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»

по дисциплине «Языки программирования и методы трансляции»

Выполнил: студент группы ИСТ-22-1б Богданов Д.О.

Проверил: Батин Сергей Евгеньевич

Пермь, 2024

**Содержание**

[Задание 1 3](#_Toc160572761)

[1.1. Постановка задачи 3](#_Toc160572762)

[1.2. Решение задачи, код программы 3](#_Toc160572763)

[1.3. Решение задачи, код программы 3](#_Toc160572764)

[1.4. Тестирование работы программы 3](#_Toc160572765)

[Задание 2 4](#_Toc160572766)

[2.1. Постановка задачи 4](#_Toc160572767)

[2.2. Решение задачи, код программы 4](#_Toc160572768)

[2.3. Решение задачи, код программы 4](#_Toc160572769)

[2.4. Тестирование работы программы 5](#_Toc160572770)

[Задание 3 6](#_Toc160572771)

[3.1. Постановка задачи 6](#_Toc160572772)

[3.2. Решение задачи, код программы 6](#_Toc160572773)

[3.3. Решение задачи, код программы 10](#_Toc160572774)

[3.4. Тестирование работы программы 11](#_Toc160572775)

[Задание 4 12](#_Toc160572776)

[4.1. Постановка задачи 12](#_Toc160572777)

[4.2. Решение задачи, код программы 12](#_Toc160572778)

[4.3. Решение задачи, код программы 14](#_Toc160572779)

[4.4. Тестирование работы программы 14](#_Toc160572780)

# Задание 1

## 1.1. Постановка задачи

Написать и запустить программу, выводящую строку Hello World.

## 1.2. Решение задачи, код программы

.ORIG x3000

LEA R0, HW

PUTS

HALT

HW  .STRINGZ "Hello globe!"

.END

## 1.3. Решение задачи, код программы

Резервируем точку начала программы в памяти, подгружаем адрес строки, выводим, стоп.

## 1.4. Тестирование работы программы

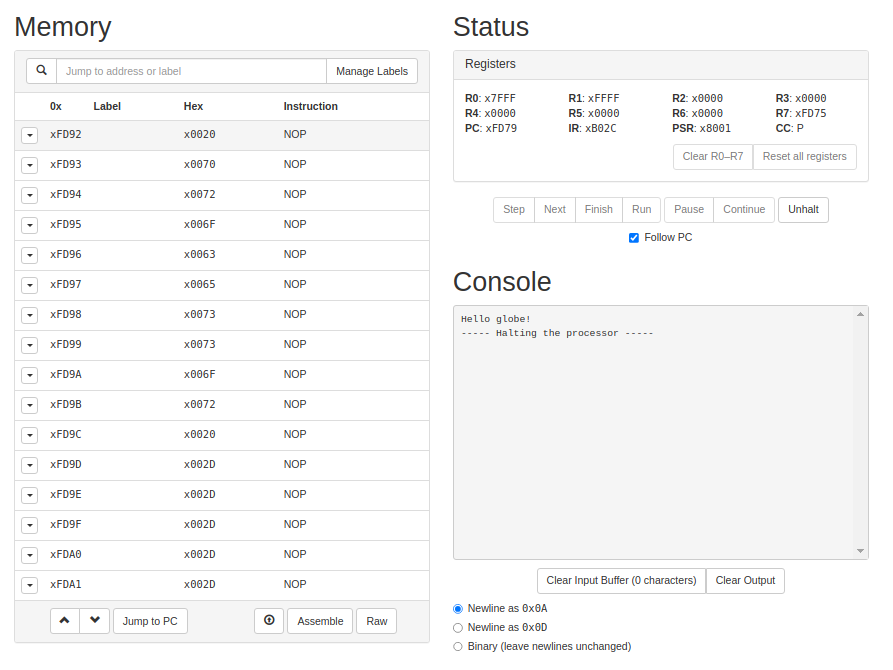


Рис.1 Тестирование программы lab4\_tsk1.asm

# Задание 2

## 2.1. Постановка задачи

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

## 2.2. Решение задачи, код программы

.ORIG x3000

WARN\_STR

    LEA R0, STG

    PUTS

    LEA R1, IO

LOOP\_IO

    GETC

    OUT

    ADD R2, R0, #-10

    BRz PRINT

    STR R0, R1, #0

    ADD R1, R1, #1

    BRnzp LOOP\_IO

PRINT

    LEA R0, IO

    PUTS

HALT

STG .STRINGZ "Please, input your string with max size of 100 characters: "

IO .BLKW #100

.END

## 2.3. Решение задачи, код программы

Выводим HS, как в 1 проге. Задаем цикл для принятия символа, отображения его в терминале. В R0 подгружаем адрес памяти со строкой, с резервом в 100 ячеек.

В начале блока проверка на enter, если аски код принятого символа равен 10, то при вычитании 10 получаем 0, конец ввода.

Основа цикла – подгрузка символа из нулевого регистра в адрес памяти строки, сдвиг адреса на 1 позицию.

Сабрутина принт выводит строку целиком, до первого нуля с помощью PUTS.

## 2.4. Тестирование работы программы

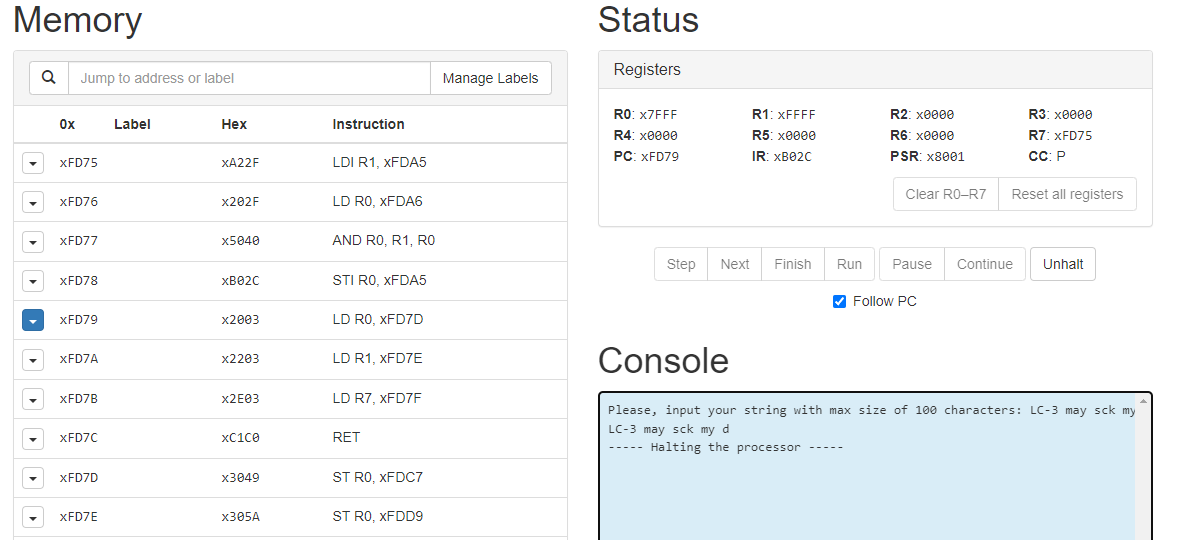


Рис.2 Тестирование программы lab4\_tsk2.asm

# Задание 3

## 3.1. Постановка задачи

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

## 3.2. Решение задачи, код программы

.ORIG x3000

LEA R0, HS; start:

PUTS

AND R0, R0, #0

LEA R1, STOR; getting digits for STOR:

GETTING

    GETC

    OUT

    LD R2, ENDLINE; check end

    ADD R2, R2, R0

    BRz PREPARE1

    LD R2, NAZERO; refrac from ascii

    ADD R0, R0, R2

    STR R0, R1, #0

    ADD R1, R1, #1

    BRnzp GETTING

PREPARE1; loading R-s

    LEA R7, ARR

    LEA R6, STOR

    LEA R5, MLTPS

    AND R0, R0, #0

PREMULT; load next digit and mltplr

    LDR R1, R6, #0; digit (cntr)

    LDR R2, R5, #0; multiplyer

MULT; multiplying every base of the number

    ADD R1, R1, #-1; decr digit (cntr)

    BRn REMULT

    ADD R4, R4, R2; res

    BRnzp MULT

REMULT; shift next digit base in STOR

    ADD R6, R6, #1

    ADD R5, R5, #1

    LDR R1, R6, #0; spc check

    LD R2, SPC

    ADD R1, R2, R1

    BRz FILL

    BRnzp PREMULT

FILL; fill in array with number

    LEA R5, MLTPS; renew MLTPS pos

    STR R4, R7, #0; store result

    ADD R7, R7, #1

    AND R4, R4, #0; clear

    ADD R3, R3, #1; incr num cntr

    LD R1, ARRCNTR; check end of cntr

    ADD R2, R3, R1

    BRz SORTOUT

    ADD R6, R6, #1; spc shift

    BRnzp PREMULT

SORTOUT

    LEA R1, ARR

    LD R0, ARRCNTR; marker occurances check

    ADD R0, R0, #1; (qnty of arr el-s) - 1

    ADD R0, R7, R0; check occur-s, if 0 go out

    BRz PREPARE2

    LD R7, CNTR; zeroing swap cntr

    LD R4, CNTR; zeroing inner cntr

    ADD R2, R1, #0; num 1 adr

    ADD R3, R1, #1; num 2 adr

SORTIN

    LDR R5, R2, #0; num 1

    LDR R6, R3, #0; num 2

    NOT R6, R6; add minus to number

    ADD R6, R6, #1

    ADD R0, R5, R6; compare

    BRnz SKIP

    BRp SWAP

SKIP

    ADD R6, R6, #-1; positive again

    NOT R6, R6

    ADD R2, R2, #1; move adr

    ADD R3, R3, #1

    ADD R4, R4, #-1; decr cntr

    BRz SORTOUT

    BRp SORTIN

SWAP

    ADD R7, R7, #-1; marker

    ADD R6, R6, #-1; positive again

    NOT R6, R6

    STR R6, R2, #0; swap

    STR R5, R3, #0

    BRnzp SORTIN

PREPARE2; prepare for division the number on its bases

    AND R3, R3, #0; clear R3

    LEA R7, ARR

    LEA R6, MLTPS

    LDR R1, R7, #0; num

DIV; division by digit base

    AND R4, R4, #0; clear compare

    LDR R2, R6, #0; minus mltpls

    NOT R2, R2

    ADD R2, R2, #1

    ADD R4, R1, R2; compare

    BRn PRESHOW

    ADD R1, R1, R2; num

    ADD R0, R0, #1; qnty of digits

    BRnzp DIV

PRESHOW; output

    LD R2, AZERO; print

    ADD R0, R0, R2

    OUT

    AND R0, R0, #0

    ADD R1, R1, #0; check remaining digits

    BRz RELOAD

    ADD R6, R6, #1; move mltpl

    BRnzp DIV

RELOAD; load next number, check end

    LD R0, ASPC; divide num-s by spc

    OUT

    AND R0, R0, #0

    ADD R3, R3, #1; incr num cntr

    LD R2, ARRCNTR; check end of cntr

    ADD R2, R3, R2

    BRz STOP

    LEA R7, ARR; shift array on 1 pos (why this fckn lc-3 is clearing Registers so unobviously?????? )

    ADD R7, R7, R3

    LDR R1, R7, #0

    LEA R6, MLTPS

    BRnzp DIV

STOP

    ADD R0, R0, #10

    OUT

    LEA R0, GB

    PUTS

    HALT

HS .STRINGZ "Pls, input ten 4-digit numbers: "; start

GB .STRINGZ "Let LC-3 burn in hell :)"

ARRCNTR .FILL #-5; CHANGE #-10

CNTR .FILL #4; CHANGE # QNTY - 1

AZERO .FILL #48

NAZERO .FILL #-48

ENDLINE .FILL #-10

SPC .FILL #16

ASPC .FILL #32

ARR .BLKW #5; CHANGE #10?

