

# **Лабораторная работа №6**

**Архитектура компьютера**

Казначеева Кристина Никитична

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>3</b>
<b>2</b>	<b>Задание</b>	<b>4</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>5</b>
<b>4</b>	<b>Вывод</b>	<b>17</b>

# 1 Цель работы

В рамках лабораторной работы мы изучим арифметические инструкции языка ассемблера NASM, чтобы в дальнейшем научиться решать простые арифметические задачи с помощью ассемблера.

## 2 Задание

В этой лабораторной работе мы познакомимся с основными типами данных в ассемблере NASM, освоим выполнение арифметических операций, научимся выводить значения регистров на экран и напишем программу для вычисления арифметических выражений.

### 3 Выполнение лабораторной работы

Создадим каталог и перейдём в него, затем создадим файл lab6-1.asm: (рис. 3.1).

```
knkaznacheeva@dk8n81 ~ $ mkdir ~/work/arch-pc/lab06  
knkaznacheeva@dk8n81 ~ $ cd ~/work/arch-pc/lab06  
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 3.1: Создание каталога и файла

Вводим в файл lab6-1.asm текст программы вывода значения регистра eax (рис. 3.2).

```

mc [knkaznacheeva@dk8n81.dk....edu.ru
/afs/.dk.sci.pfu.edu.ru/home/k/r
#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. 3.2: Ввод текста программы

Создаём исполняемый файл и запускаем его (рис. 3.3).

```

knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-1
j

```

Рис. 3.3: Запуск файла

Далее изменим текст программы и вместо символов, запишем в регистры числа (рис. 3.4).

```
/afs/.dk.sci.pfu.edu.ru/  
%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. 3.4: Замена текста программы

Создадим исполняемый файл и запустим его (рис. 3.5).

```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm  
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o  
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 3.5: Запуск файла

Создаём файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 (рис. 3.6).

```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
```

Рис. 3.6: Создание файла

Вводим в него текст программы вывода значения регистра eax (рис. 3.7).

```
mc [knkaznacheeva@dk8n81
/afs/.dk.sci.pfu.edu.ru
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

Рис. 3.7: Ввод текста программы

Создаём исполняемый файл и запускаем его (рис. 3.8):

```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 3.8: Запуск файла

Далее заменяем текст программы и вместо символов, записываем в регистры числа (рис. 3.9).



mc [knkaznacheeva@dk3

```
/afs/.dk.sci.pfu.edu.ru/  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprint  
call quit
```

Рис. 3.9: Замена текста программы

Создаём исполняемый файл и запускаем его (рис. 3.10).

```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm  
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o  
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-2  
10
```

Рис. 3.10: Запуск файла

Заменяем функцию iprintLF на iprint (рис. 3.11).

```
mc [knkaznacheeva@dk8n8
```

```
/afs/.dk.sci.pfu.edu.r  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprint  
call quit
```

Рис. 3.11: Замена подпрограммы sprintLF на sprint

При замене функции iprintLF на iprint сообщение выводится в одну строку (рис. 3.12).

```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm  
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o  
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-2  
10knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $
```

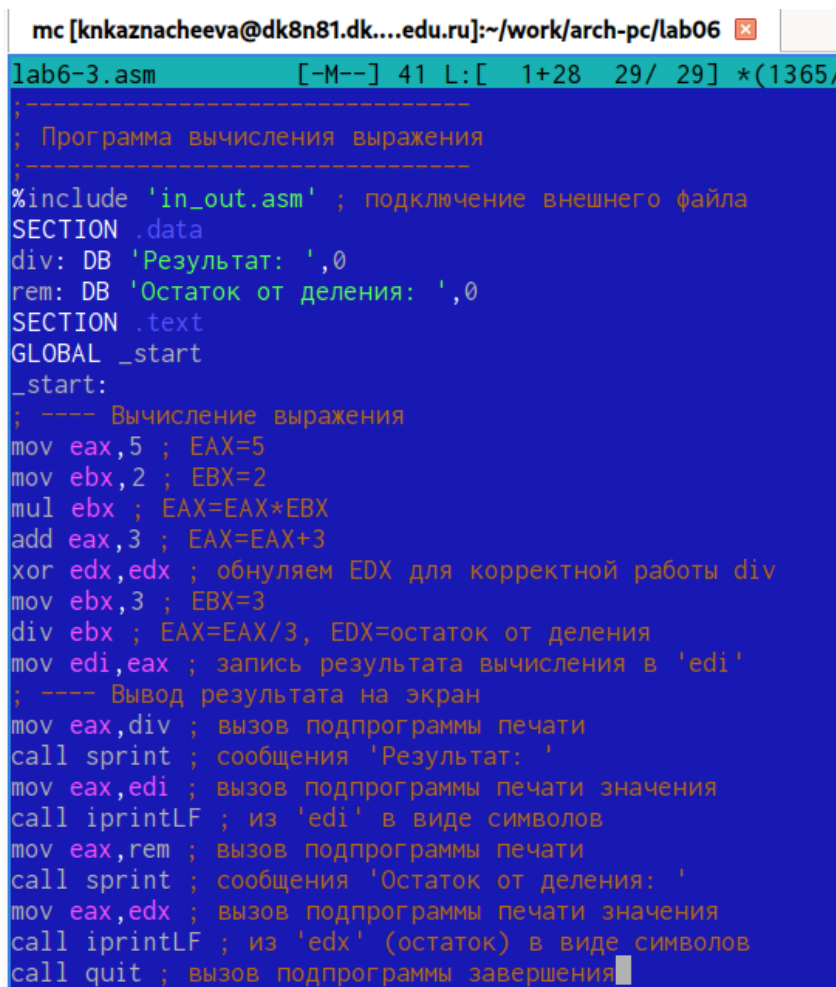
Рис. 3.12: Замена подпрограммы sprintLF на sprint

Создаём файл lab6-3.asm в каталоге ~/work/arch-pc/lab06 (рис. 3.13).

```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm  
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $
```

Рис. 3.13: Создание файла

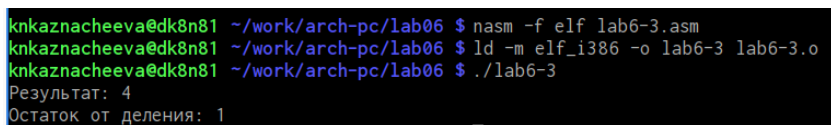
Вводим в файл lab6-3.asm текст программы вычисления выражения  $f(x) = (5 \cdot 2 + 3)/3$  (рис. 3.14).



```
mc[knkaznacheeva@dk8n81.dk....edu.ru]:~/work/arch-pc/lab06
lab6-3.asm [-M--] 41 L:[ 1+28 29/ 29] *(1365,
;
; 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 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. 3.14: Ввод текста программы

Создаём исполняемый файл и запускаем его (рис. 3.15).



```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 3.15: Запуск получившегося файла

Измените текст программы для вычисления выражения  $f(x) = (4 \cdot 6 + 2)/5$  (рис. 3.16).

```
mc [knkaznacheeva@dk8n81.dk....edu.ru]:~/work/arch-pc/lab06
lab6-3.asm [----] 41 L: [ 1+28 29/ 29] *(1365
;-----
; 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=4
mov ebx,2 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; 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,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. 3.16: Замена текста программы

Создаём исполняемый файл и проверяем его работу (рис. 3.17).

```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 3.17: Запуск получившегося файла

Создаём файл variant.asm в каталоге ~/work/arch-pc/lab06, (рис. 3.18).

```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
```

Рис. 3.18: Создание файла

Затем вводим в него текст программы вычисления варианта задания по номеру студенческого билета (рис. 3.19).

```
mc [knkaznacheeva@dk8n81.dk....edu.ru]:~/work/arch-pc/lab06
variant.asm [-M--] 9 L:[ 1+27 28/ 28]
;
; 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 sprintLF
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
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit
```

Рис. 3.19: Воод текста программы

Создаём исполняемый файл и запускаем его, затем проверим результат работы программы, вычислив номер варианта аналитически (рис. 3.20).

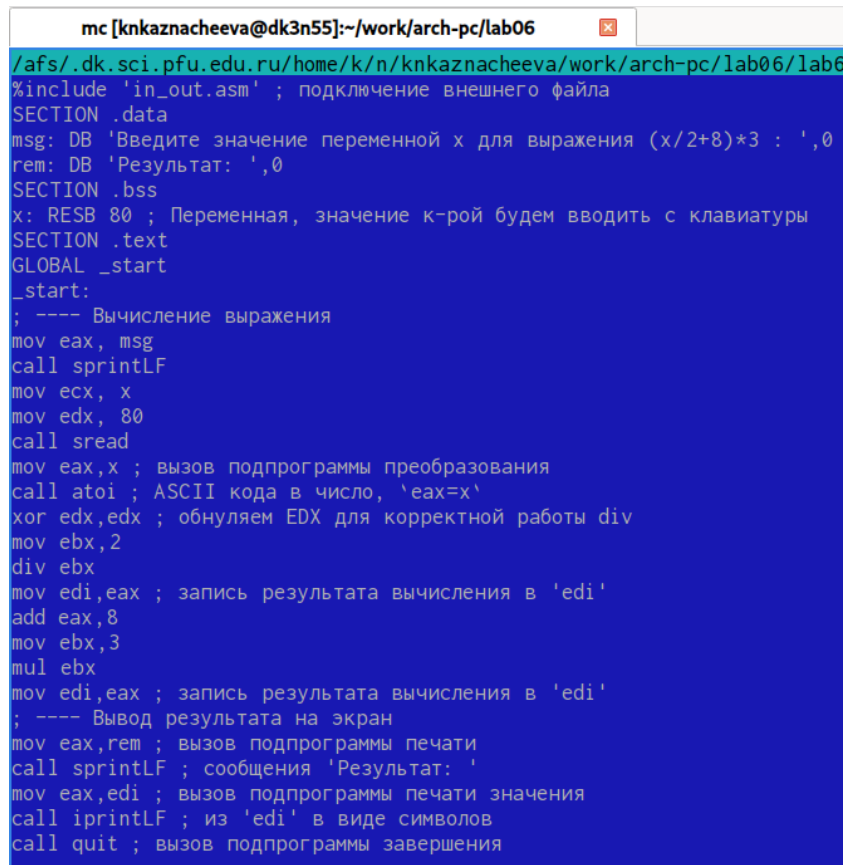
```
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
knkaznacheeva@dk8n81 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132246773
Ваш вариант: 14
```

Рис. 3.20: Запуск получившегося файла

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? • `mov eax, rem`: Эта строка загружает адрес сообщения “Ваш вариант:” (которое находится в переменной `rem`) в регистр `eax`. • `call sprint`: Эта строка вызывает функцию `sprint`, которая выводит строку, адрес которой находится в `eax`. Таким образом, эти две строки совместно выводят на экран сообщение “Ваш вариант:”.
2. Для чего используются следующие инструкции? • `mov ecx, x`: Эта инструкция загружает адрес буфера `x` в регистр `ecx`. Регистр `ecx` используется в системе вызовов для указания адреса буфера, куда будет записана введенная строка. • `mov edx, 80`: Эта инструкция загружает значение 80 в регистр `edx`. Регистр `edx` указывает максимальную длину вводимой строки (80 байт). • `call sread`: Эта инструкция вызывает функцию `sread`. Функция `sread` - это внешняя функция, которая занимается считыванием данных с клавиатуры. Функция `sread` использует значения из `ecx` (адрес буфера) и `edx` (максимальная длина) для корректного ввода и записи данных в буфер `x`.
3. Для чего используется инструкция “`call atoi`”? Инструкция `call atoi` используется для преобразования строки ASCII-символов в целое число.
4. Какие строки листинга 6.4 отвечают за вычисления варианта? Строки, которые отвечают за вычисление варианта, это: `xor edx, edx`; Обнуляем EDX для `div` `mov ebx, 20` `div ebx`; `eax = x / 20`, `edx = x % 20` `inc edx`; `edx = (x % 20) + 1`
5. В какой регистр записывается остаток от деления при выполнении инструкции “`div ebx`”? Остаток от деления при выполнении инструкции `div ebx` записывается в регистр `edx`.
6. Для чего используется инструкция “`inc edx`”? Инструкция `inc edx` увеличивает значение в регистре `edx` на 1. В данной программе она используется, чтобы вариант студента был в диапазоне от 1 до 20, а не от 0 до 19.

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений? За вывод результата вычислений отвечают следующие строки: `mov eax, rem call sprint mov eax, edx call iprintLF`

Создадим файл `lab6-4.asm` и напишем программу вычисления выражения для 14 варианта:  $y(x) = (x/2 + 8) \cdot 3$  (рис. 3.21).



```
mc [knkaznacheeva@dk3n55]:~/work/arch-pc/lab06
/afs/.dk.sci.pfu.edu.ru/home/k/n/knkaznacheeva/work/arch-pc/lab06/lab6
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x для выражения (x/2+8)*3 : ',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 sprintLF
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 ; обнуляем EDX для корректной работы div
mov ebx, 2
div ebx
mov edi, eax ; запись результата вычисления в 'edi'
add eax, 8
mov ebx, 3
mul ebx
mov edi, eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
call sprintLF ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 3.21: Программа для 14 варианта

Создаём исполняемый файл и проверяем его работу для значения  $x_1 = 1$  (рис. 3.22).

```

knkaznacheeva@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
knkaznacheeva@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
knkaznacheeva@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x для выражения (x/2+8)*3 :
1
Результат:
24

```

Рис. 3.22: Проверка файла при x1

Затем проверяем работу файла для значения  $x2 = 4$  (рис. 3.23)

```

knkaznacheeva@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x для выражения (x/2+8)*3 :
4
Результат:
30

```

Рис. 3.23: Проверка файла при x2



## 4 Вывод

В ходе лабораторной работы мы получили практические навыки работы с ассемблером NASM: освоили типы данных, арифметические операции, вывод данных на экран и реализовали программу для вычисления выражений.