|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ДОМАШНЯЯ РАБОТА №1**

**«Обработка двухмерных массивов целых чисел»**

**ДИСЦИПЛИНА: «Машинно-зависимые языки программирования»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-31Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Отрошенко Т. В.)  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Амеличева К. А.)  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

**Цель работы:** изучить особенности обработки двухмерных массивов на языке ассемблер.

**Задачи:** работа предусматривает применение основных приемов обработки массивов: создание массивов случайным образом с использованием датчика случайных чисел, ввод с клавиатуры, задание массивов по определенному закону, нахождение максимального и минимального элементов массива, перестановка строк и столбцов матрицы, сортировка строк и столбцов, с использованием алгоритма сортировки одномерного массива, перестановка блоков внутри матрицы, умножение матриц.

1. Ввести с клавиатуры и вывести на экран матрицу n×n (матрица может содержать нулевые и отрицательные элементы;

2. Реализовать простейший интерфейс взаимодействия с пользователем для выполнения задания варианта

* Транспонировать матрицу, результат вынести на экран;
* Обработка элементов матрицы (задание а, б, в условии варианта), результат выполнения отобразить на экране;
* Завершение выполнения программы.

**Вариант 25**

****а) В каждой строке разместить вначале нулевые элементы, затем все остальные.

б) Проверить строки с нечѐтными номерами на возрастание, с чѐтными номерами – на убывание.

в) Найдите среднее арифметическое положительных чисел среди элементов матрицы, выделенных чѐрным цветом.

**Описание макросов:**

|  |  |  |
| --- | --- | --- |
| Номер | Название | Страница |
| 1 | mWriteAX | 5 |
| 2 | mReadAX | 6 |
| 3 | mWriteMatrix | 7 |
| 4 | mReadMatrix | 7 |
| 5 | mCLS | 5 |
| 6 | mWriteStr | 5 |
| 7 | mTransposeMatrix | 8 |
| 8 | mZiroBeforeMatrix | 9 |
| 9 | mTestIncrease | 10 |
| 10 | mGetAverageAboveDiagonal | 11 |

**mWriteAX – макрос для вывода десятичного числа**. Сохраняем все используемые регистры в стек. Проверяем число на знак. Если число отрицательное, выводим знак минус и берем его модуль. Затем переводим число в строку следующим образом: делим число на десять, переводим остаток в символ и помещаем полученный символ в стек. Выполняем предыдущие действия, пока число не станет равно 0. После этого, достаем из стека символ и выводим его. Повторяем столько раз, сколько делили число на 10. В конце макроса возвращаем из стека значения регистров.

**mReadAX – макрос для обработки ввода десятичного числа.** Для начала сохраняем все используемые регистры в стек. Затем вводим число, которое считываем как массив символов, и переходим на новую строку. Если число отрицательное, смещаем указатель по массиву на 1. Затем переводим каждый символ массива в соответствующее число, отняв от него 30h, и, умножив предыдущую сумму на 10, прибавляем число к этой сумме. Выполняем предыдущие действия пока не достигнем конца массива. Если число отрицательное, инвертируем его. В конце возвращаем из стека значения регистров.

**mWriteMatrix – макрос для вывода матрицы.** Совершается обход поэлементный обход матрицы и при каждой итерации выводится элемент с отступом. . В конце возвращаем из стека значения регистров.

**mReadMatrix – макрос записи матрицы с консоли.** Для начала сохраняем все используемые регистры в стек. После запрашивается размер матрицы и заносится в соответствующую переменную. В двух циклах от 0 до введенного значения запрашивается ввод элемента матрицы и заносится в память. В конце возвращаем из стека значения регистров.

**mCLS – макрос очистки экрана.** Для начала сохраняем все используемые регистры в стек. Макрос принимает значение, с которого начинает очищаться экран. В конце возвращаем из стека значения регистров.

**mWriteStr – макрос вывода строки.** Для начала сохраняем все используемые регистры в стек. Макрос на вход получает адрес выводимой строки и выводит ее в текущее положение курсора. В конце возвращаем из стека значения регистров.

**mTransposeMatrix – макрос создания транспонированной матрицы.** Для начала сохраняем все используемые регистры в стек. На вход передаются адрес оригинальной матрицы, ее размерность и адрес для хранения транспонированной. Совершая поэлементный обход матрицы, значения оригинальной заносятся в транспонированную с учетом смены адресов (адрес строки встает на место столбца и наоборот). В конце возвращаем из стека значения регистров.

