

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (ДГТУ)**

Факультет Информатика и вычислительная техника

(наименование факультета)

Кафедра Кибербезопасность информационных систем

(наименование кафедры)

**ОТЧЕТ**

по лабораторным работам по предмету: Аппаратные средства вычислительной техники

Выполнил:

ст. гр. ВКБ31 Котелевец К.А.

Ростов-на-Дону

2024.

**ЛАБОРАТОРНАЯ РАБОТА №1**

# Применение логических инструкций

Задание: Дан массив из 6 байт. Рассматривая его, как массив из 48 бит,

посчитать в нём количество нулей.

Ниже представлен листинг программы моего задания.

Листинг программы:

|  |
| --- |
| ans db 0  mass db 01101010b, 10000010b, 01011001b, 00111100b, 01110111b, 10100101b  mov si, offset mass  mov cx, 6\*8  mov al, 1  beg: test [si], al  jnz @1  inc ans  @1: rol al, 1  adc si, 0  loop beg |

Контрольные вопросы:

1. Отличие команд test и and?

Команда test:

Не изменяет значение операнда

Устанавливает флагы ZF, SF, OF, PF, AF в соответствии с результатом сравнения

Команда and:

Изменяет значение операнда, выполняя логическое AND

Не устанавливает флагов ZF, SF, OF, PF, AF

2. Как сбросить 5-й бит переменной байта ВВ?

Для сброса 5-го бита можно использовать команду and с маской:

and al, 11101111b ; сбрасывает 5-й бит

3. Как установить 5-й бит переменной байта ВВ?

Для установки 5-го бита можно использовать команду or с маской:

or al, 00100000b ; устанавливает 5-й бит

4. Как инвертировать 5-й бит переменной байта ВВ?

Для инвертирования 5-го бита можно использовать команду xor:

xor al, 00100000b ; инвертирует 5-й бит

5. Как проверить установлен ли 5-й бит переменной байта ВВ?

Для проверки можно использовать команду test:

test al, 00100000b ; проверяет 5-й бит

6. Как проверить четным или нечетным является количество установленных бит в байте?

mov AL, 0AAh ; 0b10101010

mov CX, 8

xor BX, BX

LL:

shl AL, 1

jnc NO

inc BX

NO:

loop LL

LL: shl AL, 1

Инструкция shl AL, 1 выполняет логический сдвиг влево на 1 бит для регистра AL. Если самый левый бит (бит 7) был 1, то флаг переноса (CF) будет установлен.

jnc NO

Инструкция jnc NO выполняет переход к метке NO, если флаг переноса (CF) не установлен (т.е. если самый левый бит был 0).

inc BX

Если флаг переноса установлен (т.е. самый левый бит был 1), выполняется инструкция inc BX, которая увеличивает значение регистра BX на 1.

NO: loop LL

Метка NO обозначает точку, куда происходит переход, если флаг переноса не установлен.

Инструкция loop LL уменьшает значение регистра CX на 1 и выполняет переход к метке LL, если CX не равен нулю. Таким образом, цикл выполняется 8 раз.

7. Флаги условий, модифицирующиеся после выполнения команд and, or, xor?

ZF (Zero Flag): Устанавливается, если результат операции равен нулю.

PF (Parity Flag): Устанавливается, если количество единичных битов в младшем байте результата четно.

SF (Sign Flag): Устанавливается, если старший бит результата равен 1 (отрицательное число).

CF (Carry Flag): Сбрасывается (устанавливается в 0).

OF (Overflow Flag): Сбрасывается (устанавливается в 0).

AF (Auxiliary Carry Flag): Не модифицируется.

8. Основное отличие команд логических и арифметических сдвигов?

Логические сдвиги:

Не меняют знак результата

Арифметические сдвиги:

Меняют знак результата

9. Максимальное число двоичных разрядов, на которое можно сдвинуть операнд с помощью одной команды сдвига?

В emu8086 максимально возможный сдвиг составляет 31 бит. Это ограничено размером регистра EAX.

**ЛАБОРАТОРНАЯ РАБОТА №2**

# Обработка символьной информации с помощью функций DOS

Задание: ввести с клавиатуры две строки. Сравнить их. Вывести на экран номер начала второй строки в первой.

Ниже представлен листинг программы.

Листинг программы:

|  |
| --- |
| .model small  .stack 100h  .data  str1 db 100 dup('$') ; First string  str2 db 100 dup('$') ; Second string  prompt1 db 'Enter first string: $'  prompt2 db 'Enter second string: $'  result db 'Position: $'  not\_found\_msg db 'Not found$'  .code  main proc  mov ax, @data  mov ds, ax  ; Input first string  mov ah, 9  lea dx, prompt1  int 21h  mov ah, 0Ah  lea dx, str1  mov byte ptr [str1], 98  int 21h  ; New line  mov ah, 2  mov dl, 0Dh  int 21h  mov dl, 0Ah  int 21h  ; Input second string  mov ah, 9  lea dx, prompt2  int 21h  mov ah, 0Ah  lea dx, str2  mov byte ptr [str2], 98  int 21h  ; New line  mov ah, 2  mov dl, 0Dh  int 21h  mov dl, 0Ah  int 21h  ; Compare strings  lea si, str1 + 2; Skip two service bytes  lea di, str2 + 2  mov cx, 1; Initial position  compare\_loop:  mov al, [si]  cmp al, '$'  je not\_found\_label  push si  push di  push cx  check\_match:  mov al, [si]  mov bl, [di]  cmp bl, '$'  je found  cmp al, bl  jne no\_match  inc si  inc di  jmp check\_match  no\_match:  pop cx  pop di  pop si  inc si  inc cx  jmp compare\_loop  found:  pop cx  pop di  pop si  ; Output result  mov ah, 9  lea dx, result  int 21h  mov ax, cx  call print\_num  jmp exit  not\_found\_label:  mov ah, 9  lea dx, not\_found\_msg  int 21h  exit:  mov ah, 4Ch  int 21h  main endp  ; Subroutine for printing a number  print\_num proc  mov cx, 0  mov bx, 10  divide\_loop:  xor dx, dx  div bx  push dx  inc cx  test ax, ax  jnz divide\_loop  print\_loop:  pop dx  add dl, '0'  mov ah, 2  int 21h  loop print\_loop  ret  print\_num endp  end main |

Структура:

1. Сегмент данных (.data):
   * Определяет две строки (str1 и str2) длиной 100 байт каждая, инициализированные символом '$'.
   * Содержит приглашения для ввода строк и сообщения для вывода результата.
2. Основная процедура (main proc):
   * Инициализация сегмента данных. -Ввод первой строки:
     + Выводит приглашение.
     + Использует прерывание DOS для ввода строки.
     + Добавляет перевод строки послеввода.
   * Ввод второй строки (аналогично первой).
3. Алгоритм сравнения строк:
   * Инициализирует указатели на начало строк (пропуская служебные байты).
   * Использует регистр CX как счетчик позиции.
   * Цикл сравнения (compare\_loop):
     + Проверяет конец первой строки.
     + Сохраняет текущее состояние в стеке.
     + Вложенный цикл (check\_match) для посимвольного сравнения.- Обработка несовпадения (no\_match):
     + Восстанавливает состояние из стека.
     + Переходит к следующему символу первой строки.
   * Обработка совпадения (found):
     + Выводит позицию начала совпадения.
4. Вывод результата:
   * Если найдено совпадение, выводит номер позиции.
   * Если совпадение не найдено, выводит сообщение "Not found".
5. Подпрограмма печати числа (print\_num proc):
   * Преобразует число в десятичное представление.
   * Использует стек для хранения цифр.
   * Выводит цифры по одной, начиная с наиболее значимой.
6. Завершение программы:
   * Использует прерывание DOS для корректного завершения.

