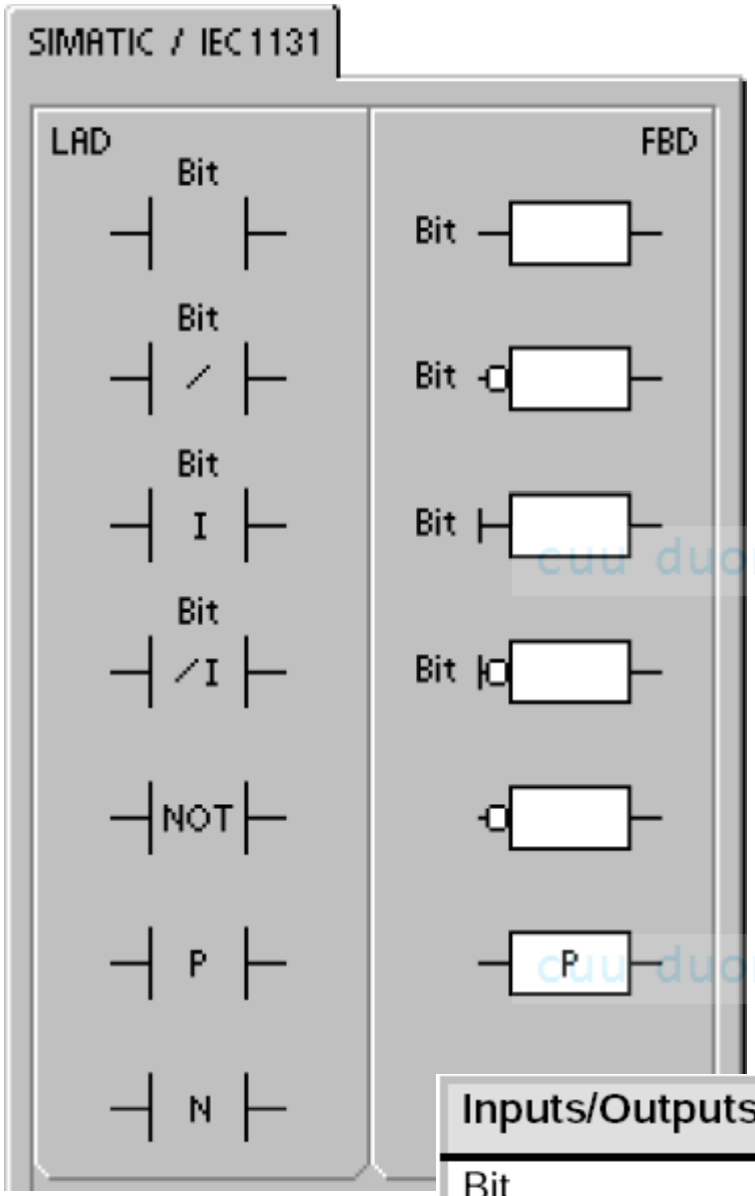
Chương 1

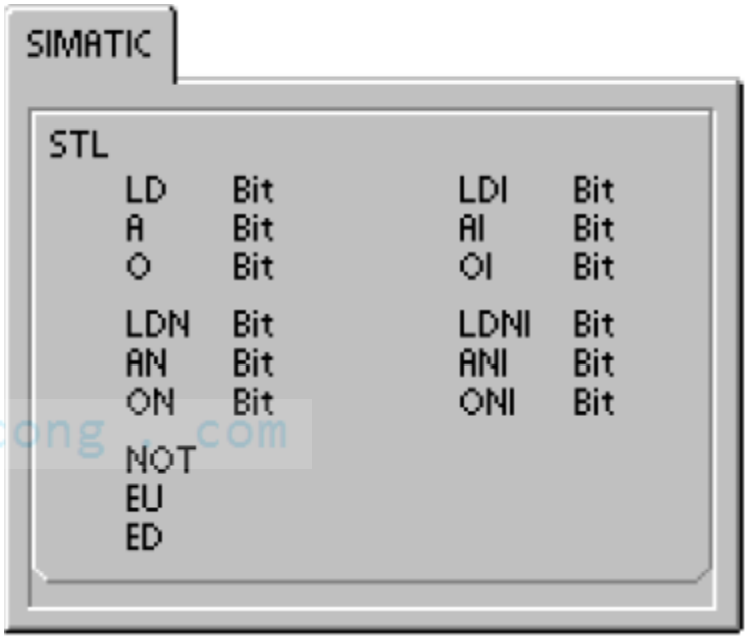
Tập lệnh PLC S7-200

Nội Dung

- Logic - Lệnh trên bit
- Timer
- Sao chép
- For…Next
- Ngắt
- PLS

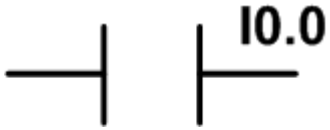
Logic – Lệnh trên Bit





Inputs/Outputs	Data Type	Operands
Bit	BOOL	I, Q, V, M, SM, S, T, C, L, Power Fl
Bit (immediate)	BOOL	I

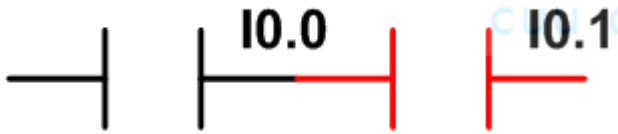
Logic – Lệnh trên Bit



Load I0.0 **LD I0.0**

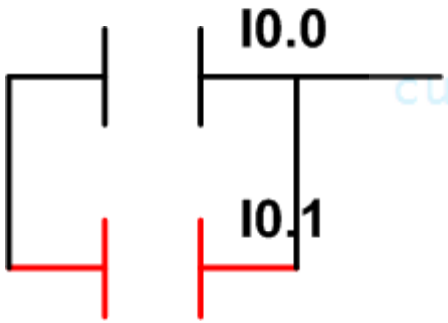
Load Not I0.0 **LDN I0.0**

Các lệnh bắt đầu một Network hay nhóm điều kiện



And I0.1 **A I0.1**

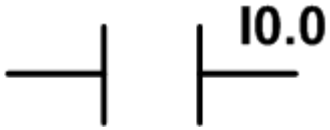
And Not I0.1 **AN I0.1**



Or I0.1 **O I0.1**

Or Not I0.1 **ON I0.1**

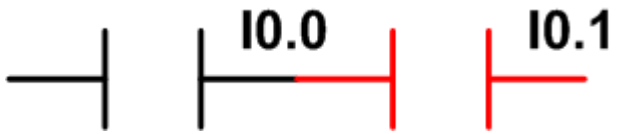
Logic – Lệnh trên Bit



Load I0.0 **LD I0.0**

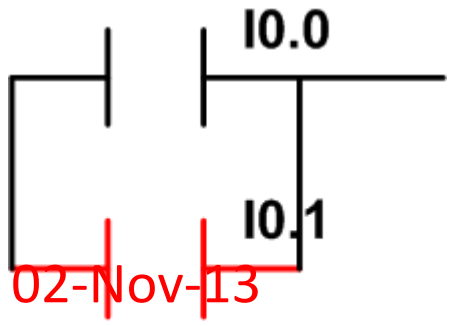
Load Not I0.0 **LDN I0.0**

Các lệnh bắt đầu một Network hay nhóm điều kiện



And I0.1 **A I0.0**

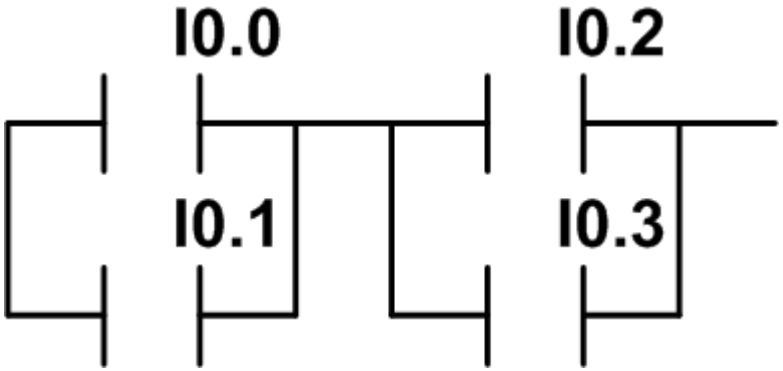
And Not I0.1 **AN I0.0**



Or I0.0 **O I0.0**

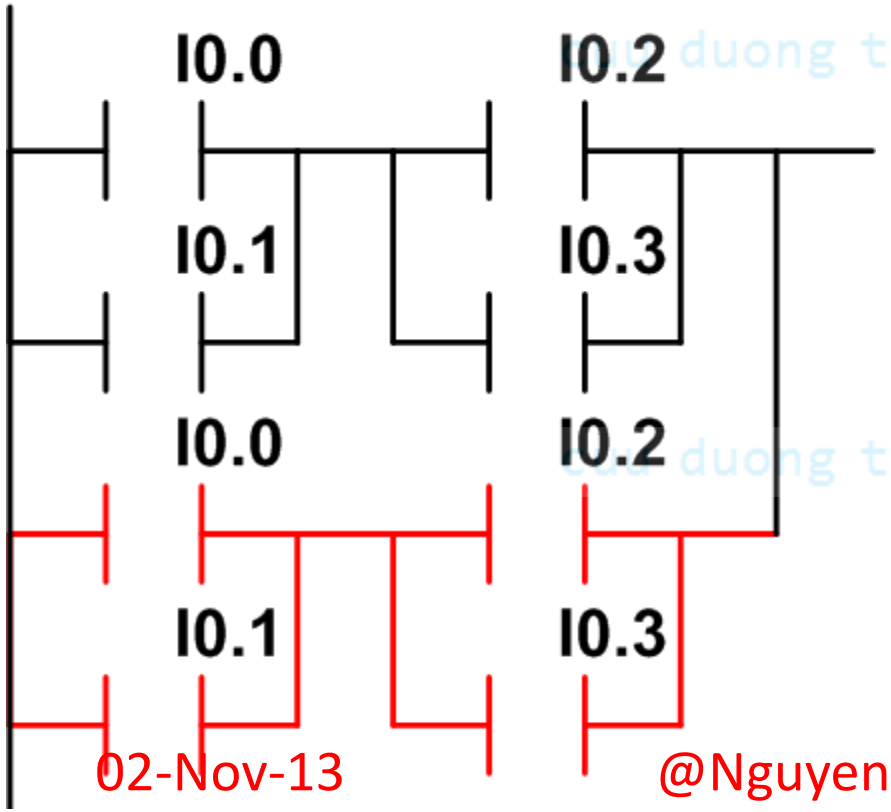
Or Not I0.0 **ON I0.0**

Logic – Lệnh trên Bit



ALD

Ghép nối tiếp 2 nhóm điều kiện



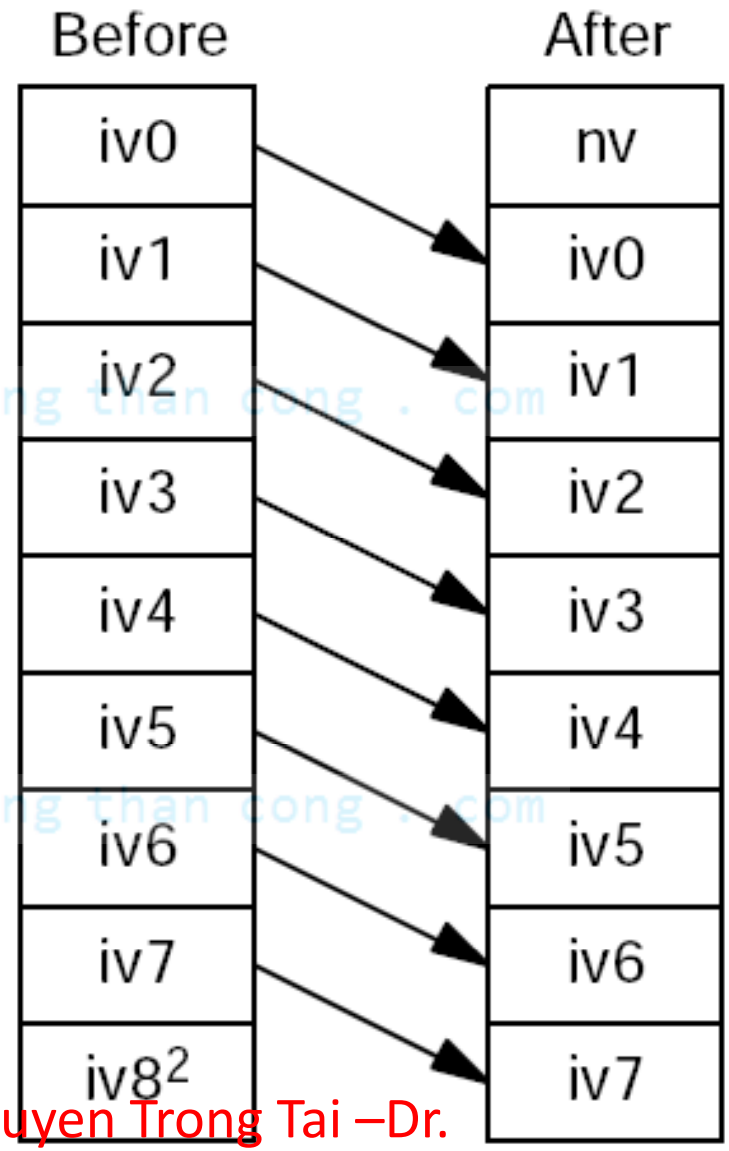
Ghép song song 2 nhóm điều kiện

OLD

Logic – Hoạt động

- **Lệnh LD, LDI, LDN, LDNI ➔ load giá trị mới vào stack**

Load (LD, LDI, LDN, LDNI)
Loads a new value (nv) onto the stack.



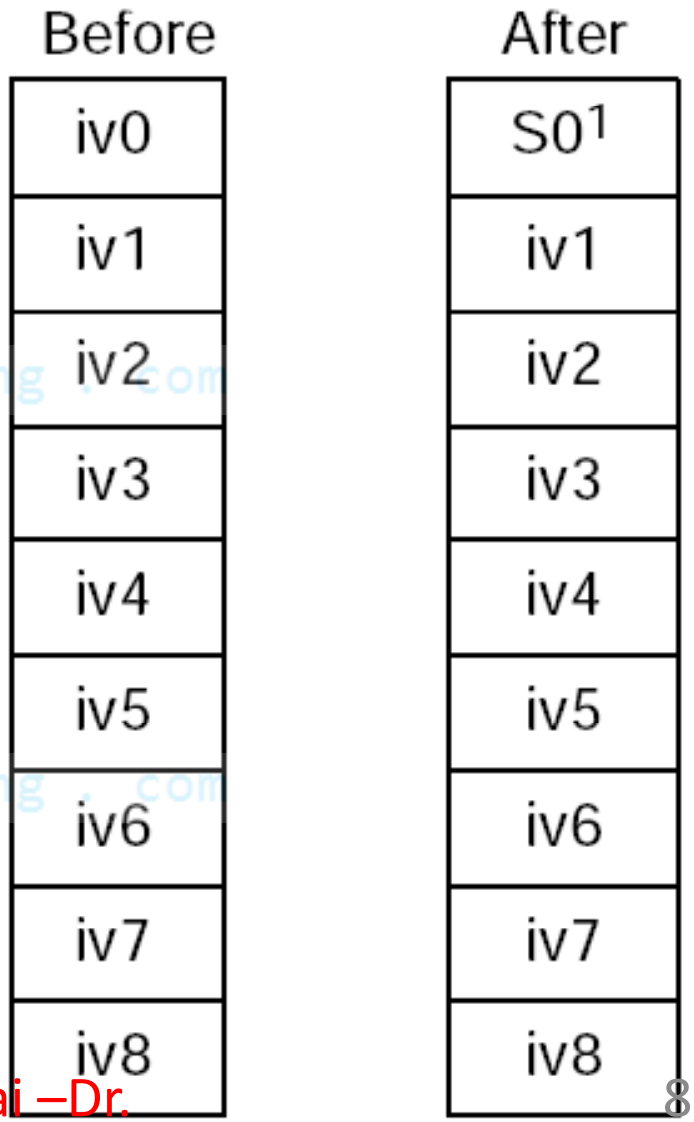
Logic – Hoạt động

❑**Lệnh And (A, AI, AN, ANI)**

And (A, AI, AN, ANI)

ANDs a new value (nv) with the initial value (iv) at the top of the stack.

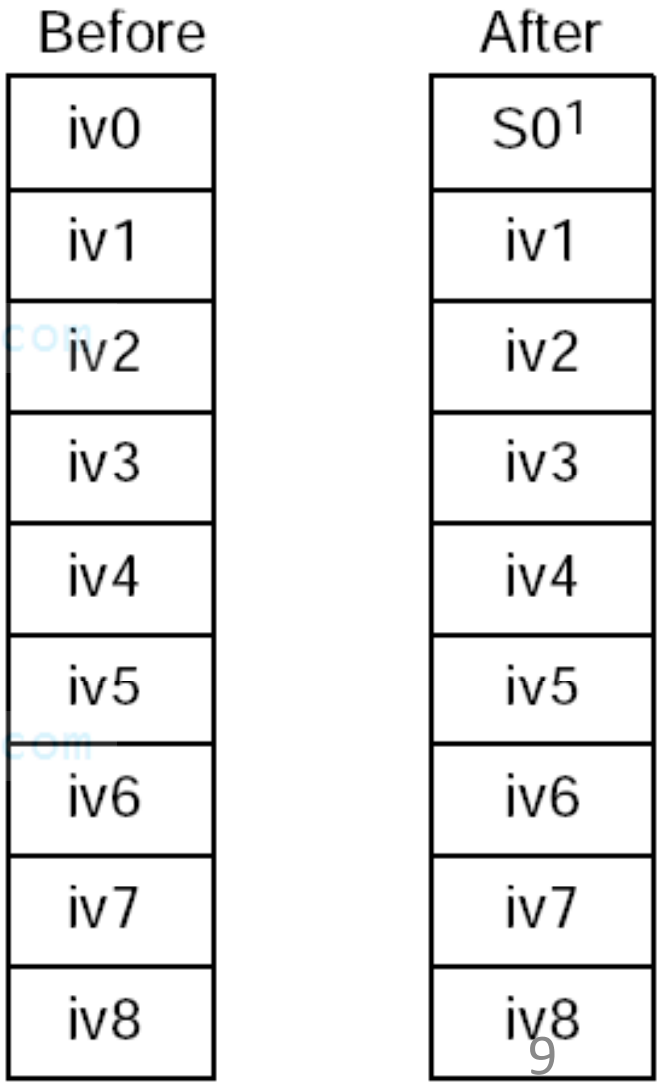
S0=iv0 AND nv



Logic – Hoạt động

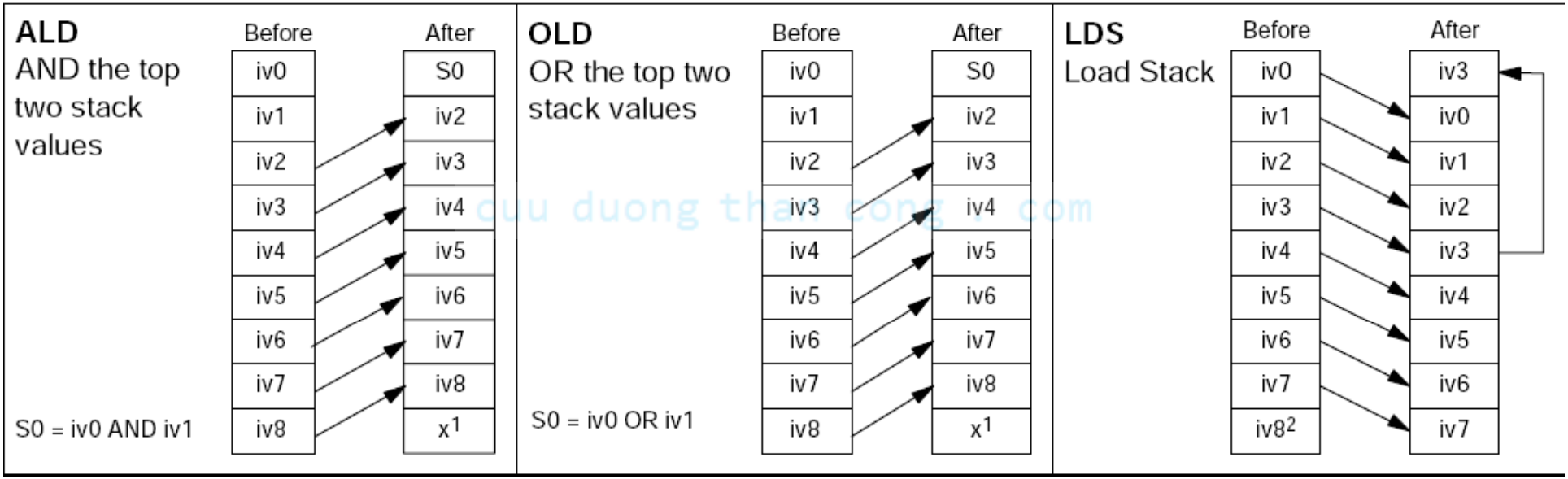
❑**Lệnh Or (O, OI, ON, ONI)**

Or (O, OI, ON, ONI)
ORs a new value (nv) with the initial value (iv) at the top of the stack.
S0=iv0 OR nv



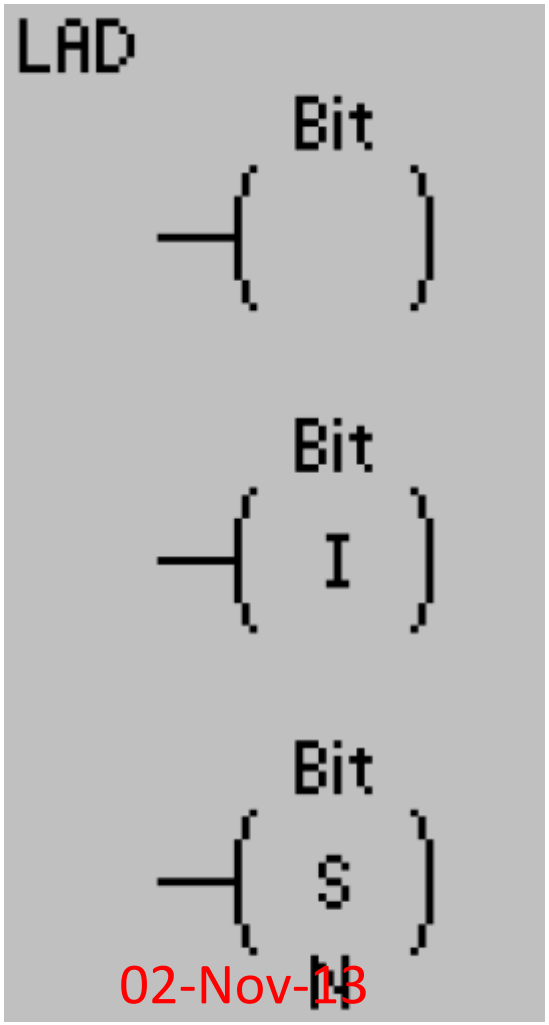
Logic – Hoạt động

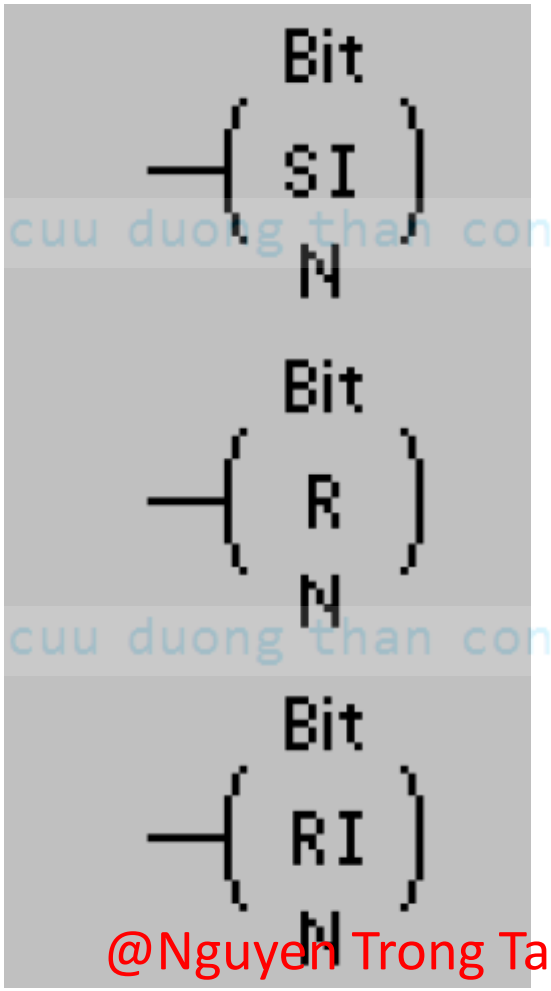
❑Lệnh And Load, Or Load

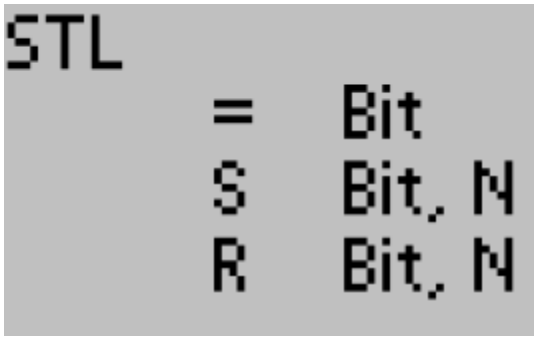


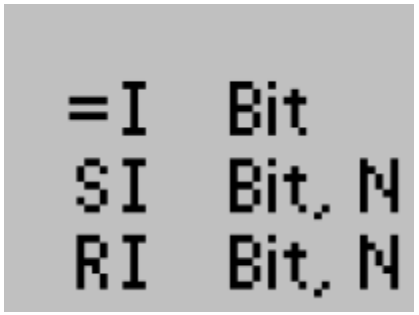
Logic – Ngõ ra

- **Lệnh ngõ ra (Out =, Out Immediate =I, Set S, Reset R, Set-Reset SR, Reset-Set RS)**









Logic – Ngõ ra

- **Lệnh ngõ ra (Out =, Out Immediate =I, Set S, Reset R, Set-Reset SR, Reset-Set RS)**

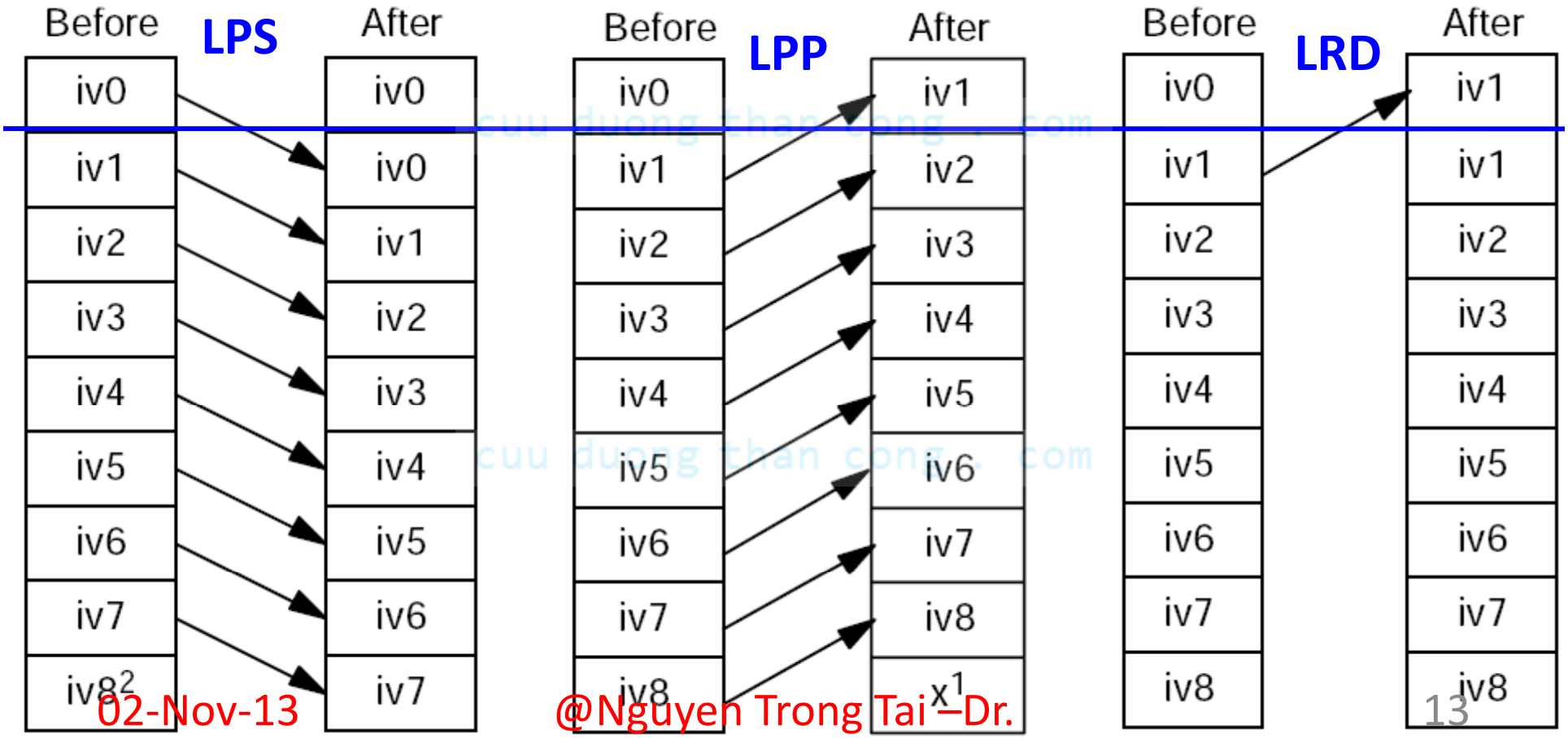
Table 6-4 Valid Operands for the Bit Logic Output Instructions

Inputs/Outputs	Data Type	Operands
Bit	BOOL	I, Q, V, M, SM, S, T, C, L
Bit (immediate)	BOOL	Q
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

Logic – Lệnh Stack

Khi có rẽ nhánh các điều kiện, các lệnh ngăn xếp được sử dụng

LPS: Logic push ➔ Lưu logic vào stack, index++
LPP: Logic pop ➔ Lấy logic từ stack, index--
LRD: Logic read ➔ Đọc logic từ stack, index

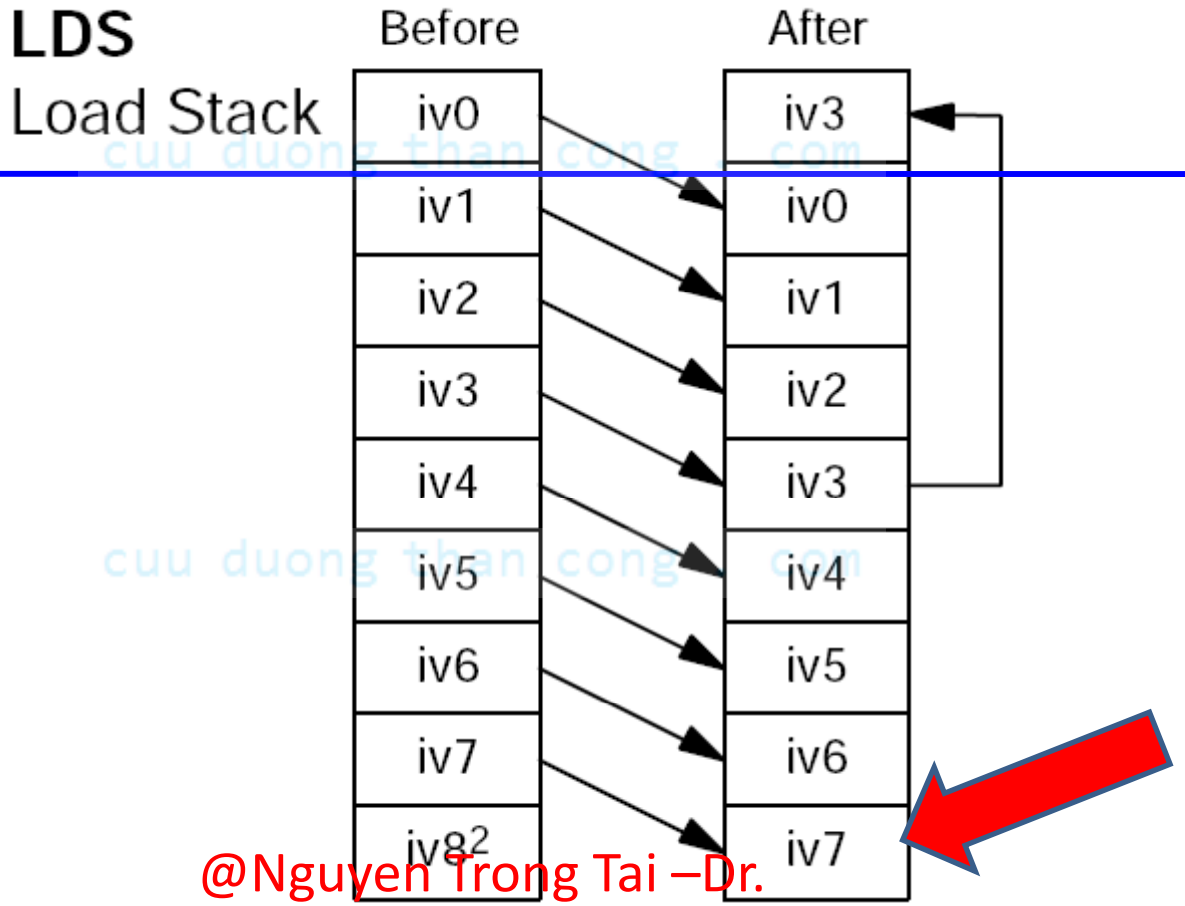


Logic – Lệnh Stack

Lệnh Load Stack

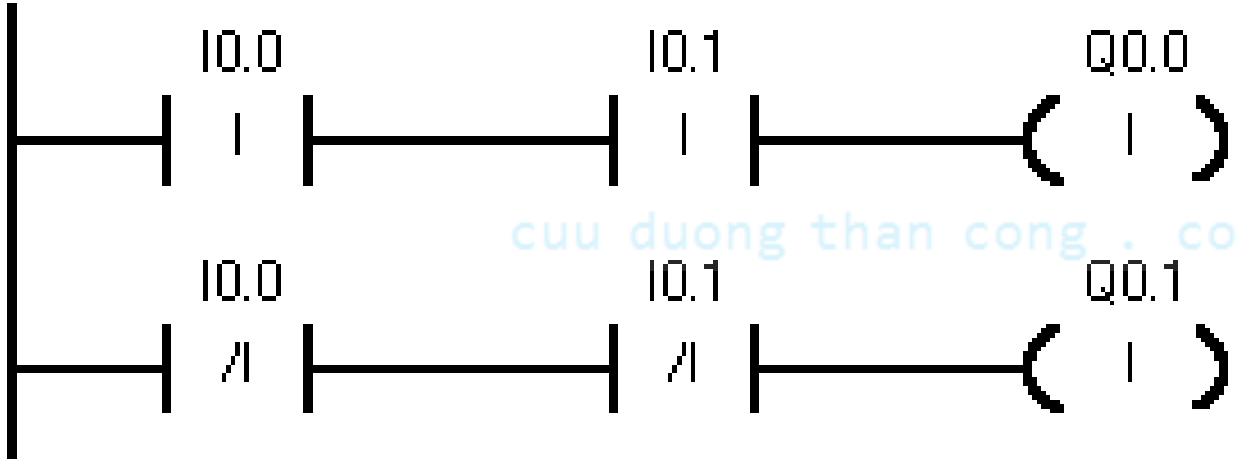
LDS N, N=1:8 (hằng số)

