

Отчёт по лабораторной работе №9

Ярослав Антонович Меркулов

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
2.1	Реализация подпрограмм в NASM	5
2.2	Отладка программ с помощью GDB	9
2.2.1	Добавление точек остановки	14
2.2.2	Работа с данными программы в GDB	14
2.2.3	Обработка аргументов командной строки в GDB	19
2.3	Выполнение самостоятельной работы	22
3	Выводы	27

Список иллюстраций

2.1	Создание каталога, переход в него, создание файла	5
2.2	Введённый текст программы	6
2.3	Работа программы	7
2.4	Изменённый файл	8
2.5	Работа новой программы	9
2.6	Создание файла	9
2.7	Lab09-2.asm	10
2.8	Компиляция и трансляция с параметрами	10
2.9	Отладчик	11
2.10	Брейкпоинт	12
2.11	Дисассимилированный код	13
2.12	Режим псевдографики	13
2.13	Точки останова	14
2.14	Содержимое регистров	15
2.15	Инструкции si	15
2.16	Просмотр значений переменных	16
2.17	Изменение значений	17
2.18	Значения регистра в разных регистрах	17
2.19	Изменение значений регистра	18
2.20	Завершение программы	19
2.21	Копирование файла	20
2.22	Запуск через gdb	21
2.23	Просмотр элементов стека	22
2.24	Изменённая программа	23
2.25	Работа программы	23
2.26	Заданный текст	24
2.27	Неправильная работа программы	24
2.28	Проблема	25
2.29	Правильная программа	25
2.30	Работа программы	26

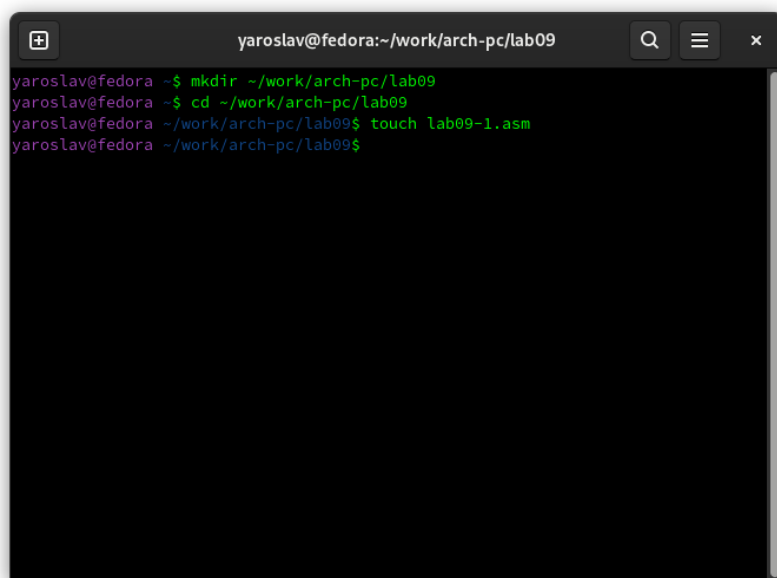
1 Цель работы

Приобрести навыки написания программ с использованием подпрограмм.
Ознакомиться с методами отладки при помощи GDB.

2 Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM

1. Создаём каталог для лабораторной работы, переходим в него и создаём файл lab09-1.asm.



```
yaroslav@fedora:~/work/arch-pc/lab09
yaroslav@fedora ~$ mkdir ~/work/arch-pc/lab09
yaroslav@fedora ~$ cd ~/work/arch-pc/lab09
yaroslav@fedora ~/work/arch-pc/lab09$ touch lab09-1.asm
yaroslav@fedora ~/work/arch-pc/lab09$
```

Рис. 2.1: Создание каталога, переход в него, создание файла

2. Вводим текст программы из листинга 9.1.

```
res: 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

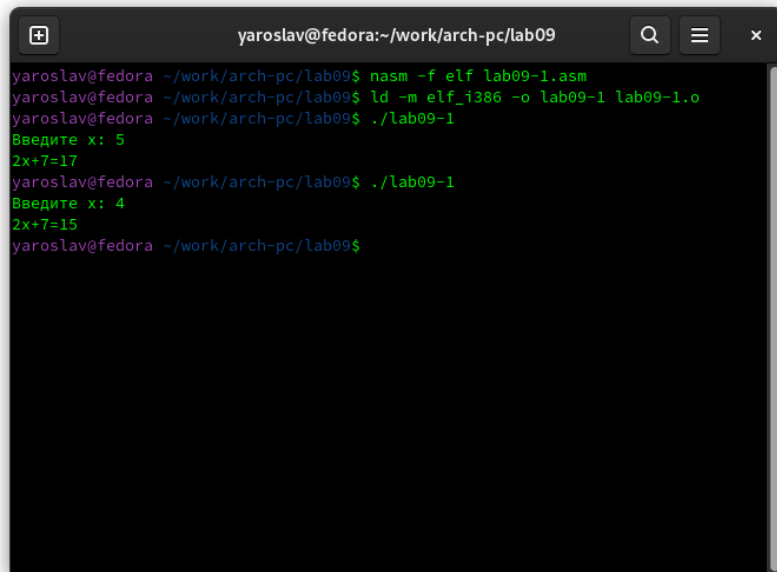
    mov eax, result
    call sprint
    mov eax, [res]
    call iprintLF

    call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax

    ret ; выход из подпрограммы
```

Рис. 2.2: Введённый текст программы

3. Создаём исполняемый файл и тестируем.



```
yaroslav@fedora ~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
yaroslav@fedora ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
yaroslav@fedora ~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2x+7=17
yaroslav@fedora ~/work/arch-pc/lab09$ ./lab09-1
Введите x: 4
2x+7=15
yaroslav@fedora ~/work/arch-pc/lab09$
```

