

Отчёт по лабораторной работе 6

Архитектура компьютера

Диденко Дмитрий Владимирович НПИбд-03-23

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	Задание для самостоятельной работы	17
3	Выводы	20

Список иллюстраций

2.1	Код программы lab6-1.asm	7
2.2	Компиляция и запуск программы lab6-1.asm	7
2.3	Код программы lab6-1.asm	8
2.4	Компиляция и запуск программы lab6-1.asm	8
2.5	Код программы lab6-2.asm	9
2.6	Компиляция и запуск программы lab6-2.asm	9
2.7	Код программы lab6-2.asm	10
2.8	Компиляция и запуск программы lab6-2.asm	10
2.9	Код программы lab6-2.asm	11
2.10	Компиляция и запуск программы lab6-2.asm	11
2.11	Код программы lab6-3.asm	12
2.12	Компиляция и запуск программы lab6-3.asm	12
2.13	Код программы lab6-3.asm	13
2.14	Компиляция и запуск программы lab6-3.asm	14
2.15	Код программы variant.asm	15
2.16	Компиляция и запуск программы variant.asm	15
2.17	Код программы program.asm	18
2.18	Компиляция и запуск программы program.asm	19

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

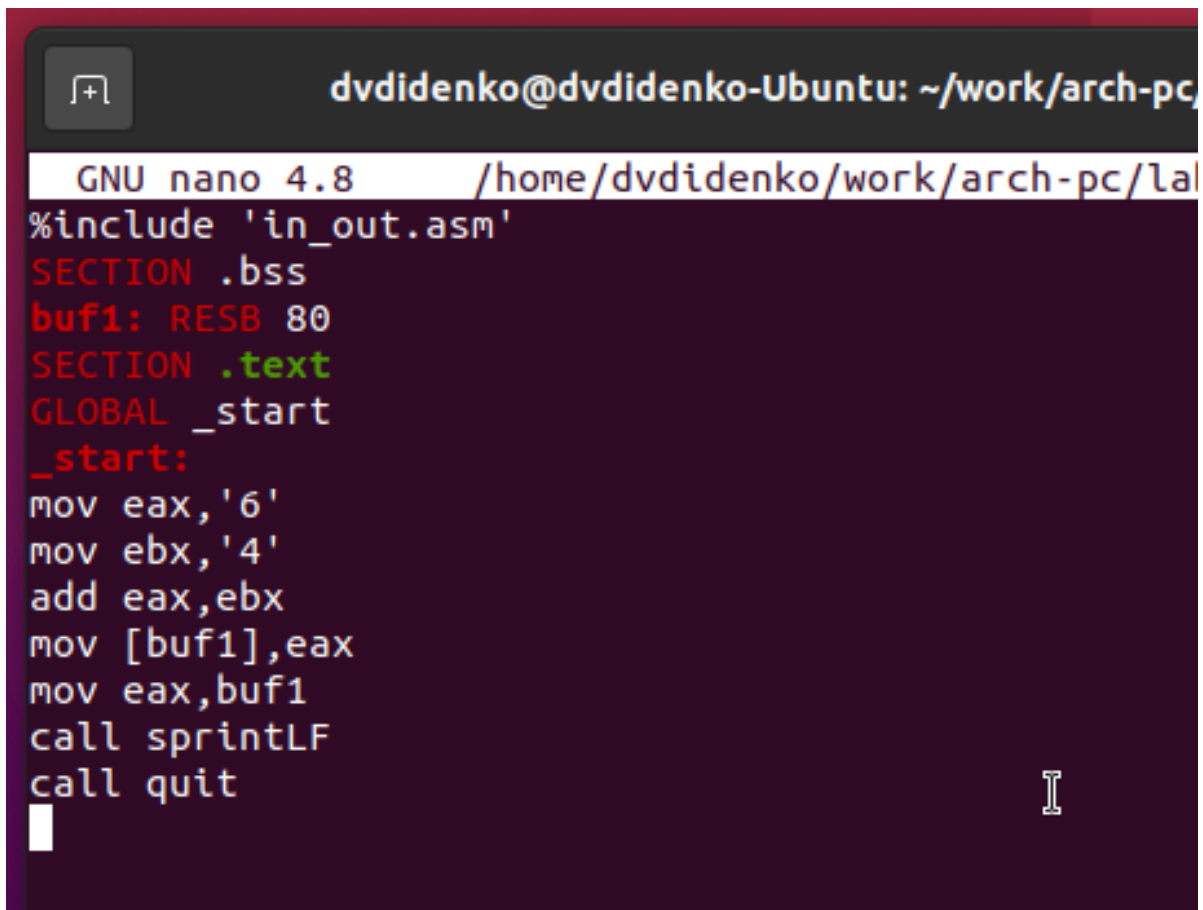
2.1 Символьные и численные данные в NASM

Создал каталог для программ лабораторной работы № 6, перешел в него и создал файл lab6-1.asm.

