

# **Лабораторная работа №6.**

**Арифметические операции в NASM.**

Селиванов Вячеслав Алексеевич

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Символьные и численные данные в NASM . . . . .	8
4.2	Выполнение арифметических операций в NASM . . . . .	14
4.2.1	Ответы на вопросы . . . . .	19
4.3	Задания для самостоятельной работы . . . . .	20
<b>5</b>	<b>Выводы</b>	<b>24</b>
	<b>Список литературы</b>	<b>25</b>

## Список иллюстраций

4.1	Создание каталога . . . . .	8
4.2	Создание файла . . . . .	8
4.3	Копирование файла . . . . .	9
4.4	Редактирование файла . . . . .	9
4.5	Запуск программы . . . . .	10
4.6	Редактирование программы . . . . .	10
4.7	Запуск исполняемого файла . . . . .	11
4.8	Создание нового файла . . . . .	11
4.9	Редактирование программы . . . . .	12
4.10	Запуск программы . . . . .	12
4.11	Редактирование программы . . . . .	13
4.12	Запуск программы . . . . .	13
4.13	Редактирование программы . . . . .	14
4.14	Запуск программы . . . . .	14
4.15	Создание файла . . . . .	14
4.16	Редактирование программы . . . . .	15
4.17	Запуск программы . . . . .	15
4.18	Редактирование программы . . . . .	16
4.19	Запуск программы . . . . .	17
4.20	Создание файла . . . . .	17
4.21	Редактирование программы . . . . .	18
4.22	Запуск программы . . . . .	18
4.23	Создание файла . . . . .	20
4.24	Редактирование программы . . . . .	21
4.25	Запуск программы . . . . .	23

## Список таблиц

# 1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

## 2 Задание

Символьные и численные данные в NASM Выполнение арифметических операций в NASM Задания для самостоятельной работы

### 3 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда 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 команде, Например: `mov ax,2`.
- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

С помощью команды `mkdir` создаю новую директорию, в которой буду создавать файлы с программами во время всей лабораторной работы №6 (рис. 4.1).

Перехожу в созданный каталог

```
[vaselivanov@fedora ~]$ mkdir ~/work/arch-pc/lab06  
[vaselivanov@fedora ~]$ cd ~/work/arch-pc/lab06  
[vaselivanov@fedora lab06]$
```

Рис. 4.1: Создание каталога

Создаю файл `lab6-1.asm`, используя команду `touch` (рис. 4.2).

```
[vaselivanov@fedora lab06]$ touch lab6-1.asm  
[vaselivanov@fedora lab06]$ ls  
lab6-1.asm
```

Рис. 4.2: Создание файла

Копирую в созданный каталог файл `in_out.asm`, потому что он будет использоваться и в других программах (рис. 4.3).



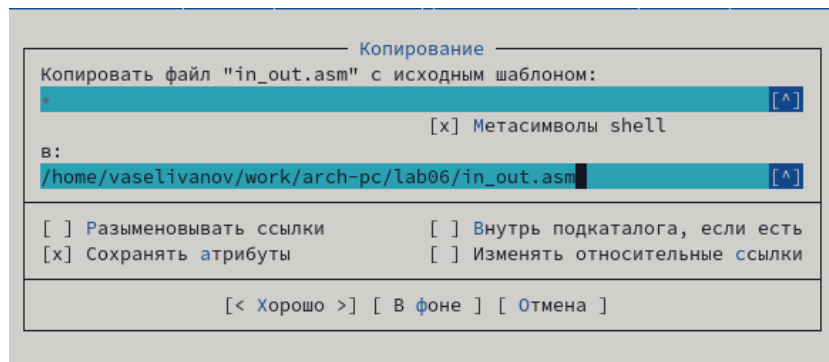


Рис. 4.3: Копирование файла

Открываю файл lab6-1.asm в nano и вставляю в него программу ввода значения регистра eax (рис. 4.4).

```
GNU nano 7.2 /home/vaselivanov/work/arch-pc/lab06/lab6-1.asm
#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 sprintf
call quit
```

