

Initiation à la programmation bas niveau sous Windows

Cet atelier a pour but d'exercer ses participants à la conception de programmes en assembleur sous Windows.

Objectifs pédagogiques

- Savoir utiliser le jeu d'instructions des processeurs modernes, tels que l'Intel IA-32.
- Prendre connaissance de quelques-uns des mécanismes internes de Windows.
- Comprendre les interactions entre DLL de sous-systèmes et le noyau de Windows (éventuellement par le biais des appels systèmes).

Thèmes abordés

- **Langage d'assemblage** Langage de bas niveau directement compréhensible par le processeur.
- **API Windows** Ensemble normalisé de fonctions permettant de se servir des fonctionnalités des différents systèmes d'exploitation de la gamme Windows.
- **Portable Executable** Format de fichier sous-jacent aux exécutables et bibliothèques sur les systèmes d'exploitation Windows 32 et 64 bits.
- **Architecture de Windows** Sous-systèmes d'environnement, DLL de sous-système (Kernel32.dll, Advapi.dll, etc.), bibliothèque générale de support système (Ntdll.dll), services natifs, services de l'exécutif, noyau (Ntoskrnl.exe), et bien d'autres.

En dehors des éléments techniques, cet atelier est fait pour que vous rencontriez concepts et terminologie. Identifiez les termes que vous ne comprenez pas et entreprenez des recherches à leur sujet.

Outils

- **Masm** Assembleur Microsoft épaulant divers projets de la firme, dont la suite de développement Visual Studio. Consultez le site <http://www.ollydbg.de> pour plus de détails et <http://www.ollydbg.de/odbg110.zip> pour le téléchargement.
- **OllyDbg** Outil d'analyse de binaires au niveau utilisateur. Consultez le site <http://masm32.com> pour plus de détails et <http://website.assemblercode.com/masm32/masm32v11r.zip> pour le téléchargement.

Travaux pratiques : injection de code

L'injection de code désigne une méthode par laquelle un programme se rend maître des instructions exécutées dans le contexte d'un autre programme. Une telle technique peut s'établir de deux façons, et être de nature statique ou dynamique. Une injection est qualifiée de statique lorsqu'elle a lieu avant l'exécution d'un programme (a fortiori avant la transformation de celui-ci en processus). A l'inverse, une injection de type dynamique se distingue par le fait de cibler un processus en cours d'exécution.

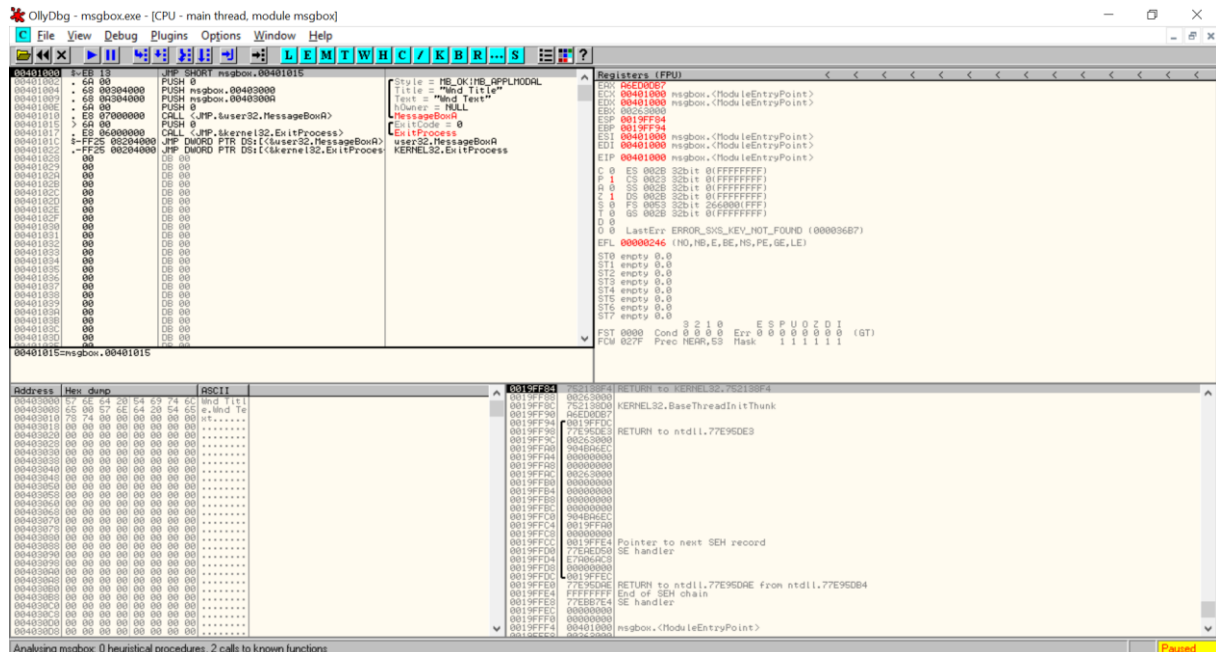
Nous allons dans ce qui suit explorer l'injection de code statique. A cet effet, nous manipulerons le programme msgbox de sorte que son exécution intègre une boîte de dialogue affichant un message.

