

Лабораторная работа №6

Архитектура компьютера

Башиянц Александра Кареновна

Содержание

1	Цель работы	3
2	Задание	4
3	Выполнение лабораторной работы	5
3.1	Ответы на вопросы	11
3.2	Задание для самостоятельной работы	12
4	Выводы	14

1 Цель работы

Цель работы — освоить арифметических инструкций языка ассемблера NASM.

2 Задание

В этой лабораторной работе необходимо изучить работу арифметических функций в NASM.

Необходимо научиться:

- Складывать 2 числа;
- Умножать 2 числа;
- Делить 2 числа;
- Находить целую часть деления и остаток от него;
- Вызывать прерывания с указанным номером.

Выполняя это задание, мы получим практический опыт работы с арифметическими командами NASM.

3 Выполнение лабораторной работы

Создадим директорию для 6 лабораторной работы и создадим файл lab6-1.asm (рис. 3.1).

```
akbashiyanc@fedora1:~$ mkdir ~/work/arch-pc/lab06
akbashiyanc@fedora1:~$ cd ~/work/arch-pc/lab06
akbashiyanc@fedora1:~/work/arch-pc/lab06$ touch lab6-1.asm
akbashiyanc@fedora1:~/work/arch-pc/lab06$ ls
lab6-1.asm
```

Рис. 3.1: Создание директории

Скопируем файл in_out.asm из lab05 с помощью mc (рис. 3.2).

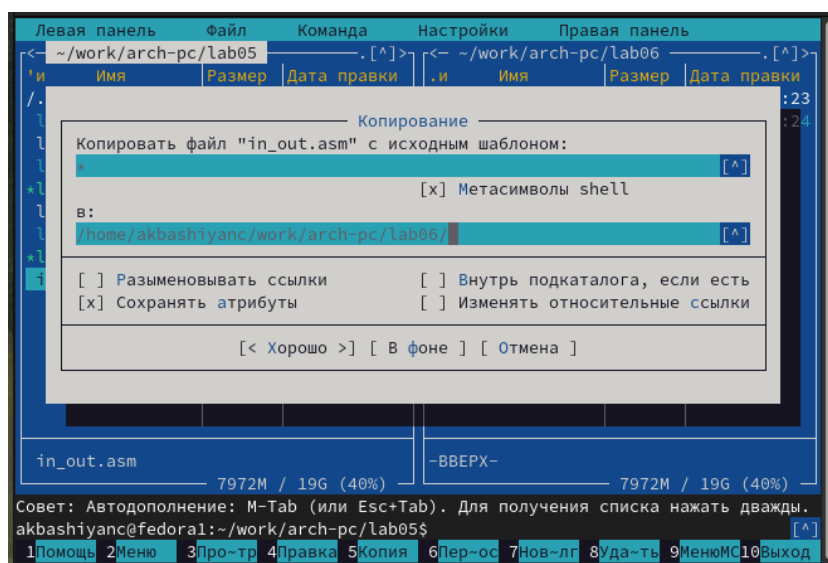


Рис. 3.2: Копирование in_out.asm

Введем код в lab6-1.asm (рис. 3.3).

```
lab6-1.asm      [-M--]  0 L: [ 1+15 16/ 17] *(190 / 204b) 0010 0x00A  [*][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
```

Рис. 3.3: Ввод кода

Создадим исполняемый файл и запустим его (рис. 3.4).

```
akbashiyanc@fedora1:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
akbashiyanc@fedora1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
akbashiyanc@fedora1:~/work/arch-pc/lab06$ ./lab6-1
j
akbashiyanc@fedora1:~/work/arch-pc/lab06$
```

Рис. 3.4: Запуск файла

Изменим текст программы и вместо символов, запишем в регистры числа (рис. 3.5).

```
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    mov [buf1],eax
    mov eax,buf1
    call sprintLF
```

Рис. 3.5: Изменение файла

Запустим измененный файл (рис. 3.6).

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

Рис. 3.6: Запуск файла

Проверим по таблице ASCII, что вывелось (рис. 3.7).

Dec	Hx	Oct	Char
0	0	000	NUL (null)
1	1	001	SOH (start of heading)
2	2	002	STX (start of text)
3	3	003	ETX (end of text)
4	4	004	EOT (end of transmission)
5	5	005	ENQ (enquiry)
6	6	006	ACK (acknowledge)
7	7	007	BEL (bell)
8	8	010	BS (backspace)
9	9	011	TAB (horizontal tab)
10	A	012	LF (NL line feed, new line)
11	B	013	VT (vertical tab)

Рис. 3.7: Проверка по таблице ASCII

Создадим файл lab6-2.asm (рис. 3.8).

```
akbashiyan@fedoral:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
```

Рис. 3.8: Создание файл

Введем код в lab6-2.asm (рис. 3.9).

```
lab6-2.asm  [----] 13 L: [ 1+10 11/ 11] *(139 / 139b) <EOF>  [*] [X]
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:
    mov eax, '6'
    mov ebx, '4'
    add eax, ebx
    call iprintLF

    call quit
```

Рис. 3.9: Ввод кода

Создадим исполняемый файл и запустим его (рис. 3.10).

