

Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Студент: Верниковская Екатерина Андреевна

Содержание

Цель работы	5
Задание	6
Выполнение лабораторной работы	7
Символьные и численные данные в NASM	7
Задание для самостоятельной работы	21
Выводы	24

Список таблиц

Список иллюстраций

1	Срздание первого файла	7
2	Копирование файла «in_out.asm»	8
3	Ввод текста программы	9
4	Создание исполняемого файла и его запуск	9
5	Изменение программы	10
6	Исполняемый файл + запуск	10
7	Создание файла «lab6-2.asm»	10
8	Ввод текста программы	11
9	Создание исполняемого файла и его запуск	11
10	Изменение программы	12
11	Исполняемый файл + запуск	12
12	Изменение программы	13
13	Исполняемый файл + запуск	13
14	Создание файла «lab6-3.asm»	13
15	Ввод текста программы	15
16	Исполняемый файл + запуск	15
17	Изменение программы	16
18	Исполняемый файл + запуск	17
19	Создание файла «variant.asm»	17
20	Ввод текста программы	19
21	Исполняемый файл + запуск	19
22	Создание файла «lab6-4.asm»	21
23	Ввод программы	22
24	Исполняемый файл + запуск	23

Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

Задание

1. Создать каталог для программ лабораторной работы №6 и в нём создать файл «lab6-1.asm».
2. Ввести в файл «lab6-1.asm» определённый текст программы. Создать исполняемый файл и запустить его.
3. Изменить текст программы. Снова создать исполняемый файл и запустить его.
4. Создать файл «lab6-2.asm» и файл «lab6-3.asm». С этими файлами проделать такие же действия как и с файлом «lab6-1.asm».
5. Создать файл «variant.asm». Написать программу, которая вычисляет вариант задания по номеру студенческого билета.
6. Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из нужной таблицы вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы.
7. Создать исполняемый файл и проверить его работы для значений x_1 и x_2 из нужной таблицы.

Выполнение лабораторной работы

Символьные и численные данные в NASM

В созданном каталоге «~/work/arch-pc/lab06» создаём файл «lab6-1.asm» (рис. [-@fig:001])

```
eavernikovskaya@ubuntu-katerok:~$ mkdir ~/work/arch-pc/lab06
eavernikovskaya@ubuntu-katerok:~$ cd ~/work/arch-pc/lab06
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ touch lab6-1.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ls
lab6-1.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 1: Срздание первого файла

Копируем из каталога «~/work/arch-pc/lab05» файл «in_out.asm» (рис. [-@fig:002])

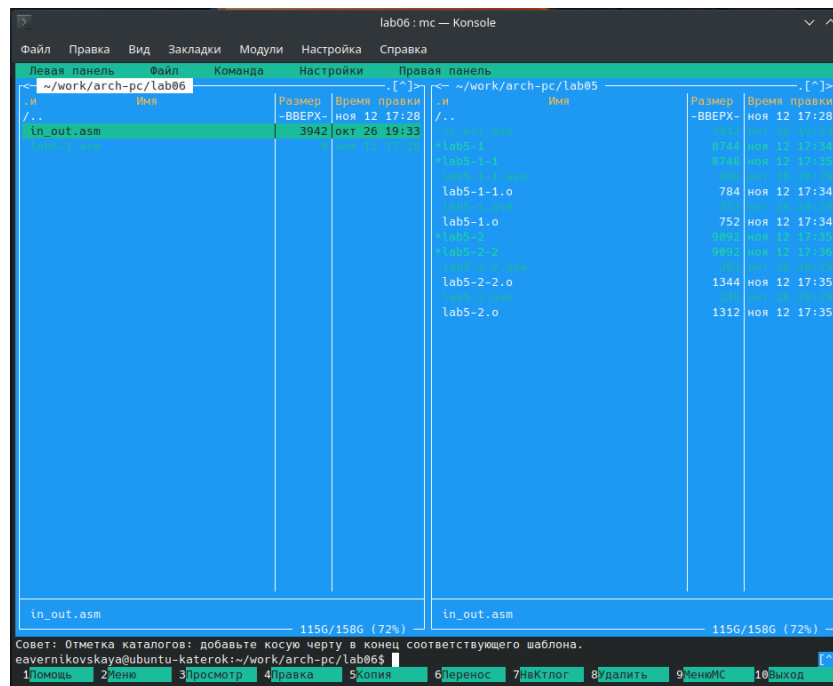


Рис. 2: Копирование файла «in_out.asm»

Вводим нужный текст программы (рис. [-@fig:003])

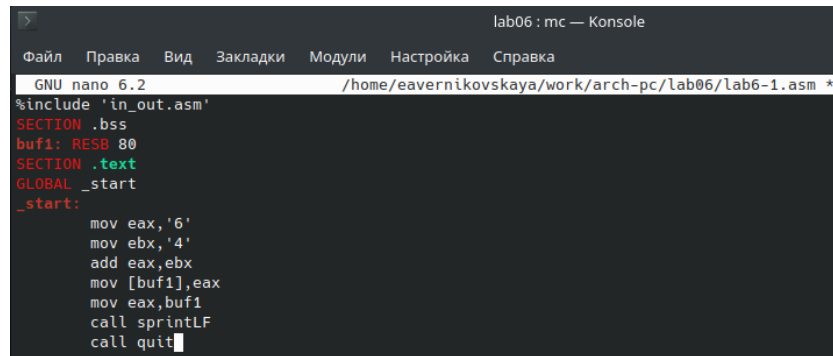
Текст программы:

```
%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 sprintLF
```

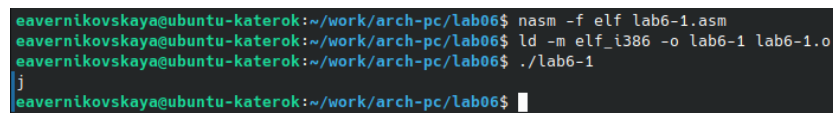

call quit



```
GNU nano 6.2 /home/eavernikovskaya/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 sprintLF
    call quit
```

Рис. 3: Ввод текста программы

Создаём исполняемый файл и запускаем его (рис. [-@fig:004])



```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ./lab6-1
6
j
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

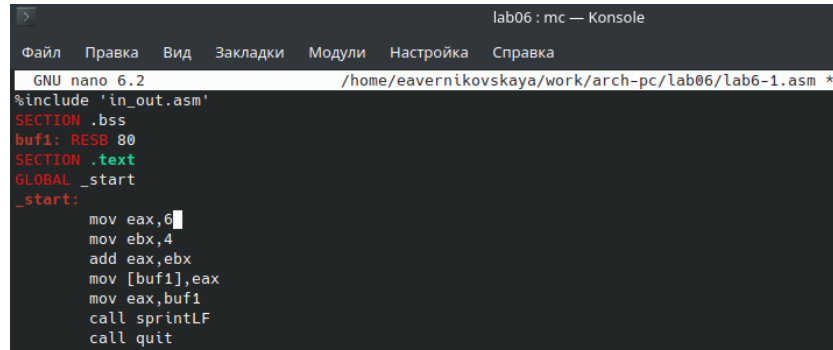
Рис. 4: Создание исполняемого файла и его запуск

Изменяем текст программы и вместо символов, записываем числа (рис. [-@fig:005])

Изменённый текст программы:

```
%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
```

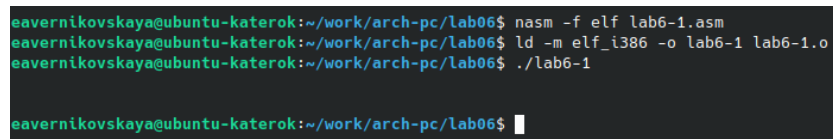
```
mov eax,buf1
call sprintLF
call quit
```



```
GNU nano 6.2 /home/eavernikovskaya/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 sprintLF
    call quit
```

Рис. 5: Изменение программы

Снова создаём исполняемый файл и запускаем его (рис. [-@fig:006])



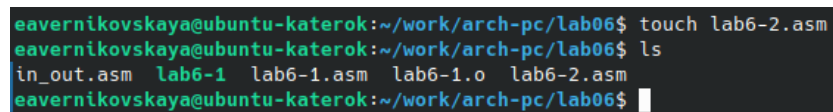
```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ./lab6-1

eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 6: Исполняемый файл + запуск

В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определяем какому символу соответствует код 10. Это символ перевода строки LF (Line Feed), который отображается при выводе на экран.

Создаём файл «lab6-2.asm» (рис. [-@fig:007])



```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ touch lab6-2.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 7: Создание файла «lab6-2.asm»

Вводим текст программы (рис. [-@fig:008])

