Отчёт по лабораторной работе №10

Нефёдова Наталия Николаевна

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

## 2.1 Реализация подпрограмм в NASM 1.

Создадим каталог для выполнения лабораторной работы № 10, перейдем в него и создадим файл lab10-1.asm (рис. 1)

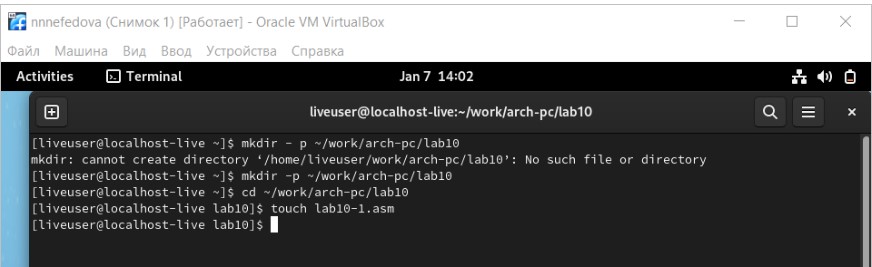


Рис. 1: 1

## 2.2 Пример программы с использованием вызова подпрограммы

Рассмотрим программу с использованием вызова подпрограммы(рис. 2), (рис. 3)

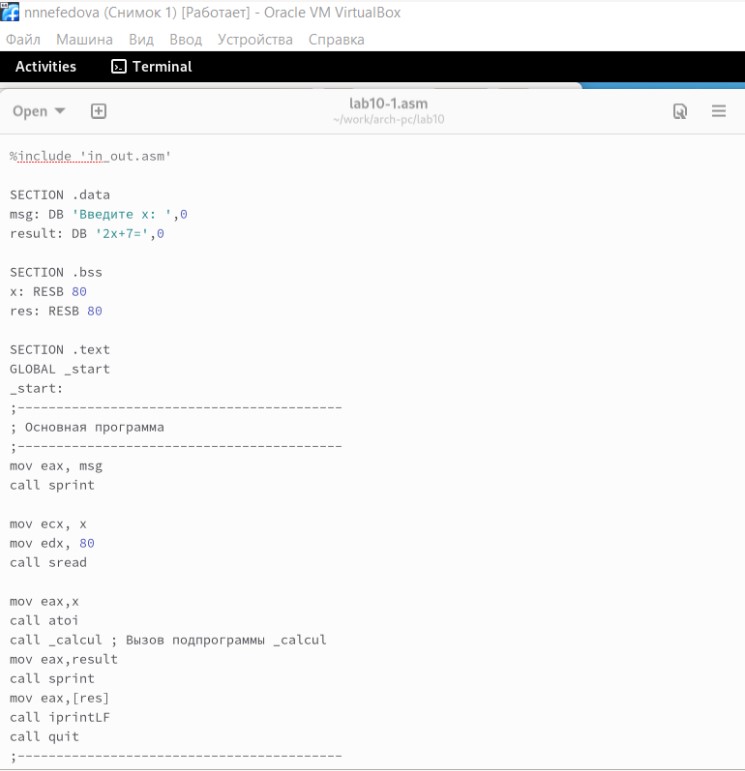


Рис. 2: 2

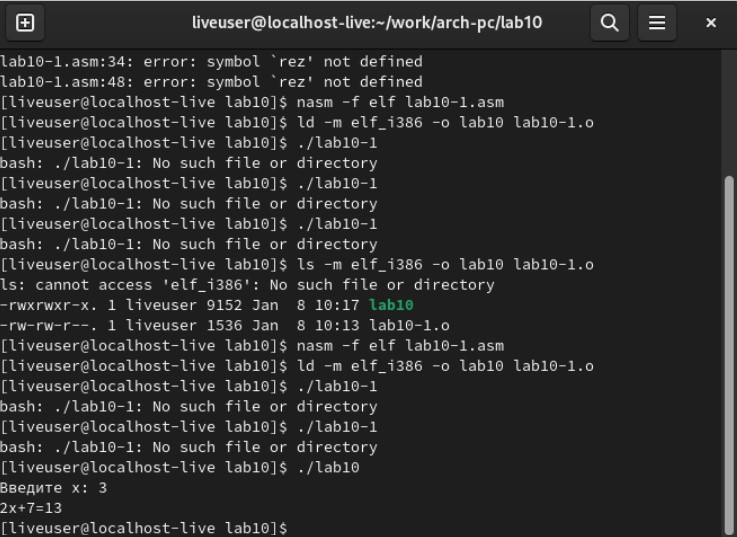


Рис. 3: 3

Первые строки программы отвечают за вывод сообщения на экран (call sprint), чтение данных введенных с клавиатуры (call sread) и преобразования введенных данных из символьного вида в численный (call atoi). После следующей инструкции call \_calcul, которая передает управление подпрограмме \_calcul, будут выполнены инструкции подпрограммы: mov ebx,2 mul ebx add eax,7 mov [rez],eax ret

