

Отчёт по лабораторной работе 6

Архитектура компьютера

Тумуреева Галина Аркадьевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	22

Список иллюстраций

2.1	Программа в файле lab6-1.asm	7
2.2	Запуск программы lab6-1.asm	7
2.3	Программа в файле lab6-1.asm	8
2.4	Запуск программы lab6-1.asm	9
2.5	Программа в файле lab6-2.asm	10
2.6	Запуск программы lab6-2.asm	10
2.7	Программа в файле lab6-2.asm	11
2.8	Запуск программы lab6-2.asm	11
2.9	Запуск программы lab6-2.asm	12
2.10	Программа в файле lab6-3.asm	13
2.11	Запуск программы lab6-3.asm	14
2.12	Программа в файле lab6-3.asm	15
2.13	Запуск программы lab6-3.asm	15
2.14	Программа в файле variant.asm	17
2.15	Запуск программы variant.asm	17
2.16	Программа в файле task.asm	20
2.17	Запуск программы task.asm	21

Список таблиц

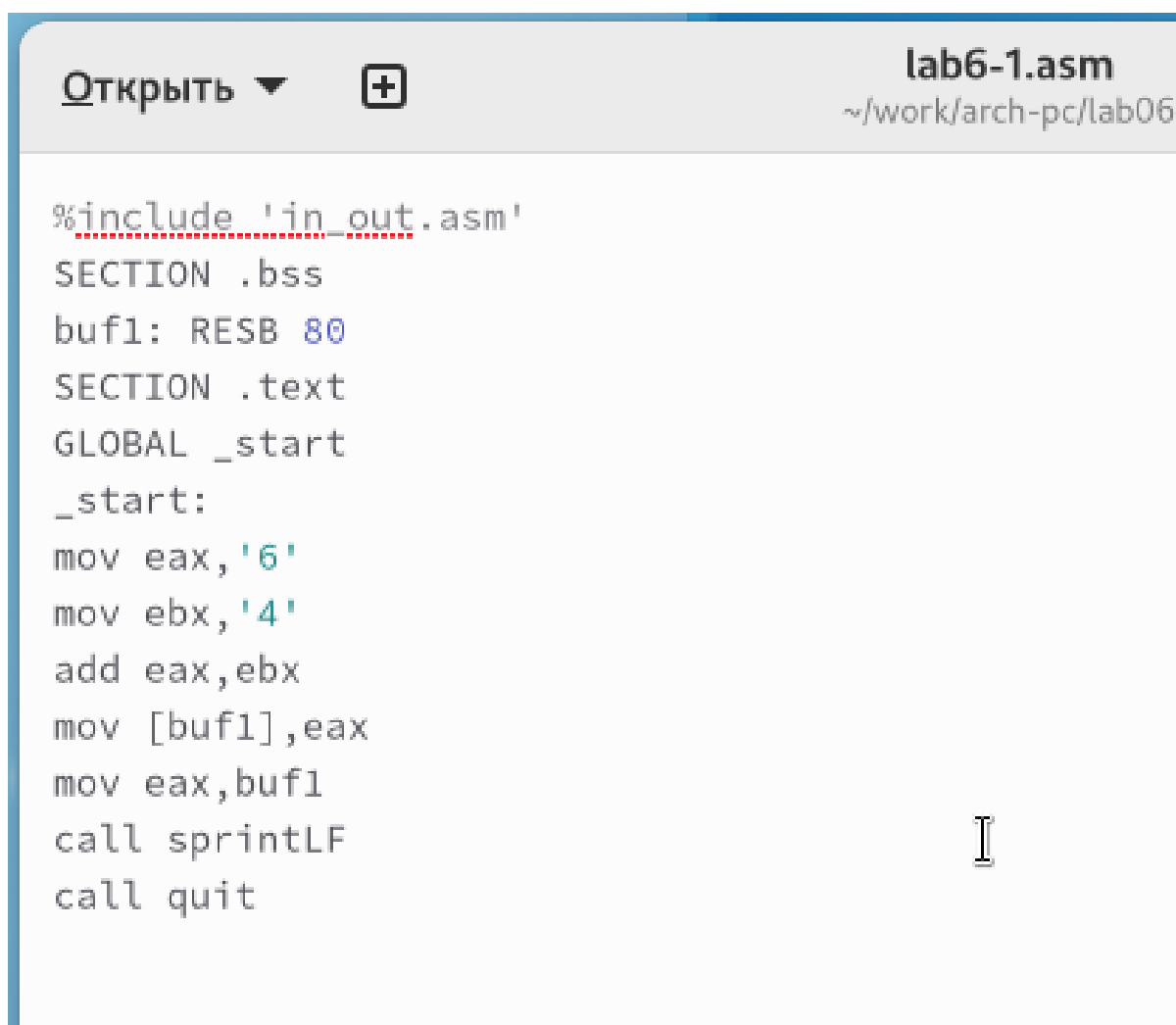
1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

1. Я создала папку для программ лабораторной работы номер шесть, затем перешла в неё и сформировала файл с именем lab6-1.asm.
2. Давайте рассмотрим примеры программ, которые отображают символы и числовые данные. Эти программы будут выводить информацию, которая была помещена в регистр `eax`.

