

Отчёт по лабораторной работе №10

дисциплина: Архитектура компьютера

Максим Александрович Мишонков

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
5	Выводы	23

Список иллюстраций

4.1	Создание каталога и файла	8
4.2	Введение текста программы	9
4.3	Проверка работы исполняемого файла	9
4.4	Изменение текста программы	10
4.5	Изменение текста программы	11
4.6	Проверка работы исполняемого файла	12
4.7	Создание файла	12
4.8	Введение текста программы	12
4.9	Проверка работы исполняемого файла	13
4.10	Запуск программы	13
4.11	Просмотр дисассимилированного кода программы	14
4.12	Переключение на отображение команд с Intel'овским синтаксисом	14
4.13	Включение режима псевдографики	15
4.14	Включение режима псевдографики	15
4.15	Проверка точки останова по имени метки (_start))	15
4.16	Установка ещё одной точки останова по адресу инструкции	16
4.17	Просмотр информации о всех установленных точках останова	16
4.18	Просмотр содержимого регистров	16
4.19	Просмотр содержимого регистров	16
4.20	Значение переменной msg1	17
4.21	Значение переменной msg2	17
4.22	Изменение первого символа переменной msg1	17
4.23	Замена символа во второй переменной msg	17
4.24	Вывод в различных форматах значения регистра edx	18
4.25	Изменение значения регистра ebx	18
4.26	Копирование файла lab9-2.asm в файл lab10-3.asm	18
4.27	Создание исполняемого файла и загрузка его в отладчик	19
4.28	Установка точки останова перед первой инструкцией в программе и её запуск	19
4.29	Команда x/x \$esp	19
4.30	Просмотр остальных позиций стека	20
4.31	Преобразование программы	20
4.32	Преобразование программы	21
4.33	Проверка работы исполняемого файла	21
4.34	Изменение текста программы	22
4.35	Проверка работы исполняемого файла	22

1 Цель работы

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

2 Задание

Научиться писать программы с использованием подпрограмм, ознакомиться с методами отладки при помощи GDB и его основными возможностями.

3 Теоретическое введение

Отладка - это процесс поиска и исправления ошибок в программе (обнаружение ошибки, поиск её местонахождения, определение причины ошибки, исправление ошибки)

Точки останова - это специально отмеченные места в программе, в которых программа-отладчик приостанавливает выполнение программы и ждёт команд.

Отладчик GDB позволяет увидеть, что происходит “внутри” программы в момент её выполнения и что делает программа в момент сбоя. GDB может 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.

Если в подпрограмме встречается одинаковый участок кода, его можно оформить в виде подпрограммы, а во всех нужных местах поставить её вызов. При этом подпрограмма будет содержаться в коде в одном экземпляре, что позволит уменьшить размер кода всей программы.

Для вызова подпрограммы из основной программы применяется инструкция `call`, которая заносит адрес следующей инструкции в стек и загружает в регистр `esp` адрес соответствующей подпрограммы. Завершается подпрограмма инструкцией `ret`, которая извлекает из стека адрес, занесённый туда соответствующей

инструкцией call, и заносит его в еір.

4 Выполнение лабораторной работы

1. Создал каталог для выполнения лабораторной работы №10, перешёл в него и создал файл lab10-1.asm. (рис. 4.1)

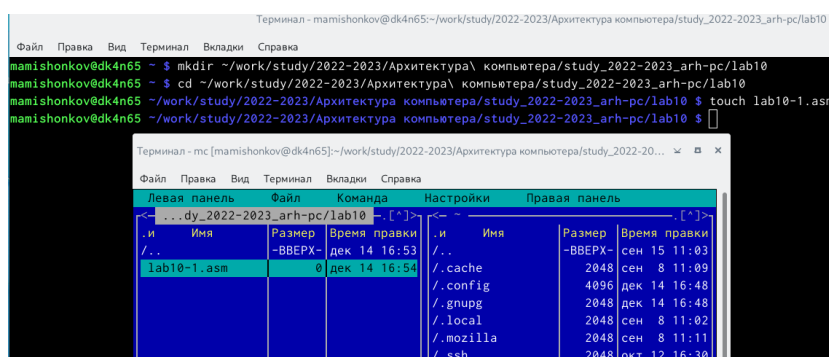


Рис. 4.1: Создание каталога и файла

2. Ввёл в файл lab10-1.asm текст программы из листинга 10.1. (рис. 4.2)


```

GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/m/a/mamishonkov
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
rez: 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, [rez]
call iprintLF
call quit

_calcul:
mov ebx, 2
mul ebx
add eax, 7
mov [rez], eax
ret

```

Рис. 4.2: Введение текста программы

3. Создал исполняемый файл и проверил его работу. (рис. 4.3)

```

mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ nasm -f elf lab10-1.asm
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-1
Введите x: 5
2x+7=17
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-1
Введите x: 6
2x+7=19
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $

```

Рис. 4.3: Проверка работы исполняемого файла

4. Изменил текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul`, для вычисления выражения $f(g(x))$, где x вводится с клавиатуры, $f(x)=2x+7$, $g(x)=3x-1$. Проверил работу исполняемого файла. (рис. 4.4, 4.5, 4.6)

```
GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/m/a/mamishonkov.  
%include 'in_out.asm'  
  
SECTION .data  
msg: DB 'Введите x: ',0  
prim1: DB 'f(x) = 2x+7',0  
prim2: DB 'g(x) = 3x-1',0  
result: DB 'f(g(x))= ',0  
  
SECTION .bss  
x: RESB 80  
res: RESB 80  
  
SECTION .text  
GLOBAL _start  
_start:  
  
mov eax,prim1  
call sprintf  
  
mov eax,prim2  
call sprintf  
  
mov eax,msg  
call sprintf  
  
mov ecx,x  
mov edx,80  
call sread  
  
mov eax,x  
call atoi
```

Рис. 4.4: Изменение текста программы

```
call _calcul

mov eax,result
call sprint
mov eax,[res]
call iprintLF

call quit

_calcul:

call _subcalcul

mov ebx,2
mul ebx
add eax,7
mov [res],eax
ret

_subcalcul:

mov ebx,3
mul ebx
sub eax,1
ret
```

Рис. 4.5: Изменение текста программы

```

mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ nasm -f elf lab10-1.asm
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ld -m elf_i386 -o lab10-1 lab10-1.o
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 3
f(g(x))= 23
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-1
f(x) = 2x+7
g(x) = 3x-1
Введите x: 2
f(g(x))= 17
mamishonkov@dk6n57 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $

```

Рис. 4.6: Проверка работы исполняемого файла

5. Создал файл lab10-2.asm. (рис. 4.7)

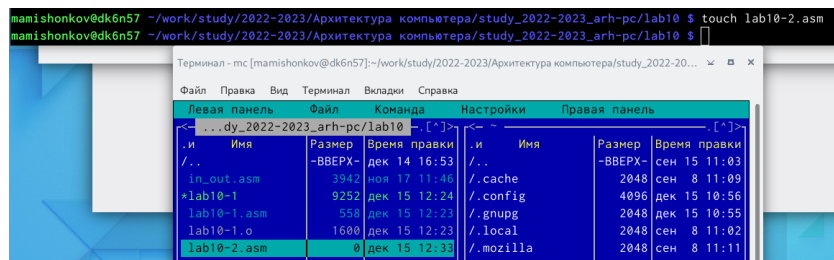


Рис. 4.7: Создание файла

6. Ввёл в файл lab10-2.asm текст программы из листинга 10.2. (рис. 4.8)

```

GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/m/a/mamishonkov
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. 4.8: Введение текста программы

7. Получил исполняемый файл. Для работы с GDB в исполняемый файл добавил отладочную информацию, проведя трансляцию программ с ключом '-g'. Загрузил исполняемый файл в отладчик gdb. Проверил работу программы, запустив её в оболочке GDB с помощью команды run. (рис. 4.9)

```
mamishonkov@dk6n57: ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ nasm -f elf -g -l lab10-2.lst lab10-2.asm
mamishonkov@dk6n57: ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ld -m elf_i386 -o lab10-2 lab10-2.o
mamishonkov@dk6n57: ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ gdb lab10-2
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab10-2...
(gdb) run
Starting program: /afs/dk.sci.pfu.edu.ru/home/m/a/mamishonkov/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10/lab10-2
Hello, world!
[inferior 1 (process 5486) exited normally]
(gdb) []
```

Рис. 4.9: Проверка работы исполняемого файла

8. Для более подробного анализа программы установил брейкпоинт на метку _start, с которой начинается выполнение любой ассемблерной программы, и запустил её. (рис. 4.10)

```
(gdb) break _start
Breakpoint 1 at 0x00400000: file lab10-2.asm, line 9.
(gdb) run
Starting program: /afs/dk.sci.pfu.edu.ru/home/m/a/mamishonkov/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10/lab10-2
Breakpoint 1, _start () at lab10-2.asm:9
9      mov eax, 1
(gdb) []
```

Рис. 4.10: Запуск программы

11. Посмотрел дисассимилированный 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     $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. 4.11: Просмотр дисассимилированного кода программы

12. Переключился 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,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. 4.12: Переключение на отображение команд с Intel'овским синтаксисом

13. Включил режим псевдографики для более удобного анализа программы. (рис. 4.13, 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
      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 5715 In: _start
(gdb)

Рис. 4.13: Включение режима псевдографики

