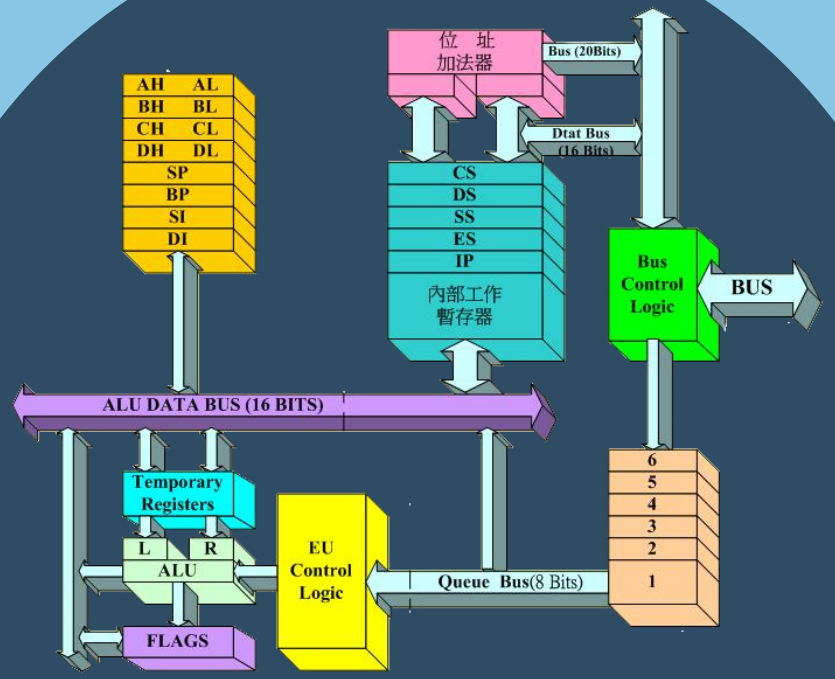
定制键盘输入处理

贺利坚　主讲



汇编语言程序设计

Assembly Language

PC机键盘的处理过程(int 9 中断例程)

- 键盘输入的处理过程
  - （1）键盘产生扫描码；
  - （2）扫描码送入60h 端口；
  - （3）引发9 号中断；

  } 由硬件系统完成的

  - （4）CPU执行int 9中断例程，处理键盘输入
    - DOS系统提供int 9中断例程
    - 按照开发需求定制处理键盘的输入
- 编程任务
  - 在屏幕中间依次显示 'a'~'z' ，并可以让人看清。
  - 在显示的过程中，按下Esc键后，改变显示的颜色。
- 工作策略
  - 尽可能忽略硬件处理细节，充分利用BIOS 提供的int 9中断例程对这些硬件细节进行处理。
  - 在改写后的中断例程中满足特定要求，并能调用BIOS 的原int 9中断例程。

实现：依次显示'a'~'z'(v0.2)

```
assume cs:code
code segment
start:  mov ax,0b800h
        mov es,ax
        mov ah,'a'
     s: mov es:[160*12+40*2],ah
        inc ah
        cmp ah,'z'
        jna s
        mov ax,4c00h
        int 21h
code ends
end start
```

- 存在的问题：无法看清屏幕上的显示
- 原因：同一位置显示字母，字母之间切换得太快，无法看清。
- 对策：在每显示一个字母后，延时一段时间
- 如何延时？让CPU 执行一段时间的空循环

```
    mov dx,10h
    mov ax,0
s: sub ax,1
    sbb dx,0
    cmp ax,0
    jne s
    cmp dx,0
    jne s
```