Текст программы:

```

#include 'in_out.asm'

SECTION .text

GLOBAL _start

_start:

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

```

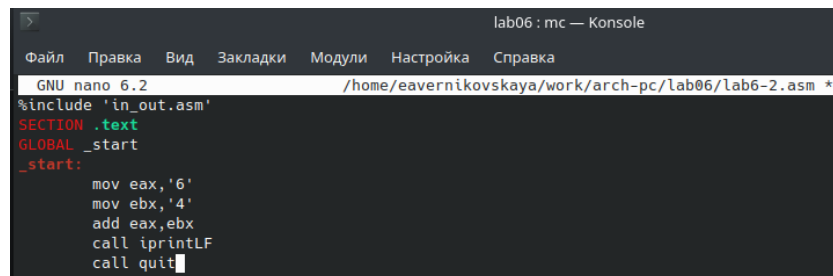


Рис. 8: Ввод текста программы

Создаём исполняемый файл и запускаем его (рис. [-@fig:009])

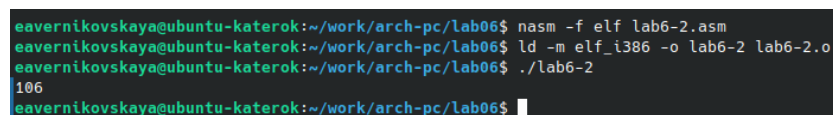


Рис. 9: Создание исполняемого файла и его запуск

Аналогично предыдущему примеру меняем символы на числа рис. [-@fig:010])

Изменённый текст программы:

```

#include 'in_out.asm'

SECTION .text

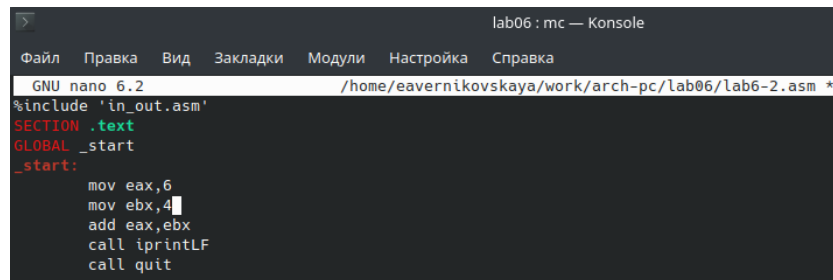
GLOBAL _start

```

```

_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprintLF
    call quit

```



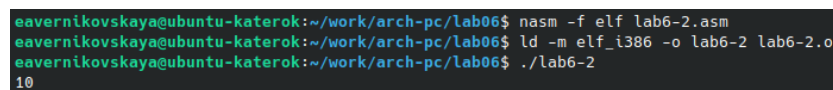
```

lab06 : mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
GNU nano 6.2 /home/eavernikovskaya/work/arch-pc/lab06/lab6-2.asm *
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprintLF
    call quit

```

Рис. 10: Изменение программы

Опять создаём исполняемый файл и запускаем его (рис. [-@fig:011])



```

eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ./lab6-2
10

```

Рис. 11: Исполняемый файл + запуск

При исполнении программы будет получен нужный результат - 10.

Меняем функцию iprintLF на iprint (рис. [-@fig:012])

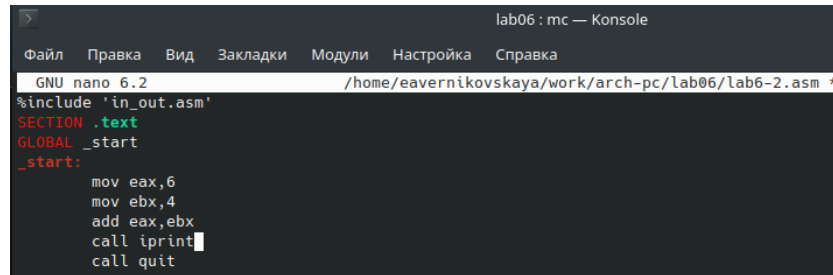
Изменённый текст программы:

```

#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,4

```

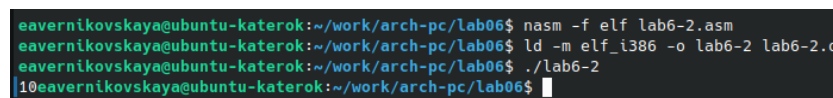
```
add eax,ebx
call iprint
call quit
```



```
lab06 : mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
GNU nano 6.2 /home/eavernikovskaya/work/arch-pc/lab06/lab6-2.asm *
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprint
    call quit
```