Рис. 4.4: Редактирование файла

```
%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

```

Создаю объектный файл и после его компоновки запускаю программу (рис. 4.5). Программа выводит символ “j”, потому что программа вывела символ, который соответствует в системе ASCII сумме двоичных символов 4 и 6.

```

[vaselivanov@fedora lab06]$ nasm -f elf lab6-1.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[vaselivanov@fedora lab06]$ ./lab6-1
j

```

Рис. 4.5: Запуск программы

Теперь изменяю в тексте программы символы ‘6’ и ‘4’ на цифры 6 и 4 (рис. 4.6).

```

GNU nano 7.2 /home/vaselivanov/work/arch-pc/lab06/lab6-1.asm
#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

```

Рис. 4.6: Редактирование программы

```

#include 'in_out.asm'
SECTION .bss
buf1: RESB 80

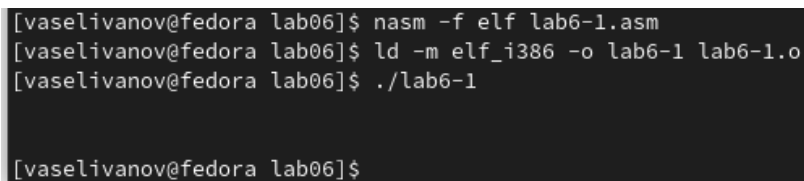
```

```

SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
mov eax,buf1
call sprintf
call quit

```

Создаю новый исполняемый файл и запускаю программу (рис. 4.7). Теперь у меня выводится символ с кодом 10, это символ перевода строки, он не отображается при выводе на экран.



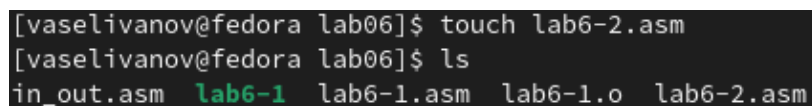
```

[vaselivanov@fedora lab06]$ nasm -f elf lab6-1.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[vaselivanov@fedora lab06]$ ./lab6-1
[vaselivanov@fedora lab06]$

```

Рис. 4.7: Запуск исполняемого файла

Создаю новый файл под названием lab6-2.asm и проверяю его наличие, используя ls (рис. 4.8).



```

[vaselivanov@fedora lab06]$ touch lab6-2.asm
[vaselivanov@fedora lab06]$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm

```

Рис. 4.8: Создание нового файла

Ввожу в файл текст уже другой программы для вывода eax (рис. 4.9).

```
GNU nano 7.2 /home/vaselivanov/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprint
call quit
```

Рис. 4.9: Редактирование программы

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

Создаю и запускаю исполняемый файл lab6-2 (рис. 4.10).Теперь программа выводит 106,потому что программа выводит именно число,не символ, хотя всё еще происходит сложение кодов символов “6” и “4”.

```
[vaselivanov@fedora lab06]$ nasm -f elf lab6-2.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[vaselivanov@fedora lab06]$ ./lab6-2
106
```

Рис. 4.10: Запуск программы

Заменяю в текста данной программы символы ‘4’ и ‘6’ на числа 6 и 4 (рис. 4.11).

```
GNU nano 7.2 /home/vaselivanov/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.11: Редактирование программы

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Снова создаю исполняемый файл программы lab6-2 (рис. 4.12). Теперь программа складывает именно числа, поэтому выводом является сумма 4+6, которая равна 10.

```
[vaselivanov@fedora lab06]$ nasm -f elf lab6-2.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[vaselivanov@fedora lab06]$ ./lab6-2
10
```

Рис. 4.12: Запуск программы

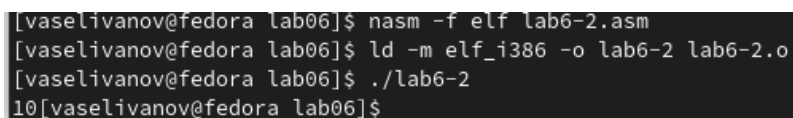
Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.13).



```
add eax, ebx
call iprint
call quit
```

Рис. 4.13: Редактирование программы

Создаю и запускаю исполняемый файл (рис. 4.14). После завершения программы меня не перебрасывает на следующую строку а оставляет на той же.

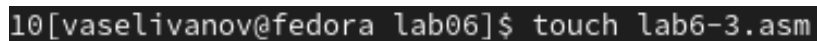


```
[vaselivanov@fedora lab06]$ nasm -f elf lab6-2.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[vaselivanov@fedora lab06]$ ./lab6-2
10[vaselivanov@fedora lab06]$
```

Рис. 4.14: Запуск программы

## 4.2 Выполнение арифметических операций в NASM

Создаю файл, называю его lab6-3.asm, используя команду touch (рис. 4.15).



```
10[vaselivanov@fedora lab06]$ touch lab6-3.asm
```

Рис. 4.15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 \cdot 2 + 3) / 3$  (рис. 4.16).

```

GNU nano 7.2
#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. 4.16: Редактирование программы

Запускаю созданный исполняемый файл (рис. 4.17).

```

[vaselivanov@fedora lab06]$ nasm -f elf lab6-3.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[vaselivanov@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1

```

Рис. 4.17: Запуск программы

Изменяю программу так, чтобы она вычисляла другое выражение  $f(x)=(4*6+2)/5$  (рис. 4.18).

```

GNU nano 7.2 /home/vaselivanov/work/arch-pc/lab06/lab6-3.asm
#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,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
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

```

Рис. 4.18: Редактирование программы

```

#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,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, 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 завершени

```

Создаю исполняемый файл и запускаю программу (рис. 4.19). Программа посчитала выражение абсолютно верно.

```

[vaselivanov@fedora lab06]$ nasm -f elf lab6-3.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[vaselivanov@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 4.19: Запуск программы

Создаю новый файл variant.asm (рис. 4.20).

```

[vaselivanov@fedora lab06]$ touch variant.asm

```

Рис. 4.20: Создание файла

Ввожу в файл текст программы, который вычисляет задания по номеру студенческого билета (рис. 4.21).

```

GNU nano 7.2
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите No 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 ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx,edx
mov ebx,20
div ebx
inc edx
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit

```

Рис. 4.21: Редактирование программы

Создаю исполняемый файл и запускаю программу (рис. 4.22). Ввожу номер своего студенческого. Программа вывела, что мой вариант №12.

```

[vaselivanov@fedora lab06]$ nasm -f elf variant.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[vaselivanov@fedora lab06]$ ./variant
Введите No студенческого билета:
1132236027
Ваш вариант: 8

```

Рис. 4.22: Запуск программы

### 4.2.1 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода

```
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. За вывод на экран результатов вычислений отвечают строки:
4. call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax
5. За вычисления варианта отвечают строки:

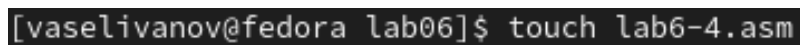
```
хот edx, edx ; обнуление eax для корректной работы div  
mov ebx, 20 ; ebx = 20  
div ebx; eax = eax/20, edx - остаток от деления  
inc edx; edx = edx + 1
```

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
```

### 4.3 Задания для самостоятельной работы

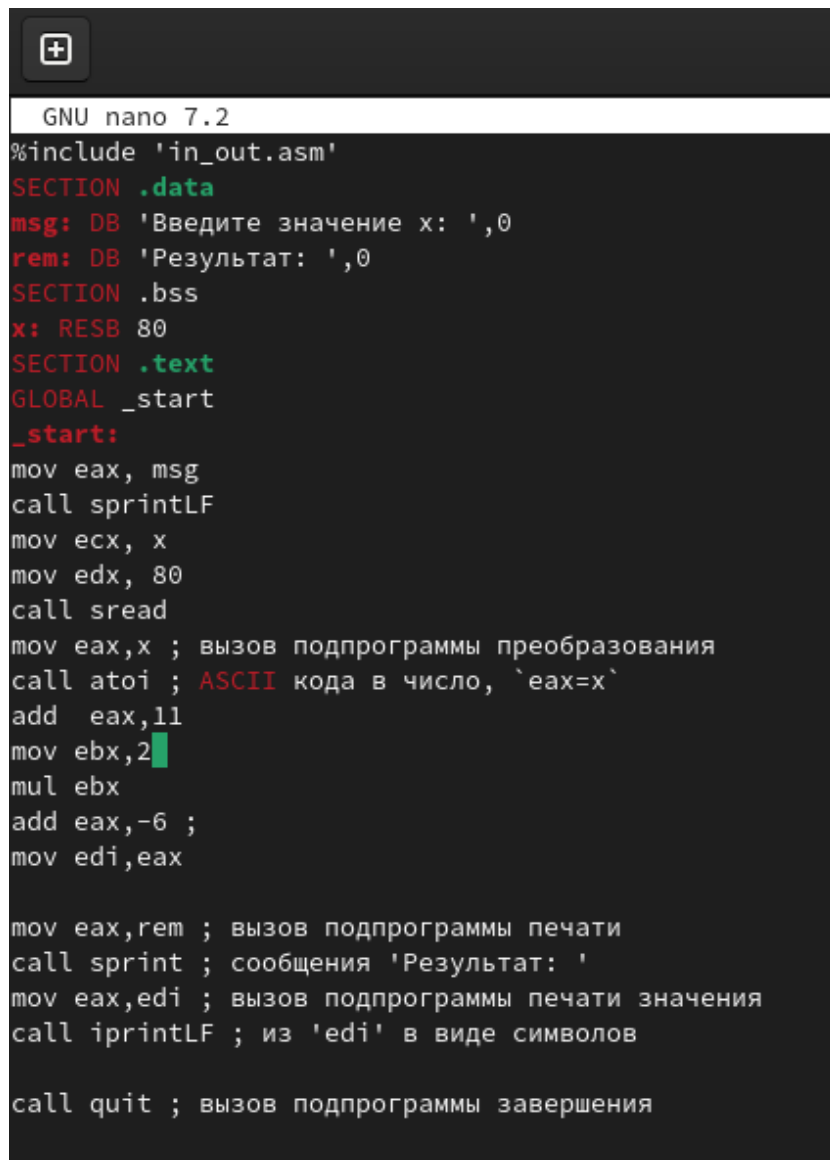
Создаю файл lab6-4.asm,используя команду touch (рис. 4.23).



```
[vaselivanov@fedora lab06]$ touch lab6-4.asm
```

Рис. 4.23: Создание файла

Открываю файл, ввожу в него для вычисления выражения,которое мне выпало (№8)  $(11 + x) * 2 - 6$  (рис. 4.24).



```
GNU nano 7.2
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите значение x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
add eax, 11
mov ebx, 2
mul ebx
add eax, -6 ;
mov edi, eax

mov eax, rem ; вызов подпрограммы печати
call sprintf ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

call quit ; вызов подпрограммы завершения
```

Рис. 4.24: Редактирование программы

```
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите значение x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
```

```

SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
add eax,11
mov ebx,2
mul ebx
add eax,-6 ;
mov edi,eax

mov eax,rem ; вызов подпрограммы печати
call sprintf ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

call quit ; вызов подпрограммы завершения

```

Создаю исполняемый файл и запускаю программу (рис. 4.25). Программа выдает правильный ответ.

```
[vaselivanov@fedora lab06]$ nasm -f elf lab6-4.asm
[vaselivanov@fedora lab06]$ ld -m elf_i386 -o lab6-4 lab6-4.o
[vaselivanov@fedora lab06]$ ./lab6-4
Введите значение x:
2
Результат: 20
[vaselivanov@fedora lab06]$ ./lab6-4
Введите значение x:
1
Результат: 18
```

Рис. 4.25: Запуск программы

## 5 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.



# Список литературы

1. Лабораторная работа №6
2. Таблица ASCII