Data Manipulation and Math Instructions

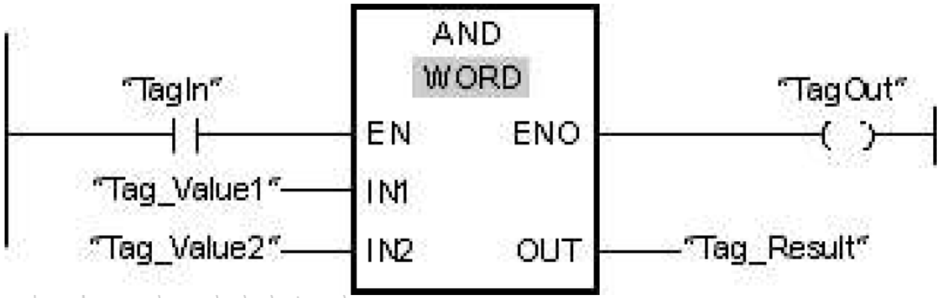
Part 2

Logic Word Instruction

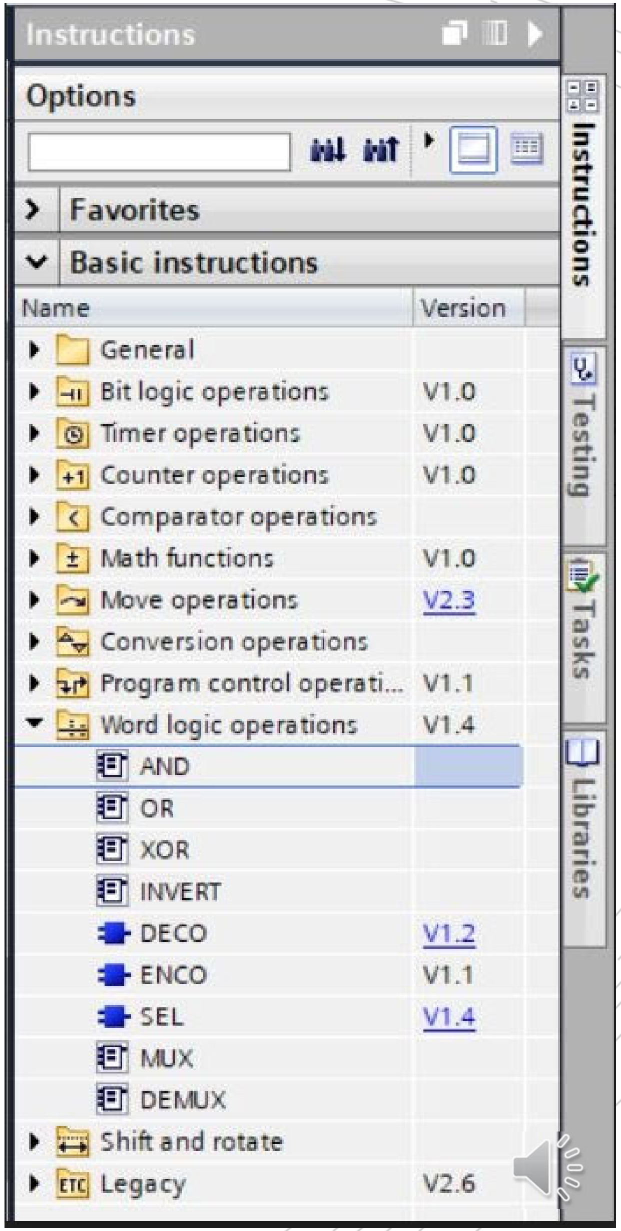
AND Operation

There would be situation that using logic word operations could be effective or suitable

- Logic AND instruction combine IN1 and IN2 values bit-by-bit by **AND logic**
- Result bit = 1 when both the bits =1
- Result is stored in **OUT**



IN1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
IN2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
OUT	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

