Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Абакумов Тимофей Александрович

Содержание

# 1 Цель работы

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

# 2 Задание

**Порядок выполнения лабораторной работы**

1. Реализация подпрограмм в NASM.
2. Отладка программам с помощью GDB.
3. Выполнение заданий для самостоятельной работы.

# 3 Выполнение лабораторной работы

## 3.1 Реализация подпрограмм в NASM

1. Создадим каталог для выполнения лабораторной работы № 9, перейдите в него и создадим файл lab09-1.asm (рис. 1).

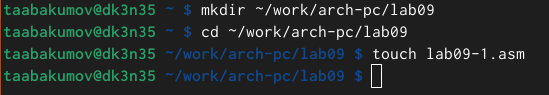


Рис. 1: Создание каталога и файла

1. В качестве примера рассмотрим программу вычисления арифметического выражения f(x) = 2x + 7 с помощью подпрограммы \_calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Введём текст программы из Листинга 9.1 (рис. 2).

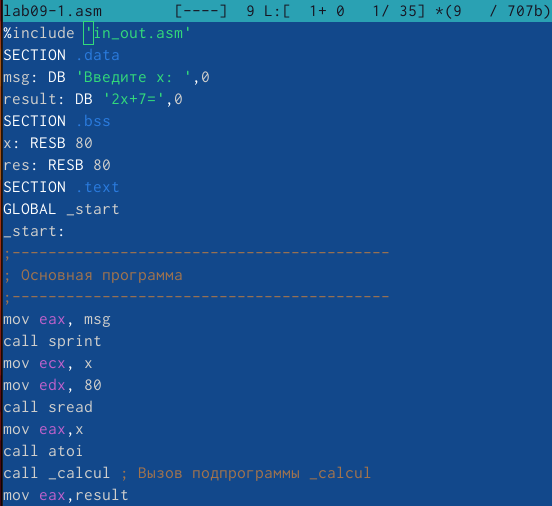


Рис. 2: Код программы

Код программы из пункта 2:

%include ‘in\_out.asm’

SECTION .data

msg: DB ‘Введите x:’,0

result: DB ‘2x+7=’,0

SECTION .bss

x: RESB 80

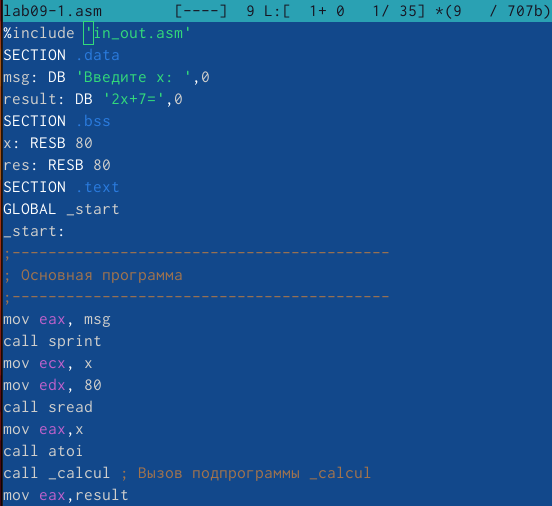
res: RESB 80

SECTION .text

GLOBAL \_start

\_start:

mov eax, msg(рис. 2).

заданий для самостоятельной работы.

call sprint

mov ecx, x

mov edx, 80

call sread

mov eax,x

call atoi

call \_calcul ; Вызов подпрограммы \_calcul

mov eax,result

call sprint

mov eax,[res]

call iprintLF

call quit

\_calcul:

mov ebx,2

mul ebx

add eax,7

mov [res],eax

ret ; выход из подпрограммы

1. Создадим исполняемый файл и проверим его работу (рис. 3).

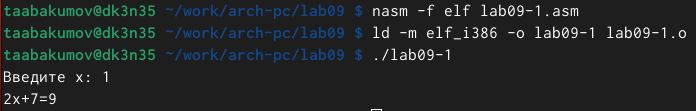


Рис. 3: Работа программы

1. Изменим 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. Т.е. x передается в подпрограмму \_calcul из нее в подпрограмму \_subcalcul, где вычисляется выражение g(x), результат возвращается в \_calcul и вычисляется выражение f(g(x)). Результат возвращается в основную программу для вывода результата на экран. (рис. 4).

