

# **Лабораторная работа №6**

**Арифметические операции в NASM**

Турсунбоев Сардорбек

# 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>22</b>

## Список иллюстраций

3.1	Код программы lab6-1.asm . . . . .	8
3.2	Тестирование программы lab6-1.asm . . . . .	8
3.3	Код программы lab6-1.asm . . . . .	9
3.4	Тестирование программы lab6-1.asm . . . . .	9
3.5	Код программы lab6-2.asm . . . . .	10
3.6	Тестирование программы lab6-2.asm . . . . .	10
3.7	Код программы lab6-2.asm . . . . .	11
3.8	Тестирование программы lab6-2.asm . . . . .	11
3.9	Тестирование программы lab6-2.asm . . . . .	12
3.10	Код программы lab6-3.asm . . . . .	13
3.11	Тестирование программы lab6-3.asm . . . . .	14
3.12	Код программы lab6-3.asm . . . . .	15
3.13	Тестирование программы lab6-3.asm . . . . .	16
3.14	Код программы variant.asm . . . . .	17
3.15	Тестирование программы variant.asm . . . . .	18
3.16	Код программы task.asm . . . . .	20
3.17	Тестирование программы task.asm . . . . .	21

## Список таблиц

# 1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

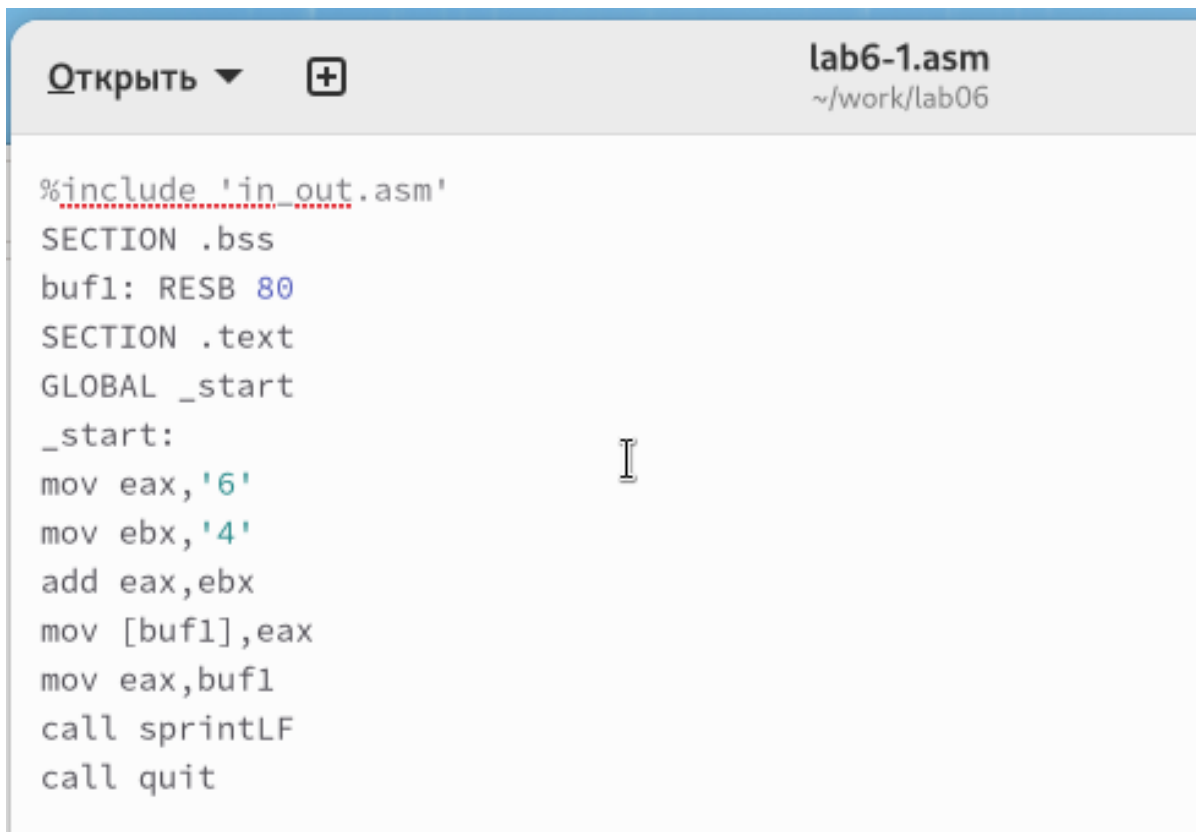
1. Изучить команды арифметических операций
2. Рассмотреть пример программы
3. Выполнить самостоятельное задание

