**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**

«Лабораторная работа #4»

студента (ки) \_\_\_\_2\_\_\_\_\_ курса, \_\_19211\_\_ группы

**Олимпиева Юрия Юрьевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

**Н.А. Беляев**

2020\_год

**СОДЕРЖАНИЕ**

Цель……….………………………………………………………………….2

Задание...….………………………………………………………………….2

Описание работы....………………………………………………………….2

Вывод…………………………………………………………………………2

Заключение.………………………………………………………………….3

Приложение 1………………………………………………………………..5

Приложение 2………………………………………………………………..8

**Цели**

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**

* Сгенерировал код ассемблера под ARM для программы на ЯП Си, вычисляющей значение экспоненты для уровней оптимизации –O0 и

-O1. (приложение 1, приложение 2)

* Сопоставил команды языка Си с командами на языке ассемблера под архитектуру ARM, переменные на языке
* Сопоставил переменные программы на языке Си с регистрами и участками памяти на стеке.
* Выявил оптимизационные преобразования кода при переходе от уровня оптимизации –O0 к –O1.
* Составил отчёт о выполненной работе.

**Вывод**

С целью оптимизации исполнения программы компилятор уменьшил количество обращений к памяти, переместил переменные в регистры для ускорения работы с ними, иногда менял порядок исполнения команд (порядок команд на языке ассемблера не всегда соответствовал порядку команд на языке Cи).

**Заключение**

В результате проделанной работы познакомился с устройством программной архитектуры ARM, получил базовые знания и навыки для анализа ассемблерного листинга программы для архитектуры ARM.

Приложение 1

(ARM GCC8.2 –O0)

McLorenExp(double, int):

push {r4, r5, fp, lr}; пуш данных на стек

add fp, sp, #12 ; fp = sp + 12 (fp - frame pointer)

sub sp, sp, #40 ; sp = sp - 40(sp - stack pointer)

str r0, [fp, #-44] ; из регистра r0 загружаем данные по адресу fp - 44

str r1, [fp, #-40] ; из регистра r1 загружаем данные по адресу fp - 40

;fp-44 и fp-40 , вероятно, аргумент функции типа double

; fp-48 - аргумент функции типа int

str r2, [fp, #-48] ; из регистра r2 загружаем данные по адресу fp - 48

mov r3, #0 ; в регистр r3 помещаем 0

mov r4, #0 ; в регистр r4 помещаем 0

str r3, [fp, #-20] ; из регистра r3 выгружаем данные по адресу fp - 20

str r4, [fp, #-16] ; из регистра r4 выгружаем данные по адресу fp - 16(double ; result = 0.0)

mov r3, #0 ; в регистр r3 помещаем 0

ldr r4, .L5 ; взять данные по метке .L5 и положить в r4

str r3, [fp, #-28] ; присвоение

str r4, [fp, #-24] ; element = 1.0

mov r3, #0

str r3, [fp, #-32] ; по адресу [fp, #-32] (счётчик цикла) помещаем 0 через r3

.L3: ;метка на начало цикла for

ldr r2, [fp, #-32] ;

ldr r3, [fp, #-48] ;

cmp r2, r3 ; i(счётчик цикла) < N(аргумент функции) ?

bge .L2 ; переход на метку .L2 если больше

sub r3, fp, #28 ; r3 = fp - 28

ldmia r3, {r2-r3} ; обращение к стеку

sub r1, fp, #20 ; r1 = fp - 20

ldmia r1, {r0-r1} ; обращение к стеку

bl \_\_aeabi\_dadd ; вызов функции для подсчёта result + element

mov r3, r0 ; из r0 -> r3

mov r4, r1 ; из r4 -> r1

str r3, [fp, #-20] ; из r3 -> fp - 20

str r4, [fp, #-16] ; из r4 -> fp - 16 выгрузка результата в result

sub r3, fp, #44 ; r3 = fp - 44()

ldmia r3, {r2-r3}

sub r1, fp, #28 ; r1 = fp - 28 ()

ldmia r1, {r0-r1}

bl \_\_aeabi\_dmul ; операция умножения (element \* x)

mov r3, r0

mov r4, r1

mov r5, r4

mov r4, r3

ldr r3, [fp, #-32] ; счётчик цикла i -> r3

add r3, r3, #1 ; (i + 1)

