

# 微型计算机原理与接口技术（第5版）

## 课后答案及问题墙

### 第一章 绪论

{崔文韬问}: 课后习题第一题, 二进制数与十进制数转换。

{崔文韬答}:  $11001010B=202D$ ,  $00111101B=61D$ ,  $01001101B=77D$ ,  $10100100B=164D$ 。

{崔文韬问}: 课后习题第二题, 16 进制数与十进制数转换。

{崔文韬答}:  $12CH=300D$ ,  $0FFH=255$ ,  $3A8DH=14989D$ ,  $5BEH=1470D$

{崔文韬问}: 课后习题第三题, 十进制数转化为二进制数和 16 进制数。

{杨艺答}:  $25D=19H=00011001B$ ,  $76D=4CH=01001100B$ ,  $128D=80H=00000001\ 00000000B$ ,  
 $134D=86H=00000001\ 0000110B$

{杨艺答}:  $128D=80H=10000000B$ ,  $134D=86H=10000110B$

{崔文韬问}: 课后习题第四题, 写出 10 进制数的 BCD 码

{杨艺答}:  $327D=(0011\ 0010\ 0111)BCD$ ,  $1256D=(0001\ 0011\ 0101\ 0110)BCD$

{杨艺答}:  $1256D=(0001\ 0010\ 0101\ 0110)BCD$

{崔文韬问}: 英文单词 About 的 ASCII 码

{沙猛答}: 3935H

{王金鑫改}: 41H, 62H, 6FH, 75H, 74H

{崔文韬问}: 数字 95 的 ASCII 码

{王金鑫答}: 39H, 35H

{崔文韬问}: 课后习题第六题: 10 进制数的原码、补码、反码

{杨艺答}:

【+42】原=00101010B=【+42】反=【+42】补

【-42】原=10101010B, 【-42】反=11010101B, 【-42】补=11010110B

【+85】原=01010101B=【+85】反=【+85】补

【-85】原=11010101B，【-85】反=10101010B，【-85】补=10101011B

{崔文韬问}：机器语言或者机器码（Machine Code），汇编语言（Assemble Language），高级语言的定义

{沙猛答}：

机器码：计算机只认得二进制数码，计算机中的所有指令都必须用二进制表示，这种用二进制表示的指令称为机器码。

汇编语言：用助记符来代替二进制的机器码的符号语言

高级语言：相对于机器语言，接近人们使用习惯的程序设计语言。

{崔文韬问}：课后习题第 10 题

{崔文韬答}：参考课本 16 页图 1.4

{崔文韬问}：课后习题第 11 题

{崔文韬答}：参考课本 11 页图 1.2

{杨艺答}：微处理器、存储器、I/O 接口，I/O 设备和总线。6

{崔文韬问}：课后习题第 12 题

{崔文韬答}：ALU：Arithmetic Logic Unit，CPU：Central Processing Unit，PC：Personal Computer，DOS：Disk Operation System

{崔文韬问}：8086 和 80386 各有多少根地址线，可直接寻址的内存空间是多少，他们的数据线各有多少根？

{杨艺答}：8086 有 20 根地址线 A19~A0，可直接寻址的内存空间是  $2^{20}$  个字节单元，有 16 根数据线；80386 有 32 根地址线，可直接寻址的内存空间是  $2^{32}$  个字节单元，有 32 根数据线。

{崔文韬问}：什么是二进制编码，常用的二进制编码有哪两种？

{杨艺答}：采用若干特定的二进制码的组合来表示各种数字、英文字母、运算符号等的编码方式叫做二进制编码，常见的二进制编码有 BCD 码和 ASCII 码两种。

{崔文韬问}：解释位，字节，字，字长的含义？

{沙猛答}：

位 bit：计算机中二进制数的每一位 0 或 1 是组成二进制信息的最小单位，称为位。

字节 byte: 8 个二进制信息组成的一个单位称为一个字节, 1 Byte=8 Bits。

字 word: 由 16 位二进制数即两个字节组成。

字长 word length: 决定计算机内部一次可以处理的二进制代码位数。

{刘玉年问}: 存在计算机中的数都是以有符号数存储的, 还是以无符号数存储的呢?

{崔文韬答}: 刘玉年同学, 你能先自己尝试回答一下这个问题, 或者说你自己的理解是什么?

{刘玉年答}: 应该是两者都不是吧, 因为在运算的时候仅仅就是二进制数的运算, 而判断结果的意义(是什么样的数)是通过标志为判断的。

{崔文韬答}: 两者都不是, 存储器中数据的含义是完全由编程人员决定的。计算机只接收二进制数据, 即 01 序列。至于原始数据是什么以及如何转化为二进制数据, 都是由编程人员决定的。计算机对所存储数据按照二进制计算法则进行运算, 为适应有符号数和无符号数运算两种情况, 通过设置标志位来计算结果在两种情况下的意义。

---

## 第二章 8086CPU

{崔文韬问}: 8086/8088 可直接寻址多少内存(字节)单元? 多少 IO 端口? 外部数据线各有多少?

{董国福答}: 8086/8088 可直接寻址 1MB 内存空间; 可以访问 64K 个 I/O 端口; 但外部数据总线 8086 有 16 根, 8088 有 8 根。

{崔文韬问}: 8086CPU 内部由那两部分组成?

{俞楠答}: 8086CPU 由总线接口单元(BIU)和指令执行单元(EU)两部分组成。

{崔文韬问}: EU, BIU, AX, BX, CX, DX, DS, CS, ES, SS, SP, BP, DI, SI 全称?

{董国福答}:

EU: Execution Unit

BIU: Bus Interface Unit

AX: Accumulator

BX: Base

CX: Count

DX: Data

DS: Data Segment

CS: Code Segment

ES: Extra Segment

SS: Stack Segment

SP: Stack Pointer

BP: Base Pointer

DI: Destination Index

SI: Source Index

{崔文韬问}: 8086CPU 内部包含哪些寄存器? 各有什么用途?

{俞楠答}:

1.数据寄存器: 用来存放 16 位数据信息或地址信息。

2.地址指针和变址寄存器: SP, BP, SI, DI 这组地址指针个变址寄存器加上基址寄存器 BX, 可与段寄存器配合使用, 一起构成内存的物理地址。

(数据寄存器和地址指针和变址寄存器则被称为通用寄存器。)

(段基址和段内偏移地址 Offset 组合起来就可形成 20 位物理地址) 5.标志寄存器: 6 个状态标志 CF, PF, AF, ZF, SF, OF 用来表示指令执行后的结果或状态特征, 根据这些特征, 由转移指令控制程序的走向。3 个控制标志, TF, IF, DF, 可以根据需要用程序设置或清除。

{崔文韬问}: 带符号数 10110100B 和 11000111B 相加, 各标志位为多少? 哪些标志位有意义? 如果作为无符号数相加, 各标志位为多少? 哪些标志位有意义?

{崔文韬答}: 二进制数所有位都参与运算

10110100

+ 11000111

1 01111011

OF SF ZF AF PF CF

1 0 0 0 1 1

程序员将该数看做有符号数, 因此 SF, OF, ZF, PF 有意义。

如果将该数看做无符号数, 计算过程一样, 标志位结果相同, CF, ZF, PF 有意义。

利用 debug 程序验证结果:

运行 dosbox, 输入 debug, r 命令查看初始寄存器数值, a 命令输入汇编指令, t 命令执行查看结果, 过程



{崔文韬问}: 课后习题十一题, SS: SP=2000,0300H, 堆栈在内存当中的物理地址范围是多少? 执行两条 PUSH 指令后, SS: SP=? 再执行一条 PUSH 指令后, SS: SP=?

{俞楠答}:

物理地址范围:(2000H\*16+0):(2000H\*16+(0300H))=20000H:20300H

执行两条 PUSH 指令后: SS: SP=2000H:(0300H-4)=2000H:02FCH

再执行一条 PUSH 指令后, SS: SP=2000H:(02FC-2)=2000H:02FAH

{刘瑾改}: 堆栈在内存当中的物理地址范围为: 2000:0000H~2000:(0300H-1)

{崔文韬问}: 课后习题十二题, 从存储单元 2000H 开始存放的字节数据为: 3AH,28H,56H,4FH,试画出示意图说明, 从 2000H 和 2001H 单元开始取出一个字数据各要进行几次操作, 取出的数据分别等于多少?

{俞楠答}:从 2000H 中取出一个字数据要进行一次操作, 取出字为 283A。从 2001H 中取出一个字数据要进行两次操作, 取出字为 5628。

{罗小东补充}: 如下表, 如果从 2000H 取出一个字数据, 则执行一次操作直接取出一个字 283AH

如果从 2001H 开始取出一个字数据, 则需进行两次操作, 分别是——先从 2000H 单元开始读取一个字 283AH, 取得低字节 28H, 舍弃 3AH; 再从 2002H 单元读取一个字数据, 4F56H, 取得其高字节 56H, 然后就可以得到 2001H 单元开始取出的一个字数据——5628H。

原理就是: 8086CPU 对存储器进行存取操作时, 都是从偶地址体开始的。

2000 3A

2001 28

2002 56

2003 4F

{罗小东问}: 一个含有 16 个字节数据的变量, 它的逻辑地址为 1000: 0100H, 那么该变量的最后一个字节数据的物理地址是\_\_\_\_H? (可不可以答案再加点简单分析呀)

{崔文韬问}: 设定 SS: SP 后形成的堆栈占据一定的物理地址范围, 是否可无限次执行 PUSH 或者 POP 指令? 为什么?

{苏子宇答}: 堆栈有一定的容量, 无限次执行 push 会超出范围, 导致覆盖设定的堆栈空间外的数据, 产生栈顶越界现象。堆栈空间是程序员向系统请, 系统开辟的安全数据空间, 空间外的数据可能具有其他用途, 任意改动可能引发错误。8086CPU 不提供检测栈顶是否越界的机制。编程时要注意栈顶越界问题, 根据可能用到的最大栈空间来安排堆栈大小, 防止入栈的数据导致栈顶越界。



{刘玉年问}: 8086 有 20 根地址总线, 可寻址的内存空间是 1M, 是不是就说 8086 里面的内存空间就只有 1M 呢? 如果不是这样的, 那么多余的地址空间又该如何寻址呢?

{崔文韬答}: 8086 的内存寻址空间真的只有 1M, 这 1M 空间分配给内存使用 (包含显存, 主内存, BIOS 的 ROM)。8086 针对外设的寻址, 通过硬件电路另外生成 64k 的 IO 地址, 供寻址 IO 接口使用

还有啊, 如果真的是内存空间的大小  $m$  和地址总线的数目  $n$  是:  $m=2^n$  的关系的话, 那么现在的片子运存是 4g 的话, 那要 32 根地址总线的, 相应的地址输入输出端口也要有 32 个, 这是不是又有些浪费?

{崔文韬答}: 内存空间大小和地址数目的关系就是如此。32 根地址线对应 4g 内存, 这个是必须的, 不存在浪费问题。

---

### 第三章 8086 寻址方式和指令系统

{崔文韬问}: 习题 1 中题目, 分别说明源操作数和目的操作数各采用的寻址方式

1. mov ax,2408h
2. mov cl,0ffh
3. mov bx,[si]
4. mov 5[bx],bl

{沙猛答}: 1.立即数, 寄存器 2.立即数, 寄存器 3.寄存器间接, 寄存器 4.寄存器, 寄存相对

{崔文韬问}: 习题 1 中题目, 分别说明源操作数和目的操作数各采用的寻址方式

1. mov [bp+100],ax
2. mov [bx+di],'S'
3. mov dx,es:[bx+si]
4. mov val[bp+di],dx

{俞楠答}:1.寄存器寻址, 寄存器相对寻址 2.立即数寻址, 基址变址寻址 3.基址变址寻址, 寄存器寻址 4.寄

寄存器寻址，相对基址变址寻址。

{崔文韬问}：习题 1 中题目，分别说明源操作数和目的操作数各采用的寻址方式

1. in al,05

2. mov ds,ax

{姚胜答} 1.寄存器，立即数。2.寄存器，寄存器

{崔文韬问}：习题 2 中题目，已知 DS=1000h，bx=0200h，si=02h，内存 10200h~10205h 的内容分别为 10h，2ah，3ch,46h,59h,6bh。下列每条指令执行完成后，ax 寄存器的内容是什么。

1. mov ax,0200h

2. mov ax,[200]

{沙猛答}：1.为 0200h 2.为 2a10h

{崔文韬问}：习题 2 中题目，已知 DS=1000h，bx=0200h，si=02h，内存 10200h~10205h 的内容分别为 10h，2ah，3ch,46h,59h,6bh。下列每条指令执行完成后，ax 寄存器的内容是什么。

1. mov ax,bx

2. mov ax,3[bx]

{于刚答}：执行指令 1 后，ax 为 10h；执行 2 后，ax 为 46h

{俞楠改}：1.0200H 2.5946H

{崔文韬问}：习题 2 中题目，已知 DS=1000h，bx=0200h，si=02h，内存 10200h~10205h 的内容分别为 10h，2ah，3ch,46h,59h,6bh。下列每条指令执行完成后，ax 寄存器的内容是什么。

1. mov ax,[bx+si]

2. mov ax,2[bx+si]

{于刚答}：执行指令 1,ax 为 3ch；执行指令 2，ax 为 59h

{俞楠改}：1.463CH 2.6B59H

{崔文韬问}：习题 3 中题目，设 ds=1000h,es=2000h,ss=3500h,si=00a0h,di=0024h,bx=0100h,bp=0200h,数据段



中变量名为 val 的偏移地址为 0030h，下列源操作数字段的寻址方式是什么？物理地址是多少？

1. mov ax,[100h]
2. mov ax,val
3. mov ax,[bx]
4. mov ax,es:[bx]

{俞楠答}:1.直接寻址 10100H 2.直接寻址 10030H 3.寄存器间接寻址 10100H 4.寄存器间接寻址 20100H

{崔文韬问}: 习题 3 中题目，设 ds=1000h,es=200h,ss=3500h,si=00a0h,di=0024h,bx=0100h,bp=0200h,数据段中变量名为 val 的偏移地址为 0030h，下列源操作数字段的寻址方式是什么？物理地址是多少？

1. mov ax,[si]
2. mov ax,[bx+10h]
3. mov ax,[bp]
4. mov ax,val[bp+si]

{俞楠答}:1.寄存器间接寻址 100A0H 2.寄存器相对寻址 10110H 3.寄存器间接寻址 35200H 4.相对基址变址 35230H

{崔文韬问}: 习题 3 中题目，设 ds=100h,es=200h,ss=3500h,si=00a0h,di=0024h,bx=0100h,bp=0200h,数据段中变量名为 val 的偏移地址为 0030h，下列源操作数字段的寻址方式是什么？物理地址是多少？

1. mov ax,val[bx+di]
2. mov ax,[bp+di]

{俞楠答}:1.相对基址变址寻址 10154H 2.基址变址寻址 35224H

{崔文韬问}: 习题 4，指令机器码，利用 debug 命令可以查看指令对应的机器码，此内容大家了解即可。

mov al,cl 机器码在内存中从低到高存放为：88c8

mov dx,cx 机器码在内存中从低到高存放为：89ca

mov word ptr [bx+100h],3150h 机器码在内存中从低到高存放为：c78700015031

通过 debug 中的 a 命令输入以上指令，通过 d 命令查看对应的二进制机器码，过程及结果如下：

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip: 0, Program: DEBUG
CLEAN BAT 20 24-03-2016 18:05
DEMO ASM 2,314 08-03-2016 14:03
DISPLAY ASM 551 09-03-2016 16:15
EXAM ASM 137 29-02-2016 9:45
EXAMPLE1 ASM 168 18-03-2016 10:08
EXAMPLE2 ASM 188 23-03-2016 13:13
EXAMPLE3 ASM 231 24-03-2016 18:11
FRAME5 ASM 192 03-03-2016 13:36
FRAME6 ASM 218 03-03-2016 15:49
HELLO ASM 335 03-03-2016 15:49
JIAFA ASM 633 03-03-2016 16:14
12 File(s) 5,060 Bytes.
2 Dir(s) 262,111,744 Bytes free.

C:\CODE>debug
~a
0740:0100 mov al,cl
0740:0102 mov dx,cx
0740:0104 mov [bx+100],3150
^ Error
0740:0104 mov word ptr [bx+100],3150
0740:010a
~d 0740:0100 0109
0740:0100 8B C8 B9 CA C7 87 00 01-50 31 .....P1
~ ^
```

{崔文韬问}：习题 6，指出指令错误

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]

{刘玉年答}：1.数据长度不一致 2.立即数不能做目的操作数 3.立即数不能向段寄存器传送数据

4.存储器与存储器之间不能直接传送数据 5.IP 寄存器的内容不能通过程序编辑 6.IP 寄存器不能作为源操作数或目的操作数 7.bx 和 bp 不允许出现在同一[]中。

{崔文韬问}：习题 6，指出指令错误

1. mov al,es:[bp]
2. mov dl,[si+di]

3. mov ax,offset 0a20h

4. mov al,offset table

5. xchg al,50

6. in bl,05h

7. out al,offeh

{刘玉年答}: 1.正确 2.si 和 di 不能出现在同一[]中 3.offset 后面应跟符号地址 4.符号 table 的偏移地址是 16 位的, 数据长度不一致 5.xchg 指令中不能使用立即数 6.目的操作数只能是 ax 或 al 7.0ffeh 大于 ff, 应用 dx 表示

{崔文韬问}: 习题 5, 已知程序的数据段为:

```
data segment
a db '$',10h
b db 'COMPUTER'
c dw 1234h,0ffh
d db 5 dup(?)
e dd 1200459ah
data ends
```

请画出上述数据变量在内存中的数据的位置示意图, 假设数据段段地址为 X。参考课本 P122 页, 图 4.2。

{俞楠答}:

山东大学					
A	24	C	34	45	
	10		12	00	
B	43		FH	12	
	4F		0F		
	4D	D	不确定		
	50		不确定		
	55		不确定		
	54		不确定		
	44		不确定		
	52	E	9A		

{崔文韬答}: 程序参考实验代码如下, 请同学们编译后使用 debug 调试执行, 观察结果, 验证答案。

```
assume cs:code,ds:data
```

```
data segment
```

```
a db '$',10h
```

```
b db 'COMPUTER'
```

```
c1 dw 1234h,0ffh ;c 及 C 是汇编语言关键字不能作为变量名。
```

```
d db 5 dup(?)
```

```
e dd 1200459ah
```

```
data ends
```

```
code segment
```

```
main:
```

```
mov ax,data
```

```
mov ds,ax ;完成数据段地址与 ds 的绑定
```

```
mov al,a ;al=24h
```

```
mov dx,c1 ;dx=1234h
```

```
xchg dl,a ;dl=24h, 变量 a 对应内存空间数据变为 34h,10h
```

```
mov bx,offset b ;bx=0002h
```

```
mov cx,3[bx] ;cx=5550h,对应 ASC 码 'UP'
```

```

lea bx,d ;bx=000eh
lds si,e ;ds=1200h,si=459ah
les di,e ;请同学们回答你实验完成后的结果 es=?,di=?
mov ax,4c00h
int 21h
code ends
end main

```

{李聪聪答}: es=5613 di=fc46

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG

```

AX=0724 BX=0002 CX=0047 DX=1224 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076E IP=0013  NU UP EI PL NZ NA PO NC
076E:0013 8B4F03      MOV     CX,[BX+03]      DS:0005=5550
-t
AX=0724 BX=0002 CX=5550 DX=1224 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076E IP=0016  NU UP EI PL NZ NA PO NC
076E:0016 8D1E0E00     LEA     BX,[000E]      DS:000E=0000
-t
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076E IP=001A  NU UP EI PL NZ NA PO NC
076E:001A C5361300     LDS     SI,[0013]      DS:0013=459A
-t
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=459A DI=0000
DS=1200 ES=075C SS=076B CS=076E IP=001E  NU UP EI PL NZ NA PO NC
076E:001E C43E1300     LES     DI,[0013]      DS:0013=FC46
-t
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=459A DI=FC46
DS=1200 ES=5613 SS=076B CS=076E IP=0022  NU UP EI PL NZ NA PO NC
076E:0022 B8004C      MOV     AX,4C00

```

{崔文韬答}: 李聪聪同学，能尝试分析一下，为什么结果是这个样子吗？

{李聪聪答}: 分析了一遍，之前代码执行的好像有问题，可能已写入其他值，les之后才会这样。正常运行之后应该 es=0000h,di=0000h。因为上一步操作更改了段寄存器 ds 的值为 1200h，而 e 的有效地址为 0013h，所以执行 les 时应从 1200: 0013h 取连续的四个字节分别给 di，es,而初始状态时存储单元中应为 0000h。（不知道是不是这样）。观察很仔细，基本原因就是这样的！赞一个！。你之前的代码执行也没有问题。按照我后面给你的提示，再尝试一下，就知道以前你的程序也没问题了。

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG

AX=0724 BX=0002 CX=5550 DX=1224 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076E IP=0016  NU UP EI PL NZ NA PO NC
076E:0016 8D1E0E00 LEA BX,[000E] DS:000E=0000
-p
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076E IP=001A  NU UP EI PL NZ NA PO NC
076E:001A C5361300 LDS SI,[0013] DS:0013=459A
-p
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=459A DI=0000
DS=1200 ES=075C SS=076B CS=076E IP=001E  NU UP EI PL NZ NA PO NC
076E:001E C43E1300 LES DI,[0013] DS:0013=0000
-p
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=459A DI=0000
DS=1200 ES=0000 SS=076B CS=076E IP=0022  NU UP EI PL NZ NA PO NC
076E:0022 B8004C MOV AX,4C00
-p
AX=4C00 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=459A DI=0000
DS=1200 ES=0000 SS=076B CS=076E IP=0025  NU UP EI PL NZ NA PO NC
076E:0025 CD21 INT 21
-

```

{崔文韬答}: 李聪聪同学, 为了验证自己的猜想, 是不是可以在运行完 les di,e 这条指令后, 使用 debug 的 d 指令查看一下 1200:0013 中连续 4 个字节的内容, 看看是否一致? 或者从 debug 显示的内容中说明其内容? 或者在运行 les di,e 指令之前, 运行 e 1200:0013 更改该存储单元连续的四个字节内容, 然后看看是否与运行后 DI, ES 内容一致呢?

{李聪聪答}: 开始分析的时候就是从 debug 显示的 DS:0013=0000 推测的。运行了 e 命令之后猜测进一步得到验证: 从 1200: 0013h 取连续的四个字节分别给 di, es, 3412 存入 di,2143 存入 es。

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG

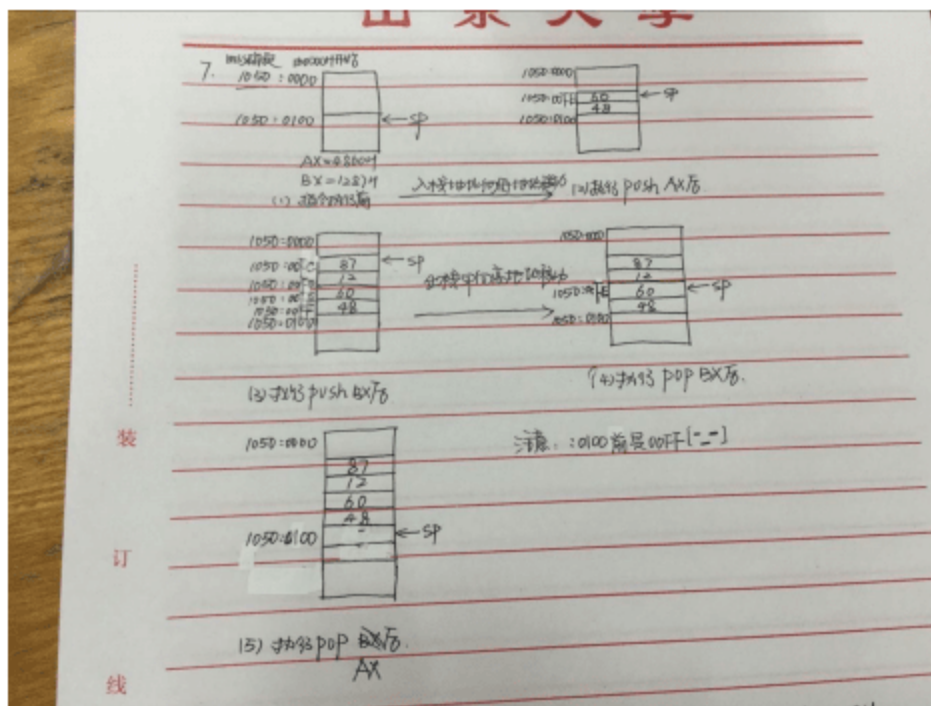
076E:0013 8B4F03 MOV CX,[BX+03] DS:0005=5550
-p
AX=0724 BX=0002 CX=5550 DX=1224 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076E IP=0016  NU UP EI PL NZ NA PO NC
076E:0016 8D1E0E00 LEA BX,[000E] DS:000E=0000
-p
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076E IP=001A  NU UP EI PL NZ NA PO NC
076E:001A C5361300 LDS SI,[0013] DS:0013=459A
-p
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=459A DI=0000
DS=1200 ES=075C SS=076B CS=076E IP=001E  NU UP EI PL NZ NA PO NC
076E:001E C43E1300 LES DI,[0013] DS:0013=0000
-e 1200:0013
1200:0013 00 12 00 34 00 43 00 21
-p
AX=0724 BX=000E CX=5550 DX=1224 SP=0000 BP=0000 SI=459A DI=3412
DS=1200 ES=2143 SS=076B CS=076E IP=0022  NU UP EI PL NZ NA PO NC
076E:0022 B8004C MOV AX,4C00
-

```

{崔文韬问}: 习题 7, 已知 ss=1050h, sp=0100h, ax=4860h, bx=1287h, 试用示意图表示执行下列指令过程中, 堆栈中的内容和堆栈指针 sp 是如何变化的 (参考例子 3.29)

{胡玲答}:





{崔文韬问}: 习题 8, 已知当前数据段中有一个十进制数字 0~9 的 7 段代码表, 其数值依次为: 40h, 79h, 24h, 30h, 19h, 12h, 02h, 78h, 00h, 18h。要求用 xlat 指令将十进制数 57 转换成相应的 7 段代码值, 存到 bx 寄存器当中, 试写出相应的程序段。(参考例子 3.31, 注意 5 和 7 要分别进行转换)

{胡玲答}: 不是图标类型的答案, 还是不要使用图片完成回答吧。最好, 可以在后边附上自己的完整代码及 debug 调试验证过程。

```
data segment
table db 40h,79h,24h,30h,19h
      db 12h,02h,78h,00h,18h
data ends
mov al,5
mov bx,offset table
xlat table
mov ah,al
mov al,7
xlat table
mov bx,ax
```

{胡玲问}: 其实吧 debug 还不太会用 debug+文件名之后用单步执行 t 指令然后不会看结果== 我错了 不会==写整个程序代码。胡玲同学, 现在能把这个题目补充完整了吗??

{崔文韬问}: 习题 9, 下列指令完成什么功能

1. add al,dh
2. adc bx,cx
3. sub ax,2710h
4. dec bx



{姚胜答}: 1.al 与 dh 相加存到 al 中; 2.bx 与 cx 与 cf 当前值相加存到 bx 中;

{胡玲答}: 3.减法指令 将 ax 寄存器的内容减去 2710h 再存在 ax 中 4.减量指令, 将 bx 寄存器的内容减一再存在 bx 中

{崔文韬问}: 习题 9, 下列指令完成什么功能

1. neg cx
2. inc bl
3. mul bx
4. div cl

{于刚答}: 1.对 cx 取负, 并把结果送回 cx; 2.bl 加 1, 并把结果送回 bl; 3.bx\*ax 结果为 32 位数, 高位字放在 dx, 低位字放在 ax; 4.ax/cl 商放在 al 中, 余数放在 ah 中

{崔文韬问}: 习题 10, 已知 ax=2508h, bx=0f36h, cx=0004h, dx=1864h,求下列每条指令执行后的结果是什么? 标志位 cf 等于什么?

1. add ah,cl {胡玲答}:ax 寄存器的高地址存的数是 29h ,cf=0
2. or bl,30h {胡玲答}:0011 0110B OR 0011 0000B=36h, cf=0
3. not ax {胡玲答}:按位操作, ax 中存的数是 DAF7h, 无进位 cf=0
4. xor cx,0fff0h {胡玲答}:0004h 和 fff0h 异或得 fff4h, cf=0

{崔文韬问}: 习题 10, 已知 ax=2508h, bx=0f36h, cx=0004h, dx=1864h,求下列每条指令执行后的结果是什么? 标志位 cf 等于什么?

1. test dh,0fh {胡玲答}:查 dh 的 D3 是否等于 1, 无进位, cf=0, 操作数不变
2. cmp cx,00h {胡玲答}:0004h-00h=0004h 即 cx=0004h, 结果不返回操作数, cf=0
3. shr dx,cl {胡玲答}:1864h 逻辑右移 4h 得 dx 为 0186h, cf=0
4. sar al,1 {胡玲答}:算数右移得 ax 为 2504h, 最高位不变, cf=0

{崔文韬问}: 习题 10, 已知 ax=2508h, bx=0f36h, cx=0004h, dx=1864h,求下列每条指令执行后的结果是什么? 标志位 cf 等于什么?

1. shl bh,cl {胡玲答}:36h 逻辑左移 04h 得 bx=f036h, cf=0, 指令中写成 bx=0f036h
2. sal ax,1 {胡玲答}:2508h 算数左移一位得 0100 1010 0001 0000B=4A10h 即 ax=4A10h, cf=0
3. rcl bx,1 {胡玲答}: 0000 1111 0011 0110B 通过进位循环左移 1 位得 0001 1110 0110 1100B=1E6Ch  
cf=0
4. ror dx,cl {胡玲答}: 1864h 循环右移 4h 位可得 dx=4186h, cf=0 且 cf 没有参加循环

{崔文韬答}: 有两处错误, 请胡玲同学改正。(已改正)

{崔文韬问}: 习题 11,假设数据段定义如下:

```
data segment
```

```
string db 'The Personal Computer & TV'
```

```
data ends
```

试用字符串操作等指令完成以下功能: 参考字符串处理指令例子

(1): 把该字符串传送到附加段中偏移量为 GET\_CHAR 开始的内存单元中。

{崔文韬答}: 参考代码如下: 同学们可以使用 notepad++ 编辑源代码, 然后使用 ml 编译连接为可执行文件, 通过 debug 调试程序, 观察结果。

