

**МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ**

ОТЧЁТ

по лабораторной работе №4

по дисциплине «Организация ЭВМ и систем»

**Тема: Представление и обработка символьной информации с
использованием строковых команд.**

Студент гр. 1383

Ковалев П. А.

Преподаватель

Ефремов М. А.

Санкт-Петербург
2022

Цель работы

Изучение программирования обработки символьной информации с использованием команд пересылки строк.

Задание

Вариант 2.

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 N_{\max} (≤ 80), с клавиатуры в заданную область памяти - на ЯВУ; если длина строки превышает N_{\max} , 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).

Выполнение работы

Пользовательский ввод реализован на языке C (компилятор GCC). Для включения в ассемблере компилятора эмуляции синтаксиса Intel выставлены флаги компилятора.

Для упрощения тестирования созданы функции C (`solveASM`) для ассемблерной вставки и (`checkASMWithC`), реализующая тесты.

В функции `solveASM` выделяется память под строку-результат, далее выполняется ассемблерная вставка. В ней с помощью инструкций `lodsrb` и

stosb символы строки считываются в регистр al, затем происходит проверка al на 0, если очередной символ равен \0, то цикл прерывается. Для уменьшения количества проверок в цикле al копируется в bl. Далее символ в bl меняется на прописной вариант (or bl, 0x20), и далее сравнивается bl.

По завершению вставки программа передает результирующую строку функции main, где строка записывается в файл.

В таблице (1) представлен протокол тестирования программы.

Таблица 1: Результаты тестирования

№	Исходные данные	Вывод программы	Комментарий
1.	test1	test1	Ответ корректен
2.	TEST1	TEST1	Ответ корректен
3.	\$test2	test2	Ответ корректен
4.	##te#&#st3 ^{^^}	test3	Ответ корректен
5.	\$test*4*9	test49	Ответ корректен
6.	t0estПРЛИ	t0est	Ответ корректен
7.	test((478%%%	test478	Ответ корректен
8.	THis is test string000—)))	THisisteststring000	Ответ корректен

Вывод

При выполнении программы были получены результаты, совпадающие с требованиями условия задачи. В ходе выполнения данной лабораторной работы был изучен способ обработки строк в языке assembler.

Приложение А

Исходный код программы

Название файла: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 2. Формирование выходной строки только из цифр
// и латинских букв входной строки.

char *solveASM(char *str);

void checkAsmWithC();

int main(void) {
    checkAsmWithC();

    char userInput[81];
    printfКовалев(" Павел, вариант 2; Введите строку: ");
    fgets(userInput, 80, stdin);
    userInput[80] = '\0';

    char *res = solveASM(userInput);

    FILE *f = fopen("result.txt", "w");
    fprintf(f, "%s\n", res);
    fclose(f);

    free(res);

    return EXIT_SUCCESS;
}

char *solveASM(char *str) {
    char *res = calloc(81, sizeof(char));
    asm volatile("    cld                                \n\t"
                 "    mov rsi, %[str]                  \n\t"
                 "    mov rdi, %[res]                  \n\t"
                 "    mov rcx, 80                       \n\t"
                 "mainLoop:                             \n\t"
```

```

        lodsbl                                "\n\t"
        cmp al, 0                             "\n\t"
        jne check                             "\n\t"
        mov rcx, 1                             "\n\t"
        jmp write                             "\n\t"
    "check:                                   "\n\t"
        cmp al, '0'                             "\n\t"
        jnl mainLoop                           "\n\t"
        cmp al, '9'                             "\n\t"
        jle write                             "\n\t"
        mov bl, al                             "\n\t"
        or bl, 0x20                             "\n\t"
        cmp bl, 'a'                             "\n\t"
        jnl mainLoop                           "\n\t"
        cmp bl, 'z'                             "\n\t"
        jg mainLoop                             "\n\t"
    "write:                                   "\n\t"
        stosb                                  "\n\t"
        loop mainLoop                           "\n\t"

    ::[str] "r"(str),
    [res] "r"(res)
    : "rax", "rbx", "rcx", "rdi", "rsi", "memory", "cc
");
return res;
}

void checkAsmWithC() {
    struct test {
        char *str;
        char *res;
    } tests[] = {
        {"test1", "test1"},
        {"TEST1", "TEST1"},
        {"$test2", "test2"},
        {"##te##st3^^^", "test3"},
        {"$test*4*9", "test49"},
        {"ПРЖМт0est", "t0est"},
        {"test((478%%", "test478"},
        {"THis is test string000---))", "THisisteststring000"},
    };
};

```

```

const size_t testCount = sizeof(tests) / sizeof(struct test);
for (size_t i = 0; i < testCount; ++i) {
    char *res = solveASM(tests[i].str);

    if (strncmp(res, tests[i].res, 80)) {
        printf("Failed test: solveASM(\"%s\"). Got = \"%s\";
expected = \"%s\"\\n",
            tests[i].str, res, tests[i].res);

        free(res);

        exit(EXIT_FAILURE);
    }

    free(res);
}
}

```

Название файла: Makefile

```

CC = gcc
CFLAGS ?= -Wall -Wextra -ggdb -masm=intel

all: main

main: main.o
    $(CC) $(CFLAGS) -o $@ $<

main.s: main.c
    $(CC) $(CFLAGS) -S -fverbose-asm -o $@ $<

%.o: %.c
    $(CC) $(CFLAGS) -c -o $@ $<

clean:
    rm -f *.o main

.PHONY: all

```