## «ВВЕДЕНИЕ В АРХИТЕКТУРУ ARM»

## Отчёт по лабораторной работе №4

## Выполнил: Ланин Д.М. ФИТ НГУ 2 курс

## Преподаватель: Артюхов А. А.

## Цель лабораторной работы:

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) cгенерировать 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. Отчет должен содержать следующее.
   * Титульный лист.
   * Цель лабораторной работы.
   * Полный компилируемый листинг реализованной программы и команды для ее компиляции.
   * Листинг на ассемблере с описаниями назначения команд с точки зрения реализации алгоритма выбранного варианта.
   * Вывод по результатам лабораторной работы.

**Для вычислений применяется алгоритм вычисления числа Пи методом Монте-Карло.**

*Далее подробно об алгоритме* : cначала в квадрат с центром в начале координат и со стороной два вписывается круг с единичным радиусом. Затем в этом квадрате случайным образом с равномерным распределением генерируются N точек. Точка может попасть в окружность или нет (условие попадания x2 + y 2 ≤ 1). Далее определяется число M точек, попавших в круг. При достаточно большом числе бросков N, по значениям M и N вычисляется число Пи: Pi = 4 \* M / N.

**Описание работы**

Выполнение работы включает следующие этапы:

1. Изучить программную архитектуру ARM(Advanced RISC Machine).
2. Для программы на языке С++ сгенерировать ассемблерный листинг
3. Провести анализ, сопоставляя команды языка С++ с машинными командами.
4. Сделать вывод о проделанной работе.

## Результат:

Изучены основы ARM. Выделены основные отличия архитектур x86/x86-64 и ARM. По результатам проведённого анализа было установлено, что в ассемблерном коде ARM обращение к памяти, работа с математическими функциями, работа с регистрами существенно отличаются от ассемблерного кода на x86/x86-64. На более высоких уровнях оптимизации, ARM использует намного больше регистров, для увеличения скорости исполнения, а так-же не обращаться к памяти, храня значения в регистрах общего назначения, больше использует коды условий для уменьшения повторного использования команд и неиспользуемые до оптимизации команды для сокращения кода.

Ассемблерный листинг для архитектуры ARM

.LC0:

        .string "%lf\n"

.LC1:

        .string "Total time: %lf seconds\n"

main:

        stp     x29, x30, [sp, -112]!

        mov     x29, sp

        mov     x0, 0

        bl      time

        bl      srand

        mov     x0, 57600

        movk    x0, 0x5f5, lsl 16

        str     x0, [sp, 88]

        str     xzr, [sp, 104]

        bl      clock

        str     x0, [sp, 80]

        str     wzr, [sp, 100]

        b       .L2

.L5:

        bl      rand

        scvtf   d0, w0

        mov     x0, 281474972516352

        movk    x0, 0x41df, lsl 48

        fmov    d1, x0

        fdiv    d0, d0, d1

        bl      \_\_extenddftf2

        str     q0, [sp, 48]

        bl      rand

        scvtf   d0, w0

        mov     x0, 281474972516352

        movk    x0, 0x41df, lsl 48

        fmov    d1, x0

        fdiv    d0, d0, d1

        bl      \_\_extenddftf2

        str     q0, [sp, 32]

        ldr     q1, [sp, 48]

        ldr     q0, [sp, 48]

        bl      \_\_multf3

        str     q0, [sp, 16]

        ldr     q1, [sp, 32]

        ldr     q0, [sp, 32]

        bl      \_\_multf3

        mov     v1.16b, v0.16b

        ldr     q0, [sp, 16]

        bl      \_\_addtf3

        adrp    x0, .LC2

        add     x0, x0, :lo12:.LC2

        ldr     q1, [x0]

        bl      \_\_letf2

        cmp     w0, 0

        bgt     .L3

        ldr     x0, [sp, 104]

        add     x0, x0, 1

        str     x0, [sp, 104]

.L3:

        ldr     w0, [sp, 100]

        add     w0, w0, 1

        str     w0, [sp, 100]

.L2:

        ldrsw   x0, [sp, 100]

        ldr     x1, [sp, 88]

        cmp     x1, x0

        bhi     .L5

        bl      clock

        str     x0, [sp, 72]

        ldr     x1, [sp, 72]

        ldr     x0, [sp, 80]

        sub     x0, x1, x0

        fmov    d0, x0

        scvtf   d0, d0

        mov     x0, 145685290680320

        movk    x0, 0x412e, lsl 48

        fmov    d1, x0

        fdiv    d0, d0, d1

        str     d0, [sp, 64]

        ldr     d0, [sp, 104]

        ucvtf   d1, d0

        fmov    d0, 4.0e+0

        fmul    d1, d1, d0

        ldr     d0, [sp, 88]

        ucvtf   d0, d0

        fdiv    d0, d1, d0

        adrp    x0, .LC0

        add     x0, x0, :lo12:.LC0

        bl      printf

        ldr     d0, [sp, 64]

        adrp    x0, .LC1

        add     x0, x0, :lo12:.LC1

        bl      printf

        mov     w0, 0

        ldp     x29, x30, [sp], 112

        ret

.LC2:

        .word   0

        .word   0

        .word   0

        .word   1073676288

Ассемблерный листик для архитектуры ARM -O2

.LC0:

        .string "%lf\n"

.LC1:

        .string "Total time: %lf seconds\n"

main:

        stp     x29, x30, [sp, -64]!

        mov     x0, 0

        mov     x29, sp

        stp     x19, x20, [sp, 16]

        mov     w19, 57600

        movk    w19, 0x5f5, lsl 16

        str     x21, [sp, 32]

        mov     x20, 0

        str     d8, [sp, 40]

        bl      time

        bl      srand

        bl      clock

        mov     x21, x0

        mov     x0, 281474972516352

        movk    x0, 0x41df, lsl 48

        fmov    d8, x0

.L4:

        bl      rand

        scvtf   d0, w0

        fdiv    d0, d0, d8

        bl      \_\_extenddftf2

        str     q0, [sp, 48]

        bl      rand

        scvtf   d0, w0

        fdiv    d0, d0, d8

        bl      \_\_extenddftf2

        mov     v2.16b, v0.16b

        ldr     q1, [sp, 48]

        str     q2, [sp, 48]

        mov     v0.16b, v1.16b

        bl      \_\_multf3

        ldr     q1, [sp, 48]

        mov     v2.16b, v0.16b

        mov     v0.16b, v1.16b

        str     q2, [sp, 48]

        bl      \_\_multf3

        mov     v1.16b, v0.16b

        ldr     q0, [sp, 48]

        bl      \_\_addtf3

        adrp    x0, .LC2

        add     x0, x0, :lo12:.LC2

        ldr     q1, [x0]

        bl      \_\_letf2

        cmp     w0, 0

        cinc    x20, x20, le

        subs    w19, w19, #1

        bne     .L4

        bl      clock

        mov     x19, x0

        ucvtf   d0, x20

        fmov    d2, 4.0e+0

        mov     x1, 236961935654912

        sub     x19, x19, x21

        movk    x1, 0x4197, lsl 48

        fmov    d1, x1

        adrp    x0, .LC0

        add     x0, x0, :lo12:.LC0

        fmul    d0, d0, d2

        fdiv    d0, d0, d1

        bl      printf

        scvtf   d0, x19

        mov     x0, 145685290680320

        movk    x0, 0x412e, lsl 48

        fmov    d1, x0

        adrp    x0, .LC1

        add     x0, x0, :lo12:.LC1

        fdiv    d0, d0, d1

        bl      printf

        ldp     x19, x20, [sp, 16]

        mov     w0, 0

        ldr     x21, [sp, 32]

        ldr     d8, [sp, 40]

        ldp     x29, x30, [sp], 64

        ret

.LC2:

        .word   0

        .word   0

        .word   0

        .word   1073676288