Отчет по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB

Карачевцева Елизавета

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 Выполнение лабораторной работы

1) Я создала каталог lab9 и создал файл lab9-1.asm

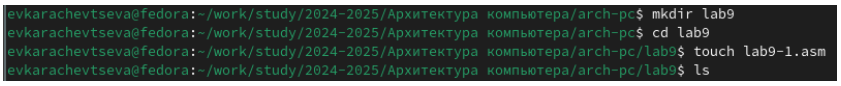

Рис. 2.1: Создание каталога и файла

2)Я ввела текст листинга в файл и запустила программу.

```
 GNU nano 7.2  /home/evkarachevtseva/work/study
%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:
;-----------------------------------------
; Основная программа
;-----------------------------------------
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
;-----------------------------------------
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
```

Рис. 2.2: Текст программы

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ nasm -f elf lab9-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ld -m elf_i386 -o lab9-1 lab9-1.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ./lab9-1
Введите x: 5
2x+7=17
```

Рис. 2.3: Работа программы

3) Я изменила текст программы, чтобы она решала выражение f(g(x)).

```
 GNU nano 7.2  /home/evkarachevtseva/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9/lab9-1.asm
%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 80
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
mov [res],eax
ret
_subcalcul:
mov ebx,3
mul ebx
sub eax,1
ret
```

Рис. 2.4: Измененный текст программы

```
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ nasm -f elf lab9-1.asm
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ld -m elf_i386 -o lab9-1 lab9-1.o
evkarachevtseva@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab9$ ./lab9-1
f(x)=2x+7
g(x)=3x-1
Введите x: 1
f(g(x))=11
```

Рис. 2.5: Проверка работы программы

4)Я создала файл lab9-2.asm и вписала туда программу.