LDS 3



Lệnh tức thì

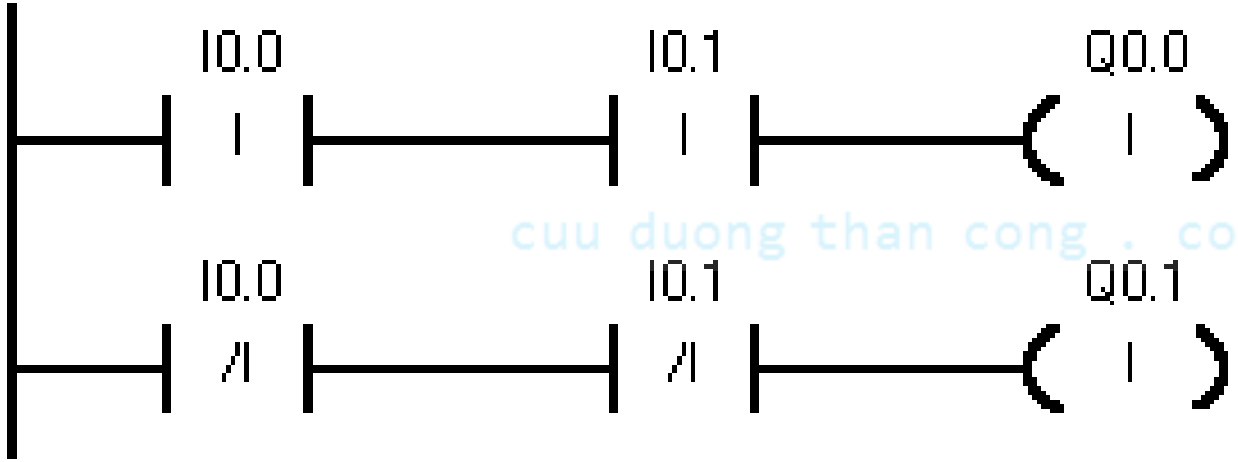
Ký hiệu bằng chữ I; bit nhớ sẽ phản ảnh ngay ngõ vào vào I, không chờ đến đầu chu kỳ quét, tương tự bít nhớ trong xuất ngay ra ngõ ra Q




```
LDI     I0.0
AI      I0.1
=I      Q0.0

LDNI    I0.0
ANI     I0.1
=I      Q0.1
```

Lệnh Lấy cạnh

Khi Chuyển trạng thái từ OFF sang ON (P) và từ ON sang OFF (N), logic sau lệnh sẽ ON trong 1 chu kỳ

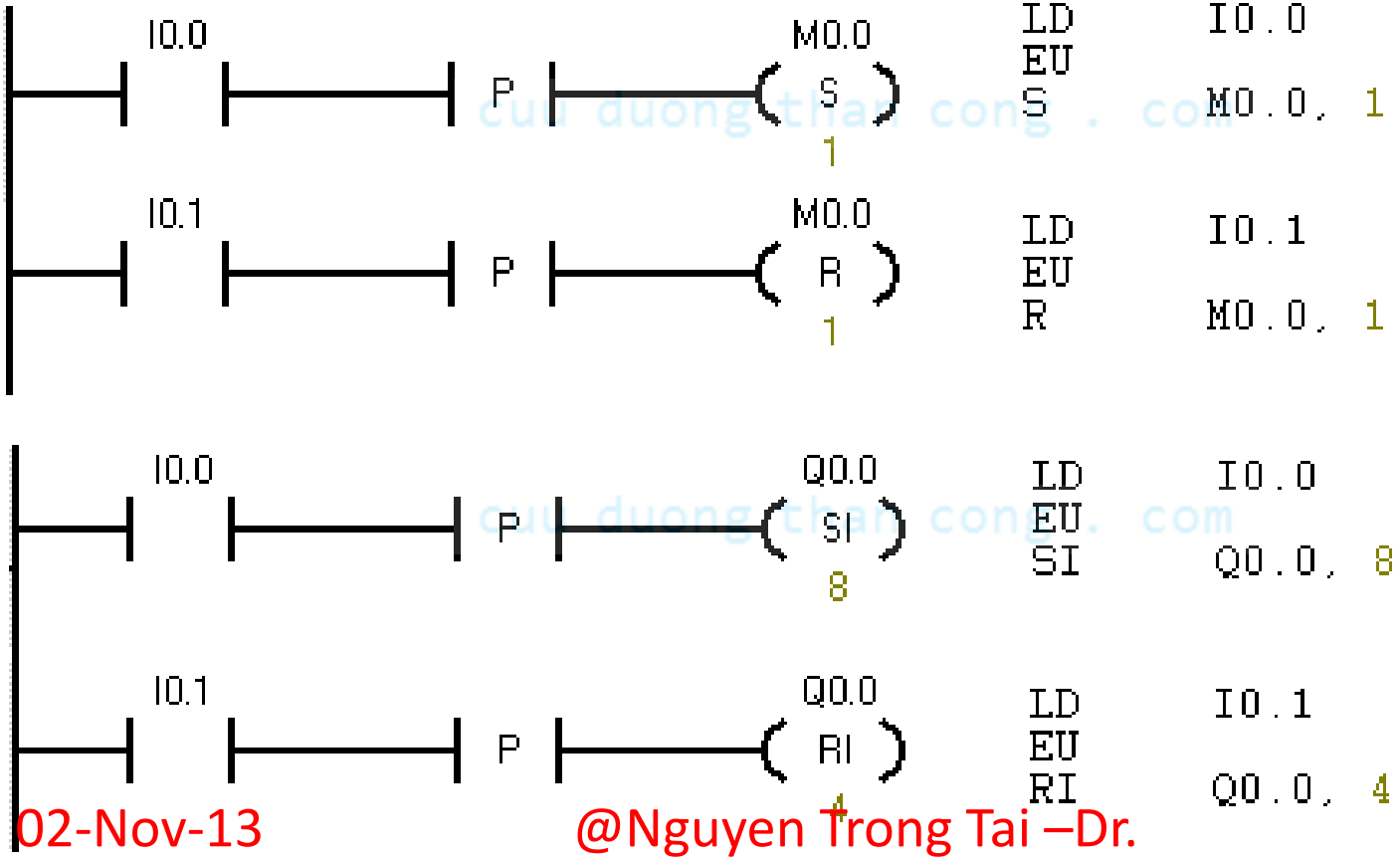



```
LDI     I0.0
AI      I0.1
=I      Q0.0

LDNI    I0.0
ANI     I0.1
=I      Q0.1
```

Lệnh Set/Reset

Lệnh S, SI cho một loạt n bit (1..255) liên tiếp nhau ON khi điều kiện ON và giữ nguyên khi điều kiện OFF trở lại

Lệnh R, RI cho một loạt n bit liên tiếp nhau OFF khi điều kiện ON và giữ nguyên khi điều kiện OFF trở lại



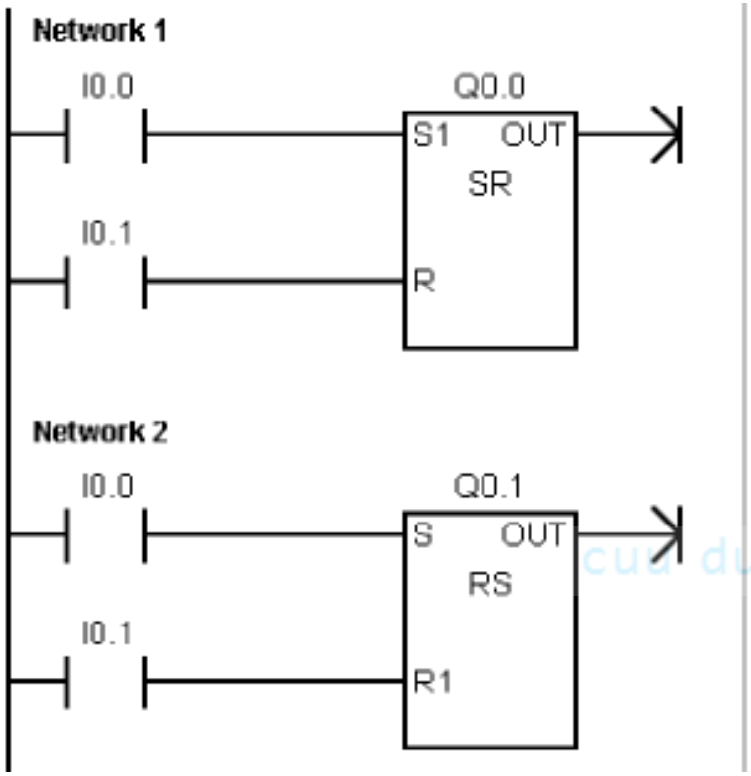
Lệnh Set/Reset

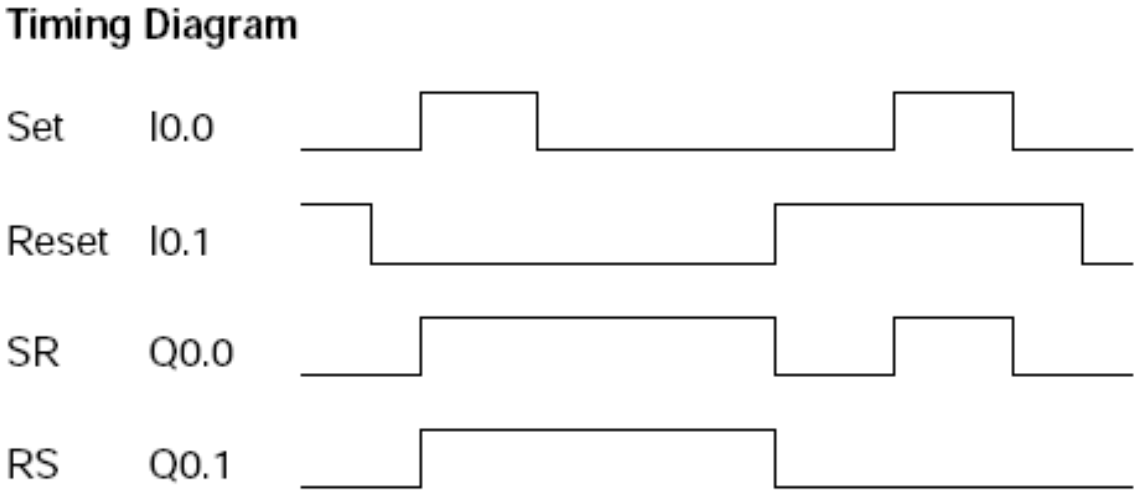
Lệnh SR có hai ngõ vào S và R
Lệnh RS có hai ngõ vào S và R

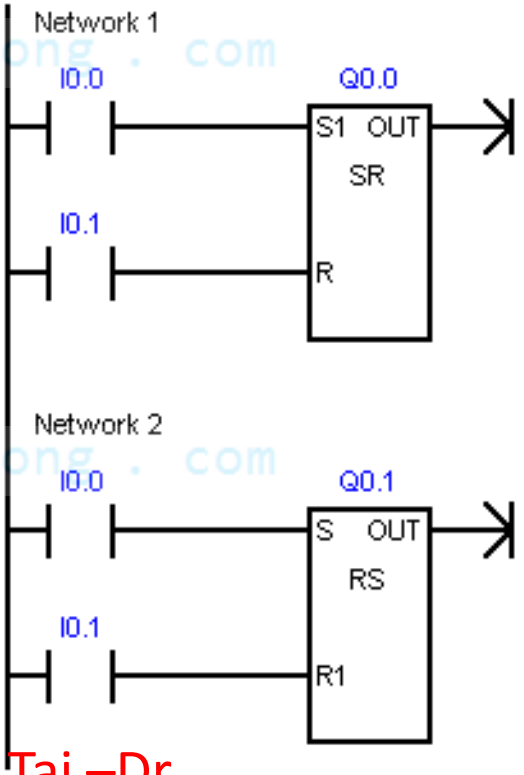
Instruction	S1	R	Out (Bit)
Set Dominant Bistable instruction (SR)	0	0	Previous state
	0	1	0
	1	0	1
	1	1	1
Instruction	**S**	**R1**	**Out (Bit)**
Reset Dominant Bistable instruction (RS)	0	0	Previous state
	0	1	0
	1	0	1
	1	1	0

Inputs/Outputs	Data Types	Operands
S1, R	BOOL	I, Q, V, M, SM, S, T, C, Power Flow
S, R1, OUT	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
Bit	BOOL	I, Q, V, M, S

Lệnh Set/Reset



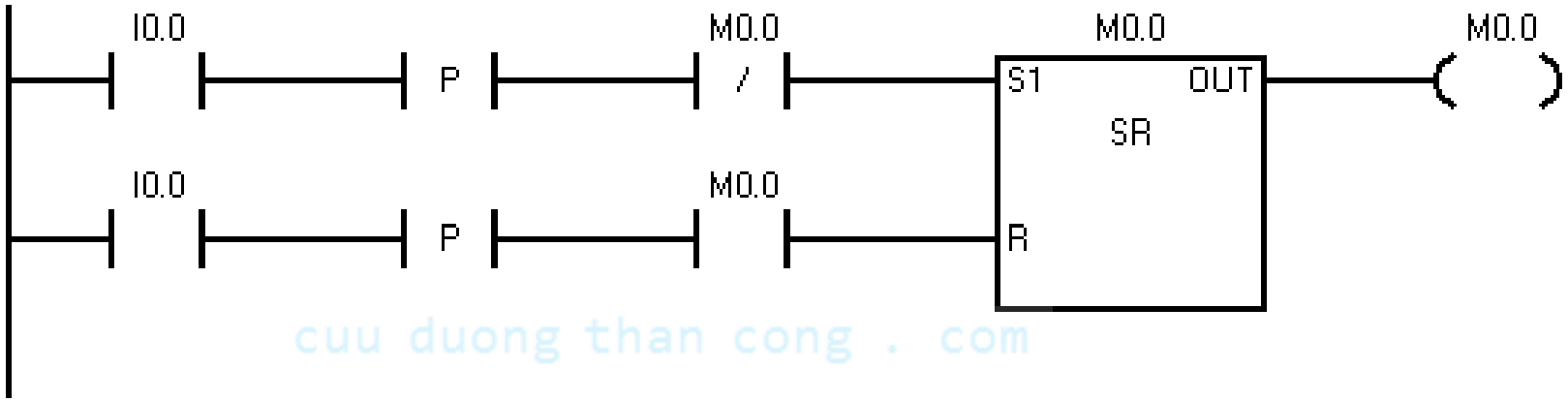


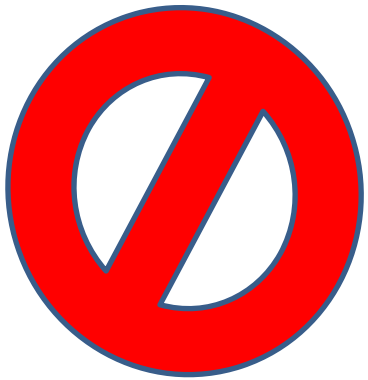


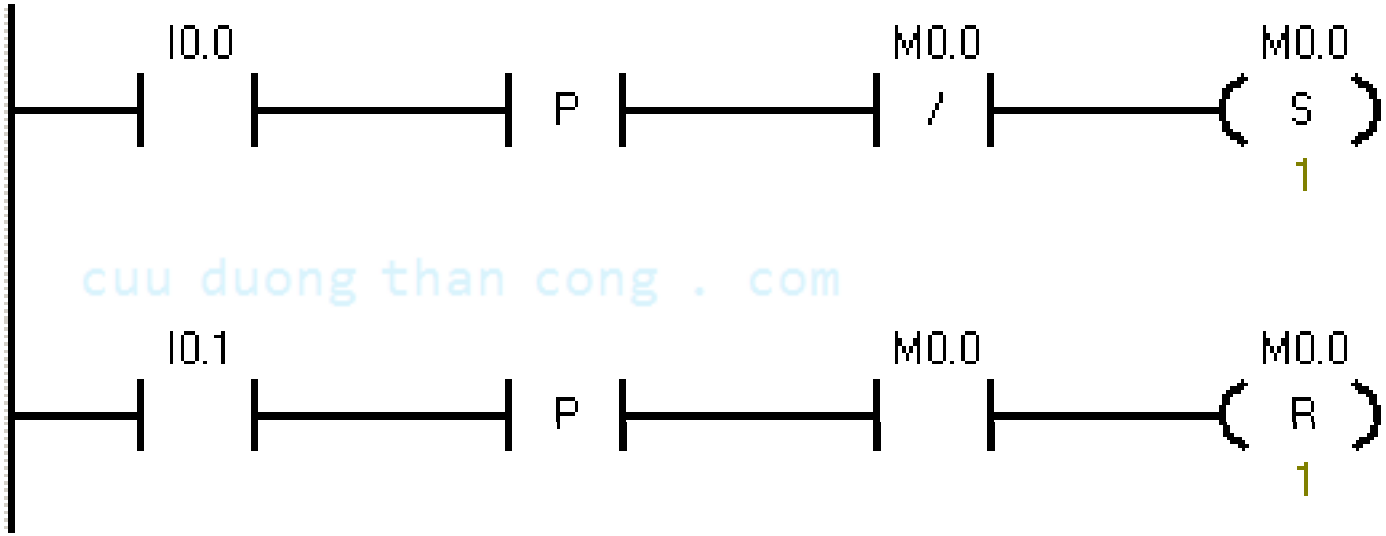

```
NETWORK 1
LD I0.0
LD I0.1
NOT
A Q0.0
OLD
= Q0.0

NETWORK 2
LD I0.0
LD I0.1
NOT
LPS
A Q0.1
= Q0.1
LPP
ALD
O Q0.1
= Q0.1
```

Lệnh Set/Reset

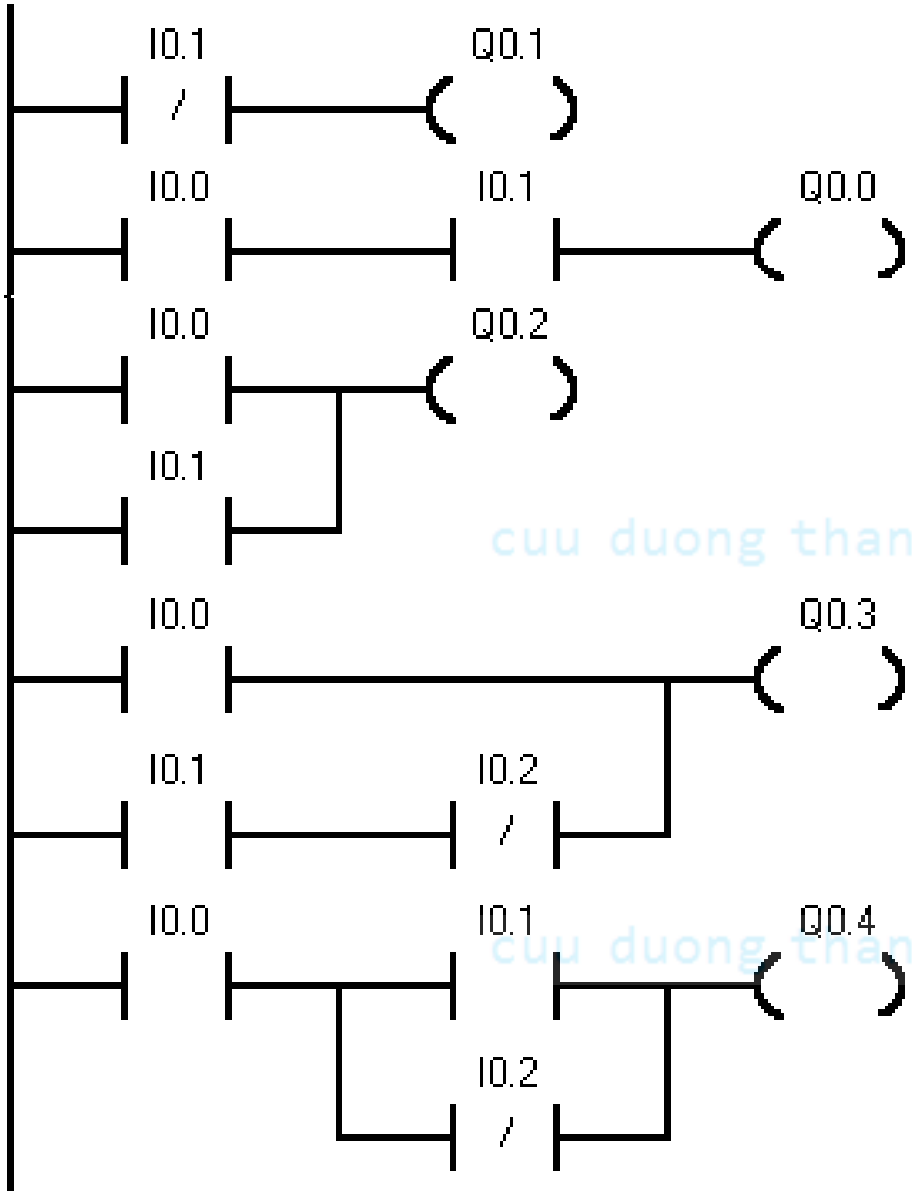
Ví dụ: bấm I0.0 thì M0.0 được set, bấm lần nữa thì M0.0 bị xóa





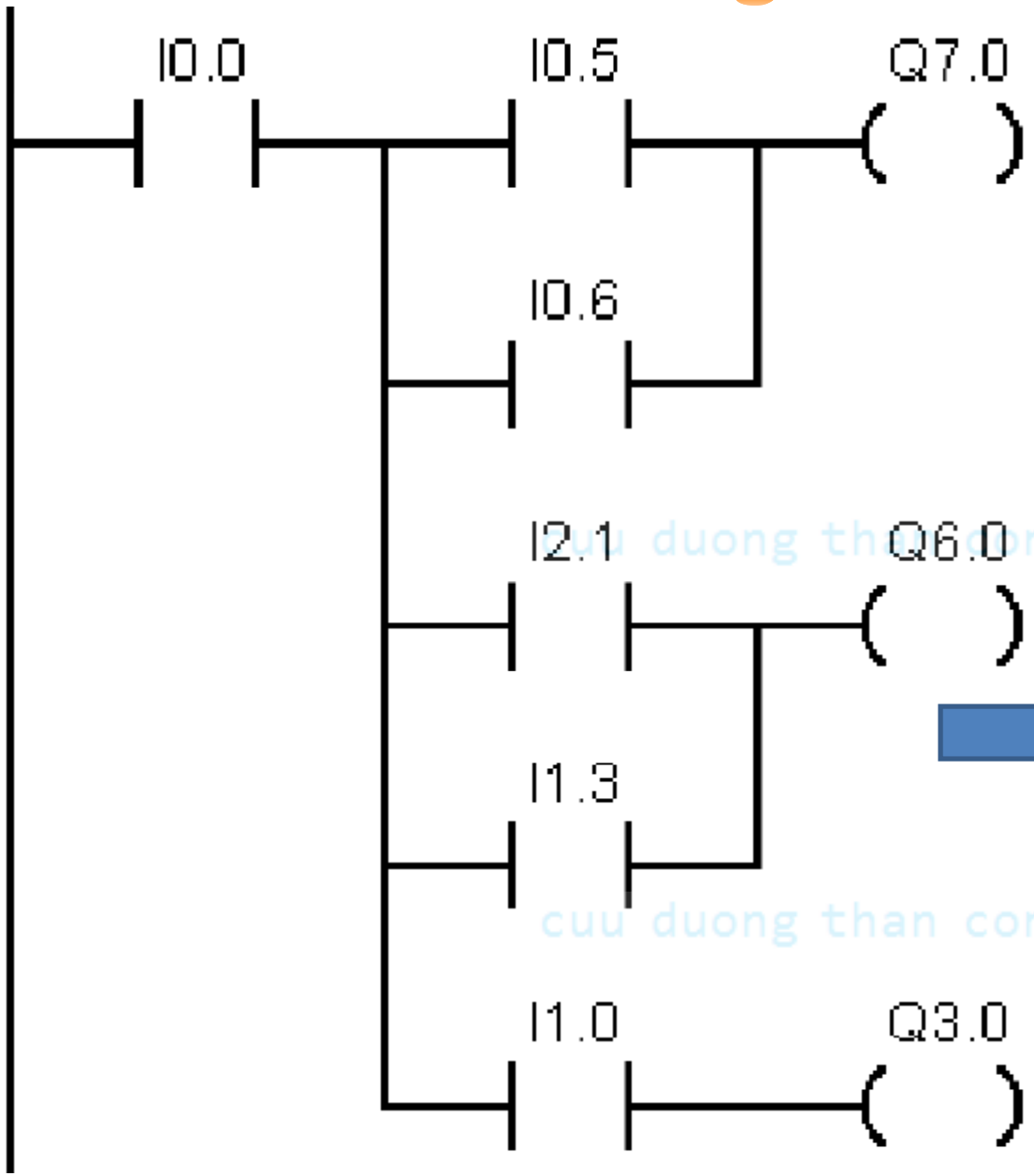


Logic – Ví dụ




```
LDN     I0.1
=       Q0.1

LD      I0.0
A       I0.1
=       Q0.0

LD      I0.0
O       I0.1
=       Q0.2

LD      I0.0
LD      I0.1
AN      I0.2
OLD
=       Q0.3
LD      I0.0
LD      I0.1
ON      I0.2
ALD
=       Q0.4
```

Logic – Ví dụ




```
LD      I0.0
LPS
LD      I0.5
O       I0.6
ALD
=       Q7.0
LRD
LD      I2.1
O       I1.3
ALD
=       Q6.0
LPP
A       I1.0
=       Q3.0
```

Logic – Ví dụ

LD I0.0	OW= AIW0, 100	ALD
LPS	ALD	A I0.4
LD M0.0	LPS	MOVW 0, MW2
AN M0.1	A I0.1	LRD
LDN M0.2	S V10.0, 3	A I0.6
A M0.3	LPP	ATCH INT_0, 8
OLD	A T37	LPP
ALD	= Q0.0	A V0.1
LPS	LPP	LD V0.2
A V0.0	A I0.2	ON V0.3
TON T37, 10	LPS	ALD
LPP	LD I0.3	SI Q2.0, 2
LDW<= AIW0, 20	O I0.5	

Lệnh so sánh

So sánh 2 số n1 và n2 số theo kiểu char (Byte) không dấu

<B; <=B ; ==B; >B; >=B; <>B

So sánh 2 số n1 và n2 số theo số nguyên 16bit (Interger) có dấu

<I; <=I ; ==I; >I; >=I; <>I

So sánh 2 số n1 và n2 số theo số nguyên 32bit (Interger) có dấu

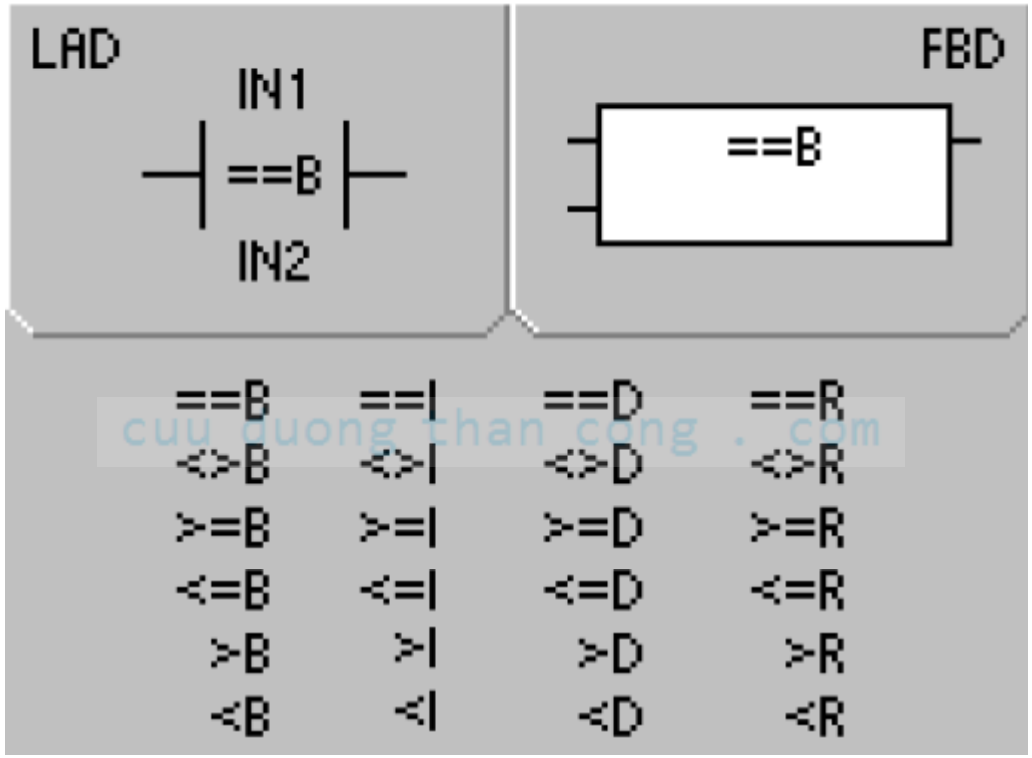
<D; <=D ; ==D; >D; >=D; <>D

So sánh 2 số n1 và n2 số theo số thực (Real) có dấu

<R; <=R ; ==R; >R; >=R; <>R

Lệnh so sánh

LADDER



Lệnh so sánh

STL

LDB= IN1,IN2	Load KQ So sánh vào stack
AB= IN1, IN2	And KQ so sánh với stack
OB= IN1,IN2	Or KQ so sánh với stack

LDB=	LDW=	LDD=	LDR=
LDB<	LDW<	LDD<	LDR<
LDB>	LDW>	LDD>	LDR>
LDB<>	LDW<>	LDD<>	LDR<>
LDB<=	LDW<=	LDD<=	LDR<=
LDB>=	LDW>=	LDD>=	LDR>=

Lệnh so sánh

STL

LDB= IN1,IN2
AB= IN1, IN2
OB= IN1,IN2

AB=	AW=	AD=	AR=
AB<	AW<	AD<	AR<
AB>	AW>	AD>	AR>
AB<>	AW<>	AD<>	AR<>
AB<=	AW<=	AD<=	AR<=
AB>=	AW>=	AD>=	AR>=

Lệnh so sánh

STL

```
LDB= IN1,IN2
AB=   IN1, IN2
OB=  IN1,IN2
```

OB=	OW=	OD=	OR=
OB<	OW<	OD<	OR<
OB>	OW>	OD>	OR>
OB<>	OW<>	OD<>	OR<>
OB<=	OW<=	OD<=	OR<=
OB>=	OW>=	OD>=	OR>=

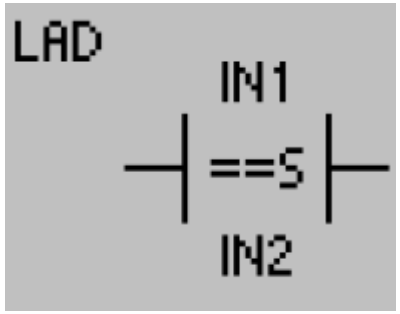
Lệnh so sánh

Vùng dữ liệu được phép sử dụng trong lệnh so sánh

Inputs/Outputs	Type	Operands
IN1, IN2	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
Output (or OUT)	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow

Lệnh so sánh chuỗi ký tự

So sánh 2 chuỗi IN1 = IN2; IN1<>IN2




```
STL
                LDS=   IN1, IN2
                AS=    IN1, IN2
                OS=    IN1, IN2

                LDS<>  IN1, IN2
                AS<>   IN1, IN2
                OS<>   IN1, IN2
```


Inputs/Outputs	Type	Operands
IN1	STRING	VB, LB, *VD, *LD, *AC, constant
IN2	STRING	VB, LB, *VD, *LD, *AC
Output (OUT)	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow

Timer

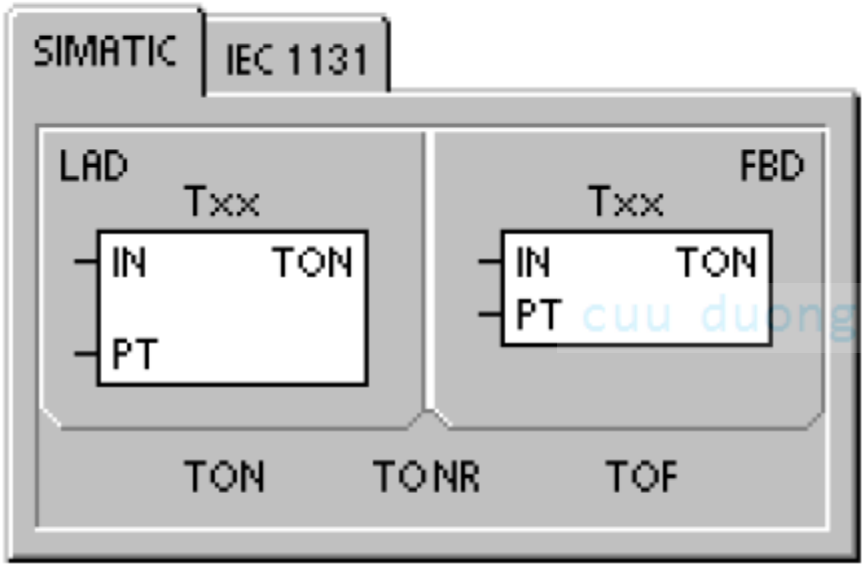
PLC S7-200 cung cấp 256 timer gồm các loại sau

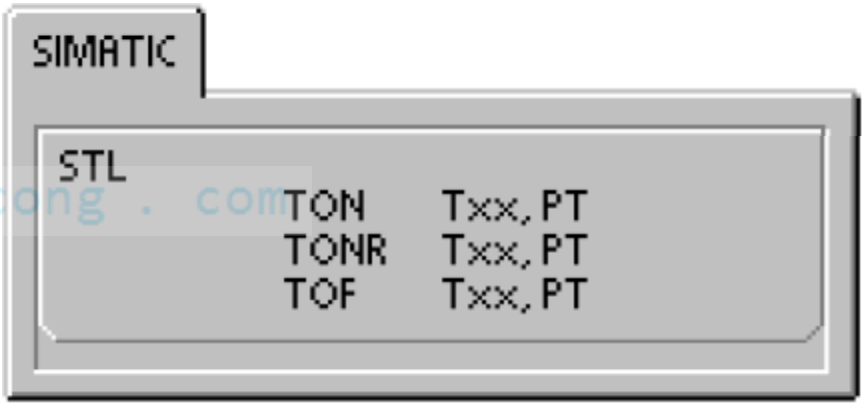
Timer Type	Resolu-tion	Maximum Value	Timer Number
TONR	1ms	32.767 s	T0, T64
	10ms	327.67 s	T1 – T4 ; T65 – T68
	100ms	3276.7 s	T5 – T31; T69 – T 95
TON, TOF	1ms	32.767 s	T32, T96
	10ms	327.67 s	T33 – T36; T97 – T100
	100ms	3276.7 s	T37 – T63; T101 – T255

Timer

PLC S7-200 cung cấp 256 timer gồm các loại sau

N (LAD): Power Flow BOOL
IN (FBD): I, Q, M, SM, T, C, V, S, L, Power Flow BOOL
PT: VW, IW, QW, MW, SW, SMW, LW, AIW, T, C, AC, Constant, *VD, *LD, *AC INT





Timer

PLC S7-200 cung cấp 256 timer gồm các loại sau:

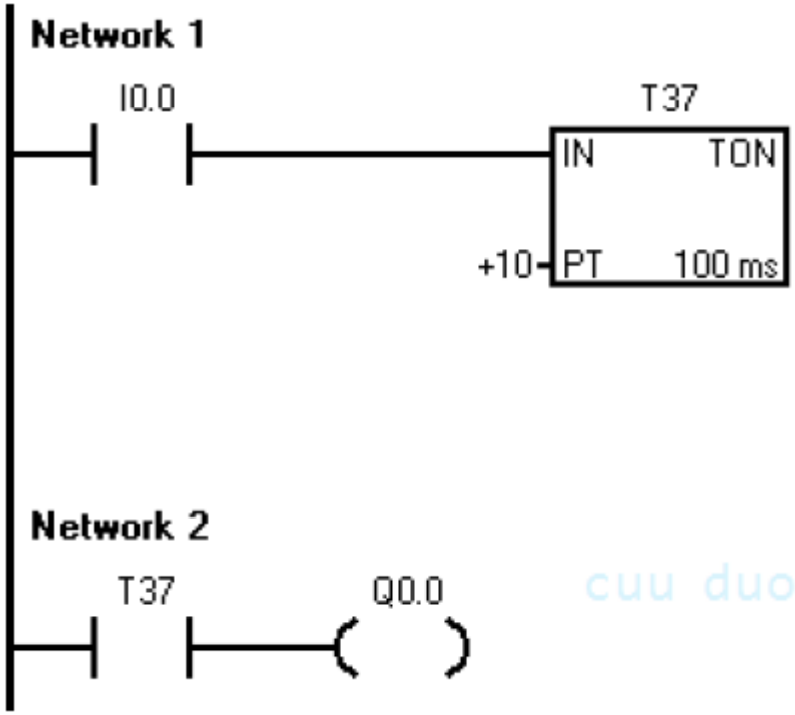
TON: Định thời cho đơn

TONR: Định thời có tích lũy

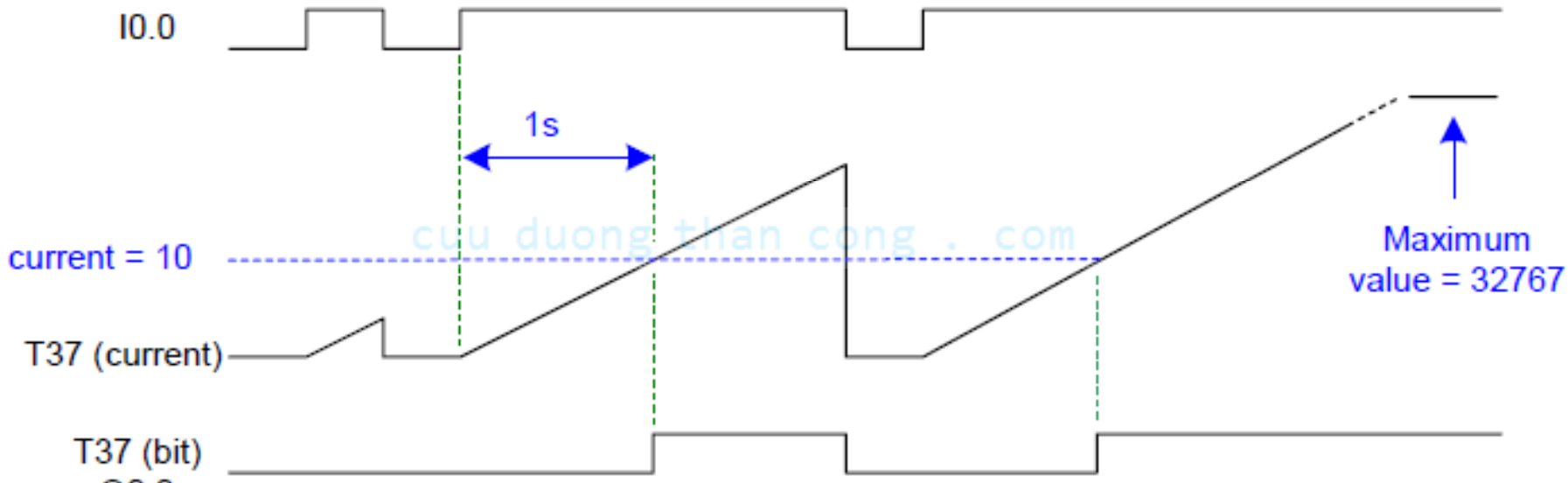
TOF: kéo dài thời gian tắt thiết bị

Type	Current >= Preset	State of the Enabling Input (IN)	Power Cycle/First Scan
TON	Timer bit on Current continues counting to 32,767	ON: Current value counts time OFF: Timer bit off, current value = 0	Timer bit off Current value = 0
TONR	Timer bit on Current continues counting to 32,767	ON: Current value counts time OFF: Timer bit and current value maintain last state	Timer bit off Current value can be maintained[1]
TOF	Timer bit off Current = Preset, stops counting	ON: Timer bit on, current value = 0 OFF: Timer counts after on-to-off transition	Timer bit off Current value = 0

