

# **Отчёт по лабораторной работе №6**

**дисциплина: Архитектура компьютеров**

Маслова Анна Павловна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
<b>3</b>	<b>Выполнение заданий для самостоятельной работы</b>	<b>18</b>
<b>4</b>	<b>Выводы</b>	<b>22</b>
	<b>Список литературы</b>	<b>23</b>

# Список иллюстраций

2.1	Создание lab06 и файла lab6-1.asm . . . . .	6
2.2	Текст программы в файле lab6-1.asm . . . . .	7
2.3	Запуск исполняемого файла lab6-1 . . . . .	7
2.4	Изменённый файл lab6-1.asm . . . . .	8
2.5	Запуск изменённого файла lab6-1 . . . . .	8
2.6	Таблица ASCII . . . . .	9
2.7	Создание файла lab6-2.asm . . . . .	9
2.8	Текст программы в файле lab6-2.asm . . . . .	10
2.9	Запуск исполняемого файла lab6-2 . . . . .	10
2.10	Изменённый файл lab6-2.asm . . . . .	11
2.11	Запуск изменённого файла lab6-2 . . . . .	11
2.12	Функция iprint в файле lab6-2.asm . . . . .	12
2.13	Запуск lab6-2 с функцией iprint . . . . .	12
2.14	Создание lab6-3.asm . . . . .	13
2.15	Текст программы в файле lab6-3.asm . . . . .	13
2.16	Запуск исполняемого файла lab6-3 . . . . .	14
2.17	Изменённый файл lab6-3.asm . . . . .	14
2.18	Запуск изменённого файла lab6-3 . . . . .	15
2.19	Создание файла variant.asm . . . . .	15
2.20	Текст файла variant.asm . . . . .	16
2.21	Запуск исполняемого файла variant . . . . .	16
3.1	Создание файла func.asm . . . . .	18
3.2	Текст файла func.asm . . . . .	20
3.3	Запуск файла func . . . . .	20

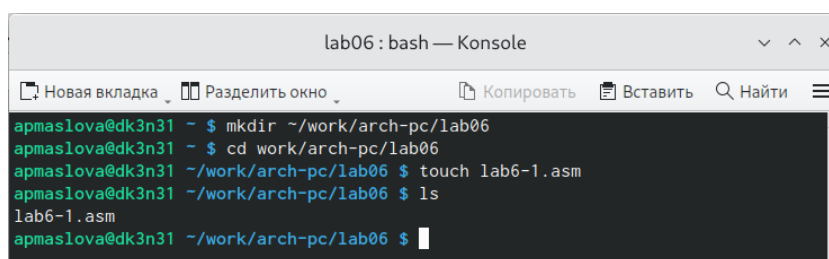
## Список таблиц

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Выполнение лабораторной работы

Сначала создадим каталог для программ лабораторной работы №6. Перейдём в него и создадим в нём файл *lab6-1.asm* (рис. 2.1).

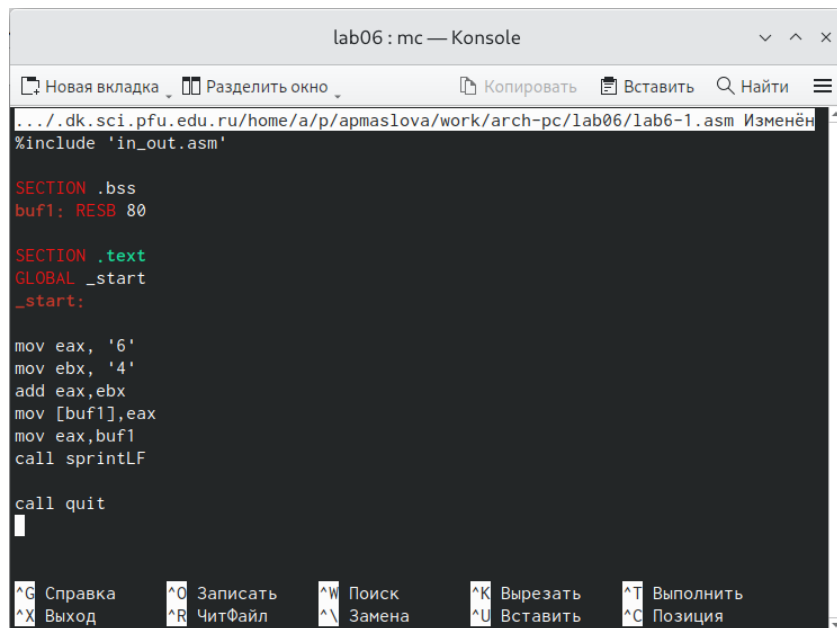


```
lab06 : bash — Konsole
Новая вкладка  Разделить окно  Копировать  Вставить  Найти  ≡
армаслова@dk3n31 ~ $ mkdir ~/work/arch-pc/lab06
армаслова@dk3n31 ~ $ cd work/arch-pc/lab06
армаслова@dk3n31 ~/work/arch-pc/lab06 $ touch lab6-1.asm
армаслова@dk3n31 ~/work/arch-pc/lab06 $ ls
lab6-1.asm
армаслова@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 2.1: Создание lab06 и файла lab6-1.asm

Файл создан.

Введём в этот файл текст программы вывода значения регистра *eax*, используя при этом функции из ранее загруженного файла *in\_out.asm*. С помощью команд *mov* в регистры *eax* и *ebx* записываем символы '6' и '4' соответственно. Затем к значению в регистре *eax* прибавим значение регистра *ebx* с помощью команды *add*. Результат этого сложения должен записаться в регистр *eax*. Выведем результат с помощью внешней функции *sprintLF*, однако для её работы в регистр *eax* должен быть помещён адрес. Используем для этого дополнительную переменную *buf1*, куда сначала запишем значение регистра, а затем адрес *buf1* поместим в *eax* (рис. 2.2).



