

Отчёт по лабораторной работе 9

Архитектура компьютера

Булут Умут

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация подпрограмм в NASM	6
2.2	Отладка программ с помощью GDB	10
2.3	Задание для самостоятельной работы	21
3	Выводы	27

Список иллюстраций

2.1	Программа в файле lab9-1.asm	7
2.2	Запуск программы lab9-1.asm	8
2.3	Программа в файле lab9-1.asm	9
2.4	Запуск программы lab9-1.asm	9
2.5	Программа в файле lab9-2.asm	10
2.6	Запуск программы lab9-2.asm в отладчике	11
2.7	Дизассимилированный код	12
2.8	Дизассимилированный код в режиме интел	13
2.9	Точка остановки	14
2.10	Изменение регистров	15
2.11	Изменение регистров	16
2.12	Изменение значения переменной	17
2.13	Вывод значения регистра	18
2.14	Вывод значения регистра	19
2.15	Программа в файле lab9-3.asm	20
2.16	Вывод значения регистра	21
2.17	Программа в файле task-1.asm	22
2.18	Запуск программы task-1.asm	23
2.19	Код с ошибкой в файле task-2.asm	24
2.20	Отладка task-2.asm	25
2.21	Код исправлен в файле task-2.asm	26
2.22	Проверка работы task-2.asm	26

Список таблиц

1 Цель работы

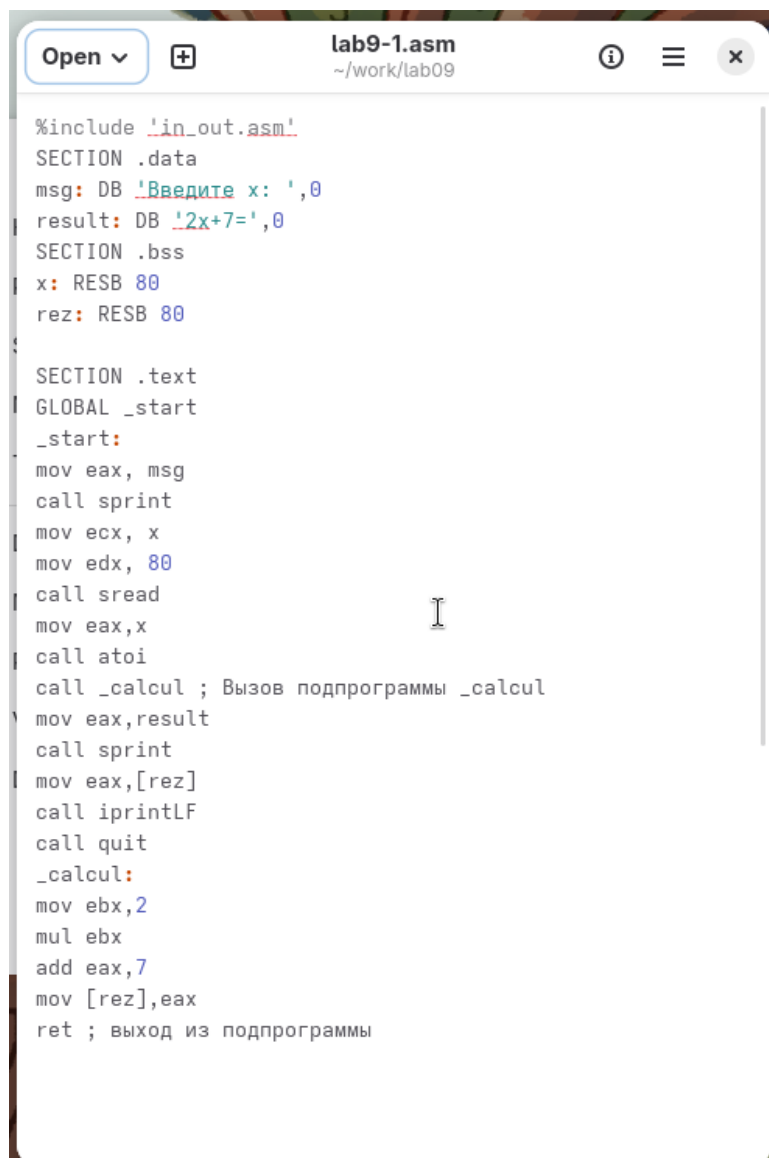
Целью работы является приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

Я создал каталог для выполнения лабораторной работы №9 и перешел в него.

В качестве примера рассмотрим программу, которая вычисляет арифметическое выражение $f(x) = 2x + 7$ с использованием подпрограммы `calcul`. В этом примере значение x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.



```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: 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 ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы
```

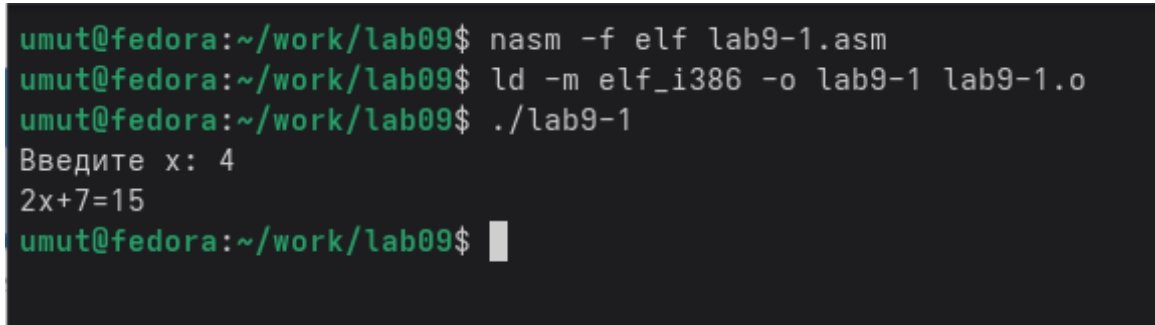
Рисунок 2.1: Программа в файле lab9-1.asm

Первые строки программы отвечают за вывод сообщения на экран (с помощью вызова `sprint`), чтение данных, введенных с клавиатуры (с помощью вызова `sread`) и преобразование введенных данных из символьного вида в численный (с помощью вызова `atoi`).

После инструкции `call _calcul`, которая передает управление подпрограмме `_calcul`, будут выполнены инструкции, содержащиеся в подпрограмме.

Инструкция `get` является последней в подпрограмме и ее выполнение приводит к возврату в основную программу к инструкции, следующей за инструкцией `call`, которая вызвала данную подпрограмму.

