

Лабораторная работа №8

Отчет

Зубов Иван Александрович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Самостоятельная работа	13
5	Выводы	15

Список иллюстраций

3.1	Создаем файл	7
3.2	Заполняем файл	7
3.3	Запускаем файл и смотрим на его работу	8
3.4	Редактируем файл	8
3.5	Запускаем файл и смотрим на его работу	8
3.6	Редактируем файл	9
3.7	Запускаем файл и смотрим на его работу	9
3.8	Создаем файл	10
3.9	Заполняем файл	10
3.10	Смотрим, что получается	10
3.11	Создаем файл листинга	10
3.12	Заполняем файл	11
3.13	Создаем объектный файл и проверяем работу программы	11
3.14	Редактируем файл	12
3.15	Создаем объектный файл и проверяем работу программы	12
4.1	Пишем программу	14
4.2	Смотрим, что все получилось	14

Список таблиц

1 Цель работы

Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

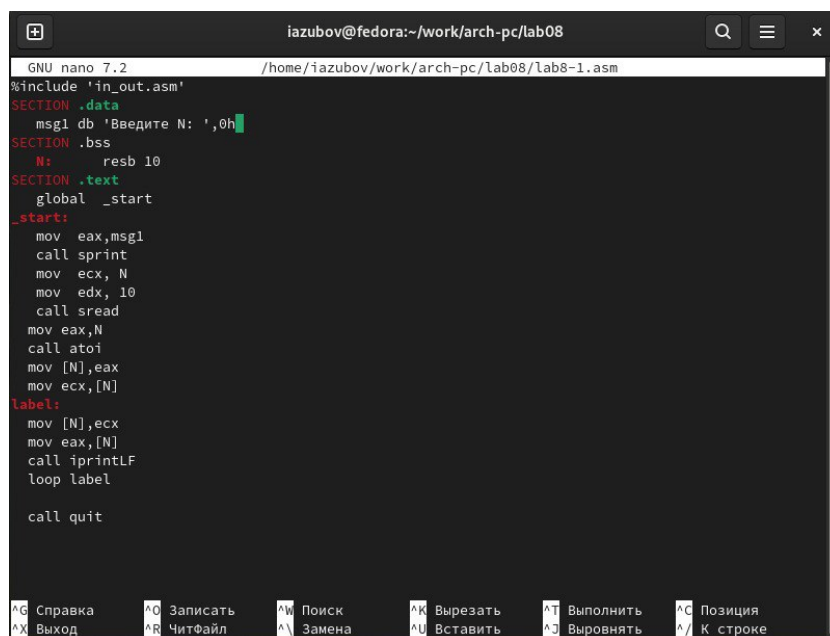
Написать программы с использованием циклов и обработкой аргументов командной строки.

3 Выполнение лабораторной работы

Создаем каталог для программ лабораторной работы № 8 с помощью команды `mkdir`, перейдем в него и создадим файл `lab8-1.asm` с помощью команды `touch`. Откроем файл в Midnight Commander и заполняем его в соответствии с листингом 8.1

```
iazubov@fedora:~$ mkdir ~/work/arch-pc/lab08
iazubov@fedora:~$ cd ~/work/arch-pc/lab08
iazubov@fedora:~/work/arch-pc/lab08$ touch lab8-1.asm
```

Рис. 3.1: Создаем файл



```
GNU nano 7.2 /home/iazubov/work/arch-pc/lab08/lab8-1.asm
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx, N
mov edx, 10
call sread
mov eax,N
call atoi
mov [N],eax
mov ecx,[N]
label:
mov [N],ecx
mov eax,[N]
call iprintLF
loop label
call quit
```

Рис. 3.2: Заполняем файл

Создаем исполняемый файл и запускаем его

```
iazubov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
iazubov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
iazubov@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
iazubov@fedora:~/work/arch-pc/lab08$
```

Рис. 3.3: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его



```
GNU nano 7.2 /home/iazubov/work/arch-pc/lab08/lab8-1.asm
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,N
mov edx,10
call sread
mov eax,N
call atoi
mov [N],eax
mov ecx,[N]
label:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintfLF
loop label
call quit
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^D Выводить ^/_ К строке

Рис. 3.4: Редактируем файл

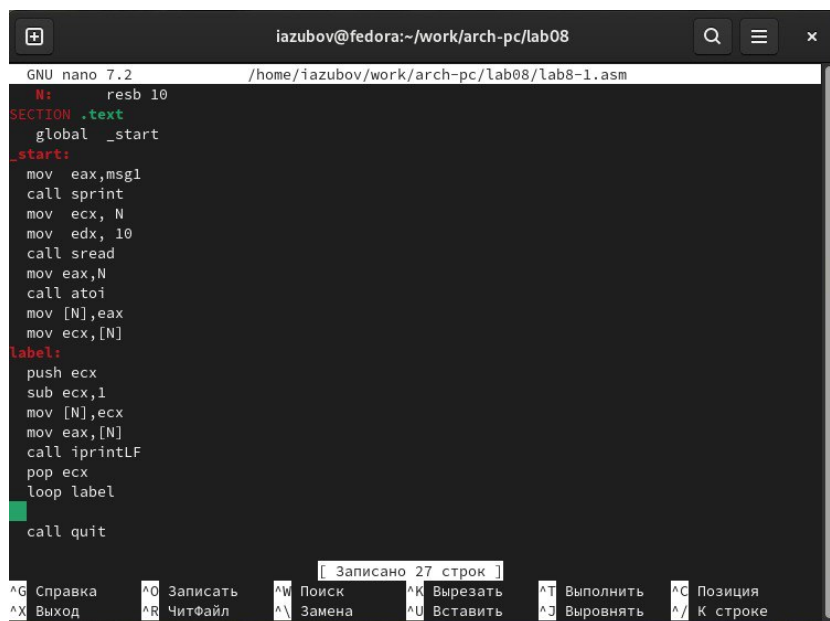
Создаем исполняемый файл и запускаем его

```
iazubov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
iazubov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
iazubov@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
iazubov@fedora:~/work/arch-pc/lab08$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

Регистру `ecx` присваиваются значения 9 7 5 3 1 - регистр уменьшается на 2. Число проходов не соответствует числу `N`, из-за уменьшения на 2.

Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле



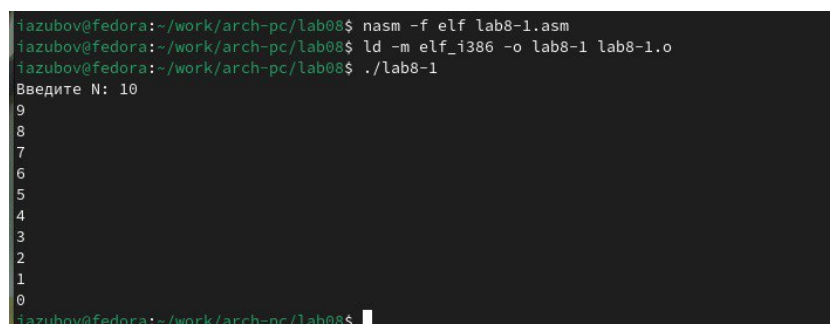
```
GNU nano 7.2 /home/iazubov/work/arch-pc/lab08/lab8-1.asm
N:      resb 10
SECTION .text
global _start
_start:
mov     eax,msg1
call    printf
mov     ecx, N
mov     edx, 10
call    sread
mov     eax,N
call    atoi
mov     [N],eax
mov     ecx,[N]
label:
push    ecx
sub     ecx,1
mov     [N],ecx
mov     eax,[N]
call    iprintLF
pop     ecx
loop    label
call    quit
```

[Записано 27 строк]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/_ К строке

Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его



```
iazubov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
iazubov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
iazubov@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
iazubov@fedora:~/work/arch-pc/lab08$
```

