

Отчёт по лабораторной работе 6

Архитектура компьютеров

Богдан Гаряев НБИбд-01-2025

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
2.1 Символьные и численные данные в NASM	6
2.2 Выполнение арифметических операций в NASM	11
2.3 Ответы на вопросы	15
2.4 Задание для самостоятельной работы	16
3 Выводы	19

Список иллюстраций

2.1 Программа lab6-1.asm	7
2.2 Запуск программы lab6-1.asm	7
2.3 Программа lab6-1.asm с числами	8
2.4 Запуск программы lab6-1.asm с числами	8
2.5 Программа lab6-2.asm	9
2.6 Запуск программы lab6-2.asm	9
2.7 Программа lab6-2.asm с числами	10
2.8 Запуск программы lab6-2.asm с числами	10
2.9 Запуск программы lab6-2.asm без переноса строки	11
2.10 Программа lab6-3.asm	12
2.11 Запуск программы lab6-3.asm	12
2.12 Программа lab6-3.asm с другим выражением	13
2.13 Запуск программы lab6-3.asm с другим выражением	13
2.14 Программа variant.asm	14
2.15 Запуск программы variant.asm	15
2.16 Программа calc.asm	17
2.17 Запуск программы calc.asm	18

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

2.1 Символьные и численные данные в NASM

Создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm.

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

В данной программе в регистр eax записывается символ 6 (используя команду mov eax, „6“), в регистр ebx записывается символ 4 (используя команду mov ebx, „4“). Далее к значению в регистре eax прибавляем значение регистра ebx (командой add eax, ebx, результат сложения запишется в регистр eax). После этого выводим результат. (рис. 2.1) (рис. 2.2)

Так как для работы функции sprintLF в регистр eax должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра eax в переменную buf1 (командой mov [buf1], eax), а затем запишем адрес переменной buf1 в регистр eax (командой mov eax, buf1) и вызовем функцию sprintLF.

The screenshot shows a code editor window with the file 'lab6-1.asm' open. The file path is 'home > brgaryaev > work > arch-pc > lab06 > lab6-1.asm'. The code itself is as follows:

```
1 %include 'in_out.asm'
2 SECTION .bss
3 bufl: 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 sprintLF
13    call quit
14
```

