

Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера.

Наговицын Арсений Владимирович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация подпрограмм в NASM.	7
3.2	Отладка программ с помощью GDB.	10
3.3	Добавление точек останова	14
3.4	Работа с данными программы в GDB.	14
3.5	Обработка аргументов командной строки в GDB.	17
3.6	Выполнение заданий для самостоятельной работы.	18

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Редактирование файла	7
3.3	Компиляция и запуск файла	8
3.4	Редактирование файла	8
3.5	Компиляция и запуск файла	10
3.6	Создание файла	10
3.7	Редактирование файла	11
3.8	Компиляция и запуск файла	11
3.9	Запуск программы в отладчике	11
3.10	Установка breakpoint	12
3.11	Дисассимилированный код	12
3.12	Дисассимилированный код	13
3.13	Графика	13
3.14	Установленные точки останова	14
3.15	Команды stepi	15
3.16	Значение переменной msg1	16
3.17	Команда set	16
3.18	Значение регистра edx	17
3.19	Команда continue и quit	17
3.20	Компиляция и запуск файла	18
3.21	Установление точки останова	18
3.22	Просмотр значений	18
3.23	Создание файла	18
3.24	Компиляция и запуск файла	20
3.25	Создание файла	20
3.26	Редактирование файла	21
3.27	Компиляция файла	21
3.28	Программа в оболочке GDB	21
3.29	Программа в оболочке GDB	22
3.30	Команда layout	22
3.31	Команды stepi	23
3.32	Редактирование файла	23

Список таблиц

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. Добавление точек останова.
4. Работа с данными программы в GDB.
5. Обработка аргументов командной строки в GDB.
6. Выполнение заданий для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Реализация подпрограмм в NASM.

Создаю каталог для программ и перехожу в него. Создаю файл (рис. 3.1).

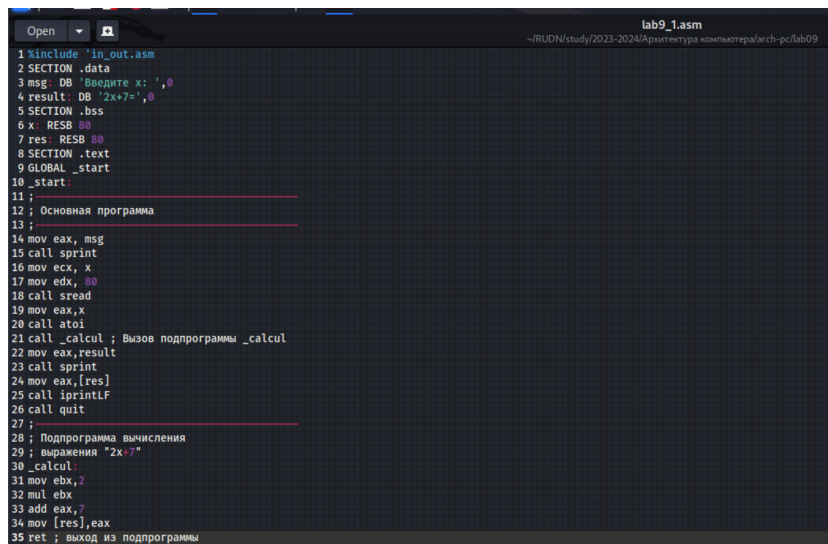
```
(avnagovicihn@avnagovicihn)-[~/../study/2023-2024/Архитектура компьютера/arch-pc]
$ mkdir lab09

(avnagovicihn@avnagovicihn)-[~/../study/2023-2024/Архитектура компьютера/arch-pc]
$ cd lab09

(avnagovicihn@avnagovicihn)-[~/../2023-2024/Архитектура компьютера/arch-pc/lab09]
$ touch lab9_1.asm
```

Рис. 3.1: Создание каталога и файла

Ввожу в файл текст программы из листинга 9.1(рис. 3.2).



```
lab9_1.asm
~(RUDN\study\2023-2024\Архитектура компьютера\arch-pc\lab09

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 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ;
12 ; Основная программа
13 ;
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax, x
20 call atoi
21 call _calcul ; Вызов подпрограммы _calcul
22 mov eax, result
23 call sprint
24 mov eax, [res]
25 call iprintf
26 call quit
27 ;
28 ; Подпрограмма вычисления
29 ; выражения "2x+7"
30 _calcul:
31 mov ebx, 2
32 mul ebx
33 add eax, 7
34 mov [res], eax
35 ret ; выход из подпрограммы
```

Рис. 3.2: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 3.3).

```
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ nasm -f elf lab9_1.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ld -m elf_i386 -o lab9_1 lab9_1.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ./lab9_1
Введите x: 2
2x+7=11

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ./lab9_1
Введите x: 5
2x+7=17

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ./lab9_1
Введите x: 0
2x+7=7
```

Рис. 3.3: Компиляция и запуск файла

Изменяю текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul` (рис. 3.4).

```
Open lab9_1.asm
~/BLUDN/study/2023-2024/Архитектура компьютера/arch-pc/lab09

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 res: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ;
12 ; Основная программа
13 ;
14 mov eax, msg
15 call sprint
16 mov ecx, x
17 mov edx, 80
18 call stread
19 mov eax, x
20 call atoi
21 call _subcalcul
22 call _calcul ; Вызов подпрограммы _calcul
23 mov eax, result
24 call sprint
25 mov eax, [res]
26 call sprintlf
27 call quit
28 ;
```

Рис. 3.4: Редактирование файла

Изменённая программа

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0

SECTION .bss
```



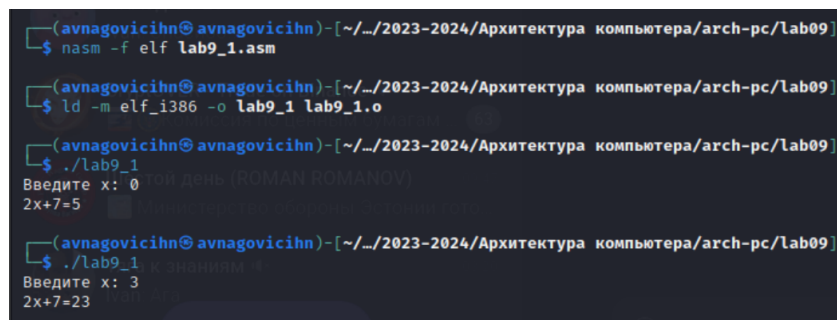
```

x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x
call atoi
call _subcalcul
call _calcul ; Вызов подпрограммы _calcul
mov eax,result
call sprint
mov eax,[res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_subcalcul:
mov ebx,3
mul ebx
add eax,-1

```

```
ret
_calcul:
mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret ; выход из подпрограммы
```

Создаю исполняемый файл и запускаю его (рис. 3.5).

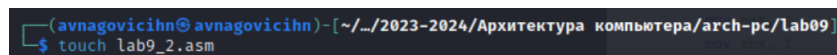


```
(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ nasm -f elf lab9_1.asm
(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ld -m elf_i386 -o lab9_1 lab9_1.o
(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ./lab9_1
Введите x: 0
2x+7=7
(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ./lab9_1
Введите x: 3
2x+7=23
```

Рис. 3.5: Компиляция и запуск файла

3.2 Отладка программ с помощью GDB.

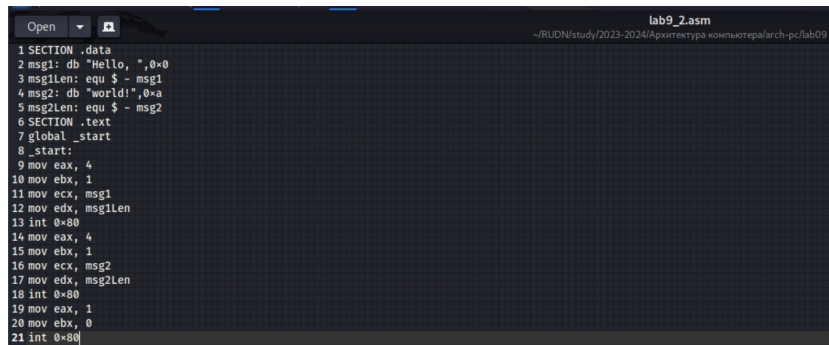
Создаю файл lab09-2.asm (рис. 3.6).



