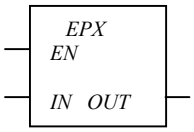
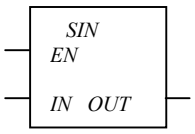
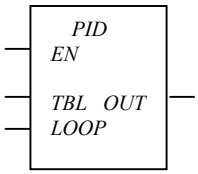
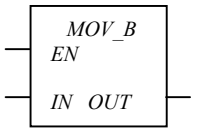
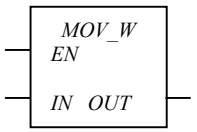
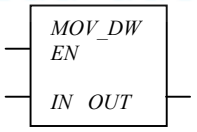
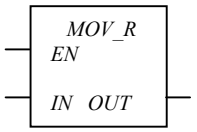
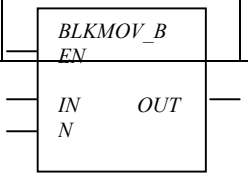
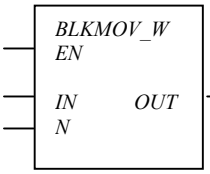
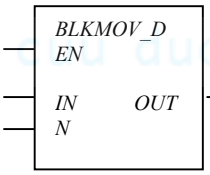
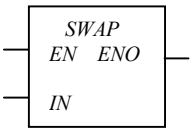
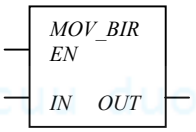
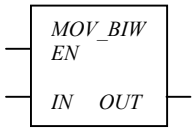
6. SIMATIC Numerical Function Instructions:

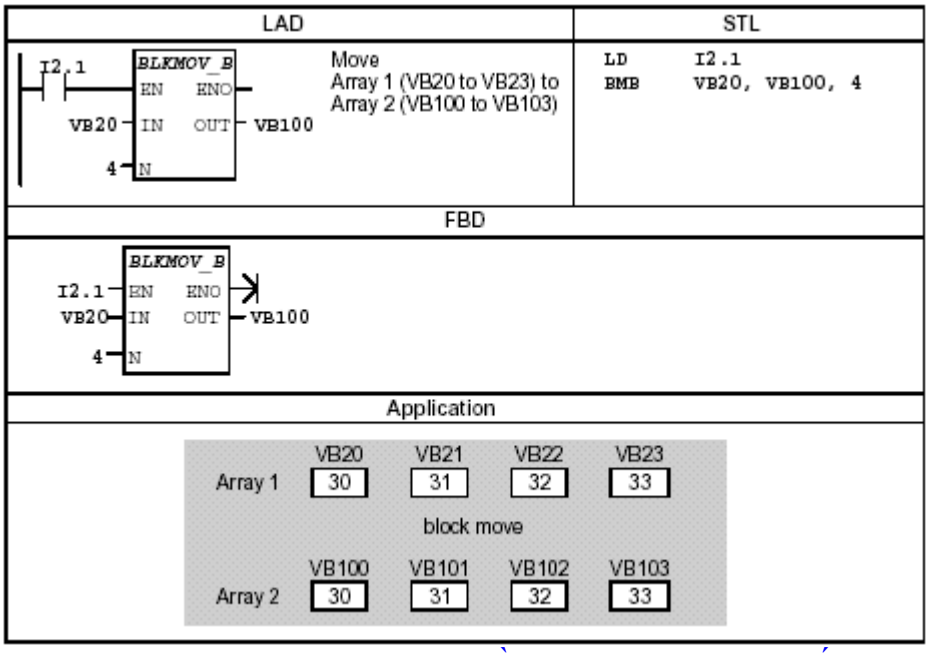
STL	LAD	Mô tả (Description)	Toán hạng (Operands)	Kiểu dữ liệu (Data Types)
Square Root				
SQRT IN, OUT		Lệnh thực hiện phép lấy căn bậc hai của số thực 32 bit. Kết quả cũng là số 32 bit được ghi vào từ kép OUT.	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	Real
Natural Logarithm (logarit tự nhiên)				
LN IN, OUT		Lệnh Natural Logarith thực hiện phép logirit tự nhiên của số thực 32 bit, Kết quả được lưu vào từ kép OUT. Lệnh này cũng được sử dụng để thực hiện phép logarit cơ số 10 từ phép lấy logarit tự nhiên.	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	Real
Natural Exponential (phép lấy tự nhiên)				
EPX IN, OUT			**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	Real

<table>
<tr><th colspan="5">Sine, Cosine and Tangent</th></tr>
<tr><td>SIN IN, OUT</td><td rowspan="3"> 
</td><td rowspan="3">Lệnh Sine, Cosine và Tangent định giá trị hàm lượng giác của góc IN (số thực 32 bit). Kết quả được lưu vào doubleword OUT. Với điều kiện: IN tính bằng radian, nếu là độ thì phải thực hiện phép chuyển từ độ sang radian bằng cách thực hiện lệnh MUL_R để nhân giá trị IN Với 1.745329E-2 (π/180)</td><td rowspan="3">IN: ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD
OUT: ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD</td><td rowspan="3">Real</td></tr>
<tr><td>COS IN, OUT</td></tr>
<tr><td>TAN IN, OUT</td></tr>
<tr><td rowspan="2">PID TBL, LOOP</td><td rowspan="2"> 
</td><td rowspan="2">Lệnh thực hiện tính toán vòng lặp, với số thứ tự là LOOP (0<=LOOP<=7) và bảng tham chiếu của quá trình là TBL.
! Trước khi thực hiện quá trình tính toán vòng lặp PID này cần phải thực hiện một số thủ tục quy định trước khi quá trình tính toán diễn ra như: việc khai báo tham số của hàm, địa chỉ của mảng dữ liệu, lấy mẫu tín hiệu vào analog đầu vào, thực hiện quá trình tính toán, chuẩn hoá, hiệu chỉnh... Phần này sẽ được trình bày cụ thể ở chương sau.</td><td>TBL:
VB</td><td>BYTE</td></tr>
<tr><td>LOOP:
Constant (0 ÷7)</td><td>BYTE</td></tr>
</table>

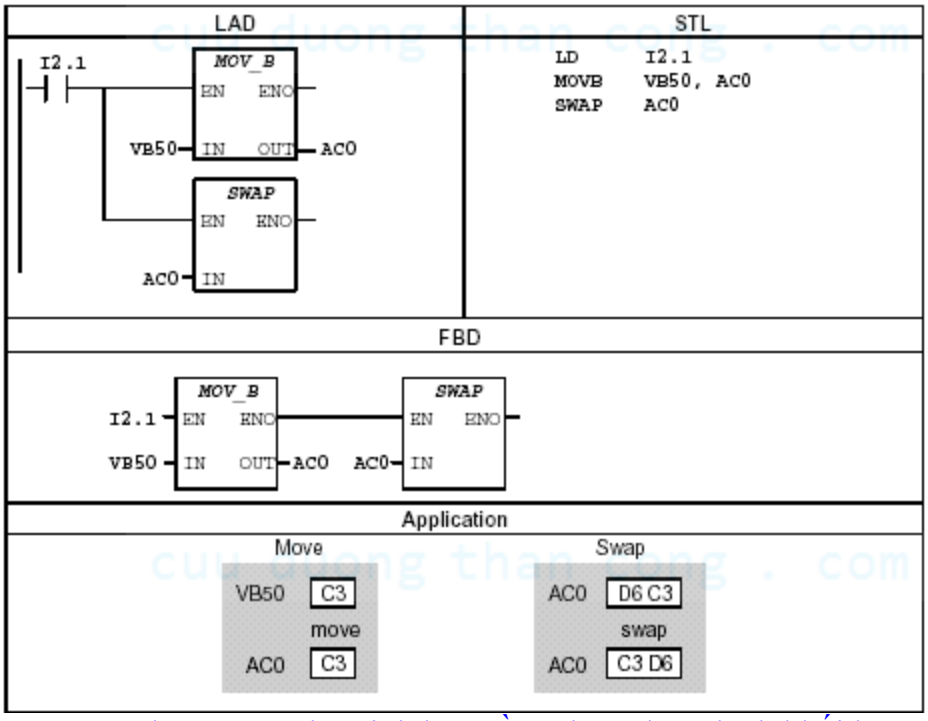
7. SIMATIC Move Instructions:

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Move Byte, Move Word, Move Double Word and Move Real				
MOVB IN, OUT		Lệnh thực hiện việc chuyển dữ liệu từ byte IN vào byte OUT khi có sườn lên của tín hiệu vào.	**IN:** IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC,* LD **OUT:**IB, QB, MB, VB, SMB, SB, LB, AC, *VD, *AC,* LD	Byte
MOVW IN,OUT		Lệnh thực hiện việc chuyển dữ liệu từ Word IN vào Word OUT khi có sườn lên của tín hiệu vào.	**IN:** IW, QW, VW, LW, SW, AIW, T, C, AC, Constant, *VD, *AC, *LD **OUT:** IW, QW, MW, SMW, VW, LW, SW, AIW, T, C, AC, *VD, *AC, *LD	Word, INT
MOVD IN, OUT		Lệnh thực hiện việc chuyển dữ liệu từ kép IN vào từ kép OUT khi có sườn lên của tín hiệu vào.	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, &VB, &IB, &QB, &SB, &MB, &T, &C, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	DoubleWord, DINT
MOVR IN, OUT		Lệnh thực hiện việc chuyển dữ liệu là số thực từ từ kép IN vào từ kép OUT khi có sườn lên của tín hiệu vào.	**IN:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, MD, VD, SMD, SD, LD, HC,AC, *VD, *AC, *LD	Real
Block Move Byte, Block Move Word, Block Move Double Word and Block Move Real				
BMB IN, OUT, N		Lệnh thực hiện việc chuyển N byte dữ liệu tính từ	**IN, OUT:** IB, QB, MB, VB, SMB, SB, LB, *VD, *AC, *LD.	Byte

		byte IN vào vùng địa chỉ tính từ byte OUT khi có sườn lên của tín hiệu vào.	**N:** IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD 1 <= N <= 255	Byte
BMW IN, OUT, N		Lệnh thực hiện việc chuyển N từ đơn dữ liệu tính từ từ đơn IN vào vùng địa chỉ tính từ từ đơn OUT khi có sườn lên của tín hiệu vào.	**IN:** IW, QW, VW, LW, SW, SMW, AIW, T, C, AC, *VD, *AC, *LD **OUT:** IW, QW, VW, LW, SW, SMW, AQW, T, C, AC, *VD, *AC, *LD	Word
			N: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD 1 <= N <= 255	Byte
BMD IN, OUT, N		Lệnh thực hiện việc chuyển N từ kép dữ liệu tính từ từ kép IN vào vùng địa chỉ tính từ từ kép OUT khi có sườn lên của tín hiệu vào.	**IN, OUT:** ID, QD, MD, VD, SMD, SD, LD, *VD, *AC, *LD.	DWord
			N: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD 1 <= N <= 255	Byte
Swap Byte				
SWAP IN		Lệnh đảo dữ liệu của 2 byte trong từ đơn IN.	**IN:** IW, QW, VW, LW, SW, SMW, AIW, T, C, AC.	Word
Move Byte Immedieate Read/ Write				
BIR IN, OUT		Lệnh đọc tức thời giá trị ở byte đầu vào ở cổng vật lý IN và ghi trực tiếp vào byte OUT.	**IN:** IB **OUT:** IB, QB, MB, VB, SMB, SB, LB, AC, *VD, *AC, *LD	Byte
BIW IN, OUT		Lệnh đọc tức thời giá trị ở byte IN và ghi trực tiếp ra đầu ra ở cổng vật lý byte OUT.	**IN:** IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD **OUT:** QB	Byte



Hình 3.31: Ví dụ minh hoạ về cách sử dụng lệnh khối hàm



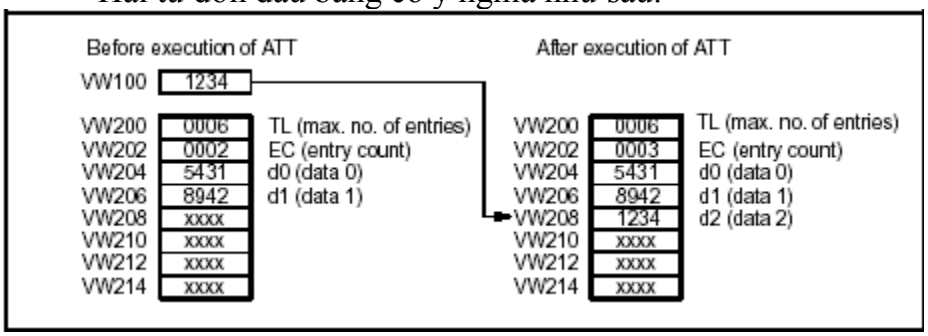
Hình 3.32: Ví dụ minh hoạ về cách sử dụng lệnh khối hàm

8. SIMATIC Table Instructions:

Các lệnh làm việc với bảng dữ liệu gọi tắc là lệnh bảng, cho phép nhập dữ liệu vào một bảng, sắp xếp số lượng theo thứ tự đã được nhập vào hoặc theo thứ tự ngược lại.

Bảng được định nghĩa là một mảng từ đơn xếp liền nhau từ địa chỉ thấp nhất tính từ đầu bảng đến địa chỉ cao nhất tính đến cuối bảng. Hai từ đơn đầu tiên của bảng dùng để quản lý bảng. Dữ liệu được ghi vào trong bảng bắt đầu từ từ đơn thứ 3 trong bảng, mỗi dữ liệu chiếm một từ đơn, một bảng chỉ chứa tối đa 100 dữ liệu. Có nghĩa là bảng lớn nhất có 204 byte.

Hai từ đơn đầu bảng có ý nghĩa như sau:



Hình 3.33: Mô tả bảng dữ liệu

+ Từ đầu ký hiệu bằng TL, chứa kích thước của bảng không kể hai từ đơn quản lý.

+ Từ đơn thứ hai ký hiệu bằng EC, để quản lý số các dữ liệu hiện có trong bảng.

Bit SM1.4 được dùng để báo trạng thái đầy bảng.

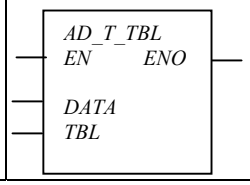
Các lệnh làm việc với bảng gồm có các lệnh:

+ Nhập thêm dữ liệu vào bảng: ATT - Add to Table (AT_T_TBL).

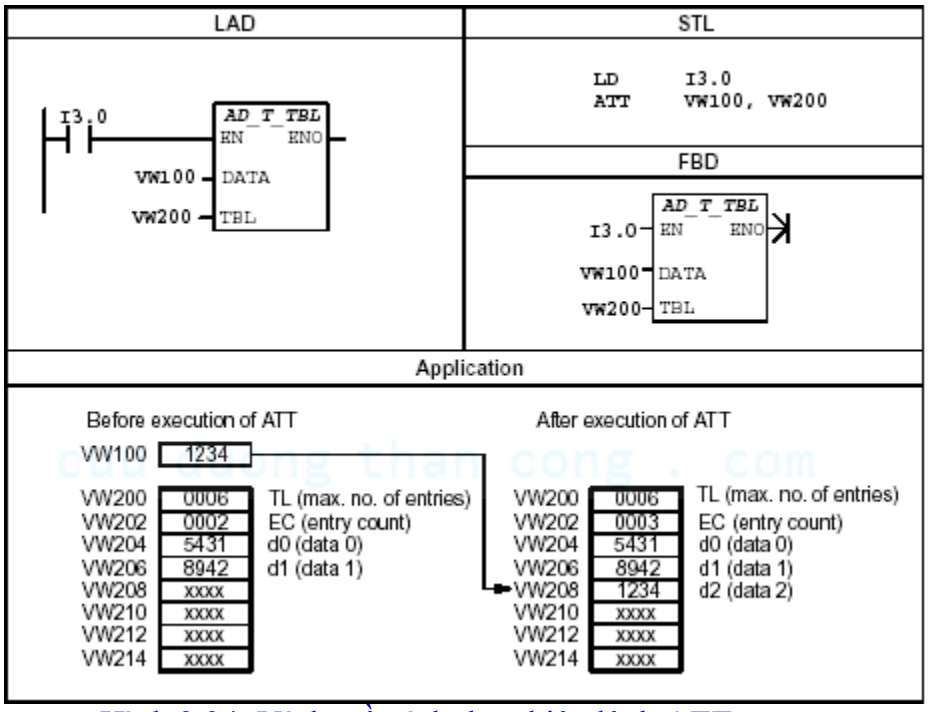
+ Lấy dữ liệu ra khỏi bảng theo thứ tự vào trước ra trước: First - In - First - Out (FIFO).

+ Lấy dữ liệu ra khỏi bảng theo thứ tự vào sau ra trước: Last - In - First - Out (LIFO).