```
lab06: mc — Konsole
.../.dk.sci.pfu.edu.ru/home/a/p/apmaslova/work/arch-pc/lab06/lab6-1.asm Изменён
#include 'in_out.asm'

SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

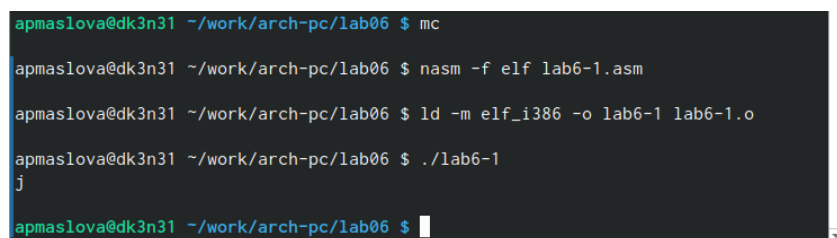
mov eax, '6'
mov ebx, '4'
add eax, ebx
mov [buf1], eax
mov eax, buf1
call sprintf

call quit
```

Рис. 2.2: Текст программы в файле lab6-1.asm

Видим, что текст сохранился в файле.

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

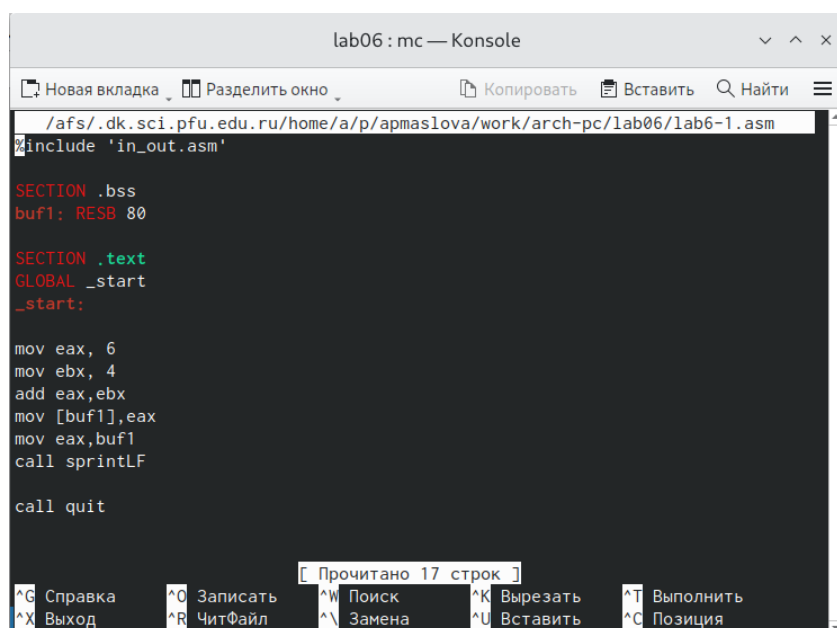


```
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ mc
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-1
j
apmaslova@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 2.3: Запуск исполняемого файла lab6-1

Ожидаемого вывода числа 10 не произошло. На экран вывелся символ j. Всё потому что команда *add* сложила двоичные коды символов '6' и '4', в результате чего получился код символа j (106).

Изменим текст программы и вместо символов '6' и '4' запишем числа 6 и 4 (рис. 2.4).



```
lab06: mc — Konsole
/afs/.dk.sci.pfu.edu.ru/home/a/p/apmaslova/work/arch-pc/lab06/lab6-1.asm
#include 'in_out.asm'

SECTION .bss
buf1: RESB 80

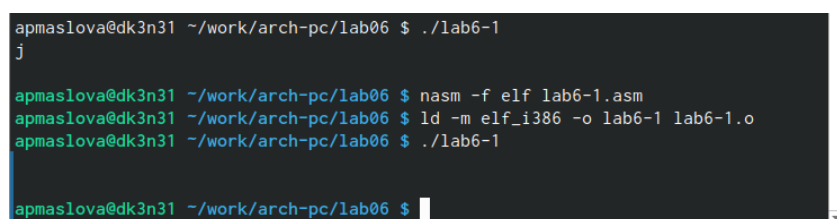
SECTION .text
GLOBAL _start
_start:

mov eax, 6
mov ebx, 4
add eax, ebx
mov [buf1], eax
mov eax, buf1
call printf

call _exit
```

