Отчет по лабораторной работе №9

Дисциплина: Архитектура компьютера

Краснова Камилла Геннадьевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Цель данной лабораторной 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 программ с помощью GDB
3. Самостоятельное выполнение заданий по материалам лабораторной работы

3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

• обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки.

Можно выделить следующие типы ошибок:

• синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают пре- рывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить доволь- но трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

Наиболее часто применяют следующие методы отладки: • создание точек кон-

троля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения);

• использование специальных программ-отладчиков. Отладчики позволяют управлять ходом выполнения программы, контролировать и изменять данные. Это помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам.

Пошаговое выполнение — это выполнение программы с остановкой после каждой строчки, чтобы программист мог проверить значения переменных и выполнить другие действия. Точки останова — это специально отмеченные места в программе, в которых программа- отладчик приостанавливает выполнение программы и ждёт команд. Наиболее популярные виды точек останова: • Breakpoint — точка останова (остановка происходит, когда выполнение доходит до определённой строки, адреса или процедуры, отмеченной программистом);

• Watchpoint — точка просмотра (выполнение программы приостанавливается, если программа обратилась к определённой переменной: либо считала её значение, либо изменила его).

Точки останова устанавливаются в отладчике на время сеанса работы с кодом програм- мы, т.е. они сохраняются до выхода из программы-отладчика или до смены отлаживаемой программы.

4 Выполнение лабораторной работы

4.1 Реализация подпрограмм в NASM

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №9. Перехожу в созданный каталог с помощью утилиты cd и с помощью утилиты touch создаю файл lab09-1.asm.(рис. 4.1).

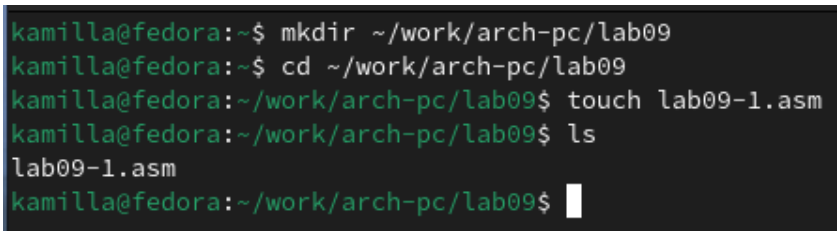


Рис. 4.1: Создание директории

Открываю созданный файл lab09-1.asm, вставляю в него программу (рис. 4.2).

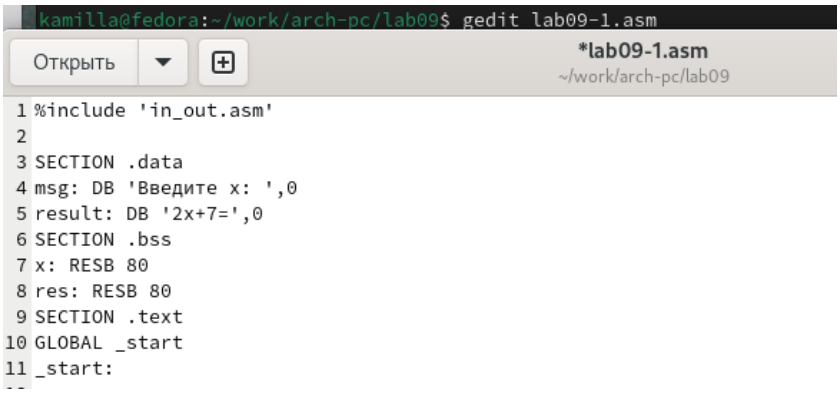


Рис. 4.2: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 4.3).

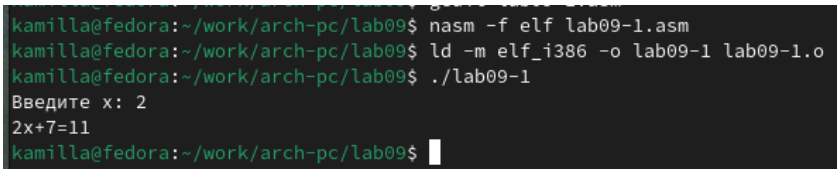


Рис. 4.3: Запуск исполняемого файла

Изменяю текст программы , добавляя подпрограмму _subcalcul в подпрограмму _calcul(рис. 4.4).

