

Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Филатов Илья Гурамович

Содержание

| | | |
|----------|---|-----------|
| 1 | Цель работы | 4 |
| 2 | Задание | 5 |
| 3 | Теоретическое введение | 6 |
| 4 | Выполнение лабораторной работы | 9 |
| 4.1 | Символьные и численные данные в NASM | 9 |
| 4.2 | Выполнение арифметических операций в NASM | 13 |
| 4.3 | Задание для самостоятельной работы | 19 |
| 5 | Выводы | 22 |
| 6 | Список литературы | 23 |

Список иллюстраций

| | | |
|------|--|----|
| 4.1 | Создание каталога и файла | 9 |
| 4.2 | Ввод программы | 9 |
| 4.3 | Копирование файла | 10 |
| 4.4 | Создание и запуск исполняемого файла | 10 |
| 4.5 | Изменение текста программы | 10 |
| 4.6 | Создание и запуск исполняемого файла | 11 |
| 4.7 | Создание файла | 11 |
| 4.8 | Ввод текста программы | 11 |
| 4.9 | Создание и запуск исполняемого файла | 12 |
| 4.10 | Изменение текста программы | 12 |
| 4.11 | Создание и запуск исполняемого файла | 12 |
| 4.12 | Замена функции | 13 |
| 4.13 | Создание и запуск исполняемого файла | 13 |
| 4.14 | Создание файла | 13 |
| 4.15 | Ввод программы | 14 |
| 4.16 | Создание и запуск исполняемого файла | 14 |
| 4.17 | Изменение файла | 15 |
| 4.18 | Создание и запуск исполняемого файла | 16 |
| 4.19 | Создание файла | 17 |
| 4.20 | Ввод программы | 17 |
| 4.21 | Создание и запуск исполняемого файла | 18 |
| 4.22 | Создание файла | 19 |
| 4.23 | Написание программы | 20 |
| 4.24 | Создание и запуск исполняемого файла | 21 |

1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Задание 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.

Существует три основных способа адресации:

- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров.
- Непосредственная адресация – значение операнда задается непосредственно в команде.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Схема команды целочисленного сложения `add` выполняет сложение двух операндов и записывает результат по адресу первого операнда.

Арифметические операции в NASM:

Команда целочисленного вычитания `sub` работает аналогично команде `add`.

Существуют специальные команды: `inc` и `dec`, которые увеличивают и уменьшают на 1 свой операнд.

Команда `neg` рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

Для беззнакового умножения используется команда `mul`, а для знакового умножения – команда `imul`. Для команд умножения один из сомножителей указывает-

ся в команде и должен находиться в регистре или в памяти, но не может быть непосредственным операндом. Второй сомножитель в команде явно не указывается и должен находиться в регистре EAX, AX или AL, а результат помещается в регистры EDX:EAX, DX:AX или AX, в зависимости от размера операнда.

Для деления, как и для умножения, существует 2 команды `div` (от англ. divide - деление) и `idiv`. В командах деления указывается только один операнд – делиТЕЛЬ, который может быть регистром или ячейкой памяти, но не может быть непосредственным операндом. Местоположение делимого и результата для команд деления зависит от размера делителя. Кроме того, так как в результате деления получается два числа – частное и остаток, то эти числа помещаются в определённые регистры.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII.

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

Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

Для выполнения лабораторных работ в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Это:

- `iprint` – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистрах необходимо записать выводимое число (`mov eax,`).
- `iprintLF` – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки.
- `atoi` – функция преобразует `ascii`-код символа в целое число и записает резуль-

тат в регистр еах, перед вызовом atoi в регистр еах необходимо записать число
(mov еах,).

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

Открываю терминал. Используя команду `mkdir` создаю каталог для лабораторной работы. Перехожу в него и командой `touch` создаю файл `lab6-1.asm` (рис. 4.1).



```
igfilatov@igfilatov:~/work/arch-pc/lab06
[igfilatov@igfilatov ~]$ mkdir ~/work/arch-pc/lab06
[igfilatov@igfilatov ~]$ cd ~/work/arch-pc/lab06
[igfilatov@igfilatov lab06]$ touch lab6-1.asm
[igfilatov@igfilatov lab06]$
```

Рис. 4.1: Создание каталога и файла

Открываю файл с помощью редактора `gedit` и ввожу текст программы из листинга 6.1 (рис. 4.2).



```
*lab6-1.asm
~/work/arch-pc/lab06

Открыть ▼ +

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

Рис. 4.2: Ввод программы

Чтобы программа, которая содержит подпрограммы из `in_out.asm`, работала корректно, командой `cp` копирую этот файл из каталога `lab05` в каталог `lab06` (рис. 4.3).

