

Лабораторная работа №10

Дисциплина: Архитектура компьютера

Ким Ангелина Павловна

Содержание

1	Цель работы	6
2	Выполнение лабораторной работы	7
3	Выводы	36
	Список литературы	37

Список иллюстраций

2.1	Создание каталога	7
2.2	Текст программы из листинга 10.1	8
2.3	Создание исполняемого файла	9
2.4	Текст измененной программы	10
2.5	Создание исполняемого файла	11
2.6	Исполняемый файл в отладчик GDB	11
2.7	Установка брейкпоинта	12
2.8	Дисассимилированный код	12
2.9	Команда set disassembly-flavor intel	13
2.10	Режим псевдографики	14
2.11	Команда info breakpoints	14
2.12	Адрес предпоследней инструкции	14
2.13	Информация о всех установленных точках останова	15
2.14	Команда stepi (1)	16
2.15	Команда stepi (2)	17
2.16	Содержимое регистров	18
2.17	Значение переменной msg1	18
2.18	Значение переменной msg2	18
2.19	Первый символ переменной msg1	18
2.20	Первый символ переменной msg2	19
2.21	Значения регистра edx	19
2.22	Команда set	20
2.23	Завершение программы	20
2.24	Копия файла	20
2.25	Создание исполняемого файла	21
2.26	Загрузка исполняемого файла в отладчик	21
2.27	Регистр esp	21
2.28	Позиции стека	22
2.29	Текст программы с подпрограммой	23
2.30	Создание исполняемого файла	24
2.31	Создание файла	24
2.32	Текст программы из листинга 10.3	25
2.33	Создание исполняемого файла	25
2.34	Режим псевдографики	26
2.35	Строчки кода (1)	27
2.36	Строчки кода (2)	28
2.37	Строчки кода (3)	29

2.38	Строчки кода (4)	30
2.39	Строчки кода (5)	31
2.40	Строчки кода (6)	32
2.41	Строчки кода (7)	33
2.42	Строчки кода (8)	34
2.43	Исправленный код	35
2.44	Исполняемый файл	35

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием подпрограмм. Знакомство с методами отладки при помощи GDB и его основными возможностями.

2 Выполнение лабораторной работы

Создали каталог для выполнения лабораторной работы №10, переходим в него и создаем файл lab10-1.asm (рис. 2.1)

```
[apkim@fedora ~]$ mkdir ~/work/arch-pc/lab10  
[apkim@fedora ~]$ cd ~/work/arch-pc/lab10  
[apkim@fedora lab10]$ touch lab10-1.asm  
[apkim@fedora lab10]$
```

Рис. 2.1: Создание каталога

Введем в файл lab10-1.asm текст программы из листинга 10.1. (рис. 2.2)

```

GNU nano 5.8 /
#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

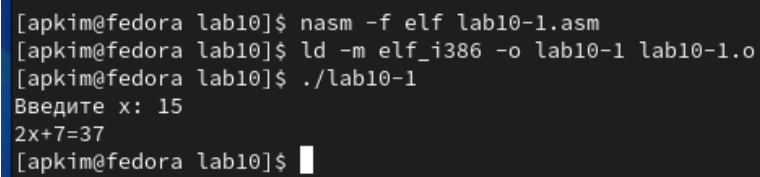
_calcul:
mov ebx,2
mul ebx
add eax,7
mov [rez],eax

ret

```

Рис. 2.2: Текст программы из листинга 10.1

Создаем исполняемый файл и проверяем его работу (рис. 2.3)

A terminal window with a dark background and light blue text. The prompt is [arkim@fedora lab10]\$. The first command is nasm -f elf lab10-1.asm. The second command is ld -m elf_i386 -o lab10-1 lab10-1.o. The third command is ./lab10-1. The output shows 'Введите x: 15' followed by '2x+7=37'. The prompt returns to [arkim@fedora lab10]\$.

```
[arkim@fedora lab10]$ nasm -f elf lab10-1.asm
[arkim@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[arkim@fedora lab10]$ ./lab10-1
Введите x: 15
2x+7=37
[arkim@fedora lab10]$
```