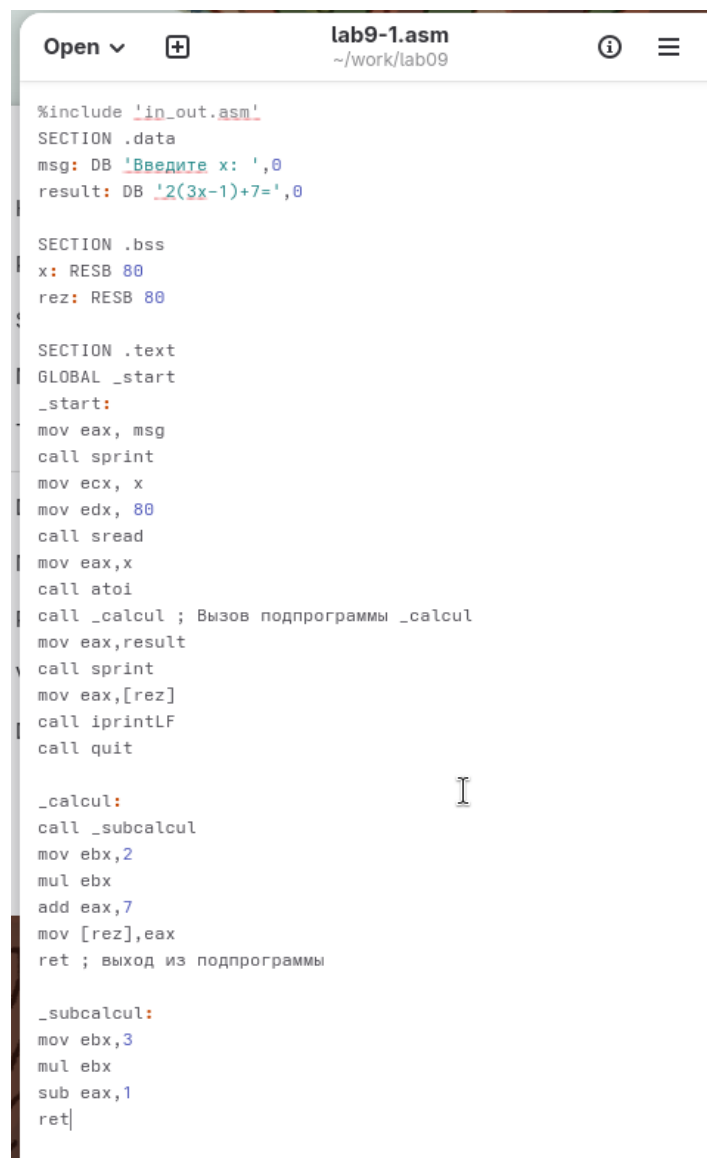
Последние строки программы реализуют вывод сообщения (с помощью вызова `sprint`), вывод результата вычисления (с помощью вызова `iprintLF`) и завершение программы (с помощью вызова `quit`).



```
umut@fedora:~/work/lab09$ nasm -f elf lab9-1.asm
umut@fedora:~/work/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
umut@fedora:~/work/lab09$ ./lab9-1
Введите x: 4
2x+7=15
umut@fedora:~/work/lab09$
```

Рисунок 2.2: Запуск программы `lab9-1.asm`

Изменил текст программы, добавив подпрограмму `subcalcul` в подпрограмму `calcul`, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$.



```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2(3x-1)+7=',0

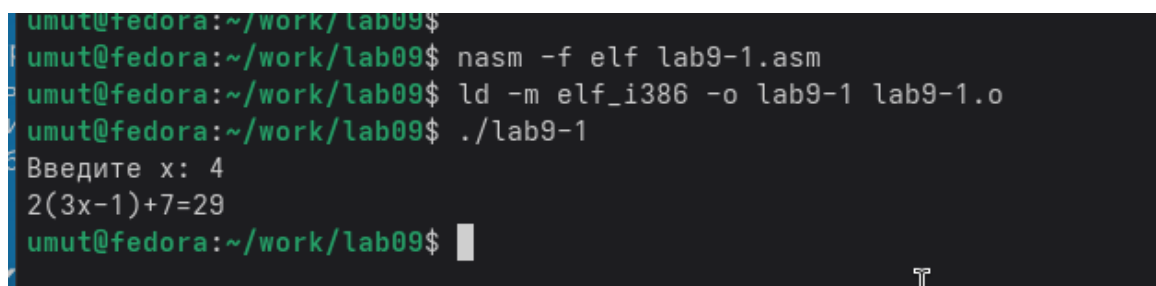
SECTION .bss
x: RESB 80
rez: RESB 80

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 ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [rez]
call iprintLF
call quit

_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret ; выход из подпрограммы

_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

Рисунок 2.3: Программа в файле lab9-1.asm

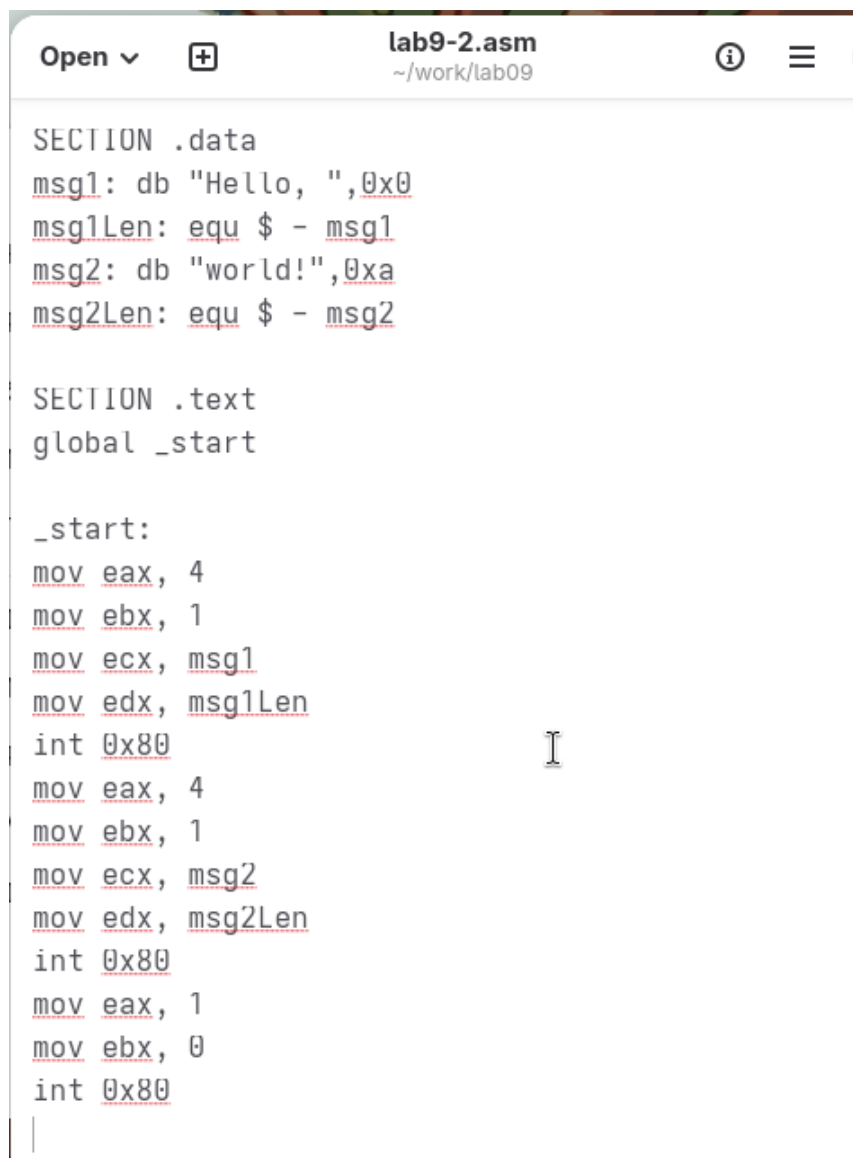


```
umut@fedora:~/work/lab09$
umut@fedora:~/work/lab09$ nasm -f elf lab9-1.asm
umut@fedora:~/work/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
umut@fedora:~/work/lab09$ ./lab9-1
Введите x: 4
2(3x-1)+7=29
umut@fedora:~/work/lab09$
```

Рисунок 2.4: Запуск программы lab9-1.asm

2.2 Отладка программ с помощью GDB

Создал файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!).



