

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Юсупова Амина Руслановна

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

1	Цель работы	2
2	Задание	2
3	Теоретическое введение.....	2
4	Выполнение лабораторной работы	3
4.1	Реализация подпрограмм в NASM	3
4.1.1	Отладка программ с помощью GDB.....	7
4.1.2	Добавление точек останова.....	11
4.1.3	Работа с данными программы в GDB	12
4.1.4	Обработка аргументов командной строки в GDB.....	16
4.2	Задание для самостоятельной работы	17
5	Выводы	26
6	Список литературы	26

1 Цель работы

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

2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Самостоятельное выполнение заданий по материалам лабораторной работы

3 Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

- обнаружение ошибки;
- поиск её местонахождения;
- определение причины ошибки;

- исправление ошибки.

Можно выделить следующие типы ошибок:

- синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка;
- семантические ошибки — являются логическими и приводят к тому, что программа запускается, отработывает, но не даёт желаемого результата;
- ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

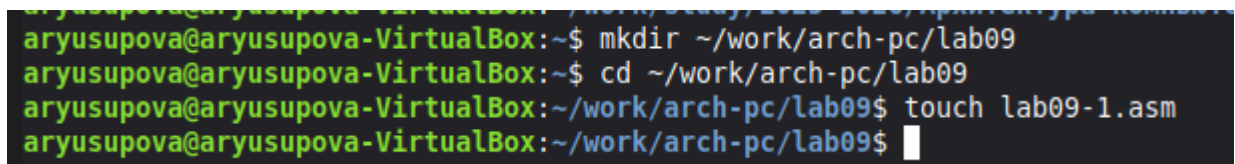
Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

4 Выполнение лабораторной работы

4.1 Реализация подпрограмм в NASM

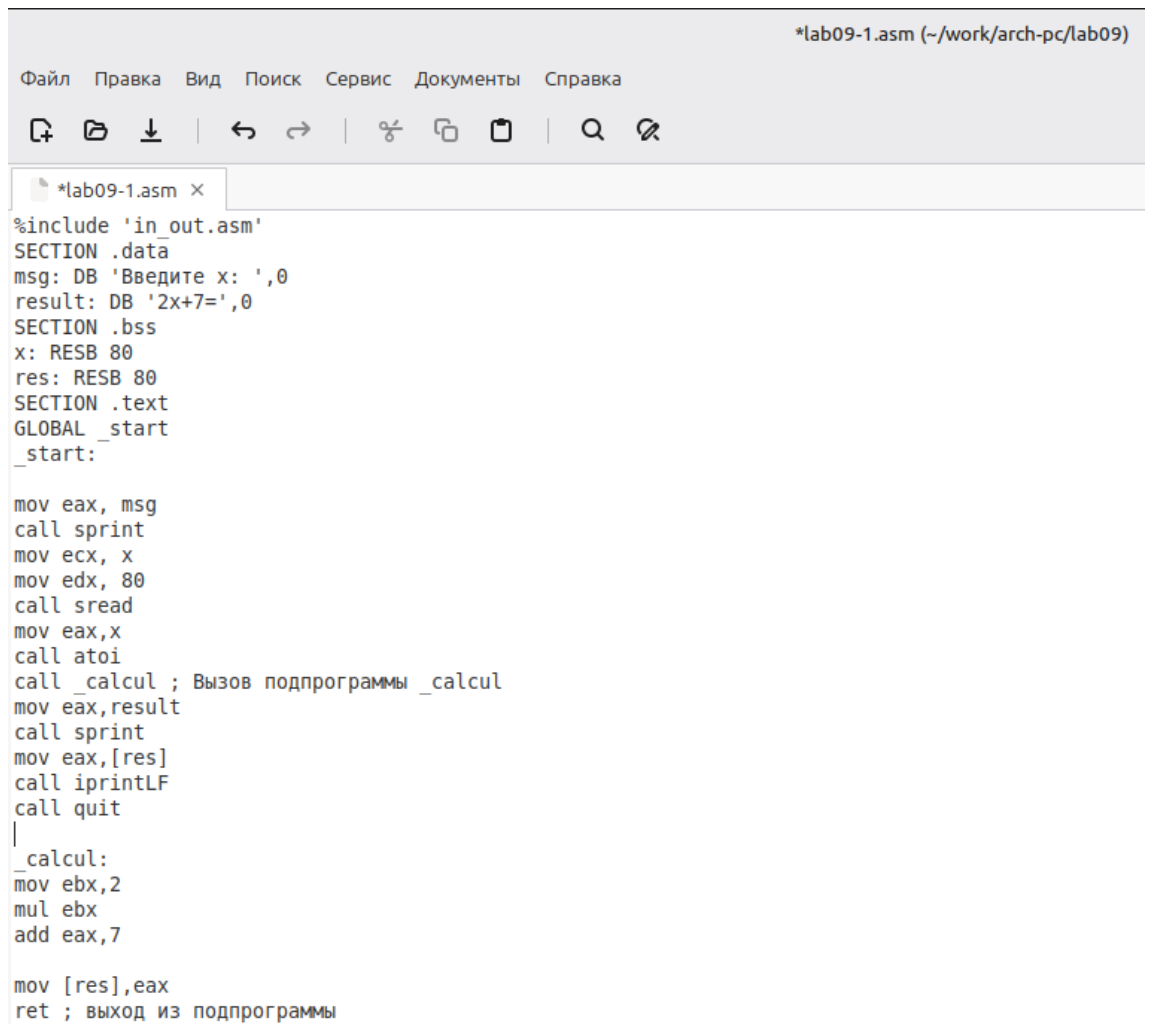
Создаю каталог lab09 для выполнения лабораторной работы №9, перехожу в него и создаю файл lab09-1.asm: (рис. 1).



```
aryusupova@aryusupova-VirtualBox:~$ mkdir ~/work/arch-pc/lab09
aryusupova@aryusupova-VirtualBox:~$ cd ~/work/arch-pc/lab09
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ touch lab09-1.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$
```

