

# **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6**

**дисциплина: Архитектура компьютера**

Газизянов Владислав Альбертович

# 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 . . . . .	12
3.3	Задание для самостоятельной работы . . . . .	13
<b>4</b>	<b>Выводы</b>	<b>15</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 . . . . .	8
3.7	результат . . . . .	9
3.8	подмена . . . . .	9
3.9	результат . . . . .	9
3.10	меняем . . . . .	9
3.11	результат . . . . .	9
3.12	пишем программу . . . . .	10
3.13	результат . . . . .	10
3.14	меняем выражение . . . . .	11
3.15	результат . . . . .	11
3.16	пишем программу . . . . .	12
3.17	результат . . . . .	12

# 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.

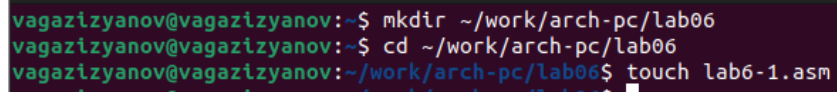
## 2 Задание

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



```
vagazizyanov@vagazizyanov:~$ mkdir ~/work/arch-pc/lab06
vagazizyanov@vagazizyanov:~$ cd ~/work/arch-pc/lab06
vagazizyanov@vagazizyanov:~/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

```

1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintf
13 call quit

```

Рис. 3.2: заполняем программу

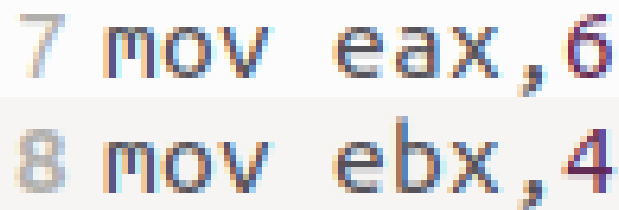
```

vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$ ./lab6-1
j

```

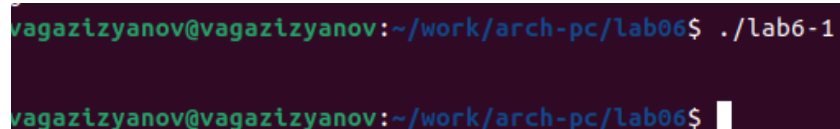
Рис. 3.3: результат

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



```
7 mov eax,6  
8 mov ebx,4
```

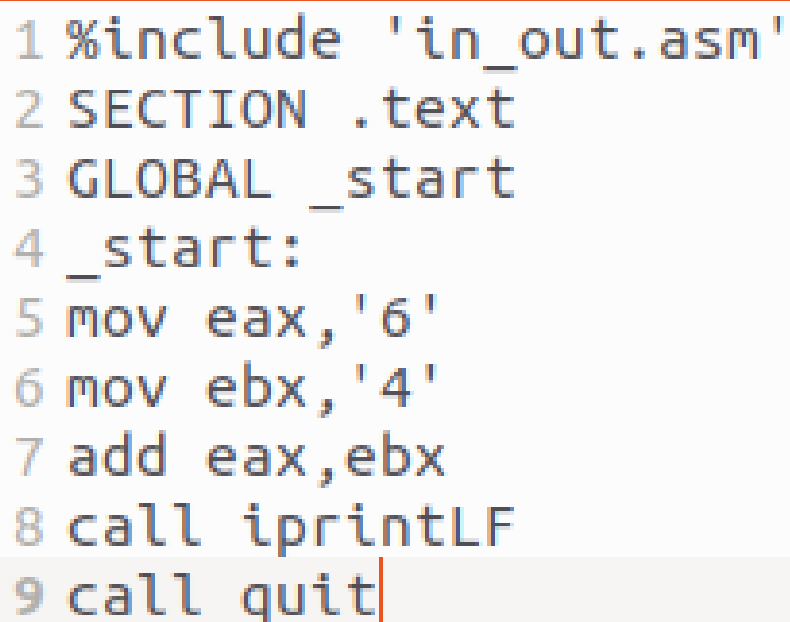
Рис. 3.4: подмена



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

Рис. 3.5: результат

Преобразуем текст программы из Листинга 6.1 с использованием этих функций.



```
1 %include 'in_out.asm'  
2 SECTION .text  
3 GLOBAL _start  
4 _start:  
5 mov eax,'6'  
6 mov ebx,'4'  
7 add eax,ebx  
8 call iprintLF  
9 call quit
```

Рис. 3.6: заполняем



```
vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$ ./lab6-2
106
```

Рис. 3.7: результат

Изменим символы на числа

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

Рис. 3.8: подмена

```
vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$ ./lab6-2
10
```

Рис. 3.9: результат

Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его.

```
8 call iprint
```

Рис. 3.10: меняем

```
vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$ ./lab6-2
10vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$
```

Рис. 3.11: результат

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

```

1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4 %include 'in_out.asm' ; подключение внешнего файла
5 SECTION .data
6 div: DB 'Результат: ',0
7 rem: DB 'Остаток от деления: ',0
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; ---- Вычисление выражения
12 mov eax,5 ; EAX=5
13 mov ebx,2 ; EBX=2
14 mul ebx ; EAX=EAX*EBX
15 add eax,3 ; EAX=EAX+3
16 xor edx,edx ; обнуляем EDX для корректной работы div
17 mov ebx,3 ; EBX=3
18 div ebx ; EAX=EAX/3, EDX=остаток от деления
19 mov edi,eax ; запись результата вычисления в 'edi'
20 ; ---- Вывод результата на экран
21 mov eax,div ; вызов подпрограммы печати
22 call sprint ; сообщения 'Результат: '
23 mov eax,edi ; вызов подпрограммы печати значения
24 call iprintLF ; из 'edi' в виде символов
25 mov eax,rem ; вызов подпрограммы печати
26 call sprint ; сообщения 'Остаток от деления: '
27 mov eax,edx ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edx' (остаток) в виде символов
29 call quit ; вызов подпрограммы завершения

```

Рис. 3.12: пишем программу

```

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

```

Рис. 3.13: результат

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

```

1 ;-----
2 ; Программа вычисления выражения
3 ;-----
4 %include 'in_out.asm' ; подключение внешнего файла
5 SECTION .data
6 div: DB 'Целая часть от деления: ',0
7 rem: DB 'Остаток от деления: ',0
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; ---- Вычисление выражения
12 mov eax,4 ; EAX=4
13 mov ebx,6 ; EBX=6
14 mul ebx ; EAX=EAX*EBX
15 add eax,2 ; EAX=EAX+2
16 xor edx,edx ; обнуляем EDX для корректной работы div
17 mov ebx,5 ; EBX=5
18 div ebx ; EAX=EAX/5, EDX=остаток от деления
19 mov edi,eax ; запись результата вычисления в 'edi'
20 ; ---- Вывод результата на экран
21 mov eax,div ; вызов подпрограммы печати
22 call sprint ; сообщения 'Целая часть от деления: '
23 mov eax,edi ; вызов подпрограммы печати значения
24 call iprintLF ; из 'edi' в виде символов
25 mov eax,rem ; вызов подпрограммы печати
26 call sprint ; сообщения 'Остаток от деления: '
27 mov eax,edx ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edx' (остаток) в виде символов
29 call quit ; вызов подпрограммы завершения

```

Рис. 3.14: меняем выражение

```

vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$ ./lab6-3
Целая часть от деления: 5
Остаток от деления: 1

```

Рис. 3.15: результат

рассмотрим программу вычисления варианта задания по номеру студенческого билета

```

1 ;-----
2 ; Программа вычисления варианта
3 ;-----
4 %include 'in_out.asm'
5 SECTION .data
6 msg: DB 'Введите № студенческого билета: ',0
7 rem: DB 'Ваш вариант: ',0
8 SECTION .bss
9 x: RESB 80
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg
14 call sprintf
15 mov ecx, x
16 mov edx, 80
17 call sread
18 mov eax,x ; вызов подпрограммы преобразования
19 call atoi ; ASCII кода в число, `eax=x`
20 xor edx,edx
21 mov ebx,20
22 div ebx
23 inc edx
24 mov eax,rem
25 call sprintf
26 mov eax,edx
27 call iprintLF
28 call quit

```

Рис. 3.16: пишем программу

```

vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1114425
Ваш вариант: 6

```

Рис. 3.17: результат

## 3.2 Ответы на вопросы

1. Строка “moveax,rem” и строка “call sprintf” 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 ecx, 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 `eax` и возвращает полученное число в регистре `eax`. Строка `"xoredx.edx"` обнуляет регистр `edx` перед выполнением деления. Строка `"movebx,20"` загружает значение 20 в регистр `ebx`. Строка `"divebx"` выполняет деление регистра `eax` на значение регистра `ebx` с сохранением частного в регистре `eax` 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 `"mov eax,edx"` передает значение остатка от деления в регистр `eax`.
- 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 выводить 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  $x_1$  и  $x_2$

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите x: ',0
4 div: DB 'Результат: ',0
5 SECTION .bss
6 rez: RESB 80
7 x: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 mov eax,msg
12 call sprintf
13
14 mov ecx,x
15 mov edx,80
16 call sread
17 mov eax,x
18 call atoi
19
20 add eax,2
21 mul eax
22 mov [rez],eax
23
24 mov eax, div
25 call sprintf
26 mov eax,[rez]
27 call iprintLF
28 call quit

```

```

vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$ ./home
Введите x:
2
Результат: 16
vagazizyanov@vagazizyanov:~/work/arch-pc/lab06$ ./home
Введите x:
8
Результат: 100

```

## 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.