

LAS INSTRUCCIONES BÁSICAS DE PROGRAMACIÓN

El programa que vamos a cargar en un equipo de PLC y que está pensado para resolver nuestra necesidad de automatización de una máquina puede ser **representado** de diversas formas. Mencionaremos a continuación 3 de ellas:

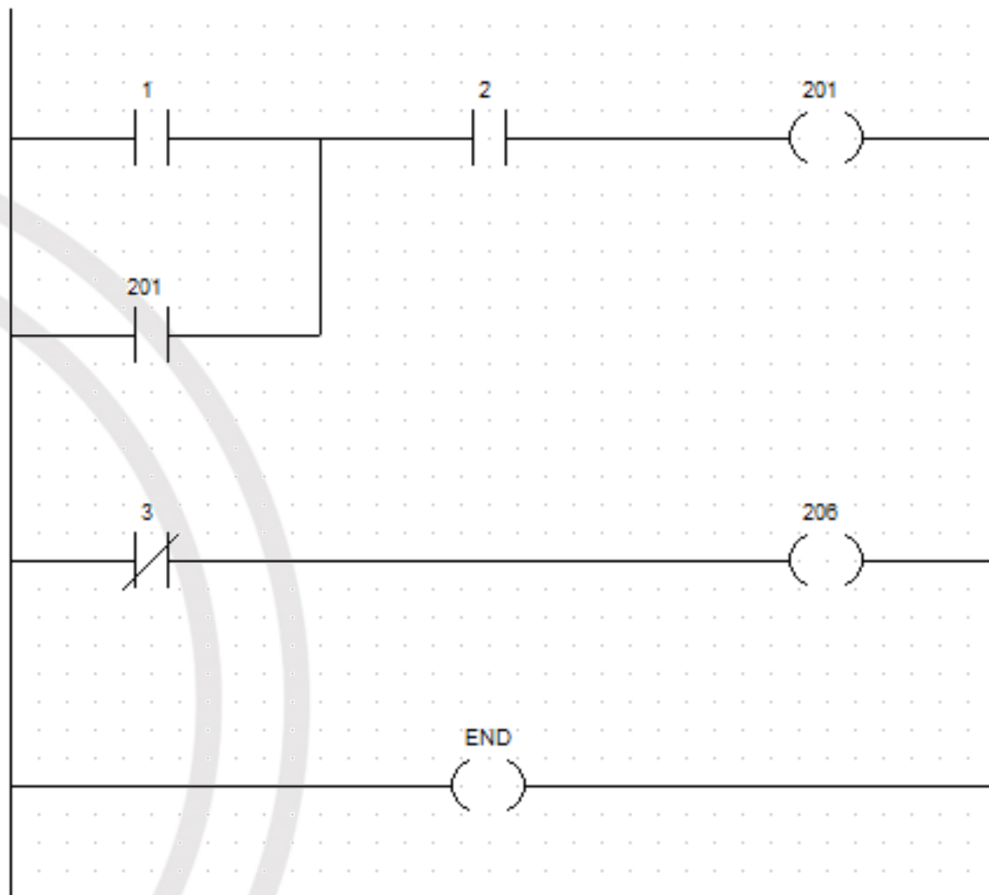
- Lenguaje FUP (representación en bloques de funciones)
- Lenguaje KOP (representación en Ladder o escalera)
- Lenguaje AWL (representación en lista de pasos)

Antes de realizar cualquier descripción, es necesario mencionar que el lenguaje FUP de representación en bloques de funciones NO lo estudiaremos en este curso. Consideramos que es un lenguaje más bien emparentado con las compuertas lógicas utilizadas en electrónica. En cambio, queremos aprender el lenguaje KOP de representación en Ladder que se asemeja mejor a los circuitos de lógica de relés, y el lenguaje AWL de representación en lista de pasos que es el que efectivamente se carga como programa en un PLC.

Pasemos entonces a explicar los lenguajes de programación que nos interesan y sus instrucciones más básicas.

