**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**  
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

Дисциплина «Архит. ВС»

Тема «Команды передачи управления»

Выполнил: ст. группы ПВ-31  
Ковалев Павел Александрович

Проверил: Осипов Олег Васильевич

**Белгород 2021**

**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 выполне- нии 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.

d

*x* – знаковое 2-байтовое

*y*, *z* – беззнаковое однобайтовое

*h* – знаковая переменная размером 1 байт

*x* – массив 1-байтовых знаковых чисел

*y*–массив беззнаковых 2-байтовых чисел

*m* – беззнаковая переменная размером 2 байта

**Выполнение работы**

№ Результат

|  |  |
| --- | --- |
| 1 0x12 0x20 0x40 | 641 |
| 2 0x20 0x50 0x20 | 2593 |
| 3 0x50 0x0 0x100 | 257 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | *ℎ* |  |  |  | Результат |
| 1 | 0x3 | [0x2,0x3] | [0x4,0x5] | 0x2 | -159452 |
| 2 | 0x10 | [0x9,0x8] | [0x1,0x6] | 0x2 | 318046 |
| 3 | 0x10 | [0x4,0x4] | [0x2,0x2] | 0x2 | 102064 |

.386

.model flat, stdcall

option casemap: none

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.DATA

x DW 020h

y DB 050h

z DB 020h

format db "a = %d", 0

.CODE

start:

xor eax,eax

xor ecx,ecx

xor ebx,ebx

xor edx,edx

;проверка y>=0 -> z+1+xy

mov al,0

cmp y,al

jae case\_Y

;проверка z<-1 -> x^2\*y^3\*z^4

mov al,-1

cmp z,al

jb case\_Z\_not\_in

;проверка z>1 -> x^2\*y^3\*z^4

mov al,1

cmp z,al

ja case\_Z\_not\_in

;если 3 перехода не сработали переходим на ветку 4x-2z

jmp case\_Z\_in

case\_Y:

;ebx = x

mov ax,x

cwde

mov ebx,eax

;eax = y

mov al,y

cbw

cwde

;ebx = x\*y

imul ebx

mov ebx,eax

;eax = z+1+x\*y

mov al,z

cbw

cwde

add eax,1

add eax,ebx

jmp print\_mark

case\_Z\_in:

;ebx = 4\*x

mov ax,x

cwde

mov ebx,4

imul ebx

mov ebx,eax

;eax = 2\*z

mov al,z

cbw

cwde

mov ecx,2

imul ecx

;eax = 4\*x-2\*z

sub ebx,eax

mov eax,ebx

jmp print\_mark

case\_Z\_not\_in:

;ebx = z\*z\*z\*z

mov al,z

cbw

cwde

mov ebx,eax

imul ebx

imul ebx

imul ebx

mov ebx,eax

;ecx = y\*y\*y

mov al,y

cbw

cwde

mov ecx,eax

imul ecx

imul ecx

mov ecx,eax

;eax = x\*x

mov ax,x

cwde

imul eax

;eax = x^2\*y^3\*z^4

imul ebx

imul ecx

jmp print\_mark

print\_mark:

push eax

push offset format

call crt\_printf

call crt\_\_getch

push 0

call ExitProcess

END start

.386

.model flat, stdcall

option casemap: none

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.DATA

h db 3h ;знаковая

x db 2h,3h; знаковая

y dw 4h,5h; беззнаковая

m dw 2h; беззнаковая

const\_mul1 dd 1000

format db "sum = %d", 0

.CODE

;eax - число, ecx - степень.

pow proc

push ecx

push edx

push ebx

mov ecx,[esp+16]

mov eax,[esp+20]

mov ebx,[esp+20]

sub ecx,1

pow\_cycle:

mul ebx

loop pow\_cycle

pop ebx

pop edx

pop ecx

ret 8

pow endp

;2 аргумента x,y результат в eax

f proc

push edx

push ebx

;eax - x

mov eax,[esp+12]

cmp eax,10

ja f\_x\_true

;ebx - y

mov ebx,[esp+16]

sub eax,ebx

mov ebx,eax

mov eax,const\_mul1

idiv ebx

jmp f\_return

f\_x\_true:

xor eax,edx

sub eax,edx

mov edx,eax

mov ax,[esp+12]

cwde

sub edx,eax

mov eax,edx

jmp f\_return

f\_return:

pop ebx

pop edx

ret 8

f endp

start:

;esi - i

;ecx - j

;ebp - аккумулятор суммы

;eax,ebx,edx - свободны

xor eax,eax

xor ebx,ebx

xor ecx,ecx

xor edx,edx

xor ebp,ebp

xor esi,esi

start\_external\_loop:

cmp si,m

jb external\_loop\_body

jmp end\_prog

external\_loop\_body:

mov cx,0

internal\_loop\_body:

;ebx = (h+17\*i)

xor eax,eax

add eax,17

imul esi

mov ebx,eax

mov al,h

cbw

cwde

add ebx,eax

;ebx = (h+17\*i)\*f(x[i],y[j])

mov ax,y[ecx\*2]

cwde

push eax

mov al,x[esi]

cbw

cwde

push eax

call f

mul ebx

mov ebx,eax

mov ax,y[esi\*2]

cwde

mul ebx

mov ebx,eax

mov al,x[esi]

cbw

cwde

push eax

push 5

call pow

add eax,ebx

add ebp,eax

mov al,h

cbw

cwde

imul esi

sub ebp,eax

inc cx

cmp cx,m

jb internal\_loop\_body

add esi,1

jmp start\_external\_loop

end\_prog:

mov eax,ebp

push eax

push offset format

call crt\_printf

END start