Отчёт по лабораторной работе №10

дисциплина: Архитектура компьютера

Максим Александрович Мишонков

Содержание

Список иллюстраций

1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ с использованием подпрограмм, знакомство с методами откладки при помощи GDB и его основными возможностями.

2 Задание

Научиться писать программы с использованием подпрограмм, ознакомиться с методами откладки при помощи GDB и его основными возможностями.

3 Теоретическое введение

Откладка - это процесс поиска и исправления ошибок в программе (обнаружение ошибки, поиск её местонахождения, определение причины ошибки, исправление ошибки)

Точки останова - это специально отмеченные места в программе, в которых программа-откладчик приостанавливает выполнение программы и ждёт команд.

Откладчик GDB позволяет увидеть, что происходит “внутри” программы в момент её выполнения и что делает программа в момент сбоя. GDB может начать выполнение программы, задав всё, что может повлиять на её поведение; остановить программу при указанных условиях; исследовать, что случилось, когда программа остановилась; изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других.

Подпрограмма - это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы.

Если в подпрограмме встречается одинаковый участок кода, его можно оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма будет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы.

Для вызова подпрограммы из основной программы применяется инструкция call, которая заносит адрес следующей инструкции в стек и загружает в регистр eip адрес соответствующей подпрограммы. Завершается подпрограмма инструкцикй ret, которая извлекает из стека адрес, занесённый туда соответствующей

инструкцией call, и заносит его в eip.

4 Выполнение лабораторной работы

1. Создал каталог для выполнения лабораторной работы №10, перешёл в него и создал файл lab10-1.asm. (рис. 4.1)

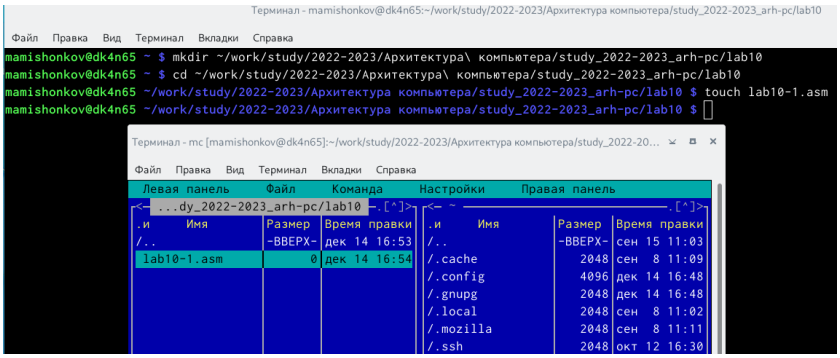

Рис. 4.1: Создание каталога и файла

2. Ввёл в файл lab10-1.asm текст программы из листинга 10.1. (рис. 4.2)

```
 GNU nano 6.3                  /afs/.dk.sci.pfu.edu.ru/home/m/a/mamishonkov
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: RESB 80
SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi
call _calcul
mov eax,result
call sprint
mov eax,[rez]
call iprintLF
call quit

_calcul:
mov ebx,2
mul ebx
add eax,7
mov [rez],eax
ret
```

Рис. 4.2: Введение текста программы

3. Создал исполняемый файл и проверил его работу. (рис. 4.3)

```
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ nasm -f elf lab10-1.asm
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-1
Введите x: 5
2x+7=17
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-1
Введите x: 6
2x+7=19
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $
```

Рис. 4.3: Проверка работы исполняемого файла

4. Изменил текст программы, добавив подпрограмму _subcalcul в подпрограмму _calcul, для вычисления выражения f(g(x)), где x вводится с клавиатуры, f(x)=2x+7, g(x)=3x-1. Проверил работу исполняемого файла. (рис. 4.4, 4.5, 4.6)