```
(avnagovicihn@ avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ touch lab9_2.asm
```

Рис. 3.6: Создание файла

Ввожу в файл текст программы из листинга 9.1(рис. 3.7).

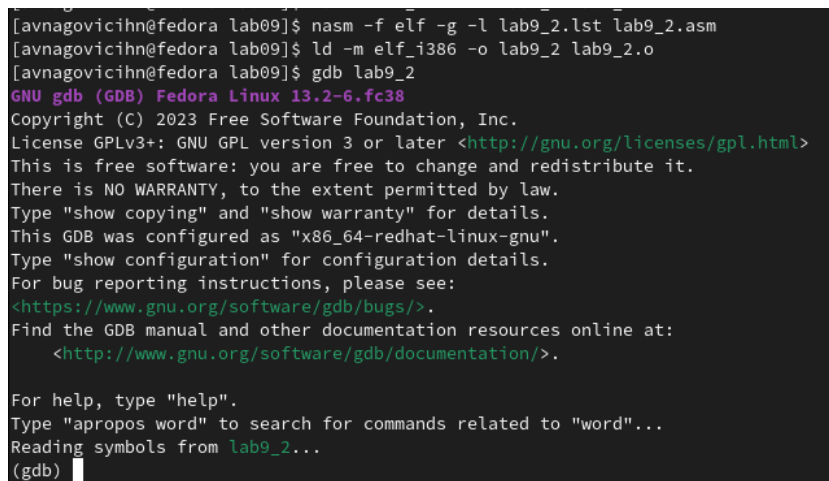


```
lab9_2.asm
~RUDN/study/2023-2024/Архитектура компьютера/arch-pc/lab09

1 SECTION .data
2 msg1: db "Hello, ",0x0
3 msg1len: equ $ - msg1
4 msg2: db "world!",0xa
5 msg2len: equ $ - msg2
6 SECTION .text
7 global _start
8 _start:
9 mov eax, 4
10 mov ebx, 1
11 mov ecx, msg1
12 mov edx, msg1len
13 int 0x80
14 mov eax, 4
15 mov ebx, 1
16 mov ecx, msg2
17 mov edx, msg2len
18 int 0x80
19 mov eax, 1
20 mov ebx, 0
21 int 0x80
```

Рис. 3.7: Редактирование файла

Создаю исполняемый файл и загружаю его в отладчик (рис. 3.8).

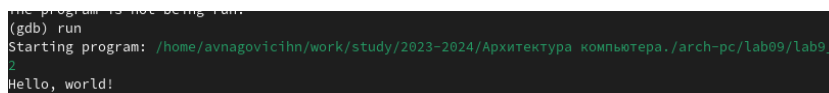


```
[avmagovicihn@fedora lab09]$ nasm -f elf -g -l lab9_2.lst lab9_2.asm
[avmagovicihn@fedora lab09]$ ld -m elf_i386 -o lab9_2 lab9_2.o
[avmagovicihn@fedora lab09]$ gdb lab9_2
GNU gdb (GDB) Fedora Linux 13.2-6.fc38
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9_2...
(gdb)
```

Рис. 3.8: Компиляция и запуск файла

Проверяю работу программы, запустив ее в оболочке GDB с помощью команды run (рис. 3.9).



```
(gdb) run
Starting program: /home/avmagovicihn/work/study/2023-2024/Архитектура компьютера./arch-pc/lab09/lab9_2
Hello, world!
```

Рис. 3.9: Запуск программы в отладчике

Ставлю точку останова на метку _start и запустил её. (рис. 3.10).

```

(gdb) break _start
Breakpoint 1 at 0x201140: file lab9_2.asm, line 9.
(gdb) run
Starting program: /home/avnagovic1hn/work/study/2023-2024/Архитектура компьютера./arch-pc/lab09/lab9_2
Breakpoint 1, _start () at lab9_2.asm:9
9      mov eax, 4
(gdb)

```

Рис. 3.10: Установка breakpoint

Просматриваю дисассимилированный код (рис. 3.11).

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x00201140 <+0>:      mov     $0x4,%eax
      0x00201145 <+5>:      mov     $0x1,%ebx
      0x0020114a <+10>:     mov     $0x203188,%ecx
      0x0020114f <+15>:     mov     $0x8,%edx
      0x00201154 <+20>:     int     $0x80
      0x00201156 <+22>:     mov     $0x4,%eax
      0x0020115b <+27>:     mov     $0x1,%ebx
      0x00201160 <+32>:     mov     $0x203190,%ecx
      0x00201165 <+37>:     mov     $0x7,%edx
      0x0020116a <+42>:     int     $0x80
      0x0020116c <+44>:     mov     $0x1,%eax
      0x00201171 <+49>:     mov     $0x0,%ebx
      0x00201176 <+54>:     int     $0x80

```

Рис. 3.11: Дисассимилированный код

Просматриваю дисассимилированный код с Intel'овским синтаксисом (рис. 3.12).

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x00201140 <+0>:      mov     eax,0x4
    0x00201145 <+5>:      mov     ebx,0x1
    0x0020114a <+10>:     mov     ecx,0x203188
    0x0020114f <+15>:     mov     edx,0x8
    0x00201154 <+20>:     int     0x80
    0x00201156 <+22>:     mov     eax,0x4
    0x0020115b <+27>:     mov     ebx,0x1
    0x00201160 <+32>:     mov     ecx,0x203190
    0x00201165 <+37>:     mov     edx,0x7
    0x0020116a <+42>:     int     0x80
    0x0020116c <+44>:     mov     eax,0x1
    0x00201171 <+49>:     mov     ebx,0x0
    0x00201176 <+54>:     int     0x80

```

Рис. 3.12: Дисассимилированный код

Включая режим графики (рис. 3.13).

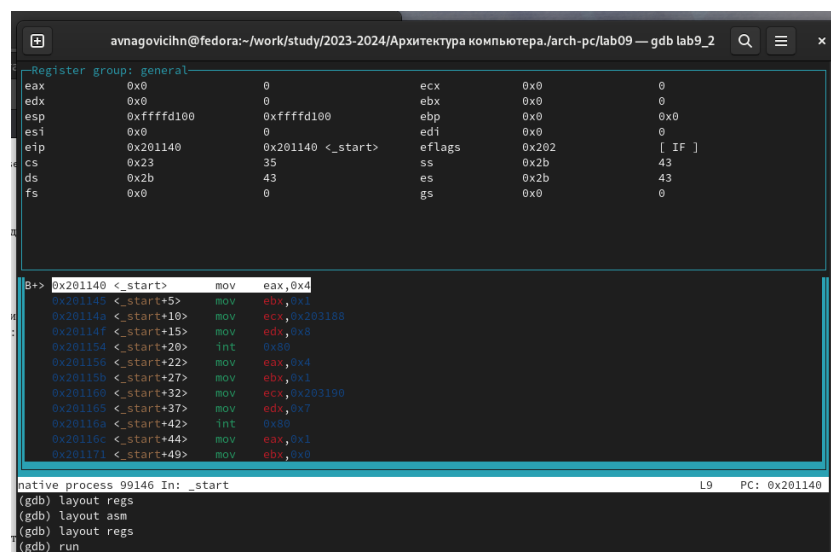
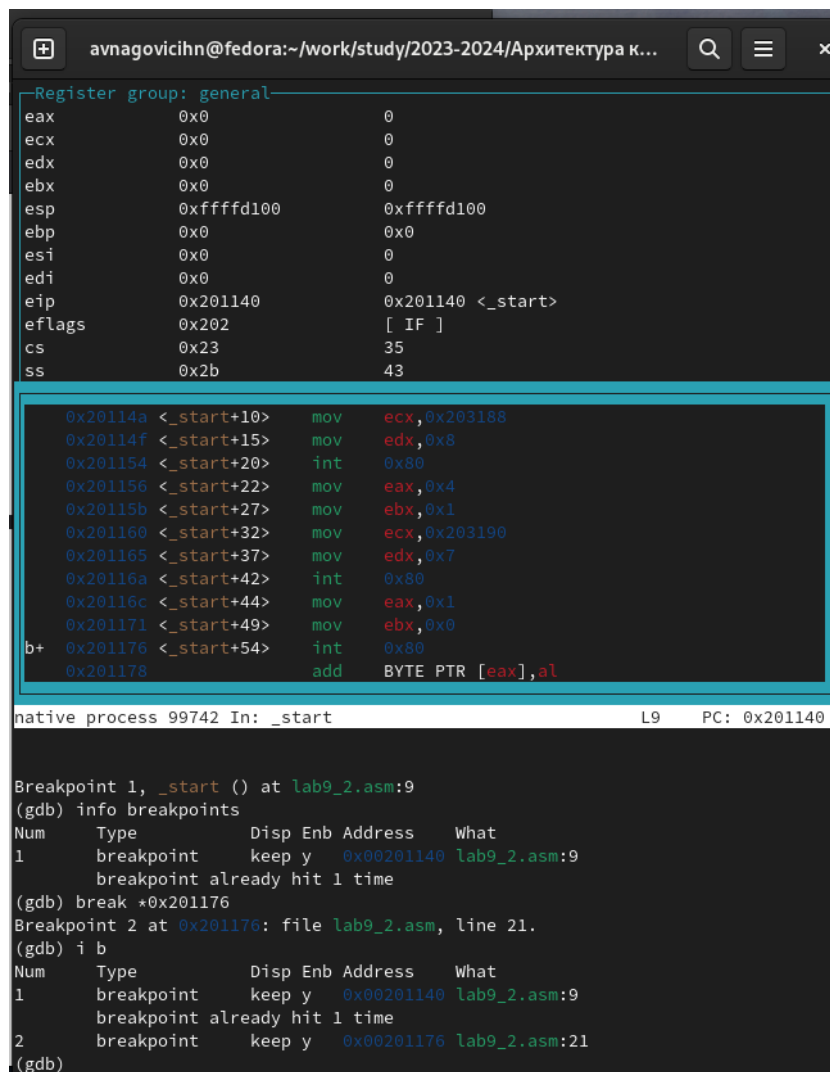


Рис. 3.13: Графика

3.3 Добавление точек останова

Проверяю установленные точки останова, добавляю еще одну точку останова((рис. 3.14).



The screenshot shows the GDB interface with the following content:

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd100 0xffffd100
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x201140 0x201140 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
```

```
0x20114a <_start+10> mov    ecx,0x203188
0x20114f <_start+15> mov    edx,0x8
0x201154 <_start+20> int    0x80
0x201156 <_start+22> mov    eax,0x4
0x20115b <_start+27> mov    ebx,0x1
0x201160 <_start+32> mov    ecx,0x203190
0x201165 <_start+37> mov    edx,0x7
0x20116a <_start+42> int    0x80
0x20116c <_start+44> mov    eax,0x1
0x201171 <_start+49> mov    ebx,0x0
b+ 0x201176 <_start+54> int    0x80
0x201178 add    BYTE PTR [eax],al
```