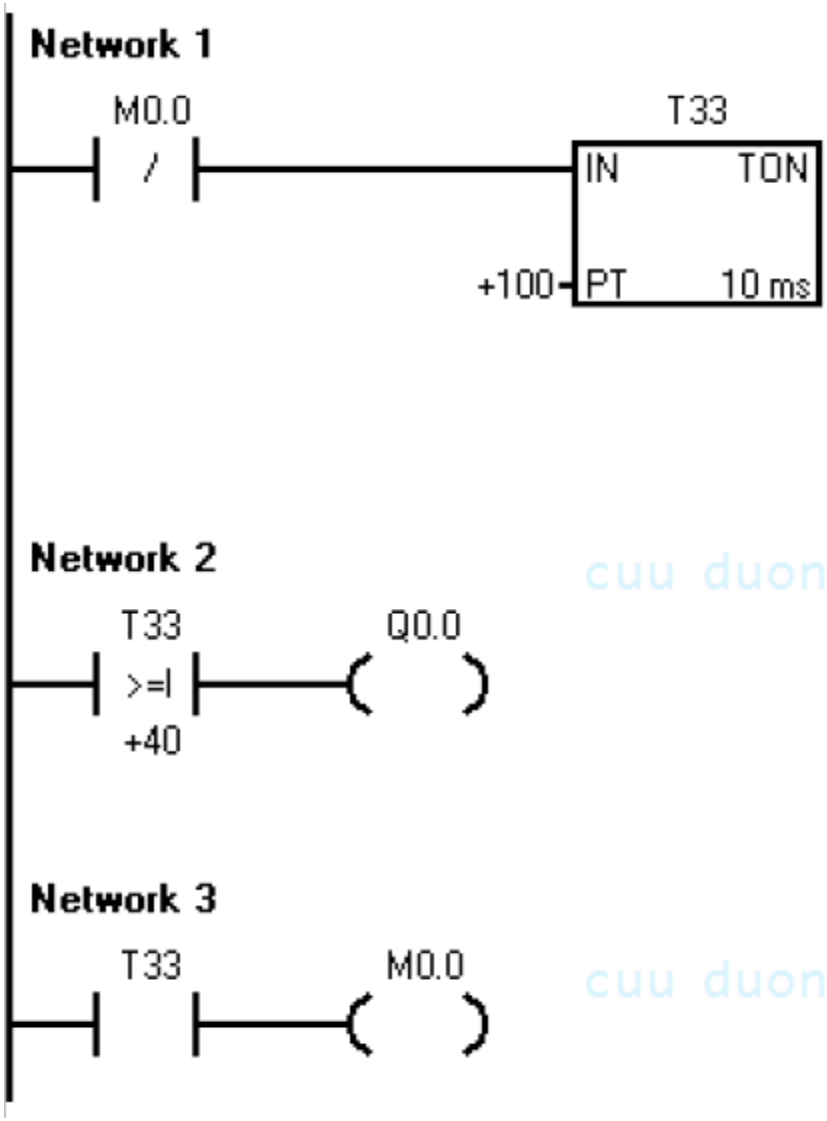
Timer ON




```
Network 1       //100 ms timer T37 times out after
                //(10 x 100 ms = 1s)
                //I0.0 ON=T37 enabled,
                //I0.0 OFF=disable and reset T37
LD      I0.0
TON     T37, +10

Network 2       //T37 bit is controlled by timer T37
LD      T37
=       Q0.0
```

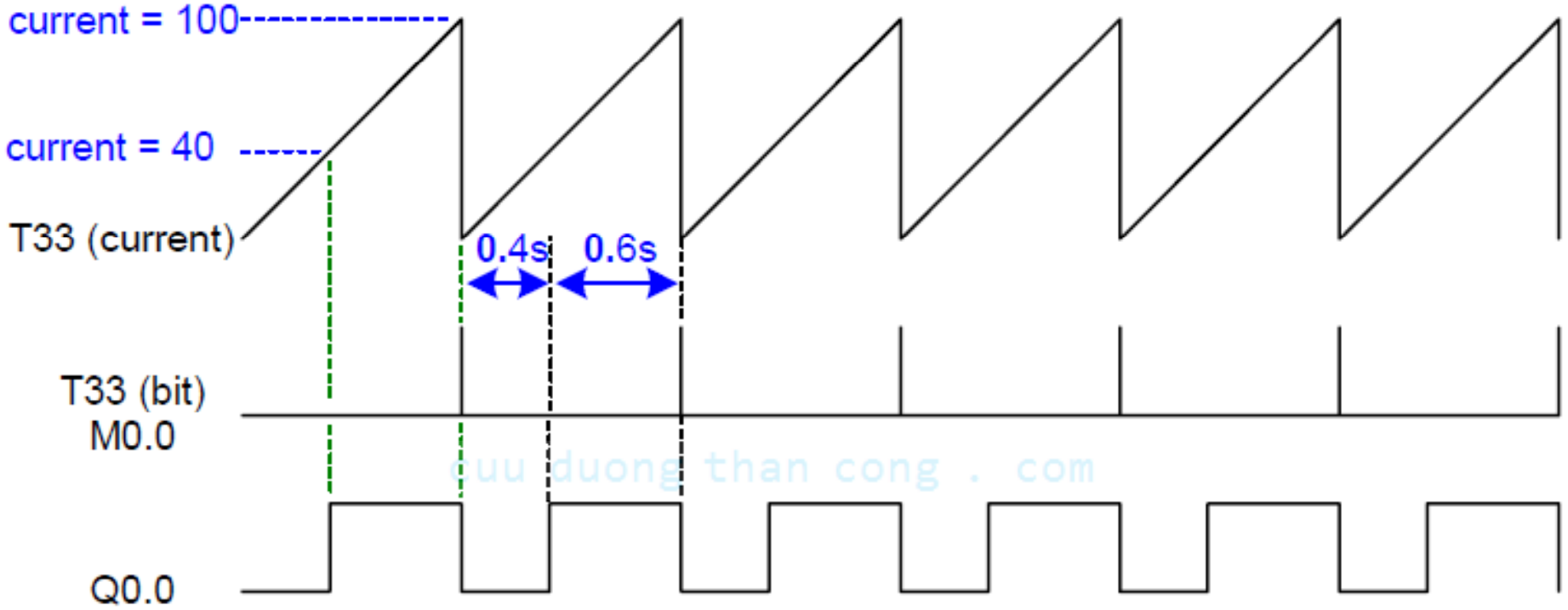


Timer ON

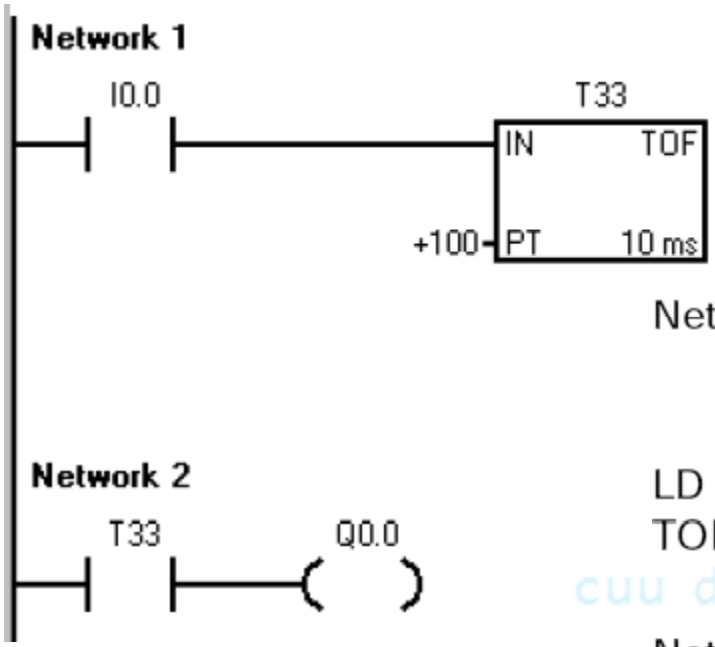



```
Network 1       //10 ms timer T33 times out after
                //(100 x 10 ms = 1s)
                //M0.0 pulse is too fast to monitor
                //with Status view
LDN     M0.0
TON     T33, +100
Network 2       //Comparison becomes true at a
                //rate that is visible with Status view
                //Turn on Q0.0 after (40 x 10 ms)
                //for a 40% OFF/60% ON waveform
LDW>=   T33, +40
=       Q0.0
Network 3       //T33 (bit) pulse too fast to monitor
                //with Status view
                //Reset the timer through M0.0 after
                //the (100 x 10 ms)  period
LD      T33
=       M0.0
```

Timer ON



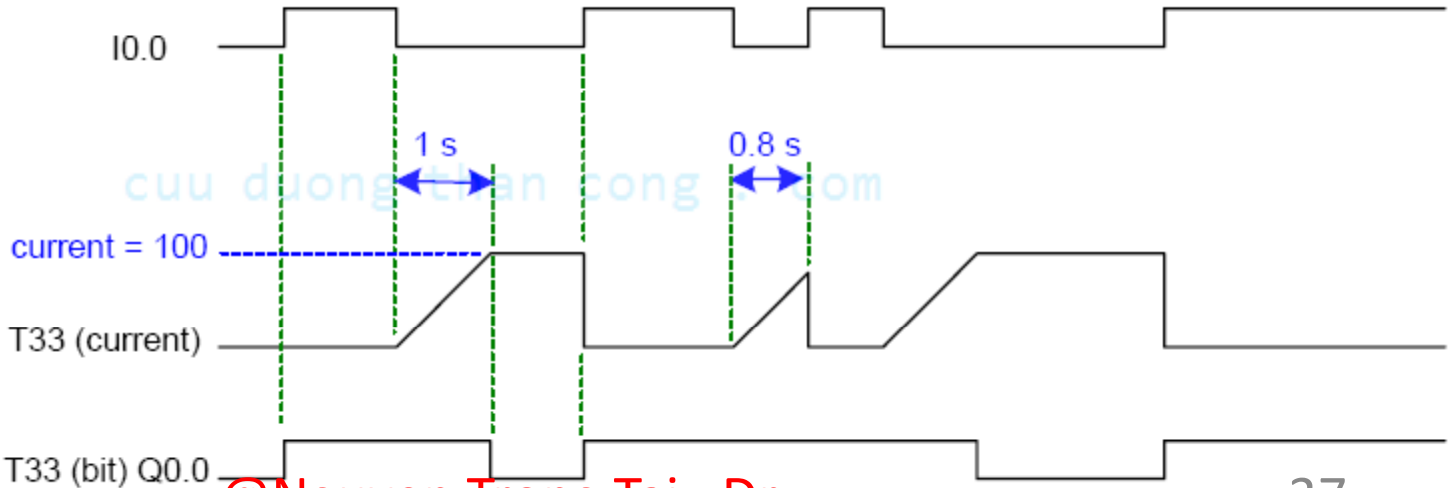
Timer OF



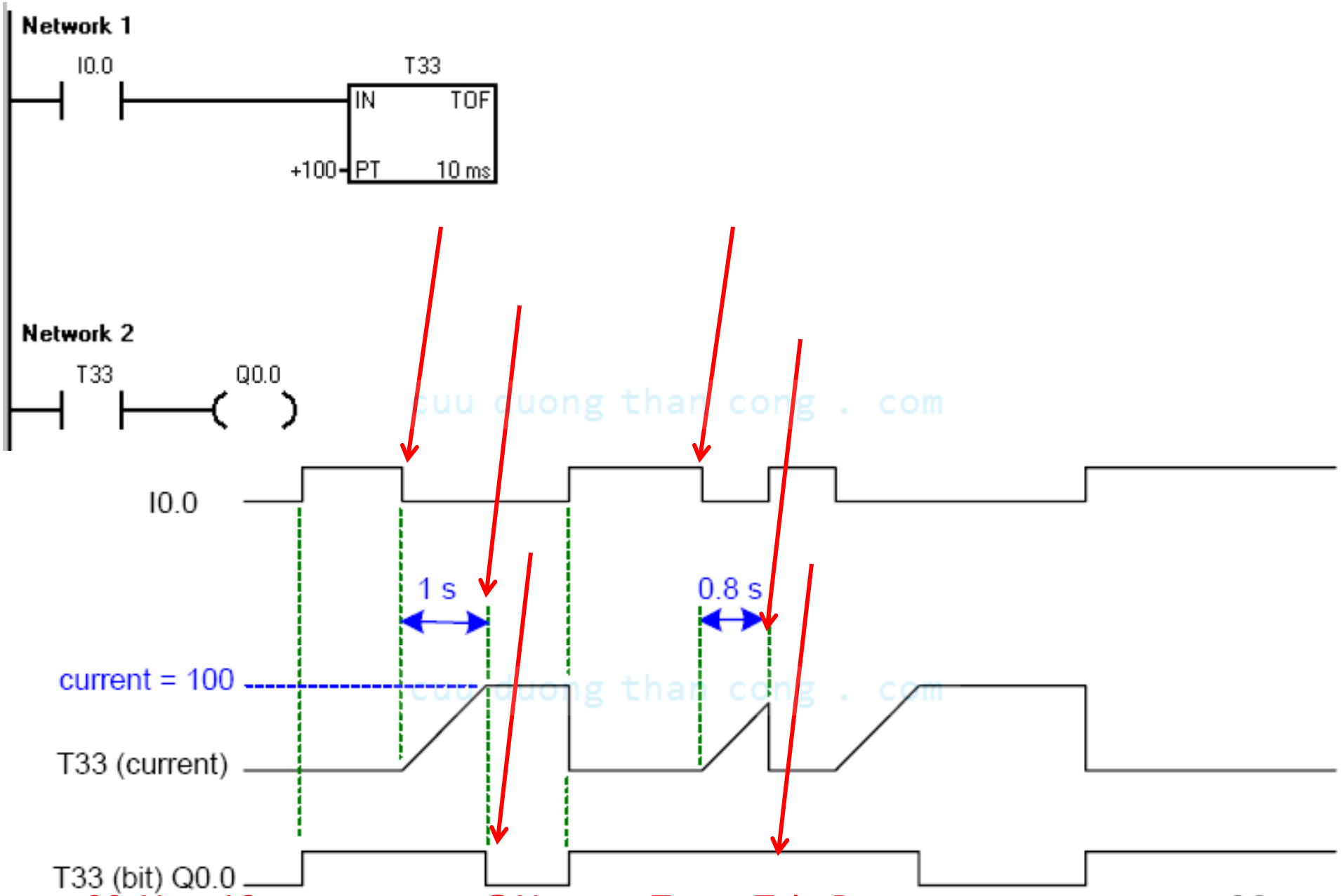

```
Network 1       //10-ms timer T33 times out after (100 x 10 ms = 1s)
                //I0.0 ON-to-OFF=T33 enabled
                //I0.0 OFF-to-ON=disable and reset T33
LD      I0.0
TOF     T33, +100
```



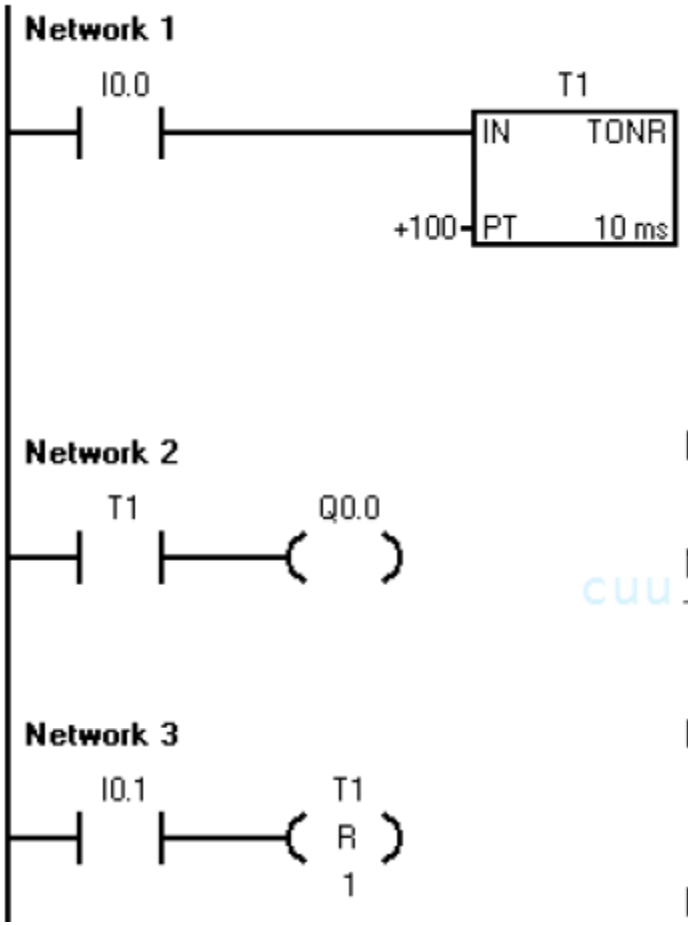
```
Network 2       //Timer T33 controls Q0.0 through timer contact T33
LD      T33
=       Q0.0
```



Timer OF

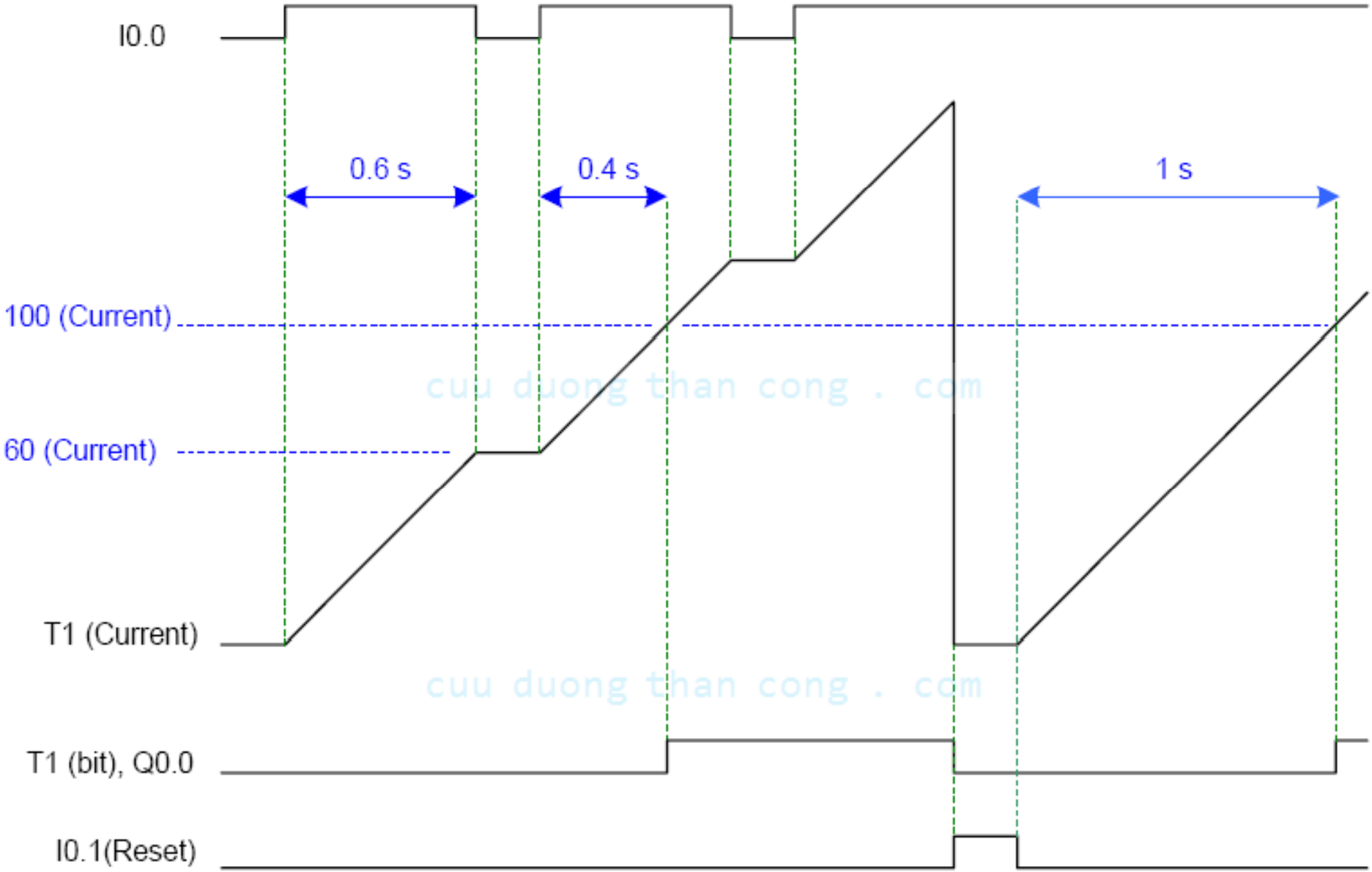


Timer ONR




```
Network 1       //10 ms TONR timer T1 times out at
                //PT=(100 x 10 ms=1s)
LD      I0.0
TONR    T1, +100

Network 2       //T1 bit is controlled by timer T1.
                //Turns Q0.0 on after the timer accumulates a total
                //of 1 second
LD      T1
=       Q0.0

Network 3       //TONR timers must be reset by a Reset instruction
                //with a T address.
                //Resets timer T1 (current and bit) when I0.1 is on.
LD      I0.1
R       T1, 1
```

Timer ONR



Counter

Đếm tăng: CTU

Đếm giảm: CTD

Đếm tăng, giảm: CTUD

Inputs/Outputs	Data Types	Operands
Cxx	WORD	Constant (C0 to C255)
CU, CD, LD, R	BOOL	I, Q, V, M, SM, S, T, C, L, Power Flow
PV	INT	IW, QW, VW, MW, SMW, SW, LW, T, C, AC, AIW, *VD, *LD, *AC, Constant

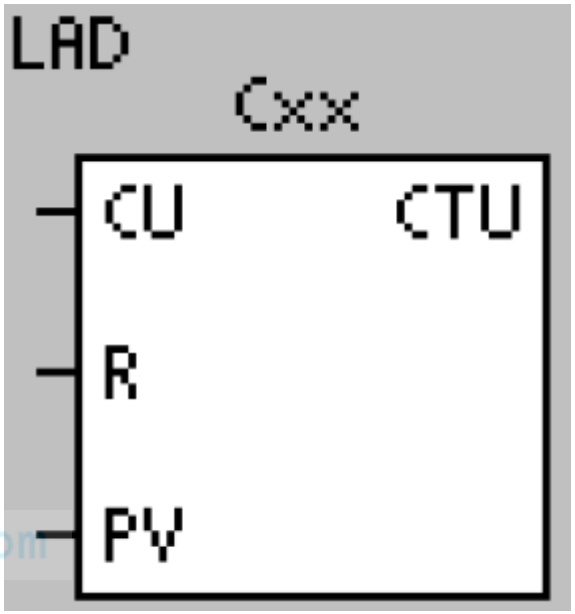
Counter CTU

Lệnh STL: CTU Cxx, PV

Khi R =1, Xóa bộ đếm, ngõ ra = 0

Khi CU chuyển từ 0→1 counter đếm lên

Khi giá trị counter =PV, ngõ ra =1



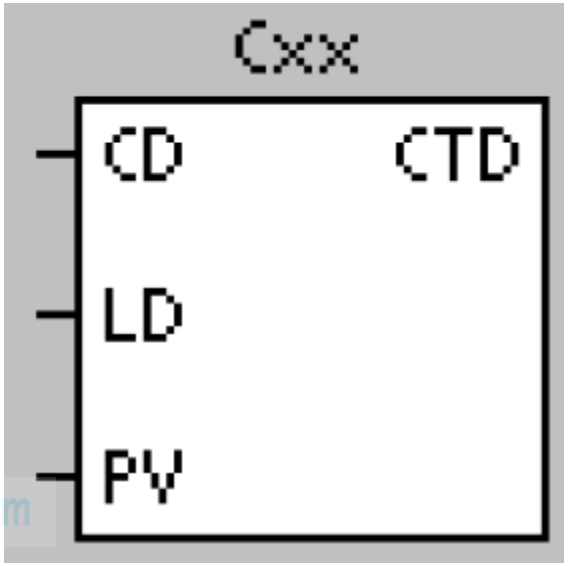
Counter CTD

Lệnh STL: CTD Cxx, PV

Khi LD =1, Load PV vào bộ đếm, ngõ ra =0

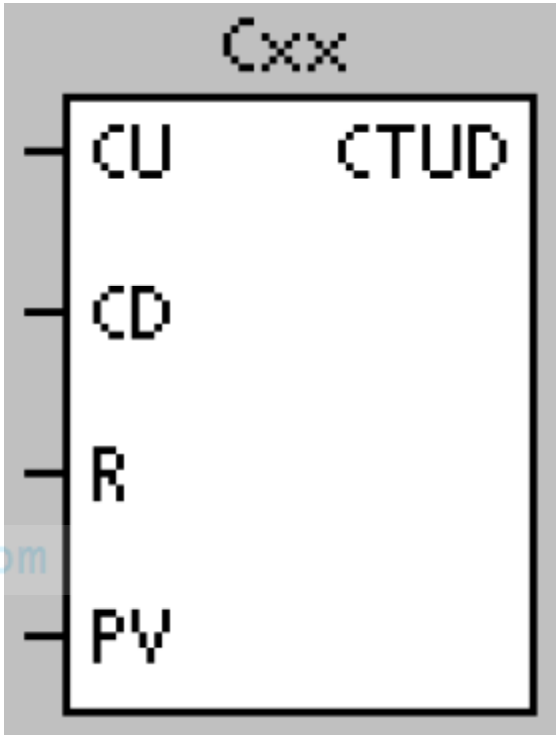
Khi CD chuyển từ 0→1 counter đếm xuống

Khi giá trị counter =0, ngõ ra =1



Counter CTUD

Lệnh STL: CTUD Cxx, PV

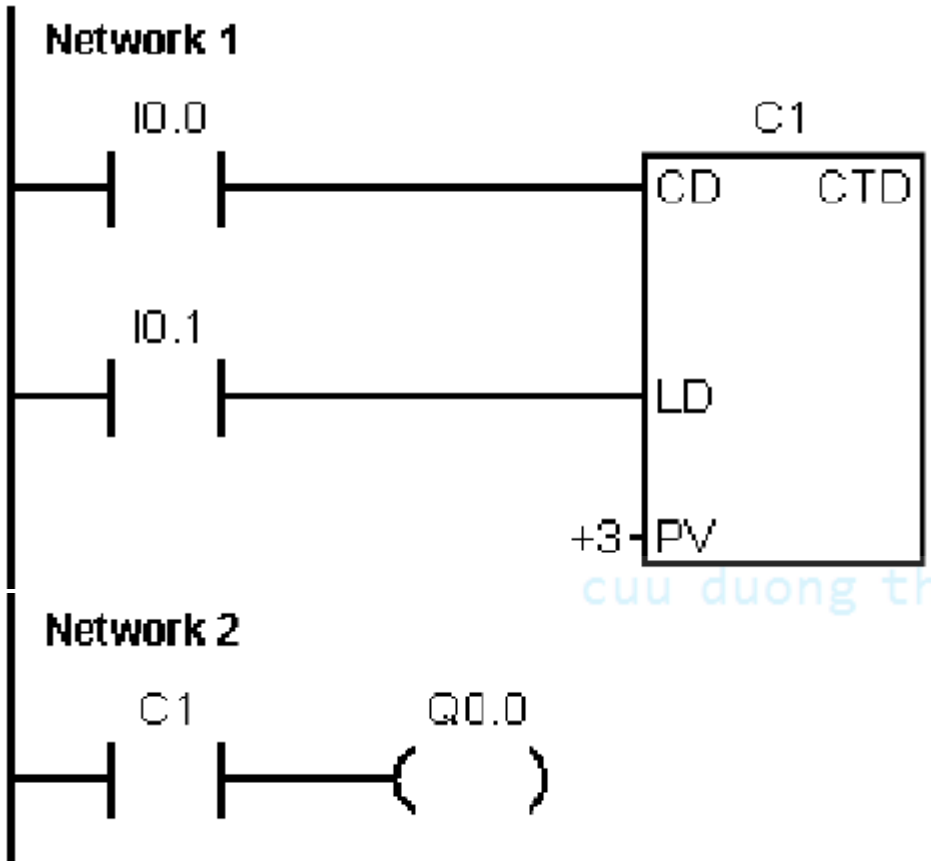


Khi R =1, Xóa bộ đếm, ngõ ra =0

Khi CU chuyển từ 0→1 counter đếm lên
Khi CD chuyển từ 0→1 counter đếm xuống

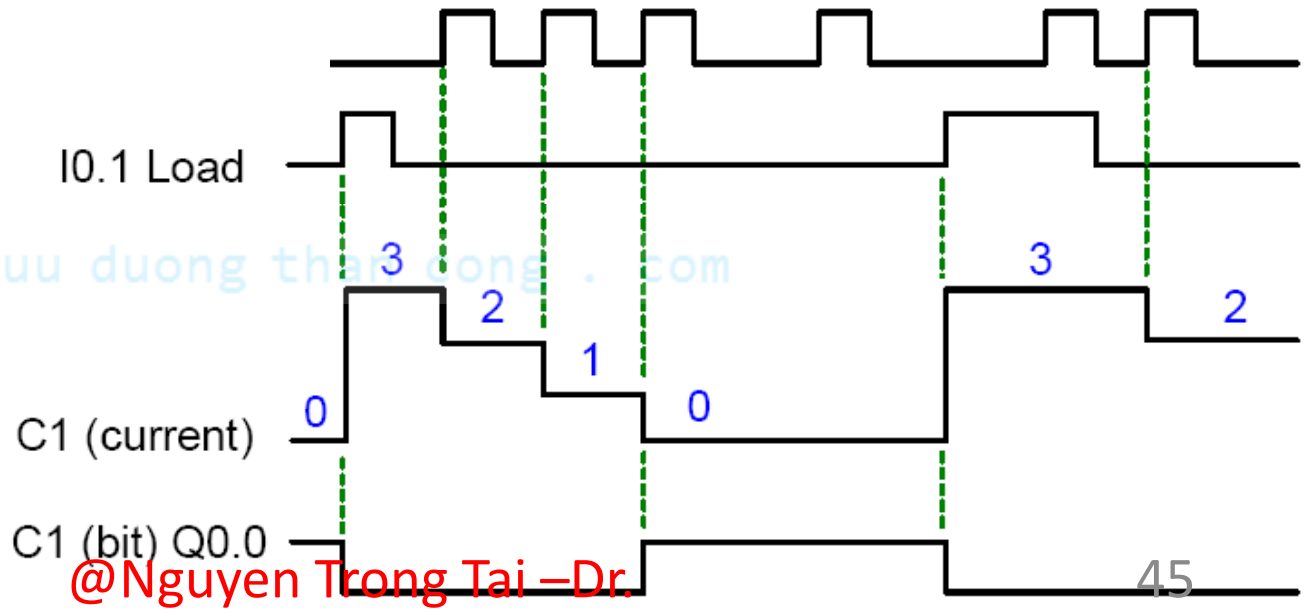
Khi giá trị counter >PV, ngõ ra =1

Counter Ví dụ

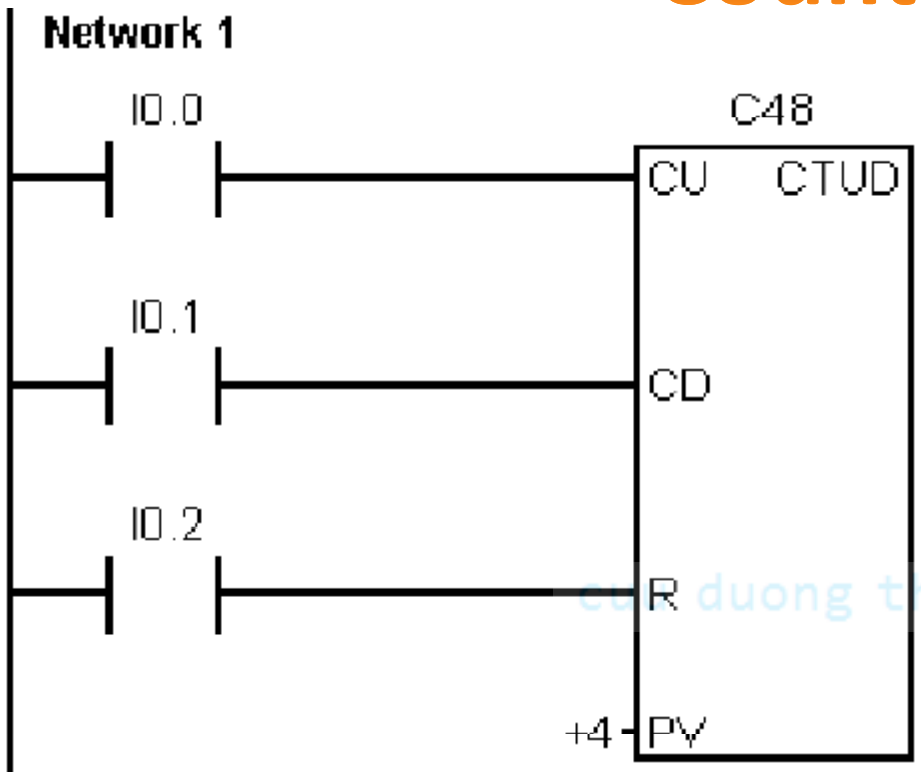


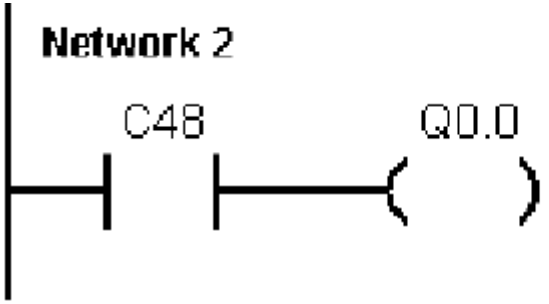
Network 1
LD I0.0
LD I0.1
CTD C1, +3

Network 2
LD C1
= Q0.0



Counter Ví dụ

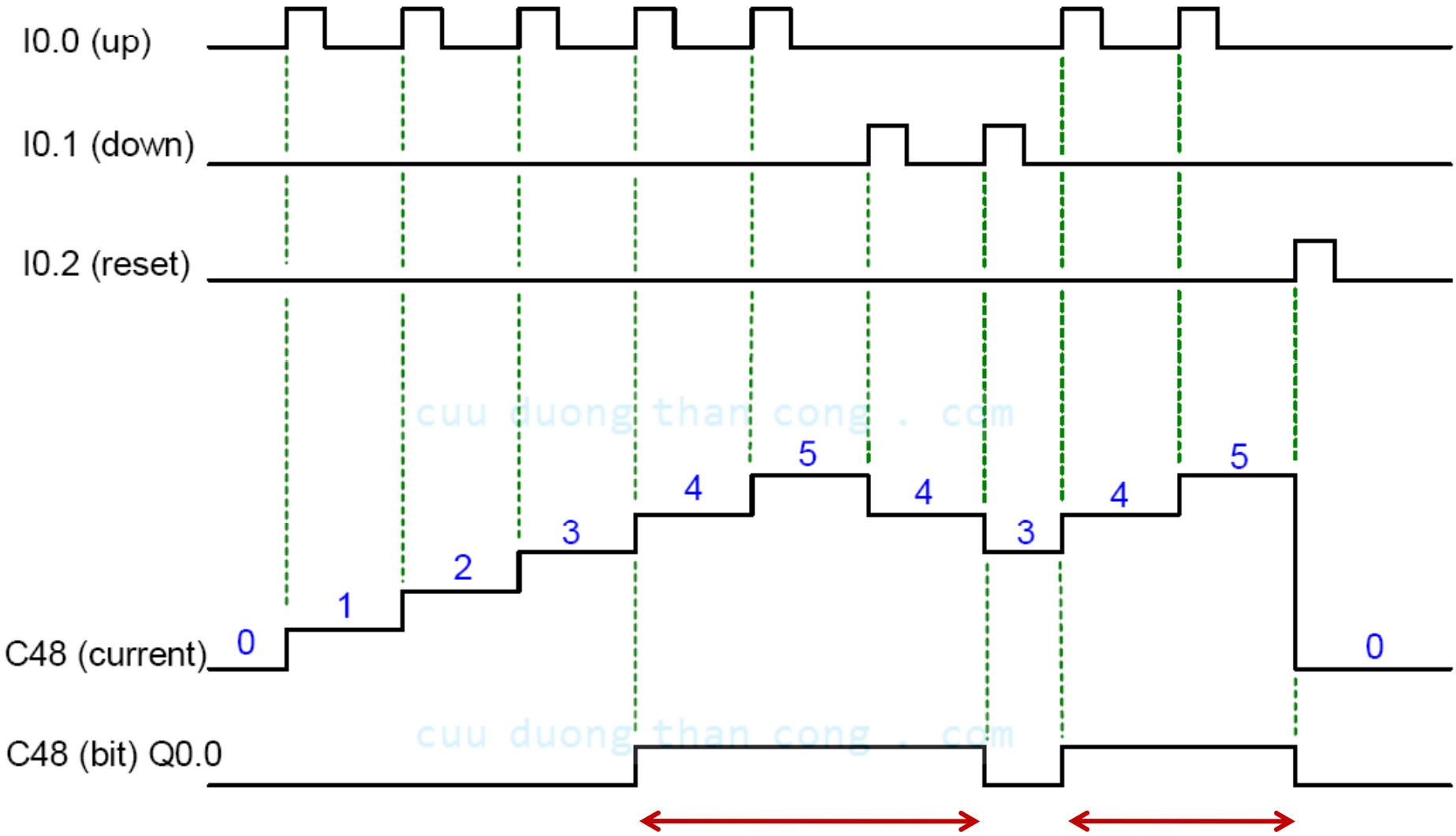




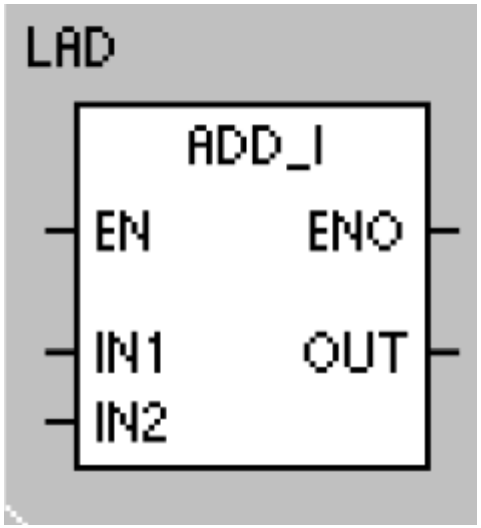
Network 1
LD I0.0
LD I0.1
LD I0.2
CTUD C48, +4

