

# Отчёт по лабораторной работе 6

дисциплина: Архитектура компьютера

Кара Егор

## Содержание

1. Цель работы.....	1
2. Выполнение лабораторной работы.....	1
3. Выводы.....	13

## 1. Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

## 2. Выполнение лабораторной работы

1. Создал каталог для программам лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.
2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения, записанные в регистр eax.

```
option -- 111
Open + • lab6-...
~/wo...ab06
preamble.tex | arch-pc-lab03-re • lab6-1.as 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 sprintLF
    call quit
```

Программа *lab6-1.asm*

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

Запуск программы *lab6-1.asm*

3. Далее изменяю текст программы и вместо символов, запишем в регистры числа.



The screenshot shows a terminal window with the following content:

```
%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
```

### Программа *lab6-1.asm*

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

```
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$
```

### Запуск программы *lab6-1.asm*

Никакой символ не виден, но он есть. Это возврат каретки LF.

4. Как отмечалось выше, для работы с числами в файле *in\_out.asm* реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал текст программы с использованием этих функций.

The screenshot shows a terminal window with the following content:

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

### Программа lab6-2.asm

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

### Запуск программы lab6-2.asm

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов '6' и '4' ( $54+52=106$ ). Однако, в отличии от прошлой программы, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим символы на числа.

The screenshot shows a terminal window titled 'lab6-2.a'. The window contains the following assembly code:

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

### Программа *lab6-2.asm*

Функция `iprintLF` позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10.

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

### Запуск программы *lab6-2.asm*

Заменил функцию `iprintLF` на `iprint`. Вывод отличается что нет переноса строки.

The screenshot shows a terminal window with the title bar 'lab6-2.asm'. The window contains the following assembly code:

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

### Программа lab6-2.asm

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

### Запуск программы lab6-2.asm

6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения

$$f(x) = (5 * 2 + 3) / 3$$

Open ▾



lab6-1.asm

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
    mov eax,5
    mov ebx,2
    mul ebx
    add eax,3
    xor edx,edx
    mov ebx,3
    div ebx
    mov edi, eax

    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF

    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF

    call quit
```

Программа lab6-3.asm

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

Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения

$$f(x) = (4 * 6 + 2) / 5$$

. Создал исполняемый файл и проверил его работу.

Open ▾

lab6-1.asm

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
    mov eax,4
    mov ebx,6
    mul ebx
    add eax,2
    xor edx,edx
    mov ebx,5
    div ebx
    mov edi,eax

    mov eax,div
    call sprint
    mov eax,edi
    call iprintLF

    mov eax,rem
    call sprint
    mov eax,edx
    call iprintLF

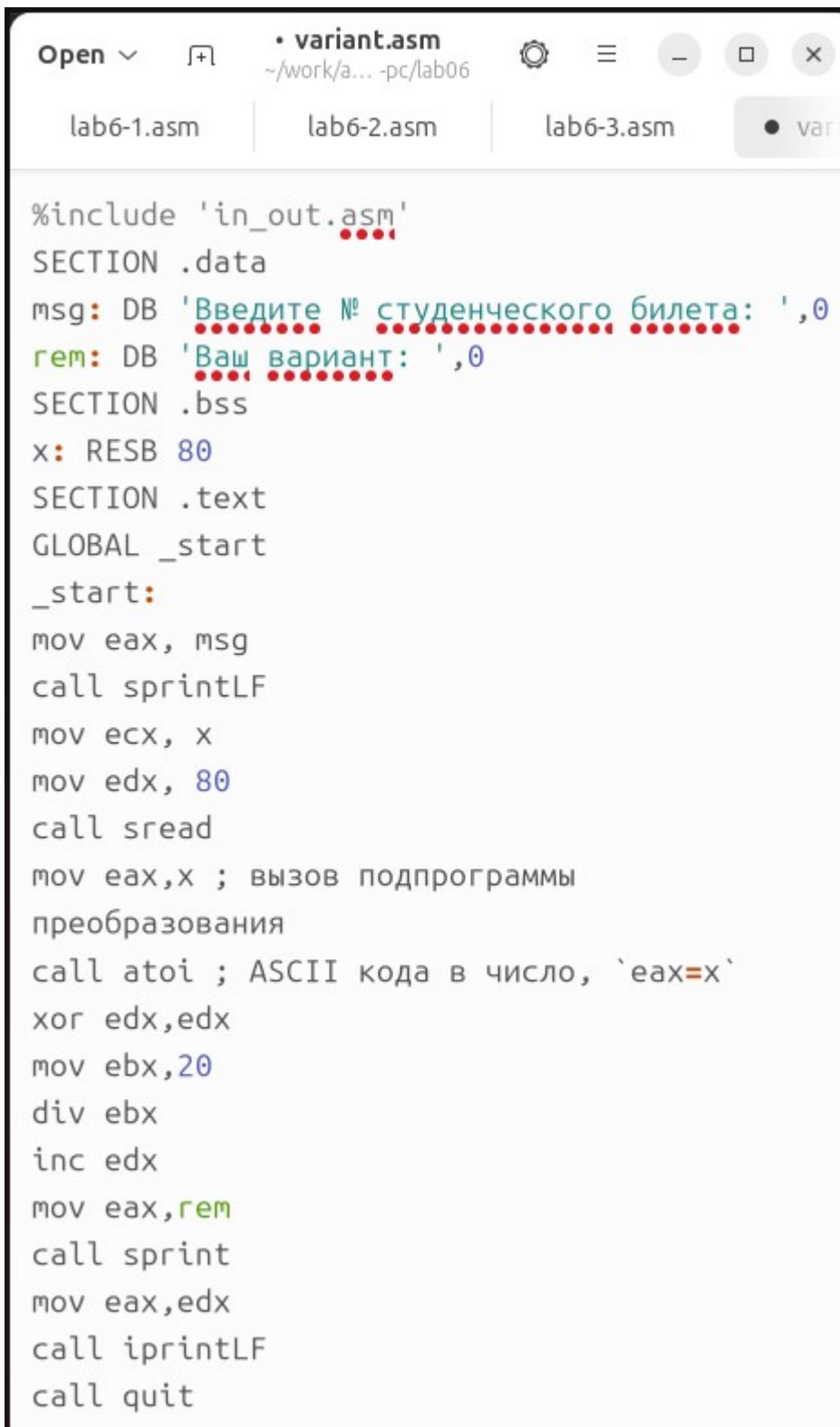
    call quit|
```

### Программа lab6-3.asm

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

### Запуск программы lab6-3.asm

7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета.



The screenshot shows a Windows Notepad window titled "variant.asm" with the file path "C:\Users\... -pc\lab06". The window contains assembly code for a program that reads a student ID number and a variant from the user and prints them back. The code includes sections for data, bss, and text, with various memory locations and registers being used.

```
%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
```

Программа *variant.asm*

```
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032253851
Ваш вариант: 12
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$
```

Запуск программы *variant.asm*

ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения 'Ваш вариант:'?

В строке `mov eax,rem` значение переменной с фразой 'Ваш вариант:' перекладывается в регистр `eax`.

Строка `call sprint` вызывает подпрограмму для вывода строки.

2. Для чего используется следующие инструкции?

`mov ecx, x` `mov edx, 80` `call sread`

`mov ecx, x` - перемещает значение переменной `X` в регистр `ecx`.

`mov edx, 80` - устанавливает значение 80 в регистр `edx`.

`call sread` - вызывает подпрограмму для чтения значения с консоли.

3. Для чего используется инструкция "call atoi"?

Эта инструкция вызывает подпрограмму, которая преобразует введенные символы в числовой формат.

4. Какие строки листинга отвечают за вычисления варианта?

`xor edx,edx` - обнуляет регистр `edx`.

`mov ebx,20` - устанавливает значение 20 в регистр `ebx`.

`div ebx` - производит деление номера студенческого билета на 20.

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

5. В какой регистр записывается остаток от деления при выполнении инструкции "div ebx"?

Остаток от деления записывается в регистр `edx`.

6. Для чего используется инструкция "inc edx"?

Инструкция inc edx увеличивает значение регистра edx на 1. В данном случае, она используется для выполнения формулы вычисления варианта, где требуется добавить 1 к остатку от деления.

7. Какие строки листинга отвечают за вывод на экран результата вычислений?

mov eax,edx - результат вычислений перекладывается в регистр eax.

call iprintLF - вызывается подпрограмма для вывода результата на экран.

8. 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. Вид функции  $f(x)$  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  $x_1$  и  $x_2$  из 6.3.

Получили вариант 12 -

$$(8x - 6)/2$$

для

$$x=1, x=5$$

The screenshot shows a Windows Notepad window with the title bar "prog.asm". The window contains assembly code for a program named "prog.asm". The code includes sections for .data, .bss, and .text, with various memory operations and system calls. Some parts of the code, such as string literals, contain red dots, likely indicating they are part of a larger file or have been partially loaded.

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите X : ',0
rem: DB 'выражение = : ',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
    mov ebx, eax
    mov eax, 8
    mul ebx
    sub eax, 6
    mov ebx, 2
    xor edx, edx
    div ebx
    mov ebx, eax
    mov eax, rem
    call sprint
    mov eax, ebx
    call iprintLF
    call quit
```

Программа *prog.asm*

```
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf prog.asm
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 prog.o -o prog
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$ ./prog
Введите X :
1
выражение = : 1
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$ ./prog
Введите X :
5
выражение = : 17
eakara@eakara-VirtualBox:~/work/arch-pc/lab06$ █
```

Запуск программы *prog.asm*

### 3. Выводы

Изучили работу с арифметическими операциями.