

Отчёт по лабораторной работе №7

Уткина Алина Дмитриевна

Содержание

1	Цель работы	1
2	Выполнение лабораторной работы.....	1
2.1	Символьные строки и численные данные в NASM	1
2.2	Выполнение арифметических операций в NASM	4
2.3	Самостоятельная работа.....	7
3	Выводы	7

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

2.1 Символьные строки и численные данные в NASM

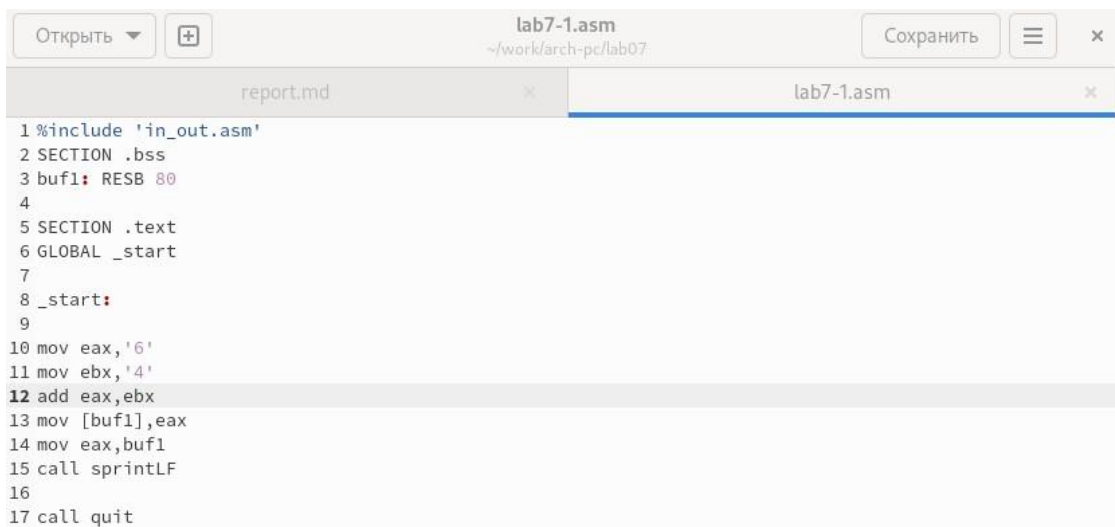
Создадим каталог для программ лабораторной работы №7, перейдем в него и создадим файл lab7-1.asm. (рис. 1)

```
[adutkina@fedora ~]$ mkdir ~/work/arch-pc/lab07
[adutkina@fedora ~]$ cd ~/work/arch-pc/lab07
[adutkina@fedora lab07]$ touch lab7-1.asm
[adutkina@fedora lab07]$ ls
lab7-1.asm
```

Рис. 1: Создание файла для выполнения работы

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр еах.

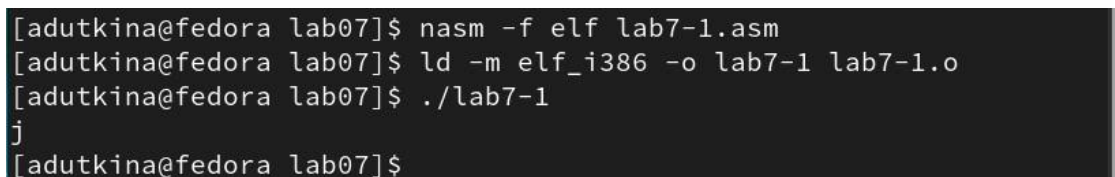
Введем в файл lab7-1.asm текст программы из листинга 7.1 (рис. 2). В данной программе в регистр еах записывается символ 6 (mov еах,'6'), в регистр еbх символ 4 (mov еbх,'4'). Далее к значению в регистре еах прибавляем значение регистра еbх (add еах,еbх, результат сложения запишется в регистр еах). Далее выводим результат. Так как для работы функции sprintLF в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную buf1 (mov [buf1],еах), а затем запишем адрес переменной buf1 в регистр еах (mov еах,buf1) и вызовем функцию sprintLF.



