Chương 2

Các lệnh lập trình bậc thang và mnemonic

2. Bước đầu với lập trình (Programming)

2.1 Các chế đô làm việc của PLC

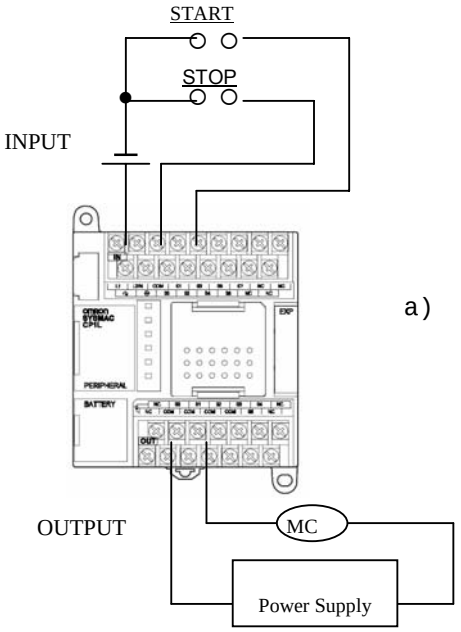
PLC có thể được đặt một trong 3 chế độ từ phần mềm lập trình CX-Programmer.

3 chế độ làm việc của PLC

- **PROGRAM** mode : Là chế độ dùng khi viết chương trình hay thực hiện các thay đổi hoặc sửa đổi đối với chương trình hiện hành
- **MONITOR** mode : Là chế độ được dùng khi thay đổi nội dung bộ nhớ trong khi PLC đang chạy (Run).
- **RUN** mode : Là chế độ dùng để thực hiện (chạy) chương trình mà ta đã lập và nạp vào PLC. Chương trình bên trong PLC không thể được thay đổi khi đang ở trong chế độ này.

Theo mặc định, PLC của Omron đều có thể được lập trình song song bằng 2 ngôn ngữ: Dòng lệnh (Statement List hay mnemonic code) & Sơ đồ bậc thang (Ladder diagram). Trong tài liệu này sẽ chủ yếu trình bày về Sơ đồ bậc thang, kèm theo bên cạnh là các lệnh tương ứng tương đương dạng Dòng lệnh (Statement List).

2.1.1) Ví dụ về một mạch tự giữ (self-holding)



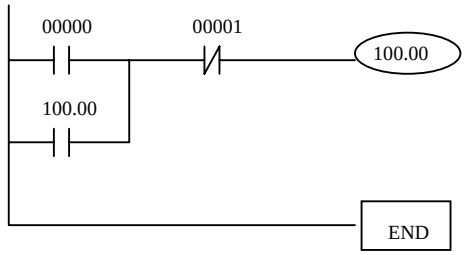
a)

Input	Thiết bị ngoài	Output	Thiết bị

00000	Nút bấm Start
00001	Nút bấm Stop

	ngoài
100.00	Motor

Ladder Diagram



b)

Mnemonic Codes

Đ. chỉ	Lệnh	Th. số
00000	LD	00000
0001	OR	100.00
0002	AND NOT	00001
0003	OUT	100.00
0004	END(01)	

c)

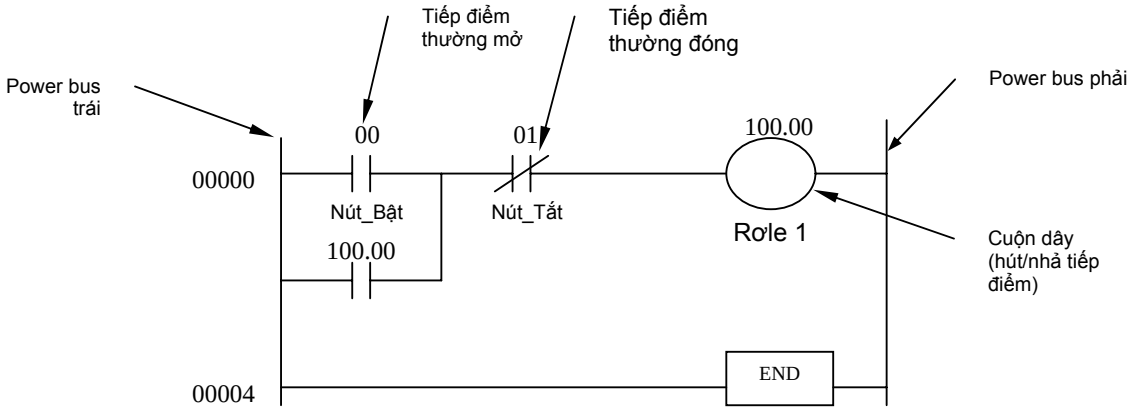
Hình 24: a) Sơ đồ nối PLC với mạch bên ngoài
b) Chương trình dạng ngôn ngữ bậc thang (Ladder Diagram)
c) Mã chương trình dạng Mnemonic Codes

Chương trình này sẽ đảm bảo đầu ra 100.00 sẽ luôn ở trạng thái ON khi 00000 lên 1 bất kể sau đó trạng thái của đầu vào 00000 như thế nào.

2.1.2) Lập trình bằng SƠ ĐỒ BẬC THANG (LADDER DIAGRAM)

Ban đầu, PLC được sử dụng chủ yếu để thay thế các sơ đồ điện phức tạp gồm rất nhiều các rơle, tiếp điểm, timer, mạch giữ, .. và các phần tử điện trung gian khác làm nhiệm vụ của các mạch logic. Tuy nhiên khi dùng PLC, các phần tử logic trung gian này được thay thế hoàn toàn bằng các sơ đồ điện "ảo" bên trong PLC do người thiết kế lập trình. Việc mô phỏng các sơ đồ điện này được lập bằng một dạng ngôn ngữ điều khiển gọi là sơ đồ bậc thang (LADDER DIAGRAM).

Ví dụ về một sơ đồ bậc thang



Thành phần cơ bản của một sơ đồ bậc thang bao gồm :

- Power bus trái và phải : giống với dây nguồn "nóng" và dây "nguội" của sơ đồ điện. Các power bus này luôn được vẽ thẳng đứng như trên hình.
- Các tiếp điểm thường đóng (NC) và thường mở (NO)
- Các cuộn dây hút/nhả các tiếp điểm khác
- Các phần tử điện khác như timer, counter,.. và các lệnh khác.

Trong sơ đồ này, cuộn dây rơle ngoài cùng bên phải sẽ chỉ nhận được điện từ power bus trái (tức dây "nóng") khi các tiếp điểm đi trước bên trái nó "cho phép" dòng điện đi qua, tức đều đóng. Do vậy các tiếp điểm (và tổ hợp đấu nối của chúng) thường được gọi là *điều kiện thực thi* (execution condition) cho cuộn dây hay các lệnh khác đi sau.
Các cuộn dây, các tiếp điểm và một số các phần tử khác luôn có một địa chỉ trong bộ nhớ để tham chiếu và sử dụng trong chương trình. Địa chỉ này được ghi phía trên ký hiệu của phần tử như trên hình. Còn các tên mô tả chức năng của chúng như Nút_Bật, Nút_Tắt, .. được ghi bên dưới. Địa chỉ của tiếp điểm sẽ điều khiển (đóng/mở) tiếp điểm này; ngược lại, cuộn dây lại điều khiển bật tắt ON/OFF địa chỉ đi kèm của cuộn dây.

2.2 Các lệnh lập trình cơ bản

PLC thường được lập trình bằng một ngôn ngữ mô phỏng giống như sơ đồ điện gọi là Ladder Diagram. Mỗi phần tử của sơ đồ là một lệnh (Instruction). Các lệnh phức tạp thường có một mã lệnh (Code) riêng.

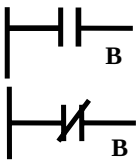
2.2.1) Lệnh tiếp điểm: Load (LD) và Load Not (LD NOT)

Lệnh LOAD hay LOAD NOT là lệnh tiếp điểm thường hở & tiếp điểm thường đóng, dùng làm điều kiện khởi đầu một thang mới trong sơ đồ bậc thang và có chức năng giống với một tiếp điểm của sơ đồ điện. Các tiếp điểm khi nối với các phần tử khác thường đóng vai trò làm **điều kiện thực hiện** (execution condition) cho các phần tử đi sau nó. Lệnh này luôn được gán với một địa chỉ bit xác định trạng thái của tiếp điểm này.
Chú ý là 2 lệnh này luôn luôn nằm ở phía trái nhất của một khối logic trong sơ đồ bậc thang (nghĩa là không có một lệnh nào loại khác được phép nằm ở phía trái của lệnh này trong khối logic).
Có 2 loại:

- Lệnh LD : Tương đương với một tiếp điểm thường mở (Normally Open - NO) trong sơ đồ điện. Khi bit đi kèm là 1 (ON), tiếp điểm sẽ đóng và các phần tử (lệnh) đi sau tiếp điểm sẽ được hoạt động (có điện) và ngược lại khi bit đi kèm là 0 (OFF), tiếp điểm sẽ mở và các phần tử đi sau tiếp điểm sẽ không được hoạt động (không có điện chạy qua tiếp điểm)
- Lệnh LD NOT : Tương đương với một tiếp điểm thường đóng (Normally Closed - NC) trong sơ đồ điện. Khi bit đi kèm là 0 (OFF), tiếp điểm sẽ đóng và các phần tử (lệnh) đi sau tiếp điểm sẽ được hoạt động (có điện) và ngược lại khi bit đi kèm là 1 (ON), tiếp điểm sẽ mở và các phần tử đi sau tiếp điểm sẽ không được hoạt động (không có điện chạy qua tiếp điểm)

LOAD-**LD (Normally open) →**

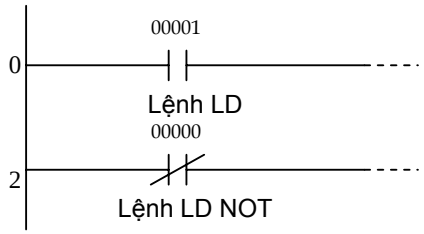
LOAD NOT-**LD NOT (Normally Closed) →**



B : BIT
IR, SR, AR, HR, TC, LR,

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ :



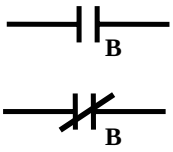
Địa chỉ	Lệnh	Th. số
00000	LD	00000
00001	Lệnh khác...	.....
00002	LD NOT	00000
00003	Lệnh khác	.....

2.2.2) Lệnh tiếp điểm: AND và AND NOT

Lệnh AND (AND NOT) dùng để tạo ra các tiếp điểm thường mở (thường đóng) theo sau (nối tiếp) với các tiếp điểm tạo ra bởi lệnh LD hay LD NOT.

AND-**AND**

AND NOT-**AND NOT**



B : BIT
IR, SR, AR, HR, TC, LR

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ: AND, AND NOT

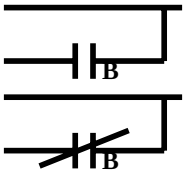
Địa chỉ	Lệnh	Th. Số
00000	LD	00000
00001	AND NOT	00100
00002	AND	LR 00000
00003	Lệnh ..	

