Лабораторная работа №6

Архитектура компьютера

Овчинников Антон Григорьевич

Содержание

# 1 Цель работы

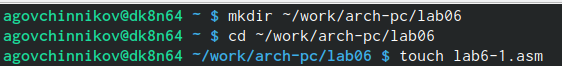
Целью работы является освоение арифметических инструкций языка ассемблера NASM

# 2 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес опе- ранда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации. Существует три основных способа адресации: • Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: mov ax,bx. • Непосредственная адресация – значение операнда задается непосредственно в ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию. Например, определим переменную intg DD 3 – это означает, что задается область памяти размером 4 байта, адрес которой обозначен меткой intg. В таком случае, команда mov eax,[intg] копирует из памяти по адресу intg данные в регистр eax. В свою очередь команда mov [intg],eax запишет в память по адресу intg данные из регистра eax. Также рассмотрим команду mov eax,intg В этом случае в регистр eax запишется адрес intg. Допустим, для intg выделена память начиная с ячейки с адресом 0x600144, тогда команда mov eax,intg аналогична команде mov eax,0x600144 – т.е. эта команда запишет в регистр eax число 0x600144

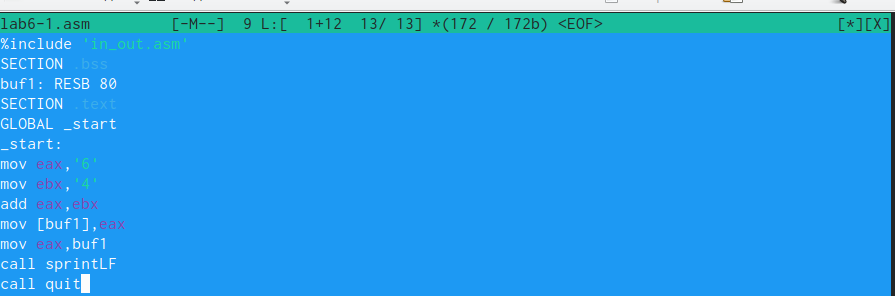
# 3 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы №6, создаю там файл lab6-1.asm (рис. ??).



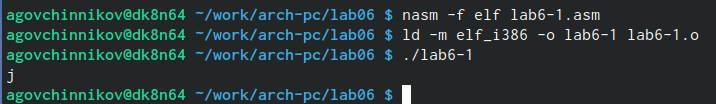
Создание каталога и файла

Ввожу в файл lab6-1.asm текст из листинга 6.1 (рис. ??)



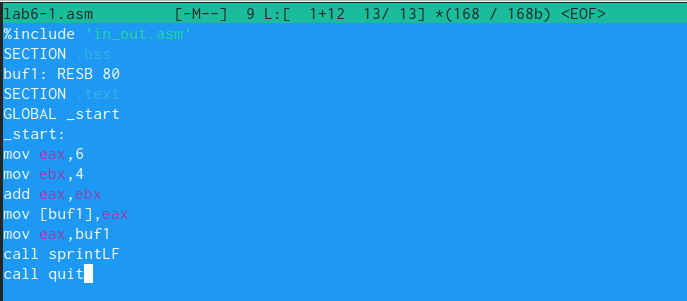
Файл lab6-1.asm

Создаю исполняемый файл и запускаю его (рис. ??)



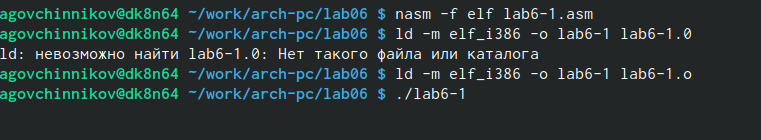
Запуск программы

Изменяю текст программы, чтобы программа вывела другой результат (рис. ??)



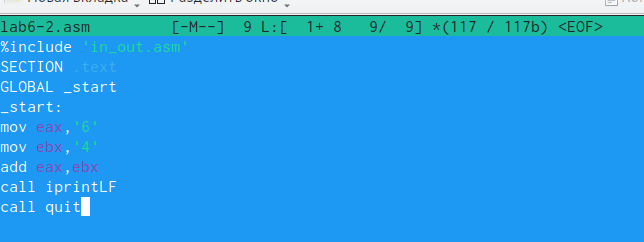
Изменненый текст программы

Запускаю изменненый файл и смотрю вывод новой команды (рис. ??)

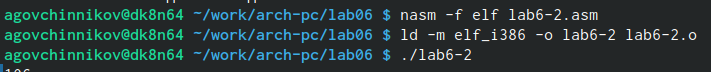


Результат программы

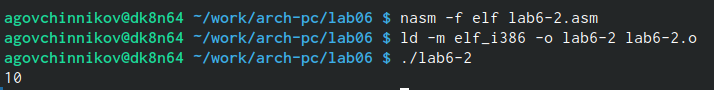
Создаю файл lab6-2.asm и ввожу туда текст из листинга (рис. ??)

{#fig:fig006 width=70%)

Создаю исполняемый файл и запускаю его (рис. ??)

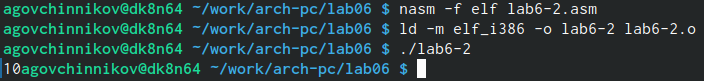
{#fig:fig007 width=70%)

Изменяю текст файла и запускаю новую программу (рис. ??)



