第3章

指令系统

主要内容：

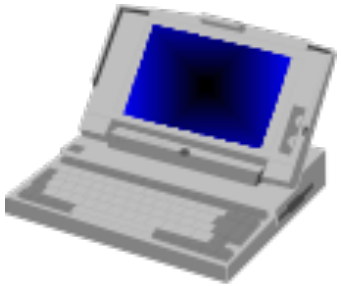
- 指令系统的一般概念
- 对操作数的寻址方式
- 六大类指令的操作原理：

操作码的含义

指令对操作数的要求

指令执行的结果

§3.1 概述



一、指令与指令系统

指令：

控制计算机完成某种操作的命令

指令系统：

处理器所能识别的所有指令的集合

指令的兼容性：

同一系列机的指令都是兼容的。

二、指令格式

指令的操作对象

指令中应包含的信息：

执行的操作

运算数据的来源

运算结果的去向

操作码　[操作数]，[操作数]

目标操作数　源操作数

①运算结果的去向

②另一运算数据的来源

参加操作的数据或数据存放的地址

执行何种操作

指令格式：

零操作数指令：　操作码

单操作数指令：　操作码　操作数

双操作数指令：　操作码　操作数，操作数

多操作数指令：　三操作数及以上

三、指令中的操作数

立即数 → 表征参加操作的数据本身

寄存器
存储器
} 表征数据存放的地址

立即数操作数

- 立即数只能作为源操作数。其本身是参加操作的数据，可以是8位或16位。

例： MOV AX，1234H

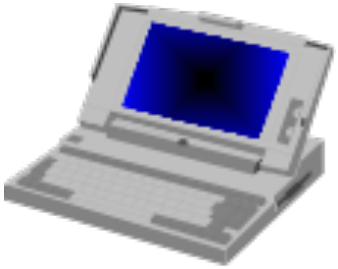
MOV BL，22H

- 立即数无法作为目标操作数
- 立即数可以是无符号或带符号数，其数值应在可取值范围内。

寄存器操作数：

- 参加运算的数存放在指令给出的寄存器中，可以是16位或8位。
- 例：
- MOV AX，BX
- MOV DL，CH
- AX BX CX DX SI DI BP BX CS DS ES SS可作为寄存器操作数，可源可目标。
- 段寄存器不能直接用立即数赋初值。

存储器操作数



- 参加运算的数存放在存储器的某一个或某两个单元中

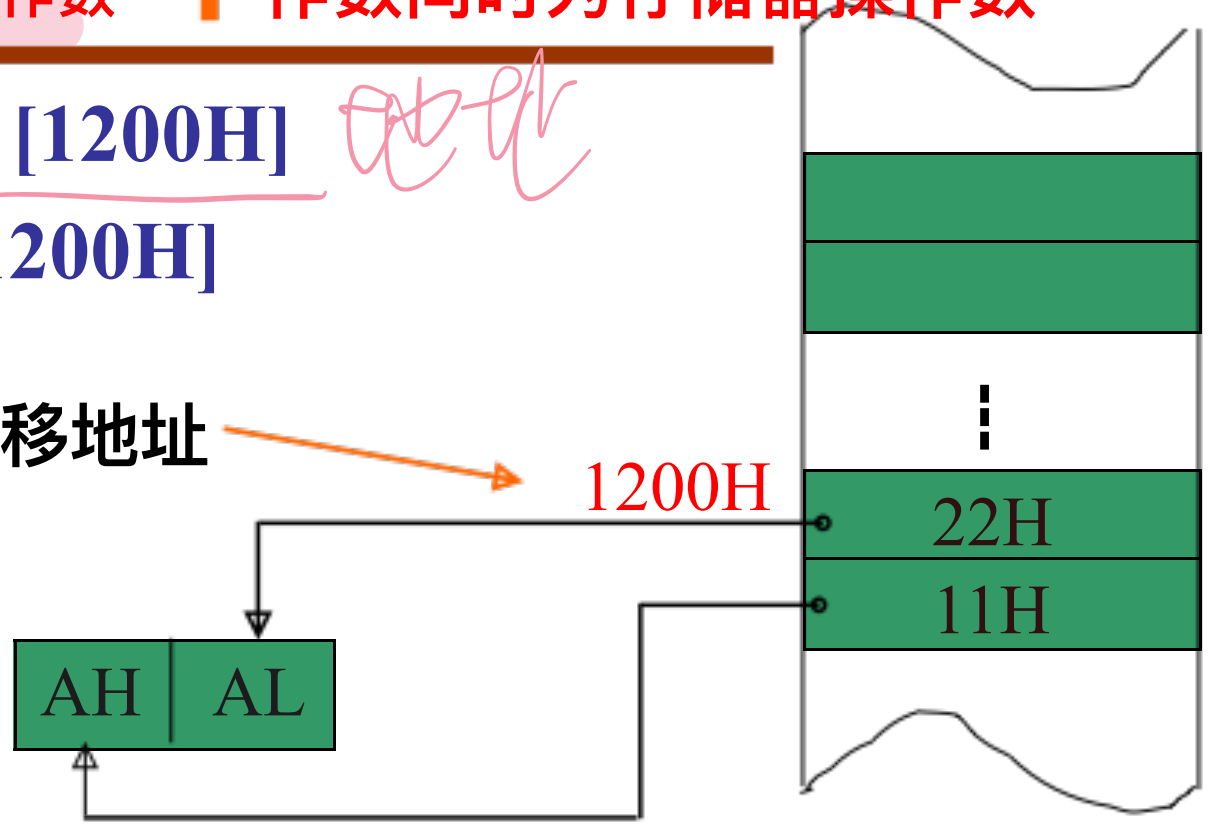
- 表现形式：[　　]

立即数或寄存器操作数

- 可作为源操作数也可作为目标操作数，但不能源操作数和目标操作数同时为存储器操作数

- 例： MOV AX，[1200H]

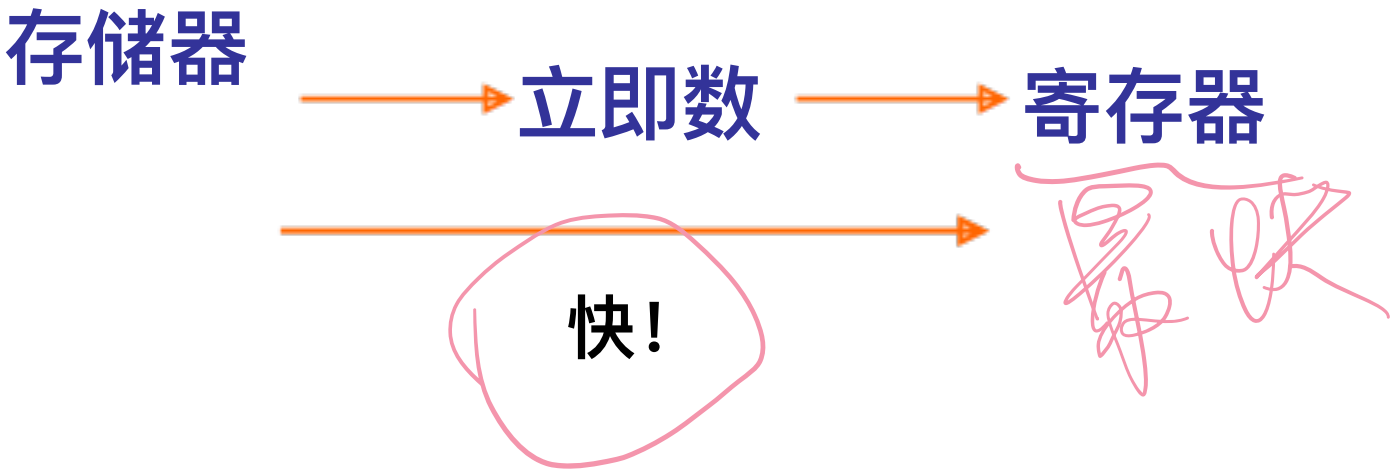
  MOV AL，[1200H]



四、指令的执行速度

- 指令的字长影响指令的执行速度

  （指令执行时间例子见第二版P97及附录C.1）
- 对不同的操作数，指令执行的时间不同：



五、指令字长与机器字长

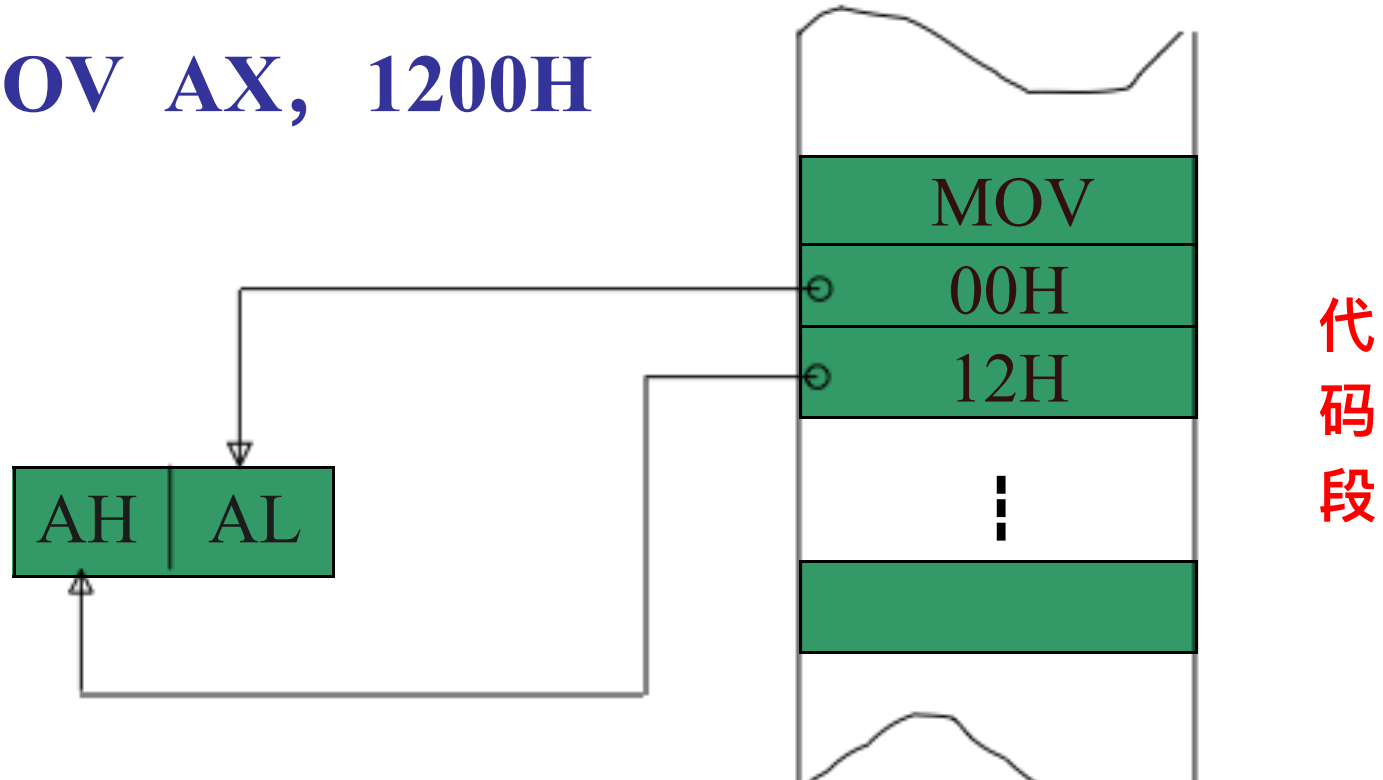
- 指令字长：
  - 由操作码的长度、操作数地址长度、操作数个数决定；
- 机器字长：
  - 计算机能够直接处理的二进制数的位数。

