Отчет по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB

Мальянц Виктория Кареновна

Содержание

# 1 Цель работы

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

# 2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программам с помощью GDB
3. Добавление точек останова
4. Работа с данными программы в GDB
5. Обработка аргументов командной строки в GDB
6. Выполнение задания для самостоятельной работы

# 3 Выполнение лабораторной работы

## 3.1 Реализация подпрограмм в NASM

Создаю каталог для программ лабораторной работы № 9, перехожу в него и создаю файл lab09-1.asm (рис. 1).

Рис. 1: Создание каталога и файла для программы

Рис. 1: Создание каталога и файла для программы

С помощью команды cp копирую файл in\_out.asm (рис. 2).

Рис. 2: Копирование файла

Рис. 2: Копирование файла

Ввожу в файл lab09-1.asm программу с использованием вызова подпрограммы (рис. 3).

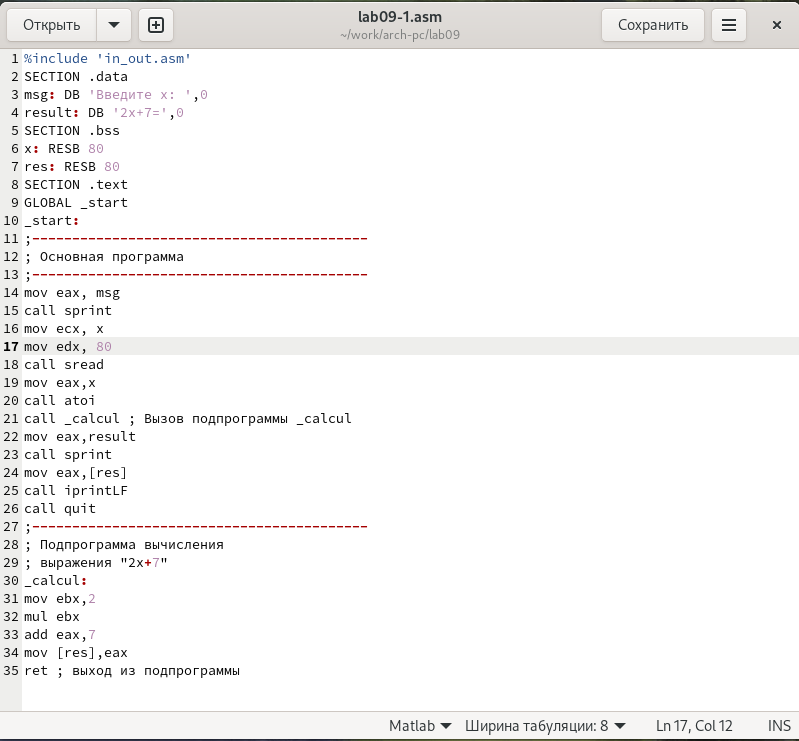


Рис. 3: Редактирование файла

Создаю исполняемый файл и запускаю его. Проверяю работу исполняемого файла для значения x равного 5. Убеждаюсь в том, что программа работает корректно (рис. 4).

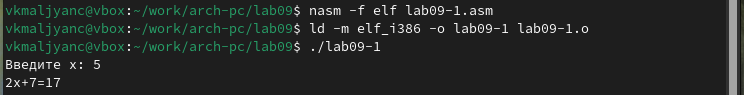


Рис. 4: Запуск исполняемого файла

Изменяю текст программы, добавив подпрограмму \_subcalcul в подпрограмму \_calcul (рис. 5).

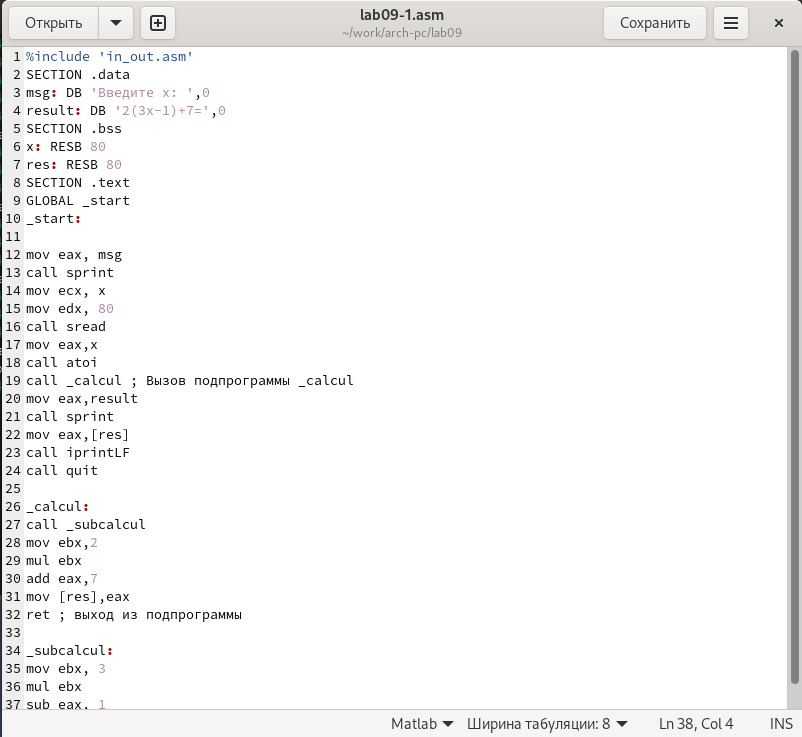


Рис. 5: Редактирование файла

Создаю исполняемый файл и запускаю его. Проверяю работу исполняемого файла для значения x равного 5. Убеждаюсь в том, что программа работает корректно (рис. 6).

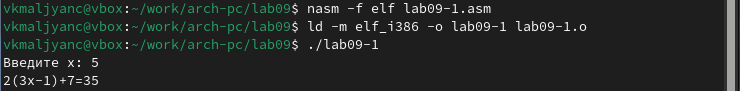


Рис. 6: Запуск исполняемого файла

Листинг программы:

%include 'in\_out.asm'  
SECTION .data  
msg: DB 'Введите x: ',0  
result: DB '2(3x-1)+7=',0  
SECTION .bss  
x: RESB 80  
res: 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 ; Вызов подпрограммы \_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 ; выход из подпрограммы  
  
\_subcalcul:  
mov ebx, 3  
mul ebx  
sub eax, 1  
ret

## 3.2 Отладка программам с помощью GDB

С помощью команды touch создаю файл lab09-2.asm (рис. 7).

Рис. 7: Создание файла

Рис. 7: Создание файла

Ввожу в файл lab09-2.asm программу вывода сообщения Hello world! (рис. 8).

