

Отчёт по лабораторной работе №9

Понятие подпрограммы. Отладчик GDB

Новикова Анастасия Андреевна

Содержание

1	Цель работы	6
2	Задание	7
3	Выполнение лабораторной работы	8
3.1	Реализация подпрограмм в NASM	8
3.2	Отладка программ с помощью GDB	12
3.3	Добавление точек останова	16
3.4	Работа с данными программы в GDB	17
3.5	Обработка аргументов командной строки в GDB	20
3.6	Задание для самостоятельной работы	21
4	Выводы	28

Список иллюстраций

3.1	Создание каталога	8
3.2	Копирование файла	8
3.3	Создание файла	8
3.4	Редактирование файла	9
3.5	Запуск программы	10
3.6	Редактирование файла	10
3.7	Запуск программы	12
3.8	Создание файла	12
3.9	Редактирование файла	13
3.10	Запуск исполняемого файла	13
3.11	Запуск программы в отладчике	14
3.12	Установка брейкпоинта	14
3.13	Запуск	14
3.14	Диссассимилированный код программы	15
3.15	Отображение с Intel'овским синтаксисом	15
3.16	Точка останова	16
3.17	Установка точки останова	16
3.18	Точки останова	16
3.19	Отслеживаем регистры	17
3.20	info register	17
3.21	Значение переменной по имени	18
3.22	Значение переменной по адресу	18
3.23	Изменение переменной	18
3.24	Изменение второй переменной	18
3.25	Изменение значений в разные форматы	19
3.26	Изменение значений ebx	19
3.27	Копирование файла	20
3.28	Создание файла	20
3.29	Запуск программы с точкой останова	20
3.30	Регистр esp	21
3.31	Позиции стека	21
3.32	Создание файла	21
3.33	Редактирование файла	22
3.34	Запуск программы	24
3.35	Редактирование файла	24
3.36	Запуск программы в отладчике	25
3.37	Действия в отладчике	26

3.38 Запуск программы 27

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. Задание для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация подпрограмм в NASM

Создаю каталог для выполнения работы №9 и перехожу в него(рис. 3.1).

```
aanovikova123@fedora:~$ mkdir ~/work/arch-pc/lab09  
aanovikova123@fedora:~$ cd ~/work/arch-pc/lab09
```

Рис. 3.1: Создание каталога

Копирую файл in_out.asm в созданный каталог, так как он понадобится для написания программ (рис. 3.3).

```
aanovikova123@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab06/in_ out.asm ~/work/arch-pc/lab09  
aanovikova123@fedora:~/work/arch-pc/lab09$ ls  
in_out.asm
```