LENGUAJE KOP O REPRESENTACIÓN EN LADDER

Este lenguaje se basa en representar las instrucciones que vamos a programar en un diagrama que se asemeja a los circuitos de lógica de relé y que, dada su construcción, tiene similitud con los peldaños de una escalera vertical de manos como las que usan los bomberos. Un programa representado en **Ladder** puede verse como el ejemplo siguiente.



La lectura del programa en representación Ladder es siempre empezando de la **esquina superior izquierda**, bajando hasta encontrar el primer peldaño (primera línea de programación) y avanzando a la derecha, siempre bajando cuando se encuentra un camino en paralelo. Al llegar al final del peldaño se prosigue bajando por la izquierda hasta encontrar el siguiente peldaño, ingresando y yendo hacia la derecha. Así hasta cubrir todos los peldaños.

En el ejemplo que se ve, empezaríamos ingresando por 1, luego bajando a 201, siguiendo por 2 y finalizando en 201. A continuación, ingresando por 3 y finalizando en 206. Por último, encontraríamos un peldaño con la instrucción END, indicando el fin de la representación Ladder.

Este lenguaje de programación se compone, en su forma más básica, de los siguientes elementos.

- Contactos



El contacto es el elemento que permite o no que pase la señal lógica (no nos referimos a señales eléctricas puesto que este diagrama es una representación del programa y no las conexiones físicas de los elementos conectados al PLC). El contacto se compone de la **representación del contacto** en sí (dos líneas paralelas) y del **espacio de memoria** que debe consultar (en este caso la entrada número 1 del PLC Izumi del laboratorio). Según el estado de la entrada física número 1 (prendida o apagada) este contacto tomará un valor (deja pasar señal o no deja pasar señal).

- Bobinas



La bobina es el elemento al cual se le envía la señal lógica resultante de haber pasado por todos los contactos (nuevamente no nos referimos a señales eléctricas). La bobina se compone de la **representación de la bobina** en sí (dos símbolos de paréntesis) y del **espacio de memoria** que debe consultar (en este caso la salida número 201 del PLC Izumi del laboratorio). Según si llega o no llega señal a la bobina, la salida física 201 pasará a estar prendida o apagada.

LENGUAJE AWL O REPRESENTACIÓN EN LISTA DE PASOS

Este lenguaje se basa en escribir las instrucciones de programación mediante el uso de las palabras reservadas por el sistema operativo para que este pueda entenderlas y calcular los resultados de la lectura y ejecución del programa. Un programa representado en **lista de pasos** puede verse como el ejemplo siguiente:

```
0 LOD 1
1 OR 201
2 AND 2
3 OUT 201
4 LOD NOT 3
5 OUT 206
6 END
```

Este lenguaje de programación se compone de los siguientes elementos:

- Número de paso

Es el primer número que se ve a la izquierda de la línea de programación y que indica el número de orden de esa línea de programación.

- Instrucción

Es la palabra reservada que el sistema operativo puede reconocer para entender cómo operar en esa línea de programación. Estas palabras pueden variar según el fabricante del equipo de PLC, por lo que es necesario consultar el manual para conocerlas.

- Espacio de memoria

Es el espacio de memoria que el PLC debe consultar para resolver la instrucción programada.

RELACIÓN ENTRE EL PROGRAMA REPRESENTADO EN LADDER Y EL PROGRAMA REPRESENTADO EN LISTA DE PASOS

Pasemos a entender por qué existen estos dos lenguajes y como relacionar uno con otro.

El lenguaje en **lista de pasos** (AWL) es el lenguaje que **el sistema operativo puede comprender**. Por ende, es fundamental que el programa se grabe en este lenguaje para que el sistema operativo lo lea y ejecute. Sin embargo, para el usuario del equipo del PLC, el lenguaje en lista de pasos puede ser muy engorroso, principalmente si hay una gran cantidad de líneas de programación. En contraste con esto, el **lenguaje en Ladder** (KOP) es el lenguaje que **el usuario puede comprender**, puesto que presenta una gran similitud con los circuitos de lógica de relés y es mucho más amigable de analizar.

Esto quiere decir que lo más natural es primero resolver el programa en Ladder para luego representar ese mismo programa en lista de pasos.

