From news-rocq.inria.fr!univ-lyon1.fr!swidir.switch.ch!scsing.switch.ch!xlink.net!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse Thu Feb 3 16:44:27 1994
Article: 342 of rec.games.corewar
Newsgroups: rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!swidir.switch.ch!scsing.switch.ch!xlink.net!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse%peruvian.cs.utah.edu@cs.utah.edu (Brant Thomsen)
Subject: The '94 Warrior (Newsletter)
Date: 2 Feb 94 22:30:27 MST
Message-ID: <1994Feb2.223028.20642@hellgate.utah.edu>
Originator: bdthomse@peruvian.cs.utah.edu
Organization: University of Utah CS Dept
Lines: 189

```
__ __| |            ) _ \  |  |    \ \        /              _)
   |   __ \   _ \  / (   | |  |     \ \  \   / _' |  __|  __| |  _ \   __|
   |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
  _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

February 2, 1994 Issue #1

This is the first edition of _The '94 Warrior_, a new bi-weekly (twice a month) newsletter that will discuss the ICWS '94 Draft hills on Stormking -- in much the same way as Paul Kline's _PUSH OFF_ newsletter covers the '88 Standard hill on KOTH. The 1994 redcode (draft) standard offers some exciting advantages and challenges not available in the current (1988) standard. It is my hope that this newsletter will help to increase enthusiasm for the new standard among the newsgroup readers, and will ease its transition. (After all, if you're going to live with it, you might as well enjoy it! ;^)

If you are unfamiliar with the '94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, or if you have any other questions.

First off, let's start with an introduction to the hills:

The ICWS '94 Draft Hill:

```
       Core size:       8000 instrucitons
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

This hill is exactly like the standard for the KOTH hill, except that it also allows you to use instrutions from the '94 standard. Any programs you have written for KOTH will work exactly the same on this hill. In fact, a recent article in the rec.games.corewar newsgroup recommended that everyone automatically submit their program to the '94 draft hill whenever they send it to the standard KOTH hill. This sounds like a good idea to me!

The current ICWS '94 Draft hill:

#	%W/	%L/	%T	Name	Author	Score	Age
1	42/	24/	35	NC II	Wayne Sheppard	160	55
2	48/	42/	10	Rave	Stefan Strack	155	38
3	47/	42/	11	Rave 3	Stefan Strack	153	27
4	45/	36/	19	Fast Food v2.1	Brant D. Thomsen	153	13
5	46/	39/	15	SJ-4	J.Layland	152	4
6	47/	43/	9	Medusa's v6	Mintardjo & Strack	151	60
7	37/	23/	40	Snake	Wayne Sheppard	151	10
8	38/	26/	35	Sphinx v5.1	W. Mintardjo	151	58
9	36/	23/	41	ttti	nandor sieben	149	11
10	45/	40/	15	Sylvester v1.0	Brant D. Thomsen	149	37
11	36/	24/	40	tti94	nandor sieben	148	6
12	45/	42/	13	Fire Storm v1.1	W. Mintardjo	148	61
13	37/	28/	35	JustTakingALookSee	J.Layland	145	54

14	43/	41/	16	Ntttgtstitd	Simon Hovell	145	1
15	42/	41/	18	Beholder's Eye v1.7	W. Mintardjo	143	67
16	40/	40/	19	tiny	J.Layland	141	35
17	42/	46/	12	infraRed	nandor sieben	137	7
18	37/	37/	27	RotLD TNG 2	nandor sieben	136	21
19	40/	54/	6	Testdec 0.8	Jonathan Wolf	127	2
20	31/	38/	31	Plasma	Wayne Sheppard	125	9

I have found the code for many of these programs on SODA, and several others have been posted to the newsgroup. It may be revealing that the program holding strong in the #1 position is written in the '88 standard. (Correct me if I'm wrong, Wayne. :^)

The ICWS '94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The size 55,440 was chosen for the core because this number has so many divisors. (The values 2 - 12 are all possible.) In addition, since the number 55,441 is prime, imp-spirals are not as effective. Because of the increased size of the core, it is still uncertain what the best strategies will be for this hill. (CG-X, by the way, is a group of modified stones.)

One big advantage of this hill is that you are _guaranteed_ to get on it. In other words, WE NEED MORE PROGRAMS!

The current ICWS '94 Experimental hill:

#	%W/	%L/	%T	Name	Author	Score	Age
1	79/	7/	14	CG-X IV	Brant D. Thomsen	250	6
2	79/	17/	4	Rave 3 (55440)	Stefan Strack	241	7
3	76/	21/	3	No Ties Allowed	Wayne Sheppard	231	12
4	69/	21/	10	BS	J.Layland	217	14
5	67/	22/	11	testing testing	Fredrik Ohrstrom	213	2
6	63/	18/	19	BigImps	James Layland	208	15
7	62/	18/	20	Frantic 0.9	A Lee	205	16
8	59/	26/	16	Road Hammer 0.3	Simon Hovell	192	13
9	59/	32/	8	bunker t3	P.Kline	186	3
10	54/	24/	22	Iron Gate	Wayne Sheppard	183	11
11	41/	53/	6	Tiny Ring	J.Layland	128	9
12	36/	64/	0	test 94X	Anonymous	107	5

13-20 No entries

HINTS and HELPS:

Some of the recent discussion in rec.games.corewar has suggested that the new standard will not result in any new forms of programs -- only shorter ones. Whether or not this is true, programming redcode using the new standard is an interesting challenge because there are so many more ways to do these same things. One of the things I would like to do with this newsletter is to share with you some of the hints, quirks, tricks and trivia I have found and discovered that relate to the new standard.

With the '94 standard, it is possible to allow every mode for both the A and B values of any instruction. One of the interesting possibilities this allows is to have an instruction use the "#" addressing mode to refer to itself. I first saw this used by W. Mintardjo in his program Lemming v1.0, although I believe it was possible to use this trick back in the '86 standard. (Again, please correct me if I'm mistaken. The '86 standard was before my time. ;-)

Take a look a the following redcode instructions:

```
        MOV     #100, 1
```

This is the ’94 equivalent of the common imp (MOV 0, 1).
By choosing the first value for this instruction at random,
and using it as an imp-spiral instead of a common imp,
you make it impossible for imp-spiral scanners like Iron Trap
to function effectively.

```
SPL     #732, <-17
```

This instruction functions exactly like SPL 0. It can be used
very effectively in a SPL/JMP bomber, or to make your core
clearing code more robust.

```
JMP     #201, <-10
```

Anyone who has seen my source code to Distance or Fast Food should
recoginize this statement. It is the ’94 equivalent to a "wimp".
If you have a program that generates a lot of processes, you can
put this statement (with one process) somewhere else in the code
as a defense in case your main code stops running. In addition, if
this instruction is the only one you have left, it will stop
any "standard" imp-spirals your opponent still may have.

Since it is possible, using the 94 standard, to manipulate the A value of instructions so effectively, having the value of the A field’s instructions be irrelevant can be a big advantage. For example, in some of my earlier programs I used a DJN.A stream. By using the above technique, my recent programs are much more robust -- and the DJN.A streams much less dangerous.

Another advantage of using this trick is that I have also been able to reduce the size of some of my programs by using this a value to store data -- instead of DAT statements. As any veteran corewar programmer can tell you, one line can make a big difference!

Of course, one disadvantage with this technique is that it can make your code visible to any A-Field scanners. ("Enigma" by Stefan Strack is the only one I’ve seen so far, but ...)

I NEED YOUR HELP:

if you have any comments or questions about the ’94 hills or the ’94 standard that you think might be of general interest, please let me know. Without your assistance, these newsletters could start to get very dry -- and nobody wants that to happen. (Right?)

Also, please submit your programs to the ’94 hills. We’d love to have your best KOTH warrior on the standard ’94 hill, and I’ve found that the best way to learn the ’94 standard is to take an ’88 standard program and try to improve it. Not only will you get the additional challenge and enjoyment of learning the new standard, but you will be better able to correct and critique the latest drafts of the new standard when they arrive.

Good luck, and happy computing!

Brant D. Thomsen, Editor
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Snail mail: 1197 E. 6290 S.
Salt Lake City, UT 84121
U.S.A.

--

Brant D. Thomsen
bdthomse@peruvian.cs.utah.edu
(University of Utah)

The teaching of BASIC in schools
should be considered a criminal act.
- Dijkstra

From news-rocq.inria.fr!univ-lyon1.fr!ghost.dsi.unimi.it!batcomputer!caen!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse Fri Feb 18 14:07:29 1994
Article: 398 of rec.games.corewar
Newsgroups: rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!ghost.dsi.unimi.it!batcomputer!caen!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse%peruvian.cs.utah.edu@cs.utah.edu (Brant Thomsen)
Subject: The ’94 Warrior
Date: 17 Feb 94 12:02:47 MST
Message-ID: <1994Feb17.120248.6541@hellgate.utah.edu>
Originator: bdthomse@peruvian.cs.utah.edu
Organization: University of Utah CS Dept
Lines: 316

```
__ __| |            ) _ \  |  |    \ \        /              _)
   |   __ \   _ \  / (   | |  |     \ \  \   / _‘ |  __|  __| |  _ \   __|
   |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
  _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

February 17, 1994 Issue #2

First off, I would like to thank everyone for the interest and comments I have received so far. There was a large increase in the number of programs submitted to both of the experimental hills on Stormking, as well as to the ICWS hill. (If anyone wants to cover that hill, it’s still available! ;-)

I also wanted to inform the masses about the program I used to create the title for this newsletter. It’s called figlet, and you can get it by anonymous FTP to ftp.isu.edu:/pub/figlet. It is also being discussed in the alt.ascii-art newsgroup.

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

The ICWS ’94 Draft Hill:

```
       Core size:       8000 instrucitons
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill:

#	%W/ %L/ %T	Name	Author	Score	Age
1	42/ 30/ 28	Killer instinct	Anders Ivner	155	5
2	38/ 25/ 37	NC II	Wayne Sheppard	152	60
3	37/ 25/ 38	Sphinx v5.1	W. Mintardjo	148	63
4	35/ 25/ 39	Snake	Wayne Sheppard	145	15
5	43/ 43/ 14	Fire Storm v1.1	W. Mintardjo	143	66
6	32/ 21/ 48	ttti	nandor sieben	143	16
7	42/ 42/ 16	Sylvester v1.0	Brant D. Thomsen	142	42
8	32/ 23/ 45	ttti94	nandor sieben	142	11
9	35/ 28/ 38	JustTakingALookSee	J.Layland	141	59
10	43/ 45/ 12	Rave 3	Stefan Strack	140	32
11	41/ 42/ 17	SJ-4	J.Layland	139	9
12	39/ 39/ 22	Christopher	Steven Morrell	139	4
13	42/ 46/ 11	Rave	Stefan Strack	138	43
14	39/ 41/ 20	Fast Food v2.1	Brant D. Thomsen	138	18
15	40/ 43/ 17	Beholder’s Eye v1.7	W. Mintardjo	138	72
16	40/ 43/ 17	Ntttgtstitd	Simon Hovell	136	6
17	38/ 42/ 20	tiny	J.Layland	134	40
18	39/ 47/ 14	Impurge 94	Fredrik Ohrstrom	132	2
19	40/ 49/ 10	Medusa’s v6	Mintardjo & Strack	132	65

20 31/ 31/ 37 CG IV Brant D. Thomsen 131 1

New programs on the hill include two warriors from the KOTH hill:
"Killer instinct" and "Christopher"; as well as two new '94 warriors:
"Impurge 94" and "CG IV".

Despite the small number of new programs, the hill is really quite different.
Killer instict jumped right to the top of the hill ... probably due to all
the vampires it was able to prey upon. It's also interesting that the
programs at the top of the hill tend to be those written in the '88
standard. Again, this is a good reflection of the intensity of competition
on the KOTH hill.

The ICWS '94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS '94 Experimental hill:

#	%W/	%L/	%T	Name	Author	Score	Age
1	79/	17/	4	Rave 3 (55440)	Stefan Strack	240	8
2	75/	11/	14	CG-X IV	Brant D. Thomsen	239	7
3	74/	23/	3	No Ties Allowed	Wayne Sheppard	226	13
4	68/	22/	10	BS	J.Layland	213	15
5	65/	23/	12	testing testing	Fredrik Ohrstrom	208	3
6	62/	18/	20	BigImps	James Layland	206	16
7	57/	19/	24	Frantic 0.9	A Lee	195	17
8	59/	33/	9	bunker t3	P.Kline	184	4
9	56/	28/	16	Road Hammer 0.3	Simon Hovell	184	14
10	53/	25/	23	Iron Gate	Wayne Sheppard	181	12
11	52/	36/	12	VJX-2	James Ojaste	169	1
12	36/	53/	11	Tiny Ring	J.Layland	119	10
13	31/	69/	0	test 94X	Anonymous	93	6

14-20 No entries

A big thanks to James Ojaste for his entry: "VJX-2".

Once again, this is an ideal hill for beginners -- you are _guaranteed_ to
get on it!

HINTS and HELPS:

Before I begin with this week's hint, I need to thank Stefan Strack for
pointing out a mistake in the last issue of _The 94 Warrior_. A

```
                    MOV     #a, B
```

command is converted to

```
                    MOV.AB  #a, B
```

in the '94 draft, which will only move the "a" value of the instruction --
instead of the entire instruction. You will need to explicitly specify that
the instruction be

```
                    MOV.I   #a, B
```

if you want an imp.

The following is an excerpt from the latest draft of the '94 redcode proposal.
I was actually planning to create a table similar to this for this issue's
hint, but since one already exists, I'll just pass it along instead.

> A.2.1.2 ICWS'88 to ICWS'94 Conversion
> The default modifier for ICWS'88 emulation is determined according
> to the table below.

Opcode	A-mode	B-mode	modifier
DAT	#$@<>	#$@<>	F
MOV,CMP	#	#$@<>	AB

```
                                $@<>    #       B
                                $@<>    $@<>    I
         ADD,SUB,MUL,DIV,MOD    #       #$@<>   AB
                                $@<>    #       B
                                $@<>    $@<>    F
         SLT                    #       #$@<>   AB
                                $@<>    #$@<>   B
         JMP,JMZ,JMN,DJN,SPL    #$@<>   #$@<>   B
         ------------------------------------------------
```

>From now on, I will explicitly be specifying the moderator for all the instructions I supply in this newsletter -- which is probably a good idea when writing ’94 warriors as well. I have found that forcing yourself to explicitly think about what modifier each instruction needs not only eases the development of ’94 warriors, but can also be useful for generating ideas as well.

* * *

Since I’m already on the subject, this issue’s hint will be on imp-rings and spirals.

For those of you that are unfamiliar with exactly what imps, imp-rings, or imp-spirals are, here is a copy of the explanation in the FAQ:

```
        Imp - Program which only uses the MOV instruction.
                    example  MOV 0, 1
                  or
                    example  MOV 0, 2
                             MOV 0, 2

        Imp-Gate - A location in core which is bombed or decremented
           continuously so that an Imp can not pass.  Also used to describe
           the program-code which maintains the gate.
                    example   ...
                              ...
                              SPL 0, <example
                              DAT <example, #0

        Imp-Ring - A minimal Imp-Spiral.
                    d         EQU (coresize+1)/3
                    A         MOV 0,d   ; copy self to B
                    B         MOV 0,d   ; copy self to C
                    C         MOV 0,d   ; copy self to A+1

        Imp-Spiral - An Imp-like program with two or more processes supporting
           each other.  A three-point spiral, with six processes running in
           this sequence:
                    d         EQU (coresize+1)/3
                    A         MOV 0,d   ; copy self to B
                    B         MOV 0,d   ; copy self to C
                    C         MOV 0,d   ; copy self to A+1
                    A+1       MOV 0,d   ; copy self to B+1
                    B+1       MOV 0,d   ; copy self to C+1
                    C+1       MOV 0,d   ; copy self to A+2
```

As Paul Kline has pointed out several times in this newsgroup, imp-rings and imp-spirals for a core of size M can be created by using steps the size of any number that is a divisor of M*N+1, where N is an integer at least as great as one.

Several people, including myself, had been under the misconception that if the number M+1 is prime, than a core of size M will have no true imp-spirals possible. This is, however, not case. For example, even though the number 55441 is prime, it is still possible to generate a 13 point imp-spiral in a 55440 size core because 55440*8+1 = 443521 = 13 * 34117.

For your convenience, I have listed the five smallest possible imp-spiral sizes and steps for some different core sizes.

For a core of size 8000:

Point	Step
3	2667
7	1143
9	889
11	5091
13	3077

For a core of size 55440:

Point	Step
13	34117
17	35873
19	29179
23	38567
29	21029

For a core of size 8192:

Point	Step
3	2731
5	3277
7	3511
9	3641
11	2979

There have been some interesting recent developments in imp-spiral usage. A "standard" imp-spiral (which is the type discussed in the FAQ) can be killed by having your code decrement the B-field of a given location in the core every turn. (Code specifically designed to kill imps, imp-rings, or imp-spirals is also referred to as an imp-gate.) However, with the advent of Cannonade (by Paul Kline), it was revealed that this decrementing is insufficient for stopping _all_ imp-spirals.

Cannonade, for instance, has a "non-standard" 2668 imp-spiral followed closely by a "standard" 2667 imp-spiral. When the 2668 imp-spiral dies on the imp-gate, it leaves behind an instruction to replace the missing instruction that occurs when the 2667 imp-spiral hits the gate. As a result, the 2667 imp-spiral is able to "pass through" the gate. There were also two 2668 imp-spirals posted by Pierre-Etienne Moreau (P.E.M. & E.C.) in early November of 1993 that had some success over-writing gates through the use of varied distances between processes.

It is possible to stop any of the above mentioned imp-spirals by using an imp-gate such as the following:

```
start   SPL.B   0, <-10
        DAT.B   <-11-2668, <-11
```

(Simply remove the ".B" modifiers if you are using the ’88 standard.)

This code will kill any "2668" step imp-spiral, but could be over-run by an imp-spiral with a different step ... such as a "1144" (7 point) imp-spiral. In fact, to the best of my knowledge, it is _impossible_ to use the ’88 (or current ’94) standard to create an imp-gate that will stop all possible imp-spiral combinations.

However, with the current draft of the ’94 standard, there are other options available instead. The imp-gating method I have had the most success with is by using the ">" operand to continuously increment the B-field of a given location in the core. This causes the imp-spiral to move an instruction past the point where it is supposed to, thus leaving a gap in the imp-spiral.

For an example of this technique, take a look at the following imp-gate:

```
start   SPL.B   0, >-20
```

```
                    DAT.B   <-11, >-21
```

The location twenty instructions before "start" is continually being incremented, so any "standard" imp-spirals will be killed here. However, if a gate-crashing imp-spiral gets past this location, it will also need to get past the location ten instructions before "start", which is being decremented every other turn. Since the gate-crashing imp-spiral will tend to be quite fragile when it reaches the second location, the decrementing will usually be enough to finish it off.

While this gate will stop the three special imp-spirals mentioned above, it is still possible to create a series of imp-spirals that can (at least occasionally) over-write it. However, this gate does have an advantage over the ’88 method in its ability to work independently of the step-size of the imp-spiral -- as well as forcing the opponent to use more complex and less dependable imp-spiral combinations in order to over-write it.

On a final note: you can easily create a ’94 imp-gate that will stop any imp-spirals -- if the imp-spiral uses a "MOV.I 0, b" instruction instead of a "MOV.I #a, b" instruction. Simply have an imp-gate that continually increments the A-field at a given location. "Sylvester" uses this technique quite effectively!

Looking to the Future:

The lastest draft of the standard has been on soda.berkeley.edu for a couple of weeks now (as /pub/corewar/incoming/icws94.0202.Z). Take a look at it and let the newsgroup know what you think. There are some interesting additions to the former ’94 draft standard.

Also, please submit your programs to the ’94 hills. We’d love to have your best KOTH warrior on the standard ’94 hill, and I’ve found that the best way to learn the ’94 standard is to take an ’88 standard program and try to improve it. Not only will you get the additional challenge and enjoyment of learning the new standard, but you will be better able to correct and critique the latest drafts of the new standard when they arrive.

If you have any comments or questions about the ’94 hills or the ’94 standard that you think might be of general interest, please let me know. Good luck, and happy computing!

Brant D. Thomsen, Editor Snail mail: 1197 E. 6290 S.
(bdthomse@peruvian.cs.utah.edu) Salt Lake City, UT 84121
University of Utah U.S.A.

--

Brant D. Thomsen
bdthomse@peruvian.cs.utah.edu
(University of Utah)

The teaching of BASIC in schools
should be considered a criminal act.
- Dijkstra

March 2, 1994 Issue #3

The interest in the ’94 standard appears to be increasing dramatically. There has been a lot of discussion in the rec.games.corewar newsgroup recently about the proposed standard, and a large number of entries on the ’94 hills. Keep it up!

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

The ICWS ’94 Draft Hill:

Core size:	8000 instrucitons
Max processes:	8000 per program
Duration:	After 80,000 cycles, a tie is declared.
Max entry length:	100 instructions

The current ICWS ’94 Draft hill:

#	%W/	%L/	%T	Name	Author	Score	Age
1	42/	29/	30	Killer instinct	Anders Ivner	154	6
2	38/	25/	37	NC II	Wayne Sheppard	151	61
3	35/	19/	46	ttti	nandor sieben	151	17
4	37/	25/	38	Sphinx v5.1	W. Mintardjo	150	64
5	35/	20/	45	tti94	nandor sieben	150	12
6	43/	42/	15	Fire Storm v1.1	W. Mintardjo	145	67
7	35/	25/	41	Snake	Wayne Sheppard	145	16
8	35/	27/	37	JustTakingALookSee	J.Layland	143	60
9	40/	42/	18	Sylvester v1.0	Brant D. Thomsen	139	43
10	40/	41/	19	Ntttgtstitd	Simon Hovell	139	7
11	38/	38/	24	Christopher	Steven Morrell	137	5
12	40/	42/	18	SJ-4	J.Layland	137	10
13	41/	46/	14	Rave	Stefan Strack	136	44
14	40/	45/	15	Rave 3	Stefan Strack	136	33
15	38/	40/	22	Fast Food v2.1	Brant D. Thomsen	135	19
16	39/	42/	19	Beholder’s Eye v1.7	W. Mintardjo	135	73
17	37/	41/	23	tiny	J.Layland	133	41
18	39/	46/	15	Impurge 94	Fredrik Ohrstrom	131	3
19	36/	42/	22	Vanity/M	Stefan Strack	130	1
20	30/	32/	37	CG IV	Brant D. Thomsen	129	2

Although there were several attempts at the hill, only one program managed to get on successfully: "Vanity/M", a SPL/JMP bombing B-scanner. It replaced "Medusa’s v6".

There are several scanners on the hill, but they all seem to be stuck in the bottom 10. The vampires on the hill are also starting to take a real beating. The programs that currently tend to be doing the best are mostly imp-spiral based, although "Killer Instinct" is a notable exception to this.

The ICWS ’94 Draft Experimental Hill:

Core size:	55,440 instructions
Max processes:	10,000 per program
Duration:	After 500,000 cycles, a tie is declared.
Max entry length:	200 instructions

The current ICWS ’94 Experimental hill:

#	%W/	%L/	%T	Name	Author	Score	Age
1	60/	12/	29	BigImp	Alex MacAulay	208	8
2	62/	30/	8	Rave 3 (55440)	Stefan Strack	195	19
3	55/	19/	26	Imperfection v1.0 X	Michael Constant	191	4
4	53/	26/	21	CG-X IV	Brant D. Thomsen	180	18
5	58/	38/	4	No Ties Allowed	Wayne Sheppard	178	24
6	48/	21/	31	BigImps	James Layland	174	27
7	55/	38/	7	BS	J.Layland	172	26
8	52/	36/	12	Dagger v6.0 X	Michael Constant	168	7
9	40/	35/	25	Iron Gate	Wayne Sheppard	145	23
10	39/	46/	16	Surprise7	James Ojaste	132	6
11	38/	45/	17	testing testing	Fredrik Ohrstrom	132	14
12	37/	45/	19	Surprise71	James Ojaste	128	3
13	35/	49/	15	Surprise7a	James Ojaste	122	2
14	39/	57/	4	Tracker	Snow	121	1
15	35/	56/	9	bunker t3	P.Kline	113	15
16	18/	31/	51	Frantic 0.9	A Lee	104	28
17	28/	53/	20	Road Hammer 0.3	Simon Hovell	103	25
18	29/	58/	13	VJX-2a	James Ojaste	100	11
19	28/	64/	8	VJX-2	James Ojaste	93	12
20	24/	67/	9	VJX-4	James Ojaste	82	10

It looks like the experimental hill has finally filled up, and Alex MacAulay’s "BigImp" program is dominating it. If I’m not mistaken, "BigImp" is a standard imp-spiral/stone combination. It looks like all your ’94 programs will still need to deal with Imps!

