Отчёт по лабораторной работе 6

Архитектура компьютера

Довлетов Довлет

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение лабораторной работы

Создан каталог для программ лабораторной работы № 6, в который был добавлен файл `lab6-1.asm`.

Рассмотрим примеры программ, выводящих символьные и числовые значения. Эти программы будут выводить данные, записанные в регистр `eax`.

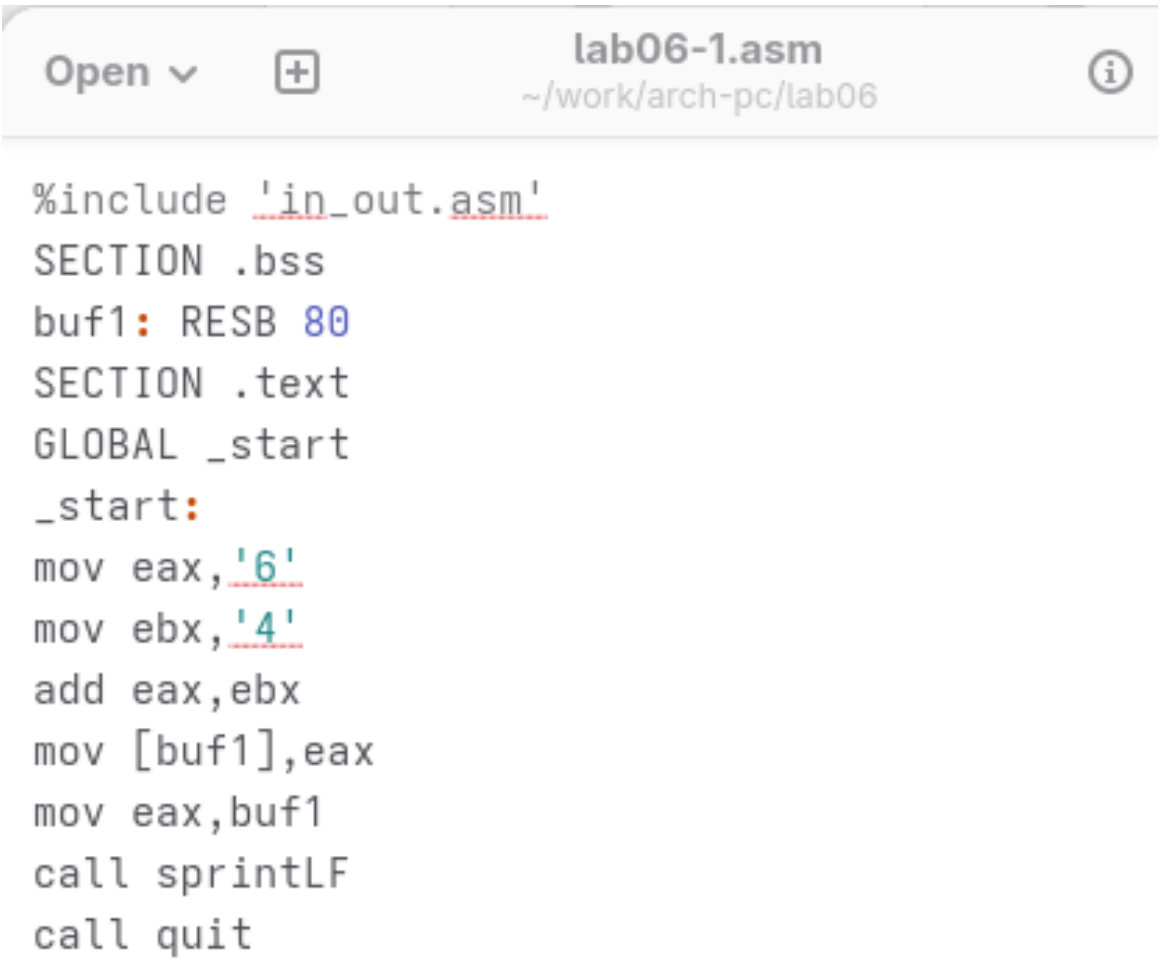

Рисунок 2.1: Программа в файле lab6-1.asm

В данной программе (рис. 2.1) в регистр eax записывается символ „6“ (команда `mov eax, '6'`), а в регистр ebx — символ „4“ (команда `mov ebx, '4'`). Затем происходит сложение значений регистров eax и ebx (команда `add eax, ebx`), результат операции записывается в регистр eax. После этого выводится результат. Для использования функции `sprintLF` необходимо, чтобы в регистре eax находился адрес, поэтому используется дополнительная переменная. Значение регистра eax записывается в переменную `buf1` (команда `mov`