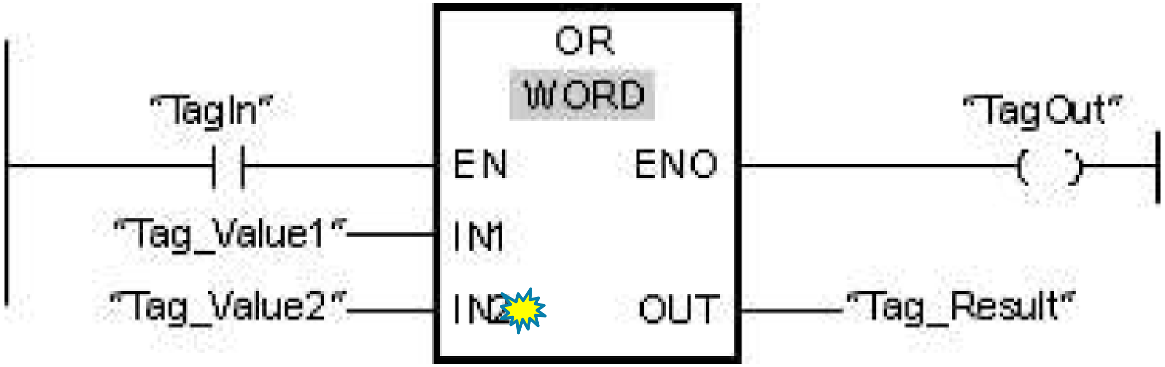


Name	Version
General	
Bit logic operations	V1.0
Timer operations	V1.0
Counter operations	V1.0
Comparator operations	
Math functions	V1.0
Move operations	V2.3
Conversion operations	
Program control operati...	V1.1
Word logic operations	V1.4
AND	
OR	
XOR	
INVERT	
DECO	V1.2
ENCO	V1.1
SEL	V1.4
MUX	
DEMUX	
Shift and rotate	
Legacy	V2.6

Logic Word Instruction

OR Operation

- Logic OR instruction combine IN1 and IN2 values bit-by-bit by **OR logic**
- Result bit = 1 when at least 1 of the 2 bits in OR operation =1
- Result is stored in **OUT**
- Number of inputs could be expanded

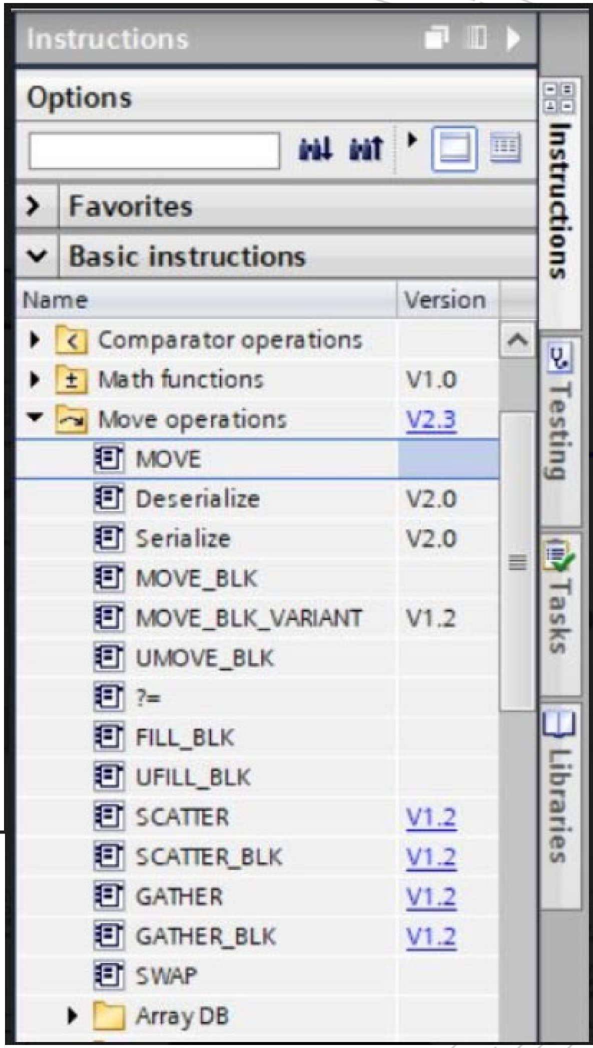


IN1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
IN2	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
OUT	0	1	0	1	0	1	0	1	0	1	0	1	1	1	1	

Move operation is very useful to transfer values or even data in array. For Array data movement, **MOVE_BLK** would be used (same data type)

