

Отчёт по лабораторной работе №9

Дисциплина: архитектура компьютера

Учаева Алёна Сергеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Реализация подпрограмм в NASM	7
4.2	Отладка программ с помощью GDB	10
4.3	Добавление точек останова	13
4.3.1	Работа с данными программы в GDB	14
4.3.2	Обработка аргументов командной строки в GDB	17
4.4	Задание для самостоятельной работы	19
5	Выводы	23
	Список литературы	24

Список иллюстраций

4.1	Создание файла	7
4.2	Запуск исполняемого файла	7
4.3	Редактирование файла	8
4.4	Запуск исполняемого файла	8
4.5	Создание файла	10
4.6	Проверка программы отладчиком	11
4.7	Запуск отладчика с брейкпоинтом	11
4.8	Дисассимилирование программы	12
4.9	Дисассимилирование программы	12
4.10	Режим псевдографики	13
4.11	Режим псевдографики	14
4.12	Установка точки останова	14
4.13	Просмотр содержимого регистров	15
4.14	Просмотр содержимого переменных	15
4.15	Изменение содержимого переменных	16
4.16	Просмотр содержимого регистров	16
4.17	Редактирование содержимого регистра ebx	17
4.18	Создание исполняемого файла	17
4.19	Установка точки останова	18
4.20	Запуск отладчика	18
4.21	Редактирование программы	19
4.22	Просмотр программы	21
4.23	Запуск исправленной программы	21

1 Цель работы

Приобрести навыки написания программ с использованием подпрограмм. Ознакомиться с методами отладки при помощи GDB и его основными возможностями.

2 Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Задание для самостоятельной 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 для выполнения лабораторной работы № 9, перехожу в него и создайте файл lab09-1.asm (рис. 4.1).

```
alena@fedora:~$ mkdir ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/lab09
alena@fedora:~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/lab09
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ touch lab09-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

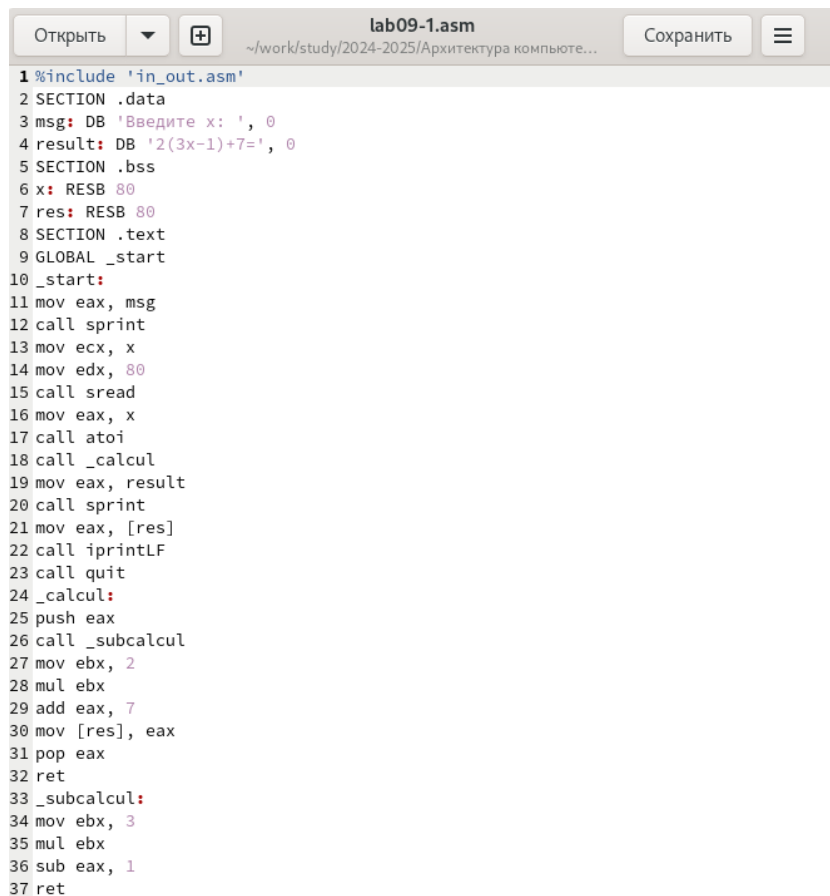
Рис. 4.1: Создание файла

Ввожу в файл lab09-1.asm текст программы из листинга 9.1. Создаю исполняемый файл и проверяю его работу. Программа выполняет вычисление функции (рис. 4.2).