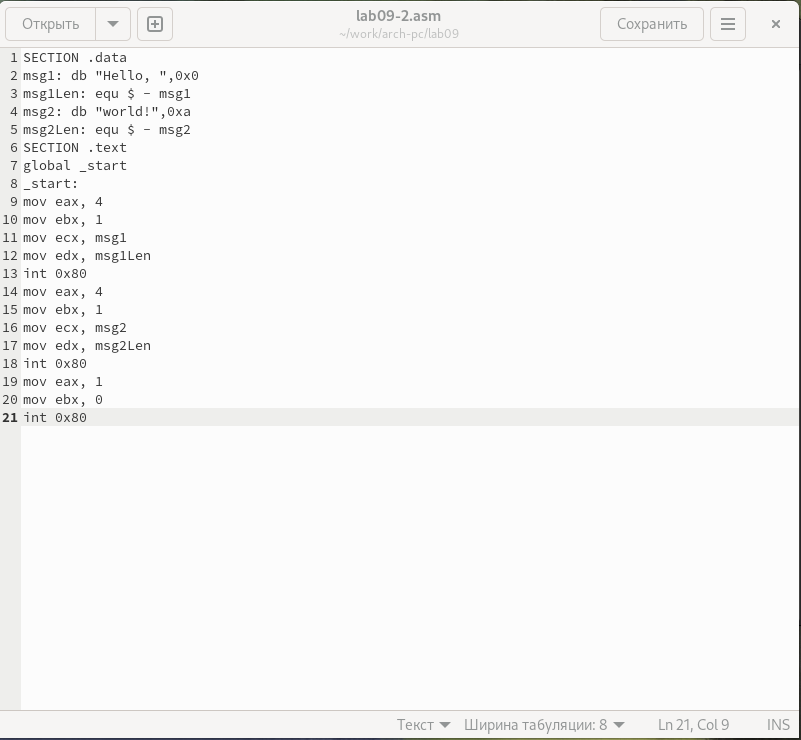


Рис. 8: Редактирование файла

В исполняемом файле добавляю отладочную информацию. Провожу трансляцию программы с ключом ‘-g’ для работы с GDB (рис. 9).

Рис. 9: Запуск исполняемого файла

Рис. 9: Запуск исполняемого файла

Загружаю исполняемый файл в отладчик GDB (рис. 10).

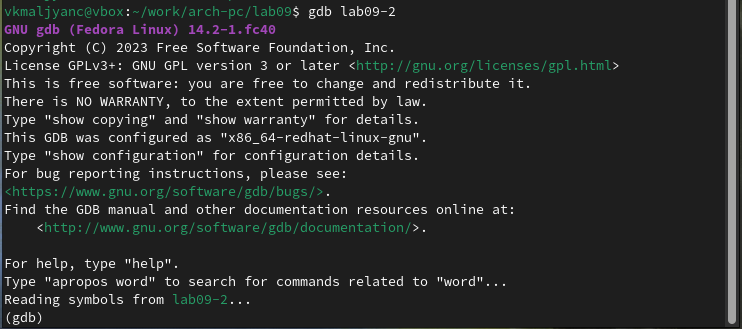


Рис. 10: Загрузка исполняемого файла в отладчик GDB

Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run. Программа работает корректно (рис. 11).

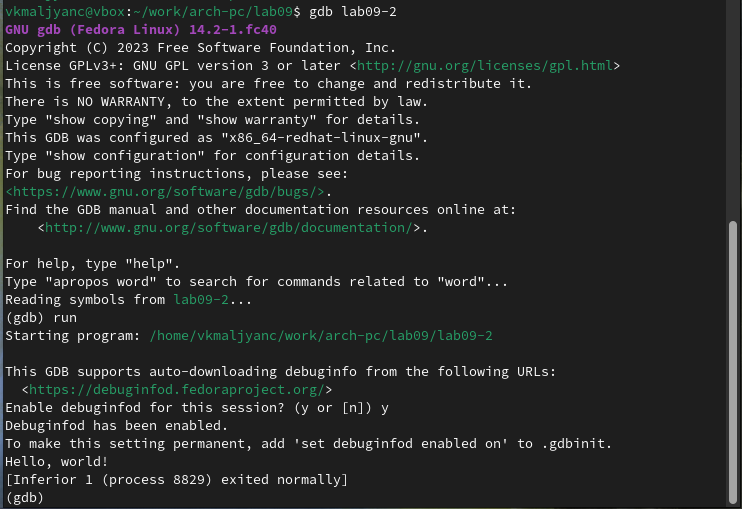


Рис. 11: Запуск программы в отладчике GDB

Устанавливаю брейкпоинт на метку \_start и запускаю программу (рис. 12).

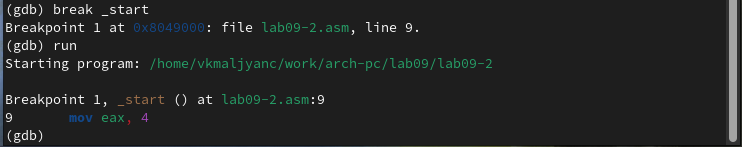


Рис. 12: Установка брейкпоинта и запуск программы в отладчике GDB

Смотрю дисассимилированный код программы с помощью команды disassemble, начиная с метки \_start (рис. 13).



Рис. 13: Дисассимилированный код программы

Переключаю на отображение команд с Intel’овским синтаксисом, введя команду set disassembly-flavor intel (рис. 14).

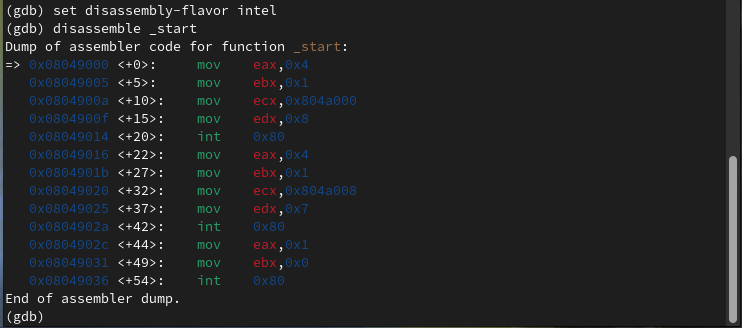


Рис. 14: Дисассимилированный код программы

Различия отображения синтаксиса машинных команд в режимах ATT и Intel: 1. Различный порядок операндов (в ATT операнды записываются в порядке источник, назначение, в Intel перанды записываются в порядке назначение, источник) 2. Различия в именах регистров (в ATT регистры обозначаются с префиксом %, в Intel регистры обозначаются без префикса) 3. Различный размер (в ATT размер данных указывается с помощью суфиксов, в Intel размер данных определяется контекстом команды)

Включаю режим псевдографики для более удобного анализа программы (рис. 15) (рис. 16) (рис. 17).

