4.45

A. 错，压入了减去 8 的 %rsp 的值

B.

movq REG, -8(%rsp)

subq $8, %rsp

4.46

A. 错，当REG为%rsp是，这样会使得%rsp的值为(%rsp) + 8 而非(%rsp)

B.

addq $8, %rsp

movq -8(%rsp), REG

4.47

A.

void bubble\_p(long \*data, long count)

{

long i, last;

for(last = count - 1; last > 0; last--)

{

for(i = 0; i < last; i++)

{

if(\*(data+i+1) < \*(data+i))

{

long t = \*(data+i+1);

\*(data+i+1) = \*(data+i);

\*(data+i) = t;

}

}

}

}

B.

.pos 0

irmovq stack, %rsp # 设置栈帧

call main

halt

# 设置数组

.align 8

array:

.quad 0x000000000005

.quad 0x000000000003

.quad 0x000000000004

.quad 0x000000000002

main:

irmovq array,%rdi

irmovq $4,%rsi

call bubble\_b

ret

#函数bubble\_b

bubble\_b:

irmovq $1, %r9 # 常数 1

irmovq $8, %r10 # 常数 8

rrmovq %rsi, %rax

subq %r9, %rax # last = %rax = count - 1

je P4

P1:

xorq %rcx, %rcx # i = %rcx = 0

P2:

rrmovq %rcx, %rdx # %rdx = i

addq %rdx, %rdx

addq %rdx, %rdx

addq %rdx, %rdx

addq %rdi, %rdx # %rdx = data + 8 \* i

mrmovq (%rdx), %r11

addq %r10, %rdx # %rdx = data + 8 \* i + 8

mrmovq (%rdx), %rbx

rrmovq %rbx, %r8

subq %r11, %rbx # %rbx = data[i + 1] - data[i]

jge P3

rmmovq %r11, (%rdx) # data[i + 1] = data[i]

subq %r10, %rdx

rmmovq %r8, (%rdx) # data[i] = data[i + 1]

P3:

addq %r9, %rcx # i++

rrmovq %rcx, %r12

subq %rax, %r12 # i - last

jl P2

subq %r9, %rax # last--

jg P1

P4:

ret

# 栈从 0x200 向低地址生长

.pos 0x200

stack:

测试结果：

前

0x0000000000000005

0x0000000000000003

0x0000000000000004

0x0000000000000002

后

0x0000000000000002

0x0000000000000003

0x0000000000000004

0x0000000000000005

4.48

mrmovq (%rdx), %r9

rrmovq %rdx, %rsi

addq %r11, %rsi

mrmovq (%rsi), %rcx

rrmovq %rcx, %r11

subq %r8, %r11

rrmovq %rcx, %r11

cmovl %r8, %r11

rmmovq %r11, (%rsi)

rrmovq %r8, %r11

cmovl %rcx, %r11

rmmovq %r11, (%rdx)

4.49

mrmovq (%rdx), %r9

rrmovq %rdx, %rsi

addq %r11, %rsi

mrmovq (%rsi), %rcx

rrmovq %rcx, %r11

subq %r8, %r11

cmovl %rcx, %r8

xorq %r12, %r12

mrmovq (%rdx), %r11

subq %r11, %r12

addq %r9, %r12

rmmovq %r9, (%rdx)

subq %r12, %rcx

rmmovq %rcx, (%rsi)

4.50

main:

irmovq array, %r10

irmovq $1,%rdi

call switchv

rmmovq %rax, (%r10)

irmovq $-1,%rdi

call switchv

rmmovq %rax, 8(%r10)

irmovq $3,%rdi

call switchv

rmmovq %rax, 16(%r10)

irmovq $5,%rdi

call switchv

rmmovq %rax, 24(%r10)

ret

# long switchv(long idx)

# idx in %rdi

switchv:

# contant number

irmovq $8, %r8

irmovq $0, %r10

irmovq $1, %r11

irmovq $0, %rax

irmovq table, %rcx

rrmovq %rdi, %rdx

subq %r8, %rdx

jg def

subq %r10, %rdi

jl def

mul:

subq %r10, %rdi

je addr

addq %r8, %rcx

subq %r11, %rdi

jmp mul

addr:

addq %r8, %rcx

mrmovq (%rcx), %rdi

pushq %rdi

ret

def:

irmovq table, %rcx

mrmovq (%rcx), %rdi

pushq %rdi

ret

L0:

irmovq $0xaaa, %rax

ret

L1:

jmp LD

L2:

jmp L5

L3:

irmovq $0xccc, %rax

ret

L4:

jmp LD

L5:

irmovq $0xbbb, %rax

ret

LD:

irmovq $0xddd, %rax

ret

4.51

iaddq V, rB

Fetch:

​ icode:ifun <-- M1[PC]

​ rA:rB <-- M1[PC+1]

​ valC <-- M8[PC+2]

​ valP <-- PC+10

Decode:

​ valB <-- R[rB]

Execute:

​ ValE <-- valB + valC

Memory:

Write back:

​ R[rB] <-- valE

PC update:

​ PC <-- valP