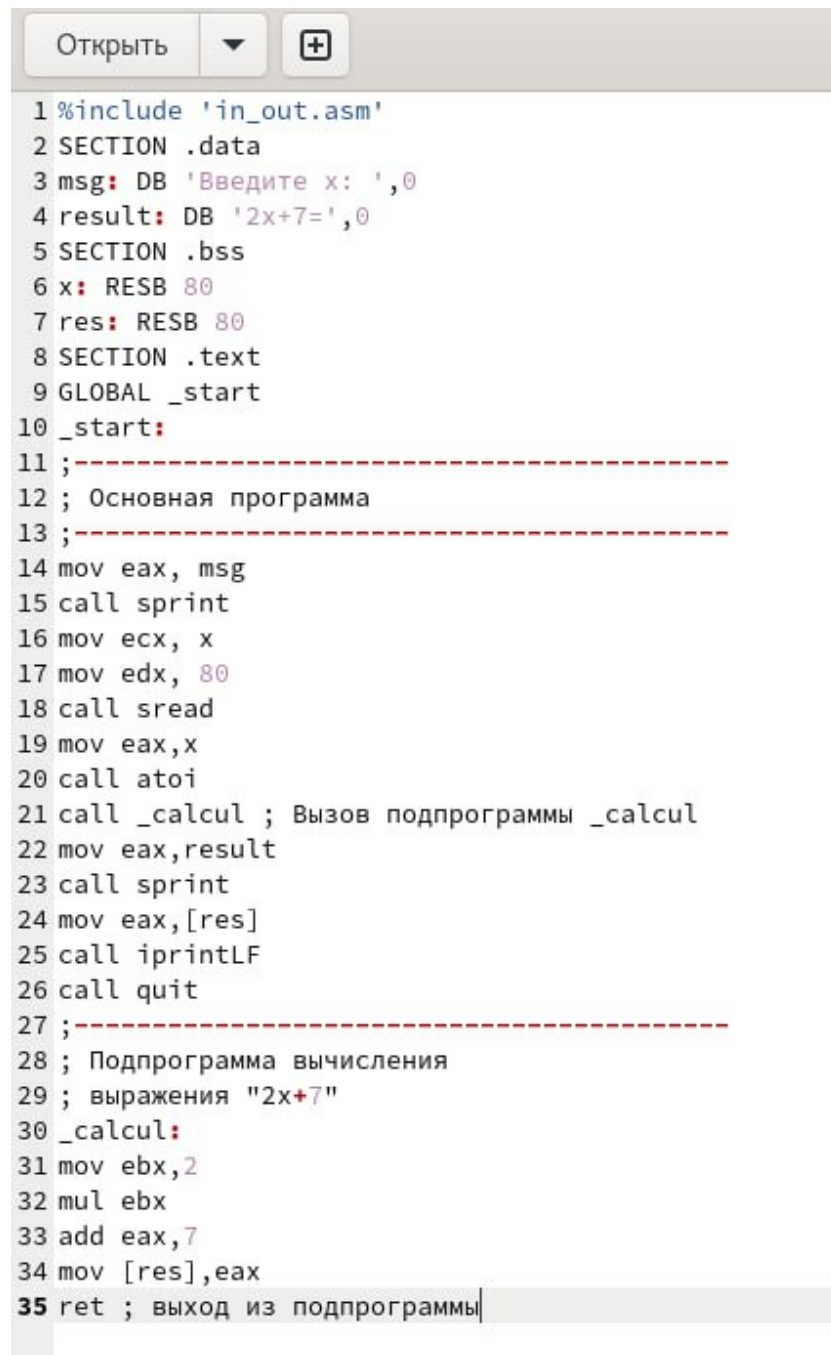
Рис. 3.2: Копирование файла

Создаю файл lab09-1.asm в новом каталоге (рис. 3.3).

```
aanovikova123@fedora:~/work/arch-pc/lab09$ touch lab09-1.asm  
aanovikova123@fedora:~/work/arch-pc/lab09$ gedit lab09-1.asm
```

Рис. 3.3: Создание файла

Открываю файл в текстовом редакторе и переписываю код программы из листинга 9.1 (рис. 3.4).



```
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 iprintLF
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. 3.4: Редактирование файла

Создаю объектный файл программы и после компоновки запускаю его (рис. 3.5). Код с подпрограммой работает успешно.

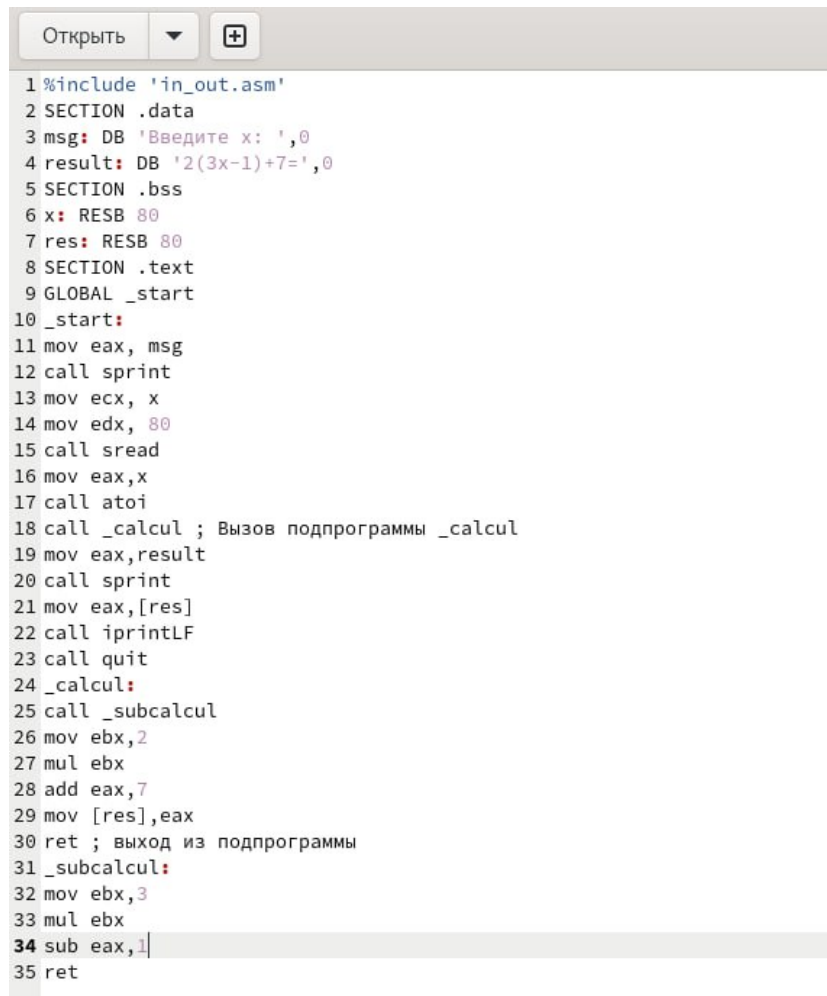
```

