

Отчёта по лабораторной работе 7

Освоение арифметических инструкций языка ассемблера NASM.

Цвелев С.А. НПИбд-02-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	20

Список иллюстраций

2.1	Пример программы	7
2.2	Работа программы	7
2.3	Пример программы	8
2.4	Работа программы	9
2.5	Пример программы	10
2.6	Работа программы	10
2.7	Пример программы	11
2.8	Работа программы	11
2.9	Работа программы	11
2.10	Пример программы	12
2.11	Работа программы	13
2.12	Пример программы	14
2.13	Работа программы	15
2.14	Пример программы	16
2.15	Работа программы	16
2.16	Пример программы	18
2.17	Работа программы	19

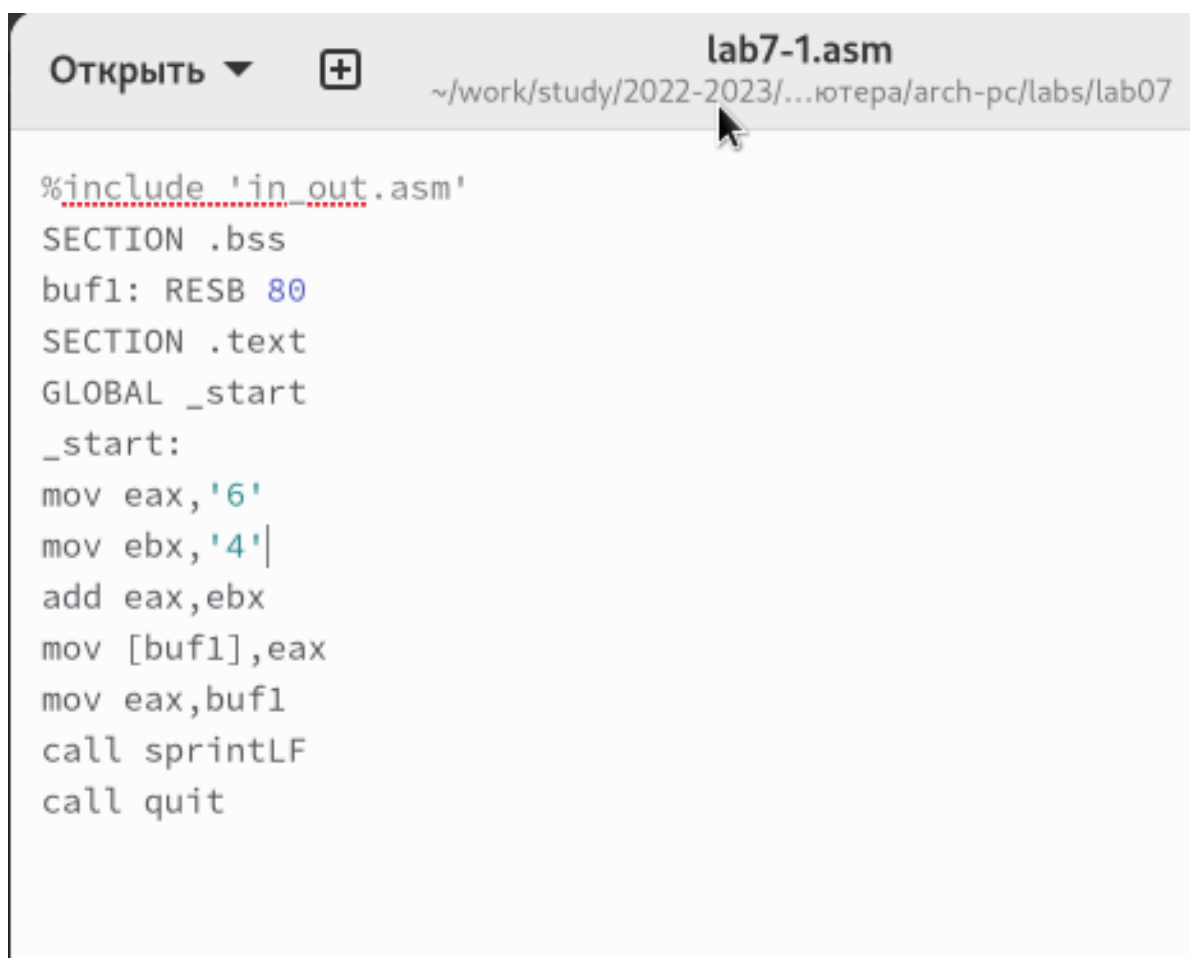
Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