HINTS and HELPS:

This issue’s hint is brought to you by Stefan Strack. Stefan is the driving force behind the creation of a ’94 ICWS Standard, as well as being heavily involved in the pMARS project. It was his tournament last fall, in fact, that got me interested in the current ’94 standard.

Rejuvenating a Classic

In the last issue of _The_’94_Warrior_, Brant mentioned that the best way to get familiar with ICWS’94 is to optimize an ICWS’88 warrior. This is what I am going to do in the following. I am going to start with a classic SPL-carpet bombing CMP-scanner, Agony 3.1, and turn it step by step into its shorter, more efficient ’94 counterpart Rave 3.

To start with, here is the code for Agony 3.1:

```
;redcode
;name Agony 3.1
;author Stefan Strack
;strategy Small-interval CMP scanner that bombs with a SPL 0 carpet.

CDIST   equ 12
IVAL    equ 42
FIRST   equ scan+OFFSET+IVAL
OFFSET  equ (2*IVAL)
DJNOFF  equ -431

scan    sub incr,comp
comp    cmp FIRST-CDIST,FIRST
        slt #incr-comp+CDIST+(bptr-comp)+1,comp
        djn scan,<FIRST+DJNOFF
        mov #CDIST+(bptr-comp)+1,count
        mov comp,bptr
bptr    dat #0
split   mov bomb,<bptr
count   djn split,#0
        jmn scan,scan
bomb    spl 0
        mov incr,<count
```

```
incr    dat <0-IVAL,<0-IVAL

        end comp
```

The first four instructions scan for enemy code, comparing 12 instructions apart. If a difference was found, a bomb carpet is laid, starting 4 addresses above the B-address and ending with the A-address. This class of CMP-scanner wastes a lot of time bombing decremented addresses or decoy code, and therefore loses a lot against stone-type bombers. You can decrease the CMP-distance of 12 to cut down on the time spent laying the carpet, but the scan pattern becomes quickly sub-optimal, especially against larger warriors. Mintardjo was able to shorten the bomb carpet without shortening the CMP-distance by decrementing the ’cmp’ instruction directly. His warrior "Medusa’s" is however one line longer than Agony, because the pre-decrement misses bombing the B-address, which therefore has to be bombed with a separate instruction:

```
;redcode
;name Medusa’s v2.1
;author Mintardjo & Stefan

CDIST   equ 12
IVAL    equ 28
FIRST   equ scan-OFFSET+IVAL
OFFSET  equ 5324

scan    sub data,comp
comp    cmp FIRST-CDIST,FIRST
        slt #data-comp+CDIST+1,comp
        djn scan,<FIRST+163
        mov #CDIST-1,count
split   mov bomb,<comp
count   djn split,#0
        add #CDIST-1,comp
        jmz scan,scan-1
bomb    spl 0, <0-IVAL+1
incr    mov data, <bomb-1
        jmp incr, <0-IVAL-1
data    DAT <0-IVAL, <0-IVAL

        end comp
```

Here is where ICWS’94’s _post_-increment comes into play: we can now use the ’cmp’ instruction as a pointer, and still bomb everything including A- and B-address with a 2-line loop. Since the B-address is now incremented, it must initially be lower than the A-address. So we don’t bomb our one code we now need to test the A-address with the slt.A instruction. Adding the .F modifier to the DJN makes it both safer (we fall thru only if both A- and B-numbers are 1) and more effective. Finally, we dispose of the last DAT instruction by combining the scan increments with the SPL instruction. As explained in the first issue, SPL #n is executed like SPL 0. Here the full code:

```
;redcode-94
;name Rave 3
;author Stefan Strack
;strategy Carpet-bombing scanner based on Agony and Medusa’s

CDIST   equ 12
IVAL    equ 42
FIRST   equ scan+OFFSET+IVAL
OFFSET  equ (2*IVAL)
DJNOFF  equ -431
BOMBLEN equ CDIST+2

        org    comp

scan    sub.f  incr,comp
comp    cmp.i  FIRST,FIRST-CDIST        ;larger number is A
```

```
        slt.a  #incr+1-comp+BOMBLEN,comp  ;compare to A-number
        djn.f  scan,<FIRST+DJNOFF       ;decrement A- and B-number
        mov.ab #BOMBLEN,count
split   mov.i  incr,>comp               ;post-increment
count   djn.b  split,#0
        sub.ab #BOMBLEN,comp
        jmz.a  scan,scan-1
incr    spl.b  #-IVAL,<-IVAL
        mov.i  1,<count
```

-Stefan (stst@vuse.vanderbilt.edu)

Looking to the Future:

The lastest draft of the standard has been on soda.berkeley.edu for about a month now (as /pub/corewar/incoming/icws94.0202.Z). Take a look at it and let the newsgroup know what you think. There are some interesting additions to the former ’94 draft standard, and there has been some intense discussion in the newsgroup lately about parsing code and adding new instructions. Now is the time to voice your opinions!

Also, please submit your programs to the ’94 hills. We’d love to have your best KOTH warrior on the standard ’94 hill, even if it is still ’88 compliant.

If you have any comments or questions about the ’94 hills or the ’94 standard that you think might be of general interest, please let me know. Good luck, and happy computing!

Brant D. Thomsen, Editor Snail mail: 1197 E. 6290 S.
(bdthomse@peruvian.cs.utah.edu) Salt Lake City, UT 84121
University of Utah U.S.A.

From news-rocq.inria.fr!univ-lyon1.fr!ghost.dsi.unimi.it!batcomputer!hookup!news.kei.com!eff!news.umbc.edu!europa.eng.gtefsd.com!library.ucla.edu!ihnp4.ucsd.edu!dog.ee.lbl.gov!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse Thu Mar 17 14:43:35 1994
Article: 510 of rec.games.corewar
Newsgroups: rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!ghost.dsi.unimi.it!batcomputer!hookup!news.kei.com!eff!news.umbc.edu!europa.eng.gtefsd.com!library.ucla.edu!ihnp4.ucsd.edu!dog.ee.lbl.gov!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse%peruvian.cs.utah.edu@cs.utah.edu (Brant Thomsen)
Subject: The ’94 Warrior
Date: 16 Mar 94 17:53:25 MST
Message-ID: <1994Mar16.175325.28347@hellgate.utah.edu>
Originator: bdthomse@peruvian.cs.utah.edu
Organization: University of Utah CS Dept
Lines: 185

```
__ __| |            ) _ \  |  |    \ \        /              _)
   |   __ \   _ \  / (   | |  |     \ \  \   / _‘ |  __|  __| |  _ \   __|
   |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
  _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

March 16, 1994 Issue #4

This newletter covers the current status of the ICWS ’94 Draft hills, and also attempts to keep up with the latest ideas in how the new standard will affect corewars in general. I hope you enjoy it!

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

The ICWS ’94 Draft Hill:

```
       Core size:       8000 instrucitons
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill:

#	%W/ %L/ %T	Name	Author	Score	Age
1	46/ 31/ 23	Killer instinct	Anders Ivner	162	9
2	42/ 23/ 34	NC II	Wayne Sheppard	162	64
3	41/ 25/ 35	Sphinx v5.1	W. Mintardjo	157	67
4	37/ 22/ 42	ttti94	nandor sieben	152	15
5	46/ 42/ 12	Fire Storm v1.1	W. Mintardjo	151	70
6	36/ 22/ 43	ttti	nandor sieben	150	20
7	38/ 26/ 36	JustTakingALookSee	J.Layland	150	63
8	43/ 41/ 16	Sylvester v1.0	Brant D. Thomsen	145	46
9	42/ 40/ 17	Request v2.0	Brant D. Thomsen	144	2
10	35/ 26/ 39	Snake	Wayne Sheppard	144	19
11	41/ 39/ 20	Christopher	Steven Morrell	143	8
12	41/ 40/ 19	Fast Food v2.1	Brant D. Thomsen	142	22
13	41/ 42/ 16	Ntttgtstitd	Simon Hovell	141	10
14	43/ 45/ 13	Rave 4	Stefan Strack	141	3
15	41/ 41/ 18	Beholder’s Eye v1.7	W. Mintardjo	140	76
16	43/ 46/ 11	Rave	Stefan Strack	140	47
17	41/ 43/ 16	SJ-4	J.Layland	140	13
18	40/ 40/ 21	tiny	J.Layland	140	44
19	42/ 46/ 12	Testing an Idea	Brant D. Thomsen	138	1
20	42/ 48/ 11	Impurge 94	Fredrik Ohrstrom	136	6

The ’94 draft hill seems to be getting quite vicious lately. There were several attempts at the hill, but not very many made it.

(The program "Testing an Idea" is discussed in this issue's hint section.)

---

th ICWS '94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS '94 Experimental hill:

#	%W/ %L/ %T	Name	Author	Score	Age
1	53/ 12/ 35	BigImp	Alex MacAulay	194	25
2	52/ 17/ 30	Imperfection v1.0 X	Michael Constant	187	21
3	60/ 34/ 6	Rave 3 (55440)	Stefan Strack	185	36
4	42/ 13/ 45	Skimp 127	Jay Han	171	6
5	54/ 42/ 4	No Ties Allowed	Wayne Sheppard	166	41
6	51/ 38/ 10	Dagger v6.0 X	Michael Constant	164	24
7	41/ 17/ 42	BigImps	James Layland	164	44
8	49/ 39/ 12	Sunburst 33	Jay Han	159	14
9	44/ 31/ 25	CG-X IV	Brant D. Thomsen	157	35
10	49/ 42/ 10	BS	J.Layland	156	43
11	41/ 34/ 25	Iron Gate	Wayne Sheppard	148	40
12	35/ 26/ 39	FatImp	Jay Han	144	1
13	40/ 50/ 10	Silly 2	James Ojaste	130	7
14	41/ 54/ 5	Silly	James Ojaste	129	9
15	33/ 46/ 21	testing testing	Fredrik Ohrstrom	121	31
16	32/ 46/ 22	Crimp	Jay Han	118	2
17	34/ 53/ 13	VJX-2a	James Ojaste	115	28
18	28/ 42/ 31	Bloom	Jay Han	114	12
19	30/ 50/ 20	Surprise7b	James Ojaste	109	10
20	32/ 59/ 8	VJX-2	James Ojaste	105	29

WOW! Due largely to the efforts of Jay Han and James Ojaste, the '94 experimental hill has really changed over the last couple of weeks. So far, it looks like imp-spiral/stone combinations and scanners are dominating the hill. (Has anyone out there tried a paper or vampiric program? I'd be curious to know how it does.)

HINTS and HELPS:

One of the most irritating things in corewars is handling those programs that refuse to die. You know what I mean: your scanner/bomber/vampire has stunned the opponent; and then, suddenly, the enemy comes to life in the middle of your core-clear and finishes you off.

The '88 solution to this problem has been the two-pass core clear. (I have no idea who was the first person to use this technique, but it is now quite common.) During the first pass of the core-clearing loop, the code fills the core with "SPL 0" statements; during the second pass, the code overwrites the core with "DAT" statements. The "SPL 0" instructions greatly increase the odds that the program is no longer a threat, and the "DAT" insructions are to kill your opponent off altogether. This method is very efficient, but it has some disadvantages as well: multiple-pass core-clears can be hard to set up, and they are often disabled by fast bombers.

A similar method that can be used with the '94 standard is to use your core-clearing routine to "confuse" the core instead. Take a look at the following code:

```
        clear   spl.B   #0, <-20        ; Equivalent to: SPL.B 0, <-20
                mov.I   2, <-11         ; Overwrite the core
        confuse djn.F   -1, >1          ; Confuse the core
                dat.F   <-21, #10       ; Give a pointer to work with
```

The first two lines are the '94 equivalent of a standard core-clear. The "SPL 0" line generates processes, which the "MOV" uses to fill the core with "DAT" instructions.

The last two lines are used to "confuse" the core. The "DJN" instruction will decrement both the A and B-field values of the instructions as it proceeds forward through the core. This decrementing is usually enough to keep any program it meets from working correctly, so the standard part of the core-clear (which is working its way backwards through the core) will have no problem completely killing the opponent when it gets to it. Since the "MOV" instruction in the core-clear gets the processes created by the "SPL 0" before the "DJN" instruction does, it will always complete first -- so the confusion will never reach the core-clear itself.

This technique has several advantages over a two-pass core-clear. In addition to being at least twice as fast, it is also easy to implement and fairly rugged. In fact, the "confuse" line can be bombed by a "DAT" statement and the core-clear will still work.

There are several attributes of using this core-clearing method that cause it to work well against imp-spirals as well. The multiple processes running during the clear will give your program a tie if the core-clear is over-run before it has completed, and the imp-gate at the end will (usually) stop imp-spirals if any remain when the core-clear is completed.

An important thing to notice is that the location pointed to by the "confuse" line has a positive value. This is because the "confusion" will not work correctly if this value is zero. (The decrementing and incrementing cancel each other out.)

As a final test to see how well this idea works, I added a "confusion" core-clear to the program "Rave 3" (from the last issue) and submitted it to the hill to see how it does. It is under the name "Testing an Idea". It looks as if there are currently so many scanners on the hill that the adding of the two extra lines turned into a disadvantage. (Perhaps I should have saved this hint for a later issue instead! :-)

Looking to the Future:

The lastest draft of the standard has been on soda.berkeley.edu for over a month now (as /pub/corewar/documents/icws94.0202.Z). Take a look at it and let the newsgroup know what you think. There are some interesting additions to the former ’94 draft standard, so it’s worth your effort to get a copy and study it.

Also, please submit your programs to the ’94 hills. We’d love to have your best KOTH warrior on the standard ’94 hill, even if it is still ’88 compliant.

If you have any comments or questions about the ’94 hills or the ’94 standard that you think might be of general interest, please let me know.

Good luck, and happy computing!

Brant D. Thomsen, Editor
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Snail mail: 1197 East 6290 South
Salt Lake City, UT 84121
U.S.A.

--

Brant D. Thomsen
(bdthomse@peruvian.cs.utah.edu)
University of Utah

The teaching of BASIC in schools
should be considered a criminal act.
- Dijkstra

From news-rocq.inria.fr!univ-lyon1.fr!ghost.dsi.unimi.it!batcomputer!hookup!news.kei.com!eff!news.umbc.edu!europa.eng.gtefsd.com!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse Tue Apr 5 14:43:10 1994
Article: 576 of rec.games.corewar
Newsgroups: rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!ghost.dsi.unimi.it!batcomputer!hookup!news.kei.com!eff!news.umbc.edu!europa.eng.gtefsd.com!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse%peruvian.cs.utah.edu@cs.utah.edu (Brant Thomsen)
Subject: The ’94 Warrior
Date: 2 Apr 94 13:48:59 MST
Message-ID: <1994Apr2.134900.8347@hellgate.utah.edu>
Originator: bdthomse@peruvian.cs.utah.edu
Organization: University of Utah CS Dept
Lines: 423

```
__ __| |            ) _ \  |  |    \ \        /              _)
   |   __ \   _ \  / (   | |  |     \ \  \   / _‘ |  __|  __| |  _ \   __|
   |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
  _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

April 2, 1994 Issue #5

This newletter covers the current status of the ICWS ’94 Draft hills, and also attempts to keep up with the latest ideas in how the new standard will affect corewars in general. I hope you enjoy it!

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

The ICWS ’94 Draft Hill:

```
       Core size:       8000 instrucitons
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill:

#	%W/ %L/ %T	Name	Author	Score	Age
1	42/ 31/ 27	Killer instinct	Anders Ivner	154	10
2	38/ 24/ 38	NC II	Wayne Sheppard	153	65
3	38/ 25/ 37	Sphinx v5.1	W. Mintardjo	150	68
4	32/ 22/ 46	ttti94	nandor sieben	143	16
5	43/ 44/ 13	Fire Storm v1.1	W. Mintardjo	142	71
6	34/ 27/ 40	JustTakingALookSee	J.Layland	141	64
7	31/ 22/ 47	ttti	nandor sieben	140	21
8	40/ 42/ 17	Sylvester v1.0	Brant D. Thomsen	139	47
9	32/ 25/ 43	Twimpede/88	Jay Han	139	1
10	39/ 43/ 18	Request v2.0	Brant D. Thomsen	136	3
11	39/ 43/ 17	Ntttgtstitd	Simon Hovell	136	11
12	31/ 27/ 42	Snake	Wayne Sheppard	134	20
13	38/ 42/ 21	Fast Food v2.1	Brant D. Thomsen	134	23
14	38/ 43/ 19	Beholder’s Eye v1.7	W. Mintardjo	132	77
15	40/ 47/ 13	Rave 4	Stefan Strack	132	4
16	40/ 48/ 12	Rave	Stefan Strack	132	48
17	38/ 44/ 18	SJ-4	J.Layland	131	14
18	37/ 42/ 21	Christopher	Steven Morrell	131	9
19	36/ 42/ 22	tiny	J.Layland	130	45
20	39/ 49/ 13	Testing an Idea	Brant D. Thomsen	128	2
21	39/ 50/ 12	Impurge 94	Fredrik Ohrstrom	127	7

It looks like the ’94 draft hill has really slowed down. I would speculate that this is a result of the greatly increased amount of attention being given to the ’94 experimental hill. The one new program since the last issue to make it onto the hill is Jay Han’s "Twimpede/88". Congratulations, Jay, on this unique honor!

The ICWS ’94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS ’94 Experimental hill:

#	%W/	%L/	%T	Name	Author	Score	Age
1	48/	28/	24	Request-55440	Brant D. Thomsen	169	35
2	33/	11/	56	Variation A-1.a	Jay Han	156	4
3	46/	40/	14	Kill Imps!!!	Steven Morrell	152	22
4	35/	24/	41	CG-X IV	Brant D. Thomsen	145	86
5	31/	17/	52	Lucky 13	Stefan Strack	144	1
6	28/	15/	57	Twimpede	Jay Han	141	2
7	44/	47/	9	Rave B4	Stefan Strack	140	44
8	21/	10/	69	Imperfection v2.3	Michael Constant	132	29
9	36/	40/	24	Vanity IIx	Stefan Strack	132	26
10	23/	19/	58	Bimper	Wayne Sheppard	127	10
11	29/	34/	36	Night Crawler	Wayne Sheppard	124	45
12	37/	51/	12	The Count	Jay Han	124	25
13	35/	50/	16	Dagger v6.0 X	Michael Constant	119	75
14	21/	22/	57	BigImps	James Layland	119	95
15	18/	20/	62	BigImp	Alex MacAulay	117	76
16	18/	22/	60	Skimp 127	Jay Han	113	57
17	27/	47/	26	Sunburst 33	Jay Han	107	65
18	10/	21/	68	bimp.c test	Wayne Sheppard	99	12
19	11/	22/	67	bimp.c test	Wayne Sheppard	99	20
20	5/	26/	70	Impale v2a	James Ojaste	84	3
21	1/	1/	3	Lucky 13	Stefan Strack	5	7

Before I even attempt to summarize what has been happening on the ’94 experimental hill, I think it would be helpful to reprint the hill as it was during the last issue. (17 Days previously.)

The current ICWS ’94 Experimental hill:

#	%W/	%L/	%T	Name	Author	Score	Age
1	53/	12/	35	BigImp	Alex MacAulay	194	25
2	52/	17/	30	Imperfection v1.0 X	Michael Constant	187	21
3	60/	34/	6	Rave 3 (55440)	Stefan Strack	185	36
4	42/	13/	45	Skimp 127	Jay Han	171	6
5	54/	42/	4	No Ties Allowed	Wayne Sheppard	166	41
6	51/	38/	10	Dagger v6.0 X	Michael Constant	164	24
7	41/	17/	42	BigImps	James Layland	164	44
8	49/	39/	12	Sunburst 33	Jay Han	159	14
9	44/	31/	25	CG-X IV	Brant D. Thomsen	157	35
10	49/	42/	10	BS	J.Layland	156	43
11	41/	34/	25	Iron Gate	Wayne Sheppard	148	40
12	35/	26/	39	FatImp	Jay Han	144	1
13	40/	50/	10	Silly 2	James Ojaste	130	7
14	41/	54/	5	Silly	James Ojaste	129	9
15	33/	46/	21	testing testing	Fredrik Ohrstrom	121	31
16	32/	46/	22	Crimp	Jay Han	118	2
17	34/	53/	13	VJX-2a	James Ojaste	115	28
18	28/	42/	31	Bloom	Jay Han	114	12
19	30/	50/	20	Surprise7b	James Ojaste	109	10
20	32/	59/	8	VJX-2	James Ojaste	105	29

Since the last issue of _The_’94_Warrior_, 51 new programs have made it onto the hill -- this is 3 new programs a day. Of the first 12 programs on the hill, only one was on the hill during the last issue. More programs have been submitted in this period than in the entire life of the hill previously.

(Are you getting the point yet?!)

Perhaps the most surprising thing about the current standings on the hill is the amount of diversity there is between the warriors. Scanners, stones, and imp-spiral combinations are all well represented. In fact, the warrior currently at the top of the hill is a vampire. (Certainly not what I would have predicted.) It looks as if this is the hill where the excitement is!

HINTS and HELPS:

To assist program development on the '94 experimental hill, I have used Michael Constant's "Optima" program to calculate the twenty most-efficient step-sizes for mods 1 to 10. (A mod 1 step-size will eventually cover every location in the core, a mod 2 step-size will eventually cover one-half of the locations in the core, etc.)

Notice that there is a bug in the version of "Optima" released last year. A quick fix is to make the variable "i" in the function "opt" long instead of short. An easier fix will be to wait until Michael releases the next version. This bug only makes itself known when using a large coresize and a large step size. In fact, it was due to this bug that I ended up using a step-size of 9719 in the 55440 coresize version of Request -- it was the best mod 1 step the program could find.

[The source code for Request-55440 will probably be included in the next issue. I figured the following tables would be more useful given the current number of programs being submitted to the '94 experimental hill. Also, there are a couple of points about program development I am finishing up for a future hint, and "Request" will be good example to use for them.]