Network 2
LD C48
= Q0.0

Counter Ví dụ



Lệnh số học

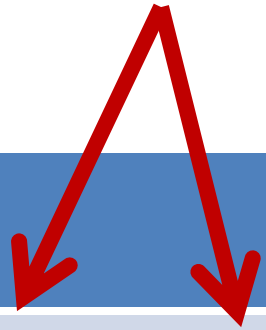


Int 16bit	Int 32bit	Real
ADD_I	ADD_DI	ADD_R
SUB_I	SUB_DI	SUB_R
MUL_I	MUL_DI	MUL_R
DIV_I	DIV_DI	DIV_R

Int 16bit	Int 32bit	Real	Result	STL
+I	+D	+R	**IN1 + OUT = OUT**	**+X IN1,OUT**
-I	-D	-R	**OUT – IN1 = OUT**	**-X IN1,OUT**
*I	*D	*R	**IN1*OUT = OUT**	***X IN1,OUT**
/I	/D	/R	**OUT/IN1 = OUT**	**/X IN1,OUT**

Lệnh số học

Ngõ vào, ngõ ra hợp lệ



Inputs/ Outputs	Data Types	Operands
IN1, IN2	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	INT	IW, QW, VW, MW, SMW, SW, LW, T, C, AC, *VD, *AC, *LD
	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Lệnh số học

Các trường hợp lỗi và cờ báo

SM	ENO	Lỗi/Trạng thái
SM1.0	**X**	*zero*
SM1.1	**0**	*Overflow, illegal value generated during operation, illegal input parameter found*
SM1.2	**X**	*Negative*
SM1.3	**0**	*Divide by zero*
0006	**0**	*Indirect address*

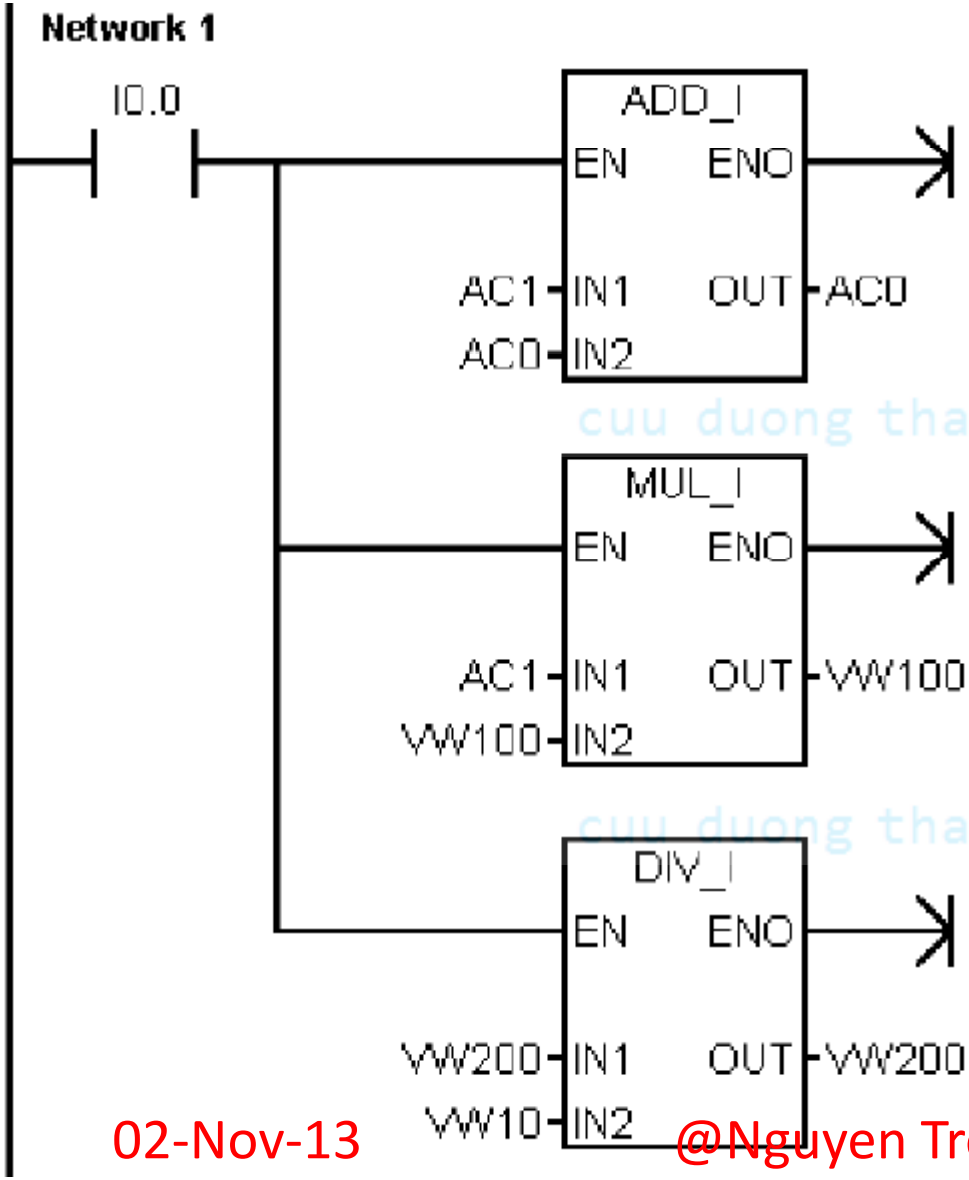
Note:

MUL_I, Nhân hai số 16bit, kết quả 16bit

DIV_I, Chia hai số 16bit, kết quả 16bit

Lệnh số học

Ví dụ




```
LD   I0.0
+I   AC1, AC0
*I   AC1, VW100
/I   VW10, VW200
```


Add

40	+	60	=	100
AC1		AC0		AC0

Multiply

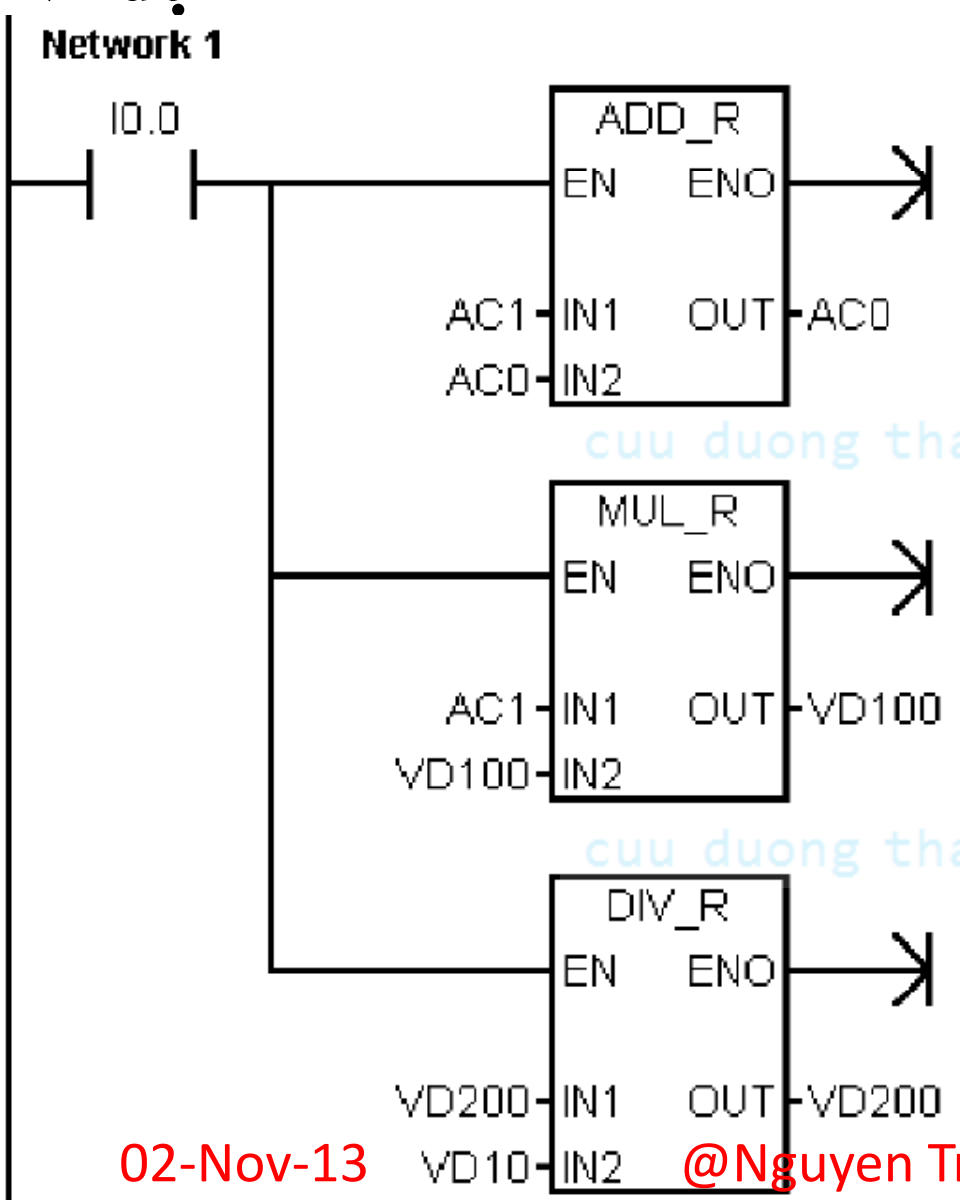
40	*	20	=	800
AC1		VW100		VW100

Divide

4000	/	40	=	100
VW200		VW10		VW200

Lệnh số học

Ví dụ




```
LD I0.0
+R AC1, AC0
*R AC1, VD100
/R VD10, VD200
```

Add

4000.0	+	6000.0	=	10000.0
AC1		AC0		AC0

Multiply

400.0	*	200.0	=	80000.0
AC1		VD100		VD100

Divide

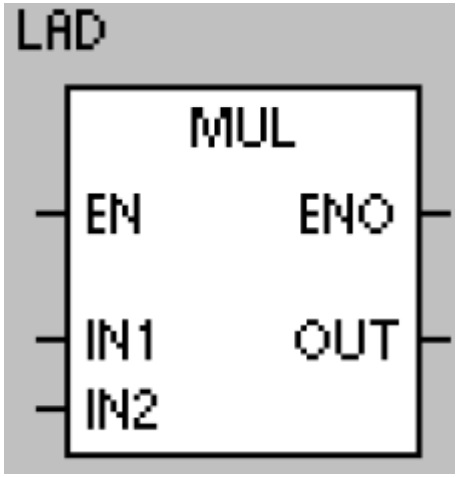
4000.0	/	41.0	=	97.5609
VD200		VD10		VD200

Lệnh số học

Lệnh đặc biệt MUL, DIV

MUL: Nhân hai số 16bit, kết quả 32bit

DIV: Chia hai số 16bit, kết quả trong word thấp, số dư trong word cao

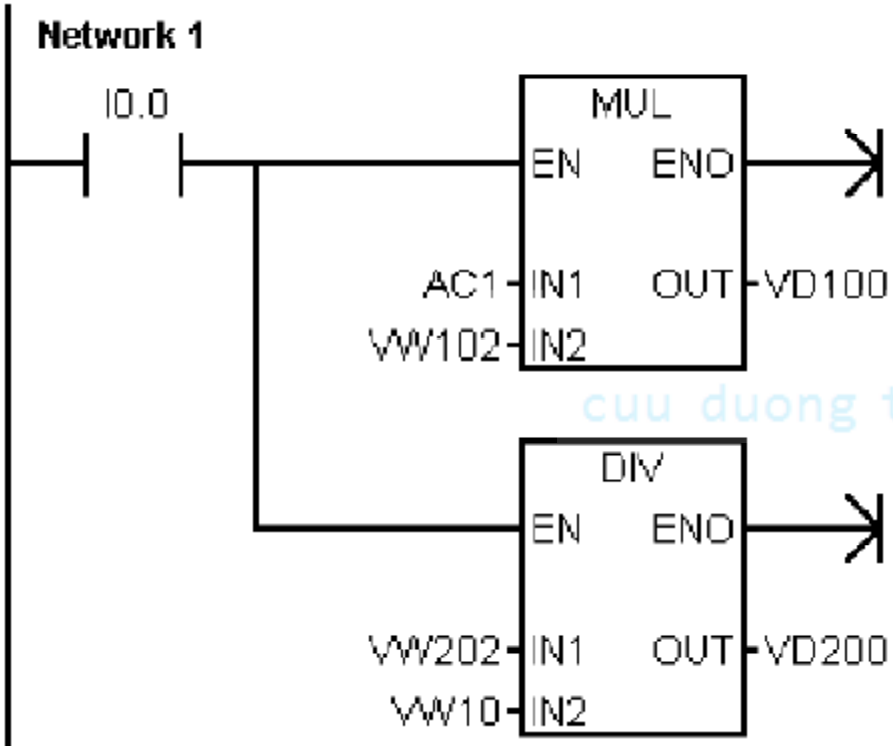


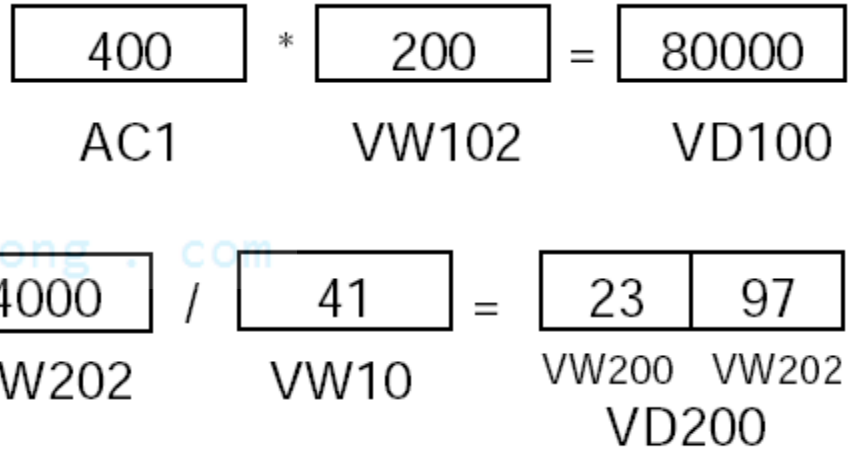
STL:
MUL IN1, OUT
DIV IN1,OUT

Inputs/ Outputs	Data Types	Operands
IN1, IN2	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
OUT	DINT	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Lệnh số học

Ví dụ

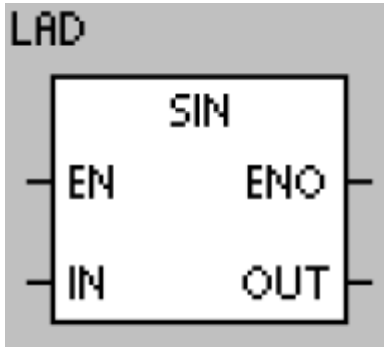





```
LD I0.0
MUL AC1, VD100
DIV VW10, VD200
```

Lệnh số học

Các hàm thông thường: SQRT, SIN, COS, TAN, EXP, LN



STL

SIN IN,OUT
COS IN,OUT
TAN IN,OUT

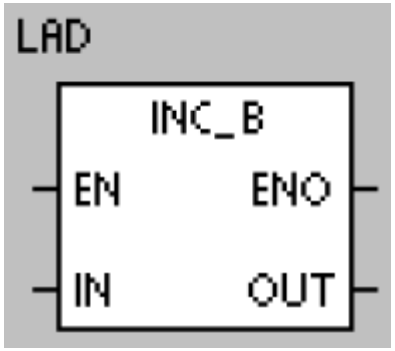
STL

EXP IN,OUT
SQRT IN,OUT

Inputs/ Outputs	Data Types	Operands
IN	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Lệnh số học

Các lệnh tăng/giảm: INC/DEC



8bit	16bit	32bit
INC_B	INC_W	INC_DW
DEC_B	DEC_W	DEC_DW

STL

INCB IN,OUT
INCW IN,OUT
INCD IN,OUT

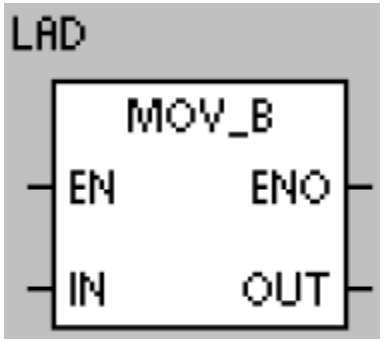
STL

DECB IN,OUT
DECW IN,OUT
DECD IN,OUT

Lệnh Duy Chuyển

Các lệnh duy chuyển: MOV

LAD



8bit	16bit	32bit	Real
MOV_B	MOV_W	MOV_DW	MOV_R

STL			
8bit	16bit	32bit	Real
MOVB IN,OUT	MOVW IN,OUT	MOVD IN,OUT	MOVR IN,OUT

Lệnh Duy Chuyển

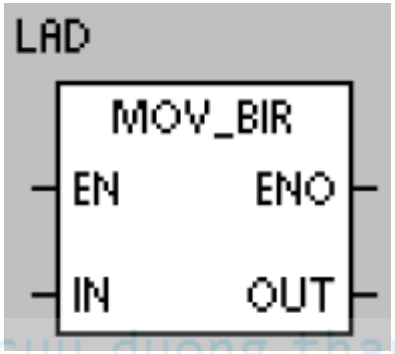
Các lệnh duy chuyển: MOV

Inputs/Outputs	Data Types	Operands
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *AC, *LD, Constant
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, HC, &VB, &IB, &QB, &MB, &SB, &T, &C, &SMB, &AIW, &AQW, AC, *VD, *LD, *AC, Constant,
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD,INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AQW, *VD, *LD, *AC
	DWORD, DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Lệnh Duy Chuyển

Các lệnh duy chuyển tức thời: MOV_BIR, MOV_BIW

(Read/Write)



STL

BIR IN,OUT

BIW IN,OUT

Lệnh BIR

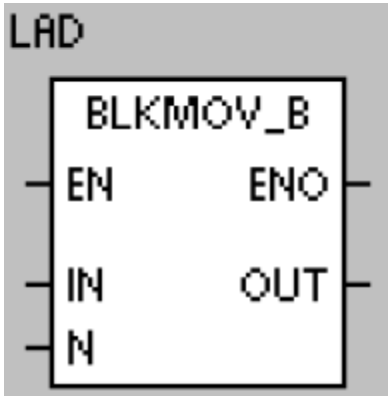
Inputs/ Outputs	Data Types	Operands
IN	BYTE	IB, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC

Lệnh BIW

Inputs/ Outputs	Data Types	Operands
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
OUT	BYTE	QB, *VD, *LD, *AC

Lệnh Duy Chuyển

Các lệnh duy chuyển khối: BLKMOV



8bit	16bit	32bit
BLKMOV_B	**BLKMOV_W**	**BLKMOV_D**

STL		
8bit	16bit	32bit
BMB IN,OUT,N	**BMW** IN,OUT,N	**BMD** IN,OUT,N

N=1:255, là số byte, word, Double word cần duy chuyển

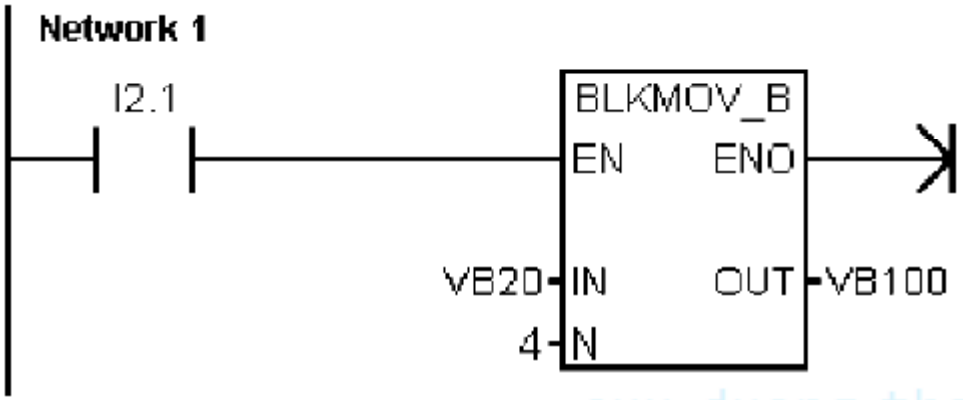
Lệnh Duy Chuyển

Các lệnh duy chuyển khốii: BLOCK MOV

Inputs/ Outputs	Data Types	Operands
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD,INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC
	DWORD, DINT	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant, *VD, *LD, *AC

Lệnh Duy Chuyển

Ví dụ: BLOCK MOV



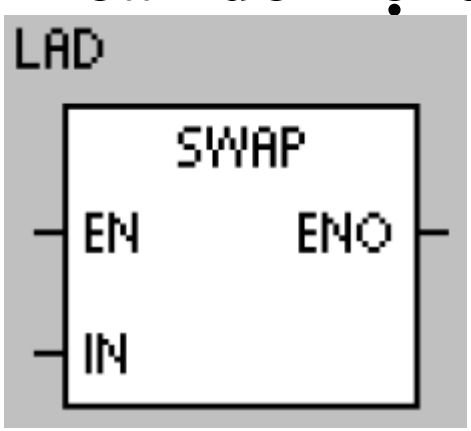

```
LD I2.1
BMB VB20, VB100, 4
```


	VB20	VB21	VB22	VB23
Array 1	30	31	32	33
	VB100	VB101	VB102	VB103
Array 2	30	31	32	33

Lệnh SWAP

Hoán đổi nội dung của 2byte trong 1 Word

LAD



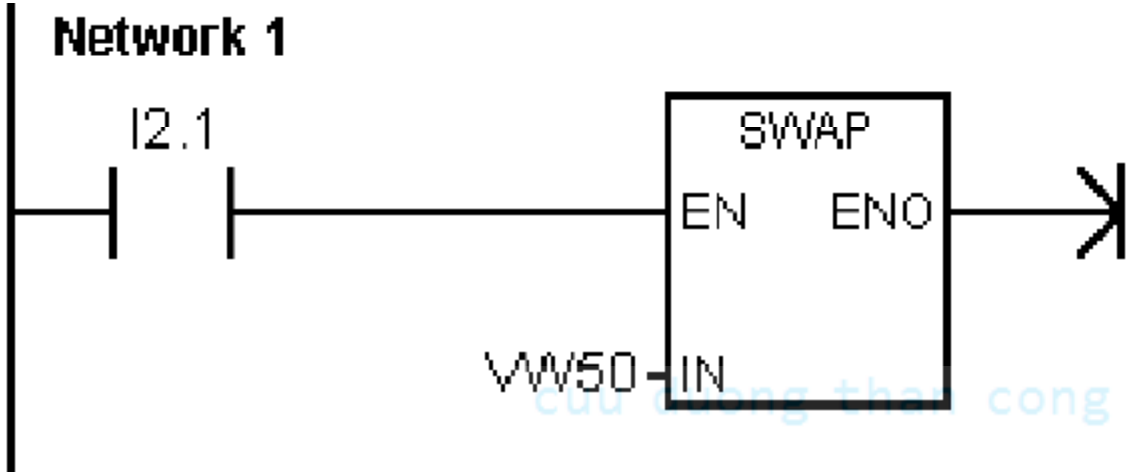
STL

SWAP IN

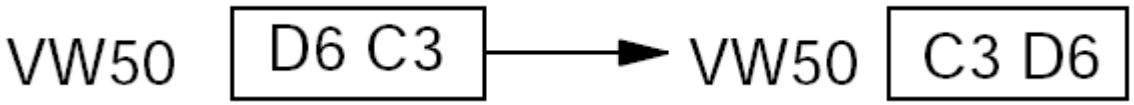
Inputs/ Outputs	Data Types	Operands
IN	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW,AC, *VD, *LD, *AC

Lệnh SWAP

Hoán đổi nội dung của 2byte trong 1 Word




```
LD I2.1
SWAP VW50
```



Lệnh Fill

Fill word value chứa ở địa chỉ IN vào vùng ô nhớ bắt đầu ở địa chỉ OUT

STL
FILL IN,OUT,N

N=1:255, là số byte, word, Double word cần duy chuyển

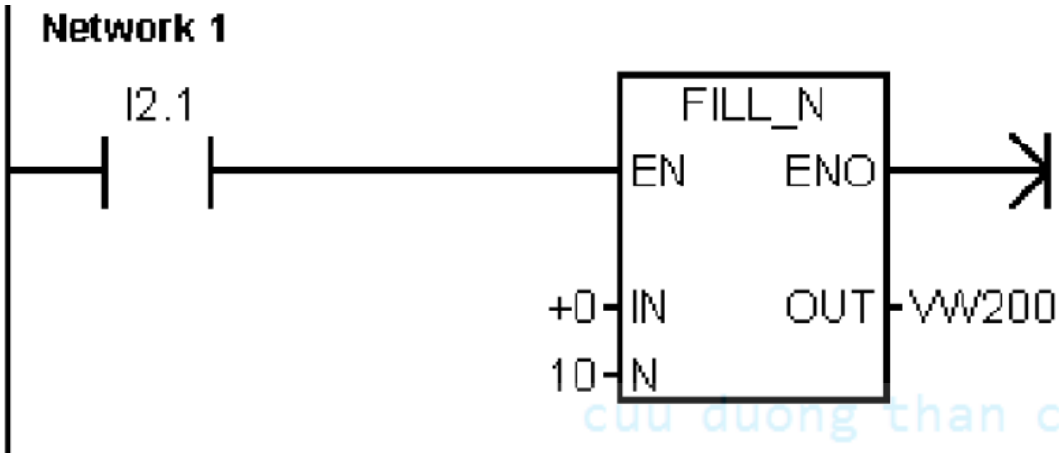
Lệnh Fill

Fill word value chứa ở địa chỉ IN vào vùng ô nhớ bắt đầu ở địa chỉ OUT

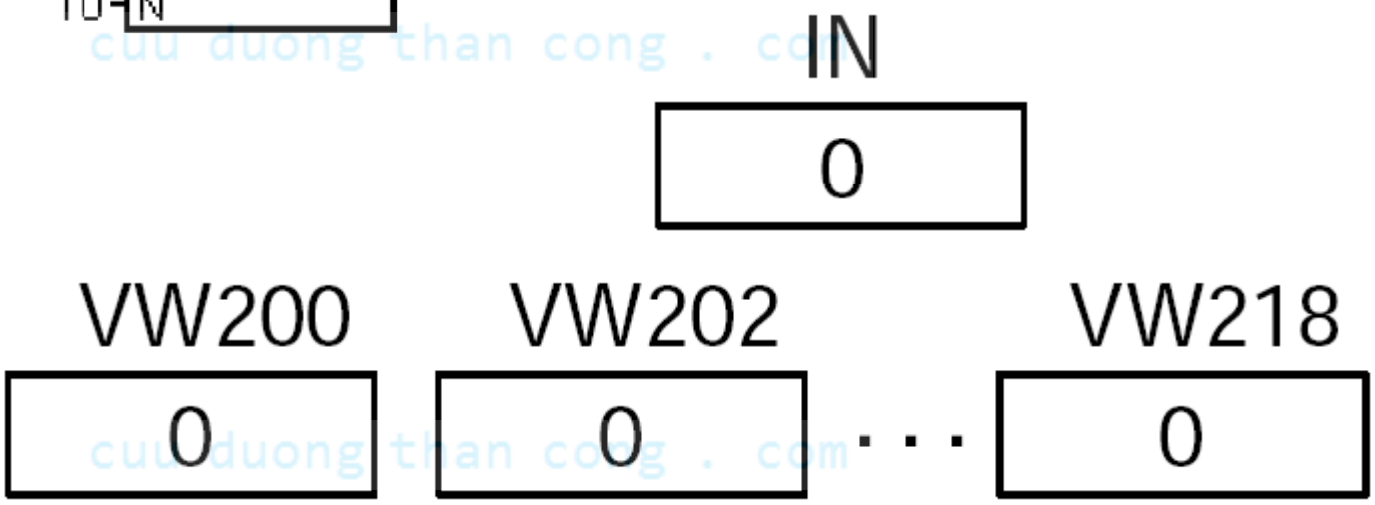
Inputs/ Outputs	Data Types	Operands
IN	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
OUT	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

Lệnh Fill

Ví dụ

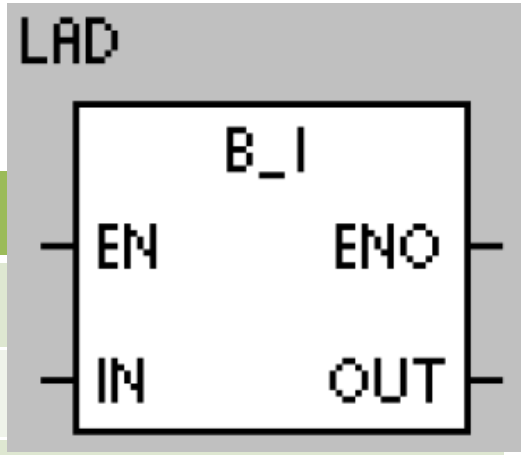



```
LD I2.1
FILL +0, VW200, 10
```



Lệnh chuyển đổi dữ liệu

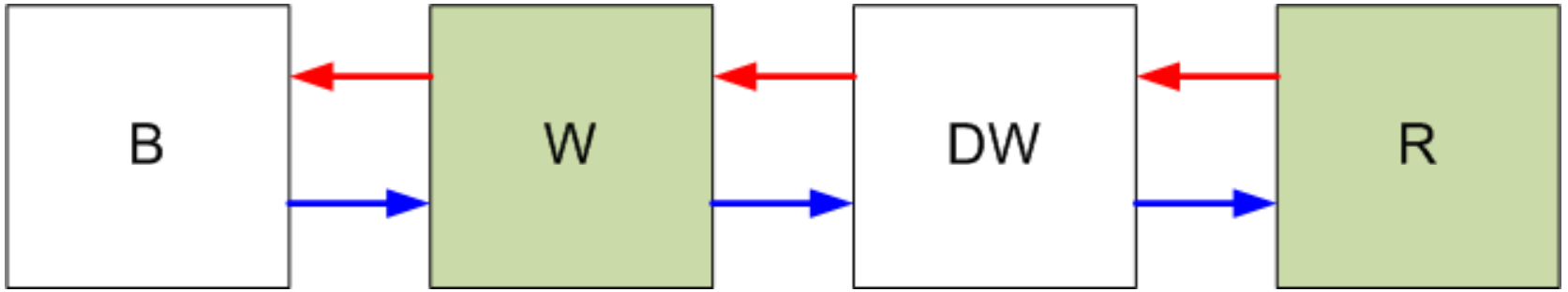
Các lệnh chuyển đổi



LAD	STL		
B_I	BTI IN,OUT		Byte to Integer
I_B	ITB IN,OUT		Integer to Byte
I_DI	ITD IN,OUT		Integer to Double Integer
DI_I	DTI IN,OUT		Double Integer to Integer
DI_R	DTR IN,OUT		Double Integer to Real
BCD_I	BCDI OUT		BCD to Integer
I_BCD	IBCD OUT		Integer to BCD
ROUND	ROUND IN,OUT		Real to Double Integer with Round
TRUNC	TRUNC IN,OUT		Real to Double Integer with floor
SEG	SEG IN,OUT		For display 7 segment led

Lệnh chuyển đổi dữ liệu

Chuyển kiểu dữ liệu



Inputs/Outputs	Data Types	Operands
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD, INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AIW, AC, *VD, *LD, *AC, Constant
	DINT	ID, QD, VD, MD, SMD, SD, LD, HC, AC, *VD, *LD, *AC, Constant
	REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC
	WORD,INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
	DINT, REAL	ID, QD, VD, MD, SMD, SD, LD, AC, *VD, *LD, *AC

Lệnh chuyển đổi dữ liệu

Một số lệnh cần chú ý

Double Integer to Integer (DTI)

Nếu IN>Interger ➔ lệnh không được thực hiện, SM1.1 =1

Integer to Byte (ITB)

Nếu IN>255 (Byte) ➔ lệnh không được thực hiện, SM1.1 =1

Để chuyển từ số nguyên sang số thực, sử dụng 2 lệnh ITD và lệnh DTR

Lệnh chuyển đổi dữ liệu

Một số lệnh cần chú ý

BCD to Integer (BCDI): Chuyển mã nhị phân BCD trong IN thành số số nguyên

Nếu IN ≠ 0→9999 BCD ➔ lệnh không được thực hiện, SM1.6 =1 (Invalid BCD)

Integer to BCD (IBCD): Chuyển giá trị integer trong IN sang mã BCD

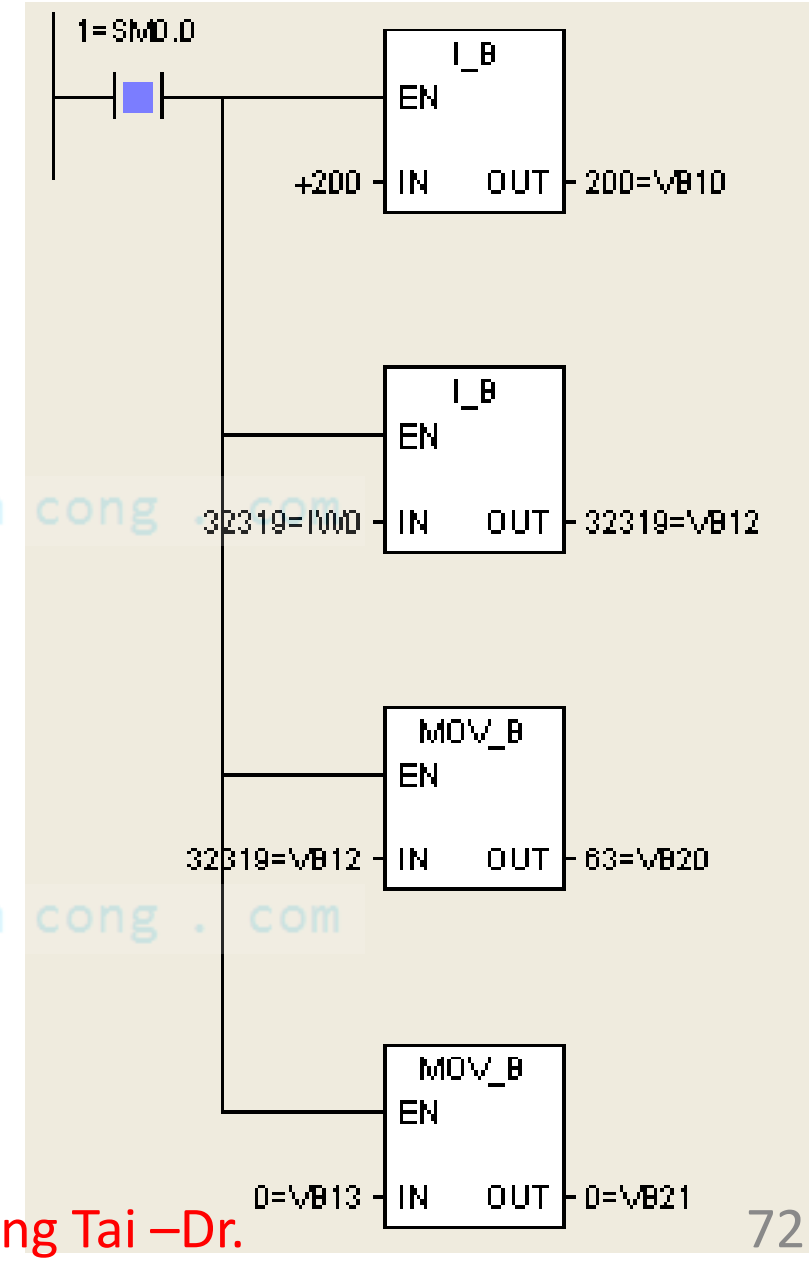
Nếu IN ≠ 0→9999 integer➔ lệnh không được thực hiện, SM1.6 =1 (Invalid BCD)

Để chuyển từ số nguyên sang số thực, sử dụng 2 lệnh ITD và lệnh DTR

Lệnh chuyển đổi dữ liệu

Chuyển kiểu dữ liệu

Khi chuyển dữ liệu từ định dạng lớn về định dạng bé, chỉ vùng dữ liệu bé được nhận, vùng cao bị mất,
nếu bị mất thì SM1.1 =1

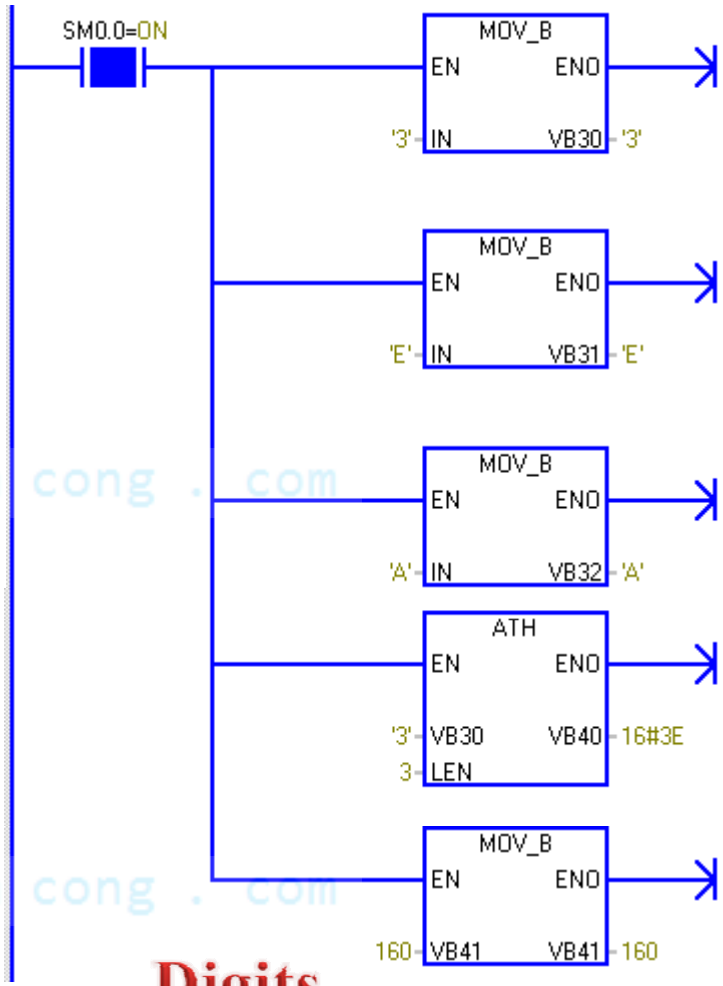


Lệnh chuyển đổi dữ liệu

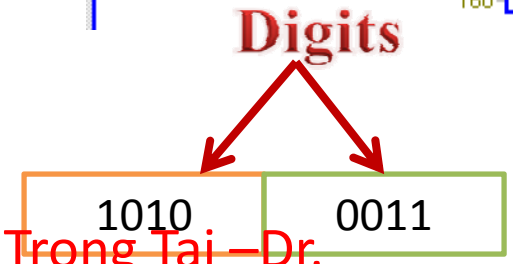
Lệnh ATH

Chuyển mã ASCII (30H..39H ='0'..'9');(41H..46H='A'..'F') sang dạng HEX
Input: mã ASCII của ký tự dạng decimal

Output: dạng HEX của mã ASCII của ký tự
LEN: số digit cần chuyển 0-255



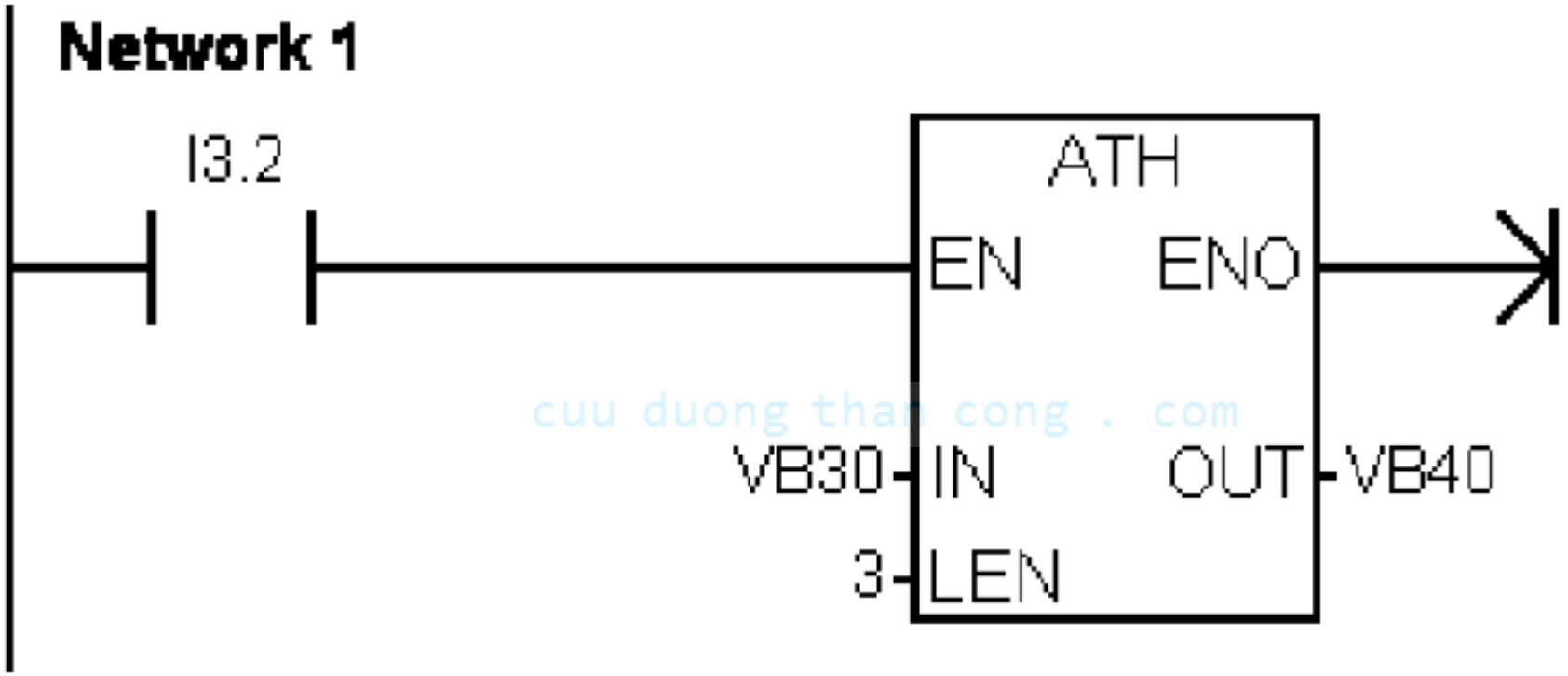
Báo lỗi bằng SM1.7

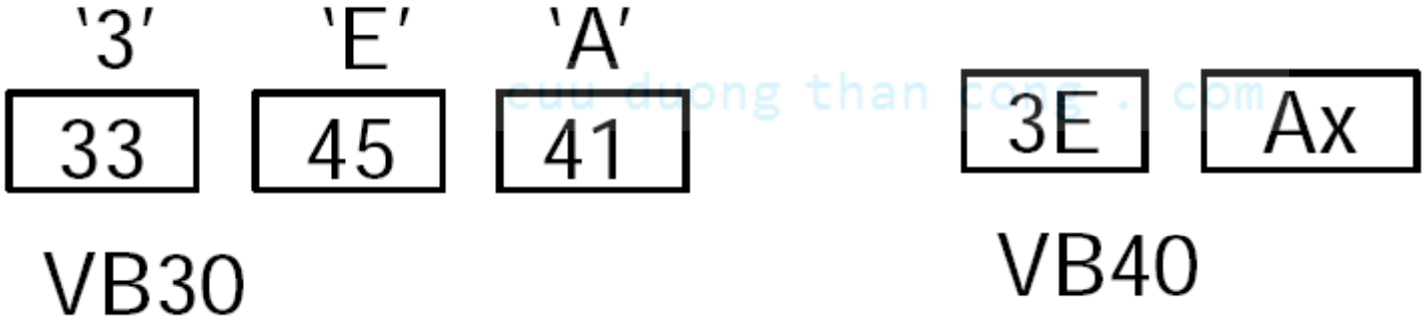


Chuyển mã ASCII (số 48→63) sang ký tự.
Tầm ngõ ra 0→F

Lệnh chuyển đổi dữ liệu

Lệnh ATH

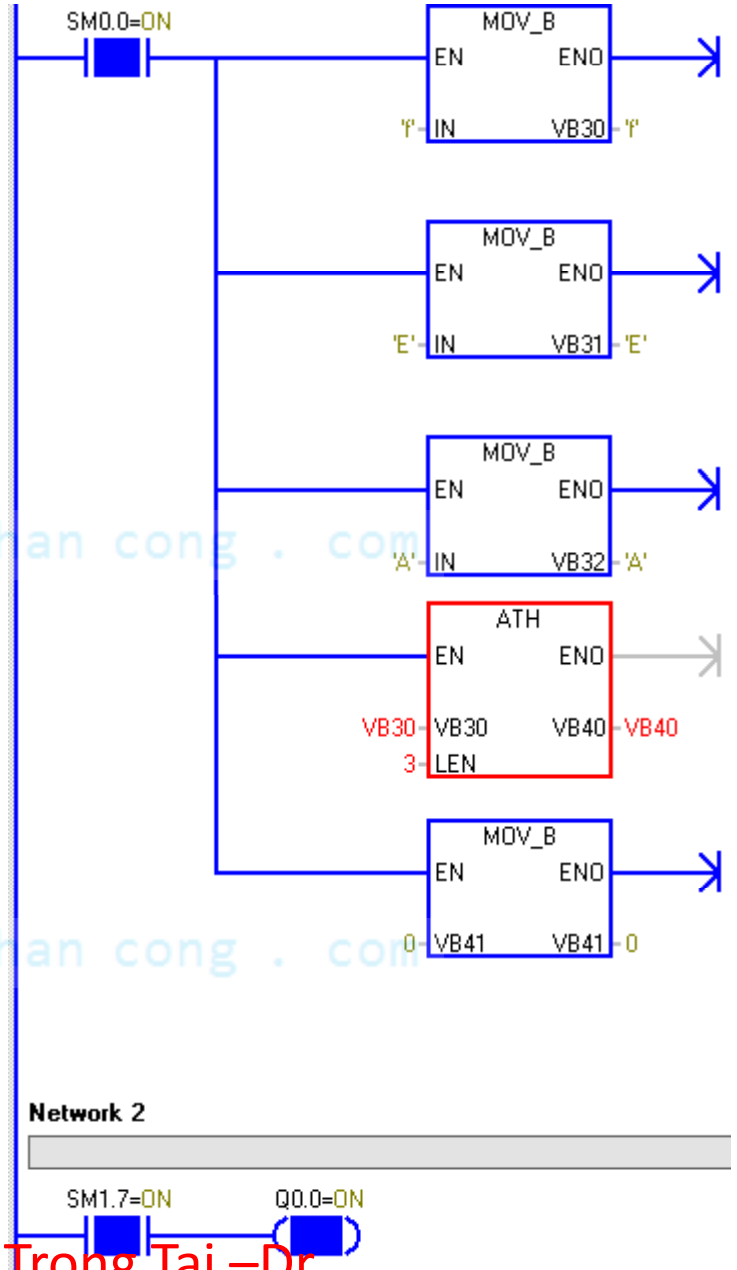




Lệnh chuyển đổi dữ liệu

Lệnh ATH

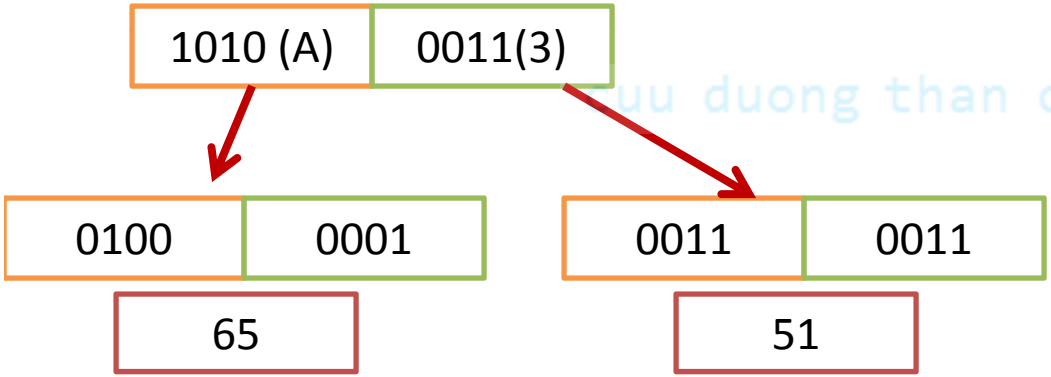
Trường hợp không hợp lệ



Lệnh chuyển đổi dữ liệu

Lệnh HTA

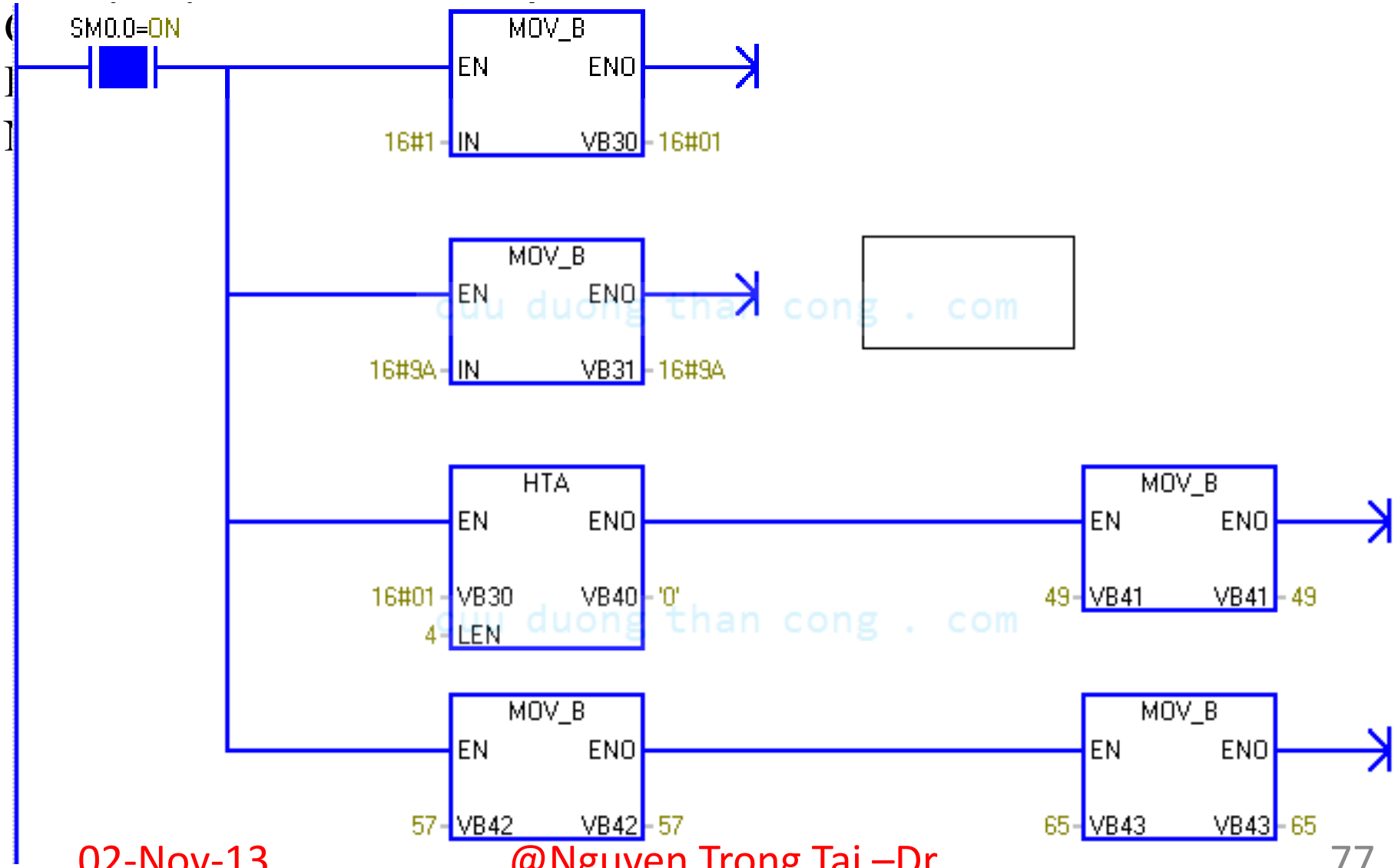
Chuyển đổi một digt 0..F chiều dài
LEN ra mã ASCII.
N là số digits mã hex cần chuyển



Lệnh HTA không có lỗi tràn

Lệnh chuyển đổi dữ liệu

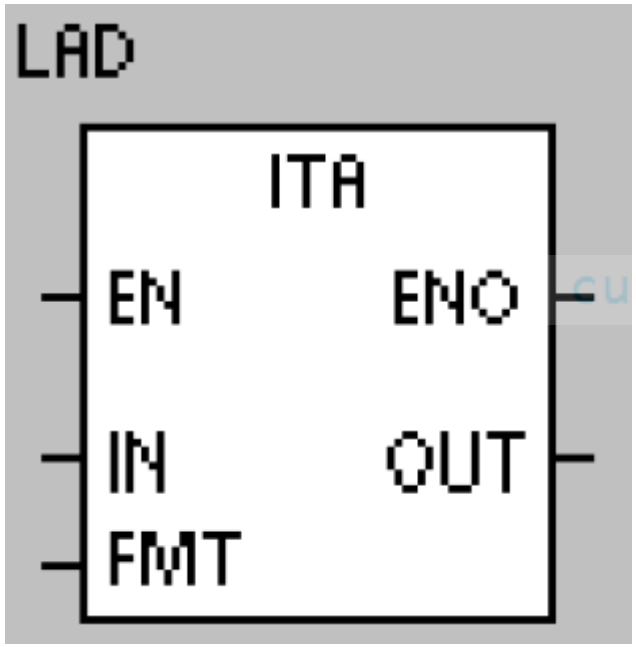
Lệnh HTA



Lệnh chuyển đổi dữ liệu

Lệnh ITA: chuyển đổi số nguyên trong Word IN sang mảng ký tự

ASCII theo format FMT và ngõ ra là 8byte



FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = comma (1) or decimal point (0)
nnn = digits to right of decimal point

Lỗi: ENO = 0 khi format sai hoặc nnn>5

Lệnh chuyển đổi dữ liệu

Lệnh ITA:

Số dương được biểu diễn không dấu

Số âm được biểu diễn dấu trừ (-)

Số 0 được thêm vào cho dấu (.)(,) có ý nghĩa - trong trường hợp thiếu số

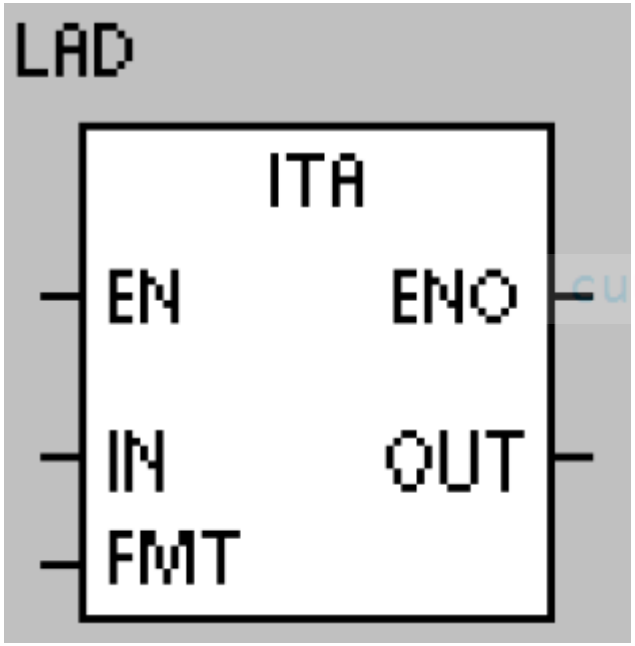
Giá trị được điền từ phải sang trái (right-justified)

Integer	Out	Out+1	Out+2	Out+3	Out+4	Out+5	Out+6	Out+7
12				0	.	0	1	2
-123			-	0	.	1	2	3
1234				1	.	2	3	4
-12345		-	1	2	.	3	4	5
FMT	0	0	0	0	0	0	1	1

Lệnh chuyển đổi dữ liệu

Lệnh DTA: chuyển đổi số nguyên trong Double Word IN sang mảng

ký tự ASCII theo format FMT và ngõ ra là 12byte



FMT

MSB							LSB
7	6	5	4	3	2	1	0
0	0	0	0	c	n	n	n

c = comma (1) or decimal point (0)
nnn = digits to right of decimal point

Lỗi: ENO = 0 khi format sai hoặc nnn>5

Lệnh chuyển đổi dữ liệu

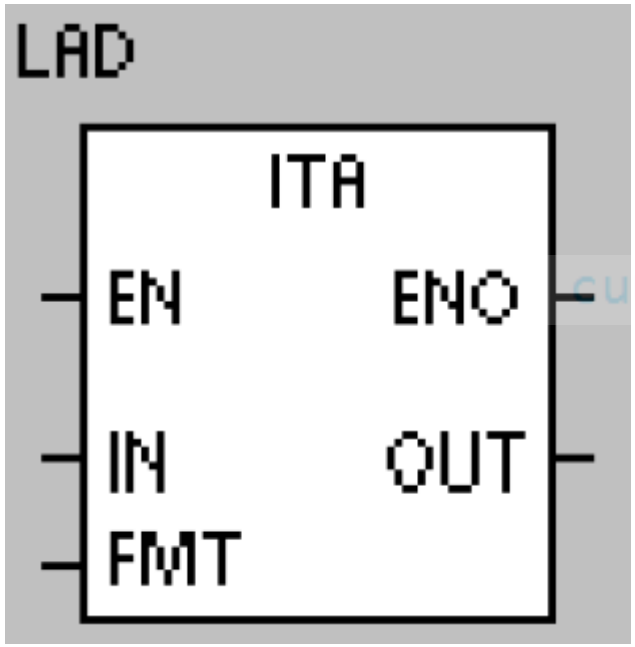
Lệnh DTA:

- Số dương được biểu diễn không dấu
- Số âm được biểu diễn dấu trừ (-)
- Số 0 được thêm vào cho dấu (.)(,) có ý nghĩa - trong trường hợp thiếu số
- Giá trị được điền từ phải sang trái (right-justified)

Integer	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7	Out +8	Out +9	Out +10	Out +11
12								0	.	0	1	2
-123							-	0	.	1	2	3
1234						-	1	2	.	3	4	5
-12345					1	2	3	4	.	5	6	7
FMT	0	0	0	0	0	0	1	1				

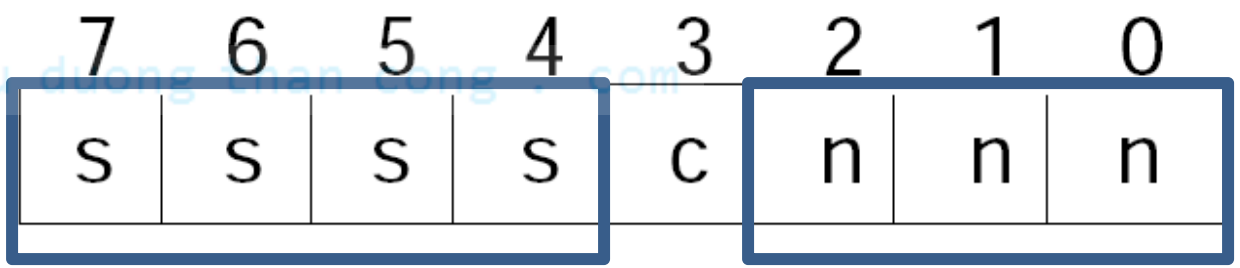
Lệnh chuyển đổi dữ liệu

Lệnh RTA: chuyển đổi số thực trong Double Word IN sang mảng ký tự ASCII theo format FMT và ngõ ra là được quy định trong FMT



FMT

Ssss = 3:15 byte



ssss = size of output buffer
c = comma (1) or decimal point (0)
nnn = digits to right of decimal point

Lỗi: ENO = 0 khi format sai hoặc nnn>5; ssss<3; ssss< số ký tự ra

Lệnh chuyển đổi dữ liệu

Lệnh RTA:

- Số dương được biểu diễn không dấu
- Số âm được biểu diễn dấu trừ (-)
- Số 0 được thêm vào cho dấu (.)(,) có ý nghĩa - trong trường hợp thiếu số
- Giá trị được điền từ phải sang trái (right-justified)
- Giá trị của buffer tối thiểu phải nhiều hơn 3 byte sovoi71 số digits nằm bên phải dấu (.),(,)
- Giá trị sau dấu (.),(,) được làm tròn theo định dạng từ format

Lệnh chuyển đổi dữ liệu

Lệnh RTA:

Số dương được biểu diễn không dấu

Số âm được biểu diễn dấu trừ (-)

Số 0 được thêm vào cho dấu (.)(,) có ý nghĩa - trong trường hợp thiếu số

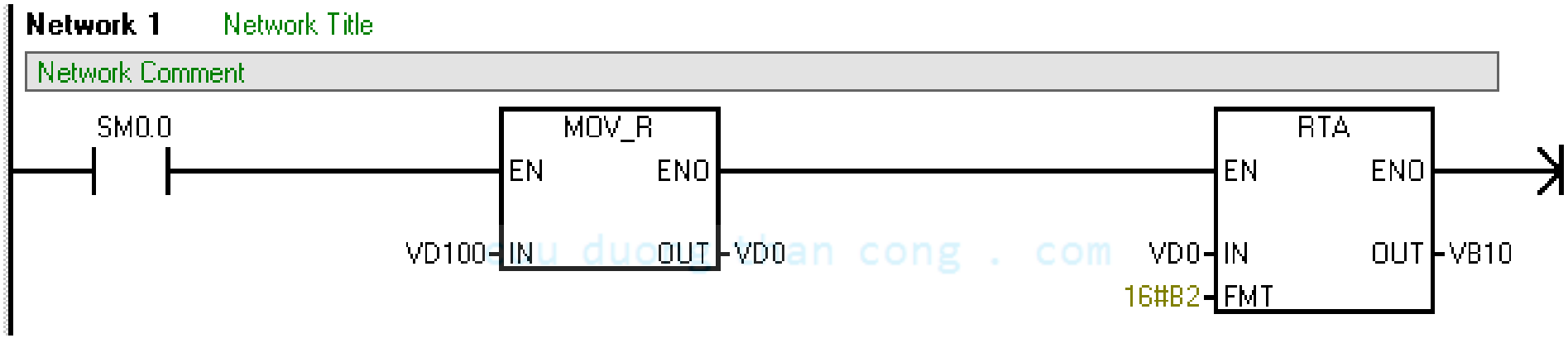
Giá trị được điền từ phải sang trái (right-justified)

Giá trị của buffer tối thiểu phải nhiều hơn 3 byte sovoi71 số digits nằm bên phải dấu (.) (,)

Integer	Out	Out +1	Out +2	Out +3	Out +4	Out +5	Out +6	Out +7	Out +8	Out +9	Out +10
1234.5					1	2	3	4	.	5	0
-0.0005								0	.	0	0
-3.67526							-	3	.	6	8
1.955								1	.	9	6
FMT	1	0	1	1	0	0	1	0			

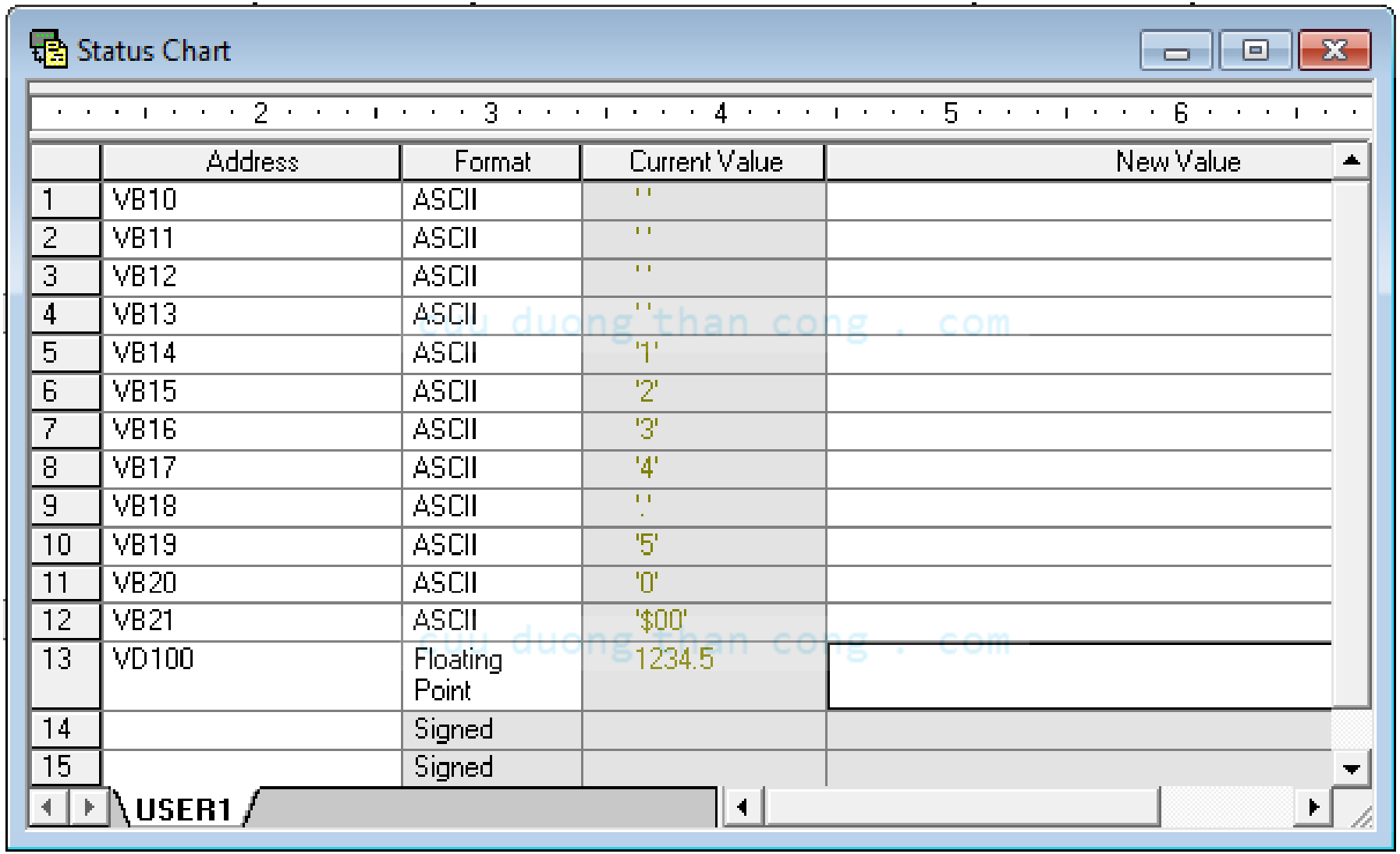
Lệnh chuyển đổi dữ liệu

Lệnh RTA:



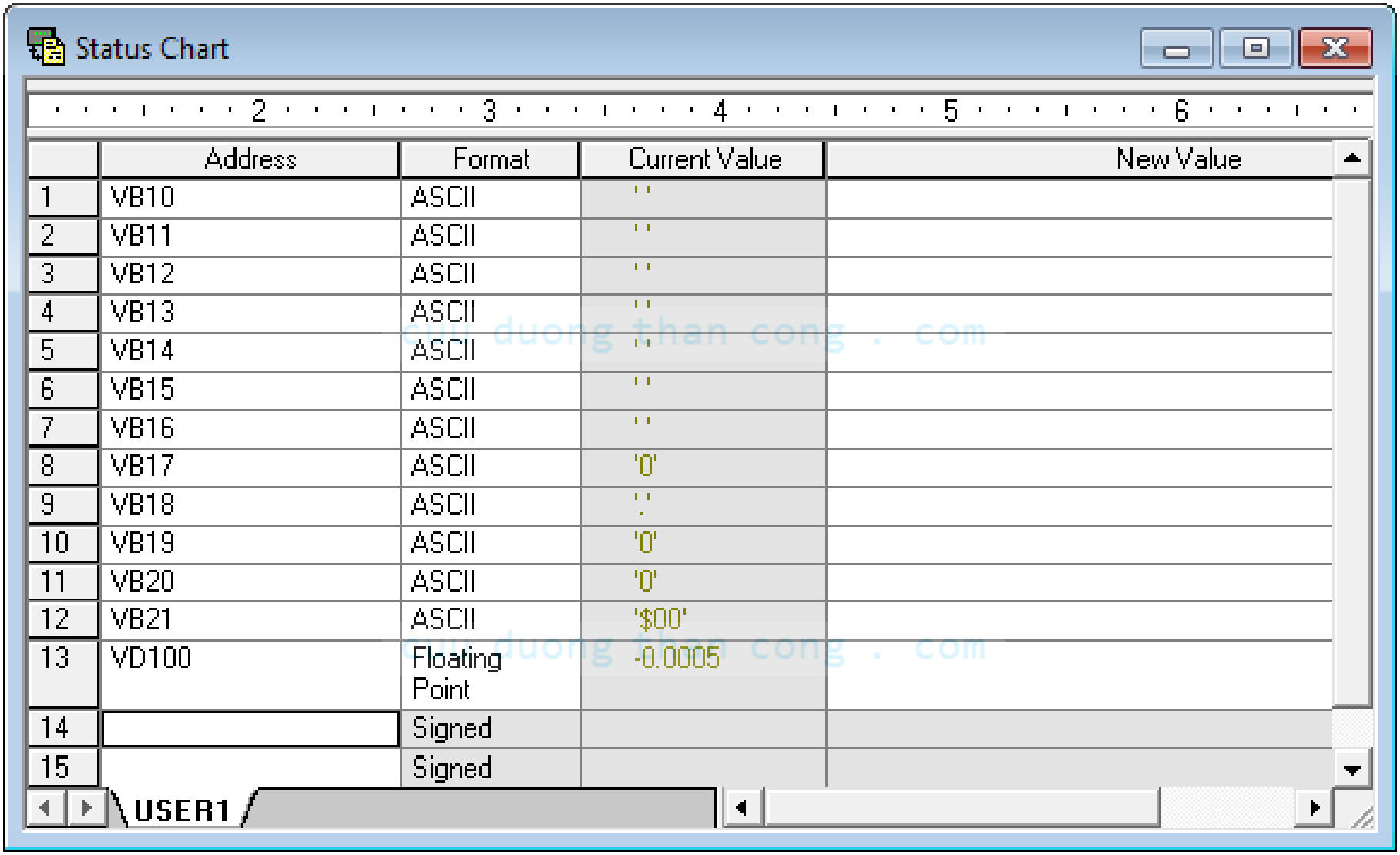
Lệnh chuyển đổi dữ liệu

Lệnh RTA:



Lệnh chuyển đổi dữ liệu

Lệnh RTA:



Lệnh chuyển đổi dữ liệu

Lệnh ITS (Integer to String), DTS (Double to String), RTS (Real to String)

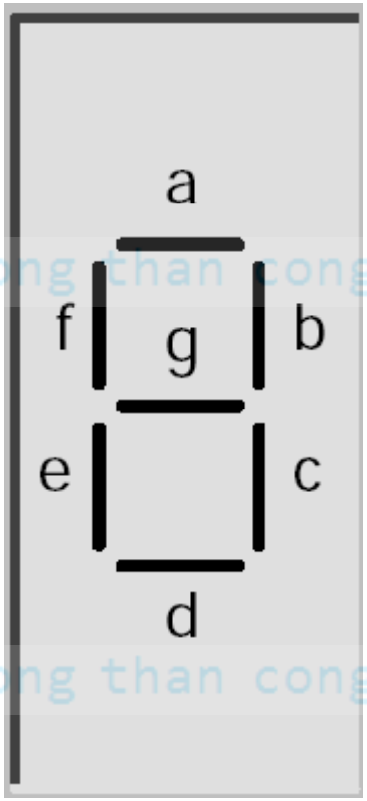
Lệnh STI (Substring to Integer), STD (Substring to Double), STR (Substring to Real)

Tự đọc trong Manual

Lệnh chuyển đổi dữ liệu

Lệnh SEG (Segment) chuyển mã để hiển thị led 7đoạn

Chuyển ký tự (byte) IN sang dạng hiển thị led 7 đoạn chứa trong byte OUT



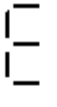
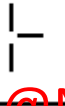
Lệnh chuyển đổi dữ liệu

Lệnh SEG (Segment) chuyển mã để hiển thị led 7đoạn

(IN) LSD	Segment Display	(OUT) - g f e d c b a
0		0 0 1 1 1 1 1 1
1		0 0 0 0 0 1 1 0
2		0 1 0 1 1 0 1 1
3		0 1 0 0 1 1 1 1
4		0 1 1 0 0 1 1 0
5		0 1 1 0 1 1 0 1
6		0 1 1 1 1 1 0 1
7		0 0 0 0 0 1 1 1

Lệnh chuyển đổi dữ liệu

Lệnh SEG (Segment) chuyển mã để hiển thị led 7đoạn

(IN) LSD	Segment Display	(OUT) - g f e d c b a
8		0 1 1 1 1 1 1 1
9		0 1 1 0 0 1 1 1
A		0 1 1 1 0 1 1 1
B		0 1 1 1 1 1 0 0
C		0 0 1 1 1 0 0 1
D		0 1 0 1 1 1 1 0
E		0 1 1 1 1 0 0 1
F		0 1 1 1 0 0 0 1

Lệnh chuyển đổi dữ liệu

Lệnh SEG (Segment) chuyển mã để hiển thị led 7đoạn

```
NETWORK1
LD    I0.0
LPS
AN    T38
TON   T37, 200
LRD
A     T37
TON   T38, 100
LRD
AN    T37
=     Q0.0
LPP
A     T37
=     Q0.1
```

```
NETWORK 2
LD    Q0.0
LPS
MOVW  +200, MW0
AENO
-I    T37, MW0
AENO
/I    +10, MW0
LRD
MOVW  MW0, MW2
IBCD  MW2
LRD
SEG   MB3, QB2
LPP
MOVB  MB3, MB4
AENO
SRB   MB4, 4
AENO
SEG   MB4, QB3
```

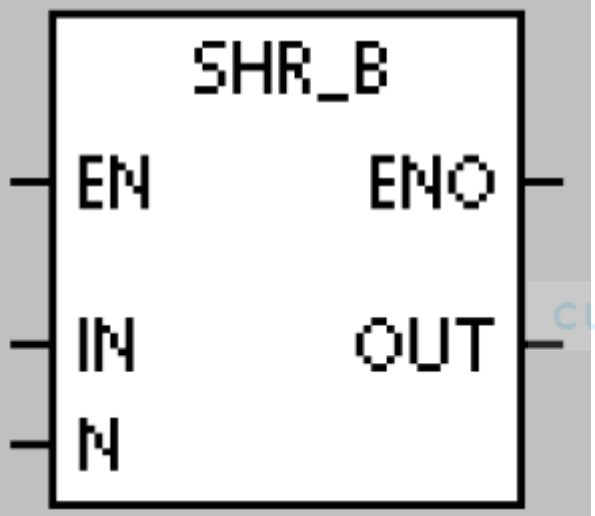
```
NETWORK 3
LD    Q0.1
LPS
MOVW  +100, MW10
AENO
-I    T38, MW10
AENO
/I    +10, MW10
LRD
MOVW  MW10, MW12
IBCD  MW12
LRD
SEG   MB13, QB2
LPP
MOVB  MB13, MB14
AENO
SRB   MB14, 4
AENO
SEG   MB14, QB3
```

Chương trình đèn giao thông hiển thị giây lùi, đèn xanh Q0.0, đèn đỏ Q0.1, đèn 7 đoạn QB3, QB2

Lệnh Dịch và Xoay

Lệnh dịch trái, dịch phải;

LAD



	8bit	16bit	32bit
LỆNH	SHR_B	SHR_W	SHR_DW
	SHL_B	SHL_W	SHL_DW
N	0<N<=8	0<N<=16	0<N<=32

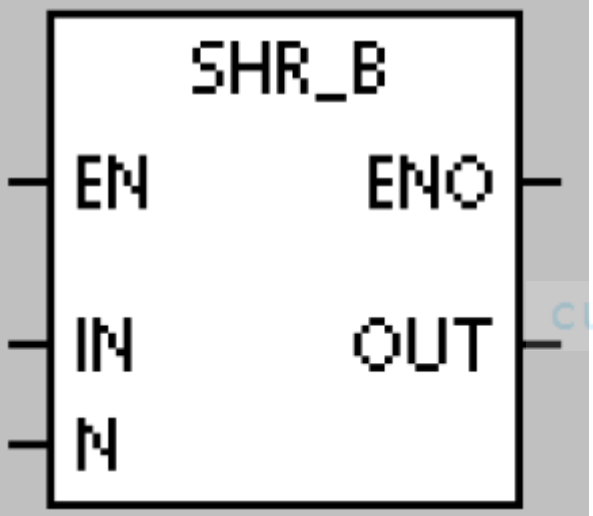
STL		
8bit	16bit	32bit
SRB OUT,N	SRW OUT,N	SRD OUT,N
SLB OUT,N	SLW OUT,N	SLD OUT,N

BIT nhớ bị tác động	
SM1.0	Được set lên 1 khi giá trị được dịch là 0
SM1.1	Chứa bit cuối cùng được dịch

Lệnh Dời và Xoay

Lệnh xoay trái, xoay phải;

LAD



	8bit	16bit	32bit
LỆNH	ROR_B	ROR_W	ROR_DW
	ROL_B	ROL_W	ROL_DW
N	0<N<=8	0<N<=16	0<N<=32

STL		
8bit	16bit	32bit
RRB OUT,N	RRW OUT,N	RRD OUT,N
RLB OUT,N	RLW OUT,N	RLD OUT,N

BIT nhớ bị tác động	
SM1.0	Được set lên 1 khi giá trị được dịch là 0
SM1.1	Chứa bit cuối cùng được dịch nếu N không là bội số của 6,16,32

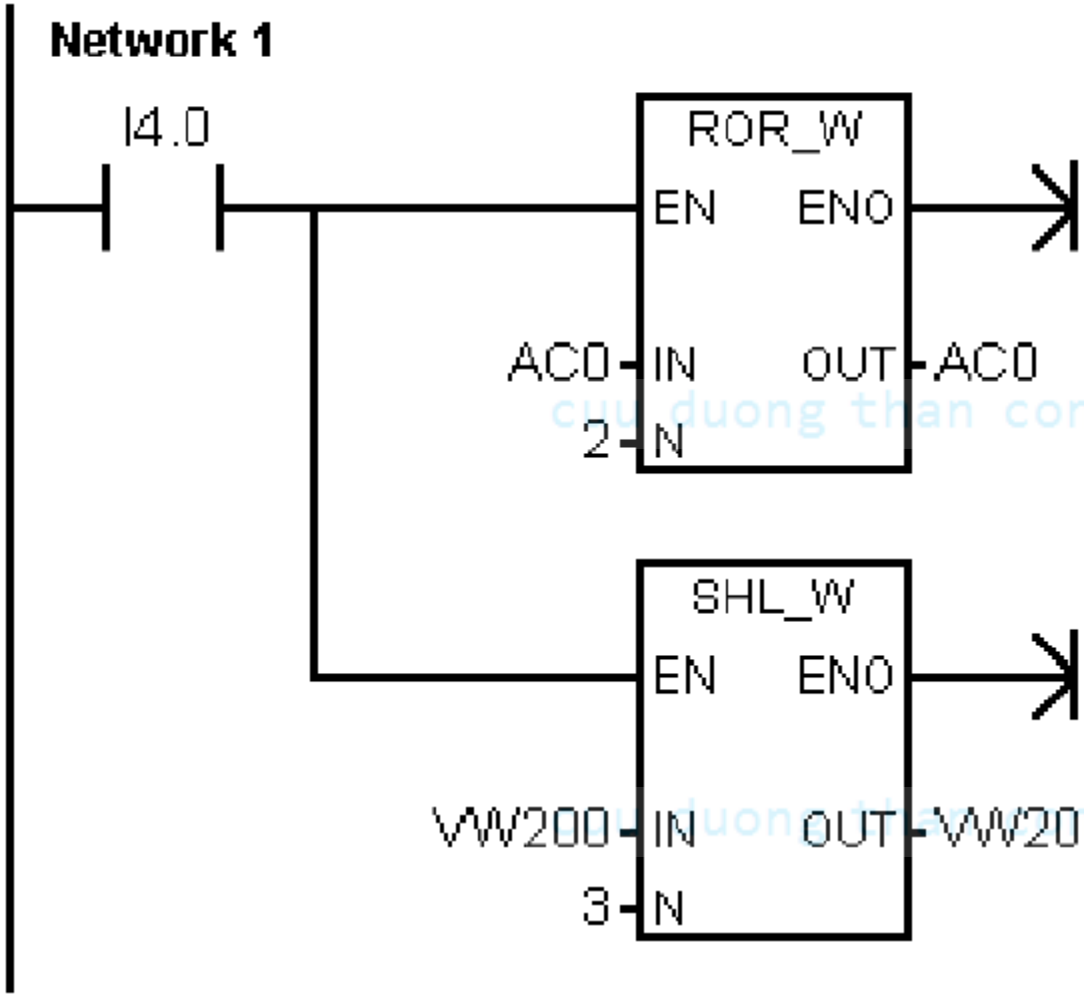
Lệnh Dời và Xoay

Địa chỉ hợp lệ

Inputs/ Outputs	Data Types	Operands
IN	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant
	DWORD	ID, QD, VD, MD, SMD, SD, LD, AC, HC, *VD, *LD, *AC, Constant
OUT	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC
	WORD	IW, QW, VW, MW, SMW, SW, T, C, LW, AQW, *VD, *LD, *AC
	DWORD	ID, QD, VD, MD, SMD, SD, LD, *VD, *LD, *AC
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, Constant, *VD, *LD, *AC

Lệnh Dời và Xoay

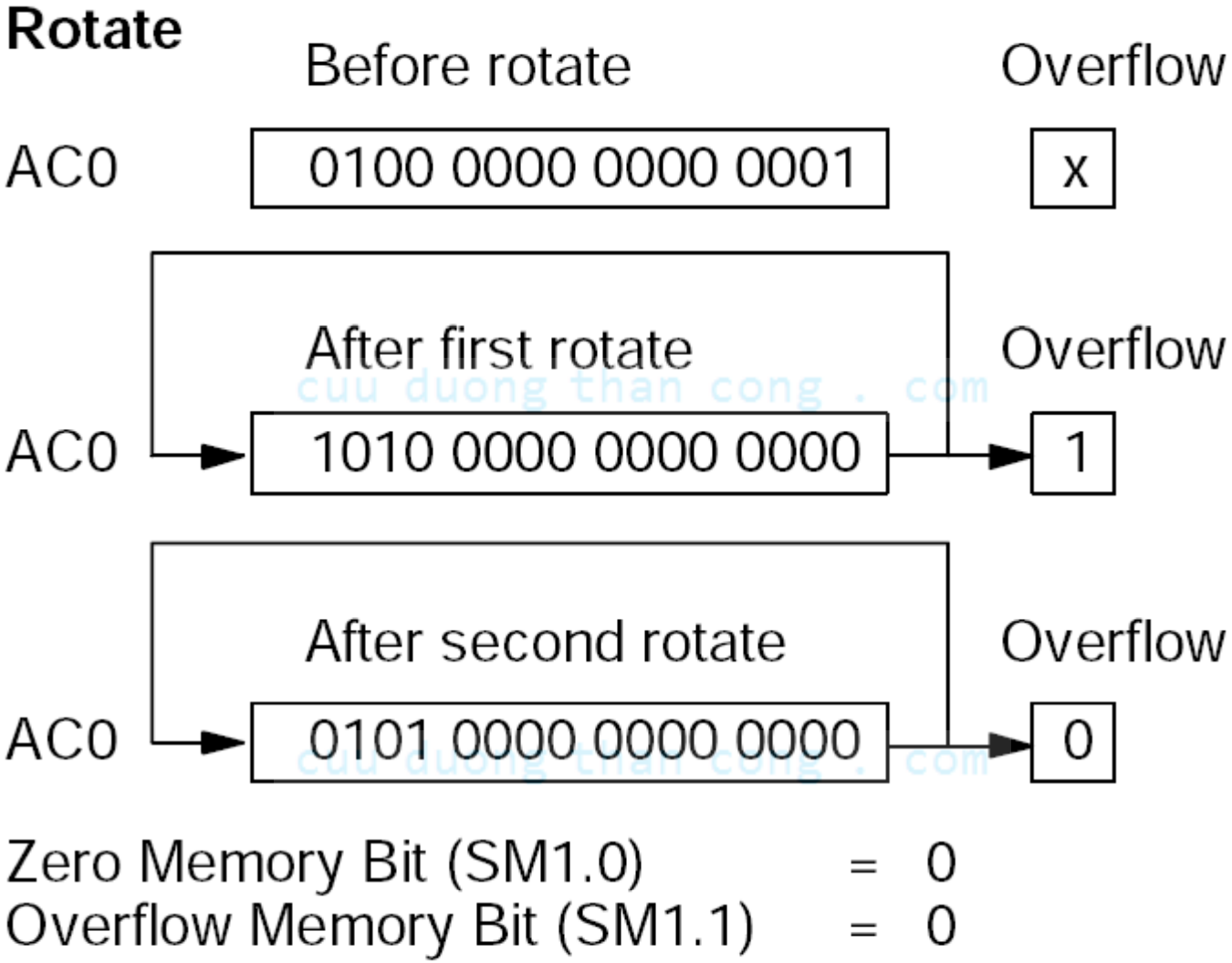
Ví dụ



Network 1
LD I4.0
RRW AC0, 2
SLW VW200, 3

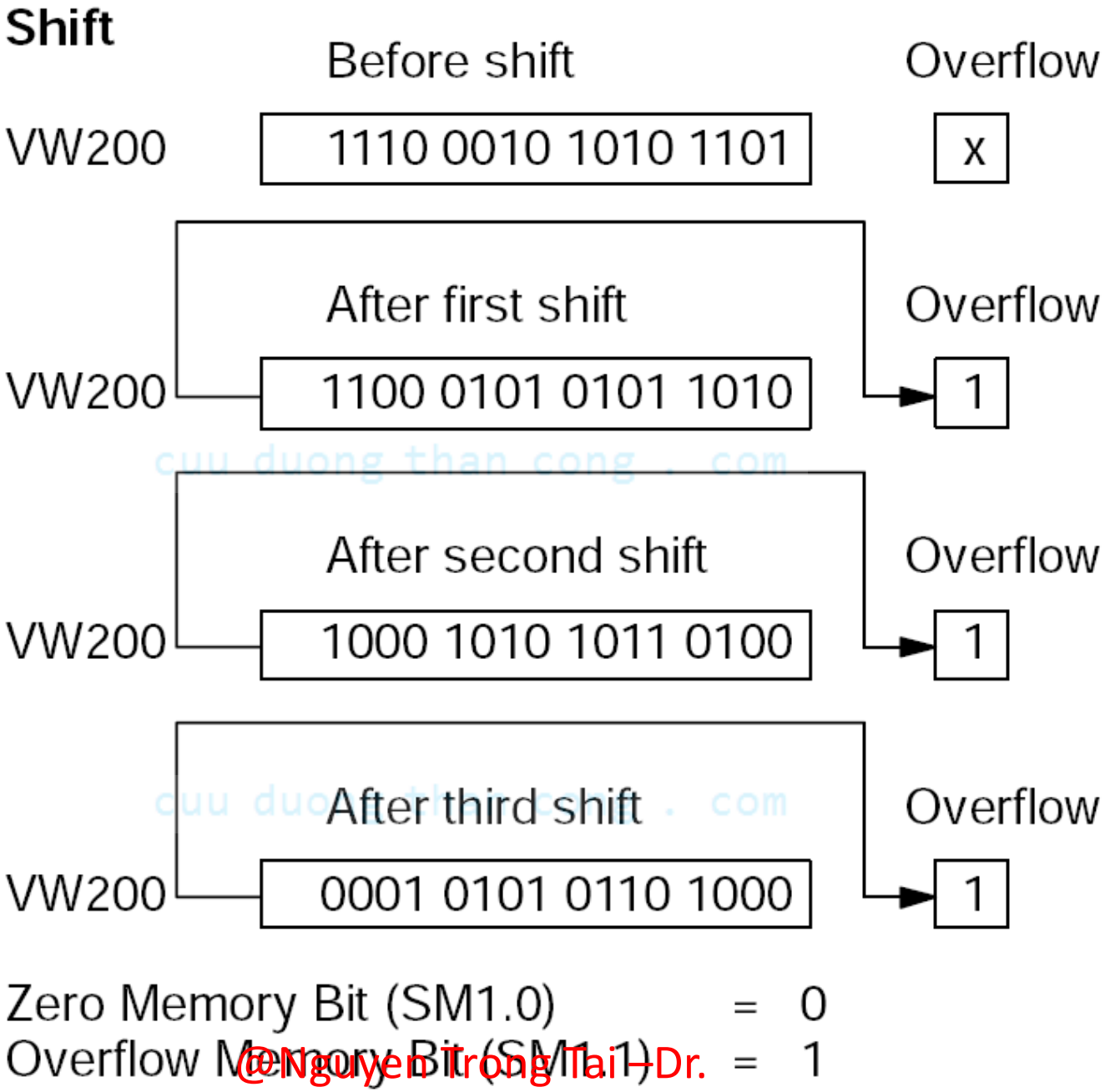
Lệnh Dời và Xoay

Ví dụ



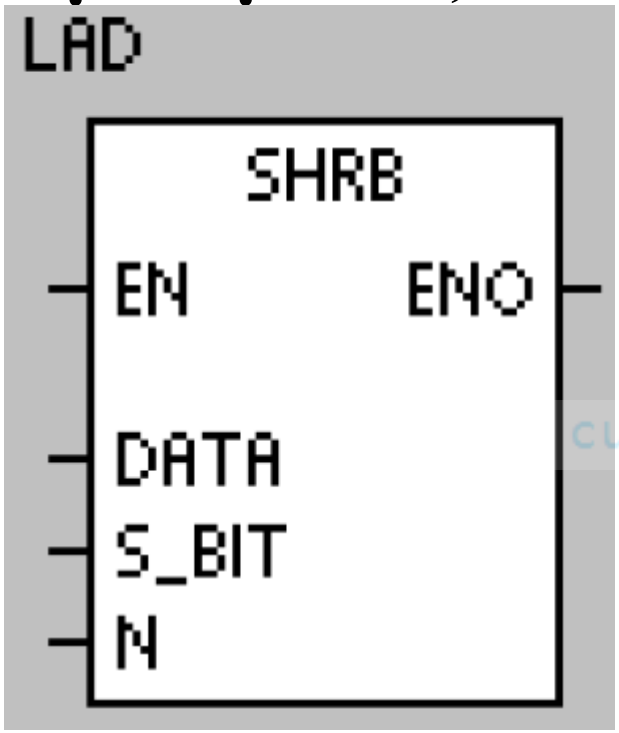
Lệnh Dời và Xoay

Ví dụ



Lệnh Dịch và Xoay

Lệnh dịch Bit;



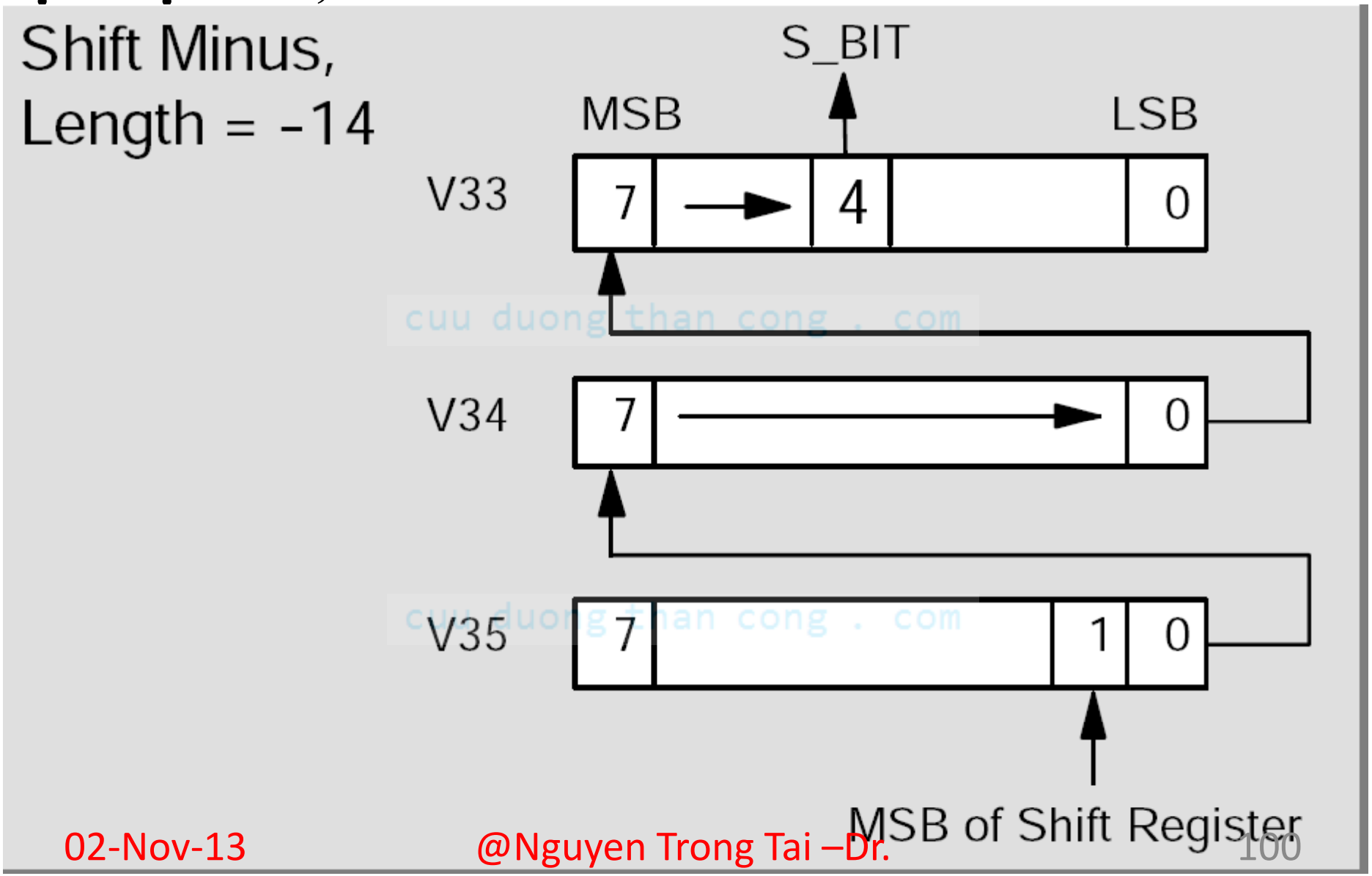
Dời một bit DATA vào thanh ghi chiều dài N có địa chỉ bit LSB là S_BIT. Khi N dương, DATA vào LSB, còn MSB dời ra SM1.1. Khi N âm, DATA vào MSB còn LSB dời ra SM1.1. N là chiều dài thanh ghi dịch. N<=64

Inputs/Outputs	Data Types	Operands
DATA, S_Bit	BOOL	I, Q, V, M, SM, S, T, C, L
N	BYTE	IB, QB, VB, MB, SMB, SB, LB, AC, *VD, *LD, *AC, Constant

Công thức xác định MSB của thanh ghi dịch
MSB.b = [Byte của S_bit + (N-1+bit của S_bit) div 8]. Số dư

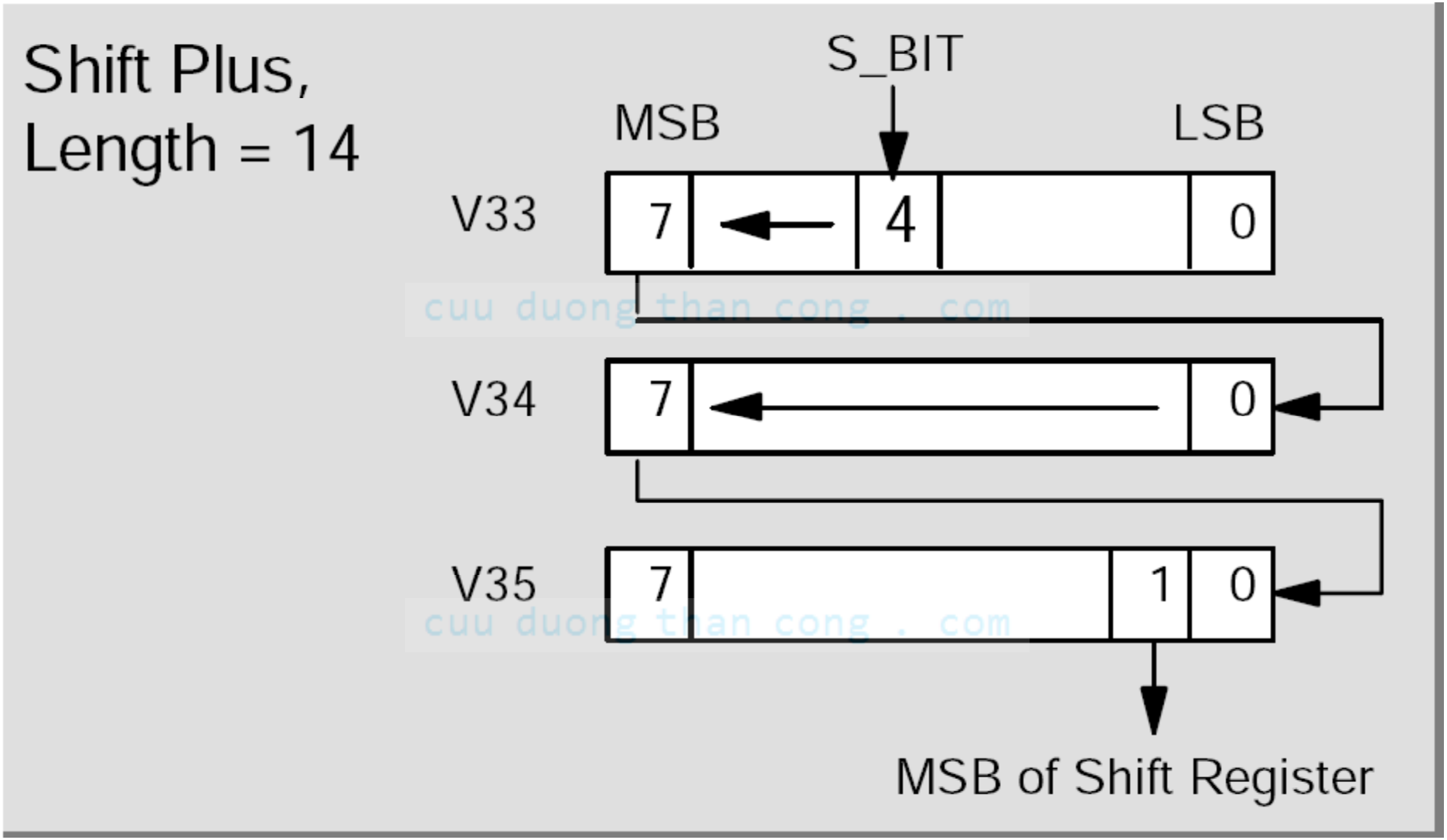
Lệnh Dịch và Xoay

Lệnh dịch Bit;



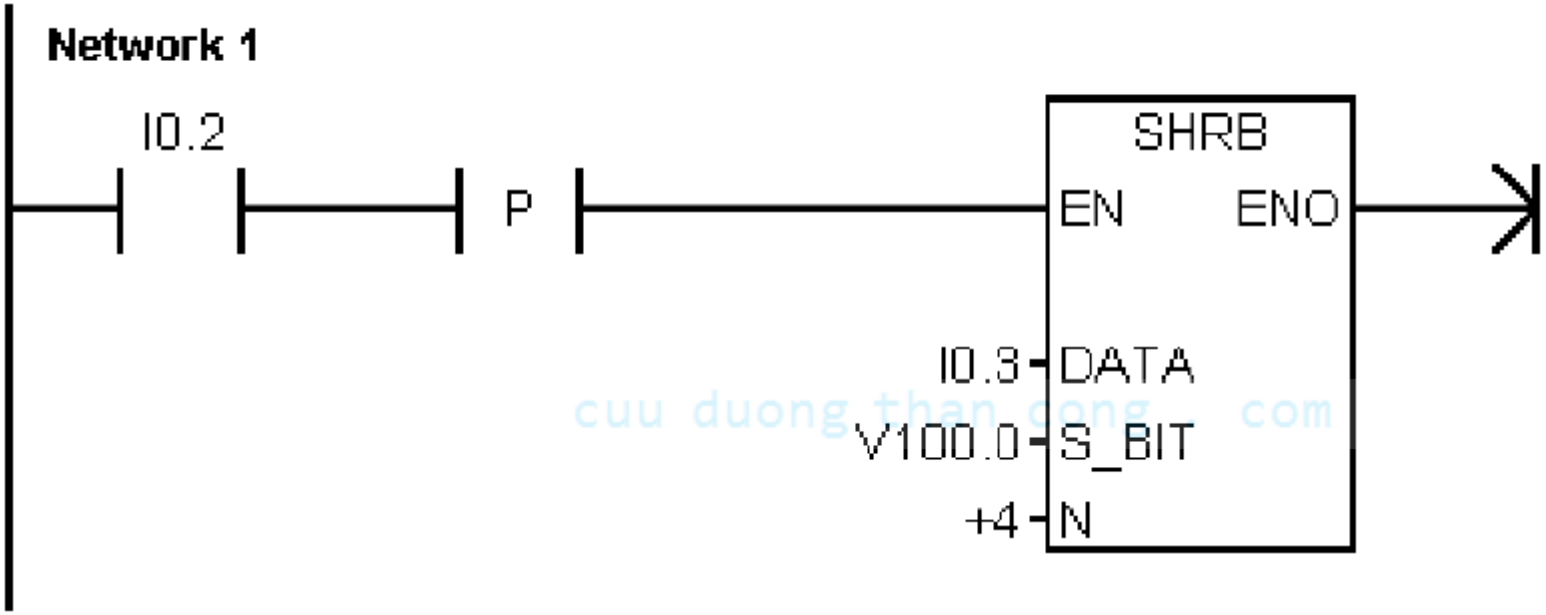
Lệnh Dịch và Xoay

Lệnh dịch Bit;



Lệnh Dịch và Xoay

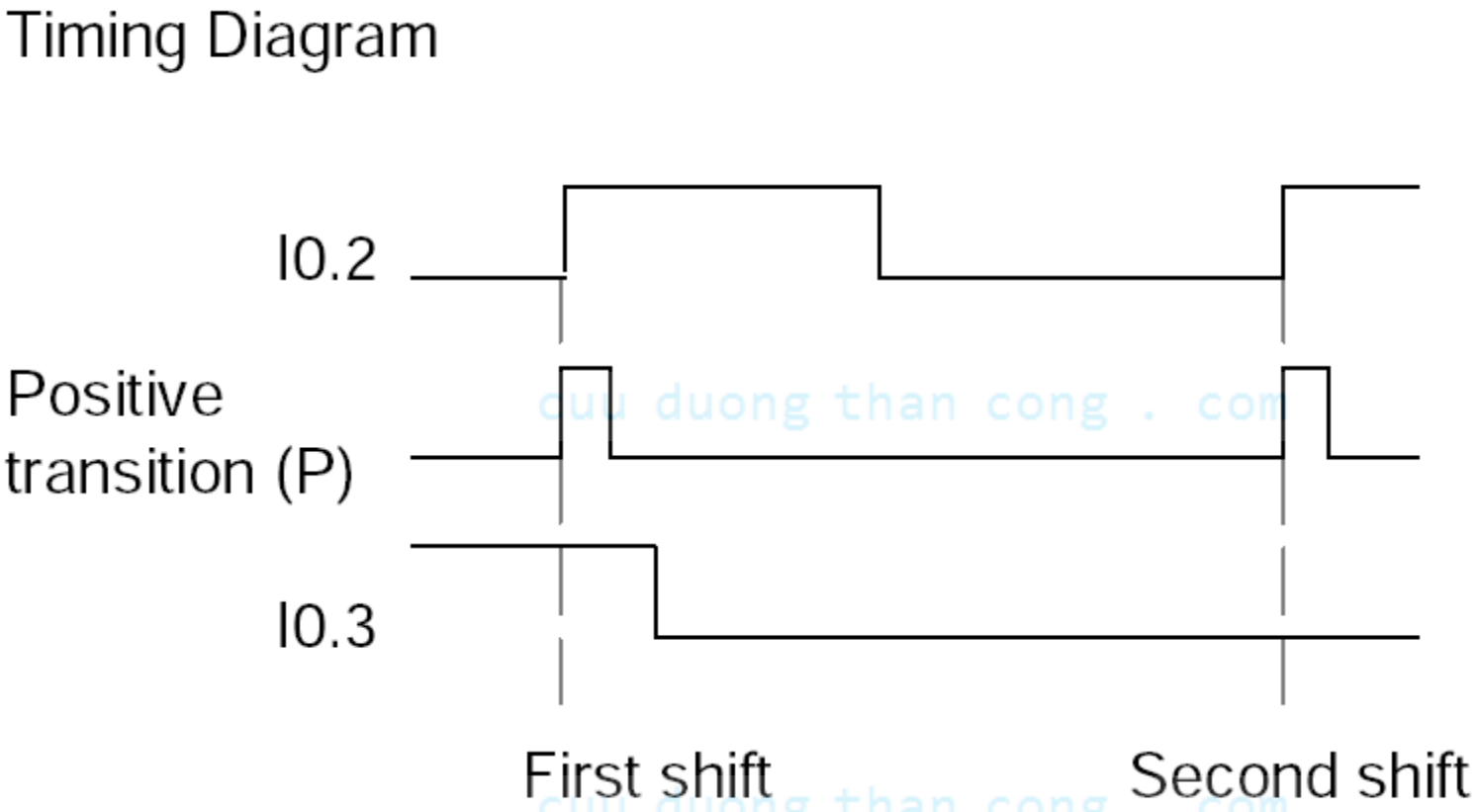
Ví dụ




```
LD I0.2
EU
SHRB I0.3, V100.0, +4
```

