Отчёт по лабораторной работе №8

Дисциплина: Комьютерные технологии и технологии программирования

Дымовой Д.Д.

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа: • обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки. Можно выделить следующие типы ошибок: • синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают пре- рывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Наиболее часто применяют следующие методы отладки: • создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообще- ния); • использование специальных программ-отладчиков.

3 Выполнение лабораторной работы

Создаю каталог для выполнения лабораторной работы №9, перехожу в него, создаю ассемблеровский файл для работы (рис. 3.1).

```
dddihmova@dk5n55 ~ $ mkdir ~/work/arch-pc/lab09
dddihmova@dk5n55 ~ $ cd ~/work/arch-pc/lab09
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ touch lab09-1.asm
```

Рис. 3.1: Создание каталога

Копирую текст листинга 9.1, создаю исполняемый файл и запускаю его (рис. 3.2).

```
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 5
2x+7=17
```

Рис. 3.2: Запуск исполняемого файла

Изменяю текст программы листинга 9.1, чтобы программа работала так, как указано в условии лабораторной работы (рис. 3.3).

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB 'f(g(x))=',0
SECTION .bss
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:

;-------------------------------------------
; Основная программа
;-------------------------------------------
mov eax, msg
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax,x
call atoi

call _subcalcul
call _calcul ; Вызов подпрограммы _calcul

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

call quit
;-------------------------------------------
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx,2
mul ebx
add eax,7
mov [res], eax
ret
_subcalcul:
mov ebx, 3
mul ebx
add eax, -1
mov [res],eax
ret ; выход из подпрограммы
```

Рис. 3.3: Программа

Создаю исполняемый файл и запускаю его (рис. 3.4).

```
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 1
f(g(x))=11
```

Рис. 3.4: Программа f(g(x))

Создаю файл lab09-2.asm, вставляю туда текст программы листинга 9.2, сохраняю, создаю исполняемый файл и запускаю его (рис. 3.5).

```
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 12.1 vanilla) 12.1
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
Find the GDB manual and other documentation resources online at:
--Type <RET> for more, q to quit, c to continue without paging--(gdb) run
```

Рис. 3.5: Запуск файла

Запускаю файл в оболочке GDB (рис. 3.6).

```
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/d/dddihmova/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4419) exited normally]
(gdb)
```

Рис. 3.6: Оболочка GDB

Устанавливаю брейкпоинт на метку _start (рис. 3.7).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/d/dddihmova/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9       mov eax, 4
```

Рис. 3.7: Установка брейкпоинта

Просматриваю дисассимилированный код программы начиная с метки (рис. 3.8).

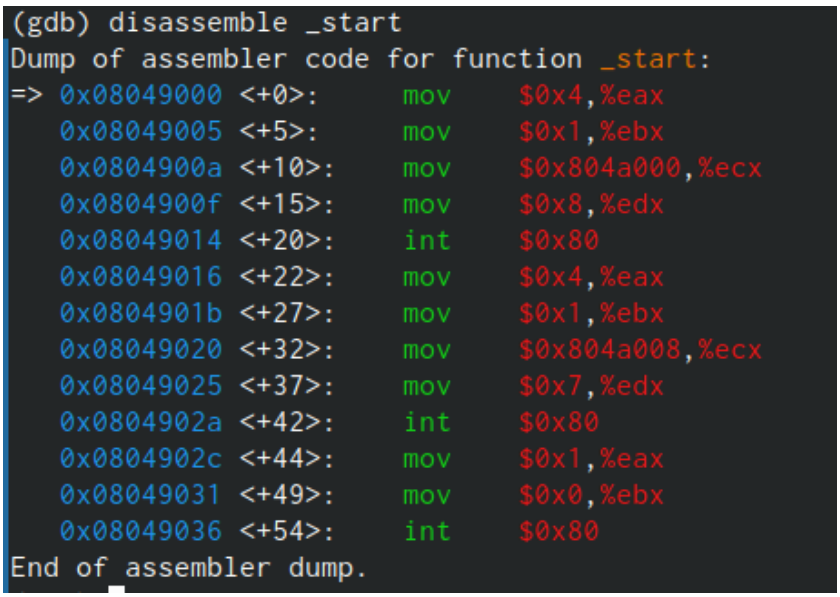

Рис. 3.8: Просмотр кода программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду set disassembly-flavor intel (рис. 3.9).

