

Отчёт по лабораторной работе 9

дисциплина: Архитектура компьютеров

Байрамов Керим Сапарович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация подпрограмм в NASM	6
2.2	Отладка программы с помощью GDB	9
2.3	Задание для самостоятельной работы	21
3	Выводы	27

Список иллюстраций

2.1	Программа lab9-1.asm	7
2.2	Запуск программы lab9-1.asm	7
2.3	Программа lab9-1.asm	8
2.4	Запуск программы lab9-1.asm	9
2.5	Программа lab9-2.asm	10
2.6	Запуск программы lab9-2.asm в отладчике	11
2.7	Дизассемблированный код	12
2.8	Дизассемблированный код в режиме intel	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	Вывод значения регистра	20
2.16	Программа prog-1.asm	21
2.17	Запуск программы prog-1.asm	22
2.18	Код с ошибкой	23
2.19	Отладка	24
2.20	Код исправлен	25
2.21	Проверка работы	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

Для выполнения лабораторной работы №9 я создал новую папку и перешел в нее. Далее был создан файл с именем `lab9-1.asm`.

В качестве примера я рассмотрел программу, которая вычисляет арифметическое выражение $f(x) = 2x + 7$ с использованием подпрограммы `calcul`. В этом примере значение переменной x вводится с клавиатуры, а само выражение вычисляется внутри подпрограммы. (рис. 2.1) (рис. 2.2)

```
Открыть ▾ + lab9-1.asm
~/work/arch-pc/lab09

%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: RESB 80

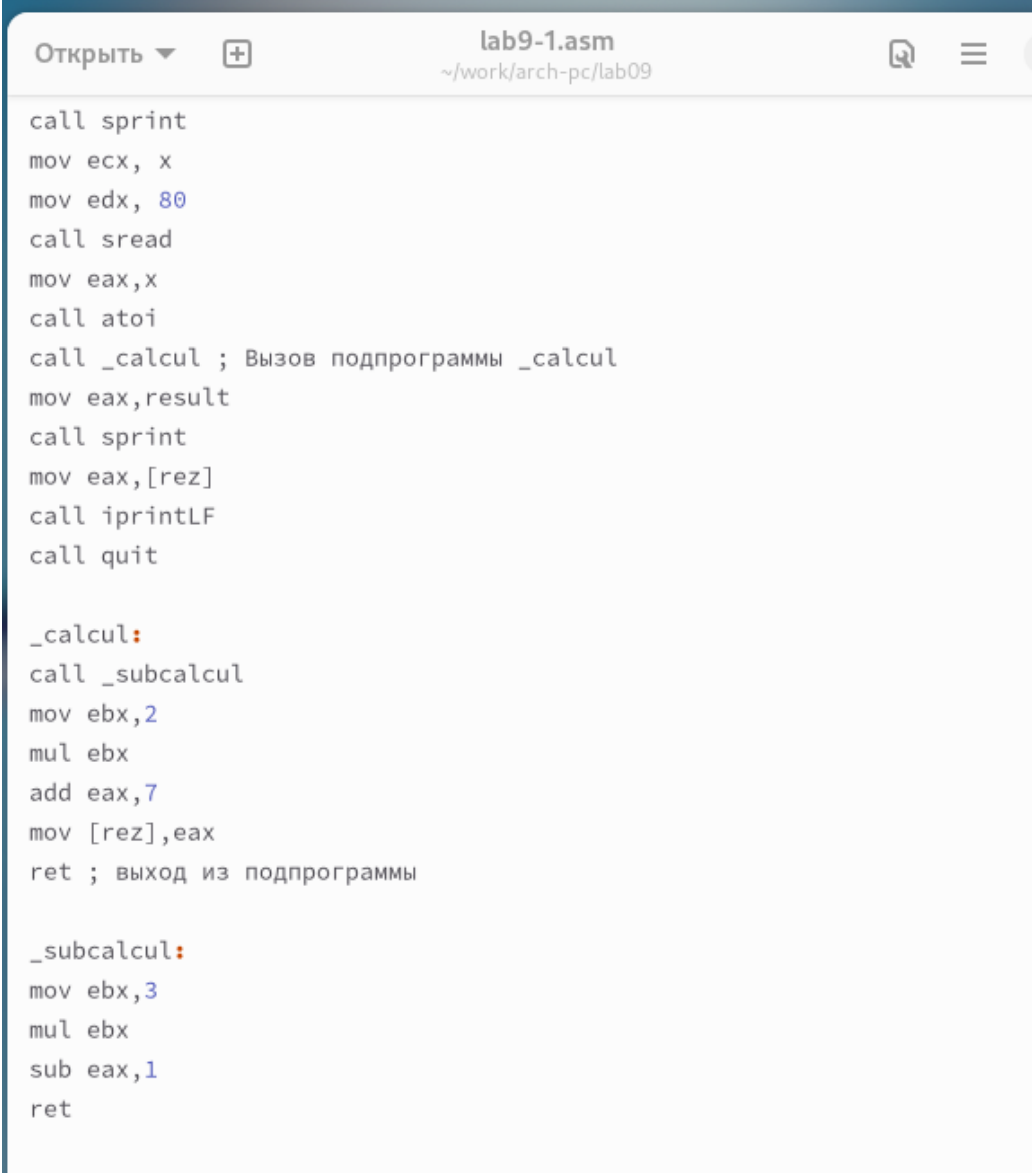
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [rez]
call iprintLF
call quit
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
```

