

Отчет по лабораторной работе №7

Смирнов Дмитрий Романович

Содержание

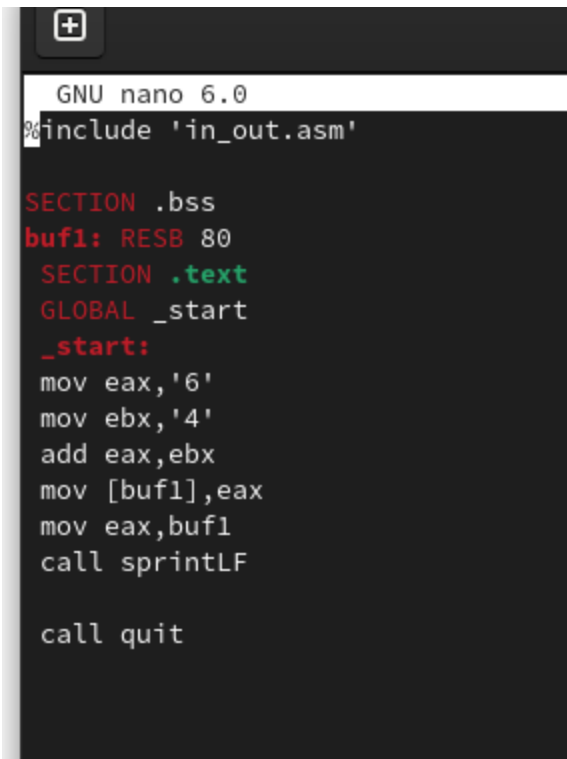
1	Цель работы	1
2	Выполнение лабораторной работы.....	1
3	Вопросы для самопроверки	14
4	Вывод	14

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Выполнение лабораторной работы

Создам каталог для программ лабораторной работы № 7, перейду в него и создам файл lab7-1.asm: Введу в файл lab7-1.asm текст программы из листинга 7.1.



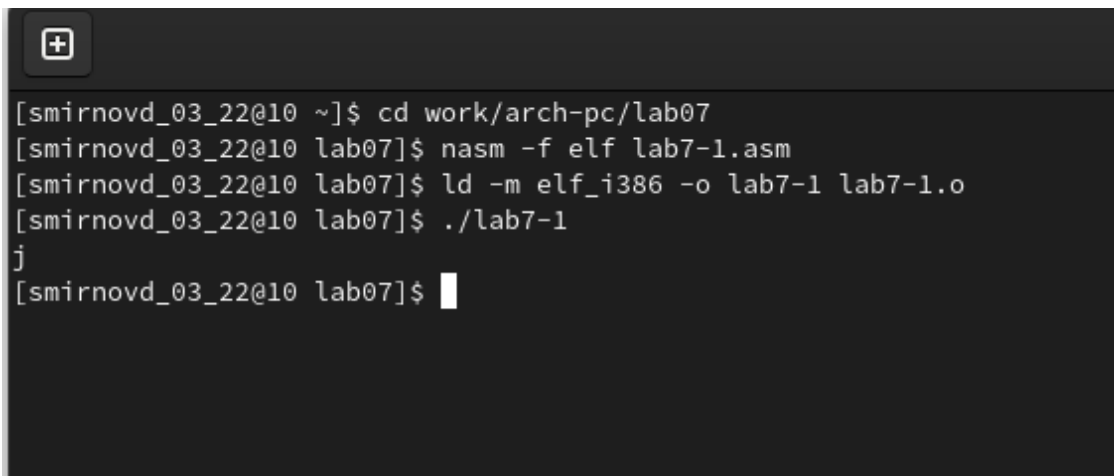
```
GNU nano 6.0
%include 'in_out.asm'

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. 1: Рис1

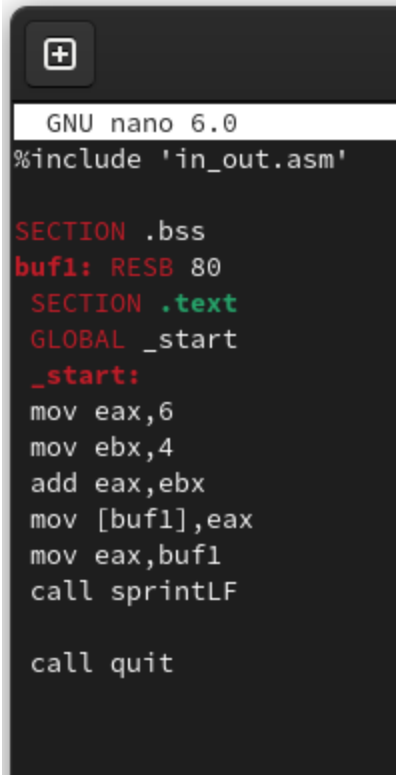
Создам исполняемый файл и запущу его.



```
[smirnovd_03_22@10 ~]$ cd work/arch-pc/lab07
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-1.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[smirnovd_03_22@10 lab07]$ ./lab7-1
j
[smirnovd_03_22@10 lab07]$
```

Рис. 2: Рис2

Далее изменю текст программы и вместо символов, запишу в регистры числа

A screenshot of a terminal window showing the GNU nano 6.0 text editor. The editor is open to a file named 'in_out.asm'. The code includes a section for .bss with a buffer 'buf1' of 80 bytes, followed by a .text section with a global '_start' label. The assembly code moves the value 6 into 'eax', shifts it left by 4 bits into 'ebx', adds 'ebx' to 'eax', moves the result into 'buf1', and then calls 'sprintLF' and 'quit'.

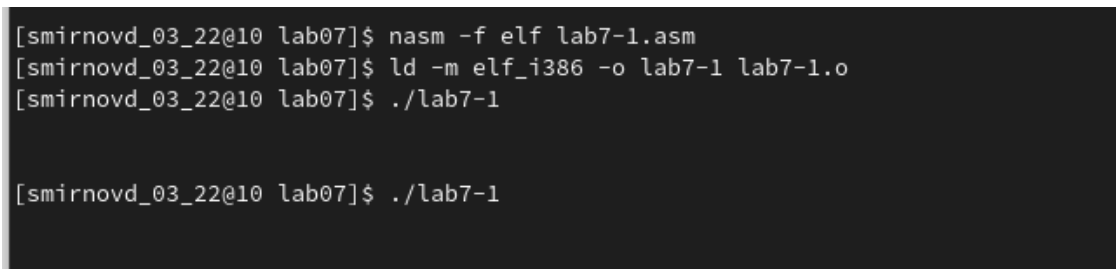
```
GNU nano 6.0
%include 'in_out.asm'

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: Рис3

Запущу файл.

A screenshot of a terminal window showing the compilation and execution of the assembly file. The user runs 'nasm -f elf lab7-1.asm' to compile the assembly file into an object file 'lab7-1.o'. Then, they run 'ld -m elf_i386 -o lab7-1 lab7-1.o' to link the object file into an executable 'lab7-1'. Finally, they run './lab7-1' to execute the program. The output shows the program running successfully.

```
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-1.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[smirnovd_03_22@10 lab07]$ ./lab7-1

[smirnovd_03_22@10 lab07]$ ./lab7-1
```

Рис. 4: Рис4

На выводится символ с кодом 10, обозначающий спец. символ LF, , который не отображается при выводе на экран. Создам файл lab7-2.asm в каталоге /work/arch-рс/lab07 и введу в него текст программы из листинга 7.2. и запущу файл.

```

[smirnovd_03_22@10 lab07]$ touch lab7-2.asm
[smirnovd_03_22@10 lab07]$ mc

[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-1.asm
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-2.asm
[smirnovd_03_22@10 lab07]$ ^C
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[smirnovd_03_22@10 lab07]$ ./lab7-2
106
[smirnovd_03_22@10 lab07]$ █

