

Отчёт по лабораторной работе №9

Дисциплина: Архитектура компьютера

Зайцева Ульяна Владимировна

Содержание

1	Цель работы	1
2	Задание	1
3	Теоретическое введение.....	1
4	Выполнение лабораторной работы.....	2
5	Выводы.....	16

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. Добавление точек останова.
4. Работа с данными программы в GDB.
5. Обработка аргументов командной строки в GDB.
6. Задания для самостоятельной 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.

Наиболее часто применяют следующие методы отладки: • создание точек контроля значений на входе и выходе участка программы (например, вывод промежуточных значений на экран — так называемые диагностические сообщения); • использование специальных программ-отладчиков. Отладчики позволяют управлять ходом выполнения программы, контролировать и изменять данные. Это

помогает быстрее найти место ошибки в программе и ускорить её исправление. Наиболее популярные способы работы с отладчиком — это использование точек останова и выполнение программы по шагам. Пошаговое выполнение — это выполнение программы с остановкой после каждой строки, чтобы программист мог проверить значения переменных и выполнить другие действия. Точки останова — это специально отмеченные места в программе, в которых программа-отладчик приостанавливает выполнение программы и ждёт команд. Наиболее популярные виды точек останова: • Breakpoint — точка останова (остановка происходит, когда выполнение доходит до определённой строки, адреса или процедуры, отмеченной программистом); • Watchpoint — точка просмотра (выполнение программы приостанавливается, если программа обратилась к определённой переменной: либо считала её значение, либо изменила его)

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; • изменить программу так, чтобы можно было поэкспериментировать с устранением эффектов одной ошибки и продолжить выявление других

Подпрограмма — это, как правило, функционально законченный участок кода, который можно многократно вызывать из разных мест программы. В отличие от простых переходов из подпрограмм существует возврат на команду, следующую за вызовом.

4 Выполнение лабораторной работы

1. Реализация подпрограмм в NASM.

Создаю каталог для выполнения лабораторной работы №9, перейдите в него и создаю файл lab9-1.asm(рис. ??).

```
ulyanazaitseva@fedora:~/work/study/2023-2024/Архитектура ...  
[ulyanazaitseva@fedora ~]$ mkdir ~/work/study/2023-2024/'Архитектура компьютера'/arch-pc/lab09  
[ulyanazaitseva@fedora ~]$ cd ~/work/study/2023-2024/'Архитектура компьютера'/arch-pc/lab09  
[ulyanazaitseva@fedora lab09]$ touchlab09-1.asm  
bash: touchlab09-1.asm: команда не найдена...  
[ulyanazaitseva@fedora lab09]$ touch lab9-1.asm  
[ulyanazaitseva@fedora lab09]$
```

Создание каталога и файла

Ввожу в файл lab9-1.asm текст программы с использованием подпрограммы из листинга 9.1. (рис. ??)

```
lab9-1.asm  
~/.work/study/2023-2024/Архитектура компьютера/arch-pc/lab09  
Л08_Зайцева_отчет.md lab9-1.asm  
%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 iprintf  
call quit  
;-----  
; Подпрограмма вычисления  
; выражения "2x+7"  
_calcul:  
mov ebx, 2  
mul ebx  
add eax, 7  
mov [res], eax  
ret ; выход из подпрограммы
```