Рис. 2.1: Программа lab9-1.asm

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2x+7=19
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$
```

Рис. 2.2: Запуск программы lab9-1.asm

После этого я внес изменения в текст программы, добавив подпрограмму `subcalcul` внутрь подпрограммы `calcul`. Это позволило вычислить составное выражение $f(g(x))$, где значение x также вводится с клавиатуры. Функции определены следующим образом: $f(x) = 2x + 7$, $g(x) = 3x - 1$. (рис. 2.3) (рис. 2.4)



```
Открыть ▾  lab9-1.asm
~/work/arch-pc/lab09

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
```

Рис. 2.3: Программа lab9-1.asm


```
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2x+7=19
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2(3x-1)+7=41
kerim@kerim-L0Q-IRX9:~/work/arch-pc/lab09$ █
```

Рис. 2.4: Запуск программы lab9-1.asm

2.2 Отладка программы с помощью GDB

Я создал файл с названием lab9-2.asm, в котором содержится программа из Листинга 9.2. Эта программа отвечает за вывод сообщения “Hello world!” на экран. (рис. 2.5)



```
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. 2.5: Программа lab9-2.asm

Затем я скомпилировал файл и получил исполняемый файл. Чтобы добавить отладочную информацию для работы с отладчиком GDB, я использовал ключ “-g”. После этого загрузил полученный исполняемый файл в отладчик GDB и проверил его работу, запустив программу с помощью команды “run” или “r”. (рис. 2.6)

```

kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.1-1.fc39
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/kerim/work/arch-pc/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 3425) exited normally]
(gdb) █

```

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

Для более детального анализа программы я установил точку остановки на метке “start”, с которой начинается выполнение любой ассемблерной программы, и запустил её. Затем я просмотрел дизассемблированный код программы. (рис. 2.7) (рис. 2.8)

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb lab9-2
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) r
Starting program: /home/kerim/work/arch-pc/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 3425) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/kerim/work/arch-pc/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:
=> 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) █
```

Рис. 2.7: Дизассемблированный код

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb lab9-2
Breakpoint 1, _start () at lab9-2.asm:11
11      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)
```

Рис. 2.8: Дизассемблированный код в режиме intel

Чтобы проверить точку остановки по имени метки “_start”, я использовал команду “info breakpoints” или “i b”. После этого установил ещё одну точку остановки по адресу инструкции, определив адрес предпоследней инструкции “mov ebx, 0x0”. (рис. 2.9)

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]

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

native process 3428 (asm) In: _start      L11  PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 22.
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab9-2.asm:11
         breakpoint already hit 1 time
2        breakpoint keep y  0x08049031 lab9-2.asm:22
(gdb) █
```

Рис. 2.9: Точка остановки

В отладчике GDB я имел возможность просматривать содержимое ячеек памяти и регистров, а также изменять значения регистров и переменных. Я выполнил 5 инструкций с помощью команды `stepi` (сокращенно `si`) и отслеживал изменение значений регистров. (рис. 2.10) (рис. 2.11)

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]

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

native process 3428 (asm) In: _start L12 PC: 0x8049005
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
(gdb) █
```

