

#|

Copyright (C) 1994 by Yuan Yu. All Rights Reserved.

This script is hereby placed in the public domain, and therefore unlimited editing and redistribution is permitted.

NO WARRANTY

Yuan Yu PROVIDES ABSOLUTELY NO WARRANTY. THE EVENT SCRIPT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SCRIPT IS WITH YOU. SHOULD THE SCRIPT PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT WILL Yuan Yu BE LIABLE TO YOU FOR ANY DAMAGES, ANY LOST PROFITS, LOST MONIES, OR OTHER SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THIS SCRIPT (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY THIRD PARTIES), EVEN IF YOU HAVE ADVISED US OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

|#

; Proof of the Correctness of a LOG2 Program
EVENT: Start with the library "mc20-2" using the compiled version.

#|

The following C function computes the integer logarithm (base 2) of a nonnegative integer. We proved the correctness of the binary of this C function. The binary is produced by Gnu C compiler.

The proof described here was worked out with Matt Kaufmann and Bill Pierce.

```
/* computes the integer logarithm of a nonnegative integer. */
log2(int n)
{
    int log = 0;
    while (n > 1) {
        log++;
    }
}
```

```

    n /= 2;}
    return(log);
}

```

Here is the MC68020 assembly code of the above LOG2 program. The code is generated by "gcc -O".

```

0x22ce <log2>:      linkw fp,#0
0x22d2 <log2+4>:    movel d2,sp@-
0x22d4 <log2+6>:    movel fp@(8),d0
0x22d8 <log2+10>:   clrl d1
0x22da <log2+12>:   bra 0x22e6 <log2+24>
0x22dc <log2+14>:   addql #1,d1
0x22de <log2+16>:   tstl d0
0x22e0 <log2+18>:   bge 0x22e4 <log2+22>
0x22e2 <log2+20>:   addql #1,d0
0x22e4 <log2+22>:   asrl #1,d0
0x22e6 <log2+24>:   movel #1,d2
0x22e8 <log2+26>:   cmpl d0,d2
0x22ea <log2+28>:   blt 0x22dc <log2+14>
0x22ec <log2+30>:   movel d1,d0
0x22ee <log2+32>:   movel fp@(-4),d2
0x22f2 <log2+36>:   unlk fp
0x22f4 <log2+38>:   rts

```

```

<log2>:      0x4e56  0x0000  0x2f02  0x202e  0x0008  0x4281  0x600a  0x5281
<log2+16>:   0x4a80  0x6c02  0x5280  0xe280  0x7401  0xb480  0x6df0  0x2001
<log2+32>:   0x242e  0xfffc  0x4e5e  0x4e75

```

```

'(78      86      0      0      47      2      32      46
  0      8      66     129     96     10     82     129
  74     128    108      2     82     128    226     128
 116      1     180    128    109    240     32      1
  36     46    255    252     78     94     78    117)

```

|#

; in Nqthm, log2 is defined as:

DEFINITION:

LOG2-CODE

```

= '(78 86 0 0 47 2 32 46 0 8 66 129 96 10 82 129 74 128
    108 2 82 128 226 128 116 1 180 128 109 240 32 1 36
    46 255 252 78 94 78 117)

```

; we define the Nqthm counterpart of log2.

DEFINITION:

$\log2(n, \log2)$
= **if** $1 < n$ **then** $\log2(n \div 2, 1 + \log2)$
 else $\log2$ **endif**

; the clock.

DEFINITION:

$\log2\text{-t0}(n)$
= **if** $1 < n$ **then** $\text{splus}(7, \log2\text{-t0}(n \div 2))$
 else 7 **endif**

DEFINITION: $\log2\text{-t}(n) = \text{splus}(5, \log2\text{-t0}(n))$

; an induction hint.

DEFINITION:

$\log2\text{-induct}(s, n, \log2)$
= **if** $1 < n$ **then** $\log2\text{-induct}(\text{stepn}(s, 7), n \div 2, 1 + \log2)$
 else t **endif**

; the preconditions on the initail state.