native process 99742 In: _start L9 PC: 0x201140

```
Breakpoint 1, _start () at lab9_2.asm:9
(gdb) info breakpoints
Num   Type             Disp Enb Address      What
1     breakpoint       keep y  0x00201140 lab9_2.asm:9
      breakpoint already hit 1 time
(gdb) break *0x201176
Breakpoint 2 at 0x201176: file lab9_2.asm, line 21.
(gdb) i b
Num   Type             Disp Enb Address      What
1     breakpoint       keep y  0x00201140 lab9_2.asm:9
      breakpoint already hit 1 time
2     breakpoint       keep y  0x00201176 lab9_2.asm:21
(gdb)
```

Рис. 3.14: Установленные точки останова

3.4 Работа с данными программы в GDB.

Выполнил 5 инструкций с помощью команды stepi (рис. 3.15).

```
avnagovicihn@fedora:~/work/study/2023-2024/Архитектура к...
Register group: general
eax      0x8      8
ecx      0x203188 2109832
edx      0x8      8
ebx      0x1      1
esp      0xffffd100 0xffffd100
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x201156 0x201156 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43

0x20114a <_start+10> mov ecx,0x203188
0x20114f <_start+15> mov edx,0x8
0x201154 <_start+20> int 0x80
> 0x201156 <_start+22> mov eax,0x4
0x20115b <_start+27> mov ebx,0x1
0x201160 <_start+32> mov ecx,0x203190
0x201165 <_start+37> mov edx,0x7
0x20116a <_start+42> int 0x80
0x20116c <_start+44> mov eax,0x1
0x201171 <_start+49> mov ebx,0x0
b+ 0x201176 <_start+54> int 0x80
0x201178 add BYTE PTR [eax],al

native process 99742 In: _start L14 PC: 0x201156

Breakpoint 1, _start () at lab9_2.asm:9
(gdb) info breakpoints
Num    Type             Disp Enb Address      What
1      breakpoint      keep y   0x00201140 lab9_2.asm:9
      breakpoint already hit 1 time
(gdb) break *0x201176
Breakpoint 2 at 0x201176: file lab9_2.asm, line 21.
(gdb) i b
Num    Type             Disp Enb Address      What
1      breakpoint      keep y   0x00201140 lab9_2.asm:9
      breakpoint already hit 1 time
2      breakpoint      keep y   0x00201176 lab9_2.asm:21
(gdb) si 5
```

Рис. 3.15: Команды stepi

Изменились значения регистров eax, ecx, edx и ebx.

Посматриваю значение переменной msg1 по имени и msg2 по адресу (рис. 3.16).

