



no invalid or jump instructions will continue executing successive instructions endlessly, eventually returning to the instruction where it started.

### Low level multiprocessing

Instead of a single instruction pointer a Redcode simulator has a process queue for each program containing a variable number of instruction pointers which the simulator cycles through. Each program starts with only one process, but new processes may be added to the queue using the SPL instruction. A process dies when it executes a DAT instruction or performs a division by zero. A program is considered dead when it has no more processes left.

### No external access

Redcode and the MARS architecture provide no input or output functions. The simulator is a closed system, with the only input being the initial values of the memory and the process queues, and the only output being the outcome of the battle, i.e., which programs had surviving processes. Of course, the simulator may still allow external inspection and modification of the memory while the simulation is running.

## Strategy

Warriors are commonly divided into a number of broad categories, although actual warriors may often combine the behavior of two or more of these. Three of the common strategies (replicator, scanner and bomber) are also known as paper, scissors and stone, since their performance against each other approximates that of their namesakes in the well-known playground game.[3]

### Paper (or replicator)

A replicator makes repeated copies of itself and executes them in parallel, eventually filling the entire core with copies of its code. Replicators are hard to kill, but often have difficulty killing their opponents. Replicators therefore tend to score a lot of ties, particularly against other replicators.

A silk **is** a special type **of** very rapid replicator, named after Silk Warrior[4] **by** Juha Pohjalainen. Most modern replicators are **of this** type. Silk replicators use parallel execution to copy their entire code with one instruction, **and** begin execution **of** the copy before it **is** finished.[5]

### Scissors (or scanner)

A scanner is designed to beat replicators. A scanner does not attack blindly, but tries to locate its enemy before launching a targeted attack. This makes it more effective against hard-to-kill opponents like replicators, but also leaves it vulnerable to decoys. A scanner usually bombs memory with SPL 0 instructions. This causes the enemy to create a huge number of processes which do nothing but create more processes, slowing down useful processes. When the enemy becomes so slow that it is unable to do anything useful, the memory is bombed with DAT instructions. Scanners are also generally more complex, and therefore larger and more fragile, than other types of warriors.[6]

A one-shot **is** a very simple scanner that only scans the core **until** it finds the first target, **and then** permanently switches to an attack strategy, usually a core clear. Myrmidon[7] **by** Roy van Rijn **is** an example **of** a oneshot.

### Stone (or bomber)

A bomber blindly copies a “bomb” at regular intervals in the core, hoping to hit the enemy. The bomb is often a DAT instruction, although other instructions, or even multi-instruction bombs, may be used. A bomber can be small and fast, and they gain an extra edge over scanning opponents

since the bombs also serve as convenient distractions. Bombers are often combined with imp spirals to gain extra resiliency against replicators.

### **Vampire (or pit-trapper)**

A vampire tries to make its opponent's processes jump into a piece of its own code called a "pit". Vampires can be based on either bombers or scanners. A major weakness of vampires is that they can be easily attacked indirectly, since they must by necessity scatter pointers to their code all over the core. Their attacks are also slow, as it takes an extra round for the processes to reach the pit. myVamp[8] by Paulsson is an example of a vampire.

### **Imp**

Imps are named after the first ever published warrior, Imp[9] by A. K. Dewdney, a trivial one-instruction mobile warrior that continually copies its sole instruction just ahead of its instruction pointer. Imps are hard to kill but next to useless for offense. Their use lies in the fact that they can easily be spawned in large numbers, and may survive even if the rest of the warrior is killed.

An imp ring (or imp spiral) consists of imps spaced at equal intervals around the core and executing alternately. The imps at each arm of the ring/spiral copy their instruction to the next arm, where it is immediately executed again. Rings and spirals are even harder to kill than simple imps, and they even have a (small) chance of killing warriors not protected against them. The number of arms in an imp ring or spiral must be relatively prime with the size of the core.

### **Quickscanner (or q-scan)**

A quickscanner attempts to catch its opponent early by using a very fast unrolled scanning loop. Quickscanning is an early-game strategy, and always requires some other strategy as a backup. Adding a quickscanning component to a warrior can improve its score against long warriors such as other quickscanners. However, the unrolled scan can only target a limited number of locations, and is unlikely to catch a small opponent.

### **Core clear**

A core clear sequentially overwrites every instruction in the core, sometimes even including itself. Core clears are not very common as stand-alone warriors, but are often used as an end-game strategy by bombers and scanners.

## **Redcode**

Redcode is the programming language used in Core War. It is executed by a virtual machine known as a Memory Array Redcode Simulator, or MARS. The design of Redcode is loosely based on actual CISC assembly languages of the early 1980s, but contains several features not usually found in actual computer systems.

Both Redcode and the MARS environment are designed to provide a simple and abstract platform without the complexity of actual computers and processors. Although Redcode is meant to resemble an ordinary CISC assembly language, it differs in many ways from "real" assembly.