Table of Contents

Core War

for Windows
v1.00

NOTE: This program is shareware.
Please support it by registering.

Core War for Windows v1.00 is a game in which user-written programs battle for supremacy in a virtual computer's memory. This simulator allows you to create and edit your own programs, or load other ICWS '88 standard programs from other simulators, as well as executing them in battle. Core War for Windows displays all addresses in memory, and memory addresses occupied by competing programs are color coded. See Overview.

This version of Core War for Windows is shareware. If you have the unregistered version, or if you have the registered version, but have not registered it yourself, please support this application by registering. Of course, comments and suggestions are welcome. For more information, choose Registration.

Overview

Core War for Windows v1.00 is a fully, ICWS '88 Redcode standard compliant Core War simulator. Core War is a game in which two (or more) user-written programs are placed in a virtual computer's memory, and each program attempts to survive the others through various strategies. The programs are written in a simple form of assembly language called Redcode; there are only eleven instructions in this standard. Once compiled and placed in the virtual computer's memory, called the "core," the programs are executed by the Memory Address Redcode Simulator (MARS), which is simply the entity that manages the emulation of the execution.

The core has 8000 memory addresses and is circular. Memory addressed above the top of memory (over 7999) automatically rolls over to the bottom of memory (0). Likewise, programs that address below the bottom of memory (below 0) are wrapped around to the top of memory. Programs never know exactly where in memory they are, although they may address memory relative to themselves.

Redcode programs are ASCII text files ending with the default extension, .WAR. They may be edited through any editor, and are selected for Battle through the Battle Setup button in the main dialog box.

See Also:
Core War
Battle Setup
Editor
Redcode Language
International Core War Society

International Core War Society

The International Core War Society (ICWS) is an organization of Core War enthusiasts from all over the world. The ICWS is responsible for a number of duties.

The ICWS handles an annual International Core War Tournament, where programs submitted by members battle for supremacy over other competitors. New programs and strategies come into focus during the competition.

The ICWS also publishes a quarterly newsletter, *The Core War Newsletter (TCWN)*, for its members. *TCWN* contains articles of interest for members, including information about the Core War standard, example programs, and other subjects related to Core War.

Also published by the ICWS, is the Redcode standard, ICWS '88. This is the standard that Core War simulators should meet.

Members of ICWS pay US$15.00 a year. The current director of the ICWS is Jon Newman, and new registering members should make US checks of $15.00 payable to "Jon Newman" (NOT ICWS) and sent to the address below:

Jon Newman (check payable to)
Director, ICWS
13824 NE 87th St.
Redmond WA 98052-1959
USA

Once a member, you will receive a copy of the ICWS '88 standard, four issues of *TCWN*, and an entry slot in the annual tournament. Once a ICWS member, you may purchase from them back issues of TCWN, the '88 Standard, and the "Big Disk" which is a DOS version disk containing simulators, etc. For more information, please send questions to Jon Newman, address above. If you have Internet access, you may contact Jon Newman at jonn@microsoft.com, which is his e-mail box (Microsoft has nothing to do with the ICWS).

If you have FTP access, you may find Core War materials at soda.berkeley.edu in the directory /pub/corewar. There also exists a newsgroup called rec.games.corewar. For more information on the newsgroup, contact Stefan Strack, stst@vuse.vanderbilt.edu.

See Also:
Overview
Core War
Redcode Language

Redcode Language

Redcode is a simple form of assembly language composed of a limited number of instructions. This Core War simulator follows the International Core War Society's '88 Redcode standard. This standard is comprised of 11 instructions and two pseudo-instructions. Each instruction fits in exactly one location in memory. Included with these instructions are four modes to address memory: direct, immediate ("#"), indirect ("@"), and predecrement indirect ("<"). Since Redcode does not allow the programmer to know, or directly access, absolute locations in memory, it does allow the programmer to reference memory in a relative manner, with addressing occurring through offsets from within the Redcode program itself. Memory is considered circular, so that memory addressed passed the top of memory, just circles itself down to the bottom of memory. Addressing memory below the bottom, will also wrap it around to the top.

Each of the 11 instructions may contain up to two operands, the A operand and the B operand. Along with the A and B operand, are the A mode and the B mode, which is the addressing mode that particular operand shall take on. The addressing modes dictate how the operands should be interpreted by the Memory Address Redcode Simulator (MARS). Also note, that certain operands in certain instructions only will accept certain modes. For example, the B-Operand of the MOV instruction can not be immediate.

It is beyond the scope of this help to completely explain the Redcode language. For more information, see ICWS.

