**Отчет по лабораторной работе №10**

дисциплина: Архитектура компьютера

Колобова Елизавета Андреевна гр. НММбд-01

Содержание

# 1 **Цель работы**

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

# 2 **Выполнение лабораторной работы**

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

mkdir ~/work/arch-pc/lab10  
cd ~/work/arch-pc/lab10  
touch lab10-1.asm

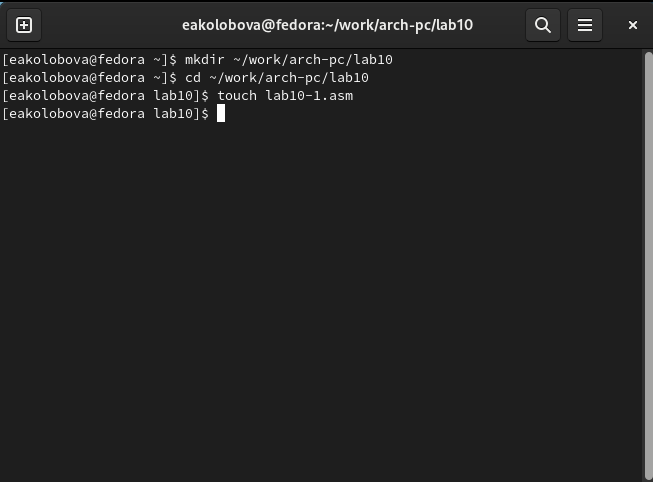


Рис. 1: Рис. 1. Создание каталога для лаб. работы №10 и файла lab10-1.asm

1. 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. Введем в файл lab10-1.asm текст программы из листинга 10.1. Создадим исполняемый файл и проверим его работу (рис. 2), рис. 3).

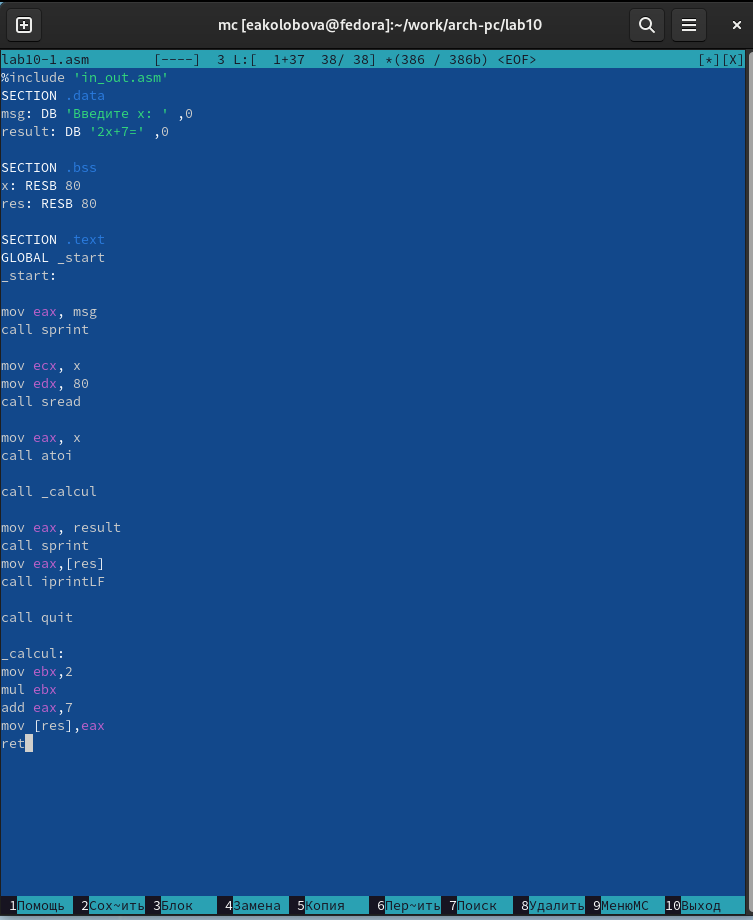


Рис. 2: Рис. 2. Ввод текста из листинга 10.1

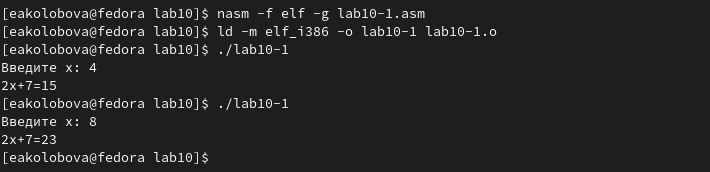


Рис. 3: Рис. 3. Компоновка и запуск файла lab10-1

1. Изменим текст программы, добавив подпрограмму \_subcalcul в подпро- грамму \_calcul, для вычисления выражения f(g(x)), где x вводится с клавиату- ры, 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 воз- вращается в \_calcul и вычисляется выражение f(g(x)). Результат возвращается в основную программу для вывода результата на экран. (рис. 4, 5):

