PROGRAMMABLE LOGIC CONTROLLER (PLC) LOGIC SCHEMATICS

AGENDA

INTRODUCTION:

- Welcome
- Introduction and Course Overview
- Activities Overview
- Ice Breaker
- PLC Logic Schematics Pre Assessment

SESSION I:

- Identify Schematic Symbols
  - N. O. and N. C. vs. Examine On and Examine Off

SESSION II:

- Converting Relay Logic to Ladder Logic
- Activities: LogixPro Introductory Lab
- Break

AGENDA

SESSION III:

- Branch Instructions and Internal Bits
- Activities: LogixPro Door Simulation Lab

LUNCH

SESSION III:

- Activities: LogixPro Door Simulation Lab (Cont'd)

AGENDA

SESSION IV:

- Boolean Logic
- Break
- Activities: LogixPro Silo Lab

REVIEW AND REFLECTION

FINAL ASSESSMENT

GENERAL AND ELECTRICAL SAFETY COURSE OVERVIEW

COURSE OVERVIEW:

- 16 hours over two days includes lecture and activities
- Provides knowledge and skills necessary to identify, read, and program PLC Logic schematics

INTRODUCTON TO PLC LOGIC SCHEMATICS COURSE OUTCOMES

- **At the conclusion of this course, participants should be able to identify, read, and program PLC Logic schematics related to:**
  - **Identify Schematic Symbols**
    - **N. O. and N. C. vs. Examine On and Examine Off**
  - **Converting Relay Logic to Ladder Logic**
  - **Branch Instructions and Internal Bits**
  - **Boolean Logic**
  - **PLC Timers**
  - **PLC Counters**

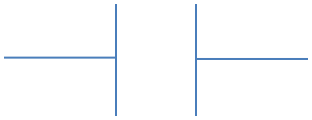
SESSION I

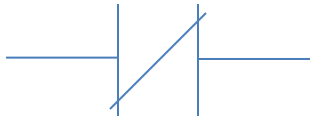
IDENTIFY SCHEMATIC SYMBOLS

SESSION I: LEARNING OBJECTIVES

- **At the conclusion of this session, participants will understand principles related to:**
  - **Identify Schematic Symbols**
    - **Normally Open**
    - **Normally Closed**
    - **Examine On**
    - **Examine Off**

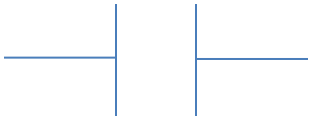
Normally Open & Normally Closed

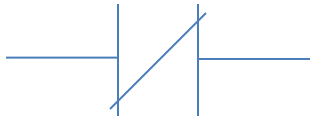




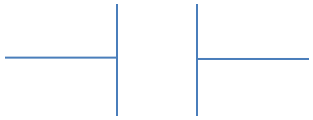
VS.

Examine On & Examine Off

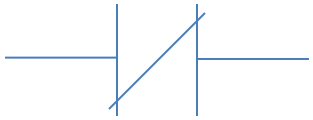




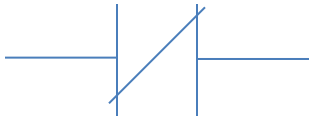
NORMALLY OPEN



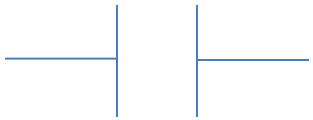
- Without any external force or pressure a normally open contact remains open
- When external force or pressure is applied a normally open contact closes



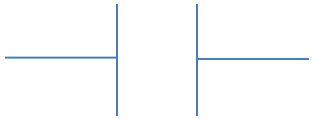
NORMALLY CLOSED



- Without any external force or pressure a normally closed contact remains closed
- When external force or pressure is applied a normally closed contact opens

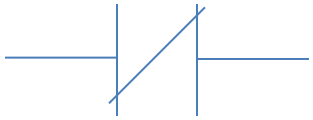


EXAMINE ON



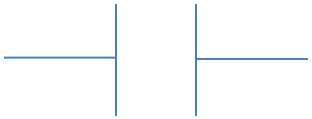
- Examine on is true when it is on
- Examine on is false when it is off

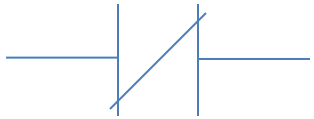
EXAMINE OFF



- Examine off is true when it is off
- Examine off is false when it is on

NORMALLY OPEN & NORMALLY CLOSED

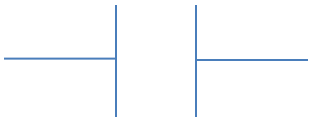


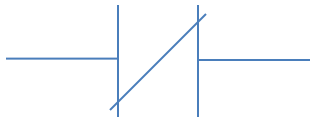


- Associated with the flow of electricity

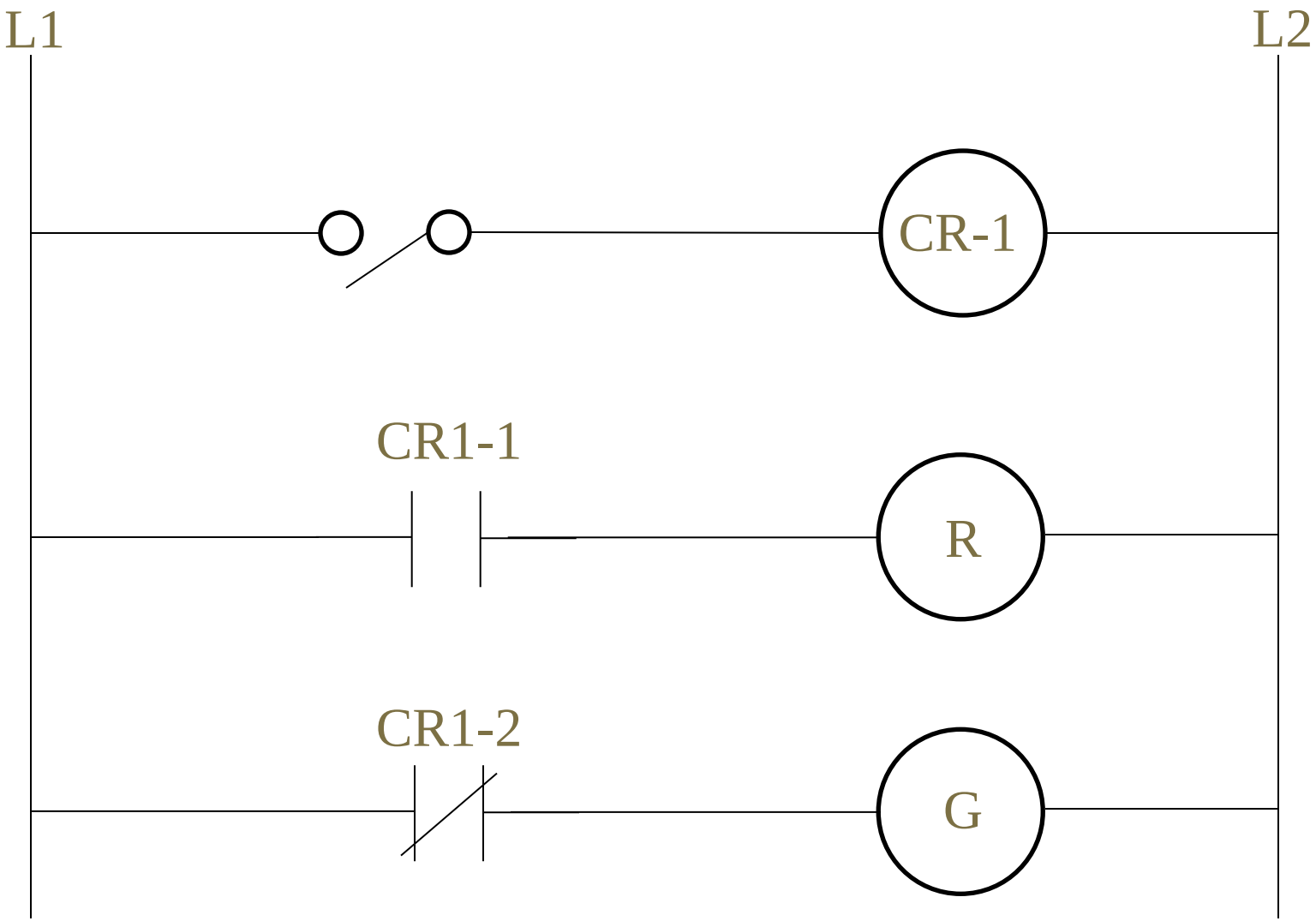
EXAMINE ON & EXAMINE OFF

- Associated with the flow of logic

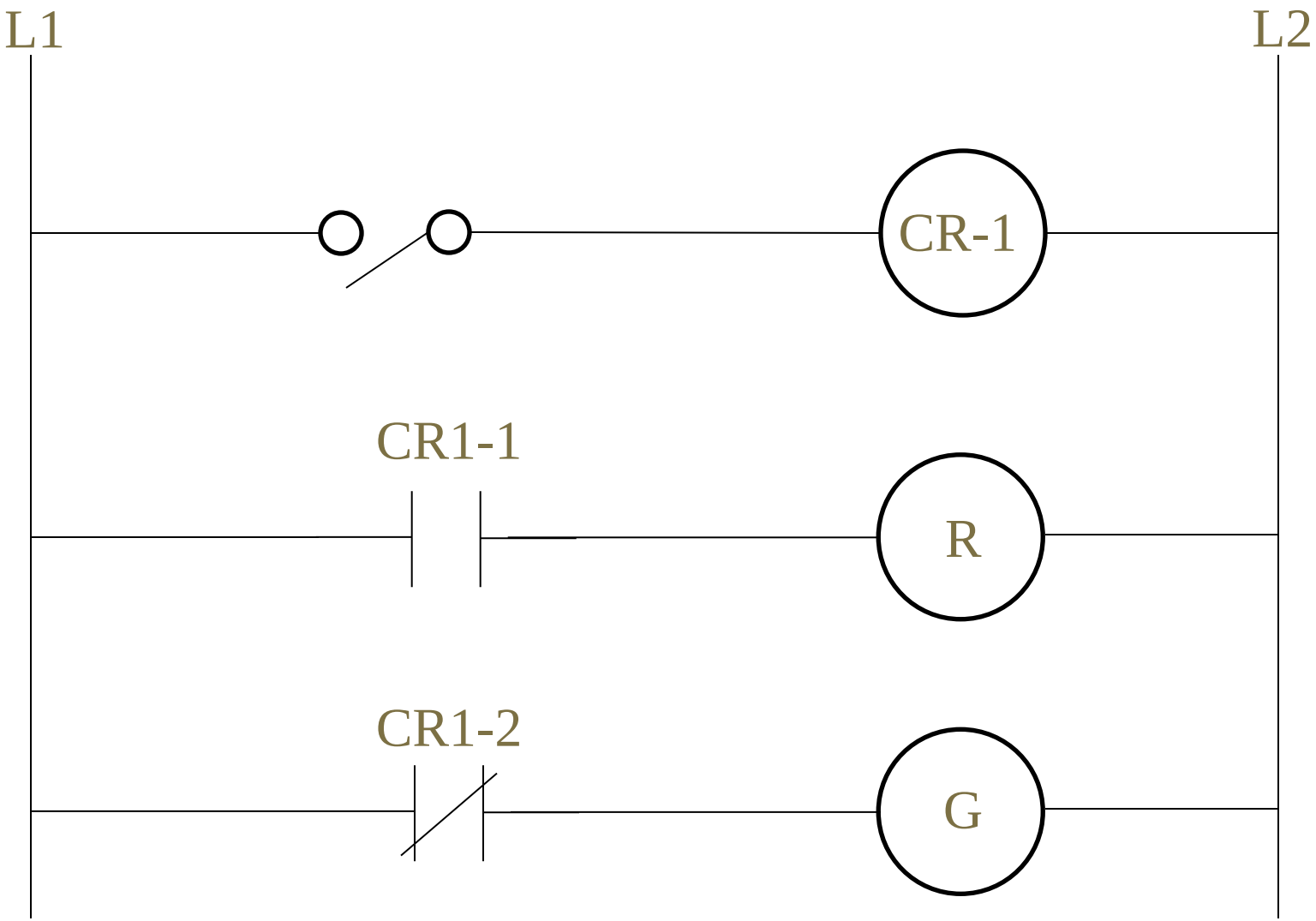




NORMALLY OPEN & NORMALLY CLOSED



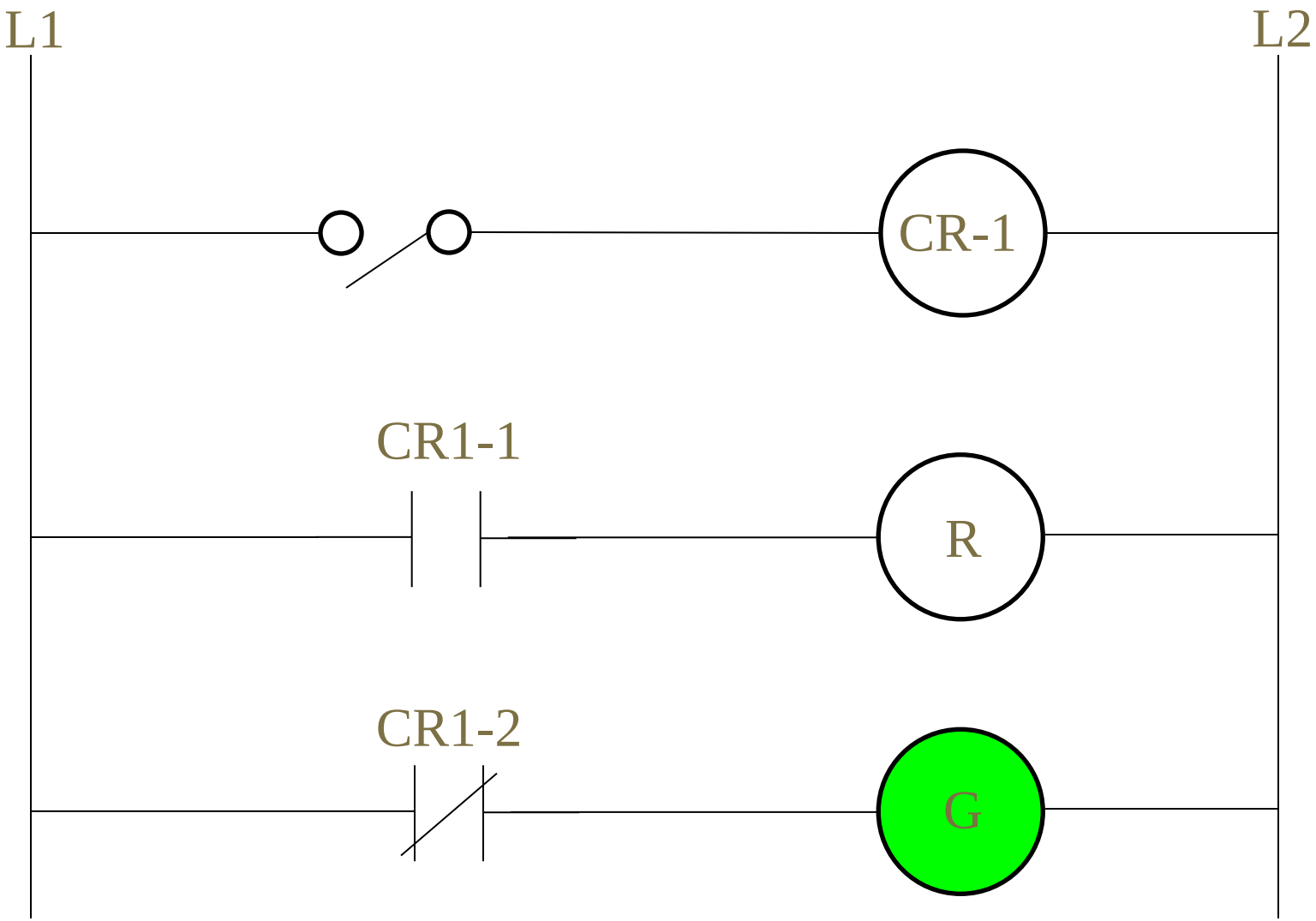
NORMALLY OPEN & NORMALLY CLOSED



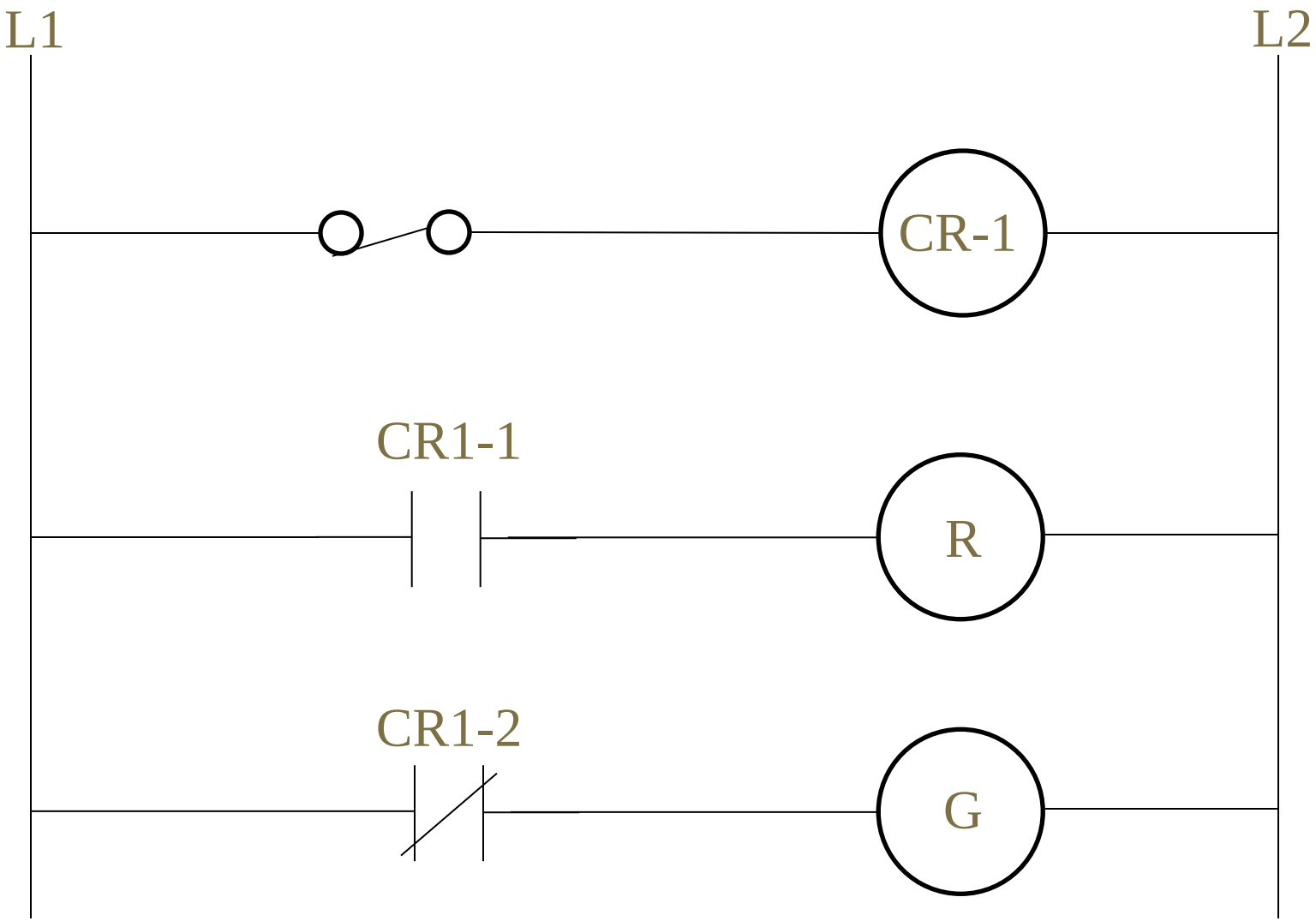
WHAT WILL HAPPEN IN THE FOLLOWING CIRCUIT WHEN THE SWITCH IS OPEN?

- When the switch is open, the N.C. contact remains closed and the green light is energized, while the N.O. contact remains open and the red light remains off

OPEN SWITCH



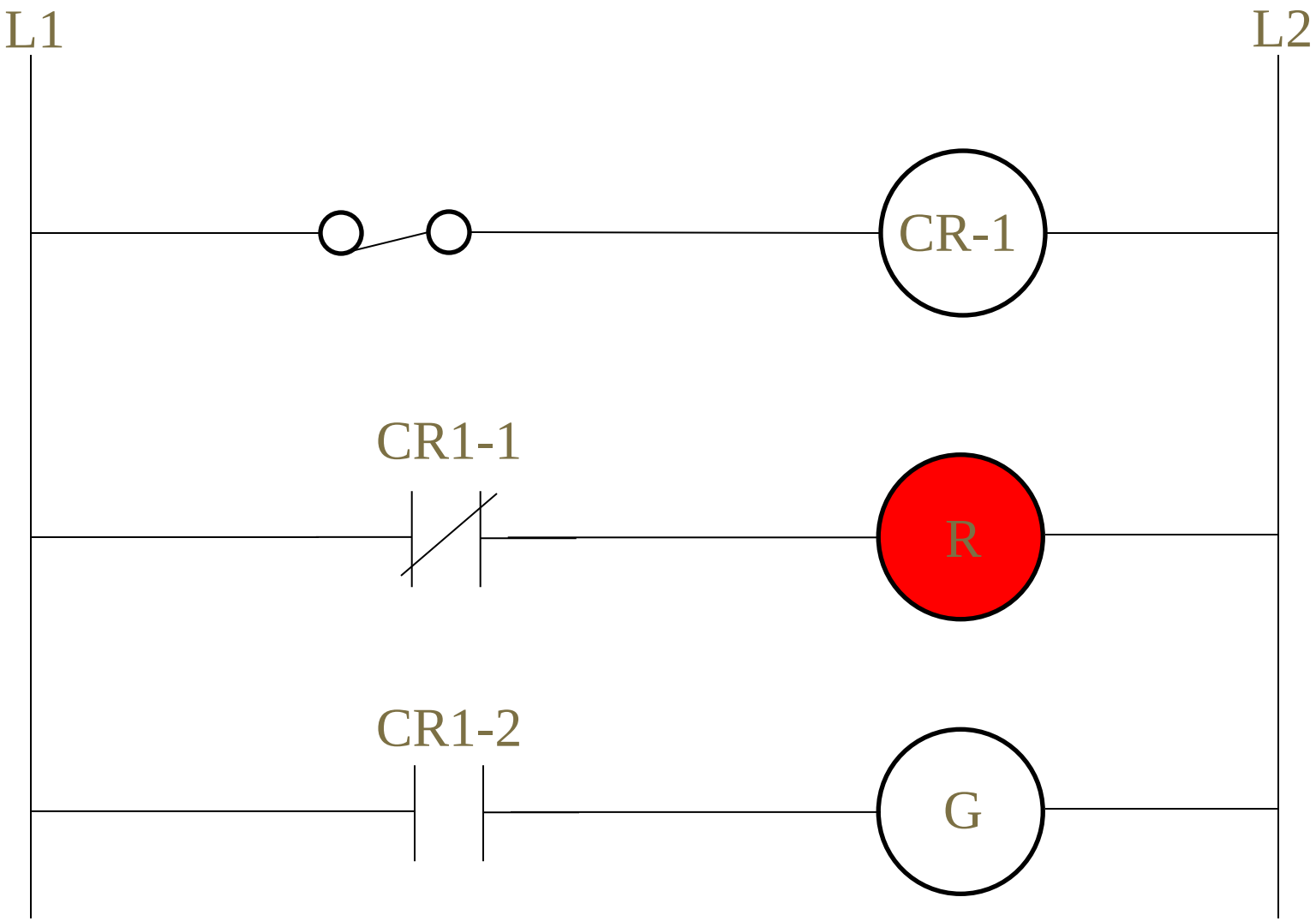
NORMALLY OPEN & NORMALLY CLOSED



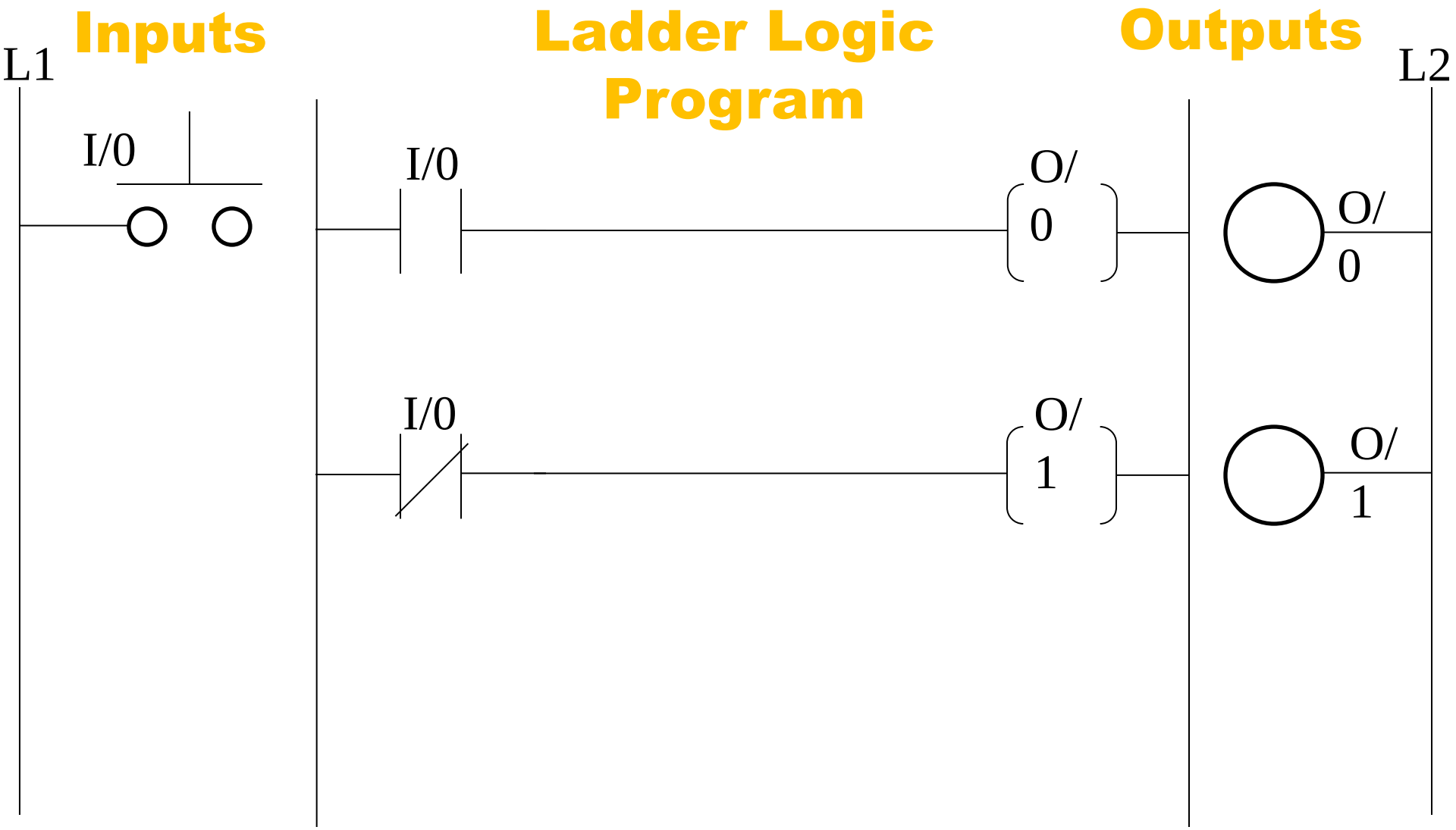
WHAT WILL HAPPEN IN THE FOLLOWING CIRCUIT WHEN THE SWITCH IS CLOSED?

- When the switch is closed, the N.C. contact opens and the green light is off, while the N.O. contact closes and the red light is energized

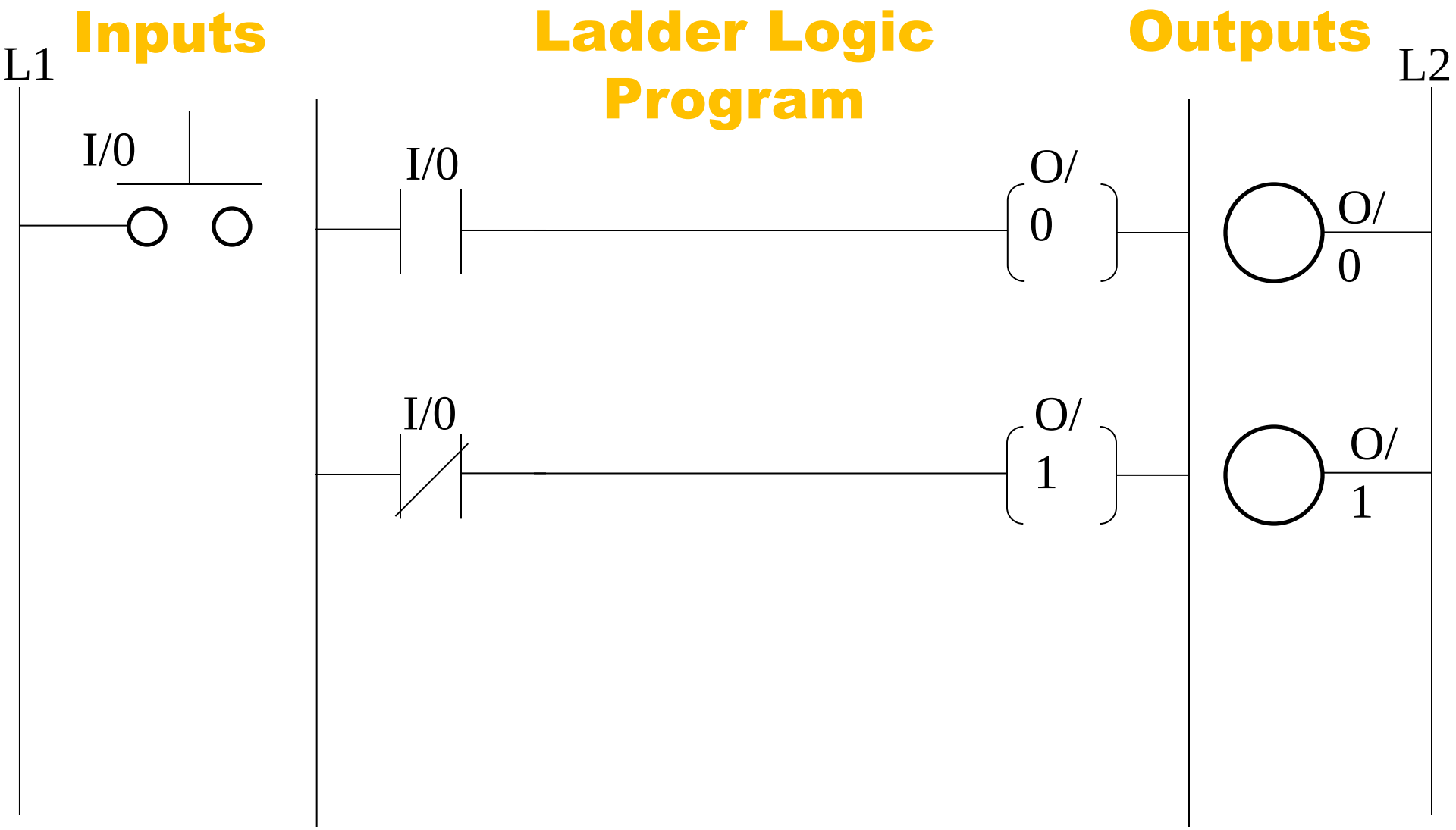
CLOSED SWITCH



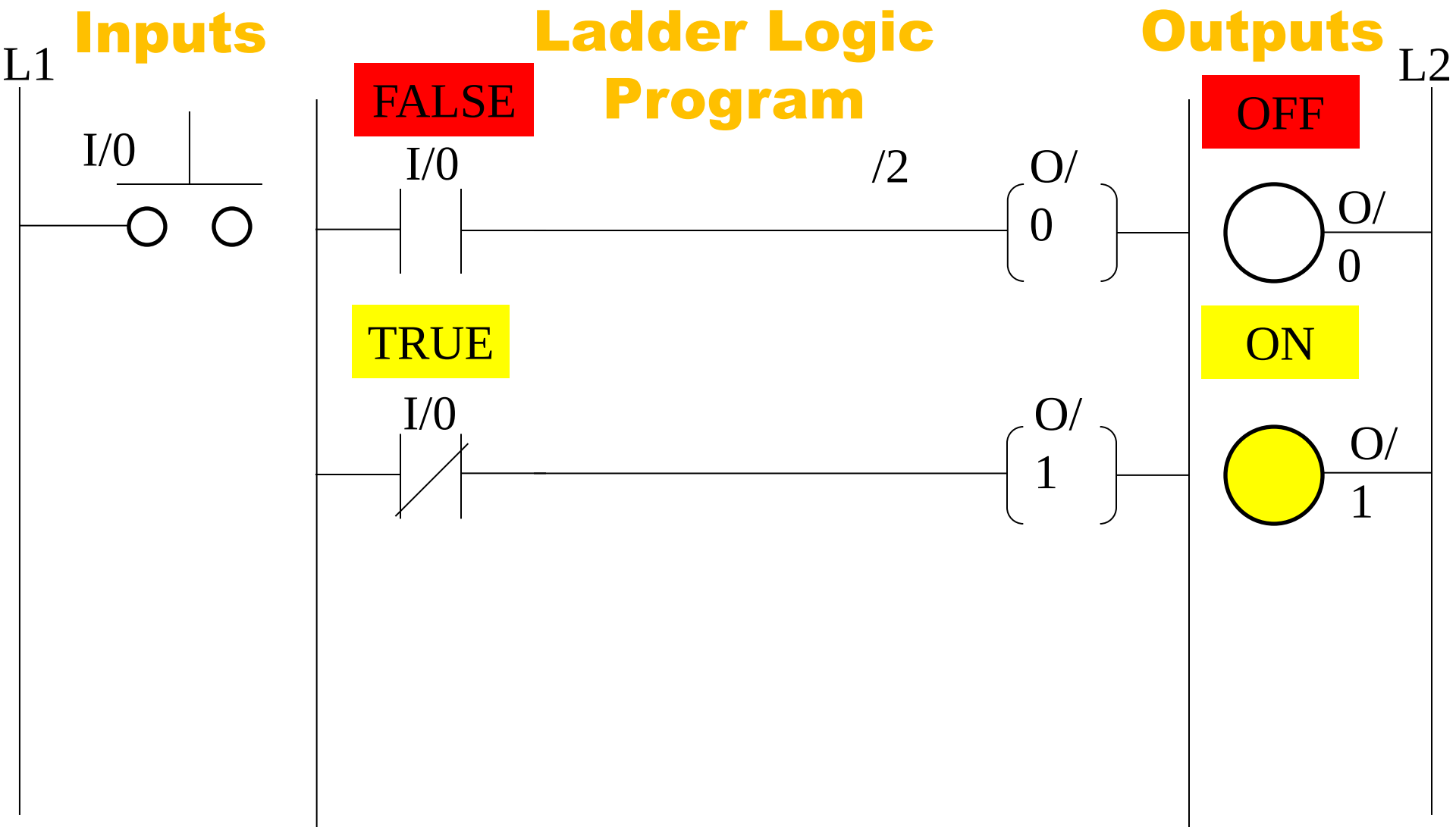
EXAMINE ON & EXAMINE OFF



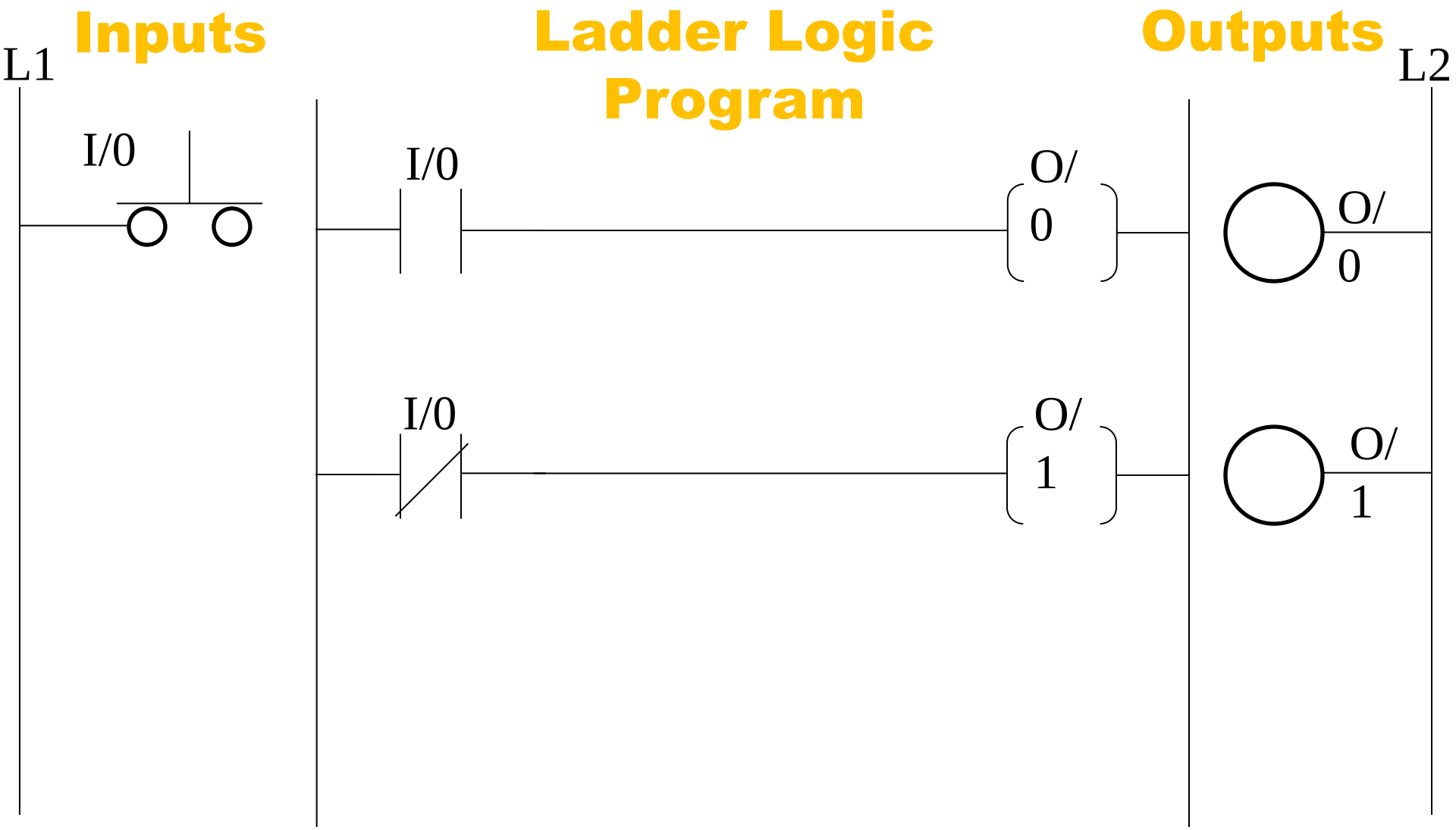
- **Which output if any will energize in this circuit?**



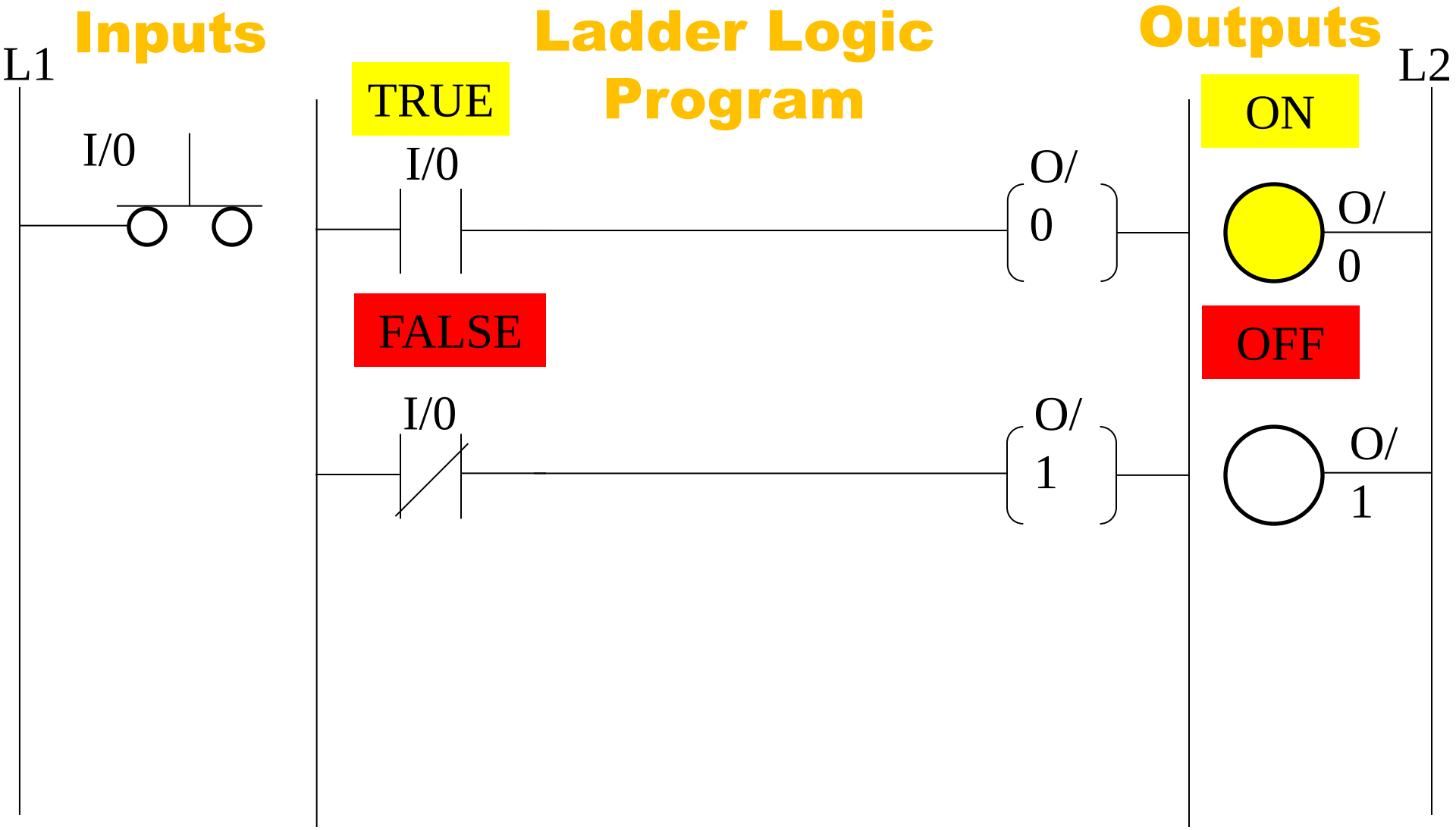
EXAMINE ON & EXAMINE OFF



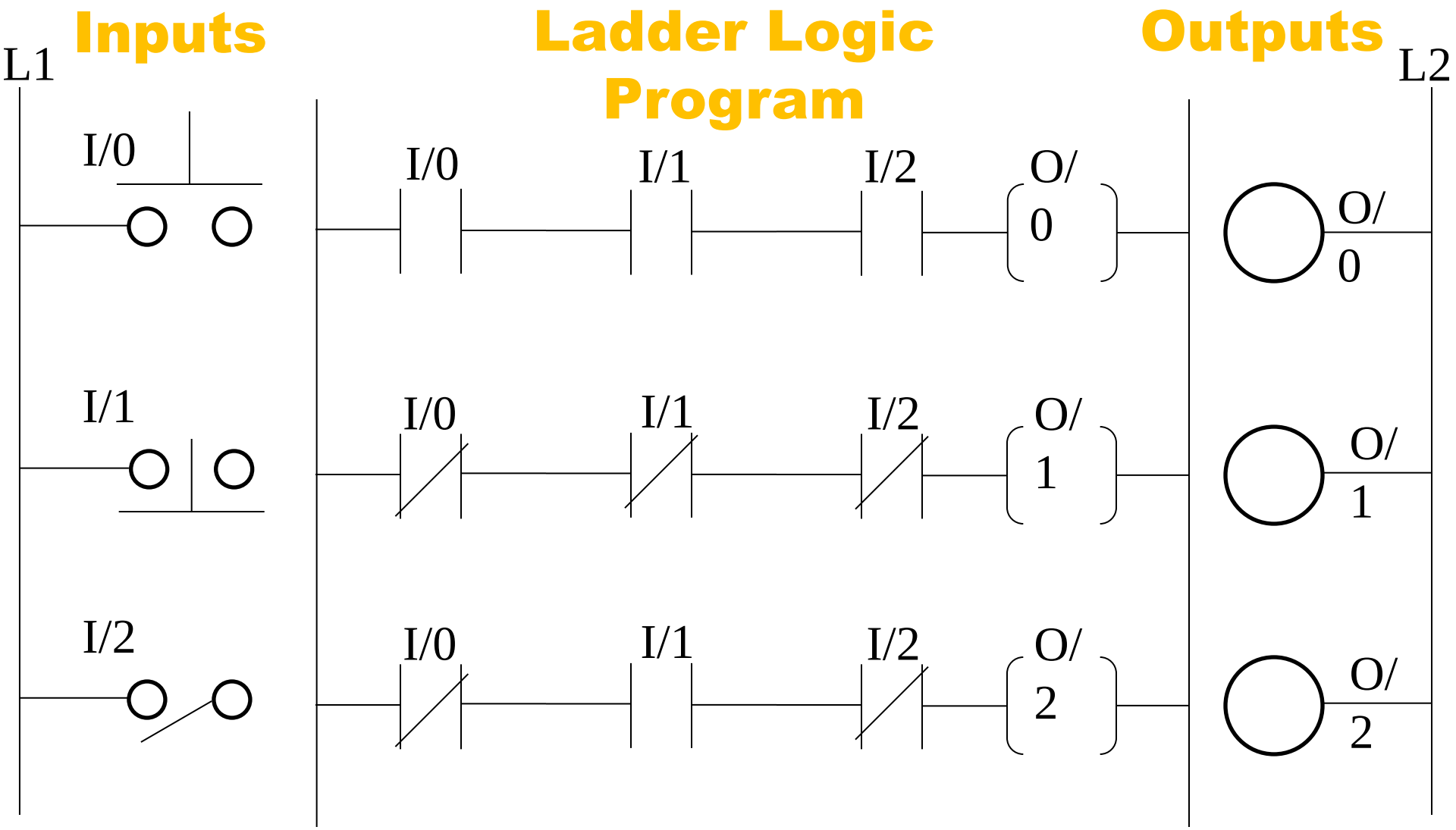
- **Which output if any will energize in this circuit?**



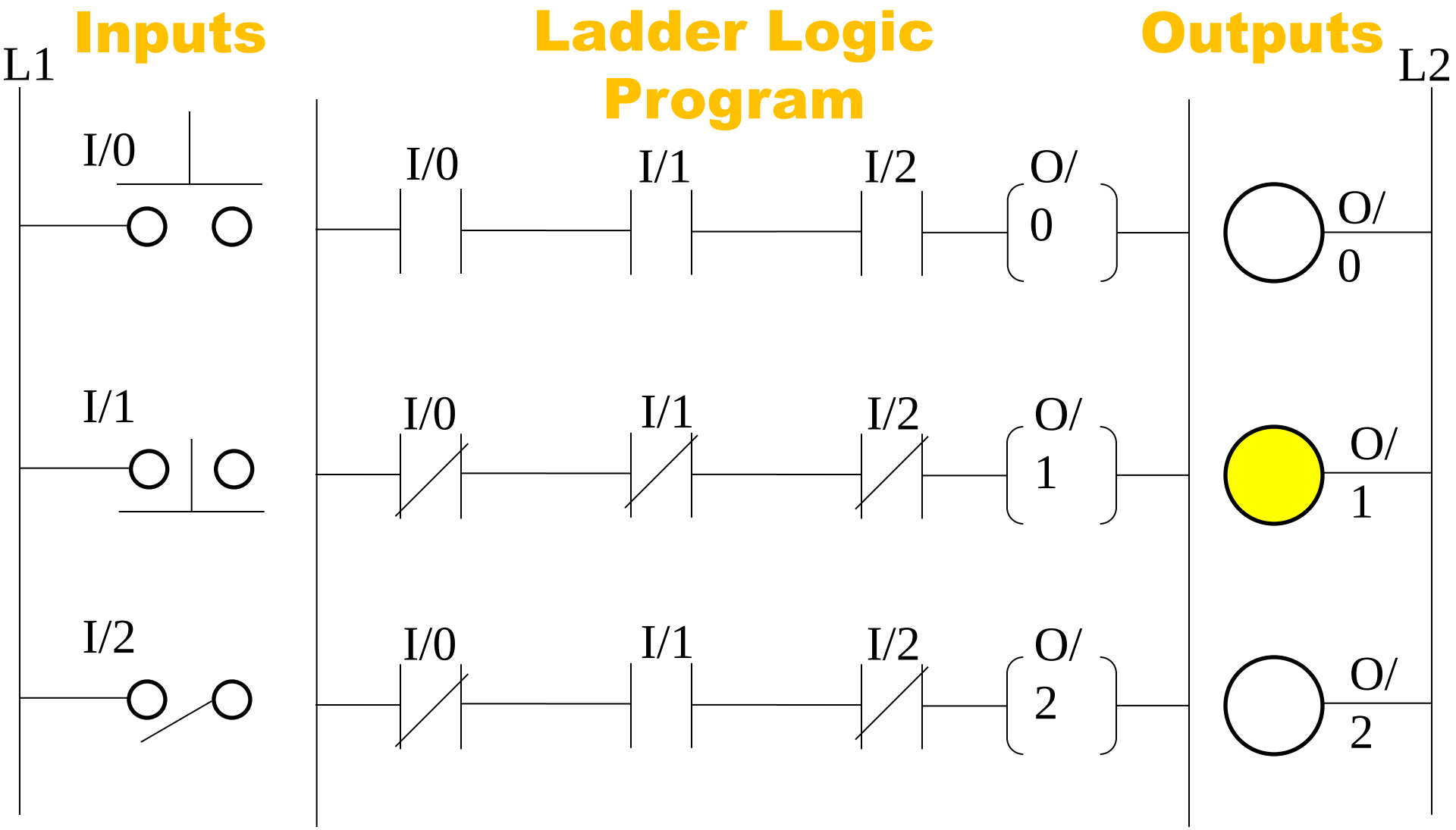
EXAMINE ON & EXAMINE OFF



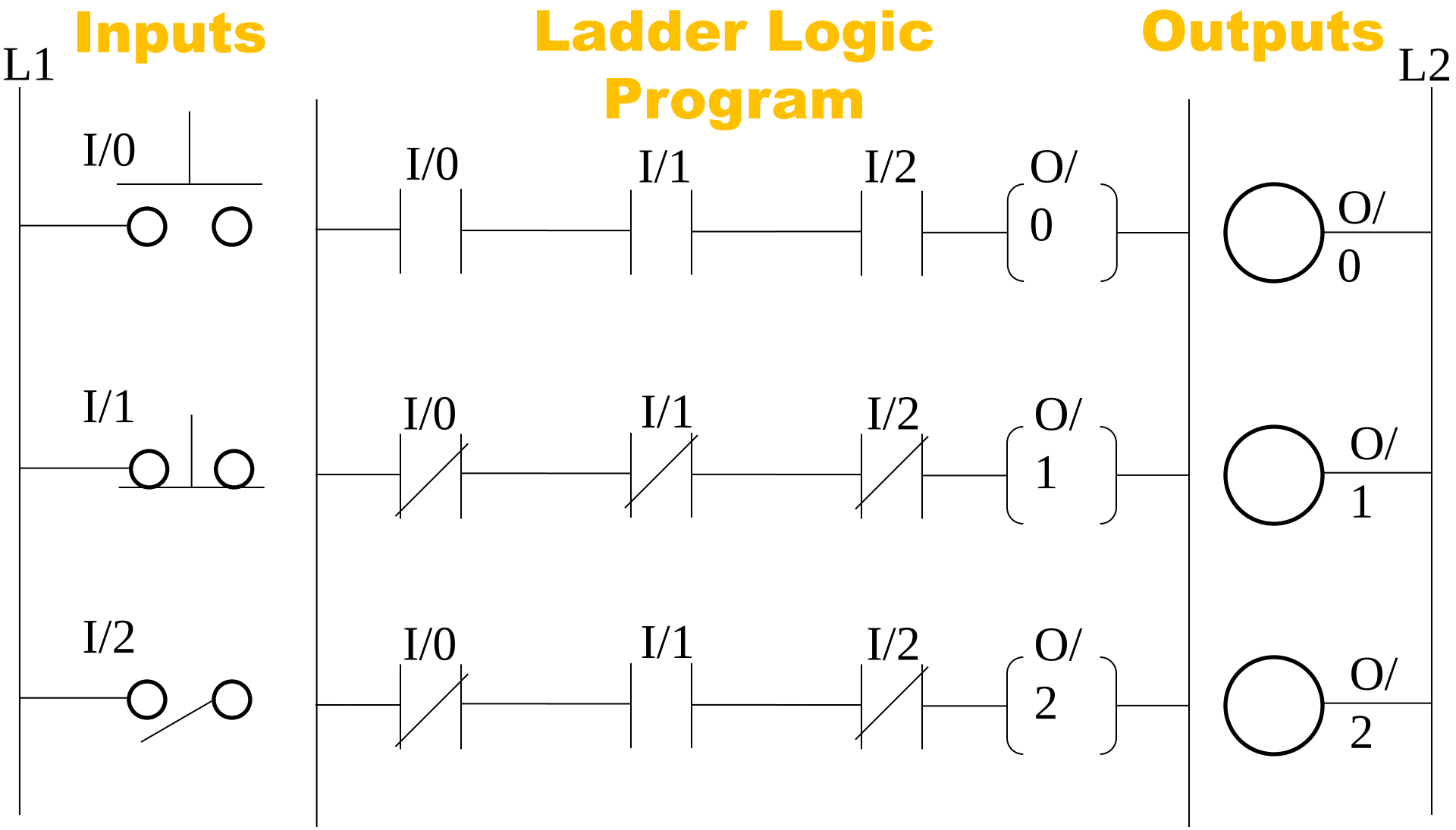
- **Look at the inputs and determine which output will energize**



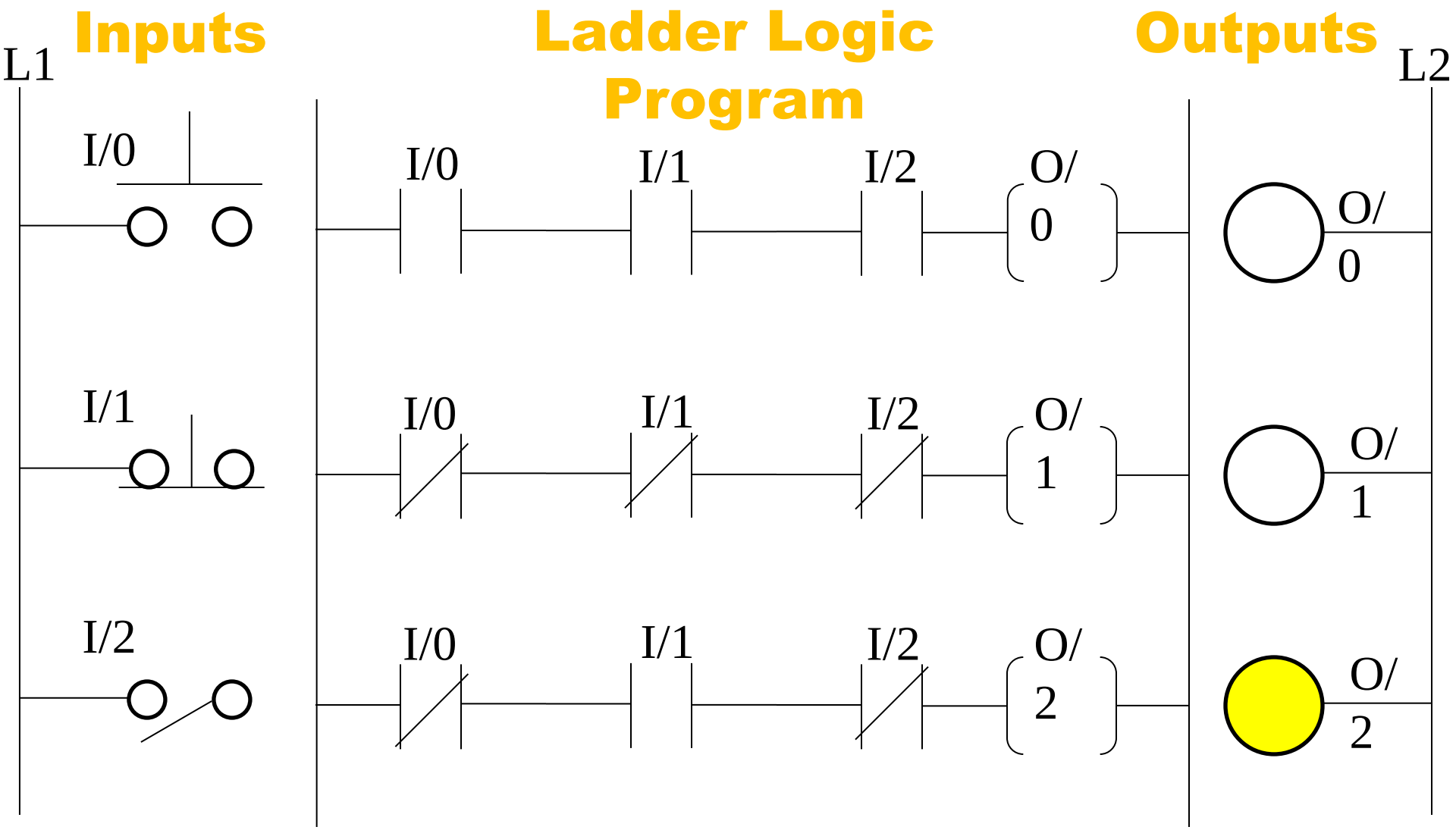
- **All inputs are open/off, so output O/1 is energized**



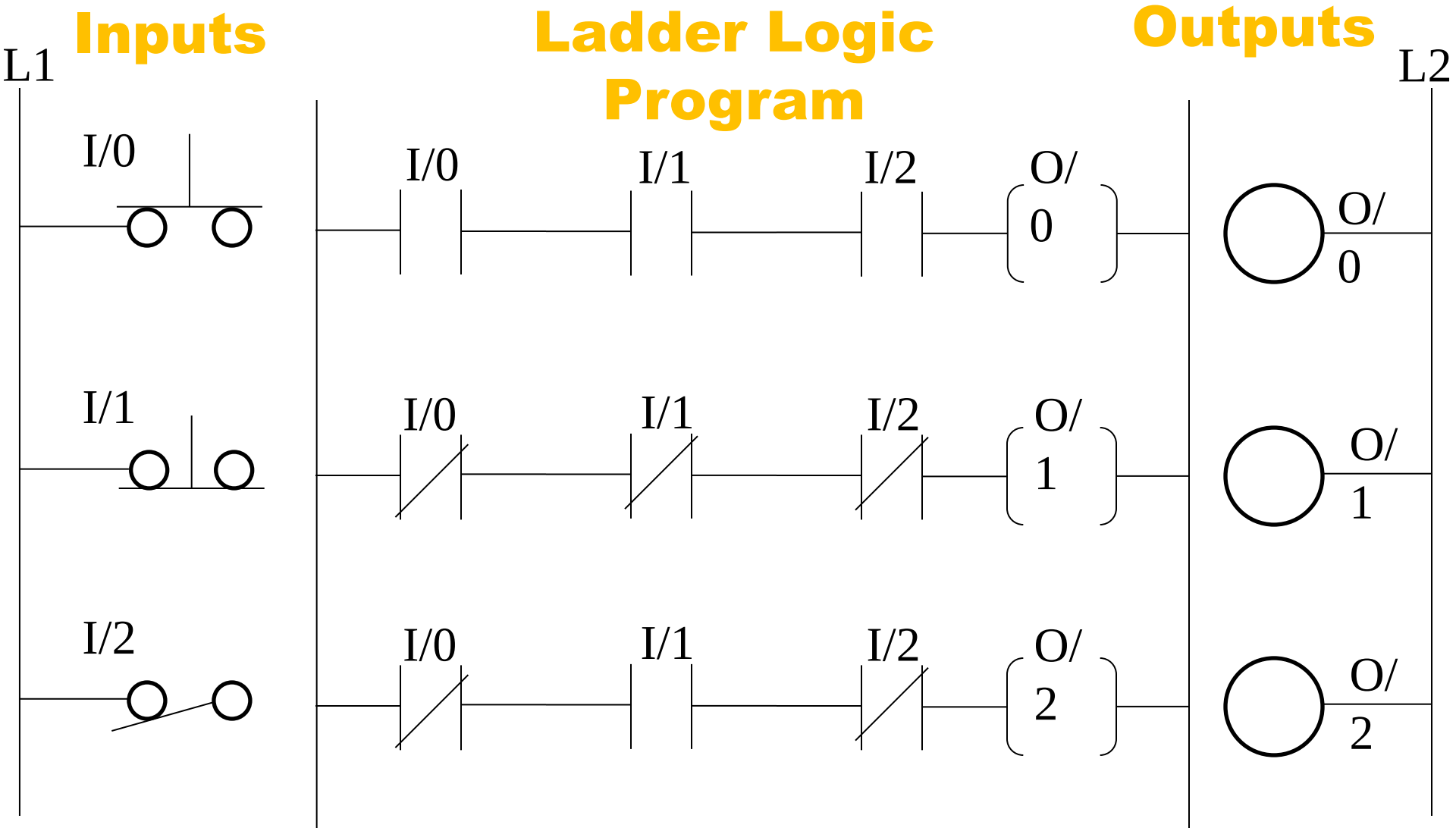
- **Look at the inputs and determine which output will energize**