Инструкция ret является последней в подпрограмме и ее исполнение приводит к возвращению в основную программу к инструкции, следующей за инструкцией call, которая вызвала данную подпрограмму. Последние строки программы реализую вывод сообщения (call sprint), результата вычисления (call iprintLF) и завершение программы (call quit). Проверим работу программы и выведем результат на экран.

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 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: 4

## 2.3 Отладка программам с помощью GDB

Создадим файл lab10-2.asm с текстом программы из Листинга 10.2. (Программа печати сообщения Hello world!): Листинг 10.2. Программа вывода сообщения Hello world! (рис. 5), (рис. 6)

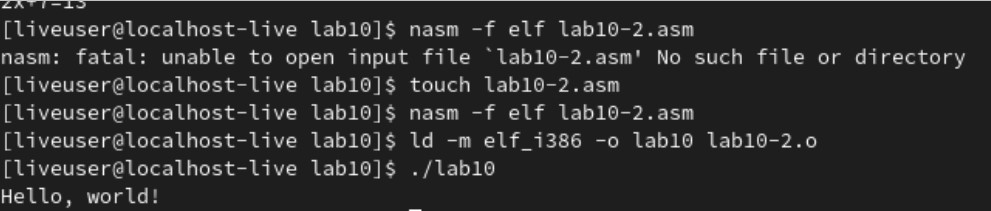


Рис. 5: 5

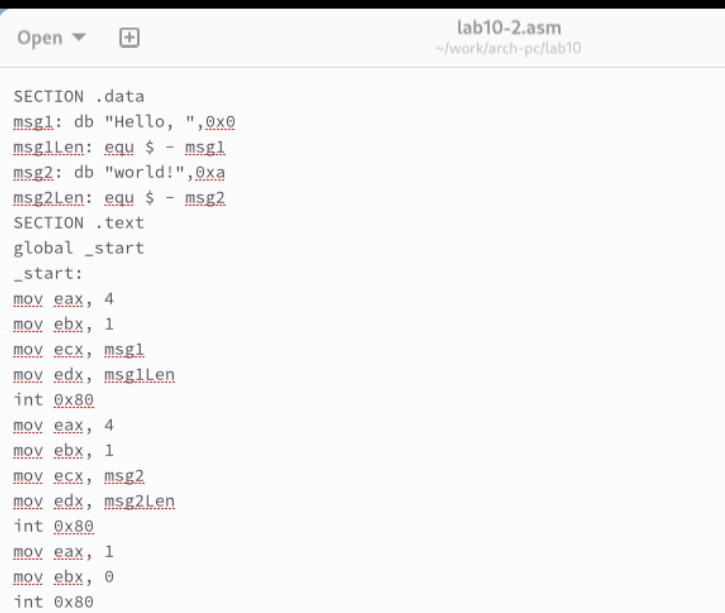


Рис. 6: 6

Получим исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом ‘-g’. nasm -f elf -g -l lab10-2.lst lab10-2.asm ld -m elf\_i386 -o lab10-2 lab10-2.o Загрузим исполняемый файл в отладчик gdb: (рис. 7)