```

Рис. 5: Рuc5

Заменяю символы на числа аналогично предыдущему примеру и запускаю файл.

```

[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-2.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[smirnovd_03_22@10 lab07]$ ./lab7-2
10
[smirnovd_03_22@10 lab07]$ mc

```

Рис. 6: Рuc6

Заменяю функцию `iprintLF` на `iprint`

```

+
GNU nano 6.0
%include 'in_out.asm'

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

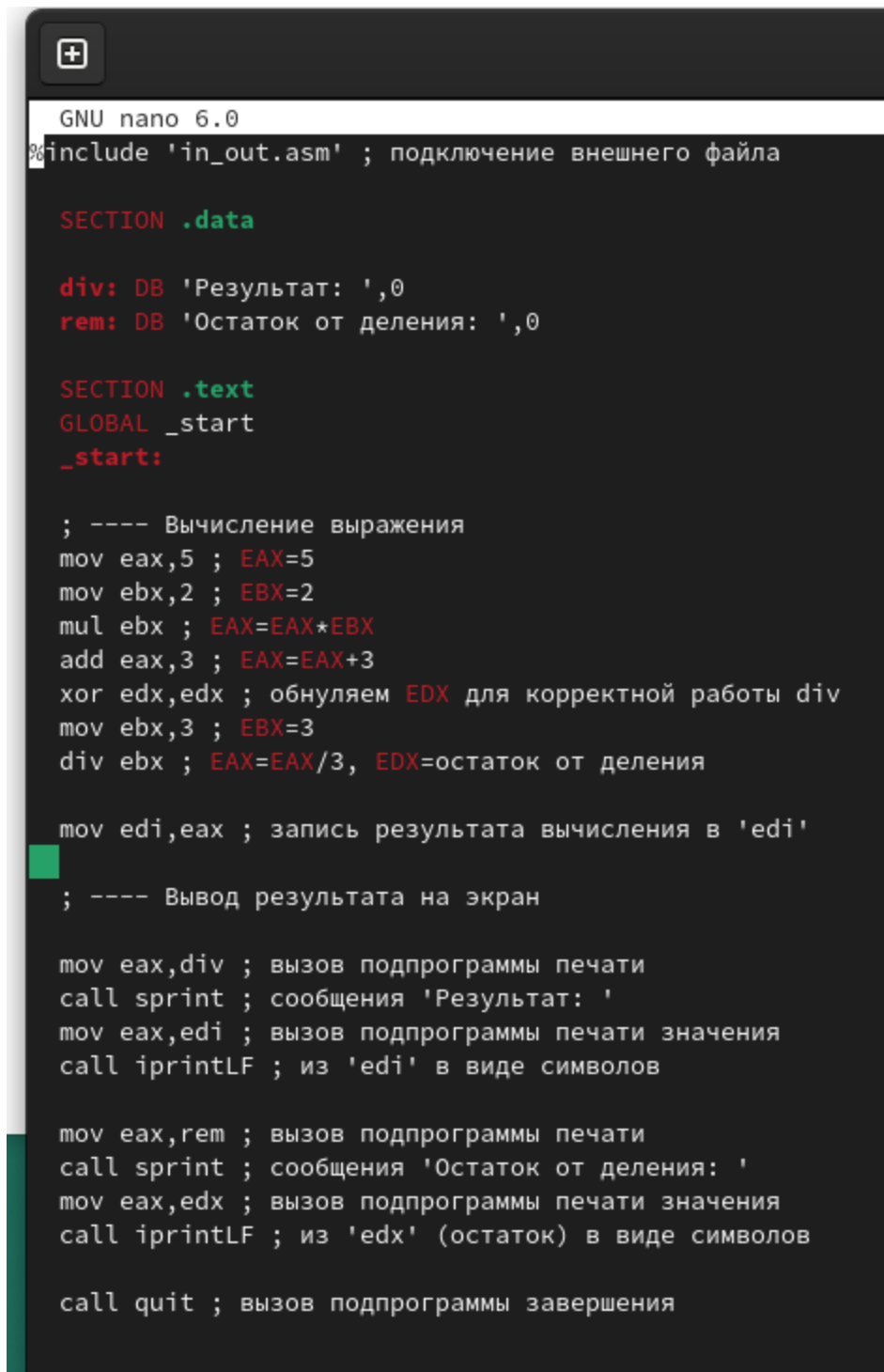
```

