# I. CẤU TRÚC VÀ CHỨC NĂNG CÁC VÙNG NHỚ

| Area | | | Số lượng | Dài địa chỉ | Sử dụng với Task | Phân cho mục đích sử dụng | Truy cập dạng bit | Truy cập dạng Word | Cho phép | | Thay đổi từ CX-Programmer | Khả năng Force Bit |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | Đọc | Ghi | | |
| CIO Area | I/O Area | Input Area | 1,600 bits (100 words) | CIO 0 đến CIO 99 | Được chia sẻ với tất cả các task | CP1L CPU Units & CP-series | OK | OK | OK | OK | OK | OK |
| | | Output Area | 1,600 bits (100 words) | CIO 100 đến CIO 199 | | Expansion Units hay Expansion I/O Units | OK | OK | OK | OK | OK | OK |
| | 1:1 Link Area | | 1,024 bits (64 words) | CIO 3000 đến CIO 3063 | | 1:1 Links | OK | OK | OK | OK | OK | OK |
| | Serial PLC Link Area | | 1,440 bits (90 words) | CIO 3100 đến CIO 3189 | | Serial PLC Links | OK | OK | OK | OK | OK | OK |
| | Work Area | | 14,400 bits (900 words) | CIO 3800 đến CIO 6143 | | --- | OK | OK | OK | OK | OK | OK |
| Work Area | | | 8,192 bits (512 words) | W000 đến W511 | | --- | OK | OK | OK | OK | OK | OK |
| Holding Area | | | 8,192 bits (512 words) | H000 đến H511 (Ghi chú 6) | | --- | OK | OK | OK | OK | OK | OK |
| Auxiliary Area | | | 15,360 bits (960 words) | A000 đến A959 | | --- | OK | --- | OK | Ghi chú 1 | Ghi chú 1 | Không |
| TR Area | | | 16 bits | TR0 đến TR15 | | --- | OK | OK | OK | OK | Không | Không |
| Data Memory Area | | | 32,768 words | D00000 đến D32767 (Ghi chú 7) | | --- | Không (Ghi chú 2) | OK | OK | OK | OK | Không |
| Timer Completion Flags | | | 4,096 bits | T0000 đến T4095 | | --- | OK | --- | OK | OK | OK | OK |
| Counter Completion Flags | | | 4,096 bits | C0000 đến C4095 | | --- | OK | --- | OK | OK | OK | OK |
| Timer PVs | | | 4,096 words | T0000 đến T4095 | | --- | --- | OK | OK | OK | OK | Không (Ghi chú 4) |
| Counter PVs | | | 4,096 words | C0000 đến C4095 | | --- | --- | OK | OK | OK | OK | Không (Ghi chú 5) |
| Task Flag Area | | | 32 bits | TK0 đến TK31 | | --- | OK | --- | OK | Không | Không | Không |
| Index Registers | | | 16 registers | IR0 đến IR15 | Chức năng riêng cho từng task (Ghi chú 3) | --- | OK | OK | Chỉ dùng cho đánh địa chỉ gián tiếp (Indirect addressing) | Tùy từng lệnh | Không | Không |
| Data Registers | | | 16 registers | DR0 đến DR15 | | --- | Không | OK | OK | OK | Không | Không |

## Ghi chú:

(1) A0 đến A447 chỉ cho phép đọc, cấm ghi. A448 đến A959 cho phép đọc/ghi (read/write)
(2) Bit này có thể được tác động bởi các lệnh TST(350), TSTN(351), SET, SETB(532), RSTB(533), & OUTB(534).
(3) Index registers & data registers có thể được dùng riêng cho từng task hay chung cho tất cả các task.
(4) Timer PVs có thể được làm tươi gián tiếp bằng cách force-setting/resetting Timer Completion Flags.
(5) Counter PVs có thể được làm tươi gián tiếp bằng cách force-setting/resetting Counter Completion Flags.
(6) H512 đến H1535 được dùng trong Function Block Holding Area. Các words có thể được dùng nội bộ bên trong các lệnh gọi function block
(7) Data Memory Area cho CPU Units với 10, 14 hay 20 I/O Points: D0 đến D9999 và D32000 đến D32767.

## 1- Vùng nhớ CIO (Common I/O) hay IR (Internal Relay):

**Vùng nhớ Input/Output**
Những bit trong vùng nhớ này dùng để đặt các địa chỉ vào/ra (I/O), nó chỉ các trạng thái ON/OFF của các tín hiệu vào/ra. Các địa chỉ không dùng cho chức năng I/O có thể sử dụng như work bit trong khi viết chương trình.

**Vùng nhớ 1:1 Link Area**

**Vùng nhớ Serial PLC Link Area**

**Vùng nhớ Work bit**
Các Work bit có thể được sử dụng tự do trong chương trình. Chúng chỉ sử dụng cho các mục đích bit/word trung gian trong chương trình, không thể gán cho các I/O bên ngoài.

## 2- Vùng nhớ Work Area:
Các bit và word trong vùng Work area có thể được sử dụng tự do trong chương trình. Chúng chỉ sử dụng cho các mục đích bit/word trung gian trong chương trình, không thể gán cho các I/O bên ngoài.

## 3- Vùng nhớ TR (Temporary Relay):
Sử dụng khi một sơ đồ ladder phức tạp cần phải rẽ nhánh, TR sẽ chứa tạm thời các trạng thái On/Off ở các nhánh chương trình.
TR chỉ sử dụng khi lập trình bằng mã Mnemonic. Khi lập trình bằng Ladder, TR sẽ thực hiện một cách tự động.

## 4- Vùng nhớ HR (Hold Relay):

Các bit HR sẽ giữ trạng thái On/Off không đổi, ngay cả khi không cấp nguồn cho PLC.

## 5- Vùng nhớ AR ( Auxiliary Relay):
Những bit này chủ yếu phục vụ như cờ (flag), các trạng thái hoạt động của PLC.

## 6- Vùng nhớ Timer:
Quản lý timer được tạo ra từ các lệnh TIM, TIMH(15)
TIM dùng để truy cập cờ (nếu sử dụng bit) và giá trị hiện thời (PV) (nếu sử dụng word) của Timer

## 7- Vùng nhớ Counter:
Quản lý counter được tạo ra từ các lệnh CNT, CNTR(12),..
CNT dùng để truy cập cờ (nếu sử dụng bit) và giá trị hiện thời (PV) (nếu sử dụng word) của Counter.

## 8- Vùng nhớ DM (Data Memory):

DM chỉ có thể truy cập theo Word.
DM được chia ra hai nhóm: Nhóm sử dụng chứa các dữ liệu một cách tự do và nhóm dùng cho các chức năng đặc biệt.

**9- Task Flag Area**

1 cờ Task Flag sẽ lên ON khi cyclic task (task theo chu kỳ) tương ứng ở trạng thái sẵn sàng chạy (RUN) và OFF khi cyclic task chưa được thực hiện (INI) hoặc ở trạng thái chờ standby (WAIT).

**10- Index Registers**

Index registers (IR0 đến IR15) được dùng để lưu địa chỉ bộ nhớ PLC (địa chỉ tuyệt đối trong RAM) để đánh địa chỉ gián tiếp. Chúng có thể dùng riêng trong từng task hay chung cho tất cả các task.

**11- Data registers**

Data registers (DR0 đến DR15) được dùng kết hợp với Index Registers trong 1 câu lệnh để xác định địa chỉ thực cần dùng, trong đó nội dung của Data registers được cộng với nội dung của Index Registers để có địa chỉ thực. Chúng có thể dùng riêng trong từng task hay chung cho tất cả các task.

## II - LỆNH CƠ BẢN KHÁC (Basic Instruction) :

Ngoài các lệnh cơ bản như:

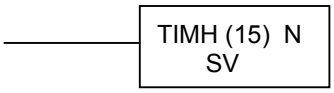
1. **Load – LD**
2. **Load Not – LD NOT**
3. **And**
4. **And Not**
5. **Or**
6. **Or Not**
7. **And Load – AND LD**
8. **Or Load – OR LD**
9. **Out, Out Not : Ngõ ra on, off khi điều kiện được thực hiện.**
10. **RESET B : Khi thực hiện bit B OFF.**
11. **SET B : Khi thực hiện bit B ON.**
12. **Keep**
13. **Timer**
14. **Counter**
15. **Reversible Counter**
16. **Differentiate Up/Down**
17. **End (01) : Điều kiện bắt buộc khi kết thúc chương trình.**

PLC OMRON còn hỗ trợ nhiều lệnh khác cho các yêu cầu điều khiển rất đa dạng trong thực tế.

Sau đây là 1 số lệnh khác.

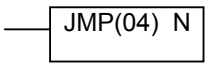
***Chú ý : Số thứ tự của TIM Và CNT không được trùng nhau, tổng số TIM Và CNT là 511 đối với* CQM1, 127 *đối với CPM1, 256 đối với CPM2A.***

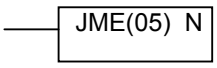
**18. High-Speed Timer** :




SV : 0.01 đến 99.99 Sec

**19. Jump – Jump End :**




**Jump 00 :**

**Nếu N=00, CPU sẽ tìm JME(05) kế đó với số thứ tự tương ứng là 00.** Khi thực hiện việc tìm kiếm này, chu kỳ quét của chương trình sẽ dài hơn so với thực hiện các Jump với số thứ tự N # 0.

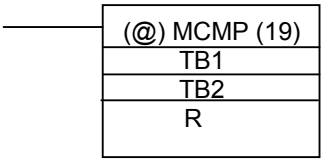
Trạng thái của Timer, Counter, Output bit, và tất cả các trạng thái khác của những lệnh ở giữa JMP(04) và JME(05) sẽ không thay đổi. Jump có số thứ tự 00 có thể dùng nhiều lần trong một chương trình.

- *DIFU(13) và DIFD(14) trong lệnh Jump:*
  Giả sử một bit ON bởi DIFU(13) hay DIFD(14) đặt trong JMP(04) và JME(05), đến chu kỳ quét kế tiếp, nếu điều kiện của Jump là OFF, bit đó sẽ vẫn giữ trạng thái ON cho đến khi điều kiện của Jump bật lên ON ( tức là khi chương trình không thực hiện rẽ nhánh ).

***Chú ý: Khi JMP(04) và JME(05) không sử dụng theo từng cặp, một thông báo lỗi (Error) sẽ hiển thị*** *khi thực hiện việc kiểm tra chương trình, tuy nhiên chương trình vẫn hoạt động bình thường.*

## III- LỆNH SO SÁNH DỮ LIỆU :

### 1. Multi-Word compare

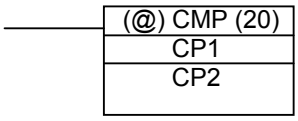



(CPM1 không có lệnh này).
TB1 đến TB1+15 phải trên cùng vùng dữ liệu.
TB2 đến TB2+15 phải trên cùng vùng dữ liệu.

