



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

第5章 FX_{2N}功能指令



5.1 功能指令的表示与执行方式

5.2 程序流向控制指令

5.3 数据传送和比较指令

5.4 算术运算和逻辑运算指令

5.5 循环与移位指令





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

5.1 功能指令表示与执行方式

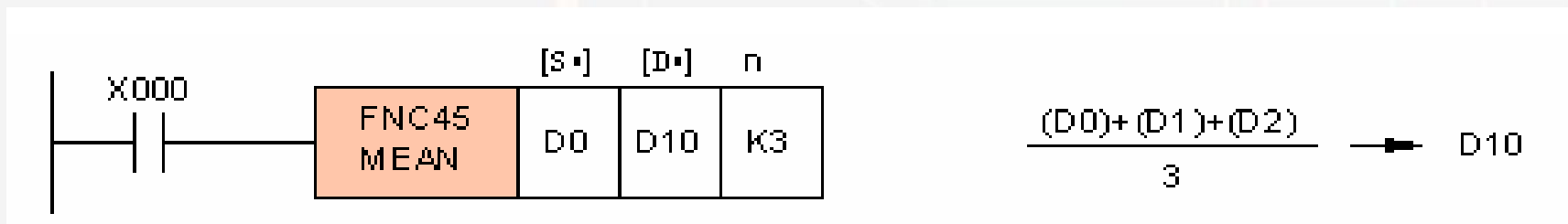


武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

指令与操作数

1. 功能指令

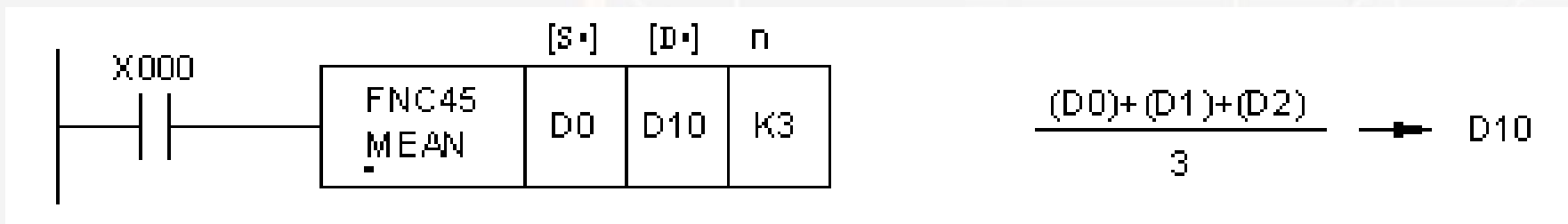
功能框的第一部分是指令 **FNC编号**。



- 有些指令仅使用指令段（FNC编号），但多数是将其与操作数结合在一起使用。

2. 操作数

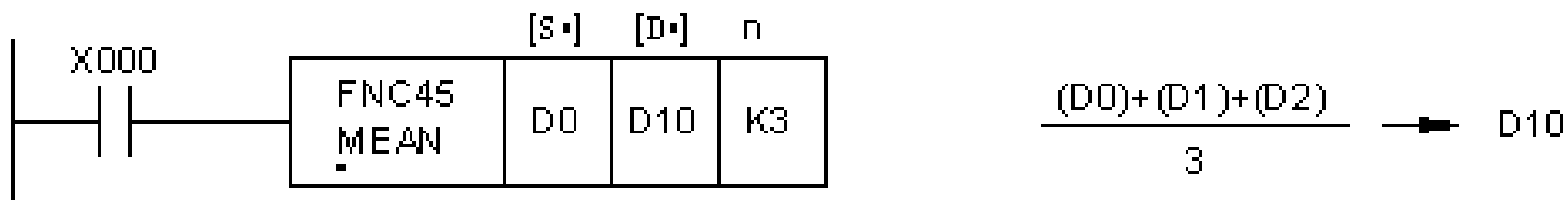
- 功能框的第一段之后都为操作数部分。
- 操作数依次由“**源操作数**”（源）、“**目标操作数**”（目）、和“**数据个数**”3部分组成。



➤ ① [S·]：源操作数，指令执行后其内容不改变。源的数量多时，以[S1·]、[S2·]等表示。以加上“·”符号表示使用变址方式，默认为无“·”，表示不能使用变址方式。

➤ ② [D·]：目标操作数，指令执行后将改变其内容。在目标数种多时，以[D1·]、[D2·]等表示。默认为无“·”，表示不能使用变址方式。

➤ ③ M、n：其他操作数，用来表示阐述或对源和目标操作数作出补充说明。表示常数时，K后跟的为十进制数，H后跟的为十六进制数。



- **程序步：指令执行所需的步数。**
- **功能指令段的程序步数通常为1步，**
- **根据各操作数是16位指令还是32位指令，会变为2步或4步。**
- **当功能指令处理32位操作数时，则在指令助记符号前加(D)表示。(D)MEAN**
- **指令前无此符号时，表示处理16位数据。**



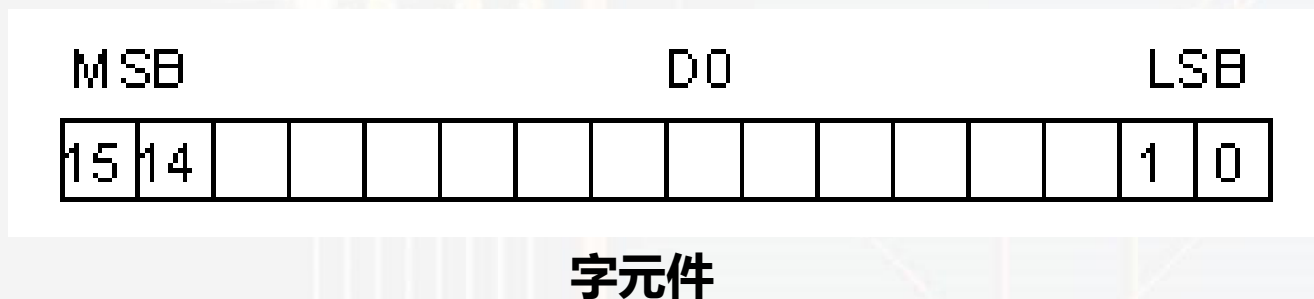
武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

指令的数据长度与执行形式

1. 字元件/双字元件

(1) 字元件

- 一个字元件是由16位的存储单元构成，其最高位（第15位）为**符号位**，第0~14位为数值位。



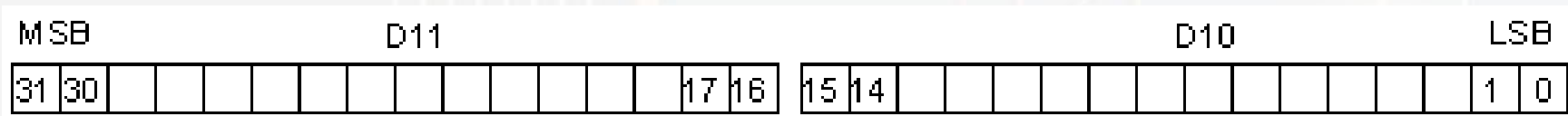


(2) 双字元件

➤ 低位元件D10存储32位数据的低16位，高位元件D11存储32位数据的高16位。

➤ 32位元件对数据的存放原则是：“低对低，高对高”

➤ 双字节元件中第31位为符号位，第0 ~ 30位为数值位

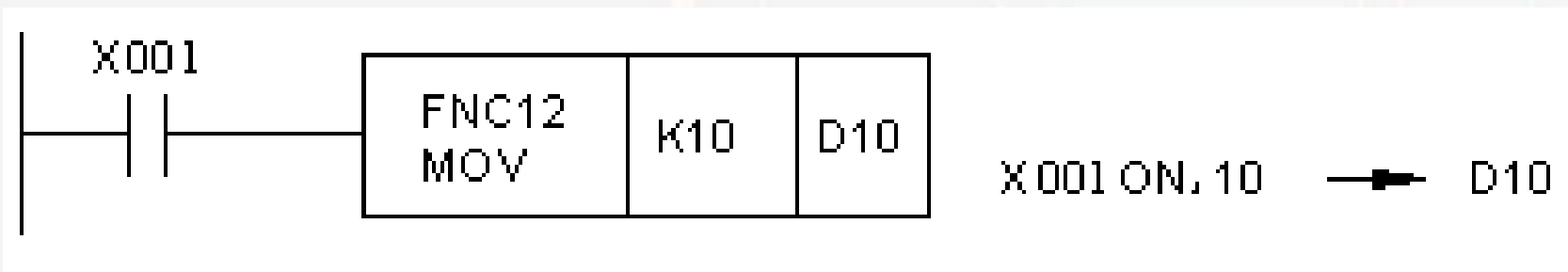


双字元件



2. 16位/32位指令执行形式

■ 16位MOV指令

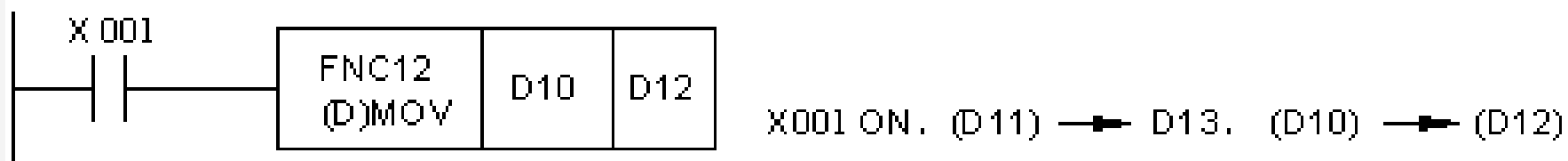


16位MOV指令

- ✓ 当X001接通时，将十进制数10传送到16位的数据寄存器D10中去。
- ✓ 当X001断开时，该指令被跳过不执行，源和目的内容都不变

■ 32位MOV指令

功能指令处理32位数据时，需在指令前面加上**前缀符号(D)**。
这时相邻的两个数据寄存器组成数据寄存器对。



32位MOV指令

- ✓ 当X001接通时，将由D11和D10组成的32位源数据传送到由D13和D12组成的目标地址中去。
- ✓ 当X001断开时，该指令不执行，源和目的内容都不变。



3. 位元件/位元件组件

位元件：只处理ON/OFF信息的软元件。如X, Y, M, S等

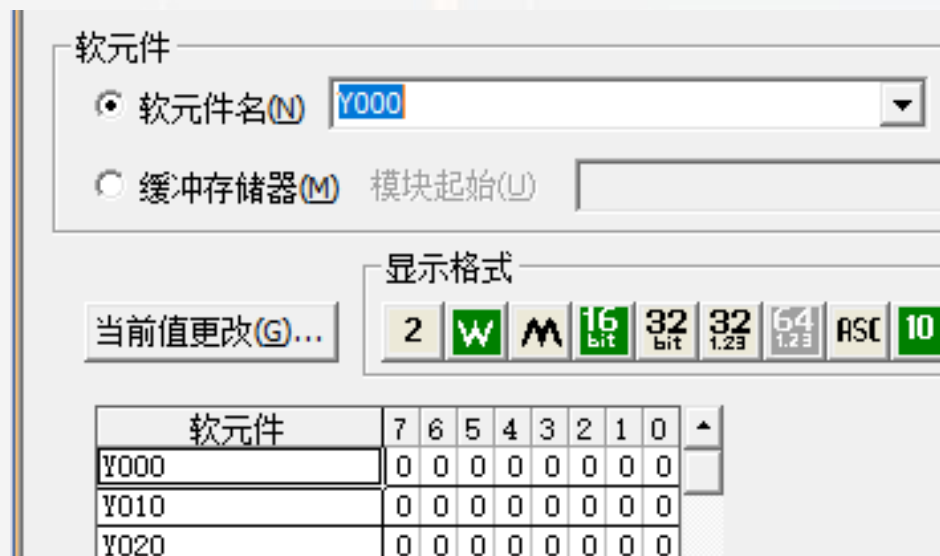
位元件组件：将多个位元件按4位一组的原则组合成4位BCD码表示一位十进制数。

助记符： K_n +最低位的位元件。如 K_nX 、 K_nY 、 K_nM

K表示十进数，n表示4位一组的组数。

例： K_2M_0 表示2组4位的位元件组成组件，最低位是 M_0 和 M_4 ； M_0 - M_3 和 M_4 - M_7 两组元件组成一个8位数。





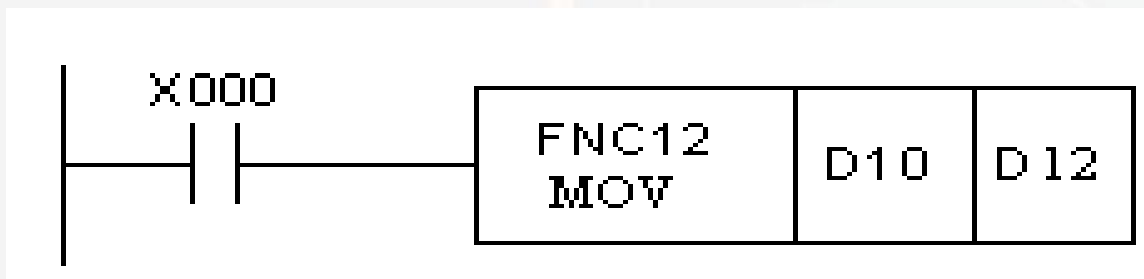






4. 连续执行型/脉冲执行型指令

(1) 连续执行指令型

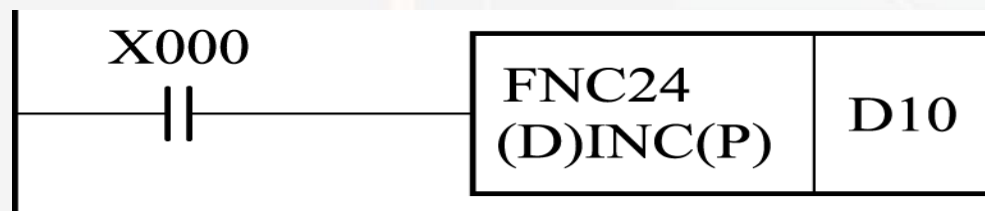


连续执行指令举例

✓ X000= ON时，指令在各扫描周期都执行。

(2) 脉冲执行型指令

- ✓ 指令只在X000由OFF→ON变化一次时执行一次，其它时候不执行。



- ✓ 如上图所示是一条INC指令，对目标组件(D10、D11)进行加1操作的。假设该指令以连续方式工作，那么只要X000=ON，则PLC在程序执行时的每个扫描周期都会对目标组件加1，而这种情况在许多实际的控制中是不希望的。为解决这类问题，设置了脉冲执行方式，并在这类助记符的后面加后缀符号 **"P"** 来表示此方式。



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

变址操作



寄存器变址操作的一般规则

- ✓ 变址方法：将变址寄存器V和Z这两个16位的寄存器放在各种寄存器的后面，充当操作数地址的偏移量。
- ✓ 操作数的实际地址：寄存器的当前值以及V和Z内容相加后的和。



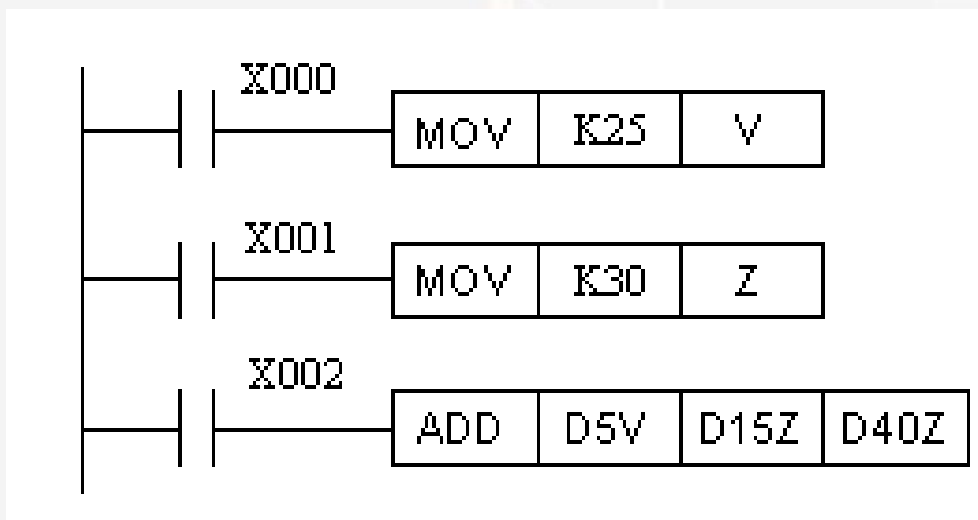
- ✓ 当源或目标寄存器用[S·]或[D·]表示时，就能进行变址操作。

对32位数据进行操作时，要将V、Z组合成32位（V，Z）来使用，这时Z为低16位，V为高16位。

- ✓ 可以用变址寄存器进行变址的软元件有X、Y、M、S、P、T、C、D、K、H、KnX、KnY、KnM、KnS。



【例】 求执行加法操作后源和目操作数的实际地址



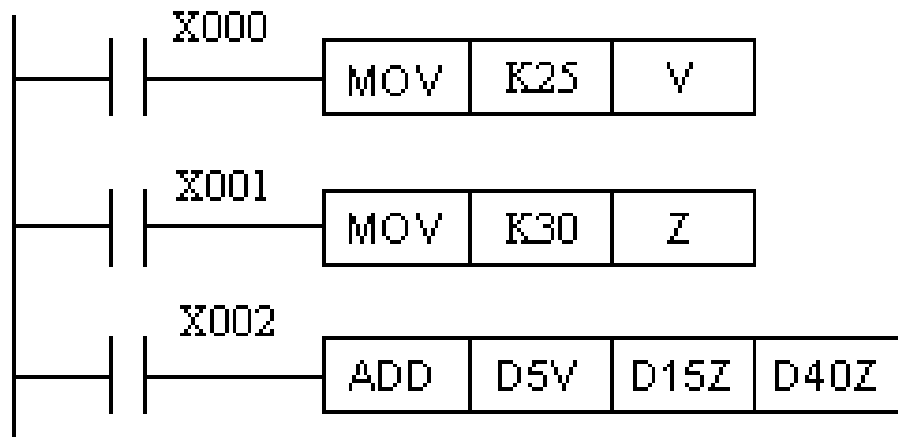
解： 第一行指令执行 $25 \rightarrow V$,

第二行指令执行 $30 \rightarrow Z$,

所以变址寄存器的值为： $V = 25, Z = 30$ 。

第三行指令执行 $(D5V) + (D15Z) \rightarrow (D40Z)$ 。

【例】 求执行加法操作后源和目操作数的实际地址



[S1·]为D5V: $D(5 + 25) = D30$ 源操作数1的实际地址

[S2·]为D15Z: $D(15 + 30) = D45$ 源操作数2的实际地址

[D·]为D40Z: $D(40 + 30) = D70$ 目操作数的实际地址

程序实现功能: $(D30) + (D45) \rightarrow (D70)$, 即D30的内容和D45的内容相加, 结果送入D70中去。



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

5.2 程序流向控制指令



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

条件跳转指令



1. 指令格式

指令编号及助记符：条件跳转指令FNC00 CJ或CJ (P)

➤ CJ指令的目标元件是指针标号，其范围是P0~P63（允许变址修改）。

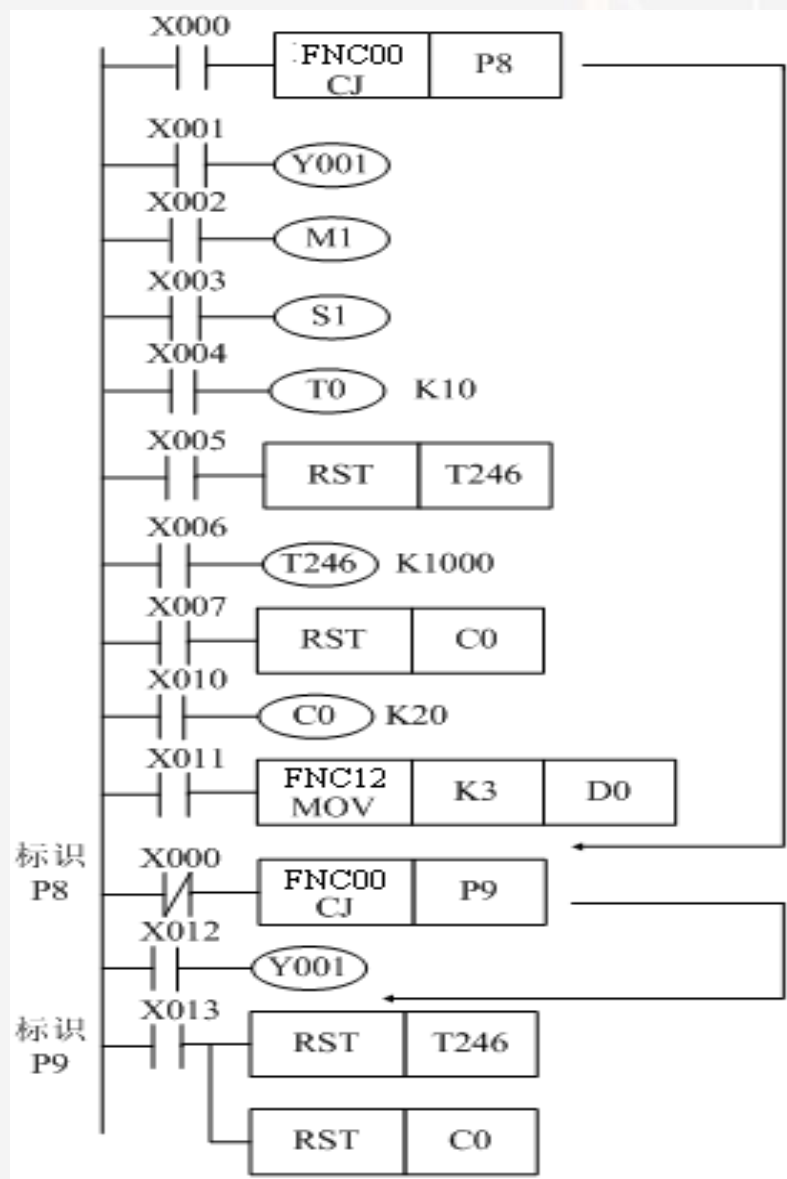
➤ 作为执行序列的一部分指令，有CJ，CJP指令，可以缩短运算周期及使用双线圈。