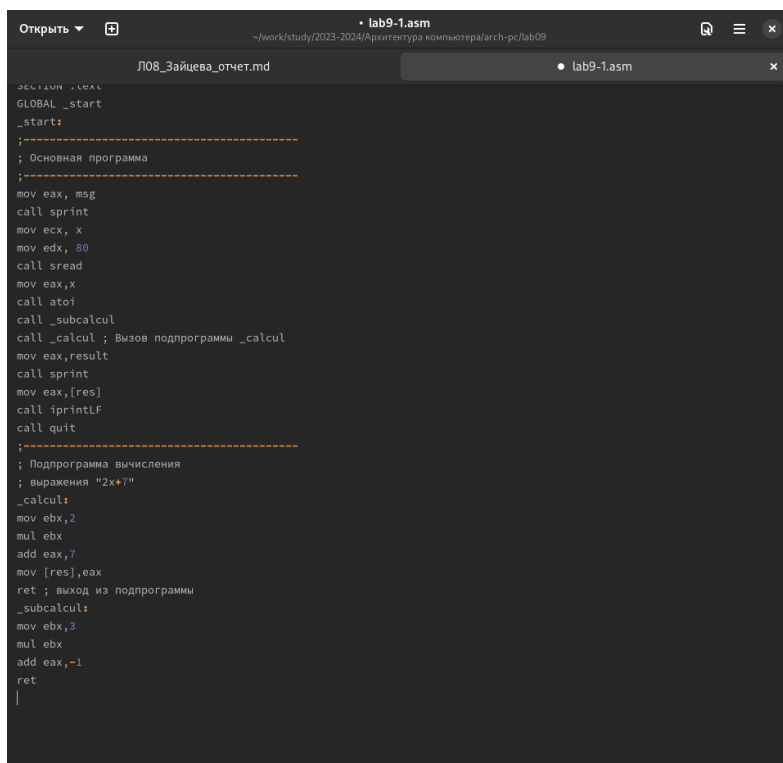
Ввод листинга 9.1

Создаю исполняемый файл и проверяю его работу(рис. ??)

```
[ulyanazaitseva@fedora lab09]$ nasm -f elf lab9-1.asm
[ulyanazaitseva@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[ulyanazaitseva@fedora lab09]$ ./lab9-1
Введите x: 7
2x+7=21
[ulyanazaitseva@fedora lab09]$
```

Проверка файла

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



```

section .text
GLOBAL _start
_start:
; Основная программа
;
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
call _subcalcul
call _calcul ; Вызов подпрограммы _calcul
mov eax, result
call sprintf
mov eax, [res]
call iprintf
call quit
;
; Подпрограмма вычисления
; выражения "2x+7"
_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx, 3
mul ebx
add eax, -1
ret
|

```

Изменяю текст программы

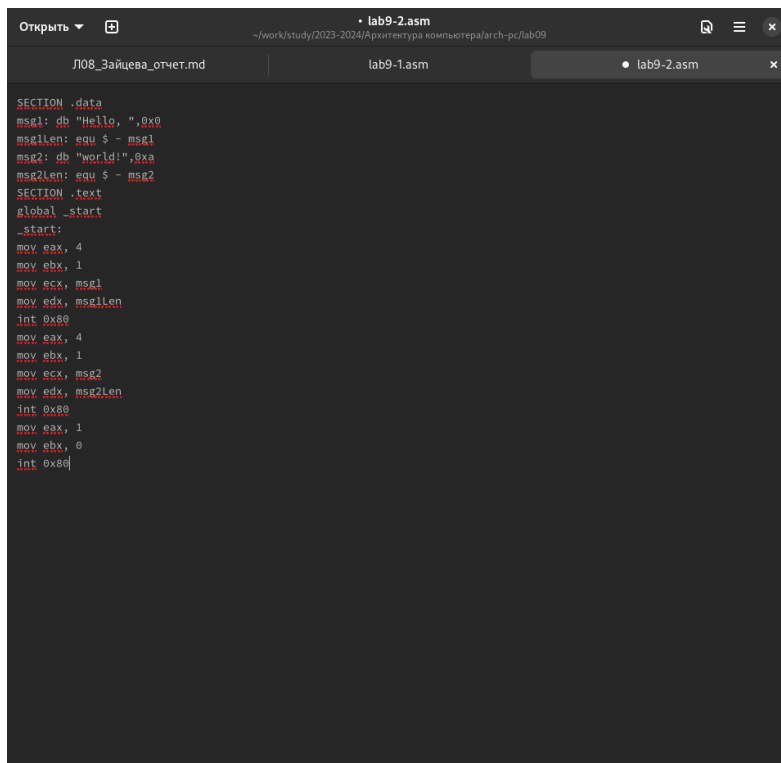
Создаю файл и проверяю его работу(рис. ??)

```
[ulyanazaitseva@fedora lab09]$ nasm -f elf lab9-1.asm
[ulyanazaitseva@fedora lab09]$ ld -m elf_i386 -o lab9-1 lab9-1.o
[ulyanazaitseva@fedora lab09]$ ./lab9-1
Введите x: 7
2x+7=47
[ulyanazaitseva@fedora lab09]$
```

Проверка работы файла

2. Отладка программ с помощью GDB.