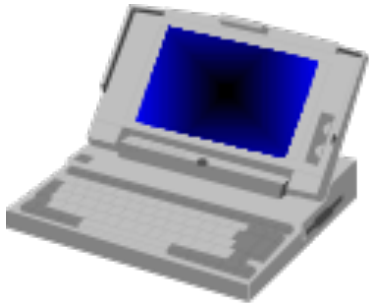
§3.2 寻址方式

寻找操作数所在地址的方法

一、立即寻址

- 指令中的源操作数是**立即数**，即源操作数是参加操作的数据本身
- 例：MOV AX，1200H





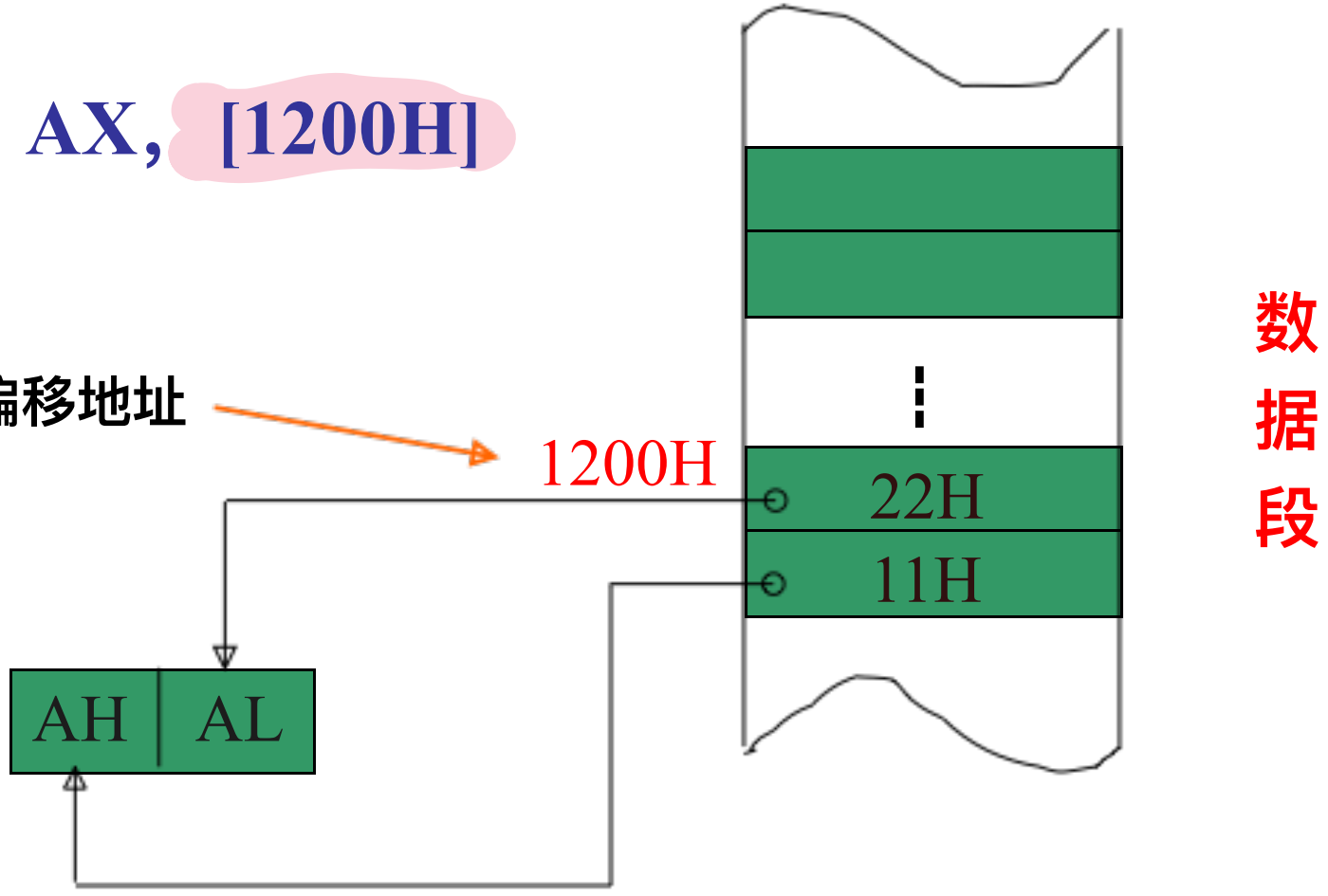
二、寄存器寻址

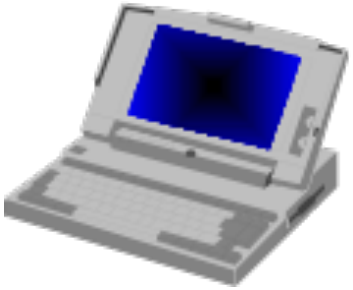
- 参加操作的操作数在CPU的**通用寄存器**中。
- 例：MOV AX，BX

三、直接寻址

- 指令中直接给出操作数的偏移地址
- 例：MOV AX, [1200H]

•MOV AX，1200H





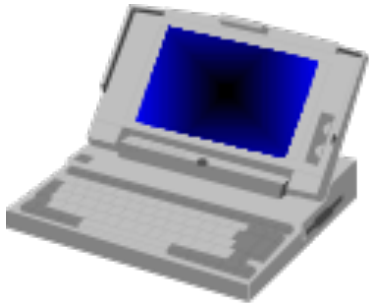
直接寻址

- 直接寻址方式下，操作数的段地址默认为数据段，但允许段重设，即由指令定义段。

- 例：MOV AX，ES：[1200H]

存储器操作数字长未确定，其字长取决于指令中另一个存储器操作数

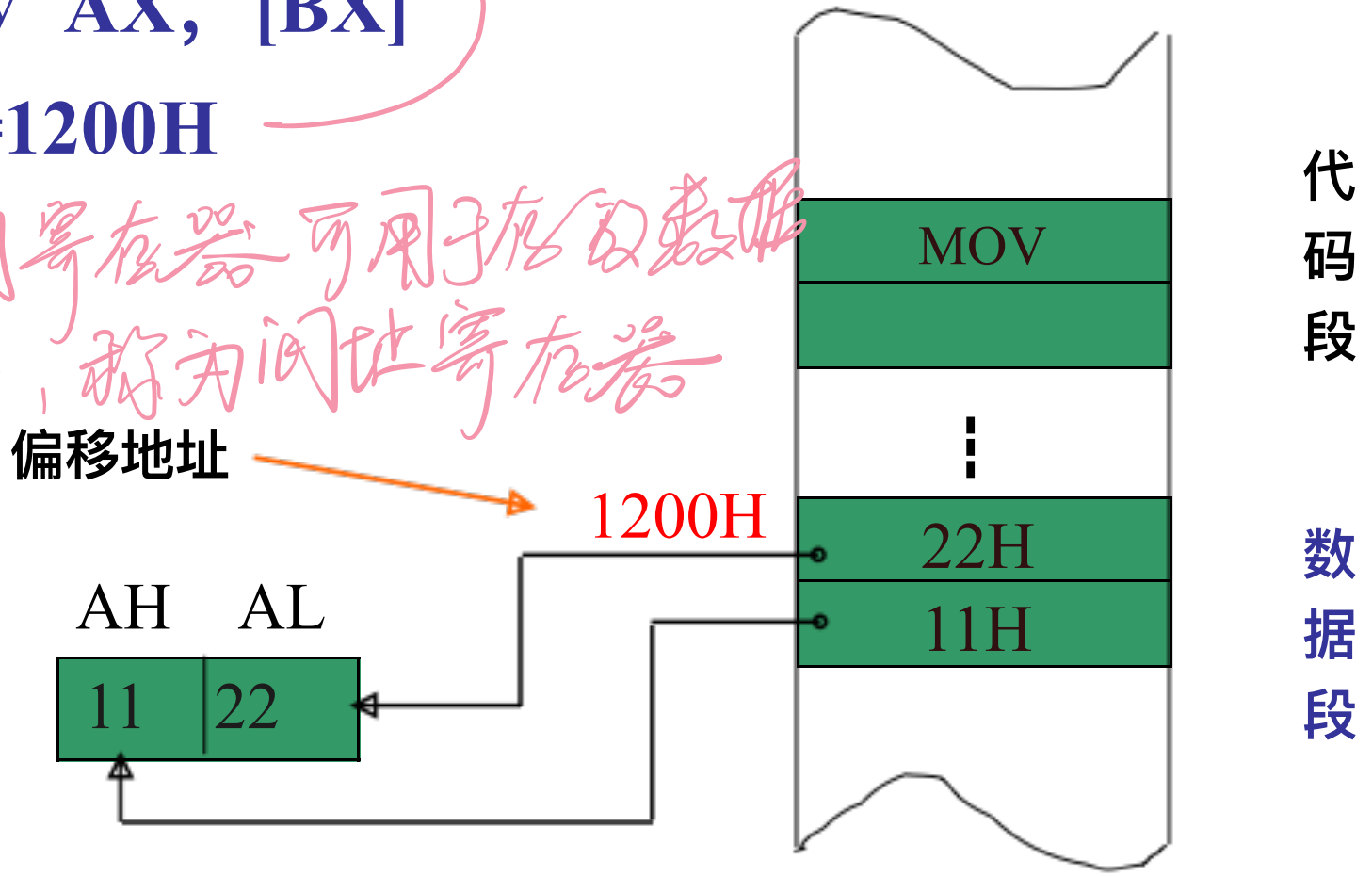
四、寄存器间接寻址

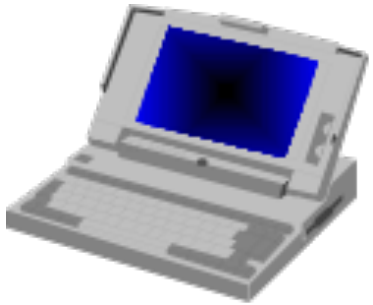


- 参与操作的操作数存放在内存中，其偏移地址为指令中的寄存器的内容。

- 例：MOV AX，[BX]

设BX=1200H



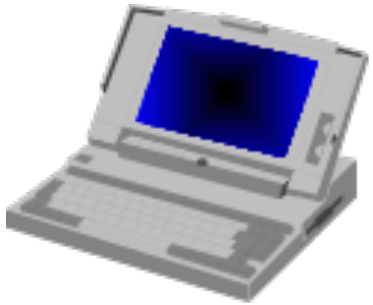


寄存器间接寻址

寄存器间接寻址

基址寻址（间址寄存器为基址寄存器BX，BP）

变址寻址（间址寄存器为变址寄存器SI，DI）

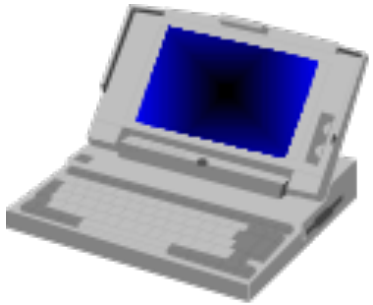


寄存器间接寻址

- 由寄存器间接给出操作数的偏移地址；
- 存放偏移地址的寄存器称为**间址寄存器**，它们是：**BX，BP，SI，DI**
- 操作数的段地址（数据处于哪个段）取决于选择哪一个间址寄存器：

BX，SI，DI → 默认在数据段

BP → 默认在堆栈段



五、寄存器相对寻址

- 操作数的偏移地址为寄存器的内容加上一个位移量

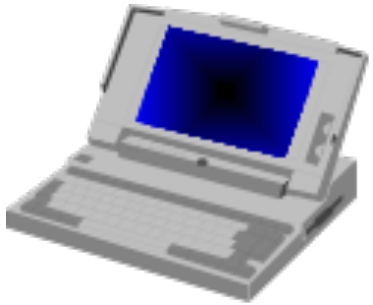
- 例：

  MOV AX，[BX+DATA]

  设：DS=2000H，BX=0220H，DATA=05H

  则：AX=[20225H]

BX—数据段
BP—堆栈段



六、基址、变址寻址

- 操作数的偏移地址为一个基址寄存器的内容加上一个变址寄存器的内容
- 操作数的段地址由选择的基址寄存器决定
- 例：MOV AX，[SI+BX]

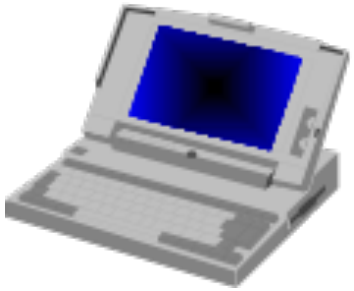
七、基址、变址、相对寻址

- 操作数的偏移地址为一个基址寄存器的内容加上一个变址寄存器的内容，再加上一个位移量。
- 操作数的段地址由选择的基址寄存器决定
- 例：MOV AX，[BP+SI+DATA]

[BP][SI] DATA

基址选BP，默认操作在堆栈段

八、隐含寻址



- 指令隐含了的一个或两个操作数的地址，即操作数在默认的地址中

- 例： MUL BL

指令的执行：AL×BL → AX

§3.3 8086指令系统

- 指令（操作）码的含义
- 指令对操作数的要求
- 指令对标志位的影响
- 指令的功能

从功能上包括六大类：

数据传送

算术运算

逻辑运算和移位

串操作

程序控制

处理器控制

数据传送指令

- 通用数据传送
- 输入输出
- 地址传送
- 标志位操作

一、通用数据传送

一般数据传送指令
堆栈操作指令
交换指令
查表转换指令
字位扩展指令

特点：

该类指令的执行对标志位不产生影响

1. 一般数据传送指令

- 一般数据传送指令 **MOV**

- 格式：MOV dest，src

- 功能：src → dest

- 例： MOV AL，BL

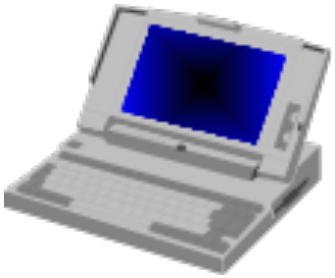
MEM/REG

IMD/MEM/REG

一般数据传送指令

注：

- 两操作数字长必须相同；
- 两操作数不允许同时为存储器操作数；
- 两操作数不允许同时为段寄存器；
- 当源操作数是立即数时，目标操作数不能是段寄存器；
- IP和CS不作为目标操作数，FLAGS一般也不作为操作数在指令中出现。

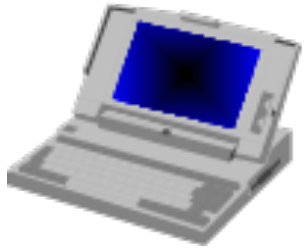


一般数据传送指令

判断下列指令的正确性：

- MOV AL，BX
- MOV AX，[SI]05H
- MOV [BX][BP]，BX
- MOV DS，1000H
- MOV DX，09H
- MOV [1200]，[SI]

一般数据传送指令应用例



将(*) 的ASCII码2AH送入内存1000H开始的100个单元中：

```
        MOV  DX,  1000H
    MOV  CX,  64H
    MOV  AL,  2AH
AGAIN：MOV  [DX],  AL
    INC  DX          ；DX+1
        DEC  CX          ；CX-1
    JNZ  AGAIN       ；CX≠0则继续
    HLT
```

上段程序在代码段中的存放形式

設CS=109EH，IP=0100H，则各条指令存放地址如下：

CS：IP 机器指令　汇编指令

109E：0100　B80010　MOV DX，1000H

109E：0103　.　MOV CX，64H

109E：0105　.　MOV AL，2AH

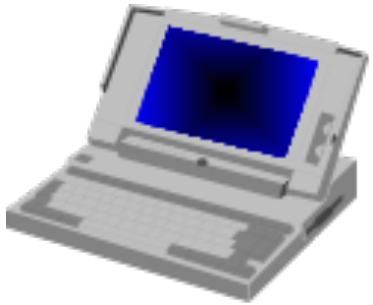
109E：0107　.　MOV [DX]，AL

109E：0109　INC DX

109E：010A　DEC CX

109E：010B　JNZ 0107H

109E：010D　HLT



数据段中的分布

送上2AH后数据段中相应存储单元的内容改变如下：

DS：1000 2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A

DS：1010 2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A

DS：1020 2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A

DS：1030 2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A

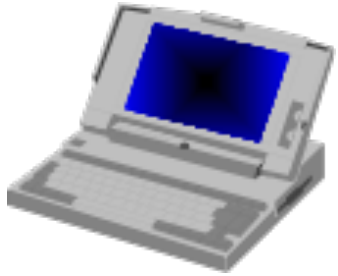
DS：1040 2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A

DS：1050 2A 2A 2A 2A 2A 2A 2A 2A-2A 2A 2A 2A 2A 2A 2A 2A

DS：1060 2A 2A 2A 2A 00 00 00 00 00 00 00 00 00 00 00 00

偏移地址[DX]

2. 堆栈操作指令



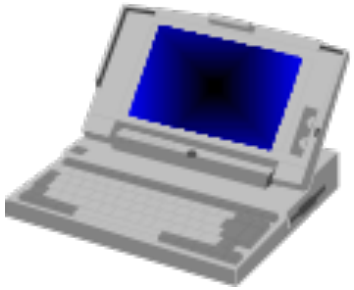
掌握：

- 有关堆栈的概念 → 栈顶、栈首、栈底
- 堆栈指令的操作原理

执行过程，执行结果

堆栈操作的原则

- 先进后出
- 以字为单位



堆栈操作指令

- 压栈指令 PUSH

  格式： PUSH OPRD

- 出栈指令 POP

  格式：POP OPRD

MEM/REG

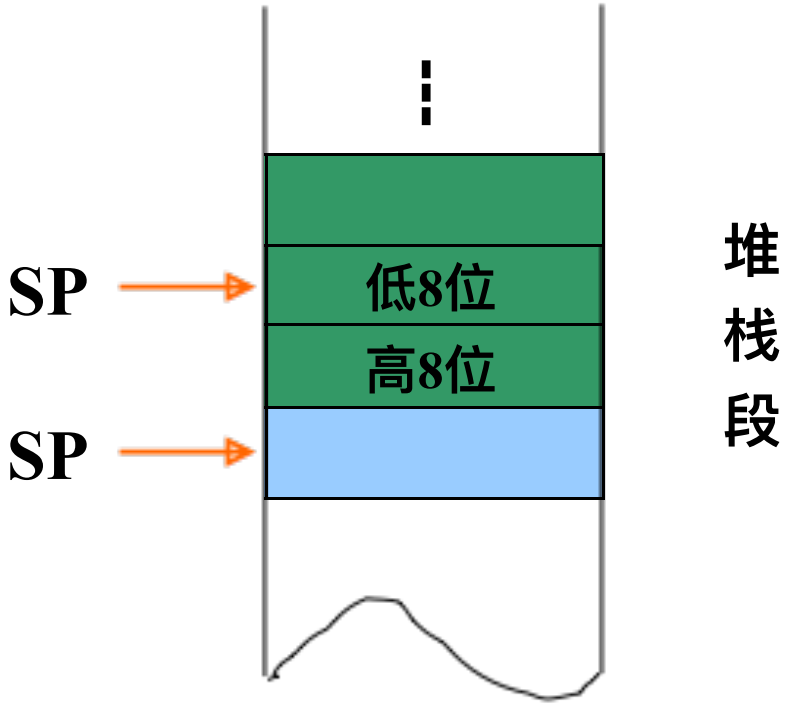
压栈指令 PUSH OPRD

- 指令执行过程：

(SP）←（SP）-2

（SP）+1←操作数高字节

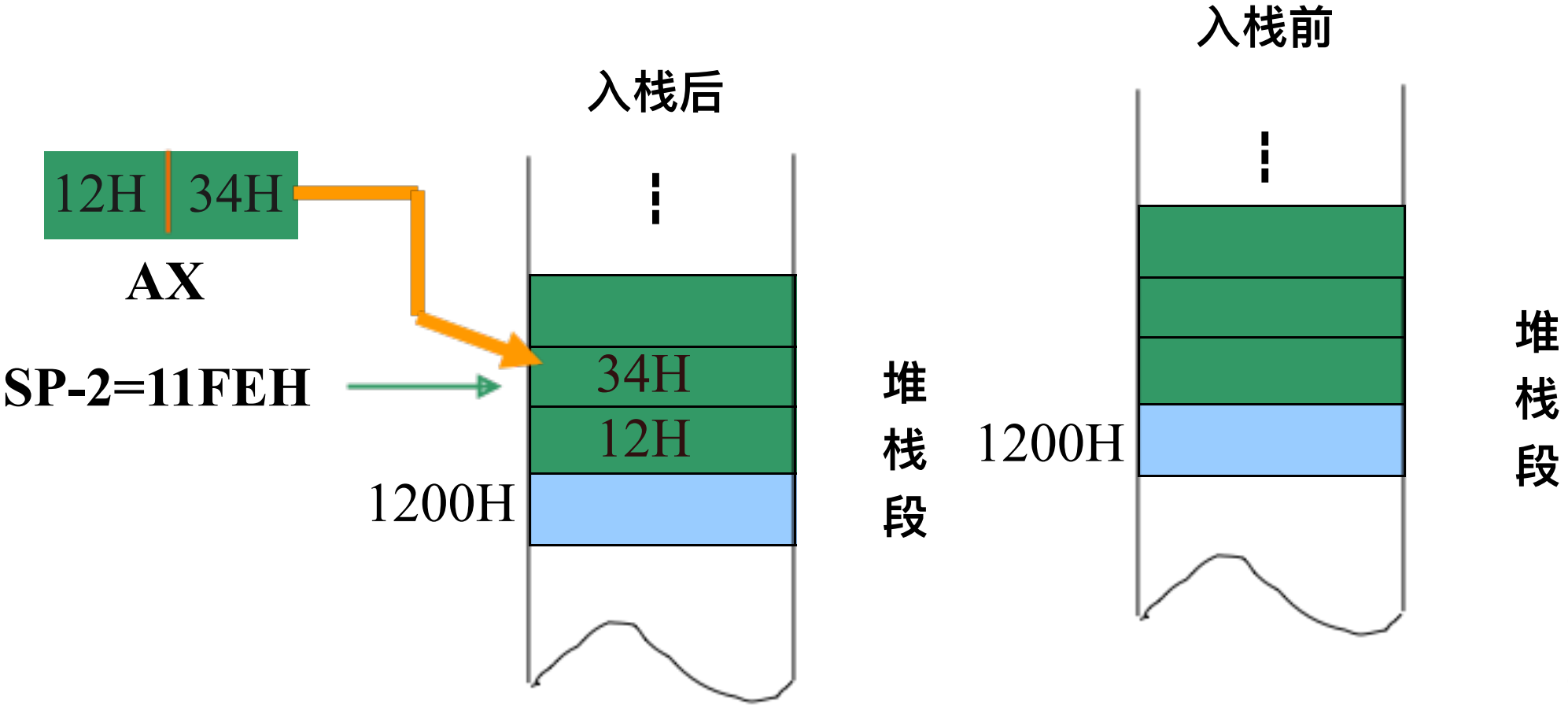
（SP）←操作数低字节



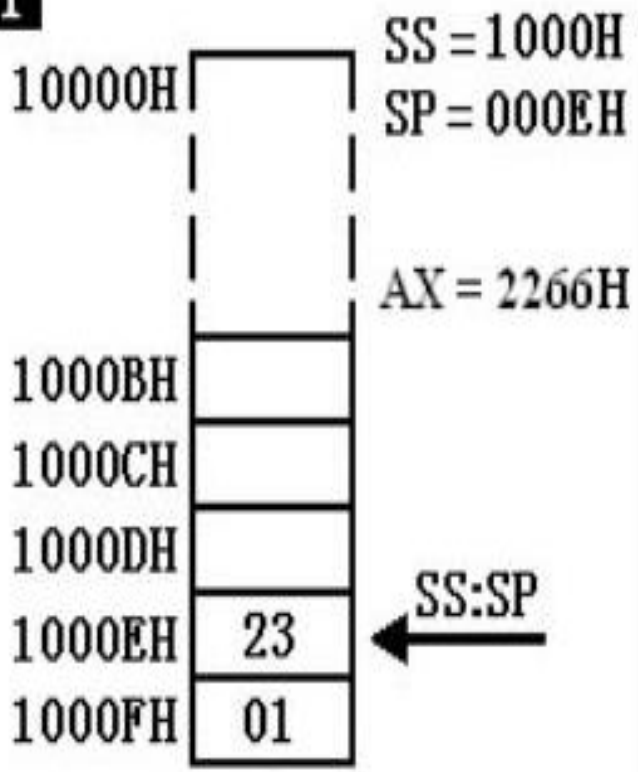
压栈指令的操作

设AX=1234H，SP=1200H

执行 PUSH AX 指令后堆栈区的状态：



1



当前的状态:

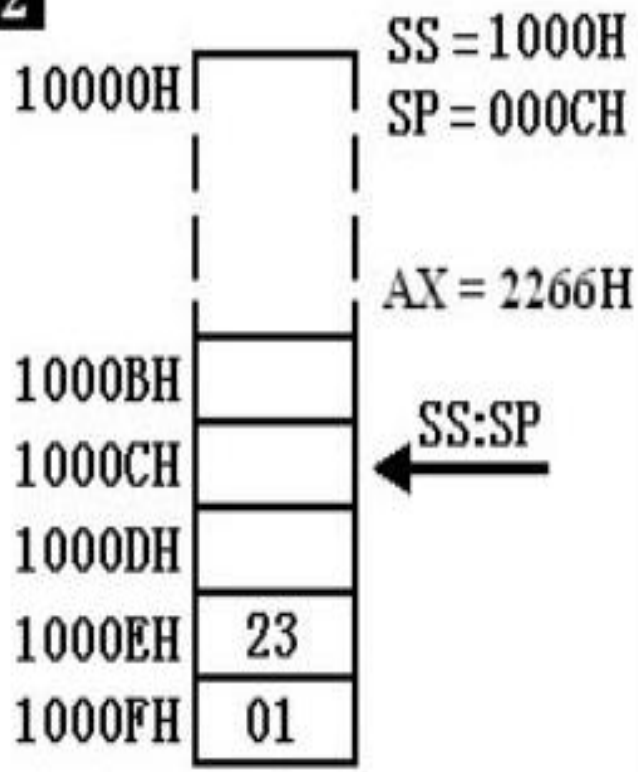
SS中的内容: 1000H

SP中的内容: 000EH

则栈顶的段地址为 1000:000E, 即1000EH。

ax中的内容: 2266H

2



CPU执行 push ax,

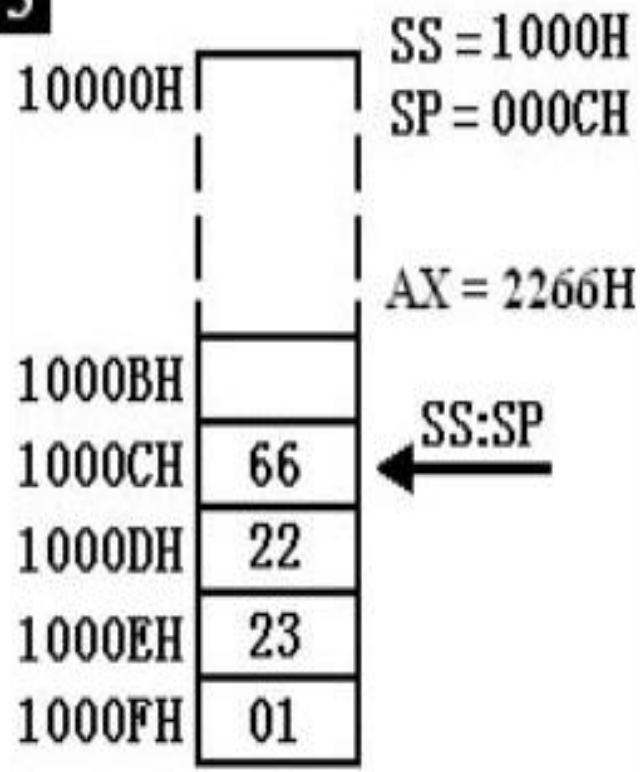
第一步: SP=SP-2

SP中的内容变为: 000CH

SS:SP指向 1000:000C,

栈顶的地址变为 1000:000C 即 1000CH。

3



CPU执行 push ax,

第二步:

将 ax 中的数据送入 SS:SP 指向的内存单元处。

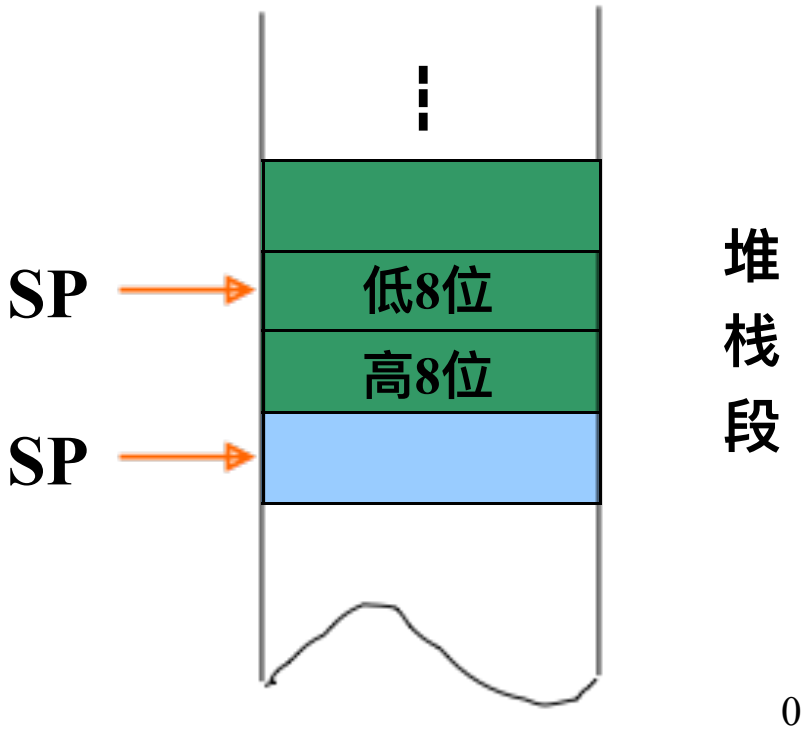
出栈指令POP OPRD

- 指令执行过程：

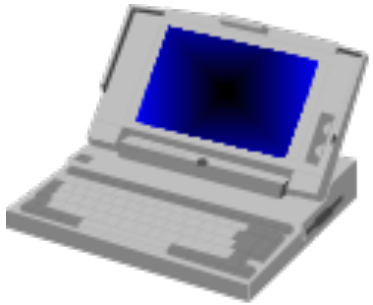
SP ⟶ 操作数低字节

SP+1 ⟶ 操作数高字节

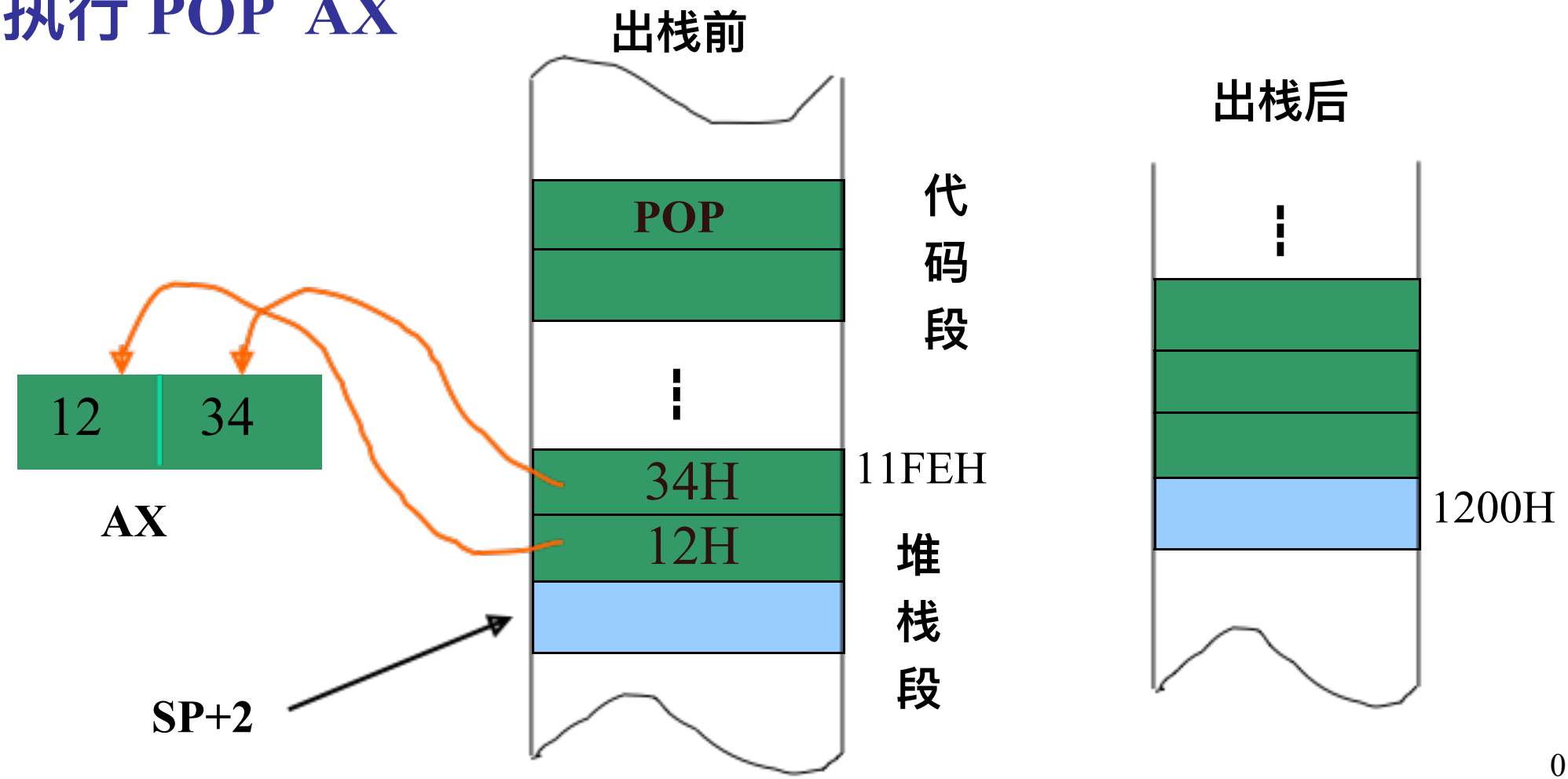
SP ← SP+2



出栈指令的操作

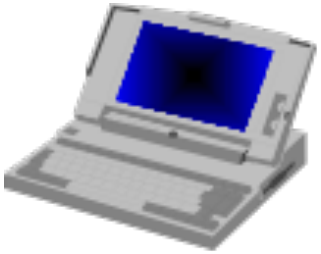


执行 POP AX



堆栈操作指令说明

- 指令的操作数必须是16位的；
- 操作数可以是寄存器或存储器两单元，但不能是立即数；
- 不能从栈顶弹出一个字给CS；
- PUSH和POP指令在程序中一般成对出现；
- PUSH指令的操作方向是从高地址向低地址，而POP指令的操作正好相反。



堆栈操作指令例

- PUSH AX
- PUSH BX
- PUSH WORD PTR[BX]

⋮

- POP WORD PTR[BX]
- POP AX
- POP BX

如此，会使AX和BX的内容互换

3. 交换指令

- 格式：

  XCHG MEM/REG，MEM/REG

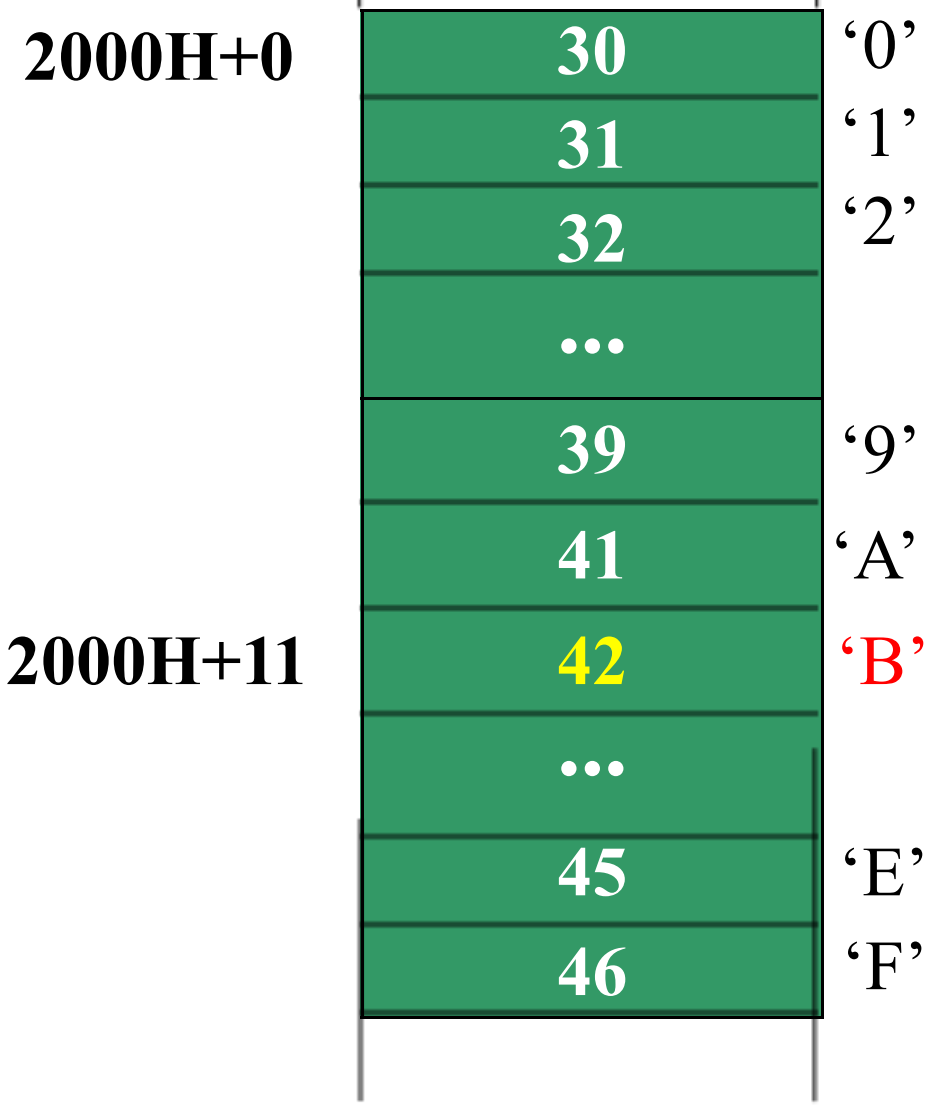
- 注：
  - 两操作数必须有一个是寄存器操作数
  - 不允许使用段寄存器。
- 例：
  - XCHG AX，BX
  - XCHG [2000]，CL