MCMP(19) so sánh nội dung của TB1 và TB2, TB1+1 và TB2+2, ...., TB1+15 và TB2+15. Kết quả của bit tương ứng trên word R sẽ Off nếu hai word so sánh tương ứng bằng nhau .

- Bit P_EQ On nếu nội dung cả hai TB1 và TB2 bằng nhau :R = 0000.
- Bit P_ER On nếu xảy ra lỗi khi thực hiện lệnh .

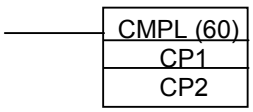
### 2. Compare :




So sánh nội dung của hai word CP1 và CP2 :

- CP1 < CP2 : LE (P_LE On)
- CP1 = CP2 : EQ (P_EQ On)
- CP1 > CP2 : P_GT (P_GT On)

### 3. Double Compare :

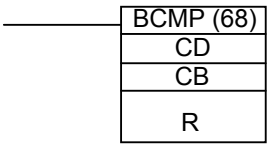



CP1 và CP1+1 phải trên cùng vùng dữ liệu.
CP2 và CP2+1 phải trên cùng vùng dữ liệu.

CMPL (60) nối 4 digit của CP1+1 và CP1 thành một 2 word (8 digits ) , Cp2+1 và CP2 thành hai word (8 digits) , trong đó những bit của CP1+1 và CP2+1 là những bit có trọng số lớn nhất. Tiếp theo CMPL (60) sẽ so sánh nội dung của hai word này :

- CP1+1,CP1 < CP2+1,CP2 : LE (P_LE On)
- CP1+1,CP1 = CP2+1,CP2 : P_EQ (P_EQ On)
- CP1+1,CP1 > CP2+1,CP2 : P_GT (P_GT On)

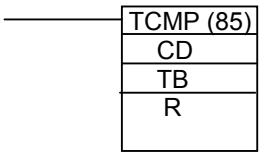
### 4. Block Compare :




CD : Dữ liệu được so sánh
CB : Word đầu tiên của khối word (block word) cần so sánh .(Nội dung của word giới hạn nhỏ nhất phải nhỏ hơn hay bằng nội dung của word giới hạn lớn nhất )
R : Word trả về kết quả. ( DM 6144 đến DM 6655 không được sử dụng ).

BCMP (68) thực hiện so sánh CD với những khoảng được tạo ra trong khối word bắt đầu từ CB. Kết quả trả về bit tương ứng trong R.
CB<=CD<=CB+1 : Bit 00
CB+2<=CD<=CB+3 : Bit 01
CB+3<=CD<=CB+4 : Bit 02
.....
CB+30<=CD<=CB+31: Bit 15

### 5. Table Compare :

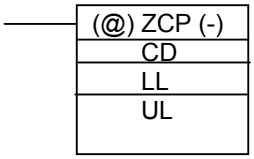



CD : Dữ liệu được so sánh
TB : Word đầu tiên trong bảng cần so sánh
R : Word trả về kết quả

TCMP (85) thực hiện so sánh CD với nội dung của các word trong bảng TB, TB+1,..., TB+15. Nếu CD bằng bất kỳ nội dung của word nào trong bảng thì bit tương ứng của R sẽ On .

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

### 6. Area Range Compare :




*Lệnh này chỉ thực hiện cho PLC loại CJ1M, CJ1, CS1. Các cờ LE, P_EQ & P_GT là các cờ đặc biệt ở vùng nhớ riêng.*

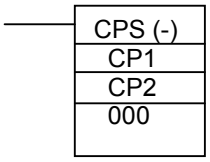
CD : Dữ liệu được so sánh
LL : Giới hạn dưới

UL : Giới hạn trên.
LL : Phải nhỏ hơn hay bằng UL

- CD < LL : LE (P_LE On)
- LL <=CD <=UL : P_EQ (P_EQ On)
- UL<CD : P_GT (P_GT On)

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

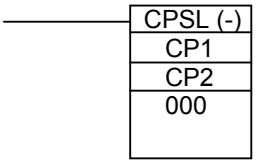
### 7. Signed Binary Compare :




*Lệnh này chỉ thực hiện cho PLC loại CJ1M, CJ1, CS1. Các cờ LE, P_EQ & P_GT là các cờ đặc biệt ở vùng nhớ riêng*

CP1 : Word so sánh thứ nhất
CP2 : Word so sánh thứ hai
000 : Không sử dụng.
CPS (-) so sánh 16 bit nhị phân có dấu (giá trị đại số) của CP1 và CP2 , kết quả trả về bit P_LE, P_EQ, P_GT.

### 8. Double Signed binary Compare :

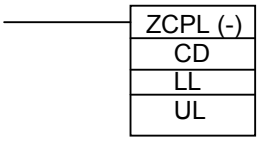



*Lệnh này chỉ thực hiện cho PLC loại CJ1M. Các cờ LE, P_EQ & P_GT là các cờ đặc biệt ở vùng nhớ riêng*

CP1 : Word so sánh thứ nhất
CP2 : Word so sánh thứ hai
000 : Không sử dụng

CPSL (-) so sánh 32 bit có dấu của CP1+1,CP1 cà CP2+1,CP2 . Kết quả trả về bit P_LE, P_EQ, P_GT.

### 9. Double Area Range Compare :

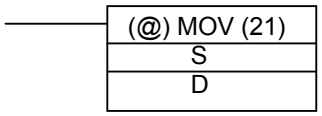



*Lệnh này chỉ thực hiện cho PLC loại CJ1M, CP1L/1H. Các cờ LE, P_EQ & P_GT.*

CD : Dữ liệu được so sánh
LL : Giới hạn dưới của khoảng cần so sánh
UL : Giới hạn trên của khoảng cần so sánh

## IV- LỆNH TRUYỀN DỮ LIỆU :

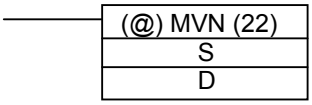
### 1. Move :




MOV(21) copy nội dung của S vào D.

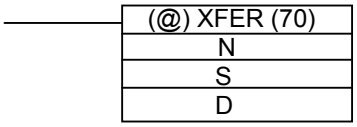
P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.
P_EQ : Bit P_EQ On khi nội dung copy vào D là 0

### 2. Move Not :




MVN(22) copy phủ định của nội dung của S sang D.
P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.
P_EQ : Bit P_EQ On khi nội dung copy vào D là 0

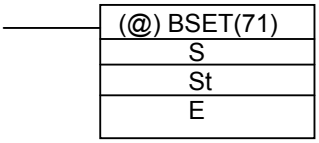
### 3. Block Transfer :




S đến S+N phải trên cùng vùng dữ liệu
D đến D+N phải trên cùng vùng dữ liệu
N : Số Word cần truyền (BCD)

XFER (70) copy nội dung các word S,..., S+N sang D,..., D+N theo thứ tự.
P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

### 4. Block Set :

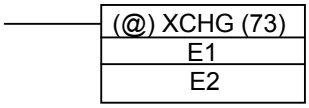



S : Dữ liệu nguồn
St : Word bắt đầu
E : Word kết thúc

St phải nhỏ hơn hay bằng E, St và E phải trên cùng một vùng dữ liệu.

BSET (71) copy nội dung của S đến tất cả các word từ St đến E.
BSET (71) có thể dùng để thay đổi giá trị đặt (PV) của timer/counter (điều này không thể thực hiện bởi MOV, MVN). BSET (71) còn được sử dụng để xoá một vùng dữ liệu, vùng DM bằng cách copy 0 đến tất cả các word của vùng dữ liệu muốn xoá.

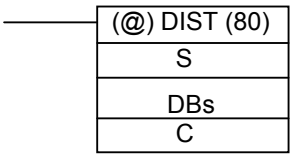
### 5. Data Exchange :




XCHG (73) thực hiện chuyển đổi nội dung giữa hai word E1 và E2.
Có thể chuyển đổi nội dung của block word bằng cách kết hợp với lệnh XFER.

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

### 6. Single Word Distribute :

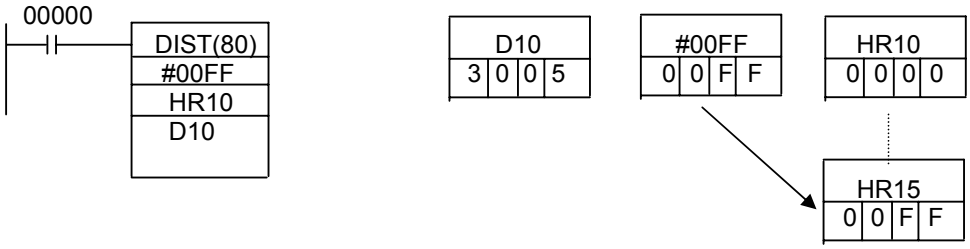



S : Word nguồn.
DBs : Word cần copy
C: Word điều khiển (BCD). ( bit 00 đến 11 là offset – Off )

Nếu bit 12 đến 15 của C bằng 0 ~ 8 (BCD), DIST(80) sẽ copy nội dung của S vào DBs+Of, trong đó nội dung của C chính là Offset.

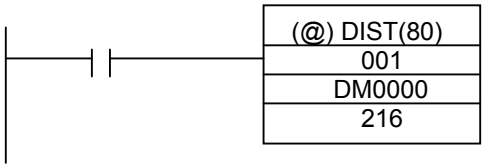
Chú ý: DBs và DBs+Of phải trên cùng vùng dữ liệu

**Td:** DIST(80) copy #00FF vào HR+Of. Nội dung của D10 là #3005, vì thế #00FF được copy vào HR15 (HR10+5) khi IR00000 On.




Nếu bit 12 đến 15 của C bằng 9 (BCD), DIST(80) thực hiện thao tác copy S vào ngăn xếp từ DBs+1 đến DBs+Of được trỏ đến.
DBs : Con trỏ ngăn xếp.
Bit 00 đến 11 của C: Xác định số lượng ngăn xếp. (000 ~999).

