**Отчет по лабораторной работе №9**

### Дисциплины: Архитектура компьютера

## Зоригоо Номун

**Содержание**

1. [Цель работы](#_bookmark0) 5
2. [Задание](#_bookmark1) 6
3. [Выполнение лабораторной работы](#_bookmark2) 7
4. [Выводы](#_bookmark39) 24

[Список литературы](#_bookmark40) 25

**Список иллюстраций**

|  |  |  |
| --- | --- | --- |
| [3.1 Рис 1](#_bookmark3) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 7 |
| [3.2 Рис 2](#_bookmark4) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
| [3.3 Рис 3](#_bookmark5) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 8 |
| [3.4 Рис 4](#_bookmark6) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 9 |
| [3.5 Рис 5](#_bookmark7) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 9 |
| [3.6 Рис 6](#_bookmark8) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
| [3.7 Рис 7](#_bookmark9) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 10 |
| [3.8 Рис 8](#_bookmark10) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
| [3.9 Рис 9](#_bookmark11) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
| [3.10 Рис 10](#_bookmark12) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 11 |
| [3.11 Рис 11](#_bookmark13) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| [3.12 Рис 12](#_bookmark14) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 12 |
| [3.13 Рис 13](#_bookmark15) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 13 |
| [3.14 Рис 14](#_bookmark16) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 13 |
| [3.15 Рис 15](#_bookmark17) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 13 |
| [3.16 Рис 16](#_bookmark18) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 14 |
| [3.17 Рис 17](#_bookmark19) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 14 |
| [3.18 Рис 18](#_bookmark20) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 14 |
| [3.19 Рис 19](#_bookmark21) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 15 |
| [3.20 Рис 20](#_bookmark22) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 15 |
| [3.21 Рис 21](#_bookmark23) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 15 |
| [3.22 Рис 22](#_bookmark24) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 15 |
| [3.23 Рис 23](#_bookmark25) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 16 |
| [3.24 Рис 24](#_bookmark26) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 16 |
| [3.25 Рис 25](#_bookmark27) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 16 |
| [3.26 Рис 26](#_bookmark28) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 16 |
| [3.27 Рис 27](#_bookmark29) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 17 |
| [3.28 Рис 28](#_bookmark30) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 17 |
| [3.29 Рис 29](#_bookmark31) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 18 |
| [3.30 Рис 30](#_bookmark32) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 18 |
| [3.31 Рис 31](#_bookmark33) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 18 |
| [3.32 Рис 32](#_bookmark34) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 19 |
| [3.33 Рис 33](#_bookmark35) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 19 |
| [3.34 Рис 34](#_bookmark36) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |
| [3.35 Рис 35](#_bookmark37) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 20 |
| [3.36 Рис 36](#_bookmark38) . | . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . | 21 |

**Список таблиц**

# Цель работы

Цель лабораторной работы – приобретение навыков написания программ с использованием подпрограмм. Введение в методы отладки с использованием GDB и его основные возможности.

# Задание

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

# Выполнение лабораторной работы

#### Реализация подпрограмм в NASM

Я создаю каталог для lab09, зайду в него и создаю файл lab09-1.asm(рис 1)

A computer screen with blue text

Description automatically generated

Рис. 3.1: Рис 1

Рассмотрим программу вычисления арифметического выражения f(x) = 2x + 7 с помощью подпрограммы \_calcul. В этом примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Я скопирую текст программы ниже и скопирую его в созданный мной файл(рис 2)

A screenshot of a computer

Description automatically generated

Рис. 3.2: Рис 2

Я создаю исполняемый файл и проверю его работу(рис 3)

A computer screen shot of a program

Description automatically generated

Рис. 3.3: Рис 3

Я отредактирую программу так, чтобы она решала функцию f(g(x)), где f(x)=2x+7 и g(x)=3x-1(рис 4)

