

Отчёт по лабораторной работе 9

Архитектура компьютеров

Иван Горбунов

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Выводы	26

Список иллюстраций

2.1 Программа в файле lab9-1.asm	7
2.2 Запуск программы lab9-1.asm	7
2.3 Программа в файле lab9-1.asm	8
2.4 Запуск программы lab9-1.asm	9
2.5 Программа в файле lab9-2.asm	10
2.6 Запуск программы lab9-2.asm в отладчике	11
2.7 Дизассимилированный код	12
2.8 Дизассимилированный код в режиме интел	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 в файле lab9-4.asm	21
2.17 Запуск программы lab9-4.asm	21
2.18 Код с ошибкой	22
2.19 Отладка	23
2.20 Код исправлен	24
2.21 Проверка работы	25

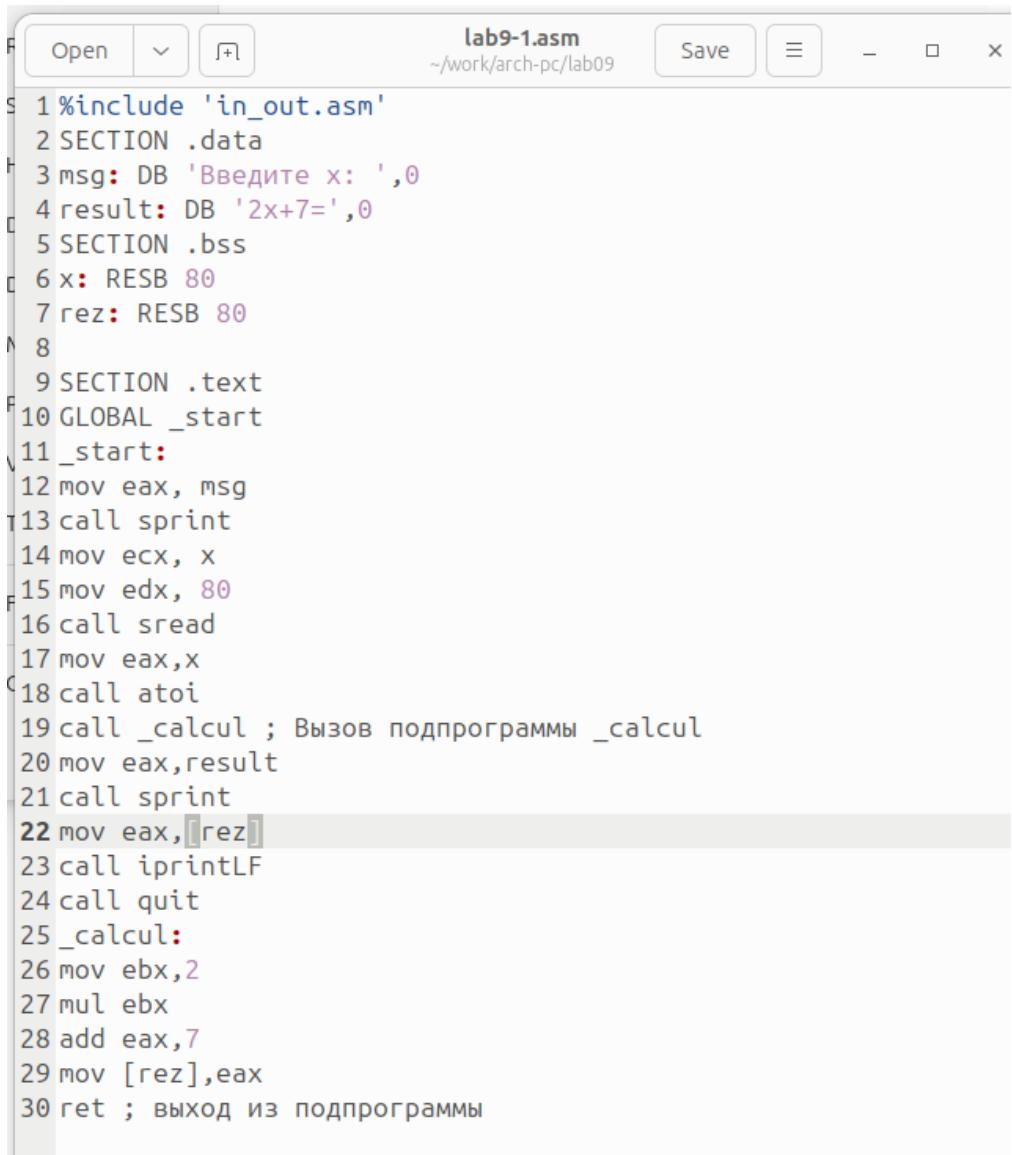
Список таблиц

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

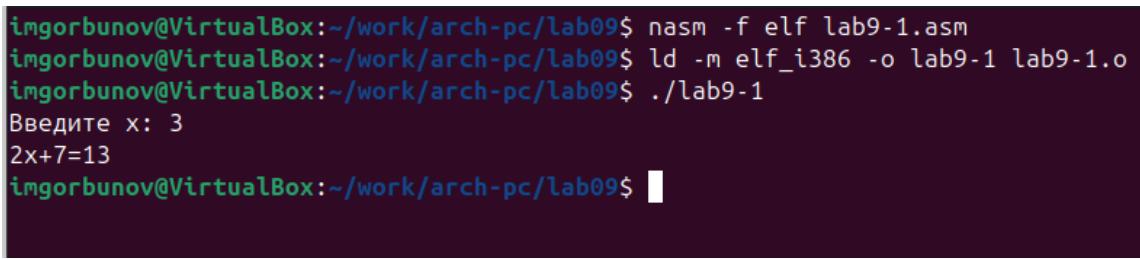