Move Operation

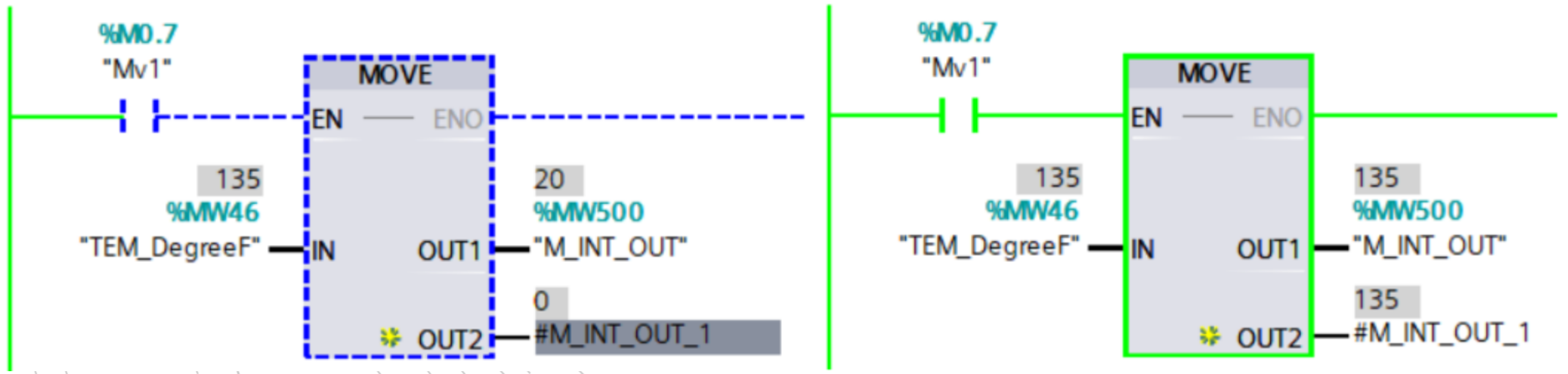
- When Move instruction is enabled (EN=1)
- Content from IN transferred to OUT
- Be mindful about the data type that it should be the same
- The number of OUT could be expanded



Instructions

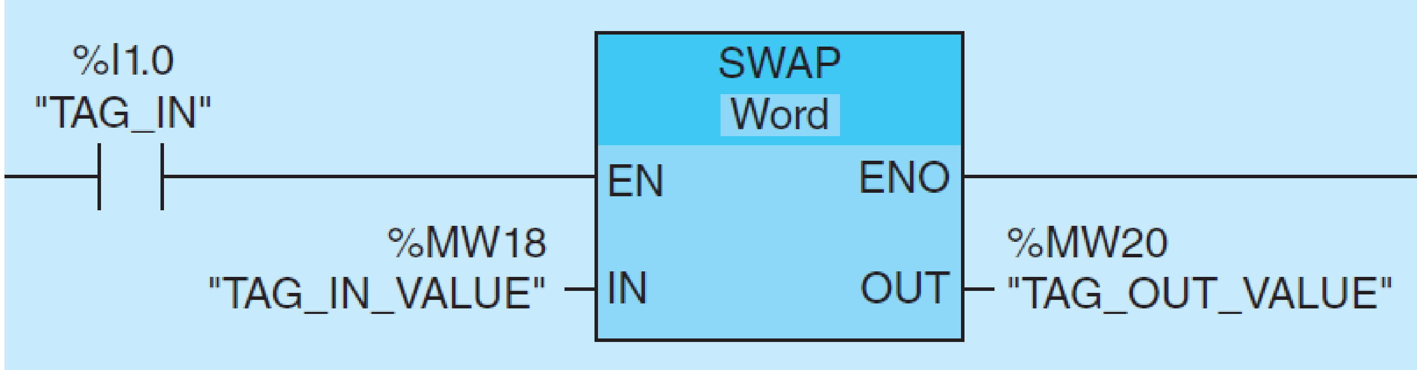
Options

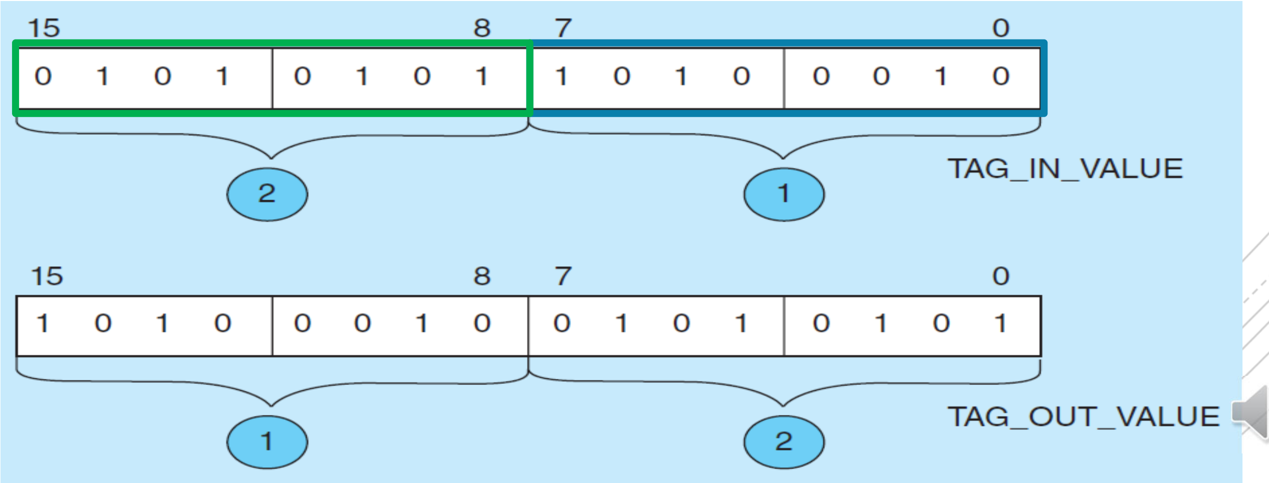
Name	Version
Comparator operations	
Math functions	V1.0
Move operations	V2.3
MOVE	
Deserialize	V2.0
Serialize	V2.0
MOVE_BLK	
MOVE_BLK_VARIANT	V1.2
UMOVE_BLK	
?=	
FILL_BLK	
UFILL_BLK	
SCATTER	V1.2
SCATTER_BLK	V1.2
GATHER	V1.2
GATHER_BLK	V1.2
SWAP	
Array DB	



