BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KINH TẾ KỸ THUẬT BÌNH DƯƠNG
A53- Đại lộ Bình Dương-P.Hiệp Thành-TX.Thủ Dầu Một –T.Bình Dương
☎: (0650)822847 – Fax: (0650)825992
Website:http://www.ktkt.edu.vn
KHOA: KỸ THUẬT- CÔNG NGHỆ
BỘ MÔN ĐIỆN- ĐIỆN TỬ

BÀI GIẢNG MÔN HỌC

LÝ THUYẾT PLC

LƯU HÀNH NỘI BỘ

BIÊN SOẠN: THS. NGÔ SỸ

BÌNH DƯƠNG 09/2009

BÀI 1:

TỔNG QUAN VỀ PLC (Programmable Logic Controller)

I. Giới thiệu

1/ Hệ thống điều khiển là gì?

Hệ thống điều khiển là tập hợp tất cả các thiết bị và dụng cụ điện tử. Nó được dùng để vận hành một quá trình hoặc một hoạt động chế tạo một cách ổn định, chính xác và thông suốt. Nó hoạt động dưới bất kỳ phương thức nào và khác nhau trong phạm vi của thiết bị, từ cung cấp năng lượng đến một thiết bị bán dẫn.

Ngày nay, việc tăng nhanh công nghệ cũng như nhu cầu tự động hóa rất cao, đặc biệt là trong công nghiệp, công việc điều khiển rắc rối phức tạp hơn được hoàn thành với một hệ tự động hóa rất cao. Thiết bị mà có thể phục vụ cho việc điều khiển này một cách thông minh, chính xác thì cần nói đến là PLC.

2/ PLC là gì?

PLC là bộ điều khiển mà tùy thuộc vào người sử dụng một loạt hay trình tự các sự kiện, các sự kiện này được kích hoạt bởi các tác nhân kích thích (hay còn gọi là ngõ vào) tác động vào PLC hoặc qua các hoạt động có trễ như thời gian định thì hay các sự kiện được đếm. Một khi các sự kiện được kích hoạt, thật sự là nó bật ON hay OFF thiết bị bên ngoài hay còn gọi các thiết bị vật lý (các thiết bị này gắn vào ngõ vào của nó). Như vậy chúng ta có thể hiểu rằng PLC là một bộ "điều khiển logic theo chương trình". Ta chỉ cần thay đổi chương trình cài đặt trong PLC là PLC có thể thực hiện được các chức năng khác nhau, điều khiển trong những môi trường khác nhau.

3/ Vai trò của PLC.

Trong hệ thông tự động hóa, PLC được xem như là trung tâm của hệ thống điều khiển. Với một chương trình ứng dụng điều khiển (được lưu trữ bên trong bộ nhớ của PLC), trong quá trình thi hành của PLC liên tục kiểm tra trạng thái của hệ thống xuyên suốt từ tín hiệu phản hồi của các thiết bị trường nhập. sau đó nó dựa vào chương trình logic để quyết định chu kỳ hành động để mang các tín hiệu điều khiển ra ngoài trường các thiết bị xuất.

Trong hệ thống điều khiển công nghiệp, PLC được sử dụng khá phổ biến bởi tính ổn định, mềm dẽo và chính xác.

Với sức mạnh của nó nên PLC được các nhà sản xuất trên thế giới đã chế tạo ra các loại PLC ngoài những đặc điểm chung còn bổ sung những thế mạnh riêng của mình.

Hiện có các nhà sản xuất PLC nổi tiếng như:

SIEMENS của Đức gồm có: S7 – 200; S7 – 300

OMRON của Nhật gồm có: LOGO, CX – Programmer, v.v….

II. Các hệ đếm (Number system)

1/ Các hệ đếm:

Hệ nhị phân (Bin – Binary)

Hệ bát phân (Oct – Octal)

Hệ thập phân (Dec – Decimal)

Hệ thập lục phân (Hex – Hexadecimal)

2/ Cách biểu diễn số nhị phân:

Vì bộ xử lý trung tâm (CPU) bên trong của PLC chỉ làm việc với 2 trạng thái: 0 hay 1 (OFF hay ON). Do đó, cần thiết phải đổi các hệ số khác và biểu diễn chúng dưới dạng các dãy số chứa 2 trạng thái: 0 hay 1.

a) Biểu diễn số thập phân dưới dạng BCD

Trong môn học kỹ thuật số, việc chuyển đổi rất đơn giản: cứ 1 ký số của số thập phân ta dùng 4 bit nhị phân để biểu diễn.

Ví dụ: Biểu diễn số 1369_{10} sang BCD

Trọng số bit của số Bin như sau:

3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
8	4	2	0	8	4	2	0	8	4	2	0	8	4	2	0

Vậy số 1369_{10} sang BCD là:

0	0	0	1	0	0	1	1	0	1	1	0	1	0	0	1

b) Biểu diễn số Hex dưới dạng số Bin:

Tương tự ta dùng 4 bit nhị phân để biểu diễn 1 ký số của số Hex. Ta có bảng sau:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0
2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
8	4	2	0	8	4	2	0	8	4	2	0	8	4	2	0

Trong PLC việc sử dụng trọng số của bit bị đảo ngược lại. Cụ thể như sau:

- Trong hệ thống số, biểu diễn trọng số bit theo trình tự:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- Trong PLC, biểu diễn trọng số bit theo trình tự:

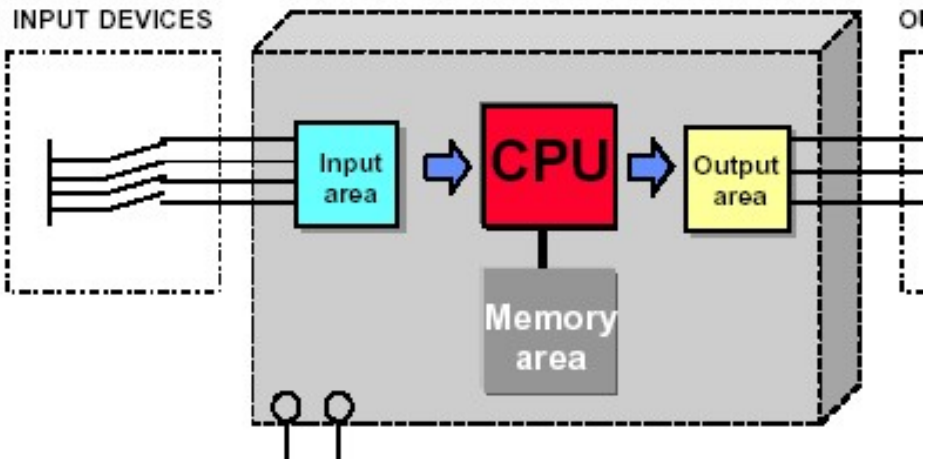
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Cho nên chúng ta cần lưu ý đến vấn đề này khi biểu diễn.

III. Cấu trúc cơ bản của PLC OMRON:

1/ Cấu trúc cơ bản của PLC OMRON:

Cấu trúc cơ bản của PLC OMRON



2/ Các phần chính trong PLC:

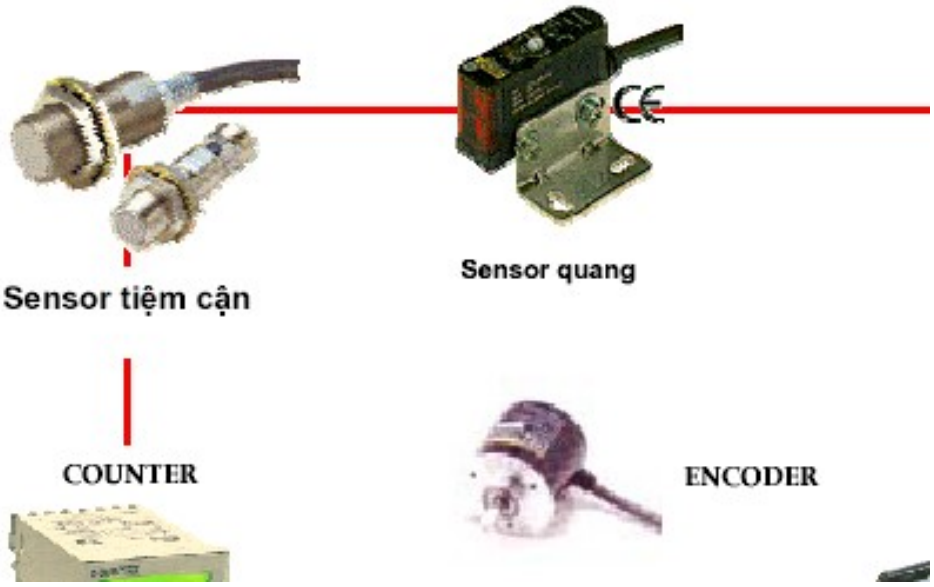
a) Phần giao diện đầu vào (input area)

Tùy loại Model mà PLC có các địa chỉ công tắc mô phỏng ngõ vào khác nhau.

Nếu là PLC OMRON CQM1 – CPU21 thì địa chỉ ngõ vào là: 000.00 ÷ 000.15 (16 bit)

Nếu là PLC OMRON CPM2A – CPU22 thì địa chỉ ngõ vào là: 000.00 ÷ 000.15 (16 bit)

Nếu là PLC OMRON CJ1M – CPU11 thì địa chỉ ngõ vào là: 0.00 ÷ 0.15 (16 bit)



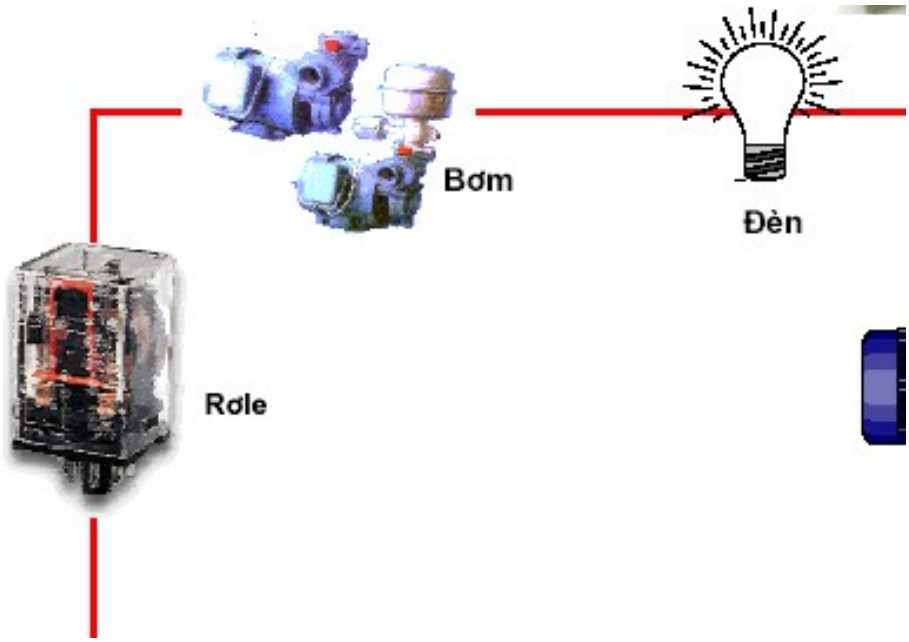
b) Phần giao diện đầu ra (output area)

Cũng như ngõ vào, tùy loại Model mà PLC có các địa chỉ đèn hiển thị để mô phỏng ngõ ra cũng khác nhau.

Nếu là PLC OMRON CQM1 – CPU21 thì địa chỉ ngõ ra là: 100.00 ÷ 100.15 (16 bit)

Nếu là PLC OMRON CPM2A – CPU22 thì địa chỉ ngõ ra là: 010.00 ÷ 010.15 (16 bit)

Nếu là PLC OMRON CJ1M – CPU11 thì địa chỉ ngõ ra là: 1.00 ÷ 1.15 (16 bit)



c) Bộ nhớ (Memory):

Lưu chương trình điều khiển được lập trình bởi người sử dụng và các dữ liệu khác như: cờ nhớ, thanh ghi tạm, trạng thái đầu vào, lệnh điều khiển đầu ra, ... Nội dung của bộ nhớ được mã hoá dưới dạng nhị phân.

d) Bộ xử lý trung tâm: CPU (Central Processing Unit)

Trình tự thực hiện các lệnh của chương trình được lưu trong bộ nhớ, xử lý các đầu vào và đưa ra kết quả ở đầu ra hay điều khiển cho phần giao diện đầu ra.

e) Nguồn cung cấp (Power Supply)

IV. Các vùng dữ liệu trong bộ nhớ PLC.

1/ Các vùng dữ liệu trong bộ nhớ PLC

Item	Specification
Internal I/O Area (work bits)	4,800 (300 words): CIO 120000 to CIO 149915 (words CIO 1200 to CIO 1499) 37,504 (2,344 words): CIO 380000 to CIO 614315 (words CIO 3800 to CIO 614 These bits in the CIO Area are used as work bits in programming to control pro execution. They cannot be used for external I/O.
Work Area	8,192 (512 words): W00000 to W51115 (words W000 to W511) These bits are used as work bits in programming to control program execution. cannot be used for external I/O. Note: When using work bits in programming, use bits in the Work Area first bef ing bits from other areas.
Holding Area	8,192 (512 words): H00000 to H51115 (words H000 to H511) Holding bits are used to control program execution, and maintain their ON/OFF when PLC is turned OFF or the operating mode is changed.
Auxiliary Area	Read-only: 7,168 (448 words): A00000 to A44715 (words A000 to A447) Read/write: 8,192 bits (512 words): A44800 to A95915 (words A448 to A959) Auxiliary bits are allocated specific functions.
Temporary Area	16 bits (TR0 to TR15) Temporary bits are used to store ON/OFF execution conditions at program brar
Timer Area	4,096: T0000 to T4095 (used for timers only)
Counter Area	4,096: C0000 to C4095 (used for counters only)
DM Area	32 Kwords: D00000 to D32767 Special I/O Unit DM Area: D20000 to D29599 (100 words × 96 Units). Used to rameters for Special I/O Units. CPU Bus Unit DM Area: D30000 to D31599 (100 words × 16 Units). Used to set eters for CPU Bus Units.
Index Registers	IR0 to IR15 Store PLC memory addresses for indirect addressing.

2/ Một số Relay đặc biệt

Relay tạo xung Clock

P_1min (CF104): xung clock 1 phút

P_0_02s (CF103): xung clock 0,02 giây

P_1s (CF102): xung clock 1 giây

P_0_2s (CF101): xung clock 0,2 giây

P_0_1s (CF100): xung clock 0,1 giây

Name	Data Type	Address / Value	Usage	Comment
P_0_02s	BOOL	CF103	Work	0.02 second clock pulse bit
P_0_1s	BOOL	CF100	Work	0.1 second clock pulse bit
P_0_2s	BOOL	CF101	Work	0.2 second clock pulse bit
P_1min	BOOL	CF104	Work	1 minute clock pulse bit
P_1s	BOOL	CF102	Work	1.0 second clock pulse bit

3/ Một số cờ (Flag) sử dụng trong PLC

P_EQ (CF006): cờ so sánh bằng (Equals Flag)

P_First_Cycle (A200.11): cờ quét chu kì đầu.

P_GE (CF000): cờ so sánh lớn hơn hay bằng

P_GT (CF005): cờ so sánh lớn hơn (Greater Than)

P_LE (CF002): cờ so sánh nhỏ hơn hay bằng

P_LT (CF007): cờ so sánh nhỏ hơn (Less Than)

P_NE (CF001): cờ so sánh không bằng

P_Off (CF114): cờ báo luôn OFF

P_On (CF113): cờ báo luôn On

Name	Data Type	Address / Value	Usage	Comment
P_AER	BOOL	CF011	Work	Access Error Flag
P_CIO	WORD	A450	Work	CIO Area Parameter
P_CY	BOOL	CF004	Work	Carry (CY) Flag
P_Cycle_Time_Er...	BOOL	A401.08	Work	Cycle Time Error Flag
P_Cycle_Time_V...	UDINT	A264	Work	Present Scan Time
P_DM	WORD	A460	Work	DM Area Parameter
P_EM0	WORD	A461	Work	EM0 Area Parameter
P_EM1	WORD	A462	Work	EM1 Area Parameter
P_EM2	WORD	A463	Work	EM2 Area Parameter
P_EM3	WORD	A464	Work	EM3 Area Parameter
P_EM4	WORD	A465	Work	EM4 Area Parameter
P_EM5	WORD	A466	Work	EM5 Area Parameter
P_EM6	WORD	A467	Work	EM6 Area Parameter
P_EM7	WORD	A468	Work	EM7 Area Parameter
P_EM8	WORD	A469	Work	EM8 Area Parameter
P_EM9	WORD	A470	Work	EM9 Area Parameter
P_EMA	WORD	A471	Work	EMA Area Parameter
P_EMB	WORD	A472	Work	EMB Area Parameter
P_EMC	WORD	A473	Work	EMC Area Parameter
P_EQ	BOOL	CF006	Work	Equals (EQ) Flag
P_ER	BOOL	CF003	Work	Instruction Execution Error (ER) Flag
P_First_Cycle	BOOL	A200.11	Work	First Cycle Flag
P_First_Cycle_Task	BOOL	A200.15	Work	First Task Execution Flag
P_GE	BOOL	CF000	Work	Greater Than or Equals (GE) Flag
P_GT	BOOL	CF005	Work	Greater Than (GT) Flag
P_HR	WORD	A452	Work	HR Area Parameter
P_IO_Verify_Error	BOOL	A402.09	Work	I/O Verification Error Flag
P_LE	BOOL	CF002	Work	Less Than or Equals (LE) Flag
P_Low_Battery	BOOL	A402.04	Work	Low Battery Flag
P_LT	BOOL	CF007	Work	Less Than (LT) Flag
P_Max_Cycle_Time	UDINT	A262	Work	Maximum Cycle Time
P_N	BOOL	CF008	Work	Negative (N) Flag
P_NE	BOOL	CF001	Work	Not Equals (NE) Flag
P_OF	BOOL	CF009	Work	Overflow (OF) Flag
P_Off	BOOL	CF114	Work	Always OFF Flag
P_On	BOOL	CF113	Work	Always ON Flag
P_Output_Off_Bit	BOOL	A500.15	Work	Output OFF Bit
P_Step	BOOL	A200.12	Work	Step Flag
P_UF	BOOL	CF010	Work	Underflow (UF) Flag
P_WR	WORD	A451	Work	WR Area Parameter

V. Cách lập trình trong PLC:

1/ Khái niệm

Tuỳ hãng PLC mà có phần mềm biên dịch khác nhau.