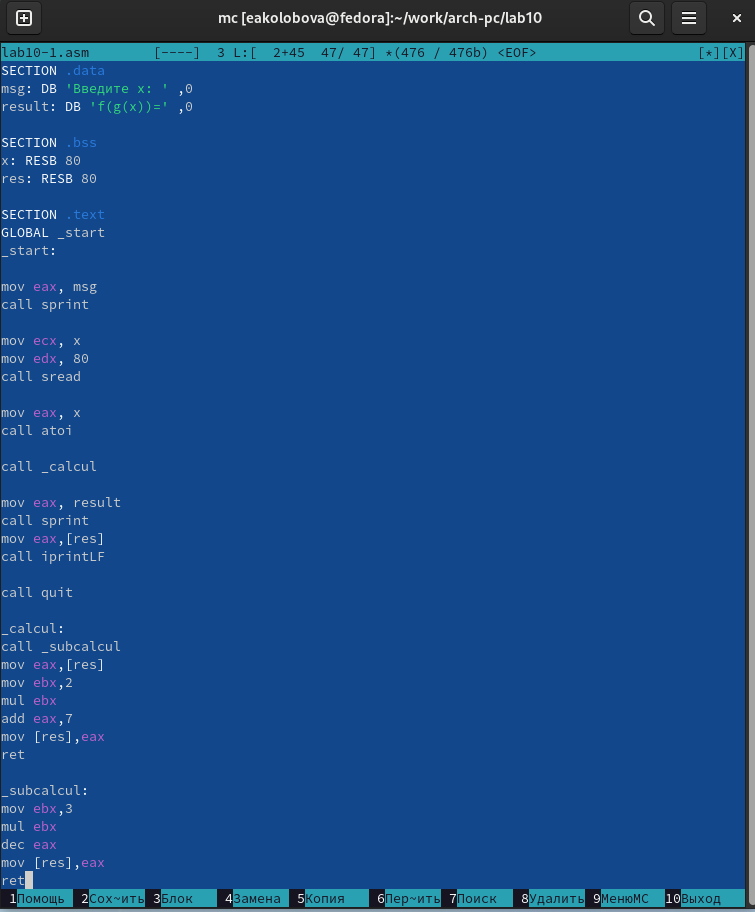


Рис. 4: Рис. 4. Изменение текста программы

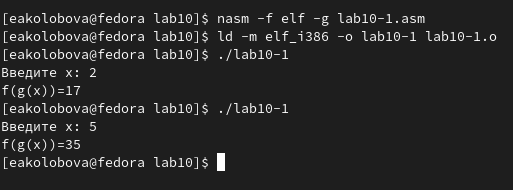


Рис. 5: Рис. 5. Компоновка и запуск измененного файла

1. Создадим файл lab10-2.asm с текстом программы из Листинга 10.2. (Программа печати сообщения Hello world!) (рис. 6)

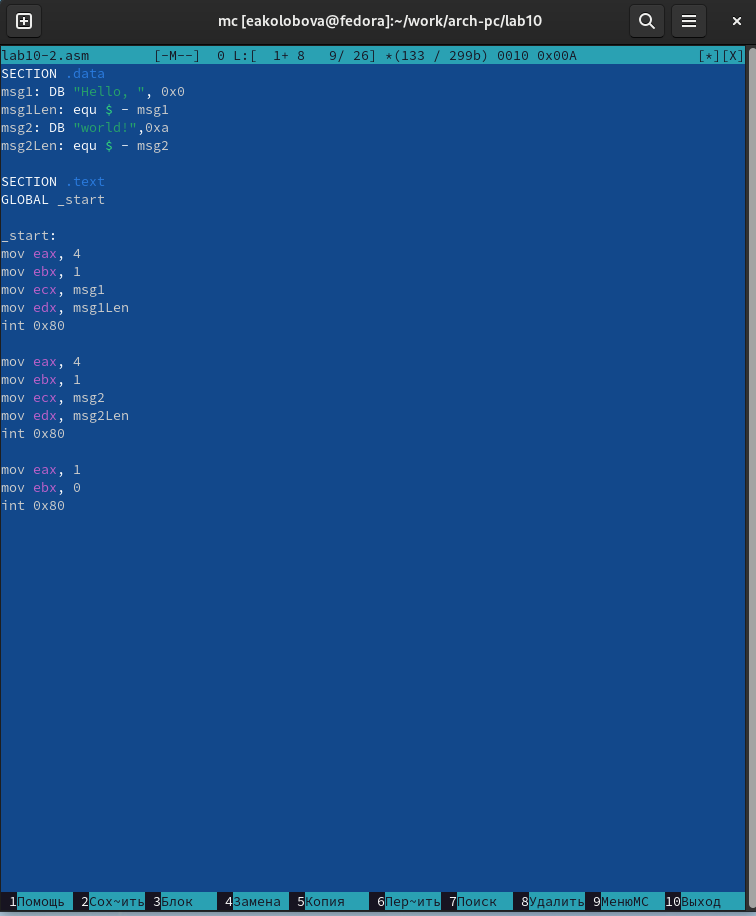


Рис. 6: Рис. 6. Ввод текста из листинга 10.2

Получим исполняемый файл. Для работы с GDB трансляцию программ необходимо проводить с ключом ‘-g’. (рис. 7)

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):

user@dk4n31:~$ gdb lab10-2

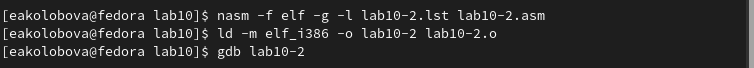


Рис. 7: Рис. 7. Компоновка файла и загрузка его в отладчик

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

(gdb) run  
Starting program: ~/work/arch-pc/lab10/lab10-2  
Hello, world!  
[Inferior 1 (process 10220) exited normally]  
(gdb)

Для более подробного анализа программы установим брейкпоинт на метку \_start, с которой начинается выполнение любой ассемблерной программы, и запустим её. (рис. 8)

(gdb) break \_start  
Breakpoint 1 at 0x8049000: file lab10-2.asm, line 12.  
(gdb) run  
Starting program: ~/work/arch-pc/lab10/lab10-2  
Breakpoint 1, \_start () at lab10-2.asm:12  
12 mov eax, 4

Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки \_start (рис. 8)

(gdb) disassemble \_start

Переключимся на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. 8)

(gdb) set disassembly-flavor intel  
(gdb) disassemble \_start

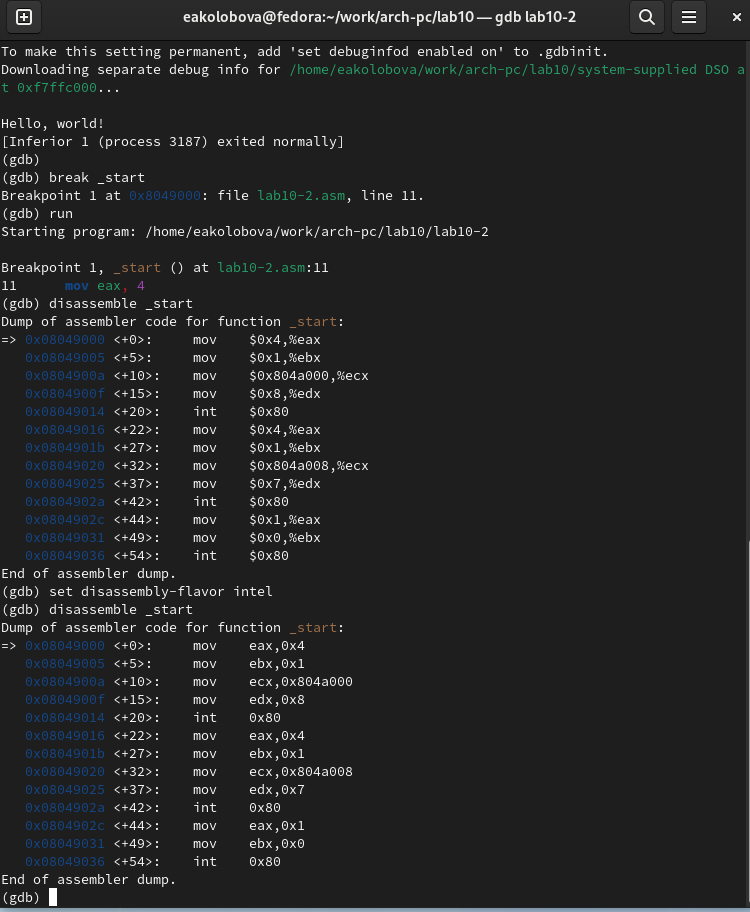


Рис. 8: Рис. 8. Окно отладчика

Перечислите различия отображения синтаксиса машинных команд в режимах ATT и Intel.

1. Включим режим псевдографики для более удобного анализа программы (рис. 9):

(gdb) layout asm  
(gdb) layout regs

Установим еще одну точку останова по адресу инструкции. Определим адрес предпоследней инструкции (mov ebx,0x0) и установим точку останова. (рис. 9)

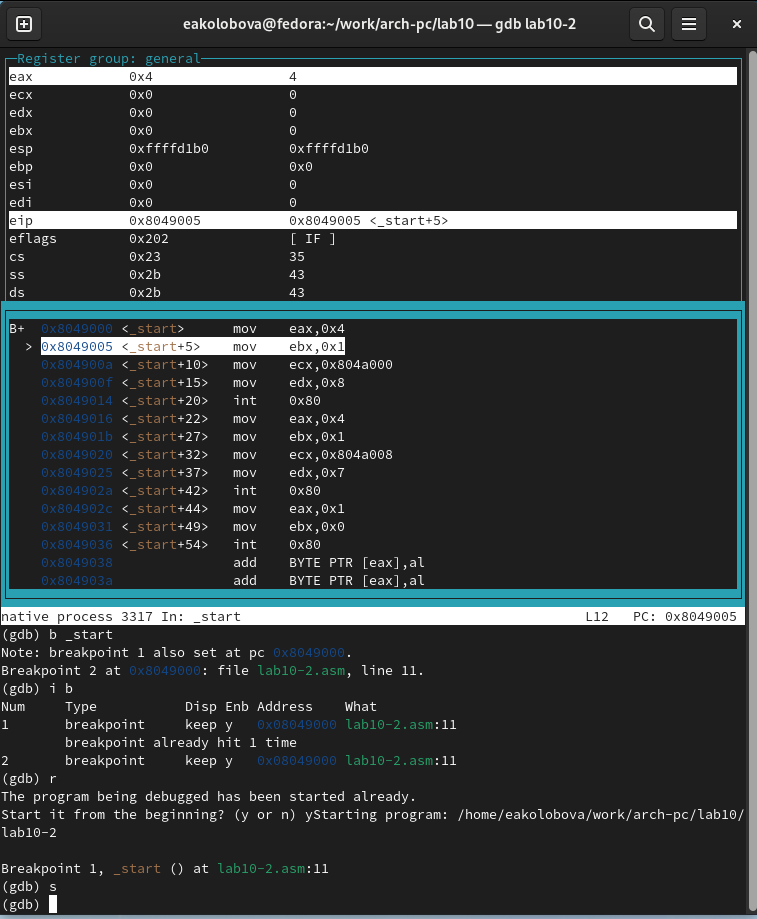


Рис. 9: Рис. 9. Режим псевдографики

Посмотрим информацию о всех установленных точках останова (рис. 10):

(gdb) i b

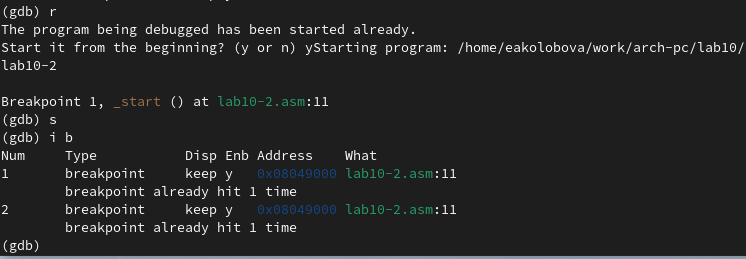


Рис. 10: Рис. 11. Просмотр информации о точке останова

Посмотрим значение переменной msg1 по имени (рис. 11)

(gdb) x/1sb &msg1  
0x804a000 <msg1>: "Hello, "