aanovikova123@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
aanovikova123@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2x+7=17
aanovikova123@fedora:~/work/arch-pc/lab09$

```

Рис. 3.5: Запуск программы

Изменяю текст файла, добавив подпрограмму `sub_calcul` в подпрограмму `_calcul` (рис. 3.6).



```

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

```

Рис. 3.6: Редактирование файла

```

#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 ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx, 3

```

```
mul ebx
sub eax,1
ret
```

Запускаю исполняемый файл (рис. 3.7). Программа работает верно.

```
aanovikova123@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
aanovikova123@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 3
2(3x-1)+7=23
aanovikova123@fedora:~/work/arch-pc/lab09$
```

Рис. 3.7: Запуск программы

3.2 Отладка программ с помощью GDB

Создаю файл lab09-2.asm, используя команду touch (рис. 3.8).

```
aanovikova123@fedora:~/work/arch-pc/lab09$ touch lab09-2.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ gedit lab09-2.asm
```

Рис. 3.8: Создание файла

Записываю код программы из листинга 9.2, который выводит сообщение Hello world (рис. 3.9).

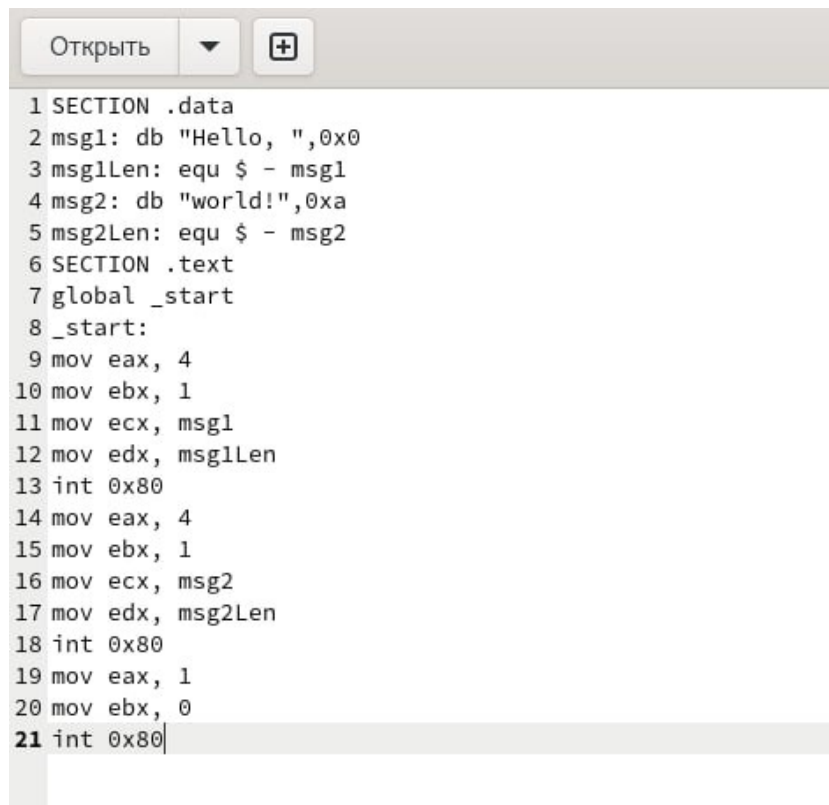


Рис. 3.9: Редактирование файла

Получаю исполняемый файл. Для работы с GDB провожу трансляцию программ с ключом “-g” и загружаю исполняемый файл в отладчик (рис. 3.10).