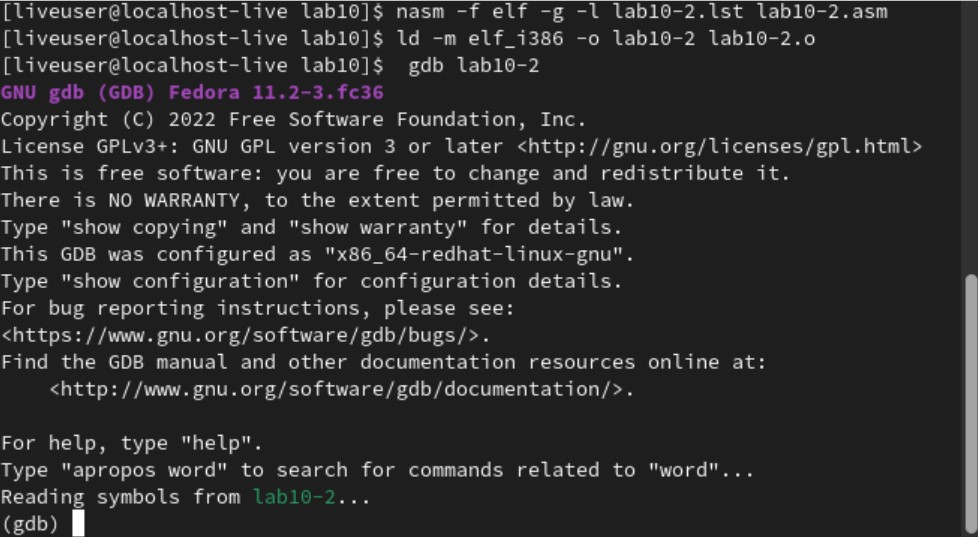


Рис. 7: 7

Проверим работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r) (рис. 8), (рис. 9)

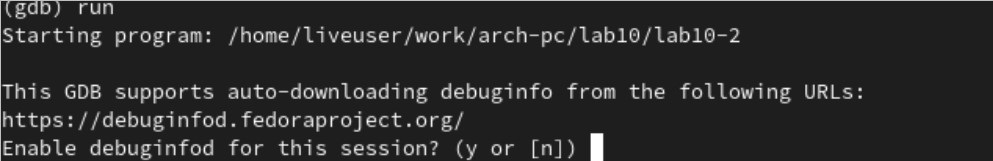


Рис. 8: 8

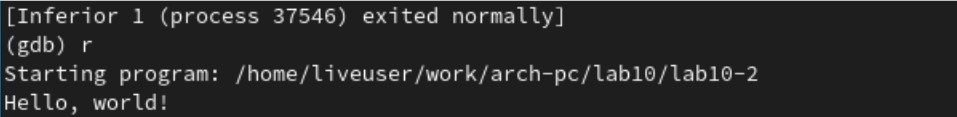


Рис. 9: 9

Для более подробного анализа программы установим брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустим её. (рис. 10), (рис. 11)