STOR .BLKW #25; CHANGE #50

MLTPS .FILL #1000

.FILL #100

.FILL #10

.FILL #1

.END

## 3.3. Решение задачи, код программы

Писать подобное на LC-3 так классно 😊.

Основная суть – подгрузить память для массива, память для массива множителей от 1 до 1000, память для хранилища принимаемых символов-цифр. Далее принять числа пользователя ровно в 4 цифры, с пробелом после каждого числа, иначе цикл не завершится, (можно было бы запариться и добавить выходи из цикла при коде энтера, но шло бы оно) цифры в хранилище.

Циклом для каждого числа по разрядам начислить по нужному множителю столько раз, каким значением обладает цифра, то есть для 4331 – 4 раза будет суммирована тысяча, 3 раза сотня и т.д., цикл проходит по каждой цифре каждого числа, записывает итоговое значение в массив ARR.

Хотел добавить вариант интересней, с помощью битового умножения, но на такое в подобной программе просто не хватает реигстров.

Далее сортировка чисел пузырьком. В R7 загружаем маркер вхождений в свап, при каждом заходе декремент, если не было ни одного захода в свап – массив отсортирован – заходим в подготовку к выводу. Каждая пара цифр в числе проходит через скип, даже, если зашла в свап.

Реализуем деление вычитанием разрядов чисел, теперь проверкой на выход из цикла служит негативный результат вычитания множителя из числа, то есть, если была 1221 вычитаем 1000, получаем 221, записываем 1, заходим заново, вычитаем получаем 221 – 1000 = -779 число отрицательное - выводим цифру. После полного цикла вычитания в R1, куда было загружено число остается 0, он служит маркером для выхода из цикла для числа, ставим пробел проделываем то же со следующим. Главный счетчик равен кол-ву изначальных цифр, копится с каждой итерацией, проверяется, получили ноль – выход, гудбай.

## 3.4. Тестирование работы программы

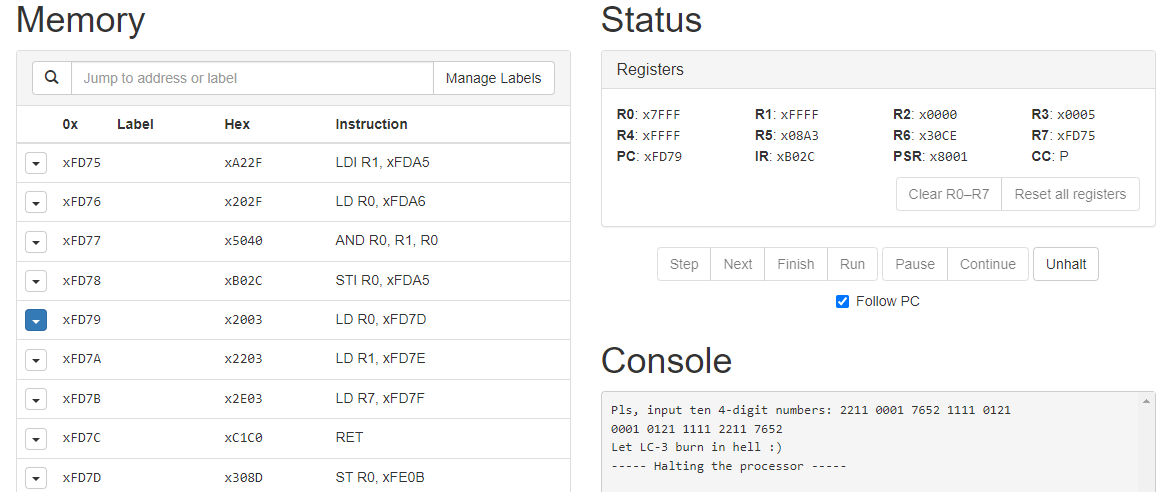


Рис.3 Тестирование программы lab4\_tsk2.asm

Тестирование для 5 цифр, чтобы было 10 нужно просто поменять значения, помеченные CHANGE в самом конце

# Задание 4

## 4.1. Постановка задачи

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

## 4.2. Решение задачи, код программы

.ORIG x3000

LEA R0, HS

PUTS

LEA R1, STRING

INP

    GETC

    OUT

    LD R2, LINE; check end

    ADD R2, R2, R0

    BRz SORTOUT

    LD R2, ZERO; refrac from ascii

    ADD R0, R0, R2

    STR R0, R1, #0

    ADD R1, R1, #1

    BRnzp INP

SORTOUT

    LEA R1, STRING

    LD R0, NCNTR; outter cntr

    ADD R0, R7, R0; check swap count

    BRz LOAD

    LD R7, CNTR; zeroing swap cntr

    LD R4, CNTR; zeroing inner cntr

    ADD R2, R1, #0; dig 1 adr

    ADD R3, R1, #1; dig 2 adr

SORTIN

    LDR R5, R2, #0; dig 1

    LDR R6, R3, #0; dig 2

    NOT R6, R6; negative digit

    ADD R6, R6, #1

    ADD R0, R5, R6; compare

    BRnz SKIP

    BRp SWAP

SKIP

    ADD R6, R6, #-1; positive again

    NOT R6, R6

    ADD R2, R2, #1; move adr

    ADD R3, R3, #1

    ADD R4, R4, #-1; decr cntr

    BRz SORTOUT

    BRp SORTIN

SWAP

    ADD R7, R7, #-1; marker

    ADD R6, R6, #-1; positive again

    NOT R6, R6

    STR R6, R2, #0; swap

    STR R5, R3, #0

    BRnzp SORTIN

LOAD

    LEA R1, STRING

    LD R2, PZERO

    LD R3, NCNTR

SHOW

    LDR R0, R1, #0

    ADD R0, R0, R2

    ADD R1, R1, #1

    OUT

    ADD R3, R3, #1

    BRp STOP

    BRnzp SHOW

STOP

    HALT

HS .STRINGZ "Pls, input a 10-digit string in random order without any separators: "

STRING .BLKW #100

CNTR .FILL #9

NCNTR .FILL #-9

LINE .FILL  #-10; new line

ZERO .FILL #-48; -0

PZERO .FILL #48; +0

.END

## 4.3. Решение задачи, код программы

Смысла описывать не вижу, сортировка такая же, как в 3. Задание 3 по сложности превосходит 4 в разы.

## 4.4. Тестирование работы программы

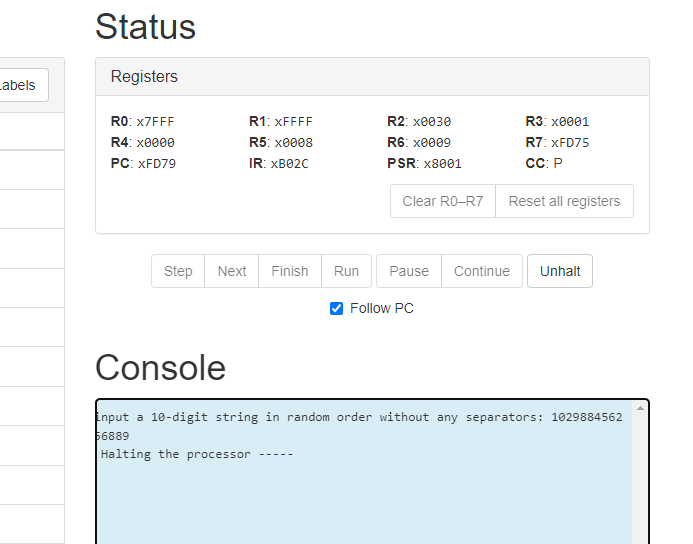
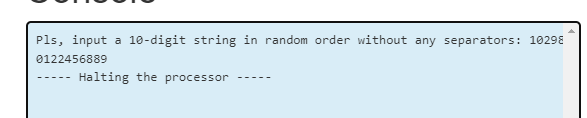
 

Рис.4 Тестирование программы lab4\_tsk4.asm