACIÓN LÓGICA NOT (COMPLEMENTO DE BOOLE)

Si tomamos 2 bits, G y H y queremos realizar la operación lógica binaria del tipo NOT (expresada como $\overline{G} = H$) veremos que hay dos posibles resultados según los valores 0 y 1 que adopte G. A continuación vemos detallados los resultados.

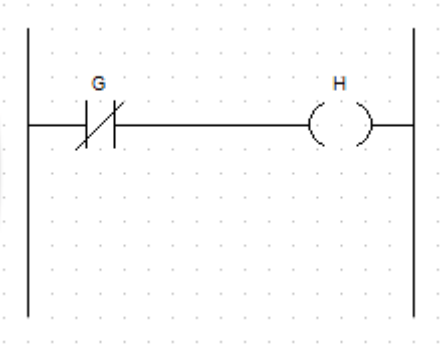
$\overline{G} = H$
0 = 1
1 = 0

En el sistema decimal no tenemos alguna operación asimilable a esta.

Estos resultados se pueden asimilar a un contacto denominado G del tipo **normal cerrado**, y una luz denominada H que prende si es que no accionamos el contacto y apaga cuando accionamos G. Para comprenderlo mejor lo podemos visualizar en el siguiente circuito.



Por ende decimos que la operación lógica NOT arroja un resultado 1 (prendido) cuando el bit involucrado tiene un valor 0 (apagado). Para el PLC sería como armar el siguiente programa en Ladder.



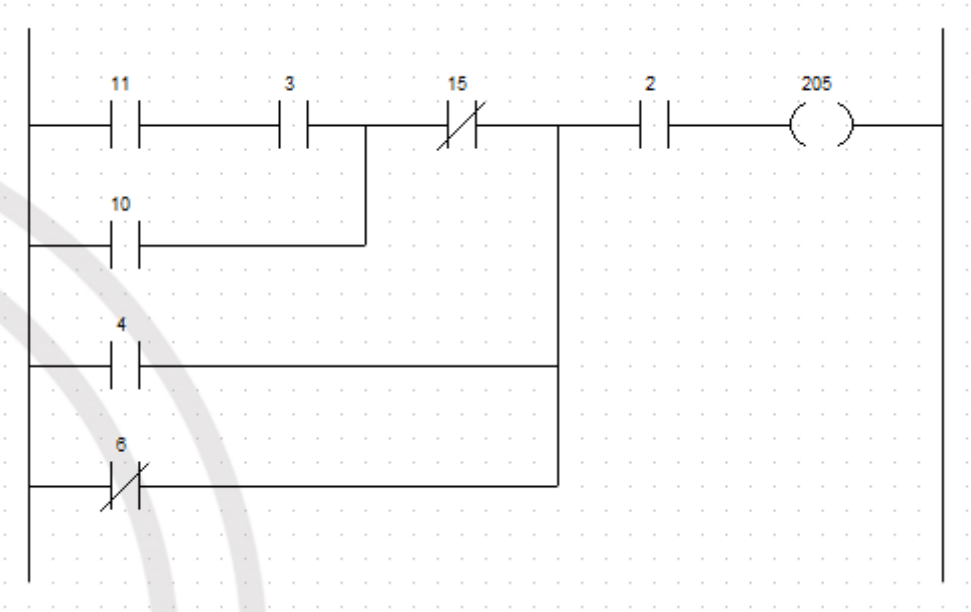
EL REGISTRO DE RESULTADOS

Los programas pueden tener muchos bits involucrados en líneas de programación con relaciones OR, AND y NOT entre ellos. Para poder llevar a cabo las operaciones lógicas entre todos ellos, el PLC se vale de un bit especial llamado el **registro de resultados** (algo así como las anotaciones que hacíamos al costado de la hoja cuando estábamos en primaria y nos hacían resolver cuentas con números grandes). El procesador escribe constantemente este registro de resultados para ir actualizando el resultado de las cuentas a medida que las resuelve.

Veamos el siguiente ejemplo. Supongamos que tenemos el siguiente programa en su representación en lista de pasos.

0 LOD 11
1 AND 3
2 OR 10
3 AND NOT 15
4 OR 4
5 OR NOT 6
6 AND 2
7 OUT 205

Este programa tendrá la siguiente representación en Ladder.



Y supongamos que el estado de los bits involucrados es

11 (prendido = 1)
3 (apagado = 0)
10 (prendido = 1)
15 (prendido = 1)
4 (apagado = 0)
6 (apagado = 0)
2 (prendido = 1)

El procesador deberá resolver en esa línea de programación que inicia en LOD 11 y culmina en OUT 205 los siguientes cálculos.

$11 \times 3 + 10 \times \overline{15} + 4 + \overline{6} \times 2 = 205$

El PLC siempre resuelve de izquierda a derecha a medida que se cargan las líneas de programación, acá no valen los conceptos matemáticos de agrupación de términos, paréntesis y demás, salvo que se especifique en el programa con otras instrucciones. Por otra parte, los números 11, 3, 10, 15, 4, 6, 2 y 205 son los nombres de los espacios de memoria del PLC, no son números decimales para sumar y multiplicar como en nuestro sistema de numeración decimal.

Resolvamos las operaciones lógicas una a una como lo haría el procesador del PLC.

11 x 3 + 10 x $\overline{15}$ + 4 + $\overline{6}$ x 2 = 205

Los datos que disponemos son 11 = 1 (prendido), 3 = 0 (apagado). Si reemplazamos:
1 x 0 = 0

Por ende anotamos un 0 en el registro de resultados (RR = 0)

RR + 10 x $\overline{15}$ + 4 + $\overline{6}$ x 2 = 205

Sabemos que RR = 0 (apagado, del cálculo anterior) y que 10 = 1 (prendido). Si reemplazamos:
0 + 1 = 1

Por ende anotamos un 1 en el registro de resultados (RR = 1)

RR x $\overline{15}$ + 4 + $\overline{6}$ x 2 = 205

Sabemos que RR = 1 (prendido) y que 15 = 1 (prendido). Pero en realidad es NOT 15, por ende, usamos su valor complemento 15 = 0. Si reemplazamos:
1 x 0 = 0

Entonces, anotamos un 0 en el registro de resultados (RR = 0)

RR + 4 + $\overline{6}$ x 2 = 205

Sabemos que RR = 0 (apagado) y que 4 = 0 (apagado). Si reemplazamos:
0 + 0 = 0

Entonces, anotamos un 0 en el registro de resultados (RR = 0)

RR + $\overline{6}$ x 2 = 205

Sabemos que RR = 0 (apagado) y que 6 = 0 (apagado). Su complemento es 6 = 1. Si reemplazamos:
0 + 1 = 1

Entonces, anotamos un 1 en el registro de resultados (RR = 1)

RR x 2 = 205

Sabemos que RR = 1 (prendido) y que 2 = 1 (prendido). Si reemplazamos:
1 + 1 = 1

Entonces, anotamos un 1 en el registro de resultados (RR = 1)

RR = 205

Llegamos al final y solo nos queda imponer el valor de RR en 205. Por ende 205 pasa al estado 1 y se prende.