



Lösung zur 2. Übung zur Vorlesung Systemprogrammierung I

Aufgabe 1: Interrupts (2+3+2)

In den Vorgaben ist eine Methode `InsertHandler` für das Einfügen von Interrupts vorhanden.

a) Erläutern Sie, wo der Interrupt-Handler eingefügt wird.

Bei dem Interrupthandler handelt es sich um eine normale Java-Methode. Das Objekt, welches die Methode enthält, wird in ein Array eingefügt. Hierbei wird die Position im Array anhand der gewünschten Geräteinterrupt-Nummer gewählt.

b) Wie sieht in Plurix der Ablauf eines Interrupts vom Controller bis zum Aufruf des jeweiligen Handlers aus?

Der Controller signalisiert den IRQ an die CPU. Nach dem `IntAck` durch die CPU wird der Index in die IDT auf den Bus gelegt. Sobald die CPU diesen entgegengenommen hat wird dies durch einen weiteren `IntAck` an den Controller bestätigt. Anhand dieses Offsets wird ein Handler aus der IDT ausgewählt und ausgeführt. Hierbei handelt es sich um einen `FirstLevelHandler`. Dieser ermittelt die Nummer des aktuellen IRQ's / der aktuellen Exception und wählt aus einem Handler-Array den entsprechenden IRQ- / Exception-Handler aus.

c) Wie wird in Plurix die Nummer des aktuellen Interrupts / Exception ermittelt?

Plurix verwendet einen einzelnen gemeinsamen Adressraum für alle Threads / Transaktionen. Es werden daher keine unterschiedlichen Segmentbereiche benötigt. Alle Segmentdeskriptoren verweisen auf den selben Adressbereich.

In der IDT kann für jeden Interrupthandler angegeben werden, in welchem Segment er sich befindet. Durch die Verwendung von vielen Segmenten, die alle auf den selben Bereich abgebildet sind, kann unter Zuhilfenahme des aktuellen Code-Segments ermittelt werden welcher Eintrag der IDT aufgerufen wurde. Der gerufene `FirstLevelHandler` muß folglich lediglich das aktuell aktive Codesegment aus dem `cs` Register auslesen und auswerten.

Aufgabe 2: Tastatur (2+3+2)

a) Lesen Sie die Scancodes der Tastatur aus und geben Sie diese auf dem Bildschirm aus. Verwenden Sie für die Abfrage der Tastatur eine einfache Schleife.

siehe Quellcode

b) Erweitern Sie Ihren Treiber und berücksichtigen Sie zusätzlich die Tasten „Shift“, „Alt“ und „Strg“ und geben Sie diese zusammen mit den Scancodes aus (siehe Beispiel).

siehe Quellcode

c) Ändern Sie Ihren Treiber, so daß er interrupt-basiert arbeitet.

siehe Quellcode

Beispielausgabe:

```
Scancode: 2 // 1 gedrückt
Scancode: 130 // 1 losgelassen
Scancode: 8 Modifier: shift // shift + 7 gedrückt
```

```
Scancode: 136   Modifier: shift      // shift gedrückt; 7 losgelassen
Scancode: 5    Modifier: shift, strg // shift+Strg+4 gedrückt
Scancode: 5    Modifier: shift      // strg losgelassen
Scancode: 5    // shift losgelassen
Scancode: 133 // 4 losgelassen
```

Aufgabe 3: Tastatur-Ringpuffer (3)

Entwickeln Sie nun einen verbesserten Tastaturhandler, der über einen 128 Bytes großen Ringpuffer für Scancodes verfügt. Bei einem Pufferüberlauf soll eine Fehlermeldung ausgegeben werden. Ferner soll Ihr Tastatur-Treiber folgende Methoden anbieten:

```
int ReadKey()
boolean KeyAvailable()
```

ReadKey: liest jeweils einen Scancode aus dem Tastaturpuffer aus und gibt diesen zurück. Ist kein Scancode im Puffer vorhanden, so wartet ReadKey solange, bis sich ein Scancode im Puffer befindet.

Key Available: liefert true zurück, wenn sich mindestens ein Scancode im Tastaturpuffer befindet, ansonsten wird false zurückgegeben.

siehe Quellcode