

Отчет по лабораторной работе №9

Дисциплина: архитектура компьютера

Ермакова Анастасия Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Реализация подпрограмм в NASM	9
4.2	Отладка программ с помощью GDB	13
4.2.1	Добавление точек останова	16
4.2.2	Работа с данными программы в GDB	17
4.2.3	Обработка аргументов командной строки в GDB	19
4.3	Выполнение заданий для самостоятельной работы	20
5	Выводы	26
6	Список литературы	27

Список иллюстраций

4.1	Создание каталога и файла	9
4.2	Текст программы файла lab9-1.asm	10
4.3	Результат работы программы	10
4.4	Измененный текст программы	11
4.5	Результат работы программы	13
4.6	Создание файла lab9-2.asm	13
4.7	Текст программы файла lab9-2.asm	14
4.8	Запуск программы в отладчике	14
4.9	Проверка работы программы	15
4.10	Установка брейкпоинта	15
4.11	Дисассемблирование программы	16
4.12	Переключение отображения команд	16
4.13	Установка брейкпоинтов и их список	17
4.14	Просмотр содержимого регистров	17
4.15	Просмотр содержимого регистров разными способами	17
4.16	Изменение значений переменных разными способами	18
4.17	Вывод значения регистра в разных форматах	18
4.18	Использование команды set	18
4.19	Использование команды set	18
4.20	Завершение программы и выход	19
4.21	Копирование файла и создание исполняемого	19
4.22	Загрузка файла в отладчик с аргументами	19
4.23	Установка брейкпоинта и запуск программы	19
4.24	Просмотр позиций стека	20
4.25	Измененный текст программы	21
4.26	Создание файла lab9-5.asm	23
4.27	Текст программы файла	23
4.28	Работа в отладчике	23
4.29	Результат работы программы	24
4.30	Поиск ошибки через отладчик	24
4.31	Результат работы программы	24

Список таблиц

1 Цель работы

Цель данной лабораторной 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. Выполнение заданий 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.

Можно выделить следующие типы ошибок:

- синтаксические ошибки — обнаруживаются во время трансляции исходного кода и вызваны нарушением ожидаемой формы или структуры языка;
- семантические ошибки — являются логическими и приводят к тому, что программа запускается, отработывает, но не даёт желаемого результата;
- ошибки в процессе выполнения — не обнаруживаются при трансляции и вызывают прерывание выполнения программы (например, это ошибки, связанные с переполнением или делением на ноль).

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

Третий этап — выяснение причины ошибки. После определения местонахождения ошибки обычно проще определить причину неправильной работы программы. Последний этап — исправление ошибки. После этого при повторном запуске программы, может обнаружиться следующая ошибка, и процесс отладки

начнётся заново.

