

# **Отчет по лабораторной работе №6**

**Дисциплина: архитектура компьютера**

Авдеенко Марьяна Дмитриевна

# 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: символьные и численные данные в NASM</b>	<b>8</b>
<b>5</b>	<b>Выполнение лабораторной работы: выполнение арифметических операций в NASM</b>	<b>12</b>
<b>6</b>	<b>Оьветы на вопоросы</b>	<b>15</b>
<b>7</b>	<b>Выполнение заданий самостостоятельной работы</b>	<b>16</b>
<b>8</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

# Список иллюстраций

4.1	Создание каталога . . . . .	8
4.2	Программа вывода значения регистра <code>eax</code> . . . . .	8
4.3	Запуск исполняемого файла . . . . .	9
4.4	Замена строк программы . . . . .	9
4.5	Проверка файла . . . . .	9
4.6	Преобразование файла с применением функций . . . . .	10
4.7	Исполняемый файл с числом . . . . .	10
4.8	Исполняемый файл с числом . . . . .	10
4.9	Исполняемый файл с числом . . . . .	11
4.10	Замена функции <code>iprintLF</code> на <code>iprint</code> . . . . .	11
4.11	Выведение числа измененной функцией . . . . .	11
5.1	Применение листинга 6.3 . . . . .	12
5.2	Проверка файла . . . . .	13
5.3	Изменение программы . . . . .	13
5.4	Проверка файла . . . . .	13
5.5	Применение листинга 6.4 . . . . .	14
5.6	Проверка файла . . . . .	14
7.1	Программа файла <code>task.asm</code> . . . . .	16
7.2	Проверка файла <code>task.asm</code> . . . . .	17

## Список таблиц

# 1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

## **2 Задание**

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

### 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.

## 4 Выполнение лабораторной работы: СИМВОЛЬНЫЕ И ЧИСЛЕННЫЕ ДАННЫЕ В NASM

- 1) Создала каталог для программ лабораторной работы 6, перешла в него и создала файл lab6-1.asm (рис. 4.1).

```
mdavdeenko@dk2n22 ~/work/arch-pc $ mkdir lab06
mdavdeenko@dk2n22 ~/work/arch-pc $ cd lab06
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 4.1: Создание каталога

- 2) Ввела в файл lab6-1.asm текст программы из листинга 6.1. (рис. 4.2).

```
report:make lab06:mc
/afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab06/lab6-1.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. 4.2: Программа вывода значения регистра eax



- 3) Создала исполняемый файл и запустила его, в данном случае при выводе значения регистра `eax` я ожидала увидеть число 10, но результатом стал символ `j` (рис. 4.3).

```
mdavdeenko@dk2n22 ~/work/arch-pc/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
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 4.3: Запуск исполняемого файла

- 4) Далее изменила текст программы и вместо символов, записала в регистры числа. Ис- правила текст программы (Листинг 6.1), так как указано в задании (рис. 4.4).

```
/afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab06/lab6-1.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. 4.4: Замена строк программы

- 5) Создала исполняемый файл и запустила его (рис. 4.5).

```
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 4.5: Проверка файла

- 6) Преобразовала текст программы из Листинга 6.1 с использованием функций в файле in\_out.asm: создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы из листинга 6.2 (рис. 4.6).

```
/afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 4.6: Преобразование файла с применением функций

- 7) Создала исполняемый файл и запустила его, в результате работы программы я получила число 106 (рис. 4.7).

```
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ./lab6-2
106
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $
```

Рис. 4.7: Исполняемый файл с числом

- 8) Аналогично предыдущему примеру изменила символы на числа, заменив строки так, как указано в задании (рис. 4.8).

```
/afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.8: Исполняемый файл с числом

- 9) Создала исполняемый файл и запустила его (рис. 4.9).