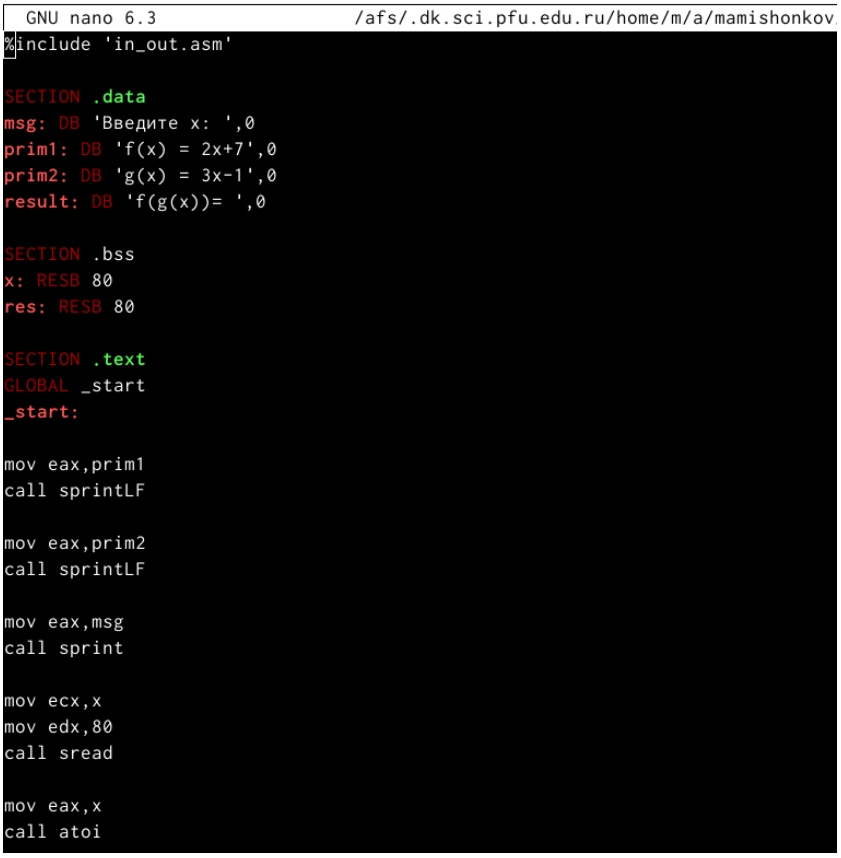

Рис. 4.4: Изменение текста программы

```
call _calcul

mov eax,result
call sprint
mov eax,[res]
call iprintLF

call quit

_calcul:

call _subcalcul

mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret

_subcalcul:

mov ebx,3
mul ebx
sub eax,1
ret
```

Рис. 4.5: Изменение текста программы

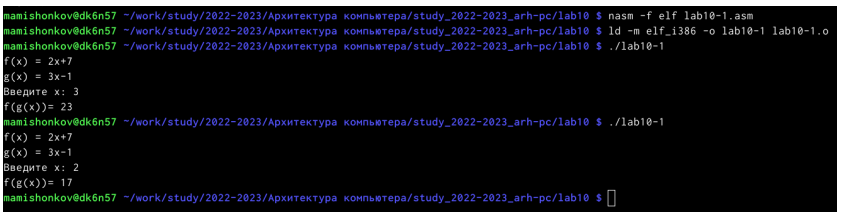

Рис. 4.6: Проверка работы исполняемого файла

5. Создал файл lab10-2.asm. (рис. 4.7)

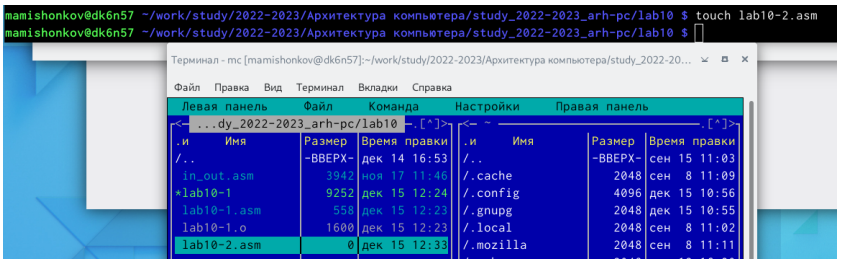

Рис. 4.7: Создание файла

6. Ввёл в файл lab10-2.asm текст программы из листинга 10.2. (рис. 4.8)