4 Выполнение лабораторной работы

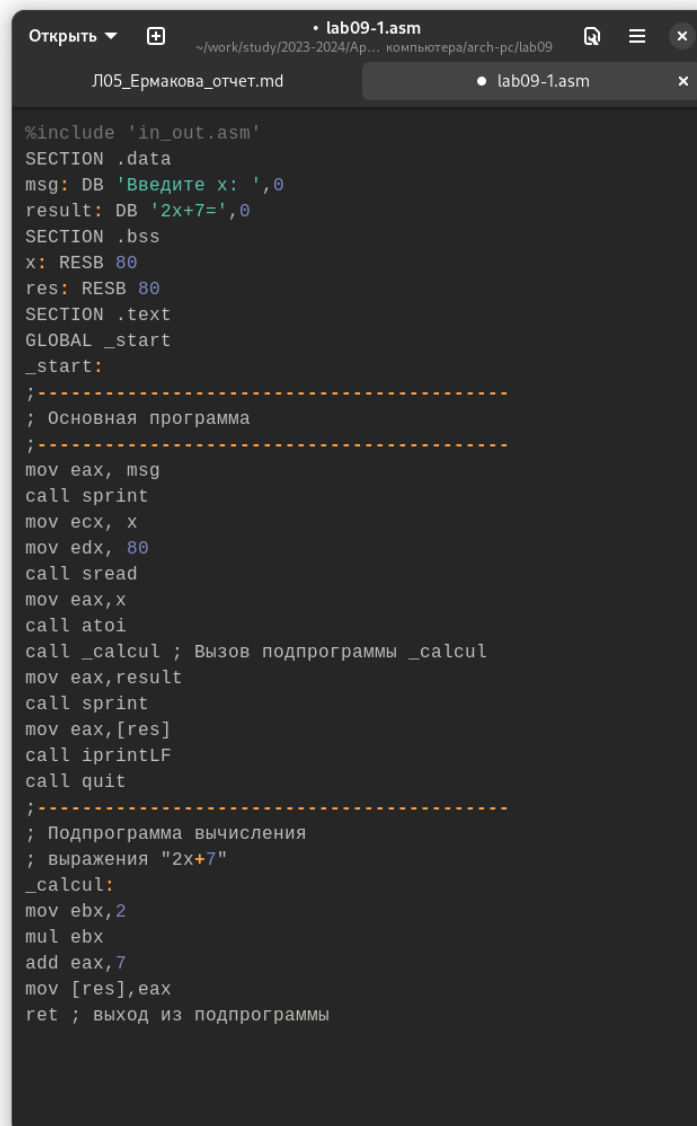
4.1 Реализация подпрограмм в NASM

Создаю каталог для выполнения лабораторной работы №9 (рис. 4.1).

```
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ mkdir lab09  
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ cd lab09  
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ touch lab09-1.asm
```

Рис. 4.1: Создание каталога и файла

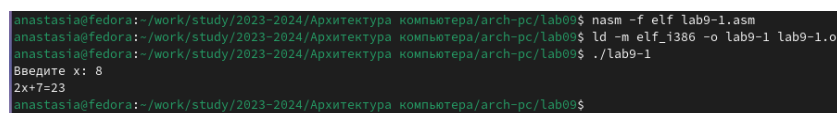
Ввожу в файл текст программы из листинга 9.1 (рис. 4.2).



```
%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
mov [res], eax
ret ; выход из подпрограммы
```

Рис. 4.2: Текст программы файла lab9-1.asm

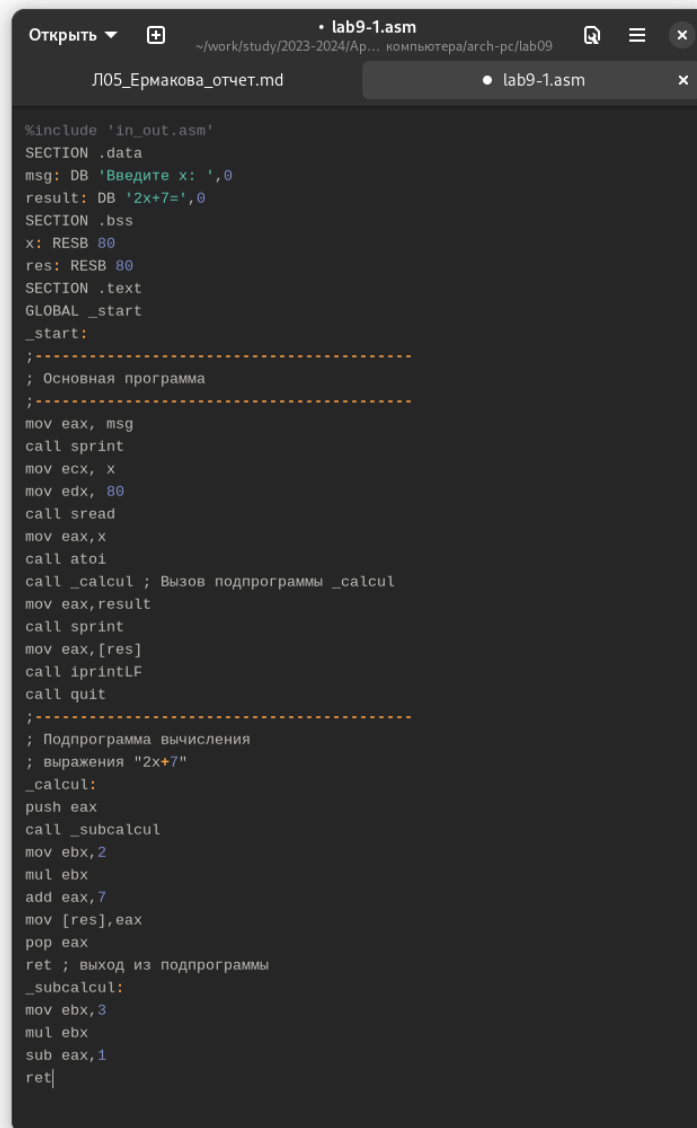
Создаю исполняемый файл и запускаю его (рис. 4.3). Программа выполняет вычисление заданной функции.



```
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab9-1.asm
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ./lab9-1
Введите x: 8
2x+7=23
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$
```

Рис. 4.3: Результат работы программы

Изменяю текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения $f(g(x))$ (рис. 4.4).



```
Открыть ▾ + lab9-1.asm
~/work/study/2023-2024/Ар... компьютера/arch-pc/lab09
Л05_Ермакова_отчет.md lab9-1.asm 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:
push eax
call _subcalcul
mov ebx, 2
mul ebx
add eax, 7
mov [res], eax
pop eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx, 3
mul ebx
sub eax, 1
ret
```

Рис. 4.4: Измененный текст программы

Код программы:

```
%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:
;-----
; 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:
push eax

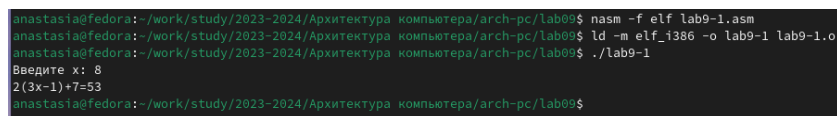
```

```

call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [res],eax
pop eax
ret ; выход из подпрограммы
_subcalcul:
mov ebx,3
mul ebx
sub eax,1
ret

```

Создаю исполняемый файл и запускаю его (рис. 4.5). Программа работает исправно.



```

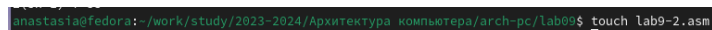
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab9-1.asm
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 lab9-1.o
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ./lab9-1
Введите x: 8
2(3x-1)+7=53
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$

```

Рис. 4.5: Результат работы программы

4.2 Отладка программ с помощью GDB

Создаю файл lab9-2.asm (рис. 4.6).



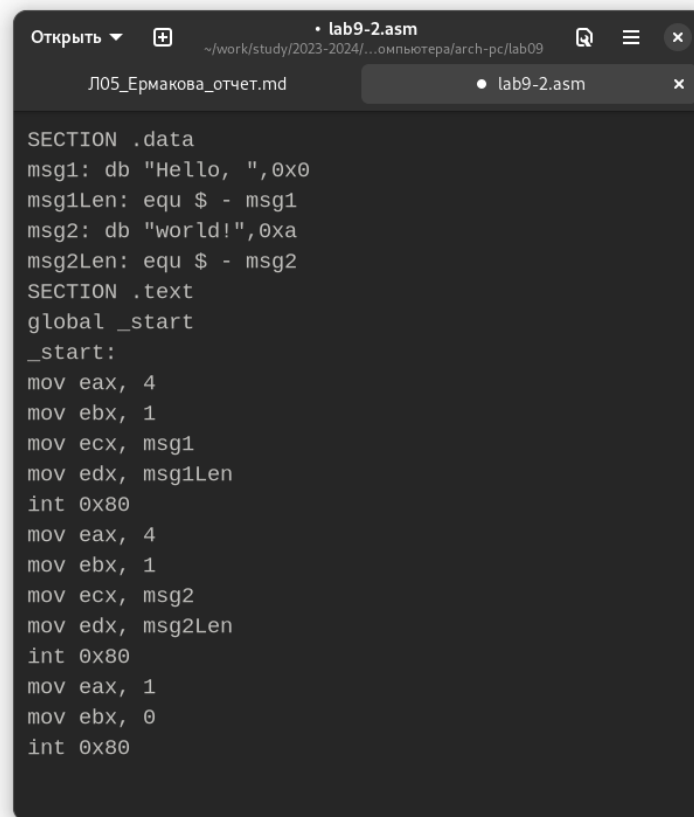
```

anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ touch lab9-2.asm

```

Рис. 4.6: Создание файла lab9-2.asm

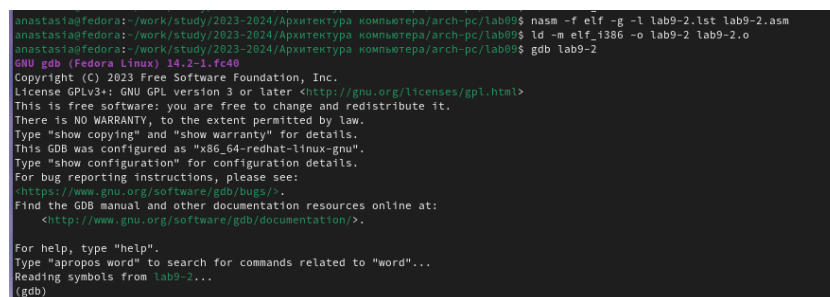
В файл копирую текст программы из листинга 9.2 (рис. 4.7).



```
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. 4.7: Текст программы файла lab9-2.asm

Получаю исполняемый файл. Для работы с GDB в исполняемый файл добавляю отладочную информацию. Загружаю файл в отладчик gdb (рис. 4.8).

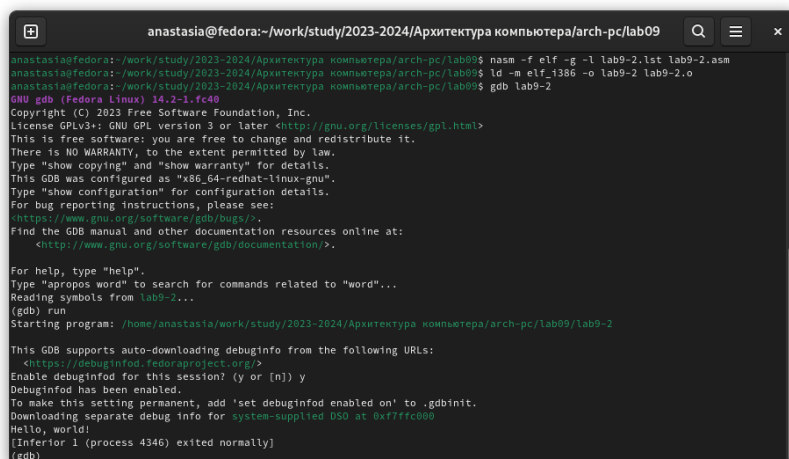


```
anastasiap@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
anastasiap@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
anastasiap@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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)
```

Рис. 4.8: Запуск программы в отладчике

Проверяю работу программы, запустив ее в отладчике с помощью команды

run (рис. 4.9).



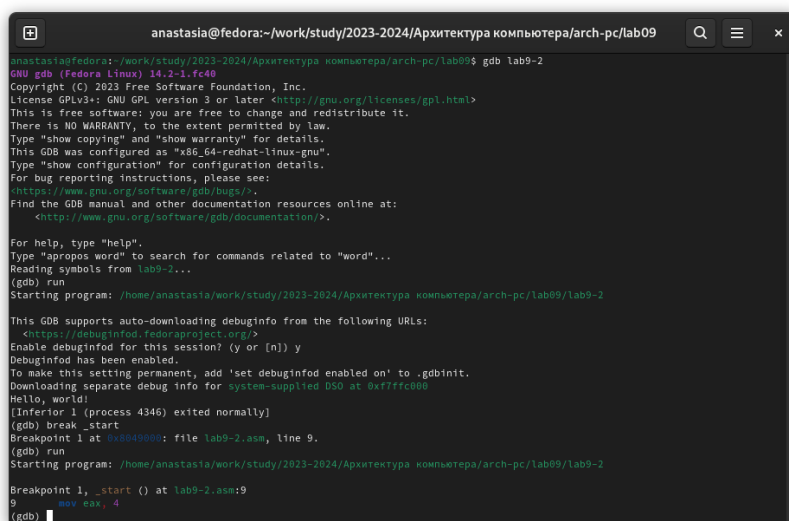
```
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab9-2.lst lab9-2.asm
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 lab9-2.o
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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) run
Starting program: /home/anastasia/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]) 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 0x7ffffc000
Hello, world!
[Inferior 1 (process 4346) exited normally]
(gdb)
```

Рис. 4.9: Проверка работы программы

Для более подробного анализа программы устанавливаю брейкпоинт на метку `_start`, с которой начинается выполнение любой ассемблерной программы, и запускаю ее (рис. 4.10).



```
anastasia@fedora:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ gdb lab9-2
GNU gdb (Fedora Linux) 14.2-1.fc40
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) run
Starting program: /home/anastasia/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]) 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 0x7ffffc000
Hello, world!
[Inferior 1 (process 4346) exited normally]
(gdb) break _start
Breakpoint 1 at 0x28045000: file lab9-2.asm, line 9.
(gdb) run
Starting program: /home/anastasia/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09/lab9-2

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

Рис. 4.10: Установка брейкпоинта

Смотрю дисассемблированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 4.11).

```
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>: mov    eax, %eax
0x08049005 <+5>: mov    %ebx, %ecx
0x0804900a <+10>: mov    $0x040000, %ecx
0x0804900f <+15>: mov    %ecx, %edx
0x08049014 <+20>: int    $0x80
0x08049016 <+22>: mov    %eax, %ecx
0x0804901b <+27>: mov    %eax, %ecx
0x08049020 <+32>: mov    $0x040000, %ecx
0x08049025 <+37>: mov    %ecx, %edx
0x0804902a <+42>: int    $0x80
0x0804902c <+44>: mov    %eax, %ecx
0x08049031 <+49>: mov    %ecx, %ecx
0x08049036 <+54>: int    $0x80
End of assembler dump.
(gdb)
```

Рис. 4.11: Дисассемблирование программы

Переключаюсь на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel` (рис. 4.12).

```
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>: mov    eax, %eax
0x08049005 <+5>: mov    ebx, %ebx
0x0804900a <+10>: mov    ecx, 0x040000
0x0804900f <+15>: mov    edx, %ecx
0x08049014 <+20>: int    $0x80
0x08049016 <+22>: mov    eax, %eax
0x0804901b <+27>: mov    ebx, %ebx
0x08049020 <+32>: mov    ecx, 0x040000
0x08049025 <+37>: mov    edx, %ecx
0x0804902a <+42>: int    $0x80
0x0804902c <+44>: mov    eax, %eax
0x08049031 <+49>: mov    ebx, %ebx
0x08049036 <+54>: int    $0x80
End of assembler dump.
(gdb)
```

Рис. 4.12: Переключение отображения команд

Различия между синтаксисом АТТ и Intel заключаются в порядке операндов (АТТ - Операнд источника указан первым. Intel - Операнд назначения указан первым), их размере (АТТ - размер операндов указывается явно с помощью суффиксов, непосредственные операнды предваряются символом \$; Intel - Размер операндов неявно определяется контекстом, как `ax`, `eax`, непосредственные операнды пишутся напрямую), именах регистров (АТТ - имена регистров предваряются символом %, Intel - имена регистров пишутся без префиксов).

4.2.1 Добавление точек останова

Включаю режим псевдографики для более удобного анализа программы. С помощью команды `i b` (`info breakpoints`) проверяю, что брейкпоинт сохранился. Далее устанавливаю еще одну точку останова по адресу инструкции и просматриваю все установленные точки останова с помощью команды `i b` (рис. 4.13).


```

B+>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,0x804a008

native process 4699 In: _start L9 PC: 0x8049000
(gdb) b *0x8049031
Breakpoint 2 at 0x8049031: file lab9-2.asm, line 20.
(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 0x8049031 lab9-2.asm:20
(gdb)

```

Рис. 4.13: Установка брейкпоинтов и их список

4.2.2 Работа с данными программы в GDB

Просматриваю содержимое регистров командой `i r` (info registers) (рис. 4.14).

```

B+>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,0x804a008

native process 4699 In: _start L9 PC: 0x8049000
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x0      0
esp      0xffffcfc0 0xffffcfc0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 4.14: Просмотр содержимого регистров

Просматриваю содержимое переменных по имени и по адресу (рис. 4.15).

