###### **МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

###### **РОССИЙСКОЙ ФЕДЕРАЦИИ**

###### **НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

###### 

###### 

**ЭВМ И ПЕРИФЕРИЙНЫЕ УСТРОЙСТВА**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

ВВЕДЕНИЕ В АРХИТЕКТУРУ x86/x86-64

Студент:

Овчаренко Дарья Ивановна, группа 23211

Преподаватель:

Ассистент кафедры ПВ ФИТ

Мичуров Михаил Антонович

Новосибирск 2024

# **ЦЕЛЬ**

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

# **ЗАДАНИЕ**

Вариант задания: 4.

1. Изучить программную архитектуру x86/x86-64: набор регистров, основные арифметико-логические команды, способы адресации памяти, способы передачи управления, работу со стеком, вызов подпрограмм, передачу параметров в подпрограммы и возврат результатов, работу с арифметическим сопроцессором, работу с векторными расширениями.
2. Сгенерировать листинги исходной программы с оптимизациями –O0 и –O3 и проанализировать полученные коды.
3. Составить отчет по лабораторной работе.

# **ОПИСАНИЕ РАБОТЫ**

В ходе задания использовался компьютер с архитектурой x86\_64, с операционной системой Ubuntu 22.04.5 LTS и процессором AMD A6-6310 APU with AMD Radeon R4 Graphics.

###### **Пошаговое описание выполненной работы**

1. Была написана компьютерная программа, которая вычисляет sin(x) с помощью разложения в степенной ряд по первым N членам этого ряда.
2. Были изучены основные принципы работы в языке ассемблер.
3. С помощью сайта GodBolt (URL: https://godbolt.org/) были сгенерированы листинги исходной программы с оптимизациями -O0 (см. Приложение 2), -O1 (см. Приложение 3) и -Ofast (см. Приложение 3).

# **ЗАКЛЮЧЕНИЕ**

Из приведенных описаний листингов с оптимизациями -O0, -O1 и -Ofast можно сделать выводы об особенностях этих оптимизаций.

Про оптимизацию -O0 можем сказать, что каждому оператору из исходного кода на Си можно чётко поставить в соответствие набор команд из ассемеблерного листинга. Из недостатков оптимизации -O0 следует отметить, что компилятор делает много лишних действий, потому что компилятор рассматривает выражение из исходного кода независимо от сделанных им ранее действий.

Про оптимизацию -Ofast можем сказать, что листинг программы с данной оптимизацией разбирать сложнее, потому что нельзя провести однозначного соответствия между ассемблерным кодом и кодом исходной программы.

**ПРИЛОЖЕНИЕ 1.** Листинг программы с библиотечной функцией clock\_gettime на языке Си

#define \_POSIX\_C\_SOURCE 199309L

#include <time.h>

#include <stdio.h>

#include <stdlib.h>

#define PI 3.1415926535897

double CalcSin(double x, long long n);

int main(int argc, char \*\*argv){

if (argc < 3) {

printf("Bad input! Few arguments. Enter x and n in command line.");

return -1;

}

if (argc > 3){

printf("Bad input! A lot of arguments. Enter x and n in command line.");

return -1;

}

struct timespec res, start, end;

if (clock\_getres(CLOCK\_MONOTONIC\_RAW, &res) == 0){ // сравнивается с 0, так как 0 значит успешное выполнение

printf("Timer resolution: %ld sec, %ld nanosec.\n", res.tv\_sec, res.tv\_nsec);

} else {

perror("Call error clock\_getres!");

}

char \*endptr\_x, \*endptr\_n;

long x = strtol(argv[1], &endptr\_x, 10);

long long n = strtol(argv[2], &endptr\_n, 10);

if (\*endptr\_x != '\0'){

printf("Error: Invalid input for x: %s\n", argv[1]);

return -1;

}

if (\*endptr\_n != '\0'){

printf("Error: Invalid input for n: %s\n", argv[2]);

return -1;

}

clock\_gettime (CLOCK\_MONOTONIC\_RAW, &start);

double sinx = CalcSin((double)x, n);

clock\_gettime(CLOCK\_MONOTONIC\_RAW, &end);

printf("%lf\n", sinx);

printf("Time takken: %lf sec.\n", end.tv\_sec - start.tv\_sec + 0.000000001\*(end.tv\_nsec-start.tv\_nsec));

return 0;

}

double CalcSin(double x, long long n){

double sinx = 0;

x = x \* PI / 180; // перевод градусы в радианы, иначе некорректные вычисления

double sum = x;

for (long long i = 1; i <= 2 \* n - 1; i += 2){

sinx += sum;

sum = (sum \* x \* x \* (-1)) / ((i + 1) \* (i + 2));

}

return sinx;

}

**ПРИЛОЖЕНИЕ 2.** Ассемблерный листинг кода с оптимизацией -O0 (x86\_64 gcc 14.2)

.LC0:

.string "Bad input! Few arguments. Enter x and n in command line."

.LC1:

.string "Bad input! A lot of arguments. Enter x and n in command line."

.LC2:

.string "Timer resolution: %ld sec, %ld nanosec.\n"

.LC3:

.string "Call error clock\_getres!"

.LC4:

.string "Error: Invalid input for x: %s\n"

.LC5:

.string "Error: Invalid input for n: %s\n"

.LC6:

.string "%lf\n"

.LC8:

.string "Time takken: %lf sec.\n"

main:

push rbp

mov rbp, rsp

sub rsp, 112

mov DWORD PTR [rbp-100], edi

mov QWORD PTR [rbp-112], rsi

cmp DWORD PTR [rbp-100], 2

jg .L2

mov edi, OFFSET FLAT:.LC0

mov eax, 0

call printf

mov eax, -1

jmp .L9

.L2:

cmp DWORD PTR [rbp-100], 3

jle .L4

mov edi, OFFSET FLAT:.LC1

mov eax, 0

call printf

mov eax, -1

jmp .L9

.L4:

lea rax, [rbp-48]

mov rsi, rax

mov edi, 4

call clock\_getres

test eax, eax

jne .L5

mov rdx, QWORD PTR [rbp-40]

mov rax, QWORD PTR [rbp-48]

mov rsi, rax

mov edi, OFFSET FLAT:.LC2

mov eax, 0

call printf

jmp .L6

.L5:

mov edi, OFFSET FLAT:.LC3

call perror

.L6:

mov rax, QWORD PTR [rbp-112]

add rax, 8

mov rax, QWORD PTR [rax]

lea rcx, [rbp-88]

mov edx, 10

mov rsi, rcx

mov rdi, rax

call strtol

mov QWORD PTR [rbp-8], rax

mov rax, QWORD PTR [rbp-112]

add rax, 16

mov rax, QWORD PTR [rax]

lea rcx, [rbp-96]

mov edx, 10

mov rsi, rcx

mov rdi, rax

call strtol

mov QWORD PTR [rbp-16], rax

mov rax, QWORD PTR [rbp-88]

movzx eax, BYTE PTR [rax]

test al, al

je .L7

mov rax, QWORD PTR [rbp-112]

add rax, 8

mov rax, QWORD PTR [rax]

mov rsi, rax

mov edi, OFFSET FLAT:.LC4

mov eax, 0

call printf

mov eax, -1

jmp .L9

.L7:

mov rax, QWORD PTR [rbp-96]

movzx eax, BYTE PTR [rax]

test al, al

je .L8

mov rax, QWORD PTR [rbp-112]

add rax, 16

mov rax, QWORD PTR [rax]

mov rsi, rax

mov edi, OFFSET FLAT:.LC5

mov eax, 0

call printf

mov eax, -1

jmp .L9

.L8:

lea rax, [rbp-64]

mov rsi, rax

mov edi, 4

call clock\_gettime

pxor xmm3, xmm3

cvtsi2sd xmm3, QWORD PTR [rbp-8]

movq rax, xmm3

mov rdx, QWORD PTR [rbp-16]

mov rdi, rdx

movq xmm0, rax

call CalcSin

movq rax, xmm0

mov QWORD PTR [rbp-24], rax

lea rax, [rbp-80]

mov rsi, rax

mov edi, 4

call clock\_gettime

mov rax, QWORD PTR [rbp-24]