Рис. 3.7: Запускаем файл и смотрим на его работу

Создаем новый файл с помощью команды `touch`, открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2

```
iazubov@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
```

Рис. 3.8: Создаем файл



```
GNU nano 7.2 /home/iazubov/work/arch-pc/lab08/lab8-2.asm
#include 'in_out.asm'
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
next:
    cmp ecx,0
    jz _end
    pop eax
    call printf
    loop next
_end:
    call quit
```

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения

```
iazubov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
iazubov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
iazubov@fedora:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
iazubov@fedora:~/work/arch-pc/lab08$
```

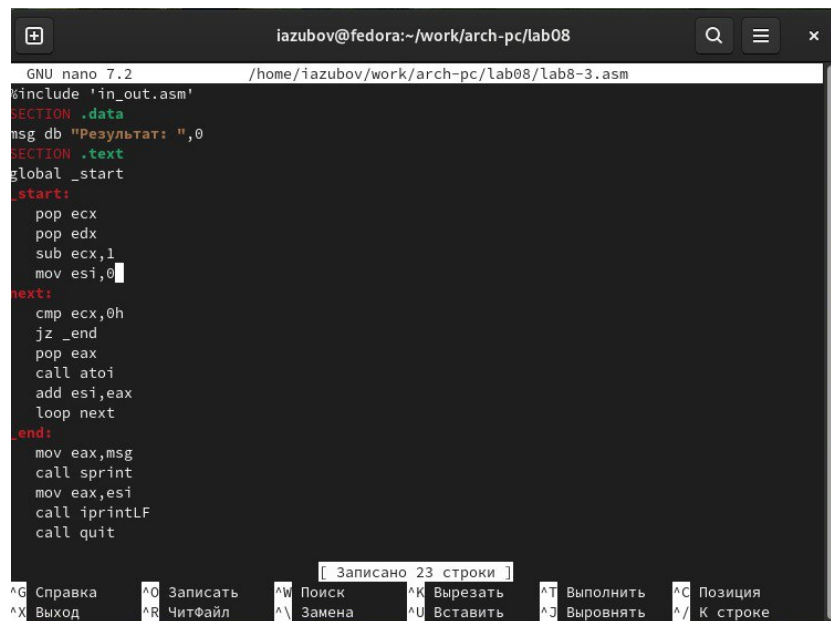
Рис. 3.10: Смотрим, что получается

Программа обрабатывает 3 аргумента

Создаем файл lab8-3.asm, вводим в него текст программы из листинга 8.3

```
iazubov@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
```

Рис. 3.11: Создаем файл листинга



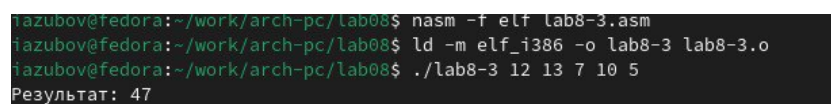
```
GNU nano 7.2 /home/iazubov/work/arch-pc/lab08/lab8-3.asm
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
    mov esi,6
next:
    cmp ecx,0h
    jz _end
    pop eax
    call atoi
    add esi,eax
    loop next
_end:
    mov eax,msg
    call sprint
    mov eax,esi
    call iprintLF
    call quit
```

[Записано 23 строки]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^N Замена ^U Вставить ^J Выводить ^_ К строке

Рис. 3.12: Заполняем файл

Создаем исполняемый файл и запускаем его



```
iazubov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
iazubov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
iazubov@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рис. 3.13: Создаем объектный файл и проверяем работу программы

Изменим программу, чтоб она выводила произведение

```
GNU nano 7.2 /home/iazubov/work/arch-pc/lab08/lab8-3.asm
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx,1
    mov esi,1
next:
    cmp ecx,0h
    jz _end
    pop eax
    call atoi
    mul esi
    mov esi,eax
    loop next
_end:
    mov eax,msg
    call sprint
    mov eax,esi
    call iprintLF
    call quit
```

[Прочитано 24 строки]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^D Выводить ^/_ К строке

Рис. 3.14: Редактируем файл

Создаем исполняемый файл и запускаем его

```
iazubov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
iazubov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
iazubov@fedora:~/work/arch-pc/lab08$ ./lab8-3 10 5 2
Результат: 100
iazubov@fedora:~/work/arch-pc/lab08$
```

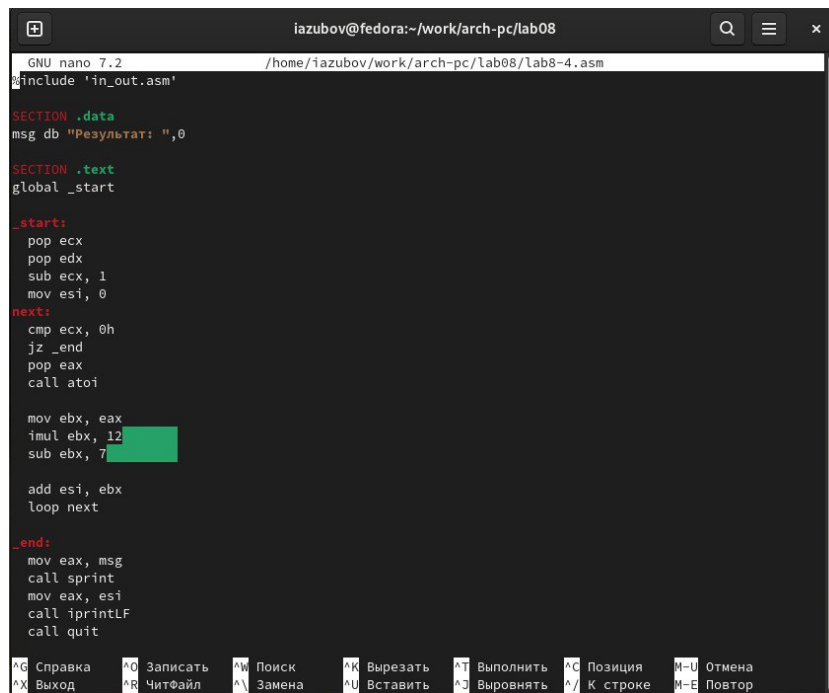
Рис. 3.15: Создаем объектный файл и проверяем работу программы

4 Самостоятельная работа

Вариант 13

- 1) . Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots$, т.е. программа должна выводить значения $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

Создаем новый файл,открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения $12x-7$



```
GNU nano 7.2 /home/iazubov/work/arch-pc/lab08/lab8-4.asm
#include "in_out.asm"

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start

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

    mov ebx, eax
    imul ebx, 12
    sub ebx, 7

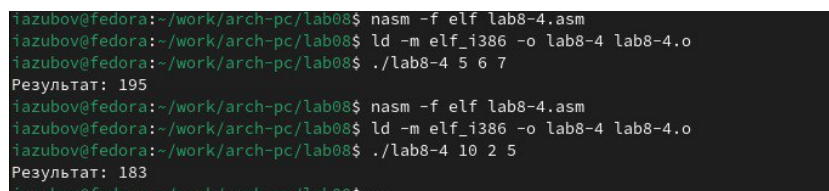
    add esi, ebx
    loop next

_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF
    call quit
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция M-U Отмена
^X Выход ^R ЧитФайл ^M Замена ^U Вставить ^D Выровнять ^/_ К строке M-E Повтор

Рис. 4.1: Пишем программу

Транслируем файл и смотрим на работу программы



```
iazubov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
iazubov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
iazubov@fedora:~/work/arch-pc/lab08$ ./lab8-4 5 6 7
Результат: 195
iazubov@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
iazubov@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
iazubov@fedora:~/work/arch-pc/lab08$ ./lab8-4 10 2 5
Результат: 183
iazubov@fedora:~/work/arch-pc/lab08$ mc
```

Рис. 4.2: Смотрим, что все получилось

5 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки