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.

Студент гр.1303 _____ Герасименко Я.Д.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучить представление и способы обработки символьной информации с использованием строковых команд.

Задание.

Вариант 4.

4. Преобразование всех заглавных латинских букв входной строки в строчные, а восьмеричных цифр в инверсные, остальные символы входной строки передаются в выходную строку непосредственно.

Основные теоретические положения.

Все команды для работы со строками считают, что строка-источник находится по адресу DS:SI (или DS:ESI), то есть в сегменте памяти, указанном в DS со смещением в SI, а строка-приемник — соответственно в ES:DI (или ES:EDI). Кроме того, все строковые команды работают только с одним элементом строки (байтом, словом или двойным словом) за один раз. Для того, чтобы команда выполнялась над всей строкой, необходим один из префиксов повторения операций.

1. Команды **MOVSB**, **MOVSW** и **MOVSD**.

Копирует один байт (MOVSB), слово (MOVSW) или двойное слово (MOVSD) из памяти по адресу DS:ESI (или DS:SI, в зависимости от разрядности адреса) в память по адресу ES:EDI (или ES:DI).

После выполнения команды регистры ESI (SI) и EDI (DI) увеличиваются на 1, 2 или 4 (если копируются байты, слова или двойные слова), если флаг DF = 0, и уменьшаются, если DF = 1. При использовании с префиксом REP команда MOVS выполняет копирование строки длиной в ECX (или CX) байт, слов или двойных слов.

2. Команды **CMPSP**, **CMPSW** и **CMPSD**.

Сравнивает один байт (CMPSB), слово (CMPSW) или двойное слово (CMPSD) из памяти по адресу DS:ESI (или DS:SI, в зависимости от разрядности адреса) с байтом, словом или двойным словом по адресу ES:EDI (или ES:DI) и устанавливает флаги аналогично команде CMP.

После выполнения команды регистры ESI (SI) и EDI (DI) увеличиваются на 1, 2 или 4 (если сравниваются байты, слова или двойные слова), если флаг DF = 0, и уменьшаются, если DF = 1. При использовании с префиксом REP команда CMPS выполняет сравнение строки длиной в ECX (или CX) байт, слов или двойных слов, но чаще ее используют с префиксами REPNE/REPNZ или REPE/REPZ. В первом случае сравнение продолжается до первого несовпадения в сравниваемых строках, а во втором — до первого совпадения.

3. Команды **SCASB**, **SCASW** и **SCASD**.

Сравнивает содержимое регистра AL (SCASB), AX (SCASW) или EAX (SCASD) с байтом, словом или двойным словом из памяти по адресу ES:EDI (или ES:DI, в зависимости от разрядности адреса) и устанавливает флаги аналогично команде CMP.

После выполнения команды регистр EDI (DI) увеличивается на 1, 2 или 4 (если сканируются байты, слова или двойные слова), если флаг DF = 0, и

уменьшается, если DF = 1. При использовании с префиксом REP команда SCAS выполняет сканирование строки длиной в ECX (или CX) байт, слов или двойных слов, но чаще ее используют с префиксами REPNE/REPNZ или REPE/REPZ. В первом случае сканирование продолжается до первого элемента строки, отличного от содержимого аккумулятора, а во втором — до первого совпадающего.

4. Команды **STOSB**, **STOSW** и **STOSD**.

Копирует регистр AL (STOSB), AX (STOSW) или EAX (STOSD) в память по адресу ES:EDI (или ES:DI, в зависимости от разрядности адреса).

После выполнения команды регистр EDI (DI) увеличивается на 1, 2 или 4 (если копируется байт, слово или двойное слово), если флаг DF = 0, и уменьшается, если DF = 1. При использовании с префиксом REP команда STOS заполнит строку длиной в ECX (или CX) числом, находящимся в аккумуляторе.

5. Команды **LODSB**, **LODSW** и **LODSD**.

Копирует один байт (LODSB), слово (LODSW) или двойное слово (LODSD) из памяти по адресу DS:ESI (или DS:SI, в зависимости от разрядности адреса) в регистр AL, AX или EAX соответственно.

При использовании с префиксом REP команда LODS выполнит копирование строки длиной в ECX (или CX), что приведет к тому, что в аккумуляторе окажется последний элемент строки. На самом деле эту команду используют без префиксов, часто внутри цикла в паре с командой STOS, так что LODS считывает число, другие команды выполняют над ним какие-нибудь действия, а затем STOS записывает измененное число в то же место в памяти.

Выполнение работы.

Результаты тестирования программы lab4.cpp представлены в табл. 1.

Таблица 1 – Тестирование программы lab4.cpp.

№ п/п	Входные данные	Вывод	Результат
1.	ABCDFG7 93 4 5 0 9 -4	abcdfg0 94 3 2 7 9 -3	Программа работает верно
2.	kJШШ3Д9	kjШШ3Д9	Программа работает верно

Выводы.

Изучены представление и способы обработки символьной информации с использованием строковых команд. Написана программа на языке c++ с ассемблеровской in-line вставкой.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД

ПРОГРАММЫ

Название файла: lab4.cpp

```
#include "pch.h"

#include <iostream>

const int len = 81;
char input[len];
char output[len];

int main()
{
   system("chcp 1251 > nul");
   setlocale(LC_CTYPE, "rus");
   wprintf(L"Преобразование всех заглавных латинских букв в строчные,а восьмеричных цифр в инверсные.\n");
   wprintf(L"Выполнил: Герасименко Ярослав\n");

   std::cin.getline(input, len);

   __asm {
      mov esi, offset input; адрес исходной строки
      mov edi, offset output; адрес выходной строки



   Check:
      lodsb //считывание
         cmp al, '\0'; проверка окончания строки
         je END

         cmp al, '0'; если код символа меньше кода 0, то результат записывается в выходной массив
         jl Rec //
         cmp al, '7'; если код символа больше 7, необходима проверка на заглавные латинские буквы
         jg letter
         mov ah, 103
         sub ah, al
         mov al, ah
         jmp Rec


         letter :
      cmp al, 'A'; Проверка на
         jl Rec
         cmp al, 'Z'; заглавную латинскую буквы
         jg Rec
         xor al, 20h; преобразование к строчной производитсяxor с 20h

         Rec :
      stosb; запись в выходную строку
         jmp Check; переходим к следующему символу

         END :
   }
```

```
    std::cout << output << std::endl;

    return 0;
}
```