

Отчет

Лабораторная работа №10

Щанкина Екатерина Викторовна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выполнение самостоятельной работы	18
5	Выводы	24
	Список литературы	25

Список иллюстраций

3.1	Каталог	7
3.2	Текст программы	8
3.3	Проверка программы	8
3.4	Текст программы	9
3.5	Проверка программы	10
3.6	Текст программы	11
3.7	Проверка программы	11
3.8	GDB	12
3.9	RUN	12
3.10	set disassembly-flavor intel	13
3.11	Псевдографика	13
3.12	info breakpoints	14
3.13	Точка останова	14
3.14	stepi	15
3.15	x	15
3.16	x	15
3.17	Изменение значения регистров	16
3.18	Изменение значения регистров	16
3.19	-args	17
3.20	стек	17
4.1	Копирование	18
4.2	Текст программы	19
4.3	Проверка	20
4.4	Проверка файла	20
4.5	Отладчик	21
4.6	Запуск кода	21
4.7	Поиск ошибки	22
4.8	Исправленный текст программы	23
4.9	Проверка программы	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

Приобрести навыки написания программ с использованием подпрограмм.

Ознакомиться с методами отладки при помощи GDB и его основными возможностями.

3 Выполнение лабораторной работы

1. Создаем каталог для выполнения лабораторной работы, переходим в него и создаем файл. (рис. 3.1)

```
evthankina@dk3n51 ~ $ mkdir ~/work/study/2022-2023/'Архитектура компьютера'/study_2022-2023_arc-pc/lab10
evthankina@dk3n51 ~ $ cd work
evthankina@dk3n51 ~/work $ cd study
evthankina@dk3n51 ~/work/study $ cd 2022-2023
evthankina@dk3n51 ~/work/study/2022-2023 $ cd 'Архитектура компьютера'
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера $ cd study_2022-2023_arc-pc
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arc-pc $ cd lab10
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arc-pc/lab10 $ touch lab10-1.asm
```

Рис. 3.1: Каталог

2. Рассмотрим программу вычисления арифметического выражения с помощью подпрограммы `_calcul`. (рис. 3.2) (рис. 3.3)


```
Открыть ▾  lab10-1.asm
~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
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 rez: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 mov eax, msg
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 call _calcul
20 mov eax, result
21 call sprint
22 mov eax, [rez]
23 call iprintLF
24 call quit
25
26 _calcul:
27 mov ebx, 2
28 mul ebx
29 add eax, 7
30 mov [rez], eax
31 ret
32
```

Рис. 3.2: Текст программы

```
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ nasm -f elf lab10-1.asm
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ./lab10-1
Введите x: 1
2x+7=9
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $
```

Рис. 3.3: Проверка программы

3. Изменила программу, добавила подпрограмму $g(x)=3x-1$ (рис. 3.4) (рис. 3.5)

Открыть ▾  lab10
~/work/study/2022-2023/Архитектура к

```
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 rez: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 mov eax, msg
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 call _calculator
20 call _calcul
21 mov eax, result
22 call sprint
23 mov eax, [rez]
24 call iprintLF
25 call quit
26
27 _calculator:
28 mov ebx, 3
29 mul ebx
30 sub eax, 1
31 mov [rez], eax
32 ret
33 _calcul:
34 mov ebx, 2
35 mul ebx
36 add eax, 7
37 mov [rez], eax
38 ret
```

