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

Практическая работа №3

ВВЕДЕНИЕ В АРХИТЕКТУРУ x86/x86-64

студента 2 курса, группы 23201

Сорокина Матвея Павловича

Направление 09.03.01 – "Информатика и вычислительная техника"

Преподаватель: А.С. Матвеев

Новосибирск, 2024 г.

Содержание

1 ЦЕЛЬ

Знакомство с программной архитектурой x86/x86-64 и анализ ассемблерного листинга программы для архитектуры x86/x86-64.

2 ЗАДАНИЕ

В ходе работы было необходимо выполнить следующие задачи:

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. Для программы на языке Си (из лабораторной работы 1) сгенерировать ассемблерные листинги для архитектуры x86 и архитектуры x86-64, используя различные уровни комплексной оптимизации.
3. Проанализировать полученные листинги и сделать следующее:
    - Сопоставьте команды языка Си с машинными командами.
    - Определить размещение переменных языка Си в программах на ассемблере (в каких регистрах, в каких ячейках памяти).
    - Описать и объяснить оптимизационные преобразования, выполненные компилятором.
    - Продемонстрировать использование ключевых особенностей архитектур x86 и x86-64 на конкретных участках ассемблерного кода.
    - Сравнить различия в программах для архитектуры x86 и архитектуры x86-64.
4. Составить отчет, отражающий этапы работы, результаты анализа, выводы.

3 ОПИСАНИЕ РАБОТЫ

1. Реализовано задание №4 - алгоритм вычисления синуса с помощью разложения в степенной ряд по превым N членам данного ряда на языке *C++*. Код программы предоставлен (см. Приложение 1).

   Для измерения времени работы программы использовалась библиотечная функция *clock_ gettime* из библиотеки *time.h*. Время замерялось перед началом и после окончания работы функции, вычисляющей синус заданного угла.
   Разность этих двух значений дает общее время выполнения функции. Для проверки точности измерений, код программы запускается несколько раз.

2. Ассемблерные листинги с различными уровнями оптимизации генерировались с помощью:

   - Компилятора:

g++ -m64 -S -o res_x86-64 SinCalculation.cpp
g++ -m32 -S -o res_x86 SinCalculation.cpp

     -m64: Указывает компилятору использовать 64-битную архитектуру. Следовательно на выход получим файл с 64-битным ассемблерным листингом ($x86-64$).
     -m32: Указывает компилятору использовать 32-битную архитектуру.

   - Сайта: godbolt.org

3. Оптимизация кода в ассемблерном листинге программы включает в себя:

   - ***Управление операций с памятью***

     Без оптимизации
     С помощью команд ***push, mov, pop*** создается *Stack pointer register* (rsp), также создается и при выходе из функции удаляется *Base pointer register* (rbp), следовательно из-за дополнительных операций по выделению памяти выполнение кода более медленное:

```
DegToRad(long double):
    push rbp
    mov rbp, rsp
    fld TBYTE PTR [rbp+16]
    fld TBYTE PTR .LC0[rip]
    fmulp st(1), st
    fld TBYTE PTR .LC1[rip]
    fdivp st(1), st
    pop rbp
    ret
```

     С оптимизацией
     Оптимизированный код использует *Stack pointer register* (rsp) напрямую:

```
DegToRad(long double):
    fld TBYTE PTR [rsp+8]
    fmul QWORD PTR .LC1[rip]
    fdiv DWORD PTR .LC2[rip]
    ret
```

- ***Упрощение операций в FPU стеках***

  Без оптимизации
  Инструкция ***fld TBYTE PTR [rbp+16]*** загружает 80-битное вещественное число (тип *long double* в *C++*) из памяти по адресу *rbp+16* в вершину *FPU* стека. Загружается 80-битное вещественное число, поскольку тип операнда - *TBYTE PTR*, что обозначает *Ten-byte* указатель.

```
SinCalculation(long double, long long):
    push rbp
    mov rbp, rsp
    mov QWORD PTR [rbp-72], rdi
    fldz
    fstp TBYTE PTR [rbp-16]
    fld TBYTE PTR [rbp+16]
    fstp TBYTE PTR [rbp-32]
    fld1
    fstp TBYTE PTR [rbp-48]
    mov QWORD PTR [rbp-56], 1
    jmp .L4
```

  С оптимизацией
  Вместо последовательных ***fld*** и ***fstp***, оптимизированная версия сразу загружает значение из стека *rsp+8* в регистр FPU стека с помощью инструкции ***fld TBYTE PTR [rbp+8]***, минуя промежуточное сохранение в локальные переменные:

```
SinCalculation(long double, long long):
    fld TBYTE PTR [rsp+8]
    cmp rdi, 1
    jle .L15
    fld st(0)
    mov ecx, 3
    mov eax, 1
    fmul st, st(1)
    fld1
    fldz
    jmp .L14
```

  Когда используется *rbp* как базовый указатель, компилятор требует дополнительных инструкций ***push*** и ***pop***, которые могут немного замедлять выполнение кода.
  В оптимизированных версиях используется стековый указатель *rsp*. Это сокращает количество инструкций и упрощает код.