Рис. 2.3: Работа программы

4. Изменяем текст программы, создаём подпрограмму `_subcalcul`.

```
lab09-1.asm
~/work/arch-pc/lab09

lab09-4.asm | in_out.asm | lab09-5.asm | • report9.md | lab09-1.asm x

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, [res]
    call iprintLF

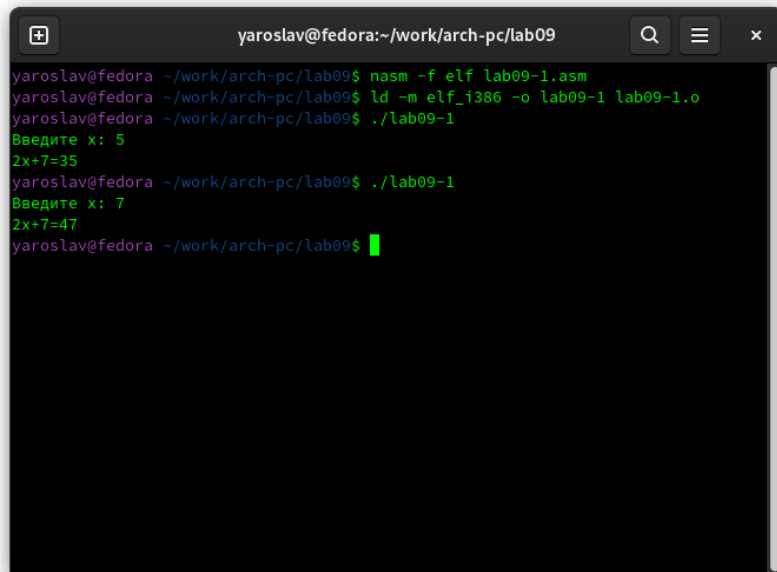
    call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
    call _subcalcul
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax

    ret ; выход из подпрограммы

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

    ret
```

Рис. 2.4: Изменённый файл

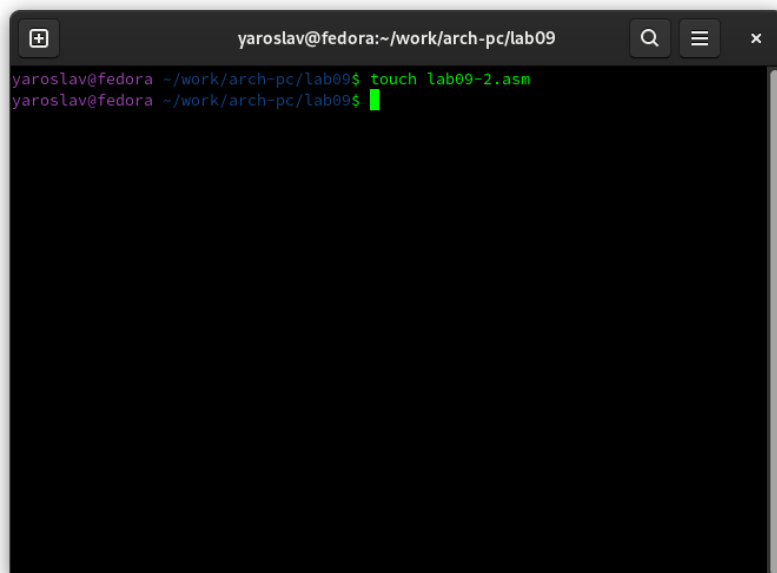


```
yaroslav@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
yaroslav@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
yaroslav@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2x+7=35
yaroslav@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 7
2x+7=47
yaroslav@fedora:~/work/arch-pc/lab09$
```

Рис. 2.5: Работа новой программы

2.2 Отладка программ с помощью GDB

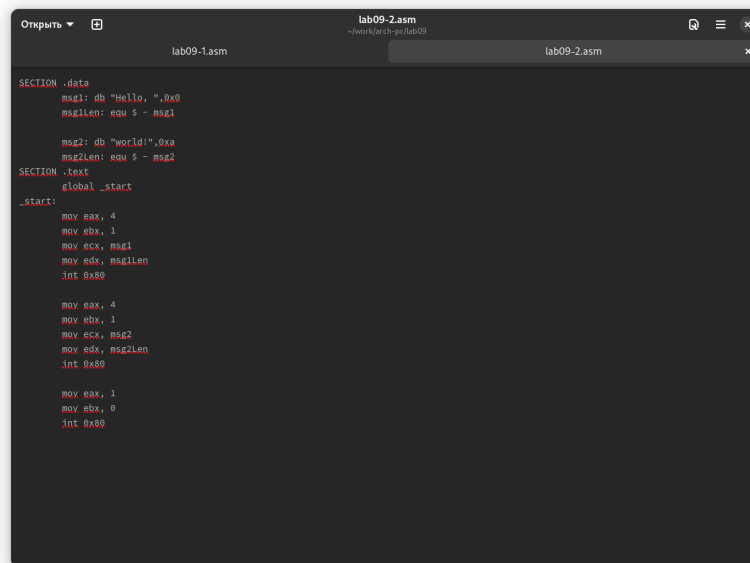
5. Создаём файл lab09-2.asm.



```
yaroslav@fedora:~/work/arch-pc/lab09$ touch lab09-2.asm
yaroslav@fedora:~/work/arch-pc/lab09$
```

Рис. 2.6: Создание файла

6. Записываем в файл листинг.



```
SECTION .data
    msg1: db "Hello, ",0x0
    msg1len: equ $ - msg1

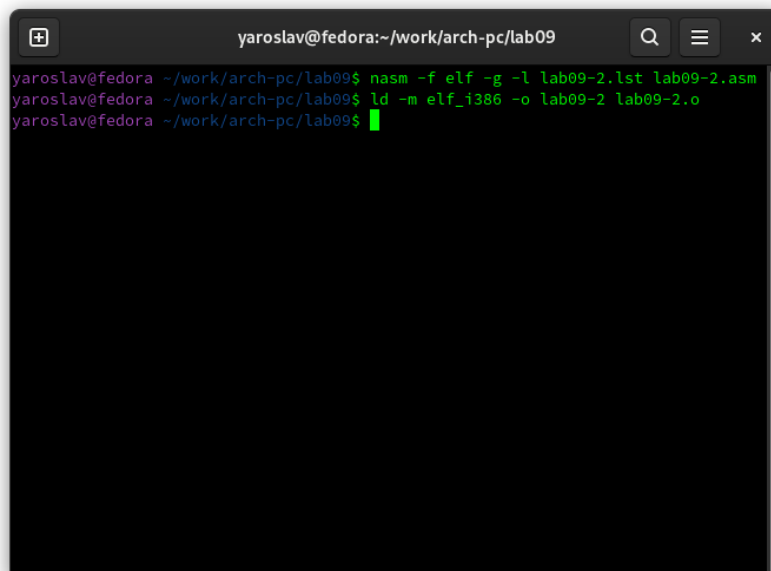
    msg2: db "world!",0xa
    msg2len: equ $ - msg2
SECTION .text
    global _start
_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, msg1len
    int 0x80

    mov eax, 4
    mov ebx, 1
    mov ecx, msg2
    mov edx, msg2len
    int 0x80

    mov eax, 1
    mov ebx, 0
    int 0x80
```

