Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Шабалина Елизавета Андреевна

Содержание

Список иллюстраций

1 Цель работы

Цель данной лабораторной работы - освоение арифметческих инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx. - Непосредственная адресация – значение операнда задается непосредственно в команде, Например: mov ax,2. - Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что

сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

С помощью утилиты mkdir создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6. Перехожу в созданный каталог с помощью утилиты cd. и с помощью утилиты touch создаю файл lab6-1.asm (рис. 4.1).

```
eashabalina@dk3n35 ~ $ mkdir ~/work/arch-pc/lab06
eashabalina@dk3n35 ~ $ cd ~/work/arch-pc/lab06
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ touch lab6-1.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание файла

Копирую в текущий каталог файл in_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. 4.2).

```
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ cp ~/Загрузки/in_out.asm in_out.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm
```

Рис. 4.2: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. 4.3).

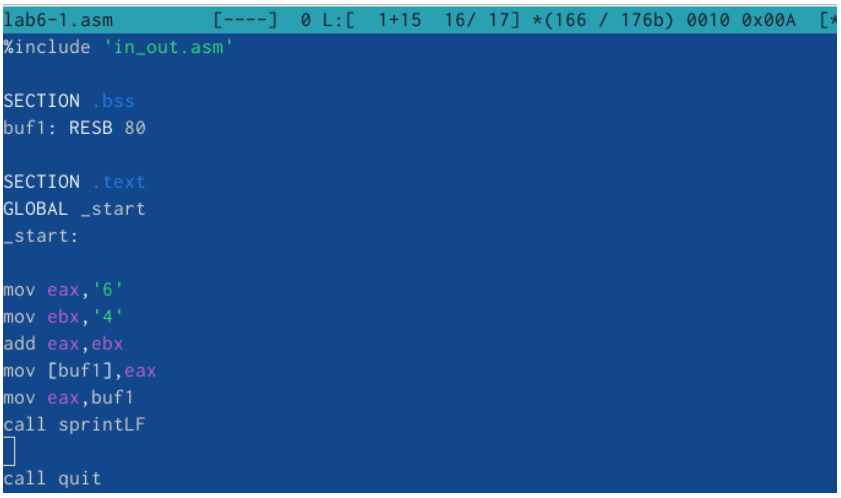

Рис. 4.3: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. 4.4). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

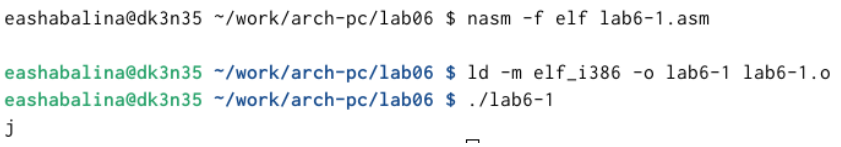

Рис. 4.4: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 4.5).

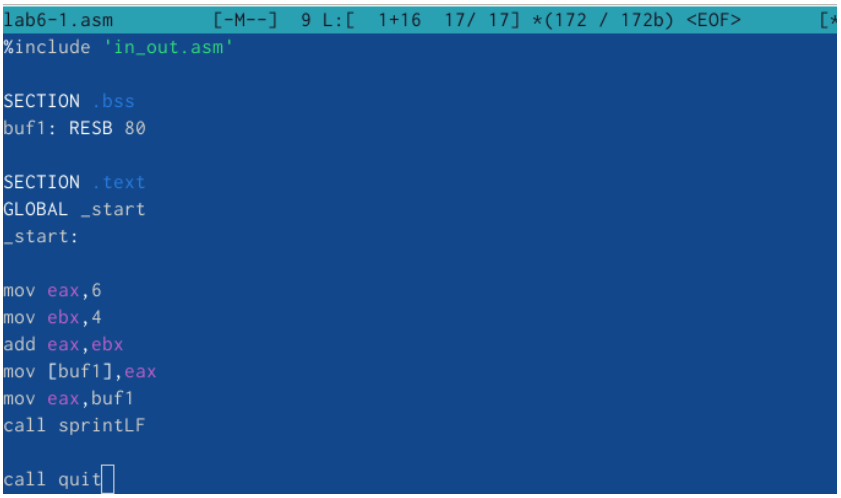

Рис. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.6). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 4.6: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. 4.7).

```
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 4.7: Создание файла

Ввожу в файл текст другойпрограммы для вывода значения регистра eax (рис. 4.8).

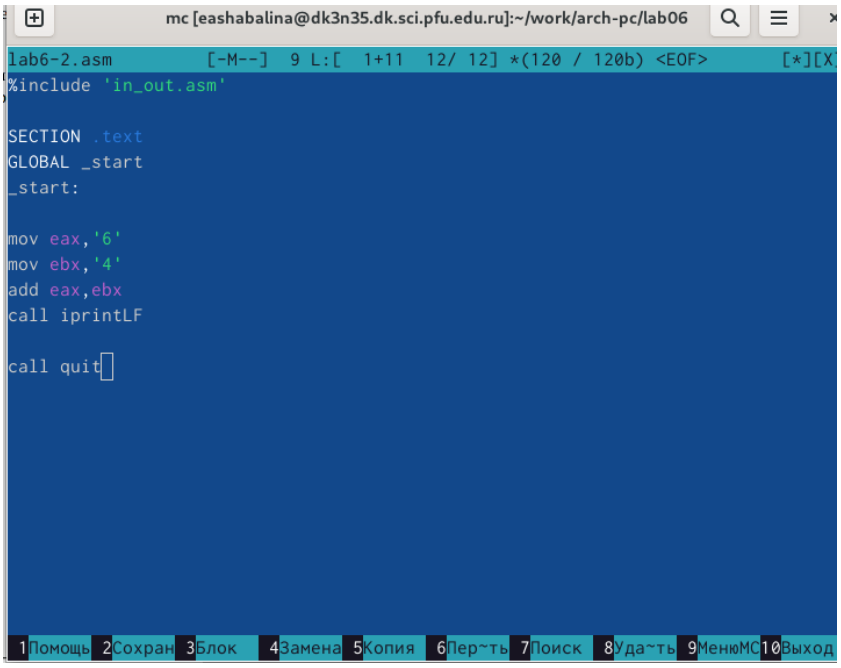

Рис. 4.8: Редактирование файла

Создаю и запускаю исполняемый файл lab7-2 (рис. 4.9). Теперь вывод число

