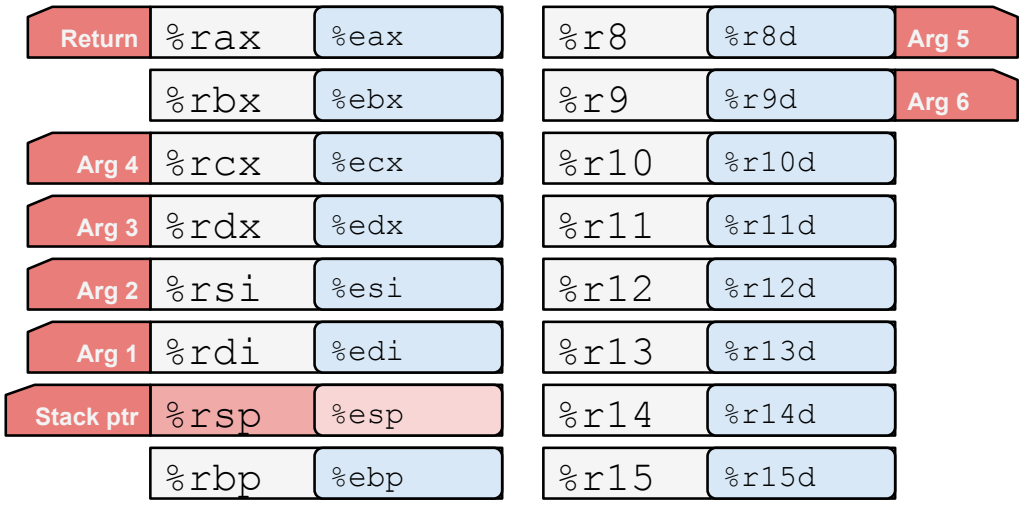
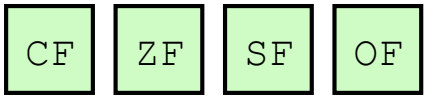
CSCI 2330 – Switch Exercises

Translate the x86-64 instructions below into a C function, which you can assume takes two arguments, the first of type **int** and the second of type **char**. **Do not use any goto statements in your function.**

```
mystery:
        subb    $'C', %sil # Recall that sil is low byte of %rsi
        cmpb    $8, %sil
        ja      .L7
        movzbl  %sil, %esi
        jmp     *.L4(,%rsi,8)
.L4:    .quad   .L6
        .quad   .L7
        .quad   .L7
        .quad   .L5
        .quad   .L7
        .quad   .L7
        .quad   .L7
        .quad   .L7
        .quad   .L3
.L7:    movl    $-0x80000000, %edi
        jmp     .L3
.L5:    subl    $32, %edi
        leal    (%rdi,%rdi,4), %eax
        movl    $9, %ecx
        cltd                # These two instructions divide %eax by
        idivl   %ecx        # %ecx, storing the quotient in %eax
        movl    %eax, %edi
.L6:    addl    $273, %edi
.L3:    movl    %edi, %eax
        ret
```





Condition codes

CF: **Carry flag** (set if carry-out bit = 1)

ZF: **Zero flag** (set if result = 0)

SF: **Sign flag** (set if result high bit = 1)

OF: **Overflow flag** (set if signed overflow)

Setting CCs

(1) `cmpq a,b` (set based on `b - a`)

(2) `testq a,b` (set based on `a & b`)

(3) arithmetic insts (implicit, all except `leaq`)

JX	Condition	Description
`jmp`	`1`	Unconditional
`je/jz`	`ZF`	Equal / Zero
`jne/jnz`	`~ZF`	Not Equal / Not Zero
`js`	`SF`	Negative
`jns`	`~SF`	Nonnegative
`jg`	`~(SF^OF)&~ZF`	Greater (Signed)
`jge`	`~(SF^OF)`	Greater or Equal (Signed)
`jl`	`(SF^OF)`	Less (Signed)
`jle`	`(SF^OF)|ZF`	Less or Equal (Signed)
`ja`	`~CF&~ZF`	Above (unsigned)
`jb`	`CF`	Below (unsigned)