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

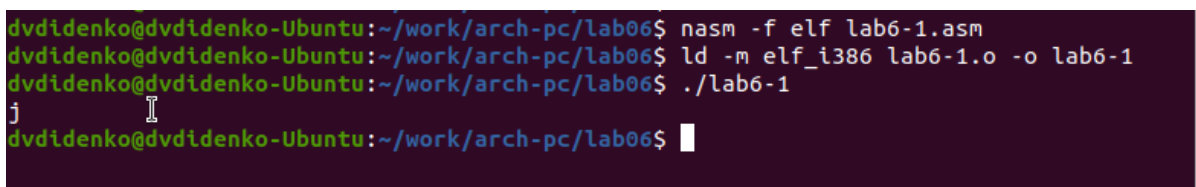
В данной программе (рис. [2.1]) в регистр еах записывается символ 6 (mov еах,'6'), в регистр ебх символ 4 (mov ебх,'4'). Далее к значению в регистре еах прибавляем значение регистра ебх (add еах,ебх, результат сложения запишется в регистр еах). Далее выводим результат. (рис. [2.2])

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



```
dvdidenko@dvdidenko-Ubuntu: ~/work/arch-pc/
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab6-1.asm
#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: Код программы lab6-1.asm



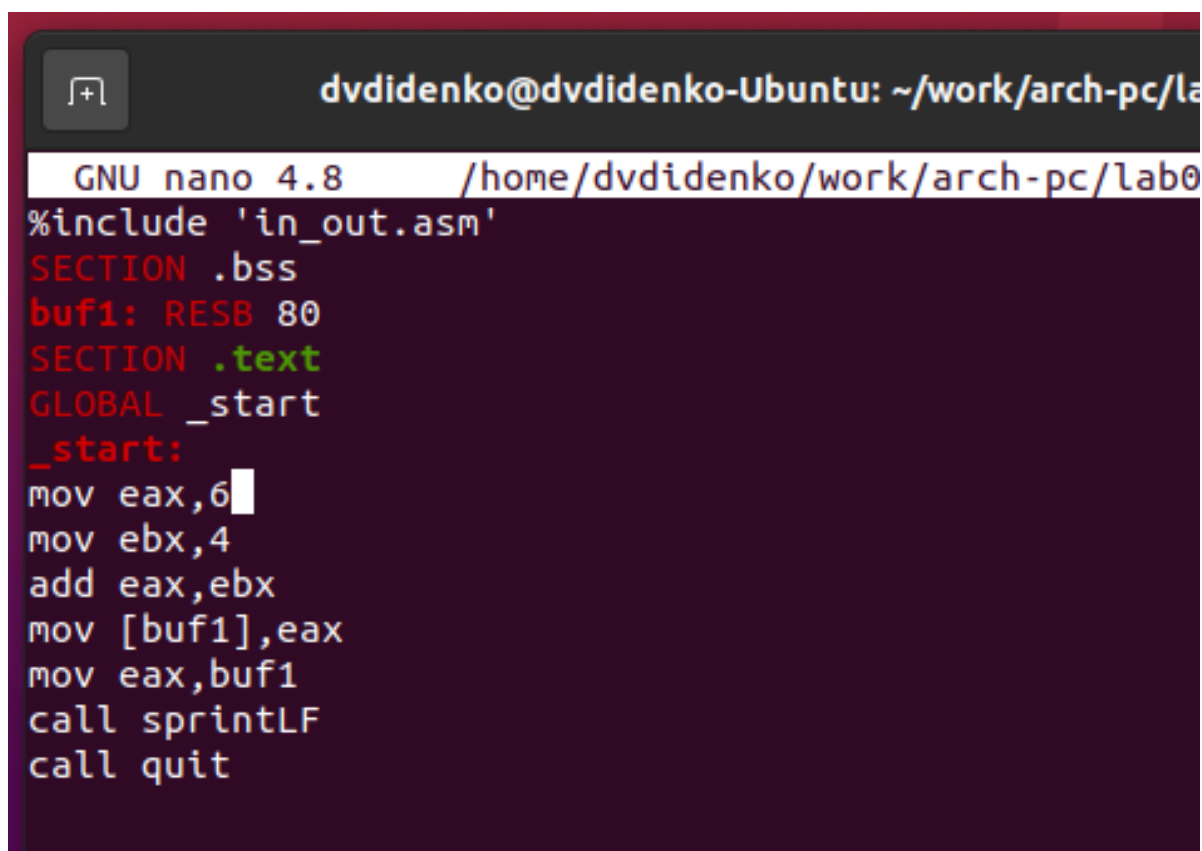
```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-1.o -o lab6-1
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ./lab6-1
j
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.2: Компиляция и 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 сим-