```

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

```

native process 5715 In: _start
(gdb) layout regs
(gdb)

Рис. 4.14: Включение режима псевдографики

14. На предыдущих шагах была установлена точка останова по имени метки (_start). Проверил это с помощью команды info breakpoints. (рис. 4.15)

```

(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab10-2.asm:9
breakpoint already hit 1 time
(gdb) 

```

Рис. 4.15: Проверка точки останова по имени метки (_start))

15. Установил ещё одну точку останова по адресу инструкции, определив адрес предпоследней инструкции. (рис. 4.16)

```
(gdb) b *0x8049000
Note: breakpoint 1 also set at pc 0x8049000.
Breakpoint 2 at 0x8049000: file lab10-2.asm, line 9.
(gdb) █
```

Рис. 4.16: Установка ещё одной точки останова по адресу инструкции

16. Посмотрел информацию о всех установленных точках останова. (рис. 4.17)

```
(gdb) i b
Num      Type           Disp Enb Address      What
1        breakpoint     keep y   0x08049000 lab10-2.asm:9
          breakpoint already hit 1 time
2        breakpoint     keep y   0x08049000 lab10-2.asm:9
(gdb) █
```

Рис. 4.17: Просмотр информации о всех установленных точках останова

17. Посмотрел содержимое регистров с помощью команды info registers. (рис. 4.18, 4.19)

```
native process 5715 In: _start
eax      0x0           0
ecx      0x0           0
edx      0x0           0
ebx      0x0           0
esp      0xffffc4a0    0xffffc4a0
ebp      0x0           0x0
esi      0x0           0
--Type <RET> for more, q to quit, c to continue without paging--█
```

Рис. 4.18: Просмотр содержимого регистров

```
native process 5715 In: _start
--Type <RET> for more, q to quit, c to continue without paging--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
(gdb) █
```

Рис. 4.19: Просмотр содержимого регистров

18. Посмотрел значение переменной msg1. (рис. 4.20)

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

Рис. 4.20: Значение переменной msg1

19. Посмотрел значение переменной msg2. (рис. 4.21)

```
(gdb) x/1sb 0x804a008
0x804a008 <msg2>:      "world!\n\034"
(gdb) 
```

Рис. 4.21: Значение переменной msg2

20. Изменил первый символ переменной msg1. (рис. 4.22)

```
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>:      "hello, "
(gdb) 
```

Рис. 4.22: Изменение первого символа переменной msg1

21. Заменял один из символов во второй переменной msg. (рис. 4.23)

```
(gdb) set {char}&msg2='k'
(gdb) x/1sb &msg2
0x804a008 <msg2>:      "kor1d!\n\034"
(gdb) 
```

Рис. 4.23: Замена символа во второй переменной msg

21. Вывел в различных форматах значение регистра edx. (рис. 4.24)

```
(gdb) p/x $edx
$1 = 0x0
(gdb) p/t $edx
$2 = 0
(gdb) p/s $edx
$3 = 0
(gdb) 
```

Рис. 4.24: Вывод в различных форматах значения регистра edx

22. С помощью команды set изменил значение регистра ebx. (рис. 4.25)

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

Рис. 4.25: Изменение значения регистра ebx

23. Скопировал файл lab9-2.asm, созданный при выполнении лабораторной работы №9, с программой, выводящей на экран аргументы командной строки, в файл с именем lab10-3.asm. (рис. 4.26)

```
namishonkov@dk6n57 ~$ cp ~/work/study/2022-2023/Архитектура\ компьютера/study_2022-2023_arh-pc/lab09/lab9-2.asm ~/work/study/2022-2023/Архитектура\ компьютера/study_2022-2023_arh-pc/lab10/lab10-3.asm
```

Рис. 4.26: Копирование файла lab9-2.asm в файл lab10-3.asm

24. Создал исполняемый файл, загрузил его в отладчик, указав аргументы. (рис. 4.27)

```

mamishonkov@ddk6n57 ~$ cd ~/work/study/2022-2023/Архитектура_компьютера/study_2022-2023_arh-pc/lab10
mamishonkov@ddk6n57 ~/work/study/2022-2023/Архитектура_компьютера/study_2022-2023_arh-pc/lab10$ nasm -f elf -g -l lab10-3.lst lab10-3.asm
mamishonkov@ddk6n57 ~/work/study/2022-2023/Архитектура_компьютера/study_2022-2023_arh-pc/lab10$ ld -m elf_i386 -o lab10-3 lab10-3.o
mamishonkov@ddk6n57 ~/work/study/2022-2023/Архитектура_компьютера/study_2022-2023_arh-pc/lab10$ gdb --args lab10-3 аргумент1 аргумент2 аргумент3
GNU gdb (Gentoo 11.2 vanilla) 11.2
Copyright (C) 2022 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-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://bugs.gentoo.org/>.
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 lab10-3...
(gdb)

```

Рис. 4.27: Создание исполняемого файла и загрузка его в отладчик

25. Установил точку останова перед первой инструкцией в программе и запустил программу. (рис. 4.28)

```

(gdb) b _start
Breakpoint 1 at 0x00000000: file lab10-3.asm, line 5.
(gdb) run
Starting program: /afs/dk.sci.pfu.edu.ru/home/m/a/mamishonkov/work/study/2022-2023/Архитектура_компьютера/study_2022-2023_arh-pc/lab10/lab10-3 аргумент1 аргумент2 аргумент3
Breakpoint 1, _start () at lab10-3.asm:5
5      pop     ecx ; Извлечь из стека в 'ecx' количество
(gdb)

```

Рис. 4.28: Установка точки останова перед первой инструкцией в программе и её запуск

26. Адрес вершины стека хранится в регистре esp и по этому адресу располагается число, равное количеству аргументов командной строки (включая имя программы). Как видно, число аргументов равно 5. Это имя программы lab10-3.asm. (рис. 4.29)

```

(gdb) x/x $esp
0xffffc450: 0x00000005
(gdb)

```

Рис. 4.29: Команда x/x \$esp

27. Просмотрел остальные позиции стека. По адресу esp+4 располагается адрес в памяти, где находится имя программы, по адресу esp+8 хранится адрес первого аргумента, по адресу esp+12 - второго аргумента и т.д. (рис. 4.30)

```

(gdb) x/s *(void**)(esp + 4)
0xffffc701:  "/afs/.dk.sci.pfu.edu.ru/home/m/a/mamishonkov/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10/lab10-3"
(gdb) x/s *(void**)(esp + 8)
0xffffc705:  "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffc709:  "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffc70d:  "2"
(gdb) x/s *(void**)(esp + 20)
0xffffc711:  "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:  <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 4.30: Просмотр остальных позиций стека

Самостоятельная работа

1. Преобразовал программу из лабораторной работы №9 (задание 1 из самостоятельной работы), реализовав вычисление значения функции $f(x)$ как подпрограмму. Создал исполняемый файл и проверил его работу. (рис. 4.31, 4.32, 4.33)

```

GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/m/a/mamishonkov
#include 'in_out.asm'

SECTION .data
ms1 db "Функция :f(x)=7(x+1) ", 0
ms2 db "Результат: ", 0

SECTION .text
global _start
_start:
mov eax,ms1
call sprintf
pop ecx
pop edx
sub ecx,1
mov esi,0
mov ebx,7

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

```

Рис. 4.31: Преобразование программы

```

_end:
mov eax, ms2
call sprint
mov eax, esi
call iprintLF
call quit

_calcul:
add eax, 1
mul ebx
add esi, eax
ret

```

Рис. 4.32: Преобразование программы

```

mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ nasm -f elf lab10-4.asm
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ld -m elf_i386 -o lab10-4 lab10-4.o
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-4 1 2 3
Функция : f(x)=7*(x+1)
Результат: 63
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-4 1 2 3 4
Функция : f(x)=7*(x+1)
Результат: 98

```

Рис. 4.33: Проверка работы исполняемого файла

2. С помощью отладчика GDB, анализируя изменения значений регистров, определил и исправил ошибку в программе вычисления выражения $(3+2)*4+5$. В результате программа должна выглядеть следующим образом. Создал исполняемый файл и проверил его работу. (рис. 4.34, 4.35)

```

GNU nano 6.3 /afs/.dk.sci.pfu.edu.ru/home/m/a/mamishonkov
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения (3+2)*4+5
mov ebx,3
mov eax,2
add eax,ebx
mov ecx,4
mul ecx
add eax,5
mov edi,eax
; ---- Вывод результата на экран
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

```

Рис. 4.34: Изменение текста программы

```

mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ touch lab10-5.asm
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ nasm -f elf lab10-5.asm
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ld -m elf_i386 -o lab10-5 lab10-5.o
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ ./lab10-5
Результат: 25
mamishonkov@dk4n65 ~/work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/lab10 $ 

```

Рис. 4.35: Проверка работы исполняемого файла

5 Выводы

В ходе выполнения данной лабораторной работы я научился писать программы с использованием подпрограмм, ознакомился с методами отладки при помощи GDB и его основными возможностями.