Results of running "Optima" on a 55440 size core:

Mod 1:

16211	6.183968
23269	6.183968
15481	6.132795
16921	6.132795
22889	6.113332
24151	6.113332
16369	6.065658
21649	6.065658
12689	6.056693
15611	6.056693
20509	6.056693
21151	6.056693
20617	6.052941
22873	6.052941
12503	6.048468
16393	6.048468
16817	6.045978
21247	6.045978
9719	6.028265
23399	6.028265

Mod 2:

16238	11.586204
20498	11.477182
21502	11.477182
22766	11.439446
23266	11.439446
15458	11.393845
16462	11.393845
12718	11.343555
23278	11.343555
11974	11.340597
20474	11.340597

15314	11.325805
22706	11.325805
16894	11.307190
23234	11.307190
20546	11.242397
21074	11.242397
20458	11.233378
24422	11.233378
20558	11.232151

Mod 3:

24357	16.517452
21381	16.368905
22971	16.368905
14907	16.340657
20373	16.340657
21057	16.330429
22503	16.330429
14691	16.305265
15261	16.305265
19461	16.144705
20571	16.144705
24483	16.085124
16473	16.023107
20607	16.023107
16827	16.003463
22773	16.003463
9687	16.002976
23007	16.002976
19911	15.997132
23439	15.997132

Mod 4:

22964	22.001587
20996	21.523631
24356	21.523631
17228	21.204705
24148	21.204705
22756	21.191717
23476	21.191717
15548	21.175265
22892	21.175265
20068	21.112923
23228	21.112923
24196	20.981312
23404	20.955336
20476	20.869038
23924	20.869038
16508	20.834981
23428	20.834981
21508	20.811025
22508	20.811025
12988	20.771196

Mod 5:

21445	26.052133
23435	26.052133
9865	25.878055
21215	25.878055
24175	25.794624
24905	25.794624
12115	25.712546
21485	25.712546

16265	25.675566
22735	25.675566
11785	25.657527
16465	25.657527
20315	25.560566
24875	25.560566
16925	25.451881
24125	25.451881
9965	25.415351
21475	25.415351
9925	25.400469
15445	25.400469

Mod 6:

16134	31.110510
21174	31.110510
15234	30.829960
21486	30.829960
21246	30.807230
20994	30.786449
16374	30.408486
14946	30.113649
19986	30.113649
19938	29.889598
21522	29.889598
17142	29.831800
24198	29.831800
16194	29.815564
24834	29.815564
20298	29.745427
24762	29.745427
20058	29.643468
21642	29.643468
17382	29.622686

Mod 7:

16373	35.304963
21427	35.304963
21007	35.243086
24367	35.243086
20041	35.102538
21049	35.102538
24157	34.939891
24997	34.939891
21623	34.906301
24983	34.906301
23429	34.649956
25459	34.649956
17129	34.616366
21161	34.616366
15043	34.480237
21077	34.480237
20293	34.479353
22547	34.479353
14651	34.413057
24899	34.413057

Mod 8:

12104	39.494299
24184	39.494299
20344	39.289941
21256	39.289941
15272	39.236831
21752	39.236831
16232	39.235676

22952	39.235676
19864	38.698802
20296	38.698802
20456	38.686102
25064	38.686102
13144	38.587964
23384	38.587964
24424	38.525617
25096	38.525617
16024	38.514071
16904	38.514071
23416	38.514071
24296	38.514071

Mod 9:

12087	44.316123
21177	44.316123
14967	43.623478
24903	43.623478
21141	43.594252
21501	43.594252
20961	43.468583
21519	43.468583
16983	43.337068
22887	43.337068
11997	43.281539
21627	43.281539
14841	43.227472
22761	43.227472
19647	43.107647
21663	43.107647
16749	42.900146
24309	42.900146
16461	42.870921
24219	42.870921

Mod 10:

19990	48.630705
21050	48.630705
21470	48.029948
22930	48.029948
16210	47.889230
24830	47.889230
16910	47.670936
24130	47.670936
13150	47.441819
23230	47.441819
9970	47.376872
22910	47.376872
15350	47.310121
22790	47.310121
12490	47.265019
19930	47.265019
21430	47.061158
23930	47.061158
16930	47.021469
19330	46.691322

Looking to the Future:

The lastest draft of the standard has been on soda.berkeley.edu for a couple of months now (as /pub/corewar/documents/icws94.0202.Z). Take a look at it and let the newsgroup know what you think. There are some interesting additions to the former ’94 draft standard, so it’s worth your effort

to get a copy and study it.

Also, please submit your programs to the ’94 hills. We’d love to have your best KOTH warrior on the standard ’94 hill, even if it is still ’88 compliant.

If you have any comments or questions about the ’94 hills or the ’94 standard that you think might be of general interest, please let me know.

Good luck, and happy computing!

Brant D. Thomsen, Editor
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Snail mail: 1197 East 6290 South
Salt Lake City, UT 84121
U.S.A.

--

Brant D. Thomsen
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Man will occasionally stumble over the truth,
but most times he will pick himself up
and carry on. - Winston Churchill

From news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse Wed Apr 20 15:07:27 1994
Article: 665 of rec.games.corewar
Newsgroups: rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse%peruvian.cs.utah.edu@cs.utah.edu (Brant Thomsen)
Subject: The ’94 Warrior
Date: 18 Apr 94 23:46:26 MDT
Message-ID: <1994Apr18.234626.6872@hellgate.utah.edu>
Originator: bdthomse@peruvian.cs.utah.edu
Organization: University of Utah CS Dept
Lines: 400

```
__ __| |            ) _ \  |  |    \ \        /              _)
   |   __ \   _ \  / (   | |  |     \ \  \   / _‘ |  __|  __| |  _ \   __|
   |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
  _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

April 18, 1994 Issue #6

This newletter covers the current status of the ICWS ’94 Draft hills, and also attempts to keep up with the latest ideas in how the new standard will affect corewars in general. I hope you enjoy it!

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

CHANGES and CORRECTIONS:

There have been several changes since the last issue of _The_’94_Warrior_ came off the electronic presses. With luck, I’ll manage to get most of them.

The first big event is the addition of two (or three) new opcodes for version 0.5 of pMARS. It was recently downloaded to soda.berkeley.edu (in the pub/corewar/incoming directory). Even if you don’t think you will ever need to use SEQ, SNE, or NOP, there are a couple of bug fixes that make it worth the effort to upgrade.

The another big event is the creation of a new hill server, "pizza". "Pizza" is operated by Thomas Davies, and he’s doing a great job at it! Most of the programs residing at the "Intel KotH" and "stormking" have already managed to migrate over, and there are several new programs on "pizza" as well. The "pizza" hills eliminate the major complaint users had with the "stormking" hills: you no longer need to wait a day or two to get back the results of a warrior submission. (The process of submitting a warrior to a hill on "pizza" is covered in the latest FAQ for the rec.games.corewar newsgroup. Send me mail if you didn’t receive a copy of it.)

Along with the creation of the "pizza" server came the passing of the Bill Shubert’s hill at Intel. Thanks, Bill, for the many years you have used your resources to allow the rest of us to hone our corewars skills. The game has improved greatly because of your efforts!

You’ll also want to catch the first chapter of Steven Morrell’s "MY FIRST COREWAR BOOK". It’s on soda.berkeley.edu in the pub/corewar/incoming directory as "chapter1.Z". Even if Steven didn’t mention my name in it and he attended some school other than the University of Utah, I would still recommend it. It’s that good!

For those of you developing warriors in large core sizes, the upgraded version

of Optima I mentioned in the last issue is now available on soda. If you are experimenting with different core sizes (like the small core on "pizza", this program is a necessity.)

And, for those of you with a competitive streak, I would highly encourage you to enter Stefan Strack's upcoming corewar tournament. In case you missed the announcement, here it is again:

> In the meantime, I am going to hold a double-elimination tournament with weekly rounds, just like the summer and fall tourneys of last year (soda.berkeley.edu:/pub/corewar/redcode/st93warriors.zip includes a rapport of this kind of tourney).
>
> Rules will alternate between '94 (coresize=8000) and '94x (coresize=55440) hill specs. If there's demand, I'll also throw in a pure '88 round. If there are enough participants, I'll divide them up into an A- and B-league (verterans and beginners).
>
> The submission deadline for round 1 is Friday, April 22nd. Rules are '94x (big) hill specs, and you may use the new opcodes. Your opponent in round 1 will be chosen randomly.
>
> And as though the fame was not enough incentive already, the tournament winner will receive a free ICWS membership and subscription to the newsletter for one year.
>
> -Stefan (stst@vuse.vanderbilt.edu)

Just in case those prizes are not good enough for you, I will also throw in a free one-year subscription to _The_'94_Warrior_ for the winner. Seriously though, Stefan always does a great job with his tournaments, and I heartily recommend you give this one a try.

The ICWS '94 Draft Hill:

```
      Core size:       8000 instrucitons
   Max processes:      8000 per program
        Duration:      After 80,000 cycles, a tie is declared.
Max entry length:      100 instructions
```

The current ICWS '94 Draft hill:

#	%W/ %L/ %T	Name	Author	Score	Age
1	45/ 20/ 34	Der Zweite Blitzkrieg - 9	Mike Nonemacher	170	13
2	47/ 31/ 21	Sauron v1.7	Michael Constant	163	3
3	42/ 30/ 28	Killer instinct	Anders Ivner	154	64
4	46/ 41/ 13	Dragon Spear	c w blue	151	63
5	36/ 22/ 42	Lucky 3	Stefan Strack	150	23
6	43/ 36/ 21	Request v2.0	Brant D. Thomsen	149	61
7	45/ 43/ 12	Iron Gate 1.5	Wayne Sheppard	148	7
8	31/ 17/ 53	Blue Funk	Steven Morrell	145	30
9	33/ 21/ 46	Twimpede+/8000-E.2c	Jay Han	144	17
10	32/ 21/ 47	Imperfection v3.1	Michael Constant	142	15
11	34/ 26/ 40	NC 94	Wayne Sheppard	142	43
12	31/ 24/ 44	Splash	Jay Han	139	1
13	38/ 38/ 25	NTA 94	Wayne Sheppard	137	51
14	40/ 43/ 17	Test	Stefan Strack	137	26
15	40/ 42/ 18	Sylvester v1.0	Brant D. Thomsen	137	60
16	38/ 41/ 22	Fast Food v2.1	Brant D. Thomsen	134	58
17	34/ 34/ 32	Match Stick	c w blue	133	41
18	31/ 32/ 37	Little Flea v1.2	Michael Constant	129	5
19	38/ 49/ 13	Dagger v6.0	Michael Constant	126	67
20	32/ 45/ 23	Assassin v1.0	Michael Constant	119	20

For this issue, I was planning to post the results of both the "stormking" and "pizza" '94 hills. However, as I haven't gotten back the results of the queries I sent to "stormking" almost four days ago, I decided to simply make a clean switch to pizza this month. Send any complaints to the editorial board.

I believe most of the programs that are on the ’94 draft hill on "stormking" have made it over to "pizza". Next issue, after things have settled down a little bit on the new hills, I will attempt to analyze what is happening.

The ICWS ’94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS ’94 Experimental (Big) hill:

#	%W/ %L/ %T	Name	Author	Score	Age
1	49/ 15/ 36	Variation G-1	Jay Han	182	1
2	47/ 15/ 38	Variation E-2.c	Jay Han	178	13
3	44/ 16/ 39	Splash 1	Jay Han	172	2
4	45/ 22/ 32	Lucky 13	Stefan Strack	168	43
5	49/ 32/ 20	Request-55440	Brant D. Thomsen	165	37
6	35/ 18/ 47	Imperfection v2.4	Michael Constant	153	8
7	47/ 41/ 12	Rave B4	Stefan Strack	153	29
8	45/ 42/ 13	Virus	Jay Han	148	3
9	44/ 43/ 13	bigproba	nandor sieben	145	41
10	42/ 46/ 12	Scanalyzer-W	Jay Han	138	4
11	42/ 46/ 13	Scanalyzer-V.3b	Jay Han	138	11
12	35/ 33/ 32	Big Flea v1.1	Michael Constant	137	7
13	38/ 46/ 15	amoeba v82	Richard van der Brug	131	18
14	38/ 47/ 15	Dagger v8.0	Michael Constant	130	24
15	38/ 46/ 16	Sunburst 33	Jay Han	130	31
16	30/ 31/ 38	Nova-A.1b	Jay	129	12
17	29/ 32/ 39	Skimp 127	Jay Han	127	30
18	34/ 42/ 23	IceCube 1.4	Richard van der Brug	126	17
19	32/ 40/ 28	Veeble Jr.	T. H. Davies	125	27
20	36/ 51/ 13	Kill Imps!!!	Steven Morrell	121	26

The comments for the other hill apply here as well.

HINTS and HELPS:

For today’s hint, I will be using the source code to "Request v2.0" for the 55440 size ’94 hill. "Request" is very similar to my earlier "Distance" vampires (I know, I’m stuck in a rut ...) with the exception that the fangs use indirect addressing instead of direct addressing to send the processes to the pit. By using a two line piece of code that contuously rewrites the reference value used by the fangs, I can "undo" any damage caused by standard anti-vampire programs.

* * *

Writing code for a large coresize provides an interesting challenge. Possibly the most frustrating thing about it is that it is hard to display and analyze 55440 locations. I find that, more than ever, I am forced to rely on other programs to help me generate the information I want for my corewars warriors.

For instance, when I wanted to make the ’94 version of my vampire "Request" run in the 55440 coresize, I tried several things to make the process easier. As you may have guessed by reading the last issue of _The_’94_Warrior_, one of the first programs I used was "Optima" by Michael Constant.

This, however, was just the beginning. Take a look at the following "c" code:

```
/* Determine coverage with different RedCode spacing for bombers */

#include <stdio.h>
#include <stdlib.h>

#define WIDTH 80
```

```
int main(int argc, char *argv[])
{
        long sp, coresize, first, step, lines = WIDTH+1, printed, count;
        int i, core[WIDTH];

        if (argc < 4 || argc > 5) {
                printf("\nUsage:  %s coresize step first [lines]\n", argv[0]);
                return 0;
        }

        coresize = atol(argv[1]);
        step = atol(argv[2]);
        if(step < 0)
                step += coresize;
        first = atol(argv[3]);

        if(argc > 4)
                lines = atol(argv[4]);

        for(i = 0; i < WIDTH; i++)
                core[i] = 0;

        printed = 0;
        sp = first;
        count = 0;
        printf("%ld:\n", step);

        do {
                if(sp >= 0 && sp < WIDTH) {
                        core[sp] = 1;
                        for(i = 0; i < WIDTH; i++)
                                if(i == first)
                                        printf("@");
                                else
                                        printf("%c", core[i] ? '*' : ' ');
                        printed++;
                }

                sp += step;
                sp %= coresize;
                count++;
        } while(sp != first && printed < lines);

        printf("\nMod: %ld\n", coresize / count);

        return 0;
}
```

This program simply takes an imaginary chunk of computer core with a given starting point and step size, and plots how that piece of core will be hit up as the bombing progresses. Then I was able to simply take the pieces of request and fit them into the spaces that were hit latest.
(Since my computer screen is 80 columns wide, I simply used that size of core. Feel free to change WIDTH as appropriate.) In the output, the "@" indicates the very first location bombed (if it is in the range printed). Of course, you can also make that the last location bombed by simply adding one step value to your starting value.

Also, take a look at the bootstrapping code itself. By using the "FOR" macro, and using the "SUB" statements in my bootstrapping code, it is trivial to change the locations of the different parts of the vampire, because the distances between them are automatically computed by the boot-strapping code based on the number of "DAT" statements between the segments. I can even change the order of the segments without touching my bootstrapping code; I simply use the cut an paste tools in my editor to rearrange the segments as I wanted them.

The end result: a competitive program that I have never run inside of a size 55440 core. Is this a good way to go about it? I have no idea. It is, however, an excellent example about how much trouble some people will go to to avoid a little bit of work.

* * *

```
;redcode-94x
;name Request-55440
;author Brant D. Thomsen
;strategy '94 Vampire
;strategy The latest program in my "quest"
;strategy to yield less wins to anti-vampiric programs.
;strategy Submitted: @date@

step    equ     9719
init    equ     (pit+1+step)

gate    equ     (jump-12)
decoy   equ     (gate-1+step)

FOR 132
        dat     #start*10, #1   ; Provide a large decoy.
ROF

start   mov     jump, <dist

        sub     #jump-adder-1, dist

        mov     adder, <dist

        sub     #adder-pit-5, dist

        mov     pit+4, <dist
        mov     pit+3, <dist
        mov     pit+2, <dist
        mov     pit+1, <dist
        mov     pit, <dist

        sub     #pit-vamp-4, dist

        mov     vamp+3, <dist
        mov     vamp+2, <dist
        mov     vamp+1, <dist
        mov     vamp, <dist
        spl     @dist, <10000   ; Start the vampire

        sub     #vamp-help-2, dist

        mov     help+1, <dist
        mov     help, <dist
        spl     @dist, <20000   ; Send 2 processes to "help" so that will
        spl     @dist, <30000   ; execute "MOV" once cycle.

        sub     #help-wimp-1, dist

        mov     wimp, <dist
        spl     @dist, <40000

        sub     #10000, dist    ; Erase record of where boot-strapped.
dist    dat     #15501          ; Then kill the process.


        ; The following is the code for the program.
        ; Spacing (for DATs between segments) is computed automatically.

jump    jmp     @decoy-init, init

FOR 8
        dat     #start*10, #1
```

```
ROF

        ; Wimp is necessary so will still have a process when pit dies.
        ; Runs no risk of starvation since no SPL in it.

wimp    jmp     #0, <gate

FOR 3
        dat     #start*10, #1
ROF

pit     mov     @8, <gate-1
        spl     #0, <gate
        spl     -2, <gate
        spl     -2, <gate
        djn.F   pit, >adder-2

FOR 9
        dat     #start*10, #1
ROF

help    mov     #pit-decoy, decoy
        jmp     help, <gate

FOR 6
        dat     #start*10, #1
ROF

adder   dat     <-step, <step

FOR 3
        dat     #start*10, #1
ROF

vamp    spl     #0, <gate
move    mov     @0, @jump
        add     adder, @move
        djn.F   -2, <-step

        end     start
```

Looking to the Future:

There promise to be some interesting times ahead for the ’94 draft, the ’94 hills, and _The_’94_Warrior_. With the advent of the "pizza" hills, I have no doubt that the number and quality of warriors on the ’94 hills will increase at an unprecedented rate. (In fact, one could probably argue that they already have.) Of course, this also means that those of you who have been putting off experimenting around with the new standard no longer have an excuse! (Well, not a _good_ excuse.)

If you have any comments or questions about the ’94 hills or the ’94 standard that you think might be of general interest, please let me know.

Good luck, and happy computing!

Brant D. Thomsen, Editor
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Snail mail: 1197 East 6290 South
Salt Lake City, UT 84121
U.S.A.

--

Brant D. Thomsen
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Man will occasionally stumble over the truth,
but most times he will pick himself up
and carry on. - Winston Churchill

From news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!news.cs.utah.edu!peruvian.cs.utah.edu!bdthomse
Fri May 6 11:20:07 1994
Article: 744 of rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!news.cs.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse@peruvian.cs.utah.edu (Brant Thomsen)
Newsgroups: rec.games.corewar
Subject: The ’94 Warrior
Date: 5 May 1994 05:06:00 GMT
Organization: University of Utah CS Dept
Lines: 320
Distribution: world
Message-ID: <2q9uro$okk@magus.cs.utah.edu>
NNTP-Posting-Host: peruvian.cs.utah.edu
Originator: bdthomse@peruvian.cs.utah.edu

```
__ __| |            ) _ \  |  |    \ \        /              _)
   |   __ \   _ \  / (   | |  |     \ \  \   / _‘ |  __|  __| |  _ \   __|
   |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
  _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

May 4, 1994 Issue #7

This newletter covers the current status of the ICWS ’94 Draft hills, and also attempts to keep up with the latest ideas in how the new standard will affect corewars in general. I hope you enjoy it!

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

CHANGES and CORRECTIONS:

Excitment is thick in the air after the first two rounds of Stefan Strack’s corewar tournament. It’s still too early to tell what will happen, and if the fierce competition on the ’94 hills is any indication of what’s to come, it will not get any easier to predict later on. (Still, I know who I want to win!) Good luck to all the remaining participants! (... and condolences to Michael Constant and Nandor Sieben, the first two participants to be eliminated.)

If you are working on warriors for a large size core on an IBM computer with Super-VGA graphics, you will definitely want to grab the latest version of pMARS off of "soda.berkeley.edu". It is now possible to watch your large warriors in action! Thanks to the pMARS team for a great effort!

Other helps now available include a faster verion of Optima (by Michael Constant) and a step-size help file for 8000 size cores by Paul Kline that is readable. Both are tools no redcode developer should be without.

The ICWS ’94 Draft Hill:

```
       Core size:       8000 instrucitons
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	44/	36/	20	Pyramid v1.0	Michael Constant	151	1
2	38/	25/	38	Der Zweite Blitzkrieg - 9	Mike Nonemacher	151	112

3	39/ 28/ 33	Torch	P.Kline	150	2
4	43/ 36/ 21	Sauron v5.0	Michael Constant	150	12
5	32/ 16/ 52	Blue Funk	Steven Morrell	148	129
6	32/ 17/ 51	Cannonade	P.Kline	147	4
7	44/ 43/ 14	Iron Gate 1.5	Wayne Sheppard	145	106
8	33/ 23/ 44	Lucky 3	Stefan Strack	144	122
9	32/ 21/ 47	Twimpede+/Small-X	Jay Han	143	5
10	40/ 38/ 21	Request v2.0	Brant D. Thomsen	143	160
11	32/ 22/ 46	NC 94	Wayne Sheppard	142	142
12	43/ 48/ 8	Rave 4.1	Stefan Strack	138	94
13	41/ 45/ 14	Dragon Spear	c w blue	137	162
14	29/ 21/ 50	^C	Steven Morrell	137	21
15	36/ 36/ 28	Christopher	Steven Morrell	135	50
16	30/ 25/ 45	Splash	Jay Han	134	100
17	29/ 25/ 46	Imperfection v3.4	Michael Constant	133	25
18	35/ 37/ 28	Yop La Boum v2.1	P.E.M & E.C.	133	92
19	38/ 44/ 18	SJ-4	J.Layland	133	36
20	40/ 48/ 12	Iron Gate 1.2b	Wayne Sheppard	131	3
21	2/ 98/ 0	Looking	Brant D. Thomsen	7	0

Since the last issue of the Warrior, exactly 99 programs have made it onto the ’94 standard hill. "Der Zweite Blitzkrieg - 9", "Pyramid v1.0", "Sauron v5.0", and "Torch" are all been fighting for the top slot on the hill, and all of them deserve it. It looks like the "pizza" hill is well represented by every type of program but "paper". (Anders Ivner’s "Killer Instinct" was in third place on the hill last month. However, it is still doing well on the "stormking" hill.) Perhaps "paper" is the one warrior type that doesn’t benefit from the newer standard. (Opinions, anyone?)

I am also including the results of the "stormking" hills in this, and future, issues of _The_’94_Warrior_. I had planned to cover the "pizza" hill exclusively, but later decided that the "stormking" hills have enough traffic to warrant their coverage as well. In addition, they also tend to have a much different "flavor" resulting from the quite different programs on the hill. If you have a program that can’t quite get onto a hill on one server, I’d recommend trying it on the other one -- it may do quite well!

The current ICWS ’94 Draft hill on "Stormking":

#	%W/ %L/ %T	Name	Author	Score	Age
1	45/ 29/ 26	Sauron v3.6	Michael Constant	160	1
2	41/ 27/ 32	Killer instinct	Anders Ivner	155	24
3	36/ 21/ 43	Twimpede+/8000-d1	Jay Han	150	14
4	44/ 38/ 17	Ntttgtstitd	Simon Hovell	150	25
5	43/ 38/ 19	Request v2.0	Brant D. Thomsen	148	17
6	34/ 21/ 44	Lucky 3	Stefan Strack	147	12
7	35/ 23/ 42	NC II	Wayne Sheppard	147	79
8	35/ 25/ 40	Sphinx v5.1	W. Mintardjo	145	82
9	43/ 41/ 17	Sylvester v1.0	Brant D. Thomsen	144	61
10	29/ 19/ 53	ttti	nandor sieben	139	35
11	32/ 26/ 42	JustTakingALookSee	J.Layland	138	78
12	31/ 24/ 45	Snake	Wayne Sheppard	138	34
13	43/ 47/ 10	Rave 4.1	Stefan Strack	138	7
14	39/ 40/ 21	tiny	J.Layland	138	59
15	29/ 20/ 51	ttti94	nandor sieben	137	30
16	39/ 42/ 19	Beholder’s Eye v1.7	W. Mintardjo	137	91
17	38/ 42/ 19	Christopher	Steven Morrell	135	23
18	39/ 43/ 18	SJ-4	J.Layland	134	28
19	37/ 43/ 20	Fast Food v2.1	Brant D. Thomsen	131	37
20	35/ 40/ 26	pepper	P.Kline	129	6

The "stormking" hill appears to be even more diverse than the "pizza" hill. This hill will be a fun one to watch as it developes. Many of the programs on it are ’88 standard programs, and several other programs have been converted from the ’88 to the ’94 standard. The "pizza" hill, on the other hand, has many warriors that are highly ’94 dependent.

The ICWS ’94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS ’94 Experimental (Big) hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	51/	32/	18	Demand-55440	M. Constant & B. Tho	170	9
2	39/	21/	40	Variation G-1	Jay Han	157	44
3	44/	35/	20	Request-55440	Brant D. Thomsen	153	80
4	35/	22/	43	Splash 1	Jay Han	149	45
5	45/	43/	12	Squint	Mike Nonemacher	148	18
6	37/	27/	36	Lucky 13	Stefan Strack	148	86
7	41/	37/	22	Vanity IIx	Stefan Strack	146	35
8	33/	24/	44	BigImps	James Layland	142	15
9	43/	45/	12	Virus	Jay Han	141	3
10	43/	46/	11	Rave B4.1	Stefan Strack	141	41
11	41/	42/	17	Raiden	Richard van der Brug	139	14
12	39/	40/	21	White Fang	Steven Morrell	139	5
13	32/	25/	42	Der Zweite Blitzkrieg - 9	Mike Nonemacher	139	42
14	28/	18/	54	Imperfection v2.4	Michael Constant	138	51
15	43/	49/	8	Scanalyzer	Jay Han	138	4
16	38/	38/	24	Sauron v3.4	Michael Constant	137	23
17	38/	38/	24	Lump	J.Layland	137	25
18	34/	32/	34	Sissy	Jay Han	137	40
19	37/	45/	18	testing2	J.Layland	129	1
20	5/	0/	0	Big J-1	Jay Han	14	2

Although many new programs have made it onto the hill, the hill hasn’t changed all that much. Scanners, imp-spirals, and vampires all seem to be well represented. Perhaps the largest change is the addition of "Demand-55440", which currently has a stronghold on the top position. "Demand" appears to be good proof of something I’ve heard repeated many times in the newsgroup by prominent redcode authors: the best warriors tend to be those that have a combination of strategies or types.

The current ICWS ’94 Experimental (Big) hill on "Stormking":

#	%W/	%L/	%T	Name	Author	Score	Age
1	53/	31/	15	Raiden	Richard van der Brug	175	1
2	44/	21/	35	Lucky 13	Stefan Strack	168	18
3	46/	30/	25	Request-55440	Brant D. Thomsen	161	52
4	39/	19/	42	Bakers Dozen	Wayne Sheppard	159	11
5	43/	28/	29	Sauron v2.4	Michael Constant	158	3
6	41/	28/	30	Variation D-1	Jay Han	155	13
7	45/	36/	20	Vanity IIx	Stefan Strack	154	6
8	33/	17/	50	Imperfection v2.3	Michael Constant	149	46
9	44/	46/	10	Rave B4.1	Stefan Strack	141	7
10	30/	23/	47	BigImps	James Layland	138	112
11	41/	45/	14	Dagger v7.0	Michael Constant	136	12
12	41/	46/	13	bigproba	nandor sieben	136	10
13	30/	24/	47	BigImp	Alex MacAulay	135	93
14	34/	34/	32	Industrious	Stefan Strack	134	2
15	41/	47/	12	The Count	Jay Han	134	42
16	34/	38/	27	Test	Stefan Strack	131	4
17	34/	38/	27	Open Arms	Stefan Strack	130	5
18	31/	34/	35	Veeble Jr.	T. H. Davies	129	14
19	35/	45/	19	IceCube 1.4	Richard van der Brug	125	8
20	37/	51/	11	Kill Imps!!!	Steven Morrell	123	39

HINTS and HELPS:

I wanted to come up with some good use for the MUL and DIV instructions for this issue’s hint. There has been a large amount of ’94 code posted recently, but I don’t recall any of it that used either of these instructions.

The two things that I was able to imagine that MUL and DIV would be useful for is a binary bomber (where the programs bombs over successively smaller intervals), and programs that must be able to run in many different possible coresizes. So, along this line, I came up with a program which will

launch an imp-spiral with the minimum number of points for any size core (given a know size limit).

At first, I tried to do this without using any constants, but the "mod" mathematical limitations became too much. Recall that for an "P" point imp-spiral to be possible in a size "C" core, then for any step-size "S" where S * P = N * C + 1 (and N is any possible integer in the range 1 <= N < P), then we have a stepsize to create an imp-spiral with that many points. The problem is, of course, that N * C + 1 becomes 1 in the core no matter what integer value N is assigned. This can make finding the possible imp-steps for large values of N quite difficult to do during run-time.

So, instead, I took a compromise. I take the CORESIZE value supplied by pMARS, and use it to determine all the possible imp-spiral step sizes I want to try. Then I use a simple loop to determine whether or not each possible imp-spiral step size will work. To do this, I take the value I want to test and multiply it by the number of points I am considering for the imp-spiral. If the result is 1 larger than the coresize, I have a successful number. The corewars interpreter automatically takes care of the modular arithmetic, making the loop surprisingly short and simple.

You may want to pay special attention to the main loop in the code, as I do some things in it that are heavily '94 dependent. Each of the modifiers in this loop had to be chosen carefully for the program to work correctly.

The DAT statements on the end of the program will find an imp-spiral for any coresize that is not divisible by 30030. (Notice that 30030 is the product of all the primes no greater than 13.) I have also added the first two 17 point imp-spiral sizes needed to handle 30030 and 60060. If you add all of the DAT statements for the 17-point imp-spiral steps, then you will be able to handle any coresize that is not divisible by 510510. (B.T.W.: 510510 = 17 * 30030) This should be adequate for most programmers. ;-)

Since a randomly sized core will be divisible by 2 half the time, the average number of cycles needed to find the wanted imp-spiral step size should be small. Most of your cycles will be taken up in generating the processes for the imp-spiral and launching it. (In reality, you would probably want less processes than I have here.) Perhaps some of you corewar experts out there could even figure out a way to have the number of processes sent to the imp-spiral vary dymanically based on the number of points it will have.

```
;redcode-94
;name Imp-Spiral Finder
;author Brant Thomsen
;strategy Launch the smallest possible imp-spiral for any coresize < 90090
;macro

boot    equ     150


imp     mov.I   #0, 0           ; Will copy value here later.

find    mov.B   <d2, #0         ; Get the value to test.
        mul.AB  @d2, find       ; See what the result would be.
        djn.B   find, find      ; If result != 1, then try next value.

launch  mov.B   @d2, imp        ; Put the value in the imp.
        mov.I   imp, imp+boot   ; Move the initial imp-spiral instruction.

        ; Begin to generate processes.

        spl.F   2
for 5
        spl.F   1
rof

        ; Should have 48 processes here.
```

```
        spl.F   2               ; Launch an imp-spiral using JMP/ADD method.
        jmp.F   @0, imp+boot    ; Nice since don't need step size in advance.
        add.F   imp, -1


        ; These are the test values to use.
        ; Notice that only prime numbers are used.

        dat.F   17, (d17 * CORESIZE + 1) / 17   ; Needed for size 30030
        dat.F   17, (d17 * CORESIZE + 1) / 17   ; Needed for size 60060
d17
for 12
        dat.F   13, (d13 * CORESIZE + 1) / 13
rof
d13
for 10
        dat.F   11, (d11 * CORESIZE + 1) / 11
rof
d11
for 6
        dat.F   7, (d7 * CORESIZE + 1) / 7
rof
d7
for 4
        dat.F   5, (d5 * CORESIZE + 1) / 5
rof
d5
for 2
        dat.F   3, (d3 * CORESIZE + 1) / 3
rof
d3      dat.F   2, (d2 * CORESIZE + 1) / 2

d2      end     find            ; This instruction is a DAT.F 0, 0
                                ; It is used at the pointer to the
                                ;  current test value.
```

Looking to the Future:

It looks as if the '94 standard is becoming widely accepted. There are more programs being sent to the '94 draft hill than to the '88 standard hill on "pizza", and the disproportion appears to be growing. I have a feeling that it won't be long before the '94 draft hill will be the _standard_ hill.

If you have any comments or questions about the '94 hills or the '94 draft standard that you think might be of general interest, please let me know.

Good luck, and happy computing!

Brant D. Thomsen, Editor Snail mail: 1197 East 6290 South
(bdthomse@peruvian.cs.utah.edu) Salt Lake City, UT 84121
University of Utah U.S.A.
--
Brant D. Thomsen
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Man will occasionally stumble over the truth,
but most times he will pick himself up
and carry on. - Winston Churchill

From news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!news.cs.utah.edu!peruvian.cs.utah.edu!bdthomse
Tue Jun 7 19:46:43 1994
Article: 896 of rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!howland.reston.ans.net!agate!dog.ee.lbl.gov!hellgate.utah.edu!news.cs.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse@peruvian.cs.utah.edu (Brant Thomsen)
Newsgroups: rec.games.corewar
Subject: The ’94 Warrior
Date: 7 Jun 1994 06:32:41 GMT
Organization: University of Utah CS Dept
Lines: 320
Distribution: world
Message-ID: <2t14a9$211@magus.cs.utah.edu>
NNTP-Posting-Host: peruvian.cs.utah.edu
Originator: bdthomse@peruvian.cs.utah.edu

```
__ __| |            ) _ \  |  |    \ \        /              _)
   |   __ \   _ \  / (   | |  |     \ \  \   / _‘ |  __|  __| |  _ \   __|
   |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
  _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

June 6, 1994 Issue #8

This newletter covers the current status of the ICWS ’94 Draft hills, and also attempts to keep up with the latest ideas in how the new standard will affect corewars in general. I hope you enjoy it!

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

CHANGES and CORRECTIONS:

The latest tournament is over, and Jay Han walked away the winner. Congratulations to Jay on his impressive achievement, and thanks again to Stefan Strack again for running it.

Jay Han also appears to have won another on-going battle. His "corestep" program has left the two competing "optima" programs in the dust. You can get the source code for it, or Stefan Strack’s varying step version "mopt", by anonymous FTP to soda.berkeley.edu.

The pMARS team has been busy as well. The next release will allow the use of a-field indirect addressing, and will introduce the ability to use ";assert" to confirm that your program is running in a core type that it was written for. Be sure to download a copy when it is released.

The newsgroup has also had some interesting discussion and results on working with genetic algorithms to create redcode warriors. (Using "genetic" methods to guide the development of "random" programs.) Watch the newsgroup for further details.

The ICWS ’94 Draft Hill:

Core size:	8000 instrucitons
Max processes:	8000 per program
Duration:	After 80,000 cycles, a tie is declared.
Max entry length:	100 instructions

The current ICWS ’94 Draft hill on "Pizza":

#	%W/ %L/ %T	Name	Author	Score	Age

1	49/ 34/ 17	Pyramid v5.5	Michael Constant	163	54
2	48/ 33/ 19	Keystone t33	P.Kline	162	76
3	40/ 29/ 30	Torch t3	P.Kline	151	15
4	43/ 43/ 15	Iron Gate 1.5	Wayne Sheppard	143	253
5	32/ 21/ 47	B-Panama IX	Steven Morrell	142	4
6	38/ 35/ 27	Stimpy v2.0	Brant D. Thomsen	142	5
7	30/ 18/ 52	Blue Funk	Steven Morrell	142	276
8	31/ 22/ 47	Cannonade	P.Kline	140	134
9	32/ 24/ 44	NC 94	Wayne Sheppard	140	289
10	41/ 43/ 16	Aleph 0	Jay Han	140	50
11	43/ 47/ 10	Rave 4.1	Stefan Strack	139	241
12	38/ 38/ 24	Christopher	Steven Morrell	139	197
13	38/ 39/ 23	Request v2.0	Brant D. Thomsen	138	307
14	40/ 44/ 16	Dragon Spear	c w blue	136	309
15	31/ 28/ 41	Lucky 3	Stefan Strack	134	269
16	29/ 25/ 46	TCh	Mintardjo W.	134	6
17	30/ 28/ 42	Aeka	T.Hsu	132	2
18	31/ 31/ 38	Der Zweite Blitzkrieg - 9	Mike Nonemacher	132	259
19	33/ 36/ 31	mmfP v2	Karl Lewin	131	7
20	5/ 0/ 0	Pyramid v5.5	Michael Constant	14	1

The above ordering seems to be a good representation of what has been happening on the hill for the last couple of weeks. Michael Constant's "Pyramid" and Paul Kline's "Keystone" have both been fighting for the top position, with "Torch" being the only close competitor. Both "Pyramid" and "Keystone" are excellent examples of complex programs that push the 100 line limit, although they are radically different from each other.

Also dramatic are the drop of Mike Nonemacher's "Der Zweite Blitzkrieg" and Michael Constant's "Sauron" (right off the hill), as both were putting up a good fight for the top position only a month ago. Although the hill is fairly old, the new programs seem to be making a big difference.

The current ICWS '94 Draft hill on "Stormking":

#	%W/ %L/ %T	Name	Author	Score	Age
1	45/ 29/ 26	Sauron v3.6	Michael Constant	160	1
2	41/ 27/ 32	Killer instinct	Anders Ivner	155	24
3	36/ 21/ 43	Twimpede+/8000-d1	Jay Han	150	14
4	44/ 38/ 17	Ntttgtstitd	Simon Hovell	150	25
5	43/ 38/ 19	Request v2.0	Brant D. Thomsen	148	17
6	34/ 21/ 44	Lucky 3	Stefan Strack	147	12
7	35/ 23/ 42	NC II	Wayne Sheppard	147	79
8	35/ 25/ 40	Sphinx v5.1	W. Mintardjo	145	82
9	43/ 41/ 17	Sylvester v1.0	Brant D. Thomsen	144	61
10	29/ 19/ 53	ttti	nandor sieben	139	35
11	32/ 26/ 42	JustTakingALookSee	J.Layland	138	78
12	31/ 24/ 45	Snake	Wayne Sheppard	138	34
13	43/ 47/ 10	Rave 4.1	Stefan Strack	138	7
14	39/ 40/ 21	tiny	J.Layland	138	59
15	29/ 20/ 51	ttti94	nandor sieben	137	30
16	39/ 42/ 19	Beholder's Eye v1.7	W. Mintardjo	137	91
17	38/ 42/ 19	Christopher	Steven Morrell	135	23
18	39/ 43/ 18	SJ-4	J.Layland	134	28
19	37/ 43/ 20	Fast Food v2.1	Brant D. Thomsen	131	37
20	35/ 40/ 26	pepper	P.Kline	129	6

The ICWS '94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS '94 Experimental (Big) hill on "Pizza":

#	%W/ %L/ %T	Name	Author	Score	Age
1	49/ 32/ 19	Pyramid v5.3	Michael Constant	166	38
2	49/ 32/ 19	ivscan6b	J.Layland	165	11
3	37/ 17/ 46	Aleph 1	Jay Han	158	9
4	46/ 34/ 20	Request-55440	Brant D. Thomsen	157	147

5	44/	36/	20	Aleph 0/1	Jay Han	153	3
6	42/	37/	21	Vanity IIx	Stefan Strack	147	102
7	35/	24/	41	Variation G-1	Jay Han	146	111
8	41/	36/	24	Stimpy v2.0	Brant D. Thomsen	146	2
9	44/	45/	11	Squint	Mike Nonemacher	143	85
10	40/	39/	20	Aleph 0	Jay Han	142	10
11	44/	47/	9	Rave B4.1	Stefan Strack	140	108
12	31/	24/	44	Der Zweite Blitzkrieg - 9	Mike Nonemacher	139	109
13	32/	26/	42	Splash 1	Jay Han	137	112
14	30/	24/	47	NotSoBigImps	James Layland	136	7
15	36/	38/	26	Lump	J.Layland	135	92
16	31/	29/	40	Lucky 13	Stefan Strack	133	153
17	40/	49/	11	Plasma v5	Wayne Sheppard	132	49
18	28/	25/	47	Blue Funk	Steven Morrell	131	1
19	27/	30/	43	Tail of the Twister	Mike Nonemacher	124	57
20	25/	33/	43	avp2	J.Layland	117	5

On this hill, just as on the standard hill, Michael Constant’s quick-scanning vampire "Pyramid" is a string first-place contender. However, unlike last month, there is a program that is giving him some close competition: "ivscan6b", an updated version of James Layland’s notable tournament entry.

Jay Han’s "Aleph" series, and Brant Thomsen’s "Stimpy" are a couple of other new programs that seem to have taken a liking to the Big hill. Perhaps scanners will be the programs that dominate this hill in the future.

The current ICWS ’94 Experimental (Big) hill on "Stormking":

#	%W/	%L/	%T	Name	Author	Score	Age
1	48/	12/	40	Variation M-1	Jay Han	184	2
2	46/	30/	24	Request-55440	Brant D. Thomsen	162	54
3	40/	20/	40	Lucky 13	Stefan Strack	161	20
4	46/	36/	18	Raiden	Richard van der Brug	157	3
5	45/	35/	19	Vanity IIx	Stefan Strack	155	8
6	35/	18/	47	Bakers Dozen	Wayne Sheppard	153	13
7	40/	30/	31	Sauron v2.4	Michael Constant	150	5
8	30/	15/	55	Imperfection v2.3	Michael Constant	146	48
9	36/	31/	33	Variation D-1	Jay Han	142	15
10	44/	47/	10	Rave B4.1	Stefan Strack	141	9
11	42/	46/	12	bigproba	nandor sieben	138	12
12	41/	46/	14	Dagger v7.0	Michael Constant	136	14
13	40/	48/	11	The Count	Jay Han	132	44
14	27/	23/	50	BigImp	Alex MacAulay	132	95
15	30/	29/	41	jmpWetPaper-94x-a	J.M.Pohjalainen	131	1
16	26/	23/	51	BigImps	James Layland	129	114
17	31/	35/	34	Veeble Jr.	T. H. Davies	126	16
18	28/	37/	34	Industrious	Stefan Strack	119	4
19	29/	40/	31	Open Arms	Stefan Strack	117	7
20	29/	41/	30	Test	Stefan Strack	116	6

HINTS and HELPS:

One of the problems that has been haunting corewar programmers through the years is the differences in behavior caused by in-register and in-memory evaluation of an instruction. Actually, it’s usually not the evaluation itself that is the problem -- instead, it is the fact that few corewar programmers know what the difference is between them, or how these differences affect the behavior of programs. I’d like to help clarify this difference, and give some pointers to help you predict what your code will do under each type.

If you think about it, you can probably guess exactly what "in-memory" and "in-register" are referring to. With in-memory evaluation, changes to the core are made directly to the memory storing the core. In-register evaluation, on the other hand, copies the current instruction and the instructions it points to into registers, and the information in the registers is then used to update the core.

There are advantages to both types of evaluation. In-memory evaluation is

much more intuitive. In other words, it is the way that beginners expect redcode programs to operate. In-register evaluation's primary advantage is that it is faster. Most of the '88 compliant corewar emulators, including the one used for the original KotH hill, are in-register.

For those of you that are curious, the '94 draft standard explicitly states that in-register evaluation should be used. In fact, much of the reason a new standard is needed is make this clarification. Although most code will function exactly the same under either type of evaluation, there are a growing number of programs where in-register/in-memory conflicts are a very important consideration.

One of these programs is Paul Kline's "Torch". For those of you who missed it, here is a copy of the source code:

```
;redcode-94 quiet
;name Torch
;author P.Kline
;strategy very rapid incendiary bombing, core-clear & gate
;macro

step     equ 73
count    equ 1500

gate     dat   <-step+1,<-step
     for 21
         dat   0,0
     rof
sp       spl   #0,<-step             ; spl half of the incendiary
in       add   #step+step,msm        ; |
msm      mov   sm,>tgt-(step*count)  ; | these 3 execute in reverse order!
msp      mov   sp,@msm               ; |
tgt      djn.f in,>3157              ; gets bombed with spl to start clear
clr      mov   gate,<-13
cp       djn.f clr,>3                ; forward decrement while clearing
     for 32
         dat   0,0
     rof
sm       mov   @0,>step              ; mov half of the incendiary
         end   sp
```

This program has a bomb that looks like this:

```
    sm  mov  sp, >sp
    ...
    sp  spl  #0, <sm+1
```

As processes execute the MOV statement, they copy the SPL instruction progressively through the core, starting with the location immediately following the MOV. Thus, the more processes there are that execute the MOV statement, the longer the line of SPL instructions (the carpet) will be.

If this bomb were executed using in-memory evaluation, then the length of the SPL carpet would always be equivalent to the number of times the MOV instruction were executed. However, with in-register evaluation, something interesting happens when the MOV instruction tries to overwrite the SPL instruction it uses as a pointer. Because of the way the values are stored in the registers, the increment is lost when the SPL instruction is copied over itself. Thus, the SPL instruction's B-field stays at 0 and the carpet never progresses beyond it -- very effectively reducing the threat of having the enemy capture your own code with the carpet it made after being bombed.

You get a very similar effect with the DJN.F instruction at "cp". If the instruction 3 locations beyond it has a B-value of 0, the DJN stream never progresses beyond that instruction. However, unlike the bomb mentioned earlier, I believe this is the same behavior you would achieve if using an evaluator with _true_ in-memory evaluation. (On Stefan Strack's CoreWar Pro,

you get a different interesting effect instead. The "post-increment" is performed _before_ the DJN, causing the DJN to bring it back to 0 and drop out of the loop.) If nothing else, ambiguities such as this one are very effective at pointing out the advantages of including actual code for a corewar emulator in the '94 standard -- it allows us to determine exactly what a piece of code should behave like. (I found lines 511-561 and lines 750-1416 [section 5.6] of the February 2 draft of the standard to be very helpful when working on this issue's hint. I'd recommend going over them if you would like to know more about exactly how instructions are evaluated.)

When I first started on this issue's hint, I had hoped to be able to come up with an exact list of the instructions that differ between in-memory and in-register evaluation. As I studied it closer, I have come to understand what a difficult task that would be. So, instead, I will simply share the rule of thumb that I have been using. If anyone else out there has some additional advice or cases where they found a program to work differently under in-register than in-memory evaluation, please pass them along.

Based on my own experience, my advice is to be extra cautious whenever you use the MOV instruction. In-register differences will only appear when the issue of self-reference is involved; this can consist of moving an instruction onto the instruction doing the moving, or moving an instruction onto the instruction holding the indirect pointer for your movement. Although potential problems exist for other instructions as well, I have never experienced any of them -- nor can I recall anyone else that ever has.

If you have an instruction that may reference itself, either directly or indirectly, be prepared to have to work around the problem or even scrap the program altogether; or, if you can, exploit it to come up with an even better warrior!

[Thanks again to Paul Kline for allowing me the use of his code.]

Looking to the Future:

There have been several modifications to the pMARS instruction set over the last couple of months, and there look to be more in the immediate future. Once again, I strongly encourage everyone out there to experiment with these modifications and let us know what you think of them. It will be too late after the '94 standard is approved!

If you have any comments or questions about the '94 hills or the '94 draft standard that you think might be of general interest, please let me know. Also, if there are any particular topics you would like to see covered in future editions of _The_'94_Warrior_, please send me e-mail on that as well.

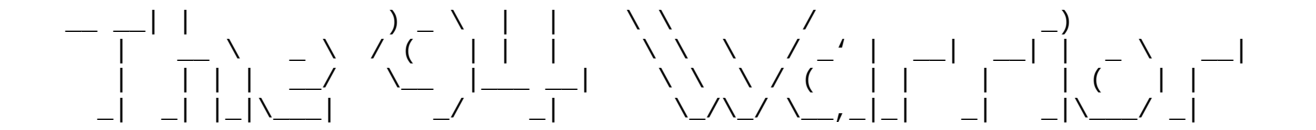
Good luck, and happy computing!

Brant D. Thomsen, Editor
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Snail mail: 1197 East 6290 South
Salt Lake City, UT 84121
U.S.A.

--
Brant D. Thomsen
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Man will occasionally stumble over the truth,
but most times he will pick himself up
and carry on. - Winston Churchill



June 21, 1994 Issue #9

This newsletter covers the current status of the ICWS ’94 Draft hills, and also attempts to keep up with the latest ideas in how the new standard will affect corewars in general. I hope you enjoy it!

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

CHANGES and CORRECTIONS:

A new version of pMARS has been released. The latest version of pMARS now has A-field indirection, and encourages the use of the ";assert" command to make sure that the hill you are running your program on is adequate. You can get it through anonymous FTP to soda.berkeley.edu.