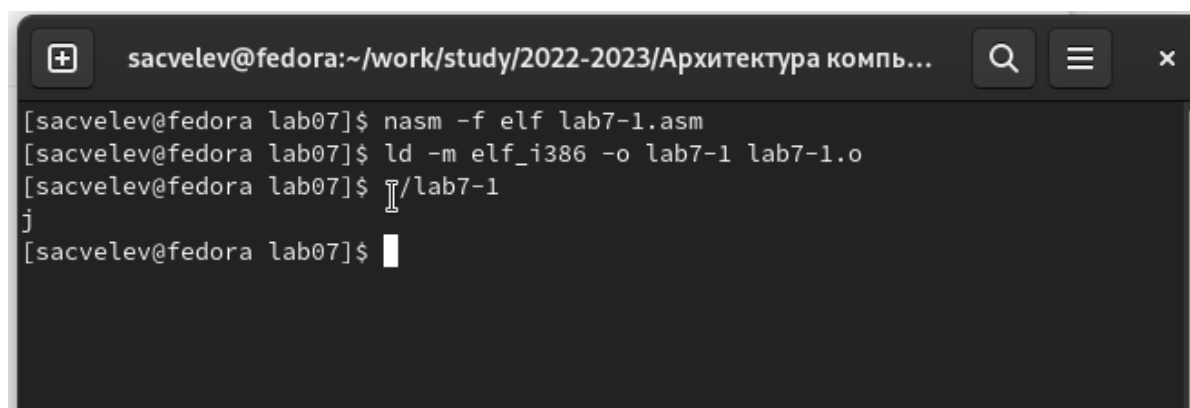
1. Создайте каталог для программ лабораторной работы № 6, перейдите в него и создайте файл lab7-1.asm:
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax. (рис. 2.1, 2.2)



```
Открыть ▾ + lab7-1.asm
~/work/study/2022-2023/...ютера/arch-pc/labs/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. 2.1: Пример программы



```
sacvelev@fedora:~/work/study/2022-2023/Архитектура компь...
[sacvelev@fedora lab07]$ nasm -f elf lab7-1.asm
[sacvelev@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[sacvelev@fedora lab07]$ ./lab7-1
j
[sacvelev@fedora lab07]$
```

Рис. 2.2: Работа программы

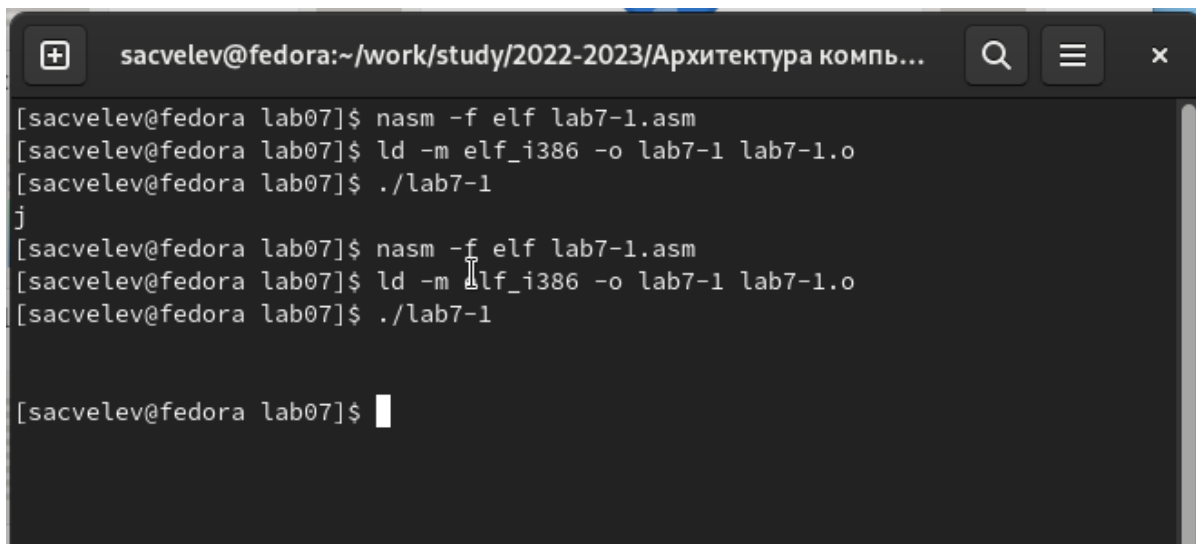
3. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 1) следующим образом: (рис.

2.3, 2.4)