Tip: *Lệnh bảng được thực hiện liên tục (một từ trong một vòng quét) khi đầu vào vẫn còn được kích. Bởi vậy trước khi gọi lệnh làm việc với bảng nên thực hiện lệnh phát hiện sườn lên (EU) cho tín hiệu đầu vào.*

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Add to Table				
ATT DATA, TABLE		Lệnh ghi thêm vào bảng một dữ liệu kiểu từ đơn, được xác định bằng nội dung của toán hạng DATA trong lệnh. Bảng được chỉ định trong lệnh bằng toán hạng TBL xác định từ đầu tiên của bảng, tức là TL. Nếu	**DATA:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT

		tiên của bảng, tức là TL. Nếu bảng đã đầy tức là EC=TL, Bit SM1.4=1. Dữ liệu mới được đưa vào sẽ nằm trong từ chưa dùng đầu tiên, tức là ngay sau dữ liệu được nhập trước đó. Khi lệnh thực hiên xong thì nội dung của từ EC tăng thêm 1 đơn vị.	**TBL:** IW, QW, VW, LW, SW, MW, SMW, T, C, *VD, *AC, *LD	Word



Hình 3.34: Ví dụ về cách thực hiện lệnh ATT

Sử dụng lệnh tìm kiếm để tìm dữ liệu theo mẫu cho trước trong một bảng. Mẫu dữ liệu định trước là nội dung của toán hạng PTN của lệnh. Tham số CMD là luật tìm kiếm, có 4 luật tìm kiếm: =, <>, <, >.

Bảng được chỉ định trong lệnh tìm kiếm được chỉ định bằng nội dung của toán hạng TBL chỉ ô nhớ nằm ngay trước vùng chứa dữ liệu của bảng (ô này chính là ô từ đơn EC).

Bảng quy định cho lệnh tìm kiếm bao gồm bộ đếm EC tức thời có kiểu từ đơn ghi số các dữ liệu có trong bảng và vùng dữ liệu của bảng. Số lượng lớn nhất các dữ liệu của bảng có thể có của bảng là 100.

Mỗi dữ liệu trong bảng có kích thước bằng từ đơn. Dữ liệu trong bảng được đánh số từ 0÷n với n có giá trị cực đại bằng 99. Số các dữ liệu có trong bảng là nội dung của từ đơn EC, ***không bắt buộc lệnh tìm kiếm phải bắt đầu từ đầu bảng***. Lệnh có thể bắt đầu công việc tìm kiếm tại một điểm bất kỳ trong vùng dữ liệu. Toán hạng INDX xác định điểm xuất phát của công việc tìm kiếm bằng việc chỉ ra chỉ số (0÷99) của dữ liệu đầu tiên trong

vùng định tìm kiếm. Như vậy muốn tìm từ đầu bảng INDX phải có giá trị bằng 0. Nội dung của INDX là số nguyên trong khoảng từ 0 đến EC.

Nếu sử dụng lệnh tìm kiếm với bảng được tạo bởi các lệnh ATT, FIFO, LIFO thì ô nhớ EC là ô nhớ đầu bảng phải được chỉ định trong lệnh tại toán hạng TBL. Khi sử dụng lệnh ATT, FIFO, LIFO đòi hỏi phải thông báo từ số các đầu vào cực đại cho lệnh (ô nhớ TL) còn khi sử dụng lệnh tìm kiếm TBL_FIND thì không cần. Toán hạng SRC của lệnh tìm kiếm là tên của ô nhớ EC (2 byte).

Cú pháp của lệnh tìm kiếm trong LAD và STL khác nhau. Trong khi cả 4 luật tìm kiếm CMD trong LAD, thì trong STL tương ứng với mỗi luật tìm kiếm có 1 lệnh tìm kiếm riêng. Như vậy trong LAD chỉ có 1 hộp cho 4 lệnh tìm kiếm thì trong STL là: FND=, FND<>, FND<, FND>.

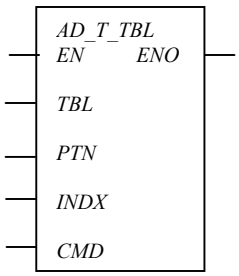
Nội dung của toán hạng trong LAD được quy định như sau:

a) CMD = 1, tìm theo luật = (bằng nhau.).

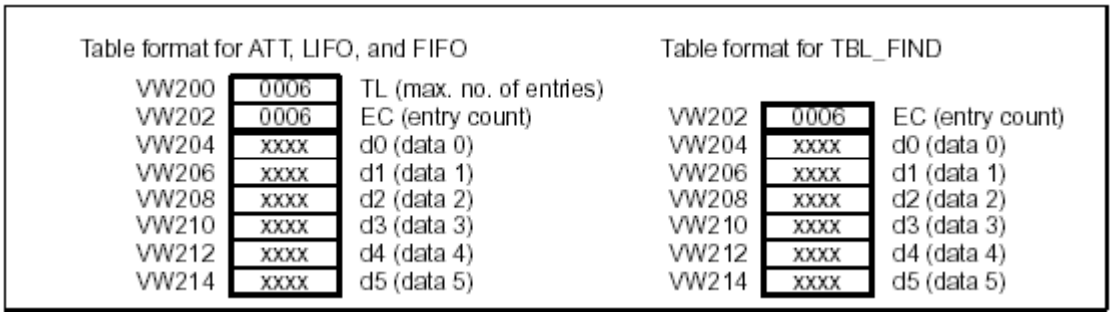
b) CMD = 2, tìm theo luật <> (khác nhau).

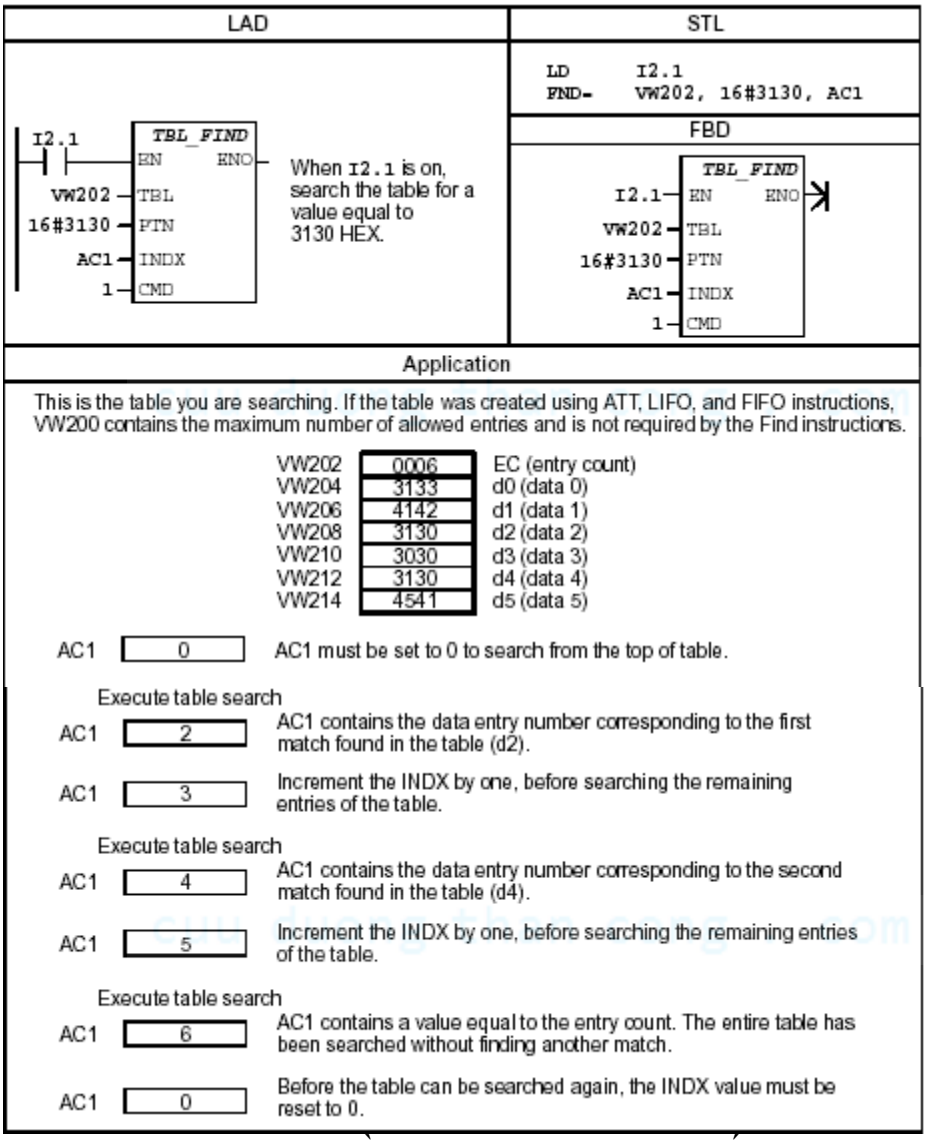
c) CMD = 3, tìm theo luật < (nhỏ hơn).

d) CMD = 4, tìm theo luật > (lớn hơn).

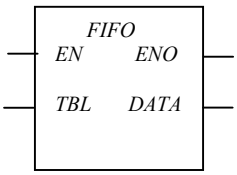
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Table Fine				
FND= TBL, PARNT, INDX FND<> TBL, PARNT, INDX FND< TBL, PARNT, INDX FND> TBL, PARNT, INDX		Thực hiện việc tìm kiếm trong bảng xác định bởi TBL , bắt đầu từ vị trí dữ liệu INDX ô nhớ chứ dữ liệu PARNT. Luật tìm kiếm được quy định bởi CMD có giá trị từ 1 đến 4 tương ứng =, <>, <, >. Khi tìm thấy , INDX sẽ chỉ vào ô dữ liệu đầu tiên tìm được trong bảng và lệnh được kết thúc. Do đó để tìm kiếm dữ liệu tiếp theo, INDX phải được tăng giá trị l và gọi lại lệnh này. Nếu như không tìm thấy INDX có giá trị đúng bằng giá trị của bộ đếm EC.	**TBL:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, *VD, *AC, *LD	Word
			PTN: IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AC, Constant, *VD, *AC, *LD	INT
			INDX: LW, SW, MW, SMW, AIW, T, C, *VD, *AC, *LD	Word
			CMD: Constant	Byte

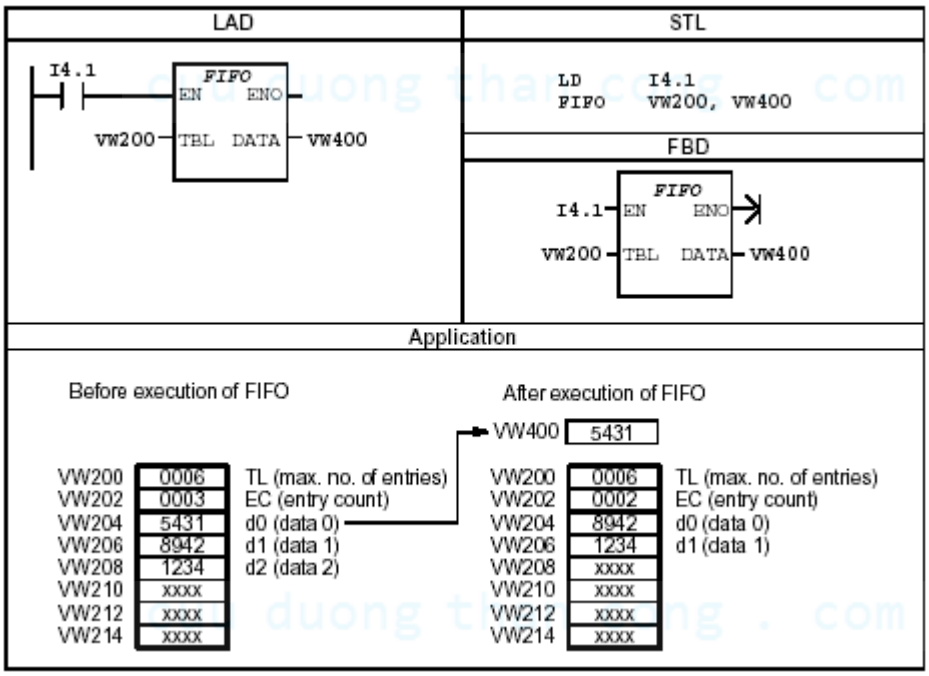
Bảng 3.5: Sự khác nhau giữa bảng dữ liệu định nghĩa bằng lệnh ATT, FIFO, LIFO và lệnh FIN





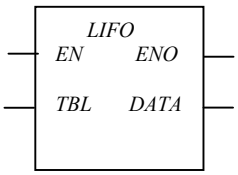
Hình 3.35: Ví dụ về cách sử dụng lệnh tìm kiếm FND

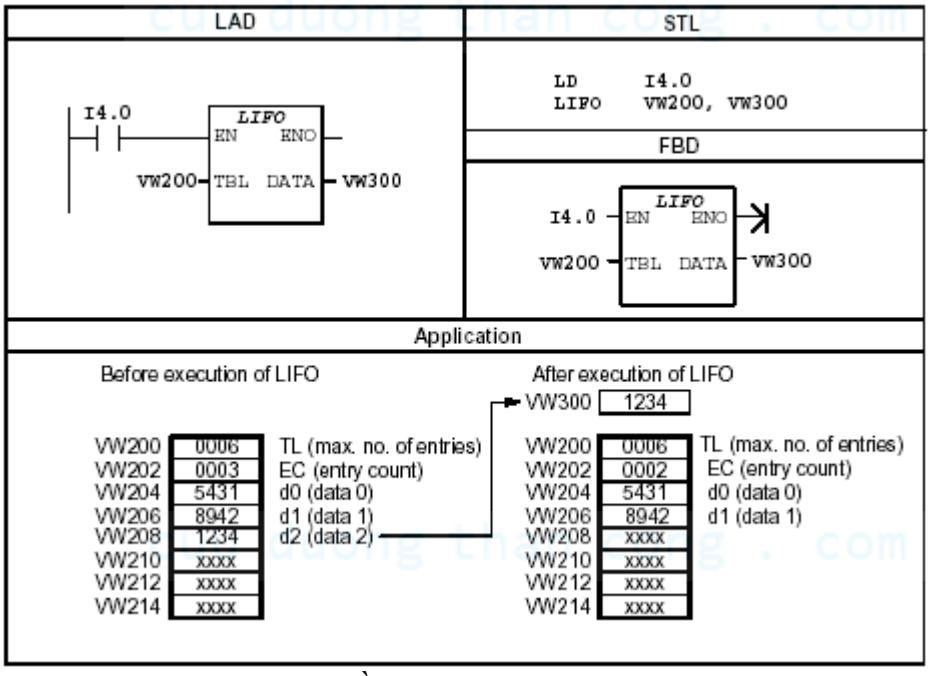
STL	LAD	Mô tả (Description)	Toán hạng (Operands)	Kiểu dữ liệu (Data Types)
Fisrt - In - Fisrf - Out				
FIFO TABLE, DATA		Lệnh lấy dữ liệu đầu tiên của bảng ra khỏi bảng. Nếu bảng đã trống có nghĩa là dữ liệu trong đó được lấy ra hết, hay EC=0, bit SM1.4=1. Dữ liệu lấy ra được ghi vào DATA (kiểu từ). Các dữ liệu còn lại được dồn lên vị trí trên để lấp chỗ trống vừa mới bị lấy đi. Khi lệnh thực hiện xong nội dung của EC giảm đi một đơn vị.	**TBL:** IW, QW, VW, LW, SW, MW, T, C, *VD, *AC, *LD	INT
			DATA: IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AQW, *VD, *AC, *LD	Word



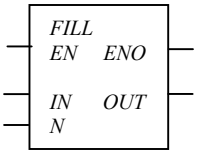
Hình 3.36: Ví dụ về cách sử dụng lệnh FIFO

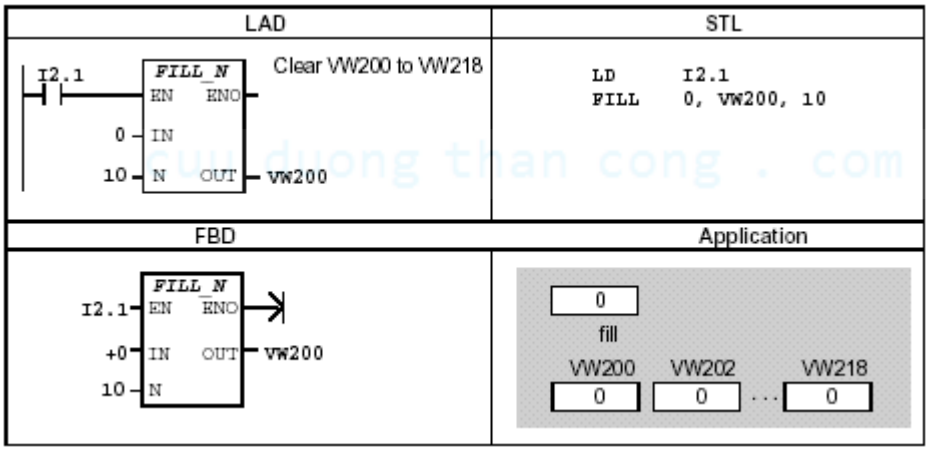
.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Last - In - Fisrf - Out				
LIFO TABLE, DATA		Lệnh lấy dữ liệu cuối cùng của bảng ra khỏi bảng tức là dữ liệu được nhập sau cùng. Nếu bảng đã trống có nghĩa là dữ liệu trong đó được lấy ra hết, hay EC=0, bit SM1.4=1. Dữ liệu lấy ra đượ ghi vào DATA (kiểu từ). Các dữ liệu còn lại được dồn lên vị trí trên để lấp chỗ trống vừa mới bị lấy đi. Khi lệnh thực hiện xong nội dung của EC giảm đi một đơn vị.	**TBL:** IW, QW, VW, LW, SW, MW, T, C, *VD, *AC, *LD	INT
			DATA: IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AQW, *VD, *AC, *LD	Word