**Td:** DIST(80) tạo một ngăn xếp từ DM0001 đến DM0005. DM0000 tác động như một con trỏ ngăn xếp.

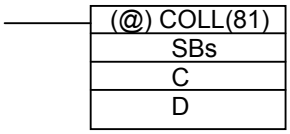



| | |
|---|---|
| IR001 | FFFF |
| IR216 | 9005 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| DM0000 | 0000 | | DM0000 | 0001 | | DM0000 | 0002 | |
| DM0001 | 0000 | First execution | DM0001 | FFFF | Second execution | DM0001 | FFFF | |
| DM0002 | 0000 | → | DM0002 | 0000 | → | DM0002 | FFFF | |
| DM0003 | 0000 | Stact pointer | DM0003 | 0000 | Stact pointer | DM0003 | 0000 | |
| DM0004 | 0000 | incremented | DM0004 | 0000 | incremented | DM0004 | 0000 | |
| DM0005 | 0000 | | DM0005 | 0000 | | DM0005 | 0000 | |

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.
P_EQ : On khi nội dung của S là 0.

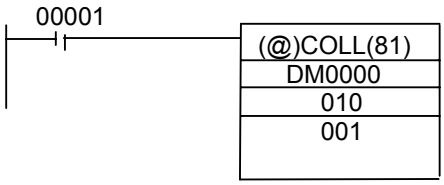
### 7. Data Collect :




SBs : Word nguồn.
C : Word điều khiển ( Bit 00 đến 11 là offset - Of). C phải là BCD.
D : Word cần copy.

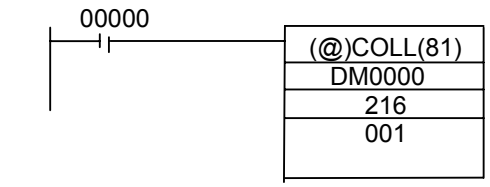
- Nếu bit 12 đến 15 của C = 0 đến 7, nội dung của C là Offset (Of). COLL(81) copy nội dung của SBs+Of vào D. SBs và SBs+Of phải trên cùng vùng dữ liệu.

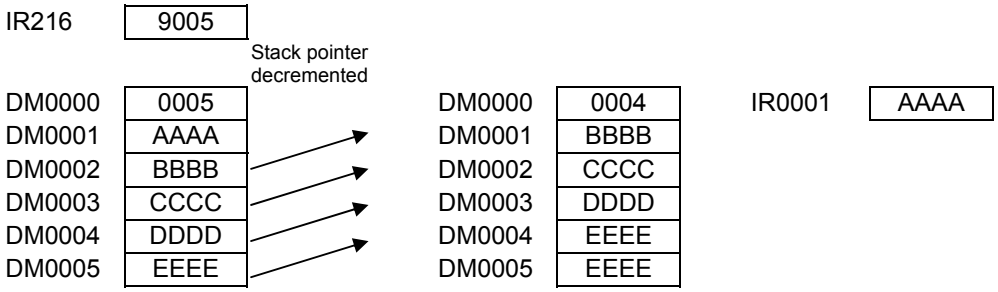
Td: COLL(81) copy nội dung của DM0000+Of vào IR001. Nội dung của 010 là #0005, vì thế nội dung của DM0005 ( DM0000+5) được copy vào IR001 khi điều kiện IR00001 On.




- Nếu bit 12 đến 15 của C bằng 9, COLL(81) thực hiện thao tác FIFO ngăn xếp , trong đó D là word mà nội dung sẽ trả về nội dung của ngăn xếp được trỏ tới, SBs là con trỏ ngăn xếp, bit 00 đến 11 của C xác định số lượng ngăn xếp.

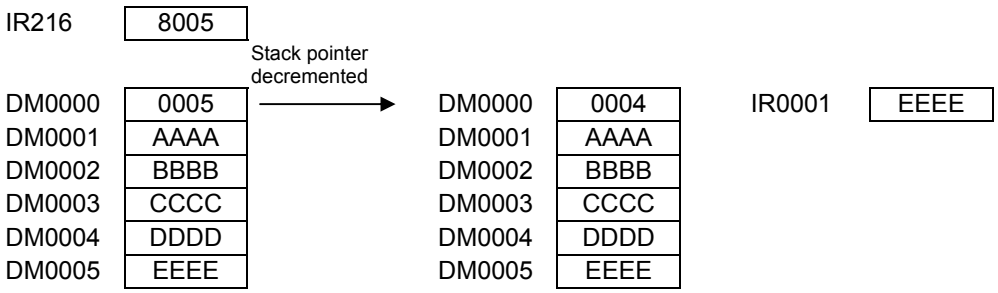
Td: Tạo một ngăn xếp giữa DM0001 và DM0005, DM0000 là con trỏ ngăn xếp.
Khi IR00000 chuyển từ Off sang On, COLL(81) dịch dữ liệu của DM0002 đến DM0005 lên một một địa chỉ, dữ liệu của DM0000 dịch sang IR001, nội dung của con trỏ ngăn xếp giảm đi 1.



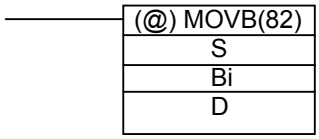

- Nếu bit 12 đến 15 của C bằng 8, COLL(81) thực hiện thao tác LIFO ngăn xếp, trong đó D là word mà nội dung sẽ trả về nội dung của ngăn xếp được trỏ tới, SBs là con trỏ ngăn xếp, bit 00 đến bit 11 của C xác định số lượng ngăn xếp.

**Td**: COLL(81) tạo một ngăn xếp giữa DM0001 và DM0005. DM0000 là con trỏ ngăn xếp.
Khi IR 00000 chuyển từ Off sang On, COLL(81) copy nội dung của DM0005 (DM0000+5) vào IR001. Nội dung của con trỏ ngăn xếp giảm một đơn vị.




P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.
P_EQ : On khi nội dung của SBs là 0.

### 8. Move Bit :



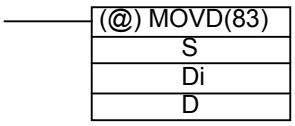

S : Word nguồn.
D : Word cần copy.
Bi : Word chỉ định.(BCD)
Hai số có trọng số nhỏ nhất (LSB) chỉ định số thứ tự của bit cần copy trong S. (00 đến 15)

Hai số có trọng số lớn nhất (MSB) chỉ định số thứ tự của bit được copy trong D.(00 đến 15)

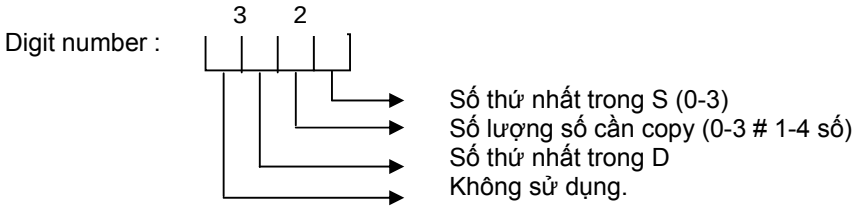
MOVB(82) sẽ copy bit được chỉ định trong S đến bit được chỉ định trong D.

P_ER : Điều kiện thực hiện lệnh không đúng.

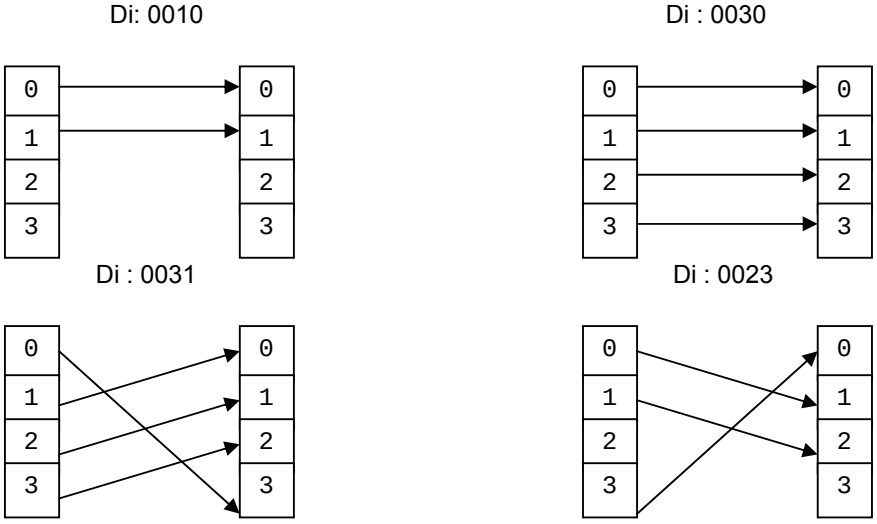
### 9. Move Digit :



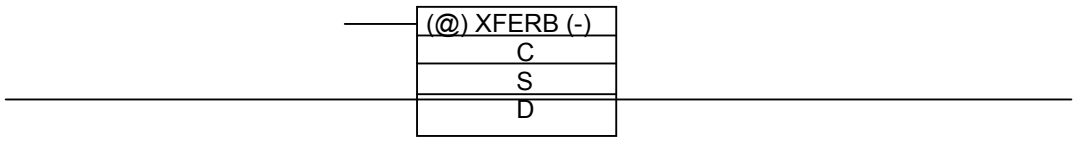

S : Word nguồn.
D : Word cần copy.
Di: Word chỉ định (BCD) .




MOVD(83) copy lần lượt những bit trong S được xác định bởi Di đến những bit trong D cũng được xác định bởi Di.

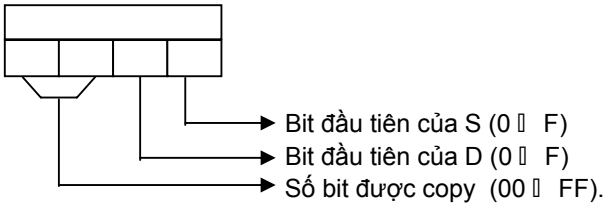



### 10. Transfer Bits :



*Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.*

S : Word nguồn đầu tiên.
D : Word cần copy đầu tiên.
C : Word control.
Hai số bên phải của C đặc trưng cho bit bắt đầu trong S và D, hai bit bên trái xác định số bit được copy.




(Có thể copy đến 255 bits ).
XFRB(-) copy những bit của word nguồn đến những bit của word cần copy được xác định bởi C.

P_ER : Bit P_ER sẽ On nếu điều kiện thực hiện lệnh không đúng.

## V- LỆNH CHUYỂN ĐỔI :

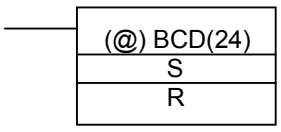
### 1. BCD-To-Binary :


S : Word nguồn ; R : Word kết quả.

BIN(23) chuyển đổi nội dung BCD của S sang dạng nhị phân (Binary), kết quả trả về Word R. Sau khi lệnh thực hiện , nội dung của R bị thay đổi, nội dung của S vẫn giữ nguyên.

