

**Лабораторная работа №6. Основы
работы с Midnight Commander (mc).
Структура программы на языке
ассемблера NASM.**

Дисциплина: Архитектура ЭВИ

Осокин Георгий Иванович НММбд-02-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Откроем через терминал MidnightCommander	6
2.2	Для своего удобства я изменю текстовый редактор по умолчанию на VIM	8
2.3	Введем текст программы из листинга	10
2.4	Подключим внешний файл	12
2.5	Изменим содержимое lab6-2.asm, используя подпрограммы из in_out.asm	14
3	Задания для самостоятельной работы	16
3.1	Создадим копию lab-1.asm с выводом ввода	16
3.2	Создадим копию lab-2.asm с выводом ввода	18
4	Выводы	19
	Список литературы	20

Список иллюстраций

2.1	Открытие Midnight Commander	6
2.2	Создание папки	7
2.3	Создание файла	7
2.4	Открытие настроек МС	8
2.5	Изменение конфигурации МС	8
2.6	Открытие текстового редактора через МС	9
2.7	Введенный текст на языке ассемблера	10
2.8	Просмотр содержимого через МС	11
2.9	Трансляция линковка и запуск lab6-1.asm	11
2.10	Открытие МС	12
2.11	Просмотр папки Downloads	12
2.12	Копирование через МС	12
2.13	Скопированный файл	13
2.14	Копирование в lab6-2.asm	13
2.15	Просмотр скопированного файла	14
2.16	Отредактированный lab6-2.asm	14
2.17	Новая копия	15
2.18	Замена подпрограммы	15
2.19	Запуск двух программ	15
3.1	Создание копии	16
3.2	Новый блок кода	16
3.3	Запуск lab-1	16
3.4	Замена buf на msg+msgLen	17
3.5	Запуск программы	17
3.6	Отнятие единицы	17
3.7	Исполнение измененной программы	17
3.8	Создание копии	18
3.9	Добавление новых строчек	18
3.10	Запуск lab-2-сору	18

