

Лабораторная работа №9

НКАбд-03-25

Арсланов Довлетгелди

Содержание

1	Цель работы.....	1
2	Задание.....	1
3	Теоретическое введение.....	1
4	Выполнение лабораторной работы.....	2
4.1	Реализация подпрограмм в NASM.....	2
4.1.1	Отладка программ с помощью GDB.....	4
4.1.2	Добавление точек останова.....	7
4.1.3	Работа с данными программы в GDB.....	8
4.1.4	Обработка аргументов командной строки в GDB.....	10
4.2	Задание для самостоятельной работы.....	11
5	Выводы.....	15
6	Список литературы.....	15

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 Релаксация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы №9 (рис. 1).

```
root@north: ~/work/arch-pc/1 x + v
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.6.87.2-microsoft-standard-WSL2 x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/pro

System information as of Sun Dec  7 19:16:20 MSK 2025

System load:  0.08      Processes:            69
Usage of /:   0.2% of 1006.85GB Users logged in:       0
Memory usage: 5%      IPv4 address for eth0: 172.19.239.27
Swap usage:   0%

This message is shown once a day. To disable it please create the
/root/.hushlogin file.
root@north:~# mkdir ~/work/arch-pc/lab09
root@north:~# cd ~/work/arch-pc/lab09
root@north:~/work/arch-pc/lab09# touch lab9-1.asm
root@north:~/work/arch-pc/lab09#
```

Рис. 1: Создание рабочего каталога

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

```
mazurskiy@vbox:~/work/arch-pc/lab09
mazurskiy@vbox:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2x+7=27
mazurskiy@vbox:~/work/arch-pc/lab09$
```

Рис. 2: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения $f(g(x))$ (рис. 3).

```
root@north: ~/work/arch-pc/l
rectory
(mousepad:704): dconf-WARNING **: 19:40:24.680: failed to commit changes to dconf: Could not connect: No such file or di
rectory
(mousepad:704): dconf-WARNING **: 19:40:24.681: failed to commit changes to dconf: Could not connect: No such file or di
rectory
(mousepad:704): dconf-WARNING **: 19:40:24.681: failed to commit changes to dconf: Could not connect: No such file or di
rectory
(mousepad:704): dconf-WARNING **: 19:40:29.824: failed to commit changes to dconf: Could not connect: No such file or di
rectory
(mousepad:704): dconf-WARNING **: 19:40:29.824: failed to commit changes to dconf: Could not connect: No such file or di
rectory
(mousepad:704): dconf-WARNING **: 19:40:29.824: failed to commit changes to dconf: Could not connect: No such file or di
rectory
(mousepad:704): dconf-WARNING **: 19:40:29.885: failed to commit changes to dconf: Could not connect: No such file or di
rectory
root@north:~/work/arch-pc/lab09# nasm -f elf lab9-1.asm
root@north:~/work/arch-pc/lab09# ld -m elf_i386 -o lab9-1 lab-1.o
ld: cannot find lab-1.o: No such file or directory
root@north:~/work/arch-pc/lab09# ld -m elf_i386 -o lab9-1 lab9-1.o
root@north:~/work/arch-pc/lab09# ./lab9-1
Введите x: 9
2x+7=25
root@north:~/work/arch-pc/lab09#
```

Рис. 3: Изменение программы первого листинга

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

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, componую и запускаю в отладчике (рис. 4).

```
root@north: ~/work/arch-pc/l
Setting up libboost-regex1.74.0:amd64 (1.74.0-14ubuntu3) ...
Setting up libipt2 (2.0.5-1) ...
Setting up libelf1:amd64 (0.186-1ubuntu0.1) ...
Setting up libsource-highlight4v5 (3.1.9-4.1build2) ...
Setting up libdw1:amd64 (0.186-1ubuntu0.1) ...
Setting up libdebuginfod1:amd64 (0.186-1ubuntu0.1) ...
Setting up libbabeltrace1:amd64 (1.5.8-2build1) ...
Setting up gdb (12.1-0ubuntu1~22.04.2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
/sbin/ldconfig.real: /usr/lib/wsl/lib/libcud.so.1 is not a symbolic link

root@north:~/work/arch-pc/lab09# gdb lab9-2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 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) |
```