The screenshot shows a debugger window with the title bar 'avnagovichn@fedora:~/work/study/2023-2024/Архитектура компьютера./...'. The main pane is divided into two sections. The top section, titled 'Register group: general', lists the following registers and their values:

Register	Value (Hex)	Value (Dec)
eax	0x4	4
ecx	0x203188	2109832
edx	0x8	8
ebx	0x1	1
esp	0xffffd100	0xffffd100
ebp	0x0	0x0
esi	0x0	0
edi	0x0	0
eip	0x20115b	0x20115b <_start+27>
eflags	0x202	[IF]
cs	0x23	35
ss	0x2b	43
ds	0x2b	43

The bottom section displays assembly code with addresses from 0x201282 to 0x20129d. The code includes instructions like 'jae', 'popa', 'jb', 'add', 'outs', 'fs: add', 'je', 'js', 'fs: popa', and 'gs: js'. A status bar at the bottom indicates 'native process 99742 In: _start L15 PC: 0x20115b'. Below the assembly code, the following memory dump is shown:

```
0x203190: "lab9_2.asm"
(gdb) x/1sb & msg1
0x203188 <msg1>: "Hello, "
(gdb) x/1sb 0x203190
0x203190 <msg2>: "world!\n"
```

Рис. 3.16: Значение переменной msg1

Изменяю первый символ переменных msg1 и msg2 (рис. 3.17).

The screenshot shows a debugger window with the following commands and output:

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x203188 <msg1>: "hello, "
(gdb) set {char}&msg2='c'
(gdb) x/1sb &msg2
0x203190 <msg2>: "corld!\n"
```

Рис. 3.17: Команда set

Вывожу в различных форматах значение регистра edx (рис. 3.18).


```
(gdb) p/s $edx
$1 = 8
(gdb) p/t $edx
$2 = 1000
(gdb) p/x $edx
$3 = 0x8
```

Рис. 3.18: Значение регистра edx

Завершаю выполнение программы с помощью команды continue и quit (рис. 3.19).

```
(gdb) c
Continuing.
corld!

Breakpoint 2, _start () at lab9_2.asm:21
(gdb) q
A debugging session is active.

        Inferior 1 [process 99742] will be killed.

Quit anyway? (y or n) █
```

Рис. 3.19: Команда continue и quit

3.5 Обработка аргументов командной строки в GDB.

Скопировав файл и переименовав его. Создаю исполняемый файл (рис. 3.20).

```

lab9_3.o: error: cannot open include file 'in_001.asm': no such file
[avnagovicihn@fedora lab09]$ nasm -f elf -g -l lab9_3.lst lab9_3.asm
[avnagovicihn@fedora lab09]$ ld -m elf_i386 -o lab9_3 lab9_3.o

```

Рис. 3.20: Компиляция и запуск файла

Устанавливаю точку останова и запускаю файл (рис. 3.21).

```

(gdb) b _start
Breakpoint 1 at 0x201228: file lab9_3.asm, line 5.
(gdb) run
Starting program: /home/avnagovicihn/work/study/2023-2024/Архитектура компьютера./arch-pc/lab
09/lab9_3 аргумент1 аргумент 2 аргумент\ 3
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>

```

Рис. 3.21: Установление точки останова

Просматриваю вершину стека по их адресам (рис. 3.22).

```

(gdb) x/x $esp
0xffffd0b0: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd268: "/home/avnagovicihn/work/study/2023-2024/Архитектура компьютера./arch-pc/lab
09/lab9_3"
(gdb) x/s *(void**)(esp + 8)
0xffffd2d2: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd2e4: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd2f5: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd2f7: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>

```

Рис. 3.22: Просмотр значений

Шаг изменения адреса равен 4, так как количество аргументов командной строки равно 4.

3.6 Выполнение заданий для самостоятельной работы.

1. Создаю файл (рис. 3.23).

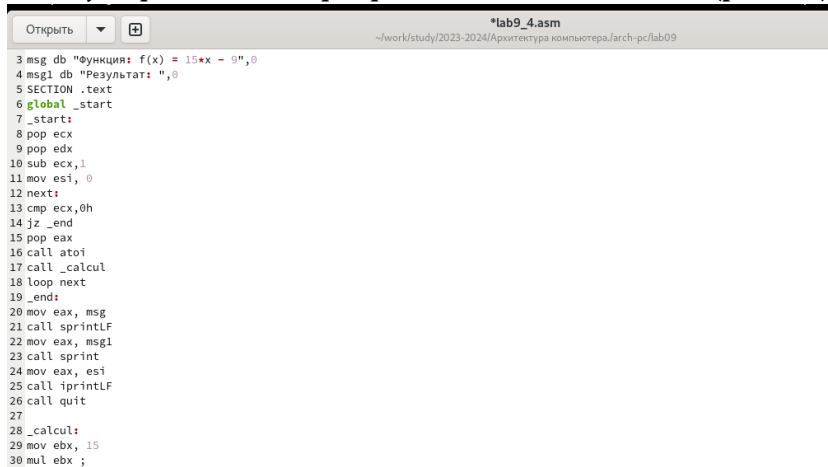
```

