

Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Толстых Александра Андреевна

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	8
3.3	Задание для самостоятельной работы	17
4	Выводы	22

Список иллюстраций

3.1	Создание каталога и файла	7
3.2	Написание программы	7
3.3	Запуск программы	8
3.4	Изменение программы	8
3.5	Запуск программы	8
3.6	Создание файла	9
3.7	Написание программы	9
3.8	Создание и загрузка исполняемого файла в отладчик	9
3.9	Запуск программы	10
3.10	Запуск программы	10
3.11	Просмотр программы	10
3.12	Просмотр программы с другим отображением	11
3.13	Режим псевдографики	11
3.14	Проверка точек останова	12
3.15	Установка и просмотр точек останова	12
3.16	Выполнение 5 инструкций	13
3.17	Просмотр значения переменной по имени	14
3.18	Просмотр значения переменной по адресу	14
3.19	Изменение переменной	14
3.20	Изменение переменной	14
3.21	Вывод значения регистра	14
3.22	Изменение значения регистра	15
3.23	Завершение выполнения и выход из отладчика	15
3.24	Создание копии файла	15
3.25	Создание исполняемого файла	15
3.26	Загрузка исполняемого файла в отладчик	16
3.27	Установка точки и запуск программы	16
3.28	Проверка количества аргументов	16
3.29	Позиции стека	17
3.30	Создание копии программы	17
3.31	Изменение программы	17
3.32	Запуск программы	18
3.33	Создание файла	18
3.34	Написание программы	18
3.35	Создание и загрузка исполняемого файла в отладчик	19
3.36	Настройка отображения команд	19
3.37	Запуск программы	20

3.38 Анализ регистров	20
3.39 Изменение регистров	20
3.40 Запуск программы	21

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. Задание для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Реализация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы № 9, перехожу в него и создаю файл lab09-1.asm (рис. 3.1).

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

Рис. 3.1: Создание каталога и файла

Ввожу в файл lab09-1.asm текст программы из листинга 9.1 (рис. 3.2).



```
21
22     mov ecx, x
23     mov edx, 00
24     call sread
25
26     mov eax, x
27     call atoi
28
29     call _calcul           ; Вызов подпрограммы _calcul
30
31     mov eax, result
32     call sprint
33     mov eax, [res]
34     call iprintfLF
35
36     call quit
37
38 ; -----
39 ; Подпрограмма вычисления
40 ; выражения "2x+7"
41
42     _calcul:
43         mov ebx, 2
44         mul ebx
45         add eax, 7
46         mov [res], eax
47
48     ret                   : выход из подпрограммы
```

Рис. 3.2: Написание программы

Создаю исполняемый файл и проверяю его работу (рис. 3.3).

```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 2
2x+7=11
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$

```

Рис. 3.3: Запуск программы

Изменяю текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$ (рис. 3.4).

```

lab09-1.asm
~/work/arch-pc/lab09
Ln 42, Col 17
35
36     call quit
37
38 ;-----
39 ; Подпрограмма вычисления
40 ; выражения "2x+7"
41
42     _calcul:
43
44         call _subcalcul
45
46         mov ebx,2
47         mul ebx
48         add eax,7
49         mov [res],eax
50         ret
51
52 ;-----
53 ; Подпрограмма вычисления
54 ; выражения "3x-1"
55
56     _subcalcul:
57
58         mov ebx,3
59         mul ebx
60         sub eax,1
61         ret

```

Рис. 3.4: Изменение программы

Создаю исполняемый файл и проверяю его работу (рис. 3.5).

```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 2
2(3x-1)+7=17
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$

```

Рис. 3.5: Запуск программы

3.2 Отладка программ с помощью GDB

Создаю файл `lab09-2.asm` (рис. 3.6).


```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ touch lab09-2.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$

```

Рис. 3.6: Создание файла

Ввожу в него текст из программы из Листинга 9.2 (рис. 3.7).



```

1 SECTION .data
2     msg1: db "Hello, ",0x0
3     msg1Len: equ $ - msg1
4     msg2: db "world!",0xa
5     msg2Len: equ $ - msg2
6
7 SECTION .text
8     global _start
9
10 _start:
11     mov eax, 4
12     mov ebx, 1
13     mov ecx, msg1
14     mov edx, msg1Len
15     int 0x80
16
17     mov eax, 4
18     mov ebx, 1
19     mov ecx, msg2
20     mov edx, msg2Len
21     int 0x80
22
23     mov eax, 1
24     mov ebx, 0
25     int 0x80

```

Рис. 3.7: Написание программы

Получаю исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ необходимо проводить с ключом '-g'. Загружаю исполняемый файл в отладчик gdb (рис. 3.8).

```

aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb)

```

Рис. 3.8: Создание и загрузка исполняемого файла в отладчик

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

```
(gdb) run
Starting program: /home/aatolstikh/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 47551) exited normally]
(gdb)
```

Рис. 3.9: Запуск программы

Для более подробного анализа программы устанавливаю брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запускаю её (рис. 3.10).