movq xmm0, rax

mov edi, OFFSET FLAT:.LC6

mov eax, 1

call printf

mov rdx, QWORD PTR [rbp-80]

mov rax, QWORD PTR [rbp-64]

sub rdx, rax

pxor xmm1, xmm1

cvtsi2sd xmm1, rdx

mov rdx, QWORD PTR [rbp-72]

mov rax, QWORD PTR [rbp-56]

sub rdx, rax

pxor xmm2, xmm2

cvtsi2sd xmm2, rdx

movsd xmm0, QWORD PTR .LC7[rip]

mulsd xmm0, xmm2

addsd xmm1, xmm0

movq rax, xmm1

movq xmm0, rax

mov edi, OFFSET FLAT:.LC8

mov eax, 1

call printf

mov eax, 0

.L9:

leave

ret

CalcSin:

push rbp

mov rbp, rsp

movsd QWORD PTR [rbp-40], xmm0

mov QWORD PTR [rbp-48], rdi

pxor xmm0, xmm0

movsd QWORD PTR [rbp-8], xmm0

movsd xmm1, QWORD PTR [rbp-40]

movsd xmm0, QWORD PTR .LC10[rip]

mulsd xmm0, xmm1

movsd xmm1, QWORD PTR .LC11[rip]

divsd xmm0, xmm1

movsd QWORD PTR [rbp-40], xmm0

movsd xmm0, QWORD PTR [rbp-40]

movsd QWORD PTR [rbp-16], xmm0

mov QWORD PTR [rbp-24], 1

jmp .L11

.L12:

movsd xmm0, QWORD PTR [rbp-8]

addsd xmm0, QWORD PTR [rbp-16]

movsd QWORD PTR [rbp-8], xmm0

movsd xmm0, QWORD PTR [rbp-16]

mulsd xmm0, QWORD PTR [rbp-40]

mulsd xmm0, QWORD PTR [rbp-40]

movq xmm1, QWORD PTR .LC12[rip]

xorpd xmm0, xmm1

mov rax, QWORD PTR [rbp-24]

lea rdx, [rax+1]

mov rax, QWORD PTR [rbp-24]

add rax, 2

imul rax, rdx

pxor xmm1, xmm1

cvtsi2sd xmm1, rax

divsd xmm0, xmm1

movsd QWORD PTR [rbp-16], xmm0

add QWORD PTR [rbp-24], 2

.L11:

mov rax, QWORD PTR [rbp-48]

add rax, rax

cmp QWORD PTR [rbp-24], rax

jl .L12

movsd xmm0, QWORD PTR [rbp-8]

pop rbp

ret

.LC7:

.long -400107883

.long 1041313291

.LC10:

.long 1413753926

.long 1074340347

.LC11:

.long 0

.long 1080459264

.LC12:

.long 0

.long -2147483648

.long 0

.long 0

**ПРИЛОЖЕНИЕ 3.** Ассемблерный листинг с оптимизацией -O1

CalcSin:

mulsd xmm0, QWORD PTR .LC1[rip]

divsd xmm0, QWORD PTR .LC2[rip]

add rdi, rdi

cmp rdi, 1

jle .L4

add rdi, 1

movapd xmm1, xmm0

mov eax, 1

pxor xmm2, xmm2

movq xmm4, QWORD PTR .LC3[rip]

.L3:

addsd xmm2, xmm1

mulsd xmm1, xmm0

mulsd xmm1, xmm0

xorpd xmm1, xmm4

lea rdx, [rax+1]

add rax, 2

imul rdx, rax

pxor xmm3, xmm3

cvtsi2sd xmm3, rdx

divsd xmm1, xmm3

cmp rax, rdi

jne .L3

.L1:

movapd xmm0, xmm2

ret

.L4:

pxor xmm2, xmm2

jmp .L1

.LC4:

.string "Bad input! Few arguments. Enter x and n in command line."

.LC5:

.string "Bad input! A lot of arguments. Enter x and n in command line."

.LC6:

.string "Timer resolution: %ld sec, %ld nanosec.\n"

.LC7:

.string "Call error clock\_getres!"

.LC8:

.string "Error: Invalid input for x: %s\n"

.LC9:

.string "Error: Invalid input for n: %s\n"

.LC10:

.string "%lf\n"

.LC12:

.string "Time takken: %lf sec.\n"

main:

push r12

push rbp

push rbx

sub rsp, 80

cmp edi, 2

jle .L15

mov rbx, rsi

cmp edi, 3

jg .L16

lea rsi, [rsp+64]

mov edi, 4

call clock\_getres

test eax, eax

jne .L10

mov rdx, QWORD PTR [rsp+72]

mov rsi, QWORD PTR [rsp+64]

mov edi, OFFSET FLAT:.LC6

call printf

.L11:

lea rsi, [rsp+24]

mov rdi, QWORD PTR [rbx+8]

mov edx, 10

call strtol

mov r12, rax

lea rsi, [rsp+16]

mov rdi, QWORD PTR [rbx+16]

mov edx, 10

call strtol

mov rbp, rax

mov rax, QWORD PTR [rsp+24]

cmp BYTE PTR [rax], 0

jne .L17

mov rax, QWORD PTR [rsp+16]

cmp BYTE PTR [rax], 0

jne .L18

lea rsi, [rsp+48]

mov edi, 4

call clock\_gettime

pxor xmm0, xmm0

cvtsi2sd xmm0, r12

mov rdi, rbp

call CalcSin

movsd QWORD PTR [rsp+8], xmm0

lea rsi, [rsp+32]

mov edi, 4

call clock\_gettime

movsd xmm0, QWORD PTR [rsp+8]

mov edi, OFFSET FLAT:.LC10

mov eax, 1

call printf

mov rax, QWORD PTR [rsp+40]

sub rax, QWORD PTR [rsp+56]

pxor xmm0, xmm0

cvtsi2sd xmm0, rax

mulsd xmm0, QWORD PTR .LC11[rip]

mov rax, QWORD PTR [rsp+32]

sub rax, QWORD PTR [rsp+48]

pxor xmm1, xmm1

cvtsi2sd xmm1, rax

addsd xmm0, xmm1

mov edi, OFFSET FLAT:.LC12

mov eax, 1

call printf

mov eax, 0

.L6:

add rsp, 80

pop rbx

pop rbp

pop r12

ret

.L15:

mov edi, OFFSET FLAT:.LC4

mov eax, 0

call printf

mov eax, -1

jmp .L6

.L16:

mov edi, OFFSET FLAT:.LC5

mov eax, 0

call printf

mov eax, -1

jmp .L6

.L10:

mov edi, OFFSET FLAT:.LC7

call perror

jmp .L11

.L17:

mov rsi, QWORD PTR [rbx+8]

mov edi, OFFSET FLAT:.LC8

mov eax, 0

call printf

mov eax, -1

jmp .L6

.L18:

mov rsi, QWORD PTR [rbx+16]

mov edi, OFFSET FLAT:.LC9

mov eax, 0

call printf

mov eax, -1

jmp .L6

.LC1:

.long 1413753926

.long 1074340347

.LC2:

.long 0

.long 1080459264

.LC3:

.long 0

.long -2147483648

.long 0

.long 0

.LC11:

.long -400107883

.long 1041313291

**ПРИЛОЖЕНИЕ 4.** Ассемблерный листинг с самой быстрой у меня оптимизацией -Ofast

.LC1:

.string "Bad input! Few arguments. Enter x and n in command line."

.LC2:

.string "Bad input! A lot of arguments. Enter x and n in command line."

.LC3:

.string "Timer resolution: %ld sec, %ld nanosec.\n"

.LC4:

.string "Call error clock\_getres!"

.LC5:

.string "Error: Invalid input for x: %s\n"

.LC6:

.string "Error: Invalid input for n: %s\n"

.LC9:

.string "%lf\n"

.LC11:

.string "Time takken: %lf sec.\n"

main:

push r12

push rbp

push rbx

sub rsp, 80

cmp edi, 2

jle .L16

cmp edi, 3

jne .L17

mov rbp, rsi

mov edi, 4

lea rsi, [rsp+32]

call clock\_getres

test eax, eax

je .L18

mov edi, OFFSET FLAT:.LC4

call perror

.L7:

mov rdi, QWORD PTR [rbp+8]

lea rsi, [rsp+16]

mov edx, 10

call strtol

mov rdi, QWORD PTR [rbp+16]

lea rsi, [rsp+24]

mov edx, 10

mov r12, rax

call strtol

mov rbx, rax

mov rax, QWORD PTR [rsp+16]

cmp BYTE PTR [rax], 0

jne .L19

mov rax, QWORD PTR [rsp+24]

cmp BYTE PTR [rax], 0

jne .L20

lea rsi, [rsp+48]

mov edi, 4

call clock\_gettime

pxor xmm0, xmm0

lea rcx, [rbx+rbx]

cvtsi2sd xmm0, r12

mulsd xmm0, QWORD PTR .LC7[rip]

cmp rcx, 1

jle .L12

movapd xmm4, xmm0

add rcx, 1

mov eax, 1

movq xmm3, QWORD PTR .LC8[rip]

mulsd xmm4, xmm0

pxor xmm1, xmm1

.L11:

lea rdx, [rax+1]

addsd xmm1, xmm0

mulsd xmm0, xmm4

add rax, 2

imul rdx, rax

pxor xmm2, xmm2

cvtsi2sd xmm2, rdx

xorpd xmm0, xmm3

divsd xmm0, xmm2

cmp rax, rcx

jne .L11

.L10:

lea rsi, [rsp+64]

mov edi, 4

movsd QWORD PTR [rsp+8], xmm1

call clock\_gettime

movsd xmm1, QWORD PTR [rsp+8]

mov edi, OFFSET FLAT:.LC9

mov eax, 1

movapd xmm0, xmm1

call printf

mov rax, QWORD PTR [rsp+72]

pxor xmm0, xmm0

sub rax, QWORD PTR [rsp+56]

cvtsi2sd xmm0, rax

pxor xmm1, xmm1

mov rax, QWORD PTR [rsp+64]

sub rax, QWORD PTR [rsp+48]

mulsd xmm0, QWORD PTR .LC10[rip]

cvtsi2sd xmm1, rax

mov edi, OFFSET FLAT:.LC11

mov eax, 1

addsd xmm0, xmm1

call printf

xor eax, eax

.L1:

add rsp, 80

pop rbx

pop rbp

pop r12

ret

.L18:

mov rdx, QWORD PTR [rsp+40]

mov rsi, QWORD PTR [rsp+32]

mov edi, OFFSET FLAT:.LC3

call printf

jmp .L7

.L12:

pxor xmm1, xmm1

jmp .L10

.L17:

mov edi, OFFSET FLAT:.LC2

xor eax, eax

call printf

.L3:

or eax, -1

jmp .L1

.L20:

mov rsi, QWORD PTR [rbp+16]

mov edi, OFFSET FLAT:.LC6

xor eax, eax

call printf

jmp .L3

.L19:

mov rsi, QWORD PTR [rbp+8]

mov edi, OFFSET FLAT:.LC5

xor eax, eax

call printf

jmp .L3

.L16:

mov edi, OFFSET FLAT:.LC1

xor eax, eax

call printf

jmp .L3

CalcSin:

mulsd xmm0, QWORD PTR .LC7[rip]

add rdi, rdi

cmp rdi, 1

jle .L24

movapd xmm4, xmm0

add rdi, 1

mov eax, 1

movq xmm3, QWORD PTR .LC8[rip]

mulsd xmm4, xmm0

pxor xmm1, xmm1

.L23:

lea rdx, [rax+1]

addsd xmm1, xmm0

mulsd xmm0, xmm4

add rax, 2

imul rdx, rax

pxor xmm2, xmm2

cvtsi2sd xmm2, rdx

xorpd xmm0, xmm3

divsd xmm0, xmm2

cmp rax, rdi

jne .L23

movapd xmm0, xmm1

ret

.L24:

pxor xmm1, xmm1

movapd xmm0, xmm1

ret

.LC7:

.long -1571644252

.long 1066524486

.LC8:

.long 0

.long -2147483648

.long 0

.long 0

.LC10:

.long -400107883

.long 1041313291

Ссылка на репозиторий с кодом: <https://github.com/dadashasha/nsu_evm/tree/main/lab3>