UNIVERSIDADE DE VIGO

Departamento de Ingeniería de Sistemas y Automática

# PROGRAMACION DE AUTOMÁTAS
# STEP 7

UNIVERSIDADE DE VIGO

# Autómatas programables

- Elementos de un autómata (PLC)
  - Hardware (lo tangible, la circuitería, ...)
  - Software (programas, lo intangible)
    - Software del sistema (Sistema Operativo)
    - Programa de aplicación (Proyecto)

# STEP 7

- Variables e instrucciones básicas.
- Programación en bloques
- Temporizadores y contadores
- Señales analógicas
- Otras instrucciones (Salto condicional, incondicional, ..)

# STEP 7 : Características generales

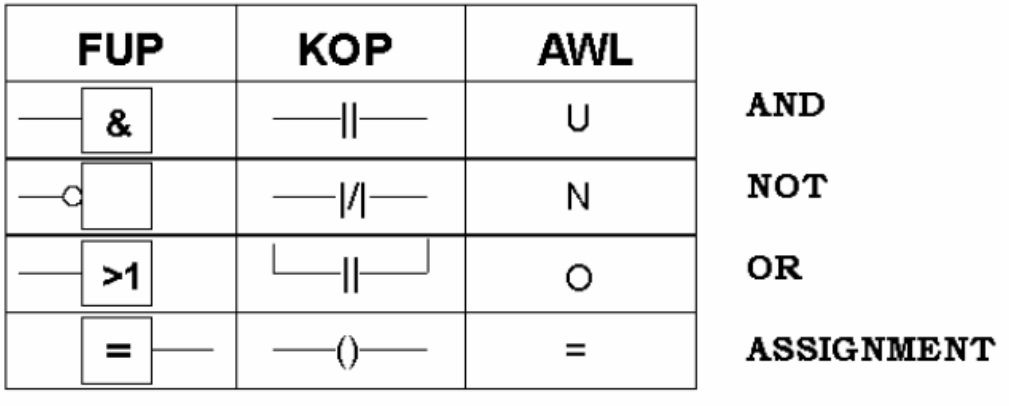
## TIPOS DE LENGUAJE

Literal

- Lista de instrucciones AWL o STL

Gráfico

- Esquema de contactos KOP
- Diagrama de funciones FUC

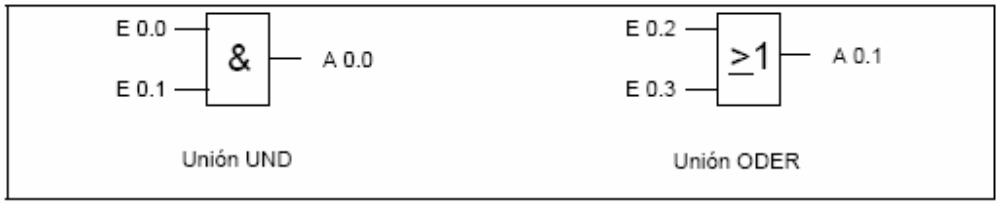


# Tipos de lenguaje

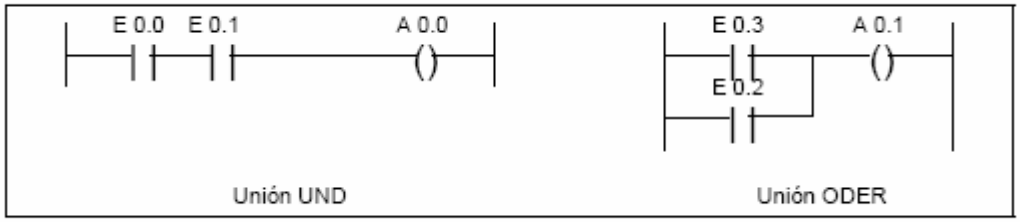
**AWL**

| Parte de la operación: | Parte del operando: Característica | Parámetro | |
|---|---|---|---|
| U | E | 0.0 | Unión UND |
| U | E | 0.1 | |
| = | A | 4.0 | |
| O | E | 0.2 | Unión ODER |
| O | E | 0.3 | |
| = | A | 4.1 | |

**FUP**




**KOP**



UNIVERSIDADE DE VIGO

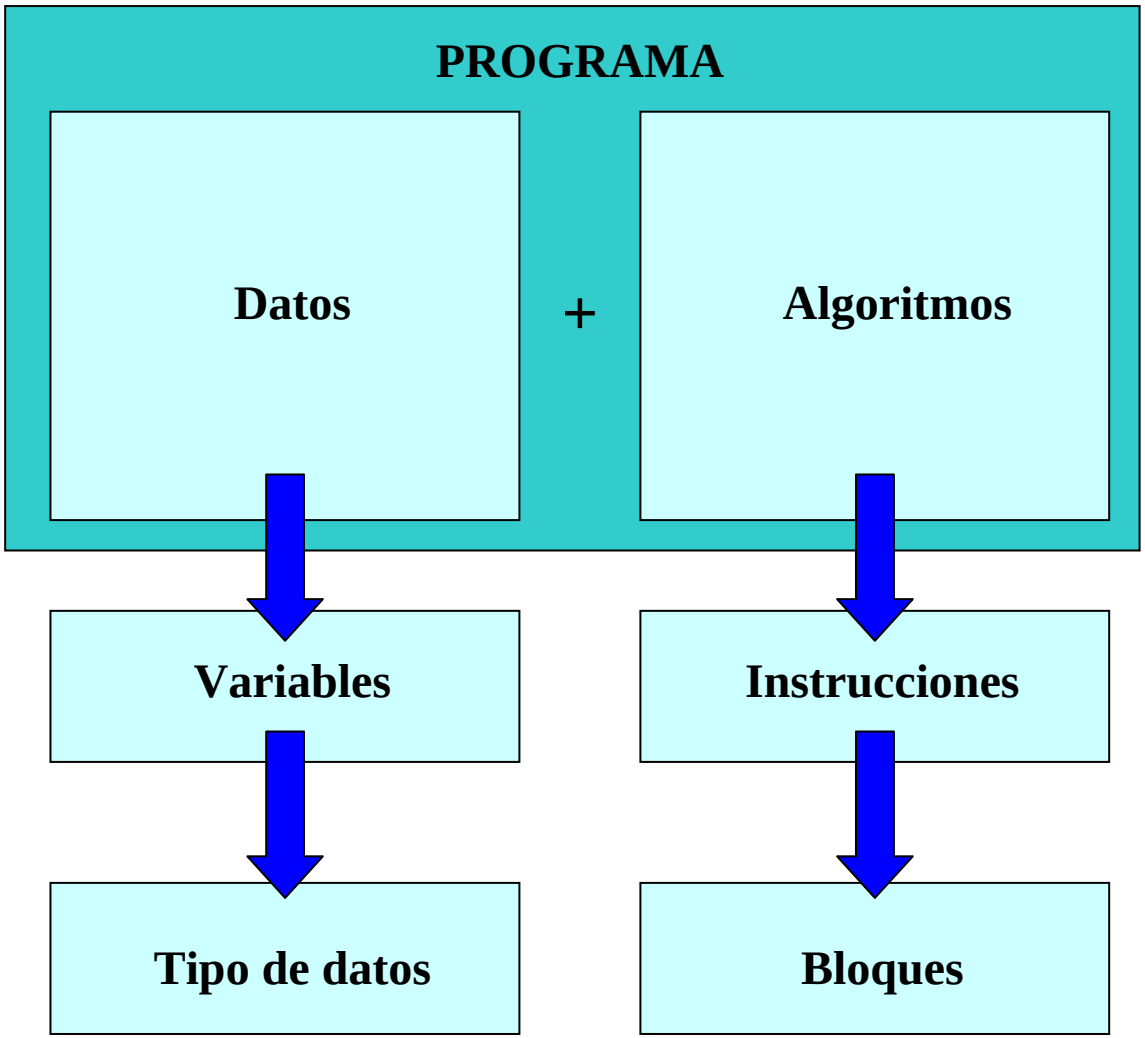
# Introducción a la programación

**ALGORITMO**: conjunto de operaciones preescrito de operaciones bien definidas para resolver un problema en un número finito de pasos.

**PROGRAMA**: Conjunto de símbolos y reglas para combinarlos que se usan para expresar algoritmos.

UNIVERSIDADE DE VIGO

# Conceptos generales



UNIVERSIDADE DE VIGO

# Variables

- Para el almacenamiento de datos se requieren variables.
- Se puede asignar diferentes tipos de datos
- Identificación de variables
    * Predefinidas (bit, byte o word)
        Entrada E n.m
        Salida A n.m
        Salida interna M n.m
    * No predefinidas
        El programador puede asignar un nombre y un tipo

UNIVERSIDADE DE VIGO

# Variables internas

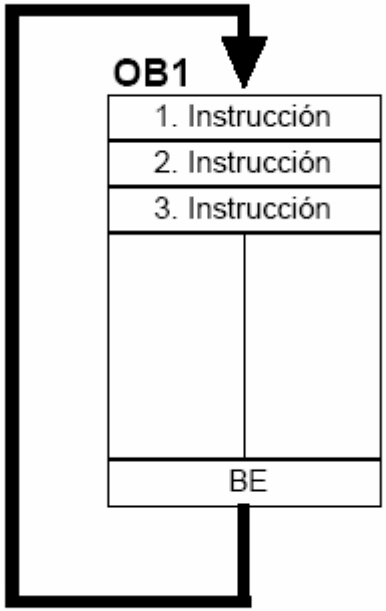
- MARCAS
  - Las marcas son bits internos de la CPU. Disponemos de una cantidad limitada de marcas. Esta cantidad depende de la CPU con la que estemos trabajando.
  - Estos bits podremos activarlos o desactivarlos como si fueran salidas. En cualquier punto del programa los podremos consultar.
  - A las marcas les llamaremos M. A continuación tenemos que decir a que bit en concreto nos estamos refiriendo. Por ejemplo tenemos las marcas, M 0.0, M 10.7, M 4.5, etc.

UNIVERSIDADE DE VIGO

# Tipos de datos

| Denominación | Tipo | Denominación | Tipo |
|---|---|---|---|
| **BOOL** | Binaria | **TIME** | Duración |
| **INT** | Entero con signo | **DATE** | Fecha |
| **DINT** | Entero doble con signo | **TIME_OF_DAY** | Hora del día |
| **REAL** | Real | **S5TIME** | Duración |
| **BYTE** | Conjunto de 8 bits | **DATE_AND_TIME** | Fecha y hora |
| **WORD** | Conjunto de 16 bits | **CHAR** | Carácter |
| **DWORD** | Conjunto de 32 bits | **STRING** | Cadena de caracteres |

# Step 7 : Lenguaje de instrucciones




**PROGRAMA = Conjunto de instrucciones**

**Instrucción = Operador + Modificador + Operando**

Ejemplo: U N E 0.0

| Instrucción de control | | |
|---|---|---|
| Parte de la operacón | Parte del operando | |
| | Característica | Parámetro |
| **U** | **E** | **0.0** |

UNIVERSIDADE DE VIGO

# Clases de instrucciones

- Instrucciones que operan con variables binarias independientes denominadas variables lógicas (bits)
- Instrucciones que operan con combinaciones binarias (octetos, palabras o dobles palabras)
- Instrucciones de temporización y contaje
- Instrucciones de control que permiten modificar la secuencia de ejecución de instrucciones del programa

UNIVERSIDADE DE VIGO

# Operaciones lógicas

- Operaciones lógicas con bits
  - Operaciones básicas
  - Instrucciones de terminación de cadenas lógicas
  - Combinación de operaciones básicas
  - Función memoria
  - Instrucciones que afectan al RLO
  - Operaciones que detectan cambios en el resultado lógico

UNIVERSIDADE DE VIGO

# Operaciones lógicas

- Las operaciones lógicas con bits operan con dos dígitos, 1 y 0.
  - Estos dos dígitos constituyen la base de un sistema numérico denominado sistema binario. Los dos dígitos 1 y 0 se denominan dígitos binarios o bits.
  - En el ámbito de los contactos y bobinas, un 1 significa activado ("conductor") y un 0 significa desactivado ("no conductor").
- Las operaciones lógicas con bits interpretan los estados de señal 1 y 0, y los combinan de acuerdo con la lógica del Álgebra de Boole.
  - Estas combinaciones producen un 1 ó un 0 como resultado y se denominan "resultado lógico" (RLO). Las operaciones lógicas con bits permiten ejecutar las más diversas funciones.

# Operaciones lógicas

1.- Las operaciones básicas para las operaciones lógicas con bits son:

| | |
|---|---|
| U | Y |
| UN | Y-No |
| O | O |
| ON | O-No |
| X | O-exclusiva |
| XN | O-exclusiva-No |

2.- Para terminar una cadena lógica se puede utilizar una de las tres operaciones:

| | |
|---|---|
| = | Asignar |
| R | Desactivar |
| S | Activar |

3.- Las siguientes operaciones permiten ejecutar una cadena lógica encerrada entre paréntesis:

| | |
|---|---|
| U( | Y con abrir paréntesis |
| UN( | Y-No con abrir paréntesis |
| O( | O con abrir paréntesis |
| ON( | O-No con abrir paréntesis |
| X( | O-exclusiva con abrir paréntesis |
| XN( | O-exclusiva-NO con abrir paréntesis |
| ) | Cerrar paréntesis |

4.- Las operaciones siguientes permiten modificar el resultado lógico (RLO):

| | |
|---|---|
| NOT | Negar el RLO |
| SET | Activar el RLO (=1) |
| CLR | Desactivar RLO (=0) |
| SAVE | Memorizar el RLO en el registro RB |

5.- Otras operaciones detectan cambios en el resultado lógico y reaccionan correspondientemente:

| | |
|---|---|
| FN | Flanco negativo |
| FP | Flanco positivo |

# Operaciones básicas

**Formato**

**U <bit>**

| Operando | Tipo de datos | Area de memoria |
|---|---|---|
| <bit> | BOOL | E, A, M, L, D, T, Z |

**Descripción de la operación**

**U** consulta el bit direccionado para saber si tiene el estado de señal "1", y combina el resultado de la consulta con el RLO realizando una Y lógica.

**Formato**

**UN <bit>**

| Operando | Tipo de datos | Area de memoria |
|---|---|---|
| <bit> | BOOL | E, A, M, L, D, T, Z |

**Descripción de la operación**

**UN** consulta el bit direccionado para saber si tiene el estado de señal "0" y combina el resultado de la consulta con el RLO realizando una Y lógica.

# Operaciones básicas

**Formato**

O <bit>

| Operando | Tipo de datos | Area de memoria |
|---|---|---|
| <Bit> | BOOL | E, A, M, L, D, T, Z |

**Descripción de la operación**

O consulta el bit direccionado para saber si tiene el estado de señal "1", y combina el resultado de la consulta con el RLO realizando una O lógica.

**Formato**

ON <bit>

| Operando | Tipo de datos | Area de memoria |
|---|---|---|
| <bit> | BOOL | E, A, M, L, D, T, Z |

**Descripción de la operación**

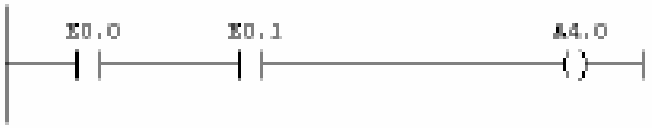
ON consulta el bit direccionado para saber si tiene el estado de señal "0", y combina el resultado de la consulta con el RLO realizando una O lógica.

# Operaciones básicas

- Serie

SOLUCIÓN EN AWL

```
U   E   0.0
U   E   0.1
=   A   4.0
```

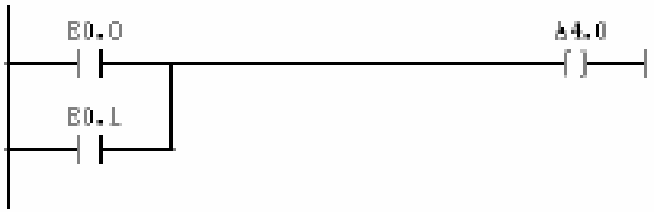



- Paralelo

SOLUCIÓN EN AWL

```
U   E   0.0
O   E   0.1
=   A   4.0
```

(también O E 0.0)

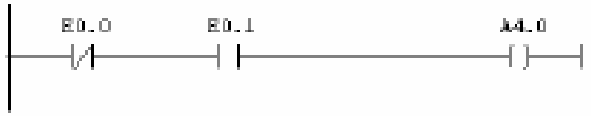


# Operaciones básicas

- Contactos negados

SOLUCIÓN EN AWL

```
UN    E    0.0
U     E    0.1
=     A    4.0
```



# Instrucción de terminación de cadenas lógicas

**Formato**

= <bit>

| Operando | Tipo de datos | Area de memoria |
|---|---|---|
| <bit> | BOOL | E, A, M, L, D, T, Z |

**Descripción de la operación**

= <bit> escribe el RLO en el bit direccionado si el Master Control Relay está conectado (MCR = 1). Si el MCR es 0, en el bit direccionado se escribe el valor "0" en vez del RLO.

# Instrucción de terminación de cadenas lógicas

**Formato**

R <bit>

| Operando | Tipo de datos | Area de memoria |
|---|---|---|
| <bit> | BOOL | E, A, M, L, D |

**Descripción de la operación**

R (Desactivar bit) escribe el valor "0" en el bit direccionado si el RLO es 1 y si el Master Control Relay (MCR = 1) está conectado. Si el MCR es 0, el bit direccionado no varía.

**Formato**

S <bit>

| Operando | Tipo de datos | Area de memoria |
|---|---|---|
| <bit> | BOOL | E, A, M, L, D |

**Descripción de la operación**

S (Activar bit) escribe el valor "1" en el bit direccionado si el RLO es 1 y si el Master Control Relay (MCR = 1) está conectado. Si el MCR es 0, el bit direccionado no varía.

# Instrucción de terminación de cadenas lógicas

**Formato**

= <bit>

| Operando | Tipo de datos | Area de memoria |
|---|---|---|
| <bit> | BOOL | E, A, M, L, D, T, Z |

**Descripción de la operación**

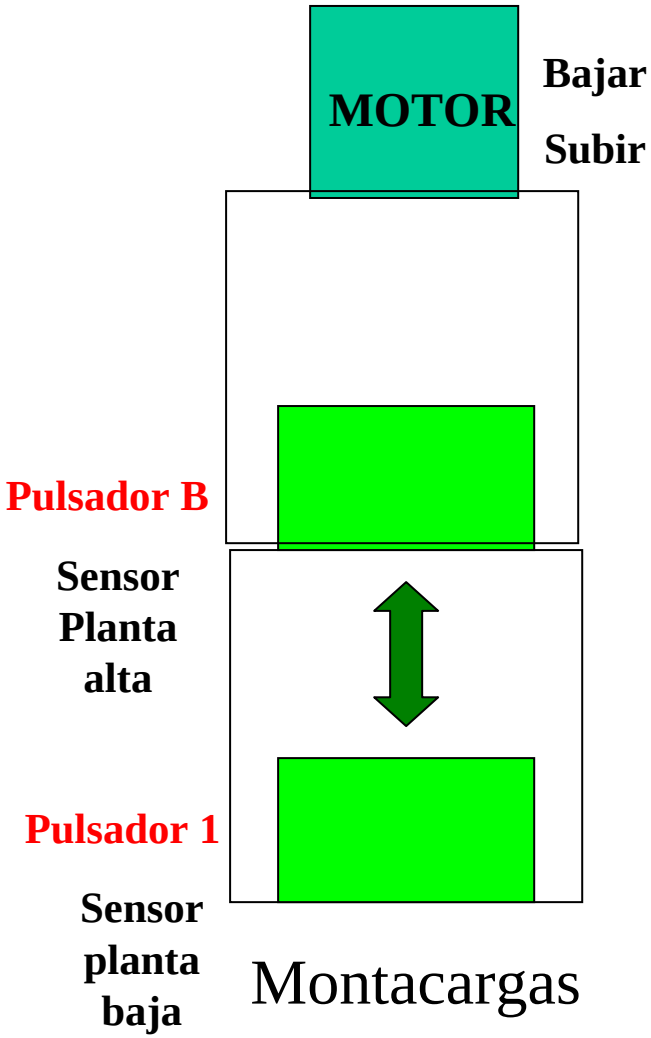
= <bit> escribe el RLO en el bit direccionado si el Master Control Relay está conectado (MCR = 1). Si el MCR es 0, en el bit direccionado se escribe el valor "0" en vez del RLO.

UNIVERSIDADE DE VIGO

# Instrucción de terminación de cadenas lógicas

- Las instrucciones SET y RESET son instrucciones de memoria.
- Si programamos un SET de una salida o de una marca con unas condiciones, se activará cuando se cumplan dichas condiciones. Aunque las condiciones dejen de cumplirse, no se desactivará hasta que se haga un RESET de la salida o marca.
- Estas instrucciones tienen prioridad. Dependen del orden en que las programemos. Siempre va a tener prioridad la última que programemos.
- En nuestro caso, si hacemos un SET y un RESET dentro del mismo ciclo de scan, al final de cada ciclo hará efecto lo último que hayamos programado.

# Ejemplo : El montacargas



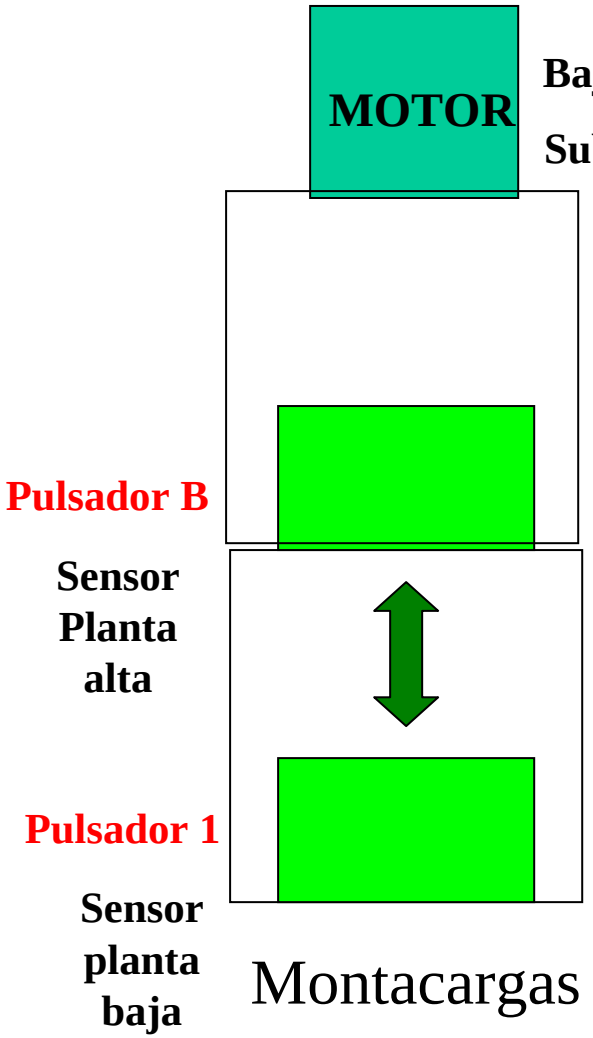

| SALIDAS (A) | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

| ENTRADAS (E) | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

**Especificación 1:**

Si el montacargas está en la planta baja **y** el Pulsador 1 esta activo el montacargas deberá subir.

# Ejemplo : El montacargas

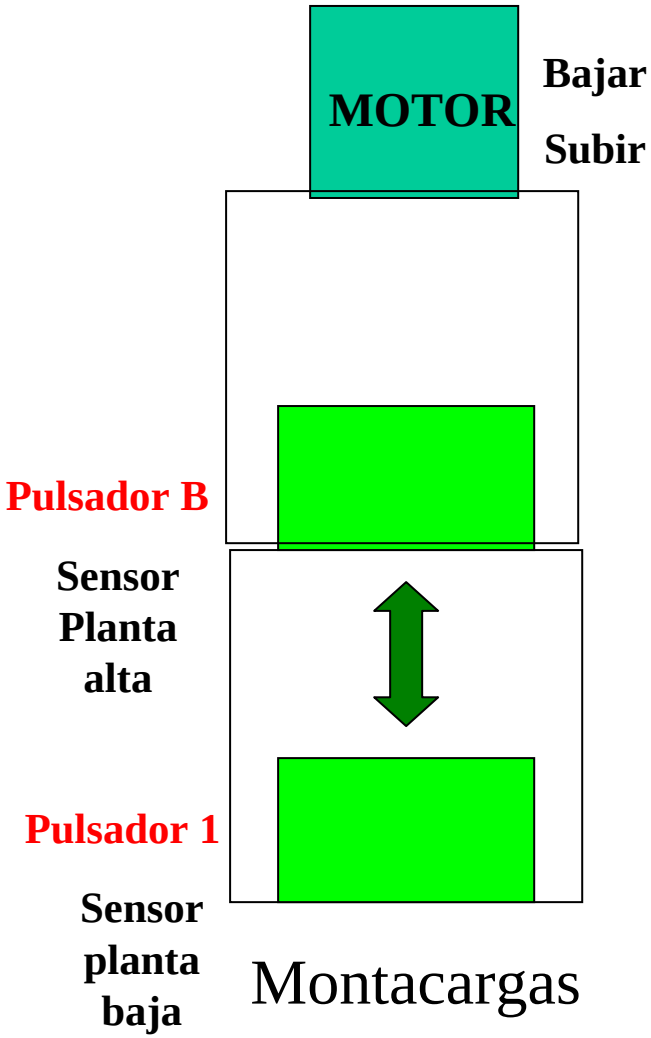



| SALIDAS (A) | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

| ENTRADAS (E) | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

| | |
|---|---|
| U E 0.3 | Sensor planta baja |
| U E 0.2 | Pulsador 1 |
| S A 4.1 | Subir |

# Ejemplo : El montacargas



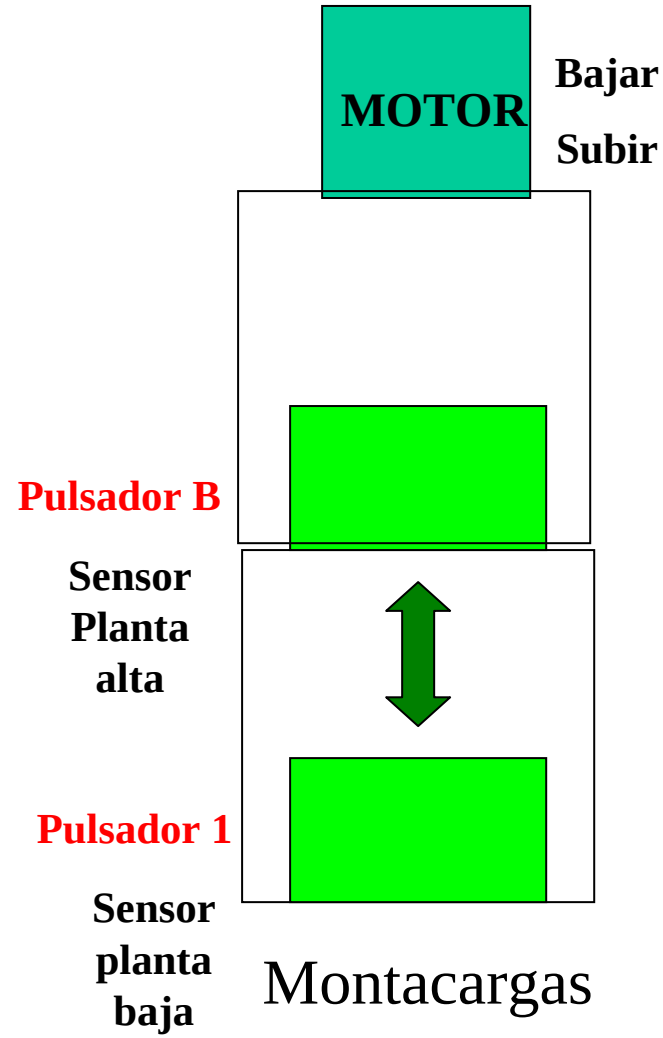

| SALIDAS (A) | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

| ENTRADAS (E) | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

**Especificación 2:**

Si el montacargas está en la planta alta **y** el Pulsador B esta activo el montacargas deberá bajar.

# Ejemplo : El montacargas

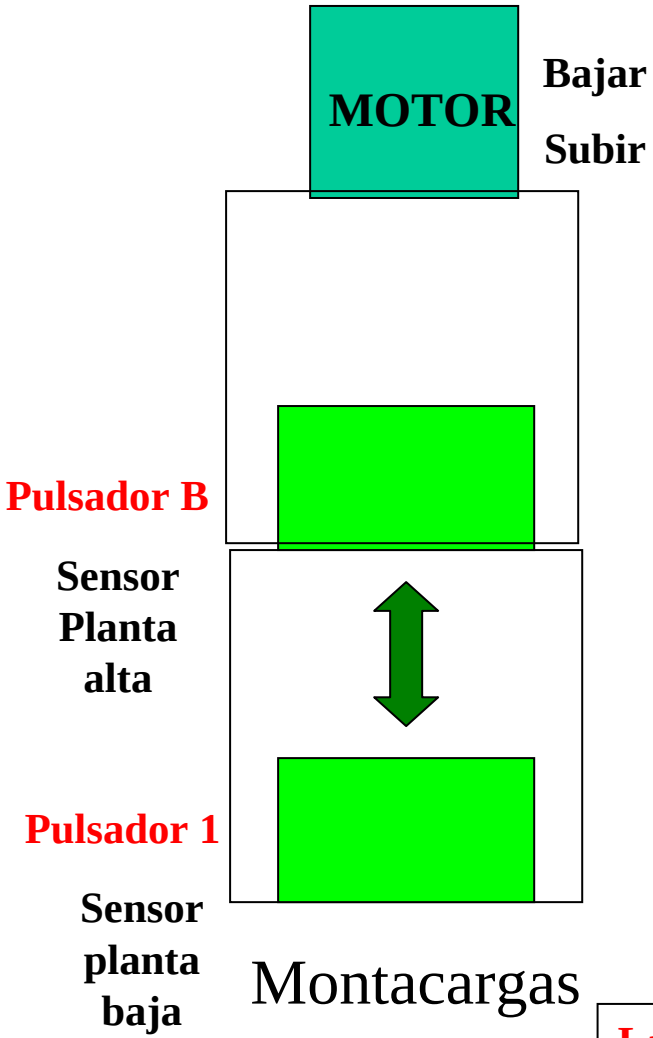



| SALIDAS (A) | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

| ENTRADAS (E) | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

| | |
|---|---|
| U E 0.4 | Sensor planta alta |
| U E 0.0 | Pulsador B |
| S A 4.0 | Bajar |

# Ejemplo : El montacargas




**SALIDAS (A)**

| | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

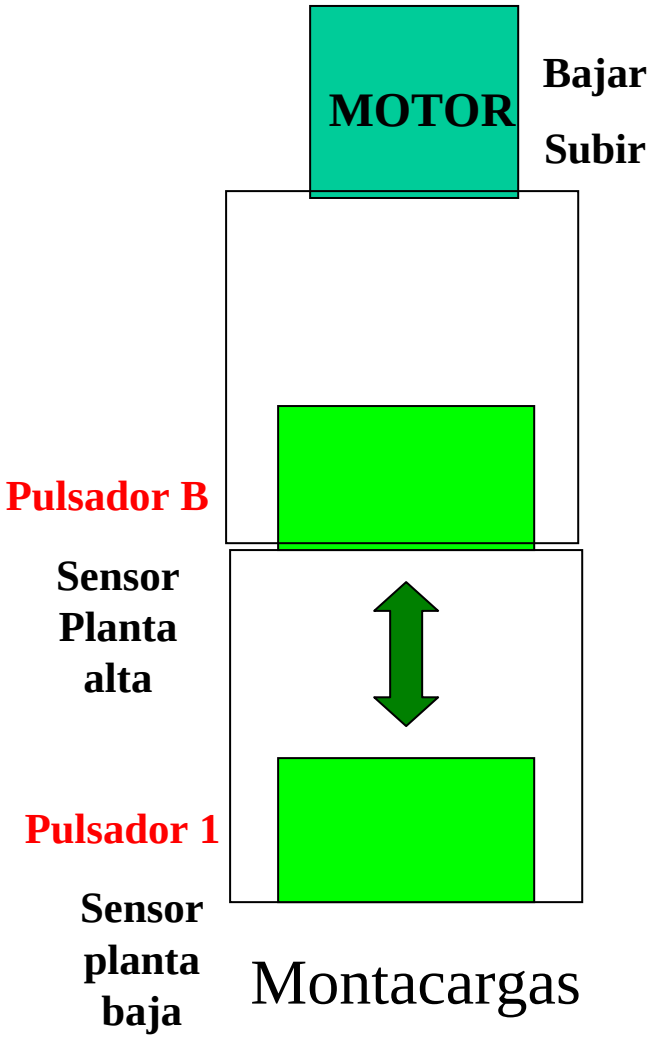
**ENTRADAS (E)**

| | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

| | |
|---|---|
| U E 0.4 | Sensor planta alta |
| U E 0.0 | Pulsador B |
| S A 4.0 | Bajar |

