

Отчёт по лабораторной работе №6

дисциплина: Архитектура компьютера

Гайдук Софья Сергеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задания для самостоятельной работы	20
4	Выводы	23

Список иллюстраций

2.1	image1	6
2.2	image2	7
2.3	image3	8
2.4	image4	9
2.5	image5	10
2.6	image6	10
2.7	image7	11
2.8	image8	11
2.9	image9	12
2.10	image10	12
2.11	image11	13
2.12	image12	13
2.13	image13	13
2.14	image14	14
2.15	image15	14
2.16	image16	15
2.17	image17	15
2.18	image18	15
2.19	image19	16
2.20	image20	16
2.21	image21	17
2.22	image22	17
2.23	image23	17
2.24	image24	18
2.25	image25	18
2.26	image26	19
3.1	image27	21
3.2	image28	22
3.3	image29	22

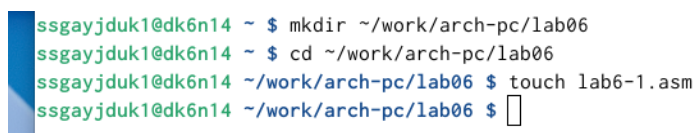
Список таблиц

1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

2 Выполнение лабораторной работы

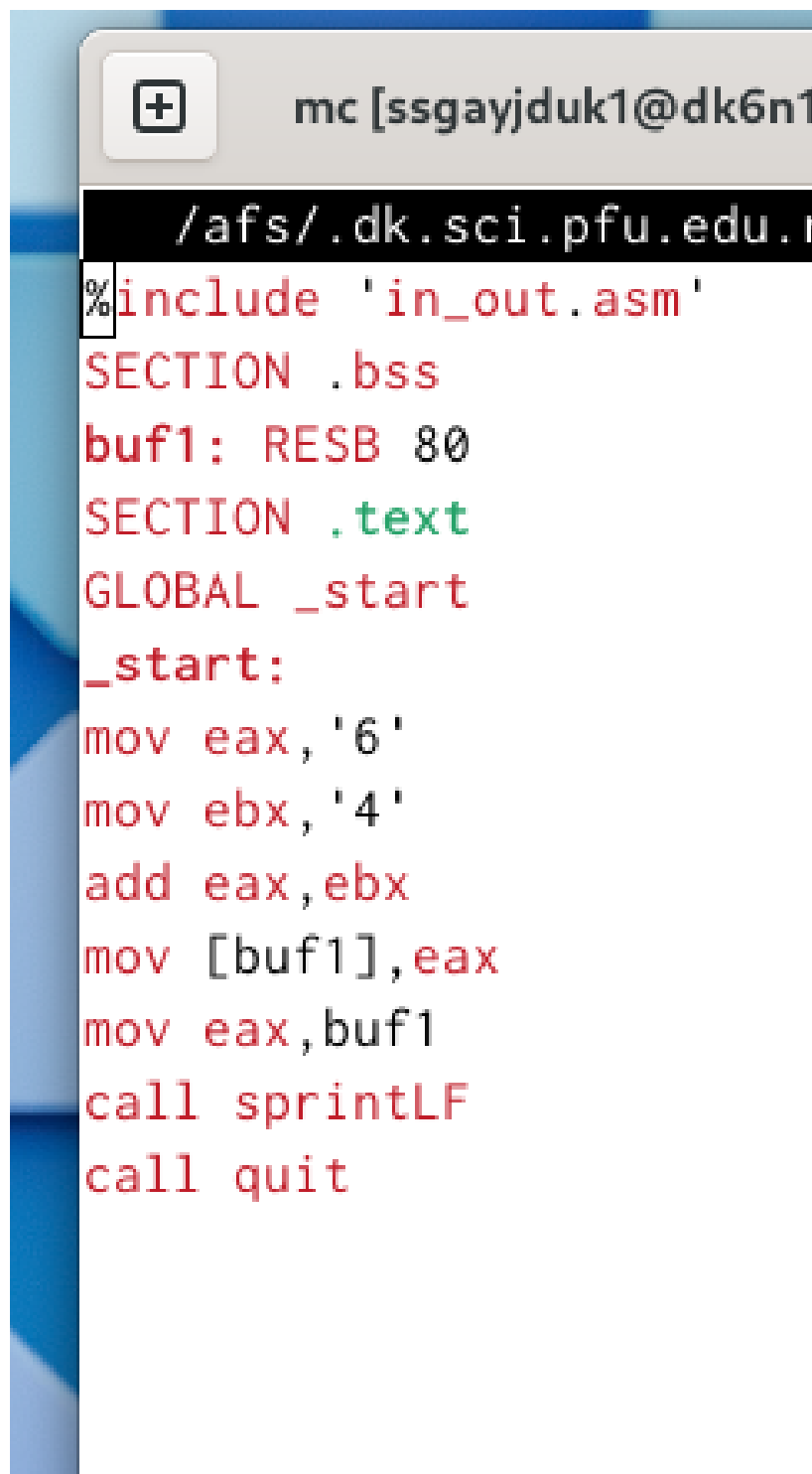
Создадим каталог для программ лабораторной работы № 6, перейдем в него и создадим файл lab6-1.asm (рис. 2.1).



```
ssgayjduk1@dk6n14 ~ $ mkdir ~/work/arch-pc/lab06
ssgayjduk1@dk6n14 ~ $ cd ~/work/arch-pc/lab06
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ touch lab6-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.1: image1

Введем в файл lab6-1.asm текст программы данный в условии (рис. 2.2).

A screenshot of a terminal window. The title bar shows a plus icon in a square and the text 'mc [ssgayjduk1@dk6n1'. The terminal content shows a file path '/afs/.dk.sci.pfu.edu.' followed by assembly code. The code includes an include directive, section declarations for .bss and .text, a global symbol _start, and several instructions: mov eax, '6', mov ebx, '4', add eax, ebx, mov [buf1], eax, mov eax, buf1, call sprintfLF, and call quit. The code is color-coded: include is red, .bss is red, buf1 is red, RESB is red, 80 is black, .text is green, GLOBAL is red, _start is red, _start: is red, mov is red, '6' is black, ebx is red, '4' is black, add is red, [buf1] is red, eax is red, sprintfLF is red, and quit is red.