```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab09-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./lab09-1
Введите x: 7
2x+7=21
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 4.2: Запуск исполняемого файла

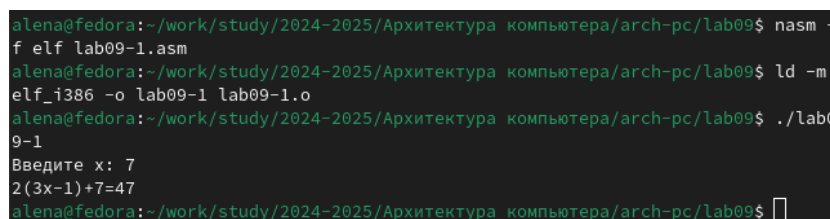
Изменяю текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`. Программа вычисляет значение функции для выражения $f(g(x))$ (рис. 4.3).



```
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
19 mov eax, result
20 call sprint
21 mov eax, [res]
22 call iprintLF
23 call quit
24 _calcul:
25 push eax
26 call _subcalcul
27 mov ebx, 2
28 mul ebx
29 add eax, 7
30 mov [res], eax
31 pop eax
32 ret
33 _subcalcul:
34 mov ebx, 3
35 mul ebx
36 sub eax, 1
37 ret
```

Рис. 4.3: Редактирование файла

Создаю исполняемый файл и проверяю его работу. Программа выполняет вычисление функции (рис. 4.4).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab09-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./lab09-1
Введите x: 7
2(3x-1)+7=47
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$
```

Рис. 4.4: Запуск исполняемого файла

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

```

4.2 Отладка программ с помощью GDB

Создаю файл lab09-2.asm с текстом программы из Листинга 9.2. (Программа печати сообщения Hello world!) (рис. 4.5).

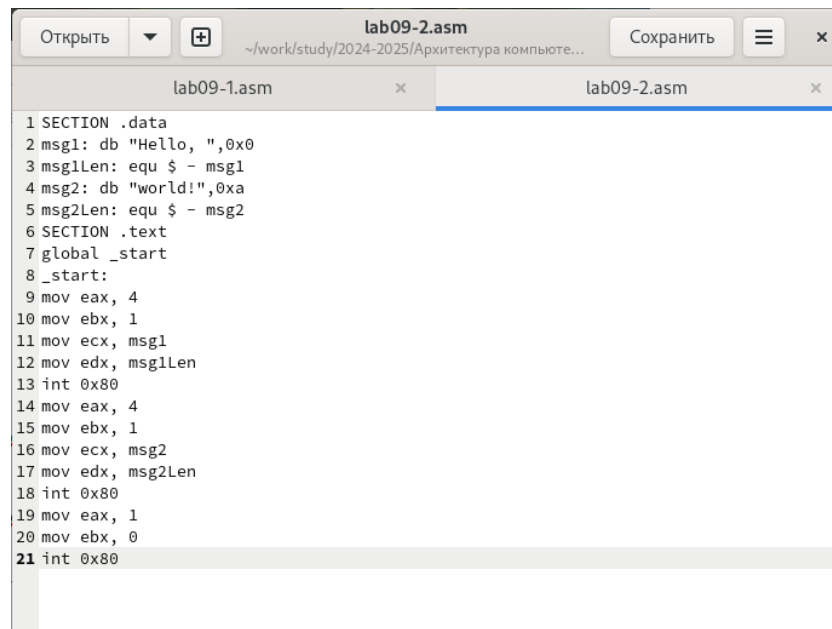


Рис. 4.5: Создание файла

Транслирую с созданием файла листинга и отладки, componую и запускаю в отладчике, программа работает корректно (рис. 4.6).

```
alena@fedora:~/work/study/2024-2025/Архитектура компьют...
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/alena/work/study/2024-2025/Архитектура компьютера/arch-p
c/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 52120) exited normally]
(gdb) run
Starting program: /home/alena/work/study/2024-2025/Архитектура компьютера/arch-p
c/lab09/lab09-2
Hello, world!
[Inferior 1 (process 52155) exited normally]
(gdb) 
```