[avnagovicihn@fedora lab09]$ touch lab9_4.asm

```

Рис. 3.23: Создание файла

Ввожу в файл текст программы из листинга 9.4(рис. ??).



```
*lab9_4.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09

3 msg db "Функция: f(x) = 15*x - 9",0
4 msg1 db "Результат: ",0
5 SECTION .text
6 global _start
7 _start:
8 pop ecx
9 pop edx
10 sub ecx,1
11 mov esi, 0
12 next:
13 cmp ecx,0h
14 jz _end
15 pop eax
16 call atoi
17 call _calcul
18 loop next
19 _end:
20 mov eax, msg
21 call sprintf
22 mov eax, msg1
23 call sprintf
24 mov eax, esi
25 call sprintf
26 call quit
27
28 _calcul:
29 mov ebx, 15
30 mul ebx ;
```

Листинг 9.4:

```
%include 'in_out.asm'

SECTION .data
msg db "Функция: f(x) = 15*x - 9",0
msg1 db "Результат: ",0

SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0
next:
cmp ecx,0h
jz _end
pop eax
call atoi
call _calcul
loop next
_end:
mov eax, msg
```

```

call sprintf
mov eax, msg1
call sprintf
mov eax, esi
call iprintLF
call quit

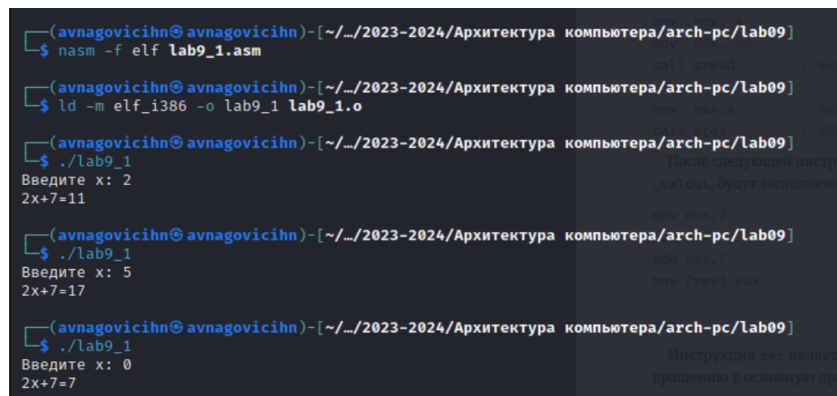
```

```

_calcul:
mov ebx, 15
mul ebx ;
add eax, -9 ;
add esi, eax
ret

```

Создаю исполняемый файл и запускаю его (рис. 3.24).



```

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ nasm -f elf lab9_1.asm

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ld -m elf_i386 -o lab9_1 lab9_1.o

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ./lab9_1
Введите x: 2
2x+7=11

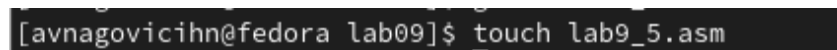
(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ./lab9_1
Введите x: 5
2x+7=17

(avnagovicihn@avnagovicihn)-[~/2023-2024/Архитектура компьютера/arch-pc/lab09]
$ ./lab9_1
Введите x: 0
2x+7=7

```

Рис. 3.24: Компиляция и запуск файла

2. Создаю файл lab9_5.asm (рис. 3.25).



```

[avnagovicihn@fedora lab09]$ touch lab9_5.asm

```

Рис. 3.25: Создание файла

Ввожу в файл текст программы из листинга 9.3(рис. 3.26).



```
lab9_5.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09

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
```

Рис. 3.26: Редактирование файла

Создаю исполняемый файл (рис. 3.27).