Рис. 3.4: Текст программы

```

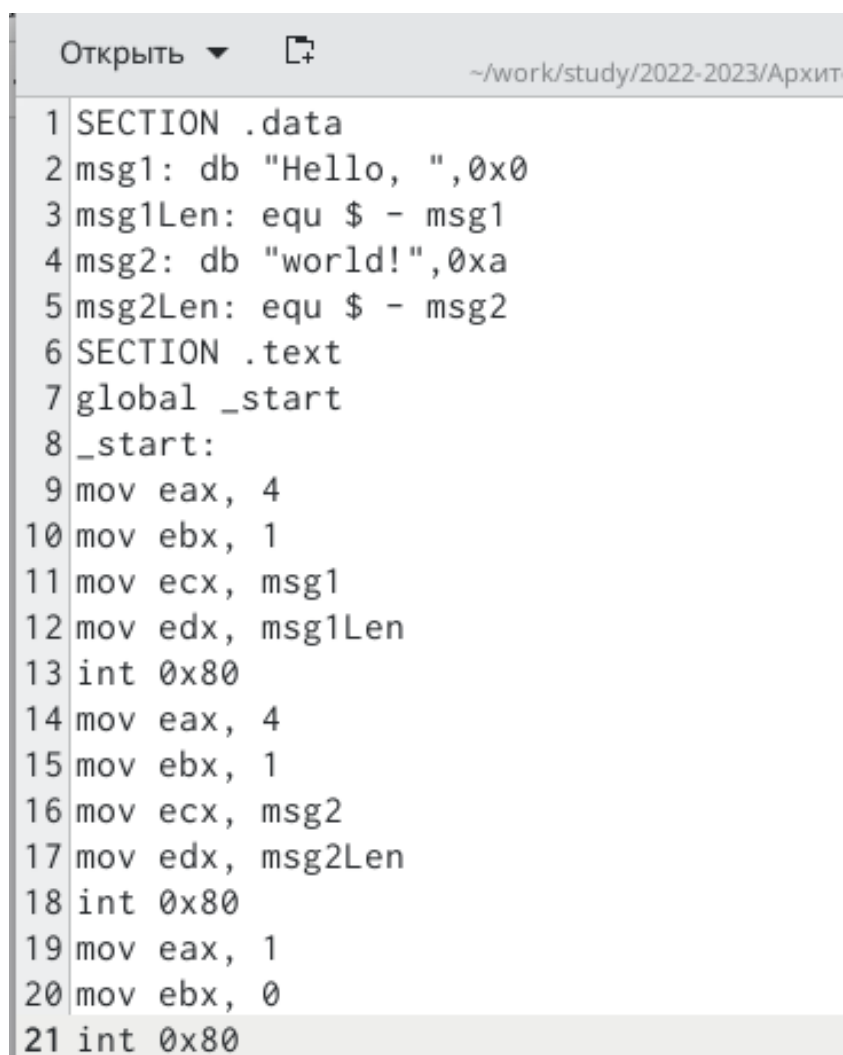
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ nasm -f elf lab10-1.asm
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ./lab10-1
Введите x: 1
2(3x-1)+7=11
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ 

```

Рис. 3.5: Проверка программы

Программа работает правильно, ведь если мы подставим $x=1$, то получим 11.

4. Создала файл lab10-2.asm для Листинга №2. Написала программу печати сообщения Hello world!. (рис. 3.6)



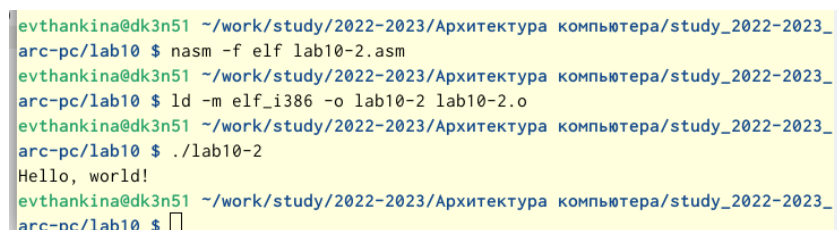
```

Открыть ▾  [icon] ~/work/study/2022-2023/Архит
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. 3.6: Текст программы

5. Проверила работу программы. (рис. 3.7)



```

evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ nasm -f elf lab10-2.asm
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ld -m elf_i386 -o lab10-2 lab10-2.o
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ./lab10-2
Hello, world!
evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ 

```

Рис. 3.7: Проверка программы

6. Загрузила исполняемый файл в отладчик. (рис. 3.8)

```

evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ gdb lab10-2
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab10-2...
(No debugging symbols found in lab10-2)
(gdb) 

```

Рис. 3.8: GDB

Проверим работу программы с помощью запуска в оболочке GDB с помощью команды 'run'. (рис. 3.9) Установила брейкпоинт на метку _start и запустила программу. (рис. 3.9) Посмотрела дисассимблированный код программы с помощью команды disassemble начиная с метки _start (рис. 3.9)

```

(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/v/evthankina/work/study/2022-20
23/Архитектура компьютера/study_2022-2023_arc-pc/lab10/lab10-2
Hello, world!
[Inferior 1 (process 3615) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/v/evthankina/work/study/2022-20
23/Архитектура компьютера/study_2022-2023_arc-pc/lab10/lab10-2

Breakpoint 1, 0x08049000 in _start ()
(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. 3.9: RUN

Переключилась на отображение команд с помощью команды set disassembly-flavor intel. (рис. 3.10)

```
(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. 3.10: set disassembly-flavor intel

Включила режим псевдографики.(рис. 3.11)

```
[ Register Values Unavailable ]