Рис. 4.6: Проверка программы отладчиком

Для более подробного анализа программы устанавливаю брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запускаю её (рис. 4.7).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/alena/work/study/2024-2025/Архитектура компьютера/arch-p
c/lab09/lab09-2

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

Рис. 4.7: Запуск отладчика с брейкпоинтом

Далее смотритю дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 4.8).

```
alena@fedora:~/work/study/2024-2025/Архитектура компьют...
Breakpoint 1 at 0x08049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/alena/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
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     $0x804a000,%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. 4.8: Дисассимилирование программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 4.9).

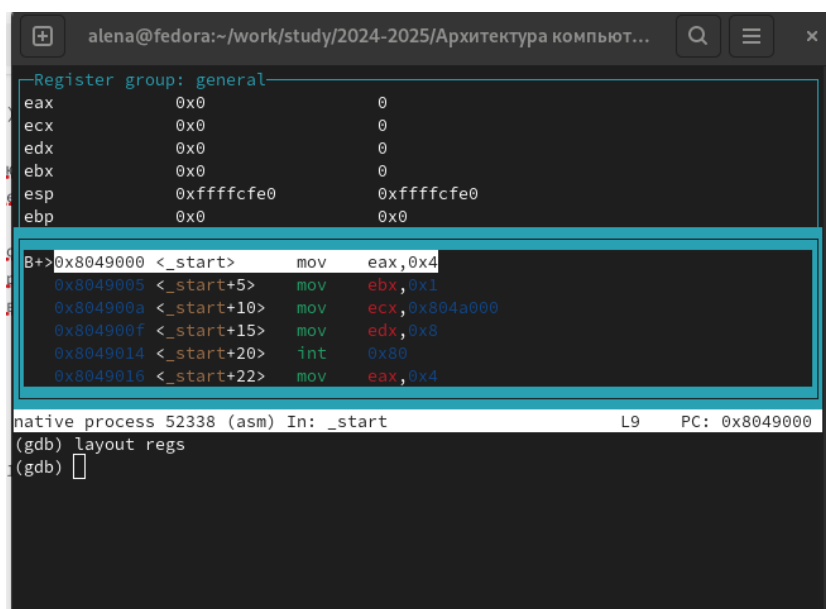
```
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:   mov     eax,0x4
0x08049005 <+5>:   mov     ebx,0x1
0x0804900a <+10>:  mov     ecx,0x804a000
0x0804900f <+15>:  mov     edx,0x8
0x08049014 <+20>:  int     0x80
0x08049016 <+22>:  mov     eax,0x4
0x0804901b <+27>:  mov     ebx,0x1
0x08049020 <+32>:  mov     ecx,0x804a000
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. 4.9: Дисассимилирование программы

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 опе-

ранды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

Далее включаю режим псевдографики для более удобного анализа программы (рис. 4.10).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьют...
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfe0 0xffffcfe0
ebp      0x0      0x0

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

native process 52338 (asm) In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) 
```

Рис. 4.10: Режим псевдографики

4.3 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 4.11).

```

+ alena@fedora:~/work/study/2024-2025/Архитектура компьют...
Register group: general
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffcfe0 0xffffcfe0
ebp 0x0 0x0

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

native process 52338 (asm) In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab09-2.asm:9
breakpoint already hit 1 time
(gdb)

```

Рис. 4.11: Режим псевдографики

Устанавливаю еще одну точку останова по адресу инструкции (рис. 4.12).

