|  |  |
| --- | --- |
| **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»***

**ЛАБОРАТОРНАЯ РАБОТА №7**

**«Команды передачи управления. Циклы»**

**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:** приобретение навыков написания программ с циклами на языке Ассемблер.

**Задача:** создать программу обработки числовых массивов используя циклические структуры и макросы для ввода и вывода десятичных чисел.

**Задание 1 (Вариант 10)**

1. Организовать ввод массива -15, -84, 76, 82, -4, 37, 29, 0, -12, 0 с клавиатуры.

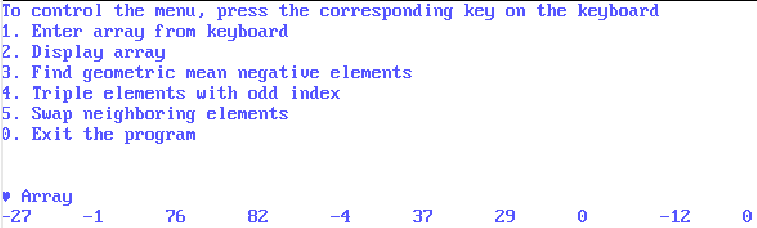
2. В заданном числовом массиве определить среднее арифметическое отрицательных элементов.

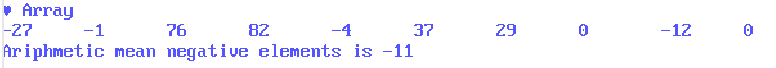
3. В заданном числовом массиве утроить элементы с нечетными индексами.

4. В заданном числовом массиве переставить местами соседние элементы с четными и нечетными индексами.

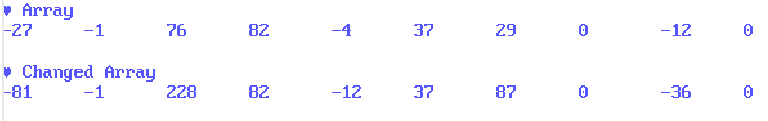
|  |
| --- |
| **.MODEL** small  **.STACK** 100h  .486 ; Включает сборку инструкций для процессора 80386  **mWriteStr** macro string  push ax ; Сохранение регистров, используемых в макросе, в стек  push dx  mov ah, 09h ; 09h - функция вывода строки на экран  mov dx, offset string  int 21h  pop dx ; Перенос 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 верхнего 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  **mReadArray** macro array, size\_ar  local colLoop  JUMPS ; Директива, делающая возможным большие прыжки  push bx ; Сохранение регистров, используемых в макросе, в стек  push cx  push si  mov cx, size\_ar  xor si, si ; Обнуляем смещение по столбцам  colLoop: ; Внутренний цикл, проходящий по столбцам  mReadAX buffer 4 ; Макрос ввода значения регистра AX с клавиатуры  mov array[si], ax  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop  pop si ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  NOJUMPS ; Прекращение действия директивы JUMPS  endm mReadArray  **mWriteArray** macro array, size\_ar  local colLoop  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push si  mov cx, size\_ar  xor si, si ; Обнуляем смещение по столбцам  colLoop: ; Внутренний цикл, проходящий по столбцам  mov ax, array[si] ; si - смещение по столбцам  mWriteAX ; Вывод текущего элемента массива  mWriteStr tab ; Вывод на экран табуляции, разделяющей элементы строки  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop  pop si ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  pop ax  endm mWriteArray  ; !!!!!!РАБОТААЕТ - НЕ ТРОГАЙ!!!!!!!!!  **mAverageNeg** macro array, size\_ar, chg\_array  local colLoop, not\_parity, next\_el  push bx ; Сохранение регистров, используемых в макросе, в стек  push cx  push si  push dx  mov cx, size\_ar  xor bx, bx  xor dx, dx  xor si, si ; Обнуляем смещение по столбцам  colLoop: ; Внутренний цикл, проходящий по столбцам  mov ax, array[si]  cmp ax, 0  jns next\_el  add bx, ax  inc dx  next\_el:  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop  mov ax, bx  idiv dl  mov ah, 0FFh  mWriteStr med\_ariph  mWriteAX  pop dx  pop si ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  endm mAverageNeg  **mTripleOdd** macro array, size\_ar, chg\_array  local colLoop, not\_parity, next\_el  push bx ; Сохранение регистров, используемых в макросе, в стек  push cx  push si  mov cx, size\_ar  xor si, si ; Обнуляем смещение по столбцам  colLoop: ; Внутренний цикл, проходящий по столбцам  mov ax, si  shr ax, 1  mov bx, 2h  div bl  cmp ah, 0  jnz not\_parity  mov ax, array[si]  mov bx, 3h  imul bl  mov chg\_array[si], ax  jmp next\_el  not\_parity:  mov ax, array[si]  mov chg\_array[si], ax  next\_el:  add si, 2 ; Переходим к следующему элементу (размером в слово)  loop colLoop  pop si ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  endm mTripleOdd  **mSwapNeigh** macro array, size\_ar, chg\_array  local colLoop  push ax ; Сохранение регистров, используемых в макросе, в стек  push bx  push cx  push si  mov cx, size\_ar  shr cx, 1  xor si, si ; Обнуляем смещение по столбцам  colLoop: ; Внутренний цикл, проходящий по столбцам  mov ax, array[si]  mov chg\_array[si+2], ax  mov ax, array[si+2]  mov chg\_array[si], ax  add si, 4 ; Переходим к следующему элементу (размером в слово)  loop colLoop  pop si ; Перенос сохранённых значений обратно в регистры  pop cx  pop bx  pop ax  endm SwapNeigh  **.DATA**  buffer db 20 dup(?)  endl db 13, 10, '$'  tab db 09, '$'  space db ' $'  inputElements db 'Enter 10 elements element by element: ', 13, 10, '$'  menuInstruction db 'To control the menu, press the corresponding key on the keyboard', 13, 10  db '1. Enter array from keyboard', 13, 10  db '2. Display array', 13, 10  db '3. Find geometric mean negative elements', 13, 10  db '4. Triple elements with odd index', 13, 10  db '5. Swap neighboring elements', 13, 10  db '0. Exit the program', 13, 10, '$'  str\_array db 03, 00, 'Array', 13, 10, '$'  str\_array\_chg db 03, 00, 'Changed Array', 13, 10, '$'  med\_ariph db 'Ariphmetic mean negative elements is ', '$'  array\_size dw 10  currentArray dw -27, -1, 76, 82, -4, 37, 29, 0, -12, 0  changedArray dw 10 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 task1  cmp al, "4"  je task2  cmp al, "5"  je task3  writearray: ; Вывод элементов массива 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 str\_array  mWriteArray currentArray, array\_size  mov ah, 07h ; Задержка экрана  int 21h  jmp menu  consoleInput: ; Ввод элементов массива из консоли  mCLS 0000b ; Макрос очистки экрана и установки вида окна  mWriteStr inputElements ; Макрос вывода строки на экран  mReadArray currentArray, array\_size  jmp writearray  task1: ; Среднее арифметическое  mAverageNeg currentArray, array\_size  mov ah, 07h ; Задержка экрана  int 21h  jmp menu  task2: ; Утроение  mTripleOdd currentArray, array\_size, changedArray  mWriteStr endl  mWriteStr str\_array\_chg  mWriteArray changedArray, array\_size  mov ah, 07h ; Задержка экрана  int 21h  jmp menu  task3: ; Обмен соседних  mSwapNeigh currentArray, array\_size, changedArray  mWriteStr endl  mWriteStr str\_array\_chg  mWriteArray changedArray, array\_size  mov ah, 07h ; Задержка экрана  int 21h  jmp menu  exit: ; Завершение программы  mov ax, 4c00h  int 21h  end Start |

**Результаты выполнения:**

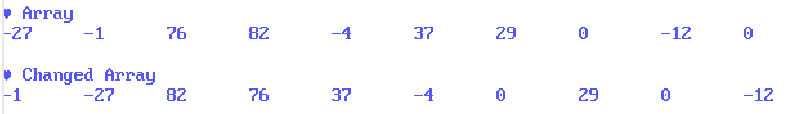
****

Выполнение первой задачи: ****

Выполнение второй задачи:

****

Выполнение третьей задачи:

****

**Вывод:** в ходе выполнения лабораторной работы были приобретены навыки написания программ с циклами на языке Ассемблер.