

Отчёт по лабораторной работе №9

Простейший вариант

Бадалов Заури Эльвин оглы

Содержание

| | | |
|---|--------------------------------|----|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Выполнение лабораторной работы | 7 |
| 4 | Вывод | 15 |

Список иллюстраций

| | | |
|------|---|----|
| 3.1 | Программа $f(g(x))$ | 7 |
| 3.2 | Запуск программы | 7 |
| 3.3 | Программа с имитацией сложной функции | 8 |
| 3.4 | Запуск программы | 8 |
| 3.5 | Текст программы файла lab09-2.asm | 8 |
| 3.6 | Работа файла в отладчике gdb | 9 |
| 3.7 | Брейкпоинт на метку <code>_start</code> | 9 |
| 3.8 | Дисассимилированный код программы | 9 |
| 3.9 | Отображение команд с Intel'овским синтаксисом | 10 |
| 3.10 | Режим псевдографики | 10 |
| 3.11 | Значения регистров на псевдографике | 10 |
| 3.12 | Получение значения переменной по имени | 11 |
| 3.13 | Получение значения переменной по адресу | 11 |
| 3.14 | Замена значение переменной <code>msg1</code> | 11 |
| 3.15 | Замена символа переменной <code>msg2</code> | 11 |
| 3.16 | Вывод значения <code>edx</code> | 11 |
| 3.17 | Присвоение значения регистру | 12 |
| 3.18 | Запуск программы lab09-3.asm посредством gdb | 12 |
| 3.19 | Содержимое регистра <code>esp</code> | 13 |
| 3.20 | Остальные аргументы в стеке | 13 |
| 3.21 | Программа файла <code>zadan9</code> | 13 |
| 3.22 | Запуск программы | 14 |
| 3.23 | Исправленный текст программы | 14 |
| 3.24 | Запуск программы | 14 |

Список таблиц

1 Цель работы

Освоить работу с подпрограммами и отладчиком gdb.

2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

3 Выполнение лабораторной работы

Создаю рабочую директорию и файл. Записываю туда программу из листинга, исправив опечатки. (рис. 3.1).

```
lab09-1.asm [----] 11 L: [ 1+ 0 1/ 40] +(11 / 503b) 0110 0x06E
#include <stdio.h>
SECTION
msg: DB "Введите x: ",0
result: DB "2x + 7 = ",0

SECTION
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 ecx, x
call atoi

call _calcul

mov ecx, result
call sprint
mov ecx, [res]
call iprintLF

call quit

_calcul:
mov ebx, 2
mul ebx

add ebx, 7
mov [res],ebx

ret
```

Рис. 3.1: Программа $f(g(x))$

Проверяю работу программы. (рис. 3.2).

```
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 9
2x + 7 = 25
```

Рис. 3.2: Запуск программы

Вношу текст программы, имитирующей сложную функцию. Функции назову `_calcul` и `_subcalcul` (рис. 3.3).

```
_calcul:
push ebx
mov ebx, 2
mul ebx

add ebx, 7

pop ebx
ret

_subcalcul:
push ebx
mov ebx, 3
mul ebx
dec ebx
mov [res],ebx
```

Рис. 3.3: Программа с имитацией сложной функции

Проверяю работу программы. (рис. 3.4).

```
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 1
f(g(x)) = 3
zebadalov@dk4n68 ~/work/arch-pc/lab09 $
```

Рис. 3.4: Запуск программы

Создаю файл `lab09-2.asm` и ввожу в него текст программы. (рис. 3.5).

```
/afs/.dk.sci.pfu.edu.ru/home/z/e/zebadalov/work/arch-pc/lab09/lab09-2.asm
SECTION .data
msg1: db "Hello, ", 0x0
msg1Len: equ $ - msg1

msg2: db "world!", 0xa
msg2Len: equ $ - msg2

SECTION .text
global _start
_start:
mov eax, 4
mov ebx, 1
mov ecx, msg1
mov edx, msg1Len
int 0x80

mov eax, 4
mov ebx, 1
mov ecx, msg2
mov edx, msg2Len
int 0x80

mov eax, 1
mov ebx, 0
int 0x80
```

Рис. 3.5: Текст программы файла `lab09-2.asm`

Транслирую файл с ключом -g. После загружаю исполняемый файл в отладчик gdb. Проверяю его работу запустив ее в отладчике gdb командой run. (рис. 3.6).

```
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
zebadalov@dk4n68 ~/work/arch-pc/lab09 $ gdb lab09-2
GNU gdb (Gentoo 12.1 vanilla) 12.1
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 lab09-2...
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/z/e/zebadalov/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 11721) exited normally]
(gdb) █
```

Рис. 3.6: Работа файла в отладчике gdb

Устанавливаю брейкпоинт на метку _start и запускаю ассемблерную программу. (рис. 3.7).

```
[Inferior 1 (process 11721) exited normally]
(gdb) break _start
Breakpoint 1 at 0x00490000: file lab09-2.asm, line 11.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/z/e/zebadalov/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:11
11      mov eax, 4
(gdb) █
```

Рис. 3.7: Брейкпоинт на метку _start

Смотрю дисассимилированный код программы, использую команду disassemble, начиная с метки _start. (рис. 3.8).

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>: mov $0x4,%eax
0x08049005 <+5>: mov $0x1,%ebx
0x0804900a <+10>: mov $0x004a000,%ecx
0x0804900f <+15>: mov $0x0,%edx
0x08049014 <+20>: int $0x80
0x08049016 <+22>: mov $0x4,%eax
0x0804901b <+27>: mov $0x1,%ebx
0x08049020 <+32>: mov $0x004a000,%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. 3.8: Дисассимилированный код программы

Переключаюсь на отображение команд с Intel'овским синтаксисом. (рис. 3.9).

```

(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. 3.9: Отображение команд с Intel'овским синтаксисом

В представлении АТТ в виде 16-ричного числа записаны первые аргументы всех команд, а в представлении Intel так записываются адреса вторых аргументов. Включим режим псевдографики, при помощи которого отображается код программы и содержимое регистров. (рис. 3.10).

```

--Register group: general--
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffc2c0 0xffffc2c0  ebp      0x0      0
esi      0x0      0      edi      0x0      0
eip      0x8049000 0x8049000 <_start>  eflags  0x202    [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       0x0      0

--lab09-2.asm
B+> 11 mov eax, 4
12 mov ebx, 1
13 mov ecx, msg1
14 mov edx, msg1Len
b+ 15 int 0x80
16
17 mov eax, 4
18 mov ebx, 1
19 mov ecx, msg2

native process 11923 In: _start
Start it from the beginning? (y or n) yStarting program: /afs/dk.eci.pfu.edu.ru/home/z/ezebadalov/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:11
(gdb) b *0x8049014
Breakpoint 2 at 0x8049014: file lab09-2.asm, line 15.
(gdb) i b
Num   Type             Disp Enb Address      What
1     breakpoint       keep y  0x8049000 lab09-2.asm:11
      breakpoint already hit 1 time
2     breakpoint       keep y  0x8049014 lab09-2.asm:15
(gdb)

```

Рис. 3.10: Режим псевдографики

Вывожу значения регистров, используя команду i r. (рис. 3.11).

```

native process 11923 In: _start
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffc2c0 0xffffc2c0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 3.11: Значения регистров на псевдографике

В отладчике можно вывести текущие значения переменных. Сделать это можно, к примеру, по имени (рис. 3.12) или по адресу (рис. 3.13).

```
(gdb) x/1sb &msg1
0x004a000 <msg1>: "Hello, "
(gdb)
```

Рис. 3.12: Получение значения переменной по имени

```
(gdb) x/1sb 0x004a000
0x004a000 <msg1>: "Hello, "
(gdb)
```

Рис. 3.13: Получение значения переменной по адресу

Так же отладчик даёт менять значения переменных прямо во время выполнения программы. (рис. 3.14).

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x004a000 <msg1>: "hello, "
(gdb)
```

Рис. 3.14: Замена значение переменной msg1

Заменяю первый символ переменной msg2 на символ отступа. (рис. 3.15).

```
(gdb) set {char}&msg2=9
(gdb) x/1sb &msg2
0x004a000 <msg2>: "\torld!\n\034"
(gdb)
```

Рис. 3.15: Замена символа переменной msg2

Выведу значение регистра edx в разных форматах: строчном, 16-ричном, двоичном. (рис. 3.16).

```
0x004a000 <msg2>: "\torld!\n\034"
(gdb) p/s $edx
$1 = 0
(gdb) p/x
$2 = 0x0
(gdb) p/t
$3 = 0
(gdb)
```

Рис. 3.16: Вывод значения edx

Регистрам тоже можно задавать значения. (рис. 3.17).

```

native process 11923 In: _start
$1 = 0
(gdb) p/x
$2 = 0x0
(gdb) p/t
$3 = 0
(gdb) set $ebx="2"
evaluation of this expression requires the program to have a function "malloc".
(gdb) set $ebx=2
(gdb) p/s $ebx
$4 = 2
(gdb)

```

Рис. 3.17: Присвоение значения регистру

Однако при попытке задать строчное значение, происходит ошибка. Завершу работу в gdb командами continue, она закончит выполнение программы, и exit, она завершит сеанс gdb.

Скопирую файл из лабораторной работы №9, переименую и создаю исполняемый файл. Открываю отладчик и задаю аргументы. Создаю точку останова на метке _start и запускаю программу. (рис. 3.18).

```

zebadalov@dk8n75 ~ $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
nasm: fatal: unable to open input file 'lab09-3.asm' No such file or directory
zebadalov@dk8n75 ~ $ mc

zebadalov@dk8n75 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
zebadalov@dk8n75 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
zebadalov@dk8n75 ~/work/arch-pc/lab09 $ gdb --args lab09-3 arg1 arg 2 "arg 3"
GNU gdb (Gentoo 12.1 vanilla) 12.1
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 lab09-3...
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 12.
(gdb) run
Starting program: /afs/.dk.sci.pfu.edu.ru/home/z/e/zebadalov/work/arch-pc/lab09/lab09-3 arg1 arg 2 arg\ 3

Breakpoint 1, _start () at lab09-3.asm:12
12      mov eax, msg1
(gdb)

```

Рис. 3.18: Запуск программы lab09-3.asm посредством gdb

Смотрю на содержимое того, что расположено по адресу, находящемуся в регистре esp. (рис. 3.19).

```
(gdb) x/x $esp
0xfffffc2a0: 0x00000005
```

Рис. 3.19: Содержимое регистра esp

Далее смотрю на все остальные аргументы в стеке. Их адреса располагаются в 4 байтах друг от друга (именно столько занимает элемент стека). (рис. 3.20).

```
(gdb) x/s *(void**)(esp + 4)
0xfffffc548: "/afs/.dk.sci.pfu.edu.ru/home/z/e/zebada1ov/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xfffffc58c: "arg1"
(gdb) x/s *(void**)(esp + 8)
0xfffffc58c: "arg1"
(gdb) x/s *(void**)(esp + 12)
0xfffffc591: "arg"
(gdb) x/s *(void**)(esp + 16)
0xfffffc595: "2"
(gdb) x/s *(void**)(esp + 20)
0xfffffc597: "arg 3"
```

Рис. 3.20: Остальные аргументы в стеке

#Задания самостоятельной работы

Программа из лабораторной работы №8 с использованием подпрограмм. (рис. 3.21).

```
zadan9.asm [----] 16 L: [ 1+35 36/ 40] *(501 / 559b) 0010 0x00A
#include "inc_out.asm"

SECTION .data
f_x db "zadan9: 1010 - 11", 0h
msg db 10, 13, "program", 0h

SECTION .text
global _start
_start:
    push ebx
    dec ecx
    mov ebx, 10
    mul ebx
    pop ebx
    ret
_start:
    pop ecx
    pop ebx
    sub ecx, 1
    mov esi, 0

next:
    cmp ecx, 0h
    jz _end
    pop ecx
    call atoi
    call _f
    add esi, eax

loop_next

_end:
    mov eax, f_x
    call sprintf
    mov ebx, msg
    call sprintf
    mov ebx, esi
    call iprintf

call quit
```

Рис. 3.21: Программа файла zadan9

Проверяю результат выполнения работы программы. (рис. 3.22).

```
zebadalov@dk8n62 ~/work/arch-pc/lab09 $ nasm -f elf zadan9.asm
zebadalov@dk8n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o zadan9 zadan9.o
zebadalov@dk8n62 ~/work/arch-pc/lab09 $ ./zadan9
функция: 10(x - 1)
результат: 0
```

Рис. 3.22: Запуск программы

Редактирую текст программы из листинга 9.3 в новом файле zadan9-1.asm после просмотра регистров для поиска ошибки. (рис. 3.23).

```
zadan9-1.asm [----] 9 L: [ 1+ 6 7/ 20] *(110 / 303b) 0010 0x00A
#include 'in_out.asm'

SECTION .data
div: DB 'Результат: ', 0
SECTION .text
GLOBAL _start
_start:
    mov ebx, 3
    mov eax, 2
    add eax, ebx
    mov ecx, 4
    mul ecx
    add ecx, 5
    mov edi, eax
    mov eax, div
    call sprint
    mov eax, edi
    call iprintLF

    call quit
```

Рис. 3.23: Исправленный текст программы

Проверяю работу программы. (рис. 3.24).

```
zebadalov@dk8n62 ~/work/arch-pc/lab09 $ nasm -f elf zadan9-1.asm
zebadalov@dk8n62 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o zadan9-1 zadan9-1.o
zebadalov@dk8n62 ~/work/arch-pc/lab09 $ ./zadan9-1
Результат: 25
zebadalov@dk8n62 ~/work/arch-pc/lab09 $
```

Рис. 3.24: Запуск программы

4 Вывод

Я освоил работу с подпрограммами и отладчиком gdb.