

# **Отчёт по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

**Жернаков Данила Иванович**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Символьные и численные данные в NASM . . . . .	9
4.2	Выполнение арифметических операций в NASM . . . . .	11
4.2.1	Ответы на вопросы по программе . . . . .	13
4.3	Выполнение заданий для самостоятельной работы . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

4.1	Создание директории и файла в ней . . . . .	9
4.2	Редактирование файла . . . . .	9
4.3	Запуск исполняемого файла . . . . .	10
4.4	Редактирование файла . . . . .	10
4.5	Запуск исполняемого файла . . . . .	10
4.6	Запуск исполняемого файла . . . . .	11
4.7	Запуск исполняемого файла . . . . .	12
4.8	Запуск исполняемого файла . . . . .	12
4.9	Запуск исполняемого файла . . . . .	13
4.10	Запуск исполняемого файла . . . . .	14

## **Список таблиц**

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

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc $ cd labsasm/
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm $ mkdir lab06
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm $ cd lab06
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ touch lab6-1.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $
```

Рис. 4.1: Создание директории и файла в ней

Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.2).

```
lab6-1.asm [-----] 9 L: [ 1+12 13/ 13] +({172 / 172b) <EOF> [X] [X]
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: Редактирование файла

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах. Создаю исполняемый файл программы

и запускаю его (рис. 4.3). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ nasm -f elf lab6-1.asm
lab6-1.asm:1: error: unable to open include file 'in_out.asm': No such file or directory
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ cp ~/Зарядки/in_out.asm in_out.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ nasm -f elf lab6-1.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ./lab6-1
j
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $
```

Рис. 4.3: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (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

Создаю новый исполняемый файл программы и запускаю его (рис. ??). 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”.

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ nasm -f elf lab6-2.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ./lab6-2
106
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $
```

Создаю новый файл lab6-2.asm с помощью утилиты touch. Ввожу в файл текст другой программы для вывода значения регистра eax. Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4. Создаю и запускаю новый исполняемый файл (рис. 4.5). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ nasm -f elf lab6-2.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ./lab6-2
10
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $
```

Рис. 4.5: Запуск исполняемого файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` Создаю и запускаю новый исполняемый файл (рис. 4.6). Вывод изменился, потому что символ переноса строки “отображался”, когда программа исполнялась с функцией `iprintLF`, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ nasm -f elf lab6-2.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ./lab6-2
10dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $
```

Рис. 4.6: Запуск исполняемого файла

## 4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch`. Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$  (рис. ??).

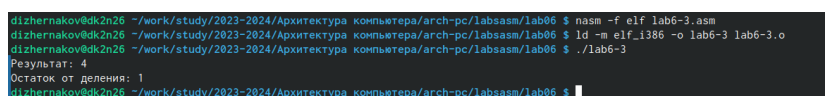
```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
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 исполняемый файл и запускаю его (рис. 4.7).



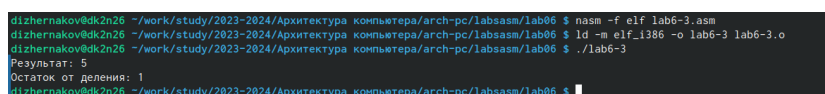
```

dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ nasm -f elf lab6-3.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $

```

Рис. 4.7: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4x^6 + 2)/5$ . (рис. ??). Создаю и запускаю новый исполняемый файл (рис. 4.8). Для проверки правильности работы программы я сам посчитал значение выражения, программа отработала верно.



```

dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ nasm -f elf lab6-3.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $

```

Рис. 4.8: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch и ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета. Создаю и запускаю исполняемый файл (рис. 4.9). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 16.

```
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ touch variant.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ nasm -f elf variant.asm
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ld -m elf_i386 -o variant variant.o
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $ ./variant
Введите № студенческого билета:
1332236095
Ваш вариант: 16
dizhernakov@dk2n26 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labsasm/lab06 $
```

Рис. 4.9: 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 lab6-4.asm с помощью утилиты touch и открываю его для редактирования, ввожу в него текст программы для вычисления значения выражения  $(10x - 5)^2$  (рис. ??). Это выражение из варианта 20.

Создаю и запускаю исполняемый файл(рис. 4.10). При вводе значения 3, вывод 625. Тут же провожу ещё один запуск исполняемого файла для проверки работы программы с другим значением на входе. Программа отработала верно.

Запуск исполняемого файла

Рис. 4.10: Запуск исполняемого файла

**\*\*Листинг 4.1. Программа для вычисления значения выражения  $(x^3)^*(1/3)+21$ .\*\***

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprint
mov ecx, x
```

```

mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
mov ebx, 10 ;
mul ebx ; x*10
add eax,-5 ; (10*x)-5
mul eax ; умножаем на себя
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,tem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

```

## **5 Выводы**

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