Контрольные вопросы:

1. Что такое программное прерывание?

Программное прерывание - это механизм, позволяющий прервать выполнение текущей программы и перейти к обработчику прерывания. Это позволяет реагировать на внешние события или внутренние состояния системы.

1. Какие возможности работы с клавиатурой имеются у программиста?

Программист имеет следующие возможности работы с клавиатурой через DOS API:

* + Чтение символа из клавиатуры (функция 08h)
  + Очистка буфера клавиатуры (функция 0Eh)
  + Включение/выключение автозагрузки клавиатуры (функции 0Dh и 0Ch)

1. Чем отличаются друг от друга различные функции DOS? выполняющие ввод с клавиатуры?

Различные функции DOS для ввода с клавиатуры имеют следующее различие:

* + Функция 08h читает один символ
  + Функция 0Ah читает строку до символа переноса строки или EOF

1. Как работает команда xlat?

Команда xlat выполняет побитовое И с содержимым регистра AL и текущего байта в памяти.

1. Можно ли выдать на экран текст ‘$1.00=25,00 rub/’, используя функции DOS 09h?

Да, можно вывести текст '$1.00=25,00 rub/'. Функция DOS 09h выводит символы по отдельности.

1. Какие режимы адресации удобно использовать при работе с одномерными массивами?

Для работы с одномерными массивами удобно использовать следующие режимы адресации:

* + Индексный: [array + index]
  + Смещенный: array[index]
  + Смещенный относительно базового индекса: array[index + base]

1. Выражения в поле операндов означают:
   * lea bx, IN\_STR+2: загрузить адрес в bx, увеличив его на 2
   * mov OUT\_STR+3, al: переместить значение из al в OUT\_STR+3
2. Как выделить старшую тетраду байта?

Старшую тетрадку байта можно выделить с помощью инструмента отладчика.

1. Как выделить младшую тетраду байта?

Младшую тетрадку байта нельзя выделить напрямую. Можно выделить половину слова (2 бита).

1. Команда lea загружает адрес в регистре, а mov загружает значение из памяти в регистр.

**ЛАБОРАТОРНАЯ РАБОТА №3**

# Подпрограммы

Задание: разработать подпрограмму, которая преобразует заданное десятичное число в двоичную систему. Разработать программу, которая вводит с клавиатуры строку десятичных цифр и выводит на экран её эквивалент в двоичной системе. Если строка не является числом, то сообщает об этом.

Ниже представлен листинг программы.

Листинг программы:

|  |
| --- |
| .model small  .data  buff db 6,7 dup(?) ; Buffer for user input  error db "incorrect number$", 0Dh, 0Ah ; Error message  msg db "Enter a number: ", 0Dh, 0Ah ; Prompt message  space db " " ; Space character  len equ $-msg ; Length of prompt message  .code  InputInt proc  mov ah,09h ; Function to display string  lea dx, msg ; Load address of prompt message  mov ah,0ah ; Function to read string  lea dx, buff ; Load address of buffer  int 21h ; Call DOS service  ;space  mov dl, ' '  mov ah, 02h  int 21h  mov si, offset buff+2 ; Point to the first character of input  cmp byte ptr [si], "-" ; Check if the first character is '-'  jnz ii1  mov di, 1 ; Set sign flag to 1 if negative  inc si ; Move to the next character  ii1:  xor ax, ax ; Clear ax register  mov bx, 10 ; Set base to 10 for multiplication  ii2:  mov cl, [si] ; Load character into cl  cmp cl, 0dh ; Check if end of string (carriage return)  jz endin  cmp cl, '0' ; Check if character is a digit (0-9)  jb er  cmp cl, '9'  ja er  sub cl, '0' ; Convert ASCII digit to its actual value  mul bx ; Multiply ax by 10  add ax, cx ; Add the converted digit to ax  inc si ; Move to the next character  jmp ii2 ; Jump back to the start of the loop  er:  mov dx, offset error ; Load address of error message  mov ah, 09h ; Function to display string  int 21h ; Call DOS service  int 20h ; Terminate program  endin:  cmp di, 1 ; Check if number was negative  jnz ii3  neg ax ; Negate ax if the number was negative  ii3:  ret ; Return to caller  InputInt endp  OutBin proc  mov bx, ax ; Copy value to bx  mov cx, 16 ; Set counter to 16  ob1:  shl bx, 1 ; Shift bx left by 1 bit  jc ob2 ; Jump if carry set (bit was 1)  mov dl, '0' ; If bit was 0, set dl to '0'  jmp ob3  ob2:  mov dl, '1' ; If bit was 1, set dl to '1'  ob3:  mov ah, 2 ; Function to display character  int 21h ; Call DOS service  loop ob1 ; Decrement cx and jump back if cx != 0  mov dl, ' '  mov ah, 02h  int 21h  OutBin endp  start:  call InputInt ; Call input function  call OutBin ; Call output function  mov ah, 4ch ; Function to terminate program  int 21h ; Call DOS service  end start |