Following is the list of instructions, their operands, and a brief explanation of each. Operands that are optional are contained within brackets ("[A]"). Operands should, but needn't always have to be, separated from each other by a comma. Comments begin with a semi-colon and continue until the end of the line. Instructions may be proceeded by labels.

See available Redcode programs for examples.

The 2 pseudo-instructions:

label EQU expression
- Equate - compiler will substitute expression for every occurrence of label found in the program.

END [label]
- End - notifies the compile that execution of this program should begin at label. If no label is included, or no END statement included, execution will begin at the first instruction.

The 11 Redcode instructions:

DAT [A,] B
- Data - Operand B is used to store a value. This statement can not be executed, and if attempted, will kill the program.

MOV A, B
- Move - Contents of address A are copied to address B.

ADD A , B
- Add - Contents of address A are added to contents of address B.

SUB A , B
- Subtract - Contents of address A are subtracted from contents of address B.

JMP A [, B]
- Jump - Transfers execution to address A.

JMZ A, B
- Jump if zero - Transfers execution to address A if contents of address B are zero.

JMN A, B
- Jump if not zero - Transfer execution to address A if contents of address B are not zero.

DJN A, B
- Decrement; jump if not zero - Subtract 1 from contents of address B and transfer execution to address A if contents of address B are not zero.

CMP A, B
- Compare - Compare contents of address A with contents of address B, and if equal, skip next instruction.

SLT A, B
- Skip if less than - Compare contents of address A with contents of address B, and if A < B, skip next instruction.

SPL A [, B]
- Split - Continue execution, but also begin execution process at address A.

See Also:
Overview
Core War
Editor
International Core War Society

Core War

The Core War for Windows main dialog box contains a large, black rectangle representing the core. Above the core is a title box that contains the names and colors of the programs loaded into memory, if any. This title box will also display the result of the battle. To the right of the title box is a number labelled "Cycle:." This number is updated continuously once a battle begins. Below the "Cycle:" number is a number labelled "Memory:." This is how much memory is in the core. Beside the core is set of buttons that give you access to all of the features of Core War.

Battle Setup - you select this button when you wish to load into the core programs for a battle. This button will take you to another dialog box.

WAR! - once programs have been successfully loaded into the core, you can start the battle by selecting this button. While a battle is in progress, this button will change to "STOP!" and choosing it will halt the battle.

Editor - select this when you wish to edit Redcode programs. Selecting this button will also maximize the Messages dialog box so that if you wished to see any error messages in it while you edit a program, you may.

Messages - this button will maximize the Messages dialog box. To minimize the Messages dialog, select the minimize arrow in the upper right hand corner of that dialog box.

Preferences - select this when you want to change simulator options such as maximum cycles or maximum programs. This option will take you to the Preferences dialog box.

Help - this help.

Exit - select this to quit the program.

See Also:
Overview
Battle Setup
Editor
Messages
Preferences
Redcode Language

Battle Setup

Battle Setup is where you choose which warriors will fight in the next battle. The number of warriors that you may choose for one battle may be one to the maximum number set under the Preferences dialog box. This may never by higher than 20 programs.

To select a warrior to enter battle, select the warrior's name for the "Available Warriors" list-box on the left, and hit the "Add >>" button. You will see that warrior's name appear in the "Committed Warriors" list-box on the right. When you have reached the maximum number that you may enter, the "Add >>" button will disable itself, and Core War will beep if you attempt to add any more warriors.

To remove a committed warrior, select that warrior's name from the "Committed Warriors" list-box on the right, and hit the "<< Remove" button. That warrior's name will disappear.

Warriors can be from different directories and have different extensions. Select the correct file extension out of the "Extension" combo box when you are searching for warriors to include in the battle. If the warrior program is located in another directory, choose that directory from the "Directory" combo box. You may select warriors to fight in the same battle even if they have different extensions and/or directories.

When you are finished selecting all the warriors for the upcoming battle, hit the "OK" button. If you wish to return to the Core War dialog without setting up a battle, select the "Cancel" button.

Once you select "OK", Core War will compile all the selected programs and return you to the main dialog box. If all programs are compiled successfully, you will see colored lines and dots in the core memory (these are where the programs have been loaded), and you will see the programs' names in the title box above the core memory. If a program fails to compile, you will see the Messages dialog box with appropriate error messages. In this case, you may run the editor, correct your program, and attempt to compile again.

See Also:
Core War
Editor
Messages
Preferences
Redcode Language

Preferences

From this dialog box, you may change options like the maximum number of cycles MARS will execute, and the maximum number of programs allowed in the core at one time. These options are retained, and will be set the next time you execute Core War for Windows.