P_ER : On khi nội dung của S không phải dạng BCD hay địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

### 2. Binary-To-BCD :



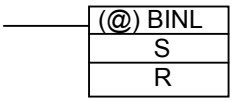

S : Word nguồn ; R : Word kết quả.

Nếu S chứa giá trị lớn hơn 270F, kết quả chuyển đổi sẽ lớn hơn 9999 nên BCD(24) không thể thực hiện được. Khi lệnh không thực hiện được, nội dung trong R vẫn không thay đổi.

BCD (24) chuyển đổi giá trị dạng nhị phân (hexadecimal) của S sang giá trị tương đương dạng BCD, kết quả trả về Word R. Sau khi thực hiện lệnh, chỉ nội dung trong R bị thay đổi, nội dung trong S không thay đổi.

P_ER : On khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

### 2. Double BCD-To-Double Binary :

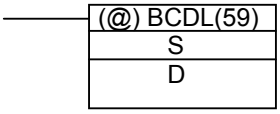



*Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.*

Nội dung trong S phải dưới dạng BCD.
BINL(58) chuyển đổi nội dung dạng BCD của S, S+1 sang dạng nhị phân 32 bits, kết quả trả về R.

P_ER : On khi nội dung trong S, S+1 không phải dạng BCD hoặc khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

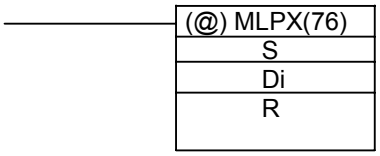
### 3. Double Binary-To-Double BCD :




*Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.*
*Nội dung của S phải dưới dạng nhị phân (Binary).*
*Nếu nội dung của S lớn hơn 05F5E0FF,khi đó kết quả chuyển đổi sẽ lớn hơn 99999999 , BINL(59) sẽ không được thực hiện. Khi lệnh không được thực hiện, nội dung trong R, R+1 vẫn không thay đổi.*

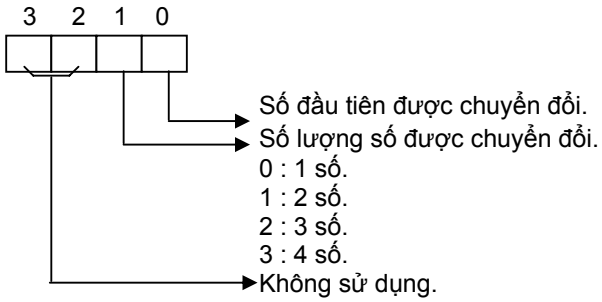
BCDL(59) chuyển đổi nội dung dạng nhị phân 32 bits của S sang dạng BCD, kết quả trả về R.
P_ER : On khi nội dung của R, R+1 vượt quá 99999999, hoặc khi địa chỉ gián tiếp DM không tồn tại.
P_EQ : On khi kết quả là 0.

### 4. 4-To-16 Decoder :



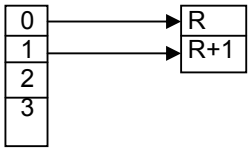

*S : Word nguồn.*
*Di: Word chỉ định. Hai số bên phải của Di phải nằm trong khoảng 0-3.*
*R : Word kết quả đầu tiên. ( Tất cả Word kết quả phải trên cùng vùng dữ liệu ).*

MLPX(76) chuyển đổi 4 số thập lục phân (hexadecimal) trong S sang giá trị thập phân từ 0 đến 15, tương ứng với mỗi giá trị thập phân đó sẽ xác định vị trí chuyển sang On trong R. Word Di sẽ chỉ định vị trí số đầu tiên và số lượng số được chuyển đổi trong S.

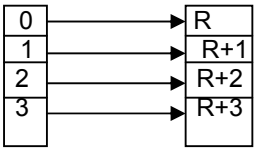



**Ví dụ :**

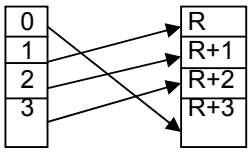
Di: 0010



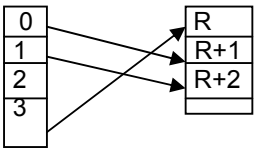

Di: 0030




Di: 0031

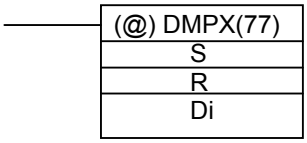



Di: 0023



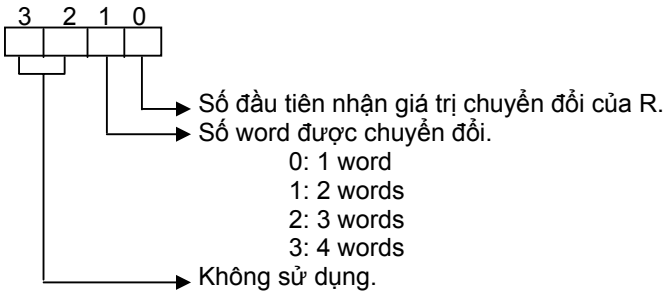

P_ER : Nếu nội dung của Di không thoả, hay số lượng R, R+1,..., vượt ngoài vùng dữ liệu.
Địa chỉ gián tiếp của DM không tồn tại.

### 5. 16-To-4 Encoder :



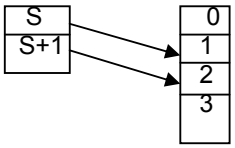

S : Word nguồn thứ nhất.
R : Word kết quả.
Di : Word chỉ định.
Hai số bên phải của Di phải nằm trong khoảng 0-3.
Tất cả word nguồn phải trên cùng vùng dữ liệu.
DMPX(77) xác định bit ở trạng thái On có trọng số lớn nhất trong S, mã hoá vị trí bit đó sang dạng thập lục phân, sau đó truyền giá trị này vào R được xác định bởi Di.

Nội dung trong Di được định nghĩa như sau :

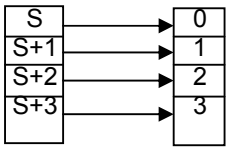



**Ví dụ :**

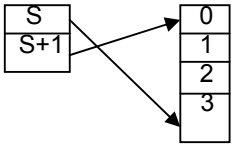
Di: 0011



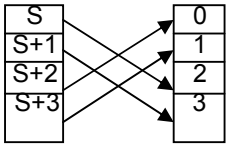

Di: 0030



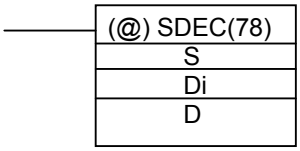

Di: 0013



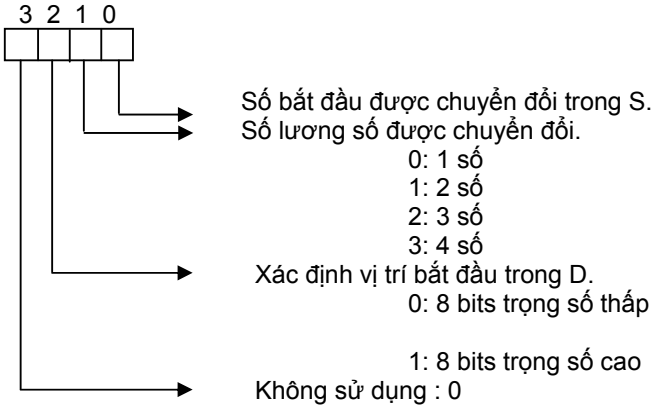

Di: 0032




### 6. 7-Segment Decoder :



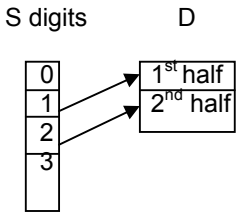

S : Word nguồn.
D : Word đích đầu tiên . ( Tất cả các word D, D+1,..., phải trên cùng một vùng dữ liệu).
Di : Word chỉ định .



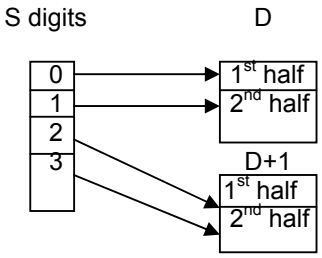

SDEC(78) chuyển đổi mỗi 4 bits trong S (được xác định bởi Di) sang mã 7 đoạn , kết quả trả về D.

**Ví dụ:**

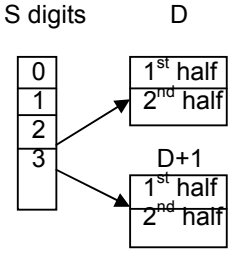
Di: 0011



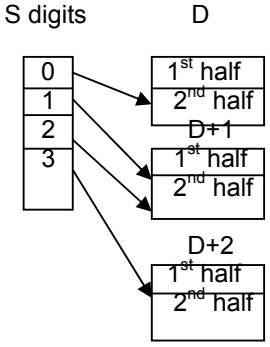

Di: 0030




Di: 0112



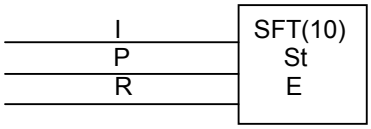

Di: 0130




P_ER : Khi những số trong word chỉ định không đúng, hoặc D vượt ngoài vùng dữ liệu.
Địa chỉ gián tiếp của DM không tồn tại.

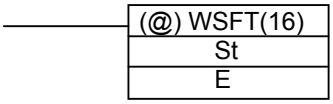
## V- LỆNH DỊCH DỮ LIỆU (Shift Instruction) :

### 1. Shift Register :




St : Word bắt đầu.
E : Word kết thúc.
E phải lớn hơn hay bằng St, E và St phải trên cùng một Word.
I : Điều kiện thực hiện lệnh.
P : Xung tác động.
R : Reset.
Khi xung P thay đổi từ Off sang On (tác động cạnh lên), SFT(10) dịch một bit sang trái. Trạng thái bit đưa vào bit trọng số nhỏ nhất của St là 1 hay 0 tuỳ thuộc vào I On hay Off. Bit trọng số lớn nhất của E sẽ mất đi khi lệnh thực hiện.
Ngõ vào R On lệnh sẽ được reset.

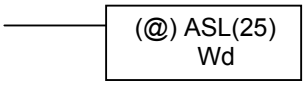
### 2. Word Shift :




E phải lớn hơn hay bằng St, Evà St phải trên cùng vùng dữ liệu.

WSFT(16) dịch nội dung một word sang trái giữa những word từ St đến E. Nội dung của St sau khi lệnh thực hiện sẽ là 0000, nội dung của E bị mất.

### 3. Arithmetic Shift Left :

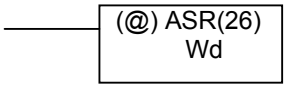



DM 6144 đến DM 6655 không được sử dụng cho Wd.