```
SECTION .data
msg1: db "Hello, ",0x0
msg1Len: equ $ - msg1
msg2: db "world!",0xa
msg2Len: equ $ - msg2

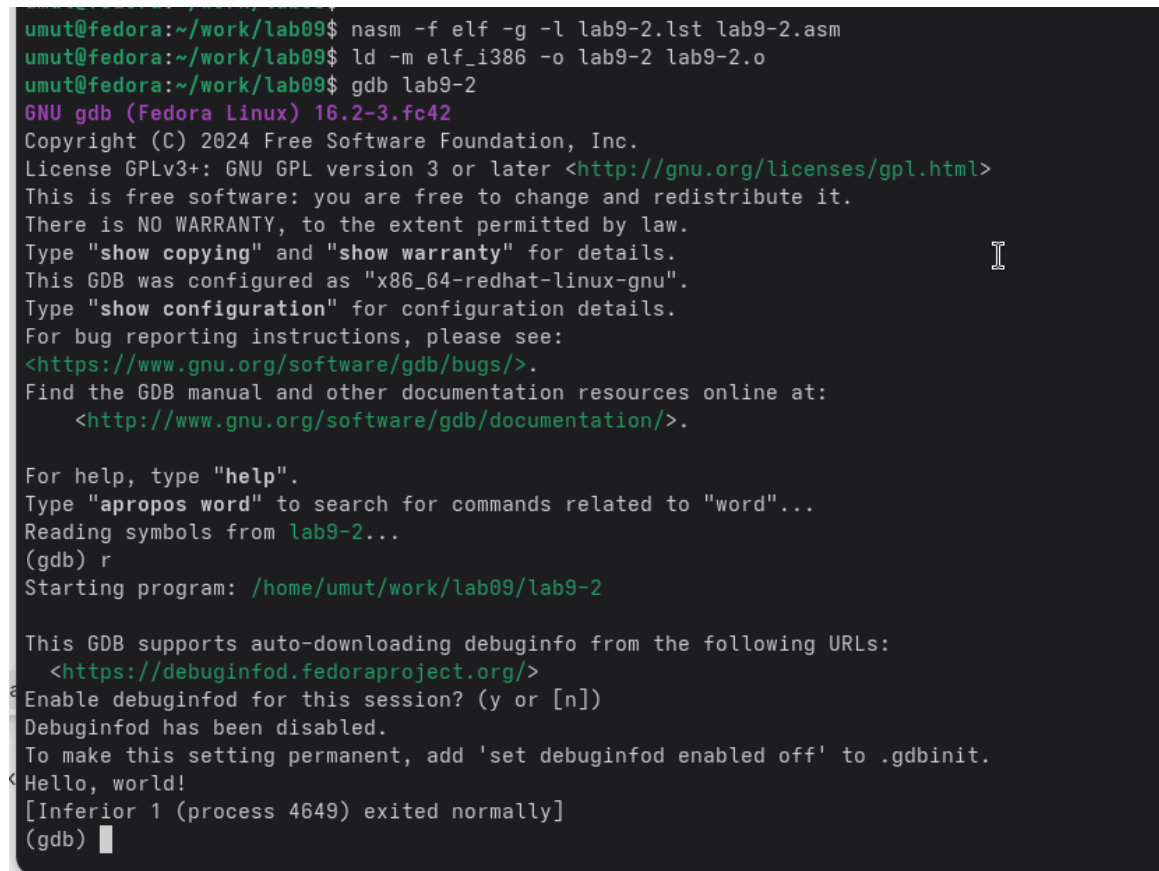
SECTION .text
global _start

_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 2.5: Программа в файле lab9-2.asm

Получил исполняемый файл. 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. Проверил работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r).

A screenshot of a terminal window with a dark background and light-colored text. The terminal shows the following sequence of commands and output:
umut@fedora:~/work/lab09\$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
umut@fedora:~/work/lab09\$ ld -m elf_i386 -o lab9-2 lab9-2.o
umut@fedora:~/work/lab09\$ gdb lab9-2
GNU gdb (Fedora Linux) 16.2-3.fc42
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-redhat-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 lab9-2...
(gdb) r
Starting program: /home/umut/work/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
 <<https://debuginfod.fedoraproject.org/>>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4649) exited normally]
(gdb) █

Рисунок 2.6: Запуск программы lab9-2.asm в отладчике

Для более подробного анализа программы установите брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запустите её. Посмотрите дисассимилированный код программы.

```
umut@fedora:~/work/lab09 — gdb lab9-2
~/work/lab09
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4649) exited normally]
(gdb)
break _start
Undefined command: "". Try "help".
(gdb)
break _start
Undefined command: "". Try "help".
(gdb) break _start
Breakpoint 1 at 0x8048080: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/umut/work/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:11
11      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>: mov $0x4,%eax
    0x08048085 <+5>: mov $0x1,%ebx
    0x0804808a <+10>: mov $0x8049000,%ecx
    0x0804808f <+15>: mov $0x8,%edx
    0x08048094 <+20>: int $0x80
    0x08048096 <+22>: mov $0x4,%eax
    0x0804809b <+27>: mov $0x1,%ebx
    0x080480a0 <+32>: mov $0x8049008,%ecx
    0x080480a5 <+37>: mov $0x7,%edx
    0x080480aa <+42>: int $0x80
    0x080480ac <+44>: mov $0x1,%eax
    0x080480b1 <+49>: mov $0x0,%ebx
    0x080480b6 <+54>: int $0x80
End of assembler dump.
(gdb) █
```