I also owe Michael Constant an apology. Sauron was removed (killed) by it’s owner, rather than being pushed off of the ’94 draft hill -- as I stated in the last issue of _The_’94_Warrior_.

The ICWS ’94 Draft Hill:

```
       Core size:       8000 instructions
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill on "Pizza":

#	%W/ %L/ %T	Name	Author	Score	Age
1	45/ 34/ 21	Pyramid v5.5	Michael Constant	155	89
2	45/ 36/ 18	Keystone t33	P.Kline	154	111
3	40/ 27/ 33	Torch t5	P.Kline	154	4
4	36/ 23/ 40	Sasami	T.Hsu	150	24
5	38/ 34/ 27	Stimpy v2.0	Brant D. Thomsen	143	40
6	39/ 37/ 23	Christopher	Steven Morrell	141	232
7	30/ 19/ 51	Aeka	T.Hsu	141	1
8	31/ 22/ 47	Cannonade	P.Kline	141	169
9	39/ 38/ 23	Request v2.0	Brant D. Thomsen	141	342
10	30/ 19/ 51	Blue Funk	Steven Morrell	140	311
11	30/ 20/ 51	Insight v1.0	Brant D. Thomsen	139	31
12	41/ 44/ 14	Iron Gate 1.5	Wayne Sheppard	138	288
13	30/ 22/ 48	B-Panama IX	Steven Morrell	138	39
14	42/ 47/ 11	Rave 4.1	Stefan Strack	137	276
15	39/ 41/ 20	Sauron v6.0	Michael Constant	137	35
16	35/ 32/ 33	mmfP v2	Karl Lewin	137	42
17	32/ 28/ 41	Lucky 3	Stefan Strack	136	304
18	40/ 44/ 16	Dragon Spear	c w blue	136	344
19	30/ 24/ 46	NC 94	Wayne Sheppard	136	324
20	38/ 42/ 20	Aleph 0+	Jay Han	135	21

The top three programs are still Pyramid, Keystone, and Torch. However, there is a new entry that seems to be giving them some stiff competition. Sasami, by T. Hsu, was submitted to the hill a few days ago, and has been clinging to fourth place ever since. The rest of the hill, as far as I can tell, is total chaos. I have a feeling that the next couple of weeks will be really interesting -- especially as more programs are submitted that take

advantage of the latest pMARS additions.

The current ICWS ’94 Draft hill on "Stormking":

#	%W/	%L/	%T	Name	Author	Score	Age
1	45/	29/	26	Sauron v3.6	Michael Constant	160	1
2	41/	27/	32	Killer instinct	Anders Ivner	155	24
3	36/	21/	43	Twimpede+/8000-d1	Jay Han	150	14
4	44/	38/	17	Ntttgtstitd	Simon Hovell	150	25
5	43/	38/	19	Request v2.0	Brant D. Thomsen	148	17
6	34/	21/	44	Lucky 3	Stefan Strack	147	12
7	35/	23/	42	NC II	Wayne Sheppard	147	79
8	35/	25/	40	Sphinx v5.1	W. Mintardjo	145	82
9	43/	41/	17	Sylvester v1.0	Brant D. Thomsen	144	61
10	29/	19/	53	ttti	nandor sieben	139	35
11	32/	26/	42	JustTakingALookSee	J.Layland	138	78
12	31/	24/	45	Snake	Wayne Sheppard	138	34
13	43/	47/	10	Rave 4.1	Stefan Strack	138	7
14	39/	40/	21	tiny	J.Layland	138	59
15	29/	20/	51	ttti94	nandor sieben	137	30
16	39/	42/	19	Beholder’s Eye v1.7	W. Mintardjo	137	91
17	38/	42/	19	Christopher	Steven Morrell	135	23
18	39/	43/	18	SJ-4	J.Layland	134	28
19	37/	43/	20	Fast Food v2.1	Brant D. Thomsen	131	37
20	35/	40/	26	pepper	P.Kline	129	6

The ICWS ’94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS ’94 Experimental (Big) hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	50/	34/	16	ivscan6b	J.Layland	165	17
2	49/	33/	18	Request-55440	Brant D. Thomsen	164	153
3	49/	35/	17	Pyramid v5.3	Michael Constant	163	44
4	37/	18/	46	Aleph 1	Jay Han	156	15
5	42/	34/	24	Stimpy v2.0	Brant D. Thomsen	151	8
6	36/	27/	37	Variation G-1	Jay Han	145	117
7	41/	38/	21	Aleph 0	Jay Han	145	16
8	42/	41/	17	Fscan	Jay Han	142	1
9	32/	25/	43	NotSoBigImps	James Layland	140	13
10	38/	37/	25	Lump	J.Layland	139	98
11	39/	39/	22	Vanity IIx	Stefan Strack	138	108
12	43/	47/	10	Rave B4.1	Stefan Strack	138	114
13	32/	27/	41	Der Zweite Blitzkrieg - 9	Mike Nonemacher	138	115
14	31/	25/	43	Blue Funk	Steven Morrell	137	7
15	42/	46/	12	Squint	Mike Nonemacher	137	91
16	33/	31/	35	Lucky 13	Stefan Strack	135	159
17	42/	48/	10	Plasma v5	Wayne Sheppard	135	55
18	31/	32/	37	Sasami / 55440	T.Hsu	131	3
19	30/	30/	39	Splash 1	Jay Han	130	118
20	7/	2/	0	Fscan	Jay Han	21	2

Not too much is happening on the big hill at the moment. I’ve noticed that the intensity tends to switch between the ’94 draft and the ’94 experimental hills. Then again, perhaps it’s just summer break.

The current ICWS ’94 Experimental (Big) hill on "Stormking":

#	%W/	%L/	%T	Name	Author	Score	Age
1	48/	12/	40	Variation M-1	Jay Han	184	2
2	46/	30/	24	Request-55440	Brant D. Thomsen	162	54
3	40/	20/	40	Lucky 13	Stefan Strack	161	20
4	46/	36/	18	Raiden	Richard van der Brug	157	3
5	45/	35/	19	Vanity IIx	Stefan Strack	155	8
6	35/	18/	47	Bakers Dozen	Wayne Sheppard	153	13
7	40/	30/	31	Sauron v2.4	Michael Constant	150	5
8	30/	15/	55	Imperfection v2.3	Michael Constant	146	48

9	36/ 31/ 33	Variation D-1	Jay Han	142	15
10	44/ 47/ 10	Rave B4.1	Stefan Strack	141	9
11	42/ 46/ 12	bigproba	nandor sieben	138	12
12	41/ 46/ 14	Dagger v7.0	Michael Constant	136	14
13	40/ 48/ 11	The Count	Jay Han	132	44
14	27/ 23/ 50	BigImp	Alex MacAulay	132	95
15	30/ 29/ 41	jmpWetPaper-94x-a	J.M.Pohjalainen	131	1
16	26/ 23/ 51	BigImps	James Layland	129	114
17	31/ 35/ 34	Veeble Jr.	T. H. Davies	126	16
18	28/ 37/ 34	Industrious	Stefan Strack	119	4
19	29/ 40/ 31	Open Arms	Stefan Strack	117	7
20	29/ 41/ 30	Test	Stefan Strack	116	6

HINTS and HELPS:

For this issue's hint, I'd like to make some suggestions about what new abilities the A-field indirection in the latest pMARS release makes possible. (I tried to come up with examples for each of these points, but couldn't find any that I was really pleased with.)

Probably the most obvious benefit is the enhanced ability to store information in the A-field of instructions. With the increasing use of SPL #a, JMP #a, and MOV.I #a instructions, the are an increasing number of instructions in a program that can have their A-field used to store information. Take a look at your old programs and see if there isn't another DAT that you can eliminate, or if you can now implement a SPL/DAT core-clear with that code.

Another feature that is made possible by the latest enhancements to pMARS is that anti-vampiric code is now much easier to write. It is no longer necessary to extract the A-field value from an instruction, as you can now simply use the value in that location as a pointer instead. (Actually, I'm not that excited about having better anti-vampiric programs, since I've grown fond of Request over the last few months.) I tried to take advantage of this ability when I submitted the program Insight to the hill, but it did much worse against vampires -- and better against several other programs -- than I expected.

Along these same lines, I have the feeling that the latest changes to pMARS will result in more intelligent programs. For instance, much more information can be found in boot-strapping code. Instead of just being able to trace where MOV statements point to, the SPL and JMP statements will always be traceable as well. I wouldn't be surprised if there is soon a warrior on the hill that uses a quick-scanner to find bootstrapping code, then tries to exploit any information in that code. Now, more than ever, I think it will become important to spend a couple of extra instructions covering up your trail after you boot-strap.

I have no doubt that there will be some interesting changes to the hill when using these new addressing modes becomes more common. I have already found myself writing programs differently to account for their effects.

Looking to the Future:

I'd like to (again) encourage everyone to experiment around with the latest additions to the pMARS language. Remember that A-field indirection, and the NOP and SNE instructions, have not been added to the draft of the '94 language. It will be your input that will be used to decide, later on, whether or not they will be.

If you have any comments or questions about the '94 hills or the '94 draft standard that you think might be of general interest, please let me know. Also, if there are any particular topics you would like to see covered in future editions of _The_'94_Warrior_, please send me e-mail on that as well.

Good luck, and happy computing!

From news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!cs.utexas.edu!swrinde!ihnp4.ucsd.edu!dog.ee.lbl.gov!news.cs.utah.edu!peruvian.cs.utah.edu!bdthomse Fri Jul 8 16:26:16 1994
Article: 994 of rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!cs.utexas.edu!swrinde!ihnp4.ucsd.edu!dog.ee.lbl.gov!news.cs.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse@peruvian.cs.utah.edu (Brant Thomsen)
Newsgroups: rec.games.corewar
Subject: The ’94 Warrior
Date: 8 Jul 1994 05:13:43 GMT
Organization: University of Utah CS Dept
Lines: 346
Distribution: world
Message-ID: <2vina7$h0b@magus.cs.utah.edu>
NNTP-Posting-Host: peruvian.cs.utah.edu
Originator: bdthomse@peruvian.cs.utah.edu

```
__ __| |            ) _ \  |  |    \ \        /              _)
   |   __ \   _ \  / (   | |  |     \ \  \   / _` |  __|  __| |  _ \   __|
   |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
  _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

July 7, 1994 Issue #10

This newsletter covers the current status of the ICWS ’94 Draft hills, and also attempts to keep up with the latest ideas on how the new standard will affect corewars in general. I hope you enjoy it!

If you are unfamiliar with the ’94 draft standard, you can learn more about it by reading the FAQ for this newsgroup. In addition, the program pMARS includes a highly recommended tutorial on the new standard. Feel free to send me e-mail if you have any difficulty finding either of them, if you need to have a corewar item mailed to you, or if you have any other questions.

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z

CHANGES and CORRECTIONS:

The next chapter in Steven Morrell’s "My First Corewar Book" is now available by anonymous FTP to ftp.csua.berkeley.edu (NOTE THE NEW ADDRESS) in the pub\corewar\incoming as "chapter.2.Z". This chapter provides an excellent overview on "stones" that every budding corewar expert will want to have a copy of.

The first ten issues of this newsletter can also be found on ftp.csua.berkeley.edu as "94warr1.txt.Z". If you missed any issues in the past, now is your chance to get them.

The ICWS ’94 Draft Hill:

```
       Core size:       8000 instructions
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	47/	32/	22	Homemade Ice Cream	P.Kline	161	1
2	40/	27/	33	Torch t5	P.Kline	154	23
3	37/	24/	39	Sasami	T.Hsu	150	43
4	35/	20/	45	Blue Funk 3	Steven Morrell	149	18
5	42/	39/	19	Pyramid v5.5	Michael Constant	146	108
6	42/	39/	19	Keystone t33	P.Kline	145	130
7	33/	22/	45	Ryooki	T.Hsu	143	5
8	30/	18/	53	Aeka	T.Hsu	141	20
9	38/	36/	26	Stimpy v2.0	Brant D. Thomsen	140	59

10	31/	22/	47	NC 94	Wayne Sheppard	139	343
11	39/	39/	21	Christopher	Steven Morrell	139	251
12	29/	19/	51	Cannonade	P.Kline	139	188
13	28/	19/	53	Insight v1.0	Brant D. Thomsen	138	50
14	31/	25/	44	Lucky 3	Stefan Strack	137	323
15	40/	43/	18	SJ-4	J.Layland	137	19
16	41/	47/	13	Iron Gate 1.5	Wayne Sheppard	135	307
17	27/	21/	53	Blue Funk	Steven Morrell	132	330
18	36/	41/	23	Sauron v6.0	Michael Constant	132	54
19	41/	49/	10	Rave 4.1	Stefan Strack	132	295
20	36/	43/	21	Request v2.0	Brant D. Thomsen	129	11

The hill became much younger over the last couple of weeks with the loss of the two oldest programs. "Dragon Spear" by C. W. Blue was kicked off the hill at the rip old age of 346. Three cycles later, "Request" by Brant Thomsen also was overcome, at the age of 347. With the loss of these two fossels, "NC (Night Crawler) 94" is now the oldest program -- and it looks like it will be sticking around for a while still. See Steven Morrell's new tutorial if you're curious how NC does so well.

"Blue Funk 3", "Ryooki" and "Homemade Ice Cream" are three new additions to the hill that all seem to be doing well. Excited corewar fans everywhere are anxiously waiting to find out what other exciting warriors Paul Kline will be cooking up in the future.

The current ICWS '94 Draft hill on "Stormking":

#	%W/	%L/	%T	Name	Author	Score	Age
1	44/	29/	26	Sauron v3.6	Michael Constant	160	3
2	42/	30/	28	Killer instinct	Anders Ivner	154	26
3	36/	20/	43	Twimpede+/8000-d1	Jay Han	152	16
4	35/	20/	45	Lucky 3	Stefan Strack	151	14
5	36/	22/	42	NC II	Wayne Sheppard	150	81
6	36/	24/	40	Sphinx v5.1	W. Mintardjo	147	84
7	43/	40/	17	Ntttgtstitd	Simon Hovell	146	27
8	41/	38/	21	Request v2.0	Brant D. Thomsen	144	19
9	38/	33/	29	stone-test	Stefan Strack	142	1
10	41/	41/	18	Sylvester v1.0	Brant D. Thomsen	141	63
11	29/	19/	52	ttti	nandor sieben	140	37
12	32/	25/	43	JustTakingALookSee	J.Layland	139	80
13	29/	20/	50	ttti94	nandor sieben	138	32
14	30/	23/	46	Snake	Wayne Sheppard	137	36
15	39/	42/	18	Beholder's Eye v1.7	W. Mintardjo	136	93
16	42/	48/	10	Rave 4.1	Stefan Strack	135	9
17	39/	44/	18	SJ-4	J.Layland	134	30
18	38/	42/	21	tiny	J.Layland	133	61
19	37/	43/	20	Christopher	Steven Morrell	131	25
20	36/	44/	20	Fast Food v2.1	Brant D. Thomsen	128	39

The ICWS '94 Draft Experimental Hill:

```
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS '94 Experimental (Big) hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	49/	34/	17	ivscan6b	J.Layland	165	22
2	47/	35/	18	Pyramid v5.3	Michael Constant	159	49
3	46/	34/	20	Request-55440	Brant D. Thomsen	158	158
4	36/	17/	47	Aleph 1	Jay Han	156	20
5	36/	24/	40	Variation G-1	Jay Han	148	122
6	43/	39/	18	Fscan	Jay Han	146	6
7	41/	37/	22	Aleph 0	Jay Han	146	21
8	40/	36/	24	Stimpy v2.0	Brant D. Thomsen	144	13
9	31/	23/	46	NotSoBigImps	James Layland	140	18
10	38/	36/	26	Lump	J.Layland	139	103
11	31/	24/	44	Der Zweite Blitzkrieg - 9	Mike Nonemacher	138	120
12	39/	39/	22	Vanity IIx	Stefan Strack	138	113

13	42/	48/	10	Rave B4.1	Stefan Strack	137	119
14	32/	29/	39	Lucky 13	Stefan Strack	136	164
15	30/	24/	46	Blue Funk	Steven Morrell	136	12
16	41/	47/	13	Squint	Mike Nonemacher	135	96
17	31/	27/	43	Splash 1	Jay Han	135	123
18	40/	49/	11	Plasma v5	Wayne Sheppard	130	60
19	27/	23/	50	Insight v1.0	Brant D. Thomsen	130	1
20	31/	32/	37	Sasami / 55440	T.Hsu	130	8

Things still appear to be quiet on the Experimental hill. Personally, I expect this hill to really pick up as some of the new programs on the ’94 hill start moving over.

I have found it especially interesting how differently programs can do between the two hills. For example, my program "Request" does much better on the Experimental hill, while "Insight" does much better on the Standard hill -- both results being exactly the opposite of what I expected!

The current ICWS ’94 Experimental (Big) hill on "Stormking":

#	%W/	%L/	%T	Name	Author	Score	Age
1	48/	12/	40	Variation M-1	Jay Han	184	2
2	46/	30/	24	Request-55440	Brant D. Thomsen	162	54
3	40/	20/	40	Lucky 13	Stefan Strack	161	20
4	46/	36/	18	Raiden	Richard van der Brug	157	3
5	45/	35/	19	Vanity IIx	Stefan Strack	155	8
6	35/	18/	47	Bakers Dozen	Wayne Sheppard	153	13
7	40/	30/	31	Sauron v2.4	Michael Constant	150	5
8	30/	15/	55	Imperfection v2.3	Michael Constant	146	48
9	36/	31/	33	Variation D-1	Jay Han	142	15
10	44/	47/	10	Rave B4.1	Stefan Strack	141	9
11	42/	46/	12	bigproba	nandor sieben	138	12
12	41/	46/	14	Dagger v7.0	Michael Constant	136	14
13	40/	48/	11	The Count	Jay Han	132	44
14	27/	23/	50	BigImp	Alex MacAulay	132	95
15	30/	29/	41	jmpWetPaper-94x-a	J.M.Pohjalainen	131	1
16	26/	23/	51	BigImps	James Layland	129	114
17	31/	35/	34	Veeble Jr.	T. H. Davies	126	16
18	28/	37/	34	Industrious	Stefan Strack	119	4
19	29/	40/	31	Open Arms	Stefan Strack	117	7
20	29/	41/	30	Test	Stefan Strack	116	6

HINTS and HELPS:

When I first put together the program "Insight", there were several things I was thinking about. Perhaps most of all was the thought that if I quickly threw something together I could have have the first program on the ’94 draft hill that uses A-field indirection. (I’m still not sure if that is the case.) However, beyond that, I wanted to create a simple program that I could use to discover weaknesses in other’s programs. I suppose, in that regard, "Insight" is rather similar to James Layland’s DJN stream experiment. (B.T.W. If you want to know why "Insight" and "Stimpy" did so well against the DJN.F program, simply look at the decoy!)

The fact that "Insight" is actually managing to stay on the hill is simply a nice bonus.

```
;redcode-94
;name Insight v1.0
;author Brant D. Thomsen
;strategy Stone/imp-spiral
;strategy Uses A-field indirection
;strategy Submitted: @date@
;assert CORESIZE == 8000

step    equ     3039
init    equ     step+1
gate    equ     -12
impstep equ     2667
```

```
cdist   equ     CORESIZE / 22

        dat.A   #1, #1  ; Large decoy to slow DJN streams,
        dat.A   #1, 1   ; while still remaining unique
        dat.A   #1, @1  ; in case of CMP scanners.
        dat.A   #1, <1
          .
          .
          .
        dat.BA  1, >1
        dat.BA  1, *1
        dat.BA  1, {1
        dat.BA  1, }1

stone   spl     #gate, <gate
loop    mov     data, *init     ; "data" placed here by last hit,
                                ; to create a "perfect" imp-gate.
        add     #step, loop
        djn.F   loop, @loop     ; Last hit is actually here
data    dat.F   <gate, #0

start   mov     stone + 4, @boot
        mov     stone + 3, <boot
        mov     stone + 2, <boot
        mov     stone + 1, <boot
        mov     stone + 0, <boot

;Binary imp launch
; 3 point, 10 processes -- generated by bimp

spiral  spl     16,                     {cdist * 1      ; Use the B-field of
        spl     8,                      {cdist * 2      ; the SPL and JMP
        spl     4,                      {cdist * 3      ; instructions to
        spl     2,                      {cdist * 4      ; decrement locations
        jmp     imp + 0,                {cdist * 5      ; in hopes of an early
        jmp     imp + impstep,          {cdist * 6      ; advantage
        spl     2,                      {cdist * 7
        jmp     imp + 2 * impstep,      {cdist * 8
        jmp     imp + 1,                {cdist * 9
        spl     4,                      {cdist * 10
        spl     2,                      {cdist * 11
        jmp     imp + impstep + 1,      {cdist * 12
        jmp     imp + 2 * impstep + 1,  {cdist * 13
        spl     2,                      {cdist * 14
        jmp     imp + 2,                {cdist * 15
        jmp     imp + impstep + 2,      {cdist * 16

; Jump to stone
        spl     @boot,                  {cdist * 17
        spl     @boot,                  {cdist * 18

        spl     2,                      {cdist * 19
        jmp     imp + 2 * impstep + 2,  {cdist * 20
        jmp     imp + 3,                {cdist * 21

imp     mov.I   #-100, impstep

        dat.F   #1, #1
        dat.F   1, 1

boot    dat.F   #0, #2550       ; Bootstrapping distance
                                ; Will be overwritten by the imp-spiral

        end     start
```

The stone is certainly nothing to get excited about. The bombs are moved to the location being referenced by the location the stone is pointing at, in the hope that it will point to something important. This is usually a safe gamble, although it is not always the case. In fact, the imp I use for the imp-spiral deliberately has a non-zero A-value to keep me from bombing it.

Here's how it does against the current hill:

Homemade Ice Cream:	08/59/33	(i.e. Insight v1.0 wins 8 times)
Torch t5:	21/32/47	
Sasami:	15/28/57	
Blue Funk 3:	04/03/93	
Pyramid v5.5:	37/14/49	
Keystone t33:	33/36/31	
Ryooki:	00/10/90	
Aeka:	00/12/88	
Stimpy v2.0:	45/27/28	
NC 94:	07/16/77	
Christopher:	60/05/35	
Cannonade:	03/06/91	
Lucky 3:	04/08/88	
SJ-4:	55/26/19	
Iron Gate 1.5:	39/41/20	
Blue Funk:	03/05/92	
Sauron v6.0:	60/27/13	
Rave 4.1:	57/30/13	
Request v2.0:	20/32/48	
Insight v1.0:	02/03/95	

The scores are also fairly traditional. As you would expect from a straight DAT bomber, the program does well against Scanners and poorly against paper-type programs.

The scores against the vampires on the hill are especially interesting. This is because "Insight" has a built-in anti-vampiric element. (Whenever the fang is targeted by the stone, the A-value it finds is automatically traced and bombed.) The only vampire "Insight" doesn't do well against is "Request", which is specifically designed to withstand attacks on the address its fangs point at.

Perhaps the one lesson I can give you from this program is that it may be a good idea to use _non-zero_ A-field values on you SPL #, JMP #, MOV #, and DJN # instructions. As there are no true A-field scanners out there, (they either look at the B-field or both fields,) it will probably give you a slight advantage to use a non-zero A-field on these instructions if the B-field is non-zero.

Looking to the Future:

There are some interesting issues that are cropping up on the future of _The_'94_Warrior_. This newsletter's original (and present) purpose is to ease and encourage the transition of the corewar community to the '94 [draft] standard. However, judging by the difference in the number of submissions to the '88 and '94 hills, and by the current level of competition on the '94 hills, it is my opinion that the transition has already taken place.

I am also running up against the interesting problem of running out of ideas for hints that apply exclusively to the '94 standard. Either we need to add some more features to pMARS, or I'll need to start using more general hints instead. (There's always IJZ, of course ...)

One other issue that needs to be dealt with is, quite honestly, the amount of time that is required to generate these newsletters. With my new job and upcoming marriage making large demands on my schedule, I find that I am spending more time working on this newsletter than in actually writing corewar code. The result of this is that it is becoming more and more difficult for me to keep up with the latest redcoding techniques and to continue to generate a quality newsletter.

I will be happy to continue to produce _The_'94_Warrior_ until someone better-qualified volunteers, but there will need to be some changes made. The newsletter will need to have less depth to it, or be produced less frequently, and the subject matter will need to be more oriented towards corewars in general -- and less towards specifics of the '94 standard.

What I would like from each of you is a list of what you would like to see most in future editions of the Warrior. Which parts of the current newsletter format do you like and dislike the most, and what new features would do the most to encourage, entertain, and educate the corewar community? What specific topics would you like to see covered?

As always, your comments, suggestions, criticisms, and submissions are encouraged and appreciated.

--

Brant D. Thomsen
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Man will occasionally stumble over the truth, but most times he will pick himself up and carry on. - Winston Churchill

From news-rocq.inria.fr!univ-lyon1.fr!swidir.switch.ch!newsfeed.ACO.net!Austria.EU.net!EU
.net!howland.reston.ans.net!cs.utexas.edu!not-for-mail Mon Jul 25 15:27:15 1994
Article: 1009 of rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!swidir.switch.ch!newsfeed.ACO.net!Austria.EU.net!E
U.net!howland.reston.ans.net!cs.utexas.edu!not-for-mail
From: bdthomse@peruvian.cs.utah.edu (Brant Thomsen)
Newsgroups: rec.games.corewar
Subject: The ’94 Warrior
Date: 24 Jul 1994 02:25:26 -0500
Organization: UTexas Mail-to-News Gateway
Lines: 350
Sender: nobody@cs.utexas.edu
Message-ID: <9407240725.AA10219@peruvian.cs.utah.edu>
NNTP-Posting-Host: news.cs.utexas.edu

```
  __ __| |            ) _ \  |  |    \ \        /              _)
     |   __ \   _ \  / (   | |  |     \ \  \   / _‘ |  __|  __| |  _ \   __|
     |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
    _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

July 23, 1994 Issue #11

In the past, _The_’94_Warrior_ was specifically dedicated to encouraging and supporting the ’94 draft standard. Starting with this issue, the newsletter is oriented towards corewars in general. I have kept the title to avoid confusion, and because it is still my hope to keep abreast of, and possibly even accelerate, the latest advances in corewars. I hope you enjoy it!

The FAQ is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z. There are also several tutorials on the ’88 and ’94 (draft) standard available on ftp.csua.berkeley.edu that will greatly ease the process of becoming a proficient "redcoder."

CHANGES and CORRECTIONS:

A slightly updated version of pMARS has been uploaded to ftp.csua.berkeley.edu. Version 0.6.2 has some minor bug fixes, and will also make it much easier to use code that you download from the newsgroup. (A very nice touch!) Thanks again to the pMARS group for an excellent job!

Starting with this issue of the Warrior, I will be covering what I consider to be the three most active corewar hills: the ’94 Draft Hill, the ’94 Experimental (Big) Hill, and the ’88 Standard Hill (all on Pizza@ecst.csuchico.edu). If there is enough interest, I can add other hills as well. (I considered adding the beginner’s hill; but, since I have no programs on it, I have no idea what is happening there. Anyone want to cover that hill for me? Would anyone object if I kept a simple program on the hill just so I could follow it for a while?)

The ICWS ’94 Draft Hill:

```
        Standard:       ’94 Draft (with extensions)
       Core size:       8000 instructions
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	41/	26/	33	Torch t5	P.Kline	155	51
2	46/	38/	16	Pyramid v5.5	Michael Constant	154	136
3	43/	33/	24	Homemade Ice Cream	P.Kline	153	23
4	38/	25/	37	Sasami	T.Hsu	152	71
5	34/	21/	44	Blue Funk 3	Steven Morrell	148	46
6	34/	20/	46	Ryooki	T.Hsu	147	12
7	43/	41/	17	SJ-4	J.Layland	145	47

8	30/ 19/ 51	Aeka	T.Hsu	141	1
9	39/ 37/ 24	Stimpy v2.0	Brant D. Thomsen	141	87
10	30/ 19/ 51	Cannonade	P.Kline	141	216
11	32/ 23/ 45	NC 94	Wayne Sheppard	141	371
12	35/ 30/ 35	FlyPaper 3.0f	J.Layland	140	2
13	38/ 37/ 25	Request v3.0	Brant D. Thomsen	140	14
14	37/ 37/ 26	IVScan 8000	James Layland	137	9
15	41/ 47/ 12	Iron Gate 1.5	Wayne Sheppard	136	335
16	39/ 43/ 18	Keystone t33	P.Kline	135	158
17	38/ 42/ 20	Christopher	Steven Morrell	134	279
18	27/ 22/ 51	Blue Funk	Steven Morrell	133	358
19	30/ 29/ 41	Lucky 3	Stefan Strack	132	351
20	41/ 52/ 7	test	P.Kline	131	4

Another one of the old programs has disappeared from the hill: Rave 4.1 by Stefan Strack was killed at the respectable age of 320. Rave is a "CMP" scanner, and was one of the very first programs written specifically for the ’94 standard.

The leaders on the hill are no surprise, as we have the same five programs on the top of the hill that were there last issue. However, the competition is increasing. The difference between the scores of the top and bottom warriors on the hill is 24, compared with 32 just last issue. That’s about as close as I have ever seen them.

The ICWS ’94 Draft Experimental Hill:

Standard:	’94 Draft (with extensions)
Core size:	55,440 instructions
Max processes:	10,000 per program
Duration:	After 500,000 cycles, a tie is declared.
Max entry length:	200 instructions

The current ICWS ’94 Experimental (Big) hill on "Pizza":

#	%W/ %L/ %T	Name	Author	Score	Age
1	49/ 34/ 17	ivscan6b	J.Layland	165	22
2	47/ 35/ 18	Pyramid v5.3	Michael Constant	159	49
3	46/ 34/ 20	Request-55440	Brant D. Thomsen	158	158
4	36/ 17/ 47	Aleph 1	Jay Han	156	20
5	36/ 24/ 40	Variation G-1	Jay Han	148	122
6	43/ 39/ 18	Fscan	Jay Han	146	6
7	41/ 37/ 22	Aleph 0	Jay Han	146	21
8	40/ 36/ 24	Stimpy v2.0	Brant D. Thomsen	144	13
9	31/ 23/ 46	NotSoBigImps	James Layland	140	18
10	38/ 36/ 26	Lump	J.Layland	139	103
11	31/ 24/ 44	Der Zweite Blitzkrieg - 9	Mike Nonemacher	138	120
12	39/ 39/ 22	Vanity IIx	Stefan Strack	138	113
13	42/ 48/ 10	Rave B4.1	Stefan Strack	137	119
14	32/ 29/ 39	Lucky 13	Stefan Strack	136	164
15	30/ 24/ 46	Blue Funk	Steven Morrell	136	12
16	41/ 47/ 13	Squint	Mike Nonemacher	135	96
17	31/ 27/ 43	Splash 1	Jay Han	135	123
18	40/ 49/ 11	Plasma v5	Wayne Sheppard	130	60
19	27/ 23/ 50	Insight v1.0	Brant D. Thomsen	130	1
20	31/ 32/ 37	Sasami / 55440	T.Hsu	130	8

I suppose it’s difficult to justify picking this as one of the three most active hills, as it hasn’t changed since the last issue of _The_’94_Warrior_. Personally, I just like the fact that this hill has such a different "feel" than the smaller ’94 hill. It makes it a completely different challenge to do well on it. (Besides, Jay Han might not forgive me if I dropped it!)

The ICWS ’88 Standard Hill:

Standard:	’88
Core size:	8000 instructions
Max processes:	8000 per program
Duration:	After 80,000 cycles, a tie is declared.

Max entry length: 100 instructions

The current Standard KotH hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	41/	26/	33	Der Zweite Blitzkrieg	Mike Nonemacher	156	41
2	46/	38/	16	SJ-4a	J.Layland	155	40
3	39/	24/	37	Night Crawler	Wayne Sheppard	154	69
4	44/	37/	19	Christopher	Steven Morrell	150	33
5	42/	36/	22	Yop La Boum v2.1	P.E.M & E.C.	149	37
6	41/	34/	25	Blue Blood	Ned Flanders	148	1
7	40/	33/	27	Keystone t21	P.Kline	147	79
8	43/	40/	17	Request v2.0	Brant D. Thomsen	147	60
9	44/	42/	13	Dragon Spear	c w blue	146	59
10	34/	22/	45	ttti	nandor sieben	146	63
11	40/	35/	25	Unknown	Ned Flanders	146	2
12	43/	43/	14	Iron Gate 1.5	Wayne Sheppard	144	90
13	44/	44/	12	Q2 version 2.0	Travis Nixon	143	4
14	34/	26/	39	CAPS KEY IS STUCK AGAIN	Steven Morrell	142	47
15	33/	26/	41	Blue Funk 88	Steven Morrell	141	6
16	40/	41/	19	Vanity II	Stefan Strack	140	81
17	44/	48/	9	Juggernaut v1.5	Anders Ivner	140	82
18	37/	36/	26	Killer instinct	Anders Ivner	138	84
19	40/	43/	17	Sauron v8.8	Michael Constant	137	36
20	35/	33/	33	FlyPaper 3.0	J.Layland	137	39

I've noticed that the '88 hill has turned into "beginners" hill of sorts. Most the hills that have contributed to the aging of this hill recently have been hanging around the bottom. This explains the absence of programs between the ages of 33 and 6. (Christoper was, I believe, the last program to move over from the original KotH hill.)

However, a couple of new programs have finally broken this trend. Q2 by Travis Nixon was right up at the top of the hill when it was first submitted, and Blue Blood by Ned Flanders is currently doing very well.

HINTS and HELPS:

This month's hint has been supplied by Ting-Yu Hsu. T. Hsu has been a powerful influence on the hill; Sasami has been in the top 5 on the '94 Pizza Draft Hill since it's submission over a month ago. I'm especially excited to be able to present this article, because replicators (also know as "paper" programs) are an area I am particularly weak in. (I know, I know, one of these days ...)

Creating a Competitive Paper

One of the warrior forms I have always enjoyed was the replicator. Unfortunately, replicators have been around so long that any good warrior (i.e., koth) can take one out. Thus, I took it upon myself to remedy this situation. Below is a side by side comparison of your typical paper and Ryooki, the only pure paper on the hill.

Line	Paper		Ryooki	
1	mov	#6 ,0	nop	>0 ,0
2	mov	<-1 ,<1	mov	<-1 ,{1
3	spl	@0 ,-3365	spl	@-3365,>800
4	mov	2 ,<-1	mov	3 ,{-1
5			mov	2 ,}25
6	jmz	-4 ,-4	jmz.f	*-4 ,*-4
7	dat	<2667,<2667*2	dat	<2667 ,<2667*2

Let's take the obvious addition first. Line 5 is an anti-vamp line. Look how easy this is to accomplish with A-field indirection. A more subtle application of line 5 is that it also serves to kill off older copies of itself. To understand this point, you must realize that 100 copies of paper running through core is more effective than 1000 copies of paper. Although paper is a replicator, every time it replicates the effectiveness of each

individual copy decreases even though the effectiveness of the whole increases. As more and more copies of paper appear in core, however, the decreasing effectiveness of each individual copy starts to outweigh the increasing effectiveness of the whole. This is best known as the point of diminishing returns. Ryooki attempt to combat this by killing off copies which are more than one generation old.

Probably the most important change is in line 6 (which forces the change in line 1). By performing a jmz.f, Ryooki vastly increases her chance of detecting a modification within her code. Additionally, by using indirection to perform the check and jump, Ryooki is actually using two lines to determine whether a modification occurred, not just one line. The change in line 1 is just a side effect of using jmz.f, since a "mov #7,0" would cause the jmz.f to consistently fail.

The final change is in line 3 (which forces the changes in lines 2 and 4). With the availability of A-field indirection, Ryooki is able to store its destination pointer in the A-field of the spl statement. Besides freeing up the B-field, which Ryooki uses to randomly trash core, it also gives the spl a chance to check whether the next copy of Ryooki is in good shape. That is, when Ryooki performs her spl, her spl pointer should be pointing to a "nop >0,0" instruction. If this is not the case, I do not care where Ryooki splits to as long as it is NOT where the spl is pointing to. This works under the assumption that jumping to a random location is better than jumping to a location which you know is corrupt.

Here is how the two replicators perform against warriors on the hill which I have code for (using mts and pmars -b -r 100 -F 2345). Some of the code is a bit out of date, but I didn't feel like browsing my mail for koth scores. (If someone out there could e-mail me more recent sources of warriors on the hill, I would appreciate it since I've been without a news feed for a few months now.)

Rank	Name	Author	%W	%L	%T	Score
1	Paper	Scott Nelson	18	38	43	1465
2	Request v2.0	Brant D. Thomsen	92	6	2	278
3	Iron Gate	Wayne Sheppard	89	7	4	271
4	SJ-4a	J.Layland	88	5	7	271
5	Stimpy v2.0	Brant D. Thomsen	86	6	8	266
6	Rave	Stefan Strack	78	10	12	246
7	Sasami	T.Hsu	41	7	52	175
8	Torch t5	P.Kline	40	13	47	167
9	Pyramid v2.0	Michael Constant	46	35	19	157
10	Blue Funk 3	Steven Morrell	7	13	80	101
11	Blue Funk	Steven Morrell	6	16	78	96
12	Cannonade	P.Kline	1	9	90	93
13	Aeka	T.Hsu	2	15	83	89
14	Insight v1.0	Brant D. Thomsen	1	13	86	89
15	NC 94	Wayne Sheppard	0	35	65	65
16	Keystone t33	P.Kline	0	81	19	19

Rank	Name	Author	%W	%L	%T	Score
1	Ryooki	T.Hsu	32	27	41	2051
2	Rave	Stefan Strack	87	10	3	264
3	Stimpy v2.0	Brant D. Thomsen	82	9	9	255
4	Iron Gate	Wayne Sheppard	76	16	8	236
5	SJ-4a	J.Layland	71	15	14	227
6	Torch t5	P.Kline	44	26	30	162
7	Sasami	T.Hsu	31	23	46	139
8	Blue Funk 3	Steven Morrell	7	16	77	98
9	Blue Funk	Steven Morrell	1	13	86	89
10	Insight v1.0	Brant D. Thomsen	1	14	85	88
11	Aeka	T.Hsu	0	13	87	87
12	Cannonade	P.Kline	1	28	71	74
13	NC 94	Wayne Sheppard	0	31	69	69
14	Request v2.0	Brant D. Thomsen	9	79	12	39
15	Keystone t33	P.Kline	0	90	10	10
16	Pyramid v2.0	Michael Constant	1	98	1	4

>From the rankings above, you should notice a few trends. First off, Ryooki beats up on the vampires due to her anti-vamp code, Ryooki does about the same against stones even though she is one line larger than paper, and finally, Ryooki tends to get more wins off programs designed to kill paper, like scanners and spl bombers.

Here are scores for Ryooki when I remove the anti-vamp line. As you can see, the 6 line version of Ryooki scores pretty much the same as the 7 line version. It does better against the stones and worse against the vampires which is to be expected. Notice the performances of Torch and Sasami, two spl bombing warriors. If you compare these scores against the ones from paper, you should notice how much better Ryooki is at surviving non-dat bombs.

Rank	Name	Author	%W	%L	%T	Score
1	Ryooki w/o anti-vamp	T.Hsu	27	30	43	1863
2	Rave	Stefan Strack	92	5	3	279
3	Stimpy v2.0	Brant D. Thomsen	83	7	10	259
4	Iron Gate	Wayne Sheppard	79	14	7	244
5	SJ-4a	J.Layland	78	13	9	243
6	Request v2.0	Brant D. Thomsen	47	50	3	144
7	Torch t5	P.Kline	29	25	46	133
8	Sasami	T.Hsu	20	20	60	120
9	Cannonade	P.Kline	1	14	85	88
10	Insight v1.0	Brant D. Thomsen	1	14	85	88
11	Blue Funk	Steven Morrell	4	21	75	87
12	Blue Funk 3	Steven Morrell	6	25	69	87
13	Aeka	T.Hsu	0	14	86	86
14	NC 94	Wayne Sheppard	0	34	66	66
15	Pyramid v2.0	Michael Constant	9	66	25	52
16	Keystone t33	P.Kline	0	84	16	16

Ryooki scored 28/22/50 when I first submitted it to koth, but since then her preferred targets, slow starting stones and vampires, have been knocked off the hill, so she is only hovering around 26/24/50 nowadays.

Finally, below in all her glory is the version of Ryooki which is currently on the hill. Just for informational purposes, Ryooki is a female rabbit (who meows like cat) from the Japanese anime "Tenchi Muyo".

T.Hsu
ting@teloquent.com

```
;------------------------------------------------------------------------------
;redcode-94
;name     Ryooki
;kill     Ryooki
;author   T.Hsu
;strategy Just a simple paper
;assert   CORESIZE == 8000 && MAXLENGTH >= 100 && VERSION >= 60
;macro
;------------------------------------------------------------------------------
;  1.0  Just an imp killing paper.
;  1.1  Use nop and jmz.f in the paper.
;  1.2  Use "mov 0,}0" instead of nop.  Use "spl @nxt" instead of "spl nxt".
;  1.3  Larger due to anti-vampire code.  Use labels instead of numbers.
;  1.4  Use "nop >0,0" instead of "mov 0,}0".

            org     boot_paper

nxt_paper   equ     -3365

boot_paper  spl     1 ,0
            spl     1 ,0
            mov.i   -1,#0

p_src       nop     >0        ,0            ; B-fld holds source
p_cpy       mov.i   <p_src    ,{p_dst
p_dst       spl     @nxt_paper,>800         ; A-fld holds destination
```

```
            mov.i   p_bomb    ,{p_dst       ; anti-imp  instruction
            mov.i   p_bomb    ,}25          ; anti-vamp instruction
            jmz.f   *p_cpy    ,*p_cpy
p_bomb      dat     <2667     ,<2667*2      ; bomb designed to kill imps

cnt         for     90
            dat     0,0
            rof
```

Looking to the Future:

I hope you enjoy the new format of the Warrior, and that you find the next one to be worth the month-long wait. As always, your comments, suggestions, criticisms, and submissions are encouraged and appreciated.

```
  __ __| |           ) _ \  |  |    \ \        /             _)
     |   __ \   _ \  / (   | |  |     \ \  \   / _` |  __|  __| |  _ \   __|
     |   | | |  __/   \__  |___ __|    \ \  \ / (   | |     |    | (   | |
    _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|    _|   _|\___/ _|
```

August 18, 1994 Issue #12

The’94_Warrior_ is a monthly newsletter dedicated to supporting and encouraging the game of Corewars. This includes coverage of the more popular "hills" (which allow users anywhere on the Internet to compete against other users), and also by encouraging traffic in the rec.games.corewar newsgroup.

The FAQ (Frequently Asked Questions) for the rec.games.corewar newsgroup is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z. There are also several tutorials on the ’88 and ’94 (draft) standard available on soda.csua.berkeley.edu that will greatly ease the process of becoming a proficient "redcoder." I would highly recommend referring to the FAQ if you are curious about exactly what Corewars is, or if you are unfamiliar with any of the terms used in this newsletter.

CHANGES and CORRECTIONS:

For those of you using a Macintosh to develop your redcode wonders, a new version of MacpMARS is available on soda.csua.berkeley.edu. Look in the "pub/corewar/incoming" directory for it.

At the request of several readers, I have added the ’94 beginner hill to the list of hills I will be covering in this and future issues of _The_’94_Warrior_.