Рис. 2.3: Создание исполняемого файла

Изменяем текст программы, добавив подпрограмму `_subcalcul` в подпрограмму `_calcul` (рис. 2.4)

```
GNU nano 5.8

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:
call _subcalcul
mov ebx,2
mul ebx
add eax,7
mov [rez],eax

ret

_subcalcul:
mov ebx,3
mul ebx
sub eax,1
mov [rez],eax
ret
```

Рис. 2.4: Текст измененной программы

Создаем исполняемый файл и проверяем его работу (рис. 2.5)

```
[apkim@fedora lab10]$ nasm -f elf lab10-1.asm
[apkim@fedora lab10]$ ld -m elf_i386 -o lab10-1 lab10-1.o
[apkim@fedora lab10]$ ./lab10-1
Введите x: 15
2x+7=95
[apkim@fedora lab10]$
```

Рис. 2.5: Создание исполняемого файла

Далее создаем файл lab10-2.asm, вносим туда текст программы из листинга 10.2, создаем исполняемый файл, для работы с GDB в исполняемый файл добавили отладочную информацию, для этого трансляцию программ провели с ключом “-g”. Загружаем исполняемый файл в отладчик GDB. Проверили работу программы, запустив ее в оболочке GDB с помощью команды run. (рис. 2.6)

```
[apkim@fedora lab10]$ gdb lab10-2
GNU gdb (GDB) Fedora 11.1-5.fc34
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.ht
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 lab10-2...
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-2
Hello, world!
[Inferior 1 (process 4400) exited normally]
(gdb)
```

Рис. 2.6: Исполняемый файл в отладчик GDB

Установили брейкпоинт на метку _start (рис. 2.7)

```

(gdb) break _start
Breakpoint 1 at 0x8049000: file lab10-2.asm, line 9.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-2

Breakpoint 1, _start () at lab10-2.asm:9
9      mov eax, 4
(gdb) █

```

Рис. 2.7: Установка брейкпоинта

Посмотрели дисассимилированный код программы с помощью команды `disassemble` начиная с метки `_start` (рис. 2.8)

```

(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. 2.8: Дисассимилированный код

Переключились на отображение команд с Intel'овским синтаксисом, введя команду `set disassembly-flavor intel`. (рис. 2.9)

```

(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. 2.9: Команда set disassembly-flavor intel

Различия отображения синтаксиса машинных кодов в режимах АТТ и Intel: в АТТ перед адресом регистра ставится \$, а перед названием регистра %, сначала записывается адрес, а потом - регистр. В Intel сначала регистр, а потом адрес, и перед ними ничего не ставится. Далее переходим на режим псевдографики для более удобного анализа программы. (рис. 2.10)

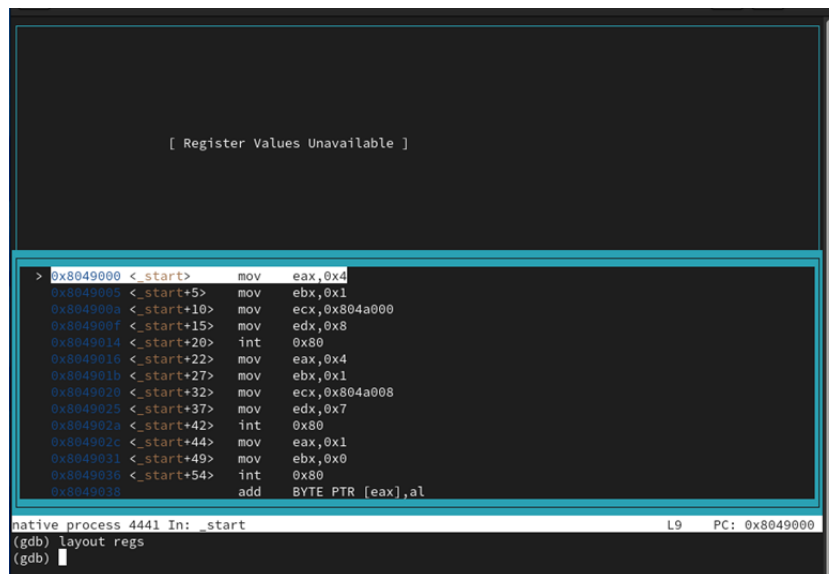


Рис. 2.10: Режим псевдографики

На предыдущих шагах была установлена точка останова по имени метки (`_start`). Проверяем это с помощью команды `info breakpoints`. (рис. 2.11)