Рисунок 2.7: Дизассимилированный код

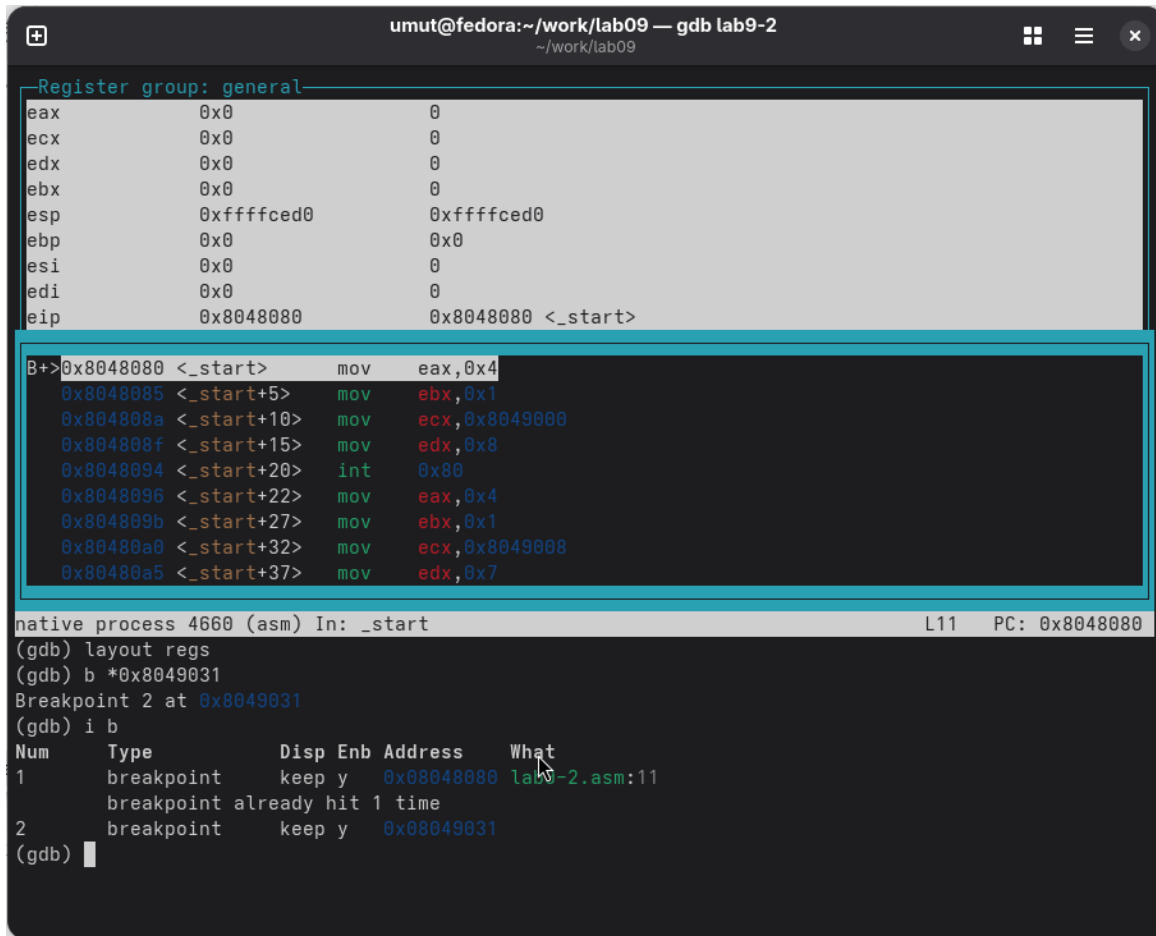
```
umut@fedora:~/work/lab09 — gdb lab9-2
~/work/lab09
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov     $0x4,%eax
    0x08048085 <+5>:    mov     $0x1,%ebx
    0x0804808a <+10>:   mov     $0x8049000,%ecx
    0x0804808f <+15>:   mov     $0x8,%edx
    0x08048094 <+20>:   int     $0x80
    0x08048096 <+22>:   mov     $0x4,%eax
    0x0804809b <+27>:   mov     $0x1,%ebx
    0x080480a0 <+32>:   mov     $0x8049008,%ecx
    0x080480a5 <+37>:   mov     $0x7,%edx
    0x080480aa <+42>:   int     $0x80
    0x080480ac <+44>:   mov     $0x1,%eax
    0x080480b1 <+49>:   mov     $0x0,%ebx
    0x080480b6 <+54>:   int     $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov     eax,0x4
    0x08048085 <+5>:    mov     ebx,0x1
    0x0804808a <+10>:   mov     ecx,0x8049000
    0x0804808f <+15>:   mov     edx,0x8
    0x08048094 <+20>:   int     0x80
    0x08048096 <+22>:   mov     eax,0x4
    0x0804809b <+27>:   mov     ebx,0x1
    0x080480a0 <+32>:   mov     ecx,0x8049008
    0x080480a5 <+37>:   mov     edx,0x7
    0x080480aa <+42>:   int     0x80
    0x080480ac <+44>:   mov     eax,0x1
    0x080480b1 <+49>:   mov     ebx,0x0
    0x080480b6 <+54>:   int     0x80
End of assembler dump.
(gdb) █
```

Рисунок 2.8: Дизассимилированный код в режиме интел

Установить точку останова можно командой `break` (кратко `b`). Типичный аргумент этой команды — место установки. Его можно задать или как номер строки программы (имеет смысл, если есть исходный файл, а программа компилировалась с информацией об отладке), или как имя метки, или как адрес. Чтобы не было путаницы с номерами, перед адресом ставится «звёздочка»

На предыдущих шагах была установлена точка остановки по имени метки (`_start`). Проверил это с помощью команды `info breakpoints` (кратко `i b`). Установил еще одну точку остановки по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определил адрес предпоследней инструкции (`mov ebx,0x0`) и установил

точку.



The screenshot shows a GDB terminal window titled "umut@fedora:~/work/lab09 — gdb lab9-2". The window is divided into three main sections. The top section, titled "Register group: general", displays the values of various registers: eax (0x0), ecx (0x0), edx (0x0), ebx (0x0), esp (0xffffced0), ebp (0x0), esi (0x0), edi (0x0), and eip (0x8048080). The middle section shows assembly instructions starting from address 0x8048080, including "mov eax,0x4", "mov ebx,0x1", "mov ecx,0x8049000", "mov edx,0x8", "int 0x80", "mov eax,0x4", "mov ebx,0x1", "mov ecx,0x8049008", and "mov edx,0x7". The bottom section shows the GDB prompt and commands: "(gdb) layout regs", "(gdb) b *0x8049031", "Breakpoint 2 at 0x8049031", "(gdb) i b", and a table of breakpoints. The table has columns for Num, Type, Disp, Enb, Address, and What, showing two breakpoints at addresses 0x08048080 and 0x08049031.

```
umut@fedora:~/work/lab09 — gdb lab9-2
~/work/lab09

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffced0 0xffffced0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048080 0x8048080 <_start>

B+>0x8048080 <_start> mov    eax,0x4
0x8048085 <_start+5> mov    ebx,0x1
0x804808a <_start+10> mov    ecx,0x8049000
0x804808f <_start+15> mov    edx,0x8
0x8048094 <_start+20> int    0x80
0x8048096 <_start+22> mov    eax,0x4
0x804809b <_start+27> mov    ebx,0x1
0x80480a0 <_start+32> mov    ecx,0x8049008
0x80480a5 <_start+37> mov    edx,0x7

native process 4660 (asm) In: _start L11 PC: 0x8048080
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08048080 lab9-2.asm:11
          breakpoint already hit 1 time
2        breakpoint keep y  0x08049031
(gdb) 
```

Рисунок 2.9: Точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды `stepi` (или `si`) и проследил за изменением значений регистров.

```
umut@fedora:~/work/lab09 — gdb lab9-2
~/work/lab09

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffced0 0xffffced0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048085 0x8048085 <_start+5>

B+ 0x8048080 <_start>    mov    eax,0x4
>0x8048085 <_start+5>    mov    ebx,0x1
0x804808a <_start+10>    mov    ecx,0x8049000
0x804808f <_start+15>    mov    edx,0x8
0x8048094 <_start+20>    int    0x80
0x8048096 <_start+22>    mov    eax,0x4
0x804809b <_start+27>    mov    ebx,0x1
0x80480a0 <_start+32>    mov    ecx,0x8049008
0x80480a5 <_start+37>    mov    edx,0x7

native process 4660 (asm) In: _start L12 PC: 0x8048085
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint      keep y 0x08048080 lab9-2.asm:11
       breakpoint already hit 1 time
2      breakpoint      keep y 0x08049031
(gdb) si
(gdb) █
```