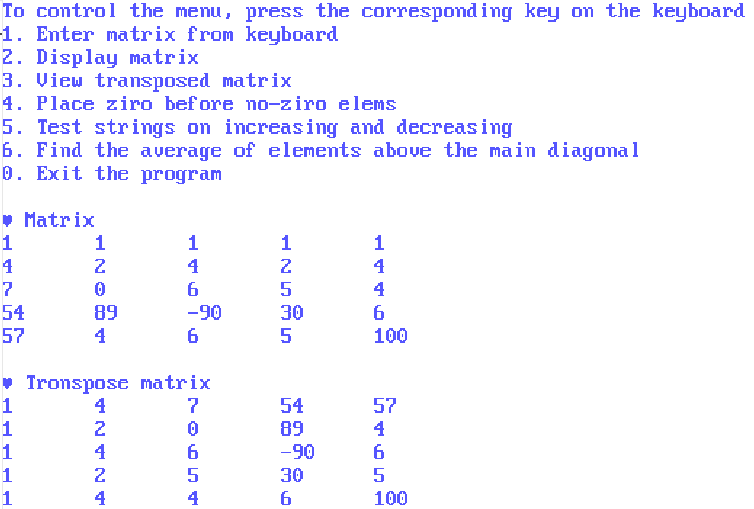
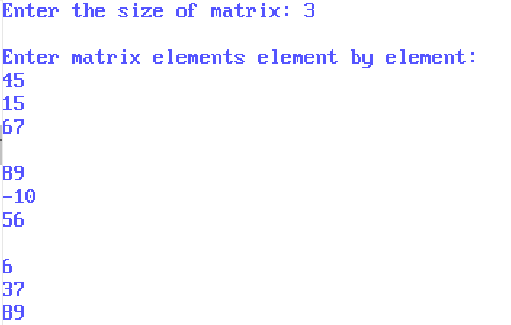
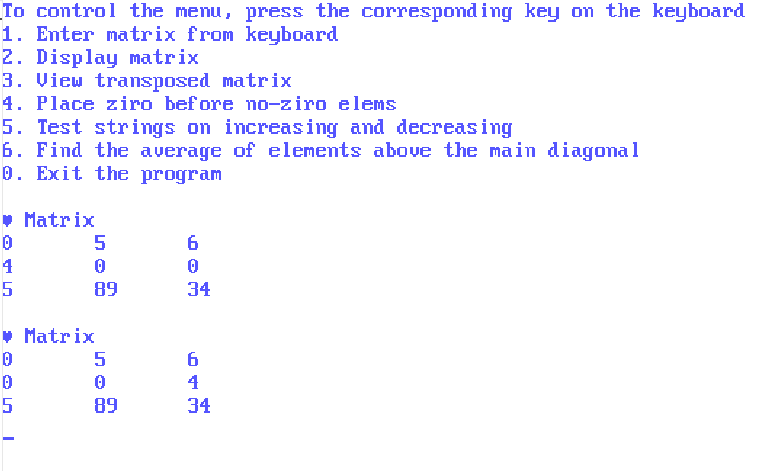
**mZiroBeforeMatrix – макрос для помещения нолей в начало строки.** Для начала сохраняем все используемые регистры в стек. Для каждой строки в матрице: проходим первый раз, помещая в стек все не нулевые элементы и считая их количество, после чего совершаем цикл такого же размера в обратно порядке, помещая элементы из стека в строку, если они есть и ноли, если они кончились (проверяется счетчик ненулевых элементов). В конце возвращаем из стека значения регистров.

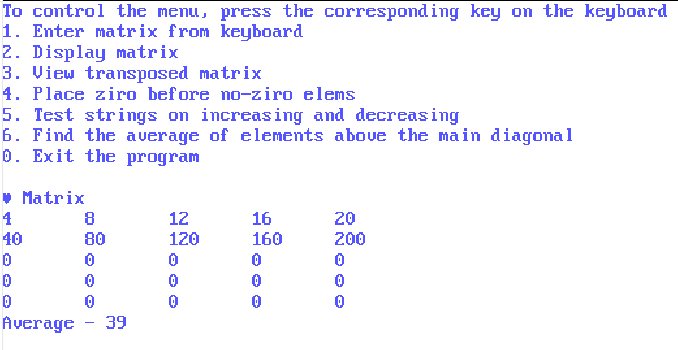
**mTestIncrease – макрос проверки монотонности строк.** Для начала сохраняем все используемые регистры в стек. Создается цикл для обхода строк матрицы. Потом выполнение разделяется на две части в зависимости от четности строки. Каждые два элемента сравниваются между собой, и если нарушается условие, то совершается переход к следующей строке. Для строк в которых ни разу не нарушилось условие монотонности (с учетом знака, зависящего от четности строки) выводится номер и информационное сообщение об успехе. В конце возвращаем из стека значения регистров.