```

native process 4699 In: _start L9 PC: 0x8049000
esi      0x0      0
edi      0x0      0
eip      0x8049000 0x8049000 <_start>
eflags   0x202    [ IF ]
--Type <RET> for more, q to quit, c to continue without paging--
Quit
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb 0x804a008
0x804a008 <msg2>: "world!\n\034"
(gdb)

```

Рис. 4.15: Просмотр содержимого регистров разными способами

Меняю значения переменных по имени и по адресу (рис. 4.16).

```

native process 4699 In: _start L9 PC: 0x8049000
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(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='x'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "xorld!\n\034"
(gdb)

```

Рис. 4.16: Изменение значений переменных разными способами

Вывожу в различных форматах значение регистра edx (рис. 4.17).

```

(gdb) p/s $edx
$3 = 8
(gdb) p/t $edx
$4 = 1000
(gdb) p/x $edx
$5 = 0x8
(gdb)

```

Рис. 4.17: Вывод значения регистра в разных форматах

С помощью команды set изменяю значение регистра ebx (рис. 4.18 - 4.19).

```

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

```

Рис. 4.18: Использование команды set

```

(gdb) set $ebx=2
(gdb) p/s $ebx
$7 = 2
(gdb)

```

Рис. 4.19: Использование команды set

Разница вывода команд p/s \$ebx заключается в том, что в первом случае переменной присваивается строковое значение '2', поэтому выводится код ASCII этого символа (это 50 в десятичной системе). Во втором случае переменной присваивается числовое значение 2, поэтому команда выводит указатель на строку.

Завершаю выполнение программы с помощью команды c (continue) и выхожу из GDB с помощью команды q (quit) (рис. 4.20).

```

(gdb) c
Continuing.
hello, world!

Breakpoint 2, _start () at lab9-2.asm:20
(gdb) q
A debugging session is active.

    Inferior 1 [process 4699] will be killed.

Quit anyway? (y or n) y

```

Рис. 4.20: Завершение программы и выход

4.2.3 Обработка аргументов командной строки в GDB

Копирую файл lab8-2.asm, созданный в ходе выполнения предыдущей лабораторной работы, в файл с именем lab9-3.asm. Создаю исполняемый файл (рис. 4.21).

```

anastasiagfedora@:/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
anastasiagfedora@:/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
anastasiagfedora@:/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$

```

Рис. 4.21: Копирование файла и создание исполняемого

Загружаю исполняемый файл в отладчик с аргументами, используя ключ `--args` (рис. 4.22). При запуске аргументы загрузились в стек.

```

anastasiagfedora@:/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ gdb --args lab9-3 arg1 arg2 'arg 3'
GNU gdb (Fedora Linux) 14.2-1.fc40
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:
<http://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)

```

Рис. 4.22: Загрузка файла в отладчик с аргументами

Устанавливаю точку останова перед первой инструкцией в программе и запускаю ее (рис. 4.23).

```

(gdb) b _start
Breakpoint 1 at 0x04090008: file lab9-3.asm, line 8.
(gdb) run
Starting program: /home/anastasia/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09/lab9-3 arg1 arg2 'arg 3'

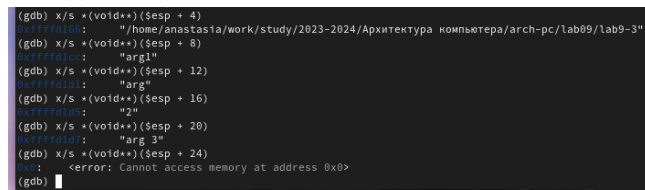
This GDB supports auto-downloading debuginfo from the following URLs:
<https://debuginfod.fedoraproject.org/>
Enable debuginfod for this session? (y or [n]) y
Debuginfod has been enabled.
To make this setting permanent, add 'set debuginfod enabled on' to .gdbinit.

