

# **Отчет по лабораторной работе №6**

**Архитектура компьютера**

Раднаев Ардан Баирович

# 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>15</b>
	<b>Список литературы</b>	<b>16</b>

## **Список иллюстраций**

## **Список таблиц**

# 1 Цель 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

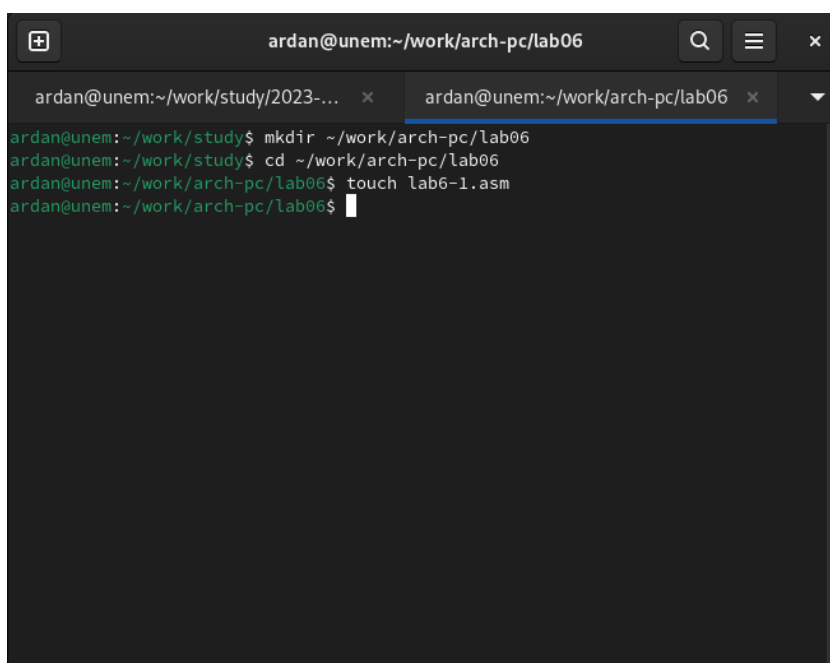
### 3 Теоретическое введение

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. 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

1. Создайте каталог для программ лабораторной работы № 6, перейдите в него и создайте файл lab6-1.asm: (рис. ??).

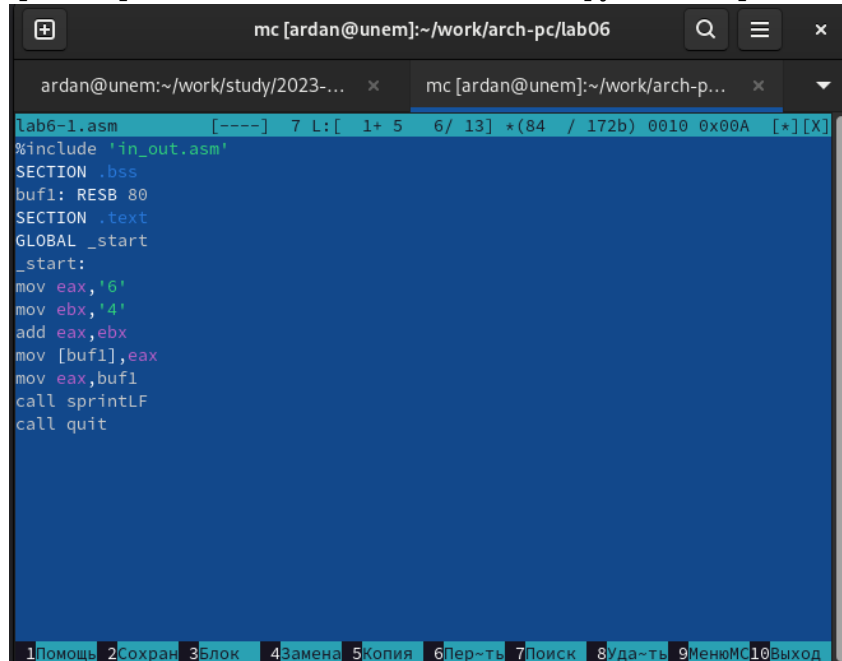


```
ardan@unem:~/work/arch-pc/lab06
ardan@unem:~/work/study/2023-... x ardan@unem:~/work/arch-pc/lab06 x
ardan@unem:~/work/study$ mkdir ~/work/arch-pc/lab06
ardan@unem:~/work/study$ cd ~/work/arch-pc/lab06
ardan@unem:~/work/arch-pc/lab06$ touch lab6-1.asm
ardan@unem:~/work/arch-pc/lab06$
```