Рис. 4: Запуск программы в отладчике

Запустив программу командой run, я убедился в том, что она работает исправно (рис. 5).

```
root@north: ~/work/arch-pc/l
Setting up libdw1:amd64 (0.186-1ubuntu0.1) ...
Setting up libdebuginfod1:amd64 (0.186-1ubuntu0.1) ...
Setting up libbabeltrace1:amd64 (1.5.8-2build1) ...
Setting up gdb (12.1-0ubuntu1~22.04.2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
/sbin/ldconfig.real: /usr/lib/wsl/lib/libcud.so.1 is not a symbolic link

root@north:~/work/arch-pc/lab09# gdb lab9-2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 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: /root/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 1336) exited normally]
(gdb) |
```

Рис. 5: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпоинт на метку _start и снова запускаю отладку (рис. 6).

```
root@north: ~/work/arch-pc/l x + v
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04.2) 12.1
Copyright (C) 2022 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: /root/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 1336) exited normally]
(gdb) break_start
Undefined command: "break_start". Try "help".
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /root/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb) |
```

Рис. 6: Запуск отладчика с брейкпоинтом

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel амд топчик (рис. 7).

Различия между синтаксисом ATT и Intel заключаются в порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - 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 (ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```
root@north: ~/work/arch-pc/l
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>: mov     $0x4,%eax
    0x08049005 <+5>: mov     $0x1,%ebx
    0x0804900a <+10>: mov     $0x804a000,%ecx
    0x0804900f <+15>: mov     $0x8,%edx
    0x08049014 <+20>: int     $0x80
    0x08049016 <+22>: mov     $0x4,%eax
    0x0804901b <+27>: mov     $0x1,%ebx
    0x08049020 <+32>: mov     $0x804a008,%ecx
    0x08049025 <+37>: mov     $0x7,%edx
    0x0804902a <+42>: int     $0x80
    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. 7: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. 8).

```
root@north: ~/work/arch-pc/l
--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd420 0xfffffd420  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

0x0804903c add    BYTE PTR [eax],al
0x0804903e add    BYTE PTR [eax],al
0x08049040 add    BYTE PTR [eax],al
0x08049042 add    BYTE PTR [eax],al
0x08049044 add    BYTE PTR [eax],al
0x08049046 add    BYTE PTR [eax],al
0x08049048 add    BYTE PTR [eax],al
0x0804904a add    BYTE PTR [eax],al
0x0804904c add    BYTE PTR [eax],al
0x0804904e add    BYTE PTR [eax],al
0x08049050 add    BYTE PTR [eax],al

native process 1456 In: _start
L9      PC: 0x8049000
tracepoints -- Tracing of program execution without stopping the program.
--Type <RET> for more, q to quit, c to continue without paging--user-defined -- User-defined commands.

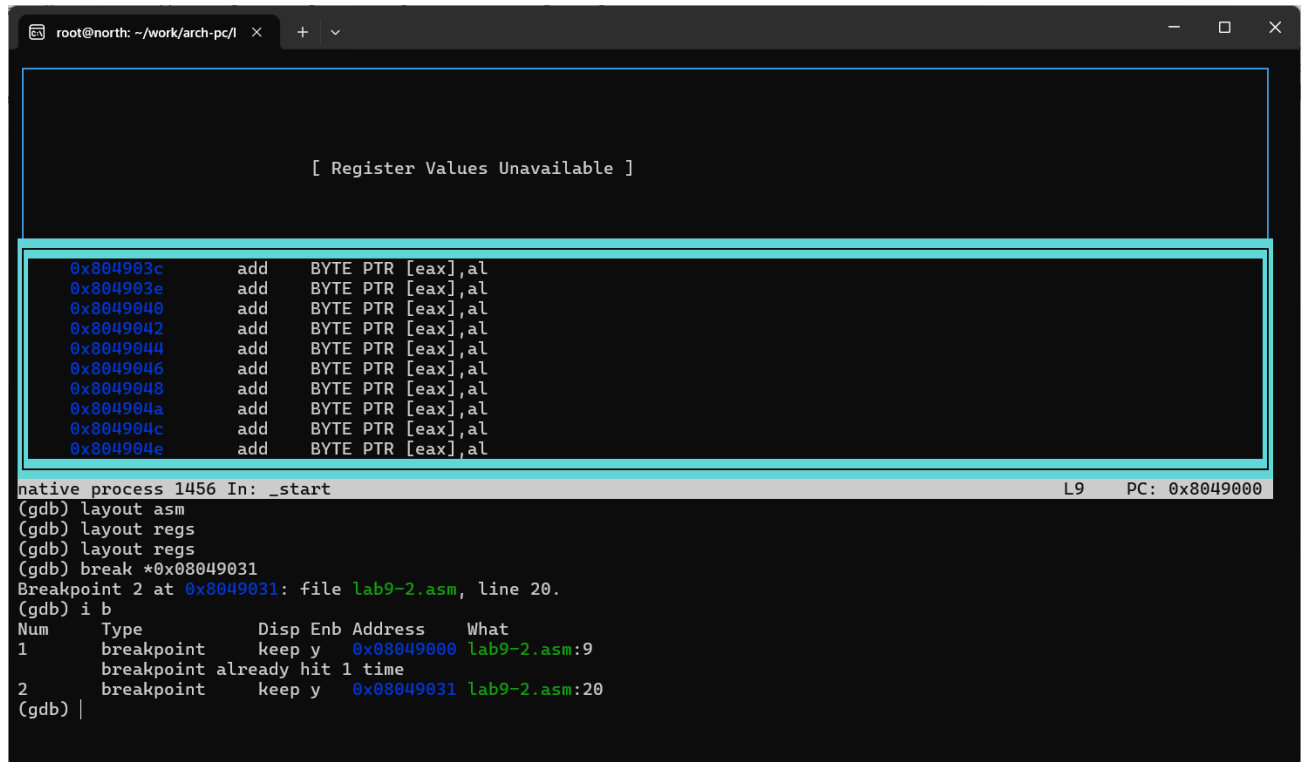
Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Type "apropos -v word" for full documentation of commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) yStarting program: /root/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
(gdb) |
```