```
aanovikova123@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
aanovikova123@fedora:~/work/arch-pc/lab09$ gdb lab09-2
```

Рис. 3.10: Запуск исполняемого файла

Проверяю работу программы в оболочке GDB с помощью команды run (рис. 3.11).

```

GNU gdb (Fedora Linux) 14.2-1.fc40
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 lab09-2...
(gdb) run
Starting program: /home/aanovikova123/work/arch-pc/lab09/lab09-2

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Hello, world!
[Inferior 1 (process 4976) exited normally]
(gdb)

```

Рис. 3.11: Запуск программы в отладчике

Для более подробного анализа устанавливаю брейкпоинт на метку `_start`, с которой начинается выполнение ассемблерной программы (рис. 3.12)

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb)

```

Рис. 3.12: Установка брейкпоинта

Запускаю её (рис. 3.13).

```

(gdb) run
Starting program: /home/aanovikova123/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb)

```

Рис. 3.13: Запуск

С помощью команды `“disassemble _start”` просматриваю дисассимилированный код программы (рис. 3.14).

```

(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     $0x4,%eax
    0x08049005 <+5>:      mov     $0x1,%ebx
    0x0804900a <+10>:     mov     $0x804a000,%ecx
    0x0804900f <+15>:     mov     $0x8,%edx
    0x08049014 <+20>:     int     $0x80
    0x08049016 <+22>:     mov     $0x4,%eax
    0x0804901b <+27>:     mov     $0x1,%ebx
    0x08049020 <+32>:     mov     $0x804a008,%ecx
    0x08049025 <+37>:     mov     $0x7,%edx
    0x0804902a <+42>:     int     $0x80
    0x0804902c <+44>:     mov     $0x1,%eax
    0x08049031 <+49>:     mov     $0x0,%ebx
    0x08049036 <+54>:     int     $0x80
End of assembler dump.
(gdb)

```

Рис. 3.14: Диссассимилированный код программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду “set disassembly-flavor intel” (рис. 3.15).

```

(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. 3.15: Отображение с Intel'овским синтаксисом

Основное различие заключается в том, что в режиме Intel пишется сначала сама команда, а потом её машинный код, в то время как в режиме АТТ идет сначала машинный код, а только потом сама команда.

3.3 Добавление точек останова

Проверяю наличие точки останова с помощью команды `info breakpoints (i b)` (рис. 3.16).

```
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y   0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
(gdb)
```

Рис. 3.16: Точка останова

Устанавливаю ещё одну точку останова по адресу инструкции, которую можно найти в средней части в левом столбце соответствующей инструкции (рис. 3.17).

```
(gdb) break *0x08049031
Breakpoint 2 at 0x08049031: file lab09-2.asm, line 20.
(gdb)
```

Рис. 3.17: Установка точки останова

Просматриваю информацию о точках останова (рис. 3.18).

```
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint      keep y   0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint      keep y   0x08049031 lab09-2.asm:20
(gdb)
```

Рис. 3.18: Точки останова

3.4 Работа с данными программы в GDB

Выполняю 5 инструкций командой si(рис. 3.19). Во время выполнения команд менялись регистры ebx, ecx, edx, eax, eip.

```
(gdb) si
10      mov ebx, 1
(gdb) si
11      mov ecx, msg1
(gdb) si
12      mov edx, msg1Len
(gdb) si
13      int 0x80
(gdb) si
Hello, 14      mov eax, 4
(gdb) 
```

Рис. 3.19: Отслеживаем регистры

Просматриваю содержимое регистров с помощью команды info registers (рис. 3.20).

```
(gdb) i r
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
```

Рис. 3.20: info register

Узнаю значение переменной msg1 по имени (рис. 3.21).

```
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
```

Рис. 3.21: Значение переменной по имени

Просматриваю значение переменной msg2 по адресу, который можно определить по дизассемблированной инструкции (рис. 3.22).