2. 指令用法

✓ 条件跳转指令用于当跳转条件成立时跳过CJ或CJ (P) 指令和指针标号之间的程序，从指针标号处连续执行，若条件不成立则继续顺序执行，以减少程序执行扫描时间



【例】条件跳转指令CJ的用法



解：

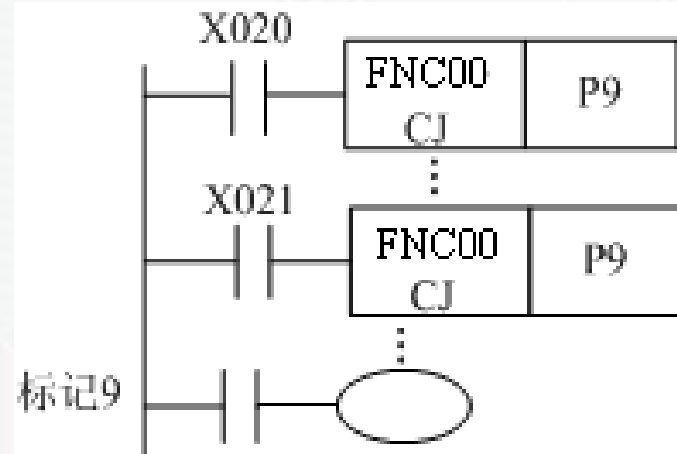
- X000“ON”，则从1步跳转到36步（标记P8的后一步）。
- X000“OFF”时，不进行跳转，从1步向4步移动，不执行跳转指令。
- 程序定时器T192 ~ T199及高速计数器C235 ~ C255，如果在驱动后跳转则继续工作，输出接点也动作。
- Y001为双线圈，X000=OFF时，不跳转，采样X001。
- X000=ON时跳转至P8，P8处不跳转，采样X012。

3. 跳转程序中软组件的状态

- ✓ 在发生跳转时，被跳过的那段程序中的驱动条件已经没有任何意义了，所以该程序段中的各种继电器和状态器、定时器等将保持跳转发生前的状态不变。

4. 跳转程序中标号的多次引用

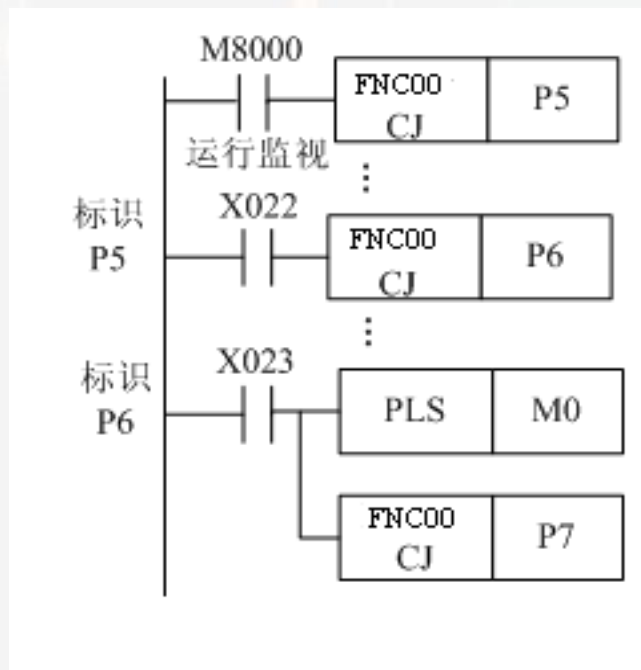
- ✓ 标号是跳转程序的入口标识地址，在程序中只能出现一次，同一标号不能重复使用。但是，同一标号可以多次被引用。



标号可以多次引用

5. 无条件跳转指令的构造

- ✓ PLC只有条件跳转指令，没有无条件跳转指令。遇到需要无条件跳转的情况，可以用条件跳转条件来构造无条件跳转指令，最常使用的是使用M8000（只要PLC处于RUN状态，则M8000总是接通的）





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

子程序调用和返回指令



1. 指令格式

■ 指令编号及助记符:

✓ 子程序调用功能指令 FNC01 CALL , CALL (P)

✓ 子程序返回功能指令 FNC02 SRET

■ 指令的目标操作元件是指针号P0 ~ P62 (允许变址修改)





2. 指令用法

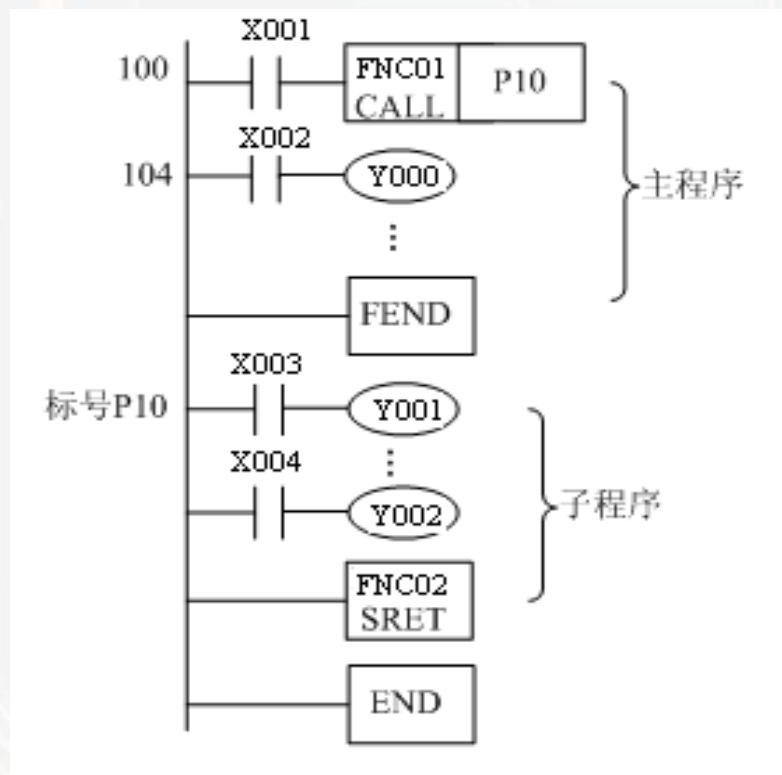
(1) 子程序与标号的位置

- **CALL指令必须和FEND, SRET一起使用。**
- **子程序标号要写在主程序结束指令FEND之后。**
- **标号P0和子程序返回指令SRET间的程序构成了P0子程序的内容。**
- **当主程序带有多个子程序时，子程序要依次放在主程序结束指令FEND之后，并用不同的标号相区别。**

➤ 子程序标号范围为P0 ~ P62，这些标号与条件转移中所用的标号相同，而且在条件转移中已经使用了标号，子程序也不能再用。

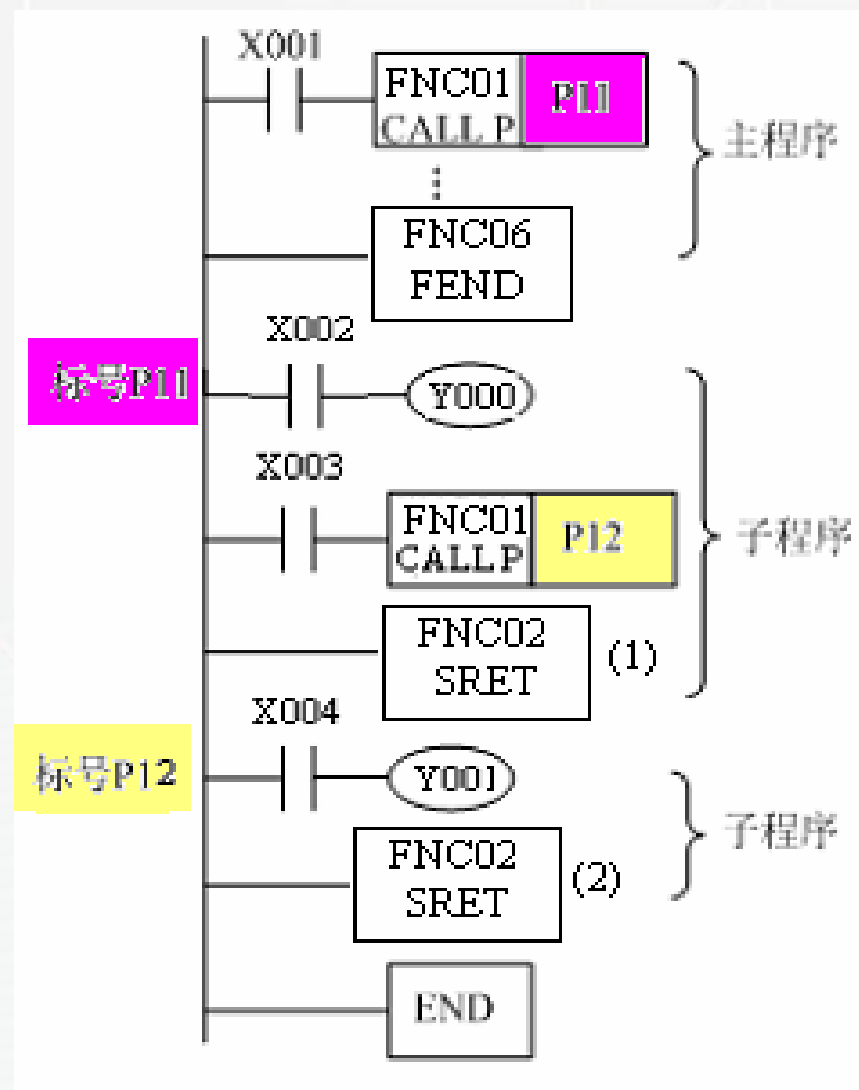
➤ 同一标号只能使用一次，而不同的CALL指令可以多次调用同一标号的子程序。

例：



CALL指令举例

(2) 子程序嵌套



子程序嵌套举例



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

中断指令



1. 指令格式

◆ 指令编号及助记符:

- 中断返回指令 FNC03 IRET
- 中断允许指令 FNC04 EI
- 中断禁止指令 FNC05 DI



2. 指令用法

(1) 3条中断指令

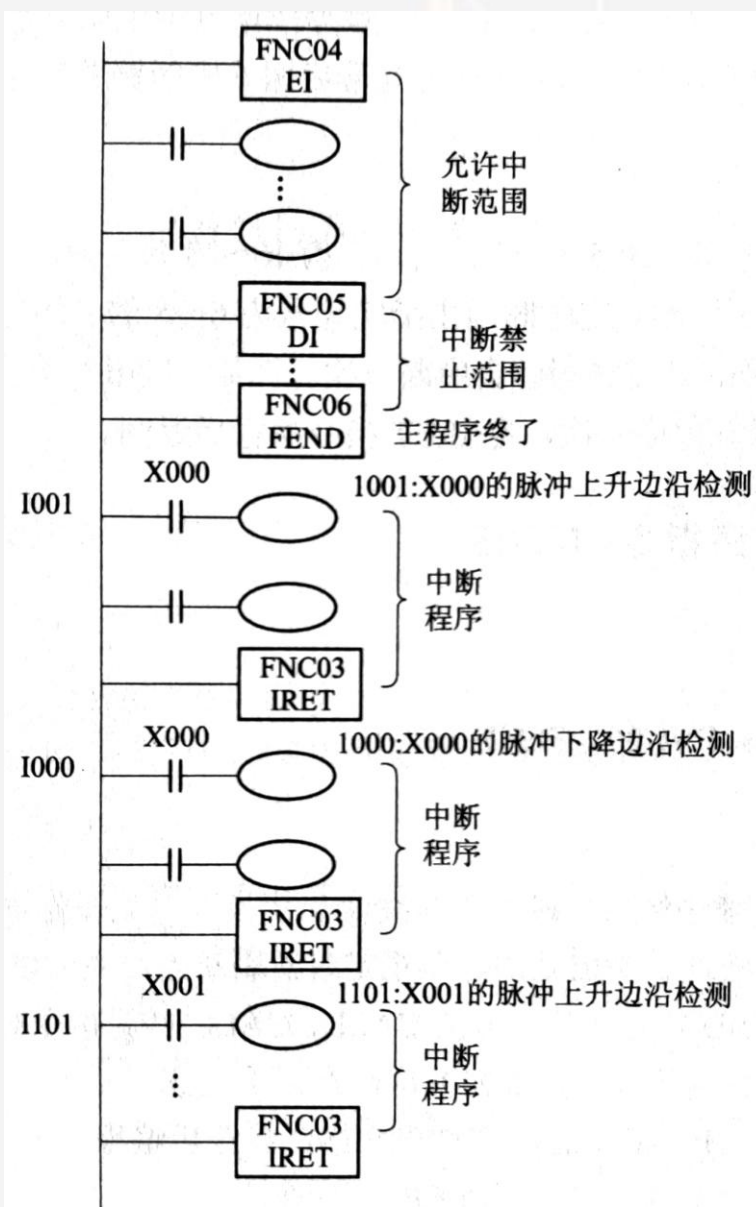
- EI: 对可以响应中断的程序段用EI来开始。
- DI: 对不允许中断的程序段用DI来禁止。
- IRET: 中断服务子程序中返回时必须用IRET。

(2) 优先级

- 依次发生中断，先发生优先；
- 同时发生中断，中断指针号较小的优先；
- 总体外部中断优先级高于内部中断。

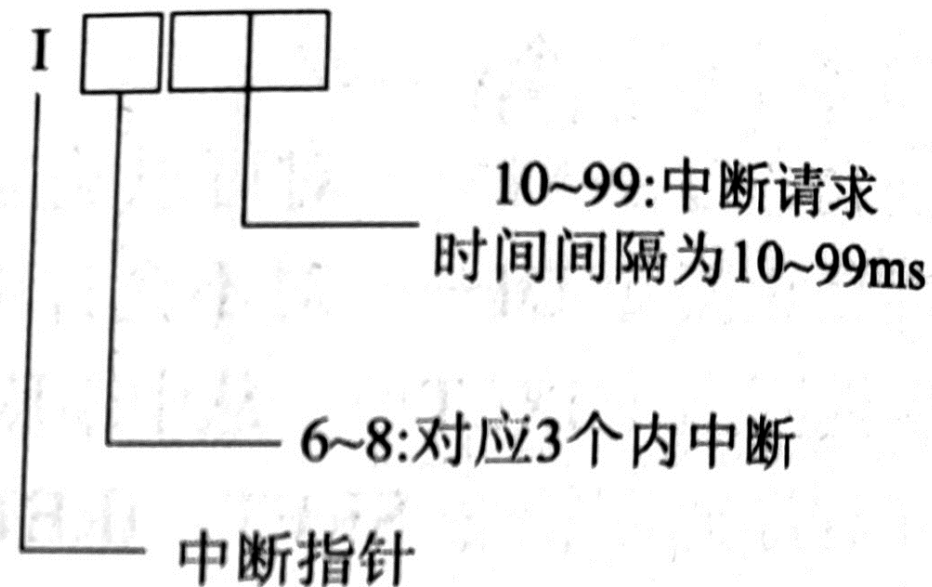
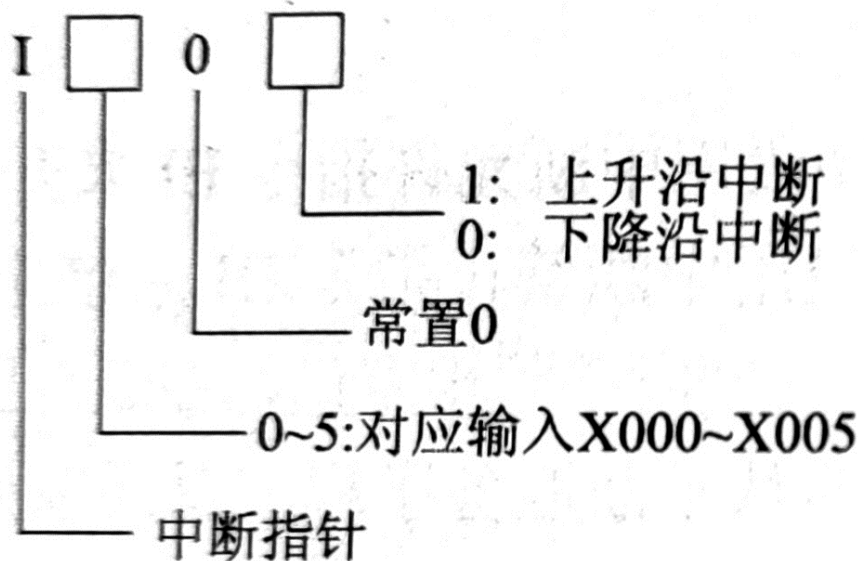


例：中断使用



3. 中断指针

FX2N中断：外部中断和内部定时中断，共9个中断点。





FX2N中断：外部中断和内部定时中断，共9个中断点。

1) 外中断I指针：I0-I5共6个点，中断对应的外部信号的输入口为X000~X005。

例：I001含义，输入X000从OFF→ON变化，执行该指针作为标号后面的中断服务子程序，最后执行IRET返回。

2) 内中断I指针：I6-I8共3个点，内中断是定时时间到去执行中断服务子程序。

例：I630的含义，每隔30ms执行该指针作为标号后面的中断服务子程序，最后执行IRET返回。





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

循环开始结束指令



1. 指令格式

◆ 指令编号及助记符:

➤ 循环开始指令 FNC08 FOR

➤ 循环结束指令 FNC09 NEXT



2. 指令用法

循环指令用于反复执行某程序段。

- FOR和NEXT必须成对使用
- 只有FOR~NEXT之间程序段执行指定次数后，才处理NEXT后面的一步。
- 循环次数由FOR后的数值指定， $n=1\sim 32767$ 。
- 可使用CJ跳出循环。
- 循环次数多时，扫描周期会延长，可能出现监视定时器错误。



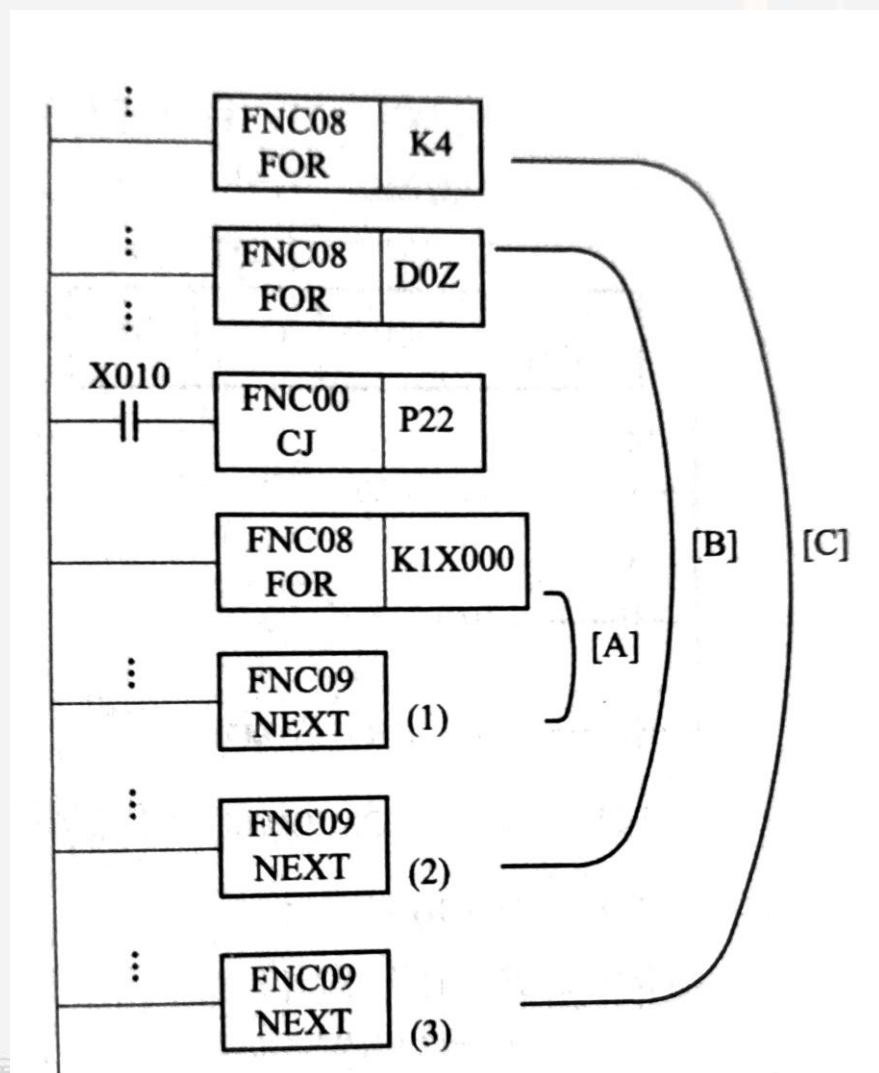


3.循环出错情况

- **NEXT指令在FOR指令之前，无NEXT指令；**
- **在FEND、END指令之后有NEXT指令；**
- **FOR与NEXT指令的个数不一致。**



