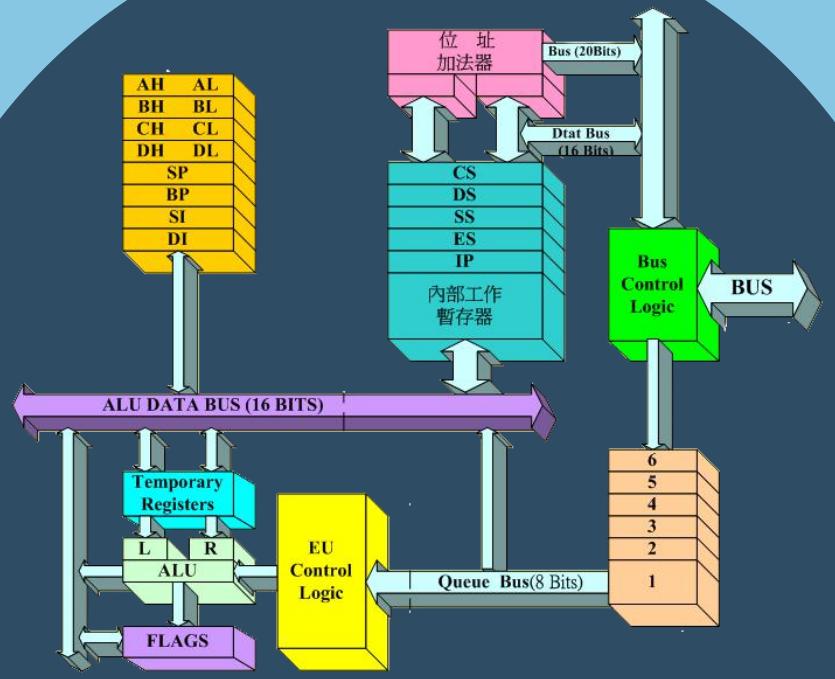
条件转移指令应用

贺利坚　主讲



汇编语言程序设计

Assembly Language

条件转移指令

- 条件转移指令：jxxx——je/jna/jae...
  - 可以根据某种“条件”，决定是否“转移”程序执行流程。
  - “转移” = 修改IP
- 如何检测条件？
  - 通过检测标志位，由标志位体现条件
  - 条件转移指令通常都和cmp相配合使用，cmp指令改变标志位
- 例：双分支结构的实现


```
    cmp ah,bh
    je s
    add ah,bh
    jmp short ok
 s: add ah,ah
ok: ret
```

```
if(a==b){
  a=a+a;
}
else{
  a=a+b;
}
```

应用示例

给出下面一组数据：

```
data segment
  db 8,11,8,1,8,5,63,38
data ends
```

请编程实现如下统计，用ax保存统计结果

（1）统计数值为8的字节的个数

（2）统计数值大于8的字节的个数

（3）统计数值小于8的字节的个数

（1）编程思路：初始设置(ax)=0，然后用循环依次比较每个字节的值，找到一个和8相等的数就将ax的值加1。

解法1

```
5  code segment
6  start:
7       mov ax,data
8       mov ds,ax
9       mov bx,0
10      mov ax,0
11      mov cx,8
12  s:  cmp byte ptr [bx],8
13      jne next
14      inc ax
15  next:
16      inc bx
17      loop s
18
19      mov ax,4c00h
20      int 21h
21 code ends
```

解法2

```
5  code segment
6  start:
7       mov ax,data
8       mov ds,ax
9       mov bx,0
10      mov ax,0
11      mov cx,8
12    s: cmp byte ptr [bx],8
13      je ok
14      jmp short next
15    ok: inc ax
16  next: inc bx
17      loop s
18
19     mov ax,4c00h
20     int 21h
21 code ends
```

应用示例

给出下面一组数据：

```
data segment
  db 8,11,8,1,8,5,63,38
data ends
```

请编程实现如下统计，用ax保存统计结果

（1）统计数值为8的字节的个数

（2）统计数值大于8的字节的个数

（3）统计数值小于8的字节的个数

（2）初始设置(ax)=0，然后用循环依次比较每个字节的值，找到一个大于8的数就将ax的值加1。

```
code segment
start:
        mov ax,data
        mov ds,ax
        mov bx,0
        mov ax,0
        mov cx,8
    s:  cmp byte ptr [bx],8
        jna next
        inc ax
next:   inc bx
        loop s

    mov ax,4c00h
    int 21h
code ends
```

应用示例

给出下面一组数据：

```
data segment
  db 8,11,8,1,8,5,63,38
data ends
```

请编程实现如下统计，用ax保存统计结果

（1）统计数值为8的字节的个数

（2）统计数值大于8的字节的个数

（3）统计数值小于8的字节的个数

（3）初始设置(ax)=0，然后用循环依次比较每个字节的值，找到一个小于8的数就将ax的值加1。

```
 5  code segment
 6  start:
 7          mov ax,data
 8          mov ds,ax
 9          mov bx,0
10          mov ax,0
11          mov cx,8
12      s:  cmp byte ptr [bx],8
13          jnb next
14          inc ax
15    next: inc bx
16          loop s
17
18          mov ax,4c00h
19          int 21h
20  code ends
```