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. 0382 _____ Самулевич В.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Рассмотреть представление и обработку строк в ассемблере, а также изучить его способы связи с языками высокого уровня.

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 19:

Заменить введенные во входной строке латинские буквы на десятичные числа, соответствующие их номеру по алфавиту, остальные символы входной строки передать в выходную строку непосредственно

Выполнение работы.

Программа состоит из двух файлов: main.cpp и copy.s. В части, написанной на C++, происходит считывание символов с клавиатуры, запись их в массив входных данных (char input[100]), создание массива для хранения результата (char output[100]), вызов ассемблерной процедуры, которая осуществляет преобразование входной строки и вывод результата на экран. Ассемблерная часть

программы представляет из себя определение функции void copy (char* input, char* output), которая используется в .cpp файле. Рассмотрим его подробнее.

Первой строчкой в определении является директива .intel_syntax noprefix, которая позволяет использовать при написании кода синтаксис masm. Далее идет пара команд, загружающая rbp в стек и перемещающая в rbp значение из rsp.Это нужно для восстановления стека и значений управляющих им регистров после выхода из процедуры (для этих же целей перед ret располагается команда pop rbp). После этого идет метка start, которая обозначает точку входа в главный цикл.

Описание главного цикла:

1) Очистись регистр ax

2) Загрузить в al один байт из входной строки.

   Загрузка происходит с помощью команды lodsb.Может возникнуть вопрос: где же для этой команды были загружены соответствующие регистры? Дело в том, что .intel_syntax noprefix хоть и позволяет использовать синтаксис masm,но не меняет при этом основные структурные правила AT&T. Одним из таких правил является отсутствие сегментных регистров, из-за чего источник и приемник адресуется только по esi и edi.А эти регистры используются g++ для передачи параметров в функции, поэтому в них уже изначально загружены адреса входной и выходной строки.

3) Проверить, является ли загруженный байт латинским символом.

   Во входной строке символы закодированы с помощью UTF-8.В этой кодировке латинские символы лежат в диапазонах [0041-005A] (заглавные) и [0061-007A](строчные). Проверка осуществляется с помощью 4 команд cmp(по одной для каждого граничного значения).Если полученный байт оказался английской строчной буквой, он передается дальше без изменений ,а если заглавной, то он приводится к строчному формату(с помощью команды sub

al,0x20).Также считанный байт может не принадлежать английскому алфавиту, или быть равным 0.В первом случае он без изменений переносится в выходную строку(с помощью команды stosb), а во втором происходит завершение главного цикла(нулевой символ означает в C++ конец строки).

4) Заменить символ на его номер по алфавиту.

Этот шаг выполняется, только если был считан латинский символ (В других случаях, с помощью команд из семейства jmp, происходит переход на новую итерацию цикла).

Сначала, с помощью команды, sub al,0x41, в al загружается этот номер в двоичной системе счисления. Затем это значение делится на 10 и частное (которое будет равно двоичному представлению значения разряда десятков) передается в процедуру transform, которая записывает в output код соответствующей полученному значению цифры (0 не записывается). Далее тоже самое повторяется для остатка от деления на 10(он будет равен двоичному представлению значения разряда единиц).

После шага 4) происходит переход на шаг 1).

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	ммfыыbblshи1	мм6ыы22л198и1	Ответ корректен
2.	zxdfаии23!_+g	262446аии23!_+7	Ответ корректен
3.	апроссмсм12f	апроссмсм126	Ответ корректен

Выводы.

Были изучены строки и способы их обработки в ассемблере, а также написана программа в которой использовались механизмы связи этого языка с языками более высокого уровня.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: copy.s

```
.text
    .globl copy
    .type  copy, @function
    .type  transform, @function


copy:
    .intel_syntax noprefix
    push rbp
    mov rbp,rsp
start:
    mov ax,0
    lodsb
    cmp al,0
    je end
    cmp al,0x41
    jb continue
    cmp al,0x5A
    jbe edit
    cmp al,0x61
    jb continue
    cmp al,0x7A
    ja continue
    sub al,0x20
edit:
    sub al,0x40
    mov dh,0xA
    idiv dh
    cmp al,0
    je units
    call transform
units:
    mov al,ah
    call transform
    jmp start
continue:
    stosb
    jmp start
end:
    stosb
stop:
    pop rbp
    ret

transform:
    add al,0x30
    stosb
    ret
```

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
using namespace std;
extern "C" void copy(char* output,char* input);



int main(){
    char input[100];
    char output[100];
    cout<<"Самулевич Василий гр. 0382\nВариант 19\nЗаменить введенные во
входной строке латинские буквы на десятичные числа,соответствующие их номеру
по алфавиту,остальные символы входной строки передать в выходную строку
непосредственно.\n";
    cin >> input;
    copy(output,input);
    std::ofstream file;
    file.open("./result.txt");
    file <<output;
    file.close();
    cout<<output;

}
```