北京航空航天大學
BEIHANG UNIVERSITY


微机原理与接口技术
第三章

林 新
Lx@buaa.edu.cn

北京航空航天大学 自动化学院

北京航空航天大学自动化学院

第三章 8086寻址方式和指令系统

主要内容：

- 寻址方式
- 指令系统

北京航空航天大学自动化学院林新

基本要求

- 所有寻址方式
- 常用指令：
  - 数据传送指令：
    掌握MOV,PUSH,POP,XCHG,XLAT,IN/OUT,LEA
  - 算术运算指令：
    掌握ADD,ADC,INC, SUB,SBB,DEC,NEG,CMP;
    了解AAA,DAA, AAS,DAS,MUL/IMUL,DIV/IDIV
  - 逻辑运算和移位指令：全部
  - 字符串处理指令：
    了解MOVS,CMPS,SCAS,LODS,STOS
  - 控制转移指令：
    掌握JMP,CALL/RET,LOOP，JZ/JA等条件转移，了解LOOPE/LOOPZ,LOOPNE/LOOPNZ

3.1 8086寻址方式

8086指令格式

操作码　　操作数, 操作数　；注释

- 指令一般分为操作码和操作数两部分
  - **必须含有操作码**
  - **根据指令的不同，操作数可能有0个、1个或者2个**
  - **分号;后面为注释，可有可无，对指令无影响**

北京航空航天大学自动化学院林新

寻址方式

操作码　操作数,操作数　；注释

目的操作数　　源操作数

- 寻址方式：执行指令时定位操作数位置的方法
- 执行指令时，在计算机系统的三个地方寻找操作数
  - 寄存器（操作数就是CPU内部的寄存器）
  - 指令队列（这种情况下操作数包含在指令中，随指令一起被取出并放在BIU的指令队列中）
  - CPU外部的存储器（或者IO接口电路）

8086的8种寻址方式

- 立即寻址方式
- 寄存器寻址方式
- 直接寻址方式
- 寄存器间接寻址方式
- 寄存器相对寻址方式
- 基址变址寻址方式
- 相对基址变址寻址方式
- 其它

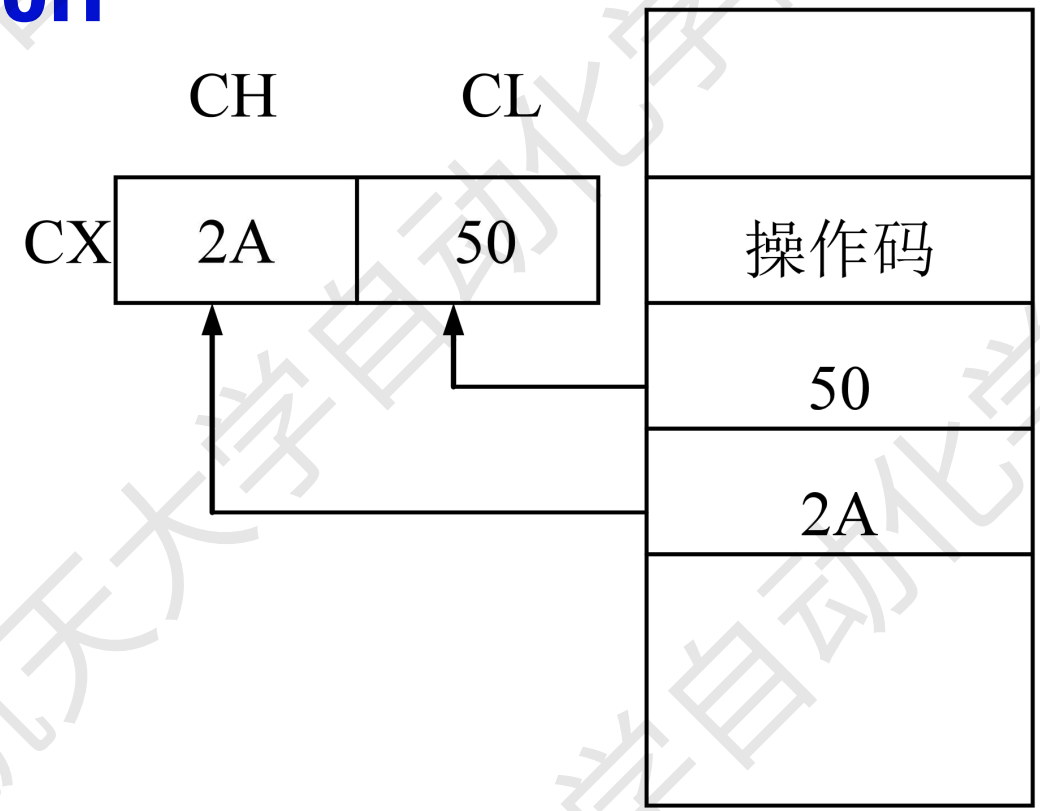
北京航空航天大学自动化学院林新

1）立即寻址方式

- 操作数直接包含在指令中，是一个8位或者16位的常数，也叫做**立即数**。这类指令翻译成机器码时，立即数作为指令的一部分，紧跟在操作码之后，存放在代码段内。
  - 数值
  - 常量

北京航空航天大学自动化学院林新

MOV CX, **2A50H**



2）寄存器寻址方式

- 操作数在寄存器中，由指令指定寄存器的名称
- 16位的操作数放在AX,BX,CX等寄存器中
- 8位的操作数放在AL,AH,BL,BH,CL,CH,DL,DH中

北京航空航天大学自动化学院林新

3）直接寻址方式

- 在直接寻址方式下，计算机的有效地址由指令给出。在它们的机器码中，有效地址放在代码段中指令的操作码之后。
  - 8086系统中操作数的偏移地址称为**有效地址**
    EA(Effective Address)
- 为了和立即数相区分，指令中有效地址两侧需要加一个方括号。

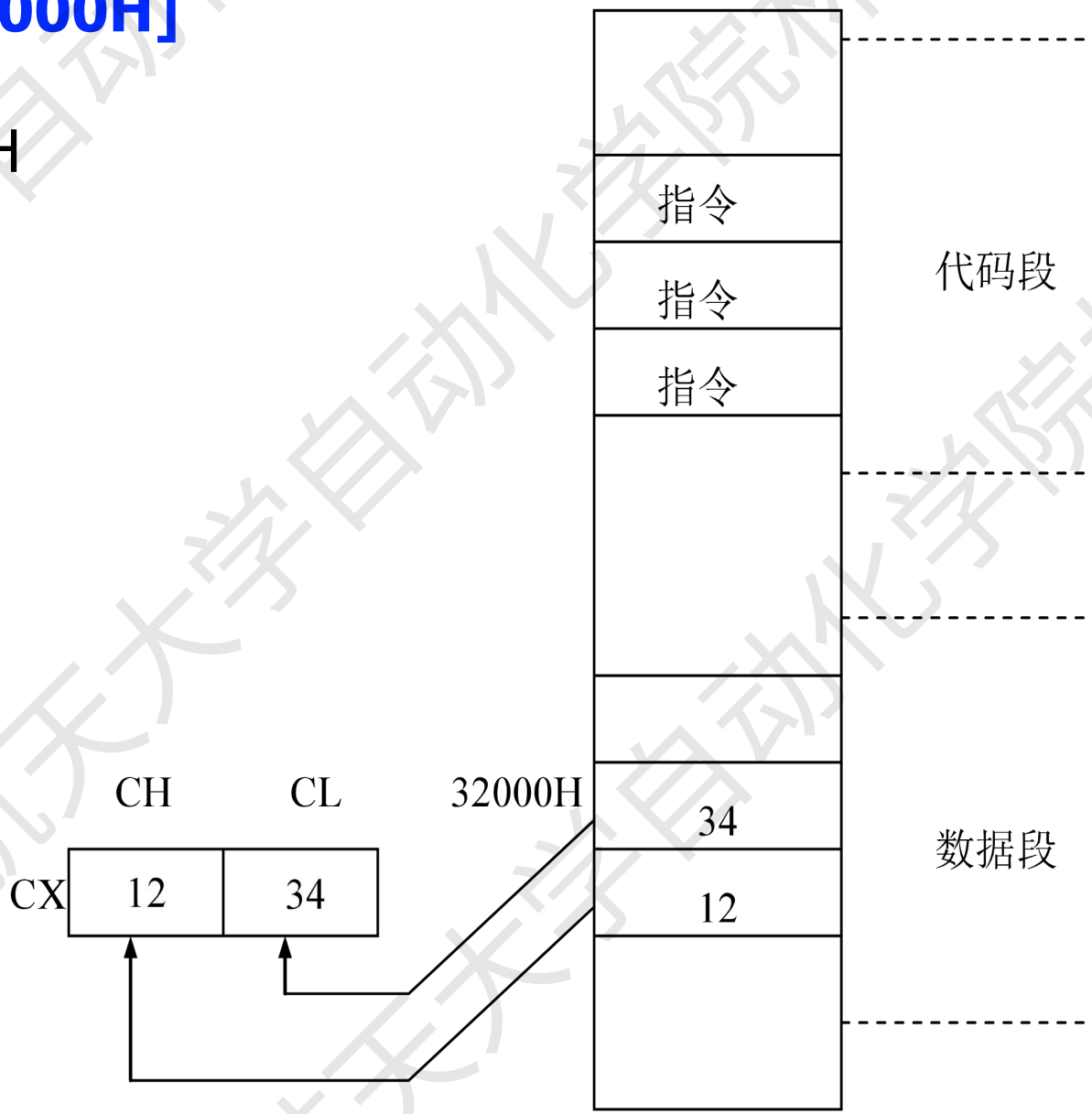
北京航空航天大学自动化学院林新

例：设DS= 3000H，(32000H)=34H，(32001H)=12H
执行指令 **MOV CX, [2000H]**

则 有效地址EA = 2000H

源操作数的物理地址
= 3000H×16 + 2000H
= 32000H

指令执行之后
CX＝1234H



北京航空航天大学自动化学院林新

1）无段超越前缀（默认情况）

2）段超越前缀

3）符号地址

北京航空航天大学自动化学院林新

4）寄存器间接寻址方式

- 这种寻址方式，指令中给出的寄存器中的值不是操作数本身，而是操作数的**有效地址**。
- 寄存器外面必须加**方括号**，以便与寄存器寻址方式相区别。
- 所使用的寄存器：**BX, BP, SI, DI**。