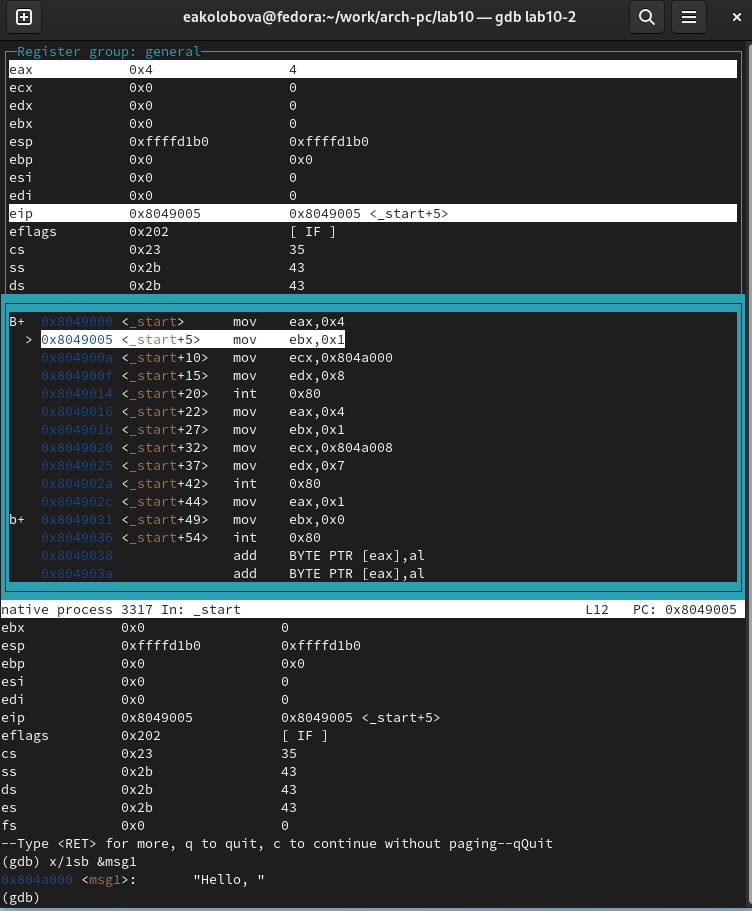


Рис. 11: Рис. 12. Просмотр значения переменной

Посмотрим значение переменной msg2 по адресу. Посмотрим инструкцию mov ecx,msg2 которая записывает в регистр ecx адрес перемененной msg Изменим первый символ переменной msg1 (рис. 12):

(gdb) set {char}msg1='h'  
(gdb) x/1sb &msg1  
0x804a000 <msg1>: "hello, "  
(gdb)

Заменим символ во второй переменной msg2. Выведем в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx. (рис. 12)

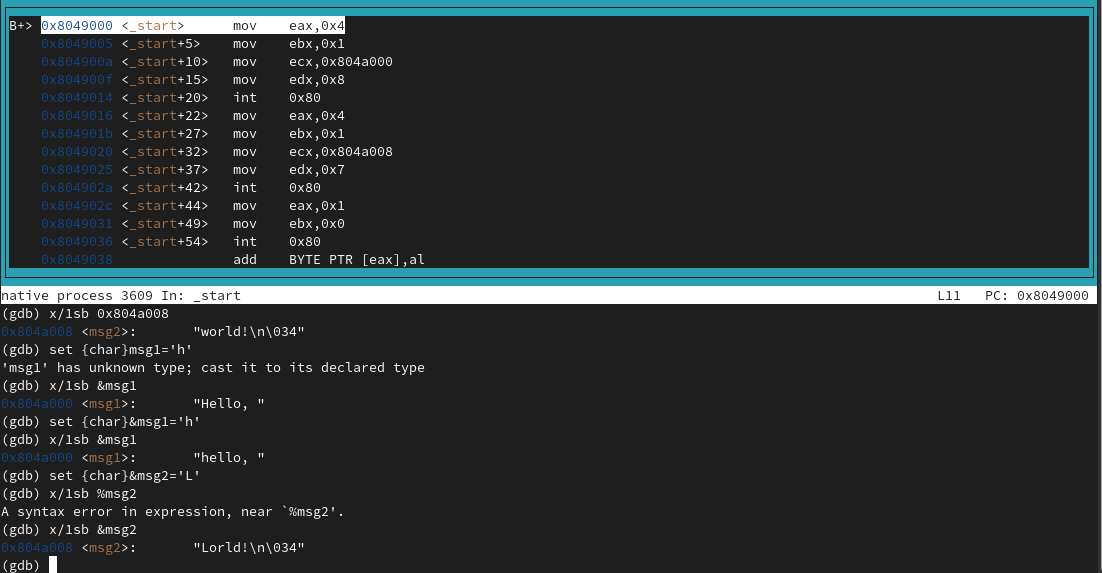


Рис. 12: Рис. 13. Замена символа и значения переменной

С помощью команды set изменим значение регистра ebx (рис. 13):

(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$3 = 50  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$4 = 2  
(gdb)

