**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №4**

**Дисциплина: Архитектура вычислительных систем**

**Тема: Команды передачи управления**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: Осипов Олег Васильевич

**Белгород 2019**

**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.

**Вариант 13**

**задание 1**

|  |  |  |
| --- | --- | --- |
| 13 |  | x – беззнаковое 2-байтовое  y – знаковое однобайтовое  z – знаковое однобайтовое |

**задание 2**

|  |  |  |
| --- | --- | --- |
| 13 |  | *x* – массив 1-байтовых беззнаковых чисел  *y* – массив 2-байтовых знаковых чисел  *k* – беззнаковая переменная размером 2 байта  *n* – беззнаковая переменная размером 1 байт |

**Ход выполнения работы**

Задание 1

.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 12 ; 26 ; 20

y db -5 ; 16 ; 12

z db 10 ; -2 ; -8

; p3 ; p2 ; p1

fmt db "%d", 0

.code

START:

MOVSX ECX, y ; записывем y в ECX

CMP ECX, 10 ; если y <= 10, отправляемся к p3

JLE p3

MOVSX ECX, z ; записываем z в ECX

CMP ECX, 0 ; если z < 0, отправляемся в neg

JL negation

abs:

CMP EAX, 5 ; если abs(z) > 5, отправляемся в p1

JG p1

JMP p2 ; если y > 10 и abs(z) <= 5, отправляемся в p2

negation:

MOV EAX, -1 ; записывем -1 в EAX

CBW

CWDE

IMUL ECX ; умножаем EAX на EDX ; abs(z)

JMP abs ; отправляемся в abs

p1:

XOR EAX, EAX ; очищаем EAX

MOV AX, x ; записываем x в EAX

MOVSX EDX, y ; записываем y в EDX

ADD EAX, EDX ; складываем EAX и EDX ; x + y

ADD EAX, 1 ; прибавляем 1 к EAX ; x + y + 1

IMUL EAX ; умножаем EAX на EAX, результат в EAX ; (x+y+1)^2

MOV EBX, EAX ; записываем EAX в EBX

MOVSX EAX, z ; записываем z в EAX

IMUL EAX ; умножаем EAX на EAX, результат в EAX ; z^2

IMUL EAX ; умножаем EAX на EAX, результат в EAX ; z^4

IMUL EBX ; умножаем EAX на EBX, результат в EAX ; (x+y+1)^2 \* z^4

MOV EBX, EAX ; записываем EAX в EBX

MOVSX EAX, y ; записываем y в EAX

IMUL EAX ; умножаем EAX на EAX, результат в EAX ; y^2

MOVSX EDX, y ; записываем y в EDX

IMUL EDX ; умножаем EAX на EDX, результат в EAX ; y^3

SUB EAX, 2 ; вычитаем 2 из EAX ; y^3 + 2

MOV ECX, EAX ; записываем EAX в ECX

MOV EAX, EBX ; записываем EBX в EAX

CDQ

IDIV ECX ; делим EAX на ECX, результат в EDX:EAX ; (x+y+1)^2 \* z^4 / (y^3 + 2)

PUSH EAX ; загрузка в стек первого аргумента

PUSH offset fmt ; загрузка в стек второго аргумента

CALL crt\_printf ; вызов подпрограммы вывода

ADD ESP, 8 ; освобождение памяти

JMP exit

p2:

MOV EAX, 1 ; записываем 1 в EAX

XOR EBX, EBX

MOV BX, x ; записываем x в EBX

IDIV EBX ; делим EAX на EBX, результат в EAX ; 1/x

MOV EBX, EAX ; записываем EAX в EBX

MOV EAX, ECX ; запимываем ECX в EAX ; z

IMUL EAX ; умножаем EAX на EAX, результат в EAX ; z^2

MOV ECX, EAX ; записываем EAX в ECX

MOV EAX, -1 ; записываем -1 в EAX

IMUL ECX ; умножаем EAX на ECX, результат в EAX ; -z^2

ADD EBX, EAX ; складываем EBX и EAX, результат в EBX ; -z^2 + 1/x

XOR EAX, EAX

MOV AX, x ; записываем x в EAX

MOV ECX, EBX ; записываем EBX в ECX

MOVSX EBX, y ; записываем y в EBX

ADD EBX, 10 ; прибавляем 10 к EBX ; y + 10

XOR EDX, EDX

IDIV EBX ; делим EAX на EBX, результат в EAX ; x/(y+10)