```
assume cs:code,ds:data,es:extra
```

```
data segment ;string 字符串长度为
```

```
string db 'The Personal Computer&TV'
```

```
str_end db '$'
```

```
data ends
```

```
extra segment
```

```
get_char db 32 dup(0)
```

```
extra ends
```

```
code segment
```

```

start:
mov ax,data
mov ds,ax ;数据段与 ds 的绑定
mov ax,extra
mov es,ax ;附加段与 es 的绑定
lea si,string ;源串偏移地址设定 mov si,offset string
lea di,get_char ;目的偏移地址设定 mov di,offset get_char
mov cx,offset str_end-string ;计算字符串长度
cld
rep movsb
mov ax,4c00h
int 21h
code ends
end start

```

{崔文韬问}: 习题 11,假设数据段定义如下:

```

data segment
string db 'The Personal Computer & TV'
data ends

```

试用字符串操作等指令完成以下功能: 参考字符串处理指令 CMPS 例子,编写参考程序, 调试并验证

(2): 比较该字符串是否与“The Computer”相同, 如果相同则将 al 寄存器的内容置 1, 否则置 0。并要求将比较次数送到 BL 寄存器中。

{崔文韬问}: 习题 11,假设数据段定义如下:

```

data segment
string db 'The Personal Computer & TV'
data ends

```

试用字符串操作等指令完成以下功能: 参考字符串处理指令 SCAS 例子, 编写参考程序, 调试并验证

(3): 检查该字符串中是否有“&”符, 如果有则用空格将其替换。

{崔文韬问}: 习题 12 题, 编程将 AX 寄存器中的内容以相反的次序传入 DX 寄存器中, 并要求 AX 中的内容不被破坏

{崔文韬答}: 参考程序

;习题 P111 页习题 12 参考程序

```

assume cs:code
code segment
start:
mov ax,1234h ;0001001000110100B
mov dx,0 ;程序执行完成后 dx=0010110001001000B=2c48h
mov cx,16
s: rol ax,1 ;循环左移 1 位，将最高位移入 cf 中
rcr dx,1 ;通过 CF 完成的循环右移 1 位
loop s ;通过循环，将 cx 倒序移入 dx 中。
mov ax,4c00h
int 21h
code ends
end start

```

{崔文韬问}: 请同学根据上述程序，修改代码，使程序在完成倒序传入的基础上，还能统计 DX 寄存器中 1 的个数是多少？1 的个数可以存在某个通用寄存器中。

{崔文韬问}: 习题 14 第 1 问，下列程序执行完后，AX，BX，CX，DX 寄存器的内容分别是什么？请先回答问题，然后通过编辑源程序验证答案，利用 debug 检验答案的正确性。

{崔文韬答}: 因为 cx=4，因此 loop 指令会使得循环体内程序执行 4 次，因此最终 ax=0005h，bx=0010h，cx=0000h，dx=0000h。

参考程序如下：

;习题 P112 页习题 14 第一问参考程序

```

assume cs:code,ss:stack
stack segment stack
db 16 dup(0)
stack ends

code segment
start:
mov ax,01h
mov bx,02h
mov dx,03h
mov cx,04h
next: inc ax

```

```

add bx,ax
shr dx,1 ;逻辑右移
loop next
mov ax,4c00h
int 21h
code ends
end start

```

利用 debug 加载编译好的程序，利用 r 命令查看寄存器初始值，例如 u 命令查看反汇编结果，确定程序退出返回 dos 的地址，利用 g 命令，连续执行程序，并查看最终寄存器中的内容。过程如下：

```

C:\CODE>debug XIT14.EXE
-r
AX=FFFF BX=0000 CX=0028 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=075C ES=075C SS=076C CS=076D IP=0000  NU UP EI PL NZ NA PO NC
076D:0000 B80100      MOV     AX,0001
-u cs:0 28
076D:0000 B80100      MOV     AX,0001
076D:0003 B80200      MOV     BX,0002
076D:0006 B80300      MOV     DX,0003
076D:0009 B90400      MOV     CX,0004
076D:000C 40          INC     AX
076D:000D 03D8      ADD     BX,AX
076D:000F D1EA      SHR     DX,1
076D:0011 E2F9      LOOP   000C
076D:0013 B8004C      MOV     AX,4C00
076D:0016 CD21      INT     21
076D:0018 46          INC     SI
076D:0019 1B7606      SBB     [BP+06],DH
076D:001C 894618      MOV     [BP+18],AX
076D:001F 89561A      MOV     [BP+1A],DX
076D:0022 B80400      MOV     AX,0004
076D:0025 50          PUSH    AX
076D:0026 0E          PUSH    CS
076D:0027 EBA60A      CALL    0AD0

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip: 0, Program: DEBUG

```

-u cs:0 28
076D:0000 B80100      MOV     AX,0001
076D:0003 BB0200      MOV     BX,0002
076D:0006 BA0300      MOV     DX,0003
076D:0009 B90400      MOV     CX,0004
076D:000C 40          INC     AX
076D:000D 03D8      ADD     BX,AX
076D:000F D1EA      SHR     DX,1
076D:0011 E2F9      LOOP    000C
076D:0013 B8004C      MOV     AX,4C00
076D:0016 CD21      INT     21
076D:0018 46          INC     SI
076D:0019 1B7606      SBB     [BP+06],DH
076D:001C 894618      MOV     [BP+18],AX
076D:001F 89561A      MOV     [BP+1A],DX
076D:0022 B80400      MOV     AX,0004
076D:0025 50          PUSH    AX
076D:0026 0E          PUSH    CS
076D:0027 EBA60A      CALL    0AD0
-g 13
AX=0005 BX=0010 CX=0000 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=075C ES=075C SS=076C CS=076D IP=0013  NU UP EI PL ZR AC PE NC
076D:0013 B8004C      MOV     AX,4C00
- ▲

```

{崔文韬问}: 习题 14 第 2 问, 下列程序执行完后, AX, BX, CX, DX 寄存器的内容分别是什么? 请先回答问题, 然后通过编辑源程序验证答案, 利用 debug 检验答案的正确性。(参考上题解答过程给出答案)。

```

start:
mov ax,01h
mov bx,02h
mov dx,03h
mov cx,04h
next: inc ax
add bx,ax
shr dx,1
loope next

```

{崔文韬问}: 习题 14 第 3 问, 下列程序执行完后, AX, BX, CX, DX 寄存器的内容分别是什么? 请先回答问题, 然后通过编辑源程序验证答案, 利用 debug 检验答案的正确性。参考上题解答过程给出答案)。

```

start:
mov ax,01h
mov bx,02h
mov dx,03h
mov cx,04h
next: inc ax

```

```

add bx,ax
shr dx,1
loopne next

```

{崔文韬问}: 习题 15, 7 名同学英语成绩低于 80 分, 分数存在 array 数组中, 试编写程序给每名同学成绩加 5 分, 结果保存到 new 数组中。(参考 P193 页例 3.93)

{崔文韬答}: 参考程序如下, 请同学们编译链接后, 利用 debug 调试, 验证程序功能。

;习题 P112 页习题 15 参考程序

```

assume cs:code,ds:data
data segment
array db 66,67,68,76,77,78,79 ;原始成绩
new db 7 dup(0) ;修改后成绩
data ends

code segment
start:
mov ax,data ; 数据段绑定到 ds
mov ds,ax
mov bx,0 ;设置数组索引的初始值
mov cx,7
add5: mov al,array[bx]
add al,5
mov new[bx],al
inc bx
loop add5
mov ax,4c00h
int 21h
code ends
end start

```

同学们可以发挥自己的想象力, 使用其他指令实现该功能。编写程序时关注如下三点: 1, 数据从哪里来, 通过何种寻址方式获得; 2, 如何处理数据, 使用哪种指令; 3, 数据到哪里去, 通过何种寻址方式存储数据。

{崔文韬问}: 习题 16, 软中断指令 INT n 中 n 的含义是什么? 取值范围是多少? 当 n=0~4 时, 分别定义什么中断? INTO 指令用于什么场合?



{胡玲答}: n 是中断类型码, 是八位二进制数, 取值范围是 0-255=0-FFh, 软件中断指令也叫陷进中断。

INT 0 定义除法错中断, INT1 定义单步中断, INT2 定义不可屏蔽中断, INT3 定义断电中断, INT4 定义溢出中断

在带符号数进行加减法运算之后必须安排一条 INTO 指令

{崔文韬问}: 习题 17, 那些指令可以使 CF, DF 和 IF 标志直接清零或者置 1?

{刘瑾答}: 执行 STC 指令可以使 CF (进位标志) 置 1; 执行 CLC 指令可以使 CF 清零。

执行 CLD 指令可以使 DF (方向标志) 清零; 执行 STD 指令可以使 DF 置 1。

执行 STI 指令可以使 IF (中断标志) 置 1; 执行 CLI 指令可以使 IF 清零。

## 第四章 汇编语言程序设计

{崔文韬问}: 习题 1, 简述从汇编语言源程序到生成可执行文件\*.exe, 需要经过哪些步骤?

{张多睿答}: 编辑程序生成源程序后, 经汇编程序 MASM 汇编后生成目标文件.OBJ, 目标文件经连接程序 LINK 后连接后, 生成可执行文件。

{崔文韬问}: 伪指令和指令语句各由那几个字段组成? 那些字段是必不可少的?

{李聪聪答}: 指令语句由 4 部分组成, 格式为: 标号: 指令助记符 操作数 ;注释  
其中指令助记符必不可少。

伪指令语句由 4 部分组成, 格式为: 名字 伪指令指示符 操作数 ;注释  
其中伪指令指示符必不可少。

{崔文韬}: 伪指令语句的作用是什么? 他与指令语句的主要区别是什么?

{李聪聪答}: 伪指令语句的作用: 在汇编过程中完成某些特定的功能, 如数据定义、分配存储区、指示程序结束等。

主要区别: 伪指令语句经汇编后不产生机器码, 不能让 CPU 执行, 其所指示的操作在程序汇编时完成, 而指令语句汇编后有对应的机器码, 其操作是在程序运行时完成。

{崔文韬问}: 下列指令完成什么功能:

1. mov al,not 10001110B {胡玲答}:逻辑非运算 将 71h 存到 al 中
2. mov cx,8 GT 00011000B {胡玲答}:8 小于 18h, 结果为假, 输出全零即 cx=0000h
3. mov dl,27/5 {胡玲答}: 除法取商得 5D=05h 即 dl=05h

4. mov bx,\$-LIST {胡玲答}:将现行地址-LIST 偏移量送到 bx 中储存

{崔文韬问}: 阅读下列程序段, 说明每条指令执行后的结果是什么?

```
x1 db 65h,78h,98h
```

```
x2 dw 06ffh,5200h
```

```
x3 dd ?
```

```
go: mov al,type x1
```

```
mov bl,type x2
```

```
mov cl,type x3
```

```
mov ah,type go
```

```
mov bh,size x2
```

```
mov ch,length x3
```

{梁皓答}: AL=1, BL=2, CL=4, AH=0FFH,BH=4, CH=1

{崔文韬问}: 画出示意图, 说明下列变量在内存中如何存放:

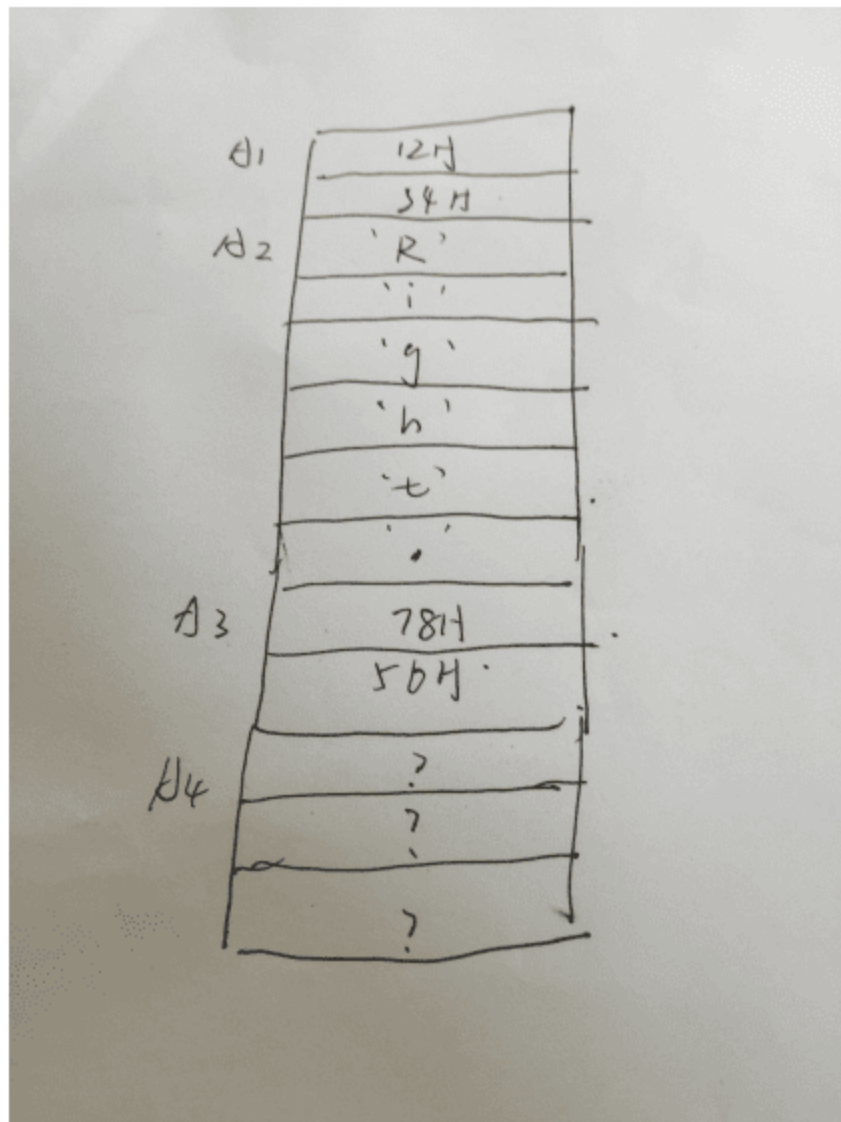
```
a1 db 12h,34h
```

```
a2 db 'Right.'
```

```
a2 db 5678h
```

```
a4 db 3 dup (?)
```

{梁皓答}:



{崔文韬问}：给出完成的汇编语言程序框架：

{刘瑾答}：

assume ds:data,ss:stack,cs:code,es: extra ;声明数据段，堆栈段，代码段的入口地址

data segment ; 数据段

data ends

extra segment ;附加段

extra ends

stack segment stack ; 堆栈段,增加 stack 字段，在代码段中无需进行 ss:sp 的绑定

db 64 dup (0)

stack ends

code segment ; 代码段

start:

mov ax,data

mov ds,ax

mov ax,extra

```
mov es,ax ;代码段中需要完成数据段扩展段与对应段寄存器的绑定
```

```
code ends
```

```
end start
```

{崔文韬问}: 从汇编语言程序返回 dos 有哪几种方法? 最常用的是哪一种?

{刘瑾答}: 从汇编程序返回 DOS 有三种方法: (1) 按程序框架设定的方法返回 (请刘瑾同学增加内容, 具体过程简要说明一下)。(2) 执行 4CH 号 DOS 功能调用。(3) 对于可执行的命令文件 (.COM 文件), 用 INT20H 指令可以直接返回 DOS。第二种最为常用。

{崔文韬问}: DOS 功能调用和 BIOS 中断调用各分那几个步骤?

