

## 第二章 32位PC汇编程序设计环境

- 32位可编程寄存器体系
- 实方式下存储器寻址机制
- 堆栈存储技术
- 程序汇编连接与调试方法



XU Aiping

计算机学院

## 第二章 32位PC汇编程序设计环境

### 2.1 32位可编程寄存器体系

#### 一、通用寄存器

EAX		AH	AL	AX	
EBX		BH	BL	BX	
ECX		CH	CL	CX	
EDX		DH	DL	DX	
ESI		SI			
EDI		DI			
EBP		BP			
ESP		SP			
	31	...	16 15	...	0



XU Aiping

计算机学院

## 2.1 32位可编程寄存器体系

- 累加器**AX**: 存放算术运算的结果, **AL**为字节累加器
- 基址器**BX**: 地址计算时, 用作存放基地址的寄存器
- 计数器**CX**: 某些指令隐含作为计数器
- 数据寄存器**DX**: 端口地址寄存器, 某些输入输出指令用来存放外部设备的**I/O**地址
- 源变址器**SI** / 目的变址器**DI**: 变址寄存器
- 基址指针**BP**: 作为堆栈数据存取操作的基本地址指针寄存器
- 栈指针**SP**: 堆栈指针寄存器, 指示堆栈的当前偏移地址



XU Aiping

计算机学院

例 2.1 已知 AH 的内容为 75H (H 表示 16 进制), AL 的内容为 72H, 则 AX 的内容为 7572H; BX 的内容为 4612H, 则 BH 的内容为 46H, BL 的内容为 12H。求执行传送指令 `MOV BX, AX` 之后 BX、BH、BL、AH、AL 和 AX 的值。

解 指令 `MOV BX, AX` 表示  $AX \rightarrow BX$ , 即  $AX = 7572H \rightarrow BX$ , 且 AX 的内容不变, 因此各项的值为:

$BX = 7572H$ , 即  $BH = 75H, BL = 72H$ ;  $AH = 75H, AL = 72H, AX = 7572H$ 。

例 2.2 已知  $DH = 10H, CL = 48H$ , 求执行加法指令

`ADD CL, DH`

之后 CL 和 DH 的值。

解 指令 `ADD CL, DH` 表示  $CL + DH \rightarrow CL$ , 即  $48H + 10H = 58H \rightarrow CL$ , 且 DH 的内容不变, 因此各项的值为:

$CL = 58H, DH = 10H$ 。

注意: 若  $CL = 47H$ , 在 CH 内容未知的情况下, 不能默认为  $CX = 0047H$ , 应先将 0  $\rightarrow CH$ 。



XU Aiping

计算机学院

**例 2.3** 已知 EAX 的内容为 98765432H, EBX 的内容为 12340011H, 求执行减法指令

`SUB EAX, EBX`


之后 EAX、AX、AL、EBX、BX 和 BH 的值。

**解** 指令 `SUB EAX, EBX` 表示  $EAX - EBX \rightarrow EAX$ , 且 EBX 的内容不变, 即

$$EAX - EBX = 98765432H - 12340011H = 86425421H \rightarrow EAX$$

因此各项的值为: EAX = 86425421H, AX = 5421H, AL = 21H, EBX = 12340011H, BX = 0011H, BH = 00H

注意: 若 EAX 的高 16 位内容未知, 而低 16 位 AX = 5432H, 则不能认定 EAX = 00005432H。其余寄存器的用法类似。


XU Aiping
计算机学院

## 2.1 32位可编程寄存器体系

### 2.1.2 基本控制寄存器

...	I	V	V	A	V	R		N	IO	O	D	I	T	S	Z		A	P	C
...	D	IF	IF	C	M	F		T	PL	F	F	F	F	F	F		F	F	F
31 ... ..	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3 2 1 0



