

Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Жернаков Данила Иванович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Самостоятельная работа	14
4	Выводы	18
	Список литературы	19

Список иллюстраций

2.1	Создание каталога, файла	6
2.2	Работа программы 1	6
2.3	Работа программы	6
2.4	Исполняемый файл	7
2.5	Загрузка файла в отладчик gdb	7
2.6	Установка брейкпоинта и запуск программы	7
2.7	Дисассимилированный код программы	8
2.8	Переключение на отображение команд	8
2.9	Режим псевдографики	9
2.10	Точка останова	9
2.11	Адрес предпоследней инструкции	10
2.12	Значение переменной 1	10
2.13	Значение переменной 2	11
2.14	Изменение первого символа msg2	11
2.15	Вывод значений регистра edx	12
2.16	Копирование	12
2.17	Создание исполняемого файла	12
2.18	Загрузка в отладчик	13
2.19	Установка точки останова	13
2.20	Позиции стека	13
3.1	Преобразованная программа	15
3.2	Результат	16
3.3	Дисассимилированный код, значения регистров	16
3.4	Измененная программа	17
3.5	Результат	17

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

Создал каталог для выполнения лабораторной работы №9, перешел в него и создал файл (рис. 2.1).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm $ mkdir lab09
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm $ cd lab09
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ touch lab9-1.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ cp ~/Загрузки/in_out.asm in_out.asm
in_out.asm lab9-1.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ls
```

Рис. 2.1: Создание каталога, файла

Создал исполняемый файл и проверил его работу (рис. 2.2).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ nasm -f elf lab9-1.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ./lab9-1
Введите x: 5
2x+7=17
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $
```

Рис. 2.2: Работа программы 1

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

Создал исполняемый файл и проверил его работу (рис. 2.3).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ nasm -f elf lab9-1.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ld -m elf_i386 -o lab9-1 lab9-1.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ./lab9-1
Введите x: 5
Результат: 35
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $
```

Рис. 2.3: Работа программы

Создал файл lab09-2.asm с текстом программы из листинга 9.2. Получил исполняемый файл (рис. 2.4).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ nasm -f elf -g -l lab9-2.lst lab9-2.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ld -m elf_i386 -o lab9-2 lab9-2.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ls
in_out.asm lab9-1 lab9-1.asm lab9-1.o lab9-2 lab9-2.asm lab9-2.lst lab9-2.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $
```

Рис. 2.4: Исполняемый файл

Загрузил исполняемый файл в отладчик gdb (рис. 2.5).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ gdb lab9-2
GNU gdb (Gentoo 13.2 vanilla) 13.2
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-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 lab9-2...
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/1/dizhernakov/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09/lab9-2
Hello, world!
[Inferior 1 (process 12476) exited normally]
(gdb)
```

Рис. 2.5: Загрузка файла в отладчик gdb

Для более подробного анализа программы установил брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запустил её (рис. 2.6).

```
(gdb) break _start
Breakpoint 1 at 0x049000: file lab9-2.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/1/dizhernakov/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09/lab9-2
Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
```

Рис. 2.6: Установка брейкпоинта и запуск программы

Посмотрел дисассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start` (рис. 2.7).

```

(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. 2.7: Дисассимилированный код программы

Переключение на отображение команд с Intelовским синтаксисом (рис. 2.8).

```

(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. 2.8: Переключение на отображение команд

Включил режим псевдографики для более удобного анализа программы: layout asm и layout regs. (рис. ??).

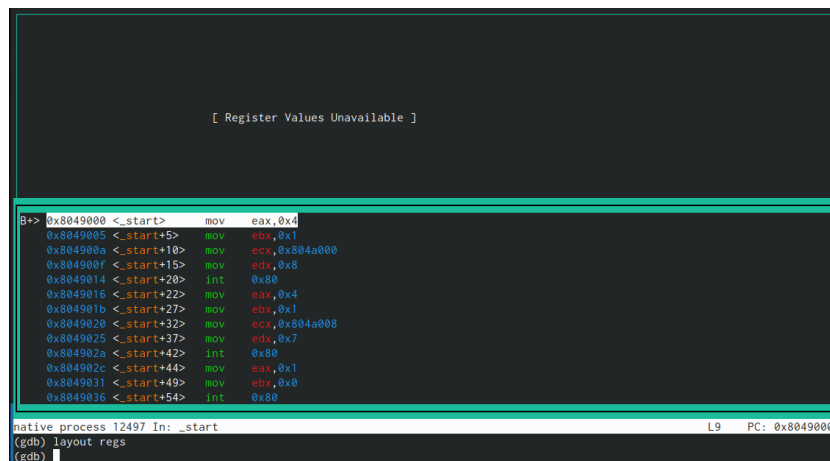


Рис. 2.9: Режим псевдографики

Проверил точку останова по имени метки (рис. 2.10).

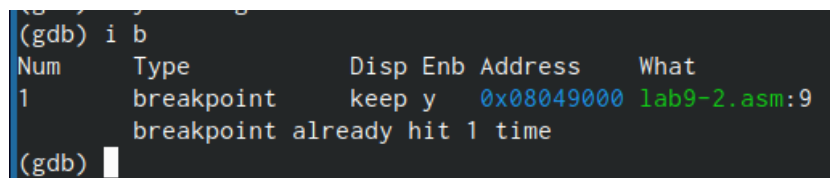


Рис. 2.10: Точка останова

Определила адрес предпоследней инструкции (mov ebx,0x0) и установила точку останова (рис. 2.11).

