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 Язов М.М.

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 1

## 1.1. Постановка задачи

Написать и запустить программу, выводящую строку Hello World

## 1.2. Код программы

exc1.asm

.ORIG x3000  
  
 LEA R0, HELLO\_STR ; Загрузка эффективного адреса строки в R0  
 PUTS ; Вывод строки  
 HALT ; Остановка программы  
  
 HELLO\_STR .STRINGZ "Hello, World!" ; Строка "Hello, World!" с нулевым завершением  
  
 .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

exc2.asm

.ORIG x3000  
  
 START  
 LEA R0, INPUTING ; Загрузка эффективного адреса строки INPUTING в R0  
 PUTS ; Вывод строки на консоль  
 LEA R1, BUFFER ; Загрузка эффективного адреса буфера BUFFER в R1  
 ADD R2, R1, #0 ; R2 используется для перемещения по буферу  
  
 READ  
 GETC ; Считывание символа с консоли  
 OUT ; Вывод символа на консоль  
 ADD R4, R0, #-10 ; Проверка, не является ли символ символом новой строки (ASCII-код - 10)  
 BRz PRINT ; Если да, перейти к подпрограмме PRINT  
 STR R0, R2, #0 ; Сохранение считанного символа в буфере на текущей позиции  
 ADD R2, R2, #1 ; Переход к следующей позиции в буфере  
 BRnzp READ ; Безусловный переход обратно к READ для продолжения считывания символов  
  
 PRINT  
 LEA R0, BUFFER ; Загрузка эффективного адреса буфера BUFFER в R0  
 PUTS ; Вывод всей строки, хранящейся в буфере  
 HALT ; Остановка выполнения программы  
  
 INPUTING .STRINGZ "Enter a string: " ; Строка с приглашением для ввода  
 BUFFER .BLKW #80 ; Резервирование блока памяти для буфера в 80 слов  
  
 .END

## 2.3. Примеры работы программы

На рис.2 показаны результаты работы программы упражнения 2.

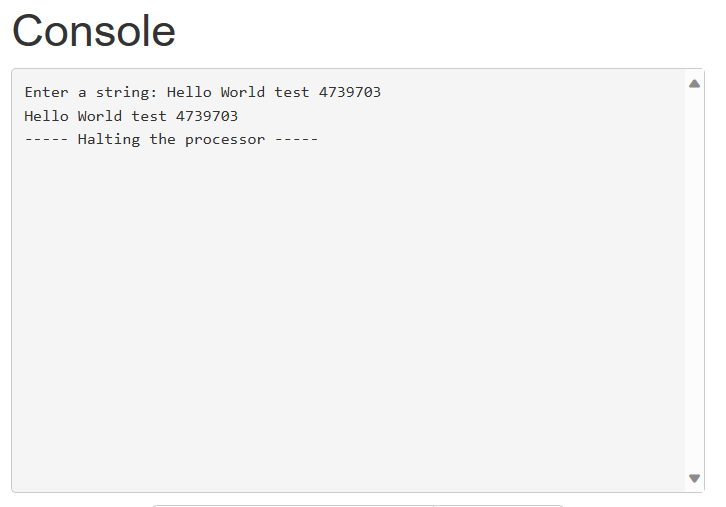


Рисунок 2. Результат программы 2 упражнения

# Упражнение 3

## 3.1. Постановка задачи

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

## 3.2 Код программы

exc3.asm

.ORIG x3000  
  
 ; Инициализация  
 LD R2, INIT ; Загрузка начального значения счетчика в R2  
 LEA R1, ARRAY ; Загрузка адреса начала массива в R1  
  
 ; Ввод данных  
 INPUT\_LOOP  
 ADD R2, R2, #1 ; Увеличение счетчика элементов массива  
 LEA R0, INPUT\_PROMPT ; Загрузка адреса строки приглашения ввода в R0  
 PUTS ; Вывод строки приглашения  
  
 GETC ; Считывание символа с консоли  
 OUT ; Вывод символа на консоль  
  
 ADD R3, R0, #0 ; Копирование символа в R3  
 LD R4, NEG\_ASCII\_ZERO ; Загрузка -48 в R4  
 ADD R3, R3, R4 ; Преобразование символа в число  
 STR R3, R1, #0 ; Сохранение числа в массиве  
 ADD R1, R1, #1 ; Переход к следующему элементу  
  
 LEA R0, NEWLINE ; Загрузка символа новой строки  
 PUTS ; Вывод новой строки  
  
 ADD R3, R2, #-10 ; Проверка на завершение ввода  
 BRz BUBBLE\_SORT ; Если введены все числа, переход к сортировке  
 BRnzp INPUT\_LOOP ; Повторение ввода следующего символа  
  
 ; Сортировка пузырьком  
 BUBBLE\_SORT  
 LD R6, ARRAY\_LEN ; Загрузка длины массива в R6  
 ADD R4, R6, #0 ; Копирование длины массива в R4 (для внешнего цикла)  
  
 OUTER\_LOOP  
 ADD R4, R4, #-1 ; Уменьшение R4 на 1  
 BRz OUTPUT\_LOOP ; Если R4 = 0, массив отсортирован  
 LEA R3, ARRAY ; Указатель на начало массива  
 ADD R5, R4, #0 ; Копирование R4 в R5 (для внутреннего цикла)  
  
 INNER\_LOOP  
 LDR R0, R3, #0 ; Загрузка текущего элемента  
 LDR R1, R3, #1 ; Загрузка следующего элемента  
 NOT R2, R1 ; Инверсия битов следующего элемента  
 ADD R2, R2, #1 ; Инкрементация инвертированного значения  
 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 ; Уменьшение счетчика  
 BRp INNER\_LOOP ; Повторение внутреннего цикла, если не все элементы проверены  
 BRnzp OUTER\_LOOP ; Повторение внешнего цикла, если не все элементы отсортированы  
  
 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 ; Загрузка ASCII-кода символа '0'  
 ADD R0, R0, R4 ; Преобразование числа обратно в символ ASCII  
 OUT ; Вывод символа  
 LEA R0, NEWLINE ; Загрузка символа новой строки  
 PUTS ; Вывод новой строки  
 ADD R1, R1, #1 ; Переход к следующему элементу  
 ADD R2, R2, #1 ; Увеличение счетчика элементов  
 ADD R3, R2, #-10 ; Проверка на конец вывода  
 BRz END ; Завершение программы, если все элементы выведены  
 BRnzp PRINT\_LOOP ; Повторение вывода следующего элемента  
  
 ; Завершение программы  
 END  
 HALT  
  
 ; Константы и данные  
 INIT .FILL #0 ; Начальное значение счетчика  
 INPUT\_PROMPT .STRINGZ "Enter a number: "  
 NEWLINE .STRINGZ "\n" ; Символ новой строки  
 ASCII\_ZERO .FILL #48 ; ASCII-код символа '0'  
 ARRAY .BLKW #10 ; Массив для хранения чисел  
 ARRAY\_LEN .FILL #10 ; Длина массива  
  
 .END

## 3.3. Пример работы программы

На рис.3 представлен результат работы программы упражнения 3.

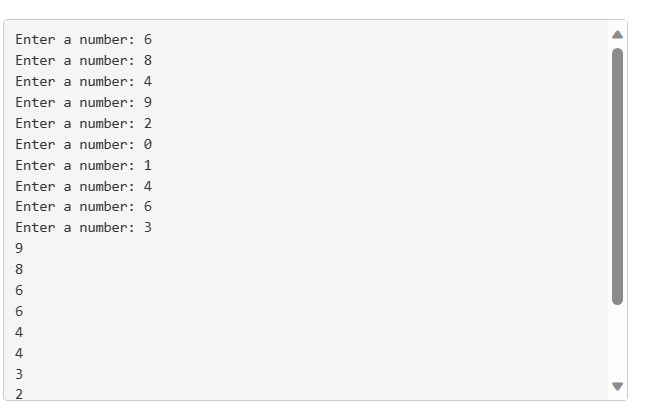
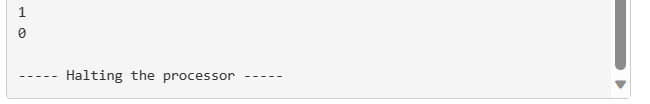
  


Рисунок 3. Пример работы программы упражнения 3

# Упражнение 4

## 4.1. Постановка задачи

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

***4.2 Код программы***

exc4.asm

.ORIG x3000  
  
 ; Инициализация  
 LD R2, INIT ; Загрузка начального значения счетчика в R2  
 LEA R1, BUFFER ; Загрузка адреса буфера ввода в R1  
 LD R4, ASCII\_ZERO ; Загрузка ASCII-кода символа '0' в R4  
 LD R5, NEG\_ASCII\_ZERO ; Загрузка отрицательного ASCII-кода символа '0' в R5  
  
 LEA R0, INPUT\_PROMPT ; Загрузка адреса строки приглашения ввода в R0  
 PUTS ; Вывод строки приглашения  
  
 ; Ввод данных  
 READ\_LOOP  
 GETC ; Считывание символа с консоли  
 OUT ; Вывод символа на консоль  
  
 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 ; Загрузка текущего символа из буфера  
 ADD R0, R0, R5 ; Преобразование символа в число  
 STR R0, R6, #0 ; Сохранение числа в массиве  
 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 ; Копирование длины массива в R4  
  
 OUTER\_LOOP  
 ADD R4, R4, #-1 ; Уменьшение R4 на 1  
 BRz OUTPUT\_LOOP ; Если R4 = 0, массив отсортирован  
 LEA R3, ARRAY ; Указатель на начало массива  
 ADD R5, R4, #0 ; Копирование R4 в R5  
  
 INNER\_LOOP  
 LDR R0, R3, #0 ; Загрузка текущего элемента  
 LDR R1, R3, #1 ; Загрузка следующего элемента  
 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 ; Переход к следующему элементу  
 ADD R5, R5, #-1 ; Уменьшение счетчика  
 BRp INNER\_LOOP ; Повторение внутреннего цикла, если не все элементы проверены  
 BRnzp OUTER\_LOOP ; Повторение внешнего цикла, если не все элементы отсортированы  
  
 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  
 ADD R0, R0, R4 ; Преобразование числа обратно в символ ASCII  
 OUT ; Вывод символа  
 LEA R0, NEWLINE ; Вывод символа новой строки  
 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