Контрольные вопросы:

1. Ближние подпрограммы - это те, которые находятся ближе к текущему месту выполнения программы. Дальние подпрограммы - это те, которые находятся дальше.
2. В данной программе используется только ближний вариант команды call. Это видно из того, что вызовы подпрограмм InputInt и OutBin происходят непосредственно после их объявления.
3. Основные способы передачи параметров в подпрограммы:
   * Через регистры (например, ax, bx)
   * Через стек (используется автоматически при вызове подпрограммы)
   * Через глобальные переменные
4. Да, массив может быть параметром подпрограммы. Однако нужно учитывать ограничения размера данных в ассемблее.
5. Да, можно использовать регистр sp вместо bp для чтения параметров из стека. Это часто предпочтительнее, так как sp обновляется автоматически при возврате из подпрограммы.
6. Операндом команды ret является количество байт, которое нужно удалить из стека.
7. Команда "ret 8" эквивалентна следующей последовательности:

|  |
| --- |
| mov sp, bp  ret |

**ЛАБОРАТОРНАЯ РАБОТА №4**

# Обработка прерываний

Задание: очистить экран. Вывести несколько строк произвольного текста (атрибут 14). Перехватив прерывание печати экрана Print Screen (Int 5h), менять атрибуты всех строк экрана циклически от 1 до 15. Каждое нажатие клавиши Print Screen вызывает изменение атрибута.

