

# **Отчет по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

Ицков Андрей Станиславович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Символьные и численные данные в NASM . . . . .	8
4.2	Выполнение арифметических операций в NASM . . . . .	10
4.3	Выполнение заданий для самостоятельной работы . . . . .	12
4.3.1	Ответы на вопросы по программе . . . . .	13
<b>5</b>	<b>Выводы</b>	<b>15</b>
<b>6</b>	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

4.1	Создание директории и файла . . . . .	8
4.2	Текст программы . . . . .	8
4.3	Запуск исполняемого файла . . . . .	8
4.4	Редактирование файла . . . . .	9
4.5	Запуск исполняемого файла . . . . .	9
4.6	Текст программы . . . . .	9
4.7	Запуск исполняемого файла . . . . .	9
4.8	Создание файла . . . . .	10
4.9	Редактирование файла . . . . .	10
4.10	Запуск исполняемого файла . . . . .	10
4.11	Создание файла . . . . .	10
4.12	Текст программы . . . . .	11
4.13	Запуск исполняемого файла . . . . .	11
4.14	Редактирование файла . . . . .	11
4.15	Запуск файла . . . . .	12
4.16	Редактирование файла . . . . .	12
4.17	Запуск исполняемого файла . . . . .	12
4.18	Текст программы . . . . .	13
4.19	Ответы . . . . .	13
4.20	Ответы . . . . .	13

# 1 Цель работы

Цель данной лабораторной работы - освоение арифметических 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, перемещаюсь туда и создаю там файл `lab6-1.asm` (рис. 4.1).

Создание директории и файла

Рис. 4.1: Создание директории и файла

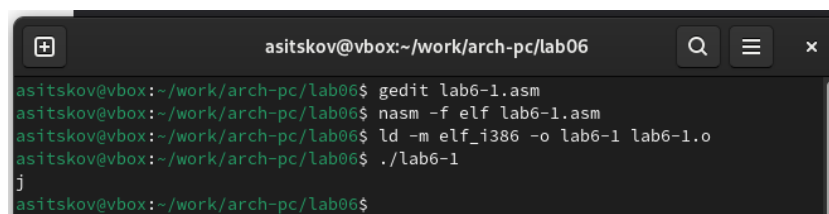
Вставляю в созданный файл данную мне программу (рис. 4.2).



```
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: Текст программы

Создаю исполняемый файл программы и запускаю его (рис. 4.3).



```
asitskov@vbox:~/work/arch-pc/lab06$ gedit lab6-1.asm
asitskov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
asitskov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
asitskov@vbox:~/work/arch-pc/lab06$ ./lab6-1
j
asitskov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.3: Запуск исполняемого файла




Ввожу в текст программы запрашиваемые изменения (рис. 4.4).



```
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. 4.4: Редактирование файла

Снова создаю исполняемый файл программы и запускаю его (рис. 4.5).



```
asitskov@vbox:~/work/arch-pc/lab06$ gedit lab6-1.asm
asitskov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
asitskov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
asitskov@vbox:~/work/arch-pc/lab06$ ./lab6-1

asitskov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.5: Запуск исполняемого файла

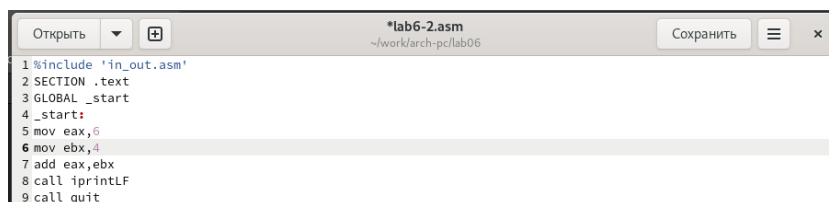
Создаю новый файл lab6-2.asm и ввожу туда текст новой программы (рис. 4.6).



```
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. 4.6: Текст программы

Редактирую текст программы (рис. 4.7).



```
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. 4.7: Запуск исполняемого файла

Создаю исполняемый файл программы и запускаю его (рис. 4.8).

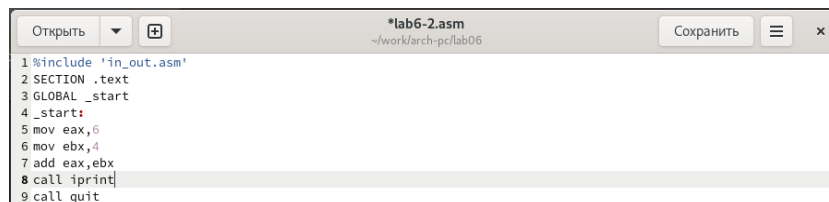
```

asitskov@vbox:~/work/arch-pc/lab06$ gedit lab6-2.asm
asitskov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asitskov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
asitskov@vbox:~/work/arch-pc/lab06$ ./lab6-2
10
asitskov@vbox:~/work/arch-pc/lab06$

```

Рис. 4.8: Создание файла

Заменяю `iprintLF` на `iprint` в тексте программы (рис. 4.9).



```

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. 4.9: Редактирование файла

Снова создаю и запускаю исполняемый файл `lab6-2` (рис. 4.10).

```

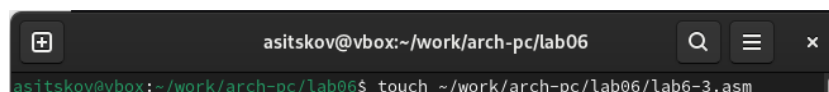
asitskov@vbox:~/work/arch-pc/lab06$ gedit lab6-2.asm
asitskov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
asitskov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
asitskov@vbox:~/work/arch-pc/lab06$ ./lab6-2
10asitskov@vbox:~/work/arch-pc/lab06$

```

Рис. 4.10: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю пустой файл `lab6-3.asm` (рис. 4.11).



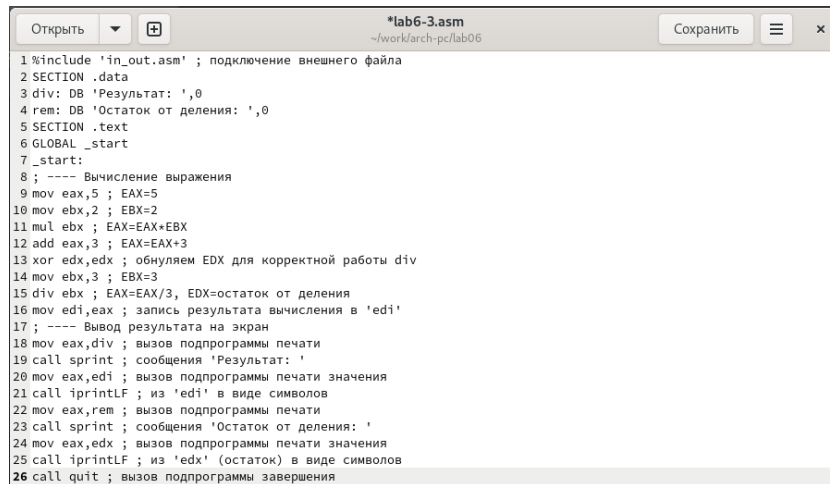
```

asitskov@vbox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm

```

Рис. 4.11: Создание файла

Ввожу туда данную мне программу (рис. 4.12).



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

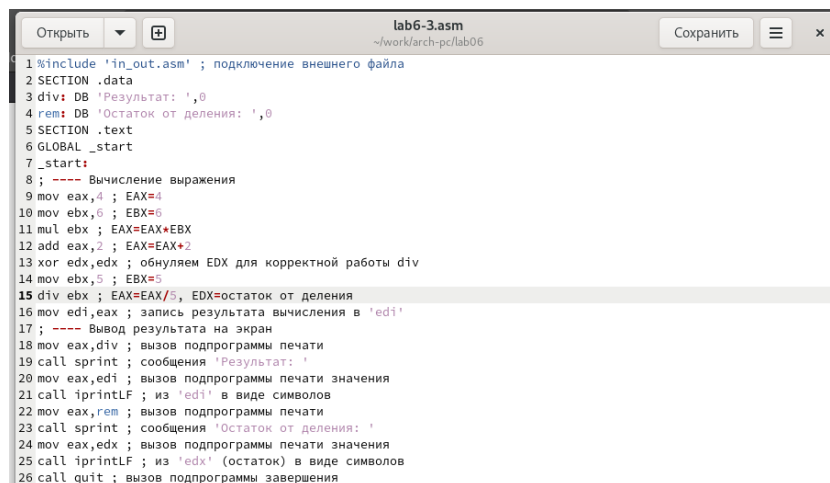
Рис. 4.12: Текст программы

Запускаю программы и получаю результат выражения (рис. 4.13).

Запуск исполняемого файла

Рис. 4.13: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$  (рис. 4.14).



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

Рис. 4.14: Редактирование файла

Запускаю программы и получаю новый результат (рис. 4.15).


```

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

```

Рис. 4.15: Запуск файла

Создаю файл `variant.asm` и ввожу туда данную мне программу (рис. 4.16).



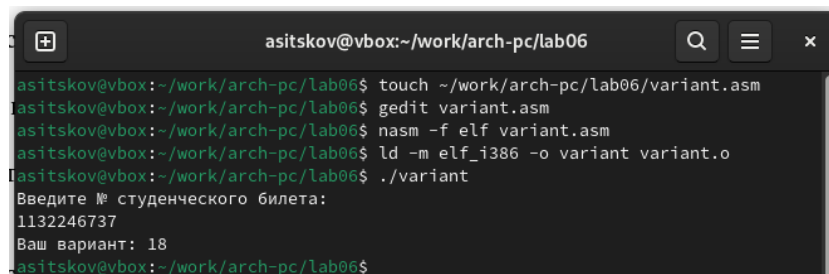
```

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

```

Рис. 4.16: Редактирование файла

Программа выдала мне номер моего варианта (рис. 4.17).



```

asitskov@vbox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
asitskov@vbox:~/work/arch-pc/lab06$ gedit variant.asm
asitskov@vbox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
asitskov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
asitskov@vbox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246737
Ваш вариант: 18
asitskov@vbox:~/work/arch-pc/lab06$

```

Рис. 4.17: Запуск исполняемого файла

## 4.3 Выполнение заданий для самостоятельной работы

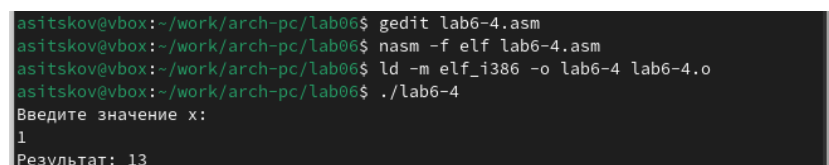
Решаю задачу (рис. 4.18).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите значение 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 sprintf
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, 'eax=x'
17 add eax, 10
18 mov ebx, 3
19 mul ebx
20 add eax, -20
21 mov edi, eax
22 mov eax, rem
23 call sprintf
24 mov eax, edi
25 call iprintLF
26 call quit
```

Рис. 4.18: Текст программы

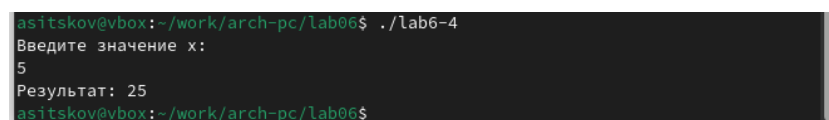
Получаю ответы (рис. 4.19).



```
asitskov@vbox:~/work/arch-pc/lab06$ gedit lab6-4.asm
asitskov@vbox:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
asitskov@vbox:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
asitskov@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение x:
1
Результат: 13
```

Рис. 4.19: Ответы

Создаю файл variant.asm с помощью утилиты touch (рис. 4.20).



```
asitskov@vbox:~/work/arch-pc/lab06$ ./lab6-4
Введите значение x:
5
Результат: 25
asitskov@vbox:~/work/arch-pc/lab06$
```

Рис. 4.20: Ответы

### 4.3.1 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
```

## 5 Выводы

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

## **6 Список литературы**

1. Лабораторная работа №7
2. Таблица ASCII