DEFINITION:

$\log2\text{-statep}(s, n)$
= $((\text{mc-status}(s) = \text{'running'})$
 $\wedge \text{evenp}(\text{mc-pc}(s))$
 $\wedge \text{rom-addrp}(\text{mc-pc}(s), \text{mc-mem}(s), 40)$
 $\wedge \text{mcode-addrp}(\text{mc-pc}(s), \text{mc-mem}(s), \text{LOG2-CODE})$
 $\wedge \text{ram-addrp}(\text{sub}(32, 8, \text{read-sp}(s)), \text{mc-mem}(s), 16)$
 $\wedge (n = \text{iread-mem}(\text{add}(32, \text{read-sp}(s), 4), \text{mc-mem}(s), 4))$
 $\wedge (n \in \mathbf{N}))$

; an intermediate state.

DEFINITION:

$\log2\text{-s0p}(s, n, \log2)$
= $((\text{mc-status}(s) = \text{'running'})$
 $\wedge \text{evenp}(\text{mc-pc}(s))$
 $\wedge \text{rom-addrp}(\text{sub}(32, 24, \text{mc-pc}(s)), \text{mc-mem}(s), 40)$
 $\wedge \text{mcode-addrp}(\text{sub}(32, 24, \text{mc-pc}(s)), \text{mc-mem}(s), \text{LOG2-CODE})$
 $\wedge \text{ram-addrp}(\text{sub}(32, 4, \text{read-an}(32, 6, s)), \text{mc-mem}(s), 16)$
 $\wedge (n = \text{iread-dn}(32, 0, s))$
 $\wedge (\log2 = \text{iread-dn}(32, 1, s))$
 $\wedge \text{int-rangep}(\log2 + n, 32)$
 $\wedge (\log2 \in \mathbf{N})$
 $\wedge (n \in \mathbf{N}))$

; from the initial state to s0: s --> s0.

THEOREM: log2-s-s0

$\text{log2-statep}(s, n) \rightarrow \text{log2-s0p}(\text{stepn}(s, 5), n, 0)$

THEOREM: log2-s-s0-else

$\text{log2-statep}(s, n)$

$\rightarrow ((\text{linked-rtS-addr}(\text{stepn}(s, 5)) = \text{rtS-addr}(s))$
 $\wedge (\text{linked-a6}(\text{stepn}(s, 5)) = \text{read-an}(32, 6, s))$
 $\wedge (\text{read-rn}(32, 14, \text{mc-rfile}(\text{stepn}(s, 5)))$
 $= \text{sub}(32, 4, \text{read-sp}(s)))$
 $\wedge (\text{rn-saved}(\text{stepn}(s, 5)) = \text{read-dn}(32, 2, s)))$

THEOREM: log2-s-s0-rfile

$(\text{log2-statep}(s, n) \wedge \text{d3-7a2-5p}(rn))$
 $\rightarrow (\text{read-rn}(oplen, rn, \text{mc-rfile}(\text{stepn}(s, 5)))$
 $= \text{read-rn}(oplen, rn, \text{mc-rfile}(s)))$

THEOREM: log2-s-s0-mem

$(\text{log2-statep}(s, n) \wedge \text{disjoint}(x, k, \text{sub}(32, 8, \text{read-sp}(s)), 16))$
 $\rightarrow (\text{read-mem}(x, \text{mc-mem}(\text{stepn}(s, 5)), k) = \text{read-mem}(x, \text{mc-mem}(s), k))$

; s0 --> exit.

; base case: s0 --> exit.

THEOREM: log2-s0-sn-base

$(\text{log2-s0p}(s, n, \text{log2}) \wedge (1 \not\leq n))$
 $\rightarrow ((\text{mc-status}(\text{stepn}(s, 7)) = \text{'running'})$
 $\wedge (\text{mc-pc}(\text{stepn}(s, 7)) = \text{linked-rtS-addr}(s))$
 $\wedge (\text{iread-dn}(32, 0, \text{stepn}(s, 7)) = \text{log2})$
 $\wedge (\text{read-rn}(32, 14, \text{mc-rfile}(\text{stepn}(s, 7))) = \text{linked-a6}(s))$
 $\wedge (\text{read-rn}(32, 15, \text{mc-rfile}(\text{stepn}(s, 7)))$
 $= \text{add}(32, \text{read-an}(32, 6, s), 8))$