### 3 Выполнение лабораторной работы

Я создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.

Давайте рассмотрим примеры программ, которые выводят символьные и числовые значения. В этих программах значения будут записываться в регистр еах.

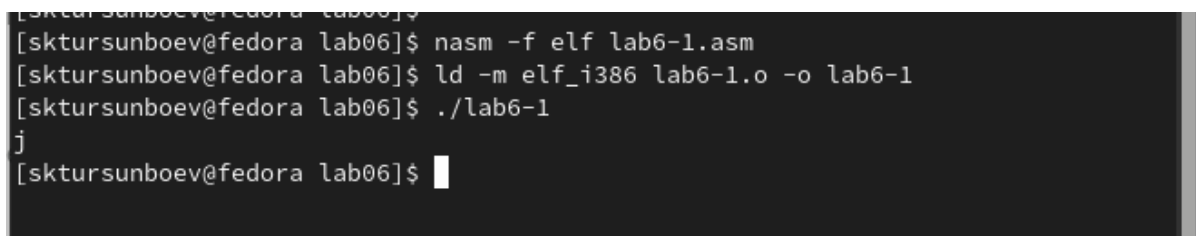
В данной программе мы записываем символ '6' в регистр еах (`mov еах, '6'`), а символ '4' в регистр ебх (`mov ебх, '4'`). Затем мы добавляем значение регистра ебх к значению в регистре еах (`add еах, ебх`, результат сложения записывается в регистр еах). После этого мы выводим результат. Однако, для работы функции `sprintf`, необходимо, чтобы в регистре еах был записан адрес, поэтому мы используем дополнительную переменную. Мы записываем значение регистра еах в переменную `buf1` (`mov [buf1], еах`), а затем записываем адрес переменной `buf1` в регистр еах (`mov еах, buf1`) и вызываем функцию `sprintf`.



```
Открыть ▾ + lab6-1.asm
~/work/lab06

%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. 3.1: Код программы lab6-1.asm



```
[sktursunboev@fedora lab06]$
[sktursunboev@fedora lab06]$ nasm -f elf lab6-1.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[sktursunboev@fedora lab06]$ ./lab6-1
j
[sktursunboev@fedora lab06]$
```

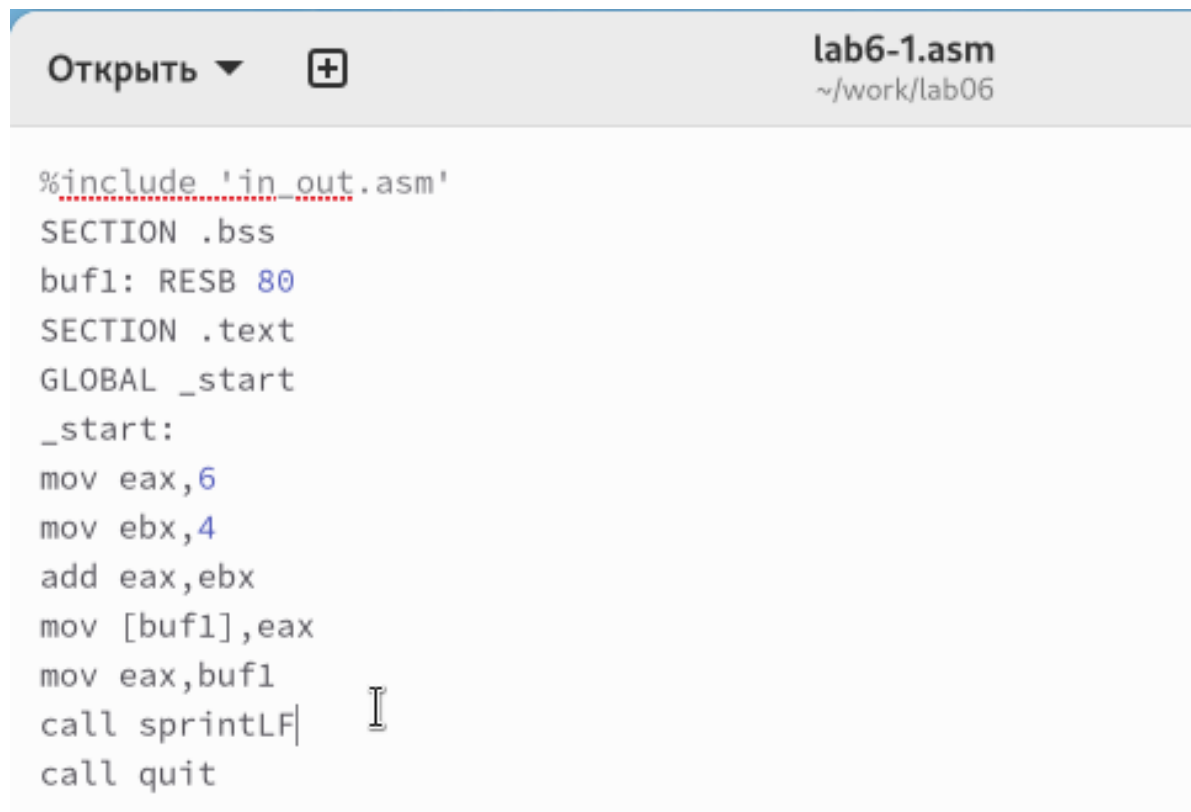
Рис. 3.2: Тестирование программы lab6-1.asm

В данном случае, когда мы ожидаем увидеть число 10 при выводе значения регистра `eax`, фактическим результатом будет символ 'j'. Это происходит из-за того, что код символа '6' равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа '4' равен 00110100 (или 52 в десятичном представлении). Когда мы выполняем команду `add eax, ebx`, результатом



будет сумма кодов - 01101010 (или 106 в десятичном представлении), который соответствует символу 'j'.

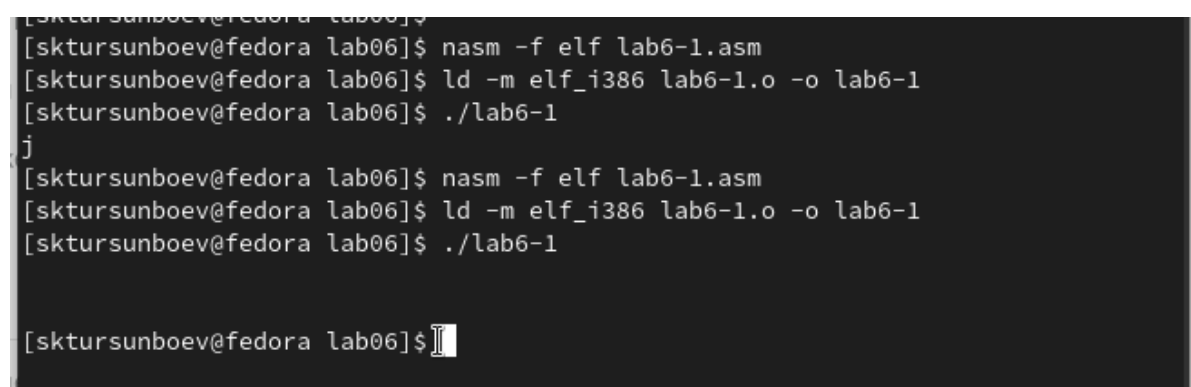
Далее изменяю текст программы и вместо символов, запишем в регистры числа.



