Questions TP2

1. **Comment vous êtes-vous réparti le travail ?**

- Sacha Conti — Mise en place du gestionnaire d'interruptions
- Thierry Mourao — Gestion du clavier
- Sylvain Thullen — Gestion du timer
- Groupe — Modification kernel et tests

2. Votre kernel comporte-t-il des bugs ? Si oui, lesquels et comment pourriez-vous les corriger ?

- La preuve visuelle du fonctionnement du timer (demandée dans l'énoncé) provoque des erreurs d'affichage, notamment quand on doit faire un scroll_up() car les caractères ne sont pas effacés. La solution serait d'enlever cet affichage du tick.
- Le kernel en soi ne présente aucun dysfonctionnement ni bug autrement.

3. Dans quel ordre vous avez initialisé les différents points ci-dessus dans votre kernel ? Justifiez.

a. *VGA* — On init d'abord le mode video avant de passer en mode protégé parce que l'on va devoir afficher les informations.

b. *GDT* — On a besoin d'allouer la mémoire pour l'utiliser après.

c. *PIC* — On remappe les interruptions avant d'initialiser l'IDT.

d. *IDT* — Initialisation de l'IDT avant le timer et le clavier car les deux appellent leur IRQ respective.

e. *TIMER* — Sans importance.

f. *KEYBOARD* — Sans importance.

4. Pourquoi remappe-t-on les IRQ 0 à 7 aux interruptions 32 à 39 ?

- Car les 32 premières sont les interruptions du processeur Intel

5. Que se passerait-il si on ne le faisait pas ?

- On rentrerait en conflit entre nos interruptions et celle du processeur, ainsi on ne saurait si le processeur a un problème.

6. Comment pouvez-vous tester que votre gestionnaire d'interruption pour les exceptions fonctionne correctement ?

- On peut volontairement faire une division par 0 et générer une exception.
- Ou faire une *segmentation fault* avec un pointeur

7. Quelles exceptions avez-vous pu générer et comment avez-vous fait ?

- Une exception de division par 0 en en faisant une volontairement dans les tests.
- Une exception 13 (General Protection) en spécifiant mal la taille du pointeur sur l'idt

8. Quelle taille de buffer clavier avez-vous choisie et pourquoi ?

- 80 car cela représente une ligne et qu'avec une horloge à 100Hz (10ms entre chaque tick) nous n'arrivons pas à le remplir entre deux appels de getc().

9. Comment pouvez-vous causer une situation de buffer plein quelle que soit la taille du buffer (dans les limites du raisonable) ?

- Dans la boucle infinie de kernel.c (while(1)) on peut ajouter un sleep() avec une valeur assez grande pour que getc() soit appelé moins souvent, de cette manière on a le temps de remplir le buffer et de s'apercevoir qu'une fois plein on affiche un message et on ne peut plus écrire dedans.