Рис. 2.7: Lab09-2.asm

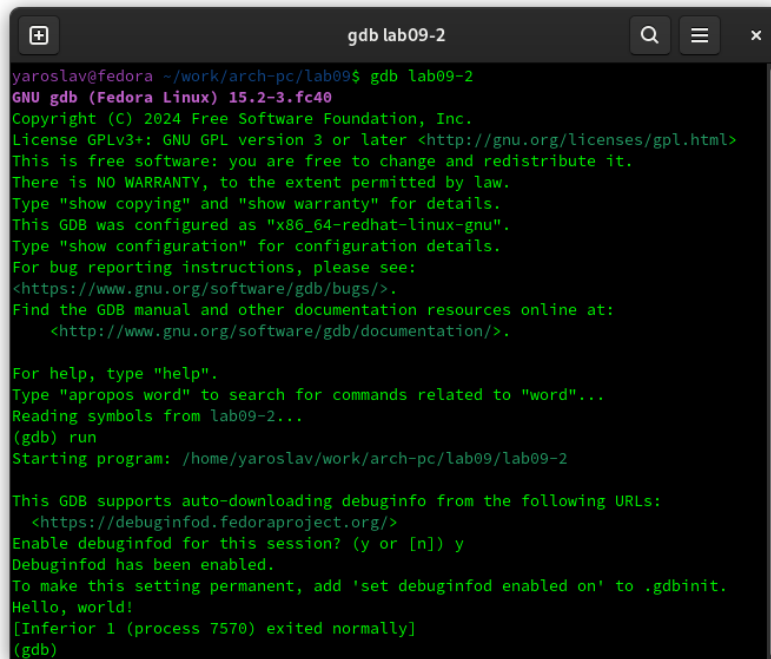
7. Получаем исполняемый файл.



```
yaroslav@fedora: ~/work/arch-pc/lab09
yaroslav@fedora ~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
yaroslav@fedora ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
yaroslav@fedora ~/work/arch-pc/lab09$
```

Рис. 2.8: Компиляция и трансляция с параметрами

8. Запускаем отладчик, запускаем в нём программу.



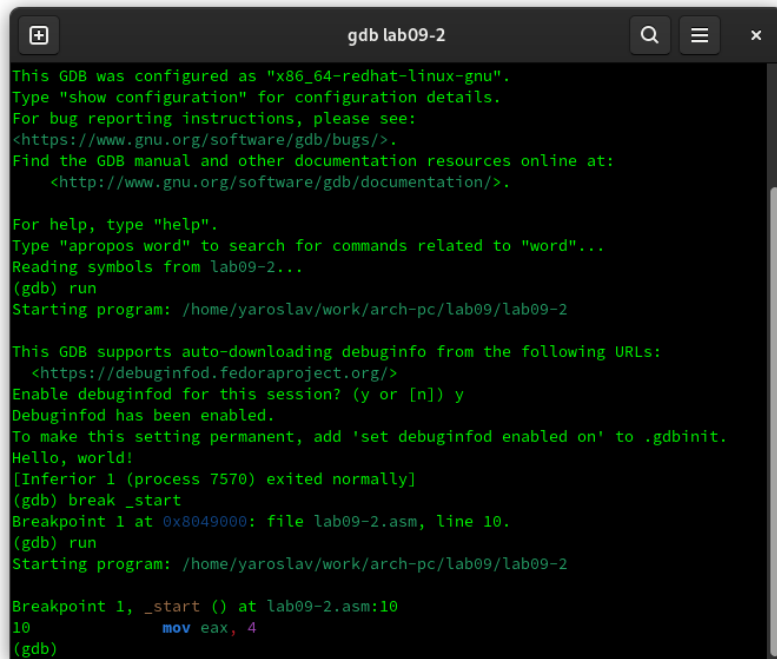
```
yaroslav@fedora ~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 15.2-3.fc40
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, 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 lab09-2...
(gdb) run
Starting program: /home/yaroslav/work/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
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.
Hello, world!
[Inferior 1 (process 7570) exited normally]
(gdb)
```

Рис. 2.9: Отладчик

9. Ставим breakpoint на `_start`.



```
gdb lab09-2

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/>.

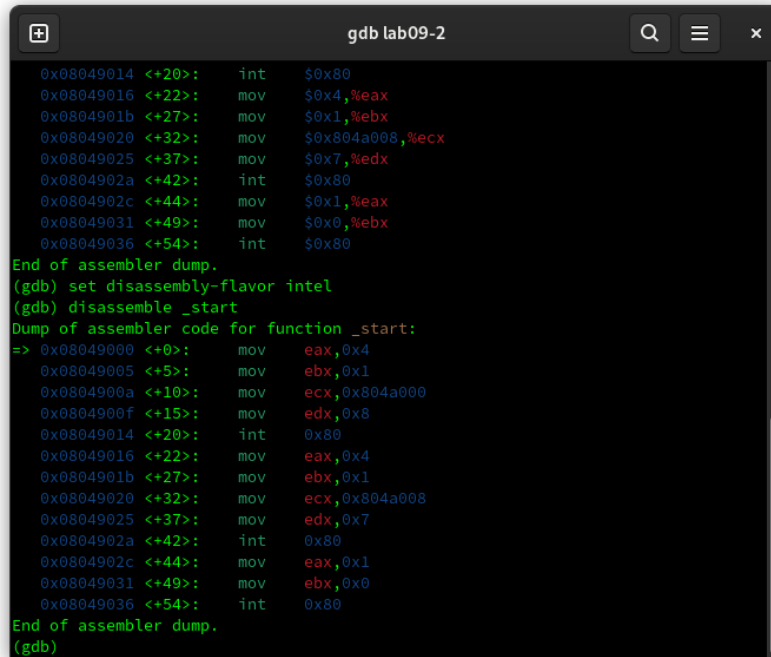
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /home/yaroslav/work/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
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.
Hello, world!
[Inferior 1 (process 7570) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 10.
(gdb) run
Starting program: /home/yaroslav/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:10
10      mov eax, 4
(gdb)
```

