

Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Ванюшкина Татьяна Валерьевна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выводы	16
	Список литературы	17

Список иллюстраций

Список таблиц

1 Цель работы

Целью работы является приобретение навыков написания программ с использованием подпрограмм и знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Теоретическое введение

GDB (GNU Debugger — отладчик проекта GNU) [1] работает на многих UNIX-подобных системах и умеет производить отладку многих языков программирования. GDB предлагает обширные средства для слежения и контроля за выполнением компьютерных программ. Отладчик не содержит собственного графического пользовательского интерфейса и использует стандартный текстовый интерфейс консоли. Однако для GDB существует несколько сторонних графических надстроек, а кроме того, некоторые интегрированные среды разработки используют его в качестве базовой подсистемы отладки. Отладчик GDB (как и любой другой отладчик) позволяет увидеть, что происходит «внутри» программы в момент её выполнения или что делает программа в момент сбоя. GDB может выполнять следующие действия:

- начать выполнение программы, задав всё, что может повлиять на её поведение;
- остановить программу при указанных условиях;
- исследовать, что случилось, когда программа остановилась;

3 Выполнение лабораторной работы

1. Реализация подпрограмм в NASM

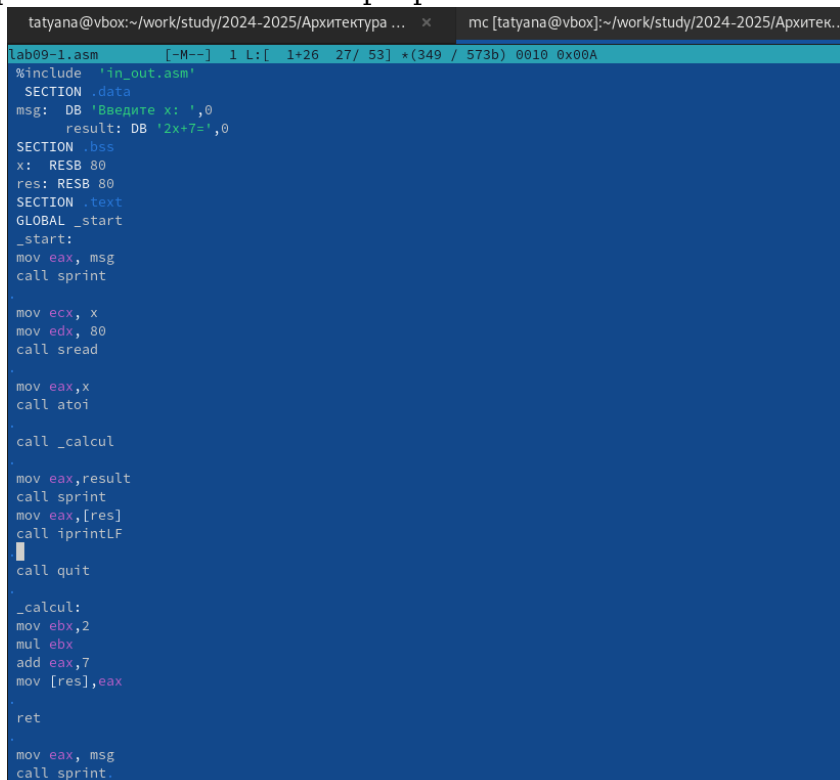
Создаю каталог для выполнения лабораторной работы №9:

(рис.1 ??) `tatyana@vbox:~$ mkdir ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09` {#fig:001}

Перехожу в него и создаю файл lab09-1.asm:

(рис.2 ??) `tatyana@vbox:~$ cd ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ touch lab09-1.asm` {#fig:002}

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



```
lab09-1.asm [-M--] 1 L: [ 1+26 27/ 53] +(349 / 573b) 0010 0x00A
%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

mov eax, msg
call sprint
```

(рис.3 ??) {#fig:003}

Создаю исполняемый файл и проверяю его работу:

```
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf lab09-1.asm
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ./lab09-1
Введите x: 10
2x+7=27
```

(рис.4 ??)

{#fig:004}

Изменяю текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения $2x + 7$, где x вводится с клавиатуры, $2x + 7$, $2x + 7 = 3x - 1$.

```
lab09-1.asm [-M--] 10 L: 1+18 19/ 46 + (274 / 520b) 0010 0x00A
%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

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:
push ebx
mov ebx, x
mul ebx
add ebx, 7
mov [res], ebx
pop ebx
ret
```

(рис.5 ??)

{#fig:005}

Создаю исполняемый файл и проверяю его работу:

```
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf lab09-1.asm
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ./lab09-1
Введите x: 10
2(3x-1)+7=65
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$
```

(рис.6 ??)

{#fig:006}

Создаю файл `lab09-2.asm` :

(рис.7 ??)

```
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ touch lab09-2.asm
```

{#fig:007}

Ввожу в него текст программы из Листинга 9.2:


```

tatyana@vbox:~/work/study/2024-2025/Архитектура ... x  inc [tatyana@vbox:~/work/study/202
lab09-2.asm [-M--] 10 L: [ 1+ 4 5/ 21] *(112 / 340b) 0032 0x020
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.8 ??) {#fig:008}

Получаю исполняемый файл.Для работы с GDB в исполняемый файл добавляю отладочную информацию,для этого трансляцию программ провожу с ключом ‘-g’.

(рис.9 ??) {#fig:009}

```

tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o

```

Загружаю исполняемый файл в отладчик gdb:

```

tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf -g -l lab09-2.lst lab09-2.asm
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
tatyana@vbox:~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ gdb lab09-2
GNU gdb (Fedora Linux) 15.2-2.fc40
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...
(gdb)

```

(рис.10 ??) {#fig:010}

Проверяю работу программы,запустив ее в оболочке GDB с помощью команды run:

```

(gdb) run
Starting program: /home/tatyana/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/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
Hello, world!
[Inferior 1 (process 11035) exited normally]
(gdb)

```

(рис.11 ??) {#fig:011}

Для более подробного анализа программы устанавливаю брейкпоинт на метку _start, с которой начинается выполнениелюбой ассемблерной программы,и запускаю её:

(рис.12 ??)

```
(gdb) break _start
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 9.
(gdb) run
Starting program: /home/tatyana/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09/lab09-2
Breakpoint 1, _start () at lab09-2.asm:9
9      mov eax, 4
(gdb) |
```

{#fig:012}

Смотрю дисассимилированный код программы с помощью команды disassemble начиная с метки _start:

(рис.13 ??)

```
(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.
```

{#fig:013}

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду set disassembly-flavor intel:

(рис.14 ??)

```
(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.
```

{#fig:014}

Включаю режим псевдографики для более удобного анализа программы:

(рис.15 ??)

```
Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcf70 0xffffcf70  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

0x08049000 <_start> mov eax,0x4
0x08049005 <_start+5> mov ebx,0x1
0x0804900a <_start+10> mov ecx,0x804a000
0x0804900f <_start+15> mov edx,0x8
0x08049014 <_start+20> int 0x80
0x08049016 <_start+22> mov eax,0x4
0x0804901b <_start+27> mov ebx,0x1
0x08049020 <_start+32> mov ecx,0x804a008
0x08049025 <_start+37> mov edx,0x7
0x0804902a <_start+42> int 0x80
0x0804902c <_start+44> mov eax,0x1
0x08049031 <_start+49> mov ebx,0x0
0x08049036 <_start+54> int 0x80

native process 11066 (asm) In: _start
(gdb) layout regs
```

{#fig:015}

2. Добавление точек останова

На предыдущих шагах была установлена точка останова по имени мет-

ки(_start).Про верьте это с помощью команды info breakpoints:

```

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcf70 0xffffcf70  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

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
0x8049036 <_start+54> int     0x80

native process 11066 (asm) In: _start
(gdb) layout regs
(gdb) info breakpoints
Num      Type          Disp Enb Address      What
1        breakpoint    keep y  0x8049000 lab09-2.asm:9
        breakpoint already hit 1 time
(gdb) break 0x8049000
Function "0x8049000" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 2 (0x8049000) pending.
(gdb)

```

(рис.16 ??)

{#fig:016}

Устанавливаю еще одну точку останова по адресу инструкции:

```

Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcf70 0xffffcf70  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

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
0x8049036 <_start+54> int     0x80

native process 11066 (asm) In: _start
(gdb) layout regs
(gdb) info breakpoints
Num      Type          Disp Enb Address      What
1        breakpoint    keep y  0x8049000 lab09-2.asm:9
        breakpoint already hit 1 time
(gdb) break 0x8049000
Function "0x8049000" not defined.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 2 (0x8049000) pending.
(gdb)

```

(рис.17 ??)

{#fig:017}

Смотрю информацию о всех установленных точках останова:

```

(gdb) i b
Num      Type          Disp Enb Address      What
1        breakpoint    keep y  0x8049000 lab09-2.asm:9
        breakpoint already hit 1 time
2        breakpoint    keep y  <PENDING> 0x8049000
(gdb)

```

(рис.18 ??)

{#fig:018}

3. Работа с данными программы в GDB

Смотрю содержание регистров с помощью команды info registers:

```

eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcf70 0xffffcf70  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

0x8049000 <_start> mov     eax,0x0
0x8049005 <_start+5> mov     ebx,0x1
0x804900a <_start+10> mov     ecx,0x804a000
0x804900f <_start+15> mov     edx,0x8
0x8049014 <_start+20> int     0x90
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
0x8049036 <_start+54> int     0x90

native process 11066 (asm) In: _start L9 PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcf70 0xffffcf70
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
--Type <RET> for more, q to quit, c to continue without paging--

```

(рис.19 ??) {#fig:019}

Смотрю значение переменной msg1 по имени:

```

(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb)

```

(рис.20 ??) {#fig:020}

Меню первый символ переменной msg1:

```

(gdb) set (char)msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb)

```

(рис.21 ??) {#fig:021}

С помощью команды set меняю значение регистра ebx:

```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50

```

(рис.22 ??) {#fig:022}

```

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

```

(рис.23 ??) {#fig:023}

4. Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки (Листинг 8.2) в файл с именем lab09-3.asm:

(рис.24 ??) {#fig:024}

```

tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ cp ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab08/lab8-2.asm ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab09/lab09-3.asm

```

Создаю исполняемый файл:

(рис.25 ??) {#fig:025}

```

tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf -g -l lab09-3.lst lab09-3.asm
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab09-3 lab09-3.o

```

Загружаю исполняемый файл в отладчик,указав аргументы:

```
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ gdb --args lab09-3 аргумент1 аргумент 2 'а
ргумент 3'
GNU gdb (Fedora Linux) 15.2-2.fc40
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 lab09-3...
(gdb)
```

(рис.26 ??)

{#fig:026}

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее:

```
(gdb) b _start
Breakpoint 1 at 0x00490e8: file lab09-3.asm, line 5.
(gdb) run
Starting program: /home/tatyana/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/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.

Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx
(gdb)
```

(рис.27 ??)

{#fig:027}

Адрес вершины стека храниться в регистре esp и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы):

```
(gdb) x/x $esp
0xfffff153: 0x00000005
(gdb)
```

(рис.28 ??)

{#fig:028}

Посматриваю остальные позиции стека—по адресу [esp+4] располагается адрес в памяти где находится имя программы, по адресу [esp+8] храниться адрес первого аргумента, по адресу [esp+12]—второго ит.д:

```
(gdb) x/s *(void**)($esp + 4)
0xfffff153: "/home/tatyana/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09/lab09-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd1bb: "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xffffd1cf: "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xffffd1e8: "2"
(gdb) x/s *(void**)($esp + 20)
0xffffd1e2: "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

(рис.29 ??)

{#fig:029}

Задание для самостоятельной работы:

1. Преобразовываю программу из лабораторной работы №8, реализовав вычисление значения функции $\pi(\pi)$ как подпрограмму:

```

#include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=7(x+1)", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintLF

pop ecx
pop edx
sub eax, 1
mov esi, 0

next:
cmp ecx, 0h
jz _end
pop eax
call atoi
add esi, eax

call _calculate_fx

add esi, eax
loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit

add eax, 1
mov ebx, 7
mul ebx

ret

```

(рис.30 ??)

{#fig:030}

2. Создаю файл lab09-5.asm:

(рис.31 ??)

```
tatyana@vbox: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ touch lab09-5.asm
```

{#fig:031}

Ввожу в него текст программы листигна 9.3:

```

lab09-5.asm [----] 10 L: [ 1+17 18/ 18] *(257 / 257b) <
#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,div
call sprint
mov eax,edi
call iprintLF
call quit

```

(рис.32 ??)

{#fig:032}

Получаю исполняемый файл и загружаю его в отладчик gdb:

```
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf -g -l lab09-5.lst lab09-5.asm
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ gdb lab09-5
GNU gdb (Fedora Linux) 15.2-2.fc40
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 lab09-5...
(gdb)
```

(рис.33 ??)

{#fig:033}

Просматриваю изменение значений регистров.

```
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09 — gdb lab09-5
tatyana@vbox: /work/s... x tatyana@vbox: /work/s... x tatyana@vbox: /work/s... x tatyana@vbox: /work/s... x
--Register group: general
eax      0x0      0      ecx      0x0      0
edx      0x0      0      ebx      0x0      0
esp      0xffffcfd0 0xffffcfd0  ebp      0x0      0x0
esi      0x0      0      edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>  eflags    0x202    [ IF ]
cs       0x23     35      ss       0x2b     43
ds       0x2b     43      es       0x2b     43
fs       0x0      0      gs       0x0      0

B> 0x80490e8 <_start> mov     ebx,0x2
0x80490f4 <_start+5> mov     eax,0x2
0x80490f2 <_start+10> add     ebx,eax
0x80490f4 <_start+12> mov     ecx,0x4
0x80490f9 <_start+17> mul     ecx
0x80490fb <_start+19> add     ebx,0x5
0x80490fe <_start+22> mov     edi,ebx
0x8049100 <_start+24> mov     eax,0x04a000
0x8049105 <_start+29> call    0x804900f <sprint>
0x804910a <_start+34> mov     eax,edi
0x804910c <_start+36> call    0x8049086 <printfLF>
0x8049111 <_start+41> call    0x80490db <quit>
0x8049115 add     BYTE PTR [eax],al

native process 12516 (asm) In: _start L7 PC: 0x80490e8
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfd0 0xffffcfd0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>
eflags    0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--
```

(рис.34 ??)

{#fig:034}

При выполнении инструкции mul результат умножения меняет edx, значение регистра ebx не обновляется напрямую. Из-за этого программа неверно подсчитывает функцию

Исправляю ошибку программы и запускаю исполняемый файл:

```
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ nasm -f elf lab09-5.asm
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
tatyana@vbox: /work/study/2024-2025/Архитектура компьютера/arch-pc/labs/lab09$ ./lab09-5
Результат: 25
```

(рис.35 ??)

{#fig:035}

4 Выводы

В ходе выполнения лабораторной работы я приобрела навыки написания программ с использованием подпрограмм и познакомилась с методами отладки при помощи GDB и его основными возможностями.

Список литературы

Курс: Архитектура компьютеров и операционные системы. Раздел “Архитектура компьютеров” (02.03.00, УГСН) (rudn.ru)