

# **Отчёт по лабораторной работе №8**

**дисциплина: Архитектура компьютера**

Гайдук Софья Сергеевна

# **Содержание**

<b>1 Цель работы</b>	<b>5</b>
<b>2 Выполнение лабораторной работы</b>	<b>6</b>
<b>3 Задания для самостоятельной работы</b>	<b>12</b>
<b>4 Выводы</b>	<b>15</b>

# Список иллюстраций

2.1	image1	6
2.2	image2	6
2.3	image3	7
2.4	image4	7
2.5	image5	8
2.6	image6	8
2.7	image7	9
2.8	image8	9
2.9	image9	9
2.10	image10	10
2.11	image11	10
2.12	image12	10
2.13	image13	11
2.14	image14	11
2.15	image15	11
3.1	image16	12
3.2	image17	13
3.3	image18	13
3.4	image19	14
3.5	image20	14

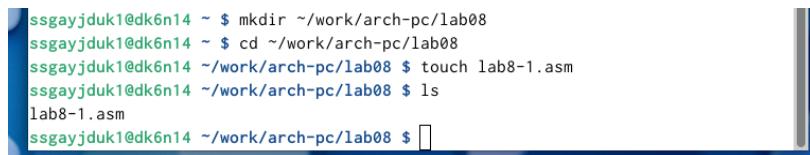
# **Список таблиц**

# **1 Цель работы**

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

## 2 Выполнение лабораторной работы

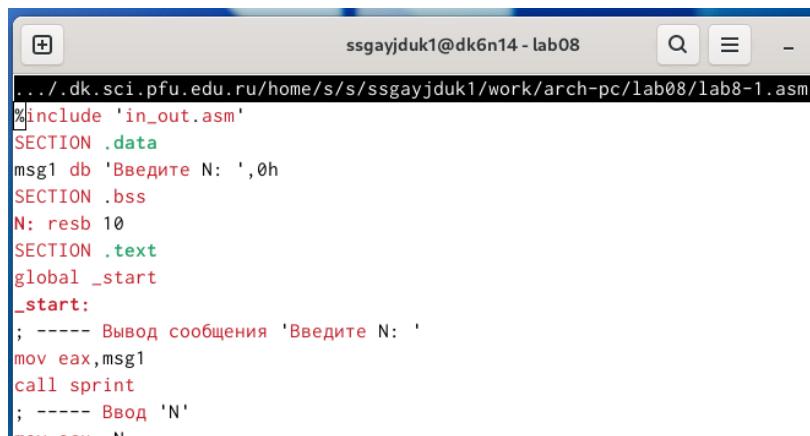
Создадим каталог для программ лабораторной работы № 8, перейдем в него и создадим файл lab8-1.asm. Проверим его наличие (рис. 2.1).



```
ssgayjduk1@dk6n14 ~ $ mkdir ~/work/arch-pc/lab08
ssgayjduk1@dk6n14 ~ $ cd ~/work/arch-pc/lab08
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ touch lab8-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ls
lab8-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

Рисунок 2.1: image1

Введем в файл lab8-1.asm текст программы из листинга 8.1. Создадим исполняемый файл и проверим его работу (рис. 2.2).



```
ssgayjduk1@dk6n14 - lab08
.... /dk.sci.pfu.edu.ru/home/s/s/ssgayjduk1/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:
; ----- Вывод сообщения 'Введите N: '
mov eax,msg1
call sprint
; ----- Ввод 'N'
mov [N], al
```

Рисунок 2.2: image2

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

Рисунок 2.3: image3

Изменим текст программы добавив изменение значение регистра ecx в цикле: (рис. 2.4).

```
; ----- 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
call quit
```

Рисунок 2.4: image4

Создадим исполняемый файл и проверим его работу (рис. 2.5).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
7
5
3
1
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

Рисунок 2.5: image5

Регистр принимает значения есх все значения от 0 до 10 в цикле. Число проходов цикла не соответствует значению N введенному с клавиатуры, так как выполняется дважды вычитание: sub ecx, 1 и loop label.

Внесем изменения в текст программы добавив команды push и pop (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла loop (рис. 2.6).