2. Введите в файл lab6-1.asm текст программы из листинга 6.1. В данной программе в регистр eax записывается символ 6 (mov eax,'6'), в регистр ebx символ 4 (mov ebx,'4'). Далее к значению в регистре eax прибавляем значение регистра ebx (add eax,ebx, результат сложения запишется в регистр eax). Далее выводим результат. Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (mov [buf1],eax), а затем запишем адрес

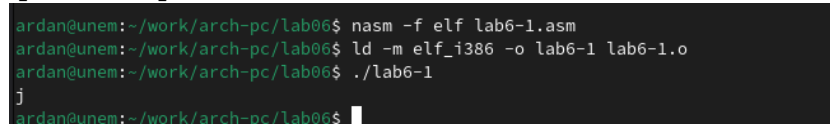


переменной buf1 в регистр eax (mov eax,buf1) и вызовом функцию sprintf



```
lab6-1.asm [----] 7 L: [ 1+ 5 6/ 13] *(84 / 172b) 0010 0x00A [*][X]
#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. ??). (рис. ??).



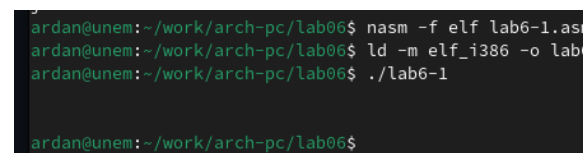
```
ardan@unem:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
ardan@unem:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
ardan@unem:~/work/arch-pc/lab06$ ./lab6-1
j
ardan@unem:~/work/arch-pc/lab06$
```

- Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 6.1) следующим образом: замените строки

mov eax,'6' mov ebx,'4'

на строки

mov eax,6 mov ebx,4

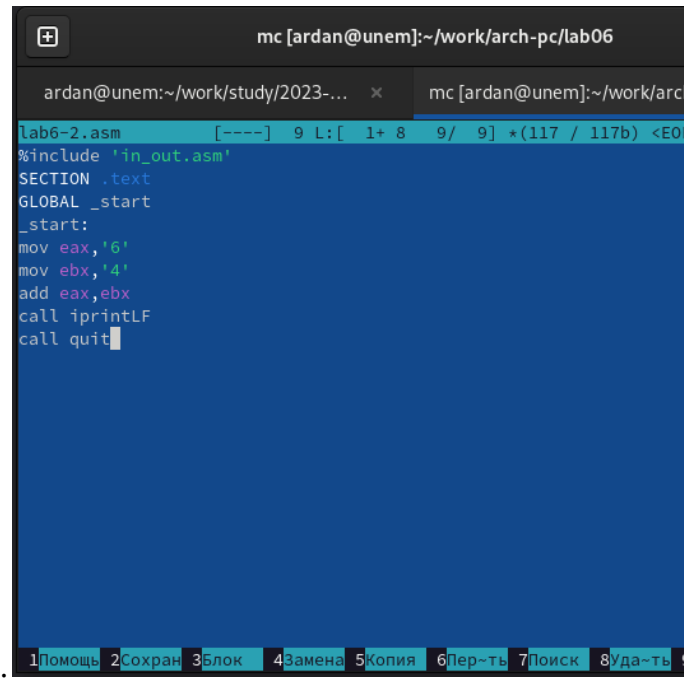


```
ardan@unem:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
ardan@unem:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
ardan@unem:~/work/arch-pc/lab06$ ./lab6-1
ardan@unem:~/work/arch-pc/lab06$
```

Создайте исполняемый файл и запустите его. (рис. ??).


На этот раз программа выдала пустую строку, потому что символ 10 означает переход на новую строку

- Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него



```
mc [ardan@unem]:~/work/arch-pc/lab06
ardan@unem:~/work/study/2023-... x mc [ardan@unem]:~/work/arch-pc/lab06
lab6-2.asm [----] 9 L: [ 1+ 8 9/ 9] *(117 / 117b) <EO
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

текст про- граммы из листинга 6.2. (рис. ??).



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

В результате работы программы мы получим число 106.(рис. ??).

