Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Новосибирский государственный технический университет»

Кафедра теоретической и прикладной информатики

Расчётно-графическая работа
по дисциплине «Управление ресурсами в вычислительных системах»

Факультет:	ПМИ
Группа:	ПМ-92
Вариант:	25
Студент:	Иванов В. В.
Преподаватель:	Стасышин В. М.

Новосибирск

2022

1. Цель работы

Изучение основных особенностей операционной системы Windows и базовых инструментов для управления ресурсами с использованием набора системных вызовов WinAPI.

2. Задание

Реализовать графическое приложение, в котором при создании окна, но до момента его отображения на экране, создается дочерний поток, который определяет **количество функциональных клавиш клавиатуры**, после чего полученный результат отображается в графическом окне.
Функция, выполняющая задание, должна быть реализована в виде динамической библиотеки, в которой используется ассемблерная вставка, определяющая **длину строки КЭШа данных третьего уровня**. Длина строки выводится вместе с основным результатом.
Для реализации основной задачи используется функция **GetKeyboardType** с параметром **2** из набора WinAPI.

3. Используемые средства

Работа была выполнена на терминальном сервере “Dev” облачной платформы НГТУ:

Характеристики устройства

Имя устройства	cloud-rdhost1
Процессор	AMD EPYC 7402 24-Core Processor 2.79 GHz (процессоров: 2)
Оперативная память	128 ГБ (доступно: 128 ГБ)
Код устройства	734BAD81-17B0-4729-88B1-C06D3578D1C7
Код продукта	00429-00002-28353-AA660
Тип системы	64-разрядная операционная система, процессор x64

Программа написана на языке C++. Ассемблерная вставка, определяющая длину строки КЭШа третьего уровня, реализована и протестирована только для процессоров модели AMD.
Среда разработки: Microsoft Visual Studio 2019.

Для запуска программы достаточно либо открыть решение **WinResMan.sln** и выполнить сборку в режиме **Release**, либо запустить исполняемый файл **Release/program.exe**.

4. Описание программы

В оконной процедуре (**WndProc**), обрабатывающей сообщения для окна приложения, внутри контекста **dc**, из динамической библиотеки принимается два значения: **funcKeysCount** (количество функциональных клавиш) и **cpuidInfo** (длина строки КЭШа третьего уровня). Данные переменные являются частью структуры **libInfo**, определенной в **library.h**. Их получает функция **GetInfo()** в рамках создаваемого потока.

GetInfo() является частью динамической библиотеки, в которой реализован алгоритм получения информации о КЭШе L3 для процессоров AMD с предварительной проверкой поддержки CPUID. Для получения информации о КЭШе вызывается CPUID в **0x80000006**, где содержится размер КЭШа, число путей и строк на тег, а также длина строки КЭШа. Необходимое значение содержится в регистре EDX.

5. Код программы

main.cpp

```
#include <shlwapi.h>
#include <windows.h>
#include "../library/library.h"
#pragma comment(lib, "library.lib")
#pragma comment(lib, "shlwapi.lib")
#define BUFSIZE 128
char BUF[BUFSIZE];
volatile library_info libInfo;

// оконная процедура, обрабатывающая сообщения для окна программы
LRESULT CALLBACK WndProc(HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam) {
    PAINTSTRUCT ps;
    HDC dc;

    switch (uMsg) {
    case WM_PAINT: // перерисовка окна
        dc = BeginPaint(hWnd, &ps); // инициализация контекста
        wnsprintf(BUF, BUFSIZE - 1, // вывод информации в буфер
            "Number of functional keys: %d",
            libInfo.funcKeysCount); // количество функциональных клавиш
        TextOut(dc, 200, 50, BUF, lstrlen(BUF)); // вывод на экран (x=200, y=50)
        wnsprintf(BUF, BUFSIZE - 1,
            "L3 cache length: %d",
            (char)(libInfo.cpuidInfo)); // длина строки КЭШа
        TextOut(dc, 200, 70, BUF, lstrlen(BUF));
        EndPaint(hWnd, &ps); // завершение контекста
        break;
    case WM_DESTROY: // уничтожение окна (возвращается функцией ExitProcess)
        PostQuitMessage(EXIT_SUCCESS); // завершение работы
        break;
    default: // обработка любых других сообщений стандартным образом
```

```
        return DefWindowProc(hWnd, uMsg, wParam, lParam);
    }
    return 0;
}

// функция, запускающаяся в рамках созданного потока
DWORD WINAPI ThreadFunc(void* lParam) {
    GetInfo((library_info*)lParam);
    return 0;
}

// точка входа графического приложения
int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrev, LPSTR CmdLine, int nCmdShow) {
    HANDLE hThread; // дескриптор потока
    DWORD IDThread; // идентификатор потока
    WNDCLASS wcl; // класс окна
    HWND hWnd; // дескриптор окна
    ATOM regReturn; // значение, возвращаемое функцией RegisterClass
    MSG msg; // сообщение

    // задание параметров окна
    wcl.hInstance = hInstance; // дескриптор
    wcl.lpszClassName = "KeyCounterWindow"; // название класса
    wcl.lpfnWndProc = WndProc; // функция-обработчик сообщений
    wcl.style = CS_VREDRAW | CS_HREDRAW; // стиль
    wcl.hIcon = NULL; // отключение пиктограммы
    wcl.hCursor = NULL; // отключение курсора
    wcl.lpszMenuName = NULL; // отключение меню
    wcl.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH); // цвет фона
    regReturn = RegisterClass(&wcl); // регистрация класса

    // создание окна
    hWnd = CreateWindow(
        "KeyCounterWindow", // вышеуказанное название класса
        "Function Keys Counter", // заголовок окна
        WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN | WS_CLIPSIBLINGS, // стили окна
        0, 0, // координаты левого верхнего угла окна при выводе на экран
        600, 200, // ширина и высота окна
        HWND_DESKTOP, // рабочий стол как родительское окно
        NULL, // дочернее окно
        hInstance, // экземпляр приложения
        NULL); // указатель на данные создания окна

    // создание потока
    hThread = CreateThread(
        NULL, // запрет наследования дескриптора
        0, // начальный размер стека
        (LPTHREAD_START_ROUTINE)ThreadFunc, // указатель на функцию
        (LPVOID)&libInfo, // переменная, передаваемая потоку
        0, // флаг моментального запуска потока
        &IDThread); // переменная, в которую будет записан идентификатор потока

    // обработка ошибок при регистрации класса, создании окна или потока
    if (!regReturn || !hWnd || !hThread)
        return GetLastError();

    WaitForSingleObject(hThread, INFINITE); // бесконечное ожидание завершения потока
```

```
    CloseHandle(hThread); // удаление дескриптора потока
    ShowWindow(hWnd, SW_SHOWNORMAL); // отображение окна
    UpdateWindow(hWnd); // обновление окна

    // цикл, обрабатывающий сообщения
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg); // преобразование в ASCII
        DispatchMessage(&msg); // отправка сообщения в оконную процедуру
    }

    ExitProcess(msg.wParam); // завершение процесса и его потоков
}
```

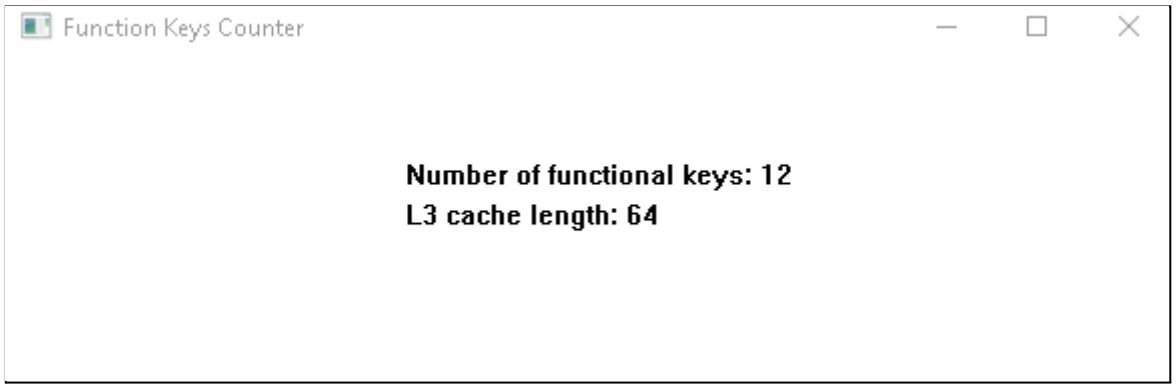
library.cpp

```
#include "stdafx.h"
#include "library.h"
#define __cpuid __mycpuidex
#define c_cpuid __mycpuidex

// получение информации о длине строки КЭШа и количестве функциональных клавиш
LIBRARY_API int GetInfo(library_info* info) {

    info -> funcKeysCount = GetKeyboardType(2); // количество функциональных клавиш

    // проверка, поддерживается ли CPUID
    __asm {
        PUSHFD // размещение регистра EFLAGS в стеке
        POP EAX // извлечение значения EFLAGS в EAX
        MOV EBX, EAX // сохранение значения в EBX
        XOR EAX, 00200000h // изменение 21-го бита
        PUSH EAX // размещение нового значения в стеке
        POPFD // сохранение нового значения в EFLAGS
        PUSHFD // снова размещение EFLAGS в стеке
        POP EAX  // значение EFLAGS теперь в EAX
        cmp EAX, EBX // проверка 21-го бита
        JZ RET_ARG // бит не изменился - CPUID не поддерживается
    }

    // получение длины строки КЭШа (схема AMD)
    __asm {
        MOV EAX, 0x80000006
        cpuid
        and EDX, 0xFF
        MOV EAX, info
        MOV[EAX], EDX
        JMP RET_ARG
    }

    __asm { RET_ARG: }
}

// точка входа динамической библиотеки
BOOL WINAPI DllMain(HMODULE hModule, DWORD fdwReason, LPVOID lpReserved) {
    switch (fdwReason) {
```

```
    case DLL_PROCESS_DETACH: // отключение DLL
        break;
    }
    return TRUE;
}
```

6. Тестирование

Результат выполнения программы:



Результат без динамической библиотеки: