

汇编语言程序设计

Review of Assembly Language Programming

目录

Contents

- 汇编语言编程基础
- PC的指令系统
- 汇编语言程序组织
- 分支与循环程序
- 子程序

汇编语言编程基础

第一部分

CPU的三种运行模式

- 实模式
 - 没有多任务，所有程序都在最高特权
 - 程序独占整个CPU
- 保护模式
 - Windows、Linux的运行模式
 - 分页分段机制、特权级保护、多任务
- 虚拟8086模式
 - 以任务形式在保护模式下运行
 - 为了支持旧的8086程序

程序可见寄存器组

位	31	16	15	8	7	0			
EAX				AH		AL	}	数据寄存器	
EBX				BH		BL			
ECX				CH		CL			
EDX				DH		DL			
ESP				SP			堆栈指针	}	指针寄存器
EBP				BP			基址指针		
ESI				SI			源变址	}	变址寄存器
EDI				DI			目的变址		
EIP				IP			指令指针	}	控制寄存器
FLAGS				FLAGS			标志		
				CS			代码段寄存器	}	段寄存器
				SS			堆栈段寄存器		
				DS			数据段寄存器		
				ES			附加段寄存器		
				FS					
				GS					

程序可见寄存器组

通用寄存器 / 数据寄存器

- 数据寄存器
 - 8位： AL AH BL BH CL CH DL DH
 - 16位： AX BX CX DX
 - 32位： EAX EBX ECX EDX [386]

程序可见寄存器组

通用寄存器 / 指针寄存器

- 堆栈指针寄存器SP、ESP（386以上）
 - 存放当前堆栈段**栈顶偏移量**，总是与SS堆栈段寄存器配合存取堆栈中的数据。
 - 实模式使用SP，保护模式使用ESP。
- 基址指针寄存器BP、EBP（386以上）
 - 存放**地址的偏移量部分或数据**。若存放偏移量时，缺省情况与**SS**配合。
 - 实模式使用BP，保护模式使用EBP。

程序可见寄存器组

通用寄存器 / 变址寄存器

- 变址寄存器SI、DI、ESI、EDI
 - 存放地址的偏移量部分或数据。若存放偏移量时，缺省情况与DS配合。
 - 实模式使用SI、DI，保护模式使用ESI、EDI。

程序可见寄存器组

段寄存器

- 段寄存器存放段基址。在实模式下存放段基地址（高16位），在保护模式下存放段选择符。
- 段选择符：用以选择描述符表中的一个描述符。
 - 描述符描述段的基地址、长度和访问权限。

程序可见寄存器组

段寄存器

- 代码段寄存器CS
 - 指定当前代码段，代码段中存放当前运行的程序段。
- 数据段寄存器DS
 - 指定当前运行程序所使用的数据段。
- 堆栈段寄存器SS
 - 指定当前堆栈段。
- 附加段寄存器ES
 - 指定当前运行程序所使用的附加数据段。
- FS、GS（386）
 - 指定当前运行程序的另外两个存放数据的存储段

程序可见寄存器组

控制寄存器

- 指令指针寄存器IP、EIP
 - 与CS段寄存器配合指出下一条要执行指令的地址，其中存放偏移量部分。
- 标志寄存器FLAGS
 - 也被称为状态寄存器，由运算结果特征标志和控制标志组成。

程序可见寄存器组

控制寄存器 / 标志寄存器

- 运算结果特征标志
 - 记录程序中运行结果的特征。
 - CF、PF、AF、ZF、SF、OF
- 控制标志
 - 控制处理器的操作，要通过专门指令才能使其变化。
 - IF、DF、TF

程序可见寄存器组

控制寄存器 / 标志寄存器 / 运算结果特征标志

- CF(Carry Flag) - 进位标志
 - 运算结果最高位是否向前产生进位或借位
- PF(Parity Flag) - 奇偶标志
 - 计算结果最低8位含1个数是否为偶数
- AF(Auxiliary carry Flag) - 辅助进位标志
 - 计算结果最低4位是否向前产生进位或借位
- ZF(Zero Flag) - 零标志 (记录是否为0)
- SF(Sign Flag) - 符号标志 (记录是否为负)
- OF(Overflow Flag) - 溢出标志 (记录是否溢出)

程序可见寄存器组

控制寄存器 / 标志寄存器 / 控制标志

- IF(Interrupt Flag) - 中断允许标志
 - =1, 允许CPU响应外部可屏蔽中断请求INTR
- DF(Direction Flag) - 方向标志
 - 专门服务于字符串操作指令
 - =1, 表示串操作数地址为自动减量(高地址到低)
 - =0, 相反
- TF(Trap Flag) - 陷阱标志
 - 用于程序调试
 - =1, CPU处于单步方式
 - =0, 处于连续方式

存储顺序

小端方式和大端方式

- 以小端方式为例
 - 低字节在前，低位保存在内存的低地址中
- 小端/大端方式一定是按字节存储的，但是可以以字节、字、双字等各种形式读出。
- 将9025H按小端方式存储到内存的1000H单元，则结果为
 - 1000H : 25H
 - 1001H : 90H

实模式寻址

分段管理

- 物理地址的计算方法
 - $10H * \text{段基址} + \text{偏移量}$

I/O地址空间

概念

- 外设与主机的信息交换是通过外设接口进行的。
- 不同的外设接口中含有的寄存器数量不同。
- 系统给每个接口中的寄存器赋予一个端口地址。
- 这些端口地址组成的地址空间为I/O地址空间。

PC的指令系统

第二部分

寻址方式

- 与数据有关的寻址方式
 - 使用MOV指令
- 与转移地址有关的寻址方式
 - 使用JMP指令

寻址方式

与数据有关的寻址方式 / 立即寻址方式

- MOV EAX, 立即数
- 用于给寄存器或者内存单元赋初值。
- 例子
 - MOV EAX,1234H

寻址方式

与数据有关的寻址方式 / 寄存器寻址方式

- 操作数直接包含在寄存器中。
 - 由指令指定寄存器号。
- 例子
 - `MOV BX,AX`

寻址方式

与数据有关的寻址方式 / 直接寻址方式

- 操作数的有效地址EA直接包含在指令中。
- MOV AL,[78H]
 - [78H]是一个普通变量的有效地址
- MOV EBX,ES:MEM
 - 段超越前缀ES：使用ES所指向的附加数据段
- MOV AL,VAR
 - VAR是内存变量名，代表一个内存单元的符号地址

寻址方式

与数据有关的寻址方式 / 直接寻址方式

- 有效地址存放在代码段的指令操作码之后，但操作数本身在存储器中，所以必须先求出操作数的物理地址。
- 普通变量缺省情况是存放在DS所指向的数据段，但允许使用段超越前缀指定为其它段。
- 物理地址=段基址*10H + 有效地址EA

寻址方式

与数据有关的寻址方式 / 直接寻址方式

操作类型	约定段寄存器	允许指定的段寄存器	偏移量
1. 指令	CS	无	IP
2. 堆栈操作	SS	无	SP
3. 普通变量	DS	ES、SS、CS	EA
4. 字符串指令的源串地址	DS	ES、SS、CS	SI
5. 字符串指令的目标串地址	ES	无	DI
6. BP用作基址寄存器	SS	DS、ES、CS	EA

寻址方式

与数据有关的寻址方式 / 寄存器间接寻址方式

- 操作数有效地址在基址寄存器BX、BP或变址寄存器SI、DI中，而操作数在存储器中的寻址方式。
- 若指令中使用的是BX、SI、DI、EAX、EBX、ECX、EDX、ESI、EDI，则缺省情况操作数在数据段，即它们默认与DS段寄存器配合。
- 若使用的是BP、EBP、ESP，则缺省情况默认与SS段寄存器配合。
- 均允许使用段超越前缀。

寻址方式

与数据有关的寻址方式 / 寄存器间接寻址方式

- MOV AL,[BX]
 - DS:[BX] -> AL
- MOV AX,[BP]
 - SS:[BP] -> AX

寻址方式

与数据有关的寻址方式 / 寄存器相对寻址方式

- 操作数的有效地址是一个基址(BX、BP)或变址寄存器(SI、DI)的内容和指令中给定的一个位移量(displacement)之和。
- 386以上允许使用任何32位通用寄存器。位移量可以是一个字节、一个字、一个双字（386以上）的带符号数。
- 有效地址
 - $EA = (\text{基址} < \text{或变址} > \text{寄存器}) + \text{disp}$
 - $EA = (\text{32位通用寄存器}) + \text{disp}$

寻址方式

与数据有关的寻址方式 / 寄存器相对寻址方式

- `MOV AL, TABLE[BX]`
 - `MOV AL, [BX+TABLE]`
 - `(DS:[BX+TABLE]) -> AL`
 - 访问一维数组
 - TABLE是数组起始地址的偏移量
 - 寄存器中是数组元素的下标乘以元素的长度(占用字节数)
- `MOV AL, 8[BX]`
 - `MOV AL, [BX+8]`

