

# **Отчёт по лабораторной работе №6**

**Дисциплина: Архитектура компьютера**

Кириянова Екатерина Андреевна

# 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 . . . . .	11
4.3	Задание для самостоятельной работы . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>16</b>
<b>6</b>	<b>Список литературы</b>	<b>17</b>

# Список иллюстраций

4.1	Создание . . . . .	8
4.2	Программа . . . . .	8
4.3	Запуск . . . . .	8
4.4	Редактирование . . . . .	9
4.5	Запуск . . . . .	9
4.6	Новый файл . . . . .	9
4.7	Вторая программа . . . . .	9
4.8	Запуск программы . . . . .	10
4.9	Редактирование . . . . .	10
4.10	Запуск . . . . .	10
4.11	Редактирование . . . . .	11
4.12	Запуск . . . . .	11
4.13	Новый файл . . . . .	11
4.14	Треть программа . . . . .	12
4.15	Запуск . . . . .	12
4.16	Редактирование . . . . .	12
4.17	Запуск . . . . .	13
4.18	Создание файла . . . . .	13
4.19	Четвертая программа . . . . .	13
4.20	Запуск . . . . .	13
4.21	Создание файла . . . . .	14
4.22	Программа . . . . .	15
4.23	Выражение . . . . .	15

# 1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

## **2 Задание**

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Задание 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 делает 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

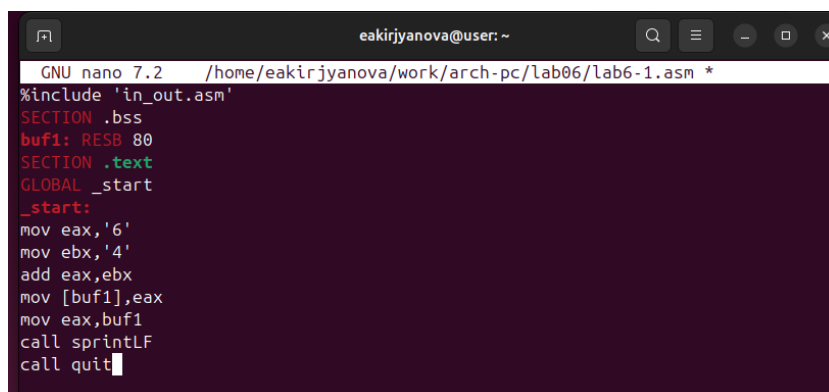
### 4.1 Символьные и численные данные в NASM

Создаю новый каталог и файл в нем (рис. 4.1).

```
eakirjyanova@user:~$ mkdir -p /work/arch-pc/lab06
eakirjyanova@user:~$ cd /work/arch-pc/lab06
eakirjyanova@user:~/work/arch-pc/lab06$ touch lab6-1.asm
```

Рис. 4.1: Создание

Ввожу текст программы из листинга 6.1 (рис. 4.2)



```
GNU nano 7.2 /home/eakirjyanova/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

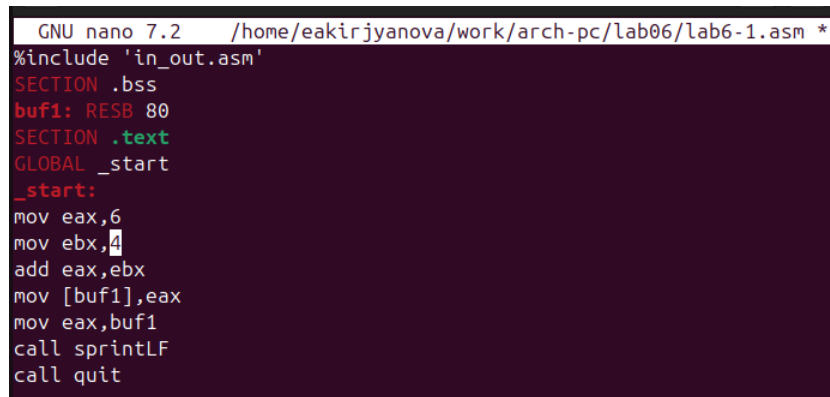
Создаю исполняемый файл и запускаю его (рис. 4.3).

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

Рис. 4.3: Запуск



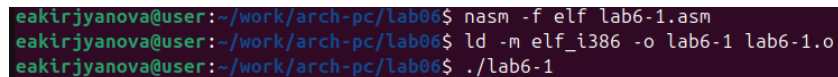
Редактирую текст программы (рис. 4.4).



```
GNU nano 7.2 /home/eakirjyanova/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: Редактирование

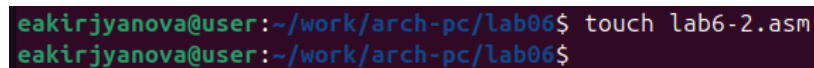
Создаю обновленный исполняемый файл и запускаю его (рис. 4.5).



