Ćwiczenie 5.

TEMAT:

Tworzenie projektu asemblerowego dla środowiska Visual Studio 2008.

CEL:

Celem ćwiczenia jest poznanie możliwości VS 2008 w zakresie tworzenia i uruchamiania aplikacji z kodem mieszanym w języku C ++ oraz asemblerze. W założeniu aplikacja składa się z dwóch elementów – aplikacji napisanej w j. C++ oraz biblioteki DLL napisanej w asemblerze dla środowiska Windows. Konstrukcja projektu zakłada możliwość wywoływania funkcji bibliotecznych napisanych w asemblerze z poziomu aplikacji oraz pokazuje prawidłową konfigurację środowiska umożliwiającą debugowanie kodu do poziomu asemblera, obserwację stanu rejestrów i flag procesora czy obszarów pamięci danych.

ZAŁOŻENIA:

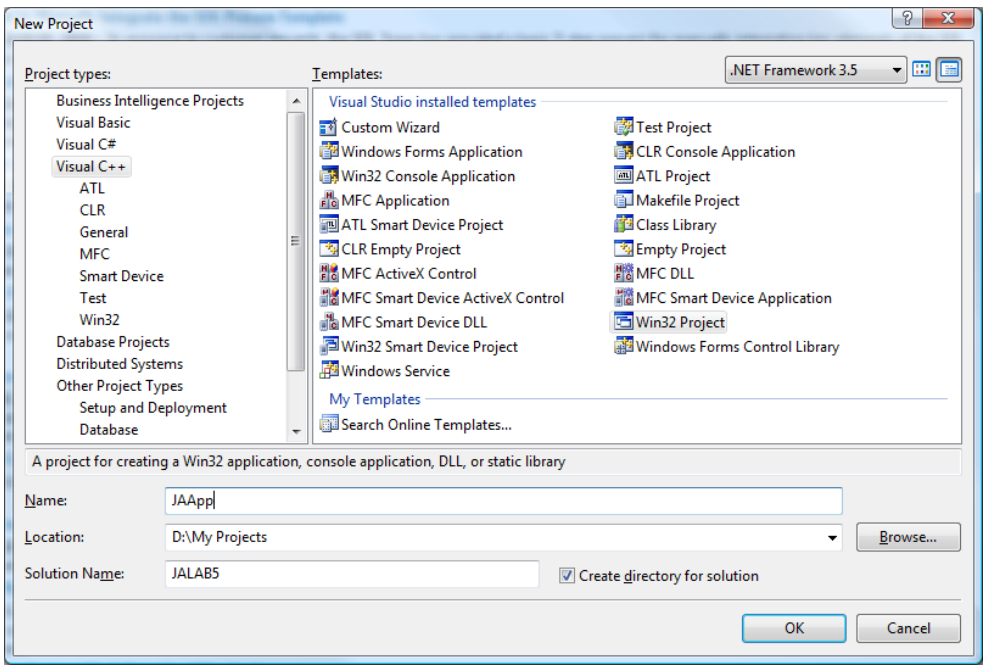
W środowisku VS 2008 zakładamy solucję składającą się z dwóch projektów:

- Projekt aplikacja WIN32 w j. C++,
- Projekt biblioteka DLL w asemblerze,

W bibliotece DLL utworzona zostanie funkcja asemblerowa, której wywołanie i przekazywanie parametrów wystąpi w aplikacji.

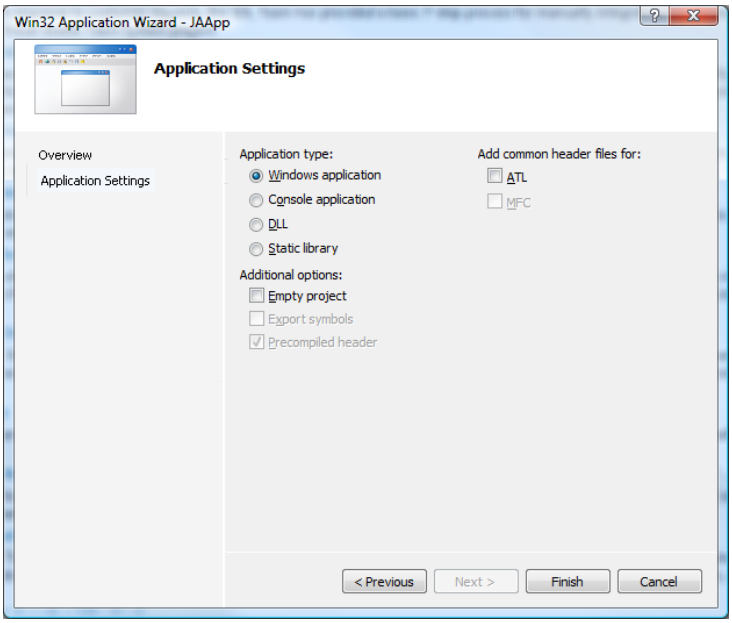
WYKONANIE:

W środowisku VS 2008 tworzymy nową solucję o nazwie JALAB5 wybierając nowy Win32 Project o nazwie JAApp jak na Rys. 1:



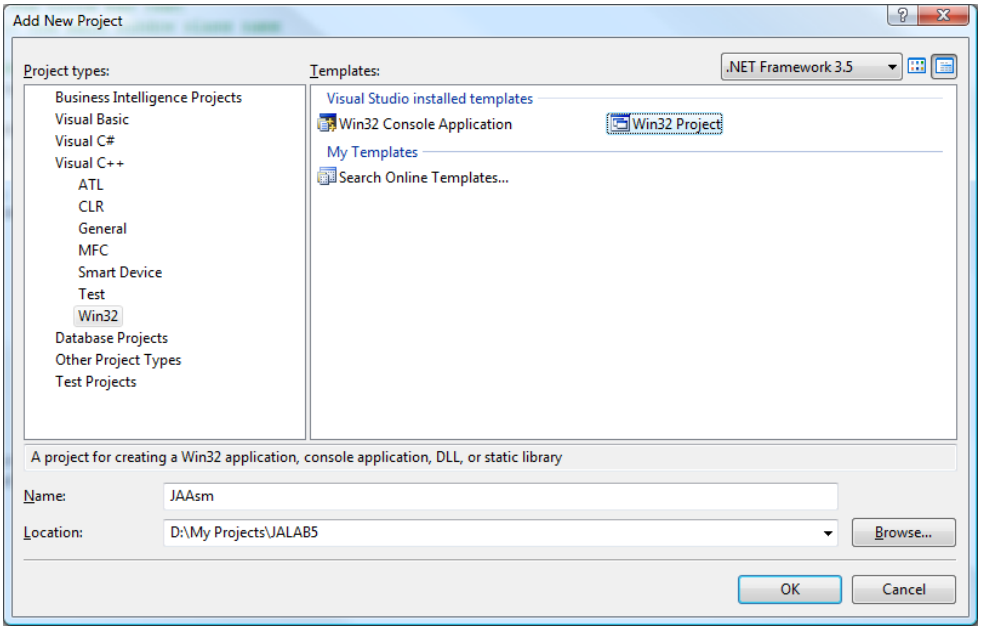
Rys.1 Nowy Projekt

Z opcjami jak na Rys. 2:



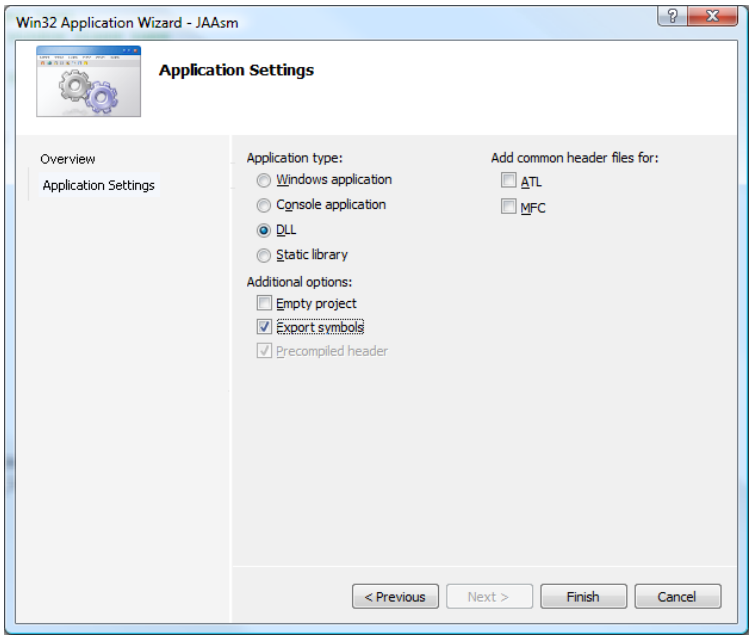
Rys. 2 Opcje projektu aplikacji C++

Następnie dodajemy nowy projekt JAAsm jak na Rys. 3:



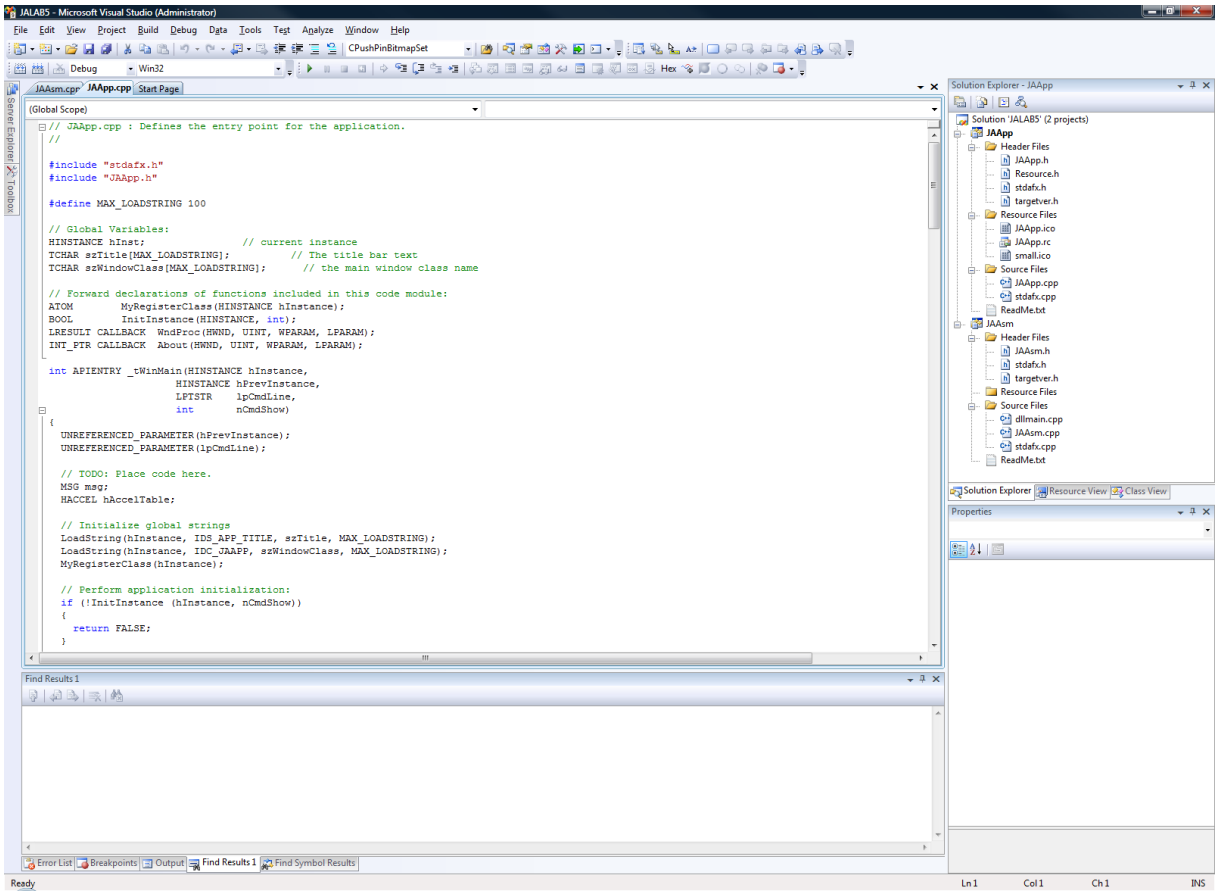
Rys. 3 Dodawanie nowego projektu do solucji

Z następującymi opcjami przedstawionymi na Rys.4:



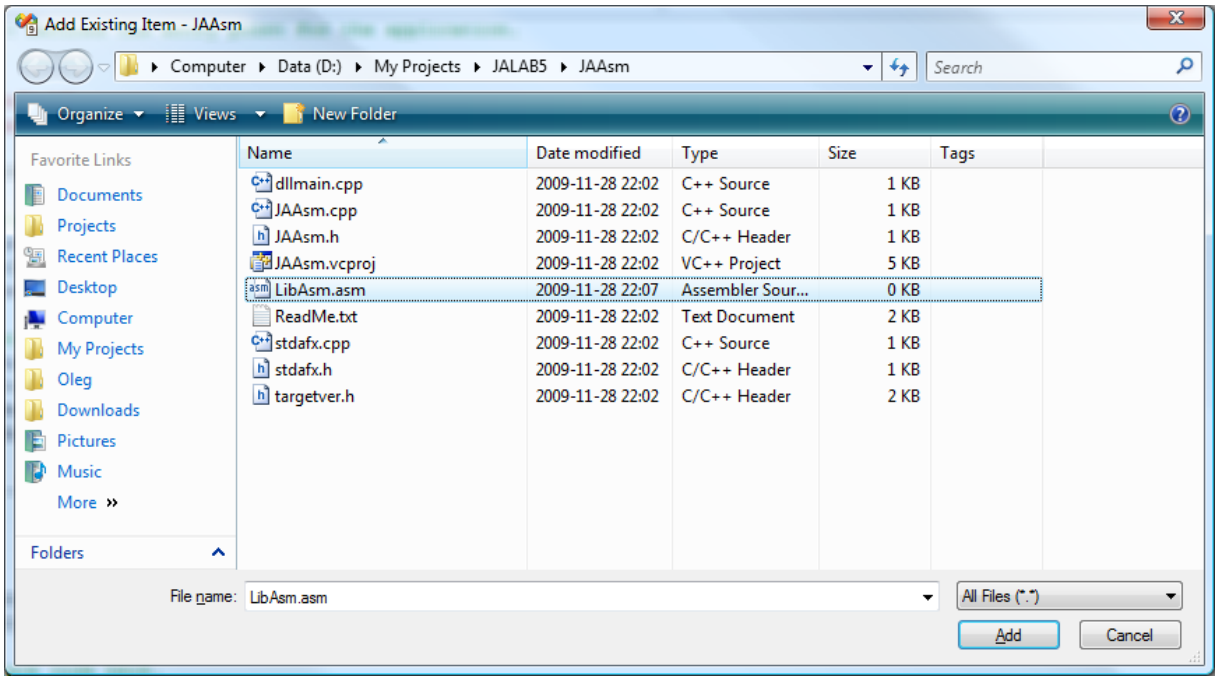
Rys. 4 Opcje projektu DLL

Po dodaniu nowego projektu wygląd środowiska przedstawia Rys.5:



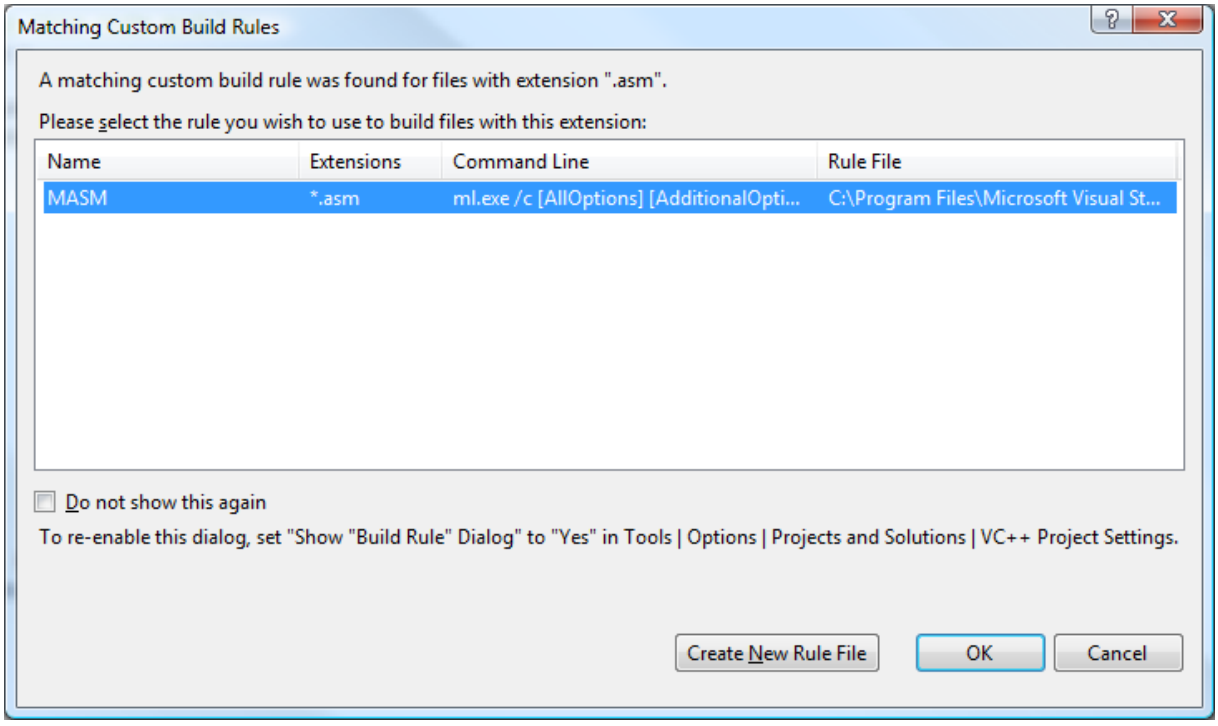
Rys. 5 Wygląd solucji składającej się z dwóch projektów

Następnie tworzymy za pomocą programu Notepad pusty plik o nazwie LabAsm.asm i umieszczamy go w katalogu projektu JAAsm. Wybieramy projekt JAAsm za pomocą myszki a następnie po wciśnięciu prawego przycisku myszki wybieramy menu **Add Existing Item**. Pojawia się okno dialogowe przedstawione na Rys.7. Zaznaczamy plik LibAsm.asm i dodajemy go do projektu.



Rys. 6 Dodanie pliku źródłowego *.asm do projektu DLL

Środowisko VS 2008 rozpoznaje rozszerzenie *.asm i wyświetlony zostaje dialog Rys. 8:



Rys. 8 Określenie sposobu kompilacji pliku *.asm

Zaznaczamy MASM oznaczającą, że dany plik asm będzie kompilowany za pomocą wbudowanego do środowiska makroasemblera ml.exe i naciskamy OK.

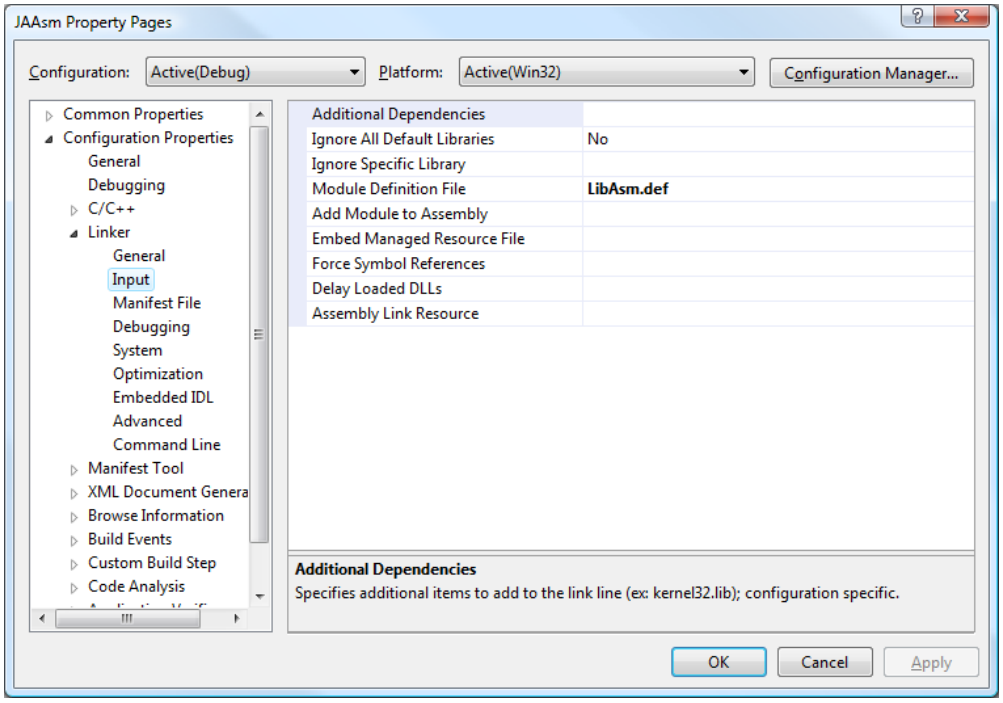
EKSPORT FUNKCJI.

Eksport funkcji napisanych w module LibAsm.asm do użycia w aplikacji JAApp można wykonać na kilka sposobów. Jednym z nich jest utworzenie pliku LibAsm.def w katalogu projektu JAAsm o przykładowej treści:

```
LIBRARY      "JAAsm"
EXPORTS
MyProc1
MyProc2
```

Gdzie eksportowi podlegają dwie funkcje o nazwach My Proc1 i MyProc2.

Plik LibAsm.def należy dodać do projektu JAAsm jak poprzednio poprzez wybranie opcji projektu Add Existing Item. Następnie w ogólnych właściwościach projektu należy wpisać jego nazwę w opcjach linkera jak na Rys. 9



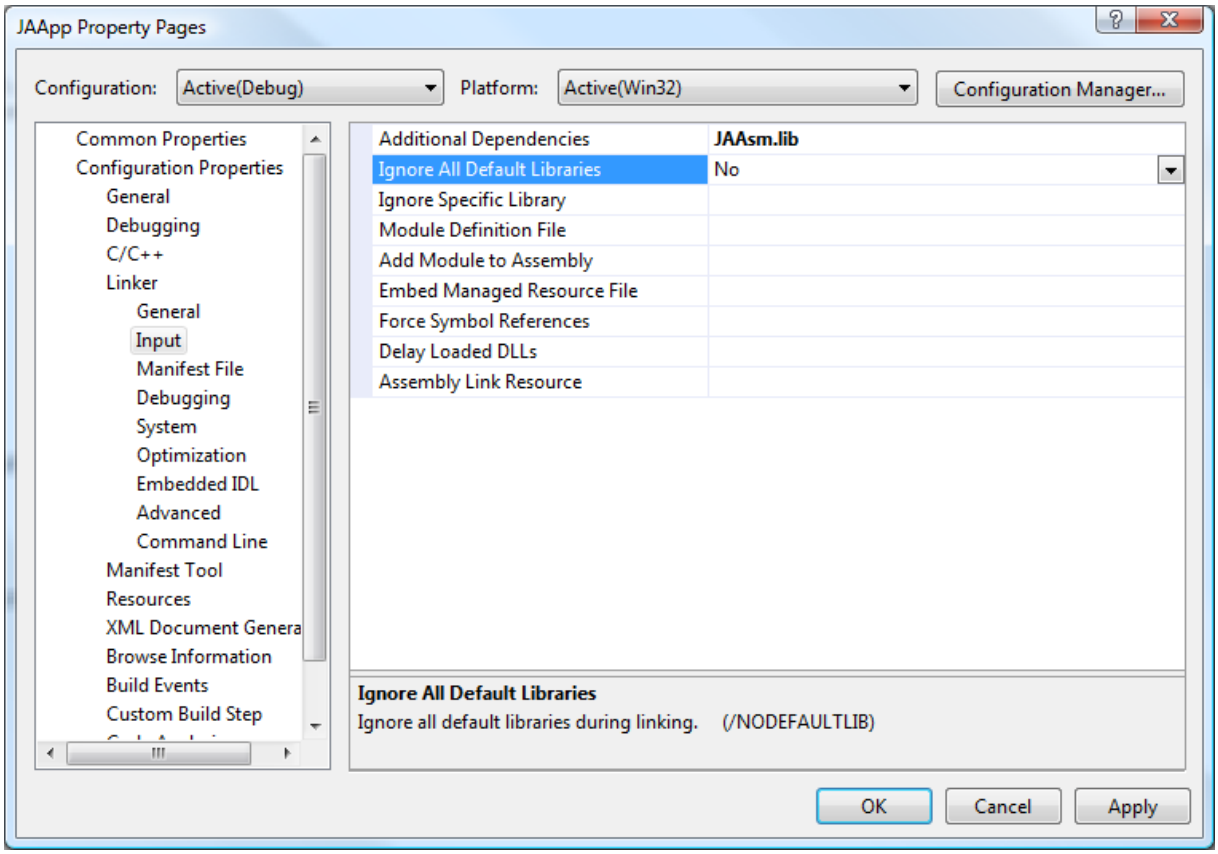
Rys. 9 Dołączenie pliku *.def do projektu DLL

Po wykonaniu tych czynności możemy przystąpić do pisania procedur asemblerowych MyProc1 i MyProc2 jakie będą wywoływane z poziomu aplikacji projektu JAApp.

W aplikacji JAApp funkcje biblioteczne można wywoływać na dwa sposoby:

- Statycznie poprzez linkowanie pliku LibAsm.lib w opcjach linkera aplikacji JAApp Rys. 10.
- Dynamicznie ładując bibliotekę JAAsm.dll utworzona za pomocą projektu JAAsm w miejscu gdzie będzie wywoływana jedna eksportowanych funkcji za pomocą funkcji API HLIBRARY LoadLibrary (LPCWSTR lplibFileName)

Statyczny model wywołania funkcji MyProc1 z biblioteki DLL w kodzie źródłowym JAApp.cpp przedstawia poniższy przykład:



Rys. 10 Statyczne linkowanie biblioteki DLL do projektu

Przykład procedury asemblerowej MyProc1 wywoływanej z poziomu aplikacji JAApp:

```
.486
.model flat, stdcall

.code

MyProc1 proc x: DWORD, y: DWORD

      xor eax,eax
      mov eax,x
      mov ecx,y
      ror ecx,1
```

```
      shld eax,ecx,2
      jnc ET1
      mul y
      ret
ET1:  mul x
      neg y
      ret

MyProc1 endp

end
```

W projekcie JAApp zawartość pliku JAApp.cpp jest następująca:

```
// JAApp.cpp : Defines the entry point for the application.
//

#include "stdafx.h"
#include "JAApp.h"

#define MAX_LOADSTRING 100

// Global Variables:
// Forward declarations of functions included in this code module:

int APIENTRY _tWinMain(HINSTANCE hInstance,
                       HINSTANCE hPrevInstance,
                       LPTSTR    lpCmdLine,
                       int       nCmdShow)
{
      UNREFERENCED_PARAMETER(hPrevInstance);
      UNREFERENCED_PARAMETER(lpCmdLine);

      // TODO: Place code here.
      MSG msg;
      HACCEL hAccelTable;

      // Initialize global strings
      LoadString(hInstance, IDS_APP_TITLE, szTitle, MAX_LOADSTRING);
      LoadString(hInstance, IDC_TEST, szWindowClass, MAX_LOADSTRING);
      MyRegisterClass(hInstance);

      int x = 3,y = 4, z = 0;

      z = MyProc1 (x, y);  // wywołanie procedury asemblerowej z biblioteki
      // Perform application initialization:
      if (!InitInstance (hInstance, nCmdShow)) {
            return FALSE;
      }
      hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_TEST));

      // Main message loop:
      while (GetMessage(&msg, NULL, 0, 0)){
            if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg)){
                  TranslateMessage(&msg);
                  DispatchMessage(&msg);
            }
      }
      return (int) msg.wParam;
}
```

W projekcie JAApp zawartość pliku JAApp.h jest następująca:

```
#pragma once

#include "resource.h"
#include <windows.h>

extern "C" int _stdcall MyProc1 (DWORD x, DWORD y);
```

Po skompilowaniu całości solucji możliwe będzie wywołanie funkcji bibliotecznej MyProc1 z aplikacji JAAsm.exe a w trakcie debugowania w trybie krokowym możliwe jest wykonywanie pojedynczych rozkazów asemblerowych i obserwowanie zmian rejestrów procesora i flag.

Dynamiczny model wywołania funkcji MyFunc1 z biblioteki DLL w kodzie źródłowym JAApp.cpp przedstawia poniższy przykład:

```
typedef DWORD (*MYPROC1)(DWORD, DWORD);

MYPROC1     MyFunc1;
HMODULE     hLib;
DWORD       dwZ,dwX,dwY;

if ((hLib = LoadLibrary  ("JAAsm.dll")) != NULL) {
  MyFunc1 = (MYPROC1)GetProcAddress (hLib,"MyProc1");
  if (MyFunc1 != NULL)
    dwZ = MyFunc1 (dwX,dwY);

  FreeLibrary (hLib);
}
```

Projekt JAApp w Windows Forms Applications.

W przypadku użycia jako aplikacji wywołującej **Windows Forms Application** istnieje konieczność modyfikacji standardowych parametrów linkera aby możliwe było debugowanie krokowe.

1. Zaczynamy od utworzenia nowego projektu:
   File -> New -> Project
   W eksploratorze wybieramy **Visual C++ -> CLR -> Windows Forms Application**
   i podajemy nazwę JAApp**.**
2. Następnie dodajemy projekt LbAsm tak jak opisano powyżej
3. Teraz klikamy PPM na **Solutions** (eksplorator z lewej strony ekranu) i wybieramy z końca **Properities** po czym wybieramy **Project Dependencies** i w rozwijalnym menu wybieramy **JAApp** oraz zaznaczamy poniżej **biblioteka.** Zabieg ten służy ustaleniu zależności pomiędzy projektami.
4. Następnie klikamy PPM na **JAApp** (i znów eksplorator z lewej strony ekranu) i wybieramy z końca **Properities** i wybieramy **Configuration Properities -> General** , a następnie w oknie **Common Languge** zmieniamy na z **(/clr :pure)** na **(/clr)**.
5. Następnie w **Configuration Properities -> Linker -> Input** w oknie wybieramy trzy kropki(podanie ścieżki) w **Additional Dependencies** i podajemy następującą ścieżkę:
   ..\Debug\LibAsm.lib co potwierdzamy poprzez **OK.**