寻址方式

与数据有关的寻址方式 / 基址变址寻址方式

- $EA = (\text{基址寄存器}) + (\text{变址寄存器})$
- 缺省使用段寄存器的情况由基址寄存器决定。若使用BP、ESP或EBP，缺省与SS配合；若使用BX或其它32位通用寄存器，则缺省与DS配合。
- 允许使用段超越前缀。

寻址方式

与数据有关的寻址方式 / 基址变址寻址方式

- MOV AL,[BX][SI]
 - MOV AL,[BX+SI]
 - (DS:[BX+SI]) -> AL
 - 访问一维数组
 - BX存放数组起始地址的偏移量
 - SI存放数组元素的下标乘以元素的长度

寻址方式

与数据有关的寻址方式 / 相对基址变址寻址方式

- $EA = (\text{基址寄存器}) + (\text{变址寄存器}) + \text{disp}$
- 缺省使用段寄存器的情况由基址寄存器决定。若使用BP、ESP或EBP，缺省与SS配合；若使用BX或其它32位通用寄存器，则缺省与DS配合。
- 允许使用段超越前缀。

寻址方式

与数据有关的寻址方式 / 相对基址变址寻址方式

- `MOV AL,ARY[BX][SI]`
 - `MOV AL,[BX+SI+ARY]`
 - `(DS:[BX+SI+ARY]) -> AL`
 - 访问二维数组
 - ARY为数组起始地址的偏移量
 - $BX = \text{行下标} * \text{一行占用的字节数}$ (某行首与数组起始地址距离)
 - $SI = \text{列下标} * \text{一列占用的字节数}$ (某列与所在行首的距离)

寻址方式

与转移地址有关的寻址方式 / 段内直接寻址方式

- 要转向的有效地址
 - $EA = (IP) + \{ 8\text{位} / 16\text{位} \} \text{ disp}$
 - $EA = (EIP) + \{ 8\text{位} / 32\text{位} \} \text{ disp}$
- 短转移
 - JMP SHORT L1
- 近转移
 - JMP L2 或 JMP NEAR PTR L2

寻址方式

与转移地址有关的寻址方式 / 段内间接寻址方式

格 式	举 例	注 释
JMP 通用寄存器	JMP BX	16 位转向地址在 BX 中，其值送给 IP
	JMP EAX	32 位转向地址在 EAX 中，其值送给 EIP
JMP 内存单元	JMP WORD PTR VAR	16 位转向地址在 VAR 字型内存变量中
	JMP WORD PTR [BX]	16 位转向地址在 BX 所指向的内存变量中
	JMP DWORD PTR DVAR	32 位转向地址在 DVAR 双字型内存变量中
	JMP DWORD PTR [EBX]	32 位转向地址在 EBX 所指向的内存变量中

寻址方式

与转移地址有关的寻址方式 / 段间直接寻址方式

- 执行时把偏移量送给IP，段基址（段选择符）送给CS，即可实现段间直接转移。
- MOV FAR PTR L1

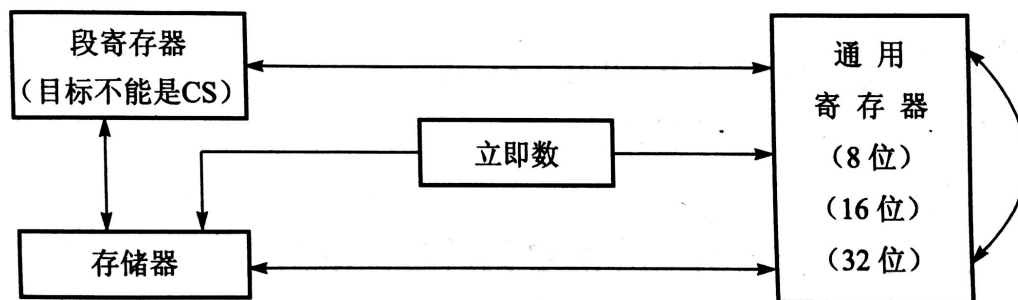
寻址方式

与转移地址有关的寻址方式 / 段间间接寻址方式

- 用一个双字内存变量（**不能用寄存器了**）中的低16位取代IP值，高16位取代CS值，从而实现段间转移。
- `MOV DWORD PTR [BX]`
 - DWORD表示[BX]指向一个双字变量
- 当操作数长度为32位时，则用一个三字内存变量(48位)中的低32位取代EIP值，高16位取代CS值。
- `MOV FWORD PTR [EBX]`
 - FWORD表示[EBX]指向一个三字变量