例：三重循环嵌套



➤ 已知：K1X000内容7，D0Z内容6。

➤ [A]循环：X010为OFF，K1X000内容7，循环7次。

➤ [B]循环：D0Z内容6，[A]循环启动6次。

➤ [C]循环：K4指定4次，[B]循环启动4次。

➤ [A]循环总计： $4 \times 6 \times 7 = 168$ 次。



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

5.3 数据传送和比较指令



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

传送指令



1. 指令格式

➤ 指令编号及助记符：传送指令 FNC12 MOV [S·] [D·]

其中：✓ [S·] 为源数据

✓ [D·] 为目标软组件

➤ 目标操作数为 T、C、V、Z、D、KnY、KnM、KnS

➤ 源操作数的软组件有 T、C、V、Z、D、K、H、KnX、KnY、KnM、KnS

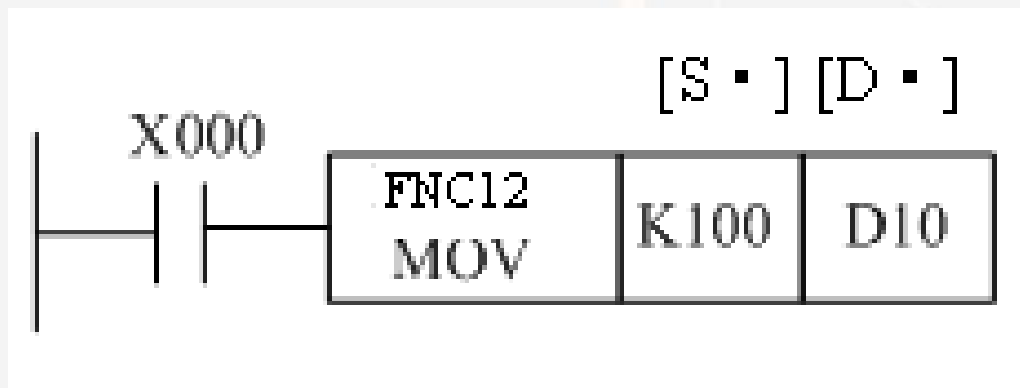




2. 指令用法

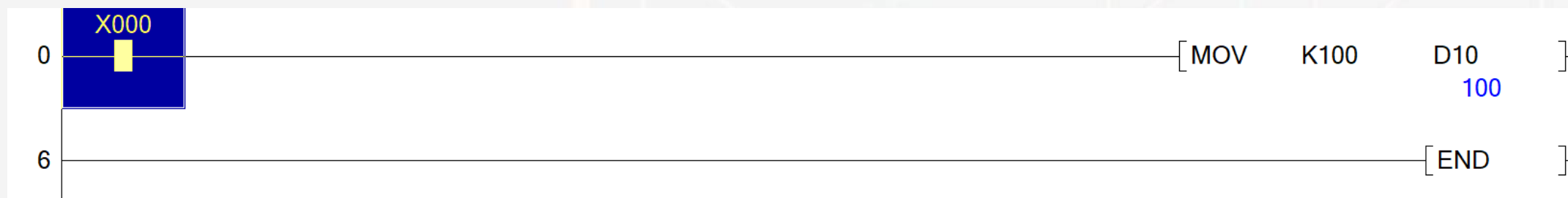
- **传送指令是将源操作数传送到指定的目标操作数，即**
 $[S\cdot] \rightarrow [D\cdot]$

3. 传送指令MOV举例



➤ 当常开触点X000闭合为ON时，每扫描到MOV指令时，就把存入[S·]源数据中操作数100（K100）转换成二进制数，再传送到目标操作数D10中去。

➤ 当X000为OFF时，则指令不执行，数据保持不变。



软元件

☒ 软元件名(N) TC设定值浏览目标

☐ 缓冲存储器(M) 模块起始(U) 地址(A)

显示格式

当前值更改(G)...

软元件	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D10	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	100
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

移位传送指令



1. 指令格式

➤ 指令编号及助记符：移位传送指令 FNC13 SMOV [S·] m1m2[D·]n

其中：

✓ [S·] 为源数据，m1被传起始位，m2传送位数

✓ [D·] 为目标软组件，n传送目标起始位

➤ 目标操作数为T、C、V、Z、D、KnY、KnM、KnS

➤ 源操作数的软组件有T、C、V、Z、D、K、H、KnX、KnY、KnM、KnS

➤ m1、m2、n的软组件有K、H



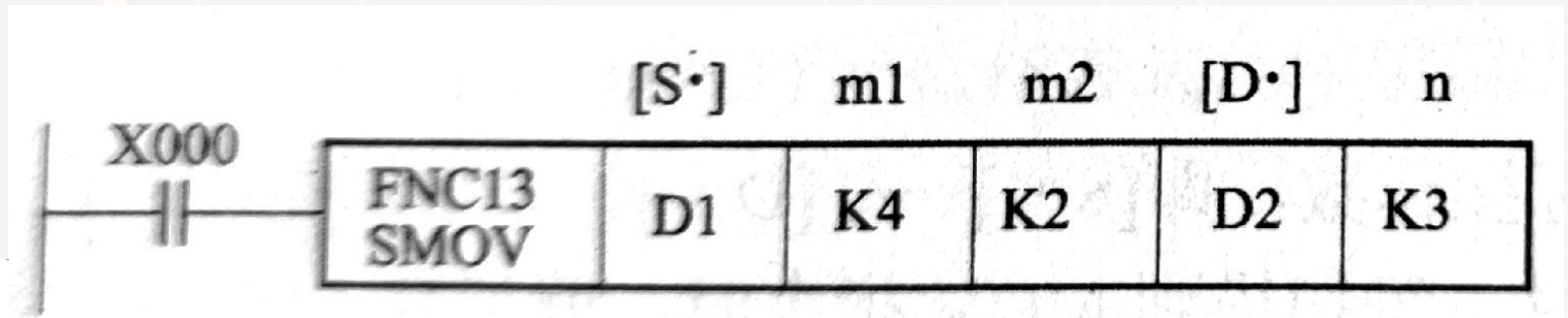


2. 指令用法

- **移位传送指令功能：将[S·]第m1位开始的m2个数移位 [D·]的第n位开始的m2个位置。**
- **m1、m2、n取值均为1~4。**
- **分开BCD码重新组合，一般用于多位BCD拨码开关的输入。**

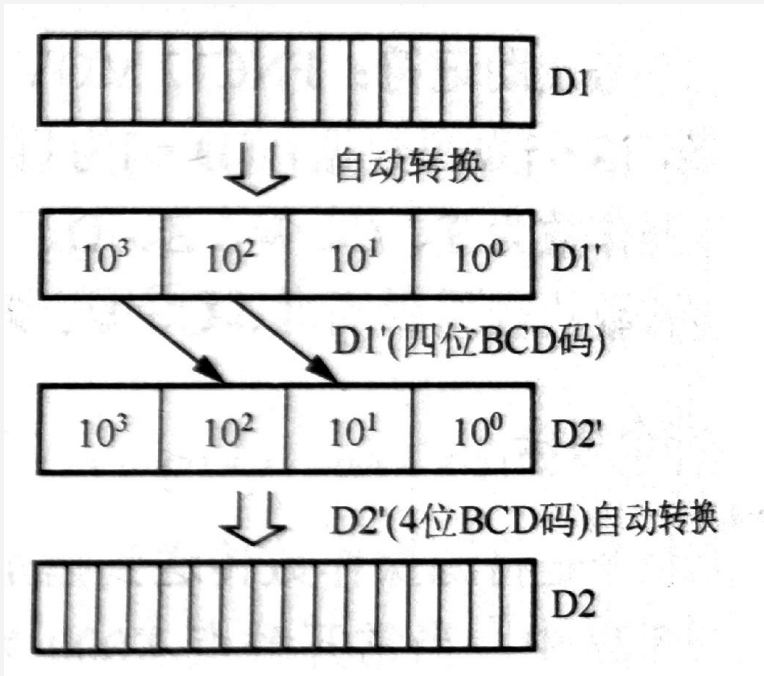


3. 移位传送指令SMOV举例

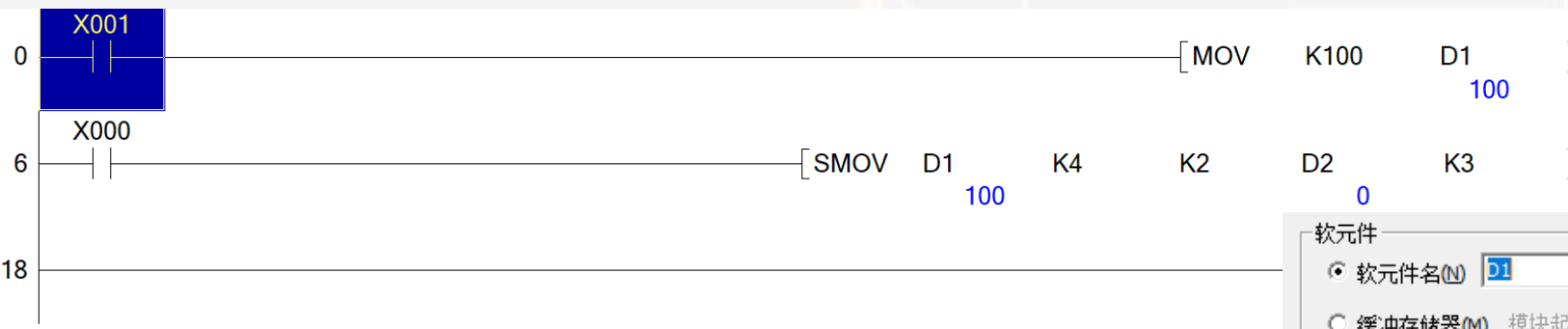


➤ 当常开触点X000闭合为ON时，执行SMOV指令，把[S·]的16位二进制转换成4位BCD码；从右起m1(4)位向右数m2(2)位的数，传送到[D·]右起第n(3),向右数m2(2)位，将[D·]中4位BCD码转换成16位二进制。

➤ 当X000为OFF时，则指令不执行，数据保持不变。



100的BCD码 0100H



软元件

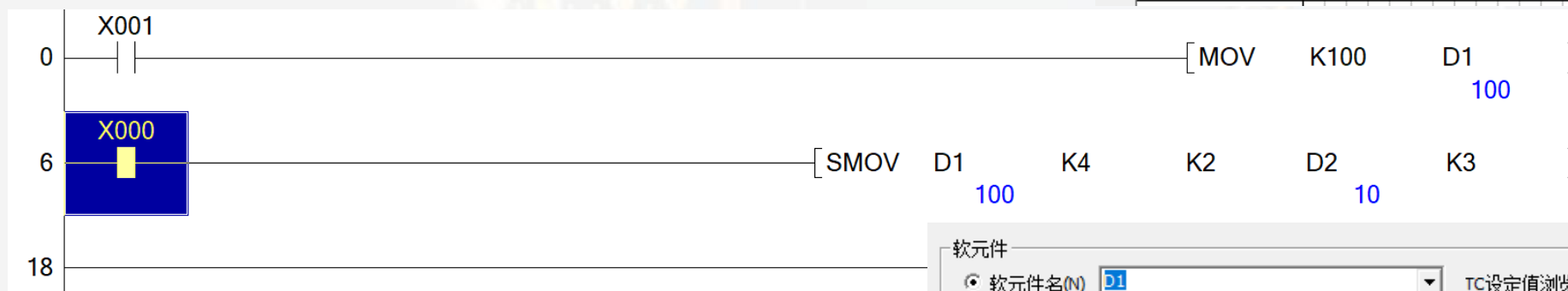
☒ 软元件名(N) D1 TC设定值浏览目标

☐ 缓冲存储器(M) 模块起始(U) 地址(A)

显示格式

当前值更改(G)... 2 W M 16 Bit 32 Bit 32 128 64 128 ASC 10 16 详细(D)... 打开(O)...

软元件	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	100
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



10的BCD码 010H

软元件

☒ 软元件名(N) D1 TC设定值浏览目标

☐ 缓冲存储器(M) 模块起始(U) 地址(A)

显示格式

当前值更改(G)... 2 W M 16 Bit 32 Bit 32 128 64 128 ASC 10 16 详细(D)... 打开(O)...

软元件	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	100
D2	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	10



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

取反传送指令



1. 指令格式

➤ 指令编号及助记符：取反传送指令 FNC14 CML [S·] [D·]

其中：✓ [S·] 为源数据

✓ [D·] 为目标软组件

➤ 目标操作数：T、C、V、Z、D、KnY、KnM、KnS

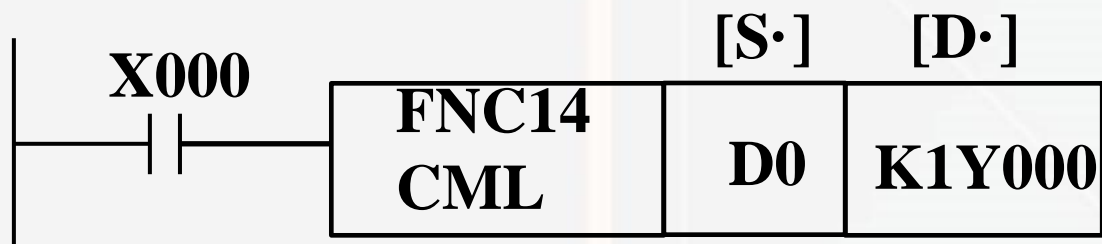
➤ 源操作数的软组件：T、C、V、Z、D、K、H、KnX、KnY、KnM、KnS



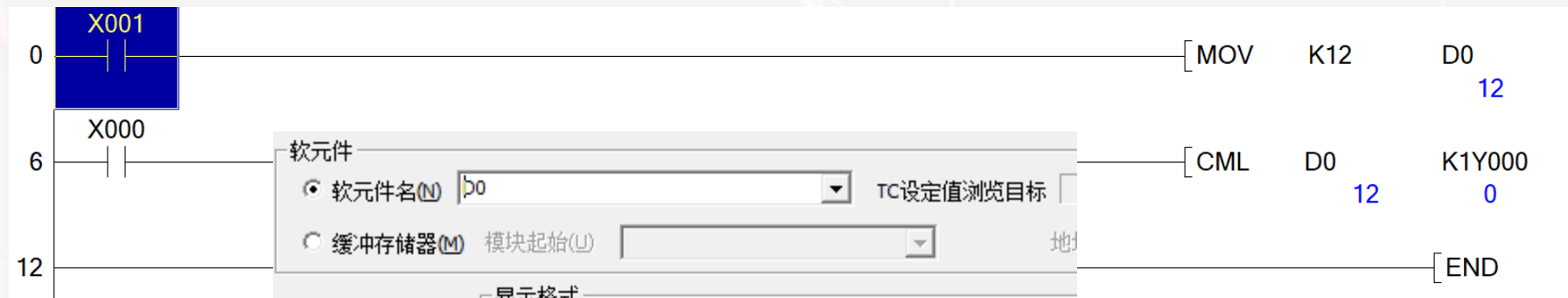
2. 指令用法

- 将[S·]源操作数按二进制的位**逐位取反**后，传送到指定的[D·] 目标操作数。

3. 取反传送指令CML举例



- 当常开触点X000闭合为ON时，执行CML指令时，
- 把存入[S·]源数据中操作数D0(12)转换成二进制数(0000,0000,0000,1100)，**取反**(1111,1111,1111,0011)，再传送到目标操作数以Y0开始的**位元件组件**中去。
- 当X000为OFF时，则指令不执行，数据保持不变。



软元件

☒ 软元件名(N) TC设定值浏览目标

☐ 缓冲存储器(M) 模块起始(U)

显示格式

当前值更改(G)...

软元件	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	12

软元件

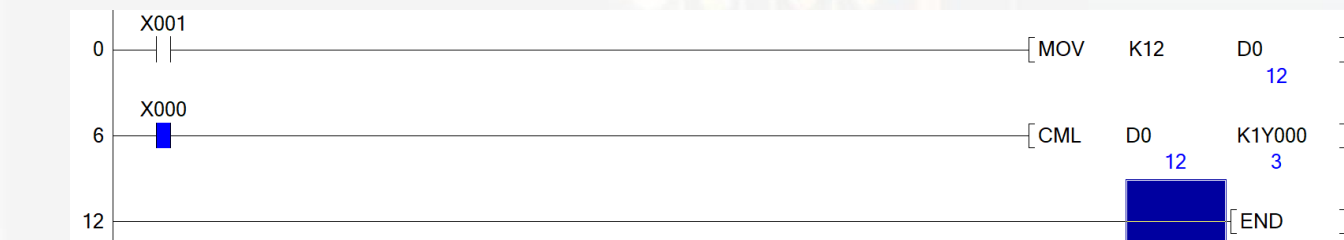
☒ 软元件名(N) T

☐ 缓冲存储器(M) 模块起始(U)

显示格式

当前值更改(G)...

软元件	7	6	5	4	3	2	1	0	
Y000	0	0	0	0	0	0	1	1	
Y010	0	0	0	0	0	0	0	0	



软元件

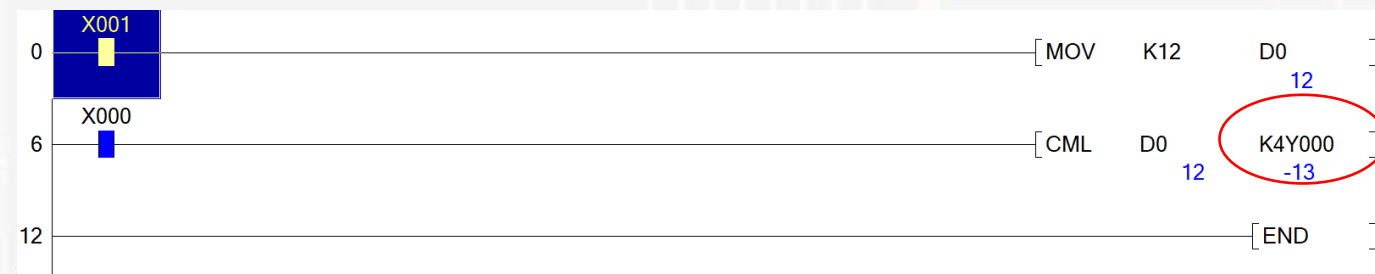
☒ 软元件名(N)

☐ 缓冲存储器(M) 模块起始(U)

显示格式

当前值更改(G)...

软元件	7	6	5	4	3	2	1	0	
Y000	1	1	1	1	0	0	1	1	
Y010	1	1	1	1	1	1	1	1	





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

块传送指令



1. 指令格式

➤ 指令编号及助记符：块传送指令 FNC15 BMOV [S·] [D·]n

其中：✓ [S·] 为源软组件

✓ [D·] 为目标软组件

✓ n 为数据块个数

➤ 目标操作数为 T、C、V、Z、D、KnY、KnM、KnS

➤ 源操作数的软组件有 T、C、V、Z、D、K、H、KnX、KnY、KnM、KnS

➤ n：常数 K、H



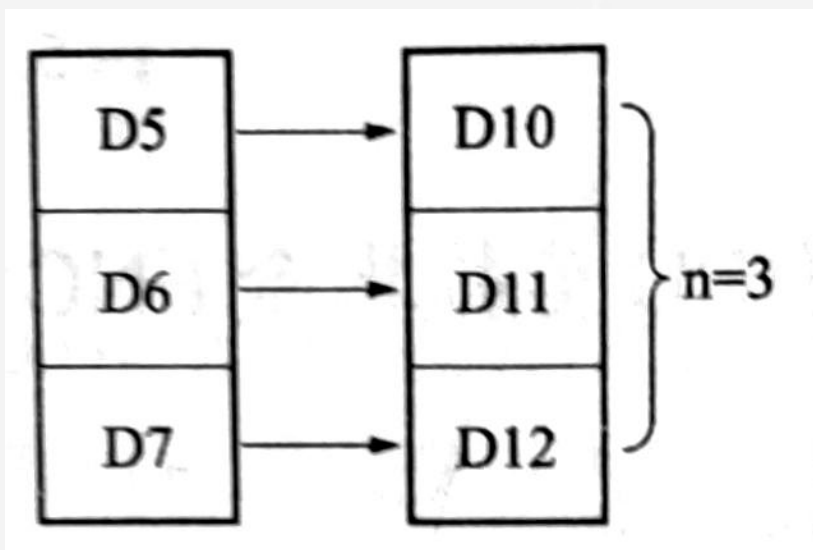
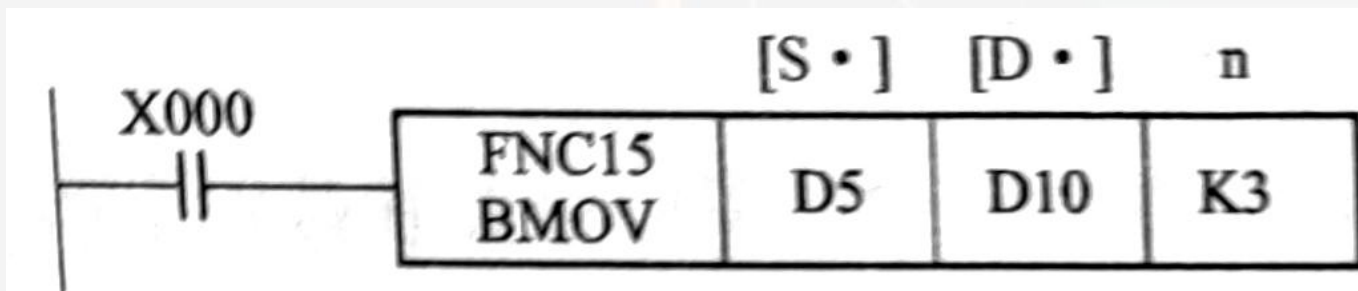


2. 指令用法

- 块传送指令是将源操作数组件中 n 个数据组成的数据块传送到指定的目标软组件中。
- 如果组件号超出允许组件号的范围，数据仅传送到允许范围内。
- 如源、目标软组件同类型，传送顺序自动决定；
- 如果源、目标软组件类型不同，位数相同就可以正确传送；

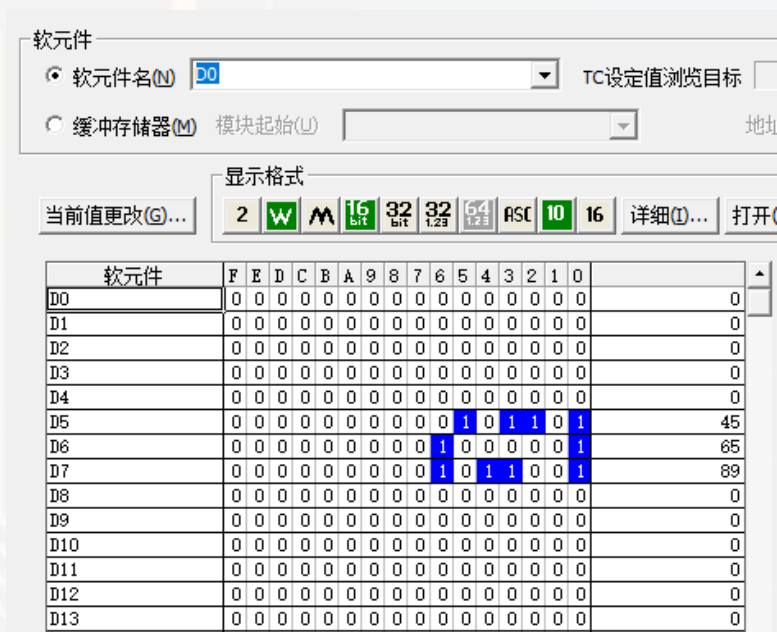


3. 块传送指令BMOV举例



➤ 当常开触点X000闭合为ON时，执行BMOV指令，把D5-D7这3个数据块，传送到目标操作数D10-D12中去。

➤ 当X000为OFF时，则指令不执行，数据保持不变。



[illegible]



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

数据交换指令



1. 指令格式

➤ 指令编号及助记符：数据交换指令 FNC17 XCH [D1·] [D2·]

其中：✓ [D1·] 为目标软组件1

✓ [D2·] 为目标软组件2

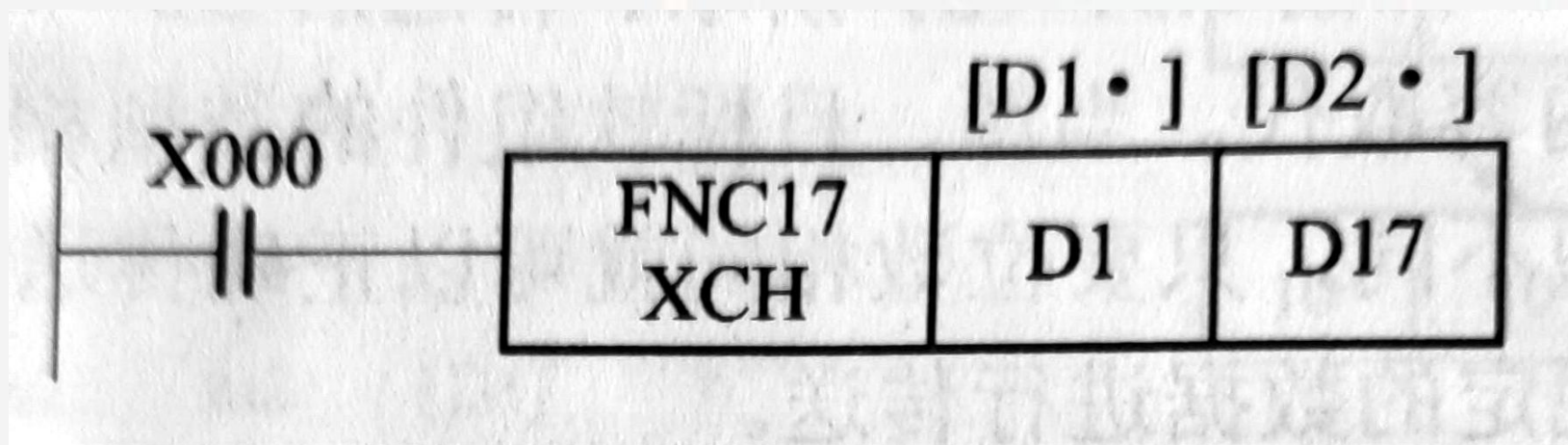
➤ 目标操作数为 T、C、V、Z、D、KnY、KnM、KnS



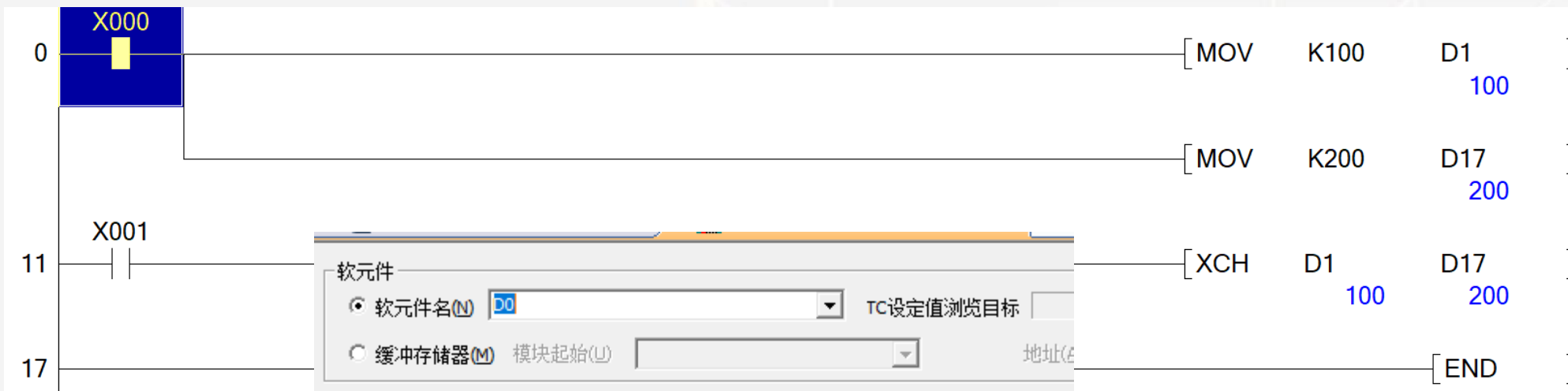
2. 指令用法

- **数据交换指令是将数据在两个指定的目标软组件之间交换，即 $[D1\cdot] \longleftrightarrow [D2\cdot]$**

3. 数据交换指令XCH举例



- 当常开触点X000闭合为ON时，执行XCH指令，将D1(100)与D17(200)中数据交换，结果D1(200),D17(100)。
- 当X000为OFF时，则指令不执行，数据保持不变。



软元件

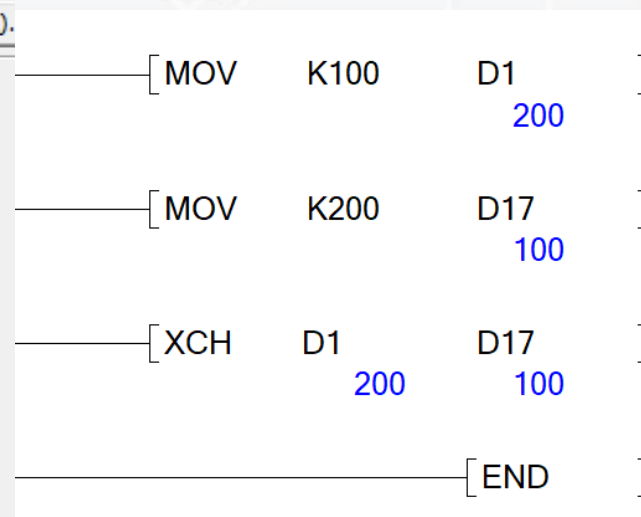
☒ 软元件名(N) TC设定值浏览目标

☐ 缓冲存储器(M) 模块起始(U) 地址(A)

显示格式

当前值更改(G)...

软元件	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
D0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D1	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	200
D2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D17	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	100





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

BCD变换指令



1. 指令格式

- 指令编号及助记符：变换传送指令FNC18 BCD [S·] [D·]

其中： ✓ [S·] 为被转换的软组件

✓ [D·]为目标软组件

- 目标操作数为T、C、V、Z、D、KnY、KnM、KnS
- 源操作数的软组件有T、C、V、Z、D、K、H、KnX、KnY、KnM、KnS



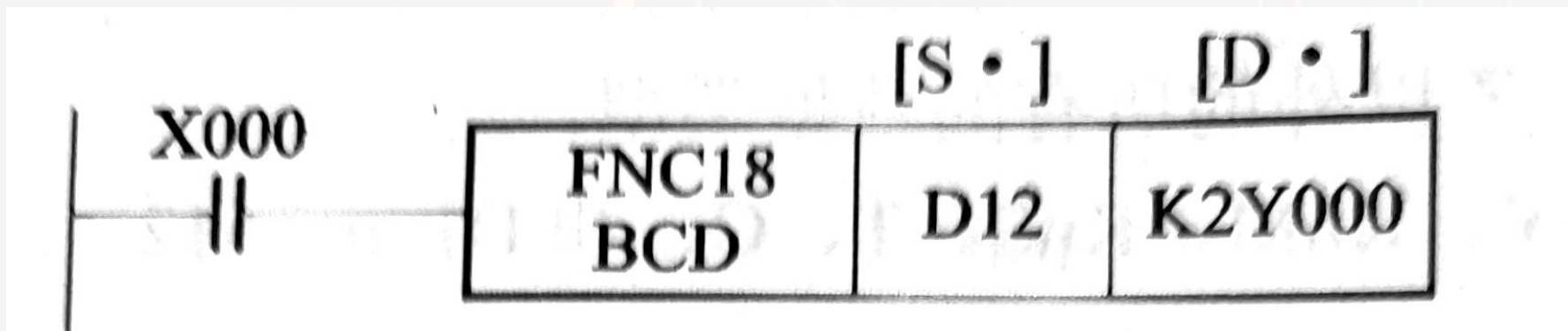


2. 指令用法

- **BCD交换指令是将源操作数中的二进制数转换成BCD码并传送到指定的目标操作数。**



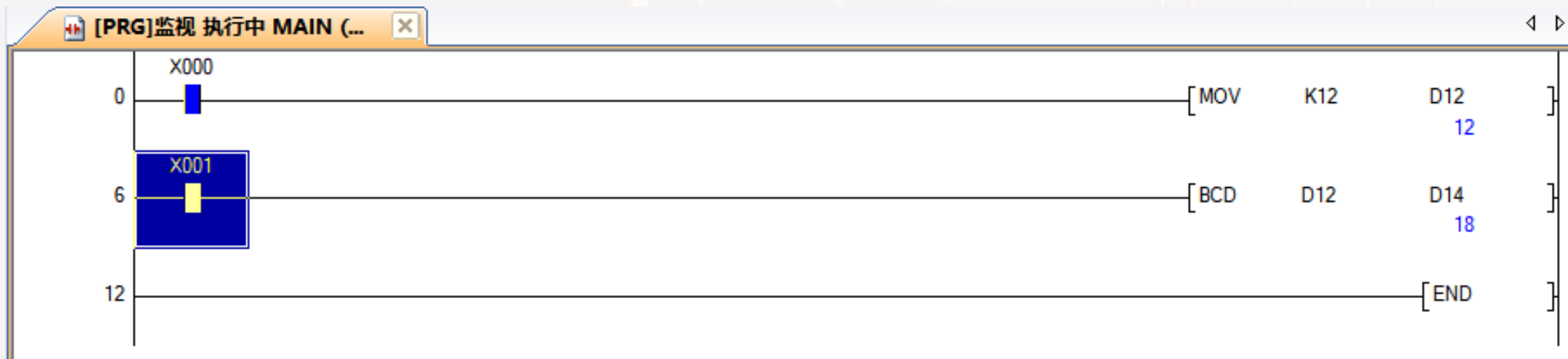
3. BCD变换指令举例



➤ 当常开触点X000闭合为ON时，执行BCD指令，把[S·]源数据D12 中的二进制数(0000,0000,0000,1100)变换成BCD(0000,0000,0001,0010)后，再译成7段码，就能输出驱动LED显示器。

➤ 当X000为OFF时，则指令不执行，数据保持不变。

4. BCD变换指令举例仿真运行



- 当常开触点X000闭合为ON时，执行MOV指令，数12 存入D12 中的二进制数(0000,0000,0001,0010)。
- 当常开触点X001闭合为ON时，执行BCD指令，D12 中的二进制数(0000,0000,0001,0010)变换成BCD(0000,0000,0001,1000)后存入D14。



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

BIN变换指令



1. 指令格式

➤ 指令编号及助记符：BIN变换指令FNC19 BIN [S·] [D·]

其中：✓ [S·] 为被转换的软组件

✓ [D·]为目标软组件

➤ 源操作数的软组件有T、C、V、Z、D、K、H、KnX、KnY、KnM、KnS

➤ 目标操作数为T、C、V、Z、D、KnY、KnM、KnS

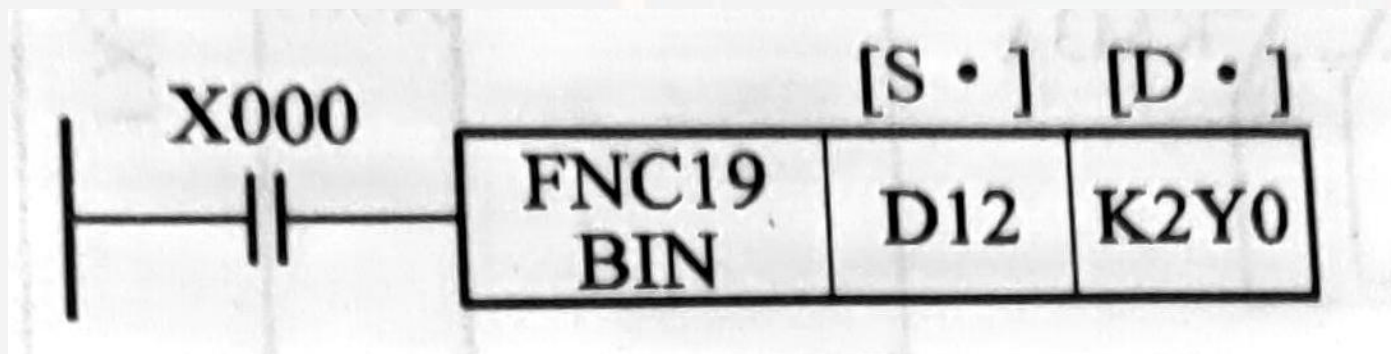




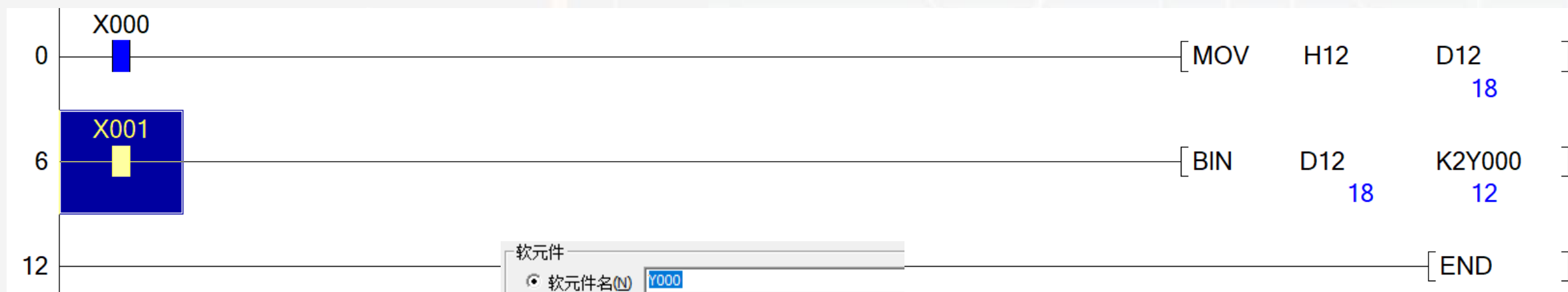
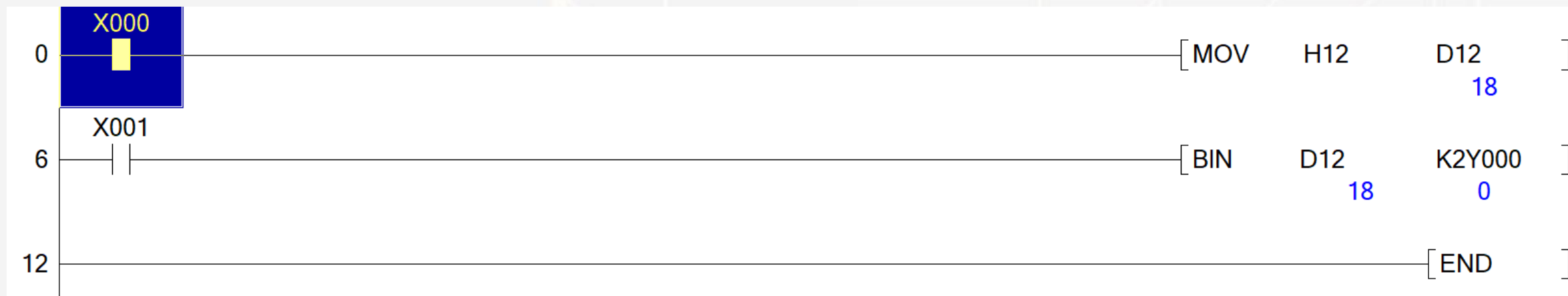
2. 指令用法

- **BIN变换指令是将源元件中的BCD码转换成二进制数传送到指定的目标操作数。**
- **四则运算与增量、减量指令与可编程控制器内部的运算都用BIN码进行。输入时，数字开关获取BCD码的信息，要用FNC19转换指令。**

3. 变换指令BIN举例



- 当常开触点X000闭合为ON时，执行BIN，就把存入[S·]源数据中D12(0000,0000,0001,0010)BCD码转换成二进制数(0000,0000,0000,1100)，再传送到目标操作数K2Y0的位元件组件。
- 当X000为OFF时，则指令不执行，数据保持不变。

[illegible]



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

比较指令



1. 指令格式

➤ 比较指令 FNC10 $\text{CMP}[\text{S1}\cdot][\text{S2}\cdot][\text{D}\cdot]$

其中：

- ✓ $[\text{S1}\cdot][\text{S2}\cdot]$ 为两个比较的源操作数。
- ✓ $[\text{D}\cdot]$ 为比较结果的标志组件，指令中给出的是标志软组件的首地址(标号最小的那个)。



