

Отчёт по лабораторной работе № 7

Архитектура компьютера

Старцева Алина Сергеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Порядок выполнения лабораторной работы	6
3.1	Символьные и численные данные в NASM	6
3.2	Выполнение арифметических операций в NASM	11
3.3	Задание для самостоятельной работы	15
4	Выводы	17

Список иллюстраций

3.1	Файл lab7-1.asm	6
3.2	Текст программы из листинга 7.1	7
3.3	Исполняемый файл	7
3.4	Текст программы	8
3.5	Исполняемый файл	8
3.6	файл lab7-2.asm	8
3.7	Текст программы из листинга 7.2	9
3.8	Исполняемый файл	9
3.9	Изменённый код	10
3.10	Результат программы	10
3.11	iprintLF на iprint	10
3.12	Работа исполняемого файла	10
3.13	Файл lab7-3.asm	11
3.14	Текст программы из листинга 7.3	11
3.15	Исполняемый файл	12
3.16	Изменённый текст программы	12
3.17	Исполняемый файл	12
3.18	Файл variant.asm	13
3.19	Текст программы из листинга 7.4	14
3.20	Работа исполняемого файла	14
3.21	Файл variant.asm	15
3.22	Работа исполняемого файла	16

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. выполнить работу с символьными и численными данными в NASM
2. Отработать на практике арифметические операции в NASM
3. Написать программу вычисления выражения с входными данными

3 Порядок выполнения лабораторной работы

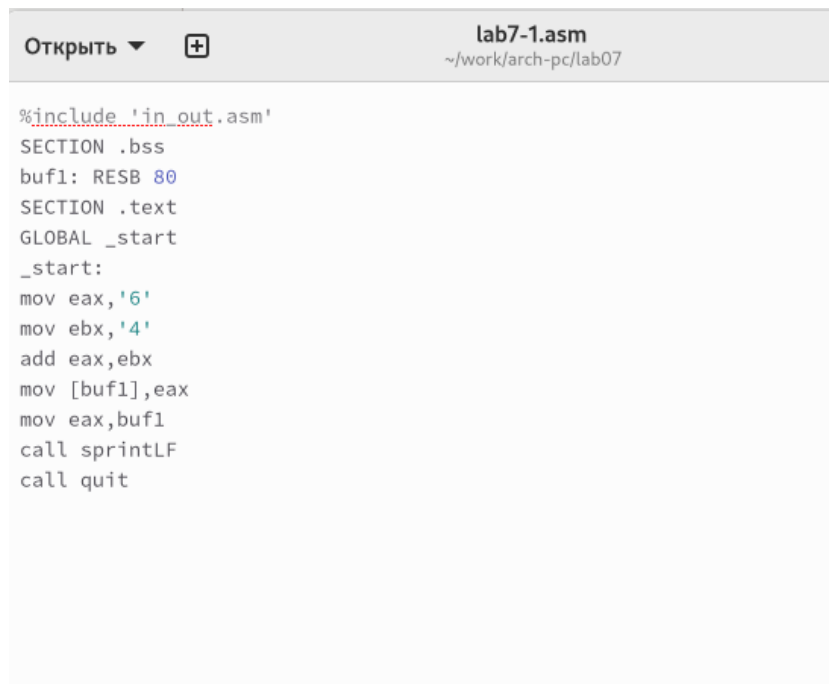
3.1 Символьные и численные данные в NASM

1. Создали каталог для программ лабораторной работы № 7, перешли в него и создайте файл lab7-1.asm. (рис. 3.1)

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

Рис. 3.1: Файл lab7-1.asm

2. Рассмотрели примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр еах. Ввели в файл lab7-1.asm текст программы из листинга 7.1. (рис. 3.2) 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 прибавляется значение регистра еbх (add еах,еbх, результат сложения запишется в регистр еах). Далее выводится результат. Так как для работы функции sprintLF в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого записали значение регистра еах в переменную buf1 (mov [buf1],еах), а затем записали адрес переменной buf1 в регистр еах (mov еах,buf1) и вызвали функцию sprintLF.