Рис. 8: Режим псевдографики

4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 9).



The screenshot shows a GDB terminal window with the following content:

```
root@north: ~/work/arch-pc/l
```

[Register Values Unavailable]

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
0x804904c	add	BYTE PTR [eax], al
0x804904e	add	BYTE PTR [eax], al

native process 1456 In: _start L9 PC: 0x8049000

```
(gdb) layout asm
(gdb) layout regs
(gdb) layout regs
(gdb) break *0x08049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y  0x08049000 lab9-2.asm:9
          breakpoint already hit 1 time
2        breakpoint      keep y  0x08049031 lab9-2.asm:20
(gdb) |
```

Рис. 9: Список брейкпоинтов

4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой `info registers` (рис. 11).

```
root@north: ~/work/arch-pc/l  X + -
-Register group: general-
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd420  0xffffd420  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000  <_start>
cs       0x23     35      eflags   0x202     [ IF ]
ds       0x2b     43      ss       0x2b     43
fs       0x0      0      es       0x2b     43
gs       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 1486 In: _start
--Type <RET> for more, q to quit, c to continue without paging--info registersfs 0x0 0
gs 0x0 0
(gdb)
```

Рис. 11: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. 12).

```
root@north: ~/work/arch-pc/l  X + -
-Register group: general-
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd420  0xffffd420  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000  <_start>
cs       0x23     35      eflags   0x202     [ IF ]
ds       0x2b     43      ss       0x2b     43
fs       0x0      0      es       0x2b     43
gs       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

native process 1486 In: _start
--Type <RET> for more, q to quit, c to continue without paging--info registersfs 0x0 0
gs 0x0 0
(gdb) x/lsb &msg1
No symbol "msg1" in current context.
(gdb) x/lsm &msg1
0x804a000 <msg1>: "Hello, "
(gdb) /lsb 0x804a008
Undefined command: "". Try "help".
(gdb) x/lsb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)
```

