

第 3 章

8086的指令系统

教学重点

第3章是本课程的一个关键内容，是程序设计的基础

- ✓ 基础是熟悉寄存器组
- ✓ 难点是各种寻址方式
- ✓ 重点是掌握8086常用指令的功能及应用



- 计算机的指令系统就是指该计算机能够执行的全部指令的集合
- 每种计算机都有它支持的指令集合
- 16位8086指令系统是Intel 80x86系列微处理器指令系统的基础

● 一定要采用调试程序**DEBUG**进行实践

- ▶ DEBUG是常用的汇编语言级调试工具，为汇编语言程序员提供了分析指令、跟踪程序的有效手段
- ▶ 常用命令：
 - A 汇编
 - T 单步执行
 - D 数据显示
 - U 反汇编
 - G 断点执行
 - R 寄存器

感性认识因深刻而显重要

➤ 对程序员来说，8086内部结构的最重要的是其寄存器组

- ❖ 8个通用寄存器
- ❖ 1个指令指针寄存器
- ❖ 1个标志寄存器
- ❖ 4个段寄存器

- 8086的16位通用寄存器是：

AX	BX	CX	DX
SI	DI	BP	SP

- 其中前4个数据寄存器都还可以分成高8位和低8位两个独立的寄存器
- 8086的8位通用寄存器是：

AH	BH	CH	DH
AL	BL	CL	DL

- 对其中某8位的操作，并不影响另外对应8位的数据

- 数据寄存器用来存放计算的结果和操作数，也可以存放地址
- 每个寄存器又有它们各自的专用目的
 - **AX**——累加器，使用频度最高，用于算术、逻辑运算以及与外设传送信息等；
 - **BX**——基址寄存器，常用做存放存储器地址；
 - **CX**——计数器，作为循环和串操作等指令中的隐含计数器；
 - **DX**——数据寄存器，常用来存放双字长数据的高16位，或存放外设端口地址。

- ▶ 变址寄存器常用于存储器寻址时提供地址
 - SI是源变址寄存器
 - DI是目的变址寄存器
- ▶ 串操作类指令中，SI和DI具有特别的功能

- 指针寄存器用于寻址内存堆栈内的数据
- SP为堆栈指针寄存器，指示栈顶的偏移地址
- SP不能再用于其他目的，具有专用目的
- BP为基址指针寄存器，表示数据在堆栈段中的基地址
- SP和BP寄存器与SS段寄存器联合使用以确定堆栈段中的存储单元地址

- 指令指针寄存器**IP**，指示代码段中指令的偏移地址
- 它与代码段寄存器**CS**联用，确定下一条指令的物理地址
- 计算机通过**CS : IP**寄存器来控制指令序列的执行流程
- **IP**寄存器是一个专用寄存器

- 标志（**Flag**）用于反映指令执行结果或控制指令执行形式
- 8086处理器的各种标志形成了一个16位的标志寄存器**FLAGS**（程序状态字**PSW**寄存器）

● 程序设计需要利用标志的状态

15 12 11 10 9 8 7 6 5 4 3 2 1 0

OF

DF

IF

TF

SF

ZF

AF

PF

CF

- ▶ 状态标志 —— 用来记录程序运行结果的状态信息，许多指令的执行都将相应地设置它

CF ZF SF PF OF AF

- ▶ 控制标志 —— 可由程序根据需要由指令设置，用于控制处理器执行指令的方式

DF IF TF

进位标志CF (Carry Flag)

- 当运算结果的最高有效位有进位（加法）或借位（减法）时，进位标志置1，即**CF = 1**；否则**CF = 0**。

3AH + 7CH = B6H，没有进位：CF = 0

AAH + 7CH = (1) 26H，有进位：CF = 1

- 若运算结果为0，则 $ZF = 1$ ；
否则 $ZF = 0$

● 注意：ZF为1表示的结果是0

$3AH + 7CH = B6H$ ，结果不是零： $ZF = 0$

$84H + 7CH = (1) 00H$ ，结果是零： $ZF = 1$

➤ 运算结果最高位为1，则**SF = 1**；
否则**SF = 0**

● 有符号数据用最高有效位表示数据的符号
所以，最高有效位就是符号标志的状态

$3AH + 7CH = B6H$ ，最高位 $D_7 = 1$ ：SF = 1

$84H + 7CH = (1) 00H$ ，最高位 $D_7 = 0$ ：SF = 0

- ▶ 当运算结果最低字节中“1”的个数为零或偶数时， $PF = 1$ ；否则 $PF = 0$

● PF标志仅反映最低8位中“1”的个数是偶或奇，即使是进行16位字操作

$3AH + 7CH = B6H = 10110110B$

结果中有5个1，是奇数： $PF = 0$

▶ 若算术运算的结果有溢出，
则OF=1； 否则 OF=0

$3AH + 7CH = B6H$ ，产生溢出：OF = 1

$AAH + 7CH = (1) 26H$ ，没有溢出：OF = 0

问题

什么是溢出？

溢出和进位有什么区别？

处理器怎么处理，程序员如何运用？

如何判断是否溢出？



➤ 运算时D₃位（低半字节）有进位或借位时，AF = 1；否则AF = 0。

这个标志主要由处理器内部使用，用于十进制算术运算调整指令中，用户一般不必关心

3AH + 7CH = B6H, D₃有进位: AF = 1

➤ 用于串操作指令中，控制地址的变化方向：

- 设置DF=0，存储器地址自动增加；
- 设置DF=1，存储器地址自动减少。

➤ CLD指令复位方向标志：DF=0

➤ STD指令置位方向标志：DF=1

➤ 用于控制外部可屏蔽中断是否可以被处理器响应：

- 设置 $IF=1$ ，则允许中断；
- 设置 $IF=0$ ，则禁止中断。

➤ CLI指令复位中断标志： $IF=0$

➤ STI指令置位中断标志： $IF=1$

- 用于控制处理器进入单步操作方式：
 - 设置TF=0，处理器正常工作；
 - 设置TF=1，处理器单步执行指令。
- 单步执行指令——处理器在每条指令执行结束时，便产生一个编号为1的内部中断
- 这种内部中断称为单步中断
- 所以TF也称为单步标志
 - 利用单步中断可对程序进行逐条指令的调试
 - 这种逐条指令调试程序的方法就是单步调试

- 寄存器是微处理器内部暂存数据的存储单元，以名称表示
- 存储器则是微处理器外部存放程序及其数据的空间
- 程序及其数据可以长久存放在外存，在程序需要时才进入主存
- 主存需要利用地址区别

➤ 计算机中信息的单位

- 二进制位Bit: 存储一位二进制数: 0或1
- 字节Byte: 8个二进制位, $D_7 \sim D_0$
- 字Word: 16位, 2个字节, $D_{15} \sim D_0$
- 双字DWord: 32位, 4个字节, $D_{31} \sim D_0$

➤ 最低有效位LSB: 数据的最低位, D_0 位

➤ 最高有效位MSB: 数据的最高位, 对应字节、字、双字分别指 D_7 、 D_{15} 、 D_{31} 位

- ▶ 每个存储单元都有一个编号；被称为存储器地址
- ▶ 每个存储单元存放一个字节的内容

0002H单元存放有一个数据34H
表达为 $[0002H] = 34H$

➤ 多字节数据在存储器中占连续的多个存储单元：

● 80x86处理器采用“低对低、高对高”的存储形式，被称为“小端方式 Little Endian”。

● 相对应还存在“大端方式 Big Endian”。

$[0002H] = 1234H$

2号“双字”单元的内容为：

$[0002H] = 78561234H$

视具体情况来确定

- ▶ 同一个存储器地址可以是字节单元地址、字单元地址、双字单元地址等等
- ▶ 字单元安排在偶地址（ $xxx0B$ ）、双字单元安排在模4地址（ $xx00B$ ）等，被称为“地址对齐（Align）”
- ▶ 对于不对齐地址的数据，处理器访问时，需要额外的访问存储器时间
- ▶ 应该将数据的地址对齐，以取得较高的存取速度


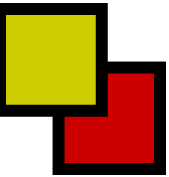
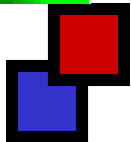
- 8086CPU有20条地址线
 - 最大可寻址空间为 $2^{20}=1\text{MB}$
 - 物理地址范围从00000H~FFFFFH
- 8086CPU将1MB空间分成许多逻辑段 (Segment)
 - 每个段最大限制为64KB
 - 段地址的低4位为0000B
- 这样，一个存储单元除具有一个唯一的物理地址外，还具有多个逻辑地址

➤ 对应每个物理存储单元都有一个唯一的**20位**编号，就是物理地址，从**00000H~FFFFFFH**。

➤ 分段后在用户编程时，采用逻辑地址，形式为

段基地址 : 段内偏移地址

分隔符

- 
- 
-
- 
- **段地址**说明逻辑段在主存中的起始位置
 - 8086规定段地址必须是模**16**地址：**xxxx0H**
 - 省略低**4**位**0000B**，段地址就可以用**16**位数据表示，就能用**16位段寄存器**表达段地址
-
- **偏移地址**说明主存单元距离段起始位置的偏移量
 - 每段不超过**64KB**，偏移地址也可用**16位**数据表示

➤ 将逻辑地址中的段地址左移4位，加上偏移地址就得到20位物理地址

➤ 每个物理地址可以由每个逻辑地址

逻辑地址	1460:100、	1380:F00
物理地址	14700H	14700H

段地址左移4位

14600H

13800H

加上偏移地址

+ 100H

+ F00H

得到物理地址

14700H

14700H

➤ 8086有4个16位段寄存器

- **CS**（代码段）指明代码段的起始地址
- **SS**（堆栈段）指明堆栈段的起始地址
- **DS**（数据段）指明数据段的起始地址
- **ES**（附加段）指明附加段的起始地址

➤ 每个段寄存器用来确定一个逻辑段的起始地址，每种逻辑段均有各自的用途

- ▶ 程序的**指令序列**必须安排在代码段
- ▶ 程序使用的**堆栈**一定在堆栈段
- ▶ 程序中的**数据**默认是安排在数据段，也经常安排在附加段，尤其是串操作的目的区必须是附加段
- ▶ 数据的存放比较灵活，实际上可以存放在任何一种逻辑段中

- 没有指明时，一般的数据访问在DS段；使用BP访问主存，则在SS段
- 默认的情况允许改变，需要使用段超越前缀指令；8086指令系统中有4个：

CS：；代码段超越，使用代码段的数据

SS：；堆栈段超越，使用堆栈段的数据

DS：；数据段超越，使用数据段的数据

ES：；附加段超越，使用附加段的数据

➤ 没有段超越的指令实例：

MOV AX, [2000H] ; AX ← DS:[2000H]
; 从默认的DS数据段取出数据

➤ 采用段超越前缀的指令实例：

MOV AX, ES:[2000H] ; AX ← ES:[2000H]
; 从指定的ES附加段取出数据

访问存储器的方式	默 认	可超越	偏移地址
取指令	CS	无	IP
堆栈操作	SS	无	SP
一般数据访问	DS	CS ES SS	有效地址EA
BP基址的寻址方式	SS	CS ES DS	有效地址EA
串操作的源操作数	DS	CS ES SS	SI
串操作的目的操作数	ES	无	DI

➤ 8086对逻辑段**要求**:

- 段地址低4位均为0
- 每段最大不超过64KB

各段独立

各段重叠

➤ 8086对逻辑段**并不要求**:

- 必须是64KB
- 各段之间完全分开（即可以重叠）

最**多**多少段？

最**少**多少段？

- 8086有8个8位通用寄存器、8个16位通用寄存器
- 8086有6个状态标志和3个控制标志
- 8086将1MB存储空间分段管理，有4个段寄存器，对应4种逻辑段
- 8086有4个段超越前缀指令，用于明确指定数据所在的逻辑段



熟悉上述内容后，就可以进入下节

- 从**8086**的机器代码格式入手，论述：
 - 立即数寻址方式
 - 寄存器寻址方式
 - 存储器寻址方式
- 进而熟悉**8086**汇编语言指令格式，尤其是其中操作数的表达方法；为展开**8086**指令系统做好准备

操作码

操作数

- 指令由操作码和操作数两部分组成
- **操作码**说明计算机要执行哪种操作，如传送、运算、移位、跳转等操作，它是指令中不可缺少的组成部分
- **操作数**是指令执行的参与者，即各种操作的对象
- 有些指令不需要操作数，通常的指令都有一个或两个操作数，也有个别指令有3个甚至4个操作数

➤ 每种指令的**操作码**:

- 用一个唯一的助记符表示（指令功能的英文缩写）
- 对应着机器指令的一个二进制编码

➤ 指令中的**操作数**:

- 可以是一个具体的数值
- 可以是存放数据的寄存器
- 或指明数据在主存位置的存储器地址

- 指令系统设计了多种操作数的来源
- 寻找操作数的过程就是操作数的寻址
- 操作数采取哪一种寻址方式，会影响机器运行的速度和效率

● 如何寻址一个操作数对程序设计很重要

3.1 8086的机器代码格式

1/2字节

0/1字节

0/1/2字节

0/1/2字节

操作码

mod reg r/m

位移量

立即数

操作数

➤ 表明采用的寻址方式

➤ 给出某些寻址方式需要的对基地址的偏移量

➤ 给出立即寻址方式需要的数值本身



mov ax,[BP+0] ; 机器代码是 **8B 46 00**

- 前一个字节**8B**是操作码（含w=1表示字操作）
- 中间一个字节**46**（**01 000 110**）是“mod reg r/m”字节
 - reg=000表示目的操作数为AX
 - mod=01和r/m=110表示源操作数为[BP+D8]
- 最后一个字节就是8位位移量（D8=）**00**

操作码

操作数

mov al,05 ; 机器代码是B0 05

- 前一个字节**B0**是操作码（含一个操作数**AL**），后一个字节**05**是立即数

mov ax,0102H ; 机器代码是B8 02 01

- 前一个字节**B8**是操作码（含一个操作数**AX**），后两个字节**02 01**是16位立即数（低字节**02**在低地址）

操作码 操作数1, 操作数2 ; 注释

- 操作数2, 称为源操作数 `src`, 它表示参与指令操作的一个对象
- 操作数1, 称为目的操作数 `dest`, 它不仅可以作为指令操作的一个对象, 还可以用来存放指令操作的结果
- 分号后的内容是对指令的解释

MOV dest, src ; dest ← src

- MOV指令的功能是将源操作数src传送至目的操作数dest，例如：

MOV AL, 05H ; AL ← 05H

MOV BX, AX ; BX ← AX

MOV AX, [SI] ; AX ← DS:[SI]

MOV AX, [BP+06H] ; AX ← SS:[BP+06H]

MOV AX, [BX+SI] ; AX ← DS:[BX+SI]

- 指令中的操作数直接存放在机器代码中，紧跟在操作码之后（操作数作为指令的一部分存放在操作码之后的主存单元中）
- 这种操作数被称为立即数imm
 - 它可以是8位数值i8（00H~FFH）
 - 也可以是16位数值i16（0000H~FFFFH）
- 立即数寻址方式常用来给寄存器赋值

MOV AL,05H ; **AL**←**05H**
MOV AX,0102H ; **AX**←**0102H**

指令功能

执行过程



➤ 操作数存放在**CPU**的内部寄存器**reg**中，
可以是：

■ 8位寄存器r8:

AH、AL、BH、BL、CH、CL、DH、DL

■ 16位寄存器r16:

AX、BX、CX、DX、SI、DI、BP、SP

■ 4个段寄存器seg:

CS、DS、SS、ES

MOV AX,1234H ; AX←1234H

MOV BX,AX ; BX←AX

指令功能

执行过程



- 指令中给出操作数的主存地址信息（偏移地址，称之为有效地址**EA**），而段地址在默认的或用段超越前缀指定的段寄存器中
- 8086设计了多种存储器寻址方式
 - 1、直接寻址方式
 - 2、寄存器间接寻址方式
 - 3、寄存器相对寻址方式
 - 4、基址变址寻址方式
 - 5、相对基址变址寻址方式

- 有效地址在指令中直接给出
- 默认的段地址在**DS**段寄存器，可使用段超越前缀改变

MOV AX,[2000H]

； **AX**←**DS:[2000H]**

； 指令代码：**A10020**

指令功能

MOV AX,ES:[2000H]

； **AX**←**ES:[2000H]**

； 指令代码：**26A10020**

执行过程



- ▶ 有效地址存放在基址寄存器**BX**或变址寄存器**SI**、**DI**中
- ▶ 默认的段地址在**DS**段寄存器，可使用段超越前缀改变

MOV AX,[SI] ; $AX \leftarrow DS:[SI]$

指令功能

执行过程



- ▶ 有效地址是寄存器内容与有符号8位或16位位移量之和，寄存器可以是**BX**、**BP**或**SI**、**DI**

有效地址 = **BX/BP/SI/DI** + 8/16位位移量

- ▶ 段地址对应**BX/SI/DI**寄存器默认是**DS**，对应**BP**寄存器默认是**SS**；可用段超越前缀改变

MOV AX,[DI+06H]

; AX ← DS:[DI+06H]

MOV AX,[BP+06H]

; AX ← SS:[BP+06H]

指令功能

执行过程



- ▶ 有效地址由基址寄存器（**BX**或**BP**）的内容加上变址寄存器（**SI**或**DI**）的内容构成：

$$\text{有效地址} = \text{BX/BP} + \text{SI/DI}$$

- ▶ 段地址对应**BX**基址寄存器默认是**DS**，对应**BP**基址寄存器默认是**SS**；可用段超越前缀改变

MOV AX,[BX+SI]

; AX ← DS:[BX+SI]

指令功能

MOV AX,[BP+DI]

; AX ← SS:[BP+DI]

执行过程

MOV AX,DS:[BP+DI]

; AX ← DS:[BP+DI]



- ▶ 有效地址是基址寄存器 (**BX/BP**)、变址寄存器 (**SI/DI**) 与一个8位或16位位移量之和:

有效地址 = **BX/BP** + **SI/DI** + 8/16位位移量

- ▶ 段地址对应**BX**基址寄存器默认是 **DS**，对应**BP**基址寄存器默认是 **SS**；可用段超越前缀改变

MOV AX,[BX+SI+06H]

; AX ← DS:[BX+SI+06H]

指令功能

执行过程



- 位移量可用符号表示
- 同一寻址方式有多种表达形式



(1) MOV AX, 1234H

(2) MOV AX, BX

(3) MOV AX, [2000H]

(4) MOV AX, [BX]

(5) MOV AX, [BX+06H]

(6) MOV AX, [BP+06H]

(7) MOV AX, [BX+SI]

(8) MOV AX, [BX+SI+06H]

已知: (BX)=2000H, (BP)=2000H, (DS)=3000H,

(SS)=4000H, (SI)=1000H, [32000H]=00H,

[32001H]=11H, [32006H]=22H, [32007H]=33H,

[33000H]=66H, [33001H]=77H, [33006H]=88H,

[33007H]=99H, [42006H]=44H, [42007H]=55H



➤ 由4部分组成:

标号:指令助记符 目的操作数,源操作数 ; 注释

- 标号表示该指令在主存中的逻辑地址
- 每个指令助记符就代表一种指令
- 目的和源操作数表示参与操作的对象
- 注释是对该指令或程序段功能的说明

➤ **r8**——任意一个**8位通用寄存器**

AH AL BH BL CH CL DH DL

➤ **r16**——任意一个**16位通用寄存器**

AX BX CX DX SI DI BP SP

➤ **reg**——代表**r8或r16**

➤ **seg**——段寄存器 **CS/DS/ES/SS**

一定要熟悉噢!

- **m8**——一个**8**位存储器操作数单元（所有主存寻址方式）
- **m16**——一个**16**位存储器操作数单元（所有主存寻址方式）
- **mem**——代表**m8**或**m16**

一定要熟悉噢！

- **i8**——一个8位立即数
- **i16**——一个16位立即数
- **imm**——代表i8或i16
- **dest**——目的操作数
- **src**——源操作数

一定要熟悉噢!

➤ Intel 8086指令系统共有117条基本指令，可分成6个功能组

- ① 数据传送类指令
- ② 算术运算类指令
- ③ 位操作类指令
- ④ 串操作类指令
- ⑤ 控制转移类指令
- ⑥ 处理机控制类指令

如何学习

- **指令的功能**——该指令能够实现何种操作。通常指令助记符就是指令功能的英文单词或其缩写形式
- **指令支持的寻址方式**——该指令中的操作数可以采用何种寻址方式
- **指令对标志的影响**——该指令执行后是否对各个标志位有影响，以及如何影响
- **其他方面**——该指令其他需要特别注意的地方，如指令执行时的约定设置、必须预置的参数、隐含使用的寄存器等



教学要求 (1)

1. 掌握8086的寄存器组和存储器组织
2. 掌握8086的寻址方式



课间休息