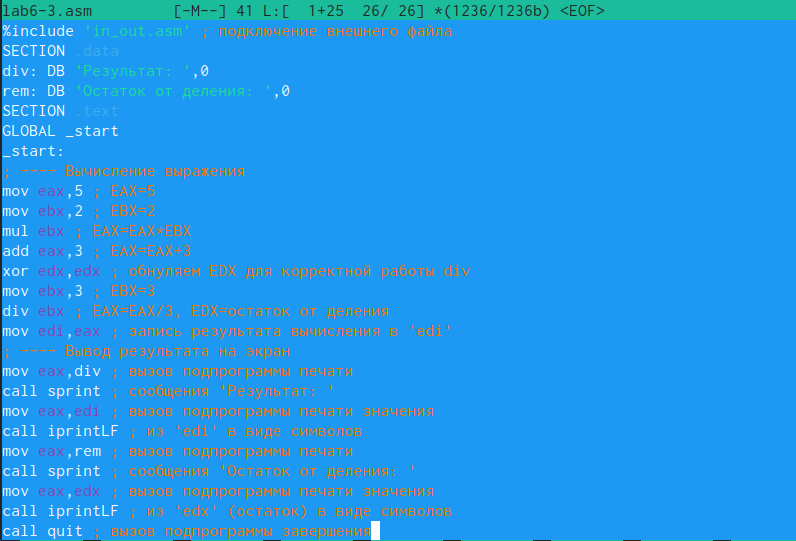
Запуск программы

Заменяю в тексте файла функцию iprintLF на iprint (рис. ??). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.



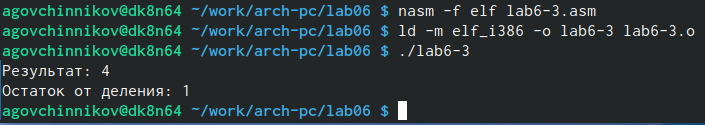
Запуск изменненой программы

Создаю файл lab6-3.asm и ввожу туда текст листинга 6.3 (рис. ??)



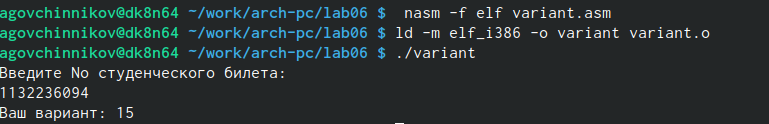
Файл lab6-3.asm

Создаю исполняемый файл и запускаю его (рис. ??)



Результат программы

Создаю файл variant.asm чтобы узнать номер своего варианта для выполнения самостоятельной работе (рис. ??)



Запуск программы

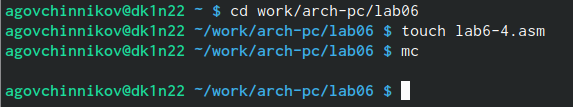
#Ответы на вопросы

1.mov eax,rem  
call sprint  
2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в регистр ecx mov edx, 80 - запись в регистр edx длины вводимой строки call sread - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры  
3. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax  
4. За вычисления варианта отвечают строки:

xor edx,edx ; обнуление edx для корректной работы div mov ebx,20 ; ebx = 20 div ebx ; eax = eax/20, edx - остаток от деления inc edx ; edx = edx + 1 5. При выполнении инструкции div ebx остаток от деления записывается в регистр edx 6. Инструкция inc edx увеличивает значение регистра edx на 1 7. За вывод на экран результатов вычислений отвечают строки: mov eax,edx call iprintLF

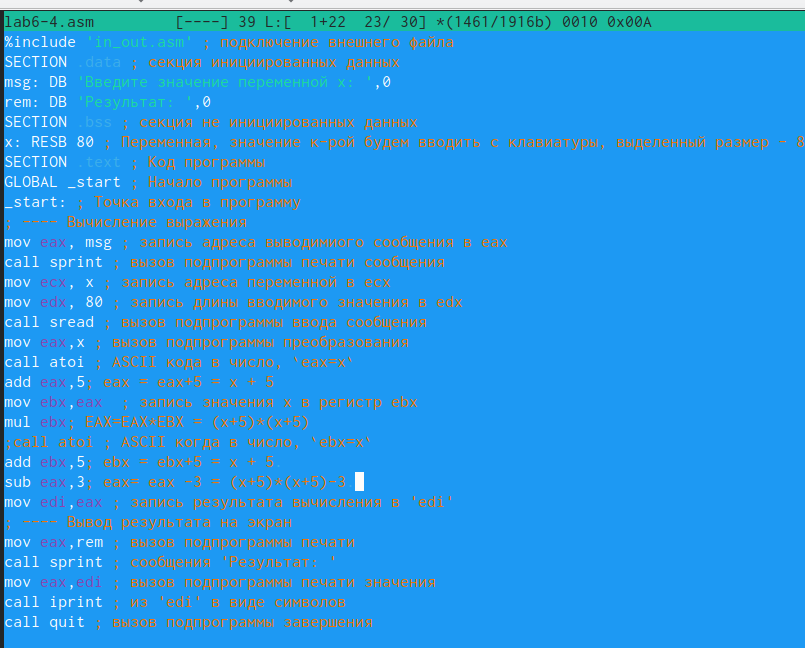
#Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью команды touch (рис. ??)



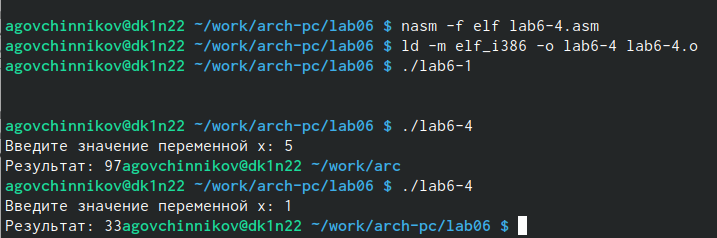
Создание файла

Открываю созданный файл и ввожу текст программы, которая решит мое уравнение (рис. ??)



Текст программы

Запуск написанной программы (рис. ??)



Вывод программы

#Листинг

%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

#Выводы

Я освоил арифметические инструкции языка ассемблера NASM.