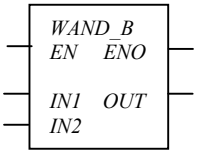
Hình 3.37: Ví dụ về cách sử dụng lệnh LIFO

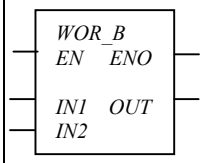
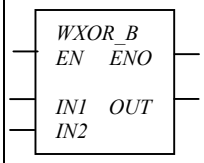
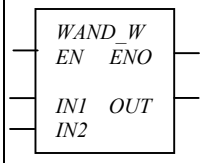
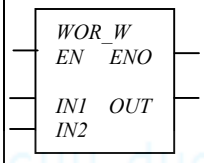
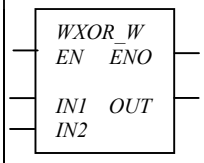
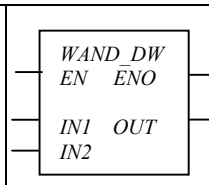
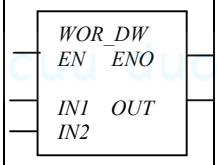
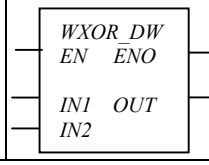
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Memory Fill				
FILL IN, OUT, N		Lệnh điền giá trị chứa trong Word IN vào mảng bắt đầu từ địa chỉ Word OUT. N là số từ đơn của mảng, 1<=N<=255	**IN:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AC, Constant, *VD, *AC, *LD	Word
			N: IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	Byte
			OUT: IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, *VD, *AC, *LD	Word

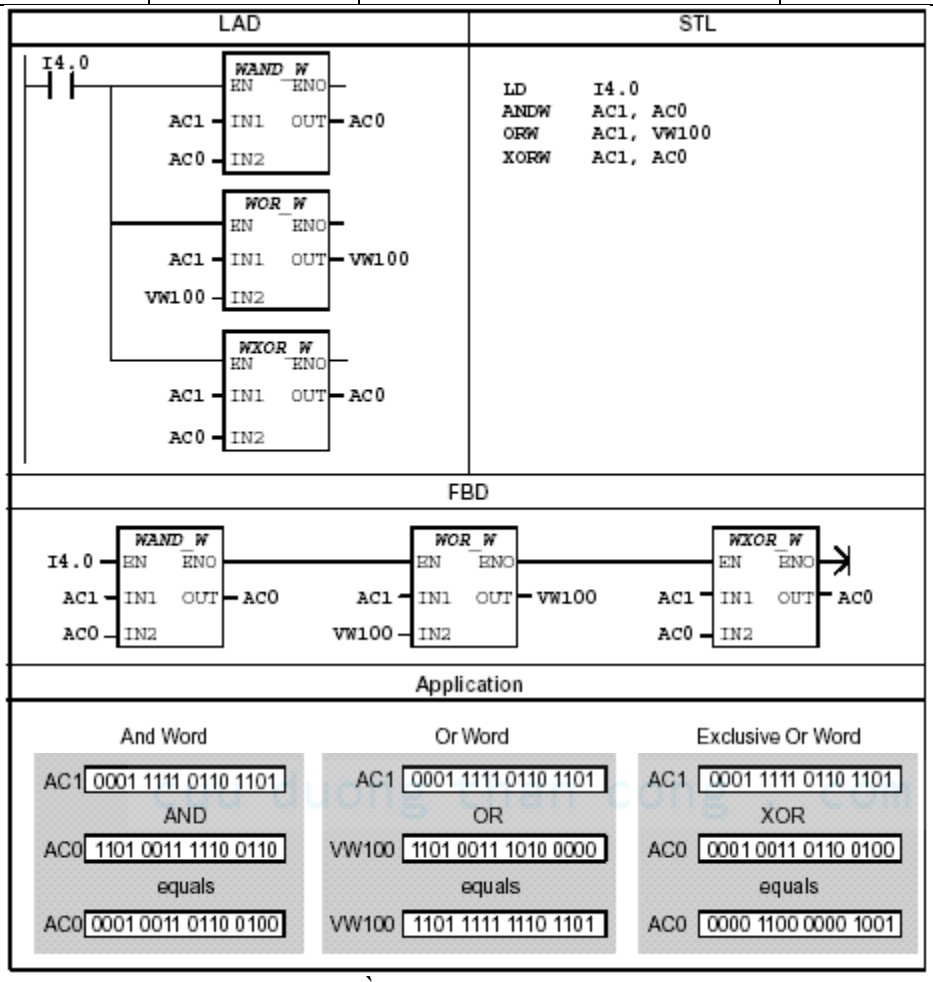


Hình 3.38: Ví dụ về cách sử dụng lệnh FILL

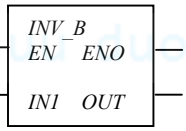
9. SIMATIC Logical Operation Instructions:

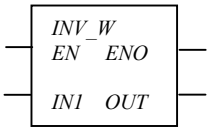
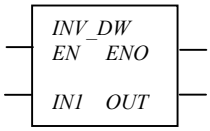
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
And Byte, Or Byte, Exclusive Or Byte				
ANDB IN1, OUT		Lệnh thực hiện AND giữa các bit tương ứng của hai Byte IN1 và IN2, kết quả ghi vào Byte OUT.	**IN1, IN2:** IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	Byte

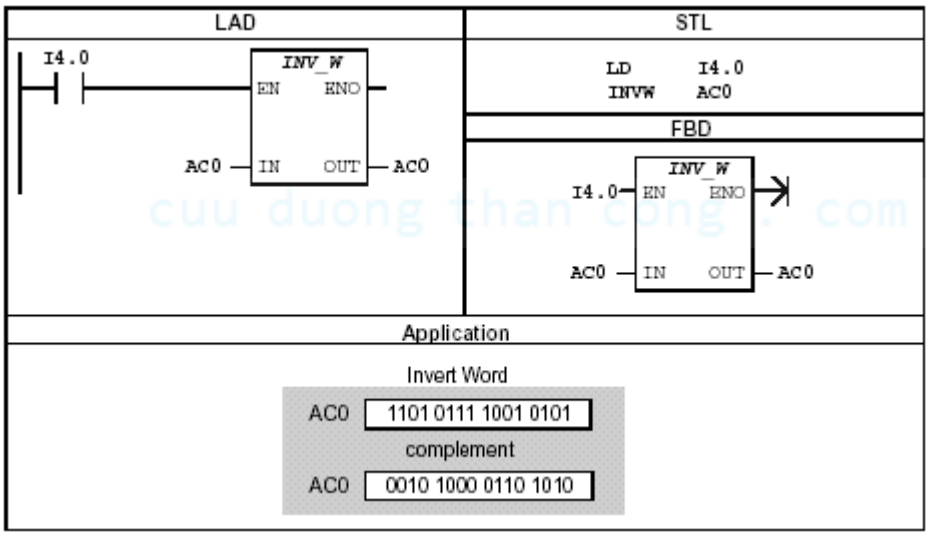
ORB IN1, OUT		Lệnh thực hiện OR giữa các bit tương ứng của hai Byte IN1 và IN2, kết quả ghi vào Byte OUT.	**OUT:** IB, QB, MB, VB, SMB, SB, LB, AC, *VD, *AC, *LD	Byte
XORB IN1, OUT		Lệnh thực hiện XOR giữa các bit tương ứng của hai Byte IN1 và IN2, kết quả ghi vào Byte OUT.		
And Word, Or Word, Exclusive Or Word				
ANDW IN1, OUT		Lệnh thực hiện AND giữa các bit tương ứng của hai Word IN1 và IN2, kết quả ghi vào Word OUT.	**IN1, IN2:** IW, QW, VW, LW, SW, MW, SMW, AIW, T, C, AC, Constant, *VD, *AC, *LD	Word
ORW IN1, OUT		Lệnh thực hiện OR giữa các bit tương ứng của hai Word IN1 và IN2, kết quả ghi vào Word OUT.		
XORW IN1, OUT		Lệnh thực hiện XOR giữa các bit tương ứng của hai Word IN1 và IN2, kết quả ghi vào Word OUT.	**OUT:** IW, QW, VW, LW, SW, MW, SMW, T, C, AC, *VD, *AC, *LD	
And DWord, Or DWord, Exclusive Or DWord				
ANDD IN1, OUT		Lệnh thực hiện AND giữa các bit tương ứng của hai từ kép IN1 và IN2, kết quả ghi vào từ kép OUT.	**IN1, IN2:** ID, QD, VD, LD, SD, MD, SMD, HD, AC, Constant, *VD, *AC, *LD	Double Word
ORD 1, OUT		Lệnh thực hiện OR giữa các bit tương ứng của hai từ kép IN1 và IN2, kết quả ghi vào từ kép OUT.		
XORD IN1, OUT		Lệnh thực hiện XOR giữa các bit tương ứng của hai từ kép IN1 và IN2, kết quả ghi vào từ kép OUT.	**OUT:** ID, QD, VD, LD, MD, SMD, AC, *VD, *AC, *LD	



Hình 3.39: Ví dụ về cách sử dụng lệnh AND, OR, XOR

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Invert Byte, Invert Word, Invert DWord				
INVB OUT		Lệnh đảo từng bit của byte đầu vào IN, kết qủa đưa ra đầu ra OUT. Thường thì đầu vào và ra cùng địa chỉ.	**IN:**IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD **OUT:**IB, QB, MB, VB, SMB, SB, LB, AC, Constant, *VD, *AC, *LD	Byte

INVW OUT		Lệnh đảo từng bit của từ đơn đầu vào IN, kết qủa đưa ra đầu ra OUT. Thường thì đầu vào và ra cùng địa chỉ.	**IN:** IW, QW, VW, LW, SW, MW, SMW, AC, AIW , T, C, Constant, *VD, *AC, *LD **OUT:** IW, QW, VW, LW, SW, MW, SMW, AC, T, C, *VD, *AC, *LD	Word
INVD OUT		Lệnh đảo từng bit của từ kép đầu vào IN, kết qủa đưa ra đầu ra OUT. Thường thì đầu vào và ra cùng địa chỉ.	**IN:** ID, QD, VD, LD, SD, MD, SMD, HD, AC, Constant, *VD, *AC, *LD **OUT:** ID, QD, VD, LD, SD, MD, SMD, AC, *VD, *AC, *LD	DWord



Hình 3.40: Ví dụ về cách sử dụng lệnh INVB, INVW, INVD

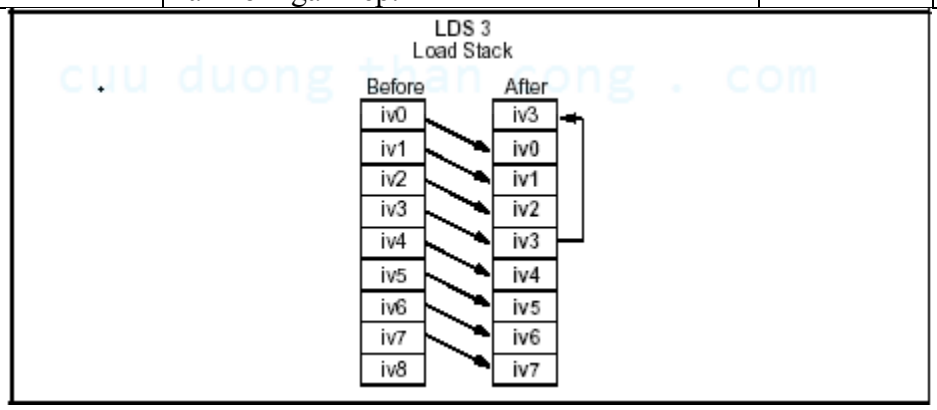
10. SIMATIC Stack Logic Instructions:

Các lệnh tiếp điểm trong đại số Boolean cho phép tạo lập được các mạch logic (không có nhớ). Trong LAD các mạch này biểu diễn thông qua cấu trúc mạch, mắc nối tiếp hay song song các mạch tiếp điểm thường đóng và các tiếp điểm thường mở. STL có thể sử dụng các lệnh A (And) và O (Or) cho các tiếp điểm mắc nối tiếp và song song là thường hở hoặc các lệnh AN (And Not) và ON (Or Not) cho các tiếp điểm mắc nối tiếp và song song là thường đóng. Giá trị của các bit trong ngăn xếp thay đổi tuỳ thuộc vào từng lệnh. Trong phần này chúng ta sẽ đi sâu hơn về sự làm việc của các bit trong ngăn xếp, việc hiểu và nắm bắt về ngăn xếp là điều rất cần thiết trong vấn đề lập trình dùng ngôn ngữ STL.

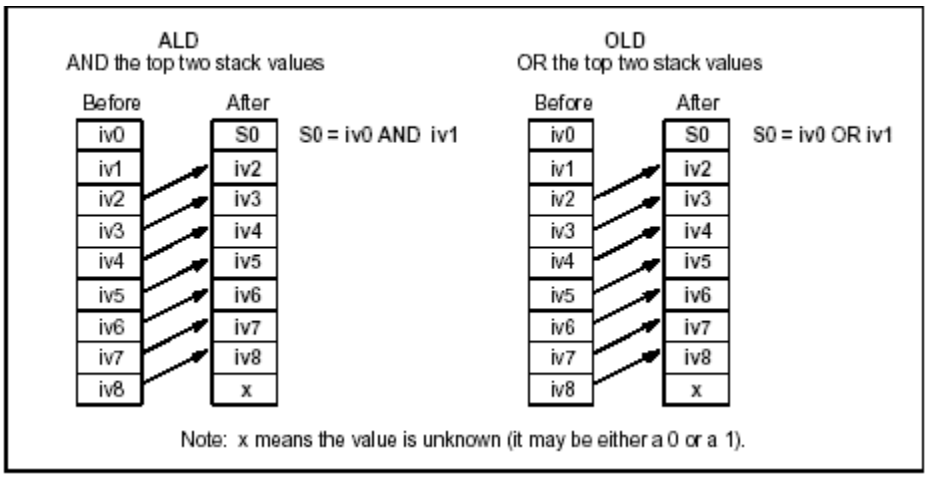
Ngoài những lệnh làm việc trực tiếp với tiếp điểm, S7-200 còn có 5 lệnh đặc biệt biểu diễn các phép tính của đại số Boolean cho các bit trong ngăn xếp, được gọi là các lệnh ***stack logic.*** Trong LAD không dùng những lệnh này. STL sử dụng các lệnh này để

thực hiện những phép toán của phương trình có nhiều biểu thức con. Sau đây là bảng tóm tắt cú pháp và hướng dẫn cách sử dụng lệnh.