Рис. 2.10: Брейкпоинт

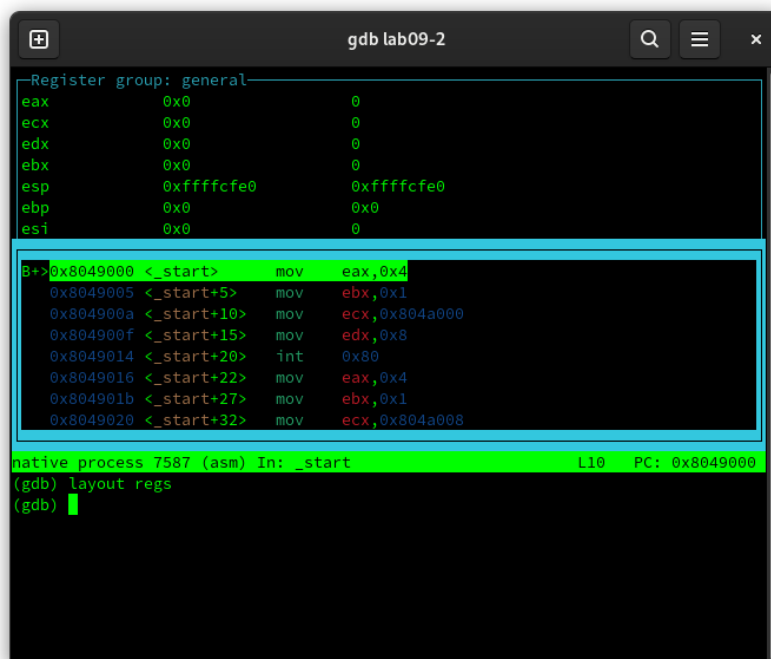
10. Смотрим дисассимилированный код программы, переключаем отображение.



```
gdb lab09-2
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)
```

Рис. 2.11: Дисассимилированный код

11. Включаем режим псевдографики.



```
gdb lab09-2
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0

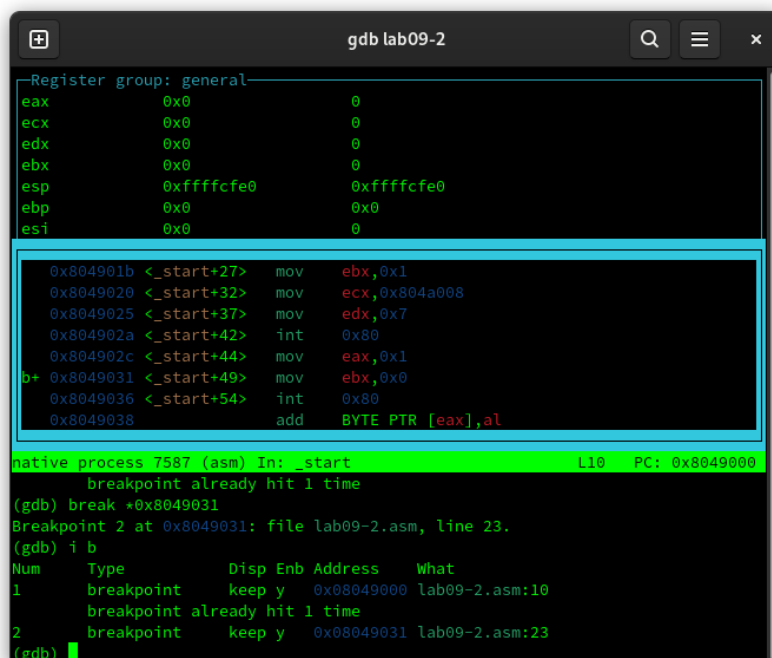
B->0x8049000 <_start> mov eax,0x4
0x8049005 <_start+5> mov ebx,0x1
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008

native process 7587 (asm) In: _start L10 PC: 0x8049000
(gdb) layout regs
(gdb)
```

Рис. 2.12: Режим псевдографики

2.2.1 Добавление точек останова

12. Ставим точку останова и выводим о них. Видим 2 точки: первая, которую мы поставили в пункте 9, и новую.



The screenshot shows the GDB interface for a program named 'lab09-2'. The top panel displays the 'Register group: general' with values for eax, ecx, edx, ebx, esp, ebp, and esi. The middle panel shows assembly code with breakpoints set at various instructions. The bottom panel shows the status of the breakpoints, indicating that breakpoint 2 at address 0x8049031 has been added.

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0

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

native process 7587 (asm) In: _start      L10      PC: 0x8049000
breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 23.
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab09-2.asm:10
          breakpoint already hit 1 time
2        breakpoint keep y  0x08049031 lab09-2.asm:23
(gdb)
```

Рис. 2.13: Точки останова

2.2.2 Работа с данными программы в GDB

13. Просматриваем содержимое регистров.

```

gdb lab09-2

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0

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

native process 7587 (asm) In: _start L10 PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 2.14: Содержимое регистров

14. Выполняем инструкции stepi (si).

```

gdb lab09-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0

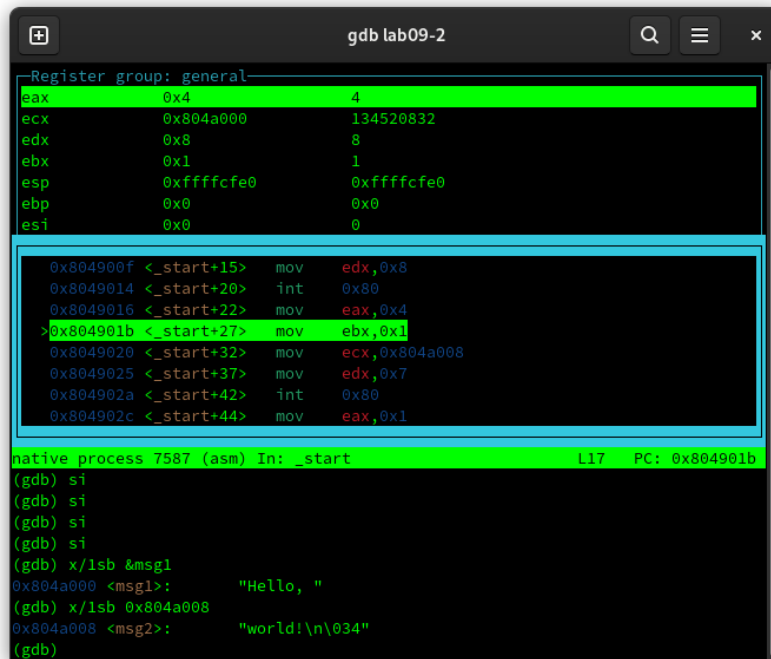
B+ 0x8049000 <_start>      mov     eax,0x4
0x8049005 <_start+5>      mov     ebx,0x1
0x804900a <_start+10>     mov     ecx,0x804a000
0x804900f <_start+15>     mov     edx,0x8
0x8049014 <_start+20>     int     0x80
>0x8049016 <_start+22>     mov     eax,0x4
0x804901b <_start+27>     mov     ebx,0x1
0x8049020 <_start+32>     mov     ecx,0x804a008

native process 7587 (asm) In: _start L16 PC: 0x8049016
es       0x2b      43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 2.15: Инструкции si

15. Смотрим значение переменной по имени и адресу.