{刘瑾答}: 1.DOS 系统功能调用可分为以下几个步骤: (1) 功能调用号送到 AH 寄存器中, AH=00—6CH。(2) 入口参数送到指定的寄存器中, 一种功能调用又包含多个子功能, 有些调用不带参数。(3) 执行 INT21H 指令。(4) 得到出口参数, 或将结果显示在 CRT 上。

2.BIOS 中断调用可分为以下几个步骤: (1)功能号送到 AH 中。(2) 设置入口参数。(3) 执行 INTn 指令。(4) 分析出口参数及状态。

{崔文韬问}: 习题 10, 编写汇编程序, 完成如下功能: 参考 p132 页, 例 4.21 和例 4.23

1. 从键盘输入字符串“Please input a number:”,存入 buff 开始的内存单元中。

2. 把内存中从 buff 单元开始存放的字符串显示在屏幕上。

3. {刘玉年答}:

```
data segment
```

```
buff db 50
```

```
db ?
```

```
db 50 dup(?)
```

```
data ends
```

```
code segment
```

```
assume cs:code,ds:data
```

```
start: mov ax,data
```

```
mov ds,ax
```

```
mov dx,offset buff
```

```
mov ah,0ah
int 21h
mov bx,offset buff
mov dx,[bx+1]
mov dh,0;
mov ax,'$'
add bx,dx
mov [bx+2],ax
call crlf
mov dx,offset buff
add dx,2
mov ah,09h
int 21h
mov ax,4c00h
int 21h
crlf: mov dl,0dh
mov ah,02h
int 21h
mov dl,0ah
mov ah,02h
int 21h
ret
code ends
end start
```

```

DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip: 0, Program: DEBUG
~p
AX=0024 BX=0000 CX=007B DX=0016 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=0770 IP=0017 NU UP EI PL NZ NA PO NC
0770:0017 03DA ADD BX,DX
~p
AX=0024 BX=0016 CX=007B DX=0016 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=0770 IP=0019 NU UP EI PL NZ NA PO NC
0770:0019 894702 MOV [BX+02],AX DS:0018=000D
~p
AX=0024 BX=0016 CX=007B DX=0016 SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=0770 IP=001C NU UP EI PL NZ NA PO NC
0770:001C E80F00 CALL 002E
~d ds:0000
076C:0000 32 16 70 6C 65 61 73 65-20 69 6E 70 75 74 20 61 2.please input a
076C:0010 20 6E 75 6D 62 65 72 3A-24 00 00 00 00 00 00 00 number:$.....
076C:0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076C:0030 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
076C:0040 B8 6C 07 8E D8 BA 00 00-B4 0A CD 21 BB 00 00 8B .l.....!....
076C:0050 57 01 B6 00 B8 24 00 03-DA 89 47 02 E8 0F 00 BA W....$....G....
076C:0060 00 00 83 C2 02 B4 09 CD-21 B8 00 4C CD 21 B2 0D .....!..L.!..
076C:0070 B4 02 CD 21 B2 0A B4 02-CD 21 C3 B8 1C 27 50 FF ...!.....!...'P.
~a

```

```

C:\CODE>XITI10.EXE
please input a number:
please input a number:
C:\CODE>

```

{崔文韬问}：习题 11，编写程序实现，在显示器上显示全部标准和扩展 ASCII 码（00~FF）字符，参考 p138 页例 4.33。

{刘玉年答}：

```

code segment
assume cs:code
start: mov cx,00ffh
al: mov bx,00ffh
sub bx,cx
mov dl,bl
mov ah,2h
int 21h
loop al
mov ax,4c00h
int 21h
code ends
end start

```

```
C:\CODE>XITI11.EXE
```

```
  00000000
```

```
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
C:\CODE>
```

{崔文韬问}: 习题 12, 编程实现, 从键盘输入一个 10 进制数字 0~9, 查表求键入数字的七段代码, 存入 DL 中, 并在键入数字之前, 显示提示信息“Please input a number:”。参考 P138 页 4.34。

{刘玉年答}:

```
data segment
```

```
table db 0c0h,0f9h,0a4h,0b0h,99h,92h,82h,0f8h,80h,90h
```

```
list db 'please input a number:', '$'
```

```
data ends ;共阳极
```

```
code segment
```

```
assume cs:code,ds:data
```

```
start: mov ax,data
```

```
mov ds,ax
```

```
mov dx,offset list
```

```
mov ah,9h
```

```
int 21h
```

```
mov ah,01h
```

```
int 21h
```

```
mov bx,offset table
```

```
sub al,'0'
```

```
and ah,0
```

```
add bx,ax
```

```
mov dl,[bx]
```

```
mov ax,4c00h
```

```
int 21h
```

```
code ends
```

```
end start
```



```

AX=096C BX=0000 CX=0051 DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076F IP=000A NU UP EI PL NZ NA PO NC
076F:000A CD21 INT 21
-p
please input a number:
AX=096C BX=0000 CX=0051 DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076F IP=000C NU UP EI PL NZ NA PO NC
076F:000C B401 MOV AH,01
-p
AX=016C BX=0000 CX=0051 DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076F IP=000E NU UP EI PL NZ NA PO NC
076F:000E CD21 INT 21
-p
6
AX=0136 BX=0000 CX=0051 DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=076C ES=075C SS=076B CS=076F IP=0010 NU UP EI PL NZ NA PO NC
076F:0010 BB0000 MOV BX,0000

```

```

;L0d1L9w fCLW1u9fCq u0LwW1jñ

```

```

-b
059E:007E CDSI IML SI
D2=059C E2=052C 22=059B C2=059E IB=007E M0 N6 EI 67 W5 W4 6E WC
BX=4C00 BX=0000 CX=0021 DX=0005 2B=0000 B6=0000 2I=0000 DI=0000

```

{崔文韬问}: 习题 17, 已知数 A=9876, 数 B=6543, 编程求两数之和。

{杨本栋答}:

```

data segment
a0 dw 9876
b0 dw 6543

sum dw 2 dup(0); 保存结果和进位

data end

code segment
assume cs: code,ds: data
main: mov ax, data
mov ds, ax
mov ax, a0
add ax, b0
mov sum,ax ;保存结果到 sum 中
jnc stop; 无进位 跳转
mov sum[1],01h; 有进位
stop: mov ax,4c00h
int 21h
code end
end main

```

{崔文韬答}: 源代码直接复制黏贴过来就可以了。你下面的代码有好几处明显错误: data end, code end。

```

002C:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
002C:0050 CD 21 CB 00 00 00 00 00 00 00 00 00 20 20 20 20 .!.
002C:0060 20 20 20 20 20 20 20 20 20 00 00 00 00 20 20 20 .....
002C:0070 20 20 20 20 20 20 20 20 20 00 00 00 00 00 00 00 .....
002C:0080 00 00 78 69 74 69 31 37 5F 31 2E 65 78 65 0D 52 ..xiti17.1.exe.R
002C:0090 3D 41 30 0D 64 64 72 65 73 73 2E 20 20 46 6F 72 =00.address. For
002C:00A0 20 65 70 61 6D 70 6C 65 3A 0D 20 6F 6E 20 4E 54 example: on NT
002C:00B0 56 44 4D 2C 20 73 70 65 63 69 66 79 20 61 6E 20 UDM, specify an
002C:00C0 69 6E 76 61 6C 69 64 0D 20 6F 6E 6C 79 2E 0D 00 invalid. only...
002C:00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
002C:00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
002C:00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
002C:0100 94 26 0F 19 00 00 00 00 00 00 00 00 00 00 00 .&.....
002C:0110 B8 3C 08 0E D6 A1 00 00 03 06 02 00 A3 04 00 73 .<.....s
002C:0120 06 C7 06 05 00 01 00 B8 00 4C CD 21 .....L.!
-g
Program terminated normally
-a ds:0100
002C:0100 94 26 0F 19 23 40 00 00 00 00 00 00 00 00 00 .&..00.....
002C:0110 B8 3C 08 0E D6 A1 00 00 03 06 02 00 A3 04 00 73 .<.....s
002C:0120 06 C7 06 05 00 01 00 B8 00 4C CD 21 91 00 00 2E .....L.!
002C:0130 89 0E 0F 91 2E 89 26 E1 91 2E 89 36 E3 91 FC 2E .....&...6....
002C:0140 89 0E 48 91 2E C7 06 4A 91 00 00 2E C7 06 5D 91 ..N...J.....l
002C:0150 00 00 2E C7 06 4E 91 00 00 2E C7 06 1A 92 5B 5D .....N.....[
002C:0160 2E C7 06 1C 92 7C 3C 2E C7 06 1E 92 3E 2B 2E C7 .....!<.....>+..
002C:0170 06 20 92 3D 3B E8 83 09 73 13 B8 FF FF 53 26 0B . .;...0....S&.

```

{崔文韬问}: 习题 13, 某一个学生的英语成绩已经存放在 BL 中, 如果低于 60 分, 则显示 F, 如果高于或者等于 85 分, 则显示 G, 否则显示 P, 试编写完整的汇编程序实现该功能。参考流程图图 4.8。

{杨本栋答}: 源代码如下:

```

data segment
pass db 'P',0dh,0ah,'$'
fail db 'F',0dh,0ah,'$'
good db 'G',0dh,0ah,'$'
data ends

code segment
assume cs:code,ds:data
main: mov bl,75
      cmp bl,60
      jnb fail_get
      cmp bl,85
      jae good_get
      mov ax,seg pass
      mov ds,ax
      mov dx,offset pass
      jmp display
fail_get:
      mov ax,seg fail
      mov ds,ax
      mov dx,offset fail
      jmp display

```

```

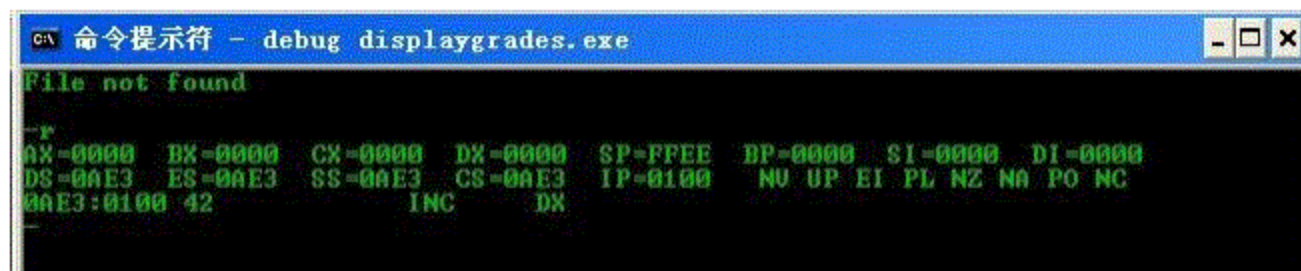
good_get:
mov ax,seg good
mov ds,ax
mov dx,offset good

display:
mov ah,9
int 21h

stop:
mov ax,4c00h
int 21h
code ends
end main

```

已经进行过编译链接 但在 debug 中遇到了问题：出现了 File not found 的提示（见下图）。重新编译链接和建一个新文件都是这样。把代码拷进之前的一个文件中，重新编译链接再调试就没问题了，没想明白为什么？



{崔文韬答}：是在虚拟机里进行的实验吧？我试验了一下，没有你说的现象？提示：debug 之前先输入 dir 命令，查看当前文件夹下是否存在先前生成的可执行文件，例如 displaygrades.exe。file not found 说明当前目录下不存在该文件。

上面程序有问题，我标记了一下。在检查一下。

Dos 文件系统中，文件名长度有显示，不能超过 8 个字符，你的文件名太长了，文件名长度小于等于 8 即可。

{杨本栋答}：之前在 AL 中存的成绩，发上来之后发现要求是 BL 存成绩，改了一下没改全。

那个问题还是没解决，dirdir 命令显示有这个文件。我之前还是可以的，昨天晚上出现的问题，之前建立的没有问题，新建的都不行。

```
C:\ 命令提示符 - debug displaygrades.exe
Definitions File [nul.def]:
LINK : warning L4021: no stack segment

C:\Assemble\Code>DIR
Volume in drive C has no label.
Volume Serial Number is 7053-7270

Directory of C:\Assemble\Code

2016-04-22  19:05    <DIR>          .
2016-04-22  19:05    <DIR>          ..
2016-04-20  21:37             444 01.asm
2016-04-21  21:11             514 chengjidengji.asm
2016-04-21  21:26             518 displaygrades.asm
2016-04-22  19:05             580 displaygrades.exe
2016-04-22  19:05             178 displaygrades.obj
2016-03-04  15:35             330 hello.asm
2016-04-21  21:24             518 xiti17_1.asm
2016-04-21  21:28             580 xiti17_1.exe
2016-04-21  21:28             173 xiti17_1.obj
2016-04-21  20:22             148 xiti17_2.asm
               10 File(s)                3,983 bytes
               2 Dir(s)      1,501,491,200 bytes free

C:\Assemble\Code>debug displaygrades.exe
File not found
```

{崔文韬问}: 习题 14, 在 table 开始的内存字节单元中, 存放了 12 个带符号数, 试编写完整的汇编程序统计其中正数、负数和零的个数, 分别存入 plus, neg 和 zero 单元中。参考例 3.92。

{刘玉年答}:

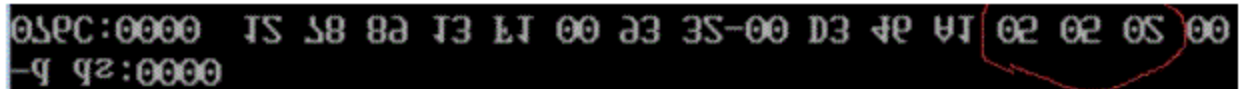
```
data segment
table db 12h,78h,89h,13h,0f1h,00h,93h,32h,00h,0d3h,46h,0a1h
plus db ?
nege db ?
zero db ?
data ends

code segment
assume cs:code,ds:data
start: mov ax,data
mov ds,ax
mov ax,0
mov bx,0
mov cl,0ch
and ch,00h
mov si,offset table
a0: mov al,[si]
cmp al,00h
je a1
test al,80h
jz a2
```

```

inc bh
jmp a3
a1: inc bl
jmp a3
a2: inc ah
jmp a3
a3: inc si
loop a0
mov [plus],ah
mov [nege],bh
mov [zero],bl
mov ax,4c00h
int 21h
code ends
end start

```



0000:0000 15 58 8d 13 11 00 03 35-00 D3 4e 41 02 02 05 00  
 -q 02:0000

{崔文韬问}: 习题 15, 在内存 buff 开始的单元中, 存有一串数据, 58,75,36,42,89, 试编写程序找出其中的最小值存入 min 单元, 并将这个数显示在屏幕上。参考例 4.38.。

{杨本栋答}:

```

stack segment stack
dw 64 dup(?)
stack ends

data segment
buff db 58,75,36,42,89
min db ? ;存最小值
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov cl,4;循环次数 数字个数减1
mov al,buff;
mov bx,1

```

```

loop1:
cmp al,buff[bx];buff[bx]比当前最小值大?
jl next;是, 转 next
mov al,buff[bx];当前值为最小值
next:
inc bx
dec cl
jnz loop1
mov min,al;最小值存入 min
dis_dec: ;以十进制形式显示最小值
mov ah,0
mov bl,10
div bl ;商存 al
mov bl,ah;余数存 bl
add al,30h;商转换为 ASCII
mov dl,al;显示商, 即十位数
mov ah,2
int 21h
add bl,30h;余数转换为 ASCII
mov dl,bl;显示余数, 即个位数
mov ah,2
int 21h
mov ax,4c00h
int 21h
code ends
end start

```

```

-d ds:0000
0B44:0000 3A 4B 24 2A 59 00 00 00-00 00 00 00 00 00 00 00 :K$wY.....
0B44:0010 B8 44 0B 0E D8 B1 04 A0-00 00 BB 01 00 3A 87 00 :D.....:..
0B44:0020 00 7C 04 8A 87 00 00 43-FE C9 75 F1 A2 05 00 B4 :!.....C..u....
0B44:0030 00 B3 0A F6 F3 8A DC 04-30 8A D0 B4 02 CD 21 80 :.....0.....!
0B44:0040 C3 30 8A D3 B4 02 CD 21-B8 00 4C CD 21 3D 75 06 :0.....!..L.?u.
0B44:0050 2E 00 0E 56 91 01 43 E8-00 00 73 C5 AC 2E 00 07 :...U..C...s....
0B44:0060 43 EB BE 4E 2E 09 36 4C-91 2E C6 07 00 2E 09 1E :C..M..6L.....
0B44:0070 5B 91 26 0B 1D 0D 36 5F-91 2E 00 3C 2F 74 36 2E :l.&...6_...</t6.
-g
36
Program terminated normally
-d ds:0000
0B44:0000 3A 4B 24 2A 59 24 00 00-00 00 00 00 00 00 00 00 :K$wY$.
0B44:0010 B8 44 0B 0E D8 B1 04 A0-00 00 BB 01 00 3A 87 00 :D.....:..
0B44:0020 00 7C 04 8A 87 00 00 43-FE C9 75 F1 A2 05 00 B4 :!.....C..u....
0B44:0030 00 B3 0A F6 F3 8A DC 04-30 8A D0 B4 02 CD 21 80 :.....0.....!
0B44:0040 C3 30 8A D3 B4 02 CD 21-B8 00 4C CD 21 3D 75 06 :0.....!..L.?u.
0B44:0050 2E 00 0E 56 91 01 43 E8-00 00 73 C5 AC 2E 00 07 :...U..C...s....
0B44:0060 43 EB BE 4E 2E 09 36 4C-91 2E C6 07 00 2E 09 1E :C..M..6L.....
0B44:0070 5B 91 26 0B 1D 0D 36 5F-91 2E 00 3C 2F 74 36 2E :l.&...6_...</t6.

```

{崔文韬问}: 习题 16, 内存中有一组无符号字节数据, 要求编程按从小到大的顺序排列。参考例 4.40。



{崔文韬问}: 习题 18, 某班有 20 个同学的微机原理成绩存放在 list 开始的单元中, 要求编程先按从高到低的次序排列好, 在求出总分和平均值, 分别存放到 sum 和 aver 开始的单元中。

{崔文韬问}: 习题 19, 编程将后跟\$符的字符串“Go to school.”中的小写字母都改成大写字母。提示: 小写字母比大写字母的 asc 码大 20h, 如‘A’=41h, ‘a’=61h。

{杨本栋答}:

```
stack segment stack
dw 64 dup(?)
stack ends
data segment
buff db 'Go to School'
count equ $-buff
data ends

code segment
assume cs:code,ds:data
start:
mov ax,data
mov ds,ax
mov cl,count;字符串长度
mov bx,0;基地址为 0
loop1:
mov al,buff[bx]
cmp al,61h;<61h
jl next;不是小写字母
cmp al,7ah;>7ah?
jg next;不是小写字母
sub al,20h;是小写字母, 改为大写
mov buff[bx],al;存入原位置
next:
inc bx;基址加 1
dec cl;字符长度减一
jnz loop1
mov ax,4c00h
```



```
int 21h
code ends
end start
```

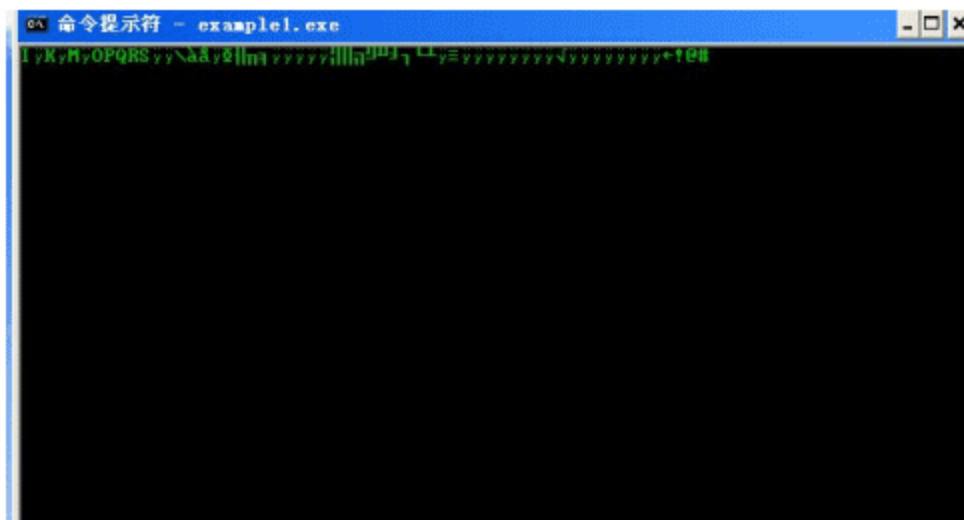
```
0B45:0003 8ED8      MOV     DS,AX
-p
AX=0B44  BX=0000  CX=00B6  DX=0000  SP=0080  BP=0000  SI=0000  DI=0000
DS=0B44  ES=0B2C  SS=0B3C  CS=0B45  IP=0005  NU UP EI PL NZ NA PO NC
0B45:0005 B10C      MOV     CL,0C
-d ds:0000
0B44:0000  47 6F 20 74 6F 20 53 63-68 6F 6F 6C 00 00 00 00  Go to School....
0B44:0010  B8 44 0B 8E D8 B1 0C BB-00 00 8A 87 00 00 3C 61  .D.....<a
0B44:0020  7C 0A 3C 7A 7F 06 2C 20-88 87 00 00 43 FE C9 75  !.<z... ..C..u
0B44:0030  E9 B8 00 4C CD 21 20 75-05 E8 3F 09 EB 26 2E F6  ...L.! u...?..&..
0B44:0040  06 56 91 41 74 1E 4E EB-1B 2E 88 07 3C 3D 75 06  .U.At.N.....<=u.
0B44:0050  2E 80 0E 56 91 01 43 E8-08 0A 73 C5 AC 2E 88 07  ...U..C....s.....
0B44:0060  43 EB BE 4E 2E 89 36 4C-91 2E C6 07 00 2E 89 1E  C..N..6L.....
0B44:0070  5B 91 26 8B 1D 8D 36 5F-91 2E 80 3C 2F 74 36 2E  l.&...6_....</t6.
-g
Program terminated normally
-d ds:0000
0B44:0000  47 4F 20 54 4F 20 53 43-48 4F 4F 4C 00 00 00 00  GO TO SCHOOL....
0B44:0010  B8 44 0B 8E D8 B1 0C BB-00 00 8A 87 00 00 3C 61  .D.....<a
0B44:0020  7C 0A 3C 7A 7F 06 2C 20-88 87 00 00 43 FE C9 75  !.<z... ..C..u
0B44:0030  E9 B8 00 4C CD 21 20 75-05 E8 3F 09 EB 26 2E F6  ...L.! u...?..&..
0B44:0040  06 56 91 41 74 1E 4E EB-1B 2E 88 07 3C 3D 75 06  .U.At.N.....<=u.
0B44:0050  2E 80 0E 56 91 01 43 E8-08 0A 73 C5 AC 2E 88 07  ...U..C....s.....
0B44:0060  43 EB BE 4E 2E 89 36 4C-91 2E C6 07 00 2E 89 1E  C..N..6L.....
0B44:0070  5B 91 26 8B 1D 8D 36 5F-91 2E 80 3C 2F 74 36 2E  l.&...6_....</t6.
-
```

{崔文韬问}：习题 20，编程将存放在 al 中的无符号二进制数，转换为 16 进制数，再转换为 asc 码并显示在屏幕上。参考例 4.43。

{崔文韬问}：习题 21，将 bx 中的 16 进制数（<9999）转换成 4 位压缩 BCD 数，存入 cx 中。参考例 4.44。

{陈志坤问}：老师，虚拟机正确编译链接完，运行后出现这一堆乱码乱码怎么回事？把你源代码贴上来我看看。

ppt 上的例子，能正常通过编译链接，代码应该没有问题，可能是虚拟机的原因？不是用 dosbox——



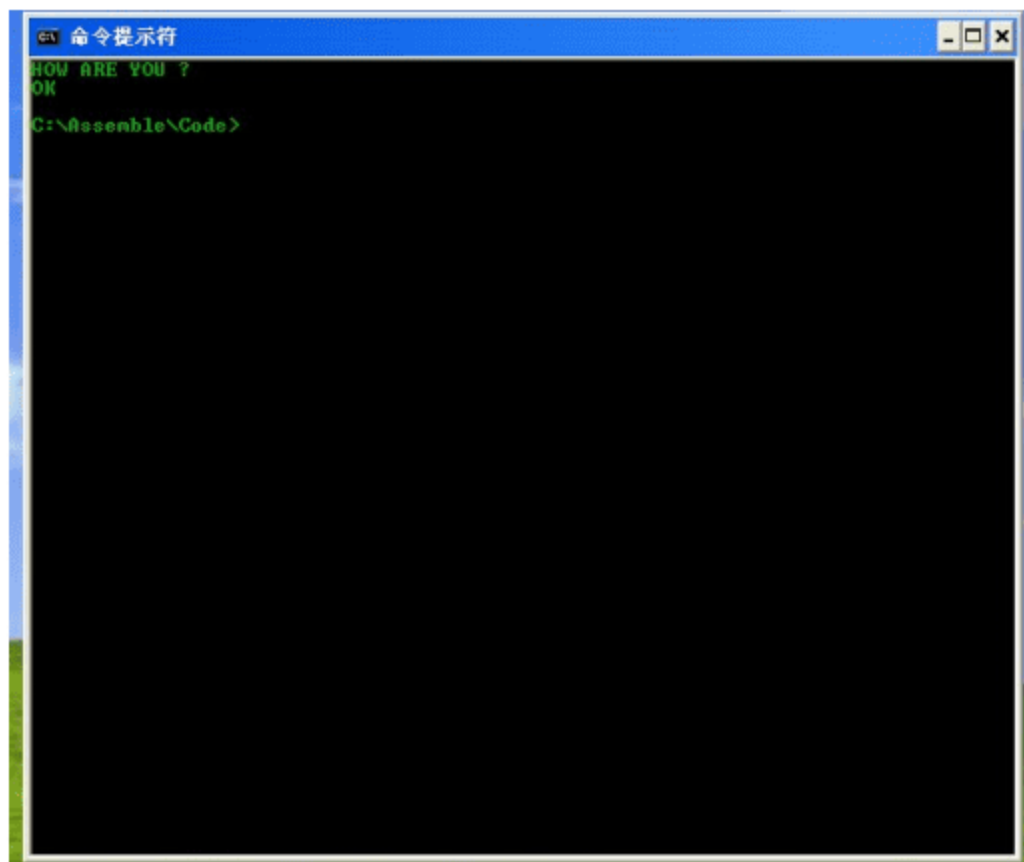
```

1 D   SEGMENT
2 D1  DB 'HOW ARE YOU ?',0DH,0AH,'$'
3 D2  DB 'OK',0DH,0AH,'$'
4 D   ENDS
5 code SEGMENT
6     ASSUME CS:code,DS:D ;说明代码段、数据段
7 BG: MOV AX,D
8     MOV DS,AX
9     MOV AH,9
10    INT 21H ;显示 "HOW ARE YOU ?"
11    MOV AH,8
12    INT 21H ;不显示方式读一字符到AL
13    CMP AL,'Y'
14    JNE NEXT ;不等则转
15    LEA DX,D2
16    MOV AH,9
17    INT 21H
18 NEXT: MOV AH,4CH
19        INT 21H
20 code ENDS
21 END BG

```

你的源代码有问题，自己好好看看，是不是漏了一行？？？编程是个细致活^\_^。加油！！！分析一下，按照你上面写的源代码，程序打印的内容是否正确或者执行过程是怎样的？如何通过 debug 验证一下？？我在虚拟机里实验了，没有问题。

噢噢，好的我再看看。谢谢老师



```

命令提示符
HOW ARE YOU ?
OK
C:\Assemble\Code>

```

{陈志坤答：}示例验证：

```

CA 命令提示符 - debug new1.exe
DS=0B3B ES=0B2B SS=0B3B CS=0B3D IP=001E NU UP EI PL NZ AC PE NC
0B3D:001E CD21 INT 21
~p
Program terminated normally
~p
C:\Assemble\Code>debug new1.exe
~r
AX=0000 BX=0000 CX=0040 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=0B2B ES=0B2B SS=0B3B CS=0B3D IP=0000 NU UP EI PL NZ NA PO NC
0B3D:0000 B83B0B MOV AX,0B3B
~u
0B3D:0000 B83B0B MOV AX,0B3B
0B3D:0003 8ED8 MOV DS,AX
0B3D:0005 B80000 MOV DX,0000
0B3D:0008 B409 MOV AH,09
0B3D:000A CD21 INT 21
0B3D:000C B408 MOV AH,08
0B3D:000E CD21 INT 21
0B3D:0010 3C59 CMP AL,59
0B3D:0012 7508 JNZ 001C
0B3D:0014 8D161000 LEA DX,[0010]
0B3D:0018 B409 MOV AH,09
0B3D:001A CD21 INT 21
0B3D:001C B44C MOV AH,4C
0B3D:001E CD21 INT 21

```

```

CA 命令提示符 - debug new1.exe
0B3D:0012 7508 JNZ 001C
0B3D:0014 8D161000 LEA DX,[0010]
0B3D:0018 B409 MOV AH,09
0B3D:001A CD21 INT 21
0B3D:001C B44C MOV AH,4C
0B3D:001E CD21 INT 21
~dds:0000 13f
0B2B:0000 CD 20 FF 9F 00 9A F0 FE-1D F0 4F 03 47 05 8A 03 . . . . .0.G...
0B2B:0010 47 05 17 03 47 05 36 05-01 01 01 00 02 FF FF FF G...G.6.....
0B2B:0020 FF FF FF FF FF FF FF FF FF FF FF F4 0A 4C 01 . . . . .L.
0B2B:0030 07 0A 14 00 18 00 2B 0B-FF FF FF FF 00 00 00 00 . . . . .+.....
0B2B:0040 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
0B2B:0050 CD 21 CB 00 00 00 00 00 00 00 00 00 20 20 20 . . . . .
0B2B:0060 20 20 20 20 20 20 20 20 20 00 00 00 00 20 20 20 . . . . .
0B2B:0070 20 20 20 20 20 20 20 20 20 00 00 00 00 00 00 00 . . . . .
0B2B:0080 00 0D 6E 65 77 31 2E 65-70 65 0D 41 53 54 45 52 ..new1.exe.ASTER
0B2B:0090 3D 41 30 0D 64 64 72 65-73 73 2E 20 20 46 6F 72 =A0.address. For
0B2B:00A0 20 65 70 61 6D 70 6C 65-3A 0D 20 6F 6E 20 4E 54 example: on NT
0B2B:00B0 56 44 4D 2C 20 73 70 65-63 69 66 79 20 61 6E 20 UDM, specify an
0B2B:00C0 69 6E 76 61 6C 69 64 0D-20 6F 6E 6C 79 2E 0D 00 invalid. only...
0B2B:00D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
0B2B:00E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
0B2B:00F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 . . . . .
0B2B:0100 40 4F 57 20 41 52 45 20-59 4F 55 20 3F 0D 0A 24 NOW ARE YOU ?..$
0B2B:0110 4F 4B 0D 0A 24 00 00 00 00 00 00 00 00 00 00 . . . . .
0B2B:0120 B0 3B 0B 0E D0 DA 00 00 00 00 00 00 00 00 00 . . . . .
0B2B:0130 3C 59 75 00 0D 16 10 00 00 00 00 00 00 00 00 . . . . .