Рис. 2.4: Изменённый файл lab6-1.asm

Создадим исполняемый файл, и при запуске мы видим пустую строку (рис. 2.5).



```
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-1
j

apmaslova@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-1

apmaslova@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 2.5: Запуск изменённого файла lab6-1

Число 10 снова не получилось. На экране мы увидели символ с кодом 10. Воспользуемся таблицей *ASCII*, чтобы определить, какому символу соответствует код 10 (рис. 2.6).



Таблица 12.1. Таблица символов ASCII

DEC	OCT	HEX	BIN	Symbol
0	0	0x00	0	NUL, \0
1	1	0x01	1	SOH
2	2	0x02	10	STX
3	3	0x03	11	ETX
4	4	0x04	100	EOT
5	5	0x05	101	ENQ
6	6	0x06	110	ACK
7	7	0x07	111	BEL
8	10	0x08	1000	BS
9	11	0x09	1001	HT, \t
10	12	0x0A	1010	LF, \n

Рис. 2.6: Таблица ASCII

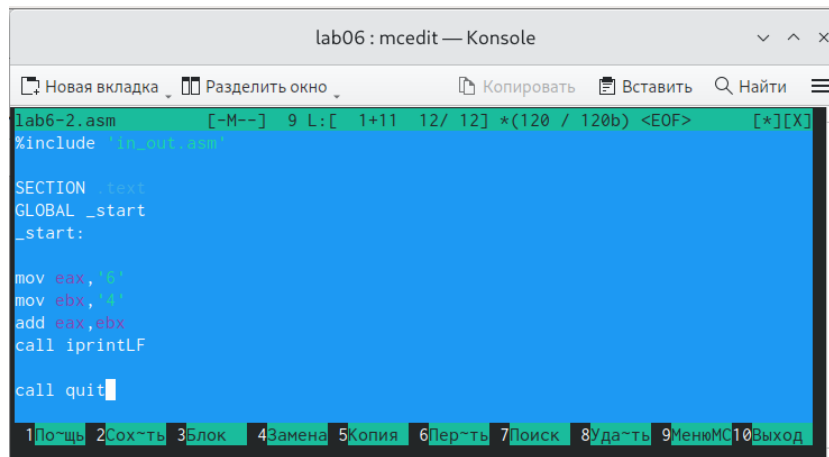
Коду 10 соответствует символ переноса строки. Поэтому мы увидели на экране пустую строку.

Далее посмотрим, как работают функции из файла *in\_out.asm* по преобразованию ASCII символов в числа и обратно. Создадим файл *lab6-2.asm* в том же каталоге (рис. 2.7).

```
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ touch lab6-2.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 2.7: Создание файла lab6-2.asm

Теперь в этот файл запишем ту же программу по выводу значения регистра *eax*, но уже с использованием подпрограммы *iprintLF* (рис. 2.8).



```
lab6-2.asm      [-M--]  9 L: [ 1+11 12/ 12] *(120 / 120b) <EOF>  [*][X]
#include "iprint.asm"

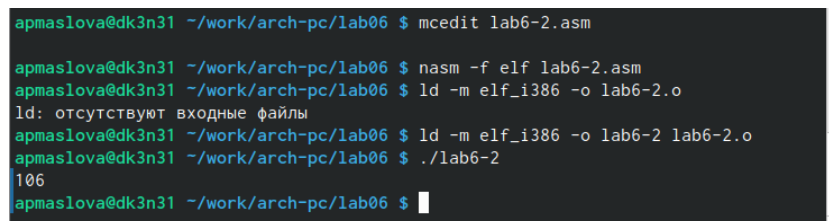
SECTION .text
GLOBAL _start
_start:

mov  eax, '6'
mov  ebx, '4'
add  eax, ebx
call iprintLF

call quit
```

Рис. 2.8: Текст программы в файле lab6-2.asm

Создадим исполняемый файл и проверим его работу (рис. 2.9).

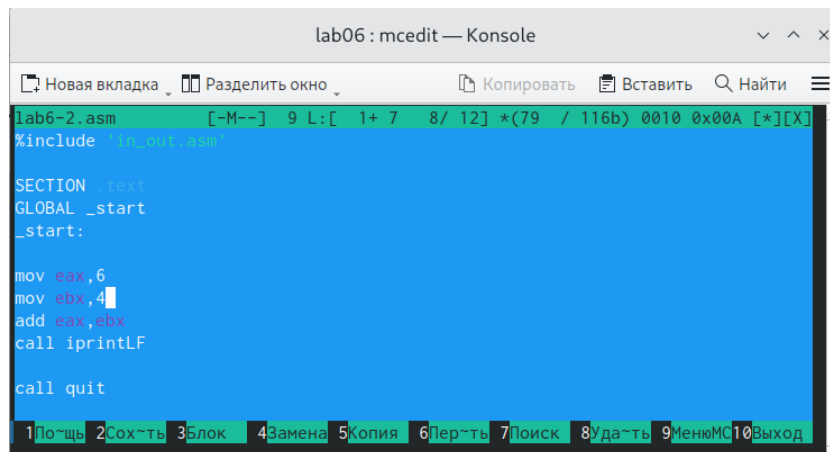


```
армаслова@dk3n31 ~/work/arch-pc/lab06 $ mcedit lab6-2.asm
армаслова@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
армаслова@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2.o
ld: отсутствуют входные файлы
армаслова@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
армаслова@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-2
106
армаслова@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 2.9: Запуск исполняемого файла lab6-2

На экране мы видим число 106. Оно является результатом сложения кодов символов '6' и '4' ( $54+52=106$ ). Функция *iprintLF* позволяет вывести именно код получившегося в результате сложения символа, а не сам символ, которому этот код соответствует.

Теперь заменим символы '6' и '4' на числа, как в предыдущем примере (рис. 2.10).



```
lab06 : mcedit — Konsole
lab6-2.asm  [-M--]  9 L: [ 1+ 7  8/ 12] *(79 / 116b) 0010 0x00A [*][X]
#include "iprint.asm"

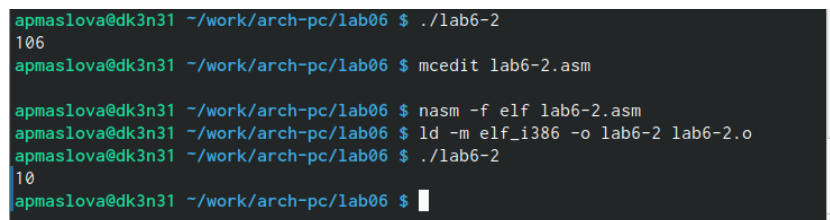
SECTION .text
GLOBAL _start
_start:

mov  eax, 6
mov  ebx, 4
add  eax, ebx
call iprintLF

call quit
```

Рис. 2.10: Изменённый файл lab6-2.asm

Создадим исполняемый файл и запустим программу (рис. 2.11).



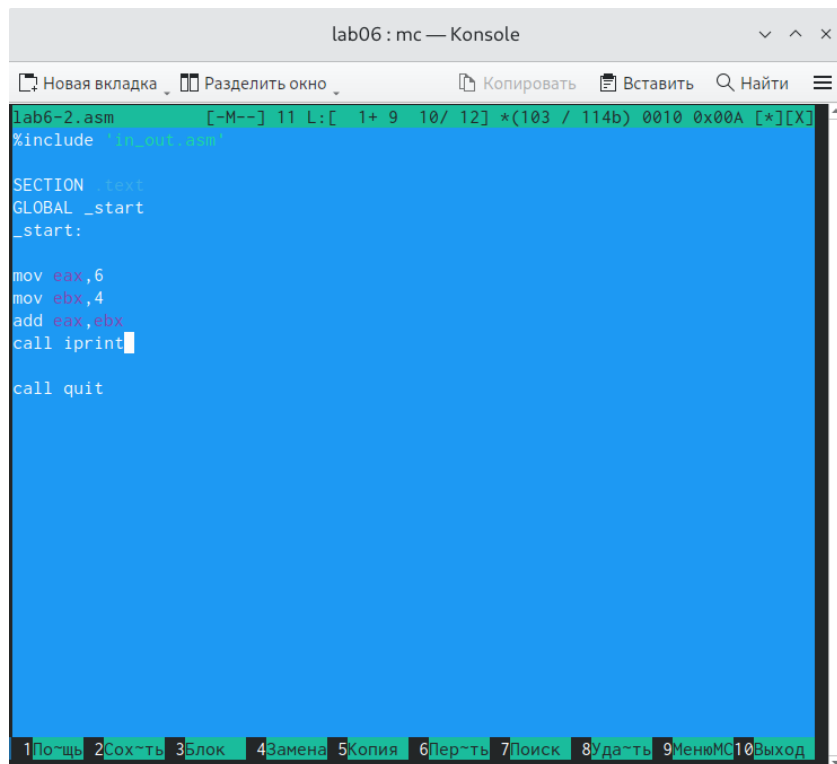
```
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-2
106
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ mcedit lab6-2.asm

apmaslova@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-2
10
apmaslova@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 2.11: Запуск изменённого файла lab6-2

На этот раз на экране видим результат сложения чисел 6 и 4 - число 10.