```
[igfilatov@igfilatov lab06]$ cd ~/work/arch-pc
[igfilatov@igfilatov arch-pc]$ cp lab05/in_out.asm lab06
[igfilatov@igfilatov arch-pc]$
```

Рис. 4.3: Копирование файла

Создаю исполняемый файл и запускаю его. Так как программа воспринимает 6 и 4 как символы (из-за записи в одинарных кавычках), то суммирует их коды и сопоставляет получившемуся коду символ из кодовой таблицы ASCII, в данном случае это “j” (рис. 4.4).

```
igfilatov@igfilatov:~/work/arch-pc/lab06
[igfilatov@igfilatov arch-pc]$ cd lab06
[igfilatov@igfilatov lab06]$ nasm -f elf lab6-1.asm
[igfilatov@igfilatov lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[igfilatov@igfilatov lab06]$ ./lab6-1
j
[igfilatov@igfilatov lab06]$
```

Рис. 4.4: Создание и запуск исполняемого файла

Открываю файл и убираю одинарные кавычки (рис. 4.5).

```
Открыть ▼ + *lab6-1.asm ~/work/arch-pc/lab06
1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 mov [buf1],eax
11 mov eax,buf1
12 call sprintLF
13 call quit
```

Рис. 4.5: Изменение текста программы

Создаю исполняемый файл и запускаю его. В отличие от предыдущего примера программа воспринимает 6 и 4 как коды, суммирует их и выводит символ с кодом их суммы - по таблице коду 10 соответствует символ перевода строки, который не отображается на экране (рис. 4.6).

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

[igfilatov@igfilatov lab06]$
```

Рис. 4.6: Создание и запуск исполняемого файла

Создаю файл lab6-2.asm и открываю его с помощью редактора gedit (рис. 4.7).

```
igfilatov@igfilatov:~/work/arch-pc/lab06 — gedit lab6-2.asm
[igfilatov@igfilatov lab06]$ touch ~/work/arch-pc/lab06/lab6-2.asm
[igfilatov@igfilatov lab06]$ gedit lab6-2.asm
```

Рис. 4.7: Создание файла

Ввожу текст программы из листинга 6.2 (рис. 4.8).

```
Открыть ▼ + *lab6-2.asm
~/work/arch-pc/lab06
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
```

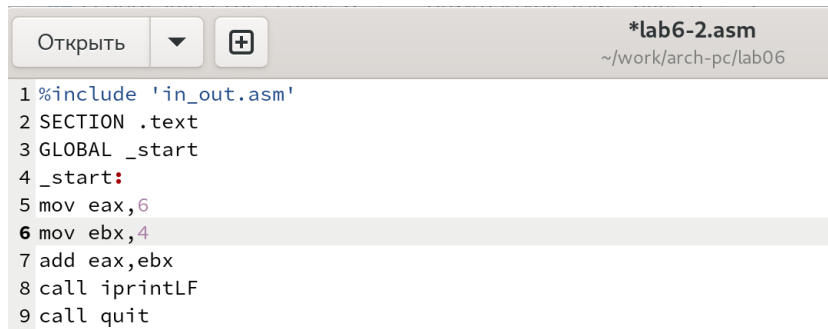
Рис. 4.8: Ввод текста программы

Создаю и запускаю исполняемый файл, который складывает коды символов 6 и 4 и, в отличие от программы из листинга 6.1, выводит число, а не символ, кодом которого является это число (рис. 4.9).

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

Рис. 4.9: Создание и запуск исполняемого файла

Открываю файл и убираю одинарные кавычки, аналогично предыдущей программе (рис. 4.10).



```
*lab6-2.asm
~/work/arch-pc/lab06

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. 4.10: Изменение текста программы

Создаю и запускаю исполняемый файл. В отличие от предыдущего примера программа складывает числа, а не коды символов, в результате мы получим число 10 (рис. 4.11).

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

Рис. 4.11: Создание и запуск исполняемого файла

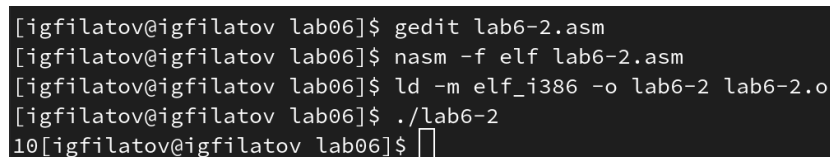
Открываю файл и с помощью программы gedit и меняю функцию iprintLF на iprint (рис. 4.12).



```
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 iprint
9 call quit
```

Рис. 4.12: Замена функции

Создаю и запускаю исполняемый файл. В отличие от предыдущего он не выводит с результатом символ переноса строки, из-за чего следующую команду можно написать на той же строке, на которой выводился ответ (рис. 4.13).

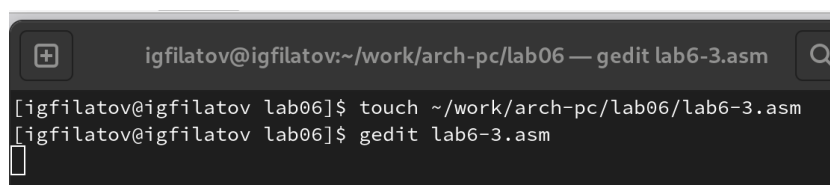


```
[igfilatov@igfilatov lab06]$ gedit lab6-2.asm
[igfilatov@igfilatov lab06]$ nasm -f elf lab6-2.asm
[igfilatov@igfilatov lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[igfilatov@igfilatov lab06]$ ./lab6-2
10[igfilatov@igfilatov lab06]$
```

Рис. 4.13: Создание и запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

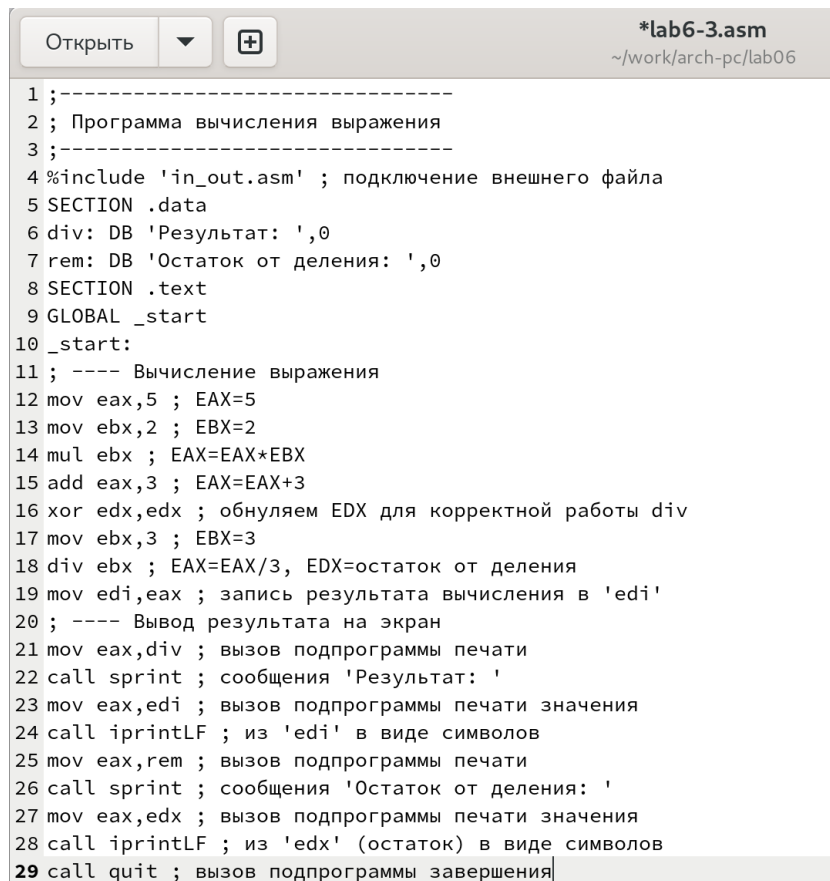
Создаю файл lab6-3.asm и открываю его с помощью редактора gedit (рис. 4.14).



```
igfilatov@igfilatov:~/work/arch-pc/lab06 — gedit lab6-3.asm
[igfilatov@igfilatov lab06]$ touch ~/work/arch-pc/lab06/lab6-3.asm
[igfilatov@igfilatov lab06]$ gedit lab6-3.asm
```

Рис. 4.14: Создание файла

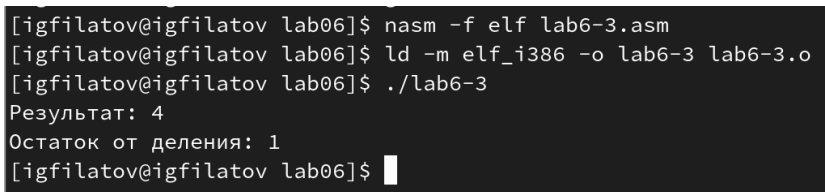
Ввожу текст из листинга 6.3 (рис. 4.15).



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

Рис. 4.15: Ввод программы

Создаю и запускаю исполняемый файл, который выводит значение выражения $(5*2+3)/3$ (рис. 4.16).



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

Рис. 4.16: Создание и запуск исполняемого файла

Открываю файл и меняю его так, чтобы он выводил значение выражения $(4*6+2)/5$ (рис. 4.17).

```

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

```

Рис. 4.17: Изменение файла

Текст программы:

```

;-----
; 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 ; EAX=4

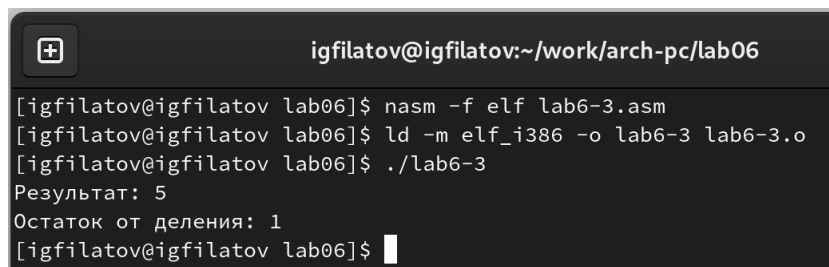
```

```

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'
; --- 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

```

Создаю и запускаю исполняемый файл, который выводит значение 5 с остатком 1, значит вычисления верны (рис. 4.18).



```

igfilatov@igfilatov:~/work/arch-pc/lab06
[igfilatov@igfilatov lab06]$ nasm -f elf lab6-3.asm
[igfilatov@igfilatov lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[igfilatov@igfilatov lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[igfilatov@igfilatov lab06]$

```

Рис. 4.18: Создание и запуск исполняемого файла

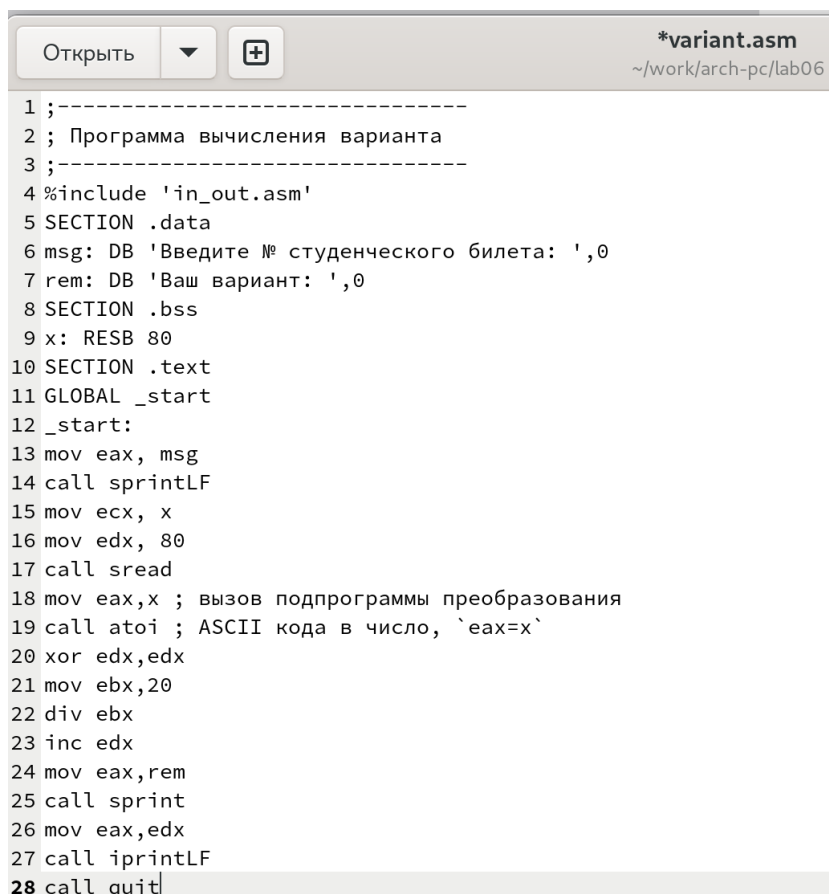
Создаю файл variant.asm и открываю его с помощью редактора gedit (рис. 4.19).