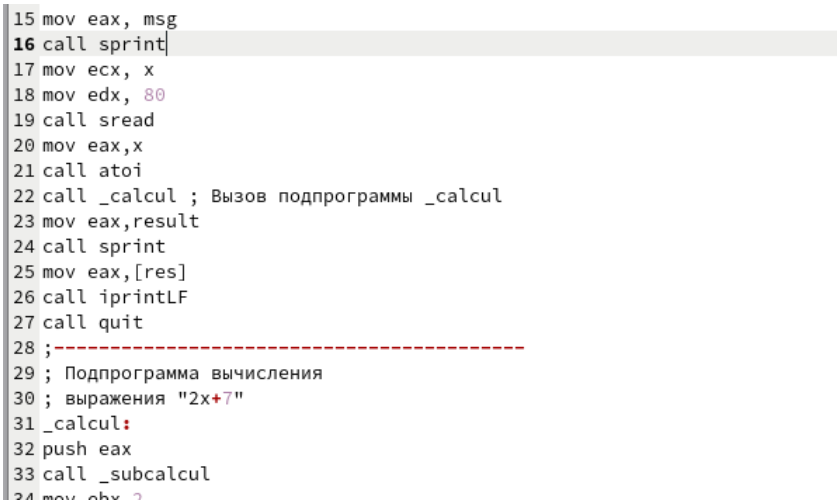


Рис. 4.4: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.5).

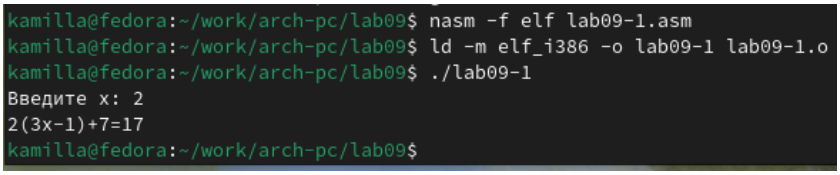


Рис. 4.5: Запуск исполняемого файла

4.2 Отладка программам с помощью GDB

Создаю файл lab09-2.asm (рис. 4.6).

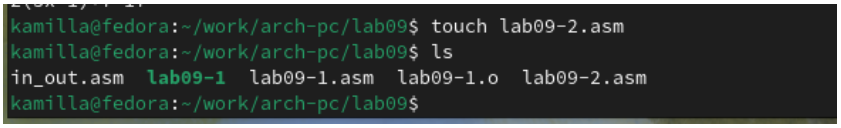


Рис. 4.6: Создание файла

Открываю созданный файл lab09-2.asm, вставляю в него программу (рис. 4.7).

Рис. 4.7: Редактирование файла

Получаю исполняемый файл. Трансляцию программ провожу с ключом ‘-g’ (рис. 4.8).

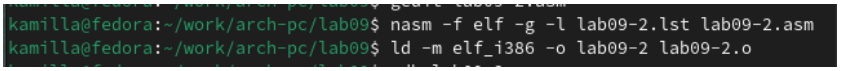


Рис. 4.8: Получение исполняемого файла

Загружаю исполняемый файл в отладчик gdb (рис. 4.9).

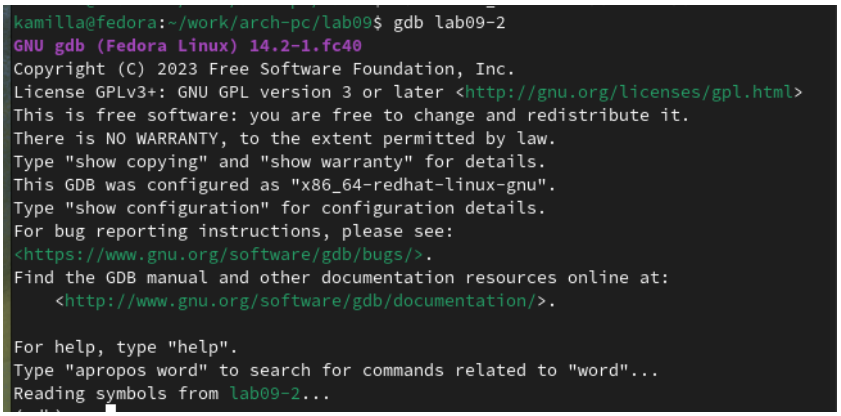


Рис. 4.9: Загрузка файла

Проверяю работу программы, запустив ее в оболочке GDB (рис. 4.10).

Рис. 4.10: Проверка работы программы

Устанавливаю брекпоинт на метку _start и запускаю программу (рис. 4.11).

Рис. 4.11: Запуск отладчика с брекпоинтом

Далее смотрю дисассимилированный код программы (рис. 4.12).

Рис. 4.12: Дисассимилирование программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду set disassembly-flavor intel (рис. 4.13).

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:     mov    eax,0x4
   0x08049005 <+5>:     mov    ebx,0x1
   0x0804900a <+10>:    mov    ecx,0x804a000
   0x0804900f <+15>:    mov    edx,0x8
   0x08049014 <+20>:    int    0x80
   0x08049016 <+22>:    mov    eax,0x4
   0x0804901b <+27>:    mov    ebx,0x1
   0x08049020 <+32>:    mov    ecx,0x804a008
   0x08049025 <+37>:    mov    edx,0x7
   0x0804902a <+42>:    int    0x80
   0x0804902c <+44>:    mov    eax,0x1
   0x08049031 <+49>:    mov    ebx,0x0
   0x08049036 <+54>:    int    0x80
End of assembler dump.
(gdb)
```

Рис. 4.13: Переключение отображения команд

Включаю режим псевдографики (рис. 4.14) и (рис. 4.15).

```
B+>0x8049000 <_start>      mov    eax,0x4
   0x8049005 <_start+5>    mov    ebx,0x1
   0x804900a <_start+10>   mov    ecx,0x804a000
   0x804900f <_start+15>   mov    edx,0x8
   0x8049014 <_start+20>   int    0x80
   0x8049016 <_start+22>   mov    eax,0x4
   0x804901b <_start+27>   mov    ebx,0x1
   0x8049020 <_start+32>   mov    ecx,0x804a008
   0x8049025 <_start+37>   mov    edx,0x7
   0x804902a <_start+42>   int    0x80
   0x804902c <_start+44>   mov    eax,0x1
   0x8049031 <_start+49>   mov    ebx,0x0
   0x8049036 <_start+54>   int    0x80
native process 9572 In: _start                          L9    PC: 0x8049000
(gdb)
```

Рис. 4.14: Режим псевдографики

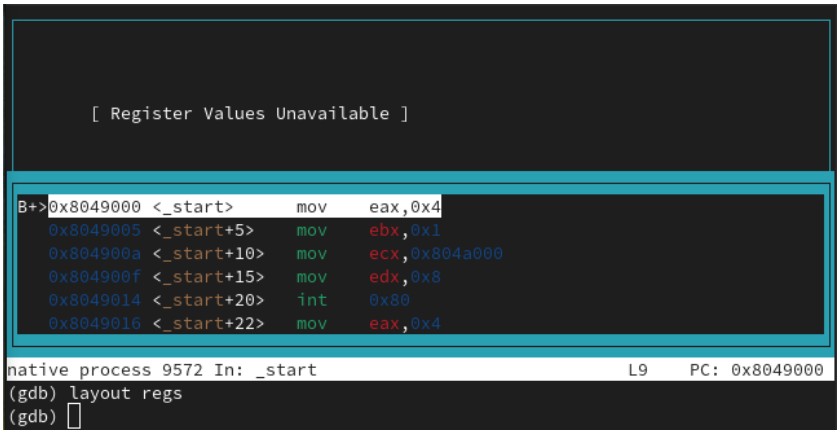


Рис. 4.15: Режим псевдографики

Различия между синтаксисом ATT и Intel заключаются в порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом $; Intel - Размер операндов неявно определяется контекстом, как ax, eax, непосредственные операнды пишутся напрямую), именах регистров(ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

4.3 Добавление точек останова

Проверяю установку точки останова по имени метки (рис. 4.16).

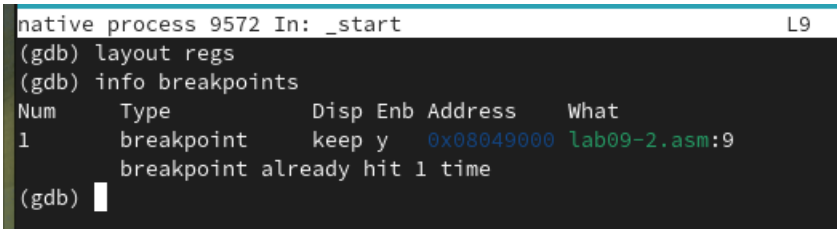


Рис. 4.16: Проверка установки точки

Устанавливаю точку останова по адресу инструкции (рис. 4.17).

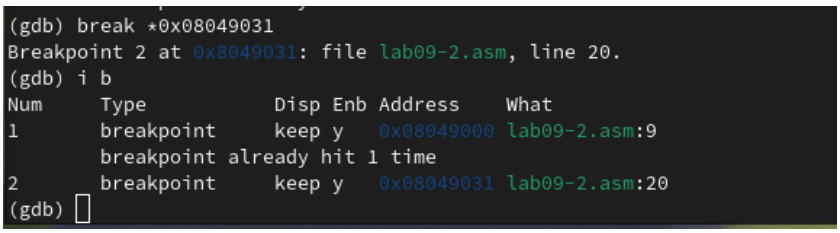


Рис. 4.17: Добавление точки останова

4.4 Работа с данными программы в GDB

Просматриваю содержимое регистров командой info registers (рис. 4.18).

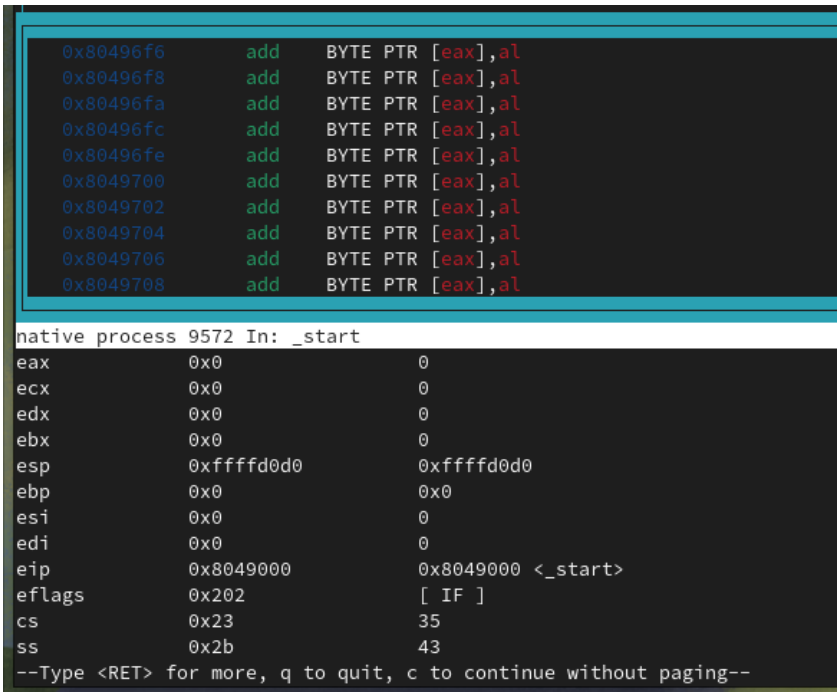


Рис. 4.18: Просмотр содержимого

Смотрю содержимое переменных по имени и по адресу (рис. 4.19).

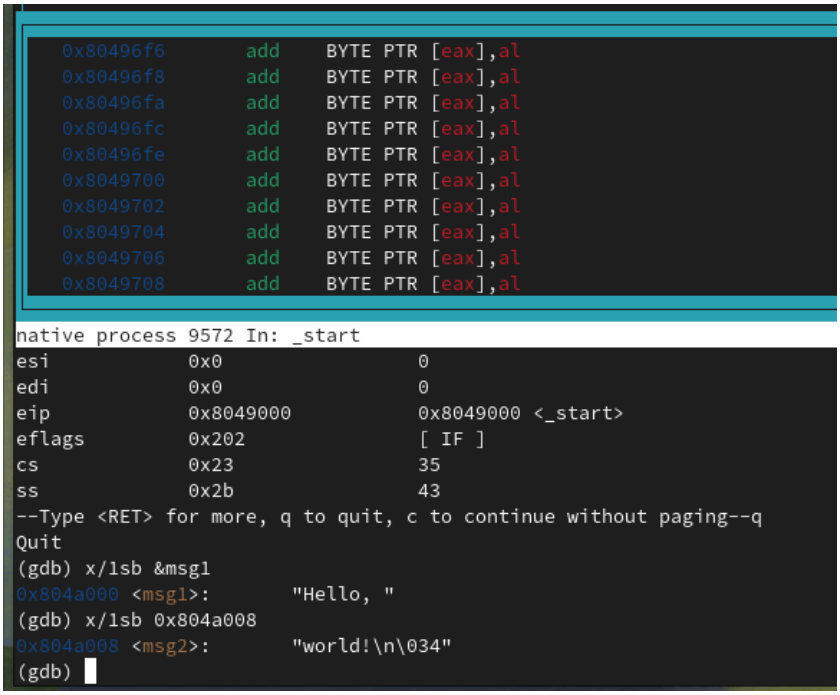
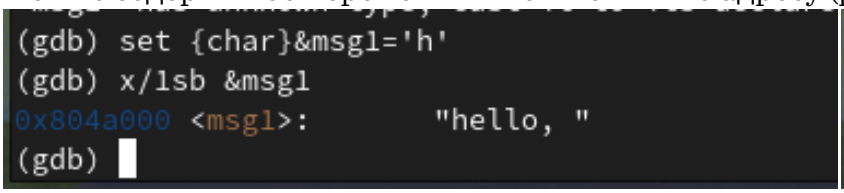
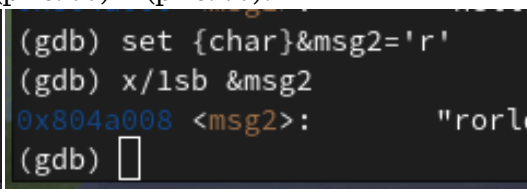


Рис. 4.19: Просмотр переменных

Меняю содержимое переменных по имени и по адресу (рис. **??**) и (рис. **??**).





Вывожу в различных форматах значение регистра edx (рис. 4.20).

Рис. 4.20: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. 4.21).

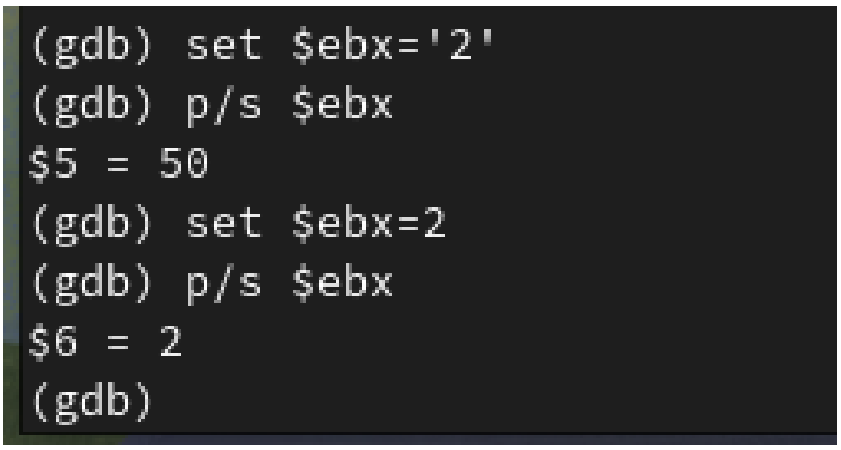


Рис. 4.21: Примеры использования команды set

4.5 Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог и и создаю исполняемый файл с файлом листинга и отладки (рис. 4.22).

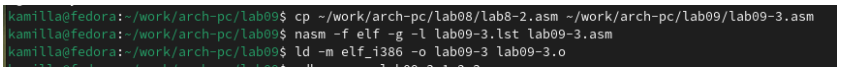


Рис. 4.22: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов (рис. 4.23).

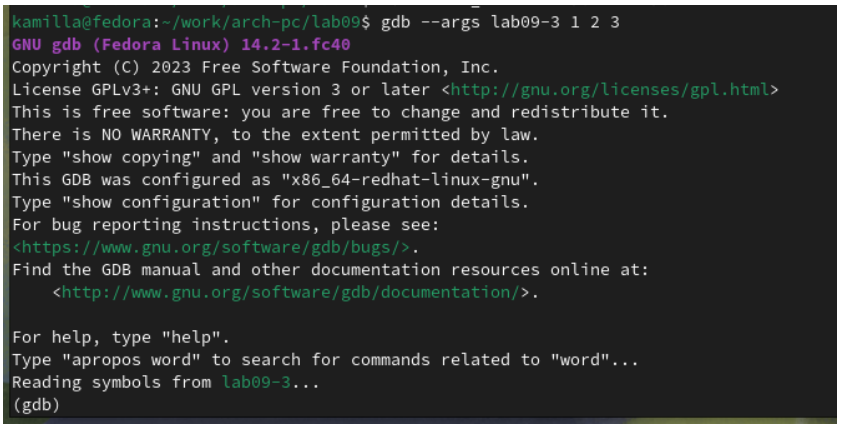


Рис. 4.23: Запуск в режиме отладки

Указываю брейкпопнт и запускаю отладку. (рис. **??**).

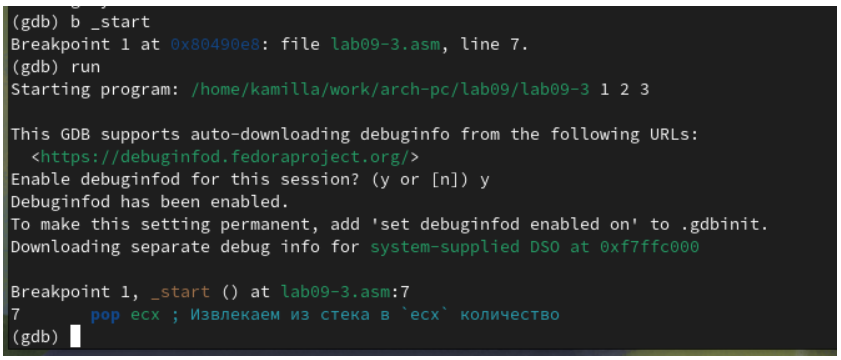


Рис. 4.24: Запуск отладки

Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а указатель void занимает как раз 4 байта. Ошибка при аргументе +20 означает, что аргументы на вход программы закончились. (рис. 4.25).

Рис. 4.25: Проверка работы стека

4.6 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 4.26).

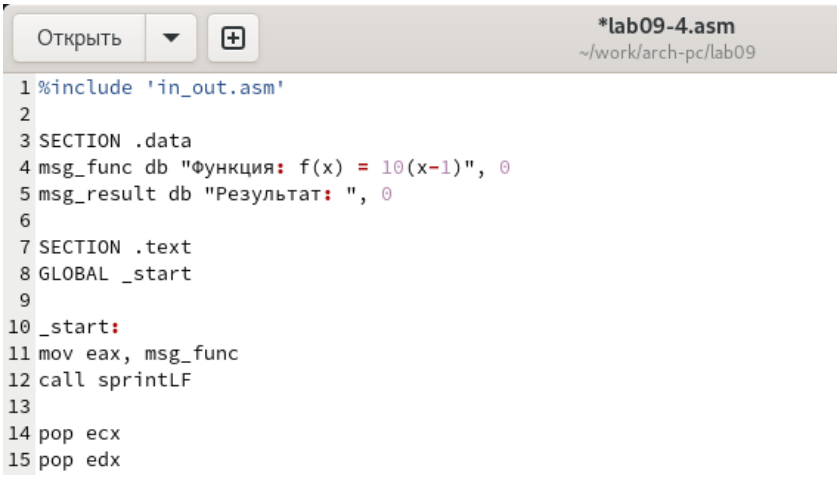


Рис. 4.26: Измененная программа предыдущей лабораторной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 10x - 4", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintLF

pop ecx
pop edx
sub ecx, 1
mov esi, 0
```

```
next:
cmp ecx, 0h
jz _end
pop eax
call atoi

call _calculate_fx

add esi, eax
loop next

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

_calculate_fx:
mov ebx, 10
mul ebx
sub eax, 4
```

2. Запускаю программу в режике отладичка и просматриваю изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому программа неверно подсчитывает функцию (рис. 4.27).

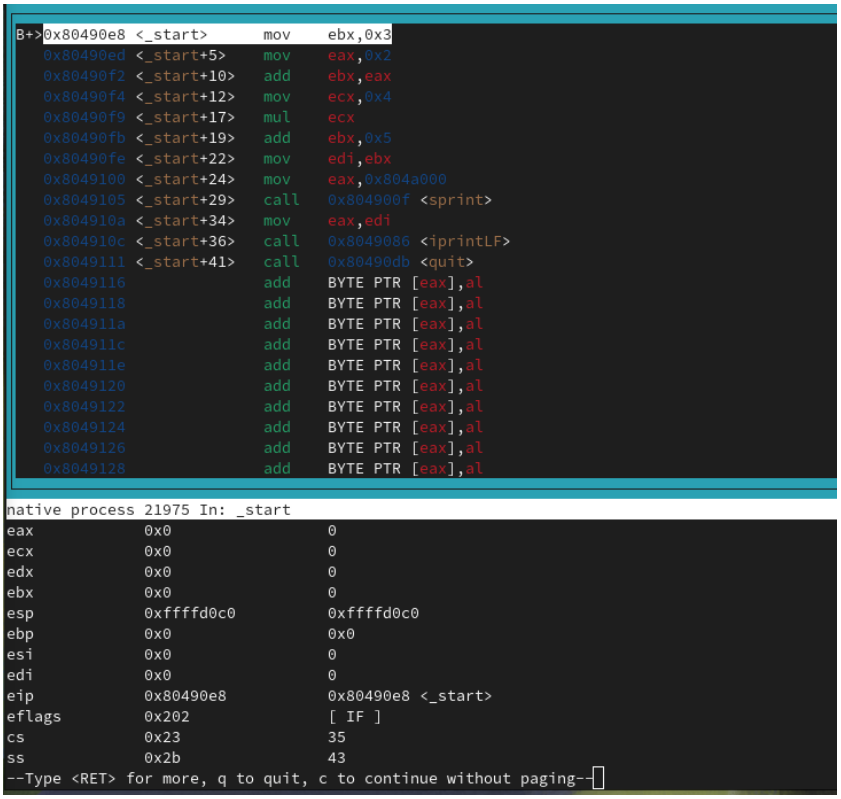


Рис. 4.27: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. 4.28).

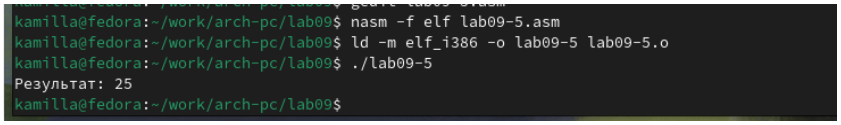


Рис. 4.28: Проверка корректировок в программме

Код измененной программы:

```
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ', 0

SECTION .text
```

```
GLOBAL _start
_start:

mov ebx, 3
mov eax, 2
add ebx, eax
mov 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
```

5 Выводы

При выполнении данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм, а так же познакомилась с методами отладки при поомщи GDB и его основными возможностями.

6 Список литературы

1. Лабораторная работа №9