

# **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**

**дисциплина: Архитектура компьютера**

Мантуров Татархан Бесланович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Порядок выполнения лабораторной работы . . . . .	6
3.2	Ответы на вопросы . . . . .	14
3.3	Задание для самостоятельной работы . . . . .	14
<b>4</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

3.1	Создаём каталог . . . . .	6
3.2	заполняем программу . . . . .	7
3.3	результат . . . . .	7
3.4	подмена . . . . .	8
3.5	результат . . . . .	8
3.6	заполняем . . . . .	9
3.7	результат . . . . .	9
3.8	подмена . . . . .	10
3.9	результат . . . . .	10
3.10	пишем программу . . . . .	11
3.11	результат . . . . .	11
3.12	меняем выражение . . . . .	12
3.13	результат . . . . .	12
3.14	пишем программу . . . . .	13
3.15	результат . . . . .	13

# 1 Цель работы

Научиться писать и анализировать ассемблерный код с арифметическими операциями и понять синтаксис. Работа поможет развить навыки низкоуровневого программирования и понимания работы процессора.

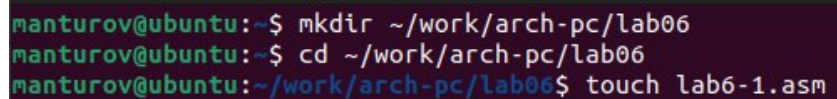
## 2 Задание

Написать несколько программ для вычислений.

## 3 Выполнение лабораторной работы

### 3.1 Порядок выполнения лабораторной работы

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



```
manturov@ubuntu:~$ mkdir ~/work/arch-pc/lab06  
manturov@ubuntu:~$ cd ~/work/arch-pc/lab06  
manturov@ubuntu:~/work/arch-pc/lab06$ touch lab6-1.asm
```

Рис. 3.1: Создаём каталог

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр eax

```

GNU nano 6.2 /h
#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: заполняем программу

```

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

```

Рис. 3.3: результат

Изменим текст программы и вместо символов, запишем в регистры числа.

```
%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. 3.4: подмена

```
manturov@ubuntu:~/work/arch-pc/lab06$ ./lab6-1
```

Рис. 3.5: результат

Преобразуем текст программы из Листинга 6.1 с использованием этих функций.



```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 3.6: заполняем

```
manturov@ubuntu:~/work/arch-pc/lab06$ ./lab6-1
```

Рис. 3.7: результат

Изменим символы на числа

```

%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit

```

Рис. 3.8: подмена

```

manturov@ubuntu:~/work/arch-pc/lab06$ ./lab6-2
10

```

Рис. 3.9: результат

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения  $f(x) = (5 * 2 + 3) / 3$

```

;-----
; 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. 3.10: пишем программу

```

manturov@ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
manturov@ubuntu:~/work/arch-pc/lab06$

```

Рис. 3.11: результат

Измените текст программы для вычисления выражения  $f(x) = (4 * 6 + 2)/5$ .  
Создайте исполняемый файл и проверьте его работу.

```

;-----
; 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
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. 3.12: меняем выражение

```

manturov@ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5

```

Рис. 3.13: результат

рассмотрим программу вычисления варианта задания по номеру студенческого билета

```

;-----
; 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
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprintf
mov eax, edx
call iprintLF
call quit

```

Рис. 3.14: пишем программу

```

manturov@ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm
manturov@ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
manturov@ubuntu:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132237379

```

Рис. 3.15: результат

## 3.2 Ответы на вопросы

1. Строка “moveax,rem” и строка “call sprint” отвечают за вывод на экран сообщения ‘Ваш вариант:’.
2. Эти инструкции используются для чтения строки с вводом данных от пользователя. Начальный адрес строки сохраняется в регистре есх, а количество символов в строке (максимальное количество символов, которое может быть считано) сохраняется в регистре edx. Затем вызывается процедура sread, которая выполняет чтение строки.
3. Инструкция “call atoi” используется для преобразования строки в целое число. Она принимает адрес строки в регистре еах и возвращает полученное число в регистре еах. Строка “xoredx,edx” обнуляет регистр. edx перед выполнением деления. Строка “movebx,20” загружает значение 20 в регистр ebx. Строка “divebx” выполняет деление регистра еах на значение регистра ebx с сохранением частного в регистре еах и остатка в регистре edx,
4. Остаток от деления записывается в регистр edx.
5. Инструкция “inc edx” используется для увеличения значения в регистре edx на
6. В данном случае, она увеличивает остаток от деления на 1. 13
7. Строка “movu еах,edx” передает значение остатка от деления в регистр еах.  
36 Строка “call iprintLF” вызывает процедуру iprintLF для вывода значения на экран вместе с переводом строки.

## 3.3 Задание для самостоятельной работы

Написать программу вычисления выражения  $y = f(x)$ . Программа должна вывести выражение для вычисления, выводить запрос на ввод значения  $x$ , вычислять заданное выражение в зависимости от введенного  $x$ , выводить результат вычислений. Создайте исполняемый файл и проверьте его работу для значений  $x1$  и



```

; -----
; Программа вычисления выражения
; -----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
result_msg: DB 'Результат: ', 0
remainder_msg: DB 'Остаток от деления: ', 0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, 3      ; x = 3
mov ebx, eax    ; сохраняем x в ebx для возведения в степень
mul ebx        ; EAX = EAX * EBX (возведение в куб)
mov ebx, 3      ; EBX = 3
div ebx        ; EAX = EAX / 3, EDX = остаток от деления
add eax, 21     ; EAX = EAX + 21
mov edi, eax    ; запись результата вычисления в 'edi'
mov esi, edx    ; запись остатка от деления в 'esi'
; ---- Вывод результата на экран
mov eax, result_msg ; вызов подпрограммы печати
call sprint        ; сообщение 'Результат: '
mov eax, edi       ; вызов подпрограммы печати значения
call iprintLF      ; из 'edi' в виде символов
; ---- Вывод остатка на экран
mov eax, remainder_msg ; вызов подпрограммы печати
call sprint        ; сообщение 'Остаток от деления: '
mov eax, esi       ; вызов подпрограммы печати значения
call iprintLF      ; из 'esi' (остаток) в виде символов
call quit         ; вызов подпрограммы завершения

; -----
; Программа вычисления выражения
; -----
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
result_msg: DB 'Результат: ', 0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, 1      ; x = 1
mov ebx, eax    ; сохраняем x в ebx для возведения в степень
mul ebx        ; EAX = EAX * EBX (возведение в куб)
mov ebx, 3      ; EBX = 3
div ebx        ; EAX = EAX / 3, EDX = остаток от деления
add eax, 21     ; EAX = EAX + 21
mov edi, eax    ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax, result_msg ; вызов подпрограммы печати
call sprint        ; сообщение 'Результат: '
mov eax, edi       ; вызов подпрограммы печати значения
call iprintLF      ; из 'edi' в виде символов
call quit         ; вызов подпрограммы завершения

```

## 4 Выводы

В работе были изучены арифметические операции в языке ассемблера NASM. Был рассмотрен синтаксис и были написаны и проанализированы программы на ассемблере, которые используют арифметические операции для решения различных задач.