

Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Козин Иван Евгеньевич

Содержание

1 Цель работы

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

2 Задание

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

3 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).

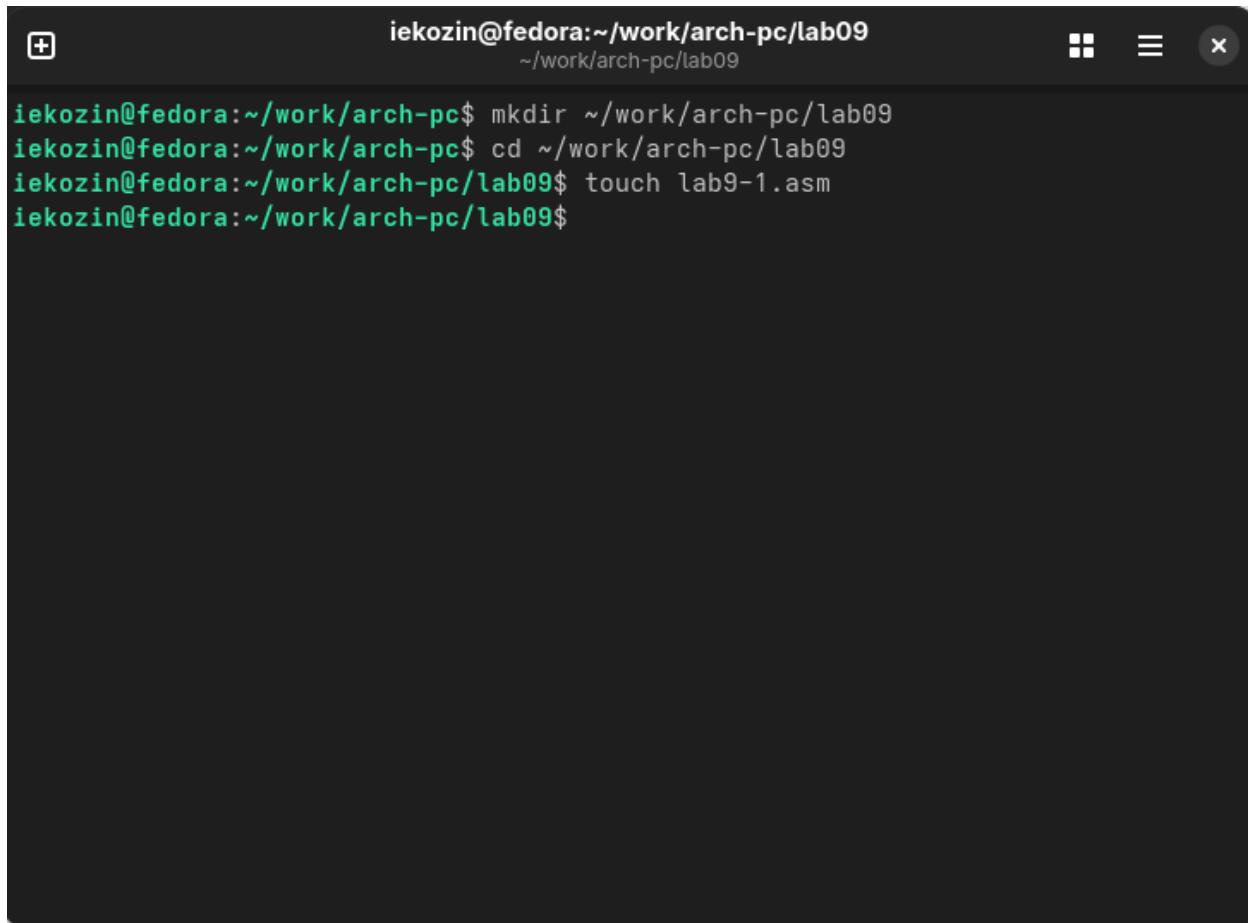
Второй этап — поиск местонахождения ошибки. Некоторые ошибки обнаружить довольно-таки трудно. Лучший способ найти место в программе, где находится ошибка, это разбить программу на части и произвести их отладку отдельно друг от друга.

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.

4 Выполнение лабораторной работы

4.1 Релазиация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы №9 (рис. 1).



A screenshot of a terminal window titled "iekozin@fedora:~/work/arch-pc/lab09". The window shows the command history:

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

Рис. 1: Создание рабочего каталога

Копирую в файл код из листинга, компилирую и запускаю его, данная программа выполняет вычисление функции (рис. 2).

```
[+] mc [iekozin@fedora]:~/work/arch-pc/lab09 — /usr/bin/mc -P /tmp/mc.p...     [x]
~/work/arch-pc/lab09

lab9-1.asm      [----]  0 L:[ 1+ 0  1/ 34] *(0    / 466b) 0037 0x025 [*][X]
%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 ; Вызов подпрограммы _calcul
        mov eax,result
        call sprint
1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9PullDown 10Quit
```

```
iekozin@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm
iekozin@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
iekozin@fedora:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 7
2x+7=21
iekozin@fedora:~/work/arch-pc/lab09$
```

Рис. 2: Запуск программы из листинга

Изменяю текст программы, добавив в нее подпрограмму, теперь она вычисляет значение функции для выражения $f(g(x))$ (рис. 3).

```
[+] mc [iekozin@fedora]:~/work/arch-pc/lab09 — /usr/bin/mc -P /tmp/mc.p... ~/work/arch-pc/lab09  
lab9-1.asm [-M--] 3 L:[ 1+ 0 1/ 49] *(3 / 476b) 0099 0x063 [*][X]  
%include 'in_out.asm'  
  
SECTION .data  
msg: DB 'Введите x: ', 0  
result: DB '2(3x-1)+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  
1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9PullDown 10Quit
```

```
iekozin@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-1.asm  
iekozin@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o  
iekozin@fedora:~/work/arch-pc/lab09$ ./lab9-1  
Введите x: 7  
2(3x-1)+7=47  
iekozin@fedora:~/work/arch-pc/lab09$
```

Рис. 3: Изменение программы первого листинга

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Ведите x: ', 0
result: DB '2(3x-1)+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:  
push eax  
call _subcalcul
```

```
mov ebx, 2  
mul ebx  
add eax, 7
```

```
mov [res], eax  
pop eax  
ret
```

```
_subcalcul:  
mov ebx, 3  
mul ebx  
sub eax, 1  
ret
```

4.1.1 Отладка программ с помощью GDB

В созданный файл копирую программу второго листинга, транслирую с созданием файла листинга и отладки, компоную и запускаю в отладчике (рис. 4).

```
[+ mc [iekozin@fedora]:~/work/arch-pc/lab09 — /usr/bin/mc -P /tmp/mc.p... ~/work/arch-pc/lab09  
lab9-2.asm [-M--] 8 L:[ 1+16 17/ 21] *(245 / 293b) 0032 0x020 [*][X]  
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  
  
1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

```
iekozin@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
iekozin@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
iekozin@fedora:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 16.3-1.fc42
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-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 lab9-2...
(gdb) 
```

Рис. 4: Запуск программы в отладчике

Запустив программу командой run, я убедился в том, что она работает исправно (рис. 5). (скриншот от запроса для удобочитаемости)

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

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.
Hello, world!
[Inferior 1 (process 30722) exited normally]
(gdb) 
```

Рис. 5: Проверка программы отладчиком

Для более подробного анализа программы добавляю брейкпойнт на метку _start и снова запускаю отладку (рис. 6).

```
(gdb) break _start
Breakpoint 3 at 0x8048080: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/iekozin/work/arch-pc/lab09/lab9-2

Breakpoint 3, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb)
```

Рис. 6: Запуск отладичка с брейкпоинтом

Далее смотрю дисассимилированный код программы, перевожу на команды с синтаксисом Intel амд топчик (рис. 7).

Различия между синтаксисом ATT и Intel заключаются в порядке операндов (ATT - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (ATT - 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(ATT - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov    $0x4,%eax
  0x08048085 <+5>:    mov    $0x1,%ebx
  0x0804808a <+10>:   mov    $0x8049000,%ecx
  0x0804808f <+15>:   mov    $0x8,%edx
  0x08048094 <+20>:   int    $0x80
  0x08048096 <+22>:   mov    $0x4,%eax
  0x0804809b <+27>:   mov    $0x1,%ebx
  0x080480a0 <+32>:   mov    $0x8049008,%ecx
  0x080480a5 <+37>:   mov    $0x7,%edx
  0x080480aa <+42>:   int    $0x80
  0x080480ac <+44>:   mov    $0x1,%eax
  0x080480b1 <+49>:   mov    $0x0,%ebx
  0x080480b6 <+54>:   int    $0x80
End of assembler dump.
(gdb)
```

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08048080 <+0>:    mov    eax,0x4
  0x08048085 <+5>:    mov    ebx,0x1
  0x0804808a <+10>:   mov    ecx,0x8049000
  0x0804808f <+15>:   mov    edx,0x8
  0x08048094 <+20>:   int    0x80
  0x08048096 <+22>:   mov    eax,0x4
  0x0804809b <+27>:   mov    ebx,0x1
  0x080480a0 <+32>:   mov    ecx,0x8049008
  0x080480a5 <+37>:   mov    edx,0x7
  0x080480aa <+42>:   int    0x80
  0x080480ac <+44>:   mov    eax,0x1
  0x080480b1 <+49>:   mov    ebx,0x0
  0x080480b6 <+54>:   int    0x80
End of assembler dump.
(gdb)
```

Рис. 7: Дисассимилирование программы

Включаю режим псевдографики для более удобного анализа программы (рис. 8).

The screenshot shows the GDB interface running on a Fedora system. The title bar reads "iekozin@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The command line shows the current file is "/work/arch-pc/lab09". The "Registers" tab is selected, displaying the general register values:

| Register | Value | Description |
|----------|-------------|-------------|
| eax | 0x0 | 0 |
| ecx | 0x0 | 0 |
| edx | 0x0 | 0 |
| ebx | 0x0 | 0 |
| esp | 0xfffffcf70 | 0xfffffcf70 |
| ebp | 0x0 | 0x0 |

The assembly code window shows the following instructions:

```
B+>0x8048080 <_start>    mov    eax,0x4
0x8048085 <_start+5>      mov    ebx,0x1
0x804808a <_start+10>     mov    ecx,0x8049000
0x804808f <_start+15>     mov    edx,0x8
0x8048094 <_start+20>     int    0x80
0x8048096 <_start+22>     mov    eax,0x4
```

The status bar at the bottom indicates the native process ID is 31077, the assembly language is selected, the instruction pointer (IP) is at address 0x8048080, and the program counter (PC) is also at 0x8048080.

Рис. 8: Режим псевдографики

4.1.2 Добавление точек останова

Проверяю в режиме псевдографики, что брейкпоинт сохранился (рис. 9).

```
mazurskiy@vbox:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Register group: general
eax          0x0          0          ecx          0x0          0
edx          0x0          0          ebx          0x0          0
esp 0xfffffcf10 0xfffffcf10          ebp          0x0          0x0
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

0x80496f2 add  BYTE PTR [eax],al
0x80496f4 add  BYTE PTR [eax],al
0x80496f6 add  BYTE PTR [eax],al
0x80496f8 add  BYTE PTR [eax],al
0x80496fa add  BYTE PTR [eax],al
0x80496fc add  BYTE PTR [eax],al
0x80496fe add  BYTE PTR [eax],al
0x8049700 add  BYTE PTR [eax],al
0x8049702 add  BYTE PTR [eax],al
0x8049704 add  BYTE PTR [eax],al

native process 5886 (asm) In: _start
breakpoint already hit 1 time
L11 PC: 0x8049000
(gdb) layout asm
(gdb) layout regs
(gdb) layout regs
(gdb) break *0x08049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 24.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x08049000 lab9-2.asm:11
breakpoint already hit 1 time
2 breakpoint keep y 0x08049031 lab9-2.asm:24
(gdb)
```

```
iekozin@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

Register group: general
eax          0x0          0
ecx          0x0          0
edx          0x0          0
ebx          0x0          0
esp 0xfffffcf70 0xfffffcf70
ebp          0x0          0x0

B+>0x8048080 <_start>    mov    eax,0x4
0x8048085 <_start+5>    mov    ebx,0x1
0x804808a <_start+10>   mov    ecx,0x8049000
0x804808f <_start+15>   mov    edx,0x8
0x8048094 <_start+20>   int    0x80
0x8048096 <_start+22>   mov    eax,0x4

native process 31077 (asm) In: _start
(gdb) layout regs
(gdb) i b
Num Type Disp Enb Address What
3 breakpoint keep y 0x08048080 lab9-2.asm:9
breakpoint already hit 1 time
(gdb)
```

Рис. 9: Список брейкпоинтов

Устанавливаю еще одну точку останова по адресу инструкции (рис. 10).

The screenshot shows a terminal window titled "iekozin@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The assembly code listed is:

```
b+>0x8048080 <_start>    mov    eax,0x4
 0x8048085 <_start+5>    mov    ebx,0x1
 0x804808a <_start+10>   mov    ecx,0x8049000
 0x804808f <_start+15>   mov    edx,0x8
 0x8048094 <_start+20>   int    0x80
 0x8048096 <_start+22>   mov    eax,0x4
 0x804809b <_start+27>   mov    ebx,0x1
 0x80480a0 <_start+32>   mov    ecx,0x8049008
b+ 0x80480a5 <_start+37>   mov    edx,0x7
 0x80480aa <_start+42>   int    0x80
 0x80480ac <_start+44>   mov    eax,0x1
 0x80480b1 <_start+49>   mov    ebx,0x0
 0x80480b6 <_start+54>   int    0x80
```

The GDB command history at the bottom shows:

```
native process 31077 (asm) In: _start
(gdb) layout asm
(gdb) layout src
(gdb) layout asm
(gdb) b _start
Breakpoint 4 at 0x8048080: file lab9-2.asm, line 9.
(gdb) b *0x80480a5
Breakpoint 5 at 0x80480a5: file lab9-2.asm, line 17.
(gdb) 
```

Рис. 10: Добавление второй точки останова

4.1.3 Работа с данными программы в GDB

Просматриваю содержимое регистров командой info registers (рис. 11).

```
ikedozin@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

b+>0x8048080 <_start>    mov    eax,0x4
    0x8048085 <_start+5>    mov    ebx,0x1
    0x804808a <_start+10>   mov    ecx,0x8049000
    0x804808f <_start+15>   mov    edx,0x8
    0x8048094 <_start+20>   int    0x80
    0x8048096 <_start+22>   mov    eax,0x4
    0x804809b <_start+27>   mov    ebx,0x1
    0x80480a0 <_start+32>   mov    ecx,0x8049008
b+ 0x80480a5 <_start+37>   mov    edx,0x7
    0x80480aa <_start+42>   int    0x80
    0x80480ac <_start+44>   mov    eax,0x1
    0x80480b1 <_start+49>   mov    ebx,0x0
    0x80480b6 <_start+54>   int    0x80

native process 31077 (asm) In: _start          L9      PC: 0x8048080
eax            0x0              0
ecx            0x0              0
edx            0x0              0
ebx            0x0              0
esp            0xfffffcf70        0xfffffcf70
ebp            0x0              0
esi            0x0              0
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 11: Просмотр содержимого регистров

Смотрю содержимое переменных по имени и по адресу (рис. 12).

The screenshot shows a terminal window with the following content:

```
iekozin@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

b+>0x8048080 <_start>      mov    eax,0x4
0x8048085 <_start+5>        mov    ebx,0x1
0x804808a <_start+10>       mov    ecx,0x8049000
0x804808f <_start+15>       mov    edx,0x8
0x8048094 <_start+20>       int    0x80
0x8048096 <_start+22>       mov    eax,0x4
0x804809b <_start+27>       mov    ebx,0x1
0x80480a0 <_start+32>       mov    ecx,0x8049008
b+ 0x80480a5 <_start+37>     mov    edx,0x7
0x80480aa <_start+42>       int    0x80
0x80480ac <_start+44>       mov    eax,0x1
0x80480b1 <_start+49>       mov    ebx,0x0
0x80480b6 <_start+54>       int    0x80

native process 31077 (asm) In: _start
(gdb) layout asm
(gdb) x/1sb &masg1
No symbol "masg1" in current context.
(gdb) x/1sb &msg1
0x8049000 <msg1>:      "Hello, "
(gdb) x/1sb 0x8049008
0x8049008 <msg2>:      "world!\n\034"
(gdb)
```

Рис. 12: Просмотр содержимого переменных двумя способами

Меняю содержимое переменных по имени и по адресу (рис. 13).

```
iekozin@fedora:~/work/arch-pc/lab09 — gdb lab9-2
~/work/arch-pc/lab09

b+>0x8048080 <_start>    mov    eax,0x4
0x8048085 <_start+5>    mov    ebx,0x1
0x804808a <_start+10>   mov    ecx,0x8049000
0x804808f <_start+15>   mov    edx,0x8
0x8048094 <_start+20>   int    0x80
0x8048096 <_start+22>   mov    eax,0x4
0x804809b <_start+27>   mov    ebx,0x1
0x80480a0 <_start+32>   mov    ecx,0x8049008
b+ 0x80480a5 <_start+37>  mov    edx,0x7
0x80480aa <_start+42>   int    0x80
0x80480ac <_start+44>   mov    eax,0x1
0x80480b1 <_start+49>   mov    ebx,0x0
0x80480b6 <_start+54>   int    0x80

native process 31077 (asm) In: _start          L9      PC: 0x8048080
0x8049008 <msg2>:      "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x8049000 <msg1>:      "hello, "
(gdb) set {char}&msg2='x'
(gdb) x/1sb &msg2
0x8049008 <msg2>:      "xorld!\n\034"
(gdb)
```

Рис. 13: Изменение содержимого переменных двумя способами

Вывожу в различных форматах значение регистра edx (рис. 14).

The screenshot shows a terminal window titled "iekozin@fedora:~/work/arch-pc/lab09 — gdb lab9-2". The assembly code listing shows the initial stack frame setup:

```
b+>0x8048080 <_start>    mov    eax,0x4
0x8048085 <_start+5>    mov    ebx,0x1
0x804808a <_start+10>   mov    ecx,0x8049000
0x804808f <_start+15>   mov    edx,0x8
0x8048094 <_start+20>   int    0x80
0x8048096 <_start+22>   mov    eax,0x4
0x804809b <_start+27>   mov    ebx,0x1
0x80480a0 <_start+32>   mov    ecx,0x8049008
b+ 0x80480a5 <_start+37>  mov    edx,0x7
0x80480aa <_start+42>   int    0x80
0x80480ac <_start+44>   mov    eax,0x1
0x80480b1 <_start+49>   mov    ebx,0x0
0x80480b6 <_start+54>   int    0x80
```

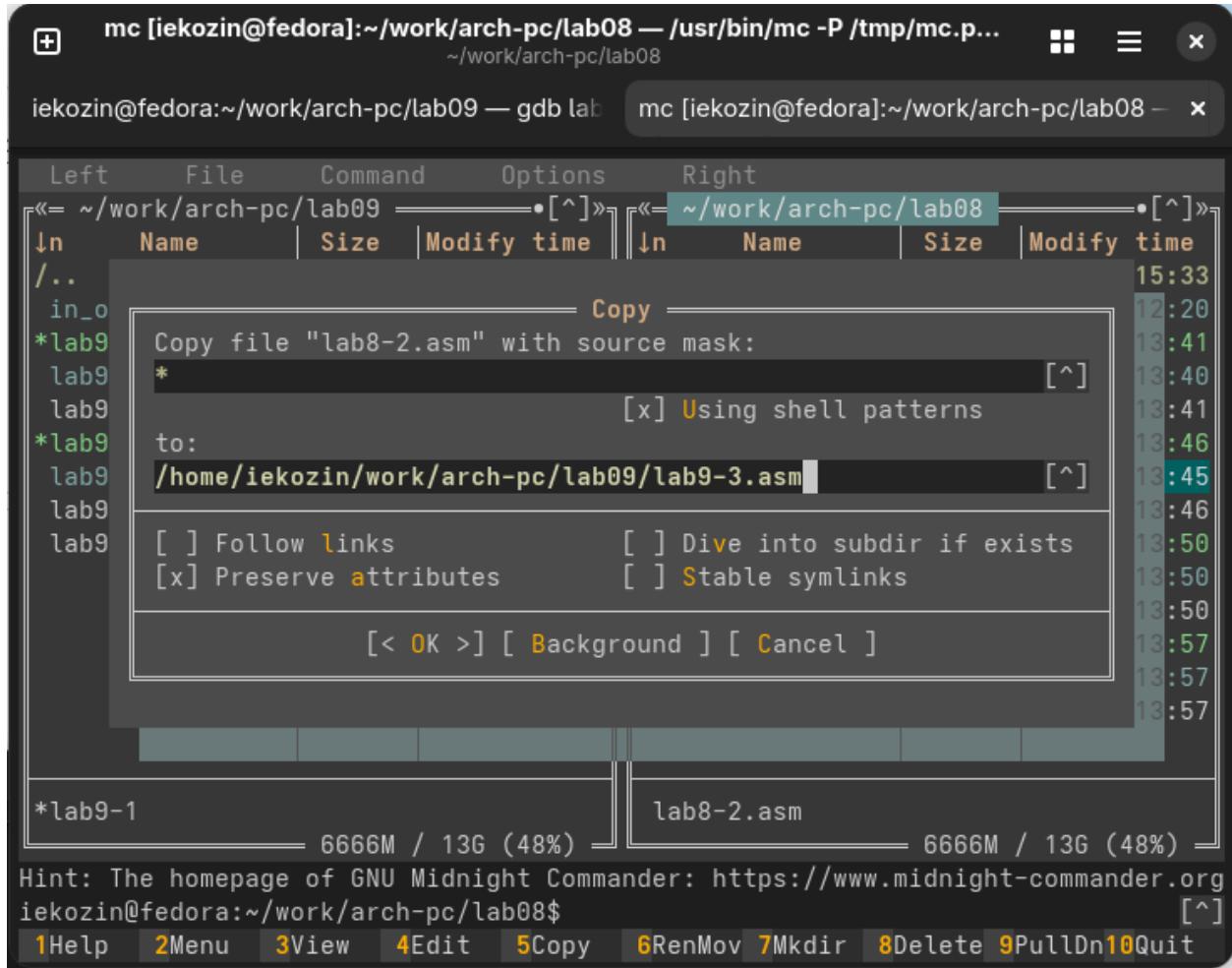
The command line shows the assembly code and current register values:

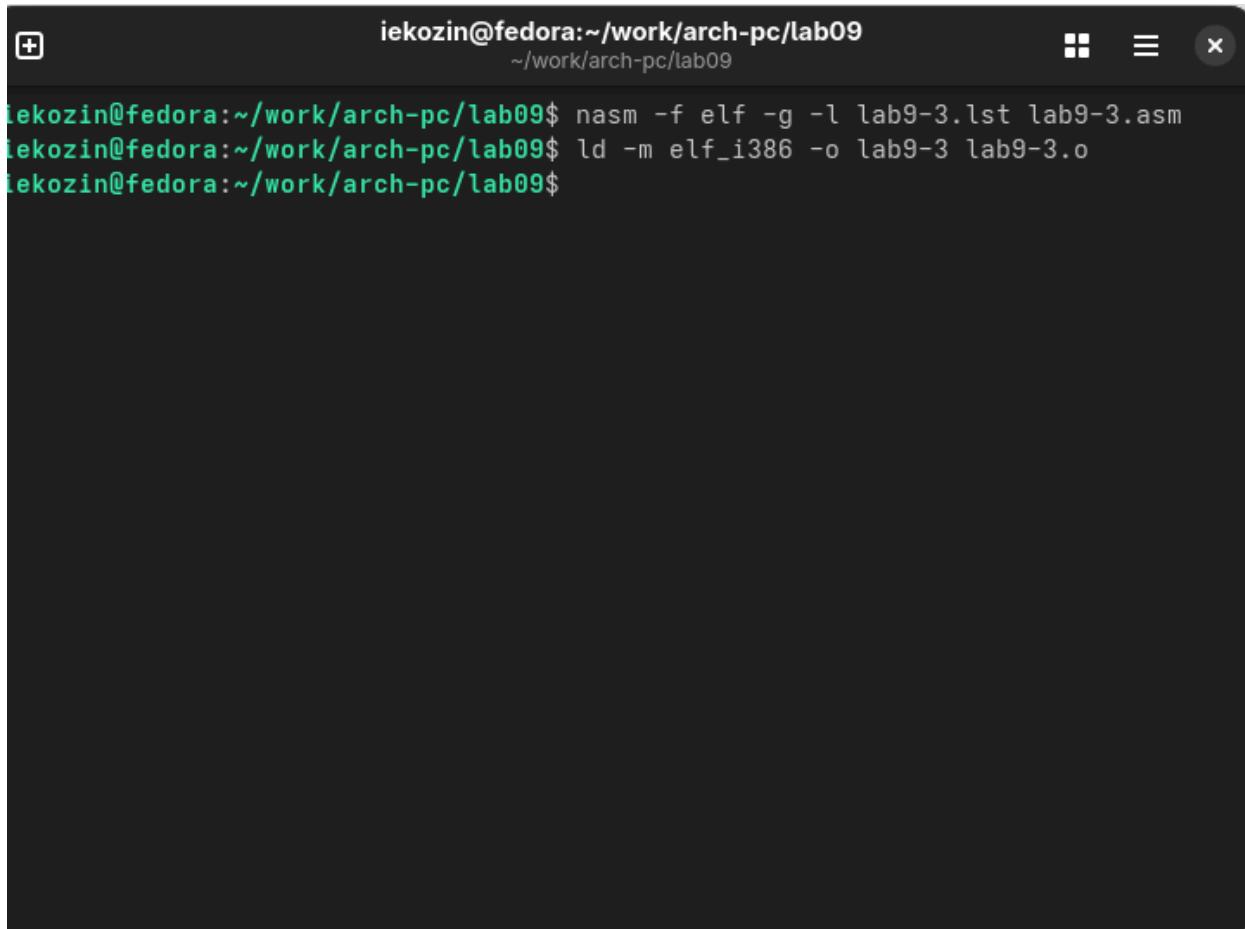
```
native process 31077 (asm) In: _start          L9      PC: 0x8048080
(gdb) p/t $ecx
$1 = 0
(gdb) p/s $edx
$2 = 0
(gdb) set $ebx='2'
(gdb) p/s $ebx
$3 = 50
(gdb)
```

Рис. 14: Просмотр значения регистра разными представлениями

4.1.4 Обработка аргументов командной строки в GDB

Копирую программу из предыдущей лабораторной работы в текущий каталог и создаю исполняемый файл с файлом листинга и отладки (рис. 15).





The screenshot shows a terminal window with the following command history:

```
iekozin@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
iekozin@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
iekozin@fedora:~/work/arch-pc/lab09$
```

Рис. 15: Подготовка новой программы

Запускаю программу с режиме отладки с указанием аргументов, указываю брейкпопнт и запускаю отладку. Проверяю работу стека, изменения 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. (рис. 16).

```
iekozin@fedora:~/work/arch-pc/lab09$ gdb --args lab9-3 аргумент1 аргумент2 'аргумент 3'
GNU gdb (Fedora Linux) 16.3-1.fc42
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-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 lab9-3...
(gdb) b _start
Breakpoint 1 at 0x8048148: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/iekozin/work/arch-pc/lab09/lab9-3 аргумент1 аргумент2 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.

Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/16x $esp
0xfffffcf30:    0x00000004    0xfffffd104    0xfffffd12c    0xfffffd13e
0xfffffcf40:    0xfffffd150    0x00000000    0xfffffd163    0xfffffd173
0xfffffcf50:    0xfffffd1c3    0xfffffd1d7    0xfffffd1ee    0xfffffd205
0xfffffcf60:    0xfffffd235    0xfffffd243    0xfffffd253    0xfffffd27c
(gdb) 
```

```
(gdb) x/s *(void**)( $esp + 4)
0xfffffd104:    "/home/iekozin/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)( $esp + 8)
0xfffffd12c:    "аргумент1"
(gdb) x/s *(void**)( $esp + 12)
0xfffffd13e:    "аргумент2"
(gdb) x/s *(void**)( $esp + 16)
0xfffffd150:    "аргумент 3"
(gdb) x/s *(void**)( $esp + 20)
0x0:    <error: Cannot access memory at address 0x0>
(gdb) x/s *(void**)( $esp + 24)
0xfffffd163:    "SHELL=/bin/bash"
(gdb)
```

Рис. 16: Проверка работы стека

4.2 Задание для самостоятельной работы

1. Меняю программу самостоятельной части предыдущей лабораторной работы с использованием подпрограммы (рис. 18).

```
lab9-4.asm [---] 21 L:[ 'i+ 0' ] * (21 / 9506) 8010 0x00A
#include "in_out.asm"

SECTION .data
msg_func    db "Функция: f(x) = 2*(x-1)", 0
msg_result  db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
    mov eax, msg_func
    call sprintf

    pop edx          ; argc
    pop ecx          ; return addr (не нужен)
    sub ecx, 1       ; исключаем имя программы

    mov esi, 0        ; сумма

next:
    cmp ecx, 0
    jz _end

    pop eax          ; eax = x
    call atoi         ; eax = x
    call _calculate_fx ; eax = 2*(x-1)

    add esi, eax      ; суммируем

    loop next         ; eax--; если не ноль - next

_end:
    mov eax, msg_result
    call sprint

    mov eax, esi
    call iprintf
    call quit

;

1Help 2Save 3Mark 4Replace 5Copy 6Move 7Search 8Delete 9PullDown 10Quit
```

Рис. 18: Измененная программа предыдущей лабораторной работы

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data

msg_func    db "Функция: f(x) = 2*(x-1)", 0
msg_result  db "Результат: ", 0

SECTION .text

GLOBAL _start

_start:
    mov eax, msg_func
```

```
call sprintLF

pop ecx      ; argc
pop edx      ; return addr (не нужен)
sub ecx, 1    ; исключаем имя программы

mov esi, 0    ; сумма

next:
cmp ecx, 0
jz _end

pop eax
call atoi      ; eax = x

call _calculate_fx ; eax = 2*(x-1)

add esi, eax    ; суммируем

loop next       ; ecx--; если не ноль → next

_end:
mov eax, msg_result
call sprint

mov eax, esi
call iprintLF
```

```
call quit

;

; f(x) = 2*(x-1)
; Вход: eax = x
; Выход: eax = 2*(x-1)
;

_calculate_fx:
    sub eax, 1      ; x - 1
    shl eax, 1      ; * 2
    ret
```

2. Запускаю программу в режиме отладчика и пошагово через si просматриваю изменение значений регистров через i r. При выполнении инструкции mul esx можно заметить, что результат умножения записывается в регистр eax, но также меняет и edx. Значение регистра ebx не обновляется напрямую, поэтому результат программы неверно подсчитывает функцию (рис. 19).

The screenshot shows the GDB interface on a Fedora system. The command `gdb lab9-5` has been run, and the program `lab9-5` is currently at the start of its execution.

Registers:

| group: general | | |
|----------------|-------------|-------------|
| eax | 0x0 | 0 |
| ecx | 0x0 | 0 |
| edx | 0x0 | 0 |
| ebx | 0x0 | 0 |
| esp | 0xfffffcf60 | 0xfffffcf60 |
| ebp | 0x0 | 0x0 |

Assembly:

```
B+>0x8048168 <_start>      mov    $0x3,%ebx
0x8048172 <_start+10>     add    %eax,%ebx
0x8048174 <_start+12>     mov    $0x4,%ecx
0x8048179 <_start+17>     mul    %ecx
0x804817b <_start+19>     add    $0x5,%ebx
```

Registers (Native Process):

| native process 34116 (asm) In: | | |
|--------------------------------|-------------|-------------|
| eax | 0x0 | 0 |
| ecx | 0x0 | 0 |
| edx | 0x0 | 0 |
| ebx | 0x0 | 0 |
| esp | 0xfffffcf60 | 0xfffffcf60 |
| ebp | 0x0 | 0x0 |
| esi | 0x0 | 0 |

PC: 0x8048168

--Type <RET> for more, q to quit, c to continue without paging--

Рис. 19: Поиск ошибки в программе через пошаговую отладку

Исправляю найденную ошибку, теперь программа верно считает значение функции (рис. 20).

The screenshot shows a terminal window with the following session:

```
iekozin@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-5.asm
iekozin@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-5.o -o lab9-5
iekozin@fedora:~/work/arch-pc/lab09$ mc

iekozin@fedora:~/work/arch-pc/lab09$ nasm -f elf lab9-5.asm
iekozin@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 lab9-5.o -o lab9-5
iekozin@fedora:~/work/arch-pc/lab09$ ./lab9-5
Результат: 25
iekozin@fedora:~/work/arch-pc/lab09$
```

Рис. 20: Проверка корректировок в программме

Код измененной программы:

```
%include 'in_out.asm'

SECTION .data
div: DB 'Результат: ', 0

SECTION .text
GLOBAL _start
_start:

    mov ebx, 3
    mov eax, 2
    add ebx, eax
    mov 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
```

5 Выводы

В результате выполнения данной лабораторной работы я приобрел навыки написания программ с использованием подпрограмм, а так же познакомился с методами отладки при помощи GDB и его основными возможностями.

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №9
3. Программирование на языке ассемблера NASM Столяров А. В.