```
igfilatov@igfilatov:~/work/arch-pc/lab06 — gedit variant.asm
[igfilatov@igfilatov lab06]$ touch ~/work/arch-pc/lab06/variant.asm
[igfilatov@igfilatov lab06]$ gedit variant.asm
```

Рис. 4.19: Создание файла

Ввожу текст из листинга 6.4 (рис. 4.20).



```
Открыть ▼ + *variant.asm
~/work/arch-pc/lab06

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

Рис. 4.20: Ввод программы

Создаю и запускаю исполняемый файл. Ввожу номер своего студенческого билета – 1132246766. Так как число 1132246760 делится на 20, то остаток от деления – 6. Программа прибавляет к этому числу 1 и выводит верный результат – 7 (рис. 4.21).

```
[igfilatov@igfilatov lab06]$ nasm -f elf variant.asm
[igfilatov@igfilatov lab06]$ ld -m elf_i386 -o variant variant.o
[igfilatov@igfilatov lab06]$ ./variant
Введите № студенческого билета:
1132246766
Ваш вариант: 7
[igfilatov@igfilatov lab06]$
```

Рис. 4.21: Создание и запуск исполняемого файла

Ответы на вопросы:

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

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

- 2.

```
mov ecx, x
mov edx, 80
call sread
```

Первая инструкция используется чтобы сохранить адрес вводимой строки в регистр ecx. Вторая – чтобы сохранить число 80 в регистр edx. Третья – для вывода подпрограммы из файла in_out.asm, которая позволяет ввести сообщение.

3. Инструкция call atoi вызывает подпрограмму из файла in_out.asm, которая преобразует ascii-код символа в целое число.

4. За вычисление варианта отвечают эти инструкции:

```
xor edx,edx
mov ebx,20
div ebx
inc edx
```

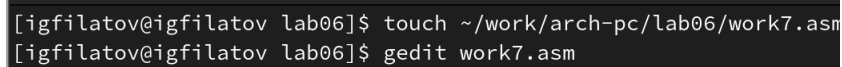
5. При выполнении инструкции “div ebx” остаток от деления записывается в регистр ebx.

6. Инструкция “inc edx” увеличивает значение, находящееся в регистре edx на 1
7. За вывод на экран результата вычислений отвечают сторки:

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

4.3 Задание для самостоятельной работы

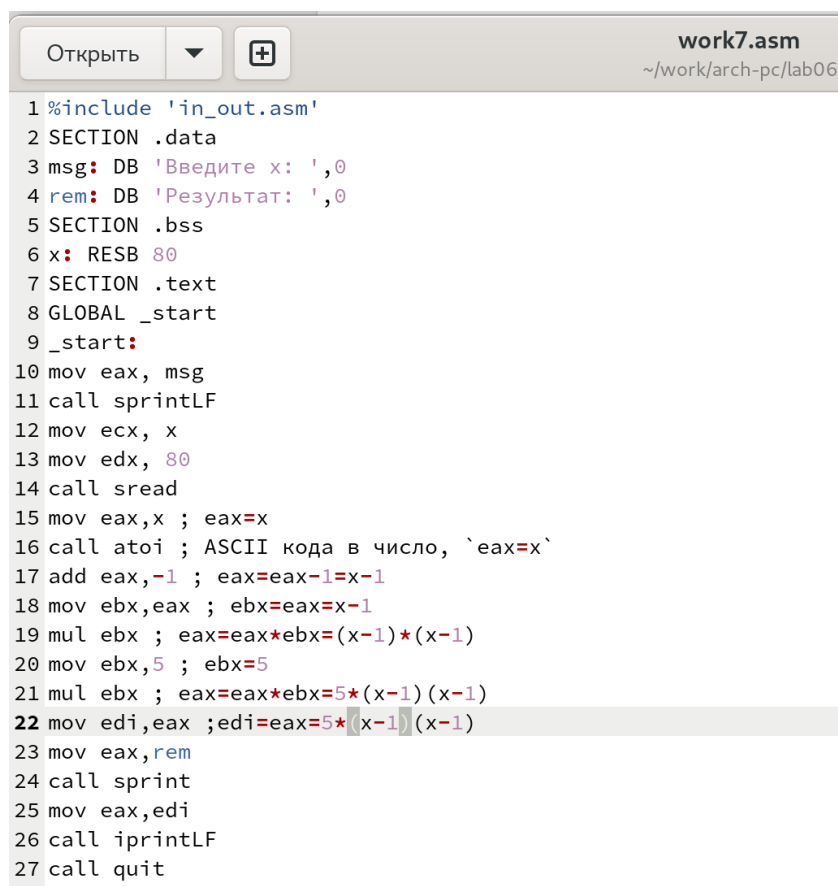
Создаю файл work7.asm и открываю его с помощью редактора gedit (рис. 4.22).



```
[igfilatov@igfilatov lab06]$ touch ~/work/arch-pc/lab06/work7.asm  
[igfilatov@igfilatov lab06]$ gedit work7.asm
```

