

Отчёт по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB

Кулаженкова Яна Сергеевна

Содержание

1 Цель работы	5
2 Задание	6
3 Теоретическое введение	7
4 Выполнение лабораторной работы	8
4.1 Реализация подпрограмм в NASM	8
4.2 Отладка программам с помощью GDB	12
4.3 Добавление точек останова	17
4.4 Работа с данными программы в GDB	18
4.5 Обработка аргументов командной строки в GDB	21
5 Задание для самостоятельной работы	24
5.1 Задание 1	24
5.2 Задание 2	24
6 Выводы	26

Список иллюстраций

4.1	Переход в каталог лабораторной работы №9	8
4.2	Содержимое файла lab09-1	9
4.3	Продемонстрируем работу файла lab09-1	10
4.4	Изменим код	11
4.5	Работа измененного кода	12
4.6	Программа для печати сообщения Hello world!	13
4.7	Загрузка исполняемого файла в отладчик	14
4.8	Установка breakpoint	14
4.9	Дизассемблированный код	15
4.10	Отображение команды с Intel синтаксисом	16
4.11	Решим псевдографики	17
4.12	Установка точек останова	18
4.13	Значение переменных msg1 и msg2	19
4.14	Использование команды set	20
4.15	Использование команды set	20
4.16	Просмотр значения регистров	21
4.17	Подготовка файла для работы	22
4.18	Установка точки останова	22
4.19	Позиции стека	23
5.1	Задание 1	24
5.2	Задание 2	25
5.3	Задание 2	25

Список таблиц

1 Цель работы

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

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 AT&T 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).

```
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report$ touch lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report$ ls
L09_Kulazhenkova.qmd Makefile _quarto.yml _resources bib image lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report$ -
```

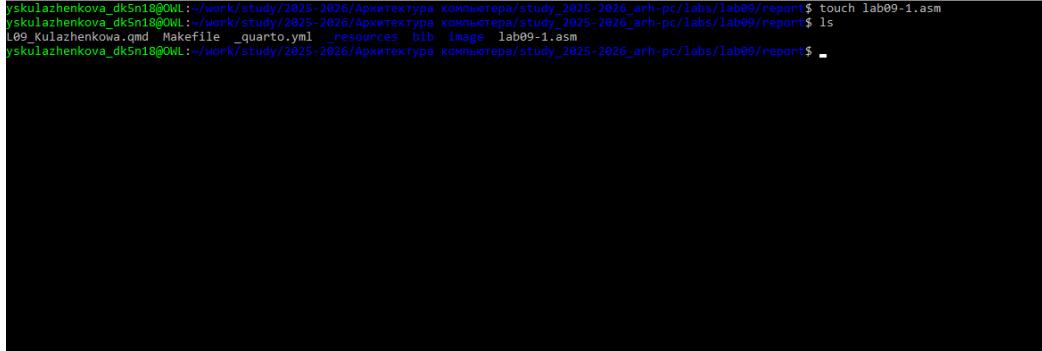


Рисунок 4.1: Переход в каталог лабораторной работы №9

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).

```
GNU nano 7.2                                lab09-1.asm *
%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
```

Рисунок 4.2: Содержимое файла lab09-1

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

```
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ nano lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ nasm -f elf lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ld -m elf_i386 -o lab09-1 lab09-1.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ./lab09-1
Ведите x: 5
2x+7=17
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ./lab09-1
Ведите x: 10
2x+7=27
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$
```

Рисунок 4.3: Продемонстрируем работу файла lab09-1

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

```

GNU nano 7.2                               lab09-1.asm *
mov eax, result_f
call sprint
mov eax, [res_f]
call iprintLF

; Вычисляем g(f(x)) = 3*f(x) - 1
mov eax, [res_f]    ; загружаем f(x) в eax
call _subcalcul
mov [res_g], eax    ; сохраняем результат g(f(x))

; Выводим результат g(f(x))
mov eax, result_g
call sprint
mov eax, [res_g]
call iprintLF

call quit

;-----
; Подпрограмма вычисления f(x) = 2x + 7
; Вход: eax = x
; Выход: eax = 2*x + 7
;-----
_calcut:
    mov ebx, 2
    mul ebx      ; eax = x * 2
    add eax, 7    ; eax = 2x + 7
    ret

;-----
; Подпрограмма вычисления g(f(x)) = 3*f(x) - 1
; Вход: eax = f(x)
; Выход: eax = 3*f(x) - 1
;-----
_subcalcul:
    mov ebx, 3
    mul ebx      ; eax = f(x) * 3
    sub eax, 1    ; eax = 3*f(x) - 1
    ret

```

Рисунок 4.4: Изменим код

Продемонстрируем работу кода программы (рис. 4.5).

```
026_arh-pc/labs/lab09/report$ nano lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ nasm -f elf lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ld -m elf_i386 -o lab09-1 lab09-1.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ./lab09-1
Введите x: 5
2x+7=17
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ nano lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ nasm -f elf lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ld -m elf_i386 -o lab09-1 lab09-1.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ./lab09-1
Введите x: 10
2x+7=27
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ nano lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ nasm -f elf lab09-1.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ld -m elf_i386 -o lab09-1 lab09-1.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ./lab09-1
Введите x: 5
f(x) = 2x+7 = 17
g(f(x)) = 3*(2x+7)-1 = 50
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ./lab09-1
Введите x: 10
f(x) = 2x+7 = 27
g(f(x)) = 3*(2x+7)-1 = 80
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ -
```

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).

```
GNU nano 7.2                                lab09-2.asm *
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
```

Рисунок 4.6: Программа для печати сообщения Hello world!

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

```
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report$ gdb lab09-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) r
Starting program: /home/yskulazhenkova_dk5n18/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report/lab09-2
Hello, world!
[Inferior 1 (process 1130) exited normally]
(gdb)
```

Рисунок 4.7: Загрузка исполняемого файла в отладчик

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

```
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) r
Starting program: /home/yskulazhenkova_dk5n18/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report/lab09-2
Hello, world!
[Inferior 1 (process 1130) exited normally]
(gdb) b _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) r
Starting program: /home/yskulazhenkova_dk5n18/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) -
```

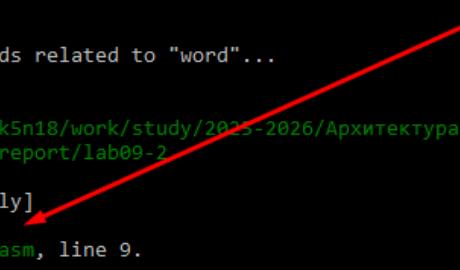


Рисунок 4.8: Установка breakpoint

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

```
Starting program: /home/yskulazhenkova_dk5n18/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov  eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:   mov    $0x4,%eax
  0x08049005 <+5>:   mov    $0x1,%ebx
  0x0804900a <+10>:  mov    $0x804a000,%ecx
  0x0804900f <+15>:  mov    $0x8,%edx
  0x08049014 <+20>:  int    $0x80
  0x08049016 <+22>:  mov    $0x4,%eax
  0x0804901b <+27>:  mov    $0x1,%ebx
  0x08049020 <+32>:  mov    $0x804a008,%ecx
  0x08049025 <+37>:  mov    $0x7,%edx
  0x0804902a <+42>:  int    $0x80
  0x0804902c <+44>:  mov    $0x1,%eax
  0x08049031 <+49>:  mov    $0x0,%ebx
  0x08049036 <+54>:  int    $0x80
End of assembler dump.
(gdb)
```

Рисунок 4.9: Дизассемблированный код

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

```

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:    mov    $0x4,%eax
    0x08049005 <+5>:    mov    $0x1,%ebx
    0x0804900a <+10>:   mov    $0x804a000,%ecx
    0x0804900f <+15>:   mov    $0x8,%edx
    0x08049014 <+20>:   int    $0x80
    0x08049016 <+22>:   mov    $0x4,%eax
    0x0804901b <+27>:   mov    $0x1,%ebx
    0x08049020 <+32>:   mov    $0x804a008,%ecx
    0x08049025 <+37>:   mov    $0x7,%edx
    0x0804902a <+42>:   int    $0x80
    0x0804902c <+44>:   mov    $0x1,%eax
    0x08049031 <+49>:   mov    $0x0,%ebx
    0x08049036 <+54>:   int    $0x80
End of assembler dump.
(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) -

```

Рисунок 4.10: Отображение команды с Intel синтаксисом

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

```
Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp 0xfffffd3b0 0xfffffd3b0
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip 0x8049000 0x8049000 <_start>
eflags        0x202      [ IF ]
cs           0x23         35
ss           0x2b         43

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
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al

native process 1133 (asm) In: _start                                L9      PC: 0x8049000
(gdb) Layout regs
Undefined command: "Layout". Try "help".
(gdb) layout regs
(gdb)
```

Рисунок 4.11: Решим псевдографики

4.3 Добавление точек останова

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

```

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
b+ 0x8049031 <_start+49>    mov    ebx,0x0
0x8049036 <_start+54>    int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al
0x804903e          add    BYTE PTR [eax],al
0x8049040          add    BYTE PTR [eax],al

native process 1133 (asm) In: _start                                L9      PC: 0x8049000
Undefined command: "Layout". Try "help".
(gdb) layout regs
(gdb) i b
Num      Type           Disp Enb Address     What
1        breakpoint     keep y  0x08049000  lab09-2.asm:9
          breakpoint already hit 1 time
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num      Type           Disp Enb Address     What
1        breakpoint     keep y  0x08049000  lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint     keep y  0x08049031  lab09-2.asm:20
(gdb) -

```

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).

```

Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp          0xfffffd3b0  0xfffffd3b0
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip          0x8049000   0x8049000 <_start>
eflags        0x202      [ IF ]
cs           0x23        35
ss           0x2b        43

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
b+ 0x8049031 <_start+49>   mov    ebx,0x0
0x8049036 <_start+54>     int    0x80

native process 1133 (asm) In: _start                                L9    PC: 0x8049000
eip          0x8049000   0x8049000 <_start>
eflags        0x202      [ IF ]
cs           0x23        35
ss           0x2b        43
ds           0x2b        43
--Type <RET> for more, q to quit, c to continue without paging--
es           0x2b        43
fs           0x0          0
gs           0x0          0
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "world!\n\034"
(gdb)

```

Рисунок 4.13: Значение переменных msg1 и msg2

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

```

Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp          0xfffffd3b0  0xfffffd3b0
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip          0x8049000  0x8049000 <_start>
eflags        0x202      [ IF ]
cs           0x23       35
ss           0x2b       43

0x80490b0    add     BYTE PTR [eax],al
0x80490b2    add     BYTE PTR [eax],al
0x80490b4    add     BYTE PTR [eax],al
0x80490b6    add     BYTE PTR [eax],al
0x80490b8    add     BYTE PTR [eax],al
0x80490ba    add     BYTE PTR [eax],al
0x80490bc    add     BYTE PTR [eax],al
0x80490be    add     BYTE PTR [eax],al
0x80490c0    add     BYTE PTR [eax],al
0x80490c2    add     BYTE PTR [eax],al
0x80490c4    add     BYTE PTR [eax],al
0x80490c6    add     BYTE PTR [eax],al
0x80490c8    add     BYTE PTR [eax],al

native process 1133 (asm) In: _start          L9      PC: 0x8049000
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char} msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set [char]msg1='h'
A syntax error in expression, near `[char]msg1='h'.
(gdb) set{char}0x8048000='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) set {char}0x804a000='h' ←
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb)

```

Рисунок 4.14: Использование команды set

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

```

(gdb) x/1sb &msg2
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}0x804a008='W'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "World!\n\034"
(gdb) -

```

Рисунок 4.15: Использование команды set

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

```
(gdb) set {char}0x804a008='Ww'  
Invalid character constant.  
(gdb) x/1sb &msg2  
0x804a008 <msg2>:      "world!\n\034"  
(gdb) set {char}0x804a008='W'  
(gdb) x/1sb &msg2  
0x804a008 <msg2>:      "World!\n\034"  
(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$1 = 50  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$2 = 2  
(gdb) -
```

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).

```

yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc$ cp labs/lab08/report/codfile/lab8-2.asm labs/lab09/report/lab09-3.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc$ cd labs/lab09/report
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ls
L09_Kulazhenkowa.qmd _resources in_out.asm lab09-1.o lab09-2.lst
Makefile bib lab09-1 lab09-2 lab09-2.o
_quarto.yml image lab09-1.asm lab09-2.asm lab09-3.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ ld -m elf_i386 -o lab09-3 lab09-3.o
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2
026_arh-pc/labs/lab09/report$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb)

```

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).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) r
Starting program: /home/yskulazhenkova_dk5n18/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/labs/lab09/report/lab09-3 аргумент1 аргумент 2 аргумент \ 3
Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в `есх` количество
(gdb) x/x $esp
0xfffffd360:    0x00000005
(gdb) -

```

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
0xffffffff360:    0x00000005
(gdb) x/s *(void**)( $esp + 4)
0xffffffff498:    "/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)
0xffffffff520:    "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xffffffff538:    "аргумент"
(gdb) x/s *(void**)( $esp + 16)
0xffffffff540:    "2"
(gdb) x/s *(void**)( $esp + 20)
0xffffffff54b:    "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).

```
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/la  
bs/lab09/report$ nasm -f elf lab09-4.asm -o lab09-4.o  
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/la  
bs/lab09/report$ ld -m elf_i386 lab09-4.o -o lab09-4  
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/la  
bs/lab09/report$ ./lab09-4  
Функция: f(x)=17+5x  
Использование: ./program x1 x2 x3 ...  
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/la  
bs/lab09/report$ ./lab09-4 5  
Функция: f(x)=17+5x  
Сумма f(x) = 42  
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/la  
bs/lab09/report$ ./lab09-4 1 2 3  
Функция: f(x)=17+5x  
Сумма f(x) = 81  
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/la  
bs/lab09/report$ ./lab09-4 -1 0 1  
Функция: f(x)=17+5x  
Сумма f(x) = 56  
yskulazhenkova_dk5n18@OWL:~/work/study/2025-2026/Архитектура компьютера/study_2025-2026_arh-pc/la  
bs/lab09/report$ -
```

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) k
Undefined command: "". Try "help".
(gdb) r
Starting program: /home/yskulazhenkova_dk5n18/work/study/2025-2026/Архитектура компьютера/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            0xfffffd400    0xfffffd400
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            0xfffffd400    0xfffffd400
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.