Рис. 3.9: Переключение

ПЕРЕЧИСЛИТЬ РАЗЛИЧИЯ ОТОБРАЖЕНИЯ СИНТАКСИСА

Включаю режим псевдографики для более удобного анализа программы (рис. 3.10).

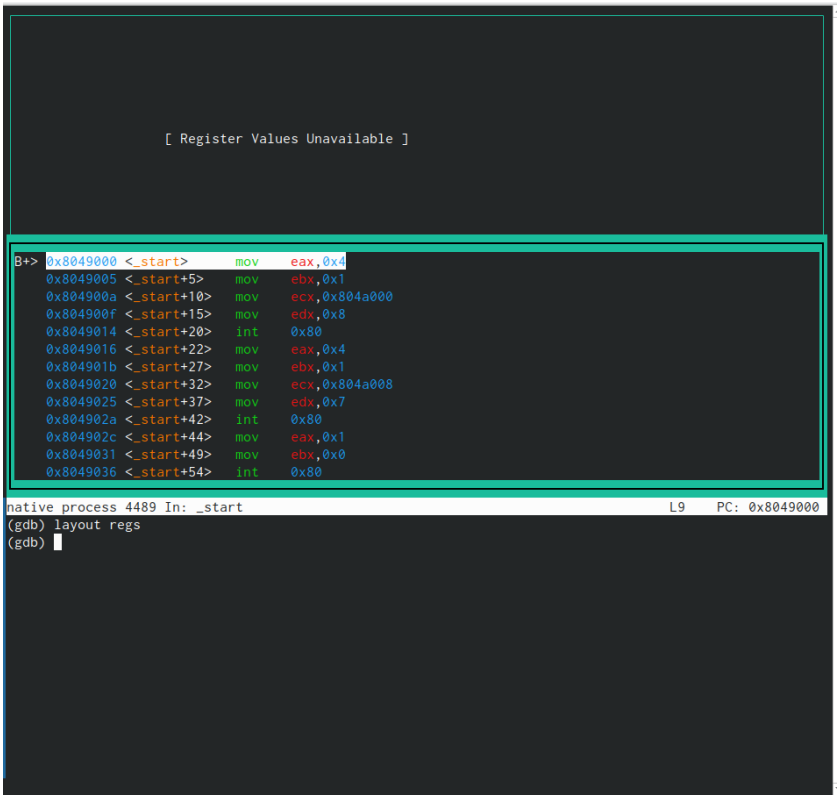

Рис. 3.10: Переключение режима

Просмотр точек брейкпоинта (рис. 3.11).

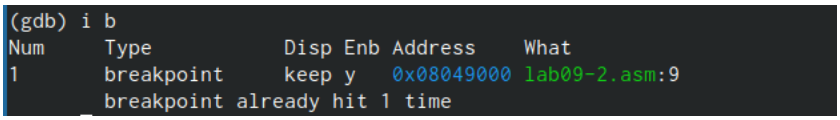

Рис. 3.11: Точки брейкпоинта

Устанавливаю брейкпоинт на инструкцию mov ebx, 0x0 (рис. 3.12).

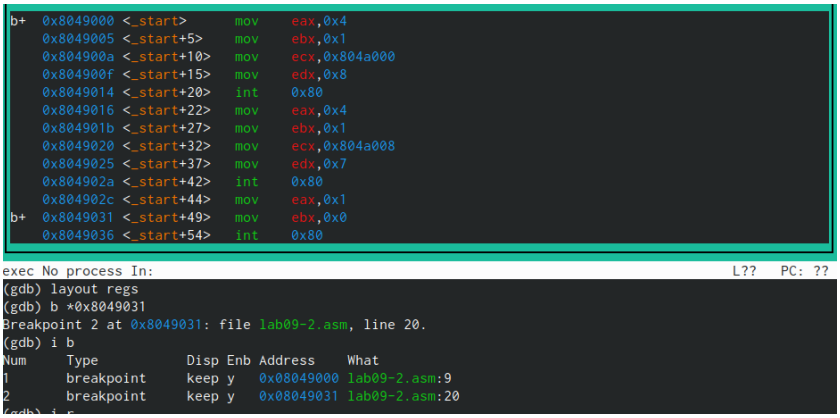

Рис. 3.12: Брейкпоинт

Информация о всех точках останова (рис. 3.13).