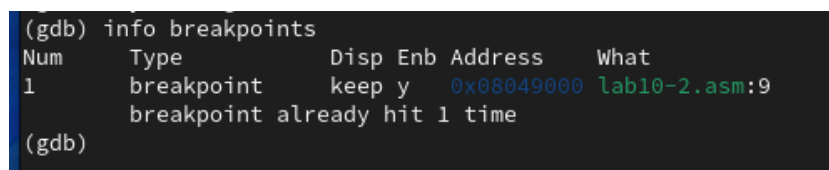


Рис. 2.11: Команда `info breakpoints`

Далее определили адрес предпоследней инструкции и установили точку останова (рис. 2.12)

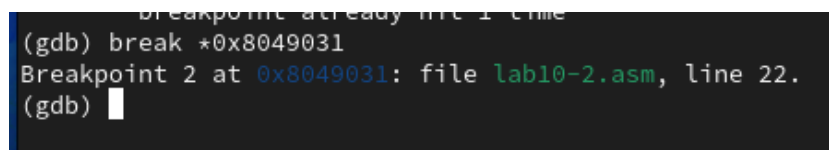


Рис. 2.12: Адрес предпоследней инструкции

Посмотрели информацию о всех установленных точках останова (рис. 2.13)

```

breakpoint 2 at 0x08049031 file lab10-2.asm, line 22
(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   0x08049031 lab10-2.asm:22
(gdb)

```

Рис. 2.13: Информация о всех установленных точках останова

Выполняем 5 инструкций с помощью команды `stepi` и проследим за изменением значения регистров (рис. 2.14)

```
apkim@fedora:~/work/arch-pc/lab10-  
Register group: general  
eax      0x4      4  
ecx      0x0      0  
edx      0x0      0  
ebx      0x0      0  
esp      0xffffd1c0 0xffffd1c0  
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  
  
> 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  
0x8049038          add    BYTE PTR [eax],al  
  
native process 4441 In: _start  
(gdb) layout regs  
(gdb) info breakpoints  
Num      Type             Disp Enb Address      What  
1        breakpoint       keep y  0x08049000 lab10-2.asm:9  
          breakpoint already hit 1 time  
(gdb) break *0x8049031  
Breakpoint 2 at 0x8049031: file lab10-2.asm, line 22.  
(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  0x08049031 lab10-2.asm:22  
(gdb) stepi  
(gdb) 
```

Рис. 2.14: Команда stepi (1)

Изменяются значения регистров: eax, ecx, edx, ebx (рис. 2.15)


```

apkim@fedora:~/work/arch-pc/lab10 —
Register group: general
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43

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
0x8049038          add    BYTE PTR [eax],al

native process 4441 In: _start
Num      Type      Disp Enb Address      What
1        breakpoint keep y  0x08049000 lab10-2.asm:9
          breakpoint already hit 1 time
(gdb) break *0x8049031
Breakpoint 2 at 0x8049031: file lab10-2.asm, line 22.
(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  0x08049031 lab10-2.asm:22
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)

```

Рис. 2.15: Команда stepi (2)

Посмотрели содержимое регистров с помощью команды info registers (рис. 2.16)

```

native process 4441 In: _start
eax      0x8      8
ecx      0x804a000 134520832
edx      0x8      8
ebx      0x1      1
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x8049016 0x8049016 <_start+22>
eflags   0x202    [ IF ]
cs       0x23     35
ss       0x2b     43
ds       0x2b     43
es       0x2b     43
fs       0x0      0
--Type <RET> for more, q to quit, c to continue without paging--

```

Рис. 2.16: Содержимое регистров

Далее посмотрели значение переменной `msg1` по имени (рис. 2.17)

```

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

```

Рис. 2.17: Значение переменной `msg1`

Посмотрели значение переменной `msg2` по адресу. Адрес переменной определили по дизассемблированной инструкции. Посмотрели инструкцию `mov ecx, msg2` которая записывает в регистр `ecx` адрес переменной `msg2`. (рис. 2.18)

```

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

