Новосибирский государственный университет

Лабораторная работа №3

Выполнил: Леонтьев Данил Вячеславович

2022

Цель:

1. Знакомство с программной архитектурой х86/х86-64
2. Анализ ассемблерного листинга программы для архитектуры х86/х86-64

Вариант задания:

Алгоритм вычисления числа Пи с помощью разложения в ряд (ряд Грегори-Лейбница) по формуле Лейбница N первых членов ряда

Полный компилируемый листинг реализованной программы

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <math.h>

#include <stdio.h>

#include <time.h>

int main() {

long int N;

double pi = 0,b;

scanf("%ld", &N);

for (int i = 0; i < N; i++) {

pi=-pi;

b=4/(double)(2\*i+1);

pi=pi+b;

}

if (pi<0)

pi=-pi;

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

return 0;

}

Листинг на ассемблере

.LC1:

        .string "%lld"

.LC4:

        .string "%lf\n"

main:

        push    rbp

        mov     rbp, rsp

        sub     rsp, 32

        pxor    xmm0, xmm0

        movsd   QWORD PTR [rbp-8], xmm0

        lea     rax, [rbp-32]

        mov     rsi, rax

        mov     edi, OFFSET FLAT:.LC1

        mov     eax, 0

        call    \_\_isoc99\_scanf

        mov     QWORD PTR [rbp-16], 0

        jmp     .L2

.L3:

        movsd   xmm0, QWORD PTR [rbp-8]

        movq    xmm1, QWORD PTR .LC2[rip]

        xorpd   xmm0, xmm1

        movsd   QWORD PTR [rbp-8], xmm0

        mov     rax, QWORD PTR [rbp-16]

        add     rax, rax

        add     rax, 1

        pxor    xmm1, xmm1

        cvtsi2sd        xmm1, rax

        movsd   xmm0, QWORD PTR .LC3[rip]

        divsd   xmm0, xmm1

        movsd   QWORD PTR [rbp-24], xmm0

        movsd   xmm0, QWORD PTR [rbp-8]

        addsd   xmm0, QWORD PTR [rbp-24]

        movsd   QWORD PTR [rbp-8], xmm0

        add     QWORD PTR [rbp-16], 1

.L2:

        mov     rax, QWORD PTR [rbp-32]

        cmp     QWORD PTR [rbp-16], rax

        jl      .L3

        pxor    xmm0, xmm0

        comisd  xmm0, QWORD PTR [rbp-8]

        jbe     .L4

        movsd   xmm0, QWORD PTR [rbp-8]

        movq    xmm1, QWORD PTR .LC2[rip]

        xorpd   xmm0, xmm1

        movsd   QWORD PTR [rbp-8], xmm0

.L4:

        mov     rax, QWORD PTR [rbp-8]

        movq    xmm0, rax

        mov     edi, OFFSET FLAT:.LC4

        mov     eax, 1

        call    printf

        mov     eax, 0

        leave

        ret

.LC2:

        .long   0

        .long   -2147483648

        .long   0

        .long   0

.LC3:

        .long   0

        .long   1074790400

Вывод

В ходе проделанной работы я узнал работу на языке ассемблер и узнал, как работают оптимизации на низком уровне.