###### 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 программы, написанной на ассемблерном коде»

студента 2 курса, группы 20203

**Синюкова Валерия Константиновича**

Направление 09.03.01 – «Информатика и вычислительная техника»

Преподаватель:

доцент кафедры параллельных вычислений

Власенко Андрей Юрьевич

Новосибирск 2021

**СОДЕРЖАНИЕ**

[ЗАДАНИЕ 3](#_Toc85619040)

[ОПИСАНИЕ РАБОТЫ 3](#_Toc85619041)

[Описание сделанных оптимизаций 4](#_Toc85619042)

[Результаты замеров времени работы первоначальной и оптимизированной программ 6](#_Toc85619043)

[ЗАКЛЮЧЕНИЕ 7](#_Toc85619044)

[Приложение 1. *Исходный ассемблерный листинг* 8](#_Toc85619045)

[\_Z5func1PdS\_: 8](#_Toc85619046)

[\_Z5func2PdS\_: 9](#_Toc85619047)

[main: 10](#_Toc85619048)

[Приложение 2. *Ассемблерный листинг оптимизированной программы* 11](#_Toc85619049)

[\_Z5func1PdS\_: 11](#_Toc85619050)

[\_Z5func2PdS\_: 12](#_Toc85619051)

[main: 13](#_Toc85619052)

# ЗАДАНИЕ

Формулировка общего задания:

1. Оптимизировать программу, написанную на ассемблере, таким образом, чтобы она осталось корректной ([см. код программы в приложении](#_Приложение_1._Исходный)).
2. Составить отчет по лабораторной работе. Отчет должен содержать следующее:
   * Титульный лист.
   * Задание лабораторной работы.
   * 2 ассемблерный листинга: первоначальный и оптимизированный.
   * Описание всех сделанный оптимизаций с пояснениями.
   * Результаты замеров времени первоначальной программы и оптимизированной – минимум по 3 запуска каждой из программ.

# ОПИСАНИЕ РАБОТЫ

Для начала ассемблерный листинг был проанализирован для того, чтобы понять, что делает программа ([см. код исходного ассемблерного листинга в соответствующем приложении](#_Приложение_1._Исходный)):

[*main*](#_main:):

динамически выделяется память под два массива. Пусть они называются a и b.

[*\_Z5func1PdS\_*](#__Z5func1PdS_:):

оба массива заполняются последовательностью псевдослучайных чисел, которая всегда будет одна и та же, так как начальная точка последовательности всегда будет одинаковой (в программе не вызывается srand()). Также каждый элемент обоих массивов умножается, делится и уменьшается на значения, известные до начала компиляции (т.е. постоянные).

[*\_Z5func2PdS\_*](#__Z5func2PdS_:):

считается сумма s = , где c – значение известное до начала компиляции.

Данная сумма выводится в поток вывода stdout, затем динамическая память, выделенная в начале программы, освобождается.

# 

# Описание сделанных оптимизаций

1. В нашей программе присутствуют два цикла, каждый из которых итерируется по 5\*107 раз, при каждой итерации несколько раз происходит обращение в ОП, для сокращения затрат времени на запись данных в ОП и чтение данных оттуда все переменные были отображены на регистры процессора. Таким образом, наша программа работает со стеком всего лишь в шести местах: в начале и в конце каждой функции, для сохранения и загрузки значения регистров, остальные взаимодействия со стеком были исключены из программы.

Далее приведен список регистров и их назначения в оптимизированной программе:

* %r13 используется для хранения указателя на первый массив.
* %r14 используется для хранения указателя на второй массив.
* %r15 используется для индексации по массивам в циклах функций [\_Z5func1PdS\_](#__Z5func1PdS_:_1) и [\_Z5func2PdS\_](#__Z5func2PdS_:_1).
* %r12 используется в функции [\_Z5func2PdS\_](#__Z5func2PdS_:_1) для промежуточных вычислений, а именно для хранения синуса элемента первого массива.
* %r10 используется в функции [\_Z5func2PdS\_](#__Z5func2PdS_:_1) для хранения итоговой суммы.

|  |  |
| --- | --- |
| **Исходная программа** | **Оптимизированная программа** |
| * movq %rax, -16(%rbp) * movq -16(%rbp), %rax   movq %rax, %rdi | * movq %rax, %r13 * movq %r13, %rdi |
| * movq %rax, -8(%rbp) * movq -8(%rbp), %rax   movq %rax, %rdi | * movq %rax, %r14 * movq %r14, %rdi |
| * movl $0, -20(%rbp) * movl $0, -12(%rbp) | * movq $0, %r15 * movq $0, %r15 |
| * movsd %xmm1, -40(%rbp) | * movq %xmm0, %r12 |
| * movsd %xmm0, -8(%rbp) * movsd -8(%rbp), %xmm1 * movsd -8(%rbp), %xmm0 | * movq %xmm0, %r10 * movq %xmm0, %r10 * movq %r10, %xmm0 |

1. В оптимизированной программе не создаются локальные копии переменных, это позволяет сократить затраты времени на копирование данных и уменьшить место, которое занимают данные, используемые нашей программой. Были удалены следующие фрагменты кода:

[*main*](#_main:):

* movq -8(%rbp), %rdx

movq -16(%rbp), %rax

movq %rdx, %rsi

movq %rax, %rdi

(данный фрагмент встречался в исходном коде дважды, перед вызовом функций \_Z5func1PdS\_ и \_Z5func2PdS\_)

[\_Z5func1PdS\_](#__Z5func1PdS_:):

* movq %rdi, -40(%rbp)

movq %rsi, -48(%rbp)

[\_Z5func2PdS\_](#__Z5func2PdS_:):

* movq %rdi, -24(%rbp)

movq %rsi, -32(%rbp)

1. Был изменен принцип индексации: вместо того, что увеличивать индекс на 1 на каждой итерации, а затем каждый раз для доступа к элементу массива умножать его на 8, индекс увеличивается на 8 на каждой итерации. Соответственно, размер кода, который отвечает за доступ к элементам массива существенно сократился:

[\_Z5func1PdS\_](#__Z5func1PdS_:):

|  |  |
| --- | --- |
| **Исходная программа** | **Оптимизированная программа** |
| cmpl $49999999, -20(%rbp) | cmpq $399999992, %r15 |
| movl -20(%rbp), %eax  cltq  leaq 0(,%rax,8), %rdx  movq -40(%rbp), %rax  leaq (%rdx,%rax), %rbx | leaq (%r13,%r15), %rbx |
| movl -20(%rbp), %eax  cltq  leaq 0(,%rax,8), %rdx  movq -48(%rbp), %rax  leaq (%rdx,%rax), %rbx | leaq (%r14,%r15), %rbx |
| addl $1, -20(%rbp) | addq $8, %r15 |

[\_Z5func2PdS\_](#__Z5func2PdS_:):

|  |  |
| --- | --- |
| **Исходная программа** | **Оптимизированная программа** |
| cmpl $49999999, -12(%rbp) | cmpq $399999992, %r15 |
| 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 | leaq 0(%r13,%r15), %rax  movq (%rax), %xmm0 |
| 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 | leaq 0(%r14,%r15), %rax  movq (%rax), %xmm0 |
| addl $1, -12(%rbp) | addq $8, %r15 |

1. В функции [\_Z5func2PdS\_](#__Z5func2PdS_:_1) умножение на постоянное значение, известное до начала компиляции, было вынесено по закону дистрибутивности, то есть вместо того, чтобы умножать каждое слагаемое на это значение, умножается вся сумма после ее вычисления:

movq %r10, %xmm0

movsd .LC4(%rip), %xmm1

mulsd %xmm1, %xmm0

1. Некоторые промежуточные перемещения данных и вычисления были изменены или удалены.

*Удаленные фрагменты:*

[*main*](#_main:):

* pxor %xmm0, %xmm0
* movq %xmm0, %rax

movq %rax, -24(%rbp)

movq -24(%rbp), %rax

movq %rax, -40(%rbp)

movsd -40(%rbp), %xmm0

*Измененные фрагменты:*

[\_Z5func2PdS\_](#__Z5func2PdS_:):

|  |  |
| --- | --- |
| **Исходная программа** | **Оптимизированная программа** |
| mulsd -40(%rbp), %xmm0  movsd -8(%rbp), %xmm1  addsd %xmm1, %xmm0  movsd %xmm0, -8(%rbp) | movq %r12, %xmm1  mulsd %xmm1, %xmm0  movq %r10, %xmm1  addsd %xmm1, %xmm0  movq %xmm0, %r10 |

# Результаты замеров времени работы первоначальной и оптимизированной программ

|  |  |  |
| --- | --- | --- |
|  | **Первоначальная программа (с)** | **Оптимизированная программа (с)** |
| 1) | user 5,628  sys 0,232 | user 5,061  sys 0,237 |
| 2) | user 5,606  sys 0,244 | user 4,999  sys 0,292 |
| 3) | user 5,608  sys 0,259 | user 5,042  sys 0,248 |

Для каждой версии программы было произведено три замера времени.

В среднем наша программа стала работать на 10% быстрее.

# ЗАКЛЮЧЕНИЕ

В результате нашей работы программа была оптимизирована и стала работать в среднем на 10% быстрее. Было выяснено, что в случае, когда в нашей программе присутствуют циклы, которые итерируются большое количество раз, и на каждой итерации несколько раз происходит взаимодействие с ОП, самым эффективным способом оптимизации будет отображение переменных на регистры процессора.

# Приложение 1. *Исходный ассемблерный листинг*

.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 оптимизированной программы*

.file "prog.cpp"

.text

.globl \_Z5func1PdS\_

.type \_Z5func1PdS\_, @function

# \_Z5func1PdS\_:

.LCFI2:

pushq %rbx

pushq %r15

movq $0, %r15

.L3:

cmpq $399999992, %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 %r10

pushq %r15

pushq %r12

pxor %xmm0, %xmm0

movq %xmm0, %r10

movq $0, %r15

.L7:

cmpq $399999992, %r15

jg .L6

leaq 0(%r13,%r15), %rax

movq (%rax), %xmm0

call sin

movq %xmm0, %r12

leaq 0(%r14,%r15), %rax

movq (%rax), %xmm0

call cos

movq %r12, %xmm1

mulsd %xmm1, %xmm0

movq %r10, %xmm1

addsd %xmm1, %xmm0

movq %xmm0, %r10

addq $8, %r15

jmp .L7

.L6:

movq %r10, %xmm0

movsd .LC4(%rip), %xmm1

mulsd %xmm1, %xmm0

.LCFI6:

popq %r12

popq %r15

popq %r10

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