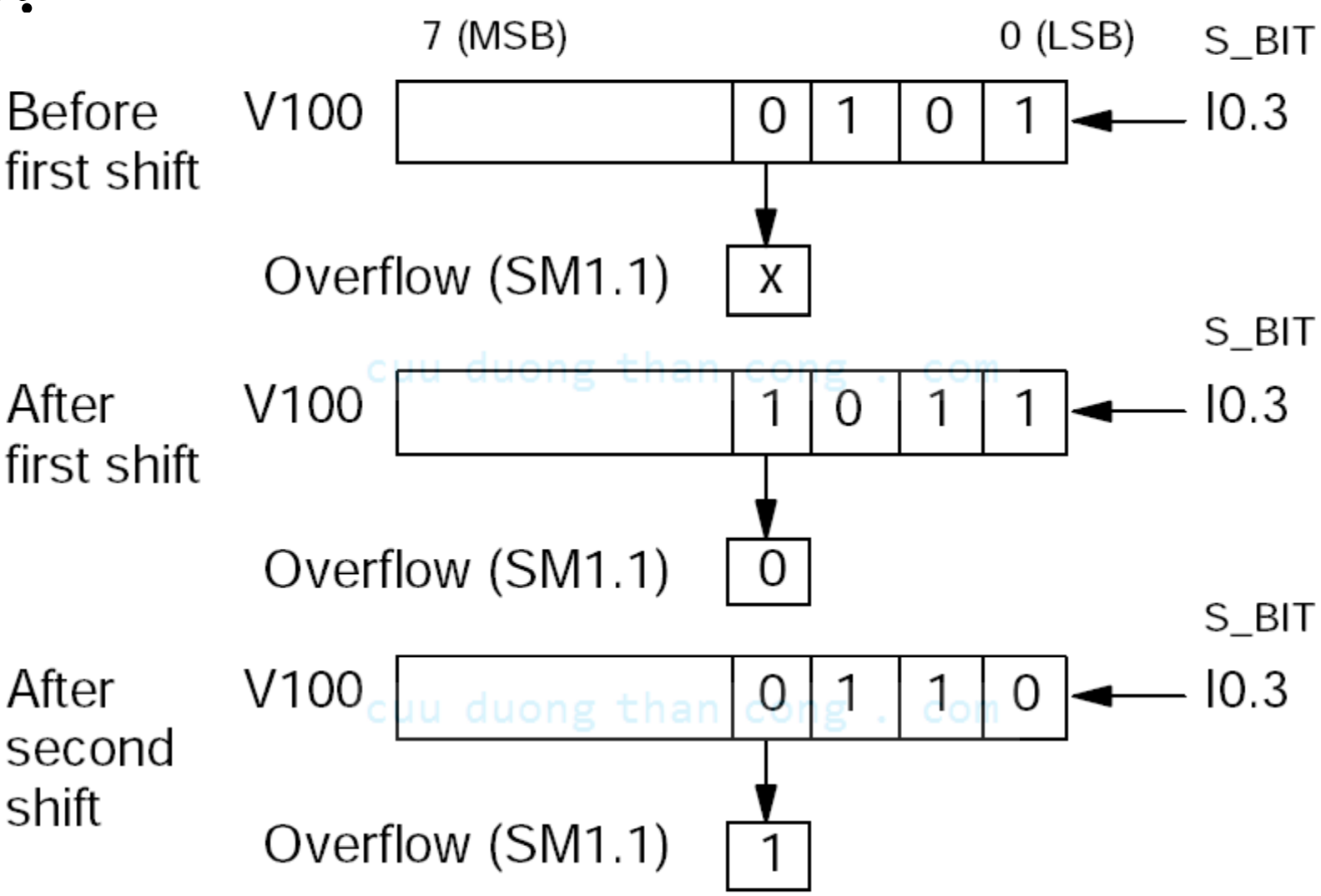
Lệnh Dịch và Xoay

Ví dụ



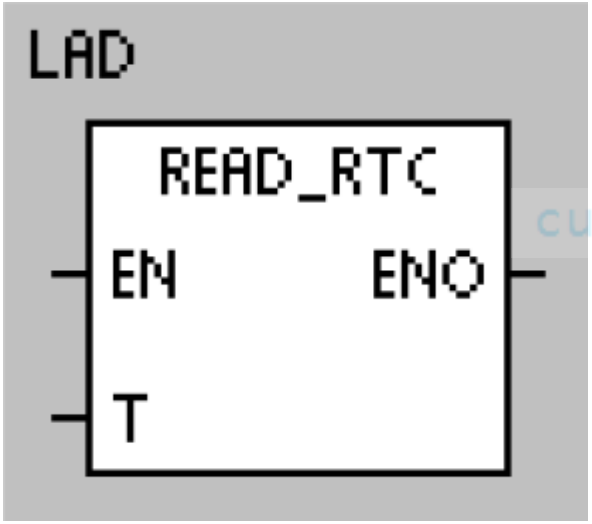
Lệnh Dịch và Xoay

Ví dụ



Lệnh Đồng hồ thời gian thực

Lệnh ghi và đọc thời gian thực READ_RTC(Read Real-time Clock); SET_RTC (Write –Set Real-Time Clock)



TOD: Time of Day

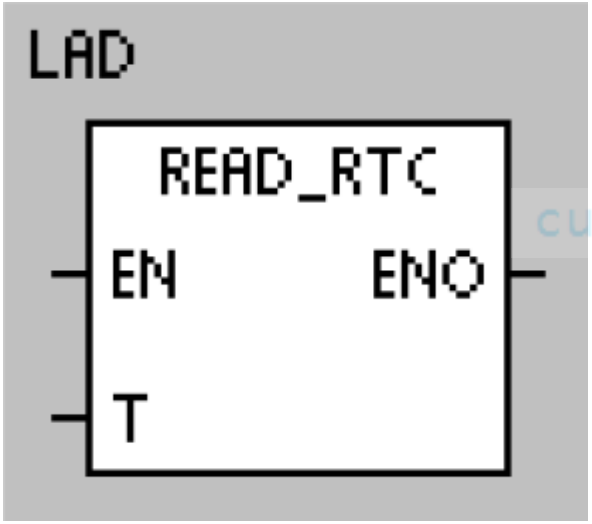
T: được định dạng BCD, 8 bite

*: 1=Sunday; 7:Saturday; 0: disable day of week

Định dạng của buffer T							
T	T+1	T+2	T+3	T+4	T+5	T+6	T+7
Year	Month	Day	Hours	Minutes	Seconds	0	Day of Week*
00:99	01:12	01:31	00:23	00:59	00:59		0:7

Lệnh Đồng hồ thời gian thực

Lệnh ghi và đọc thời gian thực READ_RTC(Read Real-time Clock); SET_RTC (Write –Set Real-Time Clock)



TOD: Time of Day

T: được định dạng BCD, 8 byte

*: 1=Sunday; 7:Saturday; 0: disable day of week

Inputs/ Outputs	Data Types	Operands
T	BYTE	IB, QB, VB, MB, SMB, SB, LB, *VD, *LD, *AC

00:99	01:12	01:31	00:23	00:59	00:59		0:7

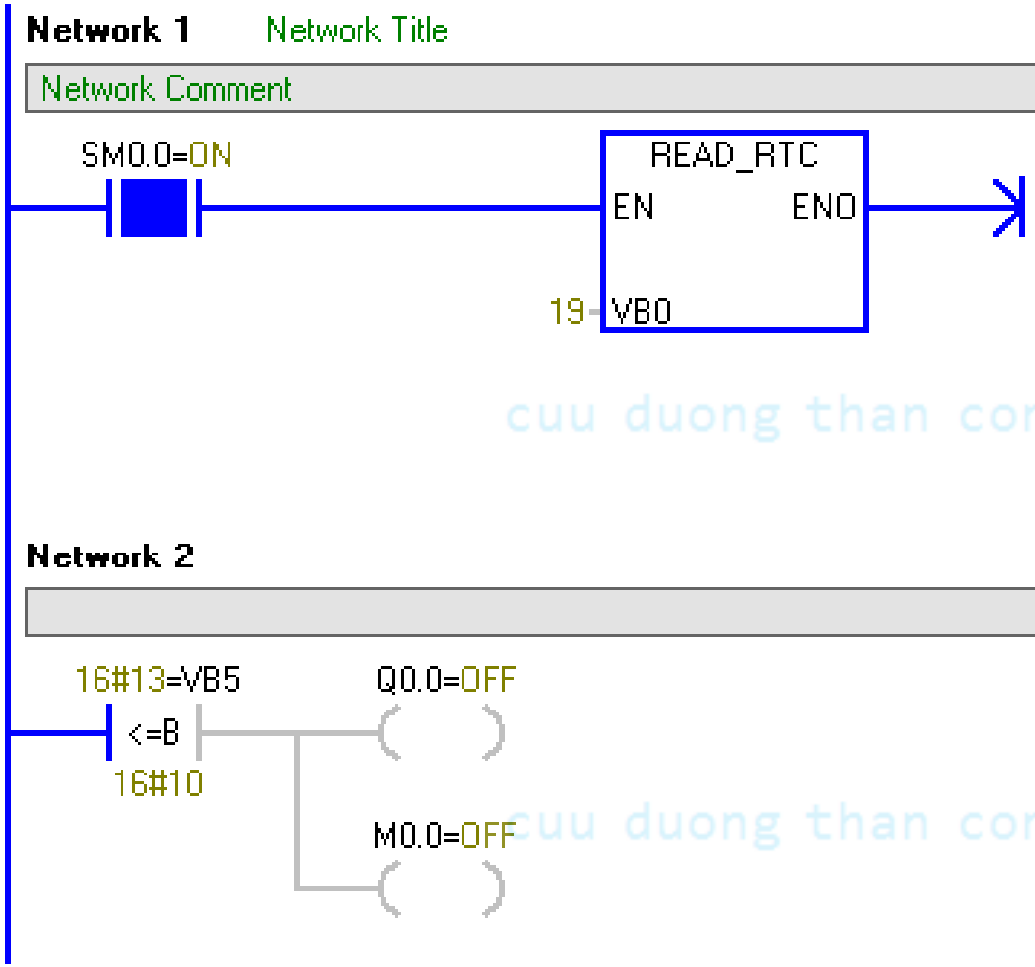
Lệnh Đồng hồ thời gian thực

Lệnh ghi và đọc Real Time Clock Mở rộng (TODRX, TODWX)

(Tự đọc trong manual)

Lệnh Đồng hồ thời gian thực

Ví dụ: Tạo xung 10s sau mỗi phút dùng thời gian thực

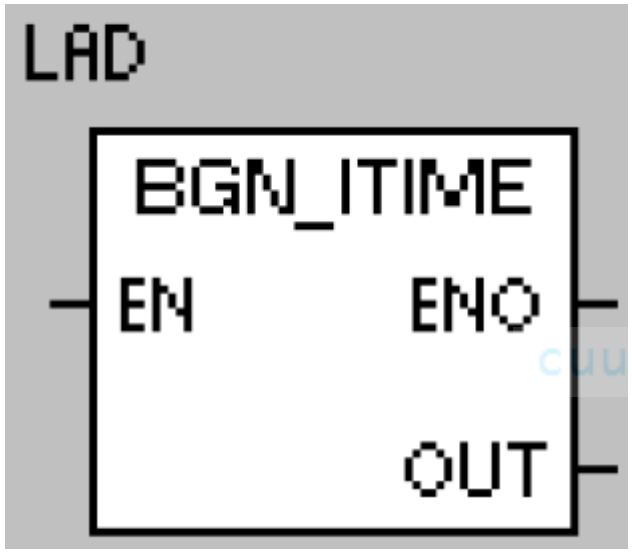


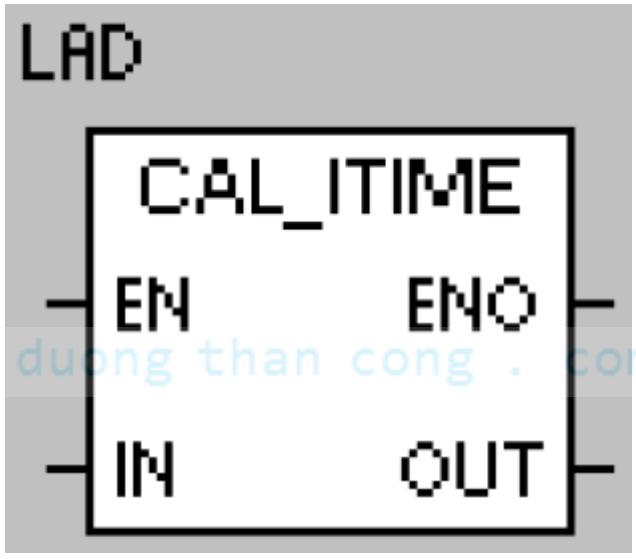
Lệnh Tính Khoảng thời gian

PLC có bộ đếm 32bit đếm xung 1msec, khi EN ON

lệnh BGN_ITIME (Beginning interval time) đưa giá trị trong bộ đếm này vào từ kép OUT;

lệnh CAL_ITIME (calculating interval time) tính hiệu số giá trị trong bộ đếm và giá trị IN chứa vào OUT

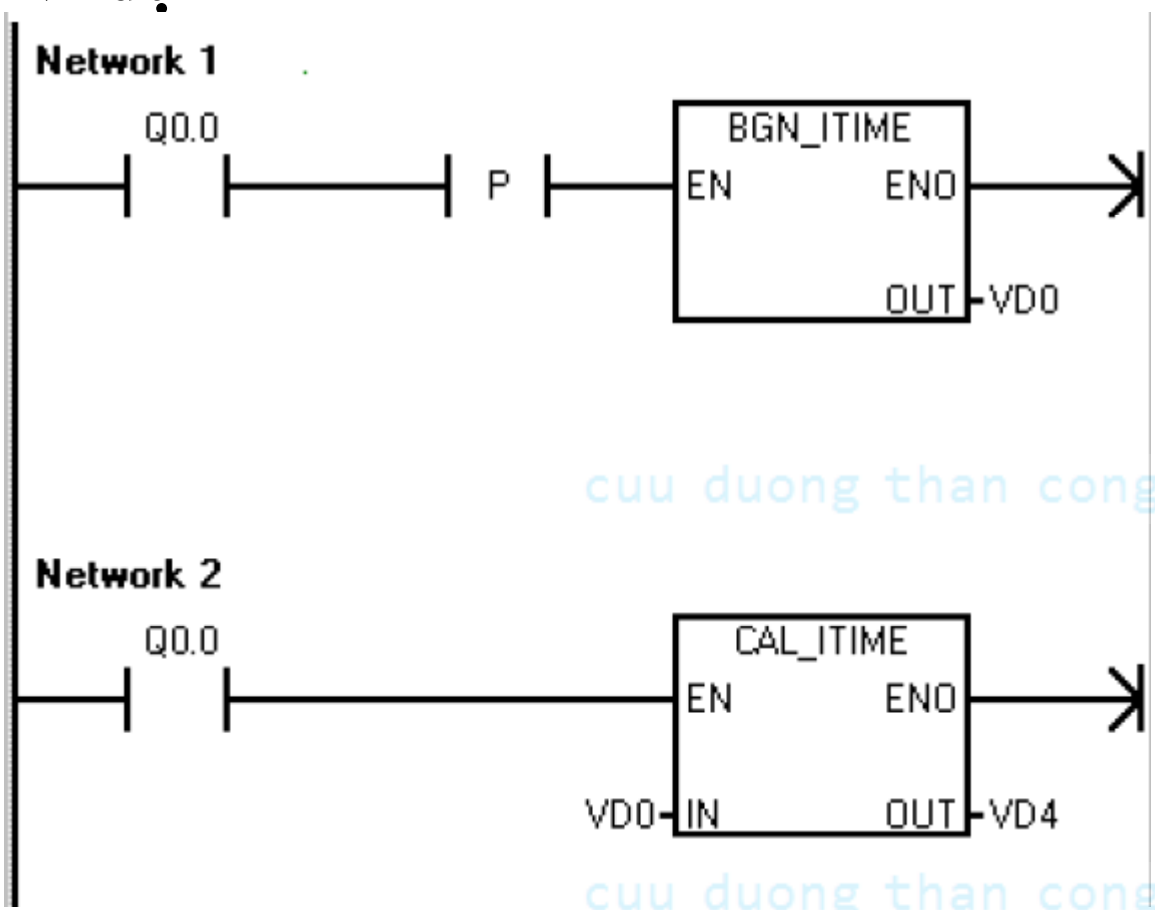




Thời gian đo tối đa là 49.7 ngày

Lệnh Tính Khoảng thời gian

Ví dụ



Network 1 //Capture the time that Q0.0 turned on.

LD Q0.0

EU

BITIM VD0

Network 2 // Calculate time Q0.0 has been on.

LD Q0.0

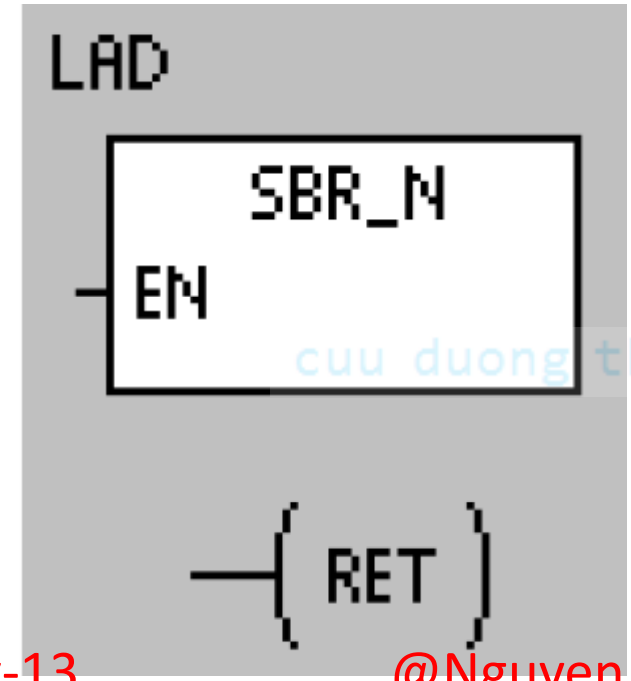
CITIM VD0, VD4

Lệnh Gọi chương trình con

Có thể gọi tối đa 64 chương trình con

Chương trình con viết sau chương trình chính và được đóng khung bằng SBR_N và RET. Chương trình con được gọi bằng lệnh CALL SRB_N. Chương trình con có thể kèm theo tham số vào, ra.

Tham số vào, ra được khai báo trong bảng biến cục bộ của chương trình con.



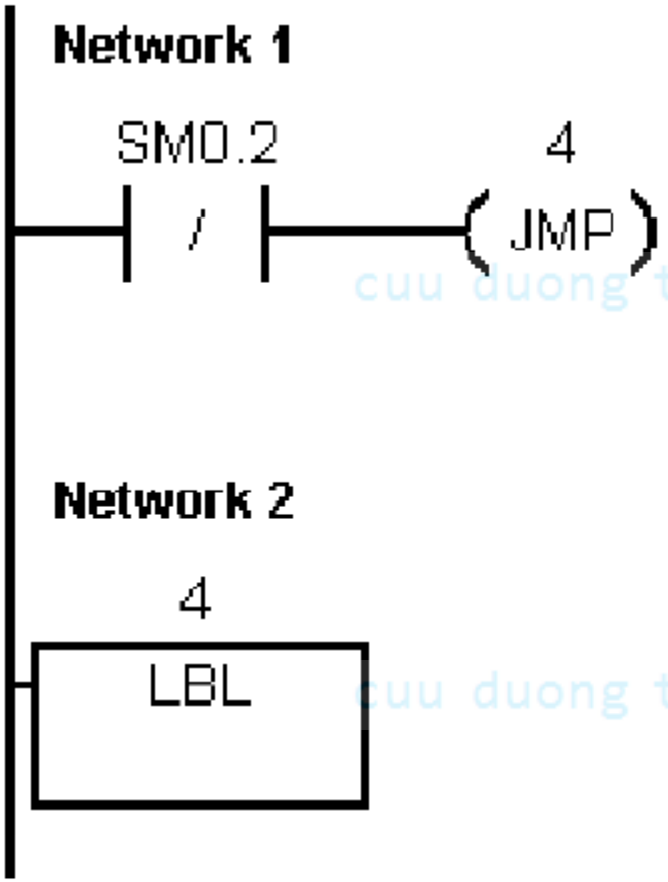
Edit > Insert > Subroutine

Lệnh Gọi chương trình con

	Symbol	Var Type	Data Type	Comment
	EN	IN	BOOL	
		IN		
		IN_OUT		
		OUT		
		TEMP		

Lệnh điều khiển chương trình

1. Lệnh **JMP n** : nhảy đến đoạn chương trình nhãn n; n = 0:255
2. Lệnh **LBL n**: đặt nhãn n;



Network 1 //If the retentive data has not been lost,
//Jump to LBL4
LDN SM0.2
JMP 4
Network 2
LBL 4

Lệnh điều khiển chương trình

3. Lệnh **SCR** (Sequence Control Relay) ➔để thực hiện một đoạn chương trình lặp

Đoạn chương trình được đóng khung bởi **SCR** và **SCRE**,

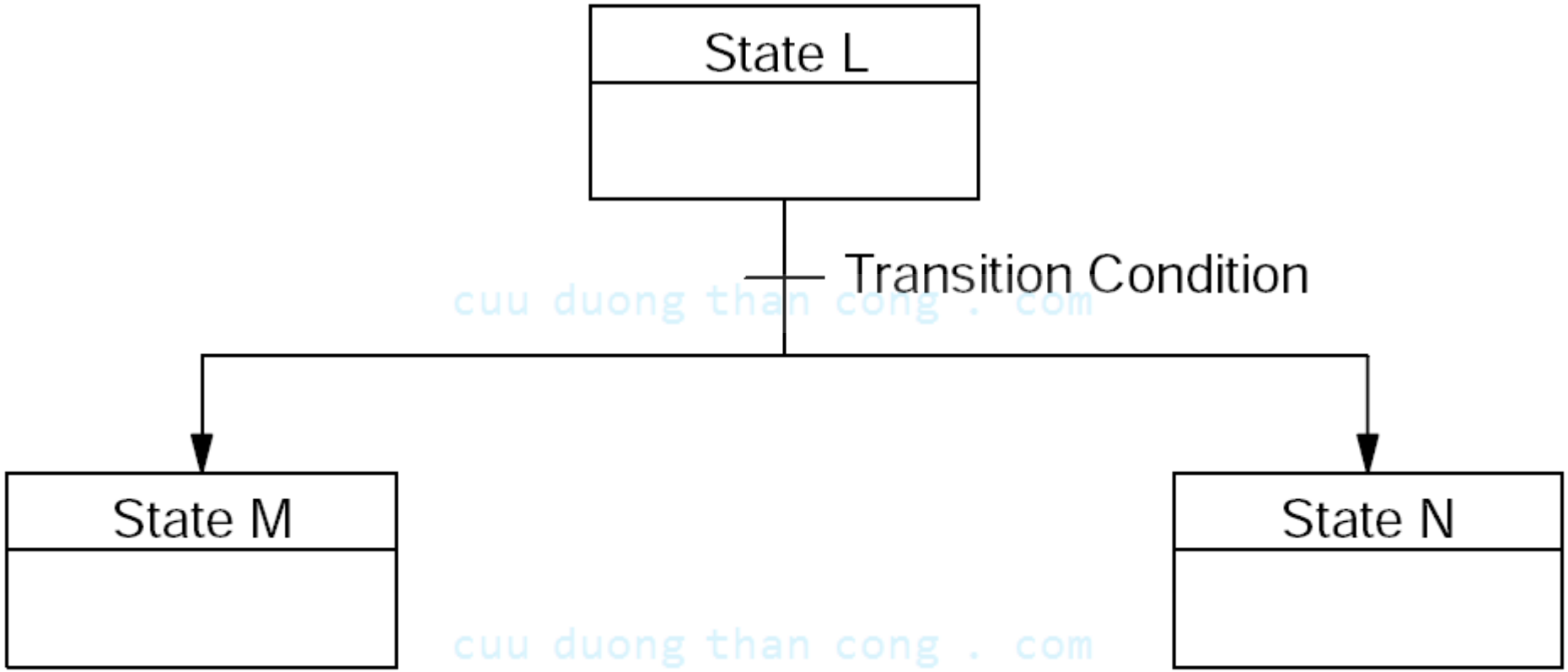
Được thực hiện lặp lại khi bit điều khiển tương ứng vùng nhớ Sx.y ON ứng với 1 trạng thái.

Muốn chuyển sang trạng thái khác, ta dùng lệnh SCRT cho một bit khác ON và tắt bit S của trạng thái hiện tại.

3 loại chuyển tiếp trạng thái:

Lệnh điều khiển chương trình

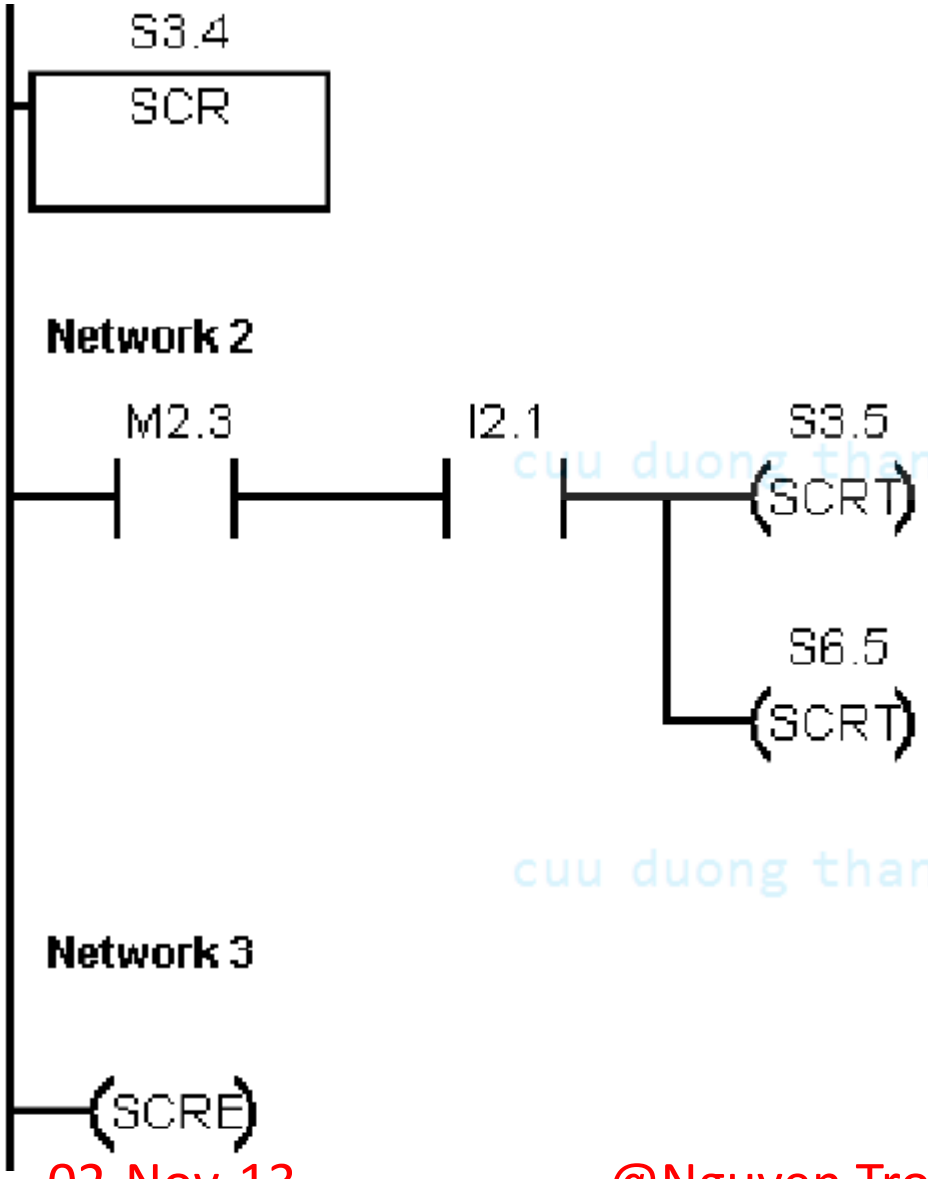
3. Lệnh **SCR**: 3 dạng chuyển tiếp trạng thái:



Dạng phân kỳ

Lệnh điều khiển chương trình

3. Lệnh **SCR**: 3 dạng chuyển tiếp trạng thái:

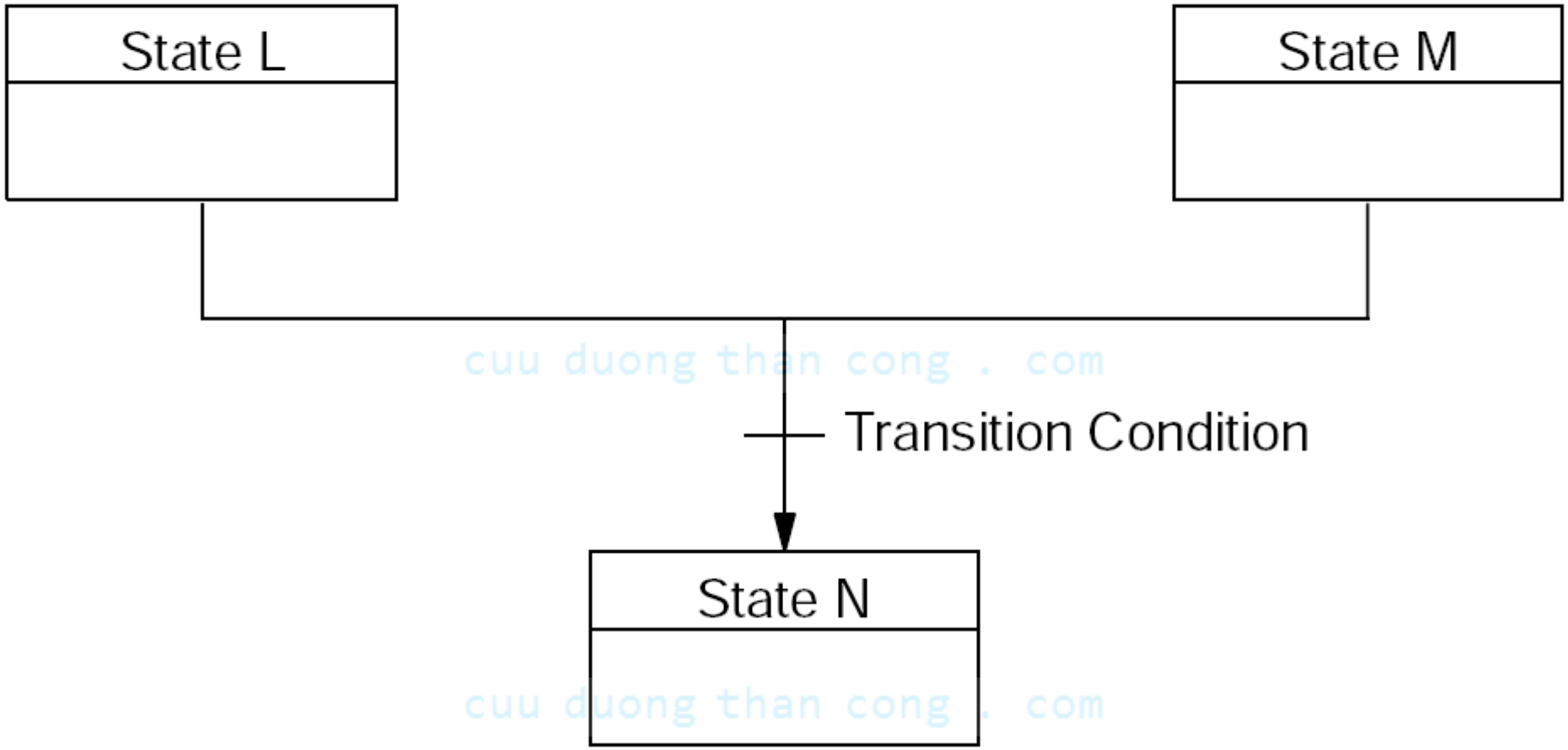


Network 1 //Beginning of State L control region.
LSCR S3.4
Network 2
LD M2.3
A I2.1
SCRT S3.5 //Transition to State M
SCRT S6.5 //Transition to State N
Network 3 //End of the State region for State L.
SCRE

Dạng phân kỳ

Lệnh điều khiển chương trình

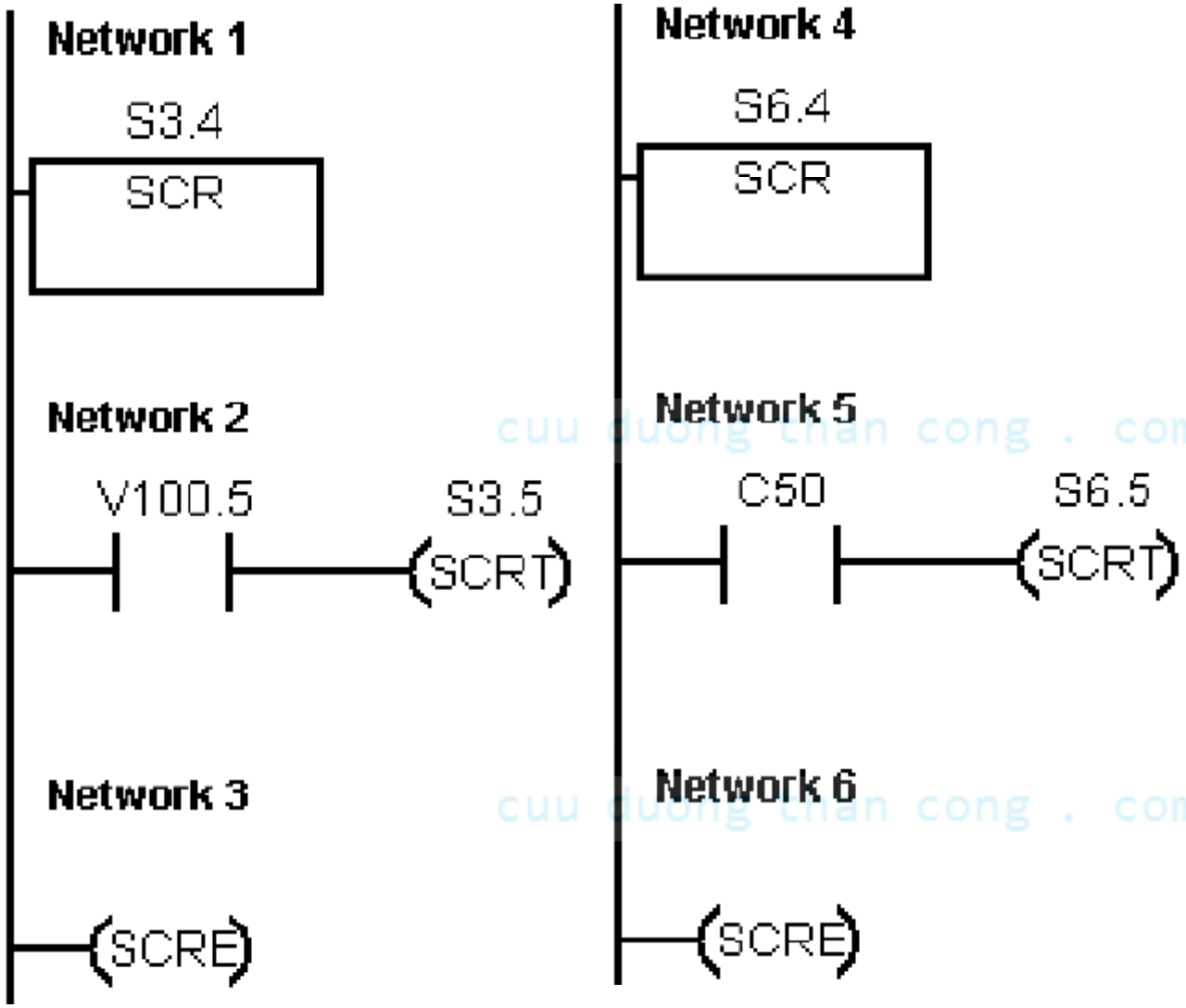
3. Lệnh **SCR**: 3 dạng chuyển tiếp trạng thái:



Dạng hội tụ

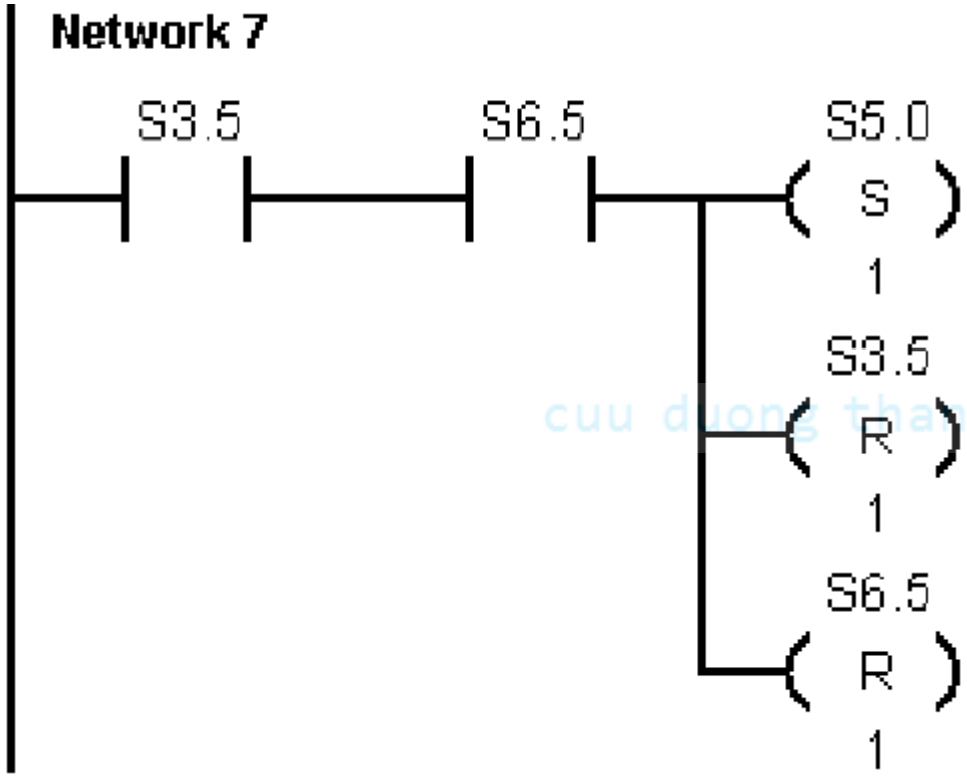
Lệnh điều khiển chương trình

3. Lệnh **SCR**: 3 dạng chuyển tiếp trạng thái:



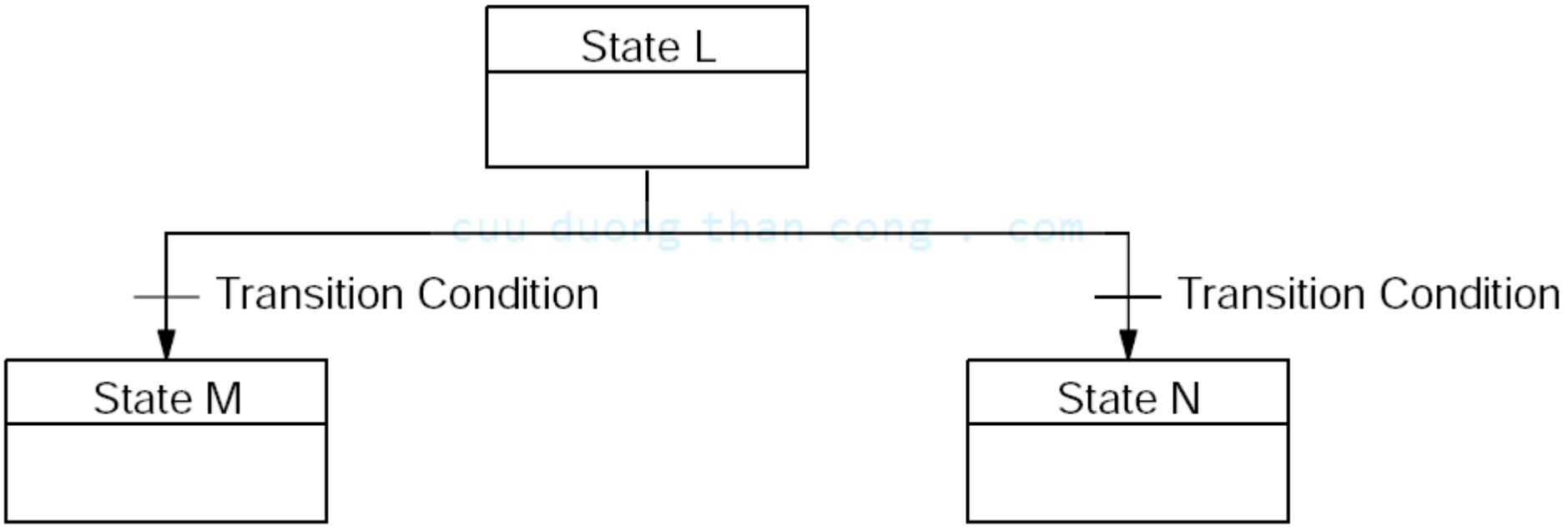
Lệnh điều khiển chương trình

3. Lệnh **SCR**: 3 dạng chuyển tiếp trạng thái:



Lệnh điều khiển chương trình

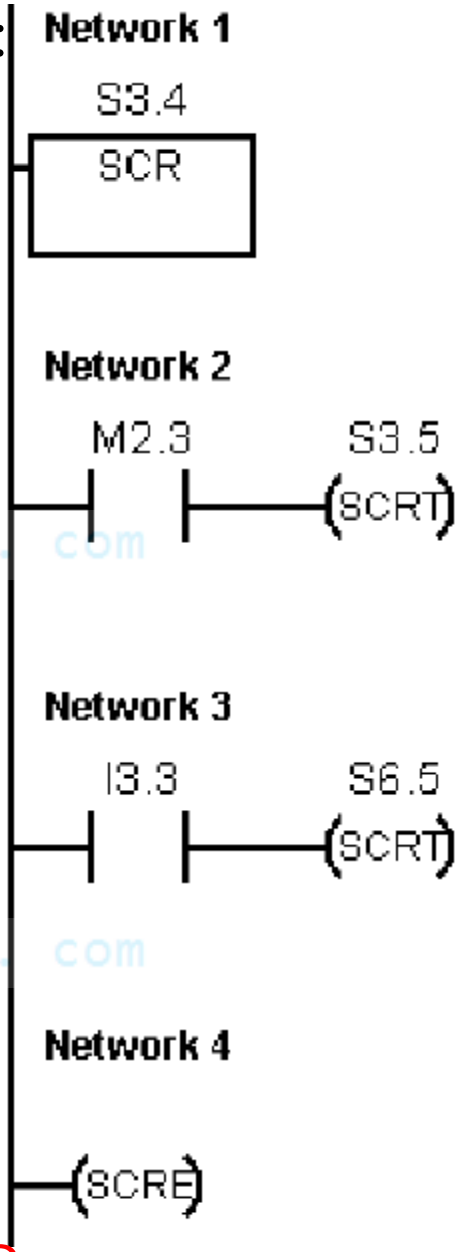
3. Lệnh **SCR**: 3 dạng chuyển tiếp trạng thái:



Dạng rẽ có điều kiện

Lệnh điều khiển chương trình

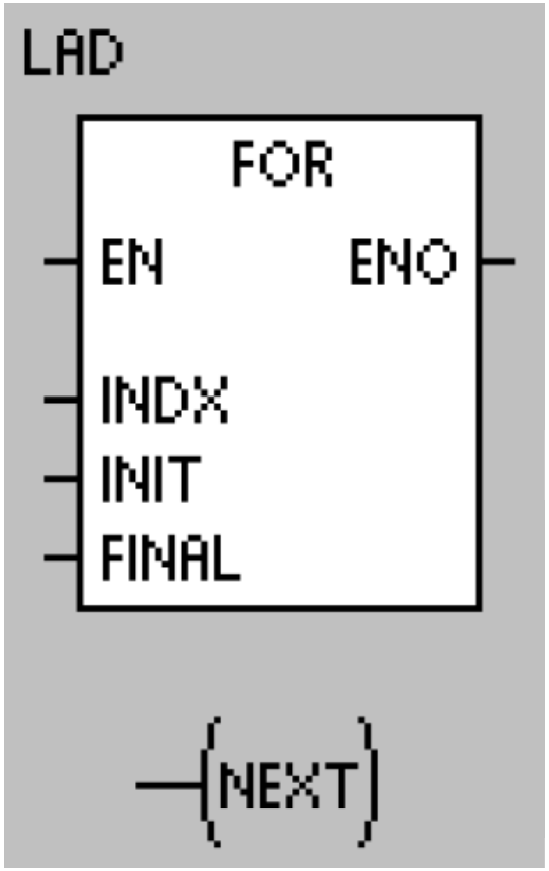
3. Lệnh **SCR**: 3 dạng chuyển tiếp trạng thái:



Dạng rẽ có điều kiện

Lệnh điều khiển chương trình

3. Lệnh **FOR … NEXT**: lặp vòng các dòng lệnh giữa FOR…NEXT



INDX: vòng lặp hiện tại

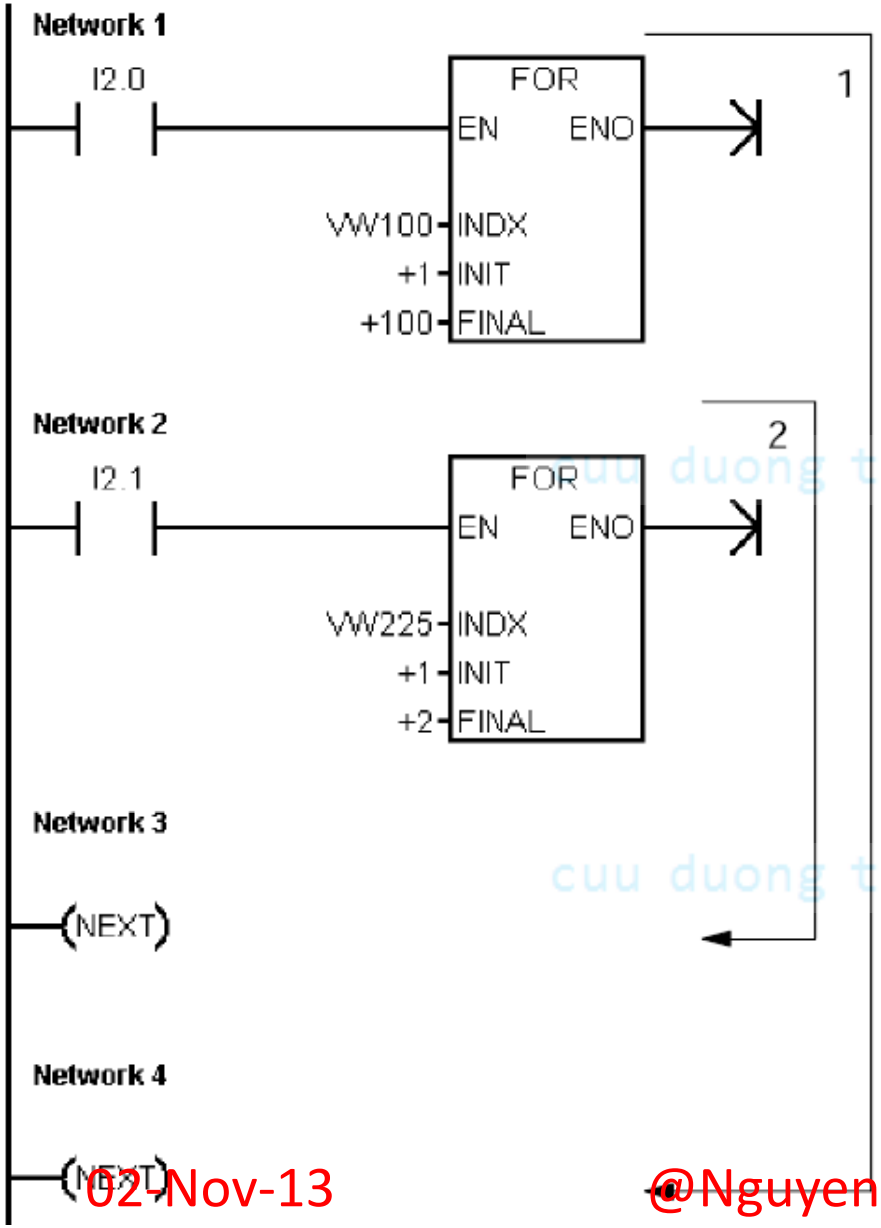
INIT: giá trị khởi đầu

FINAL: giá trị cuối cùng

Inputs/ Outputs	Data Types	Operands
INDX	INT	IW, QW, VW, MW, SMW, SW, T, C, LW, AC, *VD, *LD, *AC
INT, FINAL	INT	VW, IW, QW, MW, SMW, SW, T, C, LW, AC, AIW, *VD, *LD, *AC, Constant

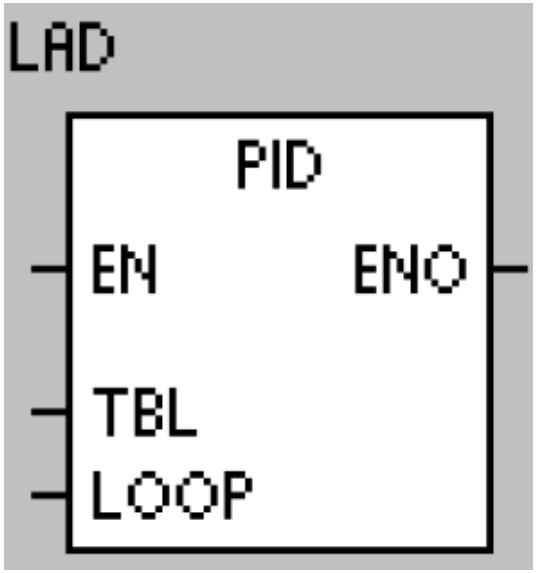
Lệnh điều khiển chương trình

3. Lệnh **FOR … NEXT**: lặp vòng các dòng lệnh giữa FOR…NEXT



Network 1
LD I2.0
FOR VW100, +1, +100
Network 2
LD I2.1
FOR VW225, +1, +2
Network 3 //End of Loop 2.
NEXT
Network 4 //End of Loop 1.
NEXT

Lệnh PID



Inputs/ Outputs	Data Types	Operands
TBL	BYTE	VB
LOOP	BYTE	Constant (0:7)

$$M(t) = MP_n + MI_n + MD_n$$

$$MP_n = K_C * (SP_n - PV_n)$$

$$MI_n = K_C * T_s / T_I (SP_n - PV_n) + MI_{n-1}$$

$$MD_n = K_C * T_D / T_S (PV_{n-1} - PV_n) + MI_{n-1}$$

STEP 7 Micro/WIN hỗ trợ PID Wizard cho phép thực hiện tự động thuật toán PID

Lệnh PID

Offset	Field	Format	Type	Description
0	Process Variable (PVn)	Real	In	Phản hồi (0.0:1.0)
4	Setpoint (SPn)	Real	In	Giá trị đặt (0.0:1.0)
8	Output (Mn)	Real	In/Out	Giá trị ngõ ra (0.0:1.0)
12	Gan (Kc)	Real	In	Hệ số tỉ lệ (>0 hoặc <0)
16	Sample time (Ts)	Real	In	Thời gian lấy mẫu (>0)
20	Integral time or reset (Ti)	Real	In	Thời gian tích phân (>0)
24	Derivative time or rate (Td)	Real	In	Thời gian vi phân (>0)
28	Bias (MX)	Real	In/Out	Giá trị Bias hoặc tổng tích phân
32	Previous Process Variable (PVn-1)	Real	In/Out	Phản hồi ở thời điểm trước đó
36-79	Reserved for auto-tuning variables			

Lệnh PID

Chuẩn hóa tín hiệu từ cảm biến

$R_{norm} = R_{Raw}/Span + Offset$	
R_{Norm}	Giá trị được chuẩn hóa
R_{Raw}	Giá trị thực
Offset	0.0 for unipolar values 0.5 for bipolar values
Span	Maximum possible value – Minimum possible value = 32000 for unipolar values (typical) = 64000 for bipolar values (typical)

```
ITD AIW0, AC0 //Convert an input value to a double word
DTR AC0, AC0 //Convert the 32-bit integer to a real number
/R 64000.0, AC0 //Normalize the value in the accumulator
+R 0.5, AC0 //Offset the value to the range from 0.0 to 1.0
MOVR AC0, VD100 //Store the normalized value in the loop TABLE
```

Lệnh PID

Chuẩn hóa tín hiệu từ cảm biến

$R_{norm} = R_{Raw}/Span + Offset$	
R_{Norm}	Giá trị được chuẩn hóa
R_{Raw}	Giá trị thực
Offset	0.0 for unipolar values 0.5 for bipolar values
Span	Maximum possible value – Minimum possible value = 32000 for unipolar values (typical) = 64000 for bipolar values (typical)

```
ITD AIW0, AC0 //Convert an input value to a double word
DTR AC0, AC0 //Convert the 32-bit integer to a real number
/R 64000.0, AC0 //Normalize the value in the accumulator
+R 0.5, AC0 //Offset the value to the range from 0.0 to 1.0
MOVR AC0, VD100 //Store the normalized value in the loop TABLE
```

Lệnh PID

Chuẩn hóa tín hiệu cho ngõ ra

$R_{Scale} = (M_n - Offset) * Span$	
R_{Norm}	Giá trị thực ngõ ra
R_{Raw}	Giá trị được chuẩn hóa
Offset	0.0 for unipolar values 0.5 for bipolar values
Span	Maximum possible value – Minimum possible value = 32000 for unipolar values (typical) = 64000 for bipolar values (typical)


```
MOVR VD108, AC0 //Moves the loop output to the accumulator
-R 0.5, AC0 //Include this statement only if the value is bipolar
*R 64000.0, AC0 //Scales the value in the accumulator
ROUND AC0, AC0 //Converts the real number to a 32-bit integer
DTI AC0, LW0 //Converts the value to a 16-bit integer
MOVW LW0, AQW0 //Writes the value to the analog output
```

Lệnh PID

Ví dụ: Đổi giá trị đo trong AIW0, chuẩn hóa 64000, Chứa trong VD100

```
XORD  AC0, AC0                  //Xóa ACC
MOVW        AIW0, AC0           //Cất trị đo vào ACC
LDW>=       AC0, 0                      //Nếu dương
JMP         0                           //Đổi sang số thực
NOT                                     //Nếu âm
ORD         16#FFFF0000, AC0            //Triển khai dấu cho AC0
LBL         0
DTR         AC0, AC0                    //Đổi sang số thực
/R          64000.0, AC0                //Chuẩn hóa
+R          0.5, AC0
MOVR AC0, VD100
```

Lệnh PID

Ví dụ: Lập trình PID điều khiển mực nước trong bồn 70%:

AIW0: sensor, AQW0: output, ts = 0.1s

```
//Main
Network 1
LD                  SM0.1
MOVR        0.75, VD104     // Điểm đặt 75%
MOVR        0.25, VD112     // Độ lợi vòng
MOVR        0.10, VD116     //
MOVR        30.0, VD120     // T_I
MOVR        0.0, VD124      // Không dùng đạo hàm
MOVB        100, SMB34      // Ngắt thời gian 0.1s gọi ID
ATCH        0,10
ENI
```

Lệnh PID

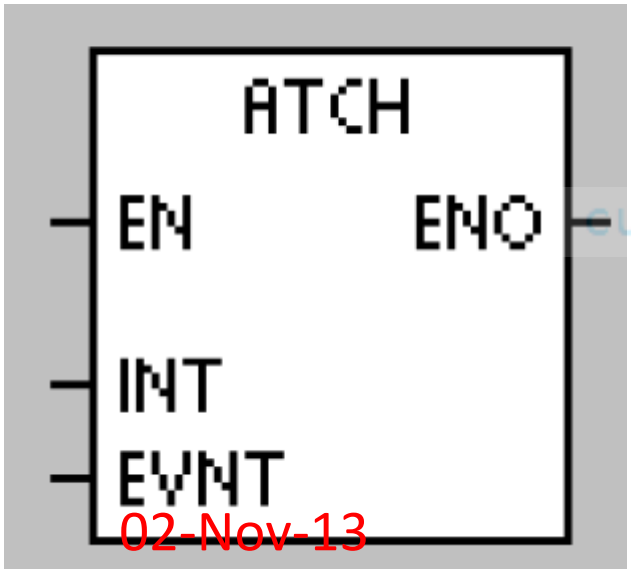
Ví dụ: Lập trình PID điều khiển mực nước trong bồn 70%:

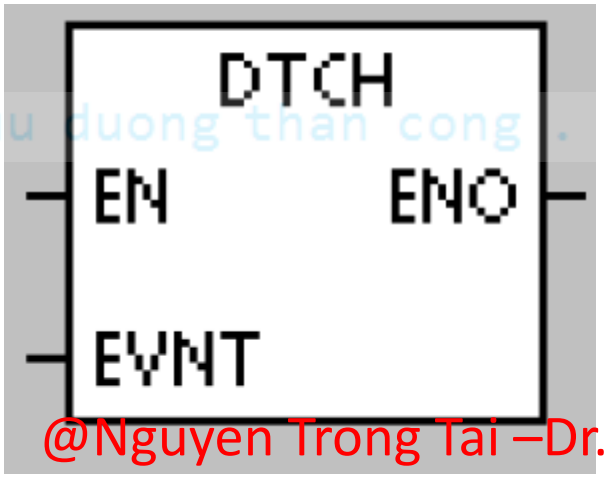
AIW0: sensor, AQW0: output, ts = 0.1s

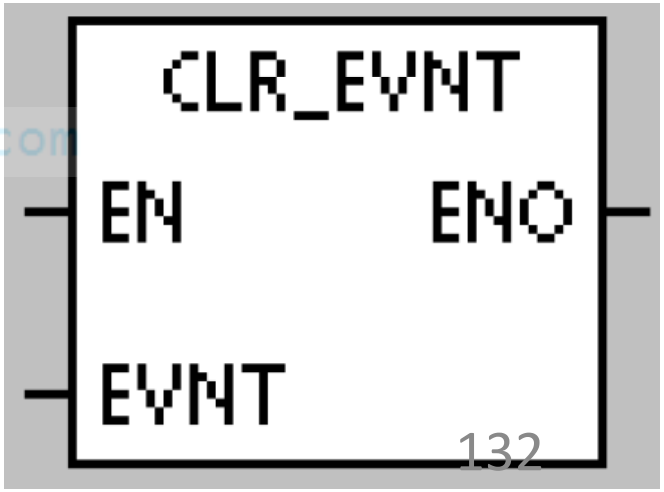
```
//INT     0
Network 1
LD              SM0.0
XORD            AC0, AC0
MOVW            AIW0, AC0          // đọc mực nước
DTR             AC0, AC0
/ R             32000.0, AC0       // chuyển sang số thực
MOVR            AC0, VD100         // đưa vào bảng
Network 2
LD              I0.0               // Khi I0.0 ON thì điều khiển PID
PID             VB100, 0
Network 3
LD              SM0.0
MOVR            VD108, AC0         // Xuất ra điều khiển
*R              32000.0, AC0
TRUNC           AC0, AC0
MOVW            AC0, AQW0
```

Lệnh Ngắt

- S7-200 cho phép tối đa 128 chương trình ngắt.
- Chương trình ngắt được đóng khung bằng **INT_n** và **RETI**
- Cho phép ngắt và cấm ngắt bằng ENI và DISI
- Ngắt được thực hiện khi có sự kiện xảy ra
- Liên kết và phá liên kết giữa sự kiện và ngắt bằng ATCH và DTCH; xóa sự kiện bằng CLR_EVNT







Lệnh Ngắt

Sự kiện	Mô tả	Nhóm ưu tiên	Ưu tiên trong nhóm
8	Nhận ký tự ở Port 0	Cao nhất	1
9	Truyền xong Port 0		1
23	Nhận xong bảng tin ở Port 0		1
24	Nhận xong bảng tin ở Port 1		2
25	Nhận ký tự ở Port 1		2
26	Truyền xong Port 1		2

Lệnh Ngắt

Sự kiện	Mô tả	Nhóm ưu tiên	Ưu tiên trong nhóm
0	Ngắt cạnh lên I0.0	Giữa	1
2	Ngắt cạnh lên I0.1		2
4	Ngắt cạnh lên I0.2		3
6	Ngắt cạnh lên I0.3		4
1	Ngắt cạnh xuống I0.0		5
3	Ngắt cạnh xuống I0.1		6
5	Ngắt cạnh xuống I0.2		7
7	Ngắt cạnh xuống I0.3		8
12	Đếm tốc độ cao HSC0: đo bằng đặt		1
13	Đếm tốc độ cao HSC1: đo bằng đặt		9
14	HSC1 đổi hướng đếm		10
15	Xóa ngoài HSC1		11

Lệnh Ngắt

Sự kiện	Mô tả	Nhóm ưu tiên	Ưu tiên trong nhóm
16	Đếm tốc độ cao HSC2: đo bằng đặt	Giữa	12
17	HSC2 đổi hướng đếm		13
18	Xóa ngoài HSC2		14
32	Đếm tốc độ cao HSC3: đo bằng đặt		19
29	Đếm tốc độ cao HSC4: đo bằng đặt		20
30	HSC4 đổi hướng đếm		21
31	Xóa ngoài HSC4		22
33	Đếm tốc độ cao HSC5: đo bằng đặt		23
19	Đếm xung PLS0 xong		15
20	Đếm xung PLS1 xong		16

Lệnh Ngắt

Sự kiện	Mô tả	Nhóm ưu tiên	Ưu tiên trong nhóm
10	Ngắt thời gian 0	Thấp nhất	1
11	Ngắt thời gian 1		2
21	Ngắt timer T32		3
22	Ngắt timer T96		4

Ngắt thời gian

- Ngắt thời gian 0/1 xảy ra theo chu kỳ cố định (tối đa 255ms) bởi nội dung của SMB34/SMB35 (đơn vị ms)
- Thường dùng để đọc hay xuất tín hiệu analog.
- Ngắt timer T32/T96 xảy ra khi T32 hoặc T96 hoàn tất thời gian trễ đã đặt.

Ngắt thời gian

Ví dụ:

NETWORK 1 ***// Subroutine 0***

LD SM0.0

MOVB 100, SMB34 *// Chu kỳ ngắt100ms*

ATCH INT_0 10 *// Liên kết sự kiện 10 với INT_0*

ENI *// Cho phép ngắt toàn cục*

NETWORK 1 // Interrupt 0

// Đọc giá trị của AIW4 sau mỗi 100ms

LD SM0.0

MOVW AIW4, VW100

Ngắt tần số cao, điều rộng xung

Lệnh PLS:

- **Chế độ PTO (Pulse Train Output): Phát dãy xung vuông tần số thay đổi, số xung thay đổi**
- **Chế độ PWM (Pulse Width Modulation): phát dãy xung điều rộng PWM có chu kỳ thay đổi**
- **Ngõ ra Q0.0 và Q0.1.**

Ngắt tần số cao, điều rộng xung

Thanh ghi báo trạng thái

SMB66/SMB76

Q0.0	Q0.1	Thanh ghi trạng thái		
			0	1
SM66.4	SM76.4	PTO profile aborted (delta calculation error)	No error	aborted
SM66.5	SM76.5	PTO profile aborted due to user command	No error	Aborted
SM66.6	SM76.6	PTO/PWM pipeline overflow/underflow:	No error	Over/underflow
SM66.7	SM76.7	PTO idle:	In progress	PTO idde

Ngắt tần số cao, điều rộng xung

Thanh ghi điều khiển SMB67/SMB77

Q0.0	Q0.1	Thanh ghi điều khiển		
			0	1
SM67.0	SM77.0	PTO/PWM update cycle time	No update	update
SM67.1	SM77.1	PWM update Pulse width time	No update	update
SM67.2	SM77.2	PTO update the pulse count value	No update	update
SM67.3	SM77.3	PTO/PWM time base	1us/tick	1ms/tick
SM67.4	SM77.4	PWM update method	asynchron ous	Sychrono us
SM67.5	SM77.5	PTO single/multiple segment operation	Single	Multiple
SM67.6	**SM77.6**	**PTO/PWM mode select**	**PTO**	**PWM**
SM67.7	**SM77.7**	**PTO/PWM enable**	**disable**	**enable**

Ngắt tần số cao, điều rộng xung

Các Thanh ghi khác

Q0.0	Q0.1	Chức năng	
SMW68	SMW78	PTO/PWM cycle time value	range: 2 to 65,535
SMW70	SMW80	PWM pulse width value	range: 0 to 65,535
SMW72	SMW82	PTO pulse count value	range: 1 to 4,294,967,295
SMW166	SMW176	Number of the segment in progress	Multiple-segment PTO operation only
SMW168	SMW178	Starting location of the profile table Multiple-segment PTO operation only (byte offset from V0)	Multiple-segment PTO operation only
SMB170	SMB180	Linear profile status byte	
SMB171	SMB181	Linear profile result register	
SMB172	SMB182	Manual mode frequency register	

Ngắt tần số cao, điều rộng xung

Một số trường hợp thiết lập thanh ghi điều khiển

Control Register	Result of Executing the PLS Instruction					
	Enable	Mode	Segment Operation	Time base	Pulse Count	Cycle Time
16#81	Yes	PTO	Single	1us/cycle		Load
16#84	Yes	PTO	Single	1us/cycle	Load	
16#85	Yes	PTO	Single	1us/cycle	Load	Load
16#89	Yes	PTO	Single	1ms/cycle		Load
16#8C	Yes	PTO	Single	1ms/cycle	Load	
16#8D	Yes	PTO	Single	1ms/cycle	Load	Load
16#A0	Yes	PTO	Multiple	1us/cycle		
16#A8	Yes	PTO	Multiple	1ms/cycle		

Ngắt tần số cao, điều rộng xung

Một số trường hợp thiết lập thanh ghi điều khiển

Control Register	Result of Executing the PLS Instruction					
	Enable	Mode	PWM Update Mode	Time base	Pulse Width	Cycle Time
16#D1	Yes	PWM	Single	1us/cycle		Load
16#D2	Yes	PWM	Single	1us/cycle	Load	
16#D3	Yes	PWM	Single	1us/cycle	Load	Load
16#D9	Yes	PWM	Single	1ms/cycle		Load
16#DA	Yes	PWM	Single	1ms/cycle	Load	
16#DB	Yes	PWM	Single	1ms/cycle	Load	Load

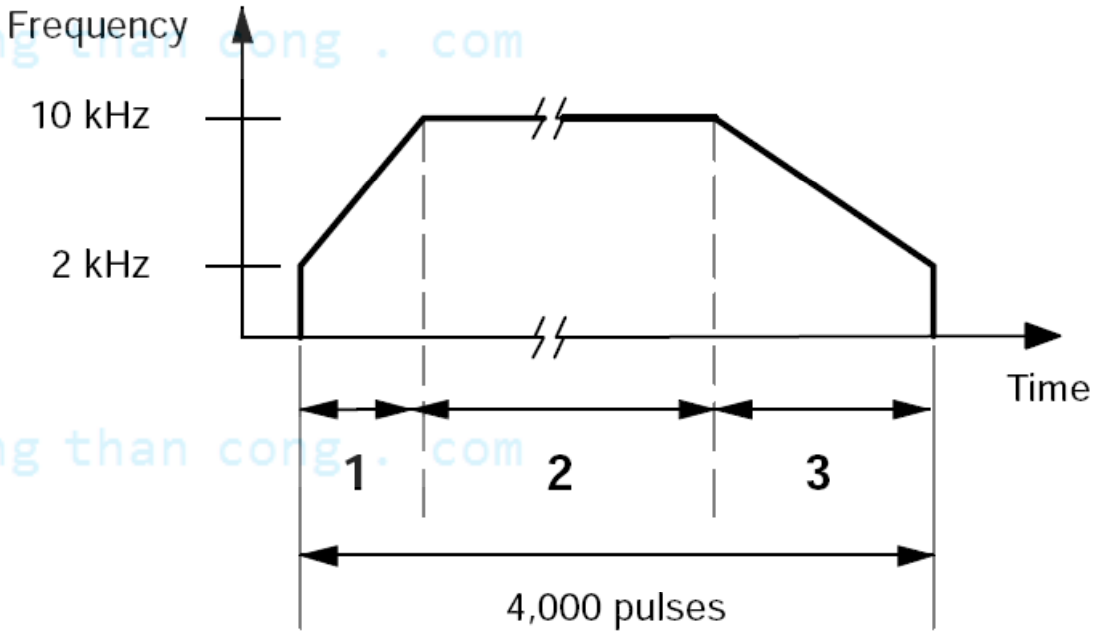
Ngắt tần số cao, điều rộng xung

Calculating Profile Table Values

Multi-segment PTO có thể ứng dụng trong nhiều trường hợp, cụ thể là điều khiển step motor

Có thể có tối đa 255 segment

Delta cycle time = (End_Ctseg - Init_CTseg) / Quantityseg



1 Segment #1 200 pulses
2 Segment #2 3400 pulses
3 Segment #3 400 pulses

Ngắt tần số cao, điều rộng xung

Calculating Profile Table Values

Ví dụ

Tần số bắt đầu và cuối là 2 kHz, tần số cực đại là 10kHz, cần 4000 xung để hoàn thành chu kỳ hoạ

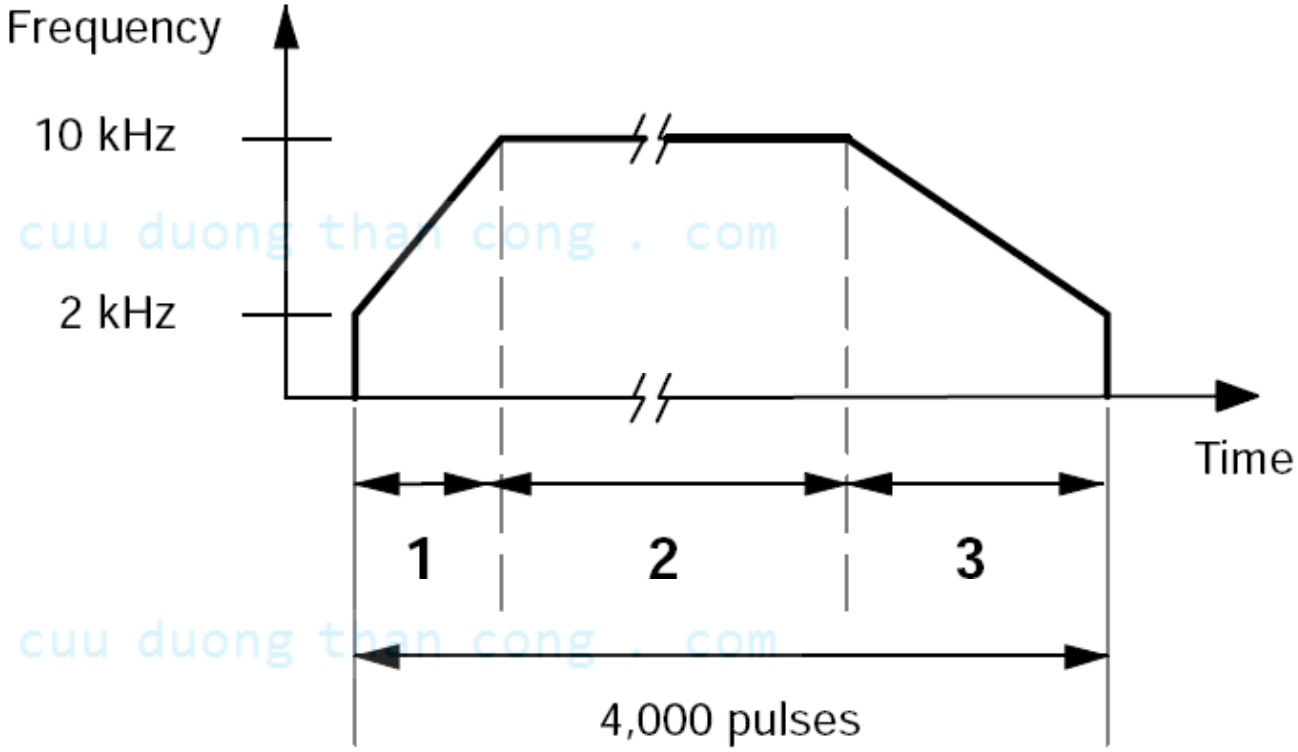
Giá trị trong profile ta
sang chu kỳ:

Chu kỳ hoạt động lúc

Chu kỳ hoạt động ở tầ

Trong quá trình giảm

Trong quá trình giảm



1 Segment #1
200 pulses

2 Segment #2
3400 pulses

3 Segment #3
400 pulses

Ngắt tần số cao, điều rộng xung

Cal

Ví d

Tần

hoàn

Giá

sang

Chu

Chu

Tron

Tron

Address	Value	Description	
VB500	3	Total number of segments	
VW501	500	Initial cycle time	Segment 1
VW503	-2	Initial delta cycle time	
VD505	200	Number of pulses	
VW509	100	Initial cycle time	Segment 2
VW511	0	Delta cycle time	
VD513	3400	Number of pulses	
VW517	100	Initial cycle time	Segment 3
VW519	1	Delta cycle time	
VD521	400	Number of pulses	

để

Ngắt tần số cao, điều rộng xung