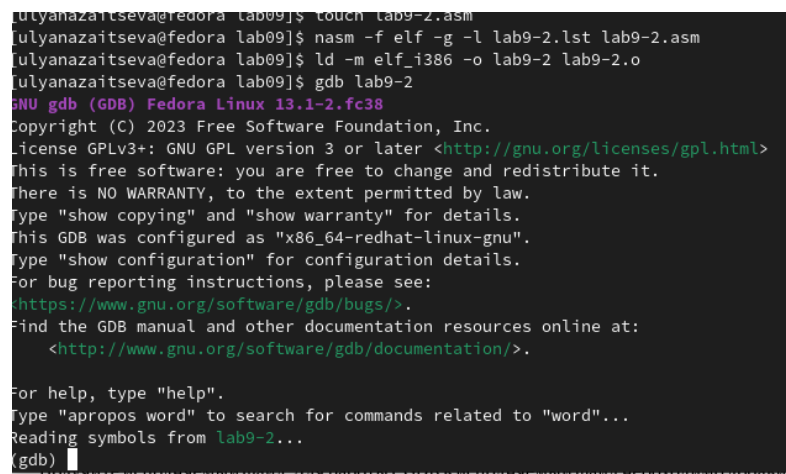
Создаю файл `lab9-2.asm` с текстом программы из Листинга 9.2. (рис. ??)



```
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
```

Листинг 9.2

Получаю исполняемый файл для работы с GDB с ключом '-g'.(рис. ??)



```
lulyanazaitseva@fedora lab09]$ touch lab9-2.asm
lulyanazaitseva@fedora lab09]$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
lulyanazaitseva@fedora lab09]$ ld -m elf_i386 -o lab9-2 lab9-2.o
lulyanazaitseva@fedora lab09]$ gdb lab9-2
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
Copyright (C) 2023 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)
```

Создаю файл и загружаю в отладчик gdb

Проверяю работу программы, запустив ее в gdb с помощью команды run(рис. ??)

```

Type "apropos word" to search for commands related to "word"...
Reading symbols from lab9-2...
(gdb) run
Starting program: /home/ulyanazaitseva/work/study/2023-2024/Архитектура компьютера/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]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.
Hello, world!
[Inferior 1 (process 4296) exited normally]
(gdb)

```

Проверка работы

Для более подробного анализа программы устанавливаю брейкпоинт на метку `_start` и запускаю её(рис. ??)

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/ulyanazaitseva/work/study/2023-2024/Архитектура компьютера/c/lab09/lab9-2

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

```

Подробный анализ программы

Просматриваем дисассимилированный код программы с помощью команды `disassemble`, начиная с метки `_start`, и переключаемся на отображение команд с синтаксисом Intel, введя команду `set disassembly-flavor intel`(рис. ??)

```
ulyanazaitseva@fedora:~/work/study/2023-2024/Архитектура компьюте...
(gdb) run
Starting program: /home/ulyanazaitseva/work/study/2023-2024/Архитектура компьюте.../arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>: mov $0x4,%eax
0x08049005 <+5>: mov $0x1,%ebx
0x0804900a <+10>: mov $0x804a000,%ecx
0x0804900f <+15>: mov $0x8,%edx
0x08049014 <+20>: int $0x80
0x08049016 <+22>: mov $0x4,%eax
0x0804901b <+27>: mov $0x1,%ebx
0x08049020 <+32>: mov $0x804a000,%ecx
0x08049025 <+37>: mov $0x7,%edx
0x0804902a <+42>: int $0x80
0x0804902c <+44>: mov $0x1,%eax
0x08049031 <+49>: mov $0x0,%ebx
0x08049036 <+54>: int $0x80
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>: mov eax,0x4
0x08049005 <+5>: mov ebx,0x1
0x0804900a <+10>: mov ecx,0x804a000
0x0804900f <+15>: mov edx,0x8
0x08049014 <+20>: int 0x80
0x08049016 <+22>: mov eax,0x4
0x0804901b <+27>: mov ebx,0x1
0x08049020 <+32>: mov ecx,0x804a000
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)
```

Дисассимилированный код программы

Включаю режим псевдографики для более удобного анализа программы с помощью команд layout asm и layout regs(забыла сделать скрин((()

3. Добавление точек останова.

Проверяю, что точка останова по имени метки _start установлена с помощью команды i b и устанавливаю еще одну точку останова. Просматриваю информацию о всех установленных точках останова.(рис. ??)