ADD EAX, ECX ; складываем EAX и ECX, результат в EAX ; -z^2 + 1/x + x/(y+10)

PUSH EAX ; загрузка в стек первого аргумента

PUSH offset fmt ; загрузка в стек второго аргумента

CALL crt\_printf ; вызов подпрограммы вывода

ADD ESP, 8 ; освобождение памяти

JMP exit

p3:

MOVSX EAX, z ; записываем z в EAX

ADD EAX, 1 ; прибавляем 1 к EAX ; z + 1

MOV EDX, 2 ; записываем 2 в EDX

IMUL EDX ; умножаем EAX на EDX, результат в EAX ; 2\*(z+1)

IMUL ECX ; умножаем EAX на ECX, результат в EAX ; 2\*(z+1)\*y

MOV EBX, EAX ; записываем EAX в EBX

XOR EAX, EAX

MOV AX, x ; записываем x в AX

IMUL EAX ; умножаем EAX на EAX, результат в EAX ; x^2

MOV ECX, -5 ; записываем -5 в ECX

IMUL ECX ; умножаем EAX на ECX, результат в EAX ; -5\*(x^2)

SUB EAX, EBX ; вычитаем EBX из EAX, результат в EAX ; -5\*(x^2) - 2\*(z+1)\*y

exit:

PUSH EAX ; загрузка в стек первого аргумента

PUSH offset fmt ; загрузка в стек второго аргумента

CALL crt\_printf ; вызов подпрограммы вывода

ADD ESP, 8 ; освобождение стека

CALL crt\_\_getch

PUSH NULL

CALL ExitProcess

END START

Набор тестовых данных

|  |  |  |  |
| --- | --- | --- | --- |
| x | y | z | a |
| 20 | 12 | -8 | 2584 |
| 26 | 16 | -2 | -3 |
| 12 | -5 | 10 | -610 |

.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 db 1, 2, 3, 2, 1

y dw 6, -6, -3, 3, -1

k dw 4

n db 5

i db 0

j db 0

f dd 0

r dd 0

save dd 0

fmt db "%d", 0

.code

func proc

PUSH EBX

PUSH ECX ; сохраняем EBX и ECX

XOR ECX, ECX

MOV CL, [ESP+8+8] ; записываем xi в ECX

CMP ECX, 0

JLE endFunc ; переходим к endFunc, если xi < 0

subLoop:

XOR EDX, EDX

MOV DL, j

MOV AX, y[EDX\*2] ; записываем yj в EAX

CWDE

MOV EBX, EAX ; записываем EAX в EBX

SUB EBX, 1 ; вычитаем 1 из EBX ; yi-1

CMP EBX, 0

JGE cont ; переходим к con, если yi-1 > 0

MOV EAX, -1 ; записываем -1 в EAX

IMUL EBX ; умножаем EAX на EBX, результат в EAX ; abs(yi-1)

MOV EBX, EAX ; запимываем EAX в EBX

cont:

MOV EAX, dword ptr f[0]

ADD EAX, EBX ; сохраняем результат работы subLoop

MOV dword ptr f[0], EAX

XOR EAX, EAX

MOV AL, j

INC AL

MOV j, AL ; сохраняем новое значение j ; j+1

DEC ECX ; уменьшаем ECX на 1

CMP ECX, 0

JG subLoop ; переходим в subLoop, если ECX >= 0

endFunc:

MOV j, 0

POP ECX

POP EBX ; возвращаем EBX и ECX

RET 8 ; выходим из функции

func endp

START:

XOR ECX, ECX

MOV CL, n ; записываем n в ECX

CMP ECX, 0

JLE exit ; переходим к mainLoop, если n < 0

mainLoop:

XOR EDX, EDX

MOV Dl, i

XOR EBX, EBX

MOV BL, x[EDX] ; записываем xi в EBX

MOV AX, y[EDX\*2] ; записываем yi в EAX

CWDE

ADD EBX, EAX ; складываем EBX и EAX, результат в EBX ; x1 + yi

XOR EAX, EAX

MOV AX, k ; записываем k в EAX

IMUL EBX ; умножаем EAX на EDX, результат в EAX ; k\*(xi+yi)

MOV EBX, EAX ; записываем EAX в EBX

XOR EAX, EAX

MOV AX, k ; записываем k в EAX

IMUL EAX ; умножаем EAX на EAX, результат в EAX ; k^2

XOR EDX, EDX

MOV DX, k ; записываем k в EDX

IMUL EDX ; умножаем EAX на EDX, результат в EAX ; k^3

ADD EBX, EAX ; складываем EBX и EAX, результат в EBX ; k\*(xi+yi) + k^3

XOR EDX, EDX

MOV DL, i

MOV AX, y[EDX\*2] ; записываем yi в EAX

CWDE

IMUL EAX ; умножаем EAX на EAX, результат в EAX ; yi^2

SUB EBX, EAX ; вычитаем EAX и EBX, результат в EBX ; k\*(xi+yi) + k^3 - yi^2

XOR EDX, EDX

MOV DL, i

XOR EAX, EAX

MOV AL, x[EDX]

PUSH EAX

MOV AX, y[EDX\*2]

CWDE

PUSH EAX

CALL func ; вызываем подпрограмму func с аргументами xi и yi

MOV EAX, dword ptr f ; f(xi,yi)

MOV f, 0

MOV dword ptr save, ECX

XOR ECX, ECX

MOV CX, k ; записываем k в EDX

XOR EDX, EDX

IDIV ECX ; делим EAX на EDX, результат в EAX ; f(xi,yi)/k

MOV ECX, dword ptr save

ADD EBX, EAX ; складываем EAX и EBX, результат в EBX ; k\*(xi+yi) + k^3 - yi^2 + f(xi,yi)/k

MOV EAX, dword ptr r

ADD EAX, EBX ; сохраняем результат работы mainLoop

MOV dword ptr r, EAX

XOR EAX, EAX

MOV AL, i

INC AL

MOV i, AL ; сохраняем новое значение i ; i+1

DEC ECX ; уменьшаем ECX на 1

CMP ECX, 0

JG mainLoop ; переходим к mainLoop, если ECX >= 0

exit:

MOV EAX, dword ptr r

PUSH EAX ; загрузка в стек первого аргумента

PUSH offset fmt ; загрузка в стек второго аргумента

CALL crt\_printf ; вызов подпрограммы вывода

ADD ESP, 8 ; освобождение стека

CALL crt\_\_getch

PUSH NULL

CALL ExitProcess

END START

Набор тестовых данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x = {1,2,3,2,1} | y = {6,-6,-3,3,-1} | k = 4 | n = 5 | r = 273 |
| x = {1,2,3} | y = {6,-6,-3} | k = 5 | n = 3 | r = 315 |
| x = {1} | y = {6} | k = 10 | n = 1 | r = 1034 |

.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

arr dw 1, 2, 3, 4, 5

n dd 5

i dd 0

fmt db "(arr[%d])^2 = %d", 10, 0

.code

output\_square proc

MOV ECX, dword ptr n

CMP ECX, 0

JLE exit

point:

MOV EDX, dword ptr i

XOR EBX, EBX

MOV BX, arr[EDX\*2]

INC EDX

MOV dword ptr i, EDX

XOR EDX, EDX

MOV EAX, EBX

MUL EAX

PUSH EAX

PUSH EBX

PUSH offset fmt

CALL crt\_printf

ADD ESP, 12

MOV ECX, dword ptr n

MOV EBX, dword ptr i

SUB ECX, EBX

CMP ECX, 0

JG point

RET 12

output\_square endp

START:

CALL output\_square

INC EAX

exit:

CALL crt\_\_getch

PUSH NULL

CALL ExitProcess

END START