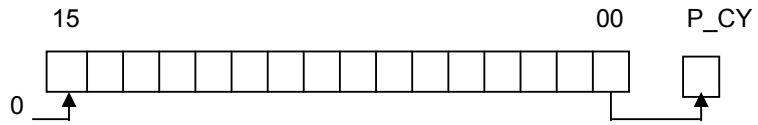
ASL(25) dịch 0 vào bit 00 của Wd, dịch một bit của Wd sang trái, và dịch trạng thái bit 15 vào P_CY.

P_ER : Địa chi gián tiếp của DM không tồn tại.
P_CY : Nhận dữ liệu của bit 00
P_EQ : On khi nội dung của Wd là 0.

### 4. Arithmetic Shift Right :

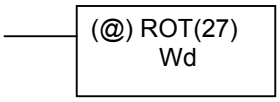



ASR(26) dịch 0 vào bit 15 của Wd trong mỗi chu kỳ quét, dịch một bit của Wd sang phải, và dịch trạng thái bit 00 vào P_CY .

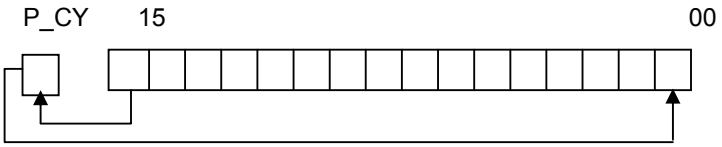



P_ER : On khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : Nhận dữ liệu từ bit 00.
P_EQ : On khi nội dung của Wd là 0.

### 5. Rotate Left :



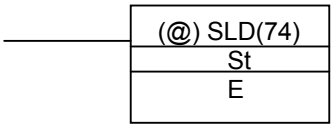

ROL(27) dịch tất cả các bit của Wd sang trái một bit, dịch P_CY vào bit 00, dịch bit15 vào P_CY.




*Chú ý :*
Dùng STC(41) hay CLC(41) để đặt hoặc xóa P_CY trước khi sử dụng ROT(27) để đảm bảo P_CY chứa trạng thái đúng trước khi lệnh thực hiện.
P_CY sẽ được dịch vào bit 15 trong mỗi chu kỳ quét. Dùng @ hay DIFU/DIFD để chỉ dịch một lần nội dung P_CY vào bit 15.

P_ER : Địa chỉ gián tiếp của DM không tồn tại.
P_CY : Nhận dữ liệu của bit 0.
P_EQ : On kho nội dung của Wd là 0.

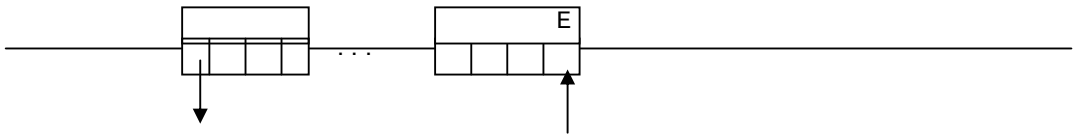
### 6. One Digit Shift Left :




St : Start Word.
E : End Word.

St và E phải trên cùng vùng dữ liệu, E phải lớn hơn hay bằng St.

SLD(74) dịch dữ liệu giữa St và E một digit (4 bits) sang trái. 0 sẽ được viết vào số có trọng số nhỏ nhất của St, nội dung của số có trọng số lớn nhất của E sẽ bị mất.



Dữ liệu sẽ bị mất 0

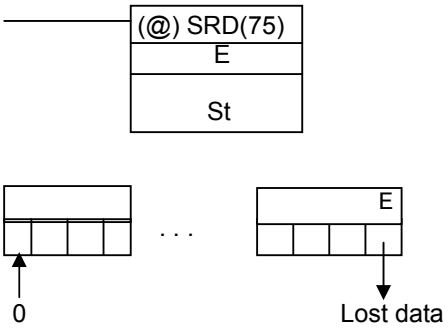
***Chú ý :***

Khi mất nguồn cung cấp trong lúc SLD(74) thực hiện dịch digit qua 50 word, lệnh có thể thực hiện không hoàn thành.

Bit 0 sẽ được dịch vào bit có trọng số nhỏ nhất của St trong mỗi chu kỳ quét.

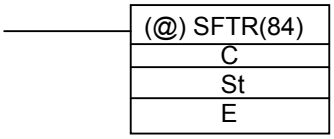
P_ER : khi St và E không trên cùng vùng dữ liệu, địa chỉ gián tiếp của DM không tồn tại.

**7. One Digit Shift Right :**



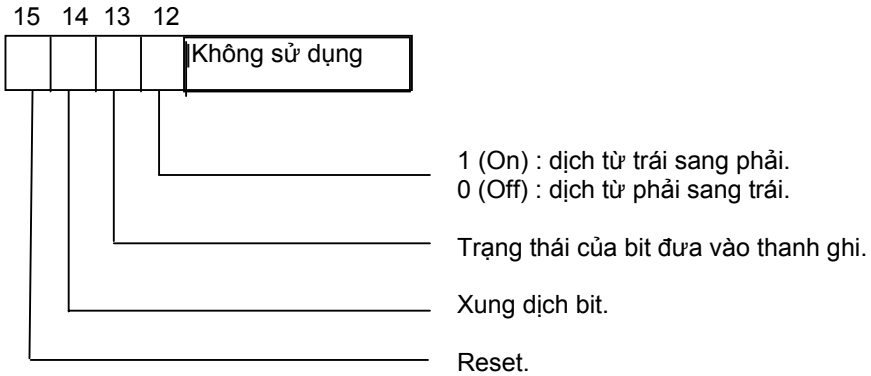

Tương tự SLD(74).

**8. Reversible Shift Register :**




C : Word điều khiển.
St : Word bắt đầu.
E : Word kết thúc.
St và E phải trên cùng vùng dữ liệu. St phải nhỏ hơn hay bằng E.

Nội dung word điều khiển như sau :

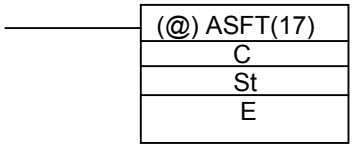



SRD(75) thực hiện dịch bit giữa St và E . Lệnh sẽ dịch trái hay phải, bit đưa vào thanh ghi là 0 hay tuỳ thuộc vào nội dung của word điều khiển C.

P_ER : Khi St và E không cùng nằm trên một vùng dữ liệu.

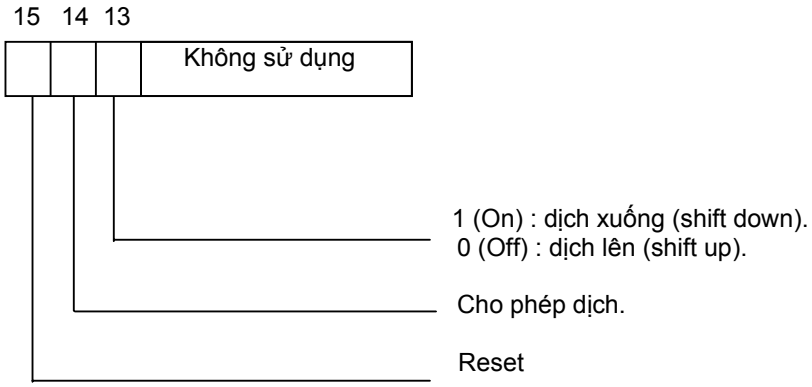
Khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : Nhận trạng thái của bit 00 của St hay bit 15 của bit E tuỳ thuộc vào bit 12 của C.

### 9. Asynchronous Shift Register :




C : Word điều khiển.
St : Word bắt đầu.
E : Word kết thúc.
St và E phải trên cùng vùng dữ liệu. E phải lớn hơn hay bằng St.

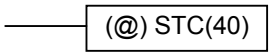
Nội dung của Word điều khiển như sau :




Thanh ghi giữa St và R sẽ có giá trị 0 khi ASFT(17) thực hiện với điều kiện reset On.
ASFT(17) thực hiện dịch ngược không đồng bộ những word trong thanh ghi được xác định bởi St và E. ASFT(17) chỉ thực hiện dịch một word khi word đứng sau nó là 0. Nếu trong thanh ghi không có word nào là 0 thì lệnh sẽ không làm gì cả.
P_ER : Khi St và E không cùng nằm trên một word.
Địa chỉ gián tiếp của DM không tồn tại.

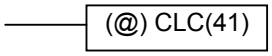
## VI- CÁC LỆNH TÍNH đếnÁN DỮ LIỆU DẠNG BCD :

### 1. Set Carry :



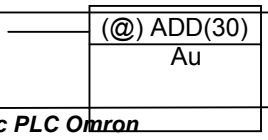

Đặt bit Carry P_CY lên On.

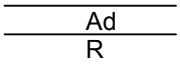
### 2. Clear carry :




Xoá bit Carry P_CY xuống Off.

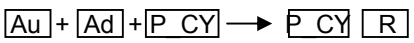
### 3. BCD Add :





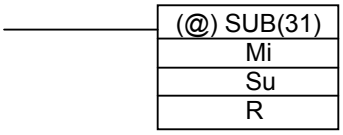

Au : Số được cộng (BCD)
Ad : Số cộng (BCD)
R : Word kết quả.

ADD(30) thực hiện cộng nội dung của Au, Ad, P_CY, kết quả trả về R. P_CY sẽ On khi kết quả lớn hơn 9999.



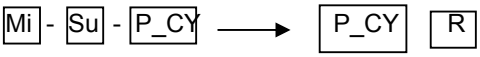

P_ER : Khi nội dung Au và/hoặc Ad không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi có nhớ trong phép cộng.
P_EQ : On khi kết quả là 0.

## 2. BCD Subtract :




Mi : Số bị trừ. (BCD)
Su : Số trừ. (BCD)
R : Word kết quả.

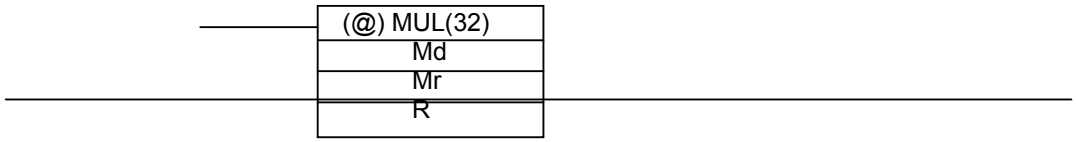
SUB(31) thực hiện trừ nội dung của Mi cho Su và P_CY, kết quả trả về R. Nếu kết quả âm P_CY sẽ On và kết quả trong R chuyển sang dạng bù 10 (10's complement), để có kết quả thật, thực hiện phép trừ 0 cho nội dung của R.




P_ER : Mi và/hoặc Su có nội dung không phải BCD.
Địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi kết quả âm.
P_EQ : On khi kết quả là 0.

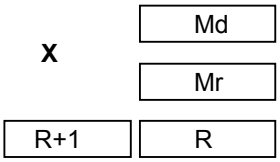
Chú ý : Xoá P_CY trước khi thực hiện SUB(31), kiểm tra P_CY sau khi thực hiện SUB(31) để biết kết quả là âm hay dương .

## 3. BCD Multiply :



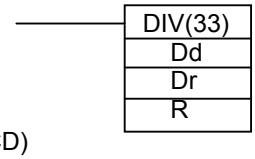
Md : Số được nhân. (BCD)
Mr : Số nhân. (BCD)
R : Wor kết quả.

MUL(32) thực hiện Md ch Mr, kết quả trả về R và R+1.



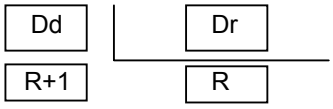

P_ER : Khi nội dung Md và/hoặc Mr không phải dạng BCD.
P_EQ : On khi kết quả là 0.

### 4. BCD Device :



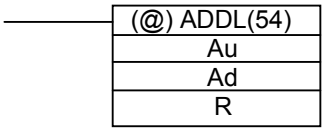

Dd : Số bị chia. (BCD)
Dr : Số chia. (BCD)
R : Word kết quả.

R và R+1 phải trên cùng một vùng dữ liệu.
DIV(33) thực hiện chia Dd cho Dr , kết quả trả về R, số dư trả về R+1.



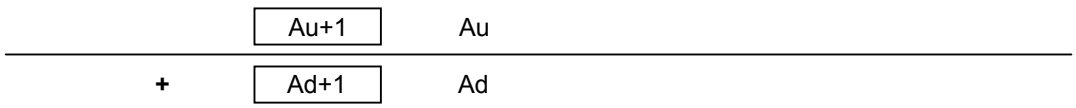

P_ER : Dd hoặc Dr không phải là BCD.
Địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

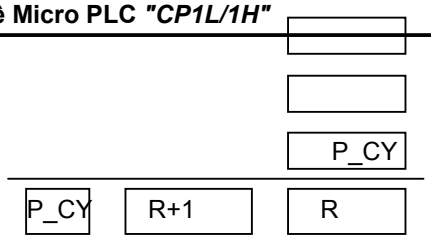
### 5. Double BCD Add :




Au : Word được cộng thứ nhất.
Ad : Word cộng thứ nhất.
R : Word kết quả thứ nhất.

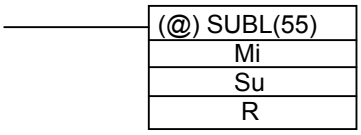
ADDL(54) thực hiện cộng nội dung của P_CY với 8 số được xác định bởi Au, Au+1 và 8 số được xác định bởi Ad, Ad+1 . Kết quả trả về word R, R+1. P_CY On nếu kết quả lớn hơn 99999999.





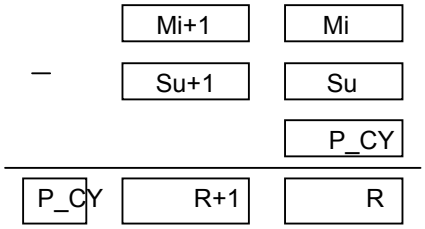

P_ER : Nội dung của Au và/hoặc Ad không phải dạng BCD.
Địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi kết quả lớn hơn 99999999 (Có nhớ trong phép cộng ).
P_EQ : On khi kết quả là 0.

### 6. Double BCD Subtract :



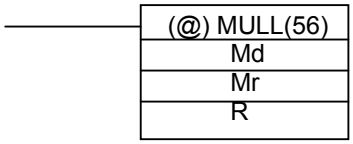

Mi : Word bị trừ đầu tiên. (BCD)
Mu: Word trừ đầu tiên.(BCD)
R : Word kết quả đầu tiên.

SUBL(55) thực hiện phép trừ Mi, Mi+1 cho Mu, Mu+1 và P_CY, kết quả trả về word R, R+1. Nếu kết quả âm, bit P_CY On và kết quả trả về word R, R+1 ở dạng bù 10. Để có được kết quả đúng thực hiện tiếp phép trừ 0 cho R, R+1.
( Chú ý : Sử dụng BSET(71) để tạo ra một hằng 8 digits có giá trị 0)



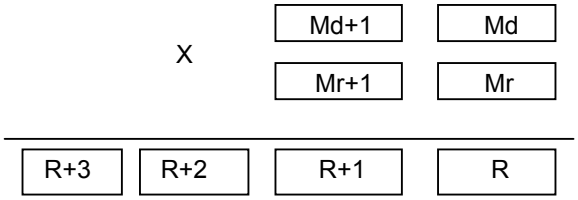

P_ER : Khi nội dung của Mi, Mi+1, Su, Su+1 không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi kết quả âm.
P_EQ : On khi kết quả là 0.

### 7. Double BCD Multiply :



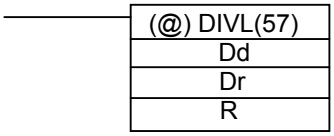

Md : Word được nhân đầu tiên. (BCD)
Mr : Word nhân đầu tiên. (BCD)
R : Word kết quả đầu tiên.

MULL(56) thực hiện nhân nội dung 8 digits của Md, Md+1 với Mr, Mr+1, kết quả trả về word R đến R+3.



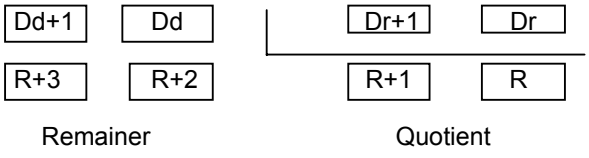

P_ER : Khi nội dung của Md, Md+1, Mr, Mr+1 không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi có nhớ trong kết quả.
P_EQ : On khi kết quả là 0.

### 8. Double BCD Divide :



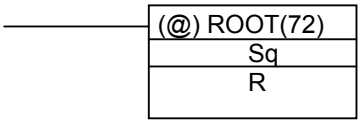

Dd : Word bị chia đầu tiên. (BCD)
Dr : Word chia đầu tiên. (BCD)
R : Word kết quả đầu tiên.

DIVL(57) thực hiện phép chia nội dung 8-digit của Dd, Dd+1 cho nội dung của Dr, Dr+1, kết quả trả về các word từ R, R+1 ; số dư trả về word R+2, R+3.




P_ER : Dr, Dr+1 có nội dung là 0.
Dd, Dd+1, Dr, Dr+1 có nội dung không phải là BCD.
Địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

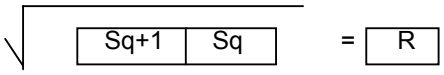
### 9. Square Root :




Sq : Word nguồn đầu tiên. (BCD)
R : Word kết quả .

Lệnh này không sử dụng cho CPM1, SRM1.

ROOT(72) thực hiện lấy căn bậc 2 nội dung 8-digit của Sq, Sq+1, kết quả trả về R. Phần thập phân của kết quả bị bỏ đi.

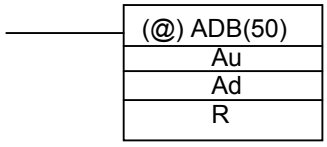



Td : $\sqrt{63250561}$ = 7953.0221...., kết quả được làm tròn 7953.

P_ER : Nội dung của Sq không phải BCD.
Word gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

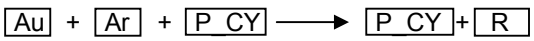
## VII- CÁC LỆNH TÍNH đếnÁN DỮ LIỆU DẠNG NHỊ PHÂN (BINARY) :

### 1. Binary Add :




Au : Word được cộng. (binary)
Ad : Word cộng. (binary)
R : Word kết quả.

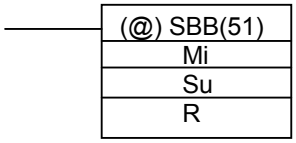
ADB(50) thực hiện phép cộng nội dung dạng nhị phân của Au, Ad, và P_CY, kết quả trả về R. P_CY sẽ ON nếu kết quả lớn hơn FFFF.




ADB(50) còn có thể được sử dụng cộng nội dung nhị phân có dấu. Với CPM1A, SRM1, bit SR 25404 và SR 25405 sẽ tác động khi kết quả vượt ngoài giới hạn trên/dưới (-32 767 đến +32 768) .

P_ER : Địa chỉ gián tiếp của DM không tồn tại.
P_CY : ON khi kết quả lớn hơn FFFF.
P_EQ : ON khi kết quả là 0.
P_OF : ON khi kết quả vượt quá giới hạn +32 767 (7FFF). (Chỉ với CJ1M, CP1L/1H, CJ1, CS1)
P_UF : ON khi kết quả vượt ngoài giới hạn dưới -32 768 (8000). (Chỉ với CJ1M, CP1L/1H, CJ1, CS1)

### Binary Subtract :




Mi : Word bị trừ. (Binary)
Su: Word trừ. (Binary)
R : Word kết quả.

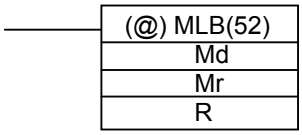
SBB(51) thực hiện trừ nội dung của Mi cho Su và P_CY, kết quả trả về word R. Nếu kết quả âm, P_CY sẽ ON và kết quả trong R có dạng bù 2.


SBB(51) cũng có thể sử dụng thực hiện phép trừ nhị phân có dấu. Với CPM1A, SRM1, bit P_OF & P_UF sẽ tác động nếu kết quả vượt ngoài giới hạn của dữ liệu nhị phân 16-bit có dấu.

P_ER : Địa chỉ gián tiếp của DM không tồn tại.
P_CY : On khi kết quả âm.
P_EQ : On khi kết quả là 0.
P_OF : On khi kết quả vượt quá 32 767 (7FFF). (Chỉ với CJ1M, CP1L/1H, CJ1, CS1)
P_UF : On khi kết quả vượt quá -32 768 (8000). (Chỉ với CJ1M, CP1L/1H, CJ1, CS1)

*Chú ý : Để chuyển từ dạng bù 2 sang dạng thông thường, sử dụng lệnh NEG(-).*

**2. Binary Multiply :**

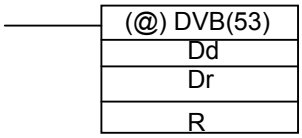



*MLB(52) không thể sử dụng nhân dữ liệu nhị phân có dấu. (Lệnh nhân nhị phân có dấu  MBS(-) - không sử dụng cho CQM1) .*

MLB(52) thực hiện nhân dữ liệu nhị phân của Md với Mr, 4-digit có trọng số thấp đặt vào R, 4-digit có trọng số cao đặt vào R+1.