```
lab6-1.asm
~/work/lab06

%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. 3.3: Код программы lab6-1.asm

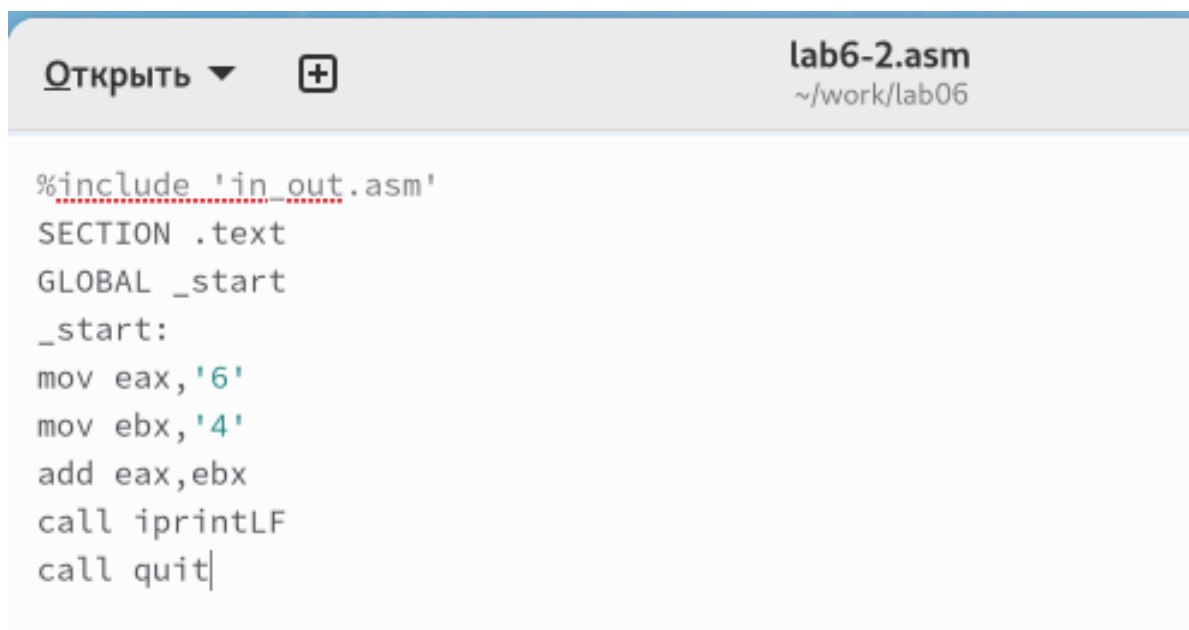


```
[sktursunboev@fedora lab06]$ nasm -f elf lab6-1.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[sktursunboev@fedora lab06]$ ./lab6-1
j
[sktursunboev@fedora lab06]$ nasm -f elf lab6-1.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[sktursunboev@fedora lab06]$ ./lab6-1
j
[sktursunboev@fedora lab06]$
```

Рис. 3.4: Тестирование программы lab6-1.asm

При изменении текста программы и записи чисел в регистры, мы также не получим число 10 при выполнении программы. Вместо этого будет выведен символ с кодом 10, который представляет собой символ конца строки (возврат каретки). В консоли этот символ не отображается, но он добавляет пустую строку.