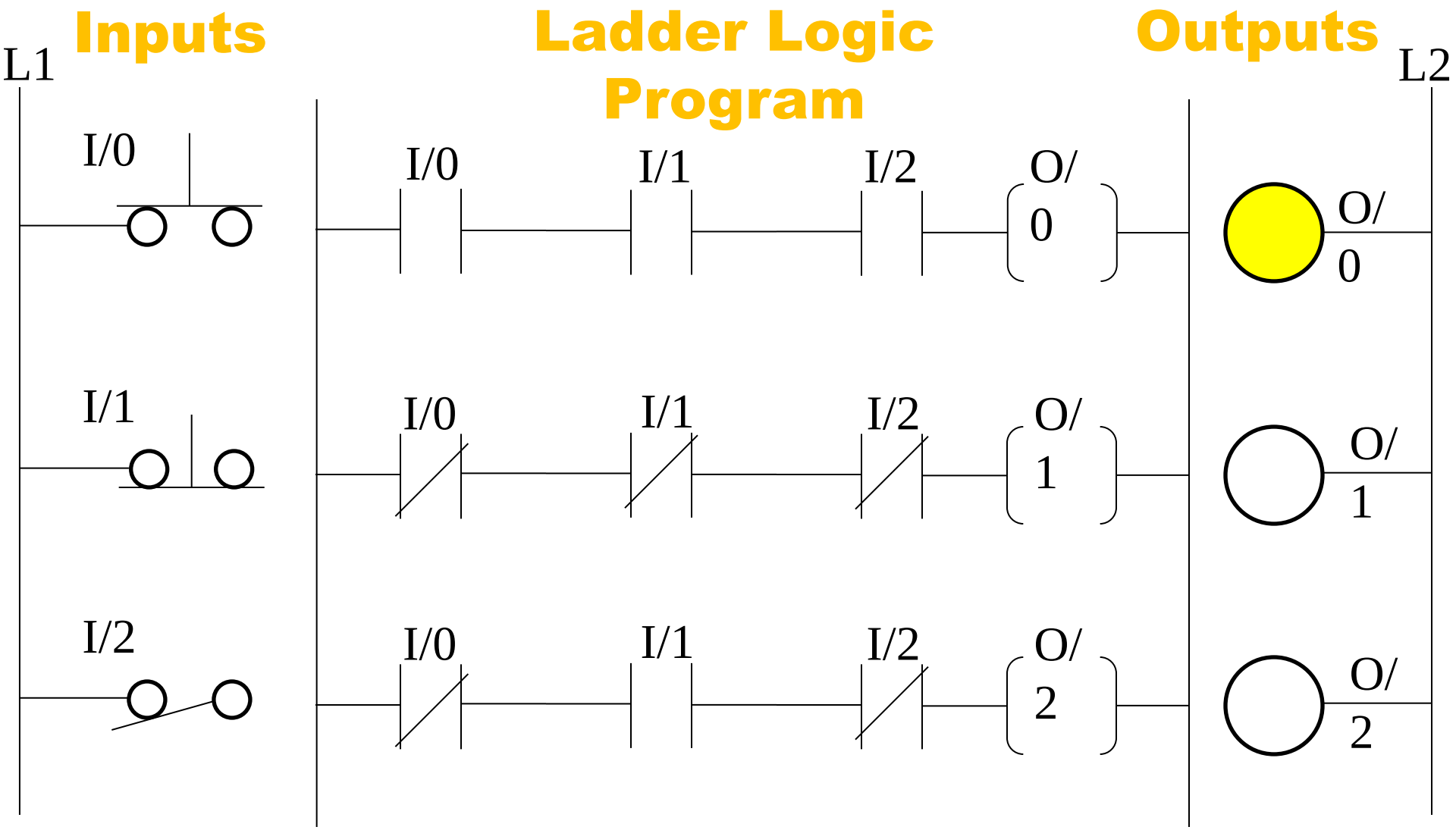
- **Output O/2 will energize**



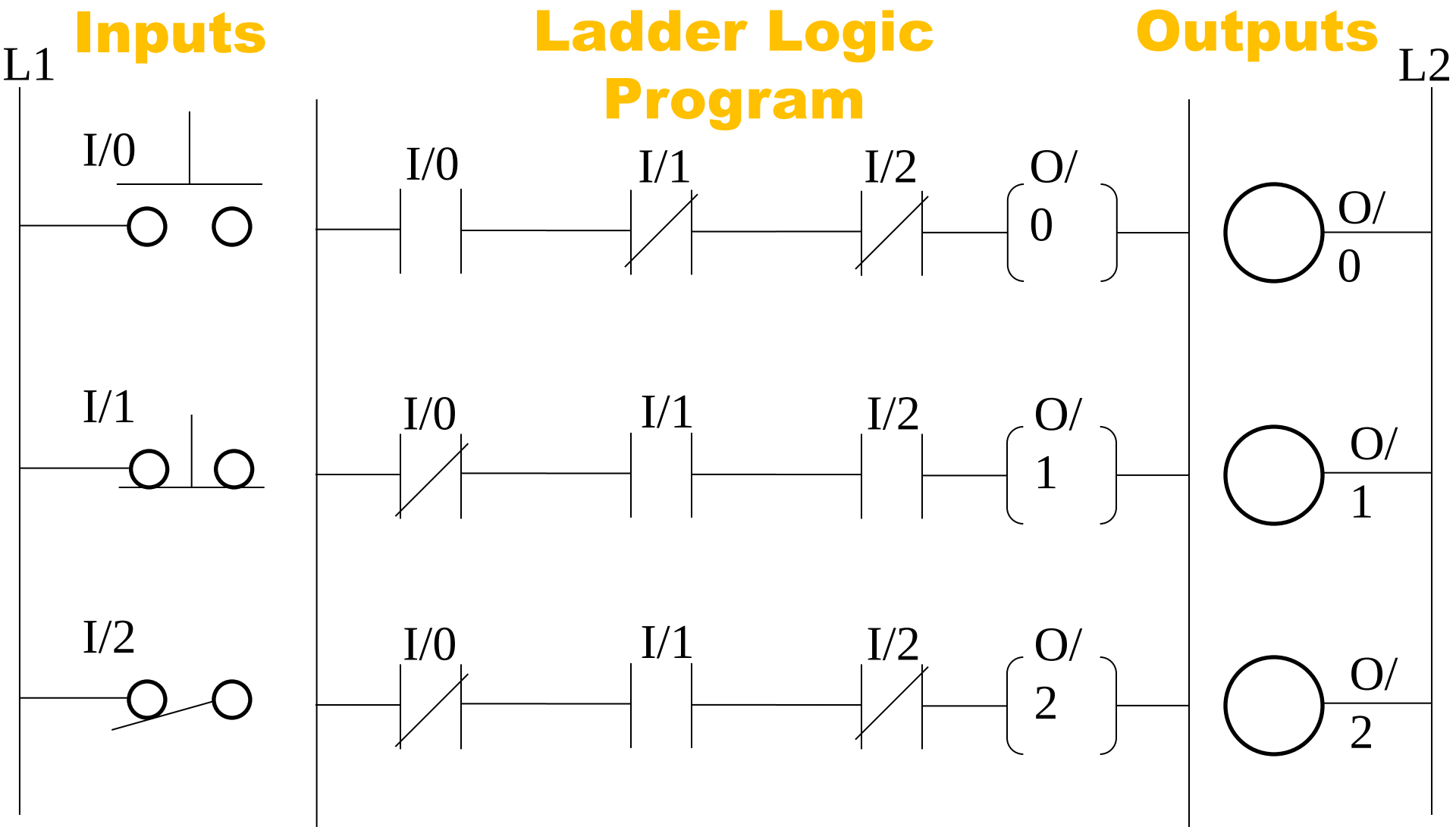
- **Look at the inputs and determine which output will energize**



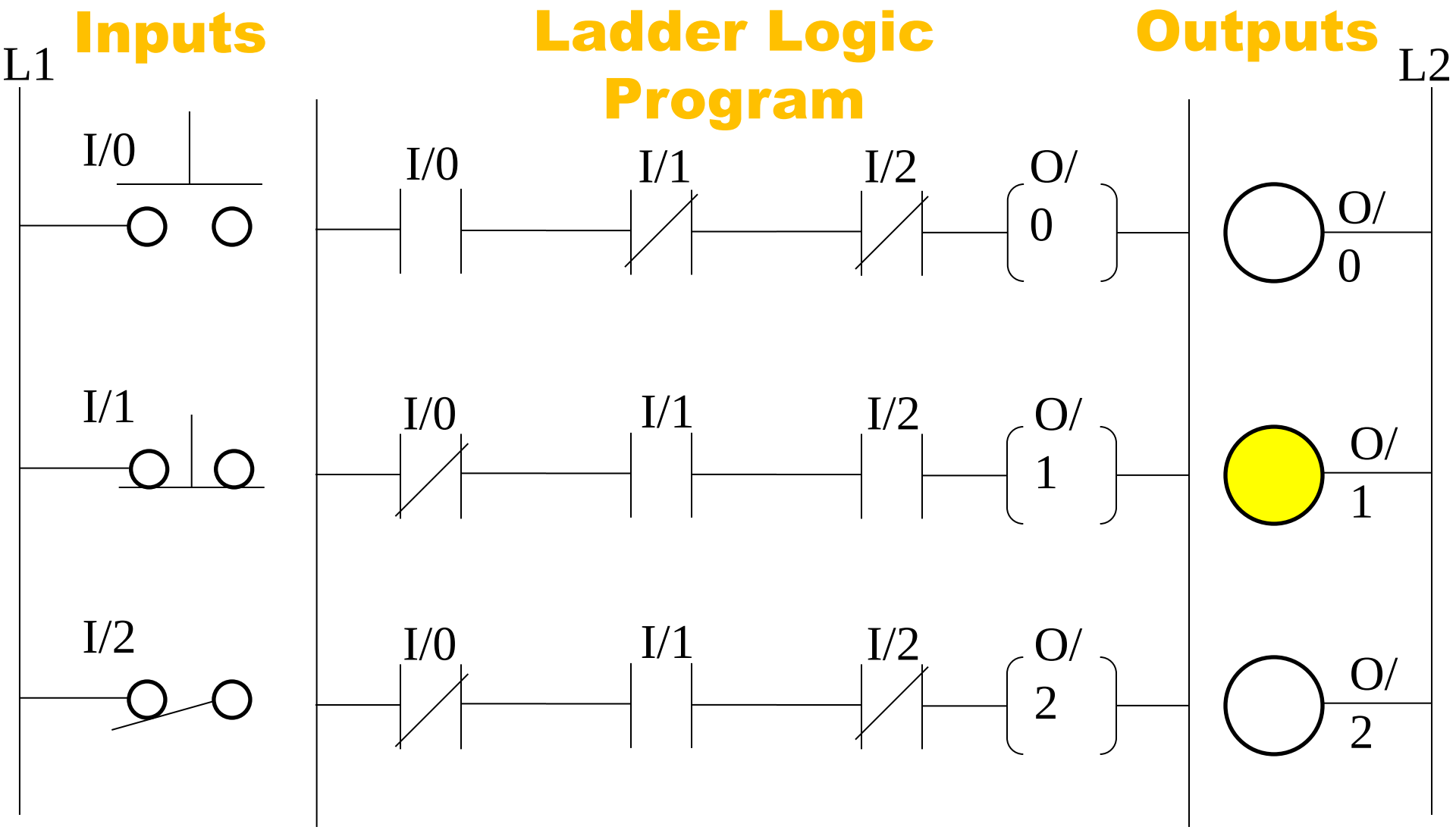
- **Output O/0 will energize**



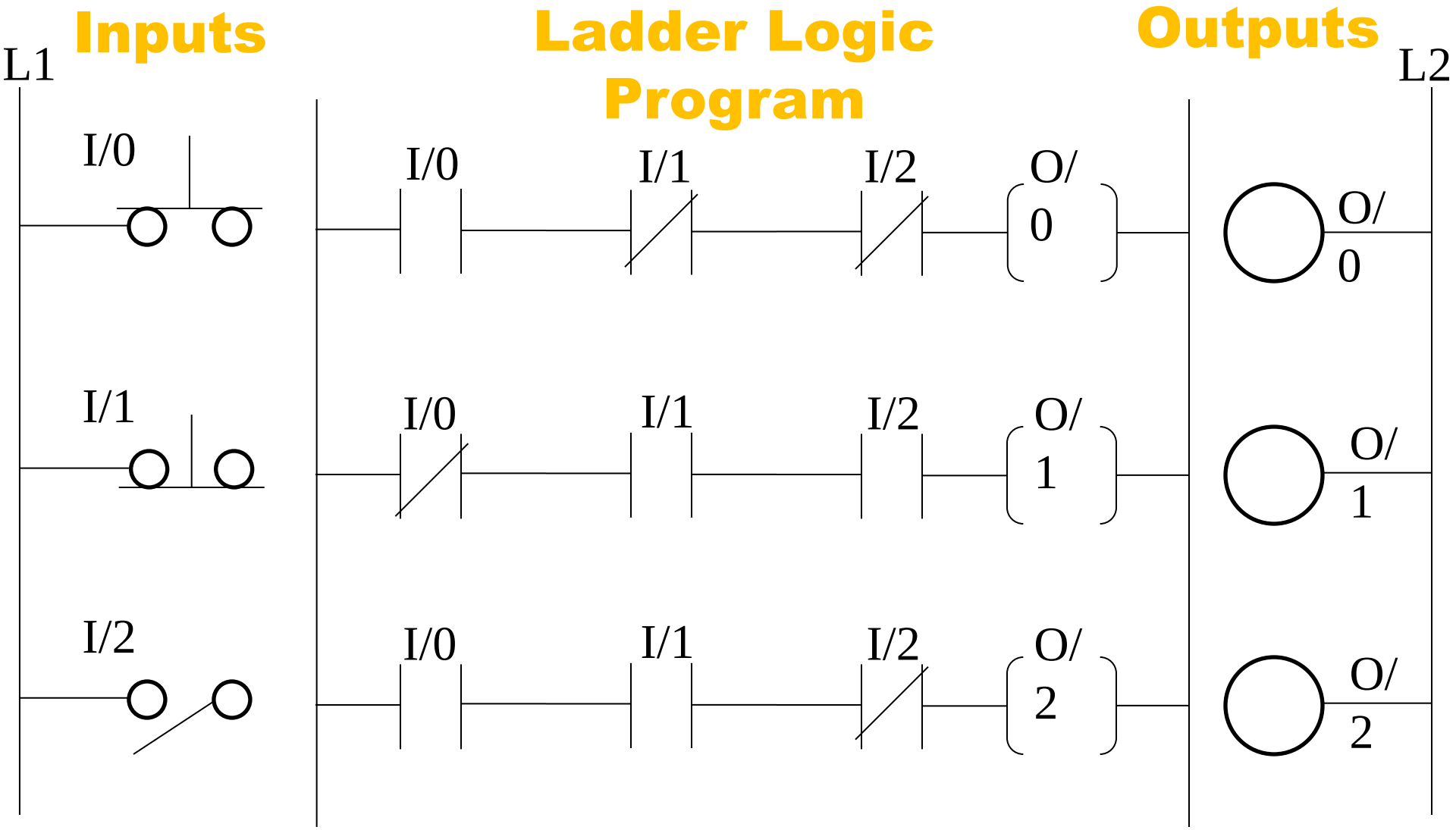
- **Look at the inputs and determine which output will energize**



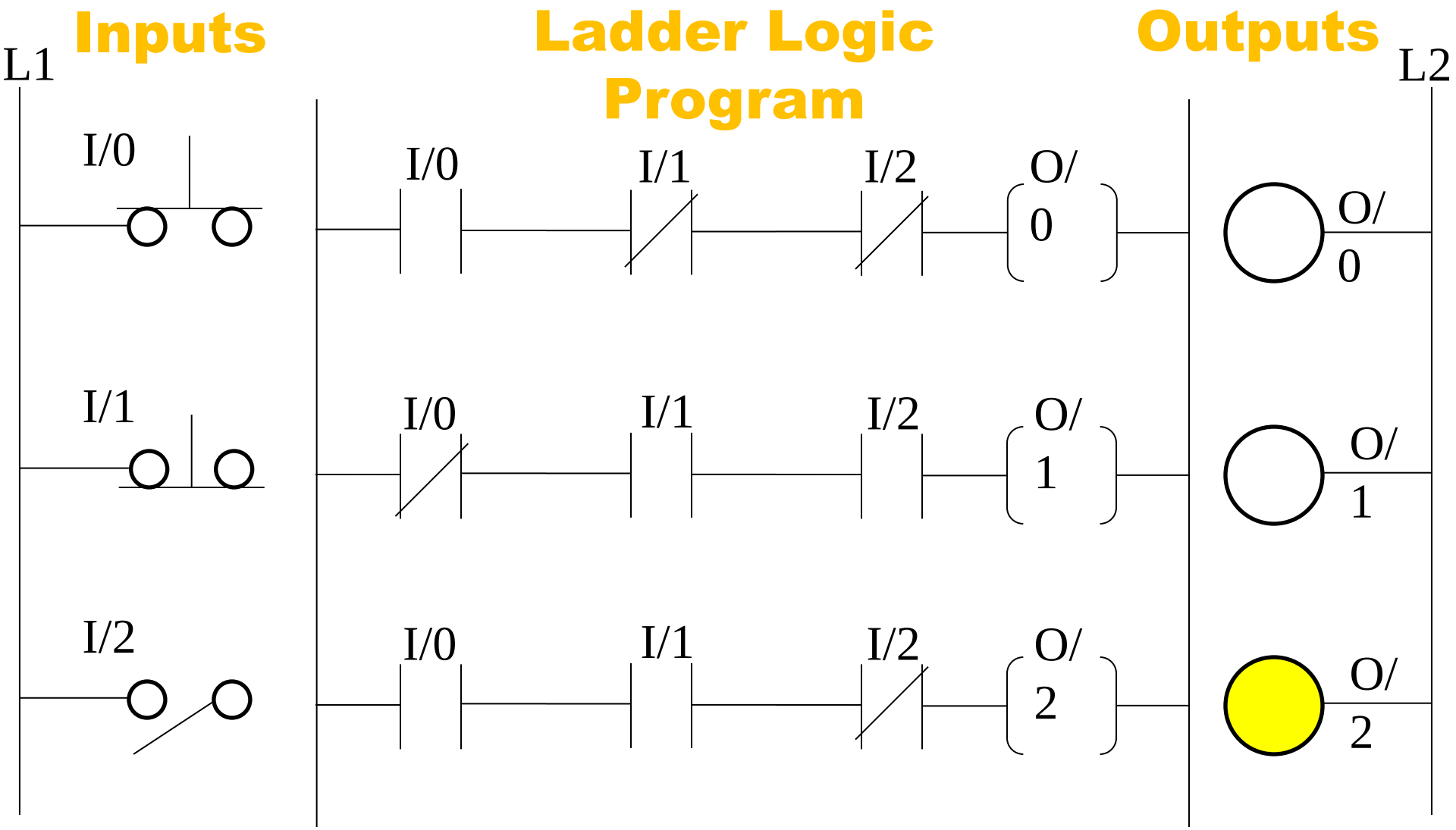
- **Output O/1 will energize**



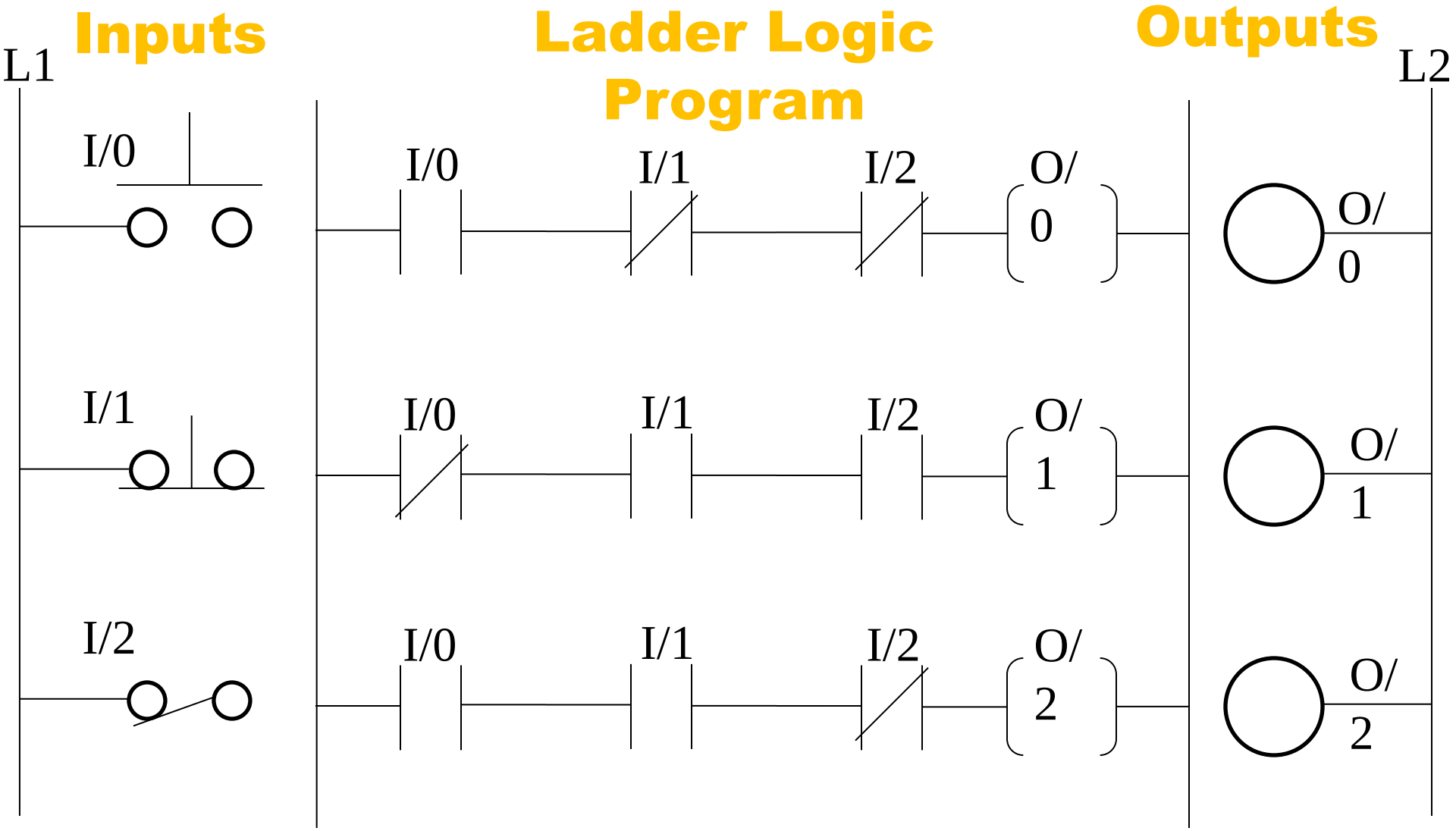
- **Look at the inputs and determine which output will energize**



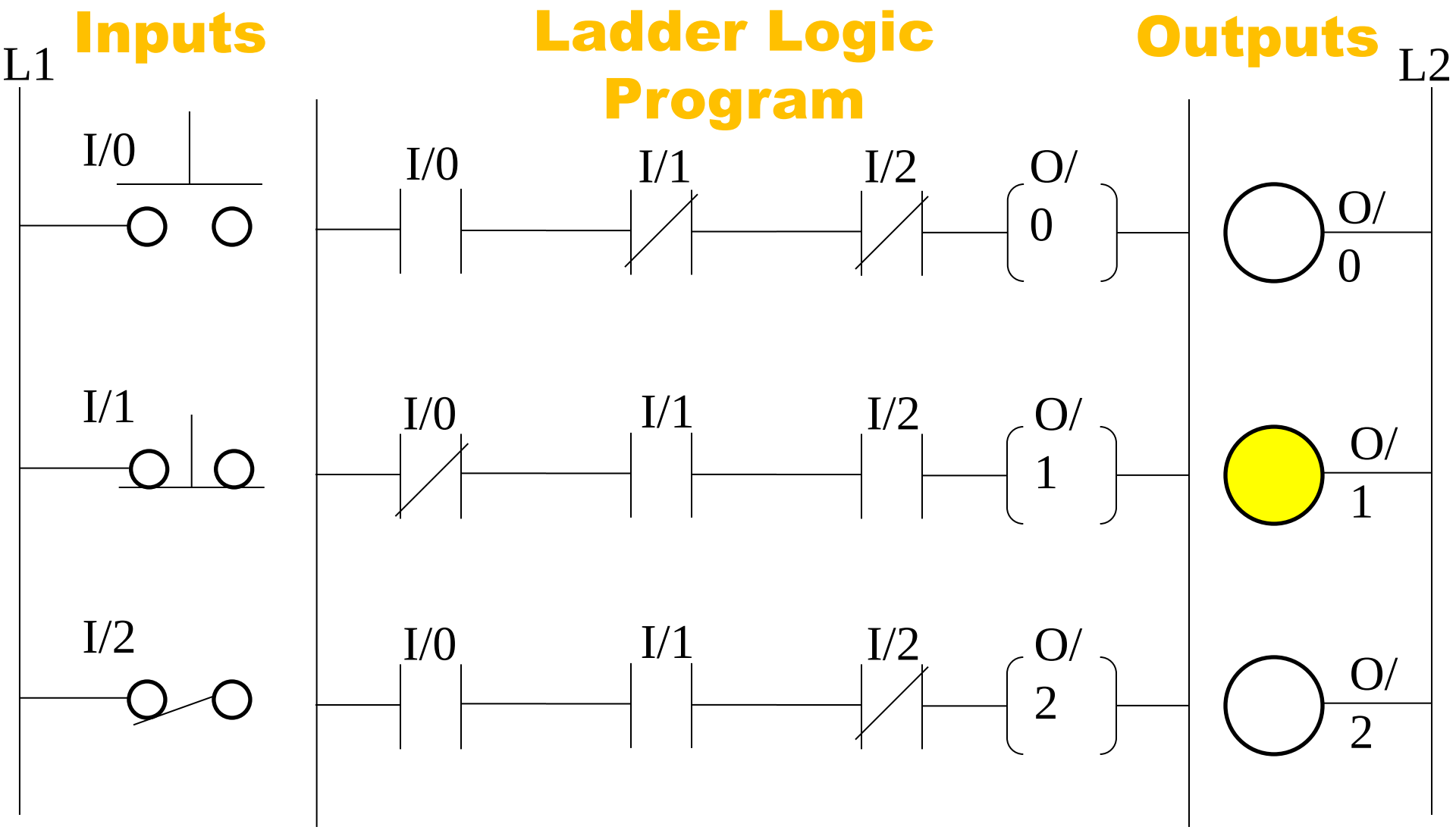
- **Output O/2 will energize**



- **Look at the inputs and determine which output will energize**



- **Output O/1 will energize**



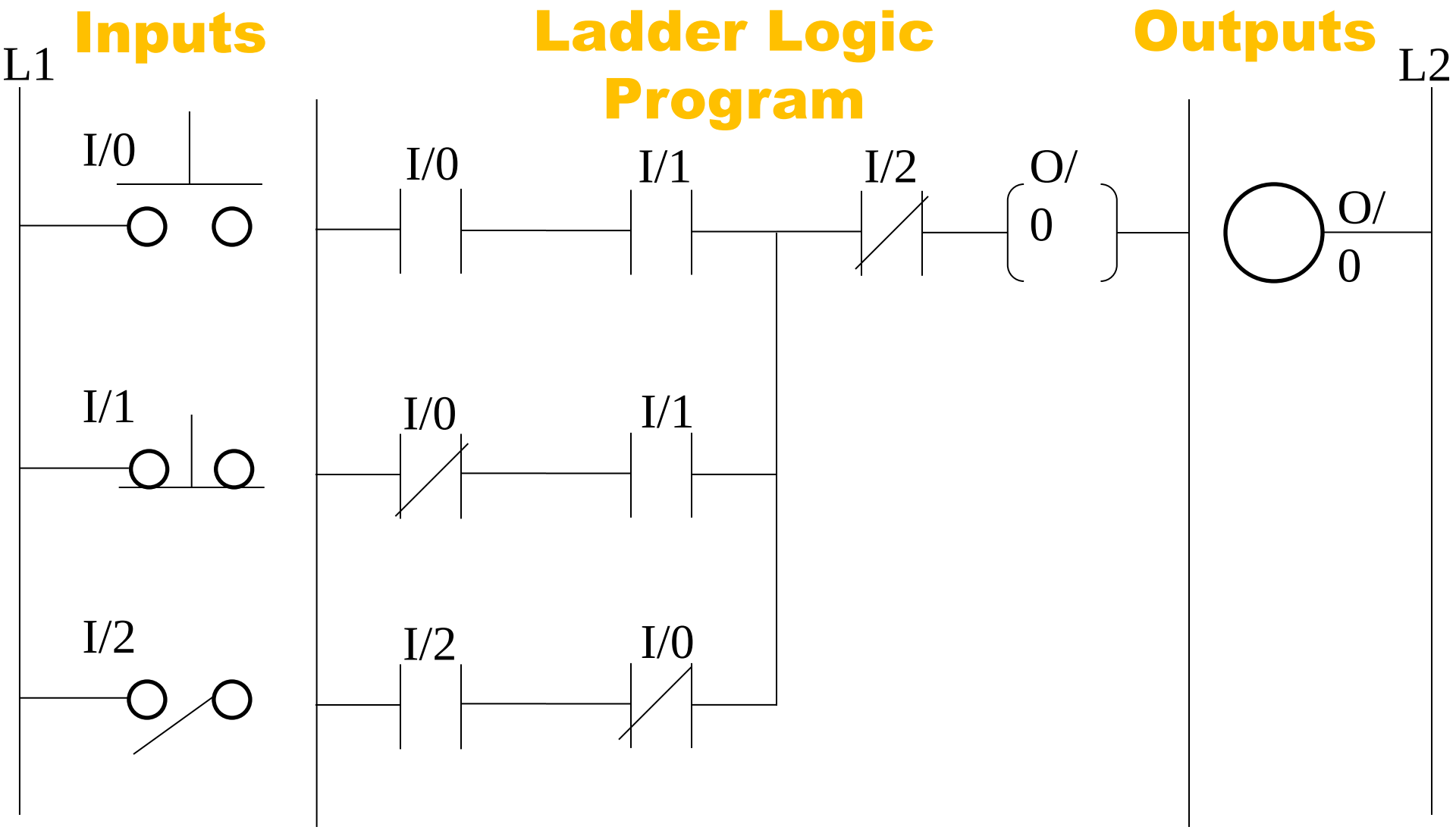
DETERMINING THE FLOW OF LOGIC WITH BRANCH INSTRUCTIONS

- Branch instructions are used to create parallel path for logic instructions to flow
- When determining the flow of logic begin at the top of the ladder diagram and trace a logic path from left to right
- When an instruction is false the flow of logic stops and moves downward in the program

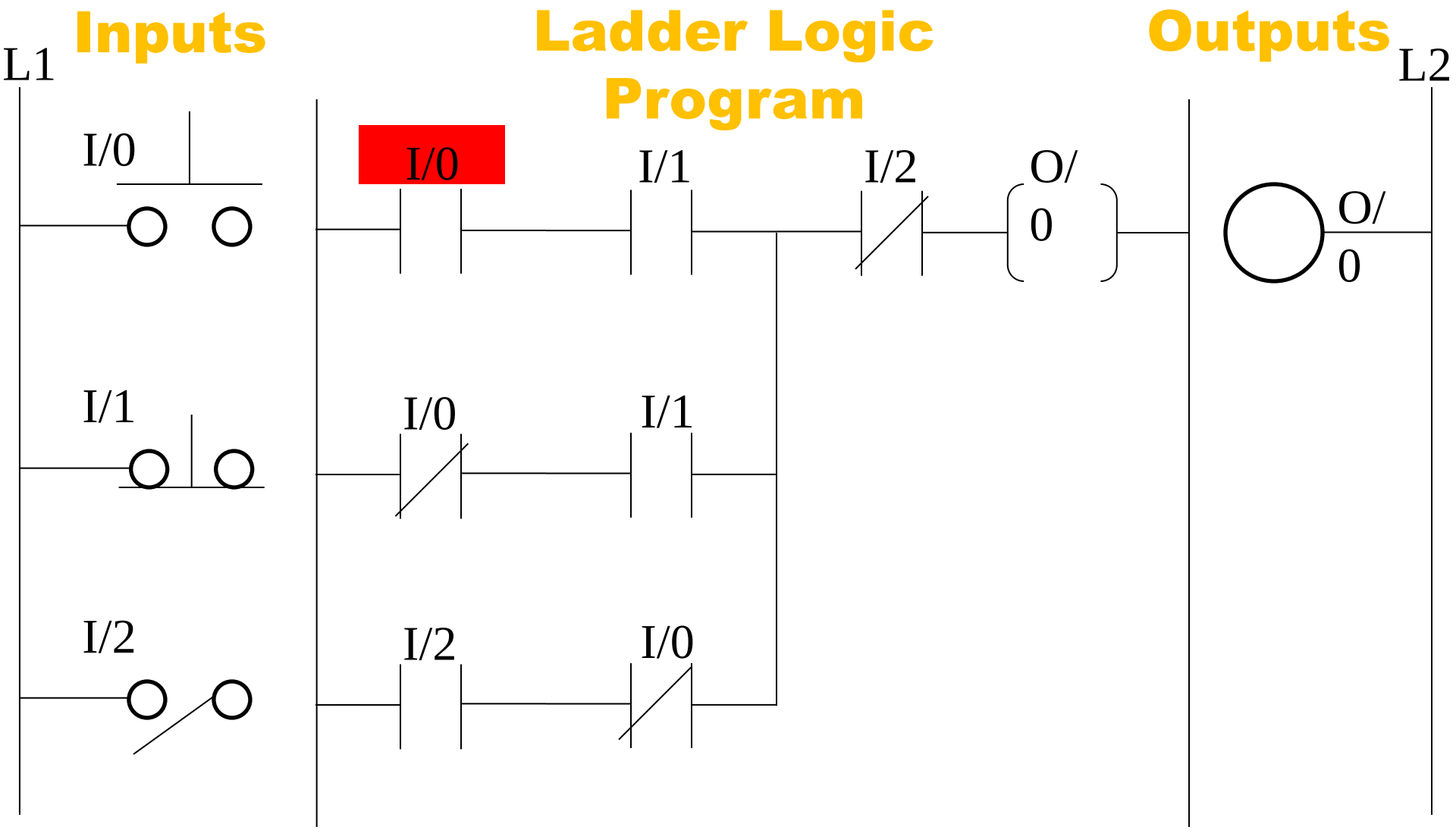
DETERMINING THE FLOW OF LOGIC WITH BRANCH INSTRUCTIONS

- When an instruction is false move down in the program and look for another path
- When a path of true instructions exists from left to right, then the output is energized

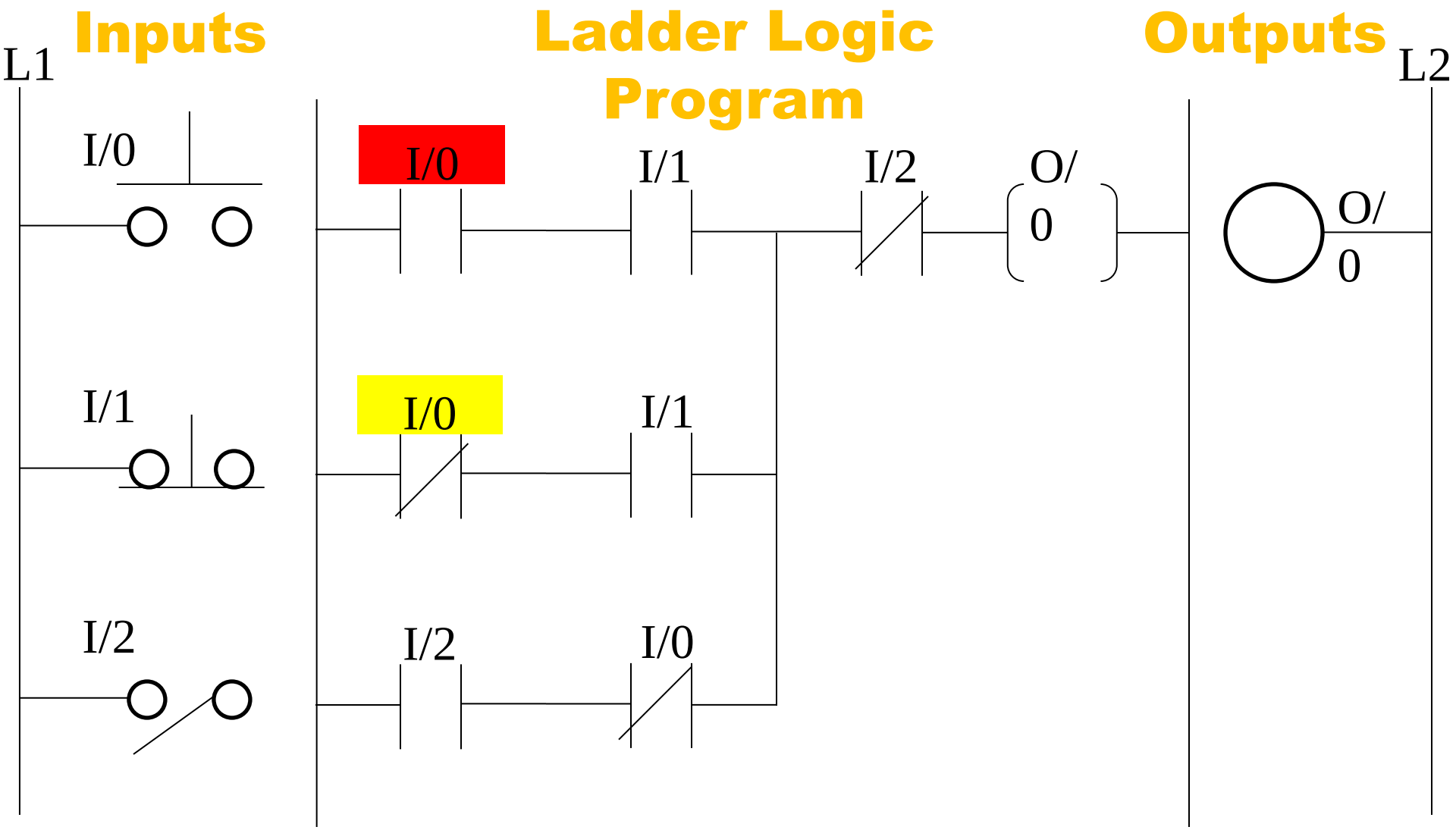
- **Look at the inputs and determine if output O/0 will energize**



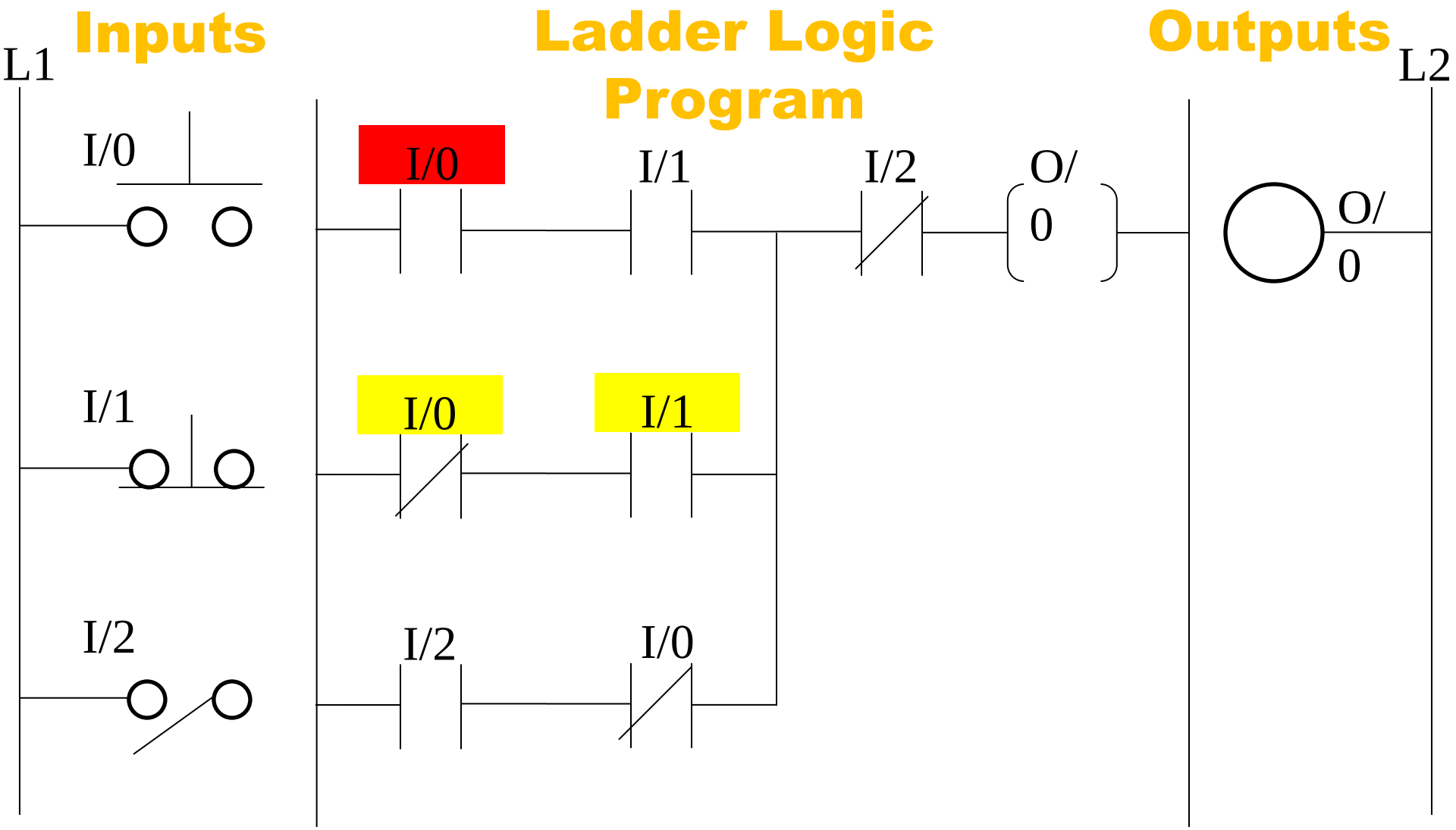
- **Look for a path that allows for the flow of logic**



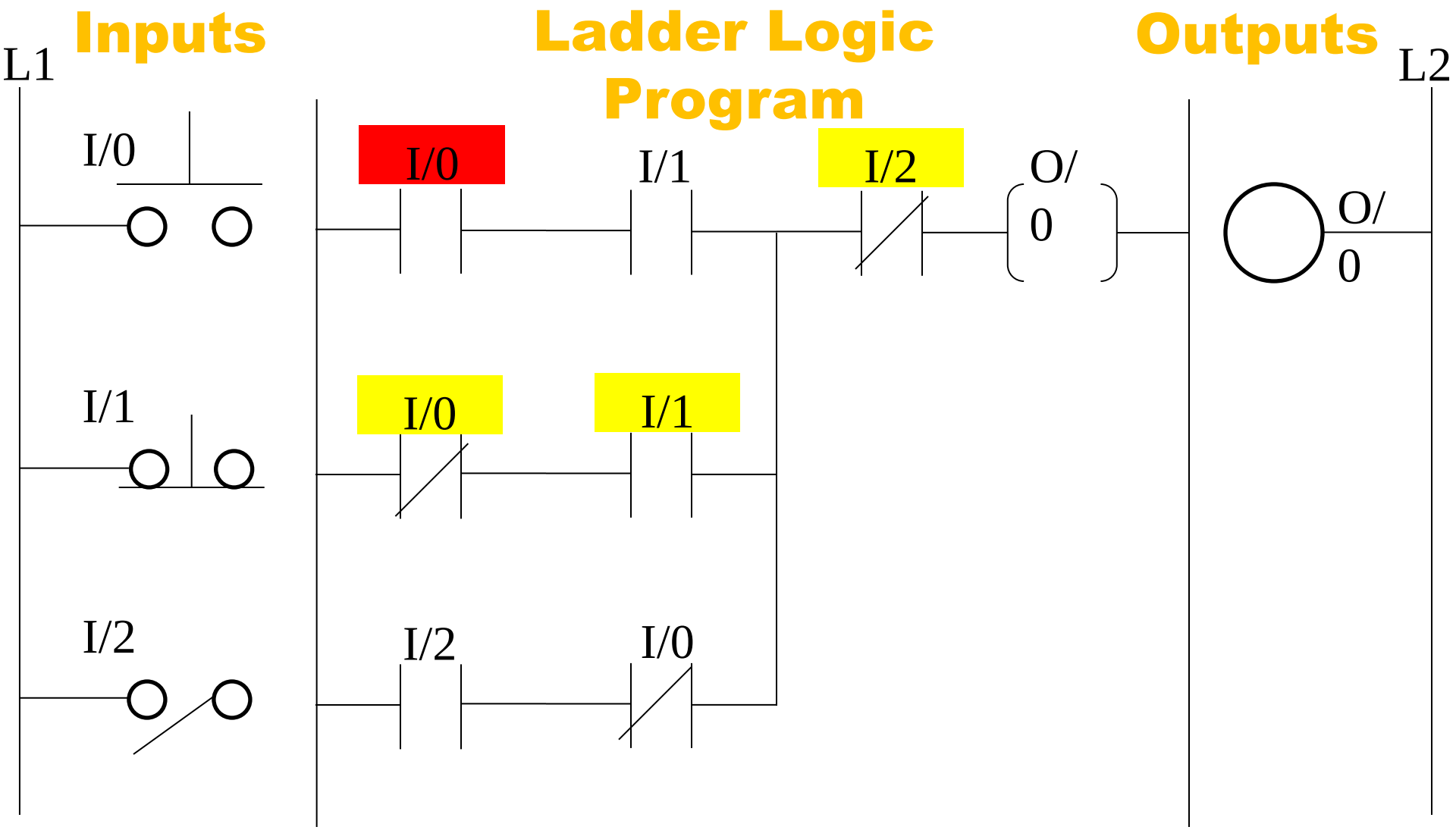
- **Look for a path that allows for the flow of logic**



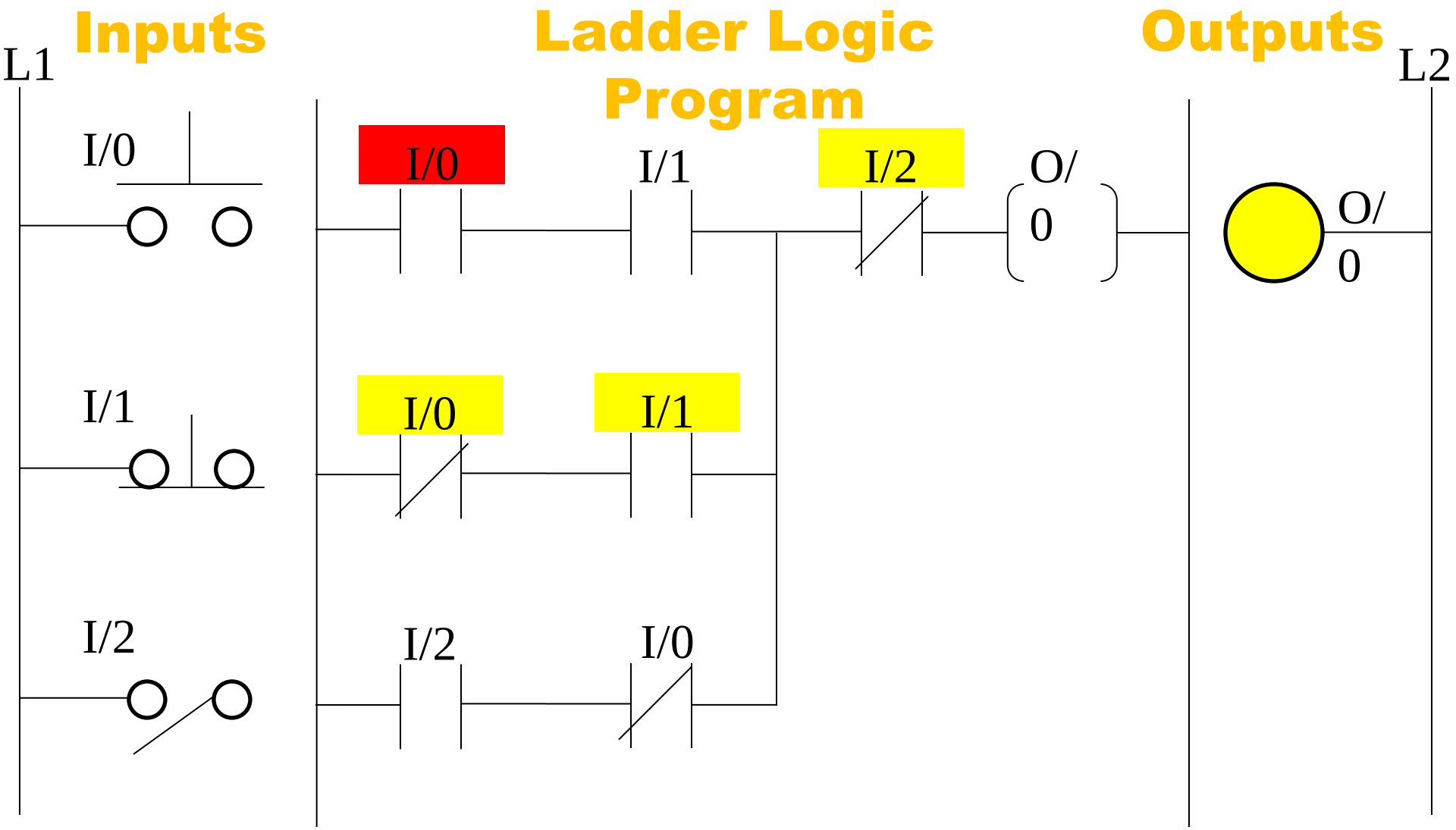
- **Look for a path that allows for the flow of logic**



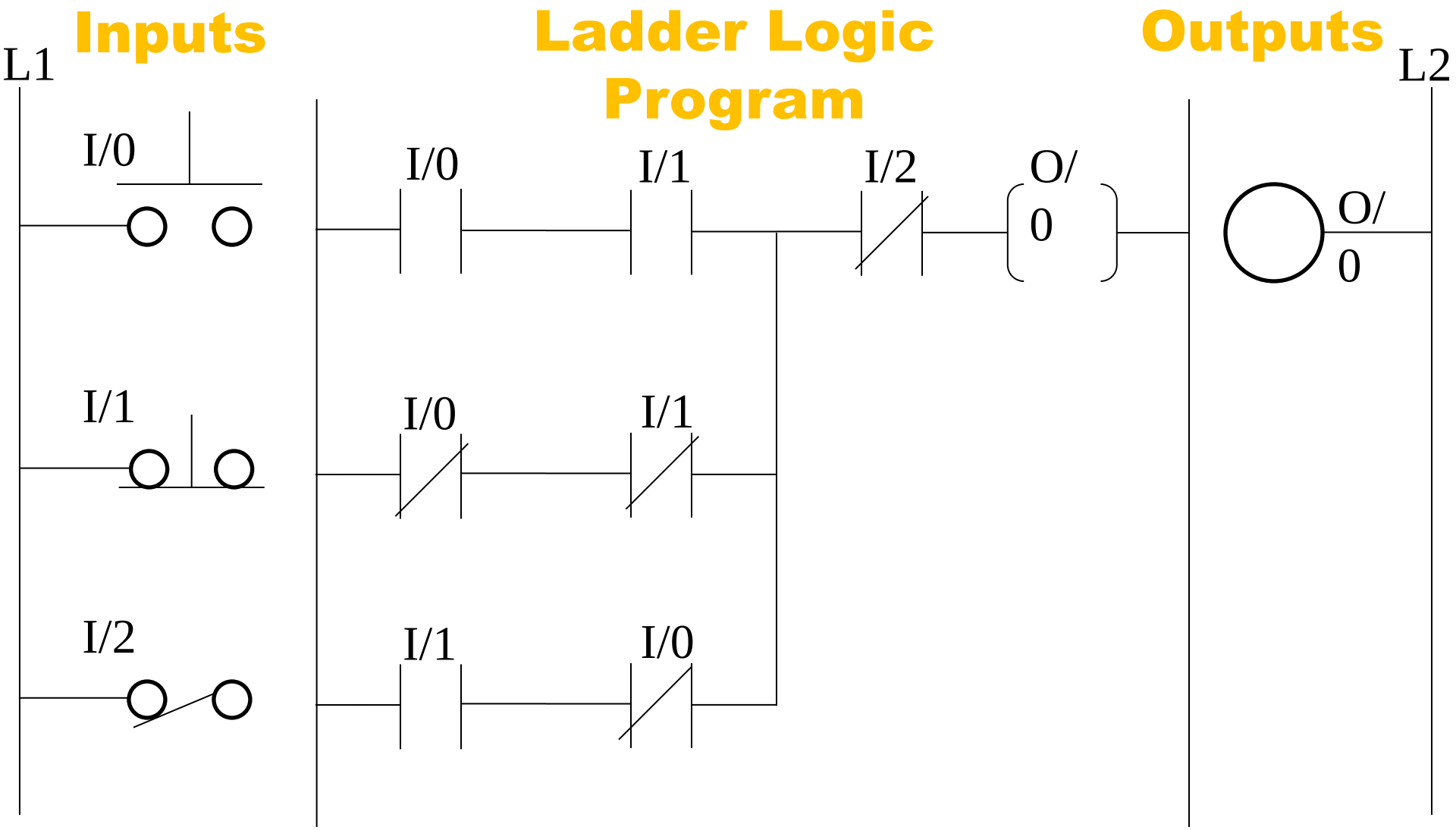
- **Look for a path that allows for the flow of logic**



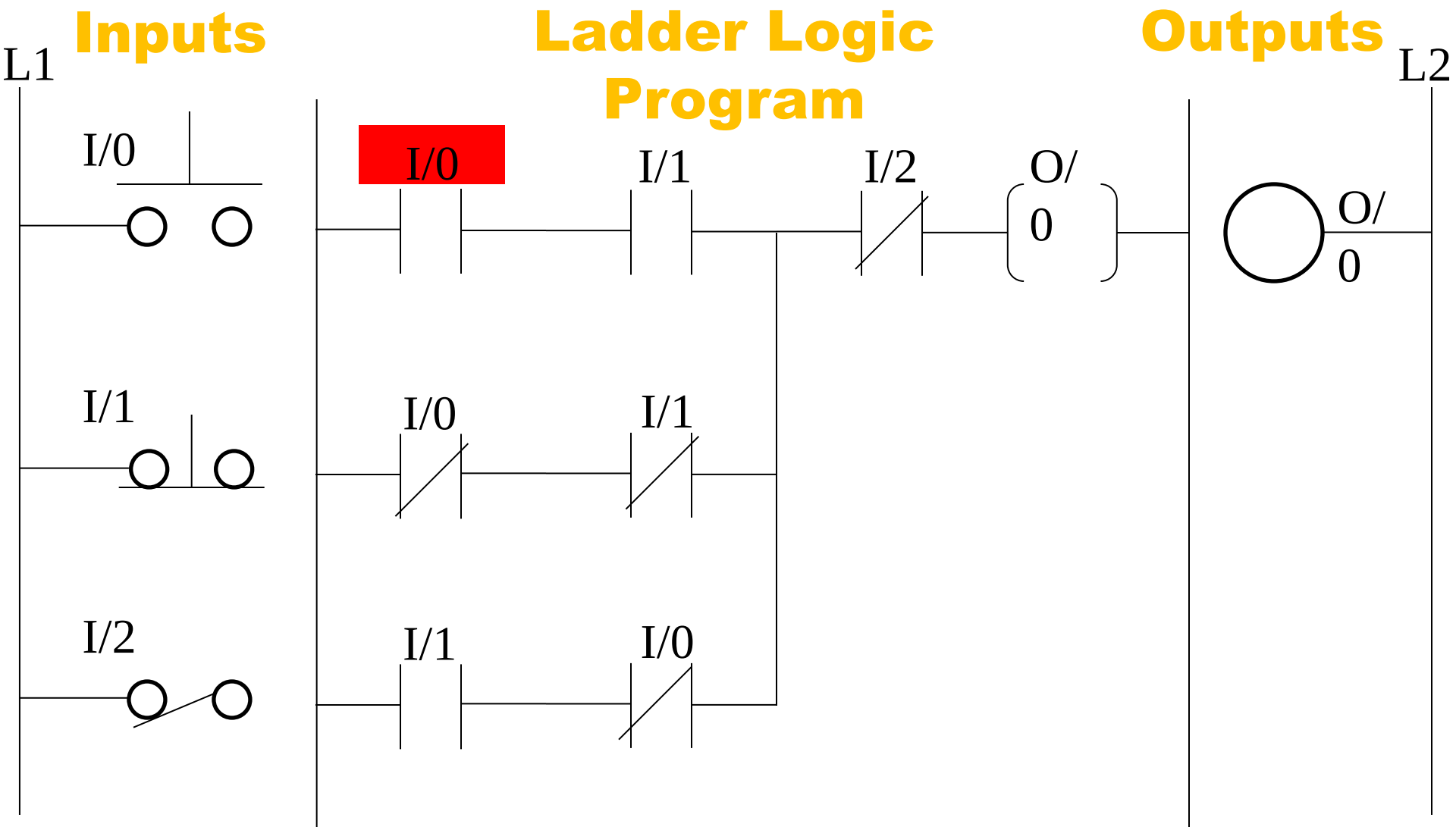
- **Yes**


Inputs
Ladder Logic Program
Outputs
L1
L2
I/0
I/1
I/2
I/0
I/1
I/2
O/0
I/0
I/1
I/2
I/0
O/0

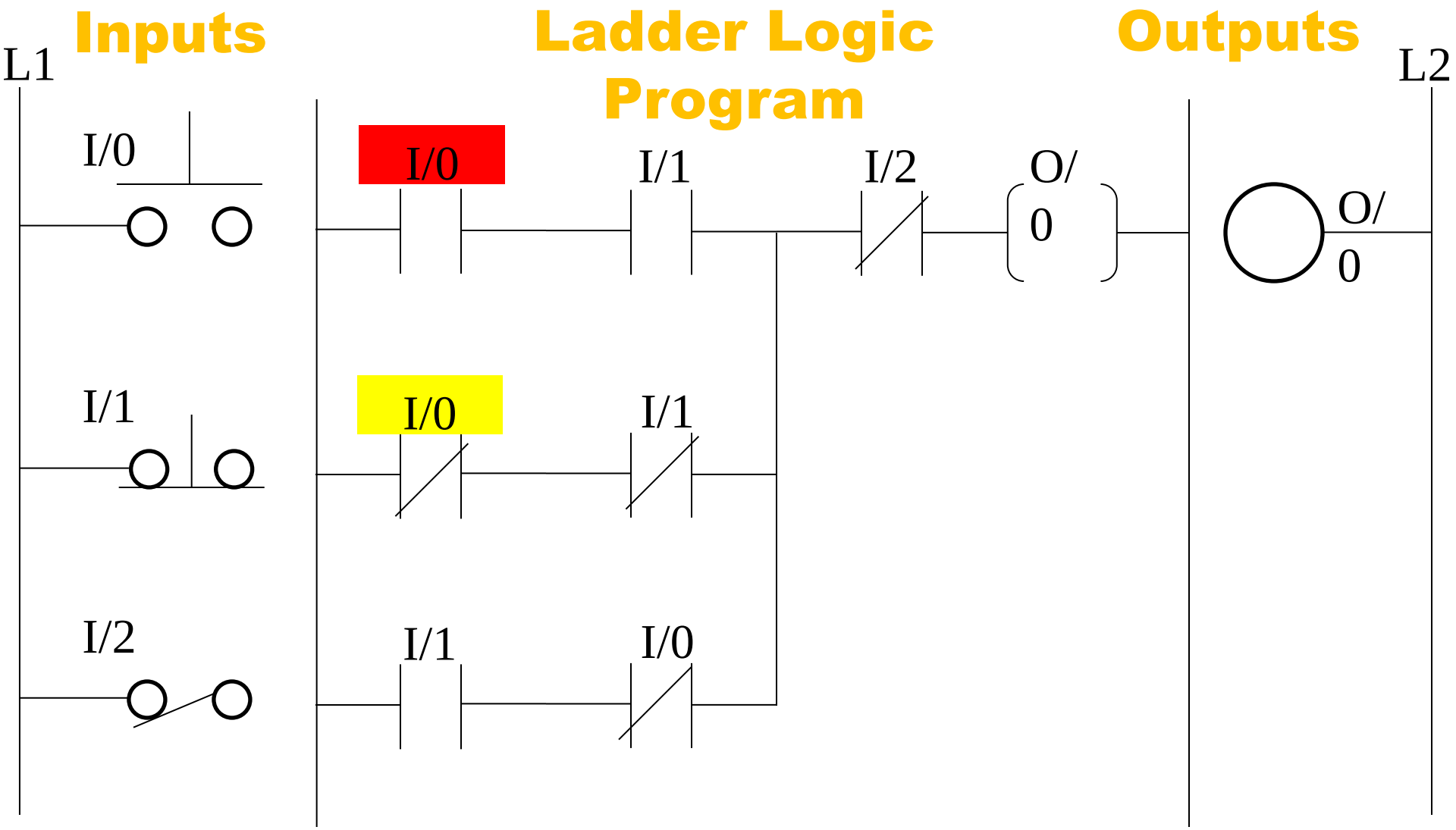
- **Look at the inputs and determine if output O/0 will energize**



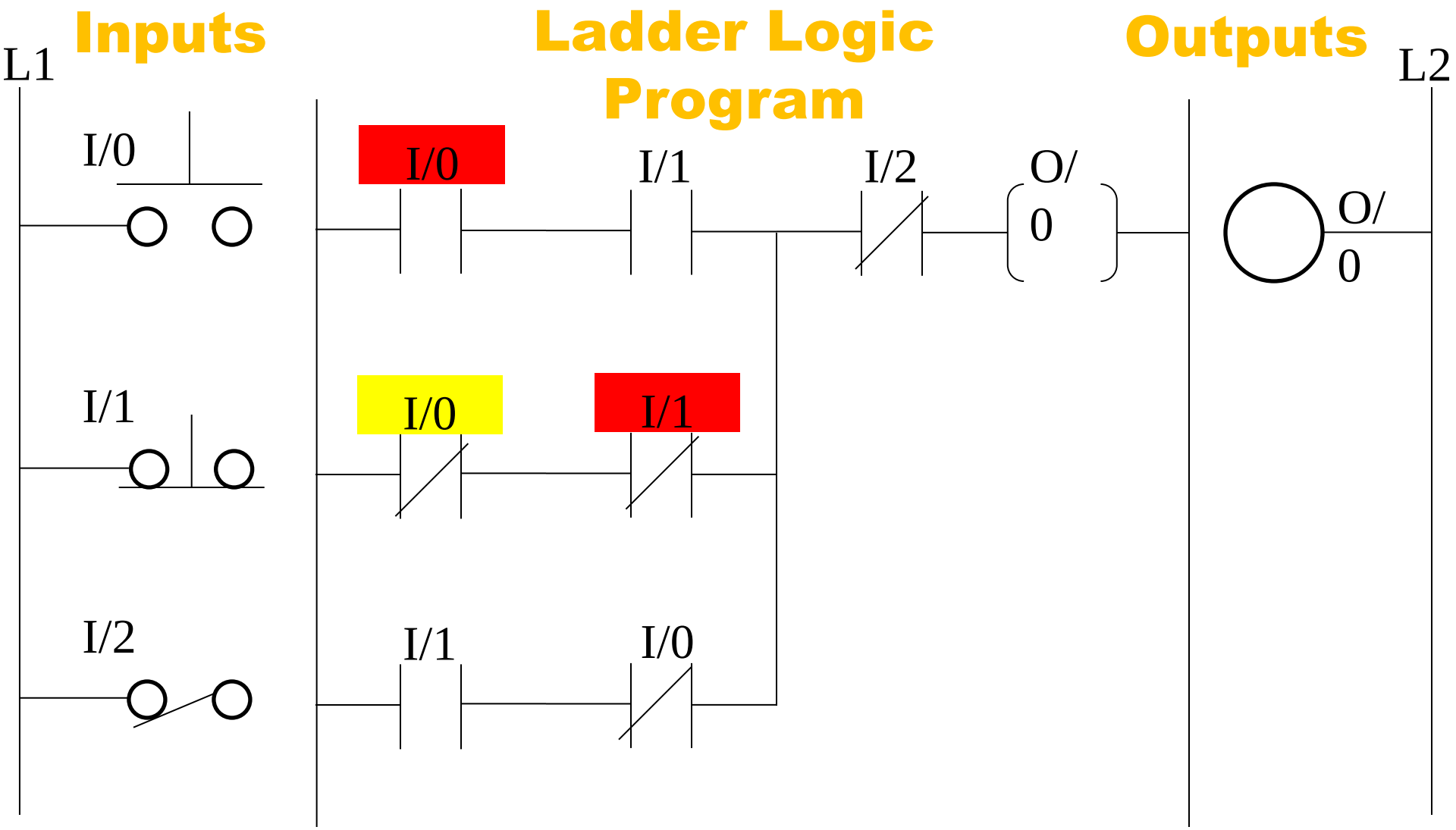
- **Look for a path that allows for the flow of logic**



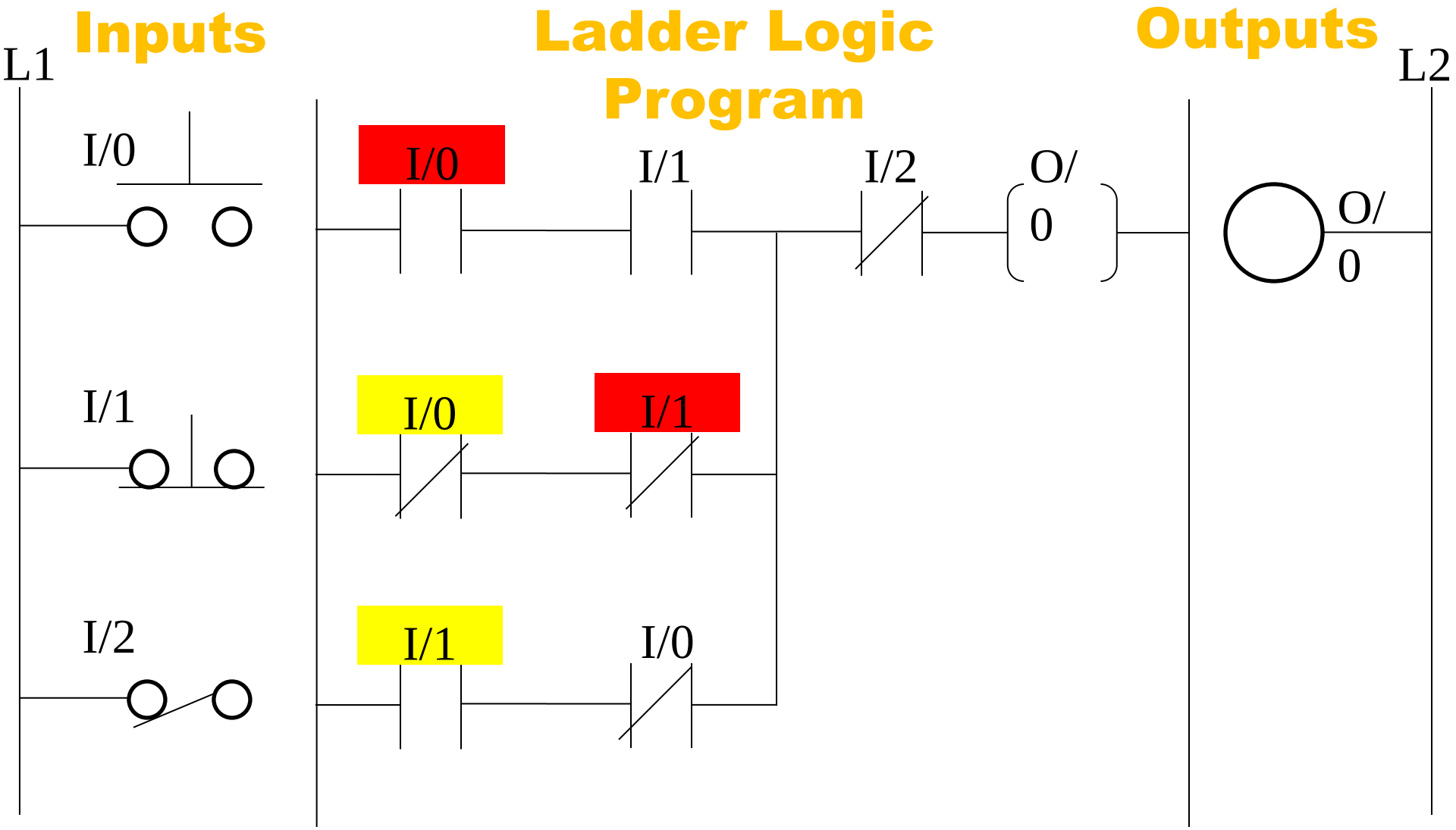
- **Look for a path that allows for the flow of logic**



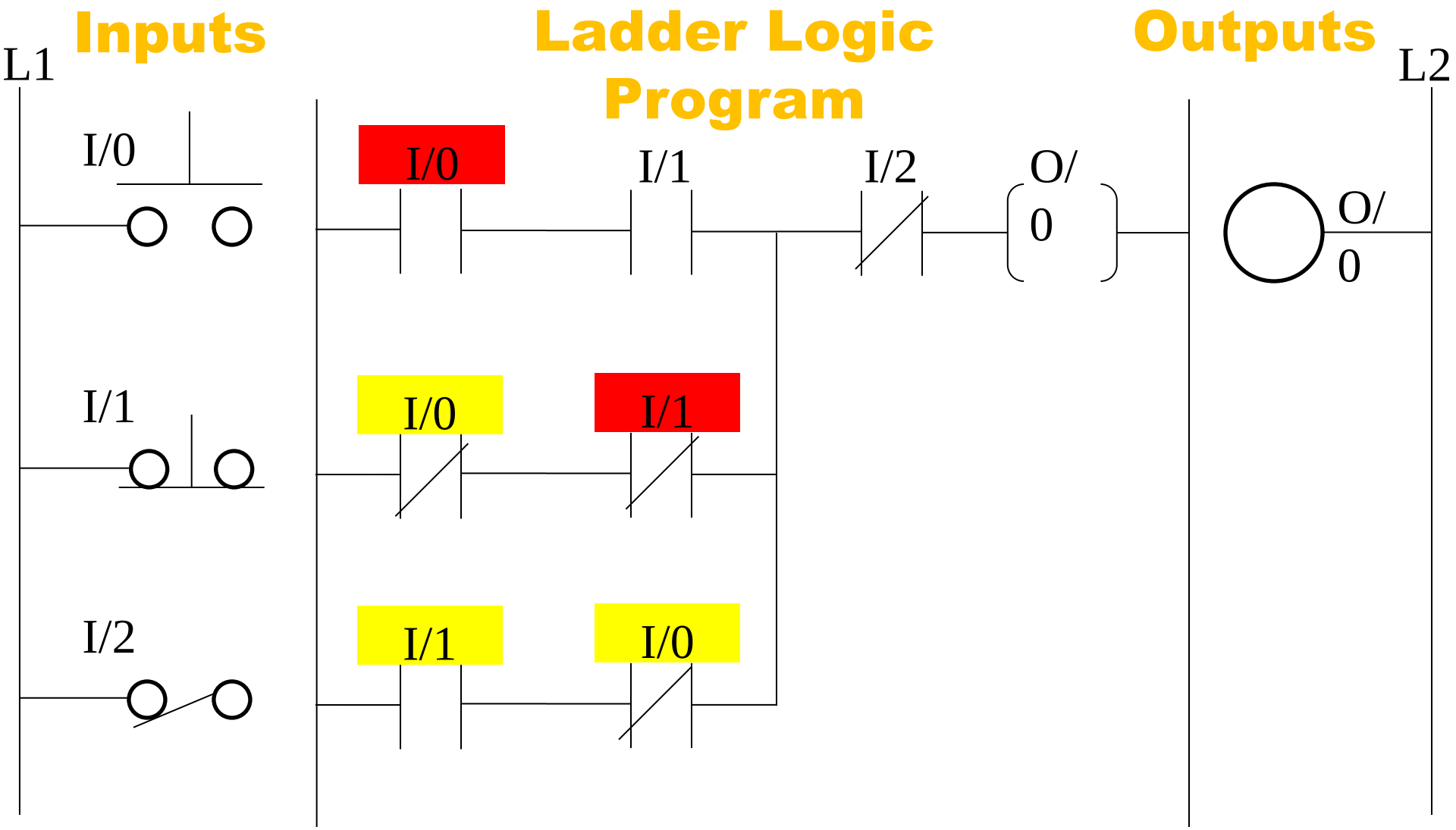
- **Look for a path that allows for the flow of logic**



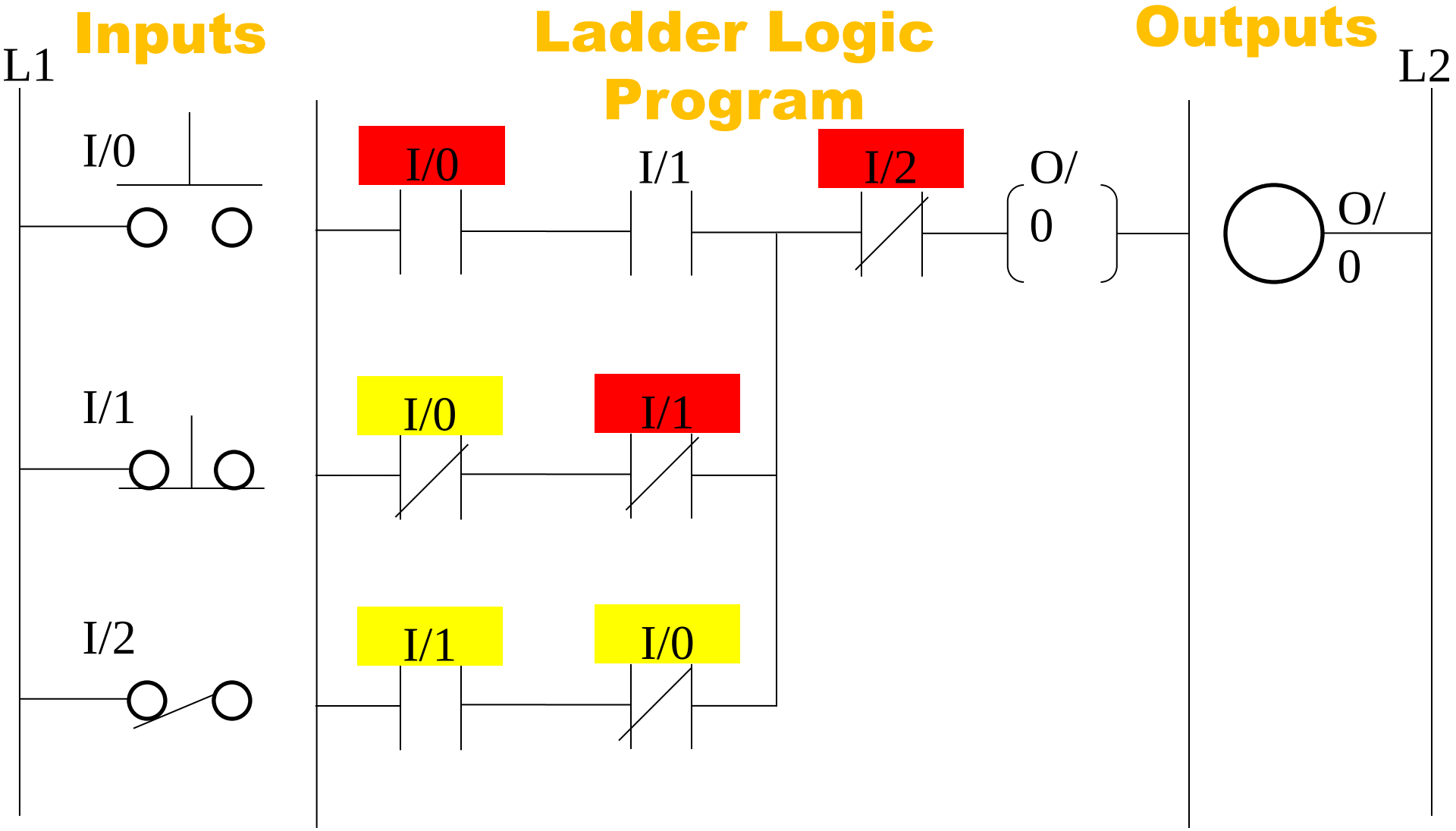
- **Look for a path that allows for the flow of logic**



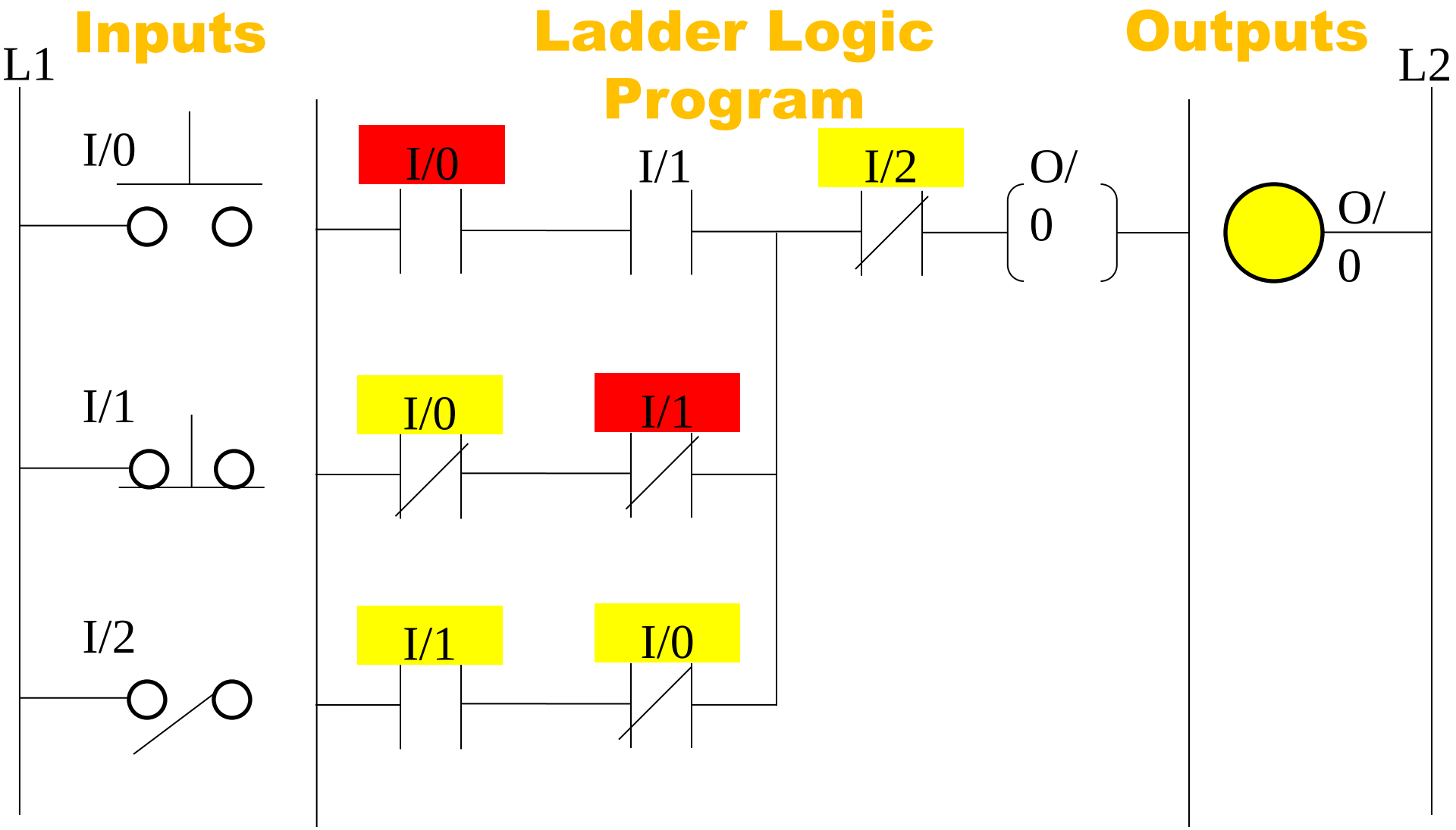
- **Look for a path that allows for the flow of logic**



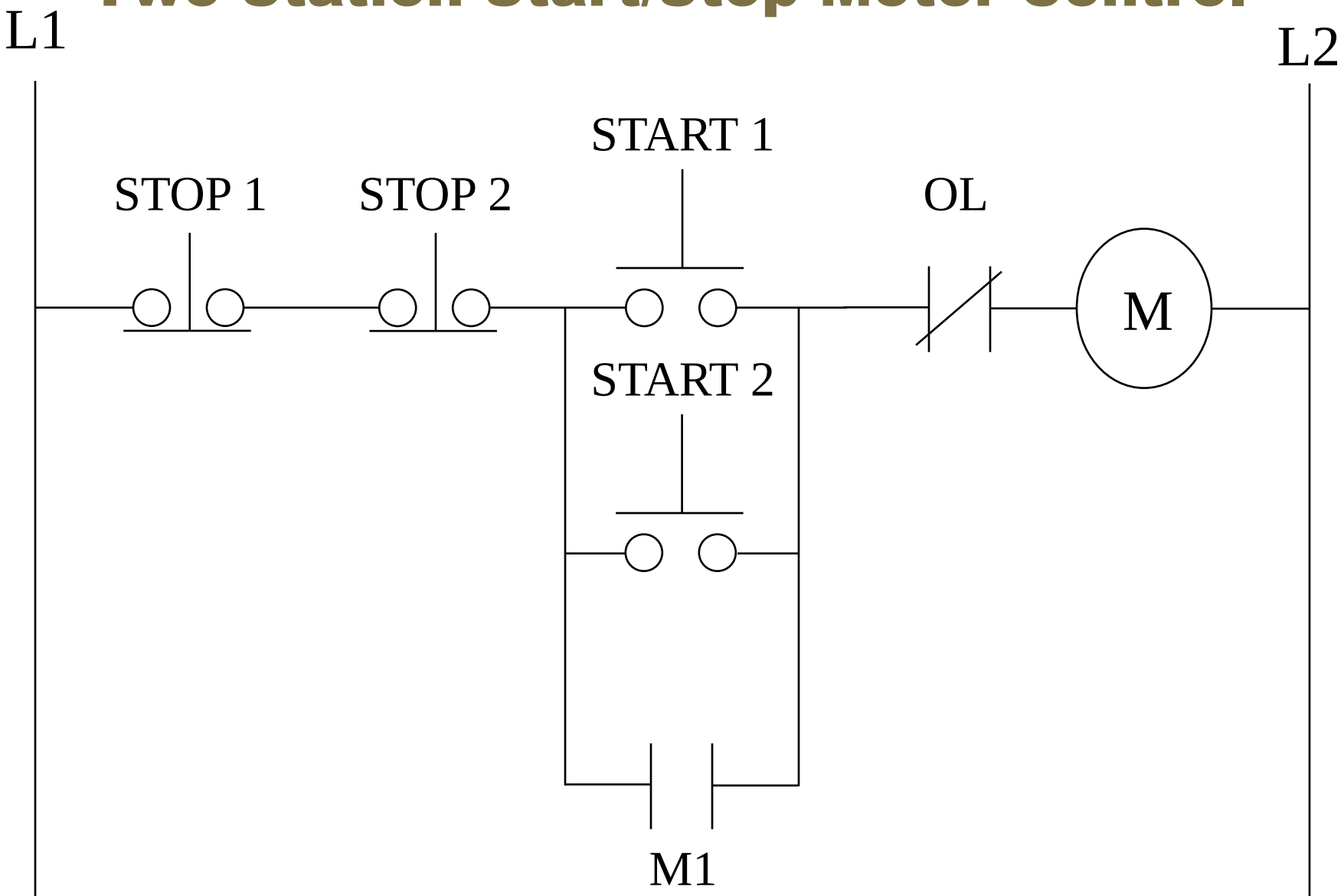
- **No I/2 must be open/off for O/0 to energize**



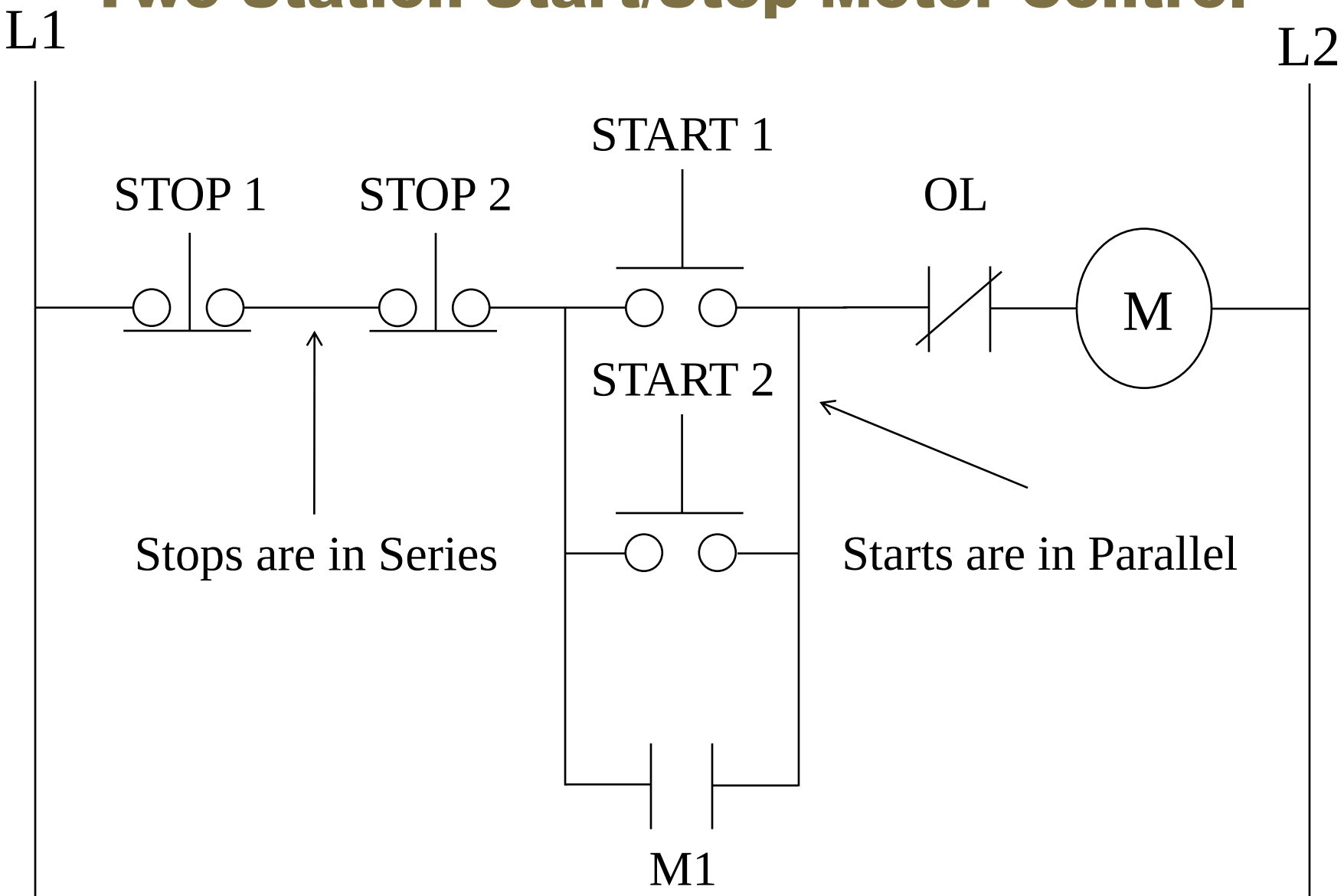
- **Now with I/2 being open/off, output O/0 will energize**



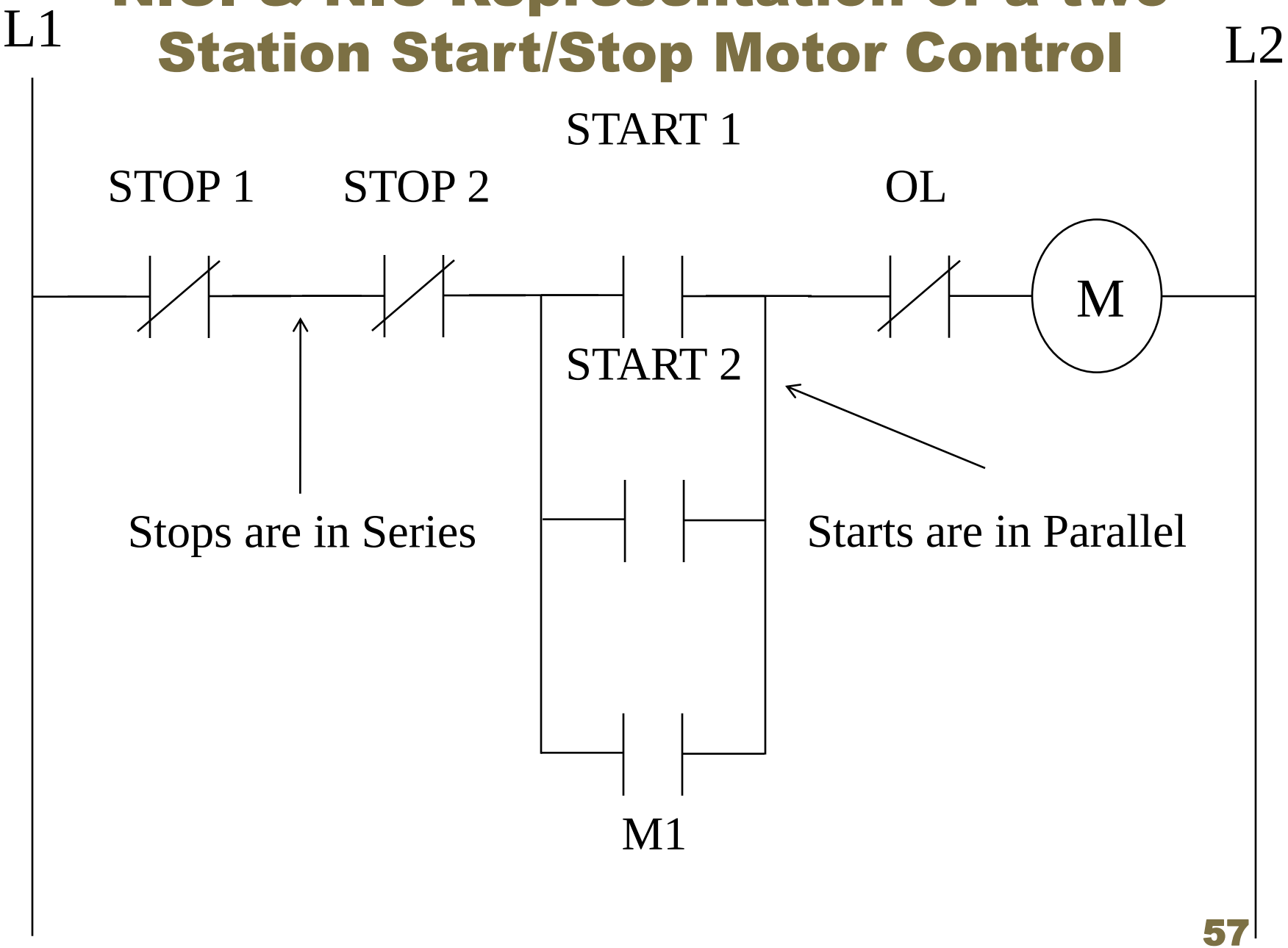
• Two Station Start/Stop Motor Control



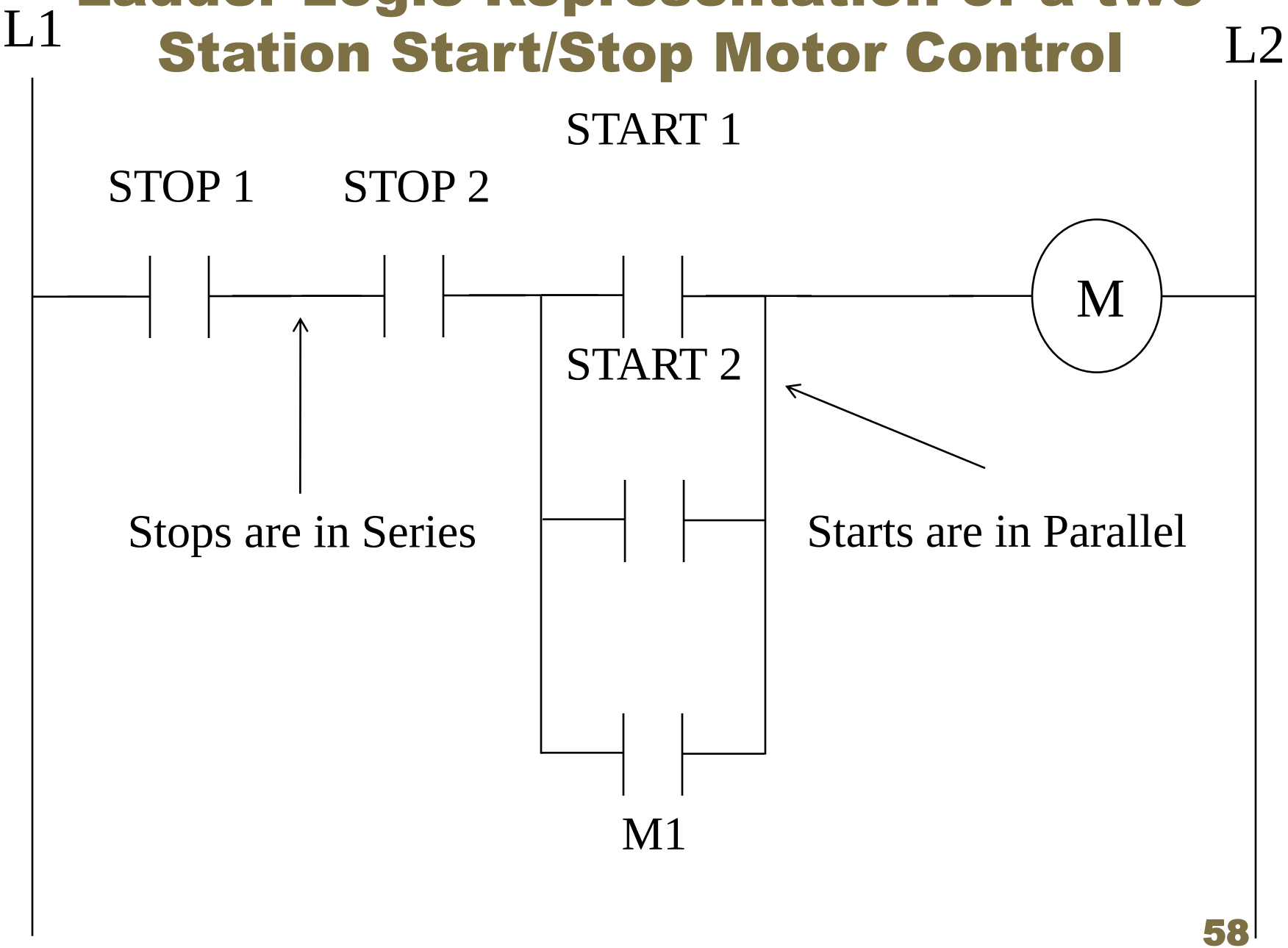
• Two Station Start/Stop Motor Control



• N.O. & N.C Representation of a two Station Start/Stop Motor Control



• Ladder Logic Representation of a two Station Start/Stop Motor Control



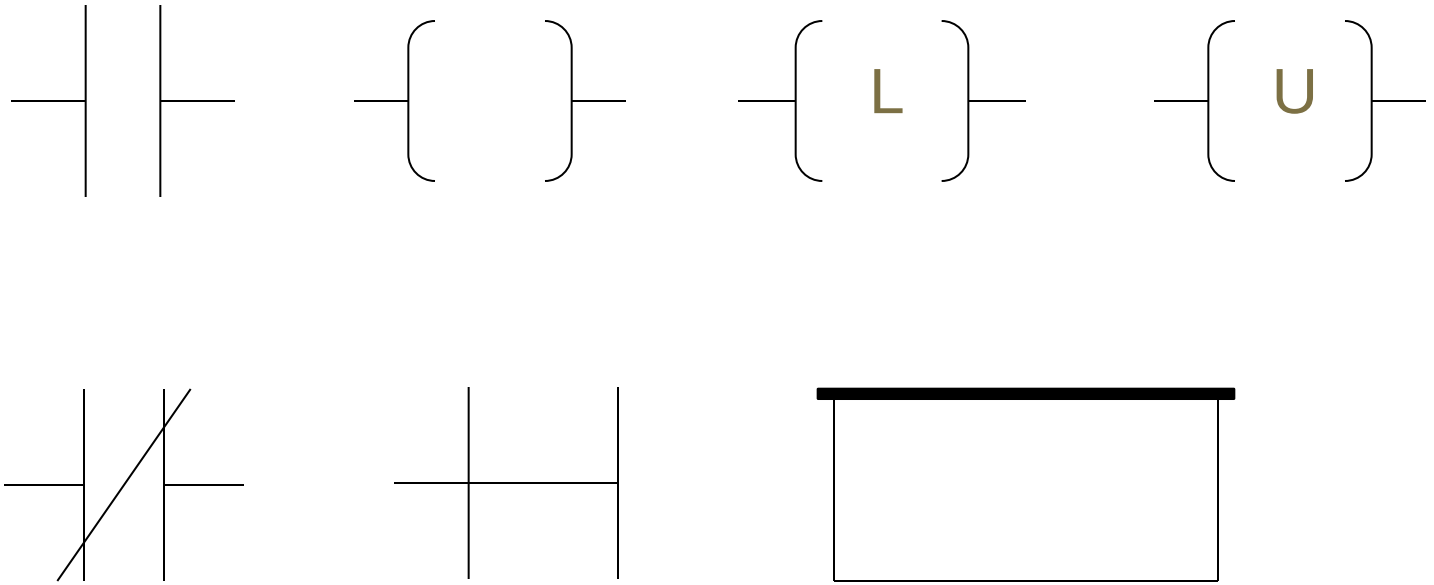
SESSION II

CONVERTING RELAY LOGIC TO LADDER LOGIC

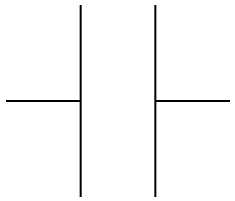
SESSION II: LEARNING OBJECTIVES

- At the conclusion of this session, participants will understand principles related to:
  - Relay Logic Instructions
  - Converting Hardwired Circuits

RELAY LOGIC INSTRUCTIONS

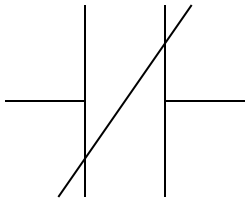


EXAMINE ON OR EXAMINE IF CLOSED (XIC)



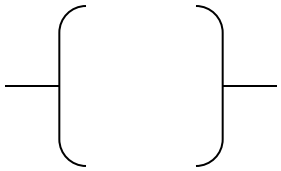
- Examine On refers to a normally open contact instruction in a logic ladder program
- Examine On is true if its addressed bit is on (1)
- Examine On is false if its addressed bit is off (0)

EXAMINE OFF OR EXAMINE IF OPEN (XIO)

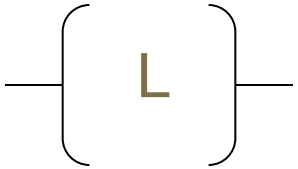


- Examine off refers to a normally closed contact instruction in a logic ladder program
- Examine off is true if its addressed bit is off (0)
- Examine off is false if its addressed bit is on (1)

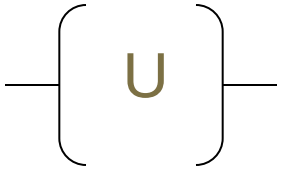
OUTPUT INSTRUCTIONS



- An Output instruction bit becomes true (1) when a complete path of logic exists

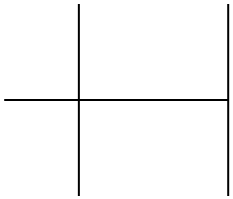


- A Latching Output instruction bit that emulates the latching action of a latching relay



- An Unlatching Output instruction bit de-energizes a latching output that has the same address

NEW RUNG AND BRANCH INSTRUCTIONS



- The new rung instruction adds a rung to the ladder logic diagram

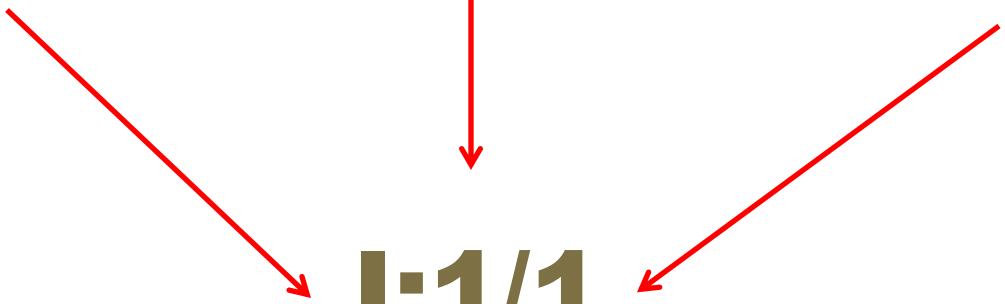
- The branch instruction adds a parallel logic path within a rung

ADDRESSING BIT INSTRUCTIONS

- Instruction addressing formats vary from one PLC family to another
- The address identifies the function of an instruction and links it to a particular bit in the data portion of the memory
- Address are formatted as file type, slot number, and bit number

ADDRESSING BIT INSTRUCTIONS

File Type Slot Number Bit Number



I:1/1

File Types

Input - I:1/2

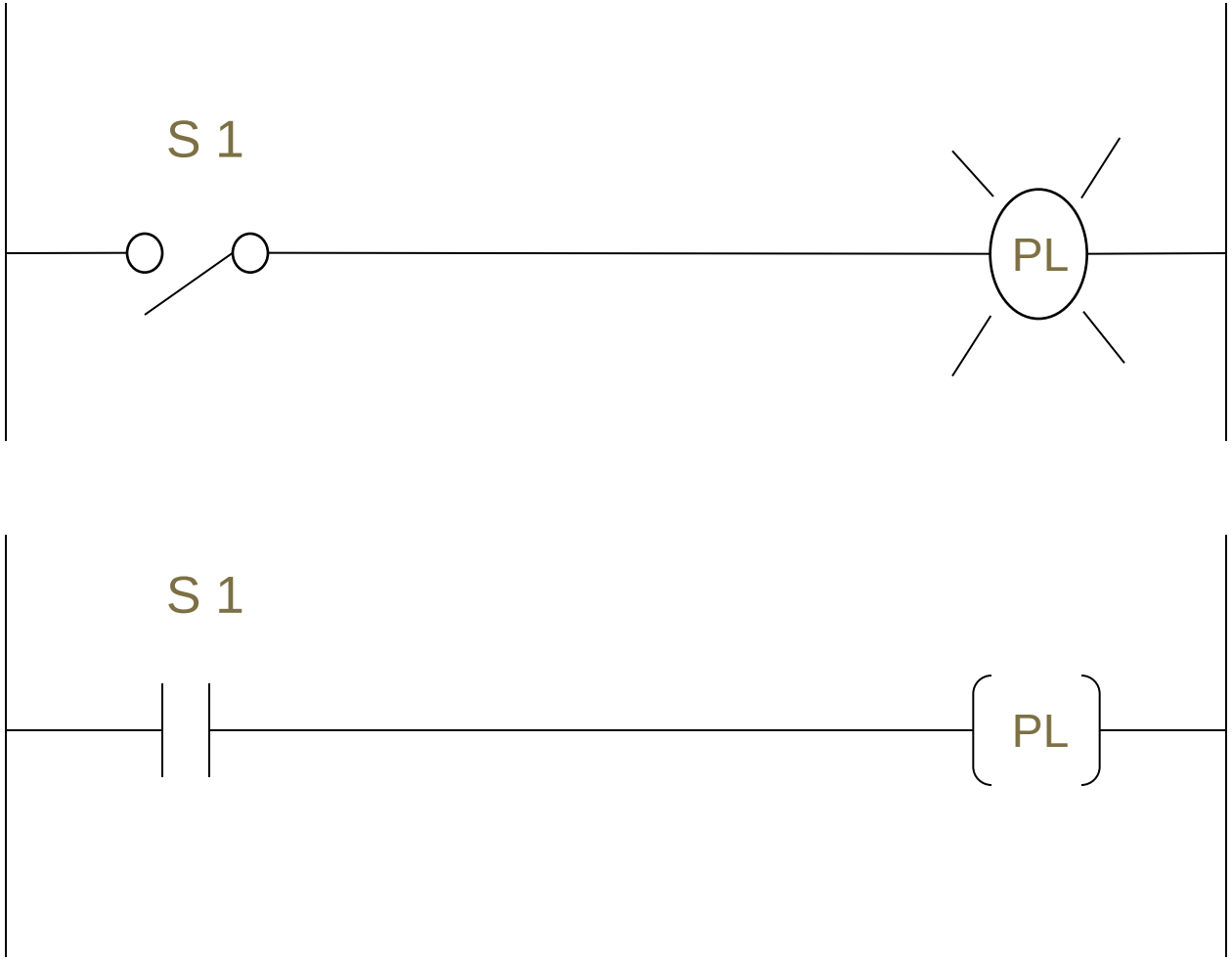
Output - O:1/0

Internal Bit- B3:2/3

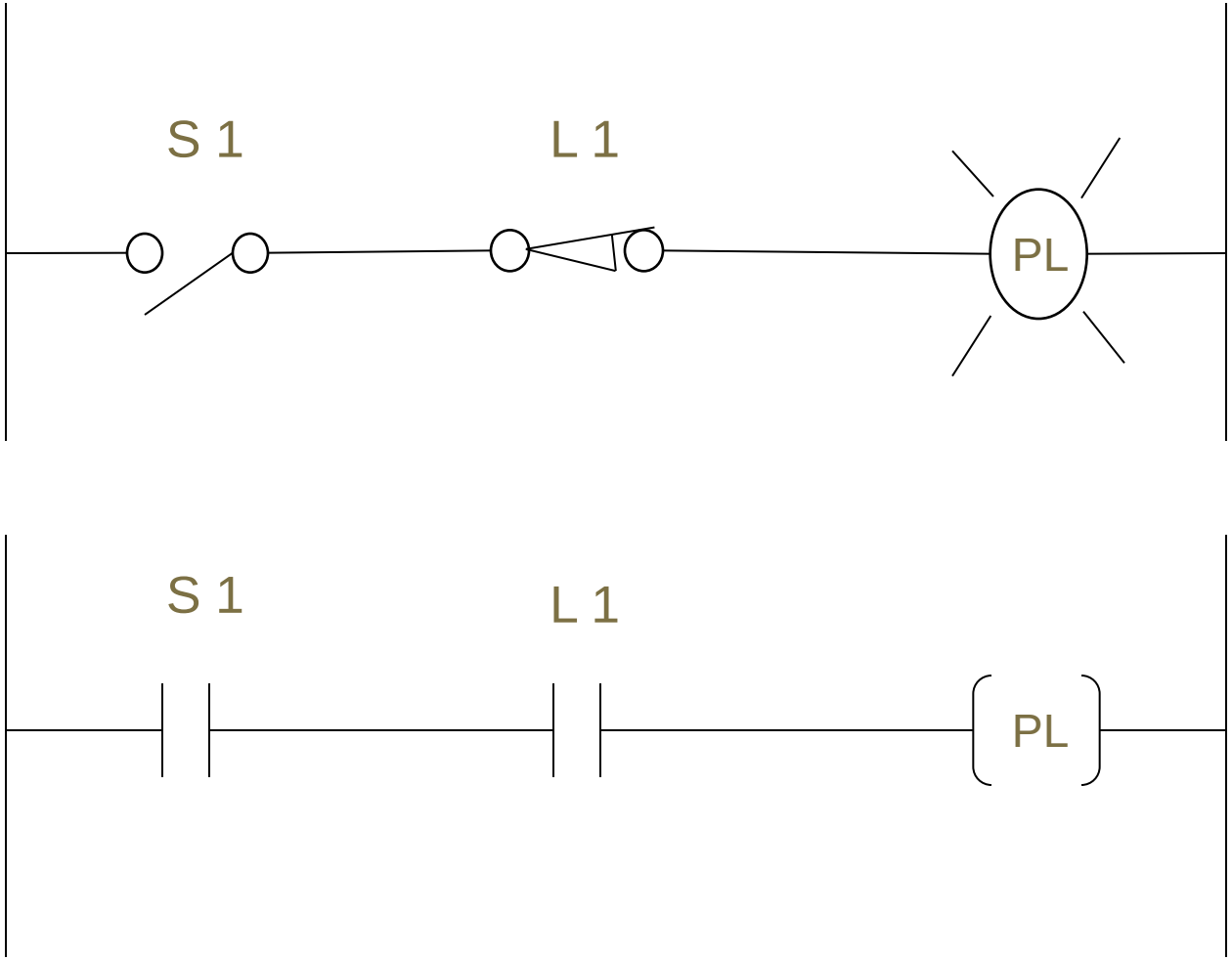
Timer – T4:0

Counter – C5:0

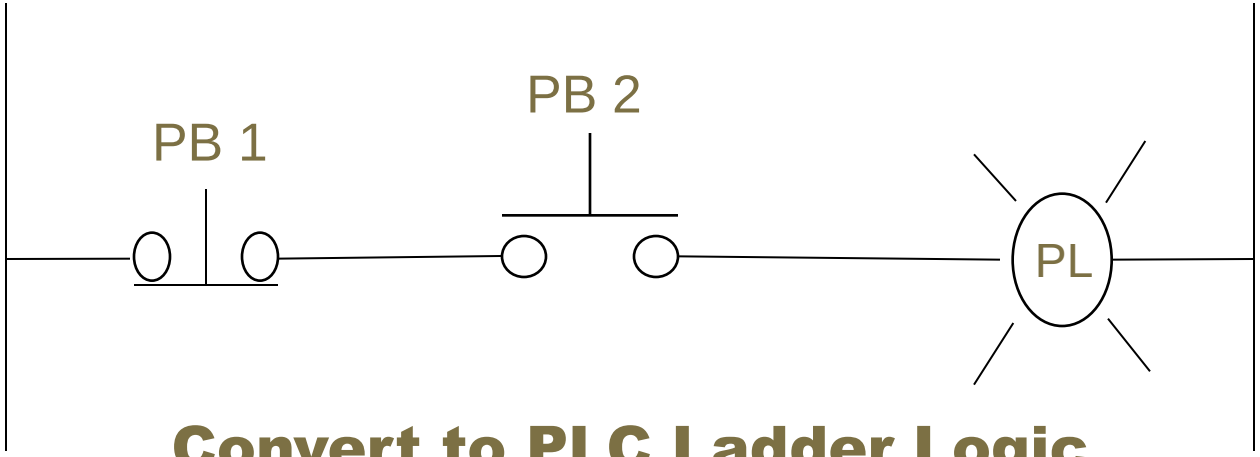
CONVERTING HARDWIRED CIRCUITS



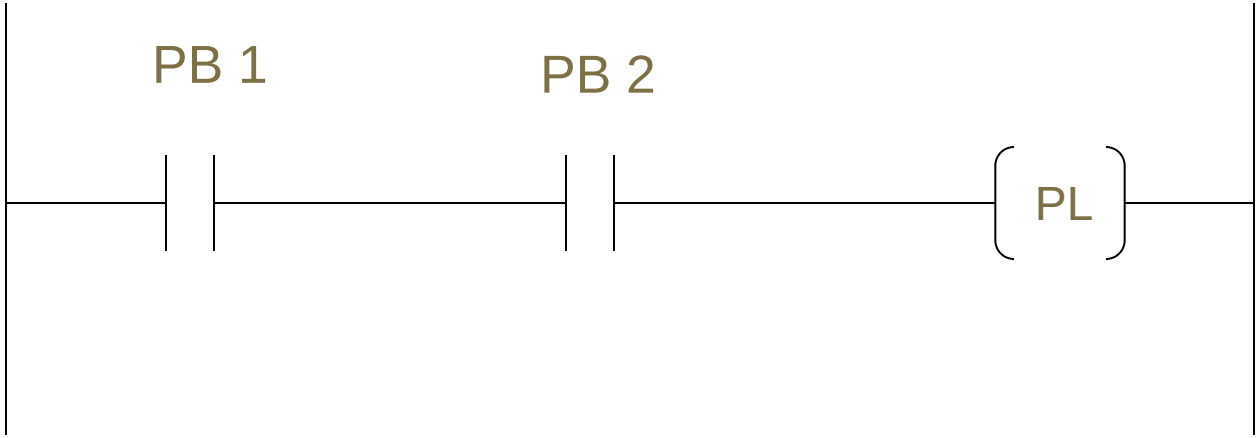
CONVERTING HARDWIRED CIRCUITS



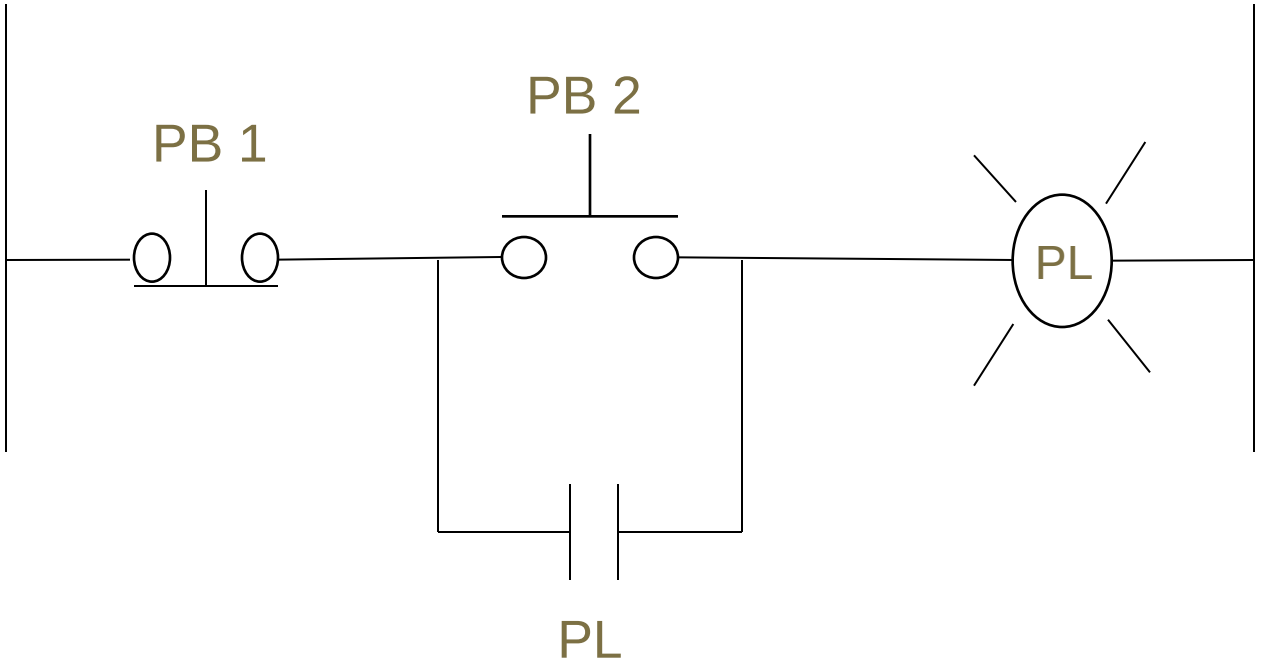
CONVERTING HARDWIRED CIRCUITS



Convert to PLC Ladder Logic

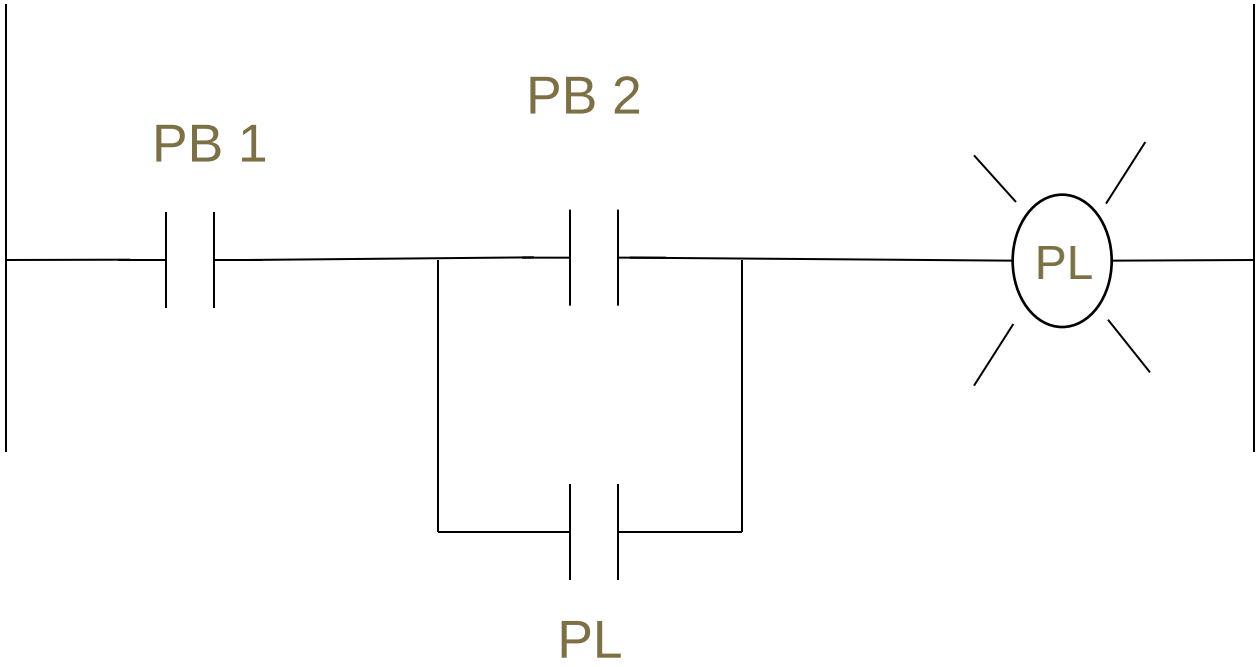


CONVERTING HARDWIRED CIRCUITS

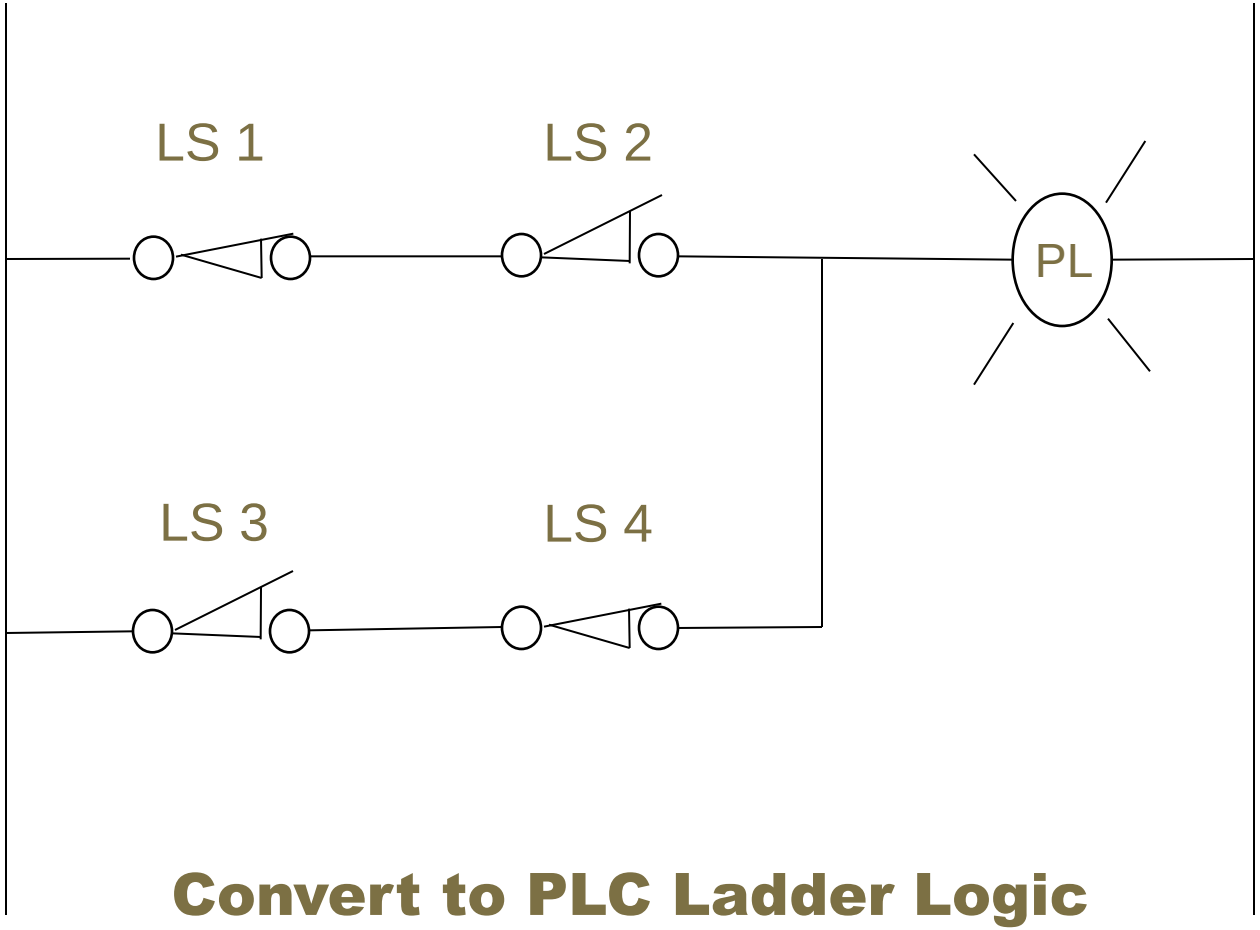


Convert to PLC Ladder Logic

CONVERTING HARDWIRED CIRCUITS

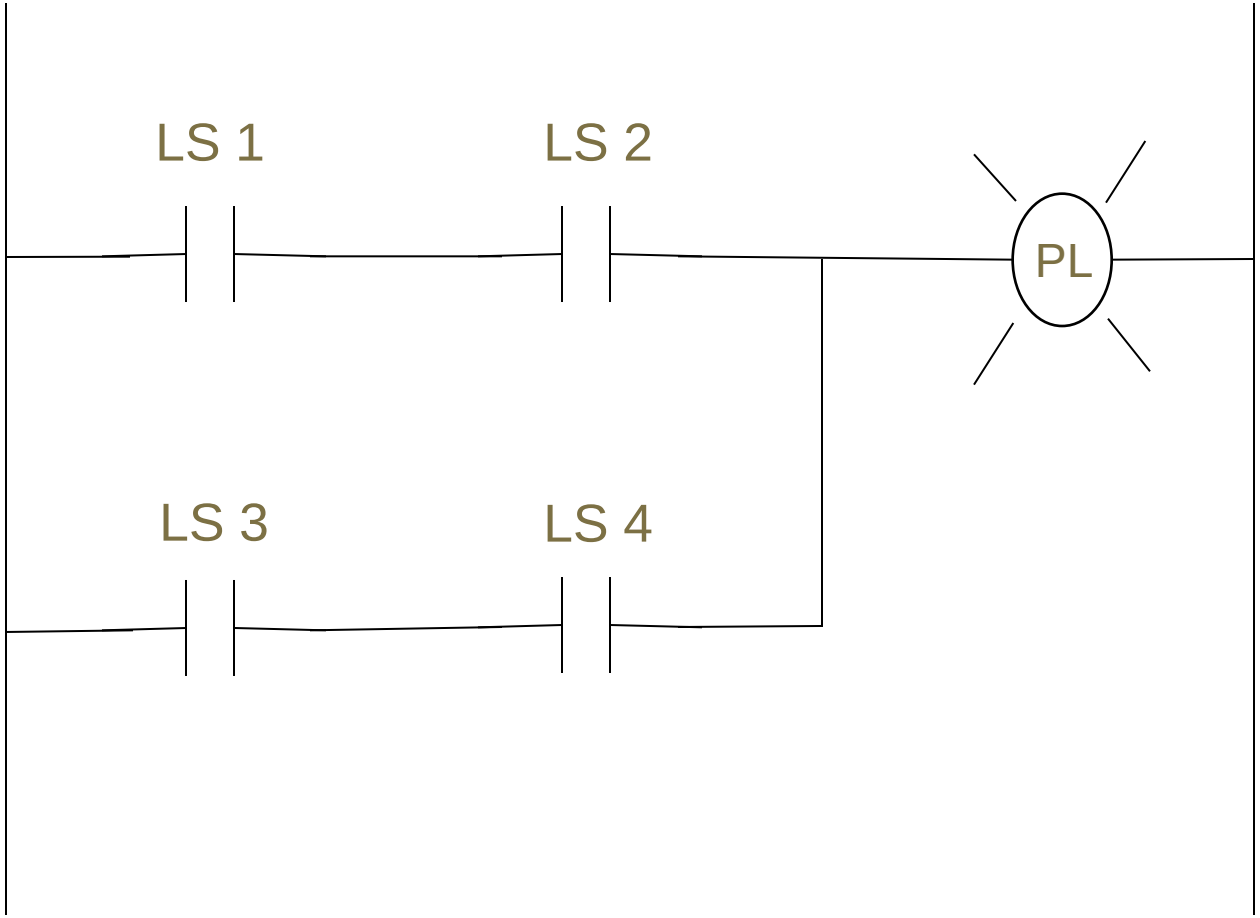


CONVERTING HARDWIRED CIRCUITS

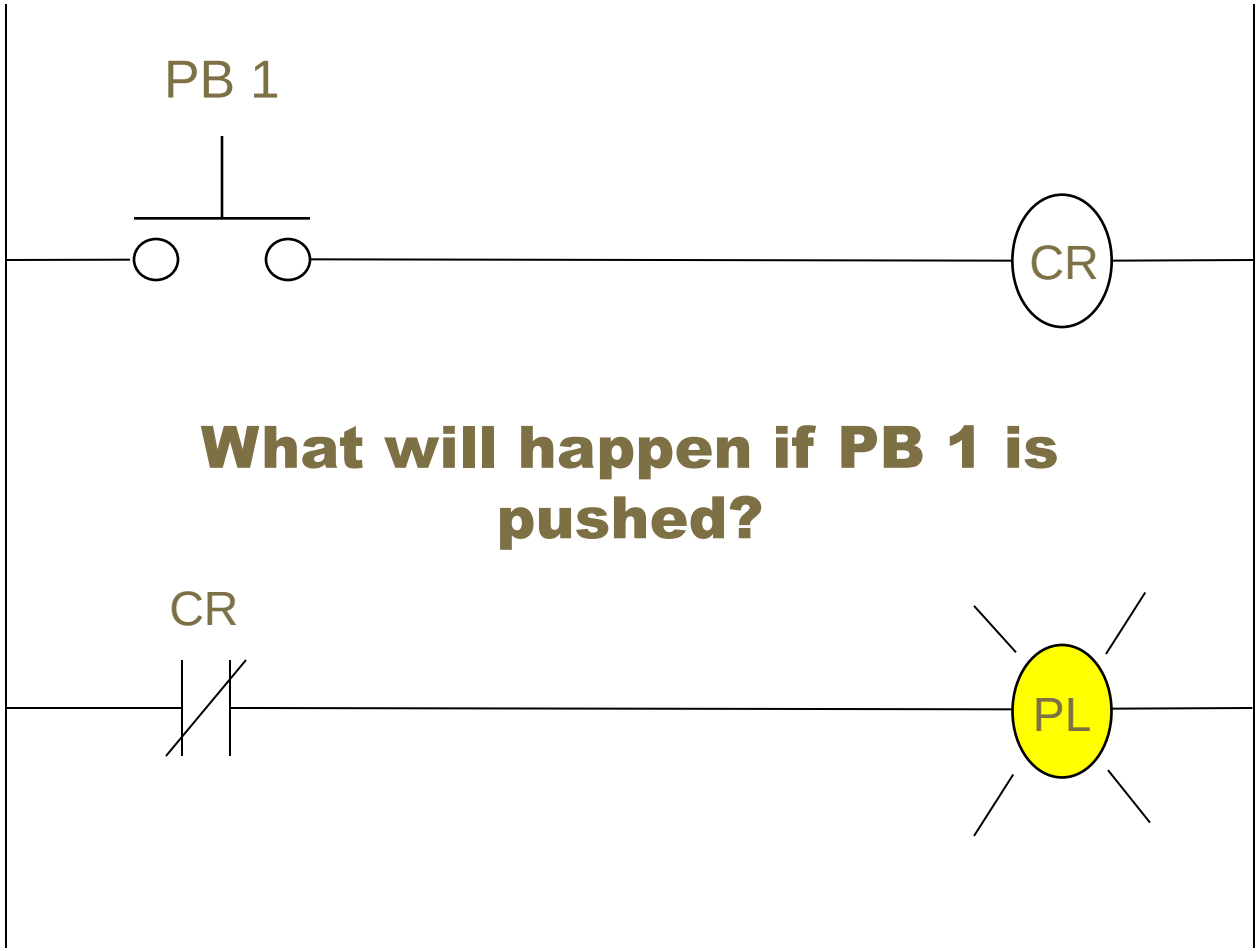


Convert to PLC Ladder Logic

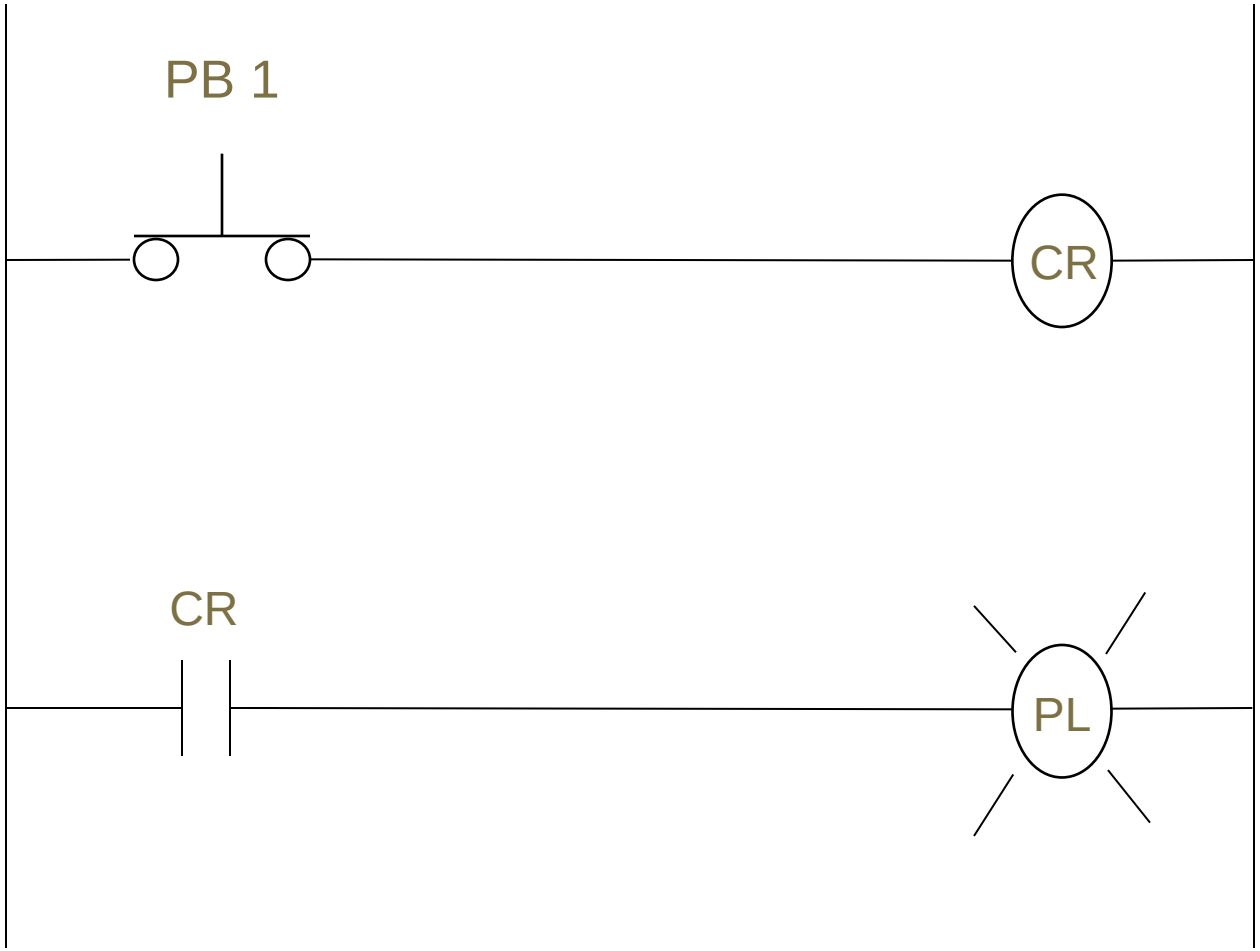
CONVERTING HARDWIRED CIRCUITS



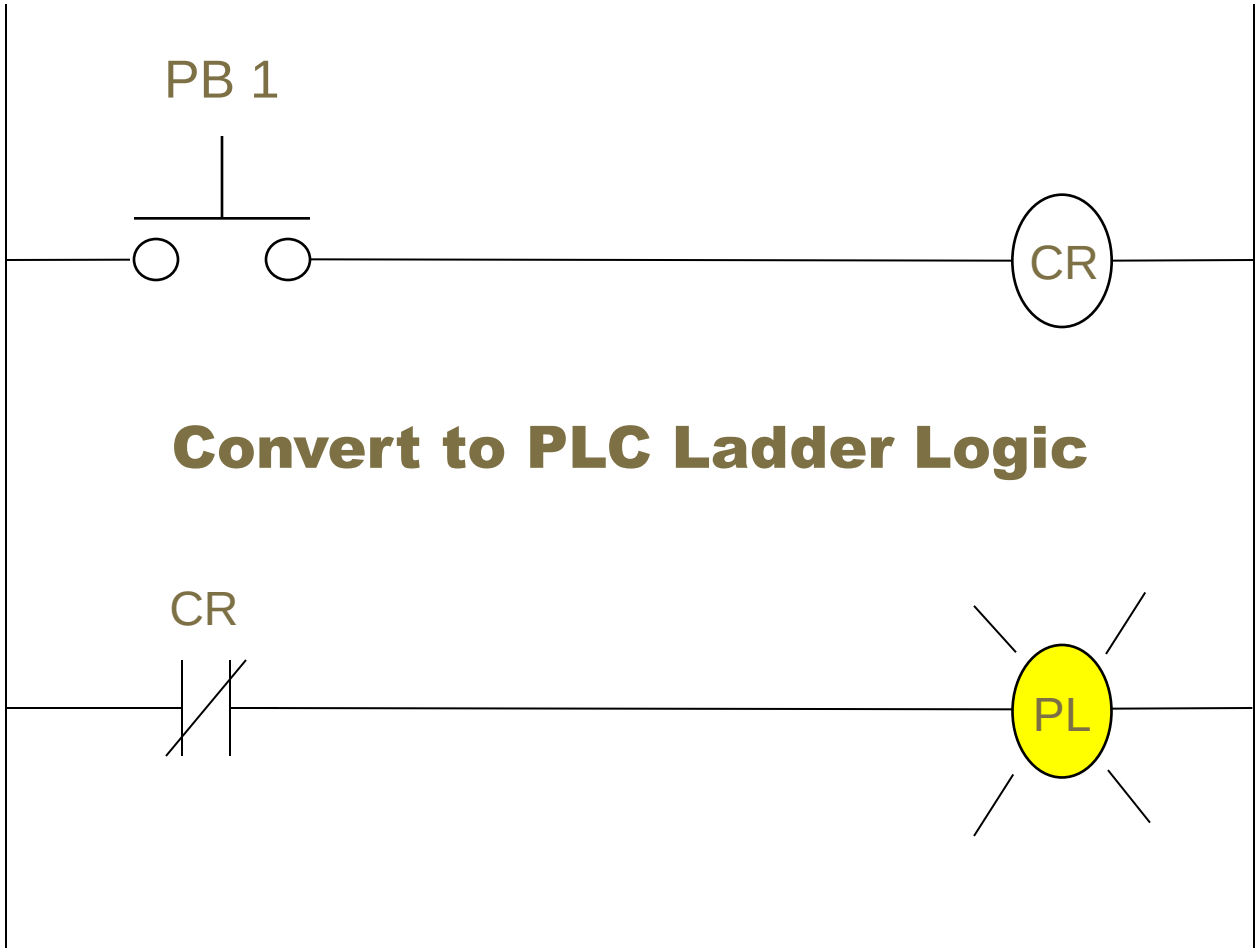
CONVERTING HARDWIRED CIRCUITS



CONVERTING HARDWIRED CIRCUITS

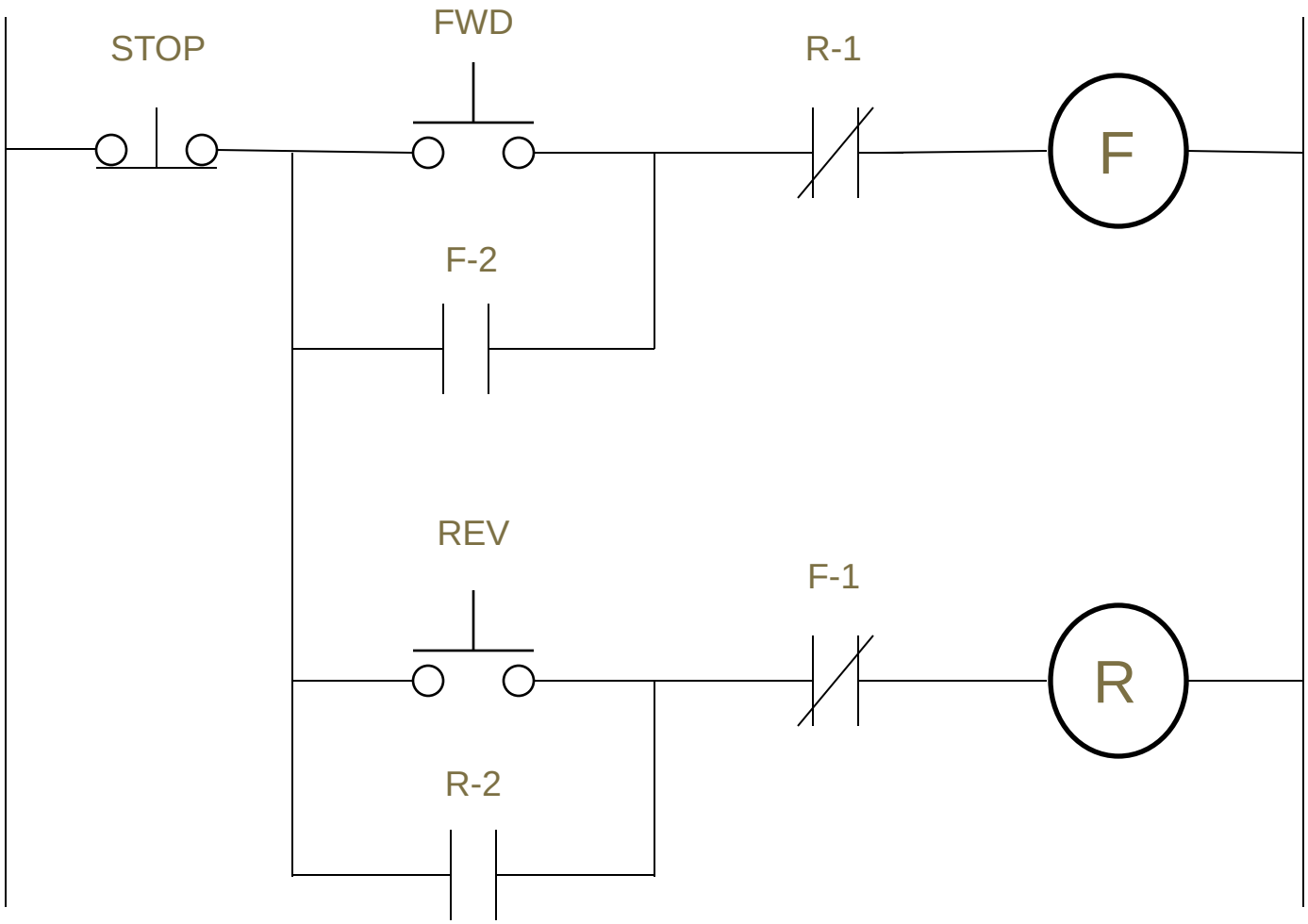


CONVERTING HARDWIRED CIRCUITS



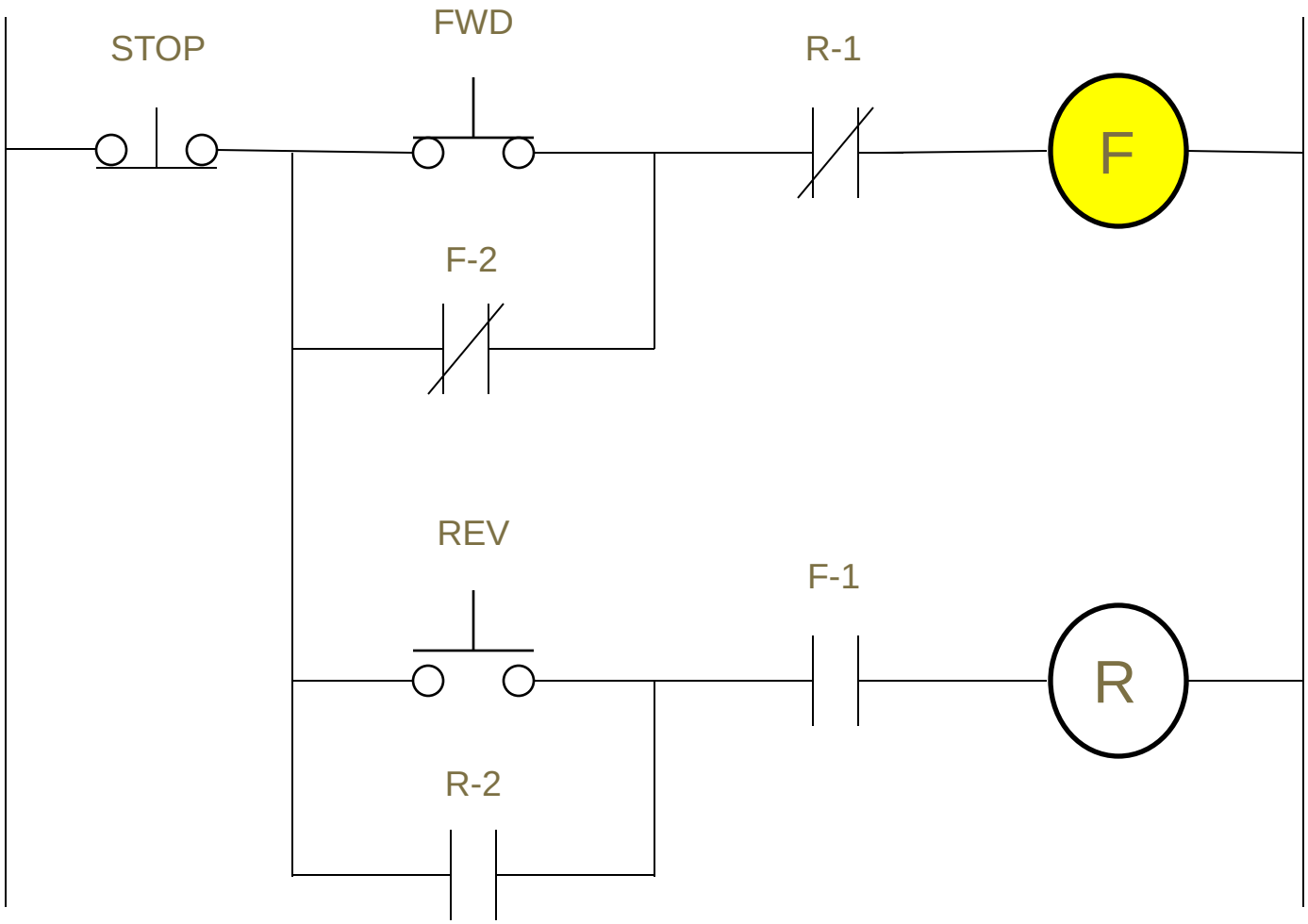
CONVERTING HARDWIRED CIRCUITS

CONVERTING HARDWIRED CIRCUITS



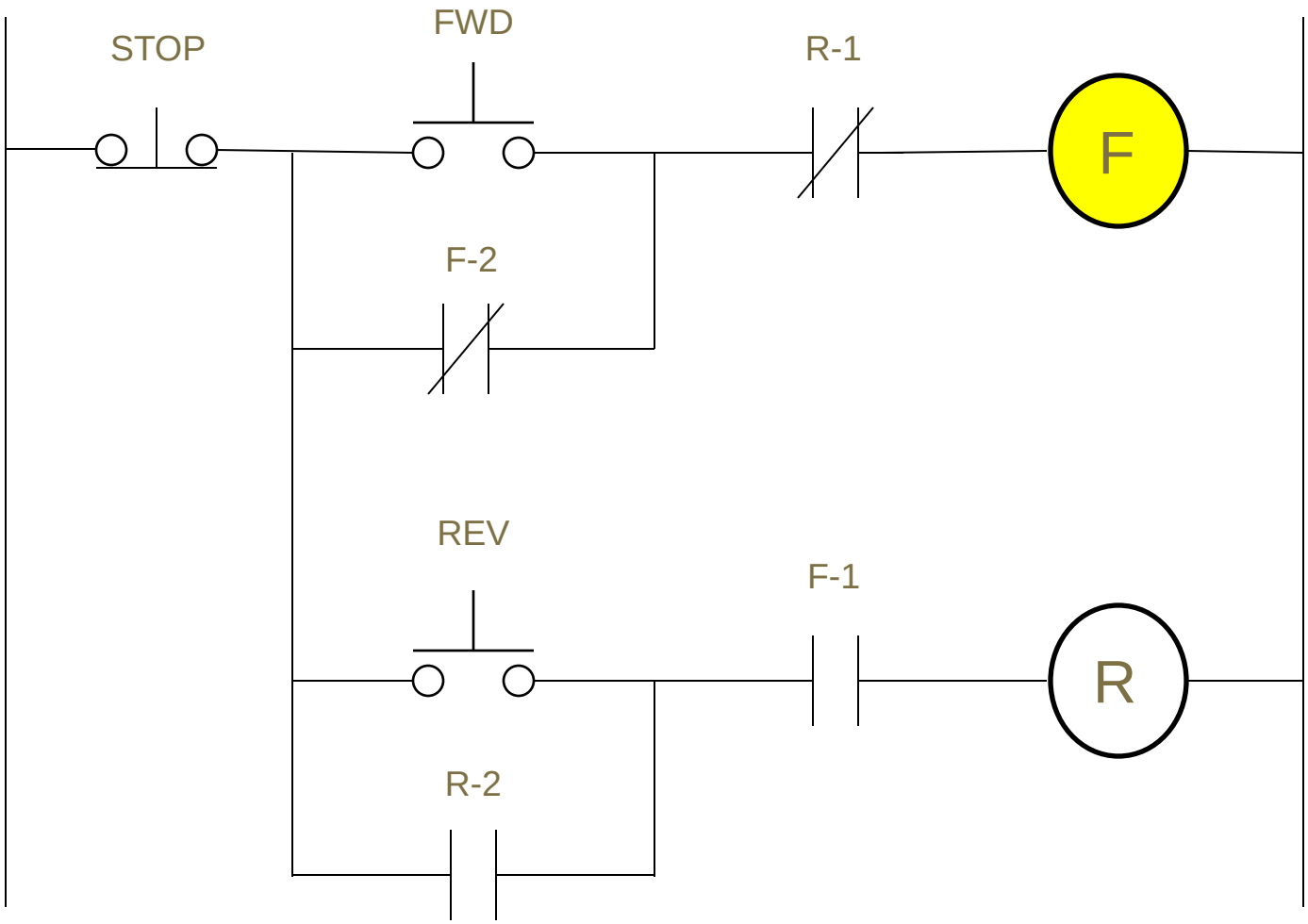
What will happen if FWD is pushed?

CONVERTING HARDWIRED CIRCUITS



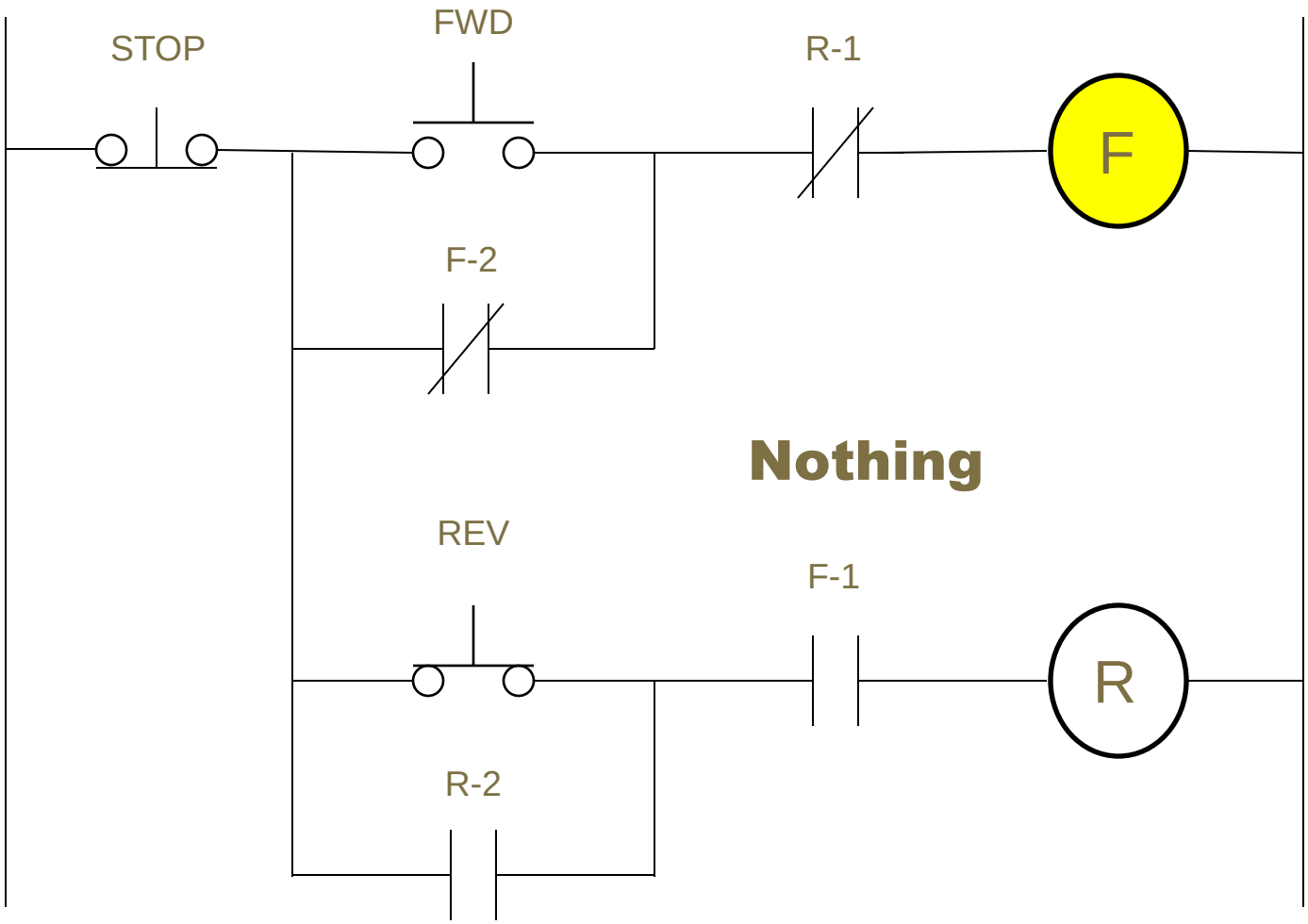
What will happen if FWD is released?

CONVERTING HARDWIRED CIRCUITS

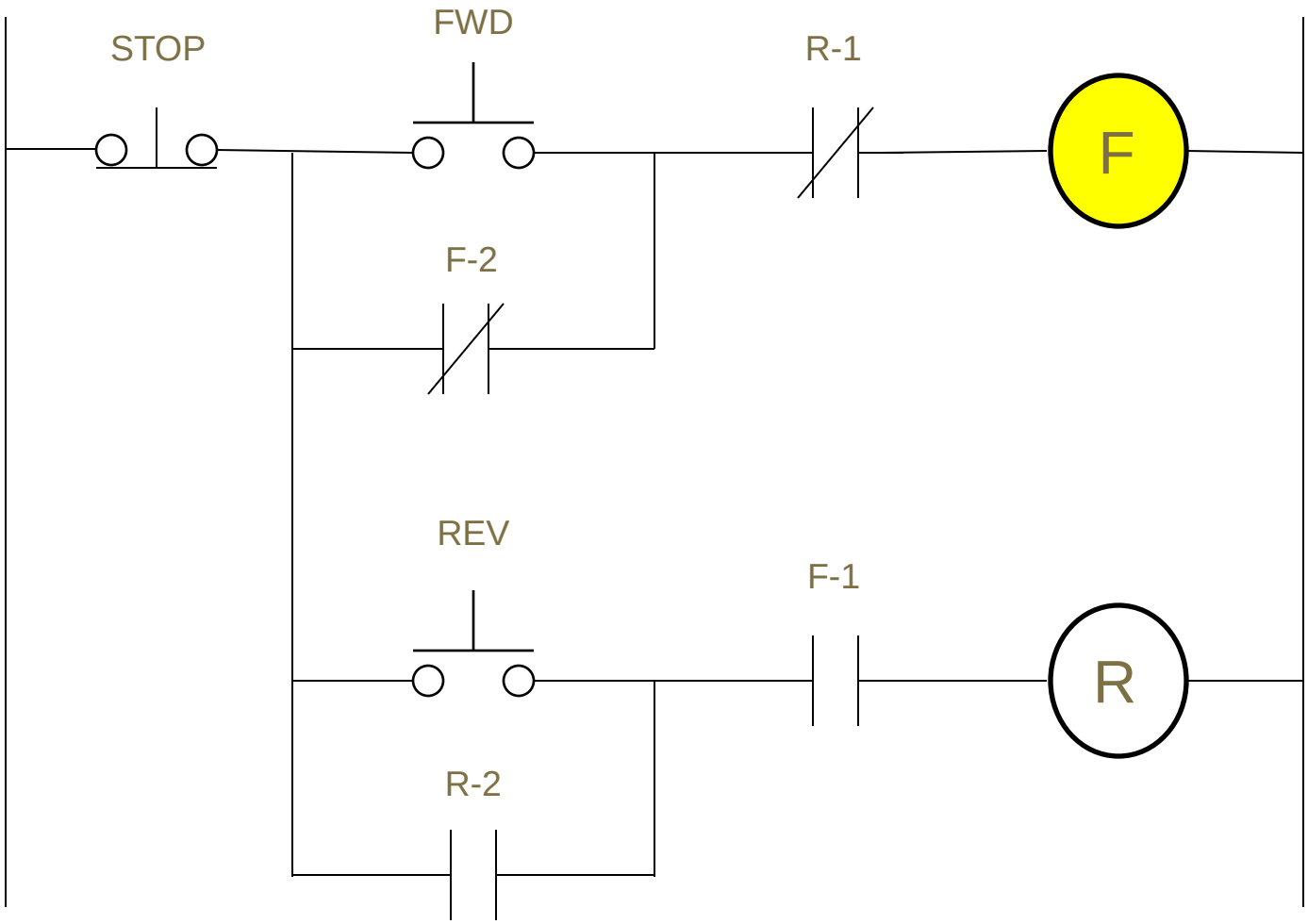


What will happen if REV is pushed?

CONVERTING HARDWIRED CIRCUITS

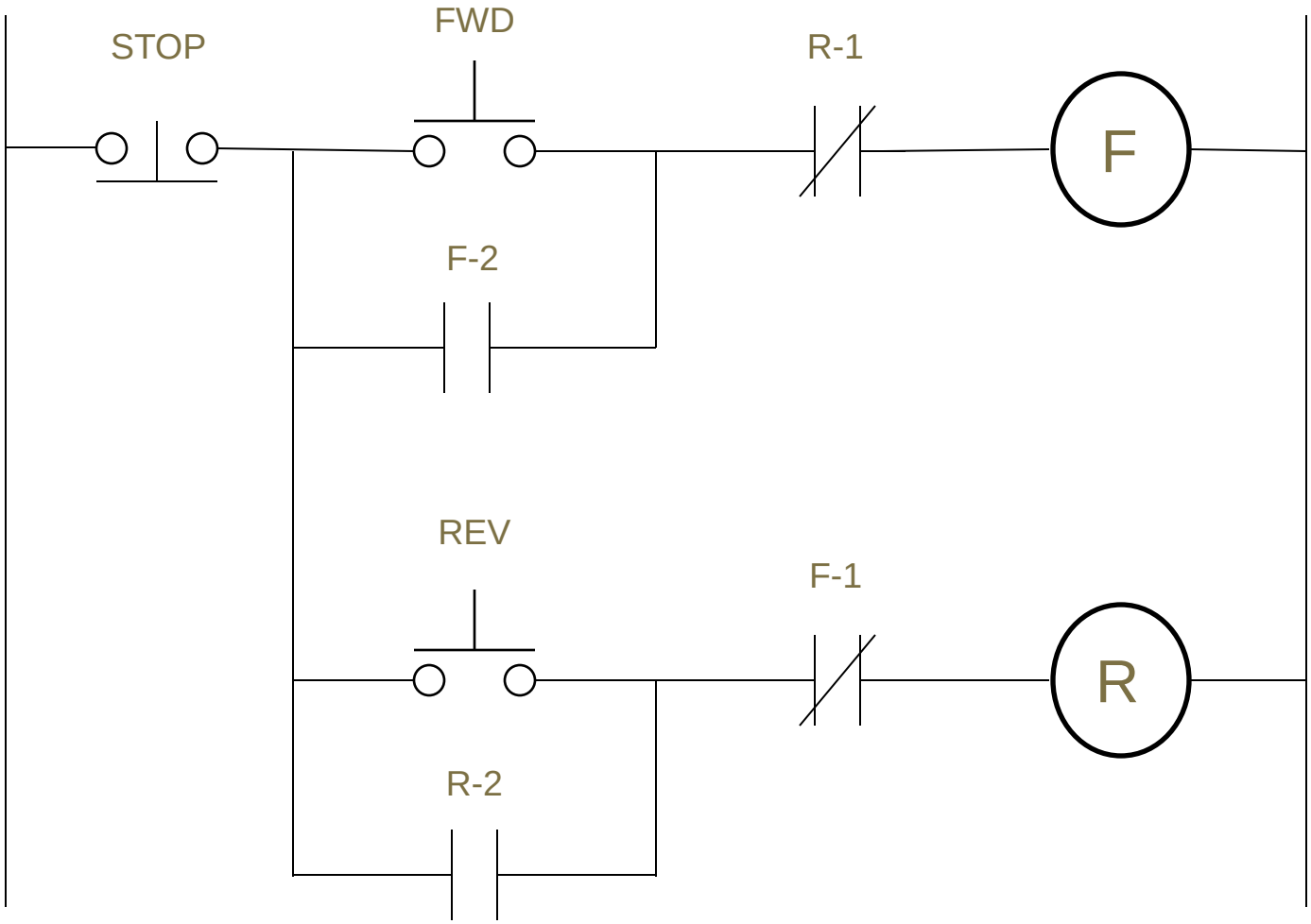


CONVERTING HARDWIRED CIRCUITS

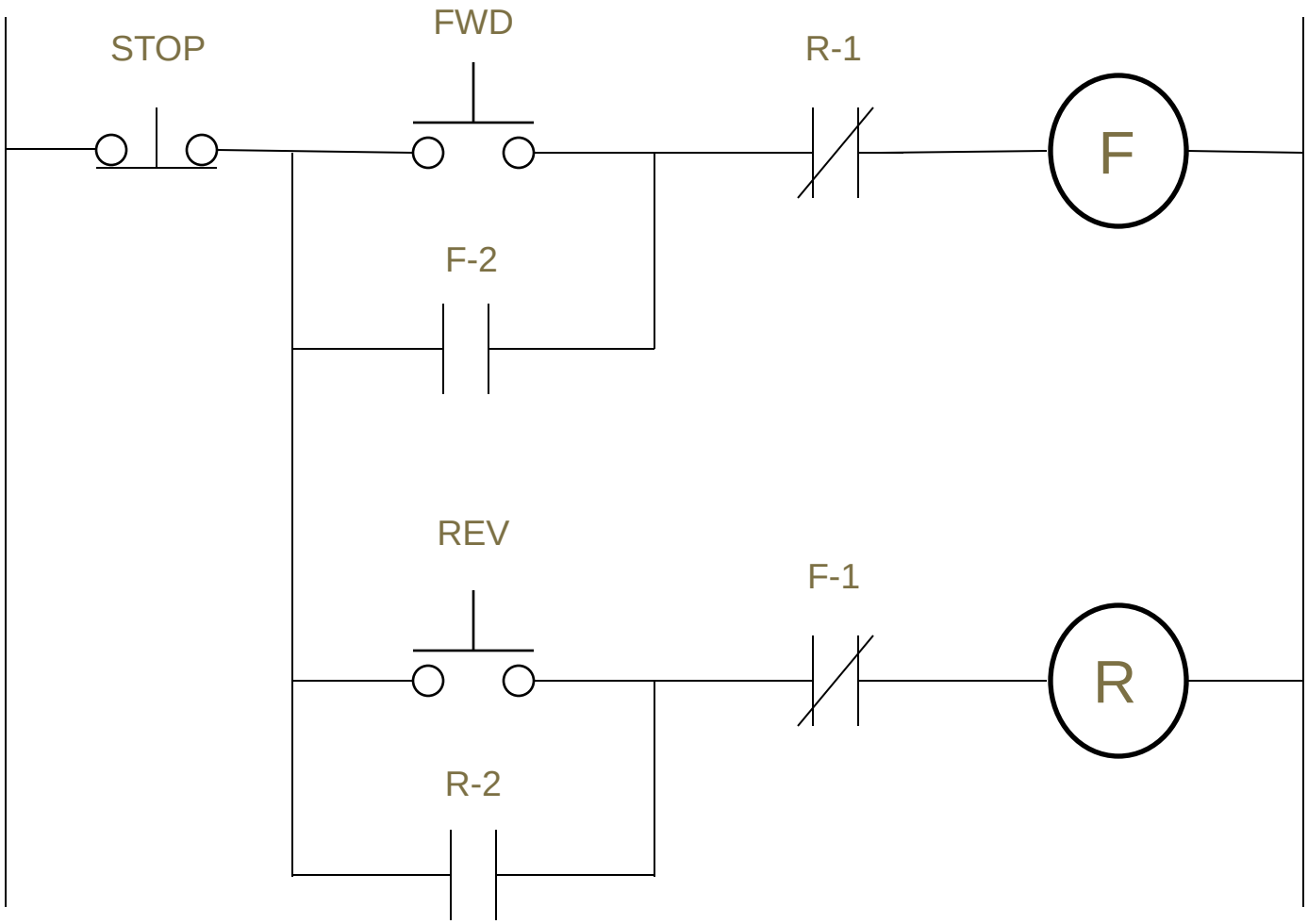


What will happen if Stop is pushed?

CONVERTING HARDWIRED CIRCUITS

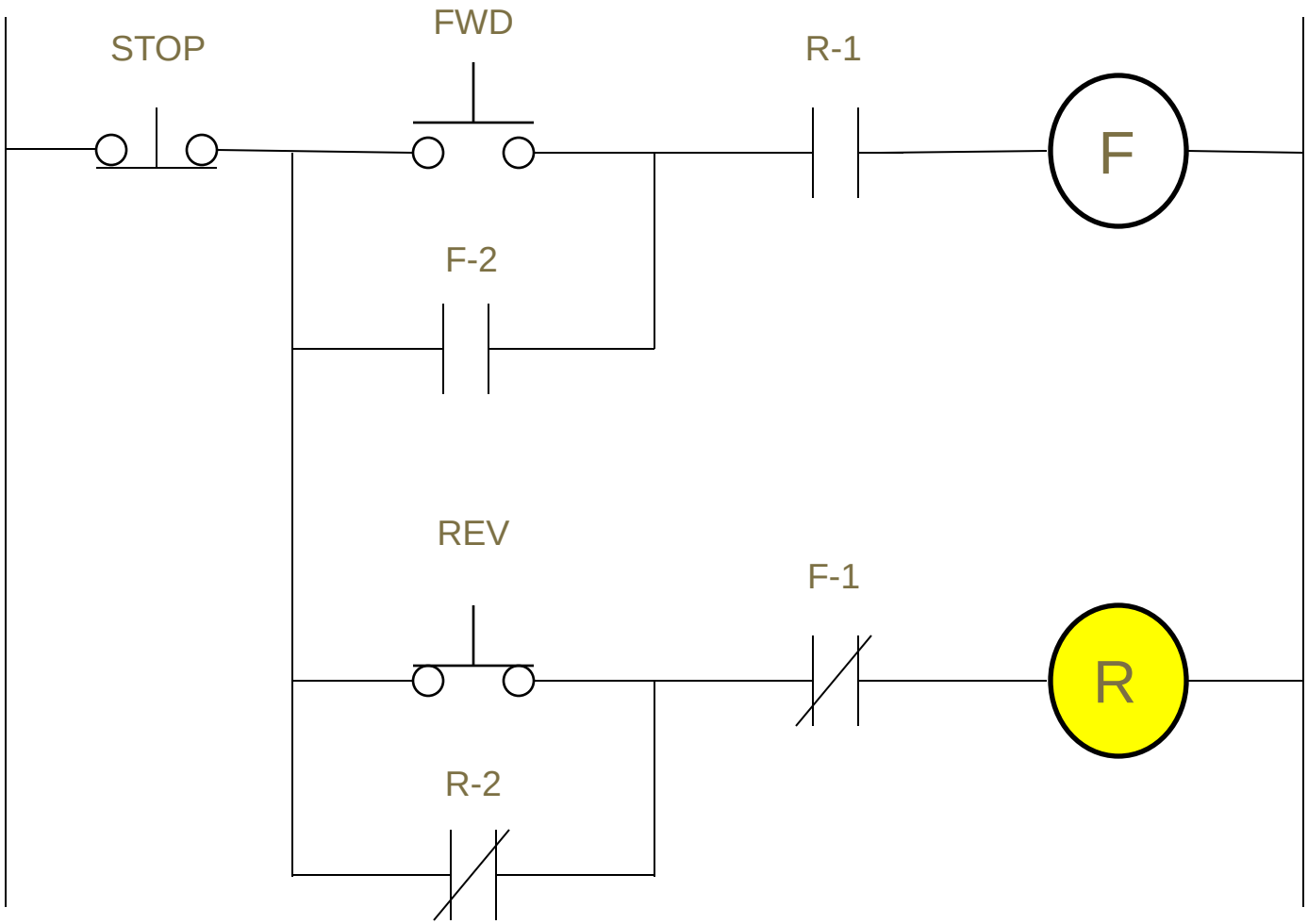


CONVERTING HARDWIRED CIRCUITS



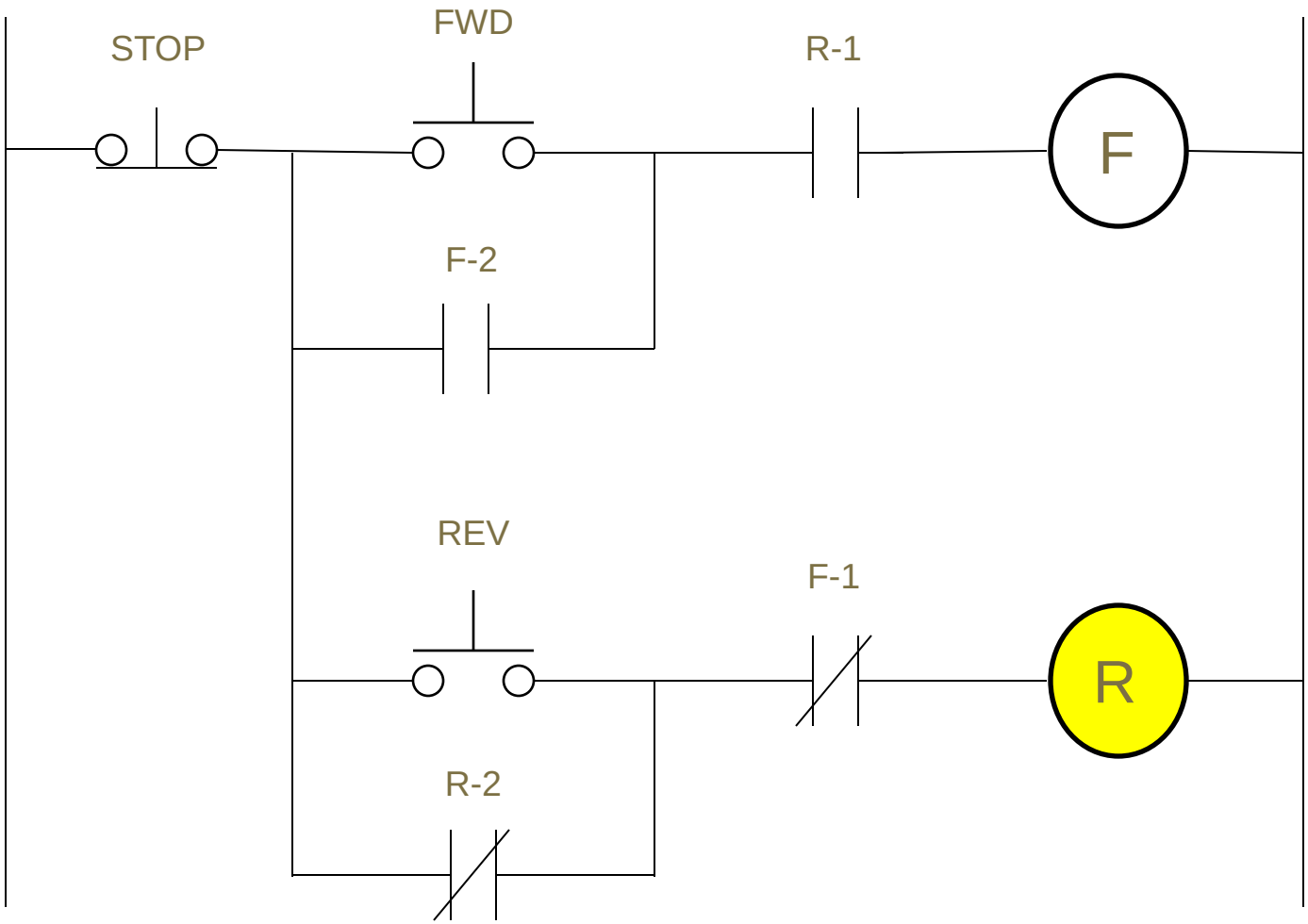
What will happen if REV is pushed?

CONVERTING HARDWIRED CIRCUITS



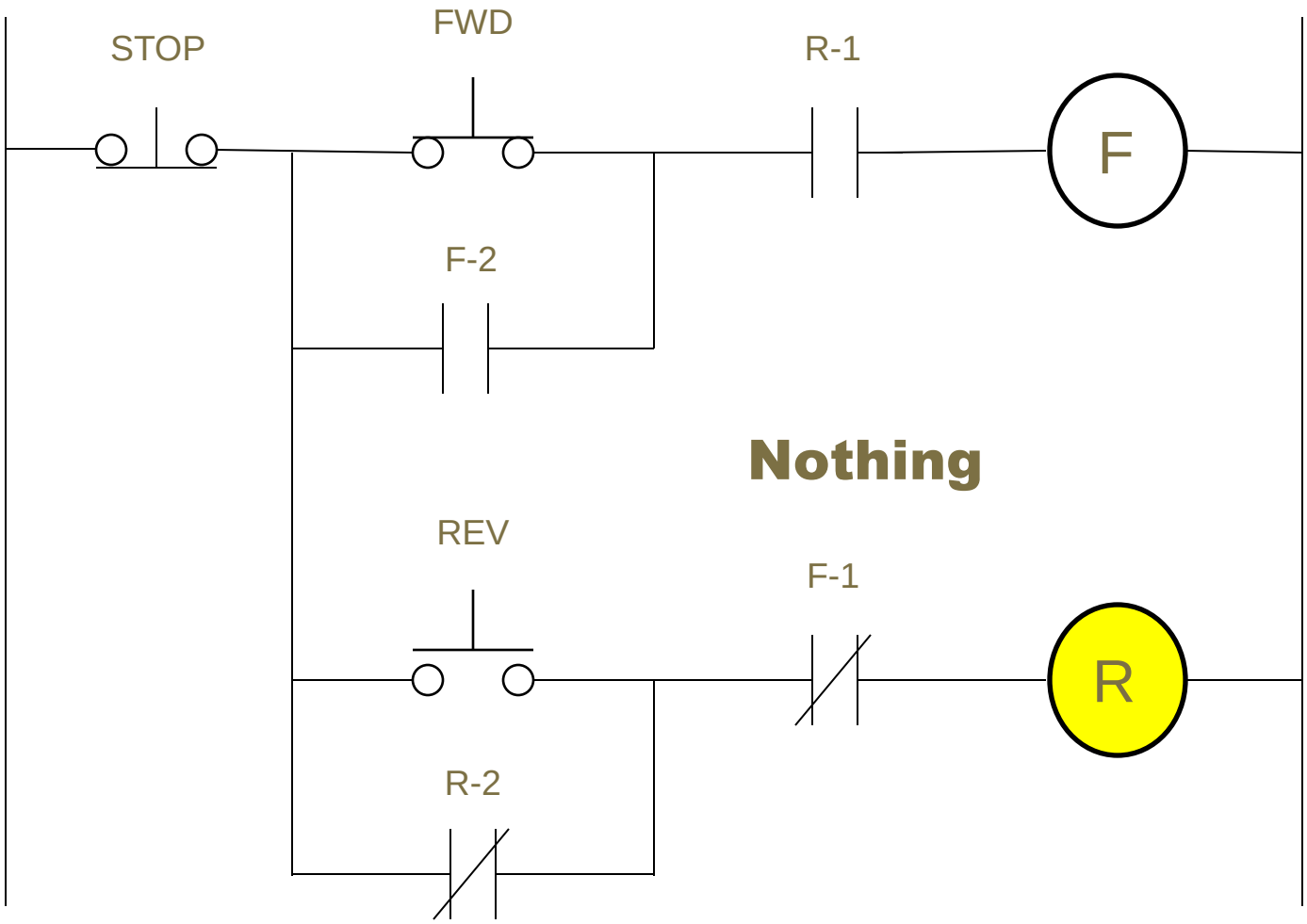
What will happen if REV is released?

CONVERTING HARDWIRED CIRCUITS

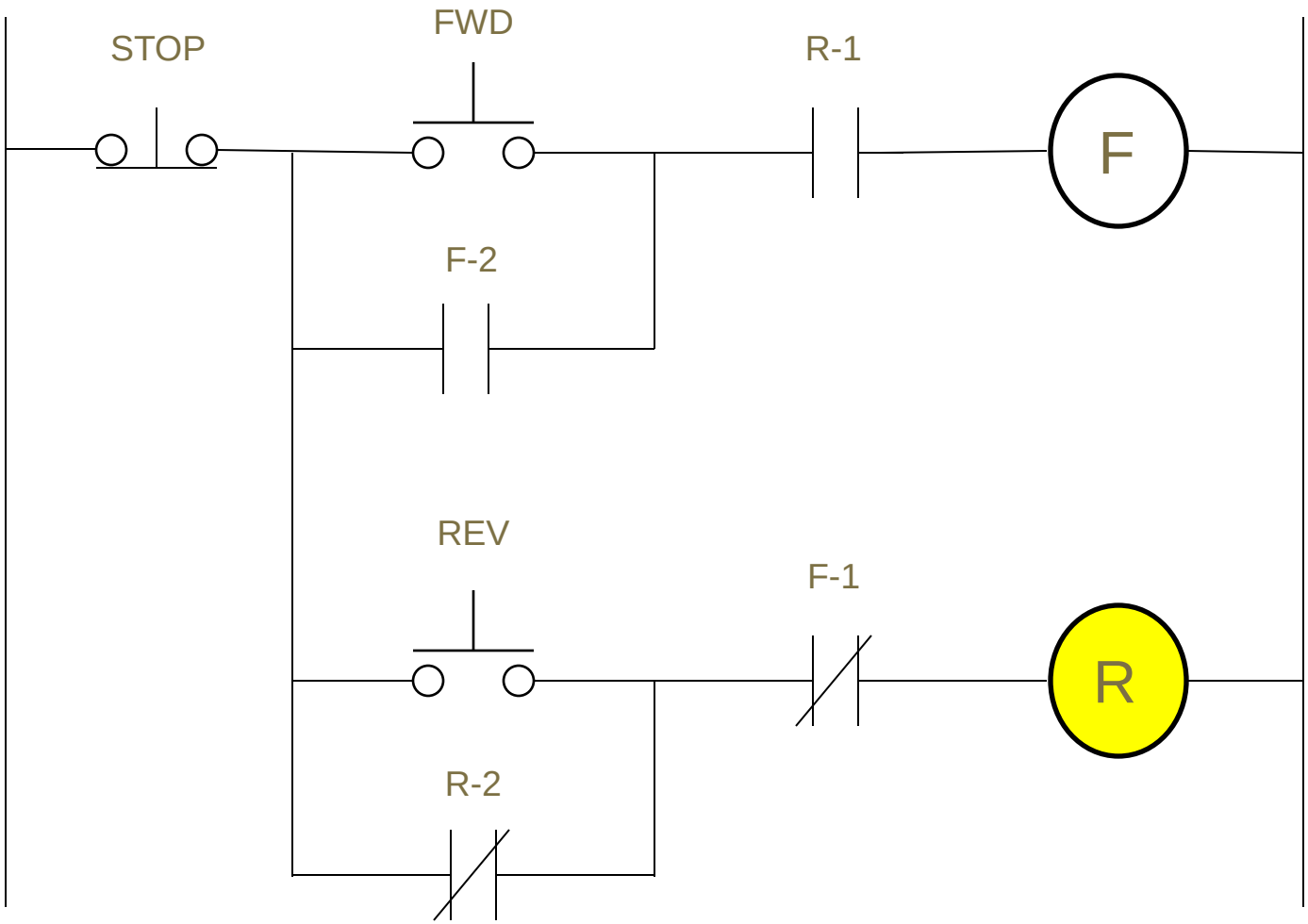


What will happen if FWD is pushed?

CONVERTING HARDWIRED CIRCUITS

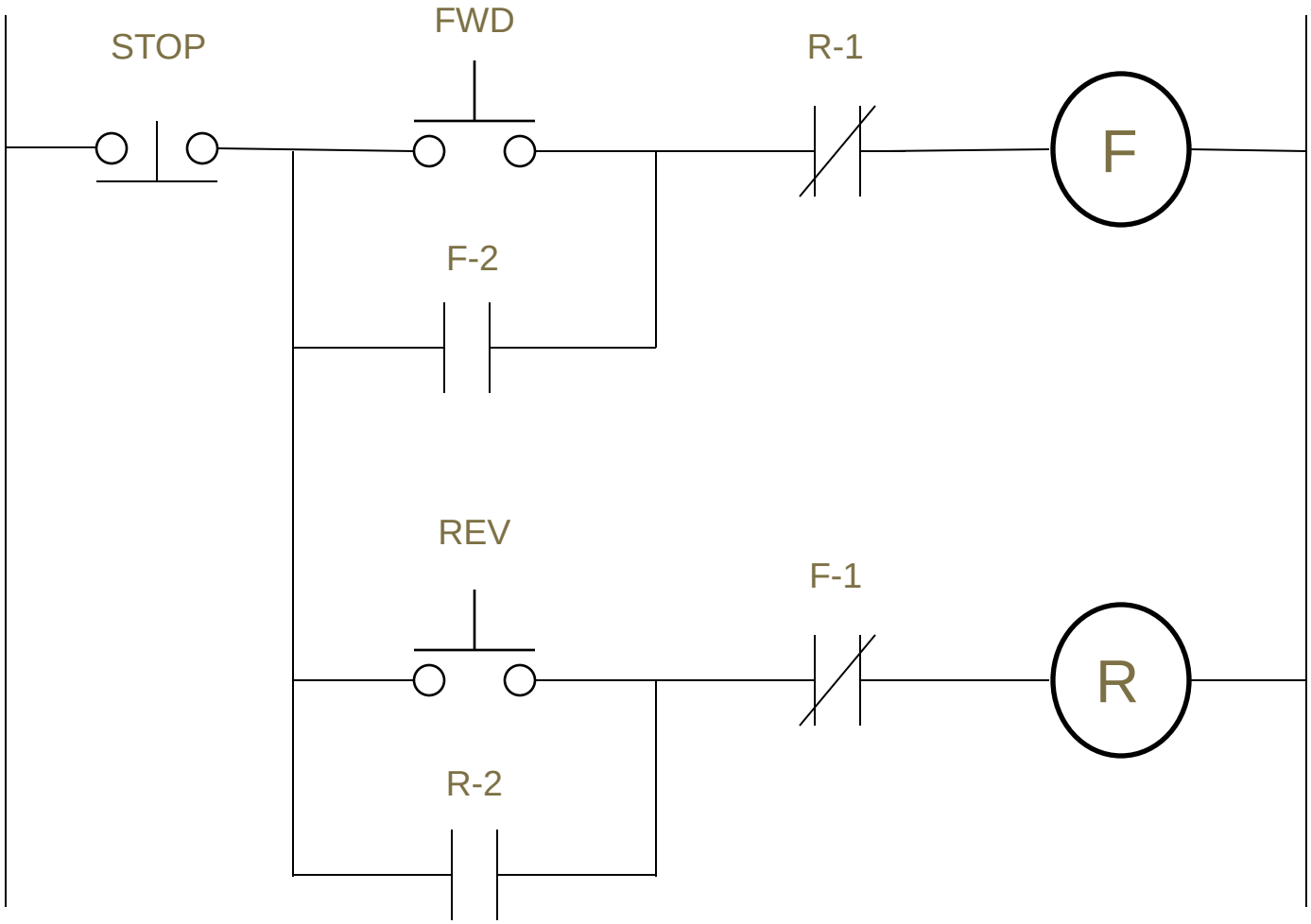


CONVERTING HARDWIRED CIRCUITS

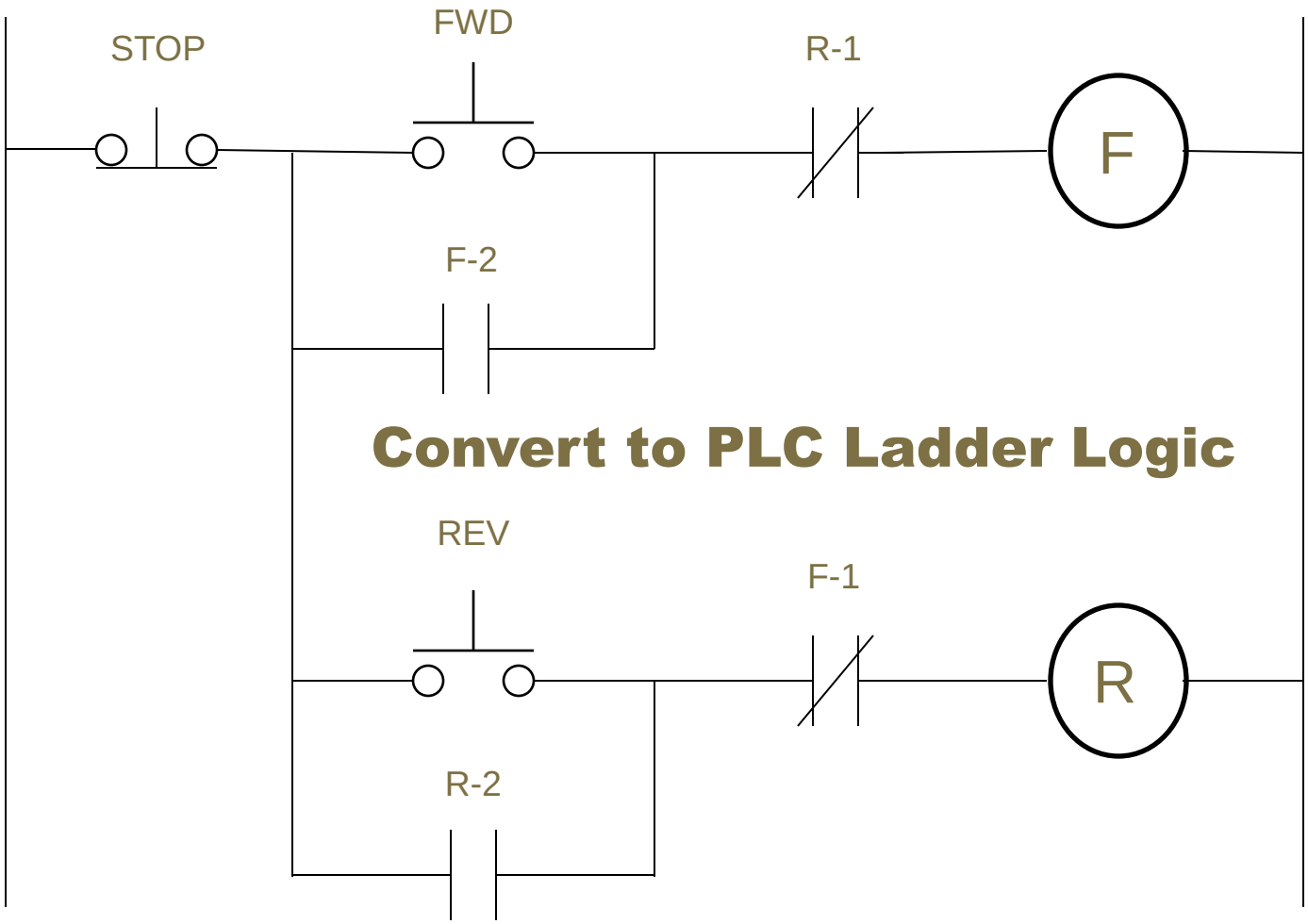


What will happen if Stop is pushed?

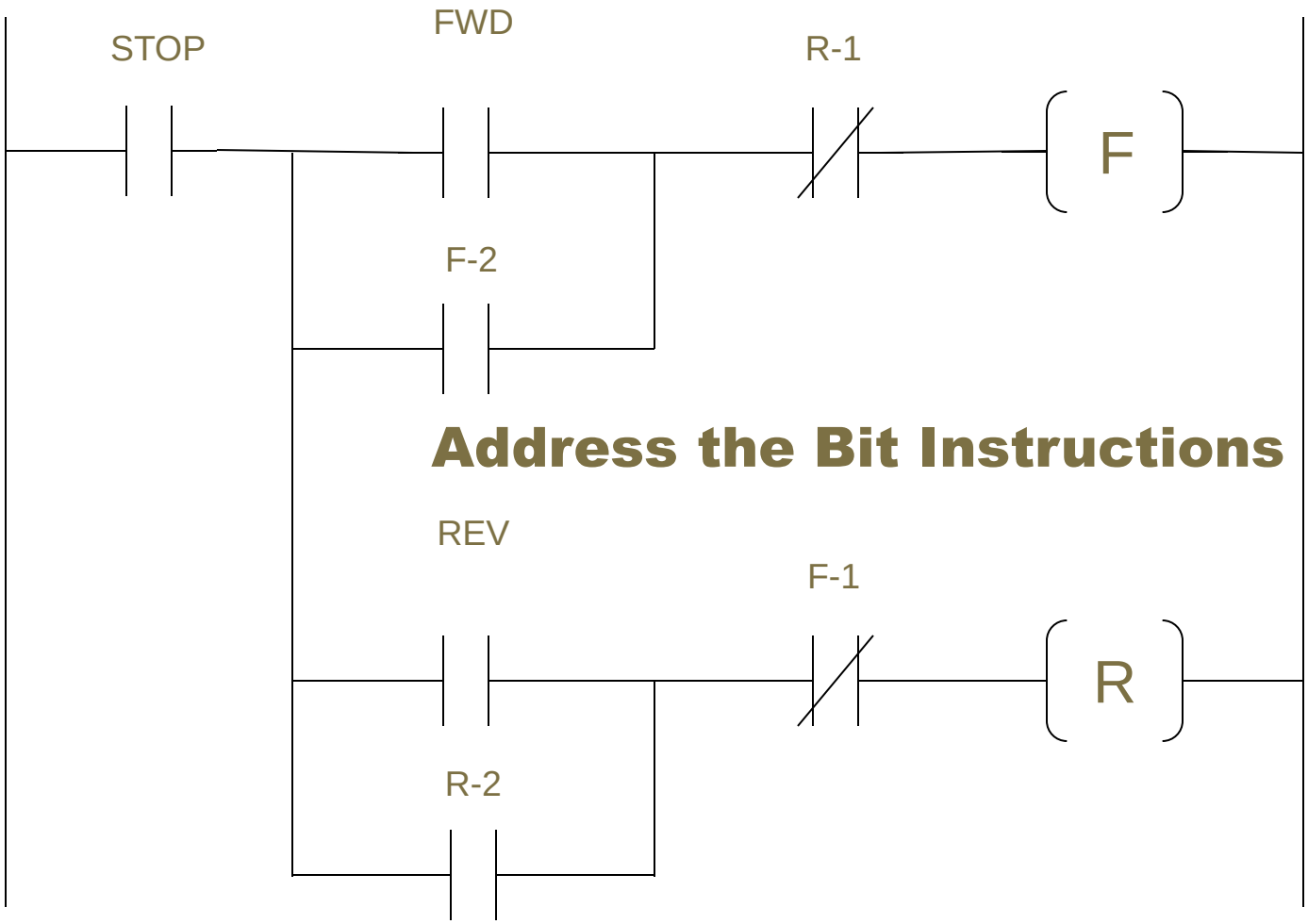
CONVERTING HARDWIRED CIRCUITS



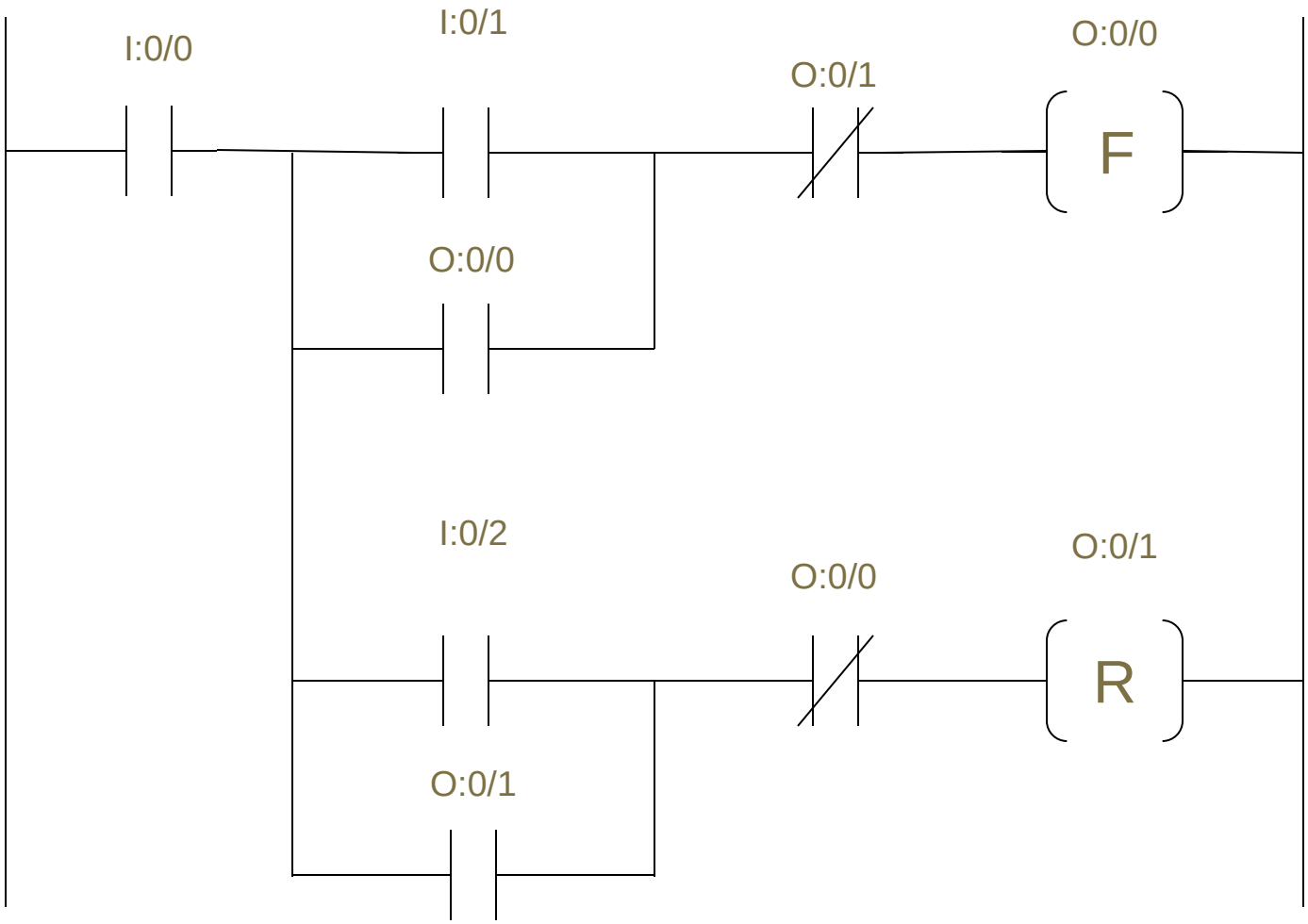
CONVERTING HARDWIRED CIRCUITS



CONVERTING HARDWIRED CIRCUITS

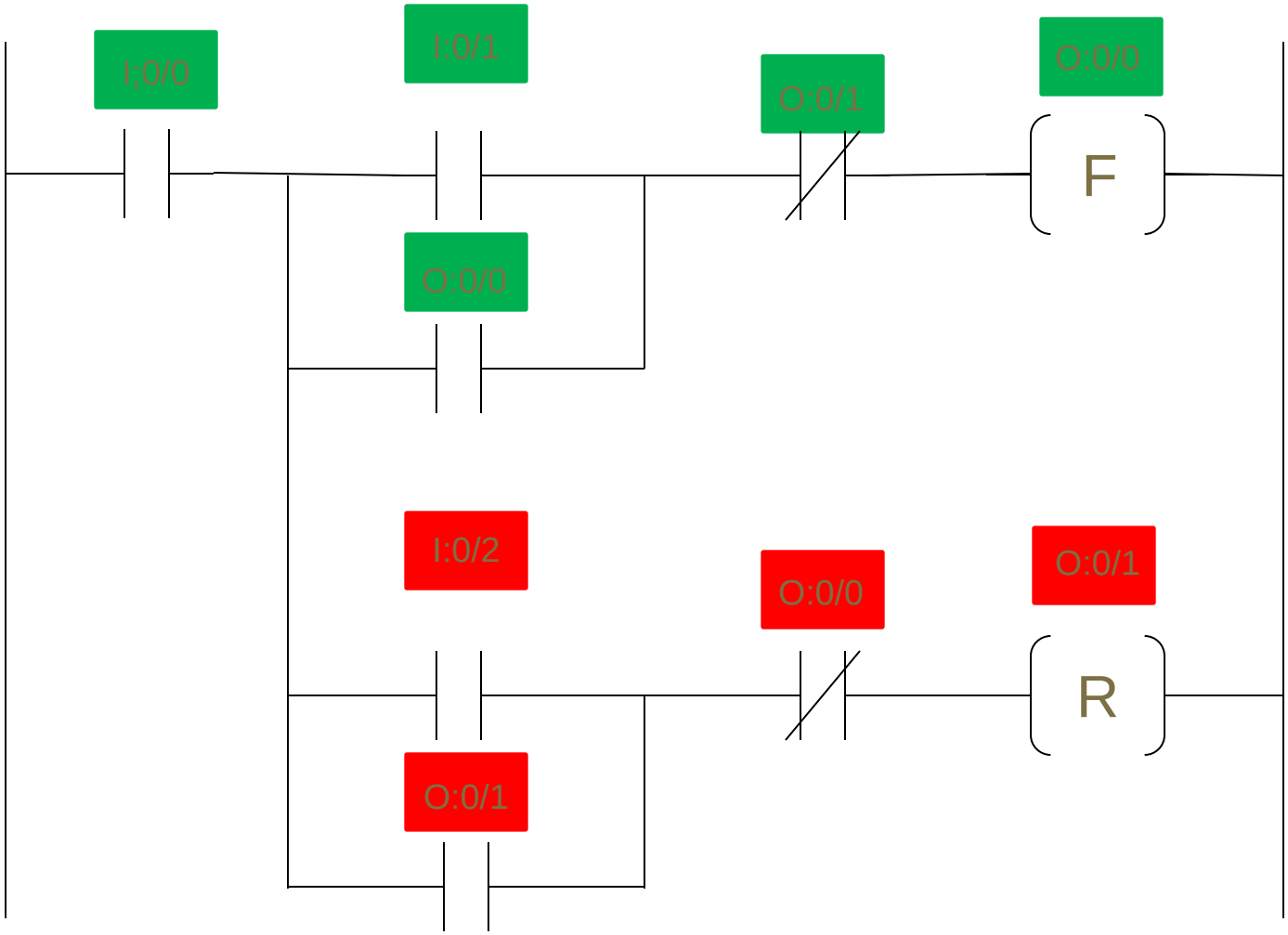


CONVERTING HARDWIRED CIRCUITS



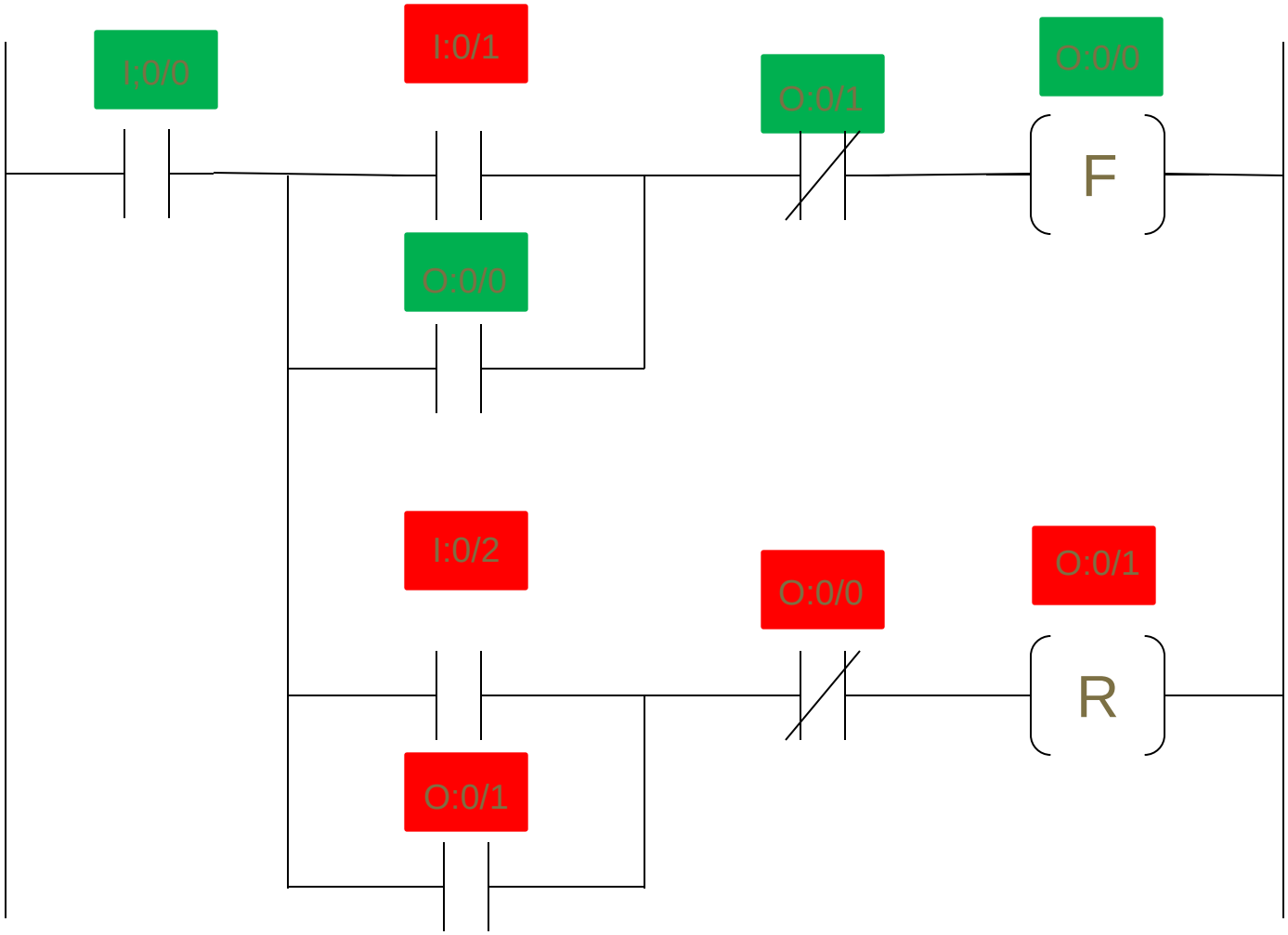
What will happen if 1:0/1 is pushed?

CONVERTING HARDWIRED CIRCUITS



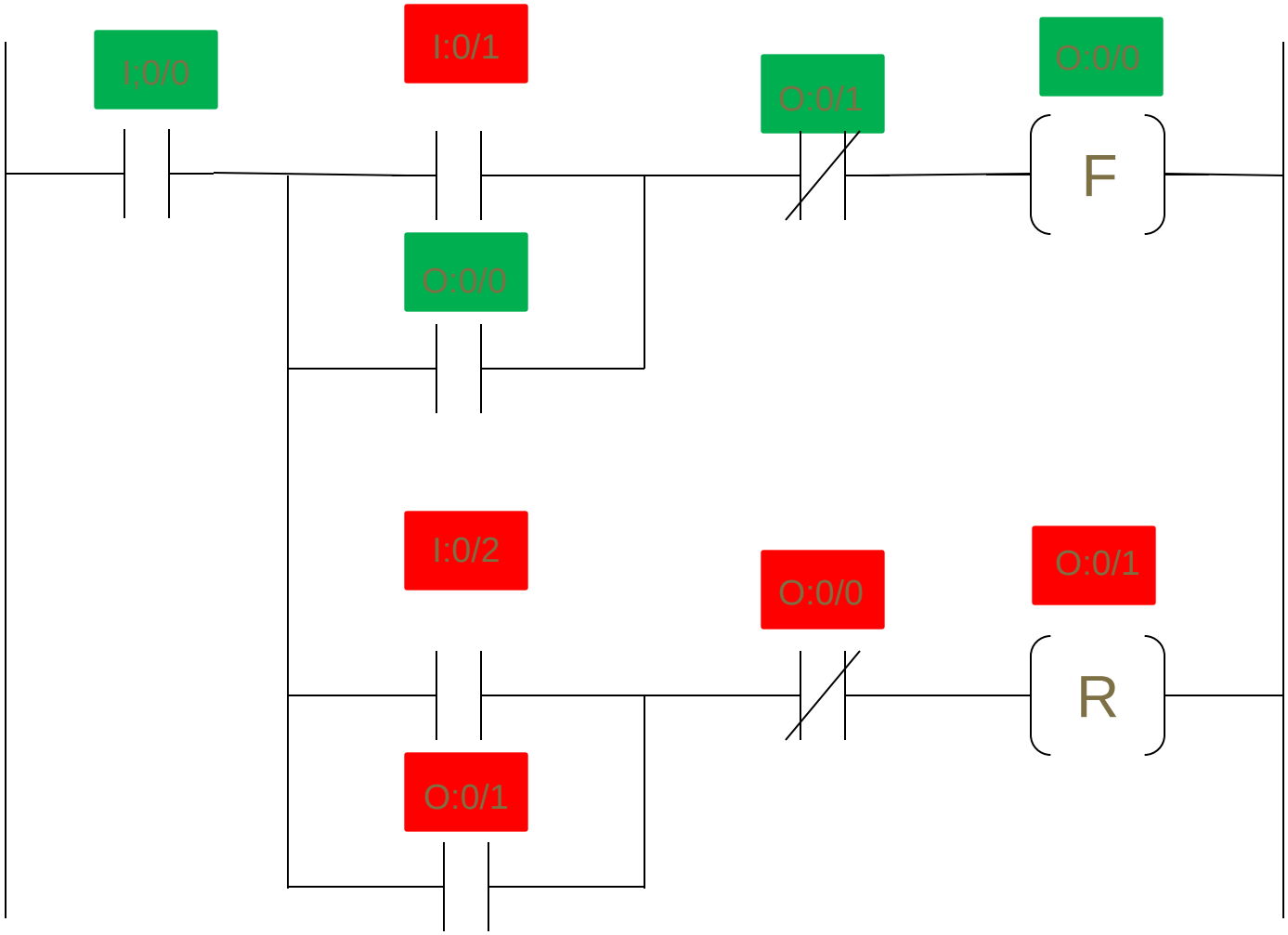
What will happen if 1:0/1 is released?

CONVERTING HARDWIRED CIRCUITS



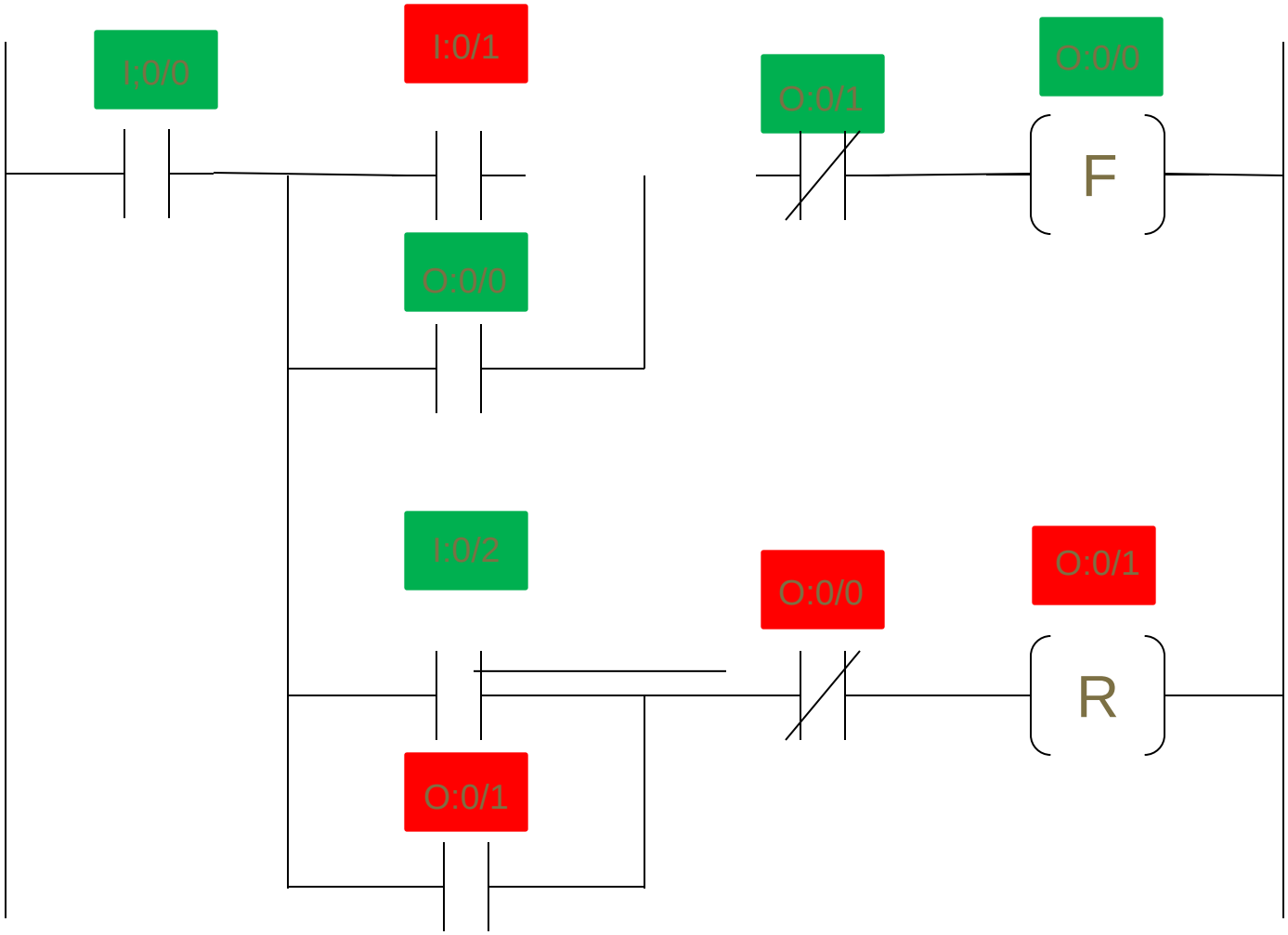
1:0/1 goes false, but O:0/0 remains on

CONVERTING HARDWIRED CIRCUITS



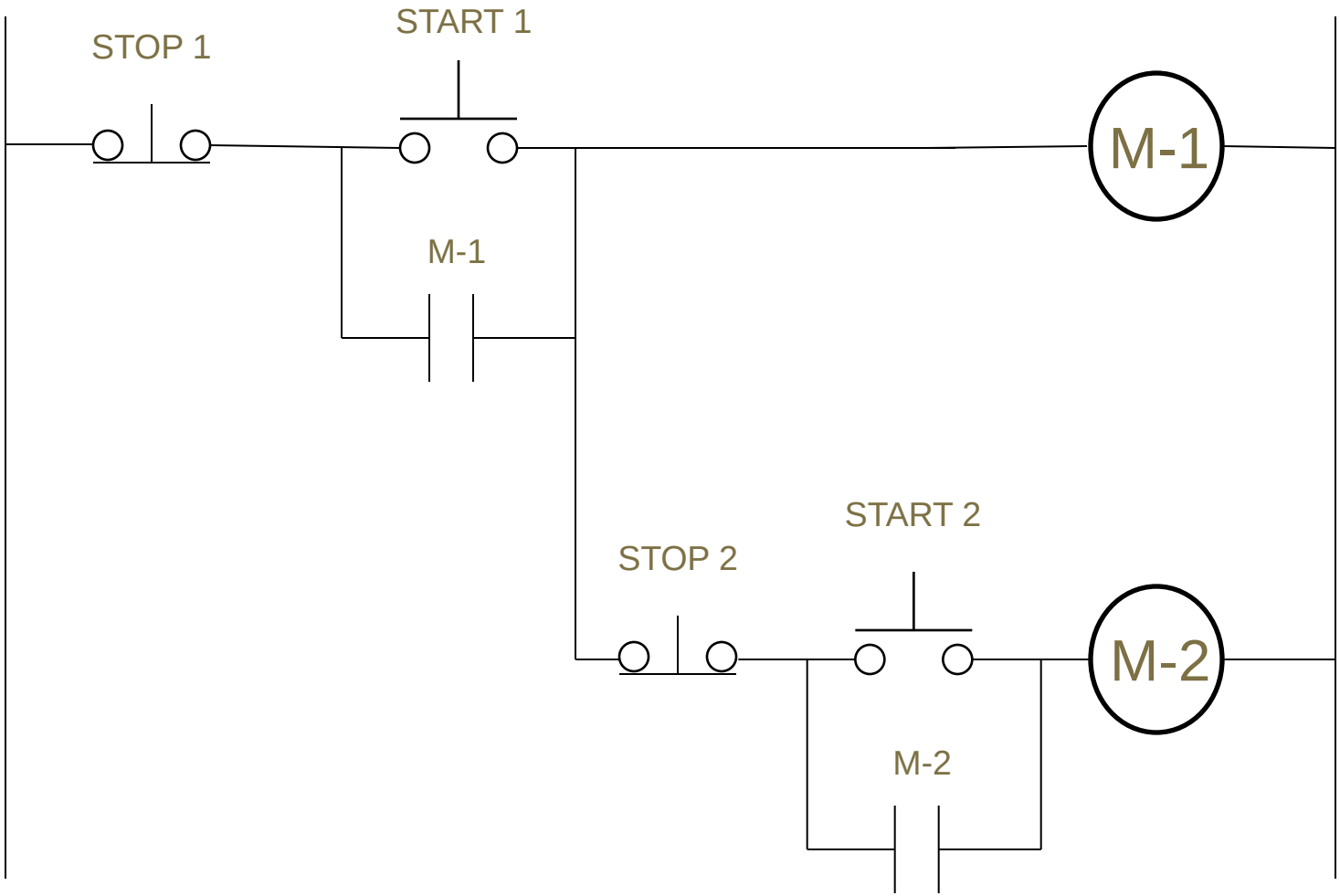
What will happen if 1:0/2 is pushed?