Afin de manipuler le programme cible, nous utiliserons l'outil d'analyse automatique nommé OllyDbg, que vous pouvez obtenir depuis l'URL suivante : <http://www.ollydbg.de>.

Décompressez odbg110.zip dans un dossier de votre choix.

A partir du chemin d'installation adéquat, démarrez OllyDbg (OllyDbg.exe). Lors du premier lancement, OllyDbg présente éventuellement à l'utilisateur un message demandant la mise à jour de certaines bibliothèques (.dll). Cliquez sur Non.

Allez dans File, Open puis sélectionnez le fichier msgbox1.exe. Dans le menu Debug, cliquez sur Run. Vous devriez à ce moment voir le code machine exécutable dudit programme.



- **E1** Prenez note de la valeur du registre EIP, à quoi correspond dans le cas présent le point d'entrée de l'exécutable (0x00401000 dans notre exemple).
- **E2** Prenez note de la première instruction immédiatement après le point d'entrée du module. (0x00401002 dans notre exemple).

Du fait de l'alignement, l'espace d'adressage virtuel afférent à tout processus contient un certain nombre de zones mémoire n'hébergeant aucune information utile. Dans les grandes lignes, l'injection statique consiste justement à profiter de la présence de ces régions afin d'y insérer diverses instructions.

Au niveau de la fenêtre des instructions de OllyDbg (juste en dessous du menu), faites défiler vers le bas jusqu'à repérer une zone remplie de zéros.

- **E3** Prenez note de l'adresse mémoire du premier octet à constituer cette zone une zone remplie de zéros. (0x00401028 dans notre exemple).

Nous allons maintenant ajouter un appel à la fonction MessageBox de Windows, lequel se présente sous la forme suivante :

```
int WINAPI MessageBox(  
    _In_opt_ HWND    hWnd,  
    _In_opt_ LPCTSTR lpText,  
    _In_opt_ LPCTSTR lpCaption,  
    _In_      UINT    uType
```

