**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**

«название работы»

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……….………………………………………………………………….3

Задание...….………………………………………………………………….3

Описание работы....………………………………………………………….3

Выводы……………………………………………………………………….4

Заключение.………………………………………………………………….4

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

Приложение 2………………………………………………………………..7

Приложение 3………………………………………………………………..9

**Цели**

1. Знакомство с программной архитектурой x86/x86-64.

2. Анализ ассемблерного листинга программы для архитектуры x86/x86-64 с несколькими различными ключами оптимизации.

**Задание**

1. Изучить программную архитектуру x86/x86-64.

2. Для программы на языке Си (из лабораторной работы 1) сгенерировать ассемблерные листинги для архитектуры x86 и архитектуры x86-64, используя различные уровни комплексной оптимизации.

3. Проанализировать полученные листинги и:

3.1 Сопоставить команды языка Си с машинными командами.

3.2 Определить размещение переменных языка Си в программах на ассемблере (в каких регистрах, в каких ячейках памяти).

3.3 Описать и объяснить оптимизационные преобразования, выполненные компилятором.

3.4 Продемонстрировать использование ключевых особенностей архитектур x86 и x86-64 на конкретных участках ассемблерного кода.

3.5Сравнить различия в программах для архитектуры x86 и архитектуры x86-64. 4.

4. Составить отчет по лабораторной работе.

**Описание работы**

1. На сайте <https://godbolt.org/> сгенерировал ассемблерные листинги (приложение 1, 2 и 3)
2. Прокомментировал полученные листинги.
3. Выделил следующие отличия архитектур x86 и x86-64:
   1. Использование сопроцессора (x86) для вещественной арифметики и использование блока XMM для вещественной арифметики(x86-64)
   2. Размерность регистров
   3. При уровне оптимизации больше –O0 в ассемблерном коде появляются специфичные для архитектуры функции(сmovge и тд).

4) Составил отчёт о выполненной работе.

**Выводы**

Для оптимизации кода программы при переходе от УО –O0 к –O3 происходят следующие изменения:

* уменьшение обращений к памяти -> использование регистровой памяти для переменных
* использование специфичных для архитектуры команд
* векторизация вычислений

**Заключение**

В результате проделанной работы исследовал устройство ассемблерного кода, получаемого из программ на ЯП Си, получил знания об архитектурах x86 и x86-64 и их устройстве и отличительных особенностях, попрактиковался в чтении и понимании ассемблерного кода.

Приложение 1

gcc –O0 –m32 lab3.c

McLorenExp(double, int):

push ebp ;пушим в стек ebp

mov ebp, esp ;esp->ebp

sub esp, 48 ;esp - 48 - место для локальных пер-нных + ret

mov eax, DWORD PTR [ebp+8] ;со стека в регистр eax записываем арг

mov DWORD PTR [ebp-40], eax ;из eax в локальную переменную

mov eax, DWORD PTR [ebp+12] ;со стека в регистр eax записываем арг

mov DWORD PTR [ebp-36], eax ;из eax в локальную переменную

fldz ;пуш на стек сопроцессора для вещественной арифметики

fstp QWORD PTR [ebp-8] ;снимаем со стека сопроцессора в локальную переменную

fld1 ;пуш на стек сопроцессора для вещественной арифметики

fstp QWORD PTR [ebp-16] ;снимаем со стека сопроцессора в локальную ;переменную

mov DWORD PTR [ebp-20], 0 ;обнуляем локальную переменную - счетчик цикла

.L3:

mov eax, DWORD PTR [ebp-20] ; в eax помещаем счетчик цикла

cmp eax, DWORD PTR [ebp+16] ;сравниваем с аргументом N

jge .L2 ;переход при условии >=

fld QWORD PTR [ebp-8] ;кладем в стек сопроцессора

fadd QWORD PTR [ebp-16];складываем с второй локальной переменной (result + ;element)

fstp QWORD PTR [ebp-8] ;кладём результат в локальную переменную

fld QWORD PTR [ebp-16] ;помещаем на стек

fmul QWORD PTR [ebp-40];перемножаем число на вершине стека сопроц с локальной ;переменной

mov eax, DWORD PTR [ebp-20] ; счетчик цикла в eax

add eax, 1 ;инкремент eax

mov DWORD PTR [ebp-44], eax ;eax в лок переменную

fild DWORD PTR [ebp-44] ;преведение типа

fdivp st(1), st ;разность вещественных чисел

fstp QWORD PTR [ebp-16] ;сохранение результата прошлой операции в ;локальную переменную

add DWORD PTR [ebp-20], 1 ;инкремент счетчика цикла

jmp .L3 ;безусловный переход

.L2:

fld QWORD PTR [ebp-8] ;пуш локальной переменной на стек сопроцессора

leave ;восстановление состояний rsp и rbp

ret ;передача управления вызывающей функции

.LC3:

.string "wb"

.LC4:

.string "out.txt"

.LC5:

.string "%lf"

main:

lea ecx, [esp+4] ; адрес esp+4 в регистр ecx

and esp, -16 ;логическое И

push DWORD PTR [ecx-4] ;пуш переменной в стек

push ebp ;пуш ebp

mov ebp esp ;esp ->ebp

push ebx ;пуш ebx

push ecx ;пуш ecx

sub esp, 16 ;esp - 16

sub esp, 12 ;esp - 12 - для лок переменных

push 51200000 ;пуш 51200000 - аргумент функции

call malloc ;вызов маллок

add esp, 16 ;смещение esp на 16

mov DWORD PTR [ebp-20], eax ; eax -> локальная переменная

mov DWORD PTR [ebp-12], 0 ;обнуление счетчика цикла

.L7:

cmp DWORD PTR [ebp-12], 6399999 ;сравнение счетчика цикла и 6399999

jg .L6 ;переход если больше

fild DWORD PTR [ebp-12] ;приведение типа

fld1 ;пуш на стек спрцссора

fdivrp st(1), st ;разность вещественных чисел

mov eax, DWORD PTR [ebp-12] ;

lea edx, [0+eax\*8] ;вычисление смещения для работы с массивом

mov eax, DWORD PTR [ebp-20] ;

lea ebx, [edx+eax] ;вычисление адреса смещения

sub esp, 4 ;выделение на стеке 4 байт

push 1000 ;пуш 1000 на стек

lea esp, [esp-8];сохранение указателя на топ стека с учетом вызова функции

fstp QWORD PTR [esp] ;передача аргумента функции через стек сопроцессора

call McLorenExp(double, int) ;вызов функции

add esp, 16 ;восстановление стека

fstp QWORD PTR [ebx] ;снятие со стеа сопроцессора

add DWORD PTR [ebp-12], 1 ;инкремент счетчика цикла

jmp .L7 ;безусловный переход

.L6:

sub esp, 8 ;место для локальных переменных

push OFFSET FLAT:.LC3 ;пуш на стек строк-констант

push OFFSET FLAT:.LC4 ;пуш на стек строк-констант

call fopen ;вызов fopen

add esp, 16 ;восстановление стека

mov DWORD PTR [ebp-24], eax ;результат fopen через eax в локвльную переменную out

mov DWORD PTR [ebp-16], 0 ;обнуление счетчика цикла

.L9:

cmp DWORD PTR [ebp-16], 6399999 ;сравнение счетчика

jg .L8 ;переход если больше

mov eax, DWORD PTR [ebp-16] ;подсчет смещения

lea edx, [0+eax\*8] ;смещение для элемента массива в соотв со значением счетчика цикла

mov eax, DWORD PTR [ebp-20]

add eax, edx

fld QWORD PTR [eax]

lea esp, [esp-8]

fstp QWORD PTR [esp]

push OFFSET FLAT:.LC5 ;пуш аргумента (константной строки) для fprintf

push DWORD PTR [ebp-24]

call fprintf ;вызов fprintf

add esp, 16

add DWORD PTR [ebp-16], 1 ;инкремент счетчика цикла

jmp .L9

.L8:

sub esp, 12 ;работа со стеком для передачи аргументов в функцию

push DWORD PTR [ebp-20]

call free ;вызов free

add esp, 16 ;работа со стеком для передачи аргументов в функцию

sub esp, 12

push DWORD PTR [ebp-24]

call fclose ;вызов функции fclose

add esp, 16

mov eax, 0

lea esp, [ebp-8]

pop ecx

pop ebx

pop ebp

lea esp, [ecx-4];восстановление состояния стека до вызова функции + 4bytes ;для возвращаемого значения

ret ;возврат управления вызывающей функции

Приложение 2

gcc –O0 lab3.c

McLorenExp(double, int):

push rbp ;запись rbp на стек

mov rbp, rsp ;перемещение rbp -> rsp

movsd QWORD PTR [rbp-40], xmm0;запись аргументов на стек

mov DWORD PTR [rbp-44], edi ;запись аргументов на стек

pxor xmm0, xmm0 ;обнуление переменной через xor

movsd QWORD PTR [rbp-8], xmm0 ;

movsd xmm0, QWORD PTR .LC1[rip];

movsd QWORD PTR [rbp-16], xmm0;

mov DWORD PTR [rbp-20], 0 ;обнуление счетчика i

.L3:

mov eax, DWORD PTR [rbp-20];перемещение значения счетчика i в регистр eax

cmp eax, DWORD PTR [rbp-44] ;сравнение eax и второго аргумента функции

jge .L2 ;условный переход

movsd xmm0, QWORD PTR [rbp-8] ;помещаем значение переменной по адресу ;rbp-8 (переменная result) в регистр xmm0

addsd xmm0, QWORD PTR [rbp-16];xmm0 = xmm0 + значение переменной по адресу ;rbp-16(переменная element)

movsd QWORD PTR [rbp-8], xmm0 ;результат сложения на прошлом шаге из xmm0 ;перемещаем в [rbp-8](переменная result)

movsd xmm0, QWORD PTR [rbp-16];в регистр xmm0 перемещаем значение переменной по ;адресу [rbp-16](переменная element)

mulsd xmm0, QWORD PTR [rbp-40];перемножение (xmm0 ~ переменная element) и QWORD ;PTR[rbp - 40] ~ первый аргумент функции (x)

mov eax, DWORD PTR [rbp-20] ;перемещение DWORD PTR [rbp-20] ~ счетчик цикла ;i в регистр eax

add eax, 1 ;инкремент eax

pxor xmm1, xmm1 ;обнуление переменной через xor

cvtsi2sd xmm1, eax ;приведение типов операндов

divsd xmm0, xmm1 ;деление вещественных операндов

movsd QWORD PTR [rbp-16], xmm0;перемещение результата деления в переменную ;element

add DWORD PTR [rbp-20], 1 ;инкремент счетчика цикла i

jmp .L3 ;безусловный переход к началу цикла по метке .L3

.L2:

movsd xmm0, QWORD PTR [rbp-8] ;перемещение значение переменной result

movq rax, xmm0 ;в регистр rax

movq xmm0, rax ;

pop rbp ;взятие адреса со стека

ret ;возврат управления вызвавшей функции

.LC2:

.string "wb"

.LC3:

.string "out.txt"

.LC4:

.string "%lf"

main: ;

push rbp ;запись rbp на стек

mov rbp, rsp ;копирование из rsp в rbp

push rbx ;запись rbx на стек

sub rsp, 40 ;вычитание из rsp 40

mov edi, 51200000 ;копирование в edi 5120000

call malloc ;вызов функции маллок

mov QWORD PTR [rbp-32], rax ;перемещение в переменную arr результата работы функции malloc, помещённый в rax

mov DWORD PTR [rbp-20], 0 ;обнуление счетчика цикла

.L7:

cmp DWORD PTR [rbp-20], 6399999 ;сравнение счетчика цикла в for()