Рис. 15: Включение режима псевдографики

Рис. 15: Включение режима псевдографики

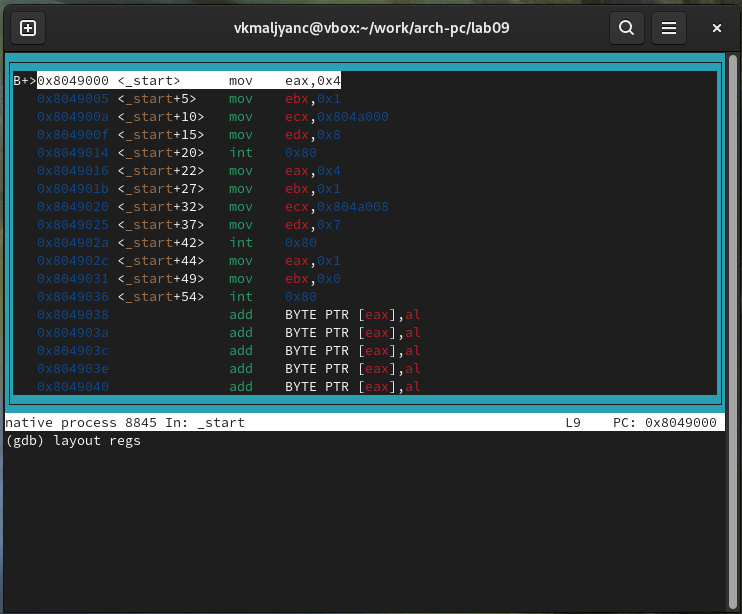


Рис. 16: Включение режима псевдографики

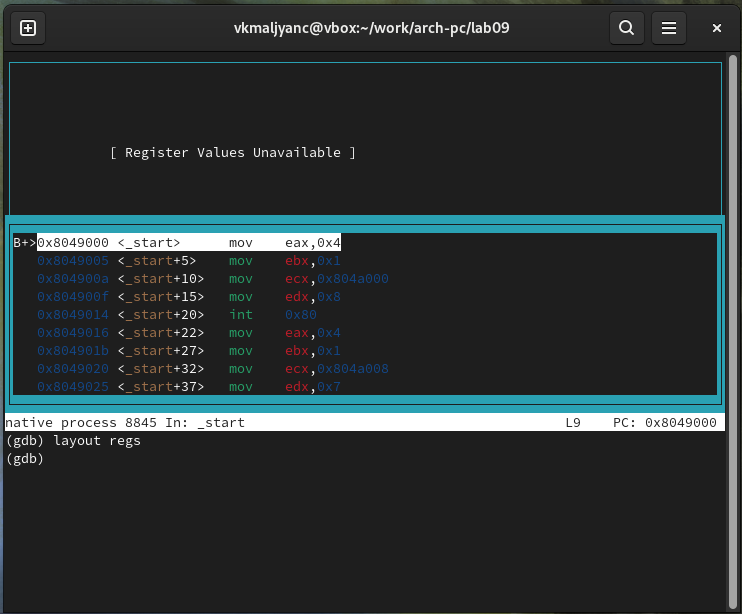


Рис. 17: Включение режима псевдографики

## 3.3 Добавление точек останова

Проверяю наличие точки останова по имени метки (\_start) с помощью команды info breakpoints. Точка останова существует (рис. 18).

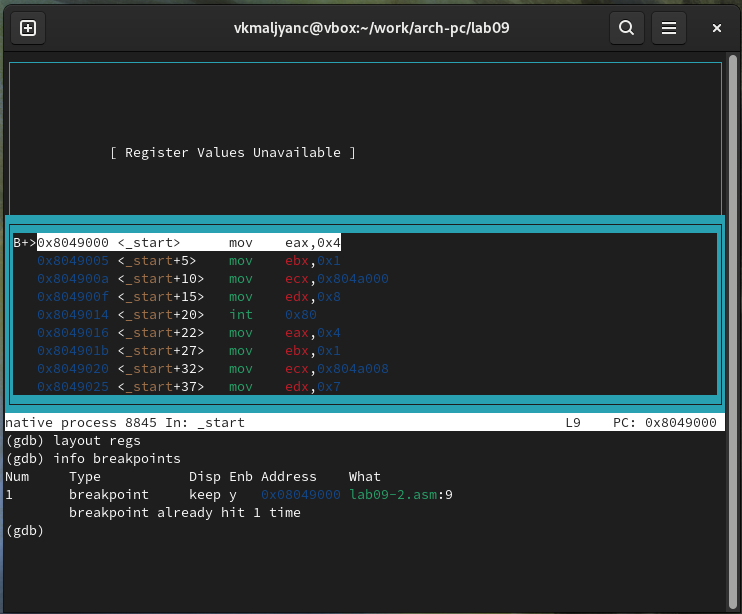


Рис. 18: Список точек останова

Определяю адрес предпоследней инструкции (mov ebx,0x0) и устанавливаю точку останова (рис. 19).

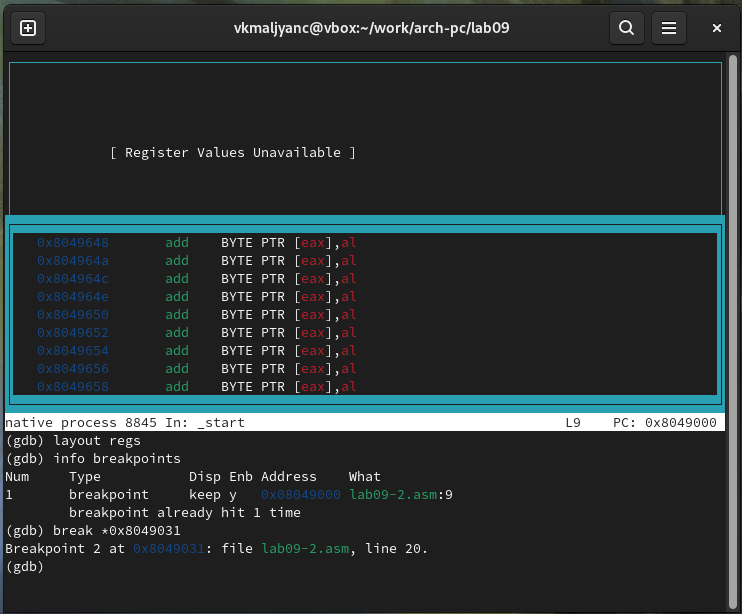


Рис. 19: Установка точки останова

Просматриваю информацию о всех установленных точках останова (рис. 20).

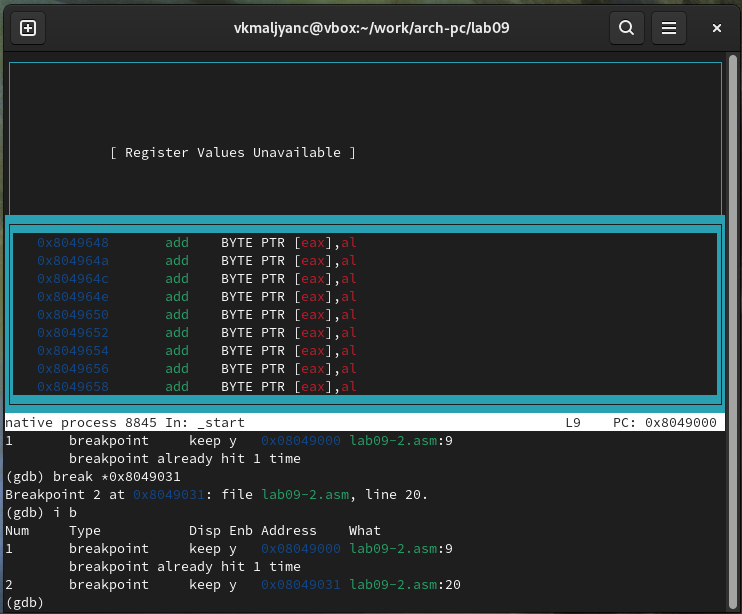


Рис. 20: Список точек останова

## 3.4 Работа с данными программы в GDB

Выполняю 5 инструкций с помощью команды stepi. Изменяются значения регистров: eax, ebx, ecx, edx (рис. 21) (рис. 22) (рис. 23) (рис. 24) (рис. 25).

