第一章 基础知识 (1/24)

汇编语言是直接在硬件之上工作的编程语言，首先要了解硬件系统的结构，才能有效的应用汇编语言对其编程。

在本章中，对硬件系统结构的问题进行一部分的探讨，以使后续的课程可在一个好的基础上进行。

大学全学科资料、速成课，请进入小程序【一刷而过】

1.1 机器语言

- 机器语言是机器指令的集合。
- 机器指令展开来讲就是一台机器可以正确执行的命令。

指令：01010000 (PUSH AX)

电平脉冲： 

1.2 汇编语言的产生

- 汇编语言的主体是汇编指令。
- 汇编指令和机器指令的差别在于指令的表示方法上。汇编指令是机器指令便于记忆的书写格式。
- 汇编指令是机器指令的助记符。
- 寄存器：简单的讲是CPU中可以有储数据的器件，一个CPU中有多个寄存器。

大学全学科资料、速成课，请进入小程序【一刷而过】

1.3 汇编语言的组成

汇编语言由以下3类组成：

1. 汇编指令（机器码的助记符）
2. 伪指令（由编译器执行）
3. 其他符号（由编译器识别）

汇编语言的核心是汇编指令，它决定了汇编语言的特性。

1.4 存储器

CPU是计算机的核心部件，它控制整个计算机的运作并进行运算，要想让一个CPU工作，就必须向它提供指令和数据。

指令和数据在存储器中存放，也就是平时所说的内存。

大学全学科资料、速成课，请进入小程序【一刷而过】

1.5 指令和数据

- 指令和数据是应用上的概念。
- 在内有或磁盘上，指令和数据没有任何区别，都是二进制。
- 二进制信息：

1000100111011000

⟶ 89D8H（数据）

1000100111011000

⟶ MOV AX BX（程序）

1.6 存储单元

- 存储器被划分为若干个存储单元，每个存储单元从0开始顺序编号。

  例如：

  一个存储器有128个存储单元，

  编号从0~127。

大学全学科资料、速成课，请进入小程序【一刷而过】

- 对于大容量的存储器一般还用以下单位来计量容量（以下用B来表示Byte）：

  1KB = 1024B　　1MB = 1024KB

  1GB = 1024MB　　1TB = 1024GB

- 磁盘的容量单位同内存的一样，实际上以上单位是微机中常用的计量单位。

大学全学科资料、速成课，请进入小程序【一刷而过】

1.7 CPU对存储器的读写

在计算机中专门有连接CPU和其他芯片的导线，通常称为总线。

物理上：一根根导线的集合；

逻辑上划分为：

- 地址总线
- 数据总线
- 控制总线

1.8 地址总线

- CPU是通过地址总线来指定存储单元的。
- 地址总线上能传送多少个不同的信息，CPU就可以对多少个存储单元进行寻址。
- 一个CPU有N根地址总线，则可以说这个CPU的地址总线的宽度N
- 这样的CPU最多可以寻找2的N次方个内存单元。

大学全学科资料、速成课，请进入小程序【一刷而过】

1.9 数据总线

- CPU与内存或其他器件之间的数据传送是通过数据总线来进行的。
- 数据总线的宽度决定了CPU和外界的数据传送速度。

大学全学科资料、速成课，请进入小程序【一刷而过】

1.10 控制总线

- CPU对外部器件的控制是通过控制总线来进行的。在这里控制总线是个总称，控制总线是一些不同控制线的集合。
- 有多少根控制总线，就意味着CPU提供了对外部器件的多少种控制。

所以，控制总线的宽度决定了CPU对外部器件的控制能力。

大学全学科资料、速成课，请进入小程序【一刷而过】

小结：

(1) 汇编指令是机器指令的助记符，同机器指令一一对应。

(2) 每一种CPU都有自己的汇编指令集。

(3) CPU可以直接使用的信息在存储器中存放。

(4) 在存储器中指令和数据没有区别，都是二进制信息。

(5) 存储单元从零开始顺序编号。

(6) 一个存储单元可以存储8个bit，即8位二进制数

(7) 1B=8b　1KB=1024B　1MB=1024KB　1GB=1024MB

(8) 每一个CPU芯片都有许多管脚,这些管脚和总线相连。也可以说这些管脚引出总线。一个CPU可以引出三种总线的宽度标志了这个CPU的不同方面的性能:

- 地址总线的宽度决定了CPU的寻址能力;
- 数据总线的宽度决定了CPU与其他器件进行数据传送时的一次数据传送量;
- 控制总线宽度决定了CPU对系统中其他器件的控制能力。

第二章 寄存器 (2/24)