mov r0, r3 (i + 1) -> r0

bl \_\_aeabi\_i2d ; приведение к типу double

mov r2, r0 ; жонглирование

mov r3, r1 ; регистрами

mov r0, r4 ;

mov r1, r5 ;

bl \_\_aeabi\_ddiv; операция деления (element \* x / (double)(i + 1))

mov r3, r0

mov r4, r1

str r3, [fp, #-28] ; выгрузка результата

str r4, [fp, #-24] ; в переменную result

ldr r3, [fp, #-32] ; тут

add r3, r3, #1 ;инкремент

str r3, [fp, #-32] ; счётчика цикла

b .L3 ; безусловный переход

.L2:

sub r4, fp, #20 ; выгрузка результата

ldmia r4, {r3-r4} ; для возврата из функции

mov r0, r3 ;

mov r1, r4 ;

sub sp, fp, #12 ; восстановление состояния указателя на стек

pop {r4, r5, fp, pc} ; вытягиваем из стека значения регистров и frame pointer и

; prog counter - указатель на следующую команду

.L5:

.word 1072693248

.LC0:

.ascii "wb\000"

.LC1:

.ascii "out.txt\000"

.LC2:

.ascii "%lf\000"

main:

push {r4, r5, fp, lr} ; пуш на стек

add fp, sp, #12 ; fp = sp + 12

sub sp, sp, #16 ; sp = sp - 16

ldr r0, .L13 ; выгружаем аргумент для malloc в r0

bl malloc ; вызов malloc

mov r3, r0 ; r3 = r0

str r3, [fp, #-24]; помещаем результат malloc на стек (перем-ая arr)

mov r3, #0 ; r3 = 0

str r3, [fp, #-16] ; обнуляем переменную на стеке по адресу fp - 16 ( i = 0)

.L9:

ldr r3, [fp, #-16] ; выгрузка значения переменной счётчика цикла -> r3

ldr r2, .L13+4 ; смещение адреса для перемещения по массиву

cmp r3, r2

bgt .L8 ; условный переход

ldr r0, [fp, #-16] ;

bl \_\_aeabi\_i2d ; приведение к типу double

mov r3, r0 ;

mov r4, r1 ; чудесные превращения

mov r2, r3 ; регистров

mov r3, r4 ;

mov r0, #0 ;

ldr r1, .L13+8 ;

bl \_\_aeabi\_ddiv ; операция деления (1.0 / (double) i)

mov r3, r0

mov r4, r1

mov r0, r3

mov r1, r4

ldr r3, [fp, #-16] ; счётчик цикла -> r3

lsl r3, r3, #3 ; логический сдвиг влево

ldr r2, [fp, #-24]

add r5, r2, r3

mov r2, #1000 ; r2 = 1000

bl McLorenExp(double, int) ; вызов функции McLorenExp

mov r3, r0

mov r4, r1

stm r5, {r3-r4} ; что-то куда-то выгружаем

ldr r3, [fp, #-16] ;

add r3, r3, #1 ; инкремент счётчика цикла for

str r3, [fp, #-16] ;

b .L9

.L8:

ldr r1, .L13+12 ; выгружаем константные строки

ldr r0, .L13+16 ; для передачи в качестве аргументов fopen

bl fopen ; вызов fopen

mov r3, r0

str r3, [fp, #-28] ; по адресу fp - 28 выгружаем результат fopen ( перем - ая out)

mov r3, #0 ; обнуление

str r3, [fp, #-20] ; счётчика цикла

.L11:

ldr r3, [fp, #-20] ;

ldr r2, .L13+4 ; смещение адреса для перемещения по массиву

cmp r3, r2

bgt .L10 ; условный переход в цикле for ()

ldr r3, [fp, #-20]

lsl r3, r3, #3 ; логический свдиг влево

ldr r2, [fp, #-24]

add r3, r2, r3 ; арифметика указателей для перемещения по массиву

ldmia r3, {r3-r4}

mov r2, r3

mov r3, r4

ldr r1, .L13+20 ; константная строка в r1 для функции fprintf()

ldr r0, [fp, #-28]

bl fprintf

; далее до .L10 - инкремент счётчика цикла

ldr r3, [fp, #-20]

add r3, r3, #1

str r3, [fp, #-20]

b .L11

;