北京航空航天大学自动化学院林新

- 如果指令中指定的是**BX, SI, DI**，则默认与**DS**搭配，操作数存放在数据段中。此时：

  物理地址 = 16×DS + BX

  或 = 16×DS + SI

  或 = 16×DS + DI

- 如果指令中指定的是**BP**，则默认与**SS**搭配，操作数放在堆栈段。此时：

  物理地址 = 16×SS + BP

例3-11　**MOV BX, [SI]**

设：DS = 1000H

SI = 2000H

（12000H） = 318BH

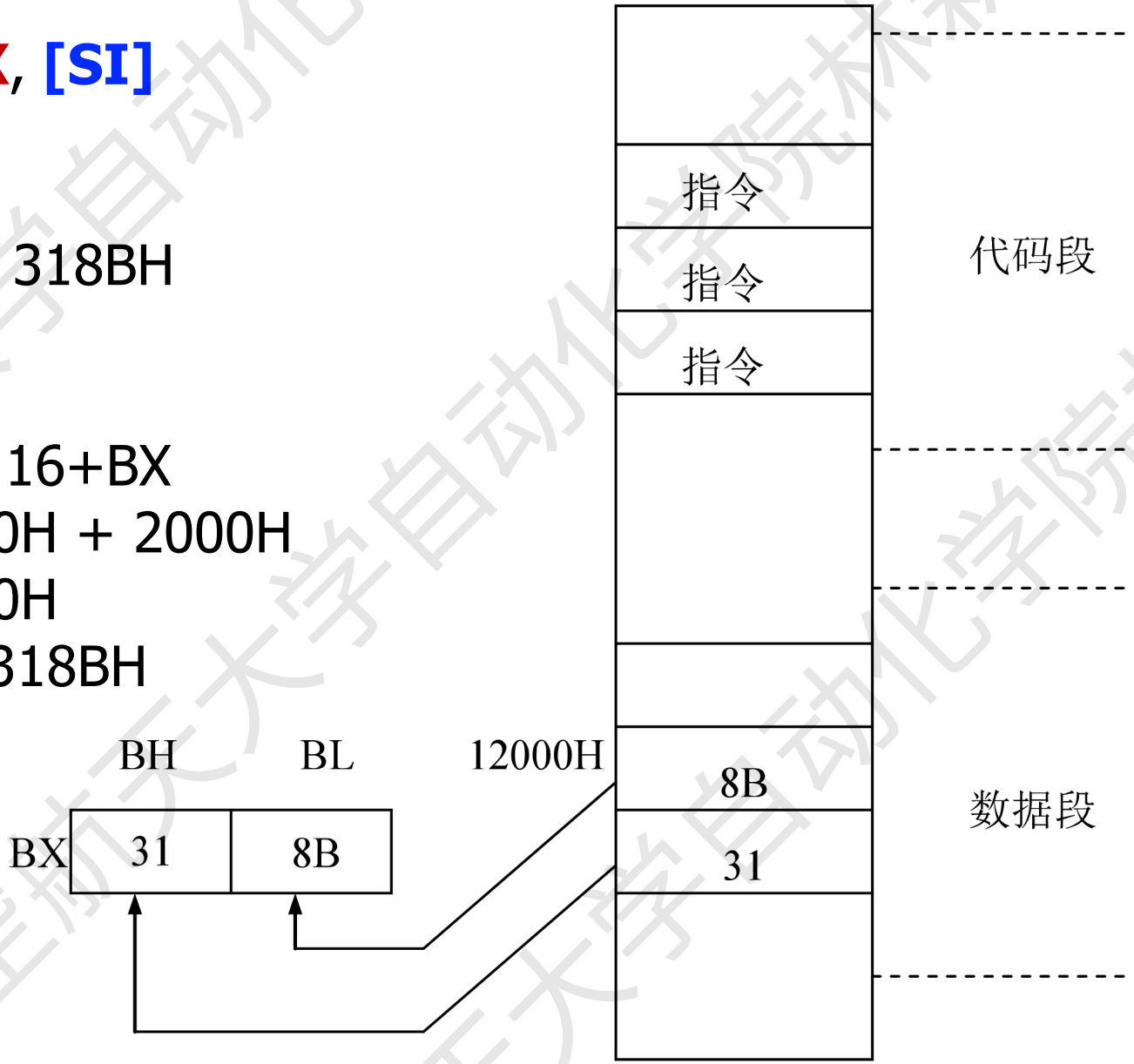
则：

物理地址 = DS×16+BX

= 10000H + 2000H

= 12000H

执行结果：BX = 318BH



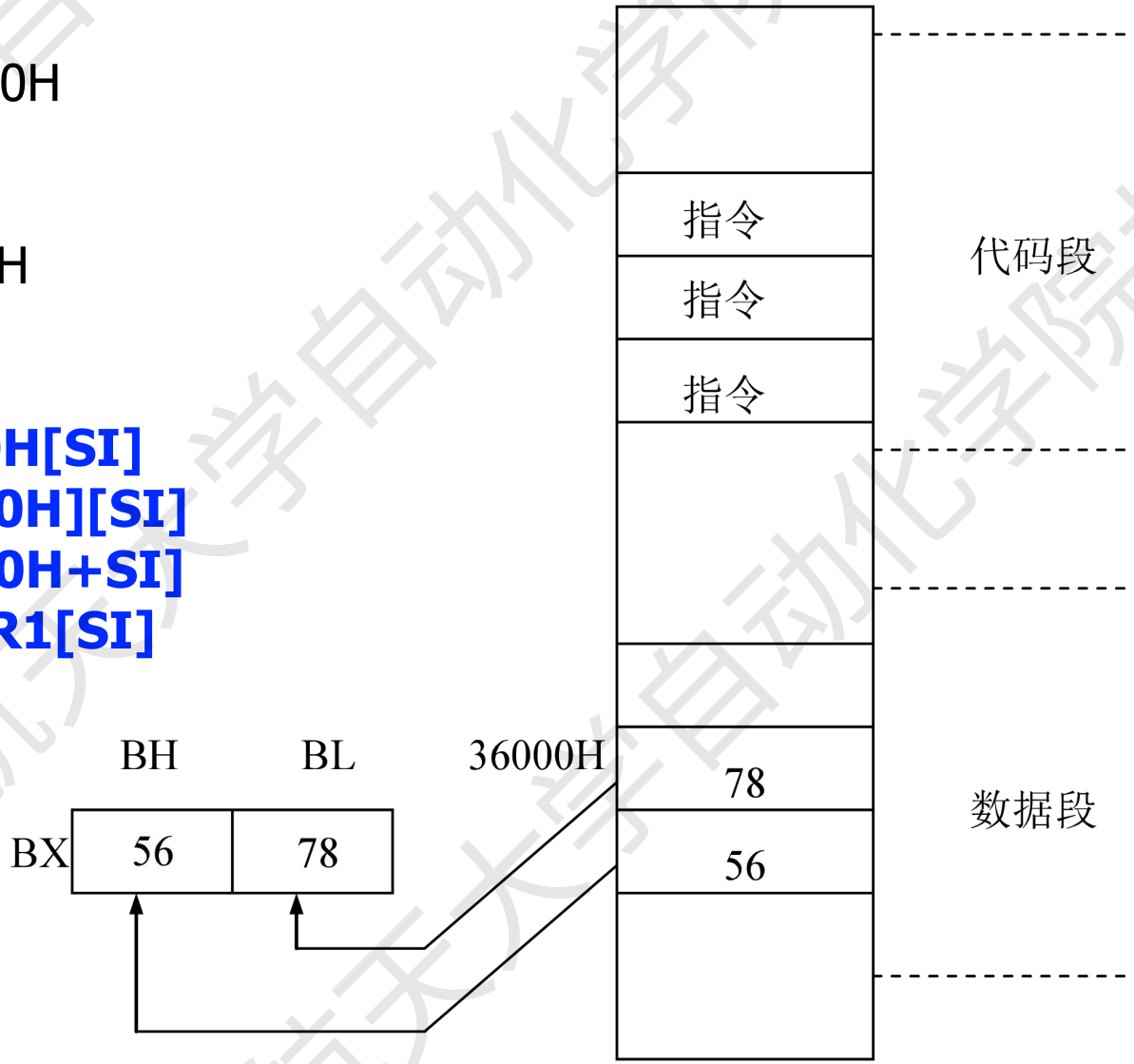
5）寄存器相对寻址方式

- 这种方式下，操作数的**有效地址**是一个**基址或变址寄存器**的内容与**指令中指定的偏移量**之和
- 基址或变址寄存器为BX, SI, DI, BP。其中前三者默认的段寄存器是DS，BP默认的段寄存器是SS
- 这种方式与方式4-寄存器间接寻址类似，不同的是有效地址还要加一个位移量

例3-12 设DS=3000H, SI=2000H,ADDR1 = 4000H, (36000H)=5678H
则：指令**MOV BX, 4000H[SI]** 的源操作数物理地址：
DS*16+4000H+SI
= 3000H*16+4000H+2000H
= 36000H

指令执行结果：BX= 5678H

而且：
指令MOV BX, **4000H[SI]**
等效于 MOV BX, **[4000H][SI]**
等效于 MOV BX, **[4000H+SI]**
等效于 MOV BX, **ADDR1[SI]**



6）基址变址寻址方式

- 操作数的有效地址是一个**基址寄存器**（BX或BP）和一个**变址寄存器**（SI或DI）的内容之和，两个寄存器均由指令指定
- 基址寄存器为**BX**时，默认的段基址寄存器为**DS**
- 基址寄存器为**BP**时，默认的段基址寄存器为**SS**

北京航空航天大学自动化学院林新

物理地址 = 16×**DS**+**BX**+SI

物理地址 = 16×**DS**+**BX**+DI

物理地址 = 16×**SS**+**BP**+SI

物理地址 = 16×**SS**+**BP**+DI

- 段基址可以使用**段超越前缀**修改

北京航空航天大学自动化学院林新

例3-13 **MOV AX, [BX][SI]**

设：DS=3000H,BX=1200H,SI=0500H, [31700H]=ABCDH

则：

源操作数物理地址

=16×DS+BX+SI

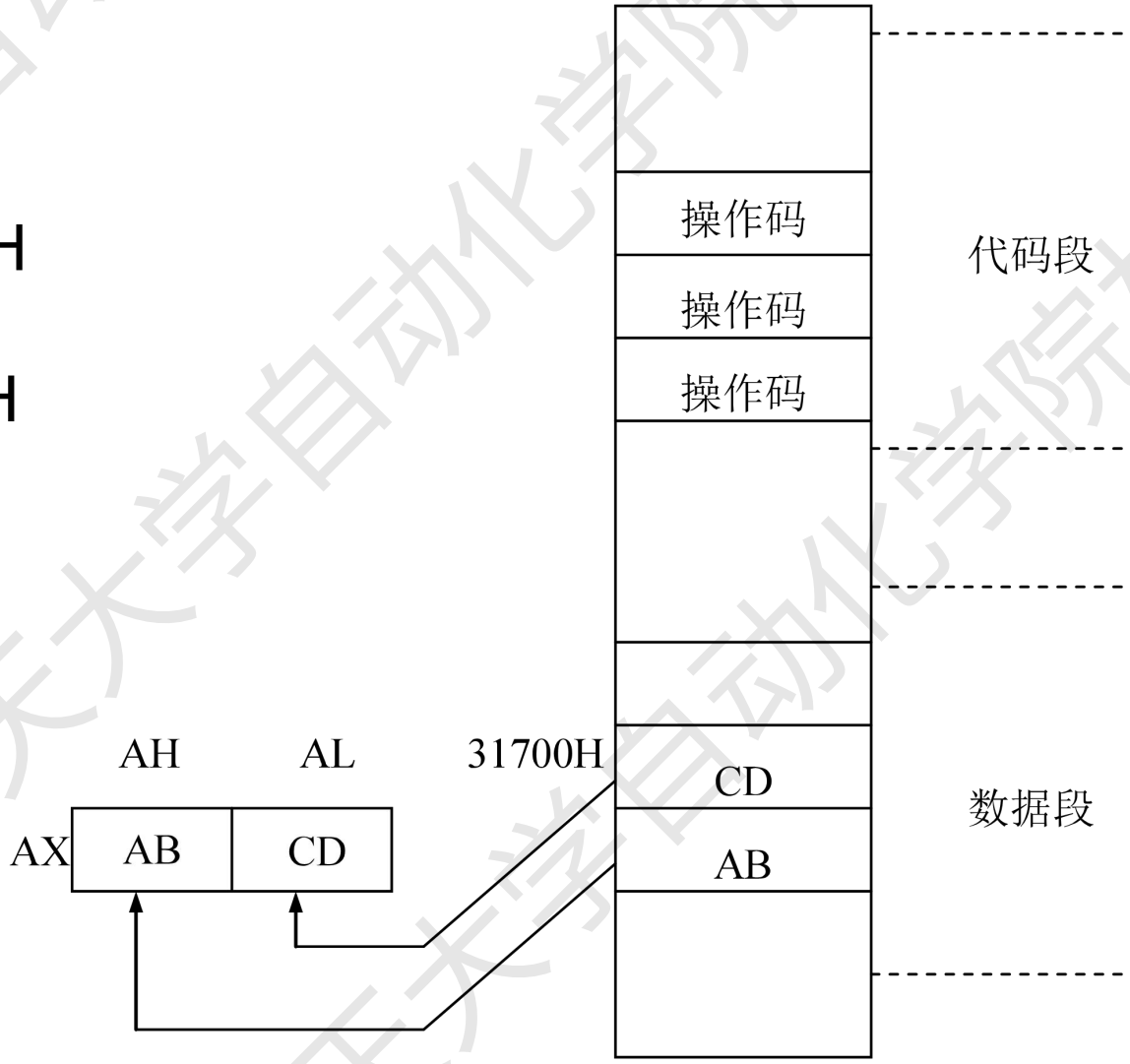
=30000H+1200h+0500H

=31700H

执行结果为：AX=ABCDH

上述指令也可以写成：

MOV AX, [BX+SI]



7）相对基址变址寻址方式

- 操作数的有效地址是一个**基址寄存器**和一个**变址寄存器**的内容，再加上指令中指定的**8位或者16位位移量**之和
- 基址寄存器为**BX**时，默认的段基址寄存器为**DS**
- 基址寄存器为**BP**时，默认的段基址寄存器为**SS**

北京航空航天大学自动化学院林新

物理地址 = 16×**DS**+**BX**+SI+位移量

物理地址 = 16×**DS**+**BX**+DI+位移量

物理地址 = 16×**SS**+**BP**+SI+位移量

物理地址 = 16×**SS**+**BP**+DI+位移量

- 段基址可以使用**段超越前缀**修改。

北京航空航天大学自动化学院林新

例3-14 **MOV AX, MASK[BX][SI]**

设：DS=2000H,BX=1500H,SI=0300H

MASK=0200H,（21A00H）=26BFH

则：

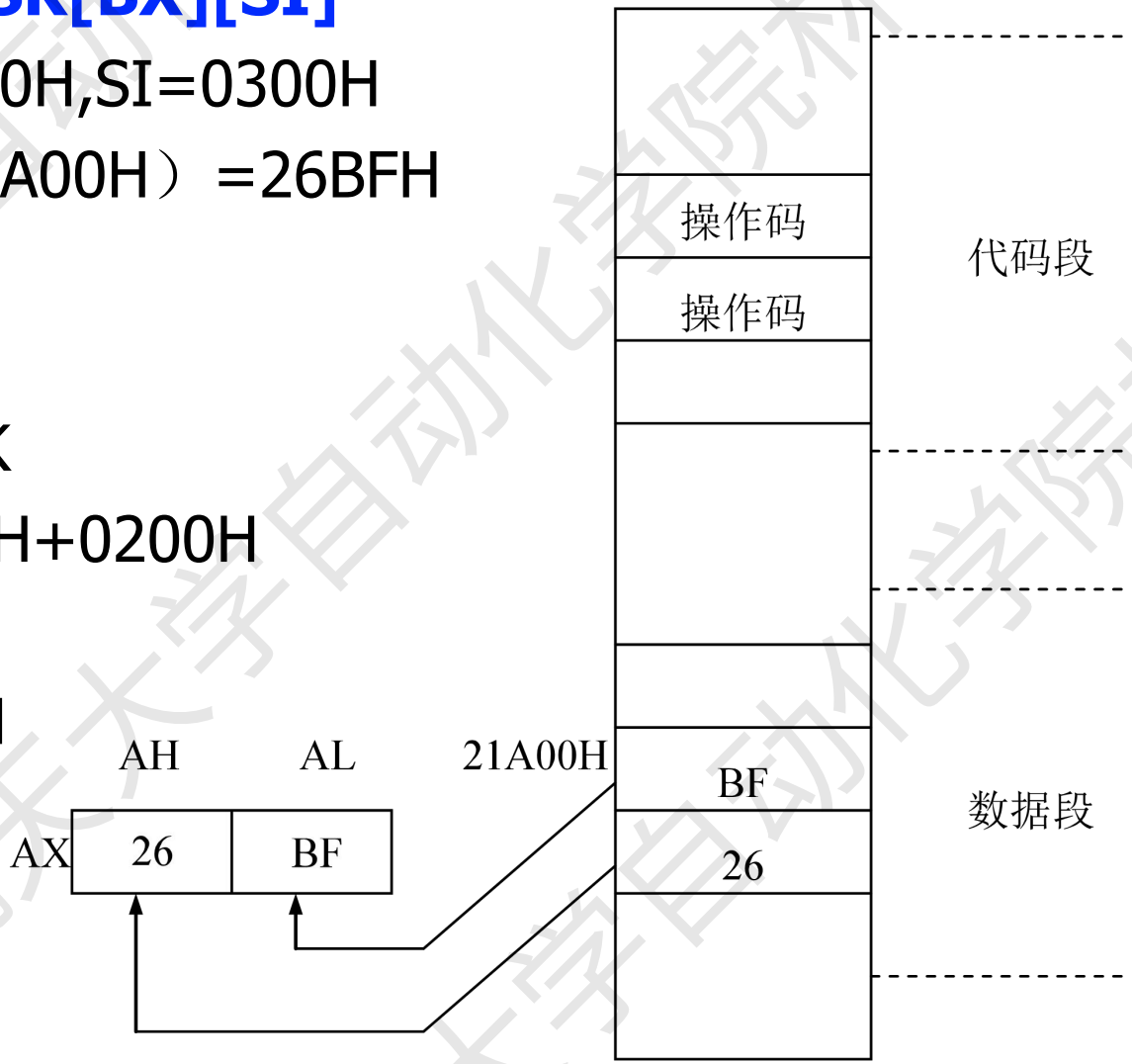
源操作数物理地址

=16×DS+BX+SI+MASK

=20000H+1500h+0300H+0200H

=21A00H

执行结果为：AX=26BFH



带方括号的地址表达式必须遵循下列规则：

1. 立即数可以程序在方括号内，表示直接地址，如[2000H]
2. 只有**BX,BP,SI,DI**可以出现在**[]**内。它们可以单独出现，也可以组合在一起（只能相加），或以寄存器和常数相加的形式出现。BX和BP**不能**同时出现在一个[]内，SI和DI也**不能**同时出现。
3. 方括号有相加的含义，下面几种写法是等价的：

```
MOV   AX, [MASK+BX+SI]
MOV   AX, MASK[BX+SI]
MOV   AX, 200H[BX][SI]
```