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4
5 SECTION .text
6 GLOBAL _start
7
8 _start:
9
10 mov eax,'6'
11 mov ebx,'4'
12 add eax,ebx
13 mov [buf1],eax
14 mov eax,buf1
15 call sprintf
16
17 call quit
```

Рис. 2: Программа вывода значения регистра *eax*

Создадим исполняемый файл и запустим его (рис. 3).

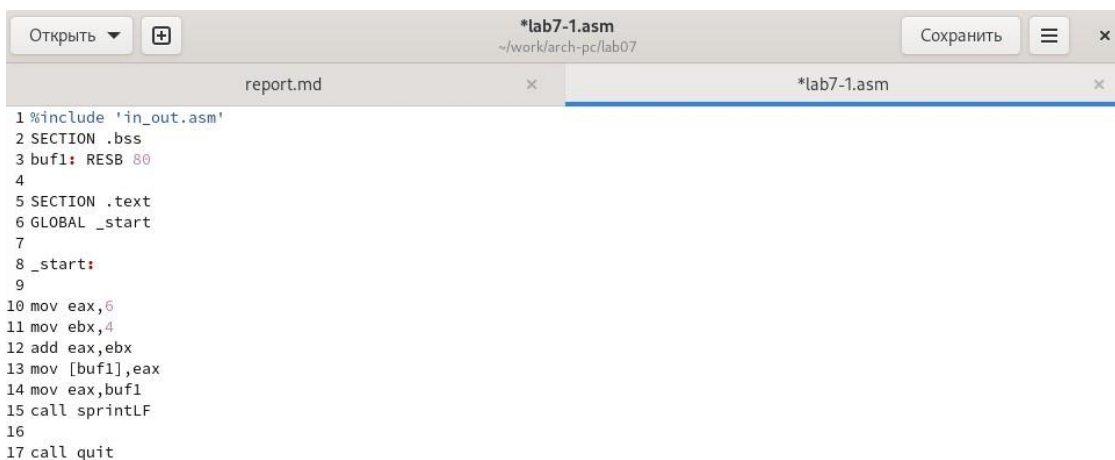


```
[adutkina@fedora lab07]$ nasm -f elf lab7-1.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[adutkina@fedora lab07]$ ./lab7-1
j
[adutkina@fedora lab07]$
```

Рис. 3: Создание исполняемого файла

В данном случае при выводе значения регистра *eax* мы ожидаем увидеть число 10. Однако результатом будет символ *j*. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда `add eax,ebx` запишет в регистр *eax* сумму кодов – 01101010 (106), что в свою очередь является кодом символа *j*.

Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы следующим образом (рис. 4):



```
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4
5 SECTION .text
6 GLOBAL _start
7
8 _start:
9
10 mov eax,6
11 mov ebx,4
12 add eax,ebx
13 mov [buf1],eax
14 mov eax,buf1
15 call sprintf
16
17 call quit
```

Рис. 4: Измененный текст программы

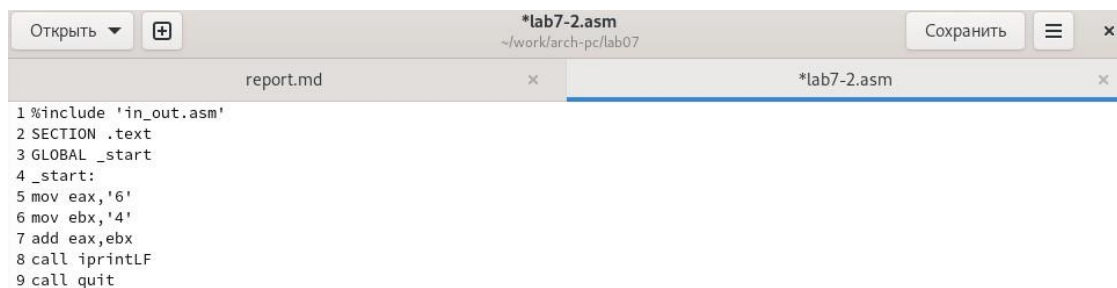
Как и в предыдущем случае мы не получаем число 10. В данном случае выводится символ с кодом 10. По таблице ASCII можно определить, что код 10 соответствует символу `"`. При запуске программы видно, что на экран выводится только пустая строка (рис. 5).