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

Рис. 4.5: Запуск

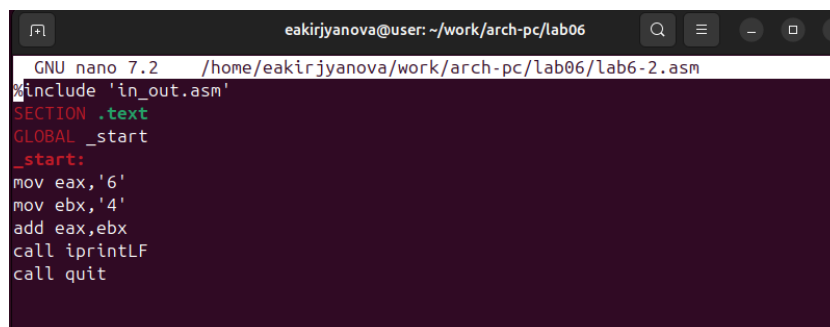
Создаю новый файл (рис. 4.6).



```
eakirjyanova@user:~/work/arch-pc/lab06$ touch lab6-2.asm
eakirjyanova@user:~/work/arch-pc/lab06$
```

Рис. 4.6: Новый файл

Ввожу текст программы из листинга 6.2 (рис. 4.7).



```
GNU nano 7.2 /home/eakirjyanova/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
```

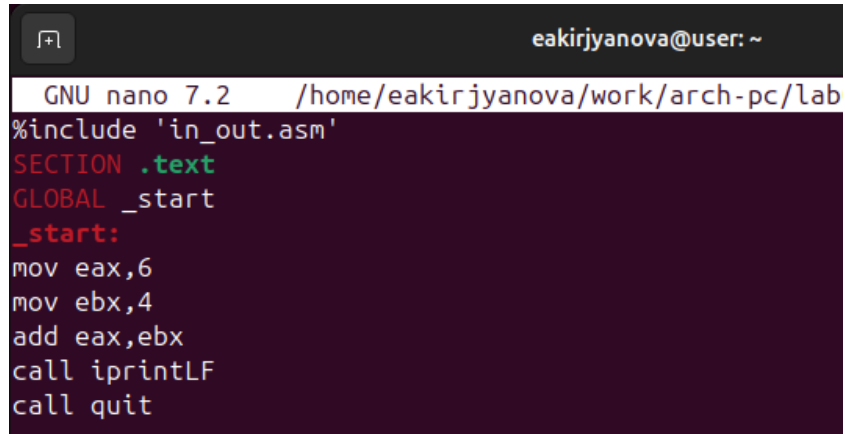
Рис. 4.7: Вторая программа

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

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

Рис. 4.8: Запуск программы

Аналогично предыдущему меняю символы на числа (рис. 4.9).



```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

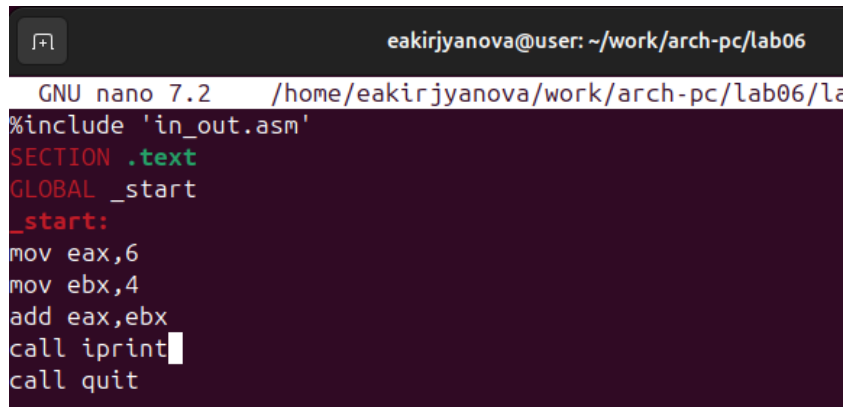
Рис. 4.9: Редактирование

Создаю исполняемый файл и запускаю его (рис. 4.10).

```
eakirjyanova@user:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
eakirjyanova@user:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
eakirjyanova@user:~/work/arch-pc/lab06$ ./lab6-2
bash: ./lab6-2: Нет такого файла или каталога
eakirjyanova@user:~/work/arch-pc/lab06$ ./lab6-2
10
```

Рис. 4.10: Запуск

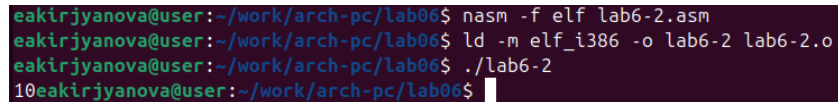
Меняю iprintLF на iprint(рис. 4.11).



