

Отчёта по лабораторной работе 10

Лисовская Арина Валерьевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работа	6
3	Выводы	15

Список иллюстраций

2.1	-	7
2.2	-	7
2.3	-	7
2.4	-	8
2.5	-	8
2.6	-	9
2.7	-	9
2.8	-	9
2.9	-	10
2.10	-	10
2.11	-	11
2.12	-	11
2.13	-	12
2.14	-	12
2.15	-	13
2.16	-	13
2.17	-	14
2.18	-	14

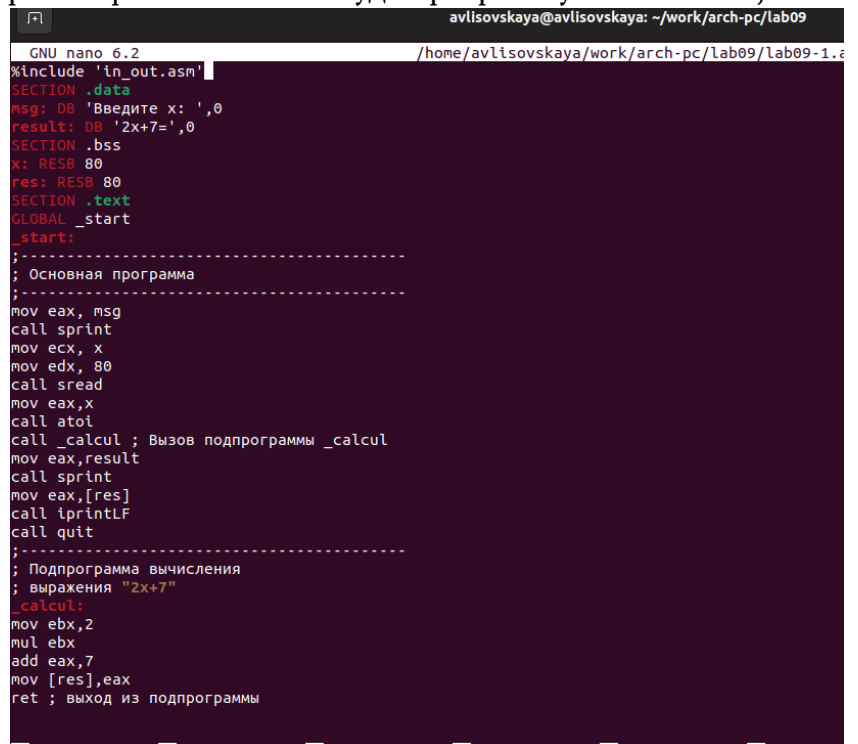
Список таблиц

1 Цель работы

Освоить работу с подпрограммами и отладчиком gdb.

2 Выполнение лабораторной работа

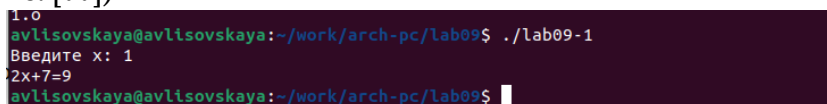
Создадим рабочую директорию и файл. Запишем туда программу из листинга,



```
GNU nano 6.2 /home/avlisovskaya/work/arch-pc/lab09/lab09-1.asm
avlisovskaya@avlisovskaya: ~/work/arch-pc/lab09
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите x: ',0
result: DB '2x+7=',0
SECTION .bss
x: RESB 80
res: RESB 80
SECTION .text
GLOBAL _start
_start:
;-----
; 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
```

исправив опечатки. (рис. [??]).

напишем программу, имитирующую сложную функцию и проверяем ее работу (рис. [??])



```
1.0
avlisovskaya@avlisovskaya: ~/work/arch-pc/lab09$ ./lab09-1
Введите x: 1
2x+7=9
avlisovskaya@avlisovskaya: ~/work/arch-pc/lab09$
```

Создадим файл lab09-2.asm и посмотрим, как она работает. Так же проассемблируем его с другими ключами, чтобы была возможность открыть этот файл через gdb. (рис. [2.1])

```

avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$ nasm -f elf lab09-2.asm
avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab09-2 lab09-2.o
avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$ ./lab09-2
Hello, world!

```

Рис. 2.1: -

Откроем lab09-2 с помощью gdb. Запустим ее там(рис. [2.2])

```

avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$ gdb lab09-2
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-2...
(gdb) run
Starting program: /home/avlisovskaya/work/arch-pc/lab09/lab09-2
Hello, world!
[Inferior 1 (process 5439) exited normally]
(gdb)

```

Рис. 2.2: -

Поставим точку останова(breakpoint) на метке _start. Посмотрим дизассемблированный код, начиная с этой метки. (рис. [2.3])

```

No symbol "_start" in current context.
(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. 2.3: -

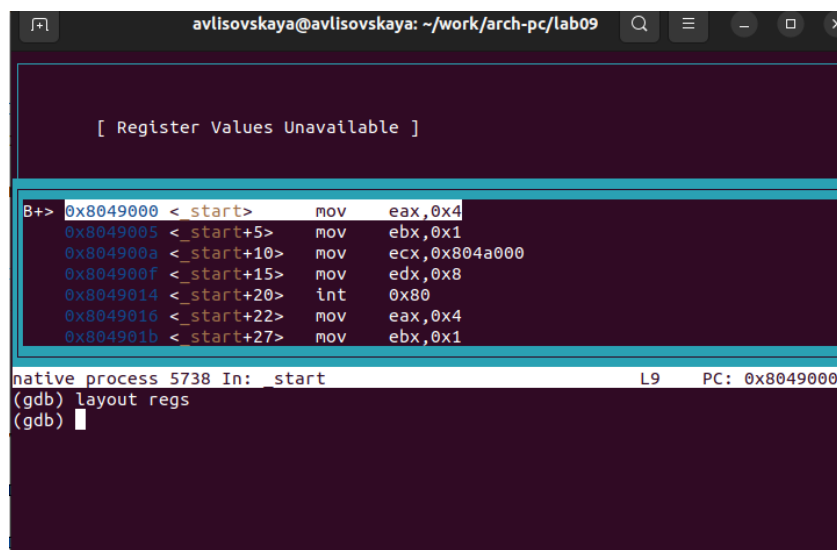
Так же посмотрим как выглядит дизассемблированный код с синтаксисом Intel (рис. [2.4])

```
End of assembler dump.
(gdb) set disassembly-flavor intel
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>:      mov     eax,0x4
   0x08049005 <+5>:      mov     ebx,0x1
   0x0804900a <+10>:     mov     ecx,0x804a000
   0x0804900f <+15>:     mov     edx,0x8
   0x08049014 <+20>:     int      0x80
   0x08049016 <+22>:     mov     eax,0x4
   0x0804901b <+27>:     mov     ebx,0x1
   0x08049020 <+32>:     mov     ecx,0x804a008
   0x08049025 <+37>:     mov     edx,0x7
   0x0804902a <+42>:     int      0x80
   0x0804902c <+44>:     mov     eax,0x1
   0x08049031 <+49>:     mov     ebx,0x0
   0x08049036 <+54>:     int      0x80
End of assembler dump.
(gdb)
```

Рис. 2.4: -

В представлении АТТ в виде 16-ричного числа записаны первые аргументы всех команд, а в представлении intel так записываются адреса вторых аргументов.

включим режим псевдографики, с помощью которого отображается код программы и содержимое регистров(рис. [2.5])



```
avlisovskaya@avlisovskaya: ~/work/arch-pc/lab09
[ Register Values Unavailable ]
B+> 0x08049000 <_start>  mov     eax,0x4
   0x08049005 <_start+5>  mov     ebx,0x1
   0x0804900a <_start+10> mov     ecx,0x804a000
   0x0804900f <_start+15> mov     edx,0x8
   0x08049014 <_start+20> int      0x80
   0x08049016 <_start+22> mov     eax,0x4
   0x0804901b <_start+27> mov     ebx,0x1
native process 5738 In: _start          L9    PC: 0x08049000
(gdb) layout regs
(gdb)
```

Рис. 2.5: -

Посмотрим информацию о наших точках останова. Сделать это можно коротко командой `i b` (рис. [2.6])

```
native process 5738 In: start L9 PC: 0x8049000
(gdb) layout regs
(gdb) i d
Ambiguous info command "d": dcache, display, dll.
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y 0x08049000 lab09-2.asm:9
        breakpoint already hit 1 time
(gdb)
```

Рис. 2.6: -

Так же можно выводить значения регистров. Делается это командой `i r`. Псевдо-

```
0x8049010 <start+22> mov    eax,0x4
0x804901b <start+27> mov    ebx,0x1

native process 3122 In: start L9 PC: 0x8049000
eax      0x0
cx       0x0
dx       0x0
bx       0x0
sp       0xffffd130 0xffffd130
bp       0x0
si       0x0
Type <RET> for more, q to quit, c to continue without paging.
```

графика представлена на (рис. [??])

В отладчике можно вывести текущее значение переменных. Сделать это можно например по имени (рис. [2.7]) или по адресу (рис. [2.7])

```
native process 5738 In: start L9 PC: 0x8049000
(gdb) i b
Num      Type      Disp Enb Address      What
1        breakpoint keep y 0x08049000 lab09-2.asm:9
        breakpoint already hit 1 time
2        breakpoint keep y 0x08049031 lab09-2.asm:20
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb)
```

Рис. 2.7: -

Так же отладчик позволяет менять значения переменных прямо во время выполнения программы (рис. [2.8])

```
(gdb) set {char}msg1='h'
'msg1' has unknown type; cast it to its declared type
(gdb) x/1sd $msg1
Value can't be converted to integer.
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb)
```

Рис. 2.8: -

Здесь тоже можно обращаться по адресам переменных(рис. [2.9]). здесь был заменен первый символ переменной msg2 на символ отступа.

```
Value can't be converted to integer.
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) set {char}&msg2=9
(gdb) x/1sb &msg2
0x804a008 <msg2>: "\torld!\n\034"
(gdb)
```

Рис. 2.9: -

Однако при попытке задать строчное значение, происходит ошибка.

Завершим работу в gdb командами continue, она закончит выполнение программы, и exit, она завершит сеанс gdb.

Скопируем файл из лабораторной 9, переименуем и создадим исполняемый файл. Откроем отладчик и зададим аргументы. Создадим точку останова на метке _start и запустим программу(рис. [2.10])

```
avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$ gdb --args lab09-3 arg1 arg 2 "ard3"
GNU gdb (Ubuntu 12.1-0ubuntu1~22.04) 12.1
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-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from lab09-3...
(gdb) b _start
Undefined command: "b_start". Try "help".
(gdb) r
Starting program: /home/avlisovskaya/work/arch-pc/lab09/lab09-3 arg1 arg 2 ard3
arg1
arg
2
ard3
[Inferior 1 (process 3719) exited normally]
(gdb)
```

Рис. 2.10: -

Посмотрим на содержимое того, что расположено по адресу, находящемуся в регистре esp (рис. [2.11])

```
(gdb) run
Starting program: /home/avlisovskaya/work/arch-pc/lab09/lab09-3 arg1 arg 2 ard3
Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffd120: 0x00000005
(gdb)
```

Рис. 2.11: -

Далее посмотрим на все остальные аргументы в стеке. Их адреса располагаются в 4 байтах друг от друга (именно столько занимает элемент стека) (рис. [2.12])

```
Breakpoint 1, _start () at lab09-3.asm:5
5      pop ecx ; Извлекаем из стека в 'ecx' количество
(gdb) x/x $esp
0xffffd120: 0x00000005
(gdb) x/s *(void**)(esp + 4)
0xffffd2fb: "/home/avlisovskaya/work/arch-pc/lab09/lab09-3"
(gdb) x/s *(void**)(esp + 8)
0xffffd329: "arg1"
(gdb) x/s *(void**)(esp + 12)
0xffffd32e: "arg"
(gdb) x/s *(void**)(esp + 16)
0xffffd332: "2"
(gdb) x/s *(void**)(esp + 20)
0xffffd334: "ard3"
(gdb) x/s *(void**)(esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)
```

Рис. 2.12: -

#Задания для самостоятельной работы Программа из лабораторной 9, но с использованием подпрограмм (рис. [2.13])

```

GNU nano 6.2 /home/avlisovskaya/work/arch-pc/lab09/self.asm
#include 'in_out.asm'
SECTION .data
f_x db "функция: 10(x - 1)",0h
msg db 10, 13, 'результат:',0h
SECTION .text
GLOBAL _start
_start:
    push ebx
    dec eax
    mov ebx,10
    mul ebx
    pop ebx
    ret

    _start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0
next:
    cmp ecx,0h
    jz _end
    pop eax
    call atoi
    call _f
    add esi, eax

    loop next

    _end:
    mov eax, f_x
    call sprint
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF

```

Рис. 2.13: -

и проверка ее работоспособности(рис. [2.14])

```

avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$ ld -m elf_i386 -o self self.o
avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$ ./self
функция: 10(x - 1)
результат:0
avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$ ./self 1 2 3 4
функция: 10(x - 1)
результат:60
avlisovskaya@avlisovskaya:~/work/arch-pc/lab09$

```

Рис. 2.14: -

Просмотр регистров, для поиска ошибки в программе из листинга 10.3 (рис. [2.15]) и (рис. [2.16])

```

Register group: general
eax      0x2      2
ecx      0x0      0
edx      0x0      0
ebx      0x5      5
esp      0xffffd0b0 0xffffd0b0
ebp      0x0      0x0

self10-1.asm
9  mov eax,2
10 add ebx,eax
> 11 mov ecx,4
12 mul ecx
13 add ebx,5
14 mov edi,ebx
15 ; ---- Вывод результата на экран

```

Рис. 2.15: -

```

Register group: general
eax      0x8      8
ecx      0x4      4
edx      0x0      0
ebx      0xa     10
esp      0xffffd0b0 0xffffd0b0
ebp      0x0      0x0

self10-1.asm
12 mul ecx
13 add ebx,5
> 14 mov edi,ebx
15 ; ---- Вывод результата на экран
16 mov eax,div
17 call sprint
18 mov eax,edi

```

Рис. 2.16: -

Ошибка была в строчках

add ebx,eax mov ecx,4 mul ecx add ebx,5 mov edi,ebx

правильно работающая программа представлена на (рис. [2.17])

```

#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. 2.17: -

Проверка корректности работы программы, после исправлений (рис. [2.18])

```

Результат: 25
[Inferior 1 (process 4710) exited normally]
(gdb) █

```

Рис. 2.18: -

3 Выводы

В результате выполнения работы, я научился организовывать код в подпрограммы и познакомился с базовыми функциями отладчика gdb.