Рис. 12: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. 13).

```
root@north: ~/work/arch-pc/l  X  +  -  X
Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd420  0xffffd420  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

native process 1486 In: _start L9 PC: 0x8049000
(gdb) /lsb 0x804a008
Undefined command: "". Try "help".
(gdb) x/lsb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}&msg1='h'
(gdb) x/lsb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/lsb &msg2
0x804a008 <msg2>: "xorld!\n\034"
(gdb)
```

Рис. 13: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. 14).

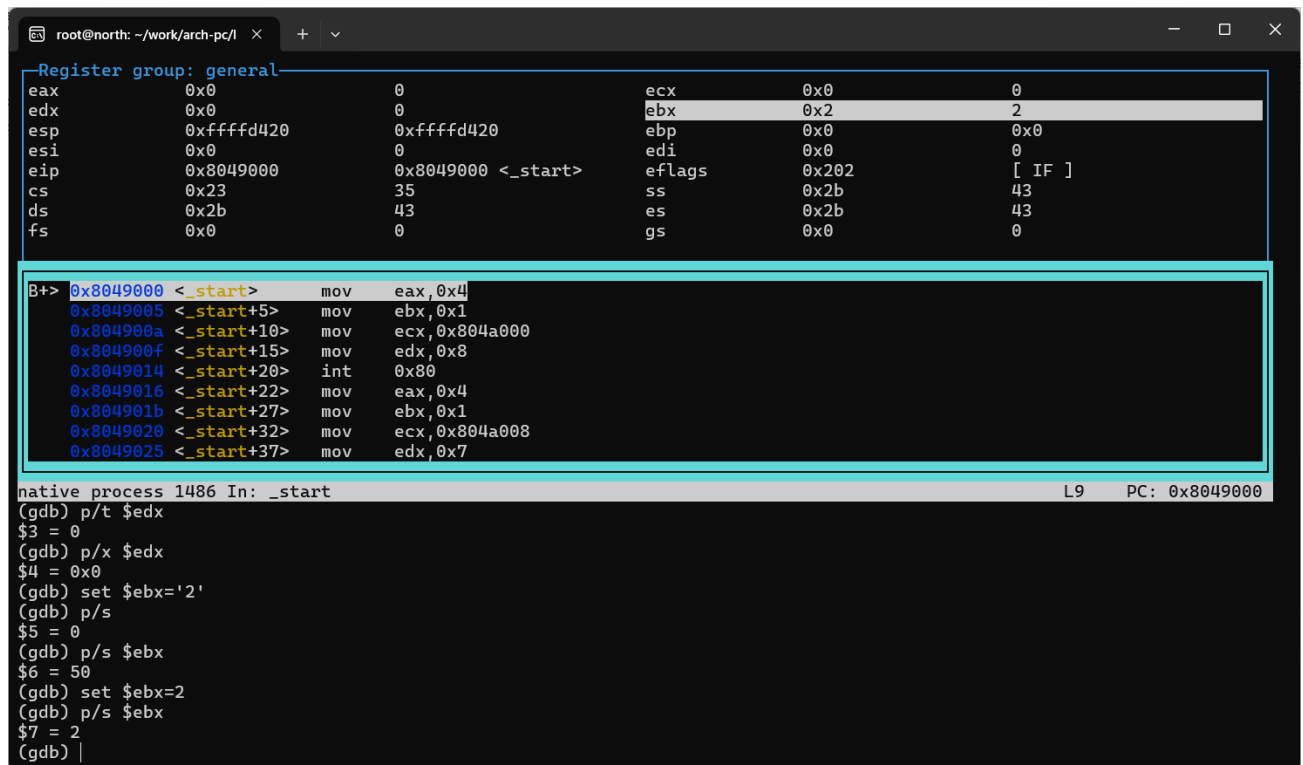
```
root@north: ~/work/arch-pc/l  X  +  -  X
Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffd420  0xffffd420  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x8049000  0x8049000 <_start>  eflags   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

native process 1486 In: _start L9 PC: 0x8049000
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/lsb &msg2
0x804a008 <msg2>: "xorld!\n\034"
(gdb) p/t $ecx
$1 = 0
(gdb) p/s $edx
$2 = 0
(gdb) p/t $edx
$3 = 0
(gdb) p/x $edx
$4 = 0x0
(gdb)
```