```
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
```

Рис. 2.6: Текст второй программы

5)Я загрузила и запустила файл второй программы в отладчик gdb.

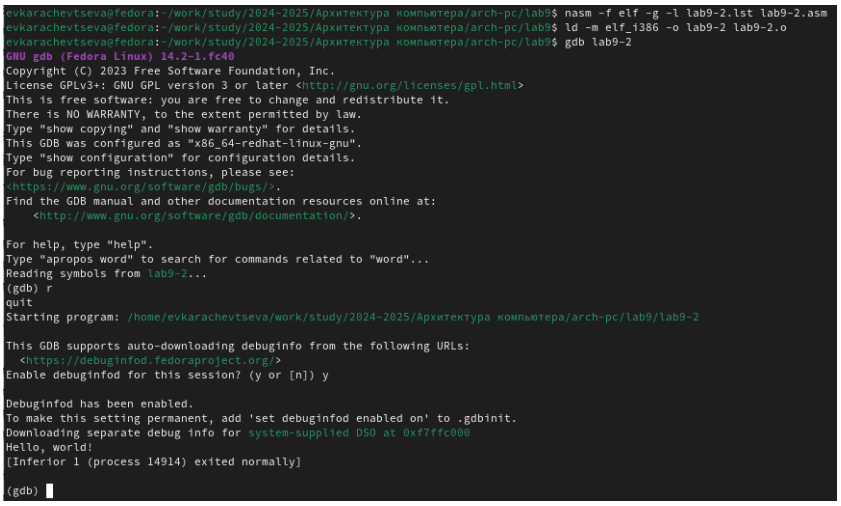

Рис. 2.7: Отладка второго файла

6) Я поставила брекпоинт на метку _start и запустила программу.

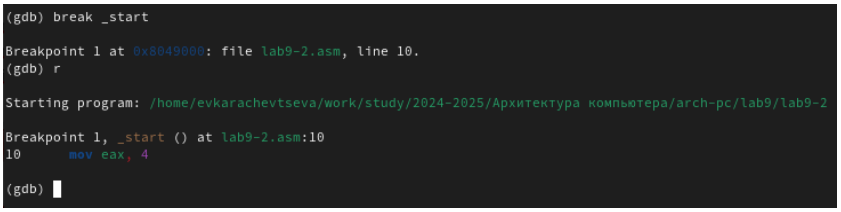

Рис. 2.8: Брекпоинт на метку _start

7)Я просмотрела дисассимплированный код программы начиная с метки.

Рис. 2.9: Дисассимплированный код

8) С помощью команды я переключилась на intel'овское отображение синтаксиса. 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. 2.10: Intel’овское отображение