- 循环100000h 次！CPU 速度太快了，用两个16 位寄存器来存放32 位的循环次数。

实现：依次显示”a”~”z”(v0.4)

```
assume cs:code
stack segment
        db 128 dup (0)
stack ends

code segment
start:  mov ax,stack
        mov ss,ax
        mov sp,128

    ; 显示字符

        mov ax,4c00h
        int 21h

    ; 定义延时函数

code ends
end start
```

```
        mov ax,0b800h
        mov es,ax
        mov ah,'a'
s:      mov es:[160*12+40*2],ah
        call delay
        inc ah
        cmp ah,'z'
        jna s
```

```
delay:  push ax
        push dx
        mov dx,10h
        mov ax,0
  s1:   sub ax,1
        sbb dx,0
        cmp ax,0
        jne s1
        cmp dx,0
        jne s1
        pop dx
        pop ax
        ret
```

- 接下来的工作：按下 Esc 键后，改变显示的颜色！
- 原理：键盘输入到达60h端口后，就会引发 9号中断，CPU 则转去执行 int 9中断例程。

按下 Esc 键后改变显示的颜色

编写int 9中断例程改变显示的颜色

（1）从60h 端口读出键盘的输入；

```
in al,60h
```

（2）调用BIOS 的int 9 中断例程，处理硬件细节；

（3）判断是否为Esc的扫描码，如果是，改 变显示的颜色后返回；如果不是则直接返回。

②如何调用原int 9指令的中断例程

- int 9已改，但仍然需要调用原int 9指令功能
- 解决方法：模拟对原中断例程的调用

（1）标志寄存器入栈

```
pushf
```

（2）IF=0，TF=0

```
pushf
pop ax
and ah,11111100b
push ax
popf
```

（3）CS、IP入栈

（4）(IP)=((ds)*16+0)

(CS)=((ds)*16+2)

```
call dword ptr ds:[0]
```

①关于中断处理程序入口地址面对的问题

- 要将中断向量表中的int 9中断例程的入口地址改为自编的中断处理程序的入口地址。
- 在新中断处理程序中调用原来的int 9中断例程，还需要是原来的int 9 中断例程的地址。
- 解决方法：保存原中断例程入口地址
- 将原来int 9中断例程的偏移地址和段地址保存在ds:[0]和ds:[2]单元中，在需要调用原来的int 9中断例程时候，到 ds:[0]、ds:[2] 找到

```
mov ax,0
mov es,ax
push es:[9*4]
pop ds:[0]
push es:[9*4+2]
pop ds:[2]

mov word ptr es:[9*4],offset int9
mov es:[9*4+2],cs
```

保存旧中断例程入口

设置新中断例程入口

实现: 按下 Esc 键后改变显示的颜色(v1.0)

```
assume cs:code

stack segment
        db 128 dup (0)
stack ends

data segment
        dw 0,0
data ends

code segment
; 代码段
code ends
end start
```

```
push ds:[0]
pop es:[9*4]
push ds:[2]
pop es:[9*4+2]
```

```
start:  mov ax,stack
        mov ss,ax
        mov sp,128
        mov ax,data
        mov ds,ax

        ; 改中断例程入口地址

        ; 显示'a'~'z'

        ; 恢复原来的地址

        mov ax,4c00h
        int 21h

        ; 定义延迟程序 (略)

        ; 定义中断例程
```

```
mov ax,0
mov es,ax
push es:[9*4]
pop ds:[0]
push es:[9*4+2]
pop ds:[2]
mov word ptr es:[9*4],offset int9
mov es:[9*4+2],cs
```

```
        mov ax,0b800h
        mov es,ax
        mov ah,'a'
s:      mov  es:[160*12+40*2],ah
        call delay
        inc ah
        cmp ah,'z'
        jna s
        mov ax,0
        mov es,ax
```

```
int9:   push ax
        push bx
        push es
        in al,60h
        pushf
        pushf
        pop bx
        and bh,11111100b
        push bx
        popf
        call dword ptr ds:[0]

        cmp al,1 ; ESC扫描码1
        jne int9ret
        ;改变颜色
        mov ax,0b800h
        mov es,ax
        inc byte ptr es:[...]

int9ret:pop es
        pop bx
        pop ax
        iret
```

160*12+40*2+1