|  |  |
| --- | --- |
| **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 №4**

**«Использование команд условного перехода»**

**ДИСЦИПЛИНА: «Системное программирование»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-32Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_Зудин Д.В.\_\_\_\_\_)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_Амеличева К.А.\_\_\_)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2022 г.  **Цель**: практическое овладение навыками разработки программного кода на языке Ассемблер. Изучение команд условного и безусловного перехода. Исследование организации переходов.  **Постановка задачи**:  Разработать программу, использующую операторы передачи управления и приёмы программирования арифметических выражений, содержащих ветвления.  **Вариант №18**  **Формулировка задания №1**    **Листинг программы для задания №1**  mWriteStr macro string ;Макрос вывода строки  push ax ;Данные в стек  push dx  mov ah, 09h ;Вывод символа на консоль  mov dx, offset string ;Поместить в регистр dx адрес string  int 21h ;Прерывание DOS  pop dx ;Данные из стека  pop ax  ENDM  mReadAX10 macro buffer, sizee ;Макрос ввода 10-чного числа в регистр АХ  local input, startOfConvert, endOfConvert  push bx ;Данные в стек  push cx  push dx  input:  mov [buffer], sizee ;Задаём размер буфера  mov dx, offset [buffer];Поместить в регистр dx строку по адресу buffer  mov ah, 0Ah ;Чтение строки из консоли  int 21h ;Прерывание DOS  mov ah, 02h ;Вывод символа на экран  mov dl, 0Dh ;Перевод каретки на новую строку  int 21h ;Прерывание DOS  mov ah, 02h ;Вывод символа на экран  mov dl, 0Ah ;Чтение строки из консоли  int 21h ;Прерывание DOS  xor ah, ah ;Очистка регистра ah  cmp al, [buffer][1] ;Проверка на пустую строку  jz input ;Переход, если строка пустая  xor cx, cx ;Очистка регистра cx  mov cl, [buffer][1] ;инициализация переменной-счётчика  xor ax, ax ;Очистка регистра ax  xor bx, bx ;Очистка регистра bx  xor dx, dx ;Очистка регистра dx  mov bx, offset [buffer][2] ;Поместить начало строки в регистр bx  cmp [buffer][2], '-' ;Проверка на знак числа  jne startOfConvert ;Переход, если число неотрицательное  inc bx ;Инкремент регистра bx  dec cl ;Декремент регистра-счетчика cl  startOfConvert:  mov dx, 10 ;Поместить в регистр ax число 10  mul dx ;Умножение на 10 перед сложением с младшим разрядом  cmp ax, 8000h ;Проверка числа на выход за границы  jae input ;Переход, если число выходит за границы  mov dl, [bx] ;Поместить в регистр dl следующий символ  sub dl, '0' ;Перевод его в числовой формат  add ax, dx ;Прибавляем его к конечному результату  cmp ax, 8000h ;Проверка числа на выход за границы  jae input ;Переход, если число выходит за границы  inc bx ;Переход к следующему символу  loop startOfConvert ;Цикл  cmp [buffer][2], '-' ;Проверка на знак числа  jne endOfConvert ;Переход, если число неотрицательное  neg ax ;Инвертирование числа  endOfConvert:  pop dx ;Данные из стека  pop cx  pop bx  endm  mWriteAX10 macro ;Макрос вывода 10-чного числа из регистра AX  local convert, write  push ax ;Данные 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 ;Проверка числа на ноль  jns convert ;Переход, если число положительное  push ax ;Регистр ax в стек  mov dx, '-' ;Поместить в регистр dx символ '-'  mov ah, 02h ;Вывод символа на экран  int 21h ;Прерывание DOS  pop ax ;Регистр ax из стека  neg ax ;Инвертирование отрицательного числа  convert:  xor dx, dx ;Очистка регистра dx  div cx ;После деления dl = остатку от деления ax на cx  add dl, '0' ;Перевод в символьный формат  inc di ;Увеличение количества цифр в числе на 1  push dx ;Регистр dx в стек  or ax, ax ;Проверка числа на ноль  jnz convert ;Переход, если число не равно нулю  write:  pop dx ;dl = очередной символ  mov ah, 02h ;Вывод символа на экран  int 21h ;Прерывание DOS  dec di ;Повторение, пока di != 0  jnz write  pop di ;Данные из стека  pop dx  pop cx  pop bx  pop ax  endm  .model small ;Количество сегментов - 2  .stack 100h ;Размер стека - 256 б  .data ;Сегмент данных  mes\_a db 'Enter the number a: ', '$'  mes\_x db 'Enter the number x: ', '$'  mes\_y1 db 13, 10, 'Number y1: ', '$'  mes\_y2 db 13, 10, 'Number y2: ', '$'  mes\_y db 13, 10, 'Result y: ', '$'  buf db ? ;Буфер для ввода числа с клавиатуры  a dw ? ;Переменная а  x dw ? ;Переменная х  y1 dw ? ;Переменная у1  y2 dw ? ;Переменная у2  y dw ? ;Результат у  .code ;Сегмент кода  start:  mov ax, @data ;Поместить адрес сегмента данных  mov ds, ax ;в регистр ds  mov ax, 0 ;Обнулить регистр ax  mWriteStr mes\_a ;Введите число a  mReadAX10 buf, 5 ;Ввод числа а в регистр ax  mov bx, ax ;Поместить в переменную a значение регистра ax  mWriteStr mes\_x ;Введите число х  mReadAX10 buf, 5 ;Ввод числа x в регистр ax  mov cx, ax ;Поместить в переменную x значение регистра ax  mov a, bx  mov x, cx  cmp x, 3 ;Сравнение числа x с 3  jl l1 ;Переход, если x < 3  mov ax, a ;Поместить в регистр ax значение переменной а  cmp ax, 0 ;Сравнение числа a с 0  jS l2 ;Переход, если SF = 1 (a < 0)  add ax, x ;Поместить в регистр ax результат а + x  jmp l3  l2:  neg ax ;Инвертировать регистр ax  add ax, x ;Поместить в регистр ax результат а + x  jmp l3  l1:  mov ax, x ;Поместить в регистр ax значение переменной а  add ax, 7 ;Поместить в регистр ax результат  l3:  mWriteStr mes\_y1 ;Вывод y1  mWriteAX10  mov y1, ax ;Поместить в переменную y1 значение регистра ax  cmp x, 5 ;Сравнение числа x с 5  jG l4 ;Переход, если x > 5  mov ax, a ;Поместить в регистр ax значение переменной а  add ax, x ;Поместить в регистр ax результат а + x  jmp l5 ;Безусловный переход  l4:  mov ax, 1 ;Поместить в регистр ax число 1  l5:  mWriteStr mes\_y2 ;Вывод y2  mWriteAX10  mov dx, 0  mov bx, y1 ;Поместить в регистр bx значение переменной y1  xchg ax, bx ;Поменять местами регистры ax и bx (y1 = ax, y2 = bx)  idiv bx ;y1 / y2 (остаток в dx)  mov y, dx ;Поместить в переменную y значение регистра dx  mov ax, y ;Поместить в регистр ax значение переменной у  mWriteStr mes\_y ;Вывод y  mWriteAX10  mov ax, 4c00h ;Завершение  int 21h ;программы  end start ;Закрыть программу  **Результат выполнения программы для задания №1**            **Формулировка задания №2**  Разработать приложение, формирующее на экране меню – элемент пользовательского интерфейса, позволяющий при нажатии на клавишу 1, 2, 0 вывести на экран соответствующее сообщение.  **Листинг программы для задания №2**  mWriteStr macro string ;Макрос вывода строки  push ax ;Данные в стек  push dx  mov ah, 09h ;Вывод символа на консоль  mov dx, offset string ;Поместить в регистр dx адрес string  int 21h ;Прерывание DOS  pop dx ;Данные из стека  pop ax  ENDM  .model small ;Количество сегментов - 2  .stack 100h ;Размер стека - 256 б  .data ;Сегмент данных  menu db '1 - Print hello', 13, 10  db '2 - Print go away', 13, 10  db '0 - Exit', 13, 10, '$'  select db 13, 10, 'Select > $'  hello db 13, 10, 'Hello!', 13, 10, 13, 10, '$'  go\_away db 13, 10, 'Go away!', 13, 10, 13, 10, '$'  .code ;Сегмент кода  start:  mov ax, @data ;Поместить адрес сегмента данных  mov ds, ax ;в регистр ds  mWriteStr menu ;Вывод меню  select\_loop:  mWriteStr select ;Вывод выбора  mov ah, 01h ;Ввод символа с клавиатуры  int 21h ;Прерывание DOS  cmp al, '1' ;Сравнение введённого символа с '1'  je c1 ;Переход, если равно  cmp al, '2' ;Сравнение введённого символа с '2'  je c2 ;Переход, если равно  cmp al, '0' ;Сравнение введённого символа с '0'  je exit ;Переход, если равно  jmp select\_loop ;Бузусловный переход  c1:  mWriteStr hello ;Вывод сообщения  jmp start ;Переход на start  c2:  mWriteStr go\_away ;Вывод сообщения  jmp start ;Переход на start  exit:  mov ax, 4c00h ;Завершение  int 21h ;программы  end start ;Закрыть программу  **Результат выполнения программы для задания №2**    **Формулировка задания №3**  Разработать приложение, выполняющее по запросу пользователя перевод полученной студентом балльной оценки из числовой в словесную форму.  **Листинг программы для задания №3**  mWriteStr macro string ;Макрос вывода строки  push ax ;Данные в стек  push dx  mov ah, 09h ;Вывод символа на консоль  mov dx, offset string ;Поместить в регистр dx адрес string  int 21h ;Прерывание DOS  pop dx ;Данные из стека  pop ax  ENDM  .model small ;Количество сегментов - 2  .stack 100h ;Размер стека - 256 б  .data ;Сегмент данных  mes db 'Enter 6 for exit', 13, 10  db 'Enter an assessment: ', '$'  er db 13, 10, 'An invalid rating has been entered!', 13, 10, 13, 10, '$'  zer db 13, 10, 'Zero!', 13, 10, 13, 10, '$'  one db 13, 10, 'One!', 13, 10, 13, 10, '$'  two db 13, 10, 'Two!', 13, 10, 13, 10, '$'  thr db 13, 10, 'Three!', 13, 10, 13, 10, '$'  for db 13, 10, 'Four!', 13, 10, 13, 10, '$'  fiv db 13, 10, 'Five!', 13, 10, 13, 10, '$'  .code ;Сегмент кода  start:  mov ax, @data ;Поместить адрес сегмента данных  mov ds, ax ;в регистр ds  prog\_loop:  mWriteStr mes ;Вывод message  mov ah, 01h ;Ввод символа с клавиатуры  int 21h ;Прерывание DOS  cmp al, '0' ;Сравнение введённого символа с '0'  je c1 ;Переход, если равно  cmp al, '1' ;Сравнение введённого символа с '1'  je c2 ;Переход, если равно  cmp al, '2' ;Сравнение введённого символа с '2'  je c3 ;Переход, если равно  cmp al, '3' ;Сравнение введённого символа с '3'  je c4 ;Переход, если равно  cmp al, '4' ;Сравнение введённого символа с '4'  je c5 ;Переход, если равно  cmp al, '5' ;Сравнение введённого символа с '5'  je c6 ;Переход, если равно  cmp al, '6' ;Сравнение введённого символа с '6'  je exit ;Переход, если равно  mWriteStr er ;Вывод ошибки  jmp prog\_loop ;Повторение цикла  c1:  mWriteStr zer ;Вывод сообщения  jmp prog\_loop ;Повторение цикла  c2:  mWriteStr one ;Вывод сообщения  jmp prog\_loop ;Повторение цикла  c3:  mWriteStr two ;Вывод сообщения  jmp prog\_loop ;Повторение цикла  c4:  mWriteStr thr ;Вывод сообщения  jmp prog\_loop ;Повторение цикла  c5:  mWriteStr for ;Вывод сообщения  jmp prog\_loop ;Повторение цикла  c6:  mWriteStr fiv ;Вывод сообщения  jmp prog\_loop ;Повторение цикла  exit:  mov ax, 4c00h ;Завершение  int 21h ;программы  end start ;Закрыть программу  **Результат выполнения программы для задания №3**      **Выводы:**  В ходе выполнения работы были изучены команды условного и безусловного перехода на языке Ассемблер; исследована их организация. | | |
|  | | |