

Отчет по Лабораторной работе №6

Архитектура компьютеров и операционные системы

Бекауов Артур Тимурович НКАбд-01-23

Содержание

1	Цель работы	5
2	Ход лабораторной работы	6
2.1	1) Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?	16
3	Ход самостоятельной работы	18
4	Выводы	21

Список иллюстраций

2.1	Создание папки для ЛОН№6 и файла “lab6-1.asm”	6
2.2	Текст программы “lab6-1.asm”	7
2.3	Первый вывод файла “lab6-1”	7
2.4	Исправленная часть “lab6-1.asm”	8
2.5	Второй вывод файла “lab6-1”	8
2.6	10-ый символ таблицы ASCII”	8
2.7	Текст программы “lab6-2.asm”	9
2.8	Первый вывод файла “lab6-2”	9
2.9	Отредактированный текст “lab6-2.asm”	10
2.10	Второй вывод файла “lab6-2”	10
2.11	Исправленная часть “lab6-2.asm”	11
2.12	Сравнение второго и третьего выводов “lab6-2”	11
2.13	Текст программы “lab6-3.asm”	12
2.14	Первый вывод файла “lab6-3”	13
2.15	Исправленная часть “lab6-3.asm”	13
2.16	Второй вывод файла “lab6-3”	13
2.17	Текст программы “variant.asm”	15
2.18	Вывод файла “variant”	16
3.1	Текст программы “lab6-4.asm”	19
3.2	Выводы программы “lab6-4”	20

