|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| logowydzialu | | Instytut Informatyki Politechniki Śląskiej  Zespół Mikroinformatyki i Teorii Automatów Cyfrowych  **Laboratorium JA** | | logoii | |
| **Rok akademicki** | | **Rodzaj studiów\*: SSI/NSI/NSM** | **Numer ćwiczenia:** | **Grupa** | **Sekcja** |
| **2017/2018** | | **SSI** | **3** | **6** | **1** |
| **Data i godzina planowana ćwiczenia:**  dd/mm/rrrr - gg:mm | | **13/03/2018-11:45** | **Prowadzący**:  OA/AO | **OA** | |
| **Data i godzina wykonania ćwiczenia:**  dd/mm/rrrr - gg:mm | | **13/03/2018-11:45** |
| ***Sprawozdanie*** | | | | | |
| **Temat ćwiczenia:**  Zapoznanie się z oprogramowaniem MASM 32, WINDBG OLLYDBG RADASM x86 (INTEL). | | | | | |
| **Skład sekcji:** | 1.Bartłomiej Krasoń | | | | |

# Cel

Celem ćwiczenia jest poznanie innych niż Microsoft VC asemblerów i programów debugerów procesorów x86.

*~Źródło – Instrukcja ćwiczenia LAB3*

# Rozwiązanie

Odpowiedzi na zadane pytania w instrukcji ćwiczenia:

**Pytanie 1**: Opisz parametry wywołania funkcji MessageBox i ExitProcess.

**Odp.:**

int WINAPI MessageBox(

\_In\_opt\_ HWND    hWnd, --uchwyt do okna wywołującego

\_In\_opt\_ LPCTSTR lpText, --tekst wiadomości do wyświetlenia

\_In\_opt\_ LPCTSTR lpCaption, --tytuł wyskakującego okna

\_In\_     UINT    uType --kombinacja flag określających zawartość oraz

zachowanie okna

);

void WINAPI ExitProcess(

\_In\_ UINT uExitCode --kod zakończenia procesu i wszystkich jego wątków

);

**Pytanie 2:** Przedstaw dwa różne sposoby wywołania funkcji MessageBox (przy użyciu push i invoke). Przeprowadzić kompilację programu z linii poleceń wykorzystując przełącznik /coff oraz opcjonalnie z przełącznikiem /c i bez niego.

**Odp.:**

a.) Wywałoanie funckji MessageBox przy użyciu **push/call**:

push MB\_OK

push offset Tytul\_okna

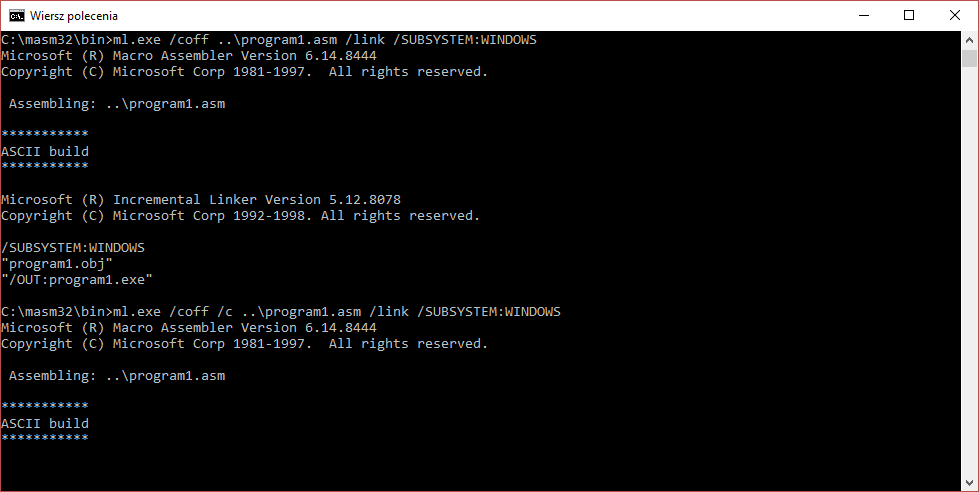
push offset Tekst\_w\_oknie

push 0

call MessageBox

b.) Wywołanie funkcji MessageBox przy użyciu **invoke**:

invoke MessageBox, MB\_OK, offset Tytul\_okna, offset Tekst\_w\_oknie, 0

Przeprowadzona kompilacja programu1.asm w konsoli:

**Pytanie 3:** Czy ml.exe automatycznie wywołuje linkera? Czy bez użycia dodatkowych przełączników program kompiluje/linkuje się poprawnie?

**Odp.:**Bez użycia dodatkowych przełączników MASM automatycznie wywołuje linkera i tworzy plik exe. Można użyć przełącznika /c który zapewnia asemblację bez linkowania.