```

+ alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — ...
Register group: general
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffcfe0 0xffffcfe0
ebp 0x0 0x0

0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
b+ 0x8049031 <_start+49> mov ebx,0x0
0x8049036 <_start+54> int 0x80
0x8049038 add BYTE PTR [eax],al
0x804903a add BYTE PTR [eax],al

native process 52338 (asm) In: _start L9 PC: 0x8049000
Note: breakpoint 2 also set at pc 0x8049031.
Breakpoint 3 at 0x8049031: file lab09-2.asm, line 20.
(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
3 breakpoint keep y 0x08049031 lab09-2.asm:20
(gdb)

```

Рис. 4.12: Установка точки останова

4.3.1 Работа с данными программы в GDB

Посмотреть содержимое регистров также можно с помощью команды `info registers` (рис. 4.13).

```

alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — ...
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
b+ 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

native process 55094 (asm) In: _start L9 PC: 0x8049000
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffcfe0 0xffffcfe0
ebp 0x0 0x0
esi 0x0 0
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 4.13: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. 4.14).

```

alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — ...
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
b+ 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

native process 55094 (asm) In: _start L9 PC: 0x8049000
ss 0x2b 43
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)

```

Рис. 4.14: Просмотр содержимого переменных

Меняю содержимое переменных по имени и по адресу (рис. 4.15).

```

alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — ...
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
b+ 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

native process 55094 (asm) In: _start L9 PC: 0x8049000
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/1sb &msg1
0x804a008 <msg1>: "hello, "
(gdb) set {char}msg2='a'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "aorld!\n\034"
(gdb)

```

Рис. 4.15: Изменение содержимого переменных

Вывожу в различных форматах значение регистра edx (рис. 4.16).

```

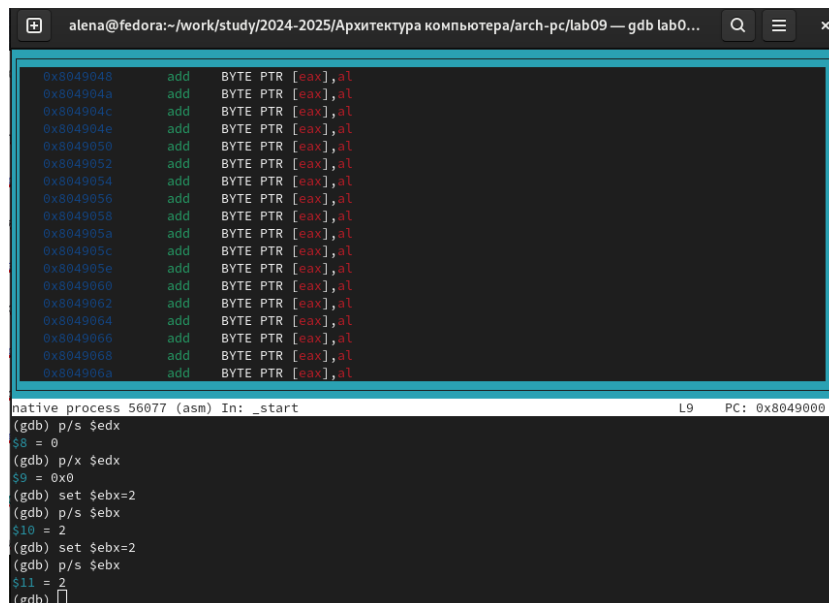
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab0...
0x8049046 add    BYTE PTR [eax],al
0x804904a add    BYTE PTR [eax],al
0x804904c add    BYTE PTR [eax],al
0x804904e add    BYTE PTR [eax],al
0x8049050 add    BYTE PTR [eax],al
0x8049052 add    BYTE PTR [eax],al
0x8049054 add    BYTE PTR [eax],al
0x8049056 add    BYTE PTR [eax],al
0x8049058 add    BYTE PTR [eax],al
0x804905a add    BYTE PTR [eax],al
0x804905c add    BYTE PTR [eax],al
0x804905e add    BYTE PTR [eax],al
0x8049060 add    BYTE PTR [eax],al
0x8049062 add    BYTE PTR [eax],al
0x8049064 add    BYTE PTR [eax],al
0x8049066 add    BYTE PTR [eax],al
0x8049068 add    BYTE PTR [eax],al
0x804906a add    BYTE PTR [eax],al

native process 56077 (asm) In: _start L9 PC: 0x8049000
(gdb) p/t $edx
$7 = 0
(gdb) p/s $edx
$8 = 0
(gdb) p/x $edx
$9 = 0x0
(gdb)