Рис. 14: Просмотр значения регистра разными представлениями

С помощью команды `set` меняю содержимое регистра `ebx` (рис. 15).



The screenshot shows a GDB terminal window with the following content:

```
root@north: ~/work/arch-pc/l
```

Register group: general

Register	Value	Register	Value
eax	0x0	ecx	0x0
edx	0x0	ebx	0x2
esp	0xffffd420	ebp	0x0
esi	0x0	edi	0x0
eip	0x8049000	eflags	0x202
cs	0x23	ss	0x2b
ds	0x2b	es	0x2b
fs	0x0	gs	0x0

Assembly code (disassembly):

```
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 1486 In: _start

```
(gdb) p/t $edx
$3 = 0
(gdb) p/x $edx
$4 = 0x0
(gdb) set $ebx='2'
(gdb) p/s
$5 = 0
(gdb) p/s $ebx
$6 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb) |
```

Рис. 15: Примеры использования команды `set`

4.1.4 Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог и создаю исполняемый файл с файлом листинга и отладки (рис. 16).

```
root@north: ~/work/arch-pc/l  × + ▾
(mousepad:1712): dconf-WARNING **: 21:05:25.747: failed to commit changes to dconf: Could not connect: No such file or directory
(mousepad:1712): dconf-WARNING **: 21:05:25.748: failed to commit changes to dconf: Could not connect: No such file or directory
(mousepad:1712): dconf-WARNING **: 21:05:25.748: failed to commit changes to dconf: Could not connect: No such file or directory
(mousepad:1712): dconf-WARNING **: 21:05:37.737: failed to commit changes to dconf: Could not connect: No such file or directory
(mousepad:1712): dconf-WARNING **: 21:05:37.738: failed to commit changes to dconf: Could not connect: No such file or directory
(mousepad:1712): dconf-WARNING **: 21:05:39.934: failed to commit changes to dconf: Could not connect: No such file or directory
(mousepad:1712): dconf-WARNING **: 21:05:47.094: failed to commit changes to dconf: Could not connect: No such file or directory
(mousepad:1712): dconf-WARNING **: 21:05:47.094: failed to commit changes to dconf: Could not connect: No such file or directory
(mousepad:1712): dconf-WARNING **: 21:05:47.143: failed to commit changes to dconf: Could not connect: No such file or directory
root@north:~/work/arch-pc/lab09# nasm -f elf -g -l lab9-3.lst lab9-3.asm
root@north:~/work/arch-pc/lab09# ld -m elf_i386 -o lab9-3 lab9-3.o
root@north:~/work/arch-pc/lab09#
```

Рис. 16: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпоинт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра `esp` на `+4`, число обусловлено разрядностью системы, а указатель `void` занимает как раз 4 байта, ошибка при аргументе `+24` означает, что аргументы на вход программы закончились. (рис. 17).

```
root@north: ~/work/arch-pc/ x + v
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) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /root/work/arch-pc/lab09/lab9-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx
(gdb) x/s *(void**)(esp + 4)
0xffffd558: "/root/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd578: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd582: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd593: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd595: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 17: Проверка работы стека