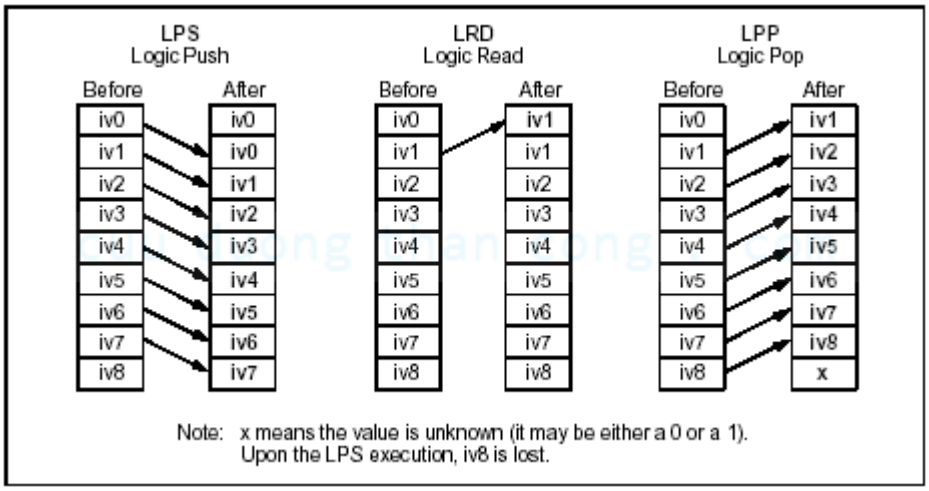
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
And Load				
ALD	none	Lệnh tổ hợp giá trị đầu tiên và giá trị của bit thứ hai trong ngăn xếp bằng phép tính ∧. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp. Giá trị còn lại được kéo lên 1 bit.	none	none
Or LoaD				
OLD	none	Lệnh tổ hợp giá trị đầu tiên và giá trị của bit thứ hai trong ngăn xếp bằng phép tính ∨. Kết quả được ghi lại vào bit đầu tiên của ngăn xếp. Giá trị còn lại được kéo lên 1 bit.	none	none
Logic PuSh				
LPS	none	Sao chép giá trị của bit đầu tiên vào bit thứ hai trong ngăn xếp. Gía trị còn lại bị đẩy xuống 1 bit. Bit cuối cùng bị đẩy ra ngoài.	none	none
Logic ReaD				
LRD	none	Lệnh sao chép giá trị của bit thứ hai vào bit đầu tiên của ngăn xếp, các giá trị còn lại của ngăn xếp vẫn giữ nguyên.	none	none
Logic PoP				
LPP	none	Lệnh kéo ngăn xếp lên 1 bit theo nguyên tắc bit sao đè lên bit trước.	none	none
LoaD Stack				
LDS n	none	Lệnh sao chép giá trị của bit thứ n (ngăn xếp có 9 bit thì bit thứ nhì được tính là 1...đến bit cuối cùng là 8) của ngăn xếp lên bit đầu tiên. Các giá trị còn lại của ngăn xếp bị đẩy lùi xuống 1 bit, bit cuối cùng bị đẩy ra khỏi ngăn xếp.	n: 1÷8	Byte



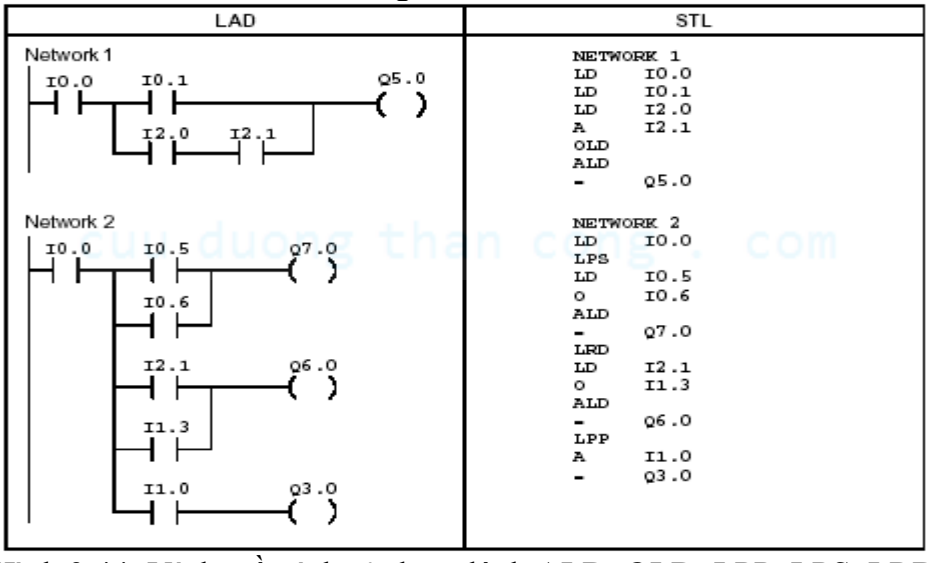
Hình 3.41: Mô tả hoạt động của lệnh LDS



Hình 3.42: Mô tả hoạt động của lệnh ALD và OLD



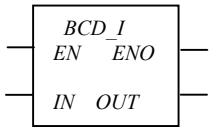
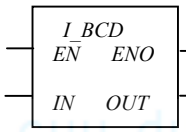
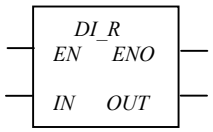
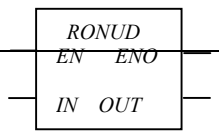
Hình 3.43: Mô tả hoạt động của lệnh LPS, LRD, LPP

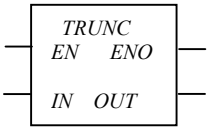
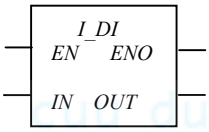
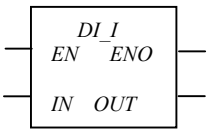


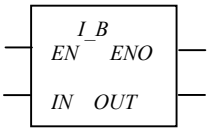
Hình 3.44: Ví dụ về cách sử dụng lệnh ALD, OLD, LPP, LPS, LRD

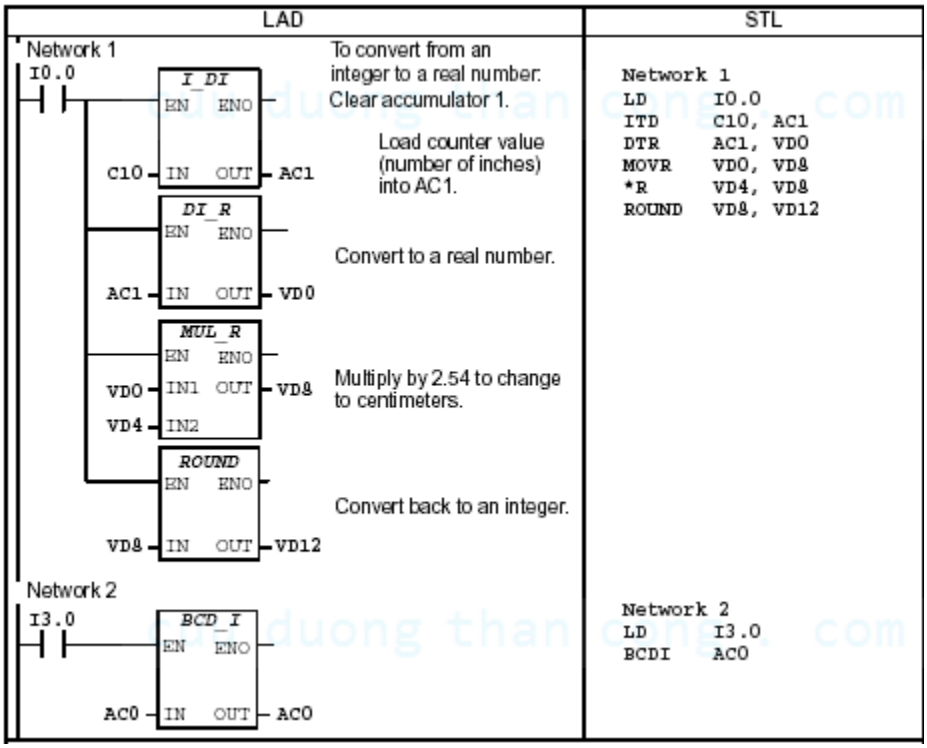
11. SIMATIC Conversion Instructions:

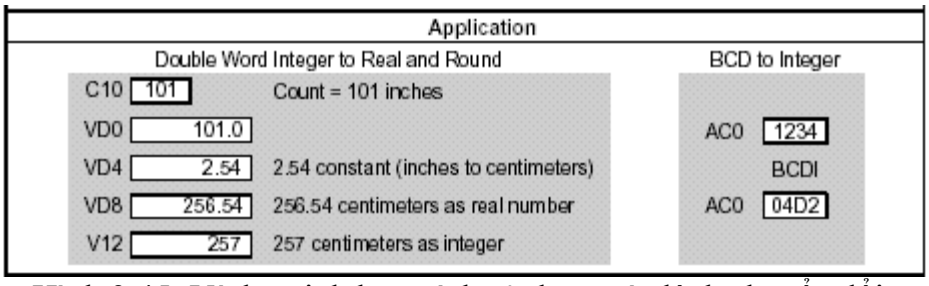
Các hàm đổi kiểu dữ liệu cho phép thực hiện việc đổi kiểu dữ liệu từ kiểu này sang kiểu khác. Sau đây là các lệnh biến đổi kiểu dữ liệu trong STL và LAD:

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
BCD to Integer and Integer to BCD				
BCDI OUT		Lệnh chuyển đổi một số nhị_thập phân IN sang số nguyên và lưu kết quả vào OUT. Giới hạn của IN: 0÷9999.	**IN:** IW, QW, VW, LW, MW, SMW, AIW ,AC, T, C, Constant, *VD, *AC, *LD, SW. **OUT:** IW, QW, VW, LW, MW, SMW, AC, T, C, *VD, *AC, *LD, SW.	Word
IBCD OUT		Lệnh chuyển đổi một số nguyên IN sang số nhị_thập phân và lưu kết quả vào OUT. Giới hạn của IN: 0÷9999.	**IN:** IW, QW, VW, LW, MW, SMW, AIW ,AC, T, C, Constant, *VD, *AC, *LD. **OUT:** IW, QW, VW, LW, MW, SMW, AC, T, C, *VD, *AC, *LD.	Word
Double Integer to Real				
DTR IN, OUT		Lệnh chuyển đổi số nguyên 32 bit IN sang số thực (32 bit) và lưu kết quả vào OUT.	**IN:** ID, QD, VD, LD, MD, SMD, AC, HD, Constant, *VD, *AC, *LD, SD. **OUT:**ID, QD, VD, LD, MD, SMD, AC, HD, *VD, *AC, *LD, SD.	DWord
Round				
ROUND IN,		Lệnh chuyển đổi số thực IN thành số nguyên double Integer (làm tròn số) và kết qủa lưu vào	**IN:** ID, QD, VD, LD, MD, SMD, AC, Constant, *VD, *AC, *LD, SD.	Real

OUT		OUT. Nếu phần lẻ >= 0.5 thì được làm tròn về phía lớn hơn 1 đơn vị.	**OUT:**ID, QD, VD, LD, MD, SMD, AC, HD, *VD, *AC, *LD, SD.	DINT
Truncate				
TRUNC IN, OUT		Hàm chuyển đổi số thực 32 bit có dấu sang số nguyên 32 bit có dấu.	**IN:** ID, QD, VD, LD, MD, SMD, AC, Constant, *VD, *AC, *LD, SD.	Real
			OUT:ID, QD, VD, LD, MD, SMD, AC, HD, *VD, *AC, *LD, SD.	DINT
Double Integer to Integer and Integer to Double Integer				
ITD IN, OUT		Lệnh chuyển đổi số nguyên 16 bit sang số nguyên 32 bit.	**IN:** IW, QW, VW, LW, MW,SW, SMW, AIW ,AC, T, C, Constant, *VD, *AC, *LD.	INT
			OUT: ID, QD,VD, LD, MD,SD, SMD, AC, *VD, *AC, *LD.	DINT
DTI IN, OUT		Lệnh chuyển đổi số nguyên 32 bit sang số nguyên 16 bit.	**IN:** ID, QD,VD, LD, MD,SD, SMD, AC,Constant, *VD, *AC, *LD.	DINT
			OUT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, *VD, *AC, *LD.	INT
Integer to Real, Byte to Integer and Integer to Byte				
(Integer to Real)	none	Không có lệnh chuyển đổi trực tiếp này. Ta có thể thực hiện được bằng cách dùng lệnh ITD (chuyển số nguyên 16 bit thành số nguyên 32 bit) sau đó dùng tiếp lệnh DTR (chuyển số nguyên 32 bit sang số thực).	none	none

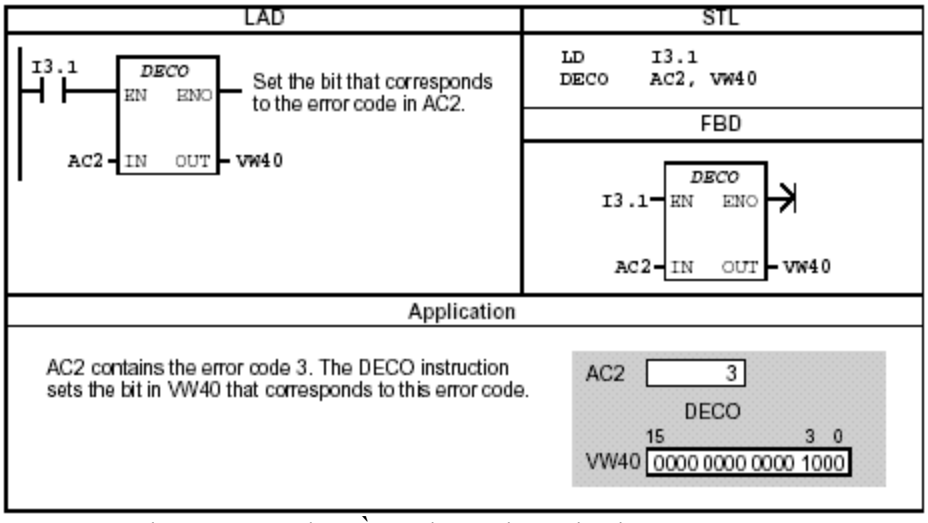
BTI IN, OUT		Lệnh chuyển đổi giá trị của Byte IN thành giá trị Integer 16 bit và lưu vào OUT.	**IN:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *AC, *LD.	Byte
			OUT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, *VD, *AC, *LD.	INT
IBT IN, OUT		Lệnh chuyển đổi giá trị trong Word IN thành giá Byte và lưu giá trị này vào OUT.	**IN:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, Constant, *VD, *AC, *LD.	INT
			OUT: IB, QB, MB, SMB, VB, SB, LB, AC, *VD, *AC, *LD.	Byte



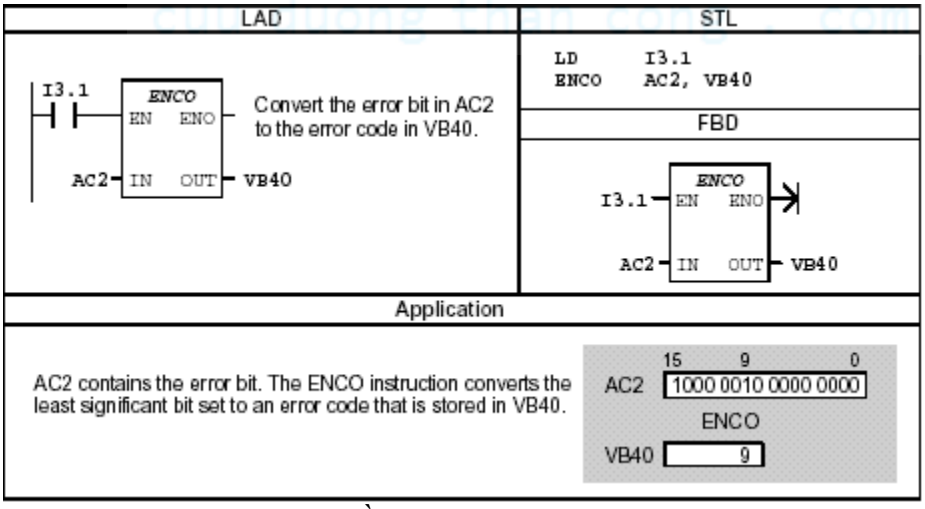


Hình 3.45: Ví dụ minh hoạ cách sử dụng các lệnh chuyển đổi

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Decode				
DECO IN, OUT		Lệnh đặt giá trị logic 1 vào bit của từ đơn OUT có chỉ số (trọng số của bit thuộc Word) bằng số nguyên nằm trong nibble (4 bit) thấp của byte đầu vào IN. Các bit còn lại của từ đơn có giá trị logic bằng 0.	**IN:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *AC, *LD	Byte
			OUT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, *VD, *AC, *LD.	Word
ENCO IN, OUT		Lệnh xác định chỉ số của bit thấp nhất trong từ đơn IN có giá trị logic 1và ghi kết quả này vào nibble thấp nhất của byte đầu ra OUT.	**IN:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, *VD, *AC, *LD.	Word
			OUT: IB, QB, MB, SMB, VB, SB, LB, AC, *VD, *AC, *LD.	Byte

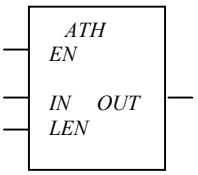
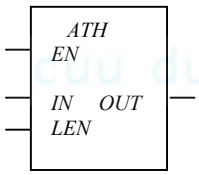


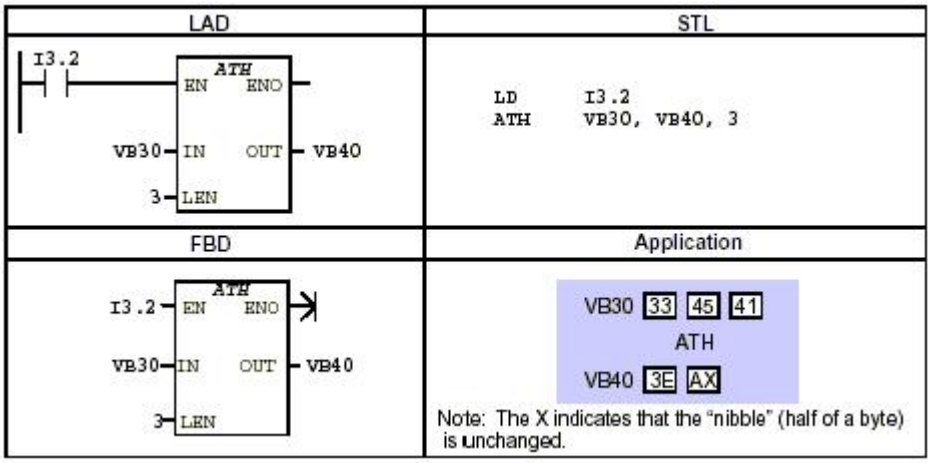
Hình 3.46: Ví dụ về cách sử dụng lệnh DECO