106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

```
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.9: Запуск исполняемого файла

Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.10).

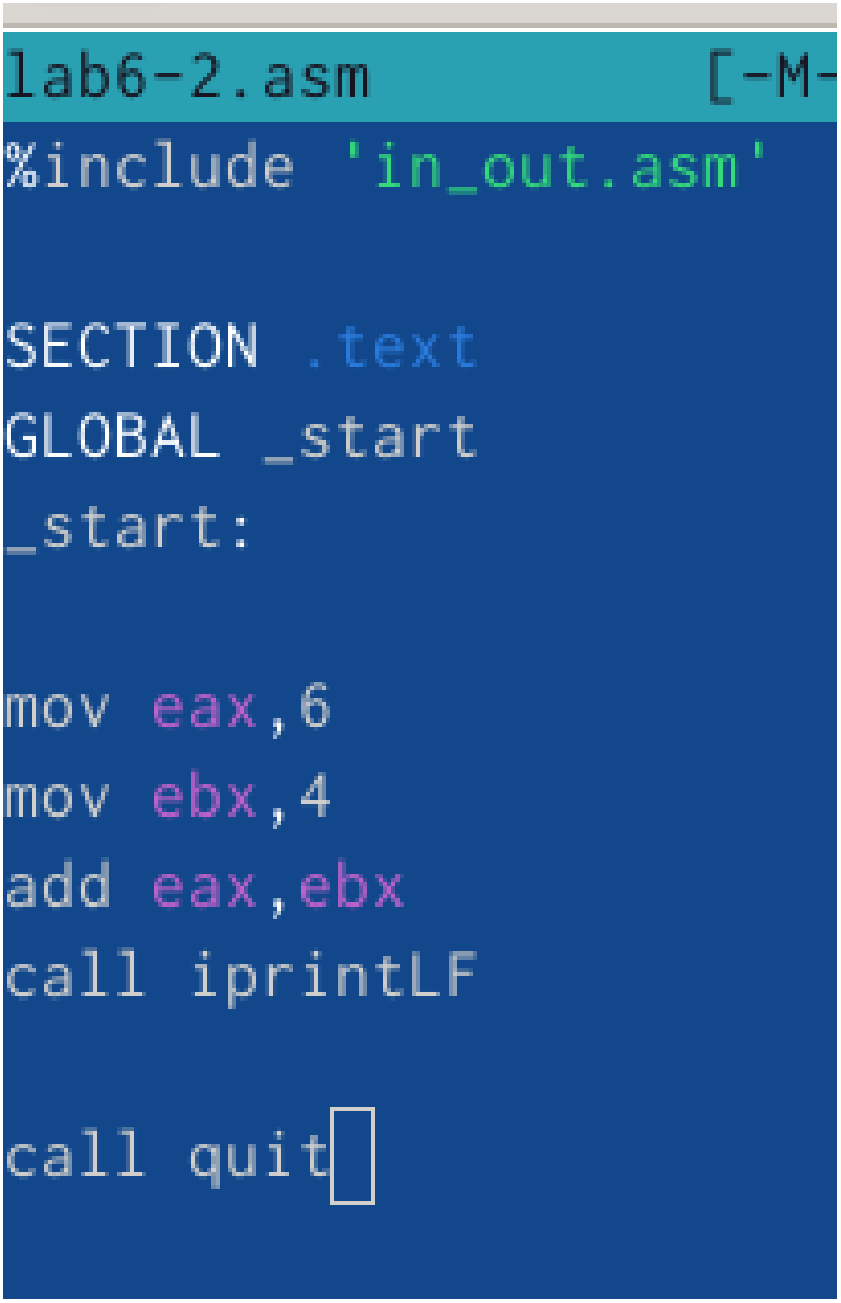

Рис. 4.10: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.11).. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 4.11: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.12).

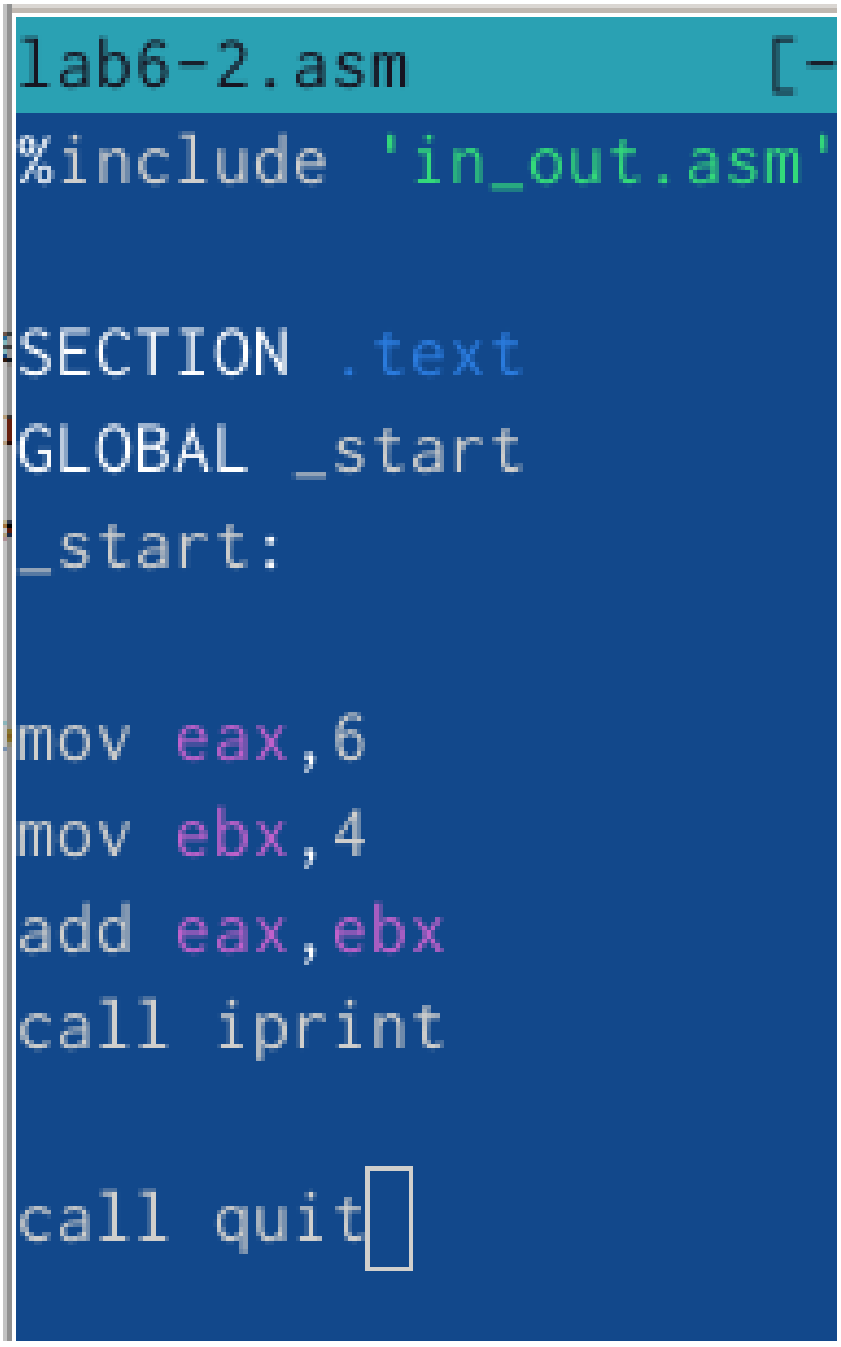

Рис. 4.12: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.13). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