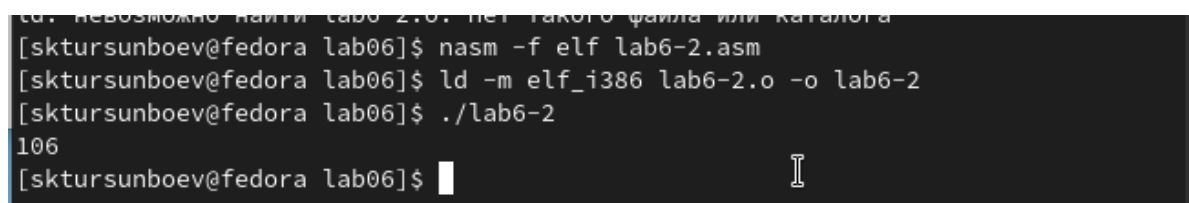
Как уже было отмечено ранее, в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Я преобразовал текст программы, используя эти функции.



```
lab6-2.asm
~/work/lab06

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

Рис. 3.5: Код программы lab6-2.asm



```
cd: невозможно найти lab6-2.0: нет такого файла или каталога
[sktursunboev@fedora lab06]$ nasm -f elf lab6-2.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[sktursunboev@fedora lab06]$ ./lab6-2
106
[sktursunboev@fedora lab06]$
```

Рис. 3.6: Тестирование программы lab6-2.asm

В результате выполнения программы мы получим число 106. В данном случае, как и в первом случае, команда `add` складывает коды символов '6' и '4' ( $54+52=106$ ).

Однако, в отличие от предыдущей программы, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа.

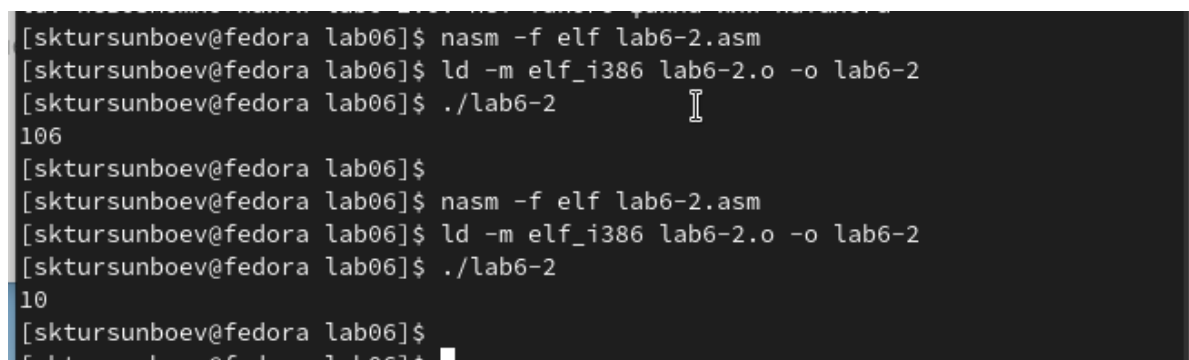


```
Открыть ▼ + lab6-2.asm
~/work/lab06

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

Рис. 3.7: Код программы lab6-2.asm

Функция `iprintLF` позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.



```
[sktursunboev@fedora lab06]$ nasm -f elf lab6-2.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[sktursunboev@fedora lab06]$ ./lab6-2
106
[sktursunboev@fedora lab06]$
[sktursunboev@fedora lab06]$ nasm -f elf lab6-2.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[sktursunboev@fedora lab06]$ ./lab6-2
10
[sktursunboev@fedora lab06]$
```

Рис. 3.8: Тестирование программы lab6-2.asm

Заменяю функцию `iPrintLF` на `iPrint`. Создаю исполняемый файл и запускаю его. Вывод отличается тем, что нет переноса строки.

```
[sktursunboev@fedora lab06]$ nasm -f elf lab6-2.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[sktursunboev@fedora lab06]$ ./lab6-2
106
[sktursunboev@fedora lab06]$
[sktursunboev@fedora lab06]$ nasm -f elf lab6-2.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[sktursunboev@fedora lab06]$ ./lab6-2
10
[sktursunboev@fedora lab06]$
[sktursunboev@fedora lab06]$ nasm -f elf lab6-2.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[sktursunboev@fedora lab06]$ ./lab6-2
10[sktursunboev@fedora lab06]$
[sktursunboev@fedora lab06]$
```

Рис. 3.9: Тестирование программы `lab6-2.asm`

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения  $f(x) = (5 * 2 + 3) / 3$ .