```
ulyanazaitseva@fedora:~/work/study/2023-2024/Архитектура компьюте... [ Register Values Unavailable ]

B-> 0x8049000 <_start> mov eax,0x4
b+ 0x8049005 <_start+5> mov ebx,0x1
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0x8049031 <_start+49> mov ebx,0x0

native process 4334 In: _start L9 PC: 0x8049000
1 breakpoint keep y 0x8049000 lab9-2.asm:9
breakpoint already hit 1 time
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab9-2.asm:9
breakpoint already hit 1 time
(gdb) b *0x8049005
Breakpoint 2 at 0x8049005: file lab9-2.asm, line 10.
(gdb) i b
Num Type Disp Enb Address What
1 breakpoint keep y 0x8049000 lab9-2.asm:9
breakpoint already hit 1 time
2 breakpoint keep y 0x8049005 lab9-2.asm:10
(gdb)
```

Новая точка останова

4. Работа с данными программы в GDB.

Выполняю инструкции с помощью команды `stepi(si)` и слежу за изменением значений регистров. (рис. ??)(рис. ??)


```
ulyanazaitseva@fedora:~/work/study/2023-2024/Архитектура компьюте... [ Register Values Unavailable ]  
B-> 0x8049000 <_start> mov eax,0x4  
b+ 0x8049005 <_start+5> mov ebx,0x1  
0x804900a <_start+10> mov ecx,0x804a000  
0x804900f <_start+15> mov edx,0x8  
0x8049014 <_start+20> int 0x80  
0x8049016 <_start+22> mov eax,0x4  
0x804901b <_start+27> mov ebx,0x1  
0x8049020 <_start+32> mov ecx,0x804a008  
0x8049025 <_start+37> mov edx,0x7  
0x804902a <_start+42> int 0x80  
0x804902c <_start+44> mov eax,0x1  
0x8049031 <_start+49> mov ebx,0x0  
native process 4334 In: _start L9 PC: 0x8049000  
eax 0x0 0  
ecx 0x0 0  
edx 0x0 0  
ebx 0x0 0  
esp 0xffffd0e0 0xffffd0e0  
ebp 0x0 0x0  
esi 0x0 0  
edi 0x0 0  
eip 0x8049000 0x8049000 <_start>  
eflags 0x202 [ IF ]  
cs 0x23 35  
ss 0x2b 43  
--Type <RET> for more, q to quit, c to continue without paging--
```

Команда і r

```

ulianazaitseva@fedora:~/work/study/2023-2024/Архитектура компьюте...
--Register group: general
eax      0x4      4
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffd0e0 0xffffd0e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049005 0x8049005 <_start+5>
eflags   0x10202  [ IF RF ]
cs       0x23     35

B+ 0x8049000 <_start> mov eax,0x4
B+ 0x8049005 <_start+5> mov ebx,0x1
0x804900a <_start+10> mov ecx,0x804a000
0x804900f <_start+15> mov edx,0x8
0x8049014 <_start+20> int 0x80
0x8049016 <_start+22> mov eax,0x4
0x804901b <_start+27> mov ebx,0x1
0x8049020 <_start+32> mov ecx,0x804a008
0x8049025 <_start+37> mov edx,0x7
0x804902a <_start+42> int 0x80
0x804902c <_start+44> mov eax,0x1
0x8049031 <_start+49> mov ebx,0x0

native process 4334 In: _start L10 PC: 0x8049005
eip      0x8049000 0x8049000 <_start>
eflags   0x202     [ IF ]
cs       0x23     35
ss       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--ccsds 0x2b
43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si 5

Breakpoint 2, _start () at lab9-2.asm:10
(gdb)

```

Инструкции si

Изменились значения регистров eax, ecx, edx и ebx.

Просматриваю значение переменной msg1 по имени с помощью команды x/1sb &msg1 и значение переменной msg2 по ее адресу. С помощью команды set изменяю первый символ переменной msg1 и заменяю первый символ в переменной msg2 на b.(рис. ??)