Список таблиц

1 Цель работы

Приобретение практических навыков работы в Midnight Commander. Освоение

2 Выполнение лабораторной работы

2.1 Откроем через терминал MidnightCommander

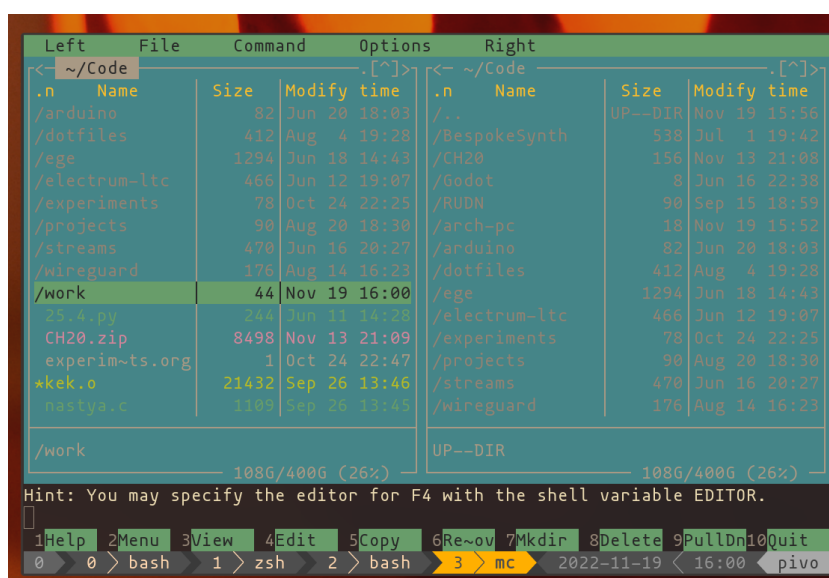


Рис. 2.1: Открытие Midnight Commander

Перейдем в каталог `./work/arch-pc` и создадим папку `./lab06`

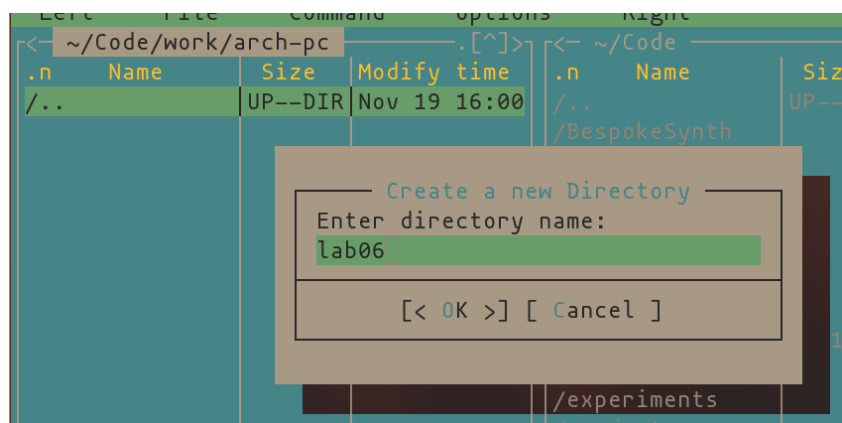


Рис. 2.2: Создание папки

Наберем в строке ввода `touch lab6.asm`

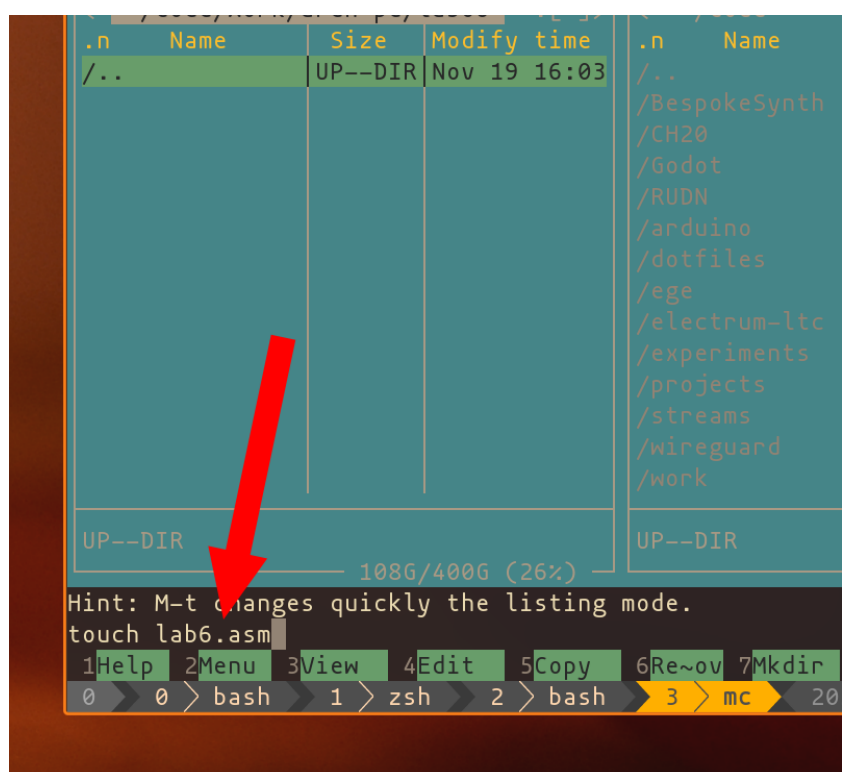


Рис. 2.3: Создание файла

2.2 Для своего удобства я изменю текстовый редактор по умолчанию на VIM

Зайдем в настройки, нажав F9, о, с

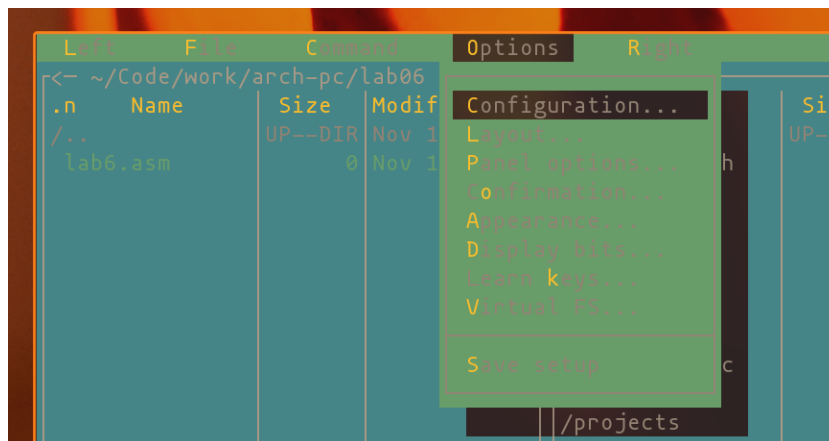


Рис. 2.4: Открытие настроек MC

Нажмем t что бы снять галочку с “Use internal edit”

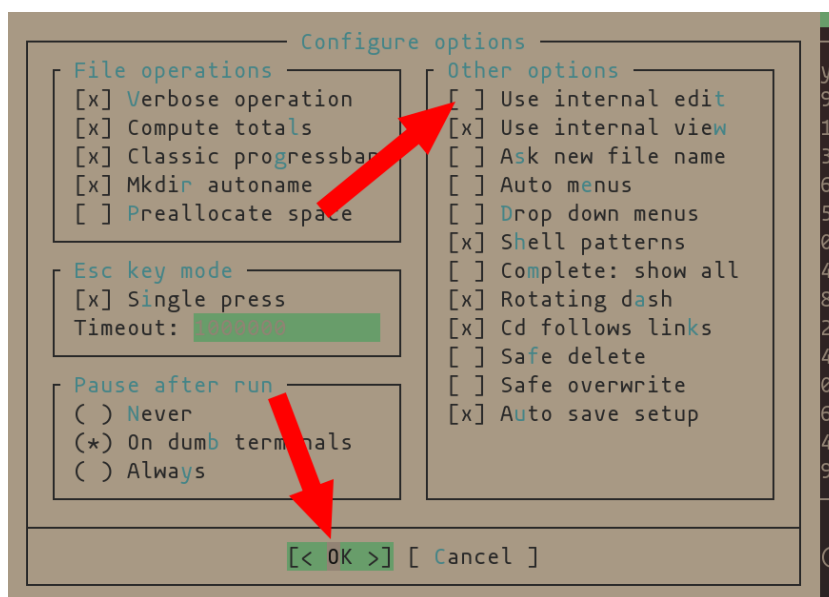


Рис. 2.5: Изменение конфигурации MC

Теперь, когда мы нажимаем F4 в MC, открывается текстовый редактор VIM



Рис. 2.6: Открытие текстового редактора через MC

2.3 Введем текст программы из листинга

```
1 lab6.asm
>> 12 ; input output string program
11
>> 10 ; --- Variable declaration --- ;
9
>> 8 SECTION .data ;initializing data section
>> 7 msg: DB 'Введите строку:', 10 ; message + \n
>> 6 msgLen: EQU $-msg ; len of 'msg'
>> 5
>> 4 SECTION .bss ; no initialized data section
>> 3 buf1: RESB 80 ; 80 bytes size buff
>> 2
>> 1 ; --- Programm text --- ;
>>13
>> 1 SECTION .text ; Code itself
>> 2 GLOBAL _start ; Beginning of the programm
>> 3 _start: ; entry point
>> 4
>> 5 ; --- `write` syscall --- ;
>> 6 ; printing string from 'msg' with len 'msgLen'
>> 7
>> 8 mov eax, 4 ; Syscal for `sys_write`
>> 9 mov ebx, 1 ; descriptor - std output
>> 10 mov ecx, msg ; mov addres of 'msg' to 'ecx'
>> 11 mov edx, msgLen ; mov size of 'msg' to 'edx'
>> 12 int 80h ; kernel call
>> 13
>> 14 ; --- `read` syscall --- ;
>> 15 mov eax, 3 ; syscall for `sys_read`
>> 16 mov ebx, 0 ; descriptor 0 - std input
>> 17 mov ecx, buf1 ; mov buffer addres for input to ecx
>> 18 mov edx, 80 ; mov string len to edx
>> 19 int 80h ; kernel call
>> 20
>> 21 ; --- `exit` syscall ---;
>> 22
>> 23 mov eax, 1 ; syscall for `sys_exit`
>> 24 mov ebx, 0 ; move 0 return code to ebx
>> 25 int 80h ; kernel call
>> 26

NORMAL lab6.asm
0 > 0 > bash 1 > zsh 2 > bash 3 > mc
```

Рис. 2.7: Введенный текст на языке ассемблера

Посмотрим содержимое измененного и сохраненного файла через МС, нажав F3

```

/home/horhik/Code/work/arch-pc/lab06/lab6.asm
; input output string program

; --- Variable declaration --- ;

SECTION .data                ;initializing data section
msg:  DB 'Введите строку:', 10 ; message + \n
msgLen: EQU $-msg            ; len of 'msg'

SECTION .bss                 ; no initialized data section
buf1:  RESB 80               ; 80 bytes size buff

; --- Programm text --- ;

SECTION .text                ; Code itsel
GLOBAL _start                ; Beginning of the programm
_start:                      ; entry point

; --- `write` syscall --- ;
; printing string from 'msg' with len 'msgLen'

mov eax, 4                   ; Syscal for `sys_write`
mov ebx, 1                   ; descriptor - std output
mov ecx, msg                 ; mov addres of 'msg' to 'ecx'
mov edx, msgLen              ; mov size of 'msg' to 'edx'
int 80h                     ; kernel call

; --- `read` syscall --- ;
mov eax, 3                   ; syscall for `sys_read`
mov ebx, 0                   ; description 0 - std input
mov ecx, buf1                ; mov buffer addres for input to ecx
mov edx, 80                  ; mov string len to edx
int 80h                     ; kernel call

; --- `exit` syscall ---;
mov eax, 1                   ; syscall for `sys_exit`
mov ebx, 0                   ; move 0 return code to ebx
int 80h                     ; kernel call

1Help      2UnWrap    3Quit      4Hex      5Goto      6
0 > 0 > bash  1 > zsh   2 > bash   3 > mc

```

Рис. 2.8: Просмотр содержимого через МС

Оттранслируем, слинкуем файл lab6-1.asm (переместив lab6 в lab6-1). Запустим исполнимый файл.

```

[giosokin:lab06]$ mv lab6.asm lab6-1.asm
[giosokin:lab06]$ nasm -f elf lab6-1.asm
[giosokin:lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[giosokin:lab06]$ ./lab6-1
Введите строку:
giosokin
[giosokin:lab06]$ 

```

Рис. 2.9: Трансляция линковка и запуск lab6-1.asm

2.4 Подключим внешний файл

Откроем MC



Рис. 2.10: Открытие MC

Найдем файл `in_out.asm` в `~/Downloads`

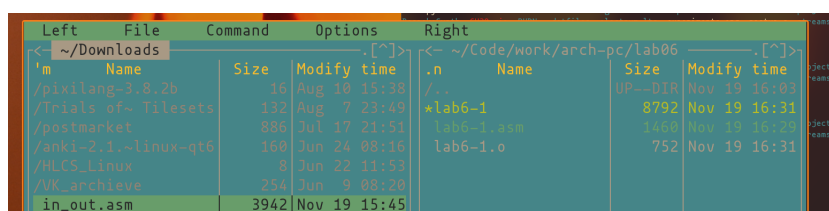


Рис. 2.11: Просмотр папки Downloads

Скопируем файл в `lab5`, нажав `F5`

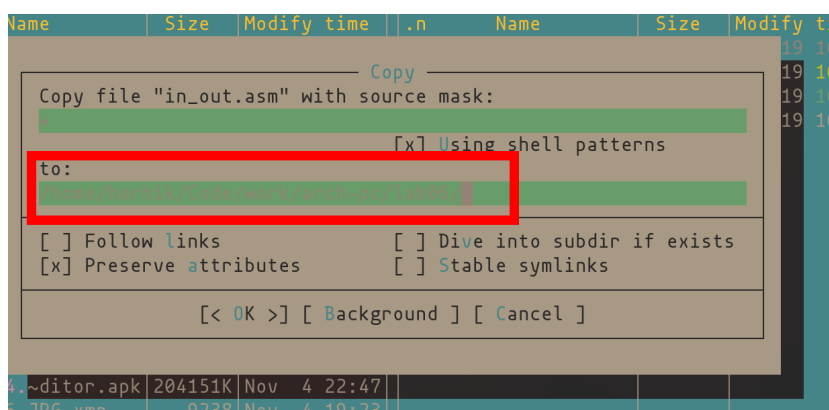


Рис. 2.12: Копирование через MC

Как видим, файл появился в директории `lab06`

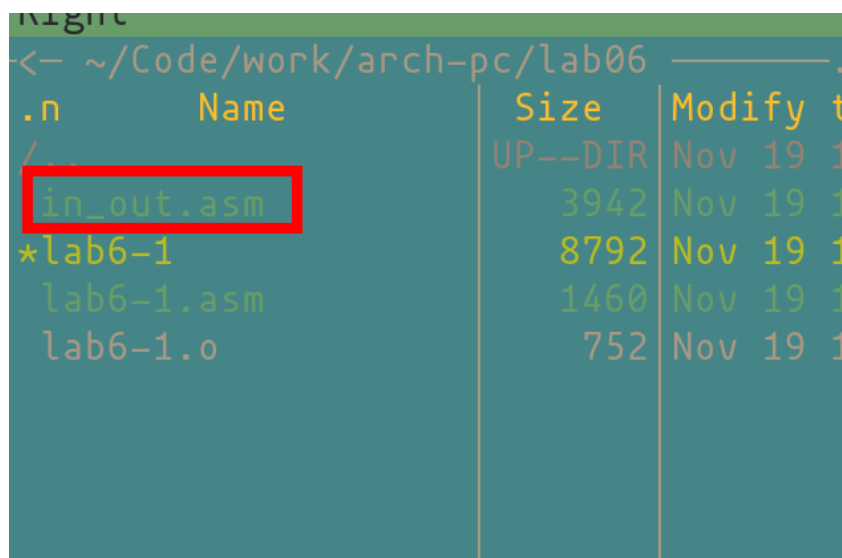


Рис. 2.13: Скопированный файл

Скопируем файл в lab6-2.asm

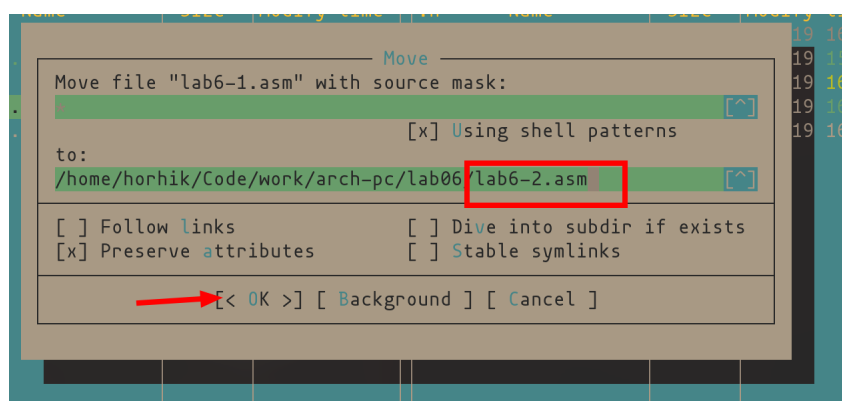


Рис. 2.14: Копирование в lab6-2.asm

Как видим, файл скопировался

NAME	SIZE
../in_out.asm	3942
*lab6-1	8792
lab6-1.asm	1460
lab6-1.o	752
lab6-2.asm	1460

Рис. 2.15: Просмотр скопированного файла

2.5 Изменим содержимое lab6-2.asm, используя подпрограммы из in_out.asm

```

lab6-2.asm
9 ; input output string program
8
7 %include 'in_out.asm'          ; including external file
6
5 ; --- Variable declaration --- ;
4
3 SECTION .data                  ;initializing data section
2 msg:  DB 'Введите строку:', 10 ; message + \n
1
10 SECTION .bss                  ; no initialized data section
1 buf1:  RESB 80                  ; 80 bytes size buff
2
3 ; --- Programm text --- ;
4
5 SECTION .text                  ; Code itsel
6 GLOBAL _start                  ; Beginning of the programm
7 _start:                        ; entry point
8
9 ; --- `write` syscall --- ;
10 ; printing string from 'msg' with len 'msgLen'
11
12 mov  eax, msg                  ; moving 'msg' addres to eax
13 call sprintf                   ; call subprogramm for printing
14
15 mov  ecx, buf1                 ; moving 'buf1' addres to ecx
16 mov  edx, 80                  ; mov string len to edx
17 call read                      ; call subprogramm for output
18
19
20 call quit                      ; call subprogram for quit
21

```

Рис. 2.16: Отредактированный lab6-2.asm

Создадим копию lab6-2.1.asm



Рис. 2.17: Новая копия

Заменим sprintf на printf



Рис. 2.18: Замена подпрограммы

Странслируем, линкуем и запускаем две эти программы

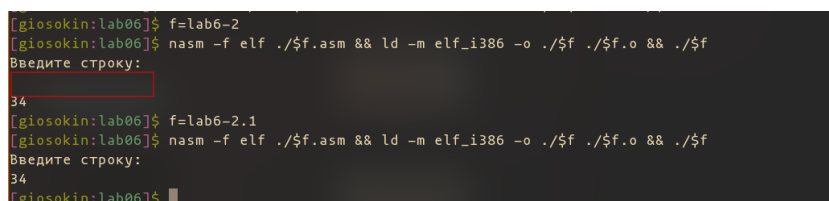


Рис. 2.19: Запуск двух программ

Как видим, у первой есть перенос строки, у второй он отсутствует

3 Задания для самостоятельной работы

3.1 Создадим копию lab-1.asm с выводом ввода

```
[giosokin:lab06]$ cp lab6-1.asm lab6-1-copy.asm  
[giosokin:lab06]$ vim lab6-1-copy.asm
```

Рис. 3.1: Создание копии

Откроем новый файл и добавим в него блок с выводом строки из buf1

```
8 ; --- `write` syscall --- ;  
7 ; printing inputted string from 'msg' with len 'msgLen'  
6  
5 mov eax, 4 ; Syscall for `sys_write`  
4 mov ebx, 1 ; descriptor - std output  
3 mov ecx, buf1  
2 mov edx, 80 ; mov size of 'msg' to 'edx'  
1 int 80h ; kernel call  
2  
1
```

Рис. 3.2: Новый блок кода

Скомпилируем и запустим

```
[giosokin:lab06]$ nasm -f elf ./lab6-1-copy.asm && ld -m elf_i386 -o ./lab6-1-copy.o && ./lab6-1-copy.o  
Введите строку:  
Осокин Георгий  
Осокин Георгий  
[giosokin:lab06]$
```

Рис. 3.3: Запуск lab-1

Заметим, что если заменить buf1 на msg+msgLen то результат будет таким же. Так как buf1 инициализируется в памяти сразу после msg и размер msg это msgLen. Поэтому, адрес msg+msgLen в точности адрес buf1


```

; --- `write` syscall --- ;
; printing inputted string from 'msg' with

mov eax, 4                ; Sys
mov ebx, 1                ; des
mov ecx, msg+msgLen       ; inc
                           ; and
mov edx, 80               ; mov
int 80h                   ; ker

```

Рис. 3.4: Замена buf на msg+msgLen

Скомпилируя этот файл мы получим такой же результат

```

[giosokin:lab06]$ vim lab06_1_copy.asm
[giosokin:lab06]$ nasm -f elf ./$.asm && ld -m elf_i386 -o ./$.f ./$.o && ./$.f
Введите строку:
Осокин Георгий!
Осокин Георгий!
[giosokin:lab06]$

```

Рис. 3.5: Запуск программы

Отнимем 1 и посмотрим что получится

```

mov ebx, 1                ;
mov ecx, msg+msgLen -1    ;
mov edx, 80               ;

```

Рис. 3.6: Отнятие единицы

Исполним

```

[giosokin:lab06]$ nasm -f elf ./$.asm && ld -m elf_i386 -o ./$.f ./$.o && ./$.f
Введите строку:
Осокин Георгий
Осокин Георгий
[giosokin:lab06]$ nasm -f elf ./$.asm && ld -m elf_i386 -o ./$.f ./$.o && ./$.f

```

Рис. 3.7: Исполнение измененной программы

Мы захватили с собой перенос строки, которым заканчивался текст msg

3.2 Создадим копию lab-2.asm с выводом ввода

```
[giosokin:lab06]$ cp lab6-2.asm lab6-2-copy.asm  
[giosokin:lab06]$ vim lab6-2-copy.asm
```

Рис. 3.8: Создание копии

Добавим две строки, первая будет перемещать значение из буфера в eax
Вторая исполнит подпрограмму вывода

```
6  
5    mov    eax, buf1  
4    call   sprintf  
3  
2
```

Рис. 3.9: Добавление новых строчек

Скомпилируем и запустим

```
[giosokin:lab06]$ f=lab6-2-copy  
[giosokin:lab06]$ nasm -f elf ./f.asm && ld -m elf_i386 -o ./f ./f.o && ./f  
Введите строку:  
Осокин Георгий  
Осокин Георгий  
[giosokin:lab06]$
```

Рис. 3.10: Запуск lab-2-copy

4 Выводы

В ходе данной лабораторной работы мы научились пользоваться Midnight Commander и освоили базовые инструкции языка ассемблера mov и int.

Список литературы