Лабораторная работа №6

Архитектура компьютера

Овчинников Антон Григорьевич

Содержание

Список иллюстраций

Список таблиц

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 символическое 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. **??**).

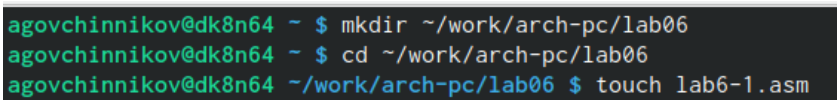


Рис. 3.1: Создание каталога и файла

Ввожу в файл lab6-1.asm текст из листинга 6.1 (рис. 3.2)

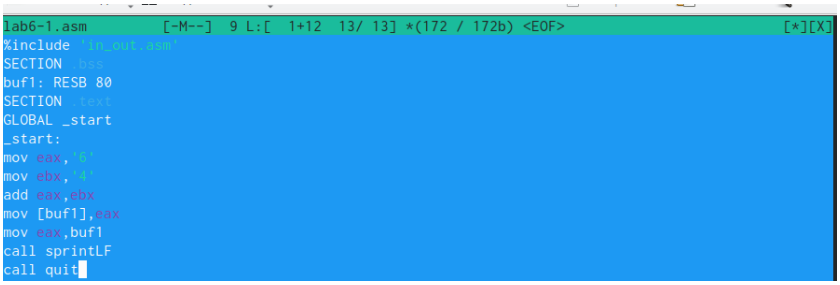


Рис. 3.2: Файл lab6-1.asm

Создаю исполняемый файл и запускаю его (рис. 3.3)

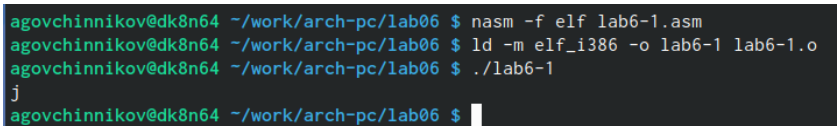


Рис. 3.3: Запуск программы

Изменяю текст программы, чтобы программа вывела другой результат (рис. 3.4)

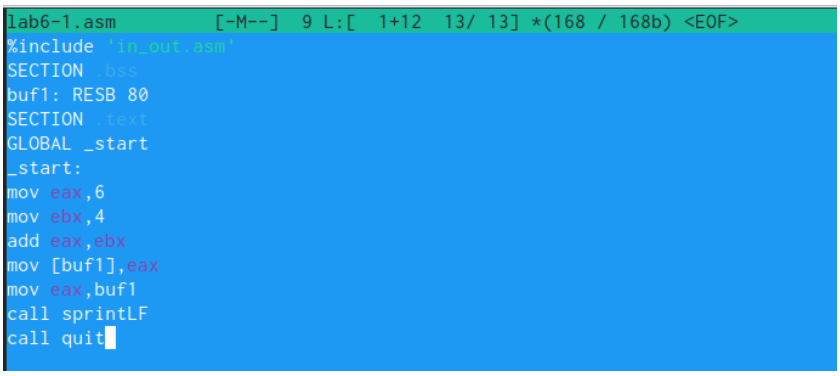


Рис. 3.4: Изменненый текст программы

Запускаю изменненый файл и смотрю вывод новой команды (рис. 3.5)

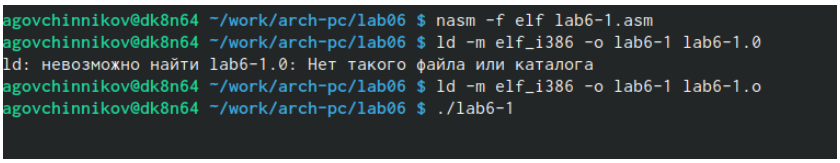
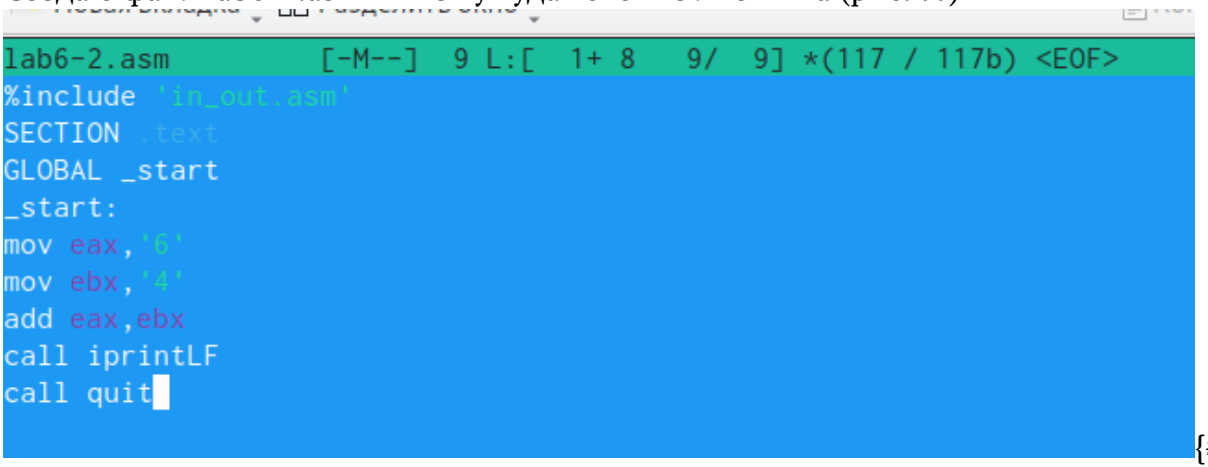


Рис. 3.5: Результат программы

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



{#fig:fig006 width=70%)

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

```
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-2
```
{#fig:fig007 width=70%)

Изменяю текст файла и запускаю новую программу (рис. 3.6)

```
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 3.6: Запуск программы

Заменяю в тексте файла функцию iprintLF на iprint (рис. 3.7). 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.

```
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-2
10agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $
```

Рис. 3.7: Запуск изменненой программы

Создаю файл lab6-3.asm и ввожу туда текст листинга 6.3 (рис. 3.8)

```
lab6-3.asm         [-M--] 41 L:[  1+25  26/ 26] *(1236/1236b) <EOF>
%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. 3.8: Файл lab6-3.asm

Создаю исполняемый файл и запускаю его (рис. 3.9)

```
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $
```

Рис. 3.9: Результат программы

Создаю файл variant.asm чтобы узнать номер своего варианта для выполнения самостоятельной работе (рис. 3.10)

```
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $  nasm -f elf variant.asm
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
agovchinnikov@dk8n64 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236094
Ваш вариант: 15
```

Рис. 3.10: Запуск программы

#Ответы на вопросы

`1.mov eax,rem`

```
call sprint
2. Инструкция mov ecx, x используется, чтобы положить адрес вводимой строки x в р
3. call atoi используется для вызова подпрограммы из внешнего файла, которая прео
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. 3.11)

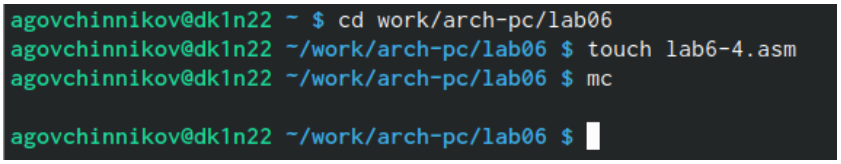


Рис. 3.11: Создание файла

Открываю созданный файл и ввожу текст программы, которая решит мое уравнение (рис. 3.12)

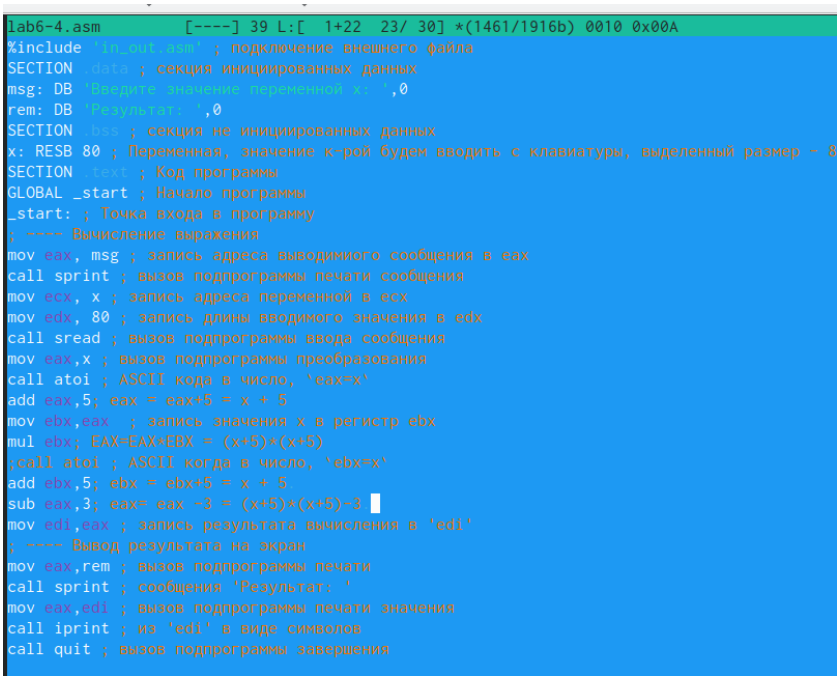


Рис. 3.12: Текст программы

Запуск написанной программы (рис. 3.13)

Рис. 3.13: 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.