Miniprojet système : développement d'un noyau de système d'exploitation

Responsable : Christophe Rippert
Christophe.Rippert@Grenoble-INP.fr

Version optimisée de l'effaçage de l'écran

Le but de cette partie facultative est de réimplanter la fonction `void efface_ecran(void)` en assembleur sans traduire directement les boucles C, en profitant d'instructions spécifiques à l'architecture Intel.
Il existe plusieurs classes d'instructions natives permettant de manipuler efficacement des données entre le processeur et la mémoire. Dans notre cas, on va utiliser les instructions de la classe `stos` (*store string*) qui sont capables de charger des mots dans la mémoire de façon optimisée.
Le principe de cette classe d'instruction est le suivant :

- on place la donnée que l'on veut charger dans la fraction de `%eax` correspondant à la taille de la donnée (`%al` pour un octet, `%ax` pour un mot de 16 bits et `%eax` pour un mot de 32 bits)
- on place l'adresse de la zone mémoire destination dans le registre `%edi` ;
- on affecte le bit `D` du registre `%eflags` pour préciser le sens de la copie (comme expliqué plus bas) ;
- on exécute l'instruction `stos` en précisant un suffixe de taille correspondant à la taille des données qu'on veut copier : `stosb` pour un octet, `stosw` pour un mot de 16 bits ou `stosl` pour un mot de 32 bits ;
- en fonction du bit `D`, le registre `%edi` est automatiquement incrémenté ou décrémenté du nombre d'octets correspondant à la taille des données qu'on a copiées.

Pour affecter le bit `D`, on utilise les instructions suivantes :

- `cld` met le bit `D` à zéro, ce qui signifie qu'on veut incrémenter `%edi` ;
- `std` met le bit `D` à un, ce qui signifie qu'on veut décrémenter `%edi`.

On pourrait donc décrire le fonctionnement de l'instruction `stosl` (donc manipulant des données de 32 bits) avec le pseudo-code suivant :

```
MEM[edi] = eax
if (D == 0) {
    edi = edi + 4
} else {
    edi = edi - 4
}
```

Bien sûr, il faudra répéter l'exécution de l'instruction `stos` pour initialiser toute la zone mémoire, et pas seulement un mot. On pourrait utiliser une boucle, mais l'architecture x86 fourni une pré-instruction particulière (que l'on appelle un préfixe) qui permet de répéter l'exécution de certaines instructions (dont la classe `stos`) de façon très efficace : l'instruction `rep`.
Le préfixe `rep` répète l'exécution de l'instruction `stos` tant que le registre `%ecx` n'est pas nul, sachant que ce registre est automatiquement décrémenté de 1 à chaque exécution du `stos`. Il suffit d'écrire le préfixe `rep` avant l'instruction `stos`, sur la même ligne, pour que le `stos` soit répété.