Новосибирский государственный университет

Лабораторная работа №4

Выполнил: Леонтьев Данил Вячеславович

2022

Цель:

1. Знакомство с программной архитектурой ARM
2. Анализ ассемблерного листинга программы для архитектуры ARM

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

.LC0:

        .ascii  "%lld\000"

.LC1:

        .ascii  "%lf\012\000"

main:

        push    {r4, r5, r7, r8, r9, r10, fp, lr}

        sub     sp, sp, #32

        add     r7, sp, #0

        mov     r2, #0

        mov     r3, #0

        strd    r2, [r7, #24]

        mov     r3, r7

        mov     r1, r3

        movw    r0, #:lower16:.LC0

        movt    r0, #:upper16:.LC0

        bl      \_\_isoc99\_scanf

        vmov.i32        d16, #0  @ di

        vstr.64 d16, [r7, #16]    @ int

        b       .L2

.L3:

        vldr.64 d16, [r7, #24]

        vneg.f64        d16, d16

        vstr.64 d16, [r7, #24]

        ldrd    r2, [r7, #16]

        adds    r10, r2, r2

        adc     fp, r3, r3

        mov     r2, r10

        mov     r3, fp

        adds    r8, r2, #1

        adc     r9, r3, #0

        mov     r0, r8

        mov     r1, r9

        bl      \_\_aeabi\_l2d

        vmov    d18, r0, r1

        vmov.f64        d17, #4.0e+0

        vdiv.f64        d16, d17, d18

        vstr.64 d16, [r7, #8]

        vldr.64 d17, [r7, #24]

        vldr.64 d16, [r7, #8]

        vadd.f64        d16, d17, d16

        vstr.64 d16, [r7, #24]

        ldrd    r2, [r7, #16]

        adds    r4, r2, #1

        adc     r5, r3, #0

        strd    r4, [r7, #16]

.L2:

        ldrd    r2, [r7]

        ldrd    r0, [r7, #16]

        cmp     r0, r2

        sbcs    r3, r1, r3

        blt     .L3

        vldr.64 d16, [r7, #24]

        vcmpe.f64       d16, #0

        vmrs    APSR\_nzcv, FPSCR

        bpl     .L4

        vldr.64 d16, [r7, #24]

        vneg.f64        d16, d16

        vstr.64 d16, [r7, #24]

.L4:

        ldrd    r2, [r7, #24]

        movw    r0, #:lower16:.LC1

        movt    r0, #:upper16:.LC1

        bl      printf

        movs    r3, #0

        mov     r0, r3

        adds    r7, r7, #32

        mov     sp, r7

        pop     {r4, r5, r7, r8, r9, r10, fp, pc}

Вывод

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