```

Рис. 2.18: Значение переменной `msg2`

Далее изменяем первый символ переменной `msg1` (рис. 2.19)

```

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

```

Рис. 2.19: Первый символ переменной `msg1`

Заменяем первый символ во второй переменной msg2 (рис. 2.20)

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

Рис. 2.20: Первый символ переменной msg2

Чтобы посмотреть значения регистров используется команда print /F . Вывели в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра edx.(рис. 2.21)

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

Рис. 2.21: Значения регистра edx

С помощью команды set изменим значение регистра ebx (рис. 2.22)

```

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

```

Рис. 2.22: Команда set

Завершили выполнение программы с помощью команды continue и выходим из GDB (рис. 2.23)

```

(gdb) c
Continuing.
World!

Breakpoint 2, _start () at lab10-2.asm:22
(gdb) c
Continuing.
[Inferior 1 (process 4441) exited normally]
(gdb)

```

Рис. 2.23: Завершение программы

Скопировали файл lab9-2.asm, созданный при выполнении лабораторной работы №9, с программой выводящей на экран аргументы командной строки (Листинг 9.2) в файл с именем lab10-3.asm (рис. 2.24)

```

[apkim@fedora lab10]$ cp ~/work/arch-pc/lab09/lab9-2.asm ~/work/arch-pc/lab10/lab10-3.asm
[apkim@fedora lab10]$

```

Рис. 2.24: Копия файла

Создаем исполняемый файл (рис. 2.25)

```
[apkim@fedora lab10]$ nasm -f elf -g -l lab10-3.lst lab10-3.asm
[apkim@fedora lab10]$ ld -m elf_i386 -o lab10-3 lab10-3.o
[apkim@fedora lab10]$
```

Рис. 2.25: Создание исполняемого файла

Для загрузки в gdb программы с аргументами необходимо использовать ключ `-args`. Загрузили исполняемый файл в отладчик, указав аргументы. Как отмечалось в предыдущей лабораторной работе, при запуске программы аргументы командной строки загружаются в стек. Исследовали расположение аргументов командной строки в стеке после запуска программы с помощью gdb. Для начала установили точку останова перед первой инструкцией в программе и запустили ее (рис. 2.26)

```
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-3.asm, line 5.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-3 аргумент1 аргумент 2 аргумент\ 3
Breakpoint 1, _start () at lab10-3.asm:5
5      pop ecx
(gdb)
```

Рис. 2.26: Загрузка исполняемого файла в отладчик

Адрес вершины стека храниться в регистре `esp` и по этому адресу располагается число равное количеству аргументов командной строки (включая имя программы) (рис. 2.27)

```
(gdb) x/x $esp
0xffffd170: 0x00000005
(gdb)
```

Рис. 2.27: Регистр `esp`

Как видно, число аргументов равно 5 – это имя программы `lab10-3` и непосредственно аргументы: `аргумент1`, `аргумент 2` и `‘аргумент 3’`. Посмотрели 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. Шаг изменения адреса равен 4 ([esp+4], [esp+8], [esp+12] и т.д.) потому что в теле цикла next 4 строки кода. (рис. 2.28)

```
(gdb) x/s *(void**)(esp + 4)
0xffffd32a:    "/home/apkim/work/arch-pc/lab10/lab10-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd351:    "аргумент1"
(gdb) x/s *(void**)(esp + 12)
0xffffd363:    "аргумент"
(gdb) x/s *(void**)(esp + 16)
0xffffd374:    "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd376:    "аргумент 3"
(gdb) x/s *(void**)(esp + 24)
0x0:    <error: Cannot access memory at address 0x0>
(gdb) █
```

Рис. 2.28: Позиции стека

Задание для самостоятельной работы. 1 задание. Текст программы из лабораторной работы №9, реализовав вычисление значения функции как подпрограмму. (рис. 2.29)

```

GNU nano 5.8 /hom
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
    mov ebx,3
    mov esi,0
next:
    cmp ecx,0h
    jz _end
    pop eax
    call atoi
    call _func
    add esi,eax
    loop next
_end:
    mov eax,msg
    call sprint
    mov eax,esi
    call iprintLF
    call quit
_func:
    add eax, 10
    mul ebx
    ret

```

Рис. 2.29: Текст программы с подпрограммой

Создаем исполняемый файл и проверяем его работу. (рис. 2.30)

```
[apkim@fedora lab10]$ nasm -f elf lab10-4.asm
[apkim@fedora lab10]$ ld -m elf_i386 -o lab10-4 lab10-4.o
[apkim@fedora lab10]$ ./lab10-4 1 2 3
Результат: 108
[apkim@fedora lab10]$
```

Рис. 2.30: Создание исполняемого файла

Создаем файл lab10-5.asm (рис. 2.31)

```
[apkim@fedora lab10]$ touch lab10-5.asm
[apkim@fedora lab10]$ mc
```

Рис. 2.31: Создание файла

Вводим в него текст программы из листинга 10.3 (рис. 2.32)


```

GNU nano 5.8
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
mov ebx,3
mov eax,2
add ebx,eax
mov ecx,4
mul ecx
add ebx,5
mov edi,ebx
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit

```

Рис. 2.32: Текст программы из листинга 10.3

Создаем исполняемый файл и проверяем его. (рис. 2.33)

```

[apkim@fedora lab10]$ nasm -f elf -g -l lab10-5.lst lab10-5.asm
[apkim@fedora lab10]$ ld -m elf_i386 -o lab10-5 lab10-5.o
[apkim@fedora lab10]$ ./lab10-5
Результат: 10
[apkim@fedora lab10]$

```

Рис. 2.33: Создание исполняемого файла

Запустили файл в отладчик GDB. Установили точку останова, запустили код, включили режим псевдографики. (рис. 2.34)

```
apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5

[ Register Values Unavailable ]

0x80490e8 <_start>    mov     $0x3,%ebx
0x80490ed <_start+5>    mov     $0x2,%eax
0x80490f2 <_start+10>   add     %eax,%ebx
0x80490f4 <_start+12>   mov     $0x4,%ecx
0x80490f9 <_start+17>   mul     %ecx
0x80490fb <_start+19>   add     $0x5,%ebx
0x80490fe <_start+22>   mov     %ebx,%edi
0x8049100 <_start+24>   mov     $0x804a000,%eax
0x8049105 <_start+29>   call    0x804908f <sprint>
0x804910a <_start+34>   mov     %edi,%eax
0x804910c <_start+36>   call    0x8049086 <iprintLF>
0x8049111 <_start+41>   call    0x80490db <quit>

exec No process in: L?? PC: ??
(gdb) layout asm
(gdb) layout regs
(gdb) █
```

Рис. 2.34: Режим псевдографики

Пошагово проходим строчки кода (рис. 2.35)

```
apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5

Register group: general
eax      0x0      0
ecx      0x0      0
edx      0x0      0
ebx      0x3      3
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490ed 0x80490ed <_start+5>
eflags   0x202    [ IF ]
cs       0x23     35

0x80490e8 <_start> mov $0x3,%ebx
> 0x80490ed <_start+5> mov $0x2,%eax
0x80490f2 <_start+10> add %eax,%ebx
0x80490f4 <_start+12> mov $0x4,%ecx
0x80490f9 <_start+17> mul %ecx
0x80490fb <_start+19> add $0x5,%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
0x804910c <_start+36> call 0x8049086 <iprintf>
0x8049111 <_start+41> call 0x80490db <quit>
0x8049116 add %al,(%eax)

native process 6512 In: _start
(gdb) stepi
The program is not being run.
(gdb) stepi
The program is not being run.
(gdb) layout asm
(gdb) layout regs
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 7.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-5

Breakpoint 1, _start () at lab10-5.asm:7
(gdb) stepi
(gdb)
```

Рис. 2.35: Строчки кода (1)

(рис. 2.36)

```
apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5
Register group: general
eax      0x2      2
ecx      0x0      0
edx      0x0      0
ebx      0x3      3
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f2 0x80490f2 <_start+10>
eflags   0x202    [ IF ]
cs       0x23     35

0x80490e8 <_start>      mov     $0x3,%ebx
0x80490ed <_start+5>    mov     $0x2,%eax
> 0x80490f2 <_start+10> add     %eax,%ebx
0x80490f4 <_start+12>    mov     $0x4,%ecx
0x80490f9 <_start+17>    mul     %ecx
0x80490fb <_start+19>    add     $0x5,%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
0x804910c <_start+36>    call   0x8049086 <iprintLF>
0x8049111 <_start+41>    call   0x80490db <quit>
0x8049116               add     %al,(%eax)

native process 6512 In: _start L9 PC: 0x8049
The program is not being run.
(gdb) stepi
The program is not being run.
(gdb) layout asm
(gdb) layout regs
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 7.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-5

Breakpoint 1, _start () at lab10-5.asm:7
(gdb) stepi
(gdb) stepi
(gdb)
```

Рис. 2.36: Строчки кода (2)

(рис. 2.37)

```
apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5

Register group: general
eax      0x2      2
ecx      0x0      0
edx      0x0      0
ebx      0x5      5
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490f4 0x80490f4 <_start+12>
eflags   0x206    [ PF IF ]
cs       0x23     35

0x80490e8 <_start>      mov     $0x3,%ebx
0x80490ed <_start+5>     mov     $0x2,%eax
0x80490f2 <_start+10>    add     %eax,%ebx
> 0x80490f4 <_start+12>  mov     $0x4,%ecx
0x80490f9 <_start+17>    mul     %ecx
0x80490fb <_start+19>    add     $0x5,%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
0x804910c <_start+36>    call   0x8049086 <iprintfLF>
0x8049111 <_start+41>    call   0x80490db <quit>
0x8049116               add     %al,(%eax)

native process 6512 In: _start L10 PC:
(gdb) stepi
The program is not being run.
(gdb) layout asm
(gdb) layout regs
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 7.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-5

Breakpoint 1, _start () at lab10-5.asm:7
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)
```

Рис. 2.37: Строчки кода (3)

(рис. 2.38)

```
apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5

Register group: general
eax      0x2      2
ecx      0x4      4
edx      0x0      0
ebx      0x5      5
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0
esi      0x0      0
edi      0x0      0
eip      0x80490f9 0x80490f9 <_start+17>
eflags   0x206    [ PF IF ]
cs       0x23     35

0x80490e8 <_start>      mov     $0x3,%ebx
0x80490ed <_start+5>    mov     $0x2,%eax
0x80490f2 <_start+10>   add     %eax,%ebx
0x80490f4 <_start+12>   mov     $0x4,%ecx
> 0x80490f9 <_start+17> mul     %ecx
0x80490fb <_start+19>   add     $0x5,%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
0x804910c <_start+36>   call   0x8049086 <iprintLF>
0x8049111 <_start+41>   call   0x80490db <quit>
0x8049116 <_start+46>   add     %al,(%eax)

native process 6512 In: _start
The program is not being run.
(gdb) layout asm
(gdb) layout regs
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 7.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-5

Breakpoint 1, _start () at lab10-5.asm:7
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)
```

Рис. 2.38: Строчки кода (4)

(рис. 2.39)

```

apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5
Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0x5      5
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490fb 0x80490fb <_start+19>
eflags   0x202    [ IF ]
cs       0x23     35

0x80490e8 <_start>      mov     $0x3,%ebx
0x80490ed <_start+5>    mov     $0x2,%eax
0x80490f2 <_start+10>   add     %eax,%ebx
0x80490f4 <_start+12>   mov     $0x4,%ecx
0x80490f9 <_start+17>   mul     %ecx
> 0x80490fb <_start+19> add     $0x5,%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
0x804910c <_start+36>   call   0x8049086 <iprintLF>
0x8049111 <_start+41>   call   0x80490db <quit>
0x8049116 <_start+46>   add     %al,(%eax)

native process 6512 In: _start
(gdb) layout asm
(gdb) layout regs
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 7.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-5

Breakpoint 1, _start () at lab10-5.asm:7
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)

```

Рис. 2.39: Строчки кода (5)