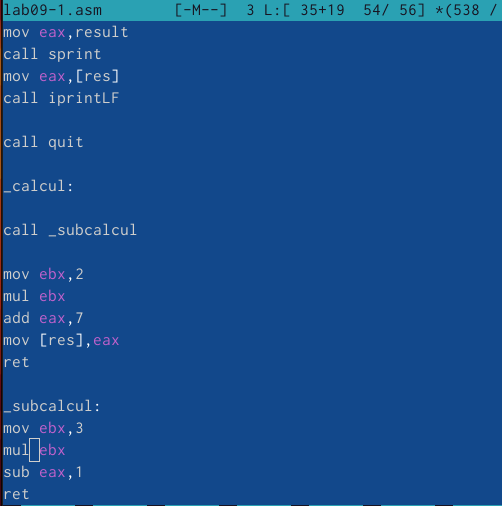


Рис. 4: Код программы

Код программы из пункта 4:

%include ‘in\_out.asm’

SECTION .data

msg: DB ‘Введите x:’,0

prim1: DB ‘f(x) = 2x+7’,0

prim2: DB ‘g(x) = 3x-1’,0

result: DB ‘f(g(x))=’,0

SECTION .bss

x: RESB 80заданий для самостоятельной работы.

res: RESB 80

SECTION .text

GLOBAL \_start

\_start:

mov eax,prim1

call sprintLF

mov eax,prim2

call sprintLF

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,[res]

call iprintLF

call quit

\_calcul:

call \_subcalcul

mov ebx,2

mul ebx

add eax,7 заданий для самостоятельной работы. mov [res],eax

ret

\_subcalcul:

mov ebx,3

mul ebx

sub eax,1

ret

1. Создадим исполняемый файл и проверим его работу (рис. 5).

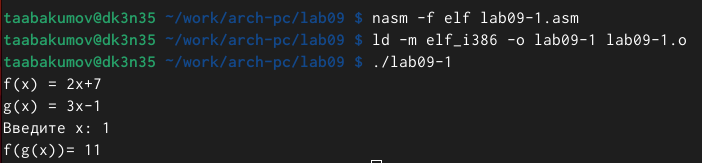


Рис. 5: Работа программы

## 3.2 Отладка программам с помощью GDB

1. Создадим файл lab09-2.asm (рис. 6).

Рис. 6: Создание файла

Рис. 6: Создание файла

1. Введём в файл lab09-2.asm текст из Лисзаданий для самостоятельной работы.тинга 9.2 (Программа печати сообщения Hello world!) (рис. 7).

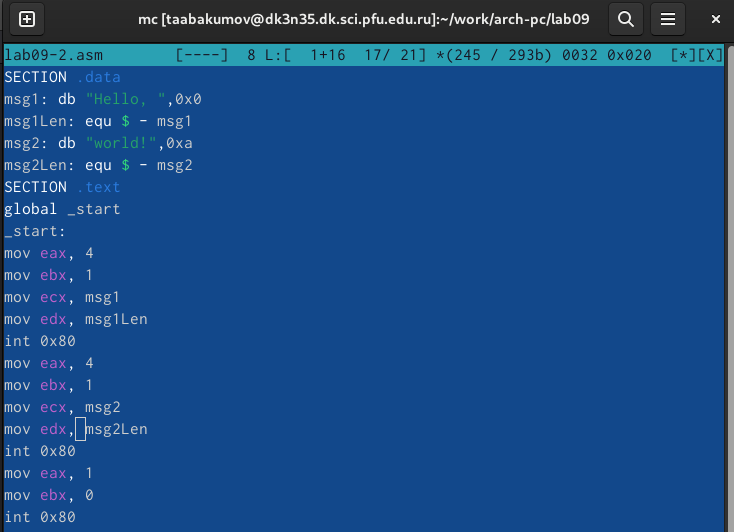


Рис. 7: Код программы

Код программы из пункта 7:

SECTION .data

msg1: db “Hello,”,0x0

msg1Len: equ $ - msg1

msg2: db “world!”,0xa

msg2Len: equ $ - msg2

SECTION .text

global \_start

\_start:

mov eax, 4

mov ebx, 1

mov ecx, msg1

mov edx, msg1Len

int 0x80

mov eax, 4

mov ebx, 1

mov ecx, msg2

mov edx, msg2Len

int 0x80

mov eax, 1

mov ebx, 0

int 0x80

1. Получим исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. После этого загрузим исполняемый файл в отладчик gdb и проверим работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 8).

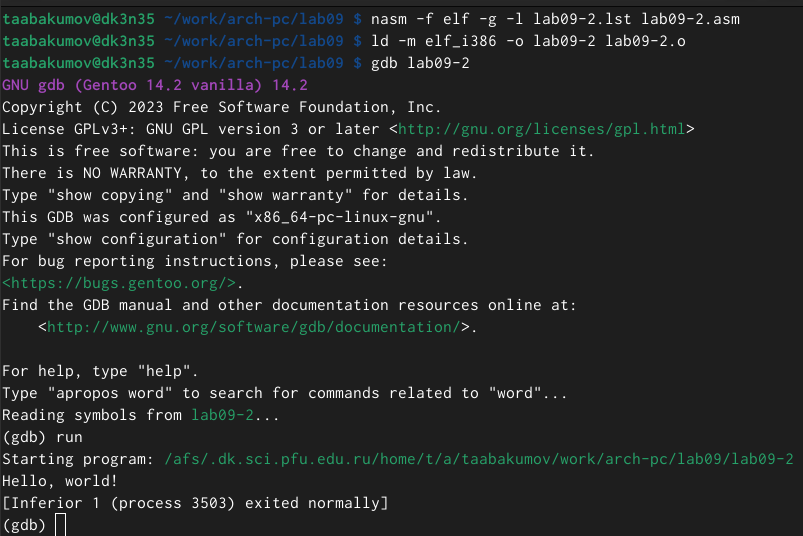


Рис. 8: Работа программы

1. Для более подробного анализа программы установим брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустим её (рис. 9).

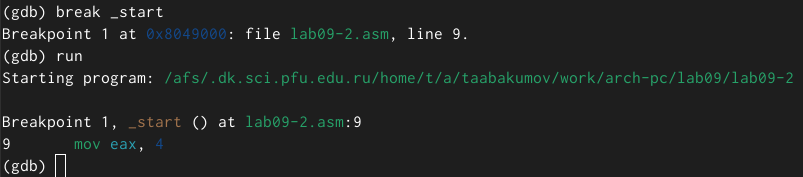


Рис. 9: Подробный анализ программы

1. Посмотрите дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 10).



Рис. 10: Дисассимилированный код

1. Переключимся на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel.Отличие заключается в командах, в диссамилированном отображении в командах используют % и $, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть (рис. 11).

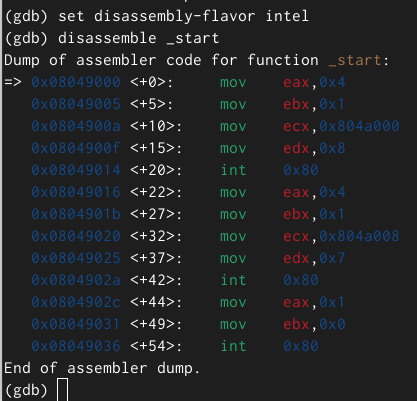


Рис. 11: Intel’овский синтаксис

1. Для удобства включим режим псевдографики (рис. 12).