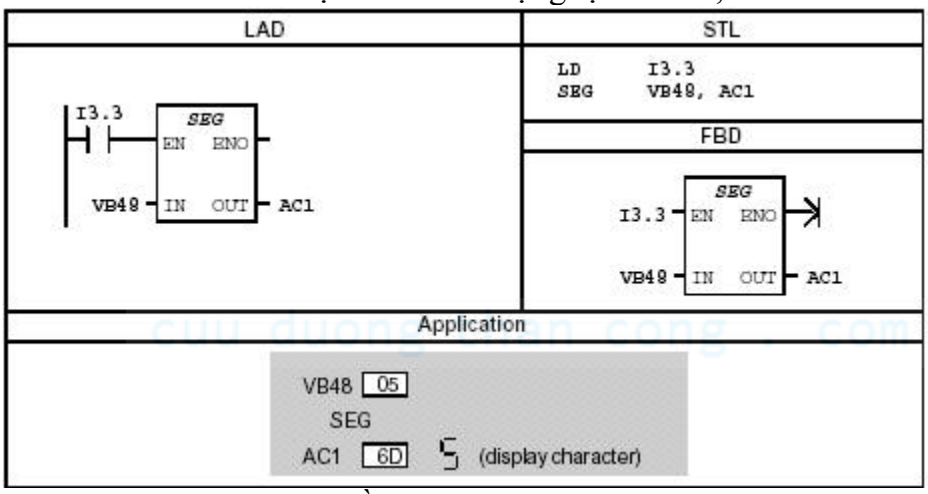
Hình 3.47: Ví dụ về cách sử dụng lệnh ENCO

STL	LAD	Mô tả (Description)	Toán hạng (Operands)	Kiểu dữ liệu (Data Types)
Segment				

SEG IN, OUT		Lệnh xuất các bit cho thanh ghi 7 đoạn tương ứng với nội dung của 4 bit thấp nhất của byte đầu vào IN. Kết quả được chi vào byte đầu ra.	**IN**: IB, QB, MB, SMB, LB, VB, AC, Constant, *VD, *AC, SB, *LD. **OUT:** IB, QB, MB, SMB, LB, VB, AC, *VD, *AC, SB, *LD.	Byte
ASCII to Hexa and Hexa to ASCII				
ATH IN, OUT, LEN	 	Thực hiện phép biến đổi một chuỗi kí tự có độ dài được chỉ thị trong toán hạng LEN, bắt đầu bằng kí tự chỉ định trong toán hạng IN, sang số nguyên hệ cơ số 16 và ghi vào vùng nhớ kể từ byte được chỉ định bởi OUT. Độ dài cực đại của chuỗi kí tự là 255. Những kí tự hợp lệ là những kí tự có mã ASCII từ 30÷39 và 41÷46 (cơ số 16, ứng với các kí tự từ 0÷9, A÷F). Nếu mã hoá một kí tự bị sai thì quá trình mã hoá bị dừng lại và bit SM1.7 có giá trị logic bằng 1.	**IN, OUT:** IB, QB, MB, SMB, LB, VB, *VD, *AC, SB, *LD. **LEN:** IB, QB, MB, SMB, LB, VB, AC, Constant, *VD, *AC, SB, *LD.	Byte Byte
HTA IN, OUT, LEN		Thực hiện đổi một dãy chữ viết trong hệ cơ số 16 thành chuỗi kí tự mã ASCII. Dãy số đầu vào được lưu trong mảng bắt đầu bằng IN và có độ dài là LEN. Độ dài cực đại của dãy số là 255. Chuỗi kí tự đầu ra được ghi vào mảng có byte đầu là OUT.	**IN, OUT:** IB, QB, MB, SMB, LB, VB, *VD, *AC, SB, *LD. **LEN:** IB, QB, MB, SMB, LB, VB, AC, Constant, *VD, *AC, SB, *LD.	Byte

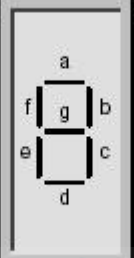


Hình 3.48: Ví dụ về cách sử dụng lệnh ATH, HTA



Hình 3.49: Ví dụ về cách sử dụng lệnh SEG

(IN) LSD	Segment Display	(OUT) -gfe dcba		(IN) LSD	Segment Display	(OUT) -gfe dcba
0		0011 1111		8		0111 1111
1		0000 0110		9		0110 0111
2		0101 1011		A		0111 0111
3		0100 1111		B		0111 1100
4		0110 0110		C		0011 1001
5		0110 1101		D		0101 1110
6		0111 1101		E		0111 1001
7		0000 0111		F		0111 0001



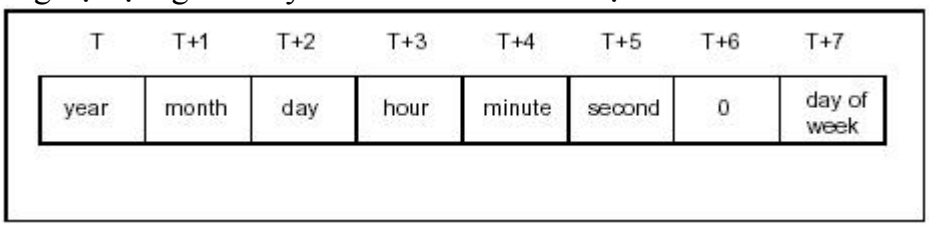
Hình 3.50: Mã hiển thị thanh ghi 7 đoạn

12. SIMATIC Clock Instrutions:

Tuyệt đối không sử dụng lệnh đọc/ghi (TODR/TODW) thời gian thực cùng một lúc trong chương trình chính và chương trình xử lý ngắt. Khi một lệnh TODR hoặc TODW đã thực hiện thì khi gọi chương trình xử lý ngắt, các lệnh làm việc với đồng hồ thời gian thực trong chương trình xử lý ngắt sẽ không được thực hiện nữa. Bit SM4.5 sẽ có mức logic 1 trong những trường hợp như vậy.

Đồng hồ thời gian thực chỉ có đối với CPU214 trở lên. Để có thể làm việc với đồng hồ thời gian thực thì CPU sẽ cung cấp 2 lệnh đọc/ghi giá trị cho đồng hồ. Những giá trị đọc được hoặc ghi được với đồng hồ thời gian thực là các giá trị về ngày, tháng, năm và các giá trị về giờ, phút, giây.

Các dữ liệu đọc/ghi với đồng hồ thời gian thực trong LAD, STL có độ dài 1 byte và phải được mã hoá theo kiểu số nhị thập phân BCD (Ex: 16#95 cho năm 95).
Chúng nằm trong bộ đệm gồm 8 byte liền nhau theo thứ tự như sau:



Hình 3.51: Bộ đệm 8 byte của lệnh đồng hồ thời gian thực

Các giá trị của các thông số phải nằm trong giới hạn:

Year/Month	yymm	yy - 0 to 99	mm - 1 to 12
Day/Hour	ddhh	dd - 1 to 31	hh - 0 to 23
Minute/Second	mmss	mm - 0 to 59	ss - 0 to 59
Day of week	d	d - 0 to 7	1 = Sunday
			0 = disables day of week (remains 0)

CPU S7-200 không thực hiện kiểm tra lại ngày tháng, ngày của tuần để điều chỉnh lại ngày tháng. Giá trị về ngày tháng như là February 30 có thể được chấp nhận. Do đó bạn sẽ phải chắc chắn rằng ngày tháng của bạn đưa vào đó là đúng.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Read Real-Time Clock and Set Real-Time Clock				
TODR T	READ_RTC EN ENO T	Lệnh đọc nội dung của đồng hồ thời gian thực vào bộ đệm 8 byte được chỉ định trong lệnh bằng toán hạng T.	**T:** VB,IB, QB, MB, SMB, SB, LB, *VD, *AC, *LD.	Byte
TODW T	SET_RTC EN ENO T	Lệnh ghi nội dung của bộ đệm 8 byte được chỉ định trong lệnh bằng toán hạng T vào đồng hồ thời gian thực.		

13. SIMATIC Program Control Instrutions:

Các lệnh của chương trình, nếu không có những lệnh điều khiển riêng, sẽ được thực hiện tuần tự từ trên xuống dưới trong một vòng quét. Lệnh điều khiển chương trình cho phép thay đổi thứ tự thực hiện lệnh. Chúng cho phép chuyển thứ tự như: Đáng lẽ ra là lệnh tiếp theo, tới một lệnh bất cứ nào khác của chương trình; trong đó nơi điều khiển chuyển đến phải được đánh dấu trước bằng ***nhãn chỉ đích***. Nhóm lệnh điều khiển chương trình gồm: lệnh nhảy, lệnh gọi chương trình con, nhãn chỉ đích (hay gọi đơn giản là nhãn), phải được đánh dấu trước khi thực hiện lệnh nhảy hay lệnh gọi chương trình con.

Việc đặt nhãn cho lệnh nhảy phải nằm trong chương trình. Nhãn của chương trình con hay nhãn của chương trình xử lý ngắt phải được khai báo ở đầu chương trình. Không thể dùng lệnh JMP để chuyển điều khiển từ chương trình chính vào nhãn bất kỳ trong chương trình con hoặc chương trình xử lý ngắt. Ngược lại cũng không được phép từ một chương trình con hay chương trình xử lý ngắt nhảy ra ngoài chương trình chính đó.

Lệnh gọi chương trình con là lệnh chuyển quyền điều khiển đến chương trình con. Sau khi chương trình con thực hiện xong thì quyền điều khiển lại được chuyển về lệnh tiếp theo trong chương trình chính ngay sau lệnh gọi chương trình con. Từ một chương trình con có thể gọi một chương trình con khác trong nó, có thể gọi như vậy nhiều nhất là 8 lần. Phép đệ quy cũng có thể thực hiện được trong S7-200, mặc dù không bị cấm song phải chú ý đến giới hạn trên.

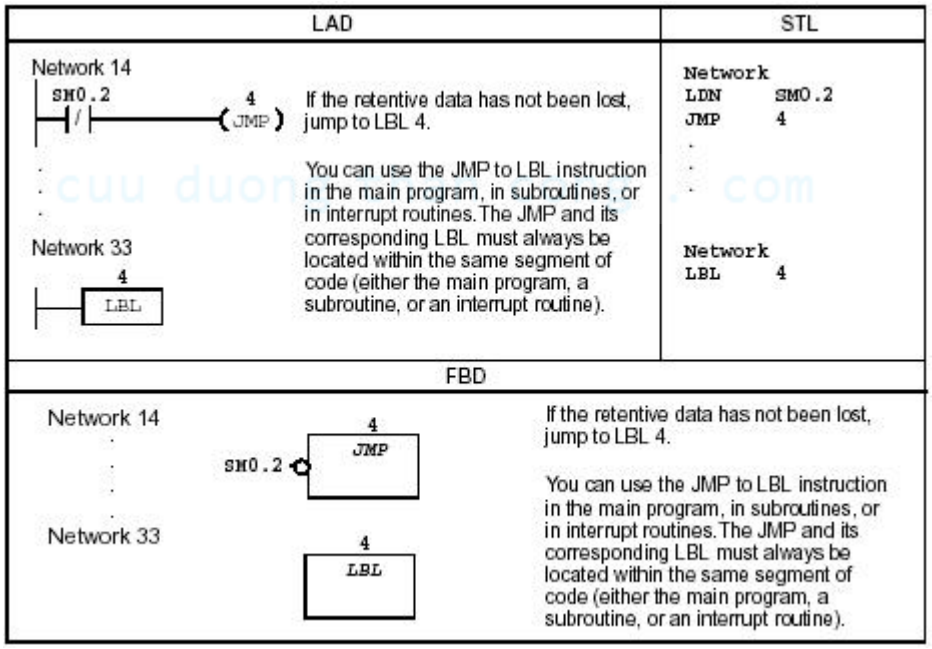
Trạng thái của ngăn xếp: Nếu lệnh nhảy hay lệnh gọi chương trình con được thực hiện thì đỉnh ngăn xếp luôn có giá trị logic bằng 1. Như vậy trong chương trình con các lệnh có điều kiện được thực hiện như lệnh không có điều kiện. Sau các lệnh **LBL** (lệnh đặt nhãn) và **SBR**, lệnh **LD** trong STL sẽ bị vô hiệu hoá**.**

Khi một chương trình con được gọi, toàn bộ nội dung trong ngăn xếp sẽ được cất đi, đỉnh của ngăn xếp nhận giá trị logic mới là 1, các bit khác còn lại của ngăn xếp nhận giá trị logic là 0 và điều khiển được chuyển đến chương trình con đã được gọi. Khi thực hiện xong chương trình con và trước khi quyền điều khiển được chuyển đến chương trình đã gọi nó thì nội dung của ngăn xếp đã được cất giữ trước đó sẽ được chuyển trở lại cho ngăn xếp.

Nội dung của thanh ghi AC không được cất giữ khi gọi chương trình con, nhưng khi một chương trình xử lý ngắt được gọi, nội dung thanh ghi AC sẽ được cất giữ trước khi thực hiện chương trình xử lý ngắt và trả lại sau khi chương trình xử lý ngắt vừa thực hiện xong. Bởi vậy chương trình xử lý ngắt có thể tự do sử dụng 4 thanh ghi AC của S7-200.

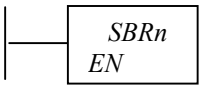
STL	LAD	Mô tả (Description)	Toán hạng (Operands)	Kiểu dữ liệu (Data Types)
Jump to Label and Label				

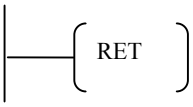
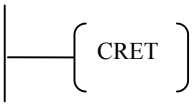
JMP n	n JMP	Lệnh nhảy thực hiện chuyển quyền điều khiển đến nhãn n trong một chương trình.	**n:** CPU 212:0 đến 63 CPU 21x khác từ 0 đến 255.	none
LBL n	n JMP	Lệnh khai báo nhãn n trong một chương trình.		

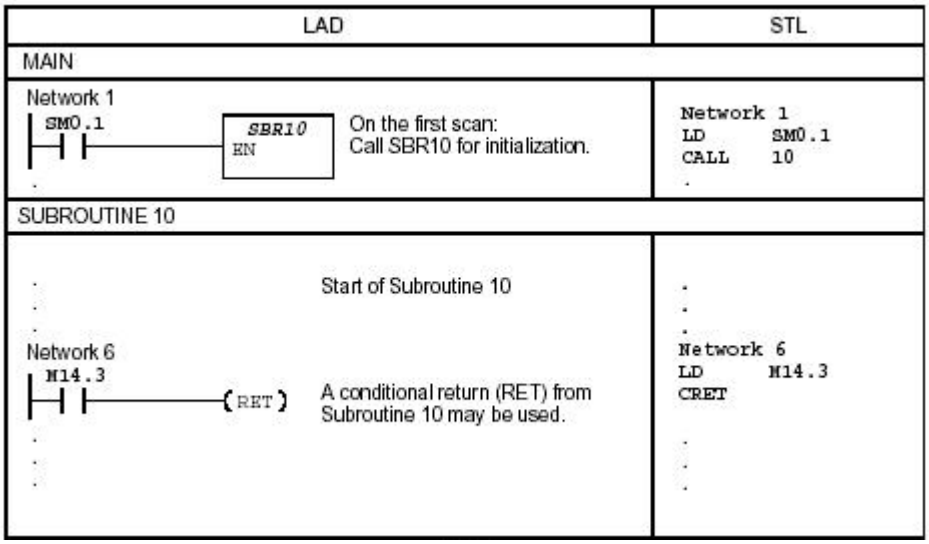


Hình 3.52: Ví dụ cách sử dụng lệnh JMP, LBL

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Subroutine and Return Subroutine				



SBR n		Lệnh gọi chương trình con, thực hiện phép chuyển quyền điều khiển đến chương trình con có nhãn n.	**n:** CPU 212:0 đến 15 CPU 21x khác từ 0 đến 255.	none
RET		Lệnh trở về chương trình đã gọi chương trình con không điều kiện.	none	none
CRET		Lệnh trở về chương trình đã gọi chương trình con có điều kiện.		



Hình 3.53: Ví dụ cách sử dụng lệnh gọi và thoát khỏi chương trình con

Các lệnh sau sẽ can thiệp vào thời gian vòng quét, nó được dùng để kết thúc chương trình đang thực hiện hoặc kéo dài thêm thời gian của vòng quét.

Trong chương trình chính, kết thúc chương trình bằng lệnh MEND, nhưng trong soạn thảo chương trình chúng ta không cần lệnh kết thúc này mà Step 7 MicroWin đã mặc định rồi. Lệnh END cũng là lệnh kết thúc chương trình nhưng là lệnh kết thúc có điều kiện.

Khi chương trình chính hoặc chương trình con gặp lệnh STOP thì chương trình sẽ kết thúc ngay tại cuối vòng quét hiện thời và CPU chuyển sang chế độ STOP.