- 一个典型的CPU由运算器、控制器、寄存器等器件组成，这些器件靠内部总线相连。

区别：

内部总线实现CPU内部各个器件之间的联系，

外部总线实现CPU和主板上其它器件的联系。

8086CPU有14个寄存器 它们名称为：

AX、BX、CX、DX、SI、DI、SP、BP、IP、CS

SS、DS、ES、PSW。

以上所有寄存器都是16位的可以放两个字节。

字在寄存器中的存储：

一个字可以存在一个16位寄存器中，这个字的高位字节和低位字节自然就存在这个寄存器的高8位寄存器和低8位寄存器中。

物理地址：

CPU访问内存单元时要给出内存单元的地址。所有的内存单元构成的存储空间是一个一维的线性空间。

我们将这个唯一的地址称为物理地址，

16位结构的CPU

1. 运算器一次最多可以处理16位的数据。
2. 寄存器的最大宽度为16位。
3. 寄存器和运算器之间的通路是16位的。

大学全学科资料、速成课，请进入小程序【一刷而过】

段的概念：

段的划分来自于CPU，由于8086CPU用"(段地址×16)+偏移地址=物理地址"的方式给出内存单元的物理地址，使得我们可以用分段的方式来管理内存。

大学全学科资料、速成课，请进入小程序【一刷而过】

段寄存器：

段寄存器就是提供段地址的。

8086CPU有4个段寄存器：

CS、DS、SS、ES

小结：

- CPU
- 存在存储器中的储存
- 物理地址
- 16位结构的CPU
- 段寄存器

第三章 寄存器(内存访问)(3/24)

在上一节中，我们主要从CPU如何执行指令的角度讲解了8086CPU的逻辑结构，形成物理地址的方法，相关的寄存器以及一些指令。

内存中字的存储

任何两个地址连续的内存单元，N号单元和N+1号单元，可以将它们看成两个内存单元，也可以看成一个地址为N的字单元中的高位和低位字节单元。

DS和[address]

- CPU要读取一个内存单元的时候，必须先给出这个内存单元的地址；
- 在8086PC中，内存地址由段地址和偏移地址组成，
- 8086CPU中有一个DS寄存器，通常用来存放要访问的数据的段地址。

大学全学科资料、速成课，请进入小程序【一刷而过】

字的传送

因为8086CPU是16位结构，有16根数据线，所以，可以一次性传送16位的数据，也就是一次性传送一个字。

大学全学科资料、速成课，请进入小程序【一刷而过】

mov、add、sub指令

已学mov指令的几种形式：

mov寄存器、数据

mov寄存器、寄存器

mov寄存器、内存单元等

大学全学科资料、速成课，请进入小程序【一刷而过】

小结

(1) 字在内存中存储时，要用两个地址连续的内存单元来存放字的低位字节存放在低地址单元中，高位字节存放在高地址单元中。

(2) 用mov指令要访问内存单元，可以在mov指令中只给出单元的偏移地址。

(3) [address]表示一个偏移地址为address的内存单元。

(4) mov, add, sub是具有两个操作对象的指令。

第四章 一个源程序从写出到执行的过程

(4/24)

现在我们将开始编写完整的汇编语言程序，用编译器将它们编译为可执行文件，在操作系统中运用。

大学全学科资料、速成课，请进入小程序【一刷而过】

一个汇编语言程序从写出到最终执行的简要过程：

一. 编写汇编源程序：

使用文本编辑器(如记事本、Nodepad++等)，用汇编语言编写汇编源程序。

大学全学科资料、速成课，请进入小程序【一刷而过】

二.对源程序进行编译连接:

使用汇编语言编译程序(MASM.EXE)对源程序文件中的源程序进行编译,产生目标文件;

再用连接程序(LINK.EXE)对目标文件进行连接,生成可在操作系统中直接运行的可执行文件。

可执行文件包括两个部分内容：

- 程序和数据。
- 相关的描述信息。

三 执行可执行文件中的程序

在操作系统中，执行可执行文件中的程序。

源程序中的“程序”

汇编源程序：

伪指令（编译器处理）

汇编指令（编译为机器码）

大学全学科资料、速成课，请进入小程序【一刷而过】

小结：

- 一个源程序从写出到执行的过程
  - 源程序
  - 编辑源程序

大学全学科资料、速成课，请进入小程序【一刷而过】

第五章 以简化的方式进行编译和连接 (5/24)

- 编译连和接的作用是什么呢？

可执行文件中的程序装入内存并运行的原理：

在DOS中，可执行文件中的程序P_1若要运行，必须有一个正在运行的程序P_2，将P_1从可执行文件中加载入内存，将CPU的控制权交给它，P_1才能得以运行；

当P_1运行完毕后，应该将CPU的控制权交还给使它得以运行的程序P_2。

小结：

- 编译连和接的作用
- 可执行文件中的程序装入内存并运行的原理

第六章 exe的执行 (6/24)

一、exe的执行过程

(1) 我们在提示符“G:\TRY”后面输入可执行文件的名字“1”，按Enter键。

(2) 1.exe中的程序运行

(3) 运行结束，返回，再次显示提示符"G:\TRY"。

· 执行第(1)步操作时，有一个正在运行的程序将1.exe中的程序加载入内存，这个

正在运行的程序是什么？

- 它将程序加载入内存后，如何使程序得以运行？

(1) 我们在Dos中直接执行1.exe时，是正在运行的command将1.exe中的程序加载入内存。

(2) command设置CPU的CS:IP指向程序的第一条指令(即程序的入口)，从而使程序得以运行。

大学全学科资料、速成课，请进入小程序【一刷而过】

小结：

- exe的执行过程
- 使程序运行的条件

大学全学科资料、速成课，请进入小程序【一刷而过】

第七章 程序执行过程的跟踪

(7/24)

为了观察程序的运行过程，我们可以使用 Debug。

大学全学科资料、速成课，请进入小程序【一刷而过】

Debug可以将程序加载入内存，设置CS:IP指向程序的入口，但Debug并不放弃对CPU的控制，这样，我们就可以使用Debug的相关命令来单步执行程序查看每条指令指令的执行结果。

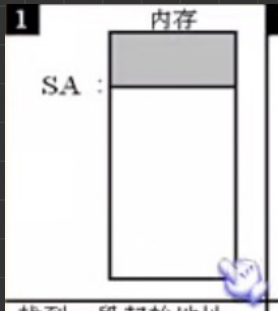
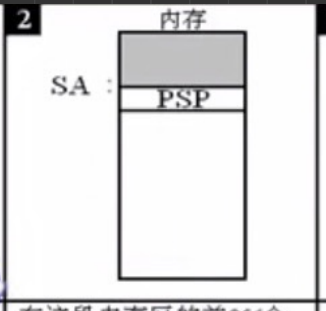
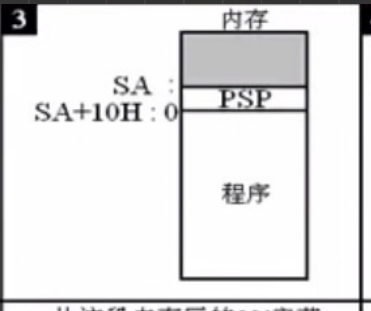
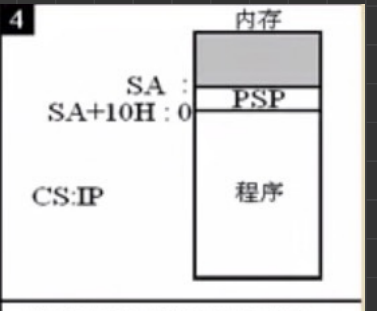
大学全学科资料、速成课，请进入小程序【一刷而过】

现在程序已从1.exe中装入内存，接下来我们查看一下它的内容，可是我们查看哪里的内容呢？

程序被装入内存的什么地方？

我们如何得知

在DOS系统中，EXE文件中的程序的加载过程如下。

1	2	3	4
			
找到一段起始地址为SA:0000（即起始地址的偏移地址为0）的容量足够的空闲内存区；	在这段内存区的前256个字节中，创建一个称为程序的前缀（PSP）的数据区，DOS要利用PSP来和被加载程序进行通信； （读者可能不理解PSP的作用，不过没有关系。我们并不研究DOS的原理，只要知道有这个东西就可以了。）	从这段内存区的256字节处开始（在PSP的后面），将程序装入，程序的地址被设为SA+10H:0； （空闲的内存区从SA:0开始，0~255字节为PSP，从256字节处开始存放程序，为更好地区分PSP和程序，DOS一般将它们划分到不同的段中，所以，有了这样的地址安排： 空闲内存区：SA:0 PSP区：SA:0 程序区：SA+10H:0 注意：PSP区和程序区虽然物理地址连续，却有不同的段地址。）	将该内存区的段地址存入DS中，初始化其他相关寄存器后，设置CS:IP指向程序的入口。

大学全学科资料、速成课，请进入小程序【一刷而过】

用U命令查看一下其他指令：

用T命令但不执行程序中的每一条指令，并观察每条指令的执行结果。

int 21执行后，显示"Program terminated normally"，返回到Debug中。

大学全学科资料、速成课，请进入小程序【一刷而过】

小结

- 观察程序的运行过程，可以用Debug
- 查看指令。

第八章 [BX]和loop指令 (8/24)

[bx]是什么呢？

和[0]有些类似，[0]表示内有单元它的偏移地址是0。

大学全学科资料、速成课，请进入小程序【一刷而过】

我们要完整地描述一个内存单元，需要两种信息：

(1) 内存单元的地址；

(2) 内存单元的长度(类型)。

大学全学科资料、速成课，请进入小程序【一刷而过】

Loop 和 [bx] 的联合应用

在实际编程中，经常会遇到，用同一种方法处理地址连续的内存单元中的数据的问题。

这时，我们就不能用常量来给出内存单元的地址，而应用变量。

大学全学科资料、速成课，请进入小程序【一刷而过】

段前缀

指令"mov ax,[bx]"中，内存单元的偏移地址由bx给出，而段地址默认在ds中。

我们可以在访问内存单元的指令中显示地给出内存单元的段地址所在的段寄存器。

大学全学科资料、速成课，请进入小程序【一刷而过】

一段安全的空间：

(1) 我们需要直接向一段内存中写入内容；

(2) DOS方式下，一般情况，0:200~0:2FF空间中没有系统或其他程序的数据或代码

(3) 以后，我们需要向一段内存中写入内容时，就使用0:200~0:2FF这段空间。

小结

- 描述内存单元
- Loop和[bx]用联合应用
- 段前缀
- 一段安全的空间

第九章 包含多个段的程序(9/24)

前面我们写的程序中，只有一个代码段，
现在有一个问题是：
如果程序需要用其他空间来存放数据
我们使用哪里呢？

大学全学科资料、速成课，请进入小程序【一刷而过】

在代码段中使用数据

程序中，我们用bx存放加2递增的偏移地址，用循环来累加。

在循环开始前，设置(bx)=0，cs:bx指向第一个数据所在的字单元。

每次循环中(bx)=(bx)+2，cs:bx指向下一个数据所在的字单元。

在代码段中使用栈

程序的思路大致如下：

程序运行时，定义的数据有放在cs:0~15单元中，共8个字单元。依次将这8个字单元中的数据入栈，然后再依次出栈到这8个字单元中，从而实现数据逆序存放。

小结：

- 在代码段中使用数据
- 在代码段中使用栈

大学全学科资料、速成课，请进入小程序【一刷而过】

第十章 更灵活的定位内存地址的方法 (10/24)

前面，我们用[0]，[bx]的方法，在访问内存的指令中，定位内存单元的地址。

大学全学科资料、速成课，请进入小程序【一刷而过】

and 和or 指令

首先我们介绍两条指令 and 和or，因为我们下面的例程中要用到它们。

(1) and 指令：逻辑与指令，按位进行与运算。 如 mov al，01100011B

and al，00111011B

执行后：al=00100011B

(2) or指令：逻辑或指令，按位进行或运算

如 mov al, 01100011B

and al, 00111011B

执行后：al=01111011B

大学全学科资料、速成课，请进入小程序【一刷而过】

关于ASCII码

世界上有很多编码方案，有种方案叫做ASCII编码，是在计算机系统中通常被采用的。

简单地说，所谓编码方案，就是一套规则，它约定了用什么样的信息来表示现实对象。

大学全学科资料、速成课，请进入小程序【一刷而过】

以字符形式给出的数据

我们可以在汇编程序中，用"……"的方式指明数据是以字符的形式给出的，编译器将把它们转化为相对应的ASCII码。

大学全学科资料、速成课，请进入小程序【一刷而过】

大小写转换的问题

我们可以将所有的字母的大写字符和小写字符所对应的ASCII码列出来。

大写	二进制	小写	二进制
A	01000001	a	01100001
B	01000010	b	01100010
C	01000011	c	01100011
D	01000100	d	01100100

小结

- and 和 or 指令
- 关于 ASCII 码
- 以字符形式给出的数据
- 大小写转换的问题

大学全学科资料、速成课，请进入小程序【一刷而过】

第十一章 数据处理的两个基本问题

(11/24)

我们知道，计算机是进行数据处理、运算的机器，那么有两个基本的问题包含其中：

(1) 处理的数据在什么地方？

(2) 要处理的数据有多长？

bx、si、di、bp

前三个寄存器我们已经用过3，现在我们进行一下总结。

(1) 在8086CPU中，只有这4个寄存器(bx、bp、si、di)可以用在“[...]”中进行内存单元的寻址。

我们来区分正确和错误的用法：

正确：

```
mov ax [bx]
mov ax,[bx+si]
mov ax [bx+di]
mov ax [bp]
mov ax [bp+si]
mov ax [bp+di]
```

错误

```
mov ax [cx]
mov ax [ax]
mov ax [dx]
mov ax [ds]
```

大学全学科资料、速成课，请进入小程序【一刷而过】

机器指令处理的数据所在位置：

绝大部分机器指令都是进行数据处理的指令，处理大致可分为三类：

读取，写入，运算

指令在执行前，所要处理的数据可以在CPU内部，内有端口。

数据位置的表达

汇编语言中用三个概念来表达数据的位置。

1. 立即数 (idata)
2. 寄存器
3. 段地址(SA)和偏移地址(EA)

小结

- bx, si, di, bp
- 机器指令处理的数据所在位置
- 数据位置的表达

第十二章 转移指令的原理

(12/24)

8086CPU的转移指令分为以下几类：

- 无条件转移指令
- 条件转移指令
- 循环指令
- 过程
- 中断

操作符 offset

操作符 offset 在汇编语言中是由编译器处理的符号，它的功能是取得标号的偏移地址。

大学全学科资料、速成课，请进入小程序【一刷而过】

jmp指令

jmp为无条件转移，可以只修改IP，也可以同时修改CS和IP；

大学全学科资料、速成课，请进入小程序【一刷而过】

jcxz指令

jcxz指令为有条件转移指令，所有的有条件转移指令都是短转移，在对应的机器码中包含转移的位移，而不是目的地址。对IP的修改范围都为-128~127。

小结

- 8086CPU的转移指令
- 操作符offset
- jmp指令
- jcxz指令

大学全学科资料、速成课，请进入小程序【一刷而过】

第十三章 call和ret指令 (13/24)

想想程序之间的加载返回过程。

call和ret指令都是转移指令，它们都修改IP，或同时修改CS和IP。

ret 和 retf

(一) ret 指令用栈中的数据，修改 IP 的内容，从而实现近转移！

CPU 执行 ret 指令时，进行下面两步操作：(1) (IP)=((SS)*16+(SP))

(2) (SP)=(SP)+2

(二) ret指令用栈中的数据，修改CS和IP的内容，从而实现近转移；

CPU执行ret指令时，进行下面两步：

(1) (IP)=((ss)*16+(sp)) (4)(sp)=(sp)+2

(2) (sp)=(sp)+2

(3) (CS)=((ss)*16+(sp))

Call指令

Call指令经常跟Ret指令配合使用，因此CPU执行Call指令，进行两步操作：

(1) 将当前的IP或CS和IP压入栈中；

(2) 转移(jmp)。

大学全学科资料、速成课，请进入小程序【一刷而过】

mul 指令

mul 是乘法指令，使用 mul 做乘法时：

(1) 相乘的两位数：要么都是8位，要么都是16位。

(2) 结果：

8位：AX中　16位：DX 和 AX中。

大学全学科资料、速成课，请进入小程序【一刷而过】

小结：

ret 和 retf

call 指令

null 指令

大学全学科资料、速成课，请进入小程序【一刷而过】

第十四章 标志寄存器 (14/24)

8086CPU的标志寄存器有16位，其中有储存的信息通常被称为程序状态字(PSW)。

ZF 标志：

例如：

```
mov ax,1
sub ax,1
```

指令执行后，结果为0，则ZF=1.

大学全学科资料、速成课，请进入小程序【一刷而过】

对于ZF的值，我们可以这样来看，ZF标记相关指令的计算结果是否为0，如果为0，则在ZF要记录下"是0"这样的肯定信息。

大学全学科资料、速成课，请进入小程序【一刷而过】

SF标志。

flag的第7位是SF，符号标志位。

它记录指令执行后，

结果为负，SF=1；

结果为正，SF=0。

大学全学科资料、速成课，请进入小程序【一刷而过】

CF标志

flag的第0位是CF，进位标志位。

一般情况下，在进行无符号数运算的时候，它记录了运算结果的最高有效位向更高位的进位值，或从更高位的借位值。

OF 标志

CF是对无符号数运算有意义的标志位；
而OF是对有符号数运算有意义的标志位。
对于无符号数运算，CPU用CF位记录是否产生了进位；
对于有符号数运算，CPU用OF位来记录是否产生了溢出。

大学全学科资料、速成课，请进入小程序【一刷而过】

小结：

- ZF 标志、
- SF 标志、
- CF 标志、
- OF 标志、

大学全学科资料、速成课，请进入小程序【一刷而过】

第十五章 内中断 (15/24)

