1. Translate the following assembly into C codes. Note that you can name local variables represented by -16(%rbp), -12(%rbp)… freely as you like.

```
        pushq   %rbp
        movq    %rsp,   %rbp
        subq    $16,    %rsp
        movl    $1,   -16(%rbp)
        movl    $1,   -12(%rbp)
        movl    $3,   -8(%rbp)
        jmp     .L2
.L3:
        movl    -16(%rbp),   %eax
        movl    %eax,    -4(%rbp)
        movl    -12(%rbp),   %eax
        movl    %eax,    -16(%rbp)
        movl    -4(%rbp),    %eax
        addl    %eax,    -12(%rbp)
        addl    $1,   -8(%rbp)
.L2:
        cmpl    $12, -8(%rbp)
        jle     .L3

    (Following code omitted)
```

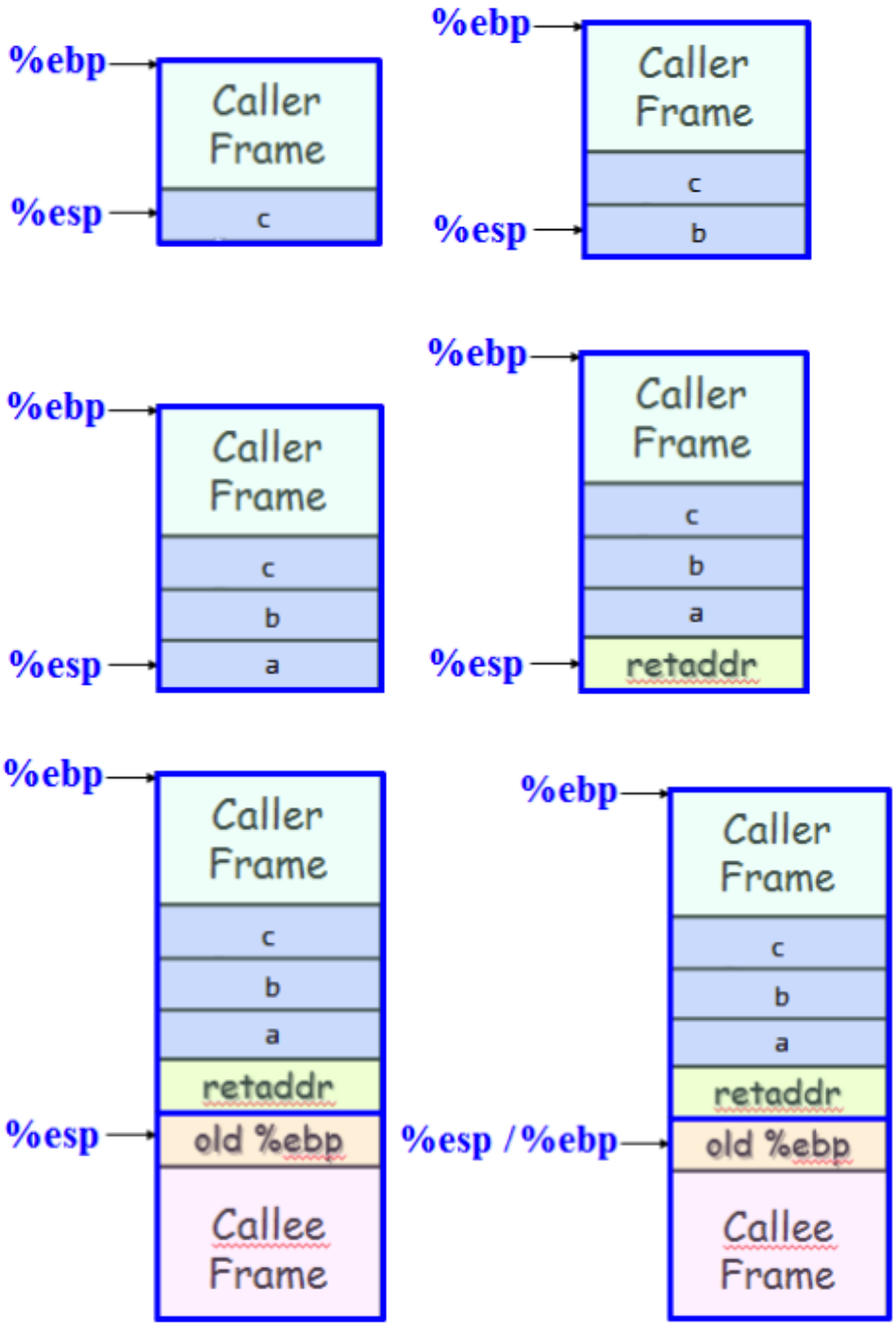
**Solution:**

```
int a = 1;
int b = 1;
int i;
for(i = 3; i <= 12; i++){
  int temp = a;
  a = b;
  b += temp;
}
```

2. Draw a stack for each step when the following function is being called and indicate where %ebp and %esp points. Here is an example where the first step takes place.

   *int Cal(int a, float b, double c);*

**Solution:**




3. Assume the initial value of the flags below is 0. Please indicate the status (0, 1) of the following flags after the instructions being executed. Assume -2 in %eax and 1 in %ebx. The following instructions execute independently and would

NOT affect each other. Assume that in the instructions below, the left is the source operand, the right is the dest operand.

**Solution:**

| Instruction | OF | SF | ZF | CF |
|---|---|---|---|---|
| leal (%edx,%edx,2),%edx | 0 | 0 | 0 | 0 |
| addl %eax, %ebx | 0 | 1 | 0 | 0 |
| xorl %eax, %eax | 0 | 0 | 1 | 0 |
| cmpl %ebx, %eax | 0 | 1 | 0 | 0 |

4. Translate the statements ***line 4 ~ line 9*** into assembly. Please use ebp(rbp for 64-bit machine) as the base address register, i.e. the immediate 1 and 2 should be put at the address -8(%ebp) and -4(%ebp) respectively.

```
 1  #include <stdio.h>
 2
 3  int main(){
 4      int a = 1;
 5      int b = 2;
 6
 7      if(a < b){
 8          a = b;
 9      }
10
11      printf("a = %d\nb = %d\n", a , b);
12  }
```

**Solution:**

```
            ...
            movl    $1,         -8(%rbp)
            movl    $2,         -4(%rbp)
            movl    -8(%rbp),   %eax
            cmpl    -4(%rbp),   %eax
            jge     next
            movl    -4(%rbp),   %eax
            movl    %eax,       -8(%rbp)
next:
            ...
```