Рис. 2.10: Изменение регистров

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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

native process 3428 (asm) In: _start      L16  PC: 0x8049016
--Type <RET> for more, q to quit, c to continue without paging--
ss      0x2b      43
ds      0x2b      43
es      0x2b      43
fs      0x0      0
gs      0x0      0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) █
```

Рис. 2.11: Изменение регистров

Я также просмотрел значение переменной `msg1` по имени и получил нужные данные. Для изменения значения регистра или ячейки памяти использовал команду `set`, указав имя регистра или адрес в качестве аргумента. Я изменил первый символ переменной `msg1`. (рис. 2.12) (рис. 2.13)


```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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

native process 3428 (asm) In: _start      L16  PC: 0x8049016
(gdb) si
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) set {char}0x804a008='L'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "Lor!d!\n\034"
(gdb) █
```

Рис. 2.12: Изменение значения переменной

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb lab9-2

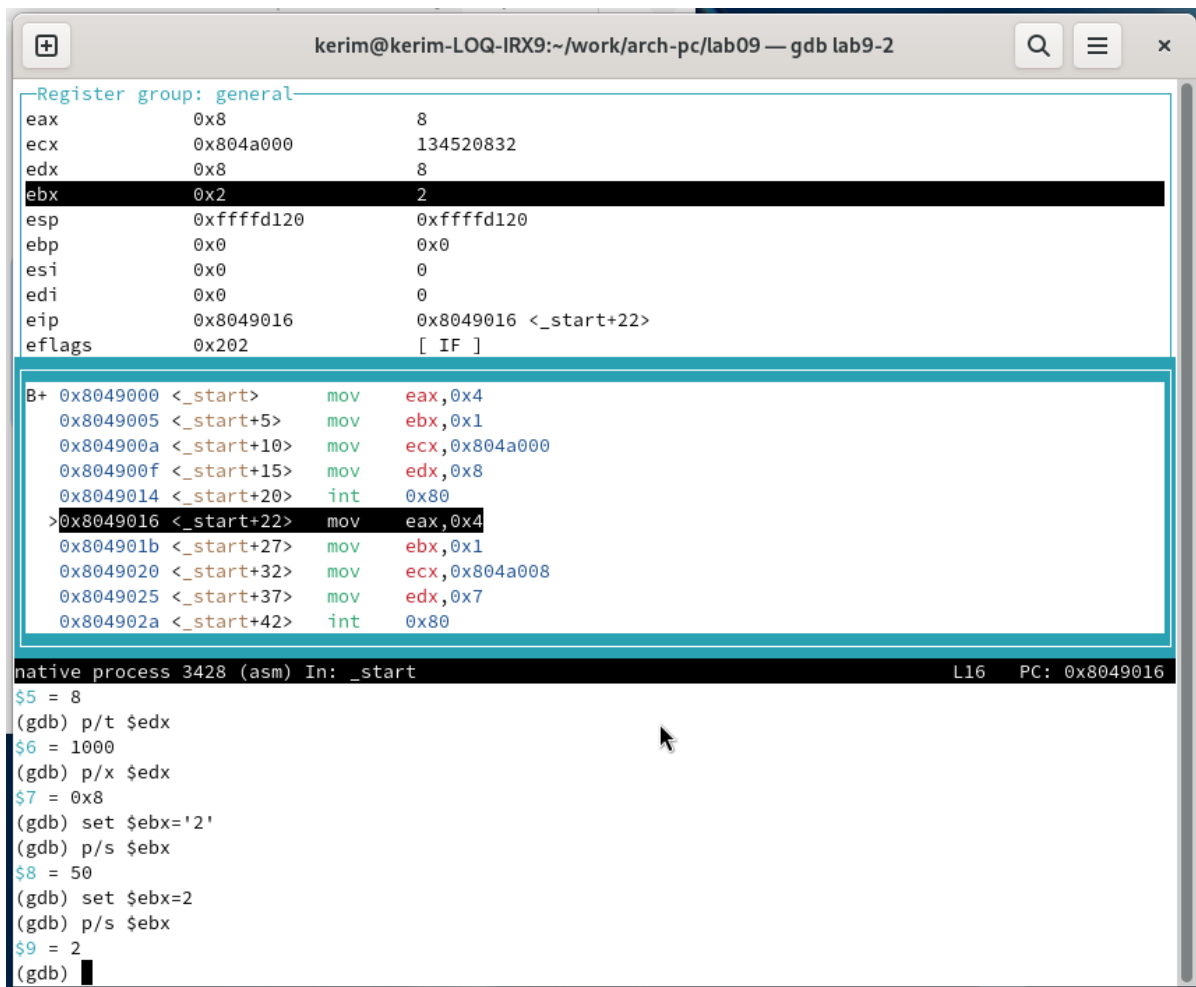
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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

native process 3428 (asm) In: _start L16 PC: 0x8049016
$2 = 1000
(gdb) p/s $ecx
$3 = 134520832
(gdb) p/x $ecx
$4 = 0x804a000
(gdb) p/s $edx
$5 = 8
(gdb) p/t $edx
$6 = 1000
(gdb) p/x $edx
$7 = 0x8
(gdb) █
```