Nếu trong chương trình xử lý ngắt gặp lệnh STOP thì ngắt cũng được dừng lại ngay lập tức, các tín hiệu xử lý ngắt đang còn nằm trong hàng đợi sẽ bị huỷ bỏ, phần còn lại của chương trình sẽ không thực hiện.Việc thực sự chuyển sang chế độ STOP xảy ra ở cuối chu kỳ vòng quét hiện thời sau giai đoạn xuất tín hiệu cho đầu ra.

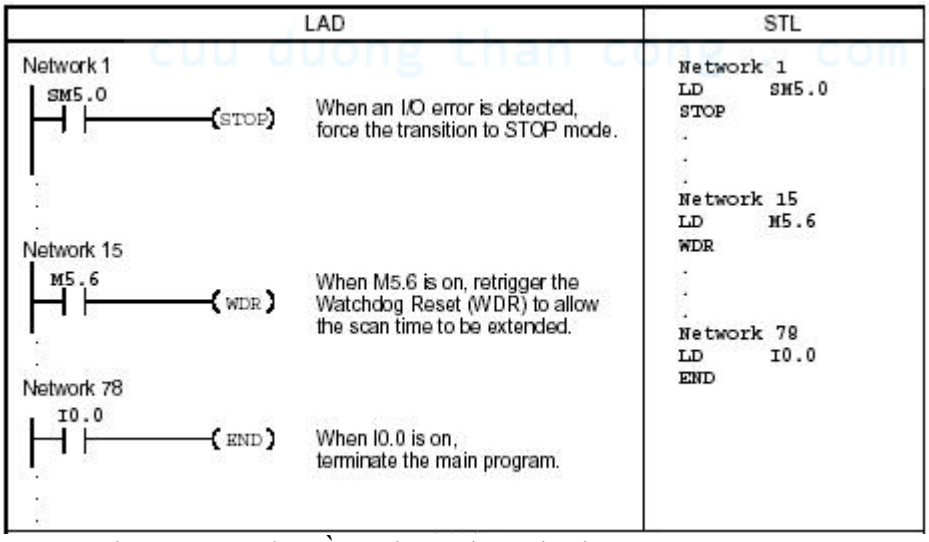
Lệnh WDR sẽ khởi động lại đồng hồ quan sát *(Watchdog Timer),* chương trình tiếp tục thực hiện trong vòng quét ở chế độ quan sát. Nên cẩn thận khi sử dụng lệnh này. Khi trong chương trình sử dụng lệnh lặp, hoặc thời gian trễ quá lớn thì những quá trình sau bị hạn chế:

- Truyền thông (loại trừ kiểu Freeport).
- Cập nhật vào ra (trừ những lệnh vào ra tức thì).
- Cập nhật cưỡng bức.
- Cập nhật các bit kiểu SM.
- Chuẩn đoán thời gian chạy.
- Với các vòng quét lớn hơn 25 giây thì các bộ Timer có độ phân giải10ms và 100ms sẽ không được chính xác.

Nếu thời gian của vòng quét lớn hơn 300ms, hoặc khi gặp một ngắt có chương trình xử lý ngắt với thời gian chạy chương trình lâu hơn 300ms thì cần phải sử dụng lệnh WDR để khởi động lại ***đồng hồ quan sát***.

Việc chuyển công tắc phần cứng sang chế độ STOP hoặc thực hiện lệnh STOP trong chương trình sẽ là nguyên nhân đặt chế độ điều khiển vào chế độ dừng trong khoảng thời gian 1.4s.

STL	LAD	Mô tả (Description)	Toán hạng (Operands)	Kiểu dữ liệu (Data Types)
End and Stop and Watchdog Timer				
END	—(END)	Lệnh kết thúc chương trình hiện hành có điều kiện.	none	none
STOP	—(STOP)	Lệnh kết thúc chương trình hiện hành và chuyển sang chế độ STOP.		
WDR	—(WDR)	Lệnh khởi động lại đồng hồ quan sát.		



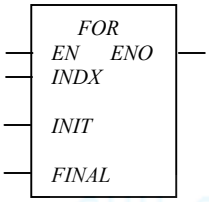
Hình 3.54: Ví dụ về cách sử dụng lệnh STOP, WDR, END

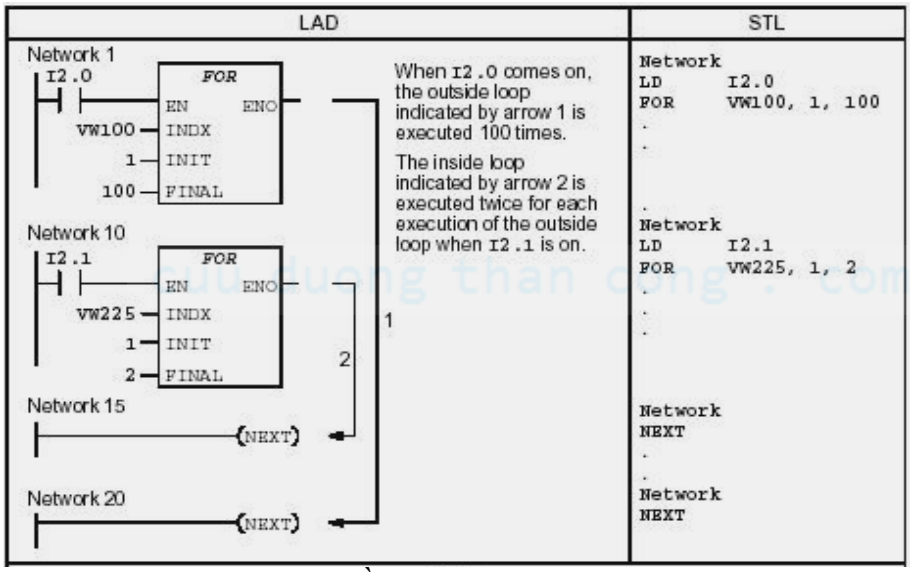
Để xây dựng cấu trúc vòng lặp nhằm thực hiện lặp một khối lệnh riêng biệt trong chương trình. Sử dụng lệnh FOR...NEXT để thiết kế một vòng lặp với số lần có thể định trước bằng hai toán hạng INIT kiểu từ đơn chỉ ***điểm khởi phát*** và FINAL cũng kiểu từ đơn chỉ ***điểm kết thúc***. Ngoài ra lệnh còn sử dụng một từ đơn INDX để lưu ***số vòng lặp tức thời.***

Mỗi một câu lệnh FOR đòi hỏi phải có một câu lệnh NEXT đứng cuối khối lệnh được lặp. Các vòng FOR...NEXT có thể được lồng vào nhau nhưng số lệnh lồng vào nhau không được vượt quá 8 lần.

Tại thời điểm bắt đầu thực hiện lệnh vòng lặp FOR, từ đơn INDX nhận giá trị của INIT. Sau đó, mỗi khi kết thúc một vòng lặp, tức là khi gặp lệnh NEXT, nội dung của INDX được tăng lên 1 đơn vị và được so sánh với nội dung của FINAL. Nếu nội dung của INDX chưa lớn hơn nội dung của FINAL thì chương trình sẽ tiếp tục thực hiện lại vòng lặp, ngược lại khi nội dung của INDX đã lớn hơn nội dung của FINAL thì chương trình sẽ kết thúc lệnh FOR...NEXT và tiếp tục thực hiện lệnh kế tiếp nằm ngay sau lệnh NEXT.

Khi lệnh NEXT thực hiện thì bit đầu tiên trong ngăn xếp có giá trị logic bằng 1.

STL	LAD	Mô tả (Description)	Toán hạng (Operands)	Kiểu dữ liệu (Data Types)
FOR...NEXT				
FOR INDX, INIT, FINAL		Ví dụ đưa vào INIT giá trị 1, FINAL giá trị là 10. Lệnh sẽ thực hiện lặp đúng 10 lần, số lần lặp được quản lý trong từ đơn INDX. Vượt quá 10 lần lệnh sẽ kết thúc và chương trình tiếp tục thực hiện các lệnh kế tiếp.	**INDX:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, *VD, *AC, *LD.	INT
			INIT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, Constant, *VD, *AC, *LD.	INT
			FINAL: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, AIW, Constant, *VD, *AC, *LD.	INT
NEXT		Lệnh kết thúc vòng lặp.	none	none



Hình 3.55: Ví dụ về cách sử dụng lệnh FOR...NEXT

14. SIMATIC Shift and Rotate Register Instrutions:

Làm việc với thanh ghi có nhóm lệnh sau:

Lệnh dịch chuyển thanh ghi, trong này cũng có hai nhóm:

+ Lệnh dịch chuyển thanh ghi 8 bit, 16 bit, 32 bit.
+ Lệnh dịch chuyển thanh ghi có độ dài tuỳ ý, được định nghĩa trong lệnh.

Lệnh quay vòng thanh ghi, trong này cũng có hai nhóm :

+ Lệnh quay vòng thanh ghi 8 bit, 16 bit, 32 bit.
+ Lệnh quay vòng thanh ghi có độ dài tuỳ ý, được định nghĩa trong lệnh.

Khi sử dụng lệnh dịch chuyển các bit của thanh ghi (Byte, Word, DWord) cần chú ý các điểm sau đây:

1. Không thực hiện việc dịch chuyển nếu số lần đẩy bằng 0.
2. Nếu số lần đẩy có giá trị lớn hơn 0, bit nhớ tràn SM1.1 sẽ có giá trị của bit cuối cùng được đẩy ra.
3. Nếu số lần đẩy lớn hơn hoặc bằng 8 đối với byte, 16 đối với Word, 32 đối với từ kép thì lệnh sẽ thực hiện lệnh đẩy lớn nhất chỉ bằng 8, 16, 32.
4. Lệnh SLB (đẩy các bit của byte sang trái), SLW (đẩy các bit của Word sang trái) và SLD (đẩy các bit của từ kép sang trái) sẽ chuyển giá trị 0 vào bit thấp nhất của Byte, Word hoặc DWord sau mỗi lần đẩy. Sau lệnh thực hiện, bit SM1.1 sẽ có giá trị logic của bit thứ 8-N, 16-N hoặc 32-N, trong đó N là số lần đẩy.
5. Lệnh SRB (đẩy các bit của byte sang phải), SRW (đẩy các bit của Word sang phải) và SRD (đẩy các bit của từ kép sang phải) sẽ chuyển giá trị 0 vào bit thấp nhất của Byte, Word hoặc DWord sau mỗi lần đẩy. Sau lệnh thực hiện, bit SM1.1 sẽ có giá trị logic của bit thứ N-1, trong đó N là số lần đẩy.
6. Bit báo kết quả 0 (bit SM1.0) sẽ có giá trị logic bằng 1 nếu như sau khi thực hiện lệnh đẩy nội dung của Byte, Word, DWord bằng 0.

Khi sử dụng lệnh quay vòng các bit của thanh ghi (Byte, Word, DWord) cần chú ý các điểm sau đây:

1. Lệnh quay thực hiện phép đẩy vòng tròn sang trái hoặc sang phải các bit của một Byte, Word, DWord. Tại mỗi một lần quay, giá trị của các bit bị đẩy ra ở một đầu của thanh ghi lại được đưa vào đầu kia của thanh ghi đó.
2. Không thực hiện việc quay vòng nếu số lần quay bằng 0. Hay bằng một bội số của 8 (đối với byte), của 16 (đối với word) và của 32 (đối với DWord).
3. Đối với các giá trị của số đếm lần quay lớn hơn 8 (đối với byte), của 16 (đối với word) và của 32 (đối với DWord) lệnh sẽ thực hiện với số đếm lần quay mới bằng phần dư của của phép chia tương ứng.
4. Khi thực hiện lệnh quay sang phải RRB (quay các bit của byte sang phải), RRW (quay các bit của Word sang phải) và RRD (quay các bit của từ kép sang phải), tại mỗi lần quay giá trị của bit thấp nhất được ghi vào bit báo tràn SM1.1. Sau khi lệnh thực hiện, bit SM1.1 sẽ có giá trị logic của bit thứ N - 1, trong đó N là số đếm lần quay.

5. Khi thực hiện lệnh quay sang trái RLB (quay các bit của byte sang trái), RLW (quay các bit của Word sang trái) và RLD (quay các bit của từ kép sang trái), tại mỗi lần quay giá trị của bit cao nhất được ghi vào bit boá tràn SM1.1.Sau khi lệnh thực hiện, bit SM1.1 sẽ có giá trị logic của bit thứ N - 8, N – 16, N – 32, trong đó N là số đếm lần quay.

6. Bit báo kết quả 0 (bit SM1.0) sẽ có giá trị logic bằng 1 nếu như sau khi thực hiện lệnh quay nội dung của Byte, Word, DWord bằng 0.

Các lệnh dich chuyển hoặc quay vòng ảnh hưởng đến kết quả của các bit đặc biệt như sau:

Lệnh	Kiểu lệnh	SM1.0 (kết quả 0)	SM1.1 (báo tràn)	SM1.2 (kết qủa âm)	SM1.3 (chia cho 0)
SRB	không dấu	có	có	không	không
SLB	không dấu	có	có	không	không
SRW	không dấu	có	có	không	không
SLW	không dấu	có	có	không	không
SRD	không dấu	có	có	không	không
SLD	không dấu	có	có	không	không
RRB	không dấu	có	có	không	không
RLB	không dấu	có	có	không	không
RRW	không dấu	có	có	không	không
RLW	không dấu	có	có	không	không
RRD	không dấu	có	có	không	không
RLD	không dấu	có	có	không	không
SHRB	không dấu	không	có	không	không

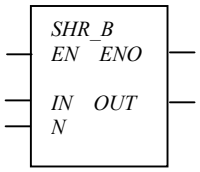
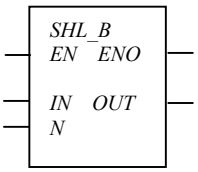
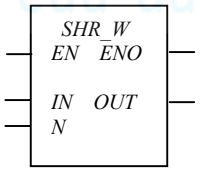
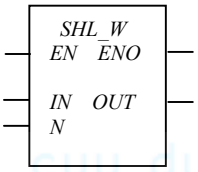
Những điều sau đây chỉ đúng với các hàm dịch chuyển bit của byte, từ đơn và từ kép:

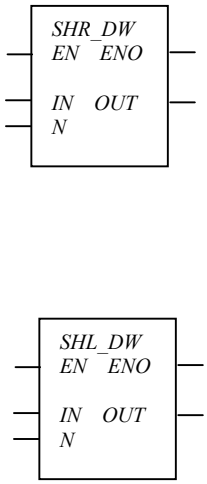
+ Nếu bộ đếm chuyển dịch có giá trị lớn hơn 0 thì bit nhớ tràn SM1.1 có giá trị logic của bit cuối cùng được đẩy ra.
+ Bit báo kết quả 0 SM1.0 có giá trị logic 1 nếu sau khi lệnh được thực hiện, byte, từ hoặc từ kép có nội dung bằng 0.