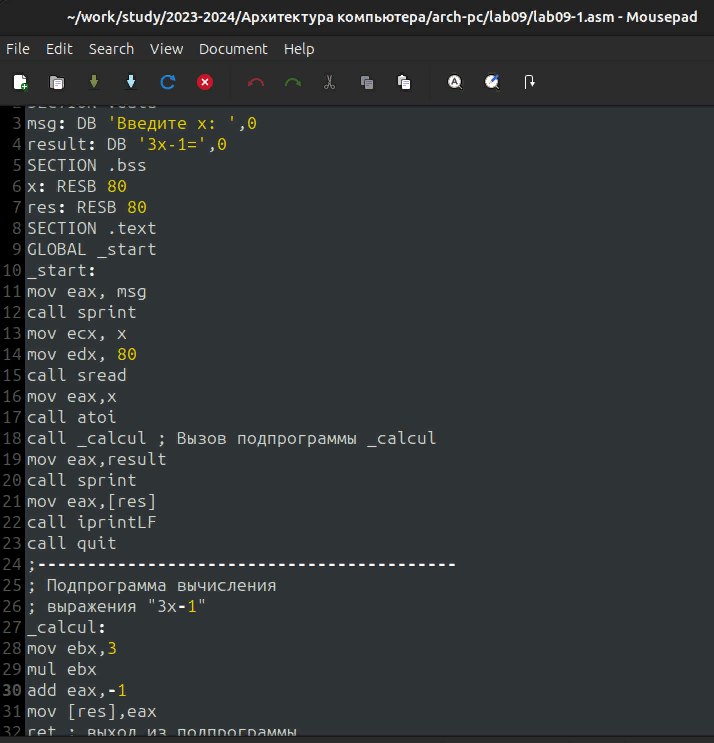


Рис. 3.4: Рис 4

Я создаю исполняемый файл и проверяю его работу(рис 5)

A computer screen with white text

Description automatically generated

Рис. 3.5: Рис 5

#### Отладка программам с помощью GDB

Я создаю новый файл lab09-2.asm и скопирую в него данную программу(рис 6)

A screenshot of a computer

Description automatically generated

Рис. 3.6: Рис 6

Я создаю исполняемый файл и запущу его с помощью отладчика GDB. Чтобы работать с GDB, мне нужно добавить в исполняемый файл отладочную информа- цию; для этого программы необходимо переводить с ключом «-g»(рис 7)

A screenshot of a computer program

Description automatically generated

Рис. 3.7: Рис 7

Я протестирую программу, запустив ее в оболочке GDB с помощью команды запуска(рис 8)

A computer screen with text

Description automatically generated

Рис. 3.8: Рис 8

Для более детального анализа программы я поставлю точку останова на метку

\_start, с которой начинается выполнение любой ассемблерной программы, и запущу ее(рис 9)

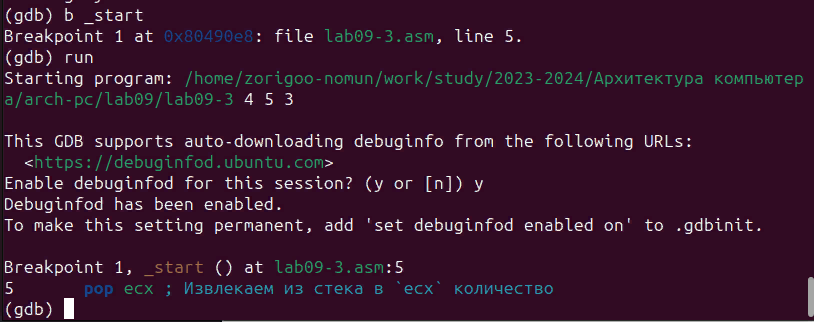


Рис. 3.9: Рис 9

Я буду просматривать дизассемблированный код программы с помощью ко- манды дизассемблирования, начиная с метки \_start(рис 10)

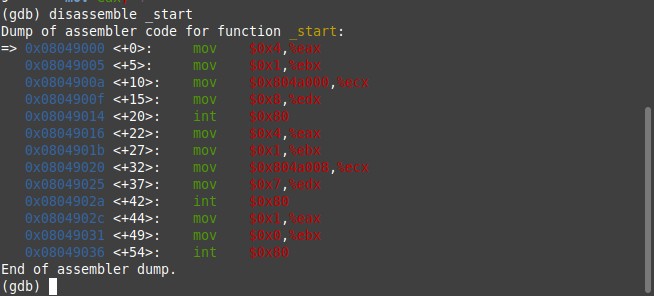


Рис. 3.10: Рис 10

Я переключусь на отображение команд с синтаксисом Intel, введя команду set disassembly-flavor intel(рис 11)