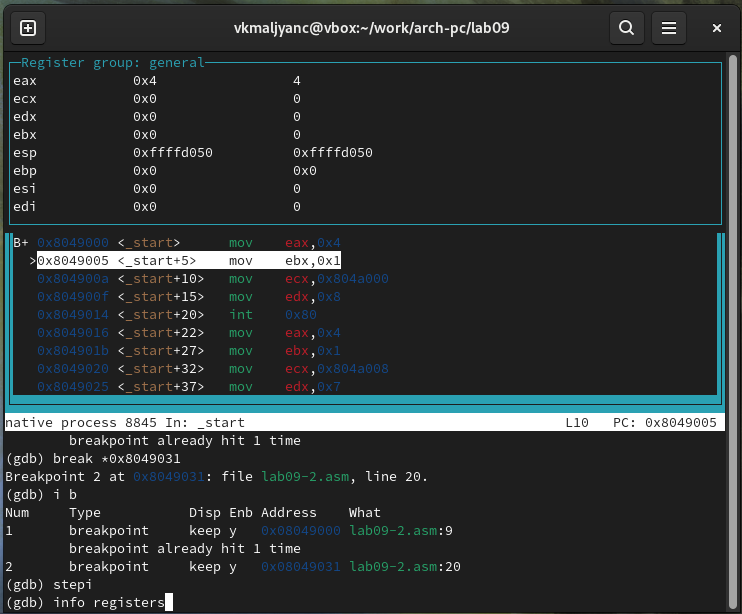


Рис. 21: stepi 1

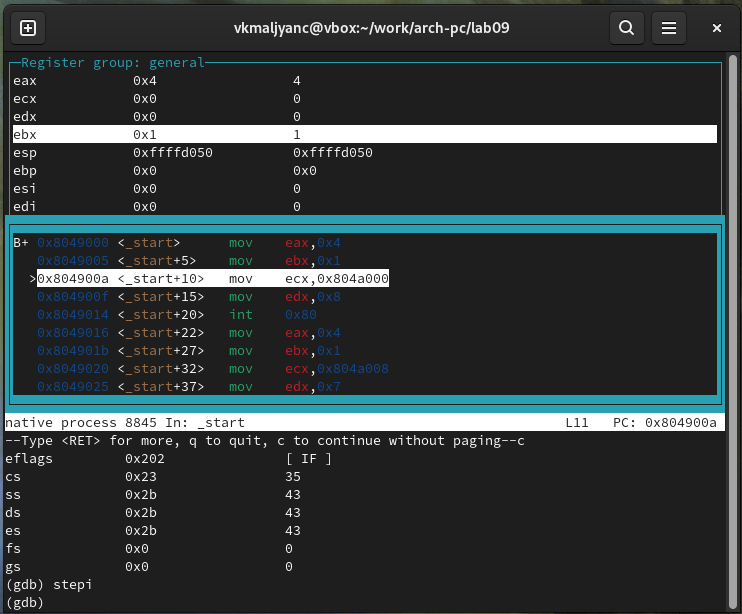


Рис. 22: stepi 2

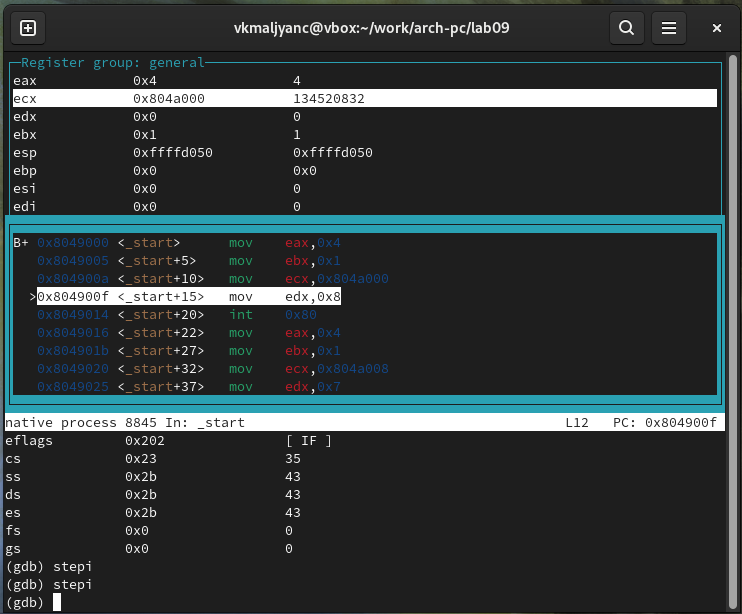


Рис. 23: stepi 3

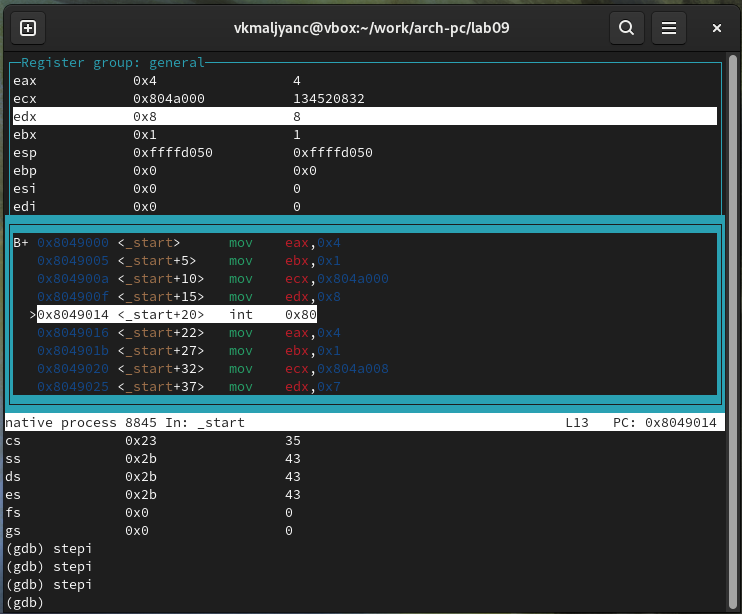


Рис. 24: stepi 4

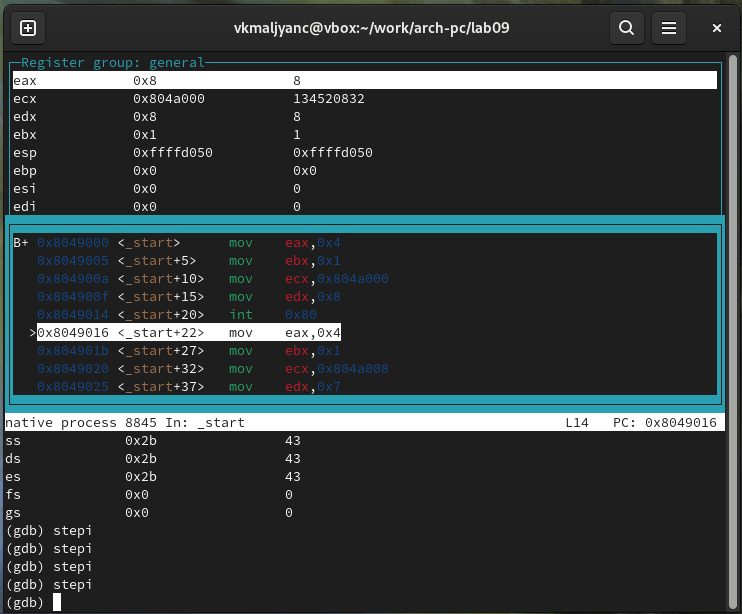


Рис. 25: stepi 5

Просматриваю содержимое регистров с помощью команды info registers (рис. 26) (рис. 27).

