

Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Абуков Ислам Ренатович

Цель работы

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

Задание

1. Реализация подпрограмм в NASM
2. Отладка программ с помощью GDB
3. Самостоятельное выполнение заданий по материалам лабораторной работы

Теоретическое введение

Отладка — это процесс поиска и исправления ошибок в программе. В общем случае его можно разделить на четыре этапа:

- обнаружение ошибки; • поиск её местонахождения; • определение причины ошибки; • исправление ошибки.

Можно выделить следующие типы ошибок:

- синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка; • семантические ошибки — являются логическими и приводят к тому, что программа запускается, отрабатывает, но не даёт желаемого результата; • ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки начнётся заново.

Выполнение лабораторной работы

Реализация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы №9

```
irabukov1@islamAbukov: ~/work/arch-pc/lab09
irabukov1@islamAbukov:~$ mkdir ~/work/arch-pc/lab09
irabukov1@islamAbukov:~$ cd ~/work/arch-pc/lab09
irabukov1@islamAbukov:~/work/arch-pc/lab09$ touch lab9-1.asm
irabukov1@islamAbukov:~/work/arch-pc/lab09$
```

Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции

```
irabukov1@islamAbukov:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ./lab9-1
bash: ./lab9-1: Нет такого файла или каталога
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2x+7=27
irabukov1@islamAbukov:~/work/arch-pc/lab09$
```

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения $f(g(x))$.

```
irabukov1@islamAbukov:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 10
2*(3x-1)+7=65
irabukov1@islamAbukov:~/work/arch-pc/lab09$
```

Отладка программ с помощью GDB

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компонирую и запускаю в отладчике

```
irabukov1@islamAbukov:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
irabukov1@islamAbukov:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-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 lab9-2...
(gdb)
```

Запустив программу командой `run`, я убедился в том, что она работает исправно

```
irabukov1@islamAbukov:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
irabukov1@islamAbukov:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-git
Copyright (C) 2024 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-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 lab9-2...
(gdb) run
Starting program: /home/irabukov1/work/arch-pc/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
    <https://debuginfod.ubuntu.com>
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
Hello, world!
[Inferior 1 (process 17669) exited normally]
(gdb)
```

Для более подробного анализа программы добавляю брейкпоинт на метку `_start` и снова запускаю отладку

```
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-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 lab9-2...
(gdb) run
Starting program: /home/irabukov1/work/arch-pc/lab09/lab9-2

This GDB supports auto-downloading debuginfo from the following URLs:
  <https://debuginfod.ubuntu.com>
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
Hello, world!
[Inferior 1 (process 17669) exited normally]
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/irabukov1/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb)
```

Далее смотрю дисассимилированный код программы, перевожу на команд с синтаксисом Intel *amd топчик*.

Различия между синтаксисом АТТ и Intel заключаются в порядке операндов (АТТ - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (АТТ - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как ax, eax, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(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
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)

```

Включаю режим псевдографики для более удобного анализа программы.

```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd030 0xffffd030
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]

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 19801 (asm) In: _start L9 PC: 0x8049000
(gdb) layout regs
(gdb)
```

Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился


```
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd030 0xffffd030
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]

0x80497b0  add    BYTE PTR [eax],al
0x80497b2  add    BYTE PTR [eax],al
0x80497b4  add    BYTE PTR [eax],al
0x80497b6  add    BYTE PTR [eax],al
0x80497b8  add    BYTE PTR [eax],al
0x80497ba  add    BYTE PTR [eax],al
0x80497bc  add    BYTE PTR [eax],al
0x80497be  add    BYTE PTR [eax],al
0x80497c0  add    BYTE PTR [eax],al
0x80497c2  add    BYTE PTR [eax],al
0x80497c4  add    BYTE PTR [eax],al

native process 19801 (asm) In: _start L9 PC: 0x8049000
(gdb) i b
Num      Type      Disp Enb Address  What
1        breakpoint keep y  0x08049000 lab9-2.asm:9
        breakpoint already hit 1 time
(gdb) b *0x08049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(gdb) i b
Num      Type      Disp Enb Address  What
1        breakpoint keep y  0x08049000 lab9-2.asm:9
        breakpoint already hit 1 time
2        breakpoint keep y  0x08049031 lab9-2.asm:20
(gdb)
```

Работа с данными программы в GDB

Просматриваю содержимое регистров командой `info registers` Просматриваю содержимое регистров командой `info registers`