# Листинг программы:

|  |
| --- |
| ; Data segment  data segment  TEXT1 db 'This is the first line of text', '$'  TEXT2 db 'Here is the second line', '$'  TEXT3 db 'And this is the third line', '$'  TEXT4 db 'Press ESC to change colors', '$'  OLD\_CS dw ? ; Old CS register value  OLD\_IP dw ? ; Old IP register value  CURRENT\_ATTR db 14 ; Starting attribute (yellow)  data ends  ; Code segment  code segment  assume cs:code, ds:data  NEW\_27 proc far ; new prerivanie int 5h  push ax  push bx  push cx  push dx  push ds  push es  ; Set up video memory address  mov ax, 0B800h  mov es, ax  ; Increment attribute, loop back to 1 if it reaches 16  inc [CURRENT\_ATTR]  cmp [CURRENT\_ATTR], 16  jne skip\_reset  mov [CURRENT\_ATTR], 1  skip\_reset:  ; Change attribute for all characters on screen  mov cx, 2000 ; 80x25 characters  xor di, di ; Start from the beginning of video memory  mov ah, [CURRENT\_ATTR]  change\_attr:  mov al, es:[di] ; Get character  stosw ; Store character with new attribute  loop change\_attr  ; Restore saved registers  pop es  pop ds  pop dx  pop cx  pop bx  pop ax  iret  NEW\_27 endp  CLS proc near  push cx  push ax  push di  mov ax, 0B800h  mov es, ax  xor di, di  mov ax, 0720h ; Space character with black background  mov cx, 2000 ; 80x25 characters  rep stosw  pop di  pop ax  pop cx  ret  CLS endp  PRINT\_TEXT proc near  push si  push di  push cx  print\_loop:  lodsb  stosw  loop print\_loop  pop cx  pop di  pop si  ret  PRINT\_TEXT endp  start:  mov ax, data  mov ds, ax  ; Save old interrupt vector  mov ah, 35h  mov al, 27h ; Changed to 27h for ESC key  int 21h  mov OLD\_IP, bx  mov OLD\_CS, es  ; Set new interrupt vector  push ds  mov dx, offset NEW\_27  mov ax, seg NEW\_27  mov ds, ax  mov ah, 25h  mov al, 27h ; Changed to 27h for ESC key  int 21h  pop ds  ; Clear screen  call CLS  ; Set up video memory  mov ax, 0B800h  mov es, ax  mov ah, 14 ; Yellow attribute  ; Print TEXT1  mov si, offset TEXT1  mov di, 160 ; Second row  mov cx, 29 ; Length of TEXT1  call PRINT\_TEXT  ; Print TEXT2  mov si, offset TEXT2  mov di, 480 ; Fourth row  mov cx, 24 ; Length of TEXT2  call PRINT\_TEXT  ; Print TEXT3  mov si, offset TEXT3  mov di, 800 ; Sixth row  mov cx, 27 ; Length of TEXT3  call PRINT\_TEXT  ; Print TEXT4  mov si, offset TEXT4  mov di, 1120 ; Eighth row  mov cx, 28 ; Updated length of TEXT4  call PRINT\_TEXT  wait\_key:  mov ah, 0  int 16h    cmp al, 48 ; ASCII code for '0'  je quit    jmp change\_color    change\_color:  int 27h ; Call our new interrupt handler  jmp wait\_key  quit:  ; Restore old interrupt vector  mov dx, OLD\_IP  mov ax, OLD\_CS  mov ds, ax  mov ah, 25h  mov al, 27h ; Changed to 27h for ESC key  int 21h  mov ax, 4C00h  int 21h  code ends  end start |

Контрольные вопросы:

1. В чём суть концепции прерывания?  
   Прерывание - это механизм, позволяющий процессору временно прекратить выполнение текущей инструкции и перейти к обработке важной события или запроса извне. Это позволяет операционной системе реагировать на различные события без потери производительности основного процесса.
2. Как работает система прерывания по вектору?  
   Система прерывания по вектору использует таблицу прерываний, где каждый элемент содержит адрес обработчика соответствующего типа прерывания. При возникновении прерывания, процессор сохраняет состояние и переходит к указанному в таблице обработчику.
3. В чём отличие команд **ret** и **iret**?  
   ret - возврат из процедуры, сохраняет только регистры eax, ebx, ecx, edx. iret - возврат после прерывания, восстанавливает все регистры, включая esi, edi, ebp, esp.
4. Какие способы получения/изменения вектора прерывания Вы знаете?

Использование BIOS-инструкций (INT 21h, INT 15h)

Программное изменение таблицы прерываний через API ОС

Настраивание аппаратного обеспечения

1. Как вызвать программное прерывание?  
   Для вызова программного прерывания используется команда int n, где n - номер прерывания (например, int 21h для DOS).
2. Какие существуют способы передачи параметров в подпрограмму обработки прерываний и возврата параметров из неё?

Через стек (наименее эффективно)

Через регистры процессора

Через специальные структуры данных (например, EAX, EBX, ECX, EDX)

1. Какие действия производит процессор при получения запроса на прерывание?

Сохранение состояния процессора

Переход к обработчику прерывания

Восстановление состояния после завершения обработки

1. В чём отличие команд **call** и **int**?

call - переход к процедуре, сохраняет состояние и передает управление. int - вызов программного прерывания, может изменить контекст выполнения.