.L10:

ldr r0, [fp, #-24] ; arr -> r0

bl free ; free(arr)

ldr r0, [fp, #-28] ; out -> r0

bl fclose ; fclode(out)

mov r3, #0 ; r3 = 0

mov r0, r3 ; r0 = r3

sub sp, fp, #12 ; возврат состояния sp

pop {r4, r5, fp, pc} ; вытягиваем данные со стека

.L13:

.word 51200000

.word 6399999

.word 1072693248

.word .LC0

.word .LC1

.word .LC2

Приложение 2

(ARM GCC8.2 –O1)

McLorenExp(double, int):

push {r3, r4, r5, r6, r7, r8, r9, r10, fp, lr} ; пуш данных на стек

subs r5, r2, #0 ; проверка условий продолжения цикла

ble .L4 ; переход за тело цикла for

mov r10, r0 ;

mov fp, r1 ;

mov r4, #0 ;

mov r6, #0 ;

ldr r7, .L7 ; element = 1.0

mov r8, #0 ;result = 0.0

mov r9, #0

.L3:

; result += element

mov r2, r6

mov r3, r7

mov r0, r8

mov r1, r9

bl \_\_aeabi\_dadd

mov r8, r0

mov r9, r1

;

mov r2, r6 ;

mov r3, r7 ;

mov r0, r10 ;

mov r1, fp ;

bl \_\_aeabi\_dmul ; x \* element

mov r6, r0

mov r7, r1

add r4, r4, #1 ; инкремент счётчика цикла

mov r0, r4

bl \_\_aeabi\_i2d ; приведение к типу double

mov r2, r0

mov r3, r1

mov r0, r6

mov r1, r7

bl \_\_aeabi\_ddiv ; деление x \* element / (double)(i + 1)

mov r6, r0

mov r7, r1

cmp r4, r5

bne .L3 ; переход по метке .L3 если r4 != r5

.L1:

mov r0, r8 ; данные для возврата

mov r1, r9

pop {r3, r4, r5, r6, r7, r8, r9, r10, fp, pc} ; вытягиваем сос тека данные, ; восстанавливаем fp

.L4:

mov r8, #0

mov r9, #0

b .L1

.L7:

.word 1072693248

main:

push {r3, r4, r5, r6, r7, r8, r9, r10, fp, lr} ; пуш данных на стек

ldr r0, .L15 ; аргумент для malloc -. r0

bl malloc ; oвызов маллок

mov r7, r0

mov r5, r0

mov r6, r0

mov r4, #0

mov r8, #0

ldr r9, .L15+4 ; подгружаем в регистры константы

mov fp, #1000

ldr r10, .L15+8 ; подгружаем в регистры константы

.L10:

mov r0, r4

bl \_\_aeabi\_i2d ; приведение типа к double

mov r2, r0

mov r3, r1

mov r0, r8

mov r1, r9

bl \_\_aeabi\_ddiv ; деление (1.0 / (double) i )

mov r2, fp

bl McLorenExp(double, int) ; вызов функции McLorenExp

stm r6!, {r0-r1}

add r4, r4, #1 ; инкремент

cmp r4, r10 ; проверка условия выхода из цикла

bne .L10

ldr r1, .L15+12 ; подгружаем константы для вызова функции fopen

ldr r0, .L15+16 ;

bl fopen ; вызов функции fopen

mov r6, r0

add r4, r7, #51118080 ;чудеса машинной оптимизации - непонятные константы =)

add r4, r4, #81920 ; вроде как r4 = r7 + 51118080 + 81920 + r7 + 51118080

ldr r8, .L15+20

.L11:

ldmia r5!, {r2-r3}

mov r1, r8

mov r0, r6

bl fprintf

cmp r5, r4

bne .L11

mov r0, r7

bl free

mov r0, r6

bl fclose

mov r0, #0

pop {r3, r4, r5, r6, r7, r8, r9, r10, fp, pc}

.L15:

.word 51200000

.word 1072693248

.word 6400000

.word .LC0

.word .LC1

.word .LC2

.LC0:

.ascii "wb\000"

.LC1:

.ascii "out.txt\000"

.LC2:

.ascii "%lf\000"