```
[adutkina@fedora lab07]$ nasm -f elf lab7-1.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[adutkina@fedora lab07]$ ./lab7-1

[adutkina@fedora lab07]$
```

Рис. 5: Результат измененной программы

Для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 7.1 с использованием этих функций. Создадим файл `lab7-2.asm` и введем в него текст программы из листинга 7.2 (рис. 6).



```
*lab7-2.asm
~/work/arch-pc/lab07
report.md x *lab7-2.asm x
1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit
```

Рис. 6: Использование функций в программе

В результате работы программы мы получили число 106. В данном случае, как и в первом, команда `add` складывает коды символов `'6'` и `'4'` ($54+52=106$). Однако, в отличие от программы из листинга 7.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число (рис. 7)

```
[adutkina@fedora lab07]$ touch lab7-2.asm
[adutkina@fedora lab07]$ gedit lab7-2.asm
[adutkina@fedora lab07]$ nasm -f elf lab7-2.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[adutkina@fedora lab07]$ ./lab7-2
106
[adutkina@fedora lab07]$
```

Рис. 7: Результат работы программы с функциями

Аналогично предыдущему примеру изменим символы на числа. При исполнении программы мы получим 10 (рис. 8).

```
[adutkina@fedora lab07]$ gedit lab7-2.asm
[adutkina@fedora lab07]$ nasm -f elf lab7-2.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[adutkina@fedora lab07]$ ./lab7-2
10
[adutkina@fedora lab07]$
```

Рис. 8: Результат измененной программы

Заменяем функцию `iprintLF` на `iprint`. Создадим исполняемый файл и запустим его. Чем отличается вывод функций `iprintLF` и `iprint`? В этом случае после вывода результата нет отступа строки (рис. 9)

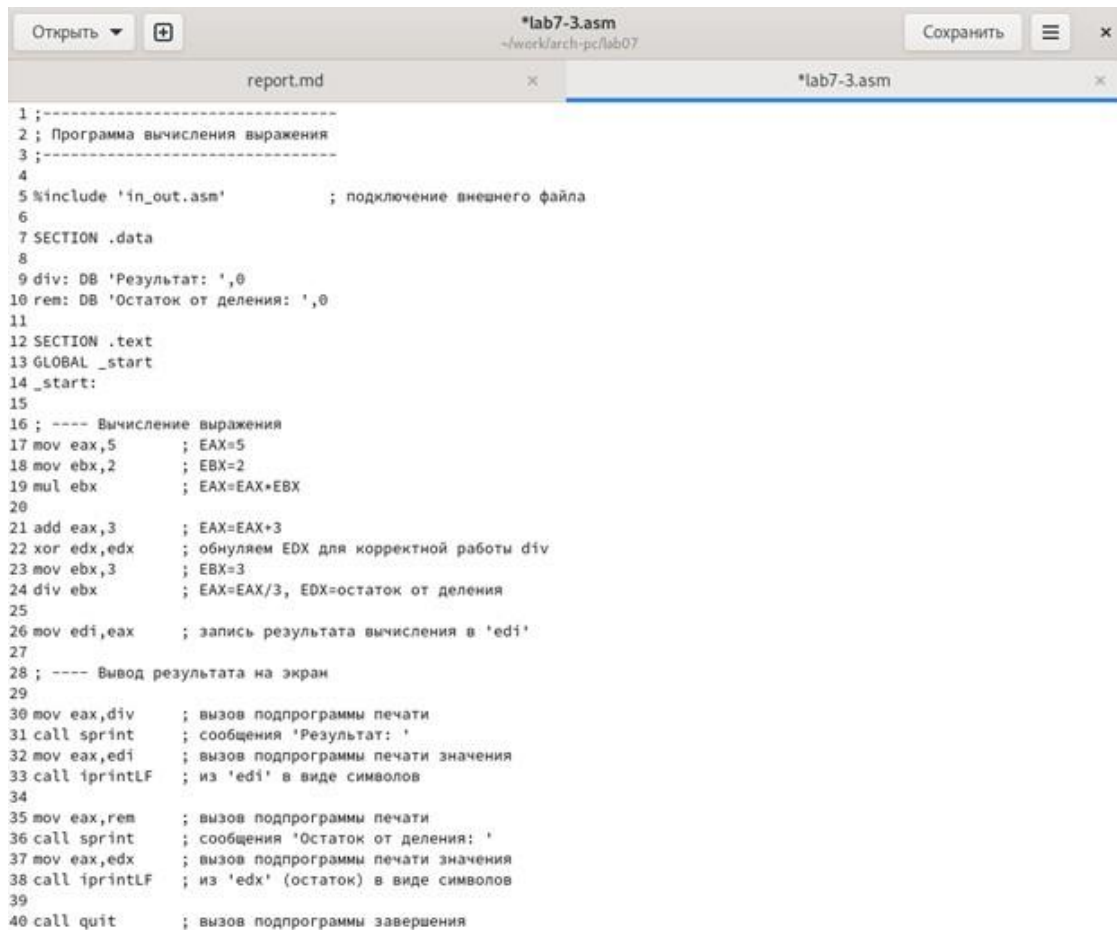
```
[adutkina@fedora lab07]$ nasm -f elf lab7-2.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[adutkina@fedora lab07]$ ./lab7-2
10[adutkina@fedora lab07]$
[adutkina@fedora lab07]$
```

Рис. 9: Изменение функции в программе

2.2 Выполнение арифметических операций в NASM

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

Создадим файл `lab7-3.asm` и введем в него текст из листинга 7.3 (рис. 10).



```
*lab7-3.asm
~/work/arch-pc/lab07
Сохранить

report.md x *lab7-3.asm x
1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4
5 %include 'in_out.asm' ; подключение внешнего файла
6
7 SECTION .data
8
9 div: DB 'Результат: ',0
10 rem: DB 'Остаток от деления: ',0
11
12 SECTION .text
13 GLOBAL _start
14 _start:
15
16 ; ---- Вычисление выражения
17 mov eax,5 ; EAX=5
18 mov ebx,2 ; EBX=2
19 mul ebx ; EAX=EAX*EBX
20
21 add eax,3 ; EAX=EAX+3
22 xor edx,edx ; обнуляем EDX для корректной работы div
23 mov ebx,3 ; EBX=3
24 div ebx ; EAX=EAX/3, EDX=остаток от деления
25
26 mov edi,eax ; запись результата вычисления в 'edi'
27
28 ; ---- Вывод результата на экран
29
30 mov eax,div ; вызов подпрограммы печати
31 call sprint ; сообщения 'Результат: '
32 mov eax,edi ; вызов подпрограммы печати значения
33 call iprintLF ; из 'edi' в виде символов
34
35 mov eax,rem ; вызов подпрограммы печати
36 call sprint ; сообщения 'Остаток от деления: '
37 mov eax,edx ; вызов подпрограммы печати значения
38 call iprintLF ; из 'edx' (остаток) в виде символов
39
40 call quit ; вызов подпрограммы завершения
```

Рис. 10: Программа вычисления выражения

При запуске программы выводятся результат и остаток от деления (рис. 11).

```
[adutkina@fedora lab07]$ touch lab7-3.asm
[adutkina@fedora lab07]$ gedit lab7-3.asm
[adutkina@fedora lab07]$ nasm -f elf lab7-3.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[adutkina@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[adutkina@fedora lab07]$
```

Рис. 11: Результат вычисления выражения

Изменим текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$ (рис. 12). Запускаем исполняемый файл (рис. 13).

```
16 ; ---- Вычисление выражения
17 mov eax,4      ; EAX=4
18 mov ebx,6      ; EBX=6
19 mul ebx        ; EAX=EAX*EBX
20
21 add eax,2      ; EAX=EAX+2
22 xor edx,edx    ; обнуляем EDX для корректной работы div
23 mov ebx,5      ; EBX=5
24 div ebx        ; EAX=EAX/5, EDX=остаток от деления
~r
```

Рис. 12: Изменение значений в программе

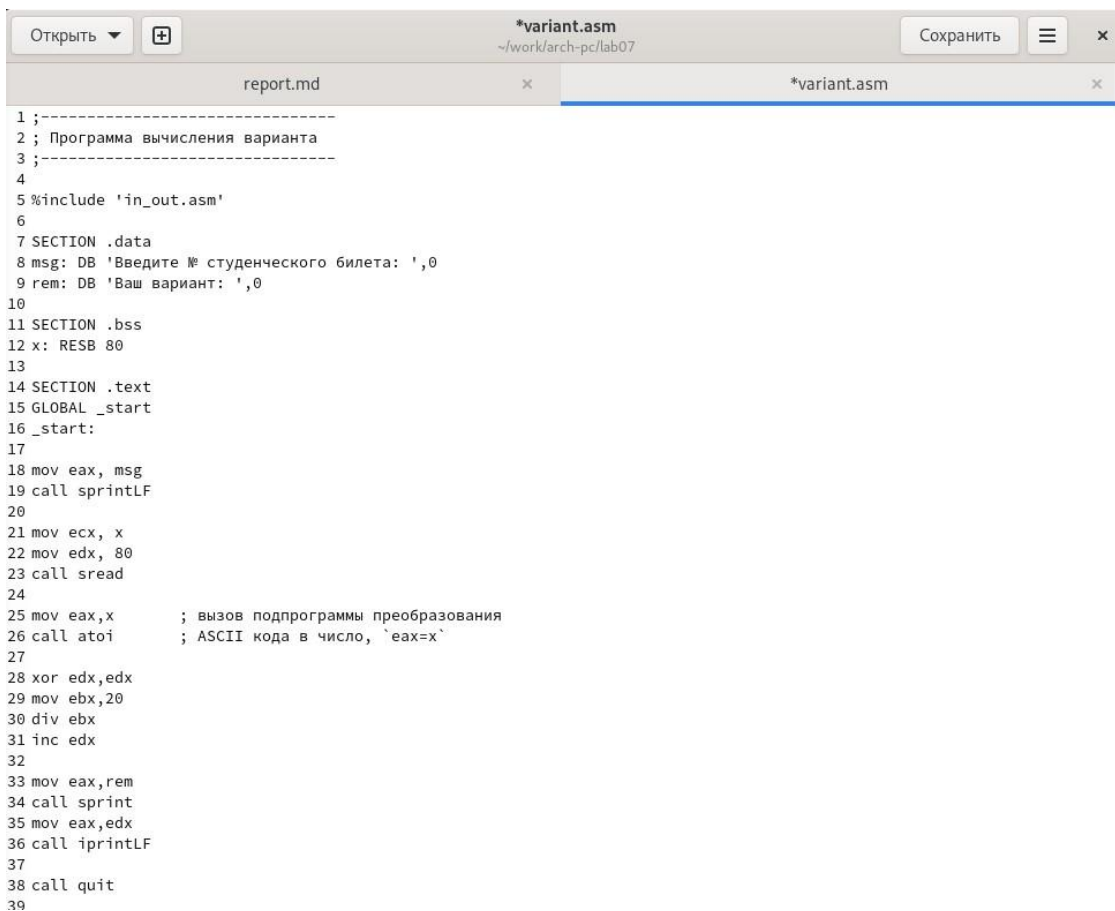
```
[adutkina@fedora lab07]$ nasm -f elf lab7-3.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[adutkina@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[adutkina@fedora lab07]$
```

Рис. 13: Проверка правильности работы программы

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: - вывести запрос на введение № студенческого билета - вычислить номер варианта по формуле: $(Sn \bmod 20) + 1$, где Sn – номер студенческого билета (В данном случае $a \bmod b$ – это остаток от деления a на b). - вывести на экран номер варианта.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция `atoi` из файла `in_out.asm`.

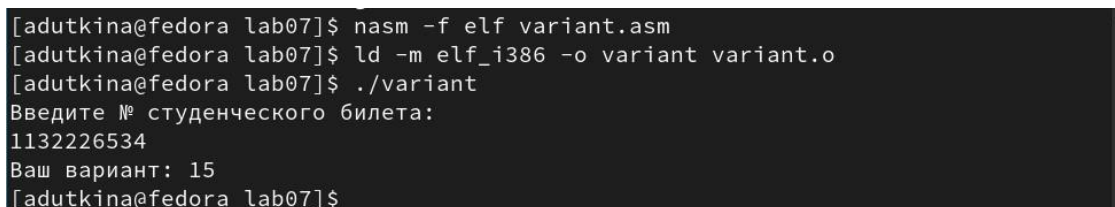
Создадим файл `variant.asm` и введем в него текст из листинга 7.4 (рис. 14).



```
1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4
5 %include 'in_out.asm'
6
7 SECTION .data
8 msg: DB 'Введите № студенческого билета: ',0
9 rem: DB 'Ваш вариант: ',0
10
11 SECTION .bss
12 x: RESB 80
13
14 SECTION .text
15 GLOBAL _start
16 _start:
17
18 mov eax, msg
19 call sprintf
20
21 mov ecx, x
22 mov edx, 80
23 call sread
24
25 mov eax,x          ; вызов подпрограммы преобразования
26 call atoi          ; ASCII кода в число, `eax=x`
27
28 xor edx,edx
29 mov ebx,20
30 div ebx
31 inc edx
32
33 mov eax,rem
34 call sprintf
35 mov eax,edx
36 call iprintLF
37
38 call quit
39
```

Рис. 14: Вычисление варианта задания по номеру студенческого билета

Создадим исполняемый файл и запустим его (рис. 15). Вычислив номер варианта аналитически мы получаем, что $1132226534 \% 20 = 14$, $14 + 1 = 15$. Значит, программа работает правильно.



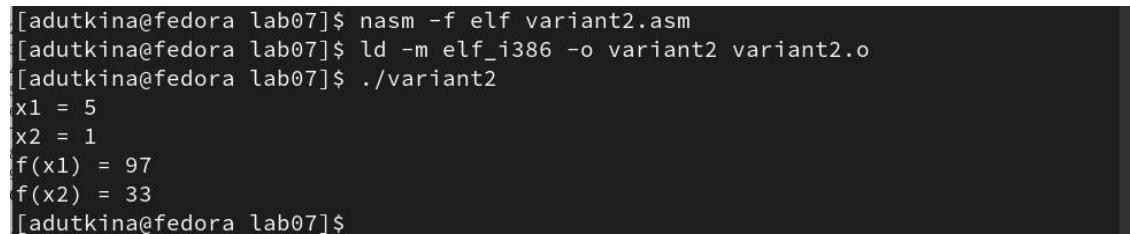
```
[adutkina@fedora lab07]$ nasm -f elf variant.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[adutkina@fedora lab07]$ ./variant
Введите № студенческого билета:
1132226534
Ваш вариант: 15
[adutkina@fedora lab07]$
```

Рис. 15: Запуск программы

Таким образом: 1. За вывод на экран сообщения “Ваш вариант” отвечают строки 33-34 kbsnbyuf 7.4; 2. Конструкция `mov ecx,x` отвечает за запись значения `x` в регистр `ecx`; `mov edx, 80` определяет длину вводимой строки; `call sread` вызывает подпрограмму ввода сообщения; 3. `call atoi` используется для преобразования ASCII кода в число; 4. За вычисление варианта отвечают строки 28-31 листинга 7.4; 5. Остаток от деления при выполнении инструкции “`div ebx`” запишется в регистр “`edx`”; 6. Инструкция “`inc edx`” используется для уменьшения значения в регистре “`edx`”; 7. За вывод на экран результата вычислений отвечают строки 35-36.

2.3 Самостоятельная работа

Напишем программу для работы с функцией $f(x) = (5+x)^2 - 3$ в файл variant2.asm. Проверим работу программы для значений $x_1=5$, $x_2=1$ (рис. 16).



```
[adutkina@fedora lab07]$ nasm -f elf variant2.asm
[adutkina@fedora lab07]$ ld -m elf_i386 -o variant2 variant2.o
[adutkina@fedora lab07]$ ./variant2
x1 = 5
x2 = 1
f(x1) = 97
f(x2) = 33
[adutkina@fedora lab07]$
```

Рис. 16: Работа программы вычисления значения функции

3 Выводы

В ходе лабораторной работы были изучены арифметические инструкции языка ассемблера NASM.