9) Для удобства я включила режим псевдографики.

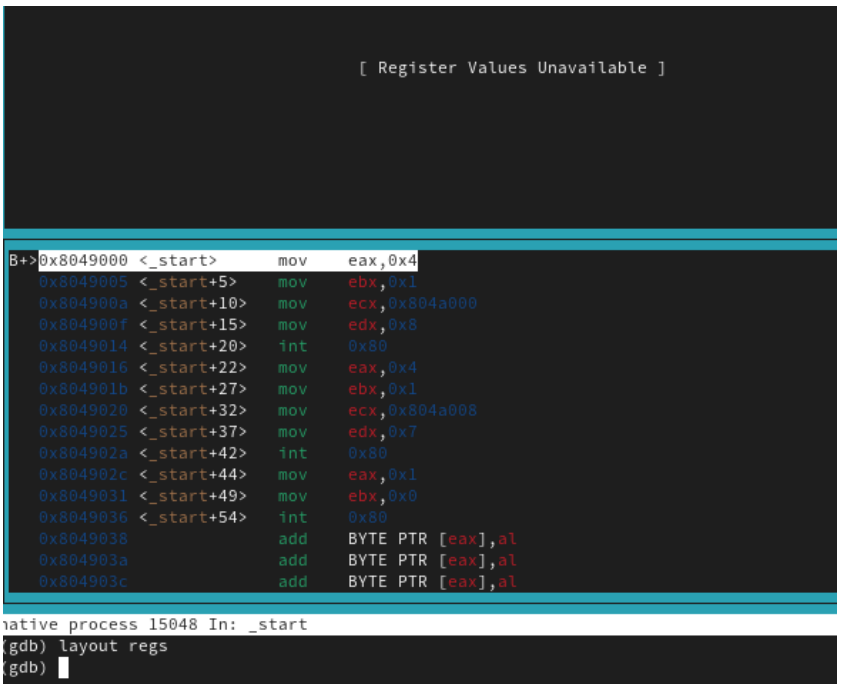

Рис. 2.11: Псевдографика

10) Я посмотрела наличие меток и добавила еще одну метку на предпоследнюю инструкцию.

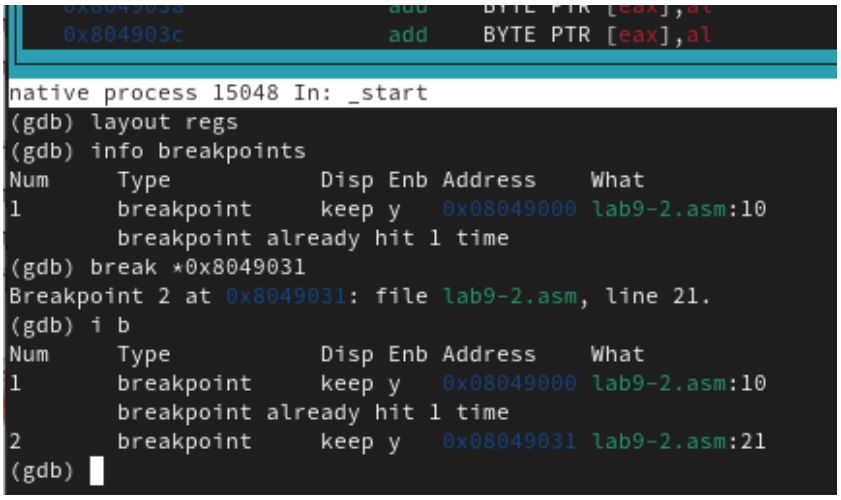

Рис. 2.12: Наличие меток

11) С помощью команды si я посмотрела регистры и изменила их.

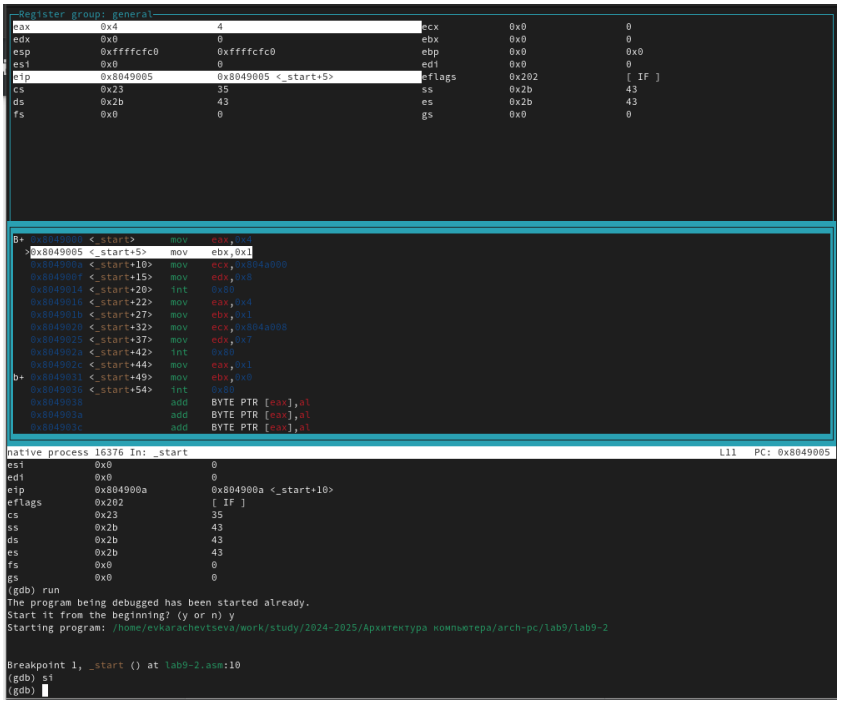

Рис. 2.13: Просмотр регистров

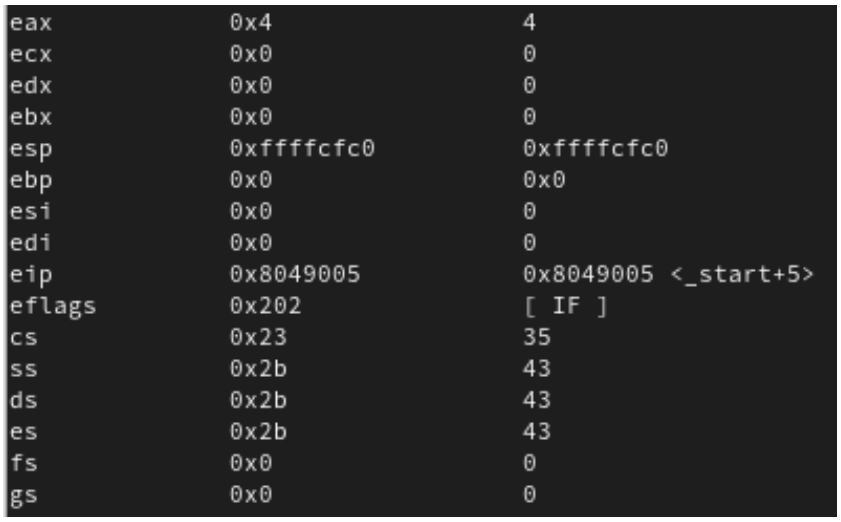

Рис. 2.14: Измененные регистры

12) С помощью команды я посмотрела значение переменной msg1.

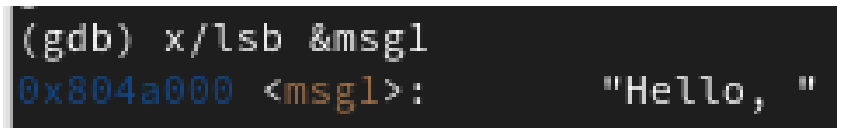

Рис. 2.15: Просмотри значения переменной

13) Следом я посмотрела значение второй переменной msg2.