EFLAGS

FLAGS

标志寄存器及其位序号


XU Aiping
计算机学院

## 2.1 32位可编程寄存器体系

(1) 条件标志：反映包含在ALU中算术逻辑运算后的结果特征

- 符号标志SF：结果为负， $SF = 1$ ；否则  $SF = 0$
- 零标志ZF：运算操作结果=0， $ZF = 1$ ，否则  $ZF = 0$
- 辅助进位AF：运算中第三位有进位， $AF = 1$ ，否则 $AF = 0$
- 奇偶标志PF：结果操作数中有偶数个“1”时， $PF=1$ ,否则 $PF=0$
- 进位标志CF：记录运算操作时最高有效位产生的进位值，有进位 $CF = 1$ ，否则 $CF = 0$
- 溢出标志OF：操作数结果超出表示范围， $OF=1$ ,否则 $OF=0$



XU Aiping

计算机学院

➤ 例2.4  $[x_1]_{\text{补}} = 0010\ 0011\ 0100\ 0101$

➤  $[x_2]_{\text{补}} = 0011\ 0010\ 0001\ 1001$

➤  $[x_1]_{\text{补}} + [x_2]_{\text{补}} = 0010\ 0011\ 0100\ 0101$

➤  $+ 0011\ 0010\ 0001\ 1001$

➤  $\hline 0101\ 0101\ 0101\ 1110$

➤ 标志位如下： $SF=0\ ZF=0\ PF=0\ CF=0\ AF=0\ OF=0$



XU Aiping

计算机学院

➤ 例2.4  $[x_1]_{\text{补}} = 0101\ 0100\ 0011\ 1001$

➤  $[x_2]_{\text{补}} = 0100\ 0101\ 0110\ 1010$

➤  $[x_1]_{\text{补}} + [x_2]_{\text{补}} = 0101\ 0100\ 0011\ 1001$

➤  $+ 0100\ 0101\ 0110\ 1010$

➤  $\hline 1001\ 1001\ 1010\ 0011$

➤ 标志位如下: SF=1 ZF=0 PF=1 CF=0 AF=1 OF=1



XU Aiping

计算机学院

➤ 减法CF的置位方法是: 将 $[x_1]_{\text{补}}$ 与 $[x_2]_{\text{补}}$ 均看成无符号数, 如果 $[x_1]_{\text{补}} \geq [x_2]_{\text{补}}$ 则说明够减, 无借位, CF=0; 否则说明不够减, 要借位, CF=1

➤ 如:  $[x_1]_{\text{补}} = 0010\ 0011$   $[x_2]_{\text{补}} = 0011\ 0010$

➤  $[x_1]_{\text{补}} - [x_2]_{\text{补}} = 0010\ 0011$

➤  $+ 1100\ 1110$

➤  $\hline 1111\ 0001$

➤ 标志位如下: SF=1 ZF=0 PF=0 CF=1 ( $[x_1]_{\text{补}} < [x_2]_{\text{补}}$ )

➤ AF=0 ( $[x_1]_{\text{补}}$ 的低4位 >  $[x_2]_{\text{补}}$ 的低4位) OF=0



XU Aiping

计算机学院

## 2.1 32位可编程寄存器体系

### (2) 控制标志

- 方向标志DF: DF=0, 处理从低位地址开始, DF=1, 处理从高位地址开始
- 中断允许标志IF: IF=1, CPU允许中断, 否则关闭中断
- 跟踪标志TF: TF=1, 机器进入单步工作方式, TF=0, 机器处于连续工作方式



XU Aiping

计算机学院

## 2.1 32位可编程寄存器体系



指令指针IP/EIP指出程序执行过程中当前要取出的下条指令的地址。当取出一条指令后, IP/EIP自动加上该指令的长度或者形成转移地址, 又指向下一条指令的地址, 从而可以控制有序的执行程序。



XU Aiping

计算机学院

## 2.1 32位可编程寄存器体系

15 ... 0

CS
DS
SS
ES
FS
GS

段寄存器

➤代码段寄存器**CS**：存放当前正在执行代码的段的起始地址

➤数据段寄存器**DS**：存放当前正在执行程序所用数据的段的起始地址

➤堆栈段寄存器**SS**：存放当前正在执行程序暂时保留信息的段的起始地址

➤附加段寄存器**ES**、**FS**和**GS**：存放程序的数据段的起始地址，为程序设计使用多个数据段带来方便

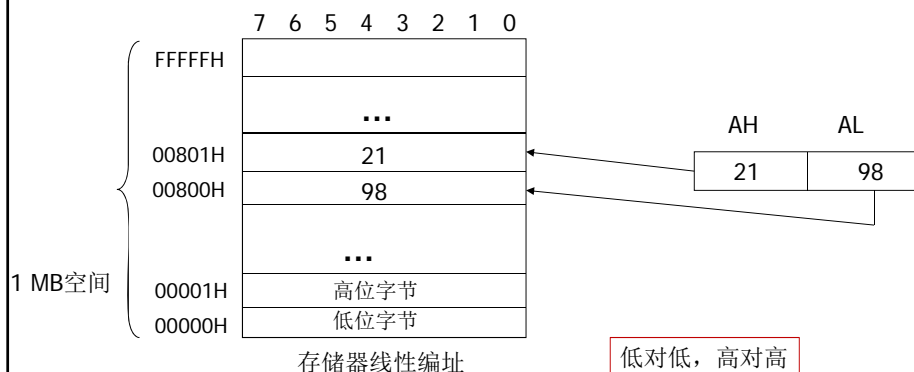


XU Aiping

计算机学院

## 2.2 实地址下的存储器寻址方式

### 一、存储单元的地址和内容



XU Aiping

计算机学院

- 4. 存储单元的内容
- 存储单元的内容是指该单元存放的二进制数据(常用16进制书写)。在程序设计过程中,有时这些数据还可以是一存储单元的地址。
- 设字单元的地址为100H, 100H地址的内容是1869H, 表示为(100H)=1869H, 则字单元包含2个连续字节单元的内容是:  
(100H)=69H. (101H)=18H。
- 设一个存储单元的地址为M, M单元的内容表示为(M), 设M单元中存放N, 而N又是一个单元的地址, 则N单元的内容可表示为: (N)=((M))。
- 其中, ((M))表示以(M)为地址的单元的内容。
- 例2. 5 若M=0CA0H, N=0BF9H, (M)=N. (0BF9H)=2630H, 求((M))=?
- 因为: (N)=(0BF9H)=2630H, 又由((M))=(N)的表示可知:
- ((M))=((0CA0H))=(N)=(0BF9H),所以: ((M))=2630H



XU Aiping

计算机学院

## 2.2 实地址下的存储器寻址方式

### 2.2.2 存储器分段寻址

- 分段寻址允许一个程序可以使用多个代码段、数据段和堆栈段
- 存储段内的每个单元的物理地址(PA), 可以用:
- “段地址: 偏移地址”来表达;
- 段地址是指段寄存器中的地址。
- 段内偏移地址: 是该单元的物理地址到段地址的相对距离。
- $PA = \text{段寄存器的内容} \times 2^4 + EA$



XU Aiping

计算机学院



- 由于实方式下段寄存器只有16位，EA由寄存器Bx、BP、SI或DI的内容来计算，也只有16位，因此用16位来获得物理地址PA需要的20位地址(1 MB空间)，显然是不可能的。为了解决用16位寄存器寻址存储单元地址为20位的矛盾，将EA作为PA的低16位，段寄存器的内容作为PA的高16位，当它们相加形成20位的结果时，就得到20位的物理地址。即程序的段地址SA是加载到16位段寄存器的内容左移4位后的地址：
- $SA = \text{段寄存器的内容} \times 2^4$
- $PA = SA + EA = \text{段寄存器的内容} \times 2^4 + EA$
- 因而20位的段地址用5位16进制数来表示时，其最末位为0。