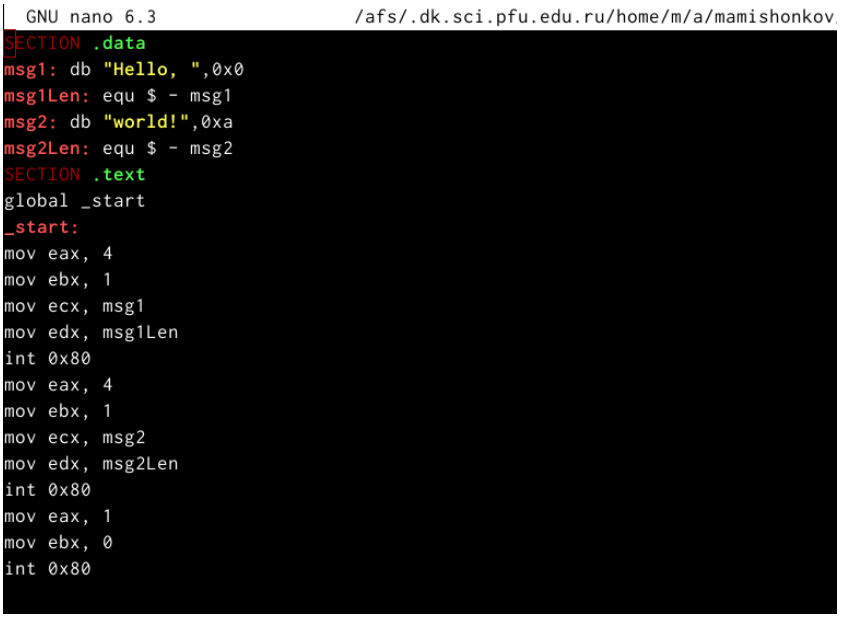

Рис. 4.8: Введение текста программы

7. Получил исполняемый файл. Для работы с GDB в исполняемый файл добавил отладочную информацию, проведя трансляцию программ с ключом ‘-g’. Загрузил исполняемый файл в отладчик gdb. Проверил работу программы, запустив её в оболочке GDB с помощью команды run. (рис. 4.9)

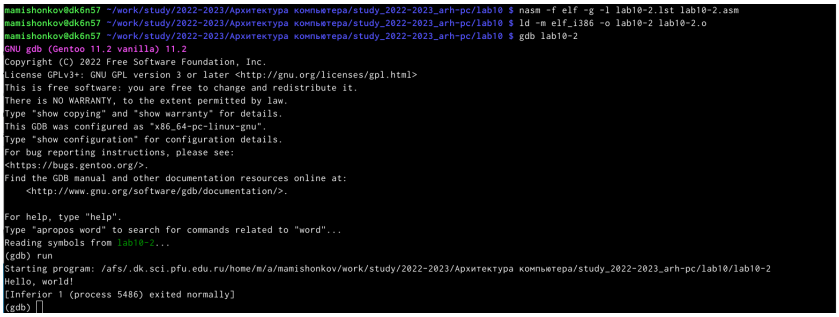

Рис. 4.9: Проверка работы исполняемого файла

8. Для более подробного анализа программы установил брейкпоинт на метку _start, с которой начинается выполнение любой ассемблерной программы, и запустил её. (рис. 4.10)

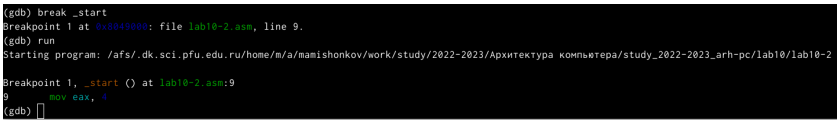

Рис. 4.10: Запуск программы

11. Посмотрел дисассимилированный код программы с помощью команды disassemble, начиная с метки _start. (рис. 4.11)

Рис. 4.11: Просмотр дисассимилированного кода программы

12. Переключился на отображение команд с Intel'овским синтаксисом, введя команду set disassembly-flavor intel. (рис. 4.12)