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

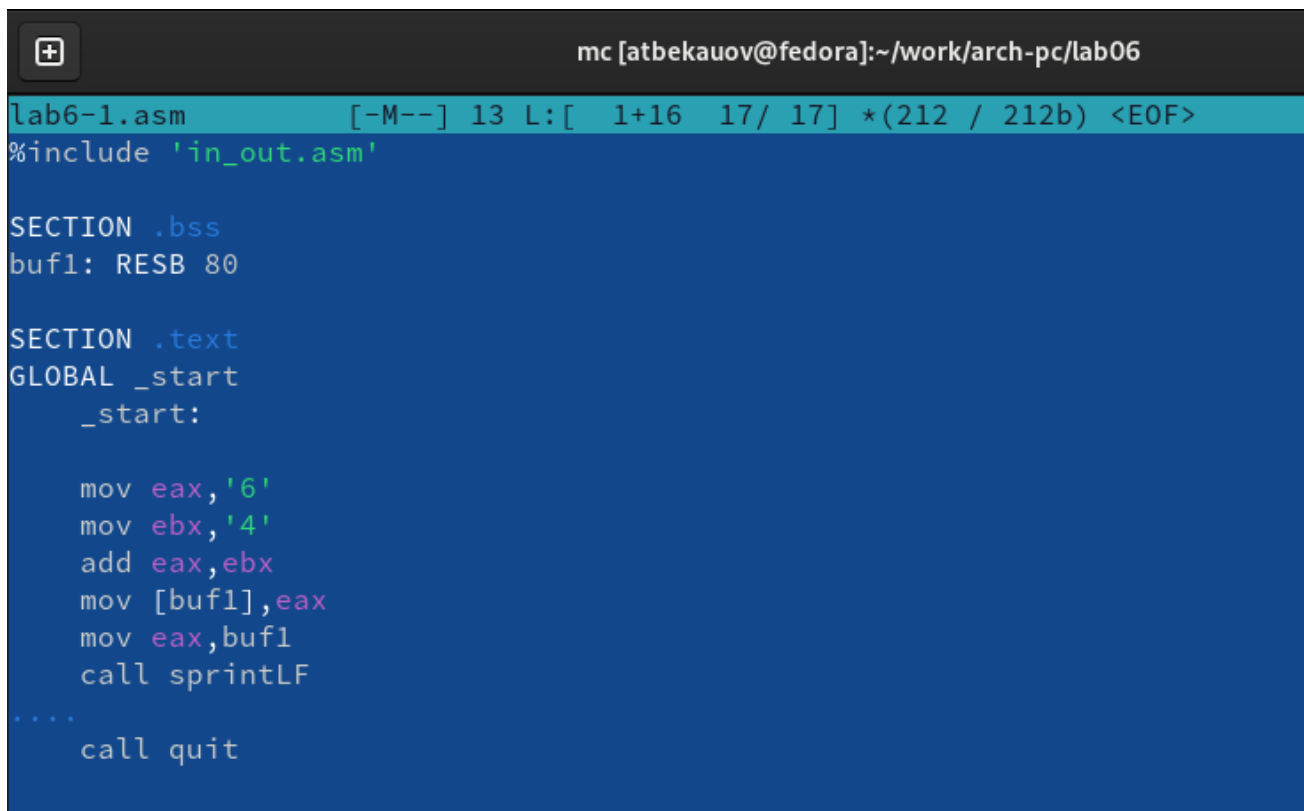
2 Ход лабораторной работы

Символьные и численные данные в NASM

Создаю каталог для программ ЛОН[№]6, перехожу в него скопирую из папки для ЛОН[№]5 файл “in_out.asm” и создаю текстовый файл “lab6-1.asm” (Рис. 2.1). Открываю файл в mcedit, и ввожу в него текст листинга 6.1 (Рис. 2.2)

```
[atbekauov@fedora ~]$ mkdir work/arch-pc/lab06
[atbekauov@fedora ~]$ cd work/arch-pc/lab06
[atbekauov@fedora lab06]$ touch lab6-1.asm
[atbekauov@fedora lab06]$ mcedit lab6-1.asm
```

Рис. 2.1: Создание папки для ЛОН[№]6 и файла “lab6-1.asm”



```
mc [atbekauov@fedora]:~/work/arch-pc/lab06
lab6-1.asm [-M--] 13 L:[ 1+16 17/ 17] *(212 / 212b) <EOF>
#include 'in_out.asm'

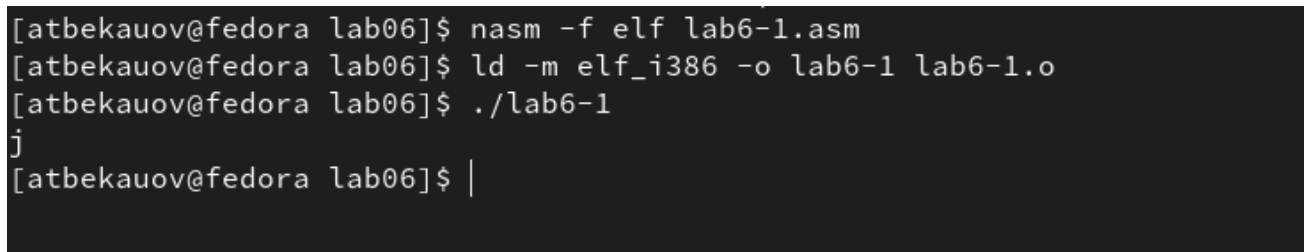
SECTION .bss
buf1: RESB 80

SECTION .text
GLOBAL _start
_start:

    mov eax, '6'
    mov ebx, '4'
    add eax, ebx
    mov [buf1], eax
    mov eax, buf1
    call sprintf
    ....
    call quit
```

Рис. 2.2: Текст программы “lab6-1.asm”

Создаю исполняемый файл “lab6-1” и запускаю его (Рис. 2.3)



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

Рис. 2.3: Первый вывод файла “lab6-1”

В качестве ответа, вопреки ожиданиям, на экране появилась буква j, а не число 10. Это происходит потому, что программа посчитала сумму кодов символов 4 и 6 в двоичном представлении и вывела символ j, код которого соответствует этой сумме.