The ICWS ’94 Draft Hill:

```
        Standard:       ’94 Draft (with extensions)
      Core size:        8000 instructions
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	34/	12/	54	B-Panama X	Steven Morrell	156	28
2	49/	42/	9	Agony II	Stefan Strack	156	22
3	42/	30/	29	Torch t8	P.Kline	154	1
4	34/	16/	50	Silk Warrior 1.4	J.Pohjalainen	152	29
5	36/	26/	39	Blue Funk 3	Steven Morrell	146	87
6	44/	44/	13	Iron Gate 1.5	Wayne Sheppard	143	376
7	41/	39/	20	SJ-4	J.Layland	143	88
8	35/	28/	37	Phoenix 1.1	J.Pohjalainen	141	24
9	38/	35/	26	Stimpy v2.0	Brant D. Thomsen	141	128
10	30/	20/	49	Test	Wayne Sheppard	141	7
11	30/	21/	49	Aeka	T.Hsu	139	42
12	36/	34/	29	Homemade Ice Cream	P.Kline	138	64
13	37/	36/	27	IVScan 8000	James Layland	138	50
14	41/	45/	14	Betrave & Rubarbe	J.P.Laurin	138	36
15	32/	27/	41	Sasami	T.Hsu	137	112
16	31/	26/	43	Blue Funk	Steven Morrell	137	399
17	39/	42/	18	Keystone t33	P.Kline	136	18
18	30/	25/	45	Cannonade	P.Kline	136	257
19	40/	45/	14	Futility	M R Bremer	135	21
20	37/	47/	16	Pyramid v5.5	Michael Constant	128	177

The hill has been taken over by replicators. Silk Warrior 1.4 and B-Panama X are both putting up strong fights for the top position. An ironic twist resulting from this "paper invasion" is how poorly vampires do on the hill as a result. Christopher and Request have disappeared entirely from the hill, and the last remaining vampire, Pyramid, has a very uncertain future. (It was in second place on this hill just a month ago.) The addition of A-field

indirection has made it much easier to add anti-vampiric elements to programs -- and it looks like the Rock/Scissors/Paper analogy has gone out the window because of it. Perhaps now would be a good time to dust off your old scanners and see how well they can do.

Among the older programs to disappear from the hill are NC 94 by Wayne Shepherd at the age of 387, Lucky 3 by Stefan Strack at the age of 350+ (I forgot to record it exactly), and Christopher by Steven Morrell at the age of 289.

Congratulations also to Steven Morrell on having his program Blue Funk reach the age of 400. (It should be at least that old when he reads this.) I believe Blue Funk is the first program to reach that age on any hill since the original KotH hill.

The ICWS ’94 Draft Experimental Hill:

Standard:	’94 Draft (with extensions)
Core size:	55,440 instructions
Max processes:	10,000 per program
Duration:	After 500,000 cycles, a tie is declared.
Max entry length:	200 instructions

The current ICWS ’94 Experimental (Big) hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	48/	33/	20	ivscan6b	J.Layland	162	29
2	46/	35/	19	Request-55440	Brant D. Thomsen	158	165
3	34/	17/	49	Aleph 1	Jay Han	151	27
4	33/	15/	52	Big Silk Warrior 1.0	J.Pohjalainen	151	7
5	44/	38/	18	Pyramid v5.3	Michael Constant	150	56
6	36/	23/	41	Variation G-1	Jay Han	148	129
7	42/	39/	18	Fscan	Jay Han	146	13
8	41/	37/	23	Stimpy v2.0	Brant D. Thomsen	145	20
9	40/	39/	21	Aleph 0	Jay Han	142	28
10	38/	39/	23	Vanity IIx	Stefan Strack	138	120
11	33/	28/	39	Lucky 13	Stefan Strack	138	171
12	43/	48/	10	Test	Stefan Strack	138	1
13	30/	23/	47	NotSoBigImps	James Layland	137	25
14	42/	48/	10	Rave B4.1	Stefan Strack	137	126
15	30/	24/	46	Der Zweite Blitzkrieg - 9	Mike Nonemacher	136	127
16	30/	25/	44	Splash 1	Jay Han	136	130
17	29/	24/	47	Blue Funk	Steven Morrell	135	19
18	35/	39/	26	Lump	J.Layland	131	110
19	39/	48/	12	Squint	Mike Nonemacher	130	103
20	25/	24/	51	Insight v1.0	Brant D. Thomsen	126	8

It looks like replicators may soon be taking over the "Big" hill, just like they have the ’94 draft hill; Big Silk Paper is close to the top of the hill. It will be interesting to see if paper programs will remove Request and Pyramid from their lofty positions, or if vampires will continue to be an effective programming technique on the "Big" hill.

The ICWS ’88 Standard Hill:

Standard:	’88
Core size:	8000 instructions
Max processes:	8000 per program
Duration:	After 80,000 cycles, a tie is declared.
Max entry length:	100 instructions

The current Standard KotH hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	36/	22/	42	Sphinx v5.1	W. Mintardjo	151	1
2	33/	16/	51	ttti	nandor sieben	149	71
3	35/	24/	41	Der Zweite Blitzkrieg	Mike Nonemacher	146	49
4	34/	22/	45	The Plauge	Anonymous	146	4
5	39/	32/	29	Yop La Boum v2.1	P.E.M & E.C.	146	45
6	33/	22/	45	NC Killer	Anonymous	145	5

7	31/	20/	49	Blue Funk 88	Steven Morrell	143	14
8	40/	37/	23	Christopher	Steven Morrell	143	41
9	32/	21/	47	CAPS KEY IS STUCK AGAIN	Steven Morrell	142	55
10	38/	35/	27	Keystone t21	P.Kline	142	87
11	40/	40/	19	Request v2.0	Brant D. Thomsen	141	68
12	32/	23/	45	Night Crawler	Wayne Sheppard	140	77
13	40/	39/	21	Vanity II	Stefan Strack	140	89
14	41/	44/	15	Armor	c w blue	139	3
15	42/	44/	14	Iron Gate 1.5	Wayne Sheppard	139	98
16	41/	44/	14	Dragon Spear	c w blue	139	67
17	39/	43/	18	SJ-4a	J.Layland	136	48
18	29/	23/	48	Imprimis 6	Anonymous	135	6
19	36/	39/	25	Unknown	Ned Flanders	134	10
20	36/	39/	25	Blue Blood	Ned Flanders	133	9

A few new programs have made things interesting on the ’88 hill. W. Mintardjo resubmitted the latest version of his classic program Sphinx, and it jumped right to the top of the hill. Many other classic programs have also been submitted anonymously, and three of them seem to have caught hold. There aren’t many new faces on this hill, but the ordering has dramatically changed.

The ICWS ’94 Beginner Hill:

Standard:	’94 Draft (with extensions)
Core size:	8000 instructions
Max processes:	8000 per program
Duration:	After 80,000 cycles, a tie is declared.
Max entry length:	100 instructions

The current Beginner KotH hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	61/	7/	32	Rabid Dwarfs v1.0	Brian Zellner	215	3
2	45/	26/	29	Rubarbe et mort-vivant	J.P.Laurin	163	11
3	52/	42/	6	Rubarbe et frappe	J.P.Laurin	163	13
4	52/	42/	6	Rubarbe et frappe	J.P.Laurin	161	12
5	42/	26/	32	Slate, v0.3a	Bharat Mediratta	157	62
6	40/	32/	28	Rubarbe et grafigne	J.P.Laurin	149	10
7	45/	43/	12	Viewmaster4.0	Ryan Case	148	141
8	43/	39/	17	Bloody Fang v2.0	Bharat Mediratta	147	118
9	40/	34/	26	Count Zero #1	Marc Schefer	145	4
10	44/	43/	14	Viewmaster 4.2	Ryan Case	145	127
11	39/	34/	28	Bloody Fang v2.1	Bharat Mediratta	144	1
12	38/	34/	28	Bloody Fang v2.1	Bharat Mediratta	143	2
13	43/	44/	13	Viewmaster 4.2a	Ryan Case	143	128
14	40/	40/	20	Slate, v0.3b	Bharat Mediratta	141	61
15	38/	35/	27	Bloody Fang v2.1	Bharat Mediratta	140	109
16	43/	50/	7	It	Hari	136	125
17	40/	49/	11	Gross 1.8	Osric	132	120
18	40/	48/	12	Another Fanging Vampire 2	Matt Jaques	131	175
19	41/	53/	5	Ecstacy (XTC)	Brant Thomsen	129	8
20	37/	50/	13	Cat v2.0	Tim Scheer	124	82

As I have only been watching this hill for a couple of weeks, I can’t really give you a summary of what is happening on it. However, I must admit to being surprised at the number of programs that are constantly being submitted to this hill. The ages of the current programs don’t reflect this, since most of the submittals never make it onto the hill, but there is still a steady stream of traffic. (In this regard, the beginner’s hill is very much the same as the ’88 hill covered above.)

If I could pass along some advice to the players currently on this hill, it would be to use the ";kill" directive to get rid of some of the duplicate copies of warriors on this hill. This will usually cause your remaining warrior to do better, as programs it does well against are not penalized as much, and it will make it easier for other programmers -- or the new program you are developing -- to get onto the hill by freeing up some slots.

HINTS and HELPS:

This month's hint has been supplied again by Ting-Yu Hsu. For those of you who have been dying of curiosity about exactly what a "vector-launched imp" is, you are about to find out!

Vector Launched Imps

When I first got back into Corewars after a 3 year hiatus, I figured that it would be trivial to place a warrior on the hill. After all, when I first played it, it took almost no thought to place a warrior on the top of the Intel hill. Boy, was I in for a shock. I ran into a wall that all newcomers to the hill quickly run into, the imp.

After going through a period of denial, I decided to get down and just learn what the hell an imp was and how it worked. In the process, I learned one major thing. I had NO desire to write binary launch code. This meant one of two things. I could write a C program to write the redcode for me, or I had to figure out a new way to launch an imp. Since I was revulsed by the idea of writing redcode via a C program, I took the latter (i.e., insane) approach.

Here were the requirements I set for myself.
1. The imp launch code must be easily written and understandable.
2. The imp launch code must be as fast or almost as fast as a binary launch.
3. The imp launch code must be shorter than a binary launch.

What I came up with, after quite a bit of trial and error, was the vector launched imp. A vector launch boots an imp in O(2*N+C) cycles and is O(log2(N-1)+(N/2)+1) in length. The first example below is a "perfect" vector launch, i.e., it is exactly as fast as a binary launch and yet it is smaller and easier to read. The second example shows an alternate method of vector launching a smaller imp.

```
;redcode-94
;name   Vector
;author T.Hsu

imp_sz  equ     2667

boot    spl     1       ,#0
        spl     1       ,#0
        spl     <0      ,#vector+1
        djn.a   @vector,#0

imp     mov.i   #0,imp_sz

        jmp     imp+imp_sz*7,imp+imp_sz*6   ; normally this is in a for/rof
        jmp     imp+imp_sz*5,imp+imp_sz*4   ; loop, but I wanted to make the
        jmp     imp+imp_sz*3,imp+imp_sz*2   ; order of the vectors obvious
vector  jmp     imp+imp_sz  ,imp

        end     boot
```

```
;redcode-94
;name   Single Vector
;author T.Hsu

imp_sz  equ     2667

        dat     0       ,#imp+imp_sz*3
boot    spl     1       ,#imp+imp_sz*2
        spl     1       ,#imp+imp_sz
vector  djn.a   @vector,#imp

imp     mov.i   #0,imp_sz

        end     boot
```

        :

```
vector  djn.a   @vector,#imp
        :
```

Shown above is the crucial instruction in making the vector launch work. There are a few peculiarities about this instruction. First off, you cannot directly follow this instruction with the vector table. This is because the "djn.a" will evaluate to zero, and thus no jump would occur. Second, you should notice that the B-field of this instruction evaluates to zero as part of the instruction. Therefore, you can put it to good use as a pointer (in the second example above, I used it as the first imp vector). And third, this is a good example of the difference between in-memory evaluation and in-register evaluation.

See how the jump point is stored into a register before the decrement and test occurs. Thus, our jump point is not affected by the decrement and test. Before the in-memory/in-register clarification by the ’94 rules, it would have been ambiguous whether a simulator would decrement/test/load_jump_pt/jump or load_jump_pt/decrement/test/jump. Obviously the latter is the correct interpretation as stated by the ’94 rules.

One final thing you should notice is that non-power of 2 imps will waste cycles while launching and that non-multiple of 2 imps are a bit tricky to put together. This is because the "spl" statement is basically optimized for power of 2 process generation. Below are side-by-side examples of the "spl" instructions necessary to create a 6 point launch and a 7 point launch.

```
    ; 6 point imp           |   ; 7 point imp
        :                   |       :
        spl     1     ,#0   |       spl     1     ,#0
        mov.i   -1    ,#0   |       spl     1     ,#0
        spl     <0    ,#vec |   mm  mov.i   2     ,#0
        djn.a   @vec-1,#0   |       djn.a   @vec-1,#0
        :                   |       spl     <0    ,#vec-mm
        :                   |       :
```

The 6 point launch wastes 1 cycle performing the "mov" instruction and the 7 point launch wastes both a cycle and an instruction. This basically means that you should try to launch power of 2 spirals whenever possible, and if that is not possible, you should try to minimize the number of "mov" instructions necessary during the process creation. Also notice that the 7 point launch requires a few modifications in the vector table since the B-field is accessed twice before an A-field is accessed.

Although I happen to be partial to "djn.a", it is not the only construct available to someone writing vector launch code. The availability of A-field indirection allows a whole host of other possibilities which I haven’t even begun to explore.

Below is Aeka, the first warrior I created using a vector launched imp. The stone and the boot code associated with the stone were taken almost directly from Cannonade by P.Kline. I have changed these constants, but the changes are currently not on koth, so the point is moot.

Aeka uses quite a few tricks within her code, but I’m not going to explain them here, since the main focus is on the vector launch code. As for some of the weird constants, well, Aeka tends to kill off a few of her imps during her own core clear, so I had to get really creative in the constants in order to minimize this problem.

Oh yeah, Aeka is an alien princess (who, of course, looks perfectly human) in the Japanese anime "Tenchi Muyo". Just in case you were wondering where the name comes from.

T.Hsu
ting@teloquent.com

```
;redcode-94
;name     Aeka
;kill     Aeka
;author   T.Hsu
```

```
;strategy Suicidal stone & vector launched, gate busting imp spiral
;assert   CORESIZE == 8000 && MAXLENGTH >= 100
;macro
;-----------------------------------------------------------------------------
;  1.0  Original based on Cannonade by P.Kline
;  1.1  Erase pointer to stone, better decoy, use immediate mode more
;  1.2  Reposition imps so that they don't cause djn.a to fall through
;  1.3  Use a decoy that looks like a pointer, put boot ptrs in unused fields
;  1.4  Use A-field indirection to shorten imp launch code
;  1.5  Use "spl <0,#vector" instead of "djn.a *(vector-1),#0"
;       Split before finishing to boot the gate busting imps, better constants
;  1.6  Back to "djn.a *(vec-1),#0" to be less vulnerable to quick-scanners
;       Move around decoy code & launch vectors, launch normal imps first
;  1.7  Back to "spl <0,#vec" to save on space, no decoy
;       More redundancy in for the imps
;  1.8  Compress boot code for the gate busting imp
;  1.9  Made B-field of "djn.a" do double duty as a boot pointer
;       Made dec_offset help gate against imps during core-clear
;  2.0  Better use of for/rof in the vector launch
;
;  Vulnerabilities
;  -- hyper-perfect gates (gate busting imps have a tough time with these)
;  -- anti-imp paper (the stone is not designed to stun/kill paper)
;  -- suicidal stone (if gate busting imp dies, we don't want stone to die)
;  -- quick scanners (due to long boot time and use of "spl <0,#vec")

imp_sz01    equ     2668
imp_sz02    equ     imp_sz01
imp_sz03    equ     2667
imp_prc01   equ     8
imp_prc02   equ     imp_prc01
imp_prc03   equ     10
imp_off01   equ     -2
imp_off02   equ     0
imp_off03   equ     -7
imp_first   equ     (start-1834)+2*imp_sz02
stone_inc   equ     190
stone_offst equ     701
dec_offst   equ     (imp_sz03*2)-stone_inc

            org     start
;-----------------------------------------------------------------------------
;  Boot strap

start       mov.i   imp_2,imp_first+imp_off02+2 ; gate busting imp
            mov.i   imp_3,imp_first+imp_off03   ; normal imp
            mov.i   <stone_src,@stone_dst2      ; stone
            mov.i   <stone_src,<stone_dst       ;   put B-field of "djn.a"
            mov.i   <stone_src,<stone_dst       ;   to use as the stone_src
            mov.i   <stone_src,<stone_dst
            spl     @stone_dst,<dec_offst
            mov.i   <stone_src,<stone_dst
;-----------------------------------------------------------------------------
;  Vector launch the imps

imp_split   spl     1,<dec_offst                ; 26 processes
            spl     1,<dec_offst
            mov.i   imp_2,<start                ; finish booting imp
stone_dst   mov.i   -2,#stone_end+1-stone_offst     ;\ notice how these ptrs
stone_dst2  mov.i   -1,#stone_end+1-(stone_offst-1) ;/ are conviently erased
            spl     <0,#imp_vector               ;\ decrement self to launch
stone_src   djn.a   @(imp_vector-1),#stone_end+1 ;/ imps, B-fld before A-fld
;-----------------------------------------------------------------------------
;  Self splitting stone and core clear

stone       mov.i   <stone_spl+5+stone_inc*800,stone_spl
stone_spl   spl     stone,<dec_offst+stone
            add.f   stone_end+1,stone
            djn.f   stone_spl,<dec_offst+stone
stone_end   mov.i   stone_inc,<-stone_inc
```

```
;-------------------------------------------------------------------------
;  Decoy

cnt         for     65
            dat     0,0
            rof
;-------------------------------------------------------------------------
;  Launch vectors

imp_2       mov.i   #(imp_sz02/2),imp_sz02
imp_3       mov.i   #(imp_sz03/2),imp_sz03

imp_A_fld   equ     imp_first+(imp_prc&who+1-2*cnt)*imp_sz&who+imp_off&who
imp_B_fld   equ     imp_first+(imp_prc&who+0-2*cnt)*imp_sz&who+imp_off&who
who         for     3
cnt         for     (imp_prc&who)/2
            jmp     imp_A_fld,imp_B_fld
            rof
            rof
imp_vector
            end
```

EDITORS NOTE:

Vector-launched imp-spirals can be cause a significant reduction in the number of lines your imp-launching code will take. A 16-point imp-spiral vector launch code requires less than half the number of lines required for the corresponding binary launch code, and a 64-point imp-spiral requires less than a third the number that a binary launch would need. With quick scanners being so common on the hill, a few less lines can make a big difference.

However, there still are reasons that binary imp-spiral launch code will not be disappearing soon. Probably the most obvious is that the ’88 standard is still going strong. Binary launches are also more flexible, as they allow the programmer can choose later times to send processes to other parts of the program (such as stones). And, of course, one final advantage of binary imp-spiral launches is that they are impervious to DJN.B streams.

Thanks again to Ting-Yu Hsu for sharing a hint which I am certain will have a significant influence on the hill in the future.

Looking to the Future:

This newsletter is now being published monthly. As always, your comments, suggestions, criticisms, and submissions are encouraged and appreciated.

From news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!howland.reston.ans.net!agate!dog.ee.lbl.gov!news.cs.utah.edu!peruvian.cs.utah.edu!bdthomse Wed Nov 2 11:08:45 1994
Article: 1206 of rec.games.corewar
Path: news-rocq.inria.fr!univ-lyon1.fr!jussieu.fr!math.ohio-state.edu!howland.reston.ans.net!agate!dog.ee.lbl.gov!news.cs.utah.edu!peruvian.cs.utah.edu!bdthomse
From: bdthomse@peruvian.cs.utah.edu (Brant Thomsen)
Newsgroups: rec.games.corewar
Subject: The ’94 Warrior
Date: 1 Nov 1994 05:56:52 GMT
Organization: University of Utah CS Dept
Lines: 230
Distribution: world
Message-ID: <394lb4$ko4@magus.cs.utah.edu>
NNTP-Posting-Host: peruvian.cs.utah.edu
Originator: bdthomse@peruvian.cs.utah.edu

```
  __ __| |            ) _ \  |  |    \ \        /              _)
     |   __ \   _ \  / (   | |  |     \ \  \   / _‘ |  __|  __| |  _ \   __|
     |   | | |  __/   \__  |___ __|    \ \  \ / (   | |    |    | (   | |
    _|  _| |_|\___|      _/    _|       \_/\_/ \__,_|_|   _|   _|\___/ _|
```

October 31, 1994 Issue #13

The’94_Warrior_ is a monthly newsletter dedicated to supporting and encouraging the game of Corewars. This includes coverage of the more popular "hills" (which allow users anywhere on the Internet to compete against other users), and also by encouraging traffic in the rec.games.corewar newsgroup.

The FAQ (Frequently Asked Questions) for the rec.games.corewar newsgroup is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z. There are also several tutorials on the ’88 and ’94 (draft) standard available on scotch.csua.berkeley.edu (128.32.43.51) that will greatly ease the process of becoming a proficient "redcoder." I would highly recommend referring to the FAQ if you are curious about exactly what Corewars is, or if you are unfamiliar with any of the terms used in this newsletter.

CHANGES and CORRECTIONS:

There are several changes that have occurred in the last two months. Please remind me of any that I may have missed, and I’ll be sure to get them in the next issue of _The_’94_Warrior_.

As you may have noticed above, the staff here at _The_’94_Warrior_ has been working diligently to print the correct Corewar Archive site. The correct address is now "scotch" instead of "soda". Accessing "ftp.csua.berkeley.edu" should still work as well.

The fall TCWN newsletter is now available on scotch as pub/corewar/documents/tcwn0994.z. I’d like to highly recommend it, but since I don’t have a PostScript printer, I’m afraid I can’t. (Where is a copy of "GhostScript" when I need it?!) However, if this issue is anything like the previous ones, I can promise that you are in for a real treat. Kudos to Mark Durham once again!

For those of you that like an occasional break from writing redcode, a C++ Robots Server is now up and running. Send mail with the subject "help summary c++robots" (without the quotes) to pbmserv@netcom.com for more information, and be sure to watch the newsgroup for more posts by Richard Rognlie about this exciting event.

Another exciting thing to watch the newsgroup for is Paul Kline’s status listings on the "hot" hills. For those that keep having their warriors kicked off the hills (such as myself), this will be a nice way to find out what is currently happening. Perhaps it will also cut down the number of times that infamous "Just Looking" warrior gets sent to the hill!

On Stormking, the hills have been enhanced. Now, when you are notified about a warrior on one of those hills, the subject line will tell you who it is you are going to be reading about. Several more enhancements are expected to follow as well, so be sure to watch the newsgroup for more information.

The August and September archives for the rec.games.corewar newsgroup have also been uploaded to scotch. Look in the "incoming" directory.

The ICWS '94 Draft Hill:

```
        Standard:       '94 Draft (with extensions)
      Core size:        8000 instructions
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS '94 Draft hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	46/	37/	17	HeremScimitar	A.Ivner,P.Kline	156	27
2	43/	31/	25	Torch t15	P.Kline	155	7
3	36/	24/	40	Blue Funk 3	Steven Morrell	149	208
4	38/	30/	32	Rude Wind	P.Kline	147	57
5	31/	15/	54	B-Panama X	Steven Morrell	147	149
6	45/	44/	11	Agony II	Stefan Strack	147	143
7	33/	23/	45	Blue Funk	Steven Morrell	143	520
8	31/	20/	48	Silk Warrior 1.4	J.Pohjalainen	142	150
9	35/	31/	34	Killer Instinct II	Anders Ivner	140	2
10	39/	38/	23	Leprechaun II	Anders Ivner	139	1
11	39/	40/	21	SJ-4	J.Layland	139	209
12	41/	44/	15	Iron Gate 1.5	Wayne Sheppard	138	497
13	37/	37/	26	Stimpy v2.0	Brant D. Thomsen	138	42
14	29/	21/	50	Phoenix 1.2	J.Pohjalainen	137	100
15	28/	22/	51	Aeka	T.Hsu	134	163
16	31/	29/	40	Sasami	T.Hsu	133	233
17	38/	44/	17	Test	Wayne Sheppard	133	6
18	27/	23/	50	Bremstone 2.X	M R Bremer	132	112
19	27/	22/	51	Cannonade	P.Kline	132	378
20	37/	43/	20	The Spanish Inquisition	Steven Morrell	132	36

The big leaders on the hill are Torch and HeremScimitar. Scimitar (by Paul Kline) used some interesting tricks to take the hill by storm when it was introduced a couple of weeks ago.

Last tile I examined the hill, replicators were the dominant force. Now it appears that stones/bombers have the top positions. Of course, the hill is 115 battles older, so quite a bit has happened since then.

Congratulations are also due for Steven Morrell, as his first Blue Funk warrior passes the 500 age mark.

