CR TP3

Exercice 1

1. La ligne 18 déplace le contenu du registre **eax** à l'adresse mémoire réservée par l'**input1**.
2. La ligne ajoute l'octet stocké à l'adresse **input2** au registre **eax**
3. Appel vers la fonction pour sortir le résultat de la somme (présente dans le registre **eax**).
4. Nettoie les registres et fini le programme (31 -> vide le registre **eax**, 32 > prépare l'arrêt et la fin de l'exécution du programme, 33 > exécute l'interruption)

Exercice 2

1.

mov eax, 0xFFFFFFFF | *rempli le registre eax* cmp eax, 0 | *compare 0 à eax* jg aff_1 | *si la valeur du registre eax est plus grande que 0, on va à aff_1, en l'occurence, la comparaison est fausse car la valeur de registre eax est négative* mov eax, 0 | *déplace la valeur 0 dans le registre **eax*** call print_int | *affiche la valeur 0 en sortie* aff_1 : mov eax, 1 | *ignorée car dans une étiquette jamais appelée, aurait déplacée la valeur 1 dans le registre **eax***
call print_int | *affiche la valeur de **eax**, en l'occurence 1*

2.

mov eax, 0xFFFFFFFF | *rempli le registre eax* cmp eax, 0 | *compare 0 à eax* ja aff_1 | *si la valeur du registre eax est plus grande que 0, on va à aff_1* mov eax, 0 | *ignorée car la comparaison est vraie* call print_int | *aussi ignorée, aurait affiché 0* aff_1 : mov eax, 1 | *déplace la valeur 1 dans le registre eax*
call print_int | *affiche la valeur de **eax**, en l'occurence 1*

Exercice 3

Voir fichier E3.asm

Exercice 4

mov edx, 0x00000000 ; edx = 0x00000000 mov eax, 0x000005DE ; eax = 0x000005DE mov ebx, 15 ; ebx = 0x0000000F div ebx

Exercice 5

Voir fichier E5.asm

Exercice 6

Pas de grand blocaque sur un point précis mais plusieurs erreurs sur la gestions des registre entre eux. finalement nous avons choisi d'utiliser presque aucune variable suplémentaire.

Exercice 7

On a mis du temps à comprendre ce qu'il fallait mettre dans chaque registre pour faire fonctionner l'opération shl. On ne savait pas non plus quelle était la meilleure méthode pour afficher les '1' et les '0', on a donc opter pour l'utilisation de la fonction 'print_string' comme dans les exercices précedents.

Exercice 10

Comme le décrit l'astuce 2, on a créé un tableau de 51 octets pour tous les nombres allant de 0 à 50, et le -1. Le script comporte deux étapes, la saisie des entiers, et leur marquage dans le mémoire. Si en entier est saisi, on l'indique en plaçant '1' à son emplacement en mémoire (on stoppe l'analyse de la saisie quand -1 est lu). La deuxième étape est la restitution des entiers non-notés. Pour cela, on utilise deux registres principalement : ecx, qui va nous servir de compteur pour parcourir les entiers de 0 à 50, et al, qui va nous permettre de faire la comparaison entre '1' et la valeur correspondant au nombre dans le tableau. Si la valeur est 1, on passe à la valeur suivante, sinon, on affiche le nombre dans la sortie. L'utilisation d'al et non pas d'eax permet d'économiser des accès inutiles en mémoire, car le tableau est un tableau d'octets, (al = 1 octets et eax = 4 octets).