Рис. 12: Изменение программы

Опять создаём исполняемый файл и запускаем его (рис. [-@fig:013])

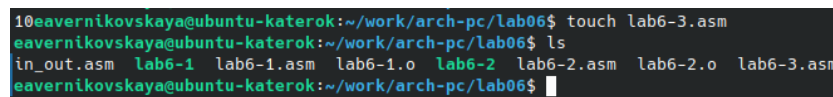


```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ./lab6-2
10eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 13: Исполняемый файл + запуск

Функция `iprintLF` выводит строку с добавлением символа LF (перевода строки), а функция `iprint` просто выводит строку без добавления этого символа (вывод будет продолжаться на той же строке).

Создаём файл «lab6-3.asm» (рис. [-@fig:014])



```
10eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ touch lab6-3.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2  lab6-2.asm  lab6-2.o  lab6-3.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 14: Создание файла «lab6-3.asm»

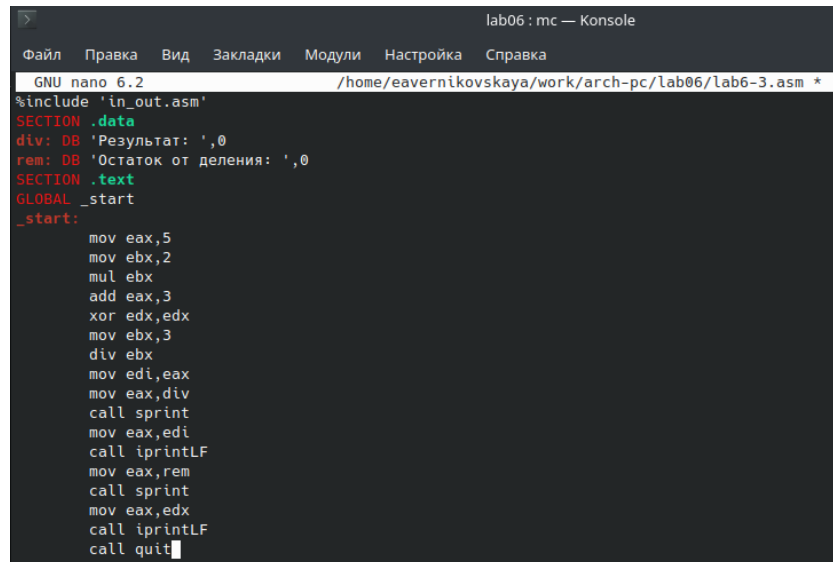
Вводим текст программы для вычисления выражения $f(x) = (5 \times 2 + 3)/3$ (рис. [-@fig:015])

Текст программы:

```

#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
    mov eax,5
    mov ebx,2
    mul ebx
    add eax,3
    xor edx,edx
    mov ebx,3
    div ebx
    mov edi,eax
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF
    call quit

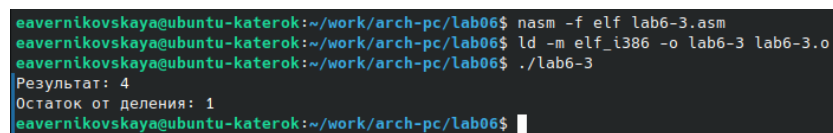
```



```
GNU nano 6.2 /home/eavernikovskaya/work/arch-pc/lab06/lab6-3.asm *
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
    mov eax,5
    mov ebx,2
    mul ebx
    add eax,3
    xor edx,edx
    mov ebx,3
    div ebx
    mov edi,eax
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF
    call quit
```

Рис. 15: Ввод текста программы

Создаём исполняемый файл и запускаем его (рис. [-@fig:016])



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

Рис. 16: Исполняемый файл + запуск

Изменяем текст программы для вычисления выражения $f(x) = (4 \times 6 + 2)/5$ (рис. [-@fig:017])