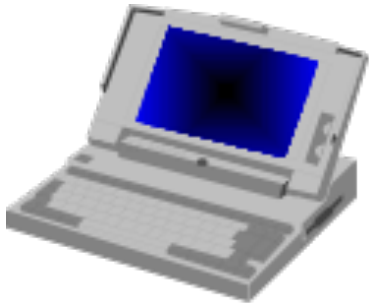
查表指令例

数据段中存放有一张 ASCII 码 转 换 表，

设 首 地 址 为2000H，

现欲查出表中第11个代码的ASCII码





4. 查表指令

- 格式： 隐含寻址（无操作数）

  XLAT

- 说明：

  用BX的内容代表表格首地址，AL内容为表内位移量，BX+AL得到要查找元素的偏移地址

- 操作：

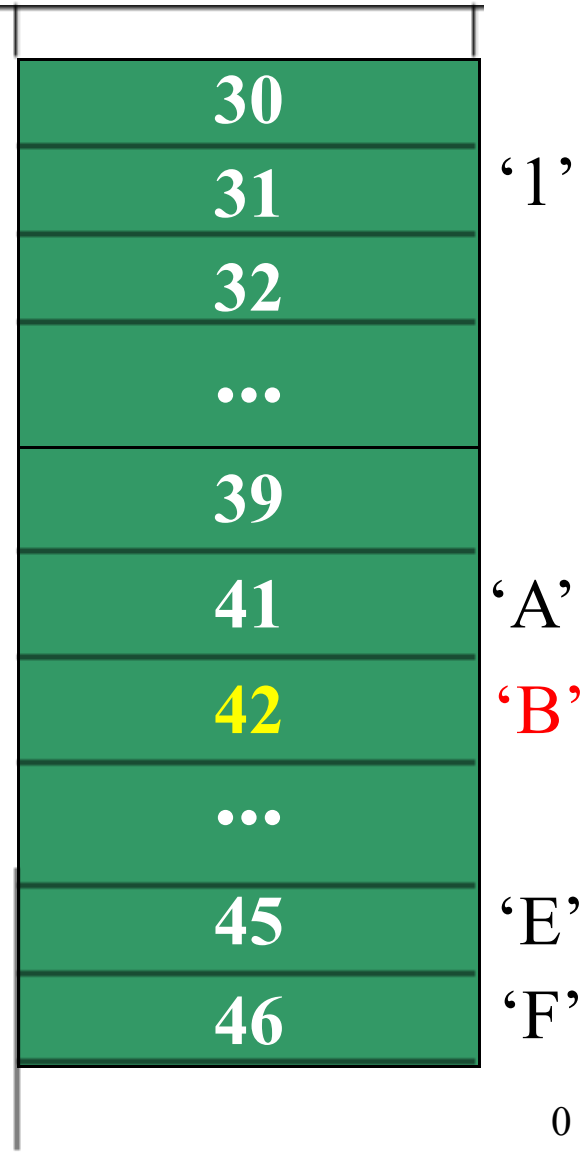
  将BX+AL所指单元的内容送AL

查表指令例

可用如下指令实现：

```
MOV  BX，2000H ；BX←表首地址
MOV  AL，0BH   ；AL←序号
XLAT           ；查表转换
```

执行后得到：AL = 42H



5. 字位扩展指令

- 将符号数的符号位扩展到高位；
- 指令为零操作数指令，采用**隐含寻址**，隐含的操作数为**AX及AX，DX**
- **无符号数的扩展规则为在高位补0**

字节到字的扩展指令

- 格式：**CBW**
- 操作：**将AL内容扩展到AX**
- 规则：
  - 若最高位=1，则执行后AH=FFH
  - 若最高位=0，则执行后AH=00H

字到双字的扩展指令

- 格式：

  CWD

- 操作：将AX内容扩展到DX AX

- 规则：
  - 若最高位=1，则执行后DX=FFFFH
  - 若最高位=0，则执行后DX=0000H

字位扩展指令例

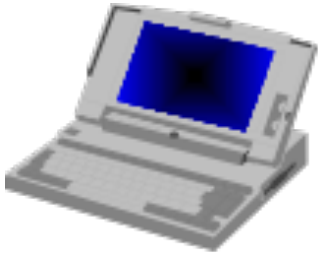
判断以下指令执行结果：

```
MOV  AL，44H
CBW

MOV  AX，0AFDEH
CWD

MOV  AL，86H
CBW
```

二、输入输出指令



掌握：

指令的格式及操作

指令的两种寻址方式

指令对操作数的要求

- 专门面向I/O端口操作的指令
- 指令格式：

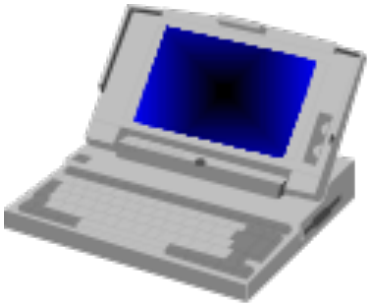
输入指令： IN acc，PORT

输出指令： OUT PORT，acc

AX，AL

端口地址

指令寻址方式



输入指令： IN acc，PORT

输出指令： OUT PORT，acc

I/O指令例

直接寻址

直接给出8位端口地址，可寻址256个端口

- IN AX，80H
- MOV DX，2400H
- IN AL，DX
- OUT DX，AX

间接寻址

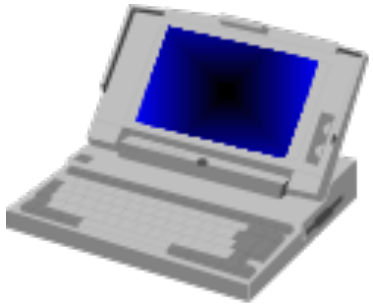
16位端口地址由DX指定，可寻址64K个端口

- OUT 35H，AL
- OUT AL，35H

格式错误 ×

三、地址传送指令

- 取偏移地址指令LEA
- *LDS指令
- *LES指令



取偏移地址指令LEA

- 操作：
  - 将变量的16位偏移地址取出送目标寄存器
- 格式：

  LEA REG，MEM

- 指令要求：
  - 源操作数必须是一个存储器操作数，目标操作数通常是间址寄存器。

LEA指令

- 比较下列指令：

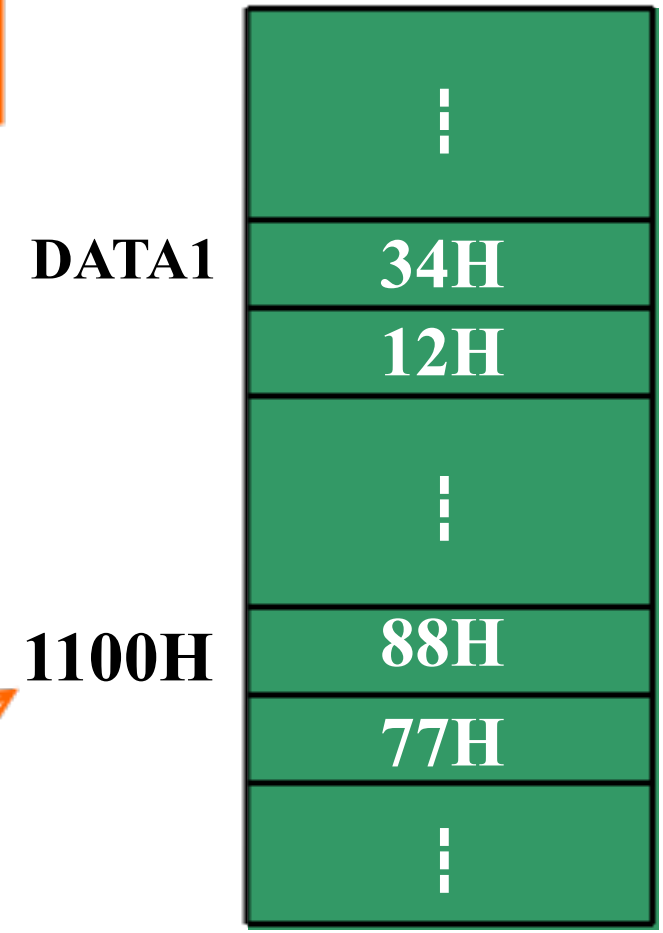
符号
地址

MOV SI，DATA1

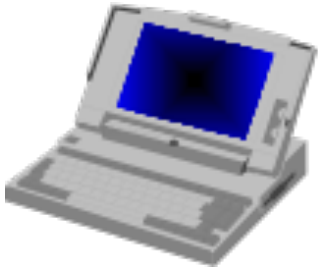
LEA SI，DATA1

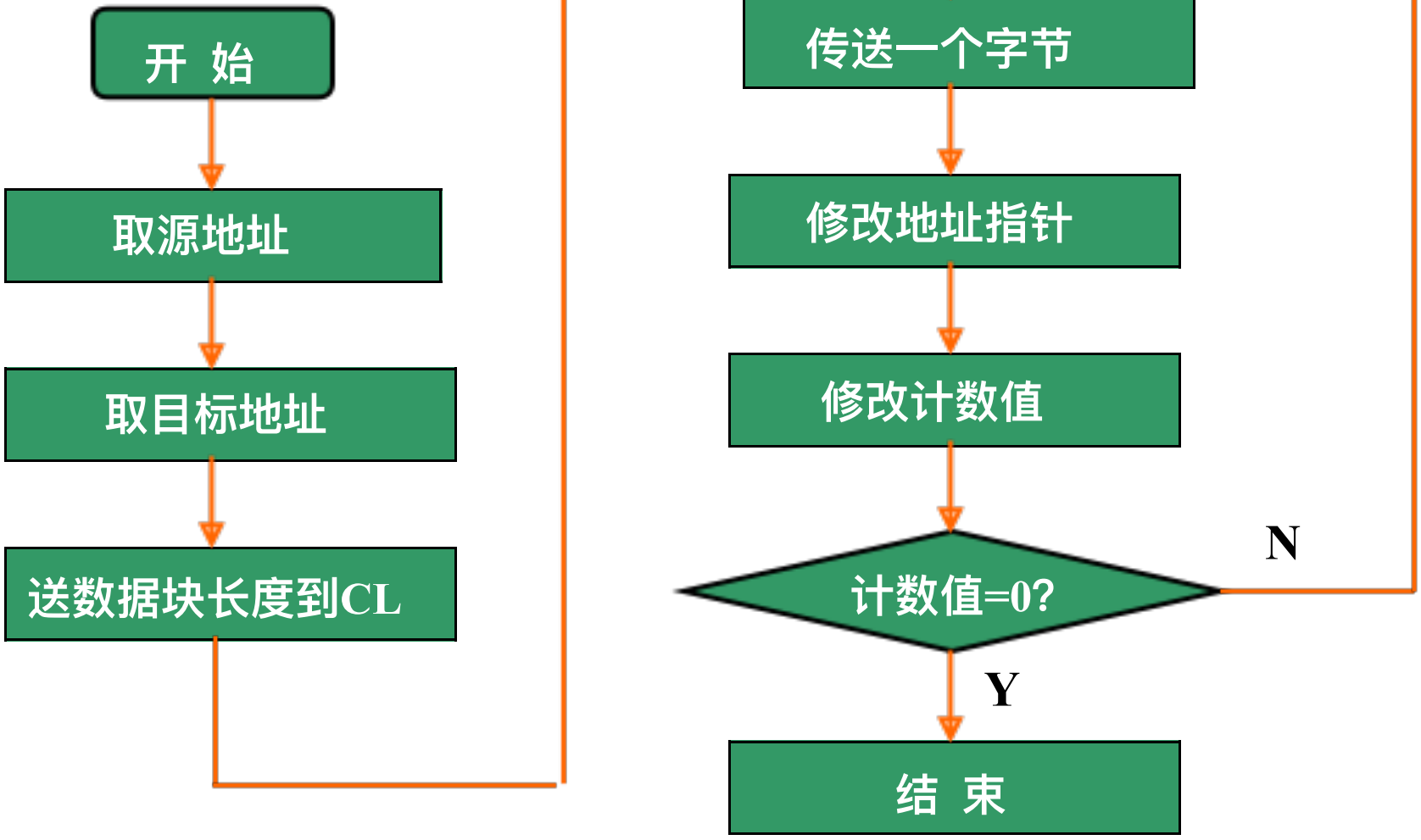
MOV BX，[BX]

