РУДН. Архитектура компьютеров

Отчёт по лабораторной работе №6

Косинов Никита Андреевич, НПМбв-02-20

Содержание

# 1 Цель работы

ЭВМ - электронно вычислительная машина, а это значит, что в первую очередь компьютер необходим для вычислений. Поэтому основная задача любого языка программирования, идущая после ввода/вывода сообщений на экран - это предоставить возможность арифметических действий. Исключением не является и язык ассемблер.

Поэтому, цель данной работы - изучить команды, позволяющие проделывать элементарные арифметические действия на ассемблере **NASM** такие, как сложение, вычитание, умножение и деление с остатком.

# 2 Ход работы

Лабораторная работа выполнена с использованием консоли **OC Linux** и языка программирования ассемблера **NASM**.

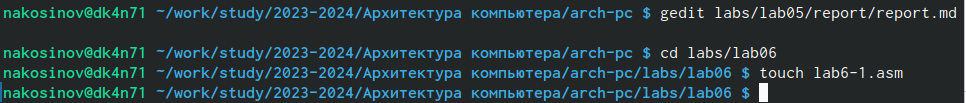
1. Вывод результата вычислений в стандартный вывод;
2. Вычисление простейшего выражения на ассемблере;
3. Применение вычислений для практических задач.

В конце выполнена самостоятельная работа.

# 3 Вывод результата вычислений в стандартный вывод

Чтобы разобраться, как работает вывод на ассемблере, используем в качестве примера операцию сложения **add**, применив её к различным вариантам данных.

1. Создадим рабочий каталог и файл программы, в которой будем проводить эксперименты.



Создание файла

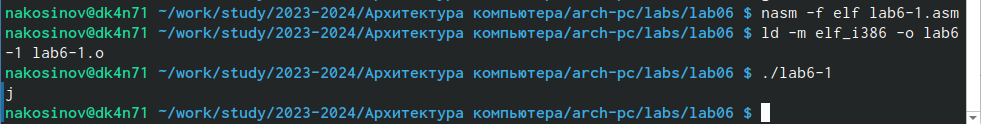
1. Напишем программу, вычисляющую сумму символов ‘4’ и ‘6’. Программа выведет результат командой **sprintLF** из подключаемого файла **in\_out.asm**.



Программа сложения символов и вывода по коду

%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

1. Скомпилируем и запустим написанную программу.



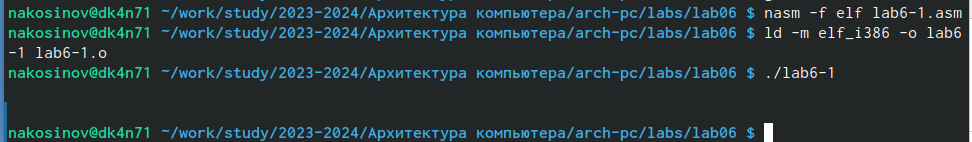
ПРезультат исполнения

1. Мы видим, что результат исполнения - это символ ‘j’. Программа так сработала, потому что при сложении символов она складывает их двоичные коды. Код символа ‘4’ в системе *ASCII* равен 00110100, а код символа ‘6’ - 00110110. Суммой этих кодов будет 01101010 (проверить это можно, например, сложив столбиком), что является кодом символа ‘j’.
2. Изменим программу, убрав апострофы, тем самым заменив символы ‘4’ и ‘6’ в регистрах на соответствующие символы.



Программа сложения чисел и вывода по коду

1. Скомпилируем полученную программу, запустим и посмотрим на результат.