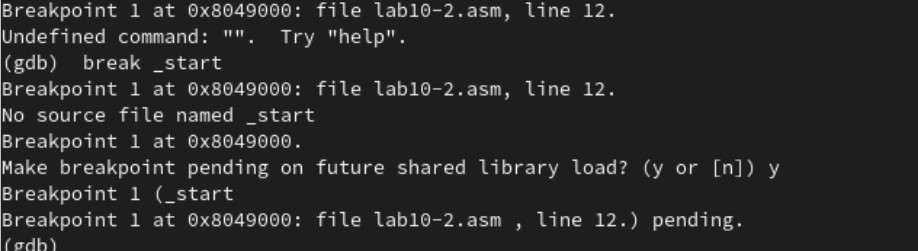


Рис. 10: 10

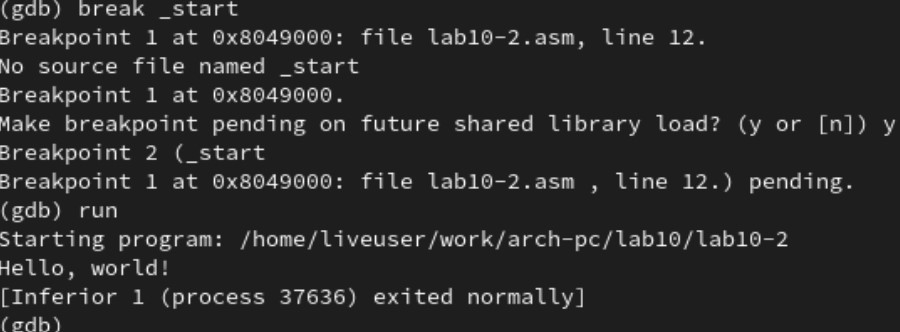


Рис. 11: 11

Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 12)

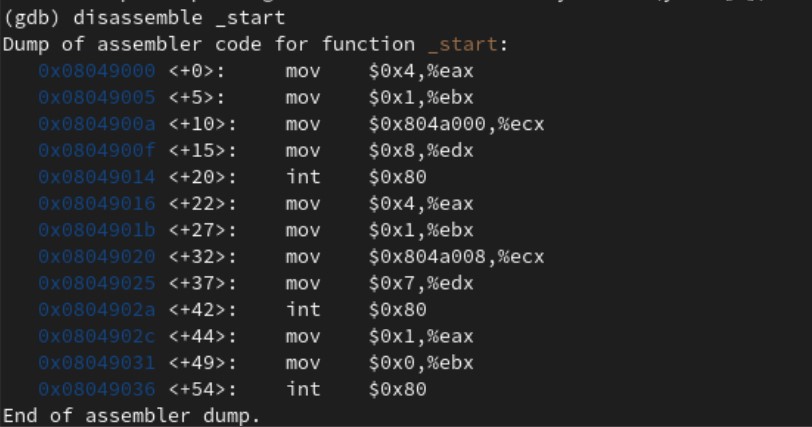


Рис. 12: 12

Переключимся на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (gdb) set disassembly-flavor intel (gdb) disassemble \_start (рис. 13)

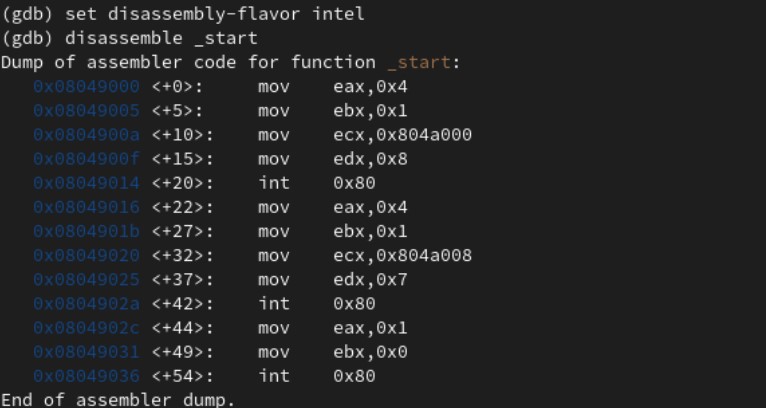


Рис. 13: 13

В этом режиме есть три окна: - В верхней части видны названия регистров и их текущие значения; - В средней части виден результат дисассимилирования программы; - Нижняя часть доступна для ввода команд. (рис. 14)

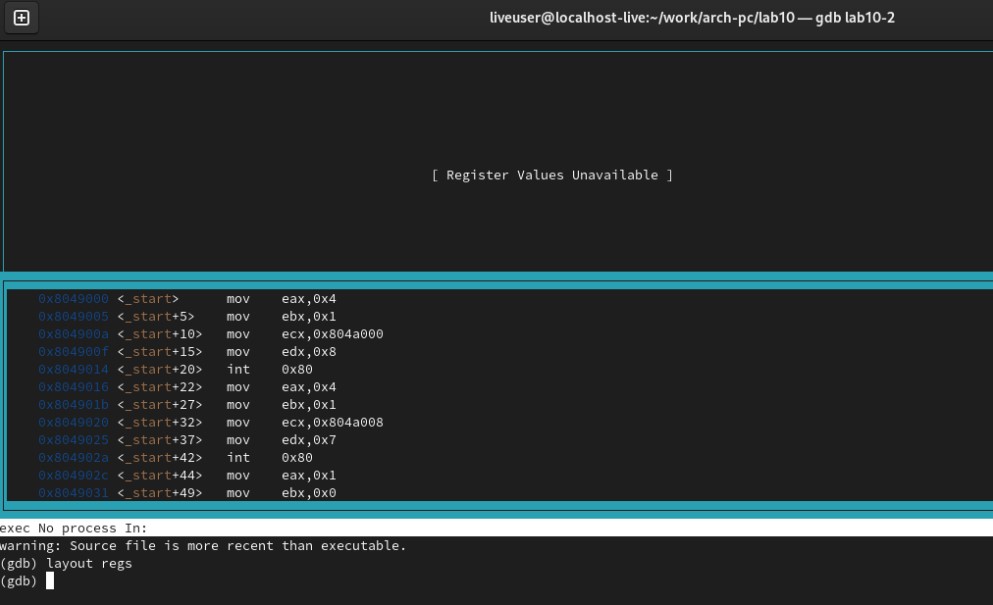


Рис. 14: 14

## 2.4 Добавление точек останова

Установить точку останова можно командой break (кратко b). Типичный аргумент этой команды — место установки. Его можно задать или как номер строки программы (имеет смысл, если есть исходный файл, а программа компилировалась с информацией об отладке), или как имя метки, или как адрес. Чтобы не было путаницы с номерами, перед адресом ставится «звёздочка»: На предыдущих шагах была установлена точка останова по имени метки (\_start). Проверим это с помощью команды info breakpoints (кратко i b): (gdb) info breakpoints. (рис. 15)

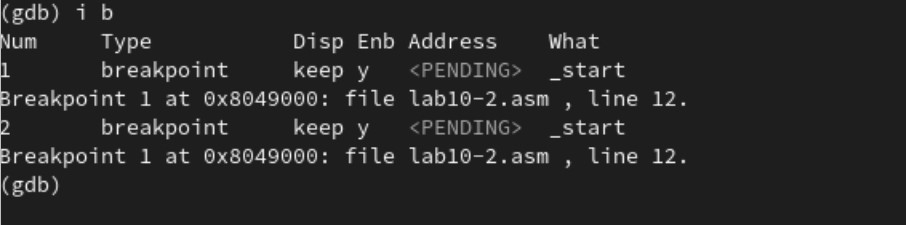


Рис. 15: 15

## 2.5 Работа с данными программы в GDB

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполним 5 инструкций с помощью команды stepi (или si) и проследим за изменением значений регистров. Посмотреть содержимое регистров также можно с помощью команды info registers (или i r). (gdb) info registers (рис. 16)

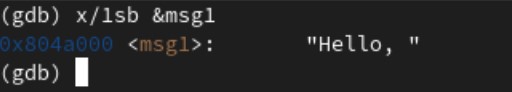


Рис. 16: 16

Посмотрите значение переменной msg2 по адресу. (рис. 17)