```
mc [ssgayjduk1@dk6n1
/afs/.dk.sci.pfu.edu.
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
mov [buf1], eax
mov eax, buf1
call sprintfLF
call quit
```

Рисунок 2.2: image2

Создадим исполняемый файл и запустим его (рис. 2.3).

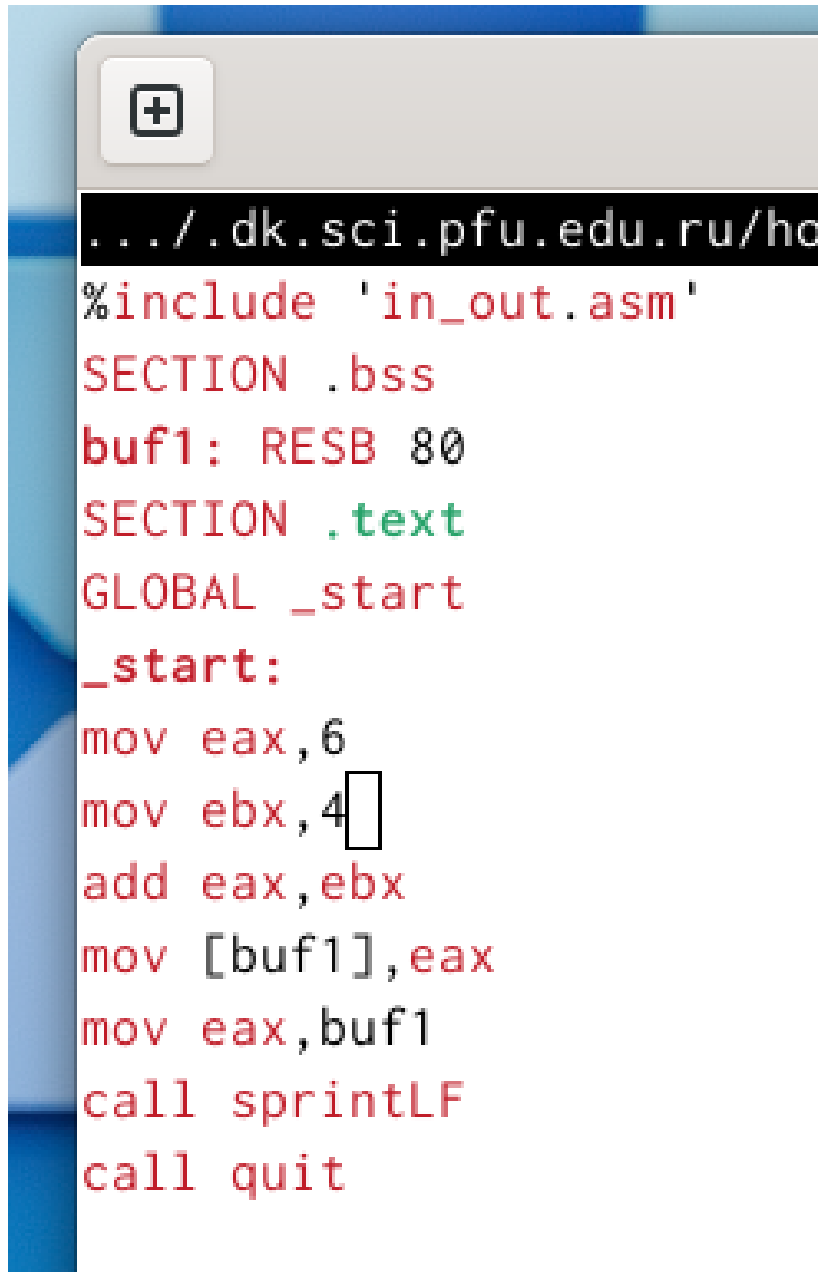
```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./lab6-1
j
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.3: image3

Результатом будет символ j.

