Einführung in CORE WAR Teil 6

Die Kämpfer bekommen Kinder

Ein völlig anderes Antlitz eines Kämpfers – man muss fast sagen Wesen – bietet das Programm *Mäuse. Mäuse* schafft sich unablässig Ebenbilder und haucht ihnen Leben ein; diese tun es dann - der Apfel fällt nicht weit vom Stamm - den Alten gleich. Es entsteht eine ganze Mäusepopulation in der sonst so lebensfeindlichen Arena. Und nicht nur das ist an *Mäuse* bemerkenswert. Der Population sind in der endlichen Speicherarena Grenzen gesetzt, so dass die Anzahl der *Mäuse* nicht ins Unermessliche wachsen kann. *Mäuse* hat für dieses Problem eine natürliche Lösung gefunden: Sobald auf einer alten *Maus* eine neue entsteht, räumt jene das Feld, indem sie freiwillig den bitteren Kelch der **DAT-**Anweisung leert.

```
; Redcode-Programm "Maeuse":

Zaehler DAT  #0          #0
Start   MOV  #Laenge     Zaehler
Kopier  MOV  @Zaehler   <Ziel      ; Es wird alles bis auf die erste
        DJN   Kopier     Zaehler   ; Anweisung kopiert.
        SPL  @Ziel                 ; Die neue Maus ist da.
        ADD  #653        Ziel      ; Das neue Kopierziel wird eingestellt.
        JMZ   Start      Zaehler   ; Selbstmord, falls `was schief ging.
Ziel    DAT  #0         #833

Laenge  EQU   Ziel-Zaehler
        END   Start
```

Hier lernen Sie endlich die eminent wichtige **SPL-**Anweisung kennen, ohne die ein erfolgreiches Redcode-Programm heute nicht mehr auskommen kann. Sie ermöglicht einem Kämpfer, sich in zwei Kämpfer, sogenannte Tasks, aufzuteilen und sich dadurch mehrere Leben zuzulegen. Wenn also ein Task eines Spielers einer Bombe zum Opfer fällt, sind die anderen Tasks und damit der Kämpfer noch lange nicht am Ende. Da im Krieg der Kerne zwar hart, aber fair gekämpft wird, darf ein Spieler nicht dadurch öfter zum Zuge kommen als sein Gegner, dass er sich mit Hilfe der **SPL**-Anweisung möglichst oft teilt. Die Tasks eines Kämpfers müssen sich das zur Verfügung stehende Kontingent an Zügen aufteilen. Nehmen Sie z. B. an, dass sich ein Kämpfer in vier Tasks aufgeteilt hat. So kommt jeder der Tasks jedes achte Mal zum Zuge (zwischendurch ist ja der Gegner an der Reihe).

Das Programm *Mäuse* besteht aus einer Kopierschleife, wobei hier besonderes Augenmerk auf die **SPL-**Anweisung zu legen ist. Wenn sie erreicht ist, hat sich *Mäuse* bereits einmal 833 Speicherzellen weiter kopiert, und der Operand **B** der **DAT-**Anweisung, die durch Ziel markiert ist, enthält die relative Anfangsadresse der Kopie. Wird die **SPL-**Anweisung ausgeführt, so wird ein neuer Task generiert, der mit der Ausführung in der Speicherzelle beginnt, die durch den Operanden **A** der **SPL-**Anweisung angegeben wird. Der Mutter-Task fährt normal mit der nächsten Anweisung fort. Danach wird durch die **ADD-**Anweisung die Position für die nächste Kopie von *Mäuse* festgelegt.

Die **JMZ-**Anweisung ist eine weitere Sprunganweisung. Es wird zu der Speicherzelle, die der Operand **A** angibt, unter der Bedingung gesprungen, dass der Operand **B** einen Wert bezeichnet, der gleich Null ist. Sie werden sich sicherlich fragen, weshalb hier nicht anstatt der **JMZ-**Anweisung die **JMP-**Anweisung verwendet wurde, um auf jeden Fall sicherzustellen, dass *Mäuse* nicht auf die **DAT-**Anweisung bei Ziel läuft. Hierin liegt unter anderem der Clou von *Mäuse:* Der Wert des Operanden B der **JMZ-**Anweisung ist der aktuelle Wert des Kopierzählers, der nach Beendigung der Kopierschleife den Wert Null hat. Ist nun irgend etwas während des Kopierens schiefgegangen - sei es durch den Einschlag einer gegnerischen Bombe oder durch eine weitere *Maus -,* ist es sehr unwahrscheinlich, dass der Kopierzähler den Wert Null hat. Der Task erkennt nun, dass er beschädigt wurde und somit nicht mehr voll funktionsfähig ist. In dieser Situation ist es sinnvoll, Selbstmord zu begehen, damit den noch funktionstüchtigen *Mäusen* das maximale Kontingent an Zügen zur Verfügung steht.

Mäuse wurde 1986 unter seinem englischen Namen Mice Weltmeister in Core War.

Quelle: Aus dem Begleitheft zu Core War – Krieg der Kerne, Spektrum der Wissenschaft 1993, S. 33 ff.