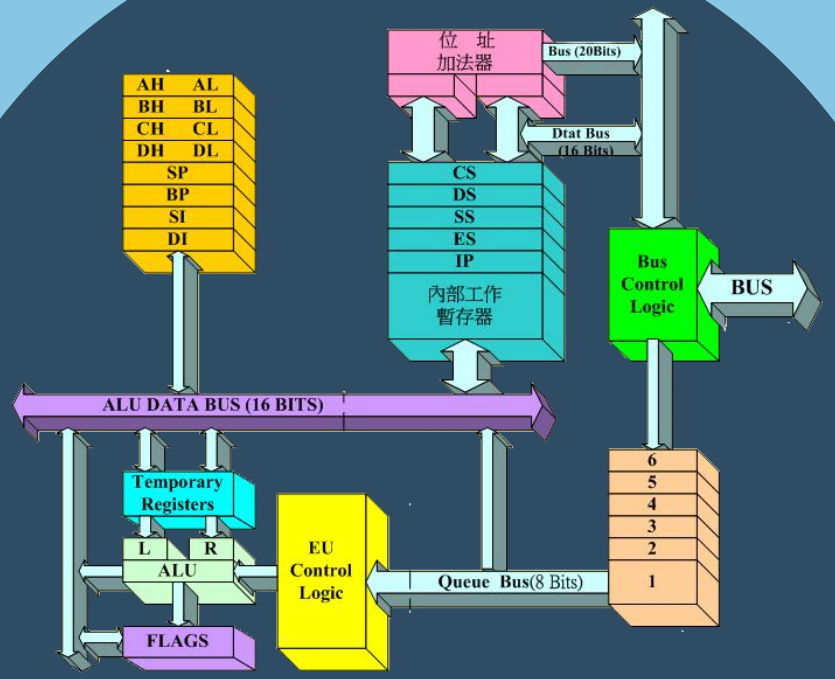
寄存器冲突问题

贺利坚　主讲



汇编语言程序设计

Assembly Language

引子

问题：编程将data段中的字符串转化为大写。

```
assume cs:code
data segment
    db 'conversation'
data ends
code segment
  start: ……
code ends
end start
```

db ‘conversation’,0

```
        mov ax,data
        mov ds,ax
        mov si,0
        mov cx,12
        call capital
        mov ax,4c00h
        int 21h
capital: and byte ptr [si],11011111b
        inc si
        loop capital
        ret
```

代码：编程将data段中的字符串转化为大写

```
assume cs:code
data segment
        db 'conversation',0
 data ends
code segment
start:  mov ax,data
        mov ds,ax
        mov si,0
        call capital
        mov ax,4c00h
        int 21h
capital: mov cl, [si]
           mov ch, 0
           jcxz ok
           and byte ptr [si], 11011111b
           inc si
           jmp short capital
      ok:  ret
code ends
end start
```

;设置字符串的起始地址，并调用子程序

实用性更好的子程序。

子程序依次读取每个字符进行检测，如果不是0，进行大写的转化，如果是0，结束处理。
——不再需要字符串的长度作为参数。

cx既用于循环，又用于读取数据——冲突！

```
; 再例将以下字符串转为大写
assume cs:code
data segment
        db 'word',0
        db 'unix',0
        db 'wind',0
        db 'good',0
data ends
```

```
code segment
start:  mov ax,data
        mov ds,ax
        mov bx,0
        mov cx,4
      s: mov si,bx
        call capital
        add bx,5
        loop s
        mov ax,4c00h
        int 21h
capital: mov cl,[si]
         mov ch,0
         jcxz ok
         and byte ptr [si],11011111b
         inc si
         jmp short capital
     ok: ret
code ends
end start
```

寄存器冲突问题的解决

- 两个可能方案
  - （1）在编写调用子程序的程序时 ，注意看看子程序中有没有用到会产生冲突的寄存器
    - 如果有，调用者使用别的寄存器；
  - （2）在编写子程序的时候，不要使用会产生冲突的寄存器。
- 我们希望
  - （1）编写调用了程序的程序的时候不必关心子程序到底使用了哪些寄存器；
  - （2）编写子程序的时候不必关心调用者使用了哪些寄存器；
  - （3）不会发生寄存器冲突。

调用子程序的程序会很麻烦，必须要小心检查所调用的子程序中是否有将产生冲突的寄存器。

要调用子程序，必须看到子程序源码！？

子程序应该是独立的，编写子程序的时候无法知道也不必知道将来的调用情况。

子程序标准框架：

子程序开始：子程序中使用的寄存器入栈

子程序内容

子程序使用的寄存器出栈

返回（ret、retf）

可行的解决方案：在子程序的开始，将要用到的所有寄存器中的内容都保存起来，在子程序返回前再恢复。

寄存器冲突问题的解决示例

子程序标准框架：

子程序开始：子程序中使用的寄存器入栈

子程序内容

子程序使用的寄存器出栈

返回（ret、retf）

```
capital: push cx
         push si
change: mov cl,[si]
         mov ch,0
         jcxz ok
        and byte ptr [si],11011111b
        inc si
        jmp short change
    ok: pop si
        pop cx
        ret
```

```
assume cs:code
data segment
        db 'word',0
        db 'unix',0
        db 'wind',0
        db 'good',0
data ends

code segment
  start: mov ax,data
         mov ds,ax
         mov bx,0
         mov cx,4
       s: mov si,bx
         call capital
         add bx,5
         loop s
         mov ax,4c00h
         int 21h
         ; 子程序
code ends
end start
```