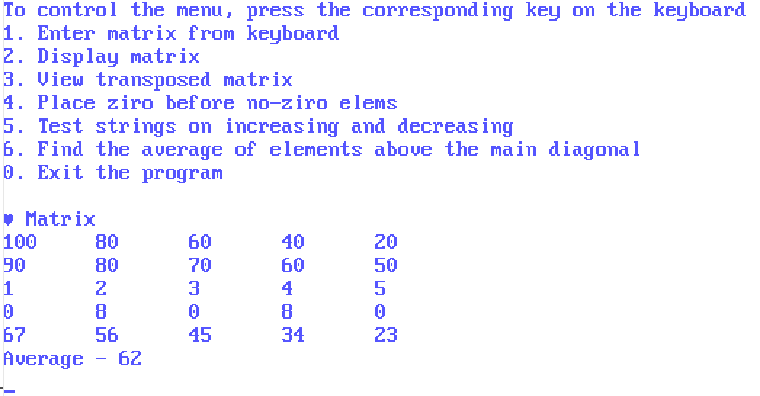
**mGetAverageAboveDiagonal – макрос нахождения среднего арифметического элементов заданного положения.** Для начала сохраняем все используемые регистры в стек. Во вложенном цикле по размеру матрицы проверяем соответствие регистров условиям: выше главной диагонали (индекс столбца больше индекса строки) и выше побочной диагонали (сумма индексов меньше половины от размера матрицы). Если элемент прошел проверку его значение добавляется к накопителю и счетчик их количества увеличивается на один, если хотя бы одно условие не соблюдено совершается переход к следующему элементу. После завершения циклов выводится значение накопителя, деленное на количество элементов. В конце возвращаем из стека значения регистров.