数据传送指令

通用数据传送指令 / 传送指令MOV



- 立即数不能作为目标操作数
- 立即数不能直接送段寄存器
- 目标寄存器不能是CS
- 段寄存器之间不能相互传送
- 两个存储单元之间不能相互传送

数据传送指令

通用数据传送指令 / 带符号扩展的数据传送指令MOVSX

- 只有386以上机器提供
- 用于对带符号数的扩展
- MOVSX DST, SRC
 - SRC -> DST, DST空出的位用SRC的符号位填充
 - DST必须为16或32位
 - SRC可以是8或16位, 或存储器操作数, 但不可为立即数

数据传送指令

通用数据传送指令 / 入栈指令PUSH

- PUSH SRC
 - 8086、8088
 - SRC是16位寄存器操作数或存储器操作数，不可为立即数
 - 80286以上
 - SRC可以是16或32位(386).....，也可以是立即数
- 功能
 - 先修改堆栈指针使其指向新的栈顶
 - SRC为16位，则 $SP-2 \rightarrow SP$
 - SRC为32位，则 $SP-4 \rightarrow SP$
 - 然后把SRC压入栈顶单元

数据传送指令

通用数据传送指令 / 出栈指令POP

- POP DST
 - DST可以是16或32位(386)的寄存器操作数和存储器操作数
 - 也可以是除CS寄存器外的任何段寄存器
- 功能
 - 先把堆栈指针所指向单元的内容弹出到DST
 - 然后修改堆栈指针以指向新的栈顶
 - DST为16位, 则 $SP+2 \rightarrow SP$
 - DST为32位, 则 $SP+4 \rightarrow SP$

数据传送指令

通用数据传送指令 / 交换指令XCHG

- XCHG OPR1,OPR2
 - OPR可以是8、16、32位(386)存储器操作数或寄存器操作数，不能是立即数
 - 其中之一必须是寄存器操作数

数据传送指令

输入输出指令 / 输入指令IN

- IN ACR,PORT
 - 把外设端口PORT的内容传给累加器ACR
- 传送8、16、32(386)的数据，相应的累加器选择AL、AX、EAX。
- 端口号在0-255之间，可以直接写在指令中；
- 端口号大于255，通过**DX寄存器** “间接寻址”。



端口号存在DX寄存器中

数据传送指令

输入输出指令 / 输出指令OUT

- OUT PORT,ACR
 - 把累加器中的内容传送给外设端口
- 传送8、16、32(386)的数据，相应的累加器选择AL、AX、EAX。
- 端口号在0-255之间，可以直接写在指令中；
- 端口号大于255，通过**DX寄存器** “间接寻址”。



端口号存在DX寄存器中

数据传送指令

地址传送指令 / 传送有效地址指令LEA

- LEA REG, SRC Load Effective Address
 - 源操作数SRC必须是存储器操作数
- 把操作数的有效地址传给指定寄存器

- LEA BX, ASC
 - 同MOV BX, OFFSET ASC
- LEA BX, ASC[SI]
 - 把DS:[SI+ASC]中的16位偏移量送BX

数据传送指令

标志传送指令

- 16位标志进栈指令PUSHF
 - SP-2 -> SP, 压入FLAGS到栈顶单元
- 16位标志出栈指令POPF
- 32位标志进栈指令PUSHFD
 - ESP-4 -> ESP, 压入EFLAGS到栈顶单元
- 32位标志出栈指令POPFD
- 标志送AH指令LAHF
 - FLAGS低8位送AH寄存器
- AH送标志寄存器指令SAHF
 - AH寄存器内容送标志寄存器FLAGS低8位

算术运算指令

类型转换指令

- 字节扩展成字CBW Convert Byte to Word
 - AL中符号位扩展到AH中
- 字扩展成双字CWD Convert Word to Double Word
 - AX符号位扩展到DX中
- [386可用]字扩展成双字CWDE CWD Extended
 - AX符号位扩展到EAX中
- [386可用]双字扩展成四字CDQ Quad-Word
 - EAX符号位扩展到EDX中

算术运算指令

二进制加法指令

- ADD DST, SRC
- ADC DST, SRC 带进位加法指令
 - $(DST) + (SRC) + CF \rightarrow DST$
- INC DST
- XADD DST, SRC 互换并加法指令
 - 原DST的内容在SRC中，和在DST中