[buf1], eax), а затем в регистр eax записывается адрес переменной buf1 (команда mov eax, buf1), после чего вызывается функция sprintLF.

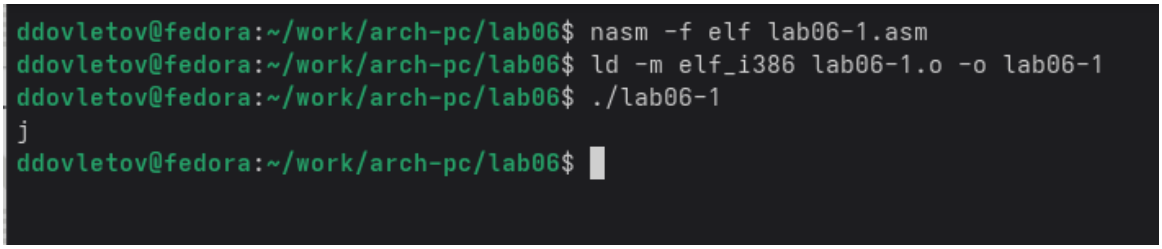

Рисунок 2.2: Запуск программы lab6-1.asm

В случае, когда ожидаем получить число 10 при выводе содержимого регистра eax, фактический результат будет символ „j“. Это объясняется тем, что код символа „6“ в двоичном представлении равен 00110110 (54 в десятичном), а код символа „4“ — 00110100 (52 в десятичном). При выполнении команды add eax, ebx результатом будет сумма этих кодов — 01101010 (106 в десятичном), что соответствует символу „j“ (рис. 2.2).

Далее изменяем программу, заменяя символы на числа (рис. 2.3).

Рисунок 2.3: Программа в файле lab6-1.asm

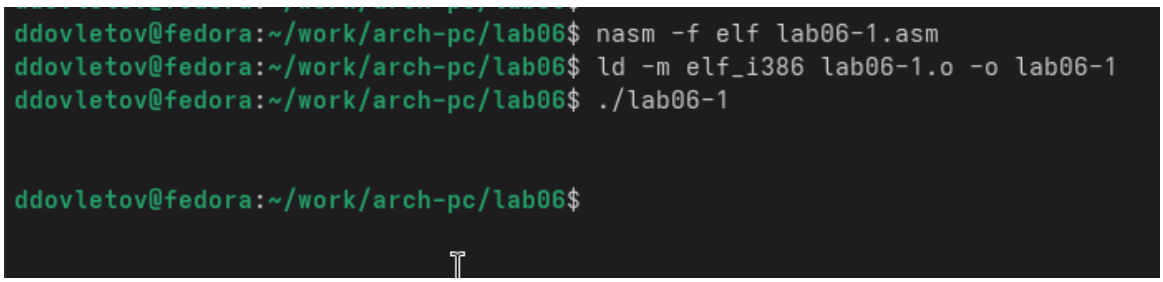

Рисунок 2.4: Запуск программы lab6-1.asm

Как и в предыдущем примере, при выполнении программы мы не получаем число 10. Вместо этого выводится символ с кодом 10, который представляет собой символ конца строки (возврат каретки). Этот символ не отображается в

консоли, но добавляет пустую строку.