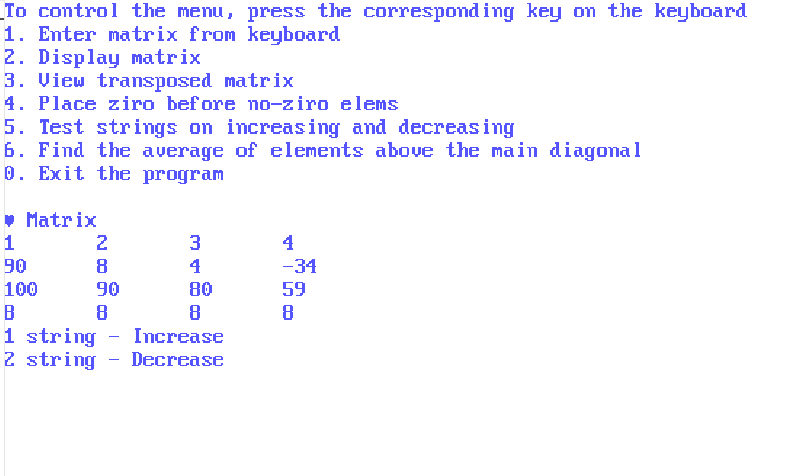
|  |
| --- |
| .MODEL small  .STACK 100h  .486 ; Включает сборку инструкций для процессора 80386  mWriteStr macro string  push ax ; Сохранение регистров, используемых в макросе, в стек  push dx  mov ah, 09h ; 09h - функция вывода строки на экран  mov dx, offset string  int 21h  pop dx ; Перенос сохранённых значений обратно в регистры  pop ax  endm mWriteStr  mCLS macro start  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push dx  mov ah, 10h  mov al, 3h  int 10h ; Включение режима видеоадаптора с 16-ю цветами  mov ax, 0600h ; ah = 06 - прокрутка вверх  mov bh, 11111001b ; белый фон, синий текст  mov cx, start ; ah = 00 - строка верхнуго левого угла  mov dx, 184Fh ; dh = 18h - строка нижнего правого угла  int 10h ; Очистка экрана и установка цветов фона и текста  mov dx, 0 ; dh - строка, dl - столбец  mov bh, 0 ; Номер видео-страницы  mov ah, 02h ; 02h - функция установки позиции курсора  int 10h ; Устанавливаем курсор на позицию (0, 0)  pop dx ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  pop ax  endm mCLS  mWriteAX macro      local convert, write      push ax ; Сохранение регистров, используемых в макросе, в стек      push bx      push cx      push dx      push di      mov cx, 10 ; cx - основание системы счисления      xor di, di ; di - количество цифр в числе      or ax, ax ; Проверяем, равно ли число в ax нулю и устанавливаем флаги      jns convert ; Переход к конвертированию, если число в ax положительное      push ax      mov dx, '-'      mov ah, 02h ; 02h - функция вывода символа на экран      int 21h ; Вывод символа "-"      pop ax      neg ax ; Инвертируем отрицательное число      convert:      xor dx, dx      div cx ; После деления dl = остатку от деления ax на cx      add dl, '0' ; Перевод в символьный формат      inc di ; Увеличиваем количество цифр в числе на 1      push dx ; Складываем в стек      or ax, ax ; Проверяем, равно ли число в ax нулю и устанавливаем флаги      jnz convert ; Переход к конвертированию, если число в ax не равно нулю      write: ; Вывод значения из стека на экран      pop dx ; dl = очередной символ      mov ah, 02h      int 21h ; Вывод очередного символа      dec di ; Повторяем, пока di <> 0      jnz write      pop di ; Перенос сохранённых значений обратно в регистры      pop dx      pop cx      pop bx      pop ax  endm mWriteAX  mReadAX macro buffer, size  local input, startOfConvert, endOfConvert  push bx ; Сохранение регистров, используемых в макросе, в стек  push cx  push dx  input:  mov [buffer], size ; Задаём размер буфера  mov dx, offset [buffer]  mov ah, 0Ah ; 0Ah - функция чтения строки из консоли  int 21h  mov ah, 02h ; 02h - функция вывода символа на экран  mov dl, 0Ah  int 21h ; Переносим курсор на новою строку  xor ah, ah  cmp ah, [buffer][1] ; Проверка на пустую строку  jz input ; Если строка пустая - переходим обратно к вводу  xor cx, cx  mov cl, [buffer][1] ; Инициализируем переменную счетчика  xor ax, ax  xor bx, bx  xor dx, dx  mov bx, offset [buffer][2] ; bx = начало строки  (строка начинается со второго байта)  cmp [buffer][2], '-' ; Проверяем, отрицательное ли число  jne startOfConvert ; Если отрицательное - пропускаем минус  inc bx  dec cl  startOfConvert:  mov dx, 10  mul dx ; Умножаем на 10 перед сложением с младшим разрядом  cmp ax, 8000h ; Если число выходит за границы, то  jae input ; возвращаемся на ввод числа  mov dl, [bx] ; Получаем следующий символ  sub dl, '0' ; Переводим его в числовой формат  add ax, dx ; Прибавляем к конечному результату  cmp ax, 8000h ; Если число выходит за границы, то  jae input ; возвращаемся на ввод числа  inc bx ; Переходим к следующему символу  loop startOfConvert  cmp [buffer][2], '-' ; Ещё раз проверяем знак  jne endOfConvert ; Если знак отрицательный, то  neg ax ; инвертируем число  endOfConvert:  pop dx ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  endm mReadAX  mReadMatrix macro matrix, row, col  local rowLoop, colLoop  JUMPS ; Директива, делающая возможным большие прыжки  push bx ; Сохранение регистров, используемых в макросе, в стек  push cx  push si  xor bx, bx ; Обнуляем смещение по строкам  mov cx, row  rowLoop: ; Внешний цикл, проходящий по строкам  push cx  xor si, si ; Обнуляем смещение по столбцам  mov cx, col  colLoop: ; Внутренний цикл, проходящий по столбцам  mReadAX buffer 4 ; Макрос ввода значения регистра AX с клавиатуры  mov matrix[bx][si], ax  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop  mWriteStr endl ; Макрос вывода строки на экран  ; Перенос курсора и каретки на следующую строку  add bx, col ; Увеличиваем смещение по строкам  add bx, col ; (дважды, так как размер каждого элемента - слово)  pop cx  loop rowLoop  pop si ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  NOJUMPS ; Прекращение действия директивы JUMPS  endm mReadMatrix  mWriteMatrix macro matrix, row, col  local rowLoop, colLoop  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push si  xor