**Звіт**

Автор: Салтиков Іван;

Група: КН-921д;

# Лабораторна робота №6. SSE- і AVX-команди. Графічні фігури

***Мета.*** Набути практичних навичок складання, налагодження та виконання програм з використанням команд SSE і AVX для платформи х64 в середовищі masm64.

Набути практичних навичок складання, налагодження та виконання програм, написаних мовою ассемблера в середовищі masm64 з віконним інтерфейсом для програмування арифметичних операцій за допомогою SSE (AVX) команд під МП.

Хід роботи

include \masm64\include64\masm64rt.inc

.data?

hInstance dq ?

hIcon dq ?

hBmp dq ?

hStatic dq ?

.data

arr1 dq -1.3, 2.1, 3.8, 1.0, 5.4, 6.12, 7.54 ; массив 1

arr2 dq -1., -5.0, -3.54, 1.5, -5.8, -6.53, 7.5 ; массив 2

len1 dq 7 ; количество чисел в массивах

arr3 dq 7 dup(0) ; результирующий массив

countFirst dq 0

countSecond dq 0

fmt db "Масив 1:",10,"-1.3 2.1 3.8 1.0 5.4 6.12 7.54",10,10,

"Массив 2:",10,"-1 -5.0 -3.54 1.5 -5.8 -6.53 7.5",10,10,

"Результат:",10,"%d %d %d %d %d %d %d",10,0

buf dq 12 dup(0),0

.code

entry\_point proc

GdiPlusBegin ; инициализация GDIPlus

mov hInstance, rv(GetModuleHandle,0)

mov hIcon,rv(LoadImage,hInstance,10,IMAGE\_ICON,32,32,LR\_DEFAULTCOLOR)

mov hBmp,rv(ResImageLoad,20)

invoke DialogBoxParam,hInstance,100,0,ADDR main,hIcon

GdiPlusEnd ; GdiPlus очистка

invoke ExitProcess,0

ret

entry\_point endp

main proc hWin:QWORD,uMsg:QWORD,wParam:QWORD,lParam:QWORD

.switch uMsg

.case WM\_INITDIALOG ; сообщение о создании диал. окна

invoke SendMessage,hWin,WM\_SETICON,1,lParam

mov hStatic, rv(GetDlgItem,hWin,102)

invoke SendMessage,hStatic,STM\_SETIMAGE,IMAGE\_BITMAP,hBmp

.return TRUE

.case WM\_COMMAND ; сообщение от меню или кнопки

.switch wParam

.case 101 ; если выбран вывод информации о задании

.data

txt2 db "Выполнить параллельное сравнение массивов по 7-мь 64-разрядных вещественных числа. Если первый массив меньше второго, то выполнить операцию деления над массивами чисел, иначе – умножения.",0

titl2 db "Информация о задании",0

.code

invoke MsgboxI,hWin,ADDR txt2,ADDR titl2,MB\_OK,10

.case 102 ; если выбран вывод информации об авторе

.data

txt1 db "Автор: Saltikov Ivan",10,"Группа: KN-921d",0

.code

invoke MsgboxI,hWin,ADDR txt1,"Информация об авторе",MB\_OK,10

.case 103

.data

msg db "Выход из программы.",0 ; вывод уведомления о выходе

.code

invoke MsgboxI,hWin,ptr$(msg),"Выход",MB\_OK,10

rcall SendMessage,hWin,WM\_SYSCOMMAND,SC\_CLOSE,0 ; уничтожение окна

.case 1001 ; выполнение ЛР с SSE

.code

mov countFirst,0 ; обнуление переменной перед вычислениями

mov countSecond,0 ; обнуление переменной перед вычислениями

mov rcx,len1 ; кол-во циклов

lea rsi,arr1 ; установка указателя на массив 1

lea rdi,arr2 ; установка указателя на массив 2

lea rbx,arr3 ; установка указателя на массив 3

jmp cycleSSE ; перейти в цикл проверки массива

FirstOrEqualSSE: ; если элемент первого массива >=

add countFirst,1 ; увеличение переменной на 1

jmp cycleSSE ; переход в цикл

SecondSSE: ; если элемент первого массива <

add countSecond,1 ; увеличение переменной на 1

cycleSSE: ; цикл попарного сравнения элементов

cmp rcx,0 ; проверка на завершение цикла

je nextSSE ; переход на проверку какой массив больше

movsd XMM0,qword ptr[rsi] ; получение элемента из массива 1

movsd XMM1,qword ptr[rdi] ; получение элемента из массива 2

add rsi,8 ; переход на следующий элемент массива 1

add rdi,8 ; переход на следующий элемент массива 1

dec rcx ; уменьшение счетчика циклов

comisd XMM0,XMM1 ; сравнение элементов

jnb FirstOrEqualSSE ; если элемент первого массива больше

jb SecondSSE ;

loop cycleSSE

nextSSE:

lea rsi,arr1 ; установка указателя на массив 1

lea rdi,arr2 ; установка указателя на массив 2

lea rbx,arr3 ; установка указателя на массив 3

mov rax,countFirst

mov rbx,countSecond

cmp rax,rbx ; проерка какой массив больше

ja MultiplicationSSE

DivisionSSE: ; деление массивов

movups XMM0,[rsi] ; получение элемента из массива 1

movups XMM1,[rdi] ; получение элемента из массива 2

divpd xmm0,xmm1 ; деление 2-х пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; пересылка в регистр EAX из MM0

mov arr3,rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 2

mov arr3[8],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

movups XMM0,[rsi] ; получение элемента из массива 1

movups XMM1,[rdi] ; получение элемента из массива 2

divpd xmm0,xmm1 ; деление 2-х пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 3

mov arr3[16],rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 4

mov arr3[24],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

movups XMM0,[rsi] ; получение элемента из массива 1

movups XMM1,[rdi] ; получение элемента из массива 2

divpd xmm0,xmm1 ; деление 2-х пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 5

mov arr3[32],rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 6

mov arr3[40],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

movups XMM0,[rsi] ; получение элемента из массива 1

movups XMM1,[rdi] ; получение элемента из массива 2

divpd xmm0,xmm1 ; деление 2-х пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 7

mov arr3[48],rax ; запись в результирующий массив

jmp \_endSSE ; переход в конец программы

MultiplicationSSE: ; умножение массивов

movups XMM0,[rsi] ; получение элемента из массива 1

movups XMM1,[rdi] ; получение элемента из массива 2

mulpd xmm0,xmm1 ; умножение двух пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 1

mov arr3,rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 2

mov arr3[8],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

movups XMM0,[rsi] ; получение элемента из массива 1

movups XMM1,[rdi] ; получение элемента из массива 2

mulpd xmm0,xmm1 ; умножение двух пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 3

mov arr3[16],rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 4

mov arr3[24],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

movups XMM0,[rsi] ; получение элемента из массива 1

movups XMM1,[rdi] ; получение элемента из массива 2

mulpd xmm0,xmm1 ; умножение двух пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 5

mov arr3[32],rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 6

mov arr3[40],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

movups XMM0,[rsi] ; получение элемента из массива 1

movups XMM1,[rdi] ; получение элемента из массива 2

mulpd xmm0,xmm1 ; умножение двух пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 7

mov arr3[48],rax ; запись в результирующий массив

\_endSSE:

mov r8,arr3 ; запись в регистр r8

mov r9,arr3[8] ; запись в регистр r9

mov r10,arr3[16] ; запись в регистр r10

mov r11,arr3[24] ; запись в регистр r11

mov r12,arr3[32] ; запись в регистр r12

mov r13,arr3[40] ; запись в регистр r13

mov r14,arr3[48] ; запись в регистр r14

invoke wsprintf, ADDR buf, ADDR fmt, r8,r9,r10,r11,r12,r13,r14

invoke MsgboxI,hWin,ADDR buf,"Результат вычислений SSE",MB\_OK,10

.case 1003 ; если выбрано выполнение ЛР с AVX

mov countFirst,0 ; обнуление переменной перед вычислениями

mov countSecond,0 ; обнуление переменной перед вычислениями

mov rcx,len1 ; кол-во циклов

lea rsi,arr1 ; установка указателя на массив 1

lea rdi,arr2 ; установка указателя на массив 2

lea rbx,arr3 ; установка указателя на массив 3

jmp cycleAVX ; перейти в цикл проверки массива

FirstOrEqualAVX: ; если элемент первого массива >=

add countFirst,1 ; увеличение переменной на 1

jmp cycleAVX ; переход в цикл

SecondAVX: ; если элемент первого массива <

add countSecond,1 ; увеличение переменной на 1

cycleAVX: ; цикл попарного сравнения элементов

cmp rcx,0 ; проверка на завершение цикла

je nextAVX ; переход на проверку какой массив больше

vmovsd XMM0,qword ptr[rsi] ; получение элемента из массива 1

vmovsd XMM1,qword ptr[rdi] ; получение элемента из массива 2

add rsi,8 ; переход на следующий элемент массива 1

add rdi,8 ; переход на следующий элемент массива 1

dec rcx ; уменьшение счетчика циклов

vcomisd XMM0,XMM1 ; сравнение элементов

jnb FirstOrEqualAVX ; если элемент первого массива больше

jb SecondAVX ;

loop cycleAVX

nextAVX:

lea rsi,arr1 ; установка указателя на массив 1

lea rdi,arr2 ; установка указателя на массив 2

lea rbx,arr3 ; установка указателя на массив 3

mov rax,countFirst

mov rbx,countSecond

cmp rax,rbx ; проерка какой массив больше

ja MultiplicationAVX

DivisionAVX: ; деление массивов

vmovups XMM0,[rsi] ; получение элемента из массива 1

vmovups XMM1,[rdi] ; получение элемента из массива 2

divpd xmm0,xmm1 ; деление 2-х пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; пересылка в регистр EAX из MM0

mov arr3,rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 2

mov arr3[8],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

vmovups XMM0,[rsi] ; получение элемента из массива 1

vmovups XMM1,[rdi] ; получение элемента из массива 2

divpd xmm0,xmm1 ; деление 2-х пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 3

mov arr3[16],rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 4

mov arr3[24],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

vmovups XMM0,[rsi] ; получение элемента из массива 1

vmovups XMM1,[rdi] ; получение элемента из массива 2

divpd xmm0,xmm1 ; деление 2-х пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 5

mov arr3[32],rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 6

mov arr3[40],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

vmovups XMM0,[rsi] ; получение элемента из массива 1

vmovups XMM1,[rdi] ; получение элемента из массива 2

divpd xmm0,xmm1 ; деление 2-х пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 7

mov arr3[48],rax ; запись в результирующий массив

jmp \_endAVX ; переход в конец программы

MultiplicationAVX: ; умножение массивов

vmovups XMM0,[rsi] ; получение элемента из массива 1

vmovups XMM1,[rdi] ; получение элемента из массива 2

mulpd xmm0,xmm1 ; умножение двух пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 1

mov arr3,rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 2

mov arr3[8],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

vmovups XMM0,[rsi] ; получение элемента из массива 1

vmovups XMM1,[rdi] ; получение элемента из массива 2

mulpd xmm0,xmm1 ; умножение двух пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 3

mov arr3[16],rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 4

mov arr3[24],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

vmovups XMM0,[rsi] ; получение элемента из массива 1

vmovups XMM1,[rdi] ; получение элемента из массива 2

mulpd xmm0,xmm1 ; умножение двух пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 5

mov arr3[32],rax ; запись в результирующий массив

unpckhpd xmm0,xmm2 ; передача старшей половины числа в младшую

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 6

mov arr3[40],rax ; запись в результирующий массив

add rsi,16 ; перемещение указателя на следующий элемент

add rdi,16 ; перемещение указателя на следующий элемент

vmovups XMM0,[rsi] ; получение элемента из массива 1

vmovups XMM1,[rdi] ; получение элемента из массива 2

mulpd xmm0,xmm1 ; умножение двух пар элементов

cvtpd2pi mm0,xmm0 ; преобразование в 32-х разрядное число

movd dword ptr eax,mm0 ; число 7

mov arr3[48],rax ; запись в результирующий массив

\_endAVX:

mov r8,arr3 ; запись в регистр r8

mov r9,arr3[8] ; запись в регистр r9

mov r10,arr3[16] ; запись в регистр r10

mov r11,arr3[24] ; запись в регистр r11

mov r12,arr3[32] ; запись в регистр r12

mov r13,arr3[40] ; запись в регистр r13

mov r14,arr3[48] ; запись в регистр r14

invoke wsprintf, ADDR buf, ADDR fmt, r8,r9,r10,r11,r12,r13,r14

invoke MsgboxI,hWin,ADDR buf,"Результат вычислений AVX",MB\_OK,10

.endsw

.case WM\_CLOSE ; если есть сообщение о закрытии окна

invoke EndDialog,hWin,0 ;

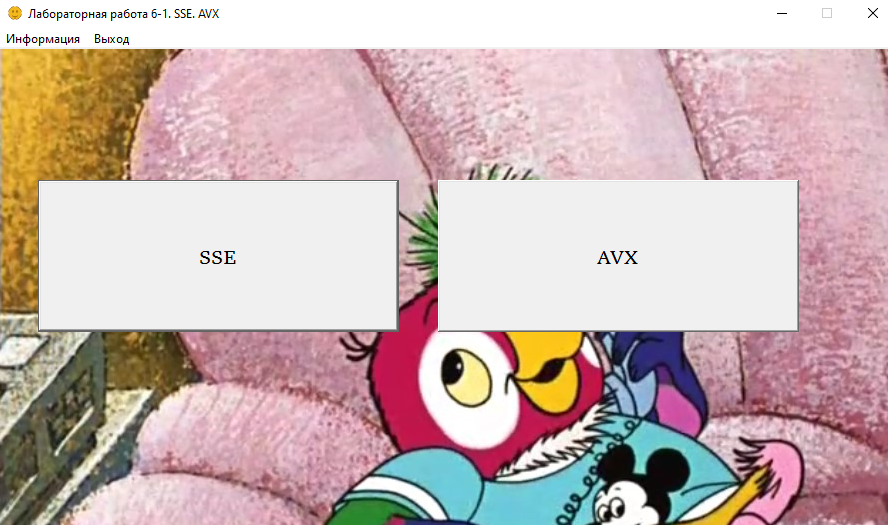
.endsw

xor rax, rax

ret

main endp

end

 #define IDD\_DLG1 100

#define IDC\_IMG1 102

#define IDC\_BTN1 1001

#define IDC\_BTN2 1003

#define IDR\_MENU1 10000

#define IDM\_Task 101

#define IDM\_Author 102

#define IDM\_Exit 103

#include "/masm64/include64/resource.h"

10 ICON DISCARDABLE "1.ico"

20 RCDATA DISCARDABLE "12.jpg"

IDD\_DLG1 DIALOGEX 10,10,276,132

CAPTION "Лабораторная работа 6-1. SSE. AVX"

FONT 15,"Sitka Small",700,0,204

MENU IDR\_MENU1

STYLE WS\_VISIBLE|WS\_CAPTION|WS\_SYSMENU|WS\_MINIMIZEBOX|DS\_CENTER|DS\_MODALFRAME

EXSTYLE WS\_EX\_DLGMODALFRAME

BEGIN

CONTROL "",IDC\_IMG1,"Static",WS\_CHILDWINDOW|WS\_VISIBLE|SS\_BITMAP,0,0,189,99

CONTROL "SSE",IDC\_BTN1,"Button",WS\_CHILDWINDOW|WS\_VISIBLE|WS\_TABSTOP,12,36,111,42

CONTROL "AVX",IDC\_BTN2,"Button",WS\_CHILDWINDOW|WS\_VISIBLE|WS\_TABSTOP,135,36,111,42

END

IDR\_MENU1 MENU

BEGIN

POPUP "Информация"

BEGIN

MENUITEM "Задание",IDM\_Task

MENUITEM "Автор",IDM\_Author

END

MENUITEM "Выход",IDM\_Exit

END

include \masm64\include64\masm64rt.inc

IDI\_ICON EQU 1001

.data?

hInstance dq ? ; дескриптор програми

hWnd dq ? ; дескриптор окна

hIcon dq ? ; дескриптор иконки

hCursor dq ? ; дескриптор курсора

sWid dq ? ; ширина монитора (колич. пикселей по x)

sHgt dq ? ; высота монитора (колич. пикселей по y)