```
Открыть ▾ + lab7-1.asm  
~/work/study/2022-2023/...ютера/arch-pc/labs/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. 2.3: Пример программы

A terminal window with a dark background. The title bar shows the user 'sacvelev@fedora' and the path '~/work/study/2022-2023/Архитектура компь...'. The terminal contains the following commands and output:


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

[sacvelev@fedora lab07]$
```

Рис. 2.4: Работа программы

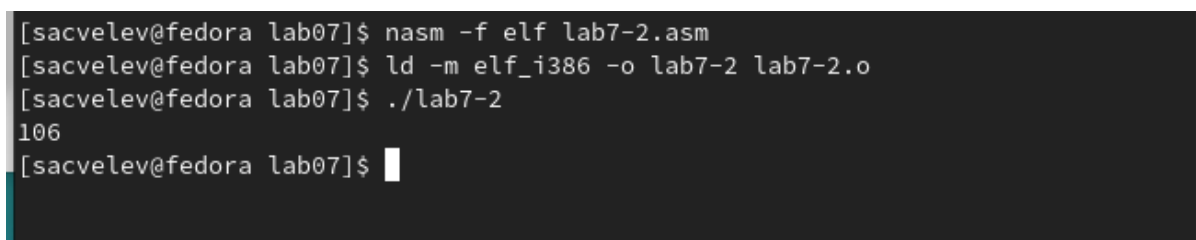
Никакой символ не виден, но он есть. Это возврат каретки LF.

4. Как отмечалось выше, 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. (рис. 2.5, 2.6)



```
lab7-1.asm    lab7-2.asm x
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
|
I
```

Рис. 2.5: Пример программы



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

Рис. 2.6: Работа программы

В результате работы программы мы получим 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. 2.7, 2.8)

Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы? – получили число 10

lab7-1.asm	lab7-2.asm
<pre>%include 'in_out.asm' SECTION .text GLOBAL _start _start: mov eax,6 mov ebx,4 add eax,ebx call iprintLF call quit</pre>	

Рис. 2.7: Пример программы

```
[sacvelev@fedora lab07]$ nasm -f elf lab7-2.asm
[sacvelev@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[sacvelev@fedora lab07]$ ./lab7-2
106
[sacvelev@fedora lab07]$ nasm -f elf lab7-2.asm
[sacvelev@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[sacvelev@fedora lab07]$ ./lab7-2
10
[sacvelev@fedora lab07]$
```

Рис. 2.8: Работа программы

Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`? - Вывод отличается что нет переноса строки. (рис. 2.9)

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

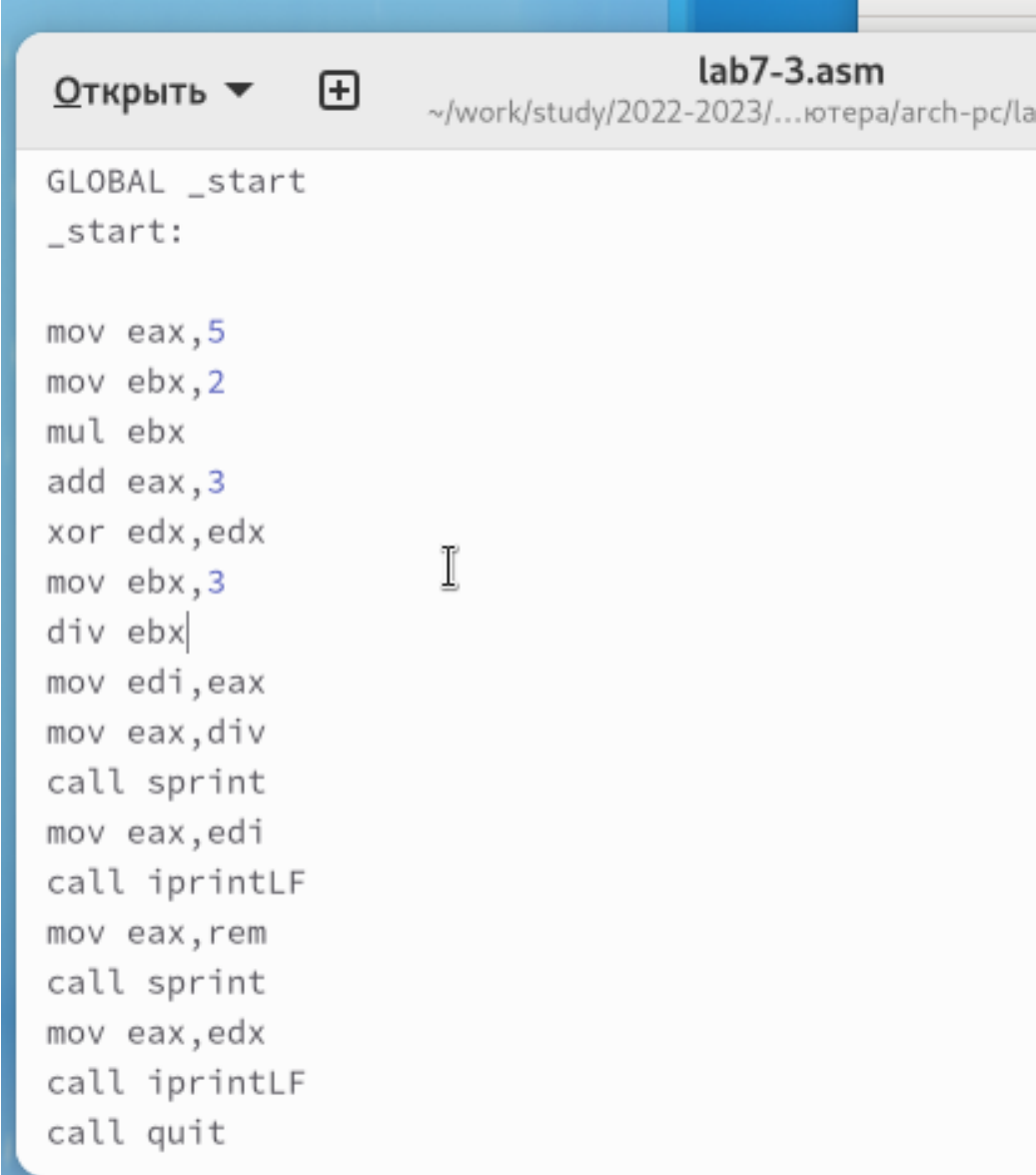
Рис. 2.9: Работа программы

6. В качестве примера выполнения арифметических операций в NASM приве-

дем программу вычисления арифметического выражения

$$f(x) = (5 * 2 + 3) / 3$$

. (рис. 2.10, рис. 2.11)