Рис. 7: Рuc7

```
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-2.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[smirnovd_03_22@10 lab07]$ ./lab7-2
10[smirnovd_03_22@10 lab07]$
```

Рис. 8: Рис8

При использовании `iprintLF` следующий ввод пользователь переносится на новую строку в отличие от `iprint`. Создам файл `lab7-3.asm` в каталоге `/work/arch-pc/lab07`: Запишу туда текст из листинга 7.3 и запущу файл.



```
GNU nano 6.0
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data

div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:

; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления

mov edi,eax ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран

mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов

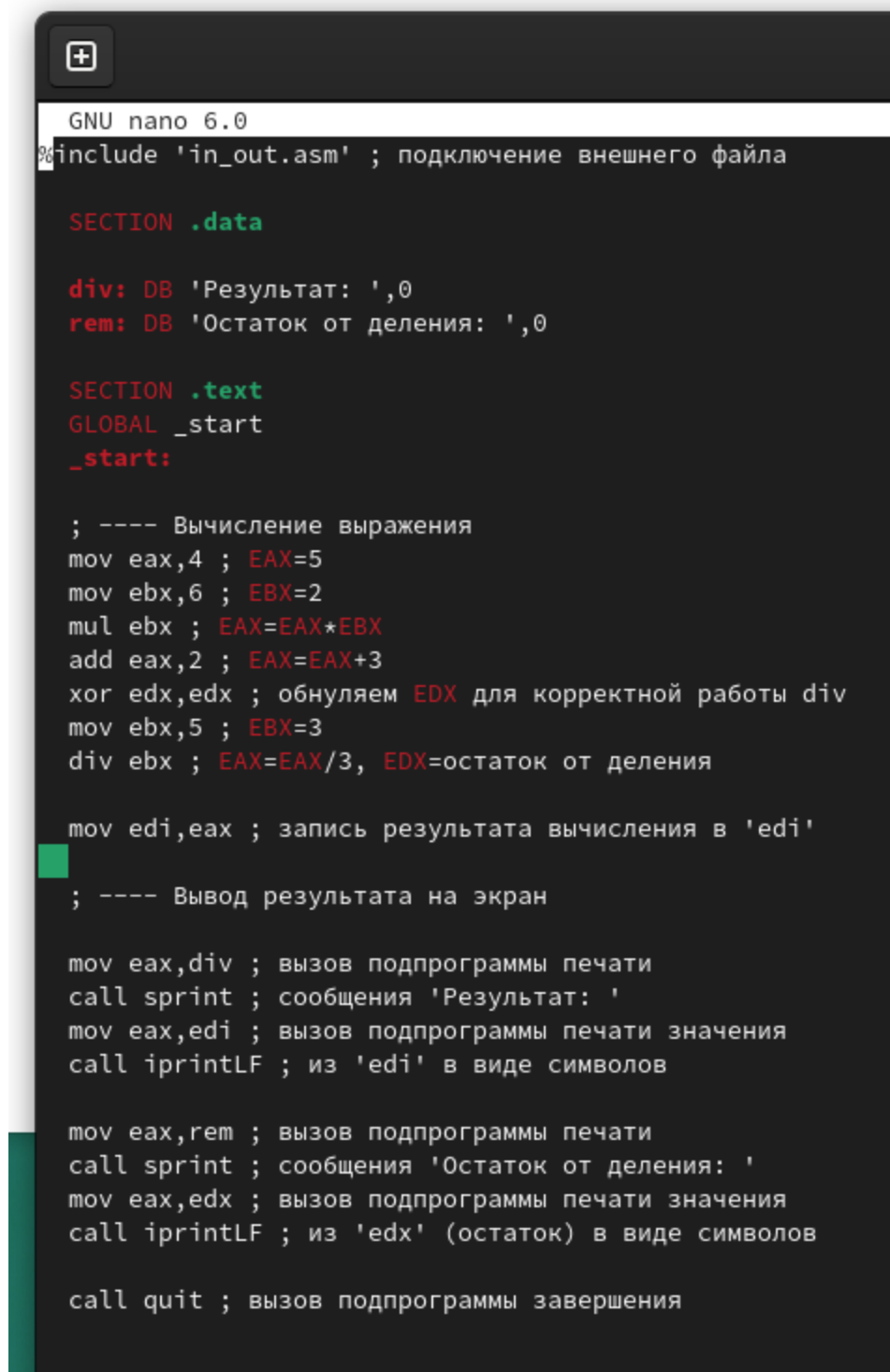
call quit ; вызов подпрограммы завершения
```