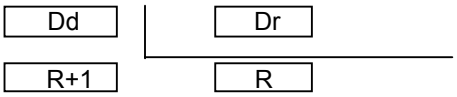

P_ER: Nếu địa chỉ gián tiếp của DM không tồn tại.
P_EQ: On khi kết quả là 0.

### 3. Binary Divide :



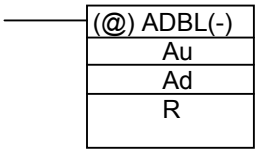

*DVB(53) thể thực hiện chia số nhị phân có dấu. ( Lệnh chia nhị phân có dấu  DBS(-)  có thể được sử dụng cho CQM1).*

DVB(53) thực hiện chia nội dung của Dd cho Dr, kết quả trả về R , số dư trả về R+1.




P_ER: Nội dung của Dr là 0.
Địa chỉ gián tiếp của DM không tồn tại.
P_EQ: On khi kết quả là 0.

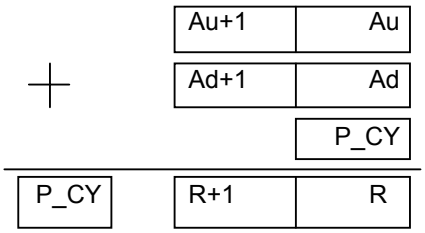
### 4. Double Binary ADD :




Au: Word được cộng thứ nhất. (Binary)
Ad: Word cộng thứ nhất. (Binary)
R : Word kết quả thứ nhất.

*Lệnh này chỉ sữ dụng cho CQM1-CPU4 - EV1.*

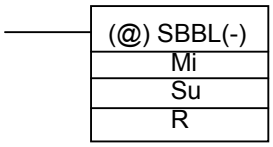
Au, Au+1 ; Ad, Ad+1 phải trên cùng vùng dữ liệu,
ADBL(-) thực hiện cộng nội dung 8-digit của Au+1, Au với nội dung 8-digit của Ad+1, Ad, và P_CY, kết quả trả về R+1, R. P_CY sẽ tác động nếu kết quả lớn hơn FFFF FFFF.




ADBL(-) có thể được sử dụng cộng dữ liệu nhị phân có dấu. Bit SR 25404 hoặc SR 25405 sẽ tác động nếu kết quả vượt ngoài giới hạn của dữ liệu 32-bit.

P_ER: Địa chỉ gián tiếp DM không tồn tại.
P_CY: On khi kết quả lớn hơn FFFF FFFF
P_EQ: On khi kết quả là 0.
P_OF: On khi kết quả lớn hơn + 2 147 483 647 (7FFF FFFF).
P_UF: On khi kết quả nhỏ hơn - 2 147 483 648 (8000 000).

### 5. Double Binary Subtract :




Mi: Word bị trừ. (Binary)
Su: Word trừ. (Binary)
R: Word kết quả.

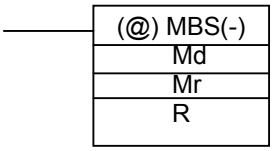
*Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.*
Mi và Mi+1, Su và Su+1, R và R+1 phải trên cùng vùng dữ liệu.

SBBL(-) thực hiện trừ nội dung của Mi+1, Mi cho Su+1, Su và P_CY, kết quả trả về R+u kết quả âm, P_CY sẽ On, nội dung trong R+1, R có dạng bù 2. Sử dụng NEG(-) chuyển từ dạng bù 2 sang kết quả thực.
SBBL(-) cũng có thể sử dụng cho phép trừ dạng nhị phân có dấu. Bit SR 25404 hoặc SR 25405 sẽ tác động nếu kết quả vượt ngoài giới hạn của dữ liệu nhị phân 32-bit.

P_ER: Địa chỉ gián tiếp của DM không tồn tại.
P_CY: On khi kết quả âm.
P_EQ: On khi kết quả là 0.
P_OF: On khi kết quả lớn hơn + 2 147 483 647 (7FFF FFFF).
P_UF: On khi kết quả nhỏ hơn - 2 147 483 648 (8000 0000).

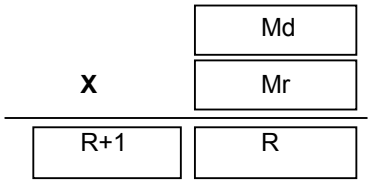
*Chú ý: 1.* Đối với dữ liệu nhị phân không dấu, P_CY dùng để chỉ ra kết quả âm. Dùng lệnh NEGL(-) để chuyển kết quả dạng bù 2 sang dạng thực.
2. Đối với dữ liệu nhị phân có dấu, bit P_OF , P_UF để chỉ kết quả vượt ngoài khoảng (- 2 147 483 647 , + 2 147 483 647).

### 6. Signed Binary Multiply :



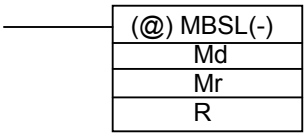

*Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.*

MBS(-) nhân nội dung nhị phân có dấu của hai word Md, Mr, kết quả là 8-digit binary có dấu trả về word R+1, R.



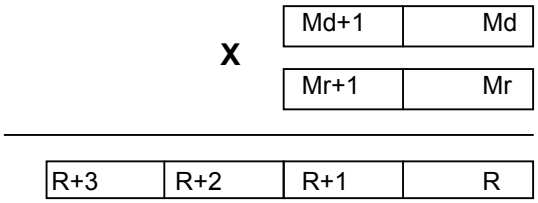

P_ER: Địa chỉ gián tiếp của DM không tồn tại.
P_EQ: On khi kết quả là 0.

### 7. Double Signed Binary Multiply :




*Lệnh này chỉ sử dụng cho PLC loại CJ1M, CP1L/1H, CJ1 & CS1.*
Md và Md+1, Mr và Mr+1 phải trên cùng vùng dữ liệu. R đến R+3 phải trên cùng vùng dữ liệu.

MBSL(-) thực hiện nhân nội dung 32-bit (8-digit) dạng nhị phân có dấu của Md+1,Md với nội dung 32-bit dạng nhị phân có dấu của Mr+1,Mr . Kết quả dạng nhị phân có dấu 16-bit trả về word R+3 đến R.




P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : ON khi kết quả là 0.

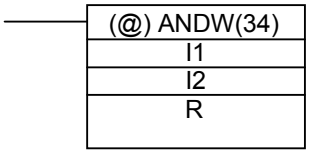
## VIII- LỆNH LOGIC :

### 1. Complement :


COM(29) xoá tất cả bit On về OFF và đặt tất cả bit OFF lên ON.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả là 0.

### 2. Logical AND :

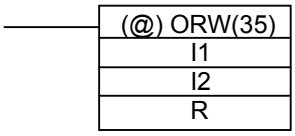



I1 : Input 1
I2 : Input 2
R : Word kết quả.

ANDW(34) thực hiện logic AND từng bit trong nội dung hai word I1 và I2 , kết quả trả về R.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

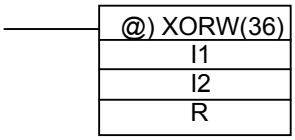
### 3. Logical OR :




I1 : Input 1
I2 : Input 2
R : Word kết quả.
ORW(35) thực hiện logic OR từng bit giữa nội dung của hai word I1 và I2, kết quả trả về R.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

### 4. Exclusive OR :

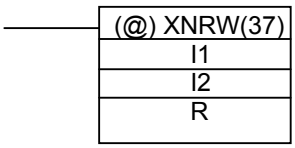



I1 : Input 1
I2 : Input 2
R : Word kết quả.

ORW(35) thực hiện logic Exclusive OR từng bit giữa nội dung của hai word I1 và I2, kết quả trả về R.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

### 5. Exclusive NOR :

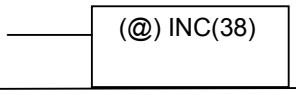



I1 : Input 1
I2 : Input 2
R : word kết quả.
ORW(35) thực hiện logic Exclusive NOR từng bit giữa nội dung của hai word I1 và I2, kết quả trả về R.

P_ER : Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

## IX- LỆNH TĂNG / GIẢM :

### 1. BCD Increment :

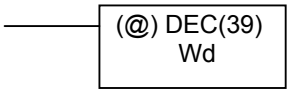


Wd

INC(38) thực hiện tăng nội dung BCD của Wd, kết quả không bị ảnh hưởng bởi P_CY.

P_ER : Nội dung trong WD không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

### 2. BCD Decrement :




INC(38) thực hiện giảm nội dung BCD của Wd, kết quả không bị ảnh hưởng bởi P_CY.

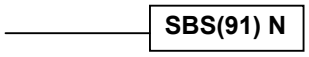
P_ER : Nội dung trong WD không phải dạng BCD.
Khi địa chỉ gián tiếp của DM không tồn tại.
P_EQ : On khi kết quả bằng 0.

## X - LỆNH CHƯƠNG TRÌNH CON (Subroutine Instructions):

Chương trình con thực hiện rẽ nhánh chương trình chính trong trường hợp cần thực hiện một hay một nhóm điều khiển vào bất kỳ thời điểm nào trong chu kỳ quét của chương trình chính. Chương trình con có thể được thực hiện một hay nhiều lần trong một chu quét của chương trình chính. Việc viết các lệnh trong chương trình con giống như đối với chương trình chính.

Khi tất cả các lệnh của chương trình con thực hiện xong, chương trình sẽ quay về vị trí ngay sau vị trí gọi chương trình con, chương trình sẽ tiếp tục thực hiện những bước kết tiếp.