Ранее перенесли in_out.asm в каталог ~/work/arch-pc/lab06.

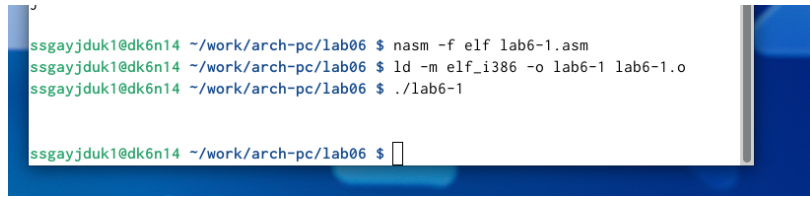
Изменим текст программы и вместо символов, запишем в регистры числа (рис. 2.4).

A screenshot of a code editor window. The window has a title bar with a plus icon in a square. The code is displayed in a monospaced font with syntax highlighting: preprocessor directives and section names are in red, labels and instructions are in black, and string literals are in green. The code defines a buffer in the .bss section and performs assembly instructions in the .text section, including moving values, adding them, and calling printf and _exit.

```
.../.dk.sci.pfu.edu.ru/hc  
%include 'in_out.asm'  
SECTION .bss  
buf1: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
mov [buf1],eax  
mov eax,buf1  
call sprintf  
call quit
```

Рисунок 2.4: image4

Создадим исполняемый файл и запустим его (рис. 2.5).

A terminal window with a blue background and white text. The prompt is 'ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 \$'. The user enters 'nasm -f elf lab6-1.asm', then 'ld -m elf_i386 -o lab6-1 lab6-1.o', then './lab6-1', and finally a blank line with a cursor.

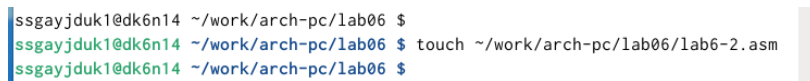
```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./lab6-1

ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.5: image5

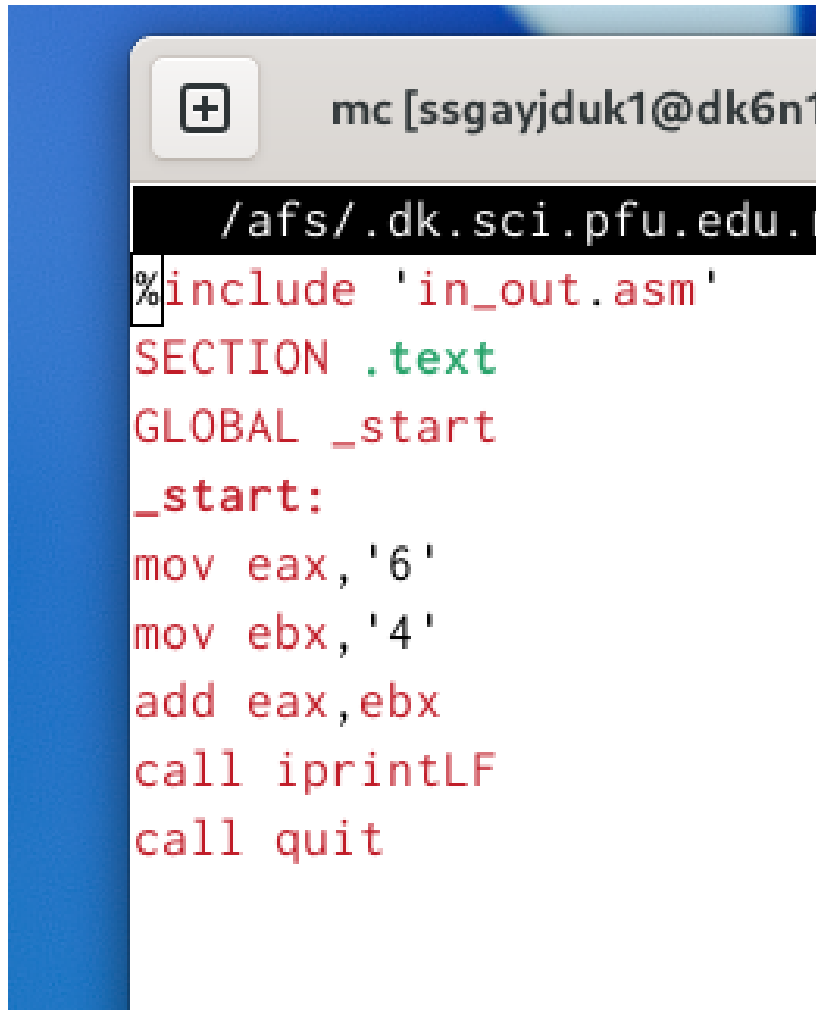
Код 10 соответствует символу пробел (" "). Он также, как и символ j, отображается при выводе на экран.

Создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введем в него текст программы из условия (рис. 2.6)

A terminal window with a blue background and white text. The prompt is 'ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 \$'. The user enters 'touch ~/work/arch-pc/lab06/lab6-2.asm' and then a blank line with a cursor.

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

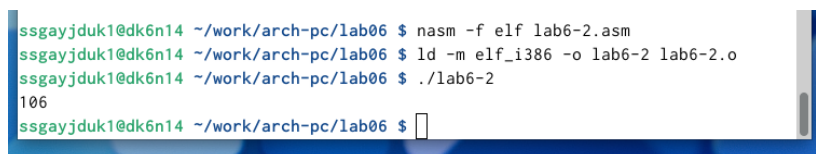
Рисунок 2.6: image6

A screenshot of a terminal window with a blue background. The terminal shows the following assembly code:

```
mc [ssgayjduk1@dk6n1  
/afs/.dk.sci.pfu.edu.  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, '6'  
mov ebx, '4'  
add eax, ebx  
call iprintLF  
call quit
```

Рисунок 2.7: image7

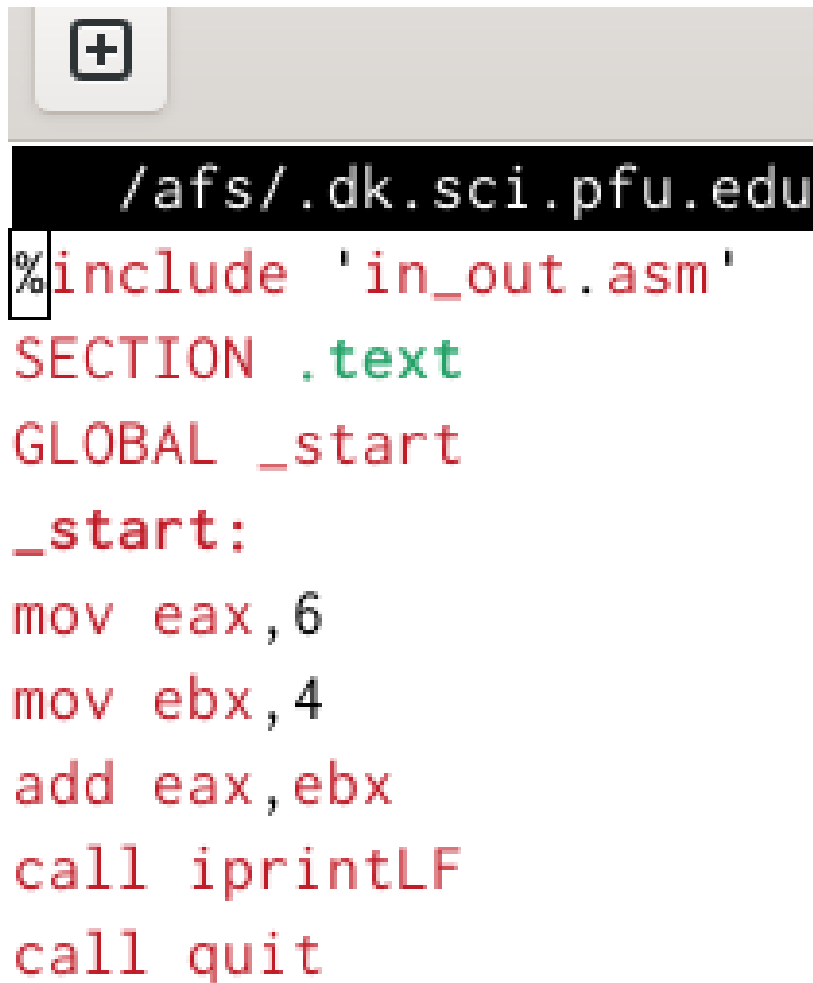
Создадим исполняемый файл и запустим его (рис. 2.8).

A screenshot of a terminal window showing the following commands and output:

```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./lab6-2  
106  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.8: image8