Рис. 4.22: Создание файла

Пишу программу которая вычисляет значение функции из варианта 7 (рис. 4.23).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintLF
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax,x ; eax=x
16 call atoi ; ASCII кода в число, `eax=x`
17 add eax,-1 ; eax=eax-1=x-1
18 mov ebx,eax ; ebx=eax=x-1
19 mul ebx ; eax=eax*ebx=(x-1)*(x-1)
20 mov ebx,5 ; ebx=5
21 mul ebx ; eax=eax*ebx=5*(x-1)*(x-1)
22 mov edi,eax ;edi=eax=5*(x-1)*(x-1)
23 mov eax,rem
24 call sprint
25 mov eax,edi
26 call iprintLF
27 call quit
```

Рис. 4.23: Написание программы

Текст программы:

```
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите x:',0
rem: DB 'Результат:',0

SECTION .bss
x: RESB 80

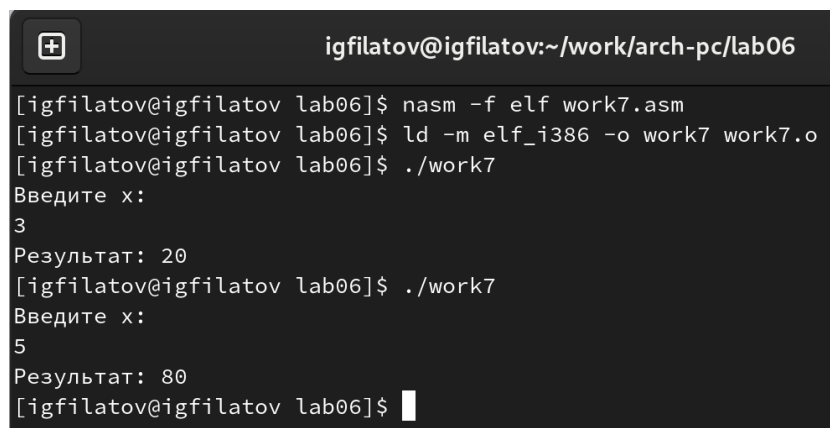
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
```

```

mov ecx, x
mov edx, 80
call sread
mov eax, x ; eax=x
call atoi ; ASCII кода в число, eax=x
add eax, -1 ; eax=eax-1=x-1
mov ebx, eax ; ebx=eax=x-1
mul ebx ; eax=eax*ebx=(x-1)(x-1)
mov ebx, 5 ; ebx=5
mul ebx ; eax=eax*ebx=5(x-1)(x-1)
mov edi, eax ; edi=eax=5*(x-1)(x-1)
mov eax, rem
call sprint
mov eax, edi
call iprintLF
call quit

```

Создаю и запускаю исполняемый файл. Проверяю программу на предложенных значениях аргумента – 3 и 5. В обоих случаях программа выводит верный результат (рис. 4.24).



```

igfilatov@igfilatov:~/work/arch-pc/lab06
[igfilatov@igfilatov lab06]$ nasm -f elf work7.asm
[igfilatov@igfilatov lab06]$ ld -m elf_i386 -o work7 work7.o
[igfilatov@igfilatov lab06]$ ./work7
Введите x:
3
Результат: 20
[igfilatov@igfilatov lab06]$ ./work7
Введите x:
5
Результат: 80
[igfilatov@igfilatov lab06]$

```

Рис. 4.24: Создание и запуск исполняемого файла

5 Выводы

Я освоил арифметические инструкции языка ассемблера NASM.

6 Список литературы

1. Архитектура ЭВМ