Как упоминалось ранее, для работы с числами в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразовал программу с использованием этих функций (рис. 2.5).

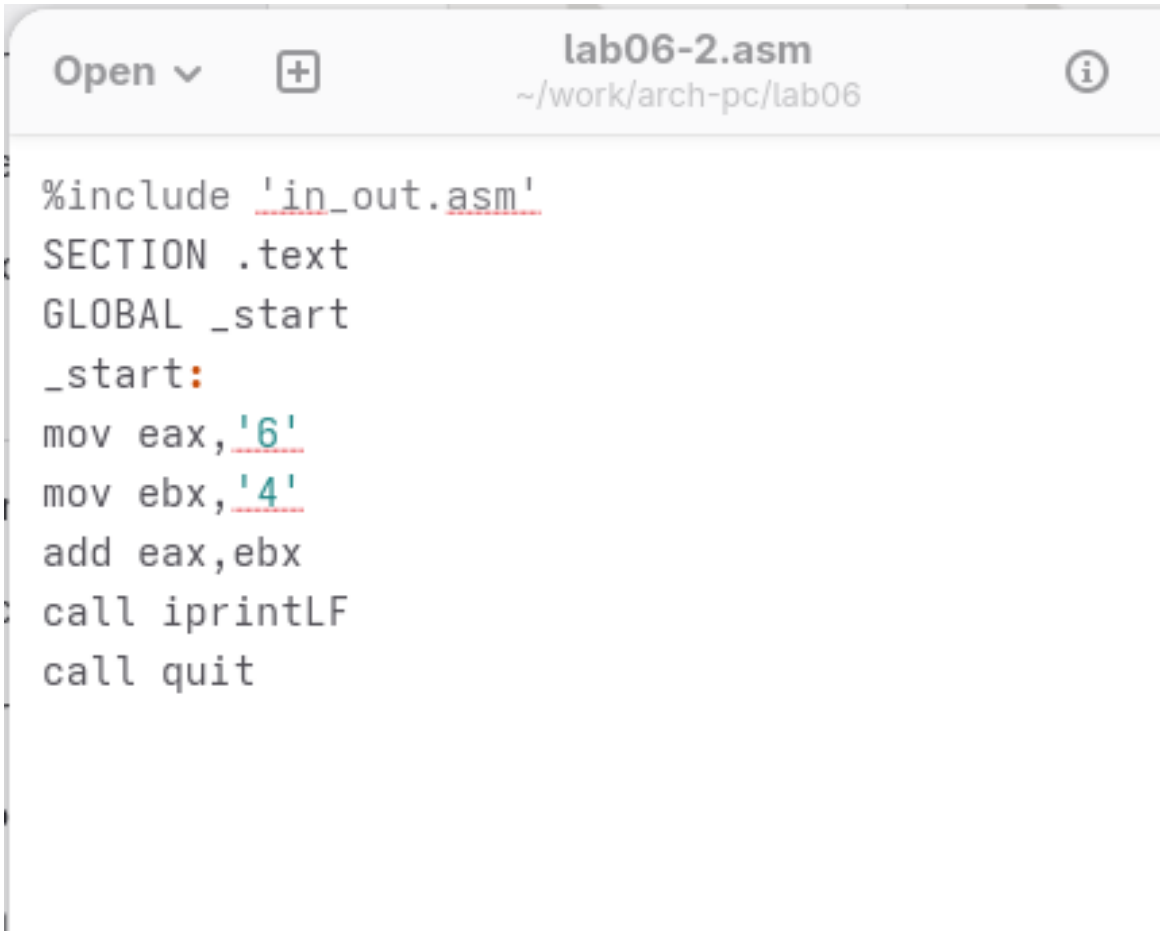

Рисунок 2.5: Программа в файле lab6-2.asm

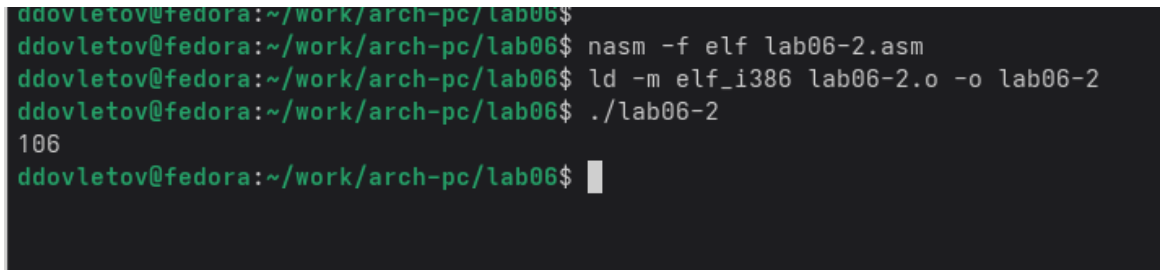

Рисунок 2.6: Запуск программы lab6-2.asm

В результате выполнения программы выводится число 106 (рис. 2.6). В этом случае команда add суммирует коды символов „6“ и „4“ (54 + 52 = 106). Однако, в отличие от предыдущей программы, функция iprintLF позволяет вывести число, а не символ, который соответствует этому числу.

Аналогично предыдущему примеру изменяем символы на числа (рис. 2.7).

