第1章 基础知识

地址总线x根 寻址空间2xB

数据总线8x根，一次传输数据xB

内存地址空间分段，不同地址编号对应不同地址空间

第2章 寄存器

8086CPU有14个寄存器 它们的名称为：

AX、BX、CX、DX、SI、DI、SP、BP、

IP、CS、SS、DS、ES、PSW

通用寄存器：AX、BX、CX、DX，

add al,86H add al,90H 此时ah和al是两个独立的寄存器，

数据传送或运算时，指令的两个操作对象的位数是一致的。  
地址加法器：物理地址=段地址×16+偏移地址 段的起始地址是16的倍数

段寄存器：CS、SS、DS、ES

CS和IP是8086CPU中最关键的寄存器，它们指示了CPU当前要读取指令的地址。

mov不能修改CS和IP的值

CS为代码段寄存器；IP为指令指针寄存器。

8086CPU CS=FFFFH,IP=0000H

jmp ax – mov IP,ax

jmp 段地址：偏移地址

第3章 寄存器

高地址内存单元中存放字型数据的高位字节，低地址内存单元中存放自行数据的低位字节。

DS：通常用来存放要访问数据的段地址

mov 寄存器名，内存单元地址

不能将数据直接送入段寄存器，要用寄存器中转

mov bx,1000H mov ds,bx

任意时刻，SS:SP指向栈顶元素 栈空时指向栈最大地址的下一地址

push ax pop ax 将ax中的数据放入栈，从栈顶取出数据送入ax

pop执行后，pop前的栈顶元素依然存在，但是不在栈中

push/pop 寄存器 push/pop 段寄存器 push/pop 内存单元

第4章 第一个程序

伪指令 由编译器执行的指令 汇编指令-被编译成机器指令后被CPU执行

段名segment – 段名ends

end 整个程序的结束

assume 假设假设某一段寄存器和某一segemen…ends定义的段相关联

程序返回 mov ax,4c00H int 21H

编写->编译->连接->执行

第5章 [bx]和loop指令

（） 描述性符号，内容 寄存器名、段寄存器名、内存单元的物理地址

idata 表示常量

[bx] ((ds)×16+(bx))

loop cx中存放循环次数 (cx)=cx-1 判断cx中的值，不为0则专职标号处执行程序，如果为0则向下执行

在访问内存单元的指令中显式地给出内存单元的段地址所在的段寄存器。

在一般的PC机中，DOS方式下，DOS和其他合法的程序一般都不会使用0:200~0:2FF（ 0:200h~0:2FFh）的256 个字节的空间。

第6章包含多个段的程序

用dw定义的数据处于代码段的最开始

end 除了通知编译器程序结束外，还可以通知编译器程序的入口在什么地方。

设置ss指向stack，设置ss:sp指向stack:20，CPU 执行这些指令后，将把stack段当做栈空间来用。CPU若要访问data段中的数据，则可用 ds 指向 data 段，用其他的寄存器（如：bx）来存放 data段中数据的偏移地址

第7章 更灵活的定位内存地址的方法

将ASCII码的第5位置为0，变为大写字母，置为1变为小写字母

a-61H A-41H

mov ax,[200+bx]

mov ax,200[bx]

mov ax,[bx].200

bx与si、di用法相同

指令mov ax,[bx+si]的数学化的描述为：(ax)=( (ds)\*16+(bx)+(si) )

该指令也可以写成如下格式（常用）：mov ax,[bx][si]

[bx+si+idata]表示一个内存单元，它的偏移地址为(bx)+(si)+idata。

（即bx中的数值加上si中的数值再加上idata）

mov ax,[bx+200+si]

mov ax,[200+bx+si]

mov ax,200[bx][si]

mov ax,[bx].200[si]

mov ax,[bx][si].200

将外层循环的cx值保存在dx中，用dx中存放的外层循环的计数值恢复cx

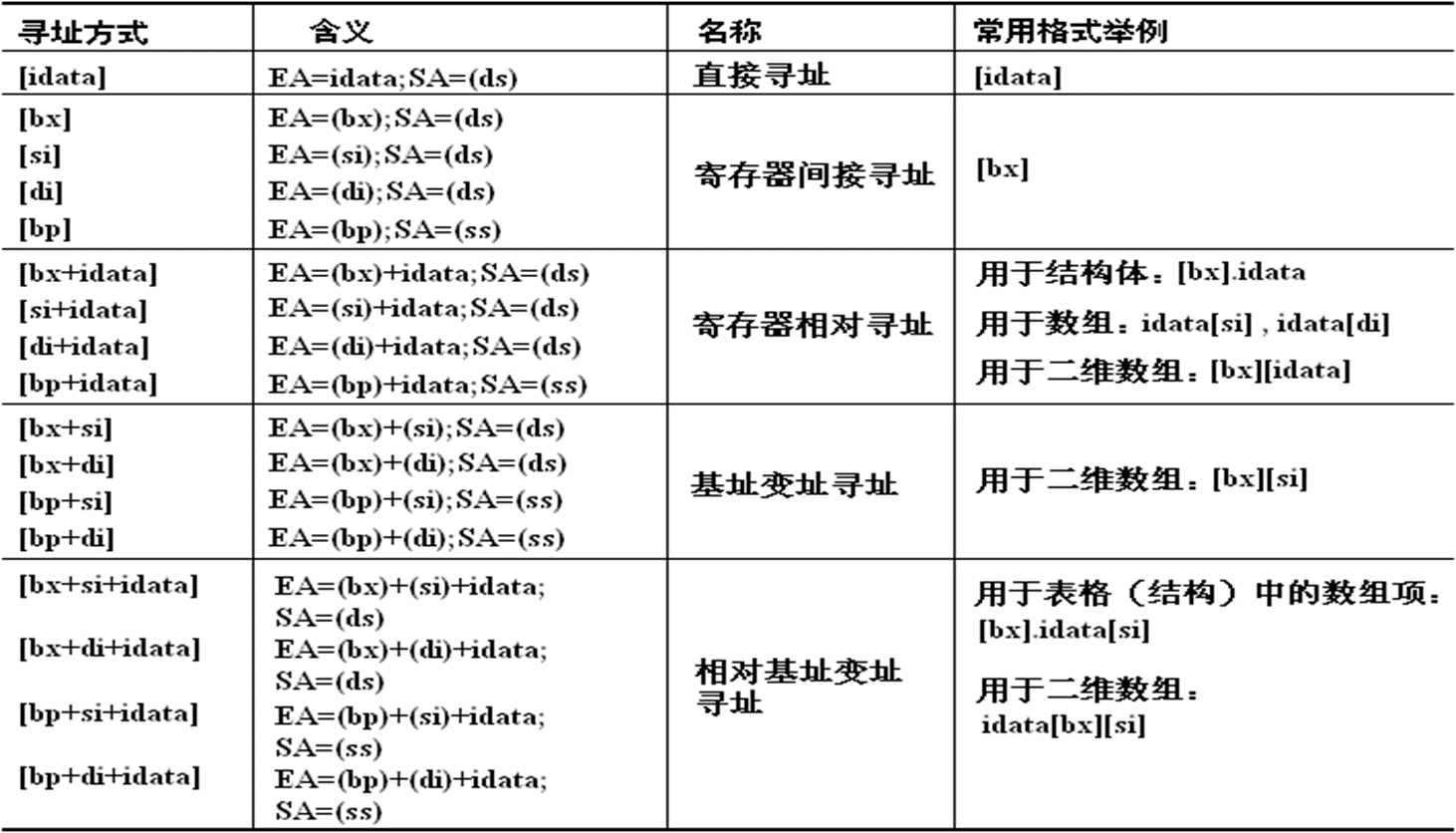
将需要暂存的数据放到内存单元中，需要使用的时候，再从内存单元中恢复。

push、pop等，可对其进行特殊的操作。

第8章 数据处理的两个基本问题

描述性符号 reg表示一个寄存器，sreg表示一个段寄存器

bp段地址默认在ss中



word ptr或byte ptr

push指令只进行字操作

div指令 16位 被除数-DX|AX 除数-reg或内存单元 商-AX 余数-DX

8位 被除数-AX 商-AL 余数-AH

伪指令dd

dup db 3 dup (0) db 3 dup (0,1,2)

db 重复的次数 dup （重复的字节型数据）

dw 重复的次数 dup （重复的字型数据）

dd 重复的次数 dup （重复的双字数据）

第9章

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。

在“jmp short 标号”指令所对应的机器码中，并不包含转移的目的地址，而包含的是转移的位移。(IP)=(IP)+8位位移

jmp near ptr 标号 它实现的时段内近转移。

指令“jmp near ptr 标号”的功能为： (IP)=(IP)+16位位移。

指令 “jmp far ptr 标号” 实现的是段间转移，又称为远转移。

指令格式：jmp 16位寄存器 功能：IP =（16位寄存器）

jmp word ptr 内存单元地址（段内转移）

jmp dword ptr 内存单元地址（段间转移）

功能：从内存单元地址处开始存放着两个字，高地址处的字是转移的目的段地址，低地址处是转移的目的偏移地址。

(CS)=(内存单元地址+2) (IP)=(内存单元地址)

“jcxz 标号”的功能相当于：

if((cx)==0)jmp short 标号

第10章 CALL和RET指令

CPU执行ret指令时，进行下面两步操作：

（1）(IP)=((ss)\*16+(sp))

（2）(sp)=(sp)+2

CPU执行retf指令时，进行下面两步操作：

（1）(IP)=((ss)\*16+(sp))

（2）(sp)=(sp)+2

（3）(CS)=((ss)\*16+(sp))

（4）(sp)=(sp)+2

CPU执行ret指令时，相当于进行：pop IP

CPU执行retf指令时，相当于进行：pop IP pop CS

call 标号（将当前的 IP 压栈后，转到标号处执行指令）

16位位移=“标号”处的地址－call指令后的第一个字节的地址；

CPU执行“call far ptr 标号”这种格式的call指令时的操作：

(1) (sp) = (sp) – 2 ((ss) ×16+(sp)) = (CS)

(sp) = (sp) – 2 ((ss) ×16+(sp)) = (IP)

(2) (CS) = 标号所在的段地址 (IP) = 标号所在的偏移地址

call 16位寄存器 功能： (sp) = (sp) – 2 ((ss)\*16+(sp)) = (IP) (IP) = (16位寄存器)

push IP jmp 16位寄存器

子程序框架

mul指令

子程序中使用的寄存器入栈

子程序内容

子程序使用的寄存器出栈

返回（ret、retf）