Los softwares de programación modernos son programados en Ladder y se transfieren al PLC. Lo que realmente se transfiere no es el programa en Ladder en sí, sino que el software de programación se encarga primero de traducirlo a lista de pasos (compila el programa KOP en AWL) y luego manda la lista de pasos al PLC. El usuario generalmente no necesita enterarse de cuál es la representación en lista de pasos del programa en Ladder.

Veamos entonces como se emparentan los programas en Ladder y en lista de pasos y analicemos las instrucciones más elementales de programación.

INSTRUCCIÓN LOD

EN LADDER



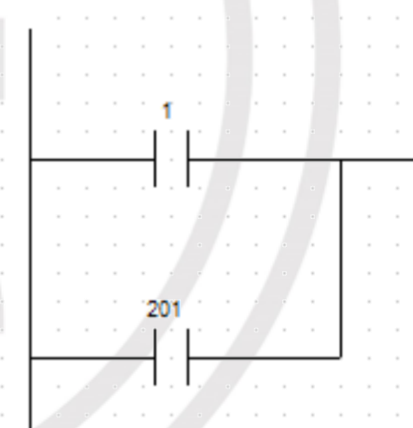
EN LISTA DE PASOS

0 LOD 1

La instrucción LOD provoca la **inicialización de una nueva línea** de programación. En Ladder corresponde a empezar un nuevo peldaño. Junto con la instrucción LOD se debe indicar cuál es el espacio de memoria que se debe leer (en este caso, el espacio de memoria de la entrada 1).

INSTRUCCIÓN OR

EN LADDER



EN LISTA DE PASOS

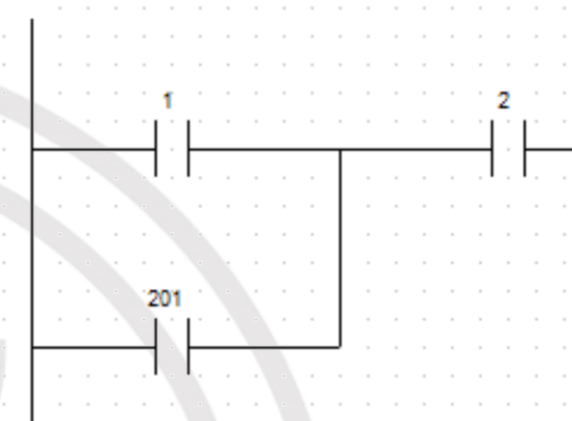
0 LOD 1

1 OR 201

La instrucción OR provoca que el espacio de memoria a programar quede en **paralelo** respecto a todo lo que ya se ha programado antes. En este caso, el espacio de memoria 201 queda en paralelo a todo lo anterior (el espacio de memoria 1).

INSTRUCCIÓN AND

EN LADDER



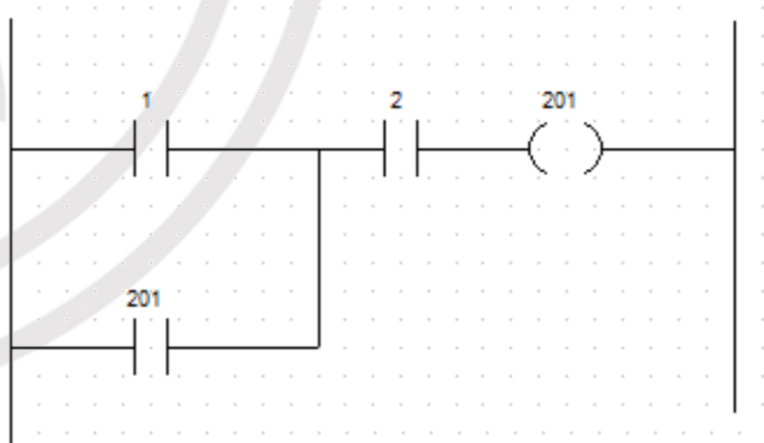
EN LISTA DE PASOS

0 LOD 1
1 OR 201
2 AND 2

La instrucción AND provoca que el espacio de memoria a programar quede en **serie** respecto a todo lo que ya se ha programado antes. En este caso, el espacio de memoria 2 queda en serie respecto a todo lo anterior (el paralelo entre 1 y 201).

