

Отчёт по лабораторной работе №9

Простейший вариант

Янушкевич Михаил денисович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	22
4	Выводы	23
	Список литературы	24

Список иллюстраций

2.1	Создание файла	6
2.2	Ввод программы	7
2.3	Создание исполняемого файла, проверка программы	8
2.4	Изменение текста программы	9
2.5	Создание исполняемого файла, проверка программы	10
2.6	Ввод программы	11
2.7	Создание исполняемого файла, проверка программы	12
2.8	Установка брейкпоинта.	13
2.9	Просмотр дисассимилированного кода, переключение синтаксиса	14
2.10	Включение режима псевдографики	15
2.11	Проверка точки останова	16
2.12	Определение адреса, просмотр информации	16
2.13	Выполнение инструкций	17
2.14	Просмотр значения	17
2.15	Просмотр значения	18
2.16	Изменение первого символа	18
2.17	Изменение символа	18
2.18	Вывод регистра	19
2.19	Изменение значения регистра	19
2.20	Копирование файла, загрузка в GDB	20
2.21	Установка точки	20

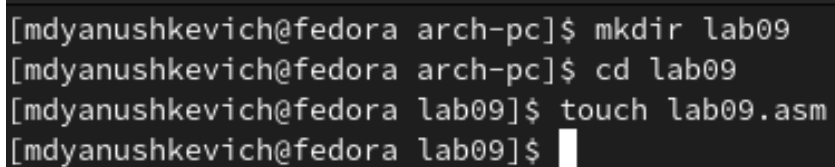
Список таблиц

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

1. Создать каталог для ЛБ9, в нём создать файл lab09-1.asm.(рис. 2.1)



```
[mdyanushkevich@fedora arch-pc]$ mkdir lab09  
[mdyanushkevich@fedora arch-pc]$ cd lab09  
[mdyanushkevich@fedora lab09]$ touch lab09.asm  
[mdyanushkevich@fedora lab09]$
```

Рис. 2.1: Создание файла

С помощью команды touch создаём файл lab09-1.asm.

2. В файл lab09-1.asm ввести программу из листинга 9.1.(рис.(2.2))

```

#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+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:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret
_subcalcul:
mov ebx, 3
mul ebx
dec eax
ret

```

Рис. 2.2: Ввод программы

Вводим в файл lab09-1.asm текст программы из листинга 9.1.

3. Создать исполняемый файл, проверить его работу.(рис. 2.3)

```
[mdyanushkevich@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.asm
[mdyanushkevich@fedora lab09]$ ./lab09-1
Введите x: 5
2x+7=17
[mdyanushkevich@fedora lab09]$
```

Рис. 2.3: Создание исполняемого файла, проверка программы

Вводим необходимые команды, чтобы создать исполняемый файл. Проверяем его работу.

4. Изменить текст программы(рис. 2.4)


```

#include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+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:

mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret

```

Рис. 2.4: Изменение текста программы

в файл lab09-1.asm вносим необходимые изменения, добавив подпрограмму _subcalcul в подпрограмму _calcul.

5. Создать исполняемый файл, проверить его работу.(рис. 2.5)

```
[mdyanushkevich@fedora lab09]$ nasm -f elf lab09-1.asm
[mdyanushkevich@fedora lab09]$ ld -m elf_i386 -o lab09-1 lab09-1.o
[mdyanushkevich@fedora lab09]$ ./lab09-1
Введите x: 2
2x+7=17
```

Рис. 2.5: Создание исполняемого файла, проверка программы

Вводим необходимые команды, чтобы создать исполняемый файл. Проверяем его работу. Число 17 является правильным ответом.

6. Создать файл lab09-2.asm, ввести в него текст программы листинга 9.2.(рис. 2.6)

```
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. 2.6: Ввод программы

Создаём файл lab09-2.asm, в него вводим код из листинга 9.2.

7. Получить исполняемый файл, проверить его работу.(рис. 2.7)

```
[mdyanushkevich@fedora lab09]$ gedit lab09-2.asm
[mdyanushkevich@fedora lab09]$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
[mdyanushkevich@fedora lab09]$ ld -m elf_i386 -o lab09-2 lab09-2.o
[mdyanushkevich@fedora lab09]$ gdb lab09-2
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
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-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 lab09-2...
(gdb) run
Starting program: /home/mdyanushkevich/work/arch-pc/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
Download failed: Нет маршрута до узла. Continuing without separate debug info
0xf7ffc000.
Hello, world!
[Inferior 1 (process 35039) exited normally]
(gdb)
```

Рис. 2.7: Создание исполняемого файла, проверка программы

Для работы с GDB в исполняемый файл необходимо добавить отладочную информацию, для этого вводим ключ -g. Загружаем файл в отладчик GDB, запускаем с помощью команды run.

8. Установить брейкпоинт, запустить программу.(рис. 2.8)

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 11.
(gdb) run
Starting program: /home/mdyanushkevich/work/arch-pc/lab09/lab09-2
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Download failed: Нет маршрута до узла. Continuing without separate debug
xf7ffc000.

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

```

Рис. 2.8: Установка брейкпоинта.