Le pseudo code en assembleur correspondant à cet appel est :

```
push 0
```

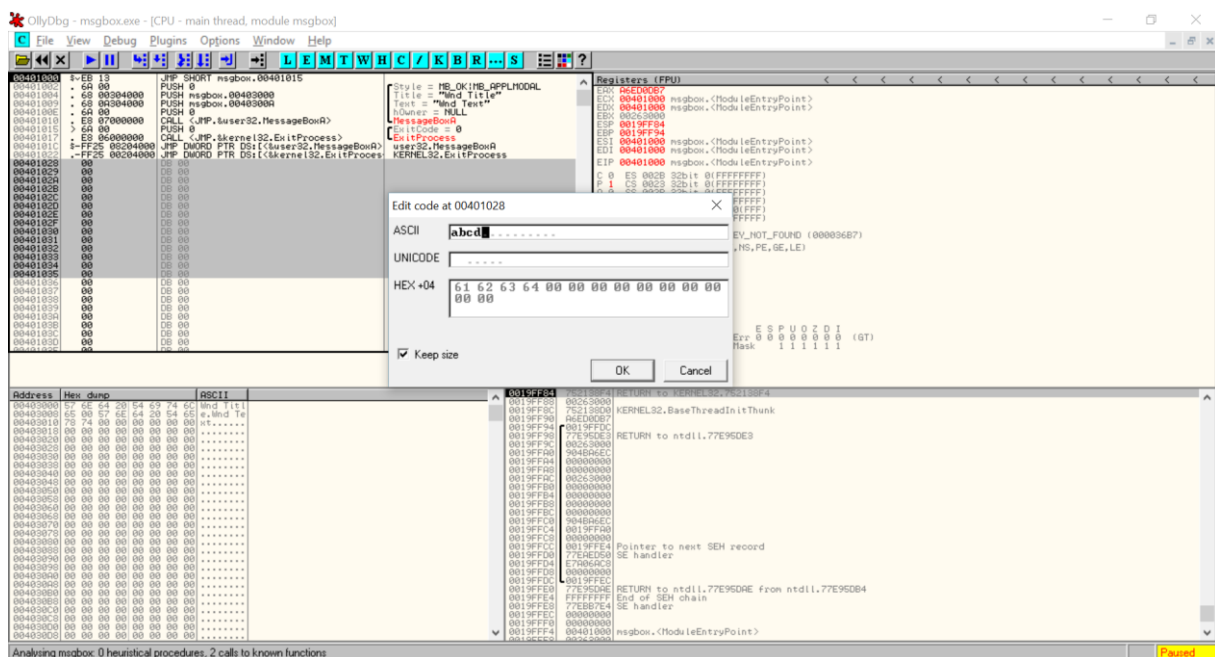
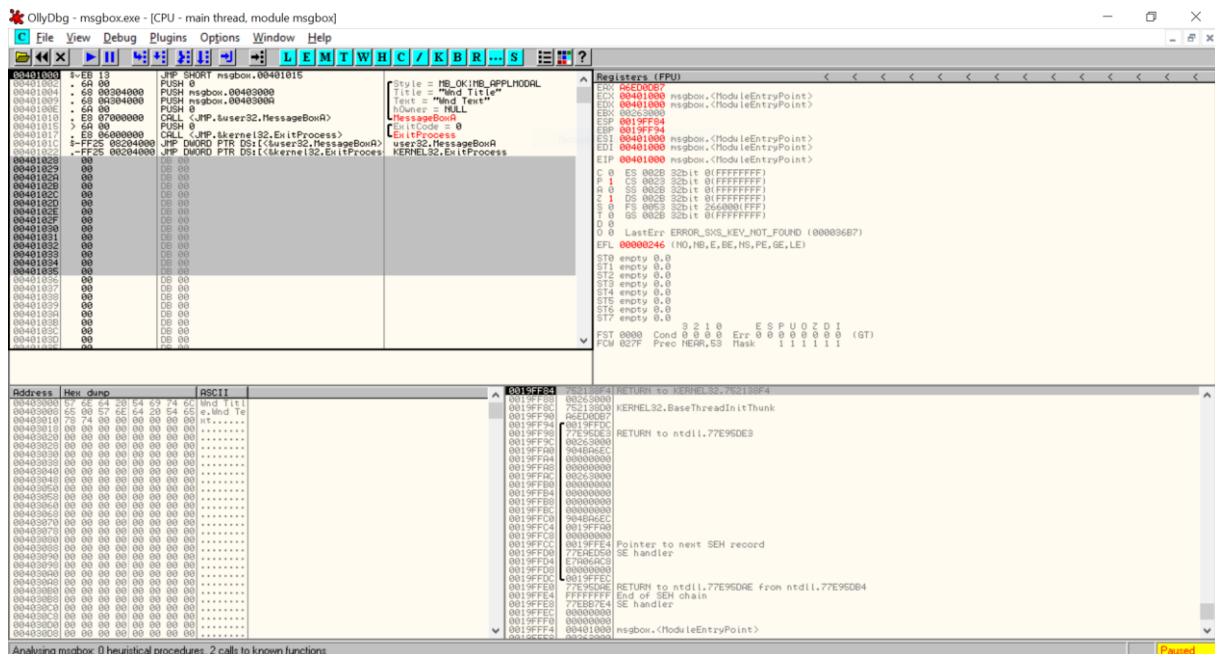
```
push "Titre de la fenetre"
```

```
push "Texte de la fenetre"
```

```
push 0
```

```
call User32.MessageBoxA
```

Nous allons maintenant ajouter le code via OllyDbg. Mettez en surbrillance un groupe d'octets parmi la zone. Faites un clic droit et sélectionnez Binary->Edit. Dans le champ ASCII de la fenêtre nouvellement ouverte, inscrivez une chaîne de caractères quelconque.



The screenshot displays the OllyDbg interface with the following components:

- Menu Bar:** File, View, Debug, Plugins, Options, Window, Help.
- Toolbar:** Standard debugging tools like Run, Step Over, Step Into, etc.
- Disassembly Window:**
 - Address range: 00401000 to 00401040.
 - Assembly code:


```

00401000 4E EB 13 JPP SHORT msgbox.00401015
00401002 68 00 PUSH 0
00401004 68 00 PUSH 0
00401006 68 00 PUSH 0
00401008 68 00 PUSH 0
0040100A 68 00 PUSH 0
0040100C 68 00 PUSH 0
0040100E 68 00 PUSH 0
00401010 68 00 PUSH 0
00401012 68 00 PUSH 0
00401014 68 00 PUSH 0
00401016 68 00 PUSH 0
00401018 68 00 PUSH 0
0040101A 68 00 PUSH 0
0040101C 68 00 PUSH 0
0040101E 68 00 PUSH 0
00401020 68 00 PUSH 0
00401022 68 00 PUSH 0
00401024 68 00 PUSH 0
00401026 68 00 PUSH 0
00401028 68 00 PUSH 0
0040102A 68 00 PUSH 0
0040102C 68 00 PUSH 0
0040102E 68 00 PUSH 0
00401030 68 00 PUSH 0
00401032 68 00 PUSH 0
00401034 68 00 PUSH 0
00401036 68 00 PUSH 0
00401038 68 00 PUSH 0
0040103A 68 00 PUSH 0
0040103C 68 00 PUSH 0
0040103E 68 00 PUSH 0
00401040 68 00 PUSH 0
00401042 68 00 PUSH 0
00401044 68 00 PUSH 0
00401046 68 00 PUSH 0
00401048 68 00 PUSH 0
0040104A 68 00 PUSH 0
0040104C 68 00 PUSH 0
0040104E 68 00 PUSH 0
00401050 68 00 PUSH 0
00401052 68 00 PUSH 0
00401054 68 00 PUSH 0
00401056 68 00 PUSH 0
00401058 68 00 PUSH 0
0040105A 68 00 PUSH 0
0040105C 68 00 PUSH 0
0040105E 68 00 PUSH 0
00401060 68 00 PUSH 0
00401062 68 00 PUSH 0
00401064 68 00 PUSH 0
00401066 68 00 PUSH 0
00401068 68 00 PUSH 0
0040106A 68 00 PUSH 0
0040106C 68 00 PUSH 0
0040106E 68 00 PUSH 0
00401070 68 00 PUSH 0
00401072 68 00 PUSH 0
00401074 68 00 PUSH 0
00401076 68 00 PUSH 0
00401078 68 00 PUSH 0
0040107A 68 00 PUSH 0
0040107C 68 00 PUSH 0
0040107E 68 00 PUSH 0
00401080 68 00 PUSH 0
00401082 68 00 PUSH 0
00401084 68 00 PUSH 0
00401086 68 00 PUSH 0
00401088 68 00 PUSH 0
0040108A 68 00 PUSH 0
0040108C 68 00 PUSH 0
0040108E 68 00 PUSH 0
00401090 68 00 PUSH 0
00401092 68 00 PUSH 0
00401094 68 00 PUSH 0
00401096 68 00 PUSH 0
00401098 68 00 PUSH 0
0040109A 68 00 PUSH 0
0040109C 68 00 PUSH 0
0040109E 68 00 PUSH 0
004010A0 68 00 PUSH 0
004010A2 68 00 PUSH 0
004010A4 68 00 PUSH 0
004010A6 68 00 PUSH 0
004010A8 68 00 PUSH 0
004010AA 68 00 PUSH 0
004010AC 68 00 PUSH 0
004010AE 68 00 PUSH 0
004010B0 68 00 PUSH 0
004010B2 68 00 PUSH 0
004010B4 68 00 PUSH 0
004010B6 68 00 PUSH 0
004010B8 68 00 PUSH 0
004010BA 68 00 PUSH 0
004010BC 68 00 PUSH 0
004010BE 68 00 PUSH 0
004010C0 68 00 PUSH 0
004010C2 68 00 PUSH 0
004010C4 68 00 PUSH 0
004010C6 68 00 PUSH 0
004010C8 68 00 PUSH 0
004010CA 68 00 PUSH 0
004010CC 68 00 PUSH 0
004010CE 68 00 PUSH 0
004010D0 68 00 PUSH 0
004010D2 68 00 PUSH 0
004010D4 68 00 PUSH 0
004010D6 68 00 PUSH 0
004010D8 68 00 PUSH 0
004010DA 68 00 PUSH 0
004010DC 68 00 PUSH 0
004010DE 68 00 PUSH 0
004010E0 68 00 PUSH 0
004010E2 68 00 PUSH 0
004010E4 68 00 PUSH 0
004010E6 68 00 PUSH 0
004010E8 68 00 PUSH 0
004010EA 68 00 PUSH 0
004010EC 68 00 PUSH 0
004010EE 68 00 PUSH 0
004010F0 68 00 PUSH 0
004010F2 68 00 PUSH 0
004010F4 68 00 PUSH 0
004010F6 68 00 PUSH 0
004010F8 68 00 PUSH 0
004010FA 68 00 PUSH 0
004010FC 68 00 PUSH 0
004010FE 68 00 PUSH 0
00401100 68 00 PUSH 0
00401102 68 00 PUSH 0
00401104 68 00 PUSH 0
00401106 68 00 PUSH 0
00401108 68 00 PUSH 0
0040110A 68 00 PUSH 0
0040110C 68 00 PUSH 0
0040110E 68 00 PUSH 0
00401110 68 00 PUSH 0
00401112 68 00 PUSH 0
00401114 68 00 PUSH 0
00401116 68 00 PUSH 0
00401118 68 00 PUSH 0
0040111A 68 00 PUSH 0
0040111C 68 00 PUSH 0
0040111E 68 00 PUSH 0
00401120 68 00 PUSH 0
00401122 68 00 PUSH 0
00401124 68 00 PUSH 0
00401126 68 00 PUSH 0
00401128 68 00 PUSH 0
0040112A 68 00 PUSH 0
0040112C 68 00 PUSH 0
0040112E 68 00 PUSH 0
00401130 68 00 PUSH 0
00401132 68 00 PUSH 0
00401134 68 00 PUSH 0
00401136 68 00 PUSH 0
00401138 68 00 PUSH 0
0040113A 68 00 PUSH 0
0040113C 68 00 PUSH 0
0040113E 68 00 PUSH 0
00401140 68 00 PUSH 0
00401142 68 00 PUSH 0
00401144 68 00 PUSH 0
00401146 68 00 PUSH 0
00401148 68 00 PUSH 0
0040114A 68 00 PUSH 0
0040114C 68 00 PUSH 0
0040114E 68 00 PUSH 0
00401150 68 00 PUSH 0
00401152 68 00 PUSH 0
00401154 68 00 PUSH 0
00401156 68 00 PUSH 0
00401158 68 00 PUSH 0
0040115A 68 00 PUSH 0
0040115C 68 00 PUSH 0
0040115E 68 00 PUSH 0
00401160 68 00 PUSH 0
00401162 68 00 PUSH 0
00401164 68 00 PUSH 0
00401166 68 00 PUSH 0
00401168 68 00 PUSH 0
0040116A 68 00 PUSH 0
0040116C 68 00 PUSH 0
0040116E 68 00 PUSH 0
00401170 68 00 PUSH 0
00401172 68 00 PUSH 0
00401174 68 00 PUSH 0
00401176 68 00 PUSH 0
00401178 68 00 PUSH 0
0040117A 68 00 PUSH 0
0040117C 68 00 PUSH 0
0040117E 68 00 PUSH 0
00401180 68 00 PUSH 0
00401182 68 00 PUSH 0
00401184 68 00 PUSH 0
00401186 68 00 PUSH 0
00401188 68 00 PUSH 0
0040118A 68 00 PUSH 0
0040118C 68 00 PUSH 0
0040118E 68 00 PUSH 0
00401190 68 00 PUSH 0
00401192 68 00 PUSH 0
00401194 68 00 PUSH 0
00401196 68 00 PUSH 0
00401198 68 00 PUSH 0
0040119A 68 00 PUSH 0
0040119C 68 00 PUSH 0
0040119E 68 00 PUSH 0
004011A0 68 00 PUSH 0
004011A2 68 00 PUSH 0
004011A4 68 00 PUSH 0
004011A6 68 00 PUSH 0
004011A8 68 00 PUSH 0
004011AA 68 00 PUSH 0
004011AC 68 00 PUSH 0
004011AE 68 00 PUSH 0
004011B0 68 00 PUSH 0
004011B2 68 00 PUSH 0
004011B4 68 00 PUSH 0
004011B6 68 00 PUSH 
```

- **E4** Prenez note de l'adresse mémoire du premier octet à constituer cette chaîne de caractères. (0x00401028 dans notre exemple).

push 0

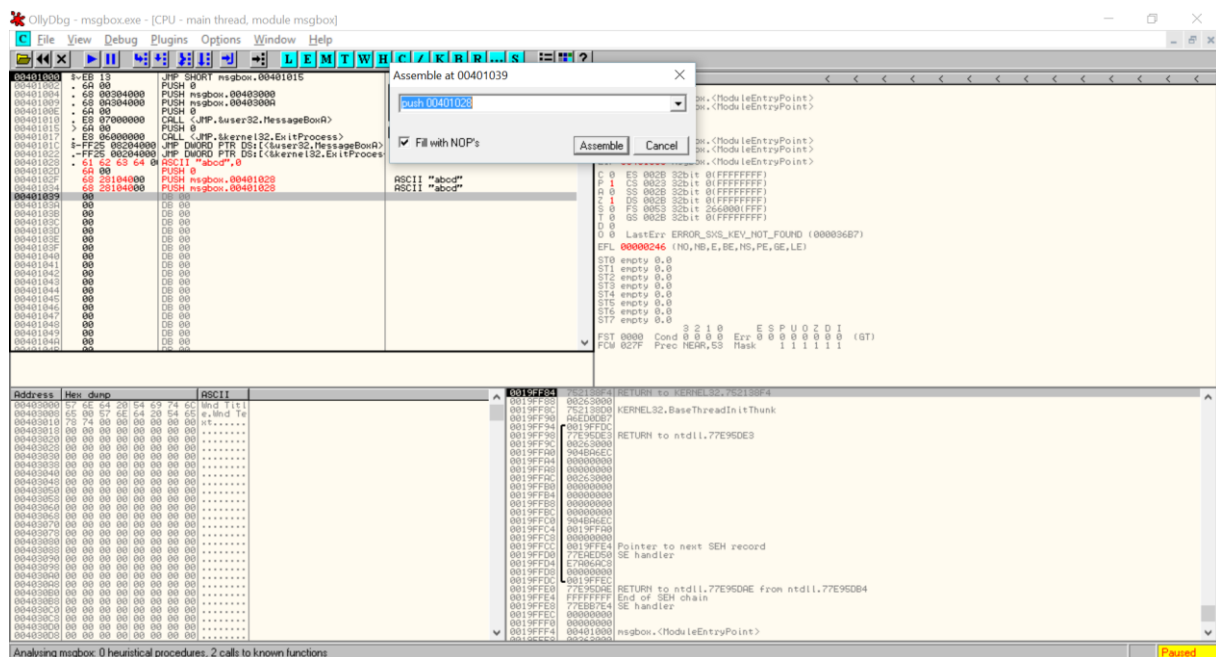
[illegible]

- **E5** Prenez note de l'adresse mémoire du premier octet à constituer ces instructions. (0x0040102d dans notre exemple).