INSTRUCCIÓN OUT

EN LADDER



EN LISTA DE PASOS

0 LOD 1
1 OR 201
2 AND 2
3 OUT 201

La instrucción OUT **finaliza** la línea de programación. De esta manera, el resultado de la señal que pasa a través de los contactos se dirige al espacio de memoria programado en esta instrucción (en este caso el espacio de memoria de salida 201).

INSTRUCCIÓN NOT

EN LADDER



EN LISTA DE PASOS

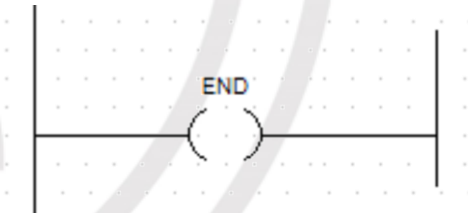
4 LOD NOT 3

La instrucción NOT es un **complemento** de las instrucciones anteriores (LOD, AND y OR). Puede programarse tanto LOD NOT, AND NOT y también OR NOT. La función que tiene esta instrucción es provocar el comportamiento **contrario** al que realmente tiene el estado de memoria programado. En este caso, si el estado del espacio de memoria de entrada 3 es prendido, este contacto NO envía señal, y si el estado del espacio de memoria de entrada 3 es apagado, este contacto SI envía señal.

Cabe aclarar que, si no se utiliza la instrucción NOT, el PLC siempre tomará como defecto que el contacto a programar es directo, no invertido.

INSTRUCCIÓN END

EN LADDER


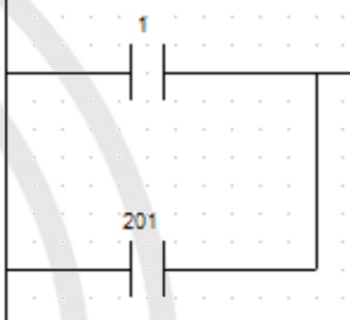

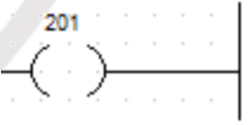
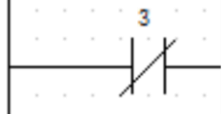
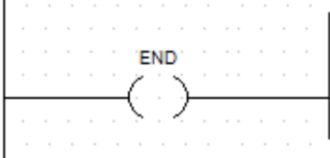


EN LISTA DE PASOS

6 END

Esta instrucción no tiene espacio de memoria asociada. Lo único que indica es la finalización de la lectura y ejecución del programa.

Veamos el siguiente cuadro que resume estas instrucciones:

Lista de pasos	Ladder	Comportamiento
LOD		<ul style="list-style-type: none"> - Hay una señal física en la entrada 1, este contacto manda señal. - No hay una señal física en la entrada 1, este contacto no manda señal.
OR		<ul style="list-style-type: none"> - Los contactos 1 y 201 tienen señal, este paralelo manda señal. - El contacto 1 tiene señal, el 201 no, este paralelo manda señal. - El contacto 1 no tiene señal, el 201 si, este paralelo manda señal. - Ni el contacto 1 ni el 201 tienen señal, este paralelo no manda señal.
AND		<ul style="list-style-type: none"> - Los contactos 1 y 2 tienen señal, esta serie manda señal. - El contacto 1 tiene señal, el 2 no, esta serie no manda señal. - El contacto 2 tiene señal, el 1 no, esta serie no manda señal. - Ni el contacto 1 ni el 2 tienen señal, esta serie no manda señal.
OUT		<ul style="list-style-type: none"> - La señal llega hasta 201, la salida física 201 se prende. - La señal no llega hasta 201, la salida física 201 no se prende.
LOD NOT		<ul style="list-style-type: none"> - Hay una señal física en la entrada 1, este contacto no manda señal. - No hay una señal física en la entrada 1, este contacto manda señal.
END		<ul style="list-style-type: none"> - Determina el fin del programa