SWAP

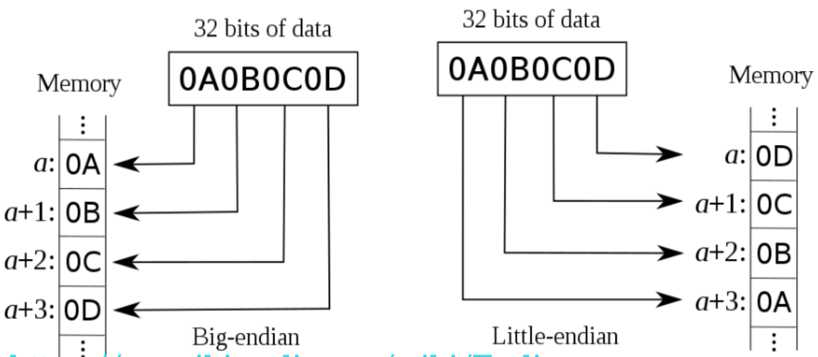
- When TAG_IN = 1
- TAG_IN_VALUE value at **IN** is swapped and the result stored in **OUT**





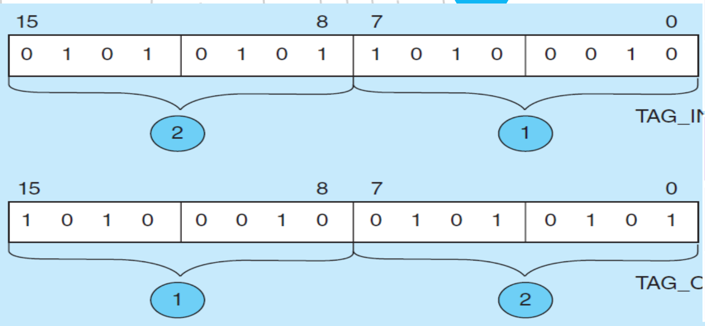
From Lecture 4

M S B							Bit 8	Bit 7							L S B
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
Upper Byte								Lower Byte							
16 – Bit Word															



https://en.wikipedia.org/wiki/Endianness

SWAP



- PLC is often integrated with other controllers or devices that transmit word or double word data
- A 16-bit integer occupy 2 consecutive bytes of memory
- There is no standard order for computer or instrument to store these 2 bytes in memory or transmit them
- Example number 16 in decimal is 0010Hex to store in memory register 2000 of a controller
- We need to know if our data is stored as Little Endian or Big Endian and the device we are communication to is in Little / Big Endian before we transmit or receive

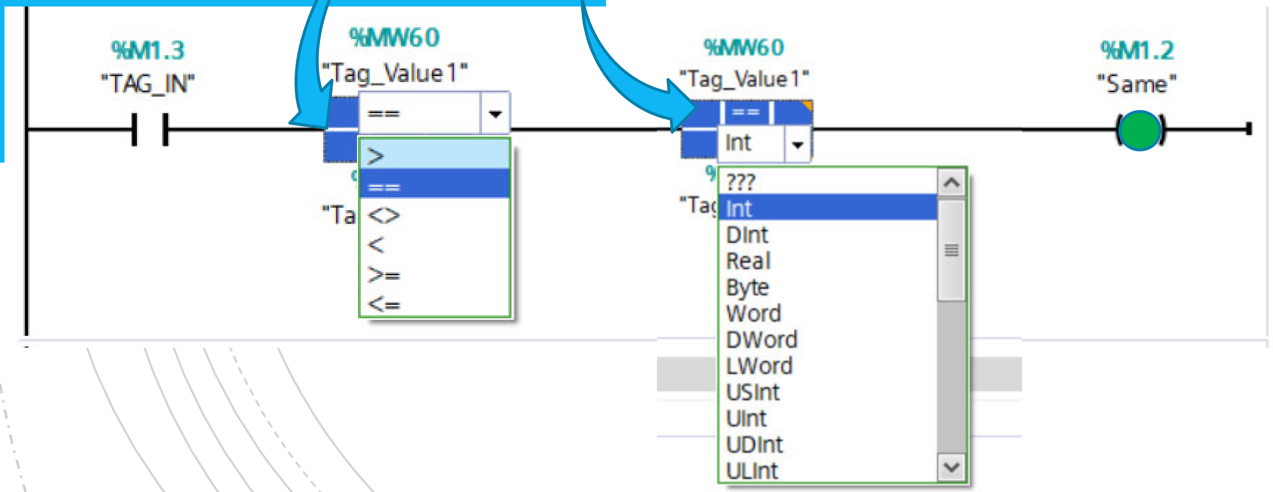
Little Endian															
00Hex (Stored in register 2001)								10Hex (Stored in register 2000)							
MSB 0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	LSB 0
Big Endian (Siemens Controller)															
10Hex (Stored in register 2000)								00Hex (Stored in register 2001)							
0	0	0	1	0	0	0	LSB 0	MSB 0	0	0	0	0	0	0	0

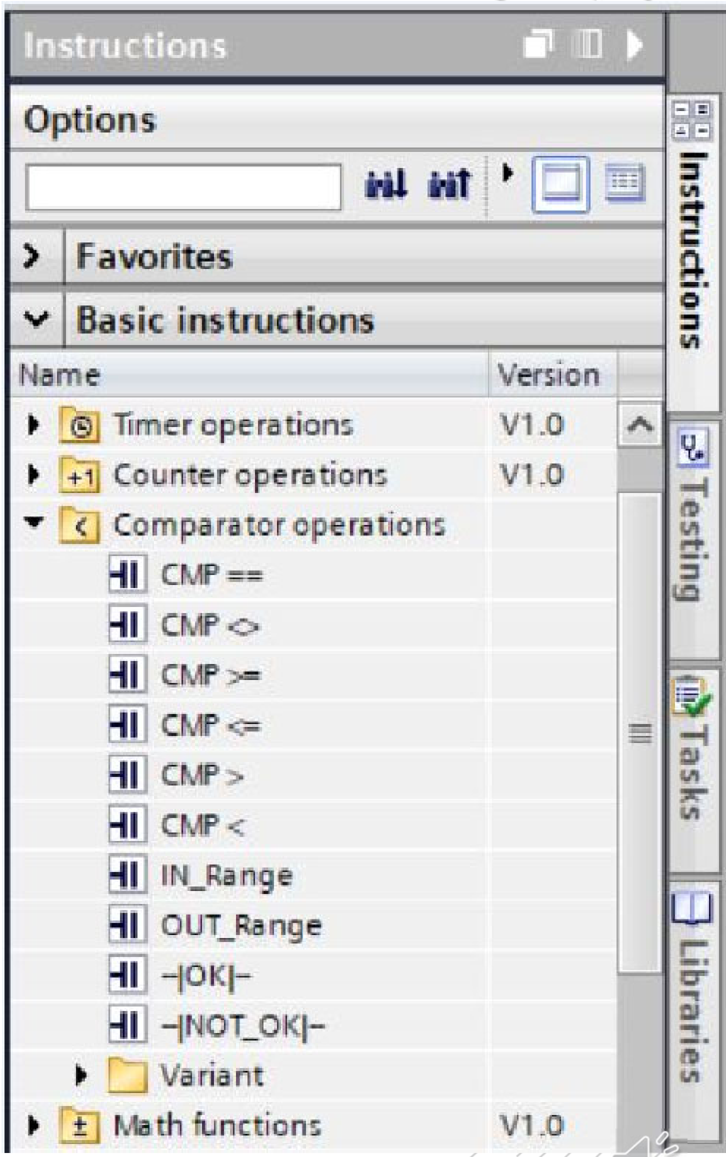
COMPARE

- There are common compare functions available to compare same data type, with Boolean output to reflect status