2.2.3) Lệnh tiếp điểm: OR, OR NOT

Lệnh OR (OR NOT) tạo ra các tiếp điểm thường mở (thường đóng) nối song song với một nhánh khác.

OR-**OR**

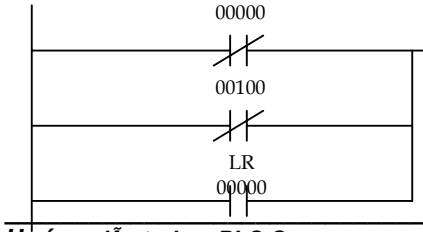
OR NOT-**OR NOT**



B : BIT
IR, SR, AR, HR, TC, LR

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ : OR, OR NOT



Instruction.

Địa chỉ	Lệnh	Tham số

00000	LD NOT	00000
00001	OR NOT	00100
00002	OR	LR 00000
00003	Instruction	

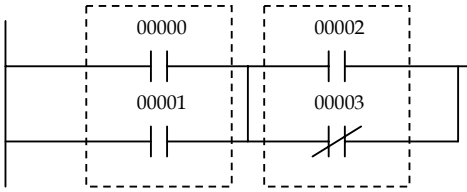
2.2.4) Lệnh AND LD và OR LD

AND LOAD-(**AND LD)** và OR LOAD-(**OR LD)**

- Lệnh AND LD nối tiếp 2 khối logic với nhau trong một sơ đồ bậc thang.
- Lệnh OR LD nối song song 2 khối với nhau trong một sơ đồ bậc thang

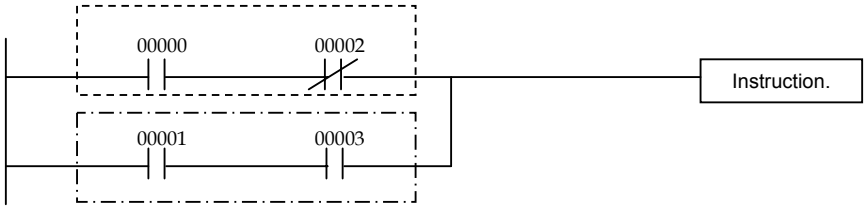
Cần chú ý thứ tự nhập lệnh này: các khối logic cần nối với nhau được nhập riêng rẽ trước, sau đó mới nhập lệnh OR LD hoặc AND LD.
Lệnh này không cần tham số & chỉ cần dùng khi viết chương trinh dạng mnemonic.

Ví dụ: AND LD



Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
00004	AND LD	
. .	. .	. .

Ví dụ OR LD

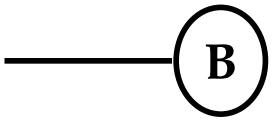


Địa chỉ	Lệnh	Th. số
00000	LD	00000
00001	AND NOT	00001
00002	LD	00002
00003	AND	00003
00004	OR LD	
00005	Lệnh ...	.....

2.2.5) Lệnh cuộn dây: OUT và OUT NOT

Lệnh OUT (OUT NOT) sẽ bật bit được gán cho lệnh này lên ON (xuống OFF) khi điều kiện thực thi đi trước nó là ON và sẽ reset bit này về OFF khi điều kiện đi trước là OFF. Lệnh OUTPUT giống với chức năng ***cuộn dây*** trong sơ đồ điện là khi một ***cuộn dây*** nhận được điện từ tiếp điểm (điều kiện) đi trước nó sẽ hút (đóng) hay nhả (mở) tiếp điểm đi kèm.

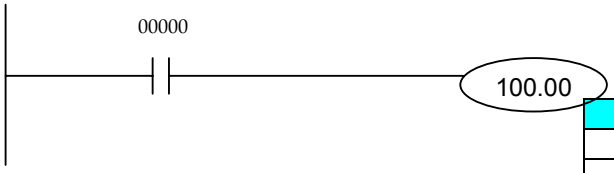
Ký hiệu: OUTPUT-**OUT**



B : BIT
IR, SR, AR, HR, LR, TR

Các địa chỉ có thể truy cập ở dạng bit

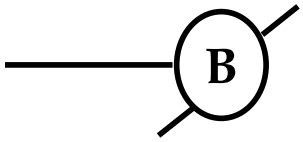
Ví dụ: Lệnh OUT



Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OUT	100.00

Tiếp điểm 00000 là điều kiện thực thi của cuộn dây 100.00.

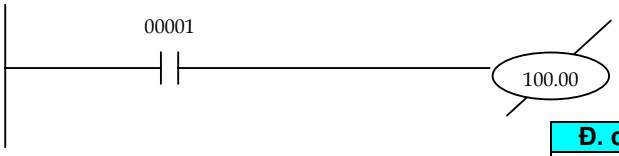
Ký hiệu: OUTPUT NOT-**OUT NOT**



B : BIT
IR, SR, AR, HR, LR, TR

Các địa chỉ có thể truy cập ở dạng bit

Ví dụ: **OUT NOT**



Đ. chỉ	Lệnh	Th. số
00000	LD	00001
00001	OUT NOT	100.00

2.3 Các hàm chức năng đặc biệt - Function (FUN)

Ngoài các lệnh điều kiện và đầu ra đơn giản trên, trong PLC loại CP1L/1H còn có các lệnh với các chức năng phức tạp khác. Mỗi lệnh này đều có một mã lệnh (code) riêng. Khi dùng CX-Programmer, ta sẽ dùng công cụ **Instruction** để thêm 1 hàm chức năng và có thể nhập mã lệnh hoặc tên lệnh đều được.
Dưới đây là mã của một số lệnh trong PLC loại CP1L/1H :

FUN 01	là lệnh	END (End Instruction)
FUN 02	,,	IL (Interlock)
FUN 03	,,	ILC (Interlock Clear)
FUN 04	,,	JMP (Jump End)
FUN 05	,,	JME (Jump End)
FUN 10	,,	SFT (Shift Register)
FUN 11	,,	KEEP (Latching Relay)
FUN 12	,,	CNTR (Reversible Counter)
FUN 13	,,	DIFU (Differentation - Up)
FUN 14	,,	DIFD (Differentation -Down)

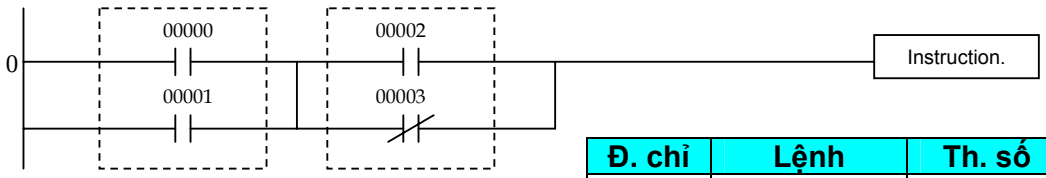
Chú ý :

- Các số 0 ở đầu các mã lệnh (ví dụ **0**1 (END), **0**2 (IL),...) phải được nhập vào. Nếu chỉ nhập chữ số sau thì kết quả có thể không đúng.
- Khi biểu diễn lệnh, người ta thường ghi kèm cả mã lệnh của lệnh đó trong dấu ngoặc đơn theo sau tên lệnh. Ví dụ: END(01), IL(02),.. Tuy nhiên khi nhập lệnh vào chương trình thì chỉ cần nhập tên lệnh hoặc mã lệnh là đủ.

2.3.1) Lệnh END (01)

Lệnh END(01) dùng để đánh dấu điểm kết thúc của chương trình. Một chương trình có thể có nhiều lệnh END (01) nhưng PLC sẽ chỉ xử lý các lệnh từ đầu chương trình đến lệnh END đầu tiên mà nó gặp, sau đó chương trình lại bắt đầu từ lệnh đầu tiên của chương trình. Nếu không có lệnh END trong chương trình, khi PLC chuyển sang chế độ RUN thì trên màn hình của bộ lập trình cầm tay sẽ báo lỗi "NO END INSTR" và chương trình sẽ không được thực hiện.

Ví dụ Chương trình dạng sơ đồ bậc thang (trên) và dạng Mnemonic tương đương (dưới) đều không có lệnh END(01), do đó sẽ bị báo lỗi và không thể chạy được:



Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OR	00001
00002	LD	00002
00003	OR NOT	00003
00004	AND LD	---

Lỗi sẽ báo như sau trên phần mềm CX-Programmer:

NO END INSTRUCTION!

2.3.2) Lệnh IL (02) và ILC (03)

Lệnh IL (Interlock) và ILC (Interlock Clear) luôn được dùng đi kèm với nhau. Khi một lệnh IL được đặt trước một đoạn chương trình, điều kiện thực hiện của IL sẽ điều khiển điều kiện thực hiện của toàn bộ các lệnh bắt đầu từ sau lệnh IL cho đến lệnh ILC đầu tiên sau lệnh IL này. Khi điều kiện thực hiện của lệnh IL là ON, chương trình vẫn được thực hiện bình thường. Khi điều kiện thực hiện của lệnh IL là OFF, tất cả các lệnh theo sau lệnh IL cho đến lệnh ILC đầu tiên đều được thi hành với điều kiện thực hiện là OFF. Nghĩa là các lệnh Output nằm giữa IL và ILC sẽ là OFF.

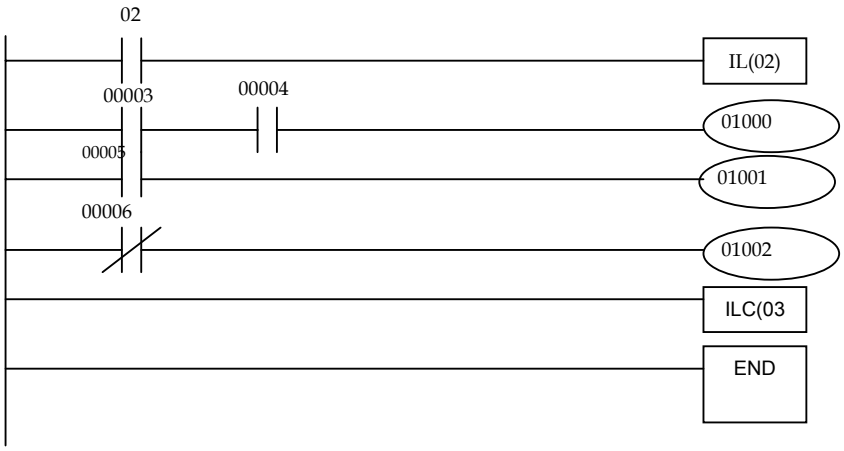
Chương trình sẽ trở lại hoạt động bình thường sau lệnh ILC.

Ví dụ:

Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	IL (02)	-
00002	LD	00003
00003	AND	00004
00004	OUT	100.00
00005	LD	00005
00006	OUT	100.01
00007	LD NOT	00006
00008	OUT	100.02
00009	ILC(03)	-
00010	END(01)	-

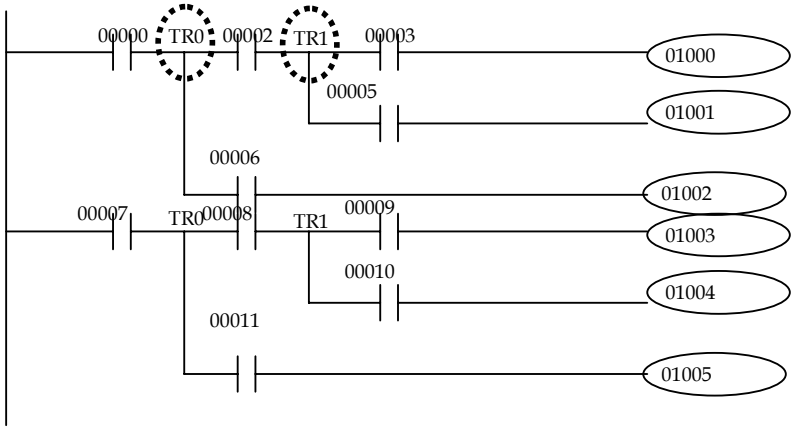
Chú ý :

- Các bit được set hoặc reset bởi lệnh KEEP đặt trong khối INTERLOCK vẫn ở trạng thái cũ của chúng.
- Timer nằm trong khối INTERLOCK sẽ bị reset khi điều kiện thực thi của IL là OFF hoặc khi mất điện.
- PV của counter nằm trong khối INTERLOCK sẽ <u>không</u> bị reset khi điều kiện thực thi của IL là OFF.

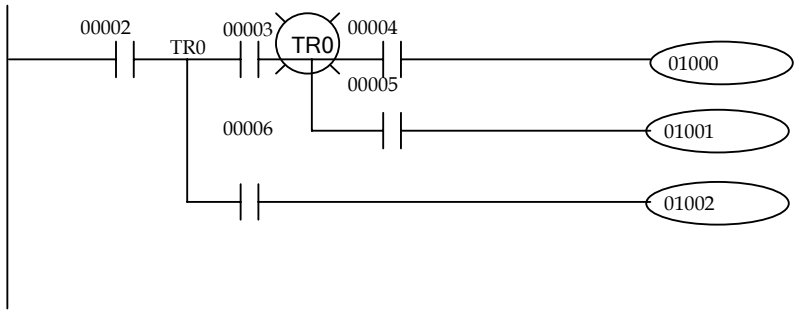


2.3.3) Bit phân nhánh - TR (Temporary Relay)

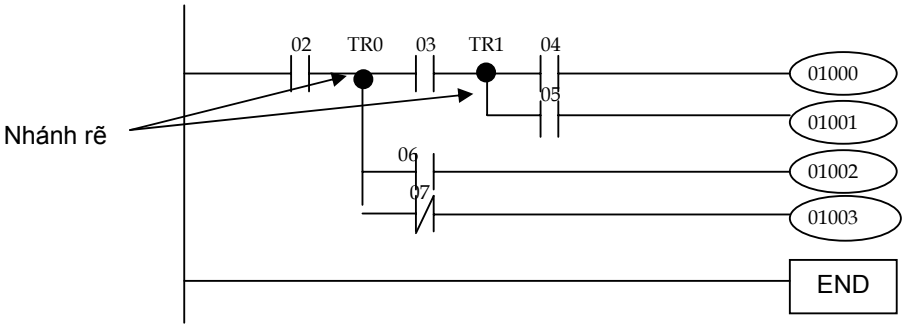
Trong các nhánh chương trình, các bit phân nhánh (7 bit từ TR0-TR7) được dùng để lưu điều kiện thực hiện tại điểm phân nhánh, giúp cho việc thực hiện chương trình tại nhánh chương trình được đúng đắn.



Chương trình sau *sai* do dùng hai lần bit TR0 trong cùng một thang chương trình:



Ví dụ : Dùng bit TR để lưu các điều kiện thực hiện khi phân nhánh



Dưới đây là chương trình trên dạng Mnemonic. Các bit TR được nhập vào bằng lệnh OUT, với tham số là số của bit TR, sau đó được dùng lại bằng lệnh LD để bắt đầu một nhánh khác của chương trình:

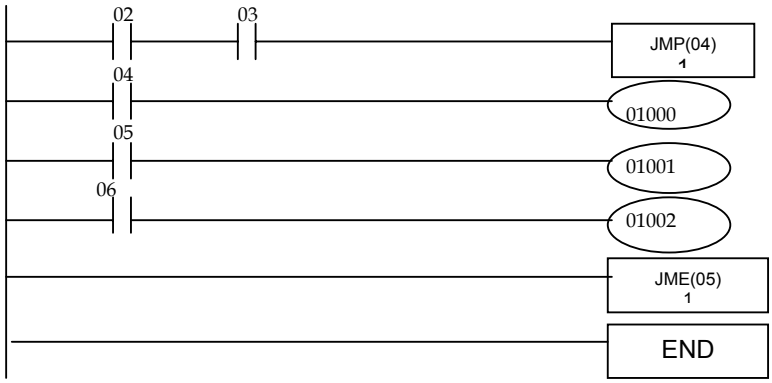
Địa chỉ	Lệnh	Th. số
0000	LD	00002
0001	OUT	TR 0
0002	AND	00003
0003	OUT	TR1
0004	AND	00004
0005	OUT	100.00
0006	LD	TR1
0007	AND	00005
0008	OUT	100.01
0009	LD	TR0
0010	AND	00006
0011	OUT	100.02
0012	LD	TR0
0013	AND-NOT	00007
0014	OUT	100.03
0015	END (01)	-

Chú ý : Các bit TR chỉ được dùng khi lập trình dạng mnemonic code. Còn khi lập trình với ladder diagram (dùng phần mềm CX-PROGRAMMER), bit này không cần thiết vì chương trình đã tự động thực hiện việc này.

2.3.4) Lệnh JMP (04) và JME (05)

Mỗi lệnh JUMP gồm cặp lệnh JMP và JME có số từ 00 đến 99; JMP và JME luôn đi theo cặp với nhau. Khi chương trình gặp lệnh JMP n (n= số của lệnh JUMP), nó sẽ bỏ qua không thực hiện các lệnh theo sau lệnh này cho đến lệnh JME n có cùng số. Khi gặp lệnh JME, chương trình sau đó lại thực hiện bình thường. Mặc dù hoạt động của JMP khá giống với hoạt động của INTERLOCK khi điều kiện thực hiện của IL là OFF, nhưng đối với lệnh JMP, các toán tử nằm giữa lệnh JMP và JME không bị OFF mà vẫn giữ nguyên trạng thái trước khi thực hiện lệnh JUMP này.

Ví du : Lệnh JUMP



Chương trình dạng mnemonic :

Đ. chỉ	Lệnh	Th. số
0000	LD	00002
0001	AND	00003
0002	JMP(04)	1
0003	LD	00004
0004	OUT	100.00
0005	LD	00005
0006	OUT	100.01
0007	LD	00006
0008	OUT	100.02
0009	JME(05)	1
0010	END(01)	

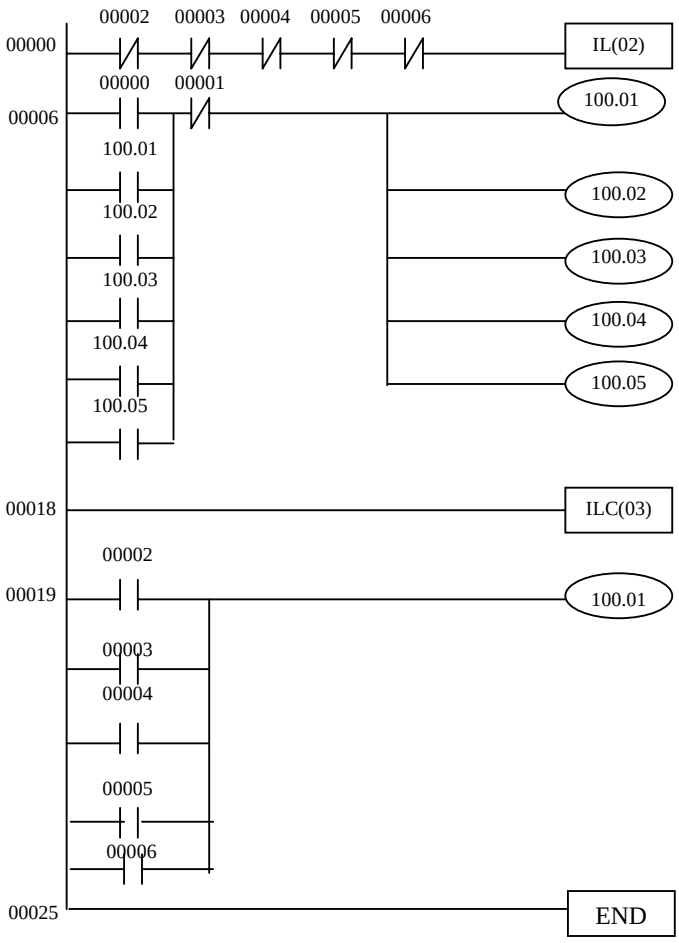
2.4 Ví dụ ứng dụng: Dừng động cơ khi có quá tải

Có 5 motor nối liên động với nhau. Khi nút PB Start được nhấn, cả 5 Motor đều khởi động và chạy nếu như không có motor nào đang bị quá tải (overload). Nếu 1 trong 5 motor này bị quá tải hoặc khi nút Stop được nhấn, cả 5 motor sẽ dừng. Đèn báo Overload sẽ sáng nếu có motor nào đó đang bị quá tải.