В одной из программ в регистр `eax` мы помещаем символ '6' (`mov eax, '6'`), а в регистр `ebx` символ '4' (`mov ebx, '4'`). После этого мы складываем значения, хранящиеся в регистрах `eax` и `ebx` (`add eax, ebx`, и результат сложения сохранится в `eax`). Затем мы выводим полученный результат на экран. Однако, поскольку функция `sprintf` требует, чтобы в регистре `eax` находился адрес, нам нужно воспользоваться дополнительной переменной. Сначала мы переносим значение из регистра `eax` в переменную `buf1` (`mov [buf1],eax`), а потом записываем адрес переменной `buf1` обратно в регистр `eax` (`mov eax, buf1`) и вызываем функцию `sprintf`.



```
lab6-1.asm
~/work/arch-pc/lab06

%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. 2.1: Программа в файле lab6-1.asm



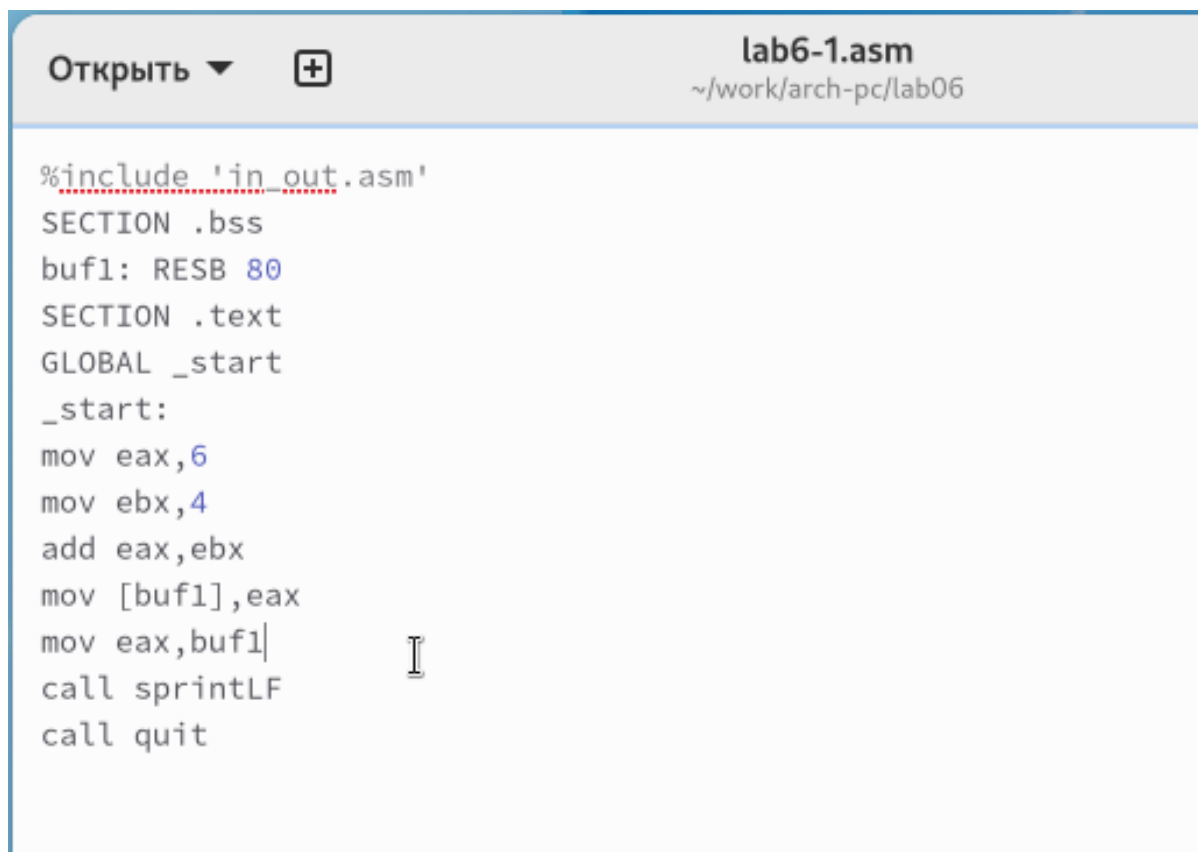
```
[gtumureeva@gtumureeva lab06]$ nasm -f elf lab6-1.asm
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[gtumureeva@gtumureeva lab06]$ ./lab6-1
j
[gtumureeva@gtumureeva lab06]$
```

Рис. 2.2: Запуск программы lab6-1.asm

Когда я смотрю на значение в регистре `eax`, я ожидаю увидеть цифру 10. Но вместо этого там отображается символ 'j'. Это происходит из-за того, что дво-

ичный код символа '6' равен 00110110, что соответствует числу 54, а двоичный код символа '4' – 00110100, или 52. Когда я использую команду add eax, ebx, то в регистр eax записывается их сумма – 01101010, что в десятичной системе равно 106, и это код для символа 'j'.

3. Затем я внесла изменения в программу, чтобы в регистры записывались числа, а не символы.