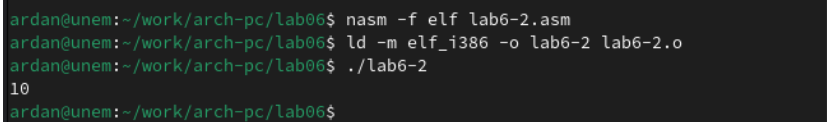
Аналогично предыдущему примеру изменим символы на числа. Замените строки

```
mov eax,'6' mov ebx,'4'
```

на строки

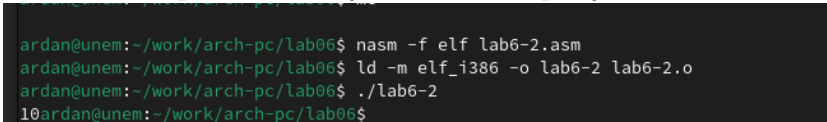
```
mov eax,6 mov ebx,4
```

Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы? Получилось число 10. (рис. ??).



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

Замените функцию iprintLF на iprint. Создайте исполняемый файл и запустите его. Чем отличается вывод функций iprintLF и iprint? Заменив функцию вывода на iprint, я получаю тот же результат, но без переноса строки (рис. ??).



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

5. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения  $\square(\square) = (5 \square 2 + 3)/3$ .

```

lab6-3.asm  [----] 39 L:[ 8+13 21/ 29] *(816 /1365b) 0010 0x00A [*][X]
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9МенюMC10Выход

```

(рис. ??). Создайте файл lab6-3.asm в каталоге ~/work/arch-pc/lab06:

Создайте исполняемый файл и запустите его. Результат работы программы должен быть следующим: user@dk4n31:~\$ ./lab6-3 Результат: 4 Остаток от деления: 1

```

ardan@unem:~/work/arch-pc/lab06
ardan@unem:~/work/arch-pc/lab06$ ./lab6-2
106
ardan@unem:~/work/arch-pc/lab06$ mc

ardan@unem:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
ardan@unem:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
ardan@unem:~/work/arch-pc/lab06$ ./lab6-2
10
ardan@unem:~/work/arch-pc/lab06$ mc

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

```

user@dk4n31:~\$ (рис. ??).

Измените текст программы для вычисления выражения  $\square(\square) = (4 \square 6 + 2)/5$ . Создай-

```
mc [ardan@unem]:~/work/arch-pc/lab06
ardan@unem:~/work/study/2023-... x mc [ardan@unem]:~/V
lab6-3.asm [----] 35 L: [ 8+21 29/ 29] *(1353/136)
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, 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 iprintLF ; из 'edi' в виде символов
mov eax,edx ; 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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8
```

те исполняемый файл и проверьте его работу. (рис. ??).

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

(рис. ??).

- 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:  $(\text{номер билета} \bmod 20) + 1$ , где  $\text{номер билета}$  – номер студенческого билета (В данном случае  $\text{номер билета} \bmod 20$  – это остаток от деления  $\text{номер билета}$  на  $20$ ).
  - вывести на экран номер варианта.

Создайте файл variant.asm в каталоге ~/work/arch-pc/lab06 Внимательно изучите текст программы из листинга 6.4 и введите в файл variant.asm. Создайте исполняемый файл и запустите его. Проверьте результат работы программы вычислив номер варианта аналитически.

```
ardan@unem:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
ardan@unem:~/work/arch-pc/lab06$ mc
ardan@unem:~/work/arch-pc/lab06$ nasm -f elf variant.asm
ardan@unem:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
ardan@unem:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246724
Ваш вариант: 5
ardan@unem:~/work/arch-pc/lab06$
```

(рис. ??).

tos-title: 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 read` - 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
```

tos-title: Задание для самостоятельной работы

Написать программу вычисления выражения  $\square = \square(\square)$ . Программа должна выводить выражение для вычисления, выводить запрос на ввод значения  $\square$ , вычислять заданное выражение в зависимости от введенного  $\square$ , выводить результат вычислений. Вид функции  $\square(\square)$  выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений  $\square_1$  и  $\square_2$  из 6.3

Буду выполнять задание для 5-го варианта  $(9\square - 8)/8$

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```

msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 9
mul ebx
sub eax, 8
mov ebx, 8
div ebx
mov edi, eax
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit

```

## 5 Выводы

По итогам Лабораторной работы я научился работать с алгебраическими функциями в NASM

## **Список литературы**