

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №9

Дисциплина: Архитектура компьютеров

Студент: Болдырева Дельгир

Группа: НКАбд-01-25

Москва
2025 г.

Оглавление

1 Цель работы.....	3
2 Задание.....	4
3 Теоретическое введение.....	5
4 Выполнение лабораторной работы.....	7
4.1 Реализация подпрограмм в Nasm.....	7
4.2 Откладка программам с помощью GDB	7
4.3 Обработка аргументов командной строки в GDB.....	
5 Задания для самостоятельной работы.....	11
6 Выводы.....	12
Список литературы.....	13

1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм.

Знакомство с методами отладки при помощи GDB и его основными

Возможностями

2 Задания

Реализация подпрограмм в NASM, Отладка программам с помощью GDB,

Задание для самостоятельной работы

3 Теоретическое введение

Отладка — процесс поиска и исправления ошибок в программе. Как правило, он состоит из четырех основных этапов:

1. Работает некорректно.

вызывающего проблему.

Обнаружение ошибки — понимание того, что программа

2. Поиск места ошибки — локализация фрагмента кода,

3. возникает ошибка.

Определение причины — анализ, почему в данном месте

4. Исправление ошибки — изменение кода для устранения

проблемы, после чего цикл может повториться для других ошибок.

Можно выделить три основных типа ошибок:

- Синтаксические ошибки — нарушение правил языка программирования; обнаруживаются на этапе трансляции (компиляции/интерпретации).
- Семантические (логические) ошибки — программа запускается и выполняется, но выдает непредусмотренный или неверный результат из-за ошибочной логики.
- Ошибки в процессе выполнения — возникают во время работы программы и могут приводить к её аварийному завершению (например, деление на ноль или переполнение памяти).

Наиболее сложным часто оказывается второй этап — поиск места ошибки.

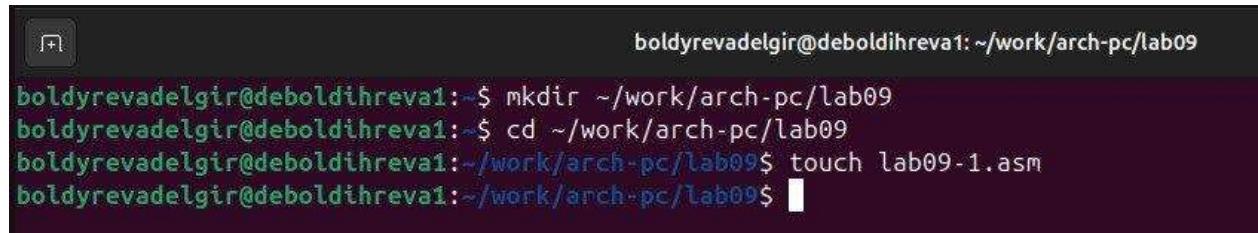
Для этого полезно разбивать программу на части и проверять их отдельно, используя методы отладки (например, вывод промежуточных значений, пошаговое выполнение, тестирование модулей).

После локализации ошибки обычно проще понять её причину и внести необходимые исправления. Однако после исправления одной ошибки могут обнаруживаться другие, и процесс отладки продолжается.

4 Выполнение работы

4.1 Реализация подпрограмм в Nasm

Создадим каталог для выполнения лабораторной работы №9, перейдем в него и создадим файл lab09-1.asm



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

Рис.4.1.1

В качестве примера рассмотрим программу вычисления арифметического выражения $f(x) = 2x + 7$ с помощью подпрограммы _calcul. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме.

Внимательно изучим текст программы и перепишем его



```
lab09-1.asm
~/work/arch-pc/lab09

%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
```

Рис.4.1.2

Создадим исполняемый файл и проверим его работу

```
boldyrevadelgir@deboldihrevali:/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
boldyrevadelgir@deboldihrevali:/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
boldyrevadelgir@deboldihrevali:/work/arch-pc/lab09$ ./lab09-1
```

```
Введите x: 10
2x+7=27
boldyrevadelgir@deboldihrevali:/work/arch-pc/lab09$
```

Рис.4.1.3

Изменим текст программы, добавив подпрограмму _subcalcul в подпрограмму _calcul, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x) = 2x + 7$, $g(x) = 3x - 1$. Т.е. x передается в подпрограмму _calcul из нее в подпрограмму _subcalcul, где вычисляется выражение $g(x)$, результат возвращается в _calcul и вычисляется выражение $f(g(x))$. Результат возвращается в основную программу для вывода результата на экран

```
Открыть ▾ lab09-1.asm
\work\arch-pc\lab09

%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 tprintLF
```