Рис. 2.13: Вывод значения регистра

Кроме того, с помощью команды `set`, я изменил значение регистра `ebx` на нужное значение. (рис. 2.14)



```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x2      2
esp      0xffffd120 0xffffd120
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]

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

native process 3428 (asm) In: _start L16 PC: 0x8049016
$5 = 8
(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)
```

Рис. 2.14: Вывод значения регистра

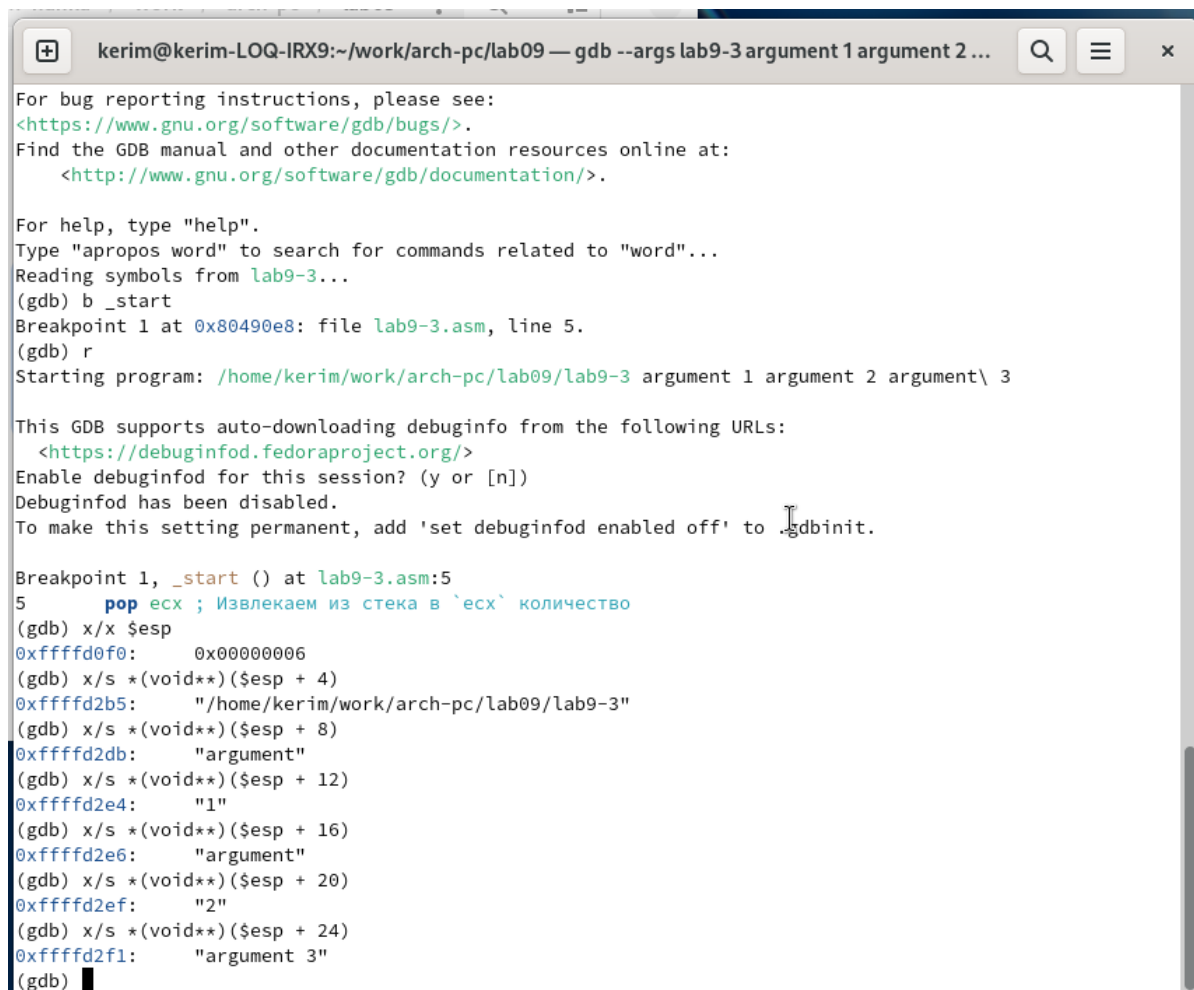
Я скопировал файл `lab8-2.asm`, который был создан в ходе выполнения лабораторной работы №8. Этот файл содержит программу для вывода аргументов командной строки. После этого создал исполняемый файл из скопированного файла.