Рис. 2.16: Значение переменной msg2

14) С помощью команды set я изменила значение переменной msg1.

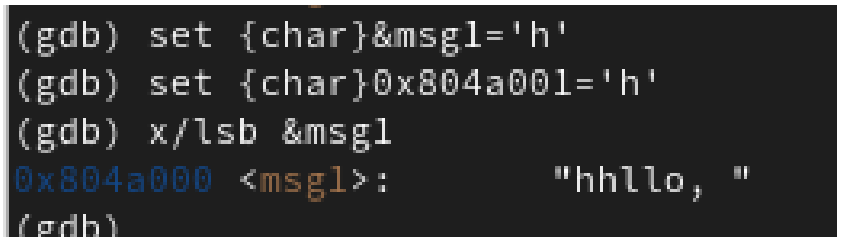

Рис. 2.17: Изменение значения переменной

15)Я изменила переменную msg2.

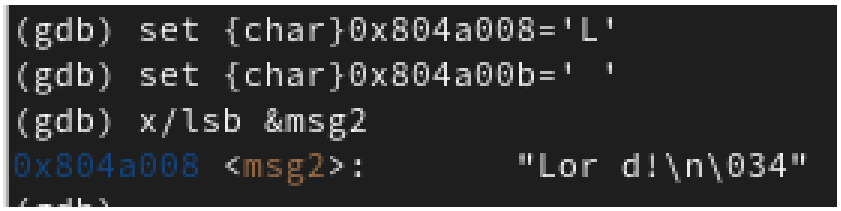

Рис. 2.18: Изменение msg2

16)Я вывела значение регистров ecx и eax.

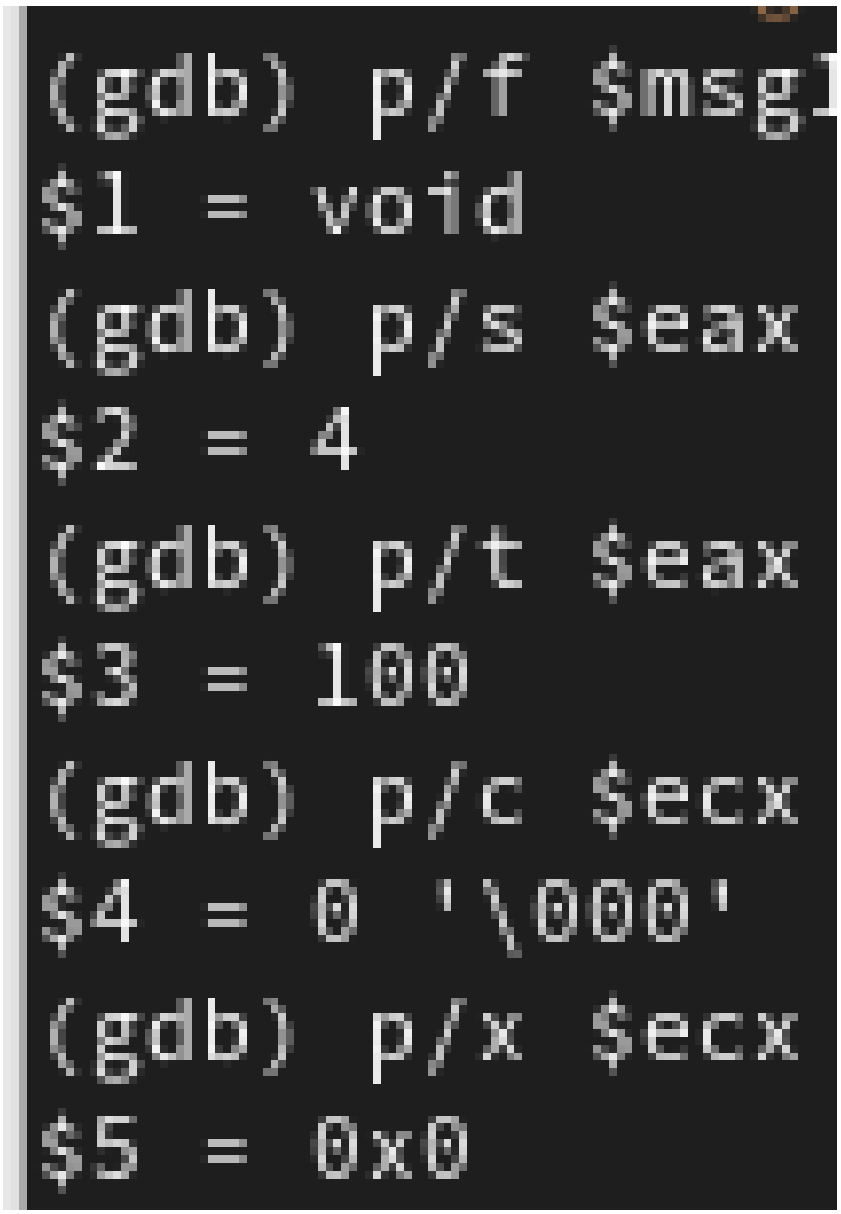

Рис. 2.19: Значение регистров ecx и eax

17) Я изменила 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.

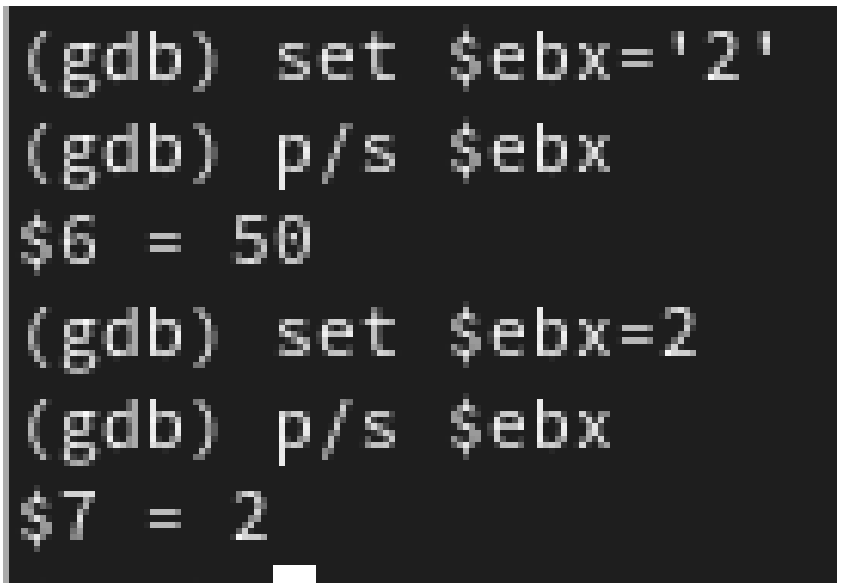

Рис. 2.20: Значение регистров ebx

18) Я завершила работу с файлом и вышла.

Рис. 2.21: Завершение работы с файлов