Далее изменю текст программы и вместо символов запишу в регистры числа (Рис. 2.4).

```
GLOBAL _start
_start:

mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
```

Рис. 2.4: Исправленная часть “lab6-1.asm”

Создаю опять исполняемый файл “lab6-1” и запускаю его (Рис. 2.5)

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

[atbekauov@fedora lab06]$ |
```

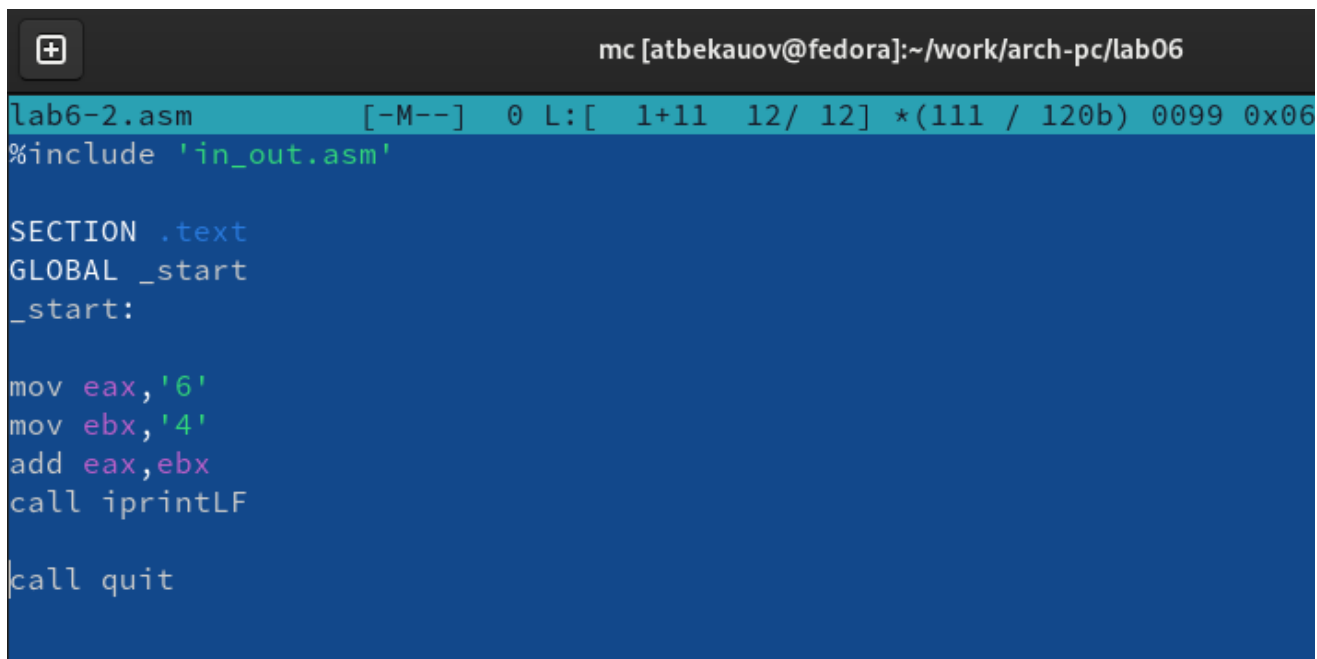
Рис. 2.5: Второй вывод файла “lab6-1”

Как видно, число 10 опять не было выведено на экран. Вместо этого выведен символ с кодом 10, пользуясь таблицей ASCII, вижу что десятичному коду 10 соответствует “line feed” - символ переноса на новую строку (Рис. 2.6). Символ как таковой не видно, но его последствия мы видим - на месте вывода пустая строка.

10	12	0x0A	1010	LF, \n
----	----	------	------	--------

Рис. 2.6: 10-ый символ таблицы ASCII”

Создаю файл “lab6-2.asm” в папке ЛОН⁶, открываю его в mcedit, и ввожу в него текст листинга 6.2 (Рис. 2.7).