B+> 0x8049000 <_start>    mov     eax,0x4
      0x8049005 <_start+5>  mov     ebx,0x1
      0x804900a <_start+10> mov     ecx,0x804a000
      0x804900f <_start+15> mov     edx,0x8
      0x8049014 <_start+20> int     0x80
      0x8049016 <_start+22> mov     eax,0x4
      0x804901b <_start+27> mov     ebx,0x1
      0x8049020 <_start+32> mov     ecx,0x804a008
      0x8049025 <_start+37> mov     edx,0x7
      0x804902a <_start+42> int     0x80
      0x804902c <_start+44> mov     eax,0x1

native process 3635 In: _start      L??  PC: 0x8049000
(gdb) layout rags
Undefined layout command: "rags". Try "help layout".
(gdb) layout regs
(gdb) layout asm
(gdb) layout regs
(gdb) □
```

Рис. 3.11: Псевдографика

В режиме псевдографики у нас есть три окна: - в верхней части названия регистров и их текущие значения. - в средней части результат дисассимилирования программы - в нижней части мы можем вводить команды. Различия синтаксиса машинных команд в режимах АТТ и Intel заключаются в том, что в синтаксисе Intel не используются символы \$ и %, операнды меняются местами.

7. Проверим точки останова с помощью команды “info breakpoints”. (рис. 3.12)

```
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint      keep y   0x08049000  <_start>
breakpoint already hit 1 time
(gdb) □
```

Рис. 3.12: info breakpoints

Установила точку останова по адресу инструкции (ebx,0x0) (рис. 3.13)

```
(gdb) break *0x804931
Breakpoint 3 at 0x804931
```

Рис. 3.13: Точка останова

8. Выполнила 5 инструкций с помощью команды stepi и проследила значение регистров. (рис. 3.14) Мы можем увидеть, что изменяются значения регистров eax, ebx, ecx, edx.

```

Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffc5a0 0xffffc5a0

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

native process 3800 In: _start L18 PC: 0x8049016
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stwep
Undefined command: "stwep". Try "help".
(gdb) stepi
(gdb) stepi
(gdb)

```

Рис. 3.14: stepi

Команда 'x' позволяет посмотреть содержимое памяти. (рис. 3.15) (рис. 3.16)

```

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

```

Рис. 3.15: x

```

(gdb) x/1sb &msg2
0x804a008:      "world!\n"
(gdb)

```

Рис. 3.16: x

Воспользуемся командой set, которая помогает изменить значение для регистра и ячейки. (рис. 3.17)

```

(gdb) x/1sb &msg1
0x804a000:      "Hello, "
(gdb) set {char}&msg1='h'
(gdb) set {char}0x804a001='h'
(gdb) x/1sb &msg1
0x804a000:      "hhlllo, "
(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008:      "Lor d!\n"
(gdb) 

```

Рис. 3.17: Изменение значения регистров

С помощью команды 'set' изменила значение регистра ebx. (рис. 3.18)

```

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

```

Рис. 3.18: Изменение значения регистров

Различие в том, что код символа "2" - "110010" а это есть число 50.

9. Копируем файл из лабораторной работы №9 с новым именем и создаем исполняемый файл. Для загрузки в GDB программы с аргументами нужно использовать ключ -args. (рис. 3.19)

Чтобы исследовать расположение аргументов командной строки усталим точку останова и запустим команду. (рис. 3.19)


```

evthankina@dk3n51 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023
arc-pc/lab10 $ gdb --args lab10-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab10-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/v/evthankina/work/study/2022-2
3/Архитектура компьютера/study_2022-2023_arc-pc/lab10/lab10-3 аргумент1 аргуме
т 2 аргумент\ 3

Breakpoint 1, _start () at lab10-3.asm:5

```

Рис. 3.19: -args

Посмотрим позиции стека.(рис. 3.20)

```

(gdb) x/s *(void**)(esp + 4)
0xffffc72f:      "/afs/.dk.sci.pfu.edu.ru/home/e/v/evthankina/work/study/2022-20
3/Архитектура компьютера/study_2022-2023_arc-pc/lab10/lab10-3"
(gdb) x/s *(void**)(esp + 8)
0xffffc7c1:      "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc7d3:      "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc7e4:      "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc7e6:      "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:      <error: Cannot access memory at address 0x0>

```

Рис. 3.20: стек

В данном случае шаг изменения адреса равен 4, т.к. в теле цикла next 4 строки ввода.

4 Выполнение самостоятельной работы

1. Скопировала файл из 9 лабораторной работы с помощью команды `cp`. (рис. 4.1)

```
evthankina@dk8n52 ~ $ cp ~/work/study/2022-2023/'Архитектура компьютера'/study_2022-2023_arc-pc/lab09/lab9-1.asm ~/work/study/2022-2023/'Архитектура компьютера'/study_2022-2023_arc-pc/lab10/lab10-1.1.asm
evthankina@dk8n52 ~ $ cd work
evthankina@dk8n52 ~/work $ cd study
evthankina@dk8n52 ~/work/study $ cd 2022-2023
evthankina@dk8n52 ~/work/study/2022-2023 $ cd 'Архитектура компьютера'
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера $ cd study_2022-2023_arc-pc
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arc-pc $ cd lab10
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arc-pc/lab10 $ ls
in_out.asm lab10-1.1.asm lab10-1.o lab10-2.asm lab10-3 lab10-3.lst
lab10-1 lab10-1.asm lab10-2 lab10-2.o lab10-3.asm lab10-3.o
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arc-pc/lab10 $
```

Рис. 4.1: Копирование

- Изменила программу с использованием подпрограмм и запустила ее. (рис. 4.2) (рис. 4.3)

```
lab10-1.1.asm
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx
8
9 pop edx
10
11 sub ecx,1
12
13 mov esi, 0
14
15 next:
16 cmp ecx,0h
17 jz _end
18
19 pop eax
20 call atoi
21 call _calcul
22
23 loop next
24
25
26 _end:
27 mov eax, msg
28 call sprint
29 mov eax, esi
30 call iprintLF
31 call quit
32
33 _calcul:
34 mov eax,eax
35 mov ebx,4
36 mul ebx
37 add eax,3
38 add esi,eax
39
40 ret
```

Рис. 4.2: Текст программы

```

evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ nasm -f elf lab10-1.1.asm
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ld -m elf_i386 -o lab10-1.1 lab10-1.1.o
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ./lab10-1.1
Результат: 0
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ./lab10-1.1 1 2 3
Результат: 33
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ./lab10-1.1 1 2
Результат: 18
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ █

```

Рис. 4.3: Проверка

2. Написала программу из листинга и проверила с помощью отладчика. (рис. 4.4)

```

evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ touch lab10-1.2.asm
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ nasm -f elf lab10-1.2.asm
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ld -m elf_i386 -o lab10-1.2 lab10-1.2.o
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ./lab10-1.2
Результат: 10

```

Рис. 4.4: Проверка файла

Результат программы неверный, для устранения ошибки запускаем отладчик.(рис. 4.5) Далее установила точку останова. (рис. 4.5)

```

arc-pc/lab10 $ gdb lab10-1.2
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab10-1.2...
(No debugging symbols found in lab10-1.2)
(gdb) b _start
Breakpoint 1 at 0x080490e8
(gdb) i b
Num      Type             Disp Enb Address      What
1        breakpoint      keep y   0x080490e8  <_start>
(gdb)

```

Рис. 4.5: Отладчик

Запустила код программы с помощью команды run. (рис. 4.6)

```

(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/e/v/evthankina/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arc-pc/lab10/lab10-1.2

Breakpoint 1, 0x080490e8 in _start ()
(gdb)

```

Рис. 4.6: Запуск кода

Включила режим псевдографики и пошагово прошла все строчки.(рис. 4.7)

```
Register group: general
eax      0x804a001      134520833
ecx      0x4            4
edx      0x0            0
ebx      0x804a000      134520832
esp      0xffffc4d4     0xffffc4d4

> 0x8049009 <nextchar+6> jmp 0x8049003 <nextchar>
0x804900b <finished> sub %ebx,%eax
0x804900d <finished+2> pop %ebx
0x804900e <finished+3> ret
0x804900f <sprint> push %edx
0x8049010 <sprint+1> push %ecx
0x8049011 <sprint+2> push %ebx

native process 4141 In: nextchar L?? PC: 0x8049009
0x08049003 in nextchar ()
(gdb) si
0x08049006 in nextchar ()
(gdb) si
0x08049008 in nextchar ()
```

Рис. 4.7: Поиск ошибки

Мы можем увидеть, что регистр `eax` должен умножаться на 4, но умножался регистр `ebx`. А еще число 5 прибавлялось не к произведению, а только к `ebx`. Исправим ошибки. (рис. 4.8) (рис. 4.9)

```

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

```

Рис. 4.8: Исправленный текст программы

```

evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ nasm -f elf lab10-1.2.asm
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ld -m elf_i386 -o lab10-1.2 lab10-1.2.o
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ ./lab10-1.2
Результат: 25
evthankina@dk8n52 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_
arc-pc/lab10 $ 

```

Рис. 4.9: Проверка программы

Программа работает верно.

5 Выводы

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

Список литературы