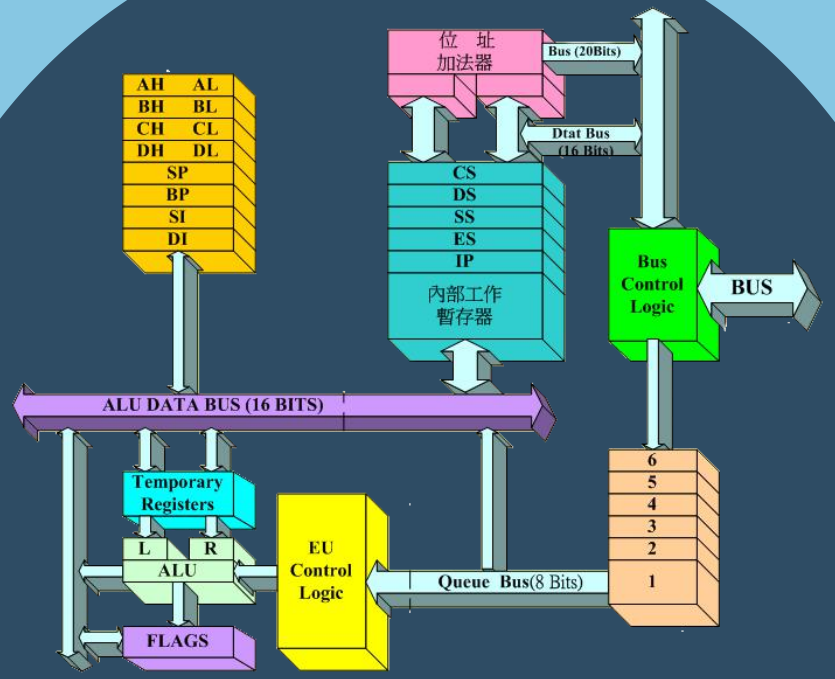
不同的寻址方式的灵活应用

贺利坚　主讲



汇编语言程序设计

Assembly Language

对内存的寻址方式

形式	名称	特点	意义	示例
[idata]	直接寻址	用一个常量/立即数来表示地址	用于直接定位一个内存单元	mov ax, [200]
[bx]	寄存器间接寻址	用一个变量来表示内存地址	用于间接定位一个内存单元	mov bx, 0 mov ax, [bx]
[bx+idata]	寄存器相对寻址	用一个变量和常量表示地址	可在一个起始地址的基础上用变量间接定位一个内存单元	mov bx, 4 mov ax, [bx+200]
[bx+si]	基址变址寻址	用两个变量表示地址		mov ax, [bx+si]
[bx+si+idata]	相对基址变址寻址	用两个变量和一个常量表示地址		mov ax, [bx+si+200]

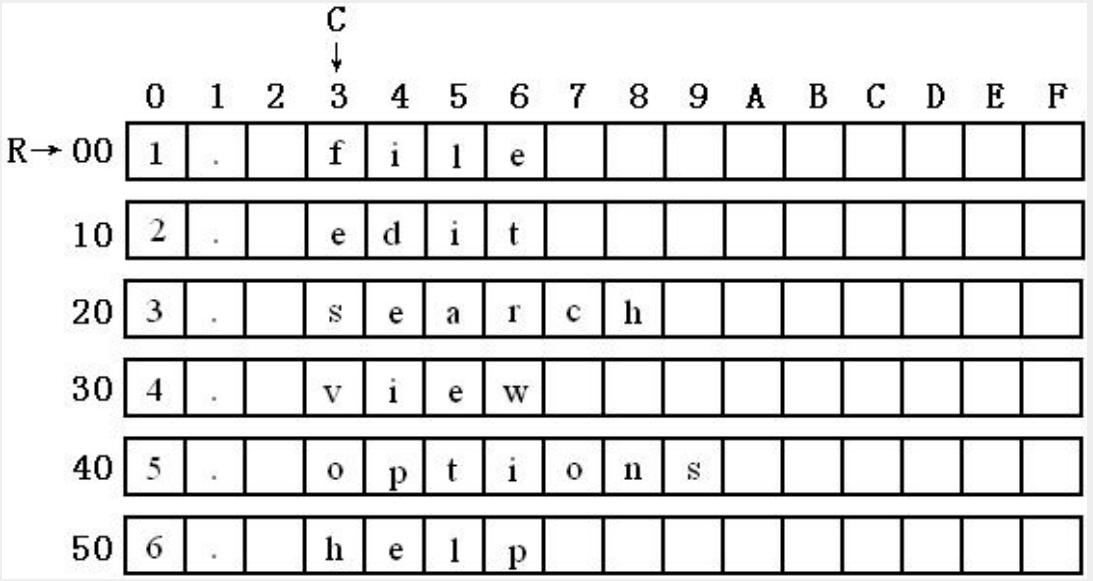
案例1：灵活应用不同的寻址方式

💻问题：编程将datasg段中每个单词的头一个字母改为大写字母。

```
assume cs:codesg,ds:datasg
datasg segment
      db '1. file          '
      db '2. edit          '
      db '3. search        '
      db '4. view          '
      db '5. options       '
      db '6. help          '
datasg ends

codesg segment
 start: ....
     mov 4c00h
     int 21h
codesg ends
end start
```