```
gdb lab09-2

Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0

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

native process 7587 (asm) In: _start      L17      PC: 0x804901b
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)
```

Рис. 2.16: Просмотр значений переменных

16. Изменяем значения у переменных (букву).


```

gdb lab09-2

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0

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

native process 4313 (asm) In: _start L10 PC: 0x8049000
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='z'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "zorld!\n\034"
(gdb)

```

Рис. 2.17: Изменение значений

17. Просматриваем регистр edx в hex,bin,char.

```

gdb lab09-2

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0

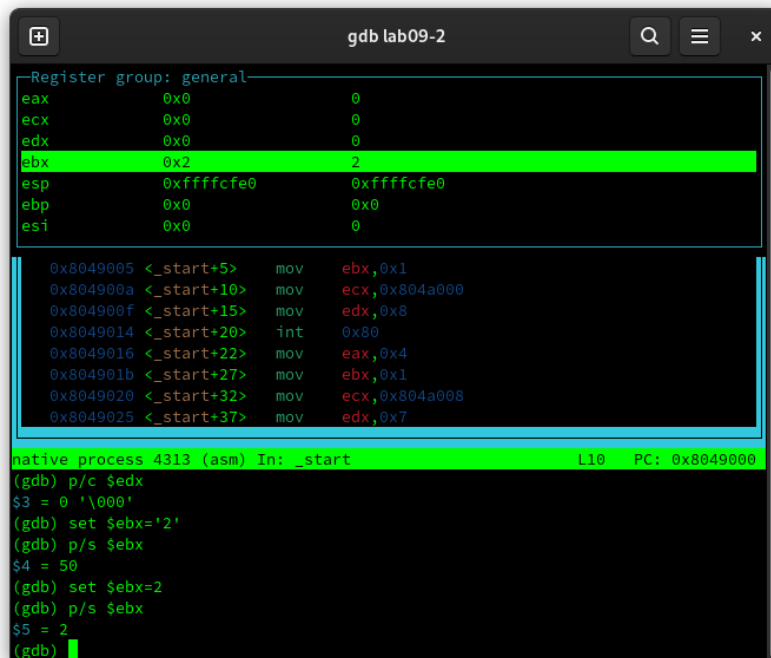
0x8049005 <_start+5>  mov     ebx,0x1
0x804900a <_start+10> mov     ecx,0x804a000
0x804900f <_start+15> mov     edx,0x8
0x8049014 <_start+20> int     0x80
0x8049016 <_start+22> mov     eax,0x4
0x804901b <_start+27> mov     ebx,0x1
0x8049020 <_start+32> mov     ecx,0x804a008
0x8049025 <_start+37> mov     edx,0x7

native process 4313 (asm) In: _start L10 PC: 0x8049000
--Type <RET> for more, q to quit, c to continue without paging--
Quit
(gdb) p/x $edx
$1 = 0x0
(gdb) p/t $edx
$2 = 0
(gdb) p/c $edx
$3 = 0 '\000'
(gdb)

```

Рис. 2.18: Значения регистра в разных регистрах

18. Меняем значения регистра `ebx`. В первый раз мы ввели вернулся номер по таблице ASCII у символа '2'. Во второй раз само число.



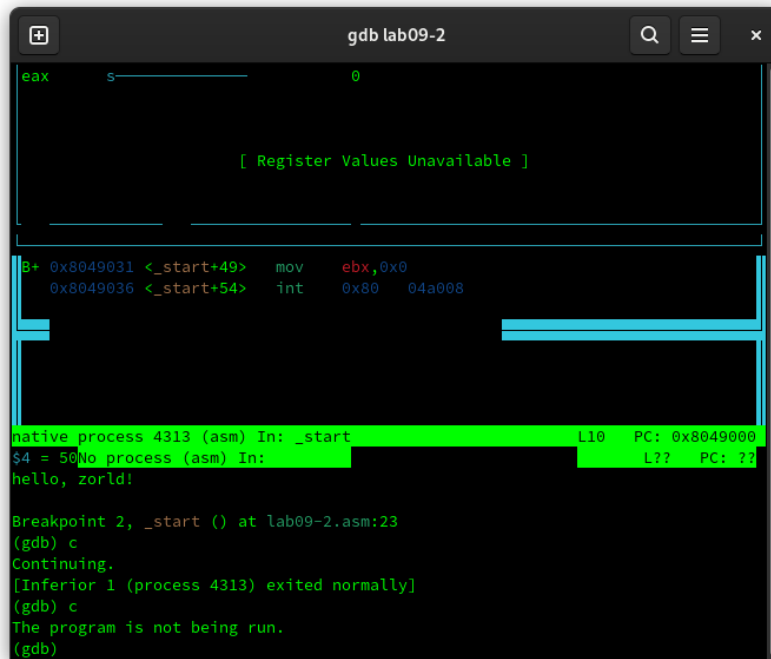
```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x2      2
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0
esi      0x0      0

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

native process 4313 (asm) In: _start L10 PC: 0x8049000
(gdb) p/c $edx
$3 = 0 '\000'
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb)
```

Рис. 2.19: Изменение значений регистра

19. Завершаем выполнение программы и выходим через `quit`.



```
gdb lab09-2

eax  s-----  0

[ Register Values Unavailable ]

B+ 0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80 04a008

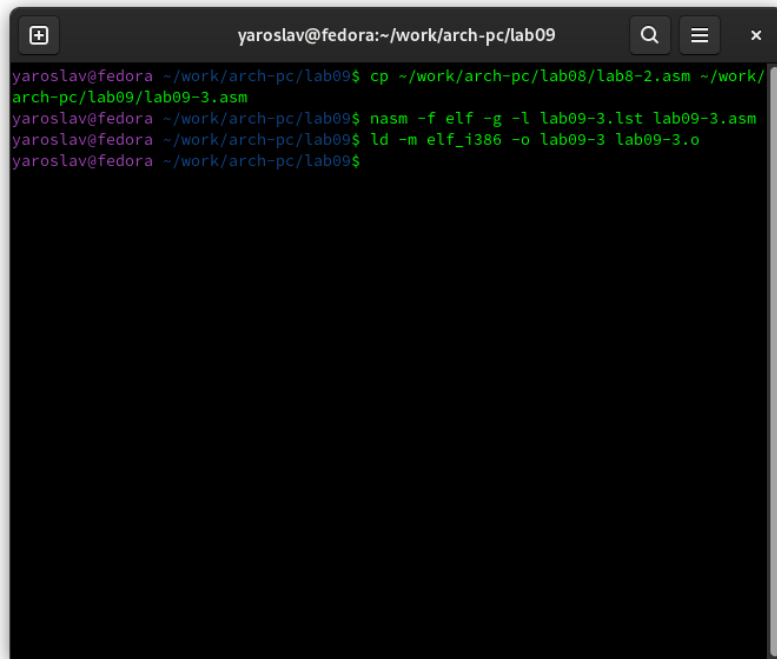
native process 4313 (asm) In: _start L10 PC: 0x8049000
$4 = 50No process (asm) In: L?? PC: ??
hello, zorld!

