

# Лабораторная работа №6

Архитектура вычислительных систем

Манукян Захар Саркисович

## Содержание

1	Цель работы.....	1
2	Задание .....	1
3	Теоретическое введение .....	1
4	Выполнение лабораторной работы.....	2
5	Ответы на вопросы: .....	8
6	Выводы .....	8
	Список литературы.....	8

## 1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

## 2 Задание

Написать программу вычисления выражения. 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 выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создать исполняемый файл и проверить его работу для значений из 6.3.

## 3 Теоретическое введение

1. Адресация в NASM 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.
2. Арифметические операции в NASM Схема команды целочисленного сложения `add` (от англ. *addition* - добавление) выполняет сложение двух

операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака.

3. Целочисленное вычитание `sub` Команда целочисленного вычитания `sub` (от англ. subtraction – вычитание) работает аналогично команде `add`.
4. Команды инкремента и декремента Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: `inc` (от англ. increment) и `dec` (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд.
5. Команда изменения знака операнда `neg` Команда рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.
6. Команды умножения `mul` и `imul` Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда `mul` (от англ. multiply – умножение). Для знакового умножения используется команда `imul`.
7. Команды деления `div` и `idiv` Для деления, как и для умножения, существует 2 команды `div` (от англ. divide - деление) и `idiv`. Для беззнакового умножения используется команда `div`. Для знакового умножения используется команда `idiv`.

## 4 Выполнение лабораторной работы

1. Создаём каталог для программам лабораторной работы No 7, перейдём в него и создаём файл `lab6-1.asm`

```
zsmankyuan@dk3n66 ~ $ mkdir ~/work/arch-pc/lab06
zsmankyuan@dk3n66 ~ $ cd ~/work/arch-pc/lab06
zsmankyuan@dk3n66 ~/work/arch-pc/lab06 $ touch lab6-1.asm
zsmankyuan@dk3n66 ~/work/arch-pc/lab06 $
```

Рис. 1: 1.png

2. Введем в файл `lab6-1.asm` текст программы из листинга 6.1.

```
GNU nano 8.1                                lab6-1.asm
#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. 2: 2.png

3. Создадим исполняемый файл и запустим его.

```
zsmannukyan@dk3n55 ~ $ cd ~/work/arch-pc/lab06
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-1
j
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 3: 3.png

4. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы.

```
GNU nano 8.1 lab6-1.asm
#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. 4: 4.png

5. Создадим исполняемый файл и запустим его (6-1).

```
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-1

zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 5: 5.png

6. Создадим файл lab6-2.asm в каталоге. Введем в него текст программы из листинга 6.2 и запустим его.

```
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ touch lab6-2.asm
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-2
106
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $
```

Рис. 6: 6.png

7. Изменим символы на числа в lab6-2. Создадим исполняемый файл и запустим его.

```

GNU nano 8.1                                lab6-2.asm
%include 'in_out.asm';
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit

```

Рис. 7: 7.png

8. Создадим файл lab6-3.asm в каталоге. Введем в файл lab6-3.asm текст программы из листинга 6.3

```

zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-2
10
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $

```

Рис. 8: 8.png

9. Создадим исполняемый файл и запустим его.

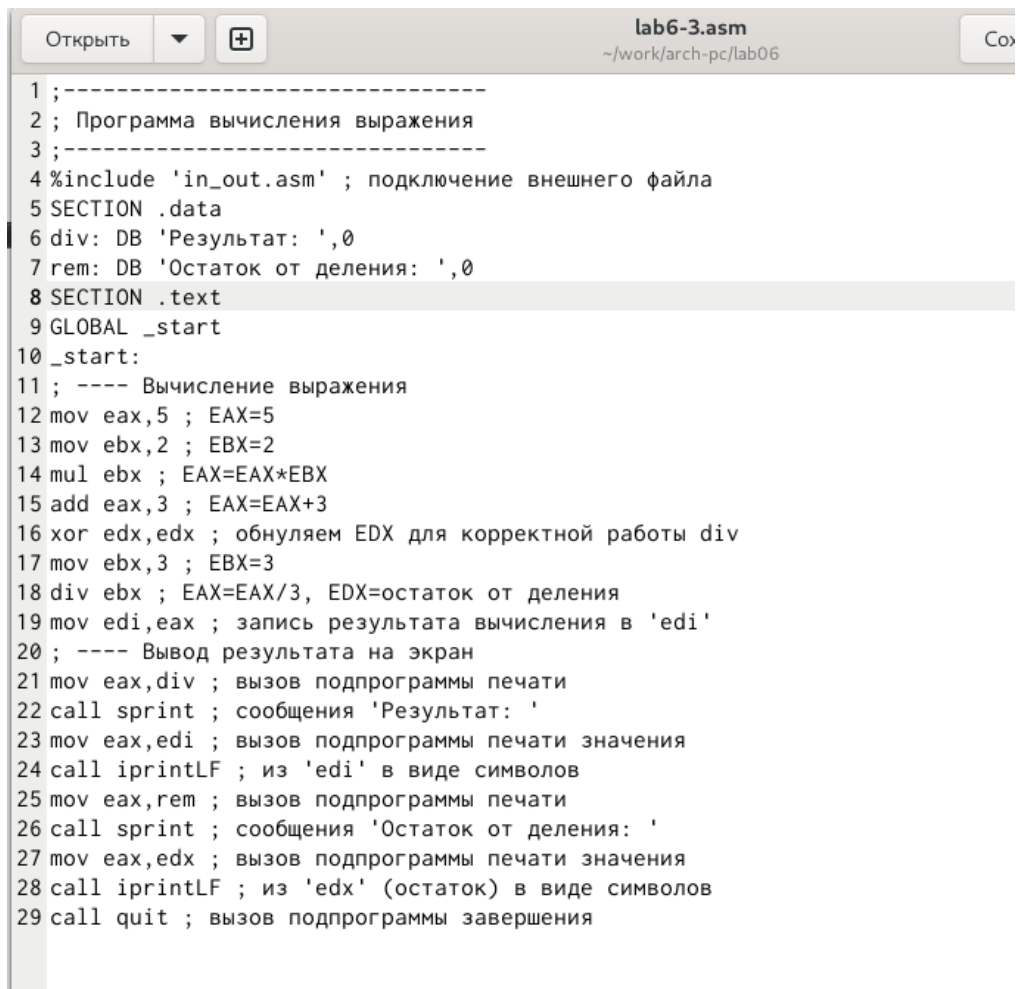
```

zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
zsmannukyan@dk3n55 ~/work/arch-pc/lab06 $ ./lab6-3