```
Открыть ▾  lab6-1.asm  
~/work/arch-pc/lab06  
  
%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. 2.3: Программа в файле lab6-1.asm

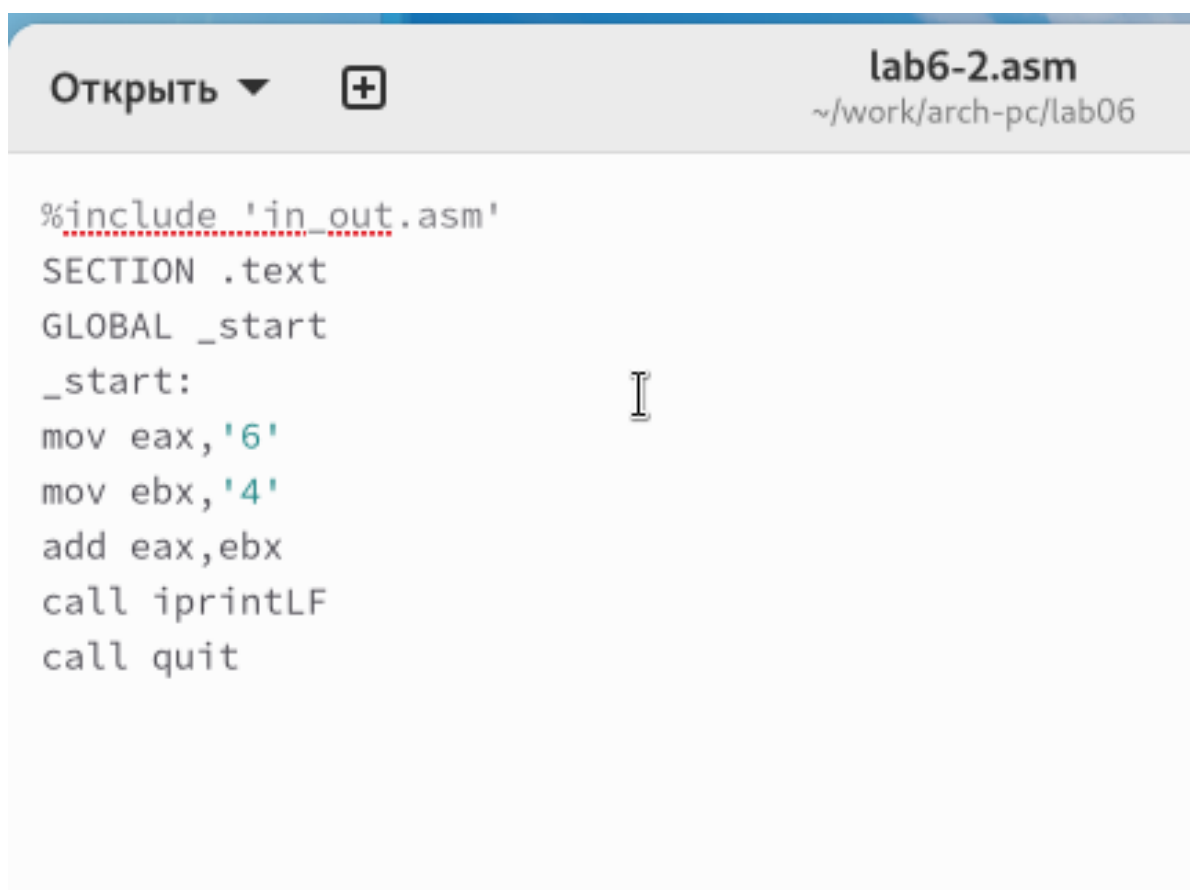

```
[gtumureeva@gtumureeva lab06]$ nasm -f elf lab6-1.asm
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 lab6-1.o -o lab6-1
[gtumureeva@gtumureeva lab06]$ ./lab6-1

[gtumureeva@gtumureeva lab06]$
```

Рис. 2.4: Запуск программы lab6-1.asm

Но даже после этих изменений, когда программа выполняется, она не показывает число 10. В этот раз выводится символ с кодом 10, который является символом конца строки. В консоли он не виден, но создает пустую строку.