I/O

Đầu vào		Đầu ra	
00000	PB Start	100.00	Lamp Overload
00001	PB Stop	100.01	Motor 1
00002	Overload M1	100.02	Motor 2
00003	Overload M2	100.03	Motor 3
00004	Overload M3	100.04	Motor 4
00005	Overload M4	100.05	Motor 5
00006	Overload M5		

Chương trình dạng sơ đồ bậc thang



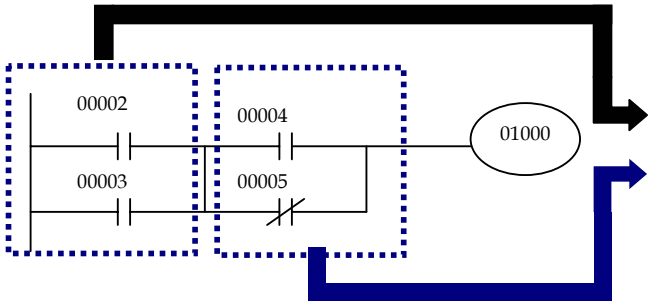
Chương trình dạng mnemonic

00000	LD	NOT	00002
00001	AND	NOT	00003
00002	AND	NOT	00004
00003	AND	NOT	00005
00004	AND	NOT	00006
00005	IL	(02)	
00006	LD		00000
00007	OR		100.01
00008	OR		100.02
00009	OR		100.03
00010	OR		100.04
00011	OR		100.05
00012	AND	NOT	00001
00013	OUT		100.01
00014	OUT		100.02
00015	OUT		100.03
00016	OUT		100.04
00017	OUT		100.05
00018	ILC	(03)	
00019	LD		00002
00020	OR		00003
00021	OR		00004
00022	OR		00005
00023	OR		00006
00024	OUT		100.01
00025	END	(01)	

- *Các lệnh **AND LD** và **OR LD** có thể được dùng để lập các sơ đồ với các phần tử kết nối phức tạp khi viết chương trình bằng mnemonic:*

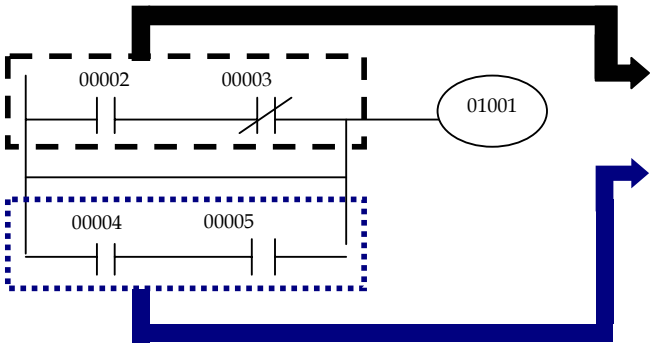
Ví dụ : Cách nhập các lệnh AND LD và OR LD:

■ **AND LD** Dùng để nối nối tiếp 2 khối logic chương trình



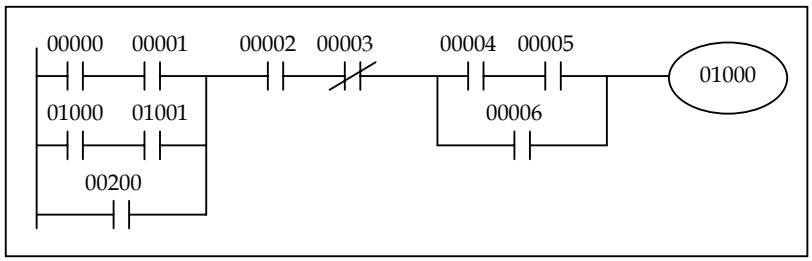
Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	OR	00003
00002	LD	00004
00003	OR NOT	00005
00004	AND LD	- - -
00005	OUT	100.00

■ **OR LD** Dùng để nối song song 2 khối logic chương trình



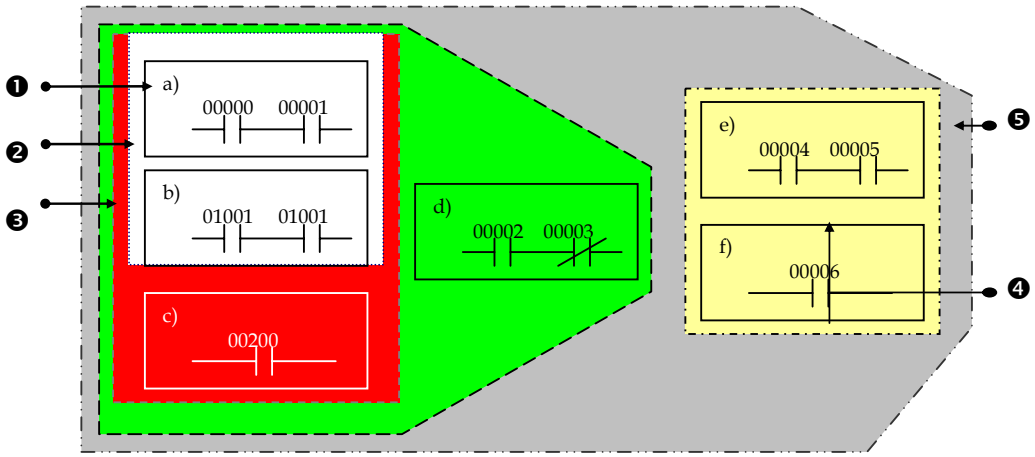
Đ. chỉ	Lệnh	Th. số
00000	LD	00002
00001	AND NOT	00003
00002	LD	00004
00003	AND	00005
0004	OR LD	- - -
00005	OUT	100.01

Ví dụ : ta có 1 đoạn chương trình với các khối logic chương trình nối kết khá phức tạp như hình dưới :

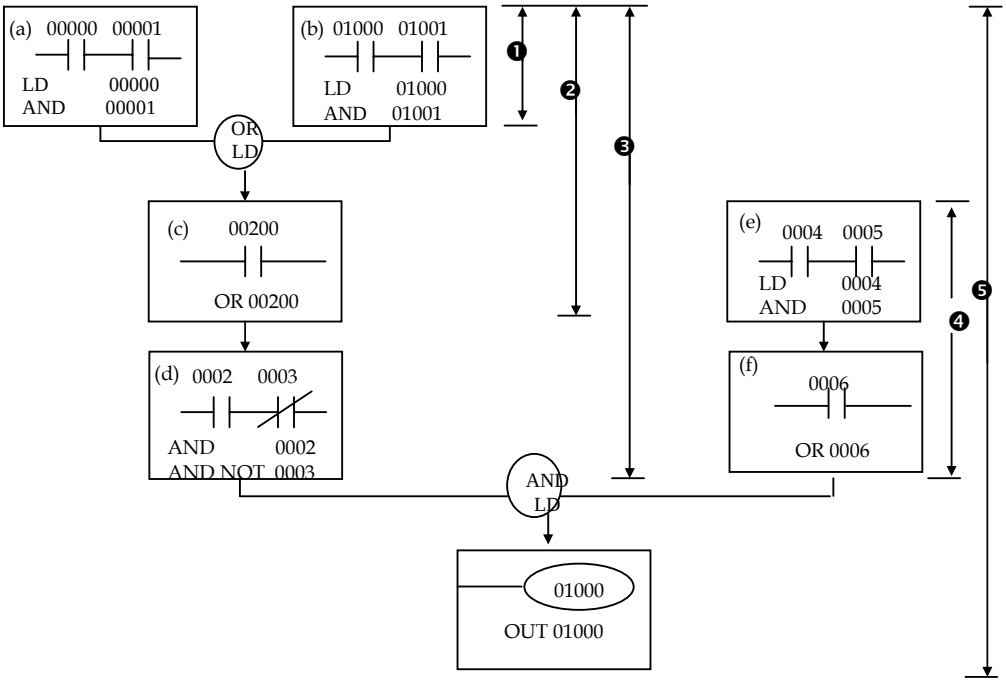


Để có thể nhập được chương trình này cũng như các khối chương trình phức tạp khác vào bằng bộ lập trình cầm tay, cần thực hiện các bước sau :

(1) Chia nhỏ đoạn chương trình thành các khối **block cơ bản [1] - [5]**



2) Nhập từng khối này vào bộ lập trình, bắt đầu từ trên xuống dưới, từ trái qua phải như bình thường theo thứ tự các khối ❶ → ❺ trên ví dụ dưới đây:

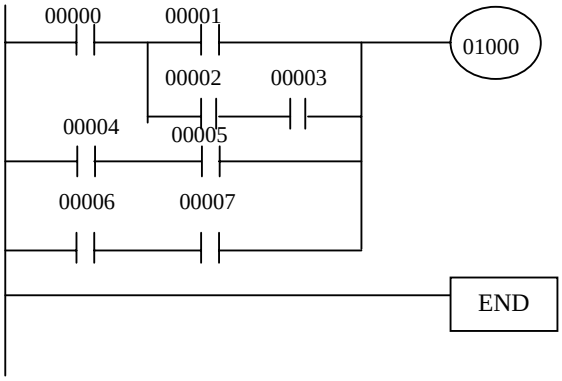


Chú ý : Các khối logic cơ bản là các khối với các phần tử có thể được nối với nhau bằng các lệnh LD, LD NOT, AND, AND NOT, OR, OR NOT, ..

Bài ôn tập :

Cho một chương trình dưới dạng Ladder Diagram dưới đây. Hãy nhập vào chuyển đổi chương trình sang dạng Mnemonic Code :

Ladder Diagram

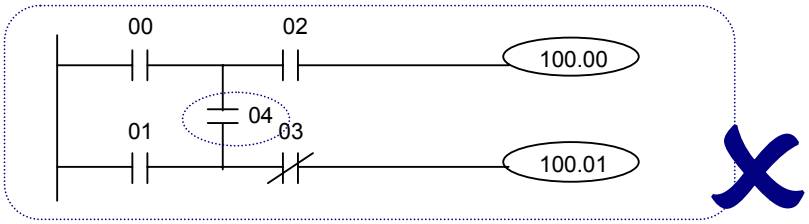


Mnemonic Codes

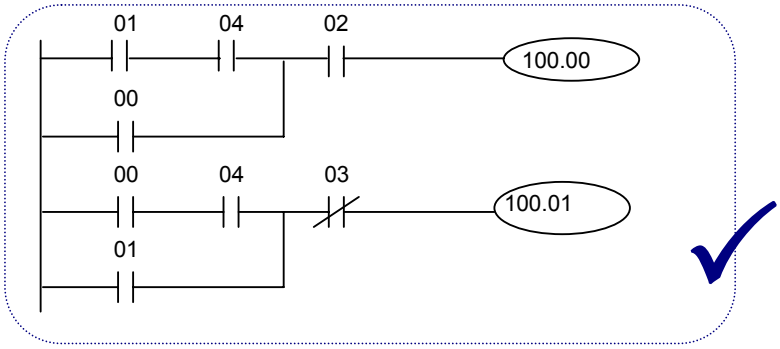
Đ. chỉ	*Lệnh*	*Th.số*

2.5 Các lưu ý khi lập một chương trình dạng Ladder Diagram

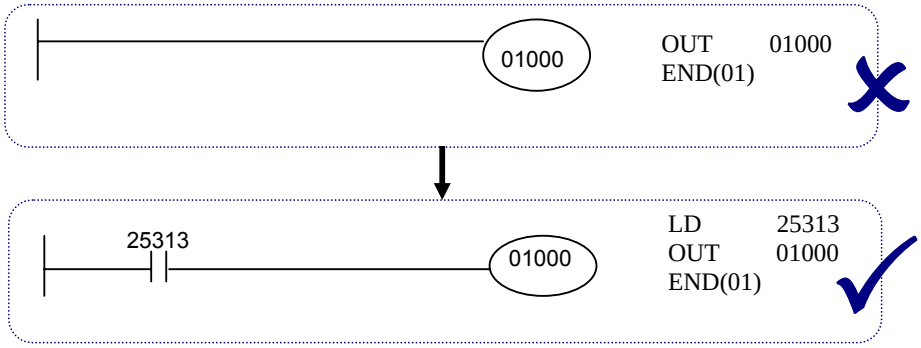
1. Hai thang khác nhau không được phép nối bằng một tiếp điểm thẳng đứng :



Đoạn chương trình trên không đúng vì hai thang được nối với nhau bằng một tiếp điểm thẳng đứng và sẽ được sửa như đoạn chương trình dưới đây :

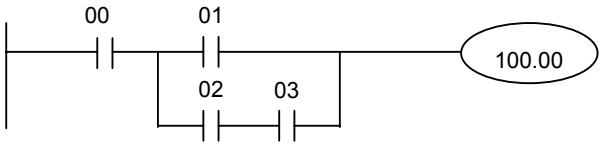


2. Nếu một lệnh OUTPUT hoặc một FUN luôn luôn cần điều kiện thực hiện là ON, lệnh này không được nối trực tíếp với thanh power bus bên trái. Thay vào đó, phải nối qua một tiếp điểm dùng cờ "ALWAYS ON" (có địa chỉ là 25313)



Trường hợp ngoại lệ : Các lệnh INTERLOCK CLEAR, JUMP END, STEP, END không tuân theo quy tắc này.

