Homework 4

I. Flags

Indicate the status (0, 1 or unchanged) of the following flags after each instruction, please write "——" if the flag doesn't change. Assume 3 in `%rax`, -8 in `%rbx`.

NOTE: Each instruction works independently and would **NOT** affect each other.

Instruction	OF	CF	ZF	SF
`addq %rbx, %rax`	0	0	0	1
`subq %rax, %rbx`	0	0	0	1
`leaq (%rax, %rax, 2), %rax`	--	--	--	--
`xorq %rax, %rax`	0	0	1	0
`salq $2, %rbx`	0	1	0	1
`cmpq %rax, %rbx`	0	0	0	1
`testq %rax, %rbx`	0	0	1	0

II. Control Flow

1. Assembly

Translate the following assembly into C codes. You can name local variables represented by `-0x8(%rbp)`, `-0x10(%rbp)`... freely as you like (for example: long a, b, c...).

```
    pushq %rbp
    movq  %rsp, %rbp
    movq  $1, -0x8(%rbp)
    movq  $1, -0x10(%rbp)
    movq  $2, -0x18(%rbp)
    jmp   .L1
.L2
    movq  -0x8(%rbp), %rax
    movq  %eax, -0x20(%rbp)
    movq  -0x10(%rbp), %rax
    movq  %rax, -0x8(%rbp)
    movq  -0x20(%rbp), %rax
    addq  %rax, -0x10(%rbp)
    addq  $1, -0x18(%rbp)
.L1
    cmpq  $9, -0x18(%rbp)
    jle   .L2
    movq  -0x10(%rbp), %rax
    popq  %rbp
    ret
```

(1) Translate into C codes.

```
long a = 1;
```

```
long b = 1;
long i = 2;
for(i; i<=9; i++){
    long tmp = a;
    a = b;
    b = a + tmp;
}
return b;
```

(2) Explain the function of this program.

compute the 10th fibonacci number

2. Jump Table

Translate the following switch statements into assembly using jump table.

```
long x = <some value>;
long result = 0;
switch(x){
    case 75:
        result = x + 1;
        break;
    case 76:
        result = x + x;
        // fall through
    case 77: case 79:
        result = result * 3;
        break;
    case 81:
        result = x;
        break;
    default:
        result = -1;
}
return result;
```

```
//Not the only solution
    .section  .rodata
    .align 8
.L4:
    .quad .L3
    .quad .L5
    .quad .L6
    .quad .L2
    .quad .L6
    .quad .L2
    .quad .L7
prog:
    movq  [x], -16(%rbp)
    movq  $0,  -8(%rbp)
    movq  -16(%rbp), %rax
    subq  $75, %rax
    cmpq  $6,  %rax
    ja    .L2
    movq  .L4(, %rax, 8), %rax
    jmp   *%rax
.L3:
    movq  -16(%rbp), %rax
    addq  $1, %rax
    movq  %rax, -8(%rbp)
    jmp   .L8
.L5:
    movq  -16(%rbp), %rax
    addq  %rax, %rax
    movq  %rax, -8(%rbp)
.L6:
    movq  -8(%rbp), %rdx
```

	leaq (%rdx, %rdx, 2), %rax
	movq %rax, -8(%rbp)
	jmp .L8
	.L7:
	movq -16(%rbp), %rax
	movq %rax, -8(%rbp)
	jmp .L8
	.L2:
	movq $-1, -8(%rbp)
	.L8:
	movq -8(%rbp), %rax

3. Conditional Move

The left program can be compiled into assembly on the right, please complete the assembly code. Note that `x` is at `%rdi` and `y` is at `%rsi`, and the return value should be put in `%rax`.

Not the only solution

long cmov_complex(long x, long y){
return x>y?x+y:x*y;
}
	movq %rdi, %rax
	addq %rsi, %rax
	movq %rdi, %rcx
	imulq %rsi, %rcx
	cmpq %rdi, %rsi
	cmovge %rcx, %rax
	ret