Breakpoint 2, _start () at lab09-2.asm:23
(gdb) c
Continuing.
[Inferior 1 (process 4313) exited normally]
(gdb) c
The program is not being run.
(gdb)
```

Рис. 2.20: Завершение программы

2.2.3 Обработка аргументов командной строки в GDB

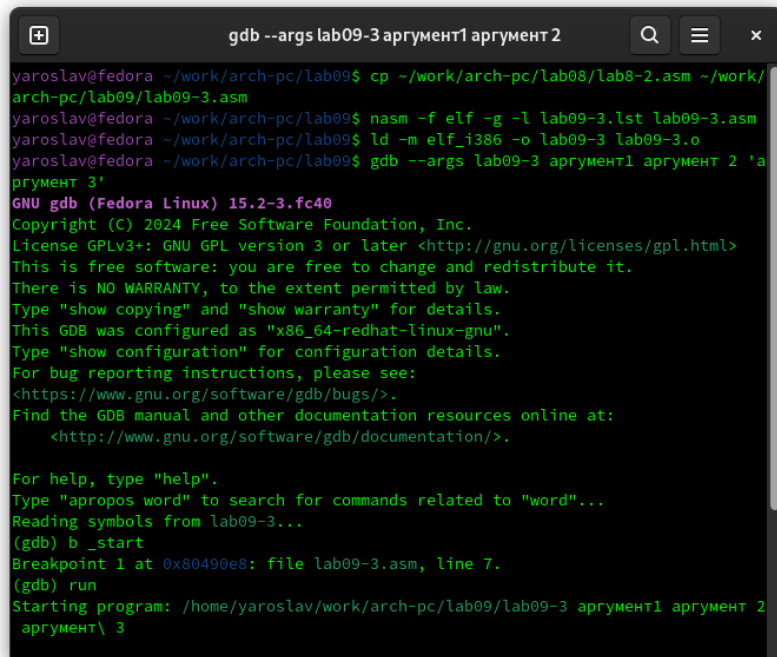
20. Копируем файл с предыдущей лабораторной работы. Создаём исполняемый файл.

A terminal window with a dark background and light-colored text. The window title is 'yaroslav@fedora:~/work/arch-pc/lab09'. The terminal shows four lines of commands and their outputs. The first line copies a file from lab08 to lab09. The second line runs the nasm assembler. The third line runs the ld linker. The fourth line shows the prompt again.

```
yaroslav@fedora ~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/
arch-pc/lab09/lab09-3.asm
yaroslav@fedora ~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
yaroslav@fedora ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
yaroslav@fedora ~/work/arch-pc/lab09$
```

Рис. 2.21: Копирование файла

21. Загружаем программу с аргументами в gdb, ставим точку остановки и за-пускаем.

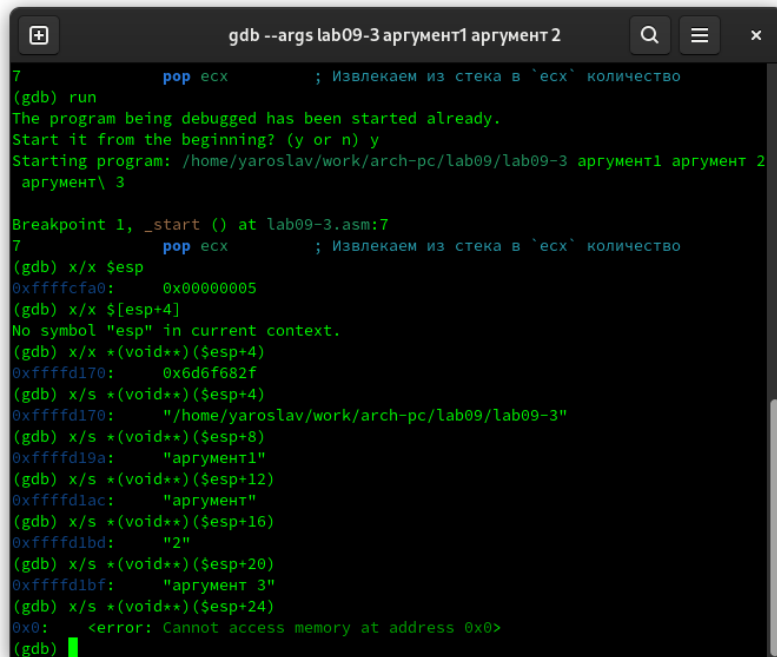


