**汇编期末**

* 非压缩BCD
* 寻址方式，什么情况用寄存器
* 分段的定义，怎么分
* 地址计算，物理地址与逻辑地址转化、定义
* MOV语法错误，MOV指令图示
* 减法指令
* xlat表格
* les、lds
* CBW
* NEG四个方面
* 指令的等价形式
* 指令对错判断，多角度
* CMP=SUB、TEST=AND区别，不改变目的操作数
* jmp、短转移、近转移范围
* jcc
* 循环条件，LOOP、LOOPZ、LOOPNZ，什么情况终止循环
* 子程序
* 变量定义、标号定义
* 简化段定义格式，每一个符号
* 汇编过程：asm-obj-exe
* DOS中断字符串输入输出
* DEBUG指令，退出快捷键Q
* 字符操作指令，lengthof
* 子程序
* 缓冲区的定义与使用
* 排序、查找算法，书上例子
* 程序作用
* DB、DW画图
* 程序设计题：

简化段完整定义格式

字符操作指令，lengthof

字符串操作程序

函数过程、子程序call

字符串中剔除空格，冒泡

程序框架图

字符串操作程序

translate指令

**第一章**

* 非压缩BCD：高四位为0，低四位表示一个BCD，16位

例：0807H：0000 1000 0000 0111B

压缩BCD：一字节两个BCD

例：87H：1000 0111B

* 寻址方式：
* 寄存器

1. 通用寄存器：
2. 数据寄存器：AX、BX、CX、DX

高位为AH（D15-D8），低位AL（D7-D0）

1. 变址寄存器：SI（源操作数）、DI（目的操作数）
2. 指针寄存器：SP（指向栈顶）
3. 标志寄存器：
4. 状态标志：

OF：Overflow（两个相同符号的数运算后结果符号相反，OF=1）

CF：Carry（加减法有进位或者借位CF=1）

SF：Sign

ZF：Zero

PF：Parity

AF：Adjust

1. 控制标志：

DF：Direction

IF：Interruptenable

TF：Trap

1. 指令指针寄存器：

IP：表示将要执行的指令在主存中的位置，不能赋值，执行完一条指令指向下一条指令

1. 段寄存器：

CS：代码段

SS：堆栈段

DS：数据段

ES：附加段

* 存储器组织

1. 数据存储格式：

二进制，从右向左0开始编号

字节：一字节，8位，D7-D0

字：两字节，16位，D15-D0

双字：四字节，32位，D31-D0

[ ]：表示存储单元的内容

例：字节：[0002H]=34H 字：[0002H]=1234H 双字：[0002H]=78561234H

15……………………………………8 7…………………………………………0 <-位偏移

|  |  |  |
| --- | --- | --- |
|  |  | 高地址 |
|  |  |  |
| 78H | 56H | 0004H |
| 12H | 34H | 0002H |
| 高字节 | 低字节 | 0000H 低地址 |

1. 存储器分段管理：

段基地址：段内偏移地址

段基地址：段地址，逻辑段在主存中的起始位置

段内偏移地址：偏移地址，主存单元距离段起始位置的偏移量

物理地址：绝对地址，每个存储单元唯一的20位地址

逻辑地址：“段地址：偏移地址”的形式

**20位物理地址=逻辑地址左移4位（二进制的4位）+偏移地址**

同一物理地址可以有多种逻辑地址

1. 段寄存器的作用

CS：存放程序的指令序列，IP指示代码段指令的偏移地址，CS:IP取得要执行的下一条指令

SS：堆栈所在的主存区域，SP指示栈顶的偏移地址，SS:SP操作堆栈中的数据

DS：当前运行程序所用数据，EA表示有效地址，通过各种主存寻址方式得到的存储器中操作数的偏移地址，DS:EA一般数据，DS:SI串操作源操作数

ES：附加的数据段，用于数据保存，串操作指令目的操作数存放区域

**分段管理符合程序模块化思想，利于编写模块化程序**

**一个段64kb，按需分配**

**两个段并不一定完全分开，甚至可以完全重叠，但内容不冲突**

**程序指令序列必须CS，堆栈必须SS，串操作必须ES**

**数据默认DS，有时ES，可以放在任意段中，需指明**

立即数不能直接赋值给段寄存器，需要通过通用寄存器转换

* 寻址方式

1. 立即数寻址方式：给寄存器/存储单元赋值

e.g. mov al, 05h

1. 寄存器寻址方式

e.g. mov bx, ax

1. 存储器寻址方式
2. 直接寻址方式

e.g. mov ax, [2000h] ;ax<-ds:[2000h]

mov ax, es:[2000h] ;ax<-es:[2000h]

