Projet : développement d'un mini système d'exploitation pour PC x86

Introduction aux interruptions

Jérôme Ermont et Emmanuel Chaput

IRIT - Toulouse INP/ENSEEIHT

Les interruptions du X86

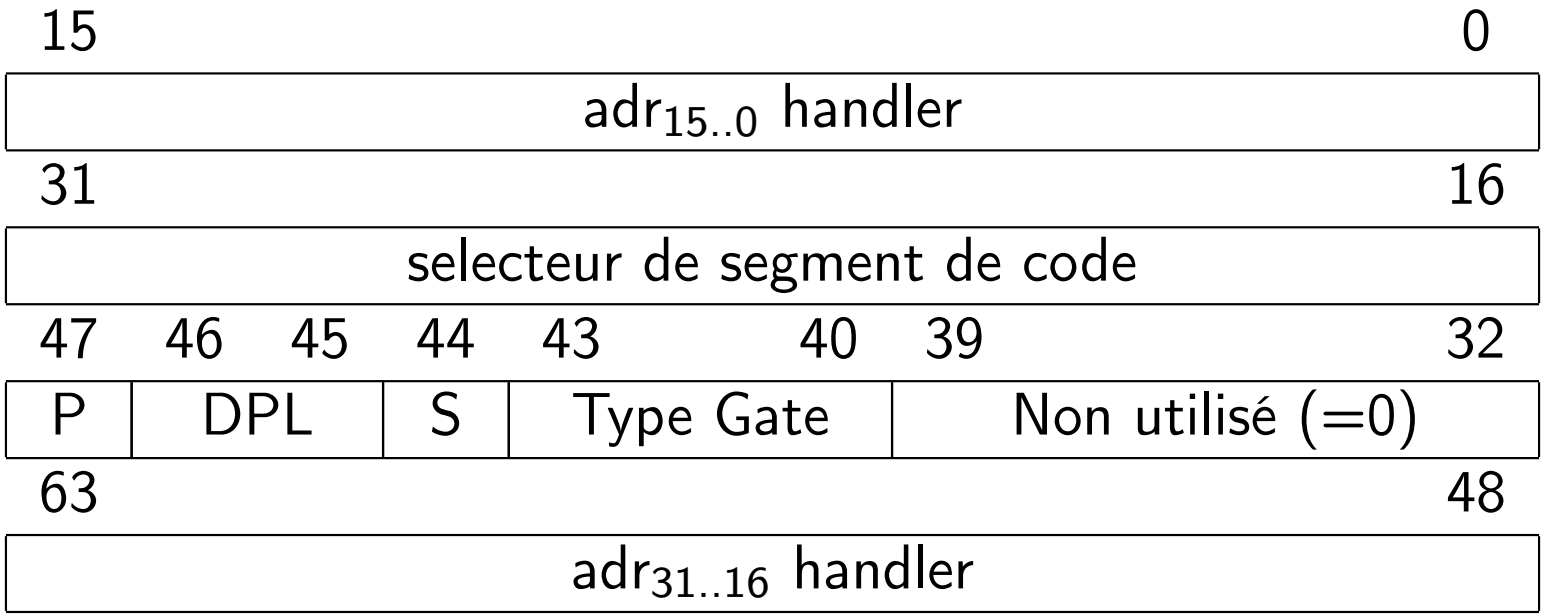
- 3 types d'interruptions :
  - Les exceptions
    - erreurs liées à l'exécution
    - division par zéro, dépassement, faute de page, ...
  - Les interruptions matérielles
    - générées par les périphériques
    - Timer, Clavier, ...
  - Les interruptions logicielles
    - émises via l'instruction `int`
- 256 interruptions possibles
- Traitement défini par une table de descipteurs d'interruptions (IDT)
- Activer les interruptions : instruction `sti` (`sti()`)
- Désactiver les interruptions : instruction `cli` (`cli()`)

Table des descripteurs d'interruption (IDT)

- Permet l'initialisation du vecteur d'interruption
- Une entrée dans la table
  - Valeur sur 64 bits
  - 3 types :
    - Interrupt gate (pour les IT)
    - Task gate (pour gérer les tâches)
    - Trap gate (pour gérer les exceptions)
- 256 entrées dans la table
- Initialisée via l'instruction assembleur `lidt`

- La table est déjà initialisée via la fonction `cpu_init`
- Pour cette partie, nous travaillerons sur les ITG

Format d'une entrée de la table



- P : entrée est configurée
- DPL : niveau de priviliège nécessaire (3 : faible, 0 : élevé)
- S : 0 = Trap gate ou Interrupt gate
- Type :
  - 5 : task gate 32 bits
  - 6 : interrupt gate 16 bits
  - 7 : trap gate 16 bits
  - 14 : interrupt gate 32 bits
  - 15 : trap gate 32 bits

Format du handler de l'interruption

```
.text
# la fonction est accessible
.globl handler_IT
# debut du traitant
handler_IT:
# sauvegarde de registres
pushl %eax
pushl %edx
pushl %ecx
# appel a la fonction C realisant le traitant
call handler_en_C
# restauration des registres sauvegardes
popl %ecx
popl %edx
popl %eax
# retour d'interruption
iret
```

Configuration d'une entrée

```
typedef struct {
   uint16_t adr_handler_inf;
   uint16_t sel_segment;
   uint8_t zero;
   uint8_t type_attr;
   uint16_t adr_handler_sup;
} idt_entry_t;
```

- `idt[i]` : *i*ème entrée dans la table
- selecteur de segment : `KERNEL_CS`

Travail à faire

1. Écrire la fonction C qui initialise la ligne `num_line` avec le traitant `handler` :

```
void init_idt_entry(int num_line,
                    (uint32_t) handler);
```

2. Tester la bonne réception d'une interruption (50 par exemple)
   1. Créer un handler assembleur pour cette interruption
   2. Écrire le code C associé qui affiche un message indiquant que l'interruption a été reçue
   3. Initialiser l'interruption dans le fichier `start.c`
   4. Activer les IT, `sti()`, dans `start.c`
   5. Envoyer l'IT depuis `start.c`
      Pour exécuter `int` dans du code C on peut utiliser :

```
__asm__ (int $num::);
```

      où num est le numéro de l'IT (50 par exemple)