МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4
по дисциплине «Организация ЭВМ и систем»
Тема: Представление и обработка символьной информации с использованием строковых команд.

Студентка гр. 1303 _____ Куклина Ю.Н.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Разработать на языке Ассемблера программу, обрабатывающую символьную информацию с использованием строковых команд.

Задание.

Разработать программу обработки символьной информации, реализующую функции:

- инициализация (вывод титульной таблички с указанием вида преобразования и автора программы) - на ЯВУ;

- ввода строки символов, длиной не более Nmax (<=80), с клавиатуры в заданную область памяти - на ЯВУ; если длина строки превышает Nmax, остальные символы следует игнорировать;

- выполнение заданного в таблице 5 преобразования исходной строки с записью результата в выходную строку - на Ассемблере;

- вывода результирующей строки символов на экран и ее запись в файл - на ЯВУ.

Ассемблерную часть программы включить в программу на ЯВУ по принципу встраивания (in-line).

Вариант 16:

Преобразование введенных во входной строке русских букв в латинские в соответствие с правилами транслитерации, остальные символы входной строки передаются в выходную строку непосредственно.

Выполнение работы:

Для хранения транслитерируемых букв создан массив типа wchar_t (так как нужно работать с русскими символами), в котором по индексу русского алфавита хранятся соответственно набор латинских символов (буквы ё и щ проверяются отдельно). Создается входная и выходная строки. Размер выходной умножается на 4, так как размер wchar_t 4 байта. Строка

считывается и начинается обработка в блоке asm.

Во входных параметрах входная и выходная строки настраиваются на регистры rsi(«S») и rdi(«D») соответственно. Транслитерируемый же массив помещаем в любой регистр(«r»).

Далее происходит считывание символа, сначала он проверяется на равенство буквам ё и щ, если равно, то отправляем в метку e_lower (или e_upper) или же в shh_lower (shh_upper). Если проверка не пройдена, то символ переходит в метку ru_lower_check, где если код не в диапазоне маленьких русских букв, то он отправляется либо в ru_upper_check (где происходит аналогичная проверка, но для больших русских букв), либо в transliterate_lowecase(uppercase), где происходит транслитерация и запись в выходную строку.

Transliterate_uppercase: Чтобы вычислить индекс в нашем массиве, вычитаем код большой буквы A (1040) и помещаем его в регистр ecx. Логическим сдвигом влево производим умножение на 3, чтобы корректно по нему перемещаться, т.к. размер одного символа в массиве 8 байт (т.к. иногда в записи есть 2 буквы). Затем в eax считываем первый символ из строки массива, вычитаем 32, чтобы получить большую букву, увеличиваем индекс на 4 и считываем второй символ из строки массива, если какой то из них равен 0, то просто переходим к следующему символу. Метка Transliterate_lowercase выполняется аналогично, но без вычитания 32. В метке write_ch записывается сивол с помощь команды stosd. В метке end_process в eax записывается символ конца строки, затем в блоке си выводится готовая строка на экран, программа завершается.

В таблице 1 приведены результаты работы программы.

Таблица 1 – Результаты тестирования

Входные данные	Результат
123 !@# qwe QWE ёйцу ЁЙЦУ	123 !@# qwe qwe eitsu EITsU
Абв Abc 123 YUvvs	Abv Abc 123 YUvvs
Ёжик идет по дороге	Ezhik idet po doroge

Исходный код программы представлен в приложении А.

Вывод.

В результате лабораторной работы была изучена обработка символьной информации с использованием языка ассемблера, а также разработана программа на ЯВУ, использующая вставку на языке ассемблера.