bx, bx ; Обнуляем смещение по строкам  mov cx, row  rowLoop: ; Внешний цикл, проходящий по строкам  push cx  xor si, si ; Обнуляем смещение по столбцам  mov cx, col  colLoop: ; Внутренний цикл, проходящий по столбцам  mov ax, matrix[bx][si] ; bx - смещение по строкам, si - по столбцам  mWriteAX ; Макрос вывода значения регистра AX на экран  ; Вывод текущего элемента матрицы  xor ax, ax  mWriteStr tab ; Макрос вывода строки на экран  ; Вывод на экран табуляции, разделяющей элементы строки  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop  mWriteStr endl ; Макрос вывода строки на экран  ; Перенос курсора и каретки на следующую строку  add bx, col ; Увеличиваем смещение по строкам  add bx, col ; (дважды, так как размер каждого элемента - слово)  pop cx  loop rowLoop  pop si ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  pop ax  endm mWriteMatrix  mTransposeMatrix macro matrix, row, col, resMatrix  local rowLoop, colLoop  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push di  push si  push dx  xor di, di ; Обнуляем смещение по строкам  mov cx, row  rowLoop: ; Внешний цикл, проходящий по строкам  push cx  xor si, si ; Обнуляем смещение по столбцам  mov cx, col  colLoop: ; Внутренний цикл, проходящий по столбцам  mov ax, col  mul di ; Устанавливаем смещение по строкам  add ax, si ; Устанавливаем смешение по столбцам  mov bx, ax  mov ax, matrix[bx]  push ax ; Заносим текущий элемент в стек  mov ax, row  mul si ; Устанавливаем смещение по строкам  add ax, di ; Устанавливаем смешение по столбцам  ; (смещения по строкам и столбцам меняются  ; местами по сравнению с оригинальной матрицей)  mov bx, ax  pop ax  mov resMatrix[bx], ax ; Заносим в новую матрицу элемент, сохранённый в стеке  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop  add di, 2 ; Переходим к следующей строке  pop cx  loop rowLoop  pop dx ; Перенос сохранённых значений обратно в регистры  pop si  pop di  pop cx  pop bx  pop ax  endm mTransposeMatrix  ; !!!!!!РАБОТААЕТ - НЕ ТРОГАЙ!!!!!!!!!  mZiroBeforeMatrix macro matrix, row, col, resMatrix  local rowLoop, colLoop, colLoopReverse, zs, placeziro, next  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push di  push si  push dx  xor di, di ; Обнуляем смещение по строкам  mov cx, row  rowLoop: ; Внешний цикл, проходящий по строкам  push cx  xor si, si ; Обнуляем смещение по столбцам  mov cx, col  mov buffnotnull, 0 ; обнуление счетчика 'ненолей'  colLoop: ; Внутренний цикл, проходящий по столбцам  mov ax, col  mul di ; Устанавливаем смещение по строкам  add ax, si ; Устанавливаем смешение по столбцам  mov bx, ax  mov ax, matrix[bx]  cmp ax, 0  je zs  inc buffnotnull ; dx хранит количество ненулевых элементов  push ax ; Заносим текущий элемент в стек  zs:  add si, 2 ; Переходим к следующему элементу (размером в слово)  dec cx  cmp cx, 0  jne colLoop ; цикл  sub si, 2 ; Переходим обратно к последнему элементу  mov cx, col  colLoopReverse: ; Внутренний цикл, проходящий по столбцам  mov ax, col  mul di ; Устанавливаем смещение по строкам  add ax, si ; Устанавливаем смещение по столбцам  mov bx, ax  cmp buffnotnull, 0  je placeziro  pop ax  mov resmatrix[bx], ax ; тащим со стека, пока элементы не кончатся  dec buffnotnull  jmp next  placeziro:  mov resmatrix[bx], 0 ; оставшееся заполняем нолями  next:  sub si, 2 ; Переходим к предыдущему элементу (размером в слово)  dec cx  cmp cx, 0  jne colLoopReverse ; цикл  add di, 2 ; Переходим к следующей строке  pop cx  loop rowLoop  pop dx ; Перенос сохранённых значений обратно в регистры  pop si  pop di  pop cx  pop bx  pop ax  endm mZiroBeforeMatrix  mTestIncrease macro matrix, row, col  local rowLoop, colLoop, colLoop1, parity, next\_str  JUMPS  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push si  push di  push dx  mov di, 1 ; di - счётчик строк, начиная с единицы  xor bx, bx ; Обнуляем смещение по строкам  mov cx, row  rowLoop:  push cx  xor si, si ; Обнуляем смещение по столбцам  mov cx, col  dec cx  mov ax, di  mov dx, 2h  div dl  cmp ah, 0  jnz parity  colLoop:  mov ax, matrix[bx][si] ; bx - смещение по строкам, si - по столбцам  cmp ax, matrix[bx][si+2]; сравниваем со следующим  jl next\_str ; если не подходит под условие - перейти к следующей строке  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop  mov ax, di  mWriteAX ; Выводим номер текущей строки  mWriteStr dec\_str ; тип проверки  jmp next\_str  parity:  colLoop1:  mov ax, matrix[bx][si] ; bx - смещение по строкам, si - по столбцам  cmp ax, matrix[bx][si+2]; сравниваем со следующим  jg next\_str ; если не подходит под условие - перейти к следующей строке  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop1  mov ax, di  mWriteAX ; Выводим номер текущей строки  mWriteStr inc\_str ; тип проверки  next\_str:  mWriteStr endl  inc di ; Увеличиваем счётчик строк  add bx, col ; Увеличиваем смещение по строкам  add bx, col ; (дважды, так как размер каждого элемента - слово)  pop cx  loop rowLoop  pop dx ; Перенос сохранённых значений обратно в регистры  pop di  pop si  pop cx  pop bx  pop ax  NOJUMPS  endm mTestIncrease  mGetAverageAboveDiagonal macro matrix, row, col  local rowLoop, colLoop, belowTheDiagonal  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push si  push di  push dx  xor ax, ax ; сумма элементов  xor dx, dx ; кол-во элементов  xor di, di ; di - счётчик строк  xor bx, bx ; Обнуляем смещение по строкам  mov cx, row  rowLoop:  push cx  xor si, si ; si - счётчик столбцов  mov cx, col  colLoop:  cmp di, si ; Сравниваем счётчики строк и столбцов  jae belowTheDiagonal ; Если элемент ниже побочной диагонали, перейти к следующему  push ax  push dx  mov dx, col  inc dx  mov ax, si  add ax, di  cmp ax, dx  pop dx  pop ax  ja belowTheDiagonal  add ax, matrix[bx][si] ; bx - смещение по строкам, si - по столбцам  inc dx  belowTheDiagonal:  add si, 2 ; Увеличиваем смещение по столбцу/счётчик столбцов  dec cx  cmp cx, 0  jne colLoop  add di, 2 ; Увеличиваем счётчик строк  add bx, col ; Увеличиваем смещение по строкам  add bx, col ; (дважды, так как размер каждого элемента - слово)  pop cx  dec cx  cmp cx, 0  jne rowLoop  div dl  mov ah, 0  mWriteStr med\_ariph  mWriteAX ; Макрос вывода значения регистра AX на экран  mWriteStr endl  pop dx ; Перенос сохранённых значений обратно в регистры  pop di  pop si  pop cx  pop bx  pop ax  endm mGetAverageveDiagonal  .DATA  buffer              db 20 dup(?)  buffnotnull         db 0 ; ячейка для хранения числа в первом задании  endl            db 13, 10, '$'  tab             db 09, '$'  space           db ' $'  inputSize       db 'Enter the size of matrix: $'  inputElements   db 'Enter matrix elements element by element: ', 13, 10, '$'  menuInstruction db 'To control the menu, press the corresponding key on the keyboard', 13, 10                  db '1. Enter matrix from keyboard', 13, 10                  db '2. Display matrix', 13, 10                  db '3. View transposed matrix', 13, 10                  db '4. Place ziro before no-ziro elems', 13, 10                  db '5. Test strings on increasing and decreasing', 13, 10                  db '6. Find the average of elements above the main diagonal', 13, 10                  db '0. Exit the program', 13, 10, '$'  matrix          db 03, 00, 'Matrix', 13, 10, '$'  transmatrix     db 03, 00, 'Tronspose matrix', 13, 10, '$'  inc\_str         db ' string - Increase', '$'  dec\_str         db ' string - Decrease', '$'  med\_ariph       db 'Average - ', '$'  rows                dw 1  cols                dw 1  currentMatrix       dw 100 dup (0)  transposedMatrix    dw 100 dup (0)  .CODE  Start:  mov ax, @data  mov ds, ax  mCLS 0000b ; Макрос очистки экрана и установки вида окна  mWriteStr menuInstruction ; Макрос вывода строки на экран  mWriteStr endl  menu: ; Вывод на экран меню, а также осуществление выбора следующего пункта программы      mov ah, 00h      int 16h ; Ожидание нажатия символа и получение его значения в al      cmp al, "0"      je exit      cmp al, "1"      je consoleInput      cmp al, "3"      je transposeMatrix      cmp al, "4"      je task1      cmp al, "5"      je task2      cmp al, "6"      je task3      writeMatrix: ; Вывод элементов матрицы на экран      mCLS 0000b ; Макрос очистки экрана и установки вида окна      mWriteStr menuInstruction ; Макрос вывода строки на экран      mWriteStr endl      mov ah, 02h      mov dx, 0900h      mov bh, 0      int 10h      mWriteStr matrix      mWriteMatrix currentMatrix, rows, cols      mov ah, 07h ; Задержка экрана      int 21h  jmp menu  consoleInput: ; Ввод элементов матрицы из консоли      mCLS 0000b ; Макрос очистки экрана и установки вида окна      mWriteStr inputSize ; Макрос вывода строки на экран      mReadAX buffer 2 ; Макрос ввода значения регистра AX с клавиатуры      mov rows, ax      mov cols, ax      mWriteStr endl ; Макрос вывода строки на экран      mWriteStr inputElements ; Макрос вывода строки на экран      mReadMatrix currentMatrix, rows, cols      jmp writeMatrix  transposeMatrix:; Получение и вывод транспонированной матрицы      mTransposeMatrix currentMatrix, rows, cols, transposedMatrix      mWriteStr endl      mWriteStr transmatrix      mWriteMatrix transposedMatrix, rows, cols      mov ah, 07h ; Задержка экрана      int 21h      jmp menu  task1: ; Перемещение нулей в начало строки      mZiroBeforeMatrix currentMatrix, rows, cols, transposedMatrix      mWriteStr endl      mWriteStr matrix      mWriteMatrix transposedMatrix, rows, cols      mov ah, 07h ; Задержка экрана      int 21h      jmp menu  task2: ; Проверка строк на монотонность      mTestIncrease currentMatrix, rows, cols      mov ah, 07h ; Задержка экрана      int 21h      jmp menu  task3: ; Получение среднего арифметического элементов выше диагоналей      mGetAverageAboveDiagonal currentMatrix, rows, cols      mov ah, 07h ; Задержка экрана      int 21h      jmp menu  exit: ; Завершение программы  mov ax, 4c00h  int 21h  end Start |

