

# **Отчет по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

**Швецов Михаил Романович**

# 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>8</b>
<b>5</b>	<b>Выводы</b>	<b>16</b>
	<b>Список литературы</b>	<b>17</b>

## Список иллюстраций

4.1	Создание директории и файла . . . . .	8
4.2	Редактирование файла . . . . .	8
4.3	копирование файла in_out.asm . . . . .	9
4.4	Создание исполняемого файла . . . . .	9
4.5	Редактирование файла . . . . .	9
4.6	Создание директории и файла . . . . .	10
4.7	Редактирование файла . . . . .	10
4.8	Запуск программы . . . . .	11
4.9	Запуск программы . . . . .	11
4.10	Запуск программы . . . . .	11
4.11	Запуск программы . . . . .	12
4.12	Запуск программы . . . . .	12
4.13	Написание программы . . . . .	13
4.14	Запуск кода . . . . .	14
4.15	Запуск кода с другими данными . . . . .	14

# Список таблиц

3.1	Описание некоторых каталогов файловой системы GNU Linux . .	7
-----	---	---

# 1 Цель работы

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

## 2 Задание

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

### 3 Теоретическое введение

Здесь описываются теоретические аспекты, связанные с выполнением работы. Например, в табл. 3.1 приведено краткое описание стандартных каталогов Unix.

Таблица 3.1: Описание некоторых каталогов файловой системы GNU Linux

Имя каталога	Описание каталога
/	Корневая директория, содержащая всю файловую
/bin	Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям
/etc	Общесистемные конфигурационные файлы и файлы конфигурации установленных программ
/home	Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя
/media	Точки монтирования для сменных носителей
/root	Домашняя директория пользователя root
/tmp	Временные файлы
/usr	Вторичная иерархия для данных пользователя

Более подробно об Unix см. в [1–6].

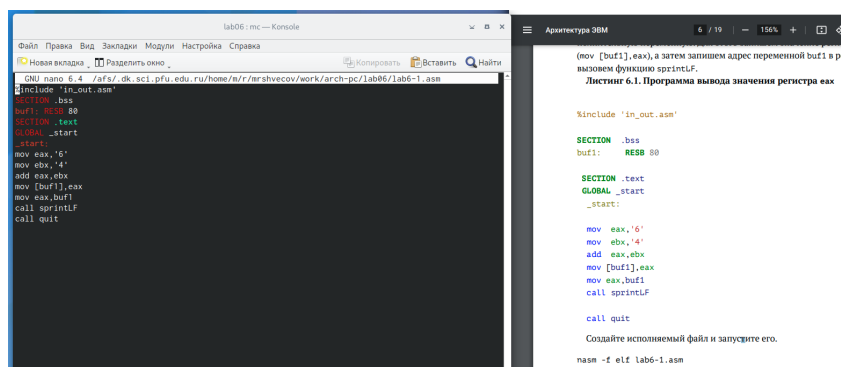
## 4 Выполнение лабораторной работы

Создал директорию, в которой буду создавать файлы с программами для лабораторной работы №6. Перехожу в созданный каталог с помощью команды `cd`. И создаю файл `lab6-1.asm`

```
mrshvecov@dk4n69 ~ $ mkdir ~/work/arch-pc/lab06
mrshvecov@dk4n69 ~ $ cd ~/work/arch-pc/lab06
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ touch lab6-1.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $
```

Рис. 4.1: Создание директории и файла

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



```
%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
```

```
%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
```

Создайте исполняемый файл и запустите его.

```
nasm -f elf lab6-1.asm
```

Рис. 4.2: Редактирование файла

Скопировал файл `in_out.asm` в директорию `lab06`



Левая панель	Файл	Команда	Настройки	Правая панель	Файл	Команда	Настройки
<-	~/work/arch-pc/lab05		.[*]>	<-	~/work/arch-pc/lab06		.[*]>
.и	Имя	Размер	Время правки	.и	Имя	Размер	Время правки
./..	-ВВЕРХ-	ноя 15 12:34		./..	-ВВЕРХ-	ноя 15 12:34	
in_out.asm	2042	ноя 8 12:40		in_out.asm	2042	ноя 8 12:40	
lab5-1.o	2744	ноя 8 13:10		lab5-1.o	2744	ноя 15 12:40	
lab5-1.asm	2005	ноя 8 13:14		lab5-1.asm	172	ноя 15 12:35	
lab5-1.o	752	ноя 8 13:14		lab6-1.o	1200	ноя 15 12:44	
lab5-1	2005	ноя 8 13:14					
lab5-2.asm	803	ноя 8 13:53					
lab5-2.o	1312	ноя 8 13:53					
lab5-3	2744	ноя 8 14:00					
lab5-3.asm	1742	ноя 8 14:00					
lab5-3.o	784	ноя 8 14:05					
lab5-4	2005	ноя 8 14:11					
lab5-4.asm	1450	ноя 8 14:10					
lab5-4.o	1344	ноя 8 14:11					

Рис. 4.3: копирование файла in\_out.asm

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

```
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 4.4: Создание исполняемого файла

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

```
lab6-1.asm [-M--] 9 L: [ 1+ 7 8/ 13] *(104 / 168b) 0010 0x00A [*][X]
%include "in_out.asm"
SECTION .text
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. 4.5: Редактирование файла

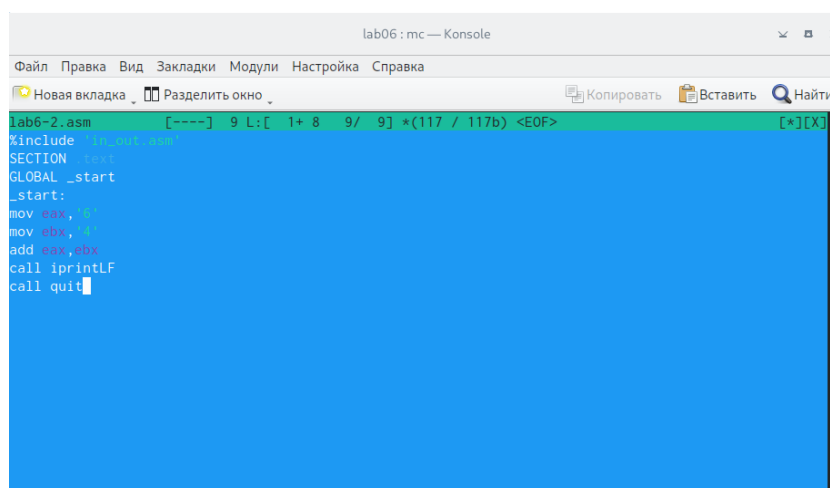
Создаю новый исполняемый файл программы и запускаю его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./lab6-1

mrshvecov@dk4n69 ~/work/arch-pc/lab06 $
```

Рис. 4.6: Создание директории и файла

Создал новый файл lab7-2.asm с помощью утилиты touch и ввожу в файл текст другой программы для вывода значения регистра eax



```
lab6-2.asm [----] 9 L: [ 1+ 8 9/ 9] *(117 / 117b) <EOF> [*][X]
#include "lab01.asm"
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add ebx, ebx
call iprintLF
call quit
```

Рис. 4.7: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2. 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”.

```

mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./lab6-2
106
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $

```

Рис. 4.8: Запуск программы

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4. Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```

mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./lab6-2
10
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $

```

Рис. 4.9: Запуск программы

Создал файл lab6-3.asm с помощью утилиты touch. Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$ . Создаю исполняемый файл и запускаю его

```

mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $

```

Рис. 4.10: Запуск программы

Изменяю программу так, чтобы она вычисляла значение выражения  $f(x) = (4 * 6 + 2)/5$ , создаю и запускаю новый исполняемый файл (рис. 4.19). Я посчитал для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $

```

Рис. 4.11: Запуск программы

Создал файл `variant.asm` с помощью утилиты `touch`, ввел в файл текст программы для вычисления варианта задания по номеру студенческого билета. Создаю и запускаю исполняемый файл (рис. 4.22). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 1.

```

mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132236100
Ваш вариант: 1
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $

```

Рис. 4.12: Запуск программы

Ответы на вопросы:

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`

Задания для самостоятельной работы:

Создаю файл `lab6-4.asm` с помощью утилиты `touch`. Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения  $(10 + 2x)/3$ . Это выражение было под вариантом 1.

```

lab7-4.asm [----] 17 L: [ 1+21 22/ 34] *(1326/2079b) 0010 0x00A [*][X]
#include "asm.h" ; подключение внешнего файла
SECTION ".data" ; секция инициализированных данных
msg: DB "Введите значение переменной x: ",0
rem: DB "Результат: ",0
SECTION ".bss" ; секция не инициализированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт
SECTION ".text" ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ebx, x ; запись адреса переменной в ebx
mov edi, 80 ; запись длины вводимого значения в edi
call sread ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, eax=x
mov ebx, 2
mul ebx
add eax, 10
xor edx, edx ; обнуляем EDX для корректной работы div
mov ebx, 3 ; EBX=3
div ebx
mov edi, eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax, rem ; 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
  
```

Рис. 4.13: Написание программы

Создаю и запускаю исполняемый файл. При вводе значения 1, вывод - 4.

```

mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ nasm -f elf lab7-4.asm
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab7-4 lab7-4.o
mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./lab7-4
Введите значение переменной x: 1
Результат: 4

```

Рис. 4.14: Запуск кода

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе. Программа отработала верно.

```

mrshvecov@dk4n69 ~/work/arch-pc/lab06 $ ./lab7-4
Введите значение переменной x: 10
Результат: 10

```

Рис. 4.15: Запуск кода с другими данными

Листинг 4.1. Программа для вычисления выражения  $(10 + 2x)/3$ .

```

%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициированных данных
msg: DB 'Введите значение переменной x:',0
rem: DB 'Результат:',0
SECTION .bss ; секция не инициированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; -- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax,x ; вызов подпрограммы преобразования

```

```

call atoi ; ASCII кода в число, eax=x
mov ebx,2
mul ebx
add eax,10
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx
mov edi,eax ; запись результата вычисления в 'edi'
; -- Вывод результата на экран
mov eax,rem ; 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

```

## **5 Выводы**

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



## Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016.  
URL: <https://www.gnu.org/software/bash/manual/>.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.