

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

дисциплина: Архитектура компьютера

Студент: Сухова Арина

Группа: НПИбд-02-25

МОСКВА

2025 г.

1. Цель работы

Познакомиться с методами отладки при помощи GDB, его возможностями.

2. Выполнение лабораторной работы

2.1 Реализация подпрограмм в NASM.

Создаем каталог для программ ЛБ9, и в нем создаем файл (рис. 1).

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

Рис.1 Создаем каталог с помощью команды mkdir и файл с помощью команды touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.1 (рис. 2).

```
mc [aesukhova@fedora]:~/work/arch-pc/lab09 — /usr/bin/mc -P /tmp/mc.pwd.I0vDkB
~/work/arch-pc/lab09

lab09-1.asm      [-M--] 27 L:[ 1+26 27/ 35] *(517 / 707b) 0045[*][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:
;-----
; 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:
mov ebx,2
mul ebx
add eax,7
```

Рис.2 Заполняем файл

Создаем исполняемый файл и запускаем его (рис. 3).

```
aesukhova@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
aesukhova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1
.o
aesukhova@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2x+7=17
```

Активация Windows
Чтобы активировать Windows, перейдите в раздел

Рис.3 Запускаем файл и проверяем его работу

Снова открываем файл для редактирования и изменяем его, добавив подпрограмму в подпрограмму(по условию) (рис. 4).

```

;-----+
; 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
_calcul:
<----->call _subcalcul
<----->mov ebx,2
<----->mul ebx
<----->add eax,7
<----->mov [res],eax
<----->ret
<----->_subcalcul:
<----->    mov ebx,3
<----->    mul ebx
<----->    sub eax,1
<----->    ret

```

Активация Windows

Рис.4 Изменяем файл, добавляя еще одну подпрограмму

Создаем исполняемый файл и запускаем его (рис. 5).

```

aesukhova@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
aesukhova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1
.o
aesukhova@fedora:~/work/arch-pc/lab09$ ./lab09-1
Введите x: 5
2(3x-1)+7=35

```

Активация Windows

Чтобы активировать Windows, перейдите в раздел
"Параметры".

Рис.5 Запускаем файл и смотрим на его работу

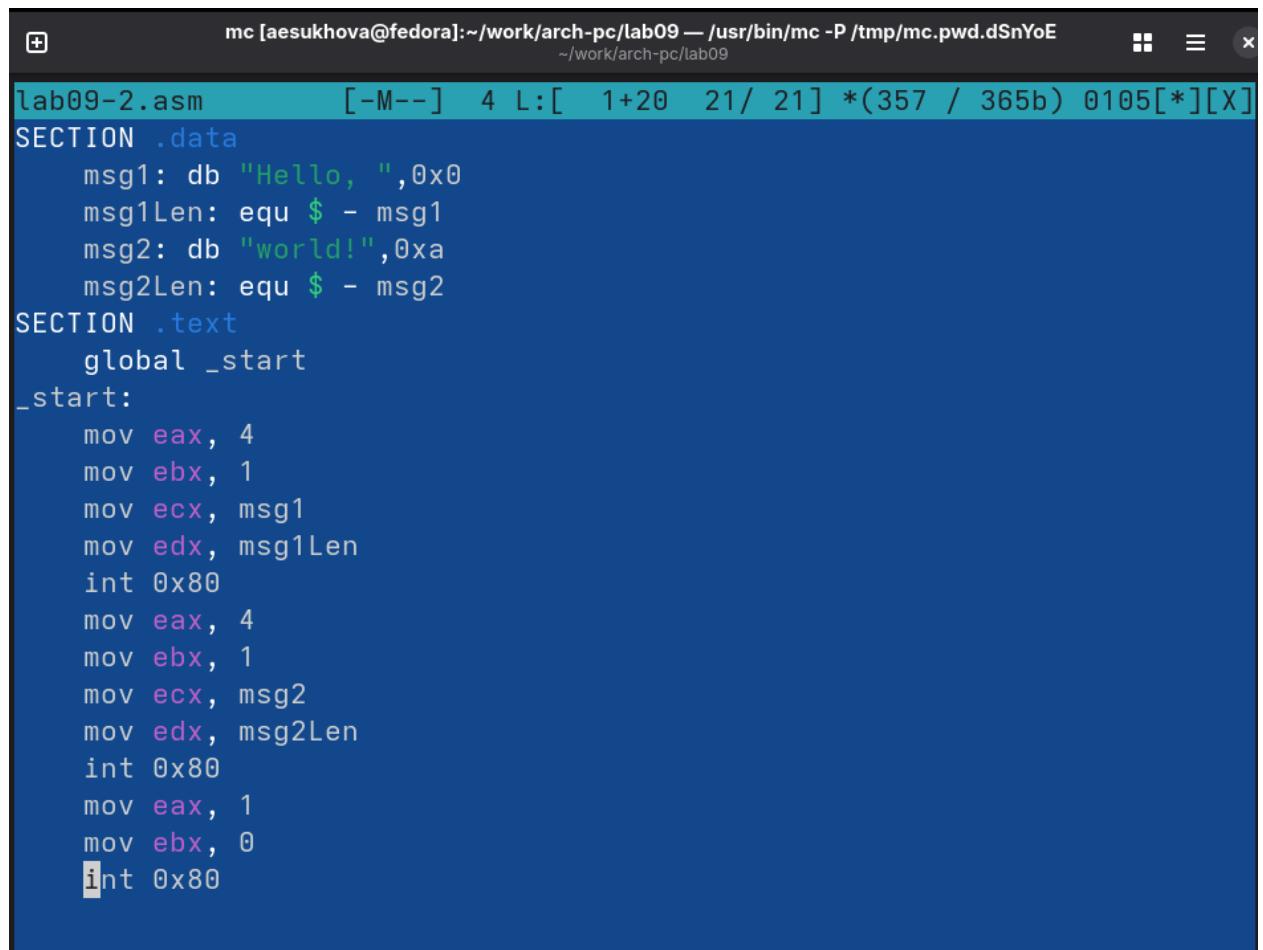
2.2Отладка программ с помощью GDB

Создаем новый файл в каталоге(рис. 6).

```
aesukhova@fedora:~/work/arch-pc/lab09$ touch lab09-2.asm  
aesukhova@fedora:~/work/arch-pc/lab09$
```

Рис.6 Создаем файл

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.2 (рис. 7).



```
mc [aesukhova@fedora:~/work/arch-pc/lab09 — /usr/bin/mc -P /tmp/mc.pwd.dSnYoE  
~/work/arch-pc/lab09  
lab09-2.asm      [-M--]  4 L:[ 1+20  21/ 21] *(357 / 365b) 0105[*][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
```

Рис.7 Заполняем файл

Получаем исходный файл с использованием отладчика gdb (рис. 8).

```
aesukhova@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
aesukhova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
aesukhova@fedora:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 16.2-3.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 lab09-2...  
Активация Windows
(gdb) █  
Чтобы активировать Windows, перейдите в раздел
"Параметры".
```

Рис.8 Загружаем исходный файл в отладчик

Запускаем команду в отладчике (рис. 9).

```
(gdb) run
Starting program: /home/aesukhova/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.
Hello, world!  
[Inferior 1 (process 7728) exited normally]  
Активация Windows
(gdb) █  
Чтобы активировать Windows, перейдите в раздел
"Параметры".
```

Рис.9 Запускаем программу командой run

Устанавливаем брейкпоинт на метку _start и запускаем программу (рис. 10).

```
(gdb) break _start
Breakpoint 1 at 0x8048080: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/aesukhova/work/arch-pc/lab09/lab09-2

Breakpoint 1, _start () at lab09-2.asm:9
9          mov    eax, 4
(gdb)
```

Активация Windows
Чтобы активировать Windows, перейдите
"Параметры".

Рис.10 Запускаем программу с брейкпоином

Смотрим дисассимилированный код программы с помощью команды disassemble, начиная с метки _start(рис. 11).

```
(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)
```

Активация Windows
Чтобы активировать Windows, перейдите в раздел
"Параметры".

Рис.11 Смотрим дисассимилированный код программы

Переключаемся на отображение команд с Intel'овским синтаксисом (рис. 12).

```
(gdb) set disassembly-flavor intel
(gdb) disassemle _start
Undefined command: "disassemle". Try "help".
(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)
```

Активация Windows

Чтобы активировать Windows, перейдите
"Параметры".

Рис.12 Переключаемся на синтаксис Intel

Различия отображения синтаксиса машинных команд в режимах ATT и Intel:

1. Порядок operandов: В ATT синтаксисе порядок operandов обратный, сначала указывается исходный operand, а затем - результирующий operand. В Intel синтаксисе порядок обычно прямой, результирующий operand указывается первым, а исходный - вторым.
2. Разделители: В ATT синтаксисе разделители operandов - запятые. В Intel синтаксисе разделители могут быть запятые или косые черты (/).
3. Префиксы размера operandов: В ATT синтаксисе размер operandана указывается перед operandом с использованием префиксов, таких как "b" (byte), "w" (word), "l" (long) и "q" (quadword). В Intel синтаксисе размер operandана указывается после operandана с использованием суффиксов, таких как "b", "w", "d" и "q".
4. Знак operandов: В ATT синтаксисе operandы с позитивными значениями предваряются символом "\$". В Intel синтаксисе operandы с позитивными значениями могут быть указаны без символа "\$".
5. Обозначение адресов: В ATT синтаксисе адреса указываются в круглых скобках. В Intel синтаксисе адреса указываются без скобок.
6. Обозначение регистров: В ATT синтаксисе обозначение регистра начинается с символа "%". В Intel синтаксисе обозначение регистра может начинаться с символа "R" или "E" (например, "%eax" или "RAX").

Включаем режим псевдографики (рис. 13).

```
aesukhova@fedora:~/work/arch-pc/lab09 — gdb lab09-2
~/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
esi            0x0          0
edi            0x0          0
eip            0x8048080 <_start>

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
    0x80480a5 <_start+37>   mov    edx,0x7
    0x80480aa <_start+42>   int    0x80

native process 7746 (asm) In: _start          L9      PC: 0x8048080
(gdb) layout regs
(gdb)
```

Рис.13 Включаем отображение регистров, их значений и результатом дисассимилирования программы

Проверяем была ли установлена точка останова и устанавливаем точку останова предпоследней инструкции (рис. 14).

```
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031
(gdb) delete 2
(gdb) break *$eip+49
Breakpoint 3 at 0x80480b1: file lab09-2.asm, line 20.
(gdb) continueHello, world!
```

Continuing.

```
Breakpoint 3, _start () at lab09-2.asm:20
(gdb) █
```

Активация Windows
Чтобы активировать Windows,
"Параметры".

Рис.14 Используем команду *info breakpoints* и создаем новую точку останова

Посмотрим информацию о всех установленных точках останова (рис. 15).

```
(gdb) i b
Num      Type            Disp Enb Address      What
1        breakpoint      keep y    0x08048080 lab09-2.asm:9
          breakpoint already hit 1 time
3        breakpoint      keep y    0x080480b1 lab09-2.asm:20
          breakpoint already hit 1 time
(gdb) █
```

Активация Windows
Чтобы активировать Windows, перейдите в р
"Параметры".

Рис.15 Смотрим информацию

Выполняем 5 инструкций командой *si* (рис. 16).

```

aesukhova@fedora:~/work/arch-pc/lab09 — gdb lab09-2
~/work/arch-pc/lab09

eax      group: general          0
esi      0x0                      0
edi      0x0                      0
eip      0x8048096              0x8048096 <_start+22>
eflags   0x202                  [ IF ]
cs       0x23                   35
ss       0x2b                   43
ds       0x2b                   43
es       0x2b                   43
fs       0x0                      0

B+ 0x8048080 <_start>    mov    $0x4,%eax
    0x8048085 <_start+5>  mov    $0x1,%ebx
    0x804808a <_start+10>  mov    $0x8049000,%ecx
    0x804808f <_start+15>  mov    $0x8,%edx
    0x8048094 <_start+20>  int    $0x80
>0x8048096 <_start+22>  mov    $0x4,%eax
    0x804809b <_start+27>  mov    $0x1,%ebx
    0x80480a0 <_start+32>  mov    $0x8049008,%ecx
    0x80480a5 <_start+37>  mov    $0x7,%edx
    0x80480aa <_start+42>  int    $0x80
native process 8228 (asm) In: _start                                L9    PC: 0x8048080
Program process 8356 (asm) In: _start                                L14   PC: 0x8048096
Breakpoint 1, _start () at lab09-2.asm:9
(gdb) i b
Num      Type            Disp Enb Address     What
1        breakpoint      keep y  0x08048080 lab09-2.asm:9
                breakpoint already hit 1 time
3        breakpoint      keep y  0x080480b1 lab09-2.asm:20
(gdb) si
(gdb) si
(gdb) si
(gdb) si
(gdb) siHello,
(gdb) █
                                         Активация Windows
Чтобы активировать Windows, перейдите в раздел
"Параметры".

```

Рис.16 Отслеживаем регистры

Во время выполнения команд менялись регистры: ebx, ecx, edx, eax, eip.

Смотрим значение переменной msg1 по имени (рис. 17).

```

(gdb) x/1sb &msg1
0x8049000 <msg1>:      "Hello, "
(gdb)

```

Рис.17 Смотрим значение переменной

Смотрим значение переменной msg2 по адресу (рис. 18).

```
(gdb) x/1sb 0x8049008
0x8049008 <msg2>:          "world!\n\034" Ак
(gdb) █
                                         ЧТо
                                         "П
```

Рис.18 Смотрим значение переменной

Изменим первый символ переменной msg1 (рис. 19).

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x8049000 <msg1>:          "hello, "
(gdb) █
```

Рис.19 Меняем символ

Изменим первый символ переменной msg2 (рис. 20).

```
(gdb) set {char}&msg2='L'
(gdb) x/1sb &msg2
0x8049008 <msg2>:          "Lord!\n\034" Ак
(gdb) █
                                         ЧТо
                                         "П
```

Рис.20 Меняем символ

Смотрим значение регистра edx в разных форматах (рис. 21).

```
(gdb) p/t $edx
$1 = 1000
(gdb) p/s $edx
$2 = 8
(gdb) p/x $edx
$3 = 0x8
```

Рис.21 Смотрим значение регистра

Изменяю регистор ebx (рис. 22).

```
(gdb) set $ebx='2'
(gdb) p/s $ebx
$4 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$5 = 2
(gdb) █
```

Рис.22 Изменяю регистор командой set

Выводится разные значения, так как команда без кавычек присваивает регистру вводимое значение.

Прописываем команды для завершения программы и выхода из GDB (рис. 23).

```
(gdb) cLorld!  
  
Continuing.  
  
Breakpoint 3, _start () at lab09-2.asm:20  
(gdb) █
```

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Рис.23 Прописываем команды с и quit

Копируем файл lab8-2.asm в файл с именем lab09-3.asm (рис. 24).

```
aesukhova@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm  
aesukhova@fedora:~/work/arch-pc/lab09$ █
```

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Рис.24 Копируем файл

Создаем исполняемый файл и запускаем его в отладчике GDB (рис. 25).

```
aesukhova@fedora:~/work/arch-pc/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm  
aesukhova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o  
aesukhova@fedora:~/work/arch-pc/lab09$ gdb --args lab09-3 2 3 '5'  
█
```

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Рис.25 Создаем и запускаем в отладчике файл

Установим точку останова перед первой инструкцией в программе и запустим ее (рис. 26).

```
(gdb) b _start  
Breakpoint 1 at 0x8048148: file lab09-3.asm, line 5.  
(gdb) run  
Starting program: /home/aesukhova/work/arch-pc/lab09/lab09-3 2 3 5  
  
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 lab09-3.asm:5  
5          pop  ecx  
(gdb) x/x &esp  
No symbol "esp" in current context.  
(gdb) x/x $esp  
0xfffffcf70:    0x00000004  
(gdb) █
```

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Рис.26 Устанавливаем точку останова

Смотрим позиции стека по разным адресам (рис. 27).

```
(gdb) x/s *(void**)( $esp + 4 )
0xfffffd149:    "/home/aesukhova/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)( $esp + 8 )
0xfffffd174:    "2"
(gdb) x/s *(void**)( $esp + 12 )
0xfffffd176:    "3"
(gdb) x/s *(void**)( $esp + 16 )
0xfffffd178:    "5"
(gdb) x/s *(void**)( $esp + 20 )
0x0: <error: Cannot access memory at address 0x0> Активация Windows
      Чтобы активировать Windows, перейдите в раздел
      "Параметры".
(gdb) █
```

Рис.27 Изучаем полученные данные

Шаг изменения адреса равен 4 потому что адресные регистры имеют размерность 32 бита(4 байта).

Задание для самостоятельной работы

Задание 1

Копируем файл lab8-4.asm(ср №1 в ЛБ8) в файл с именем lab09-3.asm (рис. 28).

```
aesukhova@fedora:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-4.1.
asm ~/work/arch-pc/lab09/lab09-4.asm Активация Windows
      Чтобы активировать Windows, перейдите в раздел
      "Параметры".
aesukhova@fedora:~/work/arch-pc/lab09$
```

Рис.28 Копируем файл

Открываем файл в Midnight Commander и меняем его, создавая подпрограмму (рис. 29).

```
mc [aesukhova@fedora]:~/work/arch-pc/lab09 — /usr/bin/mc -P /tmp/mc.pwd.W6SF6G
~./work/arch-pc/lab09

lab09-4.asm      [----]  0 L:[ 1+ 0   1/197] *(0   /3373b) 0115[*][X]
section .data
    msg_func db "Функция: f(x)=4x+3", 0xA, 0
    msg_res db "Результат: ", 0
    newline db 0xA, 0

section .bss
    sum resd 1
    buffer resb 10

section .text
    global _start

_start:
; Инициализация суммы
    mov dword [sum], 0
.....
; Получаем аргументы командной строки
    pop ecx          ; argc
    pop edx          ; argv[0] (имя программы)
    dec ecx          ; количество чисел
.....
; Выводим информацию о функции
    mov eax, msg_func
    call sprint
.....
; Проверяем есть ли аргументы
    cmp ecx, 0
    je print_result

process_args:
    pop eax          ; получаем аргумент (строку)
    call atoi         ; преобразуем в число
    Активация Windows
```

Рис.29 Изменяем файл

Создаем исполняемый файл и запускаем его (рис. 30).

```
aesukhova@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
aesukhova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4
.o
aesukhova@fedora:~/work/arch-pc/lab09$ ./lab09-4 5
Функция: f(x)=4x+3
Результат: 23
aesukhova@fedora:~/work/arch-pc/lab09$ Активация Windows
Чтобы активировать Windows, перейдите в раздел
"Параметры".
```

Рис.30 Проверяем работу программы

Задание 2

Создаем новый файл в директории (рис. 31).

```
aesukhova@fedora:~/work/arch-pc/lab09$ touch lab09-5.asm
aesukhova@fedora:~/work/arch-pc/lab09$
```

Рис.31 Создаем файл

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 9.3 (рис. 32).

```
lab09-5.asm      [----] 13 L:[ 1+17  18
%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 ecx,4
    mul ecx
    add ebx,5
    mov edi,ebx
    mov eax,dib
    call sprint
    mov eax,edi
    call iprintLF
    call quit
```

Рис.32 Изменяем файл

Создаем исполняемый файл и запускаем его (рис. 33).

```
aesukhova@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
aesukhova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5
.o
aesukhova@fedora:~/work/arch-pc/lab09$ ./lab09-5
Результат: 10
aesukhova@fedora:~/work/arch-pc/lab09$
```

Рис.33 Создаем и смотрим на работу программы(работает неправильно)

Создаем исполняемый файл и запускаем его в отладчике GDB и смотрим на изменение реегистров командой si (рис. 34).

Рис.34 Ищем ошибку регистров в отладчике

Изменяем программу для корректной работы (рис. 35).

```
mc [aesukhova@fedora]:~/work/arch-pc/lab09 — /usr/bin/mc -P /tmp/mc.pwd.mBAp9p
~/work/arch-pc/lab09

lab09-5.asm      [-M--] 15 L:[ 1+13 14/ 20] *(261 / 396b) 0010[*][X]
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
    mov ebx,3
    mov eax,2
    add eax,ebx
    mov ecx,4
    mul ecx
    add eax,5
    mov edi,eax
; ---- Вывод результата на экран
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    call quit
```

Рис.35 Меняем файл

Создаем исполняемый файл и запускаем его (рис. 36).

```
aesukhova@fedora:~/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
aesukhova@fedora:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
aesukhova@fedora:~/work/arch-pc/lab09$ ./lab09-5
Результат: 25
Чтобы активировать Windows, перейдите в раздел
Параметры".
```

Рис.36 Создаем и запускаем файл(работает корректно)

3. Выводы

Мы познакомились с методами отладки при помощи GDB и его возможностями.