Inscrivez ensuite :

push adresse

où *adresse* est l'adresse mémoire de la chaîne de caractères précédemment constituée (voir E4). Cliquez deux fois sur le bouton Assemble, de sorte à former dans un même mouvement les paramètres lpText et lpCaption pour MessageBox.



Inscrivez ensuite le quatrième et dernier argument de la fonction.

push 0

Ecrivez ensuite ce qui suit puis sur le bouton Assemble :

call user32.MessageBoxA

The screenshot shows the OllyDbg interface with the following details:

- File View Debug Plugins Options Window Help** menu bar.
- Assembly Window:** Displays assembly instructions starting from address 00401002. Instructions include:
 - `00401002 EB 2B JMP SHORT msghow.00401020`
 - `00401003 6A 00 PUSH 0`
 - `00401004 68 00304000 PUSH msghow.00403000`
 - `00401005 68 00304000 PUSH msghow.00403000`
 - `00401006 6A 00 PUSH 0`
 - `00401007 E9 87000000 JUMP JmpUser32.SendMessage`
 - `00401008 6A 00 PUSH 0`
 - `00401009 E9 05000000 JUMP JmpUser32.ExitProcess`
 - `00401010 FF25 00204000 JUMP DWORD PTR DS:[user32.SendMessage]`
 - `00401011 FF25 00204000 JUMP DWORD PTR DS:[kernel32.ExitProcess]`
 - `00401012 6A 00 PUSH 0`
 - `00401013 68 29104000 PUSH msghow.00401029`
 - `00401014 68 29104000 PUSH msghow.00401029`
 - `00401015 68 12000000 PUSH 0`
 - `00401016 EB C2 JMP SHORT msghow.00401082`
- Registers (FPU) Window:** Shows the EIP register at address 00401002.
- Dialog Box:** A small window titled "Assemble at 00401002" is open, showing the instruction `JMP 00401002` and a checkbox for "Fill with NOPs".
- Memory Dump Window:** Shows a hex dump of memory starting at address 00401000, with ASCII values displayed on the right.

Revenez à l'origine du programme et faites Run. Deux boîtes de dialogue devraient se succéder, la première résultant des instructions ajoutées, la seconde de l'exécution « normale » du processus.

Exercices

Le but des exercices qui suivent est d'écrire en assembleur un programme mettant en œuvre les différentes procédures vues à la section Travaux pratiques.

- **ex1** Injecte une boîte de dialogue dans le programme msgbox1.exe, lequel contient par ailleurs un appel à user32!MessageBoxA et une instruction de saut, les deux étant là afin de vous faciliter la tâche.
- **ex2** Injecte une boîte de dialogue dans le programme msgbox2.exe, qui contient une instruction de saut mais pas de références à la fonction Windows MessageBox.
- **ex3** Injecte une boîte de dialogue dans le programme msgbox3.exe, qui ne contient ni instruction de saut ni références à la fonction Windows MessageBox. Conseil : regardez à quoi correspond le champ de l'entête PE nommé AddressOfEntryPoint.
- **ex4** Injecte une boîte de dialogue dans un programme Windows quelconque, et cela indépendamment de la version du système d'exploitation utilisée.

Consignes de rendu

Les réponses aux différents exercices présentés en amont doivent être envoyées par mail à l'adresse maillard.arnaud@gmail.com. Pas de consignes précises à ce niveau. Si vous utilisez des directives d'assemblage particulières, précisez-le.

Consignes diverses

- Toutes les facilités d'utilisation introduites par MASM (par exemple .IF, .WHILE, etc.) sont interdites.
- En ce qui concerne les opérations sur les fichiers, vous devez utiliser le mécanisme des fichiers représentés en mémoire (MapViewOfFile et consorts).
- Tout appel de fonction externe hors du sous-système Windows est interdit. Utilisez par exemple HeapAlloc plutôt que malloc.