目 录

1 两个基本问题

两个基本问题是：

1. 处理的数据在什么地方？
2. 要处理的数据有多长？

1.1 bx、si、di 和 bp

bx、si、di 和 bp 这 4 个寄存器可以单个出现，或者只能以四种组合出现：bx 和 si、bx 和 di、bp 和 si、bp 和 di。

需要注意的是，如果使用 [bp]，那么默认段地址存放在 ss 中，而不是存放在 ds 中。

1.2 汇编语言中数据位置的表达

关于数据处理的指令可以分为三类：读取、写入和运算。对于机器指令，关心的是处理的数据在什么地方。指令在执行前，所要处理的数据可以在三个地方：CPU 内部、内存和端口。

汇编语言用如下三个概念来表达数据的位置：

1. 立即数 idata，直接包含在机器指令中的数据，也就是在 CPU 的指令缓冲器中，如下所示：

```
1    mov ax, 1
```

2. 寄存器，指令要处理的数据在寄存器中，在汇编指令中给出相应的寄存器名，如下所示：

```
1    mov ax, bx
```

3. 段地址和偏移地址，指令要处理的数据在内存中，在汇编指令中以 [X] 的格式给出，如下所示：

```
1    mov ax, [di]
2    mov ax, [bp]
```

1.2.1 寻址方式

寻址方式总结如下图所示：

寻址方式	含　义	名　称	常用格式举例
[idata]	EA=idata;SA=(ds)	直接寻址	[idata]
[bx] [si] [di] [bp]	EA=(bx);SA=(ds) EA=(si);SA=(ds) EA=(di);SA=(ds) EA=(bp);SA=(ss)	寄存器间接寻址	[bx]
[bx+idata] [si+idata] [di+idata] [bp+idata]	EA=(bx)+idata;SA=(ds) EA=(si)+idata;SA=(ds) EA=(di)+idata;SA=(ds) EA=(bp)+idata;SA=(ss)	寄存器相对寻址	用于结构体： [bx].idata 用于数组： idata[si],idata[di] 用于二维数组： [bx][idata]
[bx+si] [bx+di] [bp+si] [bp+di]	EA=(bx)+(si);SA=(ds) EA=(bx)+(di);SA=(ds) EA=(bp)+(si);SA=(ss) EA=(bp)+(di);SA=(ss)	基址变址寻址	用于二维数组： [bx][si]
[bx+si+idata] [bx+di+idata] [bp+si+idata] [bp+di+idata]	EA=(bx)+(si)+idata; SA=(ds) EA=(bx)+(di)+idata; SA=(ds) EA=(bp)+(si)+idata; SA=(ss) EA=(bp)+(di)+idata; SA=(ss)	相对基址变址寻址	用于表格(结构)中的数组项： [bx].idata[si] 用于二维数组： idata[bx][si]

1.3　要处理的数据有多长

8086CPU 的指令可以处理两种长度的数据：byte 和 word。所以需要在机器指令中指令指令进行的是字操作还是字节操作，有如下几种方式：

1. 通过寄存器名指明要处理的数据的长度。如果用 ax、bx、cx、dx，就是字操作。如果用 al、bl、cl、dl 等，就是字节操作。如下所示：

```
1    mov ax, 1
2    mov bx, ds:[0]
3
4    mov al, 1
5    mov al, bl
6    mov ds:[0], al
```

2. 通过 X ptr 操作符指明内存单元的长度。如果用 word ptr，就是字操作。如果用 byte ptr，就是字节操作。如下所示：

```
1    mov word ptr ds:[0], 1
2    mov byte ptr ds:[0], 1
```

3. 部分指令默认了访问的是字单元还是字节单元，比如“push [address]”就是默认字操作。

2 div 指令

div 指令是除法指令，对它的介绍如下：

1. 除数：有 8 位和 16 位两种，在一个寄存器或内存单元中。
2. 被除数：如果除数为 8 位，那么被除数为 16 位，默认放在 AX 中。如果除数为 16 位，那么被除数为 32 位，默认在 DX 和 AX 中存放，DX 存放高 16 位，AX 存放低 16 位。
3. 结果：如果除数为 8 位，那么 AL 存储商，AH 存储余数。如果除数为 16 位，那么 AX 存储商，DX 存储余数。
4. div 指令的格式如下：

```
1  div reg
2  div 内存单元
```

可以看出，div 指令的操作对象不可以是段寄存器 sreg，只能是寄存器 reg。

3 dup

dup 是一个操作符，和 db、dw、dd 这些指令配合使用。dup 的使用格式如下：

- 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 (重复的双字型数据)

dup 的使用如下例所示：

```
1  db 3 dup (0)
2  db 3 dup (0, 1, 2)
3  db 3 dup ('abc', 'ABC')
```