Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и Информационных технологий

Кафедра Информатики

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7**

Преподаватель \_\_\_\_\_\_\_\_ Пересунько П. В.

подпись, дата инициалы, фамилия

Студент КИ15–17Б, 031510065 \_\_\_\_\_\_\_\_\_\_\_ Радионов Т.В.

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2016

**1 Цель работы**

Требуется: разработать ассемблерную программу, исходный код которой

представляет собой программу, разделенную на основную часть и

подпрограммы (не менее двух). Результат вычислений выводится на экран.

Дана матрица, найти максимальное значение среди минимальных

по строкам.

**2 Исходные тексты программ**

.data #секция данных

txt1: .string "\n\n\Matrix:\n"

.equ txt1len,.-txt1 #длина строки

txt2: .string "\n\nResult = "

.equ txt2len,.-txt2 #длина строки

num: .string " " #строка (3 символа) для вывода числа с пробелом

nextln: .string "\n" #символ перехода на новую строку

#------------------------------

.equ matry,4 #количество строк матрицы

.equ matrx,5 #количество столбцов матрицы

matr: .byte 3,7,3,4,5 #элементы матрицы

.byte 6,44,3,4,6

.byte 5,6,33,3,1

.byte 2,0,15,3,1

mins: .space matry #массив минимальных элементов строк

#------------------------------

.text #секция кода

.globl main #\_start - точка запуска программы

main: movl $txt1,%ecx #загружаем указатель на строку

movl $txt1len,%edx #загружаем длину строки

call tprint #вызываем подпрограмму печати строки

movl $matr,%esi #загружаем начальный адрес матрицы

movl $matry,%ecx #загружаем счетчик цикла - количество строк матрицы

m1: pushq %rcx #сохраняем значение регистра в стеке

movl $matrx,%ecx #загружаем счетчик цикла - количество элементов в строке матрицы

m2: pushq %rcx #сохраняем значение регистра в стеке

#---

movb (%esi),%al #загружаем текущий элемент матрицы

pushq %rsi #сохраняем значение регистра в стеке

call nprint #вызываем подпрограмму печати числа

popq %rsi #восстанавливаем значение регистра из стека

incl %esi #инкрементируем адрес - переходим к следующему элементу матрицы

popq %rcx #восстанавливаем значение регистра из стека

loop m2 #выполняем цикл для всех элементов строки

#---

movl $nextln,%ecx #загружаем указатель на строку

movl $2,%edx #загружаем длину строки

call tprint #вызываем подпрограмму печати строки

popq %rcx #восстанавливаем значение регистра из стека

loop m1 #выполняем цикл для всех следующих слов

#---

movl $matr,%esi #загружаем начальный адрес матрицы

movl $mins,%edi #загружаем начальный адрес массива минимальных элементов строк

movl $matry,%ecx #загружаем счетчик цикла - количество строк матрицы

m3: pushq %rcx #сохраняем значение регистра в стеке

movl $matrx,%ecx #загружаем количество элементов в строке матрицы

call min #вызываем подпрограмму нахождения минимального элемента

movb %al,(%edi) #сохраняем минимальный элемент строки в массиве минимальных элементов строк

incl %edi

popq %rcx #восстанавливаем значение регистра из стека

loop m3 #выполняем цикл для всех строк матрицы

#---

movl $txt2,%ecx #загружаем указатель на строку

movl $txt2len,%edx #загружаем длину строки

call tprint #вызываем подпрограмму печати строки

#---

movl $mins,%esi #загружаем начальный адрес массива минимальных элементов строк

movl $matry,%ecx #загружаем количество элементов в массиве минимальных элементов строк

call max #вызываем подпрограмму нахождения максимального элемента

call nprint #вызываем подпрограмму печати числа

movl $nextln,%ecx #загружаем указатель на строку

movl $2,%edx #загружаем длину строки

call tprint #вызываем подпрограмму печати строки

#---

exit: movl $1,%eax #номер системного вызова - завершение программы

movl $0,%ebx #результат завершения

int $0x80 #вызов прерывания 0x80

#==============================

#=======ПОДПРОГАММЫ============ #РАЗДЕЛ ПОДПРОГРАММ

#-------TPRINT----------------- #Подпрограмма печати строки

#при входе:

#ECX - указатель на строку

#EDX - длина строки

tprint: movl $4,%eax #номер системного вызова - вывод

movl $1,%ebx #номер файлового дескриптора (1 - вывод на экран)

int $0x80 #вызов прерывания 0x80

ret #выход из подпрограммы

#------------------------------

#-------NPRINT----------------- #Подпрограмма печати числа 0..99

#при входе:

#AL - число

#при выходе:

#символы числа в строке NUM

nprint: movb $0,%ah #обнуляем старшую часть регистра AX

movb $10,%bl #загружаем делитель =10

divb %bl #производим деление AX/BL

movb $' ',(num) #очищаем (загружаем символ пробела в) 1-ый символ в строке num

cmpb $0,%al #если количестов десятков числа нулевое, то

je npr1 #переходим далее

#иначе

add $'0',%al #получаем символ 1-ой цифры - прибавляем ASCII-код символа "0" (ноль)

movb %al,(num) #сохраняем 1-ый символ в строке num

npr1: add $'0',%ah #получаем символ 2-ой цифры - прибавляем ASCII-код символа "0" (ноль)

movb %ah,(num+1) #сохраняем 2-ой символ в строке num

movl $num,%ecx #загружаем указатель на строку

movl $3,%edx #загружаем длину строки

call tprint #вызываем подпрограмму печати строки

ret #выход из подпрограммы

#------------------------------

#-------MIN-------------------- #Подпрограмма нахождения минимального элемента

#при входе:

#ESI - начальный адрес

#ECX - количество элементов

#при выходе:

#AL - значение минимального элемента

min: movb (%esi),%al #загружаем 1-ый элемент - это начальное минимальное значение

incl %esi #инкрементируем адрес - переходим к следующему элементу

decl %ecx #декрементируем количество обрабатываемых элементов

min1: cmpb (%esi),%al #если текущий элемент не меньше минимального, то

jbe min2 #переходим далее

#иначе

movb (%esi),%al #сохраняем новое минимальное значение

min2: incl %esi #инкрементируем адрес - переходим к следующему элементу

loop min1 #выполняем цикл для всех элементов

ret #выход из подпрограммы

#------------------------------

#-------MAX-------------------- #Подпрограмма нахождения максимального элемента

#при входе:

#ESI - начальный адрес

#ECX - количество элементов

#при выходе:

#AL - значение максимального элемента

max: movb (%esi),%al #загружаем 1-ый элемент - это начальное максимальное значение

incl %esi #инкрементируем адрес - переходим к следующему элементу

decl %ecx #декрементируем количество обрабатываемых элементов

max1: cmpb (%esi),%al #если текущий элемент не больше максимального, то

jae max2 #переходим далее

#иначе

movb (%esi),%al #сохраняем новое максимальное значение

max2: incl %esi #инкрементируем адрес - переходим к следующему элементу

loop max1 #выполняем цикл для всех элементов

ret #выход из подпрограммы

#------------------------------

**3 Примеры работы программ**

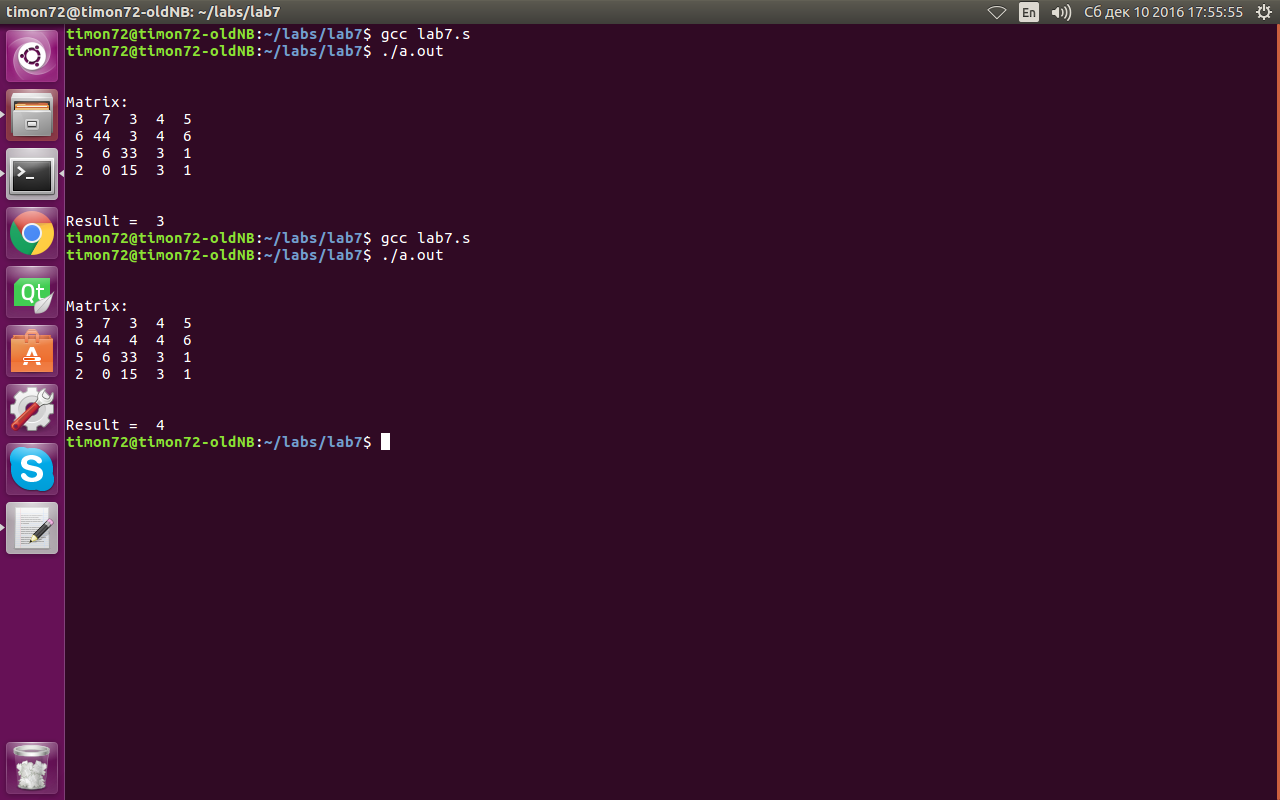


Рисунок 1