Рисунок 2.10: Изменение регистров

```
umut@fedora:~/work/lab09 — gdb lab9-2
~/work/lab09

Register group: general
eax      0x8      8
ecx      0x8049000 134516736
edx      0x8      8
ebx      0x1      1
esp      0xffffced0 0xffffced0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8048096 0x8048096 <_start+22>

B+ 0x8048080 <_start>    mov    eax,0x4
0x8048085 <_start+5>    mov    ebx,0x1
0x804808a <_start+10>   mov    ecx,0x8049000
0x804808f <_start+15>   mov    edx,0x8
0x8048094 <_start+20>   int     0x80
>0x8048096 <_start+22>   mov    eax,0x4
0x804809b <_start+27>   mov    ebx,0x1
0x80480a0 <_start+32>   mov    ecx,0x8049008
0x80480a5 <_start+37>   mov    edx,0x7

native process 4660 (asm) In: _start L16 PC: 0x8048096
(gdb) i b
Num   Type      Disp Enb Address  What
1     breakpoint keep y  0x08048080 lab9-2.asm:11
      breakpoint already hit 1 time
2     breakpoint keep y  0x08049031
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) siHello,
(gdb) █
```

Рисунок 2.11: Изменение регистров

Посмотрел значение переменной msg1 по имени. Посмотрел значение переменной msg2 по адресу.

Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Изменил первый символ переменной msg1.


```

umut@fedora:~/work/lab09 — gdb lab9-2
~/work/lab09

eax      group: general      8
eax      0x4                  4
ecx      0x8049000            134516736
edx      0x8                  8
ebx      0x1                  1
esp      0xffffced0           0xffffced0
ebp      0x0                  0
esi      0x0                  0
edi      0x0                  0
eip      0x8048094             0x8048094 <_start+20>

B+ 0x8048080 <_start>      mov     eax,0x4
0x8048085 <_start+5>      mov     ebx,0x1049000
0x804808a <_start+10>     mov     ecx,0x8049000
0x804808f <_start+15>     mov     edx,0x8
>0x8048094 <_start+20>     int     0x800x4
0x8048096 <_start+22>     mov     eax,0x4049008
0x804809b <_start+27>     mov     ecx,0x1049008
0x80480a0 <_start+32>     mov     ecx,0x8049008
0x80480a5 <_start+37>     mov     edx,0x7
nati                                          L17    PC: 0x804809b
2      process 4678 (asm) In: _start049031    L16    PC: 0x8048096
ds      process 4683 (asm) In: _start3        L15    PC: 0x8048094
0x8049000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008:      <error: Cannot access memory at address 0x804a008>
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x8049000 <msg1>:      "hello, "
(gdb) set {char}0x804a008='L'
Cannot access memory at address 0x804a008
(gdb) x/1sb 0x804a008
0x804a008:      <error: Cannot access memory at address 0x804a008>
(gdb)

```

Рисунок 2.12: Изменение значения переменной

Вывел в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx.

```
umut@fedora:~/work/lab09 — gdb lab9-2
~/work/lab09

eax      group: general      8
eax      0x4                  4
ecx      0x8049000            134516736
edx      0x8                  8
ebx      0x1                  1
esp      0xffffced0           0xffffced0
ebp      0x0                  0
esi      0x0                  0
edi      0x0                  0
eip      0x8048094             0x8048094 <_start+20>

0x8048080 <_start>      mov     eax,0x4
0x8048085 <_start+5>    mov     ebx,0x1049000
0x804808a <_start+10>   mov     ecx,0x8049000
0x804808f <_start+15>   mov     edx,0x8
>0x8048094 <_start+20>  int     0x800x4
0x8048096 <_start+22>   mov     eax,0x4049008
0x804809b <_start+27>   mov     ecx,0x1049008
0x80480a0 <_start+32>   mov     ecx,0x8049008
0x80480a5 <_start+37>   mov     edx,0x7
nati
2      process 4678 (asm) In: _start049031
ds      process 4683 (asm) In: _start3
(gdb) p/s $ecx
$3 = 134516736
(gdb) p/x $ecx
$4 = 0x8049000
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) 
```

Рисунок 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx

```

umut@fedora: ~/work/lab09 — gdb lab9-2
~/.work/lab09

eax  group: general  8
eax  0x4             4
ecx  0x8049000       134516736
edx  0x8             8
ebx  0x2             2
esp  0xffffced0      0xffffced0
ebp  0x0             0
esi  0x0             0
edi  0x0             0
eip  0x8048094        0x8048094 <_start+20>

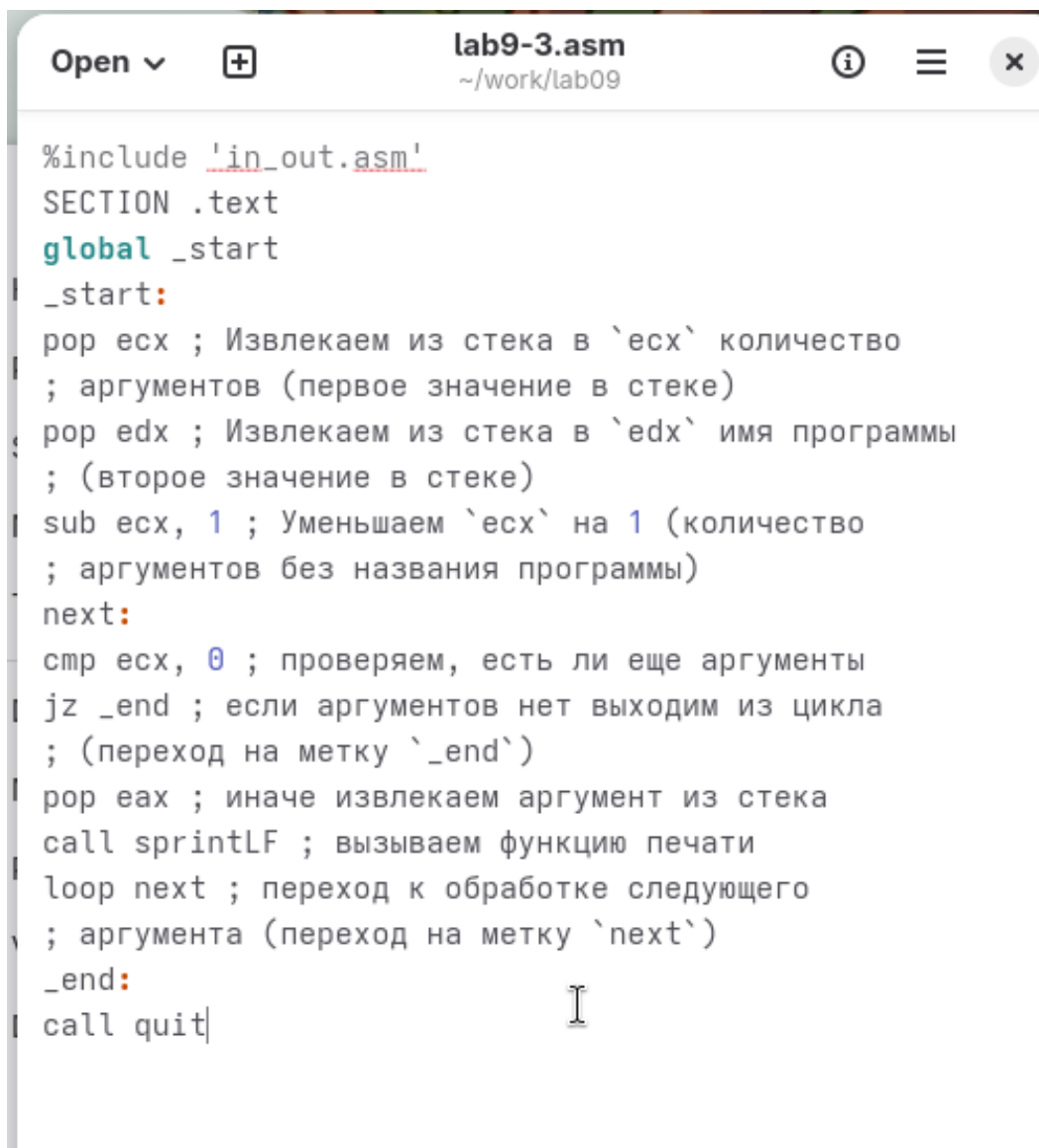
0x8048080 <_start>    mov     eax,0x4
0x8048085 <_start+5>  mov     ebx,0x1049000
0x804808a <_start+10> mov     ecx,0x8049000
0x804808f <_start+15> mov     edx,0x8
>0x8048094 <_start+20> int     0x800x4
0x8048096 <_start+22> mov     eax,0x4049008
0x804809b <_start+27> mov     ecx,0x1049008
0x80480a0 <_start+32> mov     ecx,0x8049008
0x80480a5 <_start+37> mov     edx,0x7

nati 0x80480a5 <_start+37> mov     edx,0x7
2    process 4678 (asm) In: _start049031
ds    process 4683 (asm) In: _start3
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) set $ebx='2'
(gdb) p/s $ebx
$8 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$9 = 2
(gdb)

```

Рисунок 2.14: Вывод значения регистра

Скопировал файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки. Создал исполняемый файл. Для загрузки в gdb программы с аргументами необходимо использовать ключ `-args`. Загрузил исполняемый файл в отладчик, указав аргументы.



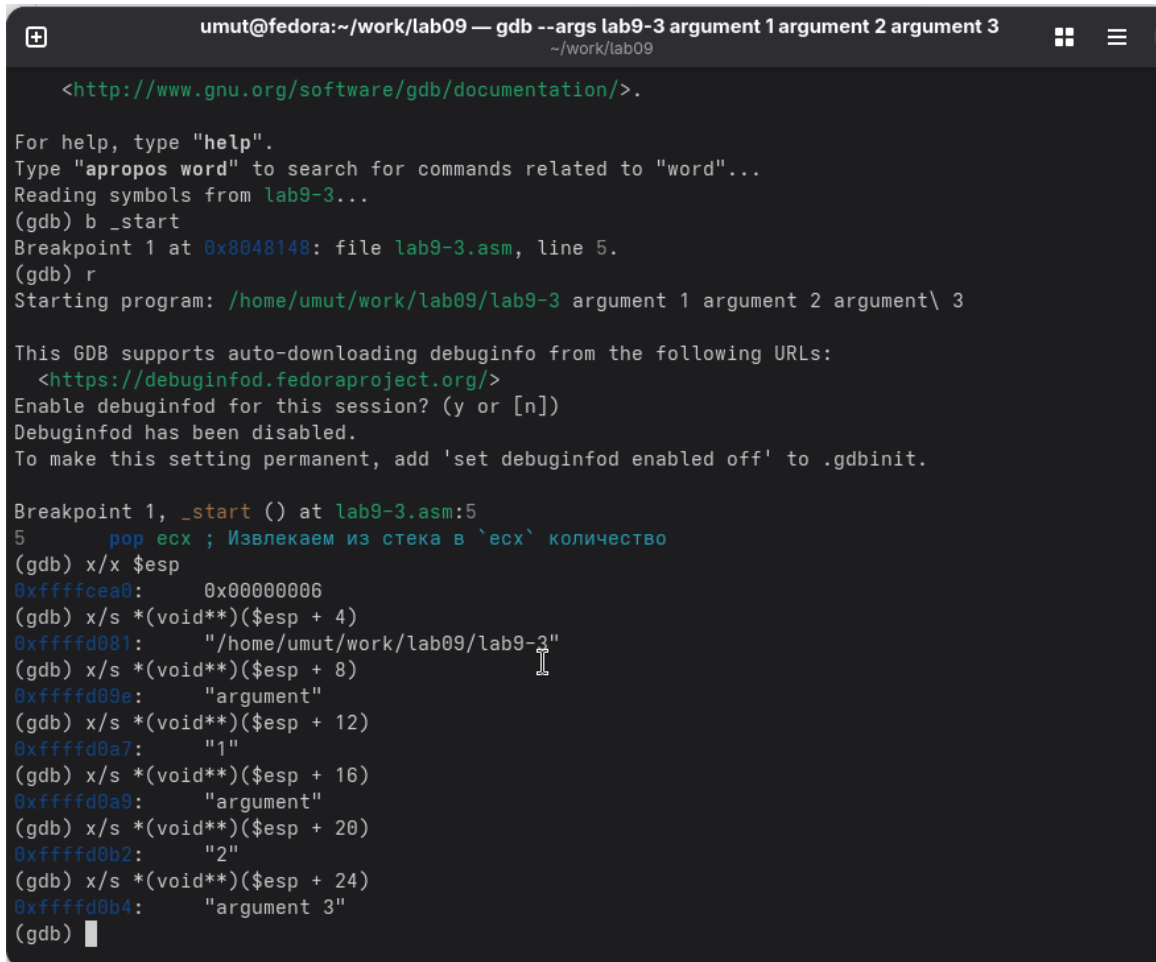
```
Open ▾ + lab9-3.asm ~/work/lab09
%include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем аргумент из стека
call sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рисунок 2.15: Программа в файле lab9-3.asm

Для начала установил точку останова перед первой инструкцией в программе и запустил ее.

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 – это имя программы lab9-3 и непосредственно аргументы: аргумент1, аргумент, 2 и „аргумент 3“.

Посмотрел остальные позиции стека – по адресу [esp+4] располагается адрес в памяти где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по адресу [esp+12] – второго и т.д.



```
umut@fedora:~/work/lab09 — gdb --args lab9-3 argument 1 argument 2 argument 3
~/work/lab09
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-3...
(gdb) b _start
Breakpoint 1 at 0x8048148: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/umut/work/lab09/lab9-3 argument 1 argument 2 argument\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