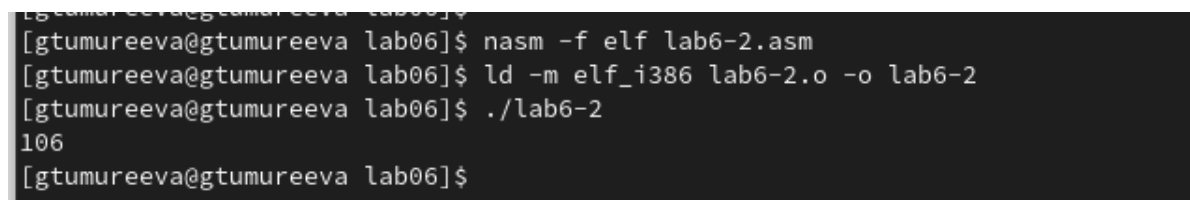
4. Как я уже упоминала, для работы с числами в файле `in_out.asm` были реализованы специальные подпрограммы, которые позволяют преобразовывать ASCII символы в числа и наоборот. Я использовала эти функции, чтобы изменить текст программы.



```
Открыть ▾ + lab6-2.asm
~/work/arch-pc/lab06

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

Рис. 2.5: Программа в файле lab6-2.asm

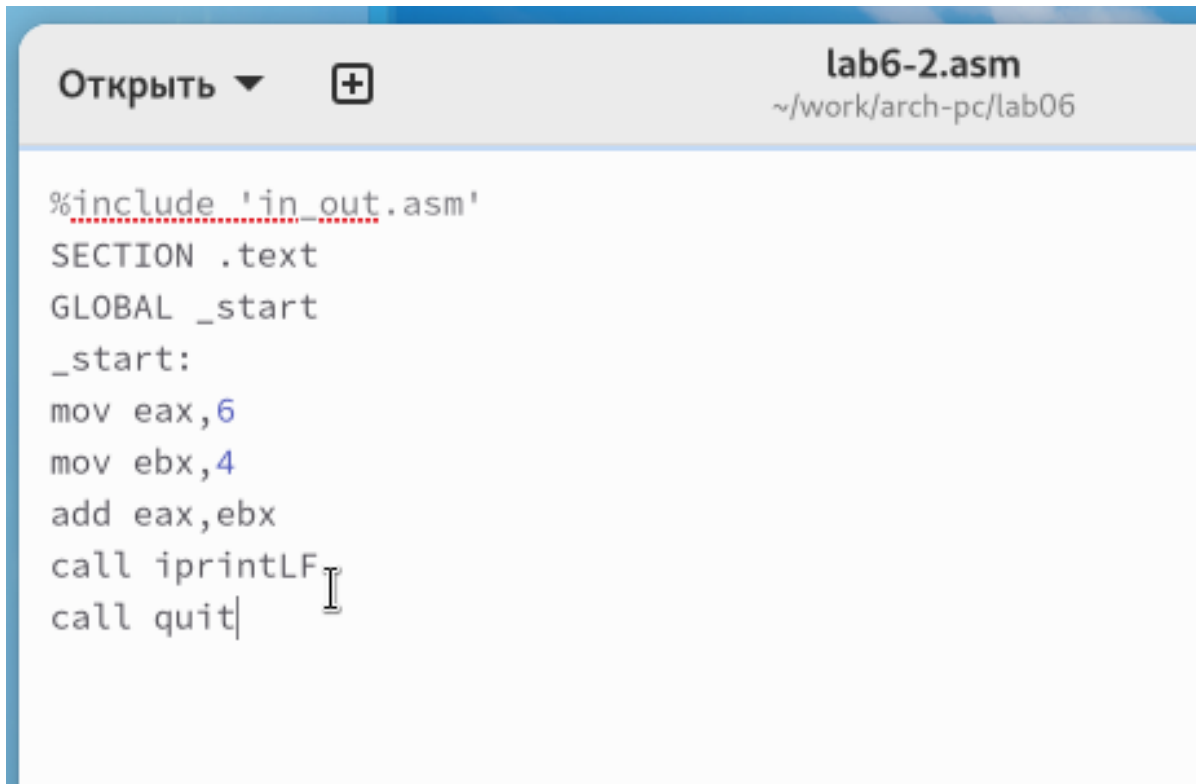