Рис.4.1.4

Создадим исполняемый файл и проверим его работу

```
boldyrevadelgir@deboldihrevali:/work/arch-pc/lab09$ nasm -f elf lab09-1.asm
boldyrevadelgir@deboldihrevali:/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-1 lab09-1.o
boldyrevadelgir@deboldihrevali:/work/arch-pc/lab09$ ./lab09-1
```

```
Введите x: 10
2(3x-1)+7=65
boldyrevadelgir@deboldihrevali:/work/arch-pc/lab09$
```

Рис.4.1.5

4.2 Отладка программ с помощью GDB

Создадим файл lab09-2.asm с текстом программы из Листинга

```
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab09$ touch lab09-2.asm
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab09$
```

lab09-2.asm
~/work/arch-pc/lab09

lab09-1.asm

lab09-2.asm

```
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.4.2.1

Загрузим исполняемый файл в отладчик gdb

```
boldyrevadelgir@deboldihrevali:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 15.0.50.20240403-0ubuntu1) 15.0.50.20240403-glt
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 lab09-2...
(gdb)
```

Рис.4.2.2

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

```
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 4696) exited normally]  
(gdb) █
```

Рис.4.2.3

Для более подробного анализа программы установим брейкпойнт на метку _start, с которой начинается выполнение любой ассемблерной программы, и запустим её

```
(gdb) break _start  
Breakpoint 1 at 0x8049000: file lab09-2.asm, line 11.  
(gdb)  
  
(gdb) run  
Starting program: /home/boldyrevadelgir/work/arch-pc/lab09/lab09-2  
  
Breakpoint 1, _start () at lab09-2.asm:11  
11      mov    eax, 4  
(gdb)
```

Рис.4.2.4

Посмотрим дисассимилированный код программы с помощью команды disassemble начиная с метки _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)
```

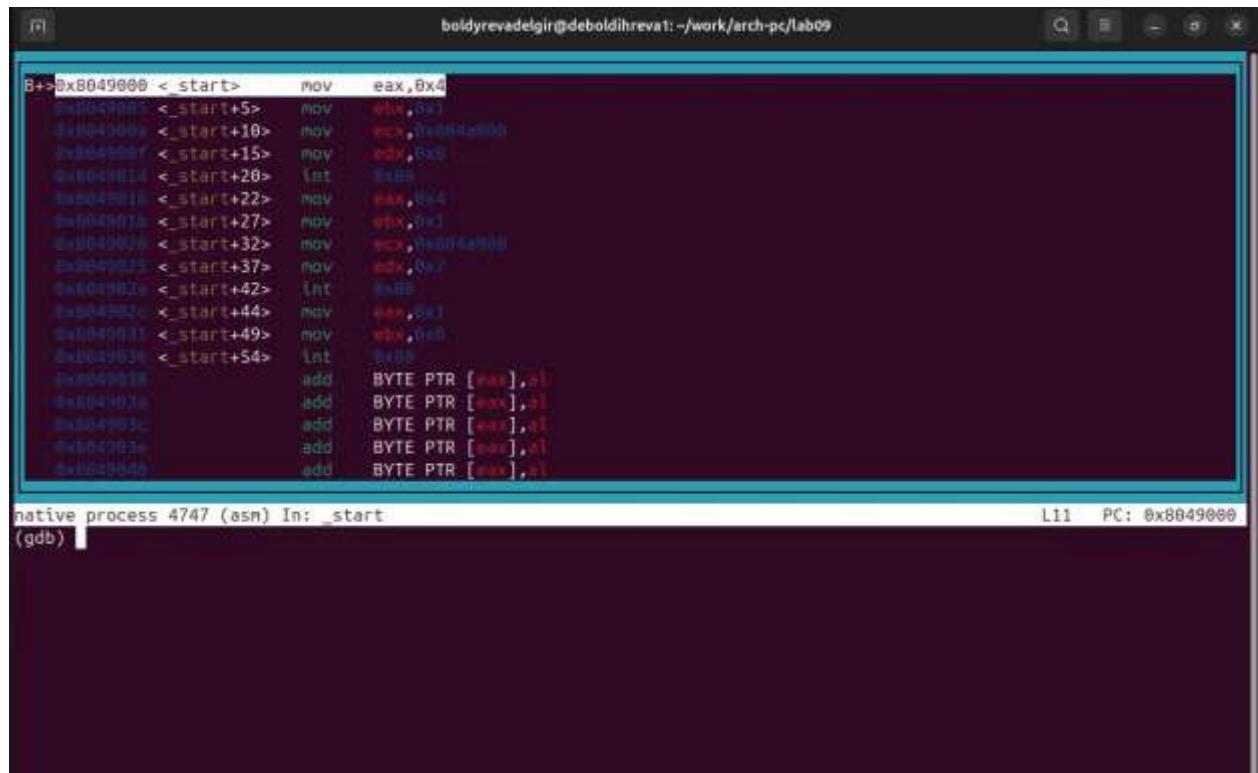
Рис.4.2.5

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

```
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)
```