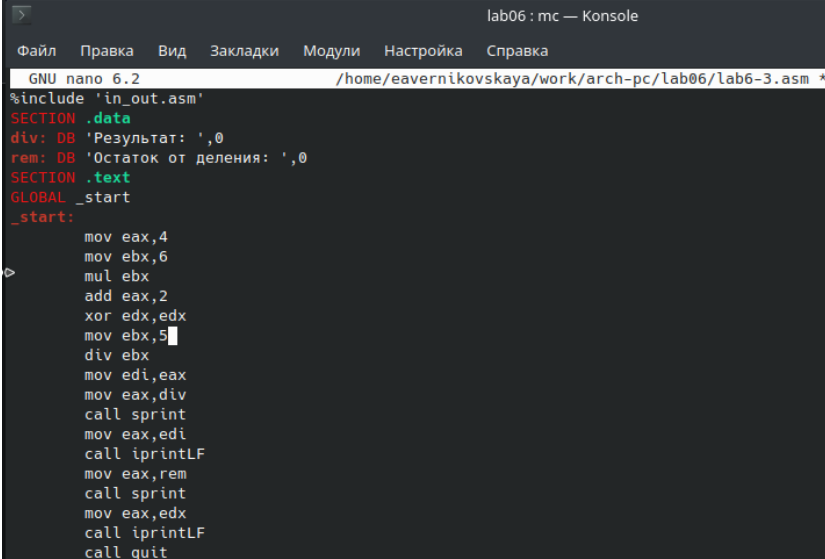
Изменённый текст программы:

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
```

```

mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit

```



```

lab06: mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
GNU nano 6.2  /home/eavernikovskaya/work/arch-pc/lab06/lab6-3.asm *
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
    mov eax,4
    mov ebx,6
    mul ebx
    add eax,2
    xor edx,edx
    mov ebx,5
    div ebx
    mov edi,eax
    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF
    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF
    call quit

```

Рис. 17: Изменение программы

Снова создаём исполняемый файл и запускаем его (рис. [-@fig:018])

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

Рис. 18: Исполняемый файл + запуск

Создаём файл «variant.asm» (рис. [-@fig:019])

```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ touch variant.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2 lab6-2.asm lab6-2.o lab6-3 lab6-3.asm lab6-3.o variant.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 19: Создание файла «variant.asm»

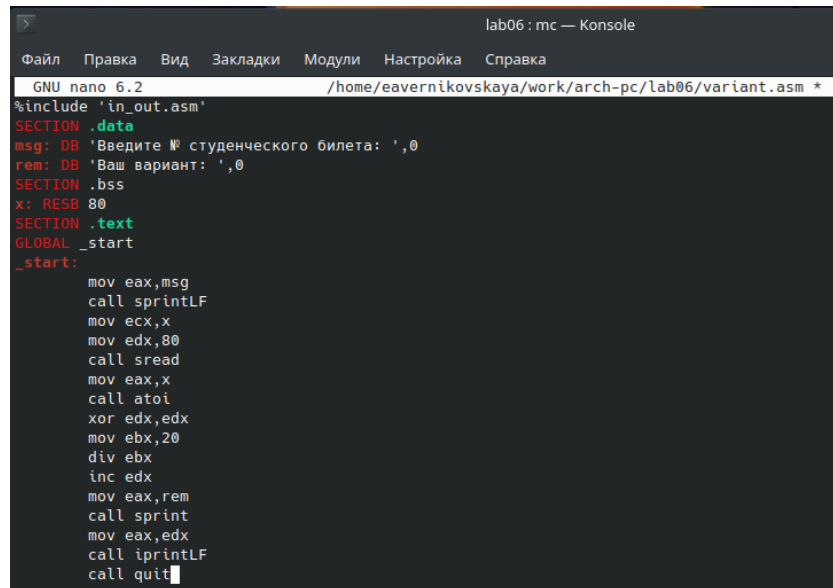
Вводим текст программы вычисления варианта задания по номер студенческого билета, которая будет работать по следующему алгоритму:

- вывести запрос на введение № студенческого билета;
- вычислить номер варианта по формуле: $(S_n \bmod 20) + 1$, где S_n – номер студенческого билета (В данном случае $a \bmod b$ – это остаток от деления a на b);
- вывести на экран номер варианта; (рис. [-@fig:020])

Текст программы:

```
%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
```

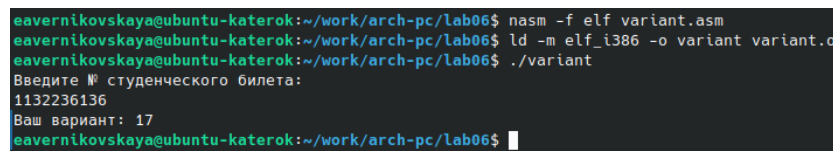
```
_start:
    mov eax,msg
    call sprintLF
    mov ecx,x
    mov edx,80
    call sread
    mov eax,x
    call atoi
    xor edx,edx
    mov ebx,20
    div ebx
    inc edx
    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF
    call quit
```



```
GNU nano 6.2 /home/eavernikovskaya/work/arch-pc/lab06/variant.asm *
#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
    call atoi
    xor edx,edx
    mov ebx,20
    div ebx
    inc edx
    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF
    call quit
```

Рис. 20: Ввод текста программы

Создаём исполняемый файл и запускаем его (рис. [-@fig:021])



```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ nasm -f elf variant.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132236136
Ваш вариант: 17
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 21: Исполняемый файл + запуск

ОТВЕТЫ НА ВОПРОСЫ:

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

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

2. Для чего используется следующие инструкции? “mov ecx,x“, “mov edx,80“, “call sread“

“mov ecx,x” - запись адреса пересенной в ecx “mov edx,80” - запись длины вводимого значения в edx “call sread” - вызов подпрограммы ввода сообщения

3. Для чего используется инструкция “call atoi”?

Вызывает функцию, которая преобразует ascii-код символа в целое числои записывает результат в регистр eax. Перед вызовом atoi в регистр eax нужно записать число.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

```
mov eax,x  
call atoi  
xor edx,edx  
mov ebx,20  
div ebx  
inc edx
```

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

При выполнении этой инструкции остаток от деления будет записываться в регистр АН.

6. Для чего используется инструкция “inc edx”?

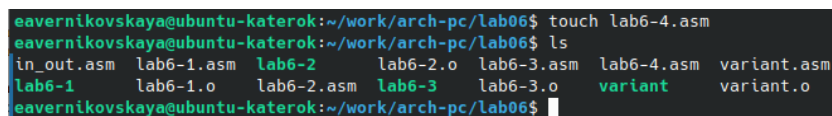
Увеличивает значение регистра edx на 1.

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

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

Задание для самостоятельной работы

Создаём файл «lab6-4.asm» (рис. [-@fig:022])



```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ touch lab6-4.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  lab6-4.asm  variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o    variant     variant.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 22: Создание файла «lab6-4.asm»

Вводим текст программы для вычисления выражения $y = f(x)$. В предыдущем задании мы получили № нашего варианта - 17. Значит пишем программу вычисления выражения $y = 18(x + 1)/6$ (рис. [-@fig:023])

```
%include 'in_out.asm'
SECTION .data
fx: DB 'y=18(x + 1)/6',0
msg: DB 'Введите x: ',0
rez: DB 'Ответ: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,fx
    call sprintLF
    mov eax,msg
    call sprint
    mov ecx,x
    mov edx,80
    call sread
    mov eax,x
```

```

call atoi
add eax,1
mov ebx,18
mul ebx
mov ecx,6
div ecx
mov edi,eax
mov eax,rez
call sprint
mov eax,edi
call iprintLF
call quit

```

```

lab06 : mc — Konsole
Файл  Правка  Вид  Закладки  Модули  Настройка  Справка
GNU nano 6.2 /home/eavernikovskaya/work/arch-pc/lab06/lab6-4.asm
%include 'in_out.asm'
SECTION .data
fx: DB 'y=18(x + 1)/6',0
msg: DB 'Введите x: ',0
rez: DB 'Ответ: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,fx
    call sprintLF
    mov eax,msg
    call sprint
    mov ecx,x
    mov edx,80
    call sread
    mov eax,x
    call atoi
    add eax,1
    mov ebx,18
    mul ebx
    mov ecx,6
    div ecx
    mov edi,eax
    mov eax,rez
    call sprint
    mov eax,edi
    call iprintLF
    call quit

```

Рис. 23: Ввод программы

Создаём исполняемый файл и запускаем его. Проверяем работу программы при введённых значениях $x1 = 3$ и $x2 = 1$ (рис. [-@fig:024])

```
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ./lab6-4
y=18(x + 1)/6
Введите x: 3
Ответ: 12
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$ ./lab6-4
y=18(x + 1)/6
Введите x: 1
Ответ: 6
eavernikovskaya@ubuntu-katerok:~/work/arch-pc/lab06$
```

Рис. 24: Исполняемый файл + запуск

Выводы

В ходе выполнения лабораторной работы мы освоили арифметические инструкции языка ассемблера NASM.