```
gdb --args lab09-3 аргумент1 аргумент 2
yaroslav@fedora ~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
yaroslav@fedora ~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
yaroslav@fedora ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
yaroslav@fedora ~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'а
ргумент 3'
GNU gdb (Fedora Linux) 15.2-3.fc40
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 lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 7.
(gdb) run
Starting program: /home/yaroslav/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2
аргумент\ 3
```

Рис. 2.22: Запуск через gdb

22. Смотрим позиции стека. Разница в 4 обусловлена размером каждой позиции в 4 байта.



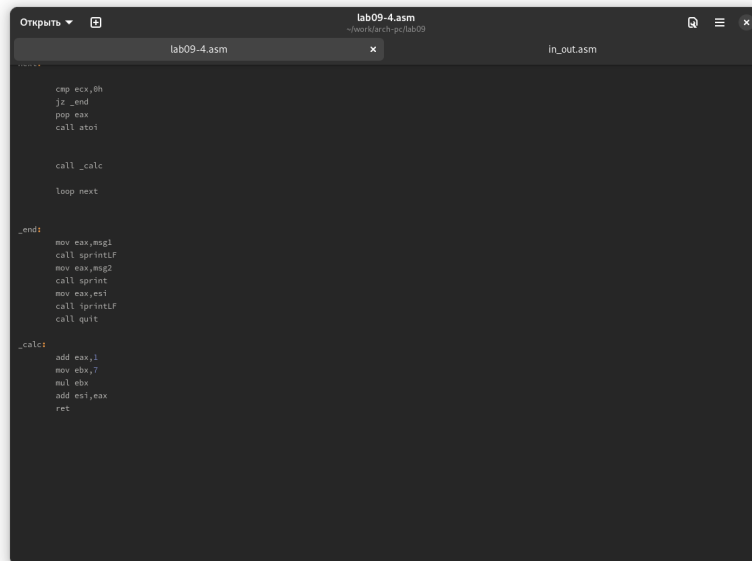
```
gdb --args lab09-3 аргумент1 аргумент 2
7      pop     ecx          ; Извлекаем из стека в `ecx` количество
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/yaroslav/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2
аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:7
7      pop     ecx          ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffcfa0: 0x00000005
(gdb) x/x $(esp+4)
No symbol "esp" in current context.
(gdb) x/x *(void**)(esp+4)
0xffffd170: 0xd6f682f
(gdb) x/s *(void**)(esp+4)
0xffffd170: "/home/yaroslav/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp+8)
0xffffd19a: "аргумент1"
(gdb) x/s *(void**)(esp+12)
0xffffd1ac: "аргумент"
(gdb) x/s *(void**)(esp+16)
0xffffd1bd: "2"
(gdb) x/s *(void**)(esp+20)
0xffffd1bf: "аргумент 3"
(gdb) x/s *(void**)(esp+24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.23: Просмотр элементов стека

2.3 Выполнение самостоятельной работы

1. Меняем текст предыдущего задания. Отправляем вычисление функции в отдельную подпрограмму `_calc`.



The screenshot shows a code editor window titled 'lab09-4.asm' with a file explorer on the left showing 'lab09-4.asm' and 'in_out.asm'. The assembly code is as follows:

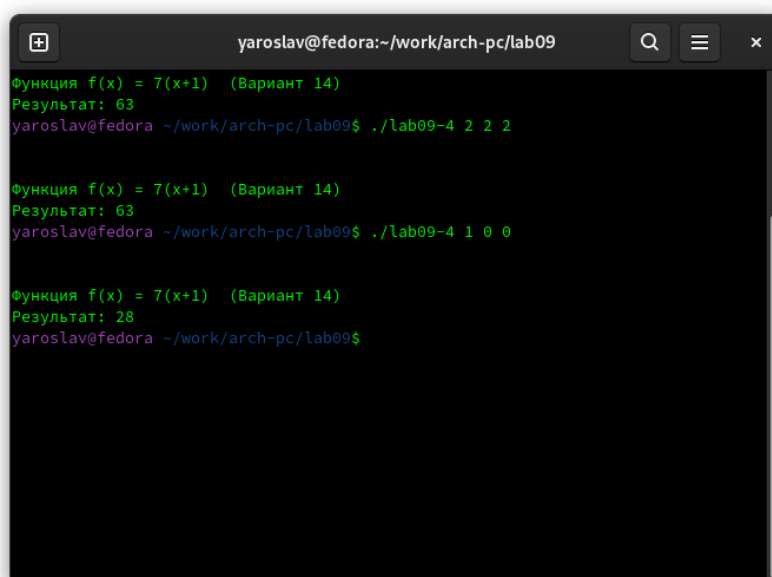
```
cap ecx,0h
jp _end
pop eax
call atoi

call _calc
loop next

_end:
mov eax,msg1
call sprintf
mov eax,msg2
call sprintf
mov eax,esi
call sprintf
call quit

_calc:
add eax,1
mov ebx,7
mul ebx
add esi,eax
ret
```

Рис. 2.24: Измененная программа



The screenshot shows a terminal window with the prompt 'yaroslav@fedora:~/work/arch-pc/lab09'. It displays the output of the program for three different input cases:

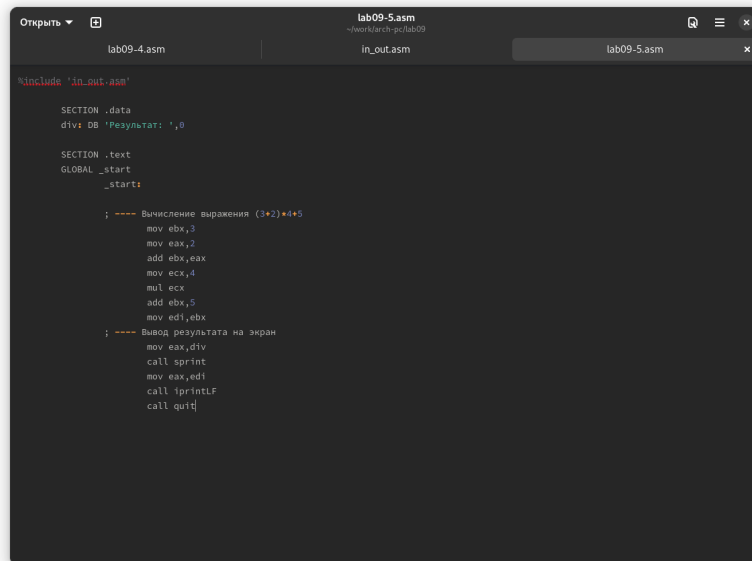
```
Функция f(x) = 7(x+1) (Вариант 14)
Результат: 63
yaroslav@fedora ~/work/arch-pc/lab09$ ./lab09-4 2 2 2

Функция f(x) = 7(x+1) (Вариант 14)
Результат: 63
yaroslav@fedora ~/work/arch-pc/lab09$ ./lab09-4 1 0 0

Функция f(x) = 7(x+1) (Вариант 14)
Результат: 28
yaroslav@fedora ~/work/arch-pc/lab09$
```

