

Лабораторная работа №9

Понятие подпрограммы. Отладчик GDB.

Борисенкова София Павловна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение задания для самостоятельной работы	18

Список иллюстраций

2.1	Создание рабочей директории и файла lab9-1.asm	6
2.2	Вставка кода из файла листинга 9.1	7
2.3	Копирование файла in_out.asm в рабочую директорию	7
2.4	Изменение файла lab9-1.asm	8
2.5	Повторная сборка программы из файла lab9-1.asm и её запуск . . .	8
2.6	Запись кода из листинга 9.2 в файл lab9-2.asm	9
2.7	Отладчик gdb	10
2.8	Запуск lab9-1.asm в gdb	10
2.9	Запуск lab9-1.asm в gdb с точкой останова	11
2.10	Отображение команд с синтаксисом intel	11
2.11	Режим псевдографики	12
2.12	Точки останова	12
2.13	Точки останова 2	13
2.14	Содержание регистров	14
2.15	Содержание msg1	14
2.16	Содержание msg2	15
2.17	Содержание msg1 и msg2	15
2.18	Содержание ebx	16
2.19	Копирование файла	16
2.20	Исследование стека	17
3.1	Код файла самостоятельной работы	18
3.2	Сборка и запуск программы первого задания самостоятельной ра- боты, а также результат выполнения	19
3.3	Код файла самостоятельной работы	19
3.4	Сборка и запуск программы второго задания самостоятельной ра- боты, а также результат выполнения	20

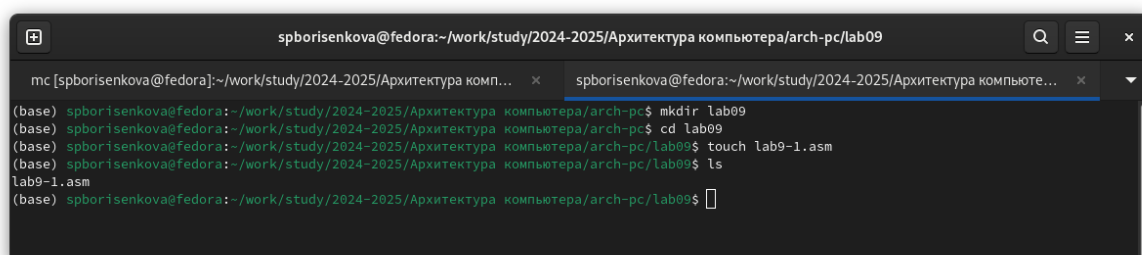
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

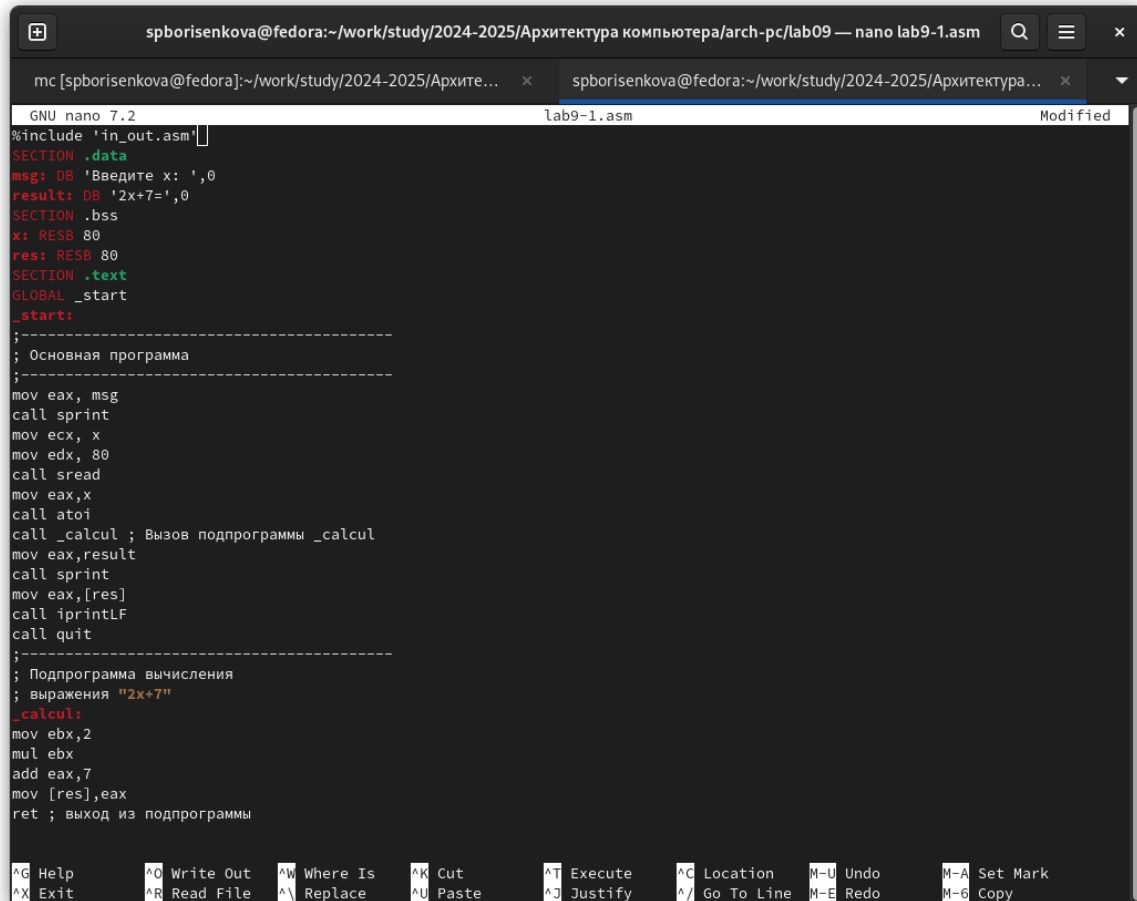
Для начала выполнения лабораторной работы создадим рабочую директорию и файл lab9-1.asm (рис. 2.1):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитектура комп... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьюте... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdir lab09
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab09
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ touch lab9-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ls
lab9-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 2.1: Создание рабочей директории и файла lab9-1.asm

Теперь, вставим в ранее созданный файл из листинга 9.1.

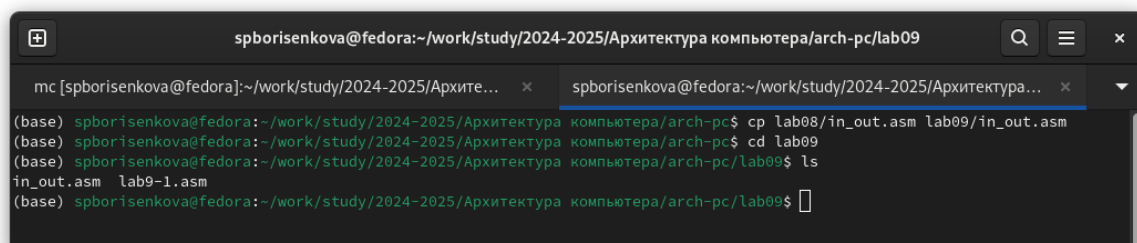


```
GNU nano 7.2 lab9-1.asm Modified
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo     M-A Set Mark
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify   ^_ Go To Line M-E Redo     M-6 Copy
```

Рис. 2.2: Вставка кода из файла листинга 9.1

Чтобы собрать код, нужен файл `in_out.asm`. скопируем его из директории прошлой лабораторной работы (рис. 2.3):

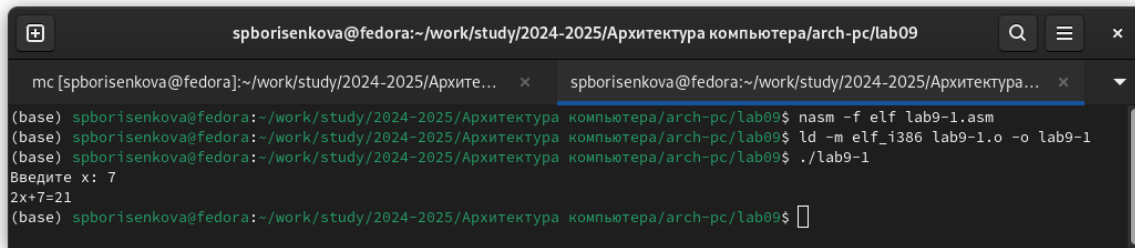


```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cp lab08/in_out.asm lab09/in_out.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab09
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ls
in_out.asm  lab9-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 2.3: Копирование файла `in_out.asm` в рабочую директорию

Теперь соберём программу и посмотрим на результат выполнения. Она работа-

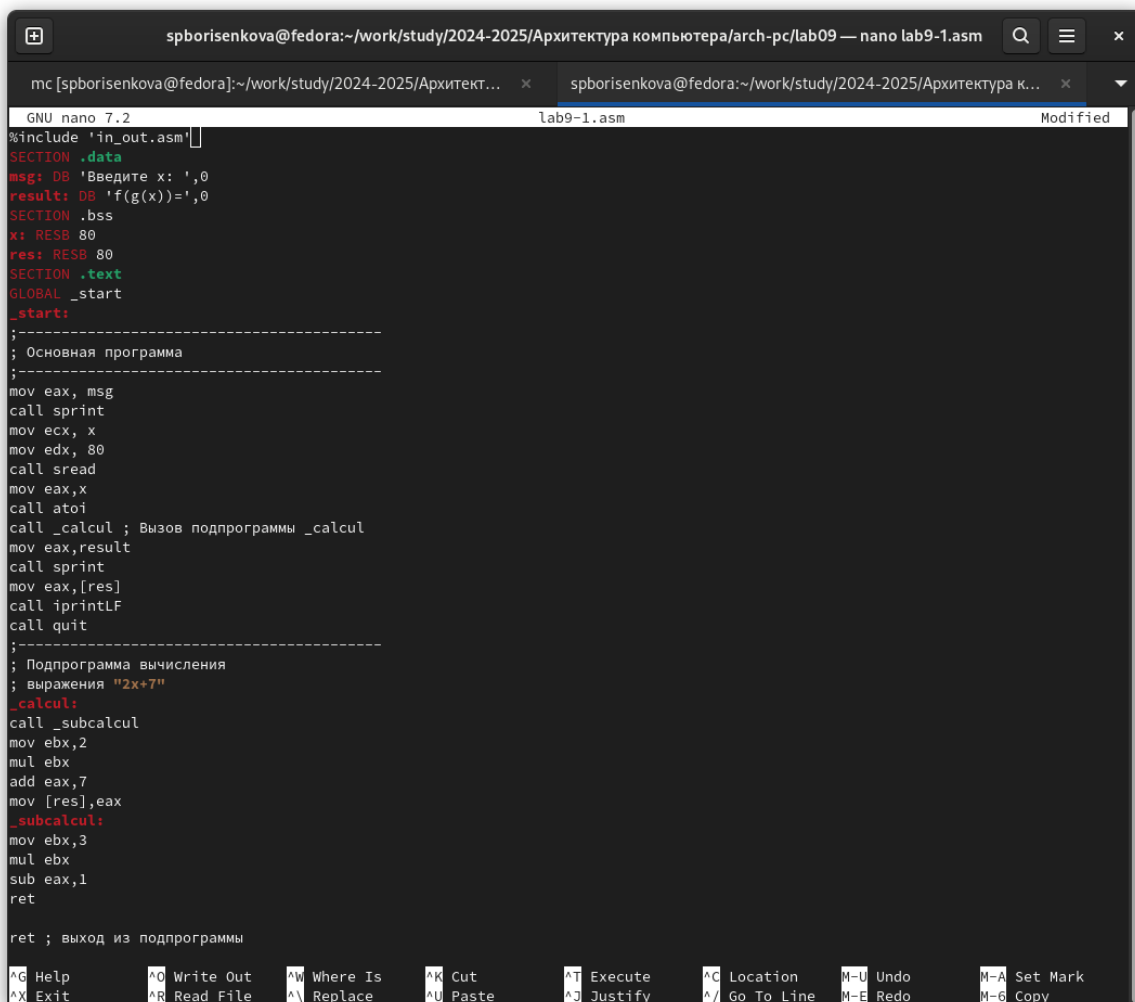
ет корректно. Изменим код, добавив подпрограмму subcalcul



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архите... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab9-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 lab9-1.o -o lab9-1
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./lab9-1
Введите x: 7
2x+7=21
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 2.4: Изменение файла lab9-1.asm

Попробуем собрать программу и запустить её (рис. 2.5):

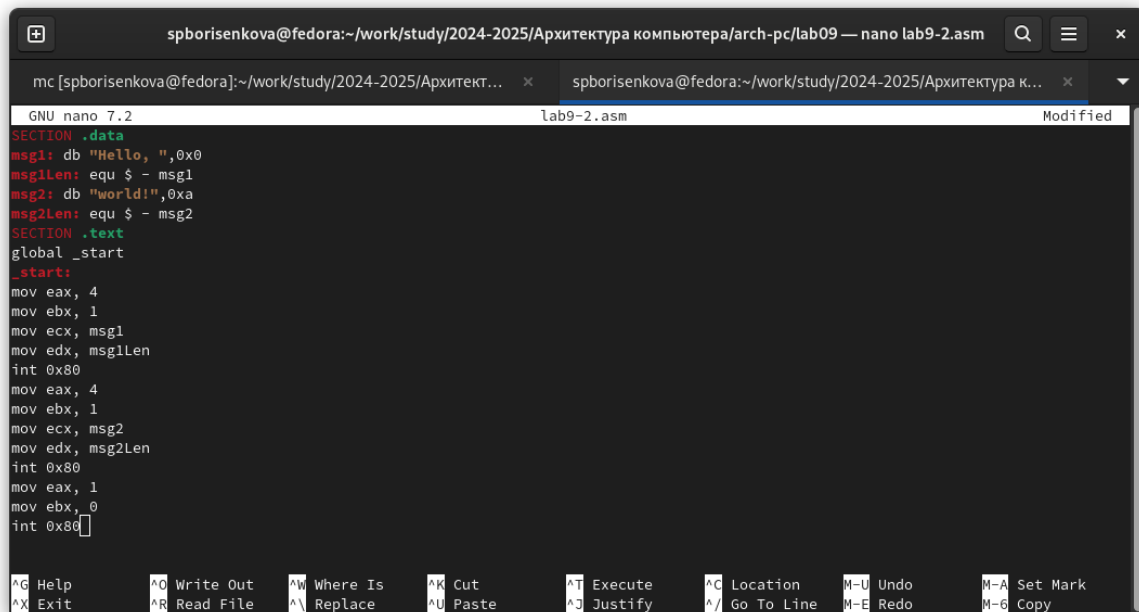


```
GNU nano 7.2 lab9-1.asm Modified
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура... x
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB 'f(g(x))=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
ret ; выход из подпрограммы

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   ^U Undo       ^A Set Mark
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_ Go To Line ^E Redo       ^M Copy
```

Рис. 2.5: Повторная сборка программы из файла lab9-1.asm и её запуск

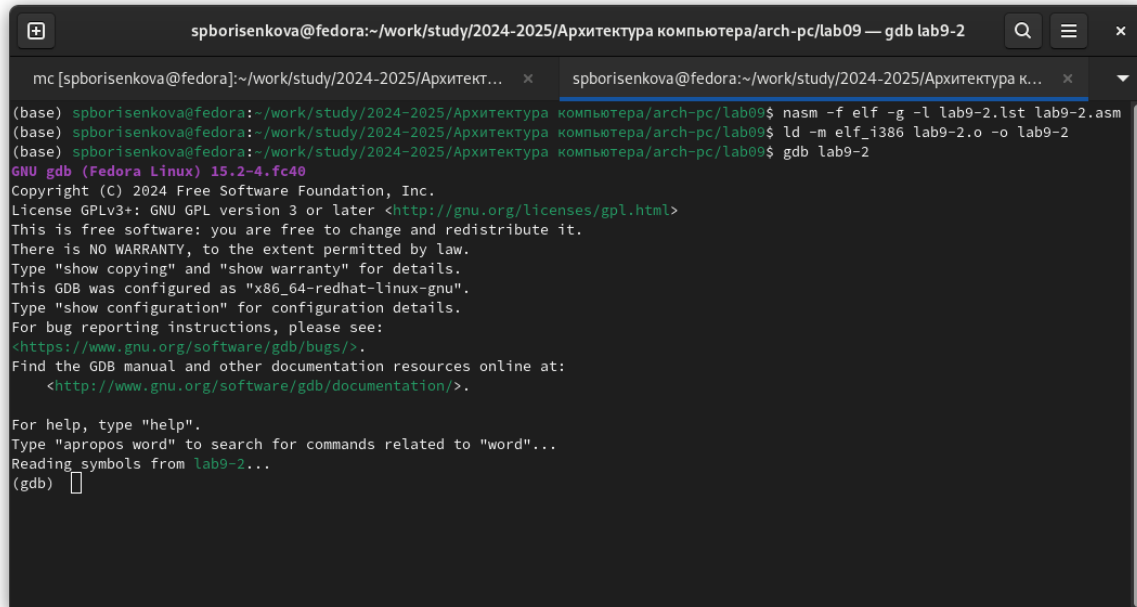
Как видим, она корректно выводит значение функции. Создадим второй файл и вставим в него код из файла листинга 9.2 (рис. 2.6):