Selection of Compare function

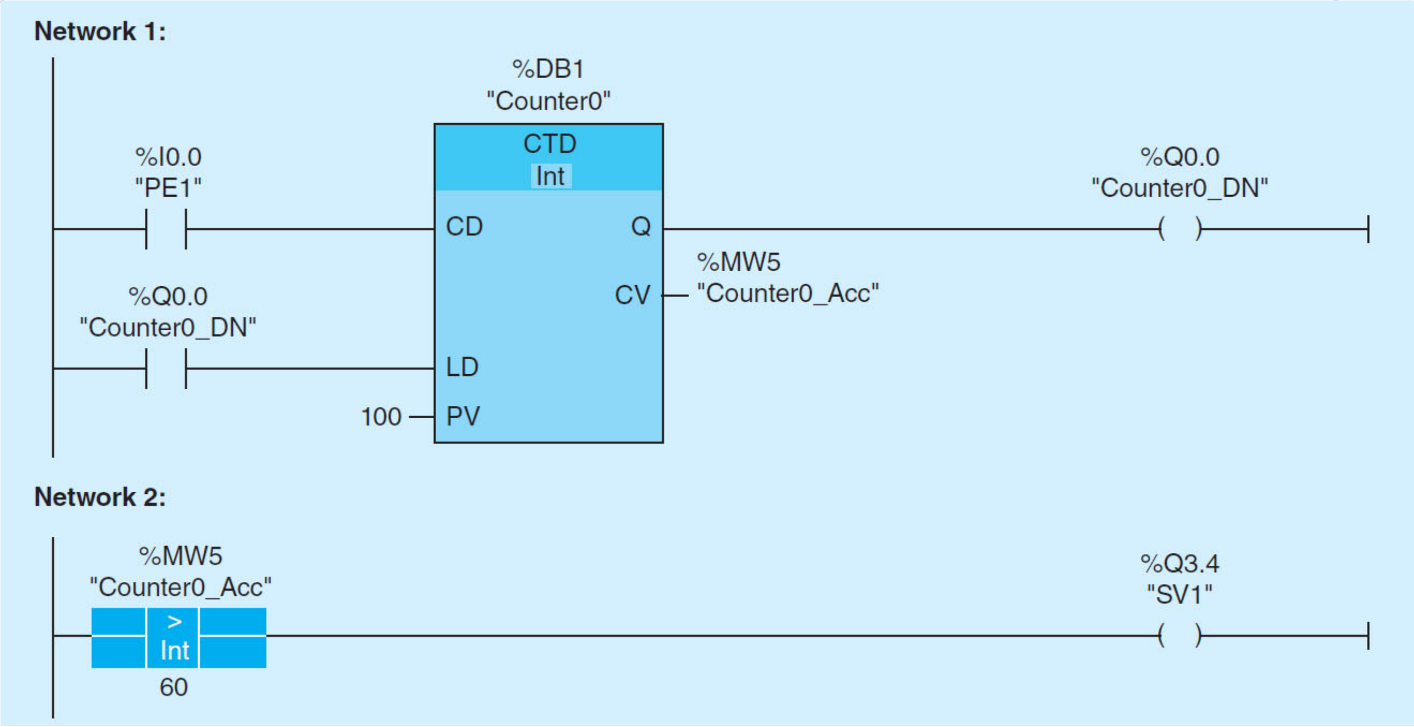
Selection of Data Type





Name	Version
Timer operations	V1.0
Counter operations	V1.0
Comparator operations	
CMP ==	
CMP <>	
CMP >=	
CMP <=	
CMP >	
CMP <	
IN_Range	
OUT_Range	
-|OK|-	
-|NOT_OK|-	
Variant	
Math functions	V1.0

Example



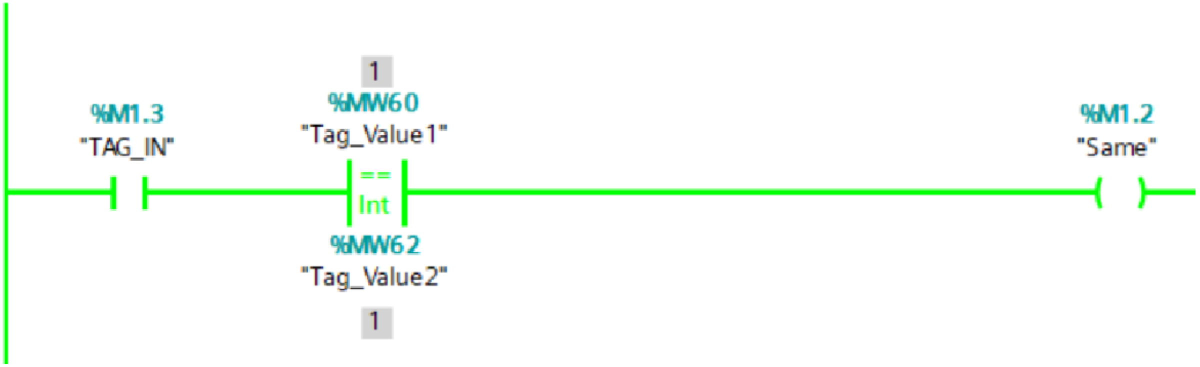
Simple control could be achieved with compare function

