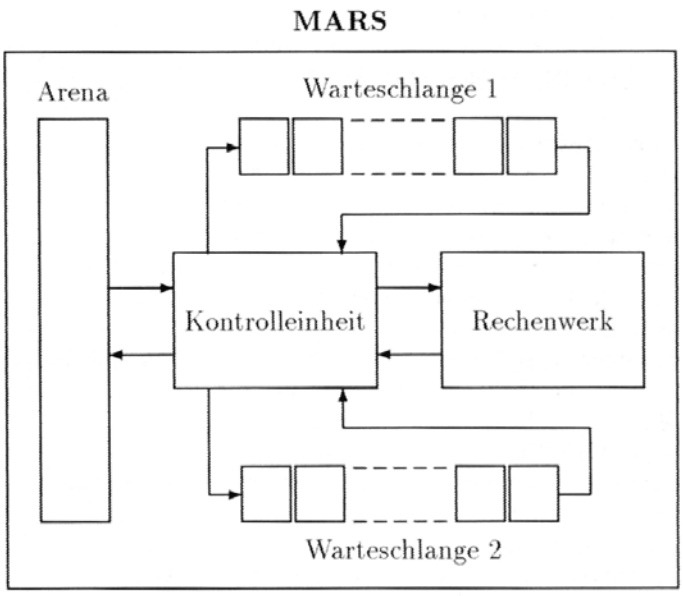
Die MARS-Maschine

als Von-Neumann-Rechner

Es ist sinnvoll, sich als Schauplatz des Kriegs der Kerne einen eigenen, recht primitiven Computer namens MARS vorzustellen. Dieser ist aus rein praktischen Gründen (wer will sich schon eigens für dieses Spiel einen Computer zusammenlöten?) in den umgebenden Rechner eingebettet, indem ein Programm innerhalb des Wirtsrechners die Wirkung seiner Komponenten modelliert. Ein Blick in die Struktur dieses virtuellen Computers ist schon deswegen interessant, weil er – mit einer Variante – das Prinzip des klassischen, 1946 von Arthur W. Burks, Herman H. Goldstine und John von Neumann vorgeschlagenen Rechners in Reinform verwirklicht. Die meisten heutigen – nicht-parallelen – Computer arbeiten nach diesem Prinzip, wenngleich der Benutzer sich dank einer Fülle von Anwendungsprogrammen in der Regel darüber keine Gedanken mehr machen muss. Allerdings kann ein Von-Neumann-Rechner in der Urform nur jeweils ein Programm abarbeiten. MARS ist insofern etwas moderner, als er zwei Programme zu steuern vermag, die ihrerseits mit der SPL-Anweisung noch Tochterprozesse starten können.

Der MARS-Rechner besteht aus einem Speicher (der Arena), in dem sich die Programme mit ihren Daten befinden, zwei Warteschlangen, in denen die Prozesse der beiden Kämpfer verwaltet werden, einem Rechenwerk, das die arithmetischen und logischen Operationen einzelner Redcode-Anweisungen ausführt, und schließlich einer Kontrolleinheit, welche die jeweils nächste auszuführende Anweisung bereitstellt, eventuell vorverarbeitet (abhängig von der Adressierungsart) und an das Rechenwerk zur Ausführung übergibt.



Eine Warteschlange *(queue)* ist ein Datenspeicher aus einer Folge gleichartiger Komponenten. Wie eine echte Warteschlange am Fahrkartenschalter ändert sie sich nur dadurch, dass an einem Ende (der Eingabeseite) weitere Komponenten angefügt oder am anderen Ende (der Ausgabeseite) Komponenten weggenommen werden. Was zuerst angefügt worden ist, kommt auch zuerst wieder heraus. Man bezeichnet Warteschlangen daher auch First-In-First-Out-Speicher, abgekürzt FIFO. Die Warteschlangen von MARS bestehen aus den Adressen von Anweisungen, die zur Ausführung vorgemerkt werden.

Das Zusammenwirken der Komponenten von MARS ist einer näheren Betrachtung wert. Normalerweise bringt MARS die Kämpfer zu Beginn an zufällige Positionen in der Arena, wobei ein vorgegebener Minimalabstand einzuhalten ist. Jede der beiden Warteschlangen enthält zu Anfang die Speicheradresse, mit der die Ausführung des zugehörigen Kämpfers begonnen werden soll. Nicht von den Kämpfern belegte Speicherzellen erhalten zu Beginn den Inhalt **`DAT #0 #0`**. Durch Zufallsauswahl wird festgelegt, welcher Kämpfer beginnt.

Wir nehmen an, dies sei Kämpfer 2. Dann entfernt die Kontrolleinheit aus der Warteschlange 2 die erste Adresse und kopiert sich die an dieser Adresse im Speicher liegende Redcode-Anweisung. In der Kopie ersetzt sie zunächst irgendwelche in der Anweisung vorhandenen indirekten Adressierungen durch die effektiven Adressen. Die so bereinigte Form der Anweisung lautet nun z. B. **MOV (2002) (2001)**, wobei die Klammern die effektiven Adressen kennzeichnen.. Die bereinigte **MOV**-Anweisung wird an das Rechenwerk zur Ausführung übergeben. Die Ausführung einer Anweisung verändert stets die Arena oder die betreffende Warteschlange oder beides.

Nach Ausführung der ersten Anweisung von Kämpfer 2 ist Kämpfer l an der Reihe. Abgesehen davon, dass für ihn die Warteschlange I zuständig ist, ist der Ablauf völlig analog dem für Kämpfer 2 beschriebenen. So wird immerfort abwechselnd für beide Kämpfer die Adresse der auszuführenden Anweisung aus der entsprechenden Warteschlange weggenommen und die Adresse der nächsten Anweisung hinten an die Warteschlange angehängt. Was ist die nächste Anweisung? Im Normalfall nach dem Von-Neumann-Prinzip die mit der nächsthöheren Adresse. Wenn jedoch bei einer Sprunganweisung die Sprungbedingung erfüllt ist, kommt die Adresse des Sprungziels in die Warteschlange.

Eine weitere Ausnahme bilden die Anweisungen **SPL** und **DAT**. Die Anweisung **SPL** hat ja den Sinn, einen weiteren Prozess zu eröffnen. Also wird zuerst die in der **SPL**-Anweisung angegebene Adresse und sofort danach die Adresse der auf die **SPL**-Anweisung folgenden Speicherzelle an die Warteschlange angehängt. Da vorher die Adresse der Anweisung selbst aus der Warteschlange entfernt wurde, hat sich die Länge der Warteschlange insgesamt um eins erhöht: Ein neuer Prozess ist geboren. Im Gegensatz dazu bewirkt die Ausführung der **DAT**-Anweisung nur das Entfernen der eigenen Adresse, aber keinerlei neuen Eintrag in die Warteschlange: Der Prozess hat ein Leben verloren. Stößt die Kontrolleinheit auf eine leere Warteschlange, so hat der zugehörige Kämpfer sein letztes Leben ausgehaucht und damit den Kampf verloren.

Quelle: Helge Robitzsch, Neues vom Krieg der Kerne, Spektrum der Wissenschaft (Begleitheft zur Diskette), Heidelberg, 1993, S. 83 ff.