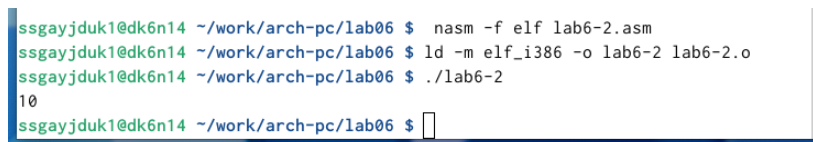
Заменяем символы на числа (рис. 2.9).



```
/afs/.dk.sci.pfu.edu
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рисунок 2.9: image9

Создадим исполняемый файл и запустим его (рис. 2.10).

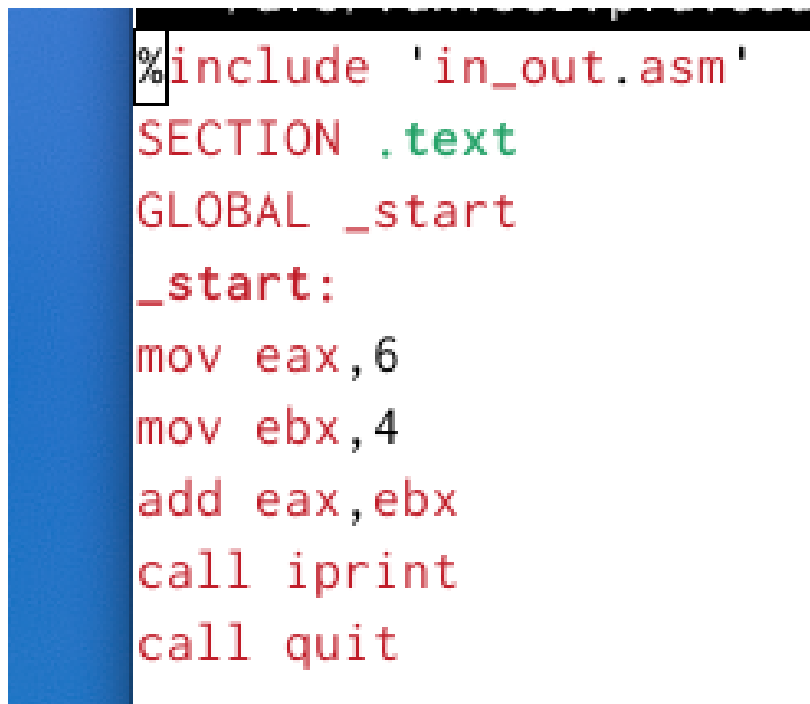


```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./lab6-2
10
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.10: image10

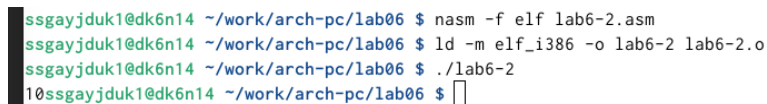
Результатом будет выведено 10 и командная строка будет ниже.

Заменяем функцию `iprintLF` на `iprint`. Создадим исполняемый файл и запустим его (рис. 2.11)



```
%  
include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, 6  
mov ebx, 4  
add eax, ebx  
call iprint  
call quit
```

Рисунок 2.11: image11



```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./lab6-2  
10ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.12: image12

Результатом будет выведено 10 и командная строка идет сразу же после вывода.

Вывод функций `iprintLF` и `iprint` отличается тем, что в первом случае командная строка смещается на следующую строку, а во втором - остается и начинается сразу после вывода.

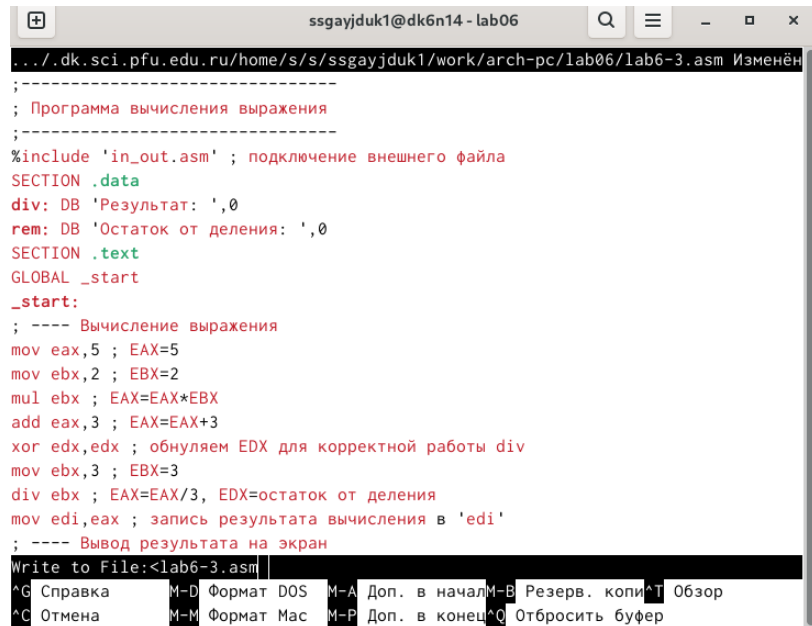
Создадим файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06` (рис. 2.13).



```
10ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm  
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.13: image13

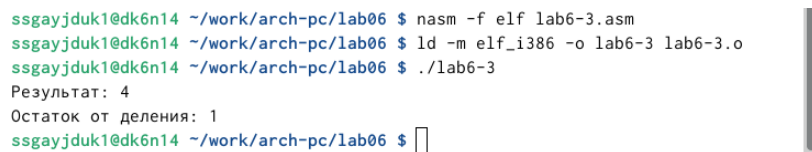
Введем в lab6-3.asm программу из условия (рис. 2.14).



```
ssgayjduk1@dk6n14 - lab06
../dk.sci.pfu.edu.ru/home/s/ssgayjduk1/work/arch-pc/lab06/lab6-3.asm Изменён
;-----
; Программа вычисления выражения
;-----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
Write to File:lab6-3.asm
^G Справка М-Д Формат DOS М-А Доп. в началм-В Резерв. копи^T Обзор
^C Отмена М-М Формат Mac М-Р Доп. в конц^O Отбросить буфер
```

Рисунок 2.14: image14

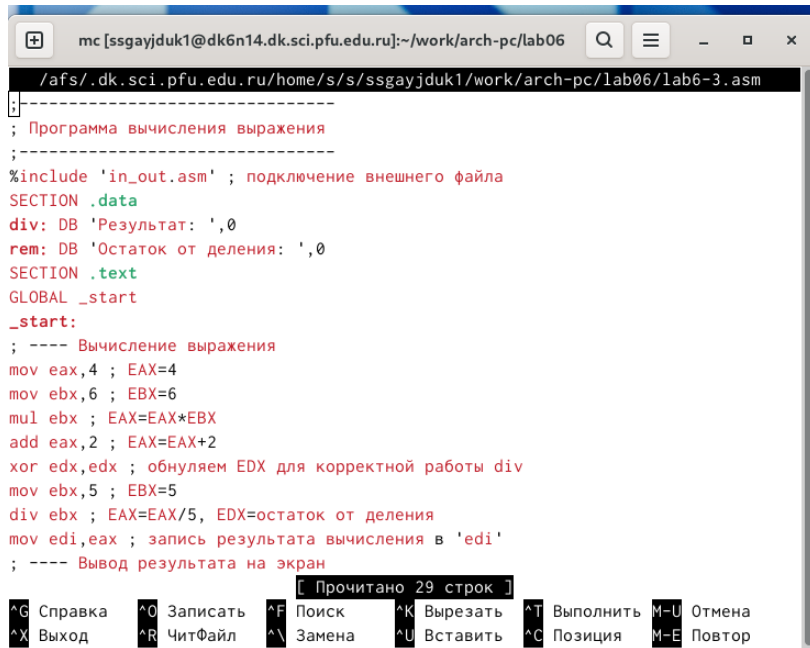
Создадим исполняемый файл и запустим его (рис. 2.15).



```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

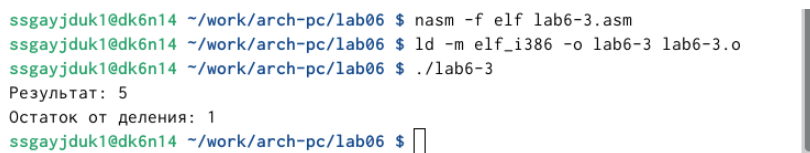
Рисунок 2.15: image15

Изменим текст программы для вычисления выражения $f(x)=(4*6+2)/5$, создадим исполняемый файл и проверим его работу (рис. 2.16)



```
mc [ssgayjduk1@dk6n14.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab06
/afs/.dk.sci.pfu.edu.ru/home/s/s/ssgayjduk1/work/arch-pc/lab06/lab6-3.asm
;-----
; Программа вычисления выражения
;-----
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
[ Прочитано 29 строк ]
^G Справка ^O Записать ^F Поиск ^K Вырезать ^T Выполнить M-U Отмена
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^C Позиция M-E Повтор
```

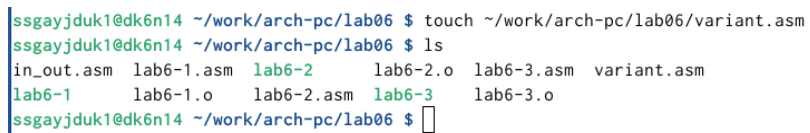
Рисунок 2.16: image16



```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.17: image17

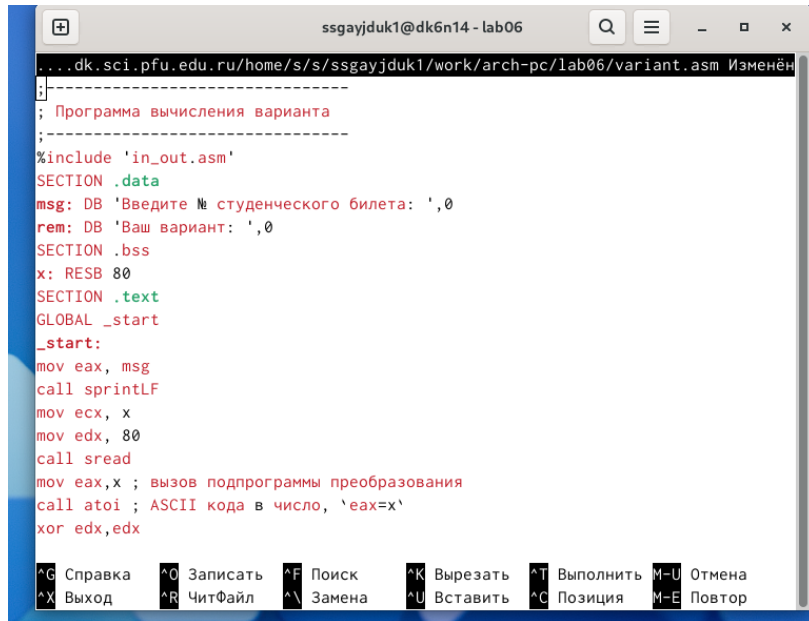
Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06 и проверим его существование (рис. 2.18)



```
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1.asm lab6-2 lab6-2.o lab6-3.asm variant.asm
lab6-1 lab6-1.o lab6-2.asm lab6-3 lab6-3.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```

Рисунок 2.18: image18

Введем в файл текст программы из листинга (рис. 2.19)