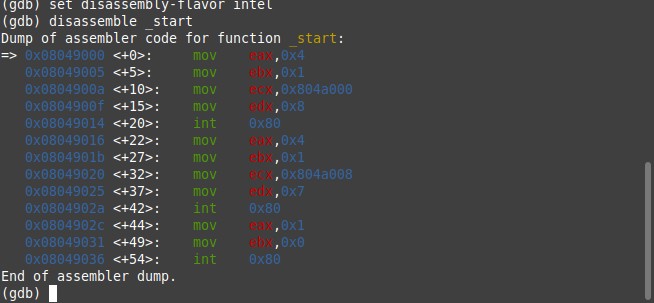


Рис. 3.11: Рис 11

Для более удобного анализа программы включу режим псевдографики(рис 12)

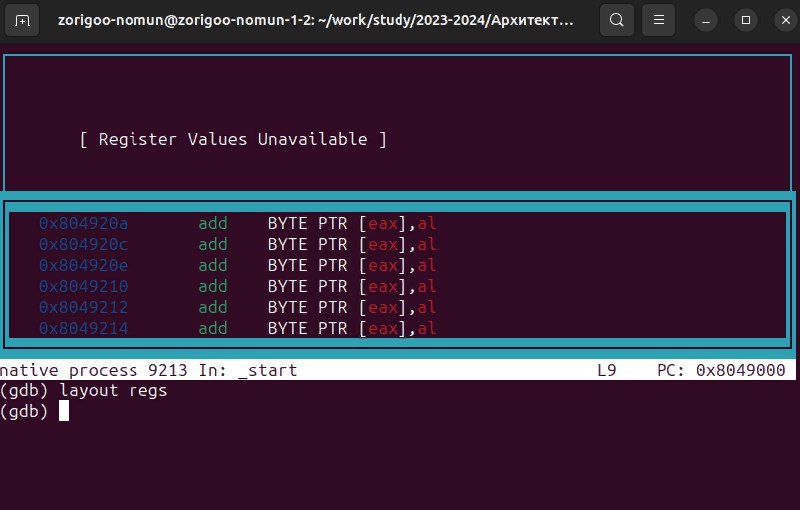


Рис. 3.12: Рис 12

*В Intel все начинается с адреса, затем с источника, а в ATT наоборот*

#### Добавление точек останова

На предыдущих шагах точка останова была установлена по имени метки (\_start).

Я проверю это с помощью команды info Breakpoints (сокращенно i b)(рис 13)

A screenshot of a computer

Description automatically generated

Рис. 3.13: Рис 13

Я поставляю еще одну точку останова по адресу инструкции(рис 14)

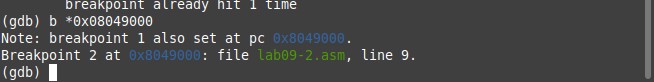


Рис. 3.14: Рис 14

Теперь я посмотрю информацию обо всех установленных точках останова.

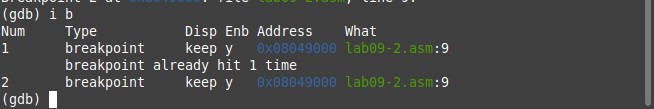


Рис. 3.15: Рис 15

#### Работа с данными программы в GDB

Я выполню 5 инструкций с помощью команды Stepi (или Si)(рис 16)

A screenshot of a computer

Description automatically generated

Рис. 3.16: Рис 16

*Значения eax,ecx,esp и edx изменились*

Содержимое регистров также можно просмотреть с помощью команды info Registers(рис17)A screenshot of a computer

Description automatically generated

Рис. 3.17: Рис 17

С помощью команды x & можно посмотреть содержимое переменной.Я поищу значение переменной msg1 по имени(рис 18)

A screenshot of a computer

Description automatically generated

Рис. 3.18: Рис 18

Теперь я посмотрю на значение переменной msg2 по адресу. Адрес переменной можно определить из дизассемблированной инструкции(рис 19)

A screenshot of a computer

Description automatically generated

Рис. 3.19: Рис 19

Вы можете изменить значение регистра или ячейки памяти с помощью ко- манды set, передав ей имя или адрес регистра в качестве аргумента. Я изменю первый символ переменной msg1(рис 20)

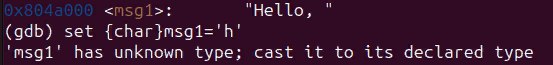


Рис. 3.20: Рис 20

Теперь я заменю символ во второй переменной msg2(рис 21)

A computer screen with white text

Description automatically generated

Рис. 3.21: Рис 21

Я буду использовать команду set для изменения значения регистра ebx(рис 22)

A screenshot of a computer program

Description automatically generated

Рис. 3.22: Рис 22

#### Обработка аргументов командной строки в GDB

Я скопирую файл lab8-2.asm, созданный во время лабораторной работы 8, с помощью программы, которая печатает аргументы командной строки, в файл с именем lab09-3.asm(рис 23)

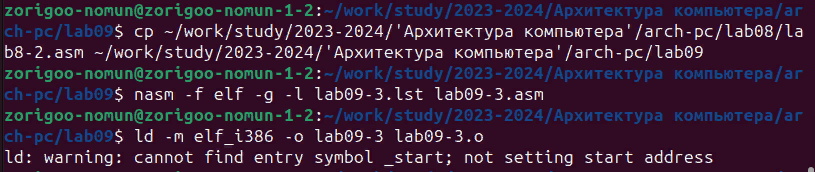


Рис. 3.23: Рис 23

Я создам исполняемый файл и загрузим исполняемый файл в отладчик с аргу- ментами, для загрузки программ с аргументами в gdb я буду использовать ключ

–args(рис 24)

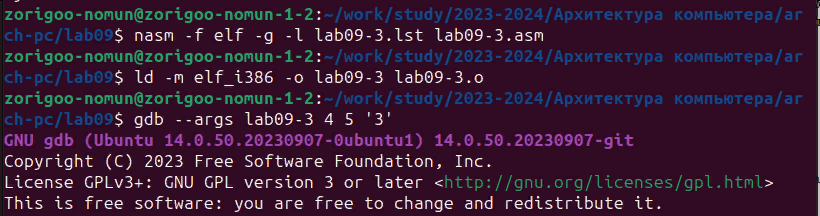


Рис. 3.24: Рис 24

Сначала я установлю точку останова перед первой инструкцией в программе и запущу ее(рис 25)

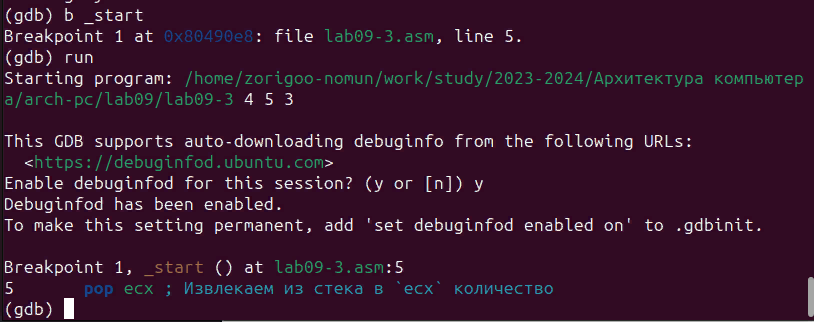


Рис. 3.25: Рис 25

Адрес вершины стека хранится в регистре esp и по этому адресу можно увидеть число, равное количеству аргументов командной строки (включая имя програм- мы)(рис 26)

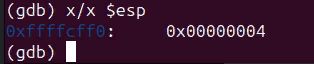


Рис. 3.26: Рис 26

*Как видите, количество аргументов равно 4 — это название программы lab09-3 и сами аргументы: аргумент1, аргумент 2 и ‘аргумент 3’*

Я взгляну на оставшиеся позиции стека — адрес [esp+4] хранит адрес в памяти, где находится имя программы, адрес [esp+8] хранит адрес первого аргумента, адрес [esp+12] ] сохраняет второй аргумент и т. д(рис 27)

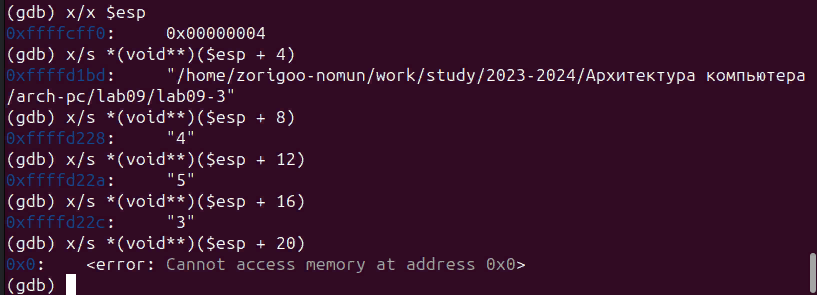


Рис. 3.27: Рис 27

*В 32-битных компьютерах информация хранится именно так: первая память выделяется 4 бита, а вторая — 4х2*

#### Задание для самостоятельной работы

1. Я создам новый файл с именем lab09-4.asm(рис 28)



