МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4

по дисциплине: Архитектура вычислительных систем

тема: «Команды передачи управления»

Выполнил: ст. группы ПВ-223

Дмитриев Андрей Александрович

Проверил:

Осипов Олег Васильевич

Белгород 2024 г.

**Цель работы:** изучение команд перехода для организации циклов и ветвлений, получение навыков создания процедур с аргументами.

**Задания для выполнения к работе**

1. Написать программу для вычисления значения арифметического выражения, используя команды условного и безусловного перехода согласно варианту задания. Подобрать набор тестовых данных (не менее 3). При выполнении операций с числами, преобразовывать их к 4-байтовым числам со знаком. Результат вывести на экран.
2. Написать программу для вычисления значения арифметического выражения, содержащего функцию. Вычисление функции организовать в виде отдельной подпрограммы по всем правилам, описанным выше. Для обработки массивов использовать команды для работы с циклами и команды условного перехода. Подобрать набор тестовых данных (не менее 3). Результат вывести на экран.

Задние варианта 2:

1.

|  |  |
| --- | --- |
|  | *x* – беззнаковое число размером один байт  *y*, *z* – знаковые числа размером 2 байта |

2.

|  |  |
| --- | --- |
|  | *h* – знаковая переменная размером один байт  *x* – массив 1-байтовых беззнаковых чисел  *y* – массив 2-байтовых беззнаковых чисел  *n* – беззнаковая переменная размером 2 байта |

**Задание 1.**

Реализация:

.686

.model flat, stdcall

option casemap: none

include windows.inc

include kernel32.inc

include msvcrt.inc

include user32.inc

includelib  kernel32.lib

includelib  msvcrt.lib

includelib user32.lib

.DATA

    x DB ?

    y DW ?

    z DW ?

    format db "Result: %d", 0Ah, 0

.CODE

START:

    ; записываем х и у и расширяем

    xor ebx, ebx        ; зануляем ebx

    mov bl, x           ; записываем беззнаковое число из x

    mov ax, y           ; ax = y

    cwde                ; расширение y до двойного слова

    xchg eax, ebx       ; обмен местами содержимого EAX и EBX

    cmp eax, 10

    jb cond1            ; беззнаковая проверка x < 10

    jmp func3           ; иначе (x >= 10)

cond1:

    cmp ebx, 2

    jl func1            ; знаковая проверка y < 2

    jmp func2           ; иначе (x >=2)

func1:

    add eax, ebx        ; складываем, получаем x + y

    push eax

    mov ax, z           ; ax = z

    cwde                ; расширение z до двойного слова

    sub eax, ebx        ; вычитаем, получаем z - y

    xchg eax, ebx       ; обмен местами содержимого EAX и EBX

    pop eax

    cdq                 ; Расширение делимого со знаком ax -> dx:ax

    idiv ebx            ; делим, получаем (x + y)/(z - y)

    jmp func\_end

func2:

    imul eax, ebx       ; умножаем, получаем x \* y

    sub eax, 6          ; умножаем, получаем x \* y - 6

    jmp func\_end

func3:

    add eax, ebx        ; складываем, получаем x + y

    imul eax, eax       ; умножаем, получаем (x+y)^2

    jmp func\_end

func\_end:

    ; Вывод результата в консоль

    push  eax

    push  offset format

    call  crt\_printf

    ; Завершаем программу

    push 0

    call ExitProcess

END START

Тестовые данные:

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | z | a |
| 6 | -2 | 2 | 1 |
| 6 | 4 | ? | 18 |
| 12 | -14 | ? | 4 |

**Задание 2.**

Реализация:

.686

.model flat, stdcall

option casemap: none

include windows.inc

include kernel32.inc

include msvcrt.inc

include user32.inc

includelib kernel32.lib

includelib msvcrt.lib

includelib user32.lib

.DATA

    h db ?

    x db ?

    y dw ?

    n dw ?

    format db "Result: %d", 0Ah, 0

.CODE

f proc

    push ebx

    push ecx ; 12+

    xor eax, eax

    mov ax, [esp+14]

    cmp eax, 1          ; x > 1

    ja f\_cond1

    jmp f\_func2

f\_cond1:

    cmp eax, 7          ; x < 7

    jl f\_func1

    jmp f\_func2

f\_func1:

    mov ebx, eax

    imul ebx, ebx       ; x^2

    push ebx

    xor ebx, ebx

    mov bx, [esp+16]

    mov ecx, eax

    mov eax, ebx

    dec ecx

    f\_l:

        imul ebx, eax   ; y^x

    loop f\_l

    pop eax

    sub eax, ebx        ; x^2 - y^x

    jmp f\_end\_func

f\_func2:

    push eax

    imul eax, eax       ; x^2

    xor ebx, ebx

    mov bx, [esp+16]

    push ebx

    imul ebx, ebx       ; y^2

    add eax, ebx

    pop ebx

    pop ecx

    imul ebx, ecx

    xchg eax, ebx

    push eax

    call crt\_abs        ; |xy|

    add eax, ebx

    pop ebx

    jmp f\_end\_func

f\_end\_func:

    pop ecx

    pop ebx

    ret 4

f endp

start:

    xor esi, esi

    ; Вычисление суммы

    movsx ecx, n

    mov ebx, 0 ; инициализация суммы

    mov eax, 0 ; для временного хранения результата

sum\_loop:

    push ebx

    ; Вычисляем i / h

    mov eax, esi

    movsx ebx, h

    mov edx, 0

    idiv ebx

    ; Умножаем 100 \* (i / h)

    mov ebx, 100

    imul eax, ebx

    ; Добавляем к сумме

    pop ebx

    add ebx, eax

    push ebx

    ; Вычисляем y[i] / x[i]

    movsx eax, y[esi\*2]

    mov edx, 0

    movsx ebx, x[esi]

    idiv ebx

    pop ebx

    ; Добавляем y[i] / x[i] к сумме

    add ebx, eax

    ; Вызываем функцию

    xor ax, ax

    mov al, x[esi]

    push ax

    push y[esi\*2]

    call f

    add ebx, eax

    inc esi

    loop sum\_loop

    ; Вывод результата в консоль

    push ebx

    push offset format

    call crt\_printf

    ; Завершаем программу

    push 0

    call ExitProcess

end start

Тестовые данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| h | x | y | n | res |
| -1 | 1,2,3,4 | 4,3,2,1 | 4 | -563 |
| -1 | 1,2,3,4 | 4,3,2,1 | 4 | -10560 |
| -1 | 1,2,3,4 | 4,3,2,1 | 4 | 37 |

**Вывод:** в ходе работы изучены команды перехода для организации циклов и ветвлений, получены навыки создания процедур с аргументами.