4. 若方括号内**包含BP，则默认使用SS**来提供段基址；其余情况默认使用DS来提供段基址。可以使用段超越前缀对默认的段寄存器进行修改。

8）其它

1）隐含寻址

指令中不指定操作数，但有隐含规定的寻址方式，例如DAA。

2）I/O端口寻址

分直接端口寻址和间接端口寻址两种。

3）一条指令中有几种寻址方式

源操作数可以用所有寻址方式指定；

目的操作数可以用除立即数之外的所有寻址方式指定。

4）转移类指令寻址

3.2 8086指令的机器码（了解）

- 计算机不能识别汇编语言程序。需要将汇编语言中的各个指令转变为二进制代码，CPU才能识别。
- 每条汇编语言指令都有其对应的二进制编码，这种二进制编码称为机器码。不同的机器码代表了不同的操作。

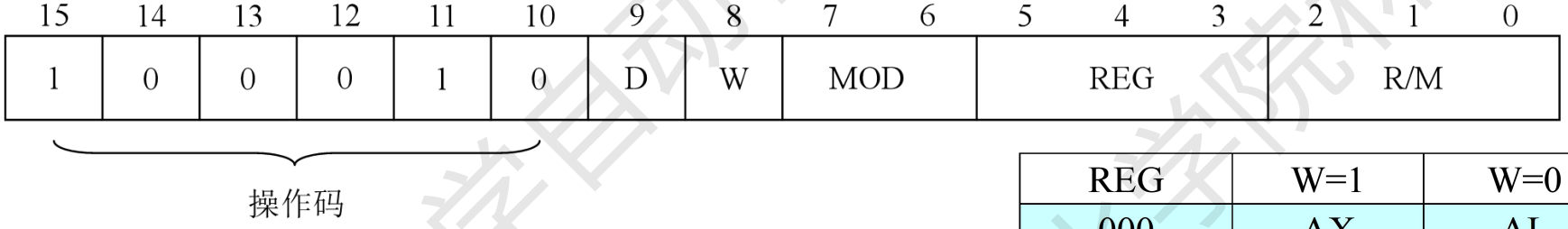
北京航空航天大学自动化学院林新

8086指令编码的特点

- 编码复杂。每种指令类型本身就有多种操作数和寻址方式，对应了多种编码。
- 每种指令可以给出基本的编码格式，对照格式填上不同的数字来表示不同的寻址方式、数据类型等等，就可以得到相应的机器码。
- 8086采用**变长指令**，可以是**1～6**个字节。

北京航空航天大学自动化学院林新

（通用）寄存器之间或寄存器与存储器之间交换数据的MOV指令



REG	W=1	W=0
000	AX	AL
001	CX	CL
010	DX	DL
011	BX	BL
100	SP	AH
101	BP	CH
110	SI	DH
111	DI	BH

MOD / R/M	00	01	10	11	
				W=0	W=1
000	[BX]+[SI]	[BX]+[SI]+D8	[BX]+[SI]+D16	AL	AX
001	[BX]+[DI]	[BX]+[DI]+D8	[BX]+[DI]+D16	CL	CX
010	[BP]+[SI]	[BP]+[SI]+D8	[BP]+[SI]+D16	DL	DX
011	[BP]+[DI]	[BP]+[DI]+D8	[BP]+[DI]+D16	BL	BX
100	[SI]	[SI]+D8	[SI]+D16	AH	SP
101	[DI]	[DI]+D8	[DI]+D16	CH	BP
110	D16(直接地址)	[BP]+D8	[BP]+D16	DH	SI
111	[BX]	[BX]+D8	[BX]+D16	BH	DI

D=0：数据从寄存器传出
D=1：数据传输给寄存器

P57-60

3.3 8086指令系统

8086指令共有六大类：

- 数据传送指令
- 算术运算指令
- 逻辑运算和移位指令指令
- 字符串处理指令
- 控制转移指令
- 处理器控制指令。

北京航空航天大学自动化学院林新

3.3.1 数据传送指令

- 要求掌握：
  - MOV，PUSH，POP，XCHG，XLAT
  - IN/OUT
  - LEA

北京航空航天大学自动化学院林新

通用数据传送指令	
MOV	字节或字的传送
PUSH	入栈指令
POP	出栈指令
XCHG	交换字或字节
XLAT	表转换
输入输出指令	
IN	输入
OUT	输出
地址目标传送指令	
LEA	装入有效地址
LDS	装入数据段寄存器
LES	装入附加段寄存器
标志传送指令	
LAHF	标志寄存器低字节装入 AH
SAHF	AH 内容装人标志寄存器低字节
PUSHF	标志寄存器入栈指令
POPF	出栈，并送入标志寄存器

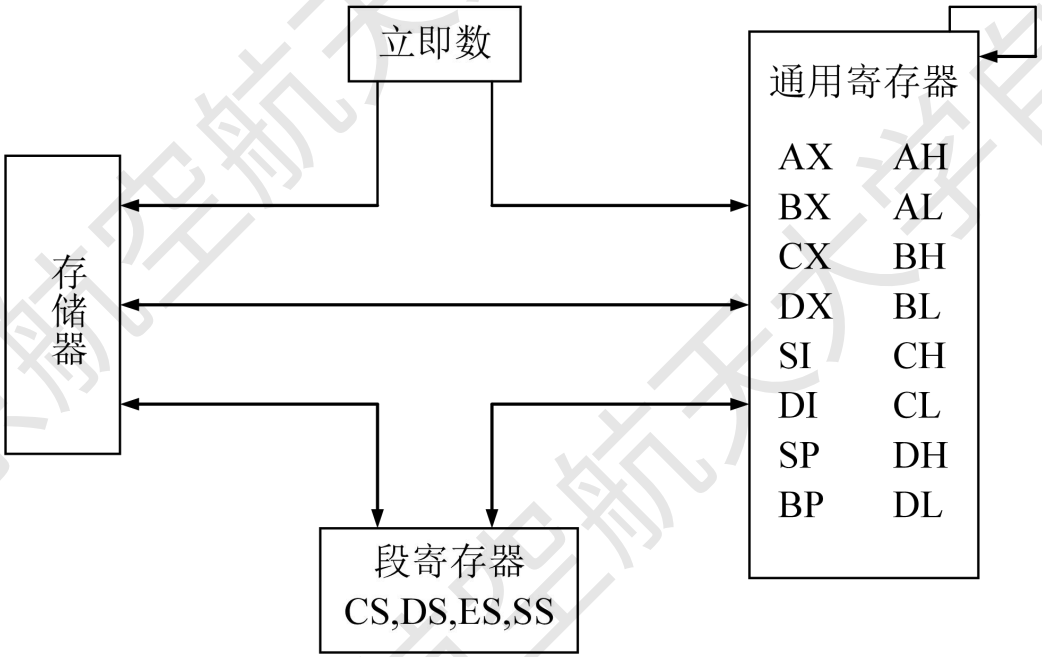
1.通用数据传输指令

1. MOV
2. PUSH
3. POP
4. XCHG
5. XLAT

北京航空航天大学自动化学院林新

（1）MOV传送指令（Move）

- 指令格式：MOV 目的，源
- 指令功能：将源操作数（一个字或一个字节）传送到目的操作数



MOV指令运算传送数据的途径

MOV指令注意事项：

- IP不能用作源操作数或目的操作数
- 目的操作数不允许用立即数和CS寄存器
- 除非是 立即数→存储器，否则操作数中必须有一个是寄存器，但不能都是段寄存器。即：
  - 不能在两个存储单元之间直接传送数据
  - 不能在两个段寄存器之间直接传送数据
- 立即数不能直接送段寄存器

数据段中定义的变量，其偏移地址默认从0开始

逻辑段名称

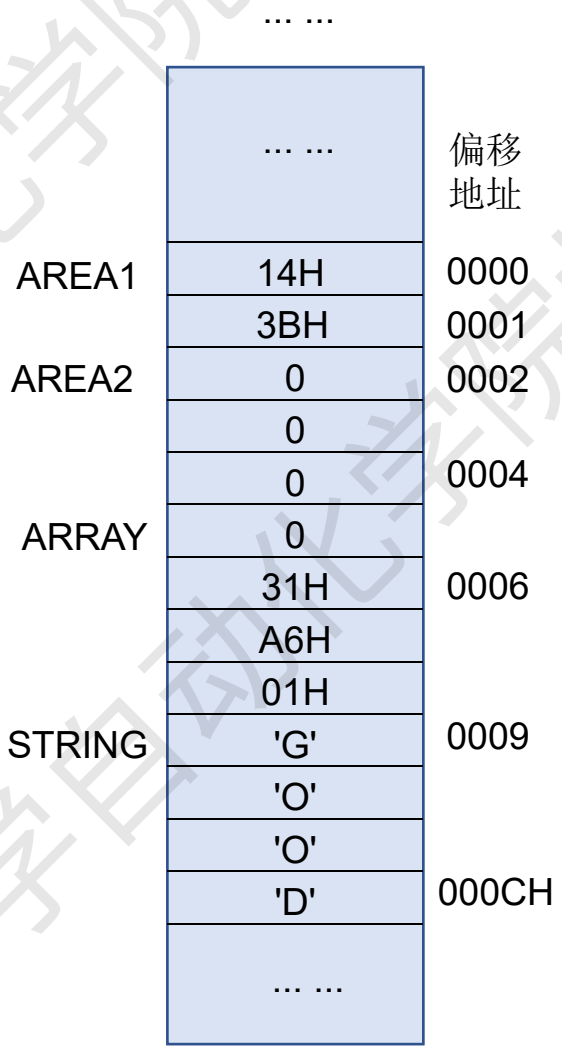
关键字
用于定义逻辑段

DB 用于定义字节
DW 用于定义字

```
DATA    SEGMENT
    AREA1   DB    14H,3BH
    AREA2   DB    3  DUP(0)
    ARRAY   DW    3100H, 01A6H
    STRING  DB      'GOOD'
DATA    ENDS
```

逻辑段名称

关键字
表示逻辑段定义结束



(2) PUSH进栈指令

- 指令格式：PUSH 源
- 指令功能：将源操作数推入堆栈
  - 源操作数可以是16位的通用寄存器、段寄存器或存储器中一个字的数据，但不能是立即数。
  - 每次执行PUSH指令，8086首先修改SP的值，SP-2→SP然后把源操作数（字）压入堆栈中SP指示的位置上。低位字节放第地址单元，高位字节放高地址单元。

(3) POP出栈指令

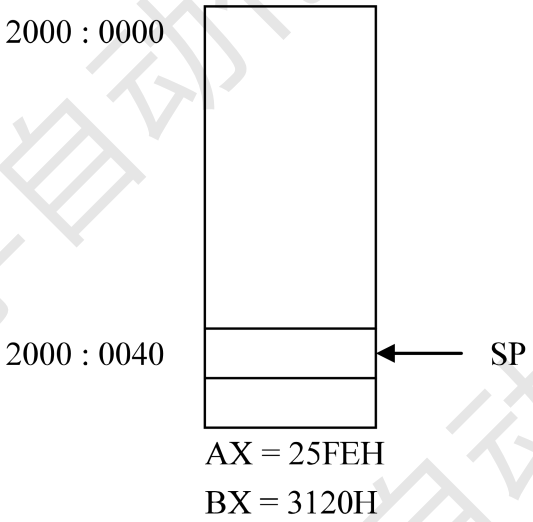
- 指令格式：POP 目的
- 指令功能：把当前SP所指向的堆栈顶部的一个字送到指定的目的操作数中。
  - 目的操作数可以是16位的通用寄存器、段寄存器或存储单元，但CS不能是目的操作数。
  - 每执行一次POP指令，SP+2→SP。

北京航空航天大学自动化学院林新

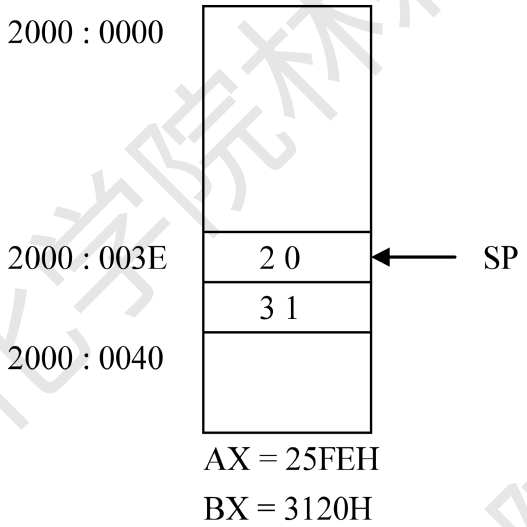
设：SS=2000H,
SP=40H,
BX=3120H,
AX=25FEH,
依次执行下列指令：

```
PUSH    BX
PUSH    AX
POP     BX
```

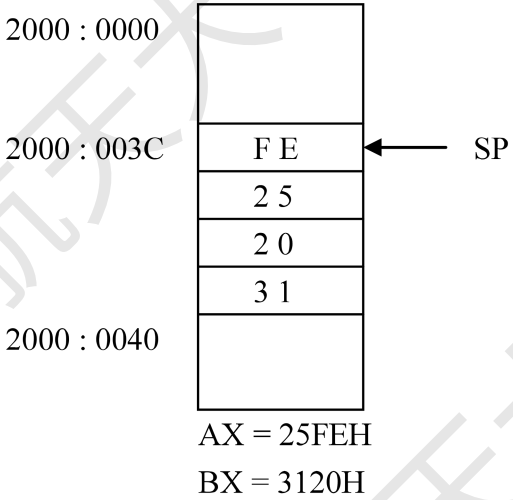
则最后结果
BX=25FEH,
SP=3EH,
(2003EH)=3120H



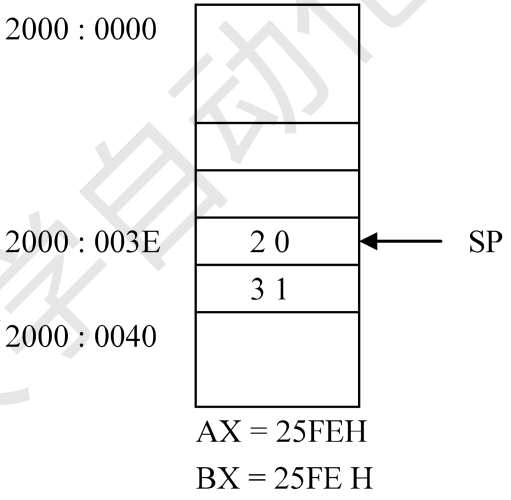
(a)指令执行前



(b)执行 PUSH　BX 之后



(c)执行 PUSH　AX 之后



(d)执行 POP　BX 之后

(4) XCHG交换指令

- 指令格式：XCHG 目的， 源
- 指令功能：把一个字或字节的源操作数和目的操作数相交换。
- 交换可以在寄存器之间、寄存器与存储器之间进行。但段寄存器不能作为操作数，也不能直接交换两个存储单元的内容。

例：

```
XCHG   AL, BL
XCHG  AREA1, DL

XCHG  AREA1,AREA2   ×
XCHG  AX, DS   ×
```

(5) XLAT表转换指令

- 指令格式：XLAT 转换表

  或 XLAT

- 指令功能：将一个字节从一种代码转换成另一种代码。
- 使用XLAT之前必须先建立一个表格，表格最多包含２５６个字节。将转换表的起始地址装入BX，将偏移量装入AL。执行XLAT之后，地址[BX+AL]中的数据就被传到AL中。

例3.31

```
TABLE   DB      40H, 79H, 24H, 30H, 19H
        DB      12H, 02H, 78H, 00H, 18H
                … …

        MOV     AL, 5
        MOV     BX, OFFSET TABLE
        XLAT    TABLE  ；查表得  AL=12H
```

2. 输入输出指令

例:
IN AL, 0AFH
或
MOV DX, 0AFH
IN AL, DX

(1) IN输入指令

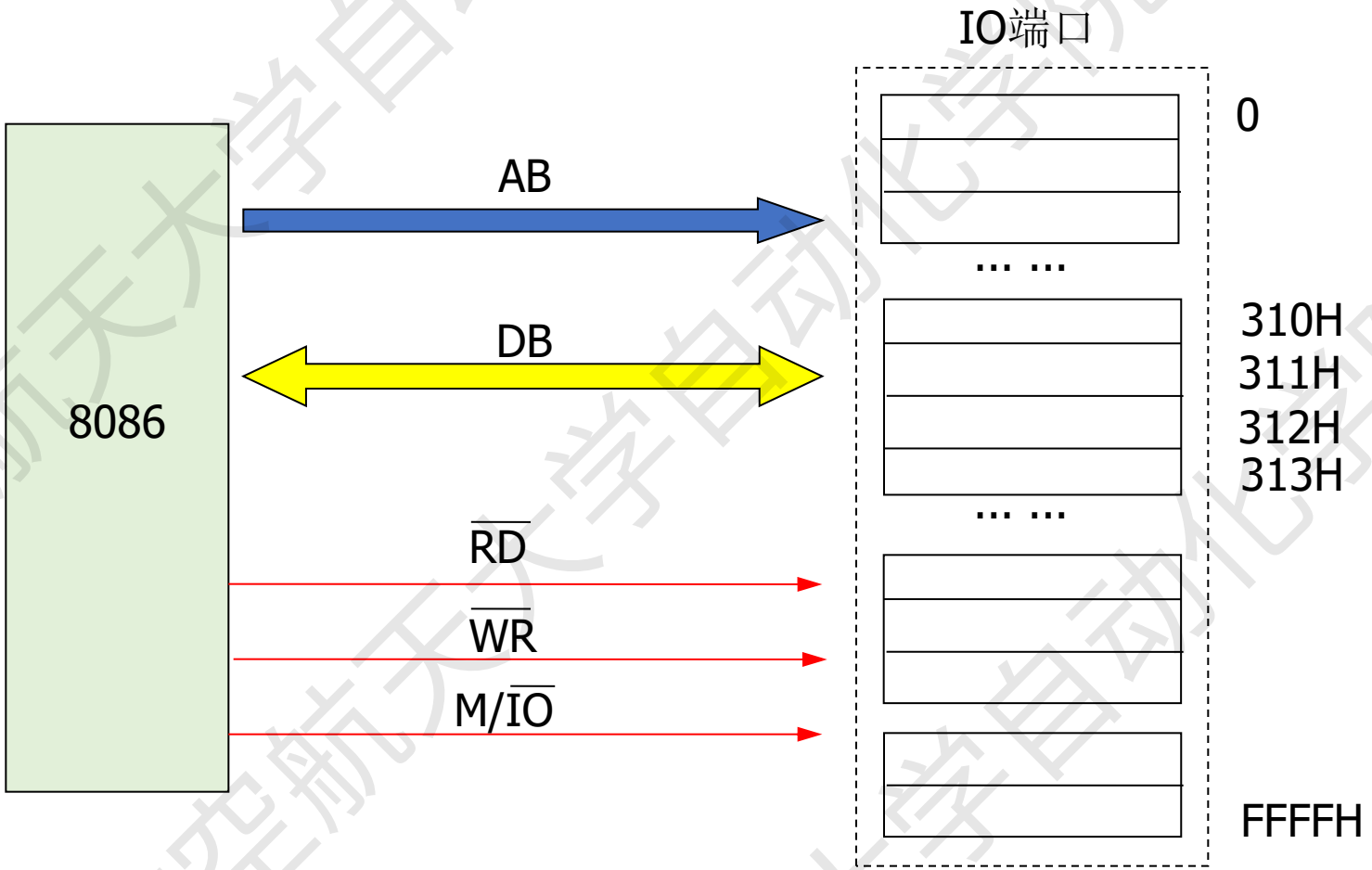
- 指令格式:
  ①IN AL, 端口地址 ;端口地址范围0~FFH
  ②IN AL, DX ;端口地址在DX中,范围0~FFFFH
- 指令功能: 从8位端口读取一个字节到AL中

例:
OUT 45H, AL
或
MOV DX, 45H
OUT DX, AL

(2) OUT指令

- 指令格式:
  ①OUT 端口地址, AL ;端口地址范围0~FFH
  ②OUT DX , AL ;端口地址在DX中,范围0~FFFFH
- 指令功能: 将AL中的一个字节写到一个8位端口

例：从端口310H读一个字节，再从312H端口输出到外设。



3. 地址目标传送指令

(1) LEA取有效地址指令 (要求掌握)

(2) LDS将双字指针送到寄存器和DS指令

(3) LES将双字指针送到寄存器和ES指令

北京航空航天大学自动化学院林新

（1）LEA取有效地址指令

- 指令格式： LEA 目的， 源
- 指令功能：取**源操作数**的**地址偏移量**，并把它传送到**目的操作数。**
  - LEA指令要求源操作数必须是存储单元，而且目的操作数必须是一个除段寄存器之外的16位寄存器。

例：

```
LEA   BX, AREA1
LEA   SI, ARRAY
LEA   DI, 10[SI]
```

```
DATA      SEGMENT
  AREA1   DB    14H,3BH
  AREA2   DB    3  DUP(0)
  ARRAY   DW    3100H, 01A6H
  STRING  DB    'GOOD'
DATA      ENDS
```

4.标志传送指令（不做要求）

(1) LAHF, 标志送到AH

(2) SAHF，AH送标志寄存器

(3) PUSHF，标志入栈

(4) POPF，标志出栈

北京航空航天大学自动化学院林新

3.3.2算术运算指令

- 加法

  ADD,ADC,INC，了解AAA,DAA
- 减法

  SUB,SBB,DEC, NEG, CMP
- 乘法（了解）

  MUL, IMUL
- 除法（了解）

  DIV, IDIV

北京航空航天大学自动化学院林新

1.加法/减法指令

(1) ADD加法指令

- 指令格式： ADD 目的， 源
- 指令功能：将源和目的操作数相加，结果送到目的操作数中。即：源 + 目的→目的

例： ADD DX, AX ; DX+AX→ DX

(2) ADC带进位的加法指令

- 指令格式：ADC 目的， 源
- 指令功能：将源、目的操作数、以及进位CF相加，结果送到目的操作数。即：源 + 目的 + CF→目的

例： ADC CX, BX ; CX+BX+CF→ CX

北京航空航天大学自动化学院林新

(3) SUB 减法指令

- 指令格式：SUB 目的，源
- 指令功能：将目的操作数减去源操作数，结果送回目的操作数。即：目的 - 源→目的

例： SUB DX, AX ; DX-AX→ DX

(4) SBB 带借位的减法指令

- 指令格式：SBB 目的，源
- 指令功能：将目的操作数减去源操作数、减去借位标志CF，结果送回目的操作数。即：目的 - 源 - CF→目的

例： SBB CX, BX ; CX-BX-CF→ CX

例：用加法指令求和：12ABC0H+34DEF0H，并分析标志位的影响

```
MOV     AX, ABC0H
MOV     BX, DEF0H
ADD     AX, BX      ; AX=8AB0H, CF=1

MOV   DX, 12H
MOV   BX,  34H
ADC    DX, BX       ; DX+BX+CF→ DX
                    ; DX = 47H
                    ;结果为（DX,AX)=478AB0H
```

例3-53 设AL=1011 0001B, DL=0100 1010B, 求SUB AL, DL

运算后标志位ZF=0, AF=1, CF=0, SF=0, PF=0, OF=1

（5）CMP 比较指令

- 指令格式：CMP 目的，源
- 指令功能：将目的操作数减去源操作数，结果不回送到目的操作数，仅仅影响标志位。即：目的 - 源 → PSW

- CMP的使用
  - 与**JA/JE/JB**配合使用：判断无符号数大小
  - 与**JG/JE/JL**配合使用：判断有符号数（补码）大小

```
例：      MOV    AL, 12H
          CMP    AL, 34H

          JA     NEXT1    ; JMP when Above
          MOV    AH, 0
          JMP    NEXT2    ; 无条件转移

NEXT1:  MOV    AH, 1
NEXT2:  MOV    DL, AH   ⟹   DL=?
```

（6）NEG 取负指令

- 指令格式：NEG 目的
- 指令功能：对目的操作数取负，即用0减去操作数，再把结果送回目的操作数： 0 - 目的→目的（此指令是补码运算）

例：MOV DL, 03H
NEG DL ; DL= 1111 1101B =-3

NEG BYTE PTR [BX]

- 注意事项
  - 执行指令后，影响所有标志位。
  - 如果操作数在存储器中，则必须标明字还是字节。

2.增量/减量指令

例:
```
INC   BX
INC   BYTE PTR [SI]
INC   WORD PTR [SI]
```

(1) INC 增量指令

- 指令格式：INC 目的
- 指令功能：对目的操作数加1，结果送回目的操作数。目的操作数可以是寄存器或内存单元。即: 目的 + 1→目的

例:
```
DEC   BX
DEC   BYTE PTR [SI]
DEC   WORD PTR [SI]
```

(2) DEC 减量指令

- 指令格式：DEC 目的
- 指令功能：对指定的目的操作数减1，结果送回此操作数。即: 目的 - 1→目的

- 注意事项：
  - 执行指令后，标志位中只有CF不受影响。
  - 如果操作数在存储器中，则必须标明字还是字节。

- 对于双操作数指令ADD, ADC,SUB, SBB, CMP：
  - 目的操作数不能是立即数
  - 两个操作数不能都是存储器单元
  - 源和目的操作数的类型必须一致，即都是字节或者字
  - 影响所有标志位：CF/OF/PF/SF/ZF/AF
- 对于单操作数指令INC、DEC、NEG：
  - 目的操作数不能是立即数
  - 如果操作数在存储器中，则必须标明字还是字节

北京航空航天大学自动化学院林新

3. BCD码/ASCII码加减运算调整指令（了解）

（1）DAA 加法的十进制调整指令

（2）AAA 加法的ASCII调整指令

（3）AAS 减法的ASCII调整指令

（4）DAS 减法的十进制调整指令

北京航空航天大学自动化学院林

（1）DAA 加法的十进制调整指令

- 指令格式：DAA
- 指令功能：将两个压缩BCD数相加后的结果调整为正确的压缩BCD数。相加后的结果必须在AL中才能用DAA指令。

*本指令不要求掌握，但实验可能会用到

例3-47 若AL=(BCD)88H, BL=(BCD)49H, 求两者之和。也就是两个数看作10进制加法88+49，得到结果137H（BCD）

```
ADD   AL, BL; 88H+49H = D1H, AF=1
DAA           ; AL= 37H ,结果为(BCD)37H，CF=1
```

（4）AAA 加法的ASCII调整指令

- 指令格式：AAA
- 指令功能：用ADD或ADC对两个非压缩BCD或ASCII码作加法后，运算结果已经存在AL寄存器中。此时用AAA指令把AL中的运算结果调整为1位非压缩BCD码数，结果仍然保存在AL中。如果有进位，则进位进到AH中，并且AF=1。

  *本指令不要求掌握，但实验可能会用到

例3-44 若AL=BCD9, BL=BCD5, 求两者之和。设AH=0。

```
ADD   AL, BL    ; 09H+05H= 0EH
AAA             ; AL= 04H,  CF=1,AF=1,AH=1
```

例3-45 求ASCII 39H与ASCII 35H之和（即9＋5）

```
MOV   AL, ’9’
MOV   BL, ‘5’
ADD    AL, BL   ; 39H+35H=6EH,
AAA             ; AL = 04H, CF=1,AF=1,AH=1
```

练习：设有两个十进制数据789和532

（1）试定义数据段和变量将其存为非压缩BCD码数据

（2）编程完成非压缩BCD码数789+532的计算并保存结果。

```
DATA   SEGMENT
    DT1         DB     7,8,9
    DT2         DB     5,3,2
    DT3         DB     4  DUP(0)
DATA   ENDS
```

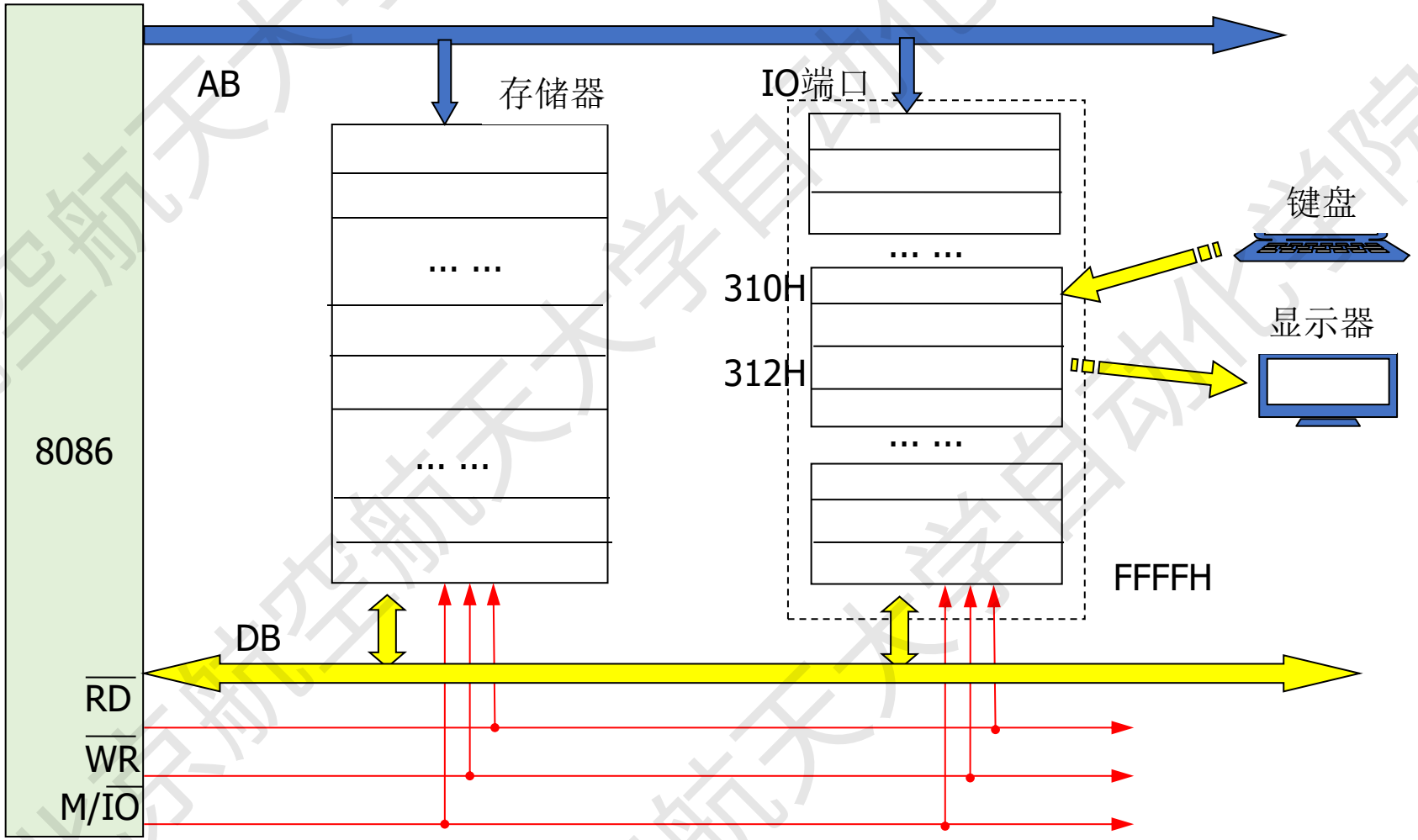
北京航空航天大学自动化学院林新

练习：设键盘的按键按下后，其ASCII码自动存入到IO端口310H中；CPU写入IO端口312H的数据自动被送往显示器并显示（只有ASCII码才能正确显示）。

（1）在显示器上显示变量AREA1后面定义的4和0BH，显示STRING后面定义的'GOOD'