```
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
```

Рис. 3.22: Значение переменной по адресу

Меняю первый символ переменной msg1 (рис. 3.23).

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
```

Рис. 3.23: Изменение переменной

Также меняю первый символ переменной msg2 (рис. 3.24).

```
(gdb) set {char}&msg2='W'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "World!\n\034"
```

Рис. 3.24: Изменение второй переменной

Вывожу значение регистра edx в различных форматах (в шестнадцатеричном, двоичном и символьном форматах) (рис. 3.25).

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

Рис. 3.25: Изменение значений в разные форматы

С помощью команды set изменяю значение регистра ebx (рис. 3.26).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb) █
```

Рис. 3.26: Изменение значений ebx

Значение регистра отличаются, так как в первом случае мы выводим код символа 2, который в десятичной системе счисления равен 50, а во втором случае выводится число 2, представленное в этой же системе.

3.5 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, который выводит на экран аргументы, в файл с именем lab09-3.asm (рис. 3.27).

```
aanovikova123@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
aanovikova123@fedora:~/work/arch-pc/lab09$
```

Рис. 3.27: Копирование файла

Создаю исполняемый файл, используя ключ `-args` для загрузки программы в GDB. Загружаю исполняемый файл, указав аргументы (рис. 3.28).

```
aanovikova123@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ d -m elf_i386 -o lab09-3 lab09-3.o
bash: d: команда не найдена...
aanovikova123@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o
aanovikova123@fedora:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Fedora Linux) 14.2-1.fc40
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 lab09-3...
(gdb)
```

Рис. 3.28: Создание файла

Устанавливаю точку останова перед первой инструкцией в программе и запускаю её (рис. 3.29).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/aanovikova123/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

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

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb)
```

Рис. 3.29: Запуск программы с точкой останова

Просматриваю адрес вершины стека, который хранится в регистре esp (рис. 3.30).

```
(gdb) x/x $esp
0xffffd020: 0x00000005
(gdb) █
```

Рис. 3.30: Регистр esp

Ввожу другие позиции стека- в отличие от адресов, располагается адрес в памяти: имя, первый аргумент, второй и т.д (рис. 3.31).

```
(gdb) x/s *(void**)(esp + 4)
0xffffd1e6: "/home/aanovikova123/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd215: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd227: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd238: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd23a: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 3.31: Позиции стека

Шаг изменения адреса равен 4, потому что адресные регистры имеют размерность 32 бита(4 байта).

3.6 Задание для самостоятельной работы

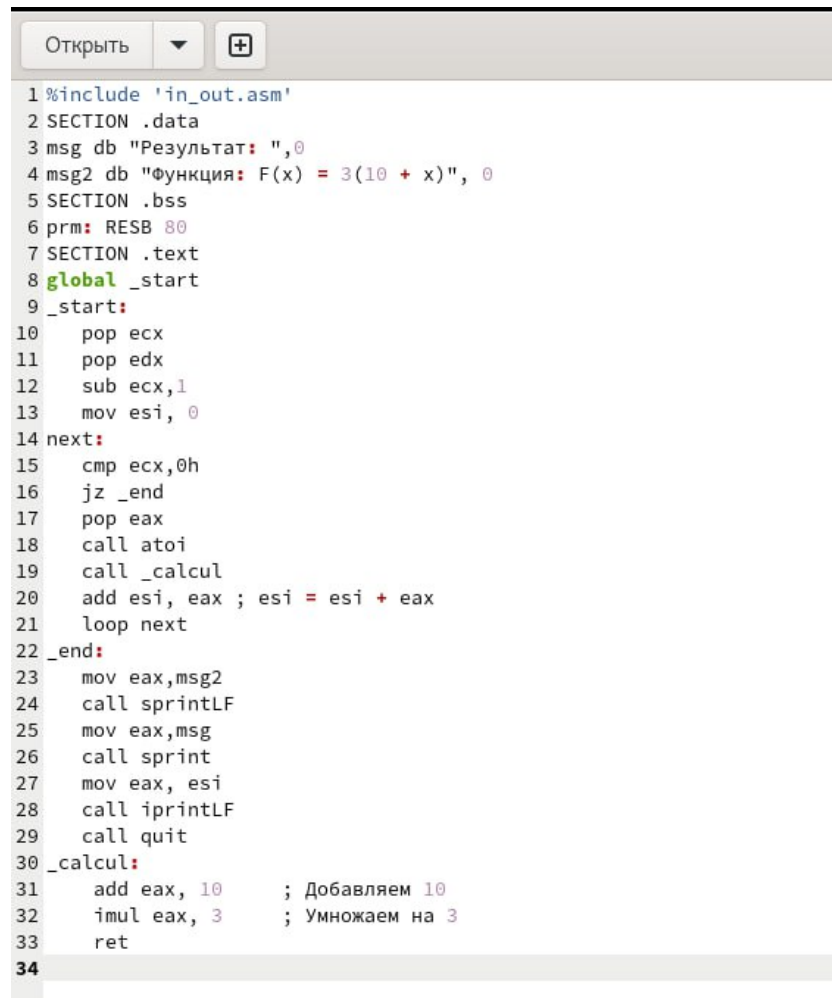
ЗАДАНИЕ 1

Создаю файл для первого самостоятельного задания, который будет называться lab09-4.asm (рис. 3.32).