```
...dk.sci.pfu.edu.ru/home/s/s/sgayjduk1/work/arch-pc/lab06/variant.asm Изменен
;-----
; Программа вычисления варианта
;-----
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
```

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

Рисунок 2.19: image19




Создадим исполняемый файл и запустим его, а также проверим результат работы программы вычислив номер варианта аналитически (с помощью языка питона) (рис. 2.20)


```
sgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
sgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
sgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1032253645
Ваш вариант: 6
sgayjduk1@dk6n14 ~/work/arch-pc/lab06 $
```


Рисунок 2.20: image20



```
4 s_n=1032253645
5 b=20
6 print(s_n%20+1)
7
```

Ln: 6, Col: 15

 Run  Share  Command Line Arguments

 6



 ** Process exited - Return Code: 0 **




Рисунок 2.21: image21

Ответ сходится.

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения „Ваш вариант:“?

За вывод на экран сообщения „Ваш вариант:“ отвечают строки `mov eax,rem` и `call sprint(image23)`, `rem` - название переменной, в которой находится сообщение „Ваш вариант:“. (рис. 2.22)

```
rem: DB 'Ваш вариант: ',0
```

Рисунок 2.22: image22

```
mov eax,rem
call sprint
```

Рисунок 2.23: image23

2. Для чего используются следующие инструкции? (рис. 2.24)

2. Для чего используются следующие инструкции?

```
mov  ecx, x
mov  edx, 80
call sread
```

Рисунок 2.24: image24

mov ecx, x - используется для указания адреса переменной x в регистре ecx. mov edx, 80 - используется для указания размера или длины в байтах, в данной ситуации 80 call sread - вызов функции sread для чтения выше заданного кода.

3. Для чего используется инструкция «call atoi»?

call atoi - вызов функции atoi для преобразования ASCII строки в целое число.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

За вычисления варианта отвечают строки (рис. 2.25)

```
mov  eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor  edx, edx
mov  ebx, 20
div  ebx
inc  edx
```

Рисунок 2.25: image25

5. В какой регистр записывается остаток от деления при выполнении инструкции «div ebx»?

Остаток от деления при выполнении инструкции «div ebx» записывается в edx.

6. Для чего используется инструкция «inc edx»?

Для увеличения значения регистра edx на единицу.

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

Строки mov eax,edx и call iprintLF (рис. 2.26)

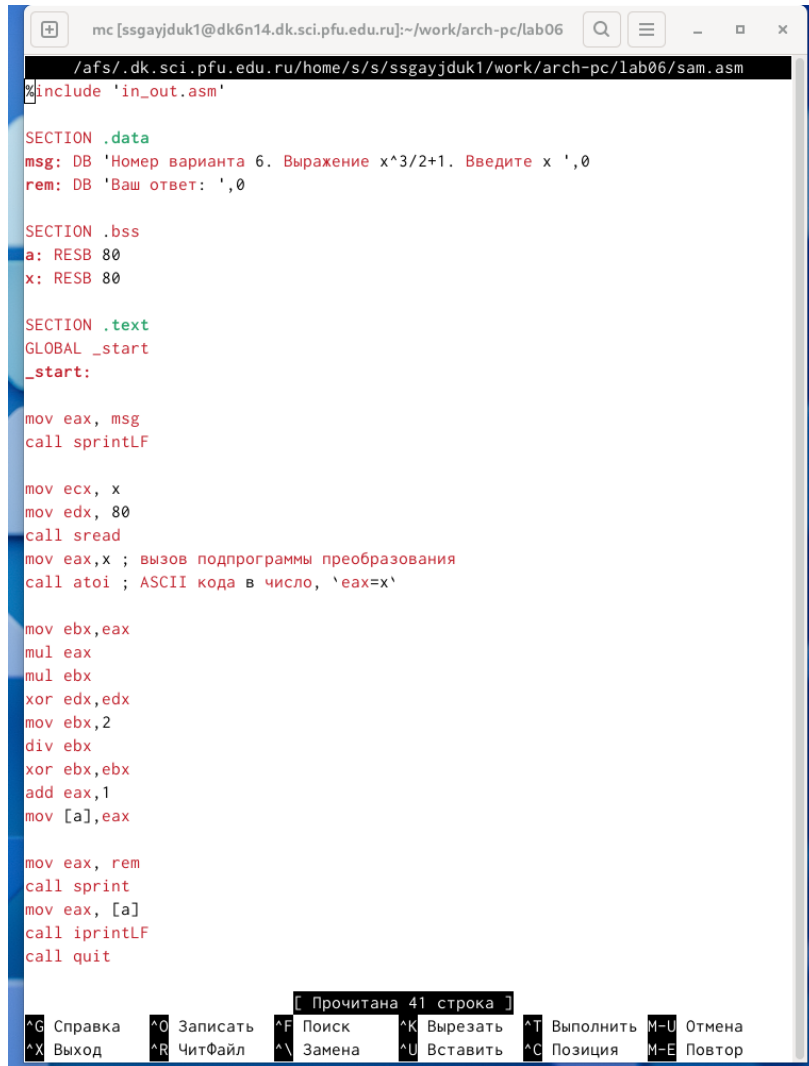