```
Register group: general
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

0x80497b0  add  BYTE PTR [eax],al
0x80497b2  add  BYTE PTR [eax],al
0x80497b4  add  BYTE PTR [eax],al
0x80497b6  add  BYTE PTR [eax],al
0x80497b8  add  BYTE PTR [eax],al
0x80497ba  add  BYTE PTR [eax],al
0x80497bc  add  BYTE PTR [eax],al
0x80497be  add  BYTE PTR [eax],al
0x80497c0  add  BYTE PTR [eax],al
0x80497c2  add  BYTE PTR [eax],al
0x80497c4  add  BYTE PTR [eax],al

native process 19801 (asm) In: _start L9 PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd030 0xffffd030
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
Type <RET> for more, q to quit, c to continue without paging.
```

Смотрю содержимое переменных по имени и по адресу

```
irabukov1@islamAbukov: ~/work/arch-pc/lab09
mc [irabukov1@islamAbukov]:~/work/arch-... x irabukov1@islamAbukov: ~/work/arch-pc/la... x
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd030 0xffffd030
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>

0x80497b0  add  BYTE PTR [eax],al
0x80497b2  add  BYTE PTR [eax],al
0x80497b4  add  BYTE PTR [eax],al
0x80497b6  add  BYTE PTR [eax],al
0x80497b8  add  BYTE PTR [eax],al
0x80497ba  add  BYTE PTR [eax],al
0x80497bc  add  BYTE PTR [eax],al
0x80497be  add  BYTE PTR [eax],al
0x80497c0  add  BYTE PTR [eax],al

native process 19801 (asm) In: _start L9 PC: 0x8049000
--Type <RET> for more, q to quit, c to continue without paging--q
Quit
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) █
```

Меняю содержимое переменных по имени и по адресу

```
irabukov1@islamAbukov: ~/work/arch-pc/lab09
mc [irabukov1@islamAbukov]:~/work/arch-... x irabukov1@islamAbukov: ~/work/arch-pc/la... x
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd030 0xffffd030
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>

0x80497b4  add  BYTE PTR [eax],al
0x80497b6  add  BYTE PTR [eax],al
0x80497b8  add  BYTE PTR [eax],al
0x80497ba  add  BYTE PTR [eax],al
0x80497bc  add  BYTE PTR [eax],al
0x80497be  add  BYTE PTR [eax],al
0x80497c0  add  BYTE PTR [eax],al
0x80497c2  add  BYTE PTR [eax],al
0x80497c4  add  BYTE PTR [eax],al

native process 19801 (asm) In: _start L9 PC: 0x8049000
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='h'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "horld!\n\034"
(gdb) █
```

Вывожу в различных форматах значение регистра edx

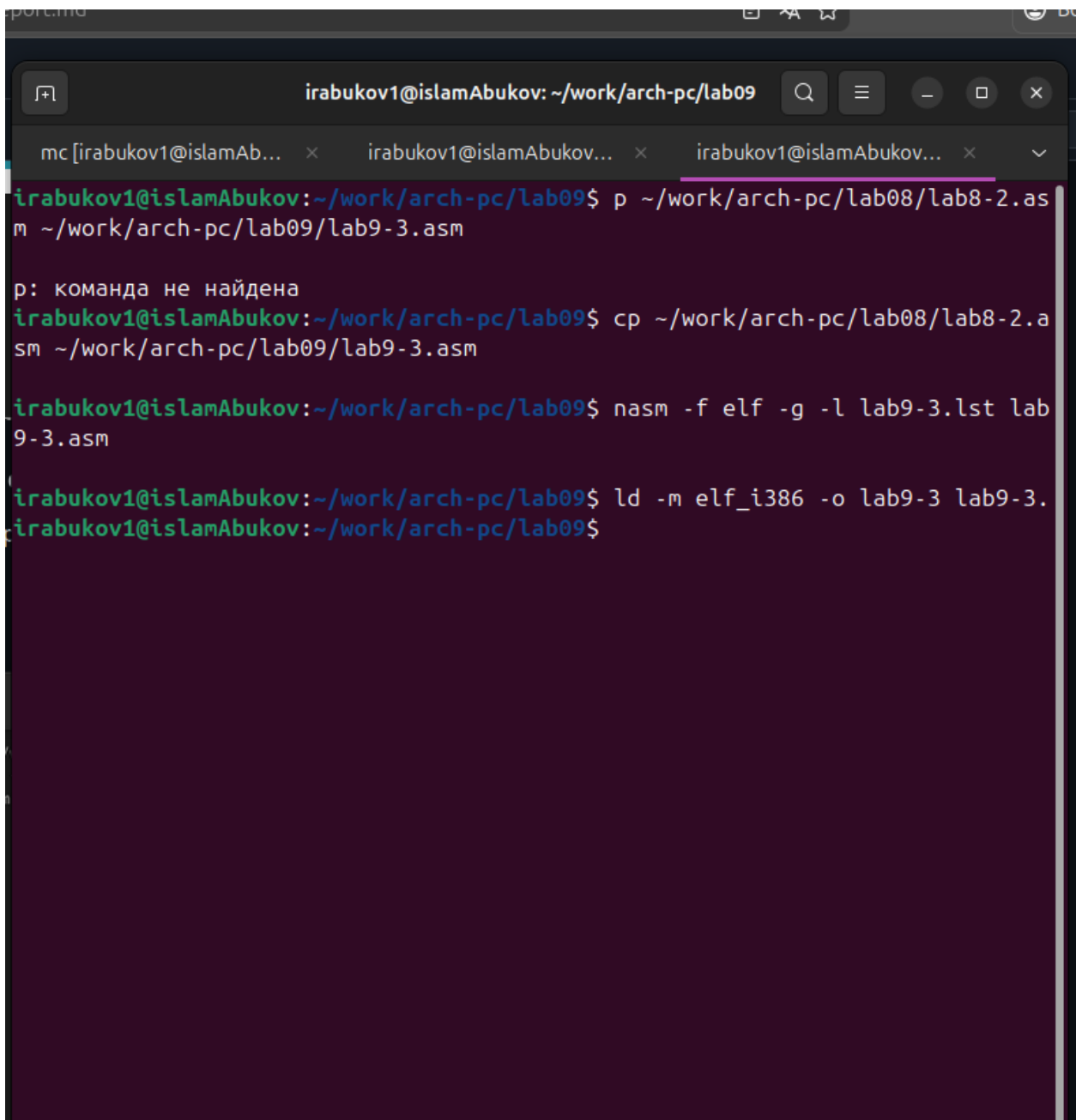
```
irabukov1@islamAbukov: ~/work/arch-pc/lab09
mc [irabukov1@islamAbukov]:~/work/arch-... x irabukov1@islamAbukov: ~/work/arch-pc/la... x
Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x2      2
esp      0xffffd030 0xffffd030
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>

0x80497b4  add  BYTE PTR [eax],al
0x80497b6  add  BYTE PTR [eax],al
0x80497b8  add  BYTE PTR [eax],al
0x80497ba  add  BYTE PTR [eax],al
0x80497bc  add  BYTE PTR [eax],al
0x80497be  add  BYTE PTR [eax],al
0x80497c0  add  BYTE PTR [eax],al
0x80497c2  add  BYTE PTR [eax],al
0x80497c4  add  BYTE PTR [eax],al

native process 19801 (asm) In: _start L9 PC: 0x8049000
($2 = 0
(gdb) p/t $edx
$3 = 0
(gdb) p/s $eax
$4 = 0
(gdb) p/x $ecx
$5 = 0x0
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb)
```

Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог и создаю исполняемый файл с файлом листинга и отладки

A screenshot of a terminal window with a dark background. The window title is 'irabukov1@islamAbukov: ~/work/arch-pc/lab09'. The terminal shows a series of commands and their outputs. The first command is 'p ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm', which results in an error message 'p: команда не найдена'. The second command is 'cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm'. The third command is 'nasm -f elf -g -l lab9-3.lst lab9-3.asm'. The fourth command is 'ld -m elf_i386 -o lab9-3 lab9-3.o'. The prompt returns after each command.

```
irabukov1@islamAbukov:~/work/arch-pc/lab09$ p ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm
p: команда не найдена
irabukov1@islamAbukov:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab9-3.asm
irabukov1@islamAbukov:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
irabukov1@islamAbukov:~/work/arch-pc/lab09$
```

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпоинт и запускаю отладку. Проверяю работу стека, изменяя аргумент команды просмотра регистра esp на +4, число обусловлено разрядностью системы, а указатель void занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились.

```
irabukov1@islamAbukov: ~/work/arch-pc/lab09
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) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/irabukov1/work/arch-pc/lab09/lab9-3 arg1 arg2 arg3

This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.ubuntu.com>
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

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx
(gdb) x/s *(void**)(esp + 4)
0xffffd1d5:    "/home/irabukov1/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd1ff:    "arg1"
(gdb) x/s *(void**)(esp + 12)
0xffffd204:    "arg2"
(gdb) x/s *(void**)(esp + 16)
0xffffd209:    "arg3"
(gdb) x/s *(void**)(esp + 20)
0x0:    <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)(esp + 24)
0xffffd20e:    "SHELL=/bin/bash"
(gdb)
```

Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы

```
GNU nano 7.2 /home/irabukov1/work/arch-pc/lab09/lab9-4.asm *
#include "in_out.asm"

SECTION .data
msg_func db "Функция: f(x) = 6x + 13", 0
msg_result db "Результат: ", 0

SECTION .bss
res: RESB 80 ; резервируем память для временного результата

SECTION .text
GLOBAL _start

_start:
; печать описания функции
mov eax, msg_func
call sprintLF

; ecx = количество аргументов (без имени программы)
xor ecx, ecx ; получаем argc
xor edx, edx ; получаем argv[0] (имя программы)
sub ecx, 1 ; уменьшаем счетчик на 1 (убираем имя программы)

cmp ecx, 0 ; проверяем, есть ли аргументы
jz no_args ; если нет аргументов, выходим

mov esi, 0 ; сумма значений f(x)

process_args:
xor eax, eax ; взять аргумент (строку)
call atoi ; перевести строку в число, результат в eax

; Вызов подпрограммы для вычисления f(x) = 6x + 13
call _calcul
```

2. Запускаю программу в режиме отладчика и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul ecx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программа неверно подсчитывает функцию


```

--Register group: general--
eax      0x2      2      ecx      0x4      4
edx      0x0      0      ebx      0x5      5
esp      0xffffcf10  0xffffcf10
esi      0x0      0      ebp      0x0      0x0
edi      0x0      0      edi      0x0      0
eip      0x80490f9  0x80490f9 <_start+17>
eflags   0x206    [ PF IF ]
cs       0x23     35     ss       0x2b     43
ds       0x2b     43     es       0x2b     43
fs       0x0      0      gs       0x0      0

B+ 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 <sprint>
0x804910a <_start+34>     mov     %edi,%eax

native process 8526 (asm) In: _start L14 PC: 0x80490f9
eax      0x2      2
ecx      0x4      4
edx      0x0      0
ebx      0x5      5
esp      0xffffcf10  0xffffcf10
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f9  0x80490f9 <_start+17>
eflags   0x206    [ PF IF ]
cs       0x23     35
--Type <RET> for more, q to quit, c to continue without paging--

```

Исправляю найденную ошибку, теперь программа верно считает значение функции

```
irabukov1@islamAbukov:~/work/arch-pc/lab09$ nasm -f elf lab9-5.asm
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ./lab5-1
bash: ./lab5-1: Нет такого файла или каталога
irabukov1@islamAbukov:~/work/arch-pc/lab09$ ./lab9-5
Результат: 25
irabukov1@islamAbukov:~/work/arch-pc/lab09$
```