```
aanovikova123@fedora:~/work/arch-pc/lab09$ touch lab09-4.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ gedit lab09-4.asm
```

Рис. 3.32: Создание файла

Редактирую код программы lab8-4.asm, добавив подпрограмму, которая вычисляет значения функции $f(x)$ (рис. 3.33).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 msg2 db "Функция: F(x) = 3(10 + x)", 0
5 SECTION .bss
6 prm: RESB 80
7 SECTION .text
8 global _start
9 _start:
10  pop ecx
11  pop edx
12  sub ecx,1
13  mov esi, 0
14 next:
15  cmp ecx,0h
16  jz _end
17  pop eax
18  call atoi
19  call _calcul
20  add esi, eax ; esi = esi + eax
21  loop next
22 _end:
23  mov eax,msg2
24  call sprintf
25  mov eax,msg
26  call sprintf
27  mov eax, esi
28  call iprintLF
29  call quit
30 _calcul:
31  add eax, 10 ; Добавляем 10
32  imul eax, 3 ; Умножаем на 3
33  ret
34
```

Рис. 3.33: Редактирование файла

```
%include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
msg2 db "Функция: F(x) = 3(10 + x)", 0
SECTION .bss
prm: RESB 80
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
    add esi, eax ; esi = esi + eax
    loop next
_end:
    mov eax,msg2
    call sprintf
    mov eax,msg
    call sprintf
    mov eax, esi
    call iprintLF
    call quit
_calcul:
    add eax, 10      ; Добавляем 10
    imul eax, 3      ; Умножаем на 3
    ret

```

Создаю исполняемый файл и ввожу аргументы (рис. 3.34). Программа работает верно.

```

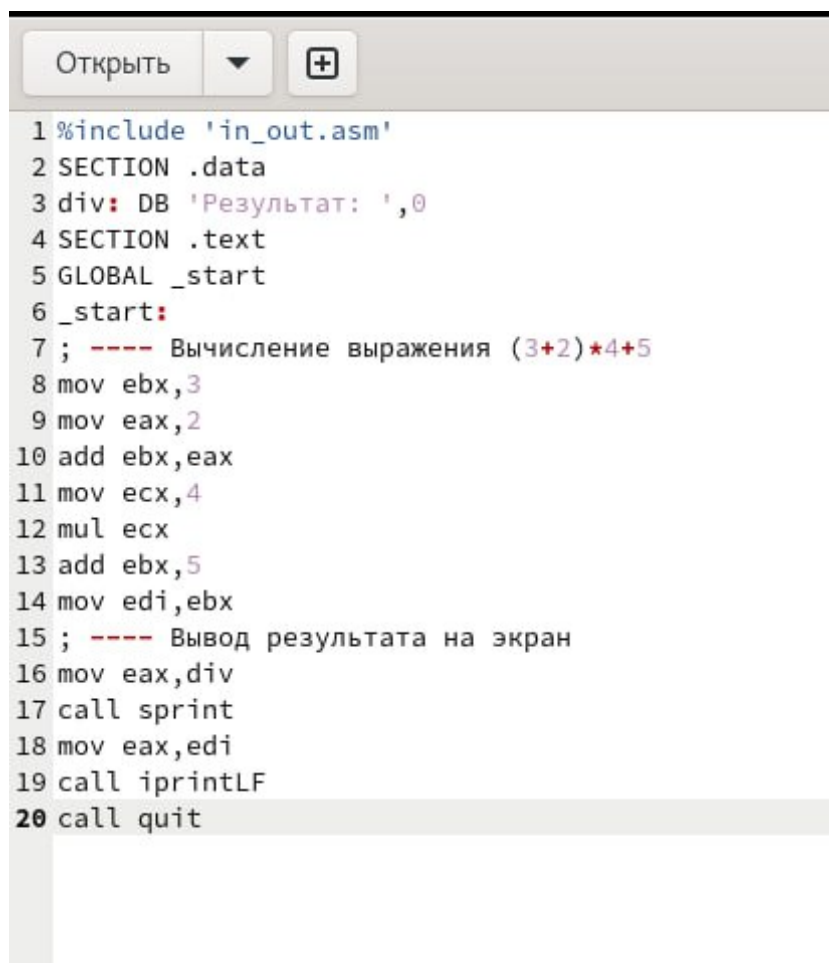
aanovikova123@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
aanovikova123@fedora:~/work/arch-pc/lab09$ ./lab09-4 1 2 3 4
Функция: F(x) = 3(10 + x)
Результат: 150