Рис. 2.25: Работа программы

2. Пишем в файл заданный листинг.



```
lab09-5.asm
~/work/arch-pc/lab09

lab09-4.asm | in_out.asm | lab09-5.asm x

#include "in_out.asm"

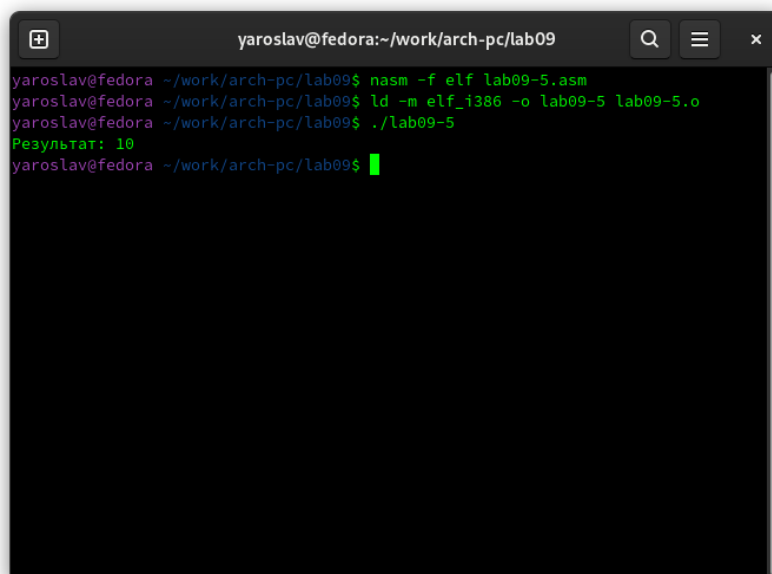
SECTION .data
div: DB "Результат: ",0

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 printf
mov eax,edi
call sprintf
call quit
```

Рис. 2.26: Заданный текст



```
yaroslav@fedora:~/work/arch-pc/lab09
yaroslav@fedora ~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
yaroslav@fedora ~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
yaroslav@fedora ~/work/arch-pc/lab09$ ./lab09-5
Результат: 10
yaroslav@fedora ~/work/arch-pc/lab09$
```

Рис. 2.27: Неправильная работа программы

3. Запускаем отладчик и находим ошибку. Команда `mul` умножает на значение регистра и перезаписывает результат в `eax` (а не в `ebx`). Меняем `ebx` на `eax`.

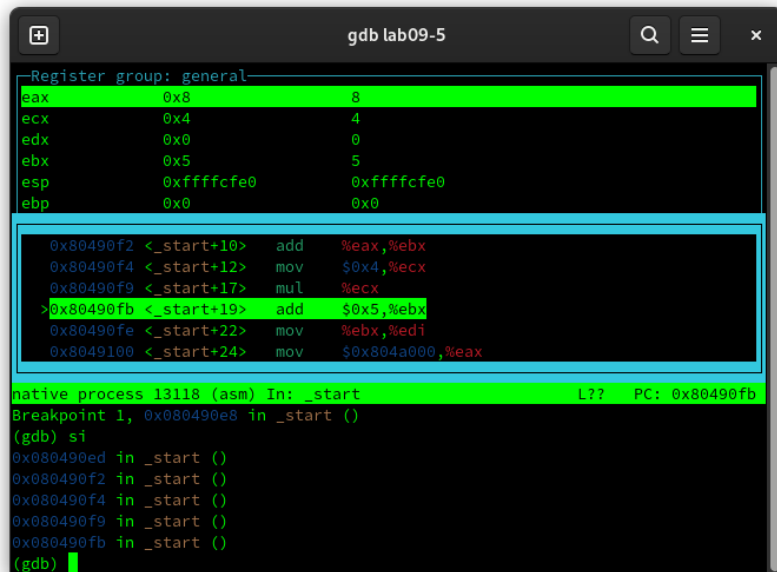


Рис. 2.28: Проблема

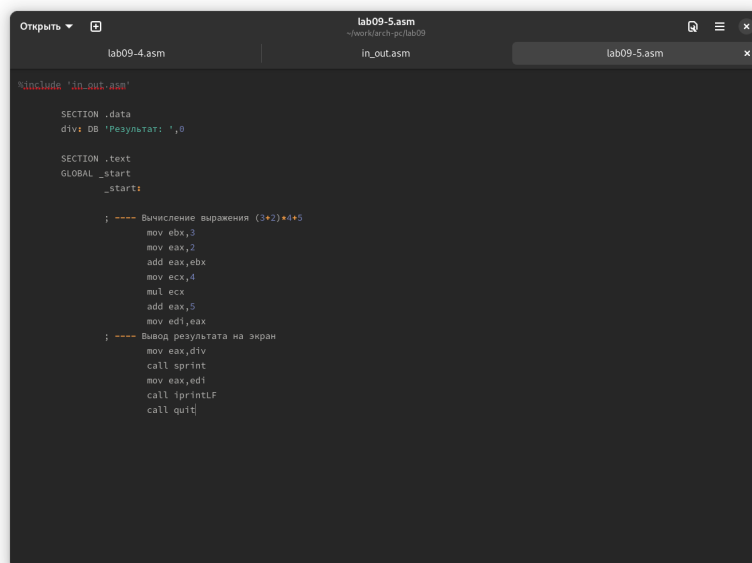
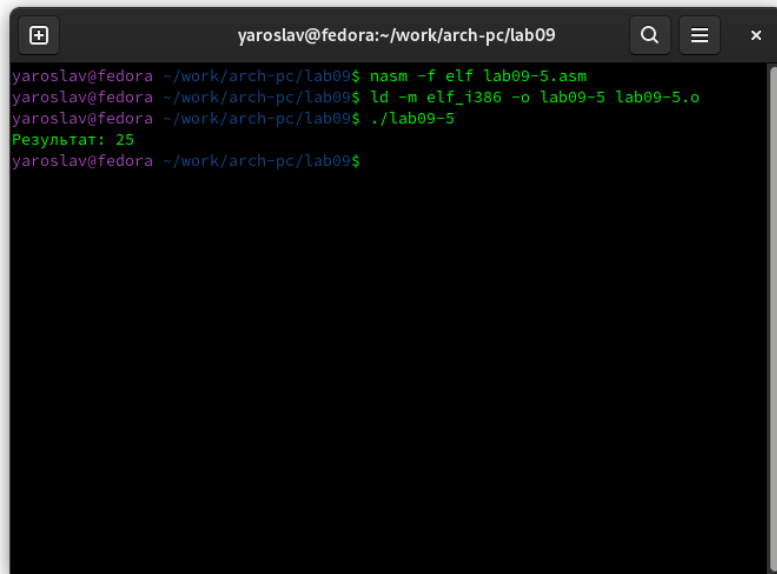


Рис. 2.29: Правильная программа

2. Проверяем работу.



```
yaroslav@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
yaroslav@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
yaroslav@fedora:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
yaroslav@fedora:~/work/arch-pc/lab09$
```

Рис. 2.30: Работа программы

3 Выводы

Были изучены подпрограммы и отладчик. Получены практические навыки их применений.