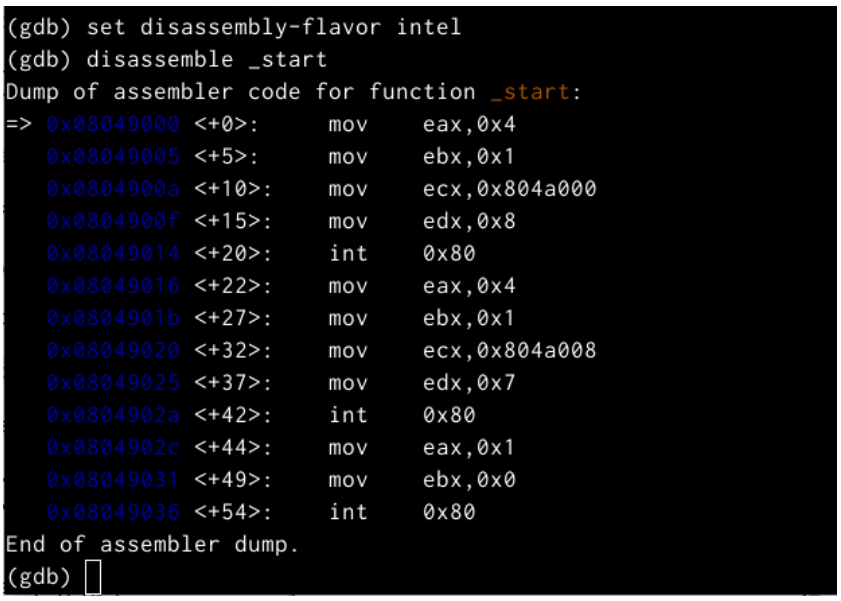

Рис. 4.12: Переключение на отображение команд с Intel'овским синтаксисом

13. Включил режим псевдографики для более удобного анализа программы. (рис. 4.13, 4.14)

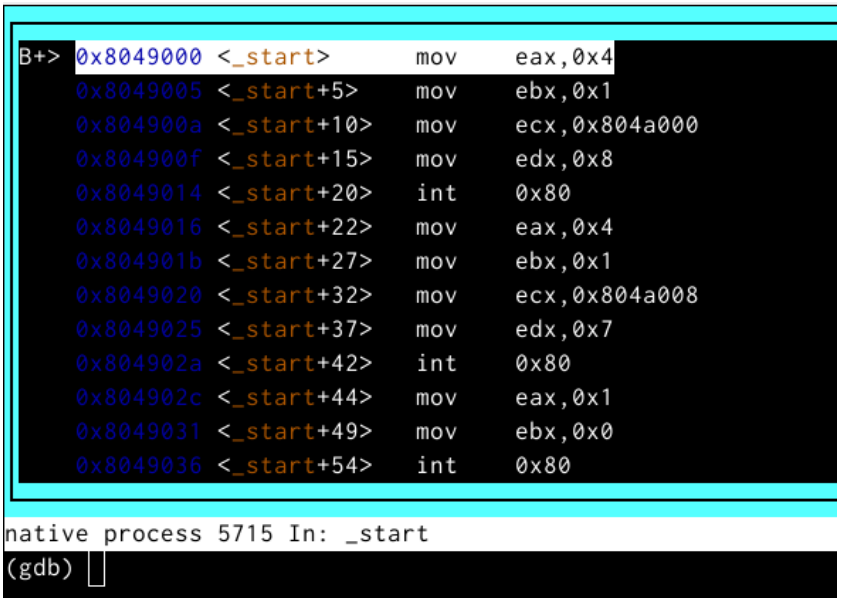

Рис. 4.13: Включение режима псевдографики

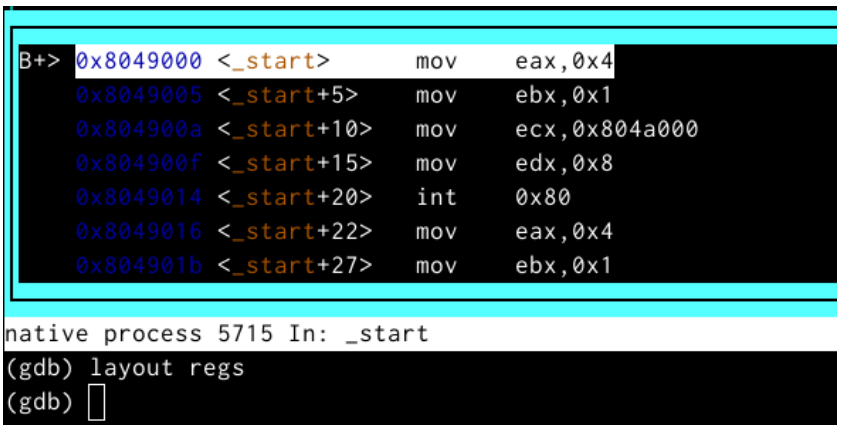

Рис. 4.14: Включение режима псевдографики

14. На предыдущих шагах была установлена точка останова по имени метки (_start). Проверил это с помощью команды info breakpoints. (рис. 4.15)