Рис.4.2.6

Перечислим различия отображения синтаксиса машинных команд в режимах ATT и Intel. Включим режим псевдографики для более удобного анализа Программы



The screenshot shows the GDB debugger interface with the assembly dump of the _start function. The assembly code is displayed in Intel syntax, showing instructions like mov, add, and int. The registers (eax, ebx, ecx, edx) and memory addresses (0x8049000 to 0x8049036) are color-coded. The bottom status bar indicates the native process ID (4747), assembly language (asm), current function (_start), line number (L11), and program counter (PC) address (0x8049000). The command '(gdb)' is visible at the bottom.

Рис.4.2.7

```

Registers group: general
eax      0x0          0          ecx      0x0          0
edx      0x0          0          ebx      0x0          0
esp     0xfffffcf00  0xfffffcf00  ebp      0x0          0x0
esi      0x0          0          edi      0x0          0
eip     0x8049000 <_start>  eflags   0x202        [ IF ]
cs       0x23         35         ss       0x2b         43
ds       0x2b         43         es       0x2b         43
fs       0x0          0          gs       0x0          0

B+>0x8049000 <_start>:    mov    eax,0x4
0x00049003 <_start+5>:    mov    edx,0x1
0x00049007 <_start+10>:   mov    ecx,0x804a000
0x0004900f <_start+15>:   mov    edx,0x0
0x00049011 <_start+20>:   int    $0x80
0x00049012 <_start+22>:   mov    ebx,0x4
0x00049015 <_start+27>:   mov    eax,0x1
0x00049020 <_start+32>:   mov    ecx,0x804a000
0x00049025 <_start+37>:   mov    edx,0x7

native process 4747 {asm} In: _start
(gdb) layout regs
(gdb)

```

Рис.4.2.7

На предыдущих шагах была установлена точка останова по имени метки(`_start`).Про верьте это с помощью команды `info breakpoints(кратко i b)`

```

Registers group: general
eax      0x0          0          ecx      0x0          0
edx      0x0          0          ebx      0x0          0
esp     0xfffffcf00  0xfffffcf00  ebp      0x0          0x0
esi      0x0          0          edi      0x0          0
eip     0x8049000 <_start>  eflags   0x202        [ IF ]
cs       0x23         35         ss       0x2b         43
ds       0x2b         43         es       0x2b         43
fs       0x0          0          gs       0x0          0

B+>0x8049000 <_start>:    mov    eax,0x4
0x00049003 <_start+5>:    mov    edx,0x1
0x00049007 <_start+10>:   mov    ecx,0x804a000
0x0004900f <_start+15>:   mov    edx,0x0
0x00049011 <_start+20>:   int    $0x80
0x00049012 <_start+22>:   mov    ebx,0x4
0x00049015 <_start+27>:   mov    eax,0x1
0x00049020 <_start+32>:   mov    ecx,0x804a000
0x00049025 <_start+37>:   mov    edx,0x7

native process 4747 {asm} In: _start
(gdb) layout regs
(gdb) info breakpoints
Num  Type            Disp Enb Address  What
1   breakpoint      keep y 0x00049000 lab89-2.asm:11
      breakpoint already hit 1 time
(gdb)

```

Рис.4.2.8

Установим еще одну точку останова по адресу инструкции. Адрес инструкции можно увидеть в средней части экрана в левом столбце соответствующей инструкции. Определите адрес предпоследней инструкции (`mov ebx,0x0`) и установим точку останова.

Registers

Register group: general					
eax	0x0	0	ecx	0x0	0
edx	0x0	0	ebx	0x0	0
esp	0xfffffcf00	0xfffffcf00	ebp	0x0	0x8
esi	0x0	0	edi	0x0	0
eip	0x8049000	0x8049000 <_start>	eflags	0x282	[IF]
cs	0x23	35	ss	0x2b	43
ds	0x2b	43	es	0x2b	43
fs	0x0	0	gs	0x0	0
B+>0x8049000 <_start>	mov	eax,0x4			
<start+5>	mov	ebx,0x1			
<start+10>	mov	ecx,0x8049000			
<start+15>	mov	edx,0x0			
<start+20>	int	0x0			
<start+22>	mov	eax,0x1			
<start+27>	mov	ebx,0x1			
<start+32>	mov	ecx,0x8049000			
<start+37>	mov	edx,0x0			

```
native process 4747 (asm) In: _start
L11 PC: 0x8049000
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab09-2.asm:11
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) break *0x8049000
Note: breakpoint 1 also set at pc 0x8049000.
Breakpoint 3 at 0x8049000 : file lab09-2.asm, line 11.
(gdb)
```

Рис.4.2.9

Посмотрим информацию о всех установленных точках останова

```
Breakpoint 3 at 0x8049000 : file lab09-2.asm, line 11.
(gdb) info breakpoints
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab09-2.asm:11
breakpoint already hit 1 time
2 breakpoint keep y <PENDING> 0x8049000
3 breakpoint keep y 0x8049000 lab09-2.asm:11
(gdb)
```

Рис.4.2.10

Посмотрим содержимое регистров также можно с помощью команды info

Registers

Registers

Register group: general					
eax	0x0	0	ecx	0x0	0
edx	0x0	0	ebx	0x0	0
esp	0xfffffcf08	0xfffffcf08	ebp	0x8	0x8
esi	0x0	0	edi	0x0	0
eip	0x8049000	0x8049000 <_start>	eflags	0x282	[IF]
cs	0x23	35	ss	0x2b	43
ds	0x2b	43	es	0x2b	43
fs	0x0	0	gs	0x0	0
B+>0x8049000 <_start>	mov	eax,0x4			
<start+5>	mov	ebx,0x1			
<start+10>	mov	ecx,0x8049000			
<start+15>	mov	edx,0x0			
<start+20>	int	0x0			
<start+22>	mov	eax,0x1			
<start+27>	mov	ebx,0x1			
<start+32>	mov	ecx,0x8049000			
<start+37>	mov	edx,0x0			

```
native process 4747 (asm) In: _start
L11 PC: 0x8049000
eax 0x0 0
ecx 0x0 0
edx 0x0 0
ebx 0x0 0
esp 0xfffffcf08 0xfffffcf08
ebp 0x8 0x8
esi 0x0 0
edi 0x0 0
eip 0x8049000 0x8049000 <_start>
eflags 0x282 [ IF ]
->Type <RET> for more, q to quit, c to continue without paging-
```

Выполним 5 инструкций с помощью команды stepi(или si) и проследите за изменением значений регистров.

```

boldyrevadelgir@deb0ldihrevat:~/work/arch-pc/lab09
Register group: general
eax      0x4          4           ecx      0x0          0
edx      0x0          0           ebx      0x0          0
esp     0xfffffcfd0  0xfffffcfd0  ebp      0x0          0x0
esi      0x0          0           edi      0x0          0
eip     0x8049005   0x8049005 <_start+5> eflags  0x202          [ IF ]
cs       0x23         35          ss       0x2b          43
ds       0x2b         43          es       0x2b          43
fs       0x0          0           gs       0x0          0

B+ < start>    mov    eax,0x1
>0x8049005 < start+5> nov    ebx,0x1
0x0049006 < start+10> nov    ecx,0x1
0x0049007 < start+15> nov    edx,0x1
0x0049008 < start+20> int    0x80
0x0049009 < start+22> nov    eax,0x1
0x004900a < start+27> nov    esp,0x1
0x004900b < start+32> nov    ebx,0x8049005
0x004900c < start+37> nov    edx,0x1

native process 4747 (asm) In: _start
L12   PC: 0x8049005
eip      0x8049000   0x8049000 <_start>
eflags   0x202          [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
cs       0x23         35
ss       0x2b         43
ds       0x2b         43
es       0x2b         43
fs       0x0          0
gs       0x0          0
(gdb) stepi
(gdb)

```

Рис.4.2.11(stepi)

```

boldyrevadelgir@deb0ldihrevat:~/work/arch-pc/lab09
Register group: general
eax      0x4          4           ecx      0x0          0
edx      0x0          0           ebx      0x0          0
esp     0xfffffcfd0  0xfffffcfd0  ebp      0x0          0x0
esi      0x0          0           edi      0x0          0
eip     0x8049005   0x8049005 <_start+5> eflags  0x202          [ IF ]
cs       0x23         35          ss       0x2b          43
ds       0x2b         43          es       0x2b          43
fs       0x0          0           gs       0x0          0

B+ < start>    mov    eax,0x1
>0x8049005 < start+5> nov    ebx,0x1
0x0049006 < start+10> nov    ecx,0x1
0x0049007 < start+15> nov    edx,0x1
0x0049008 < start+20> int    0x80
0x0049009 < start+22> nov    eax,0x1
0x004900a < start+27> nov    esp,0x1
0x004900b < start+32> nov    ebx,0x8049005
0x004900c < start+37> nov    edx,0x1

native process 4747 (asm) In: _start
L12   PC: 0x8049005
eax      0x4          4
ecx      0x0          0
edx      0x0          0
ebx      0x0          0
esp     0xfffffcfd0  0xfffffcfd0
ebp      0x0          0x0
esi      0x0          0
edi      0x0          0
eip     0x8049005   0x8049005 <_start+5>
eflags   0x202          [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис.4.2.12(info registers)

