

Отчёт по лабораторной работе 6

дисциплина: Архитектура компьютера

Кылыш Гоктюрк

Содержание

1 Цель работы	5
2 Выполнение лабораторной работы	6
3 Выводы	21

Список иллюстраций

2.1 Программа lab6-1.asm	7
2.2 Запуск программы lab6-1.asm	7
2.3 Программа lab6-1.asm	8
2.4 Запуск программы lab6-1.asm	9
2.5 Программа lab6-2.asm	10
2.6 Запуск программы lab6-2.asm	10
2.7 Программа lab6-2.asm	11
2.8 Запуск программы lab6-2.asm	11
2.9 Программа lab6-2.asm	12
2.10 Запуск программы lab6-2.asm	12
2.11 Программа lab6-3.asm	13
2.12 Запуск программы lab6-3.asm	14
2.13 Программа lab6-3.asm	15
2.14 Запуск программы lab6-3.asm	15
2.15 Программа variant.asm	16
2.16 Запуск программы variant.asm	17
2.17 Программа prog.asm	19
2.18 Запуск программы prog.asm	20

Список таблиц

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.

The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "GNU nano 7.2". The code is written in assembly language:

```
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,'6'
    mov ebx,'3'
    add eax,ebx
    mov [buf1],eax
    mov eax,buf1
    call sprintLF
    call quit
```

Рисунок 2.1: Программа lab6-1.asm

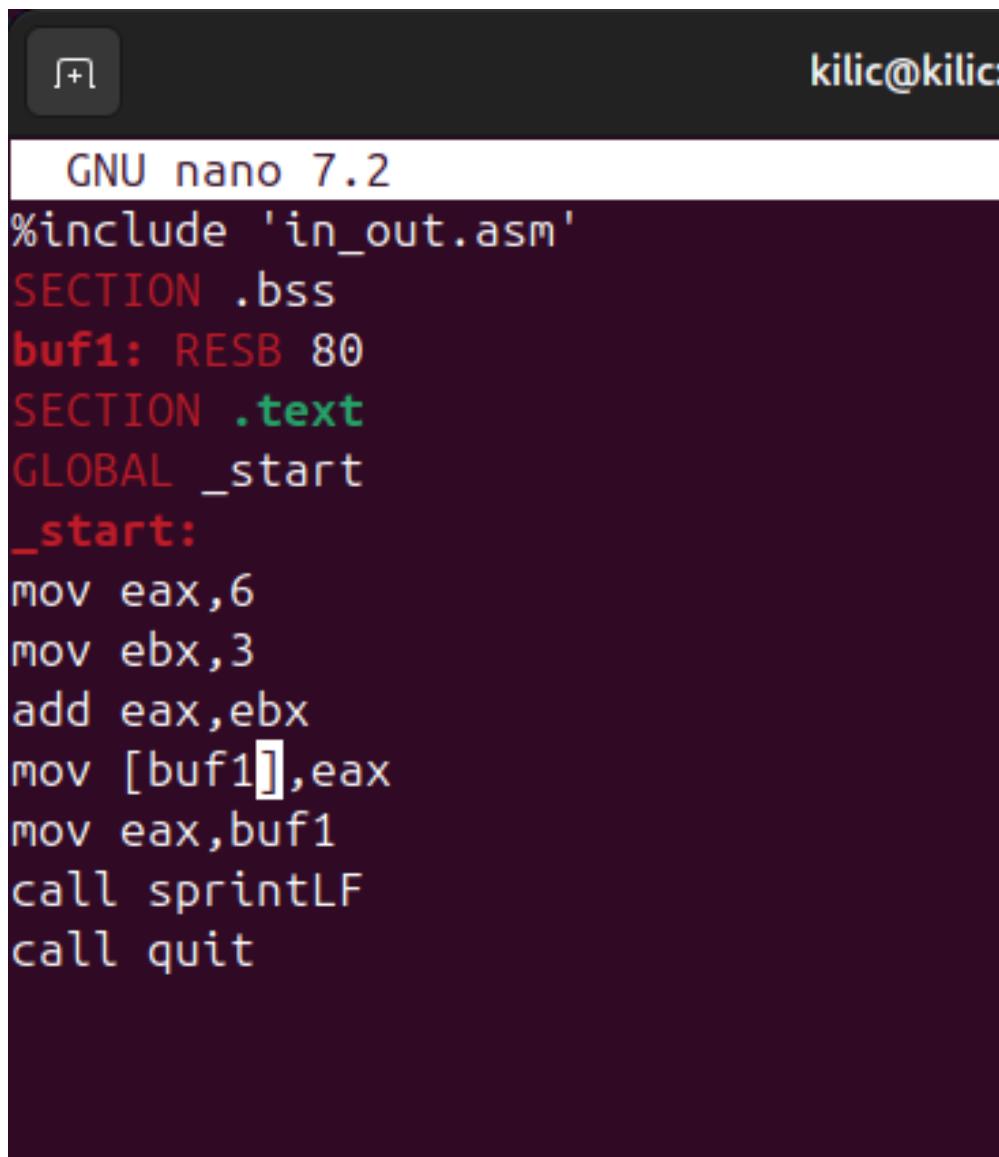
The screenshot shows a terminal window with a dark background and light-colored text. The user runs the assembly program through the terminal:

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

Рисунок 2.2: Запуск программы lab6-1.asm

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

стры числа.



The screenshot shows a terminal window with a dark background and light-colored text. At the top, it says "GNU nano 7.2" and "kilic@kilic: ~". The main area contains assembly code:

```
GNU nano 7.2
%include 'in_out.asm'
SECTION .bss
buf1: RESB 80
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,3
    add eax,ebx
    mov [buf1],eax
    mov eax,buf1
    call sprintLF
    call quit
```

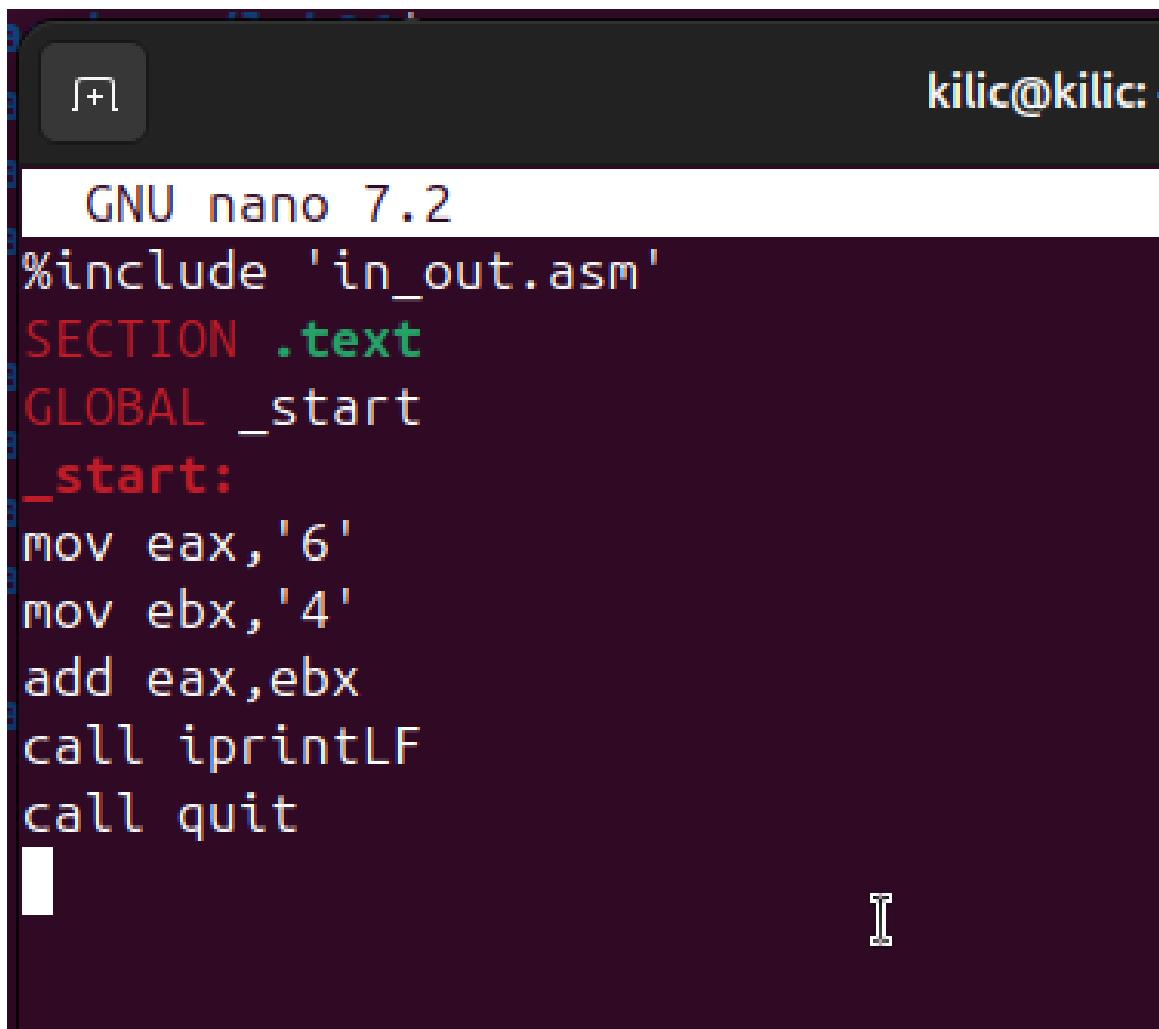
Рисунок 2.3: Программа lab6-1.asm

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

Рисунок 2.4: Запуск программы lab6-1.asm

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

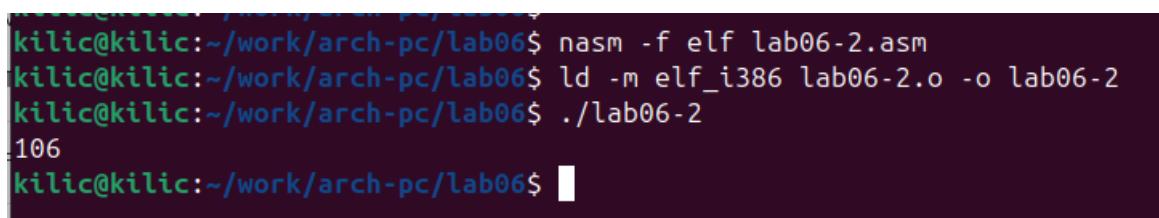
