

Отчёт по лабораторной работе №8

Иванова Анастасия Сергеевна

Содержание

1	Цель работы	5
2	Порядок выполнения лабораторной работы	6
3	Задание для самостоятельной работы	20
4	Вывод	23

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Ввод программы	7
2.3	Проверка работы программы	8
2.4	Бесконечный вывод	9
2.5	Изменение программы	11
2.6	Проверка работы программы	12
2.7	Изменение программы	13
2.8	Проверка работы программы	14
2.9	Создание файла	14
2.10	Ввод программы	15
2.11	Проверка работы программы	15
2.12	Создание файла	16
2.13	Ввод программы	16
2.14	Проверка работы программы	17
2.15	Изменение программы	18
2.16	Проверка работы программы	19
3.1	Создание файла	20
3.2	Ввод программы	21
3.3	Проверка работы программы	22


Список таблиц

1 Цель работы

Приобрети навыки написания программ с использованием циклов и обработкой аргументов командной строки.

2 Порядок выполнения лабораторной работы

1. Создадим каталог для программ лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm рис. 2.1:



```
asivanova@anastasia-750XGK:~$ mkdir ~/work/arch-pc/lab08
asivanova@anastasia-750XGK:~$ cd ~/work/arch-pc/lab08
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ touch lab8-1.asm
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$
```

Рисунок 2.1: Создание каталога и файла

Введем в файл lab8-1.asm текст программы, создадим исполняемый файл и проверим его работу рис. 2.2:

```

GNU nano 6.2                               /home/anasta
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0hSECTION .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

```

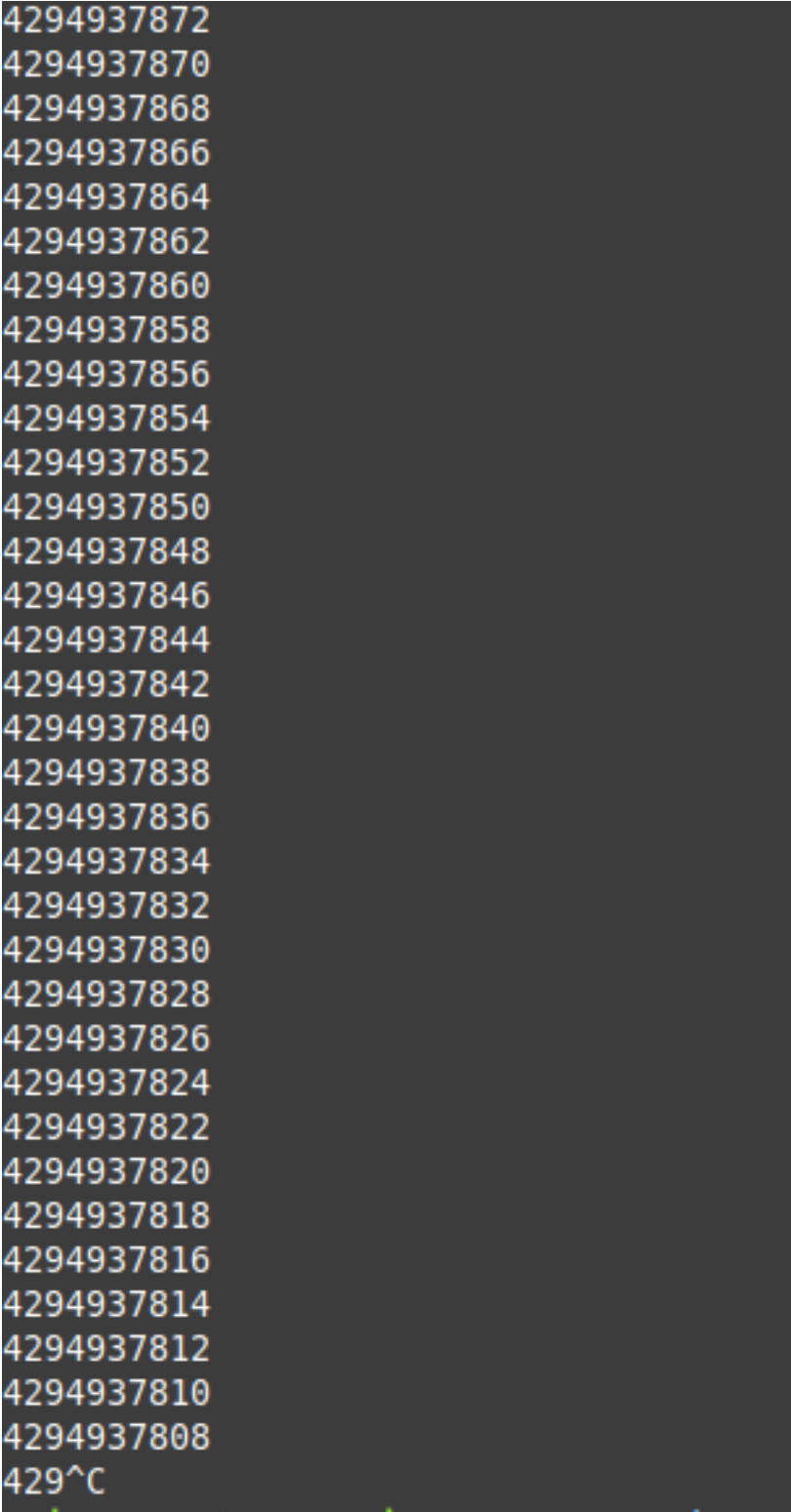
Рисунок 2.2: Ввод программы

рис. 2.3:

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 2
2
1
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
```

Рисунок 2.3: Проверка работы программы

рис. 2.4:



4294937872
4294937870
4294937868
4294937866
4294937864
4294937862
4294937860
4294937858
4294937856
4294937854
4294937852
4294937850
4294937848
4294937846
4294937844
4294937842
4294937840
4294937838
4294937836
4294937834
4294937832
4294937830
4294937828
4294937826
4294937824
4294937822
4294937820
4294937818
4294937816
4294937814
4294937812
4294937810
4294937808
429^C

Рисунок 2.4: Бесконечный вывод

Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы, при вводе некоторых значений выводит бесконечный цикл.

Изменим текст программы рис. 2.5:

```
GNU nano 6.2 /home
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resd 1
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 iprintLF
loop label
```

Рисунок 2.5: Изменение программы

Создадим исполняемый файл и проверим его работу рис. 2.6:

```

asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 3
2
1
0

```

Рисунок 2.6: Проверка работы программы

Какие значения принимает регистр `ecx` в цикле?: $3 \rightarrow 3 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 2 \rightarrow 1 \rightarrow 2 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0$ (цикл выполняется ровно 3 раза) Соответствует ли число проходов цикла значению $\boxed{3}$ введенному с клавиатуры?: да

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внесем изменения в текст программы добавив команды `push` и `pop` рис. 2.7:

```
GNU nano 6.2 /home/ana:
#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:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label
```

Рисунок 2.7: Изменение программы

Создадим исполняемый файл и проверим его работу рис. 2.8:

```

asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ nasm -f elf lab8-1.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-1 lab8-1.o
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 4
3
2
1
0
Segmentation fault (core dumped)

```

Рисунок 2.8: Проверка работы программы

2. При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрим программу, которая выводит на экран аргументы командной строки.

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы рис. 2.9:

```

asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ touch lab8-2.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ gedit lab8-2.asm

```

Рисунок 2.9: Создание файла

рис. 2.10:

```

GNU nano 6.2 /home/anastas:
%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 sprintf
loop next

_end:
call quit

```

Рисунок 2.10: Ввод программы

Создадим исполняемый файл и запустим его, указав аргументы: user@dk4n31:~\$./lab8-2 аргумент1 аргумент 2 „аргумент 3“ рис. 2.11:

```

asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2

```

Рисунок 2.11: Проверка работы программы

Программа обработала 4 аргумента.

3. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы.

Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 и введем в него текст программы рис. 2.12:

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ touch lab8-3.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ gedit lab8-3.asm
```

Рисунок 2.12: Создание файла

рис. 2.13:

```
GNU nano 6.2
#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
add esi,eax
loop next
_end:
mov eax, msg
call sprint
```

Рисунок 2.13: Ввод программы

Создадим исполняемый файл и запустим его, указав аргументы рис. 2.14:

```
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
asivanova@anastasia-750XGK:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рисунок 2.14: Проверка работы программы

Изменим текст программы для вычисления произведения аргументов командной строки рис. 2.15:

```
GNU nano 6.2 /home/ar
%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
imul esi,eax
loop next
_end:
mov eax, msg
call sprint
```

Рисунок 2.15: Изменение программы

Создадим исполняемый файл и запустим его, указав аргументы рис. 2.16:

```
asivanoval@anastasia-750XGK:~/work/arch-pc/lab08$ gedit lab8-3.asm
asivanoval@anastasia-750XGK:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
asivanoval@anastasia-750XGK:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 54600
```

Рисунок 2.16: Проверка работы программы

3 Задание для самостоятельной работы

1. Создадим файл и введем в него программу рис. 3.1:

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ touch zadanie.asm  
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ gedit zadanie.asm
```

Рисунок 3.1: Создание файла

рис. 3.2:

```

GNU nano 6.2
%include 'in_out.asm'
SECTION .data
func_msg db "Функция: f(x)=30x-11",0
result_msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0
jz _end

pop eax
call atoi

mov ebx, 30
mul ebx
sub eax, 11

add esi, eax

loop next

_end:
mov eax, func_msg
call sprintfLF

mov eax, result_msg
call sprintf
mov eax, esi
call iprintLF
call quit

```

Рисунок 3.2: Ввод программы

2. Создадим исполняемый файл и проверим работу программы рис. 3.3:

```
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ nasm -f elf zadanie.asm
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ld -m elf_i386 -o zadanie zadanie.o
asivanova1@anastasia-750XGK:~/work/arch-pc/lab08$ ./zadanie 1 2 3 4
Функция:  $f(x)=30x-11$ 
Результат: 256
```

Рисунок 3.3: Проверка работы программы

4 Вывод

Мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.