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

```

Рис. 4.23: Установка брейкпоинта и запуск программы

Просматриваю остальные позиции стека (рис. 4.24).



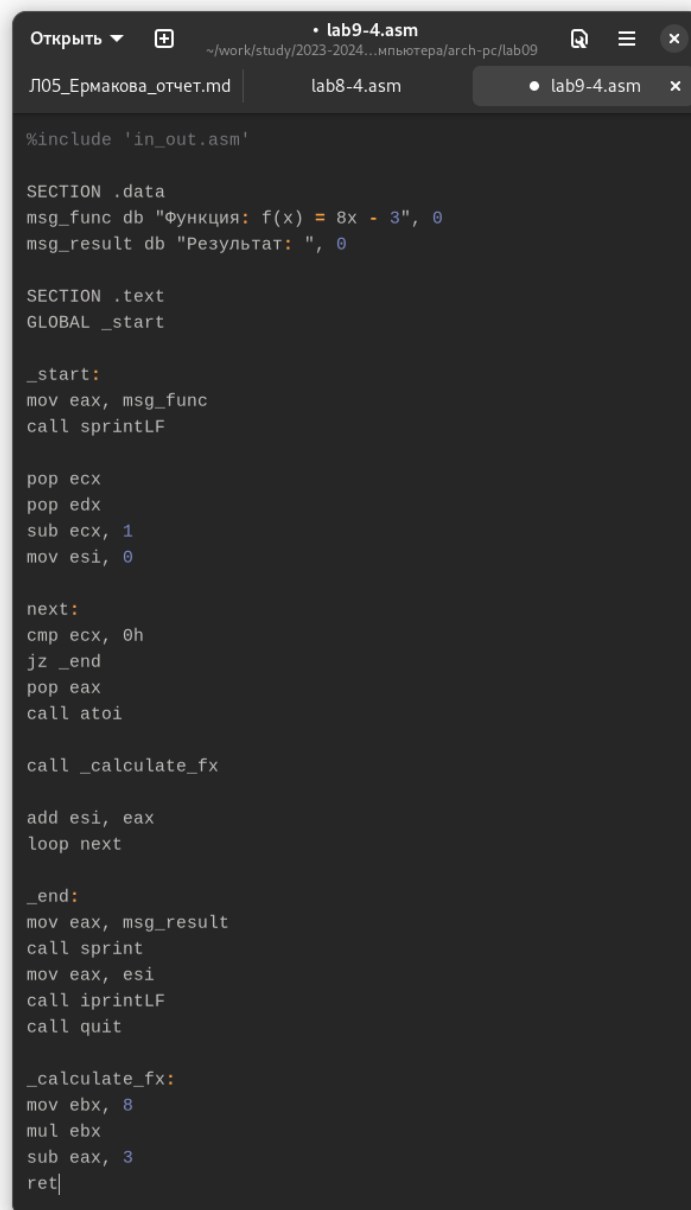
```
(gdb) x/s *(void**)(esp + 4)
0xfffff010:  "/home/anastasia/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)(esp + 8)
0xfffff01c:  "arg1"
(gdb) x/s *(void**)(esp + 12)
0xfffff024:  "arg"
(gdb) x/s *(void**)(esp + 16)
0xfffff028:  "2"
(gdb) x/s *(void**)(esp + 20)
0xfffff02c:  "arg 3"
(gdb) x/s *(void**)(esp + 24)
0x0:  <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 4.24: Просмотр позиций стека

Заметим, что шаг изменения адреса равен 4. Число обусловлено разрядностью системы, а указатель `void` занимает как раз 4 байта, ошибка при аргументе +24 означает, что аргументы на вход программы закончились.

4.3 Выполнение заданий для самостоятельной работы

1. Преобразую программу из предыдущей лабораторной работы, реализовав вычисление значения функции $f(x)$ как подпрограмму (рис. 4.25).



```
Открыть ▾ + lab9-4.asm
~/work/study/2023-2024...мпыютеpa/arch-pc/lab09
Л05_Ермакова_отчет.md | lab8-4.asm • lab9-4.asm x

%include 'in_out.asm'

SECTION .data
msg_func db "Функция: f(x) = 8x - 3", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintfLF

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

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

call _calculate_fx

add esi, eax
loop next

_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintfLF
call quit

_calculate_fx:
mov ebx, 8
mul ebx
sub eax, 3
ret
```

Рис. 4.25: Измененный текст программы

Код программы:

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

SECTION .data

msg_func db "Функция: $f(x) = 8x - 3$ ", 0

msg_result db "Результат: ", 0

SECTION .text

GLOBAL _start

_start:

mov eax, msg_func

call sprintf

pop ecx

pop edx

sub ecx, 1

mov esi, 0

next:

cmp ecx, 0h

jz _end

pop eax

call atoi

call _calculate_fx

add esi, eax

loop next

_end:

mov eax, msg_result

```

call sprint
mov eax, esi
call iprintLF
call quit

_calculate_fx:
mov ebx, 8
mul ebx
sub eax, 3
ret

```

2. Создаю файл lab9-5.asm (рис. 4.26), ввожу текст программы из листинга 9.3 (рис. 4.27).

```

anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ touch lab9-5.asm

```

Рис. 4.26: Создание файла lab9-5.asm

```

anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ touch lab9-5.asm

```

Рис. 4.27: Текст программы файла

Создаю исполняемый файл и открываю с помощью отладчика (рис. 4.28).

```

anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ nasm -f elf -g -l lab9-5.lst lab9-5.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ gdb lab9-5
GNU gdb (Fedora Linux) 14.2-1.fc40
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 lab9-5...
(gdb)

```

Рис. 4.28: Работа в отладчике

Запускаю программу с помощью команды `g` (`run`), выводится результат 10, он неверный (рис. 4.29).

```
(gdb) r
Starting program: /home/anastasia/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09/lab9-5
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.
Downloading separate debug info for system-supplied DSO at 0xf7ffc000
Результат: 10
[Inferior 1 (process 4534) exited normally]
(gdb)
```

Рис. 4.29: Результат работы программы

Затем пошагово просматриваю изменение значений регистров через `i r`. При выполнении инструкции `mul ecx` можно заметить, что результат умножения записывается в регистр `eax`, но так же меняет и `edx`. Значение регистра `ebx` не обновляется напрямую, поэтому программа неверно подсчитывает функцию (рис. 4.30).

```
Register group: general
eax    0x2      2      ecx    0x4      4
edx    0x0      0      ebx    0x5      5
esp    0xffffcf10 0xffffcf10 ebp    0x0      0x0
esi    0x0      0      edi    0x0      0
eip    0x80490f9 0x80490f9 <_start+17> eflags 0x206    [ PF IF ]
cs     0x23     35     ss     0x2b     43
ds     0x2b     43     es     0x2b     43
fs     0x0      0      gs     0x0      0

0x80490e0 <_start> mov    $0x3,%ebx
0x80490e4 <_start+5> mov    $0x2,%eax
0x80490f2 <_start+10> add    %eax,%ebx
0x80490f4 <_start+12> mov    $0x4,%ecx
>0x80490f9 <_start+17> mul    %ecx
0x80490f6 <_start+19> add    $0x1,%ebx
0x80490fe <_start+22> mov    %ebx,%edi
0x8049100 <_start+24> mov    $0x804a000,%eax
0x8049105 <_start+29> call   0x804900f <sprint>
0x804910a <_start+34> mov    %edi,%eax

native process 8526 (asm) In: _start L14 PC: 0x80490f9
eax    0x2      2
ecx    0x4      4
edx    0x0      0
ebx    0x5      5
esp    0xffffcf10 0xffffcf10
ebp    0x0      0x0
esi    0x0      0
edi    0x0      0
eip    0x80490f9 0x80490f9 <_start+17>
eflags 0x206    [ PF IF ]
cs     0x23     35
--Type <RET> for more, q to quit, c to continue without paging--
```

Рис. 4.30: Поиск ошибки через отладчик

Исправляю найденную ошибку, теперь программа верно считает функцию (рис. 4.31).

```
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ nasm -f elf lab9-5.asm
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ld -m elf_i386 -o lab9-5 lab9-5.o
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$ ./lab9-5
Результат: 25
anastasia@fedora: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab09$
```

Рис. 4.31: Результат работы программы

Код исправленной программы:


```

#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 ebx,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

```

5 Выводы

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

6 Список литературы

1. Архитектура ЭВМ