Рисунок 1: Создание рабочего каталога и переход в него

Копирую в файл код из листинга 9.1 (рис. 2), компилирую и запускаю его, данная программа выполняет вычисление функции (рис. 3).



```
*lab09-1.asm (~/work/arch-pc/lab09)

Файл  Правка  Вид  Поиск  Сервис  Документы  Справка

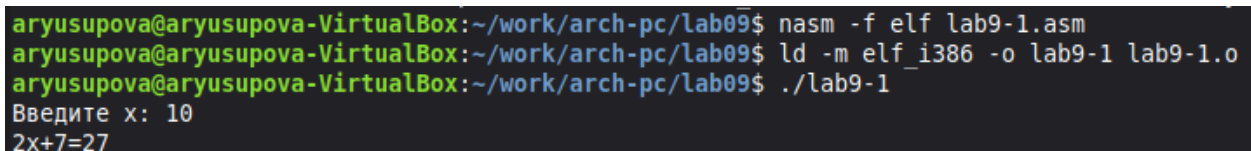
*lab09-1.asm x

%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:

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
|
_calcul:
mov ebx, 2
mul ebx
add eax, 7

mov [res], eax
ret ; выход из подпрограммы
```

Рисунок 2: Копирование кода из листинга 9.1

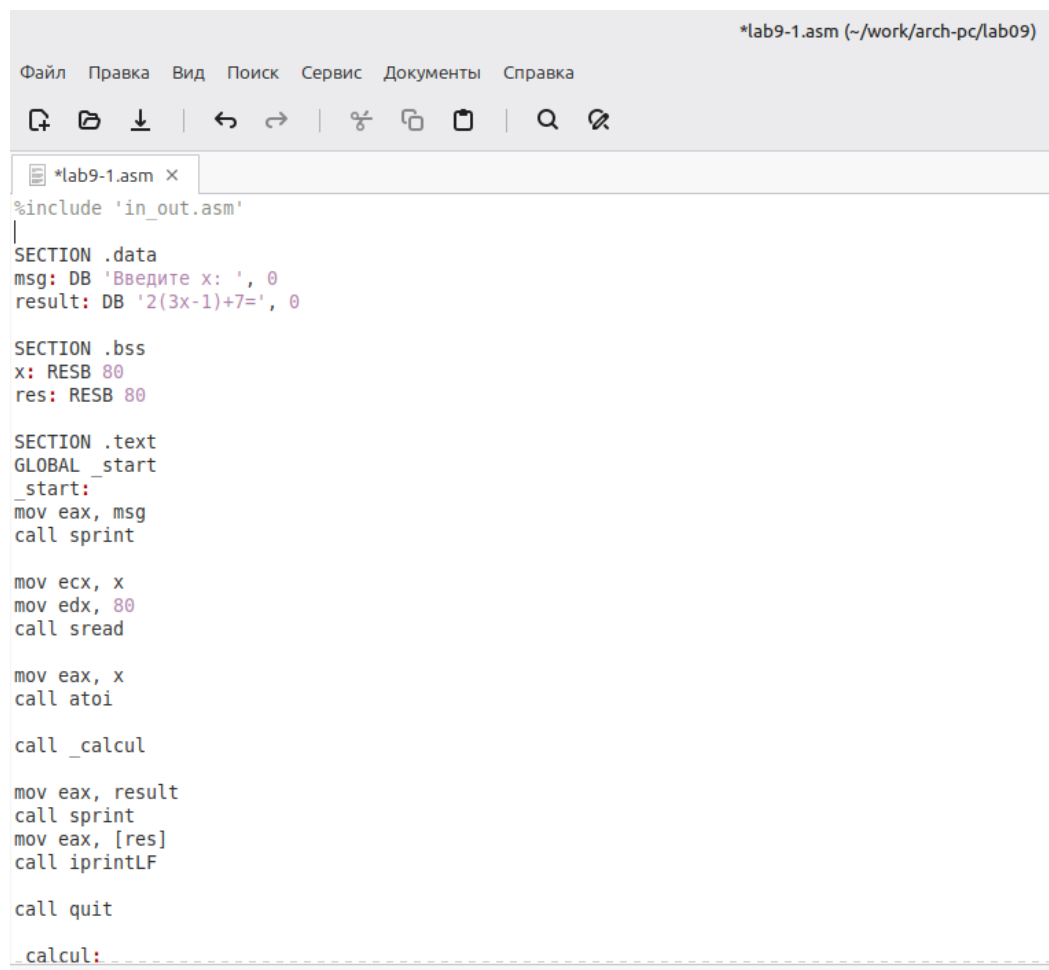


```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2x+7=27
```

Рисунок 3: Запуск программы lab9-1.asm

Изменяю текст программы, добавив в нее подпрограмму (рис. 4).

