**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 |  | Чернякова В.А. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2022

## Цель работы.

Изучить представление символьной информации с использованием строковых команд. Разработать программу обработки символьной информации на языке Ассемблер и включить в программу на языке высокого уровня – C по принципу встраивания in-line.

## Задание.

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 заданного в таблице 1 преобразования исходной строки с записью результата в выходную строку - на Ассемблере;

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

Ассемблерную часть программы включить в программу нa Pascal или Си по принципу встраивания (in-line).

*Вариант 6*

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

## Ход работы.

Для реализации задания лабораторной работы был написан программный код на языке программирования *C* с использованием принципа встраивания ассемблерной части. Разработка программы производилась в *Visual Studio 2019.*

В начале программы объявляются все библиотеки *C*, необходимые для корректной работы.

Определяются глобальные переменные – два массива символов *char* (*input\_string* для считывания строки и *result* для записи результата). Для первого выделено памяти на 81 символ (по условиям входные данные до 80 символов, и еще один для символа переноса строки *‘\0’*). Для второго – 161 (это максимальное количество символов после преобразования, так как любая десятичная цифра займёт в переводе в восьмеричную систему не более двух символов).

char input\_string[81];

char result[161];

Дальше в *int main()* мы проводим первый пункт лабораторной работы — инициализацию.

В программе и в консоли устанавливается кодировка *1251*. Это связано с тем, что в этой кодировки удобно представлять символы кириллицы — все символы в этой таблице расположены подряд, как строчные, так и заглавные. Также в головной функции создается и открывается файл для записи и выводится текст задания и автор работы.

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

FILE\* f;

f = fopen("Result.txt", "w");

SetConsoleCP(1251);

printf("Студентка - Чернякова Валерия.\nГруппа 1304.\nЗадание - преобразовать введенные во входной строке десятичные цифры в восьмеричную СС, остальные символы входной строки передаются в выходную строку непосредственно.\n");

Далее согласно второму пункту лабораторной с клавиатуры считывается с помощью функции *fgets()* входная строка и заканчивается символом перевода строки.

fgets(input\_string, 81, stdin);

input\_string[strlen(input\_string) - 1] = '\0';

Затем согласно третьему пункту начинается ассемблерная часть. После ключевого слова *\_\_asm* находится блок ассемблерного кода.

Перед непосредственно работой со строкой, регистру *es* присваивается значение регистра *ds*, так как при работе со строками команды используют адреса *ds:esi* и *es:edi* в качестве адресов источника и назначения соответственно. Также регистрам *esi* и *edi* присваиваются значения смещений *input\_string* и *result*.

push ds

pop es

mov esi, offset input\_string

mov edi, offset result

Далее в программе осуществляются следующие действия.

**Reading:** С помощью команды *lodsb* в нижний регистр *ax – al* считывается символ из источника. Код символа сравнивается с кодом символа-цифры *‘8’ (56),* представленной в кодировке, и если символы не равны, то с помощью *jne* совершается прыжок к следующей метке - *CheckNine.*

lodsb

cmp al, 56

jne CheckNine

Если равны, то в *al* записывается пробел до и после добавления в выходную строку представления цифры 8 в 8СС, пробел добавляется в строку с помощью *stosb*. Пробелы необходимы для отделения от других символов. В *ax* по адресу *es:(e)di* (с помощью команды *stows*) записывается два байта - восьмеричная запись цифры *‘8’* в обратном порядке (*‘01’*) и совершается прыжок к метке *Final*.

mov al, ' '

stosb

mov ax, '01'

stosw

mov al, ' '

stosb

jmp Final

**CheckNine:** Происходит аналогичные сравнения и прыжки (как в предыдущей метке), но сравнение производится уже с кодом следующей десятичной цифрой – *‘9’ (57).* Если символы не равны, то с помощью *jne* совершается прыжок к следующей метке - *SaveRecord.*

cmp al, 57

jne SaveRecord

Если равны, то в *al* записывается пробел до и после добавления в выходную строку представления цифры 9 в 8СС, пробел добавляется в строку с помощью *stosb*. Пробелы необходимы для отделения от других символов. В *ax* по адресу *es:(e)di* (с помощью команды *stows*) записывается два байта - восьмеричная запись цифры *‘9’* (‘*11’*) и совершается прыжок к метке *Final*.

mov al, ' '

stosb

mov ax, '11'

stosw

mov al, ' '

stosb

jmp Final

Далее в метках символы не проверяются на десятичные цифры, так как в восьмеричной и десятичной СС отличаются всего две цифры.

Метка **SaveRecord**состоит всего из одной команды *stosb* (команда обратная *lodsb*), которая производит сохранение значения регистра в память. То есть *al* сохраняется по адресу *es:(e)di*. Программа прыгает в эту метку, когда считанный символ строки не нужно заменять.

**Final**: С помощью *cmp* происходит проверка, и если символ *[esi]* не равен символу конца строки *'\0'*, совершается прыжок к метке *Reading* для следующего чтения и последующего записывания символа. Если символы равны, то выполнение ассемблерной части программы завершается.

Затем в соответствии с четвертым пунктом в консоль выводится итоговая строка, а также она записывается в файл, после чего файл закрывается.

printf("Результат работы программы:\n");

printf("%s", result);

fprintf(f, "%s", result);

fclose(f);

Исходный код программы см. в приложении.

## Тестирование.

|  |  |
| --- | --- |
| **Входная строка** | **Выходная строка** |
| апвпвыМФРЫП812872390dhshABS | апвпвыМФРЫП 10 12 10 723 11 0dhshABS |
| 12345677654331232 | 12345677654331232 |
| Восьмеричная система счисления: восемь - 8, девять - 9 | Восьмеричная система счисления: восемь - 10, девять - 11 |
| 8+123459 = ? | 10 +12345 11 = ? |
| 2413123761894623572398 | 2413123761 10 11 46235723 11 10 |

## Вывод.

Изучены основы обработки символьной информации на языке программирования Ассемблер, получены навыки включать Ассемблерную часть в программу на ЯВУ по принципу встраивания (in-line). Разработана программа, заменяющая в строке десятичные цифры на восьмеричные.

**Вопросы для защиты.**

*Внести указанные изменения (ошибки) 1-4 в код, рассматривая их по отдельности. Указать, почему и как они проявятся. Показать тесты, которые эти ошибки обнаружат и их результаты. Если приведенные в отчете тесты ошибку не находят, то внести дополнительный тест.*

# ПРИЛОЖЕНИЕ.

Исходный код программы.

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

#include "windows.h"

char input\_string[81];

char result[161];

int main() {

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

FILE\* f;

f = fopen("Result.txt", "w");

SetConsoleCP(1251);

printf("Студентка - Чернякова Валерия.\nГруппа 1304.\nЗадание - преобразовать введенные во входной строке десятичные цифры в восьме-ричную СС, остальные символы входной строки передаются в выходную строку непосредственно.\n");

fgets(input\_string, 81, stdin);

input\_string[strlen(input\_string) - 1] = '\0';

\_\_asm {

push ds; save in stack ds

pop es; es = ds, stack free 1) Удалить команду

mov esi, offset input\_string; source index - input string

mov edi, offset result; receiver index – result string

Reading :

lodsb; loading a string operand into al, reading a symbol

cmp al, 56; compare with code of number '8'

jne CheckNine; 2) Заменить на jbe CheckNine

mov al, ' '; move space in al

stosb; save al by es : (e)di

mov ax, '01'; move ax ‘01’, 8 in 8 number system

stosw; 3) Заменить на stosb

mov al, ' '; move space in al

stosb; save al by es : (e)di

jmp Final; unconditional transition by label Final

CheckNine :

cmp al, 57; compare with code of number '9'

jne SaveRecord; transition by label SaveRecord, if al code not '9'

mov al, ' '; move space in al

stosb; save al by es : (e)di

mov ax, '11'; помещаем в ax сочетание символов '11', представление числа 9 в 8СС

stosw; save ax by es : (e)di

mov al,' '; move space in al

stosb; save al by es : (e)di

jmp Final; 4) Удалить команду

SaveRecord :

stosb; save al by es : (e)di

Final :

cmp[esi], '\0'; проверку на конец строки лучше перенести в начало

jne Reading; if last symbol esi != ‘\0’, then Reading

stosb; save ‘\0’

}

printf("Результат работы программы:\n");

printf("%s", result);

fprintf(f, "%s", result);

fclose(f);

return 0;

}

## ЗАЩИТА.

1)Удаление команды pop es

Команда *pop* осуществляет извлечение операнда из стека. Благодаря *pop es* осуществляется загрузка слова из вершины стека в регистре *ES*. Перед этой командой прописан *push ds* – помещение сегмента данных в вершину стека. То есть мы заносим данные, но не забираем их. Из-за чего происходит завершение работы программы с некорректным кодом.

ТЕСТ:

**апвпвыМФРЫП812872390dhshABS**

**Ожидаемый результат: апвпвыМФРЫП 10 12 10 723 11 0dhshABS**

**Полученный результат: апвпвыМФРЫП 10 12 10 723 11 0dhshABS**

Программа выдает корректный ответ, но из-за удаление предложенной команды извлечение данных из стека не происходит. Поэтому вместо ожидаемого кода завершения программы – 0, мы получаем -1073741819, что свидетельствует об ошибке нарушения памяти. В стек было помещено значение, но не извлечено.

2)Замена на jbe CheckNine

Команда *jbe* осуществляет переход после команды *cmp op1, op2* если *op1* меньше или равен *op2.*

То есть переход к метке *CheckNine* происходит только в том случае, если код символа по таблице *ASCII* меньше или равен коду числа *8* - *56*.

Таким образом, все символы, чьи коды больше *56* буду представлены в результате как *‘10’*, коды с равным или меньшим значением останутся без изменений.

ТЕСТ:

**8+123459 = ?**

**Ожидаемый результат: 10 +12345 11 = ?**

**Полученный: 8+12345 10 10 10**

Для символов *9, =, ?* – код которых по таблице *ASCII* больше *56* переход по метке *CheckNine* не осуществляется, из-за чего они записываются как представление числа *8* в *8СС* – *‘10’*.

Для остальных же символов переход осуществился, то есть их код меньше или равен *56*. В метке *CheckNine* проверяется, что их код не равен коду числа *9* – *57*, и они записываются без изменений.

3)Замена на stosb

Команда *stosb* осуществляет запись одного байта*.*

Если записываемое значение представлено двумя байтами, то с помощью этой команды запишется только то значение, которое лежит сверху.

ТЕСТЫ:

**Восьмеричная система счисления: восемь - 8, девять – 9**

**Ожидаемый результат: Восьмеричная система счисления: восемь - 10, девять - 11**

**Полученный результат: Восьмеричная система счисления: восемь - 1 , девять - 11**

При выполнении программы, когда она встречает код символа *8 – 56*, в ax помещается *'01'* – двухбайтовое значение. Но из-за того, что *stosb* осуществляет запись только одного байта, который находится сверху ячейки памяти в случае нашего двухбайтного значения, вместо желаемого *'10'* мы получаем *'1'*.

4)Удаление команды jmp Final

Команда *jmp* – команда безусловного перехода. Выполняет переход всегда и в любом случае туда, куда этот самый переход указан.

При удалении команды *jmp Final* автоматический переход из метки *CheckNine* к метке *Final,* отвечающий за проверку символа конца строки и продолжения или завершения работы программы, не происходит.

ТЕСТ:

**2413123761894623572398**

**Ожидаемый результат: 2413123761 10 11 46235723 11 10**

**Полученный результат: 2413123761 10 11 46235723 11 10**

В ходе программы при заходе в метку *CheckNine*, если код символа по таблице *ASCII* равен коду *9 – 57*, происходит запись пробела, представления данного числа в *8СС*, и снова запись пробела (согласно заданию). Из-за удаления *jmp Final* программа автоматически переходит в метку *SaveRecord,* где работает команда *stosb*, то есть записывается еще один пробел – лишний.