```
lab7-3.asm
~/work/study/2022-2023/...ютера/arch-pc/la

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. 2.10: Пример программы

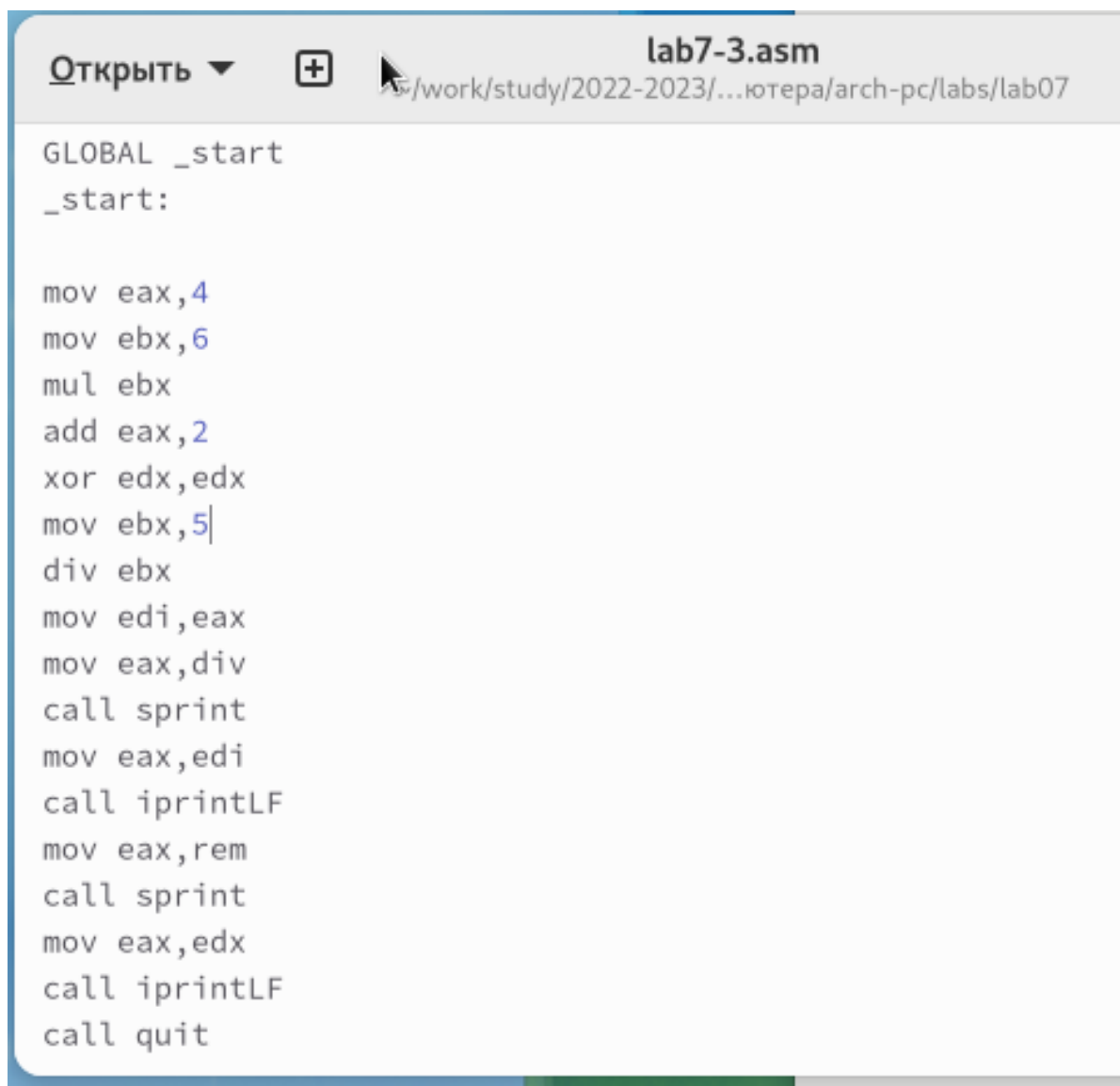
```
[sacvelev@fedora lab07]$ nasm -f elf lab7-3.asm
[sacvelev@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[sacvelev@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[sacvelev@fedora lab07]$
```