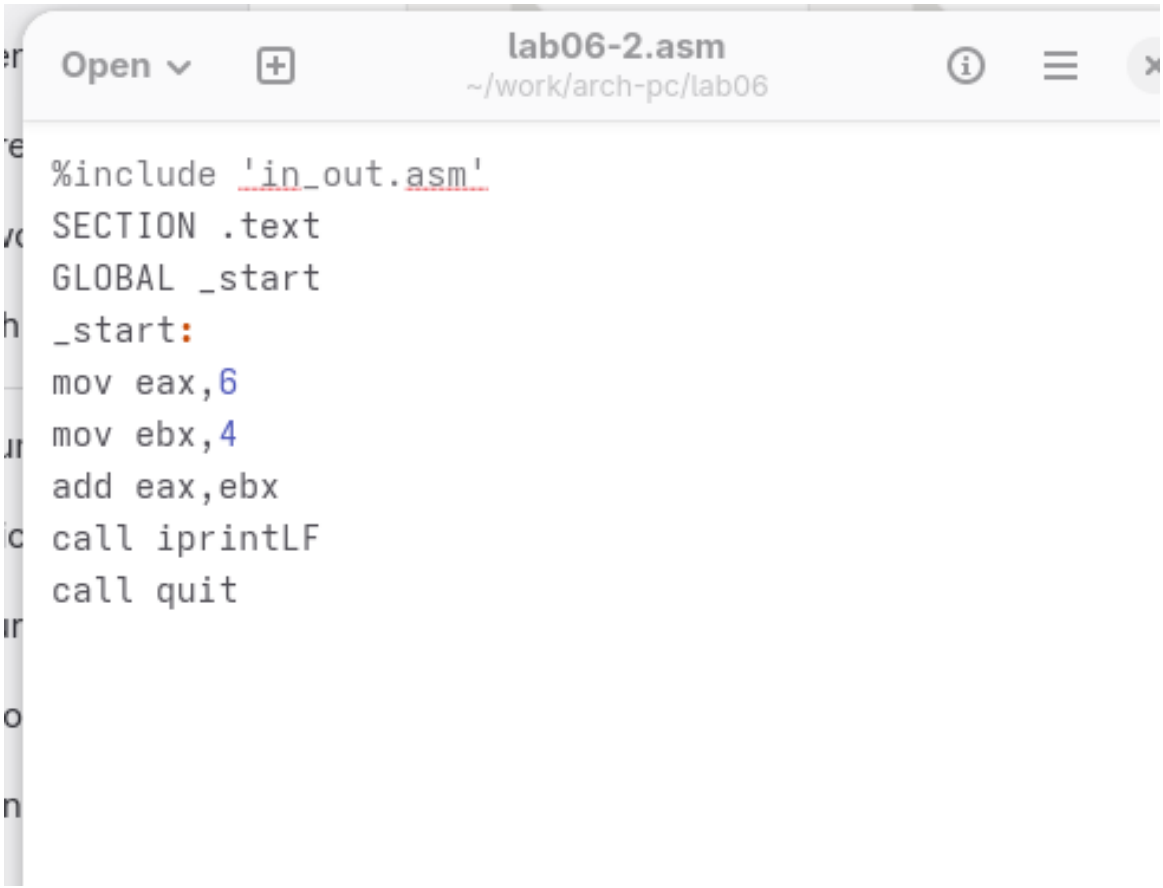

Рисунок 2.7: Программа в файле lab6-2.asm

Функция iprintLF позволяет вывести число, и операндами являются числа, а не коды символов. Поэтому на экране будет выведено число 10 (рис. 2.8).

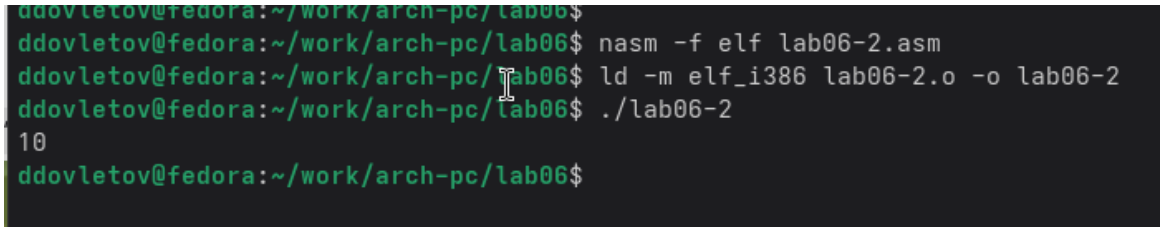

Рисунок 2.8: Запуск программы lab6-2.asm

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

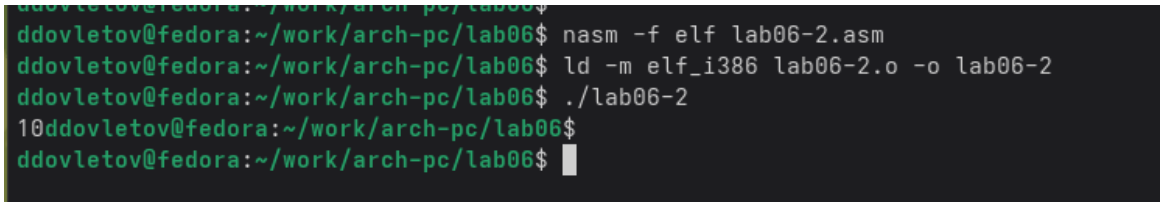

Рисунок 2.9: Запуск программы lab6-2.asm

Для примера арифметических операций в NASM привожу программу для вычисления выражения (рис. 2.10) (рис. 2.11):

$f(x) = \frac{5 \times 2 + 3}{3}$.