Những điều sau đây chỉ đúng với các hàm dịch chuyển bit của byte, từ đơn và từ kép:

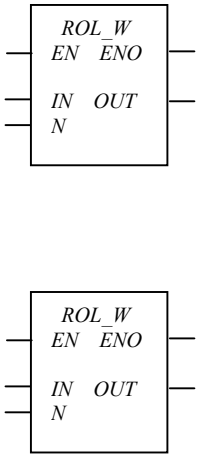
+ Nếu bộ đếm chuyển dịch không phải là bộ số nguyên của 8, 16, 32 đối với byte, Word, DWord thì giá trị của bit cuối cùng bị đẩy ra ngoài sẽ được gán cho bit nhớ tràn SM1.1.
+ Nếu bit báo kết quả 0 có giá trị logic bằng 1 thì giá trị của byte, từ hay từ kép bằng 0.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Shift Right Byte and Shift Left Byte				
		Lệnh dịch phải hay lệnh dịch trái thực		Byte

SRB OUT, N *SLB* OUT, N	 	hiện dịch chuyển các bit của Byte đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Lệnh shift điền giá trị zero vào các bit vừa bị dịch chuyển đi, bit cuối cùng bị dịch chuyển ra sẽ được đưa vào bit báo tràn SM1.1. Bit báo kết quả 0 sẽ được set lên 1 nếu giá trị của byte dịch chuyển là 0.	**IN:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *AC, *LD. **OUT:** IB, QB, MB, SMB, VB, SB, LB, AC, *VD, *AC, *LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *AC, *LD.	
Shift Right Word and Shift Left Word				
SRW OUT, N *SLW* OUT, N	 	Lệnh dịch phải hay lệnh dịch trái thực hiện dịch chuyển các bit của Word đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Lệnh shift điền giá trị zero vào các bit vừa bị dịch chuyển đi, bit cuối cùng bị dịch chuyển ra sẽ được đưa vào bit báo tràn SM1.1. Bit báo kết quả 0 sẽ được set lên 1 nếu giá trị của Word dịch chuyển là 0.	**IN:** IW, QW, VW, LW, MW,SW, SMW, AIW ,AC, T, C, Constant, *VD, *AC, *LD. **OUT:** IW, QW, VW, LW, MW,SW, SMW, AC, T, C, *VD, *AC, *LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *AC, *LD.	Word Byte

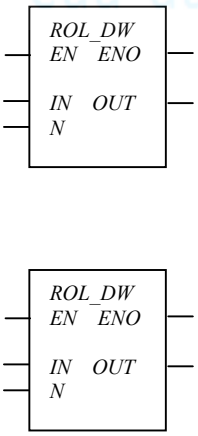
STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Shift Right Double Word and Shift Left Double Word				
SRD OUT, N *SLD* OUT, N		Lệnh dịch phải hay lệnh dịch trái thực hiện dịch chuyển các bit của từ kép đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Lệnh shift điền giá trị zero vào các bit vừa bị dịch chuyển đi, bit cuối cùng bị dịch chuyển ra sẽ được đưa vào bit báo tràn SM1.1. Bit báo kết quả 0 sẽ được set lên 1 nếu giá trị của từ kép dịch chuyển là 0.	**IN:** VD, ID, QD, MD, LD, SD, HC, SMD, AC, Constant, *VD, *AC, *LD. **OUT:** VD, ID, QD, MD, LD, SD, SMD, AC, *VD, *AC, *LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *AC, *LD.	DWord Byte
Rotate Right Byte and Rotate Left Byte				
RRB OUT, N *RLB* OUT, N		Lệnh quay vòng sang phải hay lệnh quay vòng sang trái thực hiện dịch chuyển các bit của byte đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Tại mỗi lần quay, giá trị của bit cuối cùng (bit 0) được đưa vào bit SM1.1 đồng thời đưa vào bit đầu tiên (bit 7) của byte đó nếu là quay phải, còn ngược lại đối với lệnh quay trái. Bit báo kết quả 0 sẽ có giá trị bằng 1 nếu giá trị trong byte đó bằng 0.	**IN:** IB, QB, MB, SMB, VB, SB, LB, AC, *VD, *AC, *LD. **OUT:** IB, QB, MB, SMB, VB, SB, LB, AC, *VD, *AC, *LD. **N:** IB, QB, MB, SMB, VB, SB, LB, AC, Constant, *VD, *AC, *LD.	Byte

<table>
<tr><th colspan="5">Rotate Right Word and Rotate Left Word</th></tr>
<tr>
<td>RRW OUT, N

RLW OUT, N</td>
<td> 
</td>
<td>Lệnh quay vòng sang phải hay lệnh quay vòng sang trái thực hiện dịch chuyển các bit của từ đơn đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Tại mỗi lần quay, giá trị của bit cuối cùng (bit 0) được đưa vào bit SM1.1 đồng thời đưa vào bit đầu tiên (bit 7) của byte đó nếu là quay phải, còn ngược lại đối với lệnh quay trái. Bit báo kết quả 0 sẽ có giá trị bằng 1 nếu giá trị trong từ đơn đó bằng 0.</td>
<td>IN: IW, QW, VW, LW, MW,SW, SMW, AIW ,AC, T, C, Constant, ∗VD, ∗AC, ∗LD.
OUT: IW, QW, VW, LW, MW,SW, SMW, AC, T, C, ∗VD, ∗AC, ∗LD.

N: IB, QB, MB, SMB, VB, SB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.</td>
<td>Word

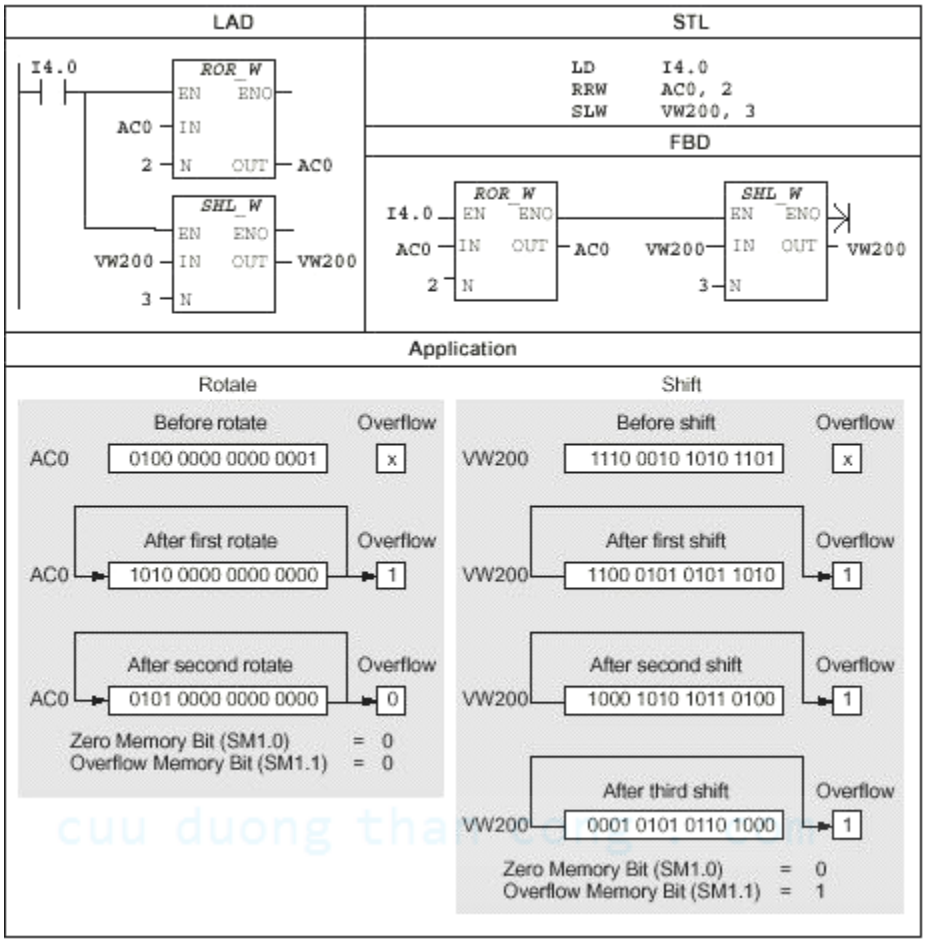
Byte</td>
</tr>
<tr><th colspan="5">Rotate Right Double Word and Rotate Left Double Word</th></tr>
<tr>
<td>RRD OUT, N

RLD OUT, N</td>
<td> 
</td>
<td>Lệnh quay vòng sang phải hay lệnh quay vòng sang trái thực hiện dịch chuyển các bit của từ kép đầu vào IN đi N lần sang phải hay trái. kết quả được lưu vào đầu ra OUT. Tại mỗi lần quay, giá trị của bit cuối cùng (bit 0) được đưa vào bit SM1.1 đồng thời đưa vào bit đầu tiên (bit 7) của từ kép đó nếu là quay phải, còn ngược lại đối với lệnh quay trái. Bit báo kết quả 0 sẽ có giá trị bằng 1 nếu giá trị trong từ kép đó bằng 0.</td>
<td>IN: VD, ID, QD, MD, LD, HC, SMD, AC, Constant, ∗VD, ∗AC, ∗LD.

OUT: VD, ID, QD, MD, LD, SMD, AC, ∗VD, ∗AC, ∗LD.

N: IB, QB, MB, SMB, VB, LB, AC, Constant, ∗VD, ∗AC, ∗LD.</td>
<td>DWord

Byte</td>
</tr>
</table>



Hình 3.56: Ví dụ về cách sử dụng lệnh dịch chuyển và quay vòng thanh ghi

Lệnh làm việc với thanh ghi có độ dài tuỳ ý:

Lệnh thuộc nhóm này cung cấp một phương pháp nối tiếp và điều khiển dòng sản phẩm hoặc dữ liệu. Thanh ghi được xác định trong lệnh bởi toán hạng S_BIT chỉ địa chỉ bit thấp của thanh ghi và độ dài là giá trị tuyệt đối của toán hạng N trong lệnh (nghĩa là thanh ghi có độ dài |N| bit). Dữ liệu được chuyển vào trong thanh ghi có tên là DATA (DATA = Bool), một lần trong một vòng quét.

S_BIT là bit thấp nhất của thanh ghi, nếu gọi cao nhất trong thanh ghi là MSB.b thì MSB.b sẽ được tính theo công thức sau:

MSB.b = [(byte của S_BIT) + phần nguyên của(|N| - 1 + bit của S_BIT)/8].[phần còn thừa của phép chia 8]

Lý do trừ đi 1 bởi vì S_BIT đã chiếm mất 1 bit của thanh ghi.

Ví dụ S_BIT là V33.4 và N = 14 thì MSB.b sẽ là:

MSB.b = [(33) + (|14| - 1 + 4)/8]*remainder of the division by 8

= (33 + 2)*remainder of the division by 8

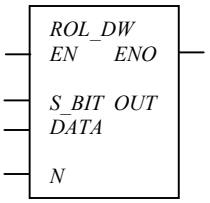
= 35.1

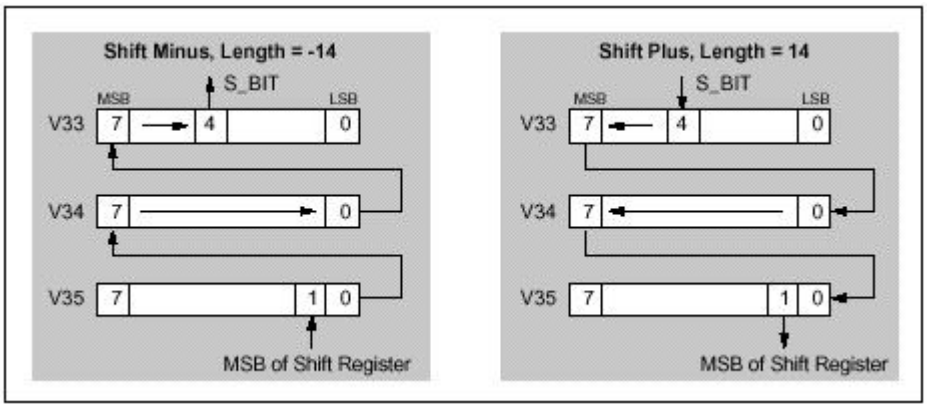
MSB.b là: V35.1

Chiều thực hiện phép dịch chuyển phụ thuộc vào dấu của toán hạng N trong lệnh. Miền giá trị cho phép của toán hạng N là: $-64 \le N \le 64$.

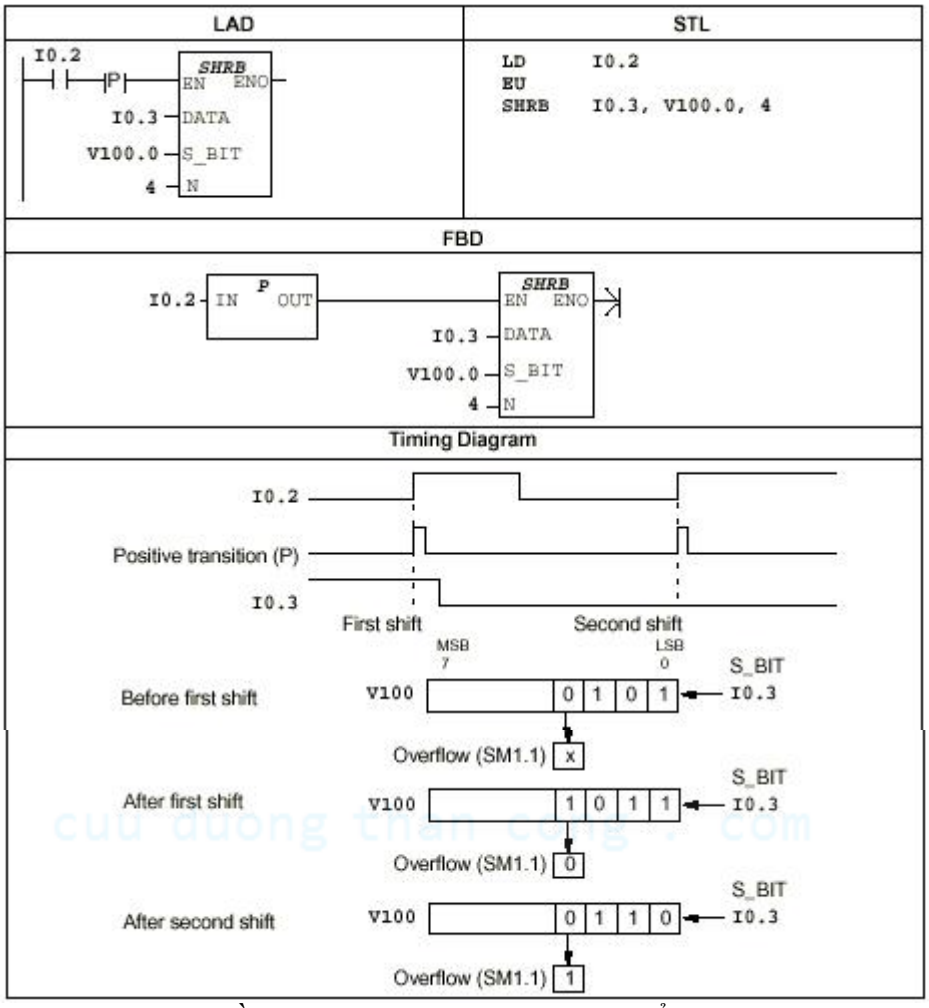
Nếu N dương thì phép dịch chuyển là phép dịch trái, giá trị của DATA được chuyển vào bit thấp nhất, giá trị logic trong bit cao nhất bị đẩy ra ngoài (vào bit báo tràn SM1.1). Ngược lại N là âm thì phép dịch chuyển là phép dịch phải, giá trị của DATA được chuyển vào bit cao nhất, giá trị logic trong bit thấp nhất bị đẩy ra ngoài (vào bit báo tràn SM1.1).

SHRB Lệnh dịch chuyển các bit của thanh ghi một vị trí trong một vòng quét. Thanh ghi được xoá trong lệnh bằng các toán hạng S_BIT chỉ địa chỉ bit thấp trong thanh ghi và |N| chỉ độ dài thanh ghi. Giá trị logic của bit bị đẩy ra khỏi thanh ghi được ghi vào bit báo tràn SM1.1.

STL	LAD	Toán hạng Operands	Kiểu dữ liệu Data Types
Shift Register Bit			
SHRB *DATA, S_BIT, N*		**DATA, S_BIT:** I, Q, V, M, SM, T, C, S, L. **N:** IB, QB, MB, SMB, VB, LB, AC, Constant, *VD, *AC, *LD.	Bool Byte



Hình 3.57: Mô tả hướng dịch chuyển của thanh ghi với toán hạng âm và dương



Hình 3.58: Ví dụ về cách sử dụng lệnh dịch chuyển thanh ghi có độ dài bất kỳ

15. SIMATIC Interupt and Comunication Instrutions:

Các chế độ ngắt và xử lý ngắt cho phép thực hiện các *quá trình tốc độ cao*, phản ứng kịp thời với các sự kiện ở bên trong và bên ngoài.

Nguyên tắc cơ bản của một chế độ ngắt cũng giống như thực hiện việc gọi một chương trình con, chỉ khác nhau ở đây là chương trình con được *gọi chủ động* bằng lệnh gọi chương trình con CALL, còn chương trình xử lý ngắt được *gọi bị động bằng tín hiệu báo ngắt.*

Khi có một tín hiệu báo ngắt, hệ thống sẽ tổ chức thực hiện gọi và thực hiện chương trình con tương ứng với tín hiệu ngắt đó, hay nói cách khác là hệ thống sẽ tổ chức xử lý tín hiệu báo ngắt đó. Chương trình con này được gọi là *chương trình xử lý ngắt.*

Do việc gọi chương trình xử lý ngắt bằng một tín hiệu báo ngắt mà thời điểm xuất hiện tín hiệu báo ngắt hoàn toàn bị động, bởi vậy hệ thống sẽ phải hỗ trợ thêm cho công việc xử lý ngắt như: cất giữ nội dung ngăn xếp, nội dung thanh ghi AC và các bit nhớ đặc biệt; tổ chức xếp hàng ưu tiên cho các tín hiệu xử lý ngắt trong trường hợp chúng chưa kịp thời xử lý.

