Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 4

тема «Знакомство с lc-3 «Языки программирования и методы трансляции»

Выполнил: студент группы ИСТ-22-1Б Синьковский Г.С.

Проверил: Батин С.Е.

Пермь, 2022

Содержимое

[Упражнение 1 3](#_Toc154792939)

[1.1. Постановка задачи 3](#_Toc154792940)

[1.2. Код программы 3](#_Toc154792941)

[1.3 Пример работы программы 3](#_Toc154792942)

[Упражнение 2 4](#_Toc154792943)

[2.1. Постановка задачи 4](#_Toc154792944)

[2.3. Примеры работы программы 4](#_Toc154792945)

[Упражнение 3 6](#_Toc154792946)

[3.1. Постановка задачи 6](#_Toc154792947)

[3.2 Код программы 6](#_Toc154792948)

[3.3. Пример работы программы 7](#_Toc154792949)

[Упражнение 4 9](#_Toc154792950)

[4.1. Постановка задачи 9](#_Toc154792951)

[4.3. Пример работы программы 11](#_Toc154792952)

[Упражнение 5 12](#_Toc154792953)

[5.1. Постановка задачи 12](#_Toc154792954)

[5.4. Пример работы программы 14](#_Toc154792955)

# Упражнение 1

## 1.1. Постановка задачи

Написать и запустить программу, выводящую строку Hello World

## 1.2. Код программы

.ORIG x3000

LEA R0, HELLO\_STR ; загрузка адреса строки в R0

PUTS ; вывод строки

HALT ; остановка программы

HELLO\_STR .STRINGZ "Hello, World!"

.END

## 1.3 Пример работы программы

На рис. 1 продемонстрирован результат работы программы 1 упражнения.



Рисунок 1. Результат программы 1 упражнения

# Упражнение 2

## 2.1. Постановка задачи

Написать и запустить программу, принимающую строку пользователя и выводящую ее в терминал.2.2. Код программы

.ORIG x3000

START LEA R0, INPUT\_MSG ; выводим приглашение к вводу

PUTS

LEA R1, BUFFER ; адрес начала буфера для строки

ADD R2, R1, #0 ; R2 используется для перемещения по буферу

READ GETC ; считываем символ

OUT ; выводим символ в терминал для отображения введенных символов

ADD R4, R0, #-10 ; проверяем, не символ ли это новой строки

BRz PRINT ; если это символ новой строки, выводим строку

STR R0, R2, #0 ; сохраняем считанный символ в буфере

ADD R2, R2, #1 ; переходим к следующему месту в буфере

BRnzp READ ; повторяем чтение следующего символа

PRINT LEA R0, BUFFER ; выводим введенную строку

PUTS

HALT

INPUT\_MSG .STRINGZ "Enter a string: "

BUFFER .BLKW 50 ; буфер для хранения строки

.END

## 2.3. Примеры работы программы

На рис.2 показаны результаты работы программы упражнения 2.

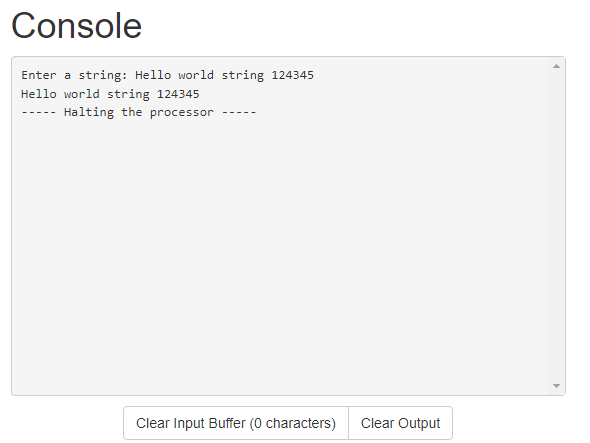


Рисунок 2. Результат программы 2 упражнения

# Упражнение 3

## 3.1. Постановка задачи

Написать и запустить программу, реализующую сортировку заданного пользователем целочисленного массива из 10 целых чисел.