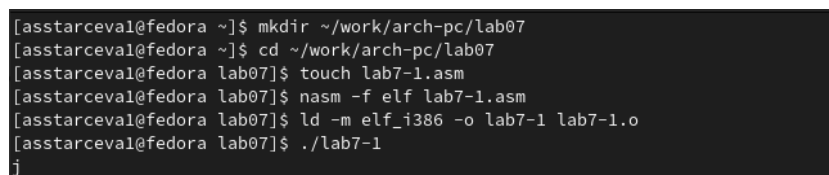


```
Открыть + lab7-1.asm
~/work/arch-pc/lab07

%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. 3.2: Текст программы из листинга 7.1

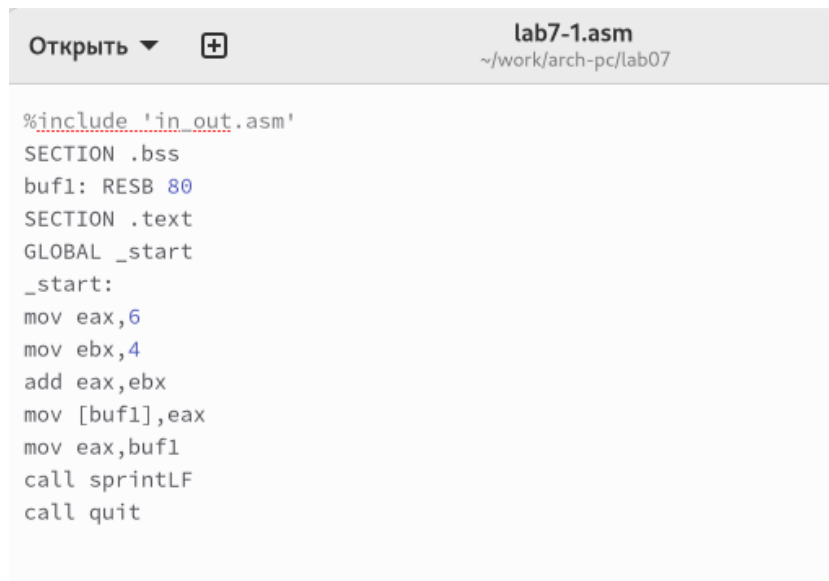
Создали исполняемый файл и запустили его. (рис. 3.3)



```
[asstarceval@fedora ~]$ mkdir ~/work/arch-pc/lab07
[asstarceval@fedora ~]$ cd ~/work/arch-pc/lab07
[asstarceval@fedora lab07]$ touch lab7-1.asm
[asstarceval@fedora lab07]$ nasm -f elf lab7-1.asm
[asstarceval@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[asstarceval@fedora lab07]$ ./lab7-1
j
```

Рис. 3.3: Исполняемый файл

3. Далее изменили текст программы и вместо символов, записали в регистры числа. (рис. 3.4)

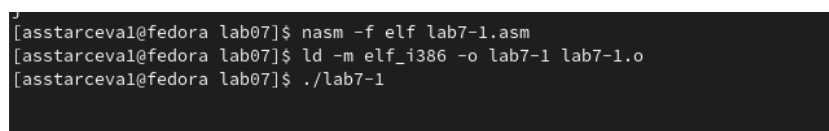


```
Открыть ▾ + lab7-1.asm
~/work/arch-pc/lab07

%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. 3.4: Текст программы

Создали исполняемый файл и запустили его. (рис. 3.5)



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

Рис. 3.5: Исполняемый файл

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определили, что код 10 соответствует символ /n. Это символ перевода строки, он не отображается.