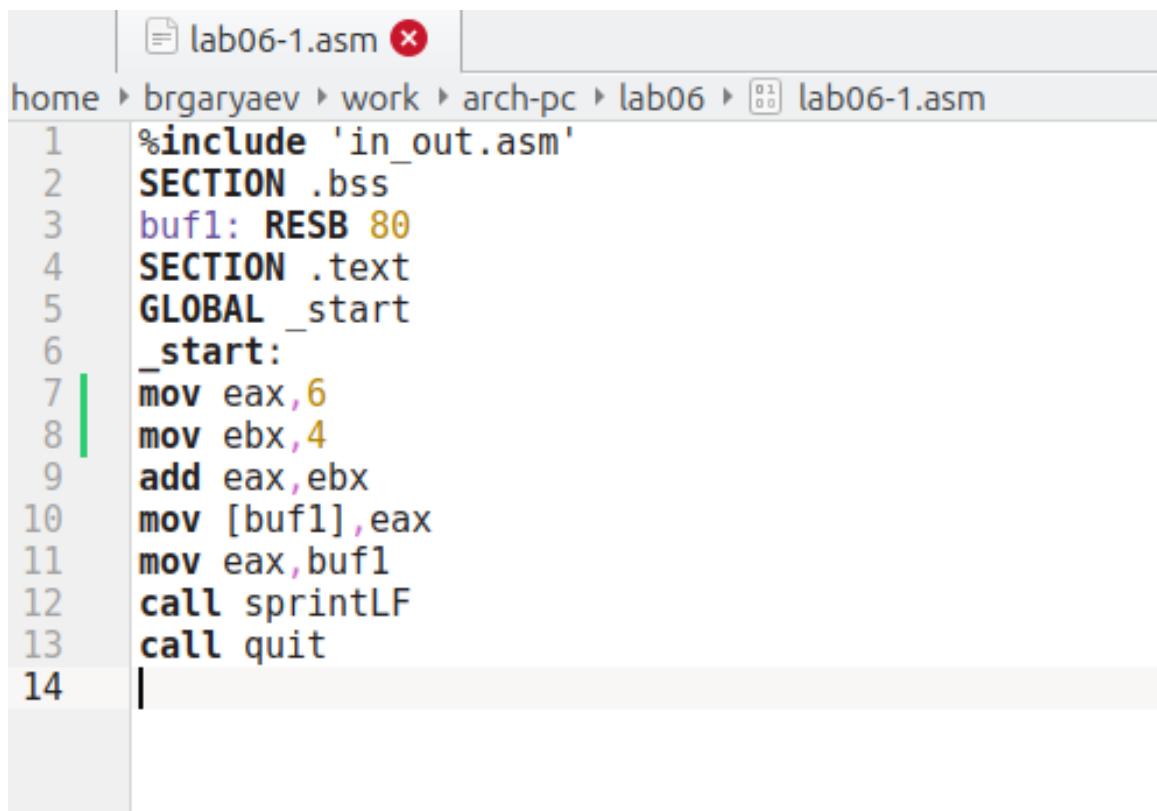
Рисунок 2.1: Программа lab6-1.asm

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

Рисунок 2.2: Запуск программы lab6-1.asm

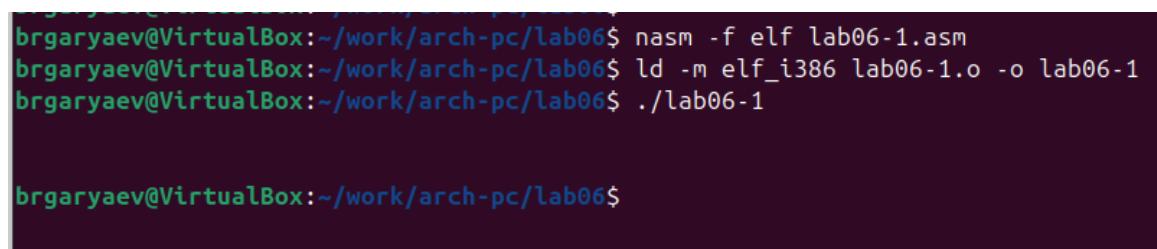
В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда add eax, ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j.

Далее изменяю текст программы и вместо символов запишу в регистры числа. (рис. 2.3) (рис. 2.4)



```
%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
```

Рисунок 2.3: Программа lab6-1.asm с числами



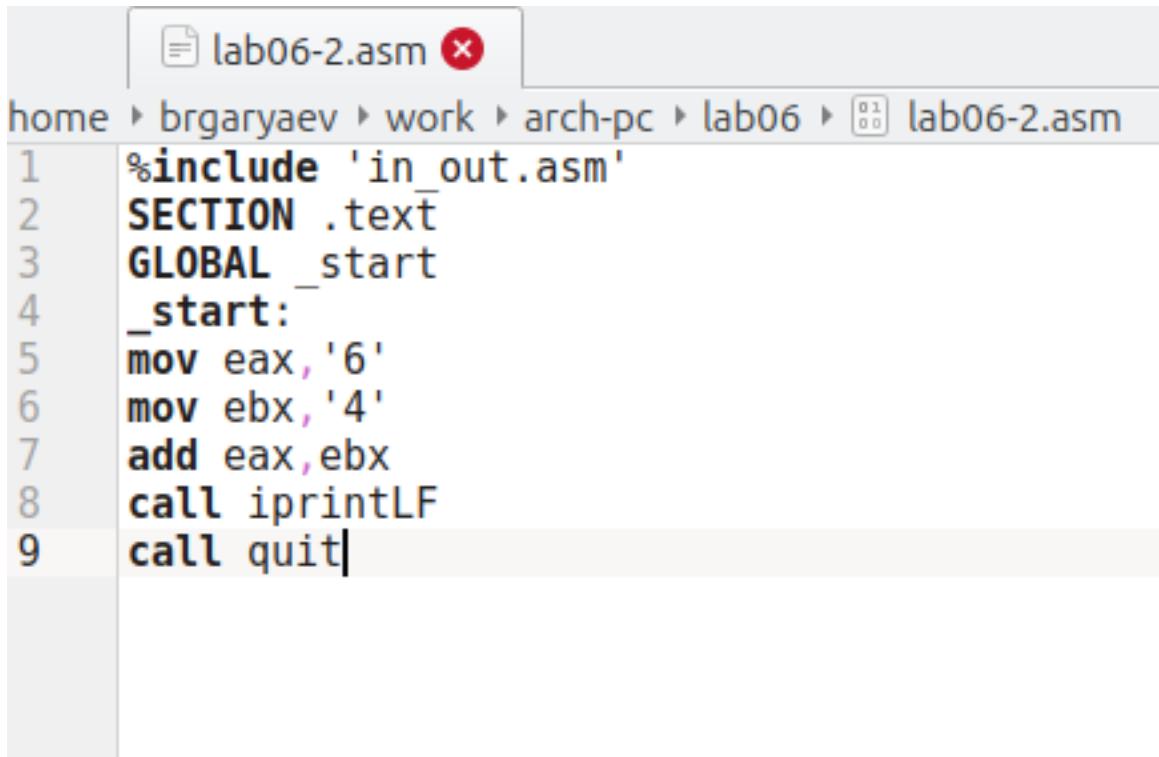
```
brgaryaev@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
brgaryaev@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
brgaryaev@VirtualBox:~/work/arch-pc/lab06$ ./lab6-1

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

Рисунок 2.4: Запуск программы lab6-1.asm с числами

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Это символ конца строки (возврат каретки). В консоли он не отображается, но добавляет пустую строку.

Как отмечалось выше, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразую текст программы с использованием этих функций. (рис. 2.5) (рис. 2.6)