```
eakirjyanova@user: ~/work/arch-pc/lab06
GNU nano 7.2 /home/eakirjyanova/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. 4.11: Редактирование

Создаю исполняемый файл и запускаю его (рис. 4.12).



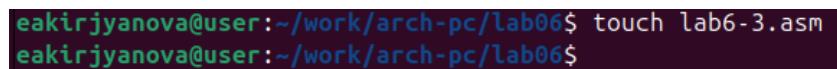
```
eakirjyanova@user:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
eakirjyanova@user:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
eakirjyanova@user:~/work/arch-pc/lab06$ ./lab6-2
10eakirjyanova@user:~/work/arch-pc/lab06$
```

Рис. 4.12: Запуск

Вывод изменился, так как `iprintLF` переносит строку, а `iprint` нет.

## 4.2 Выполнение арифметических операций в NASM

Создаю новый файл (рис. 4.13).



```
eakirjyanova@user:~/work/arch-pc/lab06$ touch lab6-3.asm
eakirjyanova@user:~/work/arch-pc/lab06$
```

Рис. 4.13: Новый файл

Ввожу текст программы из листинга 6.3 (рис. 4.14).

```

GNU nano 7.2 /home/eakirjyanova/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

```

Рис. 4.14: Треть программа

Создаю исполняемый файл и запускаю его (рис. 4.15).

```

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

```

Рис. 4.15: Запуск

Редактирую (рис. 4.16).

```

GNU nano 7.2 /home/eakirjyanova/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/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран

```

Рис. 4.16: Редактирование

Создаю исполняемый файл и запускаю (рис. 4.17).

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

Рис. 4.17: Запуск

Создаю новый файл (рис. 4.18).

```
eakirjyanova@user:~/work/arch-pc/lab06$ touch variant.asm
eakirjyanova@user:~/work/arch-pc/lab06$
```

Рис. 4.18: Создание файла

Ввожу текст программы из листинга 6.4 (рис. 4.19).

```
eakirjyanova@user: ~/work/arch-pc/lab06
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab06/variant.asm *
;-----
; Программа вычисления варианта
;-----
#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 sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, eax=x
xor edx, edx
```

Рис. 4.19: Четвертая программа

Создаю исполняемый файл и запускаю его (рис. 4.20).

```
eakirjyanova@user:~/work/arch-pc/lab06$ nasm -f elf variant.asm
eakirjyanova@user:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
eakirjyanova@user:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246781
Ваш вариант: 2
```

Рис. 4.20: Запуск

Ответы на вопросы: 1. `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. `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`

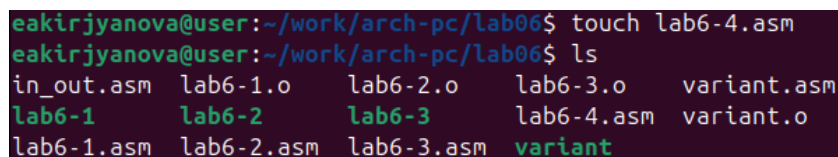
5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`

6. `inc edx` увеличивает значение регистра `edx` на 1

7. `mov eax,edx call iprintLF`

## 4.3 Задание для самостоятельной работы

Создаю файл (рис. 4.21).



```
eakirjyanova@user:~/work/arch-pc/lab06$ touch lab6-4.asm
eakirjyanova@user:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.o  lab6-2.o  lab6-3.o  variant.asm
lab6-1      lab6-2    lab6-3    lab6-4.asm  variant.o
lab6-1.asm  lab6-2.asm  lab6-3.asm  variant
```

Рис. 4.21: Создание файла

Ввожу текст программы для вычисления выражения под 2 вариантом (рис. 4.22).

```
GNU nano 7.2 /home/eakirjyanova/work/arch-pc/lab06/lab6-4.asm *
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат f(x): ',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

;Вычисление f(x) = ((12 * x) + 3) * 5
mov ebx, 12
mul ebx
```

Рис. 4.22: Программа

Создаю исполняемый файл, запускаю и ввожу значения x1, x2 для проверки (рис. 4.23).

```
eakirjyanova@user:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
eakirjyanova@user:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
eakirjyanova@user:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x:
1
75
eakirjyanova@user:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x:
6
375
```

Рис. 4.23: Выражение

Текст программы:

```
%include 'in_out.asm' SECTION .data msg: DB 'Введите значение переменной
x:',0 rem: DB 'Результат f(x):',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
;Вычисление f(x) = ((12 * x) + 3) * 5 mov ebx, 12 mul ebx add eax, 3 mov ebx,5 mul
ebx mov edi,eax mov eax, edi call iprintLF call quit
```

## **5 Выводы**

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.



## **6 Список литературы**

1. Лабораторная работа №6