```
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ./lab6-2
10
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $
```

Рис. 4.9: Исполняемый файл с числом

- 10) Заменяла функцию `iprintLF` на `iprint`. Создала исполняемый файл и запустила его. Различие функций заключается в аргументе функции `-LF`, которая задает написание результата с переносом на новую строку (рис. 4.10, 4.11).

```
/afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab06/lab6-2.asm
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

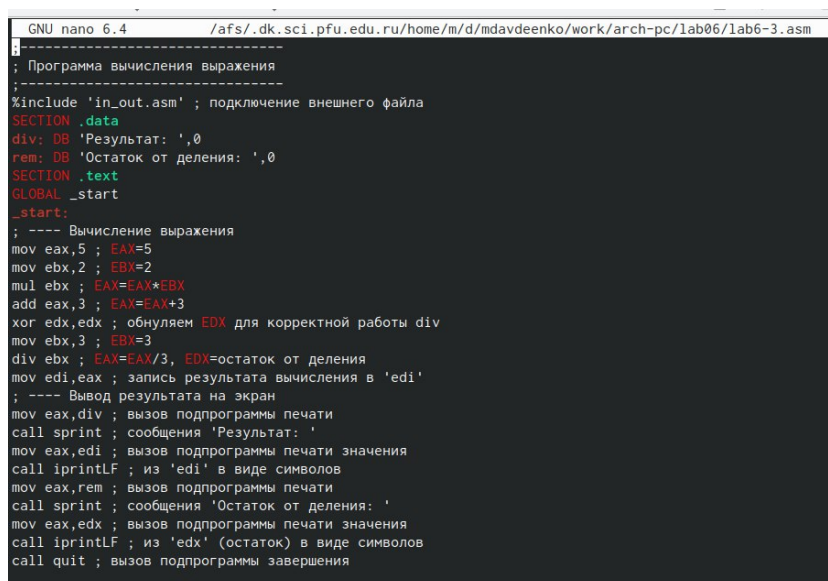
Рис. 4.10: Замена функции `iprintLF` на `iprint`

```
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $ ./lab6-2
10mdavdeenko@dk2n22 ~/work/arch-pc/lab06 $
```

Рис. 4.11: Выведение числа измененной функцией

## 5 Выполнение лабораторной работы: выполнение арифметических операций в NASM

- 11) В качестве примера выполнения арифметических операций в NASM привела программу вычисления арифметического выражения  $f(x) = (5 \times 2 + 3)/3$ . Создала файл lab6-3.asm в каталоге ~/work/arch-pc/lab06.
- 12) Изучила текст программы из листинга 6.3 и ввела его в lab6-3.asm (рис. 5.1).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab06/lab6-3.asm
;-----
; 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,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, 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 iprintfLF ; 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 iprintfLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 5.1: Применение листинга 6.3

- 13) Создала исполняемый файл и запустила его (рис. 5.2).

```
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $
```

Рис. 5.2: Проверка файла

- 14) Изменила текст программы для вычисления выражения  $f(x) = (4 \times 6 + 2)/5$  (рис. 5.3).

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/m/dmdavdeenko/work/arch-pc/lab06/lab6-3.asm
;-----
; 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, EDI=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 iprintf ; 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 iprintf ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 5.3: Изменение программы

- 15) Создала исполняемый файл и проверила его работу (рис. 5.4).

```
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $
```

Рис. 5.4: Проверка файла

- 16) В качестве другого примера рассмотрим программу вычисления варианта

задания по номеру студенческого билета, работающую по следующему алгоритму:

- вывести запрос на введение № студенческого билета
- вычислить номер варианта по формуле:  $(S_n \bmod 20) + 1$ , где  $S_n$  – номер студенческого билета (В данном случае  $a \bmod b$  – это остаток от деления  $a$  на  $b$ ).
- вывести на экран номер варианта. Для этого создала файл `variant.asm` в каталоге `~/work/arch-pc/lab06`, внимательно изучила текст программы из листинга 6.4 и ввела в файл `variant.asm` (рис. 5.5).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab06/variant.asm
;-----
; Программа вычисления варианта
;-----
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите No студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call read
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintf
call quit
```

Рис. 5.5: Применение листинга 6.4

17) Создала исполняемый файл и запустила его (рис. 5.6).

