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 ЛАБОРАТОРНОЙ РАБОТЫ

«ВВЕДЕНИЕ В АРХИТЕКТУРУ ARM»

студента 2 курса, 23209 группы
Инокова Семёна Шухратовича

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:
А. Ю. Кудинов

Новосибирск 2024

СОДЕРЖАНИЕ

ЦЕЛЬ

1. Знакомство с программной архитектурой ARM.

2. Анализ ассемблерного листинга программы для архитектуры ARM.

ЗАДАНИЕ

1. Изучить основы программной архитектуры ARM.

2. Для программы на языке Си (из лабораторной работы 1) сгенерировать ассемблерные листинги для архитектуры ARM, 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 архитектуры ARM на конкретных участках ассемблерного кода.

ОПИСАНИЕ РАБОТЫ

1. Код на языке Си был скомпилирован для архитектуры ARM и ARM64
2. Результаты компиляции были проанализированы.

ЗАКЛЮЧЕНИЕ

В ходе данной работы я познакомился с программной архитектурой ARM, научился читать и анализировать ассемблерные листинги. Были сделаны следующие выводы:

- *В архитектуре ARM арифметические команды производятся над регистрами, в отличие от x86, где для этого используется память.*
- *Листинг программы на архитектуре x86 более читабелен и интуитивно понятен, чем листинг на ARM. Это понятно, так как ARM не рассчитан на удобство программирования вручную.*

Приложение 1. *Код программы для работы*

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <time.h>
#include <math.h>

double piCalculation(long long n){
    double pi = 4;

    for (long long i = 1; i <= n; ++i)
    {
        double x = 4. / (2*i + 1);
        if (i % 2)
        {
            pi -= x;
        }
        else
        {
            pi += x;
        }
    }

    return pi;
}

int main() {
    long long n = 200;

    double pi = piCalculation(n);
    printf("Pi number: %.10lf\n", pi);

    return 0;
}
```

Приложение 2. *Ассемблерный листинг ARM64*

```
piCalculation:
    sub     sp, sp, #48
    str     x0, [sp, 8]
    fmov    d31, 4.0e+0
    str     d31, [sp, 40]
    mov     x0, 1
    str     x0, [sp, 32]
    b       .L2
.L5:
    ldr     x0, [sp, 32]
    lsl     x0, x0, 1
    add     x0, x0, 1
    scvtf   d31, x0
    fmov    d30, 4.0e+0
    fdiv    d31, d30, d31
    str     d31, [sp, 24]
```

```
    ldr     x0, [sp, 32]
    and     x0, x0, 1
    cmp     x0, 0
    beq     .L3
    ldr     d30, [sp, 40]
    ldr     d31, [sp, 24]
    fsub    d31, d30, d31
    str     d31, [sp, 40]
    b       .L4
.L3:
    ldr     d30, [sp, 40]
    ldr     d31, [sp, 24]
    fadd    d31, d30, d31
    str     d31, [sp, 40]
.L4:
    ldr     x0, [sp, 32]
    add     x0, x0, 1
    str     x0, [sp, 32]
.L2:
    ldr     x1, [sp, 32]
    ldr     x0, [sp, 8]
    cmp     x1, x0
    ble     .L5
    ldr     d31, [sp, 40]
    fmov    d0, d31
    add     sp, sp, 48
    ret
.LC0:
    .string "Pi number: %.10lf\n"
main:
    stp     x29, x30, [sp, -32]!
    mov     x29, sp
    mov     x0, 200
    str     x0, [sp, 24]
    ldr     x0, [sp, 24]
    bl      piCalculation
    str     d0, [sp, 16]
    ldr     d0, [sp, 16]
    adrp    x0, .LC0
    add     x0, x0, :lo12:.LC0
    bl      printf
    mov     w0, 0
    ldp     x29, x30, [sp], 32
    ret
```

Приложение 3. *Ассемблерный листинг ARM*

```
piCalculation:
    push    {r4, r5, r7, r8, r9, r10, fp, lr}
    sub     sp, sp, #40
    add     r7, sp, #0
    strd    r0, [r7, #8]
    mov     r2, #0
    mov     r3, #0
    movt    r3, 16400
    strd    r2, [r7, #32]
    mov     r2, #1
    mov     r3, #0
    strd    r2, [r7, #24]
    b       .L2
.L5:
    ldrd    r2, [r7, #24]
    adds    r1, r2, r2
    str     r1, [r7]
    adcs    r3, r3, r3
    str     r3, [r7, #4]
    ldrd    r2, [r7]
    adds    r10, r2, #1
    adc     fp, r3, #0
    mov     r0, r10
    mov     r1, fp
    bl      __aeabi_l2d
    vmov    d18, r0, r1
    vmov.f64        d17, #4.0e+0
    vdiv.f64        d16, d17, d18
    vstr.64 d16, [r7, #16]
    ldrd    r2, [r7, #24]
    and     r4, r2, #1
    movs    r5, #0
    orrs    r3, r4, r5
    beq     .L3
    vldr.64 d17, [r7, #32]
    vldr.64 d16, [r7, #16]
    vsub.f64        d16, d17, d16
    vstr.64 d16, [r7, #32]
    b       .L4
.L3:
    vldr.64 d17, [r7, #32]
    vldr.64 d16, [r7, #16]
    vadd.f64        d16, d17, d16
    vstr.64 d16, [r7, #32]
.L4:
    ldrd    r2, [r7, #24]
    adds    r8, r2, #1
    adc     r9, r3, #0
```

```
    strd    r8, [r7, #24]
.L2:
    ldrd    r2, [r7, #24]
    ldrd    r0, [r7, #8]
    cmp     r0, r2
    sbcs    r3, r1, r3
    bge     .L5
    ldrd    r2, [r7, #32]
    vmov    d16, r2, r3
    vmov.f64        d0, d16
    adds    r7, r7, #40
    mov     sp, r7
    pop     {r4, r5, r7, r8, r9, r10, fp, pc}
.LC0:
    .ascii  "Pi number: %.10lf\012\000"
main:
    push    {r7, lr}
    sub     sp, sp, #16
    add     r7, sp, #0
    mov     r2, #200
    mov     r3, #0
    strd    r2, [r7, #8]
    ldrd    r0, [r7, #8]
    bl      piCalculation
    vstr.64 d0, [r7]
    ldrd    r2, [r7]
    movw    r0, #:lower16:.LC0
    movt    r0, #:upper16:.LC0
    bl      printf
    movs    r3, #0
    mov     r0, r3
    adds    r7, r7, #16
    mov     sp, r7
    pop     {r7, pc}
```