Для загрузки программы с аргументами в отладчик GDB, использовал ключ `--args` и загрузил исполняемый файл в отладчик с указанными аргументами. Я установил точку останова перед первой инструкцией программы и запустил её.

Адрес вершины стека, где хранится количество аргументов командной строки (включая имя программы), находится в регистре `esp`. По этому адресу находится число, указывающее количество аргументов. В данном случае количество аргу-

ментов равно 5, включая имя программы lab9-3 и сами аргументы: аргумент1, аргумент2 и аргумент 3.

Я также просмотрел остальные позиции стека. По адресу [esp+4] находится адрес в памяти, где располагается имя программы. По адресу [esp+8] хранится адрес первого аргумента, по адресу [esp+12] — второго и так далее. Шаг изменения адреса равен 4, так как каждый следующий адрес на стеке находится на расстоянии 4 байт от предыдущего ([esp+4], [esp+8], [esp+12]). (рис. 2.15)



```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb --args lab9-3 argument 1 argument 2 ...
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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) r
Starting program: /home/kerim/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

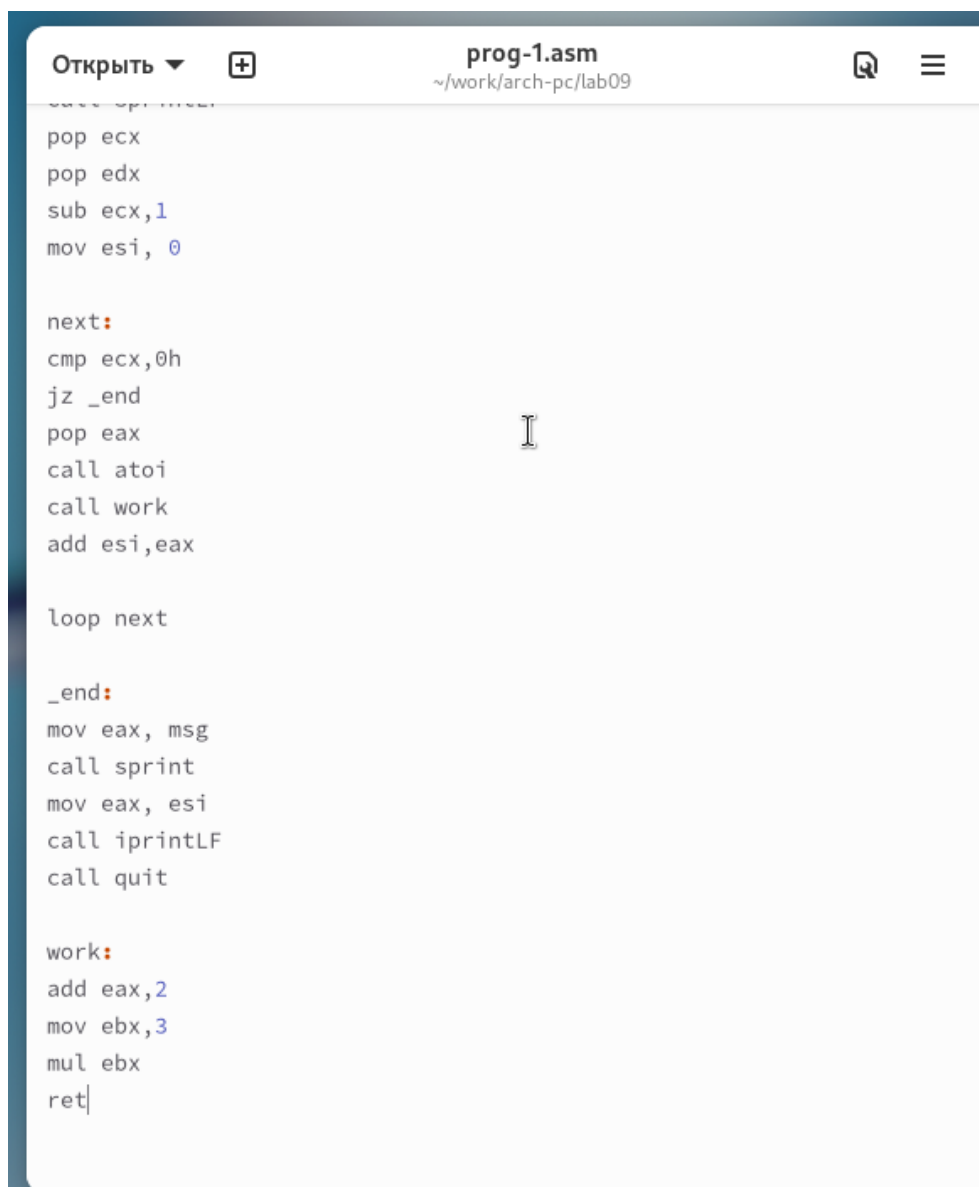
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd0f0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd2b5: "/home/kerim/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2db: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd2e4: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd2e6: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd2ef: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd2f1: "argument 3"
(gdb)
```