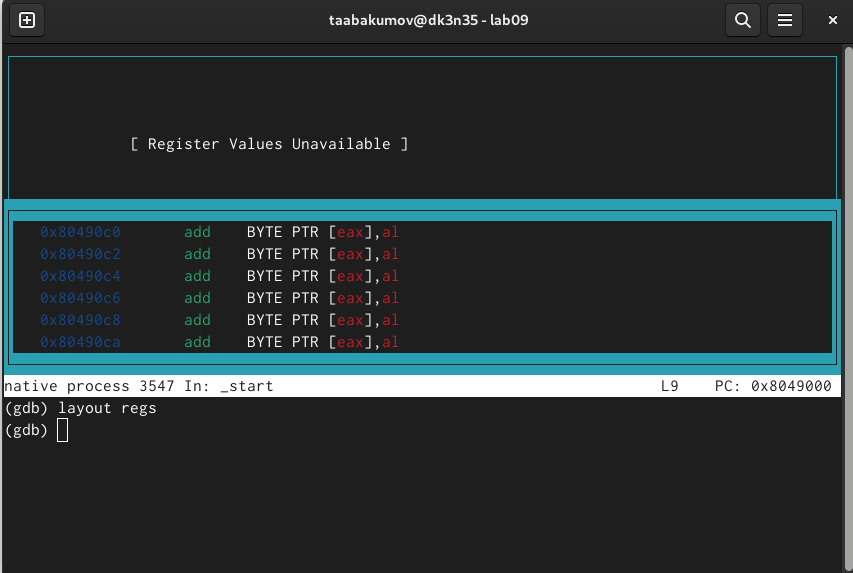


Рис. 12: Режим псевдографики

1. На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверим это с помощью команды info breakpoints (рис. 13).

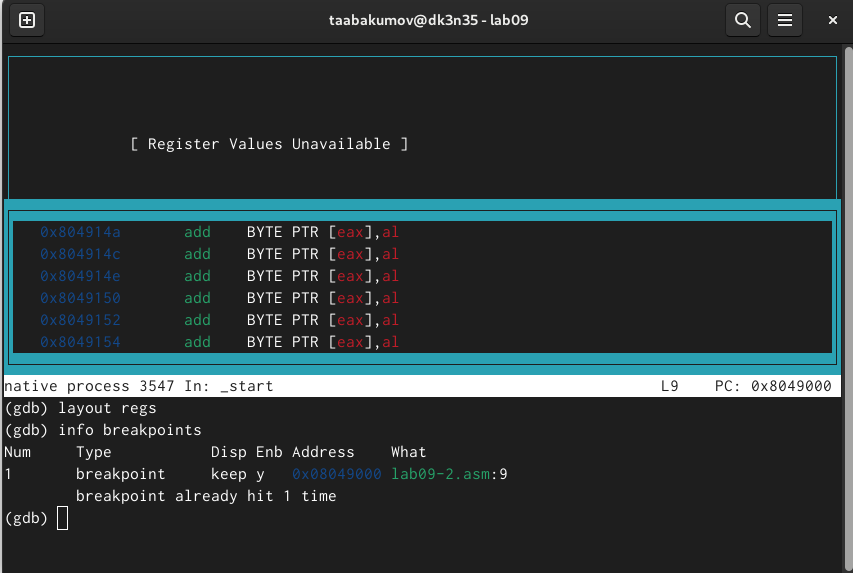


Рис. 13: Проверка точки

1. Установим еще одну точку останова по адресу инструкции. Определим адрес предпоследней инструкции (mov ebx,0x0) и установим точку останова (рис. 14).

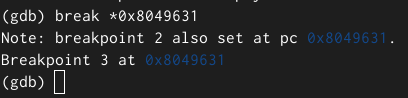


Рис. 14: Установка точки

1. Посмотрим информацию о всех установленных точках останова (рис. 15).

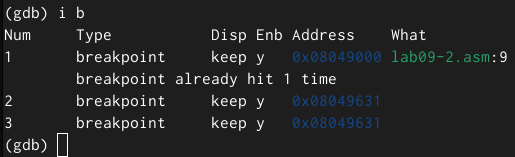


Рис. 15: Проверка информации

1. С помощью команды si посмотрим регистры и изменим их (рис. 16).