3. Chú ý đến các số lượng lệnh cần thiết để nhập một chương trình.



Hình A :

Ở sơ đồ hình A trên, ta cần có thêm lệnh OR LD và AND LD để nối nhánh dưới với nhánh trên.

Các lệnh dạng mnemonic cho sơ đồ hình A

Đ. chỉ	*Lệnh*	*Th. số*
0000	LD	00
0001	LD	01
0002	LD	02
0003	AND	03
0004	OR LD	
0005	AND LD	
0006	OUT	100.00

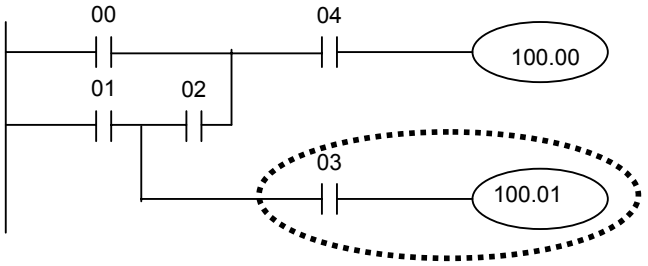
Đoạn chương trình trên có thể được sửa lại như hình B sau đây :

Các lệnh dạng mnemonic cho sơ đồ hình B

Đ. chỉ	*Lệnh*	*Th. số*
0000	LD	02
0001	AND	03
0002	OR	01
0003	AND	00
0004	OUT	100.00

Rõ ràng là với cách biểu diễn tương đương như hình B, việc biểu diễn đơn giản hơn và giảm đi được 2 lệnh AND LD và OR LD.

4. Một nhánh không được xuất phát từ một nhánh song song khác.

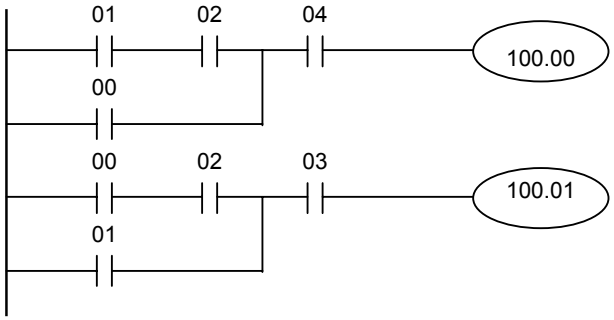


Khi các bit 01, 02, 04 là ON sẽ bật 1000 lên ON

Hình A

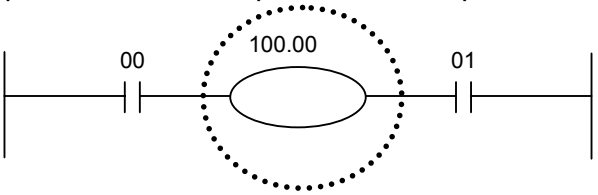
Ở hình A, ta muốn khi các bit 00, 02 và 03 là ON hoặc khi 01 và 03 là ON, bit 100.01 sẽ được bật lên ON. Tuy nhiên đó là cách biểu diễn không thích hợp với việc nhập bằng Console.

Đoạn chương trình trên được sửa lại như hình B sau :

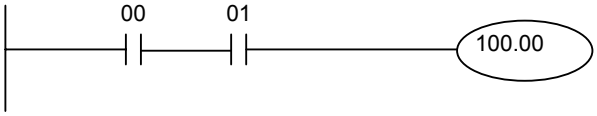


Hình B

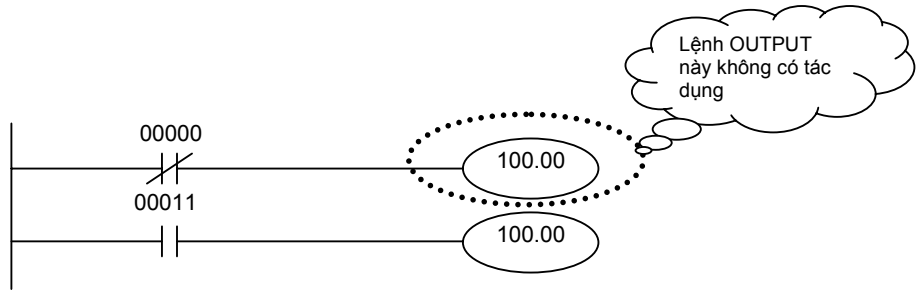
5. Lệnh OUT hoặc OUT NOT (nếu có) phải là lệnh cuối cùng trên thang và phải được nối vào power bus bên phải.



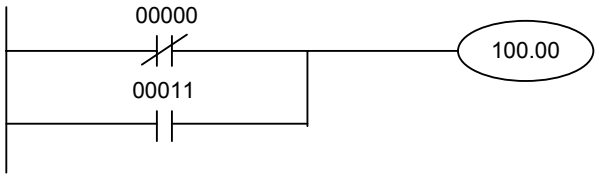
Đoạn chương trình trên không đúng vì lệnh OUT 100.00 không được nối trực tiếp vào power bus mà qua một tiếp điểm và sẽ được sửa lại như sau :



6. Nếu một địa chỉ bit được dùng lặp lại trên hai lệnh OUTPUT khác nhau, lệnh OUTPUT đi trước sẽ không có tác dụng.



Do đó, nếu 2 bit 00000 và 00011 đều dùng để điều khiển lệnh OUTPUT với bit 100.00 thì đoạn chương trình trên sẽ được sửa lại như sau :



Bài ôn tập về Lập trình cơ bản trên CP1L/1H dùng Mnemonic Code và Ladder Diagram

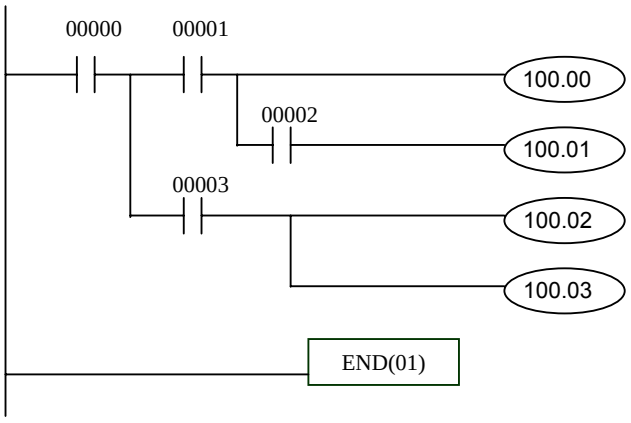
1. Hãy viết chương trình dạng Mnemonic Code cho chương trình dạng sơ đồ bậc thang dưới đây

Đ. chỉ	*Lệnh*	*Th. số*
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		

2. Cho một chương trình dạng Mnemonic Code bên dưới, hãy viết chương trình tương đương dưới dạng Ladder Diagram :

Đ. chỉ	*Lệnh*	*Th. số*
00000	LD	00000
00001	AND	00001
00002	LD NOT	00002
00003	AND	00003
00004	OR LD	
00005	LD	00004
00006	AND NOT	00005
00007	LD NOT	00006
00008	AND	00007
00009	OR LD	
00010	AND LD	
00011	OUT	100.00
00012	END (01)	

3. Hãy nhập chương trình dưới dạng Mnemonic Code cho đoạn chương trình dạng sơ đồ bậc thang dưới đây :



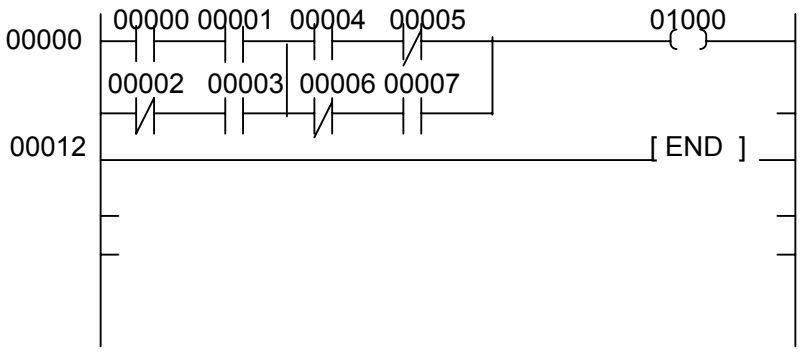
Đ. chỉ	*Lệnh*	*Th. số*
00000		
00001		
00002		
00003		
00004		
00005		
00006		
00007		
00008		
00009		
00010		

Đáp án :

1)

```
00000  LD          00000
00001  AND         00001
00002  AND NOT     00002
00003  LD          00003
00004  AND NOT     00004
00005  OR   LD
00006  AND         00005
00007  AND         00006
00008  OUT         100.00
00009  END(01)
```

2)



3)

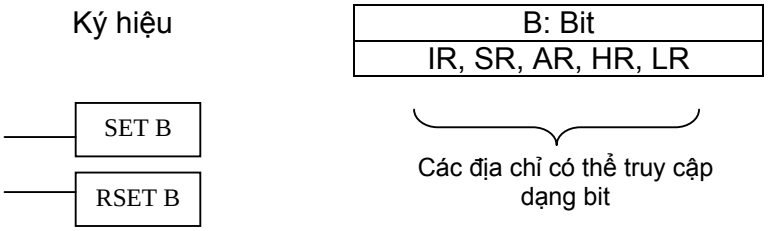
```
00000  LD              00000
00001  OUT       TR      0
00002  AND             00001
00003  OUT             100.00
00004  AND             00002
00005  OUT             100.01
00006  LD        TR      0
00007  AND             00003
00008  OUT             100.02
00009  OUT             100.03
00010  END(01)
```

Chú ý : Nhánh rẽ với lệnh AND 00002 và OUT 00001 không cần thêm bit TR vì giữa điểm rẽ nhánh và lệnh OUT 100.00 không có tiếp điểm nào.