（2）从键盘输入3个字符，保存在AREA2，并在显示器上显示

```
DATA    SEGMENT
AREA1   DB      4H,0BH
AREA2   DB      3  DUP(0)
ARRAY   DW      3100H, 01A6H
STRING  DB      ‘GOOD’
DATA    ENDS
```



4.乘法指令（了解）

例:
```
MOV  AL, 12H
MOV  BL, 34H
MUL  BL        ;AL*BL→AX
```

（1）MUL 无符号数乘法指令

- 指令格式：MUL 源
- 指令功能：把源操作数和累加器中的数作为**无符号数**进行相乘。源操作数可以是字或者字节。

① 如果操作数是字节，那么与AL相乘；乘积为16位，存放在AX中。即：AL*源→AX

② 如果操作数是字，则与AX相乘，乘积为32位，存放在（DX,AX）中，其中DX存放高16位，AX存放低16位。
即：AX*源→ (DX,AX)

例:
```
MOV  AX, 1200H
MOV  BX, 0034H
MUL  BX    ;AX*BX→(DX,AX)
```

(2) IMUL 整数乘法指令

(3) AAM乘法的ASCII调整指令

北京航空航天大学自动化学院林新

4. 除法指令（了解）

（1）DIV 无符号数除法指令

- 指令格式：DIV 源
- 指令功能：对两个**无符号数**进行除法。源操作数可以是字或者字节，可以是存储单元或者寄存器。

① 如果**源操作数为字节**，则16位的被除数必须存放在AX中。如果被除数只有8位，则放在AL中，AH清0。相除之后，8位的商在AL中，余数在AH中。

AX/源（字节）的商→AL

AX/源（字节）的余数→AH

例：MOV AX, 1200H
MOV BL, 34H
DIV BL ;AX/BL→AL, AH

② 如果**源操作数是字**，则32位的被除数中高16位放在DX中，低16位放在AX中；如果被除数只有16位，则放入AX中，DX清0。相除之后，AX中放16位商，DX中放16位余数。即：