1. Создал каталог для выполнения лабораторной работы № 9, перешел в него и создал файл lab9-1.asm.
2. В качестве примера рассмотрим программу вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.



The screenshot shows a text editor window with the following assembly code:

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 result: DB '2x+7=',0
5 SECTION .bss
6 x: RESB 80
7 rez: RESB 80
8
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax,x
18 call atoi
19 call _calcul ; Вызов подпрограммы _calcul
20 mov eax,result
21 call sprint
22 mov eax,[rez]
23 call iprintLF
24 call quit
25 _calcul:
26 mov ebx,2
27 mul ebx
28 add eax,7
29 mov [rez],eax
30 ret ; выход из подпрограммы
```

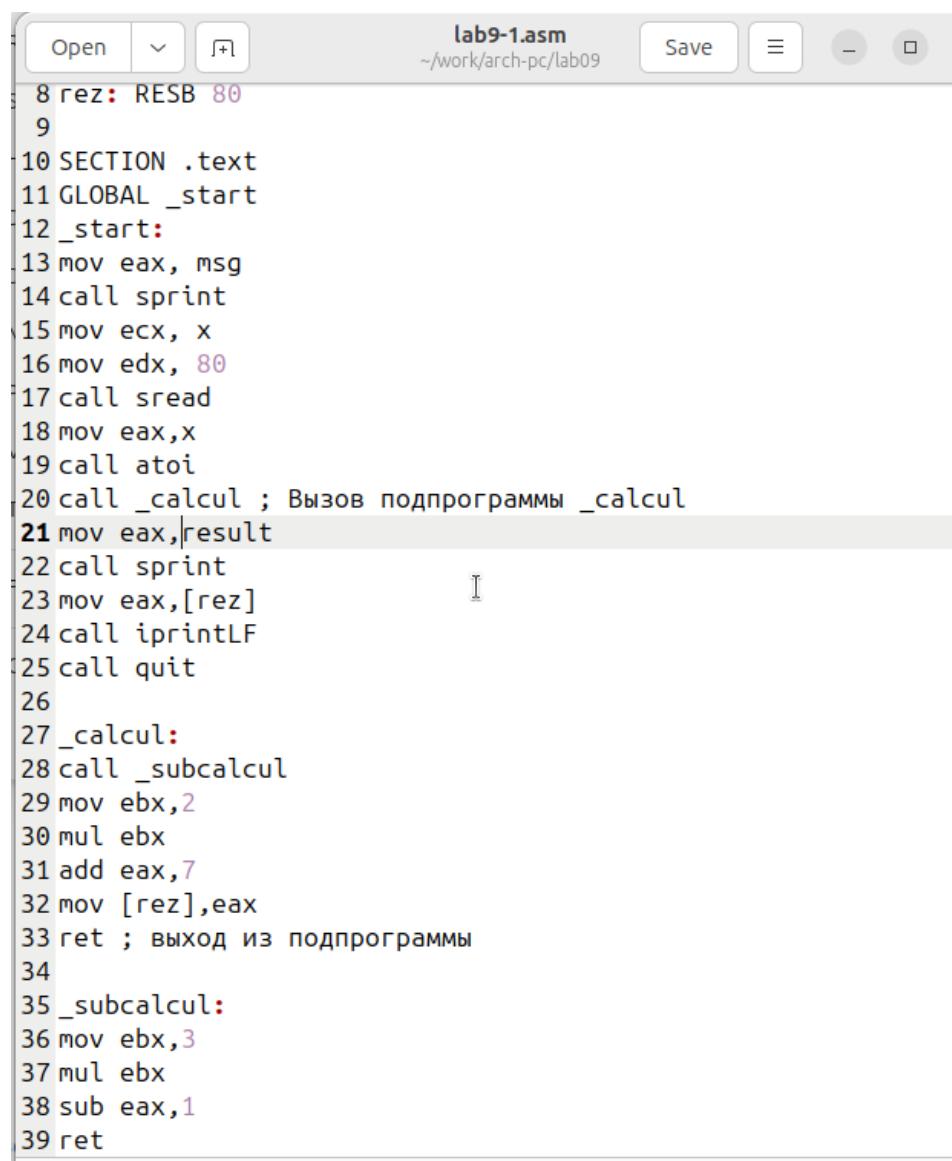
Рисунок 2.1: Программа в файле lab9-1.asm