Рис. 3.28: Рис 28

Используя функцию (f(x) = 7(x+1)) которая была у меня при выполнении lab08- задание 1, я напишу программу, которая вычисляет значение функции как под- программу(рис 29)

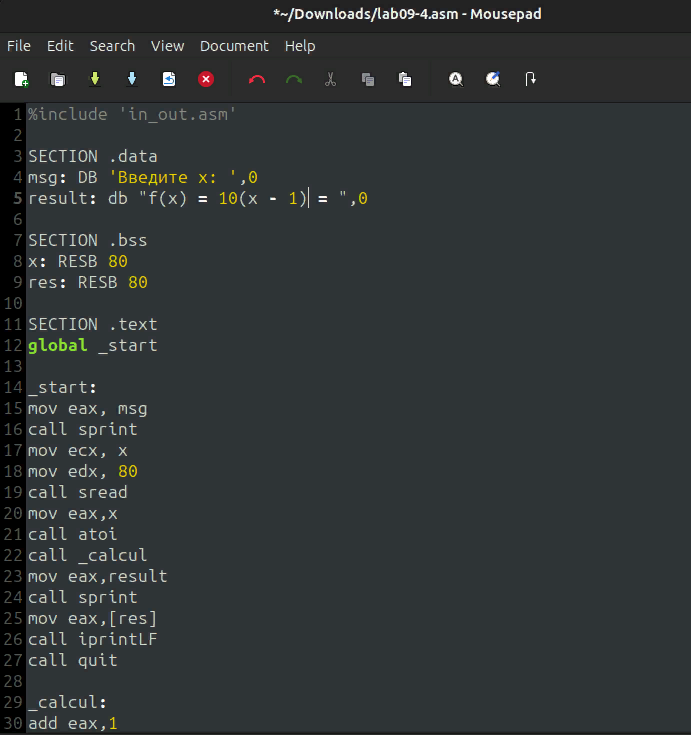


Рис. 3.29: Рис 29

Я создам исполняемый файл и запущу его(рис 30)

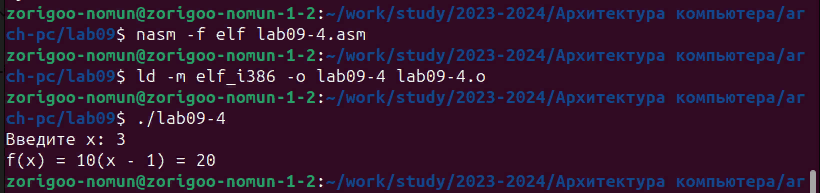


Рис. 3.30: Рис 30

1. Я создам новый файл с именем lab09-5.asm(рис 31)



Рис. 3.31: Рис 31

Я скопирую данную программу, которая вычисляет значение (3+2) 4+5(рис 32)

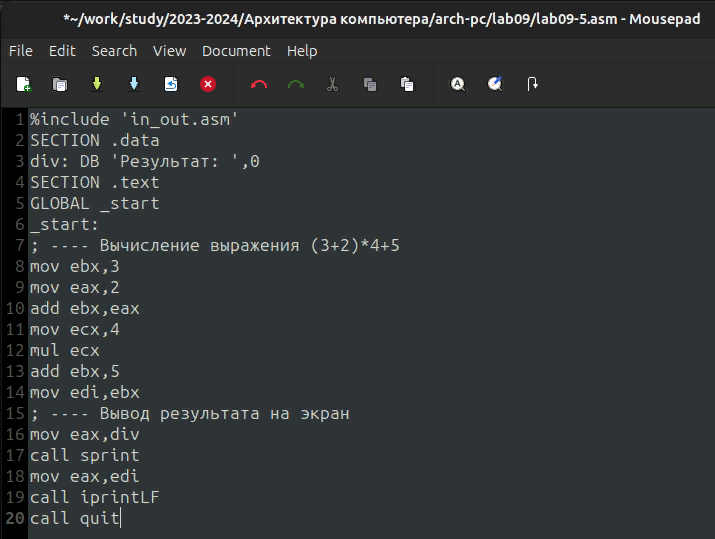


Рис. 3.32: Рис 32

Я создам исполняемый файл и запущу его с помощью GDB(рис 33)

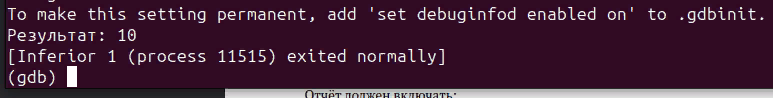


Рис. 3.33: Рис 33

Теперь я проверю, где ошибка: первый шаг нашей программы — сложить ebx, равный 3, и eax, равный 2, что делает ebx=5, затем она перемещает 4 в ecx и по умолчанию умножает ecx на eax. что дает eax 8. В-третьих, он добавит ebx к ebx, в результате чего получится 10(рис 34).