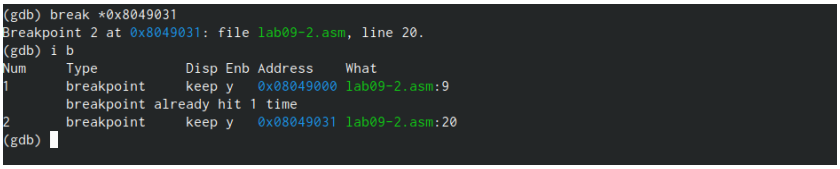

Рис. 3.13: Точки останова

Просматриваю содержимое регистров с помощью команды i r (рис. 3.14).

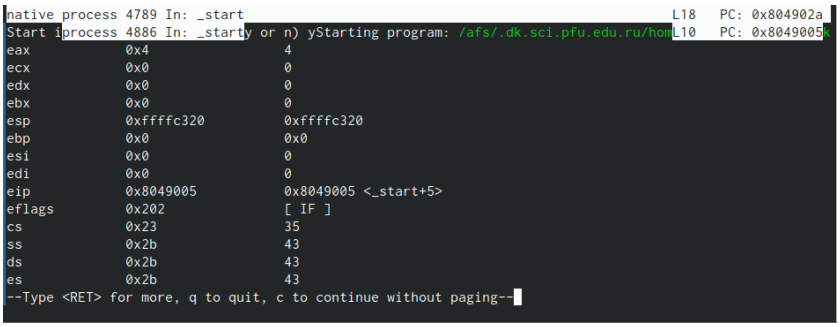

Рис. 3.14: Информация о содержимом регистров

Смотрю значение переменной msg1 (рис. 3.15).

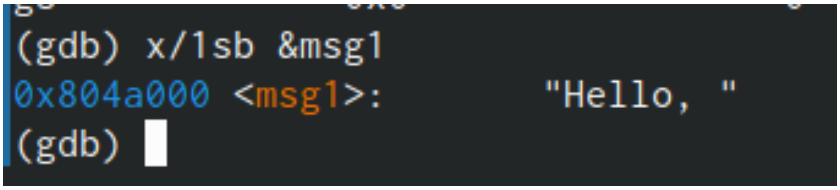

Рис. 3.15: Значение переменной

Заменяю значение переменной msg1 (рис. 3.16).

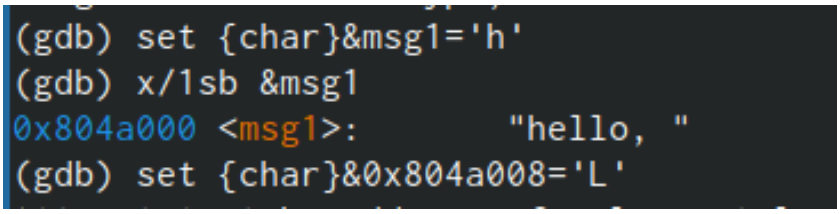

Рис. 3.16: Изменение переменной msg1

Просматриваю значение аргумента msg2. 0x804a008 адрес msg2 (рис. 3.17).

Рис. 3.17: Значение аргумента msg2

Просматриваю значение регистра ebx в разных форматах (рис. 3.18).

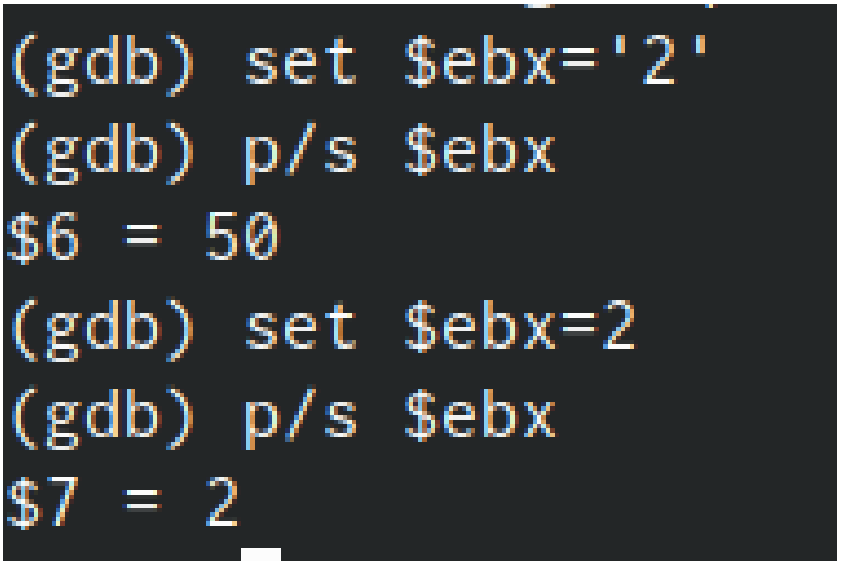

Рис. 3.18: Значение регистра ebx

Завершаю выполнение программы с помощью команды q (рис. 3.19).

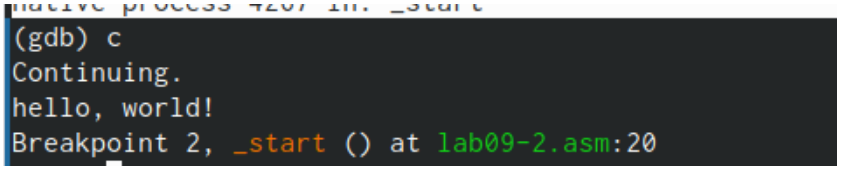

Рис. 3.19: Завершение программы

Выхожу из gdb с помощью команды q (рис. 3.20).

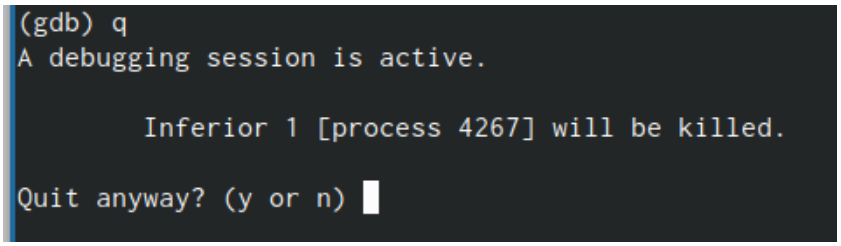

Рис. 3.20: Выход из откладки

Я копирую файл lab8-2.asm в папку с лабораторной номер 9 и называю его lab09-2.asm, создаю исполняемый файл и запускаю его через gdb. Эта программы

должна находить произведение аргументов. Также я ставлю точку останова на месту старт (рис. 3.21).

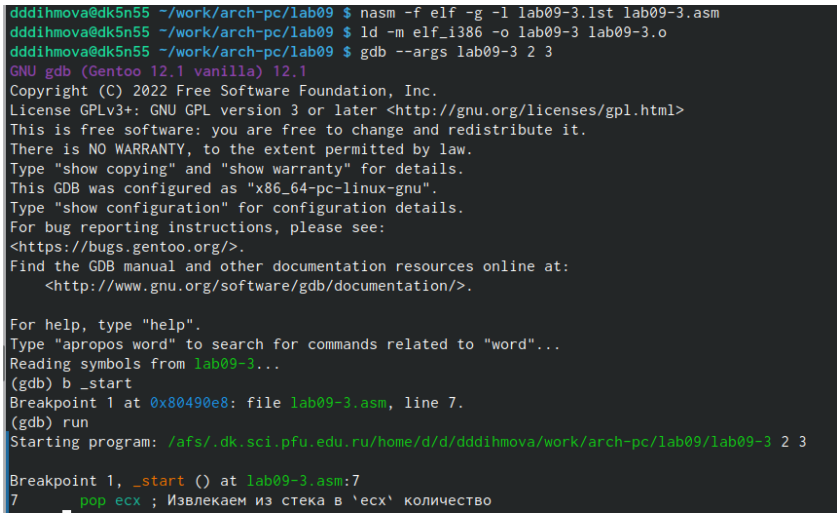

Рис. 3.21: Запуск программы с аргументами 2 3, установка брейкпоинта

Рассматриваю позиции стека, так как у меня всего два аргумента на шаге (+16) выдаёт ошибку (рис. 3.22).

Рис. 3.22: Позиции стека

4 Задание для самостоятельной работы

Создаю файл zadanie1.asm для выполнения первого задания из самостоятельной работы. Ввожу текст программы листинга 9.1 для удобства. Пишу программу согласно 13 варианту лабораторной работы номер 8 (рис. 4.1).

```
%include 'in_out.asm'

SECTION .data
    msg: DB 'Введите x: ',0
    result: DB '12x-7=',0

SECTION .bss
    x: RESB 80
    res: RESB 80

SECTION .text
GLOBAL _start
    _start:

mov eax, msg; вывод сообщение введите x
call sprint

mov ecx, x; занесение x в регистр
mov edx, 80
call sread

mov eax,x;преобразование из символа в число
call atoi

call _zadanie

mov eax,result; вывод сообщения 12x-7=
call sprint
mov eax,[res]; вывод результат
call iprintLF

call quit

_zadanie:
mov ebx,12
mul ebx
add eax, -7
mov [res],eax
ret
```

Рис. 4.1: Мною преобразованная программа

Создаю исполняемый файл и запускаю его (рис. 4.2).

```
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ nasm -f elf zadanie1.asm
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o zadanie1 zadanie1.o
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ ./zadanie1 1
Введите x: 1
12x-7=5
dddihmova@dk5n55 ~/work/arch-pc/lab09 $ ./zadanie1
Введите x: 2
12x-7=17
```

Рис. 4.2: Запуск файла

Создаю файл zadanie2.asm для выполнения второго пункта лабораторной работы, ввожу туда текст листинга 9.3, сохраняю, создаю исполняемый файл и запускаю его. Убеждаюсь, что результат неверный. Ответ должен быть 25 (рис. 4.4).

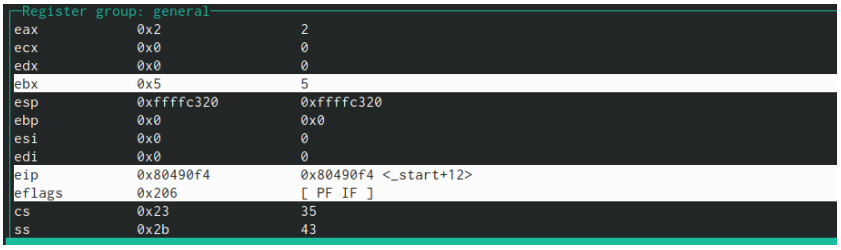

Рис. 4.3: Запуск файла

Запускаю откладчик, смотрю как изменяется регистры eax, ebx, ecx пошагово с помощью команды si. Замечаю, что на третьем шаге регистр ebx имеет значение 5, а на четвертом шаге ecx и eax перемножаются, что и дает неверный результат. Следовательно изменяю программу так, чтобы результат сложения записывался в eax (рис. 4.4).

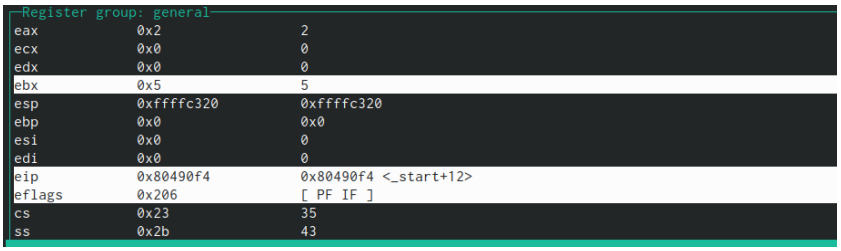

Рис. 4.4: Регистр ebx=5, а должно быть eax=5

У меня снова вывелся неправильный результат, пошагово проверяю как изменяются регистры, замечаю что в следующем сложении 5 прибавляется к регистру ebx, а должно к регистру eax, и в конце результат edi записывается в ebx, а должен в eax, изменяю и эту строчку (рис. 4.5).

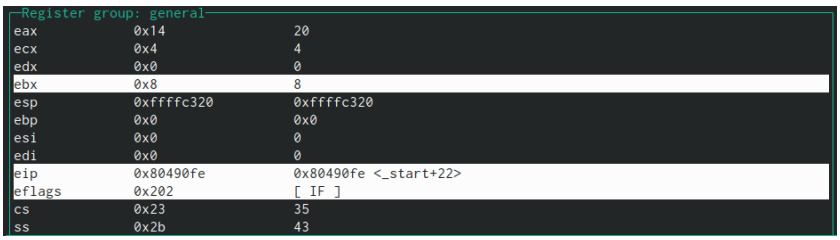

Рис. 4.5: Ошибка в сложении

Создаю исполняемый файл и запускаю его (рис. 4.6).

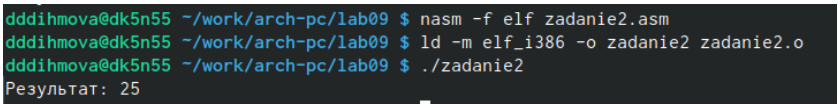

Рис. 4.6: Проверка программы

Текст моей программы (рис. 4.7).

```
%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. 4.7: Программа

5 Выводы

Я приобрела навыки написания программ с использованием подпрограмм. Познакомилась с методами отладки при помощи GDB и его основными возможностями.

Список литературы