```

Рис. 4.16: Просмотр содержимого регистров

С помощью команды set меняю содержимое регистра ebx (рис. 4.17).



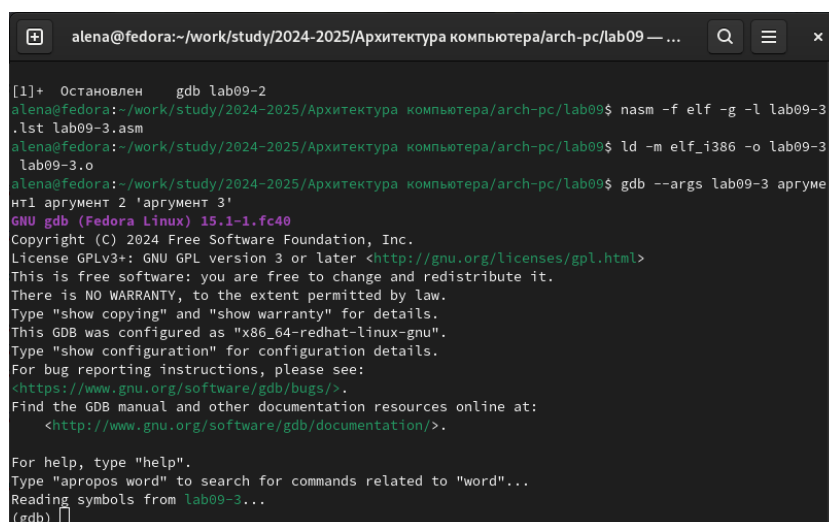
```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — gdb lab0...
0x8049048 add BYTE PTR [eax],al
0x804904a add BYTE PTR [eax],al
0x804904c add BYTE PTR [eax],al
0x804904e add BYTE PTR [eax],al
0x8049050 add BYTE PTR [eax],al
0x8049052 add BYTE PTR [eax],al
0x8049054 add BYTE PTR [eax],al
0x8049056 add BYTE PTR [eax],al
0x8049058 add BYTE PTR [eax],al
0x804905a add BYTE PTR [eax],al
0x804905c add BYTE PTR [eax],al
0x804905e add BYTE PTR [eax],al
0x8049060 add BYTE PTR [eax],al
0x8049062 add BYTE PTR [eax],al
0x8049064 add BYTE PTR [eax],al
0x8049066 add BYTE PTR [eax],al
0x8049068 add BYTE PTR [eax],al
0x804906a add BYTE PTR [eax],al

native process 56077 (asm) In: _start L9 PC: 0x8049060
(gdb) p/s $edx
$8 = 0
(gdb) p/x $edx
$9 = 0x0
(gdb) set $ebx=2
(gdb) p/s $ebx
$10 = 2
(gdb) set $ebx=2
(gdb) p/s $ebx
$11 = 2
(gdb)
```

Рис. 4.17: Редактирование содержимого регистра ebx

4.3.2 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm и создаю исполняемый файл (рис. 4.18).