Рис. 2.15: Вывод значения регистра

2.3 Задание для самостоятельной работы

Я решил преобразовать программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), добавив вычисление значения функции $f(x)$ в виде подпрограммы. (рис. 2.16) (рис. 2.17)



```
Открыть ▾ + prog-1.asm
~/work/arch-pc/lab09

pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
pop eax
call atoi
call work
add esi,eax

loop next

_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit

work:
add eax,2
mov ebx,3
mul ebx
ret
```


Рис. 2.16: Программа prog-1.asm

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ nasm -f elf prog-1.asm  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ ld -m elf_i386 prog-1.o -o prog-1  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ ./prog-1  
f(x)= 3(x + 2)  
Результат: 0  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ ./prog-1 1  
f(x)= 3(x + 2)  
Результат: 9  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ ./prog-1 3 4 6 8  
f(x)= 3(x + 2)  
Результат: 87  
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09$ █
```

Рис. 2.17: Запуск программы prog-1.asm

В листинге представлена программа для вычисления выражения $(3 + 2) * 4 + 5$. Однако, при запуске программы я обнаружил, что она даёт неверный результат. Для выявления причин я провел анализ изменений значений регистров с помощью отладчика GDB.

В процессе анализа я обнаружил, что порядок аргументов у инструкции `add` был перепутан. Кроме того, я заметил, что по окончании работы программы значение `ebx` было отправлено в `edi` вместо `eax`. (рис. 2.18)

Открыть ▾ 

prog-2.asm
~/work/arch-pc/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
```

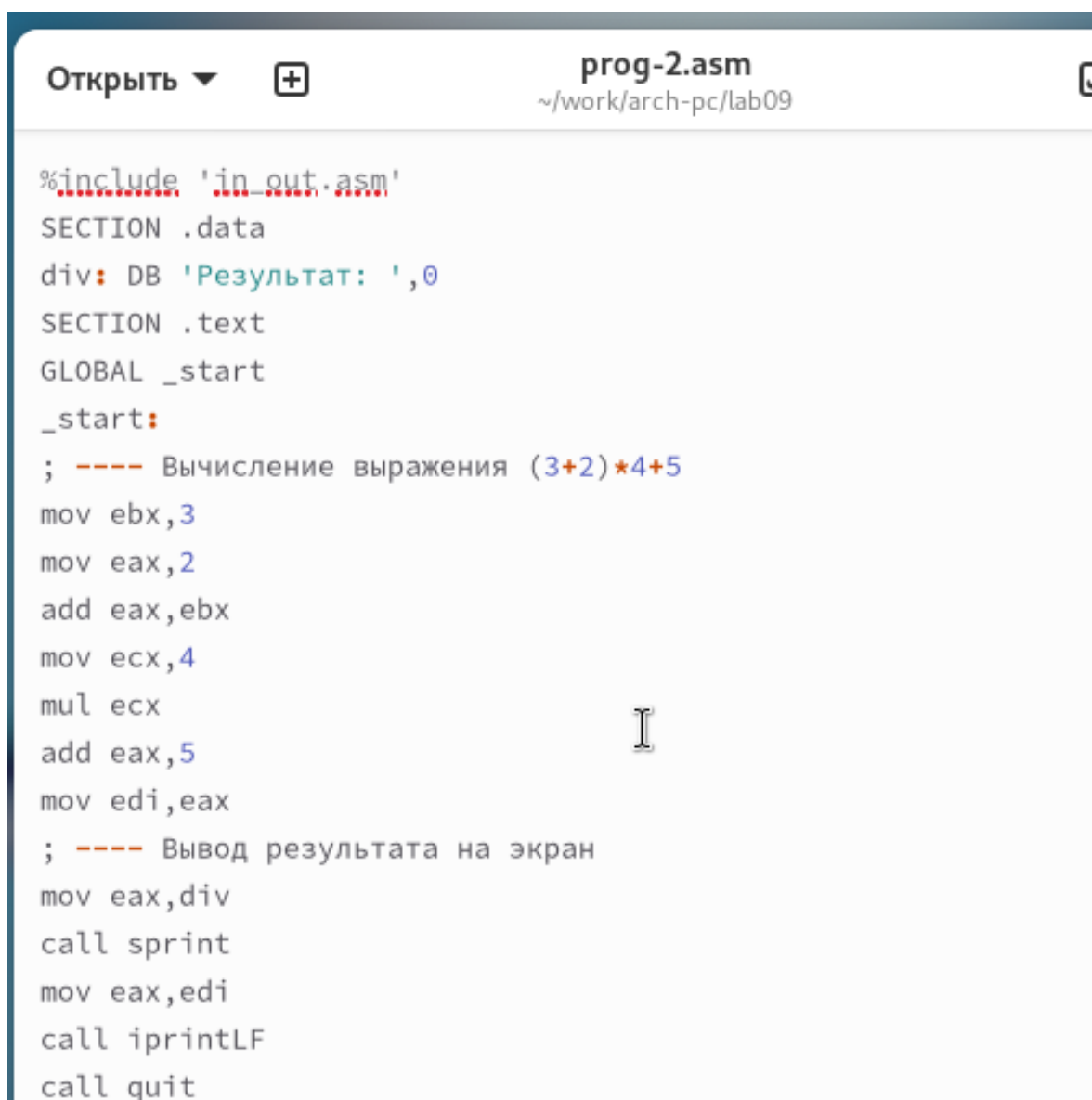
Рис. 2.18: Код с ошибкой



Рис. 2.19: Отладка

Отметим, что перепутан порядок аргументов у инструкции `add` и что по окончании работы в `edi` отправляется `ebx` вместо `eax`. (рис. 2.19)

Исправленный код программы (рис. 2.20) (рис. 2.21)



The screenshot shows a code editor window titled "prog-2.asm" with the path "~/work/arch-pc/lab09". The code is as follows:



```
Открыть ▼  prog-2.asm   
~/work/arch-pc/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
```

Рис. 2.20: Код исправлен

```
kerim@kerim-LOQ-IRX9:~/work/arch-pc/lab09 — gdb prog-2

eax 25

fffd120 xffffd120

[ Register Values Unavailable ]

0490fe x80490fe <_start+22>

0x80490fe <_start+22> mov edi,eax
0x8049100 <_start+24> add eax,eb804a000
0x8049105 <_start+29> call 0x804900f <sprint>
0x804910a <_start+34> mul eax,edi
0x804910c <_start+36> call 0x8049086 <iprintf>
>0x8049111 <_start+41> call 0x80490db <quit>
04a000
rint>

native process 3602 (asm) In: _start L14 PC: 0x80490fe
To makeNo process (asm) In: 'set debuginfo enabled off' to .gdbinit. L?? PC: ??
Breakpoint 1, _start () at prog-2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 3602) exited normally]
(gdb)
```

Рис. 2.21: Проверка работы

3 Выводы

В ходе работы я освоил работу с подпрограммами и отладчиком.