1、

A.

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

{

long i,last;

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

{

long \*a=data,\*b;

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

{

b=a+1;

if (\*b<\*a)

{

long t=\*b;

\*b=\*a;

\*a=t;

}

}

}

}

B.

.pos 0x0

init:

irmovl Stack, %esp

irmovl Stack, %ebp

call Main

halt

.align 4

ele1:

.long 0x00c

ele2:

.long 0x00a

ele3:

.long 0x00b

Main:

pushl %ebp

rrmovl %esp,%ebp

irmovl $3,%edx

pushl %edx # count = 3

irmovl ele1,%edx

pushl %edx

call bubble\_a

rrmovl %ebp,%esp

popl %ebp

ret

bubble\_a: ## @bubble\_a

## BB#0:

pushl %ebp

rrmovl %esp, %ebp

irmovl $28,%edx

subl %edx, %esp

mrmovl 12(%ebp), %eax

mrmovl 8(%ebp), %ecx

rmmovl %ecx, -4(%ebp)

rmmovl %eax, -8(%ebp)

mrmovl -8(%ebp), %eax

irmovl $1,%edx

subl %edx, %eax

rmmovl %eax, -16(%ebp)

LBB0\_1: ## =>This Loop Header: Depth=1

## Child Loop BB0\_3 Depth 2

xorl %edx,%edx

mrmovl -16(%ebp),%eax

subl %edx,%eax

jle LBB0\_10

## BB#2: ## in Loop: Header=BB0\_1 Depth=1

mrmovl -4(%ebp), %eax

rmmovl %eax, -20(%ebp)

xorl %edx,%edx

rmmovl %edx, -12(%ebp)

LBB0\_3: ## Parent Loop BB0\_1 Depth=1

## => This Inner Loop Header: Depth=2

mrmovl -12(%ebp), %eax

mrmovl -16(%ebp),%edx

subl %edx, %eax

jge LBB0\_8

## BB#4: ## in Loop: Header=BB0\_3 Depth=2

mrmovl -20(%ebp), %eax

irmovl $4,%edx

addl %edx, %eax

rmmovl %eax, -24(%ebp)

mrmovl (%eax), %eax

mrmovl -20(%ebp), %ecx

mrmovl (%ecx),%edx

subl %edx, %eax

jge LBB0\_6

## BB#5: ## in Loop: Header=BB0\_3 Depth=2

mrmovl -24(%ebp), %eax

mrmovl (%eax), %eax

rmmovl %eax, -28(%ebp)

mrmovl -20(%ebp), %eax

mrmovl (%eax), %eax

mrmovl -24(%ebp), %ecx

rmmovl %eax, (%ecx)

mrmovl -28(%ebp), %eax

mrmovl -20(%ebp), %ecx

rmmovl %eax, (%ecx)

LBB0\_6: ## in Loop: Header=BB0\_3 Depth=2

jmp LBB0\_7

LBB0\_7: ## in Loop: Header=BB0\_3 Depth=2

mrmovl -12(%ebp), %eax

irmovl $1,%edx

addl %edx, %eax

rmmovl %eax, -12(%ebp)

mrmovl -20(%ebp), %eax

irmovl $4,%edx

addl %edx, %eax

rmmovl %eax, -20(%ebp)

jmp LBB0\_3

LBB0\_8: ## in Loop: Header=BB0\_1 Depth=1

jmp LBB0\_9

LBB0\_9: ## in Loop: Header=BB0\_1 Depth=1

mrmovl -16(%ebp), %eax

irmovl $-1,%edx

addl %edx, %eax

rmmovl %eax, -16(%ebp)

jmp LBB0\_1

LBB0\_10:

irmovl $28,%edx

addl %edx, %esp

popl %ebp

ret

.pos 0x100

Stack:

注意这里的减法都需要是对signed操作，因为Y86中没有对signed 的运算，所以所有的减法如果小于0就会出现错误，导致ADR错误。故在这里需要规定都是对signed操作。

2、

C版本：

a为指向data+i的指针，b为指向data+i+1的指针

{

long t=\*a;

if (\*b<\*a) t=\*b;

\*b^=\*a;

\*a=t;

\*b^=\*a;

}

Y86:

mrmovl -20(%ebp), %eax #取出a=data

irmovl $4,%edx

addl %edx, %eax #计算b=a+1

rmmovl %eax, -24(%ebp) #存储b

mrmovl (%eax), %eax #计算\*b

mrmovl -20(%ebp), %ecx #取出a

mrmovl (%ecx),%edx #计算\*a

xorl %ecx,%ecx #清空%ecx

rrmovl %edx,%ecx ＃long t=\*a

subl %edx, %eax #if (\*b<\*a)

cmovl %eax,%ecx #t=\*b

xorl %edx,%eax #计算\*b^\*a存储在%eax中

rrmovl %ecx,%edx #将t的值存储在%edx中

xorl %edx,%eax #计算R[%eax]^t的值，存储在%eax中

mrmovl -20(%ebp),%ecx ＃取出a的指针指向的地址，存放在%ecx中

rmmovl %edx,(%ecx) #将t的值放在\*a的位置

mrmovl -24(%ebp),%ecx ＃取出b的指针指向的地址，存放在%ecx中

rmmovl %eax,(%ecx) #将%eax的值放在\*b的位置