```

(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='b'
(gdb) x/1sb $msg2
Value can't be converted to integer.
(gdb) x/1sb &msg2
0x804a008 <msg2>: "borld!\n\034"

```

Переменные msg1 и msg2

Вывожу в шестнадцатеричном формате, в двоичном формате и в символьном виде соответственно значение регистра edx с помощью команды print p/F \$val.(рис. ??)

```

0x804a008 <msg2>: "world!\n\034"
(gdb) p/x $edx
$1 = 0x0
(gdb) p/t $edx
$2 = 0
(gdb) p/c $edx
$3 = 0 '\000'
(gdb) p/f $val
$4 = void
(gdb)

```

Значение регистра edx

С помощью команды set изменяю значение регистра ebx (рис. ??)

```

(gdb) set $ebx='2'
(gdb) p/s $ebx
$5 = 50
(gdb) set $ebx=2
(gdb) p/s $ebx
$6 = 2
(gdb)

```

Изменяю значение регистра ebx

Разница вывода команд p/s \$ebx состоит в том, что в первом случае мы переводим символ в его строковый вид, а во втором случае число в строковом виде не изменяется.

Выхожу из gdb с помощью команды quit. (рис. ??)

```

$6 = 2
(gdb) q
A debugging session is active.

    Inferior 1 [process 4334] will be killed.

Quit anyway? (y or n)

```

Выход из gdb

5. Обработка аргументов командной строки в GDB.

Копирую файл lab8-2.asm с программой из листинга 8.2 в файл lab9-3.asm и создаю исполняемый файл. Загружаю исполняемый файл в отладчик gdb, указывая необходимые аргументы с использованием ключа -args. (рис. ??)

```
[ulyanazaitseva@fedora lab09]$ touch lab9-3.asm
[ulyanazaitseva@fedora lab09]$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
[ulyanazaitseva@fedora lab09]$ ld -m elf_i386 -o lab9-3 lab9-3.o
[ulyanazaitseva@fedora lab09]$ gdb --args lab9-3 аргумент1 аргумент2 'аргумент 3'
GNU gdb (GDB) Fedora Linux 13.1-2.fc38
Copyright (C) 2023 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)
```

Создание файла и загрузка в gdb

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее.(рис. ??)

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 8.
(gdb) run
Starting program: /home/ulyanazaitseva/work/study/2023-2024/Архитектура компьютера/arch-pc/c/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]) n
Debuginfod has been disabled.
To make this setting permanent, add 'set debuginfod enabled off' to .gdbinit.

Breakpoint 1, _start () at lab9-3.asm:8
8      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb)
```

Установка точки останова и запуск

Посматриваю позиции стека по их адресам.(рис. ??)

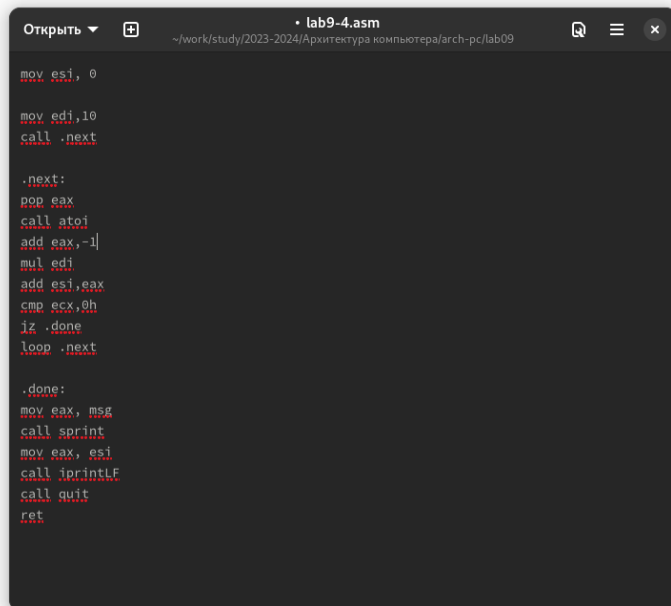
```
(gdb) x/x $esp
0xffffd090: 0x00000005
(gdb) x/s *(void**)(esp+4)
0xffffd23a: "/home/ulyanazaitseva/work/study/2023-2024/Архитектура компьютера/arch-pc/c/lab09/lab9-3"
(gdb) x/s *(void**)(esp+8)
0xffffd2a5: "аргумент1"
(gdb) x/s *(void**)(esp+12)
0xffffd2b7: "аргумент"
(gdb) x/s *(void**)(esp+16)
0xffffd2c8: "2"
(gdb) x/s *(void**)(esp+20)
0xffffd2ca: "аргумент 3"
(gdb) x/s *(void**)(esp+24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Позиции стека

Шаг изменения адреса равен 4, т.к количество аргументов командной строки равно 4.

6. Задания для самостоятельной работы.

- 1) Преобразовываю программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму. в новом файле lab9-4.asm(рис. ??)