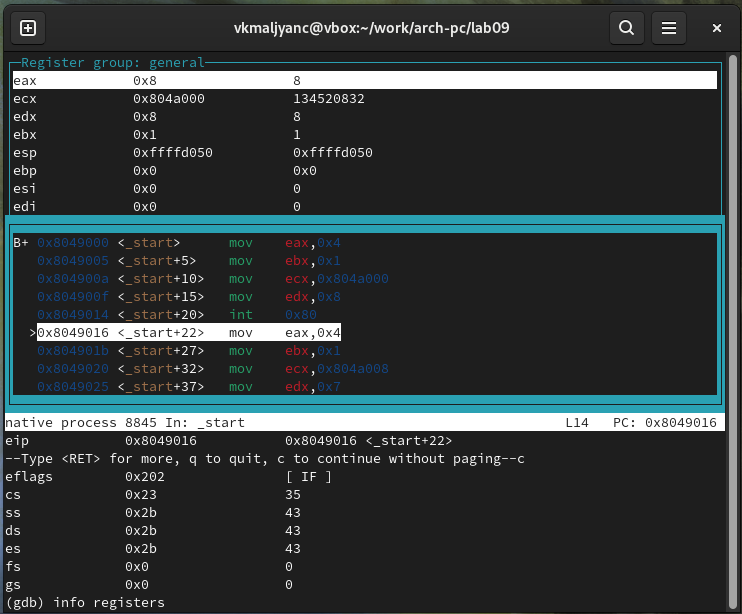


Рис. 26: Содержимое регистров

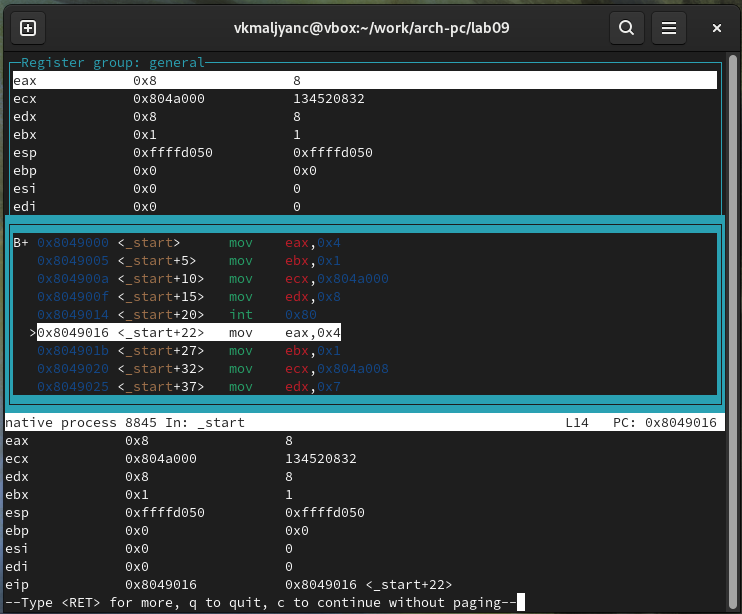


Рис. 27: Содержимое регистров

Просматриваю значение переменной msg1 по имени и переменной msg2 по адресу (рис. 28).

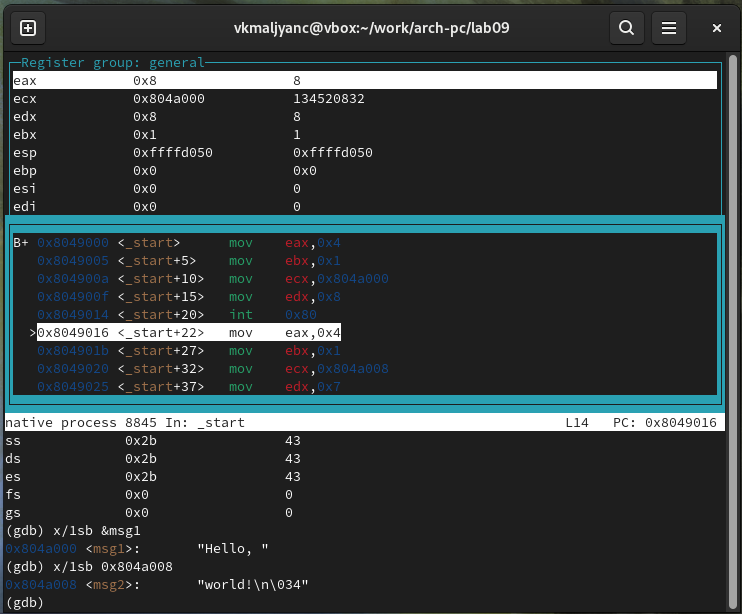


Рис. 28: Просмотр значения переменной msg1 по имени и msg2 по адресу

Изменяю первый символ переменной msg1 с помощью команды set (рис. 29).

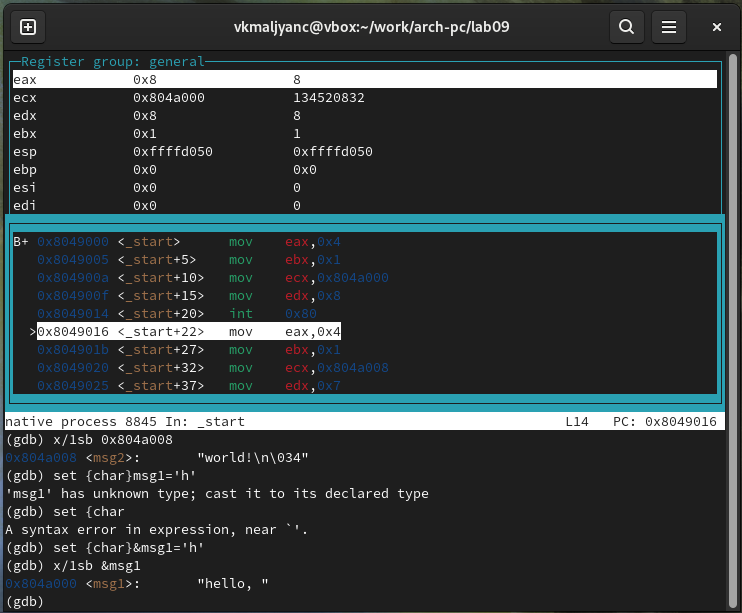


Рис. 29: Изменение первого символа переменной msg1 с помощью команды set

Изменяю первый символ переменной msg2 с помощью команды set (рис. 30).

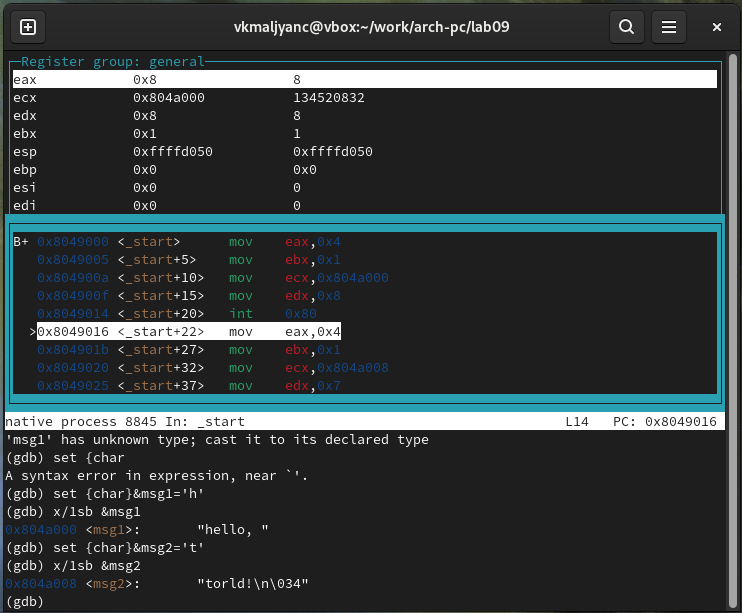


Рис. 30: Изменение первого символа переменной msg2 с помощью команды set

Вывожу в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx (рис. 31).