Приложение А

```
#include <locale.h>
#include <stdio.h>
#include <stdlib.h>
#include <wchar.h>

#define N 81

int main() {
  setlocale(LC_CTYPE, "");

  wchar_t *translit = L"a\0" // А
                      L"b\0" // Б
                      L"v\0" // В
                      L"g\0" // Г
                      L"d\0" // Д
                      L"e\0" // Е
                      L"zh"  // Ж
                      L"z\0" // З
                      L"i\0" // И
                      L"i\0" // Й
                      L"k\0" // К
                      L"l\0" // Л
                      L"m\0" // М
                      L"n\0" // Н
                      L"o\0" // О
                      L"p\0" // П
                      L"r\0" // Р
                      L"s\0" // С
                      L"t\0" // Т
                      L"u\0" // У
                      L"f\0" // Ф
                      L"kh"  // Х
                      L"ts"  // Ц
                      L"ch"  // Ч
                      L"sh"  // Ш
                      L"\0\0"// Щ
                      L"ie"  // Ъ
                      L"y\0" // Ы
                      L"\0\0"// Ь
                      L"e\0" // Э
                      L"iu"  // Ю
                      L"ia"; // Я

  wchar_t str[N];
  wchar_t out[N * 4] = {};

  fgetws(str, N, stdin);

  asm("process_str:              \n" // Loop of string processing
      "  xor  rax, rax           \n"
      "  lodsd                  \n"
      "  cmp  eax, 0            \n" // Check if input string ends
      "  je   end_process       \n"

      "  cmp  eax, 1105         \n" // ё
      "  je   e_lower           \n"
```

```
      "  cmp  eax, 1025          \n" // Ё
      "  je   e_upper            \n"
      "  cmp  eax, 1097          \n" // щ
      "  je   chsh_lower         \n"
      "  cmp  eax, 1065          \n" // Щ
      "  je   chsh_upper         \n"

      "ru_lower_check:           \n" // Check if letter is ru and in
lower case
      "  cmp  eax, 1072          \n" // 'а'
      "  jl   ru_upper_check     \n"
      "  cmp  eax, 1103          \n" // 'я'
      "  jg   write_ch           \n" // If false --> just write
letter to outer string
      "  jmp  transliterate_lowercase \n" // If true --> goto
transliteration

      "ru_upper_check:           \n" // Check if letter is ru and in
UPPER case
      "  cmp  eax, 1040          \n" // 'А'
      "  jl   write_ch           \n" // If false --> just write
letter to outer string
      "  cmp  eax, 1071          \n" // 'Я'
      "  jg   write_ch           \n"
      "  jmp  transliterate_uppercase \n" // If true --> goto
transliteration (UPPER)

      "write_ch:                 \n" // Write letter to outer string
      "  stosd                   \n"
      "  jmp  process_str        \n" // Continue loop

      "e_lower:                  \n"
      "  mov  eax, 101           \n"
      "  jmp  write_ch           \n"

      "e_upper:                  \n"
      "  mov  eax, 69            \n"
      "  jmp  write_ch           \n"

      "chsh_lower:               \n"
      "  mov  eax, 115           \n"
      "  stosd                   \n"
      "  mov  eax, 104           \n"
      "  stosd                   \n"
      "  mov  eax, 99            \n"
      "  stosd                   \n"
      "  mov  eax, 104           \n"
      "  stosd                   \n"
      "  jmp  process_str        \n"

      "chsh_upper:               \n"
      "  mov  eax, 83            \n"
      "  stosd                   \n"
      "  mov  eax, 104           \n"
      "  stosd                   \n"
      "  mov  eax, 99            \n"
      "  stosd                   \n"
      "  mov  eax, 104           \n"
```

```
      "  stosd                    \n"
      "  jmp  process_str         \n"

      "transliterate_uppercase:  \n"
      "  sub  eax, 1040           \n" // Find index of letter in ru
alphabet
      "  xor  rcx, rcx            \n"
      "  mov  ecx, eax            \n"
      "  shl  ecx, 3              \n" // Multyply index by 2 plus
size of wchar_t (4)
      "  mov  eax, [%[translit] + rcx]    \n" // Get letter from
'dictionary'
      "  cmp  eax, 0              \n" // Check if letter exists
      "  je   trans_upper_exit    \n" // If not --> goto exit
      "  sub  eax, 32             \n" // Make letter UPPER case
      "  stosd                    \n"
      "  add  rcx, 4              \n"
      "  mov  eax, [%[translit] + rcx]    \n" // Write second char
of transliteration
      "  cmp  eax, 0              \n" // Check if it exists
      "  je   trans_upper_exit    \n" // If not --> goto exit
      "  stosd                    \n"
      "trans_upper_exit:          \n" // Exit of transliteration
      "  jmp  process_str         \n" // Continue loop

      "transliterate_lowercase:  \n"
      "  sub  eax, 1072           \n"
      "  xor  rcx, rcx            \n"
      "  mov  ecx, eax            \n"
      "  shl  ecx, 3              \n"
      "  mov  eax, [%[translit] + rcx]   \n"
      "  cmp  eax, 0              \n"
      "  je   trans_lower_exit    \n"
      "  stosd                    \n"
      "  add  rcx, 4              \n"
      "  mov  eax, [%[translit] + rcx]    \n"
      "  cmp  eax, 0              \n"
      "  je   trans_lower_exit    \n"
      "  stosd                    \n"
      "trans_lower_exit:          \n"
      "  jmp  process_str         \n"

      "end_process:               \n"
      "  mov  eax, 0              \n"
      "  stosd                    \n"
      : // Output parameters
      : [out] "D"(out), [in] "S"(str), [translit] "r"(translit) //
Input parameters
      : "rcx", "rax"); // Clobber list

  wprintf(L"%ls", out);

  return 0;
}
```