## 3.2 Код программы

.ORIG x3000

; Инициализация

LD R2, INIT

LEA R1, ARRAY

; Ввод данных

INPUT\_LOOP

ADD R2, R2, #1

LEA R0, INPUT\_PROMPT

PUTS

GETC

OUT

ADD R3, R0, #0

LD R4, NEG\_ASCII\_ZERO ; Загрузка -48 в R4

ADD R3, R3, R4 ; Преобразование символа в число

STR R3, R1, #0

ADD R1, R1, #1

LEA R0, NEWLINE

PUTS

ADD R3, R2, #-10

BRz BUBBLE\_SORT

BRnzp INPUT\_LOOP

; Сортировка пузырьком

BUBBLE\_SORT

LD R6, ARRAY\_LEN ; Загрузка длины массива в R6

ADD R4, R6, #0 ; Копируем длину массива в R4

OUTER\_LOOP

ADD R4, R4, #-1 ; Уменьшаем R4 на 1

BRz OUTPUT\_LOOP ; Если R4 = 0, массив отсортирован

LEA R3, ARRAY ; Указатель на начало массива

ADD R5, R4, #0 ; Копируем R4 в R5

INNER\_LOOP

LDR R0, R3, #0 ; Загружаем текущий элемент

LDR R1, R3, #1 ; Загружаем следующий элемент

NOT R2, R1

ADD R2, R2, #1

ADD R2, R0, R2 ; R2 = R0 - R1

BRN SWAP ; Если R0 > R1, меняем местами

SKIP\_SWAP

ADD R3, R3, #1 ; Переходим к следующему элементу

ADD R5, R5, #-1 ; Уменьшаем счетчик

BRp INNER\_LOOP

BRnzp OUTER\_LOOP

SWAP

STR R1, R3, #0 ; Меняем элементы местами

STR R0, R3, #1

BRnzp SKIP\_SWAP

; Вывод данных

OUTPUT\_LOOP

LEA R1, ARRAY ; Перезагрузка адреса начала массива

LD R2, INIT ; Сброс счетчика элементов

PRINT\_LOOP

LDR R0, R1, #0 ; Загрузка элемента массива в R0

LD R4, ASCII\_ZERO

ADD R0, R0, R4 ; Преобразование числа обратно в символ ASCII

OUT ; Вывод символа

LEA R0, NEWLINE

PUTS ; Вывод новой строки

ADD R1, R1, #1 ; Переход к следующему элементу

ADD R2, R2, #1

ADD R3, R2, #-10

BRz END

BRnzp PRINT\_LOOP

; Завершение программы

END

HALT

; Константы и данные

INIT .FILL #0

INPUT\_PROMPT .STRINGZ "Enter a number: "

NEWLINE .STRINGZ "\n"

ASCII\_ZERO .FILL #48

NEG\_ASCII\_ZERO .FILL #-48

ARRAY .BLKW #10

ARRAY\_LEN .FILL #10

.END

## 3.3. Пример работы программы

На рис.3 представлен результат работы программы упражнения 3.

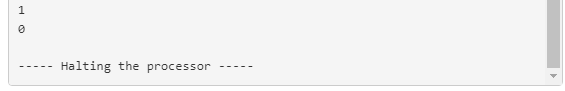


Рисунок 3. Пример работы программы упражнения 3

# Упражнение 4

## 4.1. Постановка задачи

Написать и запустить программу, реализующую сортировку заданной пользователем строки из 10 цифр.4.2 Код программы

.ORIG x3000

; Инициализация

LD R2, INIT

LEA R1, BUFFER

LD R4, ASCII\_ZERO

LD R5, NEG\_ASCII\_ZERO

LEA R0, INPUT\_PROMPT

PUTS

; Ввод данных

READ\_LOOP

GETC

OUT

ADD R3, R0, #-10 ; Проверка на символ новой строки

BRz CONVERT ; Переход к конвертации, если достигнут конец строки

STR R0, R1, #0 ; Сохранение символа в буфер

ADD R1, R1, #1

BRnzp READ\_LOOP

; Конвертация строки в числа

CONVERT

LEA R1, BUFFER ; Адрес начала буфера

LEA R6, ARRAY ; Адрес начала массива чисел

LD R2, INIT ; Сброс счетчика

CONVERT\_LOOP

LDR R0, R1, #0

ADD R0, R0, R5 ; Преобразование символа в число

STR R0, R6, #0

ADD R1, R1, #1

ADD R6, R6, #1

ADD R2, R2, #1

ADD R3, R2, #-10

BRz BUBBLE\_SORT

BRnzp CONVERT\_LOOP

; Сортировка пузырьком

BUBBLE\_SORT

LD R6, ARRAY\_LEN ; Загрузка длины массива в R6

ADD R4, R6, #0 ; Копируем длину массива в R4

OUTER\_LOOP

ADD R4, R4, #-1 ; Уменьшаем R4 на 1

BRz OUTPUT\_LOOP ; Если R4 = 0, массив отсортирован

LEA R3, ARRAY ; Указатель на начало массива

ADD R5, R4, #0 ; Копируем R4 в R5

INNER\_LOOP

LDR R0, R3, #0 ; Загружаем текущий элемент

LDR R1, R3, #1 ; Загружаем следующий элемент

NOT R2, R1

ADD R2, R2, #1

ADD R2, R0, R2 ; R2 = R0 - R1

BRN SWAP ; Если R0 > R1, меняем местами

SKIP\_SWAP

ADD R3, R3, #1 ; Переходим к следующему элементу

ADD R5, R5, #-1 ; Уменьшаем счетчик

BRp INNER\_LOOP

BRnzp OUTER\_LOOP

SWAP

STR R1, R3, #0 ; Меняем элементы местами

STR R0, R3, #1

BRnzp SKIP\_SWAP

; Вывод данных

OUTPUT\_LOOP

LEA R1, ARRAY ; Перезагрузка адреса начала массива

LD R2, INIT ; Сброс счетчика элементов

PRINT\_LOOP

LDR R0, R1, #0 ; Загрузка элемента массива в R0

LD R4, ASCII\_ZERO

ADD R0, R0, R4 ; Преобразование числа обратно в символ ASCII

OUT ; Вывод символа

LEA R0, NEWLINE

PUTS ; Вывод новой строки

ADD R1, R1, #1 ; Переход к следующему элементу

ADD R2, R2, #1

ADD R3, R2, #-10

BRz END

BRnzp PRINT\_LOOP

; Завершение программы

END

HALT

; Константы и данные

INIT .FILL #0

INPUT\_PROMPT .STRINGZ "Enter string: "

NEWLINE .STRINGZ "\n"

ASCII\_ZERO .FILL #48

NEG\_ASCII\_ZERO .FILL #-48

ARRAY .BLKW #10

ARRAY\_LEN .FILL #10

BUFFER .BLKW #10

.END

## 4.3. Пример работы программы

На рис.4 представлен результат работы программы упражнения 4.



Рисунок 4. Пример работы программы упражнения 4

# Упражнение 5

## 5.1. Постановка задачи

Реализовать с использованием flex и bison программу, транслирующую простые инструкции ветвления и цикла в ассемблер lc-3.

%{

#include <stdio.h>

#include <stdlib.h>

extern int yylex();

void yyerror(const char\* s) {

fprintf(stderr, "Ошибка: %s\n", s);

};

%}

%union {

int number;

}

%token VARIABLE SEMICOLON OPEN\_BRACE CLOSE\_BRACE OPEN\_PAREN CLOSE\_PAREN ASSIGN NEQ GT LT SUB ADD IF WHILE FOR PRINT

%token <number> NUM

%type <number> expr

**%%**

prog: stmts

;

stmts: stmt

| stmts stmt

;

stmt: assign\_stmt

| if\_stmt

| while\_stmt

| for\_stmt

| print\_stmt

;

assign\_stmt: VARIABLE ASSIGN expr { printf("LD R1, #%d\n", $3); }

;

expr: NUM { $$ = $1; }

| VARIABLE { $$ = 0; printf("ADD R1, R1, #0\n"); }

| expr SUB NUM { $$ = $1 + $3; printf("ADD R1, R1, #-%d\n", $3); }

| expr ADD NUM { $$ = $1 + $3; printf("ADD R1, R1, #%d\n", $3); }

;

if\_stmt: IF OPEN\_PAREN cond CLOSE\_PAREN OPEN\_BRACE stmts CLOSE\_BRACE

{ printf("BRzp IFEND\n"); }

;

while\_stmt: WHILE OPEN\_PAREN cond CLOSE\_PAREN OPEN\_BRACE stmts CLOSE\_BRACE

{ printf("BRzp WHILEEND\n"); }

;

for\_stmt: FOR OPEN\_PAREN assign\_stmt SEMICOLON cond SEMICOLON expr CLOSE\_PAREN OPEN\_BRACE stmts CLOSE\_BRACE

{ printf("BRzp FOREND\n"); }

;

print\_stmt: PRINT OPEN\_PAREN VARIABLE CLOSE\_PAREN { printf("LD RO, R1, #0\nOUT\n"); }

cond: VARIABLE GT NUM

{ printf("LD R1, X\nLD R2, #%d\nADD R3, R1, R2\nBRp TRUE\n", -$3); }

| VARIABLE ASSIGN NUM

{ printf("LD R1, X\nLD R2, #%d\nNOT R2, R2\nADD R2, R2, #1\nADD R3, R1, R2\nBRz TRUE\n", $3); }

| VARIABLE LT NUM

{ printf("LD R1, X\nLD R2, #%d\nADD R3, R1, R2\nBRn TRUE\n", -$3); }

| VARIABLE NEQ NUM

{ printf("LD R1, X\nLD R2, #%d\nNOT R2, R2\nADD R2, R2, #1\nADD R3, R1, R2\nBRnp TRUE\n", $3); }

;

**%%**

int main() {

yyparse();

return 0;

}

%{

#include "t5.tab.h"

%}

**%%**

"print" { return PRINT; }

"for" { return FOR; }

"if" { return IF; }

"while" { return WHILE; }

";" { return SEMICOLON; }

">" { return GT; }

"-" { return SUB; }

"+" { return ADD; }

"=" { return ASSIGN; }

"<" { return LT; }

"!=" { return NEQ; }

"x" { return VARIABLE; }

"{" { return OPEN\_BRACE; }

"}" { return CLOSE\_BRACE; }

"(" { return OPEN\_PAREN; }

")" { return CLOSE\_PAREN; }

[0-9]+ { yylval.number = atoi(yytext); return NUM; }

[ \t\n]+ { /\* игнорировать пробелы, табуляции и новые строки \*/ }

. { fprintf(stderr, "Нераспознанный символ: %s\n", yytext); }

**%%**

int yywrap() {

return 1;

}

## 5.4. Пример работы программы

На рис.5 представлен результат работы программы упражнения 5.

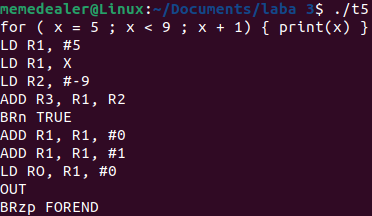


Рисунок 5. Пример работы программы упражнения 5