2.6 Các lệnh đặc biệt thông dụng khác:

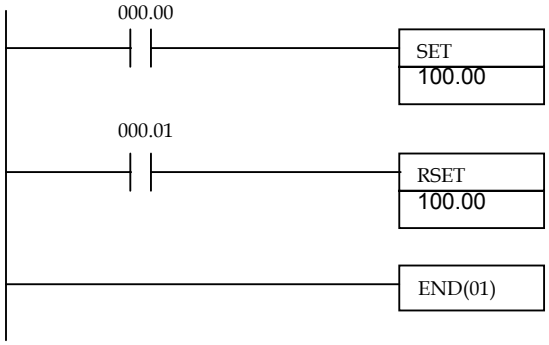
2.6.1) Bật bit (SET) và Xoá bit (RESET) SET- RSET

Lệnh SET sẽ bật bit đi kèm lên ON khi điều kiện thực thi của nó là ON. Sau đó, bit sẽ vẫn ở trạng thái ON không phụ thuộc vào việc lệnh SET có điều kiện thực hiện là ON hay OFF cho đến khi lệnh RESET (RSET) xoá nó về OFF.



Ví dụ: Bit 100.00 sẽ được bật lên ON khi điều kiện thực hiện của lệnh SET (là bit 00000) là ON. ở các chu kỳ quét sau, bit 100.00 sẽ vẫn giữ (Hold) ở trạng thái ON cho dù bit 00000 là ON hay OFF. Bit 100.00 sẽ chỉ bị xoá bởi lệnh Reset khi bit 00001 là ON.

Sơ đồ bậc thang



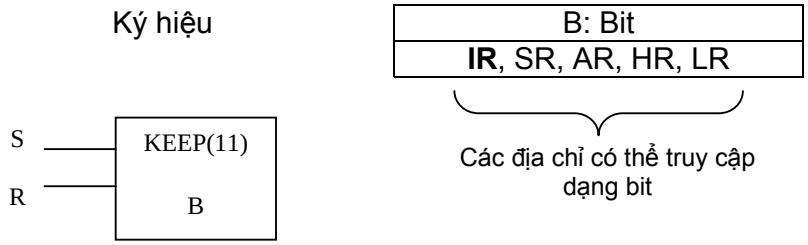
Các lệnh dạng Mnemonic

```
00000   LD      000.00
00001   SET     100.00
00002   LD      000.01
00003   RSET    100.00
00004   END
```

Chú ý : Trạng thái của bit được SET hay RSET sẽ không thay đổi khi nằm trong khối INTERLOCK hay JUMP.

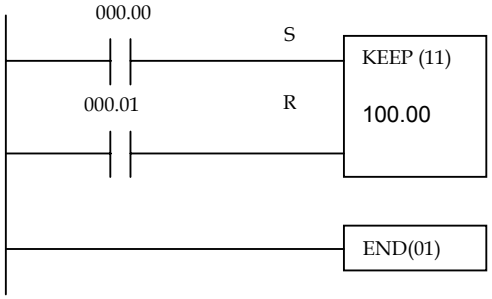
2.6.2) Lệnh giữ KEEP - KEEP(11)

Lệnh KEEP hoạt động như một rơ le chốt với hai đầu vào là SET (S) và RESET (R). Bit B sẽ được Set lên ON khi đầu vào S là ON và sẽ vẫn giữ ở ON cho đến khi B bị reset về OFF khi đầu vào R là ON.



Chú ý : Các bit được set hay reset bởi KEEP không bị reset khi nằm trong khối INTERLOCK.

Ví dụ : Bit 100.00 sẽ được set lên ON khi bit 00000 lên ON và sẽ vẫn ở ON cho dù sau đó bit 00000 là ON hay OFF. Bit 100.00 chỉ bị reset về OFF khi bit 00001 là ON (đầu vào RESET sẽ tác động)

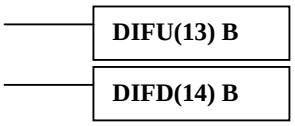


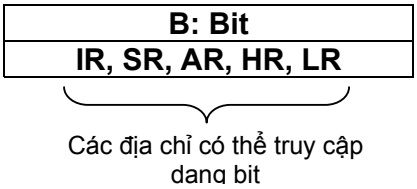

```
00000   LD      000.00
00001   LD      000.01
00002   KEEP    100.00
00003   END
```

2.6.3) DIFFERENTIATE UP và DOWN - DIFU(13) & DIFD(14)

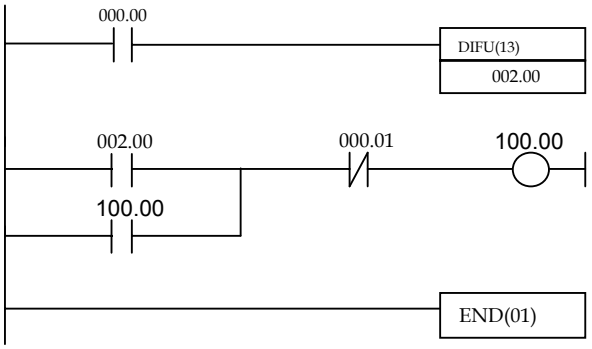
- **DIFU(13)** : Lệnh này sẽ bật bit đi kèm lên 1 trong vòng một chu kỳ quét (scan/cycle) khi điều kiện thực hiện chuyển từ OFF ở chu kỳ quét trước sang ON ở chu kỳ quét lần này. Sau đó bit lại trở về trạng thái OFF.
- **DIFD(14)** : Lệnh này sẽ bật bit đi kèm lên 1 trong vòng một chu kỳ quét (scan/cycle) khi điều kiện thực hiện chuyển từ ON ở chu kỳ quét trước sang OFF ở chu kỳ quét lần này. Sau đó bit lại trở về trạng thái OFF.

Ký hiệu





Ví dụ: Khi bit 000.00 chuyển từ OFF ở chu kỳ quét trước lên ON ở chu kỳ quét hiện hành, bit 002.00 sẽ được bật lên ON trong vòng một chu kỳ. ở chu kỳ quét sau, bit 002.00 lại được quay trở về OFF.




```
00000   LD          000.00
00001   DIFU        002.00
00002   LD          002.00
00003   OR          100.00
00004   AND NOT     000.01
00005   OUT         100.00
00006   END
```

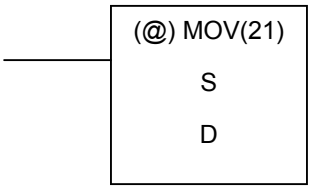
2.6.4) Lệnh copy dữ liệu MOVE - MOV(21)

Ký hiệu

S: Source word
IR, SR, AR, DM, HR, TC, LR, #

Các địa chỉ bộ nhớ truy cập ở dạng word hoặc hằng số (#)

D: Destination word
IR, SR, AR, DM, HR, LR



S = Là địa chỉ của word nguồn (Source word) hoặc một hằng số (# là ký hiệu của một hằng số, ví dụ #155,...được nhập vào ngay khi lập trình)

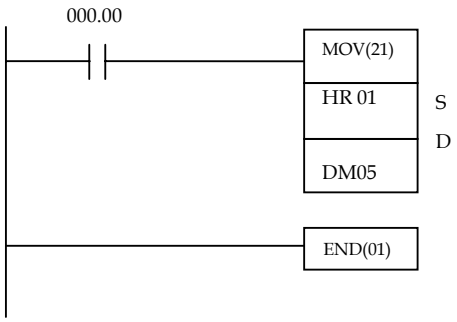
D = Là địa chỉ của word đích (Destination word)

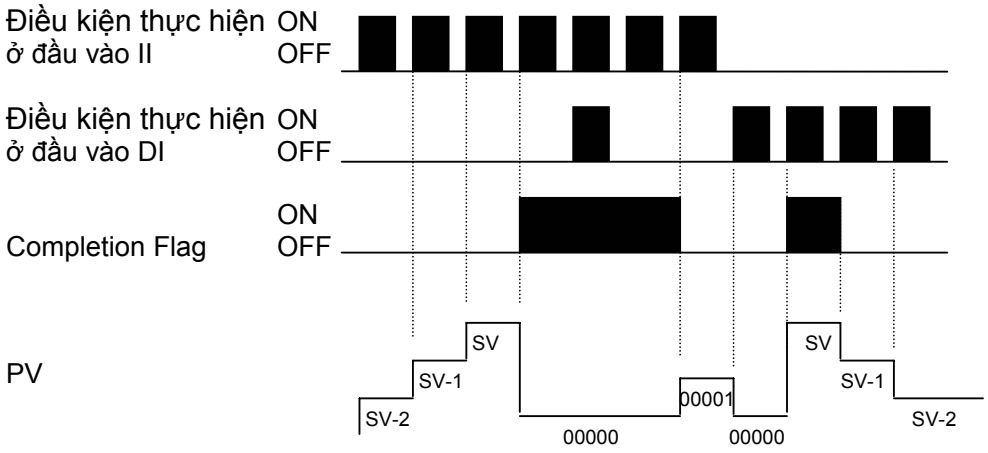
Khi lệnh MOV(21) có điều kiện thực hiện là ON, lệnh này sẽ copy hằng số hoặc nội dung của word có địa chỉ chỉ định bởi S sang word có địa chỉ chỉ định bởi D. Nội dung của word nguồn S không thay đổi khi thực hiện lệnh này. Ví dụ: Khi Source word là 000, còn Destination word là 100

Source word CH 000				**Destination word CH 100**		
00000		1	→	100.00		1
00001		1	→	100.01		1
00002		0	→	100.02		0
00003		1	→	100.03		1
00004		1		100.04		1
00005		0	.	100.05		0
00006		0	.	100.06		0
00007		1	.	100.07		1
00008		1		100.08		1
00009		1		100.09		1
00010		1		100.10		1
00011		0		100.11		0
00012		0		100.12		0
00013		0	→	100.13		0
00014		0	→	100.14		0
00015		1	→	100.15		1

Ví dụ:

Ở ví dụ dưới đây, địa chỉ word nguồn là S = HR01 (và nội dung của word này là giá trị 1500) còn địa chỉ của word đích là D = LR 05. Khi bit 000.00 lên ON, lệnh MOV(21) sẽ copy nội dung của HR01 (tức giá trị 1500) sang word LR05.





2.6.5) Bộ đếm lên xuống - Reversible Counter CNTR (FUN 12) (hay còn gọi là UP/DOWN Counter)

Chú ý : Mỗi bộ counter và timer có một số duy nhất từ 0 đến 127 và không được phép dùng trùng lặp trong lệnh đếm/timer khác của chương trình.

Số của bộ đếm và timer có 2 cách dùng như sau :

- Khi dùng như một bit, nó được dùng làm cờ báo đã đếm xong (completion flag).
- Khi dùng như một word, nó được dùng để truy cập giá trị đếm hiện tại (PV).