中断是CPU处理外部突发事件的一个重要技术。

它能使CPU在运行过程中对外部事件发出的中断请求及时地进行处理，处理完成后又立即返回断点，继续工作。

大学全学科资料、速成课，请进入小程序【一刷而过】

中断处理程序

CPU的设计者必须在中断信息和其处理程序的入口地址之间建立某种联系，使得CPU根据中断信息可以找到要执行的处理程序。

大学全学科资料、速成课，请进入小程序【一刷而过】

中断向量表

CPU用8位的中断类型码通过中断向量表找到相应的中断处理程序的入口地址。

那么什么是中断向量表呢？

中断向量表就是中断向量的列表

大学全学科资料、速成课，请进入小程序【一刷而过】

中断过程

在中断向量表中找到中断处理程序的入口，找到这个入口地址的最终目的是用它设置CS和IP，使CPU执行中断处理程序。

用中断类型码找到中断向量，并用它设置CS和IP，这个工作是由CPU的硬件自动完成的。

小结：

- 中断处理程序
- 中断向量表
- 中断过程

第十六章 int 指令 (16/24)

上一节，我们讲解了中断过程和两种内中断的处理。

这一节，我们讲解另一种重要的内中断由int指令引发的中断。

int 指令的执行过程：

CPU执行 intn指令，相当于引发一个n号中断的中断过程，执行过程如下：

(1) 取中断类型码 n；

(2) 标志寄存器入栈，IF=0，TF=0；

(3) CS、IP入栈；

(4) (IP)=(n*4)，(CS)=(n*4+2)

对int、iret和栈的深入理解。

int 7ch引发中断过程后，进入7ch中断例程，在中断过程中，当前的标志寄有器，CS和IP都要压栈。

大学全学科资料、速成课，请进入小程序【一刷而过】

BIOS中断例程应用

bh中页号的含义：内存地址空间中，B8000h～BFFFFh共32k的空间，为80*25彩色字符模式的显示缓冲区，一屏的内容在显示缓冲区中共占4000个字节。

小结：

- int 指令的执行过程
- 对 int、iret 和栈的深入理解。
- BIOS 中断例程应用

大学全学科资料、速成课，请进入小程序【一刷而过】

第十七章 端口 (17/24)

CPU可以直接读写3个地方的数据：

(1) CPU内部的寄存器；

(2) 内存单元；

(3) 端口。

端口的读写：

- 对端口的读写不能用mov、push、pop等内有读写指令。
- 端口的读写指令只有两条：in和out分别用于从端口读取数据和往端口写入数据。

大学全学科资料、速成课，请进入小程序【一刷而过】

访问内存：

mov, ax, dx:[3];

假设执行前(ds)=0

执行时，与总线相关的操作：

·CPU通过地址线将地址信息3发出；

·CPU通过控制线发出内存读命令，选中存储器芯片，并通知它，将从中读取数据

shl和shr指令：

shl逻辑左移指令，功能为：

(1)将一个寄存器或内存单元中的数据向左移位；

(2)将最后移出的一位写入CF中。

小结：

- 端口的读写
- 访问内存
- shl 和 shr 指令

第十八章 外中断 (18/24)

CPU在计算机系统中，除了能够执行指令，进行运算以外还应该能够对外部设备进行控制，接收它们的输入，并向它们输出。

大学全学科资料、速成课，请进入小程序【一刷而过】

接口芯片和端口

外设的输入不直接送入内存和CPU，而是送入相关的接口芯片的端口中；

CPU向外设的输出也不是直接送入外设，而是先送入端口中，再由相关的芯片送到外设。

外中断信息、

在PC系统中，外中断源一共有两类：

1. 可屏蔽中断　2 不可屏蔽中断

可屏蔽中断是CPU可以不响应的外中断。

CPU是否响应可屏蔽中断，要看标志寄存器的IF位的设置。

小结：

- 外中断
- 接口芯片和端口
- 外中断信息、

第十九章 直接定址表(19/24)

描述了单元长度的标号

程序中，code、a、b，start、s都是标号。这些标号仅仅表示了内有单元的地址。

但是，我们还可以使用另一种标号这种标号不但表示内有单元的地址，还表示了内有单元的长度，即表示在此标号处的单元，是一个字节单元，还是字单元，还是双字单元。

在其他段中使用数据标号：

一般来说我们不会在代码段中定义数据而是将数据定义到其他段中。

在其他段中，我们也可以使用数据标号来描述存储数据的单元的地址和长度。

大学全学科资料、速成课，请进入小程序【一刷而过】

小结：

- 描述了单元长度的标号
- 在其他段中使用数据标号

第二十章 使用BIOS进行键盘输入和磁盘读写

(20/24)

大多数有用的程序都需要处理用户的输入，键盘输入是最基本的输入。

大学全学科资料、速成课，请进入小程序【一刷而过】

int 9中断例程对键盘输入的处理

CPU在9号中断发生后，执行int 9中断例程，从60h端口读出扫描码，并将其转化为相应的ASCII码或状态信息，存储在内存的指定空间中。

大学全学科资料、速成课，请进入小程序【一刷而过】

使用int 16h中断例程读取键盘缓冲区

BIOS提供了int 16h中断例程供程序员调用。

int 16h中断例程中包含的一个最重要的功能是从键盘缓冲区中读取一个键盘输入，该功能的编号为0。

字符串的输入：

用户通过键盘输入的通常不仅仅是单个字符而是字符串。

最基本的字符串输入程序，需要：

(1)在输入的同时需要显示这个字符串；

(2)一般在输入回车符后，字符串输入结束；

小结：

- int 9中断例程对键盘输入的处理
- 使用int 16h中断例程读取键盘缓冲区
- 字符串的输入

大学全学科资料、速成课，请进入小程序【一刷而过】

第二十一章 PC机键盘的处理过程 (21/24)

键盘输入的处理过程：

1. 键盘输入
2. 引发 9号中断
3. 执行 int 9 中断例程。

大学全学科资料、速成课，请进入小程序【一刷而过】

键盘输入：

键盘上的每一个键相当于一个开关，键盘中有一个芯片对键盘上的每一个键的开关状态进行扫描。

大学全学科资料、速成课，请进入小程序【一刷而过】

按下一个键时的操作

•开关接通，该芯片就产生一个扫描码，扫描码说明了按下的键在键盘上的位置。

•扫描码被送入主板上的相关接口芯片的奇存器中，该寄存器的端口地址为60H。

松开按下的键时的操作

•产生一个扫描码，扫描码说明了松开的键在键盘上的位置。

•松开按键时产生的扫描码也被送入60H端口中。

扫描码——长度为一个字节的编码

•按下一个键时产生的扫描码——通码，通码的第7位为。

•松开一个键时产生的扫描码——断码，断码的第7位为 1

例：g键的通码为22H，断码为a2H

引发9号中断：

键盘的输入到达60H端口时，相关的芯片就会向CPU发出中断类型码为9的可屏蔽中断信息。

CPU检测到该中断信息后，如果IF=1，则响应中断，引发中断过程，转去执行int 9 中断例程。

执行 int 9 中断例程

BIOS 中提供的处理键盘输入的 int 9 中断例程的工作：

(1) 读出 60H 端口中的扫描码；

(2) 根据扫描码分情况对待。

(3) 对键盘系统进行相关的控制。

小结：

- 键盘输入
- 引发9号中断
- 执行int 9 中断例程

第二十二章 定制键盘输入处理

(22/24)

键盘输入的处理过程：

(1) 键盘产生扫描码

(2) 扫描码送入60h端口

(3) 引发9号中断

(4) CPU执行int 9中断例程。

实现：依次显示'a'~'z'（V0.2）

```
assume cs:code
code segment
start:  mov ax,0b800h
        mov es,ax
        mov ah,'a'
    s:  mov es:[160*12+40*2],ah
        inc ah
        cmp ah,'z'
        jna s
        mov ax,4c00h
        int 21h
code ends
end start
```

问题：无法看清屏幕上的显示

原因：同一位置显示字母，字母之间切换得太快，无法看清。

对策：在每显示一个字母后，延时一段时间

按下Esc键后改变显示的颜色：

（1）从60h端口读出键盘的输入；

（2）调用B10S 的int 9 中断例程，处理硬件细蒙

（3）判断是否为Esc的扫描码，如果是，改变显示的颜色后返回；如果不是则直接返回。

大学全学科资料、速成课，请进入小程序【一刷而过】

小结

- 实现：依次显示'a'~'z'(V0.2)
- 按下Esc键后改变显示的颜色

第二十三章 用中断响应外设(23/24)

