```
  call    rsum
  addq    %rbx, %rax
  popq    %rbx
.L9:
  rep; ret
```

上述代码很容易改编为 Y86-64 代码：

```
# long rsum(long *start, long count)
# start in %rdi, count in %rsi
rsum:
        xorq %rax,%rax          # Set return value to 0
        andq %rsi,%rsi          # Set condition codes
        je     return           # If count == 0, return 0
        pushq %rbx              # Save callee-saved register
        mrmovq (%rdi),%rbx      # Get *start
        irmovq $-1,%r10
        addq %r10,%rsi          # count--
        irmovq $8,%r10
        addq %r10,%rdi          # start++
        call rsum
        addq %rbx,%rax          # Add *start to sum
        popq %rbx               # Restore callee-saved register
return:
        ret
```

4.5　这道题给了你一个练习写汇编代码的机会。

```
1   # long absSum(long *start, long count)
2   # start in %rdi, count in %rsi
3   absSum:
4           irmovq $8,%r8           # Constant 8
5           irmovq $1,%r9           # Constant 1
6           xorq %rax,%rax          # sum = 0
7           andq %rsi,%rsi          # Set condition codes
8           jmp  test
9   loop:
10          mrmovq (%rdi),%r10      # x = *start
11          xorq %r11,%r11          # Constant 0
12          subq %r10,%r11          # -x
13          jle pos                 # Skip if -x <= 0
14          rrmovq %r11,%r10        # x = -x
15  pos:
16          addq %r10,%rax          # Add to sum
17          addq %r8,%rdi           # start++
18          subq %r9,%rsi           # count--
19  test:
20          jne    loop             # Stop when 0
21          ret
```

4.6　这道题给了你一个练习写带条件传送汇编代码的机会。我们只给出循环的代码。剩下的部分与练习题 4.5 的一样。

```
9   loop:
10          mrmovq (%rdi),%r10      # x = *start
11          xorq %r11,%r11          # Constant 0
12          subq %r10,%r11          # -x
13          cmovg %r11,%r10         # If -x > 0 then x = -x
14          addq %r10,%rax          # Add to sum
15          addq %r8,%rdi           # start++
16          subq %r9,%rsi           # count--
17  test:
18          jne    loop             # Stop when 0
```