```
lab6-2.asm [-M--] 0 L:[ 1+11 12/ 12] *(111 / 120b) 0099 0x06
#include 'in_out.asm'

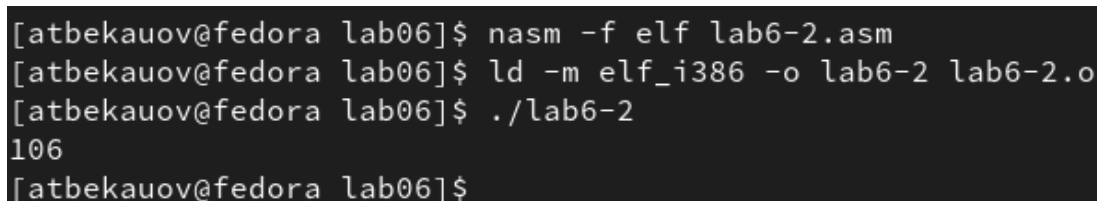
SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 2.7: Текст программы “lab6-2.asm”

Создам исполняемый файл “lab6-2” и запущу его (Рис. 2.8)

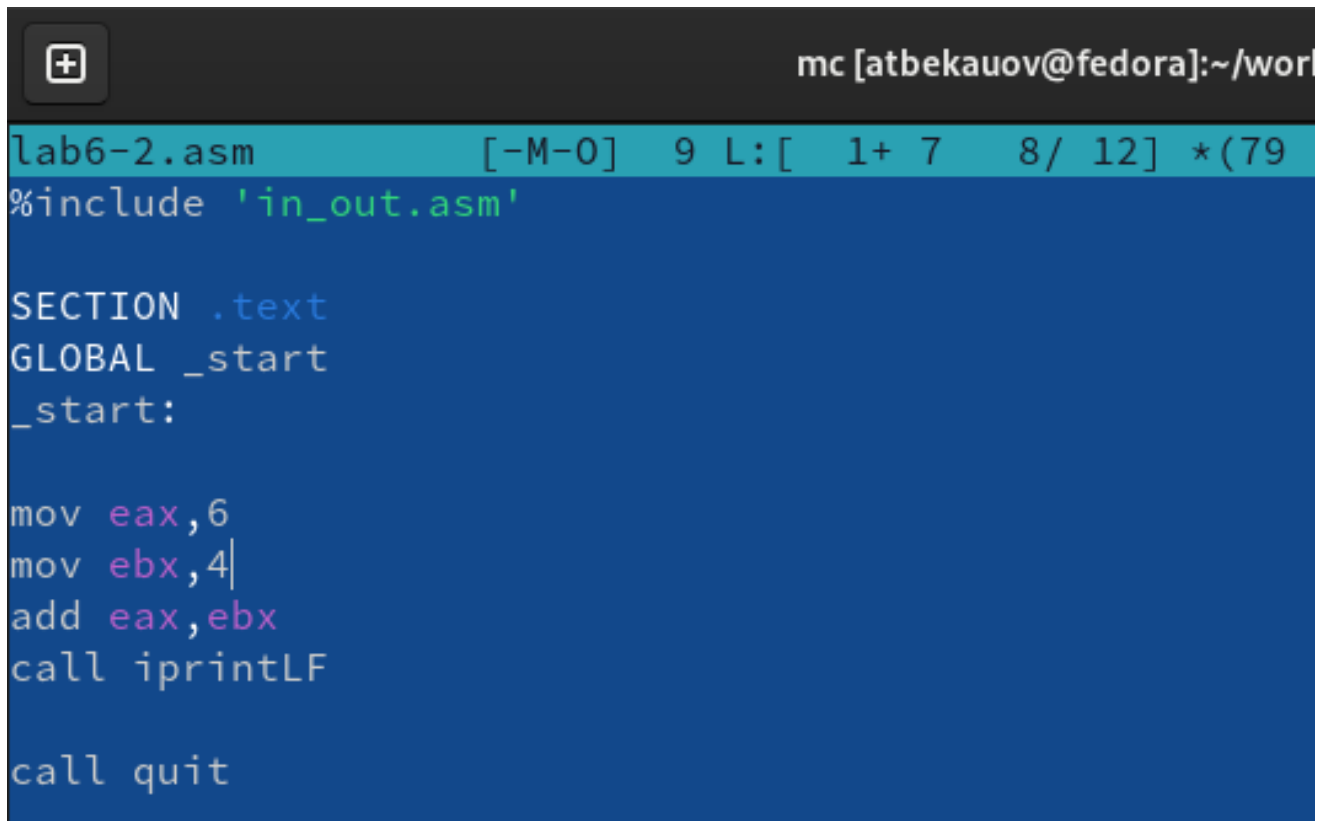


```
[atbekauov@fedora lab06]$ nasm -f elf lab6-2.asm
[atbekauov@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[atbekauov@fedora lab06]$ ./lab6-2
106
[atbekauov@fedora lab06]$
```