```
lab06-2.asm
home ▶ brgargyaev ▶ work ▶ arch-pc ▶ lab06 ▶ lab06-2.asm
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
```

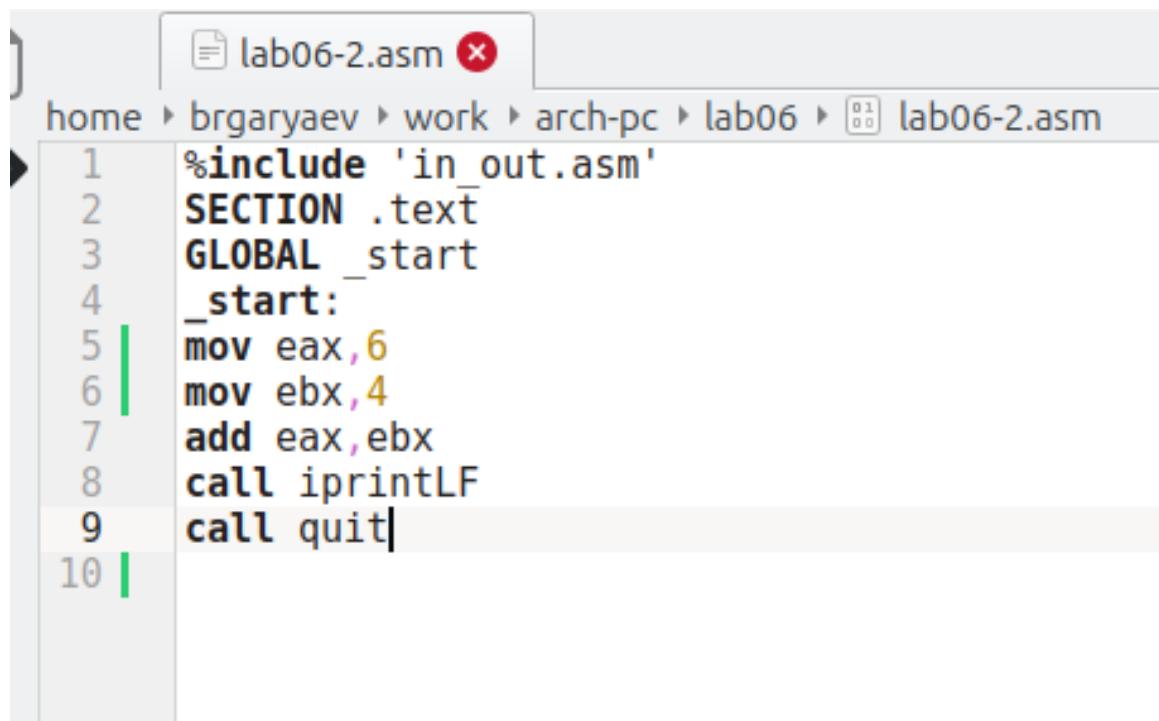
Рисунок 2.5: Программа `lab6-2.asm`

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

Рисунок 2.6: 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 отличие от прошлой программы, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа. (рис. 2.7) (рис. 2.8)



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

Рисунок 2.7: Программа lab6-2.asm с числами

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

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

Рисунок 2.8: Запуск программы lab6-2.asm с числами

Заменил функцию iprintLF на iprint. Создал исполняемый файл и запустил его. Вывод отличается тем, что нет переноса строки. (рис. 2.9)

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

Рисунок 2.9: Запуск программы lab6-2.asm без переноса строки

2.2 Выполнение арифметических операций в NASM

В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$. (рис. 2.10) (рис. 2.11)

The screenshot shows a code editor window with the file 'lab06-3.asm' open. The file path is 'home > brgaryaeve > work > arch-pc > lab06 > lab06-3.asm'. The assembly code is as follows:

```
1 %include 'in_out.asm'
2 SECTION .data
3 div: DB 'Результат: ',0
4 rem: DB 'Остаток от деления: ',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8
9 mov eax,5
10 mov ebx,2
11 mul ebx
12 add eax,3
13 xor edx,edx
14 mov ebx,3
15 div ebx
16 mov edi,eax
17 mov eax,div
18 call sprint
19 mov eax,edi
20 call iprintLF
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
25 call quit
26
```

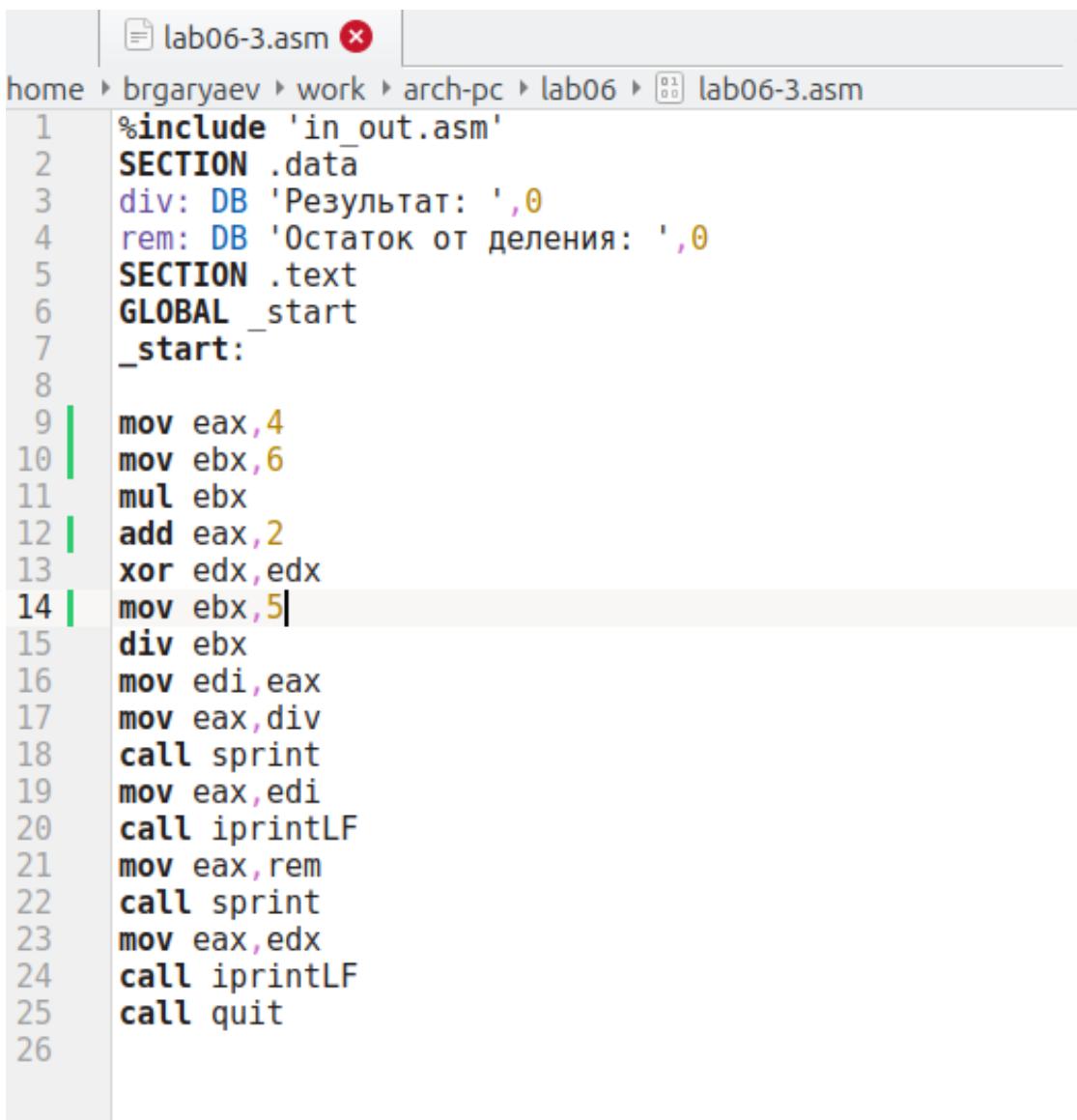
Рисунок 2.10: Программа lab6-3.asm

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

Рисунок 2.11: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$.

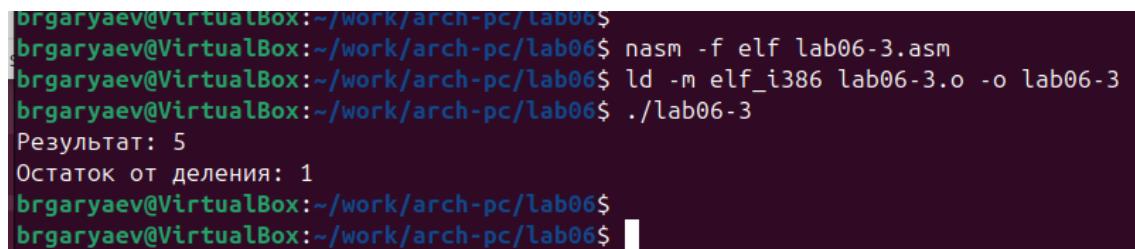
Создал исполняемый файл и проверил его работу. (рис. 2.12) (рис. 2.13)



The screenshot shows a text editor window with the file 'lab6-3.asm' open. The code is written in assembly language. The assembly code includes directives like %include, SECTION, and GLOBAL, along with various instructions such as mov, add, xor, and call. The code is numbered from 1 to 26. The assembly code is as follows:

```
%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
```

Рисунок 2.12: Программа lab6-3.asm с другим выражением



The screenshot shows a terminal window with the following command-line session:

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

Рисунок 2.13: Запуск программы lab6-3.asm с другим выражением

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета. (рис. 2.14) (рис. 2.15)

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше, ввод с клавиатуры осуществляется в символьном виде, и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция atoi из файла in_out.asm.

```
%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
call atoi
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprint
mov eax, edx
call iprintLF
call quit
```

Рисунок 2.14: Программа variant.asm

```
brgargyaev@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
brgargyaev@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
brgargyaev@VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032251945
Ваш вариант: 6
brgargyaev@VirtualBox:~/work/arch-pc/lab06$
```

Рисунок 2.15: Запуск программы variant.asm

2.3 Ответы на вопросы

1. Какие строки листинга отвечают за вывод на экран сообщения „Ваш вариант:“?
 - Инструкция mov eax, rem перекладывает значение переменной с фразой „Ваш вариант:“ в регистр eax.
 - Инструкция call sprint вызывает подпрограмму для вывода строки.
2. Для чего используются следующие инструкции?
 - Инструкция mov ecx, x используется для перемещения значения переменной x в регистр ecx.
 - Инструкция mov edx, 80 используется для перемещения значения 80 в регистр edx.
 - Инструкция call sread вызывает подпрограмму для считывания значения студенческого билета из консоли.
3. Для чего используется инструкция «call atoi»?
 - Инструкция «call atoi» используется для преобразования введенных символов в числовой формат.
4. Какие строки листинга отвечают за вычисления варианта?
 - Инструкция xor edx, edx обнуляет регистр edx.

- Инструкция `mov ebx, 20` записывает значение 20 в регистр ebx.
- Инструкция `div ebx` выполняет деление номера студенческого билета на 20.
- Инструкция `inc edx` увеличивает значение регистра edx на 1.

Здесь происходит деление номера студенческого билета на 20. В регистре edx хранится остаток, к нему прибавляется 1.

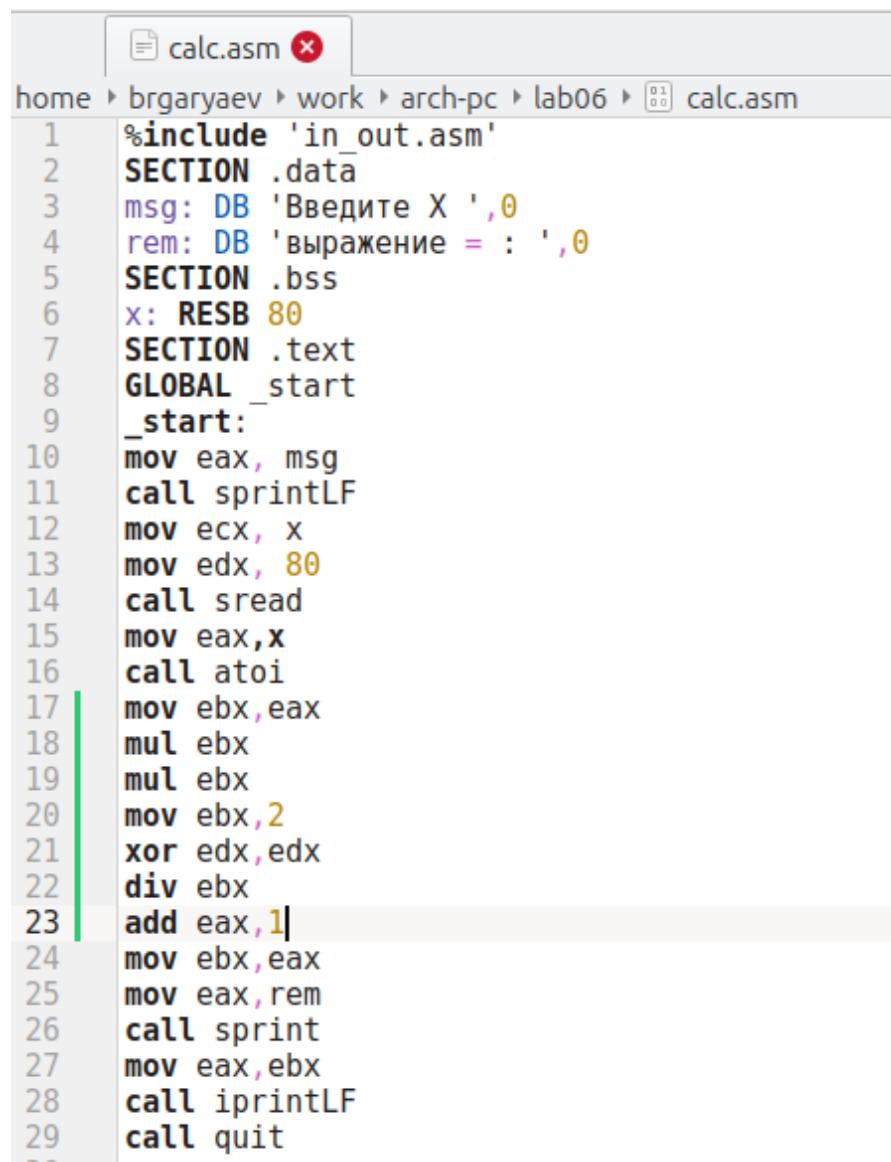
5. В какой регистр записывается остаток от деления при выполнении инструкции «`div ebx`»?
 - Остаток от деления записывается в регистр edx.
6. Для чего используется инструкция «`inc edx`»?
 - Инструкция «`inc edx`» используется для увеличения значения в регистре edx на 1, согласно формуле вычисления варианта.
7. Какие строки листинга отвечают за вывод на экран результата вычислений?
 - Инструкция `mov eax, edx` перекладывает результат вычислений в регистр eax.
 - Инструкция `call iprintLF` вызывает подпрограмму для вывода значения на экран.

2.4 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. 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. (рис. 2.16) (рис. 2.17)

Получили вариант $6 - \frac{x^3}{2} + 1$ для $x=2, x=5$



The screenshot shows a code editor window with the file 'calc.asm' open. The file path is 'home > brgaryaev > work > arch-pc > lab06 > calc.asm'. The assembly code is as follows:

```
%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
    mul ebx
    mul ebx
    mov ebx, 2
    xor edx, edx
    div ebx
    add eax, 1
    mov ebx, eax
    mov eax, rem
    call sprint
    mov eax, ebx
    call iprintLF
    call quit
```

Рисунок 2.16: Программа calc.asm

При $x=2$ получается 5.

При $x=5$ получается 63.

```
brgaryaev@VirtualBox:~/work/arch-pc/lab06$ nasm -f elf calc.asm
brgaryaev@VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 calc.o -o calc
brgaryaev@VirtualBox:~/work/arch-pc/lab06$ ./calc
Введите X
2
выражение = : 5
brgaryaev@VirtualBox:~/work/arch-pc/lab06$ ./calc
Введите X
5
выражение = : 63
brgaryaev@VirtualBox:~/work/arch-pc/lab06$ █
```

Рисунок 2.17: Запуск программы calc.asm

Программа считает верно.

3 Выводы

Изучили работу с арифметическими операциями.