А теперь заменим функцию *iprintLF* на функцию *iprint* (рис. 2.12).



```
lab6-2.asm [-M--] 11 L: [ 1+ 9 10/ 12] *(103 / 114b) 0010 0x00A [*][X]
#include "in_out.asm"

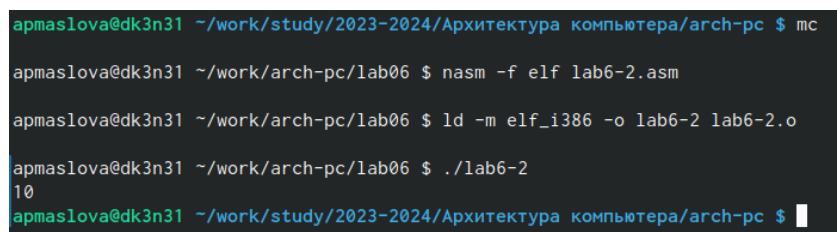
SECTION .text
GLOBAL _start
_start:

mov eax,6
mov ebx,4
add eax,ebx
call iprint

call quit
```

Рис. 2.12: Функция `iprint` в файле `lab6-2.asm`

Создадим исполняемый файл и запустим (рис. 2.13).



```
apmaslova@dk3n31 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ mc
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-2
10
apmaslova@dk3n31 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $
```

Рис. 2.13: Запуск `lab6-2` с функцией `iprint`

Как мы видим, функция `iprint` в отличие от `iprintLF` не выводит на экран перенос строки после числа 10.

Теперь познакомимся с арифметическими операциями NASM. Напишем программу вычисления арифметического выражения  $f(x) = (5 * 2 + 3)/3$ .

Создадим файл `lab6-3.asm` в том же каталоге `lab06` (рис. 2.14).

```

apmaslova@dk3n31 ~/work/arch-pc/lab06 $ touch lab6-3.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $

```

Рис. 2.14: Создание lab6-3.asm

Введём в созданный файл код программы вычисления вышеуказанного выражения (рис. 2.15).

```

lab6-3.asm  [----]  0 L: [ 1+27  28/ 28] *(769 / 769b) <EOF>  [*][X]
#include "in_out.asm"
SECTION .data
div: DB "Результат: ",0
rem: DB "Остаток от деления: ",0
SECTION .text
GLOBAL _start
_start:
; Вычисление выражения
mov eax,5
mov ebx,2
mul ebx ;EAX=EAX*EBX
add eax,3
xor edx,edx ;обнуляем EDX для корректной работы div
mov ebx,3
div ebx ;EAX=EAX/3, EDX - остаток от деления
mov edi,eax
;Вывод результата на экран
mov eax,div; Вызов подпрограммы печати
call sprint ;Сообщение "Результат: "
mov eax,edi ;Вызов подпрограммы печати значения
call iprintLF ;Из EDI в виде символов
mov eax,rem
call sprint
mov eax,edx
call iprintLF

call quit

```

Рис. 2.15: Текст программы в файле lab6-3.asm

Создадим исполняемый файл и запустим его (рис. 2.16).

```

apmaslova@dk3n31 ~/work/arch-pc/lab06 $ mcedit lab6-3.asm

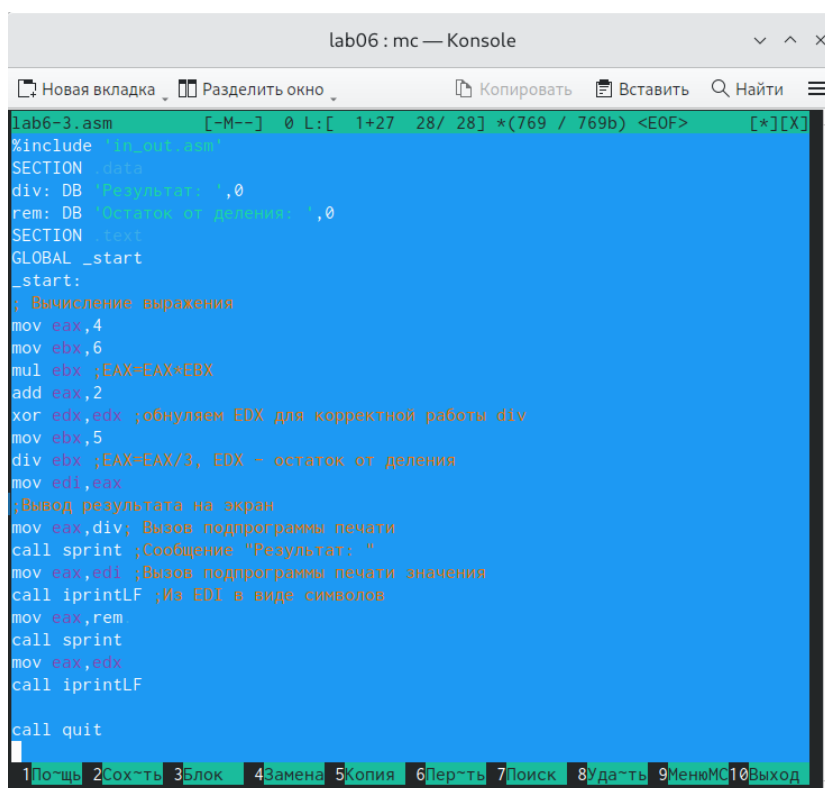
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
apmaslova@dk3n31 ~/work/arch-pc/lab06 $

```

Рис. 2.16: Запуск исполняемого файла lab6-3

Как мы видим, программа работает корректно.