```
[gtumureeva@gtumureeva lab06]$ nasm -f elf lab6-2.asm
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[gtumureeva@gtumureeva lab06]$ ./lab6-2
106
[gtumureeva@gtumureeva lab06]$
```

Рис. 2.6: Запуск программы lab6-2.asm

Когда я запустила программу, она выдала мне число 106. Так же, как в первом случае, здесь функция `add` суммирует коды символов '6' и '4', что в сумме даёт $54+52=106$. Но в отличие от предыдущей программы, здесь используется функция `iprintLF`, которая позволяет выводить число, а не символ, соответствующий этому числовому коду.

5. Подобно предыдущему примеру, я заменила символы на числа.

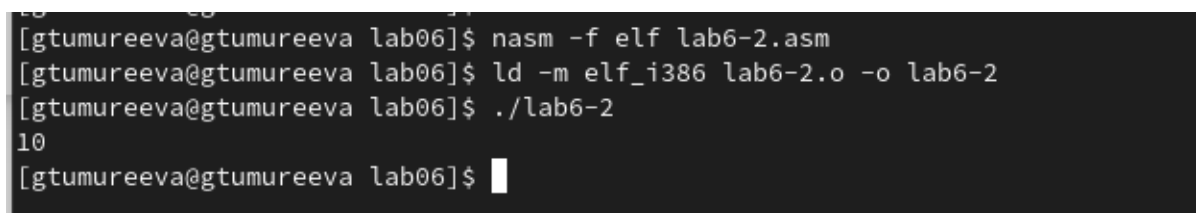


```
Открыть ▾ [icon] lab6-2.asm
~/work/arch-pc/lab06

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

Рис. 2.7: Программа в файле lab6-2.asm

Благодаря функции iprintLF, которая выводит числа, и тому, что в качестве операндов были использованы именно числа, а не коды символов, в результате получилось число 10.



```
[gtumureeva@gtumureeva lab06]$ nasm -f elf lab6-2.asm
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[gtumureeva@gtumureeva lab06]$ ./lab6-2
10
[gtumureeva@gtumureeva lab06]$
```

Рис. 2.8: Запуск программы lab6-2.asm

Я изменила функцию iprintLF на iprint, собрала исполняемый файл и запустила его. Отличие заключалось в том, что теперь вывод не сопровождался переносом

строки.


```
[gtumureeva@gtumureeva lab06]$ nasm -f elf lab6-2.asm
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 lab6-2.o -o lab6-2
[gtumureeva@gtumureeva lab06]$ ./lab6-2
10[gtumureeva@gtumureeva lab06]$
[gtumureeva@gtumureeva lab06]$
```

Рис. 2.9: Запуск программы lab6-2.asm

6. В качестве примера, демонстрирующего выполнение арифметических операций в NASM, я написала программу для вычисления арифметического выражения

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

.

Открыть ▾ 

lab6-3.asm
~/work/arch-pc/lab06

```
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
```

Рис. 2.10: Программа в файле lab6-3.asm

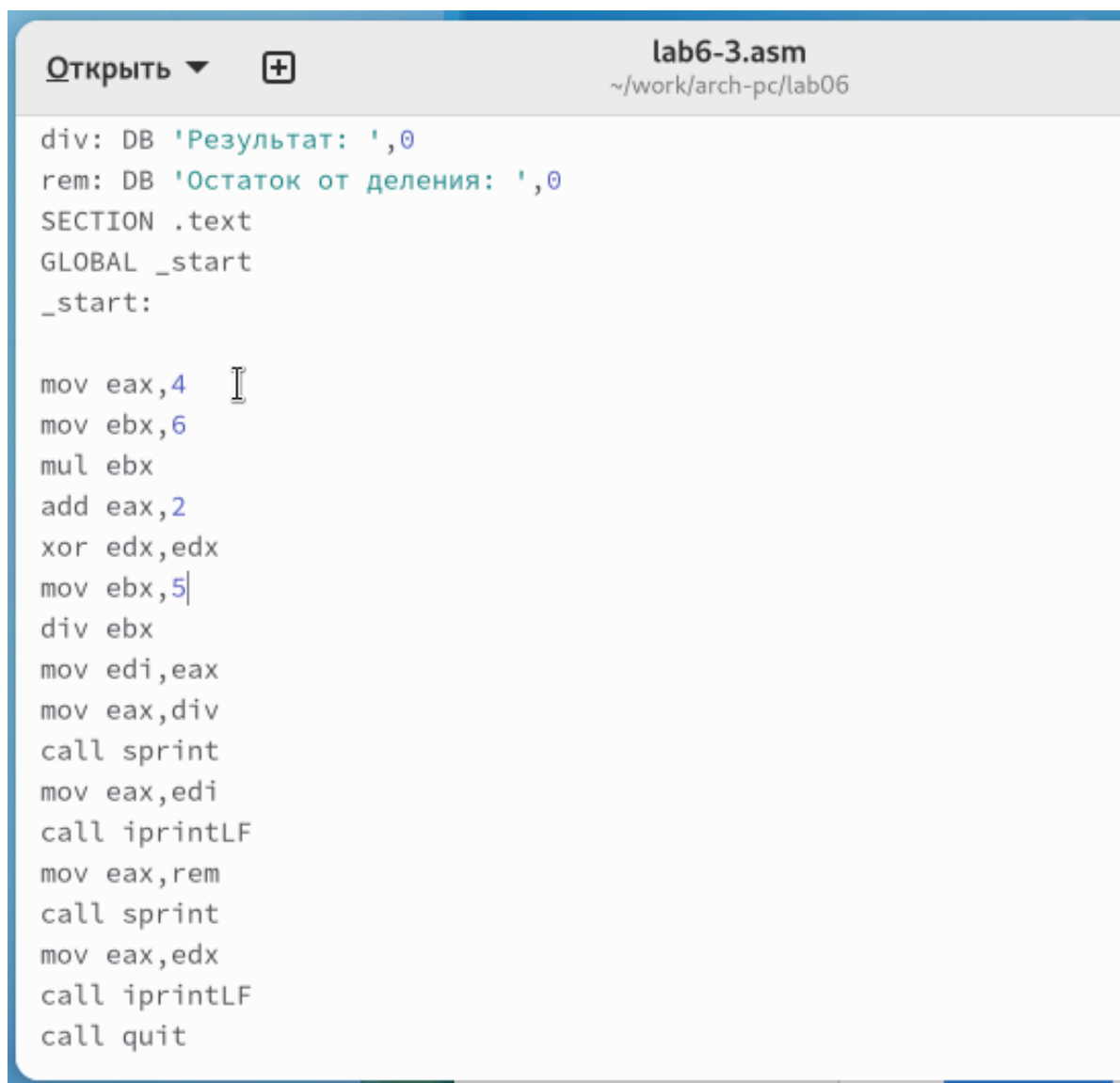
```
[gtumureeva@gtumureeva lab06]$  
[gtumureeva@gtumureeva lab06]$ nasm -f elf lab6-3.asm  
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3  
[gtumureeva@gtumureeva lab06]$ ./lab6-3  
Результат: 4  
Остаток от деления: 1  
[gtumureeva@gtumureeva lab06]$
```

Рис. 2.11: Запуск программы lab6-3.asm

Затем я изменила код программы, чтобы она вычисляла выражение

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

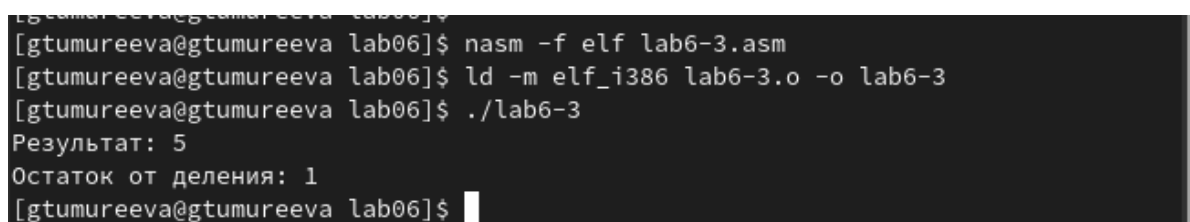
. После создания исполняемого файла я проверила, как он работает.



```
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
```

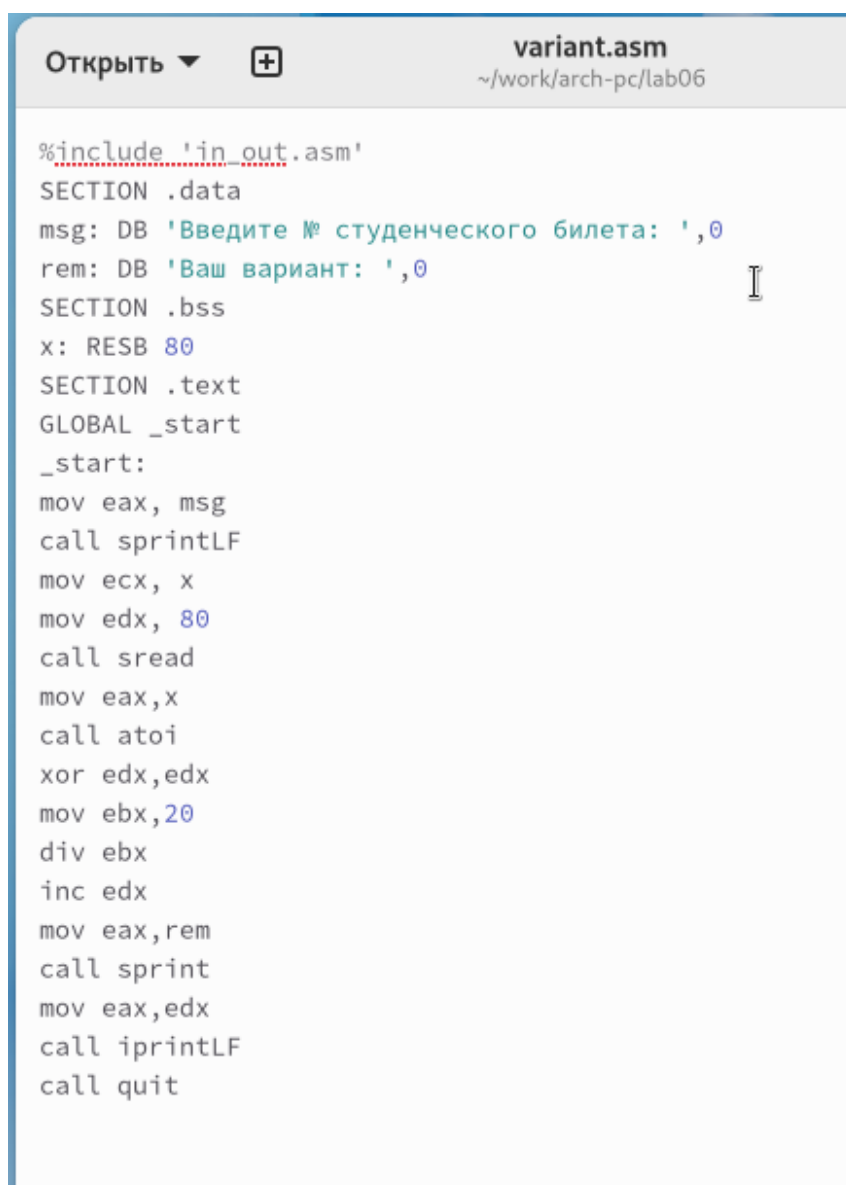
Рис. 2.12: Программа в файле lab6-3.asm



```
[gtumureeva@gtumureeva lab06]$ nasm -f elf lab6-3.asm
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 lab6-3.o -o lab6-3
[gtumureeva@gtumureeva lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[gtumureeva@gtumureeva lab06]$
```

Рис. 2.13: Запуск программы lab6-3.asm

7. Давайте возьмем для примера задачу, где нужно вычислить вариант упражнения на основе номера студенческого билета. Здесь нам придется работать с числом, которое мы введем через клавиатуру. Как я уже упоминала ранее, вводимые данные поступают в виде символов, и чтобы выполнять с ними математические операции в NASM, их нужно преобразовать в числовой формат. В этом может помочь функция `atoi`, которую можно найти в файле `in_out.asm`.



```
variant.asm
~/work/arch-pc/lab06

%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 sprintf
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 sprintf
mov eax, edx
call iprintLF
call quit
```

Рис. 2.14: Программа в файле variant.asm



```
[gtumureeva@gtumureeva lab06]$ nasm -f elf variant.asm
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 variant.o -o variant
[gtumureeva@gtumureeva lab06]$ ./variant
Введите № студенческого билета:
1132239112
Ваш вариант: 13
[gtumureeva@gtumureeva lab06]$
```

Рис. 2.15: Запуск программы variant.asm

ответы на вопросы

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

- Команда `mov eax, ret` загружает в регистр значение, соответствующее строке “Ваш вариант:”.
- Использование `call sprint` приводит к выполнению функции, отображающей строку.

2. Для чего используются следующие инструкции?

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

Они используются для ввода номера студенческого билета и его сохранения в переменной X через терминал.