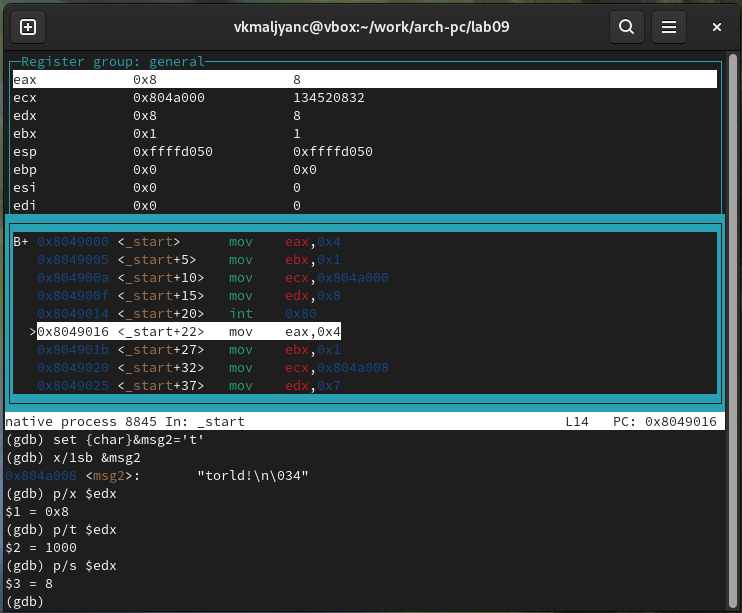


Рис. 31: Вывод значения регистра edx в различных форматах

С помощью команды set изменяю значение регистра ebx (рис. 32).

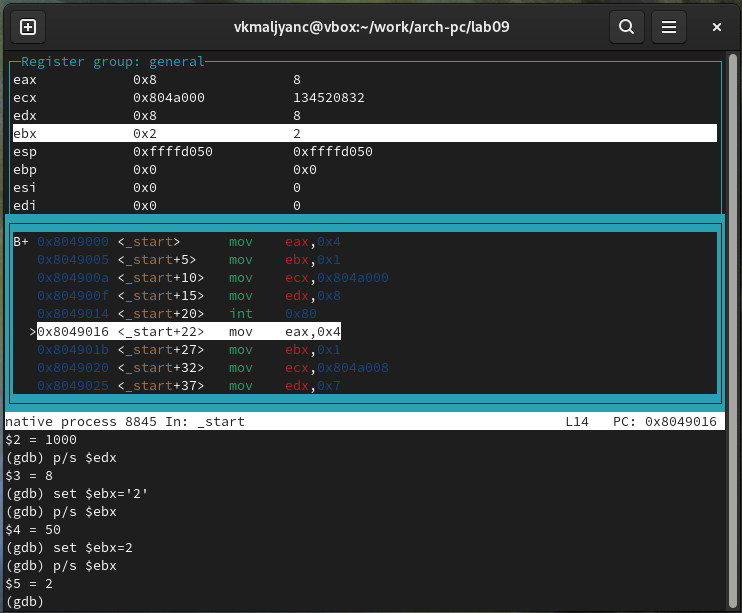


Рис. 32: Изменение значения регистра ebx

Разница вывода команд p/s $ebx: при введении команды p/s $ebx=‘2’ вывод 50, так как в программе ‘2’ рассматривается как символ, в ASCII равен 50, при введении команды p/s $ebx=2 вывод 2, так как введено число.

## 3.5 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm в файл с именем lab09-3.asm рис. 33).

Рис. 33: Копирование файла

Рис. 33: Копирование файла

Создаю исполняемый файл 34).

Рис. 34: Создание исполняемого файла

Рис. 34: Создание исполняемого файла

Загружаю исполняемый файл с использованием ключа –args в отладчик, указав аргументы 35).

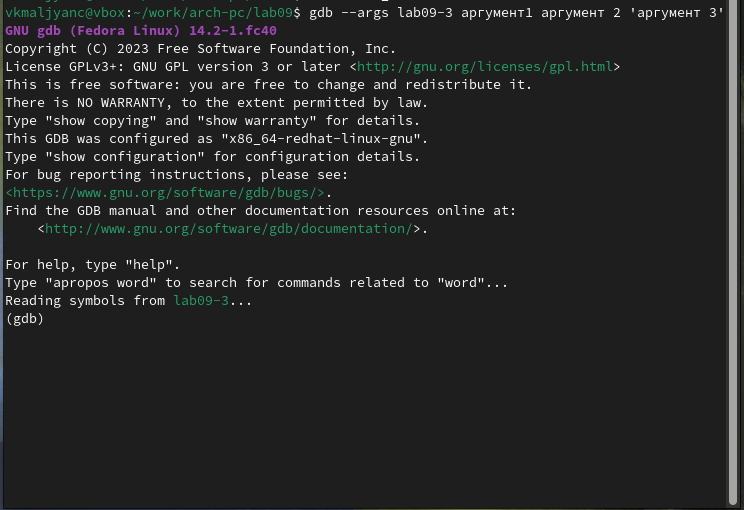


Рис. 35: Загрузка исполняемого файла

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее 36).

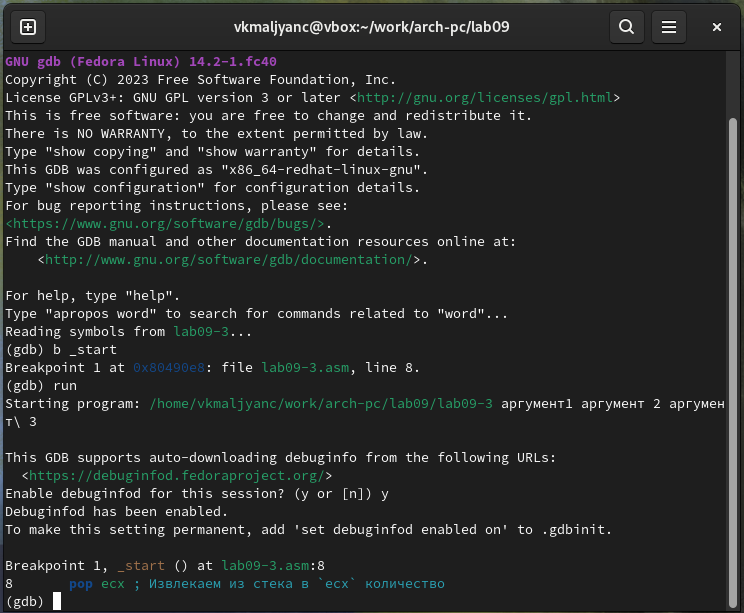


Рис. 36: Установка точки останова

Просмотриваю позиции стека 37).