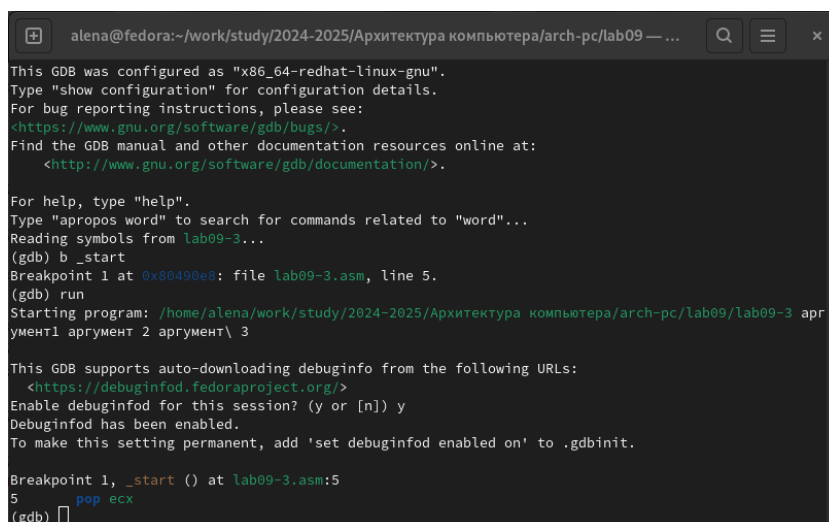


```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — ...
[1]+ Остановлен gdb lab09-2
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab09-3
.lst lab09-3.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab09-3
lab09-3.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ gdb --args lab09-3 аргуме
нт1 аргумент 2 'аргумент 3'
GNU gdb (Fedora Linux) 15.1-1.fc40
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-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. 4.18: Создание исполняемого файла

Устанавливаю точку останова перед первой инструкцией в программе и запус-

каю ее (рис. 4.19).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — ...
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/>.

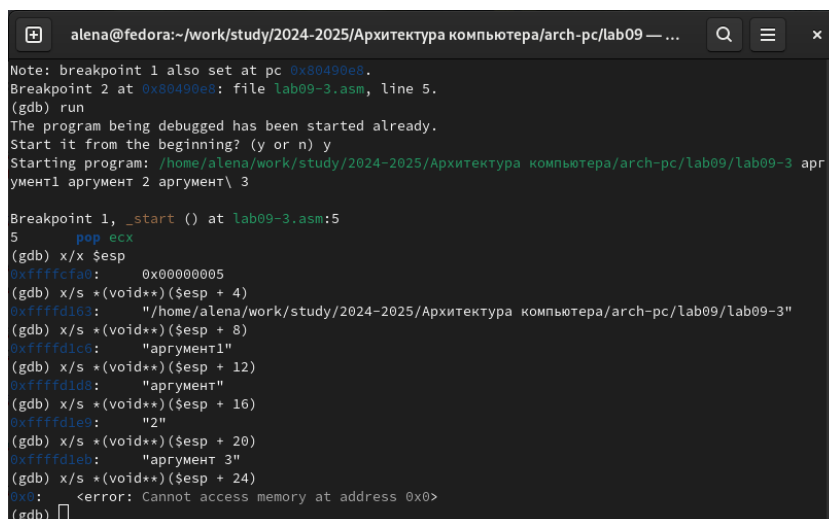
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/alena/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab09-3 аргумент1 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
(gdb) 
```

Рис. 4.19: Установка точки останова

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. 4.20).



