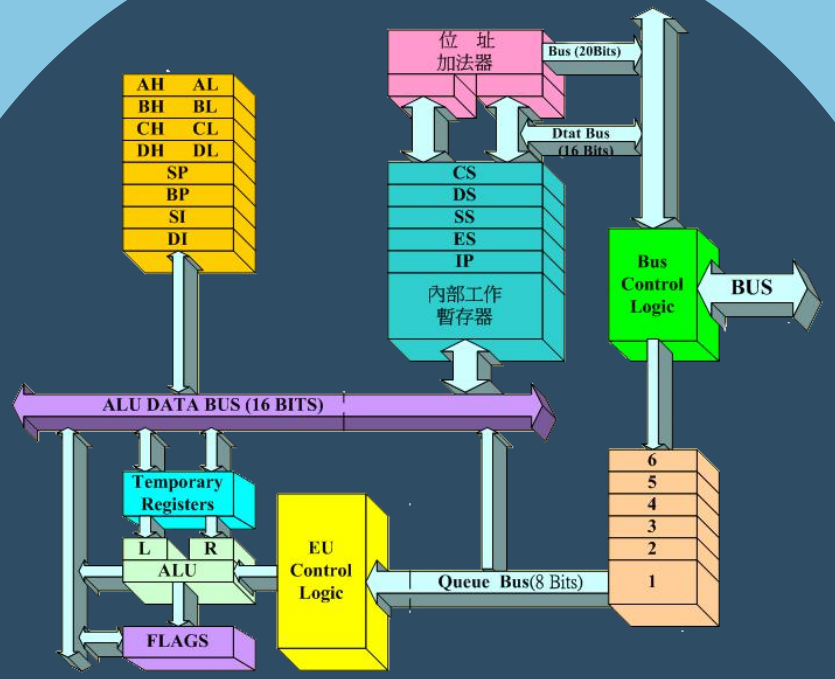
DS与数据段

贺利坚　主讲



汇编语言程序设计

Assembly Language

对内存单元中数据的访问

- 对于8086PC机，可以根据需要将一组内存单元定义为一个段。
  - 物理地址=段地址×16+偏移地址
  - 将一组长度为N（N≤64K）、地址连续、起始地址为16的倍数的内存单元当作专门存储数据的内存空间，从而定义了一个数据段。
- 例：用123B0H~123B9H的空间来存放数据
  - 段地址：123BH　起始偏移地址：0000H　长度：10字节
  - 段地址：1230H　起始偏移地址：00B0H　长度：10字节
  - ....

将哪段内存当作数据段，段地址如何定，在编程时安排。

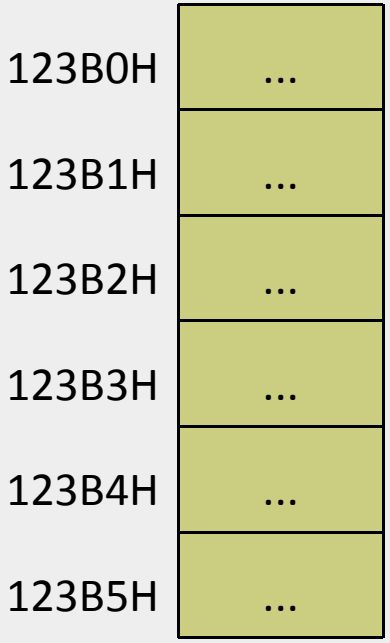
- 处理方法：(DS):([address])
  - 用DS存放数据段的段地址
  - 用相关指令访问数据段中的具体单元，单元地址由[address]指出

mov、add、sub...

将123B0H~123BAH的内存单元定义为数据段

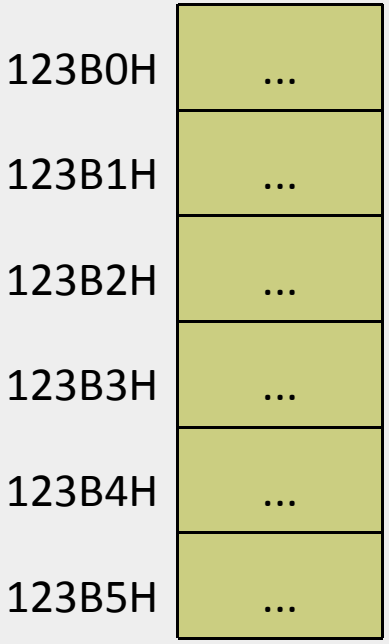
- 累加数据段中的前3个单元中的数据

```
mov ax, 123BH
mov ds, ax
mov al, 0
add al, [0]
add al, [1]
add al, [2]
```



- 累加数据段中的前3个**字型数据**

```
mov ax, 123BH
mov ds, ax
mov ax, 0
add ax, [0]
add ax, [2]
add ax, [4]
```



练习

预设数据

```
-E 0:0 70 80 F0 30 EF 60 30 E2 00 80 12 66 20 22 60
-E 0:10 62 26 E6 D6 CC 2E 3C 3B AB BA 00 00 26 06 66 68
-D 0:0 1F
0000:0000  70 80 F0 30 EF 60 30 E2-00 80 12 66 20 22 60 60   p..0.`0....f "``
0000:0010  62 26 E6 D6 CC 2E 3C 3B-AB BA 00 00 26 06 66 68   b&....<;....&.fh
```

