**ЛАБОРАТОРНАЯ РАБОТА №3 ВВЕДЕНИЕ В АРХИТЕКТУРУ x86-64**

**Цели:**

1. Знакомство с программной архитектурой x86-64.
2. Анализ ассемблерного листинга программы для архитектуры x86-64.

**Листинг программы на C:**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 140000

**void** swap(**int** \*a, **int** \*b) {

**int** tmp = \*a;

\*a = \*b;

\*b = tmp;

}

**int**\* FillArray(**long** **long** n) {

**int** \*array = calloc(n, **sizeof**(**int**));

**for** (**int** i = 0; i < n; i++) {

array[i] = rand() % 1000;

}

**return** array;

}

**int** main() {

**long** **long** n = SIZE;

**int** \*array = FillArray(n);

**for** (**int** i = 0; i < n; i++) {

**for** (**int** j = 0 ; j < n - 1; j++) {

**if** (array[j] > array[j + 1]) {

swap(&array[j], &array[j + 1]);

}

}

}

**return** 0;

}

**Команды для компиляции ассемблерного кода:**

* gcc -o lab3-O0.s -O0 main.c -S
* gcc -o lab3-O1.s -O1 main.c -S
* gcc -o lab3-O2.s -O2 main.c -S
* gcc -o lab3-O3.s -O3 main.c -S
* gcc -o lab3-Ofast.s -Ofast main.c -S
* gcc -o lab3-Os.s -Os main.c -S

**Листинг кода на ассемблере: (см. приложение 1)**

**Вывод:**

По результатам выполнения лабораторной работы мы ознакомились с программной архитектурой x86-64, научились анализировать ассемблерные листинги программ на данной архитектуре.

**Приложение 1**

**Листинг программы (уровень O0)**

.file "main.c"

.text

.globl swap

.type swap, @function

swap:

swap:

.LFB6:

.cfi\_startproc

endbr64

pushq %rbp //Сохранение указателя старого кадра, эквивалентно созданию нового кадра стека

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp //Пусть %rbp указывает на начало нового кадра стека

.cfi\_def\_cfa\_register 6

movq %rdi, -24(%rbp) //записываем &a в стек по адресу rbp-24

movq %rsi, -32(%rbp) //записываем &b в стек по адресу rbp-32

movq -24(%rbp), %rax //записываем &a в tmp

movl (%rax), %eax //записываем tmp в eax

movl %eax, -4(%rbp) //записываем a в стек по адресу rbp-4

movq -32(%rbp), %rax //записываем &b в регистр rax

movl (%rax), %edx //записываем b в регистр edx

movq -24(%rbp), %rax //записываем &a в регистр rax

movl %edx, (%rax) //записываем b в &a

movq -32(%rbp), %rax //записываем &b в регистр rax

movl -4(%rbp), %edx //записываем a в регистр edx

movl %edx, (%rax) //записываем a в &b

nop

popq %rbp //записывает указатель на вершину стека в rbp, "забывает" о стеке в функции (возврат значения)

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE6:

.size swap, .-swap

.globl FillArray

.type FillArray, @function

FillArray:

.LFB7:

.cfi\_startproc

endbr64

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $32, %rsp

movq %rdi, -24(%rbp) //записываем n в стек по адресу rbp-24

movq -24(%rbp), %rax //записываем n в регистр rax

movl $4, %esi //записываем 4 в регистр esi (2ой аргумент calloc'а)

movq %rax, %rdi //записываем n в регистр rdi (1ый аргумент calloc'а)

call calloc***@PLT***

movq %rax, -8(%rbp) //записываем &array в стек по адресу rbp-8

movl $0, -12(%rbp) //записываем 0 в стек по адресу rbp-12 ()

jmp .L3

.L4: //тело цикла for

call rand***@PLT***

movl -12(%rbp), %edx //записываем 0 в регистр edx

movslq %edx, %rdx //записываем 0 в регистр rdx

leaq 0(,%rdx,4), %rcx

movq -8(%rbp), %rdx

addq %rdx, %rcx

movslq %eax, %rdx

imulq $274877907, %rdx, %rdx

shrq $32, %rdx

sarl $6, %edx

movl %eax, %esi

sarl $31, %esi

subl %esi, %edx

imull $1000, %edx, %esi

subl %esi, %eax

movl %eax, %edx

movl %edx, (%rcx)

addl $1, -12(%rbp) //i++

.L3: //цикл for

movl -12(%rbp), %eax //записываем 0 в регистр eax

cltq

cmpq %rax, -24(%rbp) //сравнение (i < n)

jg .L4

movq -8(%rbp), %rax //записываем &array в регистр rax

leave

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE7:

.size FillArray, .-FillArray

.globl main

.type main, @function

main:

.LFB8:

.cfi\_startproc

endbr64

pushq %rbp

.cfi\_def\_cfa\_offset 16

.cfi\_offset 6, -16

movq %rsp, %rbp

.cfi\_def\_cfa\_register 6

subq $48, %rsp

movq $140000, -32(%rbp) //записываем 140000 в стек по адресу rbp-32

movq -32(%rbp), %rax //записываем 140000 в регистр rax (в n)

movq %rax, %rdi //записываем n в регистр rdi

call FillArray

movq %rax, -24(%rbp) //записываем &array в стек по адресу rbp-24

movl $0, -40(%rbp) //записываем 0 в стек по адресу rbp-40

jmp .L7

.L11:

movl $0, -36(%rbp) //записываем 0 в стек по адресу rbp-36

jmp .L8

.L10:

movl -36(%rbp), %eax //записываем 0 в регистр eax (j = 0)

cltq

leaq 0(,%rax,4), %rdx //записываем j\*4 в регистр rdx

movq -24(%rbp), %rax //записываем &array в регистр rax

addq %rdx, %rax //&array + j\*4

movl (%rax), %edx //записываем array[j] в регистр edx

movl -36(%rbp), %eax //записываем j в регистр eax

cltq

addq $1, %rax // j + 1

leaq 0(,%rax,4), %rcx //записываем (j+1)\*4 в регистр rcx

movq -24(%rbp), %rax //записываем &array в регистр rax

addq %rcx, %rax //&array + (j+1)\*4

movl (%rax), %eax //записываем array[j+1] в регистр eax

cmpl %eax, %edx //сравнение (array[j] > array[j + 1]) (если нет то переход)

jle .L9

movl -36(%rbp), %eax //записываем j в регистр eax

cltq

addq $1, %rax //j+1

leaq 0(,%rax,4), %rdx //записываем (j+1)\*4 в регистр rdx

movq -24(%rbp), %rax //записываем &array в регистр rax

addq %rax, %rdx //&array + (j+1)\*4

movl -36(%rbp), %eax //записываем j в регистр eax

cltq

leaq 0(,%rax,4), %rcx //записываем j\*4 в регистр rcx

movq -24(%rbp), %rax //записываем &array в регистр rax

addq %rcx, %rax //&array + j\*4

movq %rdx, %rsi //передача аргументов в swap

movq %rax, %rdi //rsi=array[j+1]; rdi=array[j]

call swap //передает управление функции по адресу swap, записывает из rip в стек адрес возврата

.L9:

addl $1, -36(%rbp) //j++

.L8: //цикл for(int j = 0; j < n-1; j++)

movl -36(%rbp), %eax //записываем 0 в регистр eax (j)

cltq

movq -32(%rbp), %rdx //записываем 140000 в регистр rdx (n)

subq $1, %rdx // (n-1)

cmpq %rdx, %rax //сравнение (j < n-1) (n-1 - j)

jl .L10

addl $1, -40(%rbp) //i++

.L7: //цикл for(int = 0; i < n; i++)

movl -40(%rbp), %eax //записываем 0 в регистр eax (i = 0)

cltq

cmpq %rax, -32(%rbp) //сравнение (i < n)

jg .L11 //выполняет переход, если первый операнд больше второго

movl $0, %eax //записываем 0 в регистр eax

leave //копирует указатель кадра rbp в указатель стека rsp (освобождает память)

.cfi\_def\_cfa 7, 8

ret

.cfi\_endproc

.LFE8:

.size main, .-main

.ident "GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0"

.section .note.GNU-stack,"",@progbits

.section .note.gnu.property,"a"

.align 8

.**long** ***1f*** - ***0f***

.**long** ***4f*** - ***1f***

.**long** 5

0:

.string "GNU"

1:

.align 8

.**long** 0xc0000002

.**long** ***3f*** - ***2f***

2:

.**long** 0x3

3:

.align 8

4:

**Приложение 2**

**Листинг программы (уровень O1):**

.file "main.c"

.text

.globl swap

.type swap, @function

swap: //функция не используется

.LFB39:

.cfi\_startproc

endbr64

movl (%rdi), %eax

movl (%rsi), %edx

movl %edx, (%rdi)

movl %eax, (%rsi)

ret

.cfi\_endproc

.LFE39:

.size swap, .-swap

.globl FillArray

.type FillArray, @function

FillArray:

.LFB40:

.cfi\_startproc

endbr64

pushq %r12

.cfi\_def\_cfa\_offset 16

.cfi\_offset 12, -16

pushq %rbp

.cfi\_def\_cfa\_offset 24

.cfi\_offset 6, -24

pushq %rbx

.cfi\_def\_cfa\_offset 32

.cfi\_offset 3, -32

movq %rdi, %rbp //записываем 140000 в регистр rbp (1ый аргумент calloc'а)

movl $4, %esi //записываем 4 в регистр esi (2ой аргумент calloc'а)

call calloc***@PLT***

movq %rax, %r12 //записываем &array в регистр r12

testq %rbp, %rbp //логическое сравнение (rbp & rbp) (условие прохода по циклу)(результат никуда не записывается, только изменяет флаги)

jle .L2

movl $0, %ebx //записываем 0 в регистр ebx (i = 0)

.L4: //array[i] = rand() % 1000;

call rand***@PLT***

movslq %eax, %rdx

imulq $274877907, %rdx, %rdx

sarq $38, %rdx

movl %eax, %ecx

sarl $31, %ecx

subl %ecx, %edx

imull $1000, %edx, %edx

subl %edx, %eax

movl %eax, (%r12,%rbx,4) //запись элемента в массив

addq $1, %rbx //i++

cmpq %rbx, %rbp

jne .L4 //цикл for

.L2: //все взятые значения из стека были положены туда в этом порядке в LFB40

movq %r12, %rax //записываем &array в регистр rax

popq %rbx

.cfi\_def\_cfa\_offset 24

popq %rbp

.cfi\_def\_cfa\_offset 16

popq %r12

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE40:

.size FillArray, .-FillArray

.globl main

.type main, @function

main:

.LFB41:

.cfi\_startproc

endbr64

subq $8, %rsp //rsp = rsp - 8

.cfi\_def\_cfa\_offset 16

movl $140000, %edi //записываем 140000 в регистр edi

call FillArray //вызываем функцию FillArray

movq %rax, %r8 //записываем &array в регистр r8

movl $140000, %edi //записываем 140000 в регистр edi

leaq 559996(%rax), %rsi //записываем адрес последнего элемента массива в rsi

jmp .L8

.L9:

addq $4, %rax //сдвигаемся по массиву на один элемент

cmpq %rsi, %rax //если (еще не) вышли за границы массива

je .L14 //переход если rsi == rax

.L10:

movl (%rax), %edx //записываем array[j] в регистр edx

movl 4(%rax), %ecx //записываем array[j+1] в регистр ecx

cmpl %ecx, %edx //переход если (array[j] < array[j+1])

jle .L9

movl %ecx, (%rax) //swap

movl %edx, 4(%rax)

jmp .L9

.L14:

subl $1, %edi

je .L11

.L8:

movq %r8, %rax //записываем &array в регистр rax

jmp .L10

.L11:

movl $0, %eax

addq $8, %rsp

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE41:

.size main, .-main

.ident "GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0"

.section .note.GNU-stack,"",@progbits

.section .note.gnu.property,"a"

.align 8

.**long** ***1f*** - ***0f***

.**long** ***4f*** - ***1f***

.**long** 5

0:

.string "GNU"

1:

.align 8

.**long** 0xc0000002

.**long** ***3f*** - ***2f***

2:

.**long** 0x3

3:

.align 8

4:

**Приложение 3**

**Листинг программы (уровень O2):**

.file "main.c"

.text

.p2align 4

.globl swap

.type swap, @function

swap:

.LFB39:

.cfi\_startproc

endbr64

movl (%rdi), %eax

movl (%rsi), %edx

movl %edx, (%rdi)

movl %eax, (%rsi)

ret

.cfi\_endproc

.LFE39:

.size swap, .-swap

.p2align 4

.globl FillArray

.type FillArray, @function

FillArray:

.LFB40:

.cfi\_startproc

endbr64

pushq %r12

.cfi\_def\_cfa\_offset 16

.cfi\_offset 12, -16

movl $4, %esi //записываем 4 в регистр esi (2ой аргумент calloc'а)

pushq %rbp

.cfi\_def\_cfa\_offset 24

.cfi\_offset 6, -24

pushq %rbx

.cfi\_def\_cfa\_offset 32

.cfi\_offset 3, -32

movq %rdi, %rbx //записываем 140000 в регистр rbp (1ый аргумент calloc'а)

call calloc***@PLT***

movq %rax, %r12 //записываем &array в регистр r12

testq %rbx, %rbx //условие прохода по циклу

jle .L3

xorl %ebp, %ebp

.p2align 4,,10

.p2align 3

.L5: //array[i] = rand() % 1000;

call rand***@PLT***

movslq %eax, %rdx

movl %eax, %ecx

imulq $274877907, %rdx, %rdx

sarl $31, %ecx

sarq $38, %rdx

subl %ecx, %edx

imull $1000, %edx, %edx

subl %edx, %eax

movl %eax, (%r12,%rbp,4) //запись элемента в массив

addq $1, %rbp //i++

cmpq %rbp, %rbx

jne .L5 //цикл for

.L3:

movq %r12, %rax //записываем &array в регистр rax

popq %rbx

.cfi\_def\_cfa\_offset 24

popq %rbp

.cfi\_def\_cfa\_offset 16

popq %r12

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE40:

.size FillArray, .-FillArray

.section .text.startup,"ax",@progbits

.p2align 4

.globl main

.type main, @function

main:

.LFB41:

.cfi\_startproc

endbr64

movl $140000, %edi //записываем 140000 в регистр edi

subq $8, %rsp //резервируем место в стеке

.cfi\_def\_cfa\_offset 16

call FillArray

movl $140000, %edi //записываем 140000 в регистр edi

movq %rax, %r8 //записываем &array в регистр r8

leaq 559996(%rax), %rsi //записываем адрес последнего элемента массива в rsi

.L9:

movq %r8, %rax //записываем &array в регистр rax

.p2align 4,,10

.p2align 3

.L11:

movl (%rax), %edx //записываем array[j] в регистр edx

movl 4(%rax), %ecx //записываем array[j+1] в регистр ecx

cmpl %ecx, %edx //переход если (array[j] < array[j+1])

jle .L10

movl %ecx, (%rax) //swap

movl %edx, 4(%rax)

.L10:

addq $4, %rax //переходим к следующей паре элементов (итерация по j)

cmpq %rsi, %rax

jne .L11

subl $1, %edi //итерация по циклу for по переменной i

jne .L9

xorl %eax, %eax

addq $8, %rsp

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE41:

.size main, .-main

.ident "GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0"

.section .note.GNU-stack,"",@progbits

.section .note.gnu.property,"a"

.align 8

.**long** ***1f*** - ***0f***

.**long** ***4f*** - ***1f***

.**long** 5

0:

.string "GNU"

1:

.align 8

.**long** 0xc0000002

.**long** ***3f*** - ***2f***

2:

.**long** 0x3

3:

.align 8

4:

**Листинг программы (уровень O3):**

.file "main.c"

.text

.p2align 4

.globl swap

.type swap, @function

swap:

.LFB39:

.cfi\_startproc

endbr64

movl (%rdi), %eax

movl (%rsi), %edx

movl %edx, (%rdi)

movl %eax, (%rsi)

ret

.cfi\_endproc

.LFE39:

.size swap, .-swap

.p2align 4

.globl FillArray

.type FillArray, @function

FillArray:

.LFB40:

.cfi\_startproc

endbr64

pushq %r12

.cfi\_def\_cfa\_offset 16

.cfi\_offset 12, -16

movl $4, %esi

pushq %rbp

.cfi\_def\_cfa\_offset 24

.cfi\_offset 6, -24

pushq %rbx

.cfi\_def\_cfa\_offset 32

.cfi\_offset 3, -32

movq %rdi, %rbx

call calloc***@PLT***

movq %rax, %r12

testq %rbx, %rbx

jle .L3

xorl %ebp, %ebp

.p2align 4,,10

.p2align 3

.L5:

call rand***@PLT***

movslq %eax, %rdx

movl %eax, %ecx

imulq $274877907, %rdx, %rdx

sarl $31, %ecx

sarq $38, %rdx

subl %ecx, %edx

imull $1000, %edx, %edx

subl %edx, %eax

movl %eax, (%r12,%rbp,4)

addq $1, %rbp

cmpq %rbp, %rbx

jne .L5

.L3:

movq %r12, %rax

popq %rbx

.cfi\_def\_cfa\_offset 24

popq %rbp

.cfi\_def\_cfa\_offset 16

popq %r12

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE40:

.size FillArray, .-FillArray

.section .text.startup,"ax",@progbits

.p2align 4

.globl main

.type main, @function

main:

.LFB41:

.cfi\_startproc

endbr64

movl $140000, %edi

subq $8, %rsp

.cfi\_def\_cfa\_offset 16

call FillArray

movl $140000, %edi

movq %rax, %r8

leaq 559996(%rax), %rsi

.L9:

movq %r8, %rdx

.p2align 4,,10

.p2align 3

.L11: //условие + swap

movq (%rdx), %xmm0 //записываем j в xmm0

pshufd $0xe5, %xmm0, %xmm2 //выбираем значения из xmm0 и записываем в xmm2

movd %xmm0, %ecx

movd %xmm2, %eax

pshufd $225, %xmm0, %xmm1 //реализация swap'а

cmpl %eax, %ecx

jle .L10

movq %xmm1, (%rdx)

.L10:

addq $4, %rdx

cmpq %rsi, %rdx

jne .L11

subl $1, %edi

jne .L9

xorl %eax, %eax

addq $8, %rsp

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE41:

.size main, .-main

.ident "GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0"

.section .note.GNU-stack,"",@progbits

.section .note.gnu.property,"a"

.align 8

.**long** ***1f*** - ***0f***

.**long** ***4f*** - ***1f***

.**long** 5

0:

.string "GNU"

1:

.align 8

.**long** 0xc0000002

.**long** ***3f*** - ***2f***

2:

.**long** 0x3

3:

.align 8

4:

**Листинг программы (уровень Ofast):**

.file "main.c"

.text

.p2align 4

.globl swap

.type swap, @function

swap:

.LFB39:

.cfi\_startproc

endbr64

movl (%rdi), %eax

movl (%rsi), %edx

movl %edx, (%rdi)

movl %eax, (%rsi)

ret

.cfi\_endproc

.LFE39:

.size swap, .-swap

.p2align 4

.globl FillArray

.type FillArray, @function

FillArray:

.LFB40:

.cfi\_startproc

endbr64

pushq %r12

.cfi\_def\_cfa\_offset 16

.cfi\_offset 12, -16

movl $4, %esi

pushq %rbp

.cfi\_def\_cfa\_offset 24

.cfi\_offset 6, -24

pushq %rbx

.cfi\_def\_cfa\_offset 32

.cfi\_offset 3, -32

movq %rdi, %rbx

call calloc***@PLT***

movq %rax, %r12

testq %rbx, %rbx

jle .L3

xorl %ebp, %ebp

.p2align 4,,10

.p2align 3

.L5:

call rand***@PLT***

movslq %eax, %rdx

movl %eax, %ecx

imulq $274877907, %rdx, %rdx

sarl $31, %ecx

sarq $38, %rdx

subl %ecx, %edx

imull $1000, %edx, %edx

subl %edx, %eax

movl %eax, (%r12,%rbp,4)

addq $1, %rbp

cmpq %rbp, %rbx

jne .L5

.L3:

movq %r12, %rax

popq %rbx

.cfi\_def\_cfa\_offset 24

popq %rbp

.cfi\_def\_cfa\_offset 16

popq %r12

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE40:

.size FillArray, .-FillArray

.section .text.startup,"ax",@progbits

.p2align 4

.globl main

.type main, @function

main:

.LFB41:

.cfi\_startproc

endbr64

movl $140000, %edi

subq $8, %rsp

.cfi\_def\_cfa\_offset 16

call FillArray

movl $140000, %edi

movq %rax, %r8

leaq 559996(%rax), %rsi

.L9:

movq %r8, %rdx

.p2align 4,,10

.p2align 3

.L11:

movq (%rdx), %xmm0

pshufd $0xe5, %xmm0, %xmm2

movd %xmm0, %ecx

movd %xmm2, %eax

pshufd $225, %xmm0, %xmm1

cmpl %eax, %ecx

jle .L10

movq %xmm1, (%rdx)

.L10:

addq $4, %rdx

cmpq %rsi, %rdx

jne .L11

subl $1, %edi

jne .L9

xorl %eax, %eax

addq $8, %rsp

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE41:

.size main, .-main

.ident "GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0"

.section .note.GNU-stack,"",@progbits

.section .note.gnu.property,"a"

.align 8

.**long** ***1f*** - ***0f***

.**long** ***4f*** - ***1f***

.**long** 5

0:

.string "GNU"

1:

.align 8

.**long** 0xc0000002

.**long** ***3f*** - ***2f***

2:

.**long** 0x3

3:

.align 8

4:

**Листинг программы (уровень Os):**

.file "main.c"

.text

.globl swap

.type swap, @function

swap:

.LFB24:

.cfi\_startproc

endbr64

movl (%rdi), %eax

movl (%rsi), %edx

movl %edx, (%rdi)

movl %eax, (%rsi)

ret

.cfi\_endproc

.LFE24:

.size swap, .-swap

.globl FillArray

.type FillArray, @function

FillArray:

.LFB25:

.cfi\_startproc

endbr64

pushq %r13

.cfi\_def\_cfa\_offset 16

.cfi\_offset 13, -16

movl $4, %esi

movl $1000, %r13d

pushq %r12

.cfi\_def\_cfa\_offset 24

.cfi\_offset 12, -24

pushq %rbp

.cfi\_def\_cfa\_offset 32

.cfi\_offset 6, -32

movq %rdi, %rbp

pushq %rbx

.cfi\_def\_cfa\_offset 40

.cfi\_offset 3, -40

xorl %ebx, %ebx

pushq %rcx

.cfi\_def\_cfa\_offset 48

call calloc***@PLT***

movq %rax, %r12

.L3:

cmpq %rbx, %rbp

jle .L7

call rand***@PLT***

cltd

idivl %r13d

movl %edx, (%r12,%rbx,4)

incq %rbx

jmp .L3

.L7:

popq %rdx

.cfi\_def\_cfa\_offset 40

movq %r12, %rax

popq %rbx

.cfi\_def\_cfa\_offset 32

popq %rbp

.cfi\_def\_cfa\_offset 24

popq %r12

.cfi\_def\_cfa\_offset 16

popq %r13

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE25:

.size FillArray, .-FillArray

.section .text.startup,"ax",@progbits

.globl main

.type main, @function

main:

.LFB26:

.cfi\_startproc

endbr64

pushq %rcx

.cfi\_def\_cfa\_offset 16

movl $140000, %edi

call FillArray

movl $140000, %ecx

leaq 559996(%rax), %r8

.L9:

movq %rax, %rdx

.L11:

movl (%rdx), %esi

movl 4(%rdx), %edi

cmpl %edi, %esi

jle .L10

movl %edi, (%rdx)

movl %esi, 4(%rdx)

.L10:

addq $4, %rdx

cmpq %r8, %rdx

jne .L11

decl %ecx

jne .L9

xorl %eax, %eax

popq %rdx

.cfi\_def\_cfa\_offset 8

ret

.cfi\_endproc

.LFE26:

.size main, .-main

.ident "GCC: (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0"

.section .note.GNU-stack,"",@progbits

.section .note.gnu.property,"a"

.align 8

.**long** ***1f*** - ***0f***

.**long** ***4f*** - ***1f***

.**long** 5

0:

.string "GNU"

1:

.align 8

.**long** 0xc0000002

.**long** ***3f*** - ***2f***

2:

.**long** 0x3

3:

.align 8

4: