

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №9

дисциплина: Архитектура компьютера

Холопов Илья Алексеевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	17

Список иллюстраций

2.1	Создание каталога и файла в нем lab9-1.asm	6
2.2	Содержимое файла lab9-1.asm	7
2.3	Создание и запуск исполняемого файла	8
2.4	Содержимое файла lab9-2.asm	8
2.5	Запуск исполняемого файла в отладчике gdb	9
2.6	Работа с отладчиком	10
2.7	Работа с памятью в отладчике	11
2.8	Текст программы lab8-3.asm	12
2.9	Запуск отладчика с аргументами	13
2.10	Исследование аргументов командной строки	13
2.11	Текст программы task1.asm	14
2.12	Программа, вычисляющая выражение	15
2.13	Исправленная программа	15
2.14	Создание и запуск исполняемого файла	16

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

Создадим каталог для лабораторной работы № 9, перейдем в него и создадим файл lab9-1.asm (рис. 2.1).

```
ikhologov0323@ikhologov0323:~$ mkdir work/arch-pc/lab09
ikhologov0323@ikhologov0323:~$ cd work/arch-pc/lab09
ikhologov0323@ikhologov0323:~/work/arch-pc/lab09$ touch lab9-1.asm
ikhologov0323@ikhologov0323:~/work/arch-pc/lab09$
```

Рис. 2.1: Создание каталога и файла в нем lab9-1.asm

рассмотрим программу вычисления арифметического выражения $f(x) = 2g(x) + 7$, где $g(x) = 3x - 1$ с помощью подпрограмм `_calcul` и `_subcalcul`. В данном примере x вводится с клавиатуры, а само выражение вычисляется в подпрограмме (рис. 2.2).

```

/home/ikhologov0323-c/lab09/lab9-1.asm      889/894      99%
#include 'in_out.asm'
SECTION .data
    msg: DB 'Введите x: ',0
    result: DB 'g(x)=3x-1',10,'2g(x)+7=',0
SECTION .bss
    x: RESB 80
    res: 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 ; Вызов подпрограммы _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

    mov eax, msg ; вызов подпрограммы печати сообщения
    call sprint ; 'Введите x: '
    mov ecx, x
    mov edx, 80
    call sread ; вызов подпрограммы ввода сообщения
    mov eax, x ; вызов подпрограммы преобразования
    call atoi
    mov ebx, 2
    mul ebx
    add eax, 7
    mov [res], eax

```

Рис. 2.2: Содержимое файла lab9-1.asm

Создадим исполняемый файл и запустим его (рис. 2.3).

```

ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ nasm -f elf32 lab9-1.asm
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-1 la
b9-1.o
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 3
f(x)=2x+7
g(x)=3x-1
f(g(x))=23
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ ./lab9-1
Введите x: 4
f(x)=2x+7
g(x)=3x-1
f(g(x))=29
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$

```

Рис. 2.3: Создание и запуск исполняемого файла

Создадим файл lab9-2.asm а запишем в него программу печати сообщения “Hello world!” (рис. 2.4).

```

/home/ikholopov0323/w-h-pc/lab09/lab9-2.asm 312/312
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. 2.4: Содержимое файла lab9-2.asm

Создадим исполняемый файл с отладочной информацией и запустим его в отладчике gdb(рис. 2.5).


```

ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-2.ls
t lab9-2.asm
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-2 la
b9-2.o
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ gdb lab9-2
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 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 lab9-2...
(gdb) run
Starting program: /home/ikholopov0323/work/arch-pc/lab09/lab9-2
Hello, world!
[Inferior 1 (process 2110) exited normally]
(gdb) █

```

Рис. 2.5: Запуск исполняемого файла в отладчике gdb

Создадим точку останова, посмотрим дисассимилированный код программы, переключимся на отображение команд с Intel'овским синтаксисом и включим режим псевдографики для более удобного анализа программы (рис. 2.6).

```

(gdb) run
Starting program: /home/ikholopov0323/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:9
9      mov eax, 4
(gdb) disassemble _start
Dump of assembler code for function _start:
=> 0x08049000 <+0>: mov $0x4,%eax
    0x08049005 <+5>: mov $0x1,%ebx
    0x0804900a <+10>: mov $0x804a000,%ecx
    0x0804900f <+15>: mov $0x8,%edx
    0x08049014 <+20>: int $0x80
    0x08049016 <+22>: mov $0x4,%eax
    0x0804901b <+27>: mov $0x1,%ebx
    0x08049020 <+32>: mov $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) 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) layout asm

```

Рис. 2.6: Работа с отладчиком

Установим точку останова по адресу выведем и изменим содержимое памяти (2.7).

```
0x8049da2 add BYTE PTR [eax],al
0x8049da4 add BYTE PTR [eax],al
0x8049da6 add BYTE PTR [eax],al
0x8049da8 add BYTE PTR [eax],al
0x8049daa add BYTE PTR [eax],al
0x8049dac add BYTE PTR [eax],al
0x8049dae add BYTE PTR [eax],al
0x8049db0 add BYTE PTR [eax],al
0x8049db2 add BYTE PTR [eax],al
0x8049db4 add BYTE PTR [eax],al
0x8049db6 add BYTE PTR [eax],al
0x8049db8 add BYTE PTR [eax],al
0x8049dba add BYTE PTR [eax],al
0x8049dbc add BYTE PTR [eax],al
0x8049dbe add BYTE PTR [eax],al
0x8049dc0 add BYTE PTR [eax],al
0x8049dc2 add BYTE PTR [eax],al
0x8049dc4 add BYTE PTR [eax],al
0x8049dc6 add BYTE PTR [eax],al
0x8049dc8 add BYTE PTR [eax],al
0x8049dca add BYTE PTR [eax],al
0x8049dcc add BYTE PTR [eax],al
0x8049dce add BYTE PTR [eax],al
0x8049dd0 add BYTE PTR [eax],al
0x8049dd2 add BYTE PTR [eax],al
0x8049dd4 add BYTE PTR [eax],al
0x8049dd6 add BYTE PTR [eax],al
0x8049dd8 add BYTE PTR [eax],al
0x8049dda add BYTE PTR [eax],al
0x8049ddc add BYTE PTR [eax],al

native process 2238 In: start L20 PC: 0x8049031
(gdb) r
Starting program: /home/ikholopov0323/work/arch-pc/lab09/lab9-2

Breakpoint 1, _start () at lab9-2.asm:20
(gdb) x/1sb &msg1
0x804a000 <msg1>: "Hello, "
(gdb) x/1sb &msg2
0x804a008 <msg2>: "world!\n\034"
(gdb) set {char}&msg1='h'
Value can't be converted to integer.
(gdb) set {char}&msg1='h'
(gdb) x/1sb &msg1
0x804a000 <msg1>: "hello, "
(gdb) set {char}&msg2='W'
(gdb) x/1sb &msg2
0x804a008 <msg2>: "World!\n\034"
(gdb) █
```

Рис. 2.7: Работа с памятью в отладчике

Выведем в различных форматах (в шестнадцатеричном формате, в двоичном формате и в символьном виде) значение регистра `ebx`, завершим выполнение программы и выйдем из отладчика (рис. 2.8).

```
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 No process In:                                     L??  PC: ??
(gdb) set $ebx='2'
(gdb) p/s $ebx
$1 = 50
(gdb) p/t $ebx
$2 = 110010
(gdb) p/x $ebx
$3 = 0x32
(gdb) c
Continuing.
Hello, world!
(gdb) or 1 (process 2264) exited normally]
```

Рис. 2.8: Текст программы lab8-3.asm

Скопируем файл lab8-2.asm, созданный при выполнении лабораторной работы №8, с программой выводящей на экран аргументы командной строки в файл с именем lab09-3.asm. Создадим исполняемый файл и запустим отладчик с аргументами (рис. 2.9).

```

ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ cp ../lab08/lab8-2.asm lab9-3.asm
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ nasm -f elf -g -l lab9-3.lst lab9-3.asm
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ d -m elf_i386 -o lab9-3 lab9-3.o
d: command not found
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ ld -m elf_i386 -o lab9-3 lab9-3.o
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ gdb --args lab9-3 аргумент1 аргумент2 'аргумент 3'
GNU gdb (Debian 10.1-1.7) 10.1.90.20210103-git
Copyright (C) 2021 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 lab9-3...
(gdb)

```

Рис. 2.9: Запуск отладчика с аргументами

Исследуем расположение аргументов командной строки в стеке после запуска программы с помощью gdb. Шаг изменения адреса равен 4, так как программа 32-битная (рис. 2.10).

```

(gdb) b _start
Breakpoint 1 at 0x80490e8: file lab9-3.asm, line 5.
(gdb) run
Starting program: /home/ikholopov0323/work/arch-pc/lab09/lab9-3 аргумент1 аргумент2 аргумент\ 3

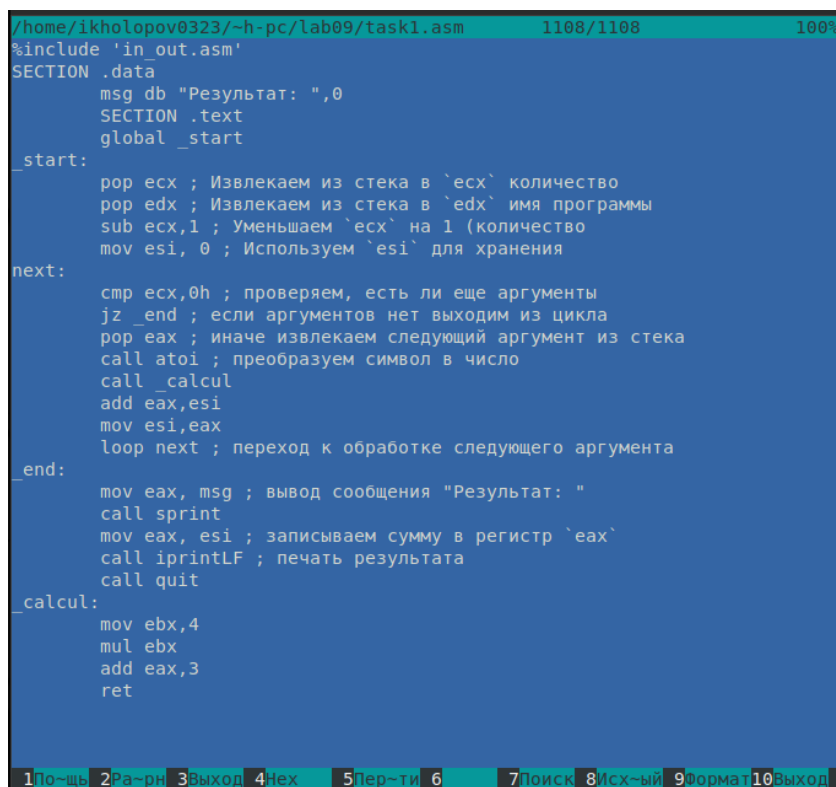
Breakpoint 1, _start () at lab9-3.asm:5
5      pop ecx ; Извлекаем из стека в `ecx` количество
(gdb) x/x $esp
0xffffd1d0: 0x00000005
(gdb) x/s *(void**)($esp + 4)
0xffffd379: "/home/ikholopov0323/work/arch-pc/lab09/lab9-3"
(gdb) x/s *(void**)($esp + 8)
0xffffd3a7: "аргумент1"
(gdb) x/s *(void**)($esp + 12)
0xffffd3b9: "аргумент"
(gdb) x/s *(void**)($esp + 16)
0xffffd3ca: "2"
(gdb) x/s *(void**)($esp + 20)
0xffffd3cc: "аргумент 3"
(gdb) x/s *(void**)($esp + 24)
0x0: <error: Cannot access memory at address 0x0>
(gdb)

```

Рис. 2.10: Исследование аргументов командной строки

Преобразуем программу из лабораторной работы №8 (Задание №1 для самостоятельной работы), реализовав вычисление значения функции $f(x)$ как

подпрограмму (рис. 2.11).



```
/home/ikholopov0323/~h-pc/lab09/task1.asm 1108/1108 100%
#include 'in_out.asm'
SECTION .data
    msg db "Результат: ",0
SECTION .text
    global _start

_start:
    pop ecx ; Извлекаем из стека в `ecx` количество
    pop edx ; Извлекаем из стека в `edx` имя программы
    sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
    mov esi, 0 ; Используем `esi` для хранения

next:
    cmp ecx,0h ; проверяем, есть ли еще аргументы
    jz _end ; если аргументов нет выходим из цикла
    pop eax ; иначе извлекаем следующий аргумент из стека
    call atoi ; преобразуем символ в число
    call _calcul
    add eax,esi
    mov esi,eax
    loop next ; переход к обработке следующего аргумента

_end:
    mov eax, msg ; вывод сообщения "Результат: "
    call sprint
    mov eax, esi ; записываем сумму в регистр `eax`
    call iprintLF ; печать результата
    call quit

_calcul:
    mov ebx,4
    mul ebx
    add eax,3
    ret
```

Рис. 2.11: Текст программы task1.asm

Создадим файл task2.asm, запишем в файл текст программы, вычисляющей выражения $(3 + 2) * 4 + 5$ (рис. 2.12).

```

/home/ikholopov0323/~h-pc/lab09/task2.asm 252/252 100%
#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. 2.12: Программа, вычисляющая выражение

При запуске данная программа дает неверный результат. С помощью отладчика GDB, анализируя изменения значений регистров, определим ошибку и исправим ее (рис. 2.13).

```

/home/ikholopov0323/~h-pc/lab09/task2.asm 252/252 100%
#include 'in_out.asm'
SECTION .data
    div: DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:
    mov ebx,3
    mov eax,2
    add 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. 2.13: Исправленная программа

Создадим и запустим исполняемый файл (рис. 2.14)

```
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ nasm -f elf -g -l task2.l  
st task2.asm  
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ ld -m elf_i386 -o task2 t  
ask2.o  
ikholopov0323@ikholopov0323:~/work/arch-pc/lab09$ ./task2  
Результат: 25
```

Рис. 2.14: Создание и запуск исполняемого файла

3 Выводы

В ходе выполнения лабораторной работы были приобретены навыки написания программ с использованием подпрограмм. Также были изучены методы отладки при помощи GDB и его основные возможности.