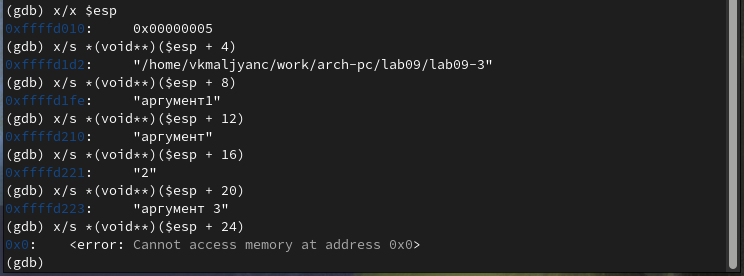


Рис. 37: Просмотр позиций стека

Шаг изменения адреса равен 4, потому что каждый элемент стека занимает 4 байта.

## 3.6 Выполнение задания для самостоятельной работы

1. С помощью команды touch создаю файл lab09-4.asm (рис. 38).

Рис. 38: Создание файла

Рис. 38: Создание файла

Преобразовываю программу из лабораторной работы №8 (Задание №1 для самостоятель- ной работы), реализовав вычисление значения функции f(x) как подпрограмму (рис. 39).

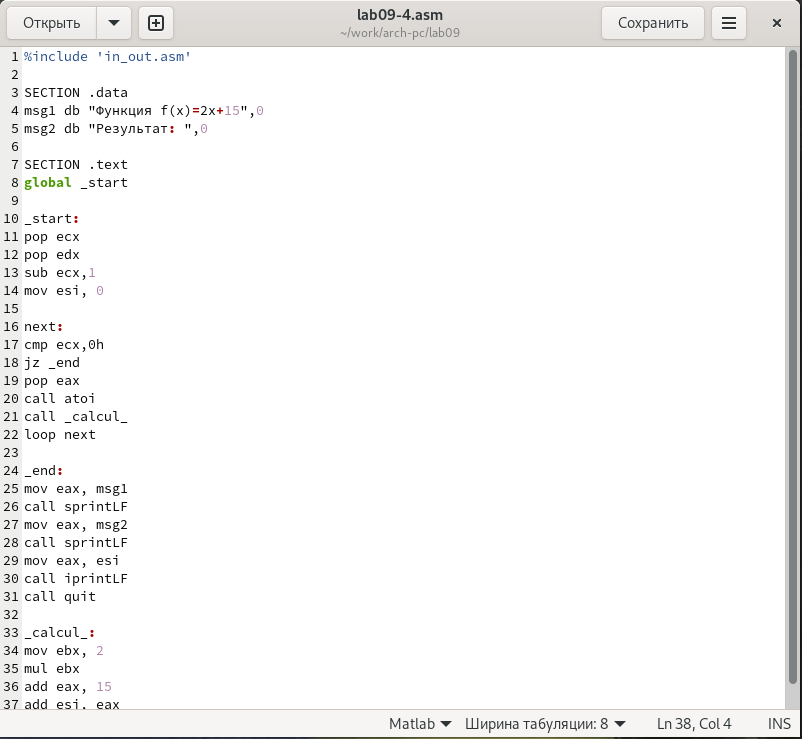


Рис. 39: Редактирование файла

Создаю исполняемый файл и запускаю его. Проверяю работу исполняемого файла, указав аргументы: 1 2 3 4 5 (рис. 40). Убеждаюсь в том, что программа работает корректно.

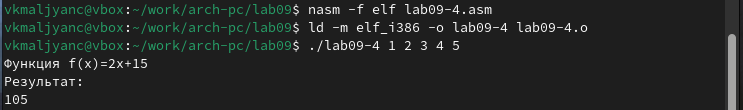


Рис. 40: Запуск исполняемого файла

Листинг программы:

%include 'in\_out.asm'  
  
SECTION .data  
msg1 db "Функция f(x)=2x+15",0  
msg2 db "Результат: ",0  
  
SECTION .text  
global \_start  
  
\_start:  
pop ecx   
pop edx   
sub ecx,1  
mov esi, 0   
  
next:  
cmp ecx,0h   
jz \_end   
pop eax   
call atoi   
call \_calcul\_  
loop next  
  
\_end:  
mov eax, msg1  
call sprintLF  
mov eax, msg2  
call sprintLF  
mov eax, esi   
call iprintLF   
call quit   
  
\_calcul\_:  
mov ebx, 2  
mul ebx  
add eax, 15  
add esi, eax  
ret

1. С помощью команды touch создаю файл lab09-5.asm (рис. 41).

Рис. 41: Создание файла

Рис. 41: Создание файла

Ввожу в файл lab09-5.asm программу вычисления выражения (3+2)\*4+5 (рис. 42).

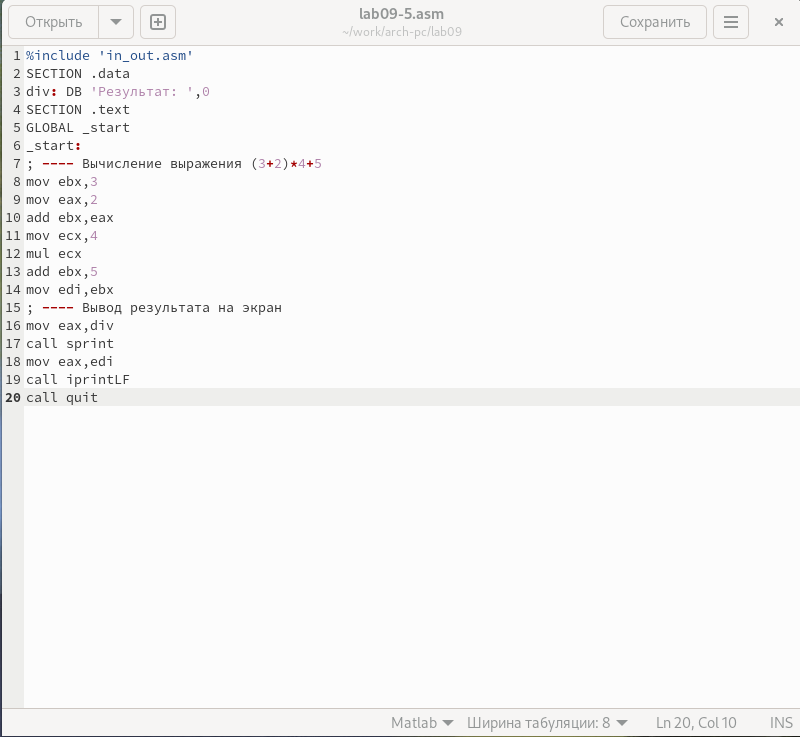


Рис. 42: Редактирование файла

Создаю исполняемый файл и запускаю его. Проверяю работу исполняемого файла и убеждаюсь в том, что программа работает некорректно (рис. 43).

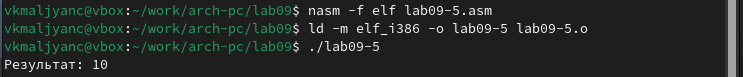


Рис. 43: Запуск исполняемого файла

В исполняемом файле добавляю отладочную информацию. Провожу трансляцию программы с ключом ‘-g’ для работы с GDB. Загружаю исполняемый файл в отладчик GDB (рис. 44).