Рис. 2.8: Первый вывод файла “lab6-2”

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 “lab6-1”, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущей программе, изменим в тексте программы “lab6-2.asm” символы на числа (Рис. 2.9)



```
lab6-2.asm      [-M-O]  9 L:[  1+ 7   8/ 12] *(79
#include 'in_out.asm'

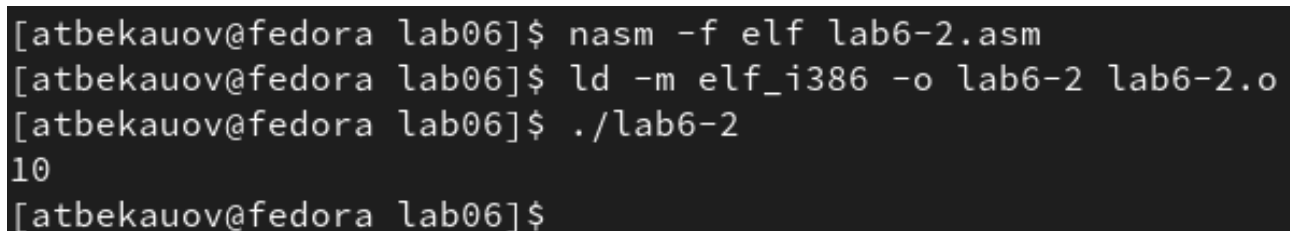
SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 2.9: Отредактированный текст “lab6-2.asm”

Создаю опять исполняемый файл “lab6-2” и запускаю его (Рис. 2.10)



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

Рис. 2.10: Второй вывод файла “lab6-2”

В этот раз всё получилось, как я и хотел и вывелось 10 - сумма чисел 4 и 6. Заменяю функцию `iprintLF` на `iprint` в тексте программы “lab6-2.asm” (Рис. 2.11) и сравню получившиеся результаты программ “lab6-2.asm” (Рис. 2.12)

```

mov ebx,4
add eax,ebx
call iprint

call quit

```

Рис. 2.11: Исправленная часть “lab6-2.asm”

```

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

```

Рис. 2.12: Сравнение второго и третьего выводов “lab6-2”

Как можно увидеть, разница в выводе двух программ заключается в том, что после команды `iprintLF` совершается переход на новую строку, а во-втором случае - после `iprint` переход не совершается, из-за чего вывод программы “lab6-2.asm” оказался на одной строке с вызовом новой команды терминалом.

Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических действий создам программу вычисления арифметического выражения $f(x)=(5*2+3)/3$. Для этого создаю файл “lab6-3.asm” в папке ЛОН⁶, открываю его в `mcedit`, и ввожу в него текст листинга 6.3 (Рис. 2.13).

```
mc [atbekauov@fedora]:~/work/arch-pc/lab06
lab6-3.asm [-M--] 0 L:[ 1+ 0 1/ 38] *(0 /1374b) 0059 0x03B
;-----
; 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: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов

call quit ; вызов подпрограммы завершения
```

Рис. 2.13: Текст программы “lab6-3.asm”

Создам исполняемый файл “lab6-3” и запущу его (Рис. 2.14)

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

Рис. 2.14: Первый вывод файла “lab6-3”

Вывод файла совпал с результатами предоставленными в методичке, также правильность его легко проверить, просто посчитав значение выражения.

Исправлю текст программы “lab6-3.asm” так, чтобы он вычислял выражение $f(x)=(4*6+2)/5$ (Рис. 2.15).

