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 _____ Кривенцова Л.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

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).

Задание.

Вариант 9.

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

Теоретические положения.

Команды LODS, LODSB, LODSW, LODSD загружают байт, слово или двойное слово из памяти по определенному адресу в регистр AL, AX или EAX соответственно. Могут использоваться для копирования какого-то участка памяти в другое место с выполнением над данными каких-либо действий. Используются в паре с командами STOS, STOSB, STOSW, STOSD, которые выполняют обратные действия по сохранению значения регистра в память.

Команда LODSB:

- Синтаксис: LODSB
- Операнды: Нет
- Назначение: Чтение байта из строки
- Процессор: 8086+
- Флаги: Не изменяются
- Комментарий: 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.
- Ограничения: Нет
- Примеры:

  ```
  mov   si,offset str1
  cld
  lodsb
  ```

Регистры общего назначения

К регистрам общего назначения относится группа из 8 регистров, которые можно использовать в программе на языке ассемблера. Все регистры имеют размер 32 бита и могут быть разделены на 2 или более частей.

Как видно из рисунка, регистры ESI, EDI, ESP и EBP позволяют обращаться к младшим 16 битам по именам SI, DI, SP и BP соответственно, а регистры EAX, EBX, ECX и EDX позволяют обращаться как к младшим 16 битам (по именам AX, BX, CX и DX), так и к двум младшим байтам по отдельности (по именам AH/AL, BH/BL, CH/CL и DH/DL).

Названия регистров происходят от их назначения:

EAX/AX/AH/AL (accumulator register) – аккумулятор;

EBX/BX/BH/BL (base register) –регистр базы;

ECX/CX/CH/CL (counter register) – счётчик;

EDX/DX/DH/DL (data register) – регистр данных;

ESI/SI (source index register) – индекс источника;

EDI/DI (destination index register) – индекс приёмника (получателя);

ESP/SP (stack pointer register) – регистр указателя стека;

EBP/BP (base pointer register) – регистр указателя базы кадра стека.

Ещё один нюанс состоит в использовании регистров в качестве базы, т.е. хранилища адреса оперативной памяти. В качестве регистров базы можно использовать любые регистры, но желательно использовать регистры EBX, ESI, EDI или EBP. В этом случае размер машинной команды обычно бывает меньше.

Выполнение работы.

Написание программы производилось на языках C++ и Ассемблер в IDE CLion с использованием компилятора Microsoft Visual Studio.

Определяются глобальные переменные – два массива символов (*input_line* для считывания строки и *result* для записи результата). Для первого выделено памяти на 81 символ (по условиям входные данные до 80ти символов, и ещё один нужен для символа завершения строки ‘\0’), для второго – 161 (это максимальное количество символов после преобразования, т.к. любая десятичная цифра займёт в переводе в восьмеричную систему не более двух символов). В главной функции *main* в первый массив из стандартного потока ввода *stdin* с помощью *fgets* считывается строка.

Далее начинается ассемблерная часть программы. Перед непосредственно работы со строкой, регистру *ES* присваивается значение регистра *DS*, ведь 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_line* и *result*.

Далее организован цикл, включающий в себя следующие метки:

readout: С помощью команды *LODSB* в нижний регистр *AX – AL* считывается символ из источника. Символ сравнивается с цифрой ‘8’, и если символы не равны, то с помощью *JNE* совершается прыжок к следующей метке.

Если равны, то в *AX* (с помощью команды *STOWS*) записывается два байта - восьмеричная запись цифры ‘8’ в обратном порядке (‘01’) и совершается прыжок к метке *ongoing*.

check: Происходит аналогичные сравнение и прыжки (как в предыдущей метке), но сравнение производится уже со следующей десятичной цифрой – ‘9’. Далее в метках символы не проверяются на десятичные цифры, т.к. в восьмеричной и десятичной СС отличаются всего две цифры.

Метка *save_record* состоит всего из одной команды *STOSB* (команда обратная *LODSB*), которая производит сохранение значения регистра в память. Программа прыгает в эту метку, когда считанный символ строки не нужно заменять.

ongoing: С помощью *CMP* происходит проверка, и если символ не равен символу конца строки, совершается прыжок к метке *readout* для следующей записи символа. Если символы равны, то выполнение ассемблерной части программы завершается.

После окончания цикла ассемблерных команд преобразованная строка *result* выводится на экран и в указанный файл.

Исходный код программы см. в приложении А.

Тестирование.

Таблица 1. Результат тестирования.

№ т.	Входные данные	Результат	Комментарий
1	1h2h3h4h5h6h 7h8h9h10	1h2h3h4h5h6h 7h10h11h10	Программа работает верно
2	Цифра 8 в восьмеричной СС выглядит как десять.	Цифра 10 в восьмеричной СС выглядит как десять.	Программа работает верно

3	Только цифры 8 and 9 изменяются.	Только цифры 10 and 11 изменяются.	Программа работает верно
4	Numbers 1-7 in CC совпадают	Numbers 1-7 in CC совпадают	Программа работает верно

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.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММ

Файл lb4.cpp

```
#include <iostream>
#include <fstream>
#include <cstdio>

char input_line[81];
char result[161];

int main(){
    std::cout << "Krivencova Lyubov 0382. Var 9. \nConversion of decimal digits entered in the input string to the octal number system.\n";
    fgets(input_line, 81, stdin);
    input_line[strlen(input_line) - 1] = '\0';

    __asm {
    push ds
    pop es
    mov esi, offset input_line
    mov edi, offset result
    readout:
        lodsb
        cmp al, '8'
        jne check
        mov ax, '01'
        stosw
        jmp ongoing

    check:
        cmp al, '9'
        jne save_record
        mov ax, '11'
        stosw
        jmp ongoing

    save_record:
        stosb
```

```
    ongoing:
        cmp [esi], '\0'
        jne readout
    }

    std::cout << result << std::endl;
    std::ofstream outfile("asm_result.txt");
    outfile << result;
    return 0;
}
```