Рис. 17: 17

Рис. 17: 17

Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. При этом перед именем регистра ставится префикс $, а перед адресом нужно указать в фигурных скобках тип данных (размер сохраняемого значения; в качестве типа данных можно использовать типы языка Си). Изменим первый символ переменной msg1. (рис. 18)

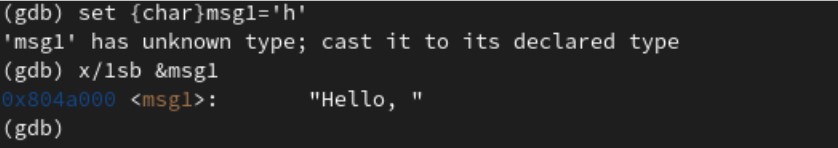


Рис. 18: 18

Заменим второй символ в переменной msg2 (рис. 19)

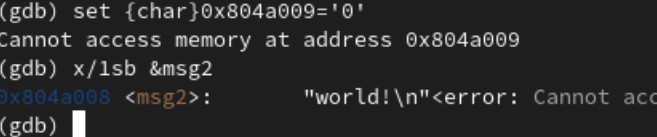


Рис. 19: 19

Чтобы посмотреть значения регистров используется команда print /F (перед именем регистра обязательно ставится префикс $.

Завершим выполнение программы с помощью команды continue (сокращенно c) и выйдем из GDB с помощью команды quit (сокращенно q).

## 2.6 Обработка аргументов командной строки в GDB

Скопируем 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 и создадим исполняемый файл. (рис. 20)

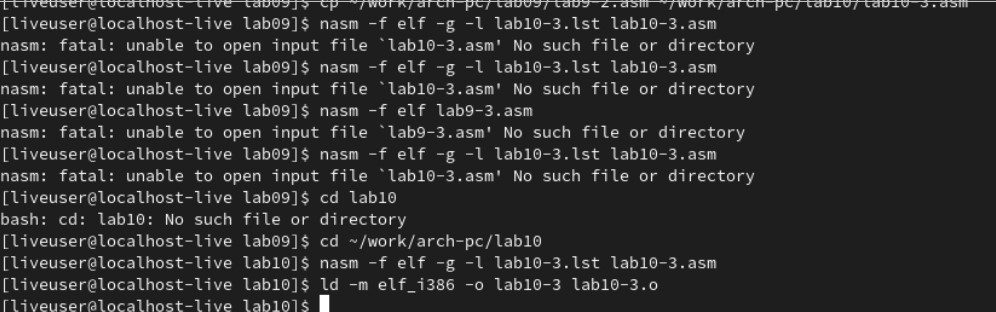


Рис. 20: 20

Как отмечалось в предыдущей лабораторной работе, при запуске программы аргументы командной строки загружаются в стек. Для начала установим точку останова перед первой инструкцией в программе и запустим ее. (рис. 21)

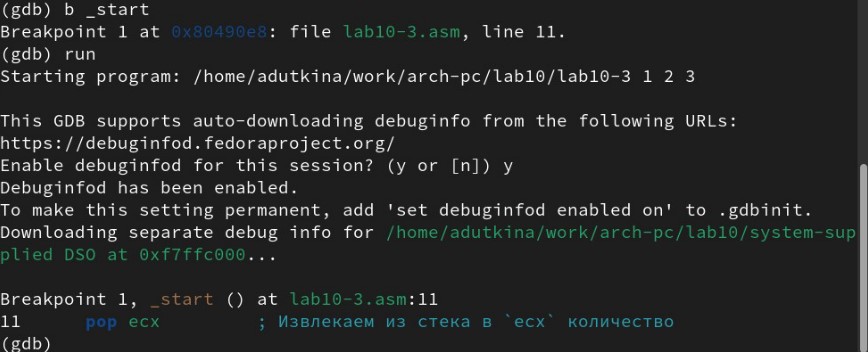


Рис. 21: 21

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 4 - расположение программы и три аргумента.