```

之前虚拟机乱码应该是我自己代码有问题。。。你上面贴的代码里少了一行，取字符串的偏移量 mov dx,offset dl。

{刘玉年问}：在定义堆栈段时：stack segment stack.....后面这个组合类型 stack，到底能不能省略呢？课本上一直在强调不能省略，但是没说为什么。而课件是又说可以省略，只是影响 ss 段寄存器值得装入。

(2)若在段定义伪指令的组合类型中，未选用“STACK”参数项，或在程序中要调换另一个堆栈段，可用类似于DS，ES的装入办法，且需几条指令来实现对SS和SP的装入。例如：

```
STACK1    SEGMENT
ST DW 50H DUP(?)
TOP EQU LENGTH ST; 堆栈的长度
STACK1    ENDS
:
:
CODE      SEGMENT
:
:
MOV  AX, STACK1
MOV  SS, AX
MOV  SP, OFFSET TOP
```

上述示例中，假设STACK1段是程序中要使用的堆栈段(50H个字)，那么TOP就是该堆栈的初始堆栈顶部。用前两条指令把堆栈段的段基值装入SS后，紧接着必须用一条指令初始化堆栈指针SP(在示例中(SP) = 100H)。中间不要插入另外的指令。

{崔文韬答}：可以省略。按照课本要求，不省略 stack 字段，代码更简洁，使用堆栈更方便，编译连接后也不会出现 warning: no stack segment. 警告。实验代码如下：

;加入 stack 字段，不需要需要在代码段中完成堆栈的初始化，即绑定 ss:sp

```
assume cs:code,ss:stack
```

```
stack segment stack
```

```
db 16 dup(255) ;为了清除地观察堆栈段位置，初始化为 255.
```

```
stack ends
```

```
code segment
```

```
start:
```

```
mov ax,12
```

```
mov bx,13
```

```
mov cx,14
```

```
push ax
```

```
push bx
```

```
push cx
```

```
mov ax,4c00h
```

```
int 21h
```

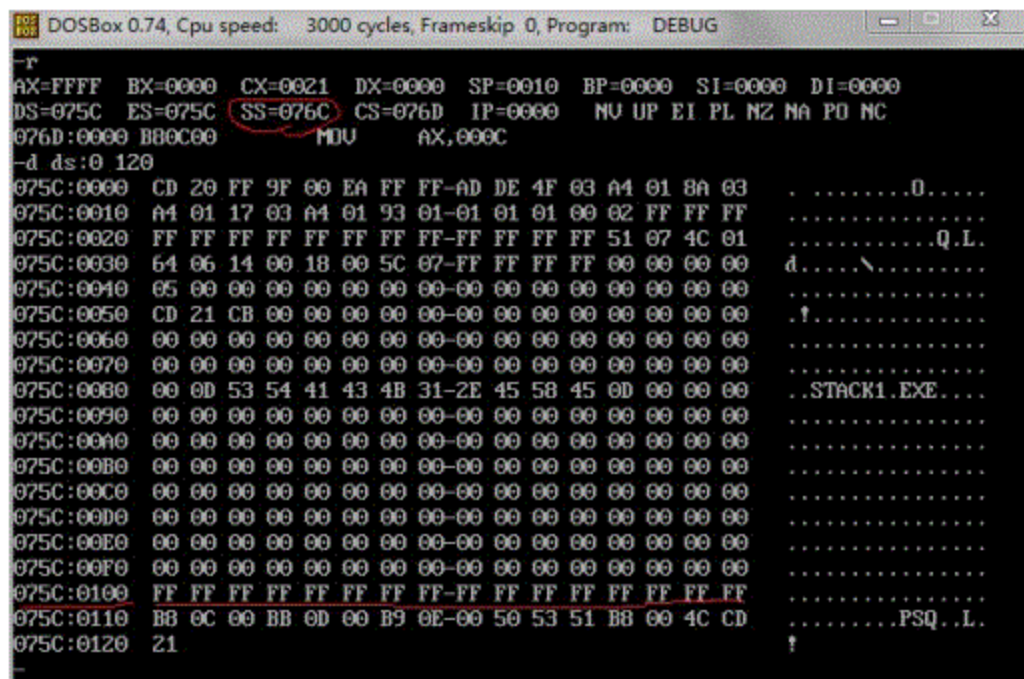
```
code ends
```



end start

编译连接后，debug 该程序，使用 d ds:0 120 查看整个内存中程序的存在形式。截图如下：

程序加载到内存后，ss: sp 自动绑定到设定好的堆栈段中。



如果不加 stack 字段，则需要在代码段中完成 ss 和 sp 的绑定，同时编译连接时会出现 warning: no stack segment 警告信息。实验代码如下：

；省略 stack 字段，需要在代码段中完成堆栈的初始化，即绑定 ss:sp

```
assume cs:code,ss:stack
```

```
stack segment
```

```
db 16 dup(255) ;给清楚看到堆栈位置，初始化为 255
```

```
stack ends
```

```
code segment
```

```
start:
```

```
mov ax,stack ;完成 ss 和 sp 的绑定，如果没有这三行代码，将无法正确使用 push 和 pop 指令
```

```
mov ss,ax
```

```
mov sp,16
```

```
mov ax,12
```

```
mov bx,13
```

```
mov cx,14
```

```
push ax
```

```
push bx
```

```
push cx
```

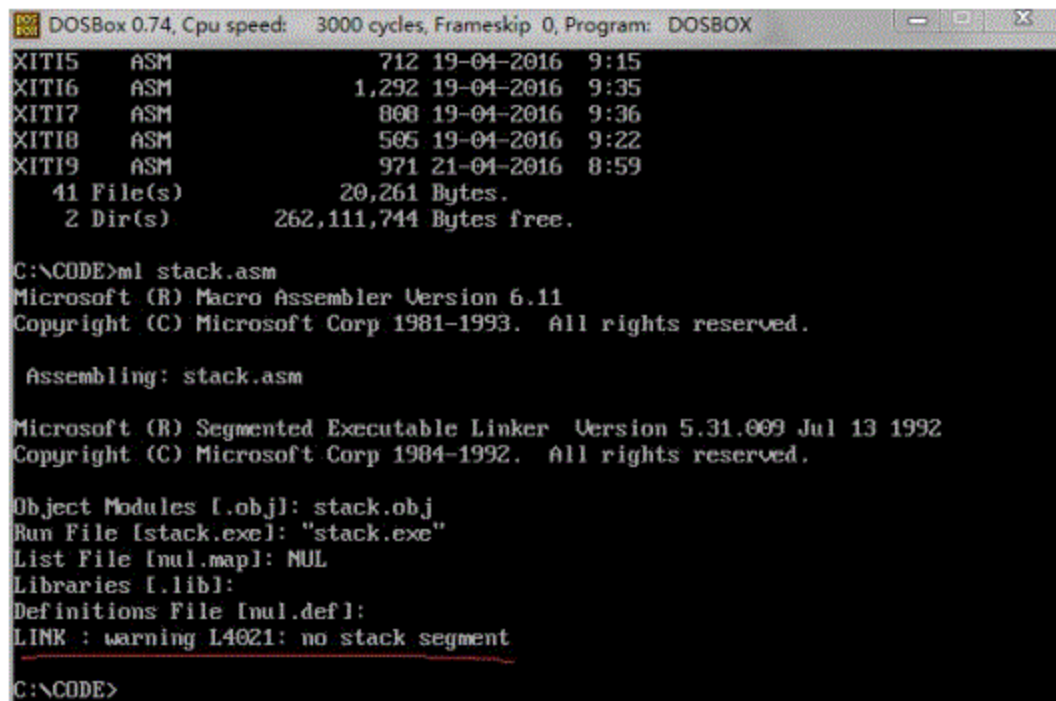
```
mov ax,4c00h
```

```
int 21h
```

```
code ends
```

```
end start
```

编译连接程序后，会出现警告信息。实验截图如下：出现 warning 警告。



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX
XITI5   ASM                712 19-04-2016  9:15
XITI6   ASM             1,292 19-04-2016  9:35
XITI7   ASM                808 19-04-2016  9:36
XITI8   ASM                505 19-04-2016  9:22
XITI9   ASM                971 21-04-2016  8:59
  41 File(s)              20,261 Bytes.
   2 Dir(s)             262,111,744 Bytes free.

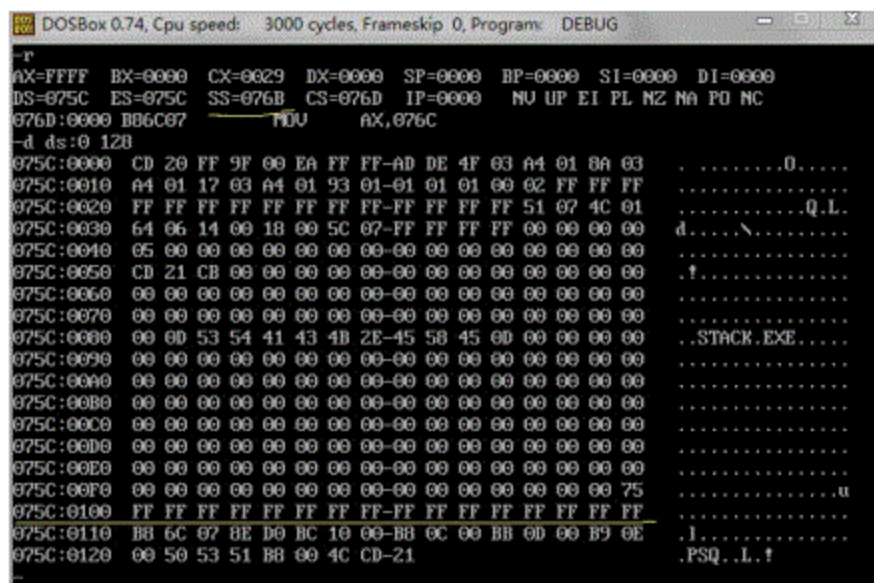
C:\CODE>ml stack.asm
Microsoft (R) Macro Assembler Version 6.11
Copyright (C) Microsoft Corp 1981-1993. All rights reserved.

Assembling: stack.asm

Microsoft (R) Segmented Executable Linker Version 5.31.009 Jul 13 1992
Copyright (C) Microsoft Corp 1984-1992. All rights reserved.

Object Modules [.obj]: stack.obj
Run File [stack.exe]: "stack.exe"
List File [nul.map]: NUL
Libraries [.lib]:
Definitions File [nul.def]:
LINK : warning L4021: no stack segment
C:\CODE>
```

使用 debug 加载调试程序，截图如下：



```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
-r
AX=00FF BX=0000 CX=0029 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=075C ES=075C SS=076B CS=076D IP=0000  NU UP EI PL NZ NA PO NC
076D:0000 B8C07  MOV     AX,076C
-d ds:0 12B
075C:0000 CD 20 FF 9F 00 EA FF FF-AD DE 4F 03 A4 01 8A 03  . . . . .0. . . .
075C:0010 A4 01 17 03 A4 01 93 01-01 01 01 00 02 FF FF FF  . . . . .Q.L. . .
075C:0020 FF FF FF FF FF FF FF FF-FF FF FF FF 51 07 4C 01  . . . . .N. . . .
075C:0030 64 06 14 00 18 00 5C 07-FF FF FF FF 00 00 00 00  . . . . . . . . .
075C:0040 05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:0080 00 00 53 54 41 43 4B 2E-45 58 45 00 00 00 00 00  . . . . .STACK.EXE.
075C:0090 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:00A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:00B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:00C0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:00D0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:00E0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  . . . . . . . . .
075C:00F0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 75  . . . . . . . . .u
075C:0100 FF FF FF FF FF FF FF FF-FF FF FF FF FF FF FF  . . . . . . . . .
075C:0110 B8 6C 07 8E D0 BC 10 00-B8 0C 00 BB 0D 00 B9 0E  . . . . . . . . .
075C:0120 00 50 53 51 B8 00 4C CD-21  . . . . .PSQ.L.!
```

程序加载后，ss: sp 为 076B: 0000,并没有指向我们开辟的堆栈段中。使用 t 命令执行，代码段中的三条堆栈设置指令后，实验截图如下：