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 11.
(gdb) run
Starting program: /home/aatolstikh/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
11      mov eax, 4
(gdb)
```

Рис. 3.10: Запуск программы

Смотрю дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 3.11).

```
(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. 3.11: Просмотр программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 3.12). Отличие заключается в командах, в дисассимилированном отображении в командах используют `%` и `$`, а в Intel отображение эти символы не используются. На такое отображение удобнее смотреть.

```

(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. 3.12: Просмотр программы с другим отображением

Включаю режим псевдографики для более удобного анализа программы (рис. 3.13).

```

native process 47633 (asm) In: _start      L11  PC: 0x08049000
(gdb) layout regs
(gdb)

```

Рис. 3.13: Режим псевдографики

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверяю это с помощью команды `info breakpoints` (рис. 3.14).

```

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcf00 0xffffcf00
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 47633 (asm) In: _start L11 PC: 0x8049000
(gdb) layout regs
(gdb) info breakpoints
Num      Type             Disp Enb Address      What
1        breakpoint       keep y  0x8049000 lab09-2.asm:11
        breakpoint already hit 1 time
(gdb)

```

Рис. 3.14: Проверка точек останова

Устанавливаю еще одну точку останова по адресу предпоследней инструкции (mov ebx,0x0) и снова смотрю информацию о всех установленных точках останова (рис. 3.15).

```

(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 24.
(gdb) info breakpoints
Num      Type             Disp Enb Address      What
1        breakpoint       keep y  0x8049000 lab09-2.asm:11
2        breakpoint       keep y  0x8049031 lab09-2.asm:24
(gdb)

```

Рис. 3.15: Установка и просмотр точек останова

Выполняю 5 инструкций с помощью команды stepi (или si). Изменяются регистры ebx, ecx, edx, eax (рис. 3.16).

```

aalolstikhk@aolstikhk: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcf00 0xffffcf00
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 48919 (asm) In: _start L12 PC: 0x8049005
(gdb) run
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/aalolstikhk/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
(gdb) si
(gdb)

```

```

aalolstikhk@aolstikhk: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x1      1
esp      0xffffcf00 0xffffcf00
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 48919 (asm) In: _start L13
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/aalolstikhk/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
(gdb) si
(gdb) si
(gdb)

```

```

aalolstikhk@aolstikhk: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x0      0
ebx      0x1      1
esp      0xffffcf00 0xffffcf00
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 48919 (asm) In: _start L14 PC: 0x804900f
Start it from the beginning? (y or n) y
Starting program: /home/aalolstikhk/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

```

aalolstikhk@aolstikhk: ~/work/arch-pc/lab09
Register group: general
eax      0x4      4
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffcf00 0xffffcf00
ebp      0x0      0x0

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

native process 48919 (asm) In: _start L15
Starting program: /home/aalolstikhk/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:11
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

```

aalolstikhk@aolstikhk: ~/work/arch-pc/lab09
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffcf00 0xffffcf00
ebp      0x0      0x0

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,0x804a000

native process 48919 (asm) In: _start L17 PC: 0x8049016

Breakpoint 1, _start () at lab09-2.asm:11
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 3.16: Выполнение 5 инструкций

Смотрю значение переменной `msg1` по имени (рис. 3.17).

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

Рис. 3.17: Просмотр значения переменной по имени

Смотрю значение переменной `msg2` по адресу (рис. 3.18).

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

Рис. 3.18: Просмотр значения переменной по адресу

Изменяю первый символ переменной `msg1` (рис. 3.19).

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

Рис. 3.19: Изменение переменной

Изменяю символ во второй переменной `msg2` (рис. 3.20).

```
(gdb) set {char}0x804a009='e'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "werld!\n\034"
(gdb)
```

Рис. 3.20: Изменение переменной

Вывожу в различных форматах значение регистра `edx` (рис. 3.21).

```
(gdb) p/x $edx
$10 = 0x8
(gdb) p/s $edx
$11 = 8
(gdb) p/a $edx
$12 = 0x8
(gdb)
```

Рис. 3.21: Вывод значения регистра

С помощью команды `set` изменяю значение регистра `ebx` (рис. 3.22). Команда выводит два разных значения, так как в первый раз мы вносим значение 2, а во второй раз регистр равен двум.


```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$15 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$16 = 2
(gdb)

