

# **Отчёта по лабораторной работе №8**

**Дисциплина: Архитектура компьютера**

Кижваткина Анна Юрьевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>13</b>

# Список иллюстраций

3.1	Создание каталога . . . . .	7
3.2	Перемещение в каталог . . . . .	7
3.3	Создание файла . . . . .	7
3.4	Ввод программы из листинга . . . . .	7
3.5	Создание исполняемого файла и запуск . . . . .	8
3.6	Изменение программы . . . . .	8
3.7	Создание исполняемого файла и запуск . . . . .	8
3.8	Изменение программы . . . . .	9
3.9	Создание исполняемого файла и запуск . . . . .	9
3.10	Создание файла . . . . .	9
3.11	Ввод программы из листинга . . . . .	10
3.12	Создание исполняемого файла и запуск . . . . .	10
3.13	Создание файла . . . . .	10
3.14	Ввод программы из листинга . . . . .	11
3.15	Создание исполняемого файла и запуск . . . . .	11
3.16	Изменение программы . . . . .	11
3.17	Создание исполняемого файла и запуск . . . . .	11
3.18	Создание файла . . . . .	12
3.19	Написание программы . . . . .	12
3.20	Создание исполняемого файла и запуск . . . . .	12

## **Список таблиц**

# 1 Цель работы

Целью данной лабораторной работы является приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

## **2 Задание**

1. Реализация циклов в NASM.
2. Обработка аргументов командной строки.
3. Выполнение лабораторной работы.

### 3 Выполнение лабораторной работы

Создаем каталог для программ лабораторной работы №8. (рис. 3.1).

```
aykizhvatkina@dk2n22 ~ $ mkdir ~/work/arch-pc/lab08
```

Рис. 3.1: Создание каталога

Переходим в созданный каталог. (рис. 3.2).

```
aykizhvatkina@dk2n22 ~ $ cd ~/work/arch-pc/lab08
```

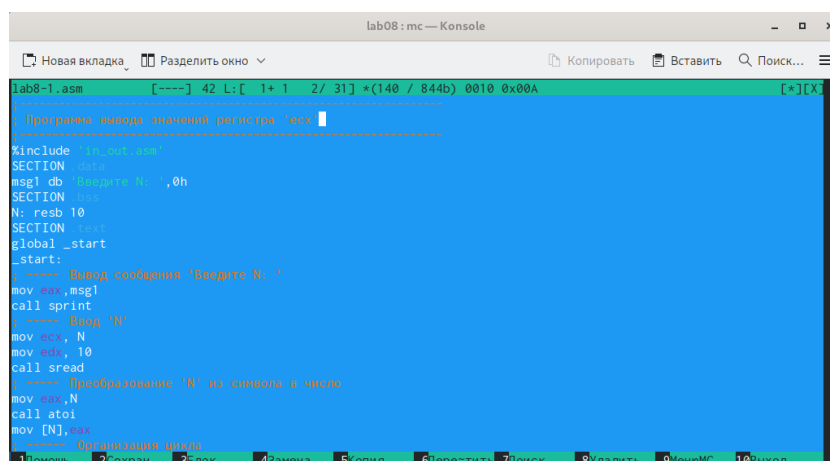
Рис. 3.2: Перемещение в каталог

Создаем файл lab8-1.asm. Проверяем наличие. (рис. 3.3).

```
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ touch lab8-1.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ls
lab8-1.asm
```

Рис. 3.3: Создание файла

Вводим в файл программу из листинга 8.1. (рис. 3.4).



```
lab8-1.asm [---] 42 L: [ 1+ 1 2/ 31] *(140 / 844b) 0010 0x00A [*][X]
; Программа вывода значений регистра 'ecx'
;-----
%include "lab08-1.asm"
SECTION ".text"
msg1 db "Введите N: ",0h
SECTION ".bss"
N: resb 10
SECTION ".data"
global _start
_start:
;----- Вывод сообщения "Введите N: "
mov eax,msg1
call sprintf
;----- Ввод 'N'
mov ecx,N
mov edx,10
call read
;----- Преобразование 'N' из символа в число
mov eax,N
call atoi
mov [N],eax
;----- Организация цикла
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пере-тить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 3.4: Ввод программы из листинга

Создаем исполняемый файл и запускаем его. (рис. 3.5).

```
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
```

Рис. 3.5: Создание исполняемого файла и запуск

Меняем label в программе согласно тексту. (рис. 3.6).

```
lab8-1.asm [----] 9 L: [ 10+22 32/ 32] *(792 / 792b) <EOF> [*][X]
global _start
_start:
; ----- Вывод сообщения 'Введите N: '
mov eax, msg1
call sprintf
; ----- Ввод 'N'
mov ecx, N
mov edx, 10
call read
; ----- Преобразование 'N' из символа в число
mov eax, N
call atoi
mov [N], eax
; ----- Организация цикла
mov ecx, [N] ; Счетчик цикла, 'ecx=N'
label:
sub ecx, 1 ; 'ecx=ecx-1'
mov [N], ecx
mov eax, [N]
call iprintLF
loop label
; переход на 'label'
call quit
```

Рис. 3.6: Изменение программы

Создаем исполняемый файл и запускаем его. (рис. 3.7).

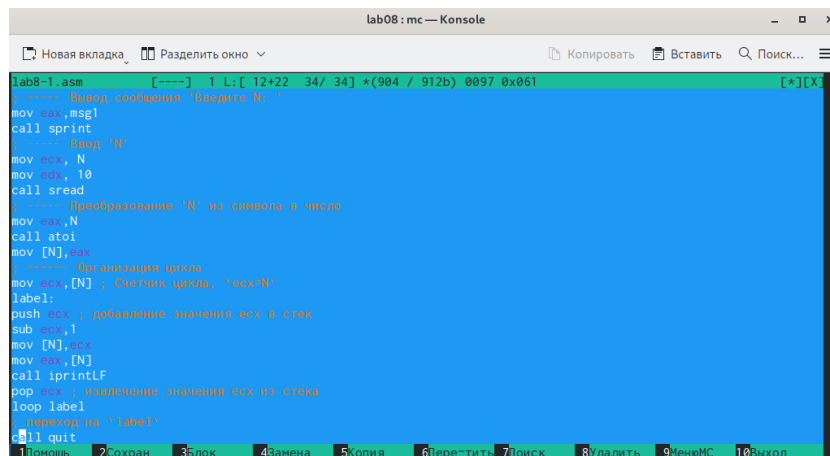
```
4294783010
4294783008
4294783006
4294783004
4294783002
4294783000
4294782998
```

Рис. 3.7: Создание исполняемого файла и запуск

Цикл закольцевался и стал бесконечным.

Меняем label в программе согласно тексту. (рис. 3.8).



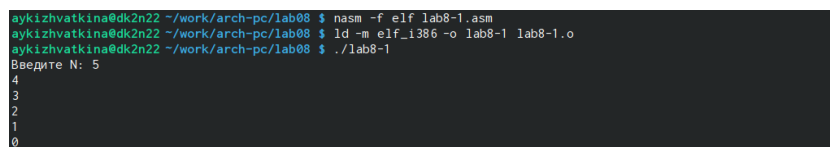


```
lab8-1.asm  [----]  1 L: [ 12+22  34/ 34] *(904 / 912b) 0097 0x061  [*][X]
; ----- Вывод сообщения "Введите N: "
mov eax,msg1
call sprintf
; ----- Ввод N
mov ecx, N
mov edx, 10
call read
; ----- Преобразование 'N' из символа в число
mov ecx,N
call atoi
mov [N],ecx
; ----- Организация цикла
mov ecx,[N] ; Столбик цикла, 'ecx=N'
label:
push ecx ; добавление значения ecx в стек
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintfLF
pop ecx ; извлечение значения ecx из стека
loop label
; переход на :label
call quit
```

Рис. 3.8: Изменение программы

В данной ситуации циклы нумеруются с 0. Число проходов совпадает с введенным значением N.

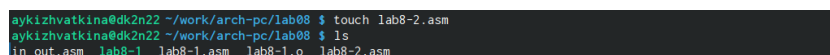
Создаем исполняемый файл и запускаем его. (рис. 3.9).



```
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
4
3
2
1
0
```

Рис. 3.9: Создание исполняемого файла и запуск

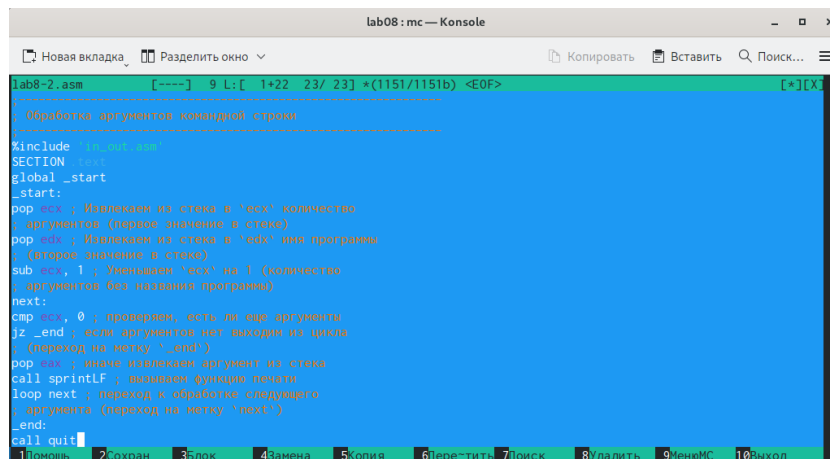
Создаем файл lab8-1.asm. Проверяем наличие. (рис. 3.10).



```
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ touch lab8-2.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o  lab8-2.asm
```

Рис. 3.10: Создание файла

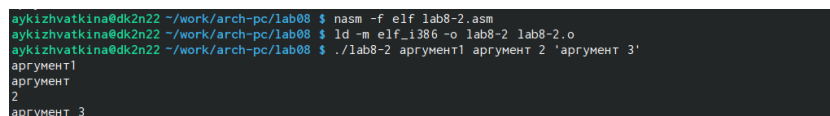
Вводим в файл программу из листинга 8.2. (рис. 3.11).



```
lab8-2.asm  [----]  9  L: [ 1+22 23/ 23] *(1151/1151b) <E0F>  [X][X]
Обработка аргументов командной строки
%include "lab08-1.asm"
SECTION .text
global _start
_start:
    pop %eax ; Извлекаем из стека в '%eax' количество
                аргументов (первое значение в стеке)
    pop %edx ; Извлекаем из стека в '%edx' имя программы
                (второе значение в стеке)
    sub %eax, 1 ; Значение '%eax' на 1 (количество
                аргументов без названия программы)
    next:
        cmp %eax, 0 ; проверяем, есть ли еще аргументы
        jz _end ; если аргументов нет, выходим из цикла
                (переход на метку '_end')
        pop %eax ; снова извлекаем аргумент из стека
        call printf ; вызываем функцию печати
        loop next ; переход к обработке следующего
                аргумента (переход на метку 'next')
    _end:
    call quit
```

Рис. 3.11: Ввод программы из листинга

Создаем исполняемый файл и запускаем его. (рис. 3.12).

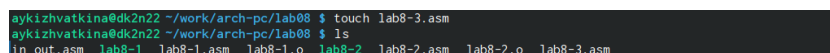


```
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент 2
аргумент 3
```

Рис. 3.12: Создание исполняемого файла и запуск

Программа выводит все 3 аргумента которые ввели, но в разной вариации.

Создаем файл lab8-1.asm. Проверяем наличие. (рис. 3.13).



```
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ touch lab8-3.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o  lab8-2  lab8-2.asm  lab8-2.o  lab8-3.asm
```

Рис. 3.13: Создание файла

Вводим в файл программу из листинга 8.3. (рис. 3.14).

```

lab8-3.asm [----] 7 L: [ 1+ 5 6/ 29] *(103 /1428b) 0010 0x00A [*][X]
%include "lab8-3.inc"
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 'exitcode')
    loop next ; переход к обработке следующего аргумента
_end:
    ; выводим результат
    mov eax, esi
    int 0x40
    ; выходим
    mov eax, 0
    int 0x80

```

Рис. 3.14: Ввод программы из листинга

Создаем исполняемый файл и запускаем его. (рис. 3.15).

```

aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47

```

Рис. 3.15: Создание исполняемого файла и запуск

Изменяем программу, чтобы вместо суммы было произведение. (рис. 3.16).

```

lab8-3.asm [----] 12 L: [ 1+ 4 5/ 33] *(94 /1412b) 0116 0x074 [*][X]
%include "lab8-3.inc"
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
    mov ebx, eax
    mov eax, esi
    imul ebx,eax
    loop next
_end:
    ; выводим результат
    mov eax, esi
    int 0x40
    ; выходим
    mov eax, 0
    int 0x80

```

Рис. 3.16: Изменение программы

Создаем исполняемый файл и запускаем его. (рис. 3.17).

```

aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 54600

```

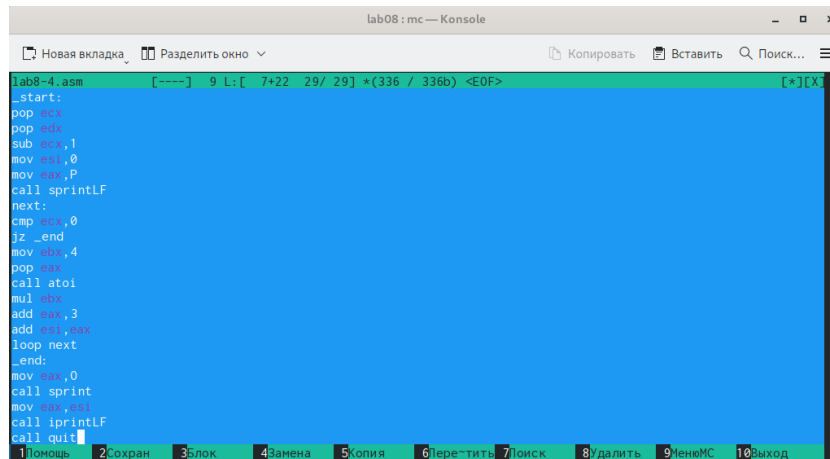
Рис. 3.17: Создание исполняемого файла и запуск

Создаем файл lab8-1.asm. Проверяем наличие. (рис. 3.18).

```
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ touch lab8-4.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ls
in_out.asm  lab8-1.asm  lab8-2      lab8-2.o    lab8-3.asm  lab8-4.asm
lab8-1      lab8-1.o    lab8-2.asm  lab8-3      lab8-3.o
```

Рис. 3.18: Создание файла

Пишем программу(рис. 3.19).



```
lab8-4.asm  [----]  9 L: [ 7*22 29/ 29] *(336 / 336b) <EOF>  [*][X]
start:
pop %ax
pop %dx
sub %cx,1
mov %ax,0
mov %ax,P
call sprintf
next:
cmp %cx,0
jz _end
mov %bx,4
pop %ax
call atoi
mul %bx
add %ax,3
add %ax,%ax
loop next
_end:
mov %ax,0
call sprintf
mov %ax,%ax
call iprintf
call quit
```

Рис. 3.19: Написание программы

Создаем исполняемый файл и запускаем его. (рис. 3.20).

```
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3
f(x)=4x+3
Результат: 33
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4
f(x)=4x+3
Результат: 52
aykizhvatkina@dk2n22 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4 5
f(x)=4x+3
Результат: 75
```

Рис. 3.20: Создание исполняемого файла и запуск

## **4 Выводы**

С помощью данной лабораторной работы приобрели навыки написания программ с использованием циклов и обработкой аргументов командной строки.