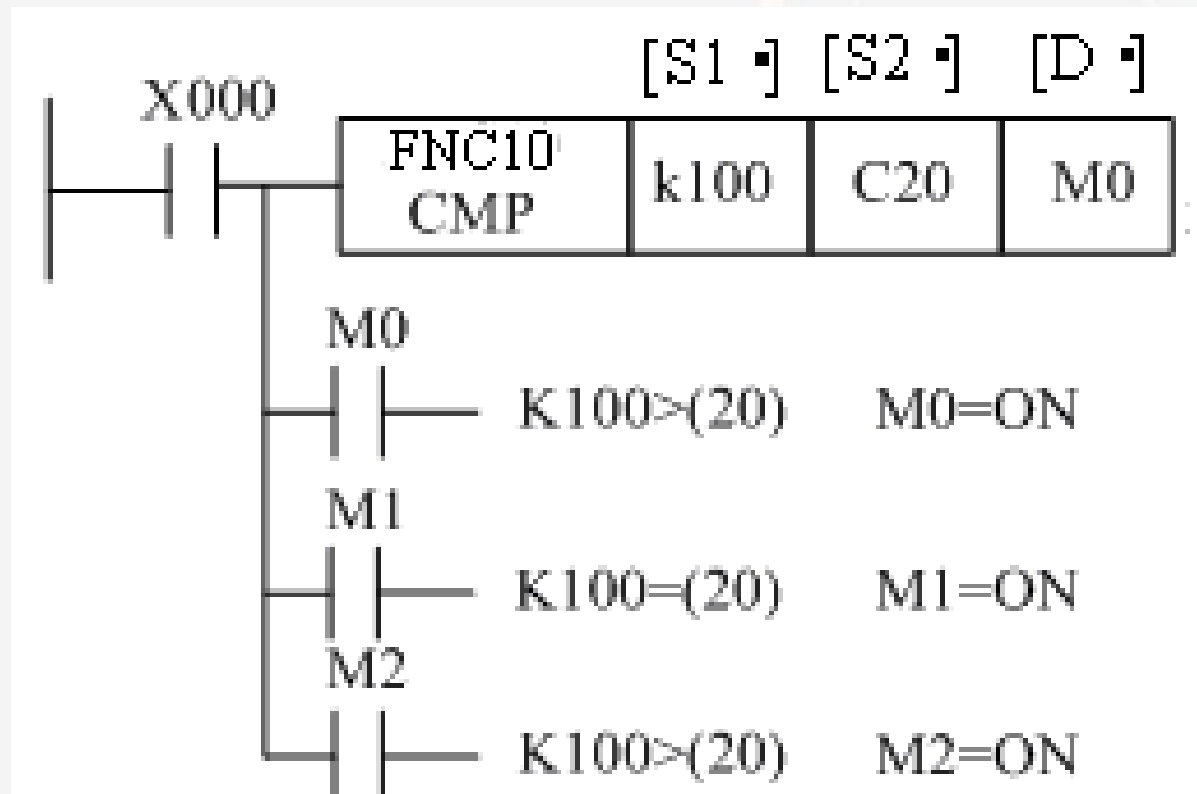


2. 指令用法

- 比较指令CMP是将源操作数[S1·]和源操作数[S2·]进行比较，结果送到目标操作数[D·]中，比较结果有三种情况：大于、等于和小于
- CMP指令也可以有脉冲操作方式，使用后缀(P)：(D)CMP(P)[S1·][S2·][D·]，只有在驱动条件由OFF→ON时进行一次比较。

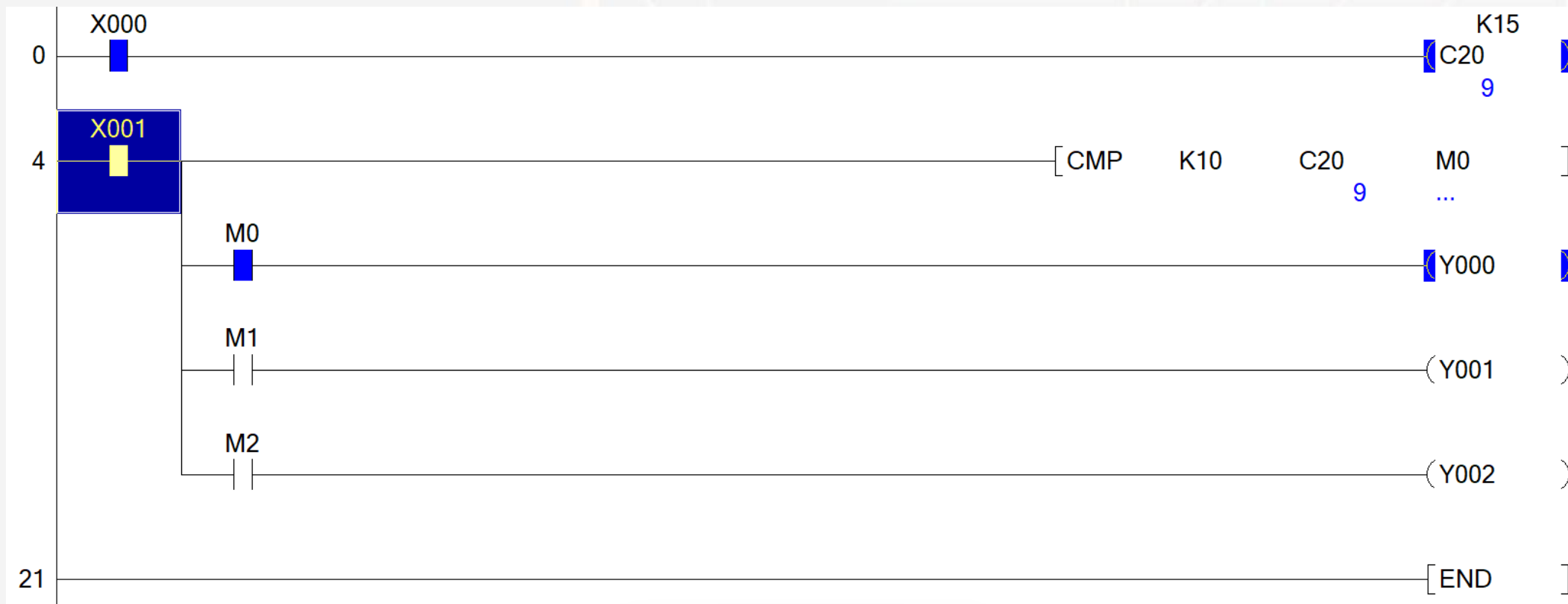


3. 比较指令CMP举例

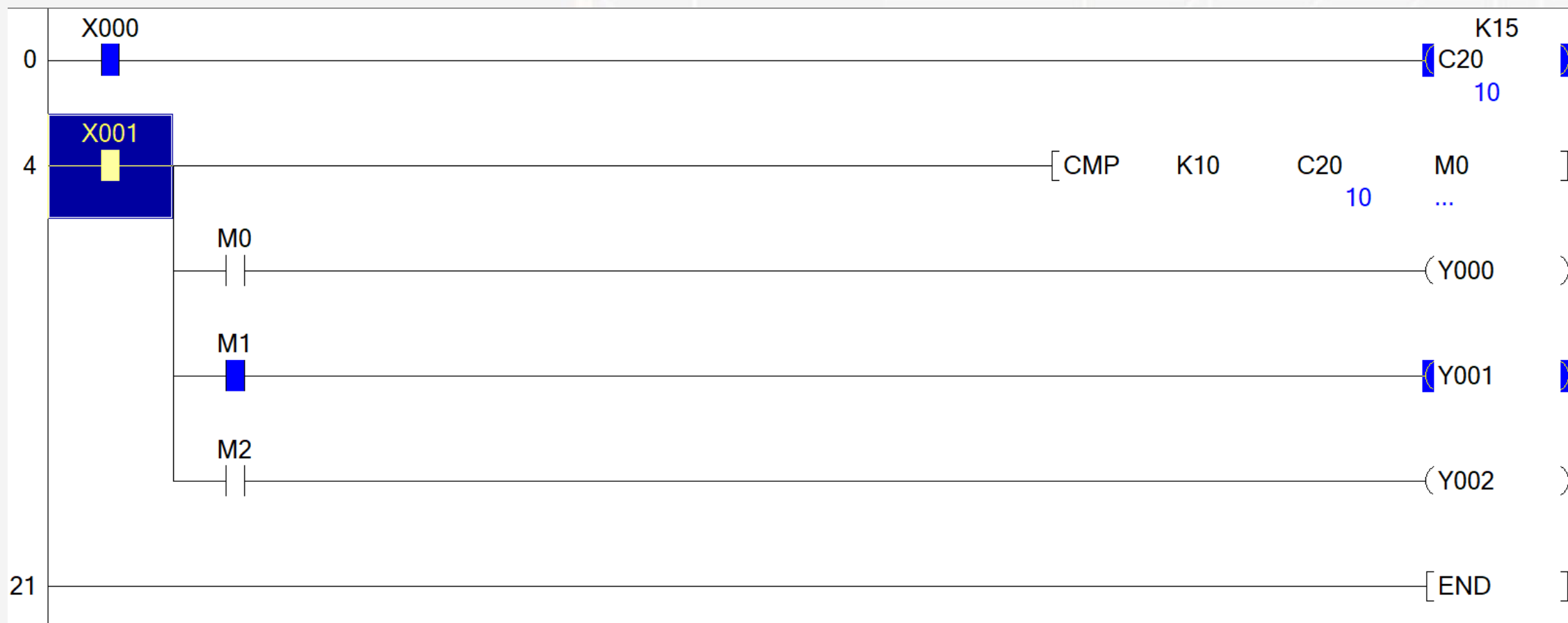


标志位操作规则:

- ✓ 若 $K100 > (C20)$, 则M0被置1
- ✓ 若 $K100 = (C20)$, 则M1被置1
- ✓ 若 $K100 < (C20)$, 则M2被置1



✓ 若 $K10 > (C20)$ ，则M0被置1



✓ 若K10=(C20), 则M1被置1



✓ 若 $K10 < (C20)$, 则M2被置1



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

5.4 算术运算和逻辑运算指令



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

加法指令



1. 指令格式

➤ **二进制加法指令** FNC 20 ADD[S1·][S2·][D·]

其中： ✓ [S1·]、[S2·]为两个作为加数的源操作数

✓ [D·]为存放相加结果的目标组件

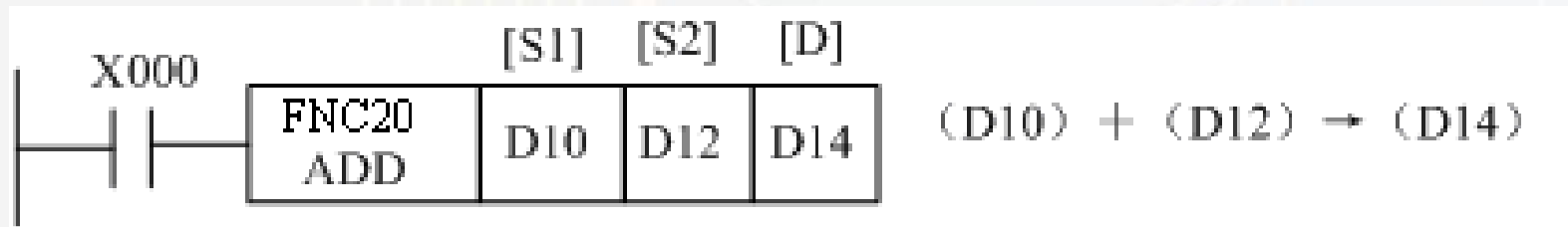
➤ **源操作数可取所有数据类型。**

➤ **目标操作数可取KnY、KnM、KnS、T、C、D、V和Z。**

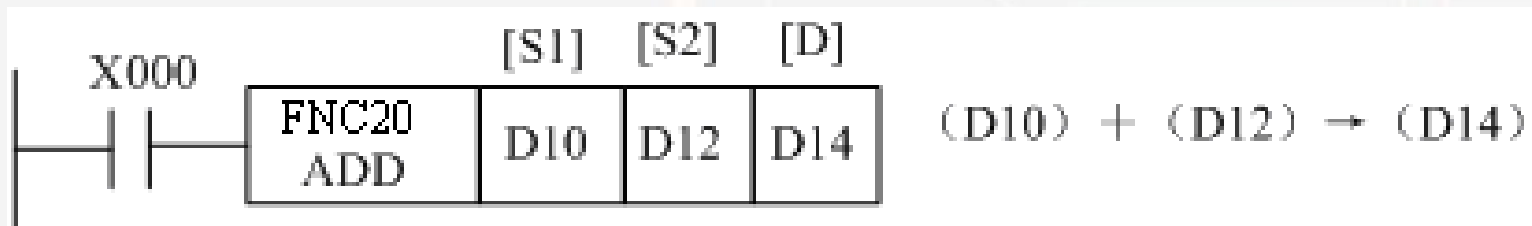


2. 指令用法

- ADD指令将两个源操作数[S1]、[S2]相加，结果放到目标元件中[D]中。

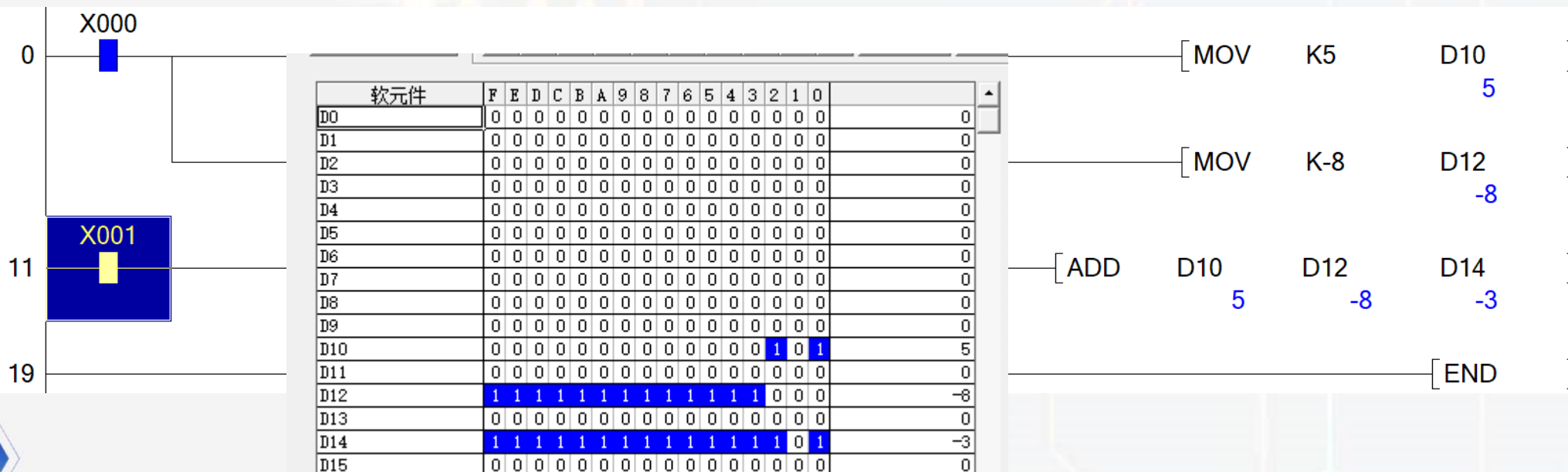


3. ADD指令举例



- 两个源数据进行二进制加法后传递到目标处，各数据的最高位是正（0）、负（1）的符号位，这些数据以代数形式进行加法运算，如

$$5 + (-8) = -3$$





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

减法指令SUB



1. 指令格式

➤ **减法指令FNC21 SUB [S1·] [S2·] [D·]**

其中：✓ [S1·] [S2·]分别为作为被减数和减数的源软件组件。

✓ [D·]为存放相减差的目标组件。

➤ **源操作数可取所有数据类型**

➤ **目标操作数可取KnY、KnM、KnS、T、C、D、V和Z**

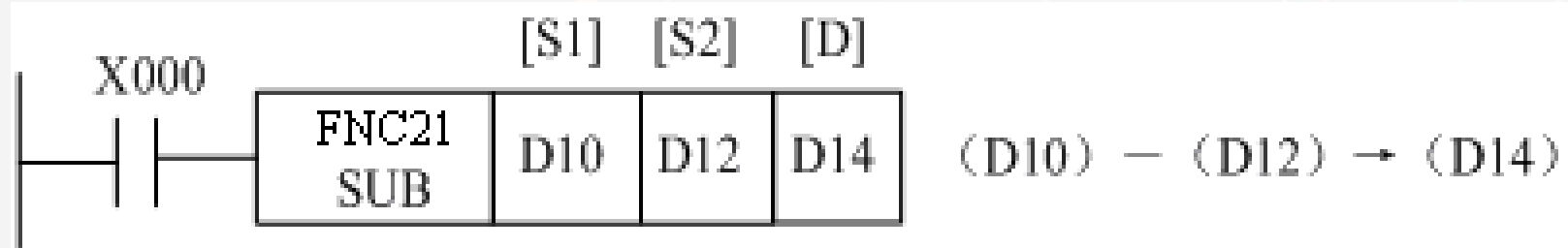




2. 指令用法

- **SUB指令是将指定的两个源软组件中的有符号数，进行二进制代数减法运算，然后将相减的结果差送入指定的目标软组件中。**
- **减法指令标志区功能，32位运算元中指定方法与加法指令相同**

3. SUB指令举例





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

乘法指令 MUL



1. 指令格式

➤ 乘法指令 FNC22 MUL [S1·] [S2·] [D·]

其中：✓ [S1·] [S2·] 分别为作为被乘数和乘数的源软件组件。

✓ [D·] 为存放相乘积的目标组件的**首地址**。

➤ 源操作数可取所有数据类型

➤ 目标操作数可取 KnY、KnM、KnS、T、C、D、V 和 Z



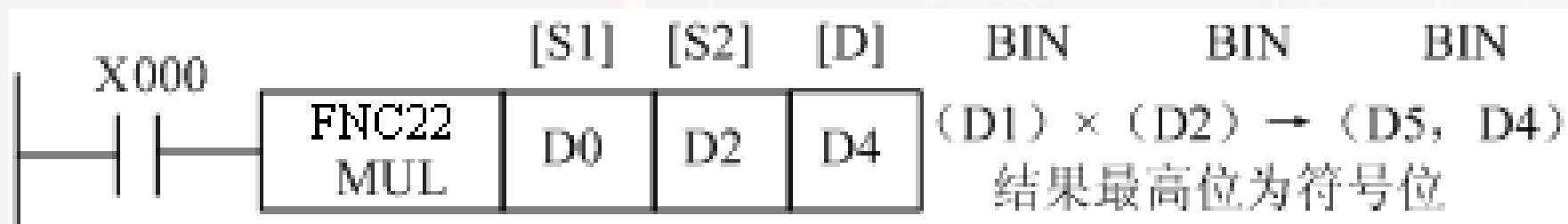


2. 指令用法

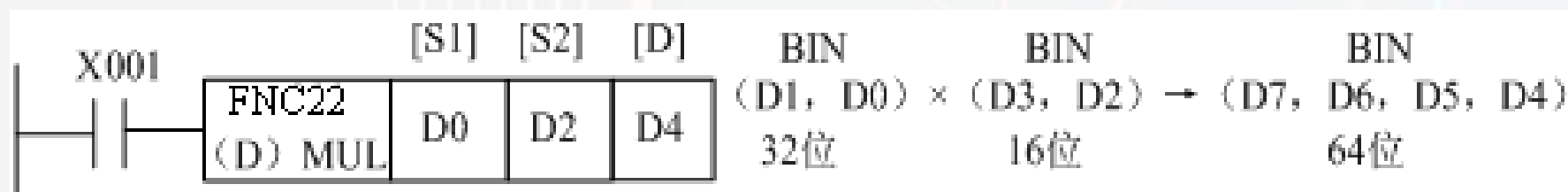
- **MUL指令的功能是将指定的[S1·]、[S2·]两个源软组件中的数进行二进制代数乘法运算，然后将相乘结果积送入指定的目标软组件中。**

2. MUL指令举例

16位 MUL指令举例之一



32位 MUL指令举例之二





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

除法指令DIV



1. 指令格式

➤ 除法指令 **FNC23 DIV [S1·] [S2·] [D·]**

其中： ✓ [S1·] [S2·] 分别为作为被除数和除数的源软组件。

✓ [D·] 为商和余数的目标组件的首地址。

➤ 源操作数可取所有数据类型。

➤ 目标操作数可取 **KnY、KnM、KnS、T、C、D、V和Z**。





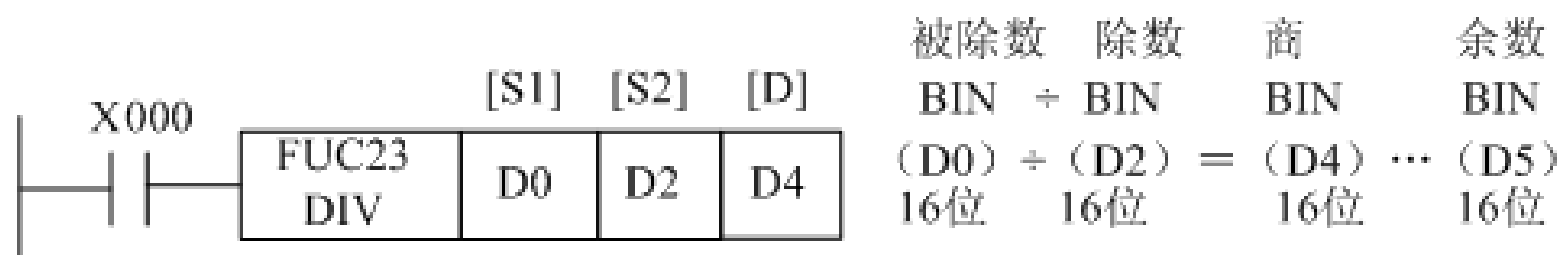
2. 指令用法

- **DIV指令的功能是将指定的两个源软组件中的数，进行二进制有符数除法运算，然后将相除的商和余数送入指定的目标软组件中。**



3. DIV指令用法举例

16位 DIV指令举例之一

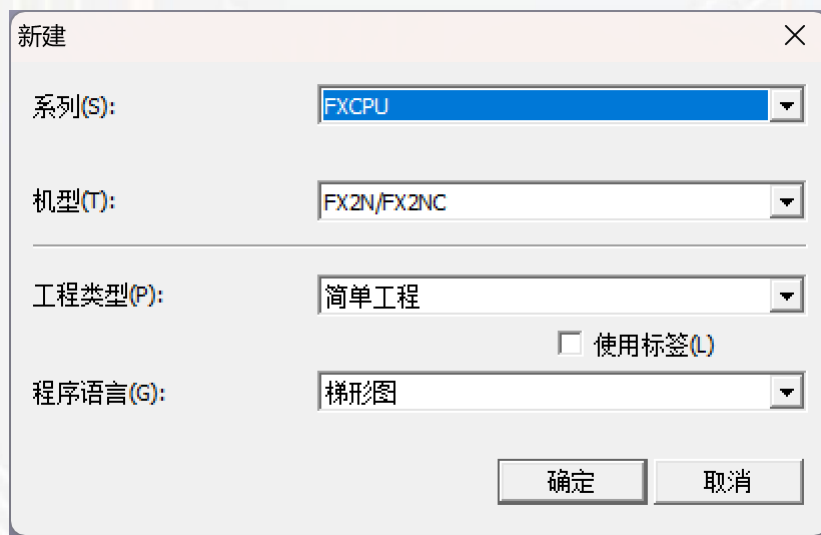


32 位DIV指令举例之二



案例：算术运算

- 实现 $45X/35+3$ 。式中：X代表从输入口K2X000送入的二进制数，运算结果送入输出口K2Y000；X020为启停开关。







武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

加1指令、减1指令



1. 指令格式

- 指令编号及助记符：加1指令FNC24 INC [D·]
减1指令FNC25 DEC [D·]

其中：

✓ [D·]是要加1（或要减1）的目标软组件

- 目操作数的软组件为KnY、KnM、KnS、T、C、D、V和Z.