**Pytanie 3:** Do czego służą użyte przełączniki?

**Odp.:**

Uprzednio użyte przełączniki służą do:

**/coff** - generuje plik obiektowy formatu COFF

**/c** - zapewnia asemblację bez linkowania

**/SUBSYSTEM:WINDOWS** - opcja linkera, określa sposób uruchomienia pliku exe

**/link** - umożliwia dodawanie/zmienianie opcji linkera

**/Fl** - generuje listing

**/Fm** - generuje mapę

**Pytanie 5:** Do czego służy dyrektywa .NOLIST?

**Odp.:** Powoduje wstrzymanie listowania programu źródłowego.

**Pytanie 6:** Do czego służy dyrektywa .NOCREF?

**Odp.:** Powoduje wstrzymanie listowania nazw symbolicznych w tablicy symboli listingu asemblacji.

**Pytanie 7:** Do czego służy dyrektywa .LISTALL?

**Odp.:** Powoduje że w listingu asemblacji umieszczone zostaną wszystkie bloki programu źródłowego.

**Pytanie 8:** Co znajduje się w pliku map?

**Odp.:** W pliku \*.map znajduje się informacja o modułach jakie linker zaimportował z dołączonych bibliotek oraz rozmieszczenie segmentów pamięci, z której korzysta program.

**Pytanie 9:** Zmodyfikować odpowiednie pliki \*.bat z katalogu c:\masm32\bin tak, aby kompilacja ze środowiska qeditor powodowała automatyczne wygenerowanie plików lst i map. Jakich modyfikacji i w których plikach dokonałeś?

**Odp.:** Modyfikacji dokonałem tylko w pliku ”bldall.bat” dodając w linii wywołania - przełącznik /Fl w celu tworzenia listingu oraz w linii dołączania opcji linkera - /map w celu utworzenia mapy.

**Pytanie 10:** Skompilować i uruchomić program 2. Czy przy kompilacji wystąpiły jakieś błędy? Jakie pliki nagłówkowe dodałeś do swojego programu, żeby go skompilować?

**Odp.:** Tak przy kompilacji wystąpił błąd kodu: „A2006: undefined symbol : wsprintf”. Dodałem następujące linie kodu (załączenie pliku nagłówkowego i biblioteki user32) w celu wyeliminowania błędu:

- include c:\masm32\include\user32.inc

- includelib c:\masm32\lib\user32.lib

**Pytanie 11:** Program 2 używa debuggera dbgwin.exe do śledzenia wartości jednego z rejestrów.W jaki sposób odbywa się to debuggowanie - porównaj ze znanymi Ci sposobami debuggowania programu.

**Odp.:** Debuggowanie to odbywa się automatycznie i polega na wypisywaniu interesujących nas informacji dzięki zastosowaniu makr (w tym przypadku zawartości rejestru EAX). Pierwszy raz spotkałem się z tego typu debuggowaniem, gdyż na ogół korzystam z debuggerów, opierających swoje działanie na breakpointach jak jest to np. w Visual Studio.

**Pytanie 12:** Jakie funkcje/makra wykorzystuje program 2 do debuggowania w dbgwin.exe? Podaj jeszcze co najmniej 3 inne funkcje służące do podobnych celów (np. analizując plik nagłówkowy dla debuggera).

**Odp.:** Program 2 wykorzystuje makra:

- PrintDec – wypisuje zawartość danego rejestru,

- PrintText – wypisuje łańcuch znaków,

Inne funkcje/makra służące do podobnych celów to np.: PrintString, PrintDouble czy DumpMem.

**Pytanie 13:** Jaki plik nagłówkowy oraz jaką bibliotekę musi inkludować plik programu 2, aby makra debuggera działały poprawnie?

**Odp.:** Program 2 musi indukować plik nagłówkowy: debug.inc oraz bibliotekę debug.lib, z odpowiednich folderów.

**Pytanie 14:** Od jakiego adresu rozpoczyna się kod programu?

**Odp.:** Kod programu rozpoczyna się od adresu 00401000h

**Pytanie 15:** Czy tzw. plik symboli programu jest potrzebny debuggerowi? Czy i gdzie ustawiamy ścieżkę do tego pliku w debuggerze?

**Odp.:** Plik symboli nie jest potrzebny debuggerowi, ale może się okazać pomocny osobie używającej debuggera. Ścieżkę do tego pliku ustawiamy następująco:

a.) w WinDbg: File -> Symbol File Path … lub Ctrl+S

b.) w OllyDbg: Debug -> Select path for symbols

**Pytanie 16:** W jaki sposób ustawiamy pułapkę w testowanym debuggerze?

**Odp.:**

a.) w WinDbg: (wybór linii) -> F9 lub uaktywnienie flagi edycji breakpointów – łapka na górnym pasku

b.) w OllyDbg: (wybór linii) -> F2 lub (wybór linii) -> PPM -> Breakpoint ->Toggle

**Pytanie 17:** Do czego służy INT 3?

**Odp.:** Jest rozkazem wykorzystywanym przez debugger – powoduje przerwanie w danym momencie wykonywania programu. Możemy sami go używać.

# Wnioski

Programy pisane w asemblerze mogą być debuggowane na kilka różnych sposobów z wykorzystaniem różnych programów. W zależności od tego co chcemy sprawdzić w kodzie naszego programu, możemy posłużyć się różnymi metodami. Te laboratoria zapoznały mnie z kilkoma wcześniej nieznanymi mi narzędziami, które można wykorzystać w tym celu.