4.2 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 18).

```
~/work/arch-pc/lab09/lab9-4.asm - Mousepad
File Edit Search View Document Help
1 #include "fn_out.asm"
2
3 SECTION .data
4 msg_func db "Функция: f(x) = 10x - 4", 0
5 msg_result db "Результат: ", 0
6
7 SECTION .text
8 GLOBAL _start
9
10 _start:
11 mov eax, msg_func
12 call sprintf
13
14 pop ecx
15 pop edx
16 sub ecx, 1
17 mov esi, 0
18
19 next:
20 cmp ecx, 0h
21 jz _end
22 pop eax
23 call atoi
24
25 call _calculate_fx
26
27 add esi, eax
28 loop next
29
30 _end:
31 mov eax, msg_result
32 call sprintf
33 mov eax, esi
34 call sprintf
35 call quit
36
37 _calculate_fx:
38 mov ebx, 10
39 mul ebx
40 sub eax, 4
41 ret
```

Рис. 18: Измененная программа предыдущей лабораторной работы

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:
```

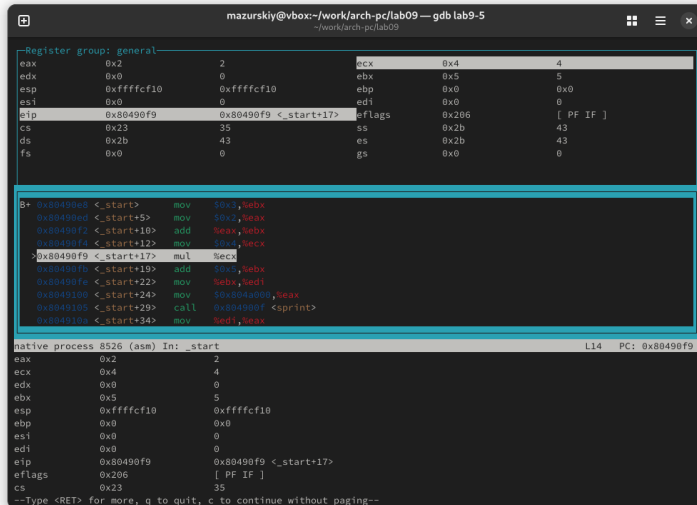
```
mov ebx, 10
```

```
mul ebx
```

```
sub eax, 4
```

2. **Запускаю программу в режиме отладчика и пошагово через si просматриваю изменение значений регистров через i r. При**

выполнении инструкции `mul ecx` можно заметить, что результат умножения записывается в регистр `eax`, но также меняет и `edx`. Значение регистра `ebx` не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию (рис. 19).



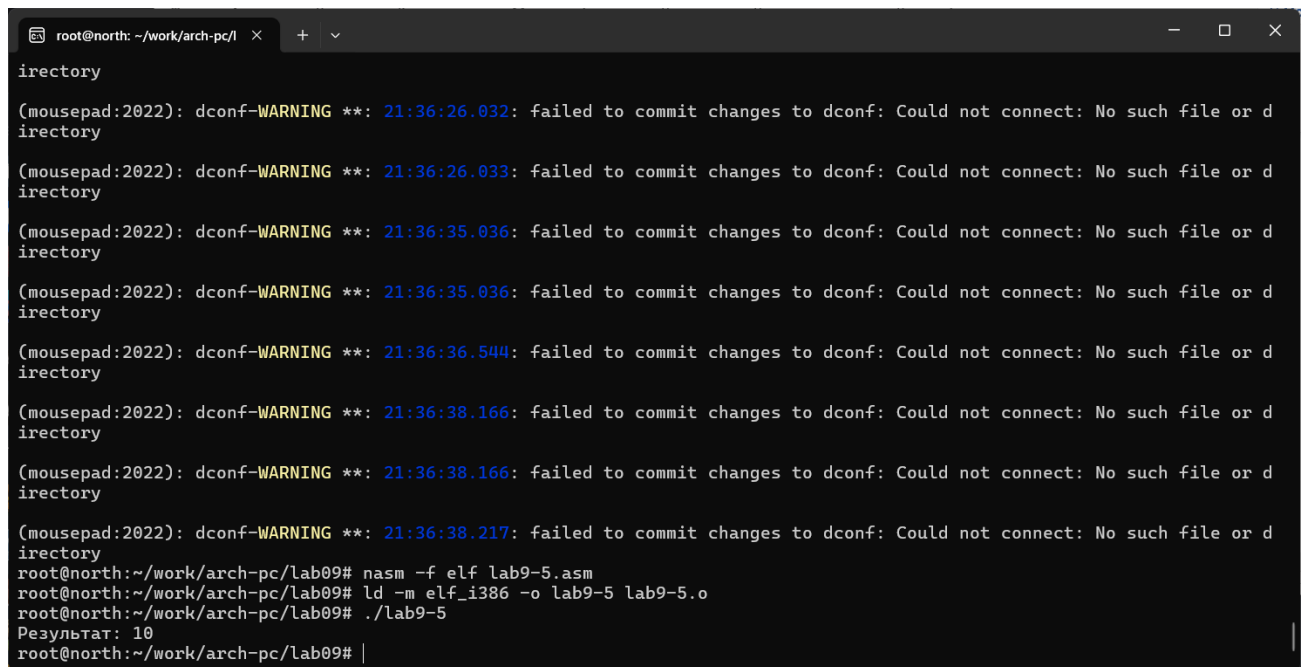
```
Register group: general
eax 0x2 2 ecx 0x4 4
edx 0x0 0 ebx 0x2 5
esp 0xffffcf10 0xffffcf10 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x80490f9 0x80490f9 <.start+17> eflags 0x206 [ PF IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

