

Шаблон отчёта по лабораторной работе №7

Дисциплина: архитектура компьютера

Пронякова Ольга Максимовна

Содержание

| | | |
|----------|---|-----------|
| 1 | Цель работы | 5 |
| 2 | Задание | 6 |
| 3 | Теоретическое введение | 7 |
| 4 | Выполнение лабораторной работы | 9 |
| 4.1 | Символьные и численные данные в NASM | 9 |
| 4.2 | Выполнение арифметических операций в NASM | 20 |
| 4.3 | Ответы на вопросы по программе | 26 |
| 4.4 | Выполнение заданий для самостоятельной работы | 27 |
| 5 | Выводы | 30 |
| | Список литературы | 31 |

Список иллюстраций

| | | |
|------|--------------------------------|----|
| 4.1 | Создание директории | 9 |
| 4.2 | Создание файла | 9 |
| 4.3 | Копирование файла | 10 |
| 4.4 | Редактирование файла | 11 |
| 4.5 | Исполнение файла | 12 |
| 4.6 | Редактирование файла | 13 |
| 4.7 | Исполнение файла | 14 |
| 4.8 | Создание файла | 14 |
| 4.9 | Редактирование файла | 15 |
| 4.10 | Исполнение файла | 16 |
| 4.11 | Редактирование файла | 17 |
| 4.12 | Исполнение файла | 18 |
| 4.13 | Редактирование файла | 19 |
| 4.14 | Исполнение файла | 20 |
| 4.15 | Создание файла | 20 |
| 4.16 | Редактирование файла | 21 |
| 4.17 | Исполнение файла | 22 |
| 4.18 | Редактирование файла | 23 |
| 4.19 | Исполнение файла | 24 |
| 4.20 | Создание файла | 24 |
| 4.21 | Редактирование файла | 25 |
| 4.22 | Исполнение файла | 26 |
| 4.23 | Создание файла | 27 |
| 4.24 | Редактирование файла | 28 |
| 4.25 | Исполнение файла | 29 |

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Ответы на вопросы по программе
4. 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. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации:

- 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 символическом 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. Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной (см. Приложение.), а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться. Среди инструкций NASM нет такой, которая выводит числа (не в символическом 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. 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 регистр `eax` 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 записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax, ...`).

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

С помощью `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7 (рис. 4.1).

```
ила или каталога
olga@olga-VirtualBox:~$ mkdir ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/lab07
olga@olga-VirtualBox:~$ cd ~/work/study/2022-2023/"Архитектура компьютера"/arch-pc/lab07
olga@olga-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.1: Создание директории

С помощью `touch` создаю файл `lab7-1.asm` (рис. 4.2).

```
olga@olga-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
olga@olga-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07$ ls
lab7-1.asm
olga@olga-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.2: Создание файла

С помощью `cp` копирую в текущий каталог файл `in_out.asm`, так как он будет использоваться в других программах (рис. 4.3).

```
olga@olga-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07  
$ cp ~/Загрузки/in_out.asm in_out.asm  
olga@olga-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07  
$ ls  
in_out.asm  lab7-1.asm  
olga@olga-VirtualBox:~/work/study/2022-2023/Архитектура компьютера/arch-pc/lab07  
$
```