```
mov eax,edx  
call iprintLF
```

Рисунок 2.26: image26

3 Задания для самостоятельной работы

Напишем программу вычисления выражения $x^{3/2}+1$ (рис. 3.1)



```
mc [ssgayjduk1@dk6n14.dk.sci.pfu.edu.ru]:~/work/arch-pc/lab06

/afs/.dk.sci.pfu.edu.ru/home/s/s/ssgayjduk1/work/arch-pc/lab06/sam.asm
#include 'in_out.asm'

SECTION .data
msg: DB 'Номер варианта 6. Выражение x^3/2+1. Введите x ',0
rem: DB 'Ваш ответ: ',0

SECTION .bss
a: RESB 80
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintf

mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'

mov ebx, eax
mul eax
mul ebx
xor edx, edx
mov ebx, 2
div ebx
xor ebx, ebx
add eax, 1
mov [a], eax

mov eax, rem
call sprintf
mov eax, [a]
call iprintLF
call quit

[ Прочитана 41 строка ]
^G Справка ^O Записать ^F Поиск ^K Вырезать ^T Выполнить M-U Отмена
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^C Позиция M-E Повтор
```

Рисунок 3.1: image27

Создадим исполняемый файл и проверим его работу для значений $x=2$ и $x=5$, заданных в условии (рис. 3.2)

```

ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ nasm -f elf sam.asm
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o sam sam.o
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./sam
Номер варианта 6. Выражение  $x^3/2+1$ . Введите x
2
Ваш ответ: 5
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ ./sam
Номер варианта 6. Выражение  $x^3/2+1$ . Введите x
5
Ваш ответ: 63
ssgayjduk1@dk6n14 ~/work/arch-pc/lab06 $ 

```

Рисунок 3.2: image28

Отправим на Github (рис. 3.3)

```

ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab
06/report $ git add .
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab
06/report $ git commit -am 'feat(main): add lab-6'
[master 2b93473] feat(main): add lab-6
18 files changed, 290 insertions(+)
create mode 100644 labs/lab06/report/arch-pc--lab06--report.docx
create mode 100644 labs/lab06/report/arch-pc--lab06--report.pdf
create mode 100644 labs/lab06/report/in_out.asm
create mode 100755 labs/lab06/report/lab6-1
create mode 100644 labs/lab06/report/lab6-1.asm
create mode 100644 labs/lab06/report/lab6-1.o
create mode 100755 labs/lab06/report/lab6-2
create mode 100644 labs/lab06/report/lab6-2.asm
create mode 100644 labs/lab06/report/lab6-2.o
create mode 100755 labs/lab06/report/lab6-3
create mode 100644 labs/lab06/report/lab6-3.asm
create mode 100644 labs/lab06/report/lab6-3.o
create mode 100755 labs/lab06/report/sam
create mode 100644 labs/lab06/report/sam.asm
create mode 100644 labs/lab06/report/sam.o
create mode 100755 labs/lab06/report/variant
create mode 100644 labs/lab06/report/variant.asm
create mode 100644 labs/lab06/report/variant.o
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab
06/report $ git push
Перечисление объектов: 27, готово.
Подсчет объектов: 100% (27/27), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (23/23), готово.
Запись объектов: 100% (23/23), 2.22 МиБ | 3.30 МиБ/с, готово.
Total 23 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), completed with 2 local objects.
To github.com:SofiaGayduk/study_2025-2026_arh-pc.git
 a0bb643..2b93473 master -> master
ssgayjduk1@dk6n14 ~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab
06/report $ 

```

Рисунок 3.3: image29

4 Выводы

Мы изучили арифметические инструкции языка ассемблера NASM.