jg .L6 ;переход по метке, если больше

pxor xmm1, xmm1 ;обнуление xmm1 xor-ом

cvtsi2sd xmm1, DWORD PTR [rbp-20] ;приведение типов

movsd xmm0, QWORD PTR .LC1[rip] ;помещение в xmm0константы 1.0

divsd xmm0, xmm1 ;деление константы 1.0 на приведённый к double счетчик цикла

movq rax, xmm0 ;результат деления -> rax

mov edx, DWORD PTR [rbp-20] ;значение указателя счетчика -> edx

movsx rdx, edx ;edx->rdx

lea rcx, [0+rdx\*8] ;смещение от начала массива arr -> rcx

mov rdx, QWORD PTR [rbp-32] ;значение переменной arr -> rdx

lea rbx, [rcx+rdx] ;сумма адреса начала массива arr и смещения, посчитанного и помещенного в rcx

mov edi, 1000 ;копирование 1000 -> edi(второй аргумент функции McLorenExp)

movq xmm0, rax ;

call McLorenExp(double, int) ;вызов функции McLorenExp

movq rax, xmm0 ;результат функции McLorenExp -> rax

mov QWORD PTR [rbx], rax ;rax -> высчитанное место в массиве arr

add DWORD PTR [rbp-20], 1 ;инкремент счетчика цикла

jmp .L7 ;безусловный переход к начаул цикла

.L6:

mov esi, OFFSET FLAT:.LC2 ;помещение в esi и edi стрковых констант

mov edi, OFFSET FLAT:.LC3 ;аргументы функции fopen

call fopen ;вызов функции fopen

mov QWORD PTR [rbp-40], rax ;результат функции fopen() из rax -> переменную out на стеке

mov DWORD PTR [rbp-24], 0

.L9:

cmp DWORD PTR [rbp-24], 6399999 ;сравнение

jg .L8 ;условный переход(если больше, то перейти по метке .L8)

mov eax, DWORD PTR [rbp-24] ;

cdqe ;какая-то магия (что-то вроде удвоения размерности знаковой части)

lea rdx, [0+rax\*8] ;расчет смещения относительно начала массива arr

mov rax, QWORD PTR [rbp-32] ;значение arr -> rax

add rax, rdx ;прибавление к rax(содержит arr) смещения в соответствии со ;значением счетчика цикла

mov rdx, QWORD PTR [rax]

mov rax, QWORD PTR [rbp-40] ;указатель на дескриптор файла(out) -> rax

movq xmm0, rdx

mov esi, OFFSET FLAT:.LC4 ;помещение в esi аргумента функции fprintf

mov rdi, rax ;помещение в rdi аргумента функции fprintf

mov eax, 1

call fprintf ;вызов функции fprintf

add DWORD PTR [rbp-24], 1 ;инкремент счетчика цикла

jmp .L9 ;безусловный переход к началу цикла

.L8:

mov rax, QWORD PTR [rbp-32] ;помещаем arr -> rax

mov rdi, rax ;из rax -> rdi для передачи в качестве аргумента функции free

call free ;вызов функции free

mov rax, QWORD PTR [rbp-40] ;аналогично помещение

mov rdi, rax ;указателя на дескриптор файла в rdi для передачи в качестве аргумента

call fclose ;вызов функции fclose

mov eax, 0 ;return 0

mov rbx, QWORD PTR [rbp-8];выглядит как восстановления состояния регистра rbx

leave ;восстановление состояния стека до вызова функции main

ret ;передача управления вызвавшему процессу

.LC1:

.long 0

.long 1072693248

Приложение 3

gcc –O2 lab3.c

McLorenExp(double, int):

movapd xmm3, xmm0 ; xmm0->xmm3

test edi, edi ;поразрядное & и установка значений бит в ;регистре eflags

jle .L4 ;переход , если меньше или равно

movsd xmm0, QWORD PTR .LC0[rip]; 1.0 -> xmm0

xor eax, eax ;обнуление eax xor-ом

pxor xmm1, xmm1 ;аналогично - xmm1

.L3:

addsd xmm1, xmm0 ;сложение xmm1 и xmm0 (xmm1== xmm0)

mulsd xmm0, xmm3 ;перемножение element \* x

pxor xmm2, xmm2 ;обнуление xor-ом

add eax, 1 ;выражение i + 1

cvtsi2sd xmm2, eax ;приведение типа

divsd xmm0, xmm2 ;операция деления

cmp eax, edi ;сравнение счетчика цикла и аргумента N функции McLorenExp

jne .L3 ;если не равны - переход

movapd xmm0, xmm1 ;xmm1->xmm0

ret ;возврат управления вызывающей функции

.L4:

pxor xmm1, xmm1 ;обнуление xor-ом

movapd xmm0, xmm1 ;xmm1->xmm0

ret ;возврат управления вызывающей функции

.LC2:

.string "wb"

.LC3:

.string "out.txt"

.LC4:

.string "%lf"

main:

push r13 ;помещаем на стек к13

mov edi, 51200000 ;5120000 -> edi для передачи в качестве ;аргумента

push r12 ;помещаем на стек к12

push rbp ;помещаем на стек rbp

push rbx ;помещаем на стек rbx

sub rsp, 8 ;уменьшаем rsp для локальных переменных

call malloc ;вызов функции malloc

movsd xmm4, QWORD PTR .LC0[rip] ;0 -> xmm4

xor edx, edx ;обнуляем edx

pxor xmm5, xmm5 ;обнуляем xmm5

mov rbp, rax ;rax->rbp

.L9:

pxor xmm0, xmm0 ;обнуляем xmm0

movapd xmm3, xmm4 ;xmm4->xmm3

movapd xmm1, xmm5 ;xmm5->xmm1

xor eax, eax ;eax = 0

cvtsi2sd xmm0, edx ;приведение типа

divsd xmm3, xmm0 ;деление

movapd xmm0, xmm4 ;xmm0 <- xmm4

.L8:

addsd xmm1, xmm0 ;сложение xmm1+= xmm0

mulsd xmm0, xmm3 ;xmm0\*=xmm3

pxor xmm2, xmm2 ;xmm = 0.0

add eax, 1 ;eax+=1

cvtsi2sd xmm2, eax ;приведение типа

divsd xmm0, xmm2 ;xmm0/= xmm2

cmp eax, 1000 ;сравнение eax и 1000

jne .L8 ;переход если не равны

movsd QWORD PTR [rbp+0+rdx\*8], xmm1 ;помещаем вычесленное значение в ячейку ;массива с некоторым смещение от начала массива arr

add rdx, 1 ;инкремент счетчика цикла

cmp rdx, 6400000 ;сравнение счетчика и 6400000

jne .L9 ;переход если не равны

mov esi, OFFSET FLAT:.LC2 ;константные строки

mov edi, OFFSET FLAT:.LC3 ;в регистры esi и edi

mov rbx, rbp ;для передачи в качестве аргументов функции fopen

call fopen ;вызов fopen

lea r13, [rbp+51200000] ;подсчет нового смещения

mov r12, rax ;r12 <- rax

.L10:

movsd xmm0, QWORD PTR [rbx]

mov esi, OFFSET FLAT:.LC4 ;аргументов в регистры

mov rdi, r12 ;для передачи в функцию fprintf

add rbx, 8 ;смещение для перехода к след элементу массива

mov eax, 1 ;инкремент счетчика

call fprintf ;вызов функции fprintf

cmp rbx, r13 ;сравнение

jne .L10

mov rdi, rbp ;rbp->rdi передача аргумента для функции free

call free ;вызов функции free

mov rdi, r12 ;12-> rdi

call fclose ;вызов функции fclose

add rsp, 8 ;восстановление регистра rsp

xor eax, eax ;обнуление eax

pop rbx ;вытаскиваем

pop rbp ;из

pop r12 ;стека

pop r13 ;всё что туда клали

ret ;передача управления вызываюшему потоку

.LC0:

.long 0

.long 1072693248