**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: «Изучение программирования обработки символьной информации с использованием команд пересылки строк»

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1304 |  | Ярусова Т. В.. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2022

Цель работы.

Изучить принцип символьной обработки информации с использованием команд пересылок строк.

**Задание.**

**Вариант 8 (30) .**

Разработать программу обработки символьной информации, реализующую функции:

- инициализация(вывод титульной таблички с указанием вида преобразования и автора программы) - на языке высокого уровня (Pascal или Си);

- ввода строки символов, длиной не более Nmax (<=80), с клавиатуры в заданную область памяти - на языке высокого уровня;

- выполнение 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 нa Pascal или Си

по принципу встраивания (in-line).

Задание, соответствующее варианту:

Преобразование введенных во входной строке десятичных цифр в двоичную СС, остальные символы входной строки передаются в выходную строку непосредственно.

**Замечания:**

1) В вариантах 7 и 8 следует разделять пробелом двоичные коды, соответствующие преобразуемой цифре;

2) При выполнении преобразования обязательно использовать команды работы со строками;

3) При выполнении преобразования нельзя портить входную строку. Результат преобразования должен записываться в выходную строку.

**Основные теоретические положения.**

Используемые команды работы со строками:

* LODSB - Чтение байта из строки. Команда LODSB копирует один байт из памяти по адресу DS:SI в регистр AL. После выполнения команды, регистр SI увеличивается на 1, если флаг DF = 0, или уменьшается на 1, если DF = 1.  
  Если команда используется в 32-разрядном режиме адресации, то используется регистр ESI.
* STOSW - Запись слова в строку. Команда STOSW сохраняет регистр AX в ячейке памяти по адресу ES:DI. После выполнения команды, регистр DI увеличивается на 2, если флаг DF = 0, или уменьшается на 2, если DF = 1.  
  Если команда используется в 32-разрядном режиме адресации, то используется регистр EDI.
* STOSB -Запись байта в строку. Команда STOSB сохраняет регистр AL в ячейке памяти по адресу ES:DI. После выполнения команды, регистр DI увеличивается на 1, если флаг DF = 0, или уменьшается на 1, если DF = 1.  
  Если команда используется в 32-разрядном режиме адресации, то используется регистр EDI.

Ход работы.

Для реализации поставленной задачи, была написана программа на языке Си с использованием принципа встраивания ассемблерной части(in-line). Программа создана в среде разработки *Visual Studio 2022*.

С помощью функций puts(), производится вывод информации об авторе работы и о типе преобразований.

С помощью *fgets()* в массив символов *input*\_*string* записывается входная строка. Максимальный размер входной строки равен 81 (80 символов строки + ‘\0’). Выходная строка будет иметь размер в среднем 401 символ, так как в худшем варианте будет считано 80 символов, которые будут конвертированы в подстроку размером 4 символа + символы пробела, а также символ конца строки.

После считывания в *input\_string* исходной строки, начинается блок ассемблерного кода (ключевое слово *\_\_asm*). Далее регистру *ES* присваивается значение *DS*, а также в *ESI* присваивается смещение *input\_string*, а в *EDI* смещение *output\_string*, поскольку при использовании команд для работы со строками, чтение происходит из *ES:ESI*, а запись идёт в *ES:EDI*.

push ds

pop es

mov esi, offset input\_string

mov edi, offset output\_string

Строка обрабатывается с метки r*ead*. С помощью *lodsb* считывается символ в *AL*, после чего он проверяется на исключительные случаи, которые обрабатываются отдельно путём перехода на соответствующие им метки. Также производится сравнение регистра BL(это необходимо для того, чтобы сравнивать предыдущий символ и текущий символ на принадлежность к числу для добавления дополнительного пробела).

read :

lodsb

cmp al, 0

je end

cmp al, '9'

ja copy\_symbol

cmp al, '0'

jb copy\_symbol

cmp bl, '0'

jb digit0

cmp bl, '9'

ja digit0

jmp add\_space

Далее организована логика такая, что если текущий символ больше или равен следующего символа, то переходим на следующую метку, указывающую на следующую цифру, иначе записываю цифру текущей метки.

После завершения преобразования строки происходит открытие файла для записи, и в него записывается преобразованная строка.

# Разработанный код см. Приложение А.

# Тестирование.

# 1. Работа программы при введенной строке “1234567890” см.рис 1.

# 

# Рисунок 1 – тест 1

# При вводе всех цифр без пробела вывод совершается корректно с добавлением разделяющих пробелов между двоичным представлением цифр.

# 2. Работа программы при введенной строке “1 2 3 4 5 6 7 8 9 0” см.рис 2.

# 

# Рисунок 2 – тест 2

# При вводе всех цифр с пробелом вывод совершается корректно без добавления дополнительного знака пробела между двоичным представлением цифр.

# 3. Работа программы при введенной строке “Hello, w0rl23d!” см.рис 3.

# 

# Рисунок 3 –тест 3

# При введение цифр и других символов в выходной строке сторонние символы были просто скопированы, а цифры переведены в двоичное представление и разделены дополнительным знаком пробела.

# 4. Работа программы при введенной строке “qwertyuiop123asdfg 456jkklzx 777 999 45 55 zxcvbnmQWEE QWEESSDD6789 87766 11!!!b@bhjfbg” см.рис. 4.

# 

# Рисунок 4 – тест 4

# При введение сторонних символов и цифр программа отработала корректно. Входная строка состоит из более 80 символов. 81 символ – это “@”. В выходную строку данный символ не записался.

**Вывод**

В ходе данной лабораторной работы изучен принцип символьной обработки информации с использованием команд пересылок строк. Написана программа на языке высокого уровня Си с ассемблерными вставками (in-line).

**ПРИЛОЖЕНИЕ А.**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <locale.h>

#include "windows.h"

#include <fstream>

char input\_string[81];

char output\_string[401];

int main() {

setlocale(LC\_ALL, "Russian.1251");

SetConsoleCP(1251);

puts("Студент: Ярусова Татьяна");

puts("Группа: 1304");

puts("Задание:");

puts("Вариант 8.Преобразование введенных во входной строке десятичных цифр в двоичную СС,");

puts("остальные символы входной строки передаются в выходную строку непосредственно.\n");

puts("Введите строку:");

fgets(input\_string, 81, stdin);

input\_string[strlen(input\_string)] = '\0';

\_\_asm {

push ds

pop es

mov esi, offset input\_string //индекс источника

mov edi, offset output\_string //индекс приемника

read :

lodsb //считывание байта из строки

cmp al, 0

je end // если в AL символ конца строки, то завершаем

cmp al, '9'

ja copy\_symbol // если символ больше символа 9, то происходит копирование символа

cmp al, '0'

jb copy\_symbol // если символ меньше символа 0, то происходит копирование копирование символа

cmp bl, '0'

jb digit0

cmp bl, '9'

ja digit0

//если в bl хранится цифра, то добавляем пробел

jmp add\_space

digit0 :

cmp al, '1'

jae digit1

stosb

mov bl, al //записываем в bl предыдущий симвоол

jmp read

digit1 :

cmp al, '2'

jae digit2

stosb

mov bl, al //записываем в bl предыдущий симвоол

jmp read

digit2 :

cmp al, '3'

jae digit3

mov ax, '01'

stosw

mov bl, al //записываем в bl предыдущий симвоол

jmp read

digit3 :

cmp al, '4'

jae digit4

mov ax, '11'

stosw

mov bl, al //записываем в bl предыдущий симвоол

jmp read

digit4 :

cmp al, '5'

jae digit5

mov ax, '01'

stosw

mov al, '0'

jmp digit0

digit5 :

cmp al, '6'

jae digit6

mov ax, '01'

stosw

mov al, '1'

jmp digit1

digit6 :

cmp al, '7'

jae digit7

mov ax, '11'

stosw

mov al, '0'

jmp digit0

digit7 :

cmp al, '8'

jae digit8

mov ax, '11'

stosw

mov al, '1'

jmp digit1

digit8 :

cmp al, '9'

jae digit9

mov ax, '01'

stosw

mov ax, '00'

stosw

mov bl, al //записываем в bl предыдущий симвоол

jmp read

digit9 :

mov ax, '01'

stosw

mov ax, '10'

stosw

mov bl, al //записываем в bl предыдущий симвоол

jmp read

copy\_symbol :

mov bl, al ///записываем в bl предыдущий симвоол

stosb

jmp read

add\_space:

mov bl, al

mov al, ' '

stosb

mov al, bl

jmp digit0

end :

stosb // записываем символ конца строки

};

puts("Преобразованная строка:");

printf("%s\n", output\_string);

FILE\* file;

file = fopen("lb4.txt", "w");

if (file) {

fprintf(file, "%s", output\_string);

fclose(file);

}

else {

puts("Невозможно открыть файл.");

}

return 0;

}