0x80490f9 <.start> mov 10i,ebx
0x80490fd <.start+5> mov 10i,ecx
0x8049102 <.start+10> add 5eax,ebx
0x8049104 <.start+12> mov 10i,ecx
0x8049105 <.start+17> mul ecx
0x8049106 <.start+19> add 5ebx,ebx
0x804910e <.start+22> mov 5ebx,edx
0x8049100 <.start+24> mov 5ebx,edx
0x8049105 <.start+29> call 0x804900f <sprint>
0x8049105 <.start+34> mov 5edx,ecx

native process 8526 (asm) In: .start L14 PC: 0x80490f9
eax 0x2 2
ecx 0x4 4
edx 0x0 0
ebx 0x5 5
esp 0xffffcf10 0xffffcf10
ebp 0x0 0x0
esi 0x0 0
edi 0x0 0
eip 0x80490f9 0x80490f9 <.start+17>
eflags 0x206 [ PF IF ]
cs 0x23 35
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 19: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. 20).



```
root@north: ~/work/arch-pc/l
iirectory

(mousepad:2022): dconf-WARNING **: 21:36:26.032: failed to commit changes to dconf: Could not connect: No such file or d
iirectory

(mousepad:2022): dconf-WARNING **: 21:36:26.033: failed to commit changes to dconf: Could not connect: No such file or d
iirectory

(mousepad:2022): dconf-WARNING **: 21:36:35.036: failed to commit changes to dconf: Could not connect: No such file or d
iirectory

(mousepad:2022): dconf-WARNING **: 21:36:35.036: failed to commit changes to dconf: Could not connect: No such file or d
iirectory

(mousepad:2022): dconf-WARNING **: 21:36:36.544: failed to commit changes to dconf: Could not connect: No such file or d
iirectory

(mousepad:2022): dconf-WARNING **: 21:36:38.166: failed to commit changes to dconf: Could not connect: No such file or d
iirectory

(mousepad:2022): dconf-WARNING **: 21:36:38.166: failed to commit changes to dconf: Could not connect: No such file or d
iirectory

(mousepad:2022): dconf-WARNING **: 21:36:38.217: failed to commit changes to dconf: Could not connect: No such file or d
iirectory
root@north:~/work/arch-pc/lab09# nasm -f elf lab9-5.asm
root@north:~/work/arch-pc/lab09# ld -m elf_i386 -o lab9-5 lab9-5.o
root@north:~/work/arch-pc/lab09# ./lab9-5
Результат: 10
root@north:~/work/arch-pc/lab09#
```

Рис. 20: Проверка корректировок в программе

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 приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при помощи GDB и его основными возможностями.

6 Список литературы

1. **Курс на ТУИС**
2. **Лабораторная работа №9**
3. **Программирование на языке ассемблера NASM Столяров А. В.**