```
Открыть ▾ + lab9-4.asm ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09

mov esi, 0

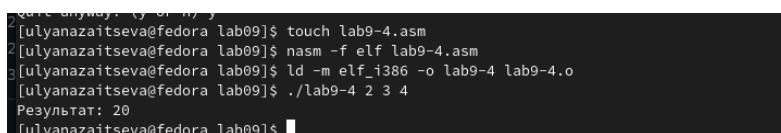
mov edi, 10
call .next

.next:
pop eax
call atoi
add eax, -1
mul edi
add esi, eax
cmp ecx, 0h
jz .done
loop .next

.done:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
ret
```

Преобразование программы из прошлой лабораторной работы

Проверяю, что она работает корректно.(рис. ??)



```
[ulyanazaitseva@fedora lab09]$ touch lab9-4.asm
2 [ulyanazaitseva@fedora lab09]$ nasm -f elf lab9-4.asm
3 [ulyanazaitseva@fedora lab09]$ ld -m elf_i386 -o lab9-4 lab9-4.o
[ulyanazaitseva@fedora lab09]$ ./lab9-4 2 3 4
Результат: 20
[ulyanazaitseva@fedora lab09]$
```

Проверка

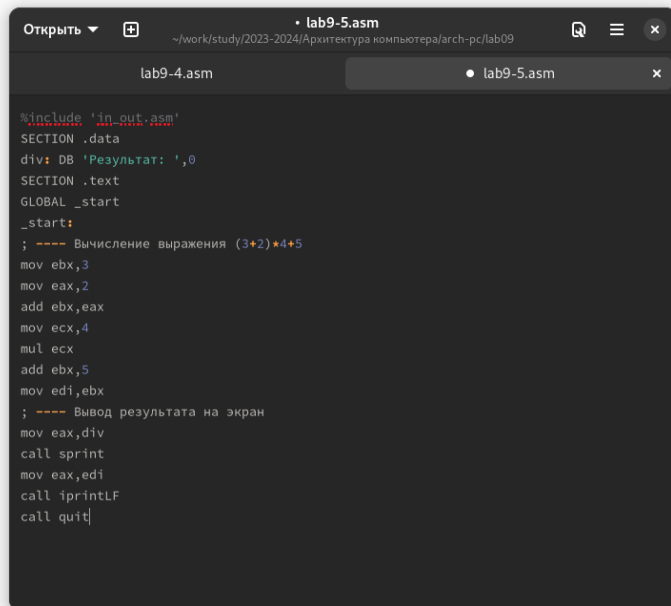
Код: ; 10(x-1) %include 'in_out.asm' SECTION .data msg db "Результат:",0 SECTION .text
global _start

_start: pop ecx pop edx sub ecx,1 mov esi, 0 mov edi,10 call .next

.next: pop eax call atoi add eax,-1 mul edi add esi,eax cmp ecx,0h jz .done loop .next

.done: mov eax, msg call sprint mov eax, esi call iprintLF call quit ret

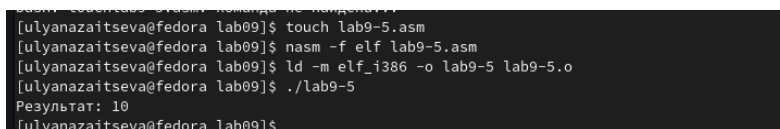
2) Ввожу в файл lab9-5.asm текст программы из листинга 9.3.(рис. ??)



