

Отчёт по лабораторной работе №9

Цатурьян Лев Вячеславович НММбд-03-23

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Задание для самостоятельной работы	14
4	Задание 1	15
5	Задание 2	17
6	Выводы	19

Список иллюстраций

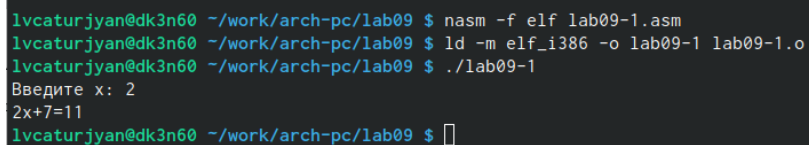
2.1	Запуск исполняемого файла	5
2.2	Изменённый текст программы	6
2.3	Создание объектного и исполняемого файлов. Запуск программы	7
2.4	Запуск отладчика	7
2.5	Установка брейкпоинта на метку_start	7
2.6	Дисассимилированный код программы с помощью команды disassemble с метки start	8
2.7	Программа с синтаксисом от Intel	8
2.8	Режим псевдографики	9
2.9	Точки останова	9
2.10	Установка точки останова по адресу и просмотр информации с помощью i b	9
2.11	5 команд si	10
2.12	Просмотр переменной msg1 по имени	10
2.13	Просмотр переменной msg2 по адресу	10
2.14	С помощью set меняем значение, выводим на экран	11
2.15	Теперь вместо w стоит W	11
2.16	Вывод значений регистра edx в разных форматах (в шестнадцате- ричном, в двоичном и в символьном виде)	11
2.17	Замена значения регистра ebx на '2' и вывод его на экран	12
2.18	Замена значения регистра ebx на 2 и вывод его на экран	12
2.19	Создание объектного и исполняемого файлов. Запуск программы с заданными аргументами	13
2.20	Просмотр позиций стека	13
4.1	Программа в mс	15
4.2	Создание объектного и исполняемого файлов. Запуск программы, работает корректно	16
5.1	Пошаговое выполнение программы в отладчике	17
5.2	Исправленный текст программы	18
5.3	Работа программы	18

1 Цель работы

Приобретение навыков использования подпрограмм. Работа с отладчиком

2 Выполнение лабораторной работы

Сначала я создал каталог для программ лабораторной работы № 9, перешёл в него и создал файл lab9-1.asm: Далее я ввёл в созданный файл текст листинга 9.1, создал объектный и исполняемый файлы



```
lvcaturjyan@dk3n60 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
lvcaturjyan@dk3n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
lvcaturjyan@dk3n60 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 2
2x+7=11
lvcaturjyan@dk3n60 ~/work/arch-pc/lab09 $
```

Рис. 2.1: Запуск исполняемого файла

Далее я изменил текст программы добавив подпрограмму subcalcul

Далее я создал объектный и исполняемый файлы и запустил программу

```

lab09-1.asm      [----] 13 L:[ 5+ 3
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; Основная программа
;-----
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprint
mov eax, [res]
call iprintLF
call quit
;-----
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы

_subcalcul:
mov ebx, 3
mul ebx
dec eax
mov [res], eax
ret

```

Рис. 2.2: Изменённый текст программы

```

lvcatjurjyan@dk3n60 ~/work/arch-pc/lab09 $ nasm -f elf lab09-1.asm
lvcatjurjyan@dk3n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-1 lab09-1.o
lvcatjurjyan@dk3n60 ~/work/arch-pc/lab09 $ ./lab09-1
Введите x: 3
f(g(x))=23

```

Рис. 2.3: Создание объектного и исполняемого файлов. Запуск программы

Все работает корректно

Далее я создал файл lab9-2.asm и ввёл в него текст из листинга 9.2 и транслировал этот файл с ключом -g

```

lvcatjurjyan@dk3n60 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-2.lst lab09-2.asm
lvcatjurjyan@dk3n60 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-2 lab09-2.o
lvcatjurjyan@dk3n60 ~/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) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/l/v/lvcatjurjyan/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 4567) exited normally]
(gdb)