```

boldyrevadelgir@deboldihrevat:~/work/arch-pc/lab09
Register group: general
eax      0x4          4           ecx      0x0          0
edx      0x0          0           ebx      0x1          1
esp      0xfffffcfd8  0xfffffcfd8  ebp      0x0          0x0
esi      0x0          0           edi      0x0          0
eip      0x804900a    0x804900a <_start+10> eflags   0x202          [ IF ]
cs       0x23         35          ss       0x2b         43
ds       0x2b         43          es       0x2b         43
fs       0x0          0           gs       0x0          0

B+ <_start>    mov    eax,0x0
<_start+5>    mov    edx,0x3
>0x804900a <_start+10>  mov    ecx,0x804a000
<_start+15>    mov    edx,0x0
<_start+20>    int    0x80
<_start+22>    mov    eax,0x4
<_start+27>    mov    edx,0x3
<_start+32>    mov    ecx,0x804a000
<_start+37>    mov    edx,0x7

native process 4747 (asm) In: _start
L13   PC: 0x804900a
eip      0x8049005    0x8049005 <_start+5>
eflags   0x202          [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
cs       0x23         35          ss       0x2b         43
ds       0x2b         43          es       0x2b         43
fs       0x0          0           gs       0x0          0
(gdb) stepi
(gdb)

```

Рис.4.2.13(stepi)

```

boldyrevadelgir@deboldihrevat:~/work/arch-pc/lab09
Register group: general
eax      0x4          4           ecx      0x0          0
edx      0x0          0           ebx      0x1          1
esp      0xfffffcfd8  0xfffffcfd8  ebp      0x0          0x0
esi      0x0          0           edi      0x0          0
eip      0x804900a    0x804900a <_start+10> eflags   0x202          [ IF ]
cs       0x23         35          ss       0x2b         43
ds       0x2b         43          es       0x2b         43
fs       0x0          0           gs       0x0          0

B+ <_start>    mov    eax,0x0
<_start+5>    mov    edx,0x3
>0x804900a <_start+10>  mov    ecx,0x804a000
<_start+15>    mov    edx,0x0
<_start+20>    int    0x80
<_start+22>    mov    eax,0x4
<_start+27>    mov    edx,0x3
<_start+32>    mov    ecx,0x804a000
<_start+37>    mov    edx,0x7

native process 4747 (asm) In: _start
L13   PC: 0x804900a
eax      0x4          4
ecx      0x0          0
edx      0x0          0
ebx      0x1          1
esp      0xfffffcfd8  0xfffffcfd8
ebp      0x0          0x0
esi      0x0          0
edi      0x0          0
eip      0x804900a    0x804900a <_start+10>
eflags   0x202          [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис.4.2.14(info registers)

boldyrevadelgir@deboldihrevat:~/work/arch-pc/lab09

Register group: general					
eax	0x4	4	ecx	0x804a000	134520832
edx	0x0	0	ebx	0x1	1
esp	0xfffffcfd0	0xfffffcfd0	ebp	0x0	0x0
esi	0x0	0	edi	0x0	0
eip	0x804900f	0x804900f <_start+15>	eflags	0x202	[IF]
cs	0x23	35	ss	0x2b	43
ds	0x2b	43	es	0x2b	43
fs	0x0	0	gs	0x0	0
B+	<_start>	MOV EAX,0x0			
0x00490000	<_start+5>	MOV EBX,0x0			
0x00490005	<_start+10>	MOV ECX,0x0			
>0x804900f	<_start+15>	MOV EDX,0x8			
0x004900f5	<_start+20>	INT 0x80			
0x004900f8	<_start+22>	MOV EAX,0x0			
0x004900fa	<_start+27>	MOV EBX,0x0			
0x004900fc	<_start+32>	MOV ECX,0x0			
0x004900fe	<_start+37>	MOV EDX,0x7			

```
native process 4747 (asm) In: _start
L14 PC: 0x804900f
(gdb) stepi
(gdb) 
```

Рис.4.2.15(stepi)

boldyrevadelgir@deboldihrevat:~/work/arch-pc/lab09

Register group: general					
eax	0x4	4	ecx	0x804a000	134520832
edx	0x0	0	ebx	0x1	1
esp	0xfffffcfd0	0xfffffcfd0	ebp	0x0	0x0
esi	0x0	0	edi	0x0	0
eip	0x804900f	0x804900f <_start+15>	eflags	0x202	[IF]
cs	0x23	35	ss	0x2b	43
ds	0x2b	43	es	0x2b	43
fs	0x0	0	gs	0x0	0
B+	<_start>	MOV EAX,0x0			
0x00490000	<_start+5>	MOV EBX,0x0			
0x00490005	<_start+10>	MOV ECX,0x0			
>0x804900f	<_start+15>	MOV EDX,0x8			
0x004900f5	<_start+20>	INT 0x80			
0x004900f8	<_start+22>	MOV EAX,0x0			
0x004900fa	<_start+27>	MOV EBX,0x0			
0x004900fc	<_start+32>	MOV ECX,0x0			
0x004900fe	<_start+37>	MOV EDX,0x7			

```
native process 4747 (asm) In: _start
L14 PC: 0x804900f
(gdb) info registers
eax          0x4          4
ecx          0x804a000      134520832
edx          0x0          0
ebx          0x1          1
esp          0xfffffcfd0    0xfffffcfd0
ebp          0x0          0x0
esi          0x0          0
edi          0x0          0
eip          0x804900f      0x804900f <_start+15>
eflags        0x202        [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис.4.2.16(info registers)

```

boldyrevadelgir@deboldihrevat:~/work/arch-pc/lab09
Register group: general
eax 0x4          4          ecx 0x804a000 134520832
edx 0x8          8          ebx 0x1          1
esp 0xfffffcfd0 0xfffffcfd0  ebp 0x0          0x0
esi 0x0          0          edi 0x0          0
eip 0x8049014    0x8049014 <_start+20> eflags 0x202 [ IF ]
cs 0x23         35         ss 0x2b         43
ds 0x2b         43         es 0x2b         43
fs 0x0          0          gs 0x0          0
B+ 0x8049000 <_start>    mov eax,0x0
0x8049004 <_start+5>    mov ebx,0x1
0x8049008 <_start+10>   mov ecx,0x804a000
0x8049012 <_start+15>   mov edx,0x0
>0x8049014 <_start+20> int 0x80
0x8049018 <_start+22>   mov eax,0x4
0x804901c <_start+27>   mov ebx,0x1
0x8049020 <_start+32>   mov ecx,0x804a000
0x8049024 <_start+37>   mov edx,0x0
native process 4747 (asm) In: _start
L15 PC: 0x8049014
eip 0x804900f    0x804900f <_start+15>
eflags 0x202     [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
cs 0x23         35
ss 0x2b         43
ds 0x2b         43
es 0x2b         43
fs 0x0          0
gs 0x0          0
(gdb) stepi
(gdb) 

```

Рис.4.2.17(stepi)

```

boldyrevadelgir@deboldihrevat:~/work/arch-pc/lab09
Register group: general
eax 0x4          4          ecx 0x804a000 134520832
edx 0x8          8          ebx 0x1          1
esp 0xfffffcfd0 0xfffffcfd0  ebp 0x0          0x0
esi 0x0          0          edi 0x0          0
eip 0x8049014    0x8049014 <_start+20> eflags 0x202 [ IF ]
cs 0x23         35         ss 0x2b         43
ds 0x2b         43         es 0x2b         43
fs 0x0          0          gs 0x0          0
B+ 0x8049000 <_start>    mov eax,0x0
0x8049004 <_start+5>    mov ebx,0x1
0x8049008 <_start+10>   mov ecx,0x804a000
0x8049012 <_start+15>   mov edx,0x0
>0x8049014 <_start+20> int 0x80
0x8049018 <_start+22>   mov eax,0x4
0x804901c <_start+27>   mov ebx,0x1
0x8049020 <_start+32>   mov ecx,0x804a000
0x8049024 <_start+37>   mov edx,0x0
native process 4747 (asm) In: _start
L15 PC: 0x8049014
eax 0x4          4
ecx 0x804a000 134520832
edx 0x8          8
ebx 0x1          1
esp 0xfffffcfd0 0xfffffcfd0
ebp 0x0          0x0
esi 0x0          0
edi 0x0          0
eip 0x8049014    0x8049014 <_start+20>
eflags 0x202     [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис.4.2.18(info registers)

При выполнении команды stepi (одной инструкции процессора) обычно изменяются:

1. RIP/EIP -указатель на следующую инструкцию (всегда меняется)
2. RAX/EAX -часто используется для вычислений и возврата значений
3. RSP/ESP -указатель стека (меняется при push/pop)
4. Флаги (ZF, CF, SF, OF) -после арифметических операций

Посмотрим значение переменной msg1 по имени

```
(gdb) x/1sb &msg1
Breakpoint 1 at 0x804a008: <msg1>: "Hello, "
(gdb) █
```

Рис.4.2.19

Посмотрим значение переменной msg2 по адресу

```
(gdb) x/1sb 0x804a008
Breakpoint 1 at 0x804a008: <msg2>: "world!\n\x034"
(gdb) █
```

Рис.4.2.20

Изменим первый символ переменной msg1

```
(gdb) x/1sb 0x804a008
Breakpoint 1 at 0x804a008: <msg2>: "world!\n\x034"
(gdb) set {char}&msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) set {char}&msg1='h'
A syntax error in expression, near '&msg1='h'.
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
Breakpoint 1 at 0x804a008: <msg1>: "hello, "
(gdb) █
```

Рис.4.2.21

Заменим любой символ во второй переменной msg2

```
(gdb) set {char}&msg2='u'
(gdb) x/1sb &msg2
Breakpoint 1 at 0x804a008: <msg2>: "uorld!\n\x034"
(gdb) █
```

Рис.4.2.22

Выведем в различных форматах (в шестнадцатеричном формате, вдвоичном формате и в символьном виде) значение регистра edx.

```
(gdb) p/s $edx
$1 = $0
(gdb) p/t $edx
$2 = 10000000010010100000000000000000
(gdb) p/s $edx
$3 = 8
(gdb) p/t $edx
$4 = 1000
(gdb) p/x $edx
$5 = 0x8
(gdb) █
```

С помощью команды set изменим значение регистра ebx

```
(gdb) p/t $edx  
$d = 1000  
(gdb) p/x $edx  
$d = 0x8  
(gdb) set $ebx='2'  
(gdb) p/s $ebx  
$6 = 50  
(gdb) set $ebx=2  
(gdb) p/s $ebx  
$7 = 2  
(gdb)
```

Рис.4.2.23

Завершим выполнение программы с помощью команды continue

(сокращенно c) или stepi (сокращенно si) и выйдите из GDB с помощью команды quit(сокращенно q)

```
(gdb) c  
Continuing.  
hello, world!  
[Inferior 1 (process 4747) exited normally]  
(gdb)
```

Рис.4.2.24

4.3 Обработка аргументов командной строки в GDB

Скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm

```
boldyrevadelgir@deb0ldihrevat:~/work/arch-pc/lab09  
boldyrevadelgir@deb0ldihrevat:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-2.asm ~/work/arch-pc/lab09/lab09-3.asm
```

Рис.4.3.1

Создадим исполняемый файл

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

Рис.4.3.2

Загрузим исполняемый файл в отладчик, указав аргументы

```
boldyrevadelgir@boldthreaval:~/work/arch-pc/lab09$ gdb --args lab09-3 аргумент1 аргумент2 'аргумент3'
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 lab09-3...
(gdb)
```

Рис.4.3.3

Установим точку останова перед первой инструкцией в программе и запустим ее.

```
(gdb) b _start
Breakpoint 1 at 0x400000: file lab09-3.asm, line 7.
(gdb) ■
```

Рис.4.3.4

```
(gdb) run
Starting program: /home/boldyrevadelgir/work/arch-pc/lab09/lab09-3 аргумент1 аргумент2 аргумент3

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 lab09-3.asm:7
7      pop    ecx
(gdb) ■
```

Рис.4.3.5

Посмотрим 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 и т.д

```
(gdb) x/x $esp
00000000: 0x00000005
(gdb) x/s *(void**)(\$esp+4)
00000004: "/home/boldyrevadelgir/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(\$esp+8)
00000008: "аргумент1"
(gdb) x/s *(void**)(\$esp+12)
0000000c: "аргумент"
(gdb) x/s *(void**)(\$esp+16)
00000010: "аргумент"
(gdb) x/s *(void**)(\$esp+20)
00000014: "аргумент 3"
(gdb) x/s *(void**)(\$esp+24)
00000018: <error: Cannot access memory at address 0x0>
(gdb) █
```

Рис.4.3.6

Каждый элемент этого массива — это адрес (указатель), который в 32-битной системе занимает 4 байта. Поэтому чтобы перейти, нужно прибавить 4 байта.

5 Задание для самостоятельной работы

Преобразуем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму

```
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$ cp ~/work/arch-pc/lab08/lab8-4.asm ~/work/arch-pc/lab09/lab09-4.asm
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$
```

Рис.5.1


```
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$ lab09-4.asm
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$
```

Screenshot of a terminal window showing the command to copy lab8-4.asm to lab09-4.asm.

The terminal window has three tabs:

- lab09-1.asm
- lab09-2.asm
- lab09-4.asm (active tab)

The content of the lab09-4.asm tab is:

```
%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x)=10x-5", 0
msg_res db "Результат: ", 0

SECTION .bss
x resd 1

SECTION .text
global _start

_start:
    pop ecx          ; argc
    pop edx          ; argv[0] (имя программы)
    sub ecx, 1        ; количество аргументов
    mov ebx, 0         ; накопитель суммы

next:
    cmp ecx, 0
    jz end

    pop eax          ; берём очередной аргумент
    call atoi          ; преобразуем в число
    mov [x], eax        ; сохраним x
```

Рис.5.2



```
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$ nasm -f elf lab09-4.asm
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-4 lab09-4.o
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$ ./lab09-4 3 0 1 2
Функция: f(x)=10x-5
Результат: 40
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$ ./lab09-4 2 3 4
Функция: f(x)=10x-5
Результат: 75
boldyrevadelgir@deboldihrevai:~/work/arch-pc/lab09$
```

Рис.5.3

В листинге приведена программа вычисления выражения

$(3+2)*4+5$. При запуске данная программа дает неверный результат. Проверьте это.

С помощью отладчика GDB, анализируя изменения значений регистров, определите ошибку и исправьте ее

```
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab09$ touch lab09-5
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab09$ gedit lab09-5
```

Рис.5.4

```
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab09$ nasm -f elf -g lab09-5.asm -o lab09-5.o
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab09$ ld -m elf_i386 lab09-5.o -o lab09-5
boldyrevadelgir@deboldihreval:~/work/arch-pc/lab09$ gdb lab09-5
GNU gdb (Ubuntu 15.0.50.20240403-Ubuntu1) 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 lab09-5...
```

Рис.5.5

```
(gdb) break _start
Breakpoint 1 at 0x400000: file lab09-5.asm, line 10.
(gdb)
```

Рис.5.6

```
(gdb) run
Starting program: /home/boldyrevadelgir/work/arch-pc/lab09/lab09-5

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 0xE7fffc000

Breakpoint 1, _start () at lab09-5.asm:10
10      mov ebx, 3
(gdb)
```

Рис.5.7

```
(gdb) print $eax
$1 = 0
(gdb) continue
Continuing.
Результат: 8
[Inferior 1 (process 5983) exited normally]
(gdb) 
```

Рис.5.8

The screenshot shows a terminal window with a GDB session. The assembly code in the middle pane is:

```

lab09-5.asm:
B* 10 mov ebx, 3
    11 mov eax, 2
    12 add ebx, eax
    13 mov ecx, 4
    14 mul ecx
> 15 add ebx, 5
    16 mov edi, eax
    17

```

The registers pane at the top shows:

Register group: general					
eax	0x8	8	ecx	0x4	4
ebx	0x0	0	ebx	0x5	5
esp	0xfffffd000	0xfffffd000	ebp	0x0	0x0
esi	0x0	0	edi	0x0	0
ebp	0x80490fb	0x80490fb < _start+19>	eflags	0x202	[IF]
cs	0x23	35	ss	0x2b	43
ds	0x2b	43	es	0x2b	43
fs	0x0	0	gs	0x0	0

At the bottom, the command line shows:

```

native process 6035 (src) In: _start
Starting program: /home/boldyrevadelgir/work/arch-pc/lab09/lab09-5
Breakpoint 1, _start () at lab09-5.asm:10
(gdb) layout regs
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)

```

Рис.5.9

The screenshot shows a code editor with four tabs open, all containing assembly code for the same program. The tabs are:

- lab09-1.asm
- lab09-2.asm
- lab09-4.asm
- lab09-5.asm

The assembly code in the lab09-5.asm tab is:

```

%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 ebx, 5
    mov edi, eax

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

    call quit

```

Рис.5.10

```
boldyrevadelgir@deb0ldihrevali:/work/arch-pc/lab09$ nasm -f elf lab09-5.asm
boldyrevadelgir@deb0ldihrevali:/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-5 lab09-5.o
boldyrevadelgir@deb0ldihrevali:/work/arch-pc/lab09$ ./lab09-5
Результат: 20
boldyrevadelgir@deb0ldihrevali:/work/arch-pc/lab09$
```

Рис.5.11

6 Выводы

Я приобрела навыки написания программ с использованием подпрограмм.

Ознакомилась с методами отладки при помощи GDB и его основными возможностями

Список литературы

1. GDB:TheGNUProjectDebugger.—URL:<https://www.gnu.org/software/gdb/>.
2. GNUBashManual.—2016.—URL:<https://www.gnu.org/software/bash/manual/>.
3. Midnight CommanderDevelopment Center.—2021.—URL: <https://midnightcommander.org/>.
4. NASMAssemblyLanguageTutorials.—2021.—URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 c. —(In a Nutshell). —ISBN 0596009658.—URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. RobbinsA. Bash Pocket Reference.—O'Reilly Media,2016.—156 c.—ISBN 978-1491941591.
7. TheNASMdocumentation.—2021.—URL:<https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash.—Packt Publishing,2017.—502 c.—ISBN 9781784396879.

