Just-In-Time Compiler

Rainer Müller

Department Informatik 4
Verteilte Systeme und Betriebssysteme
Friedrich-Alexander-Universität Erlangen-Nürnberg

WS 2014/2015

Fragestunde

Fragen zu Aufgabe 1?

Ausführen einer Instruktion

Welche Schritte müssen bei der Emulation folgender Instruktion ausgeführt werden?

```
83 2c 19 01                    subl    $0x1,(%ecx,%ebx,1)
```

Just in Time Compiler

Zerlegen Sie folgenden Code in Grundblöcke, und generieren Sie hierfür laufzeitübersetzen Code:

```
nullify:
        xorl %ecx, %ecx
nullify_loop:
        movb $0, (%ecx, %ebx)
        incl %ecx
        cmpl $500, %ecx
        jb nullify_loop
        ret
```

Klassifikation Unterbrechungen

Klassifikation aus Betriebssystemsicht

- **Interrupts**: Hardware erfordert Aufmerksamkeit
  $\Rightarrow$ Vermeiden von aktivem Warten
- **Exceptions**: Fehler bei Instruktionsausführung
  $\Rightarrow$ gezielte Fehlerbehandlung
- **Traps**: Wechsel in anderen Betriebsmodus (System Call)
  $\Rightarrow$ Rechteverwaltung, Mehrbenutzersystem

Eigenschaften Interrupts

- asynchron
- Quelle: extern
- nicht vorhersagbar
- nicht wiederholbar
- sollte transparent zum laufenden Prozess sein
- Beispiel: Tastaturcontroller hat Zeichen von Tastatur empfangen

Eigenschaften Exceptions

- Quelle: intern
- synchron
- oft(?) vorhersagbar
- oft(?) wiederholbar
- nicht beabsichtigt
- möglicherweise transparent zum laufenden Prozess
- Beispiel: Division durch 0
- Beispiel: Page Fault

Eigenschaften Traps

- synchron
- vorhersagbar
- wiederholbar
- beabsichtigt
- verändert den laufenden Prozess
- Beispiel: `int $0x80`

Anforderungen an Hardware

- Unterbrechen des Kontrollflusses
  - während oder
  - vor/nach

  der Verarbeitung von Instruktionen
- Sichern des CPU-Zustands
- Sprung an festgelegte Behandlungsroutine
- Mitteilen der Quelle/Art der Unterbrechung, ggf. auch weiterer Informationen (Traps)
- Restaurieren des CPU-Zustands
- Wiederholen der Instruktion (Interrupts, Exceptions) *oder*
- Ausführen der Folgeinstruktion (Traps)

Realisierung x86

- Register `IDTR` zeigt auf Vektortabelle `IDT`
- Vektortabelle `IDT` enthält
  - Adresse der Behandlungsroutine
  - Informationen über Art der Unterbrechung
  - Anzahl an Bytes, die vom Stack kopiert werden sollen
  - und noch einige weitere Details
- Interrupts
  - Pin am Prozessor kann Interrupt auslösen
  - CPU liest Byte vom Bus, welches den Offset in `IDT` angibt
- Exceptions
  - werden intern generiert
  - vordefinierter Offset für `IDT`, bzw.
- Traps
  - werden intern generiert
  - Parameter als Offset für `IDT`

Unterbrechungsbehandlung

1. Zustandssicherung:
   1.1 EFLAGS $\rightarrow$ Stack
   1.2 CS $\rightarrow$ Stack
   1.3 EIP $\rightarrow$ Stack
2. bei Interrupts: `IF=0`
3. Eintrag aus Interrupttabelle wird angesprungen
4. *Behandlungsroutine wird ausgeführt*
5. *IRET*
6. `EFLAGS`, `CS` und `EIP` werden vom Stack geholt und der Zustand somit wiederhergestellt

Unterbrechungsbehandlung

- Wodurch wird ausgewählt, ob eine Instruktion wiederholt werden soll?
- Wer ist für die Wiederherstellung des Flags `IF` verantwortlich?
- Wie können bei Traps – abgesehen mittels Stack – Parameter an das Betriebssystem übergeben werden?