Компилирую и запускаю программу lab9-1.asm, теперь она вычисляет значение функции для выражения $f(g(x))$ (рис. 5).



```
*lab9-1.asm (~/work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[+] [F2] [F4] [F5] | [F6] [F7] | [F8] [F9] [F10] | [F11] [F12]
*lab9-1.asm x
%include 'in_out.asm'
|
SECTION .data
msg: DB 'Введите x: ', 0
result: DB '2(3x-1)+7=', 0

SECTION .bss
x: RESB 80
res: 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

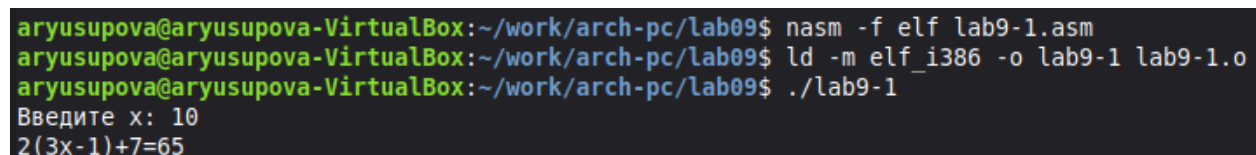
call _calcul

mov eax, result
call sprint
mov eax, [res]
call iprintLF

call quit

calcul:
-----
```

Рисунок 4: Изменение кода программы lab9-1.asm



```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2(3x-1)+7=65
```

Рисунок 5: Запуск изменённой программы lab9-1.asm

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Введите x: ', 0
```

```
result: DB '2(3x-1)+7=', 0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
res: 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

mov **eax**, result

call sprint

mov **eax**, [res]

call iprintLF

call quit

_calcul:

push **eax**

call _subcalcul

mov **ebx**, 2

mul **ebx**

add **eax**, 7

mov [res], **eax**

```
pop eax
```

```
ret
```

```
_subcalcul:
```

```
mov ebx, 3
```

```
mul ebx
```

```
sub eax, 1
```

```
ret
```

4.1.1 Отладка программ с помощью GDB

Создаю файл lab9-2.asm (рис.6), копирую программу листинга 9.2 (рис. 7), транслирую с созданием файла листинга и отладки, komponую и запускаю в отладчике (рис. 8).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ touch lab9-2.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$
```

Рисунок 6: Создание файла lab9-2.asm

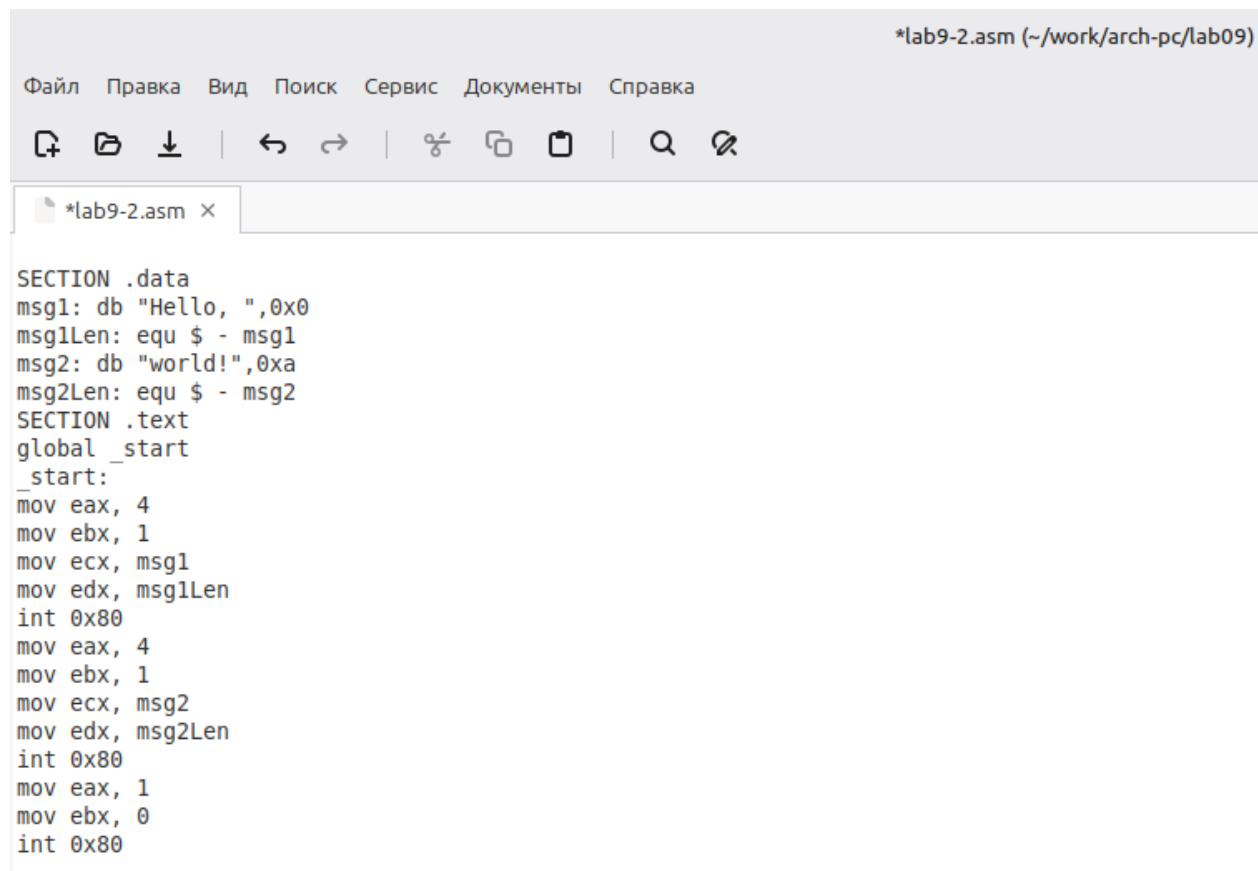


Рисунок 7: Копирование кода из листинга 9.2 в файл lab9-2.asm

```

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

```

Рисунок 8: Запуск программы lab9-2.asm

Запустив программу командой `run`, я убедилась в том, что она работает исправно (рис. 9).