```
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-2
10eashabalina@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 4.13: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью утилиты touch (рис. 4.14).

```
10eashabalina@dk3n35 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.a
touch ~/work/arch-pc/lab06/lab6-3.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
```

Рис. 4.14: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения f(x) = (5 * 2 + 3)/3 (рис. 4.15).

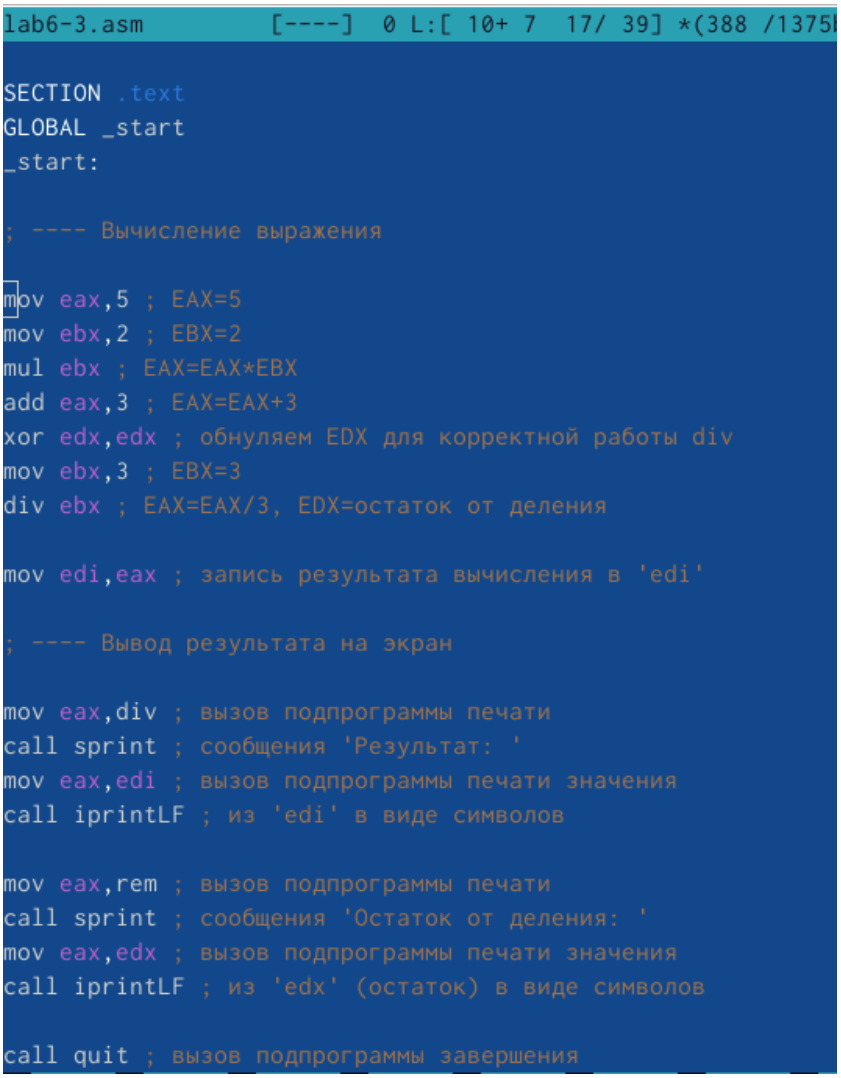

Рис. 4.15: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 4.16).

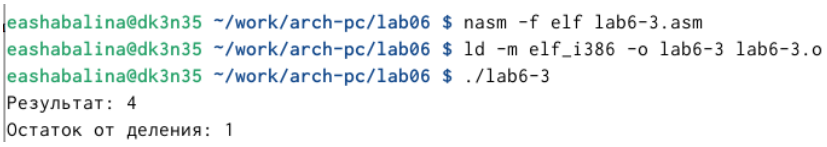

Рис. 4.16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения f(x) = (4 * 6 + 2)/5. Создаю и запускаю новый исполняемый файл. Я посчитала для проверки правильности работы программы значение выражения самостоятельно,

программа отработала верно(рис. 4.17).

```
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o

eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
eashabalina@dk3n35 ~/work/arch-pc/lab06 $
```

Рис. 4.17: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. 4.18).

```
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
eashabalina@dk3n35 ~/work/arch-pc/lab06 $ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3    lab6-3.o
```

Рис. 4.18: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.19).