1. 寄存器间接寻址方式

有效地址只能在基址寄存器BX或变址寄存器SI、DI中，默认DS段

e.g. mov ax, [si] ;ax<-ds:[si]

1. 寄存器相对寻址方式

寄存器可以是BX、BP、SI、DI

EA=BX/BP/SI/DI+8/16位位移量

BX、SI、DI默认数据段DS，BP默认堆栈段SS

e.g. mov ax, [di+06h] ;ax<-ds:[di+06h]

mov ax, [bp+06h] ;ax<-ss:[bp+06h]

mov ax, [bp] ;ax<-ss:[bp]

1. 基址变址寻址方式

EA=BX+SI或BP+DI

BX默认数据段DS，BP默认堆栈段SS

e.g. mov ax, [bx+si] ;ax<-ds:[bx+si]

mov ax, [bp+di] ;ax<-ss:[bp+di]

mov ax, ds:[bp+di] ;ax<-ds:[bp+di]

1. 相对基址变址寻址方式

EA=BX+SI+8/16位位移量或BP+DI+8/16位位移量

e.g. mov ax, [bx+si+06h]

mov ax, [si+count]

mov ax, [bx+si+wnum]

mov ax, [bx][si]

mov ax, count[si]

**第二章**

* 传送指令MOV

1. 立即数传送至通用寄存器（不包括段寄存器）或存储单元

mov reg/mem, imm

1. 寄存器传送至寄存器（包括段寄存器）或存储单元

mov reg/mem/seg, reg

1. 存储单元传送至寄存器（包括段寄存器）

mov reg/seg, mem

1. 段寄存器传送至通用寄存器（不包括段寄存器）或存储单元

mov reg/mem, seg

立即数不能对段寄存器直接赋值

不能两个都是存储器，两个都是段寄存器

不能改变立即数的值

* 地址传送

lea r16, mem ；将存储器的逻辑地址送至指定的寄存器

lds r16, mem ；r16<-mem，ds<-mem+2

；主存mem指定的字送至r16，下一字送至ds或es（les）

les r16, mem ；r16<-mem，es<-mem+2

* XCHG

xchg reg，reg/mem

不能是存储器和存储器，不能是立即数

可以是字或字节

* XLAT

将bx指定的缓冲区，al指定的位移处的数据取出赋给al

默认使用bx、al

表格首地址bx，相对表格首地址的位移量al

xlat label

* 栈

|  |  |  |
| --- | --- | --- |
| 高地址 | 存储器 |  |
|  |  |  |
|  |  |
| 栈底-> |  | <-sp（栈顶） |
|  |  |  |
|  |  |  |
|  |  |  |
| ss-> | 0000h |

栈顶地址小

push r16/m16/seg ；sp<-sp-2, ss:[sp]<-r16/m16/seg

pop r16/m16/seg ；r16/m16/seg<-ss:[sp], sp<-sp+2

push、pop只能对字节量操作，不能对字操作，al/bh

* 四则运算

**add reg，imm/reg/mem**

**add mem，imm/reg**

两个操作数不能都是存储器，立即数不能作为目的操作数

存储器指明word/byte ptr

**（add、adc、sub、sbb、cmp）**

**结果送至目的操作数**

* + 加法

1. add
2. adc：完成add的基础上加上cf的数值

e.g. 利用dx.ax进行高精度加法（dx高位，ax低位）

mov ax, 4652h ；ax=4652h

add ax, 0f0f0h ；ax=3742h，cf=1（4652h+0f0f0h=13742h）

mov dx, 0234h ；dx=0234h

adc dx, 0f0f0h ；dx=f325h，cf=0

；0234h+0f0f0h+1=f325h，加上了ax的cf=1

dx.ax=0234 4652h+f0f0 f0f0h=f325 3742h

1. inc：++

inc reg/mem

e.g. inc byte ptr [bx]

inc bx

* + 减法

1. sub
2. sbb：减去cf
3. dec：--
4. neg：求补，用0减去操作数，将结果返回操作数=操作数按位取反+1

neg reg/mem

e.g. mov ax, 0ff64h ；有符号数ff64h

neg al ；ax=0ff9ch，of=0，sf=1，zf=0，pf=1，cf=1

sub al, 9dh ；ax=0ffffh

neg ax ；ax=0001h，sf=0，pf=0

dec al ；ax=0000h，zf=1，pf=1

neg ax ；ax=0000h，cf=0

1. cmp

目的操作数减去源操作数，与sub相同，标志状态改变，目的操作数不变

* + 乘法

1. mul

mul r8/m8 ；ax<-al\*r8/m8

mul r16/m16 ；dx.ax<-al\*r16/m16

隐含使用ax和dx

1. imul

有符号

* + 除法

1. div

div r8/m8 ；al<-ax÷r8/m8商

；ah<- ax÷r8/m8余数

div r16/m16 ；ax<-dx.ax÷r16/m16商

；dx<-dx.ax÷r16/m16余数

1. idiv

有符号

* 符号扩展

cbw：al最高有效位扩展到ah

al最高有效位0，ah=00h；为1，ah=ffh

cwd：ax最高有效位扩展到dx

ax最高有效位0，dx=0000h；为1，dx=ffffh

* 十进制调整指令
  + 压缩BCD

daa

das

* + 非压缩BCD

1. aaa
2. aas
3. aam
4. aad

* 逻辑运算指令

**and reg，imm/reg/mem**

**and mem，imm/reg**

**（or、xor、test同and）**

源操作数可以为任意寻址方式，目的操作数只能为立即数外的其他寻址方式，两个操作数不同时为存储器寻址方式。

**and、or、xor、not结果送至目的操作数**

1. and

e.g. and bl，1111 0110b ；对d0、d3清零

1. or

置1

1. xor

求反

e.g. xor al, al ；清零

1. not

可以为立即数外任何寻址方式，不影响标志位

1. test

操作同and，不保存执行结果，只改变状态标志

* 移位指令

shl reg/mem, 1/cl ;逻辑左移1/cl位；最低位补0，最高位进入cf

shr reg/mem, 1/cl ;逻辑右移1/cl位；最高位补0，最低位进入cf

sal reg/mem, 1/cl ;算术左移1/cl位；同shl

sar reg/mem, 1/cl ;算术右移1/cl位；最高位不变，最低位进入cf

**shl与sal完全相同；**

**shr不考虑符号位，sar保留原符号，且最低位为1时结果与除法不同**

e.g. -5（FBH）sar为-3（FDH），idiv为-2

* 循环移位

rol reg/mem, 1/cl ；不带cf循环左移，直接舍弃

ror reg/mem, 1/cl ；不带cf循环右移

rcl reg/mem, 1/cl ；带cf循环左移，低1位入cf，将cf纳入循环位

rcr reg/mem, 1/cl ；带cf循环右移

* 无条件转移指令JMP
* 段内转移：当前代码段64KB范围内转移，不需要更改CS段地址，只改变IP偏移地址；

短转移：short jump转移范围用8位数（-128～127之间的位移量），±16KB

近转移：near jump地址位移用一个16位数表达，±32KB

* 段间转移：far jump远转移，从当前代码段跳到另一个代码段，CS、IP更改

逻辑地址：32位远指针，转移的目标地址必须用一个32位数表达

1. 段内转移，相对寻址

jmp label ；ip<-ip+位移量：label的偏移地址到目标指令的偏移地址的地址位移

1. 段内转移，间接寻址

jmp r16/m16 ；ip<-r16/m16

将16位寄存器或者主存单元内容送入IP寄存器，CS不变

e.g. jmp ax

jmp word ptr[2000h]

1. 段间转移，直接寻址

jmp far label ；ip<-label的偏移地址，cs<-label的段地址

1. 段间转移，间接寻址

jmp far ptr mem ；ip<-[mem]，cs<-[mem+2]

用一个双字存储单元表示要跳转的目标地址，存放在两个字单元中，低位送IP，高位送CS寄存器

e.g. mov word ptr[bx], 0

mov word ptr[bx+2], 1500h

jmp far ptr[bx] ;转移到1500h:0

* 条件转移指令JCC
  + 判断单个标志位

e.g. test al，80h ；测试最高位，根据状态标志0/1跳转

1. 为零或相等ZF，JZ/JE和JNZ/JNE

JZ：jump if zero，ZF=1，结果为0跳转

1. 正负SF，JS和JNS

JS：jump if sign，最高位为1，SF=1，符号为负跳转

1. 溢出OF，JO和JNO

JO：jump if overflow，OF=1，有溢出跳转

1. ‘1’的个数奇偶PF，JP/JPE和JNP/JPO

JP：jump if parity，PF=1，最低字节‘1’的个数为0或偶数跳转

1. 进位或借位CF，JC/JB/JNAE和JNC/JNB/JAE

JC：jump if carry，CP=1，有进位或者借位跳转

* + 比较无符号：JB/JNAE、JNB/JAE、JBE/JNA、JNBE/JA
  + 比较有符号：JL/JNGE、JNL/JGE、JLE/JNG、JNLE/JG
* 循环

jcxz label ；cx=0转移，否则顺序执行

loop label ；cx<-cx-1；cx≠0循环：ip<-ip+位移量；否则顺序执行

loopz/loope label ；cx<-cx-1；cx≠0且zf=1循环：ip<-ip+位移量；否则顺序执行

loopnz/loopne label；cx<-cx-1；cx≠0且zf=0循环：ip<-ip+位移量；否则顺序执行

* 子程序
* 格式

过程名 proc[near/far]

过程体

过程名 endp

；near属性只能被相同代码段的其他程序调用，段内近调用，微、小、紧凑默认

；far属性能被相同或不同的代码段程序调用，段间远调用，中、大、巨默认

* 调用指令CALL

将返回地址压入栈

1. 段内调用，相对寻址

call label ；sp<-sp-2，ss:[sp]<-ip，ip<-ip+16位位移量

1. 段内调用，间接寻址

call r16/m16 ；sp<-sp-2，ss:[sp]<-ip，ip<-r16/m16

1. 段间调用，直接寻址

call far ptr label ；sp<-sp-2，ss:[sp]<-cs

；sp<-sp-2，ss:[sp]<-ip

；ip<-label偏移地址，cs<-label段地址

1. 段间调用，间接寻址

call far ptr mem ；sp<-sp-2，ss:[sp]<-cs

；sp<-sp-2，ss:[sp]<-ip

；ip<-[mem]，cs<-[mem+2]

* 返回指令RET

直接从栈顶取内容作为返回地址

1. 无参数段内返回

ret ；ip<-ss:[sp]，sp<-sp+2

1. 有参数段内返回

ret i16 ；ip<-ss:[sp]，sp<-sp+2，sp<-sp+i16

1. 无参数段间返回

ret ；ip<-ss:[sp]，sp<-sp+2

；cs<-ss:[sp]，sp<-sp+2

1. 有参数段间返回

ret i16 ；ip<-ss:[sp]，sp<-sp+2

；ip<-ss:[sp]，sp<-sp+2，sp<-sp+i16

**第三章**

* 语句格式
* 标号：执行性语句中，处理器指令在主存中的逻辑地址，主要用于指示分支、循环等程序的目的地址，可有可无

e.g. ……

again: ………

loop again

* 名字：说明性语句中，反映变量、段、子程序的逻辑地址，可以是变量名，段名，子程序名
  1. 标号和名字的属性：

1. 地址属性：对应存储单元的逻辑地址，包括段地址和偏移地址
2. 类型属性：标号、子程序名的类型：NEAR（近：段内）、FAR（远：段间）

变量名：BYTE（字节）、WORD（字）、DWORD（双字）

1. 地址操作符：

[ ] 存储器地址指针;

$ 当前偏移地址;

：段地址寄存器;

offset 名字或标号的偏移地址;

seg 名字或标号的段地址

1. 类型操作符：

* PTR：使名字或标号具有指定的类型

WORD, BYTE, DWORD, FWORD, QWORD, TBYTE

NEAR, FAR

STRUCT, RECORD, UNION, TYPEDEF

* THIS：创建采用当前地址但为指定类型的操作数
* TYPE：返回一个字量数值，表明名字或标号的类型
* SIZEOF 返回值=LENGTHOF 返回值\*TYPE 返回值
* 保留字：关键字，编程语言本身需要使用的各种具有特定含义的标识符
* 标识符：最多31个字母、数字、\_、$、?、@组成，不能以数字开头，唯一
* 助记符：帮助记忆指令的符号，反映指令的功能（DB、MOV）
* 操作数：参与操作的对象，具体的常量、保存在寄存器中的数据、保存在存储器中的变量；逗号前目的操作数，逗号后源操作数
* 参数：常量、变量名、表达式等

例：”’Hello, Everybody!’, 0DH, 0AH, ‘$’”

* 常数：表示一个固定数值

1. 十/十六/八/二进制常数
2. 字符串常数：单个多个字符，数值是每个字符对应ASCII的值，‘der’，‘d‘=64h
3. 符号常数：equ和=

doschar equ 2

carriagereturn =13

calldos equ <int 21h>

x = x+5 ;√

x equ x+5 ; X

* 简化段定义

.model small

.stack

.data

…

.code

.startup

…

.exit 0

…

end

例：

.model small

.stack

.data

string db ‘hello’, 0ah, 0dh, ’$’

.code

.startup

mov dx, offset string

mov ah, 9

int 21h

.exit 0

end

* 程序开发过程：--**编辑**--asm--**汇编**--obj--**连接**—exe—**调试**--
* DOS系统中断

|  |  |  |  |
| --- | --- | --- | --- |
| 子功能号 | 功能 | 入口参数 | 出口参数 |
| AH=01H | 输入一个字符 |  | AL=字符ASCII |
| AH=02H | 输出一个字符 | DL=字符ASCII |  |
| AH=09H | 输出字符串 | DS:DX=字符串地址 |  |
| AH=0AH | 输入字符串 | DS:DX=缓冲区地址 |  |
| AH=0BH | 判断是否有按键按下 |  | AL=0，有；  AL=FFH，无 |
| AH=4CH | 程序执行终止 | AL=返回代码 |  |

* DEBUG

-R：观看和修改寄存器的值

-D：显示内存区域的内容

-A：输入汇编指令

-T：执行汇编程序，单步跟踪

-Q：退出DEBUG，回到DOS状态

**第四章**

* 分支循环
* 冒泡
* 空格剔除
* 子程序
* 串操作指令

cld，df=0，高地址移动；std，df=1，低地址移动

源操作数DS[SI]，允许段超越

目的操作数ES[DI]，不允许段超越

SI、DI自动修改

DF=0，地址指针+1/2，DF=1，地址指针-1/2

* 串传送

1. movs：两个串传送

movsb ；字节传送，es:[di]<-ds:[si]，si±1，di±1

movsw ；字，es:[di]<-ds:[si]，si±2，di±2

movs 目的串名，源串名

1. stos：al/ax传给串

stosb ；字节存储，es:[di]<-al， di±1

stosw ；字，es:[di]<-ax， di±2

1. lods：串串给al/ax

lodsb ；字节读取，al<-ds:[si]，si±1

lodsw ；字，ax<-ds:[si]，si±2

1. rep：每执行一次，cx-1直到cx=0结束

* 串检测

1. cmps：比较两个串的关系

cmpsb ；字节比较，ds:[si]-es:[di]，si±1，di±1

cmpsw ；字，ds:[si]-es:[di]，si±2，di±2

1. scas：比较al/ax与字符串两者关系

scasb ；字节扫描，al-es:[di]，di±1

scasw ；字，ax-es:[di]，di±2

1. repe/repz：每执行一次，cx-1，cx=0或zf=0结束
2. repne/repnz：每执行一次，cx-1，cx=0或zf=1结束