```
GNU nano 7.2 lab9-2.asm Modified
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. 2.6: Запись кода из листинга 9.2 в файл lab9-2.asm

Соберём его и откроем исполняемый файл в отладчике (рис. 2.7):

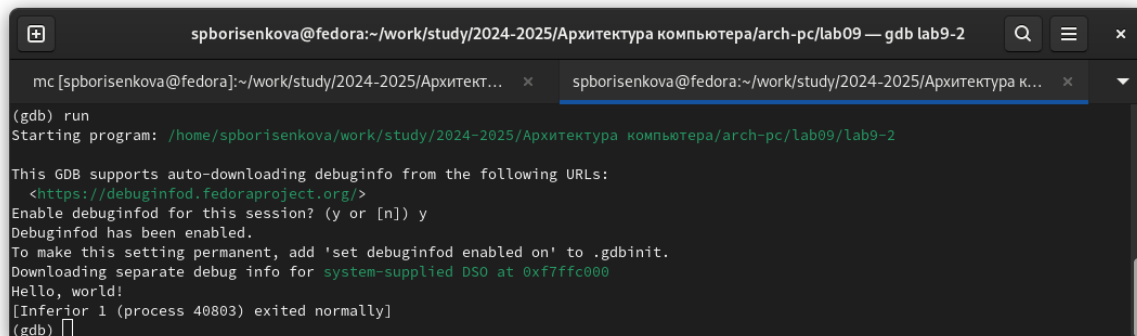


```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 lab9-2.o -o lab9-2
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 15.2-4.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 lab9-2...
(gdb) 
```

Рис. 2.7: Отладчик gdb

Запустим программу командой run (рис. 2.8):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
(gdb) run
Starting program: /home/spborisenkova/work/study/2024-2025/Архитектура компьютера/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]) 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 40803) exited normally]
(gdb) 
```

Рис. 2.8: Запуск lab9-1.asm в gdb

Установим брейкпоинт и запустим ещё раз: (рис. 2.9):

```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
(gdb) break _start
Breakpoint 1 at 0x08049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/spborisenkova/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:    mov     $0x4,%eax
0x08049005 <+5>:    mov     $0x1,%ebx
0x0804900a <+10>:   mov     $0x804a000,%ecx
0x0804900f <+15>:   mov     $0x8,%edx
0x08049014 <+20>:   int     $0x80
0x08049016 <+22>:   mov     $0x4,%eax
0x0804901b <+27>:   mov     $0x1,%ebx
0x08049020 <+32>:   mov     $0x804a008,%ecx
0x08049025 <+37>:   mov     $0x7,%edx
0x0804902a <+42>:   int     $0x80
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 2.9: Запуск lab9-1.asm в gdb с точкой останова

Переключимся на отображение команд с синтаксисом intel (рис. 2.10):

```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
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. 2.10: Отображение команд с синтаксисом intel

Главные отличия синтаксиса AT&T и Intel состоят в порядке записи переменных в командах, виде записи числовых констант, и особенностях записи имён регистров. Включим режим псевдографики (рис. 2.11):

```

spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
-Register group: general-
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffce10 0xffffce10  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0

8> 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

native process 40851 (asm) In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb)

```

Рис. 2.11: Режим псевдографики

Проверим существующие точки останова командой `info breakpoints` (рис. 2.12):

```

spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
-Register group: general-
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffce10 0xffffce10  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0

8> 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

native process 40851 (asm) In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num      Type             Disp Enb Address      What
1        breakpoint       keep y  0x08049000 lab9-2.asm:9
breakpoint already hit 1 time
(gdb)

```

Рис. 2.12: Точки останова

Установим ещё одну по адресу инструкции и опять выведем список точек останова.(рис. 2.13):

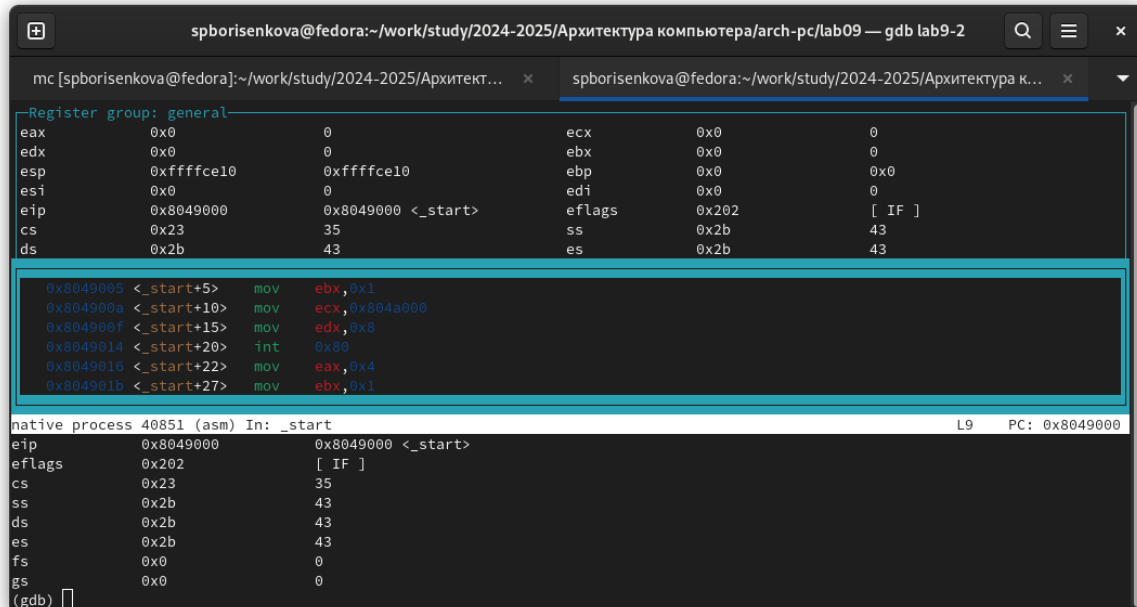
```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
-Register group: general-
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffce10 0xffffce10  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

0x8049005 <_start+5>  mov     ebx,0x1
0x804900a <_start+10>  mov     ecx,0x804a000
0x804900f <_start+15>  mov     edx,0x8
0x8049014 <_start+20>  int     0x80
0x8049016 <_start+22>  mov     eax,0x4
0x804901b <_start+27>  mov     ebx,0x1
0x8049020 <_start+32>  mov     ecx,0x804a008
0x8049025 <_start+37>  mov     edx,0x7
0x804902a <_start+42>  int     0x80
0x804902c <_start+44>  mov     eax,0x1
b+ 0x8049031 <_start+49>  mov     ebx,0x0

native process 40851 (asm) In: _start L9 PC: 0x8049000
breakpoint already hit 1 time
(gdb)
Display all 197 possibilities? (y or n)
(gdb) disassemble _start
No symbol "_start" in current context.
(gdb) disassemble _start
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y   0x08049000 lab9-2.asm:9
      breakpoint already hit 1 time
2     breakpoint       keep y   0x08049031 lab9-2.asm:20
(gdb) 
```

Рис. 2.13: Точки останова 2

Проверим содержание регистров. (рис. 2.14):



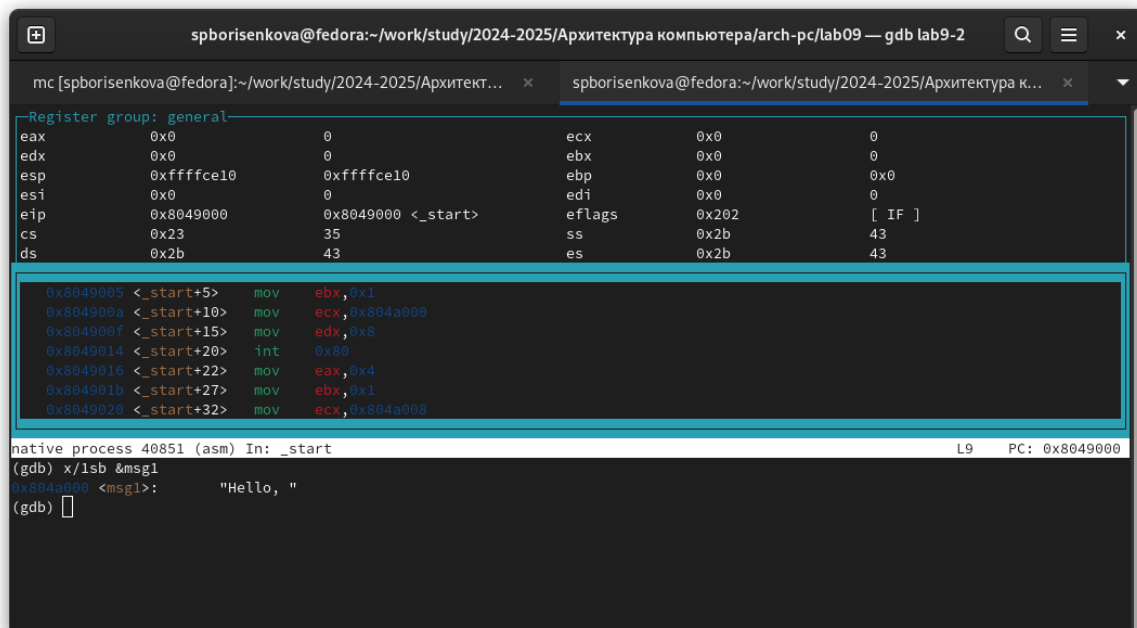
The screenshot shows the GDB interface with the 'Register group: general' window. The registers and their values are as follows:

Register	Value
eax	0x0
edx	0x0
esp	0xffffce10
esi	0x0
eip	0x8049000
cs	0x23
ds	0x2b
ecx	0x0
ebx	0x0
ebp	0x0
edi	0x0
eflags	0x202
ss	0x2b
es	0x2b

Below the register window, the assembly code for the native process 40851 is shown, starting at address 0x8049000. The code includes instructions like `mov ebx, 0x1`, `mov ecx, 0x804a000`, `mov edx, 0x8`, `int 0x80`, `mov eax, 0x4`, and `mov ebx, 0x1`.

Рис. 2.14: Содержание регистров

Проверим содержание переменной msg1 по имени. (рис. 2.15):



The screenshot shows the GDB interface with the 'Register group: general' window. The registers and their values are as follows:

Register	Value
eax	0x0
edx	0x0
esp	0xffffce10
esi	0x0
eip	0x8049000
cs	0x23
ds	0x2b
ecx	0x0
ebx	0x0
ebp	0x0
edi	0x0
eflags	0x202
ss	0x2b
es	0x2b

Below the register window, the assembly code for the native process 40851 is shown, starting at address 0x8049000. The code includes instructions like `mov ebx, 0x1`, `mov ecx, 0x804a000`, `mov edx, 0x8`, `int 0x80`, `mov eax, 0x4`, `mov ebx, 0x1`, and `mov ecx, 0x804a008`.

The GDB command `x/1sb &msg1` is entered, and the output shows the content of the variable `msg1` at address `0x804a000` as `"Hello, "`.

Рис. 2.15: Содержание msg1

Проверим содержание переменной msg2 по адресу. (рис. 2.16):

```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
-Register group: general-
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffce10 0xffffce10  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

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 40851 (asm) In: _start L9 PC: 0x8049000
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a000
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) 
```

Рис. 2.16: Содержание msg2

Изменим командой set содержание msg1 и msg2. (рис. 2.17):

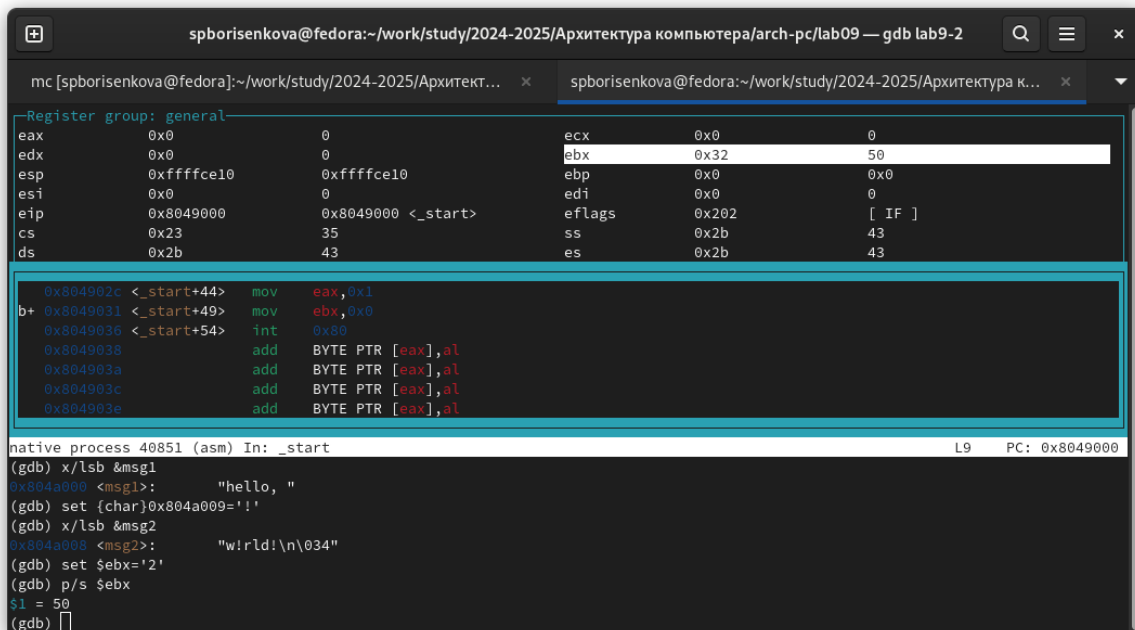
```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
-Register group: general-
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffce10 0xffffce10  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

0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038          add    BYTE PTR [eax],al
0x804903a          add    BYTE PTR [eax],al
0x804903c          add    BYTE PTR [eax],al
0x804903e          add    BYTE PTR [eax],al

native process 40851 (asm) In: _start L9 PC: 0x8049000
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}0x804a009='!'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "w!rld!\n\034"
(gdb) 
```

Рис. 2.17: Содержание msg1 и msg2

Изменим командой set содержание ebx. (рис. 2.18):



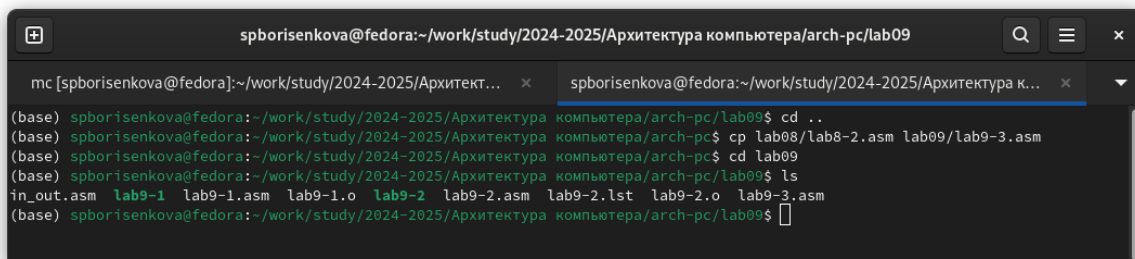
```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab9-2
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к...
Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x32      50
esp      0xffffce10 0xffffce10  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

0x804902c <_start+44> mov    eax,0x1
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80
0x8049038 add    BYTE PTR [eax],al
0x804903a add    BYTE PTR [eax],al
0x804903c add    BYTE PTR [eax],al
0x804903e add    BYTE PTR [eax],al

native process 40851 (asm) In: _start L9 PC: 0x8049000
(gdb) x/lsb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}0x804a009='!'
(gdb) x/lsb &msg2
0x804a008 <msg2>: "w!rld!\n\034"
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) 
```

Рис. 2.18: Содержание ebx

Скопируем файл lab8-2.asm в файл с именем lab9-3.asm и создадим исполняемый файл(рис. 2.19):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитект... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к...
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ cd ..
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cp lab08/lab8-2.asm lab09/lab9-3.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab09
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ls
in_out.asm lab9-1 lab9-1.asm lab9-1.o lab9-2 lab9-2.asm lab9-2.lst lab9-2.o lab9-3.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ 
```

Рис. 2.19: Копирование файла

Загрузим файл в gdb и исследуем расположение аргументов командной строки в стеке после запуска программы (рис. 2.20):


```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb --args lab9-3 аргумент1 аргумент2 аргумент3
mc [spborisenkova@fedora:~/work/study/2024-2025/Архитектура комп... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьтер... x
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) run
Starting program: /home/spborisenkova/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab9-3 аргумент1 аргумент2 аргумент3

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.

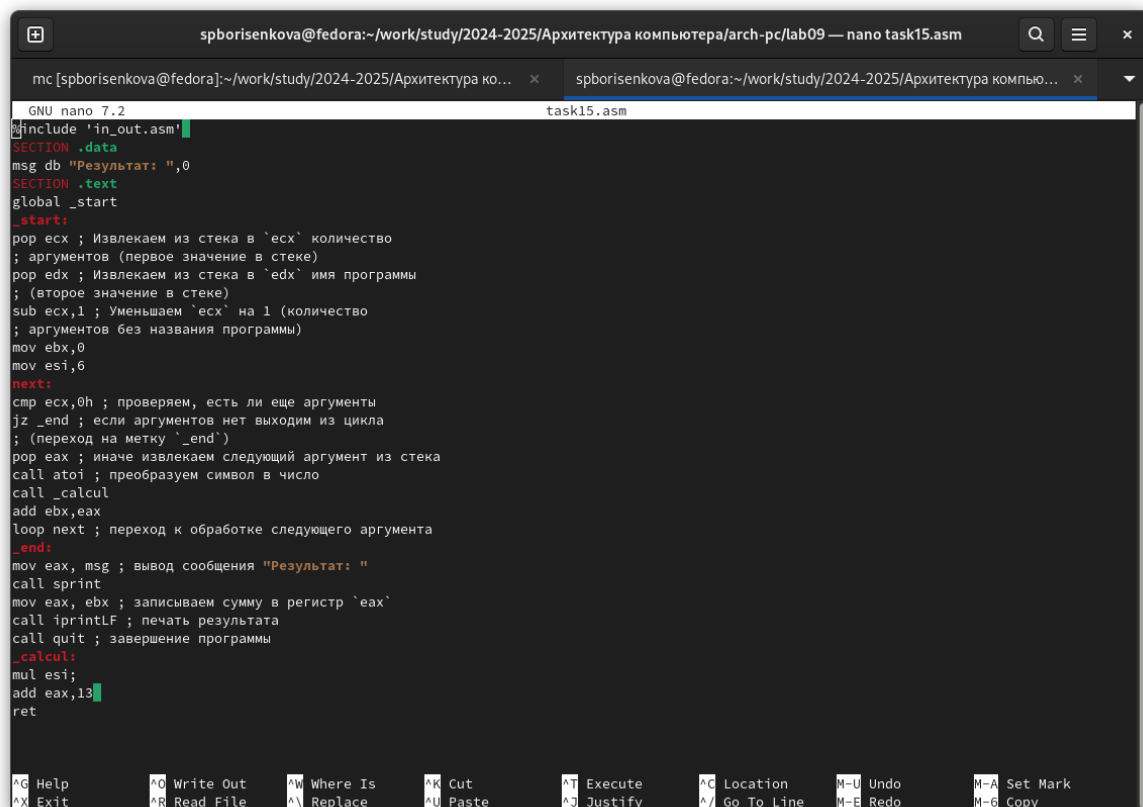
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/s *(void**)(esp + 4)
0xffffcfb1: "/home/spborisenkova/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd01b: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd02d: "аргумент2"
(gdb) x/s *(void**)(esp + 16)
0xffffd03e: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd040: "аргумент3"
(gdb) 
```

Рис. 2.20: Исследование стека

Как видим, для вывода каждого элемента стека нам нужно менять значение адреса с шагом 4. Это связано с тем, что под каждый элемент выделяется 4 байта

3 Выполнение задания для самостоятельной работы

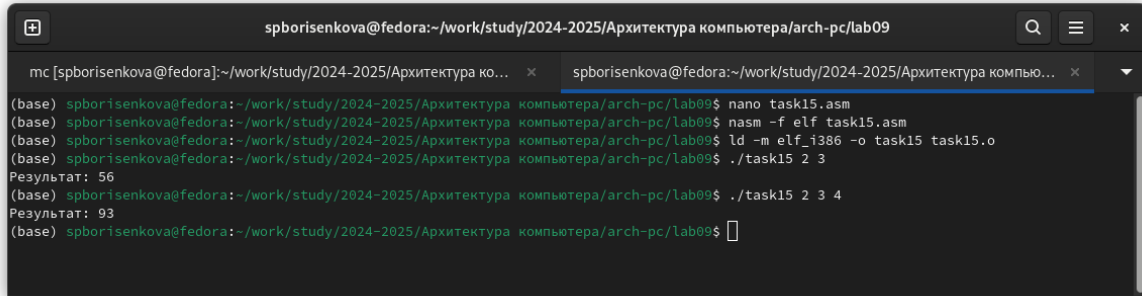
Для выполнения самостоятельной работы создадим файл task15.asm. Скопируем в него программу из работы 8 и изменим соответственно заданию(рис. 3.1):



```
GNU nano 7.2 task15.asm
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
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)
mov ebx,0
mov esi,6
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку '_end')
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
call _calcul
add ebx,eax
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
mov eax, ebx ; записываем сумму в регистр 'eax'
call iprintfLF ; печать результата
call quit ; завершение программы
_calcul:
mul esi;
add eax,13
ret
```

Рис. 3.1: Код файла самостоятельной работы

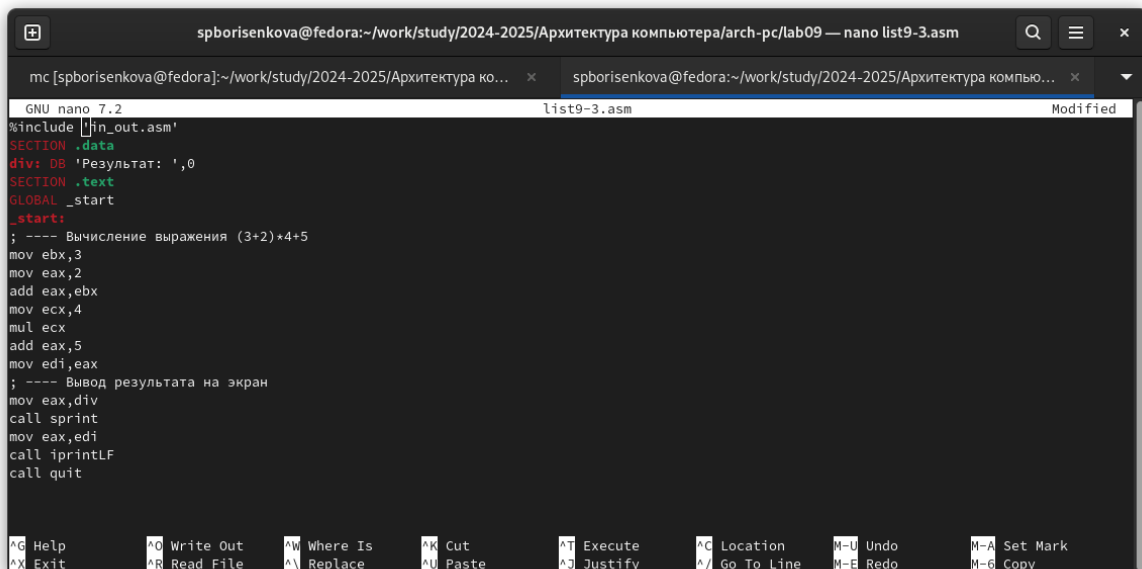
Соберём и запустим программу, вводя различные аргументы (рис. 3.2):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
mc [spborisenkova@fedora:~/work/study/2024-2025/Архитектура ко... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компью... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nano task15.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf task15.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o task15 task15.o
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./task15 2 3
Результат: 56
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./task15 2 3 4
Результат: 93
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 3.2: Сборка и запуск программы первого задания самостоятельной работы, а также результат выполнения

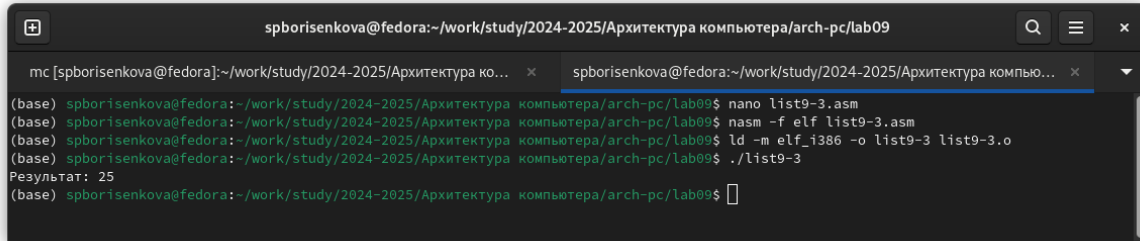
Пересчитав результат вручную, убеждаемся, что программа работает верно
Создадим файл list9-3.asm. Скопируем в него листинг и изменим соответствен-
но заданию(рис. 3.3):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — nano list9-3.asm
mc [spborisenkova@fedora:~/work/study/2024-2025/Архитектура ко... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компью... x
GNU nano 7.2 list9-3.asm Modified
%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
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^G Location M-U Undo M-A Set Mark
^X Exit ^R Read File ^_ Replace ^U Paste ^J Justify ^_ Go To Line M-E Redo M-G Copy
```

Рис. 3.3: Код файла самостоятельной работы

Соберём и запустим программу (рис. 3.4):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09
mc [spborisenkova@fedora:~/work/study/2024-2025/Архитектура ко... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компью... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nano list9-3.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf list9-3.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o list9-3 list9-3.o
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./list9-3
Результат: 25
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 3.4: Сборка и запуск программы второго задания самостоятельной работы, а также результат выполнения

Пересчитав результат вручную, убеждаемся, что теперь программа работает верно # Выводы

В результате выполнения лабораторной работы были получены представления о работе подпрограмм, а также было реализовано несколько программ, использующих подпрограммы. Также, были получены навыки работы с базовым функционалом gdb, и с помощью gdb была отловлена ошибка в коде программы