Đối với hãng OMRON thì có chương trình CX – Programmer.

2/ Lập trình bằng Programming Console

Có 3 chế độ hoạt động:

Sử dụng các lệnh gợi nhớ, các mã lệnh để viết chương trình và cho hoạt động.

3/ Lập trình bằng phần mềm CX – Programmer

Cách cài phần mềm như các phần mềm ứng dụng khác.

Ta có thể sử dụng các lệnh gợi nhớ, các mã lệnh để viết chương trình(dạng Mnemonics code) hay sử dụng các ký hiệu để vẽ chương trình hình thang(dạng Diagram Ladder) và cho hoạt động.

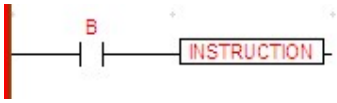
BÀI 2:

CÁC LỆNH CƠ BẢN

I. Lệnh LD, LDNOT

1/ Ký hiệu:

a) Lệnh LD



b) Lệnh LDNOT

2/ Lệnh gợi nhớ:

a) Lệnh LD:

LD B

Lệnh

b) Lệnh LDNOT:

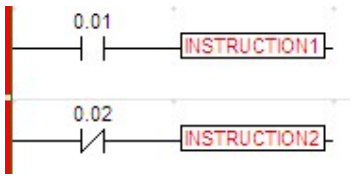
LDNOT B

Lệnh

Trong đó: B(bit) có thể là: I(input), O(output), HR(Holding Relay), TR(Temporary Register - ReLay), #, v.v..

3/ Chức năng: Khởi tạo tiếp điểm thường hở /đóng ở đầu dòng lệnh hoặc nhánh lệnh. Mọi dòng lệnh hay một chương trình đều bắt đầu với lệnh Load(LD) hoặc Load Not(LDNOT).

4/ Ví dụ:



Lệnh gợi nhớ:

LD 0.01

Lệnh 1

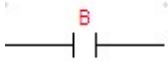
LDNOT 0.02

Lệnh 2

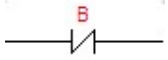
II. Lệnh AND, ANDNOT

1/ Ký hiệu:

a) Lệnh AND:



b) Lệnh ANDNOT:



2/ Lệnh gợi nhớ:

a) AND B

b) ANDNOT B

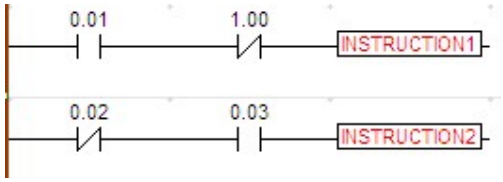
3/ Chức năng:

a) Lệnh AND: Lệnh này dùng để nối một tiếp điểm thường hở với một tiếp phía trước hoặc một nhánh, một khối tiếp điểm phía trước.

b) Lệnh ANDNOT: Lệnh này để nối một tiếp điểm thường đóng với một tiếp phía trước hoặc một nhánh, một khối tiếp điểm phía trước.

4/ Ví dụ:

a) Cho biểu đồ hình thang sau:



b) Viết sang lệnh gợi nhớ:

LD 0.01

ANDNOT 1.00

Lệnh 1

LDNOT 0.02

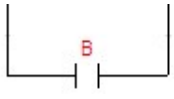
AND 0.03

Lệnh 2

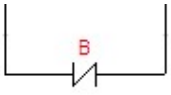
III. Lệnh OR, ORNOT

1/ Ký hiệu:

a) Lệnh OR:



b) Lệnh ORLD:



2/ Lệnh gợi nhớ:

a) OR B

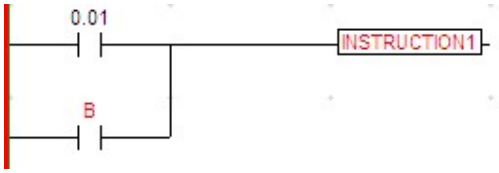
b) ORNOT B

3/ Chức năng:

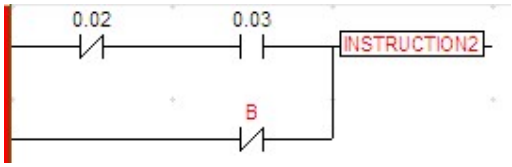
Nối song song một tiếp điểm thường hở/kín với một nhánh, khối tiếp điểm hay một tiếp điểm phía trước.

4/ Ví dụ:

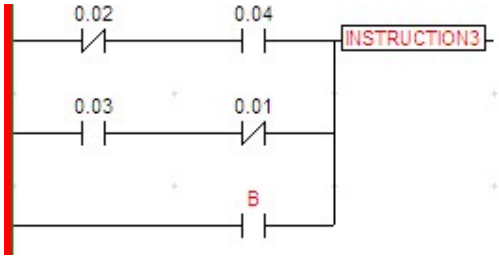
a) Trường hợp nối với một tiếp điểm:



b) Trường hợp nối với một nhánh tiếp điểm:



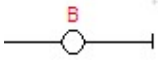
c) Trường hợp nối với một khối tiếp điểm:



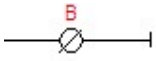
IV. Lệnh OUT, OUTNOT

1/ Ký hiệu:

a) Lệnh OUT



b) Lệnh OUTNOT



2/ Lệnh gợi nhớ:

a) Lệnh OUT

OUT B

b) Lệnh OUTNOT

OUTNOT B

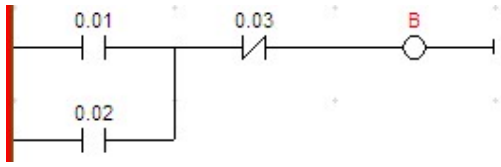
3/ Chức năng:

Ngược với Lệnh LD, LDNOT, Lệnh OUT hay OUTNOT là lệnh ngõ ra, báo hiệu kết thúc một dòng lệnh hay khối lệnh.

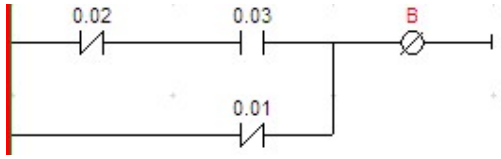
Khi đã có lệnh OUT hay OUTNOT thì trên dòng lệnh đó không thể nối nối tiếp một tiếp điểm hay một ngõ ra đứng ở phía sau lệnh OUT hay OUTNOT này nữa.

4/ Ví dụ:

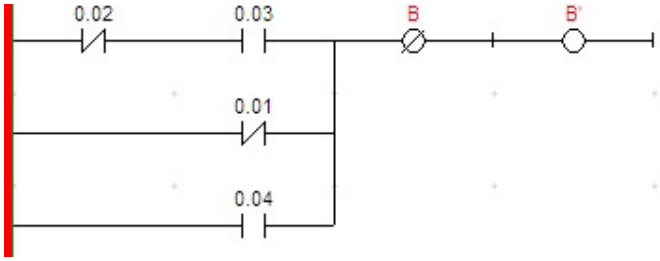
a) Viết sang lệnh gợi nhớ cho các biểu đồ hình thang sau:



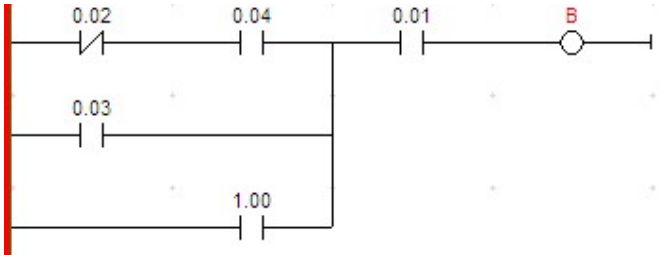
b) Viết sang lệnh gợi nhớ cho các biểu đồ hình thang sau:



c) Viết sang lệnh gợi nhớ cho các biểu đồ hình thang sau(không có trường hợp này!!!)



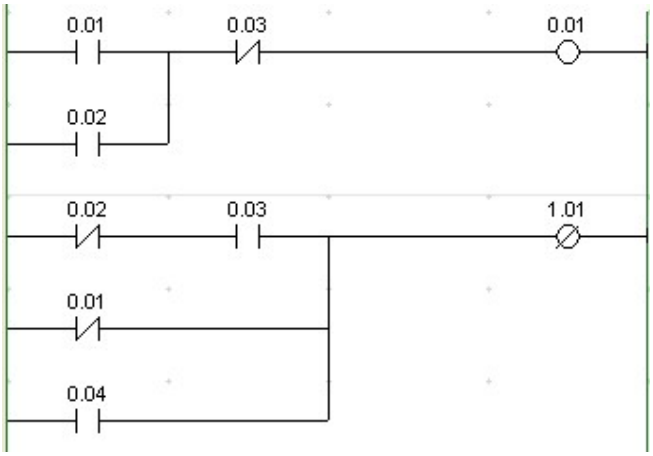
d) Viết sang lệnh gợi nhớ cho các biểu đồ hình thang sau:



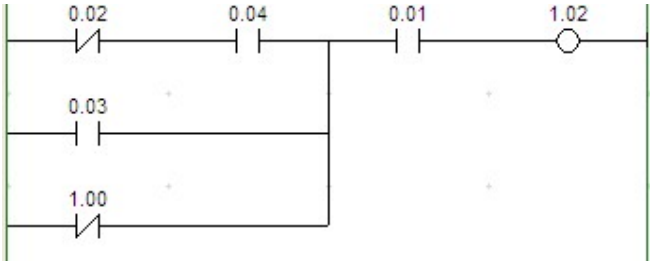
V. Bài tập

1/ Cho các biểu đồ hình thang sau:

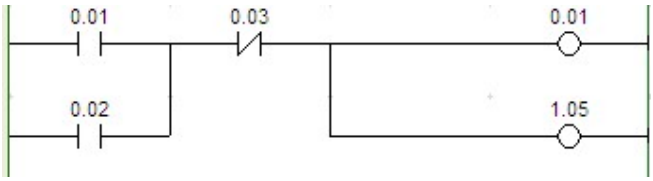
a) Viết sang lệnh gợi nhớ:



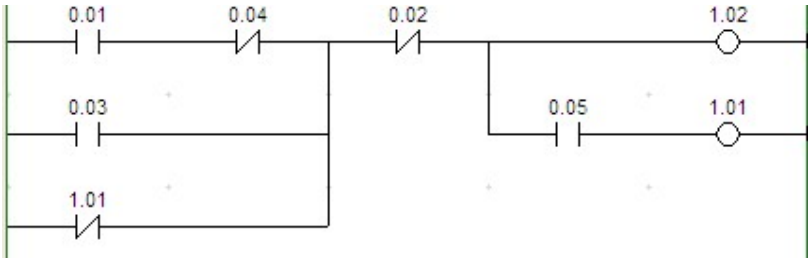
b) Viết sang lệnh gợi nhớ



c) Viết sang lệnh gợi nhớ



d) Viết sang lệnh gợi nhớ



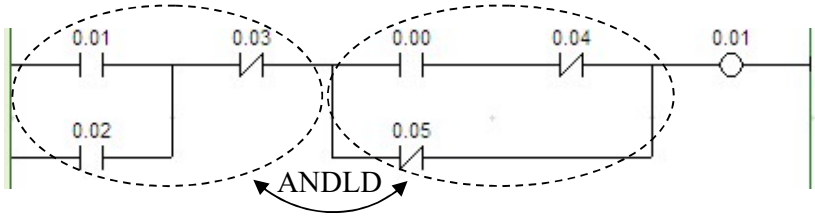
2/ Cho biết các ngõ ra ở câu 1 = “ON” khi nào?

BÀI 3:

LỆNH ANDLD, ORLD, TRx

I. Lệnh ANDLD

1/ Chức năng:



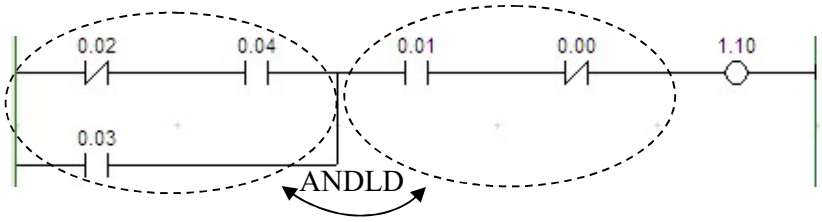
Dùng để nối hai khối hay hai nhánh tiếp điểm nối tiếp.

Ta biết rằng, để khởi tạo nhánh lệnh hay khối lệnh đều bắt đầu từ lệnh LD hay LDNOT. Vì vậy, trước khi sử dụng lệnh ANDLD phải có ít nhất 2 lệnh LD hay LDNOT

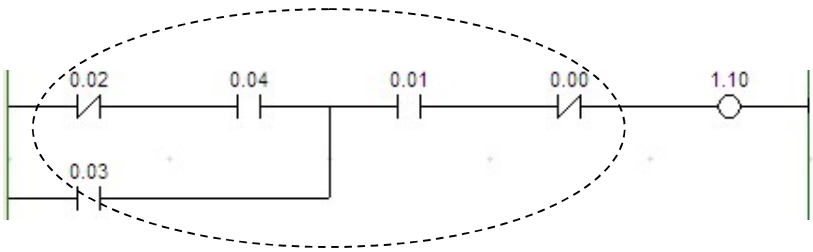
Tuỳ từng trường hợp mà ta có thể sử dụng lệnh ANDLD. Đối với những lệnh đơn giản(như ví dụ a) ta không nhất thiết phải dùng lệnh này. Nhưng có những trường hợp phải bắt buộc sử dụng(như ví dụ b,c)

2/ Ví dụ:

a) Nếu ta xem mạch này gồm có 2 khối thì lệnh gợi nhớ như sau:



LDNOT	0.02
AND	0.04
OR	0.03
LD	0.01
ANDNOT	0.00
ANDLD	
OUT	1.10



Nhưng mạch này ta có thể làm thành 1 khối. Lệnh gợi nhớ như sau:

LDNOT 0.02

AND 0.04

OR 0.03

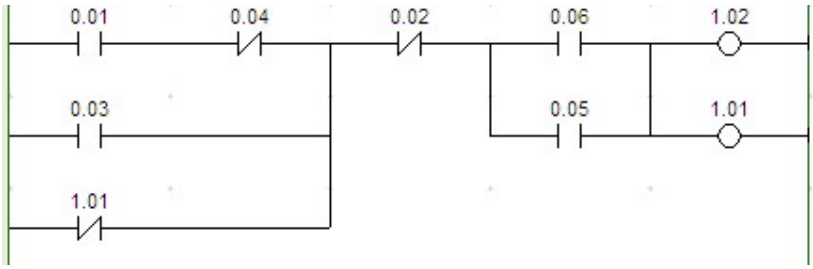
AND 0.01

ANDNOT 0.00

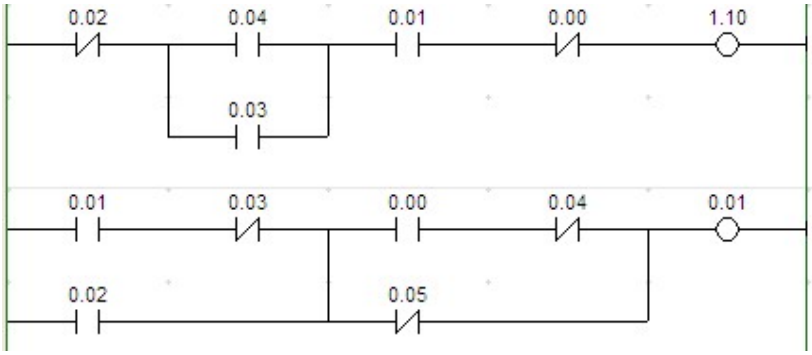
OUT 1.10

Ta thấy cách viết chương trình này ngắn hơn chương trình trên.

b) Viết sang Lệnh gợi nhớ như sau:

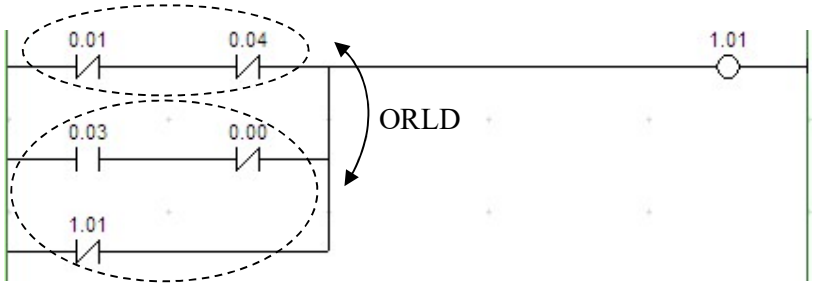


c) Viết sang Lệnh gợi nhớ như sau:



II. Lệnh ORLD

1/ Chức năng:

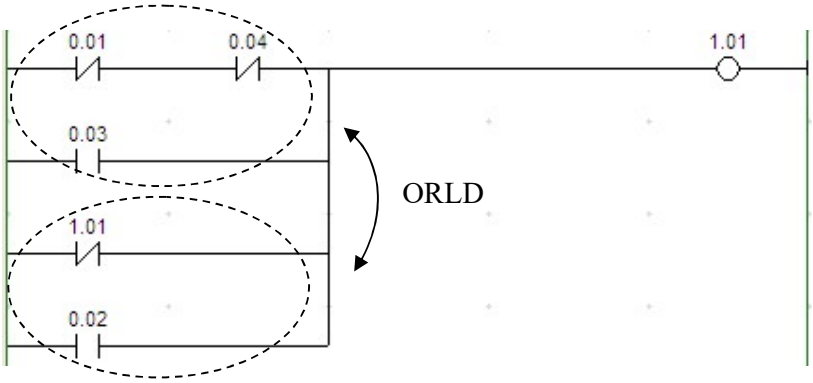


Dùng để nối hai khối hay hai nhánh tiếp điểm nối song song.

Ta biết rằng, để khởi tạo nhánh lệnh hay khối lệnh đều bắt đầu từ lệnh LD hay LDNOT. Vì vậy, trước khi sử dụng lệnh ORLD phải có ít nhất 2 lệnh LD hay LDNOT

Tuỳ từng trường hợp mà ta có thể sử dụng lệnh ORLD. Đối với những lệnh đơn giản(như ví dụ a) ta không nhất thiết phải dùng lệnh này. Nhưng có những trường hợp phải bắt buộc sử dụng(như ví dụ b,c)

2/ Ví dụ:



a) Viết sang Lệnh gợi nhớ như sau:

Nếu ta xem mạch này gồm có 2 khối thì lệnh gợi nhớ như sau:

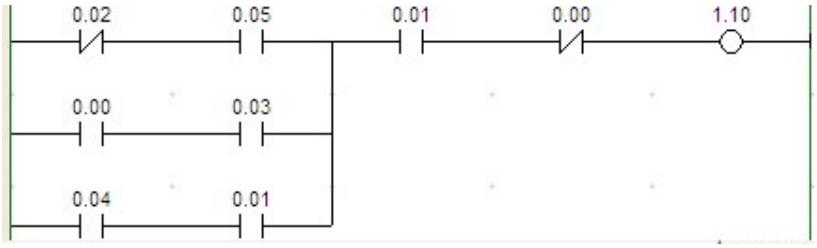
LDNOT 0.01
ANDNOT 0.04
OR 0.03
LDNOT 1.01
OR 0.02
ORLD
OUT 1.01

Nhưng mạch này ta có thể làm thành 1 khối. Lệnh gợi nhớ như sau:

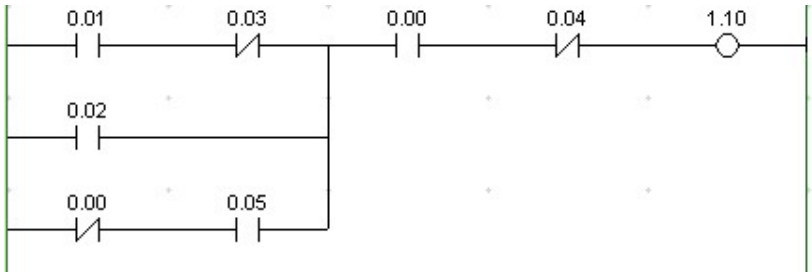
LDNOT 0.01
ANDNOT 0.04
OR 0.03
ORNOT 1.01
OR 0.02
OUT 1.01

Ta thấy cách viết chương trình này ngắn hơn chương trình trên.

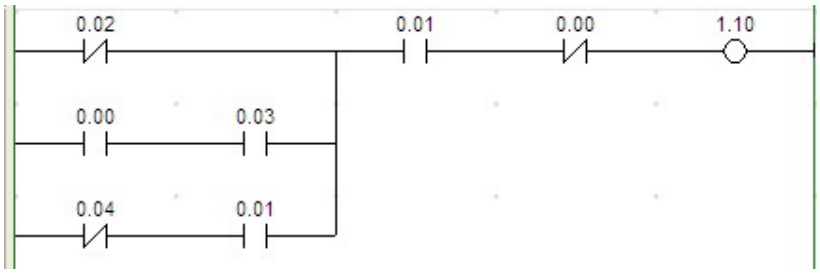
b) Viết sang Lệnh gợi nhớ như sau:



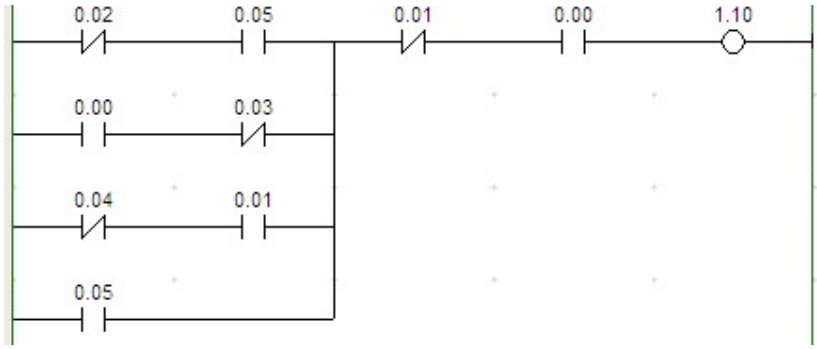
c) Viết sang Lệnh gợi nhớ như sau:(Cách vẽ này không nên sử dụng!!!)



d) Viết sang Lệnh gợi nhớ như sau: (Cách vẽ này không nên sử dụng!!!)



e) Viết sang Lệnh gợi nhớ như sau:



III. Lệnh TRx

1/ Chức năng:

TR(Temporary Register - Relay) Dùng để lưu trữ tạm thời các trạng thái ON/OFF của các tiếp điểm đứng trước, trước khi gọi lệnh này.

Lệnh TRx được sử dụng khi dòng lệnh có rẽ nhánh hay một nhóm lệnh được sử dụng nhiều lần trong 1 dòng lệnh.

TR có 8 bit là TR0 ÷ TR7. TR0 có thể sử dụng nhiều lần trong 1 chương trình.

- Cách gọi TRx: (x = 0 ÷ 7)

OUT TRx

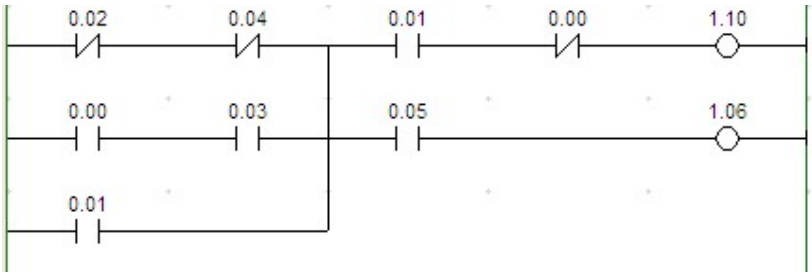
- Để sử dụng lại ta thực hiện lệnh:

LD TRx

2/ Ví dụ:

a) Viết sang Lệnh gợi nhớ như sau:

LDNOT 0.02



ANDNOT 0.04

LD 0.00

AND 0.03

OR 0.01

ORLD

OUT TR1

AND 0.01

ANDNOT 0.00

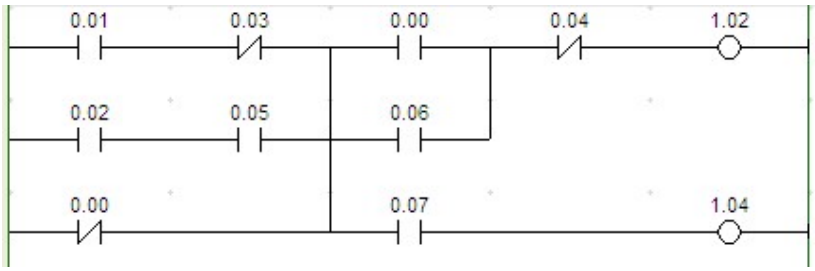
OUT 1.10

LD TR1

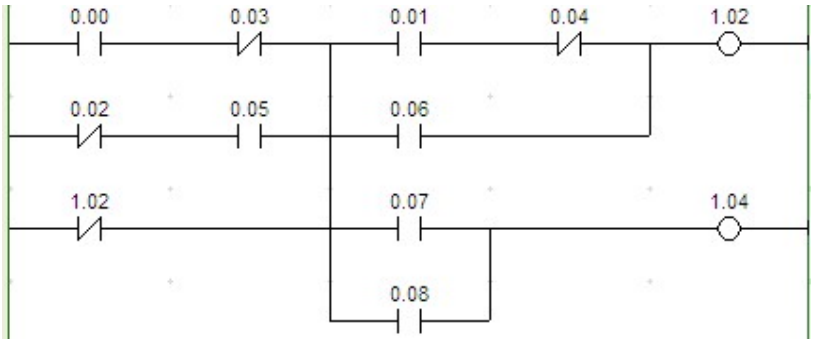
AND 0.05

OUT 1.06

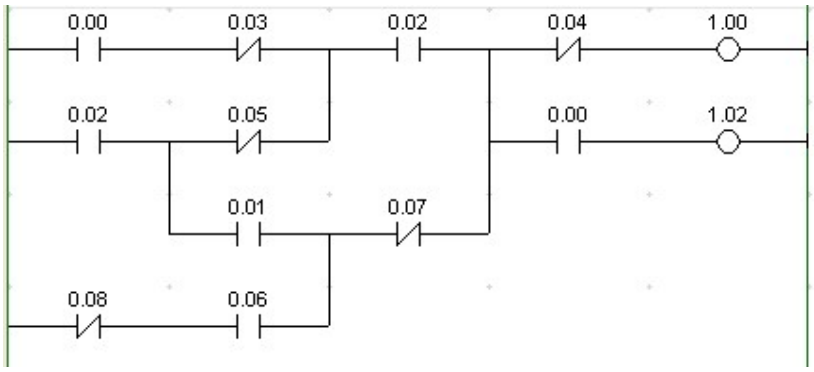
b) Viết sang Lệnh gợi nhớ như sau: (15 câu lệnh)



c) Viết sang Lệnh gợi nhớ như sau: (17 câu lệnh)



d) Viết sang Lệnh gợi nhớ như sau :



IV. Một số ứng dụng

1/ Mạch tự giữ

a) Dạng mạch:

b) Lệnh gợi nhớ:

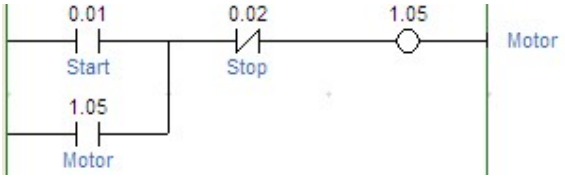
c) Nguyên lý hoạt động:

Khi ta nhấn nút nhấn/công tắc 0.00(0.00 = ON) thì ngõ ra 1.01 = ON. Sau đó, Công tắc 0.00 cho = OFF thì 1.01 vẫn ON do ta lấy ngõ ra này làm tiếp điểm ngõ vào để duy trì chính nó.

Như vậy, trạng thái ngõ ra 1.01 luôn = ON khi ta chỉ cần nhấn ngõ vào 0.00 một lần duy nhất.

2/ Mạch khởi động/tắt động cơ

a) Dạng mạch:

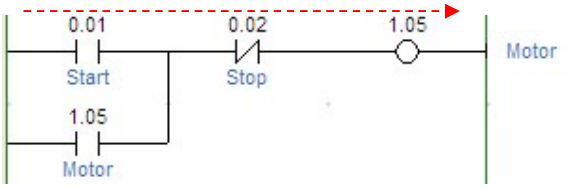


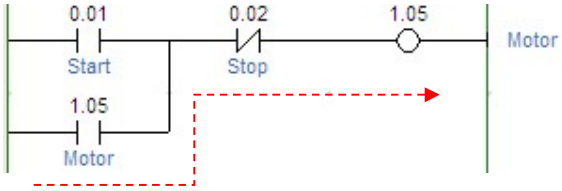
b) Lệnh gợi nhớ:

c) Nguyên lý hoạt động:

Khi ta nhấn nút nhấn Start(0.01 = ON) động cơ sẽ hoạt động do nút nhấn Stop thường đóng. Lúc này trạng thái động cơ được hồi tiếp về ngõ vào để duy trì khi nút nhấn Start không còn nhấn nữa. Chiều dòng điện chạy trong mạch được minh họa như sau:

Khi mới nhấn Công tắc:



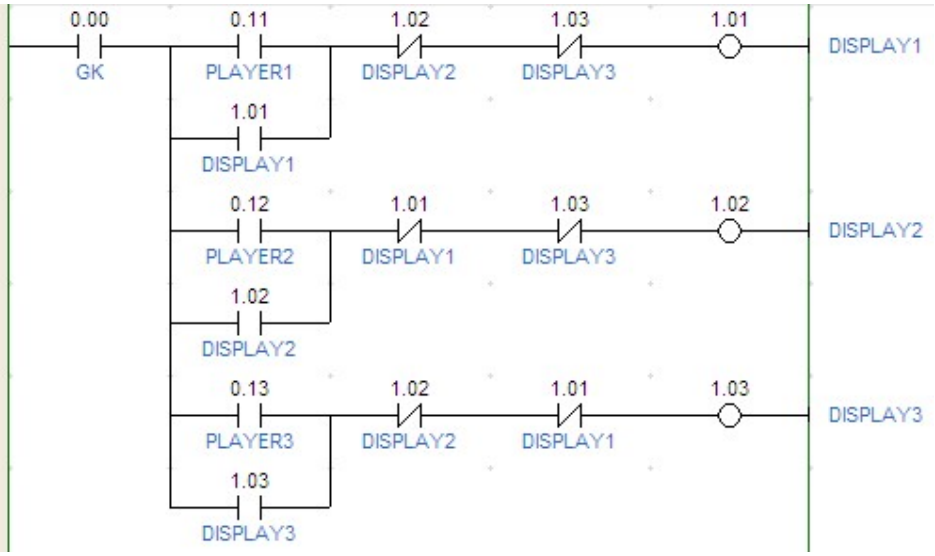


Khi nút nhấn được nhả ra:

Khi nhấn Stop thì sẽ ngắt đường dẫn điện tới Động cơ làm Động cơ không hoạt động.

3/ Mạch ưu tiên

a) Dạng mạch:



b) Lệnh gợi nhớ:

c) Nguyên lý hoạt động:

V. Bài tập

1/ Bài 1:

Vẽ biểu đồ hình thang và viết sang lệnh gợi nhớ cho mạch ưu tiên 2 đội chơi có sự điều khiển của giám khảo.

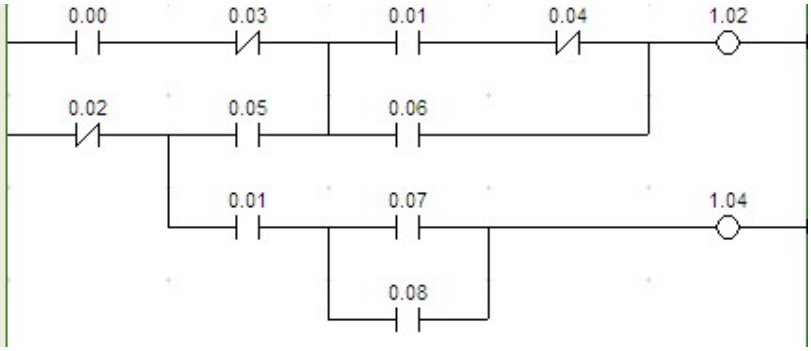
2/ Bài 2:

Vẽ biểu đồ hình thang và viết sang lệnh gợi nhớ cho mạch điều khiển 3 động cơ theo yêu cầu sau:

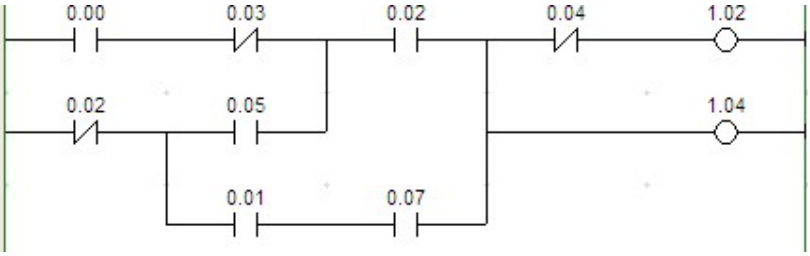
Khi nhấn Start(0.10) thì 2 Động cơ cùng hoạt động. Khi nhấn Stop1(0.11) thì Động cơ 1 dừng. Khi nhấn Stop 2(0.12) thì Động cơ 2 dừng.

3/ Bài 3:

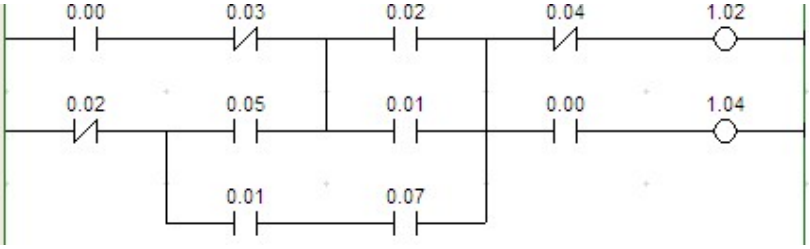
a) Viết sang Lệnh gợi nhớ như sau:



b) Viết sang Lệnh gợi nhớ như sau:



c) Viết sang Lệnh gợi nhớ như sau:



BÀI 4:

XÂY DỰNG BIỂU ĐỒ HÌNH THANG

I. Nguyên tắc vẽ biểu đồ hình thang

Nhìn tổng thể chương trình để xem chương trình có bao nhiêu lệnh liên kết, bao nhiêu lệnh TRx, lệnh điều khiển(học phần sau).

Đối với các lệnh liên kết(ANDLD, ORLD) phải đi ngược trở về trước từ các lệnh này để xác định số lệnh LD.

Đối với lệnh TRx, xác định số lần gọi sử dụng lại.

Đối với lệnh điều khiển, xác định số ngõ vào của từng lệnh.

II. Các lệnh cơ bản

1/ Chức năng:

Dùng để thực hiện một số chương trình đơn giản, giúp người học làm quen với cách xây dựng một biểu đồ hình thang từ một số lệnh.

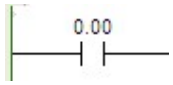
2/ Ví dụ:

a) Vẽ biểu đồ hình thang cho chương trình sau:

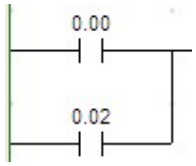
Step	Instruction	Operand
0	LD	0.00
1	OR	0.02
2	ANDNOT	0.01
3	OR	0.03
4	AND	0.04
5	OUT	1.02

Các bước vẽ như sau:

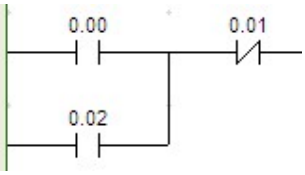
LD 0.00



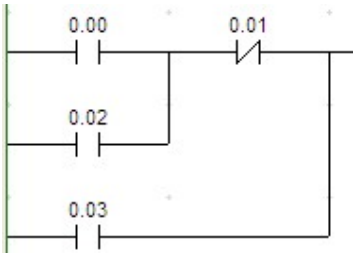
OR 0.02



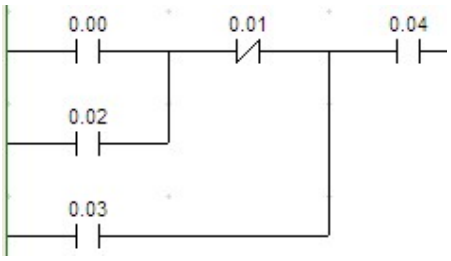
ANDNOT 0.01



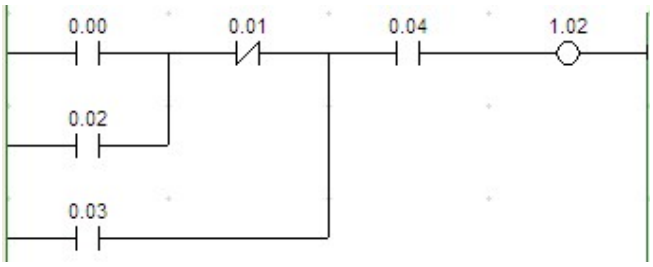
OR 0.03



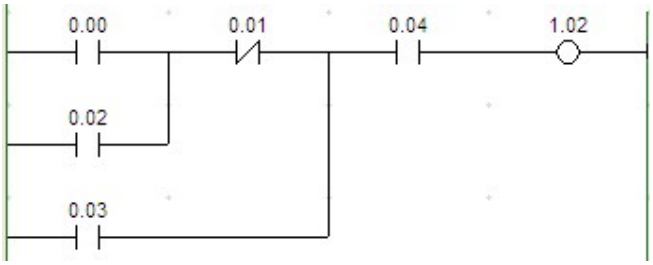
AND 0.04



OUT 1.02



Biểu đồ hình thang cho chương trình trên là:



III. Lệnh ORLD, ANDLD

1/ Chức năng:

Như ta biết ở phần trước, lệnh ANDLD dùng để nối hai khối hay hai nhánh tiếp điểm nối tiếp và lệnh ORLD dùng để nối hai khối hay hai nhánh tiếp điểm nối song song.

Đối với 1 chương trình có sự hiện diện của các lệnh liên kết này(ANDLD, ORLD) ta phải đi ngược trở về trước để xác định số lệnh LD. Cứ 02 lệnh LD/LDNOT sẽ dùng 01 lệnh liên kết(hay 01 lệnh liên kết sẽ nối 02 lệnh LD/LDNOT)

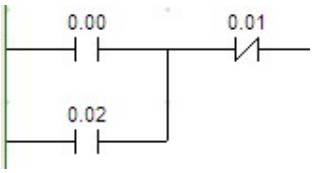
2/ Ví dụ:

a) Vẽ biểu đồ hình thang cho chương trình sau:

Step	Instruction	Operand
0	LD	0.00
1	OR	0.02
2	ANDNOT	0.01
3	LDNOT	0.03
4	AND	0.05
5	ORLD	
6	AND	0.04
7	OUT	1.02

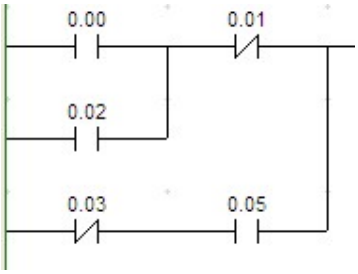
Từ chương trình trên ta thấy có 02 nhánh (bắt đầu là LD 0.00 và LDNOT 0.03) và 01 liên kết ORLD

Tương tự cách vẽ trên, ta có nhánh thứ nhất:

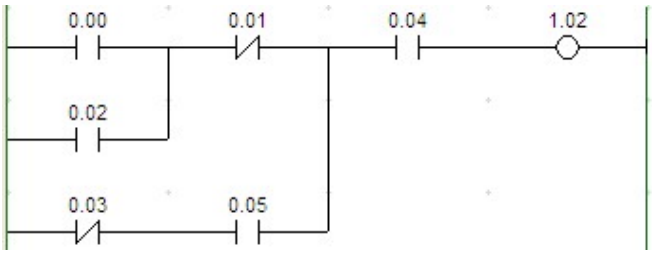


Tương tự, ta có nhánh thứ hai:

Do lệnh ORLD liên kết(nối song song) nên ta có:



Tiếp tục với lệnh còn lại sẽ có biểu đồ hoàn chỉng sau:



Vẽ biểu đồ hình thang cho chương trình sau:

Step	Instruction	Operand
0	LD	0.00
1	OR	0.03
2	ANDNOT	0.02
3	LDNOT	0.04
4	LD	0.05
5	OR	0.06
6	ANDLD	
7	ORLD	
8	LDNOT	0.04
9	AND	0.01
10	OR	0.07
11	ANDLD	
12	OUT	1.05

IV. Lệnh TRx

1/ Chức năng:

Lệnh TRx được sử dụng khi dòng lệnh có rẽ nhánh hay một nhóm lệnh được sử dụng nhiều lần trong 1 dòng lệnh.

2/ Ví dụ:

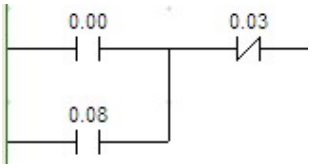
a) Vẽ biểu đồ hình thang cho chương trình sau:

Step	Instruction	Operand
0	LD	0.00
1	OR	0.08
2	ANDNOT	0.03
3	LDNOT	0.02
4	AND	0.05
5	ORLD	
6	ORNOT	1.02
7	OUT	TR0
8	LD	0.01
9	ANDNOT	0.04
10	LDNOT	0.06
11	AND	0.05
12	ORLD	
13	ANDLD	
14	OUT	1.02
15	LD	TR0
16	AND	0.07
17	OUT	1.04

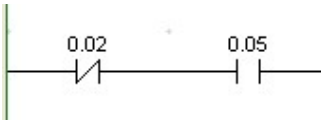
Phân tích:

Lệnh ORLD ở bước 5 để liên kết 02 lệnh LD ở bước 0 và bước 3

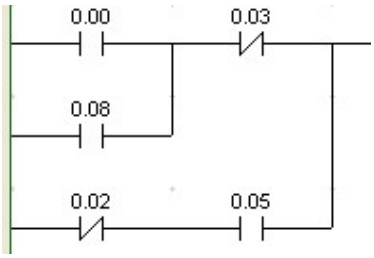
Nhánh LD ở bước 0 như sau:



Nhánh LD ở bước 3 như sau:

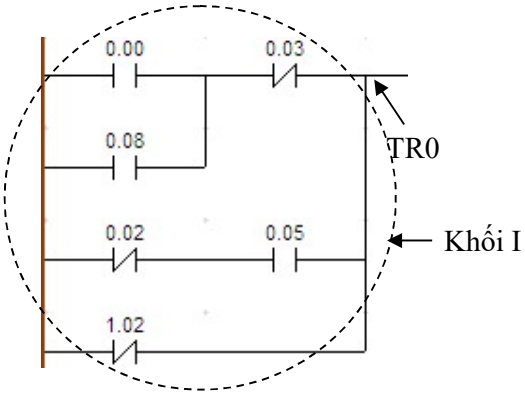


Liên kết 02 lệnh LD ở bước 0 và bước 3 dùng ORLD ở bước 5 ta được:



Tại bước 6 ta nối song song tiếp điểm thường đóng 1.02 với khối trên.

Tại bước 7 gọi TR0 để rẽ nhánh, như vậy khối sẽ dùng chung khi gọi lại TR0 là(gọi là khối I):

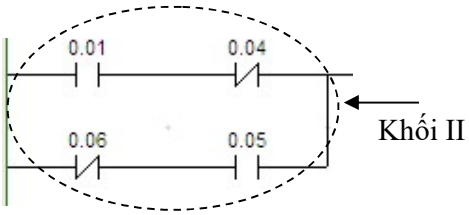


Lệnh ORLD ở bước 12 để liên kết 02 lệnh LD ở bước 8 và bước 10

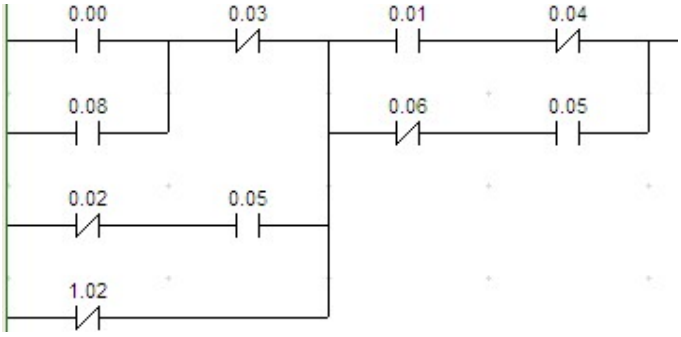
Nhánh LD ở bước 8 như sau:

Nhánh LD ở bước 10 như sau:

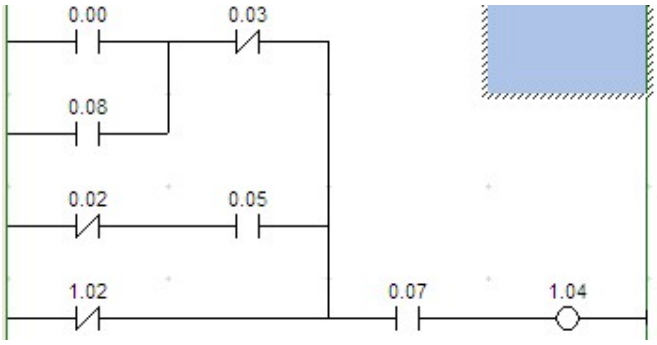
Liên kết 02 lệnh LD ở bước 8 và bước 10 dùng ORLD ở bước 12 thành 1 khối(gọi là khối II):



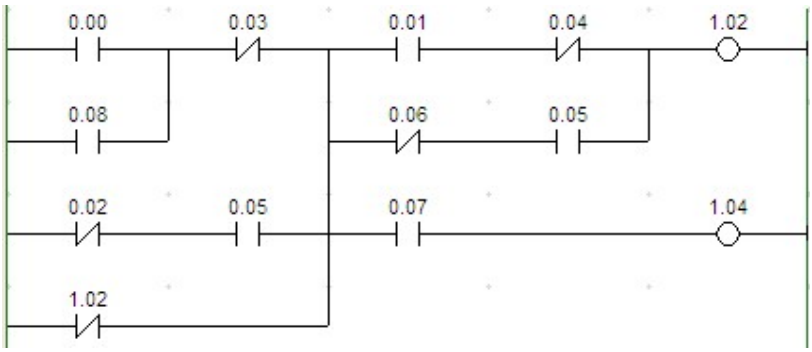
Lệnh ANDLD ở bước 13 để nối khối I và Khối II nối tiếp:



Lệnh LD TR0 ở bước 15 để gọi lại khối I, nối thêm các bước 16, 17 ta có:



Như vậy, biểu đồ hình thang của chương trình trên là:



b) Vẽ biểu đồ hình thang cho chương trình sau:

Step	Instruction	Operand
0	LD	0.00
1	OR	0.08
2	ANDNOT	0.03
3	AND	0.07
4	LDNOT	0.02
5	AND	0.05
6	OUT	TR0
7	AND	0.04
8	ORLD	
9	OUT	TR1
10	AND	0.05
11	OUT	1.04
12	LD	TR1
13	AND	0.09
14	OUT	1.06
15	LD	TR0
16	AND	0.01
17	AND	0.06
18	OUT	1.08

V. Bài tập

1/ Bài 1: Vẽ biểu đồ hình thang cho chương trình sau:

Step	Instruction	Operand
0	LDNOT	0.00
1	OR	0.08
2	AND	0.07
3	LDNOT	0.02
4	AND	0.05
5	OR	0.03
6	OUT	TR0
7	AND	0.04
8	ORLD	
9	OUT	TR1
10	AND	0.05
11	OUT	1.01
12	LD	TR1
13	AND	0.09
14	OUT	1.02
15	LD	TR0
16	ANDNOT	0.01
17	AND	0.06
18	OUT	1.03

2/ Bài 2: Vẽ biểu đồ hình thang cho chương trình sau:

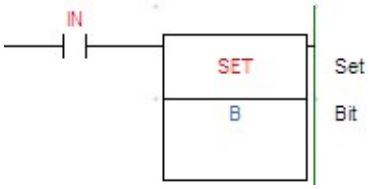
Step	Instruction	Operand
0	LDNOT	0.00
1	AND	0.07
2	LD	0.08
3	AND	0.03
4	ORLD	
5	LD	0.02
6	LD	0.05
7	OR	0.03
8	ANDLD	
9	OUT	TR0
10	AND	0.04
11	ORLD	
12	OUT	TR1
13	AND	0.05
14	OUT	1.01
15	LD	TR1
16	AND	0.09
17	LD	TR0
18	ANDNOT	0.01
19	AND	0.06
20	ORLD	
21	OUT	1.02
22	OUT	1.03

BÀI 5:

CÁC LỆNH ĐIỀU KHIỂN TRONG PLC

I. Lệnh SET

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD I

SET B

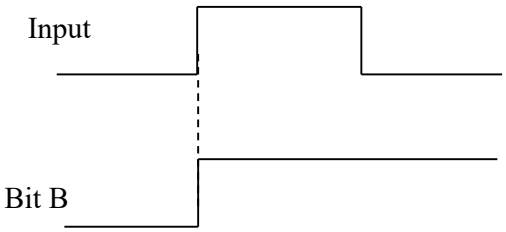
Ở đây, I là ngõ vào, nó có thể là 1 tiếp điểm hay một nhóm tiếp điểm.

B là bit cần điều khiển.

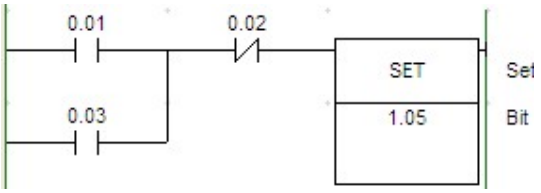
3/ Chức năng:

Lệnh SET dùng để đặt ON bit B khi điều kiện thực thi ON(I = ON) và giá trị này sẽ không thay đổi khi điều kiện thực thi là OFF.

4/ Giản đồ thời gian:

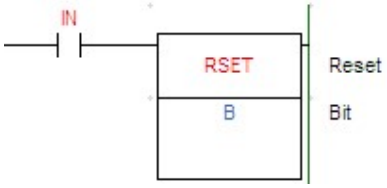


5/ Ví dụ:



II. Lệnh RESET(RSET)

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

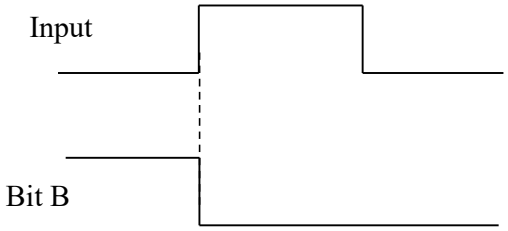
LD I

RSET B

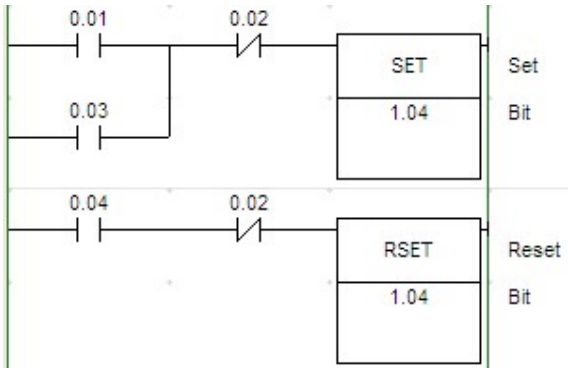
3/ Chức năng:

Lệnh RSET dùng để đặt OFF bit B khi điều kiện thực thi ON và giá trị này sẽ không thay đổi khi điều kiện thực thi là OFF.

4/ Giản đồ thời gian:

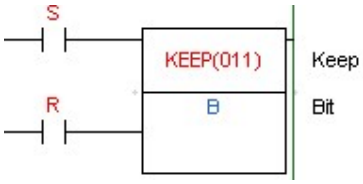


5/ Ví dụ:



III. Lệnh KEEP(011)

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD S

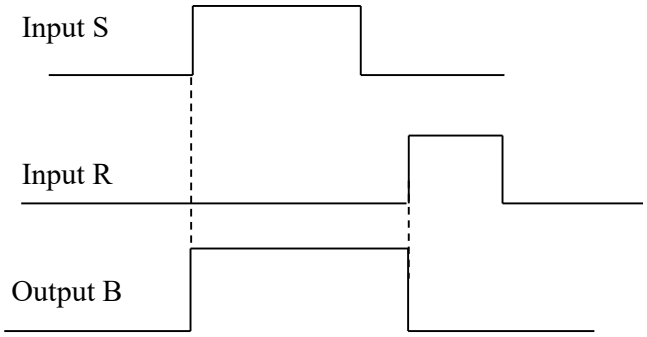
LD R

KEEP(011) B

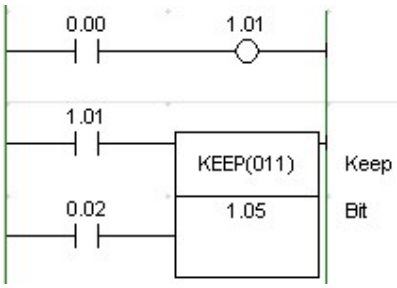
3/ Chức năng:

Dùng để duy trì trạng thái bit B. Khi S = ON thì B = ON và trạng thái B này sẽ duy trì cho đến khi R = ON

4/ Giản đồ thời gian



5/ Ví dụ: Viết sang lệnh gợi nhớ cho chương trình sau:



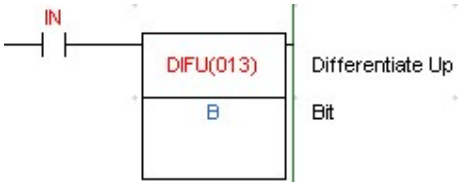
Khi 0.00 = ON; 0.02 = OFF, ngõ ra nào = ON?

Sau đó, 0.00 = OFF; 0.02 = OFF, ngõ ra nào = ON?

Sau đó, 0.00 = OFF; 0.02 = ON, ngõ ra nào = ON?

IV. Lệnh DIFU(013): Differentiate Up

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

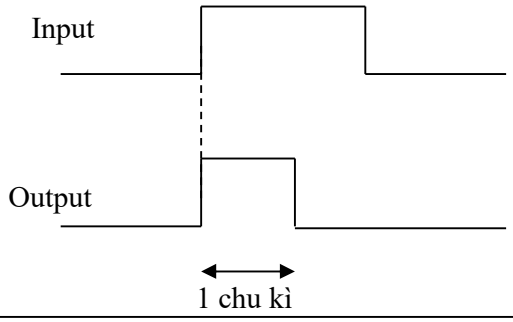
LD I

DIFU(013) B

3/ Chức năng:

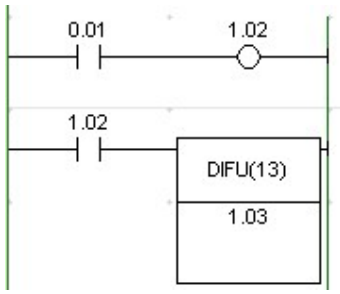
Đặt ON bit B trong một chu kỳ quét khi gặp cạnh lên tín hiệu vào (tín hiệu vào I chuyển từ OFF sang ON).

4/ Giản đồ thời gian



5/ Ví dụ:

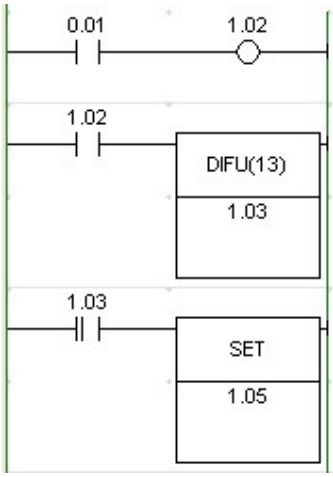
a) Viết sang lệnh gợi nhớ cho chương trình sau:



Khi 0.01 = ON, ngõ ra nào = ON?

Sau đó, 0.01 = OFF, ngõ ra nào = ON?

b) Viết sang lệnh gợi nhớ cho chương trình sau:

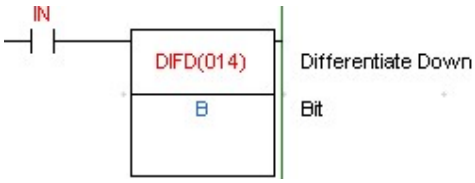


Khi 0.01 = ON, ngõ ra nào = ON?

Sau đó, 0.01 = OFF, ngõ ra nào = ON?

V. Lệnh DIFD(014): Differentiate Down

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

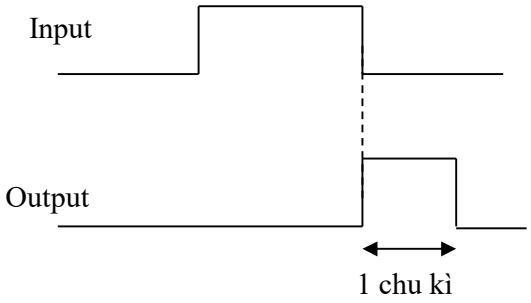
LD I

DIFD(014) B

3/ Chức năng:

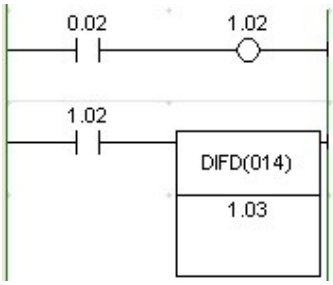
Đặt ON bit B trong một chu kỳ quét khi gặp cạnh xuống tín hiệu vào (tín hiệu vào I chuyển từ ON sang OFF).

4/ Giản đồ thời gian



5/ Ví dụ:

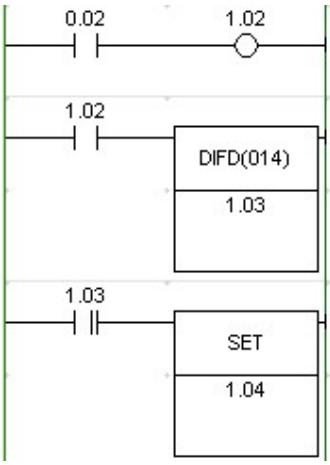
a) Viết sang lệnh gợi nhớ cho chương trình sau:



Khi 0.02 = ON, ngõ ra nào = ON?

Sau đó, 0.02 = OFF, ngõ ra nào = ON?

b) Viết sang lệnh gợi nhớ cho chương trình sau:

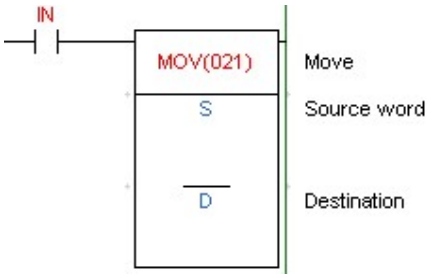


Khi 0.02 = ON, ngõ ra nào = ON?

Sau đó, 0.02 = OFF, ngõ ra nào = ON?

VI. Lệnh MOV(021): MOVE

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD I

MOV(021) S

D

3/ Hoạt động:

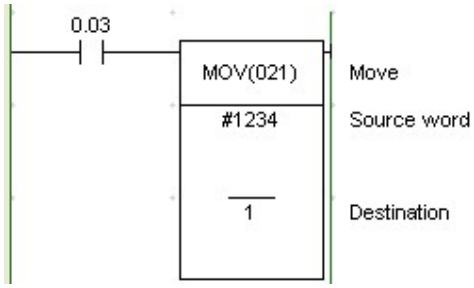
Khi ngõ vào I = ON, nội dung dữ liệu từ S sẽ được di chuyển(sao chép) sang D.

Kết quả: trong D chứa nội dung của S. Nội dung của S không đổi, nội dung của D thay đổi và bằng S.

Sau đó, Khi ngõ vào I = OFF……

4/ Ví dụ:

Viết sang lệnh gợi nhớ cho chương trình sau:

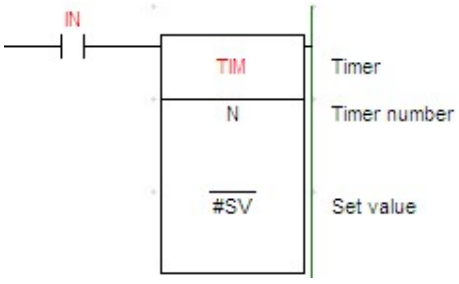


Khi 0.03 = ON, ngõ ra nào = ON?

Sau đó, 0.03 = OFF, ngõ ra nào = ON?

VII. Lệnh TIM: TIMER

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD I

TIM N

#SV

N: 000 ÷ 511(CQM1); 0000÷ 4095(CJ1M)

SV: 0000 ÷ 9999(BCD)

3/ Hoạt động:

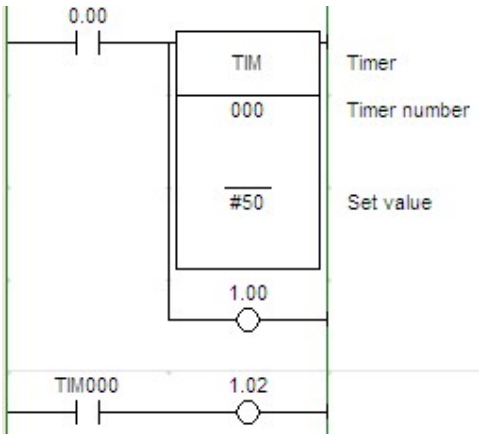
Khi ngõ vào I = ON, Timer thứ N hoạt động, giá trị SV giảm dần, khi giá trị hiện hành (PV: Present Value) của SV giảm xuống = 0 thì Timer thứ N tác động(tức tiếp điểm của nó = ON).

Khi Timer đang hoạt động, nếu ta cho ngõ vào I = OFF thì giá trị PV sẽ quay về giá trị SV ban đầu.

Thời gian để TIM N tác động được tính: t = SV x 0,1 (s)

4/ Ví dụ:

a) Ví dụ 1: mạch ON delay 5s. Viết sang lệnh gợi nhớ cho chương trình sau:

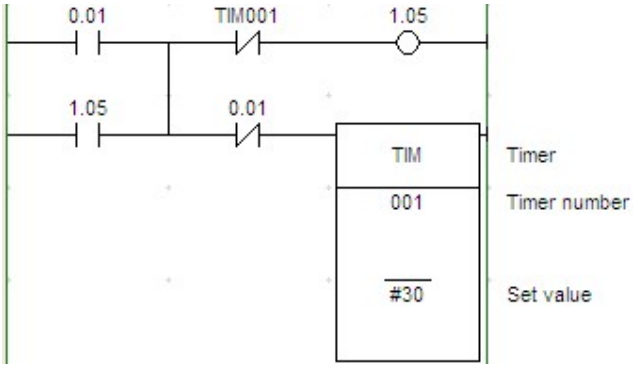


Hoạt động:

Khi ngõ vào 0.00 = ON, Ngõ ra 1.00 = 0N, 1.02 = OFF. 5s sau thì bộ TIM 000 tác động. Khi đó, tiếp điểm TIM000 đóng, ngõ ra 100.02 sẽ = ON.

Như vậy, ngõ ra 1.02 bị trễ 5s sau khi ngõ vào tác động(0.00 = ON)

b) Ví dụ 2: mạch OFF delay 3s. Viết sang lệnh gợi nhớ cho chương trình sau:



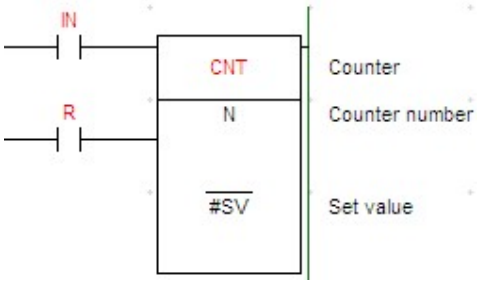
Hoạt động:

Khi công tắc 0.01 = ON, ngõ ra 1.05 = ON.

Khi 0.01 chuyển sang OFF, thì TIM 001 bắt đầu hoạt động. Sau 3s, bộ TIM 001 tác động (tức = ON) làm cho tiếp điểm TIM001= OFF nên ngõ ra 1.05 = OFF.

VIII. Lệnh CNT: COUNTER

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD I

LD R

CNT N

#SV

N: 000 ÷ 511(CQM1); 0000÷ 4095(CJ1M)

SV: 0000 ÷ 9999(BCD)

3/ Hoạt động:

Khi ngõ vào I chuyển từ OFF sang ON(tạo 1 xung) thì giá tri PV của SV giảm đi 1 đơn vị. Khi PV = 0 thì CNT thứ N tác động làm cho tiếp điểm của nó lên ON.

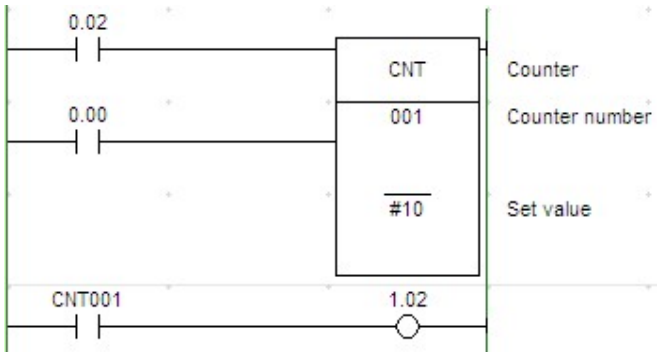
Khi CNT đang hoạt động hay đang tác động, nếu ngõ vào R = ON sẽ đặt lại giá trị PV về giá trị SV ban đầu.

Lưu ý:

Trong một chương trình, chúng ta không nên đặt tên số TIM và CNT trùng nhau.

Số TIM hay CNT 000 có thể sử dụng nhiều lần trong một chương trình. Nhưng khi sử dụng các lệnh này tốc độ xử lý sẽ chậm hơn khi sử dụng TIM/CNT có số khác 000.

4/ Ví dụ: Viết sang lệnh gợi nhớ cho chương trình sau:



Hoạt động:

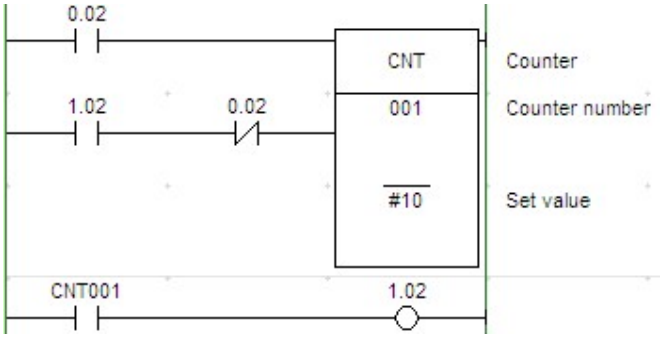
Khi ngõ vào 0.02 chuyển từ OFF sang ON 10 lần thì bộ CNT 001 tác động làm cho tiếp điểm CNT001 ON, dẫn đến 1.02 = ON.

Nếu ngõ vào 0.02 tiếp tục chuyển từ OFF sang ON thì ngõ ra 1.02 vẫn ON.

1.02 chỉ về OFF khi ngõ vào 0.00 = ON

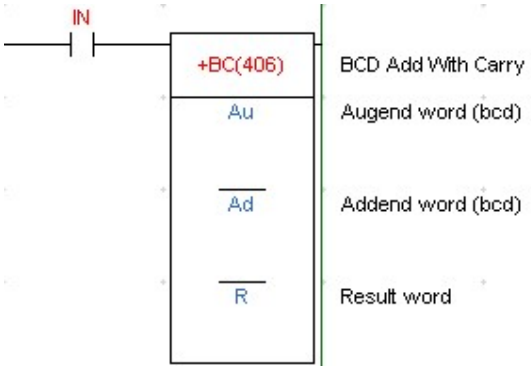
Nếu ta dùng chương trình trên để đếm sản phẩm theo yêu cầu: cứ 10 sản phẩm thì ngõ ra 1.02 báo 1 lần thì sẽ không thực hiện được vì ngõ vào Reset không thực hiện tự động.

Chương trình sau sẽ thực hiện yêu cầu trên. Lý do sao làm được ư? Các bạn tự tìm hiểu nhé! Đơn giản mà!!!



IX. Lệnh Cộng số BCD: +BC(406)

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD	I
+BC(406)	#Au
	#Ad
	R

3/ Hoạt động:

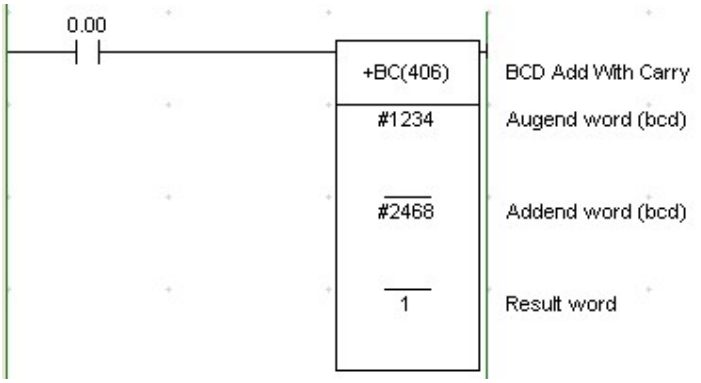
Khi ngõ vào I = ON, Lệnh +BC sẽ thực hiện cộng nội dung của Au với nội dung của Ad và CY. Kết quả đưa ra kênh R.

Bit CY(Carry Flag = 25504) sẽ ON nếu kết quả phép cộng lớn hơn 9999(BCD)

Sau đó, Khi ngõ vào I = OFF……

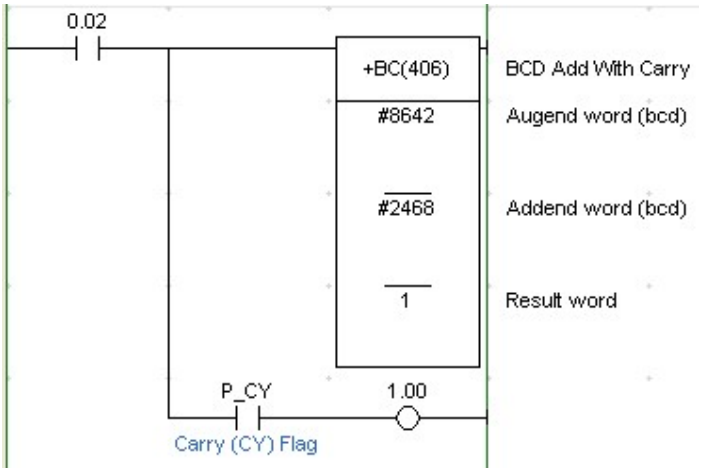
4/ Ví dụ:

a) Viết sang lệnh gợi nhớ cho chương trình sau:



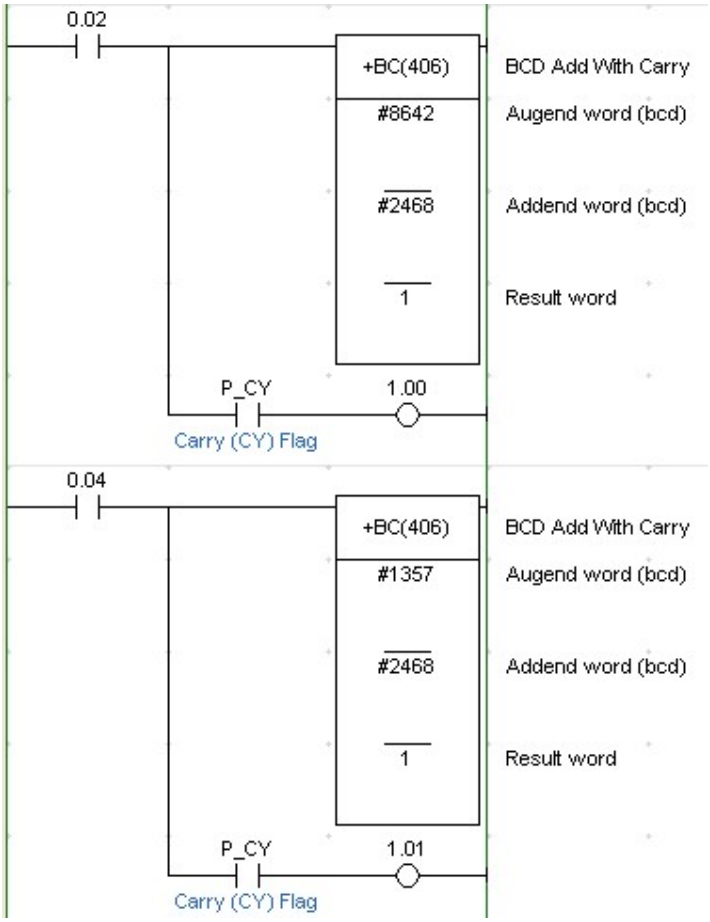
Khi 0.00 = ON, ngõ ra nào 'ON'?

b) Viết sang lệnh gợi nhớ cho chương trình sau:



Khi 0.02 = ON, ngõ ra nào 'ON'?

c) Viết sang lệnh gợi nhớ cho chương trình sau:



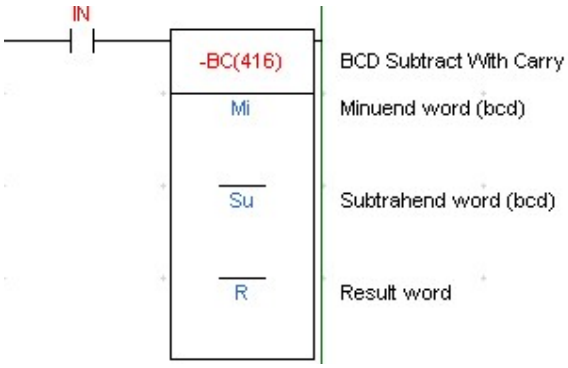
Khi 0.02 = ON, ngõ ra nào 'ON'?

Sau đó, 0.02 = OFF và 0.04 = ON, ngõ ra nào 'ON'?

Nếu cả 0.02 và 0.04 = ON thì ngõ ra nào ON?

X. Lệnh trừ số BCD: -BC(416):

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD	I
-BC(416)	#Mi
	#Su
	R

3/ Hoạt động:

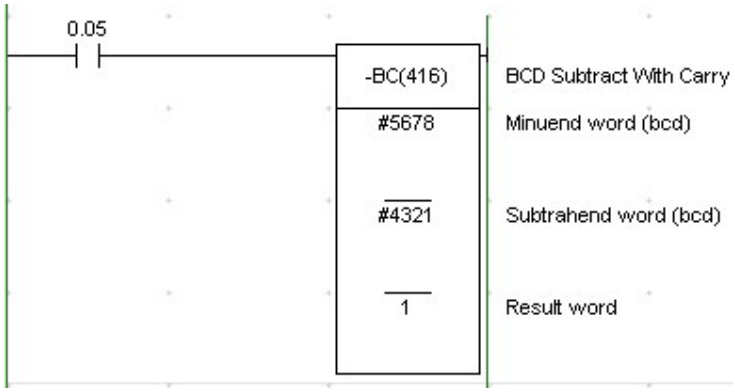
Khi ngõ vào I = ON, Lệnh -BC sẽ thực hiện trừ nội dung của Mi với nội dung của Su và CY. Kết quả đưa ra kênh R.

Bit CY(Carry Flag = 25504) sẽ ON nếu kết quả phép trừ là số âm

Sau đó, Khi ngõ vào I = OFF……

4/ Ví dụ:

a) Viết sang lệnh gợi nhớ cho chương trình sau:

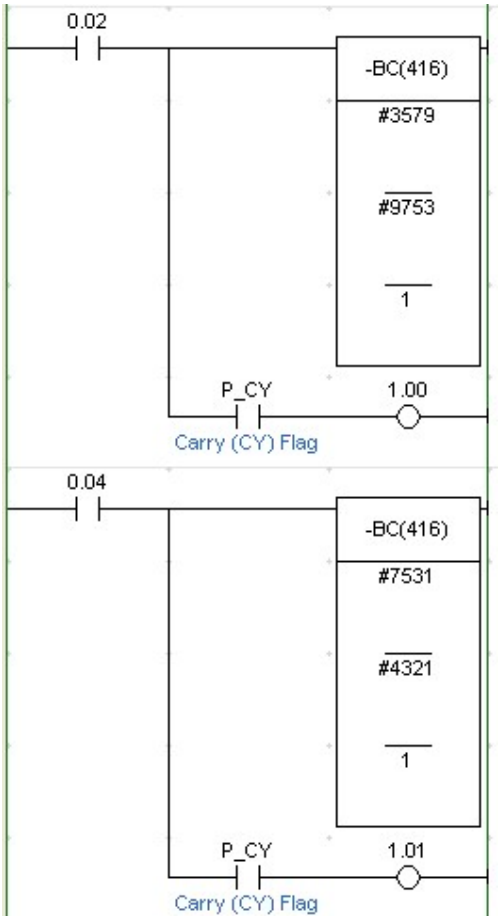


Khi 0.05 = ON, ngõ ra nào 'ON'?

b) Viết sang lệnh gợi nhớ cho chương trình sau:

Khi 0.02 = ON, ngõ ra nào 'ON'?

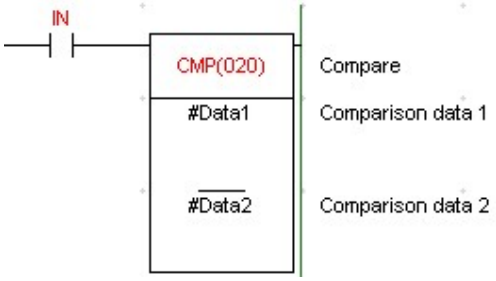
c) Viết sang lệnh gợi nhớ cho chương trình sau:



Khi 0.02 = ON, ngõ ra nào 'ON'? Sau đó, 0.02 = OFF và 0.04 = ON, ngõ ra nào 'ON'?

XI. Lệnh CMP(020)

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD	I
CMP(020)	#Data1
	#Data 2

3/ Hoạt động:

Khi ngõ vào I = ON, Lệnh CMP sẽ thực hiện so sánh nội dung của Data1 với nội dung của Data2.

Nếu nội dung của Data1 > Data2 thì cờ P_GT(Greater than) = CF005 sẽ ON.

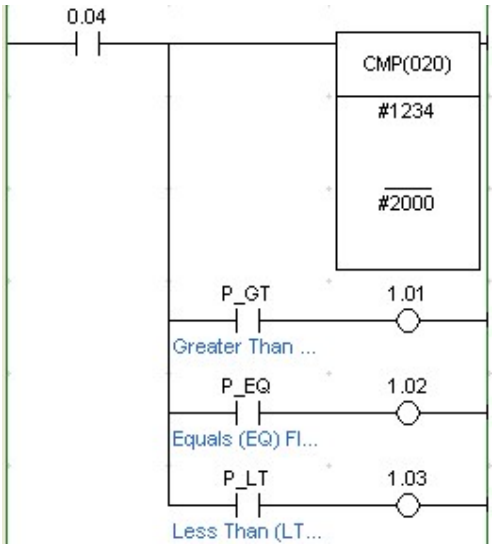
Nếu nội dung của Data1 = Data2 thì cờ P_EQ(Equals) = CF006 sẽ ON.

Nếu nội dung của Data1 < Data2 thì cờ P_LT(Less than) = CF007 sẽ ON.

Sau đó, Khi ngõ vào I = OFF……

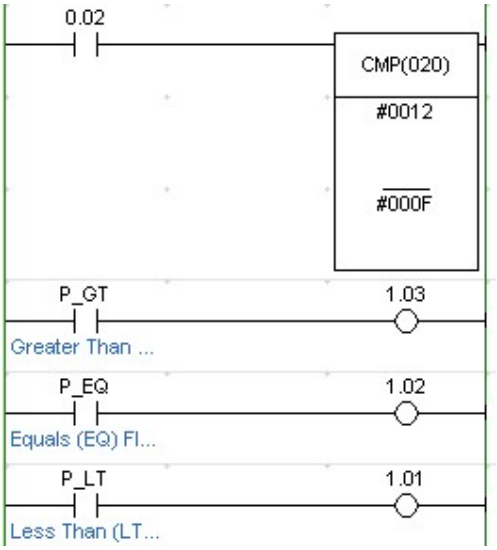
4/ Ví dụ:

a) Viết sang lệnh gợi nhớ cho chương trình sau:



Khi 0.04 = ON, ngõ ra nào 'ON'?

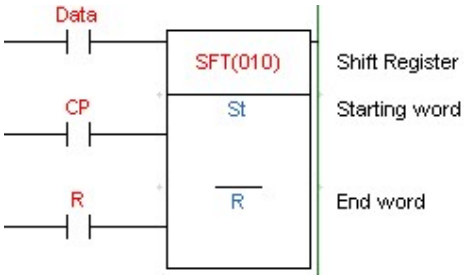
b) Viết sang lệnh gợi nhớ cho chương trình sau:



Khi 0.02 = ON, ngõ ra nào 'ON'?

XII. Lệnh SFT(010): Shift Register

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD I

LD CP

LD R

SFT(010) St

E

3/ Hoạt động:

Cứ mỗi lần xung Cp nhịp, dữ liệu ở ngõ vào IN(Data in) sẽ được dịch từ St sang E

- Nếu dữ liệu vào IN có giá trị [0] thì dữ liệu [0] này sẽ được đưa ra kệnh E.

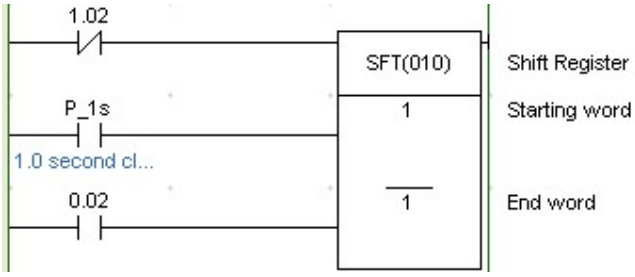
- Nếu dữ liệu vào IN có giá trị [1] thì dữ liệu [1] này sẽ được đưa ra kệnh E.

Nếu R = [1] thì giá trị của các ngõ ra kênh E sẽ bị xóa về [0]

Cách thức dịch này giống như thanh ghi dịch vào nối niếp ra song song của phần Thanh ghi trong môn học Kỹ thuật số.

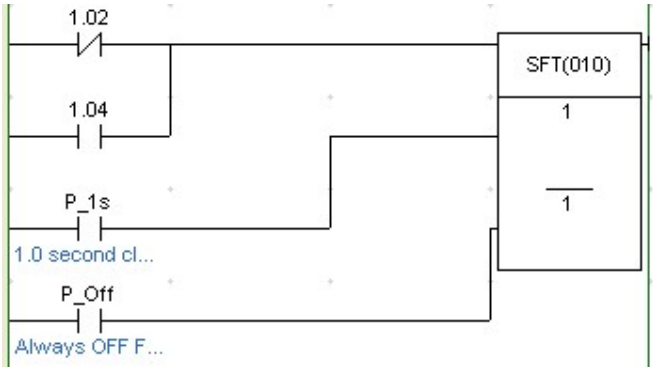
4/ Ví dụ:

a) Viết sang lệnh gợi nhớ cho chương trình sau:



Mạch trên hoạt động như thế nào?

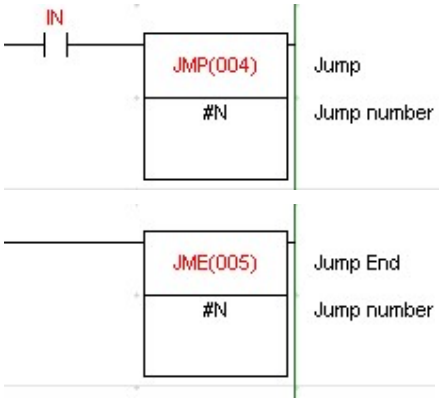
b) Viết sang lệnh gợi nhớ cho chương trình sau:



Mạch trên hoạt động như thế nào?

XIII. Lệnh JMP(004) – JME(005): Jump – Jump End

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD I

JMP(004) #N

(Nội dung của chương trình rẽ nhánh)

JME(005) #N

3/ Hoạt động:

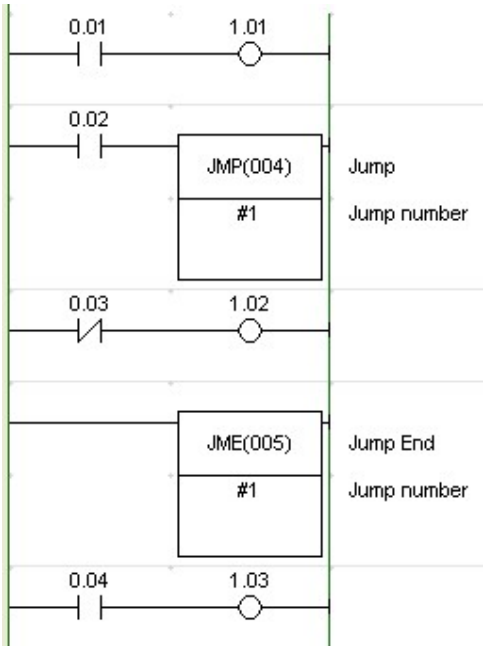
Khi ngõ vào I tác động, lệnh rẽ nhánh mới có tác dụng. Nghĩa là nó sẽ thực hiện nội dung nằm trong lệnh JMP – JME.

Khi ngõ vào I không tác động, lệnh rẽ nhánh không có tác dụng. Khi đó, chương trình sẽ bỏ qua đoạn chương trình và không thực hiện nội dung nằm trong lệnh JMP – JME.

Lưu ý: Khi chương trình rẽ nhánh đang thực hiện, sau đó, nếu ta không cho thực hiện nữa thì nội dung của nó được giữ nguyên.

4/ Ví dụ:

a) Viết sang lệnh gợi nhớ cho chương trình sau:



b) Hoạt động:

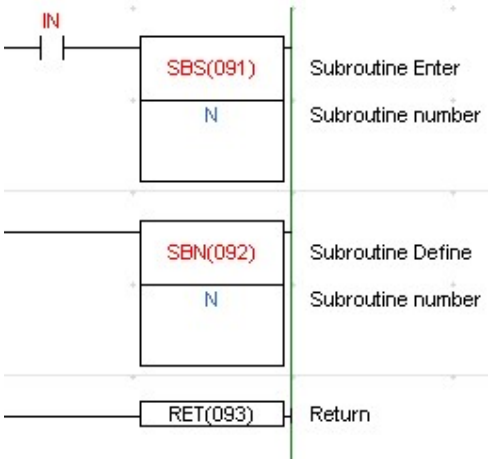
Khi chỉ 0.01 = ON, ngõ ra nào ON?.....

Khi chỉ 0.01, 0.02 = ON, ngõ ra nào ON?.....

Khi chỉ 0.04 = ON, ngõ ra nào ON?.....

XIV. Lệnh chương trình con (*)

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

LD I

SBS(091) #N

(Nội dung chương trình chính)

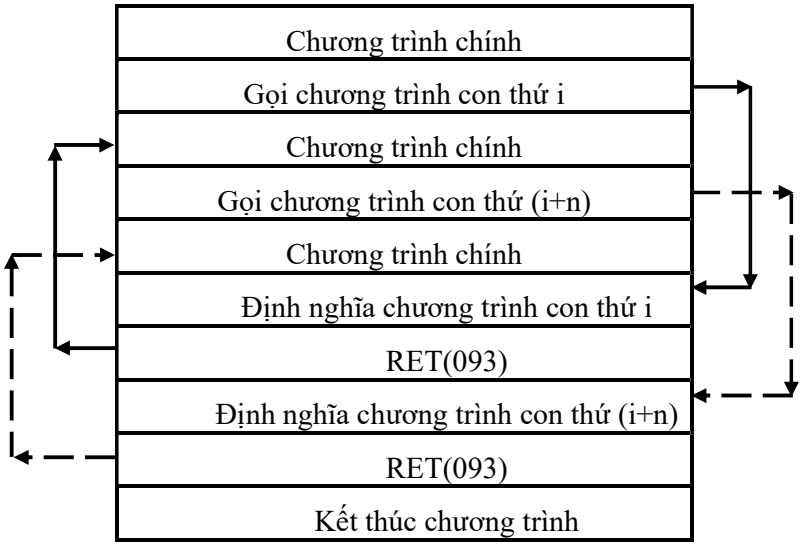
SBN(092)#N

(Nội dung chương trình con)

RET(093)

3/ Hoạt động:

Để dễ hiểu cách gọi chương trình con, ta khảo sát lưu đồ sau:



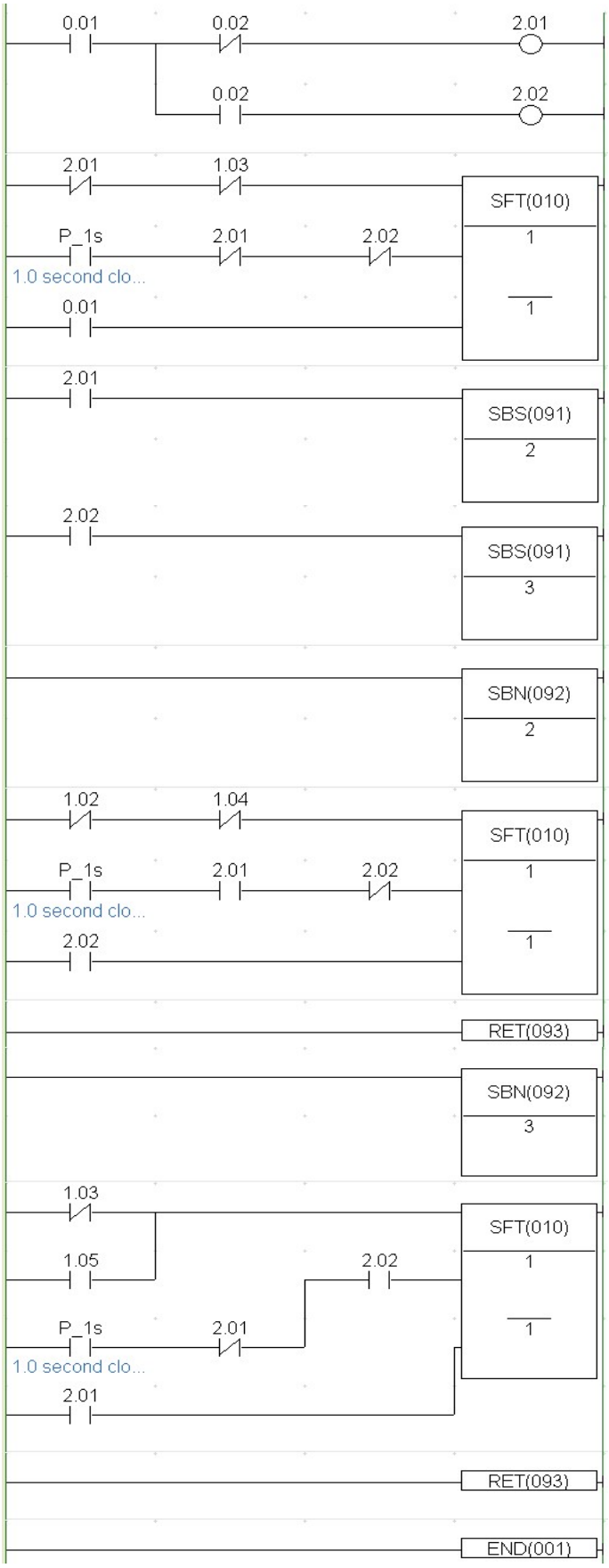
Lưu ý:

Trong một chương trình ta có thể sử dụng nhiều chương trình con cùng số 00. nhưng khi sử dụng như vậy tốc độ xử lý sẽ chậm.

Một chương trình có nhiều chương trình con hay một chương trình con thì các chương trình con này đều để ở cuối chương trình chính.

4/ Ví dụ:

a) Viết sang lệnh gợi nhớ cho chương trình sau:



b) Hoạt động:

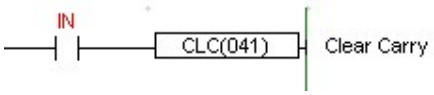
Khi 0.01, 0.02 = OFF, chương trình hoạt động như thế nào?.....

Khi 0.01 = ON, 0.02 = OFF, chương trình hoạt động như thế nào?.....

Khi 0.01, 0.02 = ON, chương trình hoạt động như thế nào?.....

XV. Lệnh CLC(041)

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

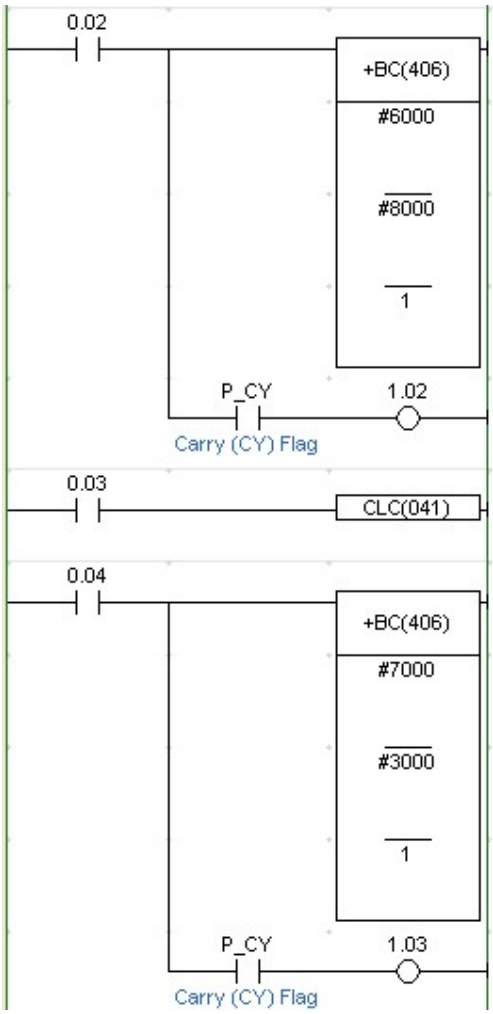
LD I

CLC(041)

3/ Chức năng:

Dùng để xóa cờ nhớ.

4/ Ví dụ:



Hoạt động:

Khi 02 = ON? …….

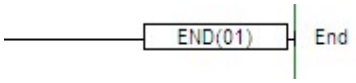
Khi 02 và 03 = ON? …….

Khi 02 và 04 = ON? …….

Khi 02;03 và 04 = ON? ……

XVI. Lệnh END

1/ Ký hiệu:



2/ Lệnh gợi nhớ:

END(01)

3/ Chức năng:

Dùng để kết thúc một chương trình.

Trong một chương trình luôn bắt đầu bằng lệnh LD hay LDNOT và kết thúc bằng lệnh END. Nếu sau lệnh END ta có viết bao nhiêu lệnh đi chăng nữa thì các lệnh này không được thực hiện.

XVII. Bài tập

1/ Thiết kế mạch điều khiển 2 động cơ theo yêu cầu sau:

Khi nhấn Start thì động cơ 1 hoạt động, 5s sau động cơ 2 hoạt động.

Khi nhấn Stop thì động cơ 1 ngừng hoạt động, 3s sau động cơ 2 ngừng hoạt động.

Vẽ biểu đồ hình thang cho chương trình.

2/ Thiết kế mạch điều khiển 2 động cơ theo yêu cầu sau:

Khi nhấn Start thì động cơ 1 hoạt động, 6s sau động cơ 2 hoạt động.

Khi nhấn Stop thì động cơ 2 ngừng hoạt động, 4s sau động cơ 1 ngừng hoạt động.

Vẽ biểu đồ hình thang cho chương trình.

3/ Thiết kế mạch ưu tiên cho 3 chơi theo yêu cầu sau:

Khi giám khảo cho phép, đội nào nhấn trước sẽ ưu tiên. Đèn báo ưu tiên của đội đó sẽ sáng trong 3s rồi tắt.

Nếu đội đó nhấn tiếp hay đội khác nhấn thì cũng không có đội nào nào ưu tiên.

Chế độ ưu tiên sẽ được hoạt động lại khi giám khảo cho phép lần sau(mỗi lần giám khảo cho phép chỉ có 1 đội nhấn trước được ưu tiên)

Vẽ biểu đồ hình thang cho chương trình.

Cột 1:	**Xanh 1(1.00)**	**Vàng 1(1.01)**	**Đỏ 1(1.02)**	
Cột 2:	**Đỏ 2(1.05)**		**Xanh 2(1.03)**	**Vàng(1.04)**

4/ Thiết kế đèn giao thông cho ngã 4 theo yêu cầu sau:

Vẽ biểu đồ hình thang cho chương trình.

BÀI 6:

ỨNG DỤNG PLC TRONG HỆ THỐNG ĐIỀU KHIỂN

I. Đèn giao thông Nâng Cao

1/ Yêu cầu:

Thiết kế đèn giao thông cho ngã 4 theo yêu cầu sau:

- Khi có cảnh sát thì đèn vàng của 2 cột chớp tắt có chu kì 1s
- Khi không có cảnh sát thì các đèn hoạt động theo trình tự sau:

2/ Biểu đồ hình thang:

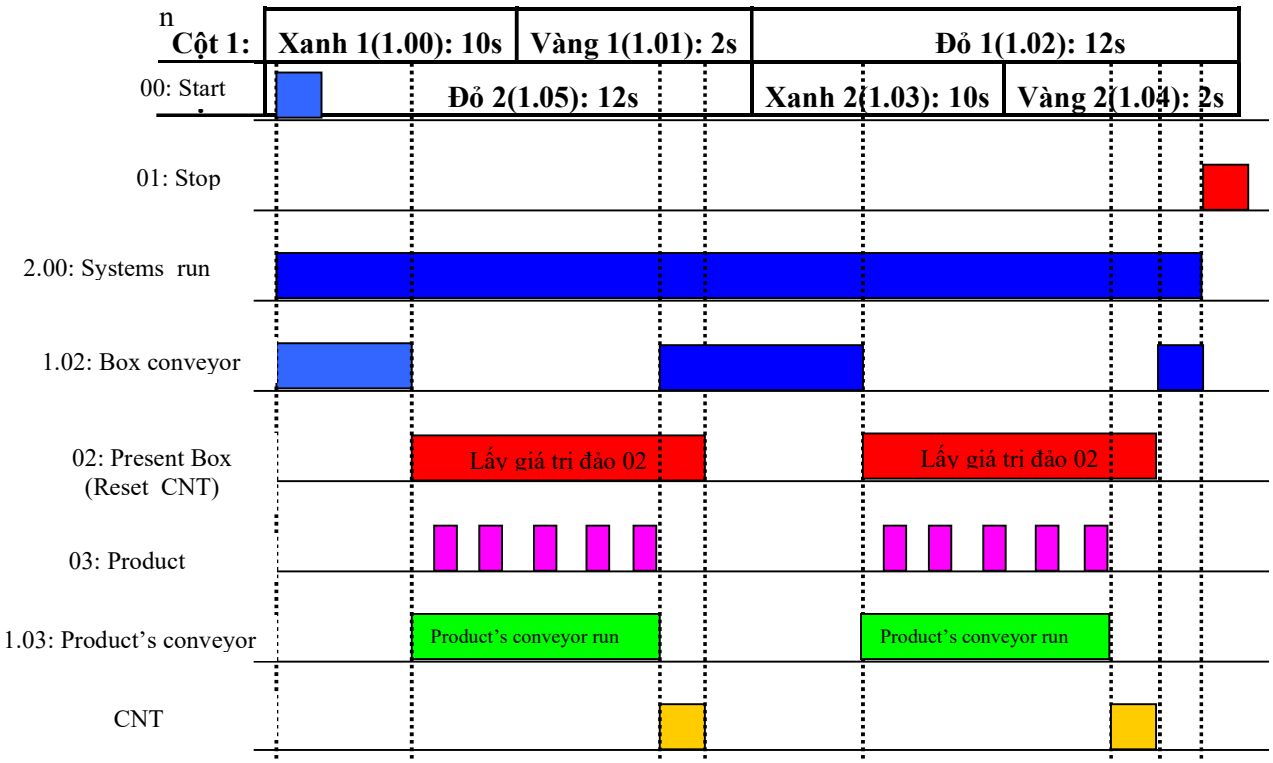
3/ Lệnh gợi nhớ:

II. Dây chuyền đóng thùng tự động

1/ Yêu cầu:

Khi nút khởi động được nhấn (00: Start Push Button) dây chuyền vận chuyển Thùng (1.02) sẽ hoạt động để đưa thùng tới vị trí. Sensor phát hiện thùng (02) sẽ nhận diện, làm cho dây chuyền đưa thùng tới sẽ dừng, lúc này dây chuyền vận chuyển sản phẩm (1.03) sẽ hoạt động. Sensor phát hiện sản phẩm(03) sẽ đếm số sản phẩm đi qua. Khi 03 đếm được 10 sản phẩm thì dây chuyền Sản phẩm sẽ dừng, và dây chuyền Thùng sẽ hoạt động trở lại.

Mạch đếm sẽ được Reset và dây chuyền sẽ được lặp lại nếu nút nhấn Stop (01) được n



hấn.

2/ Biểu đồ hình thang:

3/ Lệnh gợi nhớ:

III. Phân loại sản phẩm

1/ Yêu cầu:

Khi nút khởi động được nhấn (00: Start Push Button) hệ thống hoạt động. Khi nhấn nút Stop(01) thì hệ thống sẽ dừng.

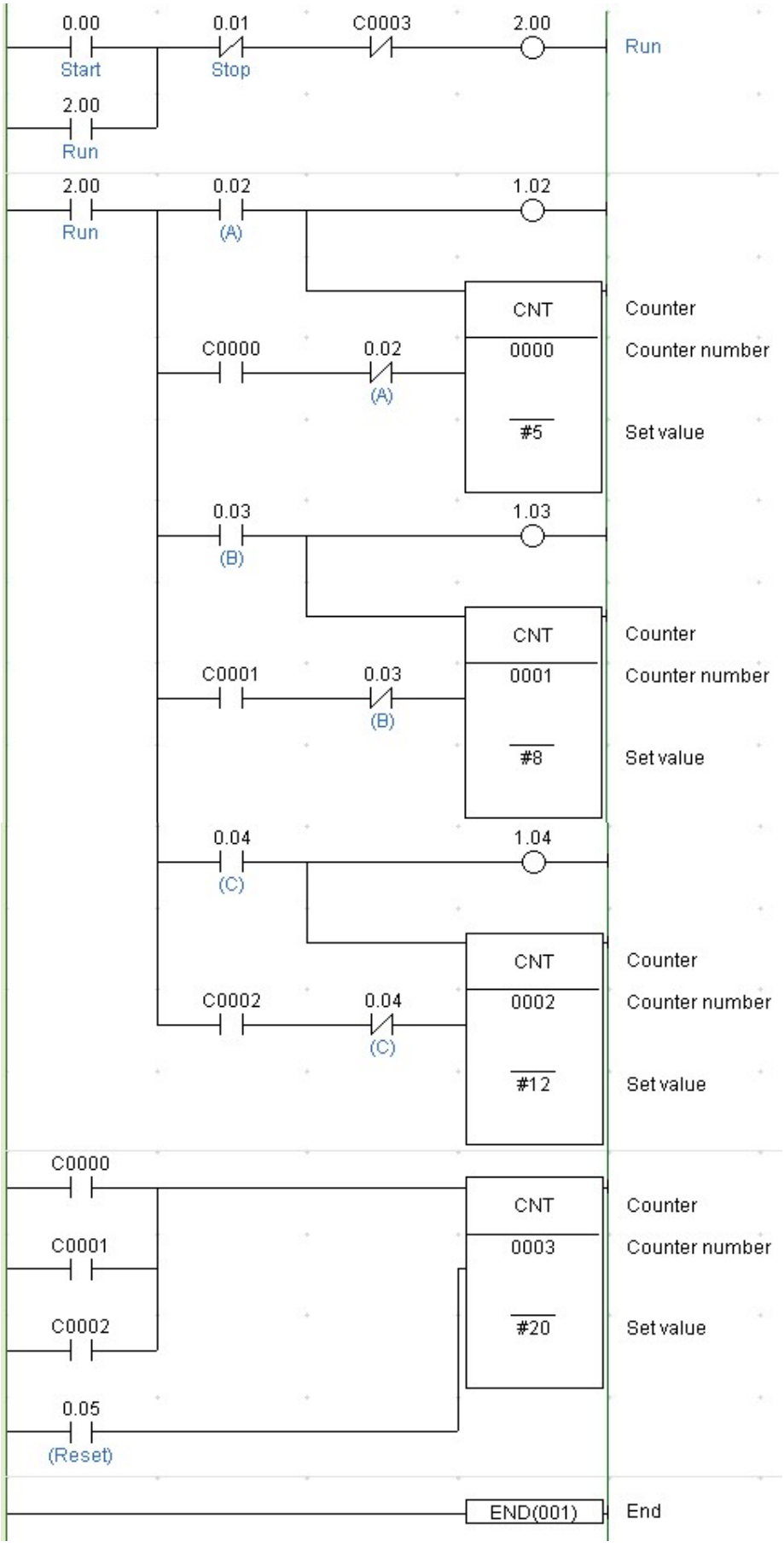
Nếu sản phẩm loại A được phát hiện (Sensor 1 sẽ báo) thì 5 sản phẩm đóng thành 1 thùng.

Nếu sản phẩm loại B được phát hiện(Sensor 2 sẽ báo) thì 8 sản phẩm đóng thành 1 thùng.

Nếu sản phẩm loại C được phát hiện (Sensor 3 sẽ báo) thì 12 sản phẩm đóng thành 1 thùng.

Hệ thống sẽ tạm dừng khi đếm được 20 thùng, lúc này không đếm sản phẩm nữa. Khi nhấn nút Reset(03) thì việc đếm sản phẩm sẽ tiếp tục.

2/ Biểu đồ hình thang:


0.00
Start
0.01
Stop
C0003
2.00
Run
2.00
Run
2.00
Run
0.02
(A)
1.02
CNT
Counter
C0000
0.02
(A)
0000
Counter number
#5
Set value
0.03
(B)
1.03
CNT
Counter
C0001
0.03
(B)
0001
Counter number
#8
Set value
0.04
(C)
1.04
CNT
Counter
C0002
0.04
(C)
0002
Counter number
#12
Set value
C0000
C0001
C0002
CNT
Counter
0003
Counter number
#20
Set value
0.05
(Reset)
END(001)
End

3/ Lệnh gợi nhớ:

IV. Khoan cắt sản phẩm

1/ Yêu cầu:

Khi nút khởi động được nhấn (00: Start Push Button) hệ thống khoan(1.00) hoạt động. Khi nhấn nút Stop(01) thì hệ thống khoan sẽ dừng.

Nếu sản phẩm khoan là loại A thì 10 sản phẩm sẽ thay dao.

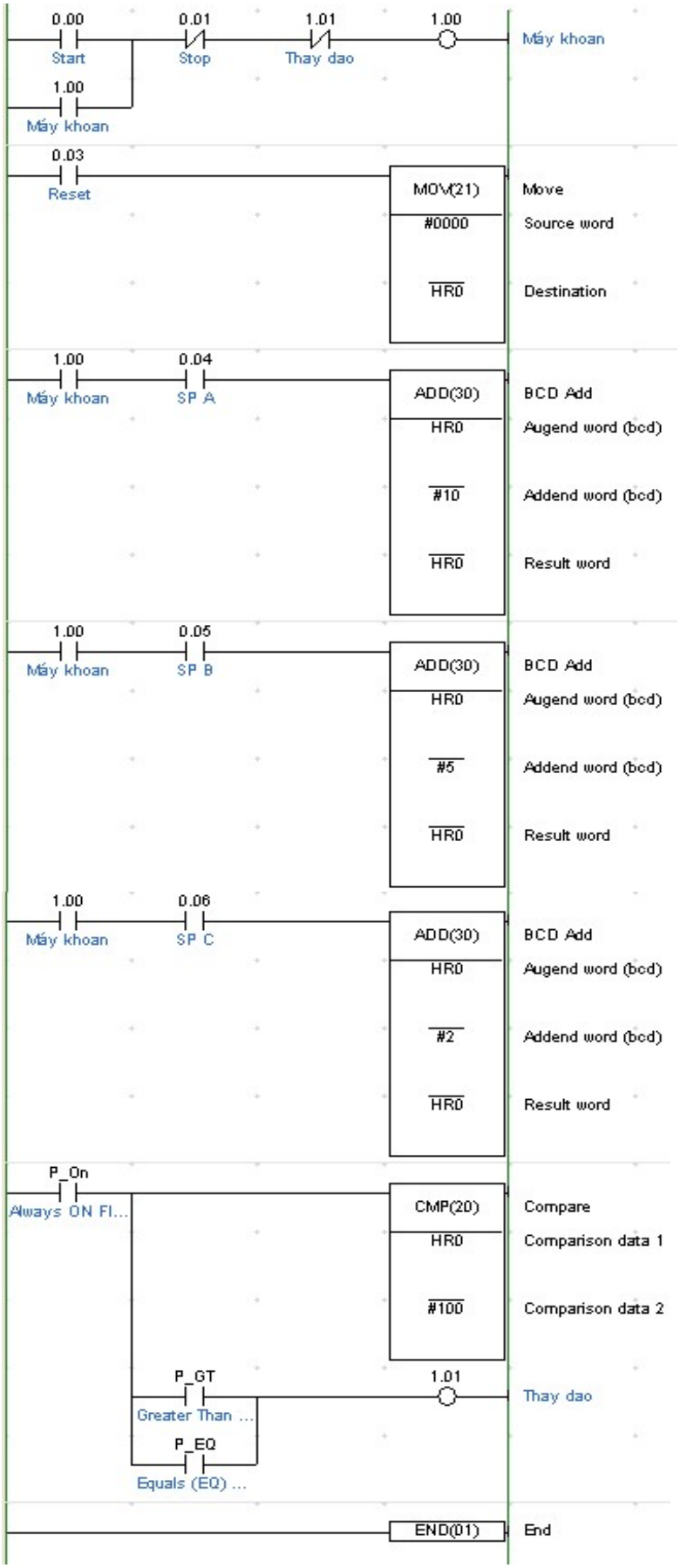
Nếu sản phẩm khoan là loại B thì 20 sản phẩm sẽ thay dao.

Nếu sản phẩm khoan là loại C thì 50 sản phẩm sẽ thay dao.

Khi đang thay dao thì hệ thống khoan sẽ dừng. Khi thay dao xong, nhấn nút Reset(03) hệ thống khoan sẽ hoạt động lại

2/ Biểu đồ hình thang:

3/ Lệnh gợi nhớ:


0.00
Start
0.01
Stop
1.01
Thay dao
1.00
Máy khoan
1.00
Máy khoan
0.03
Reset
MOV(21)
Move
#0000
Source word
HR0
Destination
1.00
Máy khoan
0.04
SP A
ADD(30)
BCD Add
HR0
Augend word (bcd)
#10
Addend word (bcd)
HR0
Result word
1.00
Máy khoan
0.05
SP B
ADD(30)
BCD Add
HR0
Augend word (bcd)
#5
Addend word (bcd)
HR0
Result word
1.00
Máy khoan
0.06
SP C
ADD(30)
BCD Add
HR0
Augend word (bcd)
#2
Addend word (bcd)
HR0
Result word
P_On
Always ON Fl...
CMP(20)
Compare
HR0
Comparison data 1
#100
Comparison data 2
P_GT
Greater Than ...
P_EQ
Equals (EQ) ...
1.01
Thay dao
END(01)
End

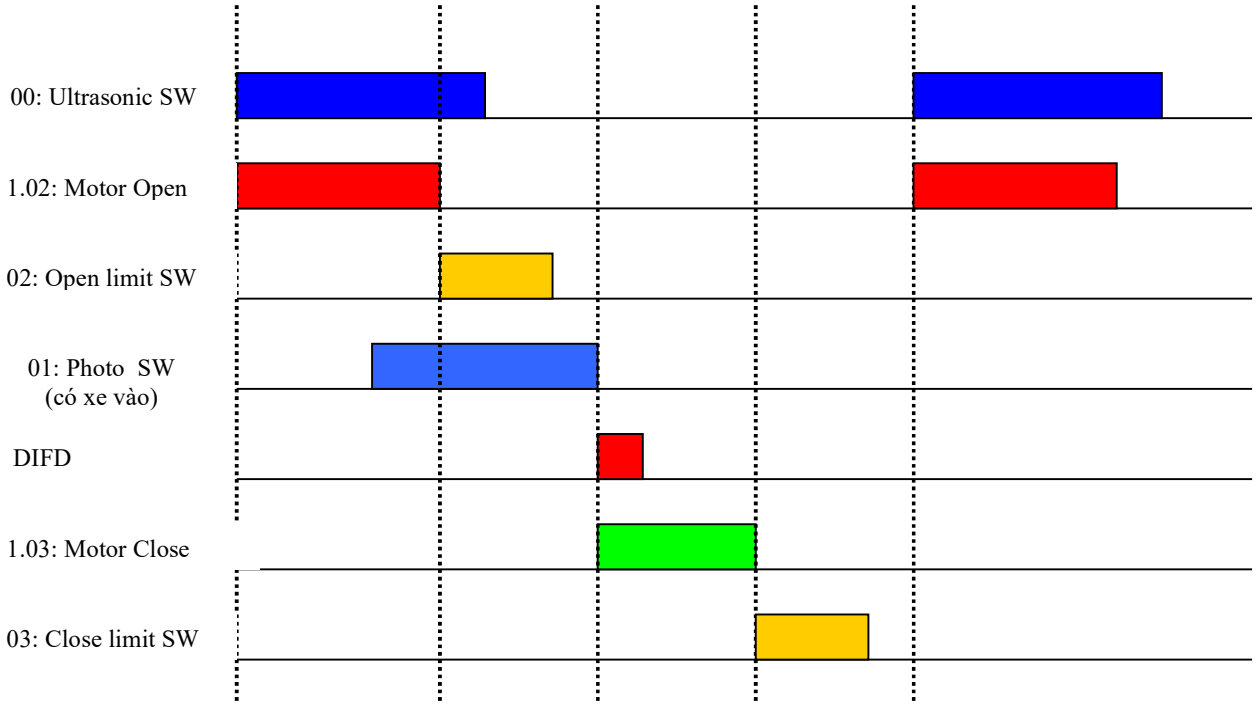
V. Đóng/mở cửa tự động

1/ Yêu cầu:

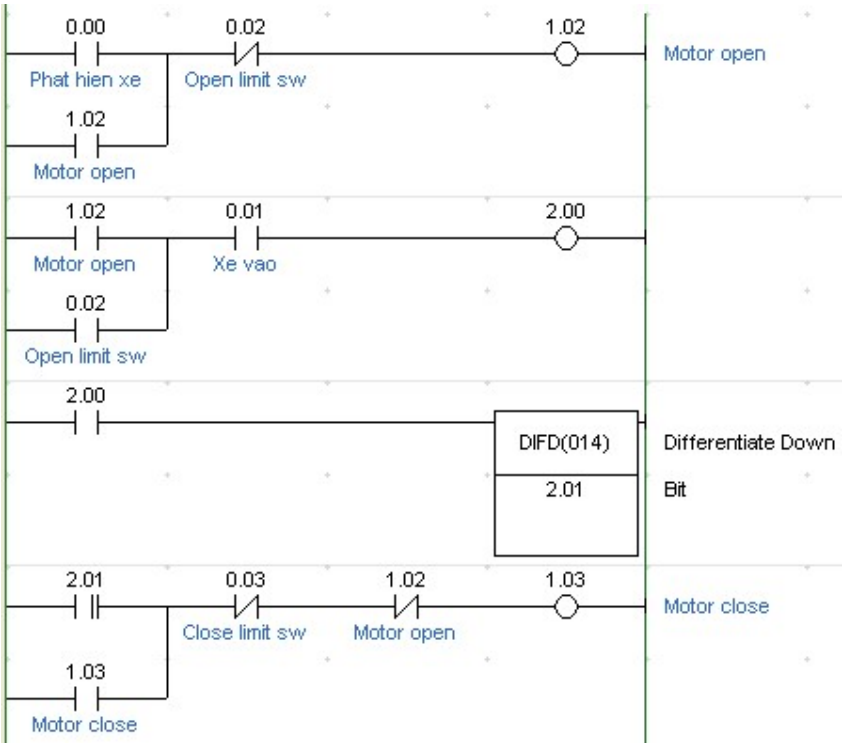
Khi sensor 00 báo có xe, động cơ 1.02 sẽ hoạt động mở cửa ra. Khi của di chuyển chạm vào SW giới hạn mở sẽ dừng.

Khi của kéo mở ra thì cho xe có thể vào cổng, 01 cảm nhận có xe vào cổng.

Khi 01 không còn nhận được xe thì DIFD hoạt động, khởi tạo cho động cơ kéo cửa đóng cổng lại. khi cửa đóng lại chạm vào SW giới hạn dưới sẽ dừng.



2/ Biểu đồ hình thang:



3/ Lệnh gợi nhớ:

VI. Điều khiển bãi đậu xe (*)

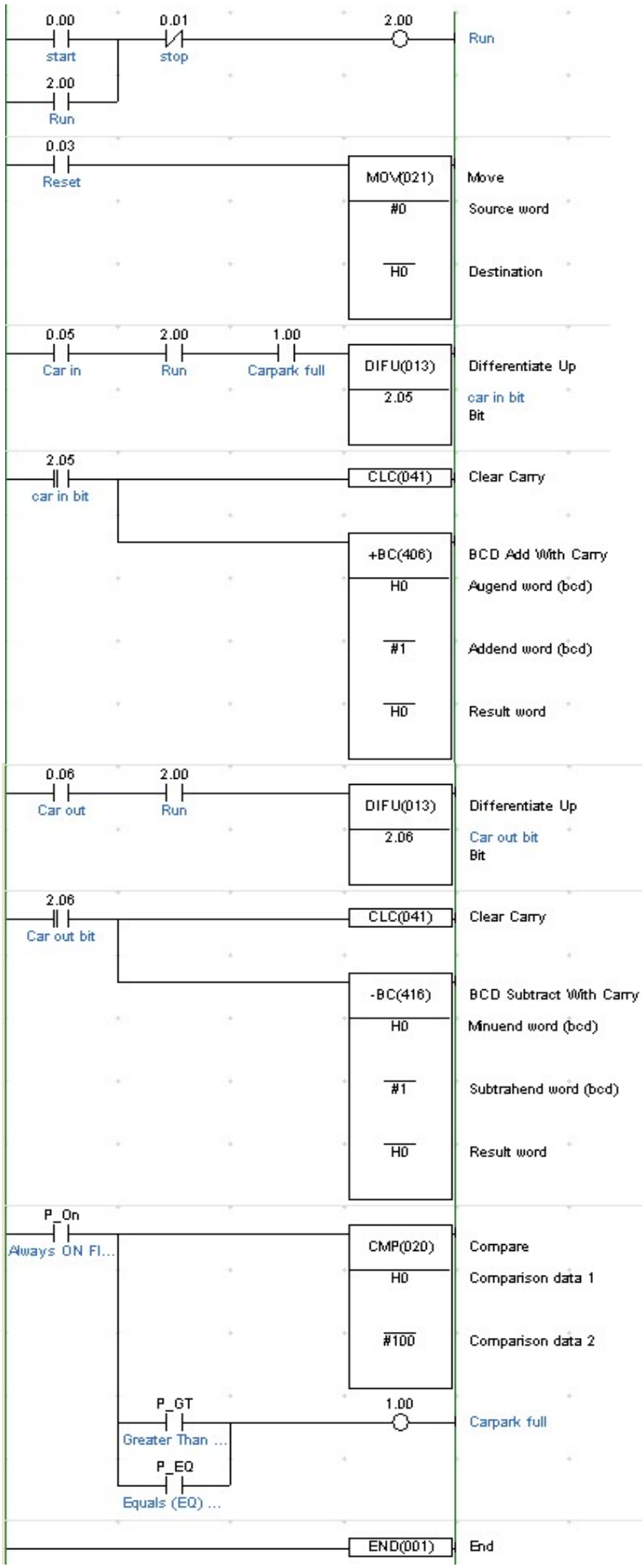
1/ Yêu cầu:

Khi nhấn Start, hệ thống hoạt động. Nhấn Stop hệ thông dừng.

Nếu có xe vào thì bộ đếm sẽ tăng lên 1. Nếu có xe ra thì bộ đếm sẽ giảm đi 1.

Khi bãi đầy xe sẽ không cho xe vào, chỉ cho xe ra.

2/ Biểu đồ hình thang:


0.00
start
0.01
stop
2.00
Run
2.00
Run
0.03
Reset
MOV(021)
#0
H0
Move
Source word
Destination
0.05
Car in
2.00
Run
1.00
Carpark full
DIFU(013)
2.05
Differentiate Up
car in bit
Bit
2.05
car in bit
CLC(041)
Clear Carry
+BC(406)
H0
#1
H0
BCD Add With Carry
Augend word (bcd)
Addend word (bcd)
Result word
0.06
Car out
2.00
Run
DIFU(013)
2.06
Differentiate Up
Car out bit
Bit
2.06
Car out bit
CLC(041)
Clear Carry
-BC(416)
H0
#1
H0
BCD Subtract With Carry
Minuend word (bcd)
Subtrahend word (bcd)
Result word
P_On
Always ON Fl...
CMP(020)
H0
#100
Compare
Comparison data 1
Comparison data 2
P_GT
Greater Than ...
P_EQ
Equals (EQ) ...
1.00
Carpark full
END(001)
End

3/ Lệnh gợi nhớ:

VII. Một số đề xuất

Môn học này mang tính ứng dụng, tùy vào khả năng của người thiết kế mà chương trình có sự khác nhau. Nhưng kết quả mới là quan trọng. Do đó, các bạn đừng ngại khó mà không làm các bài tập. Có thể bạn là người đưa ra ý tưởng mới thì sao?