

# **Отчёт по лабораторной работе 9**

**дисциплина: Архитектура компьютера**

Амина Усманова

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация подпрограмм в NASM . . . . .	6
2.2	Отладка программы с помощью GDB . . . . .	10
2.3	Работа с аргументами командной строки . . . . .	20
2.4	Задание для самостоятельной работы . . . . .	21
2.5	Выводы . . . . .	27

## Список иллюстраций

2.1	Исходный код программы lab9-1.asm . . . . .	7
2.2	Результат выполнения программы . . . . .	8
2.3	Модифицированный код программы . . . . .	9
2.4	Результат выполнения модифицированной программы . . . . .	10
2.5	Код программы lab9-2.asm . . . . .	11
2.6	Запуск программы в отладчике . . . . .	12
2.7	Дизассемблированный код . . . . .	13
2.8	Дизассемблированный код в режиме Intel . . . . .	14
2.9	Установка точки останова . . . . .	15
2.10	Изменение значений регистров . . . . .	16
2.11	Отслеживание изменений регистров . . . . .	17
2.12	Изменение переменной . . . . .	18
2.13	Отображение измененного регистра . . . . .	19
2.14	Изменение регистра ebx . . . . .	20
2.15	Просмотр аргументов командной строки . . . . .	21
2.16	Код программы prog-1.asm . . . . .	22
2.17	Результат выполнения программы . . . . .	23
2.18	Код с ошибками . . . . .	24
2.19	Результат отладки . . . . .	25
2.20	Исправленный код программы . . . . .	26
2.21	Результат проверки . . . . .	27

## **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

Сначала я создала новую папку для выполнения лабораторной работы №9 и перешла в нее. Затем создала файл с именем lab9-1.asm.

В качестве примера я реализовала программу, вычисляющую арифметическое выражение  $f(x) = 2x + 7$  с использованием подпрограммы calcul. Значение переменной  $x$  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
```

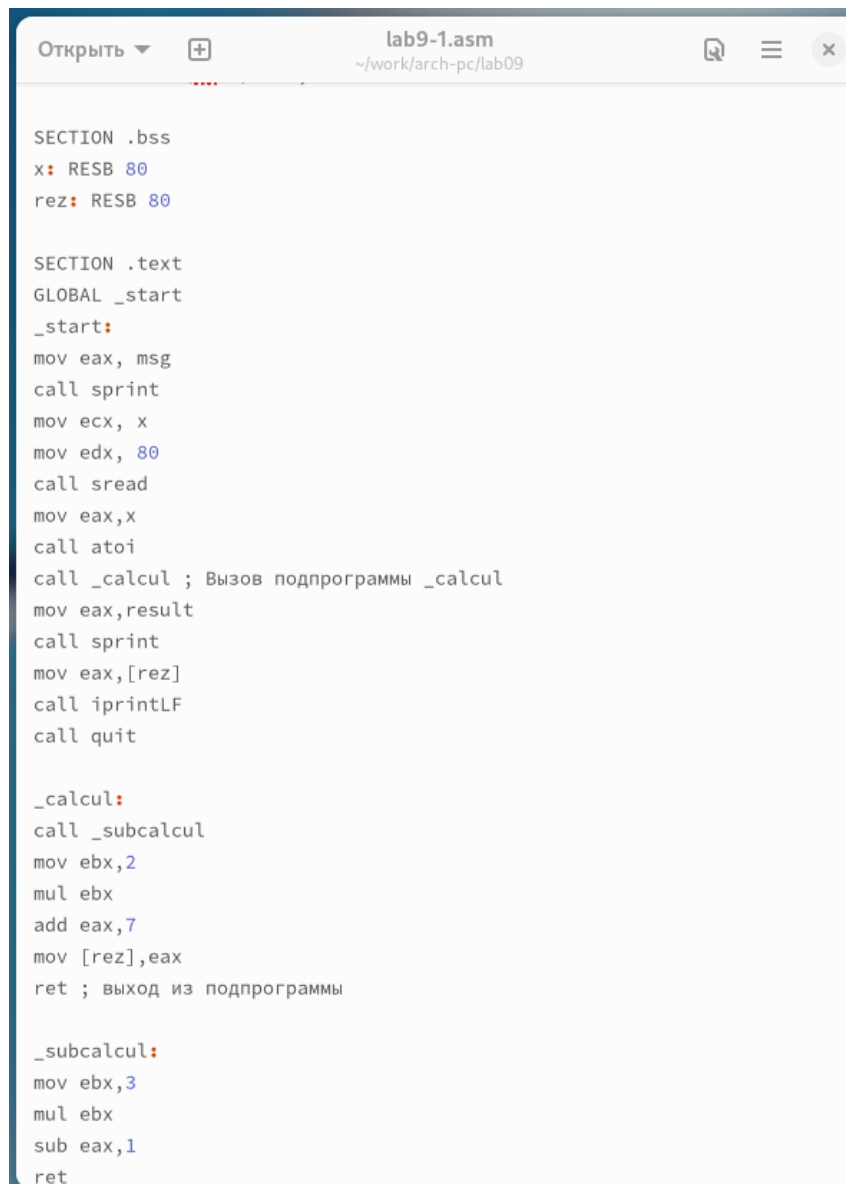
Рис. 2.1: Исходный код программы lab9-1.asm

```
abusmanova@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
abusmanova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
abusmanova@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 2
2x+7=11
abusmanova@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 1
2x+7=9
abusmanova@fedora:~/work/arch-pc/lab09$
```

Рис. 2.2: Результат выполнения программы

Затем я изменила текст программы, добавив подпрограмму `subcalcul` внутрь подпрограммы `calcul`. Это позволило вычислять составное выражение  $f(g(x))$ , где  $f(x) = 2x + 7$ ,  $g(x) = 3x - 1$ . Значение  $x$  также вводится с клавиатуры.





```
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
```


Рис. 2.3: Модифицированный код программы

```
abusmanova@fedora:~/work/arch-pc/lab09$  
abusmanova@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm  
abusmanova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o  
abusmanova@fedora:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 2  
2(3x-1)+7=17  
abusmanova@fedora:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 1  
2(3x-1)+7=11  
abusmanova@fedora:~/work/arch-pc/lab09$
```

Рис. 2.4: Результат выполнения модифицированной программы

## 2.2 Отладка программы с помощью GDB

Я создала файл lab9-2.asm, содержащий программу для вывода сообщения “Hello, world!” (Листинг 9.2).

Открыть ▾  lab9-2.asm  
~/work/arch-pc/lab09

```
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: Код программы lab9-2.asm

Скомпилировала файл с ключом -g для добавления отладочной информации и загрузила его в GDB. Затем запустила программу командой run.

```

abusmanova@fedora:~/work/arch-pc/lab09$
abusmanova@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
abusmanova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
abusmanova@fedora:~/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/abusmanova/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 3794) exited normally]
(gdb)

```

Рис. 2.6: Запуск программы в отладчике

Установила точку останова на метке `_start`, запустила программу, а затем просмотрела дизассемблированный код.

```
abusmanova@fedora:~/work/arch-pc/lab09 — gdb lab9-2

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) r
Starting program: /home/abusmanova/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 3794) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/abusmanova/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

```
abusmanova@fedora:~/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

Установила дополнительные точки останова, используя команды `info breakpoints` и `break`. Например, добавила точку на инструкции `mov ebx, 0x0`.

```

abusmanova@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd0f0 0xffffd0f0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000 0x8049000 <_start>  eflags    0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

B->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
0x804902c <_start+44> mov    eax,0x1

native process 3801 (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: Установка точки останова

С помощью команды `stepi` (или `si`) я пошагово выполняла инструкции, отслеживая изменения регистров.

```
abusmanova@fedora:~/work/arch-pc/lab09 — gdb lab9-2
Register group: general
eax      0x4      4      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd0f0 0xffffd0f0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>  eflags    0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       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
0x8049025 <_start+37>   mov    edx,0x7
0x804902a <_start+42>   int     0x80
0x804902c <_start+44>   mov    eax,0x1

native process 3801 (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 значений регистров



```
abusmanova@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x4      4      ecx      0x804a000    134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd0f0 0xffffd0f0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x804901b 0x804901b <_start+27>  eflags   0x202    [ IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       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
0x8049025 <_start+37>   mov     edx,0x7
0x804902a <_start+42>   int     0x80
0x804902c <_start+44>   mov     eax,0x1

native process 3801 (asm) In: _start L17 PC: 0x804901b
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) si
(gdb) si
```

Рис. 2.11: Отслеживание изменений регистров

Я также изменила значение переменной msg1 и регистров, используя команду set.

```
abusmanova@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax    0x4      4      ecx    0x804a000  134520832
edx    0x8      8      ebx    0x1      1
esp    0xffffd0f0 0xffffd0f0  ebp    0x0      0x0
esi    0x0      0      edi    0x0      0
eip    0x804901b 0x804901b <_start+27>  eflags 0x202     [ IF ]
cs     0x23     35     ss     0x2b     43
ds     0x2b     43     es     0x2b     43
fs     0x0      0      gs     0x0      0

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
0x804902c <_start+44>   mov     eax,0x1

native process 3801 (asm) In: _start L17 PC: 0x804901b
(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>:      "Lorld!\n\034"
(gdb) 
```

Рис. 2.12: Изменение переменной

```
abusmanova@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general
eax      0x4      4      ecx      0x804a000    134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd0f0  0xffffd0f0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x804901b  0x804901b <_start+27> eflags   0x202    [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       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
0x8049025 <_start+37>   mov    edx,0x7
0x804902a <_start+42>   int     0x80
0x804902c <_start+44>   mov    eax,0x1

native process 3801 (asm) In: _start L17 PC: 0x804901b
$2 = 100
(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: Отображение измененного регистра

Используя аналогичные команды, я изменила значение регистра ebx.

abusmanova@fedora:~/work/arch-pc/lab09 — gdb lab9-2

Register group: general

eax	0x4	4	ecx	0x804a000	134520832
edx	0x8	8	ebx	0x2	2
esp	0xffffd0f0	0xffffd0f0	ebp	0x0	0x0
esi	0x0	0	edi	0x0	0
eip	0x804901b	0x804901b <_start+27>	eflags	0x202	[ IF ]
cs	0x23	35	ss	0x2b	43
ds	0x2b	43	es	0x2b	43
fs	0x0	0	gs	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
0x8049025 <_start+37>     mov     edx,0x7
0x804902a <_start+42>     int     0x80
0x804902c <_start+44>     mov     eax,0x1
  
```

native process 3801 (asm) In: \_start L17 PC: 0x804901b

```

$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: Изменение регистра ebx

## 2.3 Работа с аргументами командной строки

Для работы с аргументами командной строки я использовала файл lab8-2.asm (из лабораторной работы №8), создав из него исполняемый файл. Затем загрузила программу в GDB с аргументами, используя ключ `-args`.

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

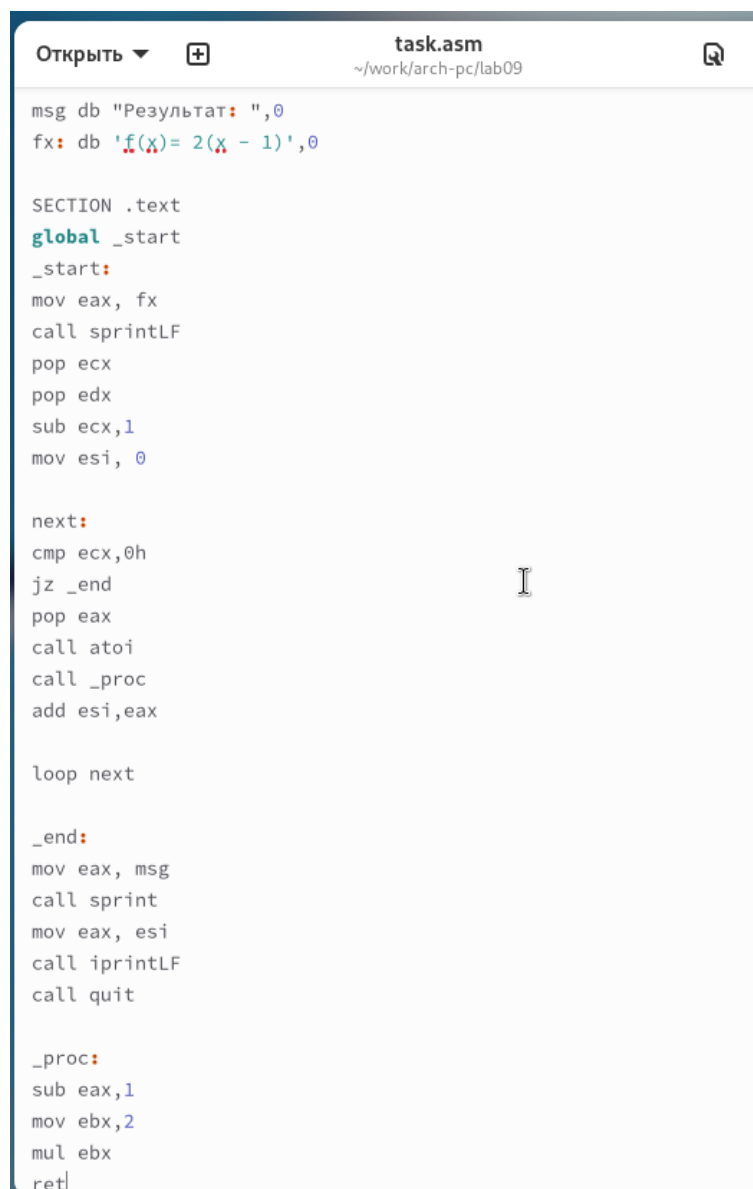
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n])
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd0c0: 0x00000006
(gdb) x/s *(void**)(esp + 4)
0xffffd230: "/home/abusmanova/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2b1: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd2ba: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd2bc: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd2c5: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd2c7: "argument 3"
(gdb) 
```

Рис. 2.15: Просмотр аргументов командной строки

## 2.4 Задание для самостоятельной работы

В рамках задания я модифицировала программу из лабораторной работы №8, добавив подпрограмму для вычисления функции  $f(x)$ .



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

msg db "Результат: ",0
fx: db 'f(x)= 2(x - 1)',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 _proc
add esi,eax

loop next

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


_proc:
sub eax,1
mov ebx,2
mul ebx
ret
```

Рис. 2.16: Код программы prog-1.asm

```
abusmanova@fedora:~/work/arch-pc/lab09$  
abusmanova@fedora:~/work/arch-pc/lab09$ nasm -f elf task.asm  
abusmanova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 task.o -o task  
abusmanova@fedora:~/work/arch-pc/lab09$ ./task 2  
f(x)= 2(x - 1)  
Результат: 2  
abusmanova@fedora:~/work/arch-pc/lab09$ ./task 4 3 1 5  
f(x)= 2(x - 1)  
Результат: 18  
abusmanova@fedora:~/work/arch-pc/lab09$
```

Рис. 2.17: Результат выполнения программы

В процессе выполнения программы я обнаружила ошибку: порядок аргументов в инструкции `add` был перепутан, а регистр `ebx` вместо `eax` отправлялся в `edi`.

Открыть ▾ 

task2.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 ошибками



The screenshot shows a GDB window titled "abusmanova@fedora:~/work/arch-pc/lab09 — gdb task2". The "Register group: general" section displays the following values:

Register	Value	Register	Value
eax	0x8	ecx	0x4
edx	0x0	ebx	0xa
esp	0xffffd0f0	ebp	0x0
esi	0x0	edi	0xa
eip	0x8049100	eflags	0x206 [ PF IF ]
cs	0x23	ss	0x2b
ds	0x2b	es	0x2b
fs	0x0	gs	0x0

The assembly window shows the following code:

```
B+ 0x80490e8 <_start>    mov     ebx,0x3
0x80490ed <_start+5>    mov     eax,0x2
0x80490f2 <_start+10>   add     ebx,eax
0x80490f4 <_start+12>   mov     ecx,0x4
0x80490f9 <_start+17>   mul     ecx
0x80490fb <_start+19>   add     ebx,0x5
0x80490fe <_start+22>   mov     edi,ebx
>0x8049100 <_start+24> mov     eax,0x804a000
0x8049105 <_start+29>   call   0x804900f <sprint>
0x804910a <_start+34>   mov     eax,edi
0x804910c <_start+36>   call   0x8049086 <iprintLF>
```



The status bar indicates "native process 3974 (asm) In: \_start L16 PC: 0x8049100". The console output shows:

```
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at task2.asm:8
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
```

Рис. 2.19: Результат отладки

После исправления ошибок программа заработала корректно.

Открыть ▾  task2.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: Исправленный код программы

The screenshot shows the GDB interface with the following components:

- Register Window:** Displays values for `eax` (25), `ecx` (0x4), and `4` (x0). Below these, memory addresses `ffffd0f0`, `xffffd0f0`, and `x8049100 <_start+24>` are shown.
- Assembly Window:** Shows assembly instructions:
  - `0x8049100 <_start+24> mov eax,0x804a000`
  - `0x8049105 <_start+29> call 0x804900f <sprint>`
  - `0x804910a <_start+34> mov ecx,0di`
  - `0x804910c <_start+36> call 0x8049086 <iprintfLF>`
  - `0x8049111 <_start+41> call 0x80490db <quit>`
- Command Window:** Contains the following text:
  - `> 04a000`
  - `rint>`
  - `86 <iprintfLF>`
- Status Bar:** Shows `native process 4021 (asm) In: _start` and `L16 PC: 0x8049100`.
- Output Window:** Displays the following text:
  - `No process (asm) In:`
  - `(gdb) si`
  - `(gdb) si`
  - `(gdb) si`
  - `(gdb) si`
  - `(gdb) si`
  - `(gdb) si`
  - `(gdb) si`
  - `(gdb) c`
  - `Continuing.`
  - `Результат: 25`
  - `[Inferior 1 (process 4021) exited normally]`
  - `(gdb)`

Рис. 2.21: Результат проверки

## 2.5 Выводы

В ходе лабораторной работы я научилась работать с подпрограммами и отладчиком GDB, а также диагностировать и исправлять ошибки в ассемблерных программах.