С помощью команды `break _start` устанавливаем брейкпоинт. С помощью команды `run` запускаем программу.

9. Посмотреть дисассимилированный код программы, переключиться на отображение команд с INTEL'овским синтаксисом.(рис. 2.9)

```

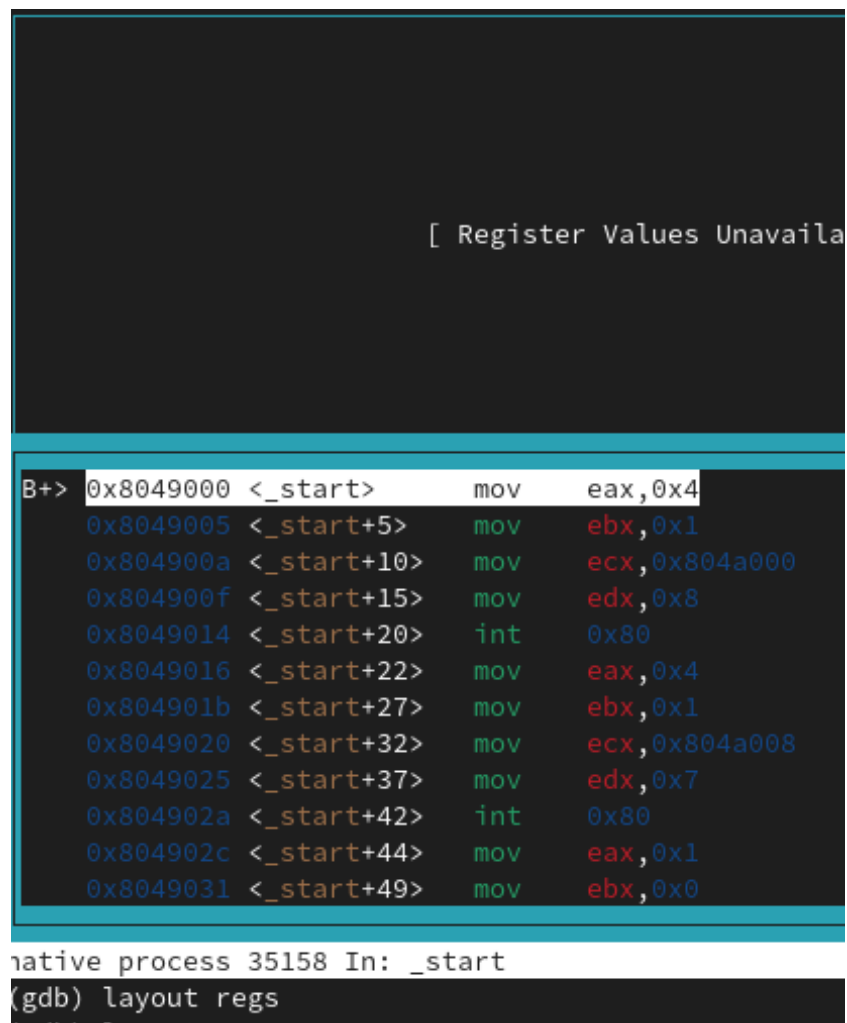
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     $0x4,%eax
    0x08049005 <+5>:      mov     $0x1,%ebx
    0x0804900a <+10>:     mov     $0x804a000,%ecx
    0x0804900f <+15>:     mov     $0x8,%edx
    0x08049014 <+20>:     int     $0x80
    0x08049016 <+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) set disassembly-flavor intel
(gdb) disassemble start
No symbol "start" in current context.
(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.

```

Рис. 2.9: Просмотр дисассимилированного кода, переключение синтаксиса

С помощью команды `disassemble` просматриваем дисассимилированный код. Далее переключаем синтаксис с помощью команды `set`.

10. Включить режим псевдографики.(рис. 2.10)



The screenshot shows a debugger window with a dark background. At the top, there is a section titled "[Register Values Unavailable" in a light blue font. Below this, a list of assembly instructions is displayed in a light blue font. The instructions are as follows:

Address	Disassembly
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
0x8049031 <_start+49>	mov ebx,0x0

Below the assembly list, the text "native process 35158 In: _start" is visible. At the bottom, the command "(gdb) layout regs" is entered in the command line.

Рис. 2.10: Включение режима псевдографики

Вводим необходимые команды для включения режима псевдографики.

11. Проверить точку останова.(рис. 2.11)

```
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:11
breakpoint already hit 1 time
(gdb) █
```

Рис. 2.11: Проверка точки останова

С помощью команды breakpoints проверяем точку останова.