Max Cycles - The maximum number of cycles MARS will execute before calling the battle a draw. This number can not be less than 0 nor greater than 1,999,999.

Max Programs - The maximum number of Redcode programs allowed in the core at one time. This number can never be less than 1 nor greater than 20.

Max Processes - The maximum number of processes each program may run. Programs are able to execute more than one process through the SPL instruction. This number may never be less than 1 nor greater than 8000.

Warrior File Extension - The default extension for a Redcode program the Core War for Windows will search for.

"Assume 'COPY'" - The default for Core War for Windows is NOT CHECKED. This option has to do with how MARS implements the MOV, ADD, and SUB instructions. According to common understanding of how a Core War system operates, the only place for data available to a program is in the core. There exist no registers for data to be placed in, and since this is the case, then data moved around must be directly moved from one location to another. Checking this option, will go contradict this thinking, and make MARS implement a register that it can use to temporarily hold data during a MOV, ADD, or SUB instruction. For example,

```
start   mov     1,      <1
finish  dat     #0,     #1
        end     start
```

The MOV instruction resolves to the A operand pointing to the "finish" location, and the B operand also pointing to the "finish" location. Therefore, the effect of this instruction is that it will copy the instruction "finish" on top of itself. What is at issue here, is the value of the data at "finish" after the MOV occurs, because of the pre-decrement at the MOV instruction.

Emulate "COPY" checked
Now, some other Core War simulators would, immediately after resolving the A operand, make a COPY of that data pointed to by the A operand and put it somewhere else (we will call this location COPY and assume it's a register) before resolving the address pointed to by the B operand. In this case, the instruction at "finish" would be copied to this register COPY. Next, the B operand would be resolved, which in this case would decrement the B operand of the "finish" instruction. At this point, the instruction in memory would be "finish dat # 0, #0" but the instruction copied to COPY beforehand would be "finish dat #0, #1". To complete the MOV instruction, MARS would take the data found in COPY and place it to where the B operand dictated: the result is "finish dat #0, #1".

Emulate "COPY" not checked
On the other hand, Core War for Windows, believes that since there are no registers available for the use of data manipulation, it must move data directly from one memory location to another: there can be no COPY register. Also appealing to logic, Core War for Windows resolves both operands (first A then B) completely before the actual execution of the instruction occurs. Operand addresses should be resolved without any consideration to the particular instruction that will be executed. The Core War for Windows scenario would first, resolve the A operand and find it pointing to the "finish" statement. Next, resolve the

B operand, which decrements the B operand of the "finish" statement and find it pointing also to "finish". Last, the MOV instruction itself is executed to end up with "finish dat #0, #0".

The difference between the two methods is that the first immediately stores a copy of what the A operand is pointing to, before the B operand is resolved, while the second one does not utilize any copying. The first method is assuming that there exists some place in memory for this copy to exist, but this goes against the "spirit" of a Core War system's architecture and muddles the understanding of how a Core War's machine would look like under the cover. Core War for Windows opposes the first case on the basis that it makes the Core War machine's architectural design illogical.

See Also:
Core War

Editor

This version of Core War runs Windows' Notepad for you. From Notepad, you may modify and create your own Redcode programs.

See Also:
Core War
Messages
Redcode Language

Messages

This list box contains compile-time messages. Searching through this list box will show you which programs compiled successfully, and which programs had errors. Programs that contain errors will have messages with a brief explanation of the error, as well as the line number were the error occurred.

See Also:
Core War
Battle Setup
Editor
Redcode Language

Registration

Please print out the file called:

register.txt

This will explain how to obtain a registered copy of Core War for Windows from Stage Research.

You may contact Stage Research through our address:

Stage Research
6315 Pearl Road
STE 301
Parma Heights 0H 44130
USA

or, if you have Internet access, you may e-mail me, Brad Rembielak, at

brem@rs6000.baldwinw.edu

On CompuServe,

Brad Rembielak <73612, 413>

Again, for registering, please print out the file "register.txt."

core - an outdated form of computer memory in which metallic rings found at the intersections of a framework of wires were polarized/depolarized to represent on/off states.

ICWS - International Core War Society

MARS - Memory Address Redcode Simulator - Handles the execution of Redcode programs

SPL - "split"

MOV - "move"

ADD - "add"

SUB - "subtract"

EQU - "equate"

END - "end"

DAT - "data"

JMP - "jump"

JMZ - "jump if zero"

JMN - "jump if not zero"

DJN - "decrement; jump if not zero"

CMP - "compare"

SLT - "skip if less than"