对标志符的影响

ADD、ADC、XADD均影响OF、SF、ZF、AF、PF、CF标志

INC不影响CF标志，影响其他5个算术运算特征标志

无符号数相加结果若CF=1，说明溢出；带符号数相加结果若OF=1，说明溢出

算术运算指令

二进制减法指令

- SUB DST, SRC
- SBB DST, SRC 带借位减法指令
 - $(DST) - (SRC) - CF \rightarrow DST$
- DEC DST
- CMP DST, SRC
 - 不改变DST和SRC的值，但是影响6个标志位
- NEG DST 求补指令
 - 对目标操作数求反加一

算术运算指令

二进制减法指令 / 求补指令NEG

- 用于求一个数的相反数
 - $0 - (\text{DST}) \rightarrow \text{DST}$
- 对CF和OF的影响如下：
 - 对操作数所能表示的最小负数（8位则为-128）求补，原操作数不变，OF置1
 - 操作数为0，CF清0
 - 对非0操作数求补，CF置1

算术运算指令

二进制乘法指令 / 无符号数乘法指令 MUL

- MUL SRC_{reg / m}
 - SRC只能是寄存器或存储器数
 - 另一个乘数提前放在累加器中
- 计算结果
 - 8位乘法：累加器AL，结果送AX
 - 16位乘法：累加器AX，结果送DX:AX
 - 32位乘法：累加器EAX，结果送EDX:EAX
 - 若乘积的高半部分为0，则CF和OF清0；否则置CF和OF为1，其它标志不确定

算术运算指令

二进制乘法指令 / 带符号数乘法指令IMUL

- IMUL SRC_{reg / m}
 - SRC只能是寄存器或存储器数
 - 另一个乘数提前放在累加器中
- 计算结果
 - 8位乘法：累加器AL，结果送AX
 - 16位乘法：累加器AX，结果送DX:AX
 - 32位乘法：累加器EAX，结果送EDX:EAX
 - 若乘积的高半部分为低半部分的符号扩展，则CF和OF清0；否则置CF和OF为1，其它标志不确定

算术运算指令

二进制乘法指令 / 带符号数乘法指令IMUL

- 还有以下几种格式 P61
 - IMUL REG, SRC_{reg / m}
 - IMUL REG, imm₈
 - IMUL REG, SRC_{reg / m}, imm₈

算术运算指令

二进制除法指令

- $\text{DIV SRC}_{\text{reg} / \text{m}}$
 - SRC是除数，被除数实现放在隐含的寄存器中
- 计算结果
 - 如果参数是 r8/m8,
 - 将把 AX 做被除数，商 \rightarrow AL，余数 \rightarrow AH
 - 如果参数是 r16/m16
 - 将把 DX:AX 做被除数，商 \rightarrow AX，余数 \rightarrow DX
 - 如果参数是 r32/m32
 - 将把 EDX:EAX 做被除数，商 \rightarrow EAX，余数 \rightarrow EDX
- 标志不确定

逻辑指令

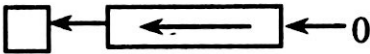
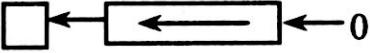
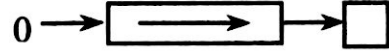
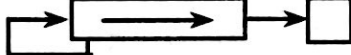
逻辑运算指令

名 称	格 式	功 能	标 志
逻辑非	NOT DST	(DST)按位变反送 DST	不影响
逻辑与	AND DST, SRC	$DST \leftarrow (DST) \wedge (SRC)$	CF 和 OF 清 0, 影响 SF、ZF 及 PF, AF 不定
逻辑测试	TEST OPR1, OPR2	$OPR1 \wedge OPR2$	同 AND 指令
逻辑或	OR DST, SRC	$DST \leftarrow (DST) \vee (SRC)$	同 AND 指令
逻辑异或	XOR DST, SRC	$DST \leftarrow (DST) \vee (SRC)$	同 AND 指令

- XOR可用于清零，或者大小写转换(20H)
- TEST用于测试某一个位(AND)，只影响标志位
 - TEST AL, 80H
 - 若D7=0，说明AND结果为0，那么ZF置1
 - 若D7=1，说明AND结果为1，那么ZF置0

逻辑指令

基本移位指令

名 称	格 式	功 能	标 志
逻辑左移	SHL DST,CNT		CF 中总是最后移出的一位, ZF、SF、PF 按结果设置,当 CNT= 1 时,移位使符号位变化,OF 置 1,否则清 0
算术左移	SAL DST,CNT		同上
逻辑右移	SHR DST,CNT		同上
算术右移	SAR DST,CNT		同上

注: 1. 当 CNT>1 时, OF 值不确定。

说明: DST 可以是 8 位、16 位或 32 位的寄存器或存储器操作数, CNT 是移位位数。

2. 对 CNT 的限定是:




当 CNT=1 时, 直接写在指令中;

适用于 8086、8088

当 CNT>1 时, 由 CL 寄存器给出;

适用于 80X86 系列的所有型号

当 CNT>1 时, 由指令中的 8 位立即数给出; 适用于 80286 以上

3. 功能图中的符号表示:  — CF;  — 数据流向;  — 操作数

逻辑指令

基本移位指令

逻辑移位和算术移位的区别

逻辑移位：连同符号位一起移动。无论是左移还是右移，空缺处都补0。

算术移位：固定符号位不动。左移则空缺处补0，右移则空缺处根据符号位补0或1。

程序控制指令

转移指令 / 无条件转移指令JMP

去第33-36页幻灯片看看
(右键->打开超链接)

程序控制指令

转移指令 / 条件转移指令 / 根据标志位跳转

JE ; 等于则跳转
JNE ; 不等于则跳转

JZ ; 为 0 则跳转
JNZ ; 不为 0 则跳转

JS ; 为负则跳转
JNS ; 不为负则跳转

JC ; 进位则跳转
JNC ; 不进位则跳转

JO ; 溢出则跳转
JNO ; 不溢出则跳转

JA ; 无符号大于则跳转
JNA ; 无符号不大于则跳转
JAE ; 无符号大于等于则跳转
JNAE ; 无符号不大于等于则跳转

JG ; 有符号大于则跳转
JNG ; 有符号不大于则跳转
JGE ; 有符号大于等于则跳转
JNGE ; 有符号不大于等于则跳转

JB ; 无符号小于则跳转
JNB ; 无符号不小于则跳转
JBE ; 无符号小于等于则跳转
JNBE ; 无符号不小于等于则跳转

JL ; 有符号小于则跳转
JNL ; 有符号不小于则跳转
JLE ; 有符号小于等于则跳转
JNLE ; 有符号不小于等于则跳转

JP ; 奇偶位置位则跳转
JNP ; 奇偶位清除则跳转
JPE ; 奇偶位相等则跳转
JPO ; 奇偶位不等则跳转

程序控制指令

转移指令 / 条件转移指令 / 测试CX或ECX的值

- JCXZ LABEL
 - CX为0则跳转
- JECXZ LABEL
 - ECX为0则跳转

程序控制指令

循环指令LOOP

- LOOP LABEL
 - $(CX)-1 \rightarrow CX$, 若 $(CX) \neq 0$, 则转向标号处执行循环体
 - LOOP指令前, 应先给CX或ECX赋初值

程序控制指令

子程序相关 / 子程序调用指令CALL

- CALL DST P79
 - 执行时先把返回地址压入堆栈，再形成子程序入口地址，最后把控制权交给子程序
- 段内直接/间接、段间直接/间接

程序控制指令

子程序相关 / 子程序返回指令RET

- RET
 - 按照CALL指令入栈的逆序，从栈顶弹出返回地址
 - 偏移量送IP/EIP，FAR型还需弹出一个字/双字到CS
 - 反汇编后，段间返回指令为RETF
- RET imm₁₆
 - 按照CALL指令入栈的逆序，从栈顶弹出返回地址
 - 偏移量送IP/EIP，FAR型还需弹出一个字到CS
 - 修改栈顶指针 $(SP) = (SP) + \text{imm}_{16}$
 - 此处对SP的修改未包含将IP和CS出栈所导致的修改，因此计算总的SP变化量时，应全部考虑

处理器控制指令

标志操作指令

汇编格式	功 能	影响标志
CLC(Clear Carry)	把进位标志 CF 清 0	CF
STC(Set Carry)	把进位标志 CF 置 1	CF
CMC(Complement Carry)	把进位标志 CF 取反	CF
CLD(Clear Direction)	把方向标志 DF 清 0	DF
STD(Set Direction)	把方向标志 DF 置 1	DF
CLI(Clear Interrupt)	把中断允许标志 IF 清 0	IF
STI (Set Interrupt)	把中断允许标志 IF 置 1	IF

串操作指令

串指令

- P85 例3.98

汇编语言程序组织

第三部分

数据与符号定义伪指令

数据定义伪指令