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. 9382 _____ Балаева М.О.

Преподаватель _____ Ефремов М.А

Санкт-Петербург

2020

Цель работы.

Изучить особенности работы с строками на языке ассемблера.

Задание.

Разработать программу обработки символьной информации, реализующую функции:

1. инициализация (вывод титульной таблички с указанием вида преобразования и автора программы) - на языке высокого уровня (Pascal или Си);
2. ввода строки символов, длиной не более Nmax (<=80), с клавиатуры в заданную область памяти - на языке высокого уровня;
3. выполнение заданного в таблице 1 преобразования исходной строки с записью результата в выходную строку - на Ассемблере;
4. вывода результирующей строки символов на экран и ее запись в файл - на ЯВУ.
5. Ассемблерную часть программы включить в программу на Pascal или Си по принципу встраивания (in-line).

Вариант 10.

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

Теоретические положения.

Для работы со строками, или цепочками символов или чисел (т.е. попросту говоря, с массивами произвольных данных) на языке ассемблера предусмотрен ряд специальных команд:

movs — пересылка строки;
cmps — сравнение двух строк;

seas — поиск в строке заданного элемента;

lods — загрузка аккумулятора (регистров AL или AX) из строки;

stos — запись элемента строки из аккумулятора (регистров AX или AL).

Хотя команды обработки строк, как правило, включаются в программу без явного указания операндов, однако каждая команда, в действительности, использует два операнда. Для команд seas и stos операндом-источником служит аккумулятор, а операнд-приемник находится в памяти. Для команды lods, наоборот, операнд-источник находится в памяти, а приемником служит аккумулятор. Наконец, для команд movs и cmps оба операнда, и источник, и приемник, находятся в памяти.

Все рассматриваемые команды, выполняя различные действия, подчиняются одинаковым правилам, перечисленным ниже. Операнды, находящиеся в памяти, всегда адресуются единообразно: операнд-источник через регистры DS:SI, а операнд-приемник через регистры ES:DI. При однократном выполнении команды обрабатывают только один элемент, а для обработки строки команды должны предваряться одним из префиксов повторения. В процессе обработки строки регистры SI и DI автоматически смещаются по строке вперед (если флаг DF = 0) или назад (если флаг DF = 1), обеспечивая адресацию последующих элементов. Каждая команда имеет модификации для работы с байтами или словами (например, movsb и movsw).

Таким образом, для правильного выполнения команд обработки строк необходимо (в общем случае) предварительно настроить регистры DS:SI и ES:DI, установить или сбросить флаг DF, занести в CX длину обрабатываемой строки, а для команд seas и stos еще поместить операнд-источник в регистр AX (или AL при работе с байтами).

Однако сама операция, после всей этой настройки, осуществляется одной командой, которая обычно даже не содержит операндов, хотя может иметь префикс повторения.

Стоит подчеркнуть, что строки, обрабатываемые рассматриваемыми командами, могут находиться в любом месте памяти: в полях данных

программы, в системных областях данных, в ПЗУ, в видеобуфере. Например, с помощью команды movs можно скопировать массив данных из одной массивной переменной в другую, а можно переслать страницу текста на экран терминала.

Ход работы.

Таблица 1 – Результаты тестирования программы

Входная строка	Выходная строка
aa7a5b	aa111a101b
7abc9	111abc1001
7b9a8	111b1001a1000
пустая строка	*пустая строка*

Вывод.

В ходе данной работы были изучены основы работы со строками на языке ассемблера, использован метод ассемблерной вставки в программе преобразования десятичных цифр исходной строки в двоичную систему счисления.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <stdio.h>
#include <string.h>
#define NMAX 80

void transform(char* str_in, char* str_out){
     __asm__ __volatile__(".intel_syntax noprefix\n\t"
          "transf:\n\t"
               "cld\n\t"
               "mov al, [esi]\n\t"
               "cmp al, '0'\n\t"
               "jl loop_end\n\t"
               "cmp al, '9'\n\t"
               "jg loop_end\n\t"

               "sub al, '0'\n\t"
               "xor cx, cx\n\t" //stack index
               "to_bin:\n\t"
                    "mov ah, al\n\t"
                    "and ah, 1\n\t"
                    "add ah, '0'\n\t"
                    "push ax\n\t"
                    "inc cx\n\t"
                    "shr al, 1\n\t"
                    "cmp al, 0\n\t"
                    "jne to_bin\n\t"

               "write_bits:"
                    "pop ax\n\t"
                    "xchg ah, al\n\t"
                    "stosb\n\t"
                    "dec cx\n\t"
                    "cmp cx, 0\n\t"
                    "jne write_bits\n\t"
               "inc esi\n\t"
               "jmp transf\n\t"

               "loop_end:\n\t"
                    "movsb\n\t"
                    "cmp al, 0\n\t"
                    "jne transf\n\t"
          :
```

```
          :"S" (str_in), "D" (str_out)
          :"%ax", "%cx", "%esp"
     );
}

int main(){
     printf("Type of transformation: decimal digits into
binary.\n");
     printf("Made by Balaeva M.\n");
     char str_in[NMAX];
     fgets(str_in, NMAX, stdin);
     *strstr(str_in, "\n") = 0;
     char str_out[NMAX * 4];
     transform(str_in, str_out);
     printf("%s\n", str_out);
     return 0;
}
```