Открыть ▾

+

lab6-3.asm  
~/work/lab06

```
%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. 3.10: Код программы lab6-3.asm

```
[sktursunboev@fedora lab06]$  
[sktursunboev@fedora lab06]$ touch lab6-3.asm  
[sktursunboev@fedora lab06]$ nasm -f elf lab6-3.asm  
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3  
[sktursunboev@fedora lab06]$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
[sktursunboev@fedora lab06]$
```

Рис. 3.11: Тестирование программы lab6-3.asm

Изменил текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ .  
Создал исполняемый файл и проверил его работу.

Открыть ▾

+

lab6-3.asm  
~/work/lab06

```
%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. 3.12: Код программы lab6-3.asm

```
[sktursunboev@fedora lab06]$ touch lab6-3.asm
[sktursunboev@fedora lab06]$ nasm -f elf lab6-3.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[sktursunboev@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
[sktursunboev@fedora lab06]$ nasm -f elf lab6-3.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[sktursunboev@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[sktursunboev@fedora lab06]$
```

Рис. 3.13: Тестирование программы lab6-3.asm

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета.

В данном случае число, над которым нужно выполнить арифметические операции, вводится с клавиатуры. Как было отмечено ранее, ввод с клавиатуры осуществляется в символьном виде, и для правильной работы арифметических операций в NASM символы должны быть преобразованы в числа. Для этой цели можно использовать функцию `atoi` из файла `in_out.asm`.



Открыть ▾

variant.asm  
~/work/lab06

```
%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
```

Рис. 3.14: Код программы variant.asm

```
[sktursunboev@fedora lab06]$ touch variant.asm
[sktursunboev@fedora lab06]$ nasm -f elf variant.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 variant.o -o variant
[sktursunboev@fedora lab06]$ ./variant
Введите № студенческого билета:
1032234223
Ваш вариант: 4
[sktursunboev@fedora lab06]$
```

Рис. 3.15: Тестирование программы variant.asm

ответы на вопросы

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

Строка “mov eax, rem” перекладывает значение переменной с фразой ‘Ваш вариант:’ в регистр eax.

Строка “call sprint” вызывает подпрограмму для вывода строки.

2. Для чего используются следующие инструкции?

Инструкция “mov ecx, x” используется для перемещения значения переменной x в регистр ecx.

Инструкция “mov edx, 80” используется для перемещения значения 80 в регистр edx.

Инструкция “call sread” вызывает подпрограмму для считывания значения студенческого билета из консоли.sread

3. Для чего используется инструкция “call atoi”?

Инструкция “call atoi” используется для преобразования введенных символов в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

Строка “xor edx, edx” обнуляет регистр edx.

Строка “mov ebx, 20” записывает значение 20 в регистр ebx.

Строка “div ebx” выполняет деление номера студенческого билета на 20.

Строка “inc edx” увеличивает значение регистра edx на 1.

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

Остаток от деления записывается в регистр edx.

6. Для чего используется инструкция “inc edx”?

Инструкция “inc edx” используется для увеличения значения в регистре edx на 1, согласно формуле вычисления варианта.

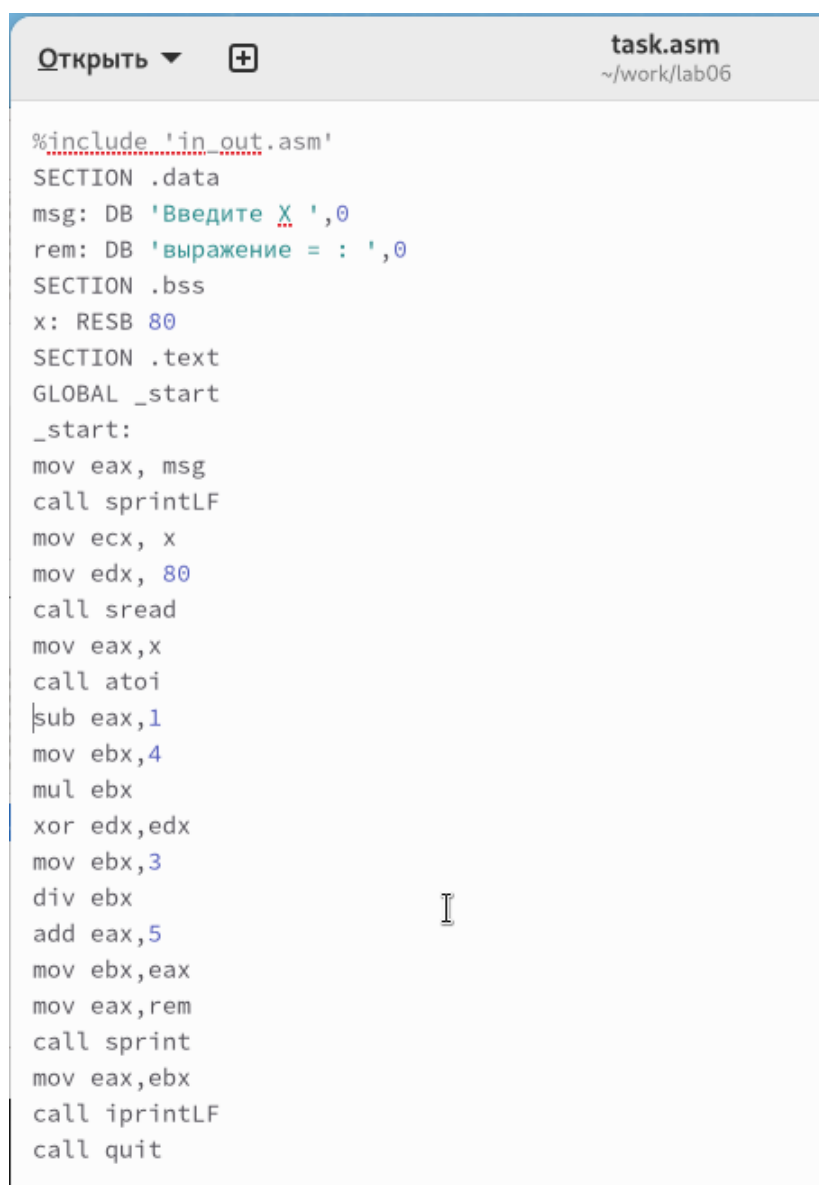
7. Какие строки листинга отвечают за вывод на экран результата вычислений?

Строка “mov eax, edx” перекладывает результат вычислений в регистр eax.

Строка “call iprintLF” вызывает подпрограмму для вывода значения на экран.

Написать программу вычисления выражения  $y = f(x)$ . Программа должна вывести 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 таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $x_1$  и  $x_2$  из 6.3.

Получили вариант  $4 - (4/3) * (x - 1) + 5$  для  $x = 4, x = 10$



```
task.asm
~\work\lab06

%include 'in_out.asm'
SECTION .data
msg: DB 'Введите X ',0
rem: DB 'выражение = : ',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
sub eax,1
mov ebx,4
mul ebx
xor edx,edx
mov ebx,3
div ebx
add eax,5
mov ebx,eax
mov eax,rem
call sprint
mov eax,ebx
call iprintLF
call quit
```

Рис. 3.16: Код программы task.asm

Также размещаю код программы в отчете.

```
[sktursunboev@fedora lab06]$ touch task.asm
[sktursunboev@fedora lab06]$ nasm -f elf task.asm
[sktursunboev@fedora lab06]$ ld -m elf_i386 task.o -o task
[sktursunboev@fedora lab06]$ ./task
Введите X
4
выражение = : 9
[sktursunboev@fedora lab06]$ ./task
Введите X
10
выражение = : 17
[sktursunboev@fedora lab06]$
[sktursunboev@fedora lab06]$
```

Рис. 3.17: Тестирование программы task.asm

## **4 Выводы**

Изучили работу с арифметическими операциями.