CONVERTING HARDWIRED CIRCUITS



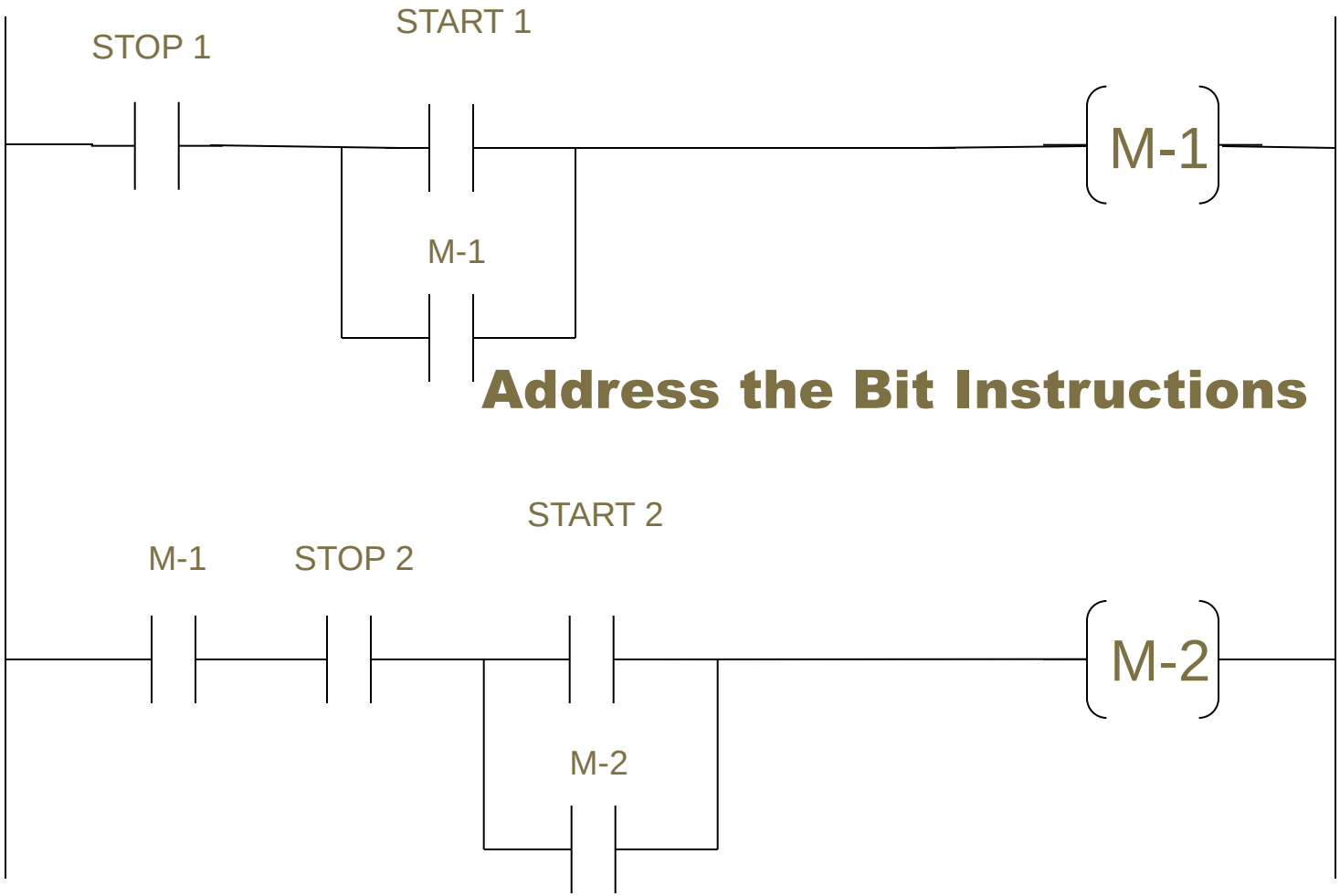
1:0/2 is true but O:0/1 remains off

CONVERTING HARDWIRED CIRCUITS

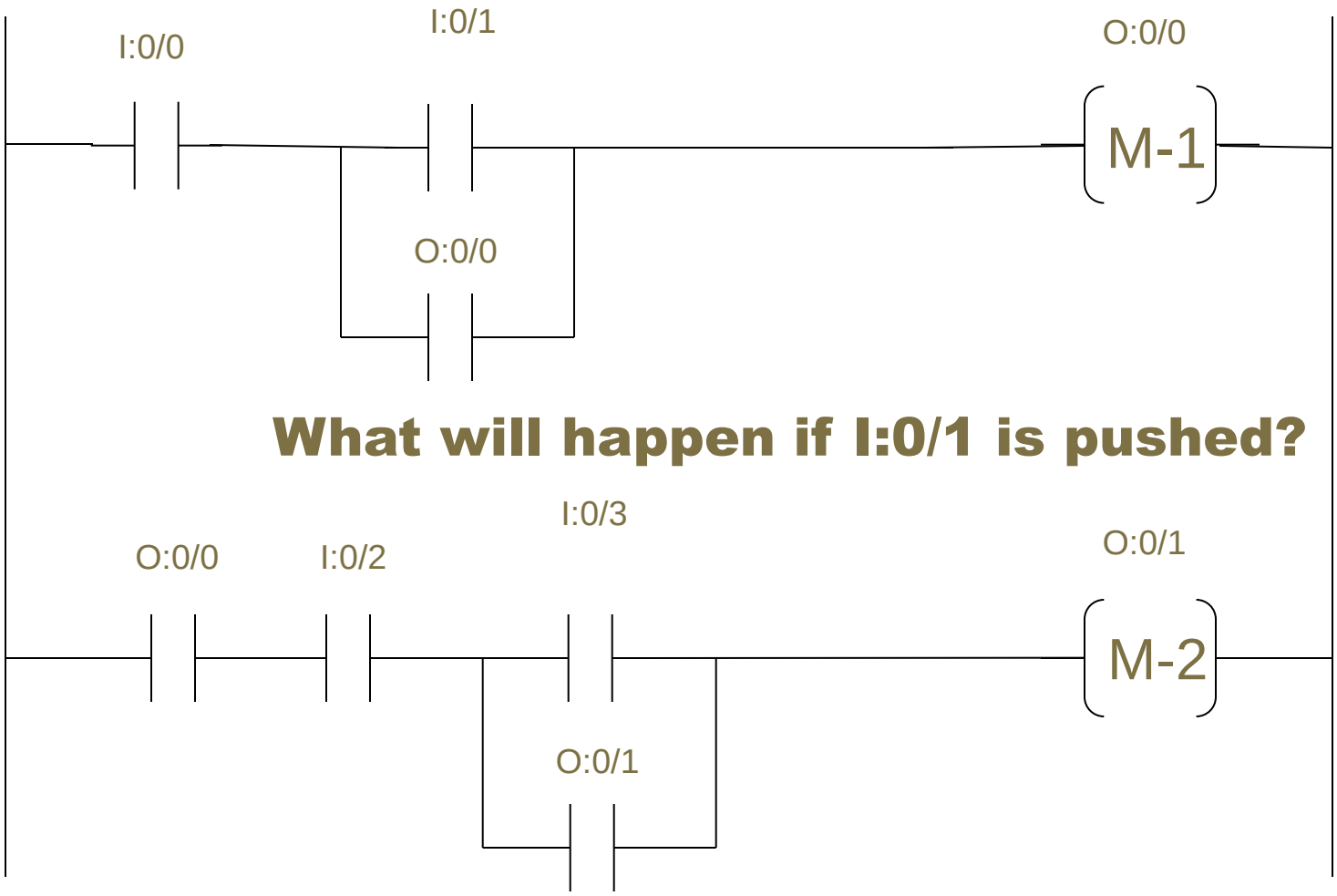


Convert to PLC Ladder Logic

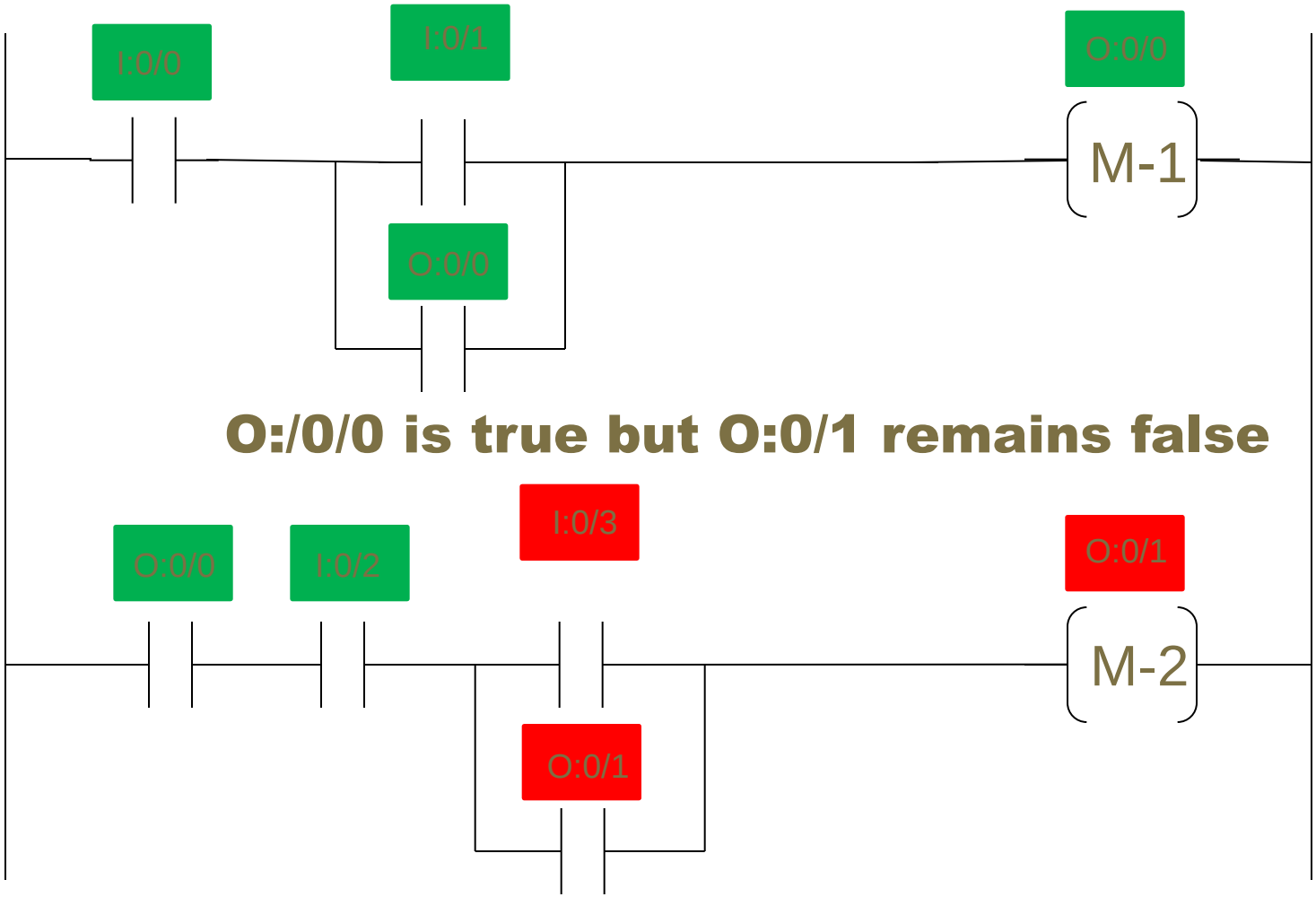
CONVERTING HARDWIRED CIRCUITS



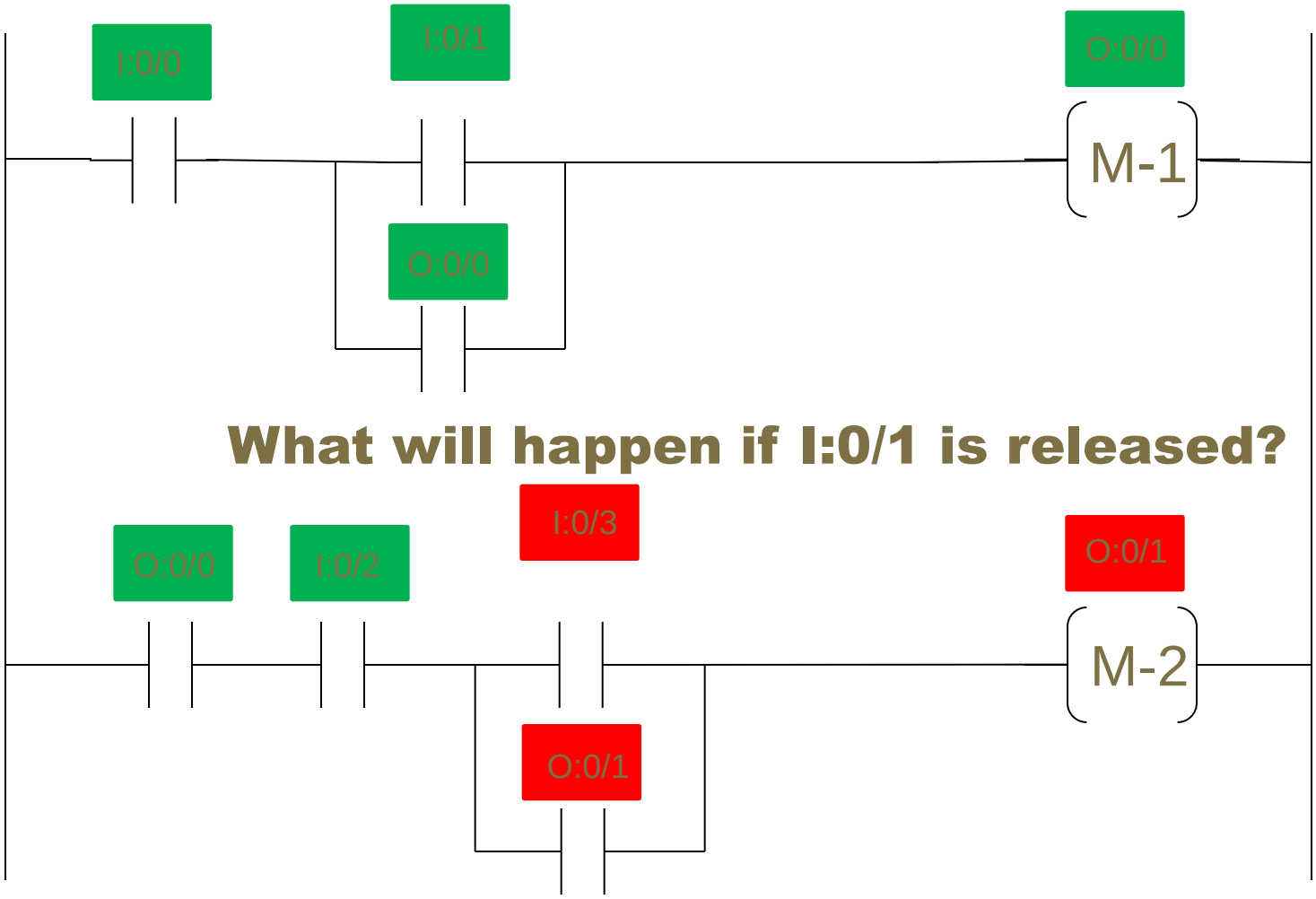
CONVERTING HARDWIRED CIRCUITS



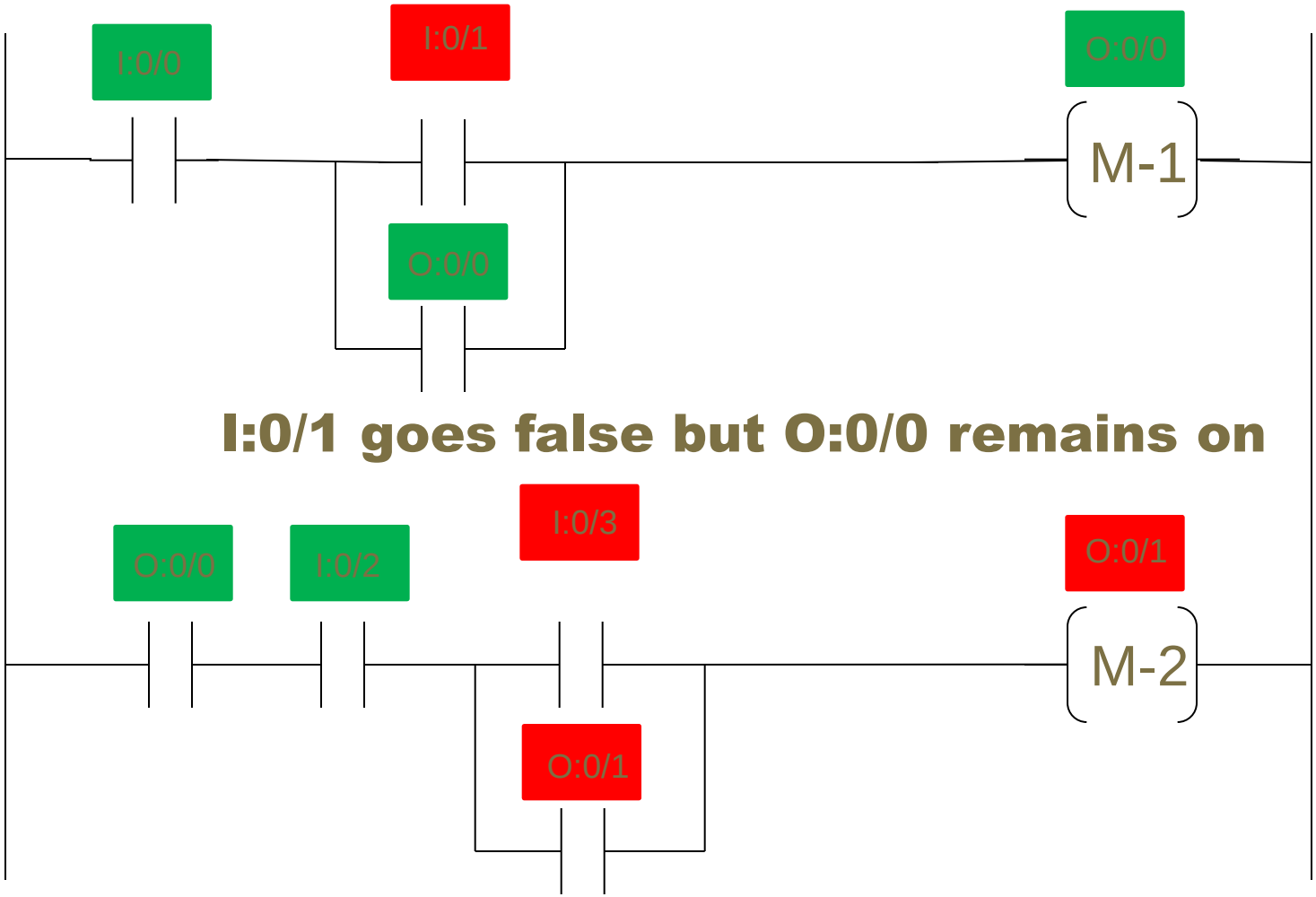
CONVERTING HARDWIRED CIRCUITS



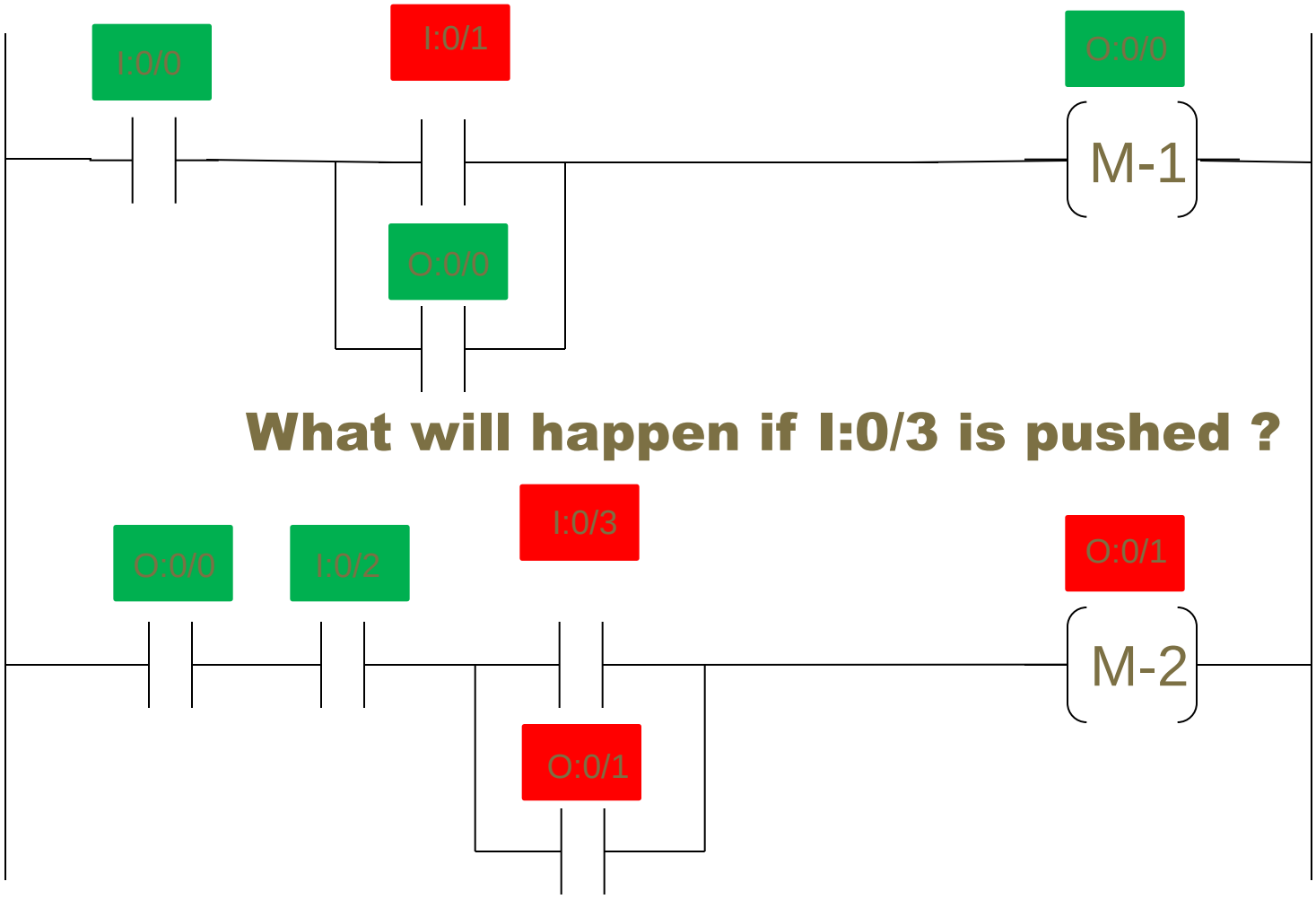
CONVERTING HARDWIRED CIRCUITS



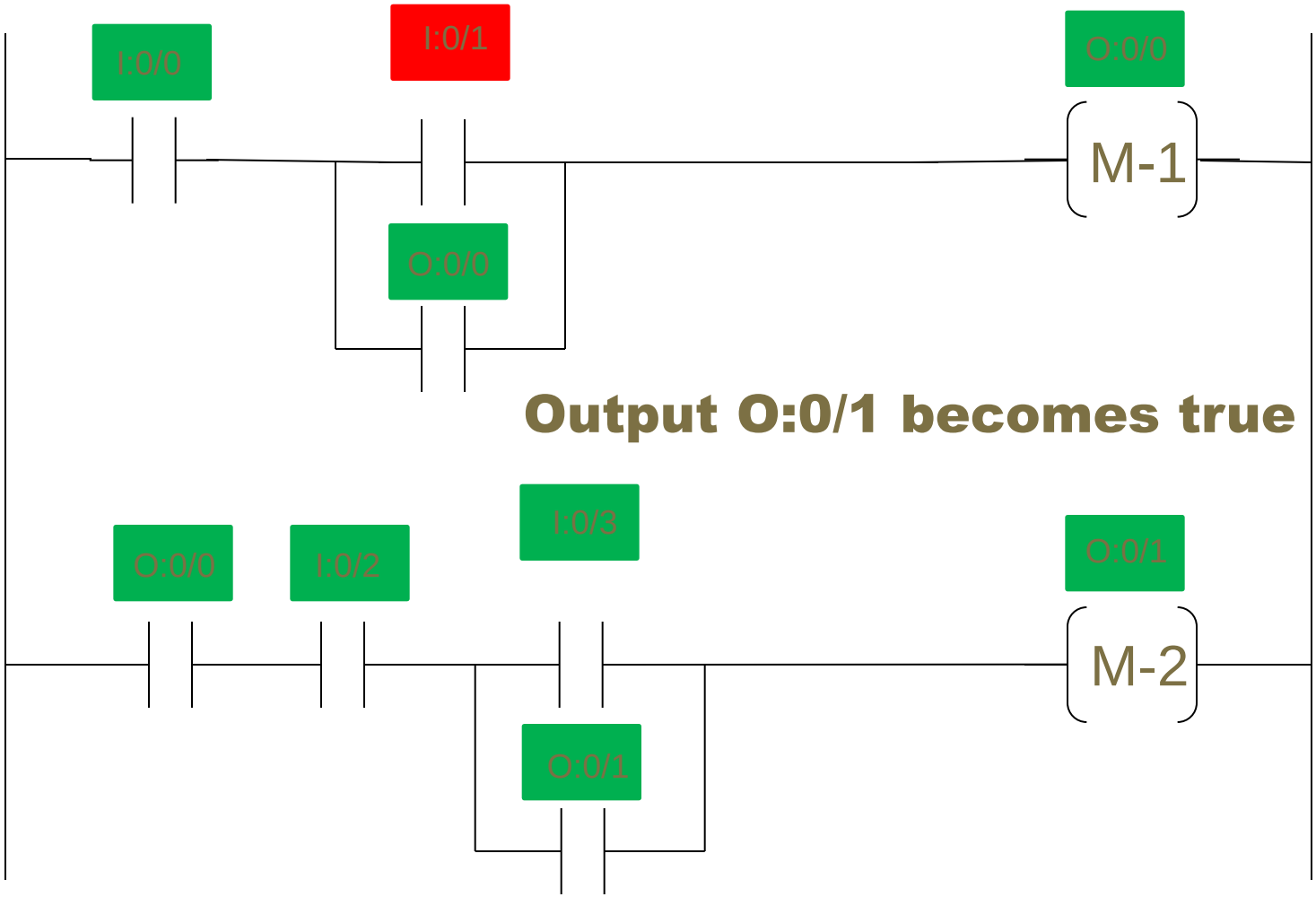
CONVERTING HARDWIRED CIRCUITS



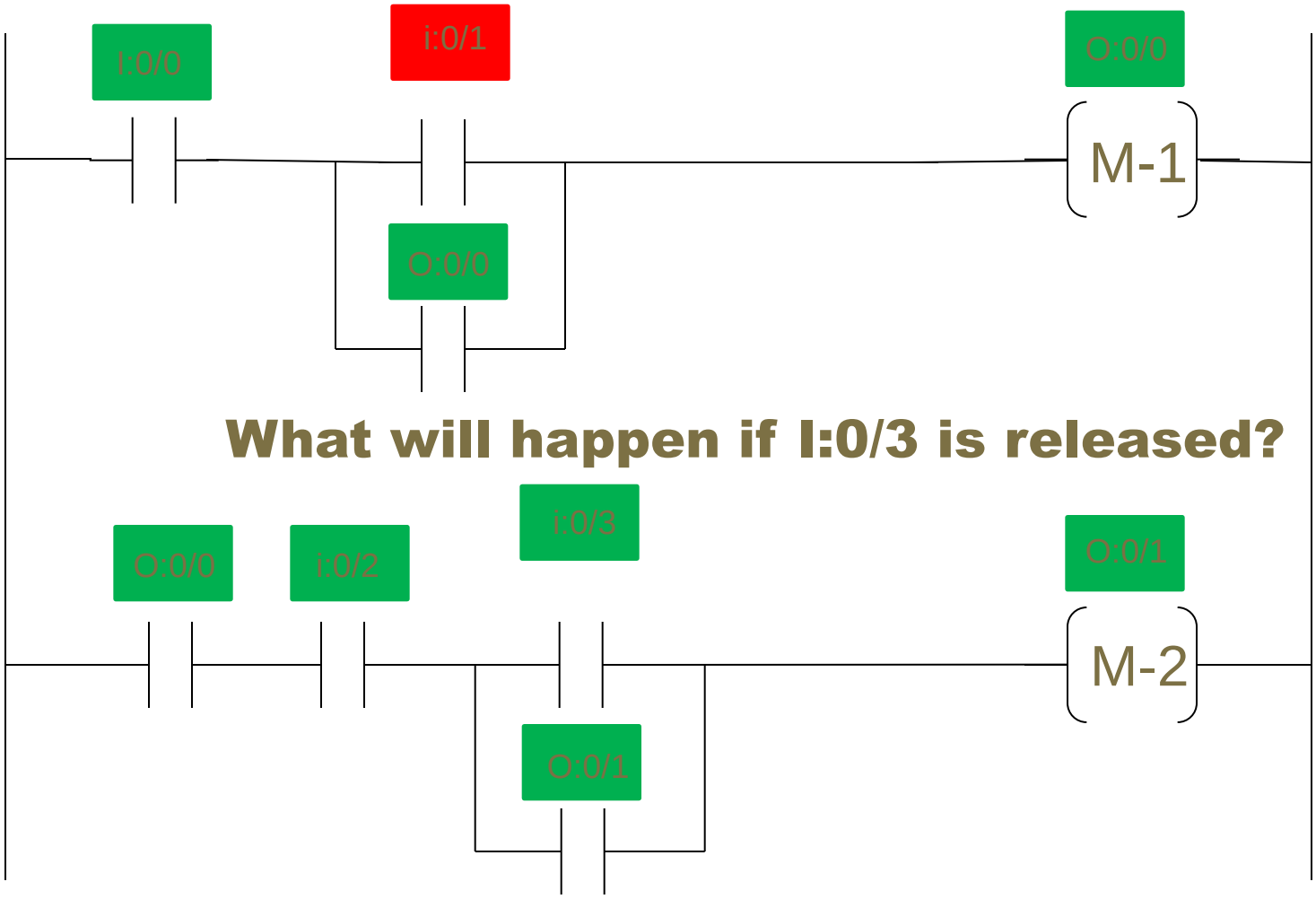
CONVERTING HARDWIRED CIRCUITS



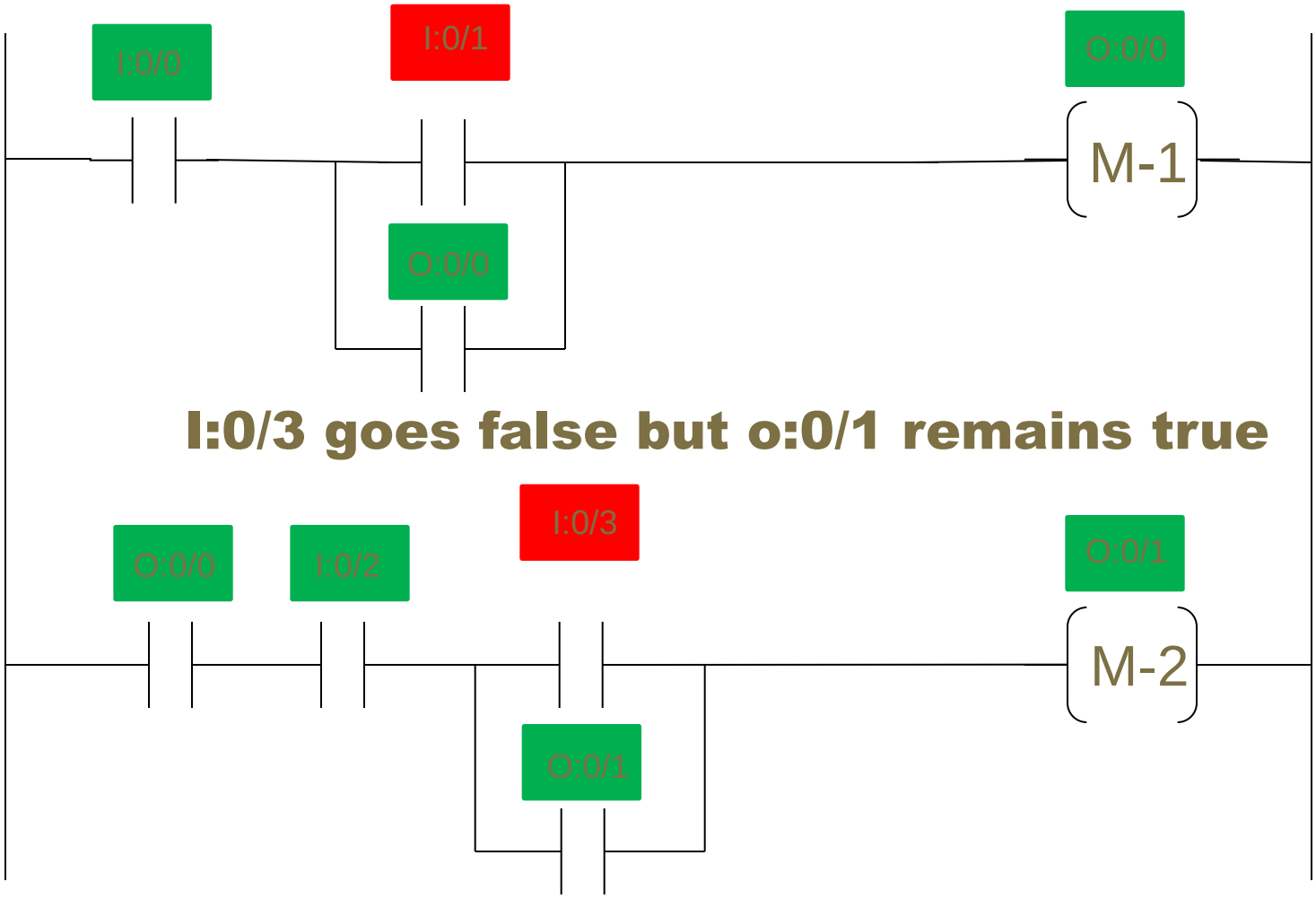
CONVERTING HARDWIRED CIRCUITS



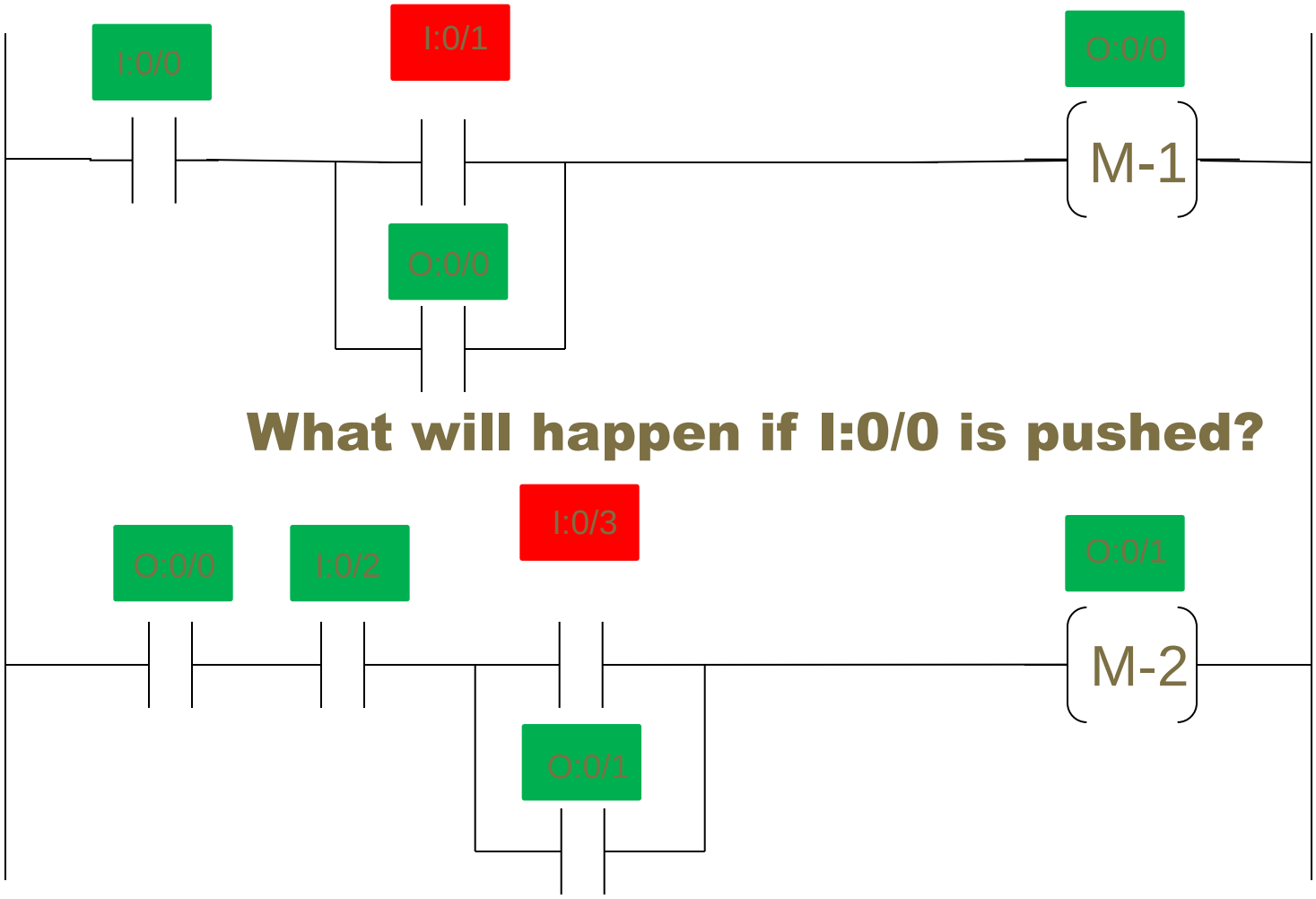
CONVERTING HARDWIRED CIRCUITS



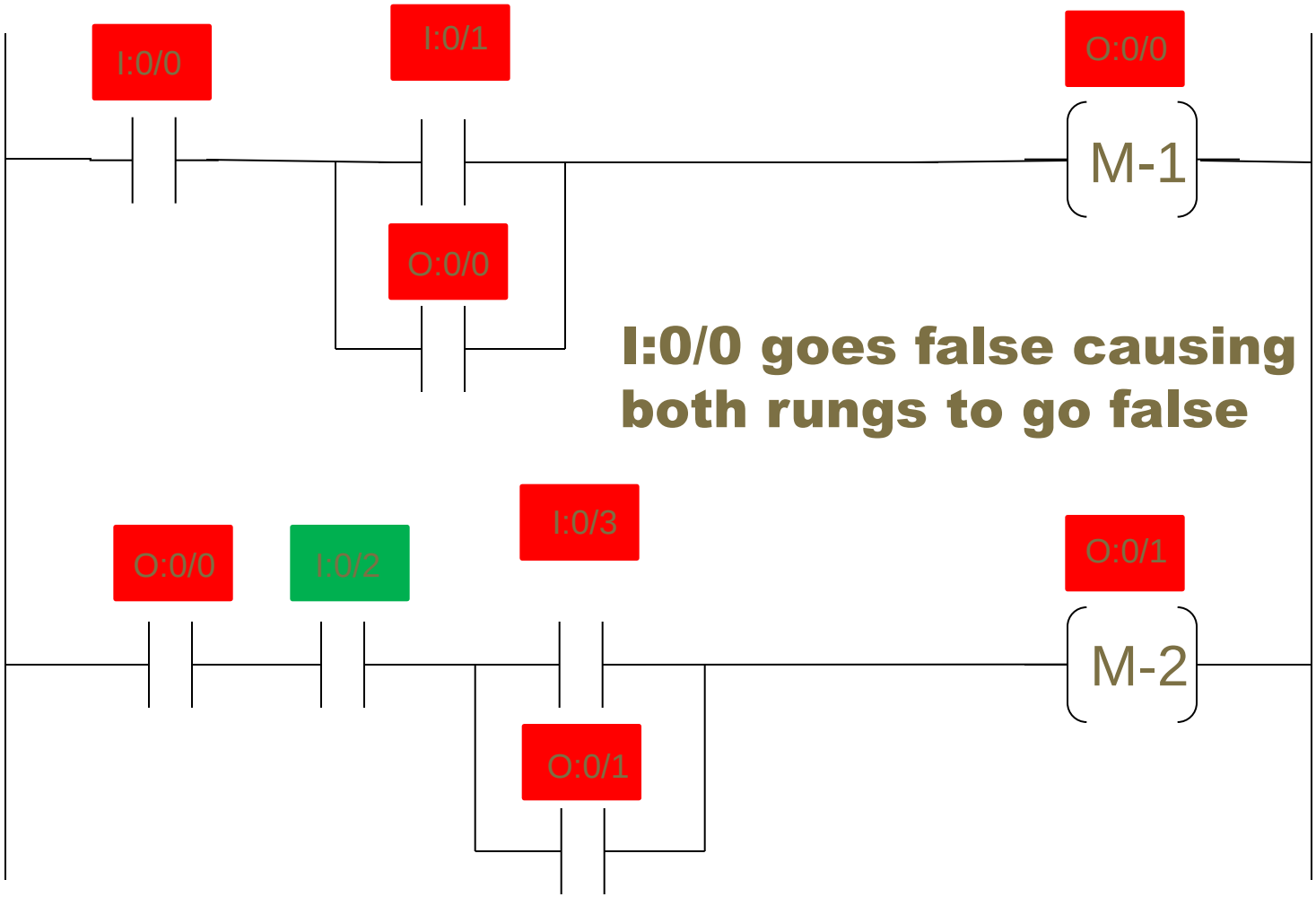
CONVERTING HARDWIRED CIRCUITS



CONVERTING HARDWIRED CIRCUITS



CONVERTING HARDWIRED CIRCUITS



ACTIVITY: LOGIXPRO INTRODUCTORY LAB

EXERCISE:

- **RSLogix Program Creation**

- **Enjoy your break!**

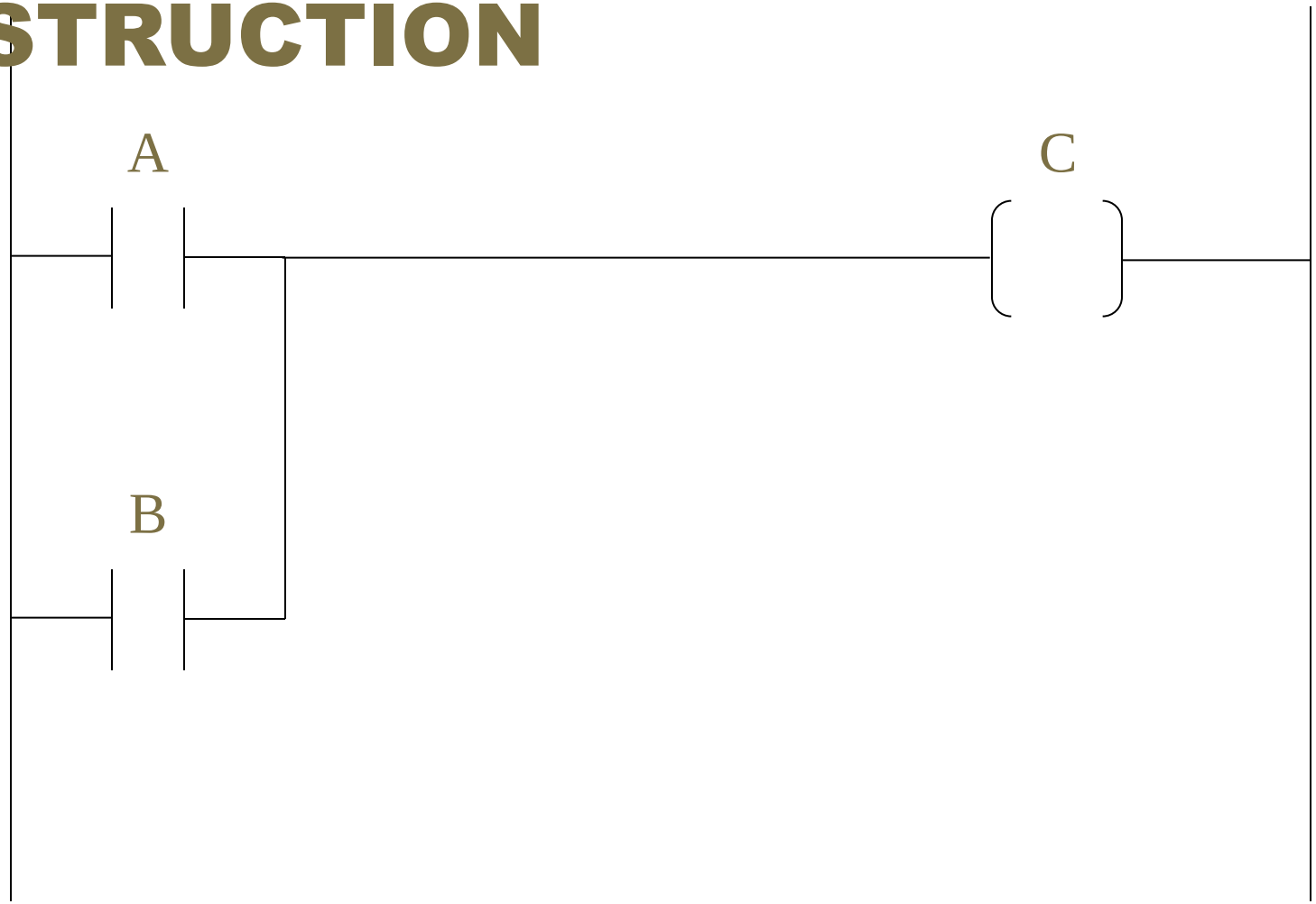
SESSION III

BRANCH INSTRUCTIONS & INTERNAL BITS

SESSION III: LEARNING OBJECTIVES

- At the conclusion of this session, participants will understand principles related to:
  - Branch instructions
  - Branch types
  - Programing limitations of PLC
  - Internal Relay instructions (Internal Bit)

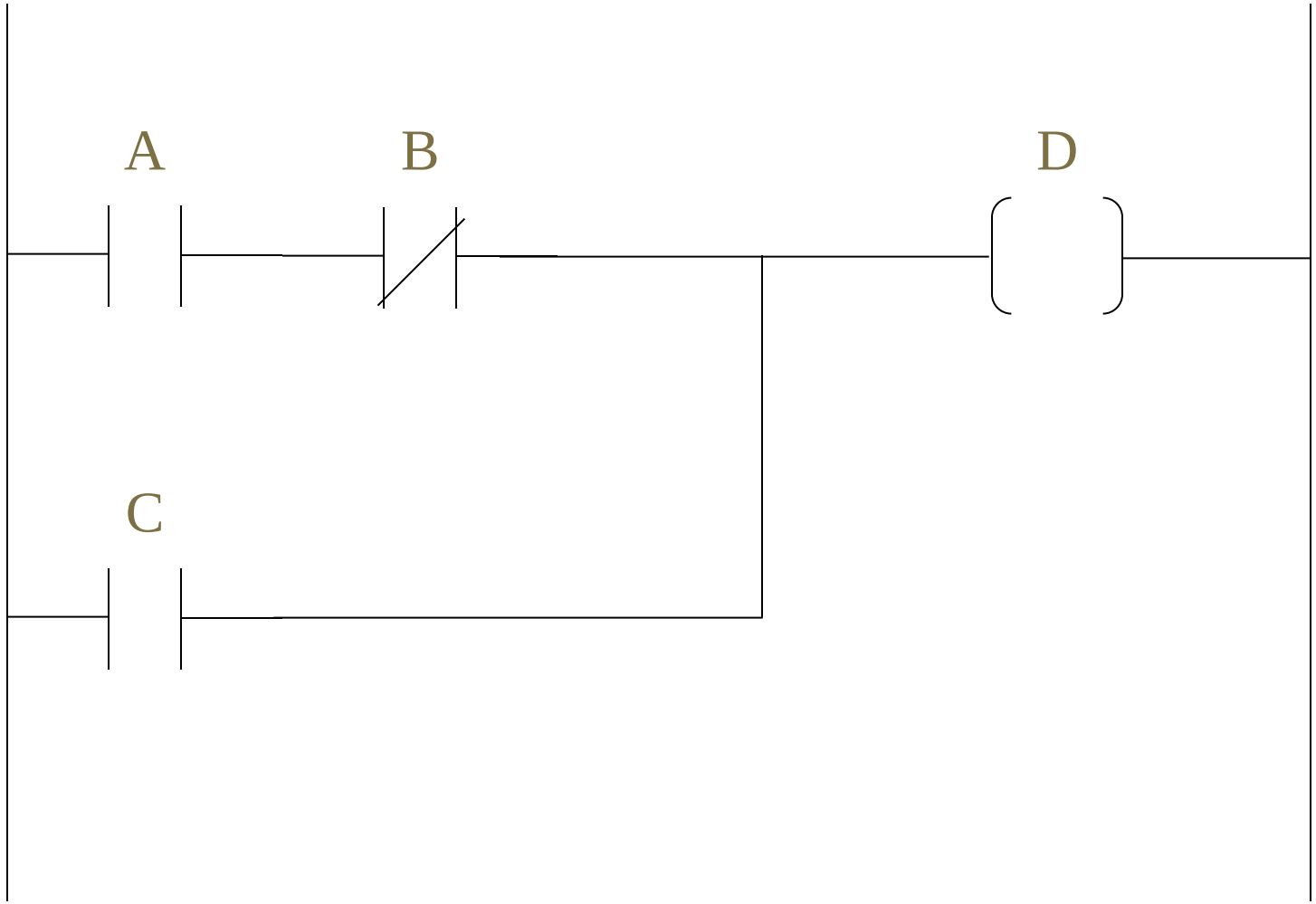
TYPICAL BRANCH INSTRUCTION

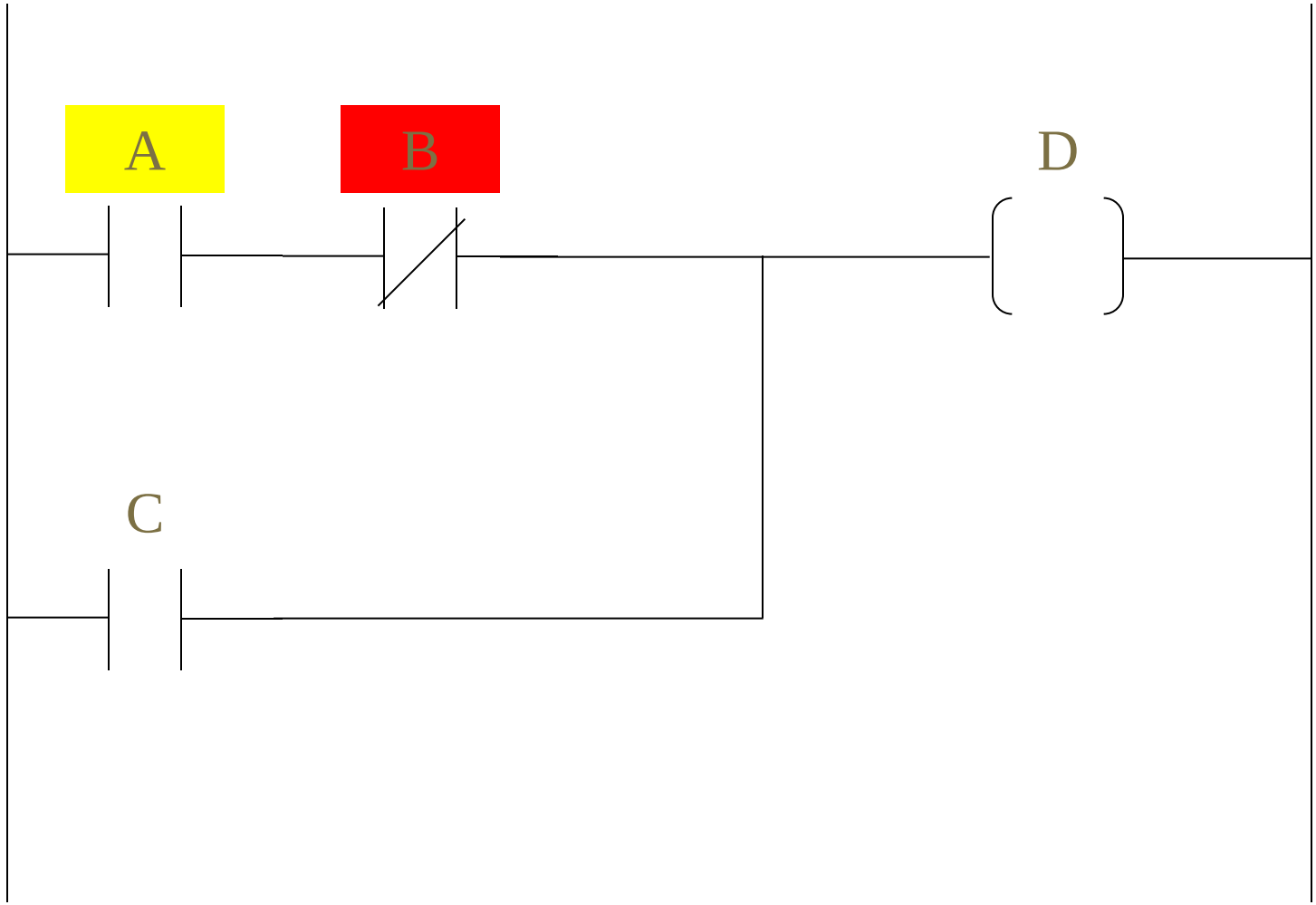


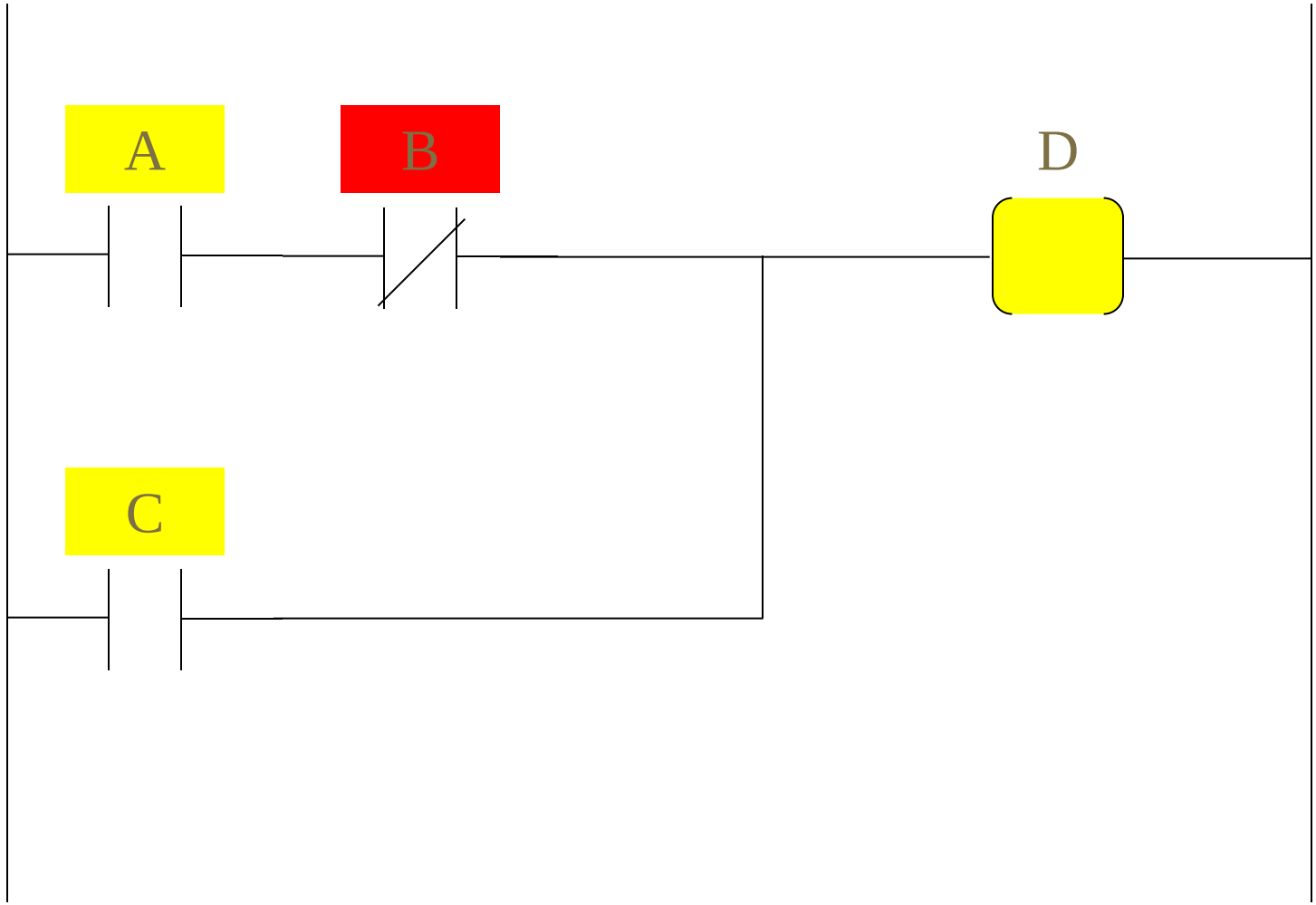
BRANCH INSTRUCTIONS

- Branch instructions are used to create parallel paths of input conditions. This allows for more than one path of logic to establish continuity in a rung
- Input branching by formation of parallel branches can be used in a program to allow a combination of input conditions
- If at least one branch forms a true logic path, the rung logic is true and the output will energize

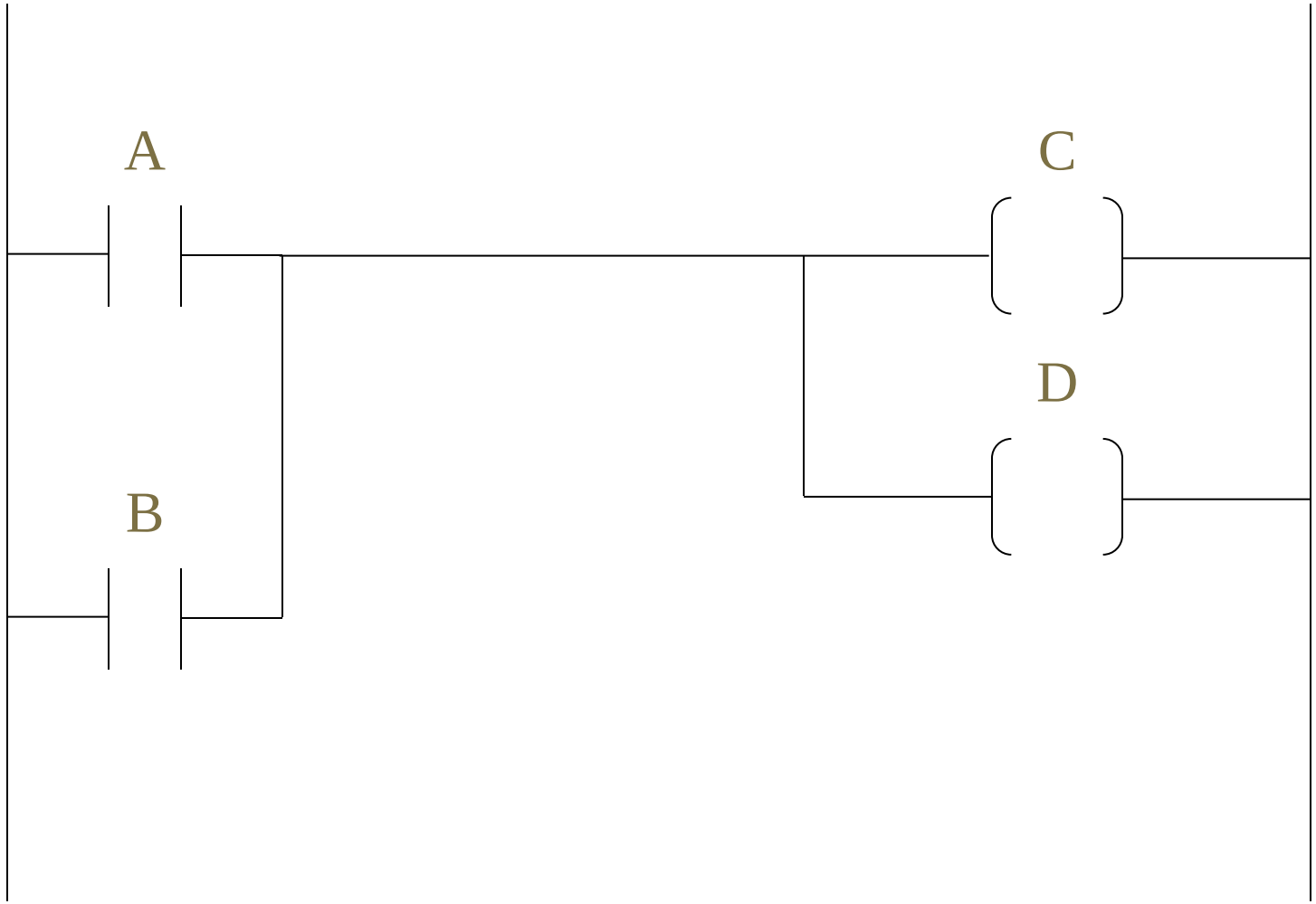
PARALLEL INPUT BRANCHES

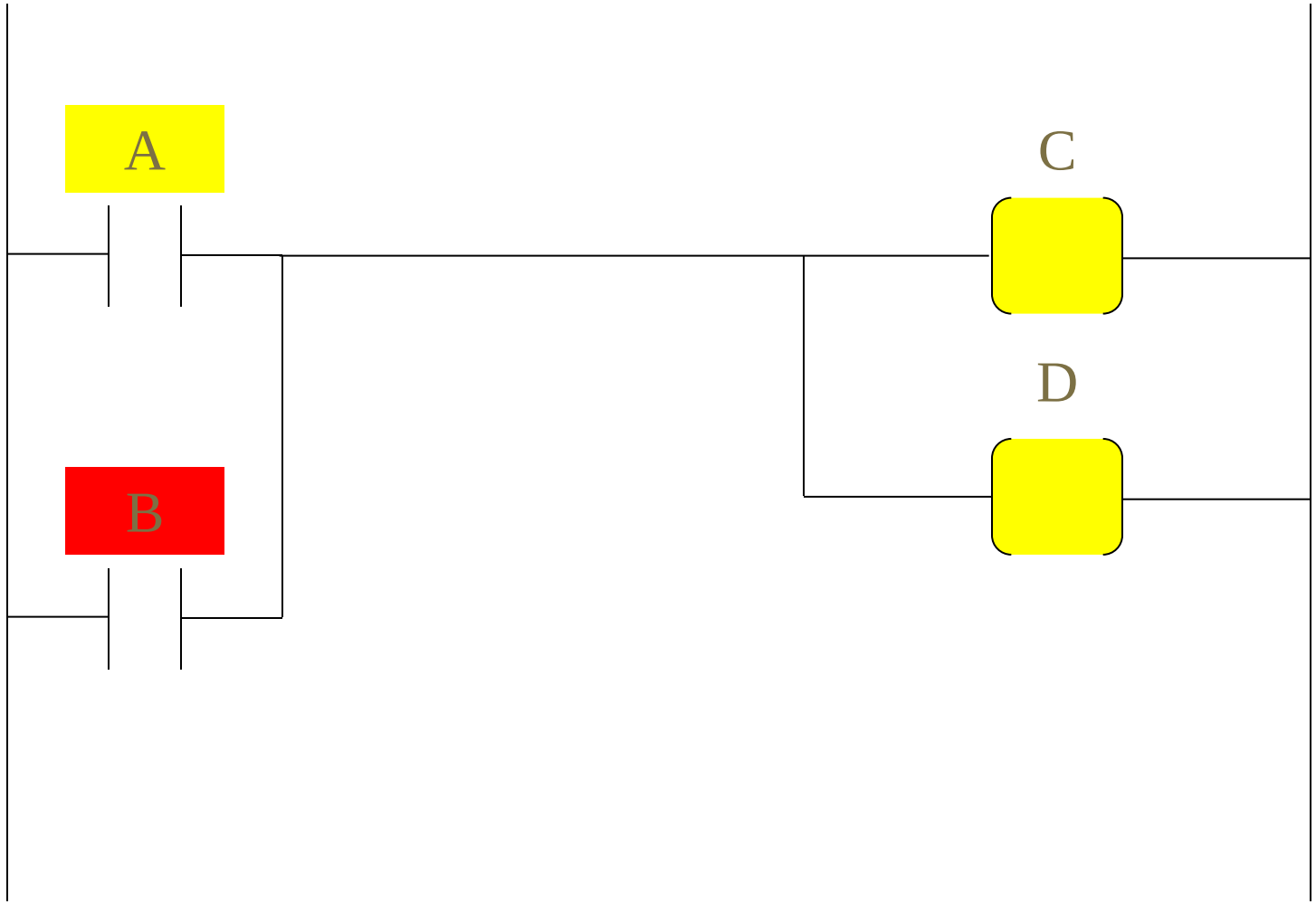



A
B
D
C


A
B
D
C

PARALLEL OUTPUT BRANCHES

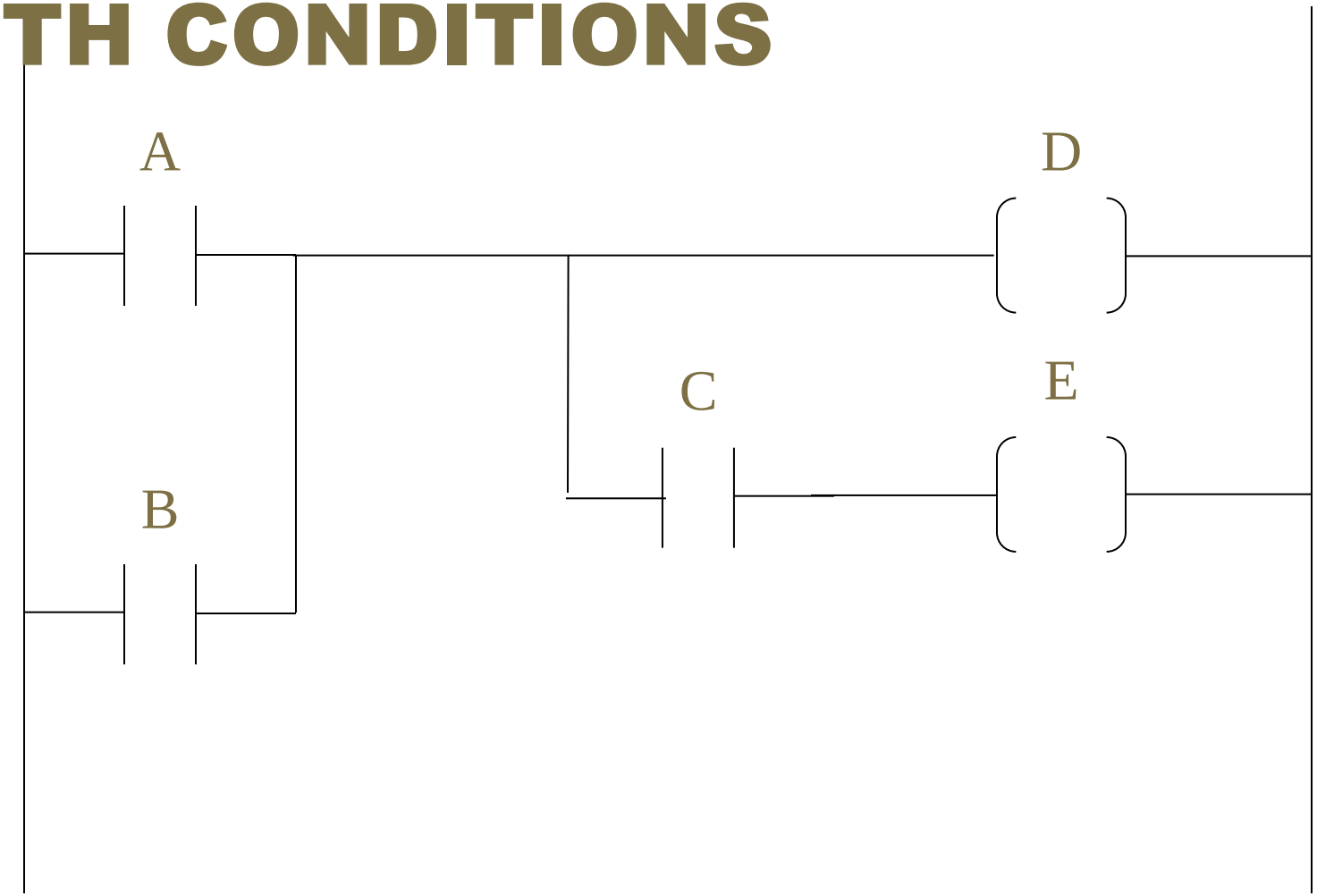



A
B
C
D

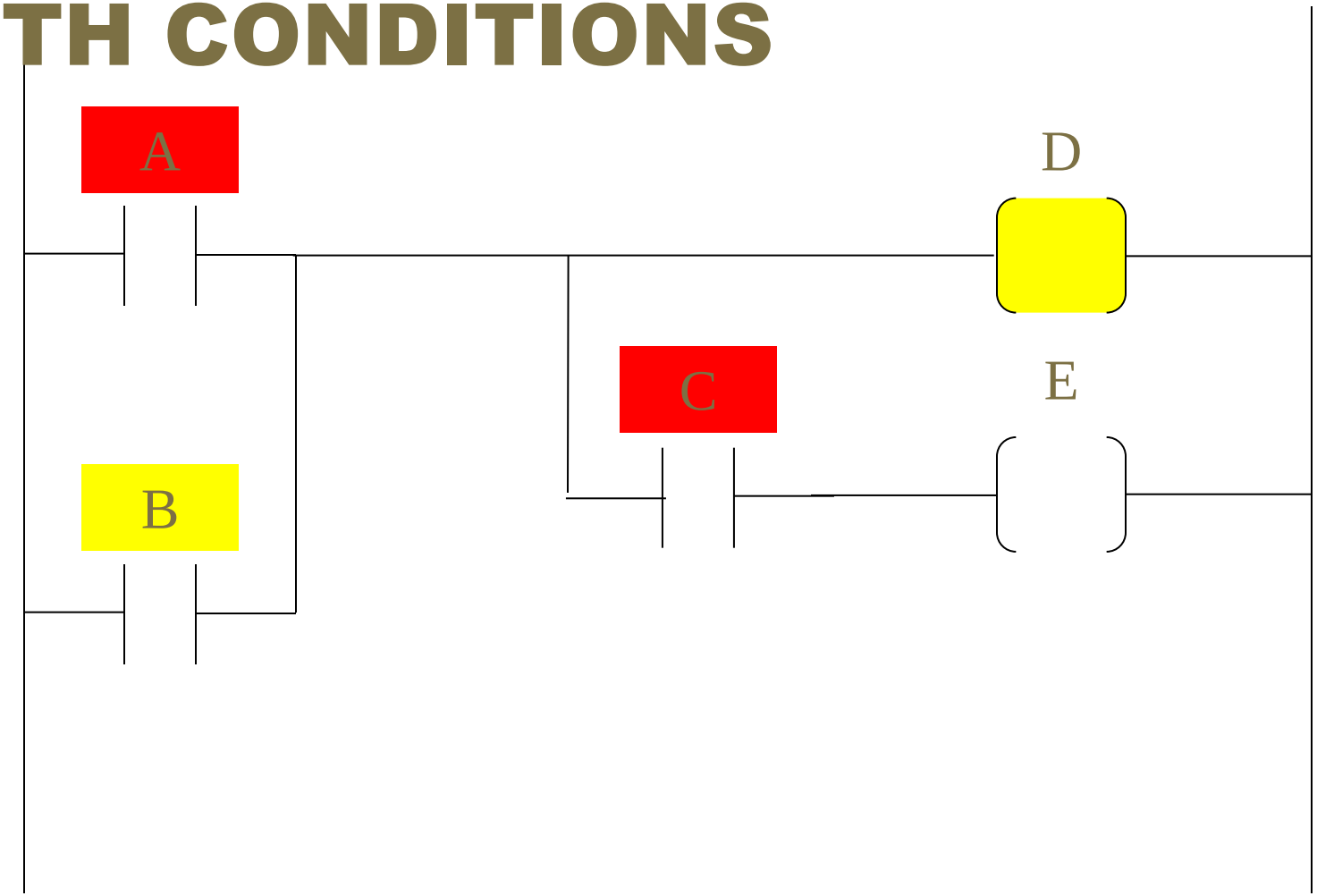
BRANCHING WITH CONDITIONS

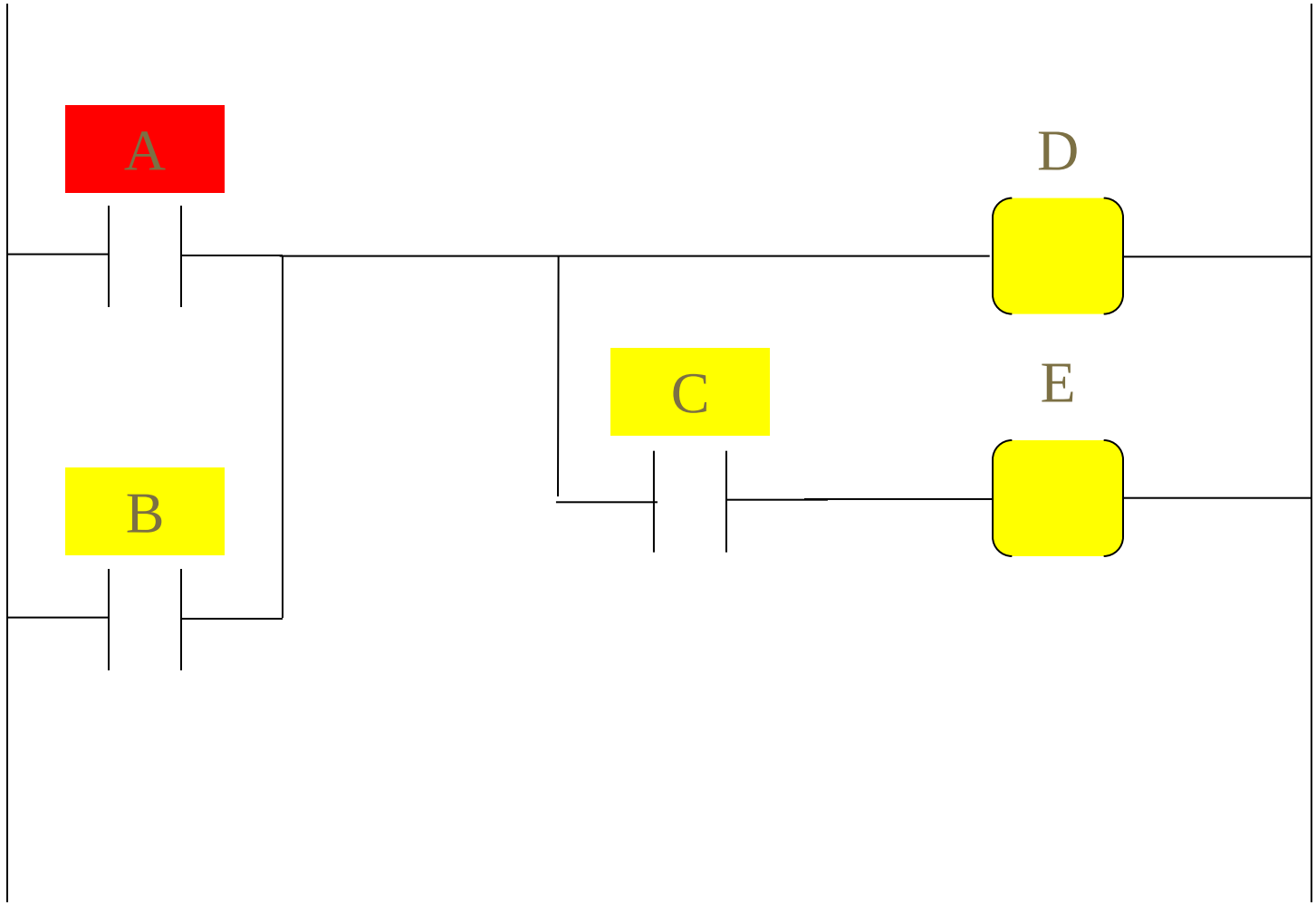
- Additional input logic instructions can be programmed in the output branches to enhance conditional control of the outputs
- When a true path of logic exists to the output branch, including extra conditions allows for greater control of the outputs
  - For example: Two sump pumps, a main and auxiliary pump are responsible for drainage. If the main pump can't keep up with demand the auxiliary pump would start up

PARALLEL OUTPUT BRANCHES WITH CONDITIONS



PARALLEL OUTPUT BRANCHES WITH CONDITIONS

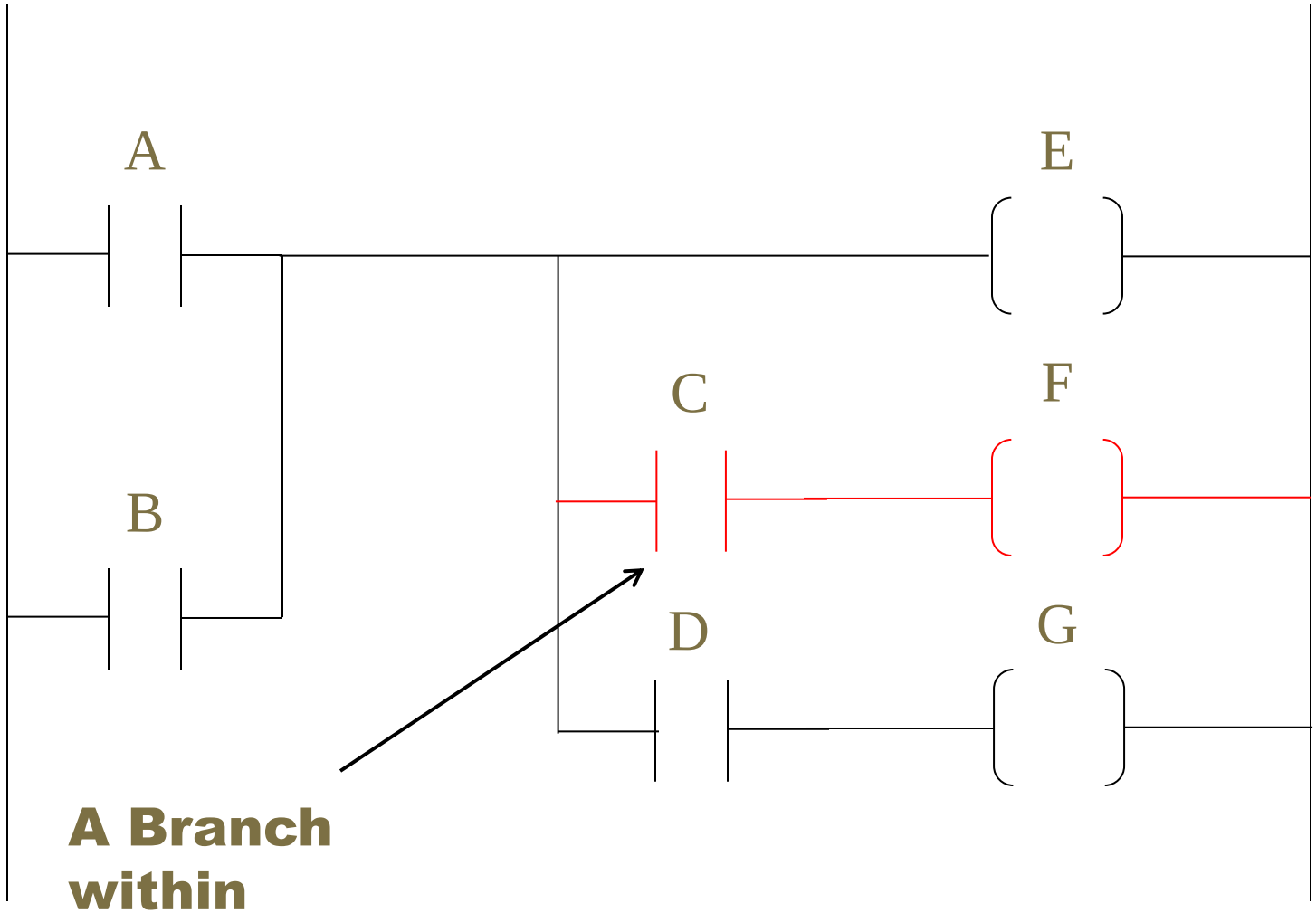



A
B
C
D
E

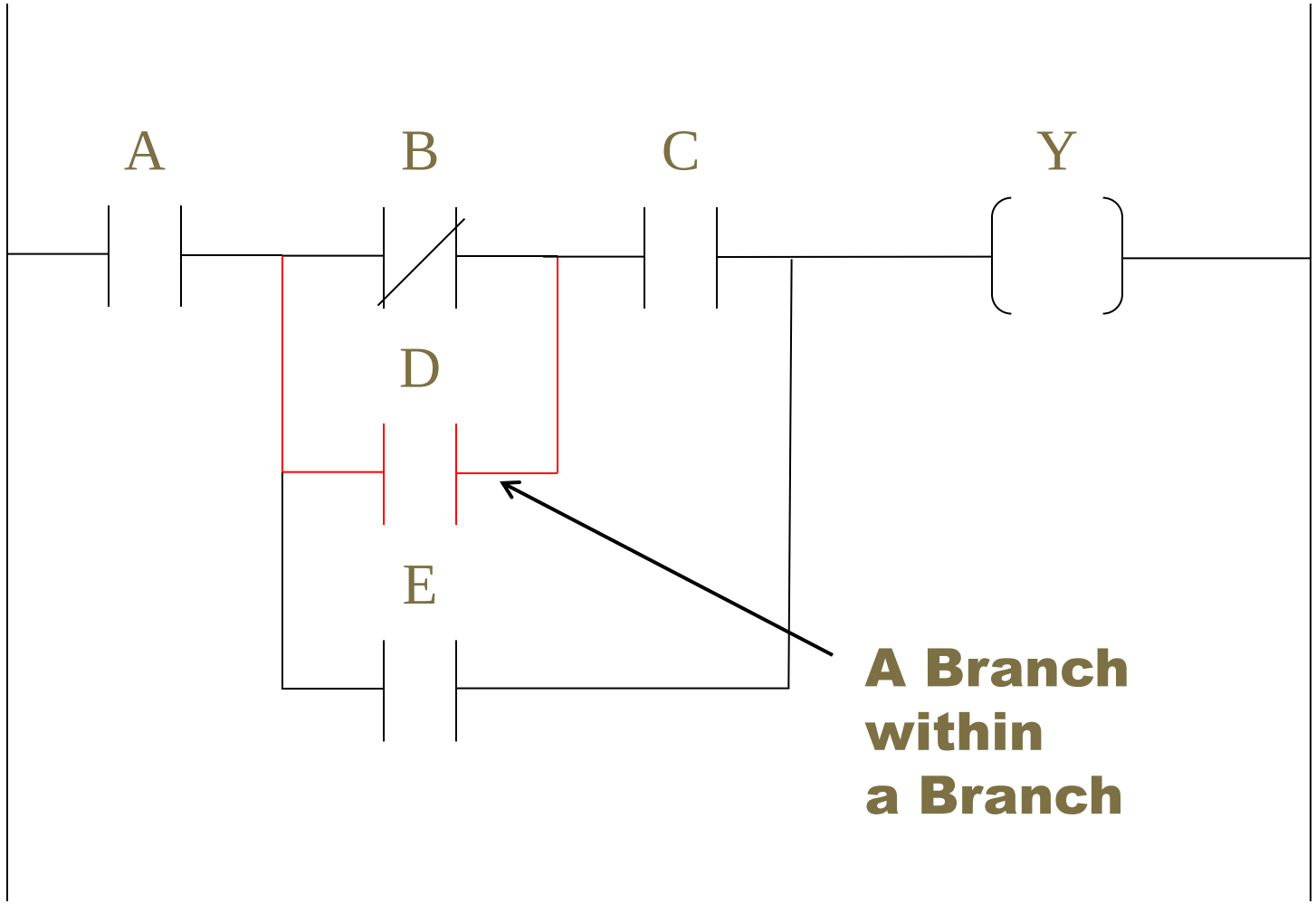
NESTED BRANCHES

- Nesting, is a branch within a branch. Input and output branches can be nested to avoid redundant instructions and to speed up processor scan time. A nested branch starts or ends within another branch
- In some PLC models, the programming of a branch circuit within a branch, or nesting, can not be done directly. It is possible though to program an equivalent branching condition

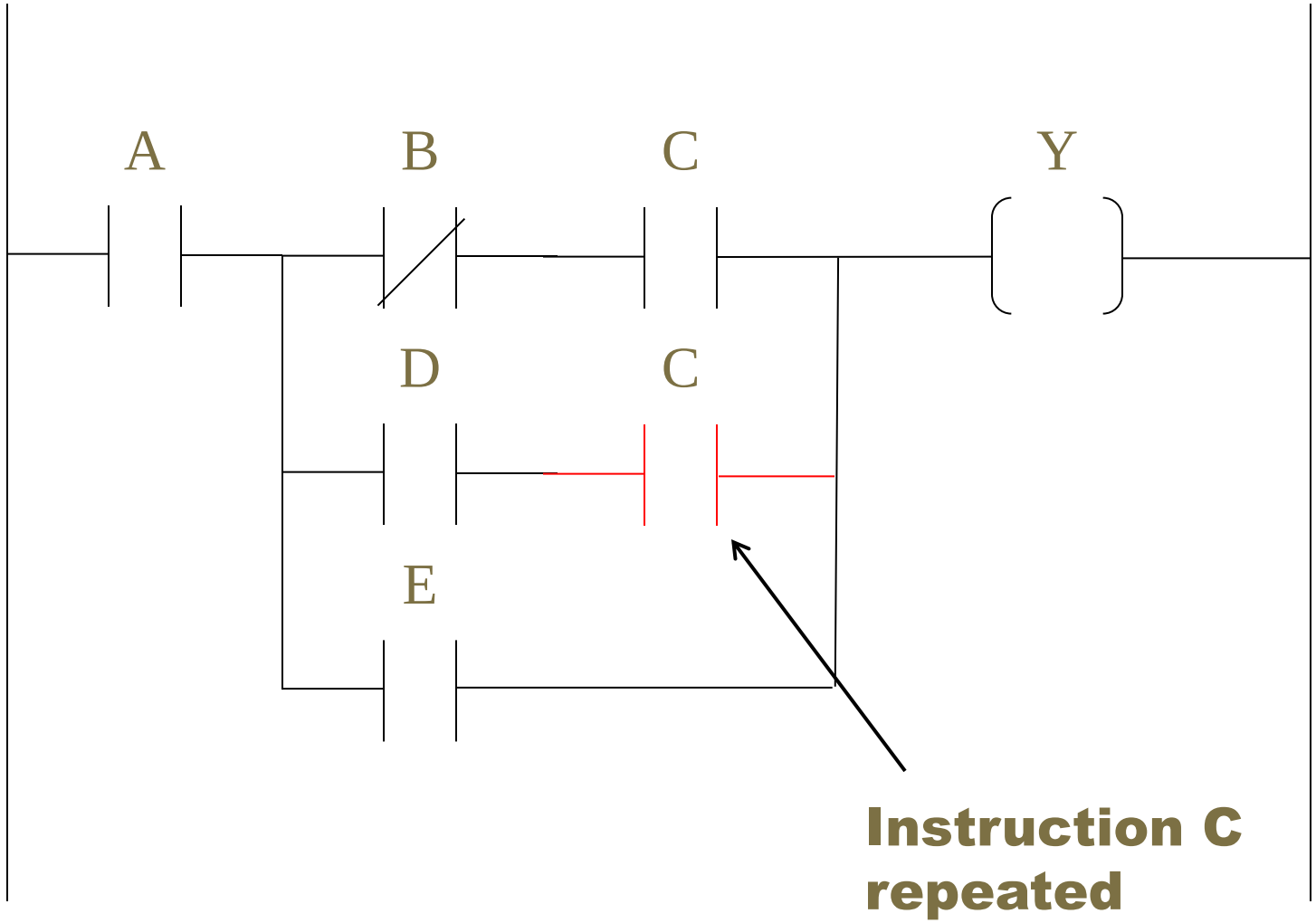
NESTED OUTPUTS



NESTED INPUT



REMOVING NESTED INPUT



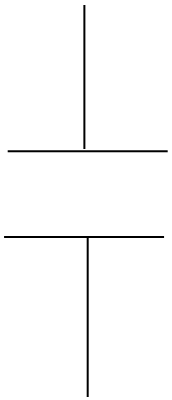
PROGRAMMING LIMITATIONS OF PLCS

- Vertical Contacts (Not Allowed)
- Number of Series Contacts on 1 Rung (Typically)
- Number of Parallel Branches on 1 Rung (Typically)

PROGRAMMING LIMITATIONS OF PLCS

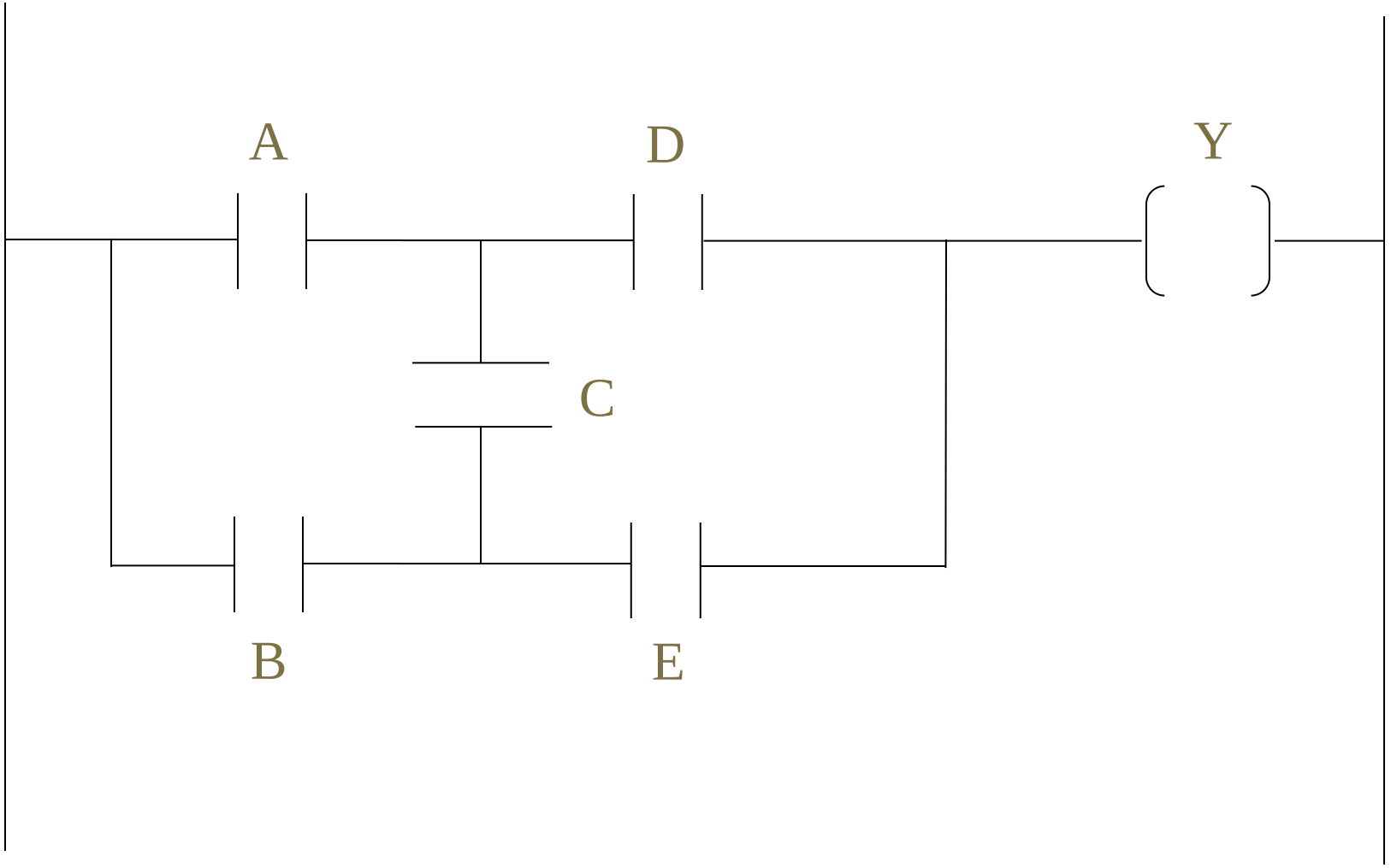
- **Number of Series Contacts on 1 Rung (Typically)**
- **Number of Outputs on 1 Rung (Typically)**

VERTICAL CONTACTS

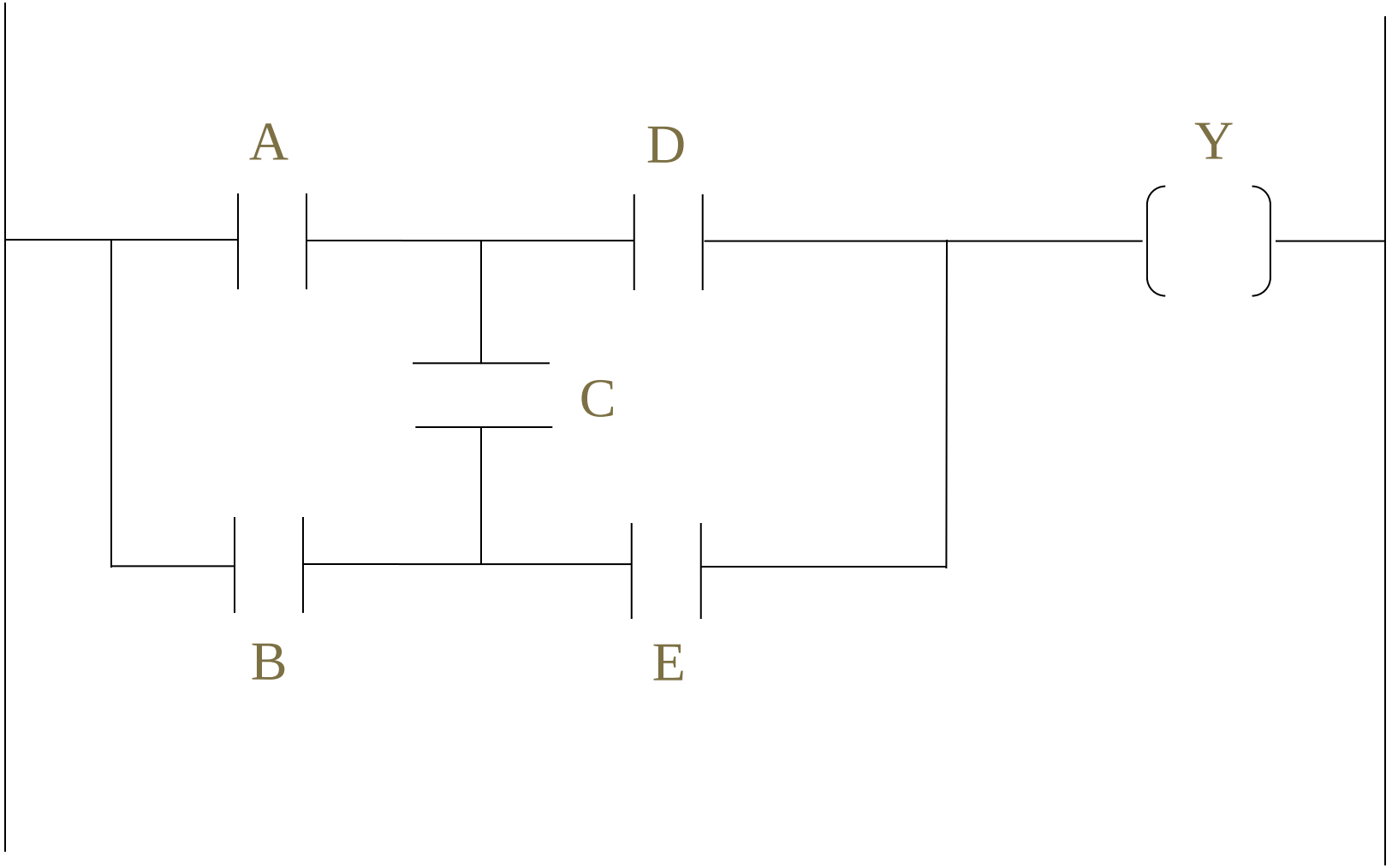


- **Because PLCs scan the logic of a program from left to right and top to bottom only, programming of vertical contacts is not allowed. Vertical contacts can be removed from programming by determining all the paths of logic that flows thru the vertical contact and rewriting it horizontally**

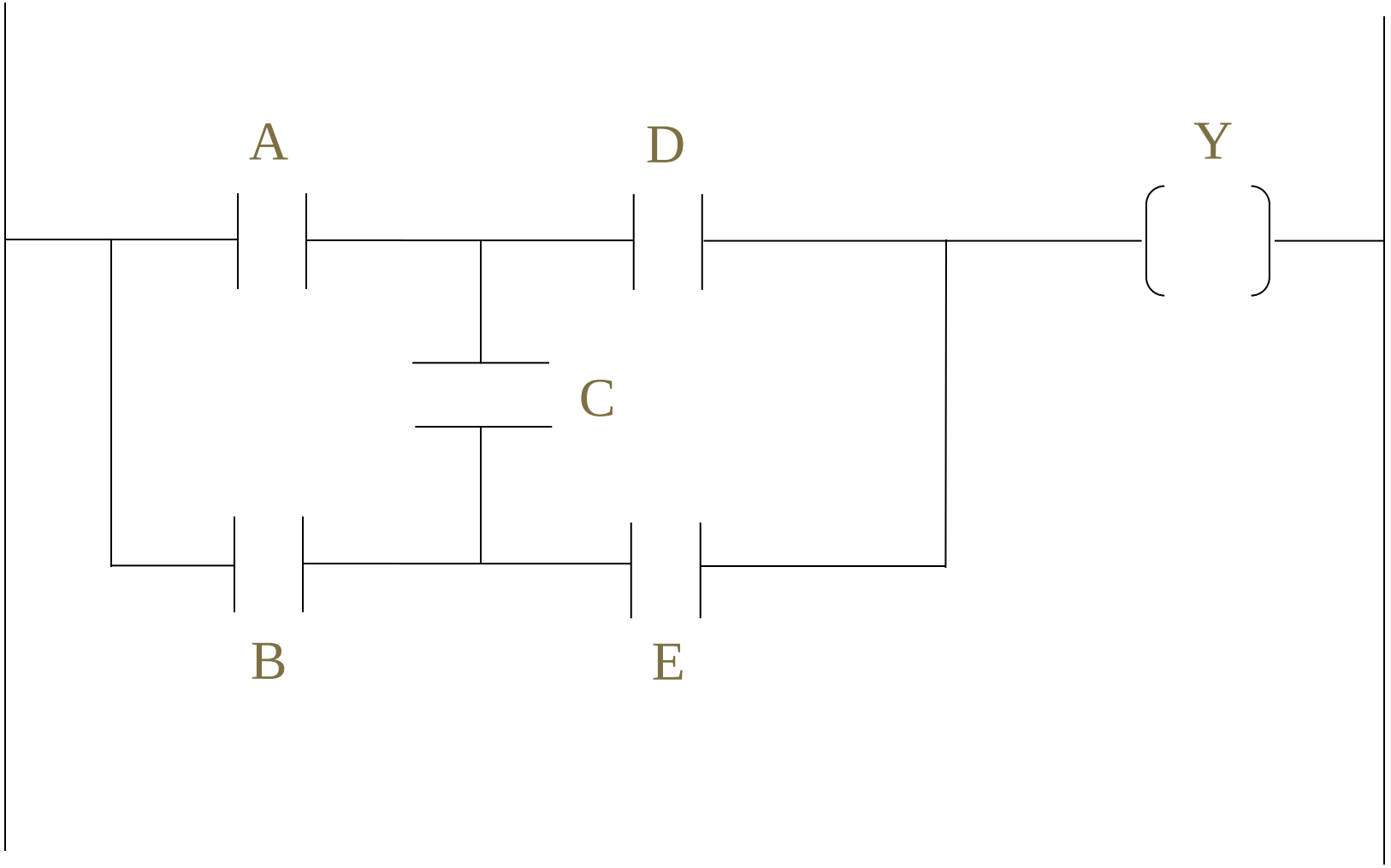
PROGRAM WITH A VERTICAL CONTACT



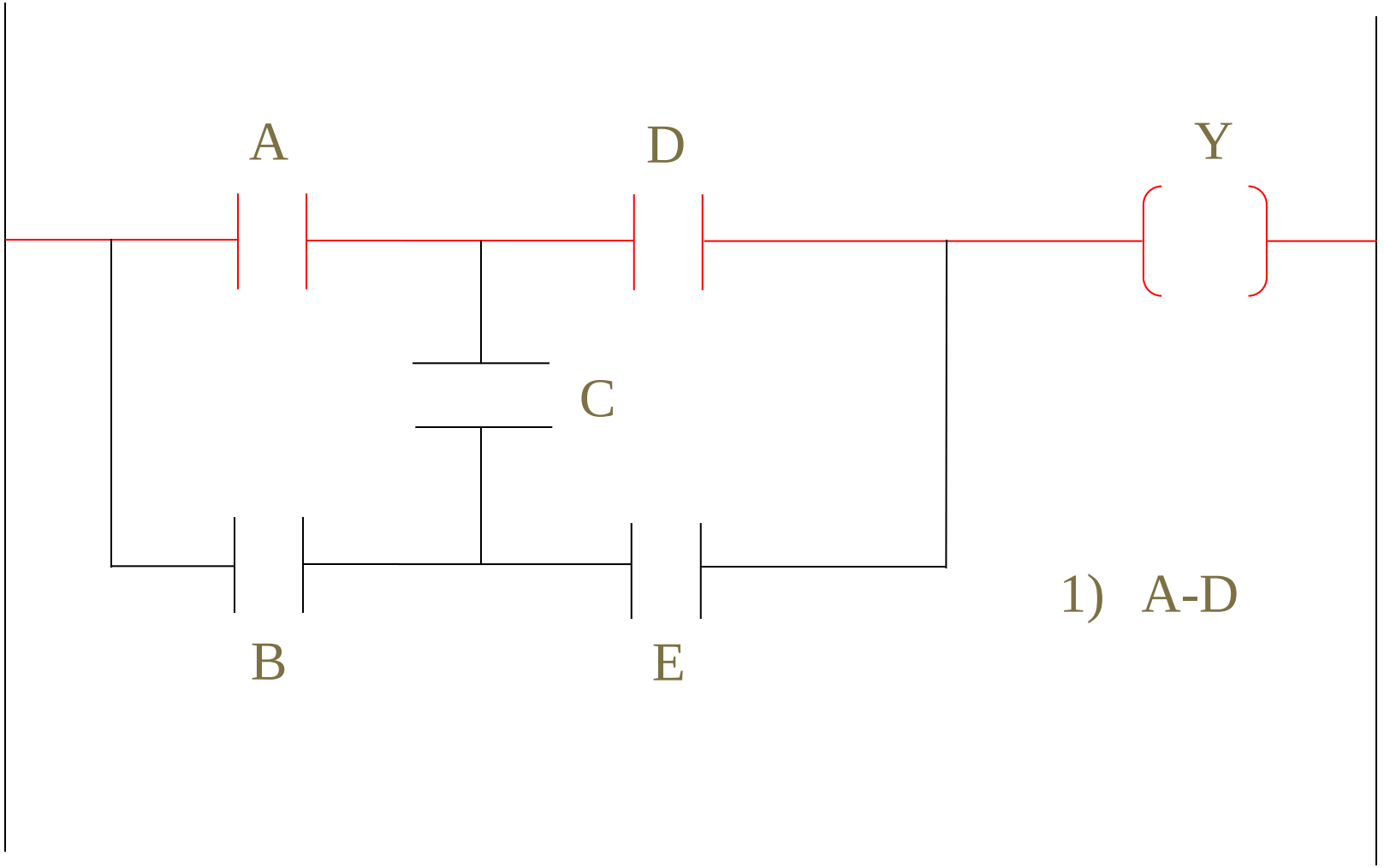
REWRITE PROGRAM TO ELIMINATE VERTICAL CONTACT



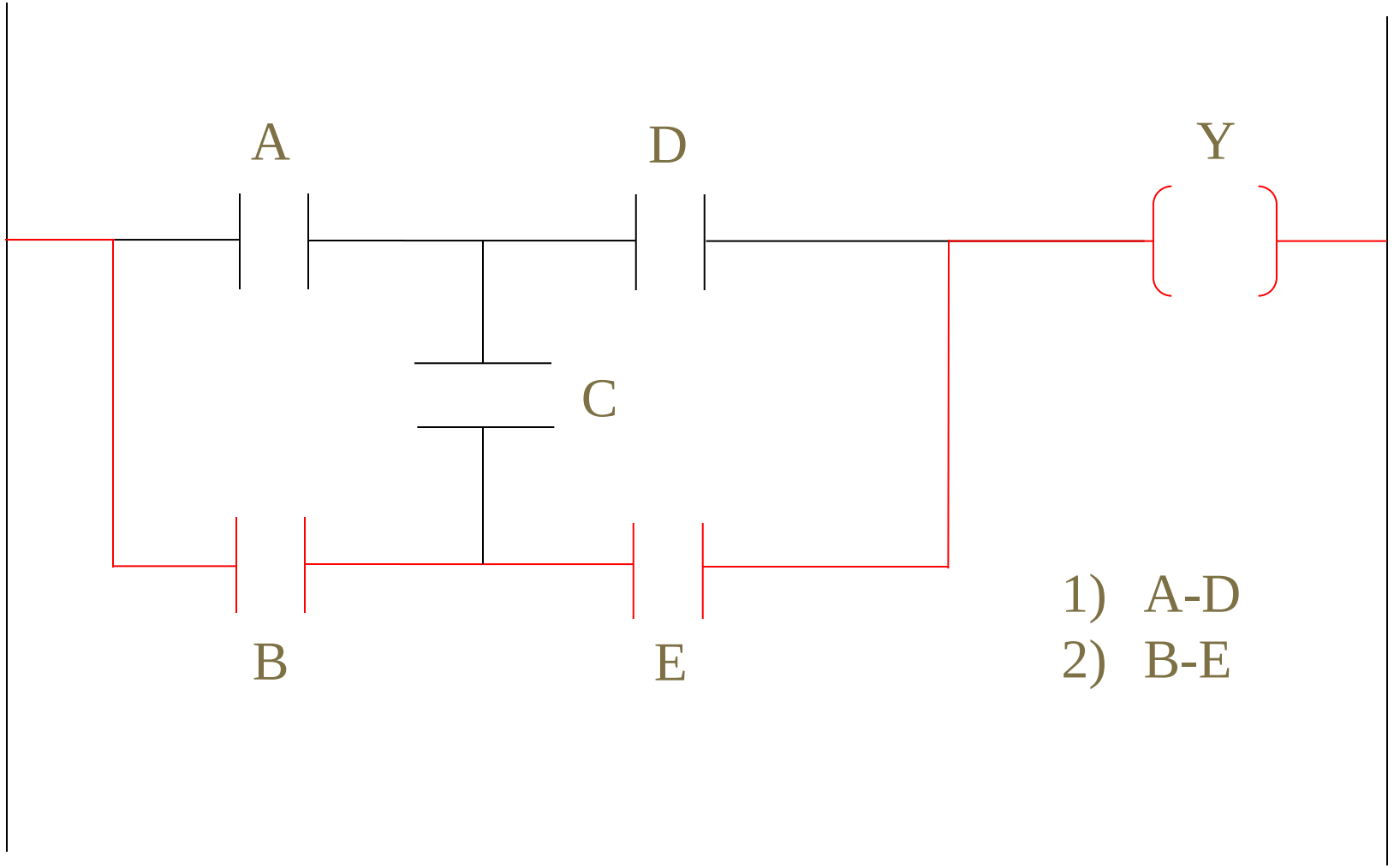
FIRST DETERMINE ALL PATHS FOR LOGIC TO FLOW



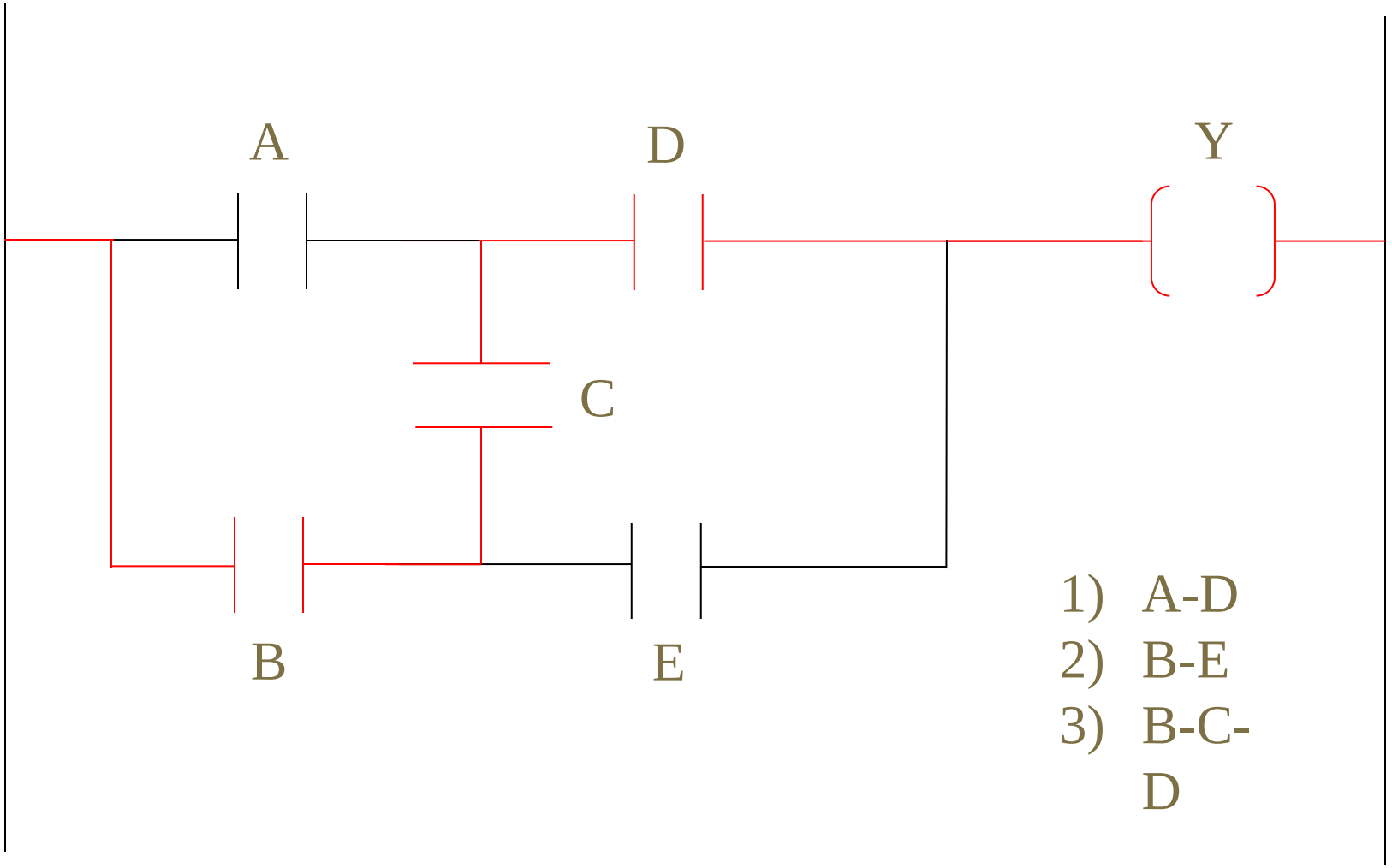
FIRST DETERMINE ALL PATHS FOR LOGIC TO FLOW



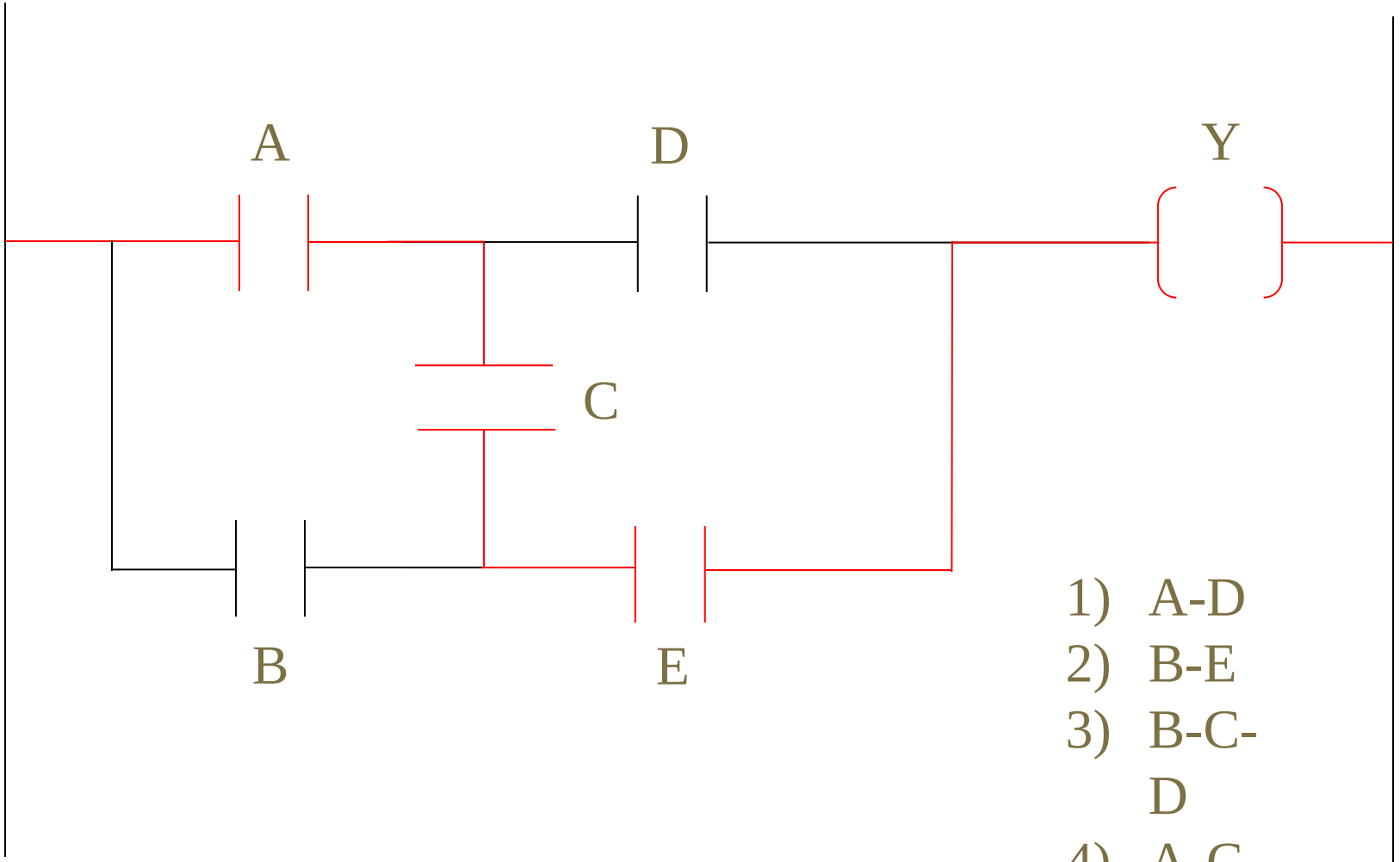
FIRST DETERMINE ALL PATHS FOR LOGIC TO FLOW



FIRST DETERMINE ALL PATHS FOR LOGIC TO FLOW

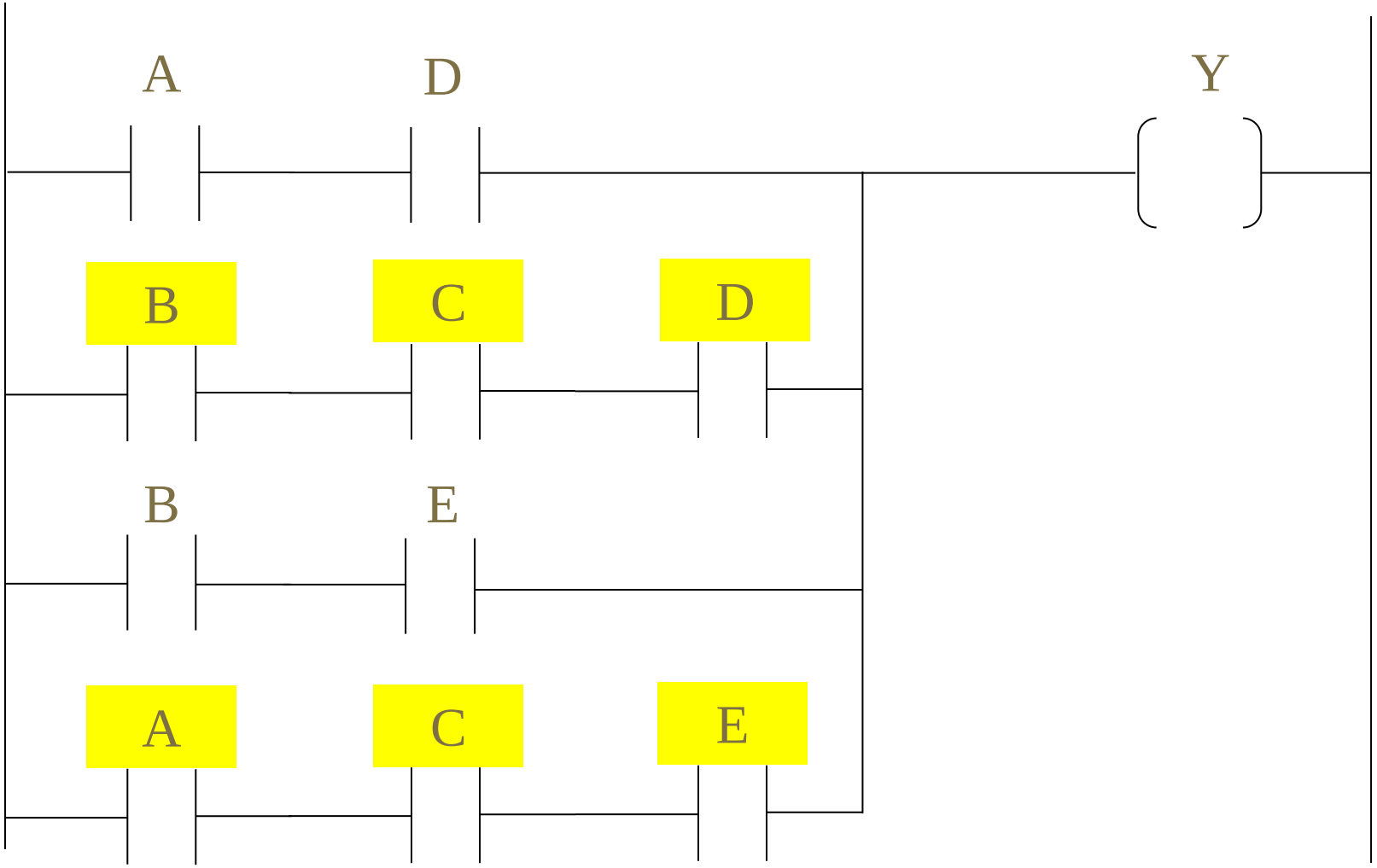


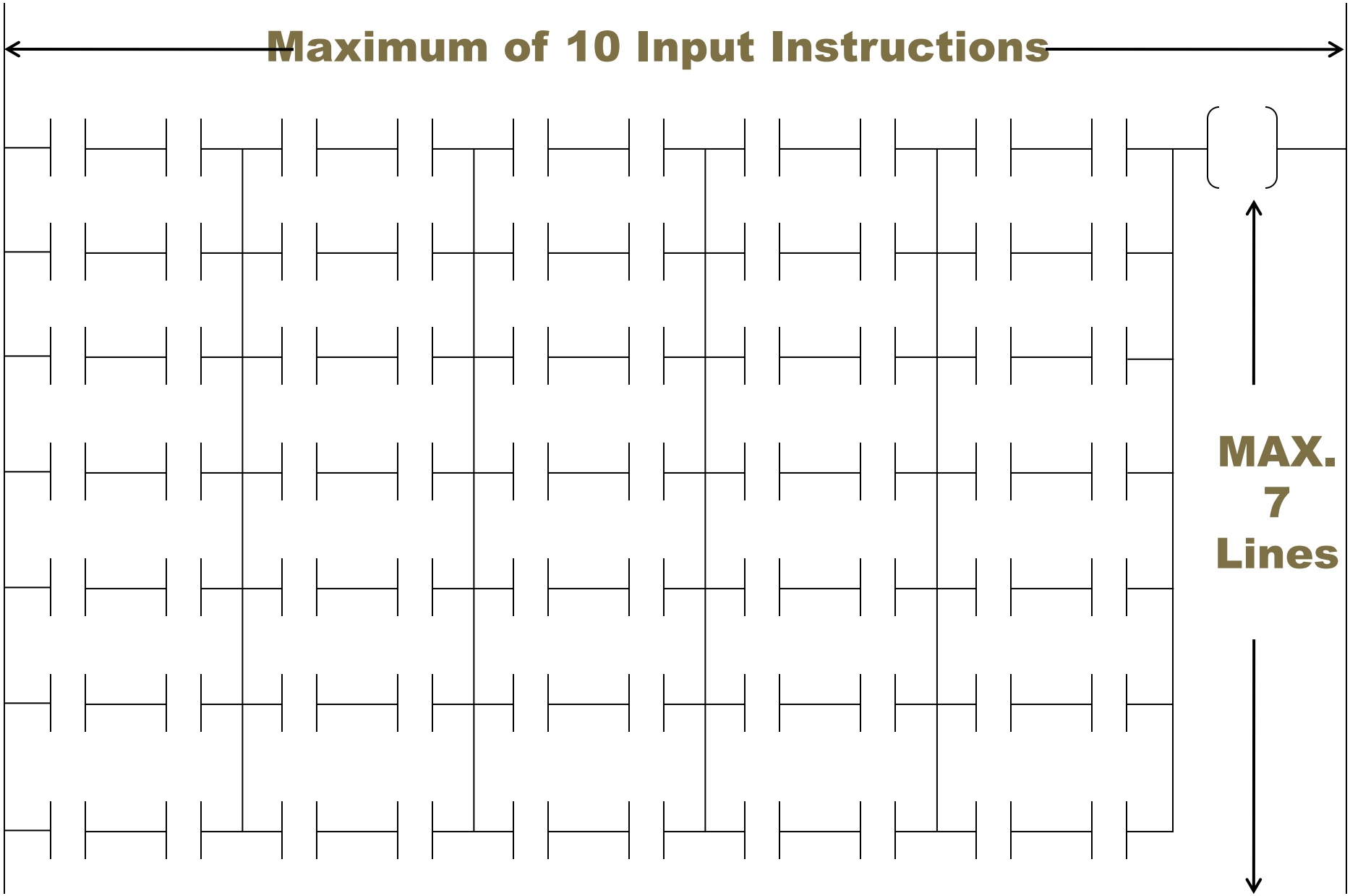
FIRST DETERMINE ALL PATHS FOR LOGIC TO FLOW



1) A-D
2) B-E
3) B-C-D
4) A-C-E

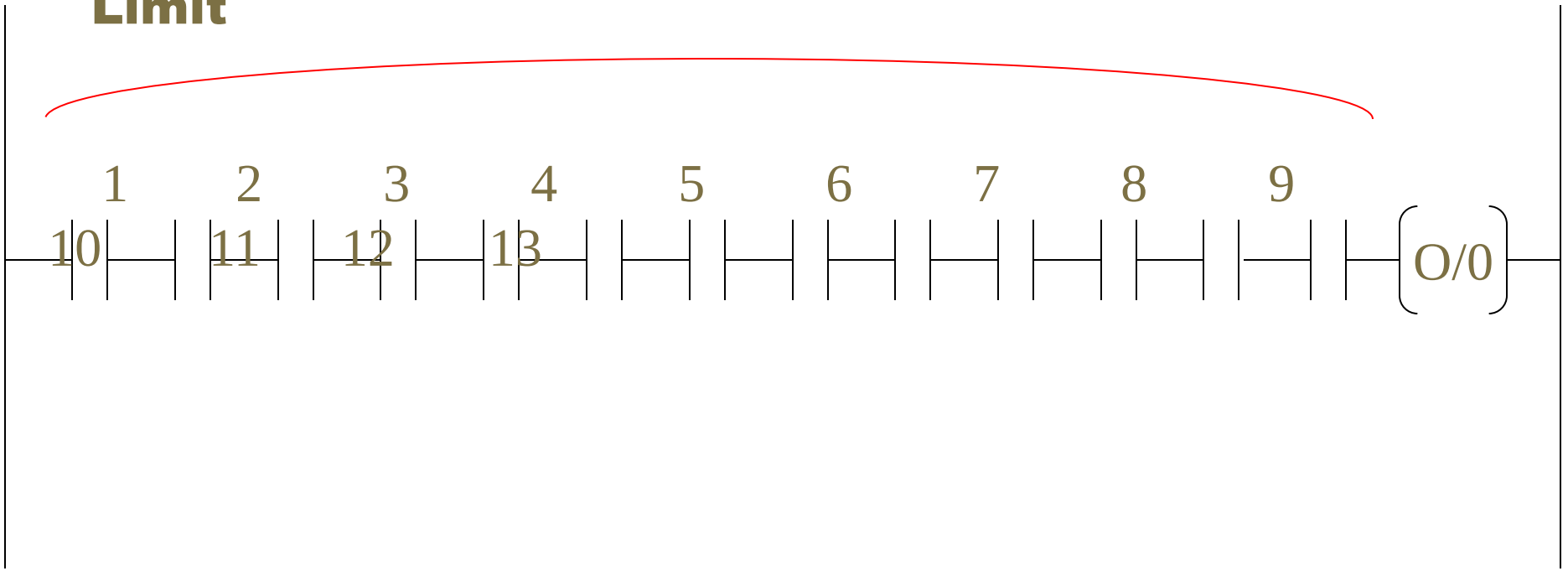
REWRITE THE PROGRAM ELIMINATING THE VERTICAL CONTACT INSTRUCTION




Maximum of 10 Input Instructions
MAX.
7
Lines

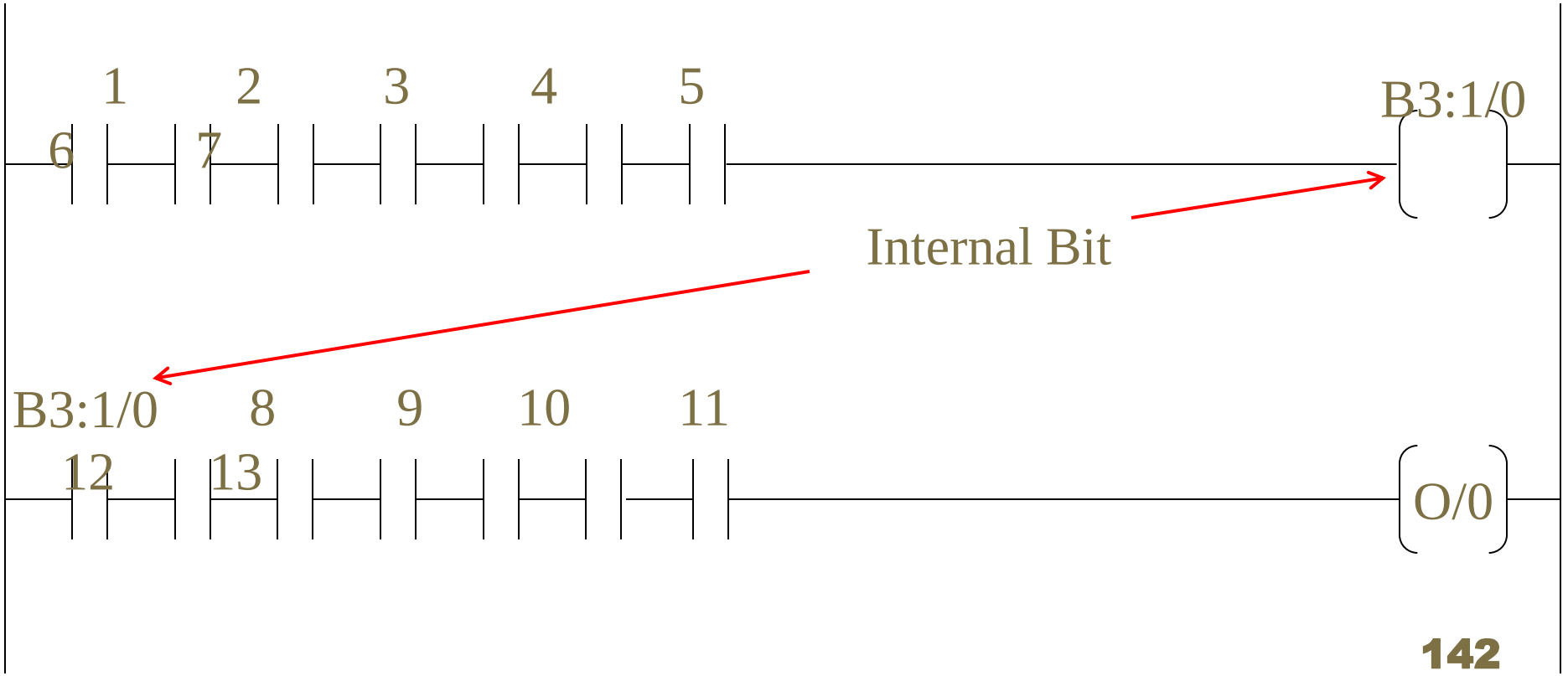
USING THE INTERNAL BIT TO SOLVE HORIZONTAL PROGRAMMING LIMITATIONS

The number of Instructions Exceeds the Horizontal Limit



USING THE INTERNAL BIT TO SOLVE HORIZONTAL PROGRAMMING LIMITATIONS

The number of Instructions can be split into two rungs using an Internal Bit



ACTIVITY: LOGIXPRO DOOR SIMULATION LAB

EXERCISE:

- Door Simulation Lab – Exercises 1 and 2

- **Enjoy your lunch!**

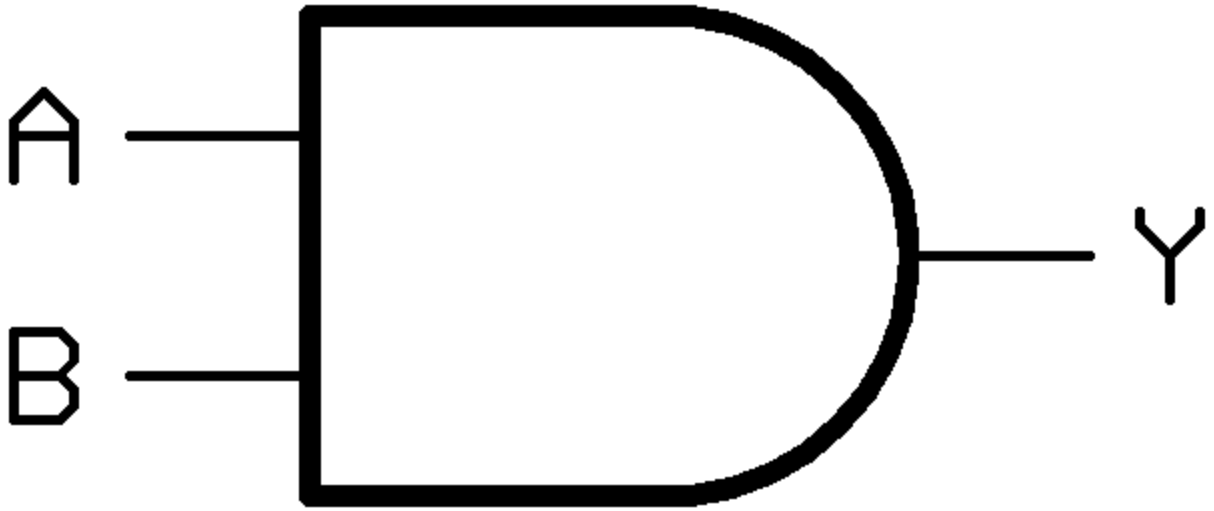
ACTIVITY: LOGIXPRO DOOR SIMULATION LAB

EXERCISE:

- **Door Simulation Lab – Exercises 1 and 2 (Cont'd)**

SESSION IV

BOOLEAN LOGIC



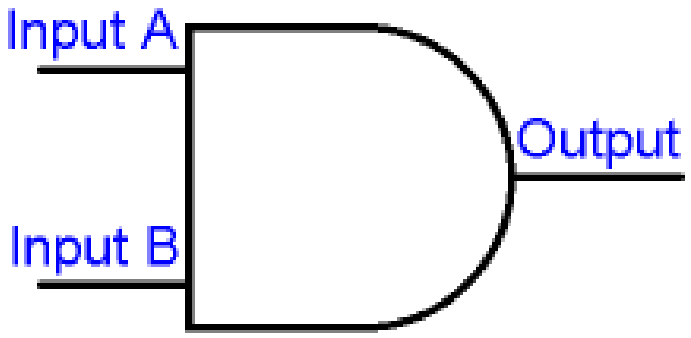
SESSION IV: LEARNING OBJECTIVES

- At the conclusion of this session, participants will understand principles related to:
  - And, Or, and Not functions
  - Variations of And, Or, and Not functions

AND, OR, & NOT FUNCTIONS

- AND, OR, and NOT, are the three fundamental logic functions performed by digital equipment
- Each function has it's own symbol, and a rule that determines the outcome of the logic
- Variations of these functions include the NAND gate, (AND NOT) and the NOR gate, (OR NOT)

THE AND FUNCTION



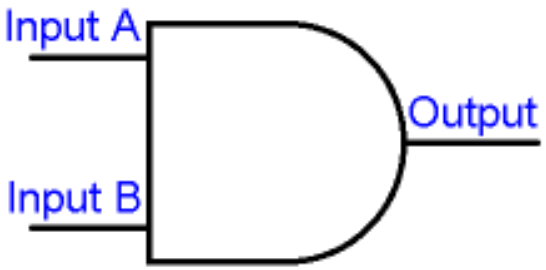
- The AND function, also known as an AND gate is a device where two inputs in series and must be examined on in order for the output to energize

AND TRUTH TABLE

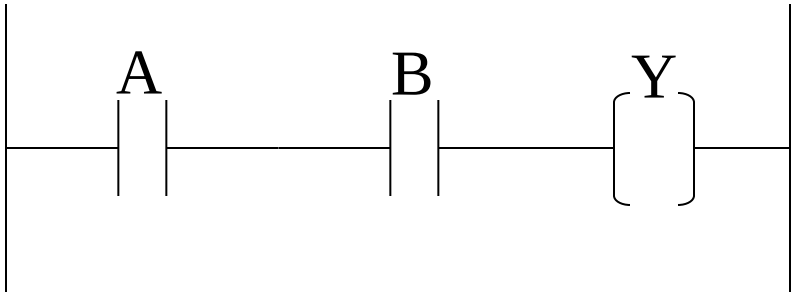
INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

VARIATIONS OF AND

BOOLEAN SYMBOL

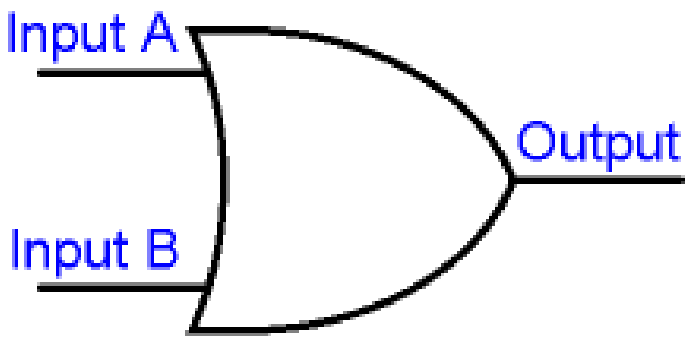


LADDER LOGIC



BOOLEAN ALGEBRA **(A X B) = Y**

THE OR FUNCTION



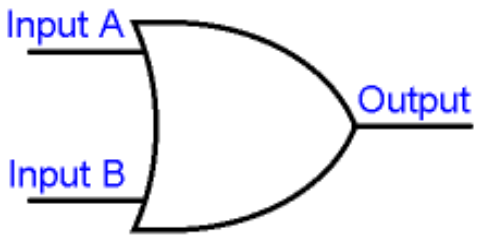
- The OR function, also known as an OR gate is a device where two inputs are in parallel. Any one of the inputs can be true in order for the output to energize

OR TRUTH TABLE

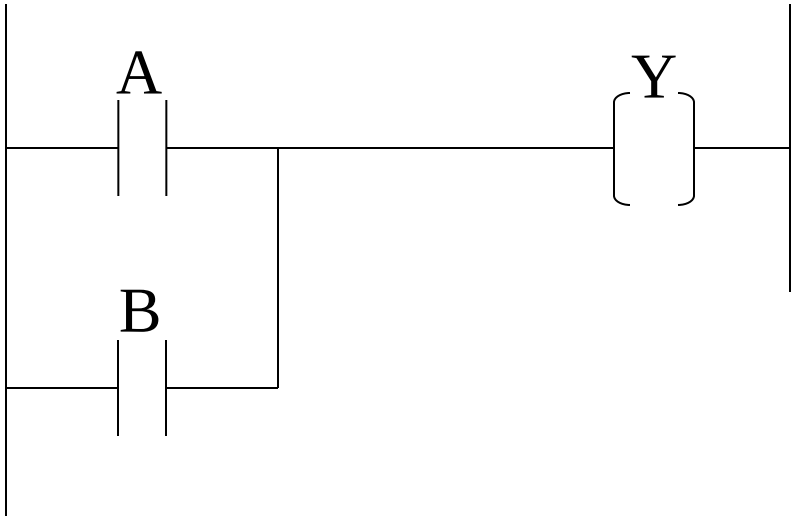
INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Variations of OR

BOOLEAN SYMBOL

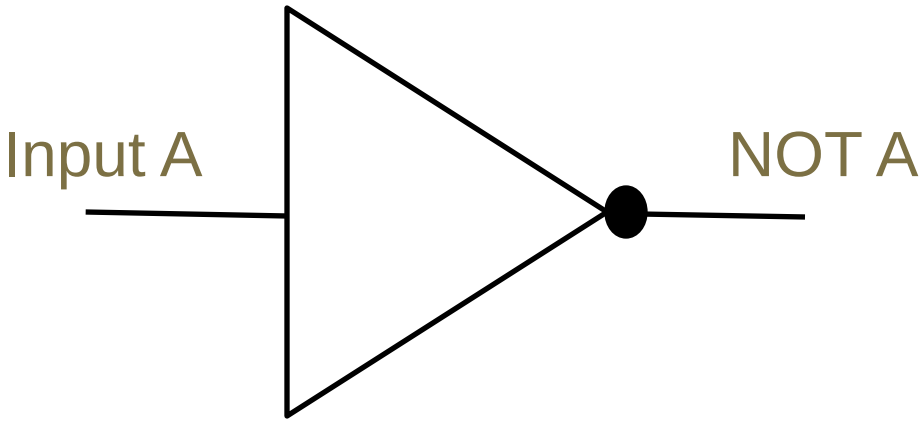


LADDER LOGIC



BOOLEAN ALGEBRA (A + B) = Y

THE NOT FUNCTION



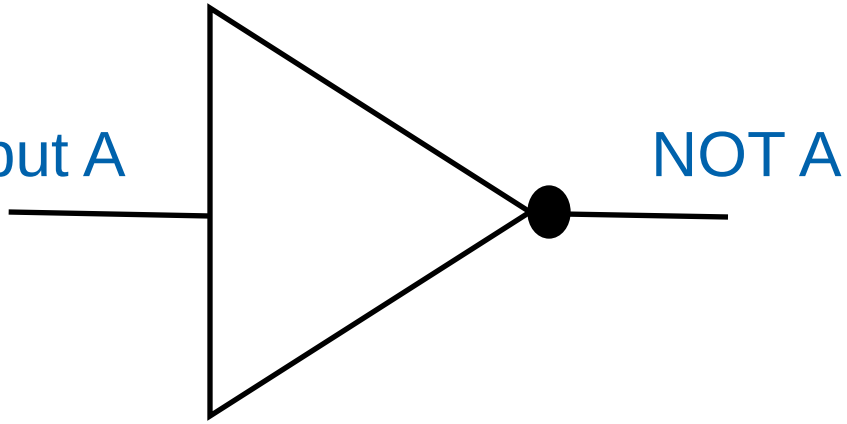
- **The NOT function is mainly used in conjunction with the AND or the OR gate in order to invert the input from a 1 status to a 0 status**

NOT TRUTH TABLE

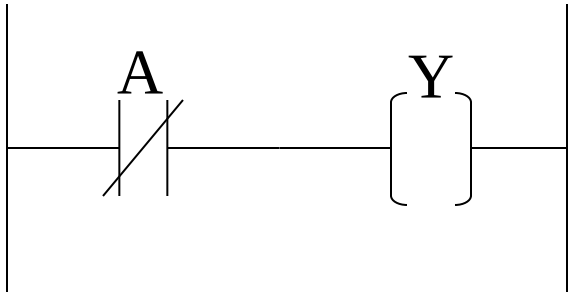
A	NOT A
0	1
1	0

VARIATIONS OF THE NOT FUNCTION

BOOLEAN SYMBOL

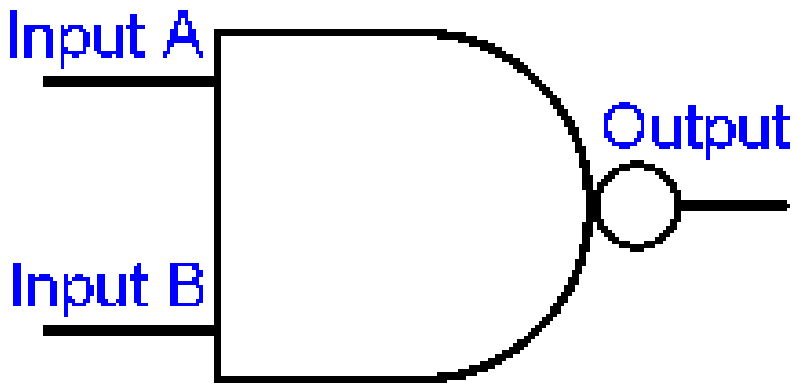


LADDER LOGIC



BOOLEAN ALGEBRA $\overline{A} = Y$

THE NAND GATE



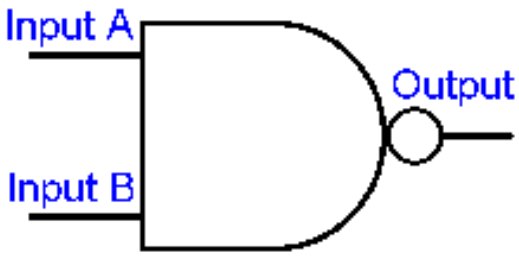
- The NAND function, also known as the NAND gate is a device where one of two inputs in series must examine off in order for the output to energize

NAND TRUTH TABLE

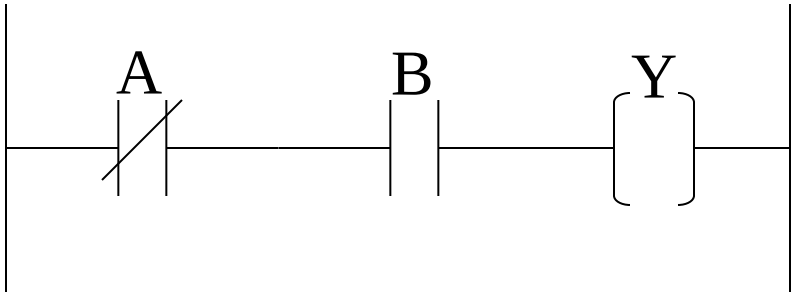
INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

VARIATIONS OF NAND

BOOLEAN SYMBOL



LADDER LOGIC



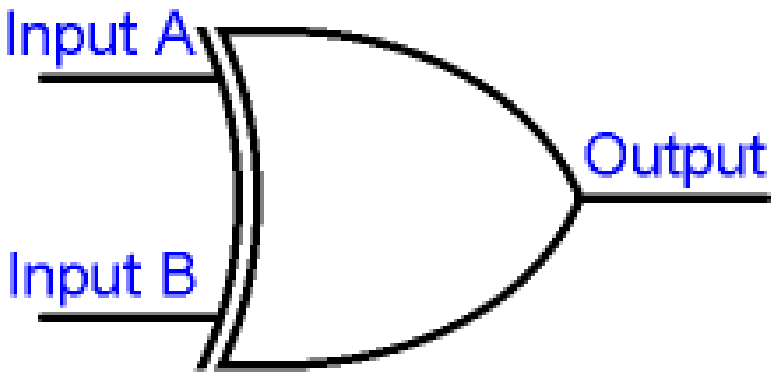
BOOLEAN ALGEBRA

$(\overline{A} \times B) = Y$

$(A \times \overline{B}) = Y$

$\overline{(A \times B)} = Y$

Exclusive OR

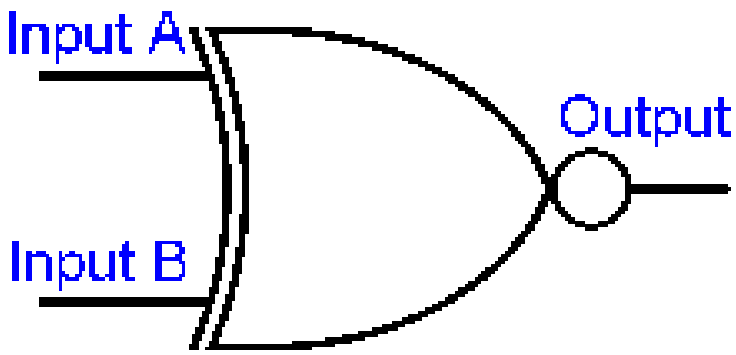


- The exclusive OR (XOR) function, is a device where only one of two inputs in parallel can examine on in order for the output to energize. It is commonly used for the comparison of two binary numbers

(XOR) TRUTH TABLE

INPUTS		OUTPUT
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive NOR



- **The exclusive NOR (XNOR) function, is a device where two inputs in parallel must either examine on, or examine off, in order for the output to energize**

EXCLUSIVE NOR TRUTH TABLE

INPUTS		OUTPUT
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Inputs		Truth Table Outputs for each Gate					
A	B	AND	NAND	OR	NOR	EX-OR	EX-NOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

- **Enjoy your break!**

ACTIVITY: LOGIXPRO SILO LAB

EXERCISE:

- **The Silo Lab Utilizing Relay Logic – Exercises 1 and 2**

AGENDA

RECAP

SESSION IV:

- Activity: LogixPro Silo Lab (Cont'd)

SESSION V:

- PLC Timers
- Activity: LogixPro Introduction to Timers Lab
- Break
- Activity: LogixPro Introduction to Timers Lab (Cont'd)

LUNCH

AGENDA

SESSION VI:

- PLC Counters
- Activity: LogixPro Introduction to Counters Lab
- Break
- Activity: LogixPro Batch Mixing Lab

REVIEW AND REFLECTION

PLC LOGIC SCHEMATICS FINAL ASSESSMENT

ACTIVITY: LOGIXPRO SILO LAB

EXERCISE:

- **The Silo Lab Utilizing Relay Logic – Exercises 1 and 2 (Cont'd)**

SESSION V

INTRODUCTION TO PLC TIMERS

SESSION V: LEARNING OBJECTIVES

- At the conclusion of this session, participants will understand principles related to:
  - Timer instructions
  - Timer quantities
  - Calculating timer duration
  - Timer bits
  - Timer types

TIMER INSTRUCTIONS

- PLC timers are the most commonly used PLC instruction, after coils and contacts
- PLC timer instructions provide the same functions as on-delay and off-delay timers
- PLC timers offer several advantages over real world mechanical and electronic timers

TIMER ADVANTAGES

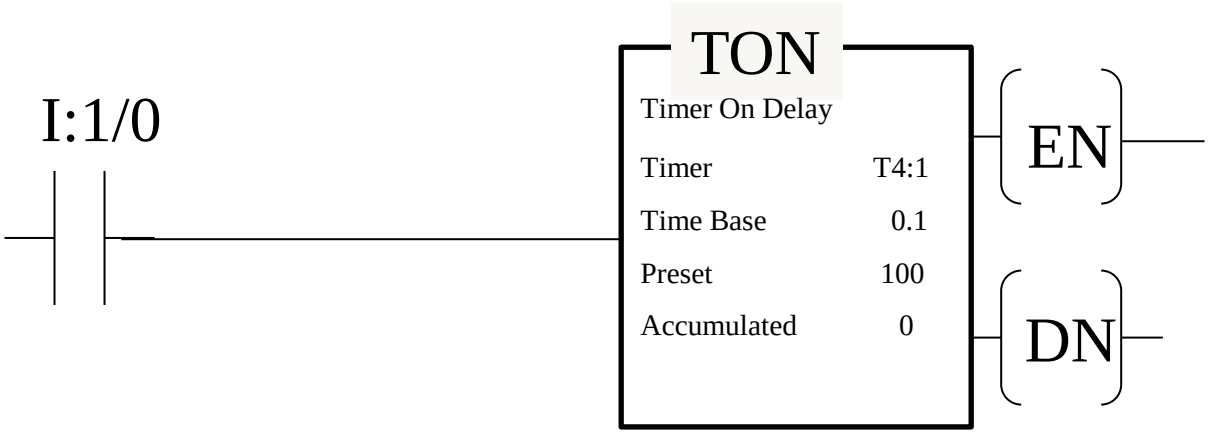
- Time setting can be easily set or changed
- Timers can be added or removed from circuit through the use of programming rather than wiring changes
- Timer accuracy and repeatability are extremely high

TIMER QUANTITIES

- **Preset Time:** This represents the time duration for the timing circuits
- **Accumulated Time:** This represents the amount of time that has elapsed from the moment the timing circuit began
- **Time Base:** The time base is a preset interval that the timer count. Typical time bases are: 1, 0.1 and 0.01

CALCULATING TIMER DURATION

To calculate the duration of a timer multiply the preset value by the time base

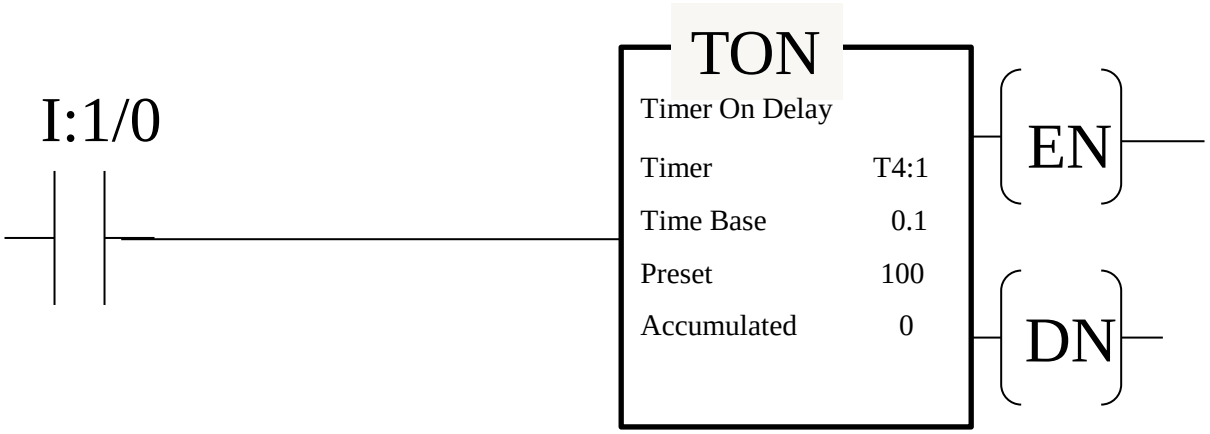


The above timers duration is set for 10 seconds

100 x 0.1 = 10

CALCULATING TIMER DURATION

What is the duration of the following timer?

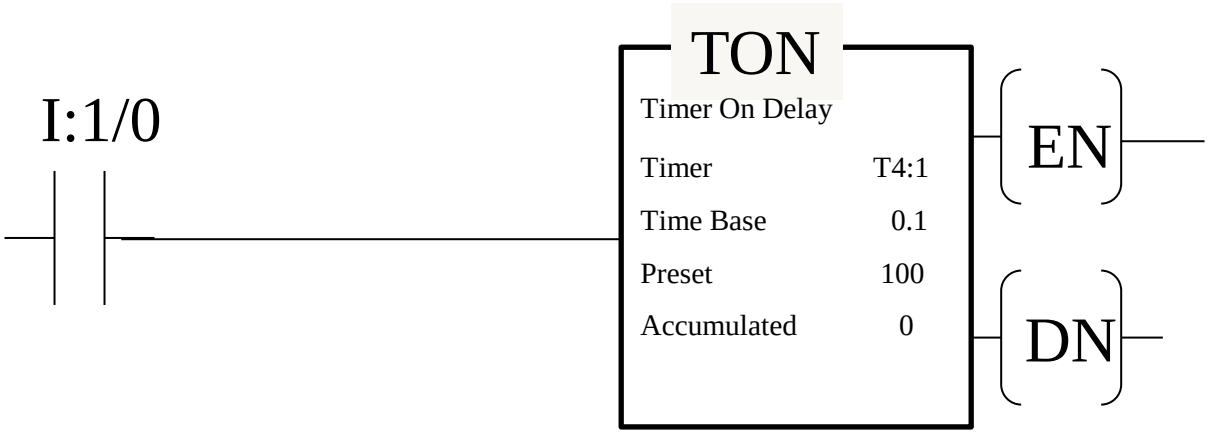


The above timers duration is set for 1 second

100 x 0.01 = 1

CALCULATING TIMER DURATION

What is the duration of the following timer?



The above timers duration is set for 30 seconds

300 x 0.1 = 30 seconds

TIMER BITS

- **Enable Bit:** The Enable Bit, (EN) is true when ever the timer instruction is true
- **Timer -Timing Bit:** The Timer-Timing Bit, (TT) is true whenever the accumulated value of the timer is changing, meaning the timer is timing
- **Done Bit:** The Done Bit, (DN) changes state whenever the accumulated value equals the preset value. Its state depends on the type of timer being used

TIMER TYPES

IN GENERAL, THERE ARE THREE TYPES OF PLC TIMERS

- On-delay Timer: (TON)
- Off-delay Timer: (TOF)
- Retentive Timer: (RTO)

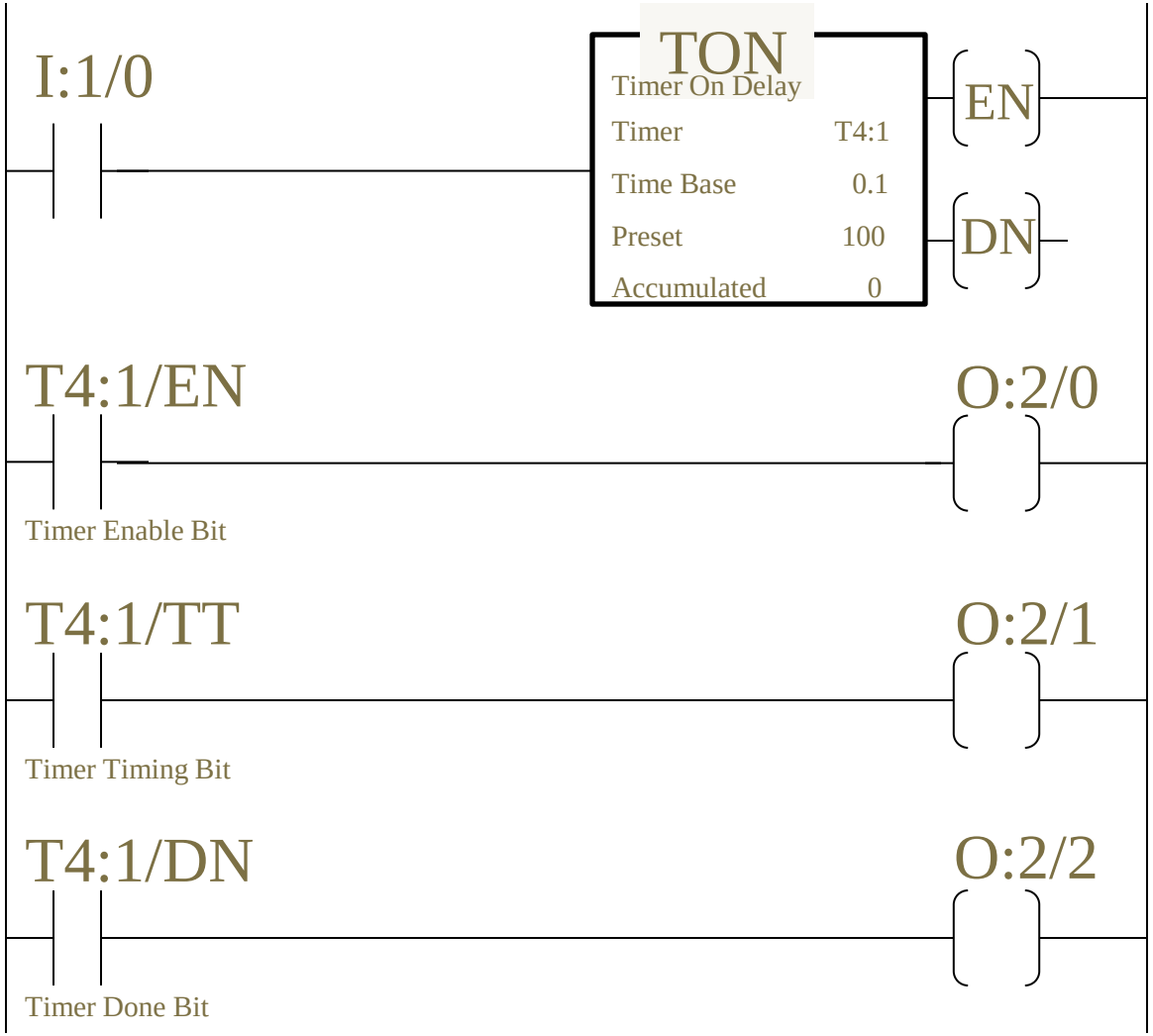
ON-DELAY TIMER INSTRUCTIONS

- The On-Delay timer counts time-based intervals when the rung instruction is true
- When the rung instruction containing the timer become true, the timer time out period begins

ON-DELAY TIMER INSTRUCTIONS

- When timer time out period ends,(Preset Value equals the Accumulated Value), an output is made true
- If the rung goes false before the time out period ends, the accumulated value will reset

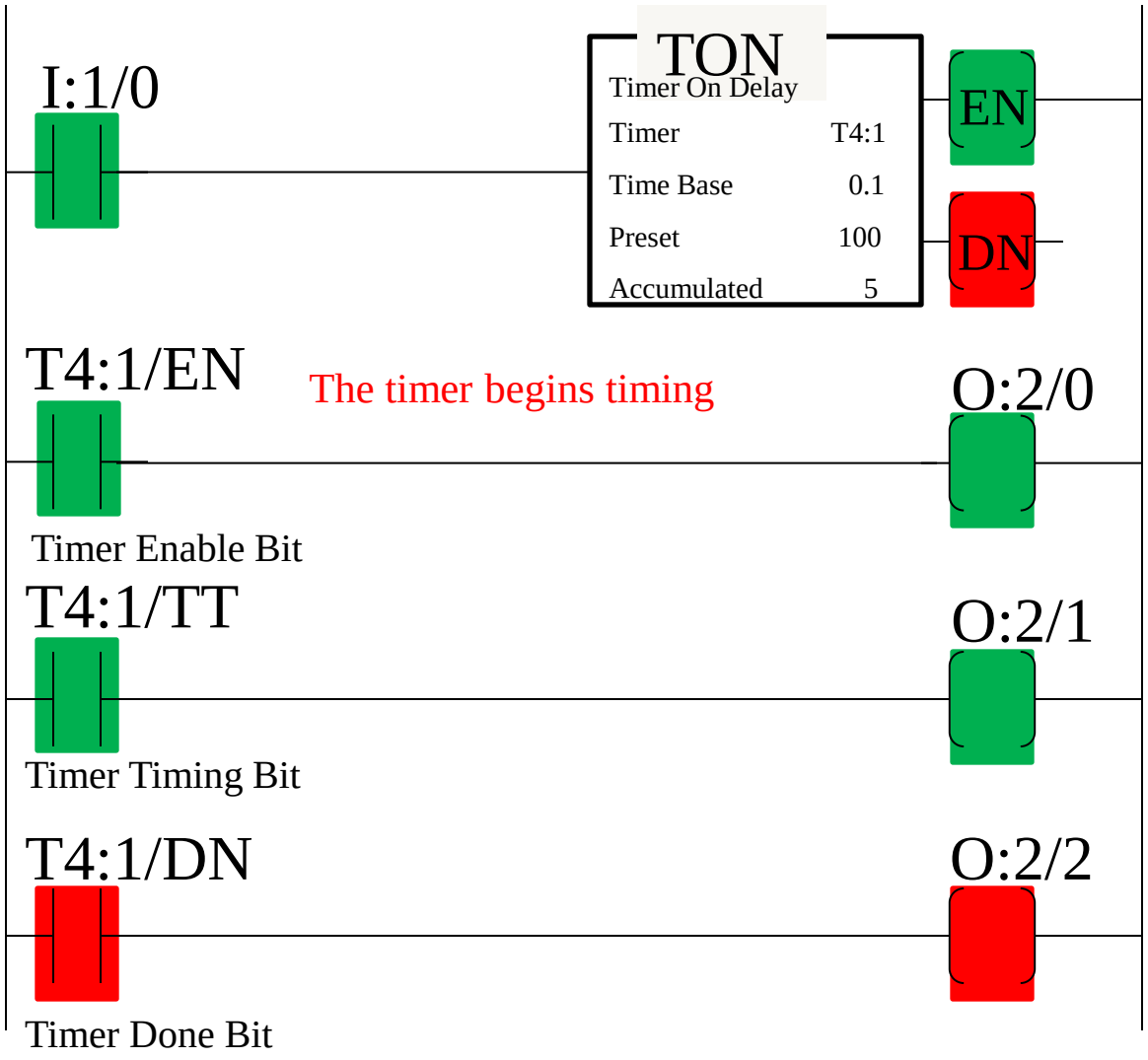
ON DELAY TIMERS



ON DELAY TIMERS

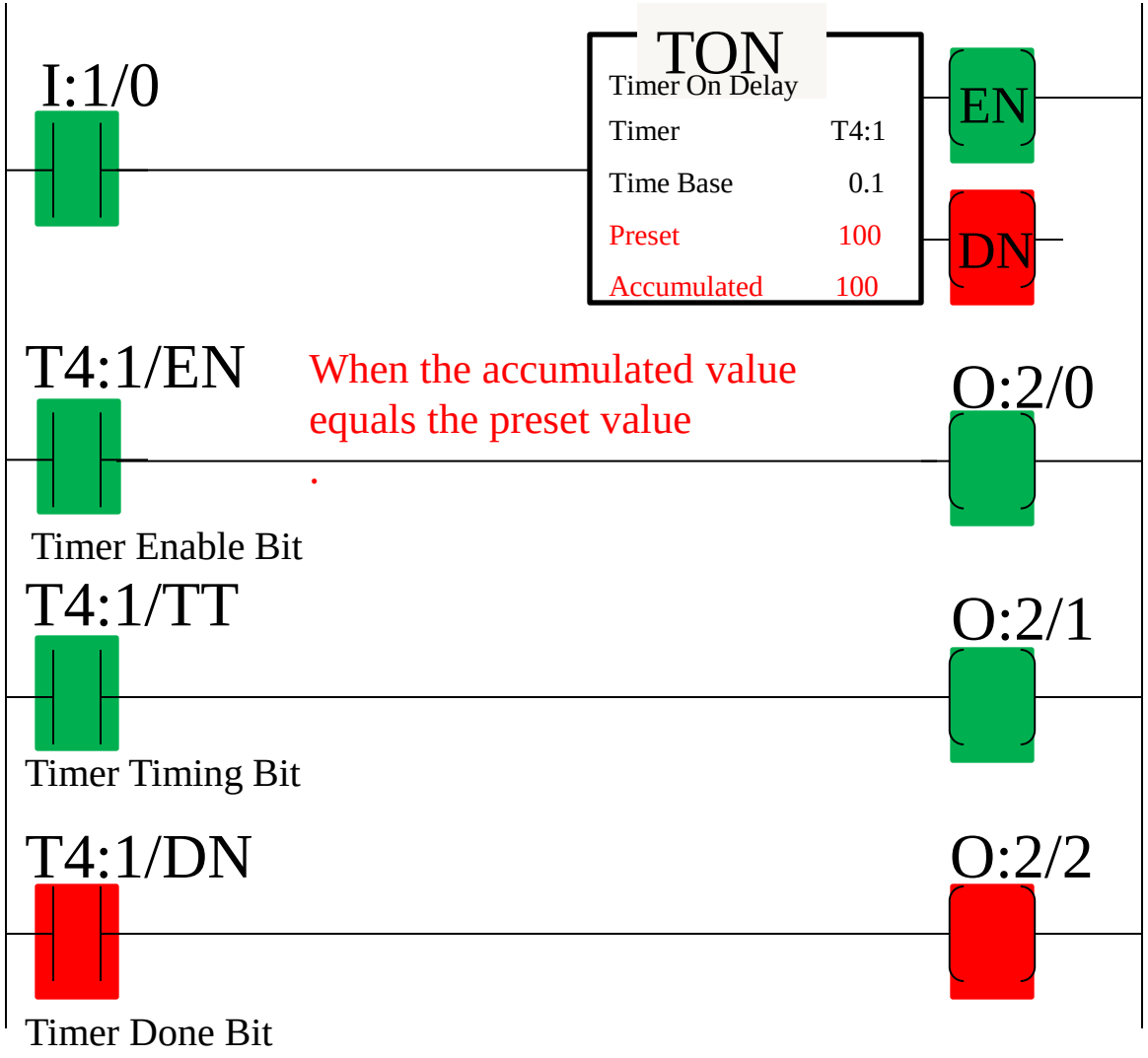
WHEN THE TIMER RUNG IS TRUE:

- **The Enable bit is true**
- **The Timer Timing bit is also true**
- **The Done bit is false**



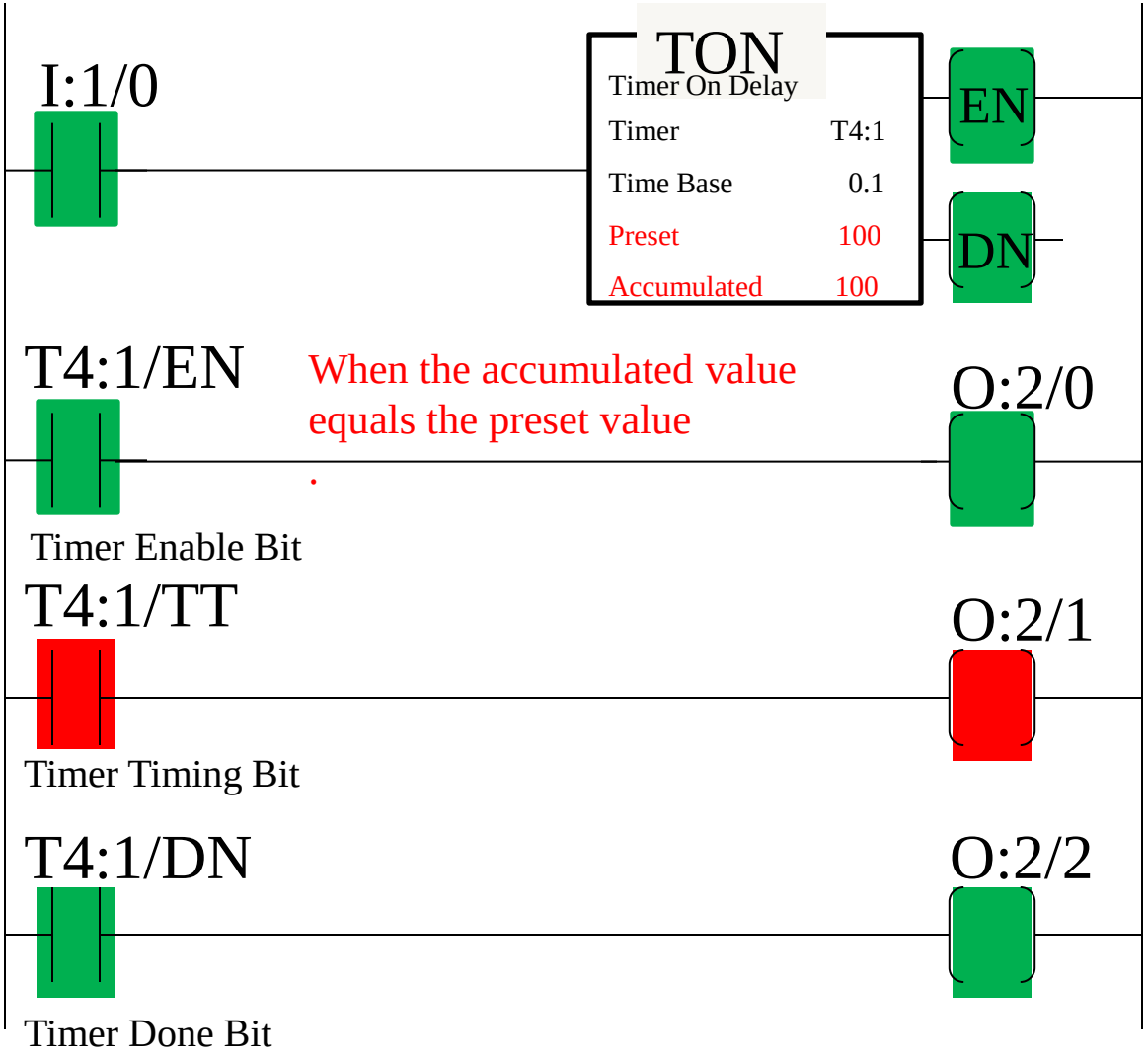
ON DELAY TIMERS

- **The Enable bit remains true**



ON DELAY TIMERS

- **The Enable bit remains true**
- **The Timer Timing bit goes false**
- **The Done bit is true**



OFF-DELAY TIMER INSTRUCTIONS

- The Off-Delay timer will keep an output energized for a set period of time after the rung containing the timer has gone false
- When the rung instruction containing the timer becomes true, the Done Bit (DN) of the timer becomes true

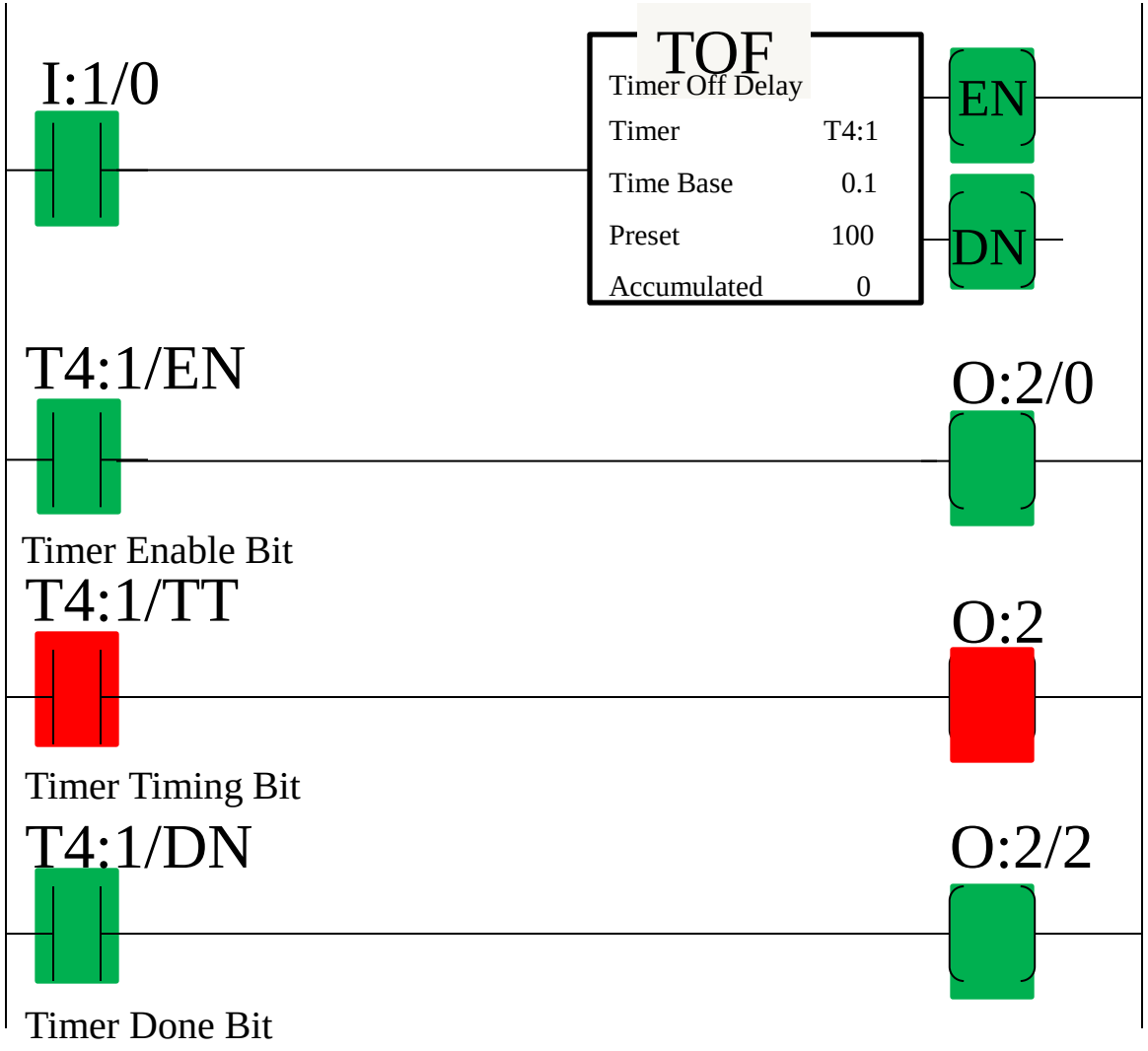
OFF-DELAY TIMER INSTRUCTIONS

- When timer time out period ends,(Preset Value equals the Accumulated Value), the Done Bit goes false
- If the rung goes true before the time out period ends, the accumulated value will reset to 0

OFF DELAY TIMERS

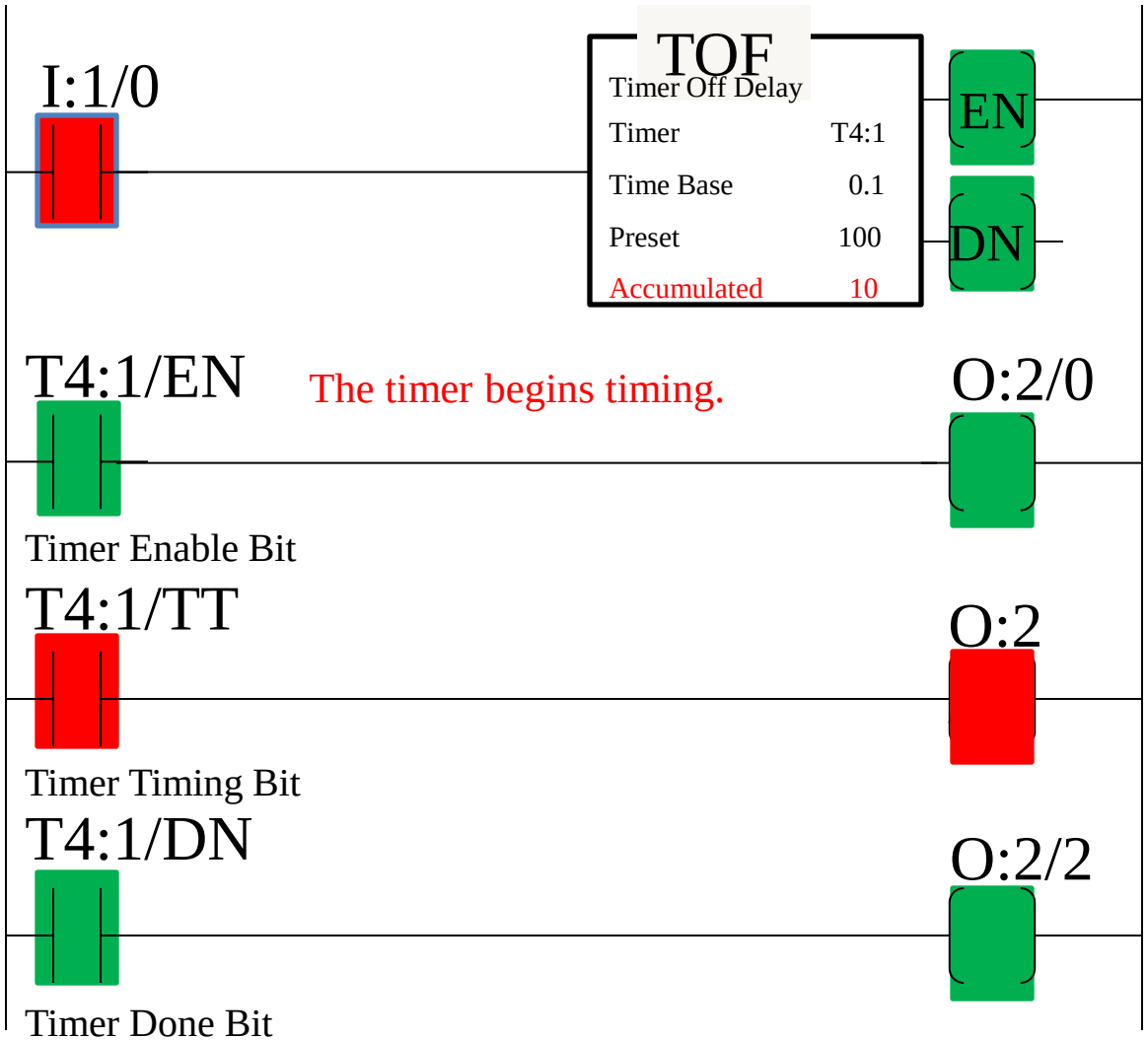
WHEN THE TIMER RUNG IS TRUE:

- **The Enable bit is also true**
- **The Timer Timing bit is false**
- **The Done Bit becomes true**



OFF DELAY TIMERS

WHEN THE TIMER RUNG TRANSITIONS FROM TRUE TO FALSE:

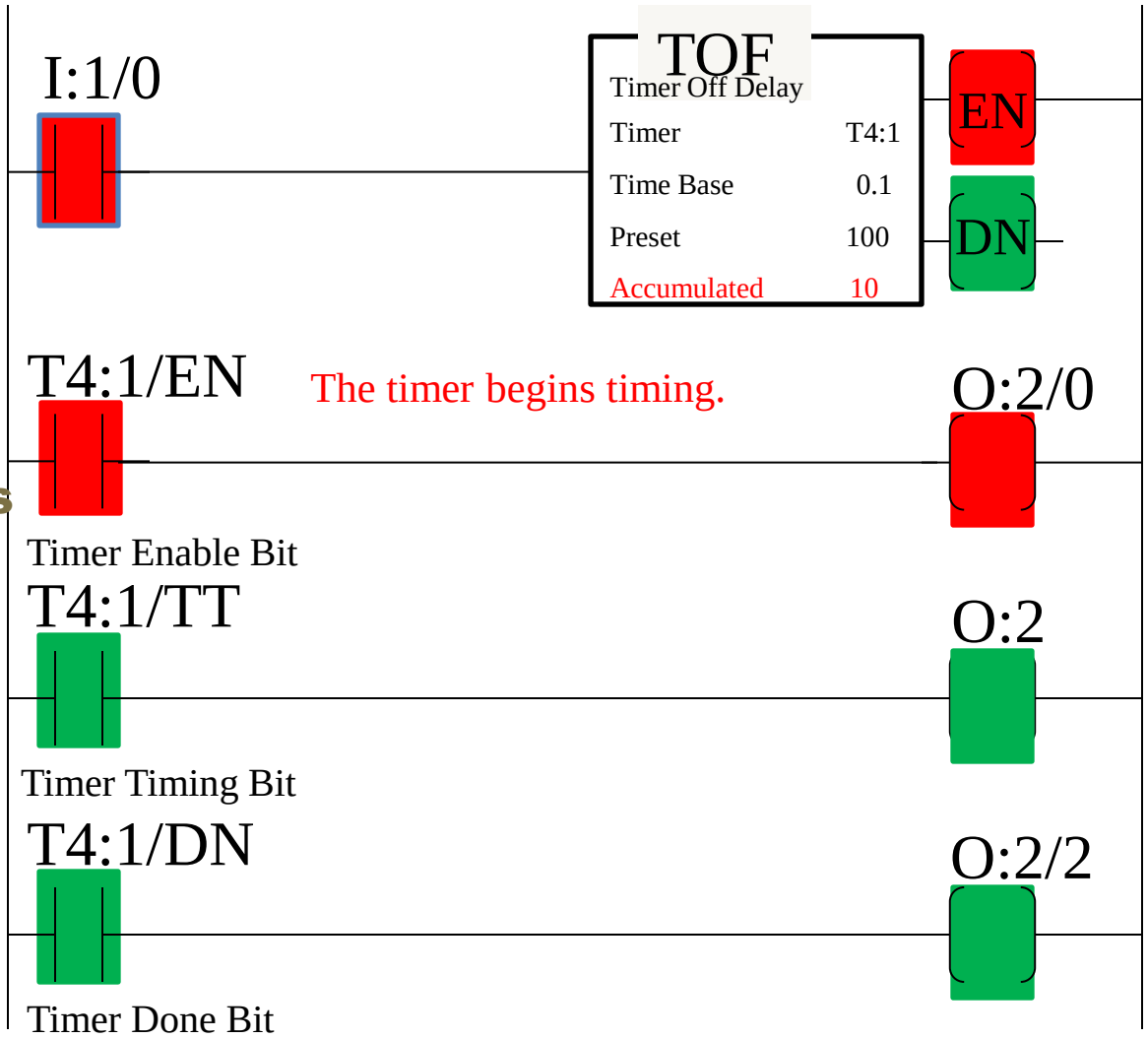


- **The Done Bit remains true**

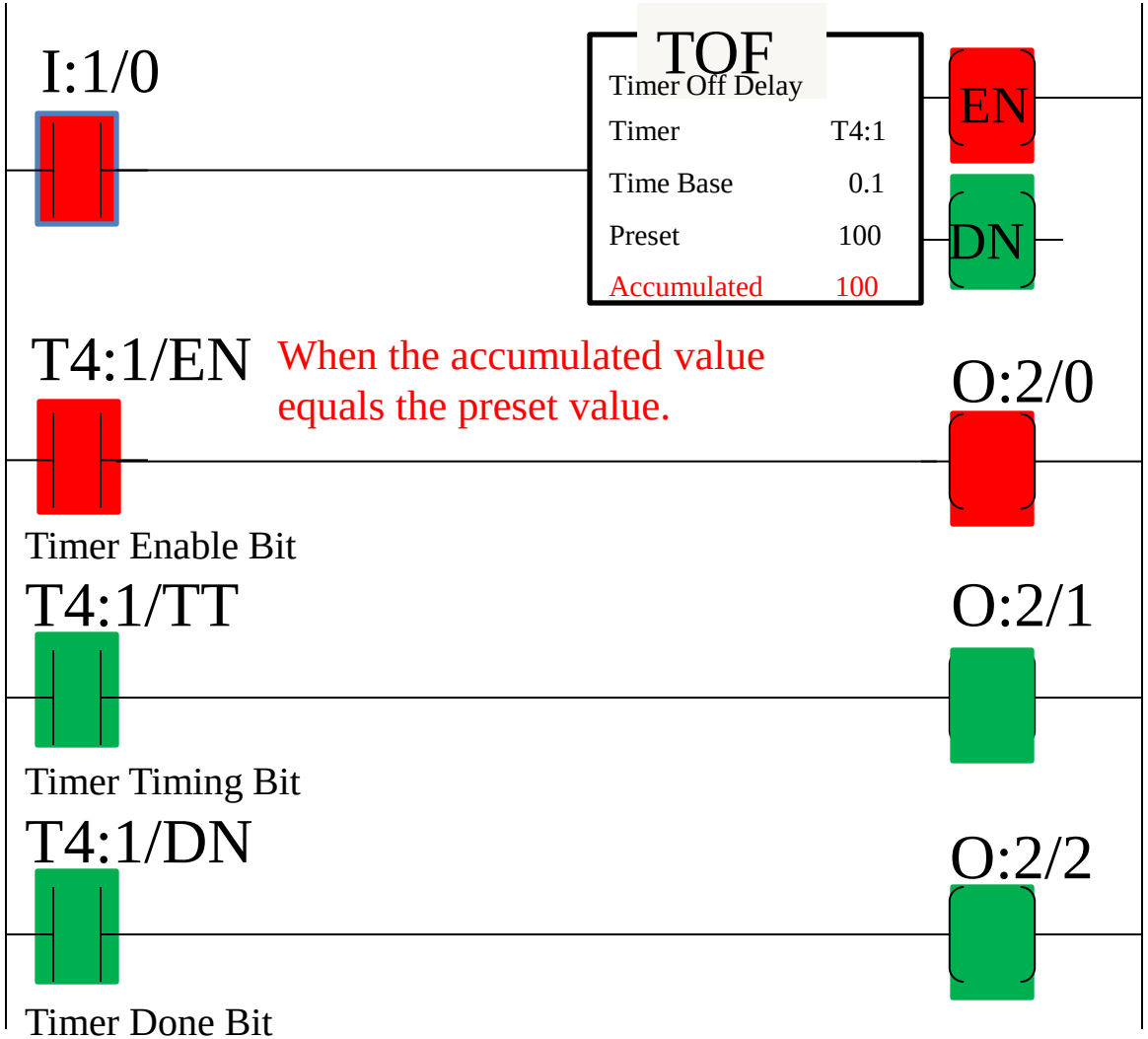
OFF DELAY TIMERS

WHEN THE TIMER RUNG TRANSITIONS FROM TRUE TO FALSE:

- **The Enable bit becomes false**
- **The Timer Timing bit becomes true**
- **The Done Bit remains true**

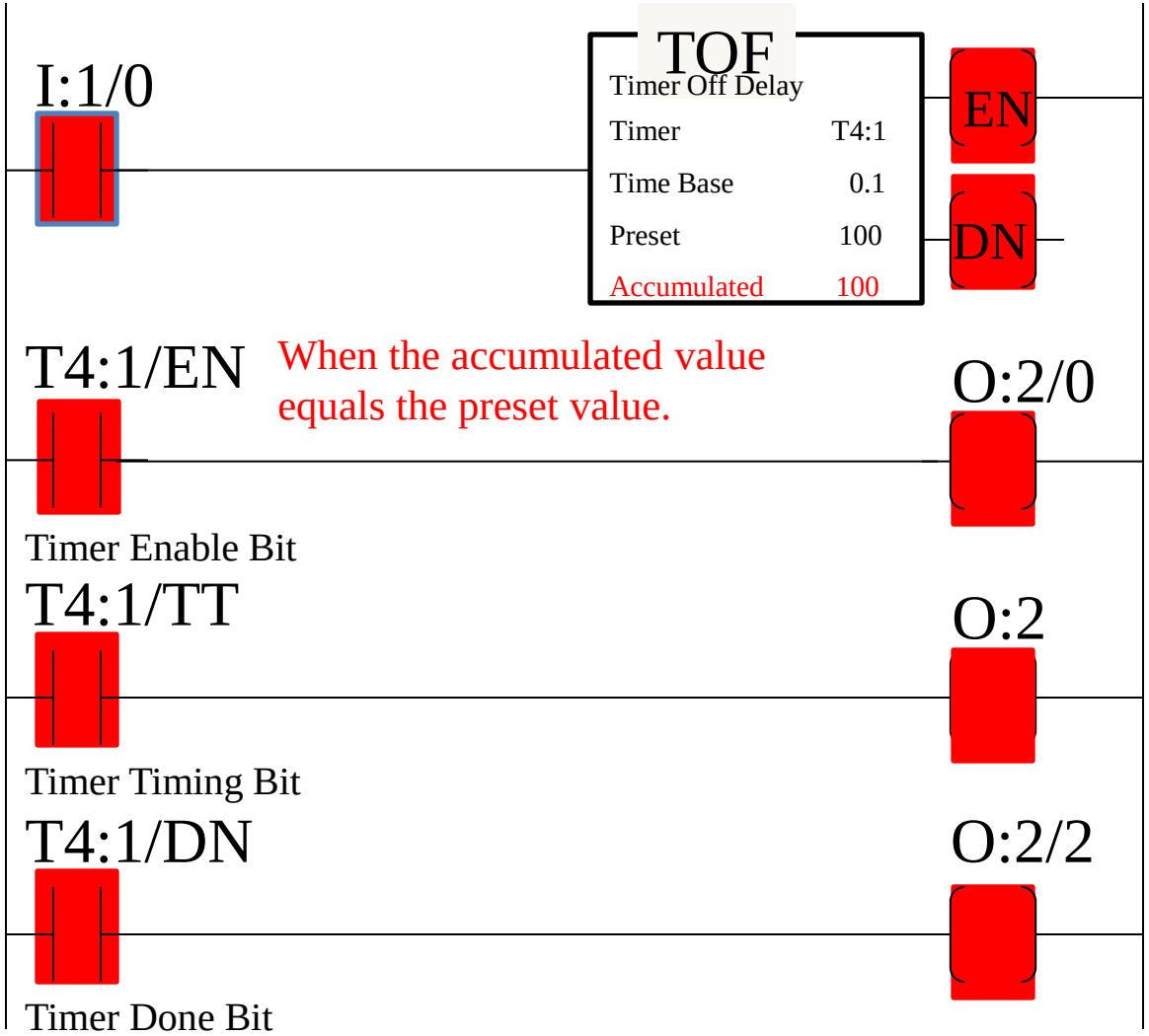


OFF DELAY TIMERS



OFF DELAY TIMERS

- **All bits go false**



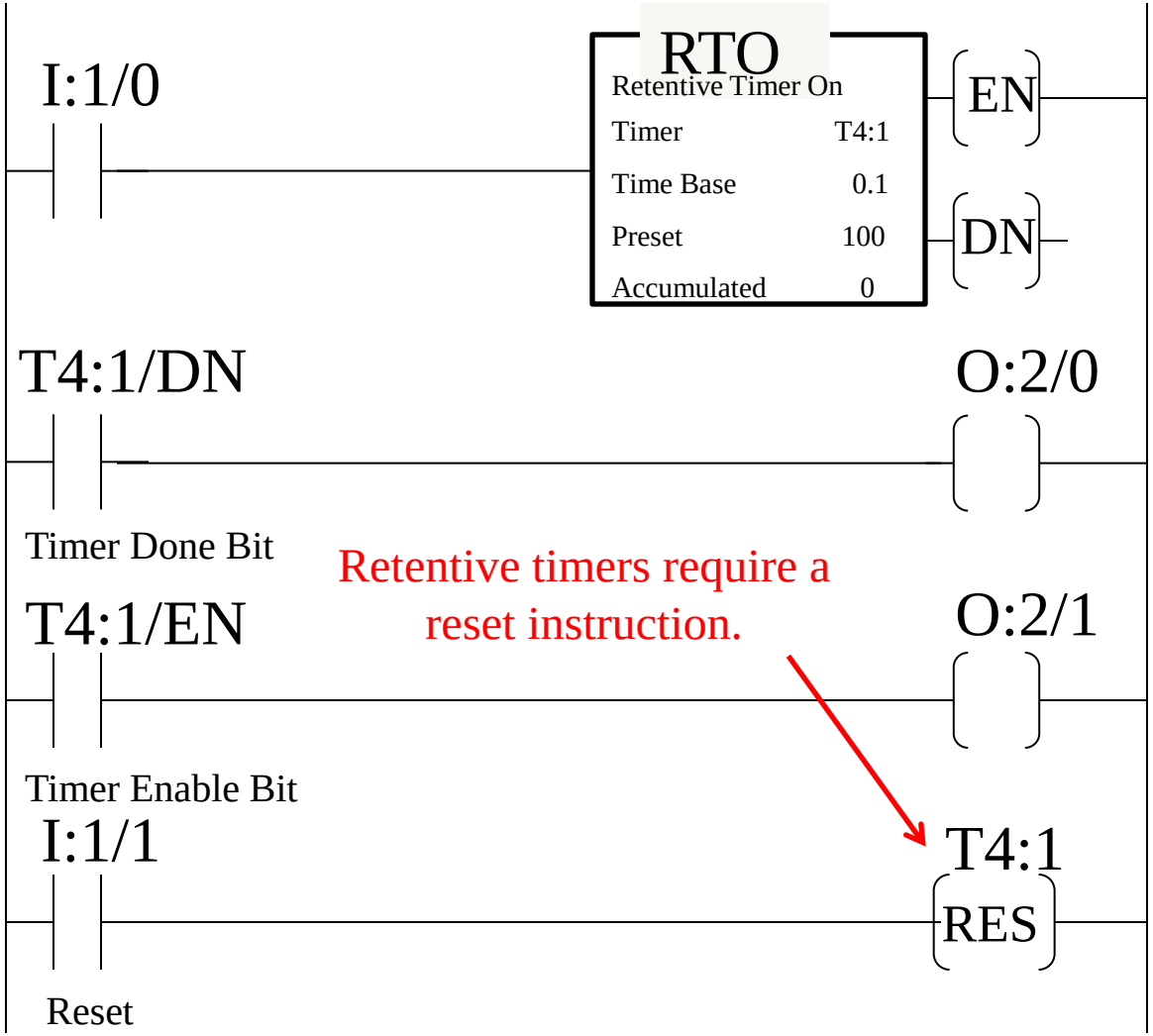
RETENTIVE TIMER INSTRUCTIONS

- The bits of a Retentive timer, act much the same as the bits for an On-Delay timer
- However when the rung instruction containing the timer transitions from true to false, the accumulative value is not reset

RETENTIVE TIMER INSTRUCTIONS

- Loss of power to the timer after it reaches its preset value does not effect the state of the bits
- A Retentive timer must be intentionally reset

RETENTIVE TIMERS



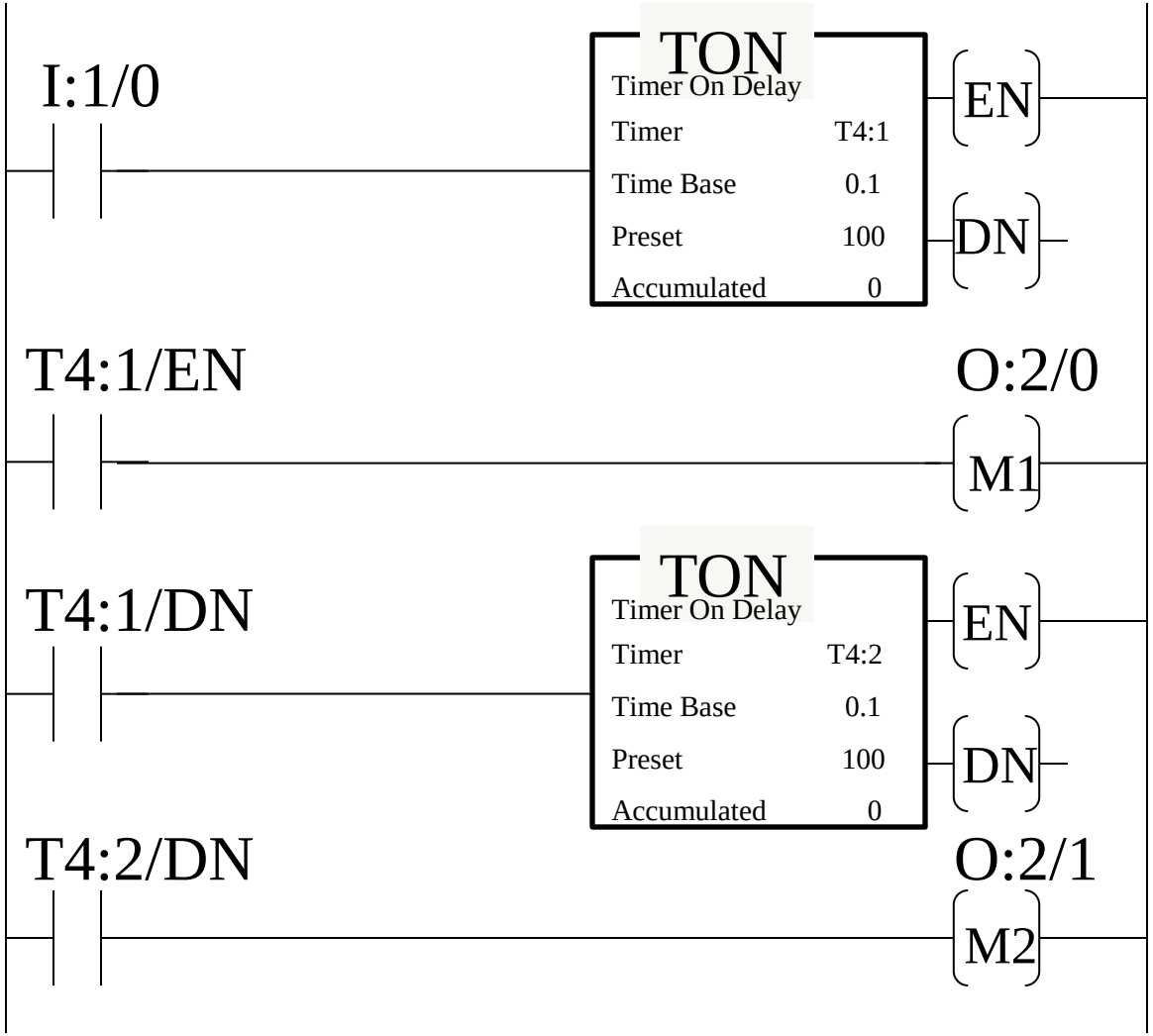
CASCADING TIMERS

- The programming of two or more timers together is called Cascading
- Timers can be interconnected to solve a variety of logic control functions

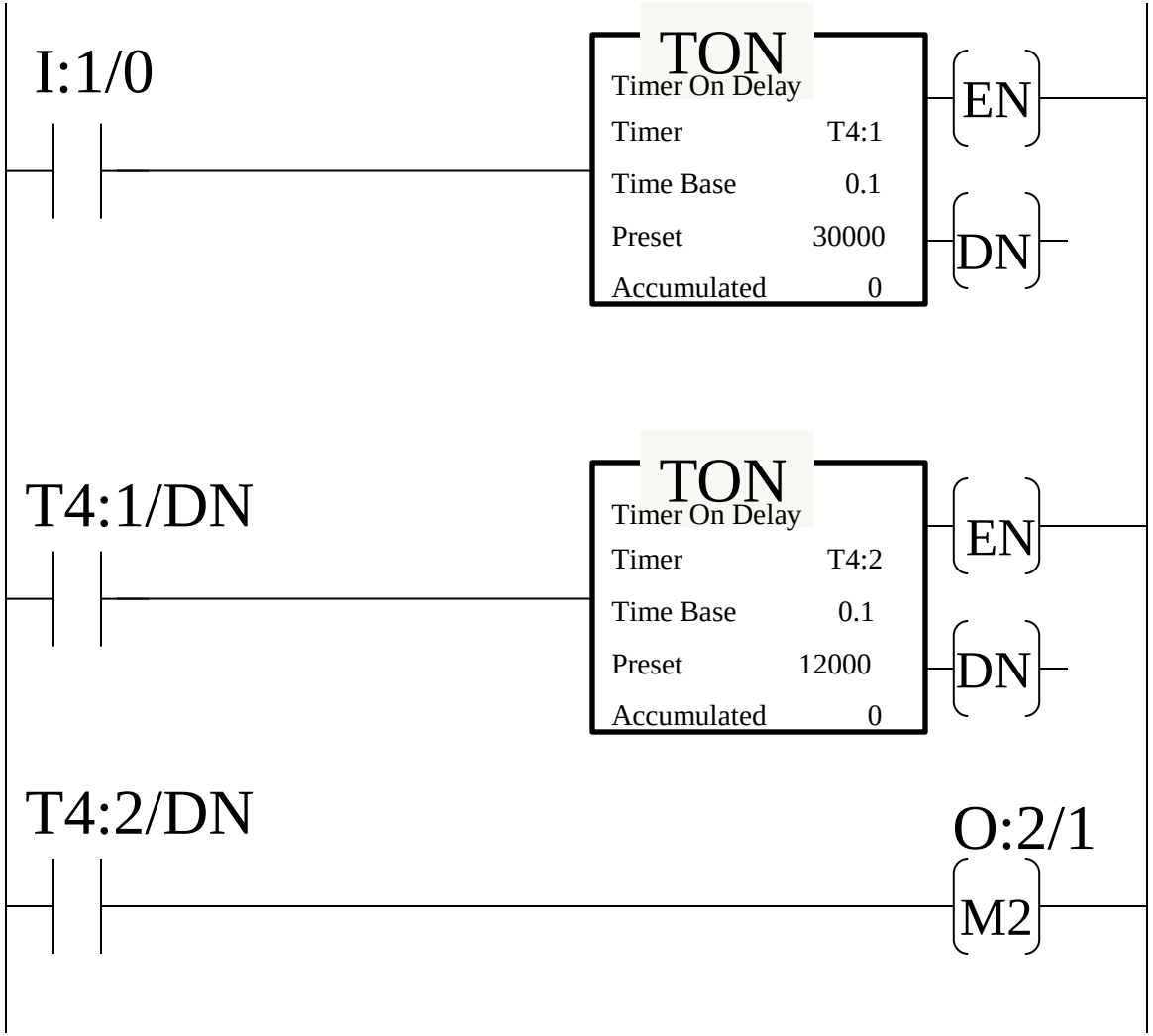
CASCADING TIMERS

- Cascading timers can allow you to exceed the maximum preset time of a single timer
- The following slide shows how two motors can be started in sequence with a 10 second time delay using Cascading timers

SEQUENCE STARTING



EXTENDING TIME DELAY



ACTIVITY: LOGIXPRO INTRODUCTION TO TIMERS LAB

EXERCISE:

- **Introduction to RSLogix Timers**

- **Enjoy your break!**

ACTIVITY: LOGIXPRO INTRODUCTION TO TIMERS LAB

EXERCISE:

- **Introduction to RSLogix Timers (Cont'd)**
- **Traffic control utilizing TON timers – Exercises 1 and 2**

- **Enjoy your lunch!**

SESSION VI

INTRODUCTION TO PLC COUNTERS

SESSION VI: LEARNING OBJECTIVES

- At the conclusion of this session, participants will understand principles related to:
  - Counter instructions
  - Counter quantities
  - Counter bits
  - Counter types

COUNTER INSTRUCTIONS

- PLC counters can serve the same function as real world counters
- All PLC manufactures offer some from of counter instructions
- PLC counters are similar to timers except they do not operate on an internal clock but are dependent on external inputs for counting

COUNTER TYPES

- There are two types of counters, Up-Counters and Down Counters
- The Up-Counter increments up by 1 every time the counter rung transitions from false to true

COUNTER TYPES

- The Down-Counter decrements down by 1 every time the counter rung transitions from false to true
- Both the Up-Counter and Down Counter are retentive so they require a reset instruction

COUNTER QUANTITIES

- **Preset Value:** Specifies the number the counter must reach before it changes the state of the done bit. It can range from -32,768 to +32,767
- **Accumulated Value:** Represents the current count of the counter. It can either increment up (CTU) or decrement down (CTD). It has the same range as the preset value

COUNTER BITS

- **Count-Up (CU) Enable Bit:** The count-up enable bit is true whenever the count-up counter instruction is true. If the rung instruction is false the enable bit is false

COUNTER BITS

- **Count-Down (CD) enable Bit:** The count-down enable bit is true whenever the Count-Down counter instruction is true. If the rung instruction is false the enable bit is false
- **Done (DN) Bit:** The done bit is true when the accumulated value is equal or greater then the preset value

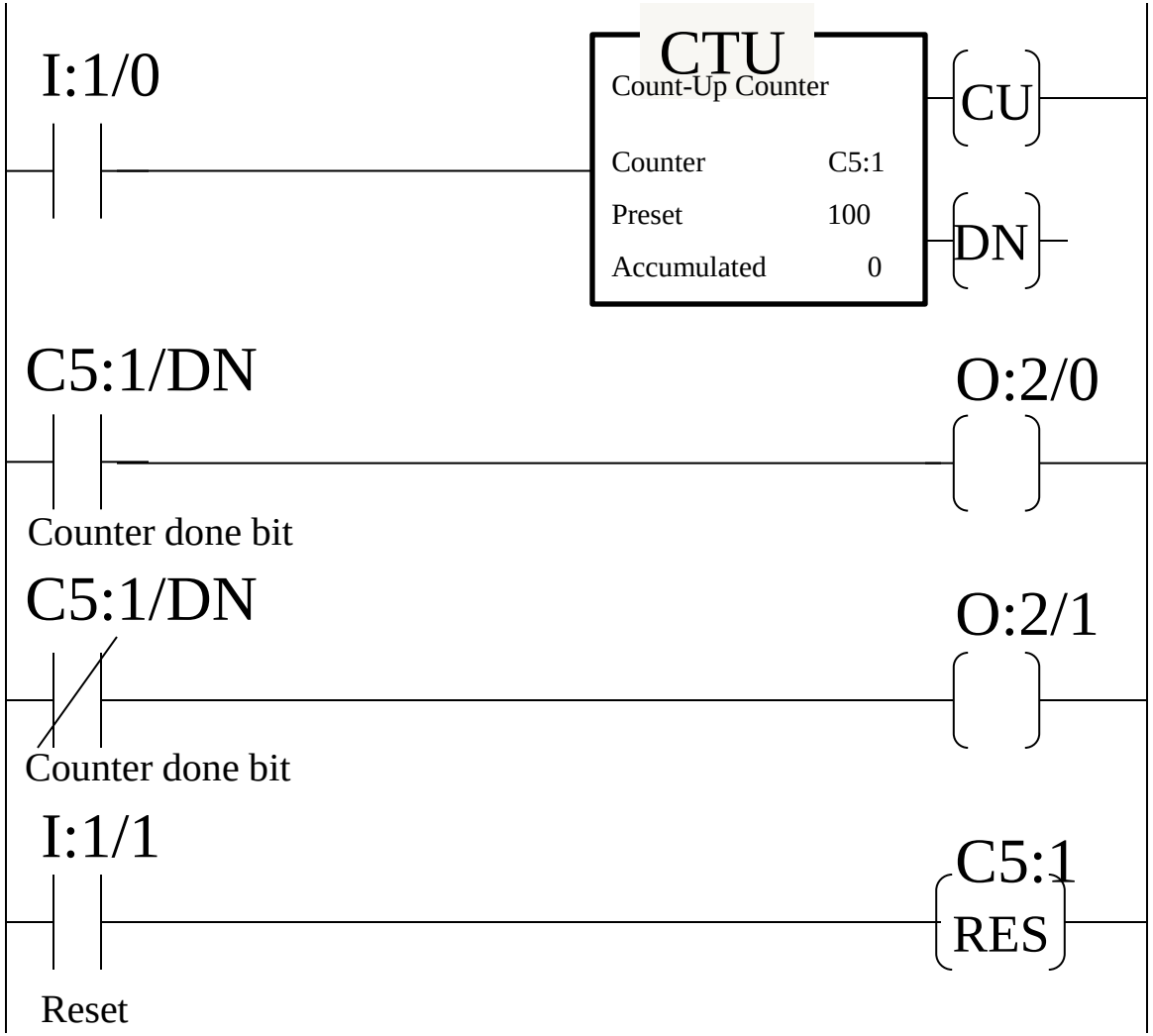
COUNTER BITS

- **Overflow (OV) Bit:** The overflow bit is true whenever the counter counts past its maximum value of 32,767. When the limit is reached the count wraps around to -32,767 and begins to count up from there

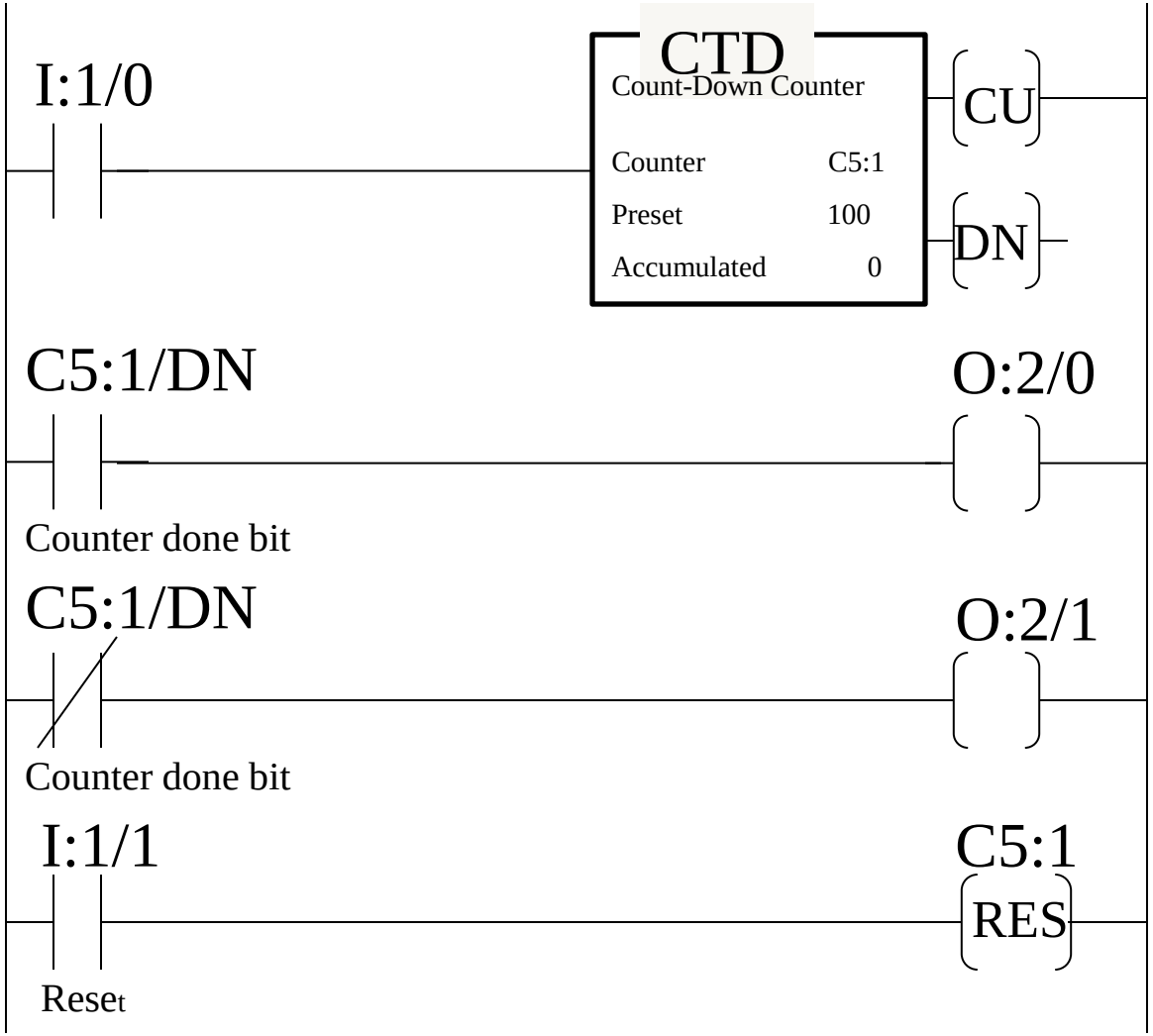
COUNTER BITS

- **Underflow (UN) Bit:** The overflow bit is true whenever the counter counts past its minimum value of -32,767. When the limit is reached the count wraps around to +32,767 and begins to count down from there

UP-COUNTER


I:1/0
CTU
Count-Up Counter
Counter C5:1
Preset 100
Accumulated 0
CU
DN
C5:1/DN
O:2/0
Counter done bit
C5:1/DN
O:2/1
Counter done bit
I:1/1
C5:1
RES
Reset

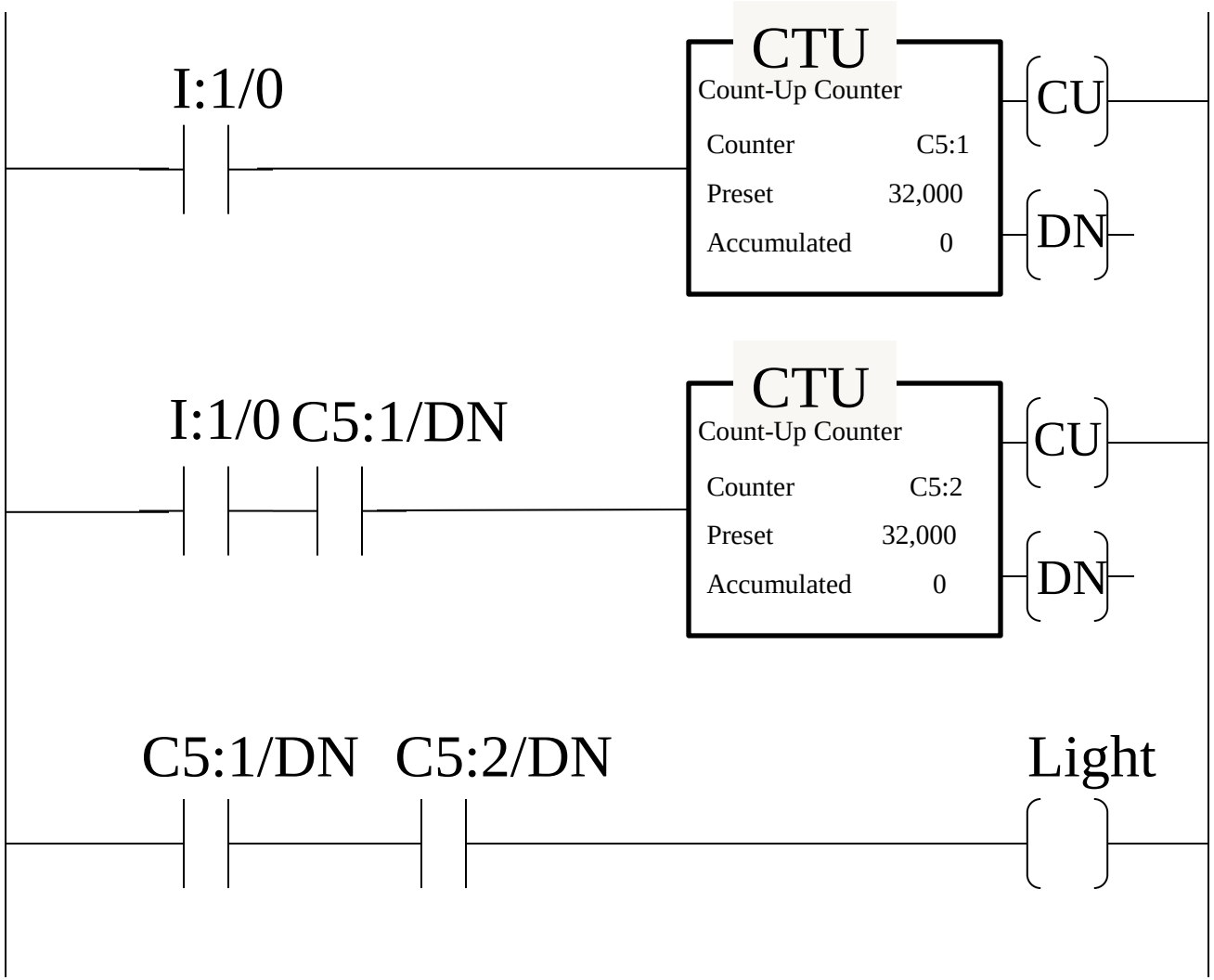
DOWN-COUNTER



CASCADING COUNTERS

- The programming of two or more counters together is called Cascading
- Cascading counters can allow you to exceed the maximum preset count of a single counter
- The following slide shows how two counter can be used to exceed the maximum value of 32,767

CASCADING COUNTERS



ACTIVITY: LOGIXPRO INTRODUCTION TO COUNTERS LAB

EXERCISE:

- **Introduction to RSLogix Counters**

- **Enjoy your break!**

ACTIVITY: LOGIXPRO BATCH MIXING LAB

EXERCISE:

- **Batch mix exercises utilizing PLC Counters**

QUESTIONS AND COMMENTS

FINAL: CONCLUDING THE COURSE

- Concluding the course
- Final Assessment
- Instructor and Course Evaluation

FINAL ASSESSMENT