19) Я скопировала файл lab9-2.asm и переименовала его. Запустила файл в отладчике и указала аргументы.

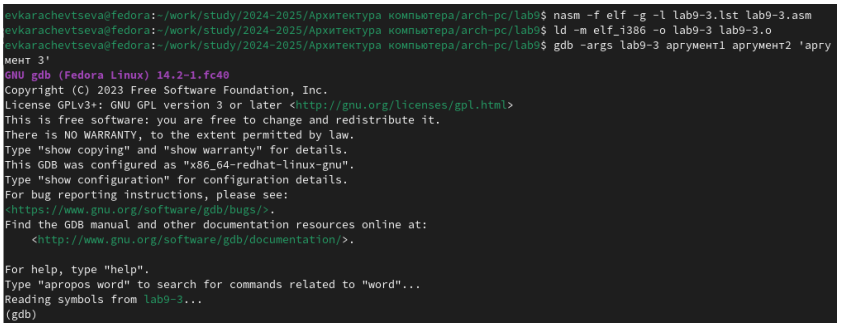

Рис. 2.22: Запуск файла в отладчике

20) Поставила метку на _start и запустила файл.

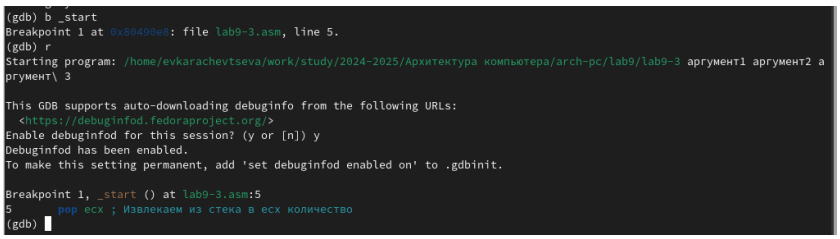

Рис. 2.23: Запуск файла lab9-3 через метку

21)Я проверила адрес вершины стека и убедилась что там хранится 5 элементов.

Рис. 2.24: Адрес вершины стека

22) Я посмотрела 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.

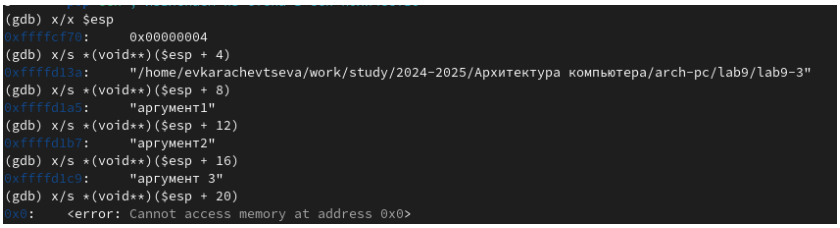

Рис. 2.25: Все позиции стека

3 Самостоятельная работа

1) Я преобразовала программу из лабораторной работы №8 и реализовала вычисления как подпрограмму.