4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовали текст программы из Листинга 7.1 с использованием этих функций. Создали файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. 3.6) и ввели в него текст программы из листинга 7.2. (рис. 3.7)



```
[asstarceval@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
```

Рис. 3.6: файл lab7-2.asm

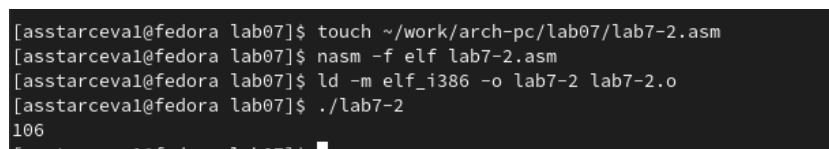


```
Открыть ▾ + lab7-2.asm
~/work/arch-pc/lab07

%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: Текст программы из листинга 7.2

Создали исполняемый файл и запустили его. (рис. 3.8)



```
[asstarceval@fedora lab07]$ touch ~/work/arch-pc/lab07/lab7-2.asm
[asstarceval@fedora lab07]$ nasm -f elf lab7-2.asm
[asstarceval@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[asstarceval@fedora lab07]$ ./lab7-2
106
[asstarceval@fedora lab07]$
```

Рис. 3.8: Исполняемый файл

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.

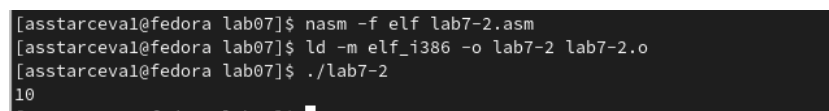
5. Аналогично предыдущему примеру изменим символы на числа. (рис. 3.9)



```
Открыть ▾ + lab7-2.asm  
~/work/arch-pc/lab07  
  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit
```

Рис. 3.9: Изменённый код

Создайте исполняемый файл и запустите его. В результате при исполнении программы получили 10. (рис. 3.10)



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

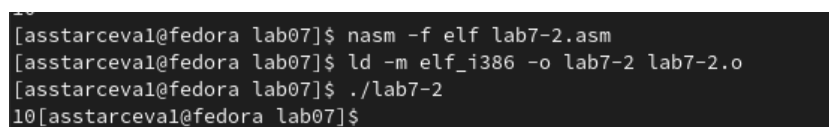
Рис. 3.10: Результат программы

Заменяли функцию iprintLF на iprint. (рис. 3.11) Создайте исполняемый файл и запустите его. (рис. 3.12) Вывод функций iprintLF и iprint отличается наличием перевода строки после вывода?



```
call iprint|
```

Рис. 3.11: iprintLF на iprint

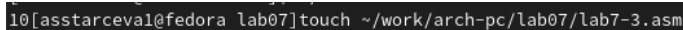


```
[asstarceval@fedora lab07]$ nasm -f elf lab7-2.asm  
[asstarceval@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o  
[asstarceval@fedora lab07]$ ./lab7-2  
10  
[asstarceval@fedora lab07]$
```

Рис. 3.12: Работа исполняемого файла

3.2 Выполнение арифметических операций в NASM

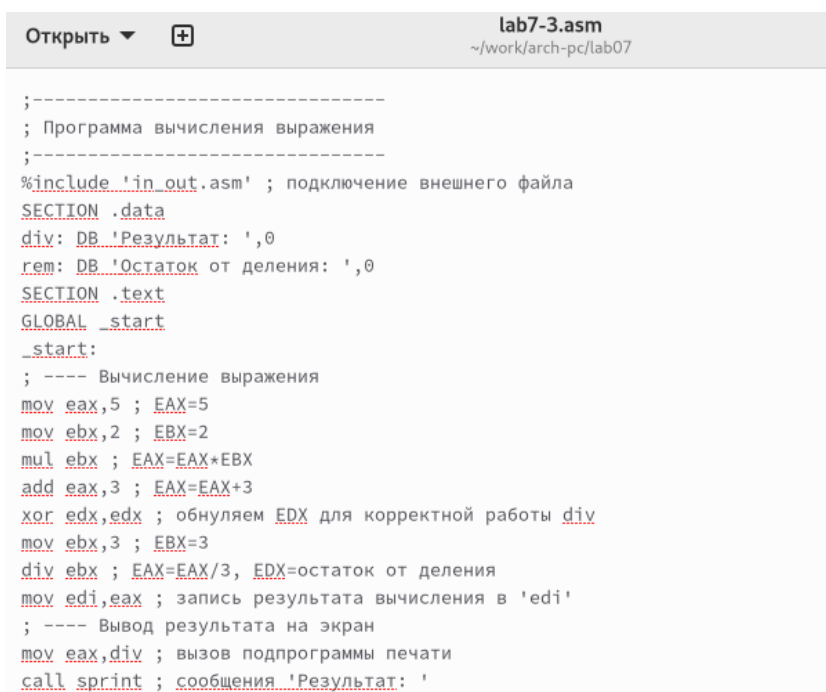
6. В качестве примера выполнения арифметических операций в NASM привели программу вычисления арифметического выражения $\boxtimes(\boxtimes) = (5 \boxtimes 2 + 3)/3$. Создали файл lab7-3.asm в каталоге ~/work/arch-pc/lab07. (рис. 3.13)



```
10[asstarceval@fedora lab07]touch ~/work/arch-pc/lab07/lab7-3.asm
```

Рис. 3.13: Файл lab7-3.asm

Внимательно изучили текст программы из листинга 7.3 и ввели в lab7-3.asm. (рис. 3.14)



```
Открыть ▾ + lab7-3.asm
~/work/arch-pc/lab07

;-----
; 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
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
```

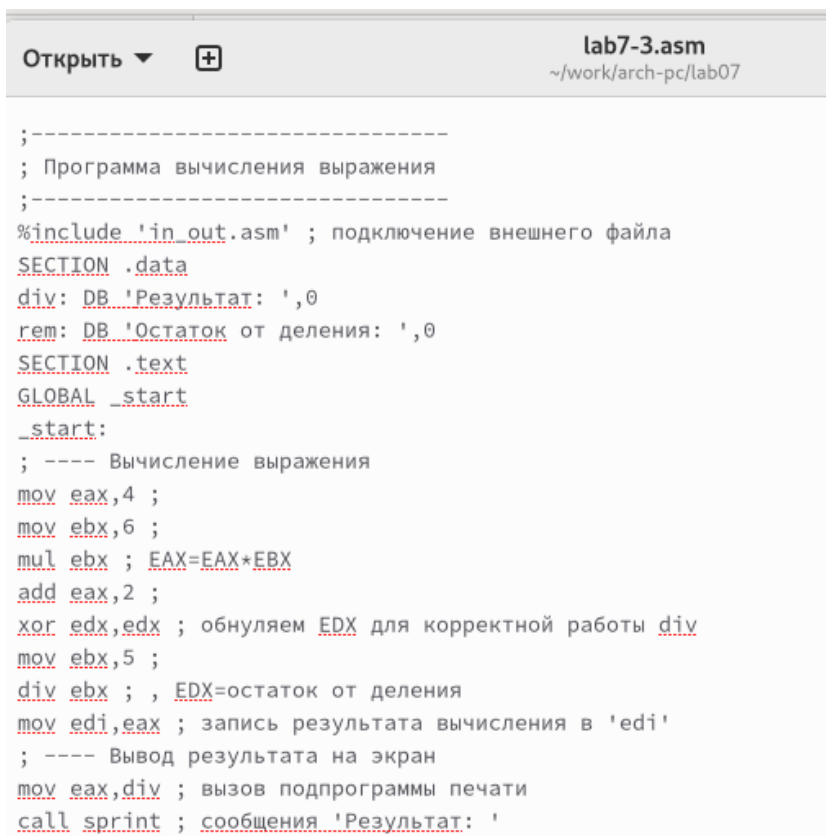
Рис. 3.14: Текст программы из листинга 7.3

Создали исполняемый файл и запустили его. Получили следующий результат. (рис. 3.15)

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

Рис. 3.15: Исполняемый файл

Изменили текст программы для вычисления выражения $\boxtimes(\boxtimes) = (4 \boxtimes 6 + 2)/5$.
(рис. 3.16)



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

;-----
; 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 ;
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 ; , EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
```

Рис. 3.16: Изменённый текст программы

Создали исполняемый файл и проверили его работу. (рис. 3.17)

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

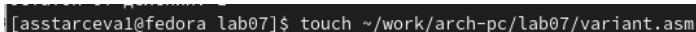
Рис. 3.17: Исполняемый файл

7. В качестве другого примера рассмотрели программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

- вывести запрос на введение № студенческого билета
- вычислить номер варианта по формуле: $(\text{№} \bmod 20) + 1$, где № – номер студенческого билета (В данном случае $\text{№} \bmod \text{№}$ – это остаток от деления № на №).
- 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 преобразуются в числа. Для этого использована функция `atoi` из файла `in_out.asm`.

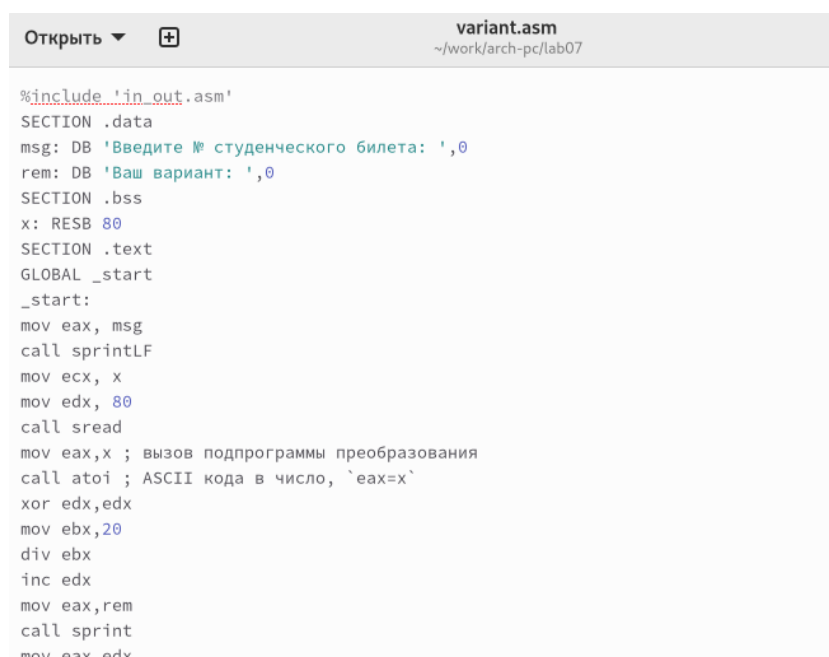
Создали файл `variant.asm` в каталоге `~/work/arch-pc/lab07`. (рис. 3.18)



```
[asstarceval@fedora lab07]$ touch ~/work/arch-pc/lab07/variant.asm
```

Рис. 3.18: Файл `variant.asm`

Внимательно изучили текст программы из листинга 7.4 и ввели в файл `variant.asm`. (рис. 3.19)

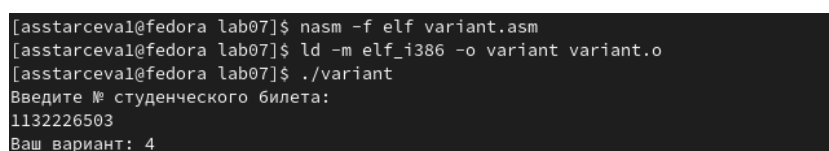


```
variant.asm
~/work/arch-pc/lab07

#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 ; 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
```

Рис. 3.19: Текст программы из листинга 7.4

Создали исполняемый файл и запустили его. (рис. 3.20) Проверили результат работы программы вычислив номер варианта аналитически.



```
[asstarceval@fedora lab07]$ nasm -f elf variant.asm
[asstarceval@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[asstarceval@fedora lab07]$ ./variant
Введите № студенческого билета:
1132226503
Ваш вариант: 4
```

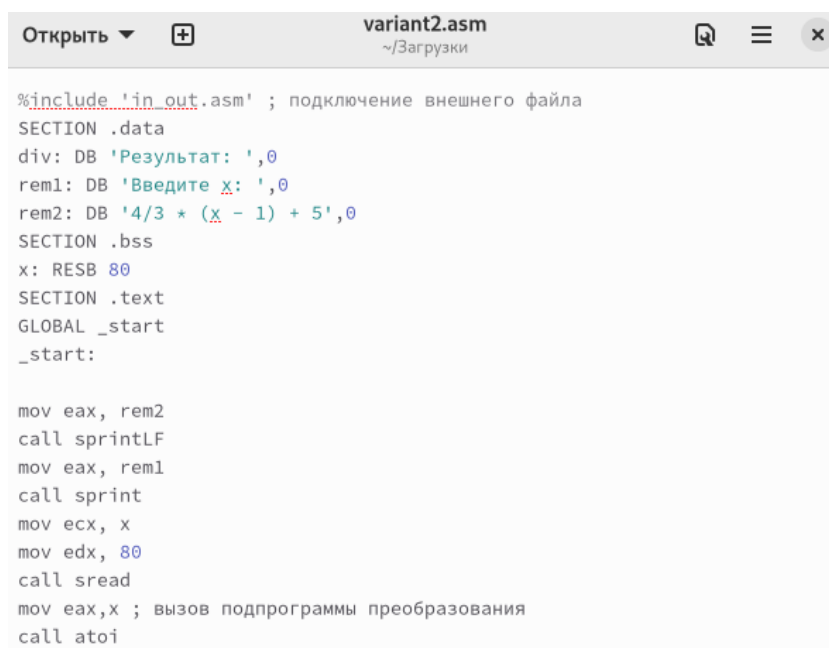
Рис. 3.20: Работа исполняемого файла

Ответы на вопросы лабораторной работы: 1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? `mov eax,rem call sprint` 2. Для чего используются следующие инструкции? `mov ecx, x` - запись входной переменной в регистр `ecx`; `mov edx, 80` - запись размера переменной в регистр `edx`; `call sread` - вызов процедуры чтения данных; 3. Для чего используется инструкция “`call atoi`”? Вызов `atoi` – функции преобразующей `ascii`-код символа в целое число и записывающий результат в регистр `eax`. 4. Какие строки листинга 7.4 отвечают за вычисления варианта? `xor edx,edx mov ebx,20 div ebx inc edx` 5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? В ре-

гистр ebx. 6. Для чего используется инструкция “inc edx”? Увеличивает значение edx на 1. 7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? `mov eax,rem call sprint mov eax,edx call iprintLF`

3.3 Задание для самостоятельной работы

Написали программу вычисления выражения $x = x(x)$. Программа выводит выражение для вычисления, выводит запрос на ввод значения x , вычисляет заданное выражение в зависимости от введенного x , выводит результат вычислений. (рис. 3.21) Вид функции $x(x)$ выбрали из таблицы 7.3 вариантов заданий, наш номер - 4, полученный при выполнении лабораторной работы. Создали исполняемый файл и проверили его работу для значений $x_1 = 1$ и $x_2 = 3$ из 7.3. (рис. 3.22).



```
Открыть + variant2.asm ~\Загрузки
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem1: DB 'Введите x: ',0
rem2: DB '4/3 * (x - 1) + 5',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:

mov eax, rem2
call sprintLF
mov eax, rem1
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi
```

Рис. 3.21: Файл variant.asm

```
[asstarceval@fedora lab07]$ nasm -f elf variant2.asm
[asstarceval@fedora lab07]$ ld -m elf_i386 -o variant2 variant2.o
[asstarceval@fedora lab07]$ ./variant2
4/3 * (x - 1) + 5
Введите x: 4
Результат: 9
[asstarceval@fedora lab07]$ ./variant2
4/3 * (x - 1) + 5
Введите x: 10
Результат: 17
[asstarceval@fedora lab07]$
```

Рис. 3.22: Работа исполняемого файла

4 Выводы

В результате выполнения лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.