###### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

###### Факультет информационных технологий

**Кафедра параллельных вычислений**

ОТЧЕТ

О ВЫПОЛНЕНИИ ПРАКТИЧЕСКОЙ РАБОТЫ

ВВЕДЕНИЕ В АРХИТЕКТУРУ x86/x86-64

студента 2 курса, группы 22204

Соломенникова Николая Александровича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

доцент

А.Ю.Власенко

Новосибирск 2023

**СОДЕРЖАНИЕ**

[ЦЕЛИ 2](#__RefHeading___21)

[ЗАДАНИЕ 2](#__RefHeading___22)

[ОПИСАНИЕ РАБОТЫ 3](#__RefHeading___23)

[ЗАКЛЮЧЕНИЕ 5](#__RefHeading___24)

[Приложение 1 6](#__RefHeading___25)

[Приложение 2 7](#__RefHeading___26)

[Приложение 3 10](#__RefHeading___27)

# ЦЕЛИ

1. Знакомство с программной архитектурой x86/x86-64.

2. Анализ ассемблерного листинга программы для архитектуры x86/x86-64.

# ЗАДАНИЕ

1. Для программы на языке Си (реализовать свою функцию sin(x) через разложение в ряд) сгенерировать ассемблерные листинги для архитектуры x86-64, используя уровни оптимизации O0 и Ofast.
2. Проанализировать полученные листинги и сделать следующее:
   * сопоставить команды языка Си с машинными командами;
   * определить размещение переменных языка Си в программах на ассемблере (в каких регистрах, в каких ячейках памяти);
   * выписать оптимизационные преобразования, выполненные компилятором;

# ОПИСАНИЕ РАБОТЫ

Соответствие команд языка Си с машинными командами легче всего рассмотреть на сайте godbolt.

Размещение переменных в памяти при уровне компиляции -O0:

Стековый кадр функции main:

|  |  |
| --- | --- |
| rbp - 48 | argv |
| rbp - 36 | argc |
| rbp - 24 | res |
| rbp – 16 (.LC2) | angle |
| rbp - 8 | n |

Стековый кадр функции mySin:

|  |  |
| --- | --- |
| rbp - 48 | n |
| rbp - 40 | x |
| rbp - 24 | i |
| rbp - 16 | d |
| rbp - 8 | ans |

Разбор листинга ассемблерного кода с оптимизацией -Ofast:

mySin(double, long long):

        fldl    .LC0(%rip) // число 2пи кладётся в FPU в st (то же самое

что st(0))

        movsd   %xmm0, -8(%rsp) // 1.57 кладётся в стек

        fldl    -8(%rsp) // 1.57 кладётся в FPU в st(0), теперь 2пи в st(1)

.L2:

        fprem // st(0) %= st(1)

        fnstsw  %ax // сохранение значения слова состояния FPU

(SW, т.е. регистра флагов FPU) в ax

        testb   $4, %ah // Проверка флага путём побитового умножения

ah & 00000100 (в двоичной)

        jne     .L2 // Переход

        fstp    %st(1) // st(1) = st, верхний регистр (st) удаляется

        fstpl   -8(%rsp) // -8(%rsp) = st = x % 2пи = 1.57

        movsd   -8(%rsp), %xmm0 // xmm0 = 1,57 (в коде это x)

        cmpq    $1, %rdi // Сравнение для возможного выхода из функции

        jle     .L5 // Переход на метку, если rdi <= 1, т.е. если n <= 1

        movapd  %xmm0, %xmm5 // xmm5 = x

        movq    .LC1(%rip), %xmm4 // xmm4 = константа для умножения на -1

        leaq    1(%rdi,%rdi), %rcx // rcx = 2n - 1

        movapd  %xmm0, %xmm2 // xmm2 = xmm0 = x (в коде это ans)

        mulsd   %xmm0, %xmm5 // xmm5 \*= xmm0 = x\*x

        movl    $3, %eax // eax = 3 (в коде это 2i - 1)

.L4:

        leaq    -1(%rax), %rdx // rdx = 2i - 2

        pxor    %xmm1, %xmm1 // xmm1 = 0

        pxor    %xmm3, %xmm3 // xmm3 = 0

        cvtsi2sdq       %rax, %xmm3 // xmm3 = 2i - 1

        mulsd   %xmm5, %xmm0 // xmm0 \*= xmm5 (на первой итерации будет x\*x\*x)

        addq    $2, %rax // i++

        cvtsi2sdq       %rdx, %xmm1 // xmm1 = 2i - 2

        mulsd   %xmm3, %xmm1 // xmm1 \*= xmm3 = (2i – 1) \* (2i - 2)

        xorpd   %xmm4, %xmm0 // xmm0 \*= -1

        divsd   %xmm1, %xmm0 // xmm0 /= xmm1

        addsd   %xmm0, %xmm2 // xmm2 += xmm0 (ans += d)

        cmpq    %rax, %rcx

        jne     .L4 // Цикл повторяется, если i != 2n + 1

        movapd  %xmm2, %xmm0 // xmm0 = ans

        ret // Возврат в main

.L5:

        movapd  %xmm0, %xmm2 // ans = x

        movapd  %xmm2, %xmm0

        ret // Возврат в main

main:

        subq    $8, %rsp // Инициализация стекового кадра

        movq    8(%rsi), %rdi // rdi = адрес аргумента argv[1]

        movl    $10, %edx // Это нигде явно не используется, но возможно это

надо для strtoll (например как длина строки)

        xorl    %esi, %esi // esi = 0

        call    strtoll // Преобразование argv[1] из строки в long long

(результат кладётся в %rax, результат это число n в исходном коде)

        movsd   .LC3(%rip), %xmm0 // xmm0 = 1.57 (angle)

        movq    %rax, %rdi // rdi = n

        call    mySin(double, long long) // Вызов с аргументами rdi = n,

xmm0 = 1.57

        movl    $\_ZSt4cout, %edi // Для доступа к объекту cout

    call std::basic\_ostream<char, std::char\_traits<char> >&

std::basic\_ostream<char, std::char\_traits<char>

>::\_M\_insert<double>(double) // Вывод res

        xorl    %eax, %eax // eax = 0

        addq    $8, %rsp // возврат rsp в исходное положение

        ret // конец программы

.LC0: // 2пи

        .long   1413754136

        .long   1075388923

.LC1: // константа для домножения на -1

        .long   0

        .long   -2147483648

        .long   0

        .long   0

.LC3: // 1.57

        .long   1374389535

        .long   1073290936

Оптимизационные преобразования:

1. При уровне компиляции -O3 или -Ofast компилятор пытается больше использовать регистры а не стек для хранения переменных. Это нужно чтобы меньше обращаться в память.
2. Убираются лишние перекладывания
3. Оптимизируются циклы и операции с переменной i.
4. Сравнение <= заменено на !=
5. Умножение, которое лишний раз делалось в цикле, вынесено за цикл
6. Умножение на -1 заменено на xor с константой

# ЗАКЛЮЧЕНИЕ

При уровне оптимизации -O0 ассемблерный код точно соответствует строкам исходного кода, поэтому легко сопоставить команды языка Си с машинными командами. При уровне -Ofast сильно меняется логика кода, и происходит много оптимизаций.

# Приложение 1

Исходный код:

#include <iostream>

#include <math.h>

double mySin(double x, long long n)

{

    x = fmod(x, 2 \* M\_PI);

    double ans = x;

    double d = x;

    for (long long i = 2; i <= n; i++)

    {

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

        ans += d;

    }

    return ans;

}

int main(int args, char \*argv[])

{

    long long n = atoll(argv[1]);

    double angle = 1.57; // for example

    double res = mySin(angle, n);

    std::cout << res;

    return 0;

}

# Приложение 2

Ассемблерный листинг (O0):

mySin(double, long long):

        pushq   %rbp

        movq    %rsp, %rbp

        subq    $48, %rsp

        movsd   %xmm0, -40(%rbp)

        movq    %rdi, -48(%rbp)

        movsd   .LC0(%rip), %xmm0

        movq    -40(%rbp), %rax

        movapd  %xmm0, %xmm1

        movq    %rax, %xmm0

        call    fmod

        movq    %xmm0, %rax

        movq    %rax, -40(%rbp)

        movsd   -40(%rbp), %xmm0

        movsd   %xmm0, -8(%rbp)

        movsd   -40(%rbp), %xmm0

        movsd   %xmm0, -16(%rbp)

        movq    $2, -24(%rbp)

        jmp     .L2

.L3:

        movsd   -16(%rbp), %xmm0

        movq    .LC1(%rip), %xmm1

        xorpd   %xmm1, %xmm0

        mulsd   -40(%rbp), %xmm0

        mulsd   -40(%rbp), %xmm0

        movq    -24(%rbp), %rax

        addq    %rax, %rax

        subq    $1, %rax

        pxor    %xmm1, %xmm1

        cvtsi2sdq       %rax, %xmm1

        divsd   %xmm1, %xmm0

        movq    -24(%rbp), %rax

        subq    $1, %rax

        addq    %rax, %rax

        pxor    %xmm1, %xmm1

        cvtsi2sdq       %rax, %xmm1

        divsd   %xmm1, %xmm0

        movsd   %xmm0, -16(%rbp)

        movsd   -8(%rbp), %xmm0

        addsd   -16(%rbp), %xmm0

        movsd   %xmm0, -8(%rbp)

        addq    $1, -24(%rbp)

.L2:

        movq    -24(%rbp), %rax

        cmpq    -48(%rbp), %rax

        jle     .L3

        movsd   -8(%rbp), %xmm0

        movq    %xmm0, %rax

        movq    %rax, %xmm0

        leave

        ret

main:

        pushq   %rbp

        movq    %rsp, %rbp

        subq    $48, %rsp

        movl    %edi, -36(%rbp)

        movq    %rsi, -48(%rbp)

        movq    -48(%rbp), %rax

        addq    $8, %rax

        movq    (%rax), %rax

        movq    %rax, %rdi

        call    atoll

        movq    %rax, -8(%rbp)

        movsd   .LC2(%rip), %xmm0

        movsd   %xmm0, -16(%rbp)

        movq    -8(%rbp), %rdx

        movq    -16(%rbp), %rax

        movq    %rdx, %rdi

        movq    %rax, %xmm0

        call    mySin(double, long long)

        movq    %xmm0, %rax

        movq    %rax, -24(%rbp)

        movq    -24(%rbp), %rax

        movq    %rax, %xmm0

        movl    $\_ZSt4cout, %edi

call    std::basic\_ostream<char, std::char\_traits<char> >::

operator<<(double)

        movl    $0, %eax

        leave

        ret

.LC0:

        .long   1413754136

        .long   1075388923

.LC1:

        .long   0

        .long   -2147483648

        .long   0

        .long   0

.LC2:

        .long   1374389535

        .long   1073290936

# Приложение 3

Ассемблерный листинг (Ofast):

mySin(double, long long):

        fldl    .LC0(%rip)

        movsd   %xmm0, -8(%rsp)

        fldl    -8(%rsp)

.L2:

        fprem

        fnstsw  %ax

        testb   $4, %ah

        jne     .L2

        fstp    %st(1)

        fstpl   -8(%rsp)

        movsd   -8(%rsp), %xmm0

        cmpq    $1, %rdi

        jle     .L5

        movapd  %xmm0, %xmm5

        movq    .LC1(%rip), %xmm4

        leaq    1(%rdi,%rdi), %rcx

        movapd  %xmm0, %xmm2

        mulsd   %xmm0, %xmm5

        movl    $3, %eax

.L4:

        leaq    -1(%rax), %rdx

        pxor    %xmm1, %xmm1

        pxor    %xmm3, %xmm3

        cvtsi2sdq       %rax, %xmm3

        mulsd   %xmm5, %xmm0

        addq    $2, %rax

        cvtsi2sdq       %rdx, %xmm1

        mulsd   %xmm3, %xmm1

        xorpd   %xmm4, %xmm0

        divsd   %xmm1, %xmm0

        addsd   %xmm0, %xmm2

        cmpq    %rax, %rcx

        jne     .L4

        movapd  %xmm2, %xmm0

        ret

.L5:

        movapd  %xmm0, %xmm2

        movapd  %xmm2, %xmm0

        ret

main:

        subq    $8, %rsp

        movq    8(%rsi), %rdi

        movl    $10, %edx

        xorl    %esi, %esi

        call    strtoll

        movsd   .LC3(%rip), %xmm0

        movq    %rax, %rdi

        call    mySin(double, long long)

        movl    $\_ZSt4cout, %edi

        call    std::basic\_ostream<char, std::char\_traits<char> >& std::basic\_ostream<char, std::char\_traits<char> >::\_M\_insert<double>(double)

        xorl    %eax, %eax

        addq    $8, %rsp

        ret

.LC0:

        .long   1413754136

        .long   1075388923

.LC1:

        .long   0

        .long   -2147483648

        .long   0

        .long   0

.LC3:

        .long   1374389535

        .long   1073290936