- We would start/restart process by activating LD “Counter0_DN = 1
- Count down counter would start with CV = PV = 100 and decrease whenever PE1 sense a new item transit from 0 to 1
- Which is where CV > 60, Solenoid Valve SV1 would activate (=1) for the first 40 items sensed by Photoelectric sensor “PE1”

COMPARE
Equal ==
Not Equal <>

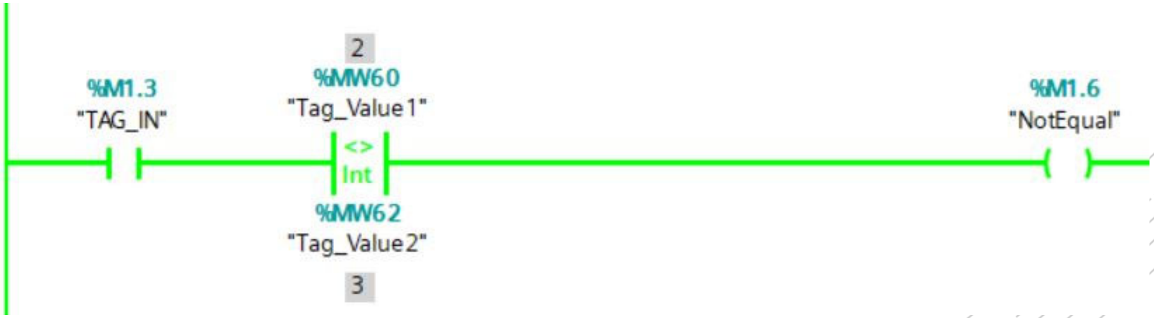
Equal

- When value of Tag_Value1 = Tag_Value2, Coil "Same" would set, else = 0



NOT Equal

- When value of Tag_Value1 is different from Tag_Value2, Coil "NotEqual" would set, else = 0