**Результат выполнения программы:**

****

****

****

****

**Вывод:** в ходе выполнения лабораторной работы были изучены особенности обработки двухмерных массивов на языке ассемблер.

**ЛИТЕРАТУРА**

**Основная литература**

1. Калашников О.А. Ассемблер- это просто. Учимся программировать [Текст] / О.А. Калашников.- СПб. БХВ-Петербург,2012.- 336 с.
2. Кирнос В. Н. Введение в вычислительную технику: основы организации ЭВМ и программирование на Ассемблере[Электронный ресурс]: учеб.пособие /В. Н. Кирнос. - Томск: Эль Контент, 2011. -172с. URL://biblioclub.ru/index.php?page=book\_red&id=208652

**Дополнительная литература**

1. Юров В. И. ASSEMBLER[Текст]. Учебник для вузов /В. И. Юров. 2-е изд.– Спб.:Питер 2010. – 637с.: ил.
2. Юров В. И. ASSEMBLER[Текст]. Практикум. / В. И. Юров. 2-е изд.–Спб.:Питер 2007. – 399 с.
3. Зубков С.В. ASSEMBLER для DOS, WINDOWS, UNIX [Текст] / С.В. Зубков-3-е изд., М.:ДМК Пресс; 2004. – 608 с.: ил.

**Электронные ресурсы:**

1. Научная электронная библиотека http://eLIBRARY.ru.
2. Электронно-библиотечная система http://e.lanbook.com.
3. Электронно-библиотечная система «Университетская библиотека онлайн» http://biblioclub.ru