Описываются проведённые действия, в качестве иллюстрации даётся ссылка на иллюстрацию (рис. 2.1)

```
apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0x0
esi      0x0      0
edi      0x0      0
eip      0x80490fe 0x80490fe <_start+22>
eflags   0x206    [ PF IF ]
cs       0x23     35

0x80490e8 <_start> mov $0x3,%ebx
0x80490ed <_start+5> mov $0x2,%eax
0x80490f2 <_start+10> add %eax,%ebx
0x80490f4 <_start+12> mov $0x4,%ecx
0x80490f9 <_start+17> mul %ecx
0x80490fb <_start+19> add $0x5,%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
0x804910c <_start+36> call 0x8049086 <iprintLF>
0x8049111 <_start+41> call 0x80490db <quit>
0x8049116 add %al,(%eax)

native process 6512 In: _start
(gdb) layout regs
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 7.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-5

Breakpoint 1, _start () at lab10-5.asm:7
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)
```

Рис. 2.40: Строчки кода (6)

(рис. 2.41)


```
apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa      10
esp      0xffffd1c0 0xffffd1c0
ebp      0x0      0
esi      0x0      0
edi      0xa      10
eip      0x8049100 0x8049100 <_start+24>
eflags   0x206    [ PF IF ]
cs       0x23     35

0x80490e8 <_start>    mov     $0x3,%ebx
0x80490ed <_start+5>   mov     $0x2,%eax
0x80490f2 <_start+10>  add     %eax,%ebx
0x80490f4 <_start+12>  mov     $0x4,%ecx
0x80490f9 <_start+17>   mul     %ecx
0x80490fb <_start+19>   add     $0x5,%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
0x804910c <_start+36>  call    0x8049086 <iprintf>
0x8049111 <_start+41>  call    0x80490db <quit>
0x8049116             add     %al,(%eax)

native process 6512 In: _start
(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 7.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-5

Breakpoint 1, _start () at lab10-5.asm:7
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)
```

Рис. 2.41: Строчки кода (7)

(рис. 2.42)

```

apkim@fedora:~/work/arch-pc/lab10 — gdb lab10-5

Register group: general
eax      0x804a000      134520832
ecx      0x4           4
edx      0x0           0
ebx      0xa           10
esp      0xffffd1c0    0xffffd1c0
ebp      0x0           0x0
esi      0x0           0
edi      0xa           10
eip      0x8049105     0x8049105 <_start+29>
eflags   0x206         [ PF IF ]
cs       0x23          35

0x80490e8 <_start>      mov     $0x3,%ebx
0x80490ed <_start+5>     mov     $0x2,%eax
0x80490f2 <_start+10>    add     %eax,%ebx
0x80490f4 <_start+12>    mov     $0x4,%ecx
0x80490f9 <_start+17>    mul     %ecx
0x80490fb <_start+19>    add     $0x5,%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
0x804910c <_start+36>    call    0x8049086 <iprintLF>
0x8049111 <_start+41>    call    0x80490db <quit>
0x8049116               add     %al,(%eax)

native process 6512 In: _start
Breakpoint 1 at 0x80490e8: file lab10-5.asm, line 7.
(gdb) run
Starting program: /home/apkim/work/arch-pc/lab10/lab10-5

Breakpoint 1, _start () at lab10-5.asm:7
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb) stepi
(gdb)

```

Рис. 2.42: Строчки кода (8)

Исправленный код (рис. 2.43)

```
GNU nano 5.8 /
#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 eax,5
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 2.43: Исправленный код

Запускаем исполняемый файл и проверяем его работу (рис. 2.44)

```
[apkim@fedora lab10]$ nasm -f elf -g -l lab10-5.lst lab10-5.asm
[apkim@fedora lab10]$ ld -m elf_i386 -o lab10-5 lab10-5.o
[apkim@fedora lab10]$ ./lab10-5
Результат: 25
[apkim@fedora lab10]$
```

Рис. 2.44: Исполняемый файл

3 Выводы

В ходе выполнения лабораторной работы были приобретены навыки написания программ с использованием подпрограмм, ознакомились с методами отладки при помощи GDB и его основными возможностями.

Список литературы