Рис. 2.11: Работа программы

Измените текст программы для вычисления выражения

$$f(x) = (4 * 6 + 2) / 5$$

. Создайте исполняемый файл и проверьте его работу. (рис. 2.12, рис. 2.13)



```
GLOBAL _start
_start:

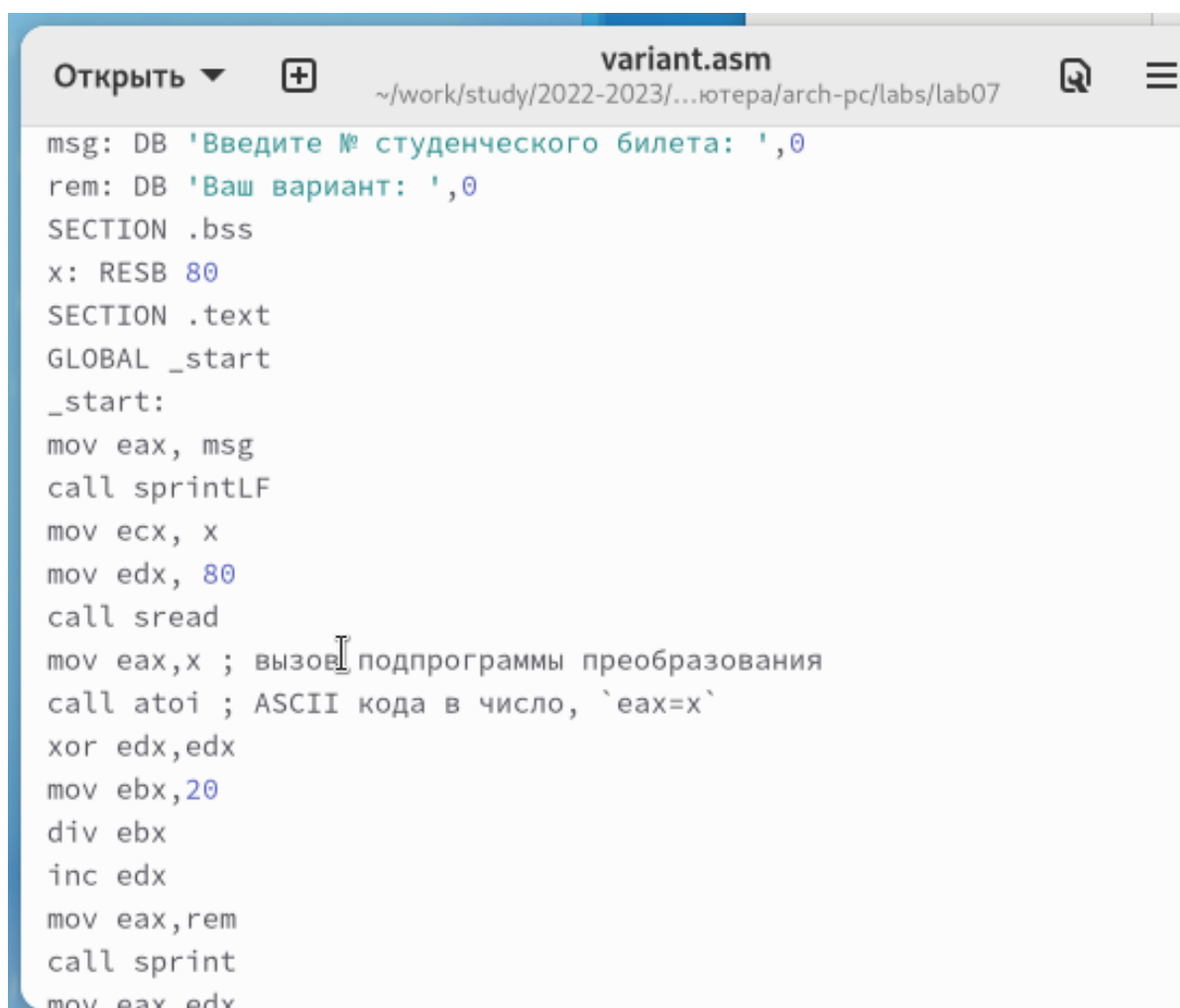
mov eax,4
mov ebx,6
mul ebx
add eax,2
xor edx,edx
mov ebx,5
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 2.12: Пример программы

```
[sacvelev@fedora lab07]$ nasm -f elf lab7-3.asm
[sacvelev@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[sacvelev@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[sacvelev@fedora lab07]$
[sacvelev@fedora lab07]$
[sacvelev@fedora lab07]$ nasm -f elf lab7-3.asm
[sacvelev@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[sacvelev@fedora lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[sacvelev@fedora lab07]$
```

Рис. 2.13: Работа программы

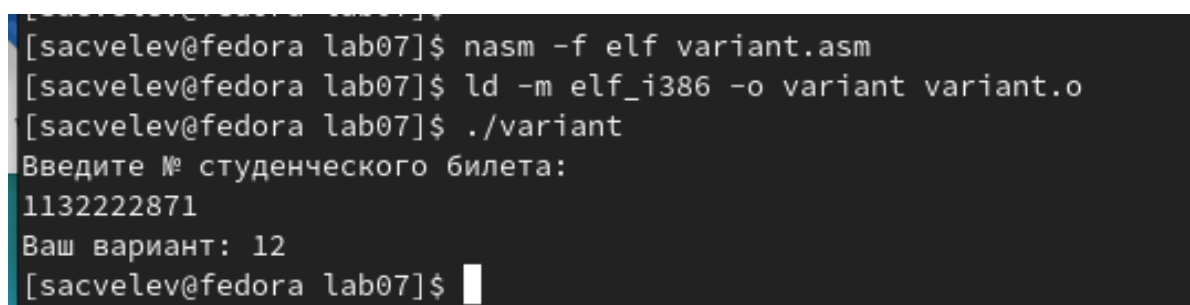
7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму: (рис. 2.14, рис. 2.15)



The screenshot shows a code editor window titled 'variant.asm' with a file path '~/.work/study/2022-2023/...ютера/arch-pc/labs/lab07'. The code is as follows:

```
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 ; 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 sprintf
mov eax, edx
```

Рис. 2.14: Пример программы



The screenshot shows a terminal window with the following commands and output:

```
[sacvelev@fedora lab07]$ nasm -f elf variant.asm
[sacvelev@fedora lab07]$ ld -m elf_i386 -o variant variant.o
[sacvelev@fedora lab07]$ ./variant
Введите № студенческого билета:
1132222871
Ваш вариант: 12
[sacvelev@fedora lab07]$
```

Рис. 2.15: Работа программы

- Какие строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант:?' – `mov eax, rem` – перекладывает в регистр значение переменной

с фразой ‘Ваш вариант:’ `call sprint` – вызов подпрограммы вывода строки

- Для чего используются следующие инструкции? `nasmmov ecx, x`
`mov edx, 80` `call sread`

Считывает значение студбилета в переменную X из консоли

- Для чего используется инструкция “`call atoi`”? – эта подпрограмма переводит введенные символы в числовой формат
- Какие строки листинга 7.4 отвечают за вычисления варианта? `xor edx, edx`
`mov ebx, 20` `div ebx`
- В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? 1 байт AH 2 байта DX 4 байта EDX – наш случай
- Для чего используется инструкция “`inc edx`”? по формуле вычисления варианта нужно прибавить единицу
- Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений `mov eax, edx` – результат перекладывается в регистр `eax` `call iprintLF` – вызов подпрограммы вывода

8. 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 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. (рис. 2.16, рис. 2.17)

Получили вариант 12 -

$$(8x - 6)/2$$

для $x=1$ и 5



```
calc.asm
~/work/study/2022-2023/...ютера/arch-pc/lab

call sprintf
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, 8
mul ebx
sub eax, 6
mov ebx, 2
div ebx

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

Рис. 2.16: Пример программы

```
[sacvelev@fedora lab07]$  
[sacvelev@fedora lab07]$ nasm -f elf calc.asm  
[sacvelev@fedora lab07]$ ld -m elf_i386 -o calc calc.o  
[sacvelev@fedora lab07]$ ./calc  
Введите X  
1  
выражение = : 1  
[sacvelev@fedora lab07]$ ./calc  
Введите X  
5  
выражение = : 17  
[sacvelev@fedora lab07]$
```

Рис. 2.17: Работа программы

3 Выводы

Изучили работу с арифметическими операциями