```

Рис. 3.34: Запуск программы

ЗАДАНИЕ 2

Создаю файл и ввожу код из листинга 9.3 (рис. 3.35).



```

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. 3.35: Редактирование файла

Открываю файл в отладчике GDB и запускаю программу (рис. 3.36). Программа выдает ответ 10.


```

aanovikova123@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
aanovikova123@fedora:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Fedora Linux) 14.2-1.fc40
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 lab09-5...
(gdb) run
Starting program: /home/aanovikova123/work/arch-pc/lab09/lab09-5

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.
Результат: 10
[Inferior 1 (process 9275) exited normally]
(gdb)

```

Рис. 3.36: Запуск программы в отладчике

Просматриваю дисассимилированный код программы, ставлю точку останова перед прибавлением 5 и открываю значения регистров на данном этапе (рис. 3.37).

```

(gdb) disassemble _start
Dump of assembler code for function _start:
0x080490e8 <+0>:  mov    $0x3,%ebx
0x080490ed <+5>:  mov    $0x2,%eax
0x080490f2 <+10>: add    %eax,%ebx
0x080490f4 <+12>:  mov    $0x4,%ecx
0x080490f9 <+17>:  mul    %ecx
0x080490fb <+19>:  add    $0x5,%ebx
0x080490fe <+22>:  mov    %ebx,%edi
0x08049100 <+24>:  mov    $0x804a000,%eax
0x08049105 <+29>:  call   0x804900f <sprint>
0x0804910a <+34>:  mov    %edi,%eax
0x0804910c <+36>:  call   0x8049086 <iprintLF>
0x08049111 <+41>:  call   0x80490db <quit>
End of assembler dump.
(gdb) b *0x080490fb
Breakpoint 1 at 0x80490fb: file lab09-5.asm, line 13.
(gdb) run
Starting program: /home/aanovikova123/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:13
13      add    ebx,5
(gdb) i r
eax            0x8            8
ecx            0x4            4
edx            0x0            0
ebx            0x5            5
esp            0xffffd060     0xffffd060
ebp            0x0            0x0
esi            0x0            0
edi            0x0            0
eip            0x80490fb       0x80490fb <_start+19>
eflags        0x202          [ IF ]
cs             0x23           35
ss             0x2b           43
ds             0x2b           43
es             0x2b           43
fs             0x0            0
gs             0x0            0
(gdb)

```

Рис. 3.37: Действия в отладчике

Как можно увидеть, регистр ecx со значением 4 умножается не на ebx, сло-
 женным с eax, а только с eax со значением 2. Значит нужно поменять значения
 регистров(например присвоить eax значение 3 и просто прибавит 2. После изме-
 нений программа будет выглядеть следующим образом:

```

#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ',0

SECTION .text
GLOBAL _start
_start:

```

```

; ---- Вычисление выражения (3+2)*4+5
mov eax,3
mov ebx,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

```

Пробуем запустить программу (рис. 3.38). Она работает верно.

```

aanovikova123@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
aanovikova123@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
aanovikova123@fedora:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
aanovikova123@fedora:~/work/arch-pc/lab09$

```

Рис. 3.38: Запуск программы

4 Выводы

В ходе лабораторной работы были приобретены навыки написания программ с использованием подпрограмм. Также было знакомство с методами отладки при помощи GDB и его основными возможностями.