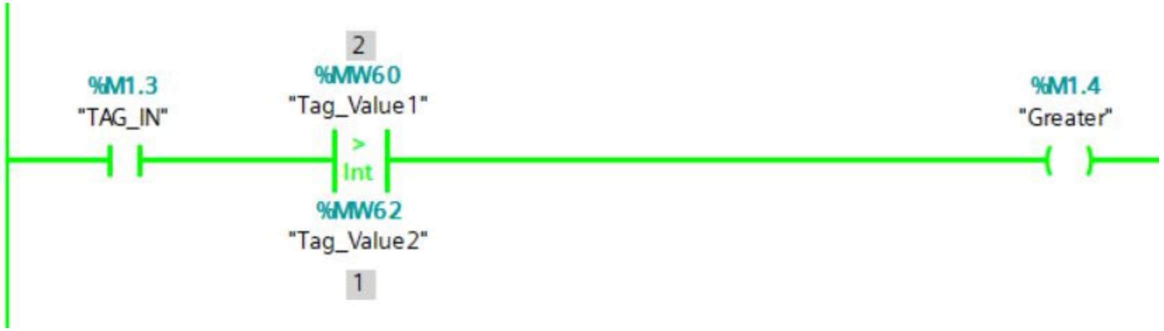


COMPARE
Greater Than >
Less Than <

Comparison is done with the top parameter to compare with the bottom parameter

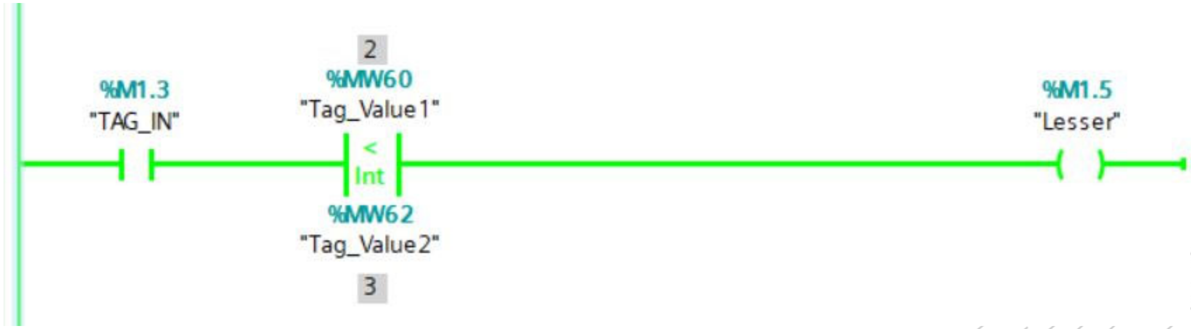
Greater Than

- When value of Tag_Value1 > Tag_Value2, Coil "Greater" would set, else = 0



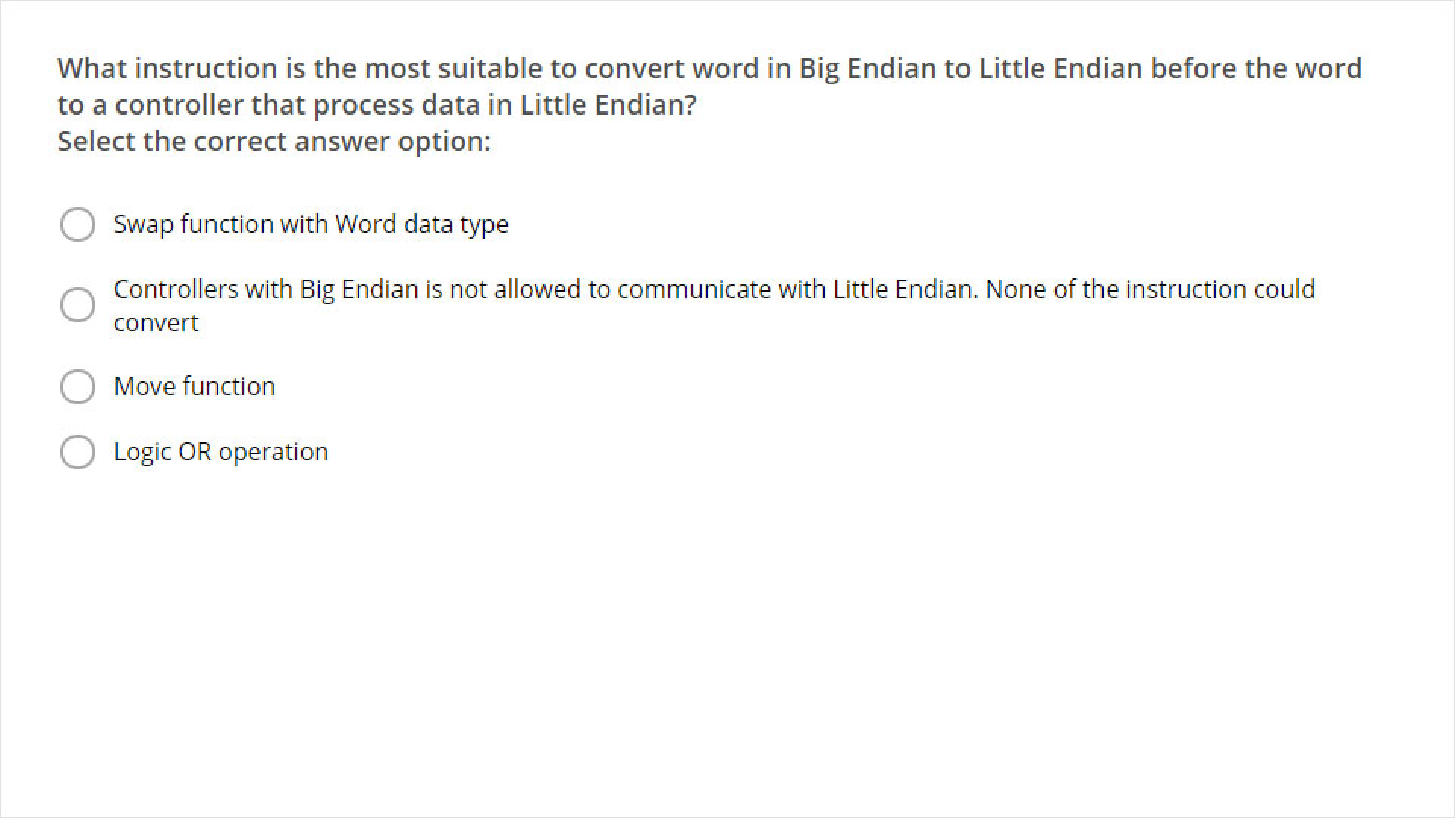
Less Than

- When value of Tag_Value1 < Tag_Value2, Coil "Lesser" would set, else = 0



Quiz

Click the **Quiz** button to edit this object



What instruction is the most suitable to convert word in Big Endian to Little Endian before the word to a controller that process data in Little Endian?
Select the correct answer option:

○ Swap function with Word data type

○ Controllers with Big Endian is not allowed to communicate with Little Endian. None of the instruction could convert

○ Move function

○ Logic OR operation