```

aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
aryusupova@aryusupova-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) run
Starting program: /home/aryusupova/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 8997) exited normally]
(gdb)

```

Рисунок 9: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпоинт на метку `_start` и снова запускаю отладку (рис. 10).


```

aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
aryusupova@aryusupova-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) run
Starting program: /home/aryusupova/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 8997) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 10.
(gdb) run
Starting program: /home/aryusupova/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:10
10      mov eax, 4
(gdb)

```

Рисунок 10: Запуск отладчика с брейкпоинт

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel *amd топчик* (рис. 11).

Различия между синтаксисом АТТ и Intel заключаются в порядке операндов (АТТ - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (АТТ - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как ax, eax, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
Файл Правка Вид Поиск Терминал Справка
(gdb) run
Starting program: /home/aryusupova/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:10
10      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) █
```

Рисунок 11: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. 12, рис. 13).

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
Файл Правка Вид Поиск Терминал Справка
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
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
0x8049040 add BYTE PTR [eax],al
0x8049042 add BYTE PTR [eax],al
0x8049044 add BYTE PTR [eax],al
0x8049046 add BYTE PTR [eax],al
0x8049048 add BYTE PTR [eax],al
0x804904a add BYTE PTR [eax],al
native process 9090 (asm) In: start
(gdb) layout regs
```

Рисунок 12: Включение режима псевдографики

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
Файл Правка Вид Поиск Терминал Справка
--Register group: general
eax 0x0 0 ecx 0x0 0
edx 0x0 0 ebx 0x0 0
esp 0xffffd080 0xffffd080 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 9090 (asm) In: start
(gdb) layout regs
(gdb)
```

Рисунок 13: Режим псевдографики

4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 14).

```
File  Правка  Вид  Поиск  Терминал  Справка
--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd080  0xffffd080  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 9090 (asm) In: start
(gdb) layout regs
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab9-2.asm:10
breakpoint already hit 1 time
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 21.
```

Рисунок 14: Список брейкпоинтов

Установим еще одну точку останова по адресу инструкции (с помощью команды (gdb) break *) и посмотрим информацию о всех установленных точках останова (с помощью команды i b) (рис. 15).

```
File  Правка  Вид  Поиск  Терминал  Справка
--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd080  0xffffd080  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 9090 (asm) In: start
(gdb) layout regs
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab9-2.asm:10
breakpoint already hit 1 time
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 21.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab9-2.asm:10
breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab9-2.asm:21
(gdb)
```

Рисунок 15: Добавление второй точки останова

4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой info registers (рис. 16).

```

aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09

--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd070 0xffffd070  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

lab9-2.asm
1
2 SECTION .data
3 msg1: db "Hello, ",0x0
4 msg1Len: equ $ - msg1
5 msg2: db "world!\n",0xa
6 msg2Len: equ $ - msg2
7 SECTION .text
8 global _start
9 _start
B+> 10 mov eax, 4
11 mov ebx, 1
12 mov ecx, msg1

native process 16104 (src) In: _start
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd070 0xffffd070
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--

```

Рисунок 16: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. 17).

```

aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09

--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd070 0xffffd070  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

lab9-2.asm
1
2 SECTION .data
3 msg1: db "Hello, ",0x0
4 msg1Len: equ $ - msg1
5 msg2: db "world!\n",0xa
6 msg2Len: equ $ - msg2
7 SECTION .text
8 global _start
9 _start
B+> 10 mov eax, 4
11 mov ebx, 1

native process 16104 (src) In: _start
--Type <RET> for more, q to quit, c to continue without paging--
ds       0x2b     43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)

```

Рисунок 17: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. 18).

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd070 0xffffd070  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

lab9-2.asm
1
2 SECTION data
3 msg1: db "Hello, ",0x0
4 msg1Len: equ $ - msg1
5 msg2: db "world!",0xa
6 msg2Len: equ $ - msg2
7 SECTION text
8 global _start
9 _start:
B+> 10 mov eax, 4
    11 mov ebx, 1

native process 16104 (src) In: start
es      0x2b     43
fs      0x0      0
gs      0x0      0
(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}&msg2='x'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "xorld!\n\034"
(gdb) █
```

Рисунок 18: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. 19). Выводит нули, не совсем понимаю по какой причине.

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd070 0xffffd070  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

lab9-2.asm
1
2 SECTION data
3 msg1: db "Hello, ",0x0
4 msg1Len: equ $ - msg1
5 msg2: db "world!",0xa
6 msg2Len: equ $ - msg2
7 SECTION text
8 global _start
9 _start:
B+> 10 mov eax, 4
    11 mov ebx, 1

native process 16104 (src) In: start
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "xorld!\n\034"
(gdb) p/t $ecx
$1 = 0
(gdb) print /t $ecx
$2 = 0
(gdb) p /s $edx
$3 = 0
(gdb) p/t $edx
$4 = 0
(gdb) p/x $edx
$5 = 0x0
(gdb) █
```

Рисунок 19: Просмотр значения регистра разными представлениями

С помощью команды set меняю содержимое регистра ebx (рис. 20).