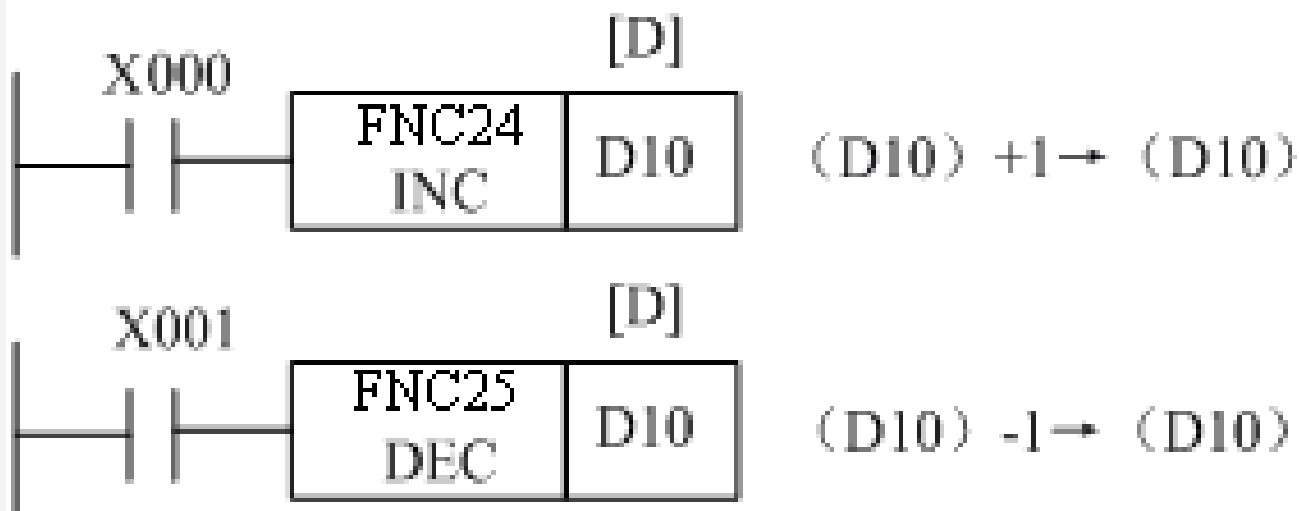
2. 指令用法

- **INC指令的功能是将指定的目标软组件的内容加1。**
- **DEC指令的功能是将指定的目标软组件的内容减1。**





3.INC和DEC指令举例

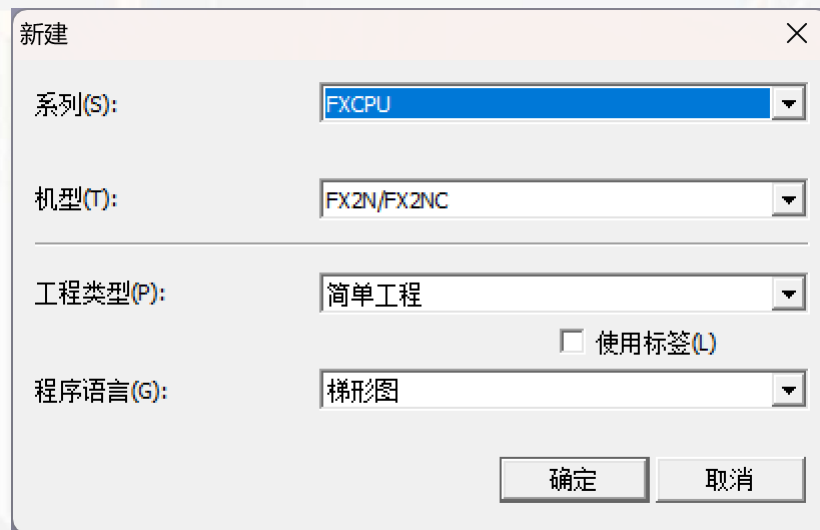


- ✓ 16位运算时，如果+32767加1变成 - 32768，标志位不置位；32位运算时，如果+2147483647加1变成 - 2147483648，标志位不置位
- ✓ 在连续执行指令中，每个扫描周期都将执行运算，必须加1注意。所以一般采用输入信号的上升沿触发运算一次
- ✓ 16位运算时，如果 - 32768再减1，值变为+32767，标志位不置位；32位运算时，如果 - 2147483648再减1，值变为+2147483647，标志位不置位



案例：彩灯控制

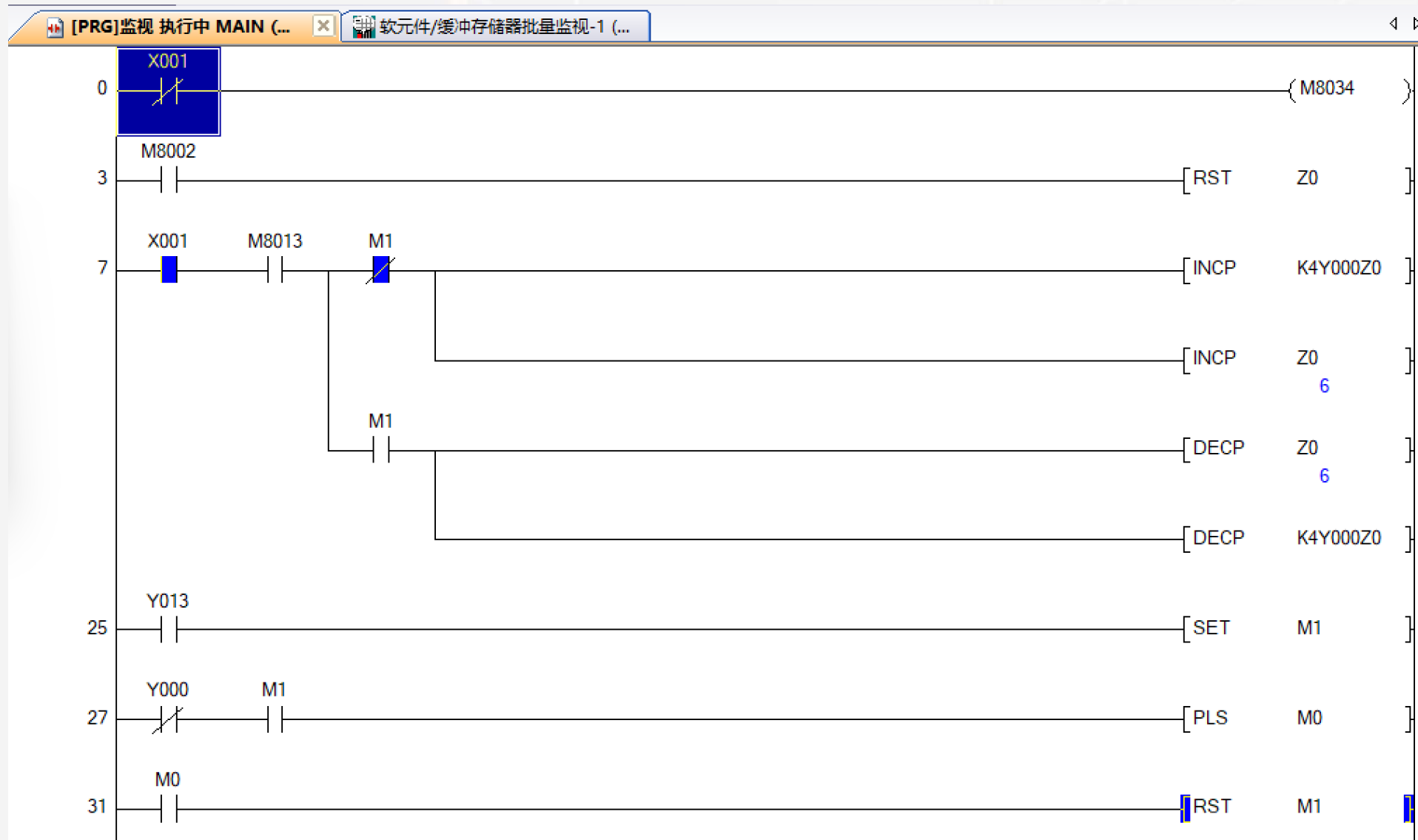
- 彩灯12盏，各彩灯状态变化的时间为1秒钟，X001控制彩灯开关。要求正序亮，到全亮，反序熄至全熄，循环控制。



M8013:1s定时



INC(P)



[PRG]监视 执行中 MAIN (只读) ... 软件元件/缓冲存储器批量监视-...

软元件

☒ 软元件名(N) Y000 TC设定值浏览目标 浏览

☐ 缓冲存储器(M) 模块起始(U) 地址(A) 10j

显示格式

当前值更改(G)... 2 W M 16 Bit 32 Bit 32 1.23 64 1.23 RSC 10 16 详细(D)... 打开(O)... 保存(S)... 不显示注释

软元件	7	6	5	4	3	2	1	0
Y000	1	1	1	1	1	1	1	1
Y010	0	0	0	0	0	0	1	1
Y020	0	0	0	0	0	0	0	0
Y030	0	0	0	0	0	0	0	0
Y040	0	0	0	0	0	0	0	0
Y050	0	0	0	0	0	0	0	0
Y060	0	0	0	0	0	0	0	0
Y070	0	0	0	0	0	0	0	0
Y100	0	0	0	0	0	0	0	0
Y110	0	0	0	0	0	0	0	0
Y120	0	0	0	0	0	0	0	0
Y130	0	0	0	0	0	0	0	0
Y140	0	0	0	0	0	0	0	0
Y150	0	0	0	0	0	0	0	0
Y160	0	0	0	0	0	0	0	0
Y170	0	0	0	0	0	0	0	0
Y200	0	0	0	0	0	0	0	0
Y210	0	0	0	0	0	0	0	0
Y220	0	0	0	0	0	0	0	0
Y230	0	0	0	0	0	0	0	0
Y240	0	0	0	0	0	0	0	0

当前值更改

软元件/标签 缓冲存储器

软元件/标签(E) X001

数据类型(T) Bit

ON OFF ON/OFF取反(I)

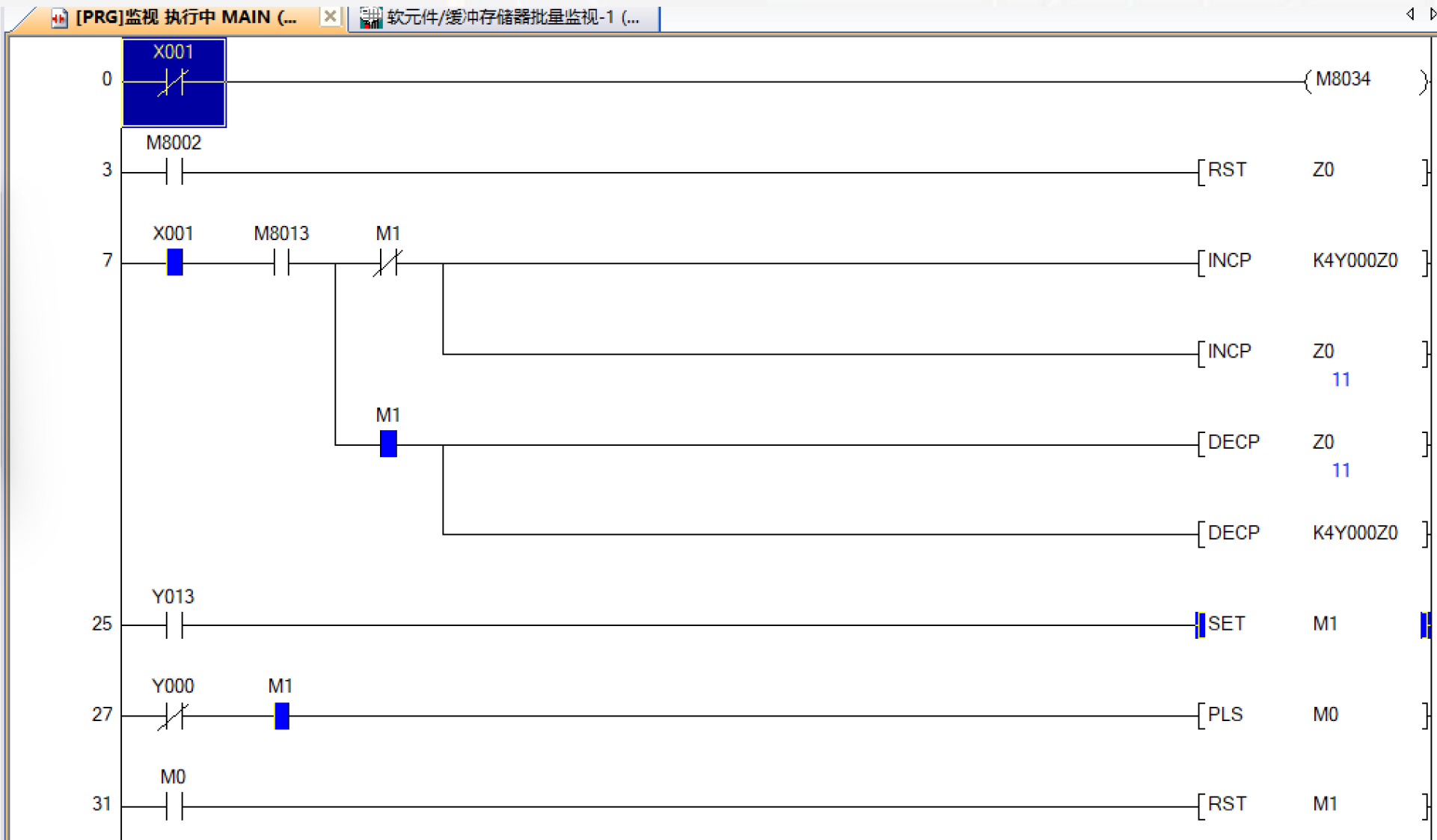
可输入范围

执行结果(R)▲ 关闭

执行结果(L)

软元件/标签	数据类型	设定值
X001	Bit	ON

DEC(P)





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

逻辑与、或和异或



1. 指令格式 指令编号及助记符:

➤ (1) 逻辑“与”指令 FNC26 WAND [S1·] [S2·] [D·]

其中: ✓ [S1·] [S2·] 两个相“与”的源软组件

✓ [D·] 相“与”结果的目标组件

➤ (2) 逻辑“或”指令 FNC27 WOR [S1·] [S2·] [D·]

其中: ✓ [S1·] [S2·] 两个相“或”的源软组件

✓ [D·] 相“或”结果的目标组件

➤ (3) 逻辑“异或”指令 FNC28 WXOR [S1·] [S2·] [D·]

其中: ✓ [S1·]、[S2·] 两个相“异或”的源软组件

✓ [D·] 相“异或”结果的目标组件



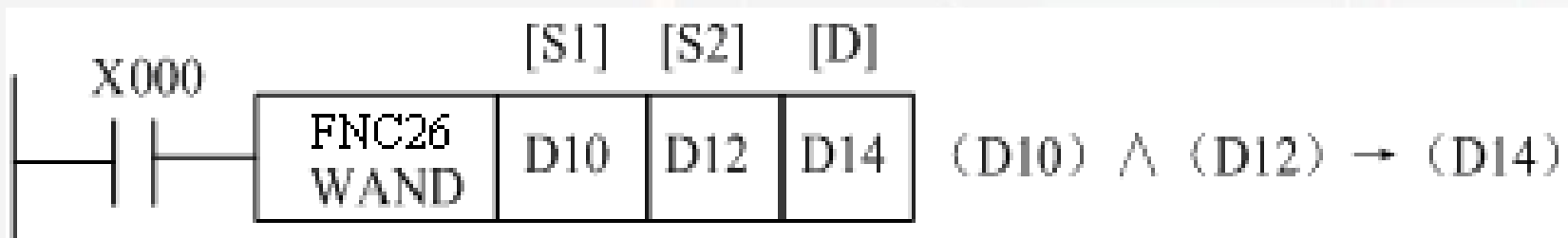


2. 指令用法

➤ (1)WAND指令功能

将指定的两个源软组件[S1]和 [S2]中的数，进行二进制按位“与”，然后将相“与”结果送入指定的目标软组件中。

WAND指令举例



- ✓ 存放在源元件 (D10) 和 (D12) 中的两个二进制数据，以位为单位作逻辑“与”运算，结果存放目标元件[D]，即 (D14) 中

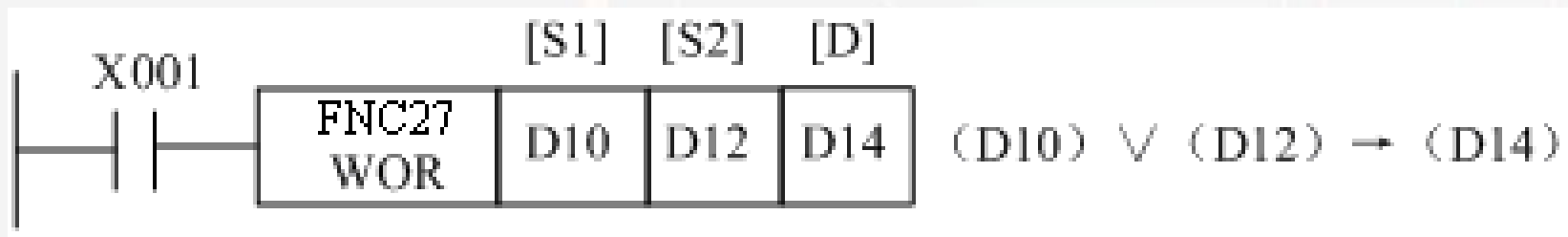


➤ (2) WOR指令功能

将指定的两个源软组件[S1]和 [S2]中的数，进行二进制按位“或”，然后将相“或”结果送入指定的目标软组件中。



WOR指令举例



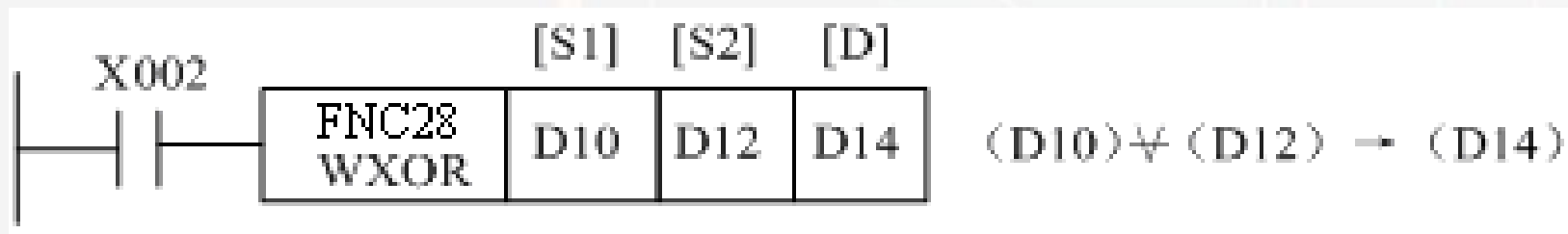
- ✓ 存放在源元件 (D10) 和 (D12) 中的两个二进制数据，以位为单位作逻辑“或”运算，结果存放目标元件[D]，即 (D14) 中



➤ (3) WXOR指令功能

将指定的两个源软组件[S1]和 [S2]中的数，进行二进制按位“异或”，然后将相“异或”结果送入指定的目标软组件中。
指令格式如图5.45

WXOR指令举例



- ✓ 存放在源元件 (D10) 和 (D12) 中的两个二进制数据，以位为单位作逻辑“异或”运算，结果存放 to 目标元件[D]，即 (D14) 中



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

求补指令



1. 指令格式

➤ 指令编号及助记符：求补指令 FNC29 NEG [D·]