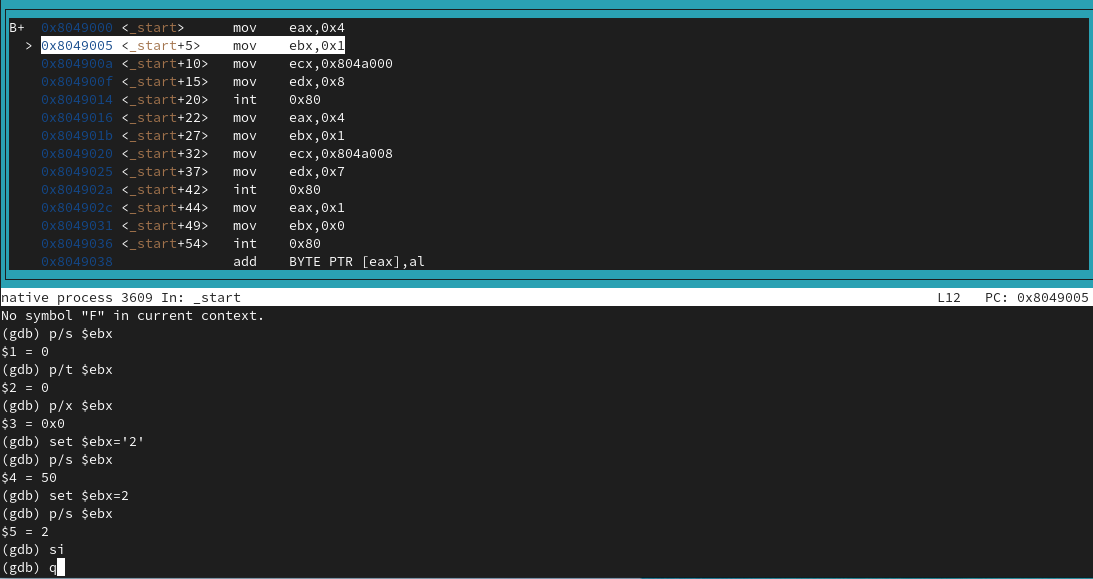


Рис. 13: Рис. 14. Изменение значения регистра ebx

Объясните разницу вывода команд p/s $ebx. Эта команда выводит строку, оканчивающуюся нулем.

Завершим выполнение программы с помощью команды continue (сокра- щенно c) и выйдем из GDB с помощью команды quit (сокращенно q). (рис. 14)

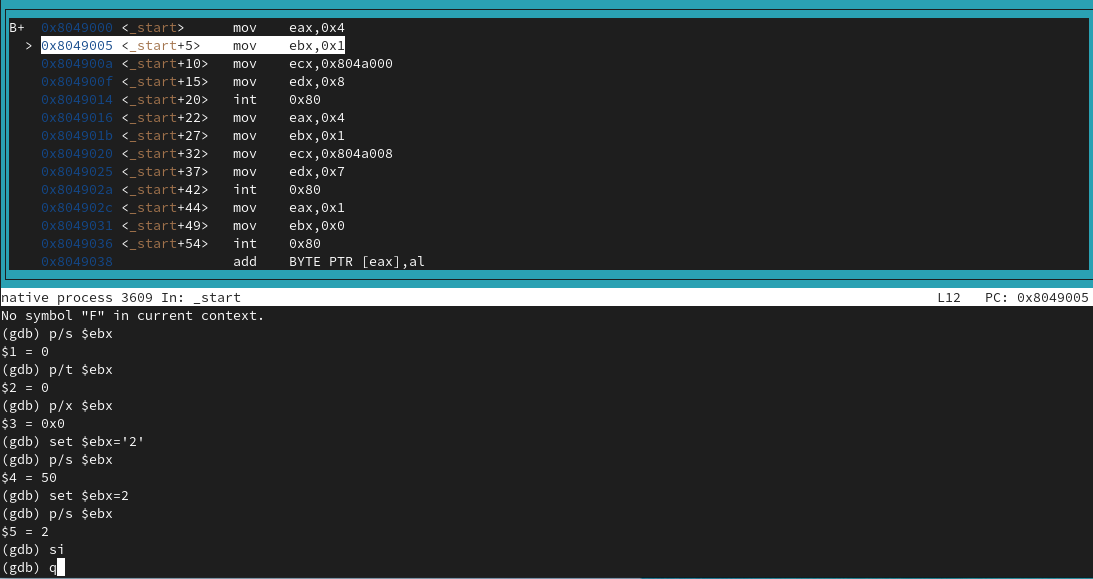


Рис. 14: Рис. 15. Завершение выполнения программы

1. Скопируем файл lab9-2.asm, созданный при выполнении лабораторной ра- боты No9, с программой выводящей на экран аргументы командной строки (Листинг 9.2) в файл с именем lab10-3.asm (рис. 15):

cp ~/work/arch-pc/lab09/lab9-2.asm ~/work/arch-pc/lab10/lab10-3.asm

Создадим исполняемый файл. (рис. 15)

nasm -f elf -g -l lab10-3.lst lab10-3.asm  
ld -m elf\_i386 -o lab10-3 lab10-3.o

Загрузим исполняемый файл в отладчик, указав аргументы (рис. 15):

gdb --args lab10-3 аргумент1 аргумент 2 'аргумент 3'

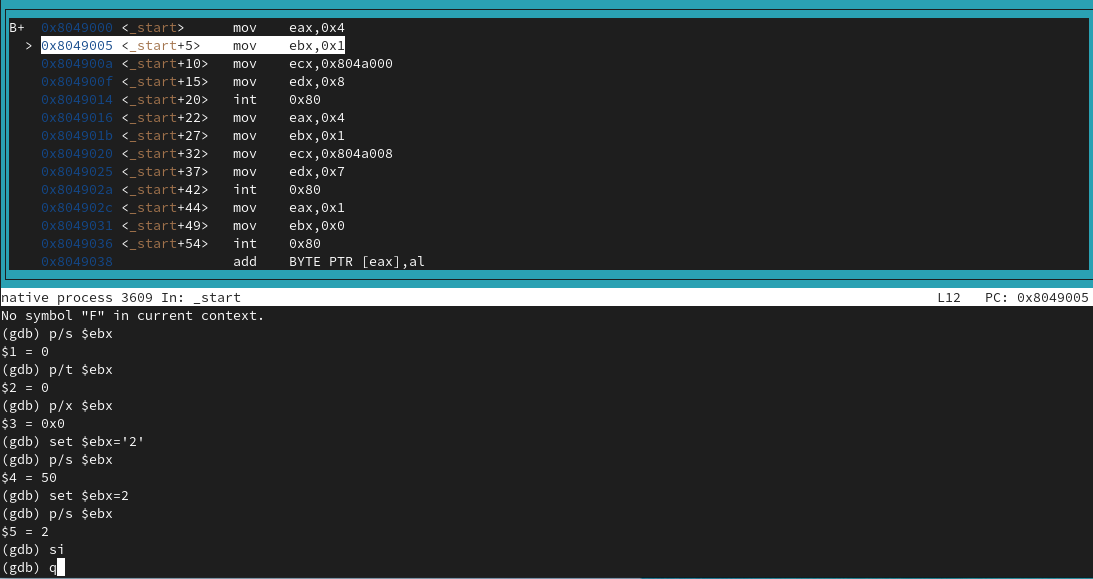


Рис. 15: Рис. 17. Создание исполняемого файла lab9-2

Для начала установим точку останова перед первой инструкцией в программе и запустим ее. (рис. 16)

(gdb) b \_start  
(gdb) run

Адрес вершины стека хранится в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы):

(gdb) x/x $esp  
0xffffd200: 0x05

Как видно, число аргументов равно 5 – это имя программы lab10-3 и непо- средственно аргументы: аргумент1, аргумент, 2 и ‘аргумент 3’. Посмотрим остальные позиции стека (рис. 17)

Почему шаг изменения адреса равен 4, т.к. программа запускается с четырьмя (вместе с названием программы) аргументами

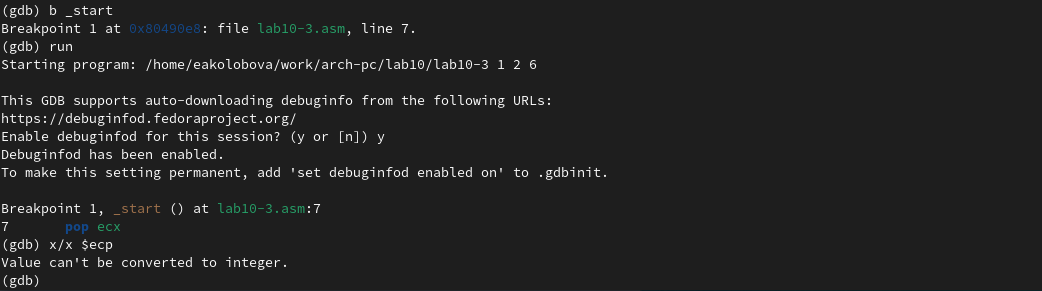


Рис. 16: Рис. 18. Установка точки останова

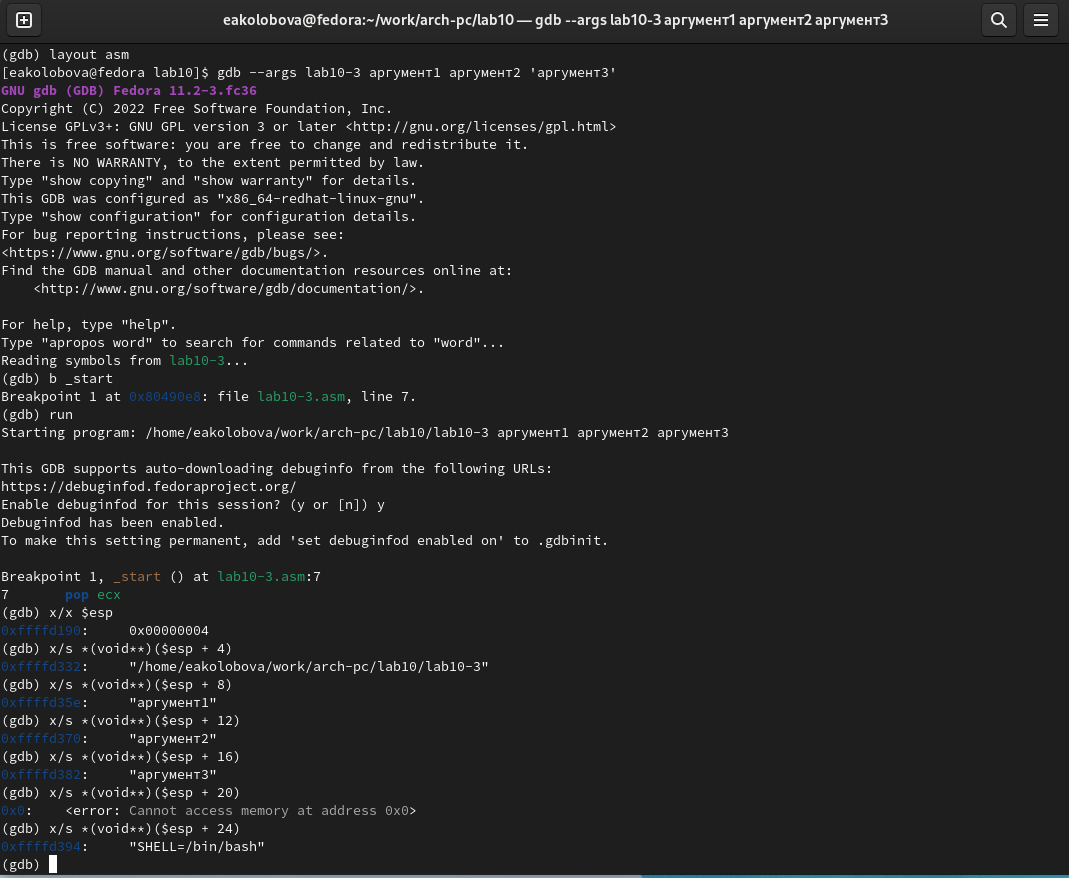


Рис. 17: Рис. 19. Просмотр позиций стека

## 2.1 **Задание для самостоятельной работы**

1. Преобразовать программу из лабораторной работы No9 (Задание No1 для самостоятельной работы), реализовав вычисление значения функции f(x), как подпрограмму. (рис. 18, 19)