```
%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 ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Листинг 9.3

При правильной работе программа должна выводить 25. Создаю исполняемый файл и запускаю его.(рис. ??)



```
[ulyanazaitseva@fedora lab09]$ touch lab9-5.asm
[ulyanazaitseva@fedora lab09]$ nasm -f elf lab9-5.asm
[ulyanazaitseva@fedora lab09]$ ld -m elf_i386 -o lab9-5 lab9-5.o
[ulyanazaitseva@fedora lab09]$ ./lab9-5
Результат: 10
[ulyanazaitseva@fedora lab09]$
```

Выполнение программы

В выводе получаем неправильный ответ. Получаю исполняемый файл для работы с gdb, запускаю его и ставлю брейкпоинты для каждой инструкции, связанной с вычислениями. Прохожусь по каждому брейкпоинту и слежу за изменениями значений регистров. При выполнении инструкции `mul ecx` происходит умножение `ecx(4)` на `eax(2)`, вместо умножения 4 на 5 (`ebx`). Это происходит потому что стоящая перед `mov ecx,4` инструкция `add ebx,eax` не связана с `mul ecx`, но связана инструкция `mov eax,2`.(рис. ??)

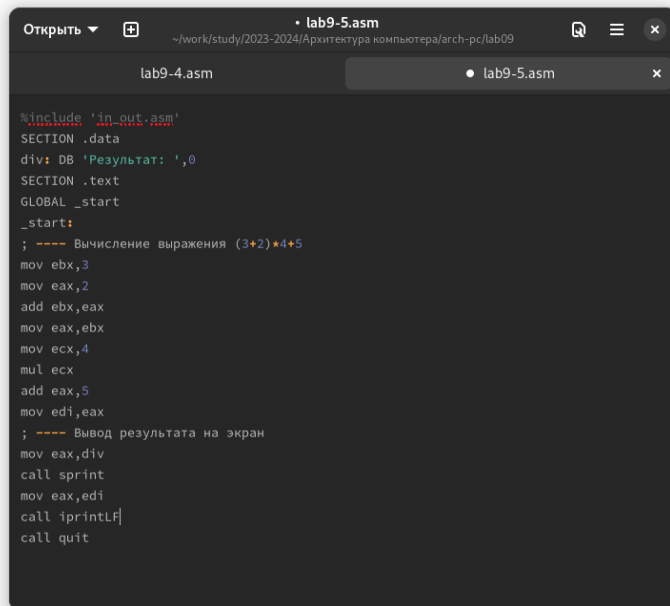
```
ulyanazaitseva@fedora:~/work/study/2023-2024/Архитектура компьюте...
--Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0x5      5
esp      0xffffd0e0 0xffffd0e0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490fb 0x80490fb <_start+19>
eflags   0x10202   [ IF RF ]
cs       0x23     35

B+ 0x80490e8 <_start>      mov     ebx,0x3
0x80490ed <_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,0x804a000
0x8049105 <_start+29>     call    0x804900f <sprint>
0x804910a <_start+34>     mov     eax,edi
0x804910c <_start+36>     call    0x8049086 <iprintLF>
0x8049111 <_start+41>     call    0x80490db <quit>

native process 5743 In: _start L?? PC: 0x80490fb
edi      0x0      0
eip      0x80490e8 0x80490e8 <_start>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
--Type <RET> for more, q to quit, c to continue without paging--ccsds 0x2b
43
es       0x2b     43
fs       0x0      0
gs       0x0      0
(gdb) si 5
0x80490fb in _start ()
(gdb) |
```

Поиск сбоя

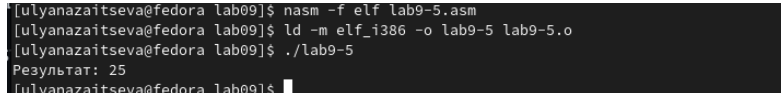
Исправляю ошибку, добавляя после add ebx,eax mov eax,ebx и заменяя ebx на eax в инструкциях add ebx,5 и mov edi,ebx. (рис. ??)



```
%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 ebx,eax
mov 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
```

Исправляю ошибку

Запускаю код и проверяю, что программа работает корректно.(рис. ??)



```
[ulyanazaitseva@fedora lab09]$ nasm -f elf lab9-5.asm
[ulyanazaitseva@fedora lab09]$ ld -m elf_i386 -o lab9-5 lab9-5.o
[ulyanazaitseva@fedora lab09]$ ./lab9-5
Результат: 25
[ulyanazaitseva@fedora lab09]$
```

Исправленный код

Код:

```
%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 ebx,eax mov
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
```

5 Выводы

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