### 1. Subroutine enter - SBS(91):

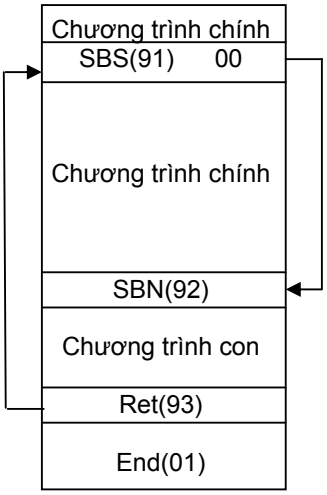



**N : số thứ tự của chương trình con. ( 000 ~ 255).**

*CJ1M : N= 000 ~ 127.*
*CPM1 : N= 000 ~ 049.*

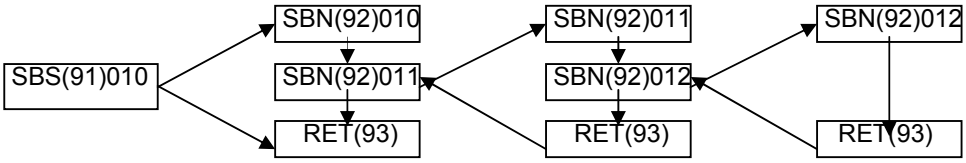
Việc rẽ nhánh chương trình có thể được thực hiện bằng cách đặt SBS(91) vào chương trình chính tại nơi cần rẽ nhánh. Số thứ tự N chỉ ra chương trình con tương ứng sẽ được gọi.

Khi điều kiện cho SBS(91) On, những lệnh giữa SBN(92) có cùng số thứ tự N và RET(93) đầu tiên sẽ được thực hiện trước, sau đó chương trình sẽ quay về thực hiện tiếp các lệnh ngay sau ngay sau SBS(91) vừa gọi.




SBS(91) có thể được dùng nhiều lần trong chương trình, có nghĩa là cùng một chương trình con có thể được gọi nhiều lần tại nhiều nơi trong chương trình chính.

SBS(91) có thể đặt trong một chương trình con để gọi tiếp một chương trình con khác, có nghĩa là chương trình con có thể đặt lồng vào nhau. Việc lồng chương trình con cho phép đến 16 mức. Khi một chương trình con kết thúc, nó sẽ quay về chương trình con có mức cao hơn đã gọi nó.




Bit P_ER ON khi :

Số thứ tự thứ tự của chương trình con không tồn tại.
Chương trình con tự gọi nó.
Gọi một chương trình con đang thực hiện.

Chú ý: SBS(91) sẽ không thực hiện và chương trình con sẽ không được gọi khi P_ER ON.

**2. Subroutine define and Return - SBN(92) / RET(93):**

SBN(92) N — N : Số thứ tự chương trình con. (000 ~ 255)

RET(93)

Điều kiện :

*CJ1M:* N=000 ~ 127
CPM1: N=000 ~ 049
CP1L/1H: N=0 ~ 255

N chỉ được sử dụng trong SBN(92) một lần.

SBN(92) là điểm bắt đầu một chương trình con, RET(93) là điểm kết thúc. Mỗi chương trình con được nhận dạng bởi N.

Tất cả chương trình con phải được lập trình ở cuối chương trình chính.Chương trình chính sẽ thực hiện một hay nhiều chương trình con ( nếu nó được gọi ) trước khi trở về địa chí 0000 để thực hiện chu kỳ quét kế tiếp.

END(01) phải được đặt ngay sau RET(93) cuối cùng.

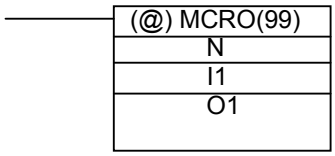
*Chú ý:*

Trong một chu kỳ quét, chương trình sẽ quay trở về địa chỉ đầu tiên để thực hiện chu kỳ quét kế tiếp nếu gặp SBN(92).

Nếu DIFU(13) hay DIFD(14) được đặt trong chương trình con, bit tác động bởi hai lệnh trên chỉ OFF khi chương trình con được gọi lại lần thứ hai, có nghĩa là thời gian ON kéo dài hơn một chu kỳ.

## XI - LỆNH ĐẶC BIỆT (Special Instructions):

### 1. Macro - MCRO(99)



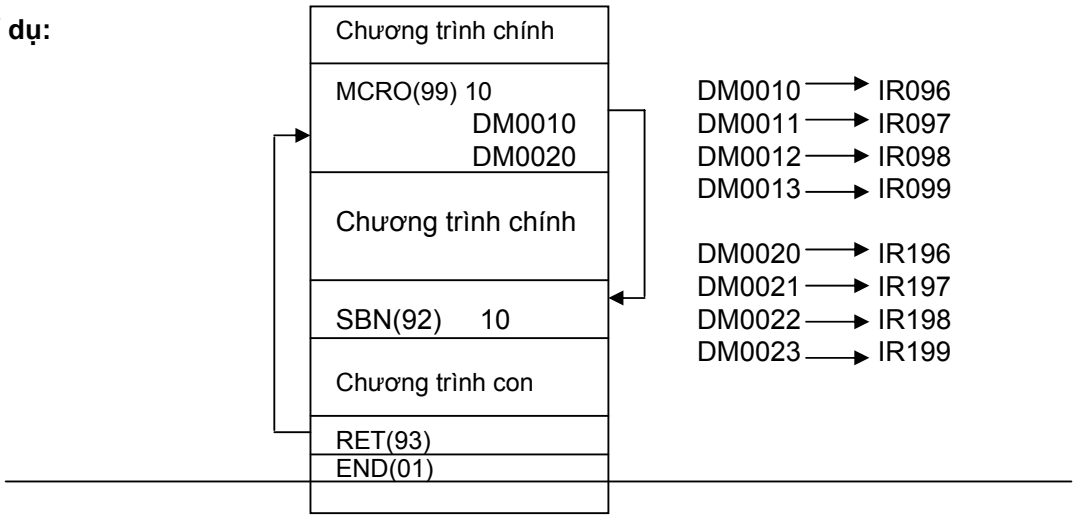

N: Số thứ tự chương trình con (000 ~ 127)
I1: Word input đầu tiên.
O1: Word output đầu tiên.

*Điều kiện:*

MCRO(99) cho phép một chương trình con có thể thực hiện nhiều chức năng khác nhau. Có nghĩa là một chương trình con có thể thay thế cho nhiều chương trình con khác có cấu trúc giống nhau nhưng kết quả hoạt động khác nhau. Có 04 word input, IR096 ~ IR099 ( IR232 ~ IR235 đối với CPM1), và 04 word output, IR196 ~ IR199 ( IR236 ~ IR239 đối với CPM1) được dùng cho MCRO(99). 08 word này được dùng trong chương trình con và dữ liệu của nó là được lấy từ các word I1 ~ I1+3 và O1 ~ O1+3 khi chương trình con làm việc.

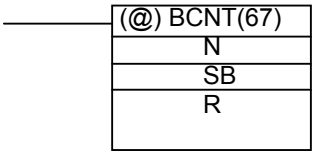
MCRO(99) copy nội dung của I1 ~ I1+3 vào IR096 ~ IR099, nội dung của O1 ~ O1+3 vào IR196 ~ IR199, sau đó gọi và thực hiện chương trình con N. Khi thực hiện xong chương trình con, nội dung của IR196 ~ IR199 được truyền trở lại O1 ~ O1+3.

**Ví dụ:**



P_ER On khi: Chương trình con hay N không tồn tại .
Vùng dữ liệu nằm ngoài vùng cho phép.
Địa chỉ tương đối của DM không tồn tại.
Chương trình con tự gọi nó.
Gọi một chương trình con đang làm việc.

### 2. Bit Counter - BCNT(67)



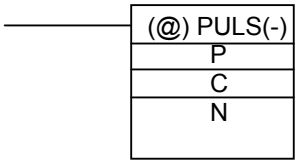

N: Số lượng word (BCD).
SB: Word nguồn đầu tiên.
R: Wor kết quả đầu tiên.

Điều kiện: N phải khác 0.

BCNT(67) đếm tất cả số lượng bit ở trạng thái ON trong tất cả các word từ SB đến SB+(N-1), kết quả được trả về R.
P_ER : N không phải là BCD hay N=0: SB ~ SB+(N-1) không ở trên cùng vùng dữ liệu.
Kết quả trong R lớn hơn 9999.
Địa chỉ gián tiếp DM không tồn tại.
P_EQ: Khi kết quả là 0. ( R : 0000)

## XII- CHỨC NĂNG NGÕ RA PHÁT XUNG

### 1- SET PULSES - PULS(-)




P: Port phát xung ( 000, 001, 002)
C: Dữ liệu điều khiển. (000 đến 005)
N: Số lượng xung (IR, SR, AR, DM, HR, LR).

***Điều kiện: N và N+1 phải trên cùng vùng dữ liệu.***

PULS(-) dùng để đặt thông số cho ngõ ra phát xung mà nó được thực hiện bởi SPED(-) hoặc ACC(-).

- **Chọn ngõ ra phát xung:**

P = 000 : Ngõ ra xung là bit output.
P = 001 : Ngõ ra xung là Port 1
P= 002 : Ngõ ra xung là Port 2.

- **Dữ liệu điều khiển C:**

| C | Chiều phát xung | Số xung được phát | Thời điểm bắt đầu cạnh xuống |
|---|---|---|---|
| 000 | Chiều thuận | Đặt trong N và N+1 | Không dùng |
| 001 | Chiều nghịch | Đặt trong N và N+1 | Không dùng |
| 002 | Chiều thuận | Đặt trong N và N+1 | Đặt trong N+2 và N+3 |
| 003 | Chiều nghịch | Đặt trong N và N+1 | Đặt trong N+2 và N+3 |
| 004 | Chiều thuận | Không dùng | Không dùng |
| 005 | Chiều nghịch | Không dùng | Không dùng |

Việc đặt chiều phát xung có tác dụng cho đến khi dừng chương trình hoặc PULS(-) được thực hiện trở lại.

- **Số xung và thời điểm bắt đầu cạnh xuống :**

Khi C = 000 hay 003, N+1 và N chứa giá trị số xung phát ra không phụ thuộc vào chế độ phát xung. N+1, N chứa giá trị từ 00000000 ~ 16777215. Xung sẽ được phát ra khi điều kiện lệnh SPED(-) hay ACC(-) cho phép và sẽ tự động dừng khi số lượng xung phát ra đạt đến số lượng đã đặt trước.

| | Leftmost 4 digits | Rightmost 4 digits | Possible range |
|---|---|---|---|
| Số lượng xung được phát: | N | N | 00000001 ~ 16777215 |

Khi C = 002 hay 003, N+3 và N+2 chứa giá trị đặt số lượng xung để thực hiện cạnh xuống trong lệnh ACC(-) ở Mode 0. N+3, N+2 có thể chứa giá trị từ 00000001 ~ 16777215. Xung ra bắt đầu bởi ACC(-) sẽ bắt đầu thực hiện cạnh xuống khi số xung đạt đến giá trị đặt ban đầu.

| | Leftmost 4 digits | Rightmost 4 digits | Possible range |
|---|---|---|---|
| Thời điểm thực hiện cạnh xuống | N+3 | N+4 | 00000001 ~ 16777215 |