(DX, AX)/源（字）的商→AX

(DX, AX)/源（字）的余数→DX

(2) IDIV 整数除法指令

(3) CBW 把字节转换为字指令

(4) CWD 把字转换为双字指令

(5) AAD 除法的ASCII调整指令

北京航空航天大学自动化学院林新

3.3.3 逻辑运算和移位指令

1. 逻辑运算

   NOT, AND, OR, XOR, TEST

2. 算术逻辑移位

   SHL/SAL, SHR, SAR

3. 循环移位

   ROL, ROR, RCL, RCR

北京航空航天大学自动化学院林新

1. 逻辑运算指令

(1) NOT取反指令

- 指令格式：NOT 目的
- 指令功能：将目的操作数求反，结果送回目的操作数，即：

$$\overline{\text{目的}} \rightarrow \text{目的}$$

  - 目的操作数可以是8位或者16位寄存器或者存储器。
  - 对于存储器操作数，要说明是字节还是字
  - 指令执行后对标志位无影响。

例：

```
MOV   AL, 1001 0011B        ; AL=1001 0011B
NOT   AL                    ; AL=0110 1100B
```

例：
```
MOV   AL, 0000 1001B  ; AL = 9
OR    AL, 0011  0000B ; AL ∨ 30H →39H
```

(2) OR逻辑或

- 指令格式：OR 目的，源
- 指令功能：对两个操作数进行按位逻辑或操作，结果送回目的操作数，即：目的∨源→目的

* OR的用途：对指定位置1；或将0~9数值变为对应的ASCII码

例：
```
MOV   AL, 1111 1111B  ;
XOR   AL, 0000 1001B ; AL=1111 0 11 0B
```

(3) XOR异或

- 指令格式：XOR 目的，源
- 指令功能：对两个操作数进行按位异或运算，结果送回目的操作数。即：目的 XOR 源→目的

* XOR的用途：对指定位取反。即：取反的位为1，不变的位为0

(4) AND逻辑与

例：
```
MOV   AL, 0011 1001B  ; AL = 39H
AND   AL, 0000 1111B  ; AL = 09H
```

- 指令格式：AND 目的，源
- 指令功能：对两个操作数进行按位逻辑与操作，结果送回目的操作数，即：目的←目的∧源

* AND的用途：对指定位清0；或将0~9的ASCII码变为对应数值

(5) TEST测试

例：
```
MOV   AL, 1011 1001B  ; AL = B9H
TEST  AL, 1000 0000B  ; AL = B9H, ZF =0
JZ      NEXT
```

- 指令格式：TEST 目的，源
- 指令功能：对两个操作数进行逻辑与操作，并修改标志位，但不回送结果。指令执行后两个操作数都不改变，即：目的∧源

* 用于测试某位是否为1（或0）。即：待测试位设为1，其它位为0
* 通常与JZ、JNZ配合使用

- 逻辑与通常用于实现对指定位清0，其余位不变；
- 逻辑或通常用于实现对指定位置1，其余位不变；
- 逻辑异或通常用于实现指定位取反，其余位不变。

北京航空航天大学自动化学院林新

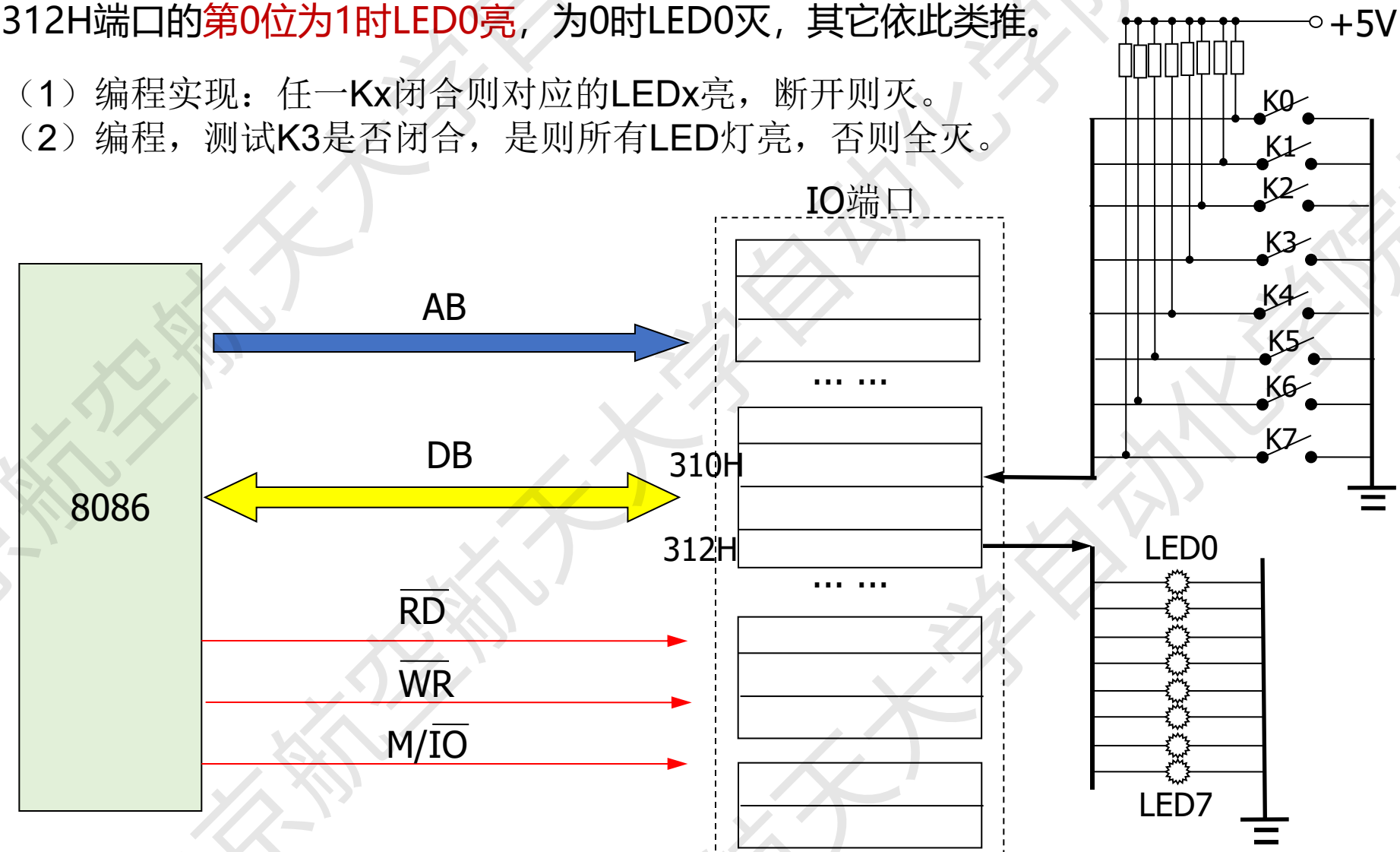
练习：TEST指令

设系统如图所示，开关状态（断开代表逻辑1，闭合代表逻辑0）被自动送往IO端口310H，K0对应310H端口的第0位，K7对应第7位，其它依此类推。

IO端口312H中的数据自动送往发光二极管LEDx，每一位都和一个LED灯相对应。如312H端口的第0位为1时LED0亮，为0时LED0灭，其它依此类推。

（1）编程实现：任一Kx闭合则对应的LEDx亮，断开则灭。

（2）编程，测试K3是否闭合，是则所有LED灯亮，否则全灭。



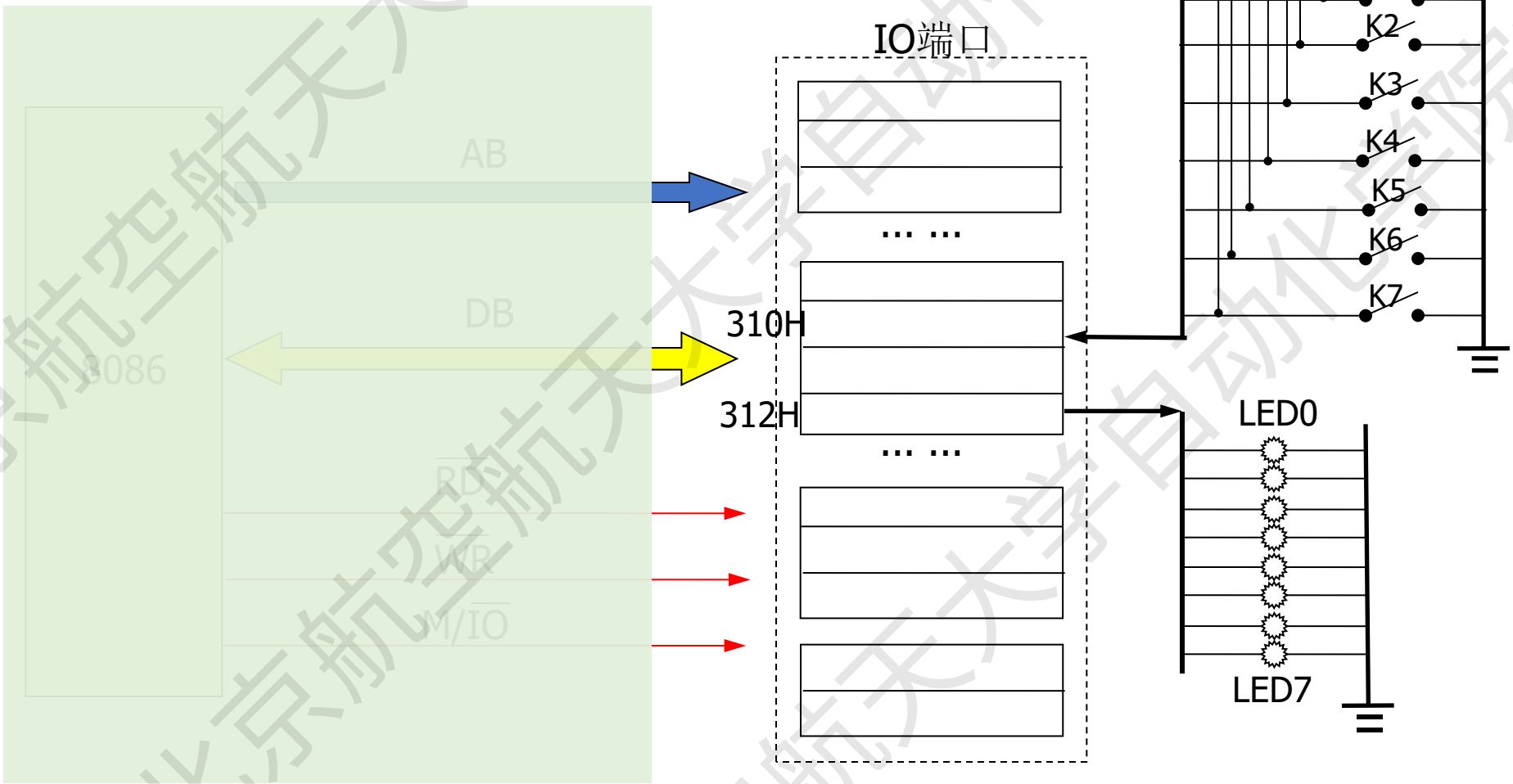
练习：TEST指令

设系统如图所示，开关状态（断开代表逻辑1，闭合代表逻辑0）被自动送往IO端口310H，K0对应310H端口的第0位，K7对应第7位，其它依此类推。

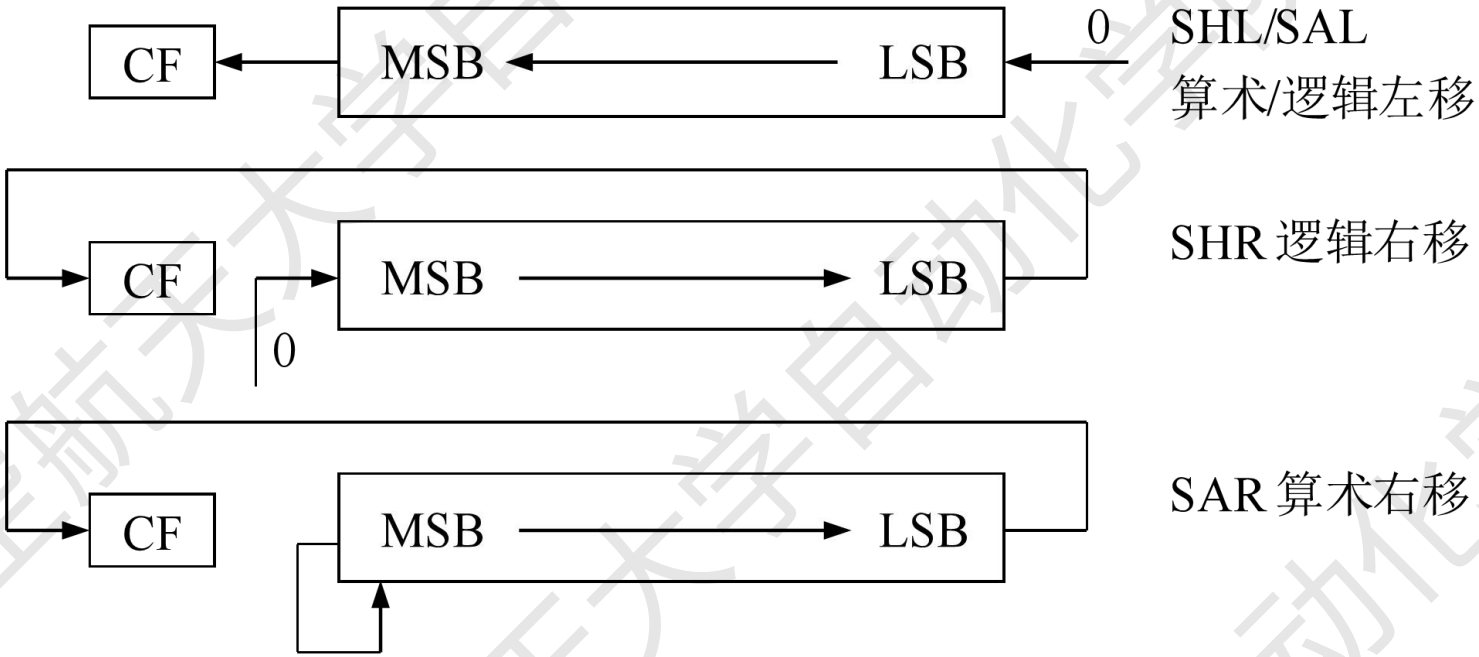
IO端口312H中的数据自动送往发光二极管LEDx，每一位都和一个LED灯相对应。如312H端口的第0位为1时LED0亮，为0时LED0灭，其它依此类推。

（1）编程实现：任一Kx闭合则对应的LEDx亮，断开则灭。

（2）编程，测试K3是否闭合，是则所有LED灯亮，否则全灭。



2. 算术逻辑及移位指令



- 若只移动一位，则可以直接在指令中把计数值写成1；
- 如果移动多于1位，则计数值放入CL中，再把CL放在指令的计数值位置上。

(1) SAL算术左移

- 指令格式：SAL 目的，计数值

(2) SHL 逻辑左移

- 指令格式：SHL 目的，计数值

- 指令功能：以上两种指令功能完全相同，都是将寄存器或存储器中目的操作数的各位向左移，每移一次，最低有效位LSB补0，最高有效位MSB进而标志位CF。
- 指令中的计数值决定移位的次数。若只移动一位，则可以直接在指令中把计数值写成1；如果移动多于1位，则计数值放入CL中，再把CL放在指令的计数值位置上。

- 每左移一位相当于目的操作数乘2(移位后不超出表达范围)

(1)

例：
```
MOV   AL, 0011 1011B   ;AL= 0011 1011B= 59
SHL    AL,1            ;AL= 0111 0110B=118
```

(2)

```
MOV   AL, 1110  0101B  ; AL=1110  0101B=-27
MOV  CL, 2
SAL   AL, CL           ; AL=1001 0100B =-108
```

- 指令功能：以上两种指令功能完全相同，都是将寄存器或存储器中目的操作数的各位向左移，每移一次，最低有效位LSB补0，最高有效位MSB进而标志位CF。
- 指令中的计数值决定移位的次数。若只移动一位，则可以直接在指令中把计数值写成1；如果移动多于1位，则计数值放入CL中，再把CL放在指令的计数值位置上。

➢ 每左移一位相当于目的操作数乘2(移位后不超出表达范围)

(3) SHR 逻辑右移

- 指令格式：SHR 目的，计数值
- 指令功能：对目的操作数中的各位进行右移，每执行一次移位操作，操作数右移一位，最低位进入CF，最高位补0。右移次数由计数值决定，同SAL/SHL指令一样。

(4) SAR算术右移

- 指令格式：SAR 目的，计数值
- 指令功能：功能与SHR相似，但是每次移位，最高位保持不变，而非补0。

➢ 每右移一位相当于目的操作数除以2(移位后不超出表达范围)

北京航空航天大学自动化学院林新

(3) SHR 逻辑右移

- 例：
- MOV AL, 1011 1011B ;AL= 1011 1011B=187
  SHR AL,1 ;AL= 0101 1101B=93

于一次
位补0。

MOV AL, 1011 1011B ; AL=1011 1011B=-69
MOV CL, 2
(4) SAR AL, CL ; AL=1110 1110B=-18

- 指令格式：SAR 目的，计数值
- 指令功能：功能与SHR相似，但是每次移位，最高位保持不变，而非补0。

➢ 每右移一位相当于目的操作数除以2(移位后不超出表达范围)

3. 循环移位

循环移位指令把操作数从一端移到另一端，移动的位不会丢失。

（1）ROL循环左移

- 指令格式：ROL 目的，计数值

（2）ROR 循环右移

- 指令格式：ROR 目的，计数值

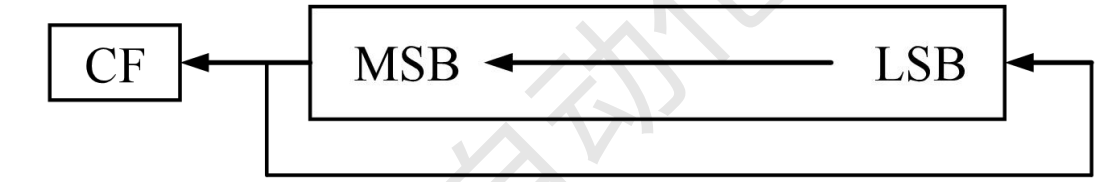
（3）RCL 通过进位位循环左移

- 指令格式：RCL 目的，计数值

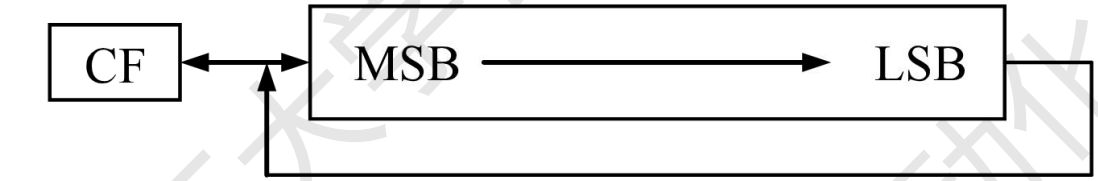
（4）RCR 通过进位位循环右移

- 指令格式：RCR 目的，计数值

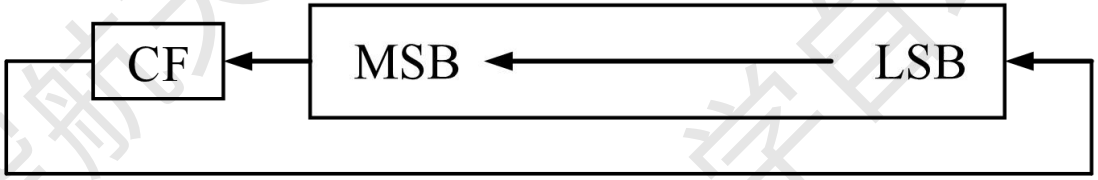
北京航空航天大学自动化学院林新



ROL 循环左移



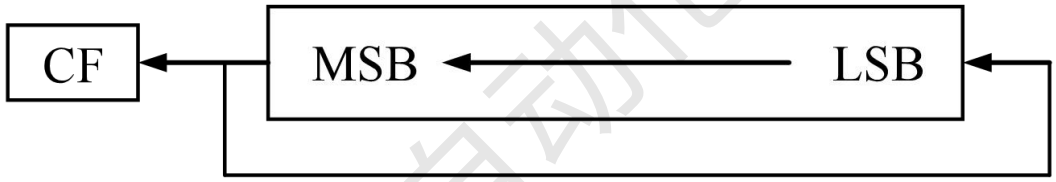
ROR 循环右移



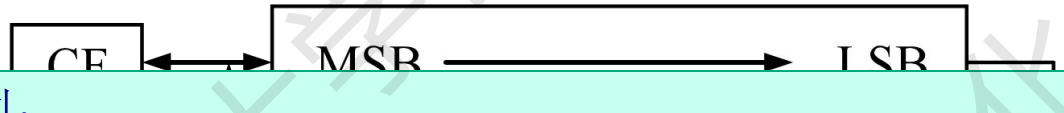
RCL 通过进位的循环左移

RCR 通过进位的循环右移

- 若只移动一位，则可以直接在指令中把计数值写成1；
- 如果移动多于1位，则计数值放入CL中，再把CL放在指令的计数值位置上。



ROL 循环左移



ROR 循环右移

例：

```
MOV   AL, 1011 1011B   ;AL= 1011 1011B
ROL    AL,1                    ;AL= 0111 0111B,   CF= 1

MOV   AL, 1011 1011B   ; AL=1011 1011B
MOV  CL, 3
ROL    AL, CL                 ; AL=1 1011101B,   CF=1
```

进位的

RCR 通过进位的循环右移

- 若只移动一位，则可以直接在指令中把计数值写成1；
- 如果移动多于1位，则计数值放入CL中，再把CL放在指令的计数值位置上。

- 4条指令都按指令中计数值的移位次数进行循环移位，移位后的结果仍送回目的操作数。目的操作数可以是8/16位的寄存器操作数或内存操作数，循环的次数可以是**1**，也可以由**CL**寄存器的值指定。
- ROL和ROR没有把CF标志位包含在内；而RCL和RCR把CF标志位也包含在循环内。

3.3.4字符串处理指令

这里的字符串是指一系列存放在存储器中的字或字节数据，不管它们是不是ASCII码。字符串长度可达64KB，组成字符串的字节或字称为字符串元素。

- MOVS
- CMPS
- SCAS
- LODS
- STOS

北京航空航天大学自动化学院林新

（1） MOVS 字符串传送指令

- 指令格式： MOVS 目的串，源串
- 指令功能：把由SI作指针的源串中的一个字节或字，传送到由DI作指针的目的串中，且自动修改指针SI和DI。 即：(DS:SI)→(ES:DI)，然后修改SI、DI

例：

```
MOVSB   ; 字节(DS:SI)→(ES:DI)；DF=0时SI、DI加1，DF=1时减1
MOVSW   ; 字(DS:SI)→(ES:DI)；DF=0时SI、DI加2，DF=1时减2
```

- 可与REP联用，方法：

```
MOV  CX, 100
REP  MOVSB  ;重复执行MOVSB指令100遍
```

例3-75 要求把数据段中以SRC_MESS为偏移地址的一串字符“HELLO!”传送到附加段中以NEW_LOC开始的单元中。

```
DATA        SEGMENT
SRC_MESS    DB 'HELLO!'
DATA        ENDS
EXTRA       SEGMENT
NEW_LOC     DB 6 DUP(?)
EXTRA       ENDS
CODE        SEGMENT
            ASSUME CS:CODE,  DS:DATA, ES:EXTRA
START:      MOV    AX, DATA
            MOV    DS, AX
            MOV    AX, EXTRA
            MOV    ES, AX
            LEA    SI, SRC_MESS
            LEA    DI, NEW_LOC
            MOV    CX, 6
            CLD
            REP    MOVSB
CODE        ENDS
            END    START
```

字符串指令的隐含约定：

1. **源串**位于当前数据段中，由DS寻址；源串的元素由SI作为指针，即源串字符的起始地址（或末地址）为DS:SI，源串允许使用段超越前缀来修改段地址。
2. **目的串**必须位于当前附加段中，目的字符串的起始地址（或末地址）为ES:DI。目的串不允许使用段超越前缀修改ES。如果要在同一段内进行串运算，必须使DS和ES指向同一段。

北京航空航天大学自动化学院林新

3. 每执行一次字符串指令，指针SI和DI会自动修改，以便指向下一个待操作的单元。

4. **DF标志**控制字符串的处理方向。可用STD使DF=1，CLD使DF=0。
   - DF=0为递增方向，DS:SI指向源串起始地址，ES:DI指向目的串起始地址。每执行一次字节串操作，SI和DI都增1；每执行一次字操作，SI和DI都增2。
   - DF=1为递减方向，DS:SI指向源串末地址，ES:DI指向目的串末地址。每执行一次字节串操作，SI和DI都减1；每执行一次字操作，SI和DI都减2。

5. 要处理的字符串长度（字数或字节数）放在CX寄存器中。

(2) CMPS字符串比较指令

- 指令格式： CMPS 目的串，源串
- 指令功能：从SI作指针的源串中减去由DI作指针的目的串的数据，相减的结果反映的标志位上，但不改变两个数据串的原始值。操作结束后SI和DI的内容自动修改。

  - 一般与REPZ或REPNZ联用，如：

    REPZ CMPSB --当ZF=1时重复执行CMPSB

```
DATA        SEGMENT
PASSWORD    DB          ‘750430LI’
IN_WORD     DB          ‘750424LE’
COUNTEQU    8
DATA        ENDS
CODE        SEGMENT
            ASSUME      DS:DATA, ES:DATA
            MOV   AX, DATA
            MOV   DS, AX
            MOV   ES, AX
            LEA   SI, PASSWORD
            LEA   DI, IN_WORD
            MOV   CX, COUNT
            CLD
            REPZ  CMPSB
            JNE   SOUND
OK:…
SOUND:
CODE        ENDS
```

（3） SCAS 字符串扫描指令

指令格式： SCAS 目的串

指令功能：从AL(字节操作)或AX（字操作）寄存器的内容减去附加段中以DI为指针的目的串元素，结果反映在标志位上，但不改变源操作数。操作后DI自动修改。

例3-77 设字符串起始地址STRING的偏移地址为0，长度为COUNT。在字符串中寻找字符’A’，搜索次数记录到BX中。没有找到则BX清0。

```
        MOV     DI, OFFSET STRING
        MOV     CX, COUNT
        MOV     AL, ‘A’
        CLD
        REPNE   SCASB
        JZ      FIND
        MOV     DI, 0
FIND:MOV    BX, DI
        HLT
```

（4） LODS 数据串装入指令

指令格式：LODS 源串

指令功能：把数据段中以SI为指针的串元素，传送到AL（字节操作）或AX（字操作）中，同时修改SI，使它指向串中的下一个元素，SI的修改量由方向标志DF和源串类型决定。

（5） STOS数据串存储指令

指令格式： STOS 目的串

指令功能：将AL或AX中的内容，传送到附加段中以DI为目的指针的目的串中。操作完后修改DI。

3.5.5 控制转移指令

- 无条件转移和过程调用指令

  掌握JMP, CALL/RET
- 条件转移

  掌握JZ/JE, JA/JNB等等
- 条件循环控制

  掌握LOOP,

  了解LOOPE/LOOPZ, LOOPNE/LOOPNZ, JCXZ
- 中断

北京航空航天大学自动化学院林新

1.无条件转移和过程调用指令

(1) JMP 无条件转移指令

- 指令格式：JMP 目的
- 指令功能：使程序无条件地转移到指令中指定的目的地址去执行。

类型	方式	寻址目标	指令举例
段内转移	直接	立即短转移（8 位）	JMP SHORT PROG_S
		立即近转移（16 位）	JMP NEAR PROG_N
	间接	寄存器（16 位）	JMP BX
		存储器（16 位）	JMP WORD PTR 5[BX]
段间转移	直接	立即转移（32 位）	JMP FAR PTR PROG_F
	间接	存储器（32 位）	JMP DWORD PTR [DI]

1）段内直接转移指令

指令格式：

1）短转移 JMP　　SHORT　　标号

- 短转移目标标号地址距离当前IP的距离在-128~127字节之间

例：JPM　SHORT　NEXT1　;

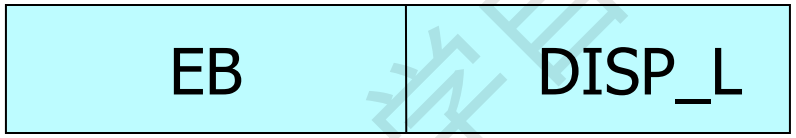
2）近转移JMP　　NEAR PTR　标号　（或 JMP 标号）

- 近转移目标标号地址距离当前IP的距离在-32768 ~32767字节之间

例：JMP　NEAR PTR NEXT2　;
　　JMP　NEXT3

北京航空航天大学自动化学院林新

段内短转移指令机器码



例1：设JMP NEXT1指令的偏移地址为25H，机器码为 EB 36H，则转移的目标地址是多少？

CPU执行操作：

$IP = IP_{当前} + DISP_8 = EA_{JMP} + 2 + DISP_8$ (SHORT)

段内近转移指令机器码

例2：设JMP NEXT1指令的偏移地址为1200H，机器码为 E9 20 EFH，则转移的目标地址是多少？

CPU执行操作：

$IP = IP_{当前} + DISP_{16} = EA_{JMP} + 3 + DISP_{16}$ (NEAR)

$DISP_8$ 为带符号8位二进制数

$DISP_{16}$ 为带符号16位二进制数

例如：

```
0000                    CODE        SEGMENT
                                    ASSUME  CS:CODE
0000  04 05             PROG_S:     ADD   AL, 05H
0002  90                            NOP
0003  EB FB                        JMP    SHORT PROG_S
0005  90                            NOP
0006                                ENDS
                                    END
```

注：执行程序时，当前IP寄存器的值指向下一条将要执行的指令

北京航空航天大学自动化学院林新

(2) CALL 和RET 过程调用和返回指令

- 指令格式：CALL 过程名
- 指令格式：RET

北京航空航天大学自动化学院林新

例：过程（子程序）定义及调用

```
CODE        SEGMENT
MAIN        PROC
            … …
            MOV     SI, 100H
            MOV     AL, 11H
            MOV     BL, 22H
            CALL    ADD_ABR
            MOV     [SI], AL
            RET
MAIN        ENDP

ADD_ABR     PROC    NEAR
            ADD     AL, BL
            RET
ADD_ABR     ENDP

CODE        SEGMENT
```

- 主程序：可执行程序的主体模块，调用其它子程序。
- 过程（Procedure）/子程序（Subroutine）
  - 某些特定功能，或者经常使用的功能，在编程序时写成独立的模块，供其它程序调用。
  - 过程开头以语句PROC开头，以语句ENDP结束。
  - 在ENDP之前放一条返回指令RET，与CALL指令相呼应。
- 过程（子程序）的调用使用CALL指令，过程执行结束后使用RET语句返回到调用点。
- 过程的调用分为近调用和远调用两种。
  - 近调用是指调用指令CALL和所调用的过程在同一代码段；
  - 远调用，指令CALL和所调用的过程在不同代码段中。

CALL和RET的四种寻址方式

1）段内直接调用

2）段内间接调用

3）段间直接调用

4）段间间接调用

北京航空航天大学自动化学院林新

CALL/RET指令的操作（近调用）

- CALL指令操作

1. 返回地址（CALL指令之后的那条指令的地址，也是当前IP寄存器的值）入堆栈。

   SP-2→SP，IP入栈

2. 转到子程序的入口地址，去执行相应的程序。

- RET指令操作

1. 从堆栈弹出一个字→IP，并且SP＋2→SP
2. 返回调用点，继续执行原程序

北京航空航天大学自动化学院

CALL/RET指令的操作（远调用）

- CALL指令操作

1. 返回地址（CALL指令之后的那条指令的地址，也是当前IP寄存器的值）入堆栈 。

对于远调用，执行的操作：

SP-2→SP，CS入栈

SP-2→SP，IP入栈

2. 转到子程序的入口地址，去执行相应的程序。

- RET指令操作

1. 从堆栈弹出一个字→IP，并且SP＋2→SP
再从堆栈弹出一个字→CS，并且SP＋2→SP
2. 返回调用点，继续执行原程序

北京航空航天大学自动化学院

1）段内直接调用和返回

例3-84

```
MOV  AX, BX
CALL   PROG_N      ;PROG_N是近标号
NOT   AX
……
```

设调用前：

CS=2000H, IP=1050H, SS=5000H, SP=0100H, PROG_N与CALL之间的字节距离等于1234H（即DISP=1234H）

CALL指令操作步骤为：

1. SP←SP-2，IP（返回地址）入栈
2. IP←IP+DISP(PROG_N)，转子程序入口
3. 执行子程序

例3-83

```
MOV  AX, BX
CALL      PROG_N
NOT       AX
```

设调用前：

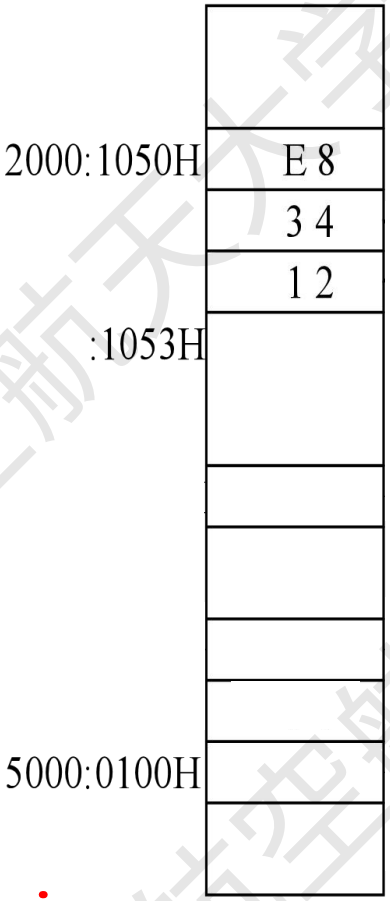
CS=2000H, IP=1050H, SS=5000H, SP=0100H,

PROG_N与CALL之间的字节距离等于1234H（即DISP=1234H）

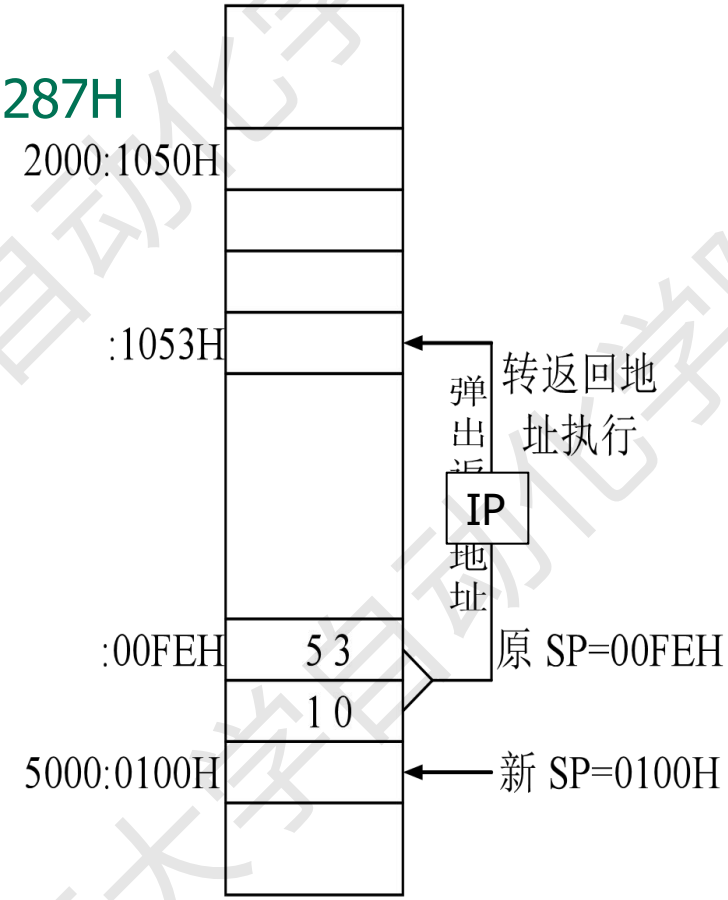
1.SP-2, 返回地址1053H入栈

2.目标地址：

IP+3+1234H=2287H



a）执行CALL指令



b） 执行RET指令

例：

```
CODE        SEGMENT
MAIN        PROC
            … …
            MOV     SI, 100H
            MOV     AL, 11H
            MOV     BL, 22H
            CALL    ADD_ABR        ；设此指令地址EA=1200H
            MOV     [SI], AL       ；设此指令地址EA=1203H
            RET                    ；设此指令地址EA=1205H
MAIN        ENDP

ADD_ABR     PROC    NEAR
            ADD     AL, BL         ；设此指令地址EA=1206H
            RET
ADD_ABR     ENDP
CODE        SEGMENT
```

2. 条件转移指令

- 条件转移指令是根据当前标志寄存器中标志位的状态来决定是否转移。
- 条件转移指令都是段内短转移指令，允许跳转的距离为-128~127。

北京航空航天大学自动化学院林新

（1）直接标志转移指令

指令助记符	测试条件	指令功能
JC	CF=1	有进位 转移
JNC	CF=0	无进位 转移
JZ/JE	ZF=1	结果为 0/相等 转移
JNZ/JNE	ZF=0	不为 0、不相等 转移
JS	SF=1	符号为负 转移
JNS	SF=0	符号为正 转移
JO	OF=1	溢出 转移
JNO	OF=0	无溢出 转移
JP/JPE	PF=1	奇偶位为 1/为偶 转移
JNP/JPO	PF=0	奇偶位为 0/为奇 转移

JC/JNC, JZ/JNZ(JE/JNE), JS/JNS

北京航空航天大学自动化学院

（2）间接标志转移

类别	指令助记符	测试条件	指令功能
无符号数比较测试	JA/JNBE	CF∨ZF=0	高于/不低于等于 转移
	JAE/JNB	CF=0	高等于于/不低于 转移
	JB/JNAE	CF=1	低于/不高于等于 转移
	JBE/JNA	CF∨ZF=1	低于等于/不高于 转移
带符号数比较测试	JG/JNLE	(SF∀OF) ∨ZF=0	大于/不小于等于 转移
	JGE/JNL	SF∀OF=0	大于等于/不小于 转移
	JL/JNGE	SF∀OF=1	小于/不大于等于 转移
	JLE/JNG	(SF∀OF) ∨ZF=1	小于等于/不大于 转移

- A—Above, B—Below, E—Equal, G—Great, L—Less, N—not

JA/JNBE, JB/JNAE, JG/JNLE, JL/JNGE…

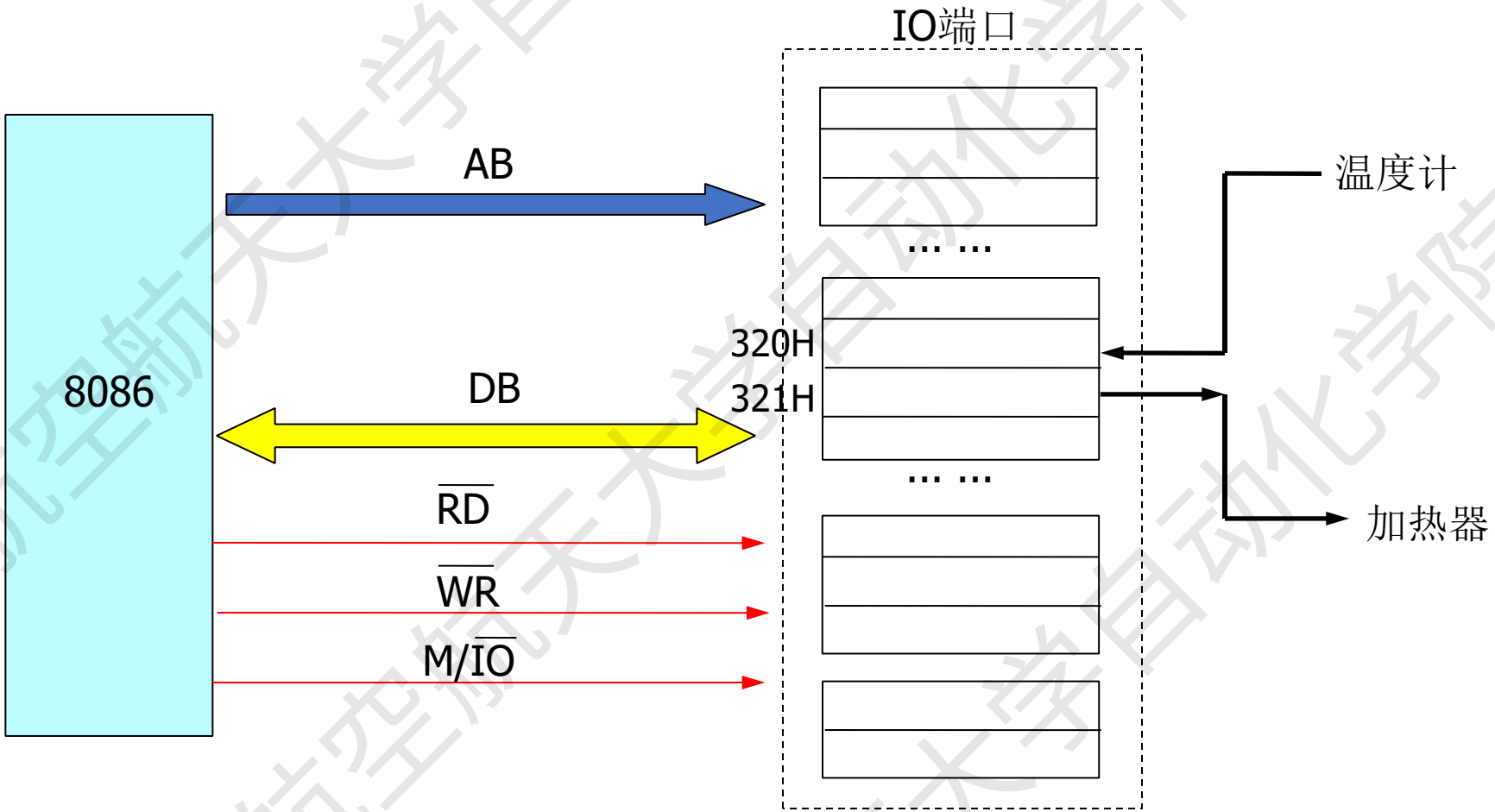
北京航空航天大学自动化学院

例3.90 设学生成绩放在AL寄存器中，若低于60则打印'F'，高于85则打印'G'，其它打印'P'

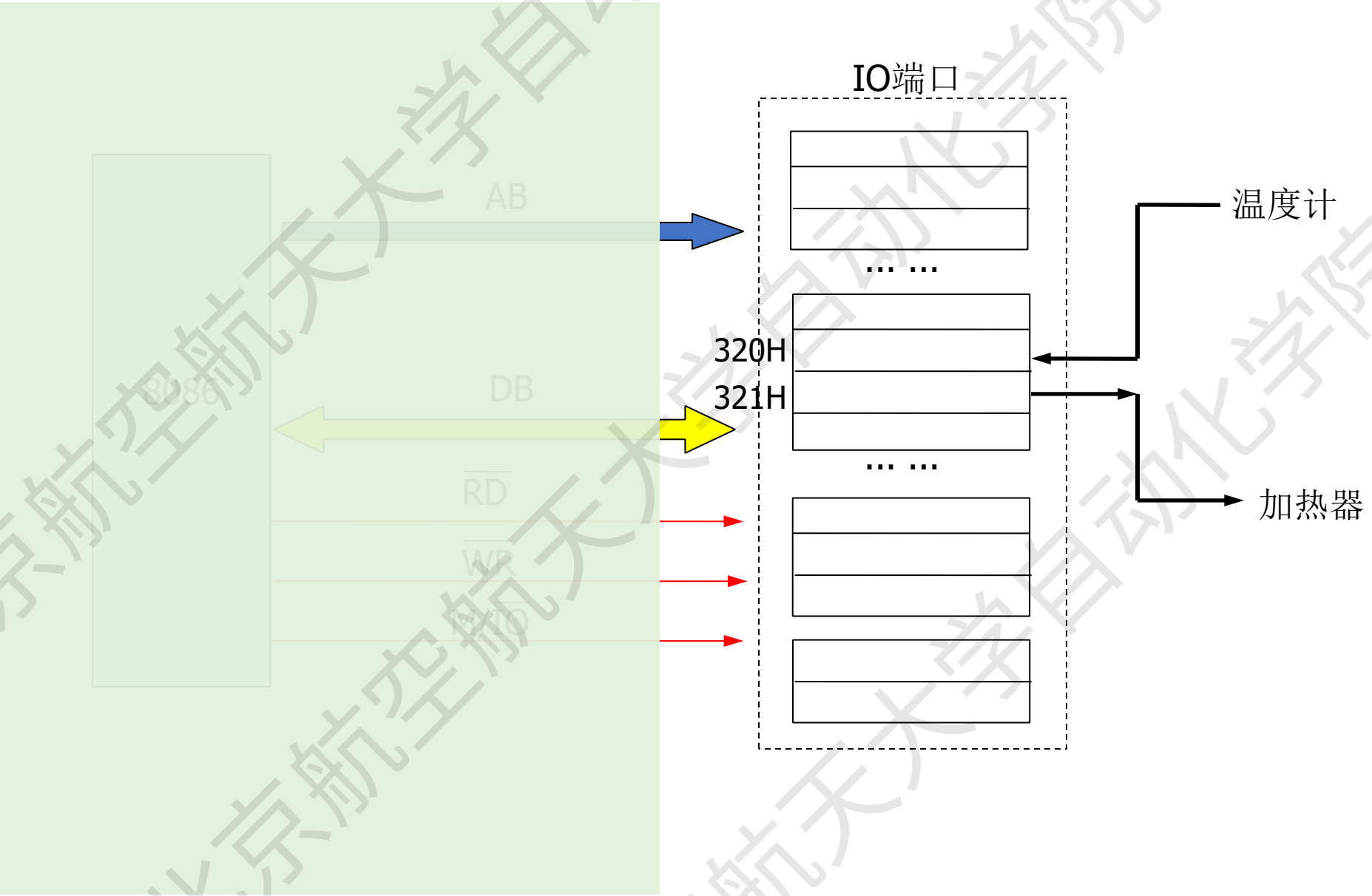
```
        CMP     AL, 60
        JB      FAIL
        CMP     AL, 85
        JA      GOOD
        MOV     AL, 'P'
        JMP     PRINT
FAIL:   MOV     AL, 'F'
        JMP     PRINT
GOOD:   MOV     AL, 'G'
PRINT:  ....
        ....
```

北京航空航天大学自动化学院

例3.91 从温度传感器（端口号320H）读入摄氏温度值，低于100度则打开加热器，等于高于100度则关闭加热器。加热器由端口321H的D0位控制，1为加热器打开，0为关闭。



例3.91 从温度传感器（端口号320H）读入摄氏温度值，低于100度则打开加热器，等于高于100度则关闭加热器。加热器由端口321H的D0位控制，1为加热器打开，0为关闭。



例3.92

在首地址为TABLE的地方放了10个带符号数，要求统计其中正数、负数、零的个数，分别放入PLUS, NEGT, ZERO中。

```
TABLE    DB    01H, 80H, 0F5H, 32H, 86H
         DB    74H, 49H, 0AFH, 25H, 40H
PLUS     DB    0
NEGT     DB    0
ZERO     DB    0
```

北京航空航天大学自动化学院林新

例3.92 在首地址为TABLE的地方放了10个带符号数，要求统计其中正数、负数、零的个数，分别放入PLUS, NEGT, ZERO中。

```
TABLE  DB     01H, 80H, 0F5H, 32H, 86H
       DB     74H, 49H, 0AFH, 25H, 40H
PLUS   DB     0
NEGT   DB     0
ZERO   DB     0
```

北京航空航天大学自动化学院

3. 循环控制指令

- 包括：LOOP（掌握），LOOPE/LOOPZ（了解），LOOPNE/LOOPNZ（了解）
- 控制一个程序段的重复执行，重复次数由CX寄存器的内容决定；
- 机器码都是2个字节，第一个字节是操作码，第二个是8位偏移量。循环的地址偏移在-128~-2之间
- 循环指令不影响任何标志位。

（1）LOOP 循环指令

- 指令格式：LOOP 短标号
- 指令功能：用于重复执行一段代码。
  - 重复次数事先放在CX寄存器中。
  - 每执行一次LOOP指令，CX自动减1，如果CX≠0,则转移到指令中的标号处继续执行；如果**自动减1后CX = 0**则循环结束，执行LOOP指令后面的指令。

例：

```
      XOR   AL, AL
      MOV   CX, 8

AGN:  INC   AL
      LOOP  AGN

AGN1: LOOP  AGN1
```

- 循环体内指令执行次数等于CX中的数值
- 当CX=0时，循环的次数最大，为2^{16}

例3.93 编程实现8种商品价格分别提价7元

```
OLD     DB    83H, 76H, 65H, 84H
        DB    71H, 49H, 62H, 58H
NEW     DB    8 DUP(?)
  …     …
        MOV   CX, 08H
        MOV   BX, 00H
NEXT:   MOV   AL,   OLD[BX]
        ADD   AL, 7
        DAA
        MOV   NEW[BX], AL
        INC   BX
        LOOP  NEXT
```

（2）LOOPE/LOOPZ 相等或结果为0时循环

- 指令格式：LOOPE 标号 （或 LOOPZ 标号）
- 指令功能：这两条指令功能完全相同。每执行一次，CX自动减1，若CX≠0和ZF=1，则转移到指令提供的标号处执行；若自动减1后CX=0或ZF=0，则结束循环，执行LOOPE或LOOPZ后面的执行。

（3）LOOPNE/LOOPNZ 不相等或结果不为0时循环

- 指令格式：LOOPNE 标号 （或 LOOPNZ 标号）
- 指令功能：这两条指令功能完全相同。每执行一次，CX自动减1，若CX≠0和ZF=0，则转移到指令提供的标号处执行；若自动减1后CX=0或ZF=1，则结束循环，执行LOOPE或LOOPZ后面的执行。

练习

编程完成以下任务：

（1）使用将DT1定义的六个无符号数复制到DT2和DT3中；

（2）然后用移位指令将DT2中的每个数据乘以2，DT3中的每个数据乘以4，要求使用循环指令。

（3）编程使得DT4中的每个数据都是DT1对应数据的10倍。

```
DATA  SEGMENT
DT1          DW    12H,1FH,35H,33H,2AH,26H
DT2          DW    6 DUP(?)
DT3          DW    6 DUP(?)
DT4          DW    6 DUP(?)
DATA  ENDS
```

编程完成以下任务：

（1）使用将DT1定义的六个无符号数复制到DT2和DT3中；

（2）然后用移位指令将DT2中的每个数据乘以2，DT3中的每个数据乘以4，要求使用循环指令。

（3）编程使得DT4中的每个数据都是DT1对应数据的10倍。

```
DATA    SEGMENT
DT1     DW  12H,1FH,35H,33H,2AH,26H
DT2     DW  6 DUP(?)
DT3     DW  6 DUP(?)
DT4     DW  6 DUP(?)
DATA    ENDS
```

北京航空航天大学自动化学院林新

4. 中断指令

- 中断的概念
- 中断源
- 中断矢量

北京航空航天大学自动化学院林新

中断指令

(1) INT n

(2) INTO

(3) IRET

北京航空航天大学自动化学院林新

3.3.6 处理器控制指令

1.标志操作指令

STC,CLC,CMC; STD,CLD; STI,CLI

2.外部同步指令

3.停机和空操作指令

HOLD，NOP

北京航空航天大学自动化学院林新

指令助记符	操作	指令名称
CLC	CF←0	进位标志清 0
CMC	CF←$\overline{CF}$	进位标志求反
STC	CF←1	进位标志置 1
CLD	DF←0	方向标志位清 0
STD	DF←1	方向标志位置 1
CLI	IF←0	中断标志位清 0
STI	IF←1	中断标志位置 1

第三章 总结

- 寻址方式：8种
- 常用指令：
  - 数据传送指令：
    掌握MOV,PUSH,POP,XCHG,XLAT,IN/OUT,LEA
  - 算术运算指令：
    掌握ADD,ADC,INC, SUB,SBB,DEC,NEG,CMP;
    了解AAA,DAA, AAS,DAS,MUL/IMUL,DIV/IDIV
  - 逻辑运算和移位指令：全部
  - 字符串处理指令：
    掌握MOVS,CMPS,了解SCAS,LODS,STOS
  - 控制转移指令：
    掌握JMP,CALL/RET,LOOP，掌握JZ/JA等条件转移、LOOPE/LOOPZ,LOOPNE/LOOPNZ

北京航空航天大学自动化学院林新

习题6 （P110）指出错误的指令及其错误的地方

1)MOV DL, AX

2)MOV 8650H，AX

3)MOV DS, 0200H

4)MOV [BX], [1200]

5)MOV IP, 0FFH

6)MOV [BX+SI+3],IP

7)MOV AX,[BX+BP]

8)MOV AL, ES:[BP]

9)MOV DL,[SI+DI]

10)MOV AX, OFFSET 0A20H

11)MOV AL, OFFSET TABLE

12)XCHG AL, 50H

13)IN BL, 05H

14)OUT AL, 0FFEH

习题11 设数据段定义如下：

```
DATA    SEGMENT
STR     DB ' ThePersonalComputer&TV'

DATA    ENDS
```

编程完成以下功能：

1）将字串传送到**附加段**中偏移量为GET_CHAR开始的内存单元中；

2）比较该字串与'TheComputer'是否相同，相同则AL置1，否则置0（提示：可在数据段中定义字符串ST1保存'TheComputer'）；

3）检查字串中是否有‘&’字符，若有则用空格（20H）替换

北京航空航天大学自动化学院林新

习题11 设数据段定义如下：

```
DATA    SEGMENT
STR     DB ' ThePersonalComputer&TV'


DATA    ENDS
```

编程完成以下功能：

1）将字串传送到**附加段**中偏移量为GET_CHAR开始的内存单元中；

2）比较该字串与'TheComputer'是否相同，相同则AL置1，否则置0（提示：可在数据段中定义字符串ST1保存'TheComputer'）；

3）检查字串中是否有‘&’字符，若有则用空格（20H）替换

习题11 设数据段定义如下：

```
DATA    SEGMENT
STR     DB ' ThePersonalComputer&TV'


DATA    ENDS
```

编程完成以下功能：

1）将字串传送到**附加段**中偏移量为GET_CHAR开始的内存单元中；

2）比较该字串与'TheComputer'是否相同，相同则AL置1，否则置0（提示：可在数据段中定义字符串ST1保存'TheComputer'）；

3）检查字串中是否有‘&’字符，若有则用空格（20H）替换

习题12 编程将AL寄存器中的内容以相反的次序传送到DL寄存器中，并要求AL中的内容不被破坏，并统计DL中1的个数。（提示，先循环左移AL，检查CF是否为1，再通过右移指令移入DL）

北京航空航天大学自动化学院