LEA BX，[BX]

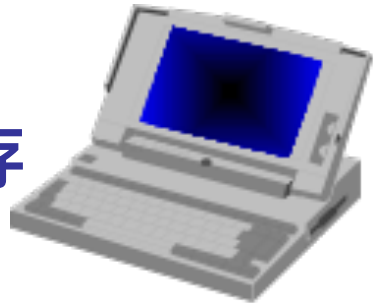


- 将数据段中首地址为MEM1 的50个字节的数据传送到同一逻辑段首地址为MEM2的区域存放。编写相应的程序段 。

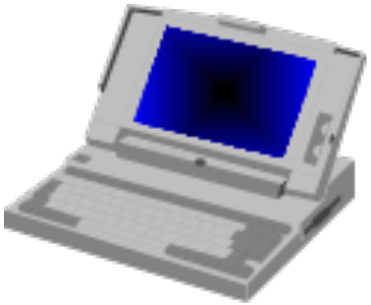




- 将数据段中首地址为MEM1 的50个字节的数据传送到同一逻辑段首地址为MEM2的区域存放。编写相应的程序段 。



```
        LEA  SI，MEM1
        LEA  DI，MEM2
        MOV  CL，50
NEXT：  MOV  AL，[SI]
        MOV  [DI]，AL
        INC   SI
        INC   DI
        DEC   CL
        JNZ   NEXT
        HLT
```



四、标志位操作指令

LAHF

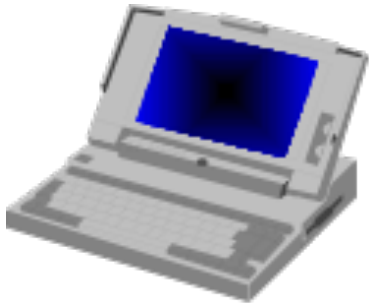
SAHF

隐含操作数AH

PUSHF

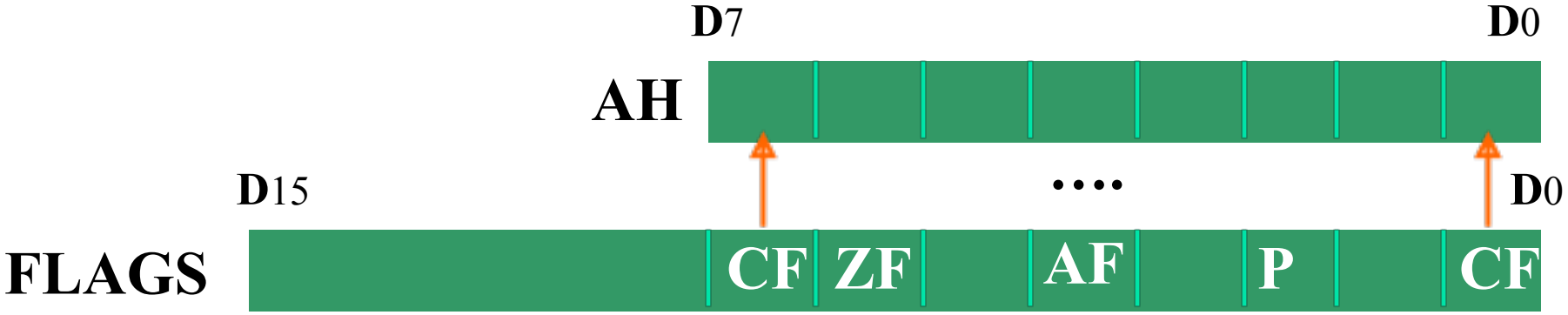
POPF

隐含操作数FLAGS



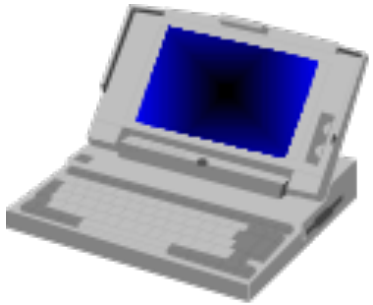
1. LAHF，SAHF

- 指令格式：LAHF
- 操作：将FLAGS的低8位装入AH



- SAHF

执行与LAHF相反的操作

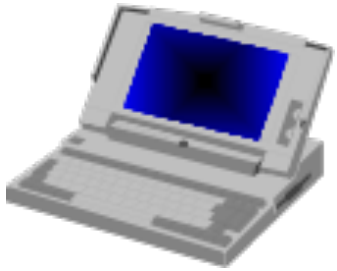


2. PUSHF，POPF

- 针对FLAGS的堆栈操作指令

将标志寄存器压栈或从堆栈弹出

算术运算类指令



- 加法运算指令
- 减法运算指令
- 乘法指令
- 除法指令

算术运算指令的执行大多对状态标志位会产生影响

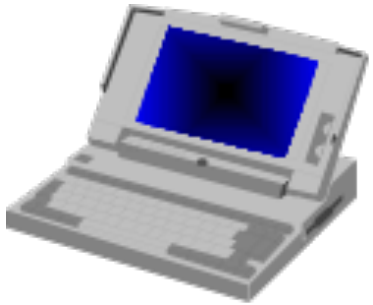
一、加法指令

普通加法指令ADD

带进位位的加法指令ADC

加1指令INC

加法指令对操作数的要求与MOV指令相同



1. ADD指令

- 格式：

  ADD OPRD1，OPRD2

- 操作：

  OPRD1+OPRD2 → OPRD1

ADD指令的执行对全部6个状态标志位都产生影响

ADD指令例

MOV AL，78H

ADD AL，99H　　指令执行后6个状态标志位的状态

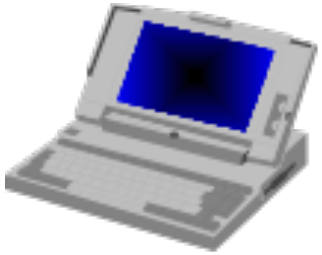
```
  01111000
+ 10011001
-----------
1 00010001
```

标志位状态：　CF= 1　　SF= 0

AF= 1　　ZF= 0

PF= 1　　OF= 0

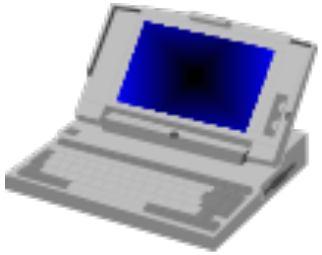
2. ADC指令



- 指令格式、对操作数的要求、对标志位的影响与ADD指令完全一样
- 指令的操作： ADC OPRD1，OPRD2

  OPRD1+OPRD2+CF → OPRD1
- ADC指令多用于多字节数相加，使用前要先将CF清零。

CLC



3. INC指令

- 格式：

  INC OPRD

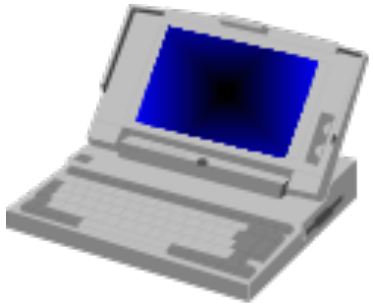
不能是段寄存器
或立即数

- 操作：

  OPRD+1 → OPRD

常用于在程序中修改地址指针

INC指令不影响CF，但影响其
它状态标志位



二、减法指令

普通减法指令SUB

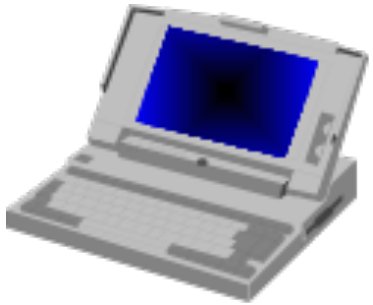
考虑借位的减法指令SBB

减1指令DEC

比较指令CMP

求补指令NEG

减法指令对操作数的要求与对应的加法指令相同



1. SUB指令

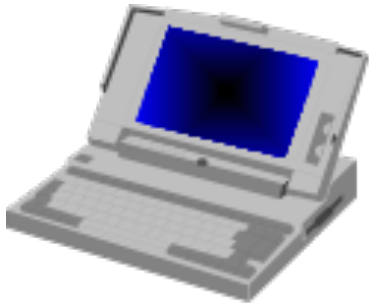
- 格式：

  SUB OPRD1，OPRD2

- 操作：

  OPRD1- OPRD2 → OPRD1

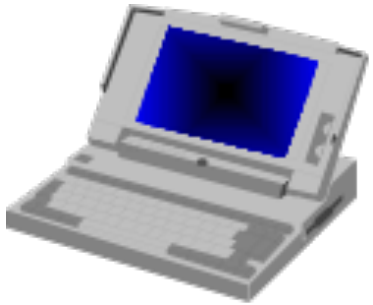
- 对标志位的影响与ADD指令同



2. SBB指令

- 指令格式、对操作数的要求、对标志位的影响与SUB指令完全一样
- 指令的操作：SBB OPRD1，OPRD2

OPRD1- OPRD2- CF → OPRD1



3. DEC指令

- 格式：

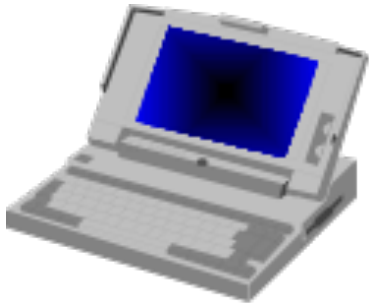
  DEC OPRD

- 操作：

  OPRD - 1 → OPRD

指令对操作数的要求与INC相同

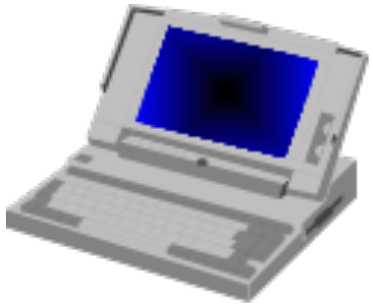
指令常用于在程序中修改计数值



应用程序例

```
        MOV  BL，2
NEXT1：MOV  CX，0FFFFH
NEXT2：  DEC  CX
        JNZ  NEXT2        ; ZF=0转NEXT2
        DEC  BL
        JNZ  NEXT1        ; ZF=0转NEXT1
        HLT               ; 暂停执行
```

延时 2×FFFF次



4. NEG指令

- 格式：

  NEG OPRD

  8/16位寄存器或存储器操作数

- 操作：

  0 - OPRD ⟶ OPRD

取反+1

用0减去操作数，相当于对该操作数求补码

OPRD≠0时，CF=1

两点注意请见教材p117

5. CMP指令

- 格式：

  CMP OPRD1，OPRD2

- 操作：

  OPRD1- OPRD2

指令执行的结果不影响目标操作数，仅影响标志位！

CMP指令

- **用途：**

  用于比较两个数的大小，可作为条件转移指令转移的条件

- 指令对操作数的要求及对标志位的影响与SUB指令相同

CMP指令

- 两个无符号数的比较：

CMP AX，BX

若 AX > BX　　CF=0　无借位

若 AX < BX　　CF=1　有借位

若 AX=BX　　CF=0，ZF=1

CMP指令

- 两个带符号数的比较

  CMP AX，BX

两个数的大小由OF和SF共同决定

OF和SF状态相同　　AX ⩾ BX

OF和SF状态不同　　AX < BX

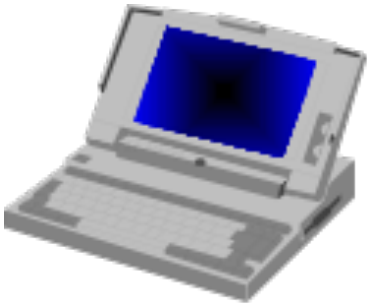
CMP指令例

```
        LEA BX，MAX
        LEA SI，BUF
        MOV CL，20
        MOV AL，[SI]
NEXT：INC  SI
        CMP AL，[SI]
        JNC  GOON        ；CF=0转移
        XCHG  [SI]，AL
```

```
GOON：DEC  CL
        JNZ  NEXT
        MOV [BX]，AL
        HLT
```

程序功能？

程序功能



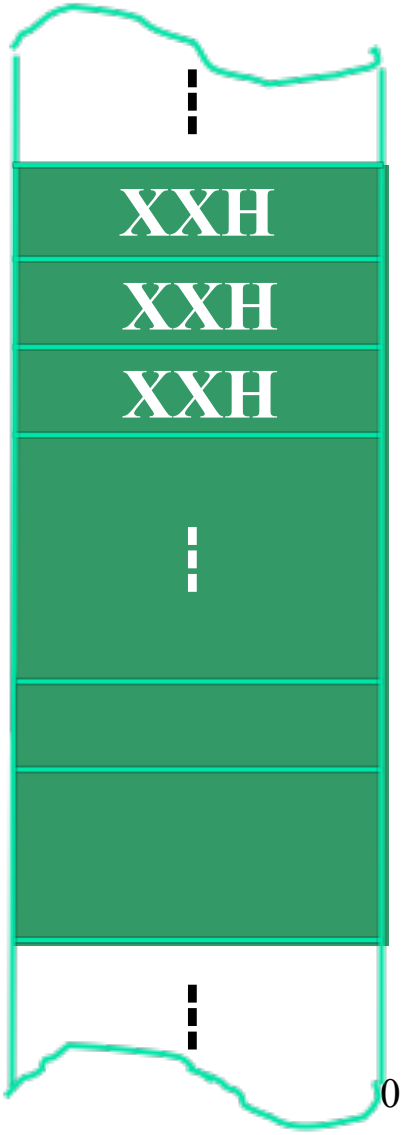
```
        LEA BX，MAX
        LEA SI，BUF
        MOV CL，20
        MOV AL，[SI]
NEXT：  INC  SI
        CMP AL，[SI]
        JNC  GOON       ；CF=0转移
        XCHG [SI]，AL
GOON：  DEC  CL
        JNZ  NEXT
        MOV [BX]，AL
        HLT
```

程序功能

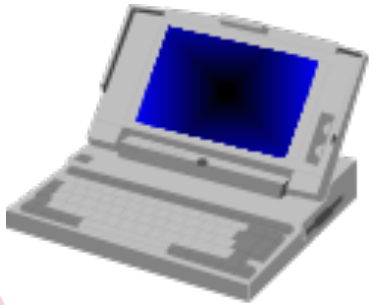
？

在20个数中找最大的数，并将其存放在MAX单元中。

MAX



三、乘法指令



无符号的乘法指令MUL OPRD

带符号的乘法指令IMUL OPRD

- 格式：　　MUL　OPRD

不能是立即数

- 操作：

OPRD为字节数　　AL×OPRD　　AX

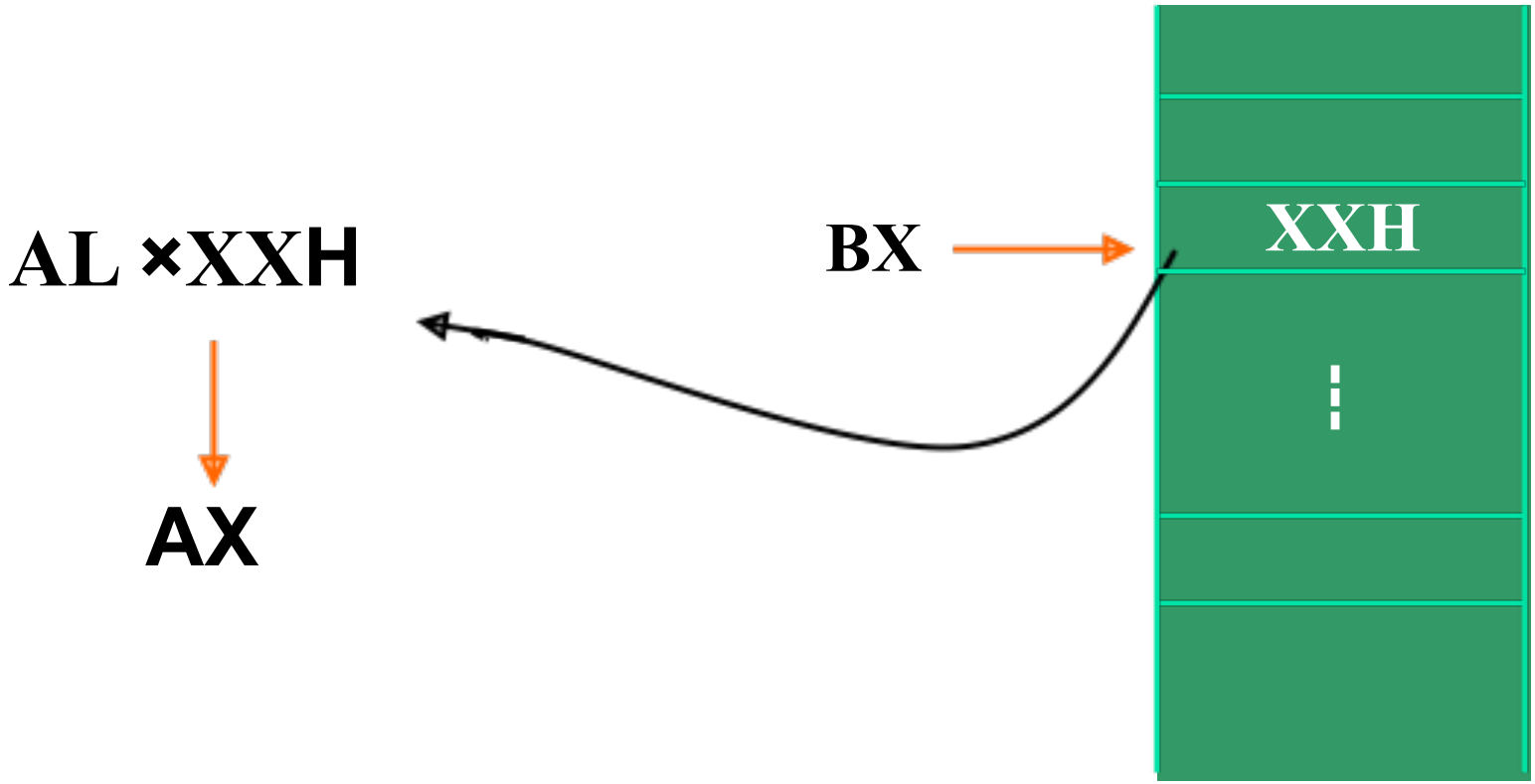
OPRD为16位数　　AX×OPRD　　DXAX

注：

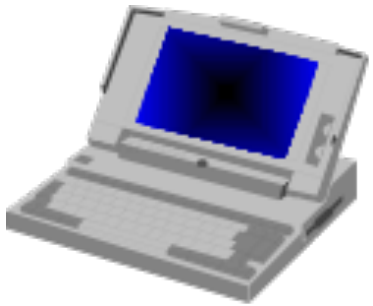
乘法指令采用隐含寻址，隐含的是存放被乘数的累加器AL或AX及存放结果的AX，DX；

无符号数乘法指令例

- **MUL BYTE PTR[BX]**



带符号数乘法指令



- 格式：

  IMUL OPRD

- **IMUL指令将OPRD视为带符号数，运算时若操作数为负数，要先将操作数求补码，运算后再将结果求补。**

两条乘法指令的比较

- 例：设：AL=FEH，CL=11H，求AL与CL的

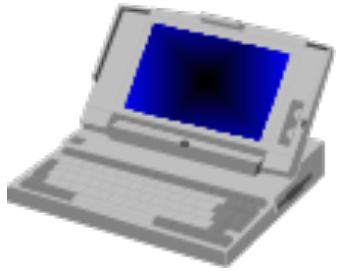
  若设为无符号数

  执行：MUL CL　结果：AX=10DEH

  若将两操作数看作有符号数

- 执行：IMUL CL　结果：AX=FFDEH=-34

四、除法指令



无符号除法指令

- 格式：
- **DIV OPRD**

有符号除法指令

- 格式：
- **IDIV OPRD**

除法指令的操作

指令要求被除数是除数的双

若OPRD是字节数

- 执行：AX/OPRD
- 结果：
  - AL=商　　AH=余数

若OPRD是双字节数

- 执行：DXAX/OPRD
- 结果：
  - AX=商 DX=余数