其中：✓ [D·]为存放求补结果的目标组件

➤ 目标操作数可取 KnY、KnM、KnS、T、C、D、V和Z

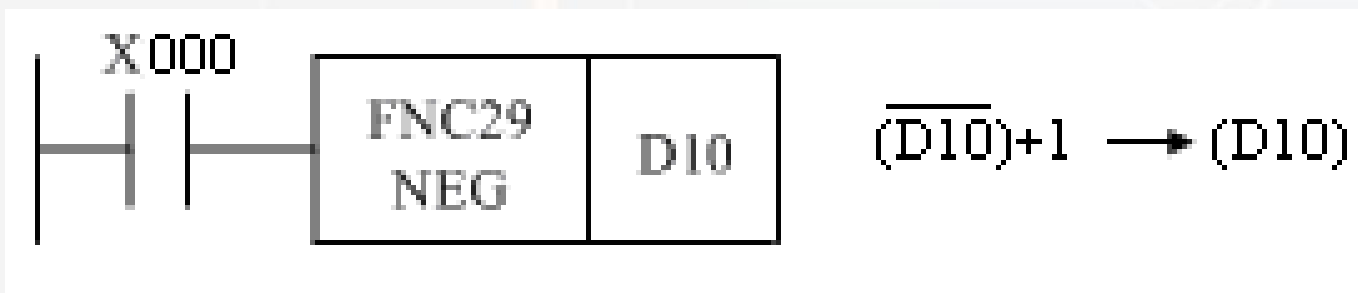


2. 指令用法

➤ NEG指令功能

将目标软组件 [D·] 的内容中的各位先取反 ($0 \rightarrow 1$, $1 \rightarrow 0$) , 然后再加1, 将其结果送入原先的目标软组件中。

求补指令NEG举例



- ✓ X000断开, 则不执行NEG指令, 源、目中的数据均保持不变
- ✓ X000接通, 则执行求补运算, 将D10中的二进制数, 进行“**连同符号位求反加1**”, 结果送入D10中



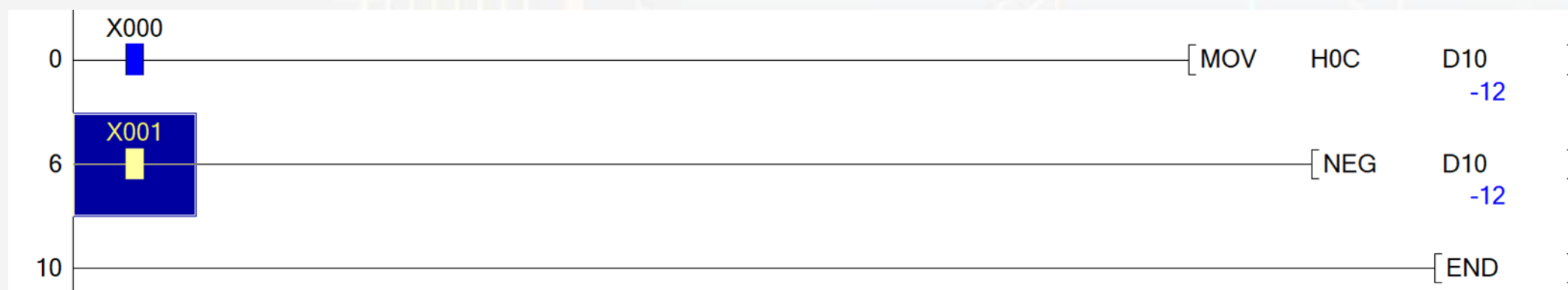
❑ 例如：假设D10中的数为十六进制的H000C，执行这条求补指令。



✓ 对它进行“连同符号位求反加1”，求补结果为HFFF4再存入D10中

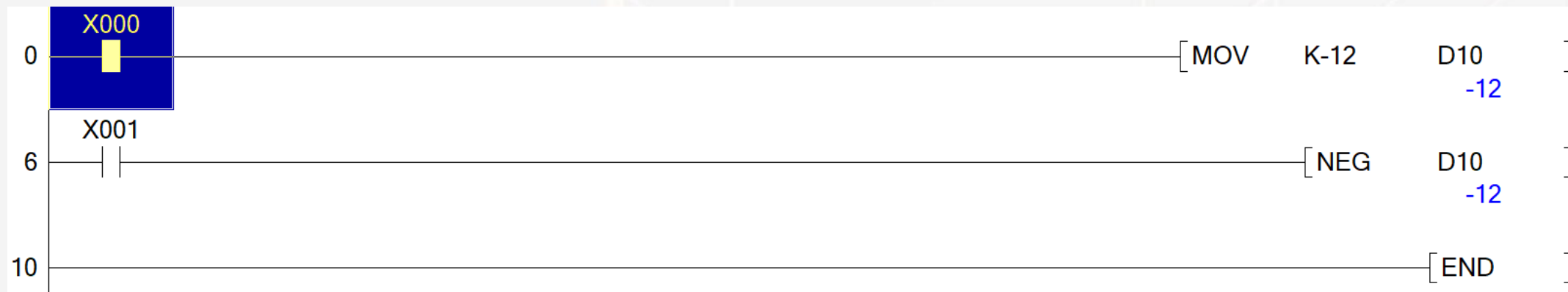


D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D10	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0			12
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0

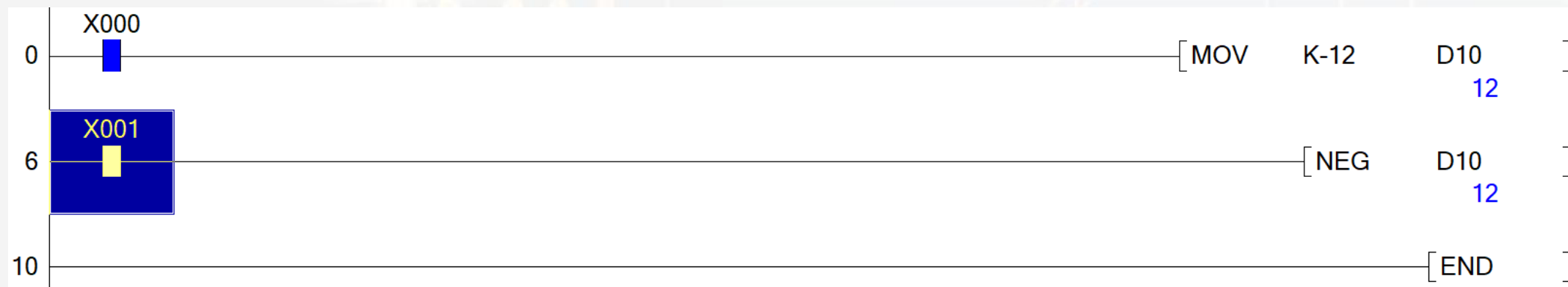


D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D10	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0			-12
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0





D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D10	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0	0	-12
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D10	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	12
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0





➤ 求补与求补码的区别

- ✓ 求补码的规则是：“符号位不变，数值位求反加1”，对800CH(-12)求补码结果将是FFF4H，两者的结果不一样。??
- ✓ 求补指令是**绝对值不变的变号运算**，求补前的H000C的真值是十进制 + 12，而求补后HFFF4的真值是十进制-12



- 求补指令可以有32位操作方式，使用前缀 (D)
- 求补指令也可以有脉冲操作方式。使用后缀 (P)，只有在驱动条件由OFF→ON时进行一次求补运算
- 求补指令的32位脉冲操作格式为(D)NEG(P) [D·]。同样， [D·]为目软组件的首地址
- 求补指令**一般使用其脉冲执行方式**，否则每个扫描周期都将执行一次求补操作





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

5.5 循环与移位指令



武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

左、右循环指令



1. 指令格式

指令编号及助记符:

- (1) 循环右移指令 FNC30 ROR [D·]n

其中: ✓ [D·] 为要移位目软组件

✓ n 为每次移动的位数

- (2) 循环左移指令 FNC31 ROL [D·]n

其中: ✓ [D·] 为要移位目软组件

✓ n 为每次移动的位数

- 目标操作数可取 KnY、KnM、KnS、T、C、D、V 和 Z。移动位数 n 为 K 和 H 指定的常数



2. 指令用法

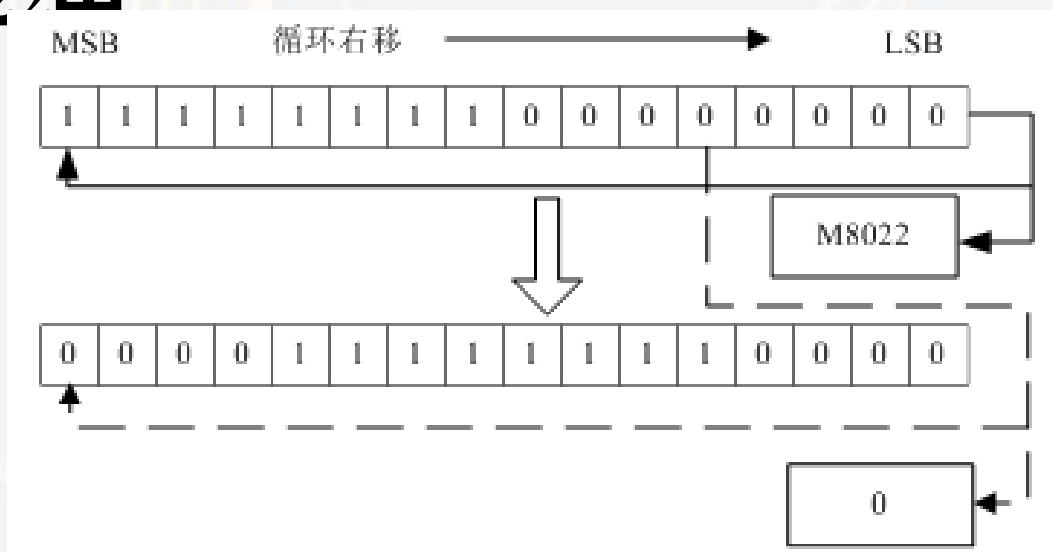
- (1) 循环右移指令ROR的功能是将指定的目标软组件中的二进制数按照指令中n规定的移动的位数由高位向低位移动，最后移出的那一位将进入进位标志位M8022。循环右移指令梯形图格式所示

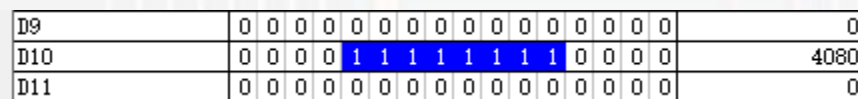


ROR指令举例

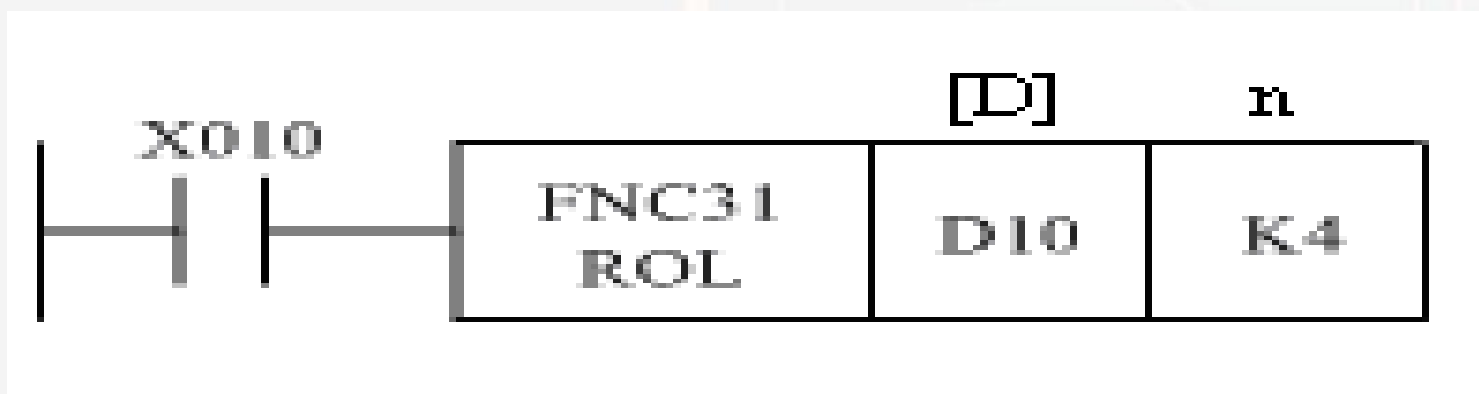
- ✓ 执行一次ROR指令，“n”位的状态向量向右移一次，最右端的“n”位状态循环移位到最左端“n”处，特殊辅助继电器M8022表示最右端的“n”位中向右移出的最后一位的状态

- ❑ 假设D10中的数据为HFF00，执行循环右移指令如图。
- ✓ 指令中K4指示每次循环右移4位，最低4位被移出，并循环回补进入高4位中。
- ✓ 循环右移4位D10中的内容将变为H0FF0了。最后移出的是第3位的“0”，除回补进入最高位外，同时进入进位标志M8022由

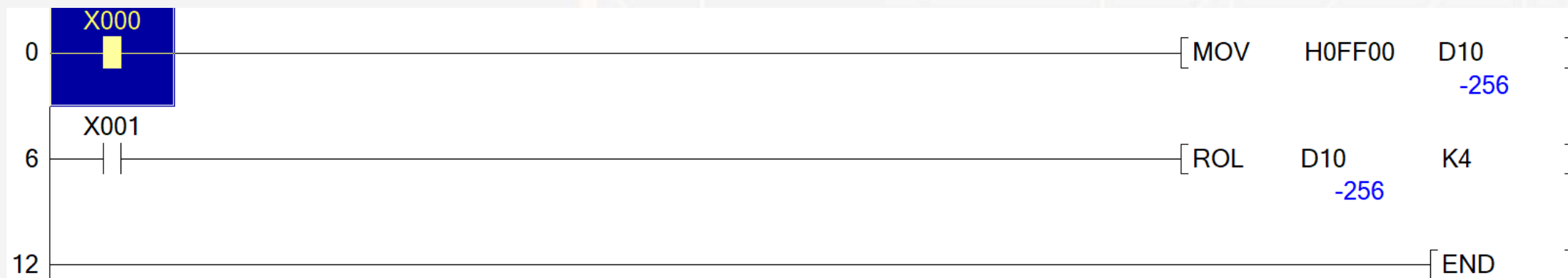


[illegible]

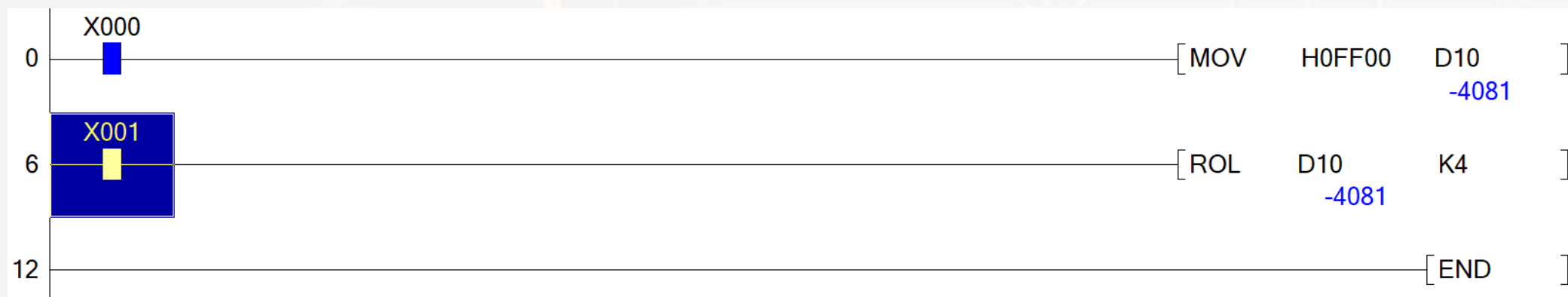
- (2) ROL指令功能是将指定的目标软组件中的二进制数按照指令规定的每次移动的位数由低位向高位移动，最后移出的那一位将进入进位标志位M8022。



- ✓ 循环左移指令梯形图格式如图。ROL指令的执行类似于ROR，只是移位方向相反。



D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D10	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



D9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D10	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	-4081
D11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

软元件	9	8	7	6	5	4	3	2	1	0	
M8022	0	0	0	0	0	0	0	0	0	1	





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

带进位的左、右循环指令



1. 指令格式

指令编号及助记符:

- 1) 带进位的循环右移指令 FNC32 RCR [D·]n
- 2) 带进位的循环左移指令 FNC33 RCL [D·]n

其中: ✓ [D·]为要移位目标组件

✓ n为每次移动的位数

- 目标操作数可取 KnY、KnM、KnS、T、C、D、V 和 Z
- 移动位数 n 为 K 和 H 指定的常数





2. 指令用法

- **RCR指令功能是将指定的目标软组件中的二进制数按照指令规定的每次移动的位数由高位向低位移动，最低位移动到进位标志位M8022。M8022中的内容则移动到最高位**
- **RCR和ROR执行动作基本相同，只是在RCR时，标志位M8022不再表示向右移出的最后一位的状态，而是作为循环移位单元中的一位处理**





- **RCL指令功能是将指定的目标软组件中的二进制数按照指令规定的每次移动的位数由低位向高位移动，最高位移动到进位标志位M8022。M8022中的内容则移动到最低位。**
- **RCL和ROL执行动作基本相同，只是在RCL时，标志位M8022不再表示向左移出的最后一位的状态，而是作为循环移位单元中的一位处理**





案例：彩灯循环控制

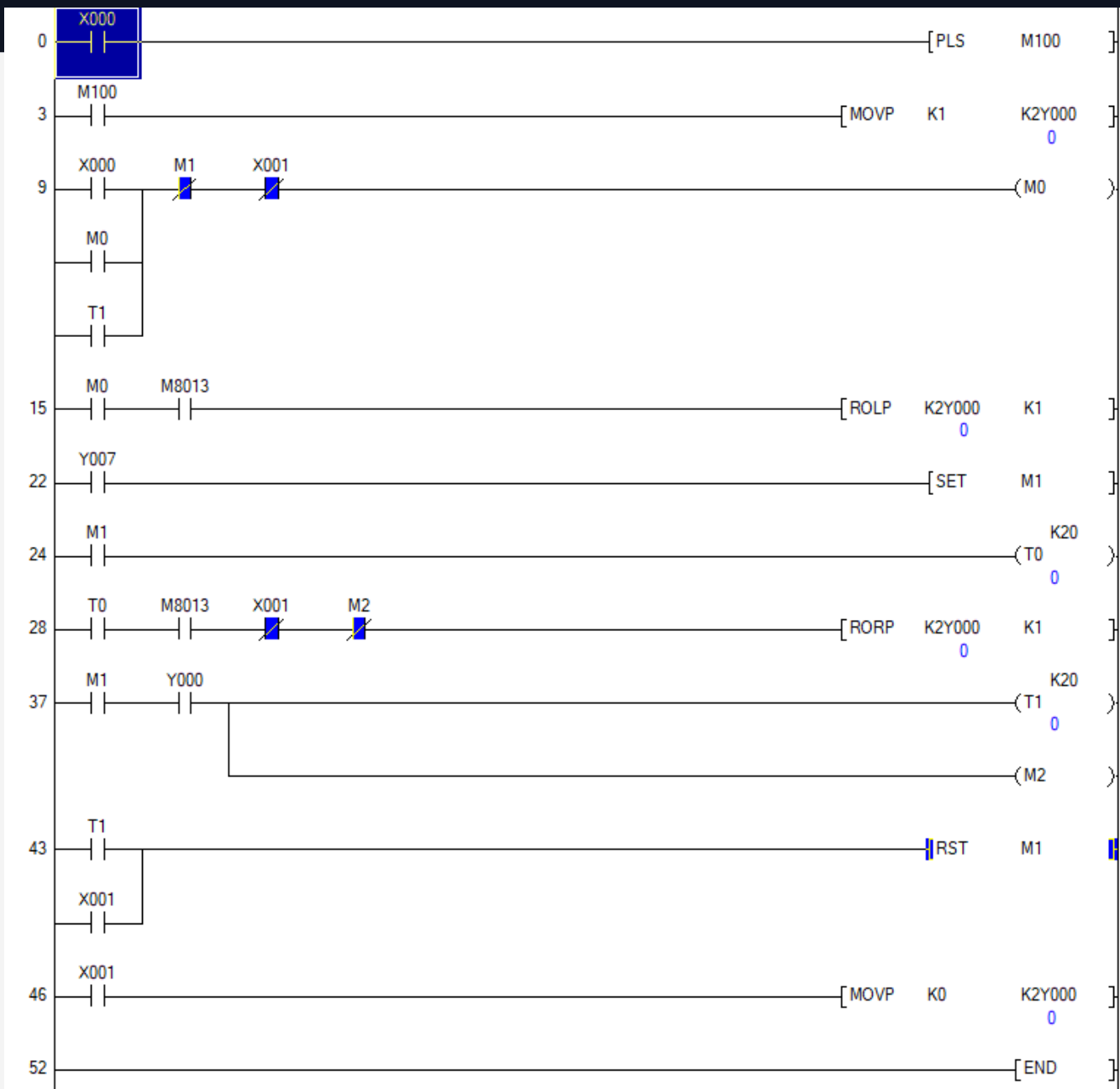
□ 控制要求：

- ✓ 有L1~L8八个灯接于K2Y000，要求当X000为ON时，灯以正序（左移）每隔1s依次点亮，当Y007亮后，停2s；
- ✓ 然后以反序（右移）每隔1s依次点亮，当Y000亮后，停2s，重复这一过程。当X001为ON时，停止工作。

