

ОТЧЕТ по лабораторной работе № 6

дисциплина: Архитектура компьютера

Студент: Идрисов Д.А

Содержание

1	Цель работы	1
2	Задания	1
3	Теоретическое введение	1
4	Выполнение лабораторной работы	2
4.1	1 Символьные и численные данные в NASM	2
4.2	Выполнение арифметических операций в NASMц	3
4.3	Задание для самостоятельной работы	5
5	Выполнение заданий	6
6	Выводы	6
7	Список литературы	7

1 Цель работы

Цель этого лабораторного исследования заключается в изучении арифметических инструкций языка ассемблера NASM.

2 Задания

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

Большинство команд в языке ассемблера требуют обработки операндов, которые представляют место, где хранятся данные для последующей обработки. Эти данные могут находиться в регистрах или в ячейках памяти. Существуют три основных типа адресации:

Регистровая адресация, при которой операнды хранятся в регистрах и их имена используются в командах, например: `mov ax, bx`. Непосредственная адресация, где значение операнда задается непосредственно в команде, например: `mov ax, 2`. Адресация памяти, где операнд указывает на адрес в памяти, используя символическое обозначение ячейки памяти, над содержимым которой выполняется операция. Ввод данных с клавиатуры и вывод на экран обычно осуществляются в символьной форме, используя коды ASCII. Каждый символ кодируется одним байтом согласно стандарту ASCII. Некоторые команды NASM не предоставляют прямого вывода чисел (не в символьной форме), поэтому для вывода чисел на экран необходимо предварительно преобразовать их цифры в ASCII-коды.

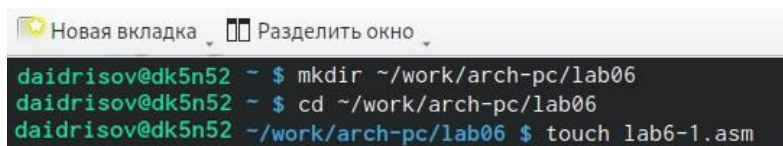
При вводе данных с клавиатуры также возникает необходимость преобразования ASCII-символов в числа для корректного выполнения арифметических операций.

Таким образом, для решения этих проблем необходимо проводить преобразование между ASCII и числовыми значениями при вводе и выводе данных.

4 Выполнение лабораторной работы

4.1 1 Символьные и численные данные в NASM

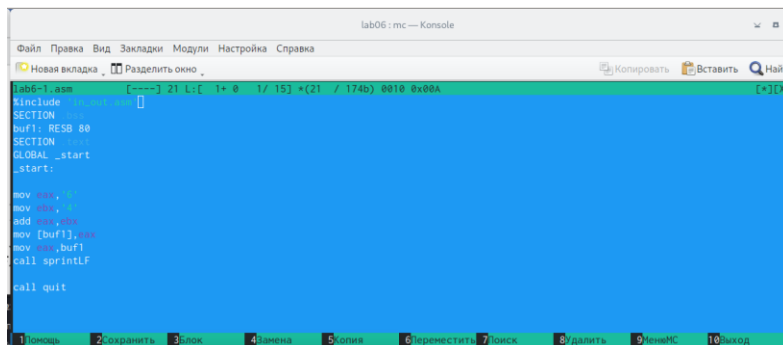
Давайте перейдем в репозиторий и используем команду для создания файла с именем `lab6-1.asm`. (рис. [??]).



```
daidrisov@dk5n52 ~ $ mkdir ~/work/arch-pc/lab06
daidrisov@dk5n52 ~ $ cd ~/work/arch-pc/lab06
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Переход в репозиторий и создание

Я скопирую файл `in_out.asm` в наш репозиторий и приступлю к редактированию файла `lab6-1.asm`. (рис. [??]).



```
lab06: mc - Console
Файл Правка Вид Закладки Модули Настройка Справка
[?] 21 L: [?] 1+ 0 1/ 15] * (21 / 1740) 0010 0x00A [?] [X]
include
SECTION
buf1: RESB 80
SECTION
GLOBAL _start
_start:
mov eax, 0
mov ebx, 0
add ebx, eax
mov [buf1], ebx
mov _buf1, buf1
call sprintf
call quit
```

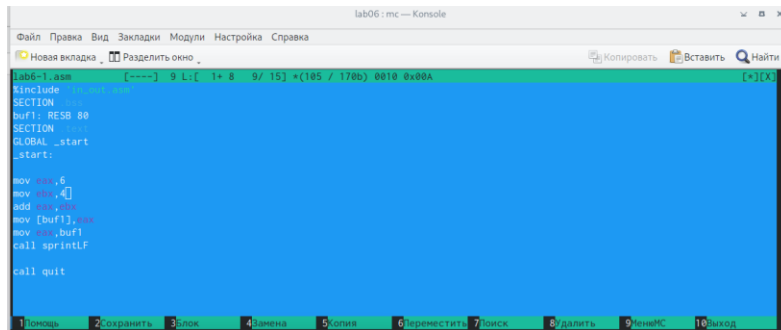
Редактирование файла

После запуска файла мы увидим символ 'j', так как программа выводит результат, соответствующий сумме двоичных кодов символов '4' и '6' в системе ASCII. (рис. [??]).

```
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Запуск файла

Я заменяю символы “6” и “4” в тексте программы на цифры 6 и 4. (рис. [??]).



Редактирование файла

После выполнения программы я замечаю, что выводится символ с кодом 10, что соответствует символу перевода строки. Этот символ не будет виден при выводе на экран. (рис. [??]).

```
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ ./lab6-1
```

Запуск файла

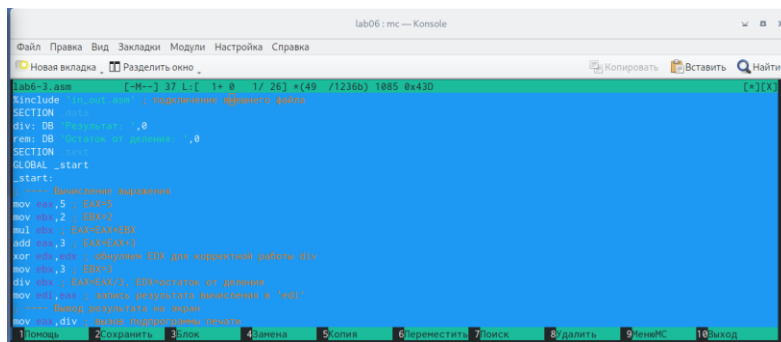
4.2 Выполнение арифметических операций в NASM

Я создам файл с именем lab6-3.asm для последующих заданий. (рис. [??]).

```
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ touch lab6-3.asm
daidrisov@dk5n52 ~/work/arch-pc/lab06 $
```

Создам файл

Я внесу новый код для вычисления математической формулы. (рис. [??]).



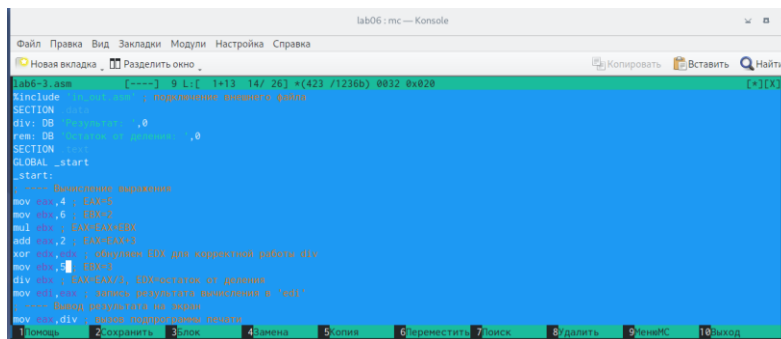
Редактирование файла

Мы выполним запуск файла и убедимся, что все операции выполняются корректно. (рис. [??]).

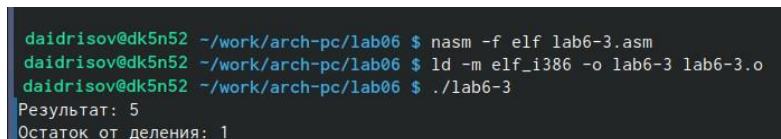


Запуск файла

Я внесу изменения в файл, после чего перезапущу программу, чтобы убедиться в правильности её работы. (рис. [??]) (рис. [??]).



Редактирование файла



Запуск файла

Я сформирую файл с названием "variant". (рис. [??]).

```
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ touch variant.asm
daidrisov@dk5n52 ~/work/arch-pc/lab06 $
```

Создаю файл

Я проведу редактирование файла. (рис. [??]).

```
variant.asm
[Меню] 9.1.1 1+14 15/253 +(275 / 384b) 0010 0x00A [X]
#include <stdio.h>
SECTION .text
msg: DB "Введите № студенческого билета: ",0
rem: DB "Ваш вариант: ",0
SECTION .data
x: RESB 80
SECTION .start
GLOBAL _start
_start:
mov esi, msg
call sprintf
mov esi, x
mov ecx, 80
call cread
mov edi, 0
call atoi
xor edx, edx
mov ebx, 20
div ebx
```

Редактирование файла

Я выполняю запуск программы. (рис. [??]).

```
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132232876
Ваш вариант: 2
daidrisov@dk5n52 ~/work/arch-pc/lab06 $
```

Запуск файла

4.3 Задание для самостоятельной работы

1. Ответственные за вывод сообщения “Ваш вариант” являются строки кода:

```
mov eax, rem
call sprint
```

2. Инструкция `mov ecx, x` применяется для помещения адреса вводимой строки `x` в регистр `ecx`. Затем, с использованием инструкции `mov edx, 80`, указывается в регистр `edx` длина вводимой строки. Далее, вызывается подпрограмма `sread` из внешнего файла с помощью команды `call sread`, обеспечивающая ввод сообщения с клавиатуры.
3. Инструкция `call atoi` применяется для вызова подпрограммы из внешнего файла, которая осуществляет преобразование ASCII-кода символа в целое число и сохраняет результат в регистре `eax`.
4. Вычисление варианта осуществляется в следующих строках кода:

```
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 помещается в регистр `edx`.
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1.
7. Ответственные за вывод результатов вычислений на экран строки кода:

```
mov eax,edx  
call iprintLF
```

5 Выполнение заданий

Я разработаю программу для выполнения поставленной задачи. (рис. [??]).

```
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ touch lab06-4.asm
```

Создам файл

Я вношу изменения в файл, чтобы обеспечить корректное выполнение и вычисление моей задачи. (рис. [??]).



Редактирование файла

Я выполняю запуск программы. (рис. [??]).

```
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ nasm -f elf lab06-4.asm  
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab06-4.asm lab06-4.o  
daidrisov@dk5n52 ~/work/arch-pc/lab06 $ ./lab06-4.asm  
Введите значение переменной x: 4  
Результат: 24
```

Запуск программы

6 Выводы

В процессе выполнения данной лабораторной работы я приобрела навыки использования арифметических инструкций в языке ассемблера NASM.

7 Список литературы

Лабораторная работа №6. Арифметические операции в NASM. ::: {#refs} :::