```
%include 'in_out.asm'
SECTION .data
prim DB 'f(x)=(x+1)*7',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

mov eax,prim
call sprintLF
next:
cmp ecx,0
jz _end

mov ebx,7
pop eax
call atoi
add eax,1
mul ebx

add esi,eax

loop next

_end:
mov eax,otv
call sprint
mov eax,esi
call iprintLF
call quit
```

Рис. 3.1: Текст программы

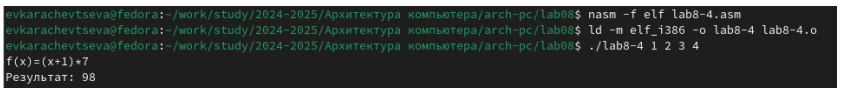

Рис. 3.2: Запуск программы

2) Я переписала программу и попробовала запустить ее чтобы увидеть ошибку. Ошибки не было, программа выдает верный ответ.

```
%include 'in_out.asm'
SECTION .data
prim DB 'f(x)=(x+1)*7',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

mov eax,prim
call sprintLF
next:
cmp ecx,0
jz _end

pop eax
call atoi
call fir
add esi,eax

loop next

_end:
mov eax,otv
call sprint
mov eax,esi
call iprintLF
call quit

fir:
mov ebx,7
add eax,1
mul ebx
ret
```

Рис. 3.3: Текст програмыы

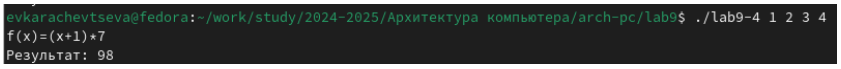

Рис. 3.4: Запуск программы

На всякий случай, я запустила программу в отладчике.

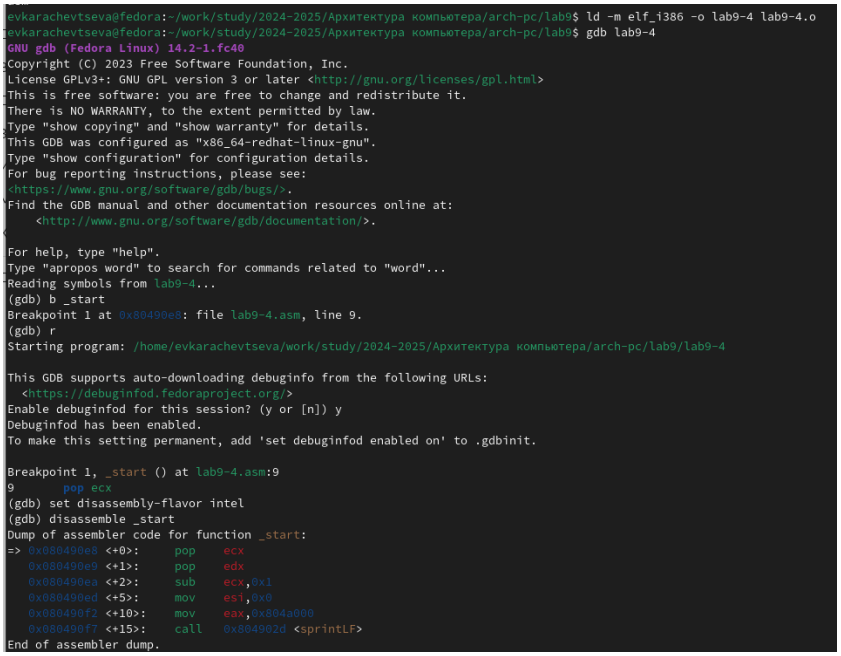

Рис. 3.5: Запуск программы в отладчике

Я открыла регистры и проанализировал их, не увидела никаких ошибок.

Вывод

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