

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Бережной Иван Александрович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Основы реализации циклов в NASM	7
3.2	Обработка аргументов командой строки	10
3.3	Задание для самостоятельной работы	14
4	Выводы	17
	Список литературы	18

Список иллюстраций

3.1	Создание файла lab8-1.asm	7
3.2	Копирование кода в lab8-1.asm	8
3.3	Проверка работы lab8-1.exe	8
3.4	Первое изменение lab8-1.asm	9
3.5	Вторая проверка работы lab8-1.exe	9
3.6	Второе изменение lab8-1.asm	10
3.7	Проверка работы исправленной программы	10
3.8	Создание файла lab8-2.asm	11
3.9	Запуск lab8-2.exe	11
3.10	Создание файла lab8-3.asm	12
3.11	Запуск lab8-3.exe	12
3.12	Изменение файла lab8-3.asm	13
3.13	Запуск изменённого файла lab8-3.asm	13
3.14	Написание программы	14
3.15	Проверка работы программы	15

Список таблиц

1 Цель работы

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

2 Задание

1. Основы реализации циклов в NASM
2. Обработка аргументов командой строки
3. Задание для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Основы реализации циклов в NASM

Создадим каталог, в котором будем работать в дальнейшем, перейдём в него и создадим файл `lab8-1.asm` (рис. 3.1). Теперь скопируем в него предложенный листинг (рис. 3.2), создадим исполняемый файл и проверим его работу (рис. 3.3).

```
[iaberezhnoy@fedora ~]$ cd ~/work/arch-pc/lab08  
[iaberezhnoy@fedora lab08]$ touch lab8-1.asm  
[iaberezhnoy@fedora lab08]$
```

Рис. 3.1: Создание файла `lab8-1.asm`

```
mc [iaberezhnuy@fedora]:~/work/arch-pc/lab08
lab8-1.asm      [----]  0 L: [ 1+28  29/ 29] *(637 / 637b) <EOF>
#include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call sread
; ----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, 'ecx=N'
label:
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения 'N'
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
; переход на 'label'
call quit
```

Рис. 3.2: Копирование кода в lab8-1.asm

```
[iaberezhnuy@fedora lab08]$ nasm -f elf lab8-1.asm
[iaberezhnuy@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[iaberezhnuy@fedora lab08]$ ./lab8-1
Введите N: 5
5
4
3
2
1
[iaberezhnuy@fedora lab08]$
```

Рис. 3.3: Проверка работы lab8-1.exe

Немного изменим программу, добавив инструкцию `sub` в блок кода `label` (рис. 3.4). Проверим работу исполняемого файла (рис. 3.5) - число итераций цикла в два раза (или же почти в два раза) меньше введённого значения, так как счётчик цикла уменьшается на один с каждым проходом этого цикла.

```
label:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
loop label ; `ecx=ecx-1` и если `ecx` не '0'
```

Рис. 3.4: Первое изменение lab8-1.asm

```
[iaberezhnoy@fedora lab08]$ nasm -f elf lab8-1.asm
[iaberezhnoy@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[iaberezhnoy@fedora lab08]$ ./lab8-1
Введите N: 10
9
7
5
3
1
[iaberezhnoy@fedora lab08]$
```

Рис. 3.5: Вторая проверка работы lab8-1.exe

Далее, чтобы сохранить значения счётчика цикла, внесём изменения в код, а именно добавим команду `push ecx` в начало цикла и команду `pop`

ecx в его конец (рис. 3.6). Проверим работу программы (рис. 3.7) - всё работает корректно.

3.2 Обработка аргументов командой строки

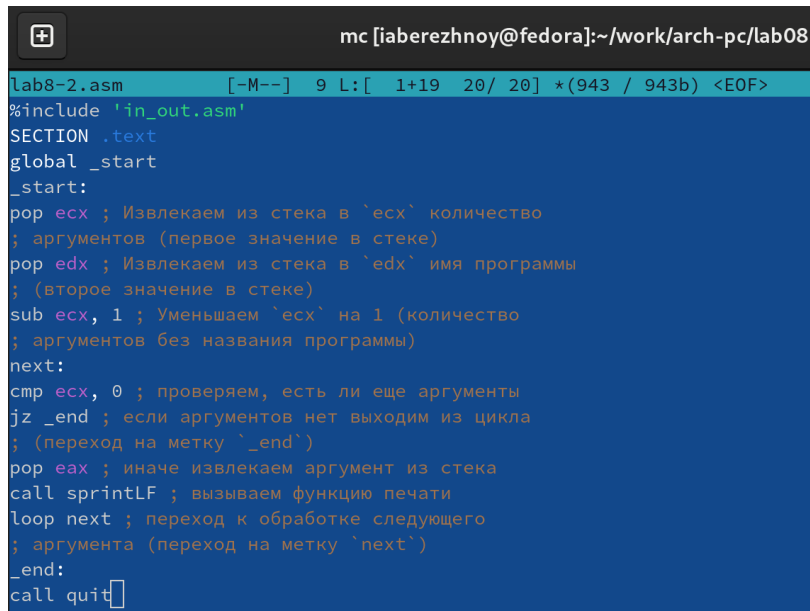
```
label:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintLF ; Вывод значения `N`
pop ecx
loop label ; `ecx=ecx-1` и если `ecx` не '0'
```