预设代码

```
-A 073F:0100
073F:0100 mov ax, 1
073F:0103 mov ds, ax
073F:0105 mov ax, [0000]
073F:0108 mov bx, [0001]
073F:010C mov ax, bx
073F:010E mov ax, [0000]
073F:0111 mov bx, [0002]
073F:0115 add ax, bx
073F:0117 add ax, [0004]
073F:011B mov ax, 0
073F:011E mov al, [0002]
073F:0121 mov bx, 0
073F:0124 mov bl, [000C]
073F:0128 add al, bl
073F:012A
```

寄存器值

```
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0000  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 B80100        MOV     AX,0001
```

提示：可以通过“R寄存器”命令修改，关键是CS和IP

执行代码

```
-t

AX=0001  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=0000  ES=073F  SS=073F  CS=073F  IP=0103   NV UP EI PL NZ NA PO NC
073F:0103 8ED8          MOV     DS,AX
-t_
```

给出00000H-0001F的数据，请写出下面代码的执行结果：

代码	AX	BX
mov ax,[0000]		
mov bx,[0001]		
mov ax,bx		
mov ax,[0000]		
mov bx,[0002]		
add ax, bx		
add ax,[0004]		
mov ax,0		
mov al,[0002]		
mov bx, 0		
mov bl, [000C]		
add al,bl		

①“人脑”计算；②“电脑”验证

用mov指令操作数据

指令形式	例示
mov 寄存器，数据	mov ax, 8
mov 寄存器，寄存器	mov ax, bx
mov 寄存器，内存单元	mov ax, [0]
mov 内存单元，寄存器	mov [0], ax
mov 段寄存器，寄存器	mov ds, ax

- 已知：mov 段寄存器，寄存器

  推测1➔ mov 寄存器，段寄存器

- 已知：mov 内存单元，寄存器

  推测2➔ mov 内存单元，段寄存器

  推测3➔ mov 段寄存器，内存单元

- 已知：mov 寄存器，数据

  推测4➔ mov 段寄存器，数据

大胆地假设，小心地求证。

验证1：

```
-a 073f:100
073F:0100 mov ax, ds
073F:0102
-r
AX=0000  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0100   NV UP EI PL NZ NA PO NC
073F:0100 8CD8          MOV     AX,DS
-t

AX=073F  BX=0000  CX=0000  DX=0000  SP=00FD  BP=0000  SI=0000  DI=0000
DS=073F  ES=073F  SS=073F  CS=073F  IP=0102   NV UP EI PL NZ NA PO NC
073F:0102 0000          ADD     [BX+SI],AL                   DS:0000=CD
-
```

验证2：

mov ax, 1000H

mov ds, ax

mov [0], ds

验证3：

mov ax, 1000H

mov ds, ax

mov ds, [0]

验证4：

mov ds, 8

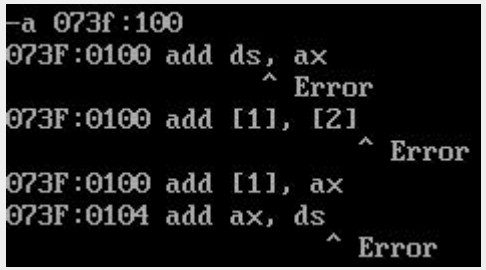
```
-a 073f:100
073F:0100 mov ds, 8
                  ^ Error
```

加法add和减法sub指令

add指令形式	例示
add 寄存器，数据	add ax, 8
add 寄存器，寄存器	add ax, bx
add 寄存器，内存单元	add ax, [0]
add 内存单元，寄存器	add [0], ax

sub指令形式	例示
sub 寄存器，数据	sub ax, 8
sub 寄存器，寄存器	sub ax, bx
sub 寄存器，内存单元	sub ax, [0]
sub 内存单元，寄存器	sub [0], ax

- 推测1➔ add 段寄存器，寄存器
- 推测2➔ add 内存单元，内存单元
- 推测...➔



用DS和[address]形式访问内存中数据段方法小结

💻指令

```
mov ax, 1000H
mov ds, ax
mov ax, 11316
mov [0], ax
mov bx, [0]
sub bx, [2]
mov [2], bx
```

💻结合体验品味

（1）字在内存中存储时 ，要用两个地址连续的内存单元来存放，字的低位字节存放在低地址单元中，高位字节存放再高地址单元中。

（2）用 mov 指令要访问内存单元，可以在mov指令中只给出单元的偏移地址，此时，段地址默认在DS寄存器中。

（3）[address]表示一个偏移地址为address的内存单元。

（4）在内存和寄存器之间传送字型数据时，高地址单元和高8位寄存器、低地址单元和低8位寄存器相对应。

（5）mov、add、sub是具有两个操作对象的指令，访问内存中的数据段（对照：jmp是具有一个操作对象的指令，对应内存中的代码段）。

（6）可以根据自己的推测，在Debug中实验指令的新格式。