|  |  |
| --- | --- |
| **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 №6**

**«Создание макросов для ввода и вывод данных»**

**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:** Практическое овладение навыками разработки программного кода на языке Ассемблер. Изучение приемов разработки макроопределений.

**Задачи:**

1. Создать макросы для ввода и вывода чисел (двух, трех и четырехзначных).
2. С использованием макросов выполнить задание, соответствующее варианту.
3. Исходные данные вводятся с клавиатуры (k, c, d ).
4. Результаты выводятся на экран.

**Теоретическая часть** – словесный алгоритм макроса ввода-вывода.

Алгоритм ввода целых чисел:

* Ввод строки символов в буфер с клавиатуры, в результате чего получится символьное представление числа в ASCII коде;
* Преобразование строки символов в коде ASCII во внутреннее представление числа.

Алгоритм для вывода обратный.

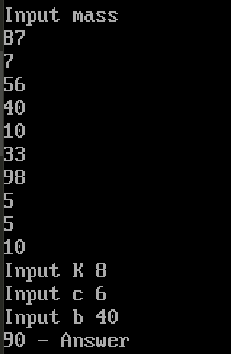
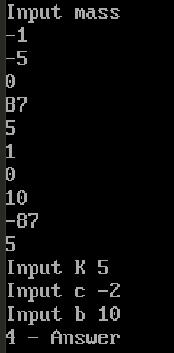
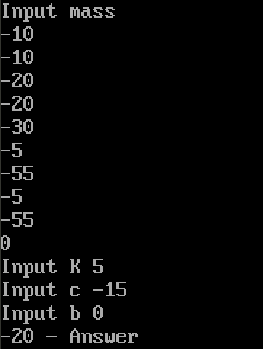
**Задание 1 (Вариант 15)**

Ввести с клавиатуры последовательность из N чисел, размером в слово. Найти сумму первых K элементов введенной последовательности удовлетворяющих условию: c<=a[i]<=d. Значение k, c, d задается с клавиатуры. Результат вывести на экран.

|  |  |
| --- | --- |
| .model small  .stack 100h  .data  buffer db 5 dup (0)  n db 10  mass dw 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ;10 dup (0)  k db 6  c dw 3  d dw 5 | str\_mass db "Input mass ", 10, 13, '$'  str\_inp\_k db "Input K ", '$'  str\_inp\_c db "Input c ", '$'  str\_inp\_d db "Input b ", '$'  str\_fin db " - Answer", '$'  .code |
| 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, 0Dh  ; int 21h ; Переводим каретку на новою строку  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 ; врзвращаемся 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  mov dx, offset str\_inp\_d  int 21h  mReadAX buffer, 8  mov d, ax  mov si, offset mass  mov bx, 0  mov ax, 0  mov cl, k  sum\_loop:  mov dx, ds:[si+bx]  cmp c, dx  jg next  cmp d, dx  jl next  add ax, dx  next:  add bx, 2  loop sum\_loop  mWriteAX  mov ah, 09h  mov dx, offset str\_fin  int 21h  mov ax, 4c00h  int 21h  end start | |

|  |  |
| --- | --- |
| start:  mov ax, @data  mov ds, ax  mov ax, 0  mov ah, 09h  mov dx, offset str\_mass  int 21h  mov cl, n  mov si, offset mass  input\_mass:  mReadAX buffer, 8  mov ds:[si+bx], ax  add bx, 2  loop input\_mass  mov ah, 09h  mov dx, offset str\_inp\_k  int 21h  mReadAX buffer, 8  mov k, al  mov ah, 09h  mov dx, offset str\_inp\_c  int 21h  mReadAX buffer, 8  mov c, ax | mov ah, 09h  mov dx, offset str\_inp\_d  int 21h  mReadAX buffer, 8  mov d, ax  mov si, offset mass  mov bx, 0  mov ax, 0  mov cl, k  sum\_loop:  mov dx, ds:[si+bx]  cmp c, dx  jg next  cmp d, dx  jl next  add ax, dx  next:  add bx, 2  loop sum\_loop  mWriteAX  mov ah, 09h  mov dx, offset str\_fin  int 21h  mov ax, 4c00h  int 21h  end start |

**Результаты выполнения:**

** ** 

**Вывод:** в ходе выполнения лабораторной работы было проведено практическое овладение навыками разработки программного кода на языке Ассемблер. Изучение приемов разработки макроопределений.