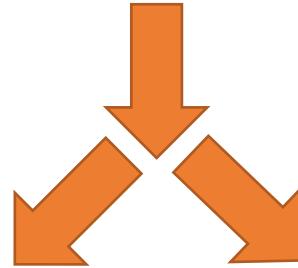


控制 Control



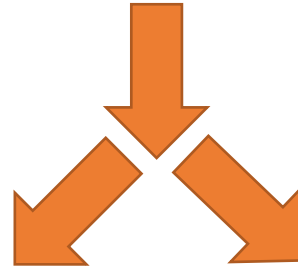
条件 Conditional

循环 Loop



Switch

控制 Control



条件 Conditional

循环 Loop



Switch

常见的条件结构

Conditional Structure

```
void decision(int x) {  
    if (x) {  
        op1();  
    } else {  
        op2();  
    }  
}
```

条件_x: 值或表达式



进行运算或判断（四则运算，逻辑运算...）

属性（是否为0，大于，小于...）

1 如何利用这一信息？

2 如何存储这一信息？

3 如何控制程序进行跳转？

条件码 Condition Code

——如何利用

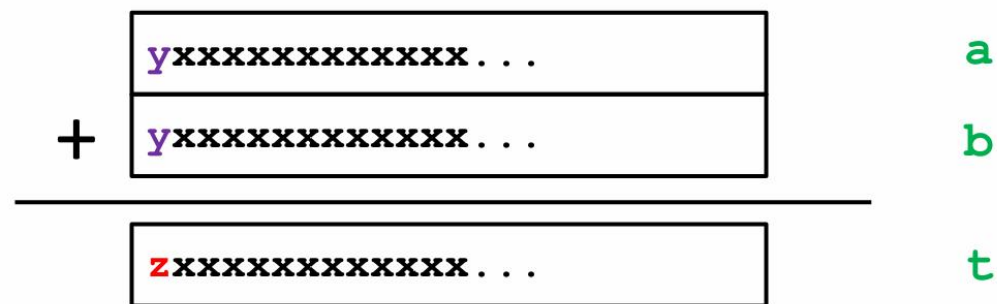
条件码：描述最近一次算术或逻辑操作的属性

CF Unsigned 进位或借位标志

ZF 最近一次操作结果为0 零标志

SF Signed 最近一次操作为负数 符号标志

OF Signed 溢出标志



$$z = \sim y$$

`(a>0 && b>0 && t<0) || (a<0 && b<0 && t>=0)`

条件码 Condition Code

CF/ZF/SF/OF

操作结束后赋值

发生 1

未发生 0

`addq Src, Dest` \leftrightarrow `t = a+b` CF/ZF/SF/OF

`cmpq Src2, Src1` CF/ZF/SF/OF

`cmpq b, a` like computing `a-b` without setting destination

`testq Src2, Src1` ZF/SF

▪ `testq b, a` like computing `a&b` without setting destination

条件码的存储与访问

SetX 指令 e.g. `sete %al` (将%rax最低位设置为此时ZF的值)

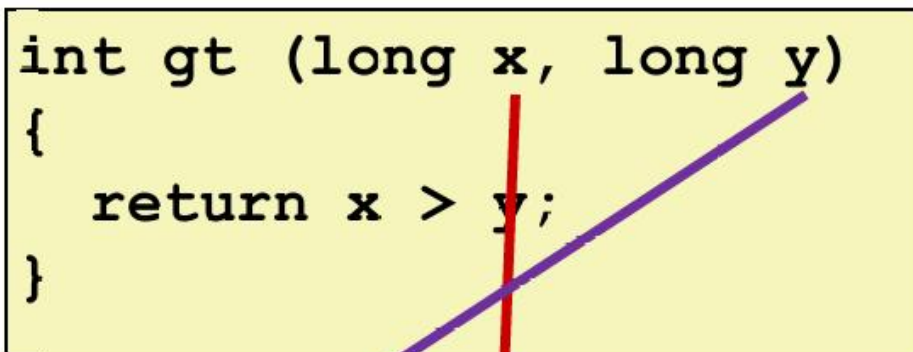
| SetX | Condition | Description |
|--------------------|--------------------------------------|---------------------------|
| <code>sete</code> | ZF | Equal / Zero |
| <code>setne</code> | \sim ZF | Not Equal / Not Zero |
| <code>sets</code> | SF | Negative |
| <code>setns</code> | \sim SF | Nonnegative |
| <code>setg</code> | $\sim (SF \wedge OF) \ \& \ \sim ZF$ | Greater (Signed) |
| <code>setge</code> | $\sim (SF \wedge OF)$ | Greater or Equal (Signed) |
| <code>setl</code> | $(SF \wedge OF)$ | Less (Signed) |
| <code>setle</code> | $(SF \wedge OF) \ \ ZF$ | Less or Equal (Signed) |
| <code>seta</code> | $\sim CF \ \& \ \sim ZF$ | Above (unsigned) |
| <code>setb</code> | CF | Below (unsigned) |

条件码的存储与访问

`setx` 指令 e.g. `sete %al` (将`%rax`最低位设置为此时ZF的值)

用`movzbl`指令 对`%rax`高位清零

```
int gt (long x, long y)
{
    return x > y;
}
```



```
cmpq    %rsi, %rdi    # Compare x:y
setg     %al           # Set when >
movzbl   %al, %eax     # Zero rest of %rax
ret
```

| Register | Use(s) |
|----------|--------------|
| %rdi | Argument x |
| %rsi | Argument y |
| %rax | Return value |

程序跳转 `jump jX`

| jX | Condition | Description |
|------------------|--|---------------------------|
| <code>jmp</code> | 1 | Unconditional |
| <code>je</code> | ZF | Equal / Zero |
| <code>jne</code> | \sim ZF | Not Equal / Not Zero |
| <code>js</code> | SF | Negative |
| <code>jns</code> | \sim SF | Nonnegative |
| <code>jg</code> | \sim (SF \wedge OF) $\&$ \sim ZF | Greater (Signed) |
| <code>jge</code> | \sim (SF \wedge OF) | Greater or Equal (Signed) |
| <code>jl</code> | (SF \wedge OF) | Less (Signed) |
| <code>jle</code> | (SF \wedge OF) $ $ ZF | Less or Equal (Signed) |
| <code>ja</code> | \sim CF $\&$ \sim ZF | Above (unsigned) |
| <code>jb</code> | CF | Below (unsigned) |

类比 goto 程序

```
if (x > y)
    result = x-y;
else
    result = y-x;
return result;
```

```
long result;
int ntest = x <= y;
if (ntest) goto Else;
result = x-y;
goto Done;
Else:
    result = y-x;
Done:
    return result;
```

循环 Loop → goto

```
loop:
```

```
.....
```

```
    if(condition) goto loop;  
done:
```

**THANK
S!**