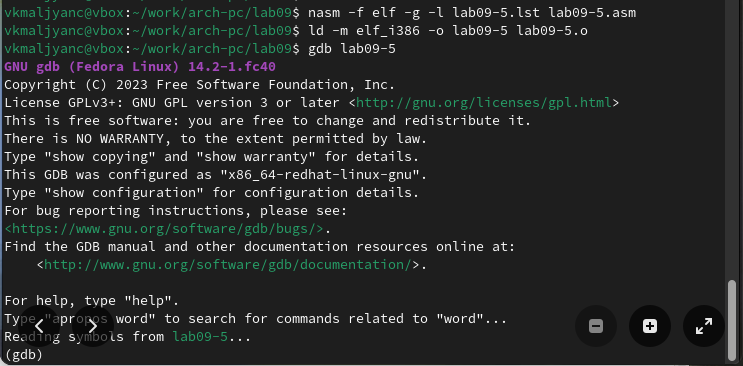


Рис. 44: Запуск исполняемого файла и его загрузка в отладчик GDB

Устанавливаю брейкпоинт на метку \_start и запускаю программу (рис. 45).

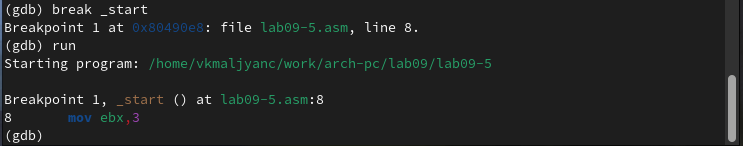


Рис. 45: Установка брейкпоинта и запуск программы в отладчике GDB

Включаю режим псевдографики для более удобного анализа программы. С помощью команды stepi просматриваю значения регистров. В программе сумму eax и ebx записывали в ebx, но так как mul ecx умножает eax на ecx и записывает результат в eax, то на ecx умножилась не сумма eax и ebx, а eax, поэтому при выполнении mul ecx значение регистра ebx не изменилось (рис. 46) (рис. 47).

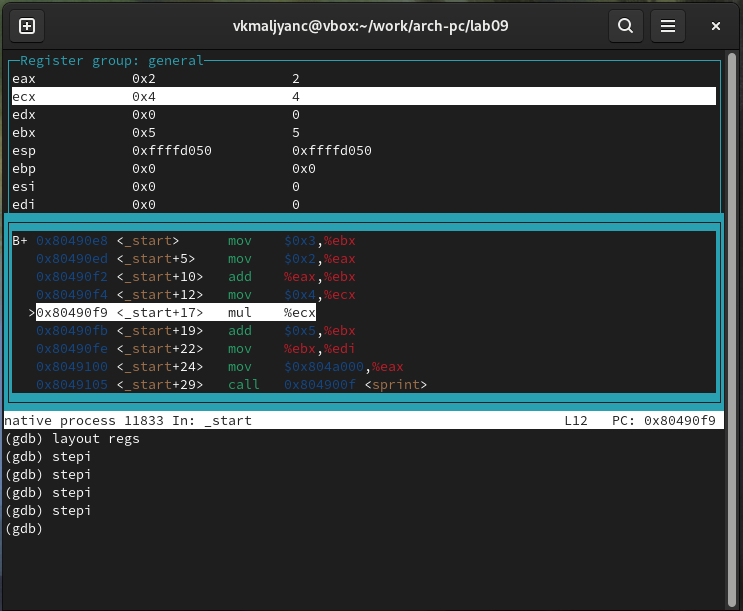


Рис. 46: Просмотр значений регистров

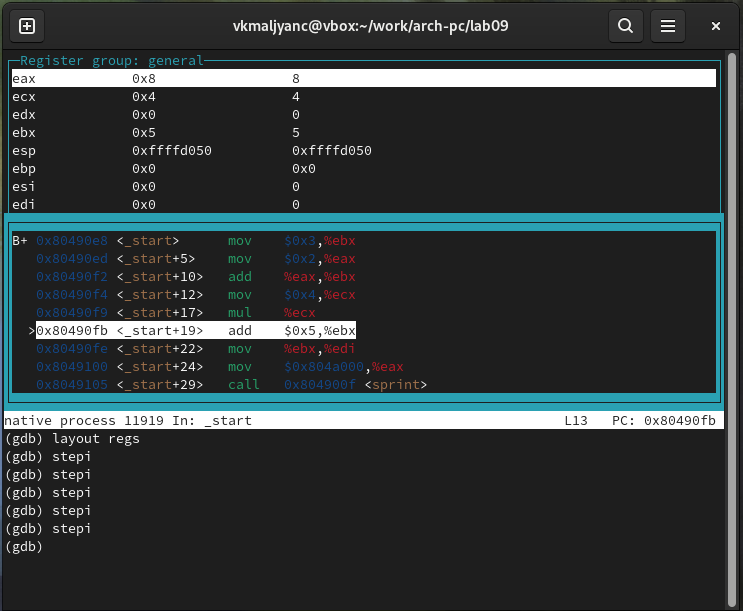


Рис. 47: Просмотр значений регистров

Изменяю текст программы, чтобы она работала корректно (рис. 48).

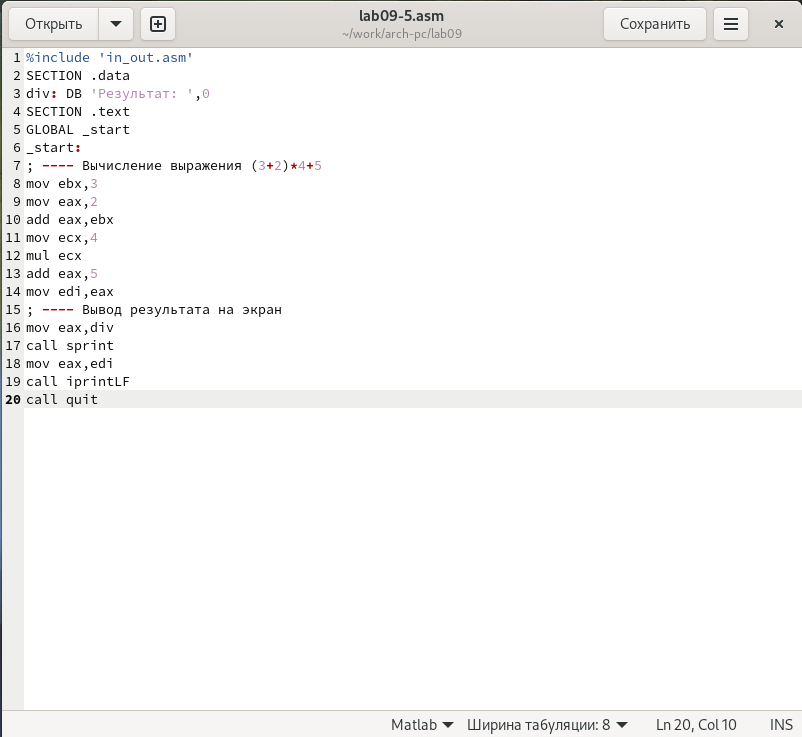


Рис. 48: Редактирование файла

Создаю исполняемый файл и запускаю его. Проверяю работу исполняемого файла и убеждаюсь в том, что программа работает корректно (рис. 49).

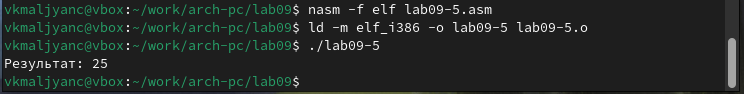


Рис. 49: Запуск исполняемого файла

Листинг программы:

%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

# 4 Выводы

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