.data

color1 dd 97123

mas dd 720

alpha dd 0.0 ; угловая координата

delta dd 0.0175 ; один градус

centerX dq ? ; середина по X

centerY dq ? ; середина по Y

tmp dd 0 ; временная переменная

divK dd 80.0 ; масштабный коэффициент

xR dd 0. ; координаты функции

yR dd 0.0

temp1 dd 0

x1 dd 0.0

x2 dq 2.0

classname db "template\_class",0

caption db "Лабораторна робота 6-2. Графічні фігури",0

PaintCa macro

; invoke DestroyWindow,hWnd ; уничтожение окна

invoke BeginPaint, hWnd, addr ps ; вызов функции начала рисования

mov hdc, rax ; сохроанение контекста

invoke GetClientRect, hWnd, addr rect ; запись в структуру rect

invoke GetSystemMetrics,SM\_CXSCREEN ; получение ширины экрана в пикселях

shr rax,1 ; определение середины экрана по X

mov centerX,320

invoke GetSystemMetrics,SM\_CYSCREEN ; получение высоты экрана в пикселях

shr rax,1 ; определение середины экрана по Y

mov centerY,260

mov r10d,mas ; сохранение количества циклов

mov temp1,r10d

finit ; инициализация сопроцессора

cycle1:

fld alpha ; загрузка a

fmul x2 ; 2a

fcos ; cos(2a)

fld divK ; загрузка коэффициента масштаба

fmul ; cos(2a)\*divK

fld alpha ; загрузка a

fcos ; cos(a)

fmul x2 ; 2cos(a)

fmul divK ; divK\*2cos(a)

fsub

fild centerY ; загрузка центра экрана по Y

fadd ; centerY + a\*divK\*cos(a)

fistp dword ptr yR ; сохранение Y для выведения на экран

fld alpha ; загрузка a

fmul x2 ; 2a

fsin ; cos(2a)

fld divK ; загрузка коэффициента масштаба

fmul ; cos(2a)\*divK

fld alpha ; загрузка a

fsin ; sin(a)

fmul x2 ; 2sin(a)

fmul divK ; divK\*2cos(a)

fsub ; a\*divK\*cos(a)

fild centerX ; загрузка центра экрана по X

fadd ; centerX + a\*divK\*sin(a)

fistp dword ptr xR ; сохранение X

invoke Sleep,1 ; задержка при рисовании

invoke SetPixel, hdc, xR, yR, color1 ; изменение цвета пикселя

movss XMM3,delta

addss XMM3,alpha

movss alpha,XMM3

dec temp1 ; уменьшение счетчика

jz \_end ; продолжение рисование

jmp cycle1 ; возвращение в начало цикла

\_end:

endm

CursorCa macro

invoke GetSystemMetrics,SM\_CXSCREEN ; получение ширины экрана в пикселях

shr rax,1 ; определение середины экрана по X

mov centerX,rax

invoke GetSystemMetrics,SM\_CYSCREEN ; получение высоты экрана в пикселях

shr rax,1 ; определение середины экрана по Y

mov centerY,rax

mov r10d,mas ; сохранение количества циклов

mov temp1,r10d

finit ; инициализация сопроцессора

cycle2:

fld alpha ; загрузка альфы

fmul x2 ; 2а

fcos ; cos(2a)

fld divK ; загрузка коэффициента масштаба

fmul ; divK\*cos(2a)

fld alpha ; загрузка а

fcos ; cos(a)

fmul x2 ; 2cos(a)

fmul divK ; divK\*2cos(a)

fsub

fild centerY ; загрузка центра экрана по Y

fadd ; centerY + a\*divK\*cos(a)

fistp dword ptr yR ; сохранение Y для выведения на экран

fld alpha ; загрузка а

fmul x2 ; 2a

fsin ; sin(2a)

fld divK ; загрузка коэффициента масштаба

fmul

fld alpha ; загрузка а

fsin ; sin(a)

fmul x2 ; 2sin(a)

fmul divK ; divK\*2sin(a) st1 = divk\*cos(2a)

fsub ; a\*divK\*cos(a)

fild centerX

fadd ; xR := centerX + ...

fistp dword ptr xR ; сохранение X

invoke Sleep,1 ; задержка при рисовании

invoke SetCursorPos,xR,yR ; установление курсора по xR, yR

movss XMM3,delta

addss XMM3,alpha

movss alpha,XMM3

dec temp1 ; уменьшение счетчика циклов

dec temp1 ; уменьшение счетчика циклов

jz \_end2 ; завершить рисование

jmp cycle2 ; переход в начало цикла

\_end2:

endm

.code

entry\_point proc

mov hInstance,rv(GetModuleHandle,0) ; получение и сохранение дескрипторa програми

mov hIcon, rv(LoadIcon,hInstance,10) ; загрузка и сохранение дескрипторa иконки

mov hCursor,rv(LoadCursor,0,IDC\_ARROW) ; загрузка курсора и сохранение

mov sWid,rv(GetSystemMetrics,SM\_CXSCREEN) ; получение кол. пикселей по х монитора

mov sHgt,rv(GetSystemMetrics,SM\_CYSCREEN) ; получение кол. пикселей по y монитора

; mov hBrush,rvcall(CreateSolidBrush,00C4C4C4h)

call main ; вызов процедуры main

rcall ExitProcess,0

ret

entry\_point endp

msgloop proc

LOCAL msg :MSG

LOCAL pmsg :QWORD

mov pmsg, ptr$(msg) ; получение адреса структуры сообщения

jmp gmsg ; jump directly to GetMessage()

mloop:

rcall TranslateMessage,pmsg

rcall DispatchMessage,pmsg

gmsg:

test rax, rvcall(GetMessage,pmsg,0,0,0) ; пока GetMessage не вернет ноль

jnz mloop

ret

msgloop endp

main proc

LOCAL wc :WNDCLASSEX ; объявление локальных переменных

LOCAL lft :QWORD ; лок. переменные содержатся в стеке

LOCAL top :QWORD ; и существуют только во время вып. проц.

LOCAL wid :QWORD

LOCAL hgt :QWORD

LOCAL rect:RECT ; резервирование стека под структуру

LOCAL ps:PAINTSTRUCT ; резервирование стека под структуру

LOCAL hdc:HDC ; резервирование стека под хендл окна

mov wc.cbSize,SIZEOF WNDCLASSEX ; количество байтов структуры

mov wc.style,CS\_BYTEALIGNCLIENT or CS\_BYTEALIGNWINDOW ; стиль окна

mov wc.lpfnWndProc,ptr$(WndProc) ; адрес процедуры WndProc

mov wc.cbClsExtra,0 ; количество байтов для структуры класса

mov wc.cbWndExtra,0 ; количество байтов для структуры окна

mrm wc.hInstance,hInstance ; заполнение пол¤ дескриптора в структуре

mrm wc.hIcon, hIcon ; хэндл иконки

mrm wc.hCursor,hCursor ; хэндл курсора

mrm wc.hbrBackground,0 ; hBrush цвет окна

mov wc.lpszMenuName,0 ; заполнение пол¤ в структуре с именем ресурса меню

mov wc.lpszClassName,ptr$(classname); имя класса

mrm wc.hIconSm,hIcon

invoke RegisterClassEx,ADDR wc ; регистраци¤ класса окна

mov wid, 640 ; ширина пользовательского окна в пикселях

mov hgt, 640 ; высота пользовательского окна в пикселях

mov rax,sWid ; количество пикселей монитора по x

sub rax,wid ; дельта Х = Х(монитора) - х(окна пользователя)

shr rax,1 ; получение середины Х

mov lft,rax

mov rax, sHgt ; количество пикселей монитора по y

sub rax, hgt ;

shr rax, 1 ;

mov top, rax ;

invoke CreateWindowEx,WS\_EX\_LEFT or WS\_EX\_ACCEPTFILES, \

ADDR classname,ADDR caption, \

WS\_OVERLAPPED or WS\_VISIBLE or WS\_SYSMENU,\

lft,top,wid,hgt,0,0,hInstance,0

mov hWnd,rax ; сохранение дескриптора окна

PaintCa

call msgloop

ret

main endp

WndProc proc hWin:QWORD,uMsg:QWORD,wParam:QWORD,lParam:QWORD

LOCAL dfbuff[260]:BYTE

LOCAL pbuff :QWORD

LOCAL rect:RECT ; резервування стека під структуру RECT

LOCAL ps:PAINTSTRUCT ; резервування стека під структуру

LOCAL hdc:HDC ; резервування стека під хендл вікна

.switch uMsg

.case WM\_COMMAND

.switch wParam

.case 10003

rcall SendMessage,hWin,WM\_SYSCOMMAND,SC\_CLOSE,NULL

.case 10002

invoke MsgboxI,hWin,"Малюємо кардіоіду","Малювання кардіоіди",MB\_OK,10

PaintCa

.case 10004

invoke MsgboxI,hWin,"Переміщення курсору","У просессі...",MB\_OK,10

CursorCa

.case 10005

.data

Task db "Намалювати кардіоіду відповідно до формул. Координати вершин в параметричних координатах задаются за формулам:",10,"X = 2Rcost(1+cost);",10,"Y = 2Rsint(1+cost);",10,"де R – рідиус окружности.",0

.code

invoke MsgboxI,hWin,ptr$(Task),"Завдання",MB\_OK,10

.case 10001

.data

msgtxt db "Автор: Іван Салтиков",10,"Группа: КН-921д",0

.code

invoke MsgboxI,hWin,ptr$(msgtxt),"Автор",MB\_OK,10

.endsw

.case WM\_CREATE

rcall LoadMenu,hInstance,10000

rcall SetMenu,hWin,rax

.return 0

.case WM\_CLOSE

rcall SendMessage,hWin,WM\_DESTROY,0,0

.case WM\_DESTROY

rcall PostQuitMessage,NULL

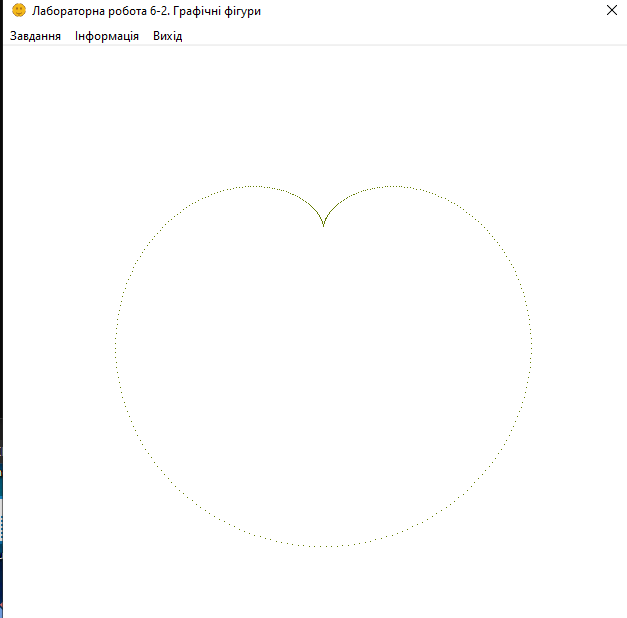
.endsw

rcall DefWindowProc,hWin,uMsg,wParam,lParam

ret

WndProc endp

end



#define IDM\_Prog 10001

#define IDI\_ICON 1001

#define IDR\_MENU1 10000

#define IDM\_Cursor 10004

#define IDM\_Paint 10002

#define IDM\_About 10005

#define IDM\_Author 10001

#define IDM\_Exit 10003

#include "C:/masm64/include64/Resource.h"

IDI\_ICON ICON DISCARDABLE "1.ico"

10 ICON DISCARDABLE "1.ico"

IDR\_MENU1 MENU

BEGIN

POPUP "Завдання"

BEGIN

MENUITEM "Змістити курсор",IDM\_Cursor

MENUITEM "Малювати кардиоду",IDM\_Paint

END

POPUP "Інформація"

BEGIN

MENUITEM "Завдання",IDM\_About

MENUITEM "Автор",IDM\_Author

END

MENUITEM "Вихід",IDM\_Exit

END

Висновки

При виконанні даної лабораторної роботи було набуто навичок при роботі з SSE- і AVX-командами та графічними фігурами.

Програми протестовані та виконуються без помилок.