Посмотрим остальные позиции стека – по адесу [esp+4] располагается адрес в памяти где находиться имя программы, по адесу [esp+8] храниться адрес первого аргумента, по аресу [esp+12] – второго и т.д.

## 2.7 Задание для самостоятельной работы

1. Преобразуем программу из лабораторной работы №9 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции f(x) как подпрограмму.
2. В листинге 10.3 приведена программа вычисления выражения (3+2)\*4+5. При запуске данная программа дает неверный результат. Проверим это. С помощью отладчика GDB, анализируя изменения значений регистров, определим ошибку и исправим ее. (рис. 22), (рис. 23), (рис. 24)

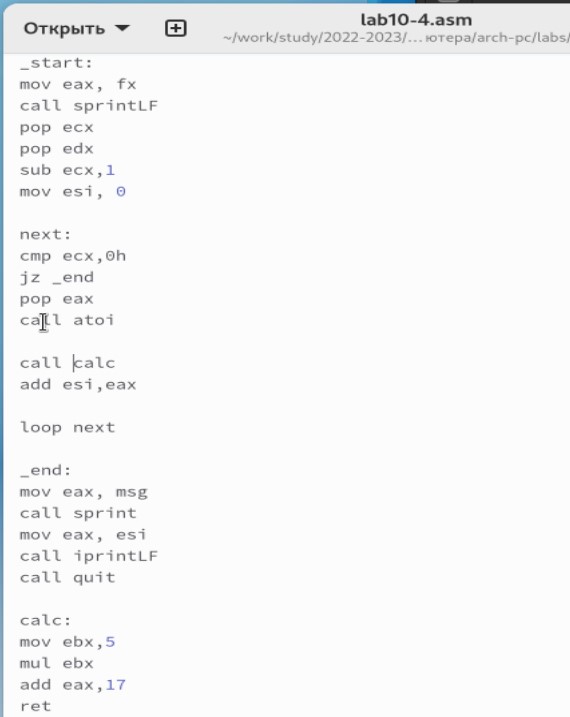


Рис. 22: 22

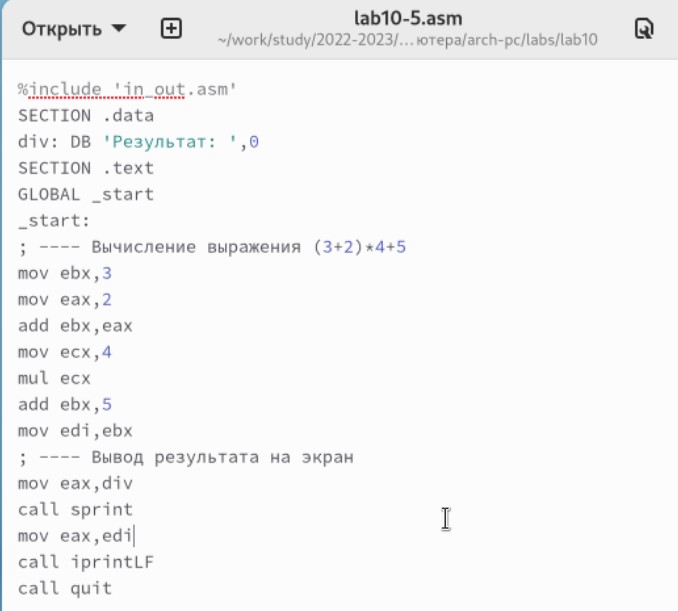


Рис. 23: 23

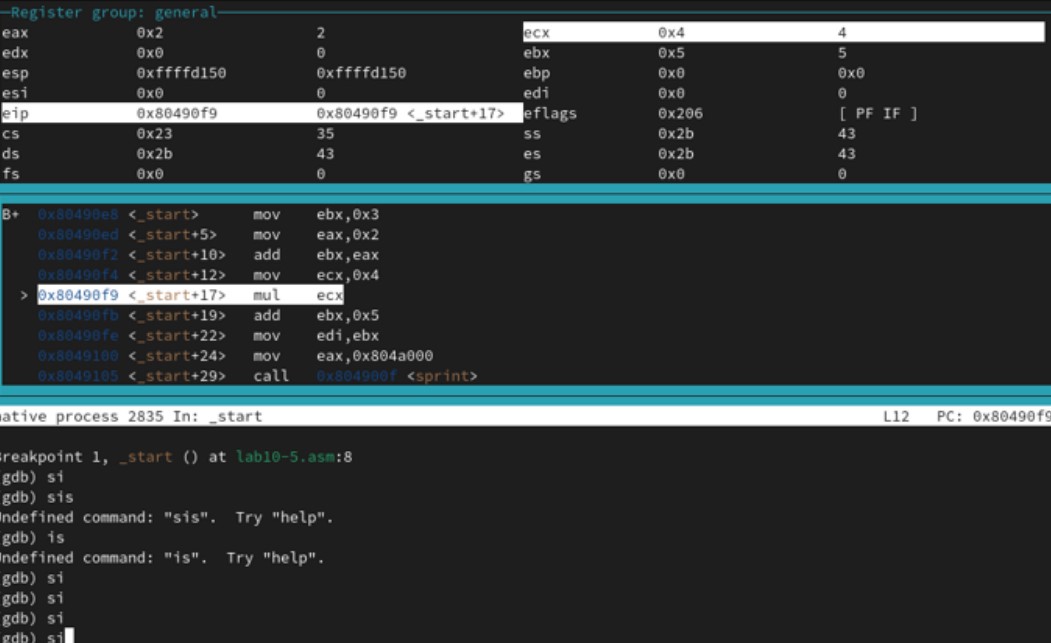


Рис. 24: 24

Отметим, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax (рис. 25)

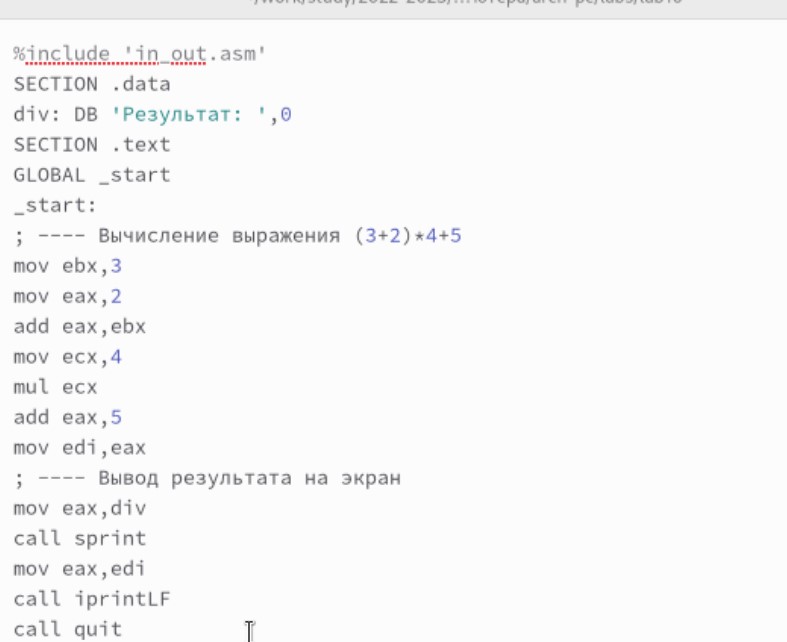


Рис. 25: 25

# 3 Выводы

В ходе лабораторной работы были приобретены навыки написания программ с использованием подпрограмм и изучены методы отладки при помощи GDB и его основные возможности