Изменим текст программы так, чтобы она вычисляла выражение  $f(x) = (4 * 6 + 2) / 5$  (рис. 2.17).



```

lab6-3.asm  [-M--]  0 L: [ 1+27 28/ 28] *(769 / 769b) <EOF>  [*][X]
#include "in_out.asm"
SECTION .data
div: DB "Результат: ",0
rem: DB "Остаток от деления: ",0
SECTION .text
GLOBAL _start
_start:
; Вычисление выражения
mov eax,4
mov ebx,6
mul ebx ;EAX=EAX*EBX
add eax,2
xor edx,edx ;обнуляем EDX для корректной работы div
mov ebx,5
div ebx ;EAX=EAX/3, EDX - остаток от деления
mov edi,eax
;Вывод результата на экран
mov eax,div; Вызов подпрограммы печати
call sprint ;Сообщение "Результат: "
mov eax,edi ;Вызов подпрограммы печати значения
call iprintLF ;ИЗ EDI в виде символов
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit

```

Рис. 2.17: Изменённый файл lab6-3.asm

Создадим исполняемый файл и проверим работу программы (рис. 2.18).

```

армаслова@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
армаслова@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
армаслова@dk3n31 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
армаслова@dk3n31 ~/work/arch-pc/lab06 $

```

Рис. 2.18: Запуск изменённого файла lab6-3

Программа верно вычисляет выражение  $f(x) = (4 * 6 + 2)/5$ .

Напишем программу вычисления варианта задания по номеру студенческого билета, которая выводит запрос на введение № студенческого билета, вычисляет номер варианта по формуле  $(\xi_n \bmod 20) + 1$  и выводит на экран номер варианта. Ввод с клавиатуры осуществляется в символьном виде, и чтобы все арифметические функции работали корректно, нужно использовать функцию *atoi* из файла *in\_out.asm*.

Создадим файл *variant.asm* в каталоге *lab06* (рис. 2.19).

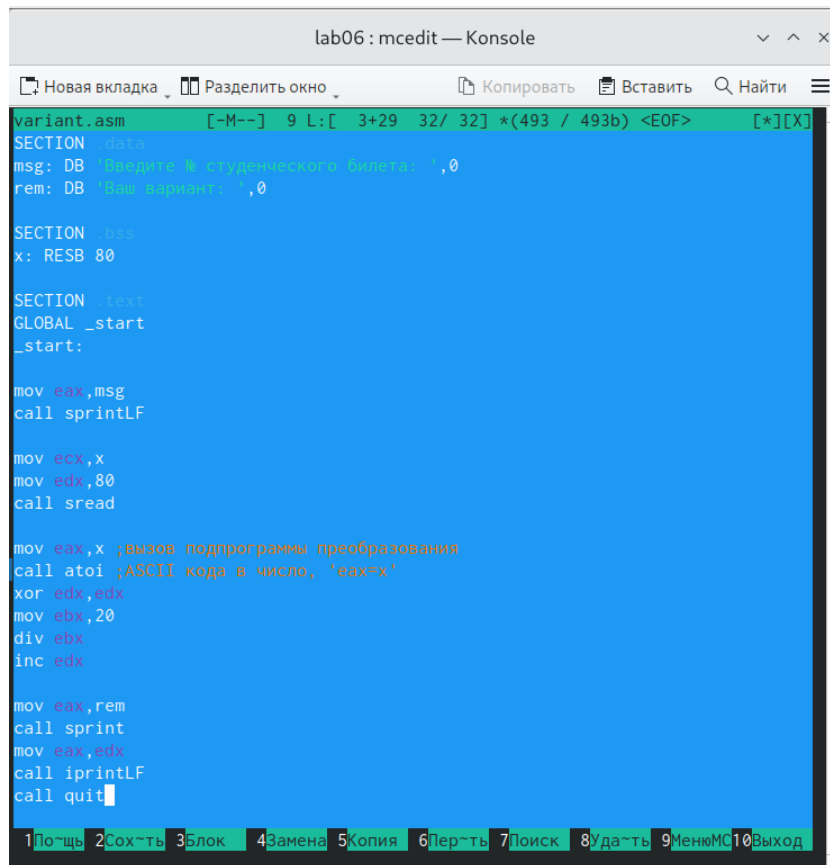
```

армаслова@dk3n31 ~/work/arch-pc/lab06 $ touch variant.asm
армаслова@dk3n31 ~/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
армаслова@dk3n31 ~/work/arch-pc/lab06 $