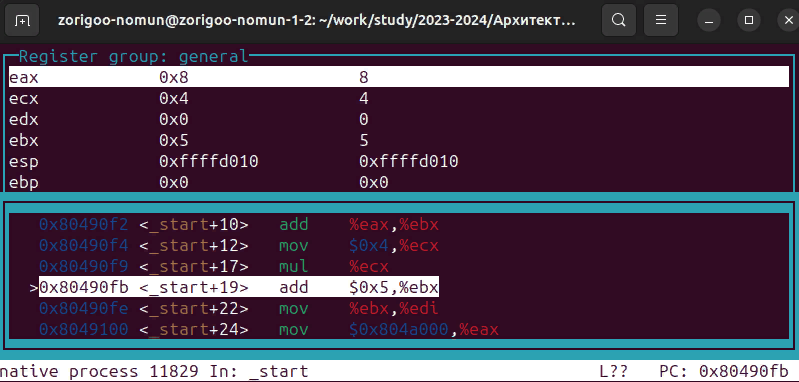


Рис. 3.34: Рис 34

Я изменю программу так, чтобы она давала мне правильный ответ(рис 35)

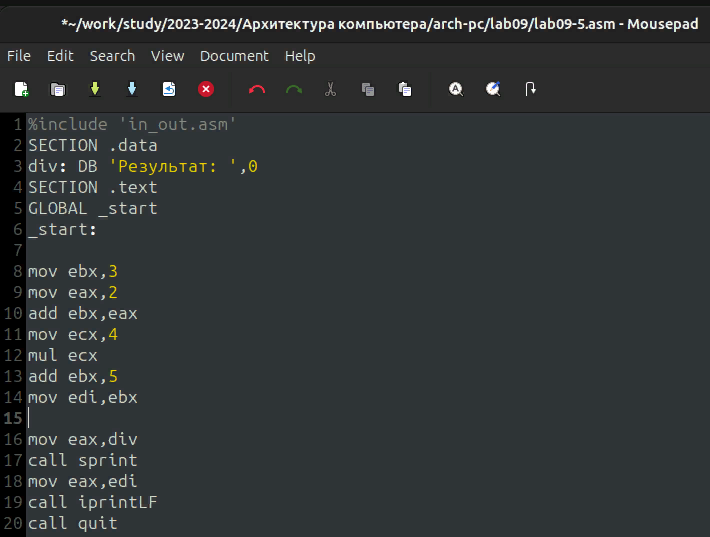


Рис. 3.35: Рис 35

Теперь я создам исполняемый файл и запущу его(рис 36)

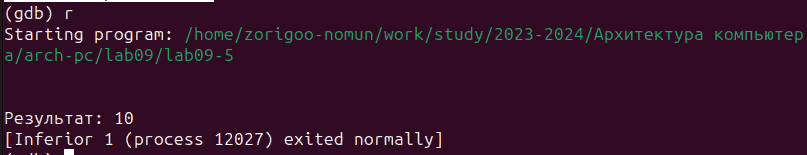


Рис. 3.36: Рис 36

*Текстовая программа для самостоятельной работы 1*

%include 'in\_out.asm'

SECTION .data

msg: DB 'Введите x: ',0

result: db "f(x) = 10(x - 1) = ",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 mov eax,result

call sprint mov eax,[res] call iprintLF call quit

\_calcul:

add eax,-1

mov ebx,10

mul ebx

mov [res],eax ret

*Текстовая программа для самостоятельной работы 2*

%include 'in\_out.asm'

SECTION .data

div: DB 'Результат: ',0

SECTION .text GLOBAL \_start

\_start:

mov ebx,3 mov eax,2 add eax,ebx mov ecx,4 mul ecx add eax,5 mov edi,eax

mov eax,div call sprint mov eax,edi call iprintLF

call quit

# Выводы

В ходе лабораторной работы я приобрел навыки написания программ с исполь- зованием подпрограмм. А также введение в методы отладки с использованием GDB и его основные возможности.

# Список литературы

Архитектура ЭВМ