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

4. Составить отчет по лабораторной работе.

ОПИСАНИЕ РАБОТЫ

Код выбранной мной программы на Си выглядит следующим образом:

```c
#include <stdio.h>

double Power(double number, double pow) {
double result = 1;
for (int i = 1; i <= pow; i++) {
result *= number;
}
return result;
}

int main() {
double n = 150;
double x = 0.5;
double sum = 0;
for (double i = 1; i <= n; i++) {
sum += (Power(-1, i + 1) * Power(x, i) / i);
}
printf("%f\n", sum);

return 0;

}
```

ARM64 GCC 13.2.0 -O0

```
1  Power:
2          sub     sp, sp, #32
3          str     d0, [sp, 8]
4          str     d1, [sp]
5          fmov    d0, 1.0e+0
6          str     d0, [sp, 24]
7          mov     w0, 1
8          str     w0, [sp, 20]
9          b       .L2
10 .L3:
11         ldr     d1, [sp, 24]
12         ldr     d0, [sp, 8]
13         fmul    d0, d1, d0
14         str     d0, [sp, 24]
15         ldr     w0, [sp, 20]
16         add     w0, w0, 1
17         str     w0, [sp, 20]
18 .L2:
19         ldr     w0, [sp, 20]
20         scvtf   d0, w0
21         ldr     d1, [sp]
22         fcmpe   d1, d0
```

```
        bge     .L3
        ldr     d0, [sp, 24]
        add     sp, sp, 32
        ret
.LC0:
        .string "%f\n"
main:
        stp   x29, x30, [sp,-64]!
        mov     x29, sp
        str     d8, [sp, 16]
        mov x0,211106232532992
        movk    x0, 0x4062, lsl 48
        fmov    d0, x0
        str     d0, [sp, 40]
        fmov    d0, 5.0e-1
        str     d0, [sp, 32]
        str     xzr, [sp, 56]
        fmov    d0, 1.0e+0
        str     d0, [sp, 48]
        b       .L6
.L7:
        ldr     d1, [sp, 48]
        fmov    d0, 1.0e+0
        fadd    d0, d1, d0
        fmov    d1, d0
        fmov    d0, -1.0e+0
        bl      Power
        fmov    d8, d0
        ldr     d1, [sp, 48]
        ldr     d0, [sp, 32]
        bl      Power
        fmul    d1, d8, d0
        ldr     d0, [sp, 48]
        fdiv    d0, d1, d0
        ldr     d1, [sp, 56]
        fadd    d0, d1, d0
        str     d0, [sp, 56]
        ldr     d1, [sp, 48]
        fmov    d0, 1.0e+0
        fadd    d0, d1, d0
        str     d0, [sp, 48]
.L6:
        ldr     d1, [sp, 48]
        ldr     d0, [sp, 40]
        fcmpe   d1, d0
        bls     .L7
        ldr     d0, [sp, 56]
        adrp    x0, .LC0
        add   x0,x0,:lo12:.LC0
```

```
        bl      printf
        mov     w0, 0
        ldr     d8, [sp, 16]
        ldp    x29,x30,[sp],64
        ret
```

ARM64 GCC 13.2.0 -Ofast

```
Power:
        fmov    d2, 1.0e+0
        mov     w0, 1
        fcmpe   d1, d2
        bmi     .L1
.L4:
        add     w0, w0, 1
        fmul    d2, d2, d0
        scvtf   d3, w0
        fcmpe   d3, d1
        bls     .L4
.L1:
        fmov    d0, d2
        ret
.LC0:
        .string "%f\n"
main:
        fmov    d5, 1.0e+0
        movi    d0, #0
       stp x29, x30, [sp,-16]!
        mov     w1, 150
        fmov    d7, d5
        fmov    d6, 5.0e-1
        mov     x29, sp
.L16:
        fmov    d4, d5
        mov     w0, 1
        fadd    d5, d5, d7
        fmov    d1, 1.0e+0
        fcmpe   d4, #0.0
        bmi     .L11
.L13:
        add     w0, w0, 1
        fneg    d1, d1
        scvtf   d2, w0
        fcmpe   d5, d2
        bge     .L13
        fcmpe   d4, d7
        mov     w0, 1
        fmov    d2, 1.0e+0
```

```
        bmi     .L11
.L15:
        add     w0, w0, 1
        fmul    d2, d2, d6
        scvtf   d3, w0
        fcmpe   d3, d4
        bls     .L15
        fmul    d1, d1, d2
.L11:
        fdiv    d1, d1, d4
        subs    w1, w1, #1
        fadd    d0, d0, d1
        bne     .L16
        adrp    x0, .LC0
        add   x0,x0,:lo12:.LC0
        bl      printf
        mov     w0, 0
        ldp    x29,x30,[sp],16
        ret
```

ЗАКЛЮЧЕНИЕ

На практике удалось убедиться в большом регистровом файле ARM архитектуры, все арифметические команды действительно выполняются над регистрами, а не над оперативной памятью.

Листинги -O0 и -Ofast отличаются хотя бы тем, что в -Ofast заметен инлайн функции Power в основной код, так же почти отсутствуют обращения в память.