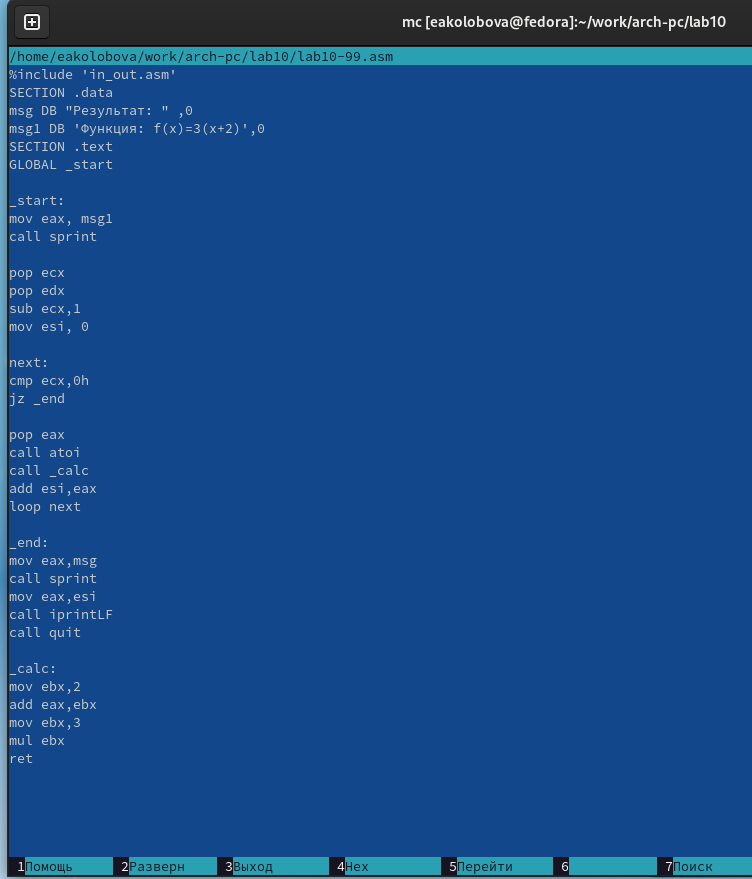


Рис. 18: Рис. 14. Текст программы

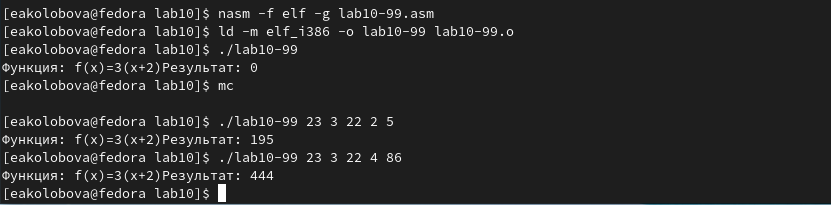


Рис. 19: Рис. 15. Компоновка и запуск файла

1. В листинге 10.3 приведена программа вычисления выражения (3+2)\*4+5. При запуске данная программа дает неверный результат. Проверить это. С помощью отладчика GDB, анализируя изменения значений регистров, определить ошибку и исправить ее.

Как видно на рис. 21, 22, результат сложения 3+2 записывается в регистр ebx, а перемножаются после этого значения регистров ecx и eax вместо ecx и ebx, как предполагается в тексте программы. После этого к значению регистра ebx - 5 - прибавляется 5, и этот результат программа выводит, как конечное значение. Ошибка заключается в том, что для операции умножения суммы на значение ecx, первый множитель - сумма - находится в регистре ebx, а умножается всегда регистр eax. (рис. 20, 21, 22)