Рис. 9: Рис9

```
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-3.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[smirnovd_03_22@10 lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[smirnovd_03_22@10 lab07]$
```

Рис. 10: Рис10

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



```
GNU nano 6.0
%include 'in_out.asm' ; подключение внешнего файла

SECTION .data

div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:

; ---- Вычисление выражения
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления

mov edi,eax ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран

mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов

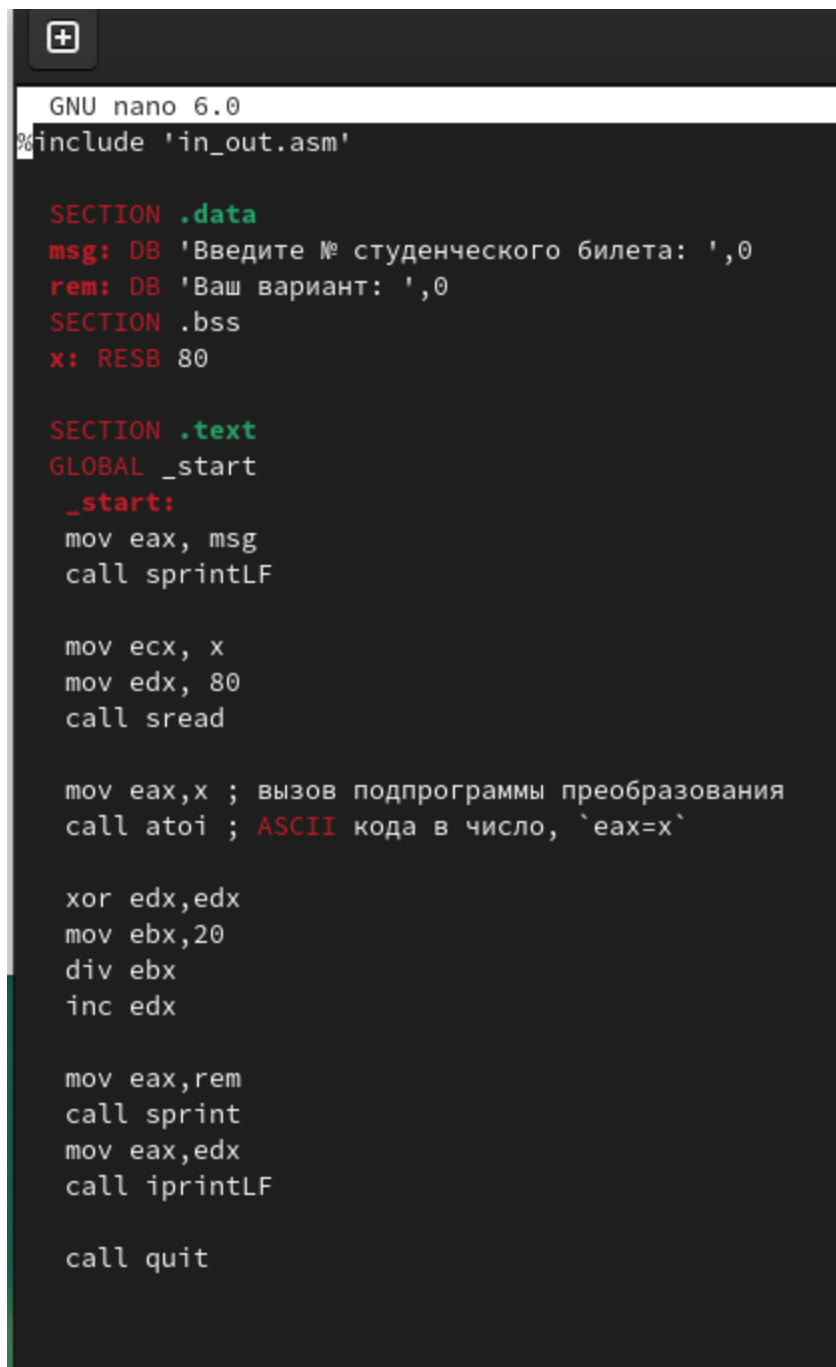
call quit ; вызов подпрограммы завершения
```