Рис. 4.3: Копирование файла

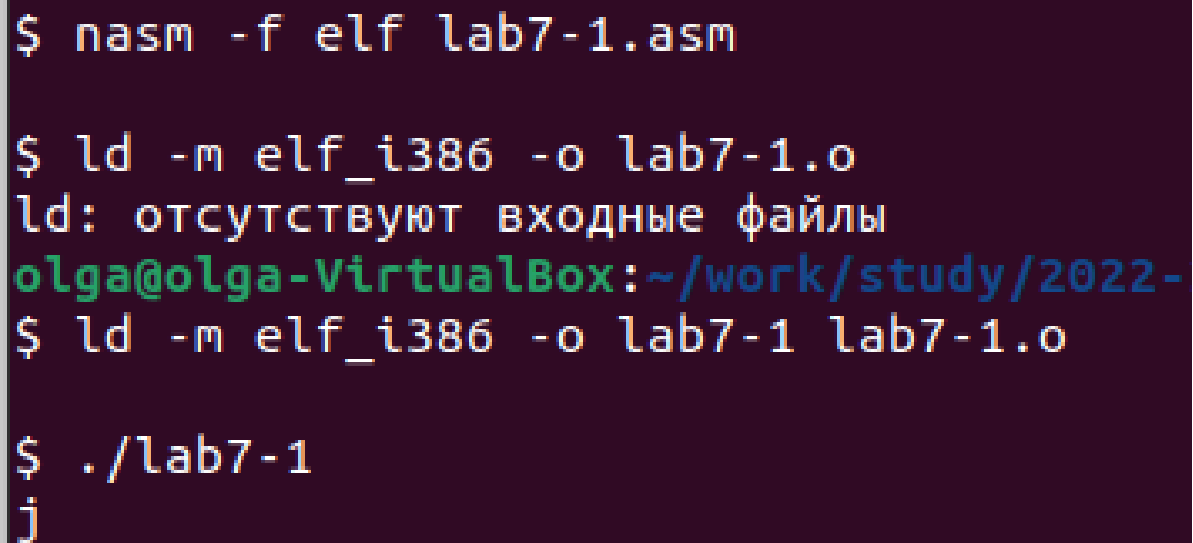
Открываю файл lab7-1.asm и вставляю в него программу вывода значения регистра eax (рис. 4.4).

```
/home/ol~b7-1.asm [---  
%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. 4.4: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла. Программа выводит j, потому что программа вывела символ,

соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6 (рис. 4.5).



```
$ nasm -f elf lab7-1.asm

$ ld -m elf_i386 -o lab7-1.o
ld: отсутствуют входные файлы
olga@olga-VirtualBox:~/work/study/2022-
$ ld -m elf_i386 -o lab7-1 lab7-1.o

$ ./lab7-1
j
```

Рис. 4.5: Исполнение файла

Изменяю в программе символы “6” и “4” на цифры 6 и 4 (рис. 4.6).

```
/home/00-07-1.asm  
%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. 4.6: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла. Теперь программа выводит символ с кодом 10, символ перевода строки, он не отображается при выводе на экран (рис. 4.7).

```
olga@olga-VirtualBox:~/work/study/2022-2023/Архитекту
$
nasm -f elf lab7-1.asm
olga@olga-VirtualBox:~/work/study/2022-2023/Архитекту
$ ld -m elf_i386 -o lab7-1 lab7-1.o
olga@olga-VirtualBox:~/work/study/2022-2023/Архитекту
$ ./lab7-1
$
```

Рис. 4.7: Исполнение файла

С помощью touch создаю файл lab7-2.asm (рис. 4.8).

```
Совет: Вы сможете видеть скрытые
$ touch lab7-2.asm
1Помощь 2Меню 3Про~тр 4Правка
```

Рис. 4.8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. 4.9).

```
/home/ol~b7-2.asm [ - - - - ] 9
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

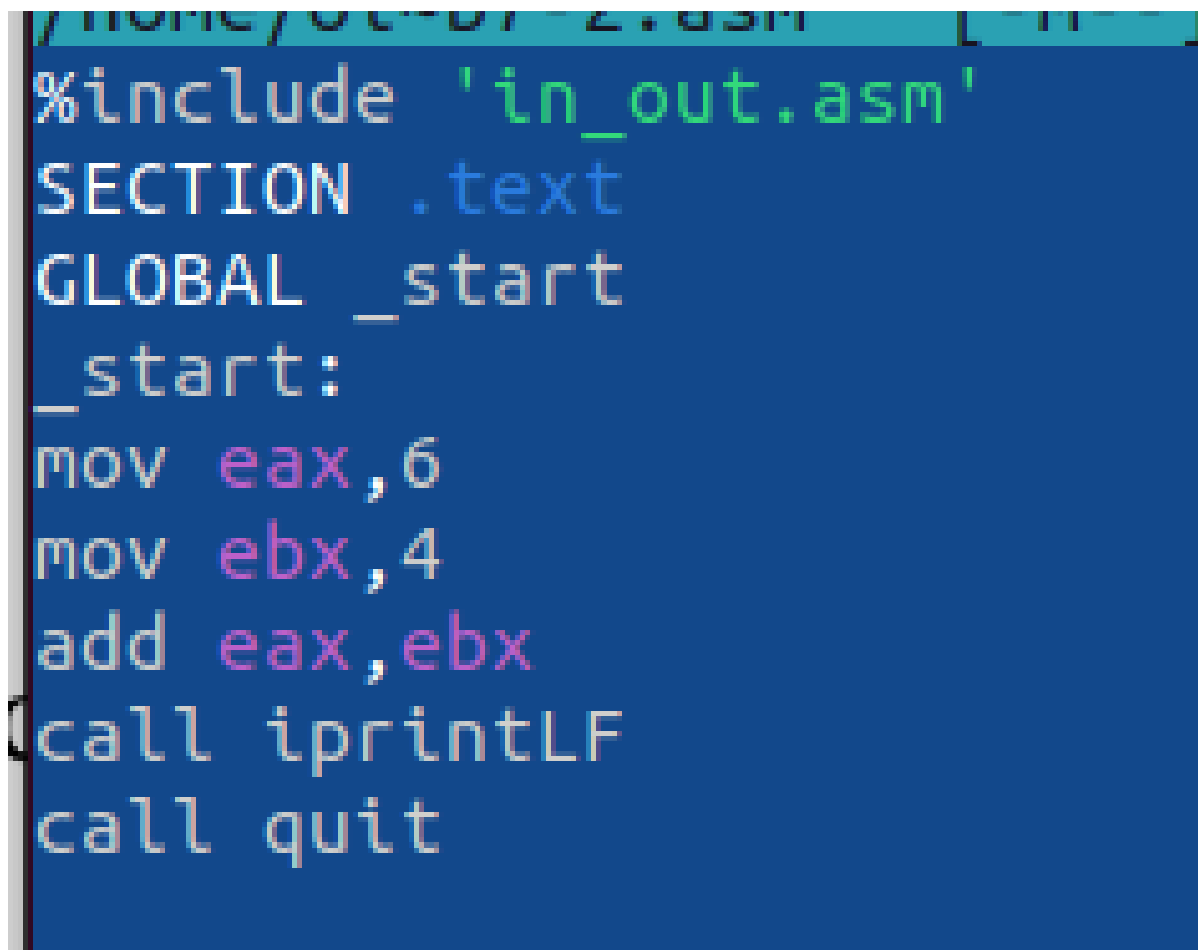
Рис. 4.9: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла. Теперь выводит число 106, потому что программа позволяет вывести именно число, а не символ (рис. 4.10).

```
$ nasm -f elf lab7-1.asm
$ nasm -f elf lab7-2.asm
$ ld -m elf_i386 -o lab7-2 lab7-2.o
$ ./lab7-2
106
$
```

Рис. 4.10: Исполнение файла

Изменяю в программе символы “6” и “4” на цифры 6 и 4 (рис. 4.11).

A screenshot of a text editor window with a dark blue background. The text is written in a monospaced font with syntax highlighting. The code is assembly language. The first line is a comment: `; home/00-01-2.asm`. The second line is `%include 'in_out.asm'`. The third line is `SECTION .text`. The fourth line is `GLOBAL _start`. The fifth line is `_start:`. The sixth line is `mov eax,6`. The seventh line is `mov ebx,4`. The eighth line is `add eax,ebx`. The ninth line is `call iprintLF`. The tenth line is `call quit`.

```
; home/00-01-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

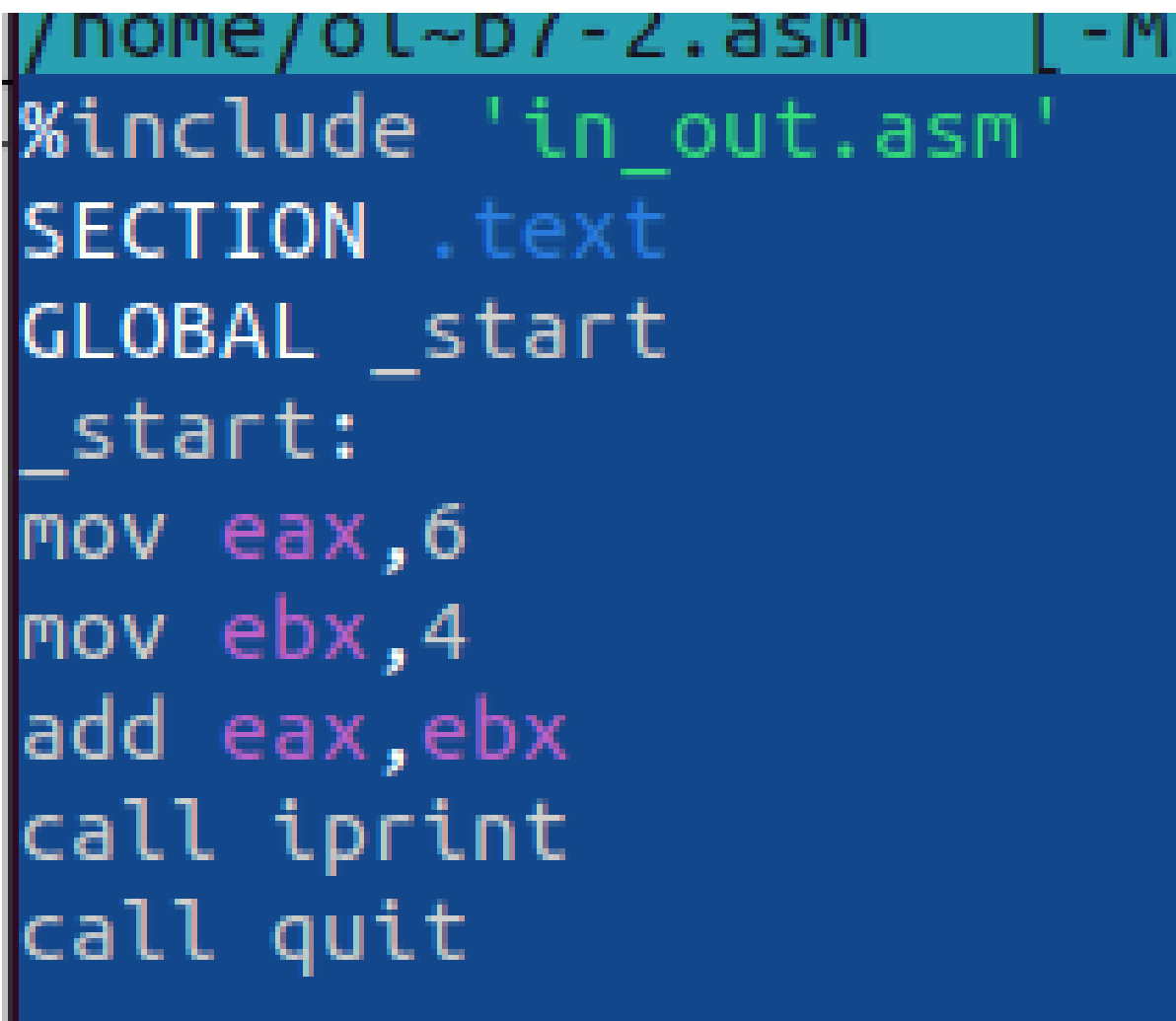
Рис. 4.11: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому выводит 10 (рис. 4.12).

```
olga@olga-VirtualBox:~/work/study/2021-01-07$  
$  
nasm -f elf lab7-2.asm  
$ ld -m elf_i386 -o lab7-2 lab7-2.o  
$ ./lab7-2  
10  
$
```

Рис. 4.12: Исполнение файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 4.13).



```
/home/ol~D/ - 2.asm [ - M
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 4.13: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла. Вывод не изменился, так как символ переноса строки не отобразился, когда программа исполнялась с функцией `iprintLF`, а `iprint` не доавляет к выводу символ переноса строки (рис. 4.14).

```

olga@olga-VirtualBox:~/work/study/202
$
nasm -f elf lab7-2.asm

$ ld -m elf_i386 -o lab7-2 lab7-2.o

$ ./lab7-2
10
$

```

Рис. 4.14: Исполнение файла

4.2 Выполнение арифметических операций в NASM

С помощью touch создаю файл lab7-3.asm (рис. 4.15).

```

25G/49G (50%)
Совет: Вы сможете видеть скрытые файлы
$ touch lab7-3.asm
1Помощь 2Меню 3Про~тр 4Правка 5Копи

```

Рис. 4.15: Создание файла

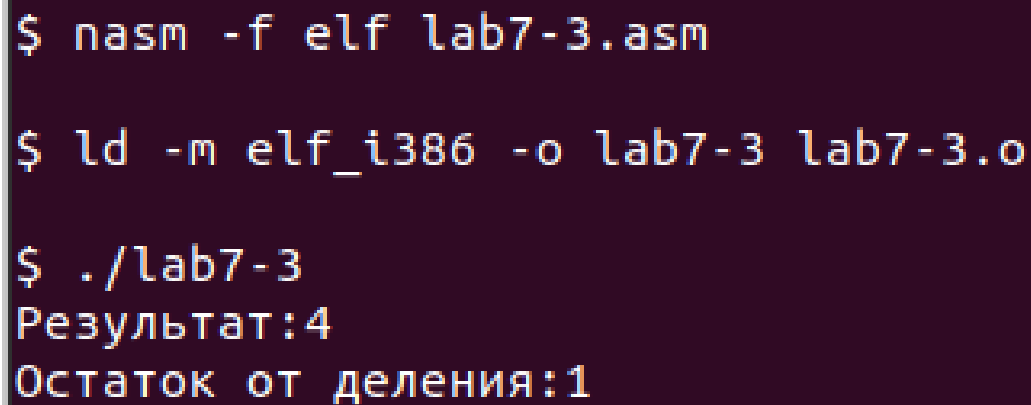
Ввожу в файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.16).

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат:',0
rem: DB 'Остаток от деления:',0
SECTION .text
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
```

1Помощь 2Сохранить 3Блок 4Замена 5Копи

Рис. 4.16: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла (рис. 4.17).

A terminal window with a dark purple background and light blue text. It shows three commands being executed: 'nasm -f elf lab7-3.asm', 'ld -m elf_i386 -o lab7-3 lab7-3.o', and './lab7-3'. The output of the third command is 'Результат:4' and 'Остаток от деления:1'.

```
$ nasm -f elf lab7-3.asm
$ ld -m elf_i386 -o lab7-3 lab7-3.o
$ ./lab7-3
Результат:4
Остаток от деления:1
```

Рис. 4.17: Исполнение файла

Изменяю программу так, чтобы она вычисляла выражение $f(x) = (4 * 6 + 2)/5$ (рис. 4.18).

```
/home/ol~b7-3.asm  [-M--]  9  L:[ 1.
%include 'in_out.asm'
SECTION .data
div: DB 'Результат:',0
rem: DB 'Остаток от деления:',0
SECTION .text
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
```

1 Помощь 2 Сохранить 3 Блок 4 Замена 5 Команда

Рис. 4.18: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла (рис. 4.19).

```
$ nasm -f elf lab7-3.asm  
  
$ ld -m elf_i386 -o lab7-3 lab7-3.o  
  
$ ./lab7-3  
Результат:5  
Остаток от деления:1
```

Рис. 4.19: Исполнение файла

С помощью touch создаю файл variant.asm (рис. 4.20).

```
Совет: Вы сможете видеть скрытые файлы  
$ touch variant.asm
```

1Помощь 2Меню 3Про~тр 4Правка 5Копи

Рис. 4.20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.21).


```

/home/ol~tant.asm [-M--] 9 L: [ 1+14 15/
#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
call atoi
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Пер

```

Рис. 4.21: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла. Ввожу номер студенческого билета, программа вывела мой вариант-14 (рис. 4.22).

```
$ nasm -f elf variant.asm

$ ld -m elf_i386 -o variant variant.o

$ ./variant
Введите № студенческого билета:
1132226453
Ваш вариант: 14

$
```

Рис. 4.22: Исполнение файла

4.3 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки: `mov eax,rem call sprint`
2. `mov ecx` - 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, обеспечивающий ввод сообщения с клавиатуры
3. `call atoi` - вызов 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 mov ebx,20 div ebx inc edx`
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 результатов вычислений отвечают строки: `mov eax,edx call iprintLF`

4.4 Выполнение заданий для самостоятельной работы

С помощью touch создаю файл lab7-4.asm (рис. 4.23).

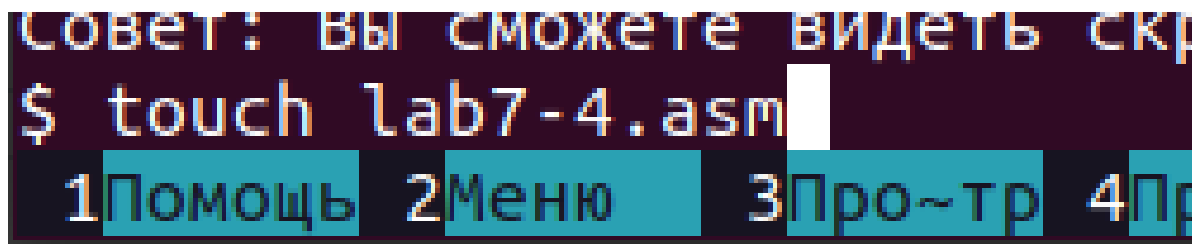


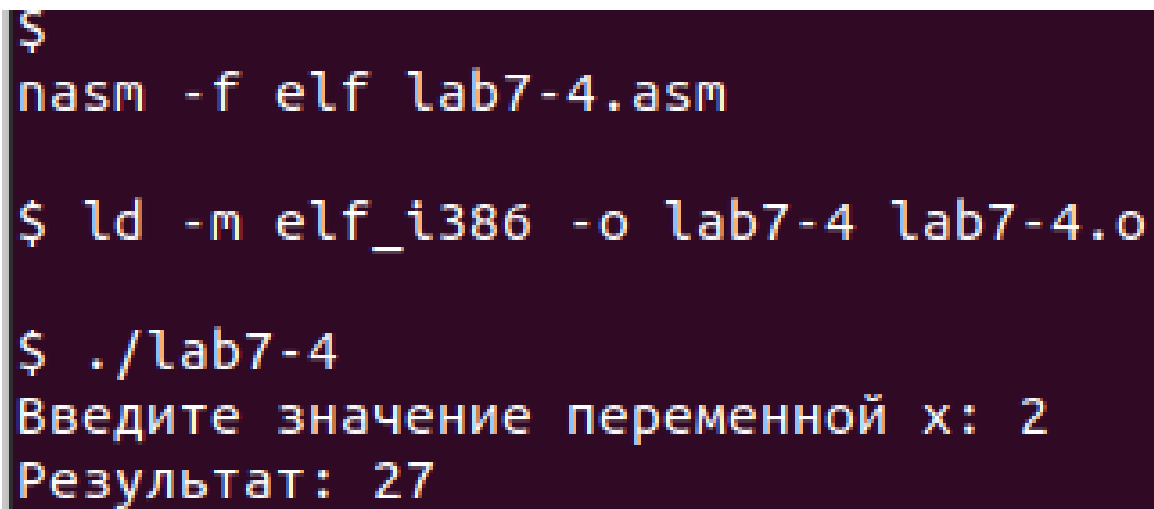
Рис. 4.23: Создание файла

Открываю файл и ввожу в него текст программы для вычисления значения выражения 14 варианта $(x/2 + 8) * 3$ (рис. 4.24).

```
/home/olga/work/study/2~ch-pc/lab07/lab7-4.a:  
%include 'in_out.asm'  
SECTION .data  
rem: DB 'Введите значение переменной x: ',0  
div: DB 'Результат: ',0  
SECTION .bss  
x: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,rem  
call sprint  
mov ecx, x  
mov edx,80  
call sread  
mov eax,x  
call atoi  
xor edx,edx  
mov ebx,2  
div ebx  
add eax,8  
mov ebx,3  
mul ebx  
mov edi,eax  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
call quit
```

Рис. 4.24: Редактирование файла

Транслирую текст программы файла в объектный файл. Выполняю компоновку объектного файла (рис. 4.25).

A terminal window with a dark background and light-colored text. The text shows a sequence of commands: a shell prompt '\$', followed by 'nasm -f elf lab7-4.asm', another prompt '\$', then 'ld -m elf_i386 -o lab7-4 lab7-4.o', and a third prompt '\$' followed by './lab7-4'. Below the commands, there are two lines of output: 'Введите значение переменной x: 2' and 'Результат: 27'.

```
$  
nasm -f elf lab7-4.asm  
  
$ ld -m elf_i386 -o lab7-4 lab7-4.o  
  
$ ./lab7-4  
Введите значение переменной x: 2  
Результат: 27
```

Рис. 4.25: Исполнение файла

5 Выводы

Я освоила арифметические инструкции языка ассемблера NASM.

Список литературы

1. Архитектура ЭВМ