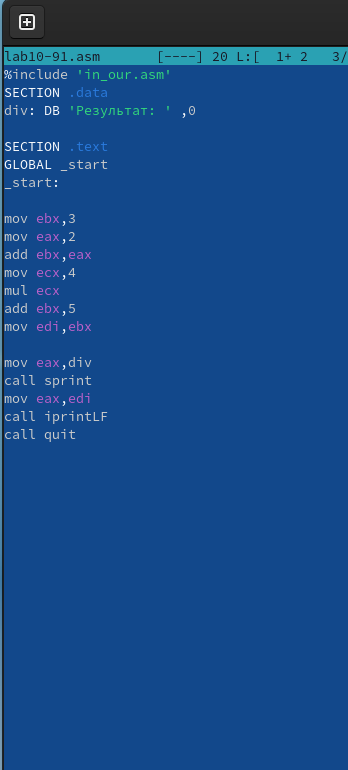


Рис. 20: Рис. 14. Текст программы

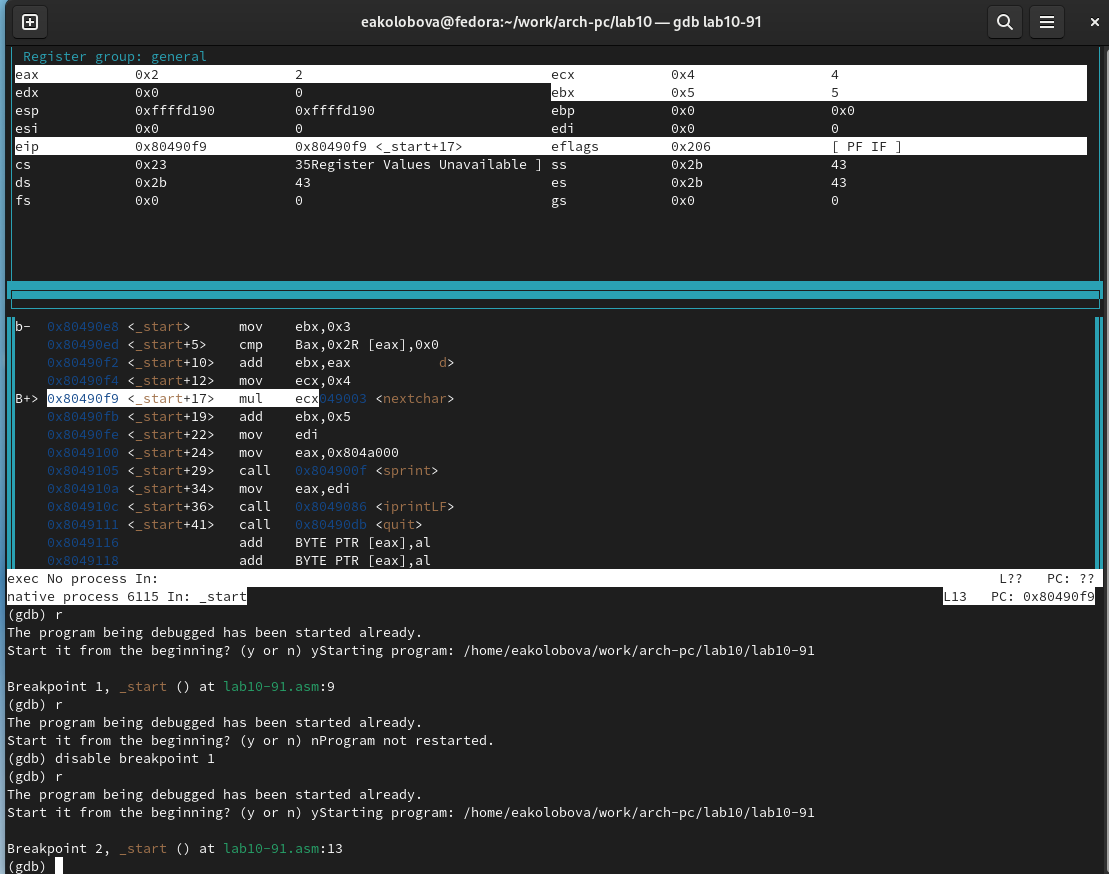


Рис. 21: Рис. 15. Отладка программы

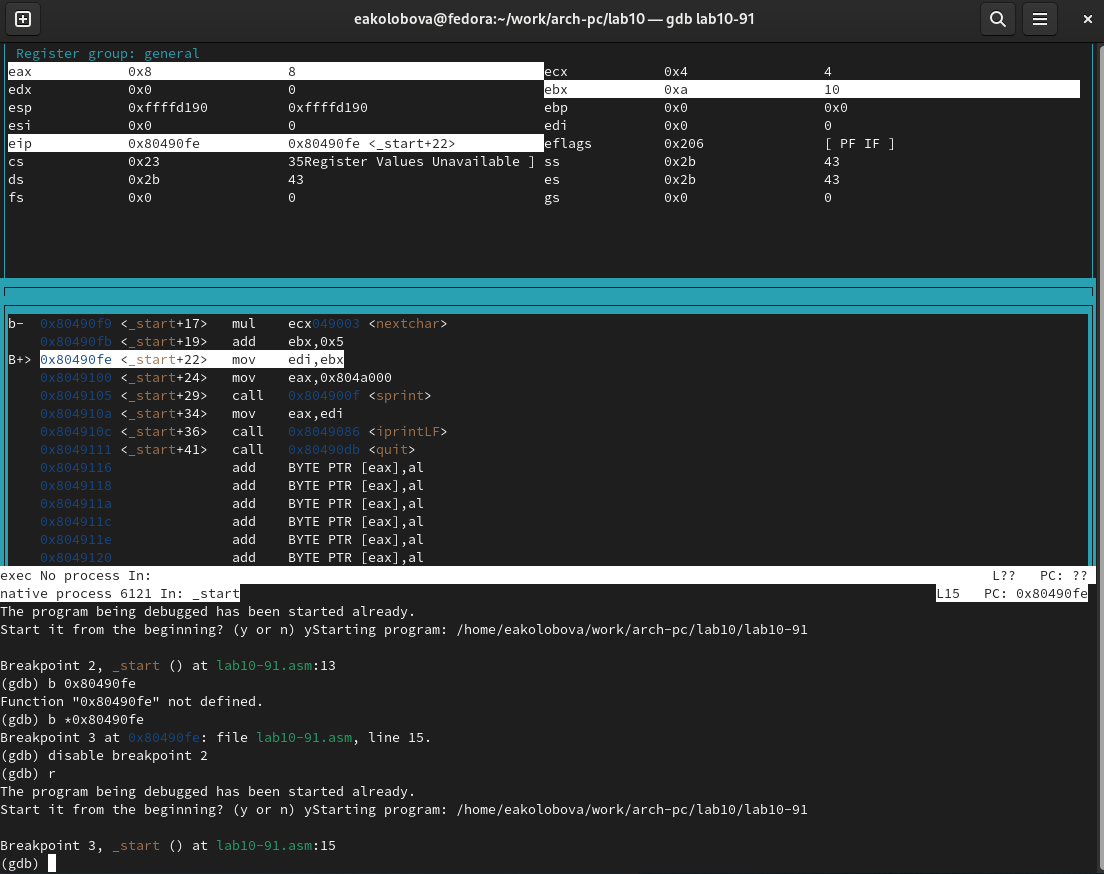


Рис. 22: Рис. 16. Отладка программы

Ссылка на репозиторий: https://github.com/eakolobova/study\_2022-2023\_arch-pc/tree/master/labs/lab010/report

# 3 **Выводы**

Результатом проведенной работы является приобретение приобретение навыков написания программ с использованием подпрограмм и знакомство с методами отладки при помощи GDB и его основными возможностями.