```
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 3
2x+7=13
imgorbunov@VirtualBox:~/work/arch-pc/lab09$
```

Рисунок 2.2: Запуск программы lab9-1.asm

3. Изменил текст программы, добавив подпрограмму subcalcul в подпрограмму calcul, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$.



The screenshot shows a text editor window with the following details:

- File menu: Open, Save, etc.
- Title bar: lab9-1.asm, ~/work/arch-pc/lab09
- Code area:

```
8 rez: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprint
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax,x
19 call atoi
20 call _calcul ; Вызов подпрограммы _calcul
21 mov eax,[rez]
22 call sprint
23 mov eax,[rez]
24 call iprintLF
25 call quit
26
27 _calcul:
28 call _subcalcul
29 mov ebx,2
30 mul ebx
31 add eax,7
32 mov [rez],eax
33 ret ; выход из подпрограммы
34
35 _subcalcul:
36 mov ebx,3
37 mul ebx
38 sub eax,1
39 ret
```

Рисунок 2.3: Программа в файле lab9-1.asm

```
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Ведите x: 3
2(3x-1)+7=23
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ 
```

Рисунок 2.4: Запуск программы lab9-1.asm

4. Создал файл lab9-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!).

The screenshot shows a text editor window with the following assembly code:

```
1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1Len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2Len: equ $ - msg2
6
7 SECTION .text
8 global _start
9
10 _start:
11 mov eax, 4
12 mov ebx, 1
13 mov ecx, msg1
14 mov edx, msg1Len
15 int 0x80
16 mov eax, 4|           ; Cursor is here
17 mov ebx, 1
18 mov ecx, msg2
19 mov edx, msg2Len
20 int 0x80
21 mov eax, 1
22 mov ebx, 0
23 int 0x80
```

Рисунок 2.5: Программа в файле lab9-2.asm

Получил исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом „-g“.

Загрузил исполняемый файл в отладчик gdb. Проверил работу программы, запустив ее в оболочке GDB с помощью команды run (сокращённо r).

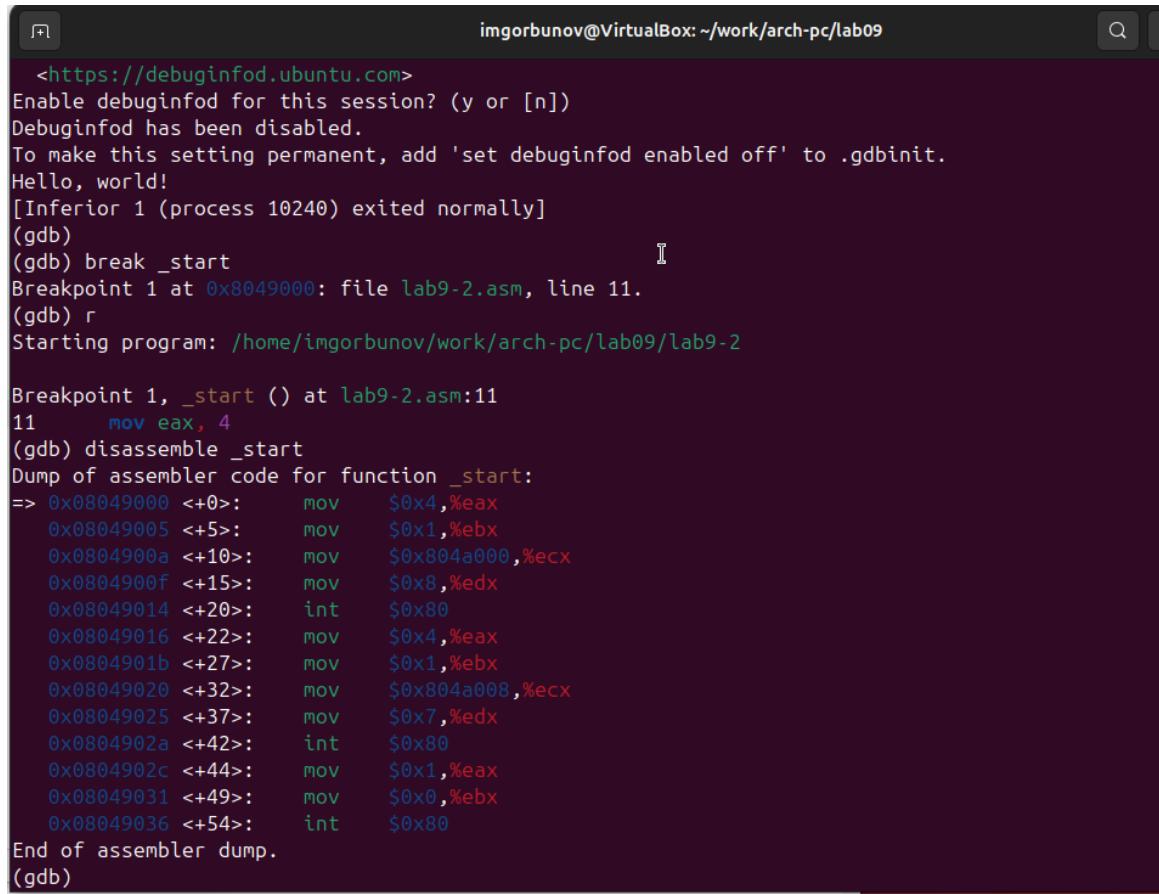
```
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-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/imgorbunov/work/arch-pc/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
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 10240) exited normally]
(gdb) █
```

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

Для более подробного анализа программы установите брейкпоинт на метку start, с которой начинается выполнение любой ассемблерной программы, и запустите её. Посмотрите дисассимилированный код программы.



The screenshot shows a terminal window with the following content:

```
<https://debuginfod.ubuntu.com>
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 10240) exited normally]
(gdb)
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 11.
(gdb) r
Starting program: /home/imgorbunov/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)
```

Рисунок 2.7: Дизассимилированный код

```
imgorbunov@VirtualBox: ~/work/arch-pc/lab09
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)
```

Рисунок 2.8: Дизассимилированный код в режиме интел

На предыдущих шагах была установлена точка остановки по имени метки (_start). Проверил это с помощью команды info breakpoints (кратко i b). Установил еще одну точку остановки по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определил адрес предпоследней инструкции (mov ebx,0x0) и установил точку.

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp 0xfffffcee0 0xfffffcee0
ebp      0x0      0x0
esi      0x0      0
edi      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

native process 10254 (asm) In: _start
(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)
```

Рисунок 2.9: Точка остановки

Отладчик может показывать содержимое ячеек памяти и регистров, а при необходимости позволяет вручную изменять значения регистров и переменных. Выполнил 5 инструкций с помощью команды stepi (или si) и проследил за изменением значений регистров.

The screenshot shows a terminal window with the following content:

```
Register group: general
eax      0x4          4
ecx      0x0          0
edx      0x0          0
ebx      0x0          0
esp      0xfffffcee0  0xfffffcee0
ebp      0x0          0x0
esi      0x0          0
edi      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

native process 10254 (asm) In: _start
--Type <RET> for more, q to quit, c to continue without paging--
eflags    0x202        [ IF ]
cs        0x23         35
ss        0x2b         43
ds        0x2b         43
es        0x2b         43
fs        0x0          0
gs        0x0          0
(gdb) si
(gdb) █
```

Рисунок 2.10: Изменение регистров

The screenshot shows a terminal window with the following content:

```
Register group: general
eax      0x8          8
ecx      0x804a000    134520832
edx      0x8          8
ebx      0x1          1
esp      0xfffffcee0  0xfffffcee0
ebp      0x0          0x0
esi      0x0          0
edi      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

native process 10254 (asm) In: _start
ds      0x2b          43
es      0x2b          43
fs      0x0           0
gs      0x0           0
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) █
```

Рисунок 2.11: Изменение регистров

Посмотрел значение переменной msg1 по имени. Посмотрел значение переменной msg2 по адресу.

Изменить значение для регистра или ячейки памяти можно с помощью команды set, задав ей в качестве аргумента имя регистра или адрес. Изменил первый символ переменной msg1.

The screenshot shows a terminal window with the following content:

```
Register group: general
eax      0x8          8
ecx      0x804a000    134520832
edx      0x8          8
ebx      0x1          1
esp      0xfffffcee0  0xfffffcee0
ebp      0x0          0x0
esi      0x0          0
edi      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

native process 10254 (asm) In: _start
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>:      "Lord!\n\034"
(gdb)
```

Рисунок 2.12: Изменение значения переменной

Вывел в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx.

The screenshot shows a terminal window with the following content:

```
Register group: general
eax          0x8          8
ecx          0x804a000    134520832
edx          0x8          8
ebx          0x1          1
esp          0xfffffcee0  0xfffffcee0
ebp          0x0          0x0
esi          0x0          0
edi          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

native process 10254 (asm) In: _start
$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)
```

Рисунок 2.13: Вывод значения регистра

С помощью команды set изменил значение регистра ebx

```

Register group: general
eax          0x8          8
ecx          0x804a000    134520832
edx          0x8          8
ebx          0x2          2
esp          0xfffffcee0  0xfffffcee0
ebp          0x0          0x0
esi          0x0          0
edi          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

native process 10254 (asm) In: _start
$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)

```

Рисунок 2.14: Вывод значения регистра

5. Скопировал файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки. Создал исполняемый файл. Для загрузки в gdb программы с аргументами необходимо использовать ключ –args. Загрузил исполняемый файл в отладчик, указав аргументы.

Для начала установил точку останова перед первой инструкцией в программе и запустил ее.

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5 – это имя программы lab9-3 и непосредственно аргументы: аргумент1, аргумент, 2 и „аргумент 3“.

Посмотрел остальные позиции стека – по адресу [esp+4] располагается адрес в памяти где находится имя программы, по адресу [esp+8] храниться адрес

первого аргумента, по адресу [esp+12] – второго и т.д.

```
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/imgorbunov/work/arch-pc/lab09/lab9-3 argument 1 argument 2 argument\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
 <https://debuginfod.ubuntu.com>
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
0xfffffcceb0: 0x00000006
(gdb) x/s *(void**)( $esp + 4)
0xfffffd088: "/home/imgorbunov/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)( $esp + 8)
0xfffffd0b3: "argument"
(gdb) x/s *(void**)( $esp + 12)
0xfffffd0bc: "1"
(gdb) x/s *(void**)( $esp + 16)
0xfffffd0be: "argument"
(gdb) x/s *(void**)( $esp + 20)
0xfffffd0c7: "2"
(gdb) x/s *(void**)( $esp + 24)
0xfffffd0c9: "argument 3"
(gdb)
```

Рисунок 2.15: Вывод значения регистра

Объясню, почему шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] - шаг равен размеру переменной - 4 байтам.

6. Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму.

The screenshot shows a text editor window titled "lab9-4.asm" with the file path "~/work/arch-pc/lab09". The code is written in NASM assembly language. It defines a global variable _start, initializes EAX to 10, and then enters a loop where it calculates $f(x) = 10x - 5$ for values from 1 to 4. The program then prints the results and exits.

```
6 SECTION .text
7 global _start
8 _start:
9 mov eax, fx
10 call sprintLF
11 pop ecx
12 pop edx
13 sub ecx, 1
14 mov esi, 0
15
16 next:
17 cmp ecx, 0h
18 jz _end
19 pop eax
20 call atoi
21 call calc
22 add esi, eax
23
24 loop next
25
26 _end:
27 mov eax, msg
28 call sprint
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 calc:
34 mov ebx, 10
35 mul ebx
36 sub eax, 5
37 ret
```

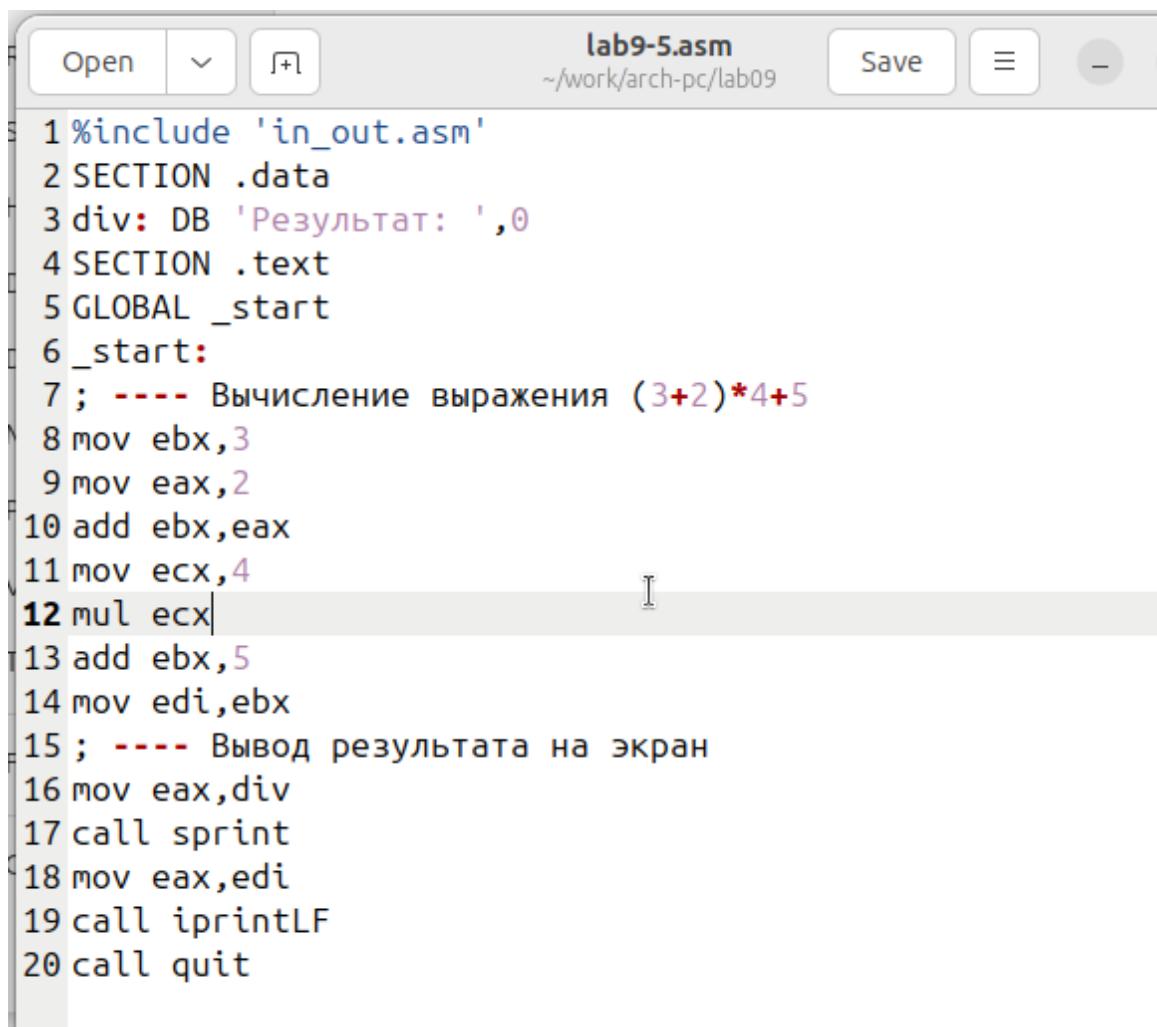
Рисунок 2.16: Программа в файле lab9-4.asm

```
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ 
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-4.asm
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-4.o -o lab9-4
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-4 1
f(x)= 10x - 5
Результат: 5
imgorbunov@VirtualBox:~/work/arch-pc/lab09$ ./lab9-4 1 2 3 4
f(x)= 10x - 5
Результат: 80
imgorbunov@VirtualBox:~/work/arch-pc/lab09$
```

Рисунок 2.17: Запуск программы lab9-4.asm

7. В листинге приведена программа вычисления выражения $(3 + 2) * 4 + 5$.

При запуске данная программа дает неверный результат. Проверил это. С помощью отладчика GDB, анализируя изменения значений регистров, определю ошибку и исправлю ее.



The screenshot shows a text editor window titled "lab9-5.asm" with the file path "~/work/arch-pc/lab09". The code is written in assembly language:

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ----- Вычисление выражения (3+2)*4+5
8 mov ebx,3
9 mov eax,2
10 add ebx,eax
11 mov ecx,4
12 mul ecx
13 add ebx,5
14 mov edi,ebx
15 ; ----- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit
```

The instruction at line 12, `mul ecx`, is highlighted with a gray background, indicating it is the current line of interest.

Рисунок 2.18: Код с ошибкой

The screenshot shows a terminal window with GDB running on a Linux system. The assembly code is displayed in the middle, with the instruction at address 0x8049105 highlighted in red. The registers are listed above the assembly code, and the stack dump is shown below.

Register	Value
eax	0x804a000
ecx	0x4
edx	0x0
ebx	0xa
esp	0xfffffcee0
ebp	0x0
esi	0x0
edi	0xa

```
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 <iprintfLF>
0x8049111 <_start+41>    call   0x80490db <quit>

native process 10427 (asm) In: _start
Breakpoint 1, _start () at lab9-5.asm:8
(gdb) si
(gdb)
```

Рисунок 2.19: Отладка

Отмечу, что перепутан порядок аргументов у инструкции add и что по окончании работы в edi отправляется ebx вместо eax

The screenshot shows a window titled "lab9-5.asm" with the path "~/work/arch-pc/lab09". The code is written in assembly language:

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 SECTION .text
5 GLOBAL _start
6 _start:
7 ; ----- Вычисление выражения (3+2)*4+5
8 mov ebx,3
9 mov eax,2
10 add eax,ebx
11 mov ecx,4
12 mul ecx
13 add eax,5
14 mov edi,eax
15 ; ----- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF      I
20 call quit|
```

Рисунок 2.20: Код исправлен

The screenshot shows a terminal window with the following content:

```
eax      s----- 25
         fffcee0      xffffcee0
                           [ Register Values Unavailable ]
[ ] 9 [ ] 5
0x8049100 <_start+24>    mul    eax,0x804a000
0x804910a <_start+34>    mov    eax,edi
0x804910c <_start+36>    call   0x8049086 <iprintLF>
0x8049111 <_start+41>    call   0x80490db <quit>

native process 10449 (asm) In: _start
(gdb) sNo process (asm) In:
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) c
Continuing.
Результат: 25
[Iinferior 1 (process 10449) exited normally]
(gdb)
```

Рисунок 2.21: Проверка работы

3 Выводы

Освоили работу с подпрограммами и отладчиком.