THEOREM: log2-s0-sn-base-rfile

$(\text{log2-s0p}(s, n, \text{log2}) \wedge (1 \not\leq n) \wedge \text{d2-7a2-5p}(rn) \wedge (oplen \leq 32))$
 $\rightarrow (\text{read-rn}(oplen, rn, \text{mc-rfile}(\text{stepn}(s, 7)))$
 $= \text{if } \text{d3-7a2-5p}(rn) \text{ then } \text{read-rn}(oplen, rn, \text{mc-rfile}(s))$
 $\text{else } \text{head}(\text{rn-saved}(s), oplen) \text{ endif})$

THEOREM: log2-s0s-n-base-mem

$(\text{log2-s0p}(s, n, \text{log2}) \wedge (1 \not\leq n))$
 $\rightarrow (\text{read-mem}(x, \text{mc-mem}(\text{stepn}(s, 7)), k) = \text{read-mem}(x, \text{mc-mem}(s), k))$

; induction case: s0 --> s0.

THEOREM: log2-rangep-la
 $(\text{int-rangep}(m + n, 32) \wedge (1 < n))$
 $\rightarrow \text{int-rangep}(1 + (m + (n \div 2)), 32)$

THEOREM: log2-s0-s0
 $(\text{log2-s0p}(s, n, \text{log2}) \wedge (1 < n))$
 $\rightarrow (\text{log2-s0p}(\text{stepn}(s, 7), n \div 2, 1 + \text{log2})$
 $\quad \wedge (\text{read-rn}(\text{oplen}, 14, \text{mc-rfile}(\text{stepn}(s, 7)))$
 $\quad \quad = \text{read-rn}(\text{oplen}, 14, \text{mc-rfile}(s)))$
 $\quad \wedge (\text{linked-a6}(\text{stepn}(s, 7)) = \text{linked-a6}(s))$
 $\quad \wedge (\text{linked-rts-addr}(\text{stepn}(s, 7)) = \text{linked-rts-addr}(s))$
 $\quad \wedge (\text{read-mem}(x, \text{mc-mem}(\text{stepn}(s, 7)), k)$
 $\quad \quad = \text{read-mem}(x, \text{mc-mem}(s), k))$
 $\quad \wedge (\text{rn-saved}(\text{stepn}(s, 7)) = \text{rn-saved}(s)))$

THEOREM: log2-s0-s0-rfile
 $(\text{log2-s0p}(s, n, \text{log2}) \wedge (1 < n) \wedge \text{d3-7a2-5p}(rn))$
 $\rightarrow (\text{read-rn}(\text{oplen}, rn, \text{mc-rfile}(\text{stepn}(s, 7)))$
 $\quad = \text{read-rn}(\text{oplen}, rn, \text{mc-rfile}(s)))$

; put together.

THEOREM: log2-s0-sn
 $\text{log2-s0p}(s, n, \text{log2})$
 $\rightarrow ((\text{mc-status}(\text{stepn}(s, \text{log2-t0}(n))) = \text{'running'})$
 $\quad \wedge (\text{mc-pc}(\text{stepn}(s, \text{log2-t0}(n))) = \text{linked-rts-addr}(s))$
 $\quad \wedge (\text{iread-dn}(32, 0, \text{stepn}(s, \text{log2-t0}(n))) = \text{log2}(n, \text{log2}))$
 $\quad \wedge (\text{read-rn}(32, 14, \text{mc-rfile}(\text{stepn}(s, \text{log2-t0}(n))))$
 $\quad \quad = \text{linked-a6}(s))$
 $\quad \wedge (\text{read-rn}(32, 15, \text{mc-rfile}(\text{stepn}(s, \text{log2-t0}(n))))$
 $\quad \quad = \text{add}(32, \text{read-an}(32, 6, s), 8))$
 $\quad \wedge (\text{read-mem}(x, \text{mc-mem}(\text{stepn}(s, \text{log2-t0}(n))), k)$
 $\quad \quad = \text{read-mem}(x, \text{mc-mem}(s), k)))$

THEOREM: log2-s0-sn-rfile
 $(\text{log2-s0p}(s, n, \text{log2}) \wedge \text{d2-7a2-5p}(rn) \wedge (\text{oplen} \leq 32))$
 $\rightarrow (\text{read-rn}(\text{oplen}, rn, \text{mc-rfile}(\text{stepn}(s, \text{log2-t0}(n))))$
 $\quad = \text{if } \text{d3-7a2-5p}(rn) \text{ then } \text{read-rn}(\text{oplen}, rn, \text{mc-rfile}(s))$
 $\quad \text{else } \text{head}(\text{rn-saved}(s), \text{oplen}) \text{ endif})$

; correctness.