如何操作外部设备？

以典型输入设计—键盘操作为例

硬件中断 int 9h → Bios中断 int 16h

Dos中断 int 21h ←

大学全学科资料、速成课，请进入小程序【一刷而过】

对键盘输入的处理的int 9h中断

int 9h 将键盘输入存入缓冲或改变状态字

键盘输入将发9号中断，BIOS提供3in9中断例程。

int 9 中断例程从60h端口读出扫描码，并将其转化为相应的ASCII码或状态信息。

对键盘输入的处理的int 16h中断

BIOS提供了int 16h中断例程供程序员调用，以完成键盘的各种操作。

例：当（AH)-0时，读取键密缓冲区
功能：从键盘缓冲区中读取一个键盘输入，并旦将其从缓冲区中删除。

```
mov ah,0
int 16h
```

结果：(ah)=扫描码，(al)=ASCI码。

调用int 16h从键盘缓冲区读取输入

综前所述：int 16h 中断例程0号功能的实现过程

(l）检测键盘缓冲区中是否有数据：

(2）没有则继续做第1步

(3）读取缓冲区第一个字单元中的键盘输入；

(4）将读取的扫描码送入ah，ASCII码送入al：

(5）将已读取的键盘输入从缓冲区中删除。

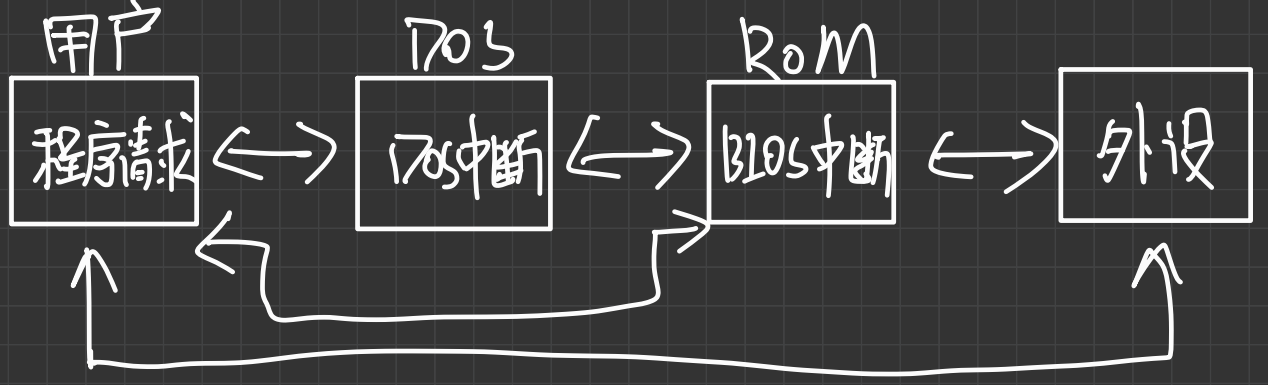
小结

- 对键盘输入的处理的int 9h 中断
- 对键盘输入的处理的int 16h中断
- 调用int 16h从键盘缓冲区读取输入

大学全学科资料、速成课，请进入小程序【一刷而过】

第二十四章 读写磁盘 (24/24)

如何操作磁盘？



BIOS 提供的磁盘直接服务 - int 13h

功能号	功能	功能号	功能	功能号	功能
00H	磁盘系统复位	0AH	读长扇区	14H	控制器内部诊断
01H	读取磁盘系统状态	0BH	写长扇区	15H	读取磁盘类型
02H	读扇区	0CH	查寻	16H	读取磁盘变化状态
03H	写扇区	0DH	硬盘系统复位	17H	设置磁盘类型
04H	检验扇区	0EH	读扇区缓冲区	18H	设置格式化媒体类型
05H	格式化磁道	0FH	写扇区缓冲区	19H	磁头保护
06H	格式化坏磁道	10H	读取驱动器状态	1AH	格式化ESDI驱动器
07H	格式化驱动器	11H	校准驱动器		
08H	读取驱动器参数	12H	控制器RAM诊断		
09H	初始化硬盘参数	13H	控制器驱动诊断		

大学全学科资料、速成课，请进入小程序【一刷而过】

DOS中断对磁盘文件的支持—int21H

目录控制功能(Directory-Control Function)
39H —创建目录　3AH —删除目录　3BH —设置当前目录　47H —读取当前目录
磁盘管理功能(Disk-Management Function)
0DH —磁盘复位　2EH —设置校验标志　0EH —选择磁盘　36H —读取驱动器分配信息 19H —读取当前驱动器　54H —读取校验标志　1BH, 1CH —读取驱动器数据
文件操作功能(File Operation Function)
……
文件操作功能(FCB)(File Operation Function)
……
记录操作功能(Record Function)
……
记录操作功能(FCB)(Record Function)
……

大学全学科资料、速成课，请进入小程序【一刷而过】

小结：

- 如何操作磁盘
- BIOS提供的磁盘直接服务 - int 13h
- DOS中断对磁盘文件的支持 - int 21h