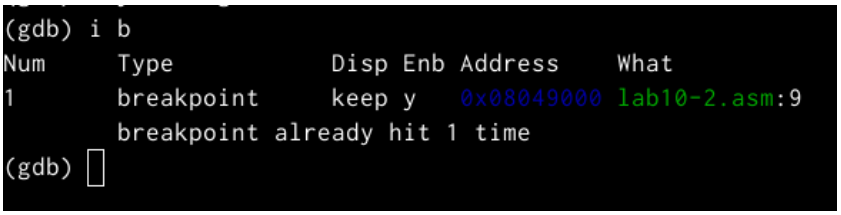

Рис. 4.15: Проверка точки останова по имени метки (_start))

15. Установил ещё одну точку останова по адресу инструкции, определив адрес предпоследней инструкции. (рис. 4.16)

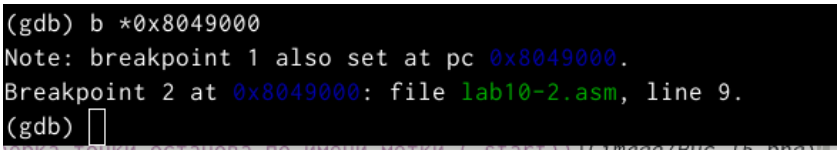

Рис. 4.16: Установка ещё одной точки останова по адресу инструкции

16. Посмотрел информацию о всех установленных точках установа. (рис. 4.17)

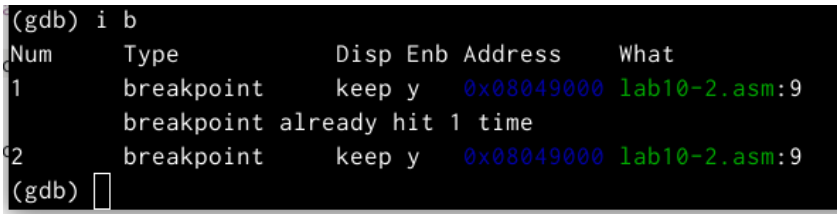

Рис. 4.17: Просмотр информации о всех установленных точках установа

17. Посмотрел содержимое регистров с помощью команды info registers. (рис. 4.18, 4.19)

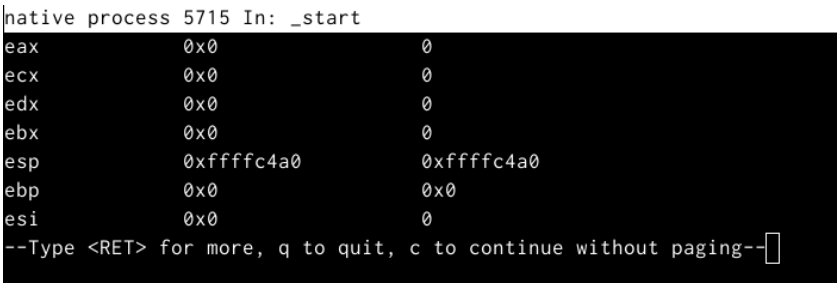

Рис. 4.18: Просмотр содержимого регистров

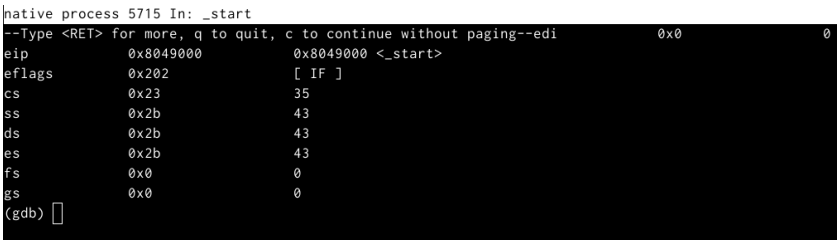

Рис. 4.19: Просмотр содержимого регистров

18. Посмотрел значение переменной msg1. (рис. 4.20)

Рис. 4.20: Значение переменной msg1

19. Посмотрел значение переменной msg2. (рис. 4.21)

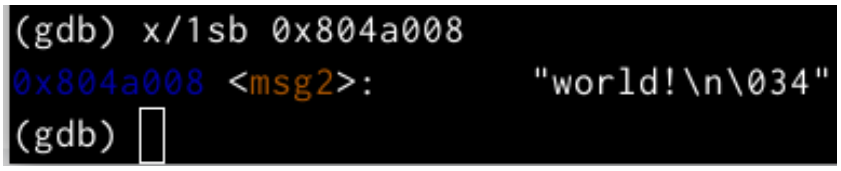

Рис. 4.21: Значение переменной msg2

20. Изменил первый символ переменной msg1. (рис. 4.22)