**La entrada E 0.0 corresponde al pulsador M**

# Ejemplo : El montacargas



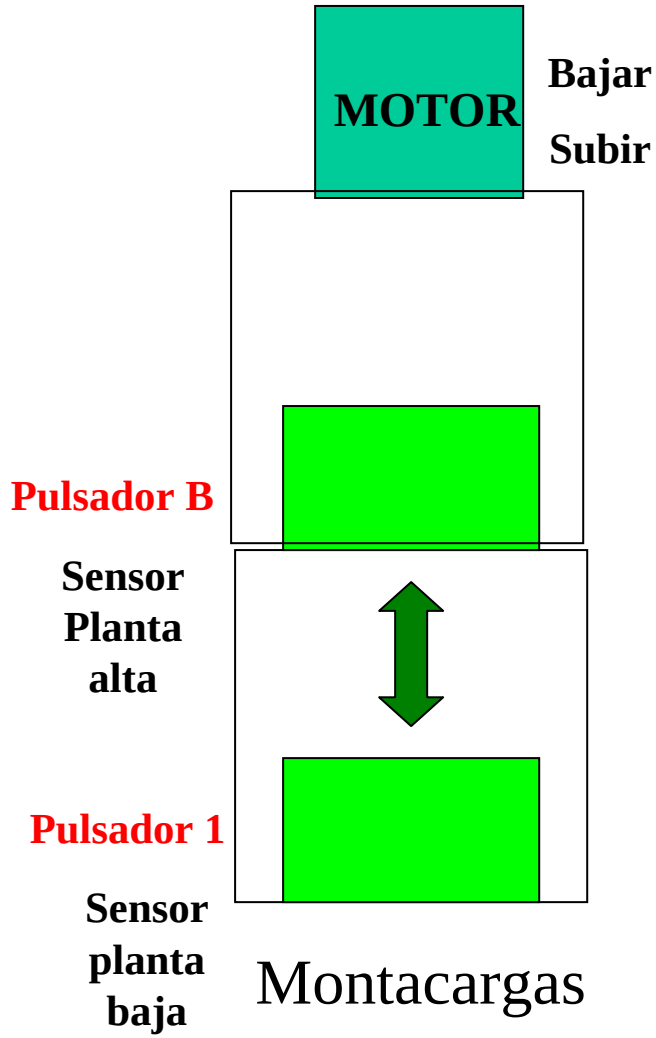

| SALIDAS (A) | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

| ENTRADAS (E) | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

**Especificación 3:**

Si el montacargas llega en la planta alta el montacargas deberá pararse.

# Ejemplo : El montacargas

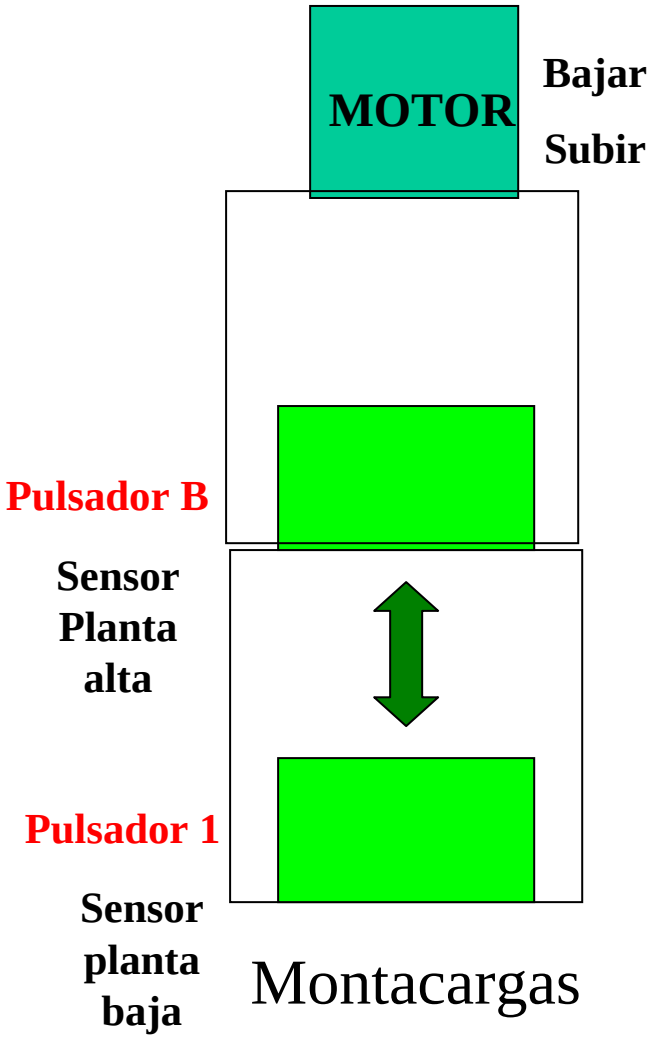



| SALIDAS (A) | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

| ENTRADAS (E) | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

| | |
|---|---|
| U E 0.4 | Sensor planta alta |
| R A 4.0 | Subir |

# Ejemplo : El montacargas



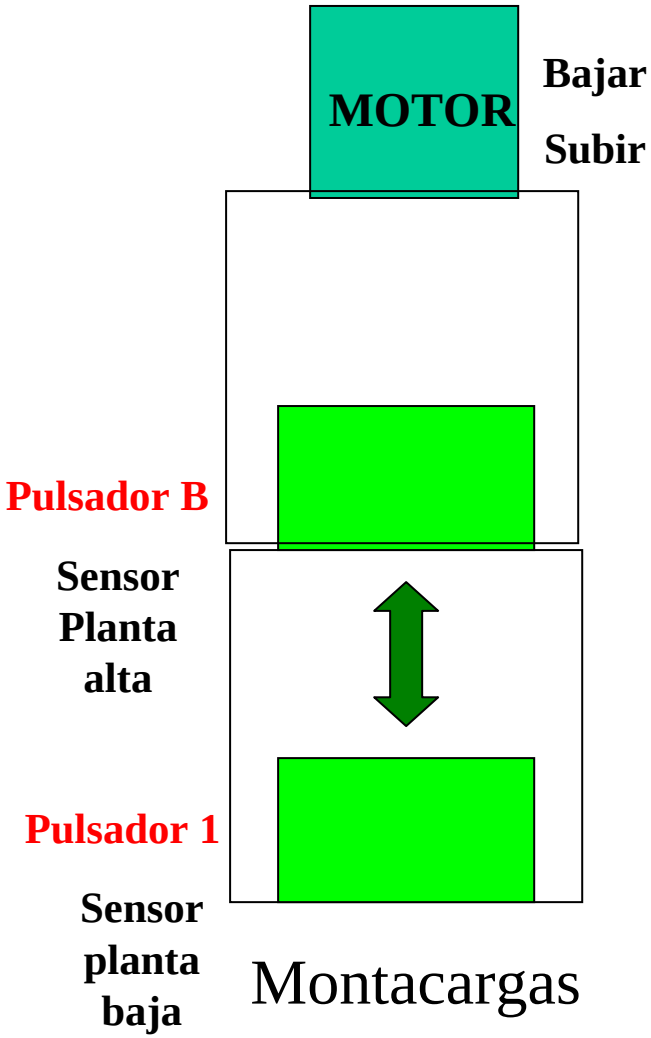

| SALIDAS (A) | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

| ENTRADAS (E) | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

**Especificación 4:**

Si el montacargas llega en la planta baja el montacargas deberá pararse.

# Ejemplo : El montacargas




| SALIDAS (A) | |
|---|---|
| BAJAR | A 4.0 |
| SUBIR | A 4.1 |

| ENTRADAS (E) | |
|---|---|
| M | E 0.0 |
| Pulsador B | E 0.1 |
| Pulsador 1 | E 0.2 |
| Sensor planta baja | E 0.3 |
| Sensor planta alta | E 0.4 |

**Especificación 5:**

Al pulsar M el montacargas se pone en marcha a la planta baja.