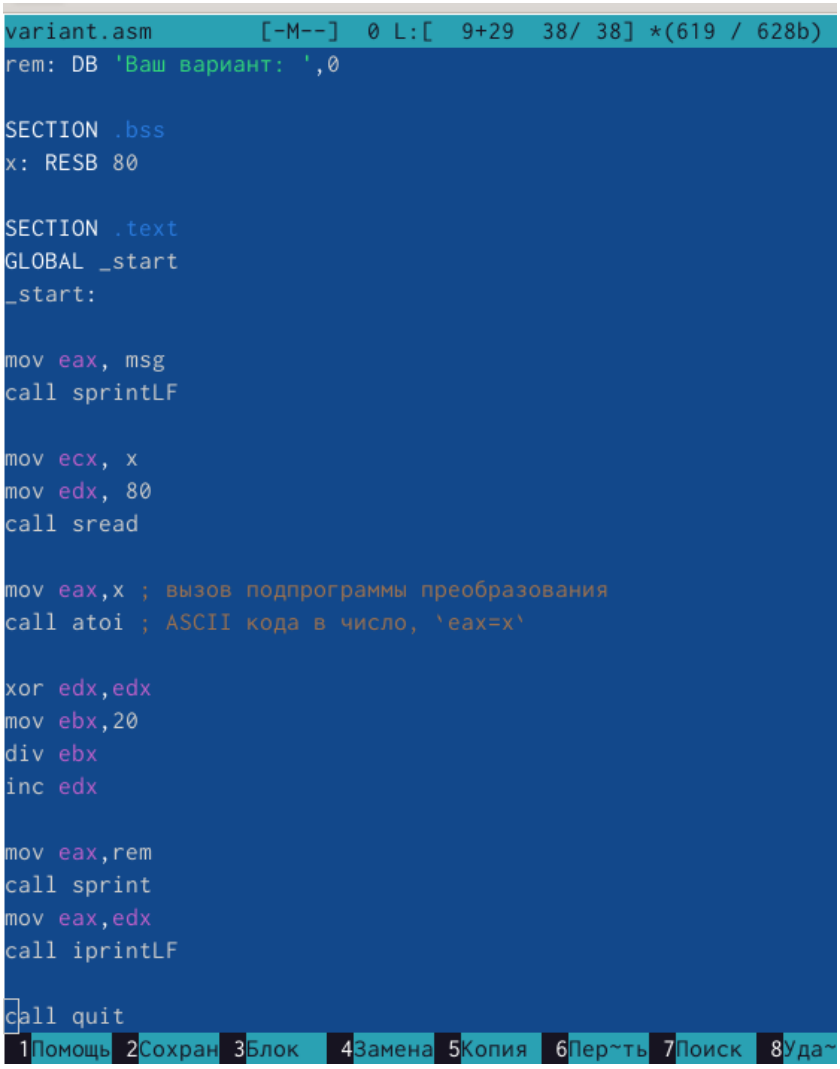

Рис. 4.19: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.20). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 8.

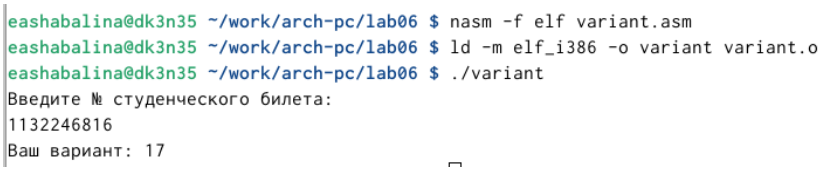

Рис. 4.20: Запуск исполняемого файла

4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

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

2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры

3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax

4. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div
mov ebx,20 ; ebx = 20
div ebx ; eax = eax/20, edx - остаток от деления
inc edx ; edx = edx + 1
```

5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx

6. Инструкция inc edx увеличивает значение регистра edx на 1

7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx
call iprintLF
```

4.3 Выполнение заданий для самостоятельной работы

Создаю файл lflf.asm с помощью утилиты touch. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения 18(x+1)/6 (рис. 4.21). Это выражение было под вариантом 8.

```
lflf.asm          [----] 11 L:[  2+31  33/ 41] *(667 / 952b) 0010 0x00A   [*]
; Программа вычисления выражения
;-----------------------------
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data
msg: DB 'Введите значение переменной: ',0
rem: DB 'Результат: ',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
add eax,1
mov ebx,18
mul ebx
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,6 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от делениядd


mov edi,eax


mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF

call quit ; вызов подпрограммы завершения
```

Рис. 4.21: Написание программы

Создаю и запускаю исполняемый файл (рис. 4.22). При вводе значения 3, вывод - 12, а при вводе значения 1, вывод - 6.

Рис. 4.22: Запуск исполняемого файла

Листинг 4.1. Программа для вычисления значения выражения 18(x+1)/6.

%include ‘in_out.asm’ ; подключение внешнего файла

SECTION .data msg: DB ‘Введите значение переменной:’,0 rem: DB ‘Результат:’,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 add eax,1 mov ebx,18 mul ebx xor edx,edx ; обнуляем EDX для корректной работы div mov ebx,6 ; EBX=3 div ebx ; EAX=EAX/3, EDX=остаток от делениядd

mov edi,eax

mov eax,rem ; вызов подпрограммы печати call sprint ; сообщения ‘Результат:’ mov eax,edi ; вызов подпрограммы печати значения call iprintLF

call quit ; вызов подпрограммы завершения

5 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.