THEOREM: log2-correct
 $\text{log2-statep}(s, n)$
 $\rightarrow ((\text{mc-status}(\text{stepn}(s, \text{log2-t}(n))) = \text{'running'})$

$$\begin{aligned}
& \wedge (\text{mc-pc}(\text{stepn}(s, \log_2\text{-t}(n))) = \text{rts-addr}(s)) \\
& \wedge (\text{iread-dn}(32, 0, \text{stepn}(s, \log_2\text{-t}(n))) = \log_2(n, 0)) \\
& \wedge (\text{read-an}(32, 6, \text{stepn}(s, \log_2\text{-t}(n))) = \text{read-an}(32, 6, s)) \\
& \wedge (\text{read-an}(32, 7, \text{stepn}(s, \log_2\text{-t}(n))) \\
& \quad = \text{add}(32, \text{read-an}(32, 7, s), 4))
\end{aligned}$$

THEOREM: log2-rfile

$$\begin{aligned}
& (\log_2\text{-statep}(s, n) \wedge \text{d2-7a2-5p}(rn) \wedge (\text{oplen} \leq 32)) \\
& \rightarrow (\text{read-rn}(\text{oplen}, rn, \text{mc-rfile}(\text{stepn}(s, \log_2\text{-t}(n)))) \\
& \quad = \text{read-rn}(\text{oplen}, rn, \text{mc-rfile}(s)))
\end{aligned}$$

THEOREM: log2-mem

$$\begin{aligned}
& (\log_2\text{-statep}(s, n) \wedge \text{disjoint}(x, k, \text{sub}(32, 8, \text{read-sp}(s), 16))) \\
& \rightarrow (\text{read-mem}(x, \text{mc-mem}(\text{stepn}(s, \log_2\text{-t}(n))), k) = \text{read-mem}(x, \text{mc-mem}(s), k))
\end{aligned}$$

; the correctness of the Nqthm function log2.

THEOREM: log2-log

$$(i \in \mathbf{N}) \rightarrow (\log_2(n, i) = (i + \log(2, n)))$$

; $2^{\log_2(n)} \leq n$.

THEOREM: log2-thm1

$$(1 < n) \rightarrow (n \not\leq \exp(2, \log_2(n, 0)))$$

; $n < 2^{(\log_2(n)+1)}$.

THEOREM: log2-thm2

$$n < \exp(2, 1 + \log_2(n, 0))$$

Index

add, 3–6

d2-7a2-5p, 4–6
d3-7a2-5p, 4, 5
disjoint, 4, 6

evenp, 3
exp, 6

head, 4, 5

int-rangep, 3, 5
iread-dn, 3–6
iread-mem, 3

linked-a6, 4, 5
linked-rts-addr, 4, 5
log, 6
log2, 3, 5, 6
log2-code, 2, 3
log2-correct, 5
log2-induct, 3
log2-log, 6
log2-mem, 6
log2-rangep-la, 5
log2-rfile, 6
log2-s-s0, 4
log2-s-s0-else, 4
log2-s-s0-mem, 4
log2-s-s0-rfile, 4
log2-s0-s0, 5
log2-s0-s0-rfile, 5
log2-s0-sn, 5
log2-s0-sn-base, 4
log2-s0-sn-base-rfile, 4
log2-s0-sn-rfile, 5
log2-s0p, 3–5
log2-s0s-n-base-mem, 4
log2-statep, 3–6
log2-t, 3, 5, 6
log2-t0, 3, 5
log2-thm1, 6
log2-thm2, 6

mc-mem, 3–6
mc-pc, 3–6
mc-rfile, 4–6
mc-status, 3–5
mcode-addrp, 3

ram-addrp, 3
read-an, 3–6
read-dn, 4
read-mem, 4–6
read-rn, 4–6
read-sp, 3, 4, 6
rn-saved, 4, 5
rom-addrp, 3
rts-addr, 4, 6

plus, 3
stepn, 3–6
sub, 3, 4, 6