```
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N'
label:
push ecx ; добавление значения есх в стек
sub ecx,1 ; `ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx ; извлечение значения есх из стека
loop label

call quit
```

Рисунок 2.6: image6

Создайте исполняемый файл и проверьте его работу (рис. 2.7).

```

ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ 

```

Рисунок 2.7: image7

Число проходов цикла, в данном случае, соответствует значению N введенному с клавиатуры.

Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08, проверим его наличие и введем в него текст программы из листинга 8.2 (рис. 2.8).

```

ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ touch lab8-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o  lab8-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ 

```

Рисунок 2.8: image8

```

mc [ssgayjduk1@dk6n14.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab08
.../dk.sci.pfu.edu.ru/home/s/s/ssgayjduk1/work/arch-pc/lab08/lab8-2.asm Изменён
; Обработка аргументов командной строки
;
%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 sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего

```

Справка ^G Справка ^O Записать ^F Поиск ^K Вырезать ^T Выполнить M-U Отмена  
Выход ^X ЧитФайл ^\ Замена ^U Вставить ^C Позиция M-E Повтор

Рисунок 2.9: image9

Создадим исполняемый файл и запустим его, указав аргументы: аргумент1 аргумент 2 „аргумент 3“ (рис. 2.10).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

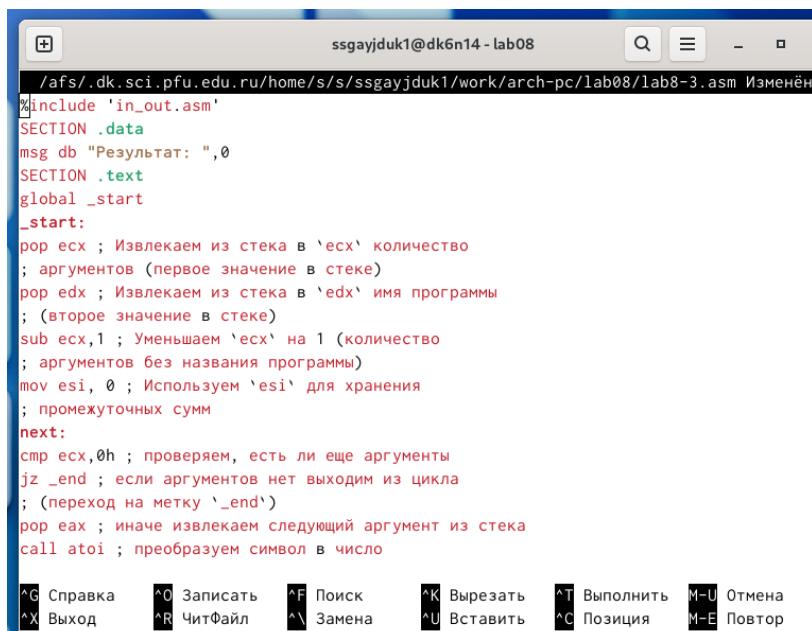
Рисунок 2.10: image10

Программой было обработано 4 аргумента.

Создадим файл lab8-3.asm в каталоге ~/work/archpc/lab08 и введем в него текст программы из листинга 8.3 (рис. 2.11).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ touch lab8-3.asm
ssgayjduk1@dk6n14 ~/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
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

Рисунок 2.11: image11



The screenshot shows a terminal window with the title "ssgayjduk1@dk6n14 - lab08". The command entered was "cat lab8-3.asm". The assembly code listed is:

```
%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
```

Рисунок 2.12: image12

Создадим исполняемый файл и запустим его, указав аргументы 12, 13, 7, 10, 5 (рис. 2.13).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

Рисунок 2.13: image13

Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 2.14).

```
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
mov ebx,ebx
mul ebx ; умножаем

mov esi,eax
loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg ; вывод сообщения "Результат: "
call sprint
```

Справка    Записать    Поиск    Вырезать    Выполнить    Отмена  
Выход    ЧитФайл    Замена    Вставить    Позиция    Повтор

Рисунок 2.14: image14

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-3 12
Результат: 12
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 54600
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

Рисунок 2.15: image15

## 3 Задания для самостоятельной работы

Создадим файл lab8-4.asm в каталоге ~/work/archpc/lab08 (рис. 3.1).

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ touch lab8-4.asm  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

Рисунок 3.1: image16

Напишем программу, которая находит сумму значений функции  $f(x)$  для  $x=x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1)+f(x_2)+\dots+f(x_n)$ . Значения  $x_i$  передаются как аргументы. Вид функции  $f(x)$  номер 6 в соответствии с вариантом, полученным при выполнении лабораторной работы №6. Создадим исполняемый файл и проверим его работу на нескольких наборах  $x=x_1, x_2, \dots, x_n$  (рис. 3.2).

```
/afs/.dk.sci.pfu.edu.ru/home/s/s/ssgayjduk1/work/arch-pc/lab08
%include 'in_out.asm'
SECTION .data
msg1 db "Функция: f(x)=4x-3",0
msg2 db "Результат: ",0
msg3 db "Номер варианта 6",0
SECTION .text
global _start
_start:
pop ecx ; Извлекаем из стека в `ecx` количество аргументов (первое значение в с)
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 значений функций
mov edi,0 ; Сумма функций
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 eax,eax
mov ebx, 4
mul ebx
sub eax,3
add edi,eax

loop next ; переход к обработке следующего аргумента
_end:
mov eax, msg3 ; вывод сообщения "Номер варианта 6"
mov eax, msg1 ; вывод сообщения "Функция: f(x)=4x-3 "
call sprintLF
mov eax, msg2 ; вывод сообщения "Результат: "
call sprint
mov eax, edi ; записываем сумму в регистр `eax`
call iprintLF ; печать результата
call quit ; завершение программы

[ Прочитано 36 строк ]
^G Справка ^O Записать ^F Поиск ^K Вырезать ^T Выполнить M-U Отмена
^X Выход ^R ЧитФайл ^Y Замена ^U Вставить ^C Позиция M-E Повтор
```

Рисунок 3.2: image17

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4
Функция: f(x)=4x-3
Результат: 28
ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $ ./lab8-4 1 1 1 1
Функция: f(x)=4x-3
Результат: 4

ssgayjduk1@dk6n14 ~/work/arch-pc/lab08 $
```

Рисунок 3.3: image18

Отправим файлы на Github (рис. 3.4).

```
08/report $ git add .
warning: in the working copy of 'labs/lab08/report/lab08/in_out.asm', CRLF will
be replaced by LF the next time Git touches it
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab
08/report $ git commit -am 'feat(main): add lab-8'
[master e5978bf] feat(main): add lab-8
 36 files changed, 197 insertions(+), 32 deletions(-)
create mode 100644 labs/lab08/report/arch-pc--lab08--report.docx
create mode 100644 labs/lab08/report/arch-pc--lab08--report.pdf
create mode 100644 labs/lab08/report/image/image1.jpg
```

Рисунок 3.4: image19

```
-----.
create mode 100644 labs/lab08/report/lab08/lab8-4.o
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab
08/report $ git push
Перечисление объектов: 54, готово.
Подсчет объектов: 100% (54/54), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (43/43), готово.
Запись объектов: 100% (44/44), 2.96 МиБ | 3.86 МиБ/с, готово.
Total 44 (delta 11), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (11/11), completed with 5 local objects.
To github.com:SofiaGayduk/study_2025-2026_arh-pc.git
  e84ba42..e5978bf master -> master
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab
08/report $ █
```

Рисунок 3.5: image20

## **4 Выводы**

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