Рис. 11: Puc11


```
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-3.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[smirnovd_03_22@10 lab07]$ ./lab7-3
Результат: 5
Остаток от деления: 1
[smirnovd_03_22@10 lab07]$
```

Рис. 12: Рис12

Далее запишу программу, рассчитывающую вариант в зависимости от номера студенческого



```
GNU nano 6.0
%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 ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`

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

mov eax, rem
call sprintf
mov eax, edx
call iprintLF

call quit
```

Рис. 13: Puc13

```
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-4.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[smirnovd_03_22@10 lab07]$ ./lab7-4
Введите № студенческого билета:
1132221813
Ваш вариант: 14
[smirnovd_03_22@10 lab07]$
```

Рис. 14: Puc14

X=1

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:

    mov eax, 1
    xor edx,edx
    mov ebx, 2
    div ebx
    add eax, 8
    mov ecx, 3
    mul ecx

    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. 15: Puc15

```
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-5.asm
lab7-5.asm:15: error: invalid combination of opcode and operands
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-5.asm
lab7-5.asm:15: error: invalid combination of opcode and operands
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-5.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-5 lab7-5.o
[smirnovd_03_22@10 lab07]$ ./lab7-5
Результат: 24
Остаток от деления: 0
[smirnovd_03_22@10 lab07]$
```

Рис. 16: Рис16

X=4

Открыть ▾  ~/

```
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:

    mov eax, 4
    xor edx,edx
    mov ebx, 2
    div ebx
    add eax, 8
    mov ecx, 3
    mul ecx

    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. 17: Puc17

```
[smirnovd_03_22@10 lab07]$ nasm -f elf lab7-6.asm
[smirnovd_03_22@10 lab07]$ ld -m elf_i386 -o lab7-6 lab7-6.o
[smirnovd_03_22@10 lab07]$ ./lab7-6
Результат: 30
Остаток от деления: 0
[smirnovd_03_22@10 lab07]$
```

Рис. 18: Puc18

3 Вопросы для самопроверки

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант: '? `mov eax,rem` `call sprint`
2. Для чего используются следующие инструкции `mov ecx, x` ; `mov edx, 80` ; `call sread` `mov ecx` – `ecx` принимает значение `x` (строки с клавиатуры) `mov edx, 80` – `edx` принимает значение `80` (длина допустимого сообщения) `call sread` – выводит строки с клавиатуры в `ecx`-адрес длины `edx`
3. Для чего используется инструкция "call atoi"? Переводит ASCII кода в число
4. Какие строки листинга 7.4 отвечают за вычисления варианта? `xor edx,edx` ; `mov ebx,20` ; `div ebx` ; `inc edx`
5. Для чего используется инструкция "inc edx"? Увеличивает операнда на единицу
6. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? `mov eax,rem` ; `call sprint` ; `mov eax,edx` ; `call iprintLF`

4 Вывод

Я освоил арифметические инструкции языка ассемблер NASM