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

```
%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 2.10: Программа в файле lab6-3.asm

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

Рисунок 2.11: Запуск программы lab6-3.asm

Изменил программу для вычисления выражения:

$f(x) = \frac{4 \times 6 + 2}{5}$.

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

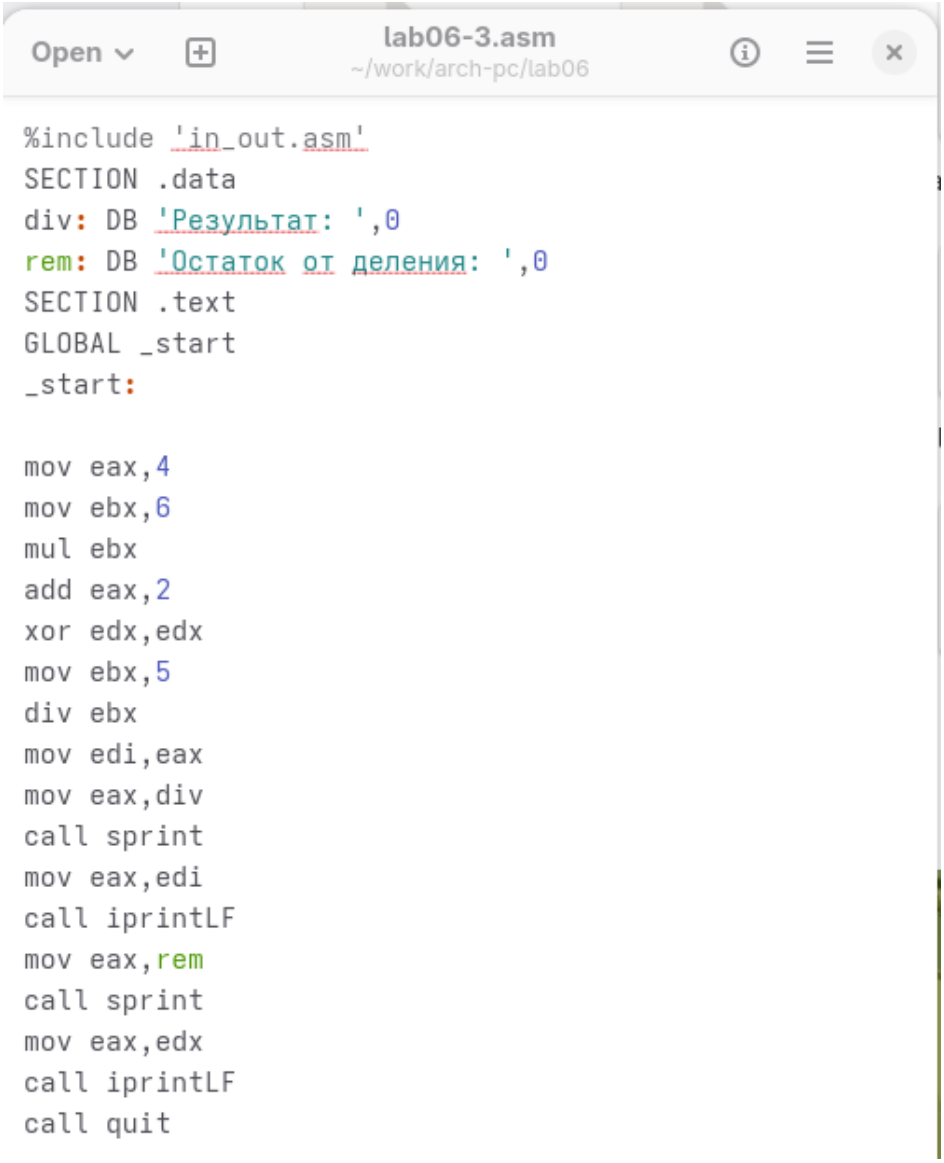

Рисунок 2.12: Программа в файле lab6-3.asm

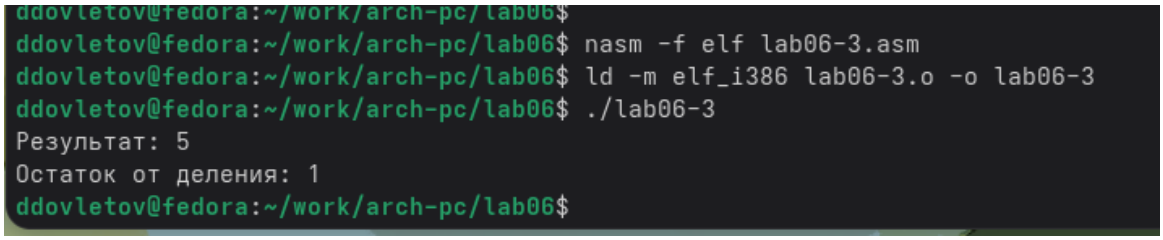

Рисунок 2.13: Запуск программы lab6-3.asm

Рассмотрим еще один пример программы для вычисления варианта задания по номеру студенческого билета (рис. 2.14) (рис. 2.15).

В этом примере число, с которым нужно проводить арифметические операции, вводится с клавиатуры. Так как ввод осуществляется в виде символов, для корректной работы арифметических операций символы необходимо преобразовать в числа. Для этого используется функция `atoi` из файла `in_out.asm`.

Open 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 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 в файле variant.asm