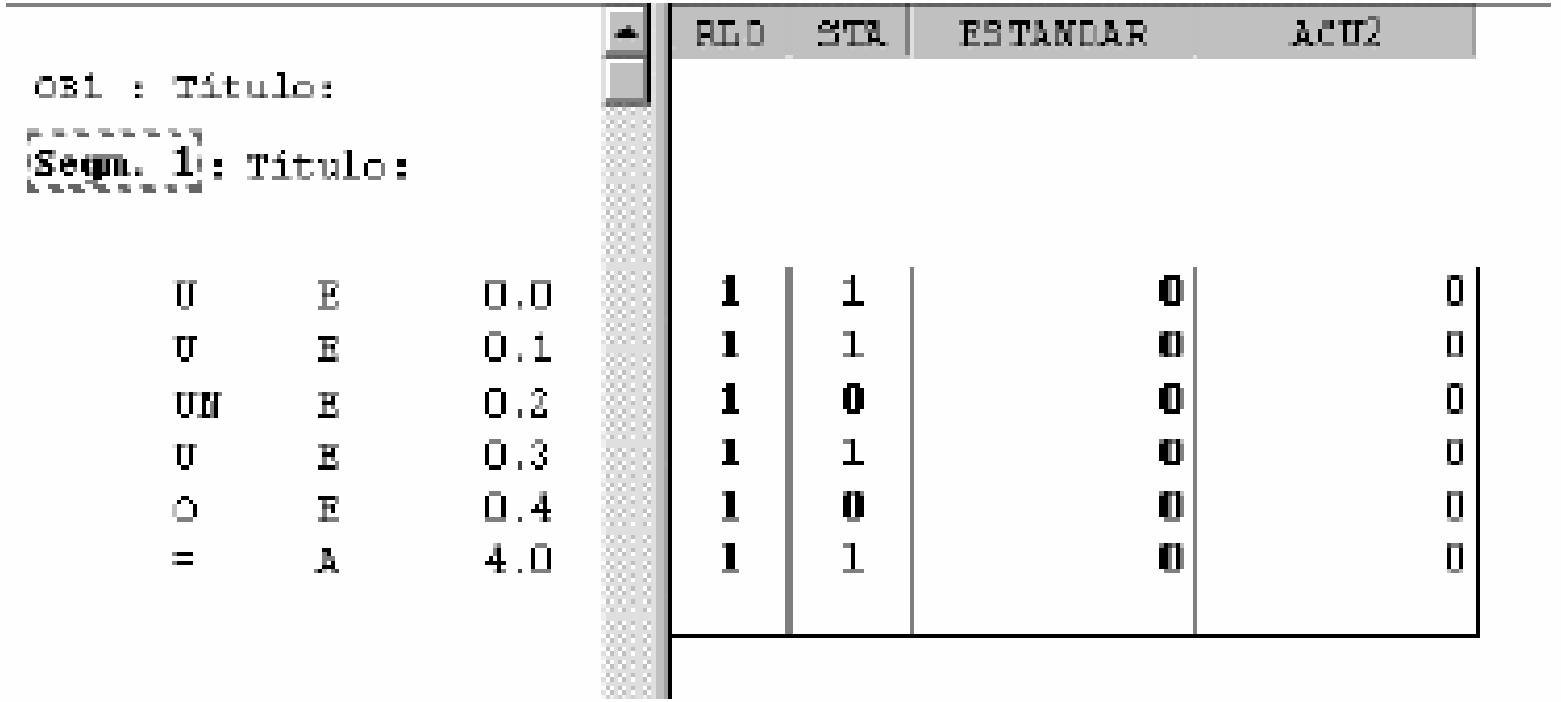
# Instrucción de terminación de cadenas lógicas

- RLO
  - Las instrucciones vistas hasta ahora son consultas y asignaciones. Esto significa: el procesador examina el estado de las señales de entrada, salida y marcas y le asigna a un estado de señal a las salidas y a las marcas.
  - Dos o más primeras consultas generan una operación lógica. El resultado de estas consultas es el resultado de la operación lógica (RLO). El resultado de la operación lógica proveniente de una operación lógica AND o una OR puede ser asignado a una salida o a una marca.

- Primera Consulta
  - La instrucción que hace la primera consulta después de una asignación se denomina de primera consulta. Esto significa que se genera un resultado de la operación lógica completamente nuevo, independiente del resultado previo de la operación lógica. Carece de importancia si la instrucción de primera consulta es una AND o una OR.

| | RLO | estado de señal | |
|---|---|---|---|
| U E 1.0 | .... | .... | |
| UN E 1.1 | .... | .... | |
| U M 4.0 | .... | .... | |
| = A 8.0 | .... | .... | |
| U E 2.0 | | | Primera consulta |

# Instrucción de terminación de cadenas lógicas



# Combinación de operaciones lógicas

**Formato**

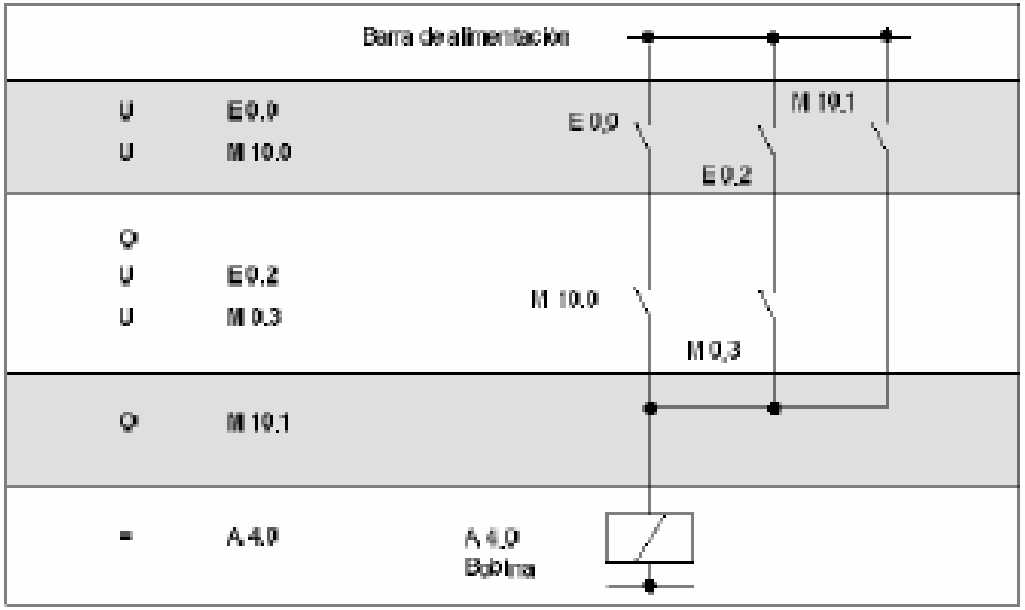
O

**Descripción de la operación**

La operación **O** realiza una O lógica de combinaciones Y siguiendo la regla Y antes de O.