```
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
AX=076C BX=0000 CX=0029 DX=0000 SP=0010 BP=0000 SI=0000 DI=0000
DS=075C ES=075C SS=076C CS=076D IP=0000 NU UP EI PL NZ NA PO NC
076D:0000 B0C000 MOV AX,000C
-d ds:0 12B
075C:0000 CD 20 FF 9F 00 EA FF FF-AD DE 4F 03 A4 01 8A 03 . . . . .0. . . .
075C:0010 A4 01 17 03 A4 01 93 01-01 01 01 00 02 FF FF FF . . . . .Q. . I
075C:0020 FF FF FF FF FF FF FF FF-FF FF FF FF 51 07 F1 49 . . . . .\ . . . .
075C:0030 A4 01 14 00 18 00 5C 07-FF FF FF FF 00 00 00 00 . . . . .
075C:0040 05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:0050 CD 21 CB 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:0060 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:0070 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:0080 00 00 53 54 41 43 4B 2E-45 5B 45 0D 00 00 00 00 . . . . .STACK.EXE. . . .
075C:0090 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:00A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:00B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:00C0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:00D0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:00E0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 . . . . .
075C:00F0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 75 . . . . .u
075C:0100 FF FF FF FF FF FF 6C 07-00 00 00 00 00 07 A4 01 . . . . .l . . . .m . .
075C:0110 BB 6C 07 0E D0 BC 10 00-BB 0C 00 BB 0D 00 B9 0E . . . . .l . . . . .
075C:0120 00 50 53 51 B8 00 4C CD-21 . . . . .PSQ. .L. !
```

经过三条指令完成对 ss 和 sp 的设置后，现在 ss 和 sp 指向了 076c 和 0010h，是我们在系统中开辟的堆栈区域。

{王浩问}:LOOP A0 ; MOV [DI],BL 其中 A0 为某一代码段，DI 为数据的指针，后面的 MOV 语句是循环的结束标志吗？是的话为什么能起到中断作用？谢谢老师 o(^o^)o

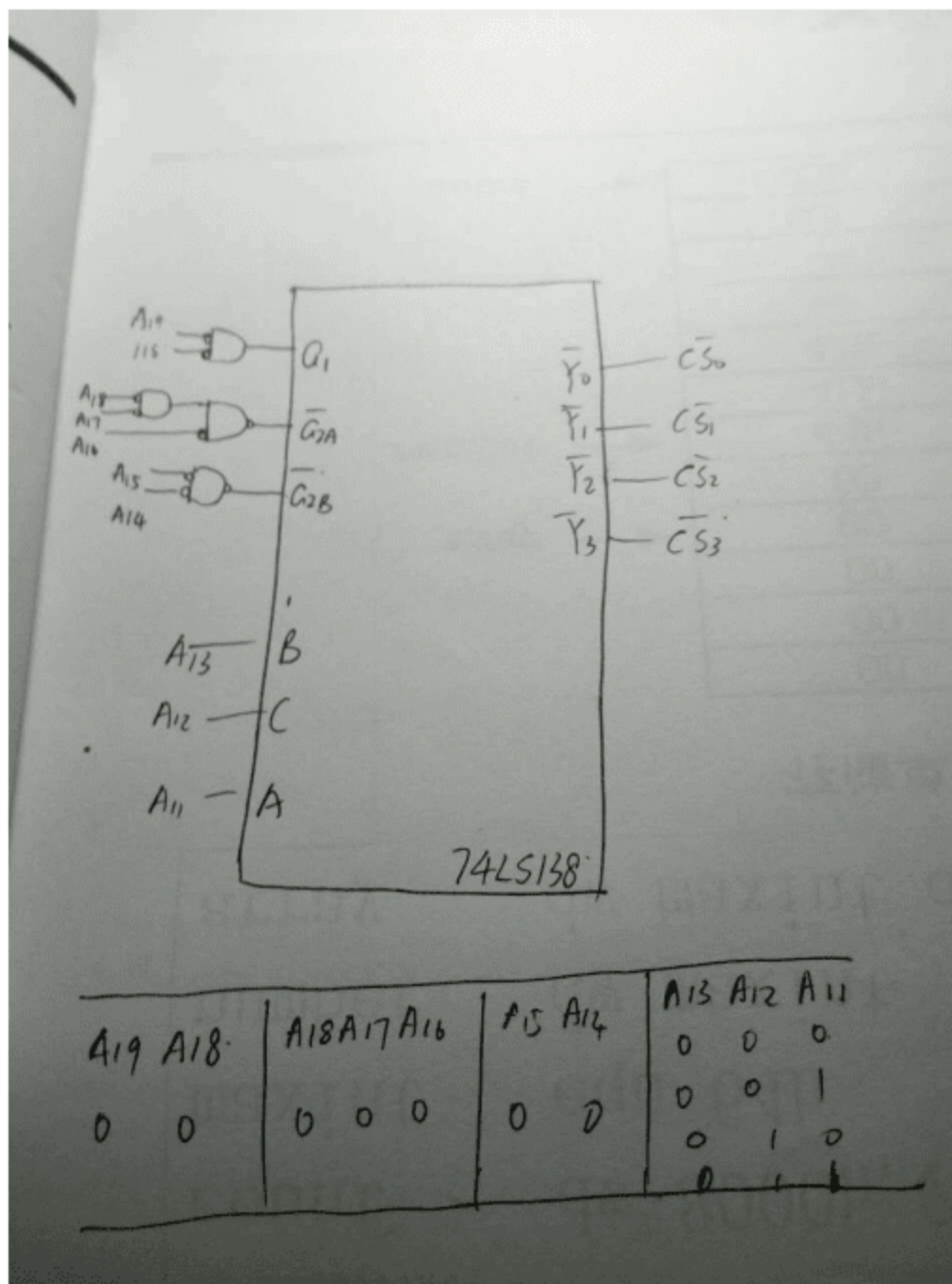
{崔文韬答}:mov 指令不是循环结束标志，loop 循环操作结束是受 cx 的数值控制的。循环结束与中断没有关系。我觉得你应该是有其他问题。你可以把你的问题完整的叙述以下。

## 第五章 存储器

{崔文韬问}: 课后习题 15，在一个有 20 位地址线的系统中，采用 2KX4 的 SRAM 芯片构成容量为 8KB 的 8 位存储器，要求采用全译码方式，请画出该存储器系统的示意图，并回答：共需要（8）块 RAM 芯片，必须将地址（A0）~（A10）连接到每个存储器芯片上，并用地址线（A11）~（A19）作为地址译码器的输入，需要译码器产生（4）个片选信号。（参考图 5.19 和图 5.21）

{郑颖改}:





{崔文韬问}: 74LS138 译码器还有输入 D 引脚???

{李万里答}: 2K 等于 2 的 11 次方, 并联的 SRAM 地址线为 A0~A10, 使用全译码法, A11~A19 为地址译码输入。

{崔文韬问}: 课后习题 16, 对于图 5.22 的部分译码方法, 若将存储器改为 8KX8 位的 6264EPROM 芯片, 译码仍然采用 74LS138, 参与译码的地址线仍是 A0~A17, 试参考改图设计出新的译码方案, 并列出一组连续的可用地址范围。

{郑颖改}:

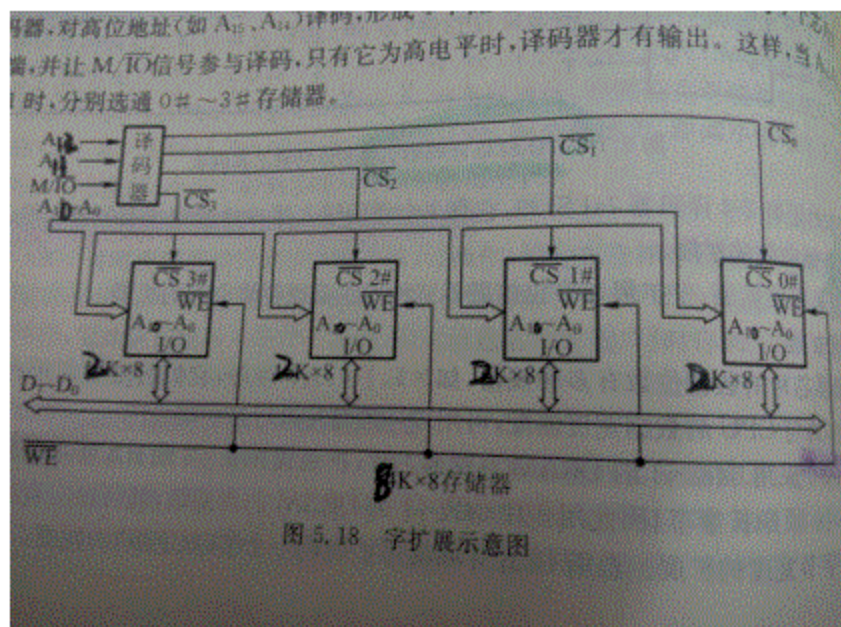
{崔文韬答}: 请重新做这题, 仔细看清题目要求, 参与地址译码的是 A17~A0

{李万里答}:

芯片	$A_{16} A_{17}$	$A_{15} \sim A_{13}$	$A_{12} \sim A_0$	可用地址范围
1	10	000	全0~全1	20000~20FFFH
2	10	001	全0~全1	21000~21FFFH
3	10	010	全0~全1	22000~22FFFH
4	10	011	全0~全1	23000~23FFFH

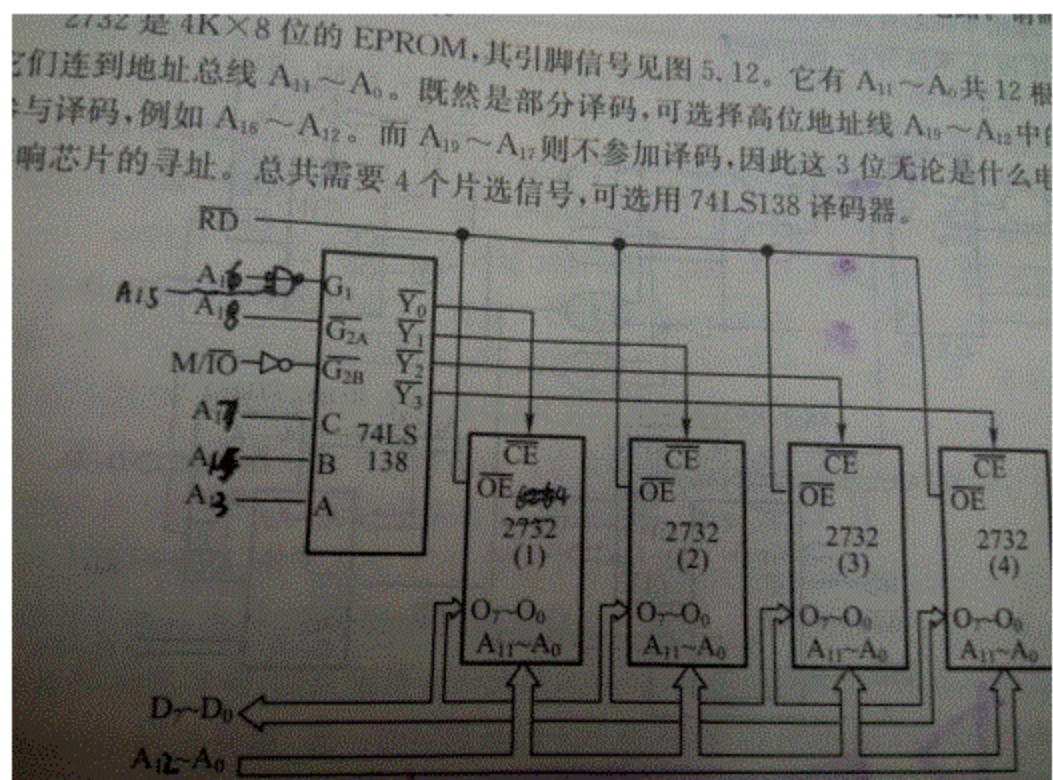
{崔文韬问}: 课后习题 17, 用若干 2KX8 的 RAM 芯片, 扩展成 8KX8 的存储器, 画出扩展后的存储器示意图, 参考例 5.3。

{郑颖答}:



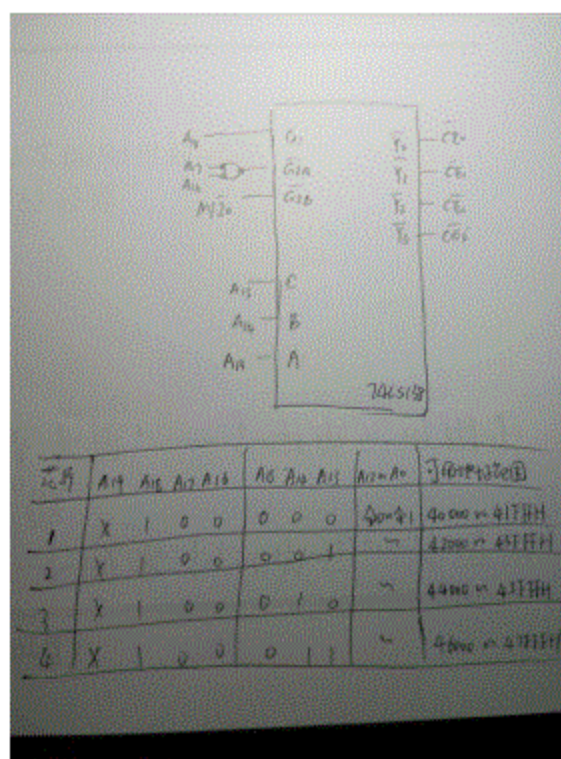
{崔文韬问}: 课后习题 18, 用 8KX8 的 RAM 存储器芯片, 构成 32KX8 的存储器, 存储器的起始地址为 18000H, 要求各存储器芯片地址连续, 用 74LS138 作为译码器, 系统中只用到了地址总线  $A_{18} \sim A_0$ , 采用部分译码法设计译码电路。试画出硬件连线图, 并列表说明每块芯片的地址范围, 参考例 5.7。

{郑颖答}:



{崔文韬答}: 请重新做一下这题, 好像不正确呀??

{郑颖改}:



## 第六章 IO 接口和并行接口芯片 8255A

{崔文韬问}: 课后习题第 2 题

{刘一萱答}: (1) 在接口电路中, CPU 与外设传送的信息 (包括数据信息, 状态信息和控制信息) 分别进入不同的寄存器, 这些寄存器和它们的控制逻辑统称为 I/O 端口。

(2) 数据端口, 状态端口, 命令端口。



(3) 存储器映象寻址方式和 I/O 指令寻址方式。

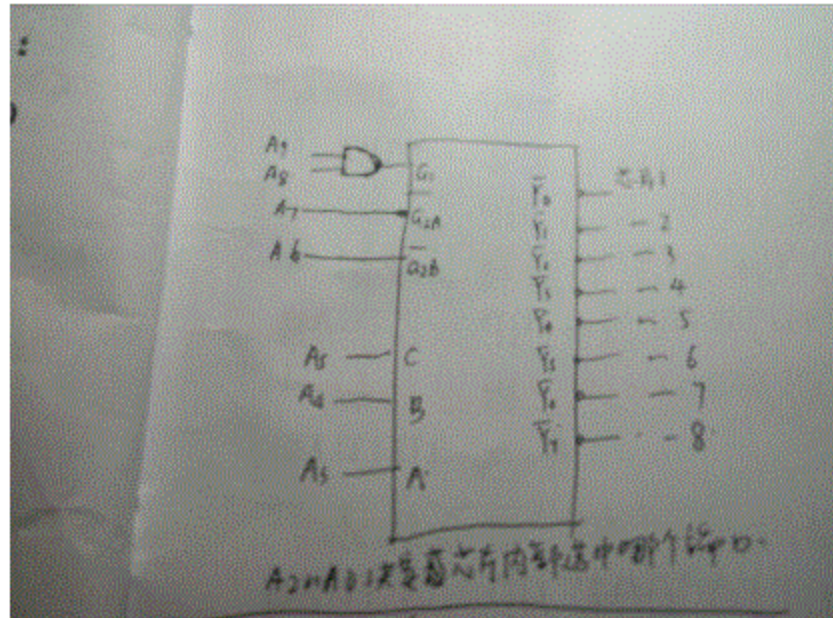
(4) 8086/8088CPU 常用 I/O 指令寻址方式。

{崔文韬问}: 课后习题第 3 题

{刘一萱答}: CPU 与外设间的数据传送方式主要有三种: 程序控制方式, 中断方式, DMA 方式。

{崔文韬问}: 课后习题第 6 题

{郑颖改}:



$A_9 A_8 A_7 A_6$	$A_5 A_4 A_3$	$A_2 A_1 A_0$	访问地址	接芯片
1 1 0 0	0 0 0	全 0 (令)	300H ~ 307H	1
1 1 0 0	0 0 1	:	308H ~ 30FH	2
1 1 0 0	0 1 0	:	310H ~ 317H	3
1 1 0 0	0 1 1	:	318H ~ 31FH	4
1 1 0 0	1 0 0	:	320H ~ 327H	5
1 1 0 0	1 0 1	:	328H ~ 32FH	6
1 1 0 0	1 1 0	:	330H ~ 337H	7
1 1 0 0	1 1 1	:	338H ~ 33FH	8

{崔文韬答}: A8A9 后接的逻辑电路正确吗? 其输出在 A9A8=11 时能是 1?

{崔文韬问}: 课后习题第 8 题

{刘一萱答}: 8255A 具有三种基本的工作方式

方式 0: 基本输入输出方式, 适用场合: 不需要用应答信号的简单输入输出场合。

方式 1: 选通输入/输出方式, 适用场合: A,B 口作为数据口, 均可工作于输入或输出方式。

方式 2: 双向选通传送方式, 适用场合: 在主机和软盘驱动器交换数据时可采用。

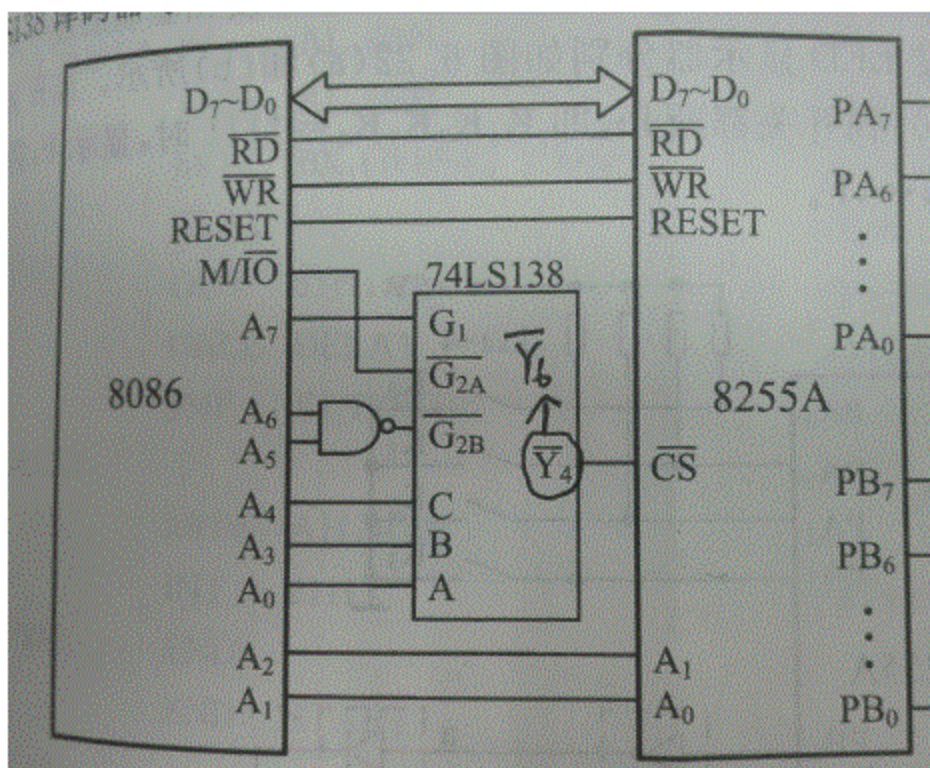
A 口可工作于: 方式 0, 方式 1, 方式 2。B 口可工作于: 方式 0, 方式 1。C 口可工作于: 方式 0。

{崔文韬问}: 课后习题第 9 题

{郑颖答}: 都写入控制字寄存器。用 D7 位加以区分, 方式控制字的 D7 位为 1, 置位/复位控制字的 D7 位为 0。

{崔文韬问}: 课后习题第 10 题

{何林松答}: A 口地址为 0F8H, B 口地址为 0FAH, C 口地址为 0FCH, 控制字寄存器端口地址为 0FEH。



当  $A_7A_6A_5 = 111$ ,  $A_4A_3A_2 = 110$  时,  $(Y_6 \text{ 非}) = 0$ , 选中 8255A

{崔文韬问}: 课后习题第 11 题

{张多睿答}: MOV DX,86H

MOV AL,10001010B

OUT DX,AL

{崔文韬问}: 课后习题第 12 题

OUT 86H,AL ;置 PC6 为低电平

JMP TEST\_LE

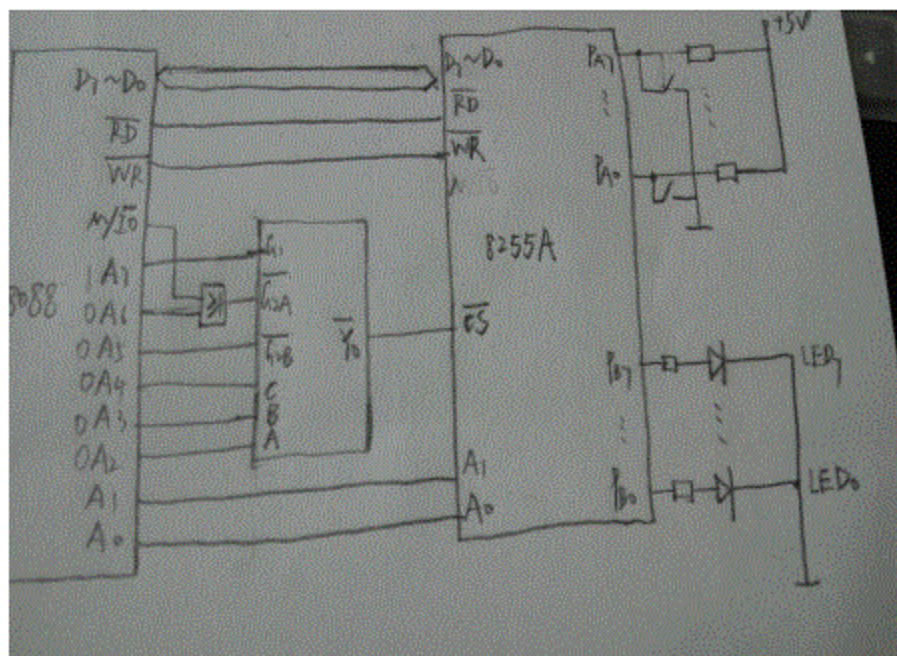


TABLE DB 3fH,06H,5bH,4fH,66H,6dH,7dH,07H

