

Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Аннаоразов Сердар Аннаоразович

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 программ с помощью GDB	9
4.3	Работа с данными программы в GDB	13
4.4	Обработка аргументов командной строки в GDB	16
5	Самостоятельная работа	18
5.1	Первая задача	18
5.2	Вторая задача	19
6	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Создания каталога и файла	8
4.2	Введения программы в файл	9
4.3	Создание исполняемого файла и запуск его	9
4.4	Создание файла lab09-2.asm	9
4.5	Введения программы печати сообщения Hello world	10
4.6	Создание исполняемого файла lab09-2.asm и запуск его	10
4.7	Загрузка исполняемого файла в отладчик gdb	10
4.8	проверил работу программы	11
4.9	установка брейкпоинт на метку _start	11
4.10	Просмотр дисассимилированный код программы	11
4.11	Переключения на отображение команд с Intel'овским синтаксисом	12
4.12	режим псевдографики	12
4.13	Установка и Проверка точек сотанова	13
4.14	Просмотр содержимое регистров	13
4.15	Просмотр значение и изменения региястра	14
4.16	Просмотр значение переменной msg1	14
4.17	Просмотр значение переменной msg2	14
4.18	Изменения значения переменной msg1	14
4.19	Изменения переменной msg2	15
4.20	Вывод значения регистров еsx и еax	15
4.21	Изменения значения регистра ebx	15
4.22	Выход	15
4.23	Копирования файла	16
4.24	Запуск файла с аргументами	16
4.25	Запуск файла через метку	16
4.26	Адрес вершины стека	17
5.1	Копирования файла ил lab08	18
5.2	Преоброзования файла	18
5.3	Запуск файла	19
5.4	Запуск программы в окладчике	19
5.5	Исправления ошибок	20
5.6	Запуск исполняемого файла	20

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

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Добавление точек останова
4. Работа с данными программы в GDB
5. Обработка аргументов командной строки в GDB
6. Задание для самостоятельной работы

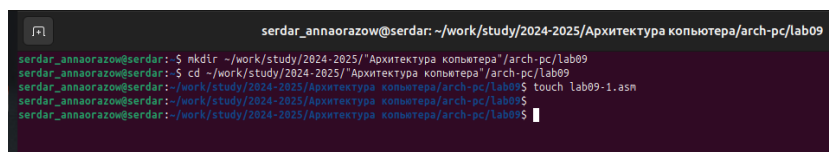
3 Теоретическое введение

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, отработывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль). Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно 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 пр

4 Выполнение лабораторной работы

4.1 Реализация подпрограмм в NASM

Создал каталог для выполнения лабораторной работы № 9, перешел в него и создал файл lab09-1.asm:



```
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
serdar_annaorazow@serdar: $ mkdir ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
serdar_annaorazow@serdar: $ cd ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ touch lab09-1.asm
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 4.1: Создания каталога и файла

В качестве примера рассмотрел программу вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы `_calcul`. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме. Написал программу в Файл


```
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
GNU nano 7.2 /home/serdar_annaorazow/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/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 iprintlnf
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
```

Рис. 4.2: Введения программы в файл

Создал исполняемый файл и запустил его

```
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab09-1.asm
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -n elf_i386 -o lab09-1 lab09-1.o
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./lab09-1
Введите x: 25
2x+7=57
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 4.3: Создание исполняемого файла и запуск его

4.2 Отладка программ с помощью GDB

Я создал новый файл lab09-2.asm

```
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ touch lab09-2.asm
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 4.4: Создание файла lab09-2.asm

ввел туда программу печати сообщения Hello world!

```
serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
GNU nano 7.2 /home/serdar_annaorazow/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/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
```

Рис. 4.5: Введения программы печати сообщения Hello world

Я создал исполняемый файл и запустил его

```
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab09-2.asm
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./lab09-2
Hello, world!
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 4.6: Создание исполняемого файла lab09-2.asm и запуск его

Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'. Загрузил исполняемый файл в отладчик gdb

```
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ 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) █
```

Рис. 4.7: Загрузка исполняемого файла в отладчик gdb

Проверил работу программы, запустив ее в оболочке GDB с помощью команды run

```

serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
serdar_annaorazow@serdar:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ 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) run
Starting program: /home/serdar_annaorazow/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 31049) exited normally]
(gdb)

```

Рис. 4.8: проверил работу программы

Для более подробного анализа программы установил брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустил её.

```

(gdb) break _start
Breakpoint 1 at 0x00490000: file lab09-2.asm, line 11.
(gdb) r
Starting program: /home/serdar_annaorazow/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
11      mov eax, 4
(gdb)

```

Рис. 4.9: установка брейкпоинт на метку `_start`

Посмотрел дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start`

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x00490000 <+0>:    mov     $0x1,%eax
0x00490005 <+5>:      mov     $0x1,%ebx
0x0049000a <+10>:     mov     $0x004a0000,%ecx
0x0049000f <+15>:     mov     $0x8,%edx
0x00490014 <+20>:     int     $0x80
0x00490016 <+22>:     mov     $0x4,%eax
0x0049001b <+27>:     mov     $0x1,%ebx
0x00490020 <+32>:     mov     $0x004a0000,%ecx
0x00490025 <+37>:     mov     $0x7,%edx
0x0049002a <+42>:     int     $0x80
0x0049002c <+44>:     mov     $0x1,%eax
0x00490031 <+49>:     mov     $0x0,%ebx
0x00490036 <+54>:     int     $0x80
End of assembler dump.
(gdb)

```

Рис. 4.10: Просмотр дисассимилированный код программы

Переключился на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`

```

(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,0x004a000
      0x0804900f <+15>:   mov     edx,0xb
      0x08049014 <+20>:   int     0xb0
      0x08049019 <+25>:   mov     eax,0xb
      0x0804901d <+27>:   mov     ebx,0x1
      0x08049020 <+32>:   mov     ecx,0x004a000
      0x08049025 <+37>:   mov     edx,0x7
      0x0804902a <+42>:   int     0xb0
      0x0804902c <+44>:   mov     eax,0x1
      0x08049031 <+49>:   mov     ebx,0x0
      0x08049036 <+54>:   int     0xb0
End of assembler dump.
(gdb)

```

Рис. 4.11: Переключения на отображение команд с Intel'овским синтаксисом

Перечислил различия отображения синтаксиса машинных команд в режимах АТТ и Intel. Включил режим псевдографики для более удобного анализа программы

```

serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
--Register group: general--
eax    0x0    0      ecx    0x0    0      edx    0x0    0
ebx    0x0    0      esp    0xffffcf30 0      ebp    0x0    0x0
esi    0x0    0      edi    0x0    0      ebp    0x0    0x0
eflags 0x202  [ IF ]  cs     0x23   35     ss     0x2b   43
ds     0x2b   43     es     0x2b   43     fs     0x0    0
gs     0x0    0

0x08049000 <_start>  mov     eax,0x4
0x08049005 <_start+5> mov     ebx,0x1
0x0804900a <_start+10> mov     ecx,0x004a000
0x0804900f <_start+15> mov     edx,0xb
0x08049014 <_start+20> int     0xb0
0x08049019 <_start+25> mov     eax,0xb
0x0804901d <_start+27> mov     ebx,0x1
0x08049020 <_start+32> mov     ecx,0x004a000
0x08049025 <_start+37> mov     edx,0x7
0x0804902a <_start+42> int     0xb0
0x0804902c <_start+44> mov     eax,0x1
0x08049031 <_start+49> mov     ebx,0x0
0x08049036 <_start+54> int     0xb0

native process 3117 (asm) int: _start
(gdb) layout reg.
(gdb)

```

Рис. 4.12: режим псевдографики

Проверил установленные точки останова с помощью команды `info breakpoints`. Установил еще одну точку останова по адресу инструкции. Адрес инструкции смог увидеть в средней части экрана в левом столбце соответствующей инструкции. Определил адрес предпоследней инструкции (`mov ebx,0x0`) и установил точку останова. Потом посмотрел информацию о всех установленных точках останова:

```

serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09

--Register group: general--
eax    0x0      0      ecx    0x0      0      edx    0x0      0
ebx    0x0      0      esp    0xffffcf30  0xffffcf30  ebp    0x0      0x0
esi    0x0      0      edi    0x0      0      eip    0x8049000  0x8049000 <_start>
eflags 0x202    [ IF ]  cs     0x23      35      ss     0x2b      43
ds     0x2b      43      es     0x2b      43      fs     0x0      0
gs     0x0      0

b> 0x0049000 <_start> mov    eax,0x4
0x0049000 <_start+5> mov    ecx,ecx
0x0049000 <_start+10> mov    esi,0x0049000
0x0049000 <_start+15> mov    edi,edx
0x0049000 <_start+20> int    0x0
0x0049000 <_start+25> mov    eax,edx
0x0049000 <_start+30> mov    esi,0x0049000
0x0049000 <_start+35> mov    edi,edx
0x0049000 <_start+40> int    0x0
0x0049000 <_start+45> mov    esi,ecx
0x0049000 <_start+50> mov    edi,edx
0x0049000 <_start+55> int    0x0

native process 31117 (asm) In: _start
(gdb) Layout regs
(gdb) break *0x0049001
Breakpoint 2 at 0x0049001: file lab09-2.asm, line 26.
(gdb) t b
Num Type Disp Enb Address What
1 breakpoint keep y 0x0049000 lab09-2.asm:11
breakpoint already hit 1 time
2 breakpoint keep y 0x0049001 lab09-2.asm:26
(gdb)

```

Рис. 4.13: Установка и Проверка точек сотанова

4.3 Работа с данными программы в GDB

Для начала посмотрел содержимое регистров с помощью команды info registers

```

serdar_annaorazow@serdar: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09

--Register group: general--
eax    0x0      0      ecx    0x0      0      edx    0x0      0
ebx    0x0      0      esp    0xffffcf30  0xffffcf30  ebp    0x0      0x0
esi    0x0      0      edi    0x0      0      eip    0x8049000  0x8049000 <_start>
eflags 0x202    [ IF ]  cs     0x23      35      ss     0x2b      43
ds     0x2b      43      es     0x2b      43      fs     0x0      0
gs     0x0      0

b> 0x0049000 <_start> mov    eax,0x4
0x0049000 <_start+5> mov    ecx,ecx
0x0049000 <_start+10> mov    esi,0x0049000
0x0049000 <_start+15> mov    edi,edx
0x0049000 <_start+20> int    0x0
0x0049000 <_start+25> mov    eax,edx
0x0049000 <_start+30> mov    esi,0x0049000
0x0049000 <_start+35> mov    edi,edx
0x0049000 <_start+40> int    0x0
0x0049000 <_start+45> mov    esi,ecx
0x0049000 <_start+50> mov    edi,edx
0x0049000 <_start+55> int    0x0

native process 31117 (asm) In: _start
L11 PC: 0x0049000
eax    0x0
ecx    0x0
edx    0x0
ebx    0x0
esp    0xffffcf30  0xffffcf30
ebp    0x0
esi    0x0
edi    0x0
eip    0x8049000  0x8049000 <_start>
eflags 0x202    [ IF ]
cs     0x23      35
ss     0x2b      43
ds     0x2b      43
es     0x2b      43
fs     0x0
gs     0x0
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 4.14: Просмотр содержимое регистров

С помощью команды si я посмотрел регистры и изменил их.

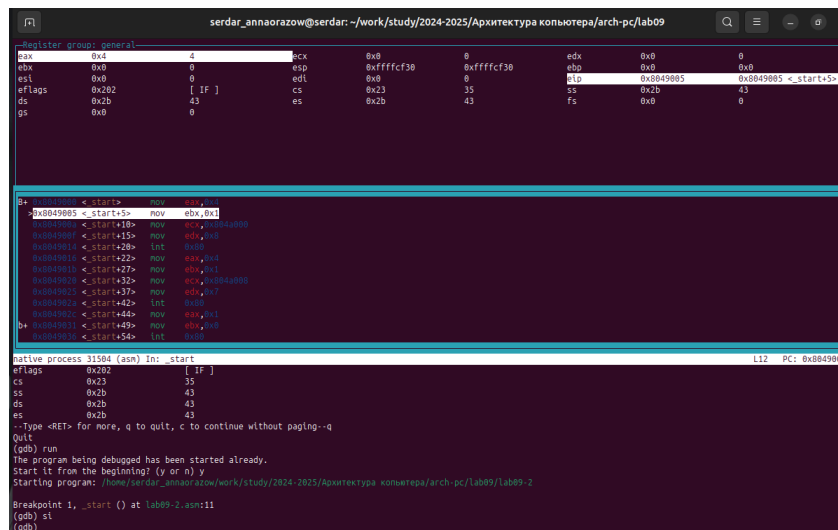


Рис. 4.15: Просмотр значение и изменения регистра

С помощью команды я посмотрел значение переменной msg1.

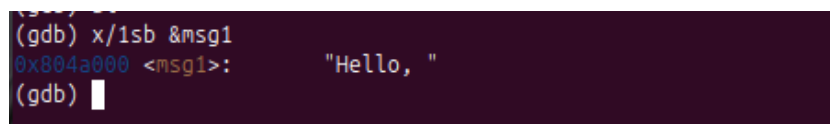


Рис. 4.16: Просмотр значение переменной msg1

Следом я посмотрел значение второй переменной msg2

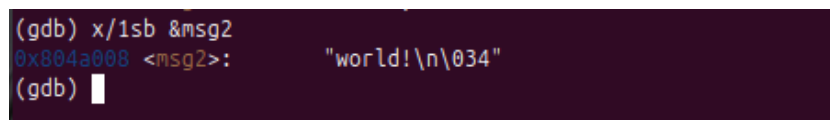


Рис. 4.17: Просмотр значение переменной msg2

С помощью команды set я изменил значение переменной msg1.

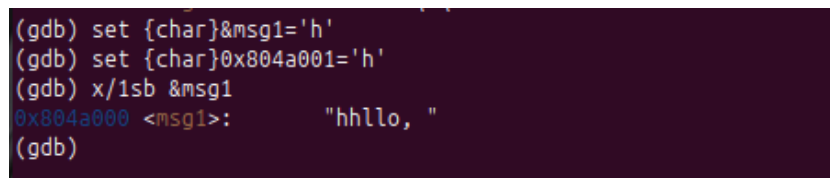


Рис. 4.18: Изменения значения переменной msg1

Я изменил переменную msg2

```
(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "Lor d!\n\034"
(gdb)
```

Рис. 4.19: Изменения переменной msg2

Я вывел значение регистров ecx и eax.

```
(gdb) p/f $msg1
$1 = void
(gdb) p/s $eax
$2 = 4
(gdb) p/t $eax
$3 = 100
(gdb) p/c $ecx
$4 = 0 '\000'
(gdb) p/x $ecx
$5 = 0x0
(gdb)
```

Рис. 4.20: Вывод значения регистров ecx и eax

Я изменил 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.

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb)
```

Рис. 4.21: Изменения значения регистра ebx

Я завершил работу с файлов вышел

```
Inferior 1 [process 35279] will be killed.
```

Рис. 4.22: Выход

4.4 Обработка аргументов командной строки в GDB

Я скопировал файл lab9-2.asm и переименовал его. Я сделал это через МС так как по мне это удобнее всего

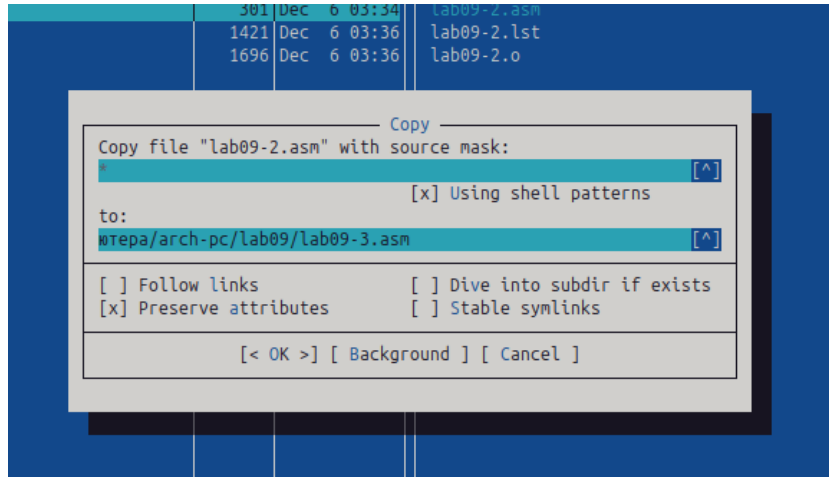


Рис. 4.23: Копирования файла

Запустил файл в отладчике и указал аргументы.

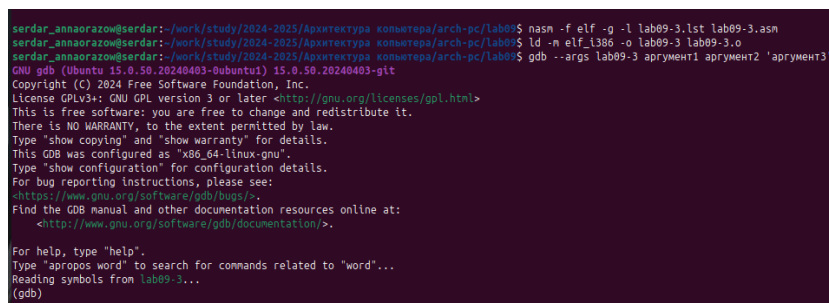


Рис. 4.24: Запуск файла с аргументами

Поставил метку на _start и запустил файл

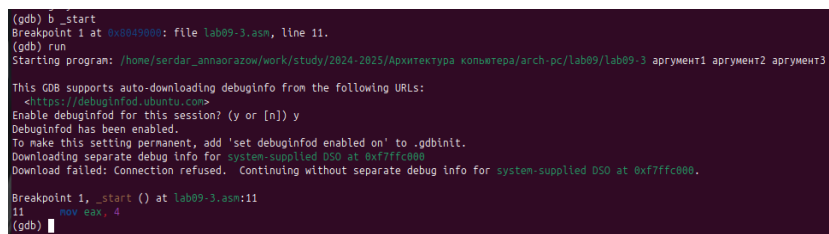
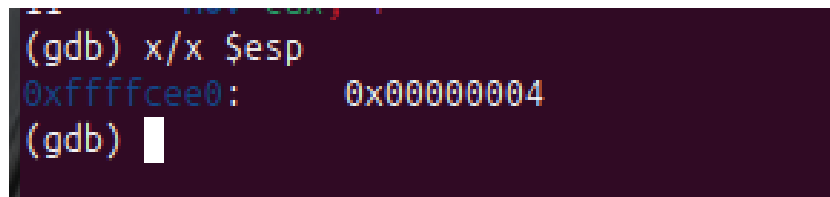


Рис. 4.25: Запуск файла через метку

Я проверил адрес вершины стека и убедился что там хранится 5 элементов.

A screenshot of a GDB terminal window with a dark background. The text is as follows:

```
(gdb) x/x $esp  
0xffffcee0: 0x00000004  
(gdb) █
```

Рис. 4.26: Адрес вершины стека

Я посмотрел 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.

5 Самостоятельная работа

5.1 Первая задача

Я с начала копировал файл в котором делал самостоятельную работу 8-ой лабораторной работы. Я это сделал с помощью МС.

```
(gdb) x/s *(void**)(Sesp + 4)
0xffffd00e: "/home/serdar_annaorazow/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(Sesp + 8)
0xffffd110: "аргумент1"
(gdb) x/s *(void**)(Sesp + 12)
0xffffd12d: "аргумент2"
(gdb) x/s *(void**)(Sesp + 16)
0xffffd13f: "аргумент3"
(gdb) x/s *(void**)(Sesp + 20)
0x8: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 5.1: Копирования файла ил lab08

Потом я преобразовал программу из лабораторной работы №8 и реализовал вычисления как подпрограмму.

Left	File	Command	Options	Right
..	None			..
lab09-1.asm	976	Dec 6 03:29	Dec 6 03:29	lab09-1.asm
*lab09-1	9152	Dec 6 03:32	Dec 6 03:32	*lab09-1
lab09-1.o	700	Dec 6 03:31	Dec 6 03:31	lab09-1.o
lab09-2	1456	Dec 6 03:32	Dec 6 03:32	lab09-2
lab09-2.asm	9220	Dec 6 03:36	Dec 6 03:36	lab09-2.asm
lab09-2.lst	501	Dec 6 03:34	Dec 6 03:34	lab09-2.lst
lab09-2.o	1421	Dec 6 03:36	Dec 6 03:36	lab09-2.o
*lab09-3	1696	Dec 6 03:36	Dec 6 03:36	*lab09-3
lab09-3.asm	9220	Dec 7 17:11	Dec 7 17:11	lab09-3.asm
lab09-3.lst	201	Dec 6 03:34	Dec 6 03:34	lab09-3.lst
lab09-3.o	1421	Dec 7 17:10	Dec 7 17:10	lab09-3.o
*самостоятельная_работа_1	1696	Dec 7 17:10	Dec 7 17:10	*самостоятельная_работа_1
самостоятельная_работа_1.asm	9140	Dec 7 18:16	Dec 7 18:16	самостоятельная_работа_1.asm

Рис. 5.2: Преобразования файла

5.2 Вторая задача

Я создал новый файл и ввел туда программу который был в задаче. После этого я сохдал исполняемый файл и запустил его чтоб увидеть в чем ошибка.

```
serdar_annaorazov@serdar: /work/study/2024-2025/Архитектура_компьютера/arch-pc/lab09$ gedit самостоятельная_работа_2.asm
serdar_annaorazov@serdar: /work/study/2024-2025/Архитектура_компьютера/arch-pc/lab09$ nasm -f elf самостоятельная_работа_2.asm
serdar_annaorazov@serdar: /work/study/2024-2025/Архитектура_компьютера/arch-pc/lab09$ ld -m elf_i386 -o самостоятельная_работа_2.o самостоятельная_работа_2.o
serdar_annaorazov@serdar: /work/study/2024-2025/Архитектура_компьютера/arch-pc/lab09$ ./самостоятельная_работа_2
Результат: 10
serdar_annaorazov@serdar: /work/study/2024-2025/Архитектура_компьютера/arch-pc/lab09$
```

Рис. 5.3: Запуск файла

После проявление ошибки я запустил программу в окладчике.

```
GNU gdb (Ubuntu 15.0.50-20240403-0ubuntu1) 15.0.50-20240403-g1t
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 самостоятельная_работа_2.o...
(gdb) b _start
Breakpoint 1 at 0x00401000: file самостоятельная_работа_2.asm, line 8.
(gdb) run
Starting program: /home/serdar_annaorazov/work/study/2024-2025/Архитектура_компьютера/arch-pc/lab09/самостоятельная_работа_2

This GDB supports auto-downloading debugInfo from the following URLs:
<https://debuginfod.ubuntu.com>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0x77ffc000

Breakpoint 1, _start () at самостоятельная_работа_2.asm:8
8      mov     ebx, 3
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x00401000 <+0>: mov     eax, 0x3
0x00401002 <+5>: mov     eax, 0x0
0x00401004 <+10>: add     ebx, eax
0x00401006 <+15>: mov     ecx, 0x4
0x00401008 <+17>: mul     ecx
0x0040100a <+19>: add     ebx, 0x5
0x0040100c <+22>: mov     edi, ebx
0x0040100e <+24>: mov     esi, 0x00401000
0x00401010 <+29>: call   0x0040100f <sprintf>
```

Рис. 5.4: Запуск программы в окладчике

Я открыл регистры и проанализировал их, понял что некоторые регистры стоят не на своих местах и исправил это.

6 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при помощи GDB и его основными возможностями.

Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089096/mod_resource/content/0/%D0%9B%D0%B0%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%D0%9F%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D0%B5%D0%BF%D0%BE%D0%B4%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA.pdf