The ICWS '94 Draft Experimental Hill:

```
        Standard:       '94 Draft (with extensions)
      Core size:        55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS '94 Experimental (Big) hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	46/	32/	22	ivscan6b	J.Layland	161	35
2	47/	36/	17	Genocide 94x	Mike Nonemacher	158	1
3	34/	16/	50	Big Phoenix 1.0	J.Pohjalainen	152	6
4	34/	18/	48	Aleph 1	Jay Han	150	33
5	44/	38/	19	Request-55440	Brant D. Thomsen	149	171
6	42/	36/	22	Stimpy v2.0	Brant D. Thomsen	148	26
7	46/	44/	10	Test	Stefan Strack	147	7
8	43/	41/	16	Pyramid v5.3	Michael Constant	145	62

9	44/	45/	11	Rave B4.1	Stefan Strack	144	132
10	30/	16/	54	Big Silk Warrior 1.0	J.Pohjalainen	143	13
11	34/	25/	41	Variation G-1	Jay Han	142	135
12	38/	39/	23	Vanity IIx	Stefan Strack	137	126
13	39/	42/	19	Fscan	Jay Han	136	19
14	38/	41/	21	Aleph 0	Jay Han	136	34
15	30/	27/	43	Splash 1	Jay Han	133	136
16	29/	26/	45	Blue Funk	Steven Morrell	133	25
17	29/	25/	47	NotSoBigImps	James Layland	132	31
18	40/	49/	11	Squint	Mike Nonemacher	131	109
19	31/	31/	38	Lucky 13	Stefan Strack	131	177
20	29/	27/	44	Der Zweite Blitzkrieg - 9	Mike Nonemacher	131	133

Although ivsan6b is still on top of the hill, it now has some new competition. Genocide and Big Phoenix are two new warriors that have jumped right onto the hill. Genocide is a F-scanner, and Big Phoenix is almost a paper.

The ICWS ’88 Standard Hill:

Standard:	’88
Core size:	8000 instructions
Max processes:	8000 per program
Duration:	After 80,000 cycles, a tie is declared.
Max entry length:	100 instructions

The current Standard KotH hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	43/	28/	29	Yop La Boum v2.1	P.E.M & E.C.	158	55
2	33/	22/	45	Sphinx v5.1	W. Mintardjo	145	11
3	33/	23/	44	Der Zweite Blitzkrieg	Mike Nonemacher	144	59
4	44/	43/	13	Dragon Spear	c w blue	144	77
5	31/	20/	49	CAPS KEY IS STUCK AGAIN	Steven Morrell	142	65
6	29/	15/	56	ttti	nandor sieben	142	81
7	31/	20/	49	The Plauge	Anonymous	141	14
8	40/	40/	19	Vanity II	Stefan Strack	141	99
9	43/	45/	12	Iron Gate 1.5	Wayne Sheppard	141	108
10	30/	20/	49	NC Killer	Anonymous	140	15
11	39/	40/	21	Request v2.0	Brant D. Thomsen	139	78
12	29/	20/	51	Night Crawler	Wayne Sheppard	139	4
13	29/	19/	52	Blue Funk 88	Steven Morrell	138	24
14	38/	39/	22	Christopher	Steven Morrell	138	51
15	31/	25/	44	B-Panama V.i	Steven Morrell	136	9
16	35/	35/	30	Annihilator v2.1	Carlos Paredes	136	2
17	36/	37/	27	Keystone t21	P.Kline	135	97
18	27/	20/	53	Imprimis 6	Anonymous	134	16
19	37/	41/	22	Titled	Steven Morrell	132	1
20	37/	44/	19	SJ-4a	J.Layland	129	58

The ’88 hill hasn’t aged much (10), but don’t let that fool you. This hill is still very busy. I really enjoy watching as warriors try to make the transition from the beginner hill to the standard hill. Good luck!

The ICWS ’94 Beginner Hill:

Standard:	’94 Draft (with extensions)
Core size:	8000 instructions
Max processes:	8000 per program
Duration:	After 80,000 cycles, a tie is declared.
Max entry length:	100 instructions

The current Beginner KotH hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	56/	34/	10	Samba2.2	Mr. Jones	179	13
2	54/	35/	11	Viewmaster4.0	Ryan Case	172	214
3	53/	40/	7	Ecstacy (XTC)	Brant Thomsen	166	81
4	34/	10/	55	Rabid Dwarfs v1.3	Brian Zellner	158	56
5	42/	27/	31	Ox v1.0	Brian Zellner	158	10
6	40/	36/	24	Bloody Fang v2.1	Bharat Mediratta	145	1

7	36/ 34/ 31	Green Knight v1.7	Brian Zellner	137	57
8	30/ 24/ 46	Slate, v0.3a	Bharat Mediratta	136	135
9	25/ 17/ 58	Paperous	Mr. Jones	133	6
10	31/ 31/ 38	Count Zero #1	Marc Schefer	131	77
11	27/ 23/ 50	Rubarbe et mort-vivant	J.P.Laurin	130	84
12	27/ 24/ 48	War Machine v2.65	Matthias Wollnik	130	2
13	27/ 24/ 49	War Machine v2.6	Matthias Wollnik	129	3
14	37/ 52/ 11	Cat v2.0	Tim Scheer	122	155
15	24/ 27/ 49	War Machine v2.5	Matthias Wollnik	121	4
16	35/ 53/ 12	It	Hari	117	198
17	21/ 25/ 54	Mostly Harmless	Frank J. T. Wojcik	117	45
18	21/ 31/ 47	The Foo Bar and Grill	James Jesensky	111	7
19	27/ 45/ 29	Rubarbe et grafigne	J.P.Laurin	109	83
20	33/ 62/ 5	Stepper v1.0	Gollum	104	37

Although I haven’t been keeping an accurate count, I can promise that this is the busiest of the four hills that I watch. Many of the warriors submitted don’t make it, but there certainly are many attempts.

HINTS and HELPS:

As I have been without a (home) computer for the last couple of months, I’m afraid I haven’t had the time to put together a hint without delaying this newsletter another week or so. Of course, next month’s hint will be twice as good to make up for it!

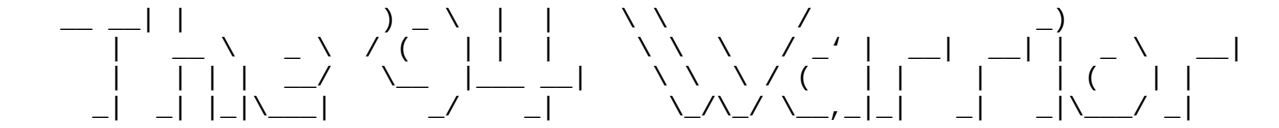
Looking to the Future:

This newsletter is now (again) being published monthly. As always, your comments, suggestions, criticisms, and submissions are encouraged and appreciated.

--

Brant D. Thomsen
(bdthomse@peruvian.cs.utah.edu)
University of Utah

Man will occasionally stumble over the truth,
but most times he will pick himself up
and carry on. - Winston Churchill



November 30, 1994 Issue #14

The’94_Warrior_ is a monthly newsletter dedicated to supporting and encouraging the game of Corewars. This includes coverage of the more popular "hills" (which allow users anywhere on the Internet to compete against other users), and also by encouraging traffic in the rec.games.corewar newsgroup.

The FAQ (Frequently Asked Questions) for the rec.games.corewar newsgroup is available through anonymous FTP to rtfm.mit.edu, as /pub/usenet/news.answers/games/corewar-faq.Z. There are also several tutorials on the ’88 and ’94 (draft) standard available on scotch.csua.berkeley.edu (128.32.43.51) that will greatly ease the process of becoming a proficient "redcoder." I would highly recommend referring to the FAQ if you are curious about exactly what Corewars is, or if you are unfamiliar with any of the terms used in this newsletter.

CHANGES and CORRECTIONS:

It looks as if we may soon have a version of pMARS that is capable of handling several warriors at the same time. (I, for one, would be very interested in seeing what types of warriors do well in that type of environment. :)

The FAQ for the newsgroup has also been updated. Be sure to save a copy of the new one for your collection!

The ICWS ’94 Draft Hill:

```
        Standard:       ’94 Draft (with extensions)
       Core size:       8000 instructions
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current ICWS ’94 Draft hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	50/	39/	10	Taking names	P.Kline	161	9
2	45/	31/	24	Torch t15	P.Kline	159	43
3	44/	35/	21	Stimpy 3.0	Brant D. Thomsen	154	15
4	46/	38/	16	HeremScimitar	A.Ivner,P.Kline	153	63
5	47/	42/	10	Agony II	Stefan Strack	152	179
6	40/	28/	32	Blue Funk 3	Steven Morrell	151	244
7	33/	17/	50	B-Panama X	Steven Morrell	148	185
8	43/	38/	19	SJ-4	J.Layland	147	245
9	39/	32/	30	Rude Wind	P.Kline	146	93
10	36/	28/	36	Blue Funk	Steven Morrell	145	556
11	41/	38/	22	Leprechaun II	Anders Ivner	144	37
12	43/	44/	14	Iron Gate 1.5	Wayne Sheppard	141	533
13	41/	42/	18	Drowning III	Mike Nonemacher	140	26
14	29/	20/	51	Ryooki	T.Hsu	138	31
15	30/	24/	46	Silk Warrior 1.4	J.Pohjalainen	137	186
16	39/	43/	18	The Spanish Inquisition	Steven Morrell	134	72
17	34/	34/	32	Killer Instinct II	Anders Ivner	133	38
18	32/	31/	37	Jet Black Reptile	M R Bremer & Jippo	133	2
19	33/	51/	17	Grasp v0.1d	Brant D. Thomsen	115	5

Of course, the most obvious change to this hill is the addition of Paul Kline’s new warrior "Taking Names," which has been clinging to first place since its submission.

("Stimpy" has also jumped up several positions since I tacked a quick-scanner onto the front of it. Check out the hint section of this newsletter if you’d like more details! ;)

The ICWS ’94 Draft Experimental Hill:

```
        Standard:       ’94 Draft (with extensions)
       Core size:       55,440 instructions
   Max processes:       10,000 per program
        Duration:       After 500,000 cycles, a tie is declared.
Max entry length:       200 instructions
```

The current ICWS ’94 Experimental (Big) hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	46/	32/	22	ivscan6b	J.Layland	161	35
2	47/	36/	17	Genocide 94x	Mike Nonemacher	158	1
3	34/	16/	50	Big Phoenix 1.0	J.Pohjalainen	152	6
4	34/	18/	48	Aleph 1	Jay Han	150	33
5	44/	38/	19	Request-55440	Brant D. Thomsen	149	171
6	42/	36/	22	Stimpy v2.0	Brant D. Thomsen	148	26
7	46/	44/	10	Test	Stefan Strack	147	7
8	43/	41/	16	Pyramid v5.3	Michael Constant	145	62
9	44/	45/	11	Rave B4.1	Stefan Strack	144	132
10	30/	16/	54	Big Silk Warrior 1.0	J.Pohjalainen	143	13
11	34/	25/	41	Variation G-1	Jay Han	142	135
12	38/	39/	23	Vanity IIx	Stefan Strack	137	126
13	39/	42/	19	Fscan	Jay Han	136	19
14	38/	41/	21	Aleph 0	Jay Han	136	34
15	30/	27/	43	Splash 1	Jay Han	133	136
16	29/	26/	45	Blue Funk	Steven Morrell	133	25
17	29/	25/	47	NotSoBigImps	James Layland	132	31
18	40/	49/	11	Squint	Mike Nonemacher	131	109
19	31/	31/	38	Lucky 13	Stefan Strack	131	177
20	29/	27/	44	Der Zweite Blitzkrieg - 9	Mike Nonemacher	131	133

This hill is unchanged since last month.

The ICWS ’88 Standard Hill:

```
        Standard:       ’88
       Core size:       8000 instructions
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current Standard KotH hill on "Pizza":

#	%W/	%L/	%T	Name	Author	Score	Age
1	42/	26/	32	Yop La Boum v2.1	P.E.M & E.C.	158	56
2	44/	44/	13	Dragon Spear	c w blue	144	78
3	32/	21/	47	Sphinx v5.1	W. Mintardjo	143	12
4	32/	21/	47	Der Zweite Blitzkrieg	Mike Nonemacher	142	60
5	43/	45/	12	Iron Gate 1.5	Wayne Sheppard	141	109
6	40/	41/	19	Vanity II	Stefan Strack	140	100
7	29/	19/	51	CAPS KEY IS STUCK AGAIN	Steven Morrell	140	66
8	39/	40/	21	Request v2.0	Brant D. Thomsen	138	79
9	26/	15/	59	ttti	nandor sieben	136	82
10	28/	19/	53	The Plauge	Anonymous	136	15
11	27/	19/	53	NC Killer	Anonymous	135	16
12	37/	39/	23	Christopher	Steven Morrell	135	52
13	34/	33/	32	Annihilator v2.1	Carlos Paredes	135	3
14	27/	18/	55	Blue Funk 88	Steven Morrell	135	25
15	27/	19/	54	Night Crawler	Wayne Sheppard	134	5
16	28/	24/	47	B-Panama V.i	Steven Morrell	132	10
17	37/	41/	22	Titled	Steven Morrell	132	2
18	26/	19/	55	Imprimis 6	Anonymous	132	17
19	24/	16/	61	Peace	Mr. Jones	132	1
20	34/	38/	28	Keystone t21	P.Kline	131	98

The ’88 hill hasn’t aged much (1), but don’t let that fool you. This hill is still very busy. The old standards still appear to be holding their own!

The ICWS ’94 Beginner Hill:

```
        Standard:       ’94 Draft (with extensions)
       Core size:       8000 instructions
   Max processes:       8000 per program
        Duration:       After 80,000 cycles, a tie is declared.
Max entry length:       100 instructions
```

The current Beginner KotH hill on "Pizza":

#	%W/ %L/ %T	Name	Author	Score	Age
1	58/ 31/ 10	Samba2.2	Mr. Jones	185	21
2	57/ 33/ 10	Viewmaster4.0	Ryan Case	181	222
3	56/ 37/ 7	Ecstacy (XTC)	Brant Thomsen	176	89
4	35/ 10/ 56	Peace	Mr. Jones	160	5
5	39/ 27/ 34	Ox v1.0	Brian Zellner	150	18
6	28/ 12/ 60	Rabid Dwarfs v1.3	Brian Zellner	144	64
7	33/ 23/ 44	Slate, v0.3a	Bharat Mediratta	143	143
8	40/ 37/ 23	Bloody Fang v2.1	Bharat Mediratta	142	9
9	37/ 36/ 27	AB Scanner 1.4a	Chris Hodson	138	1
10	35/ 36/ 29	Green Knight v1.7	Brian Zellner	134	65
11	33/ 34/ 33	Count Zero #1	Marc Schefer	131	85
12	39/ 50/ 12	It	Hari	127	206
13	26/ 25/ 49	Rubarbe et mort-vivant	J.P.Laurin	126	92
14	24/ 23/ 53	War Machine v2.65	Matthias Wollnik	125	10
15	21/ 17/ 62	Paperous	Mr. Jones	124	14
16	38/ 52/ 10	Cat v2.0	Tim Scheer	123	163
17	23/ 24/ 53	War Machine v2.6	Matthias Wollnik	123	11
18	25/ 30/ 45	strange carpet	Erik Hughes	121	3
19	20/ 26/ 53	War Machine v2.5	Matthias Wollnik	114	12
20	18/ 28/ 54	Mostly Harmless	Frank J. T. Wojcik	109	53

Congratulations are due to Mr. Jones: his program "Peace" is not only doing well on the beginner’s hill, but has also made a place for itself on the "Standard" hill. (At least, I assume it’s the same program.) Welcome to the Big League!

HINTS and HELPS:

For the hint this month, I will be discussing quick-scans.

Scanning, bombing, and movement are often talked about in terms of "C", where "C" is one instruction per turn. Standard scanning techniques will allow you to scan one location every two turns, or two instructions every three turns, giving you a scanning speed of 50% or 67%. (Granted, the numbers can look even better if you account for mutilation of the core as a side effect as well.) However, with a quick-scan, you can scan at the fastest possible speed -- 2 instructions per turn, or 200%. The disadvantage is that you cannot use this scan in a loop (without reducing the speed), so even a short scan can be several instructions long.

The original quick-scan, as used in Paul Kline’s QuickFreeze, consists of alternating CMP (SEQ) and MOV statements. When each CMP instruction is executed, the following MOV instruction will only be executed if the two instructions compared are not equal. Each MOV instruction, when executed, will store the values for the previous comparison in a standard location. This allows you to scan two locations for every two instructions, and keep track of the last location where you found something of interest.

When the series of CMP instructions is done executing, the process can then check the standard location referenced by the MOV instructions to see if any of the comparisons failed. If any did, then your program can take advantage of this knowledge by attacking whatever was found. Since what was found is very likely to be your opponent’s original code, it is often possible to harm your opponent while they are still getting started. (Imp-Spirals, for example, take a long time to setup.)

Below is a quick-scan that will run under the ’88 standard. You may need to shorten it, as it only leaves eleven lines for the program you will execute

after the quick-scan. My recommendation would be to quick-scan fewer locations if your warrior needs space for a few more instructions.

```
;redcode
;name Quick-Scan '88 Prototype
;author Brant D. Thomsen
;strategy Add this to the front of your warrior,
;strategy  and see if it improves your score.
;assert CORESIZE == 8000

space   equ     (8000/81)       ; Step when scanning for code.
qbomb   equ     6               ; Step when bombing whatever we found.

scan    cmp     space*1+bottom, space*41+bottom
        mov     #space*1+bottom-found, found
        cmp     space*2+bottom, space*42+bottom
        mov     #space*2+bottom-found, found
        cmp     space*3+bottom, space*43+bottom
        mov     #space*3+bottom-found, found
        cmp     space*4+bottom, space*44+bottom
        mov     #space*4+bottom-found, found
        cmp     space*5+bottom, space*45+bottom
        mov     #space*5+bottom-found, found
        cmp     space*6+bottom, space*46+bottom
        mov     #space*6+bottom-found, found
        cmp     space*7+bottom, space*47+bottom
        mov     #space*7+bottom-found, found
        cmp     space*8+bottom, space*48+bottom
        mov     #space*8+bottom-found, found
        cmp     space*9+bottom, space*49+bottom
        mov     #space*9+bottom-found, found
        cmp     space*10+bottom, space*50+bottom
        mov     #space*10+bottom-found, found
        cmp     space*11+bottom, space*51+bottom
        mov     #space*11+bottom-found, found
        cmp     space*12+bottom, space*52+bottom
        mov     #space*12+bottom-found, found
        cmp     space*13+bottom, space*53+bottom
        mov     #space*13+bottom-found, found
        cmp     space*14+bottom, space*54+bottom
        mov     #space*14+bottom-found, found
        cmp     space*15+bottom, space*55+bottom
        mov     #space*15+bottom-found, found
        cmp     space*16+bottom, space*56+bottom
        mov     #space*16+bottom-found, found
        cmp     space*17+bottom, space*57+bottom
        mov     #space*17+bottom-found, found
        cmp     space*18+bottom, space*58+bottom
        mov     #space*18+bottom-found, found
        cmp     space*19+bottom, space*59+bottom
        mov     #space*19+bottom-found, found
        cmp     space*20+bottom, space*60+bottom
        mov     #space*20+bottom-found, found

        jmn     found, found    ; Get out early if found something.

        cmp     space*21+bottom, space*61+bottom
        mov     #space*21+bottom-found, found
        cmp     space*22+bottom, space*62+bottom
        mov     #space*22+bottom-found, found
        cmp     space*23+bottom, space*63+bottom
        mov     #space*23+bottom-found, found
        cmp     space*24+bottom, space*64+bottom
        mov     #space*24+bottom-found, found
        cmp     space*25+bottom, space*65+bottom
        mov     #space*25+bottom-found, found
        cmp     space*26+bottom, space*66+bottom
        mov     #space*26+bottom-found, found
        cmp     space*27+bottom, space*67+bottom
```

```
        mov     #space*27+bottom-found, found
        cmp     space*28+bottom, space*68+bottom
        mov     #space*28+bottom-found, found
        cmp     space*29+bottom, space*69+bottom
        mov     #space*29+bottom-found, found
        cmp     space*30+bottom, space*70+bottom
        mov     #space*30+bottom-found, found
        cmp     space*31+bottom, space*71+bottom
        mov     #space*31+bottom-found, found
        cmp     space*32+bottom, space*72+bottom
        mov     #space*32+bottom-found, found
        cmp     space*33+bottom, space*73+bottom
        mov     #space*33+bottom-found, found
        cmp     space*34+bottom, space*74+bottom
        mov     #space*34+bottom-found, found
        cmp     space*35+bottom, space*75+bottom
        mov     #space*35+bottom-found, found
        cmp     space*36+bottom, space*76+bottom
        mov     #space*36+bottom-found, found
        cmp     space*37+bottom, space*77+bottom
        mov     #space*37+bottom-found, found
        cmp     space*38+bottom, space*78+bottom
        mov     #space*38+bottom-found, found
        cmp     space*39+bottom, space*79+bottom
        mov     #space*39+bottom-found, found
        cmp     space*40+bottom, space*80+bottom
        mov     #space*40+bottom-found, found

        jmz     warrior, found  ; Don't Quick-bomb if nothing found.

found   cmp     bottom, 0       ; Find which instruction is not blank.
        jmp     2               ; CMP / JMP 2 is the '88 equivalent of SNE.

        add     #40*space, found ; Goto second location if first is blank.

forward mov     jump, @found    ; Use a SPL/JMP bomb.
        mov     split, <found
        add     #(qbomb+1), found
        jmn     forward, @found ; Keep bombing while B-Field is not 0.


        ; Regular warrior starts here.
        ; The first instruction should be labeled "warrior".
        ; Must include the code for a two-line bomb.
        ; (Or, of course, you are welcome to use a different bomb,
        ;  such as a single DAT statement.)

warrior jmp     0, <-100        ; Replace this with your own code.

split   spl     0
jump    jmp     -1

bottom  end     scan
```

(One trick QuickFreeze used, that I haven't seen since, was to have two processes execute two separate quick-scans in parallel. That way, the program won't be killed if a bomber drops a DAT statement onto one of the CMP instructions in the quick-scan. If you can handle having more than one process executing the code for your warrior, this could be a good idea.)

A slightly different type of quick-scan -- first suggested by Wayne Sheppard -- is possible if you use the SNE instruction. (F.Y.I.: The SNE instruction is not mentioned in the '94 draft standard as it currently stands.) By using a combination of SNE and SEQ instructions, the quick-scan code size can be reduced by 25%. The problem with using a SNE/SEQ quick-scan, instead of a standard quick-scan, is that there are four possible locations where the instruction you found could be located. A standard quick-scan narrows it down to two.

The following quick-scan is very similar to the one I used with Stimpy 3.0. It uses ranges that are a set distance away from each other, so that I can use a small loop to re-find the instruction located by the scan. (I use an "X" instead of a "F" in the comparison to guarantee that the instruction found will have a non-zero value in one of its fields.) Once this instruction has been found (again), the the block of code surrounding it is systematically bombed.

This code, and the code above, is slightly longer than it needs to be, as I have gone out of my way to make the quick-scan as fast as possible. Still, there are currently up to 25 lines free for your code and bootstrapping procedure. (Bootstrapping is highly recommended, since a scanner will find you really quickly otherwise.)

```
;redcode-94
;name Quick-Scan '94 Prototype
;author Brant D. Thomsen
;strategy Add this to the front of your warrior,
;strategy  and see if it improves your score.

space   equ     (CORESIZE/81)   ; Step when scanning for code.
qbomb   equ     6               ; Step when bombing whatever we found.

scan    sne.X   space*1+bottom, space*3+bottom
        seq.X   space*2+bottom, space*4+bottom
        mov     #space*1+bottom-found, found
        sne.X   space*5+bottom, space*7+bottom
        seq.X   space*6+bottom, space*8+bottom
        mov     #space*5+bottom-found, found
        sne.X   space*9+bottom, space*11+bottom
        seq.X   space*10+bottom, space*12+bottom
        mov     #space*9+bottom-found, found
        sne.X   space*13+bottom, space*15+bottom
        seq.X   space*14+bottom, space*16+bottom
        mov     #space*13+bottom-found, found
        sne.X   space*17+bottom, space*19+bottom
        seq.X   space*18+bottom, space*20+bottom
        mov     #space*17+bottom-found, found
        sne.X   space*21+bottom, space*23+bottom
        seq.X   space*22+bottom, space*24+bottom
        mov     #space*21+bottom-found, found
        sne.X   space*25+bottom, space*27+bottom
        seq.X   space*26+bottom, space*28+bottom
        mov     #space*25+bottom-found, found
        sne.X   space*29+bottom, space*31+bottom
        seq.X   space*30+bottom, space*32+bottom
        mov     #space*29+bottom-found, found
        sne.X   space*33+bottom, space*35+bottom
        seq.X   space*34+bottom, space*36+bottom
        mov     #space*33+bottom-found, found
        sne.X   space*37+bottom, space*39+bottom
        seq.X   space*38+bottom, space*40+bottom
        mov     #space*37+bottom-found, found

        jmn.B   found, found    ; Get out early if found something.

        sne.X   space*41+bottom, space*43+bottom
        seq.X   space*42+bottom, space*44+bottom
        mov     #space*41+bottom-found, found
        sne.X   space*45+bottom, space*47+bottom
        seq.X   space*46+bottom, space*48+bottom
        mov     #space*45+bottom-found, found
        sne.X   space*49+bottom, space*51+bottom
        seq.X   space*50+bottom, space*52+bottom
        mov     #space*49+bottom-found, found
        sne.X   space*53+bottom, space*55+bottom
        seq.X   space*54+bottom, space*56+bottom
        mov     #space*53+bottom-found, found
        sne.X   space*57+bottom, space*59+bottom
```

```
        seq.X   space*58+bottom, space*60+bottom
        mov     #space*57+bottom-found, found
        sne.X   space*61+bottom, space*63+bottom
        seq.X   space*62+bottom, space*64+bottom
        mov     #space*61+bottom-found, found
        sne.X   space*65+bottom, space*67+bottom
        seq.X   space*66+bottom, space*68+bottom
        mov     #space*65+bottom-found, found
        sne.X   space*69+bottom, space*71+bottom
        seq.X   space*70+bottom, space*72+bottom
        mov     #space*69+bottom-found, found
        sne.X   space*73+bottom, space*75+bottom
        seq.X   space*74+bottom, space*76+bottom
        mov     #space*73+bottom-found, found
        sne.X   space*77+bottom, space*79+bottom
        seq.X   space*78+bottom, space*80+bottom
        mov     #space*77+bottom-found, found

        jmn.B   found, found    ; Quick-bomb if found something.
        jmp     warrior         ; Execute regular code, because nothing found.

        add     #space, found
found   jmz.F   -1, 0           ; Goto the location where we found something.

        mov.B   found, backwd   ; Save this value for use in backward bomb.

forward mov.I   split, >found
        mov.I   jump, @found
        add     #(qbomb-1), found
        jmn.F   forward, @found

        sub     #(2*qbomb), backwd      ; Don't re-bomb over forward-scan.

backwd  mov.I   jump, 0
        mov.I   split, <backwd
        sub     #(qbomb-1), backwd
        jmn.F   backwd, @backwd


        ; Regular warrior starts here.
        ; The first instruction should be labeled "warrior".
        ; Must include the code for a two-line bomb.
        ; (Or, of course, you are welcome to use a different bomb,
        ;  such as a single DAT statement.)

warrior jmp     #0, <-100       ; Replace this with your own code.

split   spl     #0
jump    jmp     -1

bottom  end     scan
```

Looking to the Future:

Merry Christmas!

This newsletter is published monthly. As always, your comments, suggestions, criticisms, and submissions are encouraged and appreciated.