```
DB 7fH,6fH,77H,7cH,39H,5eH,79H,71H
DATA ENDS
CODE SEGMENT
ASSUME CS:CODE, DS:DATA
MOVE AL,90H
OUT 63H,AL
IN _PORTA:IN AL,60H
AND AL,0FH
MOV BF,OFFSET TABLE
XLAT
OUT 61H,AL
CALL DELAY
JMP IN _PORTA
DELAY: MOV AH,4CH
INT 21H
CODE: ENDS
END START
```

---

## 第七章 可编程计数器定时器及应用

{崔文韬问}: 课后习题第一题

{梁皓答}: 一-8253 具有 3 个独立的 16 位计数器通道, 每个计数通道均可以工作于 6 种工作方式

{崔文韬问}: 课后习题第三题

{梁皓答}: (1) 写入控制字 (2) 写入计数初值

{崔文韬问}: 课后习题第四题

{王金鑫答}: 将图 7-9 中的 74LS138 中的 Y4 改为 Y0 即可。

;通道 0 初始化

MOV AL,00110111B ;方式 3, BCD 计数, 初值 2000

OUT 306H,AL



```

MOV AL,00H
OUT 300H,AL
MOV AL,20H
OUT 300H,AL
;通道 1 初始化
MOV AL,01110100B ;方式 2，二进制计数，初值 20000
MOV 306H,AL
MOV AL,20H
OUT 302H,AL
MOV AL,4EH
OUT 302H,AL
;通道 2 初始化
MOV AL,10110011 ;方式 1，BCD 计数，初值 800
OUT 306H,AL
MOV AL,00H
OUT 304H,AL
MOV AL,08H
OUT 304H,AL

```

{崔文韬问}：课后习题第五题

{王金鑫问}：如何分频？

{王金鑫答}：假设时钟五分频 时钟信号频率为 1MHz。

```

;通道 0 初始化
MOV AL,00110111B ; 方式 3，BCD 计数，初值 1000
OUT 43H,AL
MOV AL,00H
OUT 40H,AL
kMOV AL,10H
OUT 40H,AL
;通道 1 初始化
MOV AL,01110111B ; 方式 3，BCD 计数，初值 8000
MOV 43H,AL
MOV AL,00H
OUT 41H,AL
MOV AL,80H

```

## 第八章 中断和可编程中断控制器 8259A

{崔文韬问}: 课后习题第一题及第二题

{梁皓答}: 一: 中断是指计算机在执行正常程序的过程中, 由于某些事件的发生, 需要暂时中止当前程序的运行, 转到中断处理程序去处理临时发生的事件, 处理完之后又恢复到原来的程序的运行, 这个过程叫做中断。

二: 引起中断的原因或能发出中断请求的来源叫做中断源。8086 分为两种中断源, 一种是外部中断或硬件中断, 另一种是内部中断或者软件中断。

{崔文韬问}: 课后习题第三题

{梁皓答}: 从 NMI 引脚引入的是不可屏蔽中断, 从 INTR 引脚引入的中断请求是可屏蔽中断。内部中断分为 (1) 外部中断 (2) 内部中断 (3) 溢出中断 (4) 软中断指令 (5) 断点中断

{崔文韬问}: 课后习题第四题

{梁皓答}: 每类中断有一个入口地址需要 4 个字节储存 CS 和 IP, 256 类中断入口地址要占据 1k 字节, 他们存在内存 0000~003FFH。

{崔文韬问}: 课后习题第五题

{梁皓答}: 除法错中断, 单步中断

{王金鑫答}: 专用中断: 除法错中断, 单步中断, NMI 中断, 断点中断, 溢出中断。

00H,04H,08H,0CH,10H 开始的 4 个连续单元中。

20H,24H,28H,2CH,30H,34H,38H,3CH 开始的 4 个连续单元中。

{崔文韬问}: 课后习题第六题

{王金鑫答}:

10H	16
	00
12H	85
	04

{崔文韬问}：课后习题第七题

{王金鑫答}：中断类型号 10H。入口地址=D169:240BH。

曲洋答：入口地址：D169:240BH

{刘一萱问}：入口地址是如何算的？

{崔文韬问}：课后习题第八题

{王金鑫答}：从高到低为

除法错，INT n、INTO

NMI

INTR

单步中断

{崔文韬问}：课后习题第九题

{姚胜答}：IR2 和 IR5 同时提出中断请求时，先响应优先级高的 IR2。在 IR2 的中断服务器中用 STI 指令开中断，允许更高级的中断进入。

{崔文韬问}：8259A 内部有哪些寄存器？主要功能有哪些？

{姚胜答}：中断请求寄存器 IRR：用来存放从外部 IR7—IR0 引脚上引入的所有中断请求信号。

中断屏蔽寄存器 IMR：用于存放中断屏蔽信号，有选择地禁止某些设备请求中断。

中断服务寄存器 ISR：用来保存当前正在处理的中断请求信号。

{崔文韬问}：课后习题第 15 题

{张衷豪答}：ICW2 的编程设置如下：

MOV AL,00001000B

OUT 21H,AL

OCW1 的编程设置如下：

MOV AL,00111011B

OUT 21H,AL ;允许中断设为 0, 不允许则设为 1

{崔文韬问}: 课后习题第 16 题

{姚胜答}: (1) MOV AL,20H ;OCW2 的 EOI 命令

OUT 20H,AL ; 发 EOI 命令

(2) MOV AL,01100011B ;OCW2 的 SEOI 命令, L2-L0=011 (IR3)

OUT 20H,AL ;将 IS3 清 0, 结束 3 级中断

{崔文韬问}: 课后习题第 17 题

{张衷豪答}:

MOV AL,00001010B

OUT 0A0H,AL

IN AL,0A0H ;获取中断请求寄存器 IRR 的内容

MOV AL,00001100B

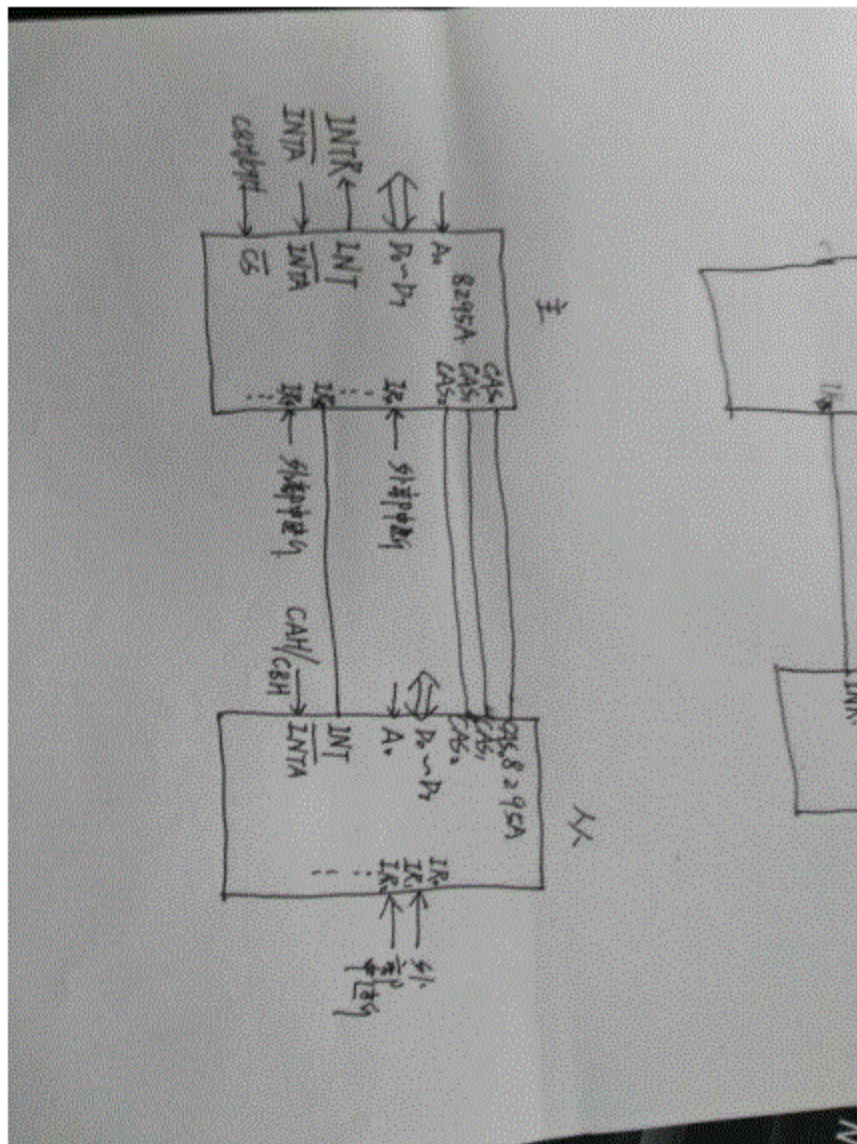
OUT 0A0H,AL

IN AL,0A0H ;获取中断查询字

当中断查询字=1000 0010B 时, 表示有中断请求, 并且 IR1 优先级最高 (终端查询字各位的信息于教材 266 页有详细说明)

{崔文韬问}: 课后习题第 18 题

{张衷豪答}:



主片初始化:

MOV AL,00011001B

OUT 0C8H,AL ;初始化 ICW1

MOV AL,00110000B

OUT 0C9H,AL ;初始化 ICW2

MOV AL,00001000B

OUT 0C9H,AL ;初始化 ICW3

MOV AL,00010001B;初始化 ICW4

OUT 0C9H,AL

MOV AL,11100110;允许 IR0,IR3,IR4 触发中断

OUT 0C9H,AL

从片初始化:

MOV AL,00011001B

```

OUT 0CAH,AL
MOV AL,40H
OUT 0CBH,AL
MOV AL,00000011B
OUT 0CBH,AL
MOV AL,00000001B
OUT 0CBH,AL
MOV AL,11111001; 允许 IR1 和 IR2 中断
{崔文韬问}: 课后习题第 20 题
{张衷豪答}:
MOV AX,2000H
MOV DS,AX
MOV DX,3600H
MOV AX,2544H
INT 21H

```

---

### 附录：课本改错

{崔文韬答}: 课本 P143 页，例 4.38，在一串给定个数的数据中寻找最大值，存放到 MAX 存储单元中。课本所给答案有错误，正确方法如下：

```
assume cs:code,ss:stack,ds:data
```

```
data segment
```

```
buf dw 3200h,1234h,4832h,5600h
```

```
count equ ($-buf)/2
```

```
max dw ?
```

```
data ends
```

```
stack segment stack
```

```
stapn db 100 dup(?)
```

```
top equ length stapn
```

```
stack ends
```

```
code segment
```

```
start:
```

```
mov ax,data
```

```
mov ds,ax
mov cx,count-1
lea bx,buf
mov ax,[bx]
again: inc bx
inc bx
cmp ax,[bx]
jge next
mov ax,[bx]
next: loop again
mov max,ax
mov ax,4c00H
int 21h
code ends
end start
```

还有其它方法可以实现该功能，欢迎同学提供答案。