4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал текст программы с использованием этих функций.



The screenshot shows a terminal window with a dark background. In the top right corner, the text "kilic@kilic:" is visible. The main area of the terminal contains the following assembly code:

```
GNU nano 7.2
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
    mov eax,'6'
    mov ebx,'4'
    add eax,ebx
    call iprintLF
    call quit
```

Рисунок 2.5: Программа lab6-2.asm



The screenshot shows a terminal window with a dark background. The command line shows the steps to build and run the assembly program:

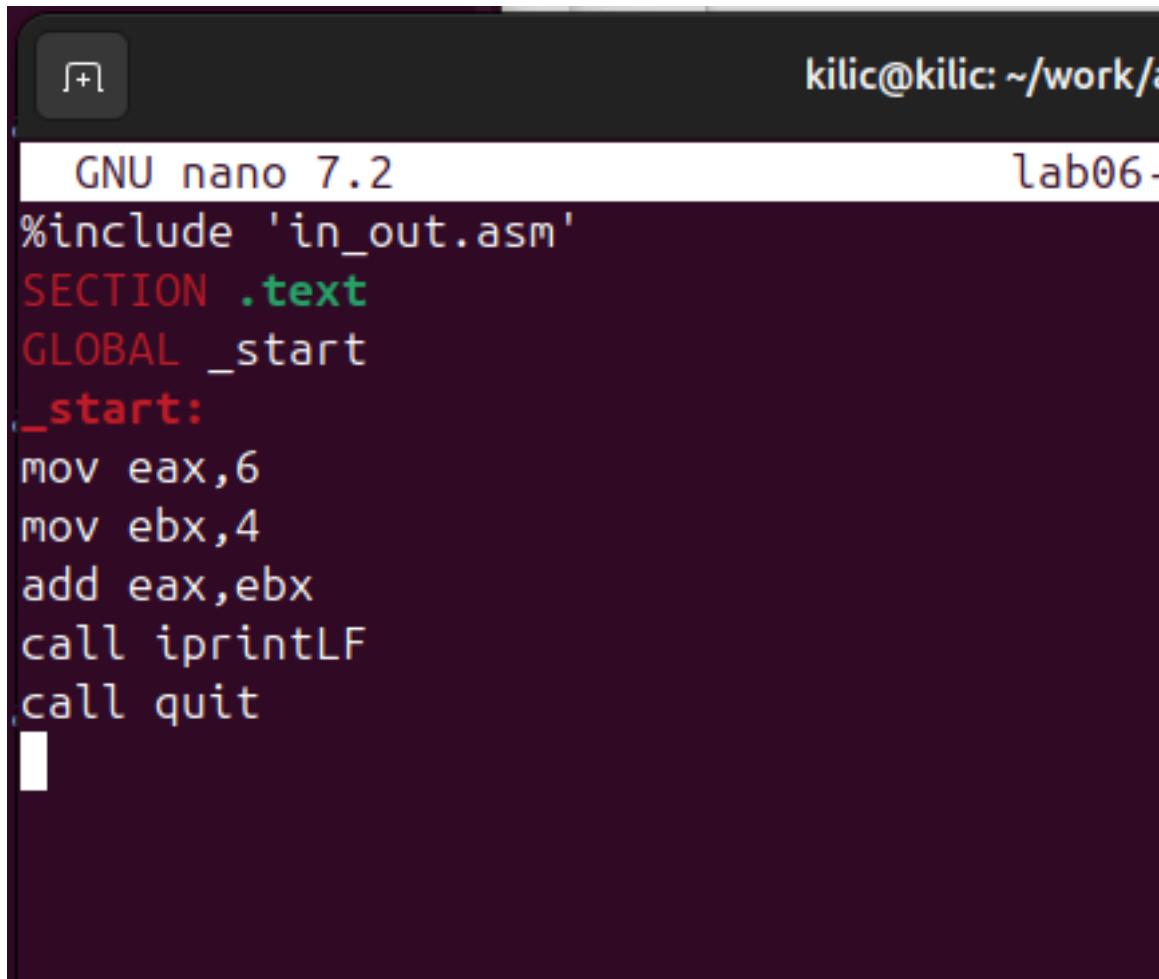
```
kilic@kilic:~/work/arch-pc/lab06$ nasm -f elf lab06-2.asm
kilic@kilic:~/work/arch-pc/lab06$ ld -m elf_i386 lab06-2.o -o lab06-2
kilic@kilic:~/work/arch-pc/lab06$ ./lab06-2
106
kilic@kilic:~/work/arch-pc/lab06$
```