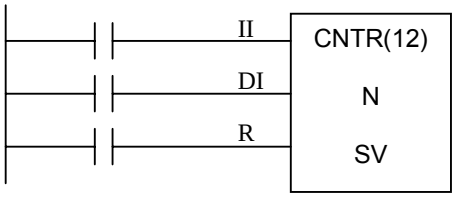
CNTR là một bộ đếm có thể đếm theo hai chiều tăng - giảm:

- Bộ đếm sẽ tăng giá trị của PV (Present Value) lên 1 mỗi khi đầu vào II (Increment Input) chuyển từ OFF lên ON.
- Bộ đếm sẽ giảm giá trị của PV (Present Value) đi 1 mỗi khi đầu vào DI (Decrement Input) chuyển từ OFF lên ON. Khi bộ đếm giảm đến 0, giá trị hiện tại của PV được gán cho SV và cờ báo hoàn thành (completion flag - chính là bit CNTR n với n = số của counter) sẽ lên ON cho đến khi bộ đếm lại giảm tiếp.
- Bộ đếm sẽ reset PV về 0 khi đầu vào Reset Input (R) chuyển từ OFF lên ON.

- Khi PV bằng với giá trị đặt SV (Set Value), PV được reset về 0 và cờ báo hoàn thành sẽ bật lên ON cho đến khi bộ đếm lại tiếp tục đếm tăng.
- Khi cả II và DI đều cùng chuyển từ OFF lên ON, bộ đếm vẫn giữ nguyên giá trị.

Ký hiệu

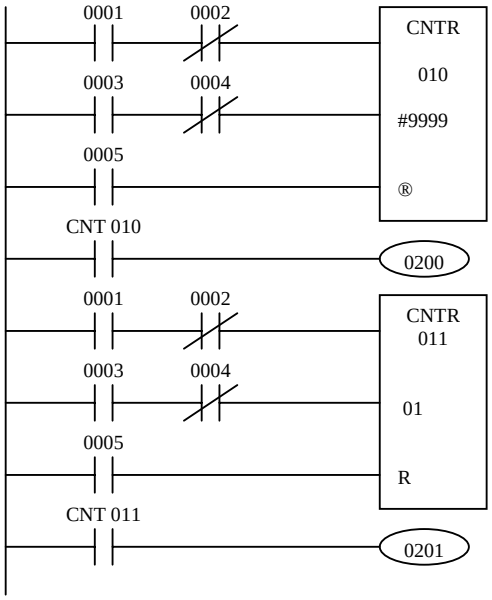
N : số của counter # (0-127)
SV: Set Value (word, BCD)
IR, SR, AR, DM, HR, LR, #



- II : Đầu váo đếm tăng
- DI : Đầu váo đếm giảm
- R : đầu vào reset giá trị PV
- SV : Giá trị đặt trước

Ví dụ minh hoạ Bộ đếm tăng giảm (UP/DOWN counter)

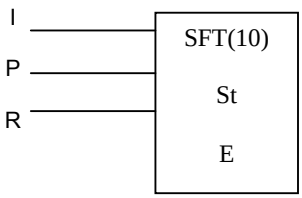
Ladder Diagram



Mnemonic Code

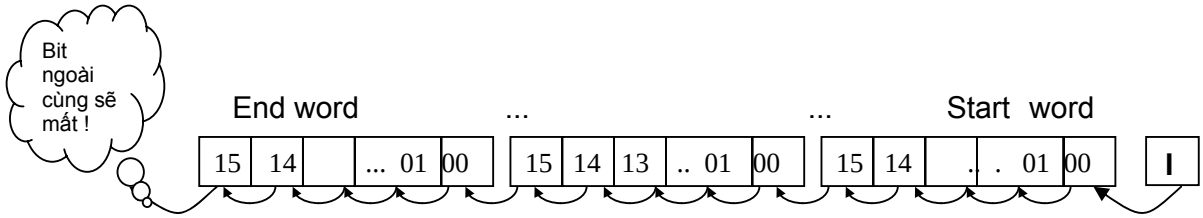
Đ. chỉ	Lệnh	Th. số	
0200	LD		0001
0201	AND-NOT		0002
0202	LD		0003
0203	AND-NOT		0004
0204	LD		0005
0205	CNTR(12)		010
		#	9999
0206	LD	CNT	010
0207	OUT		0200
0208	LD		0001
0209	AND-NOT		0002
0210	LD		0003
0211	AND-NOT		0004
0212	LD		0005
0213	CNTR(12)		011
			01
0214	LD	CNT	011
0215	OUT		0201

2.6.6) Thanh ghi dịch - SHIFT REGISTER - SFT(10)



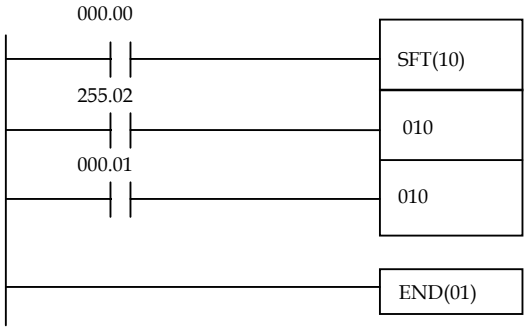
St = Word đầu tiên của thanh ghi dịch
E = Word cuối của thanh ghi dịch
I = Bit dịch (Input bit)
P = Bit xung nhịp ((Shifting) Pulse Input)
R = Đầu vào xoá (Reset Input)

Thanh ghi dịch được định nghĩa là các word bắt đầu từ Word đầu tiên St cho đến Word cuối E (địa chỉ Word cuối phải > Word đầu).



Ví dụ minh hoạ:

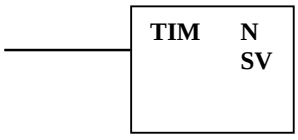
Ở ví dụ dưới đây, ta có 1 thanh ghi dịch dài 1 word (St= 010, E =010) tại địa chỉ 010. Lệnh SFT(10) sẽ dịch các bit của của thanh ghi dịch sang bên trái một vị trí bit và bit 000.00 được dịch vào bit ngoài cùng bên phải (tức bit 010.00) của thanh ghi này mỗi khi bit 255.02 chuyển từ OFF lên ON. Bit 255.02 này là một bit xung nhịp 1 giây do đó thanh ghi dịch sẽ được dịch sang trái, bit ngoài cùng bên trái (tức bit 010.15) sẽ mất mỗi giây một lần. Khi bit 000.01 (đầu vào Reset) lên ON, nội dung của thanh ghi dịch sẽ được reset về 0 (các bit đều bị reset về 0).



Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	LD	25502
00002	LD	00001
00003	SFT(10)	
		010
		010
00004	END(01)	

2.6.7) Rơle thời gian (TIMER) - TIM

Ký hiệu



N: Số của Timer
Hằng số (#)

SV: Set Value (Word, BCD)
IR, SR, AR, DM, HR, LR, #

Các địa chỉ bộ nhớ truy cập ở dạng word hoặc hằng số (#)

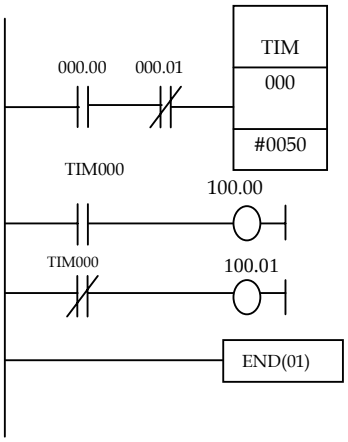
N = Số của timer hiện dùng (Timer Number) (số hợp lệ là từ 000 - 127)
SV = Giá trị đặt trước Set Value tính theo đơn vị là 0,1s (SV phải ở dạng số BCD hoặc chỉ đến một Word có chứa giá trị BCD). Giá trị của SV phải nằm trong khoảng từ 0000 - 9999 (0 - 999,9 Giây.)

Khi đầu vào điều kiện thực thi của hàm TIM là ON, hàm TIM sẽ đếm giảm thời gian từ giá trị thời gian đặt trước SV đến khi bằng 0 thì completion flag (TIM

n) lên ON. Completion flag sẽ vẫn ở ON cho đến khi bị reset bởi đầu vào điều kiện thực hiện về OFF.

Ví dụ: Timer số 000 (TIM000) có đầu vào điều kiện thực hiện do hai bit 000.00 và 000.01 quyết định. Khi bit 000.00 là ON và bit 000.01 là OFF, timer bắt đầu đếm giảm thời gian PV theo từng đơn vị là 0,1 giây từ giá trị đặt trước SV là 5,0 giây. Khi giá trị thời gian hiện tại PV về đến 0, cờ completion flag TIM000 sẽ lên ON và bật bit 010.00 lên ON còn bit 010.01 về OFF.

Ladder Diagram

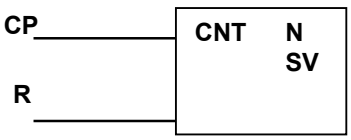


Mnemonic Code

```
00000   LD          000. 00
00001   AND NOT     000. 01
00002   TIM         000     #0050
00003   LD          TIM000
00004   OUT         100.00
00005   LD NOT      TIM000
00006   OUT         100.00
00007   END
```

2.6.8) Bộ đếm giảm (COUNTER) - CNT

Lúc khởi đầu giá trị PV được đặt bằng SV (Set Value). Mỗi khi đầu vào xung đếm CP chuyển từ OFF lên ON, giá trị đếm hiện tại PV (Present Value) sẽ giảm một đơn vị. Khi PV giảm đến 0, cờ báo kết thúc sẽ lên ON và sẽ ở ON cho đến khi counter được reset bởi đầu vào R (Reset).



N: Số của bộ đếm
Hằng số (#)

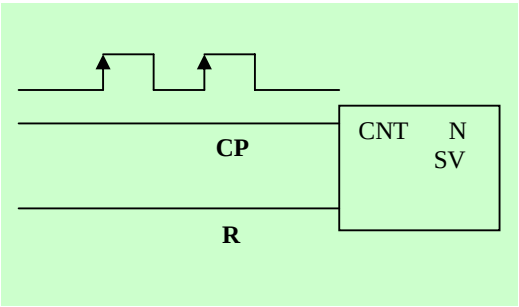
SV: Set Value (Word, BCD)
IR, SR, AR, DM, HR, LR, #

Các địa chỉ bộ nhớ truy cập ở dạng word hoặc hằng số (#)

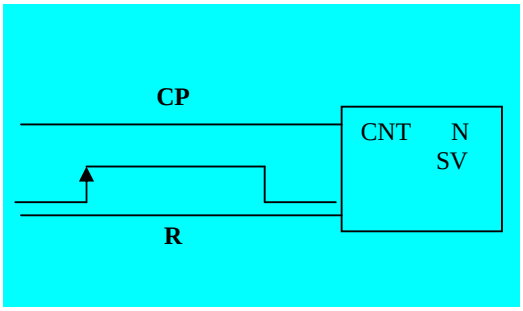
- N : Số của bộ đếm (từ 000 đến 127)
- SV : Giá trị đặt (từ 0 đến 9999) và phải ở dạng BCD
- CP : Đầu vào xung đếm
- R : Đầu vào reset

Ví dụ:

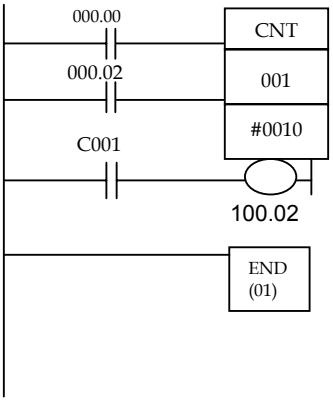
Bộ đếm đang đếm



Bộ đếm bị reset



Ví dụ: Giá trị hiện hành (PV) của bộ đếm CNT001 sẽ giảm từ giá trị SV khi đầu vào Input 00000 chuyển từ OFF lên ON. Khi số lần chuyển từ OFF lên ON của input 00000 là 10 lần (bằng với SV=10), cờ CNT001 sẽ lên ON và do đó bật đầu ra 100.02 lên ON. Cờ CNT001 và PV của bộ đếm sẽ bị reset khi đầu vào input 00002 lên ON.




```
00000   LD    000.00
00001   LD    000.02
00002   CNT   001
              #0010
00003   LD    CNT001
00004   OUT   100.02
00005   END
```

2.6.9) Ví dụ về ứng dụng COUNTER và TIMER

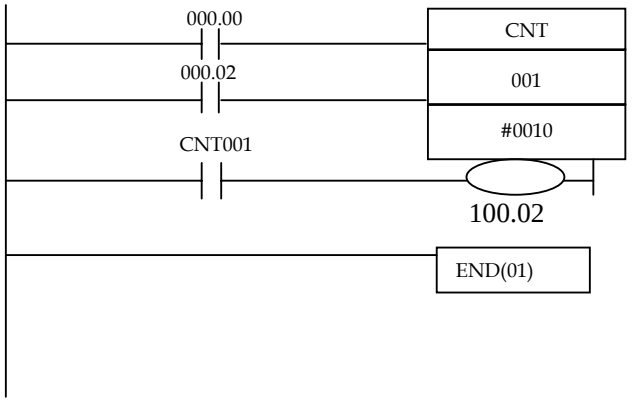
Ví dụ 1 Mở rộng khả năng đếm của counter

Một Photo Switch được dùng để phát hiện sản phẩm và đưa vào đầu vào của counter. Yêu cầu cần phải đếm được 20.000 sản phẩm thì cho ra đèn OUTPUT LAMP (tuy nhiên bộ đếm CNT chuẩn chỉ cho phép đếm tới 9.999).

I/O	
PHOTO SWITCH	000.00
OUTPUT LAMP	010.00
PB RESET	000.01

Ladder Mnemonic Code

Đ. chỉ	Lệnh	Th.số
00000	LD	00000
00001	LD	CNT 001
00002	CNT	001
		#0100
00003	LD	CNT 001
00004	LD	00001
00005	CNT	002
		#0200
00006	LD	CNT 002
00007	OUT	100.02
00010	END(01)	




```
00000   LD      000.00
00001   LD      000.02
00002   CNT     001
                #0010
00003   LD      CNT001
00004   OUT     100.02
00005   END
```

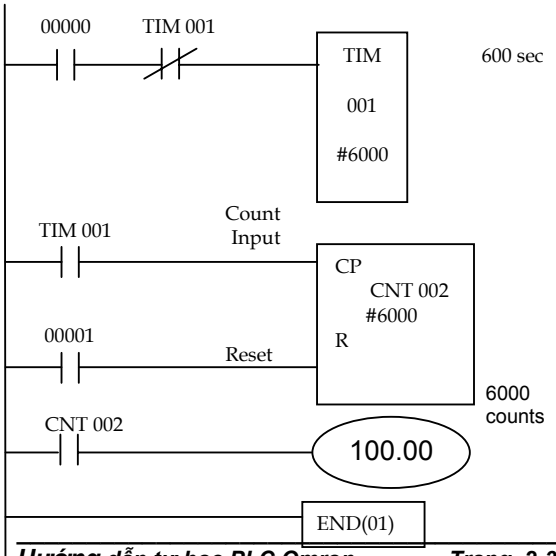
Ví dụ 2 Kéo dài thời gian trễ của timer lên 1.000 giờ

TIM chuẩn chỉ cho phép đặt thời gian tới 999,9 giây. Chương trình sau đây cho phép kéo dài khả năng của TIM lên 1.000 giờ.

I/O

PB START	000.00
PB RESET	000.01
VALUE LUBRICATE	100.00

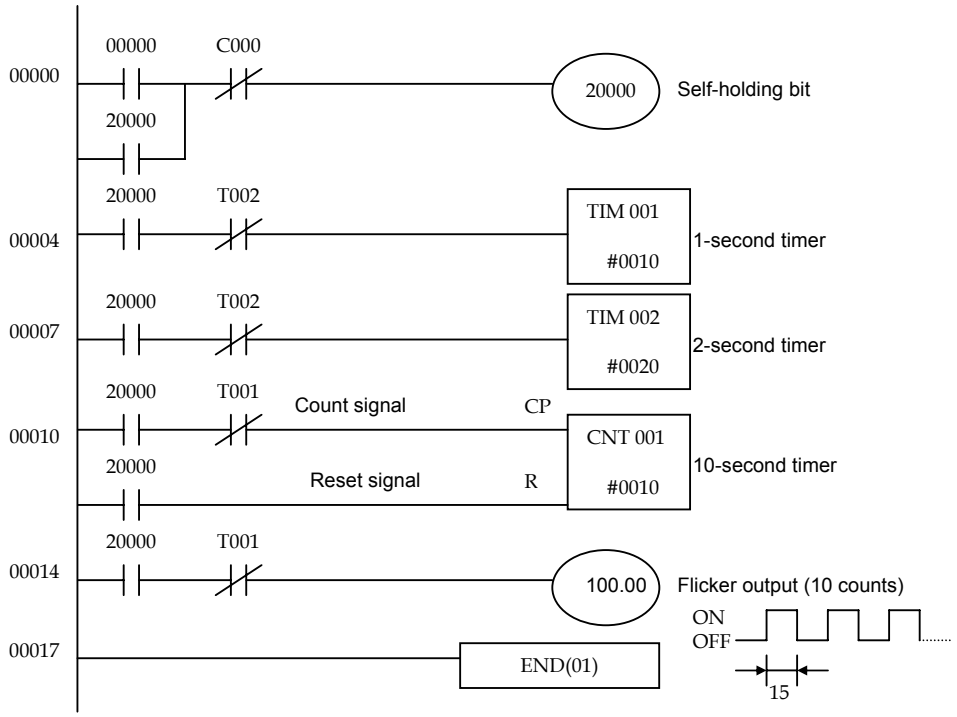
Ladder



Mnemonic Code

Đ. chỉ	Lệnh	Th.số
00000	LD	00000
00001	AND-NOT	TIM 001
00002	TIM	001
		# 6000
00003	LD	TIM 001
00004	LD	00001
00005	CNT	002
		# 6000
00006	LD	CNT 002
00007	OUT	100.00
00008	END (01)	

Ví *dụ 3* Chương trình này sẽ làm nhấp nháy (flicker) đầu ra 100.00 (bật 1 giây, tắt 1 giây) ON/OFF 10 lần sau khi bit 000.00 lên ON.

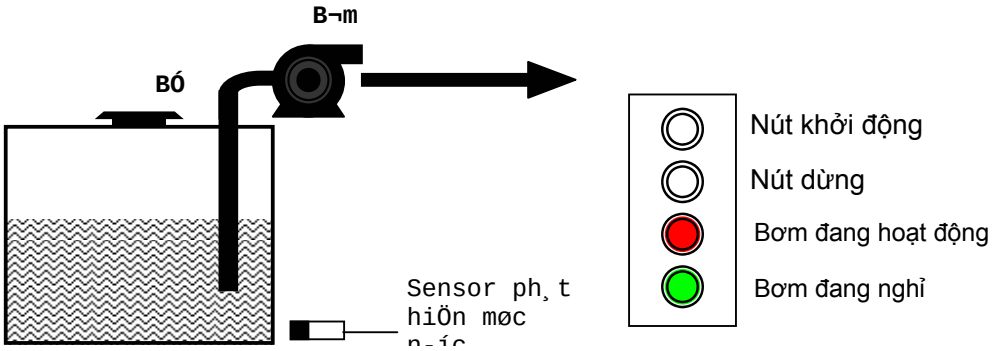


Đ. chỉ	Lệnh	Th. số		Chú thích
00000	LD		00000	(1) Self-holding bit
00001	OR		20000	
00002	AND NOT	C*	000	
00003	OUT		20000	
00004	LD		20000	(2) 1-Second timer
00005	AND NOT	T**	002	
00006	TIM		001	
		#	0010	
00007	LD		20000	(3) 2-Second timer
00008	AND NOT	T	002	
000009	TIM		002	
		#	0020	
00010	LD		20000	(4) 10-count counter
00011	AND	T	001	
00012	LD NOT		20000	
00013	CNT		000	
		#	0010	
00014	LD		20000	(5) Flicker output (10 counts)
00015	AND NOT	T	001	
00016	OUT		100.00	
00017	END(01)		- - -	(6) END(01) Lệnh

* : C= Counter
** : T = Timer

Ví dụ 4: Một hệ thống điều khiển máy bơm đơn giản

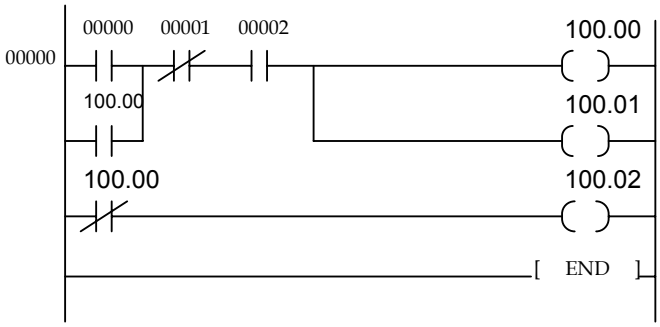
Khi nút Khởi động START được bấm, bơm sẽ kiểm tra mức nước xem có thể bơm được không qua tín hiệu từ sensor đo mức nước, nếu mức nước đạt thì bơm sẽ bơm liên tục cả khi nút Khởi động đã nhả. Bơm sẽ dừng khi nút dừng STOP được bấm hoặc khi mức nước xuống thấp quá. Kèm theo là các đèn chỉ thị tình trạng bơm.



Các đầu vào ra (I/O)

I/O	Địa chỉ trên PLC	Chức năng
INPUT	00000	Nút khởi động
	00001	Nút dừng
	00002	Sensor phát hiện mức nước
OUTPUT	100.00	Đầu ra điều khiển bơm
	100.01	Đèn báo bơm đang chạy
	100.02	Đèn báo bơm đang nghỉ

Chương trình dạng sơ đồ bậc thang



Chương trình dạng Mnemonic code:

Đ. chỉ	Lệnh	Th. số
00000	LD	00000
00001	OR	01000
00002	AND NOT	00001
00003	AND	00002
00004	OUT	100.00
00005	OUT	100.01
00006	LD NOT	100.00
00007	OUT	100.02
00008	END (01)	