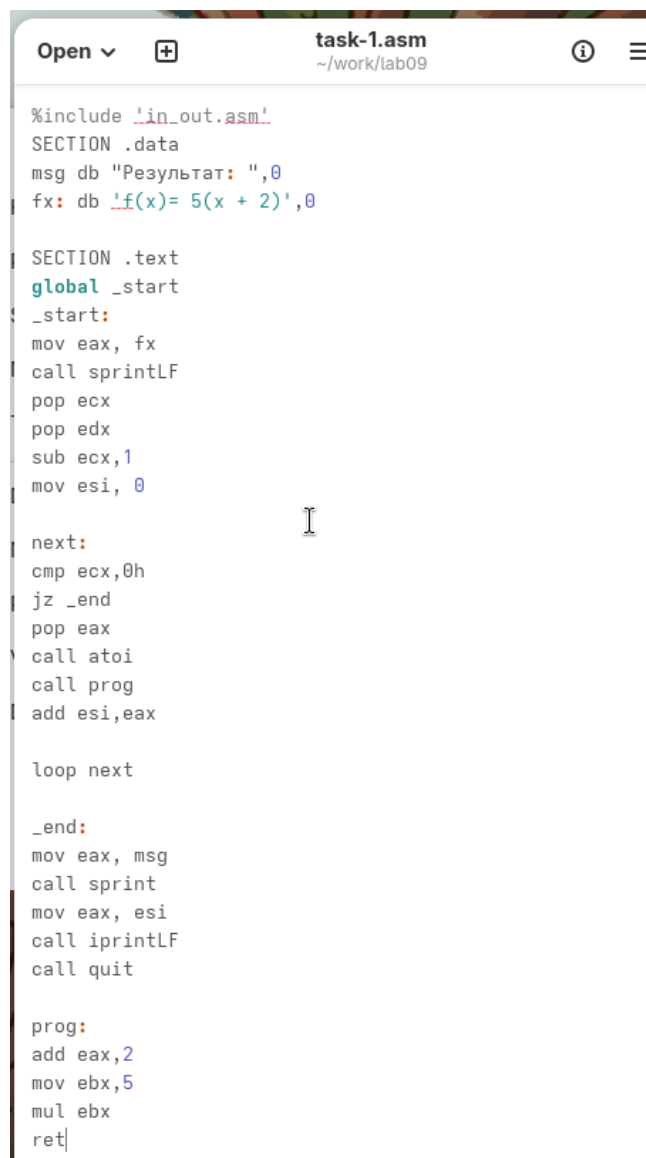
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffcea0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd081: "/home/umut/work/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd09e: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd0a7: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd0a9: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd0b2: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd0b4: "argument 3"
(gdb) █
```

Рисунок 2.16: Вывод значения регистра

Объясню, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12]) - шаг равен размеру переменной - 4 байтам.

2.3 Задание для самостоятельной работы

Я переписал программу из лабораторной работы №8, чтобы вычислить значение функции $f(x)$ в виде подпрограммы.




```
Open ▾  task-1.asm  
~/work/lab09  
  
%include 'in_out.asm'  
SECTION .data  
msg db "Результат: ",0  
fx: db 'f(x)= 5(x + 2)',0  
  
SECTION .text  
global _start  
_start:  
mov eax, fx  
call sprintf  
pop ecx  
pop edx  
sub ecx,1  
mov esi, 0  
  
next:  
cmp ecx,0h  
jz _end  
pop eax  
call atoi  
call prog  
add esi,eax  
  
loop next  
  
_end:  
mov eax, msg  
call sprintf  
mov eax, esi  
call iprintLF  
call quit  
  
prog:  
add eax,2  
mov ebx,5  
mul ebx  
ret
```

Рисунок 2.17: Программа в файле task-1.asm

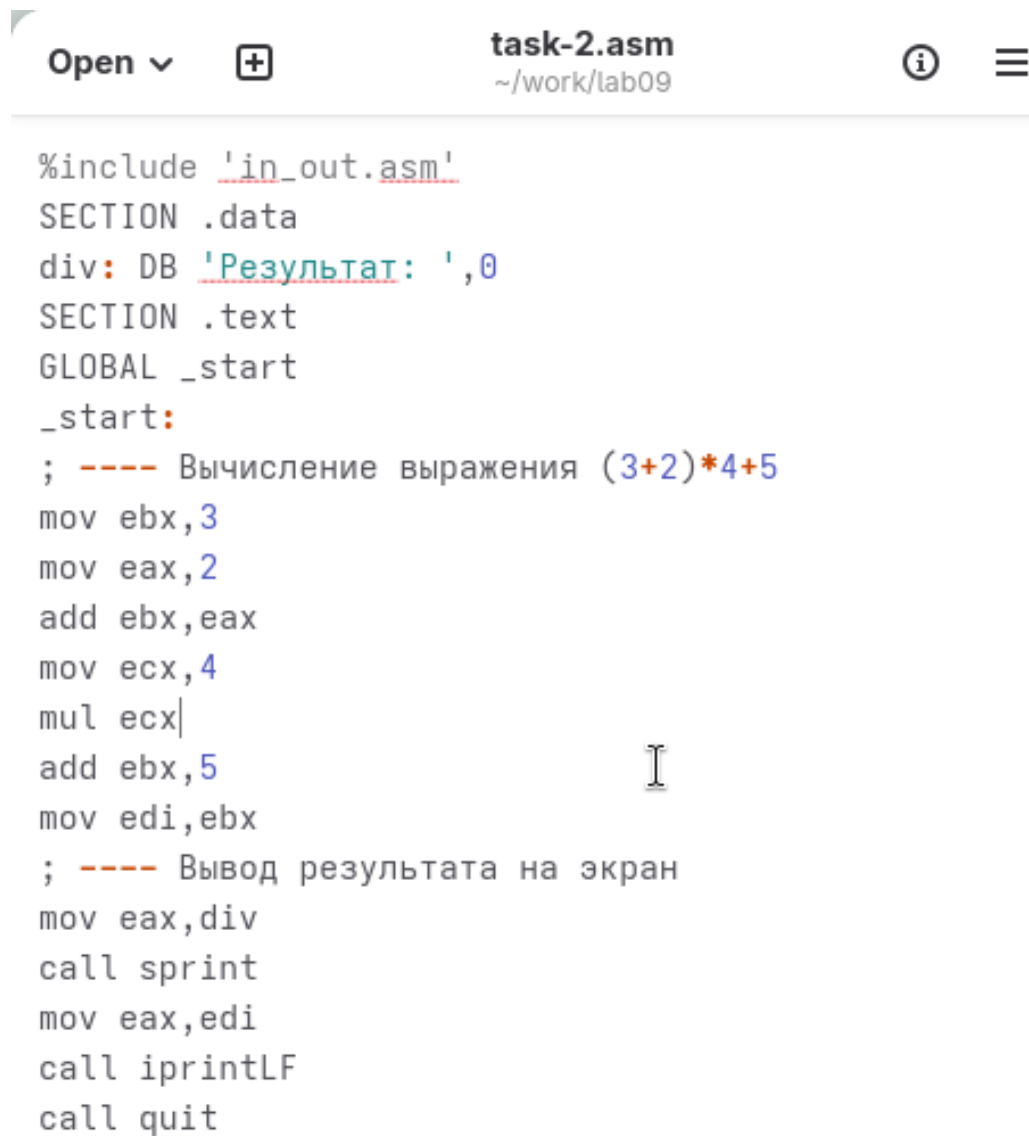
```

umut@fedora:~/work/lab09$ nasm -f elf task-1.asm
umut@fedora:~/work/lab09$ ld -m elf_i386 task-1.o -o task-1
umut@fedora:~/work/lab09$ ./task-1
f(x)= 5(x + 2)
Результат: 0
umut@fedora:~/work/lab09$ ./task-1 1
f(x)= 5(x + 2)
Результат: 15
umut@fedora:~/work/lab09$ ./task-1 1 3 4 5 6
f(x)= 5(x + 2)
Результат: 145
umut@fedora:~/work/lab09$ █

```

Рисунок 2.18: Запуск программы task-1.asm

Приведенный ниже листинг программы вычисляет выражение $(3+2)*4+5$. Однако, при запуске, программа дает неверный результат. Я проверил это и решил использовать отладчик GDB для анализа изменений значений регистров и определения ошибки.






```
Open ▾  task-2.asm  
~/work/lab09    
  
%include 'in_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
SECTION .text  
GLOBAL _start  
_start:  
; ---- Вычисление выражения (3+2)*4+5  
mov ebx,3  
mov eax,2  
add ebx,eax  
mov ecx,4  
mul ecx  
add ebx,5  
mov edi,ebx  
; ---- Вывод результата на экран  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
call quit
```

Рисунок 2.19: Код с ошибкой в файле task-2.asm

The screenshot shows a GDB window titled "umut@fedora:~/work/lab09 — gdb task-2". The top panel displays register values: `eax` is 8, `edi` is 0, and `ecx` is fffced0. Below this, a list of assembly instructions is shown with their addresses and disassembled code:

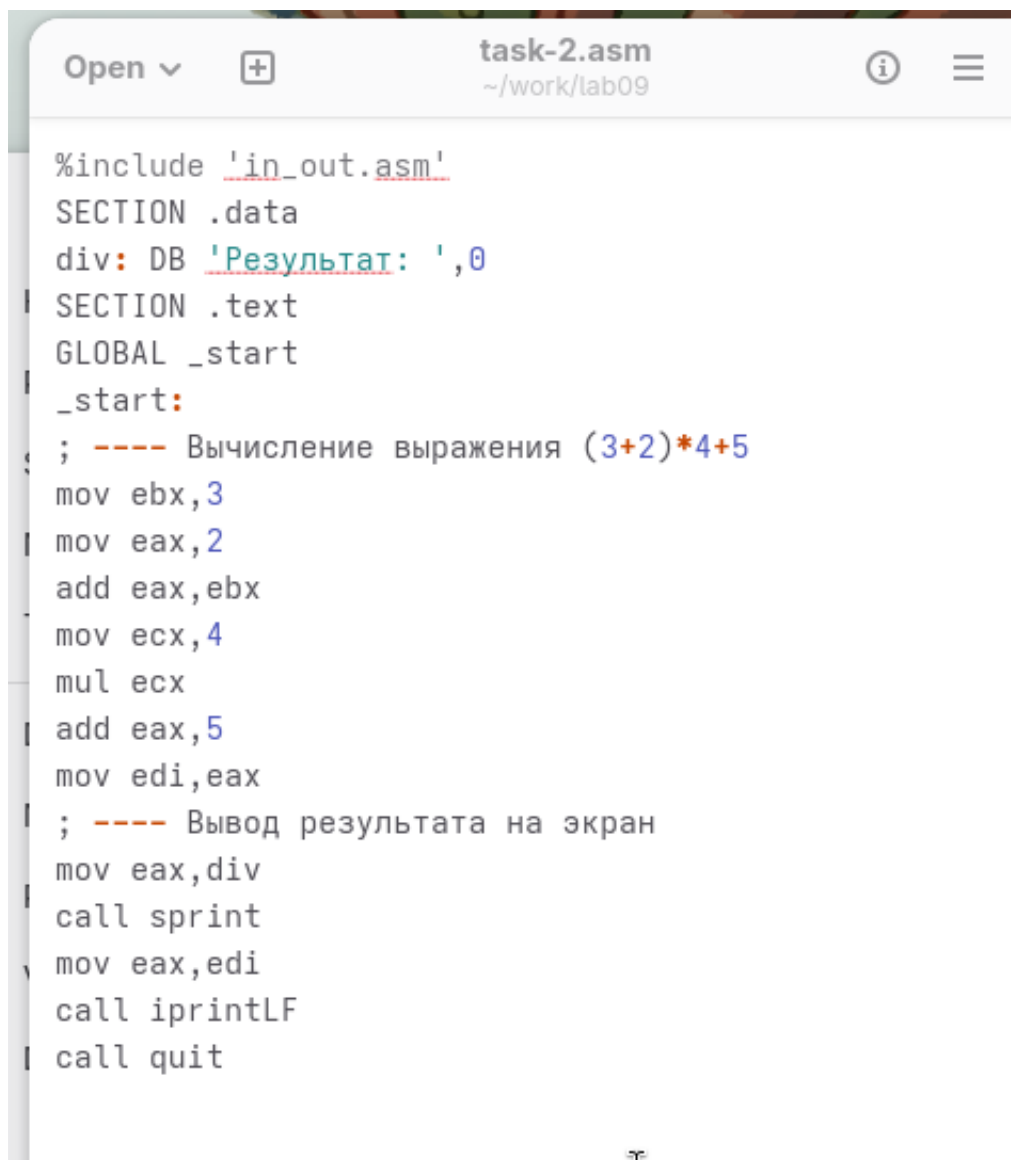
```
0x8048180 <_start+24>  mul    eax,0x8049000
0x804818a <_start+34>  mov    eax,edi
0x804818c <_start+36>  call   0x8048106 <iprintLF>
0x8048191 <_start+41>  call   0x804815b <quit>
```

The bottom panel shows the GDB command line with the following commands and output:

```
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) cРезультат: 10
Continuing.
[Inferior 1 (process 4857) exited normally]
(gdb)
```

Рисунок 2.20: Отладка task-2.asm

Я заметил, что порядок аргументов в инструкции `add` был перепутан и что при завершении работы, вместо `eax`, значение отправлялось в `edi`. Вот исправленный код программы:




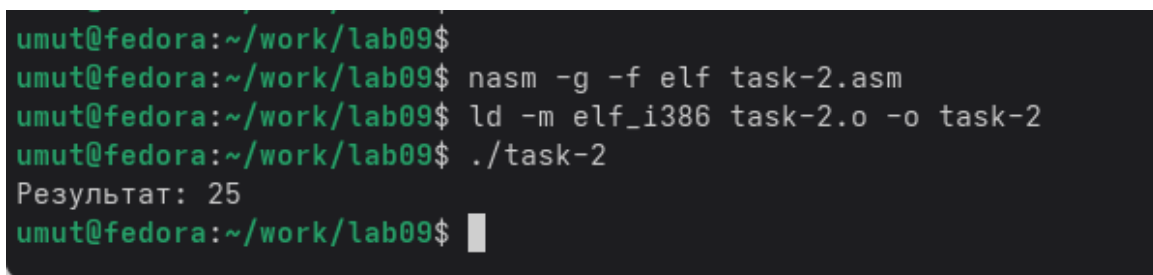
```
Open ▾  task-2.asm  
~/work/lab09  
  
%include 'in_out.asm'  
SECTION .data  
div: DB 'Результат: ',0  
SECTION .text  
GLOBAL _start  
_start:  
; ---- Вычисление выражения (3+2)*4+5  
mov ebx,3  
mov eax,2  
add eax,ebx  
mov ecx,4  
mul ecx  
add eax,5  
mov edi,eax  
; ---- Вывод результата на экран  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
call quit
```

Рисунок 2.21: Код исправлен в файле task-2.asm



```
umut@fedora:~/work/lab09$  
umut@fedora:~/work/lab09$ nasm -g -f elf task-2.asm  
umut@fedora:~/work/lab09$ ld -m elf_i386 task-2.o -o task-2  
umut@fedora:~/work/lab09$ ./task-2  
Результат: 25  
umut@fedora:~/work/lab09$
```

Рисунок 2.22: Проверка работы task-2.asm

3 Выводы

Освоили работу с подпрограммами и отладчиком.