```

Рис. 3.22: Изменение значения регистра

Завершаю выполнение программы с помощью команды `continue` (сокращенно `c`) и выхожу из GDB с помощью команды `quit` (сокращенно `q`) (рис. 3.23).

```

$16 = 2
(gdb) c
Continuing.
world!

Breakpoint 2, _start () at lab09-2.asm:24
(gdb) q

```

Рис. 3.23: Завершение выполнения и выход из отладчика

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` (рис. 3.24).

```

(gdb) layout asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm
~/work/arch-pc/lab09/lab09-3.asm
aamolstihkh@aamolstihkh:~/work/arch-pc/lab09$

```

Рис. 3.24: Создание копии файла

Создаю исполняемый файл (рис. 3.25).

```

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

```

Рис. 3.25: Создание исполняемого файла

Для загрузки в `gdb` программы с аргументами необходимо использовать ключ `-args`. Загружаю исполняемый файл в отладчик, указав аргументы (рис. 3.26).

```

aamolstikhkh@aamolstikhkh:~/work/arch-pc/lab09$
aamolstikhkh@aamolstikhkh:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 '
аргумент 3'
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb)

```

Рис. 3.26: Загрузка исполняемого файла в отладчик

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее (рис. 3.27).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 11.
(gdb) run
Starting program: /home/aamolstikhkh/work/arch-pc/lab09/lab09-3 аргумент1 аргумент 2 аргумент\ 3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
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:11
11      pop ecx                ; Извлекаем из стека в `ecx` количество
(gdb)

```

Рис. 3.27: Установка точки и запуск программы

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы). Убеждаюсь, что там число 5 (рис. 3.28).

```

(gdb) x/x $esp
0xfffffceb0: 0x00000005
(gdb)

```

Рис. 3.28: Проверка количества аргументов

Смотрю все позиции стека (рис. 3.29). По первому адресу хранится адрес, в остальных адресах хранятся элементы. Элементы расположены с интервалом в 4 единицы, так как стек может хранить до 4 байт, и для того чтобы данные

сохранялись нормально и без помех, компьютер использует новый стек для новой информации.

```
(gdb) x/s *(void**)(esp + 4)
0xffffd0ac: "/home/aatolstikh/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd0d9: "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd0eb: "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd0fc: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd0fe: "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 3.29: Позиции стека

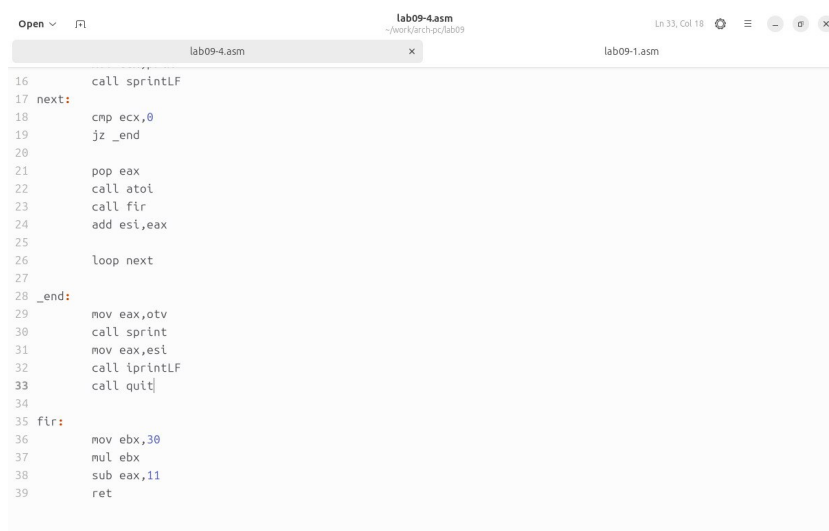
3.3 Задание для самостоятельной работы

Копирую файл из предыдущей лабораторной (рис. 3.30).

```
aatolstikhkh@aatolstikhkh:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-4.asm ~/work/arch-pc/lab09/lab09-4.asm
aatolstikhkh@aatolstikhkh:~/work/arch-pc/lab09$
```

Рис. 3.30: Создание копии программы

Преобразую программу, реализовав вычисление значения функции $f(x)$ как подпрограмму (рис. 3.31).



```
Open ▾  lab09-4.asm  lab09-1.asm
lab09-4.asm
Ln 33, Col 18

16      call sprintf
17 next:
18      cmp ecx,0
19      jz _end
20
21      pop eax
22      call atoi
23      call fir
24      add esi,eax
25
26      loop next
27
28 _end:
29      mov eax,otv
30      call sprintf
31      mov eax,esi
32      call fprintf
33      call quit
34
35 fir:
36      mov ebx,30
37      mul ebx
38      sub eax,11
39      ret
```

Рис. 3.31: Изменение программы

Создаю исполняемый файл и проверяю его работу (рис. 3.32).

```
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ ./lab09-4 1 2 3
f(x)=30x-11
Результат: 147
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ ./lab09-4 1
f(x)=30x-11
Результат: 19
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ ./lab09-4 2
f(x)=30x-11
Результат: 49
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ ./lab09-4 3
f(x)=30x-11
Результат: 79
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$
```

Рис. 3.32: Запуск программы

Создаю новый файл для дальнейшей работы с программой из листинга 9.3 (рис. 3.33).

```
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ touch lab09-5.asm
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$
```

Рис. 3.33: Создание файла

Ввожу в него текст программы (рис. 3.34).

```
Open  ▾  lab09-5.asm  Ln 4, Col 14
~/work/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 iprintf
20    call quit
```

Рис. 3.34: Написание программы

Получаю исполняемый файл. Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого трансляцию программ

необходимо проводить с ключом '-g'. Загружаю исполняемый файл в отладчик gdb и запускаю программу при помощи команды 'r' (рис. 3.35).

```
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
aatolstihkh@aatolstihkh:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-5...
(gdb) run
Starting program: /home/aatolstihkh/work/arch-pc/lab09/lab09-5

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
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 63047) exited normally]
(gdb) █
```

Рис. 3.35: Создание и загрузка исполняемого файла в отладчик

Результат программы неправильный, поэтому настраиваю отображение команд для дальнейшего изучения (рис. 3.36).

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
   0x080490e8 <+0>:    mov     ebx,0x3
   0x080490ed <+5>:    mov     eax,0x2
   0x080490f2 <+10>:   add     ebx,eax
   0x080490f4 <+12>:   mov     ecx,0x4
   0x080490f9 <+17>:   mul     ecx
   0x080490fb <+19>:   add     ebx,0x5
   0x080490fe <+22>:   mov     edi,ebx
   0x08049100 <+24>:   mov     eax,0x804a000
   0x08049105 <+29>:   call   0x0804900f <sprint>
   0x0804910a <+34>:   mov     eax,edi
   0x0804910c <+36>:   call   0x08049086 <iprintf>
   0x08049111 <+41>:   call   0x080490db <quit>
End of assembler dump.
(gdb) █
```

Рис. 3.36: Настройка отображения команд

Для более подробного анализа программы устанавливаю брейкпоинт на метку _start, с которой начинается выполнение любой ассемблерной программы, и запускаю её (рис. 3.37).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-5.asm, line 8.
(gdb) r
Starting program: /home/aatolstikh/work/arch-pc/lab09/lab09-5

Breakpoint 1, _start () at lab09-5.asm:8
8      mov ebx,3
(gdb)
```

Рис. 3.37: Запуск программы

Анализируя регистры, замечаю, что некоторые стоят не на своих местах (рис. 3.38).

```
Register group: general
eax    0x0      0      ecx    0x0      0
edx    0x0      0      ebx    0x0      0
esp    0xffffcf00 0xffffcf00  ebp    0x0      0x0
esi    0x0      0      edi    0x0      0
eip    0x80490e8 0x80490e8 <_start>  eflags 0x202     [ IF ]
cs     0x23     35     ss     0x2b     43
ds     0x2b     43     es     0x2b     43
fs     0x0      0      gs     0x0      0

0x80490e8 <_start>  mov     ebx,0x3
0x80490ed <_start+5> mov     eax,0x2
0x80490f2 <_start+10> add     ebx,eax
0x80490f4 <_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
```

Рис. 3.38: Анализ регистров

Открываю программу и изменяю места регистров на правильные (рис. 3.39).

```
Open  ▾  lab09-5.asm  Ln 14, Col 41  report.md
lab09-5.asm  x

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 eax,ebx          ; изменен порядок регистров
11    mov ecx,4
12    mul ecx
13    add eax,5            ; изменен регистр
14    mov edi,eax          ; изменен регистр
15 ; ---- Вывод результата на экран
16    mov eax,div
17    call sprint
18    mov eax,edi
19    call iprintLF
20    call quit
```

Рис. 3.39: Изменение регистров

После нахождения ошибки и изменения программы заново создаю исполняемый файл и запускаю его (рис. 3.40). Теперь программа работает корректно.

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

Рис. 3.40: Запуск программы

4 Выводы

В ходе выполнения лабораторной работы я приобрела навыки написания программ с использованием подпрограмм. А также познакомилась с методами отладки при помощи GDB и его основными возможностями.