```

Рис. 2.4: Запуск отладчика

Я запустил программу с помощью run

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) r
Starting program: /afs/.dk.sci.pfu.edu.ru/home/l/v/lvcatjurjyan/work/arch-pc/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb)

```

Рис. 2.5: Установка брейкпоинта на метку _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. 2.6: Дисассимилированный код программы с помощью команды disassemble с метки 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) set disassembly-flavor intel

```

Рис. 2.7: Программа с синтаксисом от Intel

Различия состоят в том, что в дисассимилированном отображении вместо

названия регистров пишутся их адреса

Затем я включил режим псевдографики

```
[ Register Values Unavailable ]

B+> 0x8049000 <_start> mov $0x4,%eax
0x8049005 <_start+5> mov $0x1,%ebx
0x804900a <_start+10> mov $0x804a000,%ecx
0x804900f <_start+15> mov $0x8,%edx
0x8049014 <_start+20> int $0x80
0x8049016 <_start+22> mov $0x4,%eax

native process 4580 In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb) █
```

Рис. 2.8: Режим псевдографики

Далее я проверил наличие точек останова в программе с помощью `info breakpoints`

```
(gdb) layout regs
(gdb) info breakpoints
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
(gdb) █
```

Рис. 2.9: Точки останова

```
(gdb) b *0x08049031
Breakpoint 2 at 0x8049031: file lab09-2.asm, line 20.
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab09-2.asm:9
          breakpoint already hit 1 time
2        breakpoint     keep y   0x08049031 lab09-2.asm:20
(gdb) █
```

Рис. 2.10: Установка точки останова по адресу и просмотр информации с помощью `i b`


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

Рис. 2.14: С помощью set меняем значение, выводим на экран

Такую же операцию я проделал с символом w в msg2

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

Рис. 2.15: Теперь вместо w стоит W

```
(gdb) p/s $edx
$3 = 8
(gdb) p/x $edx
$4 = 0x8
(gdb) p/t $edx
$5 = 1000
```

Рис. 2.16: Вывод значений регистра edx в разных форматах (в шестнадцатеричном, в двоичном и в символьном виде)

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

Рис. 2.17: Замена значения регистра ebx на '2' и вывод его на экран

```
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$6 = 2  
(gdb) 
```

Рис. 2.18: Замена значения регистра ebx на 2 и вывод его на экран

Разница в том, что в первом случае '2' - это символ, а 2 - это число

После этого я завершил выполнение программы с помощью с и вышел из отладчика с помощью quit

Далее я скопировал файл lab8-2.asm и назвал его lab9-3.asm, создал и загрузил исполняемый файл в отладчик, указав аргументы

```
lvcaturljyan@dk4n68 ~/work/arch-pc/lab09 $ nasm -f elf -g -l lab09-3.lst lab09-3.asm
lvcaturljyan@dk4n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o lab09-3 lab09-3.o
lvcaturljyan@dk4n68 ~/work/arch-pc/lab09 $ gdb --args lab09-3 аргумент1 аргумент 2 'аргумент 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) █
```

Рис. 2.19: Создание объектного и исполняемого файлов. Запуск программы с заданными аргументами

Далее я задал точку останова перед start и просмотрел позиции стека, в которых находятся аргументы

```
(gdb) x/s *(void**)($esp + 4)
0xfffffc550:  "/afs/.dk.sci.pfu.edu.ru/home/l/v/lvcaturljyan/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xfffffc598:  "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xfffffc5aa:  "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xfffffc5bb:  "2"
(gdb) x/s *(void**)($esp + 20)
0xfffffc5bd:  "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0:  <error: Cannot access memory at address 0x0>
(gdb) █
```

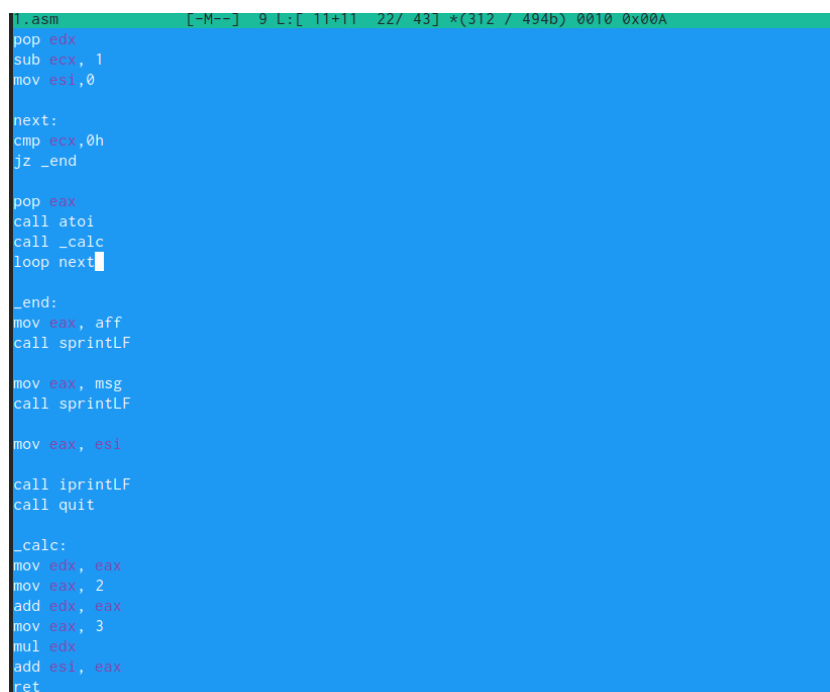
Рис. 2.20: Просмотр позиций стека

Значение меняется на 4, потому что именно на столько байт смещен каждый элемент относительно вершины стека

3 Задание для самостоятельной работы

4 Задание 1

Это измененная версия программы из лабораторной работы 8, теперь вычисление значения функции $y = 3(x+2)^2$ происходит как подпрограмма



```
l.asm [-M--] 9 L:[ 11+11 22/ 43] *(312 / 494b) 0010 0x00A
pop edx
sub ecx, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end

pop eax
call atoi
call _calc
loop next

_end:
mov eax, aff
call sprintf

mov eax, msg
call sprintf

mov eax, esi
call iprintf
call quit

_calc:
mov edx, eax
mov ecx, 2
add edx, ecx
mov ecx, 3
mul edx
add esi, ecx
ret
```

Рис. 4.1: Программа в мс

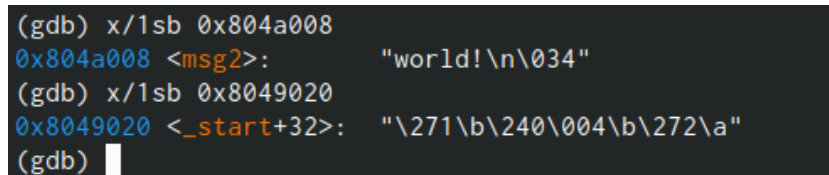
Её листинг:

```
%include 'in_out.asm' ; Цатурьян Лев НММбд-03-23 SECTION .data aff db "Вариант
7. Функция вида  $y = 3(x+2)^2$ ", 0 msg db "Результат:", 0 SECTION .text global _start _start:
pop ecx pop edx sub ecx, 1 mov esi, 0
next: cmp ecx, 0h jz _end
```

```

pop eax call atoi call _calc loop next
_end: mov eax, aff call sprintLF
mov eax, msg call sprintLF
mov eax, esi
call iprintLF call quit
_calc: mov edx, eax mov eax, 2 add edx, eax mov eax, 3 mul edx add esi, eax ret

```



```

(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) x/1sb 0x8049020
0x8049020 <_start+32>: "\271\b\240\004\b\272\a"
(gdb) █

```

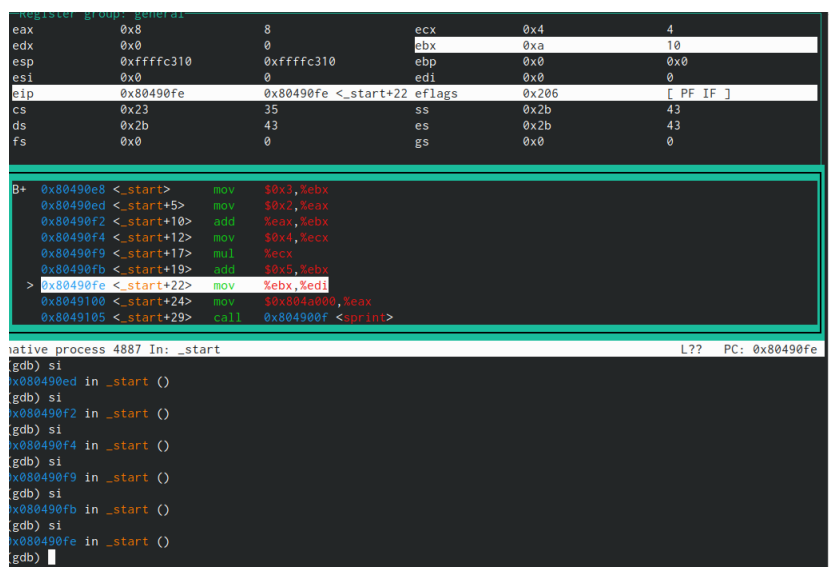
Рис. 4.2: Создание объектного и исполняемого файлов. Запуск программы, работает корректно

5 Задание 2

Я открыл программу с помощью отладчика и стал последовательно вводить команды si (выполнение программы по шагам)

Спустя несколько шагов значения, выдаваемые программой и значения вычислений из моей головы перестали совпадать

Я увидел, что результат вычисления выражения $(3+2)*4$ записан в регистрах, после чего к нему нужно прибавить 5, чтобы ответ получился верный, но программа добавляла 5 не к значению регистра eax, а к значению ebx, я исправил эту ошибку



```
Register window:
eax 0x8 8 ecx 0x4 4
edx 0x0 0 ebx 0xa 10
esp 0xfffffc310 0xfffffc310 ebp 0x0 0x0
esi 0x0 0 edi 0x0 0
eip 0x80490fe 0x80490fe <_start+22> eflags 0x206 [ PF IF ]
cs 0x23 35 ss 0x2b 43
ds 0x2b 43 es 0x2b 43
fs 0x0 0 gs 0x0 0

Disassembly window:
0x80490e8 <_start> mov $0x3,%ebx
0x80490ed <_start+5> mov $0x2,%eax
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
0x8049100 <_start+24> mov $0x804a000,%eax
0x8049105 <_start+29> call 0x804900f <sprintf>

Command window:
native process 4887 In: _start
(gdb) si
0x80490ed in _start ()
(gdb) si
0x80490f2 in _start ()
(gdb) si
0x80490f4 in _start ()
(gdb) si
0x80490f9 in _start ()
(gdb) si
0x80490fb in _start ()
(gdb) si
0x80490fe in _start ()
(gdb)
```

Рис. 5.1: Пошаговое выполнение программы в отладчике

```

2.asm [----] 10 L:[ 1+15 16/ 20] *(245
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov eax,3
mov ebx,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

```

Рис. 5.2: Исправленный текст программы

Текст программы: %include 'in_out.asm' SECTION .data div: DB 'Результат:',0 SECTION .text GLOBAL _start _start: ; -- Вычисление выражения (3+2)*4+5 mov eax,3 mov ebx,2 add eax,ebx mov ecx,4 mul ecx add eax,5 mov edi,eax mov eax,div call sprint mov eax,edi call iprintLF call quit

```

lvcaturjyan@dk4n68 ~/work/arch-pc/lab09 $ nasm -f elf 2.asm
lvcaturjyan@dk4n68 ~/work/arch-pc/lab09 $ ld -m elf_i386 -o 2 2.o
lvcaturjyan@dk4n68 ~/work/arch-pc/lab09 $ ./2
Результат: 25

```

Рис. 5.3: Работа программы

Программа работает корректно, результаты вычислений были проверены мной вручную

6 Выводы

Я приобрёл навыки использования подпрограмм и работы с отладчиком