4. Взаимодействие FPU стека и стека вызовов на примере функции ***SinCalculation(long double x, long long n)***.

   Стек вызовов - фрейм стека вызовов создается при вызове функции, где размещаются локальные переменные и сохраненные регистры.

   FPU стек — это отдельная структура данных, используемая сопроцессором для

работы с числами с плавающей запятой. Он содержит восемь регистров st(0) до st(7), организованных как стек *(Last In, First Out)*.

Когда в ассемблерном листинге используются команды FPU, такие как: ***fld***, ***fstp***, данные загружаются из обычного стека вызовов в FPU стек и обратно:

- инструкция ***fld TBYTE PTR [rbp-48]*** загружает значение переменной sign из стека вызовов (обычный стек) в вершину FPU стека (st(0)).
- инструкция ***fstp TBYTE PTR [rbp-16]*** выгружает значение из вершины FPU стека (st(0)) обратно в обычный стек вызовов, где хранятся переменные (в данном случае, переменная sin).

5. Размещение переменных языка Си в программах на ассемблере на примере функции ***SinCalculation(long double x, long long n)***.

   В функции есть 4 переменные, каждая имеет свой адрес:

   - *long double sin* — начальное значение 0;
   - *long double prev* — начальное значение x;
   - *long double sign* — начальное значение 1.0;
   - *long long i* — переменная для итерации в цикле, начальное значение 1.

   После вызова функции создается стандартный стековый фрейм (стек вызовов), где размещаются локальные переменные:

   - *sin* хранится по адресу [rbp-16].
   - *prev* хранится по адресу [rbp-32].
   - *sign* хранится по адресу [rbp-48].
   - Переменная *i*, используемая в цикле, хранится по адресу [rbp-56].

   В блоке ***.L5*** (см. Приложение 3):

```
.L5:
    fld TBYTE PTR [rbp-48]
    fld TBYTE PTR [rbp-32]
    fmulp st(1), st
    fld TBYTE PTR [rbp-16]
    faddp st(1), st
    fstp TBYTE PTR [rbp-16]
    fld TBYTE PTR [rbp+16]
    fld st(0)
    fmulp st(1), st
    mov rax, QWORD PTR [rbp-56]
    add rax, rax
    add rax, 1
    imul rax, QWORD PTR [rbp-56]
    add rax, rax
    mov QWORD PTR [rbp-96], rax
    fild QWORD PTR [rbp-96]
    fdivp st(1), st
    fld TBYTE PTR [rbp-32]
```

```
fmulp st(1), st
fstp TBYTE PTR [rbp-32]
fld TBYTE PTR [rbp-48]
fchs
fstp TBYTE PTR [rbp-96]
mov rax, QWORD PTR [rbp-96]
mov edx, DWORD PTR [rbp-88]
mov QWORD PTR [rbp-48], rax
mov DWORD PTR [rbp-40], edx
add QWORD PTR [rbp-56], 1
```

- Загружается значение *sign* (адрес [rbp-48]) и *prev* (адрес [rbp-32]) в FPU стеки, выполняется умножение с последующим добавлением к переменной sin.
- Переменная *prev* умножается на квадрат x, делится на факториал и сохраняется обратно в стек.
- Знак переменной *sign* меняется на противоположный командой ***fchs***, сохраняется в стек к следующей итерации цикла.

По итогу:
Каждая переменная функции имеет своё уникальное смещение от регистра базового указателя, (*Base pointer register*) rbp, что позволяет ассемблерным инструкциям манипулировать ими же в процессе выполнения программы.

4 ЗАКЛЮЧЕНИЕ

В ходе данной лабораторной работы я познакомился с программной архитектурой x86/x86-64 и научился анализировать ассемблерный листинг программы для архитектуры x86/x86-64.

Без оптимизации компилятор стремится сохранить структуру исходного кода на языке программирования, что делает ассемблерный код более понятным, но менее эффективным в плане выполнения.

С оптимизацией компилятор активно использует доступные регистры для временного хранения и обработки данных, что существенно ускоряет выполнение программы.

Упрощение циклов, удаление лишних операций и более эффективное использование регистров и памяти приводят к более эффективному выполнению программы.

5 ПРИЛОЖЕНИЯ

Приложение 1: Исходный код программы *SinCalculation.cpp*

```
#include <iostream>
#include <time.h>
#include <cstdlib> // for atoi and atof
#include <cmath> // for pow

long double DegToRad(long double deg) {
    return deg * M_PI / 180;
}

long double SinCalculation(long double x, long long n) {
    long double sin = 0;
    long double prev = x;
    long double sign = 1.0;

    for (long long i = 1; i < n; i++) {
        sin += sign * prev;
        prev *= (x * x) / ((2 * i) * (2 * i + 1));
        sign = -sign;
    }

    return sin;
}

int main(int argc, char *argv[]) {
    if (argc != 3) {
        std::cerr << "Usage: <angle in degrees> <number of terms>" << std::endl
            ↪ ;
        return 0;
    }

    struct timespec start, end;
    long double x = atoll(argv[1]);
    long long n = atoll(argv[2]);

    std::cout << "x = " << x << ", n = " << n << std::endl;

    int runs = 5;
    double time_total = 0;

    for (long long i = 0; i < runs; i++) {
        clock_gettime(CLOCK_MONOTONIC_RAW, &start);
        double rad_x = DegToRad(x);
        long double sin = SinCalculation(rad_x, n);
        clock_gettime(CLOCK_MONOTONIC_RAW, &end);
```

```
        double taken_time = (end.tv_sec - start.tv_sec) + (end.tv_nsec - start.
            ↪ tv_nsec) / 1e9;
        std::cout << "sin(" << x << ") = " << sin << std::endl;
        std::cout << "Run " << i + 1 << " took " << taken_time << " seconds to
            ↪ complete" << "\n" << std::endl;
        time_total += taken_time;
    }

    std::cout << "Average time: " << time_total / runs << " seconds" << std::
        ↪ endl;
    return 0;
}
```

Приложение 2: ассемблерный листинг программы *SinCalculation.cpp* с ключом *-O0*, архитектурой x86-64

```
    DegToRad(long double):
    push rbp
    mov rbp, rsp
    fld TBYTE PTR [rbp+16]
    fld TBYTE PTR .LC0[rip]
    fmulp st(1), st
    fld TBYTE PTR .LC1[rip]
    fdivp st(1), st
    pop rbp
    ret
SinCalculation(long double, long long):
    push rbp
    mov rbp, rsp
    mov QWORD PTR [rbp-72], rdi
    fldz
    fstp TBYTE PTR [rbp-16]
    fld TBYTE PTR [rbp+16]
    fstp TBYTE PTR [rbp-32]
    fld1
    fstp TBYTE PTR [rbp-48]
    mov QWORD PTR [rbp-56], 1
    jmp .L4
.L5:
    fld TBYTE PTR [rbp-48]
    fld TBYTE PTR [rbp-32]
    fmulp st(1), st
    fld TBYTE PTR [rbp-16]
    faddp st(1), st
    fstp TBYTE PTR [rbp-16]
    fld TBYTE PTR [rbp+16]
    fld st(0)
    fmulp st(1), st
    mov rax, QWORD PTR [rbp-56]
```

```
    add rax, rax
    add rax, 1
    imul rax, QWORD PTR [rbp-56]
    add rax, rax
    mov QWORD PTR [rbp-96], rax
    fild QWORD PTR [rbp-96]
    fdivp st(1), st
    fld TBYTE PTR [rbp-32]
    fmulp st(1), st
    fstp TBYTE PTR [rbp-32]
    fld TBYTE PTR [rbp-48]
    fchs
    fstp TBYTE PTR [rbp-96]
    mov rax, QWORD PTR [rbp-96]
    mov edx, DWORD PTR [rbp-88]
    mov QWORD PTR [rbp-48], rax
    mov DWORD PTR [rbp-40], edx
    add QWORD PTR [rbp-56], 1
.L4:
    mov rax, QWORD PTR [rbp-56]
    cmp rax, QWORD PTR [rbp-72]
    jl .L5
    fld TBYTE PTR [rbp-16]
    pop rbp
    ret
.LC5:
    .string "Usage: <angle in degrees> <number of terms>"
.LC6:
    .string "x = "
.LC7:
    .string ", n = "
.LC10:
    .string "sin("
.LC11:
    .string ") = "
.LC12:
    .string "Run \342\204\226"
.LC13:
    .string " took "
.LC14:
    .string " seconds to complete"
.LC15:
    .string "\n"
.LC16:
    .string "Average time: "
.LC17:
    .string " seconds"
main:
    push rbp
    mov rbp, rsp
```

```
    sub rsp, 160
    mov DWORD PTR [rbp-132], edi
    mov QWORD PTR [rbp-144], rsi
    cmp DWORD PTR [rbp-132], 3
    je .L8
    mov esi, OFFSET FLAT:.LC5
    mov edi, OFFSET FLAT:std::cerr
    call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
        ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
        ↪ char> >&, char const*)
    mov esi, OFFSET FLAT:std::basic_ostream<char, std::char_traits<char> >&
        ↪ std::endl<char, std::char_traits<char> >(std::basic_ostream<char,
        ↪ std::char_traits<char> >&)
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >::operator<<(std::
        ↪ basic_ostream<char, std::char_traits<char> >& (*)(std::basic_ostream
        ↪ <char, std::char_traits<char> >&))
    mov eax, 0
    jmp .L12
.L8:
    mov rax, QWORD PTR [rbp-144]
    add rax, 8
    mov rax, QWORD PTR [rax]
    mov rdi, rax
    call atoll
    mov QWORD PTR [rbp-152], rax
    fild QWORD PTR [rbp-152]
    fstp TBYTE PTR [rbp-32]
    mov rax, QWORD PTR [rbp-144]
    add rax, 16
    mov rax, QWORD PTR [rax]
    mov rdi, rax
    call atoll
    mov QWORD PTR [rbp-40], rax
    mov esi, OFFSET FLAT:.LC6
    mov edi, OFFSET FLAT:std::cout
    call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
        ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
        ↪ char> >&, char const*)
    push QWORD PTR [rbp-24]
    push QWORD PTR [rbp-32]
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >::operator<<(long
        ↪ double)
    add rsp, 16
    mov esi, OFFSET FLAT:.LC7
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
        ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
        ↪ char> >&, char const*)
```

```
    mov rdx, rax
    mov rax, QWORD PTR [rbp-40]
    mov rsi, rax
    mov rdi, rdx
    call std::basic_ostream<char, std::char_traits<char> >::operator<<(long
        ↪ long)
    mov esi, OFFSET FLAT:std::basic_ostream<char, std::char_traits<char> >&
        ↪ std::endl<char, std::char_traits<char> >(std::basic_ostream<char,
        ↪ std::char_traits<char> >&)
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >::operator<<(std::
        ↪ basic_ostream<char, std::char_traits<char> >& (*)(std::basic_ostream
        ↪ <char, std::char_traits<char> >&))
    mov DWORD PTR [rbp-44], 5
    pxor xmm0, xmm0
    movsd QWORD PTR [rbp-8], xmm0
    mov QWORD PTR [rbp-16], 0
    jmp .L10
.L11:
    lea rax, [rbp-112]
    mov rsi, rax
    mov edi, 4
    call clock_gettime
    push QWORD PTR [rbp-24]
    push QWORD PTR [rbp-32]
    call DegToRad(long double)
    add rsp, 16
    fstp QWORD PTR [rbp-56]
    fld QWORD PTR [rbp-56]
    mov rax, QWORD PTR [rbp-40]
    lea rsp, [rsp-16]
    fstp TBYTE PTR [rsp]
    mov rdi, rax
    call SinCalculation(long double, long long)
    add rsp, 16
    fstp TBYTE PTR [rbp-80]
    lea rax, [rbp-128]
    mov rsi, rax
    mov edi, 4
    call clock_gettime
    mov rdx, QWORD PTR [rbp-128]
    mov rax, QWORD PTR [rbp-112]
    sub rdx, rax
    pxor xmm1, xmm1
    cvtsi2sd xmm1, rdx
    mov rdx, QWORD PTR [rbp-120]
    mov rax, QWORD PTR [rbp-104]
    sub rdx, rax
    pxor xmm0, xmm0
    cvtsi2sd xmm0, rdx
```

```
movsd xmm2, QWORD PTR .LC9[rip]
divsd xmm0, xmm2
addsd xmm0, xmm1
movsd QWORD PTR [rbp-88], xmm0
mov esi, OFFSET FLAT:.LC10
mov edi, OFFSET FLAT:std::cout
call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
    ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
    ↪ char> >&, char const*)
push QWORD PTR [rbp-24]
push QWORD PTR [rbp-32]
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char> >::operator<<(long
    ↪ double)
add rsp, 16
mov esi, OFFSET FLAT:.LC11
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
    ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
    ↪ char> >&, char const*)
push QWORD PTR [rbp-72]
push QWORD PTR [rbp-80]
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char> >::operator<<(long
    ↪ double)
add rsp, 16
mov esi, OFFSET FLAT:std::basic_ostream<char, std::char_traits<char> >&
    ↪ std::endl<char, std::char_traits<char> >(std::basic_ostream<char,
    ↪ std::char_traits<char> >&)
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char> >::operator<<(std::
    ↪ basic_ostream<char, std::char_traits<char> >& (*)(std::basic_ostream
    ↪ <char, std::char_traits<char> >&))
mov esi, OFFSET FLAT:.LC12
mov edi, OFFSET FLAT:std::cout
call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
    ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
    ↪ char> >&, char const*)
mov rdx, rax
mov rax, QWORD PTR [rbp-16]
add rax, 1
mov rsi, rax
mov rdi, rdx
call std::basic_ostream<char, std::char_traits<char> >::operator<<(long
    ↪ long)
mov esi, OFFSET FLAT:.LC13
mov rdi, rax
call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
    ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
    ↪ char> >&, char const*)
```

```
    mov rdx, rax
    mov rax, QWORD PTR [rbp-88]
    movq xmm0, rax
    mov rdi, rdx
    call std::basic_ostream<char, std::char_traits<char> >::operator<<(double)
    mov esi, OFFSET FLAT:.LC14
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
       ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
       ↪ char> >&, char const*)
    mov esi, OFFSET FLAT:.LC15
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
       ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
       ↪ char> >&, char const*)
    mov esi, OFFSET FLAT:std::basic_ostream<char, std::char_traits<char> >&
       ↪ std::endl<char, std::char_traits<char> >(std::basic_ostream<char,
       ↪ std::char_traits<char> >&)
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >::operator<<(std::
       ↪ basic_ostream<char, std::char_traits<char> >& (*)(std::basic_ostream
       ↪ <char, std::char_traits<char> >&))
    movsd xmm0, QWORD PTR [rbp-8]
    addsd xmm0, QWORD PTR [rbp-88]
    movsd QWORD PTR [rbp-8], xmm0
    add QWORD PTR [rbp-16], 1
.L10:
    mov eax, DWORD PTR [rbp-44]
    cdqe
    cmp QWORD PTR [rbp-16], rax
    jl .L11
    mov esi, OFFSET FLAT:.LC16
    mov edi, OFFSET FLAT:std::cout
    call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
       ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
       ↪ char> >&, char const*)
    mov rdx, rax
    pxor xmm1, xmm1
    cvtsi2sd xmm1, DWORD PTR [rbp-44]
    movsd xmm0, QWORD PTR [rbp-8]
    divsd xmm0, xmm1
    movq rax, xmm0
    movq xmm0, rax
    mov rdi, rdx
    call std::basic_ostream<char, std::char_traits<char> >::operator<<(double)
    mov esi, OFFSET FLAT:.LC17
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >& std::operator<< <
       ↪ std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
       ↪ char> >&, char const*)
```

```
    mov esi, OFFSET FLAT:std::basic_ostream<char, std::char_traits<char> >&
        ↪ std::endl<char, std::char_traits<char> >(std::basic_ostream<char,
        ↪ std::char_traits<char> >&)
    mov rdi, rax
    call std::basic_ostream<char, std::char_traits<char> >::operator<<(std::
        ↪ basic_ostream<char, std::char_traits<char> >& (*)(std::basic_ostream
        ↪ <char, std::char_traits<char> >&))
    mov eax, 0
.L12:
    leave
    ret
.LC0:
    .long 560513024
    .long -921707870
    .long 16384
    .long 0
.LC1:
    .long 0
    .long -1275068416
    .long 16390
    .long 0
.LC9:
    .long 0
    .long 1104006501
```

Приложение 3: ассемблерный листинг функции *SinCalculation(long double x, long long n)* с ключом *-O0*, архитектурой x86-64 и комментариями

```
SinCalculation(long double, long long):         ; ИНИЦИАЛИЗАЦИЯ
    push    rbp                                 ; сохранение старого значения rbp
    mov     rbp, rsp                            ; устанавливаем новое значение rbp
        для фрейма стека (стека вызова функции)
    mov     QWORD PTR [rbp-72], rdi
    fldz                                        ; загружаем 0.0 в FPU стек
    fstp    TBYTE PTR [rbp-16]                  ; сохраняем 0.0 в переменную sin
    fld     TBYTE PTR [rbp+16]                  ; загрузка x в FPU стек
    fstp    TBYTE PTR [rbp-32]                  ; созранение x в prev
    fld1                                        ; загружаем 1.0 в FPU стек
    fstp    TBYTE PTR [rbp-48]                  ; сохранение 1.0 в sign
    mov     QWORD PTR [rbp-56], 1               ; установка счетчика i в 1
    jmp     .L4                                 ; переход в начало цикла
.L5:                                            ; ОСНОВНОЙ ЦИКЛ
    fld     TBYTE PTR [rbp-48]
    fld     TBYTE PTR [rbp-32]
    fmulp   st(1), st                           ; sign * prev, результат в st(0)
    fld     TBYTE PTR [rbp-16]
    faddp   st(1), st                           ; sin += sign * prev
    fstp    TBYTE PTR [rbp-16]                  ; Сохранение нового значения sin
```

```
    fld     TBYTE PTR [rbp+16]
    fld     st(0)
    fmulp   st(1), st                       ; x * x
    mov     rax, QWORD PTR [rbp-56]         ; загрузка i
    add     rax, rax                        ; 2i
    add     rax, 1                          ; 2i + 1
    imul    rax, QWORD PTR [rbp-56]         ; (2i + 1)*i
    add     rax, rax
    mov     QWORD PTR [rbp-96], rax
    fild    QWORD PTR [rbp-96]
    fdivp   st(1), st                       ; (x * x) / ((2 * i) * (2 * i + 1))
    fld     TBYTE PTR [rbp-32]
    fmulp   st(1), st                       ; умножение prev на резкльтат деления
    fstp    TBYTE PTR [rbp-32]
    fld     TBYTE PTR [rbp-48]
    fchs                                    ; изменение знака sign
    fstp    TBYTE PTR [rbp-96]
    mov     rax, QWORD PTR [rbp-96]         ; загрузка sign
    mov     edx, DWORD PTR [rbp-88]         ; загрузка i
    mov     QWORD PTR [rbp-48], rax         ; сохранение нового sign
    mov     DWORD PTR [rbp-40], edx         ; сохранение нового i
    add     QWORD PTR [rbp-56], 1           ; увеличение i на 1
.L4:
    mov     rax, QWORD PTR [rbp-56]         ; загрузка i
    cmp     rax, QWORD PTR [rbp-72]         ; сравнение i и n
    jl      .L5                             ; i < n => переход в начало цикла
    fld     TBYTE PTR [rbp-16]              ; загрузка sin в FPU стек
    pop     rbp                             ; восстановление старого rbp
    ret                                     ; возврат из функции
```

Приложение 4: ассемблерный листинг программы *SinCalculation.cpp* с ключом *-O3*, архитектурой x86-64

```
    std::basic_ostream<char, std::char_traits<char> >& std::endl<char, std::
       ↪ char_traits<char> >(std::basic_ostream<char, std::char_traits<char>
       ↪ >&) [clone .isra.0] [clone .cold]:
    DegToRad(long double):
            fld TBYTE PTR [rsp+8]
            fmul QWORD PTR .LC1[rip]
            fdiv DWORD PTR .LC2[rip]
            ret
    SinCalculation(long double, long long):
            fld TBYTE PTR [rsp+8]
            cmp rdi, 1
            jle .L15
            fld st(0)
            mov ecx, 3
            mov eax, 1
            fmul st, st(1)
```

```
        fld1
        fldz
        jmp .L14
.L17:
        fxch st(1)
.L14:
        fld st(3)
        mov rdx, rcx
        add rcx, 2
        fmul st, st(2)
        imul rdx, rax
        add rax, 1
        add rdx, rdx
        faddp st(1), st
        mov QWORD PTR [rsp-16], rdx
        fild QWORD PTR [rsp-16]
        fdivr st, st(3)
        fmulp st(4), st
        fxch st(1)
        fchs
        cmp rdi, rax
        jne .L17
        fstp st(0)
        fstp st(1)
        fstp st(1)
        ret
.L15:
        fstp st(0)
        fldz
        ret
.LC9:
        .string "Usage: <angle in degrees> <number of terms>"
.LC10:
        .string "x = "
.LC11:
        .string ", n = "
.LC13:
        .string "sin("
.LC14:
        .string ") = "
.LC15:
        .string "Run \342\204\226"
.LC16:
        .string " took "
.LC17:
        .string " seconds to complete"
.LC18:
        .string "\n"
.LC19:
        .string "Average time: "
```

```
.LC21:
        .string " seconds"
main:
        push r15
        push r14
        push r13
        push r12
        push rbp
        push rbx
        sub rsp, 88
        cmp edi, 3
        je .L19
        mov edi, OFFSET FLAT:std::cerr
        mov edx, 43
        mov esi, OFFSET FLAT:.LC9
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ __ostream_insert<char, std::char_traits<char> >(std::
           ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
           ↪ long)
        mov edi, OFFSET FLAT:std::cerr
        call std::basic_ostream<char, std::char_traits<char> >& std::endl<
           ↪ char, std::char_traits<char> >(std::basic_ostream<char, std::
           ↪ char_traits<char> >&) [clone .isra.0]
.L20:
        add rsp, 88
        xor eax, eax
        pop rbx
        pop rbp
        pop r12
        pop r13
        pop r14
        pop r15
        ret
.L19:
        mov rdi, QWORD PTR [rsi+8]
        mov rbx, rsi
        mov edx, 10
        xor esi, esi
        call strtoll
        mov rdi, QWORD PTR [rbx+16]
        mov edx, 10
        xor esi, esi
        mov QWORD PTR [rsp+16], rax
        fild QWORD PTR [rsp+16]
        fstp TBYTE PTR [rsp]
        call strtoll
        mov edx, 4
        mov esi, OFFSET FLAT:.LC10
        mov edi, OFFSET FLAT:std::cout
        mov rbx, rax
```

```
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ __ostream_insert<char, std::char_traits<char> >(std::
           ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
           ↪ long)
        push QWORD PTR [rsp+8]
        mov edi, OFFSET FLAT:std::cout
        push QWORD PTR [rsp+8]
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ basic_ostream<char, std::char_traits<char> >::_M_insert<long
           ↪ double>(long double)
        pop rdi
        mov edx, 6
        pop r8
        mov rbp, rax
        mov esi, OFFSET FLAT:.LC11
        lea r13, [rsp+48]
        mov rdi, rax
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ __ostream_insert<char, std::char_traits<char> >(std::
           ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
           ↪ long)
        mov rdi, rbp
        mov rsi, rbx
        xor ebp, ebp
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ basic_ostream<char, std::char_traits<char> >::_M_insert<long
           ↪ long>(long long)
        lea r12, [rsp+64]
        mov rdi, rax
        call std::basic_ostream<char, std::char_traits<char> >& std::endl<
           ↪ char, std::char_traits<char> >(std::basic_ostream<char, std::
           ↪ char_traits<char> >&) [clone .isra.0]
        mov QWORD PTR [rsp+24], 0x000000000
.L29:
        mov rsi, r13
        mov edi, 4
        call clock_gettime
        fld TBYTE PTR [rsp]
        fmul QWORD PTR .LC1[rip]
        fdiv DWORD PTR .LC2[rip]
        fstp QWORD PTR [rsp+16]
        fld QWORD PTR [rsp+16]
        cmp rbx, 1
        jle .L30
        fld st(0)
        mov ecx, 3
        mov eax, 1
        fmul st, st(1)
        fldz
        fld1
```

```
.L22:
        fld st(0)
        mov rdx, rcx
        add rcx, 2
        fmul st, st(4)
        imul rdx, rax
        add rax, 1
        add rdx, rdx
        faddp st(2), st
        mov QWORD PTR [rsp+16], rdx
        fild QWORD PTR [rsp+16]
        fdivr st, st(3)
        fmulp st(4), st
        fchs
        cmp rbx, rax
        jne .L22
        fstp st(0)
        fstp st(1)
        fstp st(1)
.L21:
        mov rsi, r12
        mov edi, 4
        fstp TBYTE PTR [rsp+32]
        call clock_gettime
        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]
        cvtsi2sd xmm1, rax
        mov edx, 4
        mov esi, OFFSET FLAT:.LC13
        divsd xmm0, QWORD PTR .LC12[rip]
        mov edi, OFFSET FLAT:std::cout
        addsd xmm0, xmm1
        movsd QWORD PTR [rsp+16], xmm0
        call std::basic_ostream<char, std::char_traits<char> >& std::
            ↪ __ostream_insert<char, std::char_traits<char> >(std::
            ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
            ↪ long)
        push QWORD PTR [rsp+8]
        mov edi, OFFSET FLAT:std::cout
        push QWORD PTR [rsp+8]
        call std::basic_ostream<char, std::char_traits<char> >& std::
            ↪ basic_ostream<char, std::char_traits<char> >::_M_insert<long
            ↪ double>(long double)
        mov esi, OFFSET FLAT:.LC14
        mov r14, rax
```

```
        pop rax
        pop rdx
        mov edx, 4
        mov rdi, r14
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ __ostream_insert<char, std::char_traits<char> >(std::
           ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
           ↪ long)
        fld TBYTE PTR [rsp+32]
        mov rdi, r14
        sub rsp, 16
        fstp TBYTE PTR [rsp]
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ basic_ostream<char, std::char_traits<char> >::_M_insert<long
           ↪ double>(long double)
        pop rcx
        pop rsi
        mov r14, rax
        mov rax, QWORD PTR [rax]
        mov rax, QWORD PTR [rax-24]
        mov r15, QWORD PTR [r14+240+rax]
        test r15, r15
        je .L26
        cmp BYTE PTR [r15+56], 0
        je .L24
        movsx esi, BYTE PTR [r15+67]
.L25:
        mov rdi, r14
        add rbp, 1
        call std::basic_ostream<char, std::char_traits<char> >::put(char)
        mov rdi, rax
        call std::basic_ostream<char, std::char_traits<char> >::flush()
        mov edx, 7
        mov esi, OFFSET FLAT:.LC15
        mov edi, OFFSET FLAT:std::cout
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ __ostream_insert<char, std::char_traits<char> >(std::
           ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
           ↪ long)
        mov rsi, rbp
        mov edi, OFFSET FLAT:std::cout
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ basic_ostream<char, std::char_traits<char> >::_M_insert<long
           ↪ long>(long long)
        mov edx, 6
        mov esi, OFFSET FLAT:.LC16
        mov rdi, rax
        mov r14, rax
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ __ostream_insert<char, std::char_traits<char> >(std::
```

```
        ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
        ↪ long)
        movsd xmm0, QWORD PTR [rsp+16]
        mov rdi, r14
        call std::basic_ostream<char, std::char_traits<char> >& std::
        ↪ basic_ostream<char, std::char_traits<char> >::_M_insert<
        ↪ double>(double)
        mov edx, 20
        mov esi, OFFSET FLAT:.LC17
        mov r14, rax
        mov rdi, rax
        call std::basic_ostream<char, std::char_traits<char> >& std::
        ↪ __ostream_insert<char, std::char_traits<char> >(std::
        ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
        ↪ long)
        mov edx, 1
        mov esi, OFFSET FLAT:.LC18
        mov rdi, r14
        call std::basic_ostream<char, std::char_traits<char> >& std::
        ↪ __ostream_insert<char, std::char_traits<char> >(std::
        ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
        ↪ long)
        mov rax, QWORD PTR [r14]
        mov rax, QWORD PTR [rax-24]
        mov r15, QWORD PTR [r14+240+rax]
        test r15, r15
        je .L26
        cmp BYTE PTR [r15+56], 0
        je .L27
        movsx esi, BYTE PTR [r15+67]
.L28:
        mov rdi, r14
        call std::basic_ostream<char, std::char_traits<char> >::put(char)
        mov rdi, rax
        call std::basic_ostream<char, std::char_traits<char> >::flush()
        movsd xmm2, QWORD PTR [rsp+24]
        addsd xmm2, QWORD PTR [rsp+16]
        movsd QWORD PTR [rsp+24], xmm2
        cmp rbp, 5
        jne .L29
        mov edx, 14
        mov esi, OFFSET FLAT:.LC19
        mov edi, OFFSET FLAT:std::cout
        call std::basic_ostream<char, std::char_traits<char> >& std::
        ↪ __ostream_insert<char, std::char_traits<char> >(std::
        ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
        ↪ long)
        mov edi, OFFSET FLAT:std::cout
        movsd xmm0, QWORD PTR [rsp+24]
        divsd xmm0, QWORD PTR .LC20[rip]
```

```
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ basic_ostream<char, std::char_traits<char> >::_M_insert<
           ↪ double>(double)
        mov edx, 8
        mov esi, OFFSET FLAT:.LC21
        mov rbx, rax
        mov rdi, rax
        call std::basic_ostream<char, std::char_traits<char> >& std::
           ↪ __ostream_insert<char, std::char_traits<char> >(std::
           ↪ basic_ostream<char, std::char_traits<char> >&, char const*,
           ↪ long)
        mov rdi, rbx
        call std::basic_ostream<char, std::char_traits<char> >& std::endl<
           ↪ char, std::char_traits<char> >(std::basic_ostream<char, std::
           ↪ char_traits<char> >&) [clone .isra.0]
        jmp .L20
.L24:
        mov rdi, r15
        call std::ctype<char>::_M_widen_init() const
        mov rax, QWORD PTR [r15]
        mov esi, 10
        mov rax, QWORD PTR [rax+48]
        cmp rax, OFFSET FLAT:_ZNKSt5ctypeIcE8do_widenEc
        je .L25
        mov rdi, r15
        call rax
        movsx esi, al
        jmp .L25
.L27:
        mov rdi, r15
        call std::ctype<char>::_M_widen_init() const
        mov rax, QWORD PTR [r15]
        mov esi, 10
        mov rax, QWORD PTR [rax+48]
        cmp rax, OFFSET FLAT:_ZNKSt5ctypeIcE8do_widenEc
        je .L28
        mov rdi, r15
        call rax
        movsx esi, al
        jmp .L28
.L30:
        fstp st(0)
        fldz
        jmp .L21
main.cold:
.LC1:
        .long 1413754136
        .long 1074340347
.LC2:
        .long 1127481344
```

```
.LC12:
        .long 0
        .long 1104006501
.LC20:
        .long 0
```