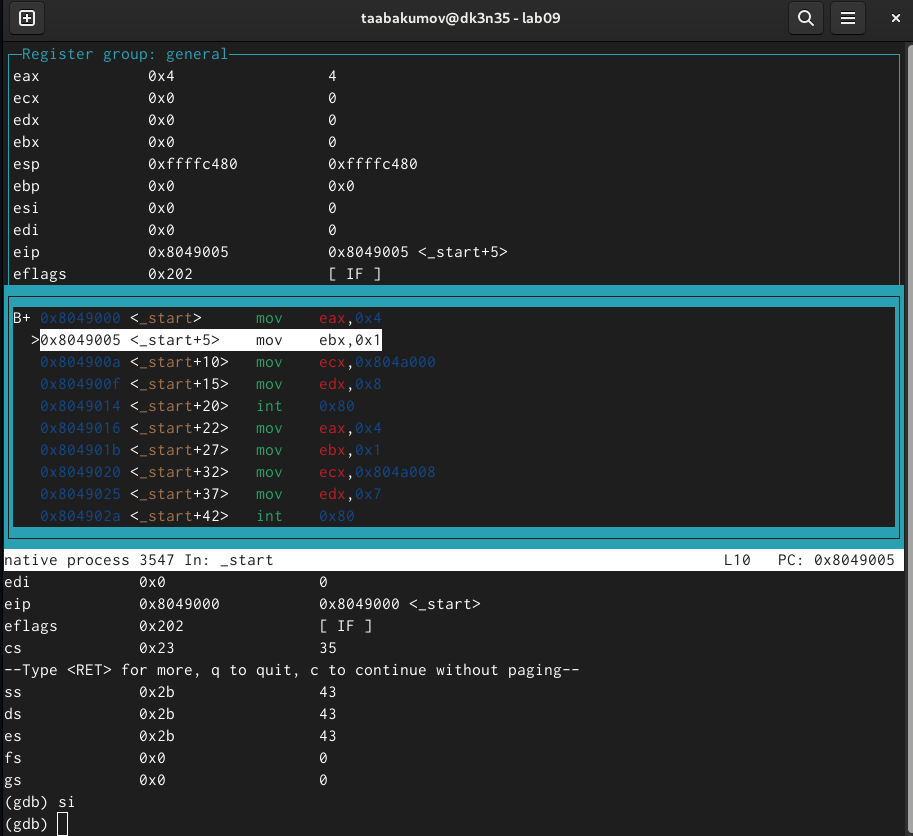


Рис. 16: Изменение регистров

1. С помощью команды посмотрим значение переменной msg1 (рис. 17).

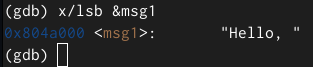


Рис. 17: Значение переменной msg1

1. Следом посмотрим значение второй переменной msg2 (рис. 18).

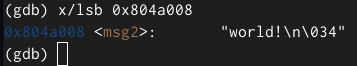


Рис. 18: Значение переменной msg2

1. С помощью команды set изменим значение переменной msg1 (рис. 19).

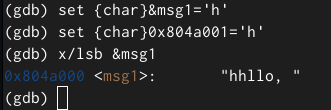


Рис. 19: Изменение значение переменной msg1

1. Также изменим переменную msg2 (рис. 20).

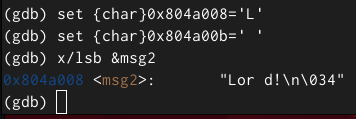


Рис. 20: Изменение значение переменной msg2

1. Выведем значения регистров ecx и eax (рис. 21).

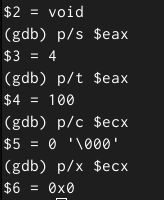


Рис. 21: Значения регистров

1. Изменим значение регистра ebx. Команда выводит два разных значения так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум, поэтому и значения разные (рис. 22).

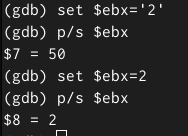


Рис. 22: Изменение значения регистра

1. После всего завершим работу с файлов (рис. 23).

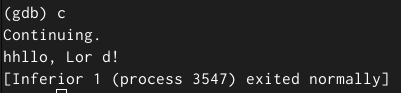


Рис. 23: Завершение работы с файлом

1. Далее скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm. После чего создадим исполняемый файл (рис. 24).

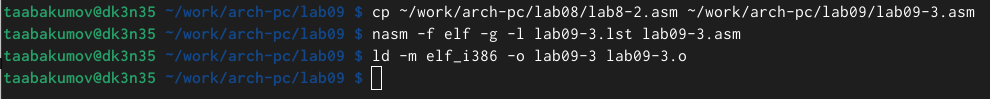


Рис. 24: Копирование и создание исполняемого файла

1. Загрузим исполняемый файл в отладчик, указав аргументы. Установим точку останова перед первой инструкцией в программе (рис. 25).

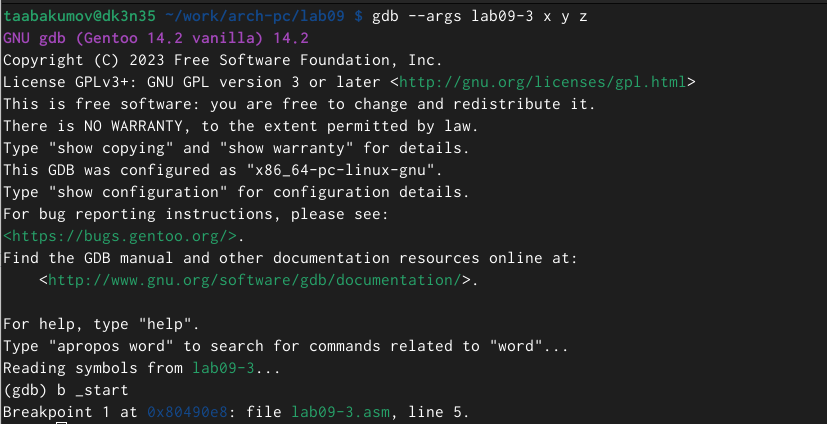


Рис. 25: Загрузка файла в отладчик

1. Далее запустим программу (рис. 26).

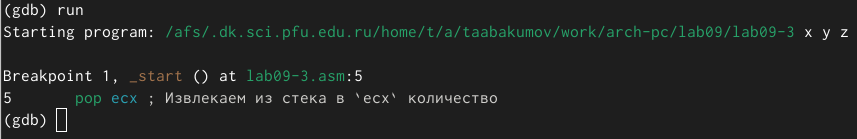


Рис. 26: Запуск программы

1. Проверим адрес вершины стека и убедимся, что там хранится 4 элемента (рис. 27).

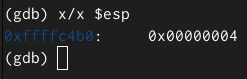


Рис. 27: Проверка адреса

1. Посмотрим все позиции стека. По первому адресу хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные сохранялись нормально и без помех, компьютер использует новый стек для новой информации (рис. 28).

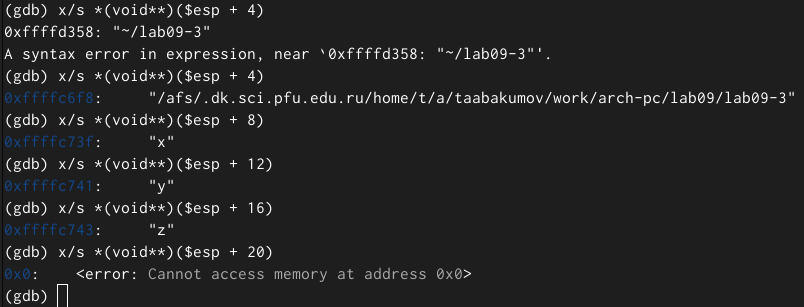


Рис. 28: Просмотр всех позиций стека

## 3.3 Задания для самостоятельной работы

Задание 1. Преобразуйте программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму.

1. Для начала создадим файл lab09-4.asm (рис. 29).

Рис. 29: Создание файла

Рис. 29: Создание файла

1. Преобразуем программу из лабораторной работы №8 и реализуем вычисления как подпрограмму (рис. 30).

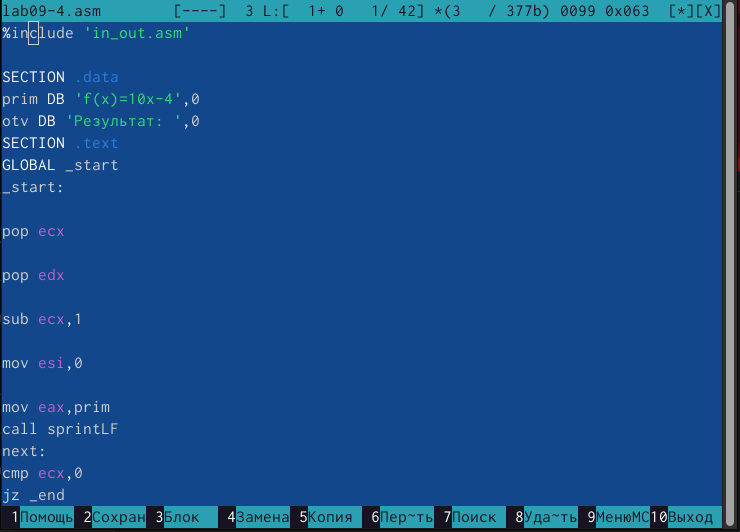


Рис. 30: Код программы

1. Создадим исполняемый файл и проверим его работу (рис. 31).

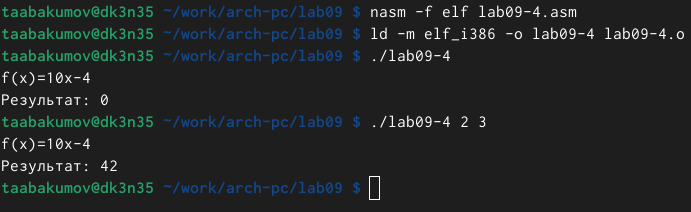


Рис. 31: Работа программы

Задание 2. В листинге 9.3 приведена программа вычисления выражения (3 + 2) \* 4 + 5. При запуске данная программа дает неверный результат. Проверьте это. С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее.

1. Для начала создадим файл lab09-5.asm (рис. 32).

Рис. 32: Создание файла

Рис. 32: Создание файла

1. Перепишем программу из Листинга 9.3 в созданный файл (рис. 33).

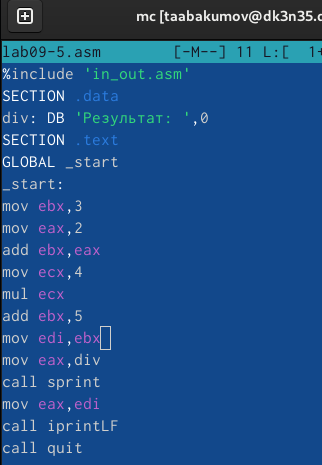


Рис. 33: Код программы

Код программы из пункта 33:

%include ‘in\_out.asm’

SECTION .data

div: DB ‘Результат:’,0

SECTION .text

GLOBAL \_start

\_start:

mov ebx,3

mov eax,2

add ebx,eax

mov ecx,4

mul ecx

add ebx,5

mov edi,ebx

mov eax,div

call sprint

mov eax,edi

call iprintLF

call quit

1. Создадим исполняемый файл и проверим его работу. Ошибка оказалась арифметическая, так как вместо 25, программа выводит 10 (рис. 34).

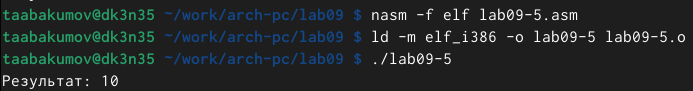


Рис. 34: Работа программы

1. Из-за появления ошибки, запустим программу в отладчике (рис. 35).

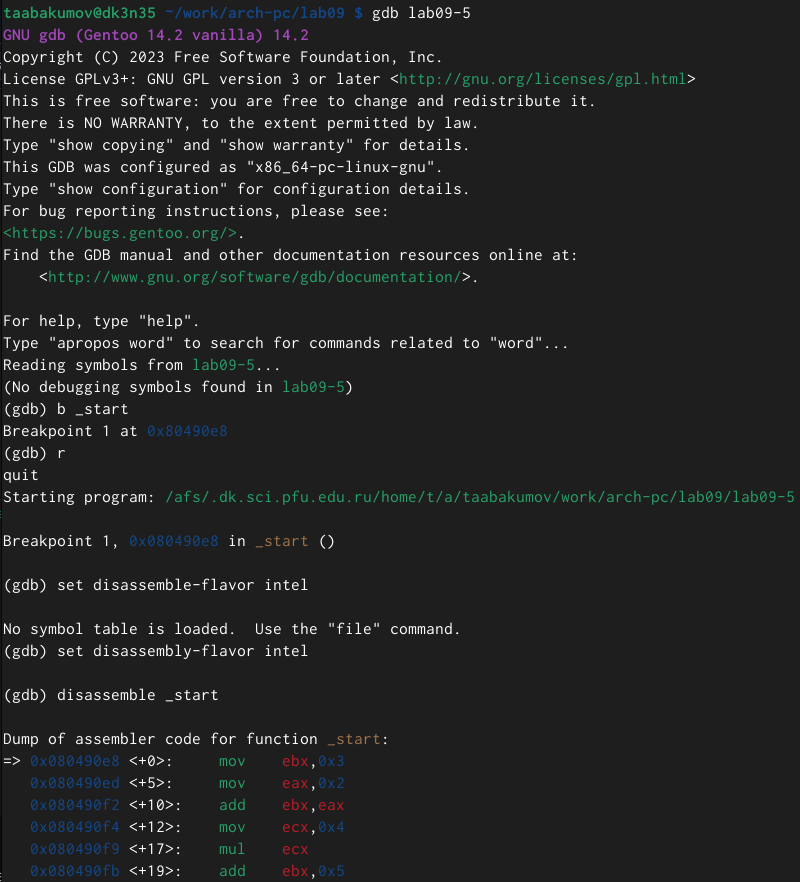


Рис. 35: Запуск программы в отладчике

1. Изменим регистры и запустим программу, программа вывела ответ 25, то есть все работает правильно (рис. 36).

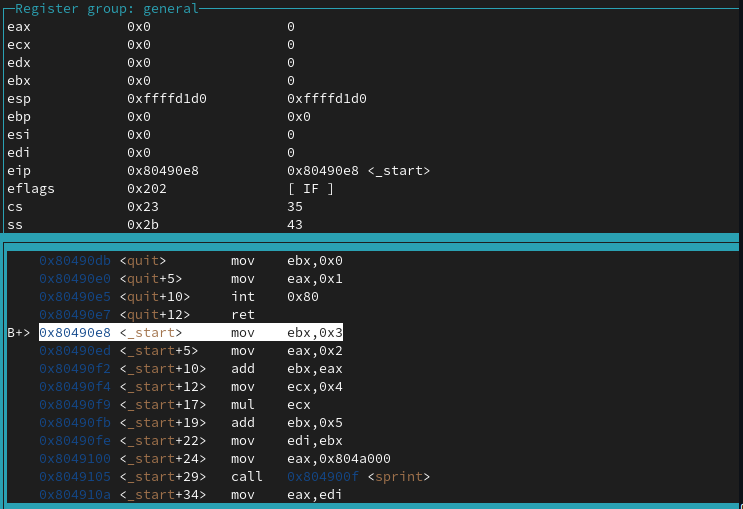


Рис. 36: Изменение регистров

(рис. 37).

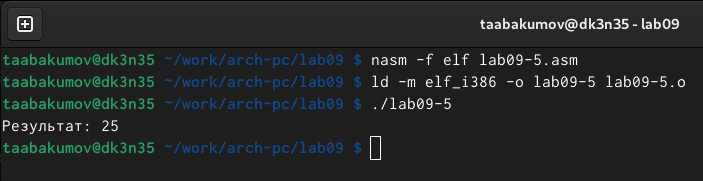


Рис. 37: Работа программы

# 4 Выводы

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