➤ 软元件作用：

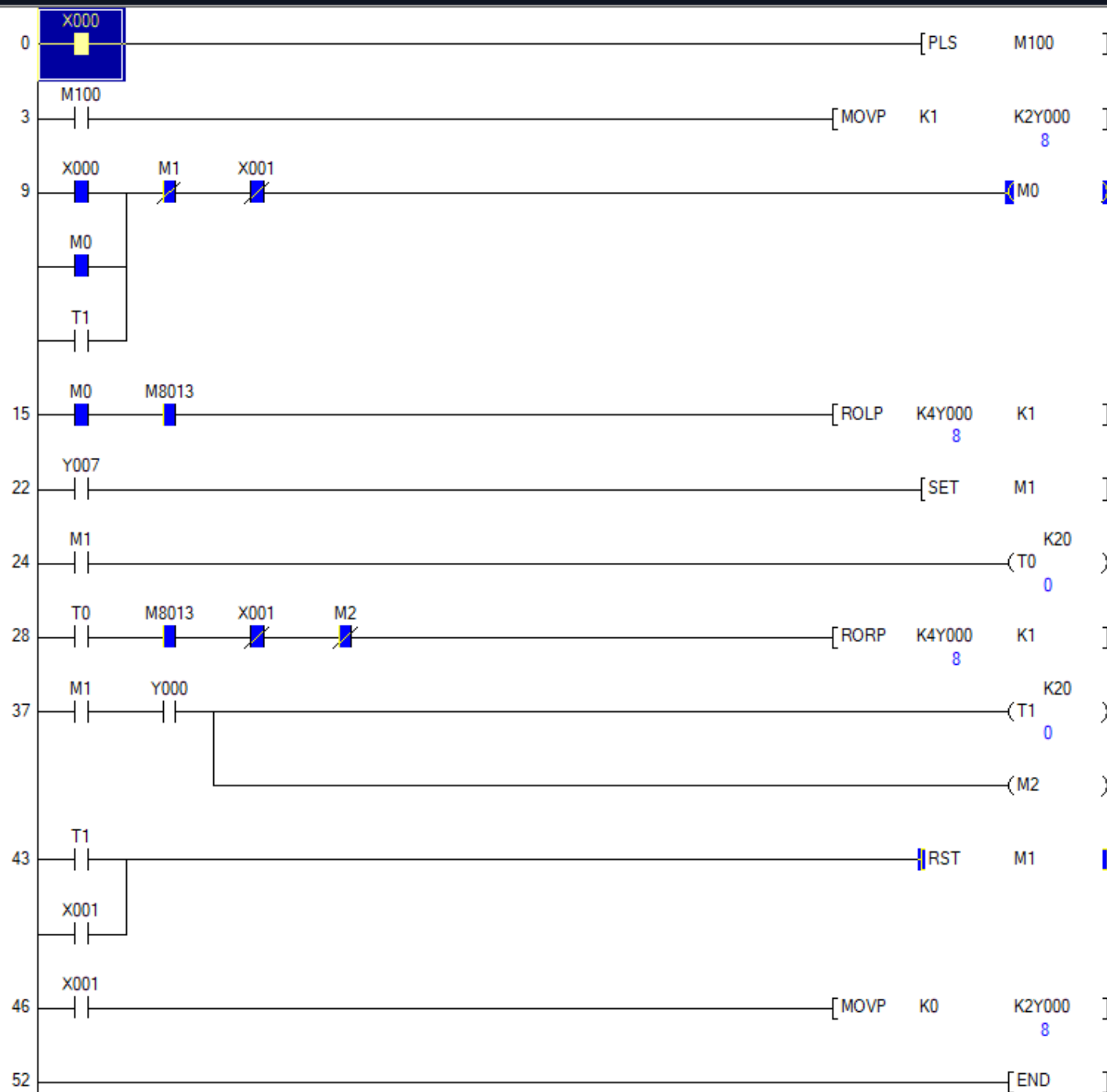
- ✓ 输入SBQ-X000， SBT-X001
- ✓ 输出L1-L8 Y000-Y007。
- ✓ 辅助M0 –正序， M1正转反， M2反转正，
- ✓ M100置初值Y000=ON，
- ✓ M8013秒脉冲。





M8013:1s定时







武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

位组件左移、位组件右移指令



1. 指令格式 指令编号及助记符:

- 位组件右移指令 FNC34 SFTR [S·][D·] n1 n2
- 位组件左移指令 FNC35 SFTL [S·][D·] n1 n2

其中: ✓ [S·]为移位的源位组件首地址

✓ [D·]为移位的目位组件首地址

✓ n1为目位组件个数

✓ n2为源位组件移位个数

➤ 源操作数是Y、X、M、S

➤ 目操作数为Y、M、S

➤ n1和n2为常数K和H





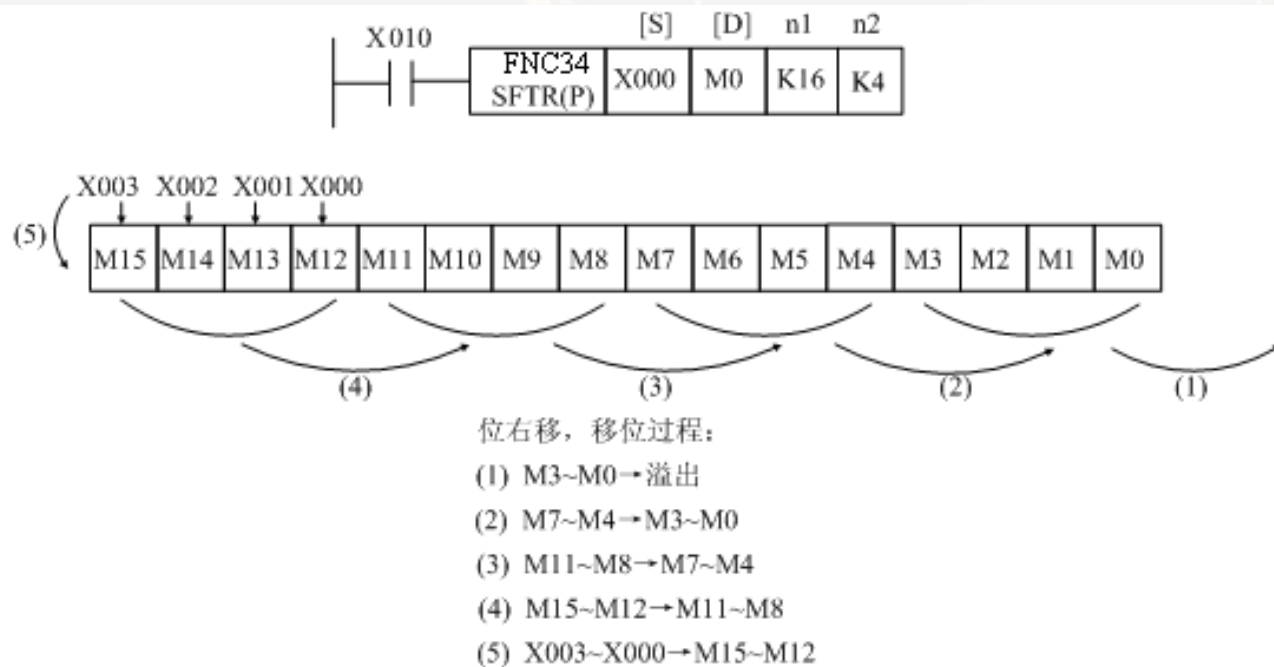
2. 指令用法

➤ 位右移

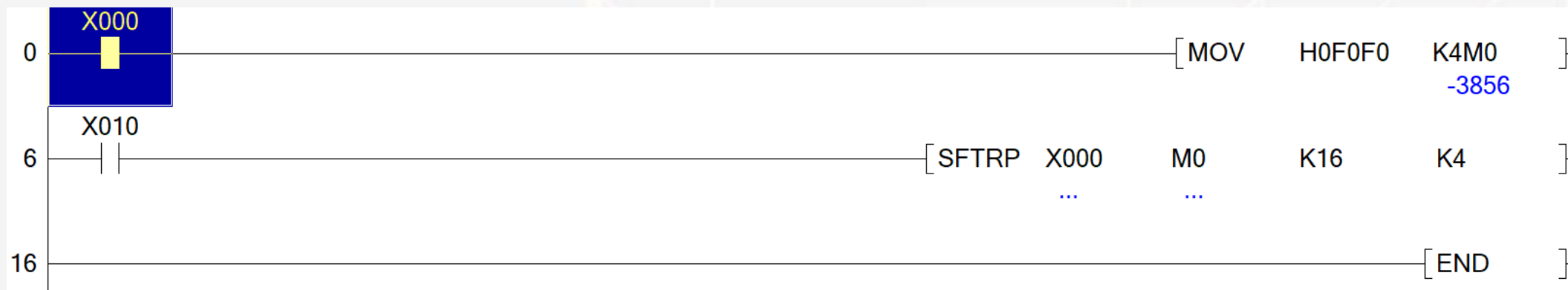
- ✓ 源位组件的低位将从目的高位移入，
- ✓ 目位组件向右移 n_2 位，源位组件中的数据保持不变。
- ✓ 位右移指令执行后， n_2 个源位组件中的数被传送到了目的高 n_2 位中，目位组件中的低 n_2 位数从其低端溢出



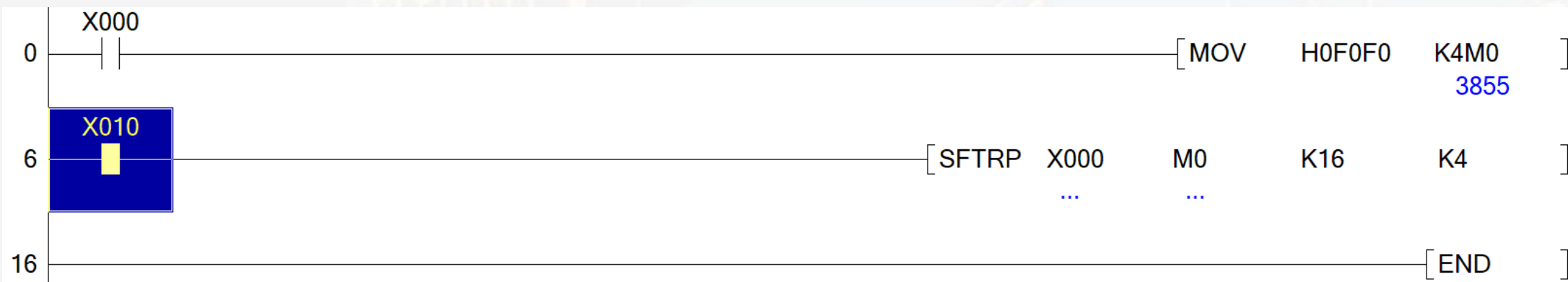
SFTR指令举例



- ✓ 如果X010断开，则不执行这条SFTR指令，源、目中的数据均保持不变。
- ✓ 如果X010接通，则将执行位组件的右移操作，即源中的4位数据X003~X000将被传送到目位组件中的M15~M12。目位组件中的16位数据M15~M0将右移4位，M3~M0等4位数据从目的低位端移出，所以M3~M0种原来的数据将丢失，但源中X003~X000的数据保持不变



软元件	9	8	7	6	5	4	3	2	1	0	▲
M0	0	0	1	1	1	1	0	0	0	0	
M10	0	0	0	0	1	1	1	1	0	0	
M20	0	0	0	0	0	0	0	0	0	0	



软元件	9	8	7	6	5	4	3	2	1	0	▲
M0	1	1	0	0	0	0	1	1	1	1	
M10	0	0	0	0	0	0	0	0	1	1	
M20	0	0	0	0	0	0	0	0	0	0	



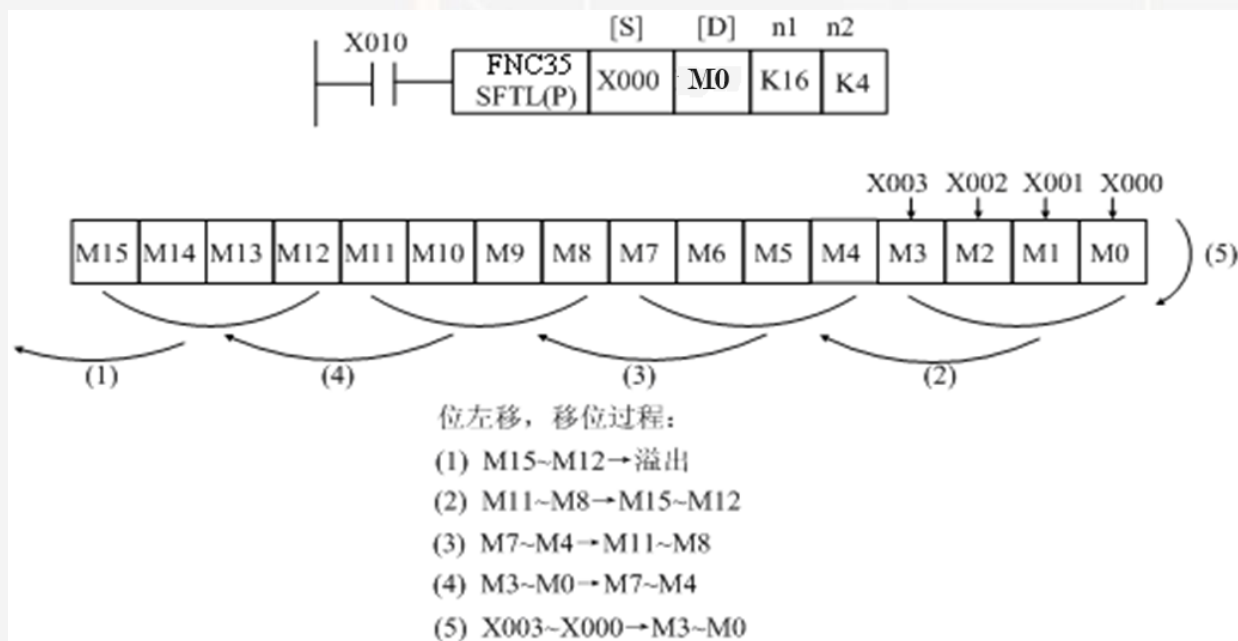


➤ 位左移

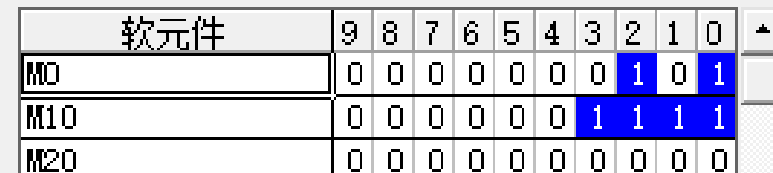
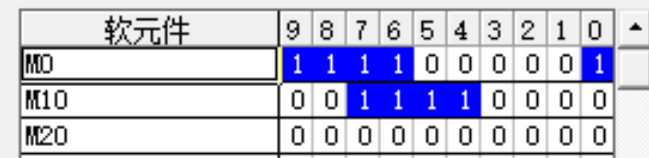
- ✓ 源位组件的高位将从目的低位移入，
- ✓ 目位组件向左移 n_2 位，源位组件中的数据保持不变。
- ✓ 位左移指令执行后， n_2 个源位组件中的数被传送到了目的低 n_2 位中，目位组件中的高 n_2 位数从其高端溢出。



SFTL指令举例



- ✓ 如果X010断开，则不执行这条SFTL指令，源、目中的数据均保持不变。
- ✓ 如果X010接通，则执行位组件左移操作，源中的4位数据X003~X000将被传送到目位组件中的M0~M3。目位组件中的16位数据M15~M0将左移4位，M15~M12等4位数据从目的高位端移出，所以M15~M12原来的数据将丢失，但源中X003~X000的数据保持不变。





武汉理工大学
WUHAN UNIVERSITY OF TECHNOLOGY

字元件右移、字元件左移指令



1. 指令格式 指令编号及助记符:

- 字右移指令 FNC36 WSFR(P) [S·][D·] n1 n2
- 字左移指令 FNC37 WSFL(P) [S·][D·] n1 n2

其中: ✓ [S·]为移位的源字元件首地址

✓ [D·]为移位的目字元件首地址

✓ n1为目字元件个数

✓ n2为源字元件移位个数

➤ 源操作数可取KnX、KnY、KnM、KnS、T、C和D

➤ 目标操作数可取KnY、KnM、KnS、T、C和D

➤ n1, n2 可取K, H





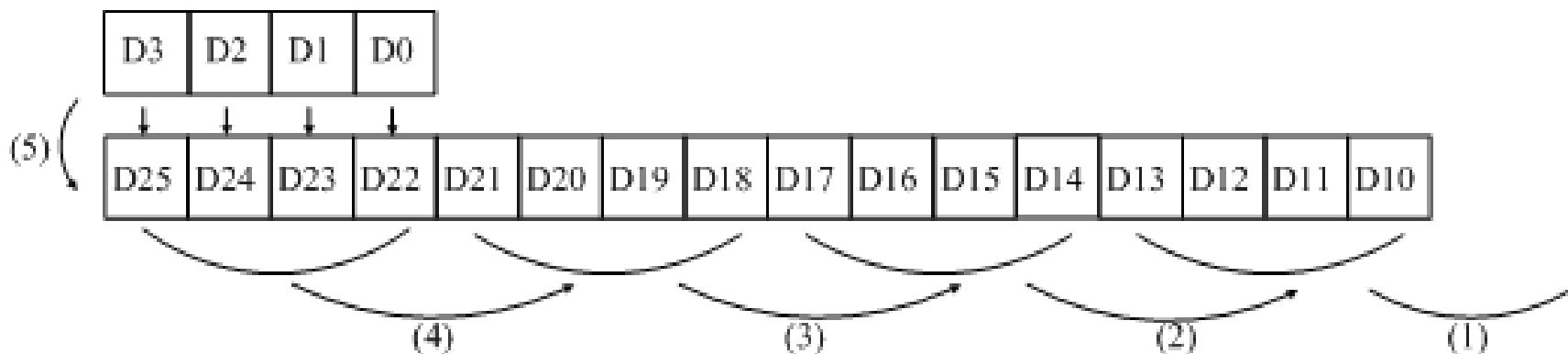
2. 指令用法

- 字元件右移和字元件左移指令以字为单位，其工作的过程与位移位相似，是将 $n1$ 个字右移或左移 $n2$ 个字。
- 使用字右移和字左移指令时应注意：
 - ✓ (1) 字移位指令只有16位操作，占用 9 个程序步
 - ✓ (2) $n1$ 和 $n2$ 的关系为 $n2 \leq n1 \leq 512$



➤ 字元件右移

WSFR指令举例

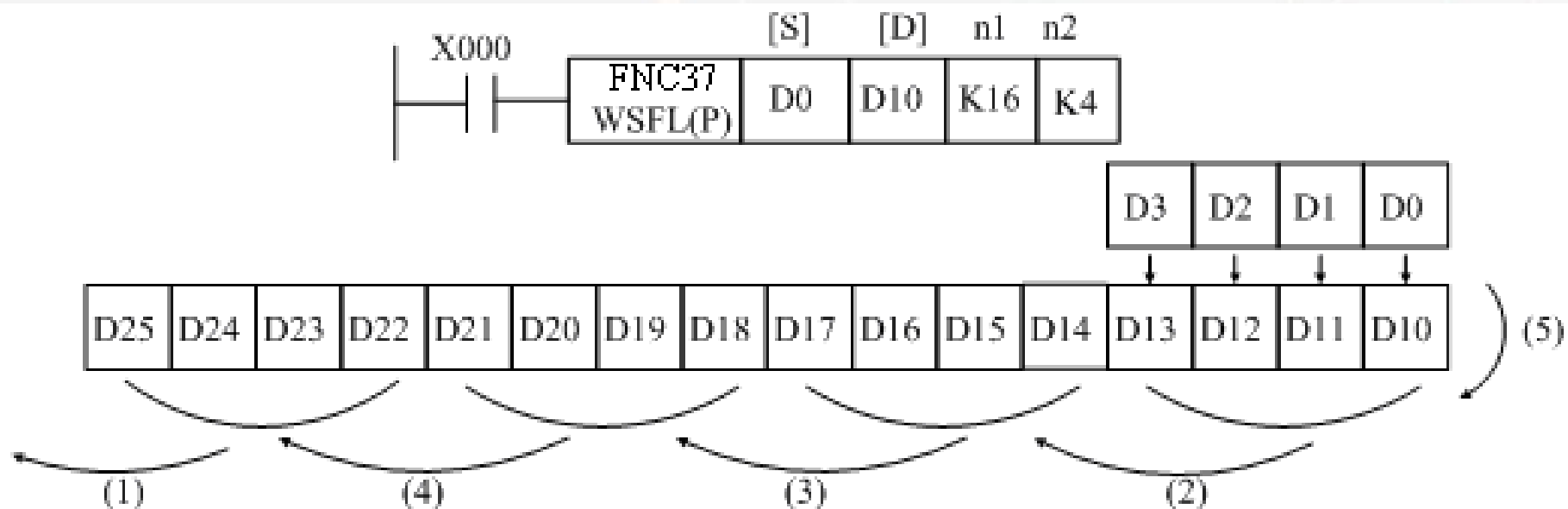


字右移，移位过程：

- (1) D13~D10→溢出
- (2) D17~D14→D13~D10
- (3) D21~D18→D17~D14
- (4) D25~D22→D21~D18
- (5) D3~D0→D25~D22

➤ 字元件左移

WSFL指令举例



字左移，移位过程：

- (1) D25~D22→溢出
- (2) D21~D28→D25~D22
- (3) D17~D14→D21~D18
- (4) D13~D10→D17~D14
- (5) D3~D0→D13~D10



武汉理工大学

WUHAN UNIVERSITY OF TECHNOLOGY

谢谢观看