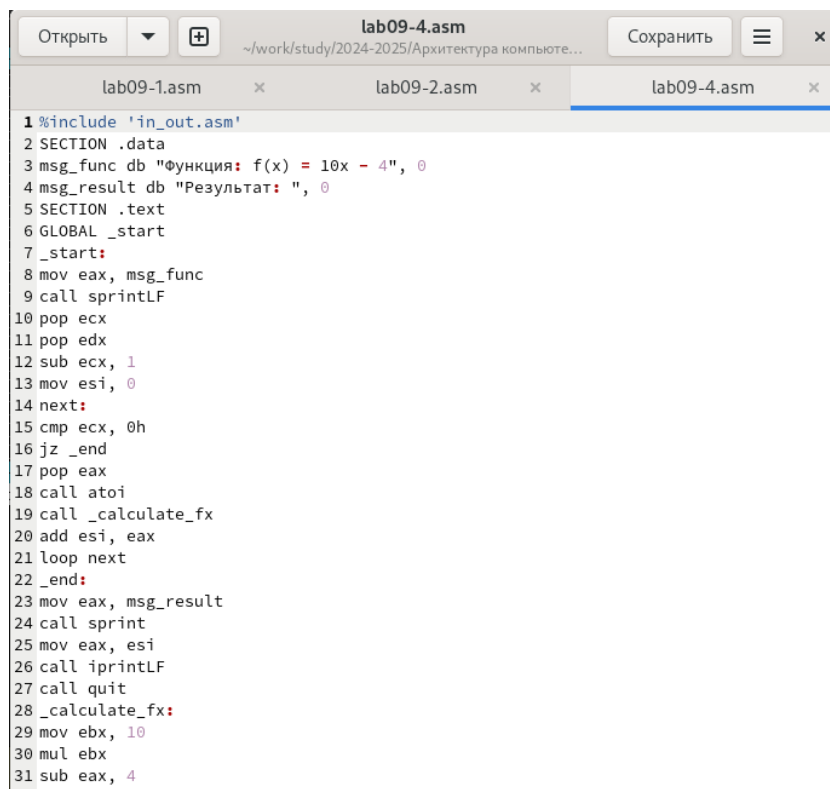
```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09 — ...
Note: breakpoint 1 also set at pc 0x80490e8.
Breakpoint 2 at 0x80490e8: file lab09-3.asm, line 5.
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/alena/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx
(gdb) x/x $esp
0xffffcfa0: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd163: "/home/alena/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd1c6: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd1d8: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd1e9: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd1eb: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb) 
```

Рис. 4.20: Запуск отладчика

4.4 Задание для самостоятельной работы

Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 4.21).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg_func db "Функция: f(x) = 10x - 4", 0
4 msg_result db "Результат: ", 0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 mov eax, msg_func
9 call sprintf
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 _calculate_fx
20 add esi, eax
21 loop next
22 _end:
23 mov eax, msg_result
24 call sprintf
25 mov eax, esi
26 call iprintf
27 call quit
28 _calculate_fx:
29 mov ebx, 10
30 mul ebx
31 sub eax, 4
```

Рис. 4.21: Редактирование программы

Новая программа:

```
%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 sprintf
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 sprintf
mov eax, esi
call iprintLF
call quit
_calculate_fx:
mov ebx, 10
mul ebx
sub eax, 4

```

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. 4.22).

```

B> 0x80490e8 <_start> mov ebx, 0x3
0x80490ed <_start+5> mov eax, 0x2
0x80490f2 <_start+10> add ebx, eax
0x80490fa <_start+12> mov ecx, 0x4
0x80490f9 <_start+17> mul ecx
0x80490fb <_start+19> add ebx, 0x5
0x80490fe <_start+22> mov edi, ebx
0x8049100 <_start+24> mov eax, 0x804a000
0x8049105 <_start+29> call 0x804900f <sprint>
0x804910a <_start+34> mov eax, edi
0x804910e <_start+36> call 0x8049086 <iprintf>
0x8049111 <_start+41> call 0x80490db <quit>
0x8049116 add BYTE PTR [eax], al
0x8049118 add BYTE PTR [eax], al
0x804911a add BYTE PTR [eax], al
0x804911c add BYTE PTR [eax], al
0x804911e add BYTE PTR [eax], al
0x8049120 add BYTE PTR [eax], al

native process 57213 (asm) In: _start L7 PC: 0x80490e8
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xffffcfe0 0xffffcfe0
ebp 0x0 0
esi 0x0 0
edi 0x0 0
eip 0x80490e8 0x80490e8 <_start>
eflags 0x202 [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 4.22: Просмотр программы

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. 4.23).

```

alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$ ./lab09-5
Результат: 25
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09$

```

Рис. 4.23: Запуск исправленной программы

Новая программа:

```

#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

При выполнении данной лабораторной работы я приобрела навыки написания программ с использованием подпрограмм. Ознакомилась с методами отладки при помощи GDB и его основными возможностями.

Список литературы

1. Архитектура ЭВМ