**Palabra de estado**

| | RB | A1 | A0 | OV | OS | OR | STA | RLO | /ER |
|---|---|---|---|---|---|---|---|---|---|
| se escribe: | - | - | - | - | - | x | 1 | - | x |

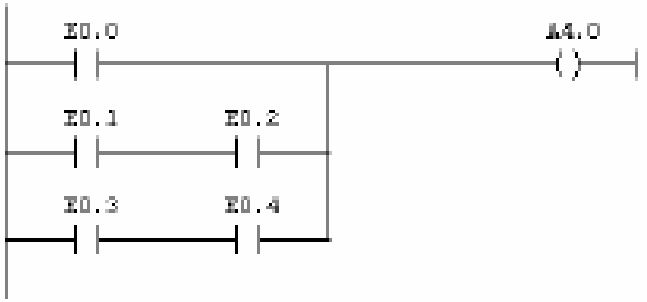


# Combinación de operaciones lógicas

- Utilización de parentesis

SOLUCIÓN EN AWL

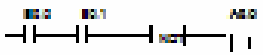
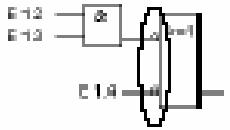
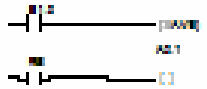
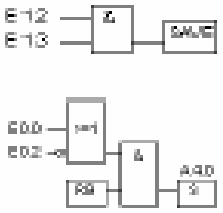
```
U  E    0.0
O(
U  E    0.1
U  E    0.2
)
O
U  E    0.3
U  E    0.4
=  A    4.0
```



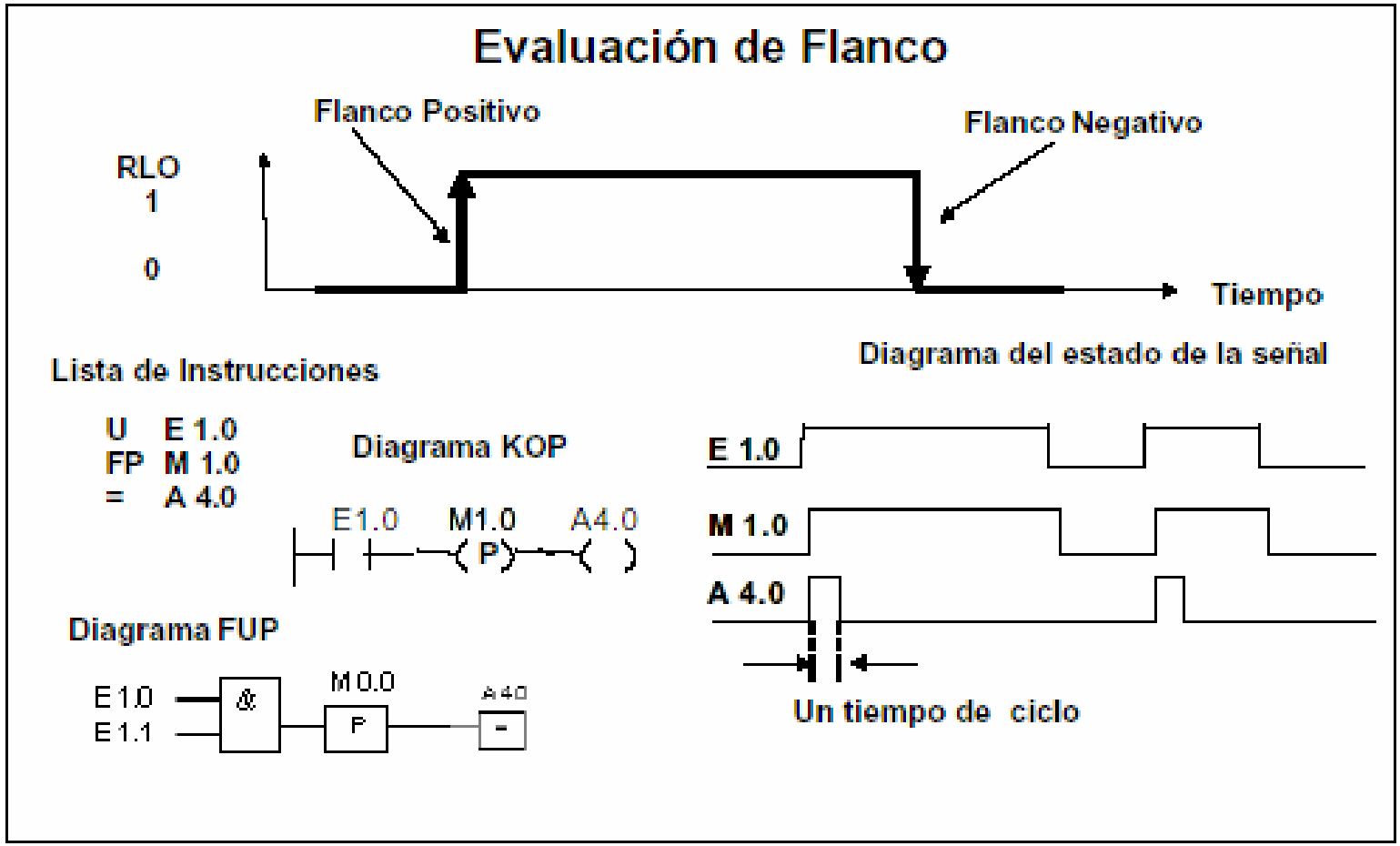
# Instrucciones que afectan al RLO

- **NOT**
  - NOT es la instrucción para invertir el RLO.
- **CLR/SET**
  - El RLO pasa a 0 con la instrucción borrar CLEAR, y el RLO pasa a1 con la instrucción SET.
- **SAVE/U RB**
  - Con la instrucción SAVE (grabar memoria), el contenido del RLO se almacena en un registro (palabra de estado). El RLO almacenado puede ser consultado de nuevo con la instrucción U RB.

**Instrucciones que Afectan al RLO**

| | KOP | FUP | AWL |
|---|---|---|---|
| NOT |  |  | U E0.0<br>U E0.1<br>NOT<br>= A8.0 |
| CLR<br>SET | No se visualiza en KOP | No se visualiza en FUP | CLR<br>SET |
| SAVE |  |  | U E1.6<br>SAVE<br>U RB<br>= A8.1 |

# Operaciones que detectan cambios en el resultado lógico



# Operaciones que detectan cambios en el resultado lógico

- En ocasiones necesitamos que una determinada acción sólo se realice una vez mientras se cumplan las condiciones para la activación de la misma.
- Una gran cantidad de sets de variables mejorarían si se les aplicase una señal de flanco positivo a sus condiciones de activación.
- La señal de flanco, tanto positivo como negativo en el Step 7 requiere de una marcha que no puede ser utilizada en otra parte del programa, por lo que es importante simbolizarla como exclusiva de ese flanco en cuestión

```
U     E     0.0          "Marcha"              --
FP    M     0.0          "Flanco_P_Marcha"     --
S     M     0.1          "Marca_Motor_Brazo"   --
U     E     0.1          "Paro"                --
R     M     0.1          "Marca_Motor_Brazo"   --
U     M     0.1          "Marca_Motor_Brazo"   --
=     A     4.0          "Motor_Brazo"         --
```

# Programación Step 7

En próximos temas se abordará lo siguiente:

- Programación en bloques
- Temporizadores y contadores
- Señales analógicas
- Otras instrucciones (Salto condicional, incondicional, .)