CHIP PROGRAMMIER-PRAXIS 84029/92003
34,– DM

Assembler

Ausgabe
3

Assembler auf dem PC

- Der Protected-Mode und seine Tücken
- Grafik im Textmodus
- Superschnelle Fenster-Technik
- Compiler-Code gezielt optimieren
- Programmierung von Super-VGA-Karten
- OOP in Assembler
- Der integrierte Assembler von TP 6.0

Im Heft: Alle Sources auf Diskette

Interrupt-gestütztes Debugging

Eine alte Regel und viel Programmiererfahrung besagen, daß 20 Prozent eines Programmes 80 Prozent der gesamten Entwicklungszeit beanspruchen. Die Fehlersuche, das sogenannte Debugging, nimmt dabei wahrscheinlich den größten Teil dieser 80 Prozent in Anspruch. Die Folgerung: Es besteht ein Bedarf an Hilfsprogrammen zur effektiven Fehlersuche.

Bewährte Hilfsmittel wie der Turbo Debugger oder gar das gute alte DEBUG leisten bei kleinen, kompakten Programmen gute Dienste. Aus diesem Grund werden die folgenden Zeilen ein Hilfsprogramm vorstellen, welches zwar bei einfachen Programmen das Debugging kaum unterstützt, bei großen Projekten jedoch Fehlerquellen sehr gut einzukreisen vermag.

Ein besonders gefürchteter Fehler zeigt folgendes Symptom: Ein neues Projekt sollte eigentlich komplett laufen, hängt sich an einer unbekannten, dubiosen Stelle aber unweigerlich auf.

Die zahlreichen Bootversuche mit eingerechnet, vergehen im Durchschnitt mehrere Stunden, bis die Fehlerquelle gefunden ist. Hier setzt INT.ASM an. Dieses Programm - es befindet sich übrigens auf der Heft-Diskette - kommentiert jeden Aufruf eines bestimmten Interrupts auf dem Drucker beziehungsweise dem Bildschirm, ohne den Ablauf eines anderen Programmes zu stören.

Nehmen wir einmal an, Ihr Programm ruft des öfteren den Interrupt 33h (Maus- Interrupt) auf. Programmieren Sie in diesem Fall INT.ASM auf Interrupt 33h, und lassen Sie sich jeden einzelnen Aufruf auf dem Drucker dokumentieren, bis sich der Computer an der gesuchten fehlerhaften Stelle aufhängt.

Die Dokumentation zu den einzelnen Interrupt-Aufrufen besteht aus allen wichtigen Registern (AX, BX, CX, DX, ES, DI, BP, SI, DS) sowie derjenigen Offset- und Segmentadresse, von der aus der besagte Interrupt aufgerufen wurde (IP und CS). Anhand dieser Daten sind Sie in der Lage, festzustellen, zwischen welchen Interrupt-Aufrufen der Rechner ins Nirwana hinüberschritt und wieviele Male bestimmte kritische Adressen angesprungen wurden.

Schließlich sind hartnäckige Fehler geistig schon so weit, nicht beim ersten Aufruf abzustürzen, wäre die Fehlersuche dadurch doch drastisch verkürzt. Der Aufruf von INT.ASM geschieht nach erfolgter Assemblierung folgendermaßen:

```
INT [/P] [/S] INTERRUPTNUMMER
```

Die Schalter /P und /S müssen bei der Deinstallation nicht angegeben werden. Die Installation verlangt mindestens einen der besagten Schalter.

```
/P : Ausgabe der Interruptdokumentation auf dem
     Drucker

/S : Ausgabe der Interruptdokumentation auf dem
     Bildschirm
```

Die Angabe der INTERRUPTNUMMER muß ein- oder zweistellig erfolgen. Die Zahl selbst hat im Hexadezimalsystem vorzuliegen.

Zuerst sollten Sie sich ein wenig an INT.ASM gewöhnen. Starten Sie dazu INT.ASM wie folgt:

```
INT /P /S 33
```

oder, wenn Sie keinen Drucker besitzen, mit:

```
INT /S 33
```

Alle zu diesem Beitrag gehörenden Files finden Sie im Unterverzeichnis DEBUG auf beiliegender Diskette

Bei komplexeren Projekten und dem 67. Breakpoint aber geht selbst bei Borland die Übersicht verloren

Geben Sie dazu folgendes Programm mit DEBUG von MS-DOS oder SID von DR-DOS ein:

```
mov  ax, 0000
int  33
mov  ax, 0001
int  33
mov  ax, 0002
int  33
mov  ax, 0003
int  33
xor  ax, ax
int  21h
```

Nach dem Start werden Sie die Aktivitäten von INT.ASM schnell bemerken. Auf dem Bildschirm und dem Drucker (falls Sie einen besitzen) vermerkt es die wichtigsten Registerwerte. Unter CS:IP finden Sie diejenige Adresse, von welcher aus der Interrupt aufgerufen wurde.

Vergleichen Sie diese Werte ruhig einmal mit DEBUG beziehungsweise SID. Hätte sich zwischen diesen Zeilen auch ein Fehler eingeschlichen, beispielsweise ein Sprung ins Nirgendwo, wäre ein Einsatz von INT.ASM sicher nicht unbedingt notwendig gewesen. Doch stellen Sie sich einmal vor, zwischen den besagten Zeilen stünden noch mehrere Kilobyte Quellcode und der Computer verabschiedete sich beim Aufruf. In solchen Fällen leistet INT.ASM gute Dienste.

Folgende Zeile erlaubt eine Deinstallation von INT.ASM:

```
INT 33
```

Doch INT.ASM kann noch mehr. Nehmen wir beispielsweise an, Sie haben ein Programm geschrieben, das über den Interrupt 13h verschiedene Daten nachladen soll, was es nicht zu tun scheint. INT.ASM, aktiviert für Interrupt 13h, wir in diesem Fall alle verwendeten Aufrufe des Interrupt 13 kommentieren, und Sie können leicht feststellen, ob die Reihenfolge der Interrupts stimmt. In einem verschachtelten Programm eine falsche Reihenfolge zu entdecken, würde mit herkömmlichen Mitteln an Zauberei grenzen und eine Menge Arbeit kosten.

Bei der Arbeit mit INT.ASM stößt der Anwender auf eine Unannehmlichkeit, die hier nicht verschwiegen werden soll. Haben Sie beispielsweise INT.ASM auf Interrupt 13 (Disketten-Interrupt) programmiert, so wird nicht nur Ihr Programm, sondern auch DOS auf diesen Interrupt zurückgreifen, und das nicht gerade selten. Bis Ihr Programm geladen wäre, hätte unter normalen Umständen schon die eine oder andere Seite Papier geopfert werden müssen.

Aus diesem Grund folgender Tip: Laden Sie ein fehlerhaftes Programm über den Turbo Debugger und schalten Sie Ihren Drucker erst ein, wenn beide Programme, der Turbo Debugger und Ihr fehlerhaftes Programm, komplett im Speicher stehen. In diesem Fall wird der Drucker alle Interrupt-Aufrufe nach dem Einschalten kommentieren und die vorhergehenden übergehen. Leider funktioniert der Trick nicht, wenn Sie die Daten auf dem Bildschirm ausgeben wollen oder den Drukker einfach auf Standby-Betrieb schalten. Letzteres würde dazu führen, daß das ablaufende Programm anhält und auf den Drucker wartet.

Markus Mück