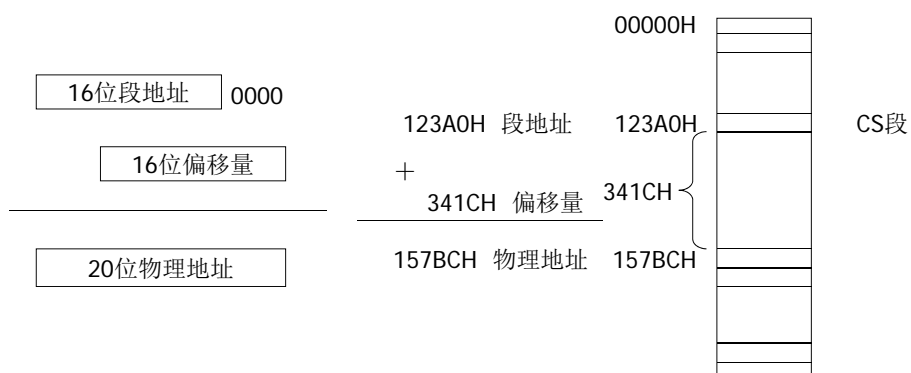


XU Aiping

计算机学院

## 2.2 实地址下的存储器寻址方式

- 物理地址的计算与形成



XU Aiping

计算机学院

- 例2.6 设数据1000H在存储器的数据段，段地址80480H的前4位在DS中，该数据所在的存储单元地址PA到段地址的距离为EA=0602H。求PA=?  
(PA)=?
- $2^4=16=10H$ .  $SA=DS \times 2^4=8048H \times 10H$ :  
80480H
- $PA=SA+EA=80480H+0602H=80A82H$
- $(PA)=(80A82H)=1000H$



XU Aiping

计算机学院

## 举例

- 1. 写出下列逻辑地址的段地址、偏移地址和物理地址
- (1) 2314H: 0024H    (2) 1FD0: 001A
- $SA=23140H$      $EA=0024H$      $PA=23140+0024=$
- 2. 若物理地址为34567H，可以采用的逻辑地址有\_\_?\_\_H:  
4567H、3450H: \_\_?\_\_H等。
- 解:  $\times\times\times\times 0H + 4567H = 34567H$      $\times\times\times\times=3000H$ ;
- $34500H + \times\times\times\times = 34567H$      $\times\times\times\times=0067H$ ;
- 可见对应唯一的物理地址，其逻辑地址不是唯一的。



XU Aiping

计算机学院

- 在传统方式或实方式的存储管理下，每一个存储段(段区)的大小最大可达**64 KB**，最小有**16B**。初始段地址一般是能被**16**除尽的地址，或称节地址(段边界地址)。
- 各段在存储器中的分配是由操作系统负责的。每个段可以独立占用**64 KB**的存储区，如下图所示。

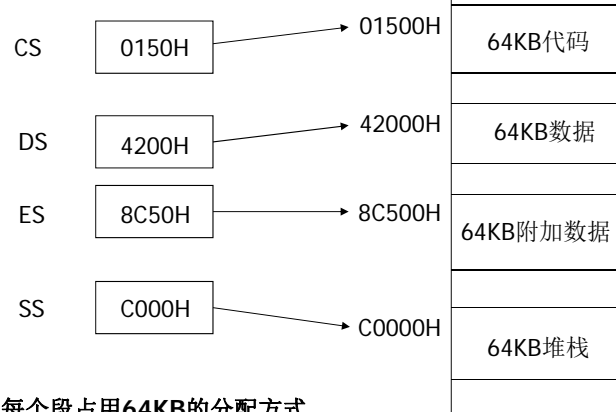


XU Aiping

计算机学院

## 2.2 实地址下的存储器寻址方式

### 三、段的分配方式

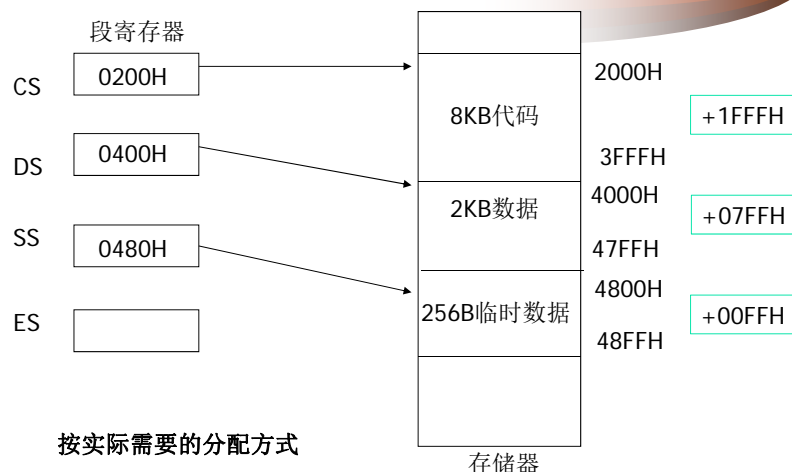


XU Aiping

计算机学院

- 实际上，一般根据程序或数据的大小来占用存储区。设程序代码空间有**8 KB**，数据有**2 KB**，临时存放信息的堆栈有**256 B**，此时段的分配方式如下图所示。

## 2.2 实地址下的存储器寻址方式



因此，用户根据需要将程序分成若干个逻辑段，每个逻辑段由操作系统在执行时动态分配并安排各个段在物理器上对应存储段。

## 2.3 堆栈存储技术

### 2.3.1 堆栈概念

- 堆栈是由程序在**RAM**中开辟的一片内存区域，具有主存储器的功能和特性；
- 设置堆栈段寄存器**SS**，用来指示堆栈起始位置的指针；  
栈顶寄存器**SP**用来指示堆栈顶部到堆栈起始位置的距离，即栈顶指针；
- 栈顶是一个动态的概念，用**SP**来记录堆栈操作时变化的指针，它动态地指向当前可以压入信息到堆栈中的偏移地址，或者从堆栈中当前需要弹出信息的偏移地址；
- 堆栈单元的物理地址是： $PA = SS \times 2^4 + SP$



XU Aiping

计算机学院

### 2.3.2 堆栈操作原则

- “后进先出”或“先进后出”
- 下推式：堆栈是向下生长的，从堆栈的高地址先压入内容，再从相邻的低地址压入内容。
- 堆栈操作指令（**PUSH**、**POP**）均是对**16位**（**2**个字节）的数据进行操作。



XU Aiping

计算机学院

➤ 3. 堆栈对字操作

- 压入操作PUSH或弹出操作POP均是对16位(2个字节)的数据进行操作。  
PUSH和POP的操作如下(分号；之后表示注释):

➤ **PUSH源操作数** ；将源操作数压入堆栈

➤ 执行操作: ①  $SP-2 \rightarrow SP$  ；先修改SP, 即将SP指针减2送SP。

➤ ② 源操作数  $\rightarrow [SP]$  ；后将源操作数送入SP表示的单元,  
其中[SP]表示SP的内容作为地址。

➤ 例2. 7 设  $AX=7809H$ ,  $SP=0100H$ , 执行PUSH AX指令后,  $SP-2$ ,  $([SP])=?$

➤  $SP-2=0100H-2=00FEH \rightarrow SP$      $AX=7809H \rightarrow [SP]$ ,

➤ 则  $([SP])=(00FEH)=7809H$

➤



XU Aiping

计算机学院

➤ **POP 目的操作数** ；将当前SP指向堆栈单元的内容弹出到目的操作数

➤ 执行操作:

➤ ①  $([SP]) \rightarrow$  目的操作数; 先将SP单元的内容送入目的操作数。

➤ ②  $SP+2 \rightarrow SP$  ；后修改SP, 即SP指针加2送SP。

➤ 例 2. 8 设  $SP=00FCH$ ,  $(00FCH)=4613H$ , 执行POP BX指令后,  
 $BX=?$ ,  $SP=?$

➤  $([SP])=(00FCH)=4613H \rightarrow BX$ .     $SP+2=00FCH+2=00FEH \rightarrow SP$

➤ 所以:  $BX=4613H$      $SP=00FEH$



XU Aiping

计算机学院

### 2.3.3 堆栈操作示例

- 设AX=4130H,BX=2010H,堆栈的初始栈顶SP=100H,分析执行下列指令序列后, SP=?

PUSH AX

PUSH BX

若再执行 POP BX

POP CX 则 CX=? SP=?

分析过程如下图所示,显然执行 PUSH BX 后, SP = 0FCH

执行 POP CX 后,

CX = 4130H SP = 100H



XU Aiping

计算机学院

- 习题: 2.2 2.3 2.4 2.6 2.10



XU Aiping

计算机学院