3. Для чего используется инструкция “call atoi”?

Данная функция преобразует введенные пользователем символы в числовое значение.

4. Какие строки листинга отвечают за вычисления варианта?

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

Эти команды выполняют операцию деления номера студенческого билета на 20 и увеличивают остаток от деления на единицу.

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

Остаток от деления помещается в регистр edx.

6. Для чего используется инструкция “inc edx”?

Эта команда увеличивает значение в регистре edx на единицу, что необходимо для расчёта номера варианта по заданной формуле.

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

- mov eax, edx – перемещает результат вычислений в регистр eax.
- call iprintLF – иницирует функцию, которая выводит результат на экран с переводом строки.

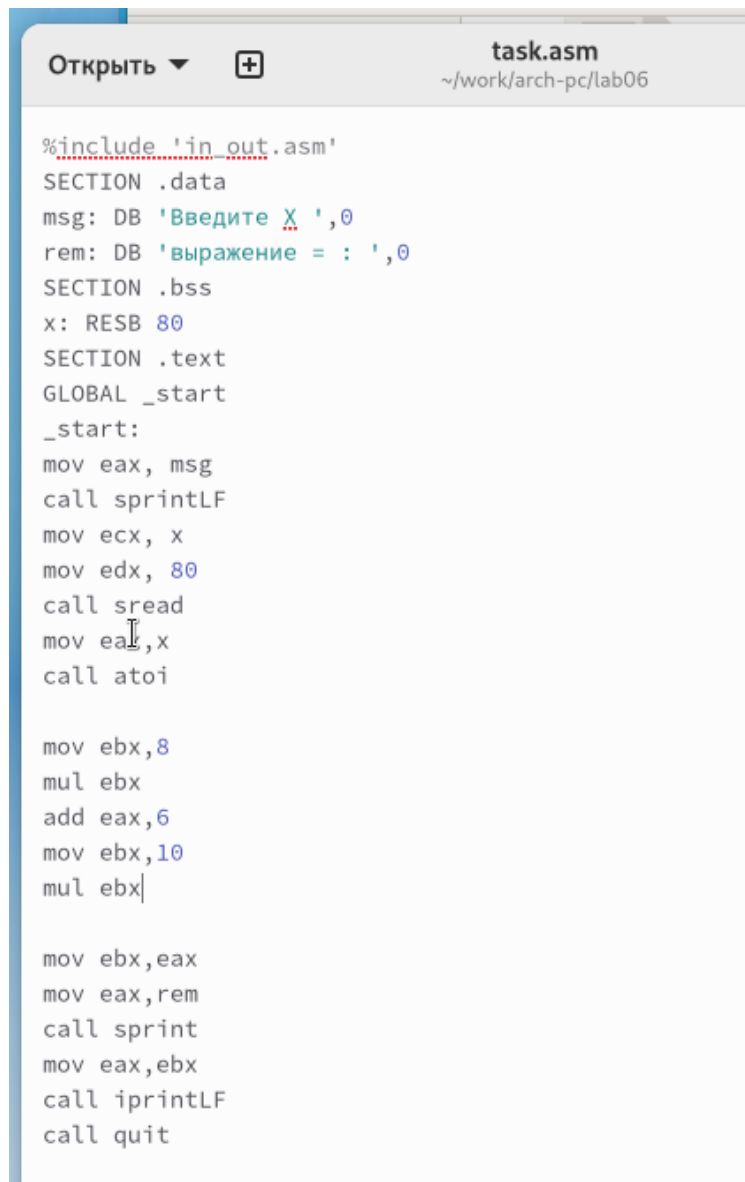
8. Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии 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 13 -

$$(8x + 6) * 10$$

для

$$x_1 = 140, x_4 = 380$$



```
task.asm
~/work/arch-pc/lab06

%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, 8
mul ebx
add eax, 6
mov ebx, 10
mul ebx

mov ebx, eax
mov eax, rem
call sprint
mov eax, ebx
call iprintLF
call quit
```

Рис. 2.16: Программа в файле task.asm

```
[gtumureeva@gtumureeva lab06]$ nasm -f elf task.asm
[gtumureeva@gtumureeva lab06]$ ld -m elf_i386 task.o -o task
[gtumureeva@gtumureeva lab06]$ ./task
Введите X
1
выражение = : 140
[gtumureeva@gtumureeva lab06]$ ./task
Введите X
4
выражение = : 380
[gtumureeva@gtumureeva lab06]$
```

Рис. 2.17: Запуск программы task.asm

3 Выводы

Изучили работу с арифметическими операциями.