Bảng3.7:Liệt kê các tín hiệu báo ngắt tương ứng với từng loại CPU 21x

Kiểu ngắt	Mô tả tín hiệu ngắt	CPU 212	CPU 214	CPU 215_2DP	CPU 216
0	Ngắt theo sườn lên của I0.0*	Y	Y	Y	Y
1	Ngắt theo sườn xuống của I0.0*	Y	Y	Y	Y
2	Ngắt theo sườn lên của I0.1		Y	Y	Y
3	Ngắt theo sườn xuống của I0.1		Y	Y	Y
4	Ngắt theo sườn lên của I0.2		Y	Y	Y
5	Ngắt theo sườn xuống của I0.2		Y	Y	Y
6	Ngắt theo sườn lên của I0.3		Y	Y	Y
7	Ngắt theo sườn xuống của I0.3		Y	Y	Y
8	Ngắt để nhận kí tự ở Port 0	Y	Y	Y	Y
9	Ngắt để báo việc truyền dữ liệu đã hoàn tất ở Port 0	Y	Y	Y	Y
10	Ngắt thời gian 0	Y	Y	Y	Y
11	Ngắt thời gian 1		Y	Y	Y
12	Ngắt theo HSC0, khi giá trị tức thời bằng giá trị đặt trước*.	Y	Y	Y	Y
13	Ngắt theo HSC1, khi giá trị tức thời bằng giá trị đặt trước*.	Y	Y	Y	Y
14	Ngắt theo HSC1, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.		Y	Y	Y
15	Ngắt theo HSC1, khi có tín hiệu Reset từ ngoài		Y	Y	Y
16	Ngắt theo HSC2, khi giá trị tức thời bằng giá trị đặt trước*.		Y	Y	Y
17	Ngắt theo HSC2, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.		Y	Y	Y
18	Ngắt theo HSC2, khi có tín hiệu Reset từ ngoài		Y	Y	Y
19	PLS0 Ngắt báo hoàn tất việc đếm xung		Y	Y	Y
20	PLS1 Ngắt báo hoàn tất việc đếm xung		Y	Y	Y
21	Ngắt theo bộ định thời T32, khi giá tức thời CT=PT.			Y	Y
22	Ngắt theo bộ định thời T96, khi giá tức thời CT=PT.			Y	Y
23	Ngắt báo hoàn tất việc nhận 1 gói tin ở Port 0			Y	Y
24	Ngắt báo hoàn tất việc nhận 1 gói tin ở Port 1				Y
25	Ngắt để nhận kí tự ở Port 1				Y
26	Ngắt để báo việc truyền dữ liệu đã hoàn tất ở Port 1				Y

**Nếu khai báo kiểu ngắt 12 (HSC0, PV=CV) thì hai kểu ngắt 0 và 1 bị vô hiệu hoá. Ngược lại, nếu sử dụng kiểu ngắt 0 và 1 thì kiểu ngắt 12 bị vô hiệu hoá.*

Bảng 3.8: Liệt kê các tín hiệu báo ngắt tương ứng với từng loại CPU 22x

Kiểu ngắt	Mô tả tín hiệu ngắt	CPU 221	CPU 222	CPU 214, 224XP	CPU 226, 226XM
0	Ngắt theo sườn lên của I0.0	Y	Y	Y	Y
1	Ngắt theo sườn xuống của I0.0	Y	Y	Y	Y
2	Ngắt theo sườn lên của I0.1	Y	Y	Y	Y
3	Ngắt theo sườn xuống của I0.1	Y	Y	Y	Y
4	Ngắt theo sườn lên của I0.2	Y	Y	Y	Y
5	Ngắt theo sườn xuống của I0.2	Y	Y	Y	Y
6	Ngắt theo sườn lên của I0.3	Y	Y	Y	Y
7	Ngắt theo sườn xuống của I0.3	Y	Y	Y	Y
8	Ngắt để nhận kí tự ở Port 0	Y	Y	Y	Y
9	Ngắt để báo việc truyền dữ liệu đã hoàn tất ở Port 0	Y	Y	Y	Y
10	Ngắt thời gian 0, SNB34	Y	Y	Y	Y
11	Ngắt thời gian 1, SMB35	Y	Y	Y	Y
12	Ngắt theo HSC0, khi giá trị tức thời bằng giá trị đặt trước CV=PV.	Y	Y	Y	Y
13	Ngắt theo HSC1, khi giá trị tức thời bằng giá trị đặt trước CV=PV.			Y	Y
14	Ngắt theo HSC1, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.			Y	Y
15	Ngắt theo HSC1, khi có tín hiệu Reset từ ngoài			Y	Y
16	Ngắt theo HSC2, khi giá trị tức thời bằng giá trị đặt trước CV=PV.			Y	Y
17	Ngắt theo HSC2, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.			Y	Y
18	Ngắt theo HSC2, khi có tín hiệu Reset từ ngoài			Y	Y
19	PLS0 Ngắt báo hoàn tất việc đếm xung	Y	Y	Y	Y
20	PLS1 Ngắt báo hoàn tất việc đếm xung	Y	Y	Y	Y
21	Ngắt theo bộ định thời T32, khi giá tức thời CT=PT.	Y	Y	Y	Y
22	Ngắt theo bộ định thời T96, khi giá tức thời CT=PT.	Y	Y	Y	Y
23	Ngắt báo hoàn tất việc nhận 1 gói tin ở Port 0	Y	Y	Y	Y
24	Ngắt báo hoàn tất việc nhận 1 gói tin ở Port 1				Y
25	Ngắt để nhận kí tự ở Port 1				Y
26	Ngắt để báo việc truyền dữ liệu đã hoàn tất ở Port 1				Y
27	Ngắt theo HSC0, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.	Y	Y	Y	Y
28	Ngắt theo HSC0, khi có tín hiệu Reset từ ngoài	Y	Y	Y	Y
29	Ngắt theo HSC4, khi giá trị tức thời bằng giá trị đặt trước CV=PV.	Y	Y	Y	Y
30	Ngắt theo HSC4, khi có tín hiệu báo đổi hướng đếm từ bên ngoài.	Y	Y	Y	Y
31	Ngắt theo HSC4, khi có tín hiệu Reset từ ngoài	Y	Y	Y	Y
32	Ngắt theo HSC3, khi giá trị tức thời bằng giá trị đặt	Y	Y	Y	Y

	trước CV=PV.				
33	Ngắt theo HSC5, khi giá trị tức thời bằng giá trị đặt trước CV=PV.	Y	Y	Y	Y

Thứ tự ưu tiên (priority) và hàng đợi (Queuing) của các kiểu ngắt:

Thứ tự ưu tiên của các kiểu ngắt khác nhau đã được cứng hoá từ trước theo nguyên tắc tín hiệu nào có trước thì xử lý trước. Nếu cùng một lúc có nhiều tín hiệu báo ngắt thì hệ thống sẽ sắp hàng đợi theo thứ tự ưu tiên sau:

Nhóm ngắt truyền thông (nối tiếp).

Nhóm ngắt vào ra (kể cả ngắt cho bộ đếm HSC và ngắt truyền xung).

Nhóm các tín hiệu báo ngắt thời gian.

Tại mỗi thời điểm chỉ có 1 chương trình xử lý ngắt được thực hiện. Cũng nói thêm rằng, nhóm ngắt truyền thông có vị trí ưu tiên cao nhất và ngắt thời gian có vị trí ưu tiên thấp nhất nhưng khi hệ thống đang xử lý ngắt thời gian mà có tín hiệu báo nhắt thời gian thì hệ thống vẫn tiếp tục xử lý đến khi kết thúc mới tiếp tục xử lý ngắt truyền thông.

Bảng hàng đợi lớn nhất mà từng CPU có thể có:

Nhóm ưu tiên	212	214	215	216	221	222	224	226
Ngắt truyền thông	4	4	4	8	4	4	4	8
Ngắt vào ra	4	16	16	16	16	16	16	16
Ngắt thời gian	2	4	8	8	8	8	8	8

Riêng đối với tín hiệu báo ngắt truyền thông, mặc dù chưa được xử lý, nhưng kí tự nhận được cùng bit kiểm tra chẵn lẻ vẫn được ghi nhớ lại trong bộ đệm kèm theo đúng thứ tự của tín hiệu báo ngắt.

bit Start	7 hoặc 8 bit của kí tự	Parity	Stop

Khi hàng đợi đã đầy thì bit báo tràn tương ứng cho từng nhóm ngắt sẽ set lên 1:

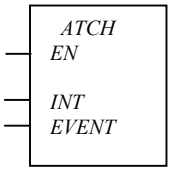
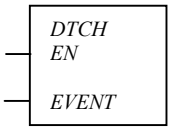
Nhóm ưu tiên	Bit báo tràn
Ngắt truyền thông	SM4.0
Ngắt vào ra	SM4.1
Ngắt thời gian	SM4.2

Cùng với việc chuyển vào chế độ RUN của PLC, tất cả các chế độ ngắt trước đã khai báo trước đó sẽ tự động huỷ (vô hiệu hoá). Nó được kích lại bằng lệnh ENI (kích ngắt toàn cục).

Khai báo một chế độ ngắt phải thực hiện hai việc:

1. Kích tín hiệu báo ngắt cho chế độ ngắt tương ứng (bằng cách khai báo tại toán hạng EVENT) bằng lệnh ATCH.
2. Sau đó soạn thảo nội dung của chương trình ngắt trong khối INT_x.

Có thể gọp nhiều tín hiệu báo ngắt vào cùng một chương trình (chính hoặc con) nhưng một tín hiệu báo ngắt chỉ có duy nhất một chương trình xử lý ngắt. Khi huỷ tín hiệu ngắt bằng lệnh DISI thì các ngắt vẫn tiếp tục nằm vào hàng đợi cho đến khi chúng được kích klại bằng lệnh ENI.

STL	LAD	Mô tả Description	Toán hạng Operands	Kiểu dữ liệu Data Types
Attach Interupt				
ATCH INT, EVENT		Lệnh khai báo ngắt mã hiệu INT (khối ngắt), Kiểu ngắt EVENT	**INT:** 0 ÷ 127 **EVENT:** xem bảng liệt kê các tín hiệu báo ngắt tương ứng với từng loại CPU	Byte
Detach Interupt				
DTCH EVENT		Lệnh huỷ ngắt cục bộ tương ứng với kiểu ngắt EVENT.	**EVENT:** xem bảng liệt kê các tín hiệu báo ngắt tương ứng với từng loại CPU	Byte
Enable Interupt				
ENI		Lệnh kích ngắt toàn cục.	none	none
Disable Interupt				
DISI		Lệnh huỷ tất cả các ngắt cùng một lúc.	none	none
Conditional Return from Interupt				
CRETI		Lệnh thoát tức thời khỏi chương trình ngắt khi chương trình ngắt chưa kết thúc.	none	none
Return from Interupt				
RETI		Lệnh kết thúc chương trình xử lý ngắt, ở cuối chương trình.	none	none

Chương trình xử lý ngắt:

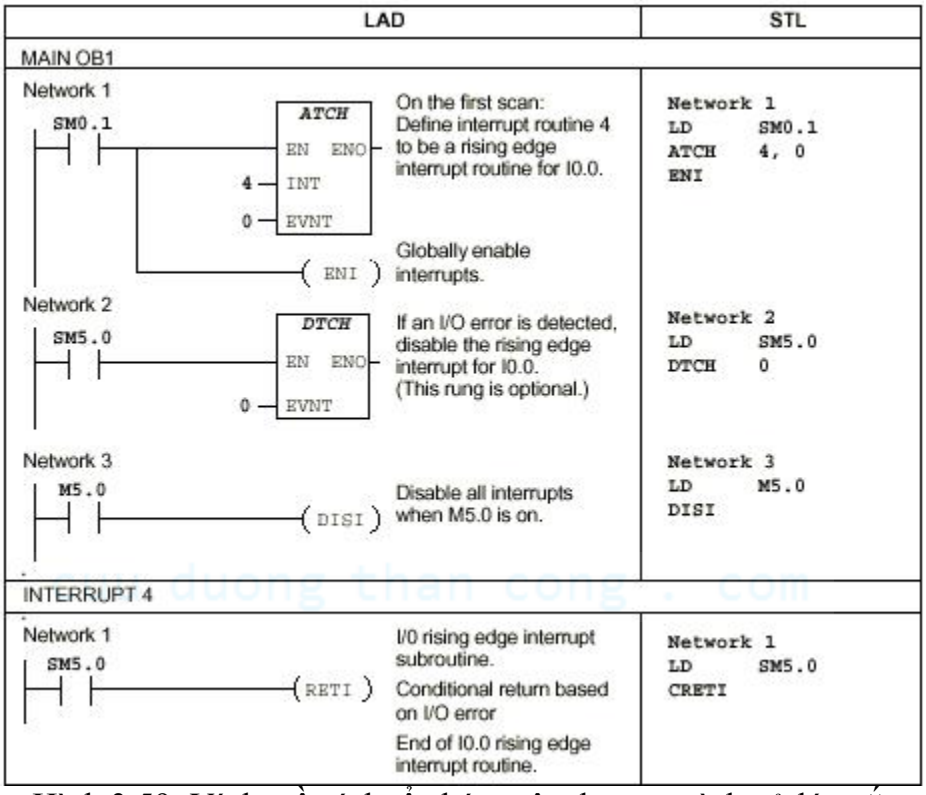
Cũng như chương trình con, mỗi chương trình xử lý ngắt có một nhãn riêng được đánh dấu tại điểm đầu của chương trình. Nhãn này được khai báo bằng lệnh INT.

Tất cả các lệnh nằm giữa nhãn của chương trình xử lý ngắt và lệnh quay về không điều kiện RETI của chương trình xử lý ngắt đều thuộc về nội dung của chương trình xử lý ngắt. Có thể kết thúc chương trình xử lý ngắt sớm hơn bằng lệnh CRETI, nhưng lệnh RETI vẫn là lệnh kết thúc của chương trình xử lý ngắt. Nhưng lệnh này không cần khai

báo vì chương trình STEP đã tự động khai báo giống như lệnh MEND (kết thúc chương trình chính), lệnh RET (lệnh kết thúc chương trình con).

Chương trình xử lý ngắt cần phải được viết tối ưu, càng nhanh càng tốt, không nên thực hiện chương trình xử lý ngắt quá lâu.

Không được sử dụng các lệnh sau trong CTXLN: DISI, ENI, CALL, HDEF, FOR...NEXT, END.



Hình 3.59: Ví dụ về cách tổ chức một chương trình xử lý ngắt

Ngắt tryền thôngnối tiếp:

Cổng truyền thông nối tiếp của PLC có thể điều khiển bằng chương trình viết trong LAD, STL. Chương trình điều khiển này gọi là điều khiển cổng tự do (*Freeport Control*). Trước khi thực hiện quá trình truyền thông, các vấn đề sau đây cần phải được thực hiện:

Kiểu biên bản truyền/nhận (giao thức truyền_Protocol).

Tốc độ truyền/nhận tín hiệu.

Số bit được truyền cho 1 kí tự (7 or 8 bit).

Chế độ kiểm tra lỗi (cho kí tự nhận) chẵn lẻ Parity.

Tất cả các vấn đề này được định nghĩa trong byte đặc biệt SMB30 sau:

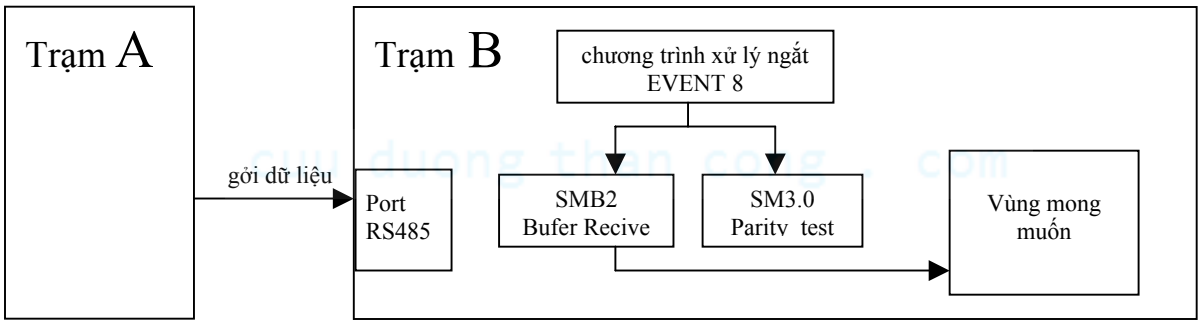
Port 0	Port 1	Description
Format of SMB30	Format of SMB130	MSB 7 … LSB 0 p p d b b b m m — Freeport mode control byte
SM30.6 and SM30.7	SM130.6 and SM130.7	pp: Parity select 00 = no parity 01 = even parity 10 = no parity 11 = odd parity
SM30.5	SM130.5	d: Data bits per character 0 = 8 bits per character 1 = 7 bits per character
SM30.2 to SM30.4	SM130.2 to SM130.4	bbb: Freeport Baud rate 000 = 38,400 baud 001 = 19,200 baud 010 = 9,600 baud 011 = 4,800 baud 100 = 2,400 baud 101 = 1,200 baud 110 = 600 baud 111 = 300 baud
SM30.0 and SM30.1	SM130.0 and SM130.1	mm: Protocol selection 00 = Point-to-Point Interface protocol (PPI/slave mode) 01 = Freeport protocol 10 = PPI/master mode 11 = Reserved (defaults to PPI/slave mode)
Note: One stop bit is generated for all configurations.		

Hình 3.60: Mô tả byte định nghĩa việc truyền thông nối tiếp

! Khi truyền thông ở chế độ Freeport thì PLC không làm việc với máy lập trình PG.

- Byte SMB2 làm bộ đệm ghi nhớ kí tự nhận được.
- Bit SM3.0 dùng để kiểm tra lỗi chẵn lẻ kí tự nhận được, nếu có lỗi chẵn lẻ được phát hiện thì SM3.0 set lên 1.
- Sử dụng để thông báo việc truyền thông đã hoàn tất.

Các vấn đề về gởi/nhận message được mô tả như sau:



Hình 3.61: Mô tả cách nhận message của PLC