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

Рис. 3.10: Запуск файла

Изменим текст программы и вместо символов, запишем в регистры числа (рис. 3.11).

```
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprintLF
```

Рис. 3.11: Изменение файла

Запустим измененный файл (рис. 3.12).

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

Рис. 3.12: Запуск файла

Заметим, что теперь программа выполняется корректно.

Изменим текст программы и вместо `iprintLF` напишем `iprint` (рис. 3.13).

```
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprint
```

Рис. 3.13: Изменение файла

Запустим измененный файл (рис. 3.14).

```
akbashiyanc@fedoral:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
akbashiyanc@fedoral:~/work/arch-pc/lab06$ ./lab6-2
10
```

Рис. 3.14: Запуск файла

Заметим, что при изменении `iprintLF` на `iprint` вывод программы не изменился. Создадим файл `lab6-3.asm` (рис. 3.15).

```
akbashiyanc@fedora1:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
```

Рис. 3.15: Создание файл

Вычислим арифметическое выражения $f(x)=(5*2+3)/3$. Введем код в `lab6-3.asm` (рис. 3.16).

```
lab6-3.asm  [----]  0 L: [ 1+ 0 1/ 31] *(0 /1435b) 0059 0x03B [*][X]
;
; -----
; 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
```

Рис. 3.16: Ввод кода

Создадим исполняемый файл и запустим его (рис. 3.17).

```
akbashiyanc@fedora1:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
akbashiyanc@fedora1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
akbashiyanc@fedora1:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 3.17: Запуск файла

Изменим текст программы так, чтобы мы вычислили $f(x)=(4*6+2)/5$ (рис. 3.18).

```
; ---- 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/5, EDX=остаток от деления
```

Рис. 3.18: Изменение файла

Запустим измененный файл (рис. 3.19).

```

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

```

Рис. 3.19: Запуск файла

Создадим файл variant.asm (рис. 3.20).

```

akbashiyan@fedoral:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm

```

Рис. 3.20: Создание файл

Введем код в variant.asm (рис. 3.21).

```

variant.asm  [----]  0 L:[ 1+ 3  4/ 32] *(127 / 686b) 0037 0x025  [*][X]
;-----
; Программа вычисления варианта
;-----
%include 'in_out.asm'

SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0

```

Рис. 3.21: Ввод кода

Создадим исполняемый файл и запустим его (рис. 3.22).

```

akbashiyan@fedoral:~/work/arch-pc/lab06$ nasm -f elf variant.asm
akbashiyan@fedoral:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
akbashiyan@fedoral:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246777
Ваш вариант: 18

```

Рис. 3.22: Запуск файла

Проверим правильность выполнения программы с помощью калькулятора (рис. 3.23).

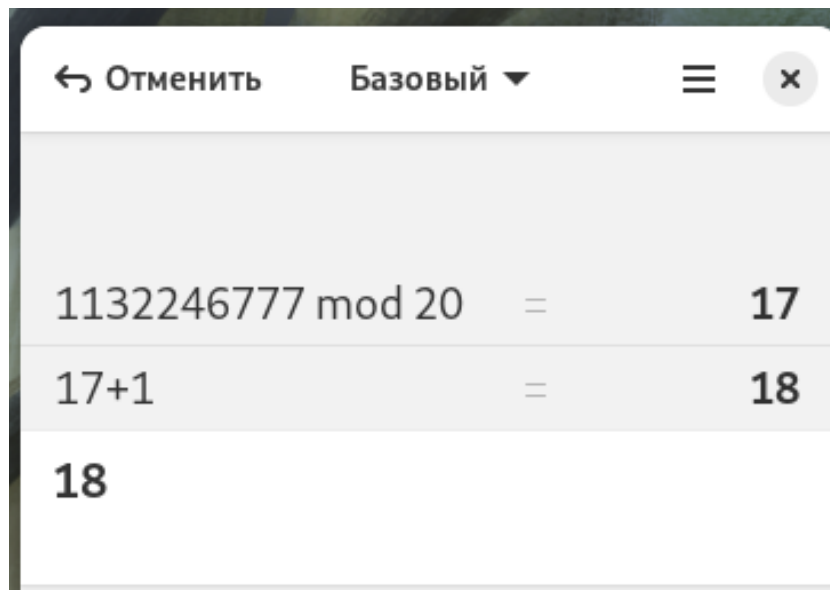


Рис. 3.23: Проверка подсчетов

3.1 Ответы на вопросы

1. Строки “mov eax,rem ; call sprint” отвечают за вывод на экран сообщения ‘Ваш вариант:’.
2. Эта инструкция используется для получения данных из переменной х.
3. call atoi используется для превращение ASCII кода в число.
4. За вычисления варианта отвечают строки:

```
xor edx,edx  
mov ebx,20  
div ebx  
inc edx
```

5. Остаток от деления при выполнении инструкции “div ebx” записывается в регистр eax.

6. Для чего используется Инструкция “inc edx” используется для увеличение операнда на 1.

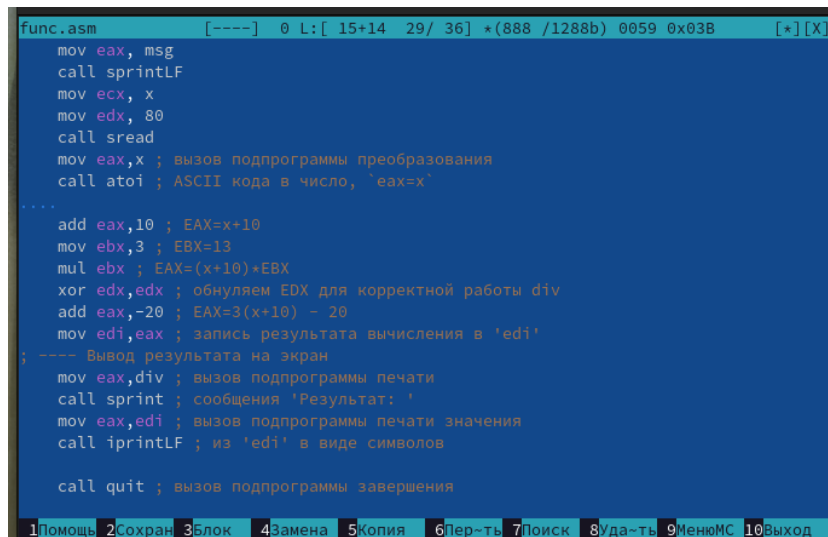
3.2 Задание для самостоятельной работы

Создадим файл func.asm (рис. 3.24).

```
akbashiyan@fedora1:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/func.asm
```

Рис. 3.24: Создание файл

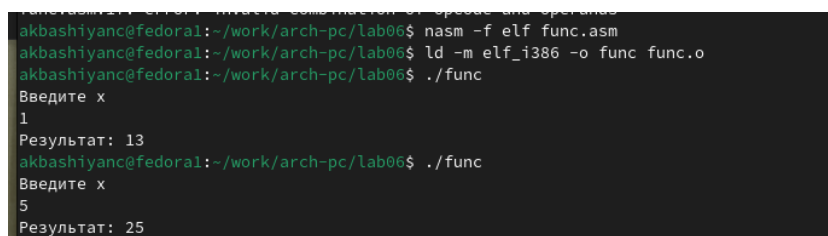
Введем код для подсчета функции $f(x)=3(x+10)-20$ (вариант 18) в func.asm (рис. 3.25).



```
func.asm  [----]  0 L: [ 15+14  29/ 36] *(888 /1288b) 0059 0x03B  [*] [X]
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'
....
add eax, 10 ; EAX=x+10
mov ebx, 3 ; EBX=13
mul ebx ; EAX=(x+10)*EBX
xor edx, edx ; обнуляем EDX для корректной работы div
add eax, -20 ; EAX=3(x+10) - 20
mov edi, eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax, div ; вызов подпрограммы печати
call sprintf ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call sprintf ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 3.25: Ввод кода

Создадим исполняемый файл и запустим его (рис. 3.26).



```
akbashiyan@fedora1:~/work/arch-pc/lab06$ nasm -f elf func.asm
akbashiyan@fedora1:~/work/arch-pc/lab06$ ld -m elf_i386 -o func func.o
akbashiyan@fedora1:~/work/arch-pc/lab06$ ./func
Введите x
1
Результат: 13
akbashiyan@fedora1:~/work/arch-pc/lab06$ ./func
Введите x
5
Результат: 25
```

Рис. 3.26: Запуск файла

Проверим правильность выполнения программы с помощью калькулятора (рис. 3.27).

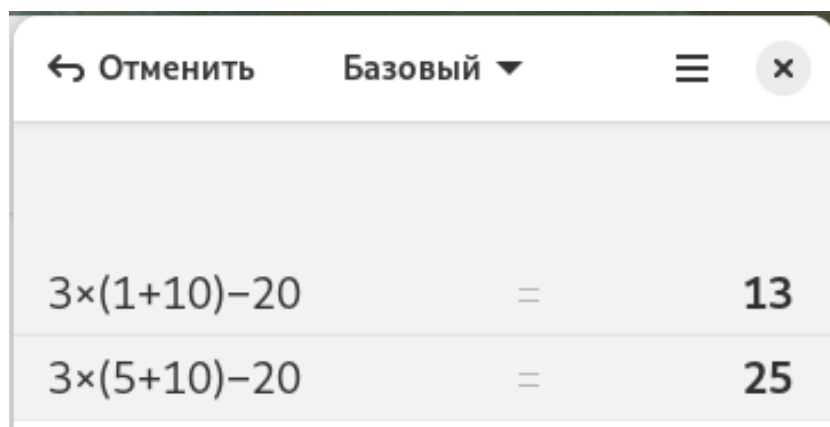


Рис. 3.27: Проверка подсчетов

4 Выводы

В ходе выполнения работы были получены навыки практической работы с арифметическими функциями в NASM.