###### 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 ПРАКТИЧЕСКОЙ РАБОТЫ

«Оптимизация ассемблерного листинга архитектуры x86/x86-64»

студента 2 курса, группы 20204

**Дронова Дениса Юрьевича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

Кандидат технических наук

А. Ю. Власенко

Новосибирск 2021

**СОДЕРЖАНИЕ**

[ЦЕЛЬ 3](#_Toc91332641)

[ЗАДАНИЕ 3](#_Toc91332642)

[ОПИСАНИЕ РАБОТЫ 4](#_Toc91332643)

[ЗАКЛЮЧЕНИЕ 5](#_Toc91332644)

[Приложение 1. *Ассемблерный листинг asm0.s* 6](#_Toc91332645)

[Приложение 2. *Ассемблерный листинг asm\_optimized.s* 10](#_Toc91332646)

# ЦЕЛЬ

1. Научиться оптимизировать ассемблерные листинги архитектуры x86/x86\_64 сохранив при этом корректность исполнения.

# ЗАДАНИЕ

Дана программа на ассемблере, скомпилированная с ключом оптимизации O0 (prog0.s). В ней реализован вычислительный алгоритм. Необходимо оптимизировать программу так, чтобы она осталась корректной. Получить выигрыш в производительности 6–10%. Отчет должен содержать:

· 2 ассемблерных листинга: первоначальный и оптимизированный;

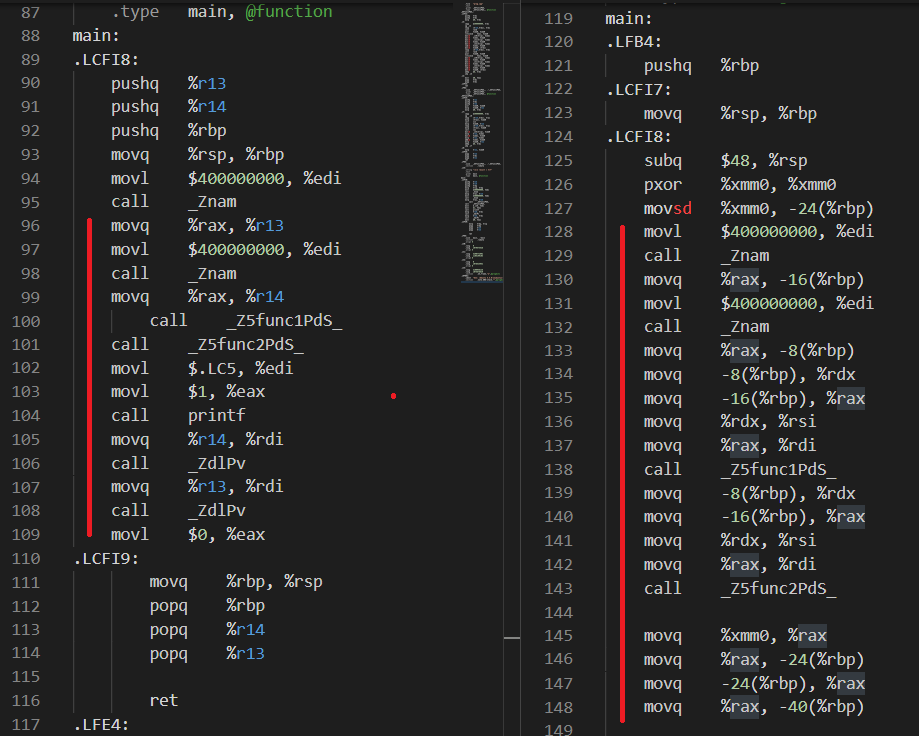
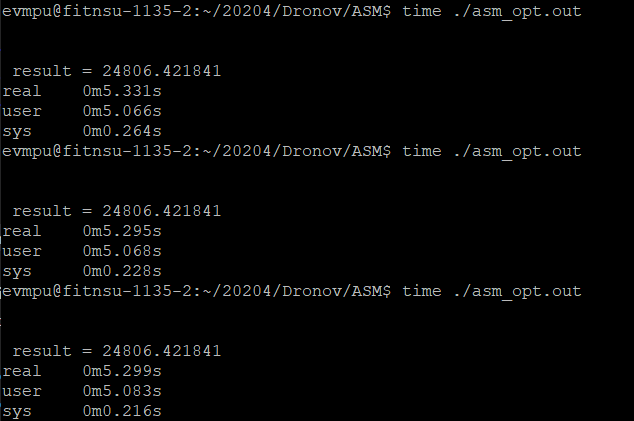
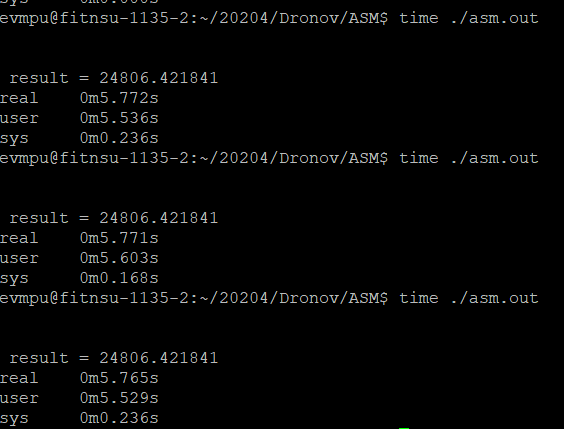
· описание всех сделанных оптимизаций с пояснениями;

· результаты замеров времени первоначальной программы и оптимизированной – минимум по 3 запуска каждой из программ.

# ОПИСАНИЕ РАБОТЫ

После подробного разбора приведенного листинга программы, было выяснено, какую задачу она выполняет. Основываясь на этом, были произведены ручные оптимизации, включающие в себя сведение к минимуму обращений к памяти(стеку) и операций с ним, т. к. подобные являются наиболее затратными по времени. Примеры оптимизаций приведены ниже а также в приложении (см. листинги программ asm0.s, asm\_optimized.s).

Строка компиляции: g++ asm\_optimized.s -o asm\_optimized.out -no-pie



После выполнения перечисленных оптимизаций было получено среднее время исполнения программы: ~5.308 секунд, что означает ~8% прирост скорости исполнения по сравнению с изначальной программой (5.769 секунд).

# ЗАКЛЮЧЕНИЕ

Во время выполнения практической работы были рассмотрены и применены на практике некоторые способы оптимизации ассемблерных листингов архитектуры x86/x86\_64. Основываясь на результатах проделанной работы, можно сделать вывод о том, что одним из самых эффективных способов оптимизации является сведение к минимуму работы с памятью.

# Приложение 1. *Ассемблерный листинг asm0.s*

    .file   "prog.cpp"

    .text

    .globl  \_Z5func1PdS\_

    .type   \_Z5func1PdS\_, @function

\_Z5func1PdS\_:

.LFB2:

    pushq   %rbp

.LCFI0:

    movq    %rsp, %rbp

.LCFI1:

    pushq   %rbx

    subq    $40, %rsp

.LCFI2:

    movq    %rdi, -40(%rbp)

    movq    %rsi, -48(%rbp)

    movl    $0, -20(%rbp)

.L3:

    cmpl    $49999999, -20(%rbp)

    jg  .L2

    movl    -20(%rbp), %eax

    cltq

    leaq    0(,%rax,8), %rdx

    movq    -40(%rbp), %rax

    leaq    (%rdx,%rax), %rbx

    call    rand

    pxor    %xmm0, %xmm0

    cvtsi2sd    %eax, %xmm0

    movsd   .LC0(%rip), %xmm1

    mulsd   %xmm1, %xmm0

    movsd   .LC1(%rip), %xmm1

    divsd   %xmm1, %xmm0

    movsd   .LC2(%rip), %xmm1

    subsd   %xmm1, %xmm0

    movsd   %xmm0, (%rbx)

    movl    -20(%rbp), %eax

    cltq

    leaq    0(,%rax,8), %rdx

    movq    -48(%rbp), %rax

    leaq    (%rdx,%rax), %rbx

    call    rand

    pxor    %xmm0, %xmm0

    cvtsi2sd    %eax, %xmm0

    movsd   .LC0(%rip), %xmm1

    mulsd   %xmm1, %xmm0

    movsd   .LC1(%rip), %xmm1

    divsd   %xmm1, %xmm0

    movsd   .LC2(%rip), %xmm1

    subsd   %xmm1, %xmm0

    movsd   %xmm0, (%rbx)

    addl    $1, -20(%rbp)

    jmp .L3

.L2:

    movl    $0, %eax

    addq    $40, %rsp

    popq    %rbx

    popq    %rbp

.LCFI3:

    ret

.LFE2:

    .size   \_Z5func1PdS\_, .-\_Z5func1PdS\_

    .globl  \_Z5func2PdS\_

    .type   \_Z5func2PdS\_, @function

\_Z5func2PdS\_:

.LFB3:

    pushq   %rbp

.LCFI4:

    movq    %rsp, %rbp

.LCFI5:

    subq    $48, %rsp

    movq    %rdi, -24(%rbp)

    movq    %rsi, -32(%rbp)

    pxor    %xmm0, %xmm0

    movsd   %xmm0, -8(%rbp)

    movl    $0, -12(%rbp)

.L7:

    cmpl    $49999999, -12(%rbp)

    jg  .L6

    movl    -12(%rbp), %eax

    cltq

    leaq    0(,%rax,8), %rdx

    movq    -24(%rbp), %rax

    addq    %rdx, %rax

    movq    (%rax), %rax

    movq    %rax, -40(%rbp)

    movsd   -40(%rbp), %xmm0

    call    sin

    movapd  %xmm0, %xmm1

    movsd   .LC4(%rip), %xmm0

    mulsd   %xmm0, %xmm1

    movsd   %xmm1, -40(%rbp)

    movl    -12(%rbp), %eax

    cltq

    leaq    0(,%rax,8), %rdx

    movq    -32(%rbp), %rax

    addq    %rdx, %rax

    movq    (%rax), %rax

    movq    %rax, -48(%rbp)

    movsd   -48(%rbp), %xmm0

    call    cos

    mulsd   -40(%rbp), %xmm0

    movsd   -8(%rbp), %xmm1

    addsd   %xmm1, %xmm0

    movsd   %xmm0, -8(%rbp)

    addl    $1, -12(%rbp)

    jmp .L7

.L6:

    movsd   -8(%rbp), %xmm0

    leave

.LCFI6:

    ret

.LFE3:

    .size   \_Z5func2PdS\_, .-\_Z5func2PdS\_

    .section    .rodata

.LC5:

    .string "\n\n result = %lf"

    .text

    .globl  main

    .type   main, @function

main:

.LFB4:

    pushq   %rbp

.LCFI7:

    movq    %rsp, %rbp

.LCFI8:

    subq    $48, %rsp

    pxor    %xmm0, %xmm0

    movsd   %xmm0, -24(%rbp)

    movl    $400000000, %edi

    call    \_Znam

    movq    %rax, -16(%rbp)

    movl    $400000000, %edi

    call    \_Znam

    movq    %rax, -8(%rbp)

    movq    -8(%rbp), %rdx

    movq    -16(%rbp), %rax

    movq    %rdx, %rsi

    movq    %rax, %rdi

    call    \_Z5func1PdS\_

    movq    -8(%rbp), %rdx

    movq    -16(%rbp), %rax

    movq    %rdx, %rsi

    movq    %rax, %rdi

    call    \_Z5func2PdS\_

    movq    %xmm0, %rax

    movq    %rax, -24(%rbp)

    movq    -24(%rbp), %rax

    movq    %rax, -40(%rbp)

    movsd   -40(%rbp), %xmm0

    movl    $.LC5, %edi

    movl    $1, %eax

    call    printf

    movq    -16(%rbp), %rax

    movq    %rax, %rdi

    call    \_ZdlPv

    movq    -8(%rbp), %rax

    movq    %rax, %rdi

    call    \_ZdlPv

    movl    $0, %eax

    leave

.LCFI9:

    ret

.LFE4:

    .size   main, .-main

    .section    .rodata

    .align 8

.LC0:

    .long   0

    .long   1079574528

    .align 8

.LC1:

    .long   4290772992

    .long   1105199103

    .align 8

.LC2:

    .long   0

    .long   1078525952

    .align 8

.LC4:

    .long   3100958126

    .long   1075678820

    .section    .eh\_frame,"a",@progbits

.LEFDE1:

    .ident  "GCC: (Ubuntu 5.5.0-12ubuntu1~16.04) 5.5.0 20171010"

    .section    .note.GNU-stack,"",@progbits

# Приложение 2. *Ассемблерный листинг asm\_optimized.s*

    .file   "prog.cpp"

    .text

    .globl  \_Z5func1PdS\_

    .type   \_Z5func1PdS\_, @function

\_Z5func1PdS\_:

.LCFI2:

    pushq   %rbx

    pushq   %r15

    movq    $0, %r15

.L3:

    cmpq    $399999999, %r15

    jg .L2

    leaq    (%r13,%r15), %rbx

    call    rand

    pxor    %xmm0, %xmm0

    cvtsi2sd    %eax, %xmm0

    movsd   .LC0(%rip), %xmm1

    mulsd   %xmm1, %xmm0

    movsd   .LC1(%rip), %xmm1

    divsd   %xmm1, %xmm0

    movsd   .LC2(%rip), %xmm1

    subsd   %xmm1, %xmm0

    movsd   %xmm0, (%rbx)

    leaq    (%r14,%r15), %rbx

    call    rand

    pxor    %xmm0, %xmm0

    cvtsi2sd    %eax, %xmm0

    movsd   .LC0(%rip), %xmm1

    mulsd   %xmm1, %xmm0

    movsd   .LC1(%rip), %xmm1

    divsd   %xmm1, %xmm0

    movsd   .LC2(%rip), %xmm1

    subsd   %xmm1, %xmm0

    movsd   %xmm0, (%rbx)

    addq    $8, %r15

    jmp .L3

.L2:

    movl    $0, %eax

    popq    %r15

    popq    %rbx

.LCFI3:

    ret

.LFE2:

    .size   \_Z5func1PdS\_, .-\_Z5func1PdS\_

    .globl  \_Z5func2PdS\_

    .type   \_Z5func2PdS\_, @function

\_Z5func2PdS\_:

.LCFI5:

    pushq   %r11

    pushq   %r15

    pushq   %r12

    pxor    %xmm0, %xmm0

    movq    %xmm0, %r11

    movq    $0, %r15

.L7:

    cmpq    $399999999, %r15

    jg  .L6

    leaq    (%r13,%r15), %rax

    movq    (%rax), %xmm0

    call    sin

    movq    %xmm0, %r12

    leaq    (%r14,%r15), %rax

    movq    (%rax), %xmm0

    call    cos

    mulsd   .LC4(%rip), %xmm0

    movq    %r12, %xmm1

    mulsd   %xmm1, %xmm0

    movq    %r11, %xmm1

    addsd   %xmm1, %xmm0

    movq    %xmm0, %r11

    addq    $8, %r15

    jmp .L7

.L6:

    movq    %r11, %xmm0

.LCFI6:

    popq    %r12

    popq    %r15

    popq    %r11

    ret

.LFE3:

    .size   \_Z5func2PdS\_, .-\_Z5func2PdS\_

    .section    .rodata

.LC5:

    .string "\n\n result = %lf"

    .text

    .globl  main

    .type   main, @function

main:

.LCFI8:

    pushq   %r13

    pushq   %r14

    pushq   %rbp

    movq    %rsp, %rbp

    movl    $400000000, %edi

    call    \_Znam

    movq    %rax, %r13

    movl    $400000000, %edi

    call    \_Znam

    movq    %rax, %r14

        call    \_Z5func1PdS\_

    call    \_Z5func2PdS\_

    movl    $.LC5, %edi

    movl    $1, %eax

    call    printf

    movq    %r14, %rdi

    call    \_ZdlPv

    movq    %r13, %rdi

    call    \_ZdlPv

    movl    $0, %eax

.LCFI9:

        movq    %rbp, %rsp

        popq    %rbp

        popq    %r14

        popq    %r13

        ret

.LFE4:

    .size   main, .-main

    .section    .rodata

    .align 8

.LC0:

    .long   0

    .long   1079574528

    .align 8

.LC1:

    .long   4290772992

    .long   1105199103

    .align 8

.LC2:

    .long   0

    .long   1078525952

    .align 8

.LC4:

    .long   3100958126

    .long   1075678820

    .section    .eh\_frame,"a",@progbits

.LEFDE1:

    .ident  "GCC: (Ubuntu 5.5.0-12ubuntu1~16.04) 5.5.0 20171010"

    .section    .note.GNU-stack,"",@progbits