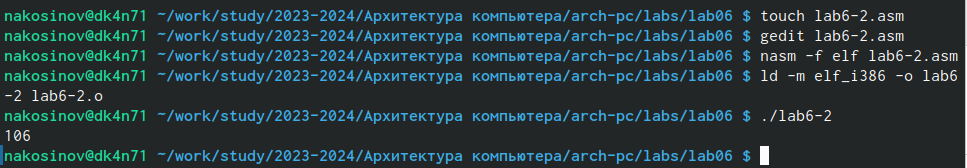
Результат исполнения

1. На первый взгляд, в консоли ничего не напечатано. Но это не так - обратим внимание на лишний образовавшийся отступ. Действительно, сумма чисел 4 и 6 равна 10, но команду вывода мы не поменяли, т.е. она показывает символ с кодом 10, а это как раз перенос каретки.
2. Снова изменим нашу программу. На этот раз добавим апострофы, чтобы складывать именно символы, но изменим вывод на **iprintLF**.



Программа сложения символов и вывода результата

1. Скомпилируем и запустим.



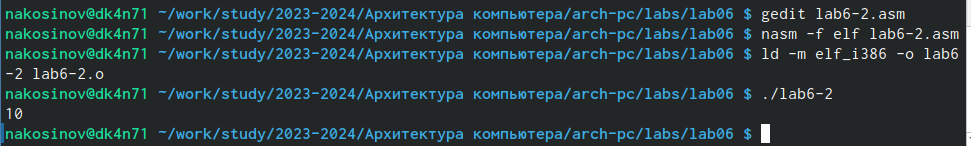
Результат исполнения

1. На терминале получаем число 106. Действительно, **iprint** вываодит уже численный результат операции, но складывали мы символы, а операция **add** при этом складывает их коды. Тем самым мы получаем в результате 01101010, как в первом случае, но теперь мы выводим этот код как число, причём в десятичной записи: .
2. Снова изменим программу, оставив численный вывод и заменив символы на числа.



Программа сложения чисел и вывода результата

1. Скомпилируем и запустим.



Результат исполнения

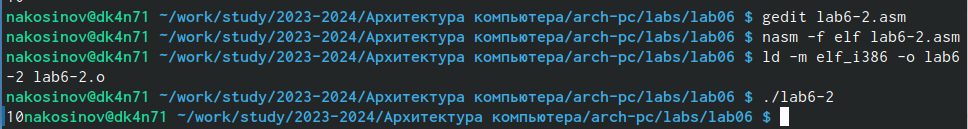
1. Мы, наконец, получаем результат сложения чисел 4 и 6.
2. В подключаемом файле **in\_out.asm** наряду с командами **sprintLF** и **iprintLF** есть команды **sprint** и **iprint**. Чтобы понять разницу в их работе, изменим команду вывода в последней программе.



Программа с выводом без переноса на новую строку

%include 'in\_out.asm'  
  
SECTION .text  
 GLOBAL \_start  
  
\_start:  
 mov eax,6  
 mov ebx,4  
 add eax,ebx  
 call iprint  
 call quit

1. Скомпилируем и запустим.



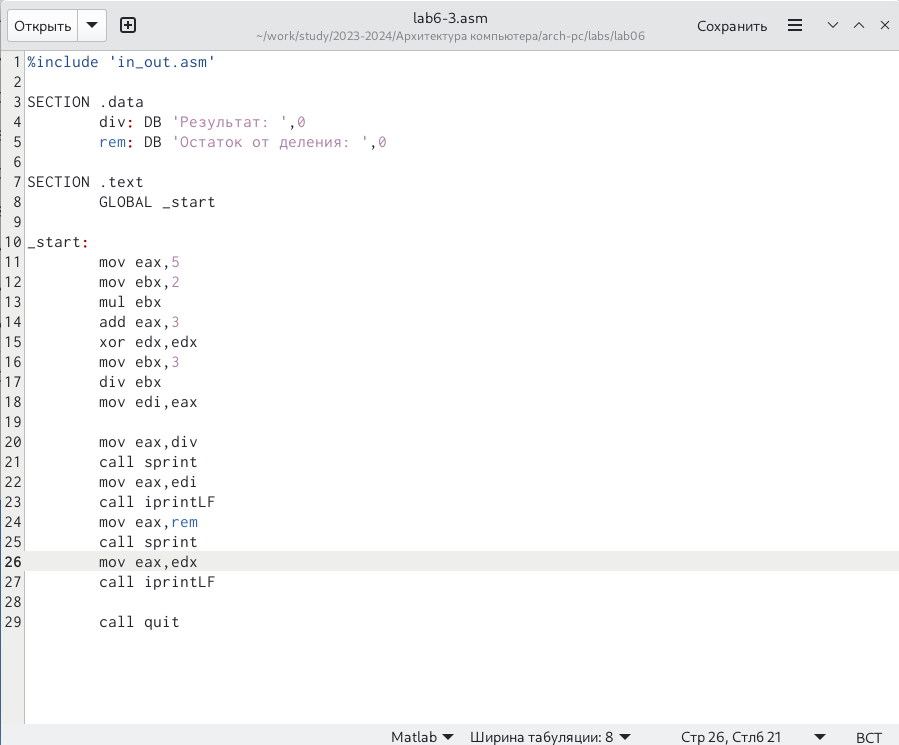
Результат исполнения

1. Результат: без добавленного **LF** в конце команды вывода, мы не получаем перенос строки на новую.

# 4 Вычисление простейшего выражения на ассемблере

Выводить результат в численном виде мы научились, теперь разберём, как этот результат получить. Предлагается с помощью языка посчитать значения выражения .

1. Напишем программу, вычисляющую данное выражение.

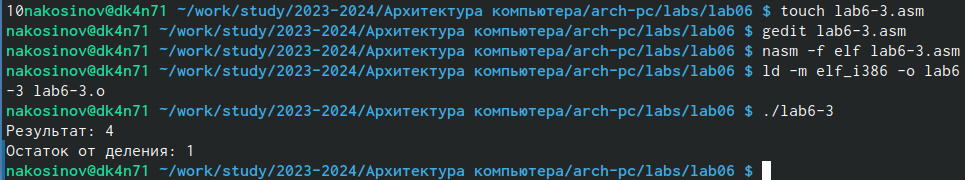


Программа вычисления A

1. Разберём, что происходит в коде:

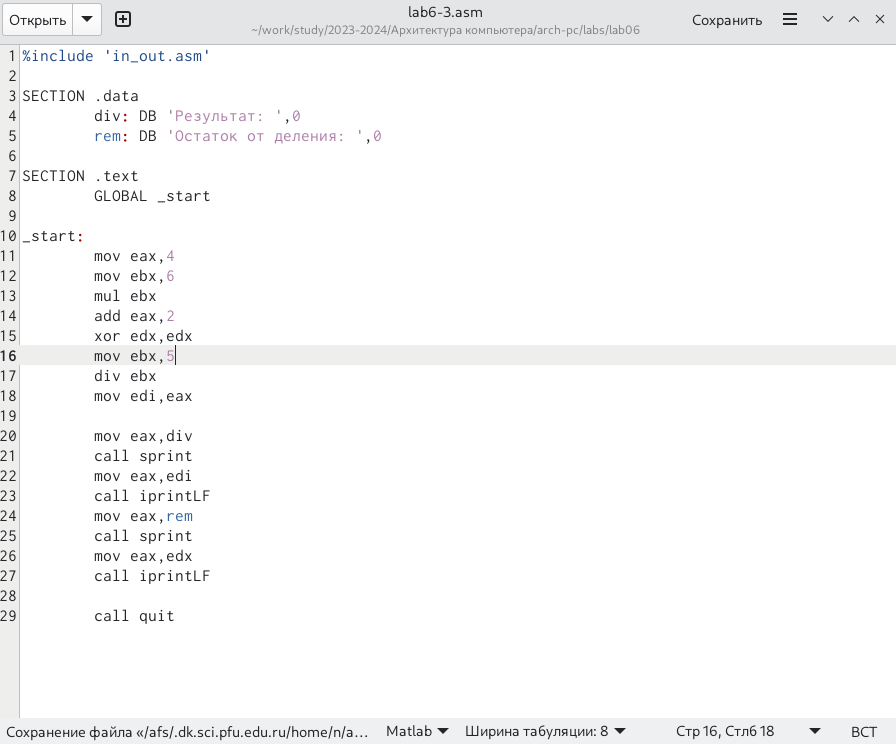
* 2.1. Записываем 5 и 2 в регистры **eax** и **ebx**.
* 2.2. Увеличиваем число из **eax**, т.е. 5, в число раз, записанное в **ebx**, т.е. 2. Получаем 10.
* 2.3. Увеличиваем **eax**, равное 10, на 3.
* 2.4. Обнуляем значение, оставшееся с прошлых разов в регистре **edx**.
* 2.5. Записываем в **ebx** делитель - число 3.
* 2.6. Делим число **eax** (13) на **ebx**. Неполное частное от деления (4) остаётся в **eax**, а остаток уходит в регистр **edx**.
* 2.7. Для вывода нам нужен свободный регистр **eax**, поэтому мы переписываем полученное частное в другой - **edi**, дабы его не потерять.
* 2.8. Последовательно выводим сообщения на экран и полученные значения.

1. Результат после компиляции.



Результат исполнения

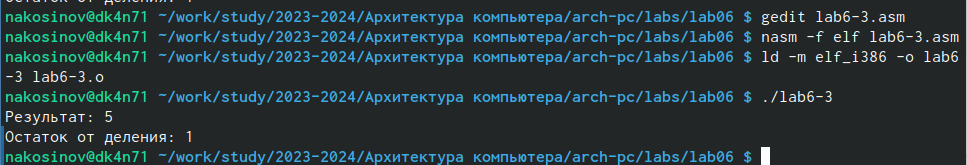
1. Изменим код программы для вычисления другого выражения: .



Программа вычисления B

%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

1. Скомпилируем и запустим. Убедимся, что программа выполнена корректно.

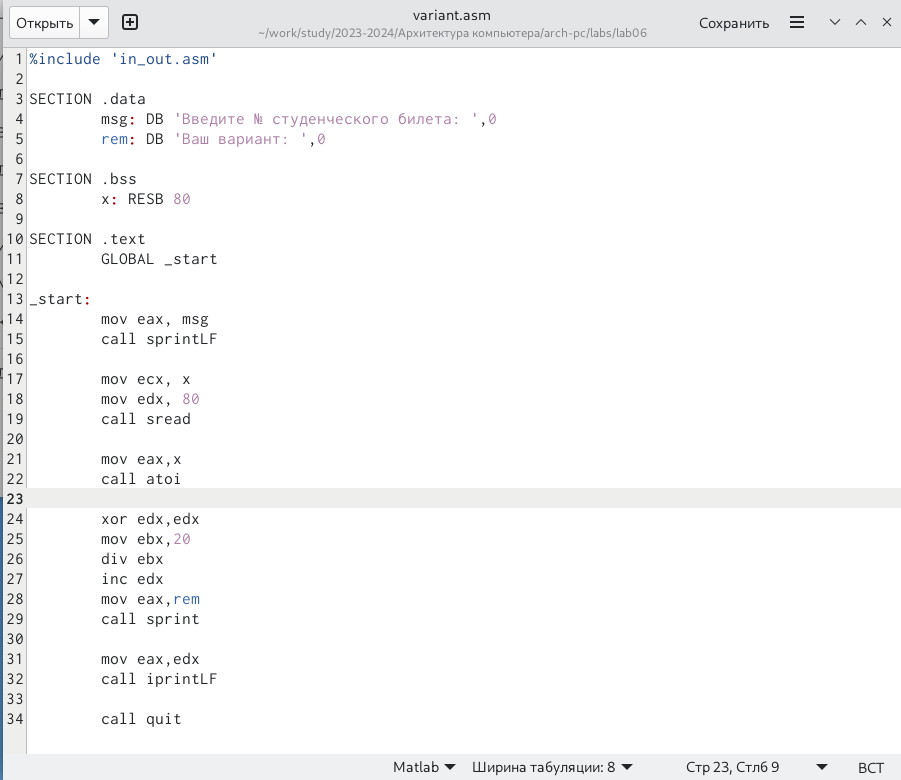


Запуск менеджера

# 5 Применение вычислений для практических задач

Разберём вариант задачи, где вычисления нужны для какой-то практической цели. Для большинства таких задач нам необходимо посчитать не просто значение какого-то числа, но в зависимости от введённых пользователем данных.

1. Напишем программу, вычисляющую номер варианта работы по номеру студенческого билета, как остаток от деления этого номера на 20 (количество вариантов).



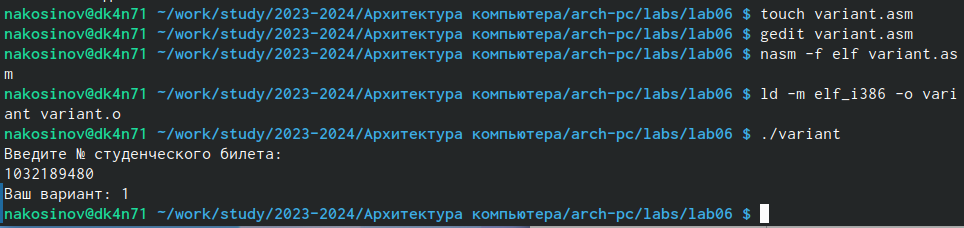
Программа вычисления варианта

%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

1. Разберём код:

* 2.1. Сохраняем стандартные сообщения для общения с пользователем в переменные **msg** и **rem**.
* 2.2. Выведем сообщение, предлагающие ввести номер билета.
* 2.3. Считываем номер билета, предварительно подготовив регистры **ecx** и **edx**.
* 2.4. Переносим билет в рабочий регистр **eax**, с которым и будем проводить операции. Функцией **atoi** из подключаемого файла преобразуем объект в регистре из кода в число.
* 2.5. Считаем остаток от деления на 20. Он сохраняется в заранее обнулённом регистре **edx**.
* 2.6. Остаток при делении на 20 может принимать значения от 0 до 19, а варианты заданий - от 1 до 20. Чтобы нормировать эти два набора, увеличим остаток на единицу инструкцией **inc**.
* 2.7. Выведем на экран сообщение с результатом - номером варианта задания.

1. Скомпилируем и запустим. Ввожу номер своего студенческого билета: 1032189480.

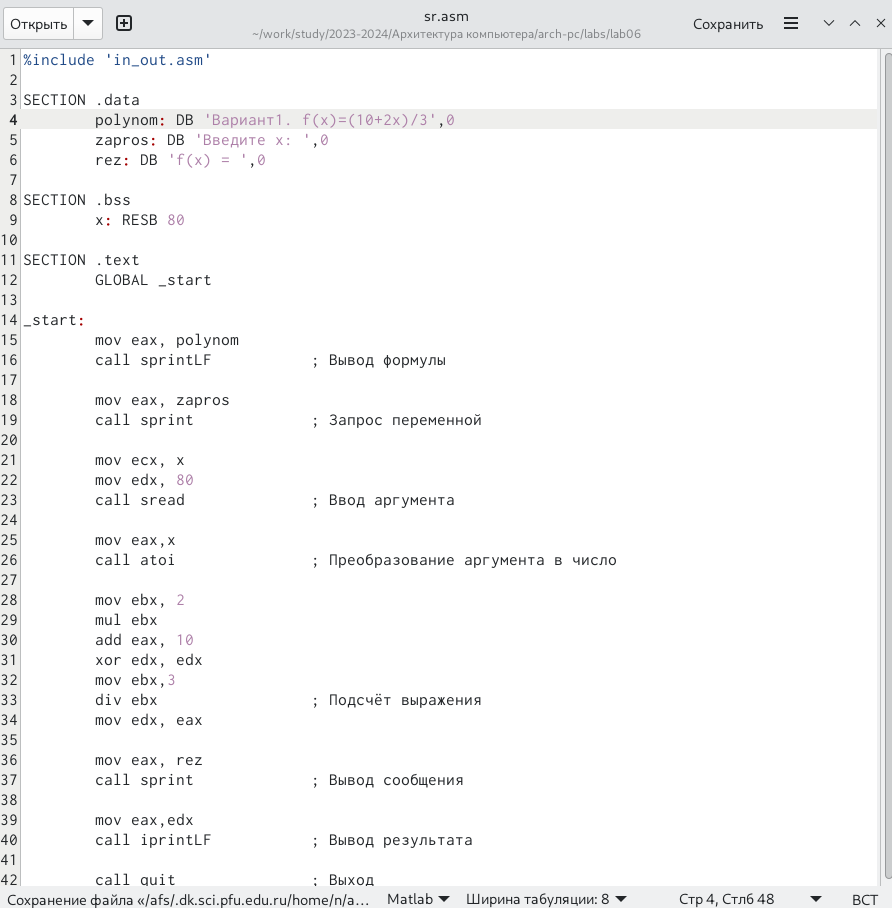


Результат исполнения

# 6 Самостоятельная работа

При запуске предыдущей программы, мне по номеру билета выпал 1-й вариант. Действительно, 1032189480 делится нацело на 20. Поэтому самостоятельная работа состоит в вычислении значений выражения для с точностью до остатка для значений и .

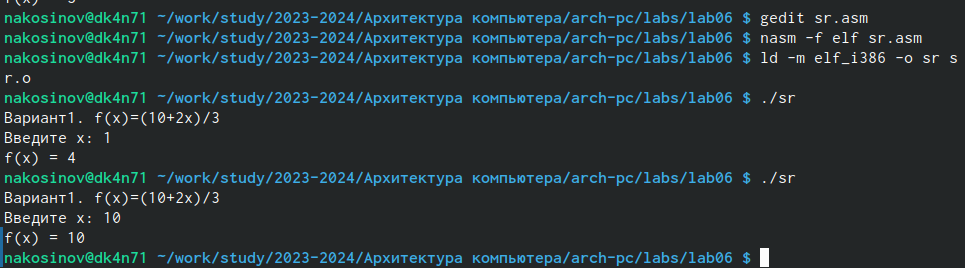
1. Напишем код программы. Добавим приглашающие сообщения и комментарии.



Программа вычисления выражения f(x)

%include 'in\_out.asm'  
  
SECTION .data  
 polynom: DB 'Вариант1. f(x)=(10+2x)/3',0  
 zapros: DB 'Введите x: ',0  
 rez: DB 'f(x) = ',0  
  
SECTION .bss  
 x: RESB 80  
  
SECTION .text  
 GLOBAL \_start  
  
\_start:  
 mov eax, polynom  
 call sprintLF ; Вывод формулы  
   
 mov eax, zapros  
 call sprint ; Запрос переменной  
   
 mov ecx, x  
 mov edx, 80  
 call sread ; Ввод аргумента  
   
 mov eax,x   
 call atoi ; Преобразование аргумента в число  
   
 mov ebx, 2  
 mul ebx   
 add eax, 10   
 xor edx, edx  
 mov ebx,3   
 div ebx ; Подсчёт выражения  
 mov edx, eax  
   
 mov eax, rez  
 call sprint ; Вывод сообщения  
   
 mov eax,edx  
 call iprintLF ; Вывод результата  
   
 call quit ; Выход

1. Скомпилируем и запустим. Убедимся, что результат выполнения верный. Действительно, и .



Результат исполнения

# 7 Выводы

В ходе данной лабораторной работы мы научились пользоваться основными арифметическими операциями, выводить результат вычислений на экран в десятичной форме с помощью подключаемого файла, а также рассмотрели, как работает операция сложения для нечисленных объектов.