[bx+idata]方式



datasg中的数据的存储结构

```
    R=第一行的地址
    mov cx,6
s:  改变R行3列的字母为大写
    R=下一行的地址
    loop s
```

```
    mov ax,datasg
    mov ds,ax

    mov bx,0
    mov cx,6
s:  mov al,[bx+3]
    and al,11011111b
    mov [bx+3],al
    add bx,16
    loop s
```

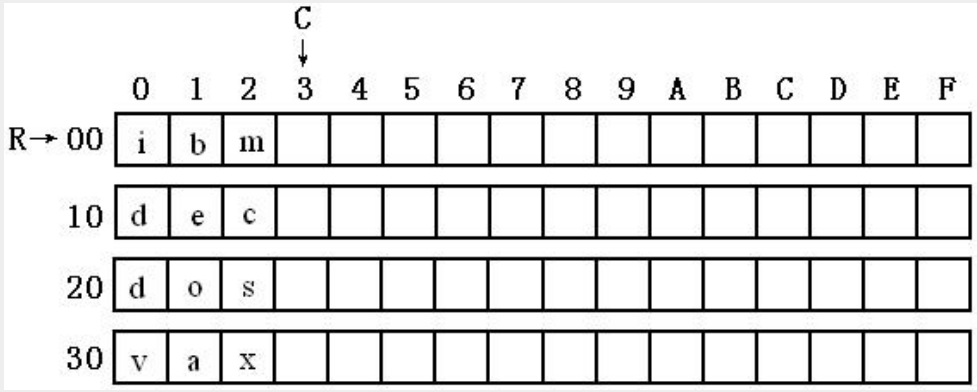
案例2：灵活应用不同的寻址方式

- 问题：编程将datasg段中每个单词改为大写字母。

- 4 个字符串，看成一个 4行16列的二维数组
- 要修改二维数组的每一行的前3列
- 构造4x3次的二重循环

```
assume cs:codesg,ds:datasg
datasg segment
  db 'ibm             '
  db 'dec             '
  db 'dos             '
  db 'vax             '
datasg ends

codesg segment
 start: ......
codesg ends
end start
```

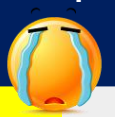


[bx+si] 方式

```
    R=第一行的地址；
    mov cx,4
s0: C=第一列的地址
    mov cx,3
 s: 改变R 行，C列字母为大写
    C=下一列的地址；
    loop s
    R=下一行的地址
    loop s0
```

```
    mov ax,datasg
    mov ds,ax
    mov bx,0
    mov cx,4
s0: mov si,0
    mov cx,3
 s: mov al,[bx+si]
    and al,11011111b
    mov [bx+si],al
    inc si
    loop s
    add bx,16
    loop s0
```

循环次数由cx定，可是，cx只有一个哇！



二重循环问题的处理-法1

问题：编程将datasg段中每个单词改为大写字母。

```
assume cs:codesg,ds:datasg
datasg segment
  db 'ibm          '
  db 'dec          '
  db 'dos          '
  db 'vax          '
datasg ends

codesg segment
 start: ......
codesg ends
end start
```

```
    ; 有缺陷的程序
    mov ax,datasg
    mov ds,ax
    mov bx,0
    mov cx,4
s0: mov si,0
    mov cx,3
 s: mov al,[bx+si]
    and al,11011111b
    mov [bx+si],al
    inc si
    loop s
    add bx,16
    loop s0
```

dx已经被用了呢？别的寄存器也有用处了呢？寄存器只有14个！

```
    mov ax,datasg
    mov ds,ax
    mov bx,0
    mov cx,4
s0: mov dx,cx
    mov si,0
    mov cx,3
 s: mov al,[bx+si]
    and al,11011111b
    mov [bx+si],al
    inc si
    loop s
    add bx,16
    mov cx,dx
    loop s0
```

将外层循环的cx值保存在dx中

cx设置为内存循环的次数

用dx中存放的外层循环的计数值恢复cx

(cx)=(cx)-1针对外层循环

方法1：用dx保存数据

二重循环问题的处理-法2、法3

方法2：用固定的内存空间保存数据

```
    mov ax,datasg
    mov ds,ax
    mov bx,0
    mov cx,4
s0: mov ds:[40H],cx
    mov si,0
    mov cx,3
 s: mov al,[bx+si]
    and al,11011111b
    mov [bx+si],al
    inc si
    loop s
    add bx,16
    mov cx,ds:[40H]
    loop s0
```

将外层循环的cx值保存在datasg:40H单元中

cx设置为内存循环的次数

用datasg:40H单元中的值恢复cx

方法3：用栈保存数据

```
stacksg segment
    dw 0,0,0,0,0,0,0,0
stacksg ends
```

```
    mov ax,stacksg
    mov ss,ax
    mov sp,16
    mov ax,datasg
    mov ds,ax
    mov bx,0
    mov cx,4
s0: push cx
    mov si,0
    mov cx,3
 s: mov al,[bx+si]
    and al,11011111b
    mov [bx+si],al
    inc si
    loop s
    add bx,16
    pop cx
    loop s0
```

将外层循环的cx值压栈

cx设置为内存循环的次数

从栈顶弹出原cx的值，恢复cx