Рис. 3.6: Второе изменение lab8-1.asm

```
[iaberezhnoy@fedora lab08]$ nasm -f elf lab8-1.asm
[iaberezhnoy@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[iaberezhnoy@fedora lab08]$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
[iaberezhnoy@fedora lab08]$
```

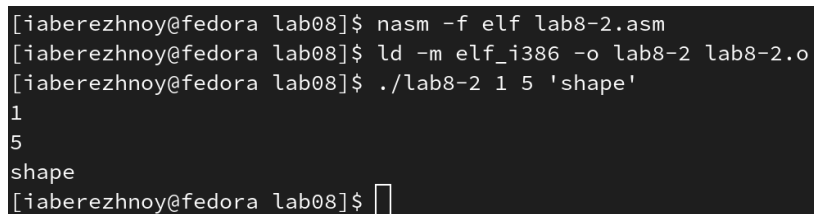
Рис. 3.7: Проверка работы исправленной программы

Создадим файл lab8-2.asm и скопируем в него второй листинг (рис. 3.8). Посмотрим, что делает программа (рис. 3.9). Программой было обработано 3 аргумента.



```
mc [iaberezhnuy@fedora]::~/work/arch-pc/lab08
lab8-2.asm [-M--] 9 L:[ 1+19 20/ 20] *(943 / 943b) <EOF>
#include 'in_out.asm'
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
next:
cmp ecx, 0 ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем аргумент из стека
call sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку `next`)
_end:
call quit
```

Рис. 3.8: Создание файла lab8-2.asm

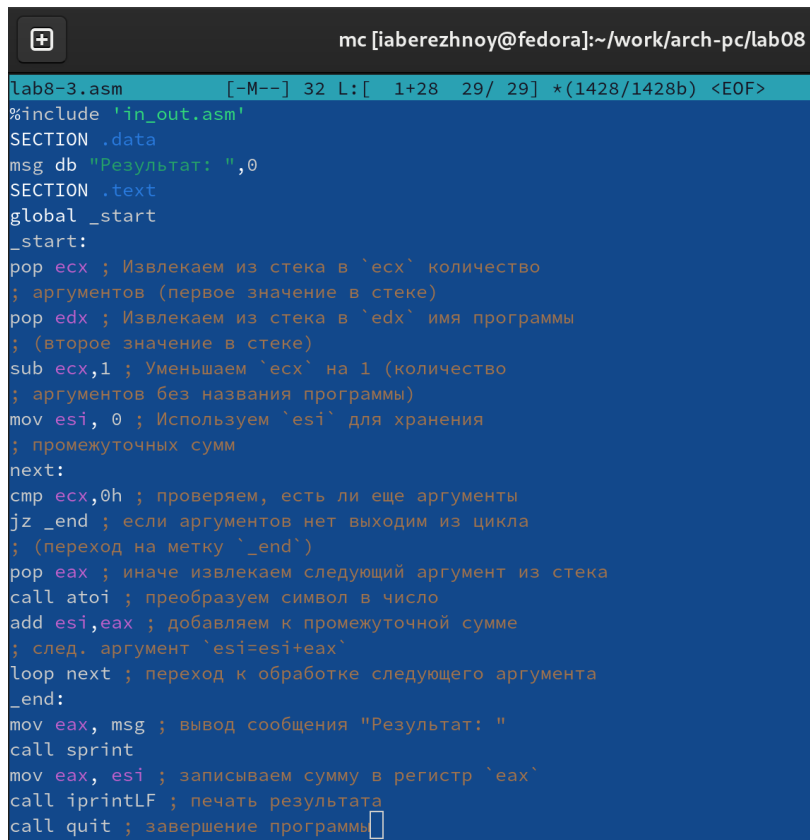


```
[iaberezhnuy@fedora lab08]$ nasm -f elf lab8-2.asm
[iaberezhnuy@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[iaberezhnuy@fedora lab08]$ ./lab8-2 1 5 'shape'
1
5
shape
[iaberezhnuy@fedora lab08]$
```

Рис. 3.9: Запуск lab8-2.exe

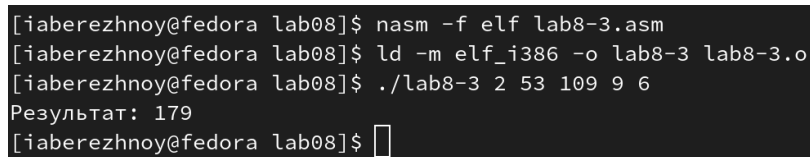
Создадим файл lab8-3.asm и скопируем в него третий листинг (рис.

3.10). Запустим программу (рис. 3.11).



```
mc [iaberezhnoy@fedora]:~/work/arch-pc/lab08
lab8-3.asm  [-M--] 32 L:[ 1+28 29/ 29] *(1428/1428b) <EOF>
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 0 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
add esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=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 ; завершение программы
```

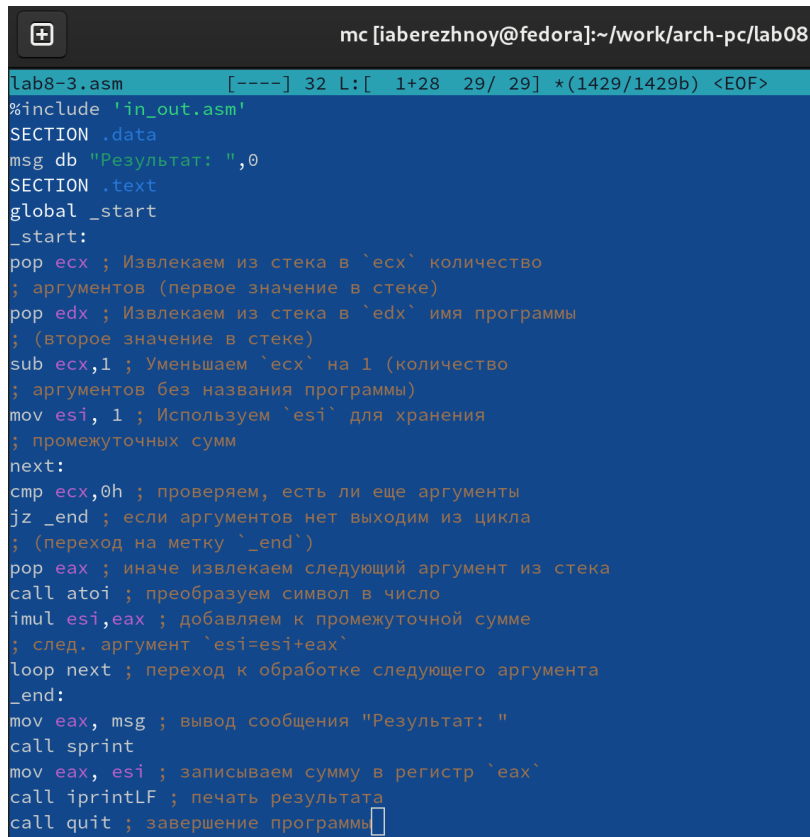
Рис. 3.10: Создание файла lab8-3.asm



```
[iaberezhnoy@fedora lab08]$ nasm -f elf lab8-3.asm
[iaberezhnoy@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[iaberezhnoy@fedora lab08]$ ./lab8-3 2 53 109 9 6
Результат: 179
[iaberezhnoy@fedora lab08]$
```

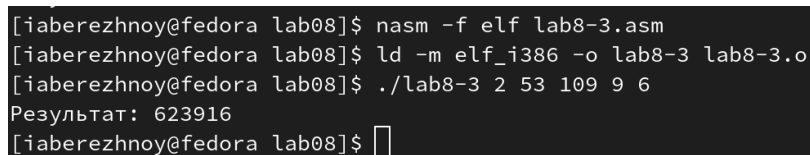
Рис. 3.11: Запуск lab8-3.exe

Изменим текст последней программы так, чтобы в результате мы получили произведение аргументов (рис. 3.12). Запустим программу и проверим её работу (рис. 3.13).



```
mc [iaberezhnuy@fedora]:~/work/arch-pc/lab08
lab8-3.asm      [----] 32 L: [ 1+28 29/ 29] *(1429/1429b) <EOF>
#include 'in_out.asm'
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество
; аргументов (первое значение в стеке)
pop edx ; Извлекаем из стека в `edx` имя программы
; (второе значение в стеке)
sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
; аргументов без названия программы)
mov esi, 1 ; Используем `esi` для хранения
; промежуточных сумм
next:
cmp ecx,0h ; проверяем, есть ли еще аргументы
jz _end ; если аргументов нет выходим из цикла
; (переход на метку `_end`)
pop eax ; иначе извлекаем следующий аргумент из стека
call atoi ; преобразуем символ в число
imul esi,eax ; добавляем к промежуточной сумме
; след. аргумент `esi=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 ; завершение программы
```

Рис. 3.12: Изменение файла lab8-3.asm

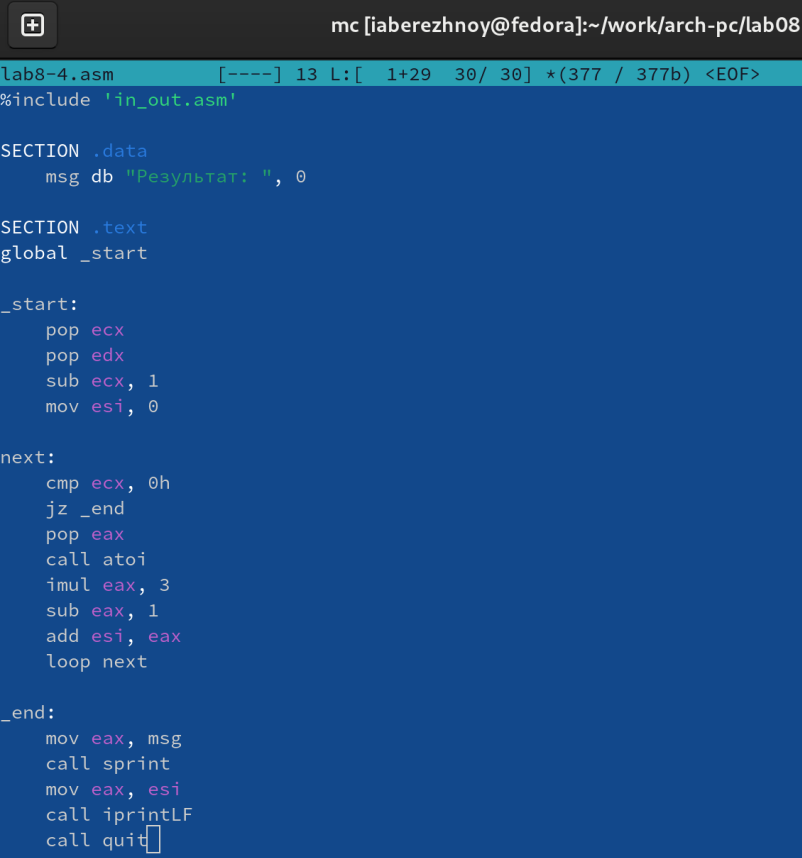


```
[iaberezhnuy@fedora lab08]$ nasm -f elf lab8-3.asm
[iaberezhnuy@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[iaberezhnuy@fedora lab08]$ ./lab8-3 2 53 109 9 6
Результат: 623916
[iaberezhnuy@fedora lab08]$
```

Рис. 3.13: Запуск изменённого файла lab8-3.asm

3.3 Задание для самостоятельной работы

Напишем программу на языке ассемблера NASM, которая будет складывать результаты вычисления значений функции при введенных значениях аргумента. Функцию берём из таблицы согласно полученному в лабораторной работе №6 варианту - 2 (рис. 3.14). Создадим исполняемый файл и проверим его работу на разных наборах значений аргументов (рис. 3.15).



```
mc [iaberezhnoy@fedora]:~/work/arch-pc/lab08
lab8-4.asm  [----] 13 L: [ 1+29 30/ 30] *(377 / 377b) <EOF>
#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
    imul eax, 3
    sub eax, 1
    add esi, eax
    loop next

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

Рис. 3.14: Написание программы

```

[iaberezhnoy@fedora lab08]$ nasm -f elf lab8-4.asm
[iaberezhnoy@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[iaberezhnoy@fedora lab08]$ ./lab8-4 1 2 3 4
Результат: 26
[iaberezhnoy@fedora lab08]$ ./lab8-4 2 4 6 7
Результат: 53

```

Рис. 3.15: Проверка работы программы

Листинг 8.1. Программа вычисления суммы значений функции при заданных аргументах

```

#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
imul eax, 3
sub eax, 1
add esi, eax
loop next
```

_end:

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


4 Выводы

В ходе выполнения лабораторной работы мы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.

Список литературы

::: Архитектура ЭВМ