```
; ---- Вычисление выражения
mov eax,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDX=остаток от деления

mov edi,eax ; запись результата вычисления в 'edi'
```

Рис. 2.15: Исправленная часть “lab6-3.asm”

Создаю опять исполняемый файл “lab6-2” и запускаю его (Рис. 2.16)

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

Рис. 2.16: Второй вывод файла “lab6-3”

Вывод файла совпадает с моими расчетами.

В качестве другого примера создам программу вычисления варианта задания по номеру студенческого билета. Для этого создаю файл “variant.asm” в папке ЛОН№6, открываю его в mscedit, и ввожу в него текст листинга 6.4 (Рис. 2.17).

```

variant.asm      [-M--]  0 L:[  1+29  30/ 38] *(551 / 628b) 0100 0x064
;-----
; Программа вычисления варианта
;-----

%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
call iprintLF

call quit

```

Рис. 2.17: Текст программы “variant.asm”

Создам исполняемый файл “variant” и запущу его (Рис. 2.18)

```
[atbekauov@fedora lab06]$ nasm -f elf variant.asm
[atbekauov@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[atbekauov@fedora lab06]$ ./variant
Введите № студенческого билета:
1132236049
Ваш вариант: 10
[atbekauov@fedora lab06]$ |
```

Рис. 2.18: Вывод файла “variant”

Вывод программы совпадает с моими вычислениями ($1132236049 \bmod 20 = 9$, $9 + 1 = 10$). Мой вариант 10.

Вопросы к ходу лабораторной работы:

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

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

- 2) Для чего используются следующие инструкции? `mov ecx, x` `mov edx, 80` `call sread` -Эти инструкции запрашивают ввод (студенческого билета)
- 3) Для чего используется инструкция “`call atoi`”? -Для преобразования символьного текста введенного с клавиатуры (номера студенческого билета) в число.
- 4) Какие строки листинга 6.4 отвечают за вычисления варианта?
 - `mov ebx,20` `div ebx` `inc edx`
- 5) В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? `ebx`? -В регистр `eax`.

- 6) Для чего используется инструкция “inc edx”? -Для того,чтобы увеличить значение регистра edx на 1.
- 7) Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?
- mov eax,edx call iprintLF

3 Ход самостоятельной работы

Самостоятельное задание для моего варианта заключается в том, чтобы создать файл, получающий с клавиатуры значение x и вычисляющий значение выражения $f(x)=5*(x+18)-28$, а затем проверить результат выполнения программы на $x_1=2$, $x_2=3$. Для этого создаю файл “lab6-4.asm” в папке ЛОН^{№6}, открываю его в mscedit, и ввожу в него следующий текст программы (Рис. 3.1).

```

lab6-4.asm      [-M--] 27 L:[ 1+28 29/ 43] *(452 / 736b) 0010 0x00A
;-----
; Программа вычисления варианта
;-----

%include 'in_out.asm'

SECTION .data
fun: DB 'y=5*(x+18)-28
Введите значение x: ',0
ans: DB 'Результат : ',0

SECTION .bss
x: RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, fun
call sprint

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

; ---- Вычисление выражения
add eax, 18 ; EAX=x+18
mov ebx, 5 ; EBX=5
mul ebx ; EAX=EAX*EBX=(x+18)*5
sub eax, 28 ;EAX=EAX-28=5*(x+18)-28

mov edi, eax ;Запись результата в edi

; ---- Вывод результата на экран
mov eax,div.
call sprint
mov eax, edi
call iprintLF

call quit

```

Рис. 3.1: Текст программы “lab6-4.asm”

Создам исполняемый файл “lab6-4” и запущу его дважды, вводя значения x1 и x2(Рис. 3.2)

```
[atbekauov@fedora lab06]$ nasm -f elf lab6-4.asm
[atbekauov@fedora lab06]$ ld -m elf_i386 -o lab6-4 lab6-4.o
[atbekauov@fedora lab06]$ ./lab6-4
y=5*(x+18)-28
Введите значение x : 2
Результат :72
[atbekauov@fedora lab06]$ ./lab6-4
y=5*(x+18)-28
Введите значение x : 3
Результат :77
[atbekauov@fedora lab06]$ |
```

Рис. 3.2: Выводы программы “lab6-4”

4 Выводы

В ходе лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.