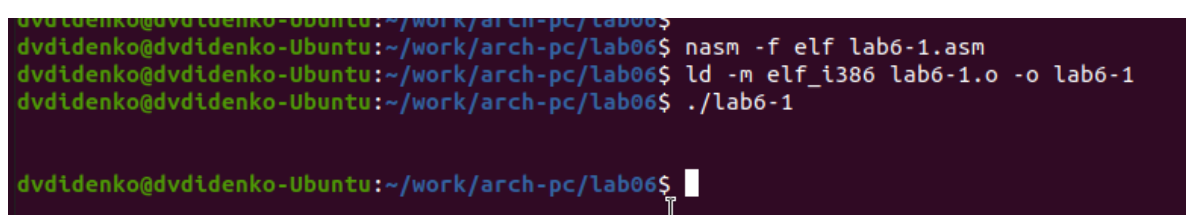
вола j.

Далее изменяю текст программы и вместо символов, запишем в регистры числа. (рис. [2.3])



```
dvdidenko@dvdidenko-Ubuntu: ~/work/arch-pc/lab06
GNU nano 4.8 /home/dvdidenko/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 sprintf
call quit
```

Рис. 2.3: Код программы lab6-1.asm



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

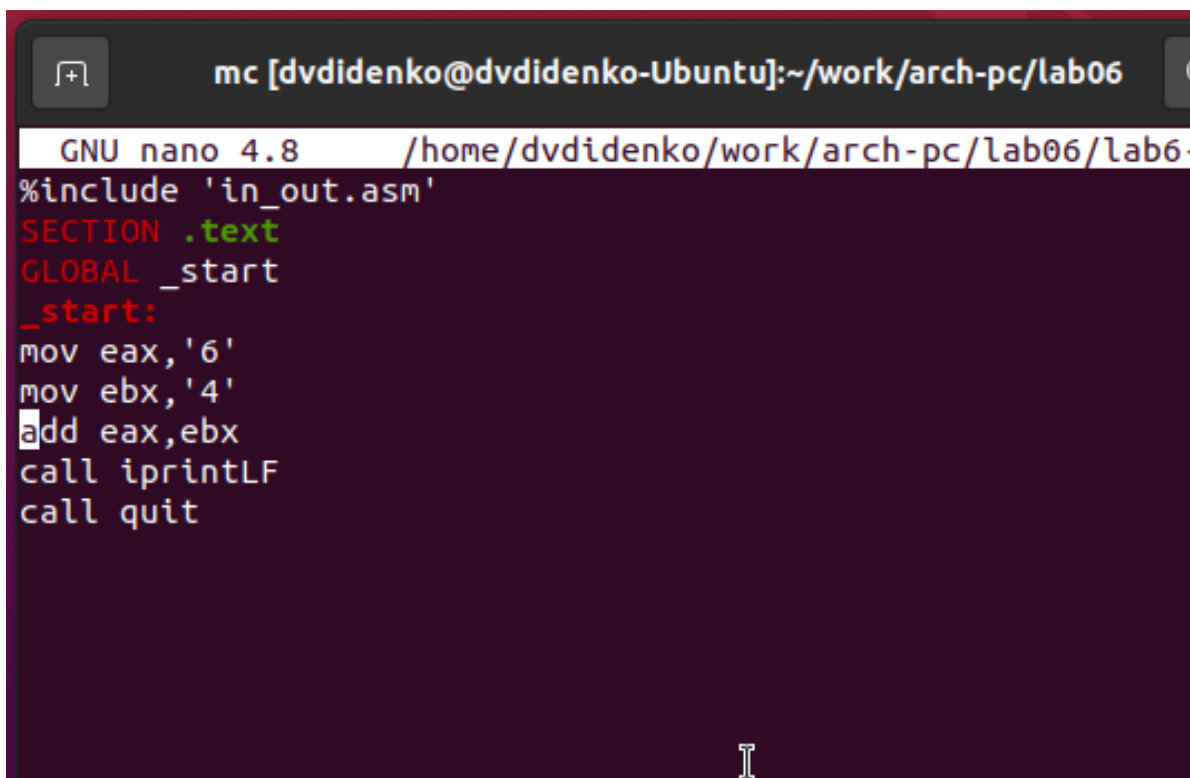
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.4: Компиляция и запуск программы lab6-1.asm

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10 (рис. [2.4]). Это символ конца

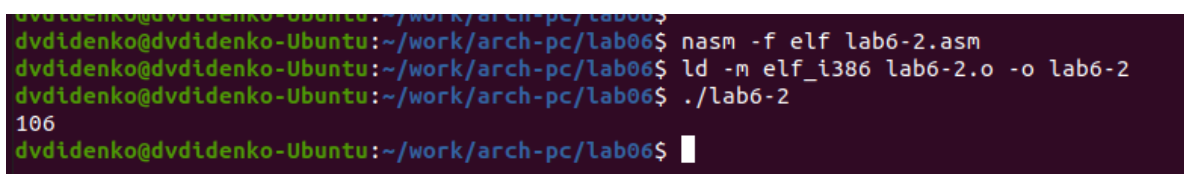
строки (возврат каретки). В консоле он не отображается, но добавляет пустую строку.

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



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab06
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab06/lab6-
%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: Код программы `lab6-2.asm`



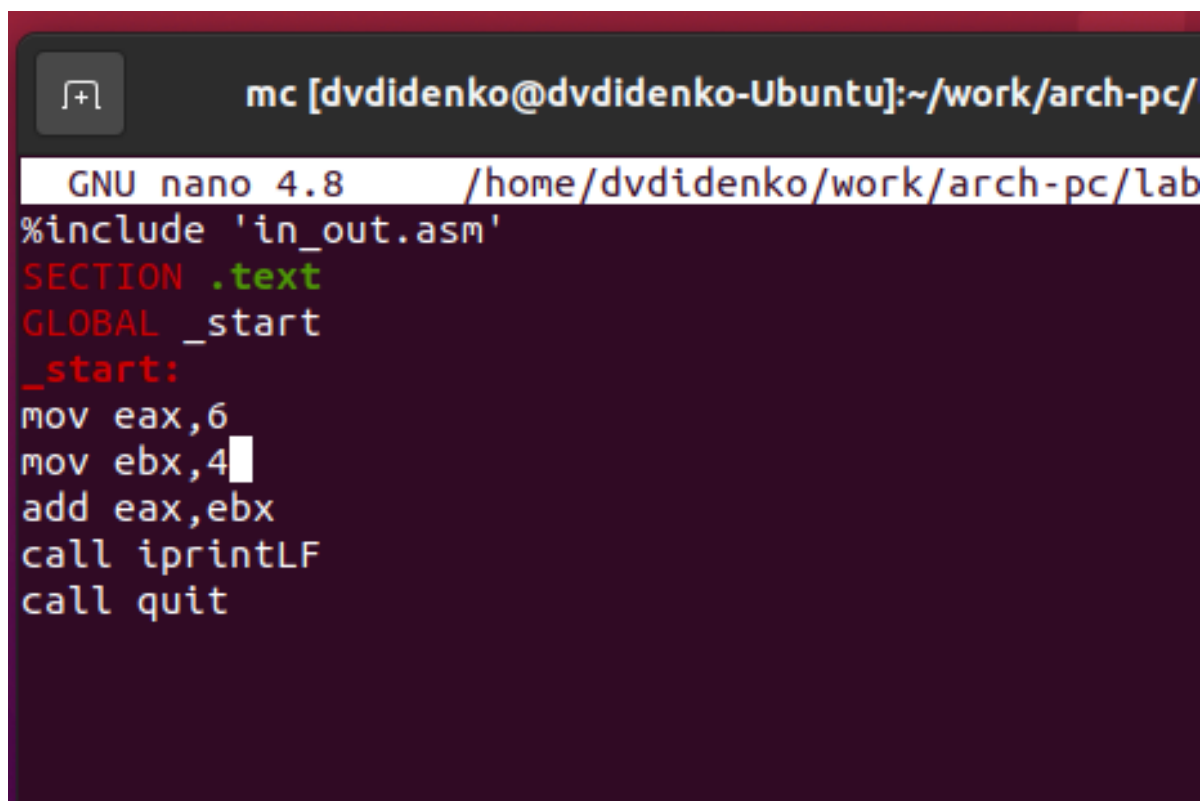
```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-2.o -o lab6-2
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ./lab6-2
106
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$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

Рис. 2.6: Компиляция и запуск программы `lab6-2.asm`

В результате работы программы мы получим число 106.(рис. [2.6]) 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.

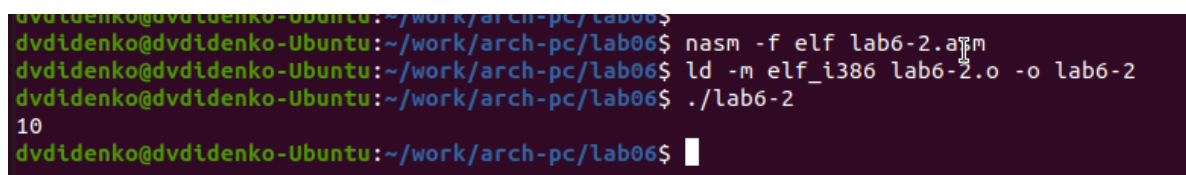
Аналогично предыдущему примеру изменим символы на числа. (рис. [2.7])



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab
%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: Код программы `lab6-2.asm`

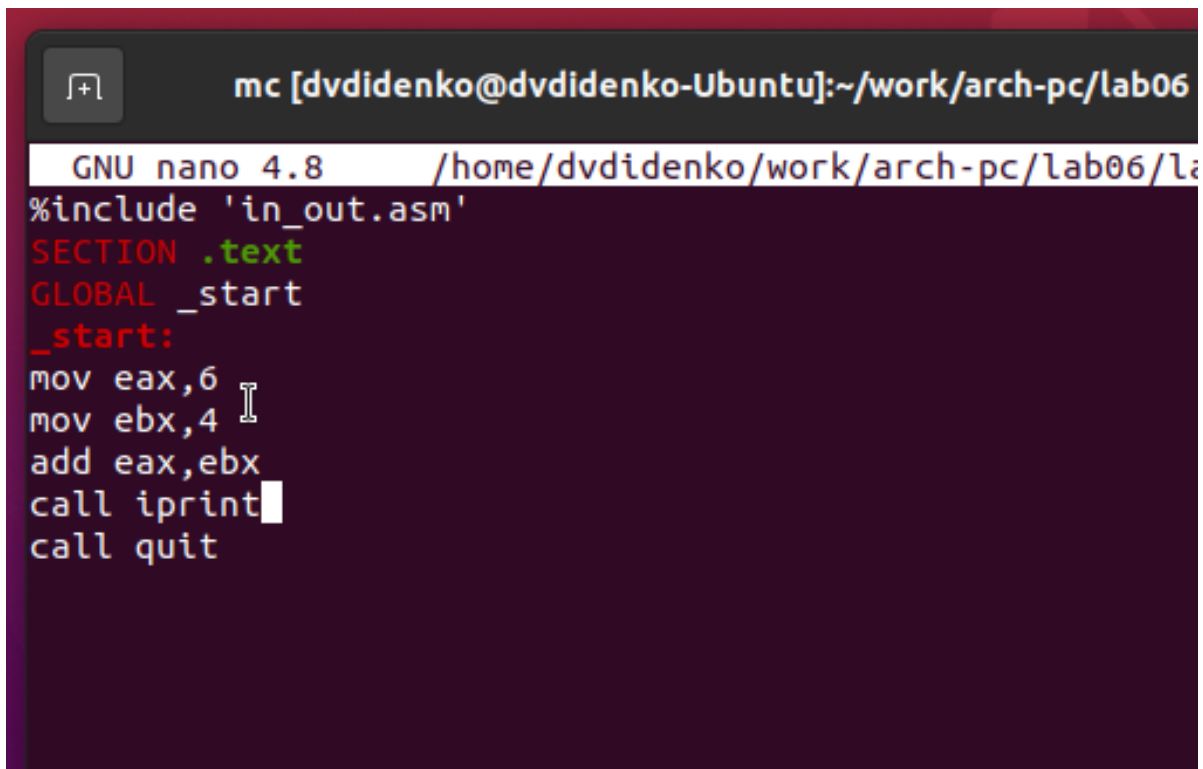
Функция `iprintLF` позволяет вывести число и операндами были числа (а не коды символов). Поэтому получаем число 10. (рис. [2.8])



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

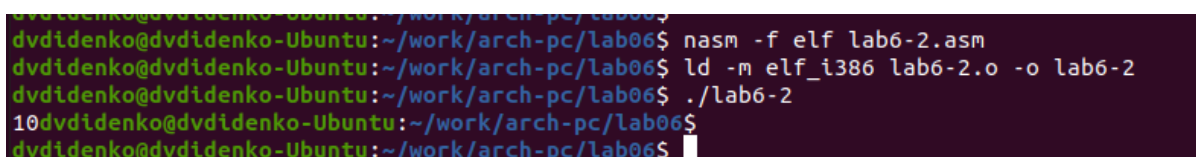
Рис. 2.8: Компиляция и запуск программы `lab6-2.asm`

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



```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab06
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 2.9: Код программы lab6-2.asm



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

Рис. 2.10: Компиляция и запуск программы lab6-2.asm

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. [2.11]) (рис. [2.12])

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab6-3.asm
%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
```

Рис. 2.11: Код программы lab6-3.asm

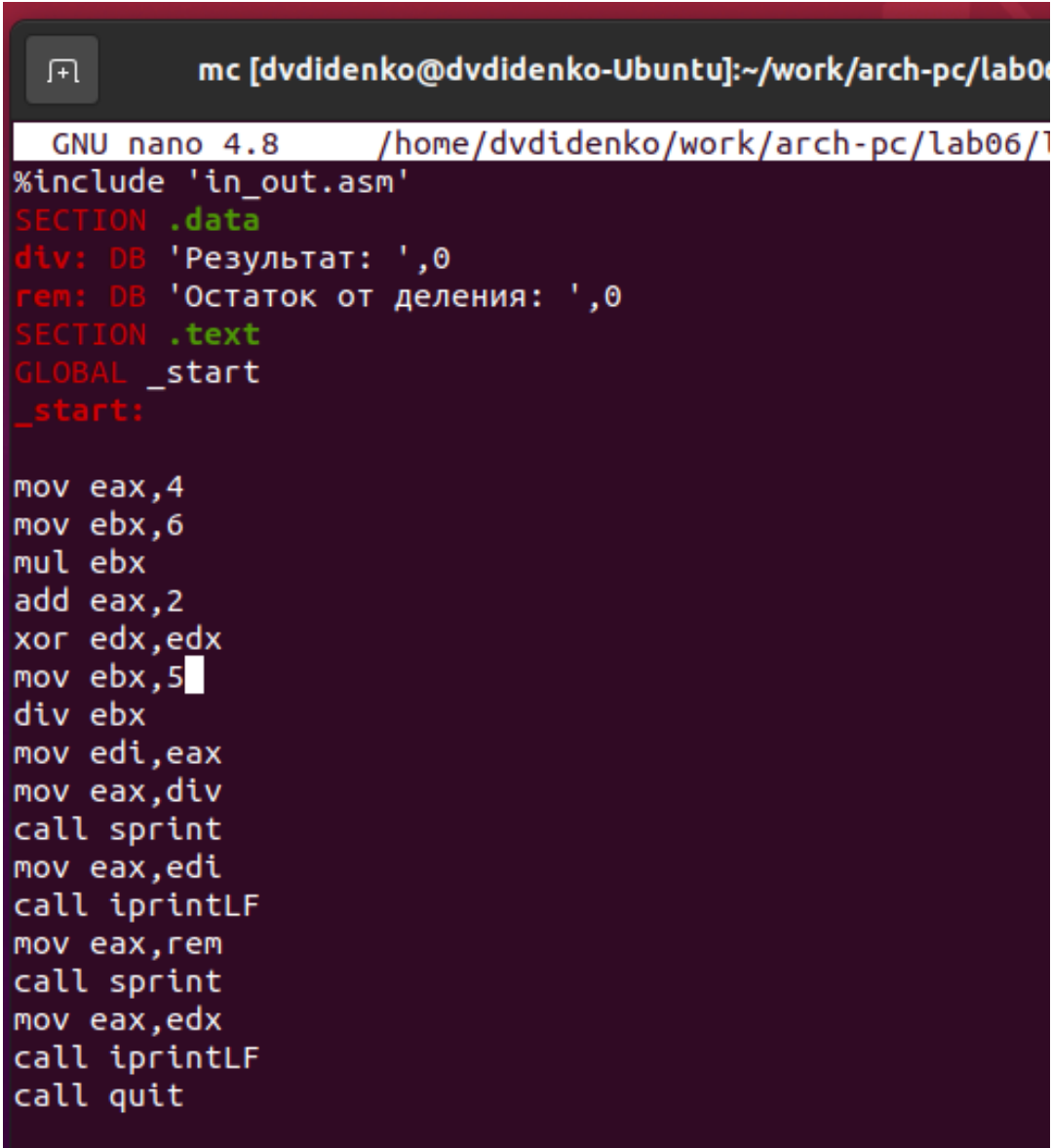
```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 lab6-3.o -o lab6-3
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1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

Рис. 2.12: Компиляция и 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. [2.13]) (рис. [2.14])

A screenshot of a terminal window with a dark background. The title bar shows the user 'mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab06/'. The terminal content shows the GNU nano 4.8 editor editing a file at /home/dvdidenko/work/arch-pc/lab06/1. The code is assembly language, including a data section with strings for 'Результат:' and 'Остаток от деления:', and a text section with instructions to calculate (4*6+2)/5 and print the results using 'sprint' and 'iprintlnf' functions.

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab06/
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab06/1
%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 iprintlnf
mov eax,rem
call sprint
mov eax,edx
call iprintlnf
call quit
```

Рис. 2.13: Код программы lab6-3.asm

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

Рис. 2.14: Компиляция и запуск программы lab6-3.asm

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

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`. (рис. [2.15]) (рис. [2.16])

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab06
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab06/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 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 iprintf
call quit
```

Рис. 2.15: Код программы variant.asm

```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf variant.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032230532
Ваш вариант: 13
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.16: Компиляция и запуск программы variant.asm

ответы на вопросы

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

Переменная с фразой "Ваш вариант:" перекладывается в регистр `eax` с помощью строки `mov eax, ptr`.

Для вызова подпрограммы вывода строки используется строка `call sprint`.

2. Для чего используются следующие инструкции?

- `mov ecx, x` - перекладывает регистр `ecx` в переменную
- `mov edx, 80` - устанавливает значение 80 в регистр `edx`.
- `call sread` - вызывает подпрограмму для считывания значения 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?

- `xor edx, edx` - обнуляет регистр `edx`.
- `mov ebx, 20` - устанавливает значение 20 в регистр `ebx`.
- `div ebx` - выполняет 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 используется для увеличения значения 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?

- Результат вычислений перекладывается в регистр eax с помощью строки `mov eax, edx`.
- Для вызова подпрограммы вывода используется строка `call iprintLF`.

2.3 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. (рис. [2.17]) (рис. [2.18]) Создайте исполняемый файл и проверьте его работу для значений x_1 и x_2 из 6.3.

Вариант 13 - $(8x + 6) \cdot 10$ для $x=1$, $x=4$

```
mc [dvdidenko@dvdidenko-Ubuntu]:~/work/arch-pc/lab06
GNU nano 4.8 /home/dvdidenko/work/arch-pc/lab06/program.asm
rem: DB 'выражение = : ',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

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

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

Рис. 2.17: Код программы program.asm

при $x=1$ $f(x) = 140$

при $x=4$ $f(x) = 380$

```
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ nasm -f elf program.asm
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ld -m elf_i386 program.o -o program
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ./program
Введите X
1
выражение = : 140
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$ ./program
Введите X
4
выражение = : 380
dvdidenko@dvdidenko-Ubuntu:~/work/arch-pc/lab06$
```

Рис. 2.18: Компиляция и запуск программы program.asm

Программа считает верно.

3 Выводы

Изучили работу с арифметическими операциями.