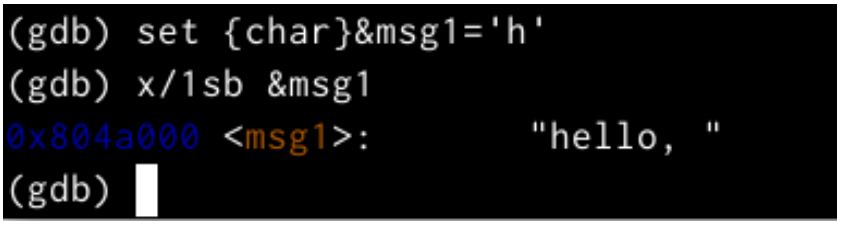

Рис. 4.22: Изменение первого символа переменной msg1

21. Заменил один из символов во второй переменной msg. (рис. 4.23)

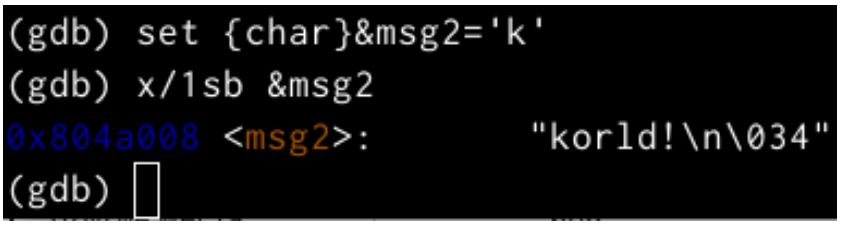

Рис. 4.23: Замена символа во второй переменной msg

21. Вывел в различных форматах значение регистра edx. (рис. 4.24)

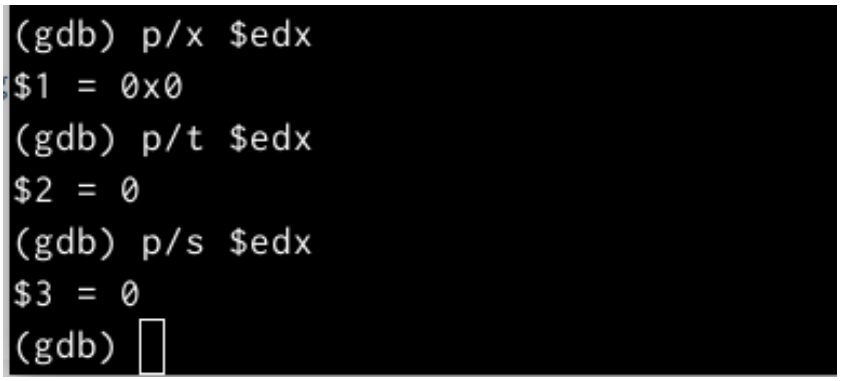

Рис. 4.24: Вывод в различных форматах значения регистра edx

22. С помощью команды set изменил значение регистра ebx. (рис. 4.25)

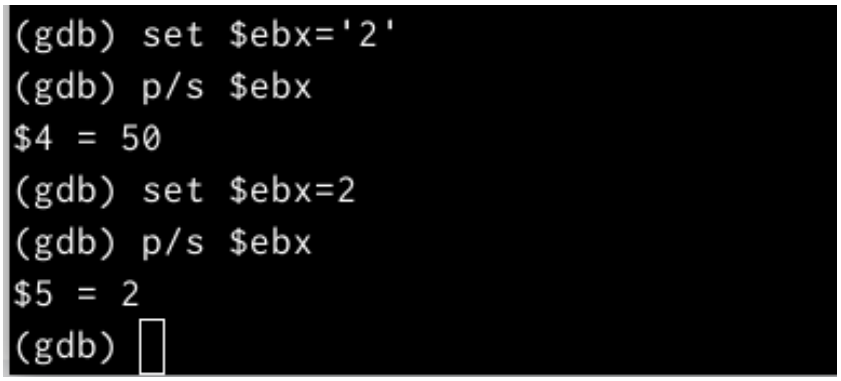

Рис. 4.25: Изменение значения регистра ebx

23. Скопировал файл lab9-2.asm, созданный при выполнении лабораторной работы №9, с программой, выводящей на экран аргументы командной строки, в файл с именем lab10-3.asm. (рис. 4.26)

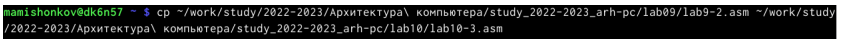

Рис. 4.26: Копирование файла lab9-2.asm в файл lab10-3.asm

24. Создал исполняемый файл, загрузил его в отладчик, указав аргументы. (рис. 4.27)

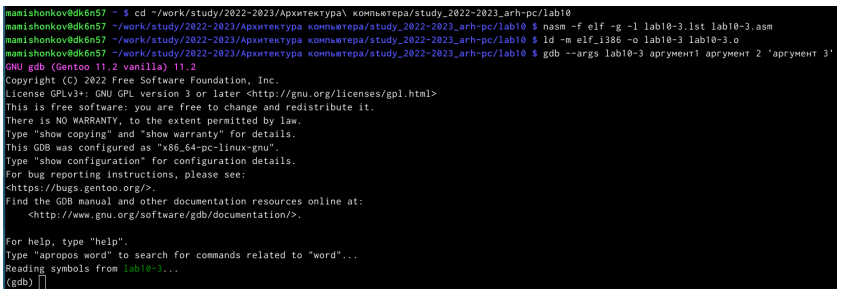

Рис. 4.27: Создание исполняемого файла и загрузка его в отладчик

25. Установил точку останова перед первой инструкцией в программе и запустил программу. (рис. 4.28)

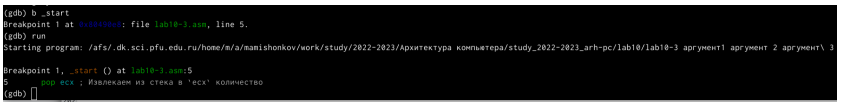

Рис. 4.28: Установка точки останова перенд первой инструкцией в программе и её запуск

26. Адрес вершины стека хранится в регистре esp и по этому адресу располагается число, равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5. Это имя программы lab10-3.asm. (рис. 4.29)

Рис. 4.29: Команда x/x $esp

27. Просмотрел остальные позиции стека. По адресу esp+4 располагается адрес в памяти, где находится имя программы, по адресу esp+8 хранится адрес первого аргумента, по адресу esp+12 - второго аргумента и т.д. (рис. 4.30)

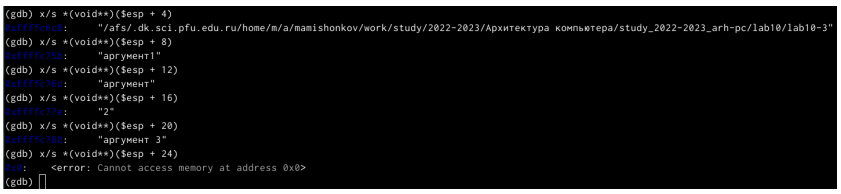

Рис. 4.30: Просмотр остальных позиций стека

Самостоятельная работа

1. Преобразовал программу из лабораторной работы №9 (задание 1 из самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму. Создал исполняемый файл и проверил его работу. (рис. 4.31, 4.32, 4.33)

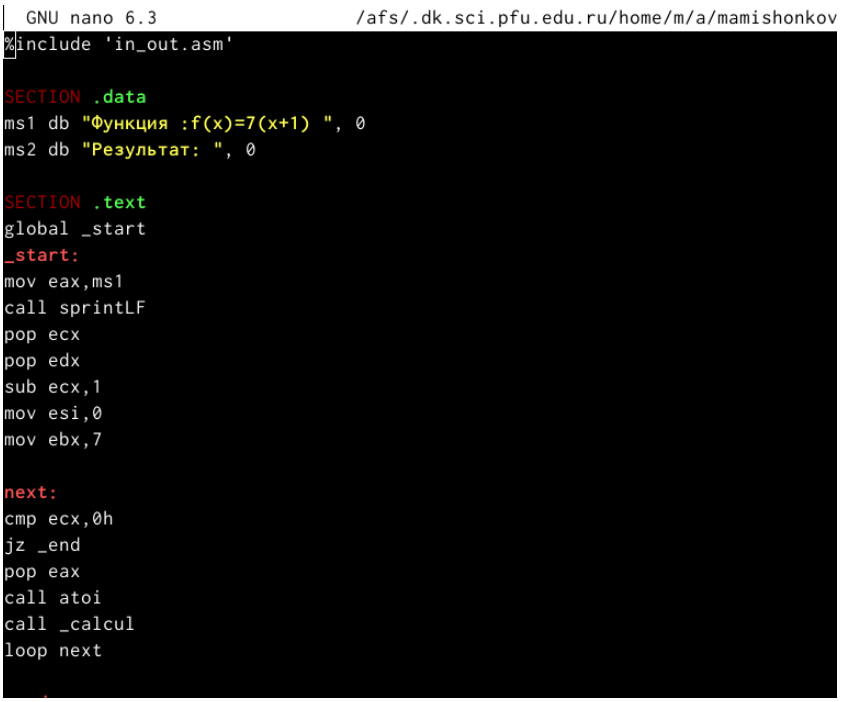

Рис. 4.31: Преобразование программы

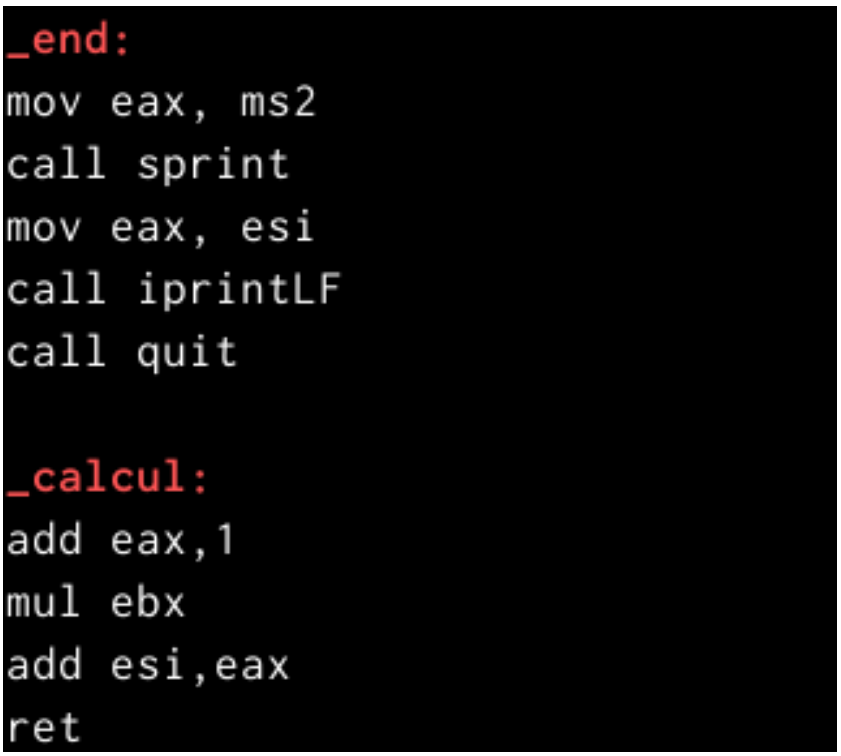

Рис. 4.32: Преобразование программы

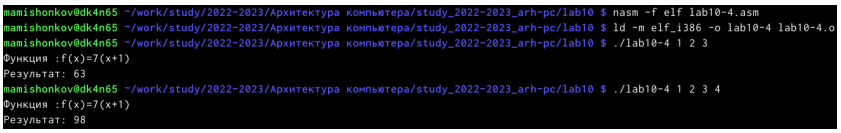

Рис. 4.33: Проверка работы исполняемого файла

2. С помощью отладчика GDB, анализируя изменения значений регистров, определил и исправил ошибку в программе вычисления выражения (3+2)*4+5. В результате программа должна выглядеть следующим образом. Создал исполняемый файл и проверил его работу. (рис. 4.34, 4.35)

```
 GNU nano 6.3                  /afs/.dk.sci.pfu.edu.ru/home/m/a/mamishonkov
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 4.34: Изменение текста программы

```
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ touch lab10-5.asm
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ nasm -f elf lab10-5.asm
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ld -m elf_i386 -o lab10-5 lab10-5.o
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-5
Результат: 25
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $
```

Рис. 4.35: Проверка работы исполняемого файла

5 Выводы

В ходе выполнения данной лабораторной работы я научился писать программы с использованием подпрограмм, ознакомился с методами откладки при помощи GDB и его основными возможностями.