```
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
mdavdeenko@dk3n63 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236068
Ваш вариант: 9
```

Рис. 5.6: Проверка файла

## 6 Ответы на вопросы

- 1) За вывод сообщения “Ваш вариант” отвечают строки кода:

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

- 2) Инструкции `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx`; `mov edx, 80` - для записи в регистр `edx` длины вводимой строки; `call sread` - вызова подпрограммы из внешнего файла, обеспечивающего 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 вычисление варианта отвечают строки:

```
xor edx,edx ; обнуляем 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.

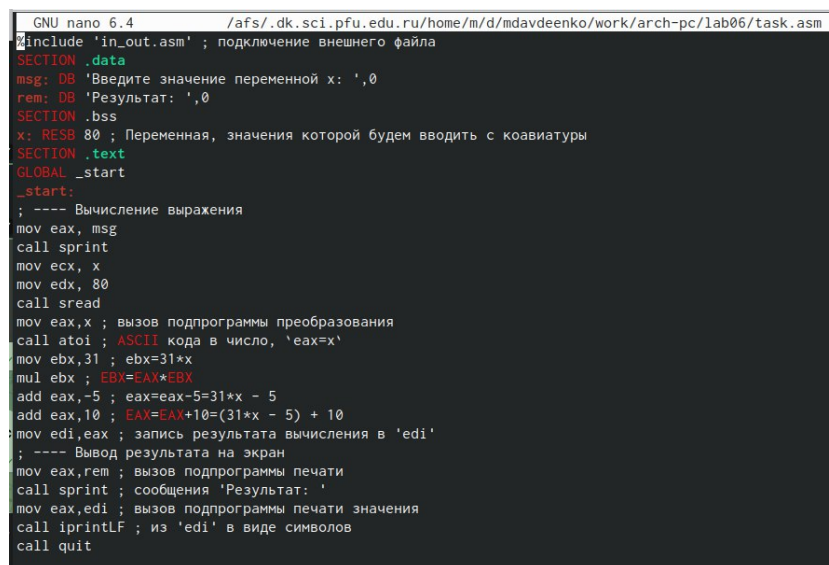
- 7) За вывод на экран результатов вычислений отвечают строки:

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

## 7 Выполнение заданий

### самостоятельной работы

- 1) Создала файл task.asm, открываю файл и ввожу в него текст программы для функции в 9 варианте (вариант, который был рассчитан на предыдущих шагах) (рис. 7.1).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/m/d/mdavdeenko/work/arch-pc/lab06/task.asm
#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, значения которой 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, 31 ; ebx=31*x
mul ebx ; EBX=EBX*EBX
add eax, -5 ; eax=eax-5=31*x - 5
add eax, 10 ; EAX=EAX+10=(31*x - 5) + 10
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 iprintf ; из 'edi' в виде символов
call quit
```

Рис. 7.1: Программа файла task.asm

- 2) Создала исполняемый файл и проверила его работу для значений  $x=3$  и  $x=1$ , программа вывела числовые значения, соответственно она работает верно (рис. 7.2).



```
mdavdeenko@dk3n65 ~/work/arch-pc/lab06 $ nasm -f elf task.asm
mdavdeenko@dk3n65 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o task task.o
mdavdeenko@dk3n65 ~/work/arch-pc/lab06 $ ./task
Введите значение переменной x: 3
Результат: 98
mdavdeenko@dk3n65 ~/work/arch-pc/lab06 $ ./task
Введите значение переменной x: 1
Результат: 36
mdavdeenko@dk3n65 ~/work/arch-pc/lab06 $
```

Рис. 7.2: Проверка файла task.asm

## 8 Выводы

В ходе данной лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.

## **Список литературы**