Assembleur

TP 2

# Objectif :

L’objectif de ce TP est d’analyser une première fonction en assembleur x86 32bits et de trouver son équivalent en C. Puis de découvrir une méthode pour apprendre progressivement à lire et comprendre l’assembleur.

# Prérequis :

TP1

# Étape 1 :

|  |
| --- |
| **A quoi servent ces deux instructions en début de fonction ?**  **push ebp**  **mov ebp, esp** |
|  |

|  |
| --- |
| **Ecrivez les instructions inverses que permettent de terminer une fonction ?** |
|  |
| **Comment appelle-t-on ces instruction qui terminent la fonction ? A quoi servent-elles ?** |
|  |

|  |
| --- |
| **Si ebp correspond à une frame et si var\_A est égal à 8, à quoi correspond [ebp + var\_A] dans l’instruction cmp[ebp+var\_A], 0 ?** |
|  |

|  |
| --- |
| **Analysez ces 3 instructions et écrivez l’équivalent avec eax = 4 en C ?**   1. **push eax** 2. **call sub\_401000** 3. **add esp, 4** |
|  |

# Étape 2 :

|  |
| --- |
| **Analysez et expliquez cette fonction ?**  **; =============== S U B R O U T I N E =======================================**  **; Attributes: bp-based frame**  **sub\_401000 proc near ; CODE XREF: sub\_401000+21↓p**  **; sub\_401050+45↓p**  **arg\_0 = dword ptr 8**  **push ebp**  **mov ebp, esp**  **cmp [ebp+arg\_0], 0**  **jnz short loc\_40100D**  **mov eax, 1**  **jmp short loc\_40102D**  **; ---------------------------------------------------------------------------**  **loc\_40100D: ; CODE XREF: sub\_401000+7↑j**  **cmp [ebp+arg\_0], 1**  **jnz short loc\_40101A**  **mov eax, 1**  **jmp short loc\_40102D**  **; ---------------------------------------------------------------------------**  **loc\_40101A: ; CODE XREF: sub\_401000+11↑j**  **mov eax, [ebp+arg\_0]**  **sub eax, 1**  **push eax**  **call sub\_401000**  **add esp, 4**  **imul eax, [ebp+arg\_0]**  **loc\_40102D: ; CODE XREF: sub\_401000+B↑j**  **; sub\_401000+18↑j**  **pop ebp**  **retn**  **sub\_401000 endp** |
|  |

# Étape 3 :

|  |
| --- |
| **Ecrivez une fonction de votre choix en C, puis compilez la dans un programme.**  **Vérifiez le bon fonctionnement en exécutant votre programme.** |
|  |
| **Ecrivez en assembleur x86 cette fonction :** |
|  |

|  |
| --- |
| **Recherchez l’option de GCC pour convertir votre .C en .ASM ; puis écrivez la ligne de commande.** |
|  |

|  |
| --- |
| **Etudiez et comparez le fichier .ASM avec votre code.** |
|  |

|  |
| --- |
| **Amusez-vous à refaire le même exercice avec des fonctions en C de plus en plus complexe, puis analysez le code assembleur produit. Cette méthode est super intéressante pour progresser à son rythme.** |
|  |