Dans quel ordre vous avez initialisé les différentes points ci-dessus dans votre kernel?

- GDT
- Display
- IDT
- PIC
- Timer (100Hz)

Après tout cela, nous appelons `sti()` pour débloquer les interruptions. Le plus important est d'appeler `sti()` après toutes les autres initialisations, sinon des interruptions vont se déclencher mais nous ne serions pas prêts à les gérer.

Nous n'avons pas d'initialisation du clavier, car il n'y a rien à initialiser dans notre implémentation.

Pourquoi remappe-t-on les IRQ 0 à 7 aux interruptions 32 à 39 ? Que se passerait-il si on ne le faisait pas ?

Parce que si on les laisse de 0 à 7, elles seraient en conflit avec les 8 premières interruptions processeur. Du coup il faut les déplacer à 32, car les interruptions processeurs réservent jusqu'à l'entrée 31.

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

On peut essayer de lancer une division par zéro par exemple, cela va lancer une exception 0 et afficher le message de "Divide Error" et arrêter le système.

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

Nous avons un buffer de taille 10. Il n'y a pas de raison particulière, en soi, on pourrait avoir un buffer de 256 ou plus même, mais étant donné que le buffer se remplit qu'en cas de non-lecture, ce qui dans notre cas signifie être dans une phase de `sleep`, il n'était pas forcément nécessaire d'avoir un buffer plus grand.

Comment pouvez-vous causer une situation de buffer plein, même pour un buffer de grande capacité ?

Il y a deux façons de le faire. On peut ne pas appeler `getc()` et écrire dans le buffer, ce qui va remplir le buffer sans jamais le vider. Ou on peut également mettre un `sleep` d'une certaine durée dans notre code avant l'appel à `getc()` et remplir le buffer pendant le sleep, le buffer sera plein, et ensuite vidé d'un char lors de chaque appel à `getc()`.

Est-ce que votre fonction sleep possède une limitation ? Si oui, laquelle et pourquoi ?

La limitation dépend de la fréquence qu'on utilise. C'est-à-dire que la granularité de notre sleep dépend de la fréquence. Par exemple, si nous avons une fréquenc de 100Hz, on ne pourra pas faire de `sleep` avec une précision de moins de 10 ms