```
ddovletov@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
ddovletov@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 variant.o -o variant
ddovletov@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032254169
Ваш вариант: 10
ddovletov@fedora:~/work/arch-pc/lab06$
```

Рисунок 2.15: Запуск программы variant.asm

2.1 Ответы на вопросы по программе variant.asm

1. **Какие строки листинга отвечают за вывод на экран сообщения «Ваш вариант:»?**

Строка `mov eax, rem` записывает в регистр значение переменной с фразой «Ваш вариант:», а строка `call sprint` вызывает подпрограмму для вывода этой строки на экран.

2. **Для чего используются следующие инструкции?**

- `nasm`: используется для компиляции кода на языке ассемблера NASM.
- `mov ecx, x`: перемещает значение переменной `x` в регистр `ecx`.
- `mov edx, 80`: перемещает значение 80 в регистр `edx`.
- `call sread`: вызывает подпрограмму для считывания значения студенческого билета с 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`: записывает 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` увеличивает значение в регистре `edx` на 1, что соответствует формуле вычисления варианта.

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

Строка `mov eax, edx` перекладывает результат вычислений в регистр `eax`, а строка `call iprintLF` вызывает подпрограмму для вывода этого результата на экран.

2.2 Самостоятельное задание

Написана программа для вычисления выражения $ y = f(x) $. Программа выводит формулу для вычисления, запрашивает ввод значения $ x $, вычисляет выражение в зависимости от введенного $ x $ и выводит результат. В зависимости от лабораторного задания, был выбран вариант 10 — $ 5(x + 18) - 28 $ для $ x_1 = 2 $, $ x_2 = 3 $ (рис. 2.16) (рис. 2.17).

Рисунок 2.16: Программа в файле task.asm

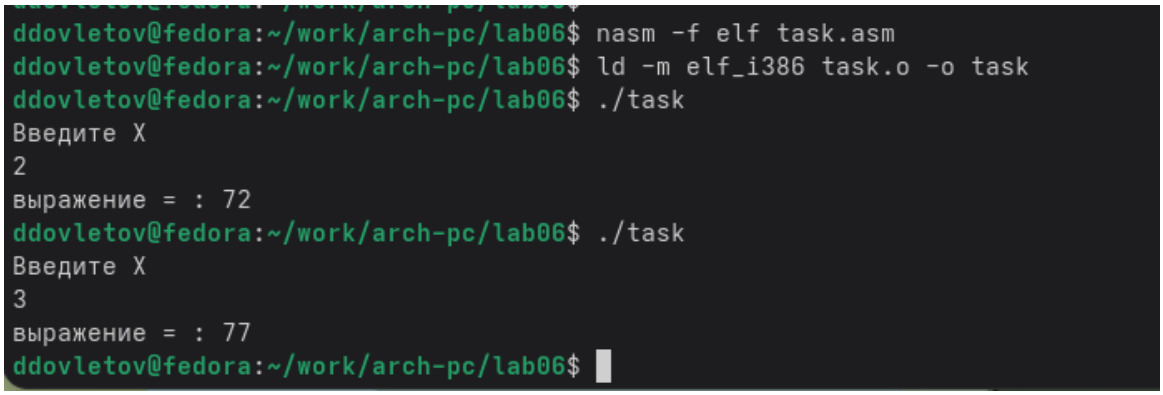

Рисунок 2.17: Запуск программы task.asm

3 Выводы

Изучили работу с арифметическими операциями.