Отчёт по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB

Кулаженкова Яна Сергеевна

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 Задание

Освоить принципы организации подпрограмм в ассемблере, изучить инструменты отладки с использованием GDB, включая установку точек останова, пошаговую отладку и обработку аргументов командной строки.

3 Теоретическое введение

Отладка – это систематический процесс поиска и исправления ошибок в программном коде, включающий этапы обнаружения, локализации, диагностики и устранения дефекта. Для её проведения используются методы контрольных точек и специализированные программы-отладчики, такие как GDB (GNU Debugger). GDB предоставляет ключевые возможности для низкоуровневого анализа: управление выполнением программы (запуск, остановка), установку точек останова командой `break`, пошаговое прохождение кода (`stepi`) и исследование состояния программы через регистры (`info registers`) и память (`x`).

Важной особенностью GDB при работе с машинным кодом является функция дизассемблирования, позволяющая отображать выполняемые инструкции в синтаксисе ATT или Intel. Это необходимо для детального понимания логики работы программы на уровне процессора. Анализ особенно важен при использовании подпрограмм, механизм вызова (`call`) и возврата (`ret`) которых напрямую влияет на состояние стека и потока управления.

Таким образом, владение инструментарием GDB, включая установку точек останова, пошаговое выполнение и анализ данных, является основополагающим навыком для диагностики и устранения ошибок в программном обеспечении, в том числе, написанном на ассемблере.

4 Выполнение лабораторной работы

4.1 Реализация подпрограмм в NASM

Перед началом выполнения лабораторной работы создадим каталог для файлов (рис. 4.1).

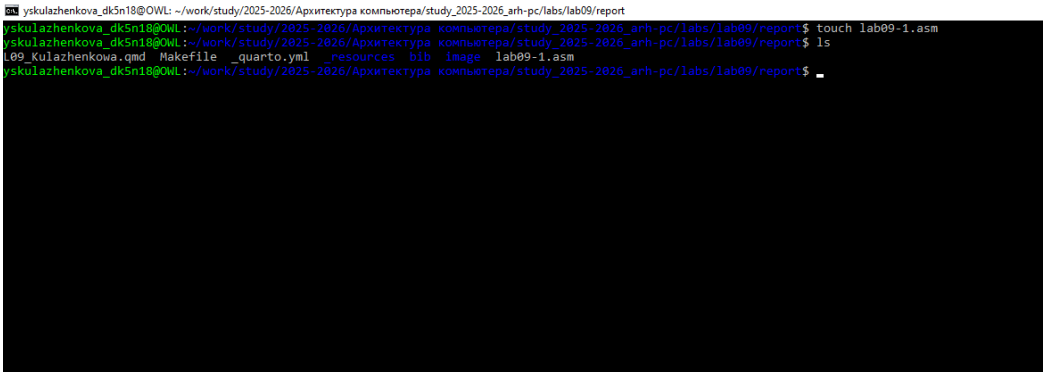

Рисунок 4.1: Переход в каталог лабораторной работы №9

В качестве примера рассмотрим программу вычисления арифметического выражения с помощью подпрограммы _calcul. Для этого в файл lab09-1.asm введем текст программы из листинга 9.1 (рис. 4.2).

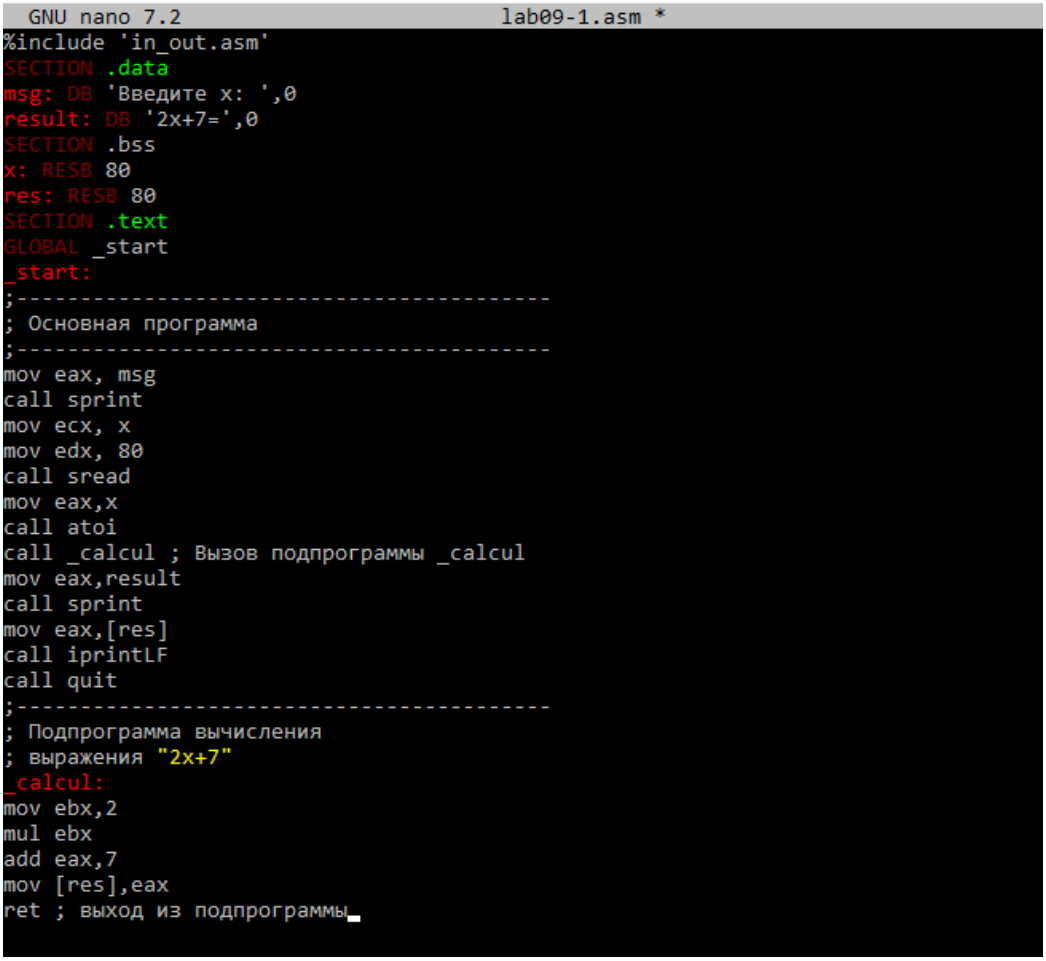

Рисунок 4.2: Содержимое файла lab09-1

Создадим исполняемый файл и проверим его работу (рис. 4.3).

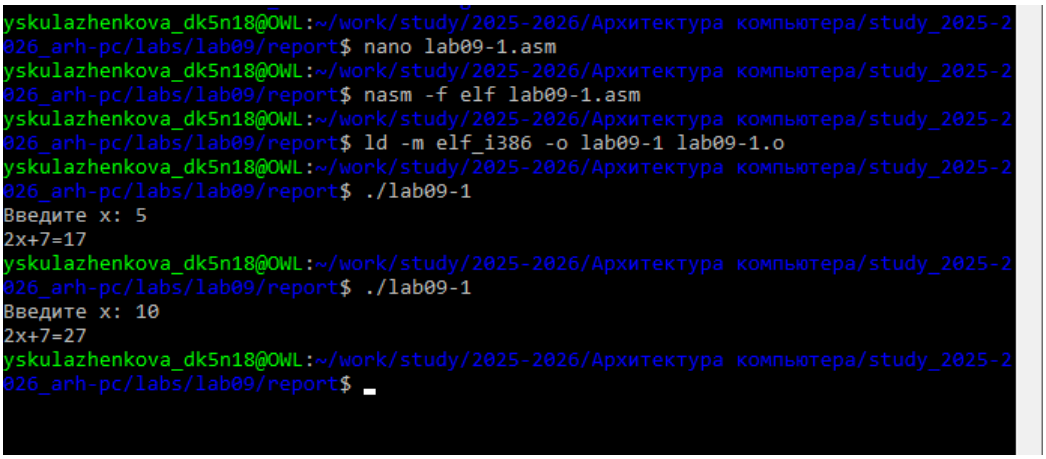

Рисунок 4.3: Продемонстрируем работу файла lab09-1

Рассмотрим еще один пример. Измените текст программы (рис. 4.4).

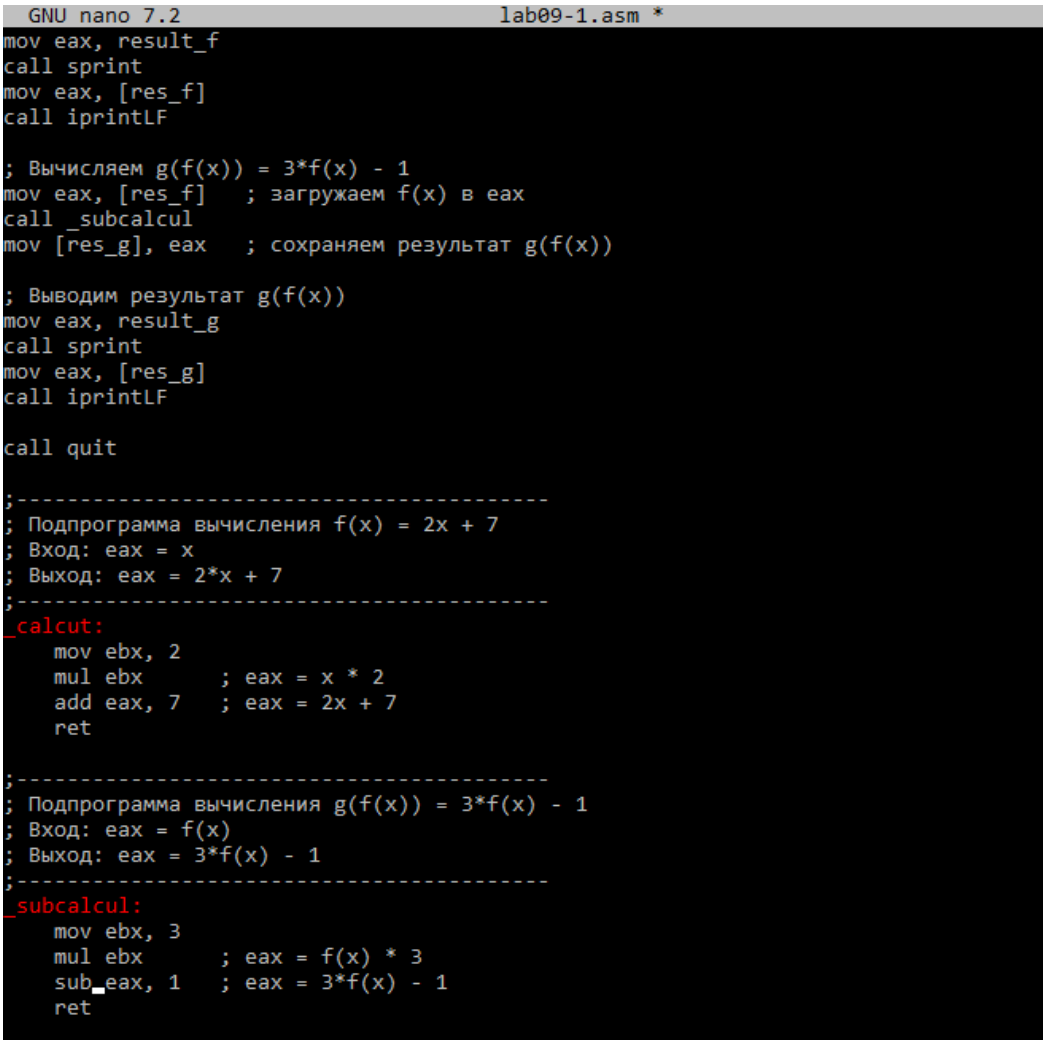

Рисунок 4.4: Изменим код

Продемонстрируем работу кода программы (рис. 4.5).

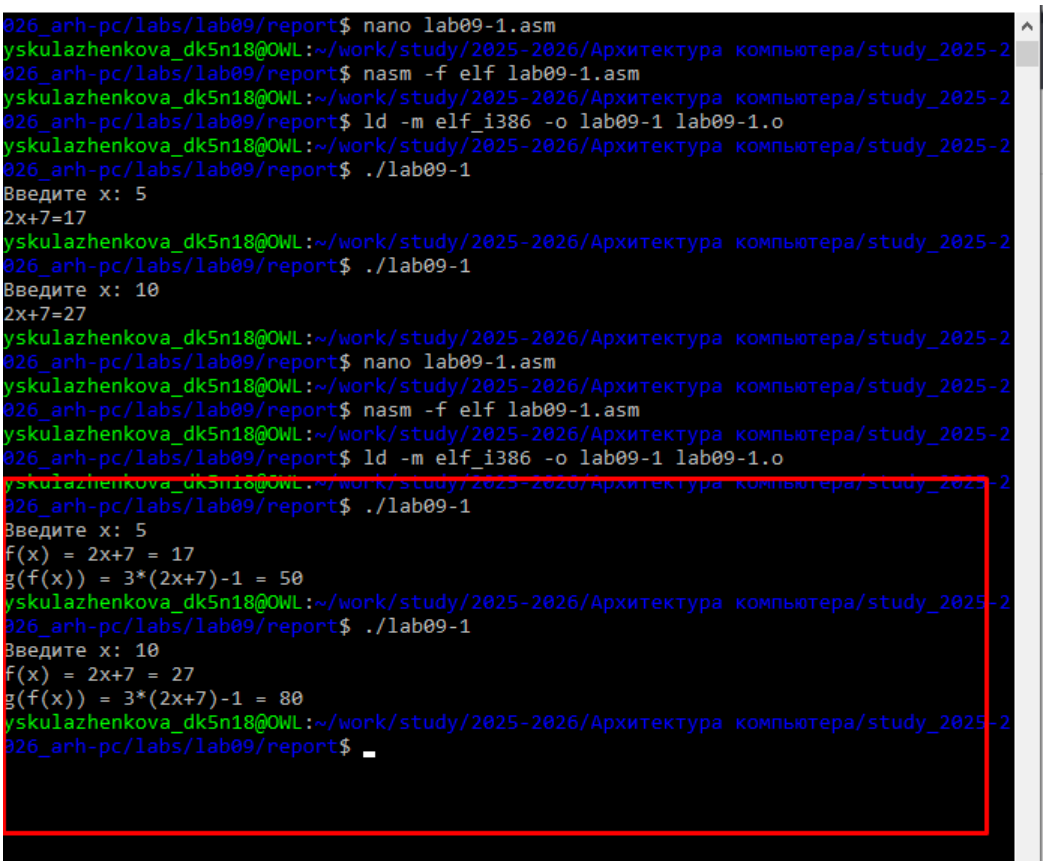

Рисунок 4.5: Работа измененного кода

4.2 Отладка программам с помощью GDB

Далее создадим файл lab09-2.asm с текстом программы для печати сообщения Hello world (рис. 4.6).

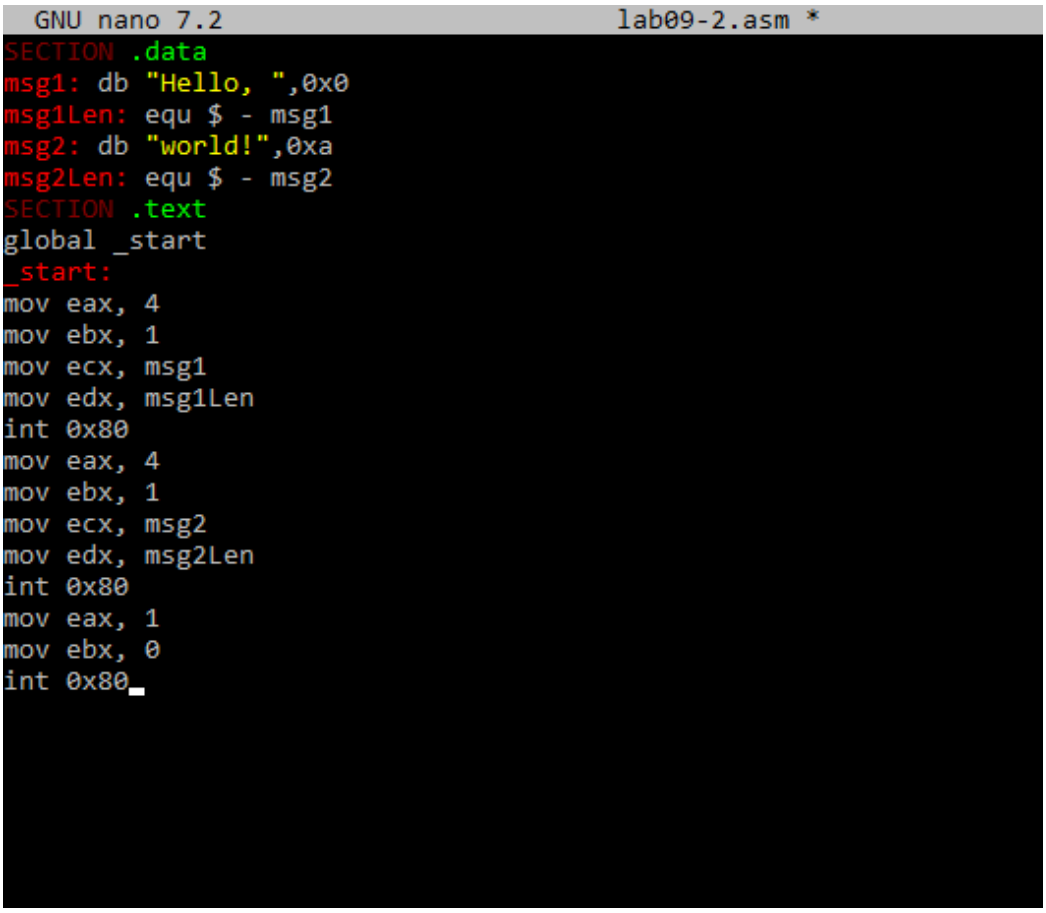

Рисунок 4.6: Программа для печати сообщения Hello world!

Получим исполняемый файл и загрузим его в отладчик gdb (рис. 4.7).

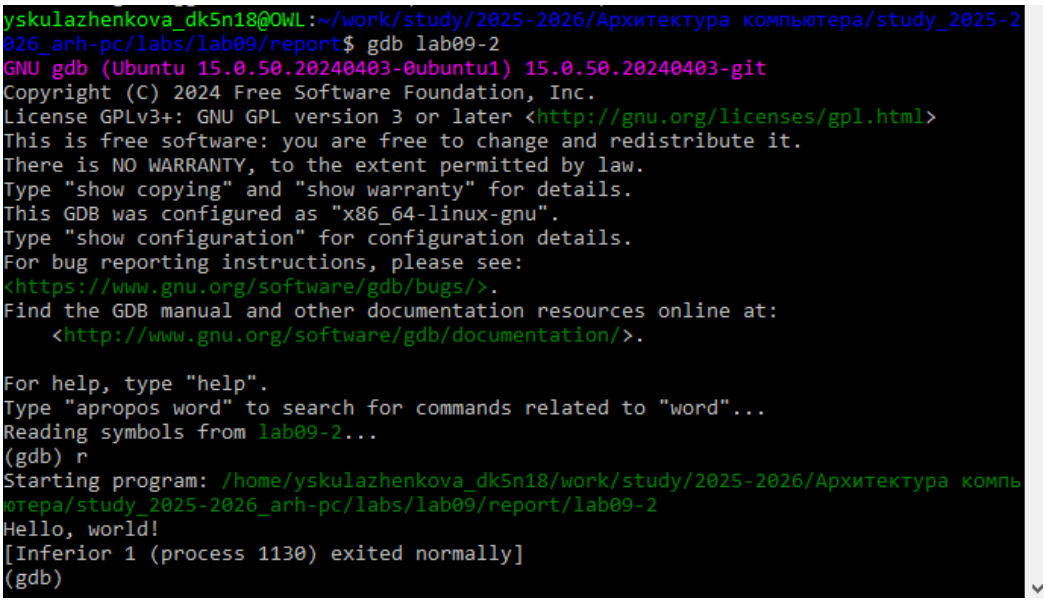

Рисунок 4.7: Загрузка исполняемого файла в отладчик

Теперь установим брейкпоинт на метку _start (рис. 4.8).

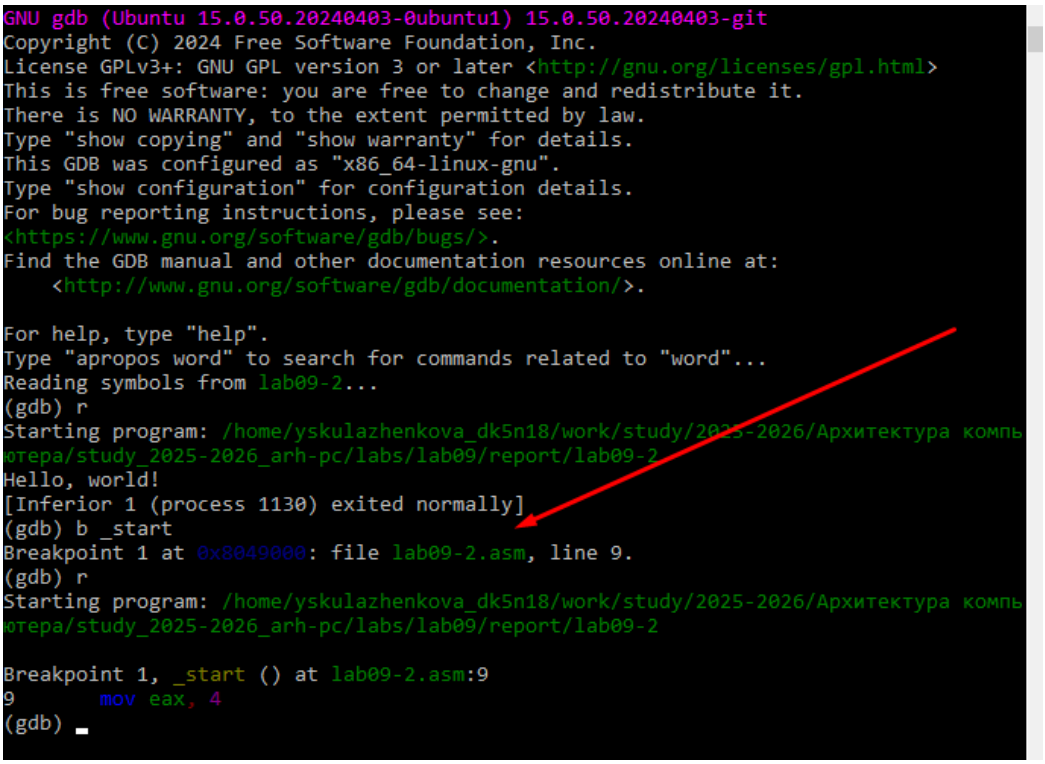

Рисунок 4.8: Установка breakpoint

Посмотрим дизассемблированный код программы с помощью команды disassemble (рис. 4.9).

Рисунок 4.9: Дизассемблированный код

Посмотрим также на отображение команд с Intel'овским синтаксисом, введя команду set (рис. 4.10).

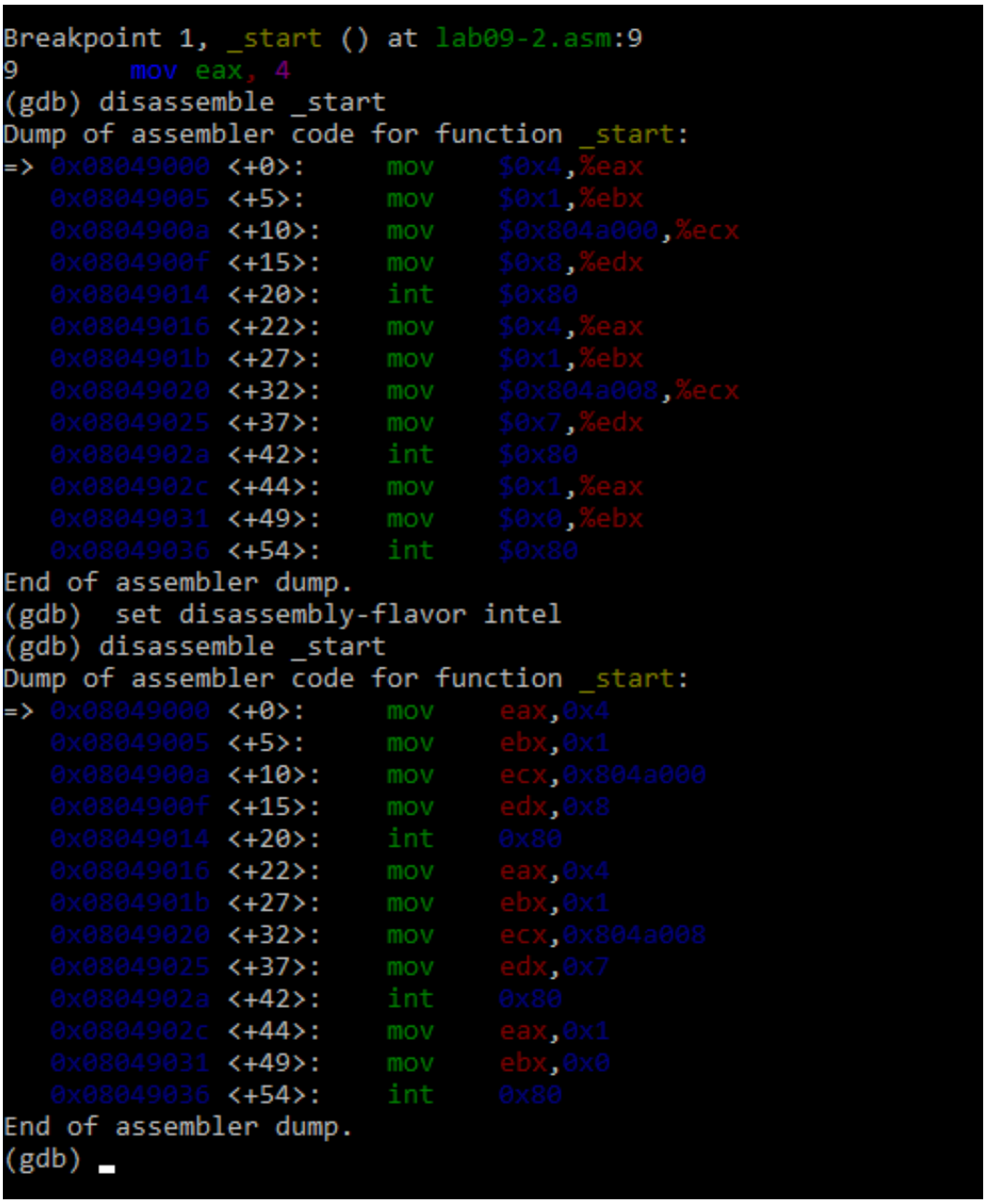

Рисунок 4.10: Отображение команды с Intel синтаксисом

Включим режим псевдографики для более удобного анализа программы (рис. 4.11).

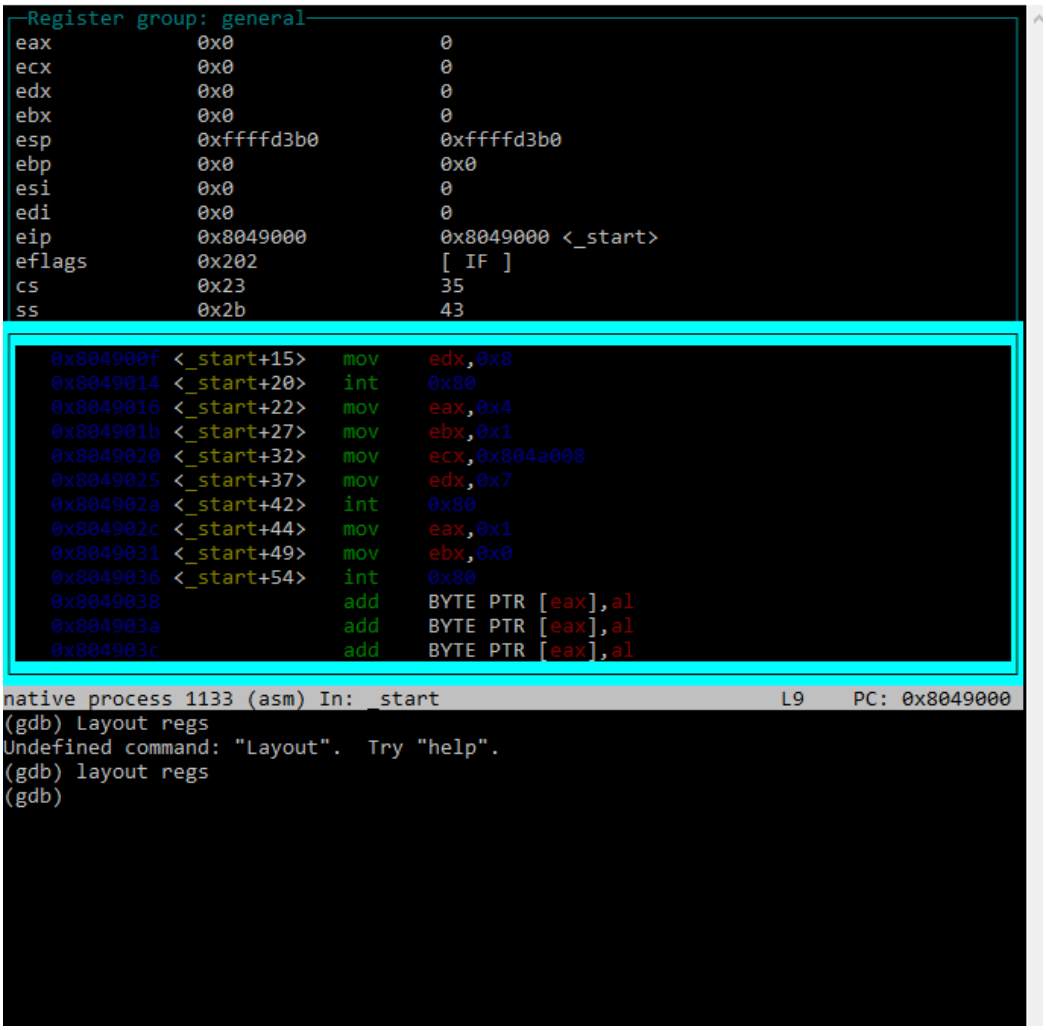

Рисунок 4.11: Решим псевдографики

4.3 Добавление точек останова

Для установки точки останова воспользуемся командой break (рис. 4.12).

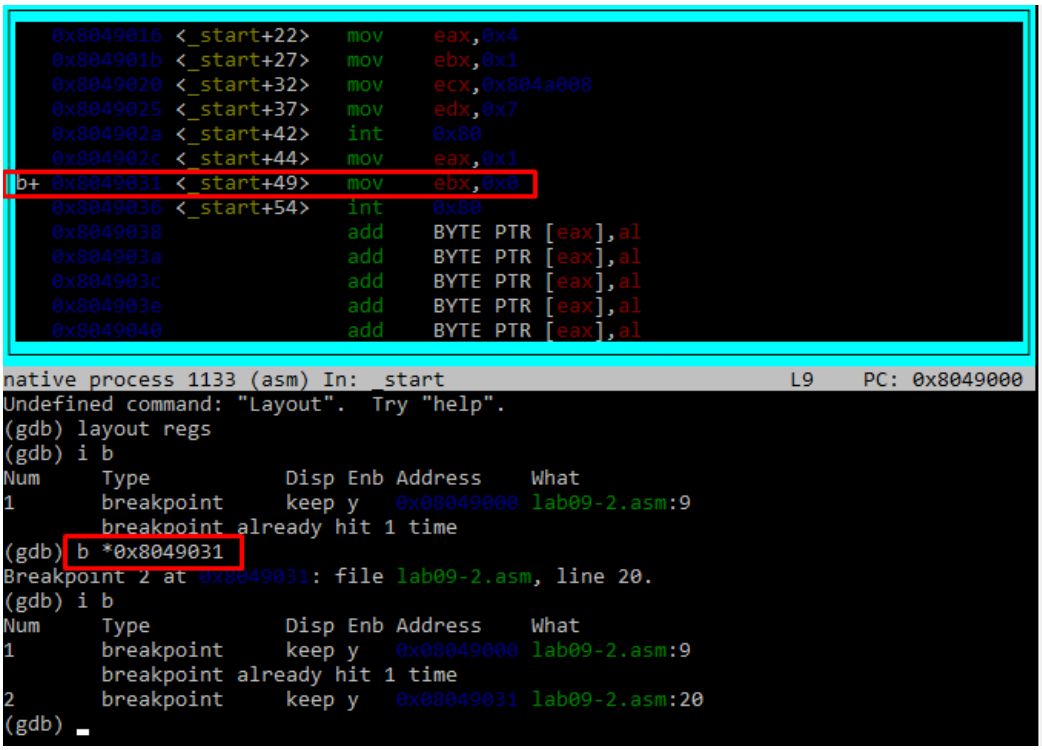

Рисунок 4.12: Установка точек останова

4.4 Работа с данными программы в GDB

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Изучим способы применения команды stepi и info registers. Посмотрим значение переменных msg1 по имени и msg2 по адресу (рис. 4.13).

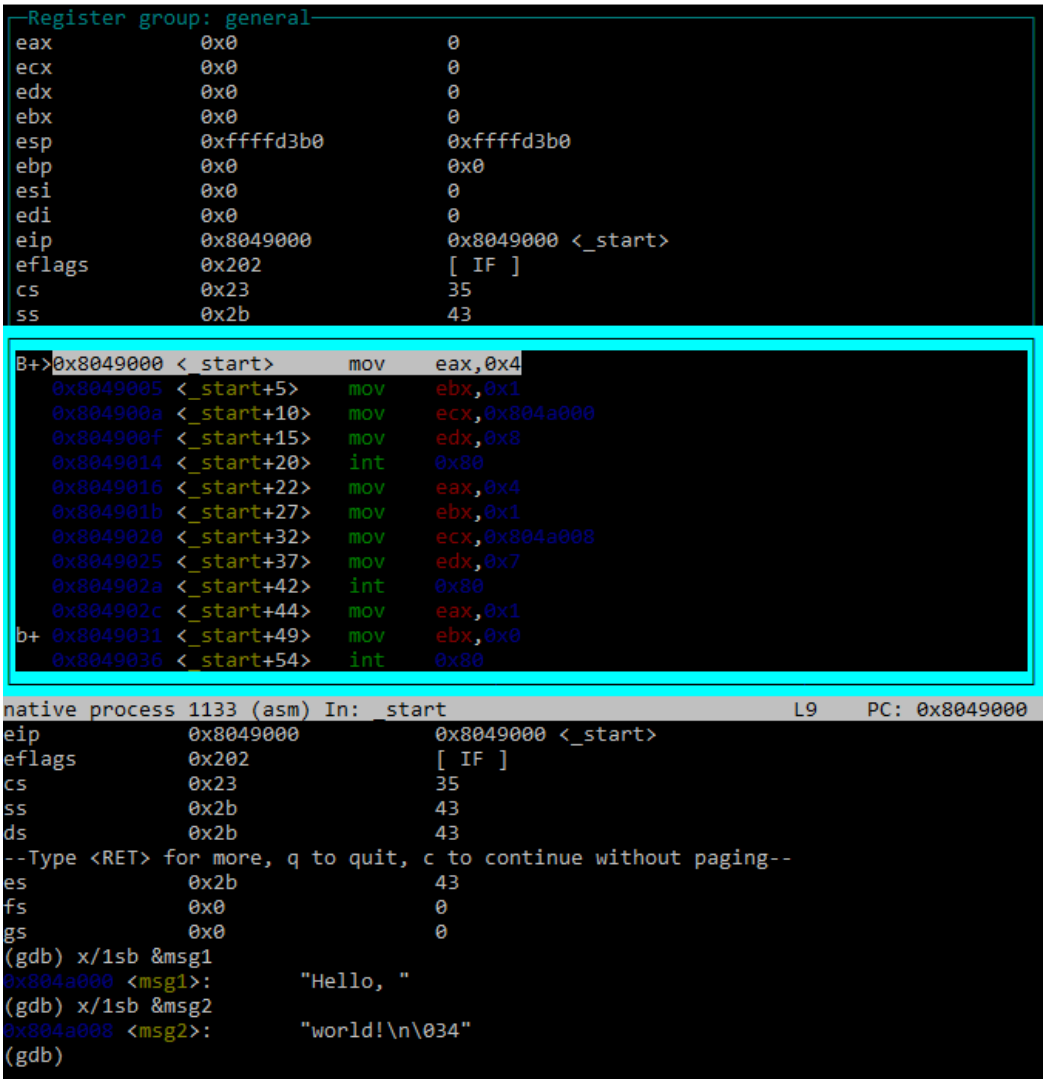

Рисунок 4.13: Значение переменных msg1 и msg2

Теперь изучим команду set. Изменим первый символ переменной msg1 (рис. 4.14).

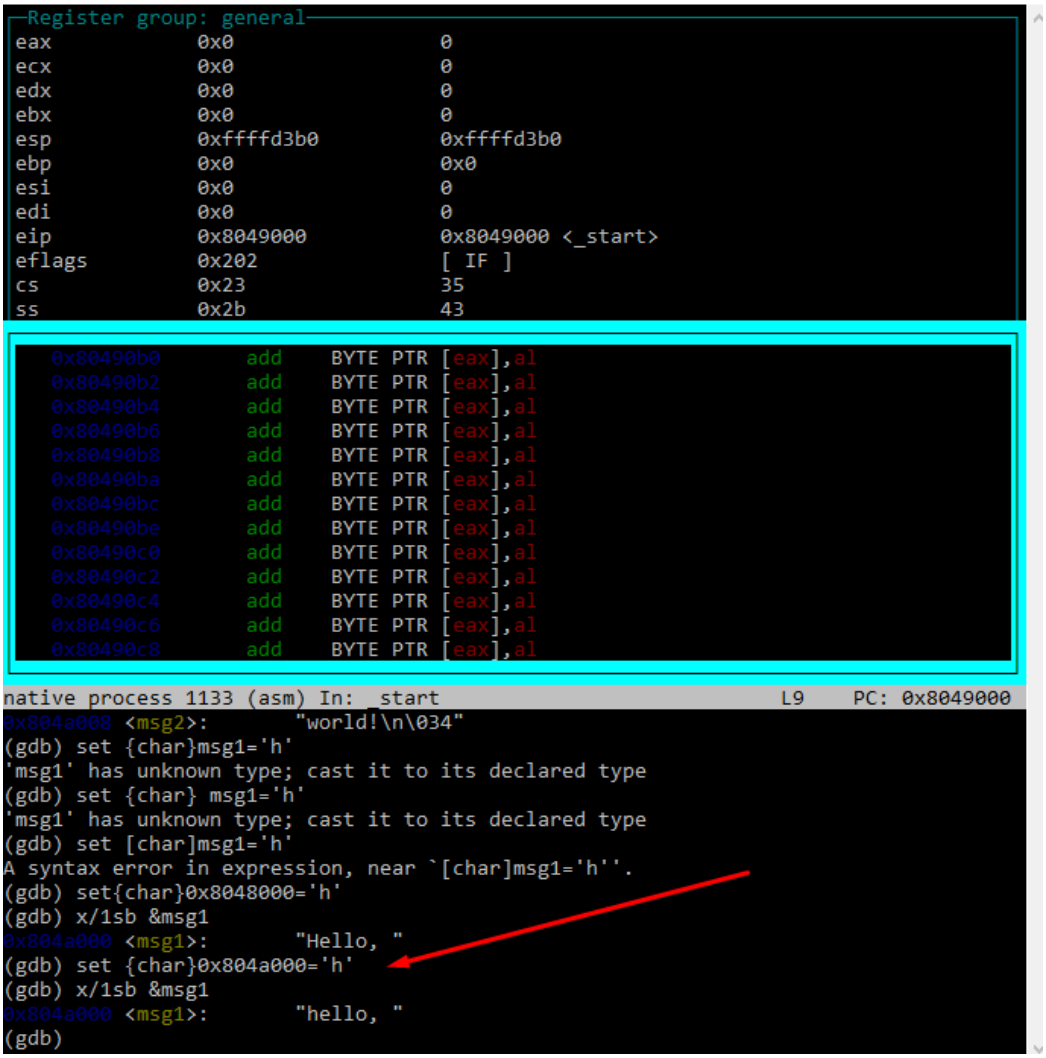

Рисунок 4.14: Использование команды set

Также заменим любой символ во второй переменной msg2 (рис. 4.15).

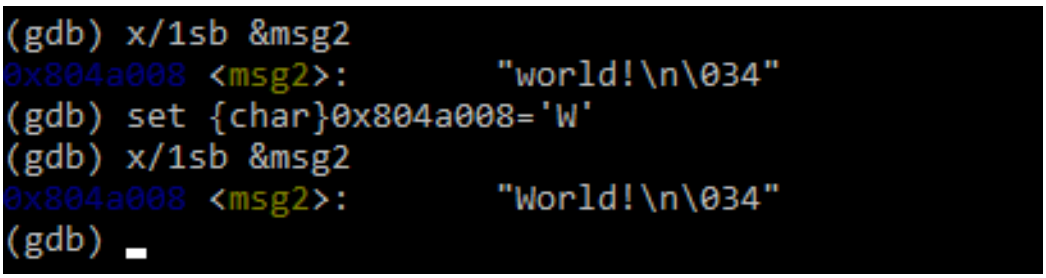

Рисунок 4.15: Использование команды set

Чтобы посмотреть значения регистров используем команду print /F (рис. 4.16).

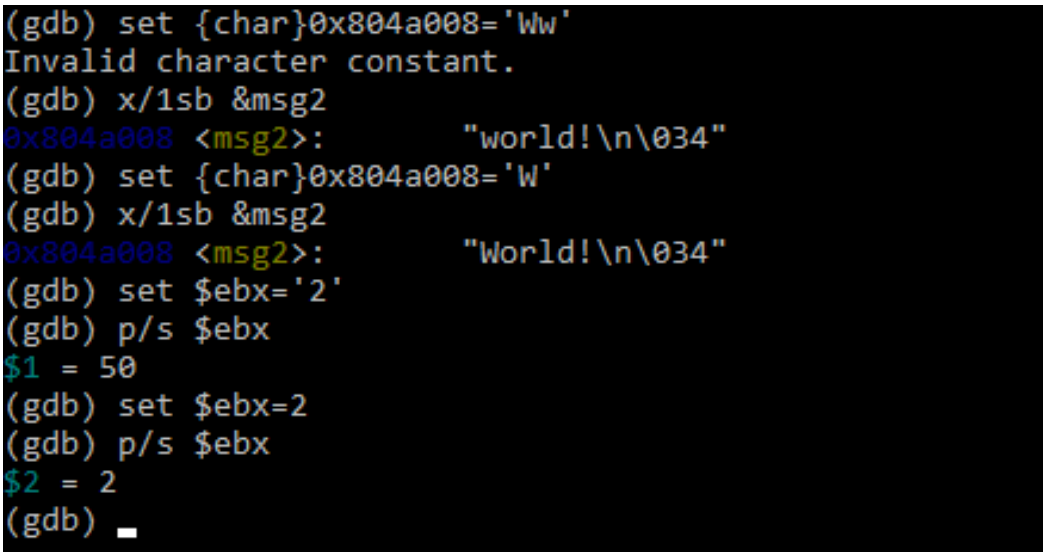

Рисунок 4.16: Просмотр значения регистров

4.5 Обработка аргументов командной строки в GDB

Подготовим файлы для работы. Скопируем файл lab8-2.asm и создадим исполняемый файл (рис. 4.17).

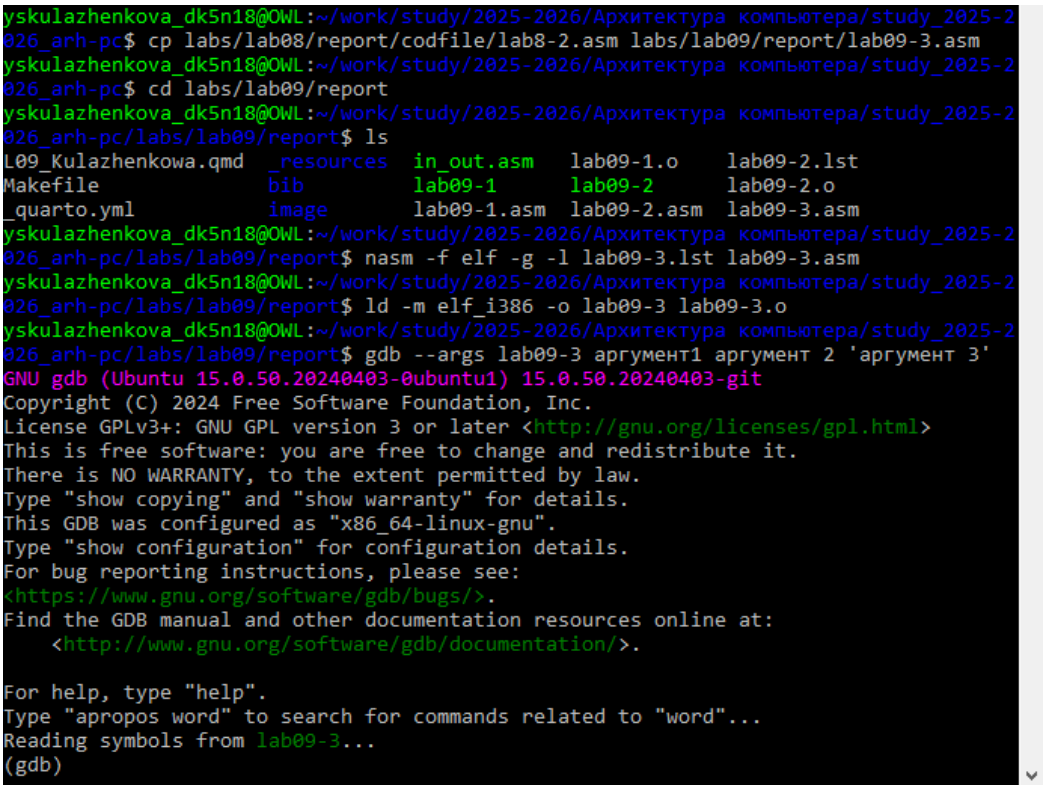

Рисунок 4.17: Подготовка файла для работы

Исследуем расположение аргументов командной строки в стеке после запуска программы с помощью gdb. Для этого установим точку останова перед первой инструкцией в программе и запустим ее (рис. 4.18).

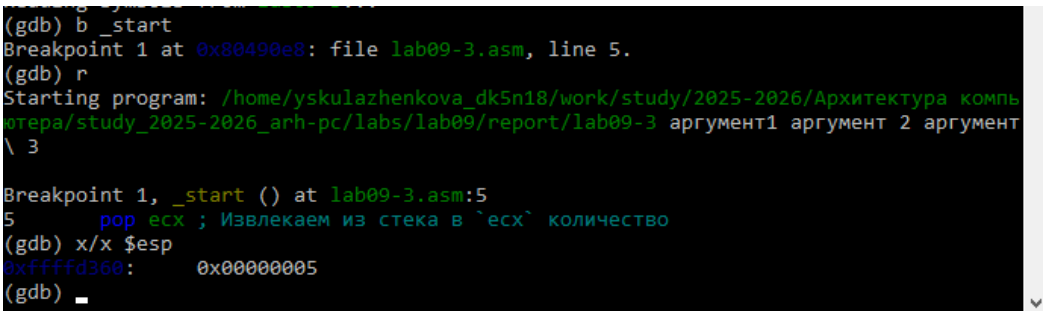

Рисунок 4.18: Установка точки останова

Продемонстрируем остальные позиции стека (рис. 4.19).

```
5       pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd360:     0x00000005
(gdb) x/s *(void**)($esp + 4)
0xffffd498:     "/home/yskulazhenkova_dk5n18/work/study/2025-2026/Архитектура компью
тера/study_2025-2026_arh-pc/labs/lab09/report/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd526:     "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xffffd538:     "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xffffd549:     "2"
(gdb) x/s *(void**)($esp + 20)
0xffffd54b:     "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb)
```

Рисунок 4.19: Позиции стека

5 Задание для самостоятельной работы

5.1 Задание 1

Реализуем вычисление значения функции как подпрограмму (рис. 5.1).

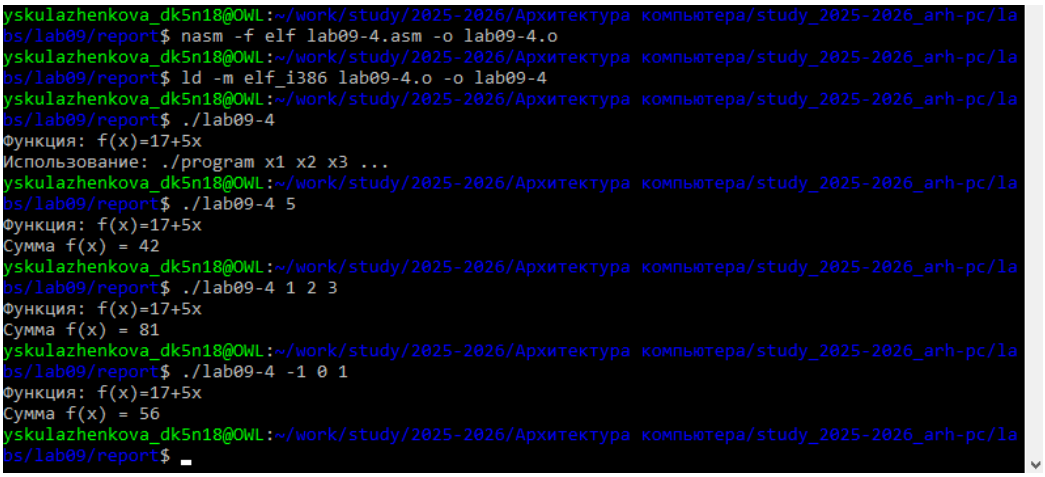

Рисунок 5.1: Задание 1

5.2 Задание 2

С помощью отладчика GDB определим ошибку и исправим ее (рис. 5.2) и (рис. 5.3).

```
(gdb) break _start
Breakpoint 1 at 0x80490e8: file programm.asm, line 8.
(gdb) к
Undefined command: "".  Try "help".
(gdb) r
Starting program: /home/yskulazhenkova_dk5n18/work/study/2025-2026/Архитектура компьютер
а/study_2025-2026_arh-pc/labs/lab09/report/programm

Breakpoint 1, _start () at programm.asm:8
8       mov ebx,3
(gdb) stepi
9       mov eax,2
(gdb) i r
eax            0x0                 0
ecx            0x0                 0
edx            0x0                 0
ebx            0x3                 3
esp            0xffffd400          0xffffd400
ebp            0x0                 0x0
esi            0x0                 0
edi            0x0                 0
eip            0x80490ed           0x80490ed <_start+5>
eflags         0x202               [ IF ]
cs             0x23                35
ss             0x2b                43
ds             0x2b                43
es             0x2b                43
fs             0x0                 0
gs             0x0                 0
(gdb) stepi
10      add ebx,eax
(gdb) i r
eax            0x2                 2
ecx            0x0                 0
edx            0x0                 0
ebx            0x3                 3
esp            0xffffd400          0xffffd400
ebp            0x0                 0x0
esi            0x0                 0
edi            0x0                 0
eip            0x80490f2           0x80490f2 <_start+10>
eflags         0x202               [ IF ]
cs             0x23                35
```

Рисунок 5.2: Задание 2

```
Continuing.
Результат: 10
[Inferior 1 (process 1330) exited normally]
(gdb) q
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_
arh-pc/labs/lab09/report$ nano programm.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_
arh-pc/labs/lab09/report$ nasm -f elf programm.asm -o programm.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_
arh-pc/labs/lab09/report$ ld -m elf_i386 -o programm programm.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_
arh-pc/labs/lab09/report$ ./programm
Результат: 25
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_
arh-pc/labs/lab09/report$
```

Рисунок 5.3: Задание 2

6 Выводы

В результате выполнения лабораторной работы были достигнуты следующие ключевые цели: 1. Освоены основы построения подпрограмм в языке ассемблера NASM и механизма их вызова посредством инструкций `call` и `ret`. 2. Получены практические навыки использования отладчика GDB для эффективного поиска и устранения ошибок в программах. 3. Ознакомились с процедурой дизассемблирования программы и работой с разными форматами отображения машинного кода.

Таким образом, закреплены знания о принципах организации и отладки программ на низком уровне, что обеспечит успешное дальнейшее развитие практических навыков программирования и понимания архитектуры вычислительных систем.