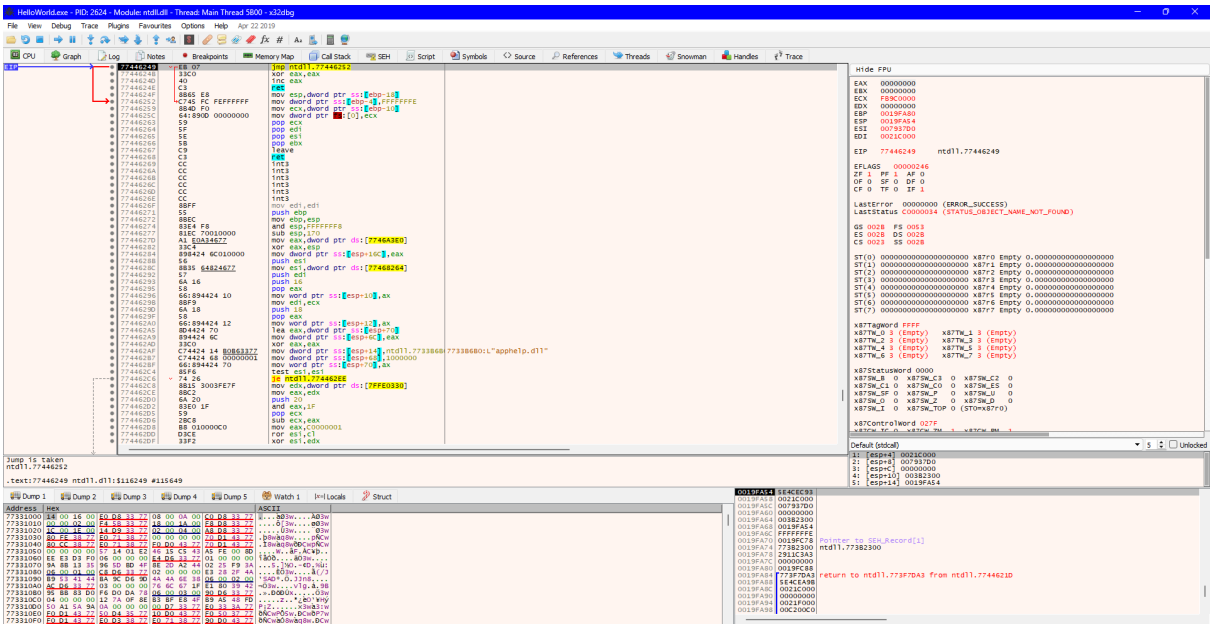
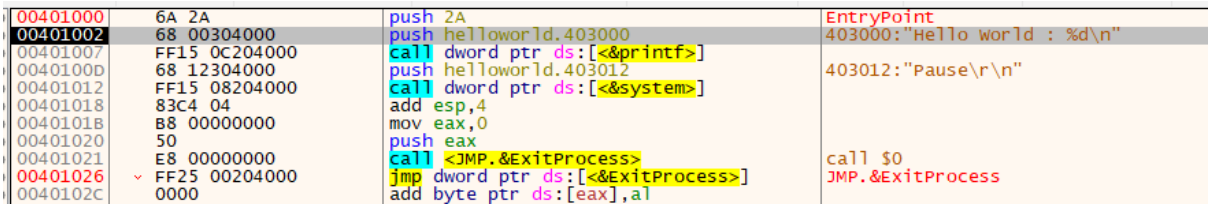
A. Appel de fonctions : a-b

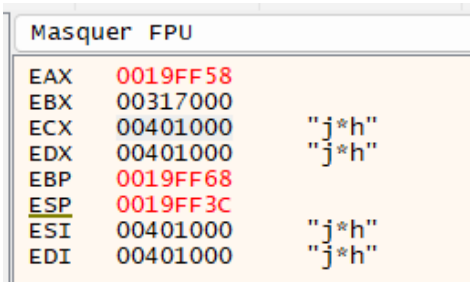
Après avoir compilé le programme HelloWorld, je l'ai exécuté à l'aide d'un débogueur (ici x32dbg) pour observer comment s'effectue l'appel à la fonction printf.



Dans un premier temps, on peut voir que le programme effectue plusieurs instructions push, notamment pour placer la valeur 2A (en hexadécimal, soit 42 en décimal) ainsi qu'une adresse pointant vers une chaîne de caractères dans la pile. Cela correspond aux arguments que la fonction printf va utiliser. Ces arguments sont empilés dans l'ordre inverse de leur utilisation dans la fonction.



Ensuite, la fonction printf est appelée via l'instruction CALL. Cette instruction pousse l'adresse de retour sur la pile, puis effectue un saut vers le code de la fonction printf. Une fois dans printf, celle-ci doit récupérer les arguments précédemment empilés. Pour cela, elle s'appuie sur le contenu de la pile à partir de la base de pile (EBP), qui ne change pas durant l'exécution de la fonction.

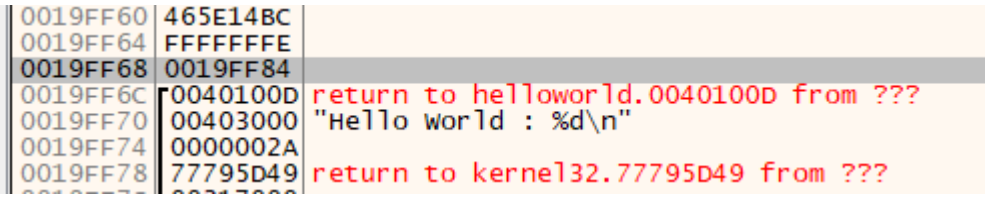


Dans notre cas, à un instant donné de l'exécution dans printf, le registre EBP contient la valeur 0019FF68.

En observant la pile à cette adresse, on retrouve bien la structure attendue :

EBP + 8 → adresse de la chaîne "Hello World : %d\n"

EBP + C (soit +12 en décimal) → valeur entière 42



Cela montre que les arguments sont bien empilés dans l'ordre attendu et accessibles via des décalages fixes par rapport à EBP.