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

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x2      2
esp      0xffffd070 0xffffd070  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

lab9-2.asm
1
2 SECTION data
3 msg1: db "Hello, ",0x0
4 msg1len: equ $ - msg1
5 msg2: db "world!",0xa
6 msg2len: equ $ - msg2
7 SECTION text
8 global _start
9 _start
B+> 10 mov eax, 4
    11 mov ebx, 1

native process 16104 (src) In: _start
$3 = 0
(gdb) p/t $edx
$4 = 0
(gdb) p/x $edx
$5 = 0x0
(gdb) set $ebx='2'
(gdb) p/s
$6 = 0
(gdb) p/s $ebx
$7 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$8 = 2
(gdb)
```

Рисунок 20: Пример использования команды set

p/s \$ebx: Интерпретирует значение регистра как указатель на строку (адрес в памяти) и выводит строку, начиная с этого адреса, до нулевого терминатора.

Разница в выводе (50 vs 2) обусловлена разницей между ASCII-кодом символа '2' и числом 2.

Завершаю выполнение программы с помощью команды continue (сокращенно c) и выхожу из GDB с помощью команды quit (сокращенно q) (рис. 21).

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
eax    0x0      0      ecx    0x0      0
eax    0x1      1      ecx    0x804a008  134520040
edx    0x7      7      ebp    0x0      0
esi    0x0      0      edi    0x0      0
eip    0x8049000 0x8049000 < start>  eflags 0x202    [ IF ]
eip    0x8049031 0x8049031 < start+49>  ss      0x2b     43
ds      0x2b     43      es      0x2b     43
fs      0x0      0      gs      0x0      0

lab9-2.asm
1
17 mov ecx, msg;
18 mov edx, msg.len;
19 int 0x80
20 mov eax, 1
B+> 21 mov ebx, 0
22 int 0x80
B+> 1 .4

native process 16104 (src) In: start
L10 PC: 0x8049000
2.1 3.1
(gdb) set $ebx=2
(gdb) p/s $ebx
$8 = 2
(gdb) c
Continuing.
hello, world!
Breakpoint 2, _start() at lab9-2.asm:21
(gdb) q
A debugging session is active.

Inferior 1 [process 16104] will be killed.
Quit anyway? (y or n) y
```

Рисунок 21: Завершение программы

4.1.4 Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог (рис. 22) и создаю исполняемый файл с файлом листинга и отладки (рис. 23).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm
aryusupova@aryusupova-VirtualBox:~/work/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
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$
```

Рисунок 22: Копирование файла из лабораторной №8

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
```

Рисунок 23: Создание исполняемого файла

Для загрузки в gdb программы с аргументами необходимо использовать ключ --args. Загружаю исполняемый файл в отладчик, указав аргументы (рис. 24).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ gdb --args lab9-3 аргумент1 аргумент2 'аргумент3'
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 lab9-3...
(gdb)
```

Рисунок 24: Загрузка исполняемых файлов в отладчике

Исследуя расположение аргументов командной строки в стеке после запуска программы с помощью gdb. Для начала устанавливаю точку останова перед первой инструкцией в программе и запускаю ее.

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/aryusupova/work/arch-pc/lab09/lab9-3 аргумент1 аргумент2 аргумент3

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx
(gdb) █
```

Рисунок 25: Установка точки останова и её запуск

В 32-битных системах размер указателя (адреса) составляет 4 байта. Каждый элемент массива `argv[]` - это указатель на строку. Поэтому каждый следующий аргумент смещен на 4 байта относительно предыдущего

<code>\$esp + 0:</code>	<code>argc</code>	(количество аргументов)
<code>\$esp + 4:</code>	<code>argv[0]</code>	→ указатель на имя программы
<code>\$esp + 8:</code>	<code>argv[1]</code>	→ указатель на первый аргумент
<code>\$esp + 12:</code>	<code>argv[2]</code>	→ указатель на второй аргумент
<code>\$esp + 16:</code>	<code>argv[3]</code>	→ указатель на третий аргумент
<code>\$esp + 20:</code>	<code>argv[4]</code>	→ NULL (0x0) - конец массива
<code>\$esp + 24:</code>	<code>envp[0]</code>	→ первая переменная окружения

4.2 Задание для самостоятельной работы

1. Задание

Для выполнения этого задания для начала копирую файл `lab8-1.asm` в каталог `~/work/arch-pc/lab09` (рис. 26).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-4.asm ~/work/arch-pc/lab09/lab9-4.asm
aryusupova@aryusupova-VirtualBox:~/work/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 lab9-3.asm lab9-3.lst lab9-3.o lab9-4.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ █
```

Рисунок 26: Копирование файла в рабочий каталог

Меняю программу `lab9-4.asm` из самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 27).

```
*lab9-4.asm (-/work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка

*lab9-4.asm x
#include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 10x - 4", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintLF

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

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

call _calculate_fx

add esi, eax
loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
```

Рисунок 27: Изменение программы lab9-4.asm

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg_func db "Функция: f(x) = 10x - 4", 0
```

```
msg_result db "Результат: ", 0
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg_func
```

```
call sprintLF
```

```
pop ecx
```

```

pop edx
sub ecx, 1
mov esi, 0

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

call _calculate_fx

add esi, eax
loop next

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

_calculate_fx:
mov ebx, 10
mul ebx
sub eax, 4

```

2. Задание

Для начала создаю файл lab9-5.asm для работы в этом файле (рис.28).

```

aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ touch lab9-5.asm
aryusupova@aryusupova-VirtualBox:~/work/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 lab9-3.asm lab9-3.lst lab9-3.o lab9-4.asm lab9-5.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$

```

Рисунок 28: Создание файла lab9-5.asm

Копируем код из листинга 9.3 и вставляем в файл lab9-5.asm (рис. 29). Далее делаем отладку в gdb файла lab9-5.asm (рис. 30).

```

*lab9-5.asm (~/.work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка
[Иконки]
*lab9-5.asm x
#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
|

```

Рисунок 29: Код из листинга

```

aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf -g lab9-5.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ gdb ./lab9-5
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 ./lab9-5...
(gdb)

```

Рисунок 30: Отладка в gdb

Включаю режим псевдографики для более удобного анализа программы (рис. 31).

