

# **Отчёт по лабораторной работе 9**

**Архитектура компьютеров**

Эйвази Мани НПИбд-03-24

# 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 . . . . .	9
2.3	Задание для самостоятельной работы . . . . .	19
<b>3</b>	<b>Выводы</b>	<b>26</b>

## Список иллюстраций

2.1	Текст программы lab9-1.asm . . . . .	7
2.2	Запуск программы lab9-1.asm . . . . .	7
2.3	Модифицированная программа lab9-1.asm . . . . .	8
2.4	Запуск модифицированной программы lab9-1.asm . . . . .	9
2.5	Код программы lab9-2.asm . . . . .	10
2.6	Запуск программы lab9-2.asm в GDB . . . . .	11
2.7	Дизассемблированный код программы . . . . .	12
2.8	Дизассемблированный код в Intel-синтаксисе . . . . .	13
2.9	Настройка точки останова . . . . .	14
2.10	Отслеживание изменений регистров . . . . .	15
2.11	Детальный анализ регистров . . . . .	16
2.12	Изменение значения переменной msg1 . . . . .	17
2.13	Просмотр регистра после изменений . . . . .	18
2.14	Анализ стека программы . . . . .	19
2.15	Код программы lab9-prog.asm . . . . .	20
2.16	Запуск программы lab9-prog.asm . . . . .	21
2.17	Код с ошибкой . . . . .	22
2.18	Процесс отладки программы . . . . .	23
2.19	Исправленный код программы . . . . .	24
2.20	Проверка исправленного кода . . . . .	25

## 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. Затем я создал файл с именем lab9-1.asm.

В качестве примера была рассмотрена программа, которая вычисляет арифметическое выражение  $f(x) = 2x + 7$  с использованием подпрограммы calcul. Значение переменной  $x$  вводится с клавиатуры, а вычисление производится внутри подпрограммы. (рис. 2.1) (рис. 2.2)

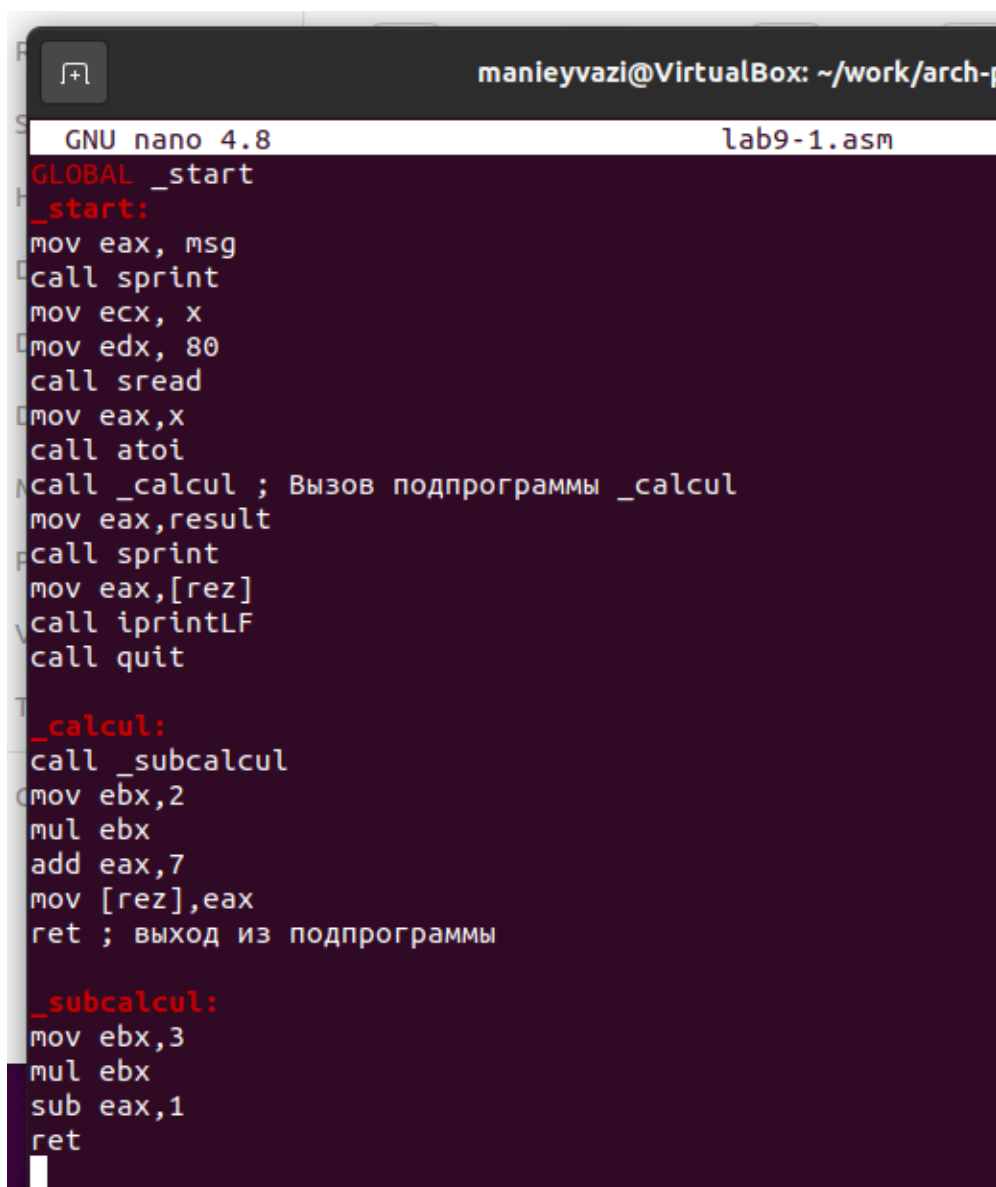
```
manieyvazi@VirtualBox: ~/work/arch-pc/lab
GNU nano 4.8 lab9-1.asm
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов 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: Текст программы lab9-1.asm

```
manieyvazi@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
manieyvazi@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-1.o -o lab9-1
manieyvazi@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 6
2x+7=19
manieyvazi@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 9
2x+7=25
manieyvazi@VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.2: Запуск программы lab9-1.asm

Далее я модифицировал программу, добавив подпрограмму `subcalcul` внутрь подпрограммы `calcul`. Это позволило вычислять составное выражение  $f(g(x))$ , где  $f(x) = 2x + 7$ , а  $g(x) = 3x - 1$ . Значение  $x$  вводится с клавиатуры. (рис. 2.3) (рис. 2.4)



```
manieyvazi@VirtualBox: ~/work/arch-p
GNU nano 4.8 lab9-1.asm
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: Модифицированная программа lab9-1.asm

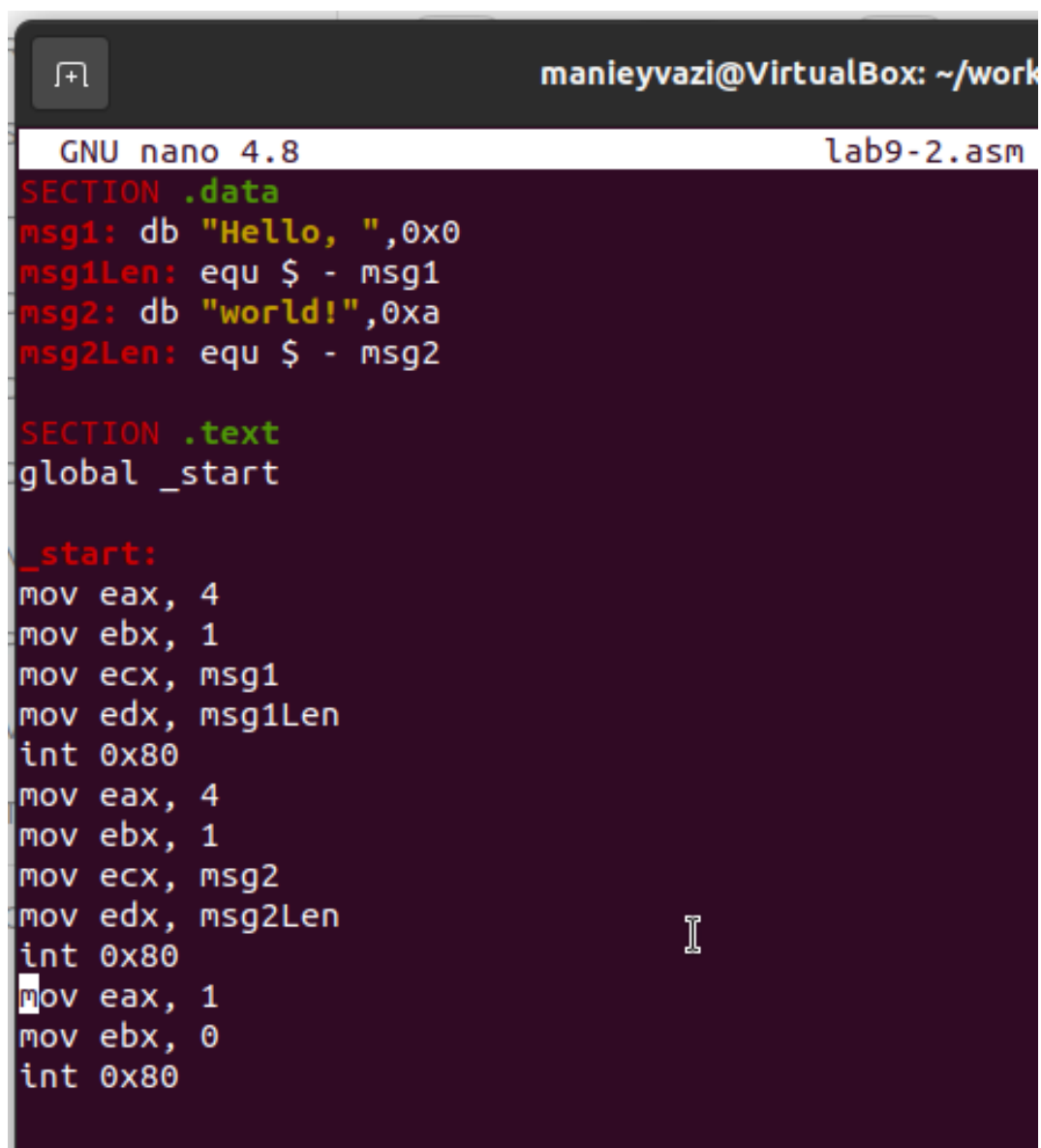


```
manleyvazi@VirtualBox:~/work/arch-pc/lab09$  
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm  
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-1.o -o lab9-1  
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 6  
2(3x-1)+7=41  
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 9  
2(3x-1)+7=59  
manleyvazi@VirtualBox:~/work/arch-pc/lab09$
```

Рис. 2.4: Запуск модифицированной программы lab9-1.asm

## 2.2 Отладка программы с помощью GDB

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



```
manieyvazi@VirtualBox: ~/work
GNU nano 4.8 lab9-2.asm
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

После компиляции с ключом -g для добавления отладочной информации я загрузил исполняемый файл в GDB. Запустил программу с помощью команды run или r. (рис. 2.6)

```

manieyvazi@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
manieyvazi@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
manieyvazi@VirtualBox:~/work/arch-pc/lab09$ gdb lab9-2

GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2
Copyright (C) 2020 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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) run
Starting program: /home/manieyvazi/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 4763) exited normally]
(gdb)

```

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

Для анализа программы я установил точку остановки на метке `_start` и запустил выполнение. Затем изучил дизассемблированный код программы. (рис. 2.7) (рис. 2.8)

```
manleyvazi@VirtualBox: ~/work/arch-pc/lab09
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2
Copyright (C) 2020 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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) run
Starting program: /home/manleyvazi/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 4763) exited normally]
(gdb) break _start
Breakpoint 1 at 0x08049000
(gdb) run
Starting program: /home/manleyvazi/work/arch-pc/lab09/lab9-2

Breakpoint 1, 0x08049000 in _start ()
(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 программы

```
manieyvazi@VirtualBox: ~/work/arch-pc/lab09

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)
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 Intel-синтаксисе

Для проверки точки останова я использовал команду `info breakpoints (i b)`. Установил дополнительную точку останова по адресу инструкции `mov ebx, 0x0`. (рис. 2.9)

```
manleyvazi@VirtualBox: ~/work/arch-pc/lab09

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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
b+ 0x8049031 <_start+49> mov    ebx,0x0
0x8049036 <_start+54> int    0x80

native process 4778 In: _start L?? PC: 0x8049000
(gdb) layout regs
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031
(gdb) i b
Num    Type           Disp Enb Address      What
1      breakpoint      keep y   0x08049000  <_start>
      breakpoint already hit 1 time
2      breakpoint      keep y   0x08049031  <_start+49>
(gdb) |
```

Рис. 2.9: Настройка точки останова

С помощью команды `stepi (si)` выполнил пошаговую отладку, отслеживая изменения регистров. (рис. 2.10) (рис. 2.11)

```
manleyvazi@VirtualBox: ~/work/arch-pc/lab09

Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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
b+ 0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80

native process 4778 In: _start L?? PC: 0x8049005
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si
0x08049005 in _start ()
(gdb) 
```

Рис. 2.10: Отслеживание изменений регистров

```
manieyvazi@VirtualBox: ~/work/arch-pc/lab09

Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049020 0x8049020 <_start+32>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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,0x804a000
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

native process 4778 In: start L?? PC: 0x8049020
(gdb) si
0x08049005 in _start ()
0x0804900a in _start ()
(gdb) si
0x0804900f in _start ()
(gdb) si
0x08049014 in _start ()
(gdb) si
0x08049016 in _start ()
(gdb) si
0x0804901b in _start ()
(gdb) si
0x08049020 in _start ()
(gdb) 
```

Рис. 2.11: Детальный анализ регистров

Я также просмотрел значение переменной `msg1` по имени и изменил первый символ переменной с помощью команды `set`. (рис. 2.12) (рис. 2.13)



```
manleyvazi@VirtualBox: ~/work/arch-pc/lab09

Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049020 0x8049020 <_start+32>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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
b+ 0x8049031 <_start+49> mov     ebx,0x0
0x8049036 <_start+54>   int     0x80

native process 4778 In: _start L?? PC: 0x8049020
0x804901b in _start ()
(gdb) si
0x8049020 in _start ()
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n"
(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"
(gdb) 
```

Рис. 2.12: Изменение значения переменной msg1

```
manleyvazi@VirtualBox: ~/work/arch-pc/lab09

Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049020 0x8049020 < _start+32>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

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,0x804a000
    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

native process.4778 In: _start L?? PC: 0x8049020
$1 = 4
(gdb) p/t $eax
$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: Просмотр регистра после изменений

Для проверки программы с аргументами я скопировал файл lab8-2.asm из лабораторной работы №8, создал исполняемый файл и загрузил его в GDB с помощью ключа -args. Затем исследовал стек, где хранились адреса аргументов. (рис. 2.14)

```
manleyvazi@VirtualBox: ~/work/arch-pc/lab09
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ gdb --args lab9-3 argument 1 argument 2 'argument 3'

GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.2) 9.2
Copyright (C) 2020 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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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
(gdb) run
Starting program: /home/manleyvazi/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

Breakpoint 1, 0x80490e8 in _start ()
(gdb) x/x $esp
0xffffd190: 0x00000006
(gdb)
0xffffd194: 0xffffd357
(gdb) x/s *(void**)(esp + 4)
0xffffd357: "/home/manleyvazi/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd382: "argument"
(gdb) x/s *(void**)(esp + 12)
0xffffd38b: "1"
(gdb) x/s *(void**)(esp + 16)
0xffffd38d: "argument"
(gdb) x/s *(void**)(esp + 20)
0xffffd396: "2"
(gdb) x/s *(void**)(esp + 24)
0xffffd398: "argument 3"
(gdb) 
```

Рис. 2.14: Анализ стека программы

## 2.3 Задание для самостоятельной работы

Я модифицировал программу из лабораторной работы №8, добавив вычисление функции  $f(x)$  в виде подпрограммы. (рис. 2.15) (рис. 2.16)

```
manieyvazi@VirtualBox: ~/work/arch-p
GNU nano 4.8 lab9-prog.asm
mov eax, fx
call sprintf
pop ecx
pop edx
sub ecx,1
mov esi, 0

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

loop next

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

prog:
mov ebx,2
mul ebx
add eax,7
ret
```

Рис. 2.15: Код программы lab9-prog.asm

```
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-prog.asm
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-prog.o -o lab9-p
rog
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ ./lab9-prog
f(x)= 7 + 2x
Результат: 0
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ ./lab9-prog 3
f(x)= 7 + 2x
Результат: 13
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ ./lab9-prog 3 6 7 9
f(x)= 7 + 2x
Результат: 78
manleyvazi@VirtualBox:~/work/arch-pc/lab09$ █
```

Рис. 2.16: Запуск программы lab9-prog.asm

При запуске программы я обнаружил ошибку: результат вычислений был неверным. Анализ с помощью GDB показал, что аргументы инструкции add перепутаны, а по окончании программы значение регистра ebx вместо eax отправляется в edi. (рис. 2.17) (рис. 2.18)

```
manieyvazi@VirtualBox: ~/work
GNU nano 4.8 lab9-prog2.as
#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. 2.17: Код с ошибкой

```
manieyvazi@VirtualBox: ~/work/arch-pc/lab09

eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd1d0 0xffffd1d0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490fe 0x80490fe <_start+22>
eflags   0x206    [ PF IF ]
cs       0x23     35

B+ 0x80490e8 <_start>      mov     ebx,0x3
B+ 0x80490e8 <_start+5>    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,0x5
>0x80490fb <_start+19>     add     ebx,0x5
0x80490fe <_start+22>     mov     edi,ebx04a000
0x8049100 <_start+24>     mov     eax,0x804a000rint>
0x8049105 <_start+29>     call    0x804900f <sprint>
0x804910a <_start+34>     mov     eax,edi86 <iprintLF>
0x804910c <_start+36>     call    0x8049086 <iprintLF>

native process 4840 In: _start L?? PC: 0x80490fe
(gdb) sNo process In: L?? PC: ??
(gdb) si
0x080490f4 in _start ()
(gdb) si
0x080490f9 in _start ()
(gdb) si
0x080490fb in _start ()
(gdb) si
0x080490fe in _start ()
(gdb) c
Continuing.
Результат: 10
[Inferior 1 (process 4840) exited normally]
(gdb) 
```

Рис. 2.18: Процесс отладки программы

После исправления ошибок я проверил работу программы. (рис. 2.19) (рис. 2.20)

```
manieyvazi@VirtualBox: ~/wo
GNU nano 4.8 lab9-prog2.a
#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. 2.19: Исправленный код программы



```
manleyvazi@VirtualBox: ~/work/arch-pc/lab09

eax      0x19      25
ecx      0x4       4
edx      0x0       0
ebx      0x3       3
esp      0xffffd1d0 0xffffd1d0
ebp      0x0       0x0
esi      0x0       0
edi      0x0       0
eip      0x80490fe 0x80490fe <_start+22>
eflags   0x202     [ IF ]
cs       0x23      35

B+ 0x80490e8 <_start>      mov     ebx,0x3
B+ 0x80490e8 <_start+5>    mov     ebx,0x3
0x80490ed <_start+5>      mov     eax,0x2
0x80490f2 <_start+10>     add     eax,ebx
0x80490f4 <_start+12>     mov     ecx,0x4
0x80490f9 <_start+17>     mul     ecx,0x5
>0x80490fb <_start+19>     add     eax,0x5
0x80490fe <_start+22>     mov     edi,eax04a000
0x8049100 <_start+24>     mov     eax,0x804a000rint>
0x8049105 <_start+29>     call    0x804900f <sprint>
0x804910a <_start+34>     mov     eax,edi86 <iprintLF>
0x804910c <_start+36>     call    0x8049086 <iprintLF>

native process 9480 In: _start L?? PC: 0x80490fe
(gdb) sNo process In: L?? PC: ??
(gdb) si
0x080490f4 in _start ()
(gdb) si
0x080490f9 in _start ()
(gdb) si
0x080490fb in _start ()
(gdb) si
0x080490fe in _start ()
(gdb) c
Continuing.
Результат: 25
[Inferior 1 (process 9480) exited normally]
(gdb) █
```

Рис. 2.20: Проверка исправленного кода

## 3 Выводы

Я освоил работу с подпрограммами и отладчиком GDB, научился находить и исправлять ошибки в коде с помощью анализа стеков, регистров и дизассемблированного кода.