```

Рис. 9: 9.png

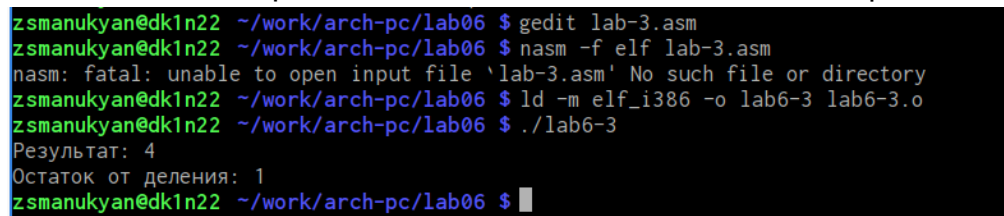
10. Введем в файл lab6-3 программу вычисления выражения .



```
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. 10: 10.png

11. Создадим файл variant.asm в каталоге ~/work/arch-pc/lab06:



```
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ gedit lab-3.asm
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ nasm -f elf lab-3.asm
nasm: fatal: unable to open input file 'lab-3.asm' No such file or directory
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $
```

Рис. 11: 11.png

12. Вводим номер студенческого и получаем вариант для выполнения задания

```

zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ touch variant.asm
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ ./variant
bash: ./variant: Нет такого файла или каталога
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
ld: невозможно найти variant.o: Нет такого файла или каталога
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132243807
Ваш вариант: 8

```

Рис. 12: 12.png

13. Составляем программу для нашего варианта lab6-4 (Самостоятельная работа).

```

mc [zsmannukyan@dk1n22.dk.sci.....edu.ru]:~/work/arch-pc/lab06
lab6-4.asm [----] 0 L: [ 1+28 29/ 29] *(1826/1826b) <EOF> [*][X]
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss ; секция не инициализированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный размер - 80 ба
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
add eax, 11; eax = eax + 11 = x + 11
mov ebx, 2 ; запись значения 2 в регистр ebx
mul ebx; EAX=EAX*EBX = (x+11)*2
add eax, -6; eax = eax - 6 = (x+11)*2 - 6
mov edi, eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 13: 13.png

14. Запускаем программу и вводим два числа из условия, убеждаемся что программа работает верно.

```

zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ touch lab6-4.asm
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ mc

zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 3
Результат: 22zsmannukyan@dk1n22 ~/work/arch-pc/lab06 $ mc

```

Рис. 14: 14.png

## 5 Ответы на вопросы:

1. строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант:  
`mov eax` и `rem call sprint`;
2. `mov ecx, x` - запись входной переменной в регистр `ecx`; `mov edx, 80` - запись размера переменной в регистр `edx`; `call sread` - вызов процедуры чтения данных;
3. `call atoi` - функция преобразующая ASCII код символа в целое число и записывающая результат в регистр `eax`;
4. `xor edx, edx` `mov ebx, 20` `div ebx`, `inc edx`;
5. `div ebx` - `ebx`;
6. `inc` - используется для увеличения операнда на единицу;
7. Следующие строки листинга отвечают за вывод на экран результата вычислений `mov eax, rem` `call sprint` `mov eax, edx` `call iprintLF`.

## 6 Выводы

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

## Список литературы