```

Рис. 2.19: Создание файла variant.asm

В этот файл введём текст программы вычисления варианта задания по номеру студенческого билета (рис. 2.20).



```
variant.asm [-M--] 9 L: [ 3+29 32/ 32] *(493 / 493b) <EOF> [*][X]
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 sprintLF

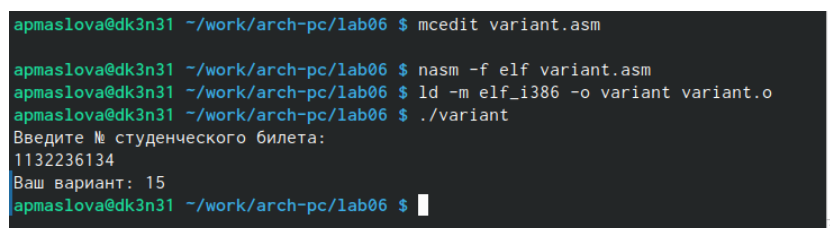
mov ecx,x
mov edx,80
call sread

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

mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 2.20: Текст файла variant.asm

Создадим исполняемый файл и проверим его работу (рис. 2.21).



```
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ mcedit variant.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132236134
Ваш вариант: 15
apmaslova@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 2.21: Запуск исполняемого файла variant

Ввели номер студенческого билета, программа посчитала номер варианта по формуле и вывела на экран число 15. Проверим аналитически: остаток от деления 1132236132 на 20 равен 14, а 14+1=15. Следовательно, программа работает правильно.



Ответим на вопросы по лабораторной работе:

1. В листинге 6.4 за вывод на экран сообщения 'Ваш вариант:' отвечают строки:

- `rem: DB 'Ваш вариант:',0` ; в строке мы объявляем переменную `rem`, куда записали искомую строку
- `mov eax,rem` ; помещаем строку в регистр `eax`
- `call sprint` ; вызываем подпрограмму вывода из файла `in_out.asm`

2. Инструкции `mov ecx,x -> mov edx,80 -> call sread` используются для того, чтобы ввести с клавиатуры строку отведённого размера (80) и поместить её по адресу `x`. Для этого `x` помещаем в регистр `ecx`, а длину строки (80) - в регистр `edx`. `call sread` - вызов функции печати.

3. Инструкция `call atoi` используется для преобразования символов в числа.

4. За вычисление варианта отвечают строки:

- `mov eax,x` ; поместили `x` в регистр `eax`
- `call atoi` ; преобразование символов в число
- `xor edx,edx` ; обнуляем `edx`
- `mov ebx,20` : поместили в регистр `ebx` число 20
- `div ebx` ; поделили число, лежащее в `eax`, на число, лежащее в `ebx`
- `inc edx` ; `edx + 1`

5. Остаток от деления при выполнении `div ebx` записывается в регистр `edx`.

6. Инструкция `inc edx` используется для увеличения значения регистра `edx` на 1.

7. За вывод на экран результата вычислений отвечают строки:

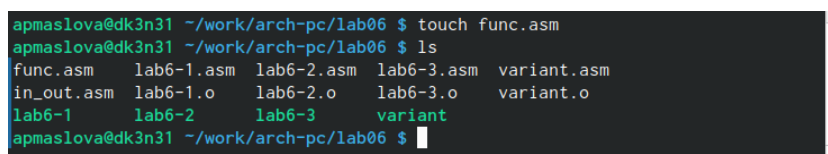
- `mov eax,edx` ; помещаем результат вычислений в регистр `eax`
- `call iprintLF` ; выводим на экран содержимое регистра `eax`

### 3 Выполнение заданий для самостоятельной работы

Необходимо написать программу, вычисляющую значение заданной функции  $f(x)$  в зависимости от введённого значения  $x$ . Варианту 15 соответствует формула следующей функции:

$$f(x) = (5 + x)^2 - 3$$

Создадим файл *func.asm* (рис. 3.1).



```
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ touch func.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ls
func.asm  lab6-1.asm  lab6-2.asm  lab6-3.asm  variant.asm
in_out.asm lab6-1.o    lab6-2.o    lab6-3.o    variant.o
lab6-1     lab6-2     lab6-3     variant
apmaslova@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 3.1: Создание файла *func.asm*

В этот файл введём необходимый текст программы, представленный на листинге 3.1 и на (рис. 3.2).

#### Листинг 3.1. Программа вычисления значения функции

;Вариант №15

```
%include 'in_out.asm'
```

```
SECTION .data
```

```

msg: DB 'x = ',0
rem: DB 'y = ',0
SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