12. Определить адрес предпоследней инструкции и установить точку останова, посмотреть информацию о всех установленных точках останова.(рис. 2.12)

```
(gdb) b *0x8049000
Note: breakpoint 1 also set at pc 0x8049000.
Breakpoint 2 at 0x8049000: file lab09-2.asm, line 11.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:11
breakpoint already hit 1 time
2        breakpoint     keep y   0x08049000 lab09-2.asm:11
(gdb)
```

Рис. 2.12: Определение адреса, просмотр информации

С помощью команды break устанавливаем точку останова, с помощью команды i b просматриваем информацию о других точках.

13. Выполнить 5 инструкций, проследить за изменением значений регистров.(рис. 2.13)


```

eax      0x8      8      ecx      0x804a000      134520832
edx      0x8      8      ebx      0x1      1
esp      0xffffd190      0xffffd190      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

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,0x804a000
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0x8049031 <_start+49> mov ebx,0x0

active process 35158 In: _start L16 PC: 0x8049016
(gdb) b *0x8049000
Note: breakpoint 1 also set at pc 0x8049000.
Breakpoint 2 at 0x8049000: file lab09-2.asm, line 11.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab09-2.asm:11
2 breakpoint already hit 1 time
3 breakpoint keep y 0x8049000 lab09-2.asm:11
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb)

```

Рис. 2.13: Выполнение инструкций

С помощью команды `si` просматриваем изменение значений регистров. Изменяются `eax`, `ebx`, `ecx`, `edx`.

14. Просмотреть значение переменной `msg1`. (рис. 2.14)

```

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

```

Рис. 2.14: Просмотр значения

Вводим необходимую команду, смотрим значение переменной.

15. Просмотреть значение переменной `msg2`. (рис. 2.16)

```

0x804a000 <msg1>:      "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) 

```

Рис. 2.15: Просмотр значения

Вводим необходимую команду, смотрим значение переменной.

16. Изменение первого символа переменной msg1.(рис. ??)

```

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

```

Рис. 2.16: Изменение первого символа

С помощью команды set изменяем символ 'H' на 'h'.

17. Изменение любого символа переменной msg2.(рис. 2.17)

```

(gdb) set {char}0x804a008='L'
(gdb) set {char}0x804a00b=' '
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "Lor d!\n\034"
(gdb) 

```

Рис. 2.17: Изменение символа

С помощью команды set изменяем несколько символов в переменной msg2.

18. Вывести регистр edx в различных форматах.(рис. 2.18)

```
(gdb) p/s $eax
$6 = 8
(gdb) p/t $edx
$7 = 1000
(gdb) p/x $edx
$8 = 0x8
(gdb) 
```

Рис. 2.18: Вывод регистра

Выводим значение регистра `edx` с шестнадцатеричным, двоичным и символьным форматах.

19. Изменение значения регистра `ebx`. (рис. 2.19)

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$9 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$10 = 2
(gdb) 
```

Рис. 2.19: Изменение значения регистра

С помощью команды set изменяем значение регистра ebx.

20. Скопировать файл lab8-2.asm, создать исполняемый файл, загрузить в GDB.(рис. 2.20)

```
mdyanushkevich@fedora lab09]$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
mdyanushkevich@fedora lab09]$ ld -m elf_i386 -o lab09-3 lab09-3.o
mdyanushkevich@fedora lab09]$ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 3'
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
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-redhat-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://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. 2.20: Копирование файла, загрузка в GDB

Копируем файл lab8-2.asm, создаём исполняемый файл, загружаем в GDB, используя ключ -args.

21. Установка точки останова, запуск инструкции.(рис. 2.21)

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab09-3.asm, line 7.
(gdb) run
Starting program: /home/mdyanushkevich/work/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.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Download failed: Нет маршрута до узла. Continuing without separate debug info for system-supplied DSO
.

Breakpoint 1, _start () at lab09-3.asm:7
7       pop ecx
(gdb)
```

Рис. 2.21: Установка точки

Устанавливаем точку останова, запускаем инструкцию с помощью команды run.

22. Посмотреть все позиции стека в регистре esp.(рис. ??)

Просмотр позиций стека

С помощью определенных команд просматриваем все позиции стека в регистре esp, находящиеся в памяти.

3 Задание для самостоятельной работы

1. Создать файл lab09-4.asm.(рис. ??)

Создание файла

С помощью команды touch создать файл lab09-4.asm.

2. Вводим в файл текст измененной программы(рис. ??)

Ввод программы

4 Выводы

Благодаря этой лабораторной работе я приобрёл навыки по использованию GDB, и с помощью неё смог ознакомиться с методами отладки программ. Также смог написать программы с использованием подпрограмм.

Список литературы