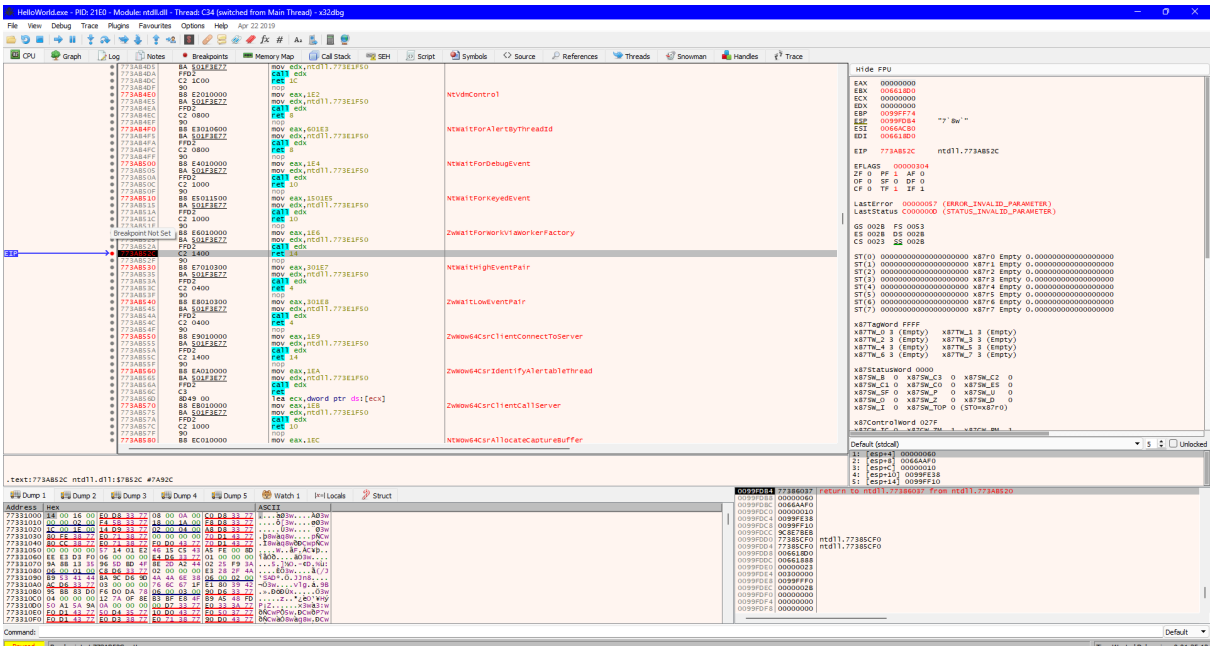
A. Appel de fonctions : a-c

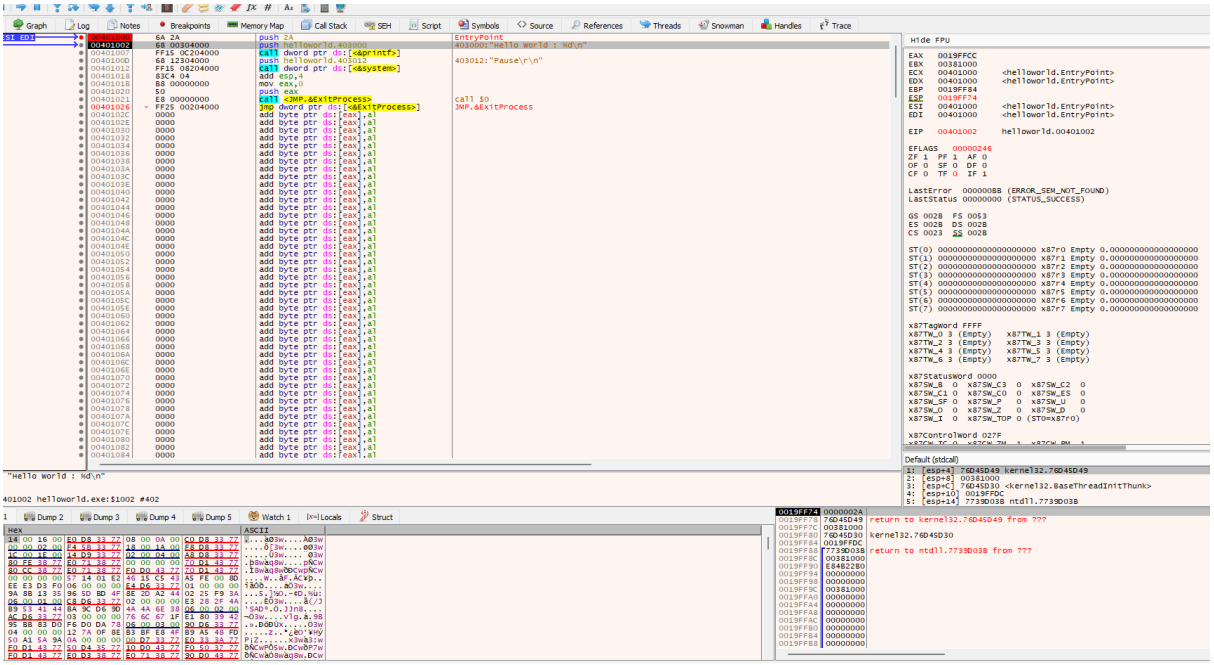
En lançant le programme dans le débogueur, j'ai essayé de suivre ce qui se passait dans la pile pendant l'exécution, notamment autour de l'appel à printf.



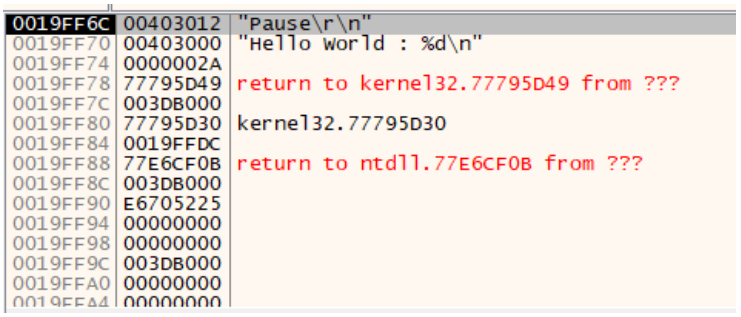
Avant d'appeler la fonction printf, le programme fait plusieurs push pour placer les arguments dans la pile. Par exemple, ici, il pousse l'adresse du texte "Hello World : %d\n" et aussi la valeur 2A (qui est 42 en décimal).

Ensuite, quand on entre dans la fonction printf, il y a une adresse de retour qui est automatiquement mise dans la pile. C'est pour que le programme sache où revenir une fois que printf a fini. Par exemple, on voit l'adresse 0040100D, c'est probablement l'endroit où le programme continue après.

Il y a aussi d'autres adresses qui apparaissent, comme 77795D49, qui pointent vers des fonctions système. printf appelle d'autres fonctions internes, et il garde lui aussi des adresses pour pouvoir revenir au bon endroit ensuite.



L’argument "Hello World : %d\n" est bien visible dans la pile, un peu plus bas.
On voit que printf va chercher ses paramètres à partir de la pile, en se basant sur le registre EBP. Par exemple, "Hello World : %d\n" est à EBP + 8 et 42 à EBP + 12.



À la fin, j’ai vu qu’il y avait aussi une chaîne "Pause\r\n" dans la pile. Ce que je ne comprends pas encore totalement, c’est pourquoi elle se retrouve là alors qu’on n’a pas encore fait appel à Pause (ou, du moins, je ne l’ai pas vu clairement). Peut-être qu’elle est poussée dans la pile pour une utilisation future, ou pour arrêter l’exécution ensuite.