; ВВОД x
mov eax,msg
call sprint
mov ecx,x
mov edx,80
call sread

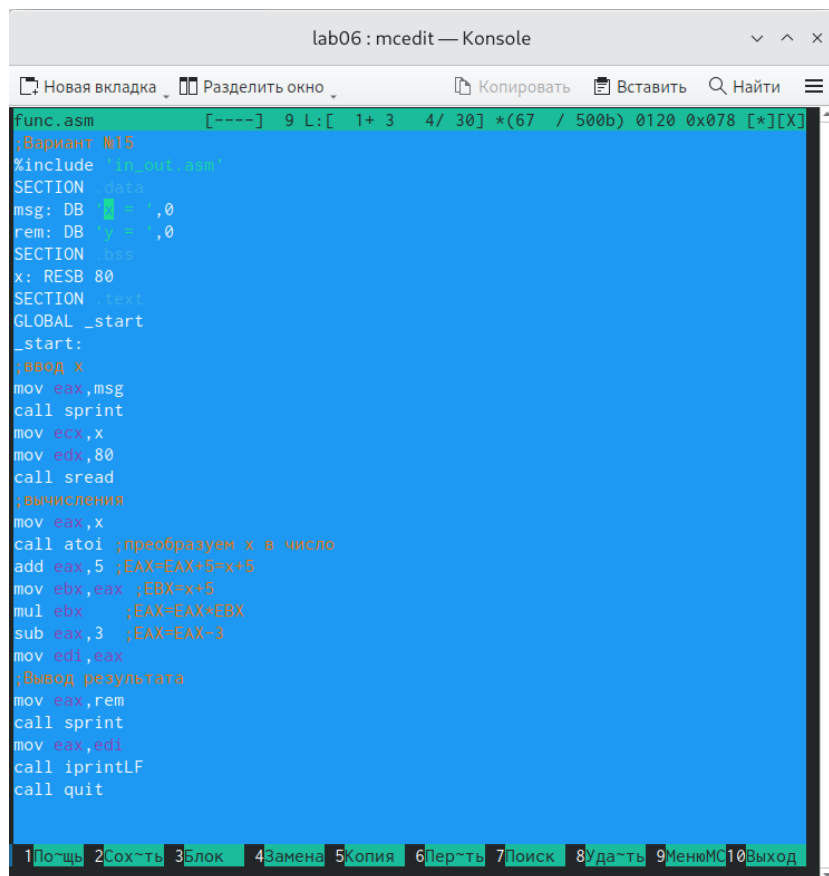
; ВЫЧИСЛЕНИЯ
mov eax,x
call atoi ;преобразуем x в число
add eax,5 ;EAX=EAX+5=x+5
mov ebx,eax ;EBX=x+5
mul ebx ;EAX=EAX*EBX
sub eax,3 ;EAX=EAX-3
mov edi,eax

; Вывод результата
mov eax,rem
call sprint
mov eax,edi

```

```
call iprintLF
```

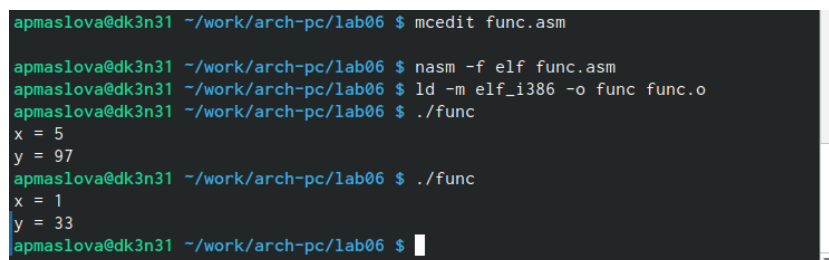
```
call quit
```



```
func.asm [----] 9 L: [ 1+ 3 4/ 30] *(67 / 500b) 0120 0x078 [*][X]
;Вариант №15
#include "in_out.asm"
SECTION .data
msg: DB "x = ",0
rem: DB "y = ",0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
;ввод x
mov eax,msg
call sprint
mov ecx,x
mov edx,80
call sread
;вычисления
mov eax,x
call atoi ;преобразуем x в число
add eax,5 ;EAX=EAX+5=x+5
mov ebx,eax ;EBX=x+5
mul ebx ;EAX=EAX*EBX
sub eax,3 ;EAX=EAX-3
mov edi,eax
;Вывод результата
mov eax,rem
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 3.2: Текст файла func.asm

Теперь создадим исполняемый файл и проверим его работу для значений  $x_1 = 5, x_2 = 1$  (рис. 3.3)



```
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ mcedit func.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ nasm -f elf func.asm
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o func func.o
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./func
x = 5
y = 97
apmaslova@dk3n31 ~/work/arch-pc/lab06 $ ./func
x = 1
y = 33
apmaslova@dk3n31 ~/work/arch-pc/lab06 $
```

Рис. 3.3: Запуск файла func

Как мы видим, программа работает корректно: вычисляются верные значения для  $f(x)$ :  $f(5)=97$ ,  $f(1)=33$ .

## 4 Выводы

Мы освоили арифметические инструкции языка ассемблера NASM, научились составлять арифметические программы.

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс,
- 11.