|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Расчетнографическое задание | | |
| по дисциплине «Управление ресурсами в вычислительных системах» | | |
| Управления ресурсами в ОС Windows | | |
|  | | |
|  | Вариант 6 | ПМ-13 Вострецова Е.В |
|  |
|  |
|  |
|  |
|  |
| Преподаватели | СТАСЫШИН В. М. |
|  | сИВАК м.а. |
| Новосибирск, 2024 | | |

1. Задание

Разработка простейшего Windows-приложения определяющего максимальную ширину экрана в пикселях и поддержку процессором технологии MMX.

Выбран 4 уровень РГЗ (вариант 6).

Данное приложении должно:

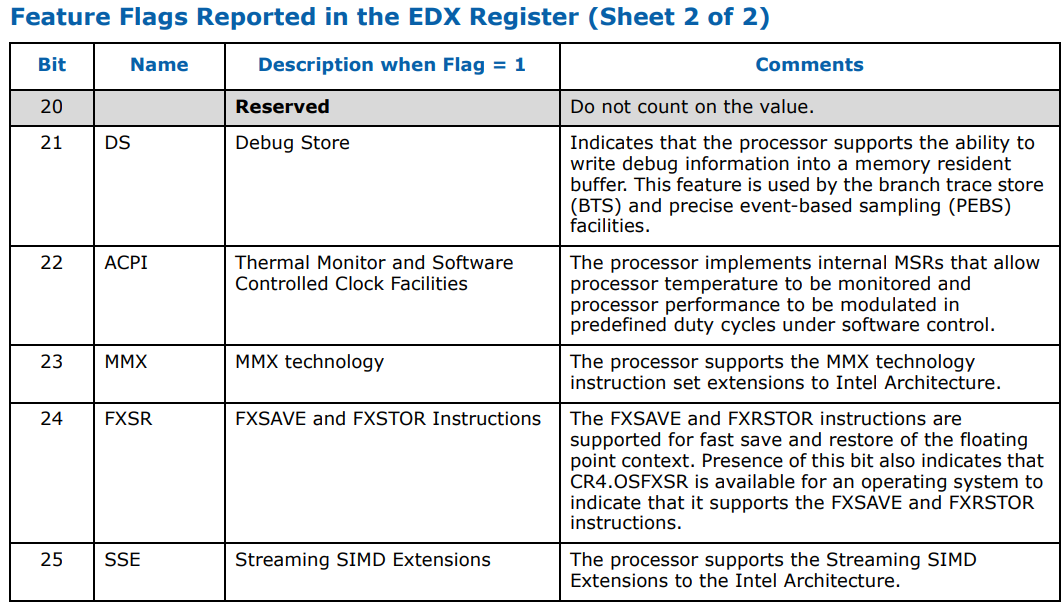
* создавать дочерний поток, в котором будет вызываться функция, реализующая требования варианта
* Обладать графическим интерфейсом
* Функция из динамической библиотеки, реализующей функции выбранного варианта
* Определять поддержку процессором технологии SSE, реализована при помощи ассемблерной вставки в функцию реализующей функционал основного задания.

1. Анализ задачи

2.1. Определение максимальной ширины экрана в пикселях

Чтобы определить максимально возможную ширину экрана в пикселях нужно вызвать функцию **GetSystemMetrics** с параметром **SM\_CYSCREEN**

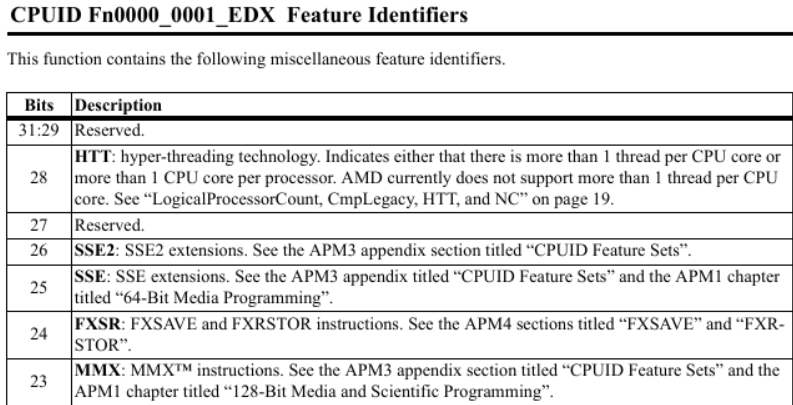
2.2. Определение наличия поддержки SSE технологии для процессоров Intel



Исходя из документации по утилите **CPUID** для процессоров компании Intel нужно

* Приравнять регистр **EAX** единице
* вызвать утилиту **CPUID**
* **SSE** технология поддерживается если 25 бит регистра **EDX** равен 1

2.3 Определение наличия поддержки MMX технологии для процессоров AMD



Исходя из документации по утилите **CPUID** для процессоров компании AMD нужно

* Приравнять регистр **EAX** единице
* вызвать утилиту **CPUID**
* **SSE** технология поддерживается если 25 бит регистра **EDX** равен 1.

1. Общий алгоритм программы

Создание и отображении окна приложения. Запуск цикла обработки сообщений.

Создание дочернего потока, запуск процедуры в рамках созданного потока.

Загрузка динамической библиотеки. Вызов функции реализующей требования варианта:

- Определение максимальной высоты экрана в пикселях

- Определение наличия поддержки SSE технологии

Вывод полученных результатов при помощи графического интерфейса.

1. Описание разработанных функций

4.1. Функция динамической библиотеки

|  |  |
| --- | --- |
| Функция | Описание |
| void GetInformation (int& maxHeight, bool& isSSESupported) | Определение максимальной высоты экрана в пикселях и наличие поддержки SSE технологии процессором. |

4.2. Функции основного приложения

|  |  |
| --- | --- |
| Функция | Описание |
| DWORD WINAPI ThreadFunc(void \*) | Функция, выполняющаяся в отдельном потоке. Она включает в себя следующие этапы:  1. Подключение динамической библиотеки  2. Загрузка динамической библиотеки  3. Выполнение функцию из библиотеки  4. Высвобождаем библиотеку  5. Интерпретируем полученные данные |
| LRESULT CALLBACK WindowFunc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam) | Процедура обработки сообщений. Шаги:  1. Создание шрифта  2. Создание потока в рамках которого исполняется функция **ThreadFunc**.  3. Инициализация контекста устройства, т.е. вывод строк **ScreenInfo** и **SSEInfo** |
| int WINAPI WinMain(HINSTANCE hThisInst, HINSTANCE hPrevInst, LPSTR str, int nWinMode) | Главная функция программы. Она включает в себя следующие этапы:  1. Регистрация класса окна  2. Создание главного окна  3. Цикл обработки сообщений:  - Получение сообщения  - Посылка сообщения в нужную процедуру |

1. Спецификация

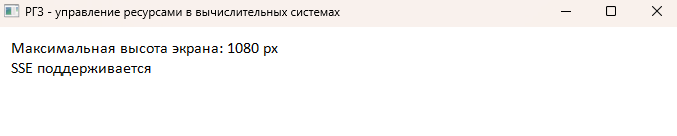
Программа находится в папке “РГЗ\_ПМ-13\_Вострецова\_Е.В ”.

Чтобы собрать программу нужно открыть Developer Command Prompt и перейти в папку с проектом. Запустить скрипт “compile.bat”

Для запуска программы нужно запустить файл “driver.exe”.

1. Тесты

Процессор: AMD Ryzen 7 3750H with Radeon Vega Mobile Gfx 2.30 GHz



Процессор: Intel® Core™ i7-3520m

1. Текст программы

**Файл info.cpp**

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #define UNICODE  #include <stdio.h>  #include <Windows.h>  // Функция динамической библиотеки  extern "C" \_\_declspec(dllexport) void GetInformation(int& maxHeight, bool& isSSESupported)  {  // Определяем максимальную высоту окна (в пикселях)  maxHeight = GetSystemMetrics(SM\_CYSCREEN);    // Определяем наличие поддержки SSE технологии  int fSupported;  \_\_asm  {  mov eax, 1 // CPUID уровня 1  cpuid  and edx, 0x2000000 // проверяем 25 бит регистра EDX  mov fSupported, edx // если SSE поддерживается, то EDX[25] == 1  }  if (fSupported != 0)  isSSESupported = true;  } |

**Файл driver.cpp**

|  |
| --- |
| #define \_CRT\_SECURE\_NO\_WARNINGS  #define UNICODE  #include <windows.h>  #include <string>  #include <stdio.h>  #include <Windows.h>  LPCWSTR szClassName = L"MyClass";  LPCWSTR szTitle = L"РГЗ - управление ресурсами в вычислительных системах";  TCHAR ScreenInfo[256];  TCHAR SSEInfo[256];  // Функция выполняющаяся в отдельном потоке  DWORD WINAPI ThreadFunc(void\*)  {  typedef int(\*ImportFunction)(int&, bool&);  ImportFunction DLLGetInformation; // Подключаем динамическую библиотеку  HINSTANCE hntLib = LoadLibrary(TEXT("info.dll"));  DLLGetInformation = (ImportFunction)GetProcAddress(hntLib, "GetInformation");  int maxHeight = 0;  bool isSSESupported = false;  DLLGetInformation(maxHeight, isSSESupported); // Получаем данные  FreeLibrary(hntLib); // Высвобождаем библиотеку  // Интерпретация выходных данных функции DLLGetInformation  wsprintf(ScreenInfo, L"Максимальная высота экрана: %d px", maxHeight);  if (isSSESupported)  wsprintf(SSEInfo, L"SSE поддерживается");  else  wsprintf(SSEInfo, L"SSE не поддерживается");  return 0;  }  // Процедура обработки сообщений  LRESULT CALLBACK WindowFunc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM lParam)  {  PAINTSTRUCT ps;  HDC hDC;  // Создание шрифтра  HFONT hFont = CreateFont(20, 0, 0, 0, FW\_THIN, FALSE, FALSE, FALSE, DEFAULT\_CHARSET, OUT\_OUTLINE\_PRECIS,  CLIP\_DEFAULT\_PRECIS, CLEARTYPE\_NATURAL\_QUALITY, VARIABLE\_PITCH, TEXT("Calibri"));  switch (msg)  {  // Создание потока  case WM\_CREATE:  HANDLE hThread;  DWORD IDThread;  hThread = CreateThread(NULL, 0, ThreadFunc, NULL, 0, &IDThread);  WaitForSingleObject(hThread, INFINITE);  CloseHandle(hThread);  break;  // Закрытие окна  case WM\_DESTROY:  PostQuitMessage(0);  break;  // Инициализация контекста устройства  case WM\_PAINT:  hDC = BeginPaint(hWnd, &ps);  SelectObject(hDC, hFont); // Меняем шрифт  TextOut(hDC, 15, 10, ScreenInfo, 256);  TextOut(hDC, 15, 30, SSEInfo, 256);  EndPaint(hWnd, &ps);  break;  default:  return DefWindowProc(hWnd, msg, wParam, lParam);  }  return 0;  }  // Главная функция Windows-приложения  int WINAPI WinMain(HINSTANCE hThisInst,  HINSTANCE hPrevInst, LPSTR str, int nWinMode)  {  MSG msg;  WNDCLASS wcl;  HWND hWnd;  // Создание класса окна  wcl.hInstance = hThisInst;  wcl.lpszClassName = szClassName;  wcl.lpfnWndProc = WindowFunc;  wcl.style = CS\_HREDRAW | CS\_VREDRAW;  wcl.hIcon = LoadIcon(NULL, IDI\_APPLICATION);  wcl.hCursor = LoadCursor(NULL, IDC\_ARROW);  wcl.lpszMenuName = NULL;  wcl.cbClsExtra = 0;  wcl.cbWndExtra = 0;  wcl.hbrBackground = (HBRUSH)GetStockObject(WHITE\_BRUSH);  // Регистрация класса окна  RegisterClass(&wcl);  // Создание окна на базе его класса  hWnd = CreateWindow(szClassName, szTitle,  WS\_OVERLAPPEDWINDOW | WS\_CLIPCHILDREN |  WS\_CLIPSIBLINGS,  100, 50, 700, 150, HWND\_DESKTOP,  NULL, hThisInst, NULL);  // Отображение окна  ShowWindow(hWnd, nWinMode);  UpdateWindow(hWnd);  // Цикл обработки сообщений  while (GetMessage(&msg, NULL, 0, 0))  {  TranslateMessage(&msg);  DispatchMessage(&msg);  }  return msg.wParam;  } |

**Файл compile.bat**

|  |
| --- |
| if exist \*.exe del \*.exe  if exist \*.dll del \*.dll  cl /EHsc driver.cpp kernel32.lib user32.lib gdi32.lib advapi32.lib  cl /LD info.cpp kernel32.lib user32.lib gdi32.lib advapi32.lib  del \*.obj \*.lib \*.exp |