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 отличии от прошлой программы, функция iprintLF позволяет выве-

сти число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим символы на числа.

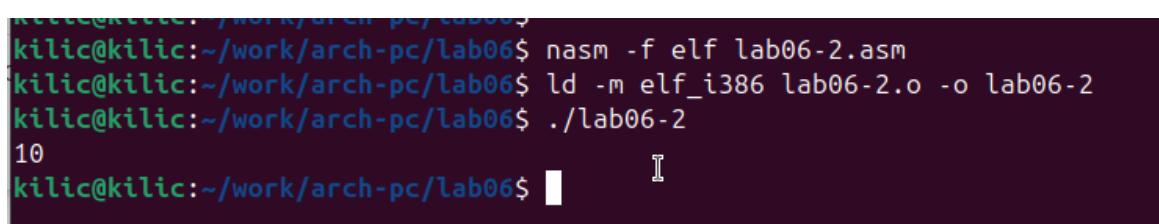


The screenshot shows a terminal window titled 'kilic@kilic: ~/work/lab06-' containing assembly code in the 'nano' editor. The code defines a global symbol '_start' and contains instructions to add the values of registers EAX and EBX, then print the result using the iprintLF function, and finally quit. The assembly code is as follows:

```
GNU nano 7.2
%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

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

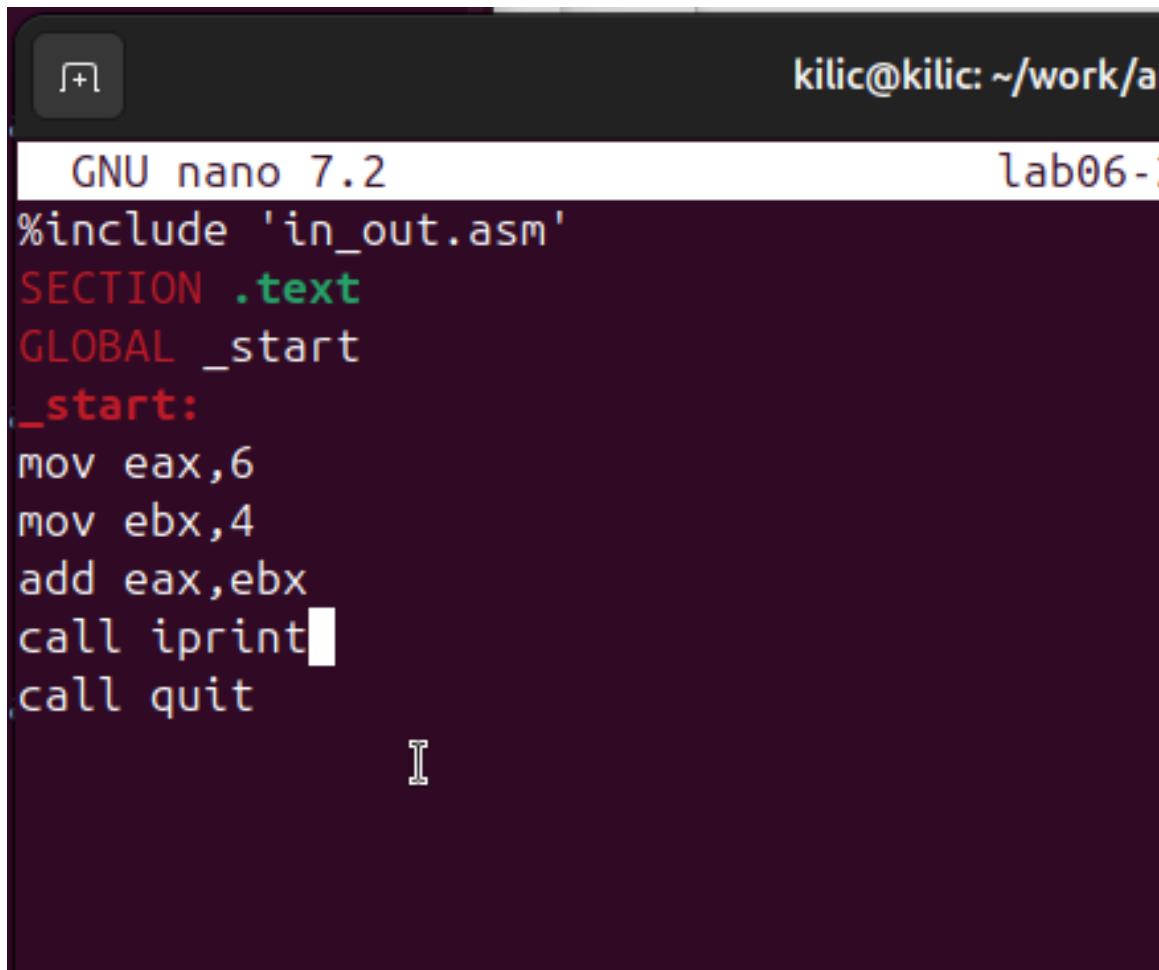


The screenshot shows a terminal window with the following command history:

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

Рисунок 2.8: Запуск программы lab6-2.asm

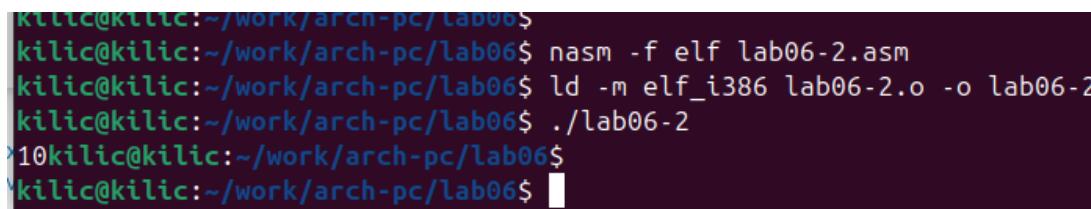
Заменил функцию iprintLF на iprint. Вывод отличается что нет переноса строки.



The screenshot shows a terminal window titled 'kilic@kilic: ~/work/lab06-' containing the assembly code for 'lab6-2.asm'. The code includes directives like %include 'in_out.asm', SECTION .text, and GLOBAL _start, followed by the assembly instructions mov eax,6, mov ebx,4, add eax,ebx, call iprint, and call quit.

```
GNU nano 7.2
kilic@kilic: ~/work/lab06-
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprint
    call quit
```

Рисунок 2.9: Программа lab6-2.asm



The screenshot shows a terminal window with the following command sequence: nasm -f elf lab06-2.asm, ld -m elf_i386 lab06-2.o -o lab06-2, ./lab06-2. The output shows the program's execution, which prints the value 10 to the console.

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

Рисунок 2.10: Запуск программы lab6-2.asm

6. В качестве примера выполнения арифметических операций в NASM

приведем программу вычисления арифметического выражения

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

```
GNU nano 7.2                                     lab06-3.
%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
||
```

Рисунок 2.11: Программа lab6-3.asm

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

Рисунок 2.12: Запуск программы lab6-3.asm

Изменил текст программы для вычисления выражения

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

. Создал исполняемый файл и проверил его работу.

```
GNU nano 7.2                                     lab06-3..  
%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 2.13: Программа lab6-3.asm

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

Рисунок 2.14: Запуск программы lab6-3.asm

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

```
GNU nano 7.2                               variant.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 2.15: Программа variant.asm

```
kilic@kilic:~/work/arch-pc/lab06$ nasm -f elf variant.asm
kilic@kilic:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
kilic@kilic:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032254880
Ваш вариант: 1
kilic@kilic:~/work/arch-pc/lab06$ █
```

Рисунок 2.16: 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 iprintfLF - вызывается подпрограмма для вывода результата на экран.

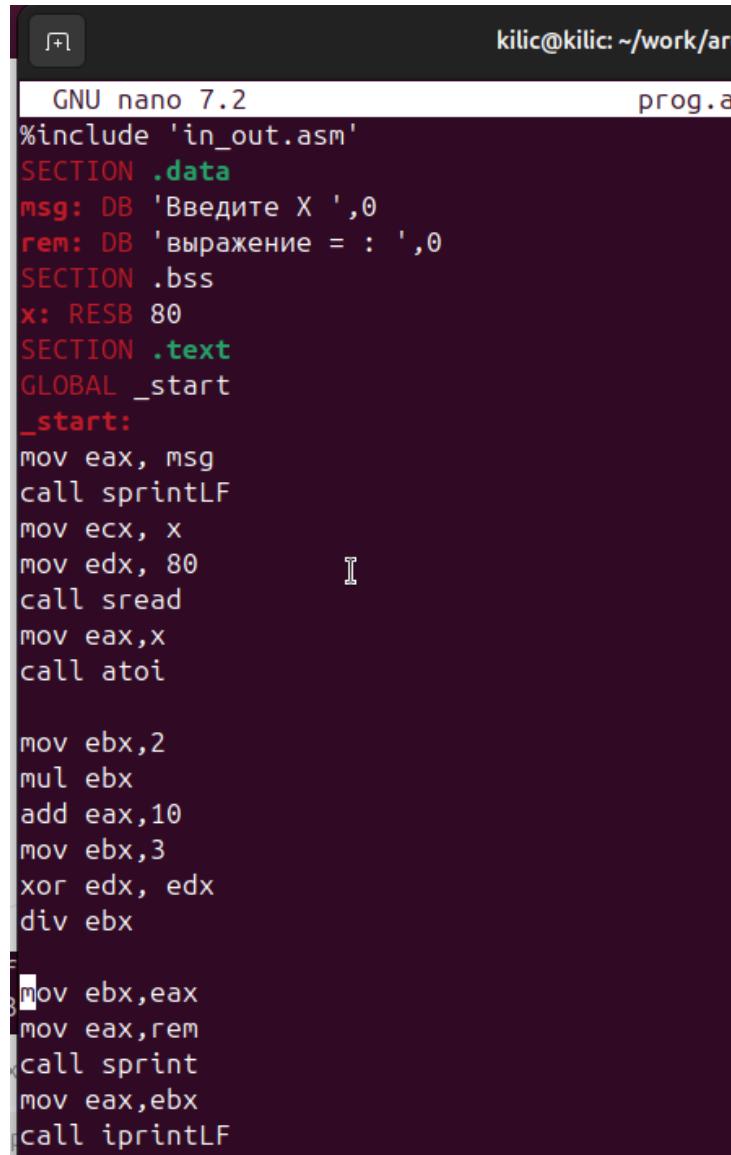
8. Написать программу вычисления выражения $y = f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии 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 1 -

$$(10 + 2x)/3$$

для

$$x = 1, x = 10$$



GNU nano 7.2 kilic@kilic: ~/work/ar
prog.asm

```
%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,2  
    mul ebx  
    add eax,10  
    mov ebx,3  
    xor edx, edx  
    div ebx  
  
    mov ebx,eax  
    mov eax,rem  
    call sprint  
    mov eax,ebx  
    call iprintLF
```

Рисунок 2.17: Программа prog.asm

```
kilic@kilic:~/work/arch-pc/lab06$ nasm -f elf prog.asm
kilic@kilic:~/work/arch-pc/lab06$ ld -m elf_i386 prog.o -o prog
kilic@kilic:~/work/arch-pc/lab06$ ./prog
Введите X
1
выражение = : 4
kilic@kilic:~/work/arch-pc/lab06$ ./prog
Введите X
10
выражение = : 10
kilic@kilic:~/work/arch-pc/lab06$ █
```

Рисунок 2.18: Запуск программы prog.asm

3 Выводы

Изучили работу с арифметическими операциями.