```
[avnagovicihn@fedora lab09]$ nasm -f elf -g -l lab9_5.lst lab9_5.asm
[avnagovicihn@fedora lab09]$ ld -m elf_i386 -o lab9_5 lab9_5.o
```

Рис. 3.27: Компиляция файла

Загружаю и запускаю файл программы в оболочке GDB (рис. 3.28).

```
[avnagovicihn@fedora lab09]$ gdb lab9_5
GNU gdb (GDB) Fedora Linux 13.2-6.fc38
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9_5...
```

Рис. 3.28: Программа в оболочке GDB

Ставлю точку Останова на метку _start (рис. 3.29).

```
(gdb) break _start
Breakpoint 1 at 0x201228: file lab9_5.asm, line 8.
(gdb) set disassembly-flavor intel
```

Рис. 3.29: Программа в оболочке GDB

Включаю режим графики (рис. 3.30).

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd100 0xffffd100
ebp      0x0      0x0

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

native process 108213 In: _start L8 PC: 0x201228
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) yDebuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

Breakpoint 1, _start () at lab9_5.asm:8
(gdb)
```

Рис. 3.30: Команда layout

Ввожу команду stepi 5 раз, замечаю что умножается не тот регистр (рис. 3.31).

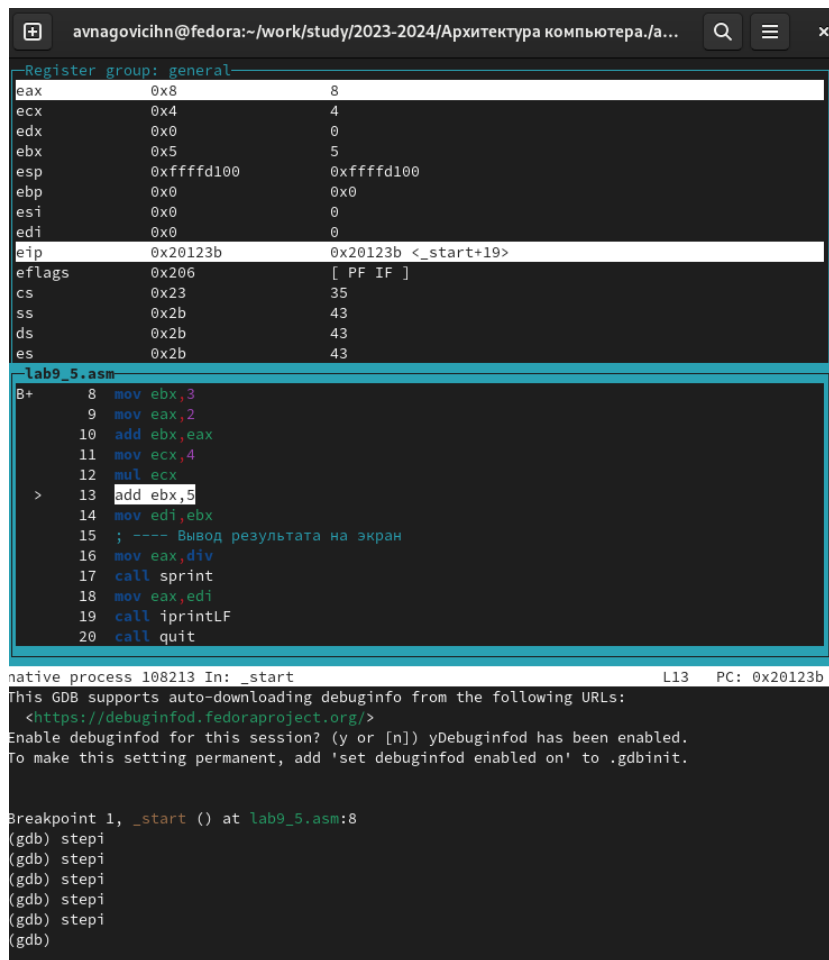


Рис. 3.31: Команды stepi

Изменяю текст программы. (рис. 3.32).



Рис. 3.32: Редактирование файла

Листинг 9.5:

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Создаю исполняемый файл и запускаю его (рис. ??).

```
[avnagovicihn@fedora lab09]$ nasm -f elf lab9_5.asm
[avnagovicihn@fedora lab09]$ ld -m elf_i386 -o lab9_5 lab9_5.o
[avnagovicihn@fedora lab09]$ ./lab9_5
Результат: 25
```

Выводы

В этой лабораторной работе были приобретены навыки написания программ с использованием подпрограмм, а также были рассмотрены методы отладки при помощи GDB и его основные возможности.