```

Register group: general
eax 0x8 8 ecx 0x804a000 134520832
edx 0x8 8 ebx 0x1 1
esp 0xfffffc00 0xfffffc00 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x8049016 0x8049016 <_start+22> eflags 0x202 [ IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

0+ 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

native process 12497 In: _start L14 PC: 0x8049016
(gdb) layout regs
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab9-2.asm:9
breakpoint already hit 1 time
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 2.11: Адрес предпоследней инструкции

Посмотрел значение переменной msg1 по имени (рис. 2.12).

```

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

```

Рис. 2.12: Значение переменной 1

Посмотрел значение переменной msg2 по адресу (рис. 2.13).

```

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

native process 12497 In: _start
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--qQuit
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb)

```

Рис. 2.13: Значение переменной 2

Изменил первый символ переменной msg1.

Заменял некоторый символ во второй переменной msg2 (рис. 2.14).

```

(gdb) x/1sb &msg1
0x804a000 <msg1>:      "Hello, "
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) ser {char}0x804a00b='G'
Undefined command: "ser". Try "help".
(gdb) set {char}0x804a00b='G'
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "worGd!\n\034"
(gdb)

```

Рис. 2.14: Изменение первого символа msg2

Вывел в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра eax (рис. 2.15).

```

$1 = 1.12103877e-44
(gdb) p/d &eax
No symbol "eax" in current context.
(gdb) p/d $eax
$2 = 8
(gdb) p/s $eax
$3 = 8
(gdb) 

```

Рис. 2.15: Вывод значений регистра edx

Завершил выполнение программы с помощью команды `continue` (сокращенно `c`) или `stepi` (сокращенно `si`) и вышла из GDB с помощью команды `quit` (сокращенно `q`).

Скопировал 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. 2.16).

```

dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ cp ~/work/study/2023-2024/Архитектура
a\ компьютера/arch-pc/labsasm/lab08/lab8-2.asm lab9-3.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ls
in_out.asm lab9-1 lab9-1.asm lab9-1.o lab9-2 lab9-2.asm lab9-2.lst lab9-2.o lab9-3.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ 

```

Рис. 2.16: Копирование

Создал исполняемый файл (рис. 2.17).

```

dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ cp ~/work/study/2023-2024/Архитектура
a\ компьютера/arch-pc/labsasm/lab08/lab8-2.asm lab9-3.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ls
in_out.asm lab9-1 lab9-1.asm lab9-1.o lab9-2 lab9-2.asm lab9-2.lst lab9-2.o lab9-3.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ nasm -f elf -g -l lab9-3.lst lab9-3.
asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ld -m elf_i386 -o lab9-3 lab9-3.o

```

Рис. 2.17: Создание исполняемого файла

Загрузил исполняемый файл в отладчик, указав аргументы (рис. 2.18).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ gdb --args lab9-3 1 2 '3'
GNU gdb (Gentoo 13.2 vanilla) 13.2
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-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 lab9-3...
(gdb)
```

Рис. 2.18: Загрузка в отладчик

Для начала устанавливаю точку останова перед первой инструкцией в программе и запускаю ее (рис. 2.19).

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/i/dizhernakov/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09/lab9-3 1 2 3
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb)
```

Рис. 2.19: Установка точки останова

Адрес вершины стека храниться в регистре esp, и по этому адресу располагается число, равное количеству аргументов командной строки (включая имя программы) Посмотрел остальные позиции стека (рис. 2.20).

```
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xfffffc0c: 0x00000004
(gdb) x/s *(void**)(esp + 4)
0xfffffc35: "/afs/.dk.sci.pfu.edu.ru/home/d/i/dizhernakov/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc3db: "1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc3dd: "2"
(gdb) x/s *(void**)(esp + 16)
0xfffffc3df: "3"
(gdb)
```

Рис. 2.20: Позиции стека

Шаг изменения адреса равен 4, потому что число аргументов равно 4.

3 Самостоятельная работа

Создал файл для самостоятельной работы Преобразовал программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $\Phi(x)$ как подпрограмму (рис. 3.1).

```

ab8-4-1.asm      [----] 10 L:[ 1+12 1.
1  %include 'in_out.asm'
2  SECTION .data
3  msg0 db "Функция: f(x)=30x-11"
4  msg1 db "Результат: ",0
5  SECTION .text
6  global _start
7  _start:
8  pop ecx
9  pop edx
10 sub ecx,1
11 mov esi, 0
12 next:
13 mov edx,30
14 cmp ecx,0h
15 jz _end
16 pop eax
17 call atoi
18 call _calc
19 add esi,eax
20 loop next
21 _end:
22 mov eax, msg0
23 call sprintf
24 mov eax, msg1
25 call sprintf
26 mov eax, esi
27 call sprintf
28 call quit
29
30 _calc:
31 mul edx
32 sub eax,11
33 ret

```

Рис. 3.1: Преобразованная программа

Получил верный ответ (рис. 3.2).

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab08 $ ./lab8-4-1 1 2 3
Функция: f(x)=30x-11Результат:
Результат:
147
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab08 $
```

Рис. 3.2: Результат

Создал файл для второго задания самостоятельной работы. Ввел программу из листинга 9.3. Попробовал запустить программу.

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ touch lab9-4
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ nasm -f elf _
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ld -m elf_i386 _
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ./lab9-4
Результат: 10
```

Просматрел дисассимилированный код программы, поставил точку останова перед инструкцией `_start` и открыл значения регистров на этапе первого сложения (рис. 3.3).

```
Register group: general
eax 0x2 2 ecx 0x0 0
edx 0x0 0 ebx 0x3 3
esp 0xffffc060 0xffffc060 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x80490f2 0x80490f2 <_start+10> eflags 0x202 [ IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

0x80490d9 <atoi.restore+3> pop %ebx
0x80490da <atoi.restore+4> ret
0x80490db <quit> mov $0x0,%ebx
0x80490de <quit+5> mov $0x1,%eax
0x80490e5 <quit+10> int $0x80
0x80490e7 <quit+12> ret
B+ 0x80490e8 <_start> mov $0x3,%ebx
0x80490ed <_start+5> mov $0x2,%ecx
> 0x80490f2 <_start+10> add %eax,%ebx
0x80490f4 <_start+12> mov $0x4,%ecx
0x80490f9 <_start+17> mul %ecx
0x80490fb <_start+19> add $0x5,%ebx
0x80490fe <_start+22> mov %ebx,%edi

native process 13576 In: _start L?? PC: 0x80490f2
(gdb) layout regs
(gdb) disassemble _start
(gdb) p/s $eax
No registers.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/d/i/dizhernakov/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/l
b09/lab9-4
Breakpoint 1, 0x80490e8 in _start ()
(gdb) si
0x80490ed in _start ()
(gdb) si
0x80490f2 in _start ()
(gdb)
```

Рис. 3.3: Дисассимилированный код, значения регистров

Регистр `ecx` со значением 4 умножается не на `ebx`, сложенным с `eax`, а только с `eax` со значением 2. Меняю (рис. 3.4).


```

lab9-4.asm      [----] 11 L: [ 1+13 14/ 20] *(233 / 34
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
11 mov ecx,4
12 mul ecx
13 add eax,5
14 mov edi,eax
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi
19 call iprintLF
20 call quit

```

Рис. 3.4: Измененная программа

Получаю верный ответ (рис. 3.5).

```

dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ nasm -f elf lab9-4.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ld -m elf_i386 -o lab9-4 lab9-4.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $ ./lab9-4
Результат: 25
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab09 $

```

Рис. 3.5: Результат

4 Выводы

В ходе работы я приобрел навыки написания программ с использованием подпрограмм, познакомился с методами отладки при помощи GDB и его основными возможностями.

Список литературы