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
0x80490e8 < start>    mov     $0x3,%ebx
0x80490ed < start+5>    mov     $0x2,%eax
0x80490f2 < start+10>   add     %eax,%ebx
0x80490f4 < start+12>   mov     $0x4,%ecx
0x80490f9 < start+17>   mul     %ecx
0x80490fb < start+19>   add     $0x5,%ebx
0x80490fe < start+22>   mov     %ebx,%edi
0x8049100 < start+24>   mov     $0x804a000,%eax
0x8049105 < start+29>   call    0x804900f <sprint>
0x804910a < start+34>   mov     %edi,%eax
0x804910c < start+36>   call    0x8049086 <iprintLF>
0x8049111 < start+41>   call    0x80490db <quit>

exec No process (asm) In:
(gdb) layout regs
```

Рисунок 31: Режим псевдографики

Для более подробного анализа программы установила брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустила её (рис. 32).

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd070 0xffffd070  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>  eflags    0x202    [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       0x0      0

B->0x80490e8 < start>    mov     $0x3,%ebx
0x80490ed < start+5>    mov     $0x2,%eax
0x80490f2 < start+10>   add     %eax,%ebx
0x80490f4 < start+12>   mov     $0x4,%ecx
0x80490f9 < start+17>   mul     %ecx
0x80490fb < start+19>   add     $0x5,%ebx
0x80490fe < start+22>   mov     %ebx,%edi
0x8049100 < start+24>   mov     $0x804a000,%eax
0x8049105 < start+29>   call    0x804900f <sprint>
0x804910a < start+34>   mov     %edi,%eax
0x804910c < start+36>   call    0x8049086 <iprintLF>
0x8049111 < start+41>   call    0x80490db <quit>

native process 18434 (asm) In: start
(gdb) layout regs
(gdb) break _start
Breakpoint 1 at 0x80490e8: file lab9-5.asm, line 8.
(gdb) run
Starting program: /home/aryusupova/work/arch-pc/lab09/lab9-5
Breakpoint 1, _start () at lab9-5.asm:8
(gdb)
```

Рисунок 32: Запуск программы и установка метки

Устанавливаем отображение регистров по шагам:

Шаг 1: Устанавливаем отображение регистров (mov ebx,3) (рис. 33)

Шаг 2: Проверка ошибки. Обошлось без них (mov eax,2) (рис. 34)

Шаг 3: Снова проверка ошибки (add ebx,ebx) (рис. 35)

Шаг 4: Ещё одна проверка ошибки (mov ecx,4)(рис. 36).

Шаг 5: Ошибка найдена (рис. 37). Команда mul ecx умножает регистр eax на ecx, но в eax осталось значение 2 из предыдущей операции, а не результат сложения (5).

Программа вычисляет:

$(2 * 4) + 5 = 13$ вместо $(3 + 2) * 4 + 5 = 25$

```
aryusupova@aryusupova-VirtualBox: ~/work/arch-pc/lab09
--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x3      3
esp      0xffffd070 0xffffd070  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490ed 0x80490ed < start+5>  eflags   0x10202  [ IF RF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       0x0      0

B+ 0x80490e8 < start>  mov $0x3,%ebx
>0x80490ed < start+5>  mov $0x2,%eax
0x80490f2 < start+10>  add %eax,%ebx
0x80490f4 < start+12>  mov $0x4,%ecx
0x80490f9 < start+17>  mul %ecx
0x80490fb < start+19>  add $0x5,%ebx
0x80490fe < start+22>  mov %ebx,%edi
0x8049100 < start+24>  mov $0x804a000,%eax
0x8049105 < start+29>  call 0x804900f <sprint>
0x804910a < start+34>  mov %edi,%eax
0x804910c < start+36>  call 0x8049086 <iprintf>
0x8049111 < start+41>  call 0x80490db <quit>

native process 18434 (asm) In: start
Breakpoint 1, _start () at lab9-5.asm:8
(gdb) stepi
(gdb) info registers eax
eax      0x0
(gdb) display /d $eax
1: /d $eax = 0
(gdb) display /d $ebx
2: /d $ebx = 3
(gdb) display /d $ecx
3: /d $ecx = 0
(gdb) display /d $edx
4: /d $edx = 0
(gdb) █
```

Рисунок 33: Шаг 1 (Установление отображения регистров)

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

--Register group: general--
eax      0x2      2      ecx      0x0      0
edx      0x0      0      ebx      0x3      3
esp      0xffffd070 0xffffd070  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490f2 0x80490f2 < start+10>  eflags   0x10202  [ IF RF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

B+ 0x80490e8 < start>      mov     $0x3,%ebx
    0x80490ed < start+5>   mov     $0x2,%eax
>0x80490f2 < start+10>    add     %eax,%ebx
0x80490f4 < start+12>     mov     $0x4,%ecx
0x80490f9 < start+17>     mul     %ecx
0x80490fb < start+19>     add     $0x5,%ebx
0x80490fe < start+22>     mov     %ebx,%edi
0x8049100 < start+24>     mov     $0x804a000,%eax
0x8049105 < start+29>     call    0x804900f <sprint>
0x804910a < start+34>     mov     %edi,%eax
0x804910c < start+36>     call    0x8049086 <iprintLF>
0x8049111 < start+41>     call    0x80490db <quit>

native process 18434 (asm) In: start
1: /d $eax = 0
(gdb) display /d $ebx
2: /d $ebx = 3
(gdb) display /d $ecx
3: /d $ecx = 0
(gdb) display /d $edx
4: /d $edx = 0
(gdb) stepi
1: /d $eax = 2
2: /d $ebx = 3
3: /d $ecx = 0
4: /d $edx = 0
(gdb) |
```

Рисунок 34: Шаг 2 проверки

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

--Register group: general--
eax      0x2      2      ecx      0x0      0
edx      0x0      0      ebx      0x5      5
esp      0xffffd070 0xffffd070  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490f4 0x80490f4 < start+12>  eflags   0x10206  [ PF IF RF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

B+ 0x80490e8 < start>      mov     $0x3,%ebx
    0x80490ed < start+5>   mov     $0x2,%eax
    0x80490f2 < start+10>  add     %eax,%ebx
>0x80490f4 < start+12>    mov     $0x4,%ecx
0x80490f9 < start+17>     mul     %ecx
0x80490fb < start+19>     add     $0x5,%ebx
0x80490fe < start+22>     mov     %ebx,%edi
0x8049100 < start+24>     mov     $0x804a000,%eax
0x8049105 < start+29>     call    0x804900f <sprint>
0x804910a < start+34>     mov     %edi,%eax
0x804910c < start+36>     call    0x8049086 <iprintLF>
0x8049111 < start+41>     call    0x80490db <quit>

native process 18434 (asm) In: start
(gdb) display /d $edx
4: /d $edx = 0
(gdb) stepi
1: /d $eax = 2
2: /d $ebx = 3
3: /d $ecx = 0
4: /d $edx = 0
(gdb) stepi
1: /d $eax = 2
2: /d $ebx = 5
3: /d $ecx = 0
4: /d $edx = 0
(gdb)
```

Рисунок 35: Шаг 3 проверки

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

--Register group: general--
eax      0x2      2      ecx      0x4      4
edx      0x0      0      ebx      0x5      5
esp      0xffffd070 0xffffd070  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490f9 0x80490f9 < start+17>  eflags   0x10206   [ PF IF RF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

B+ 0x80490e8 < start>    mov    $0x3,%ebx
0x80490ed < start+5>    mov    $0x2,%eax
0x80490f2 < start+10>   add    %eax,%ebx
0x80490f4 < start+12>   mov    $0x4,%ecx
>0x80490f9 < start+17>   mul    %ecx
0x80490fb < start+19>   add    $0x5,%ebx
0x80490fe < start+22>   mov    %ebx,%edi
0x8049100 < start+24>   mov    $0x804a000,%eax
0x8049105 < start+29>   call   0x804900f <sprint>
0x804910a < start+34>   mov    %edi,%eax
0x804910c < start+36>   call   0x8049086 <iprintLF>
0x8049111 < start+41>   call   0x80490db <quit>

native process 18434 (asm) In: start
3: /d $ecx = 0
4: /d $edx = 0
(gdb) stepi
1: /d $eax = 2
2: /d $ebx = 5
3: /d $ecx = 0
4: /d $edx = 0
(gdb) stepi
1: /d $eax = 2
2: /d $ebx = 5
3: /d $ecx = 4
4: /d $edx = 0
(gdb) █
```

Рисунок 36: Шаг 4 проверки

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

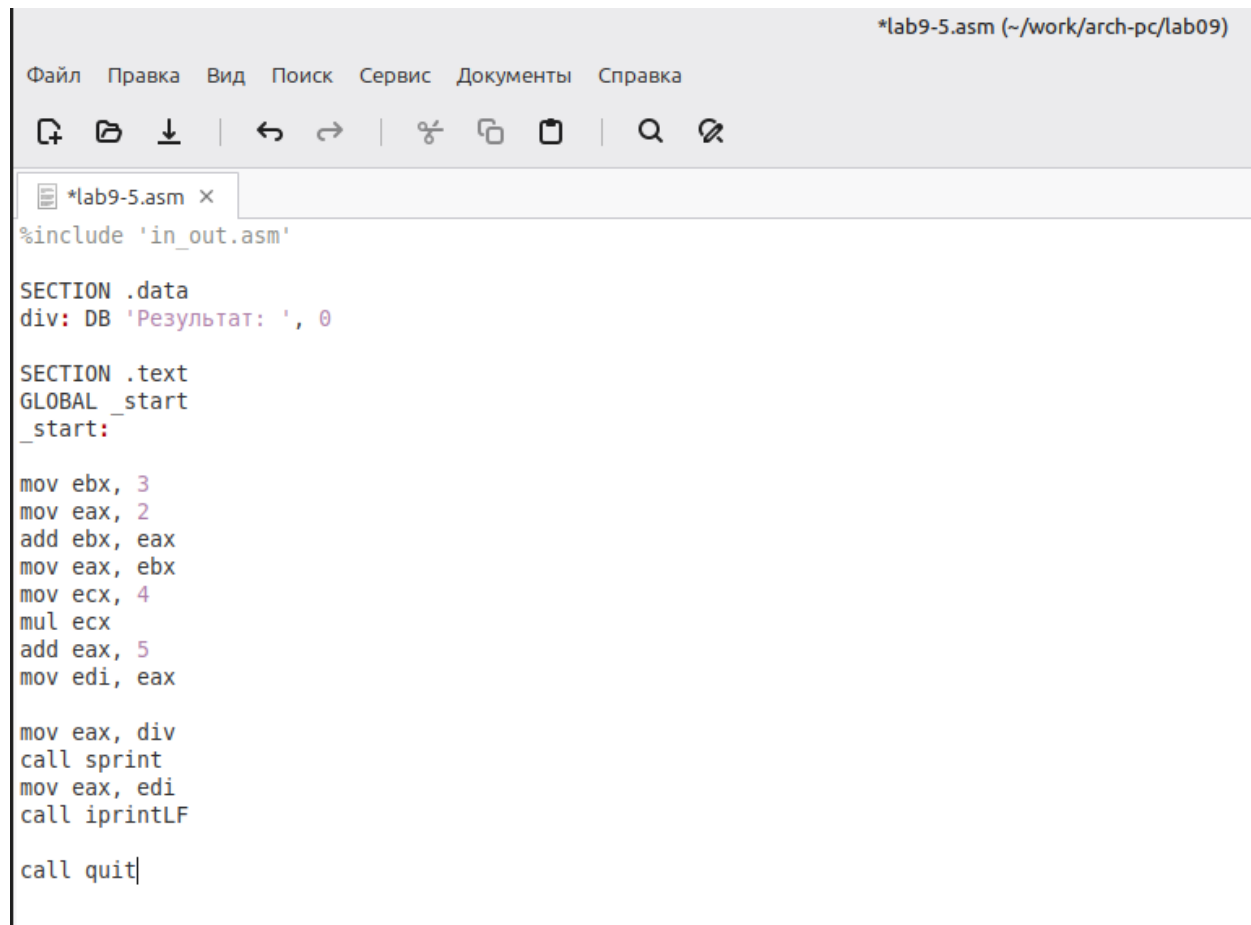
--Register group: general--
eax      0x8      8      ecx      0x4      4
edx      0x0      0      ebx      0x5      5
esp      0xffffd070 0xffffd070  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490fb 0x80490fb < start+19>  eflags   0x10202   [ IF RF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

B+ 0x80490e8 < start>    mov    $0x3,%ebx
0x80490ed < start+5>    mov    $0x2,%eax
0x80490f2 < start+10>   add    %eax,%ebx
0x80490f4 < start+12>   mov    $0x4,%ecx
0x80490f9 < start+17>   mul    %ecx
>0x80490fb < start+19>   add    $0x5,%ebx
0x80490fe < start+22>   mov    %ebx,%edi
0x8049100 < start+24>   mov    $0x804a000,%eax
0x8049105 < start+29>   call   0x804900f <sprint>
0x804910a < start+34>   mov    %edi,%eax
0x804910c < start+36>   call   0x8049086 <iprintLF>
0x8049111 < start+41>   call   0x80490db <quit>

native process 18434 (asm) In: start
3: /d $ecx = 0
4: /d $edx = 0
(gdb) stepi
1: /d $eax = 2
2: /d $ebx = 5
3: /d $ecx = 4
4: /d $edx = 0
(gdb) stepi
1: /d $eax = 8
2: /d $ebx = 5
3: /d $ecx = 4
4: /d $edx = 0
(gdb) █
```

Рисунок 37: Шаг 5 (Найденная ошибка)

Исправим ошибки кода в файле lab9-5.asm (рис. 38).



```
*lab9-5.asm (~/work/arch-pc/lab09)
Файл  Правка  Вид  Поиск  Сервис  Документы  Справка

*lab9-5.asm ×
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ', 0

SECTION .text
GLOBAL _start
_start:

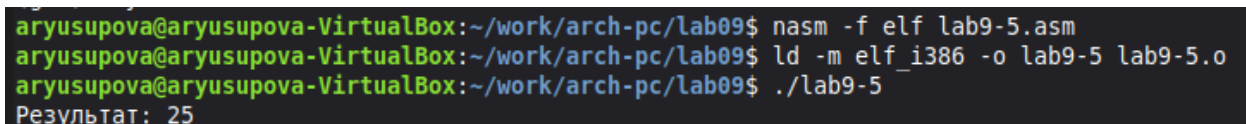
mov ebx, 3
mov eax, 2
add ebx, eax
mov eax, ebx
mov ecx, 4
mul ecx
add eax, 5
mov edi, eax

mov eax, div
call sprint
mov eax, edi
call iprintLF

call quit|
```

Рисунок 38: Изменение программы lab9-5.asm

Запускаю изменённую программу и вижу, что она работает корректно (рис. 39).



```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ nasm -f elf lab9-5.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab09$ ./lab9-5
Результат: 25
```

Рисунок 39: Запуск изменённой программы lab9-5.asm

Код измененной программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
div: DB 'Результат: ', 0
```

```
SECTION .text
```

```
GLOBAL _start
```

`_start:`

```
mov ebx, 3
mov eax, 2
add ebx, eax
mov eax, ebx
mov ecx, 4
mul ecx
add eax, 5
mov edi, eax
```

```
mov eax, div
call sprint
mov eax, edi
call iprintLF
```

```
call quit
```

5 Выводы

В результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм, а так же познакомилась с методами отладки при помощи GDB и его основными возможностями.

6 Список литературы

1. <https://esystem.rudn.ru/mod/resource/view.php?id=1030457>
2. <https://esystem.rudn.ru/mod/resource/view.php?id=1030557>
3. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).