```
#|

 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.

|#

;           Function Pointer Constrains with a Witness GT
```

EVENT: Start with the library "mc20-2" using the compiled version.

```
#|
/* greater than relation */
gt(int a, int b)
{
  if (a == b)
    return 0;
  else if (a > b)
    return 1;
  else return -1;
}

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

0x22de <gt>:    linkw fp,#0
```

```
0x22e2 <gt+4>:  movel fp@(8),d1
0x22e6 <gt+8>:  movel fp@(12),d0
0x22ea <gt+12>: cmpl d1,d0
0x22ec <gt+14>: bne 0x22f2 <gt+20>
0x22ee <gt+16>: clrl d0
0x22f0 <gt+18>: bra 0x22fc <gt+30>
0x22f2 <gt+20>: cmpl d1,d0
0x22f4 <gt+22>: bge 0x22fa <gt+28>
0x22f6 <gt+24>: movel #1,d0
0x22f8 <gt+26>: bra 0x22fc <gt+30>
0x22fa <gt+28>: movel #-1,d0
0x22fc <gt+30>: unlk fp
0x22fe <gt+32>: rts

The machine code:

0x22de <gt>:    0x4e56  0x0000  0x222e  0x0008  0x202e  0x000c  0xb081  0x6604
0x22ee <gt+16>: 0x4280  0x600a  0xb081  0x6c04  0x7001  0x6002  0x70ff  0x4e5e
0x22fe <gt+32>: 0x4e75

'(78     86      0       0       34      46      0       8
  32     46      0       12      176     129     102     4
  66     128     96      10      176     129     108     4
  112    1       96      2       112     255     78      94
  78     117)
|#

; In Nqthm logic, gt-code defines the programs.
```

DEFINITION:
GT-CODE
= '(78 86 0 0 34 46 0 8 32 46 0 12 176 129 102 4 66 128
96 10 176 129 108 4 112 1 96 2 112 255 78 94 78 117)

DEFINITION:
gt (a, b)
= **if** $a = b$ **then** 0
elseif ilessp (b, a) **then** 1
else -1 **endif**

DEFINITION:
gt-t (a, b)
= **if** $a = b$ **then** 9
elseif ilessp (b, a) **then** 11
else 10 **endif**

`; preconditions on the initial state.`

DEFINITION:
gt-statep $(s,\ a,\ b)$
$$
\begin{aligned}
= \quad & ((\text{mc-status}\,(s) = \texttt{'running}) \\
& \quad \wedge \quad \text{evenp}\,(\text{mc-pc}\,(s)) \\
& \quad \wedge \quad \text{rom-addrp}\,(\text{mc-pc}\,(s),\ \text{mc-mem}\,(s),\ \texttt{34}) \\
& \quad \wedge \quad \text{mcode-addrp}\,(\text{mc-pc}\,(s),\ \text{mc-mem}\,(s),\ \text{GT-CODE}) \\
& \quad \wedge \quad \text{ram-addrp}\,(\text{sub}\,(\texttt{32},\ \texttt{4},\ \text{read-sp}\,(s)),\ \text{mc-mem}\,(s),\ \texttt{16}) \\
& \quad \wedge \quad (a = \text{iread-mem}\,(\text{add}\,(\texttt{32},\ \text{read-sp}\,(s),\ \texttt{4}),\ \text{mc-mem}\,(s),\ \texttt{4})) \\
& \quad \wedge \quad (b = \text{iread-mem}\,(\text{add}\,(\texttt{32},\ \text{read-sp}\,(s),\ \texttt{8}),\ \text{mc-mem}\,(s),\ \texttt{4})))
\end{aligned}
$$

`; correctness.`

EVENT: Disable ilessp.

THEOREM: gt-s-sn
$$
\begin{aligned}
& \textbf{let } sn \textbf{ be } \text{stepn}\,(s,\ \text{gt-t}\,(a,\ b)) \\
& \textbf{in} \\
& \text{gt-statep}\,(s,\ a,\ b) \\
\rightarrow \quad & ((\text{mc-status}\,(sn) = \texttt{'running}) \\
& \quad \wedge \quad (\text{mc-pc}\,(sn) = \text{rts-addr}\,(s)) \\
& \quad \wedge \quad (\text{read-rn}\,(\texttt{32},\ \texttt{14},\ \text{mc-rfile}\,(sn)) \\
& \qquad\quad = \quad \text{read-rn}\,(\texttt{32},\ \texttt{14},\ \text{mc-rfile}\,(s))) \\
& \quad \wedge \quad (\text{read-rn}\,(\texttt{32},\ \texttt{15},\ \text{mc-rfile}\,(sn)) \\
& \qquad\quad = \quad \text{add}\,(\texttt{32},\ \text{read-sp}\,(s),\ \texttt{4})) \\
& \quad \wedge \quad (\text{d2-7a2-5p}\,(rn) \\
& \qquad\quad \rightarrow \quad (\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(sn)) \\
& \qquad\qquad\quad = \quad \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s)))) \\
& \quad \wedge \quad (\text{disjoint}\,(x,\ k,\ \text{sub}\,(\texttt{32},\ \texttt{4},\ \text{read-sp}\,(s)),\ \texttt{4}) \\
& \qquad\quad \rightarrow \quad (\text{read-mem}\,(x,\ \text{mc-mem}\,(sn),\ k) \\
& \qquad\qquad\quad = \quad \text{read-mem}\,(x,\ \text{mc-mem}\,(s),\ k))) \\
& \quad \wedge \quad (\text{iread-dn}\,(\texttt{32},\ \texttt{0},\ sn) = \text{gt}\,(a,\ b))) \textbf{ endlet}
\end{aligned}
$$

EVENT: Disable gt-t.

EVENT: Disable gt.

`; function constrain with gt as witness.`

CONSERVATIVE AXIOM: p-disjointness
$$(\text{p-disjoint}\,(x,\ n,\ s) \wedge ((j + \text{index-n}\,(y,\ x)) \leq n)) \rightarrow \text{p-disjoint}\,(y,\ j,\ s)$$

Simultaneously, we introduce the new function symbol *p-disjoint*.

Conservative Axiom: fn*-correctness

$$
\begin{array}{l}
\text{fn*-statep}\,(s,\ a,\ b)\\
\rightarrow\quad \textbf{let}\ sn\ \textbf{be}\ \text{stepn}\,(s,\ \text{fn*-t}\,(a,\ b))\\
\qquad \textbf{in}\\
\qquad (\text{mc-status}\,(sn) = \texttt{'running})\\
\qquad \wedge\quad (\text{mc-pc}\,(sn) = \text{rts-addr}\,(s))\\
\qquad \wedge\quad (\text{read-rn}\,(32,\ 14,\ \text{mc-rfile}\,(sn))\\
\qquad\qquad =\quad \text{read-rn}\,(32,\ 14,\ \text{mc-rfile}\,(s)))\\
\qquad \wedge\quad (\text{read-rn}\,(32,\ 15,\ \text{mc-rfile}\,(sn))\\
\qquad\qquad =\quad \text{add}\,(32,\ \text{read-sp}\,(s),\ 4))\\
\qquad \wedge\quad (((\mathit{oplen} \le 32) \wedge \text{d2-7a2-5p}\,(rn))\\
\qquad\qquad \rightarrow\quad (\text{read-rn}\,(\mathit{oplen},\ rn,\ \text{mc-rfile}\,(sn))\\
\qquad\qquad\qquad =\quad \text{read-rn}\,(\mathit{oplen},\ rn,\ \text{mc-rfile}\,(s))))\\
\qquad \wedge\quad (\text{p-disjoint}\,(x,\ k,\ s)\\
\qquad\qquad \rightarrow\quad (\text{read-mem}\,(x,\ \text{mc-mem}\,(sn),\ k)\\
\qquad\qquad\qquad =\quad \text{read-mem}\,(x,\ \text{mc-mem}\,(s),\ k)))\\
\qquad \wedge\quad (\text{iread-dn}\,(32,\ 0,\ sn) = \text{fn*}\,(a,\ b))\ \textbf{endlet}
\end{array}
$$

Simultaneously, we introduce the new function symbols *fn*-statep*, *fn*-t*, and *fn**.

```
;           Proof of the Correctness of a MAX Function
#|
The following C function max computes the maximum of integers a and b,
according to the comparison function comp.  Here, we study the problem
of function pointer.

max(int a, int b, int (*comp)(int, int))
{
  if ((*comp)(a, b) < 0)
    return b;
  else return a;
}

The MC68020 assembly code of the C function max on SUN-3 is given as
follows.  This binary is generated by "gcc -O".

0x2320 <max>:           linkw fp,#0
0x2324 <max+4>:         moveml d2-d3,sp@-
0x2328 <max+8>:         movel fp@(8),d3
0x232c <max+12>:        movel fp@(12),d2
0x2330 <max+16>:        movel d2,sp@-
0x2332 <max+18>:        movel d3,sp@-
0x2334 <max+20>:        moveal fp@(16),a0
```

```
0x2338 <max+24>:        jsr a0@
0x233a <max+26>:        tstl d0
0x233c <max+28>:        bge 0x2342 <max+34>
0x233e <max+30>:        movel d2,d0
0x2340 <max+32>:        bra 0x2344 <max+36>
0x2342 <max+34>:        movel d3,d0
0x2344 <max+36>:        moveml fp@(-8),d2-d3
0x234a <max+42>:        unlk fp
0x234c <max+44>:        rts

The machine code of the above program is:

<max>:      0x4e56  0x0000  0x48e7  0x3000  0x262e  0x0008  0x242e  0x000c
<max+16>:   0x2f02  0x2f03  0x206e  0x0010  0x4e90  0x4a80  0x6c04  0x2002
<max+32>:   0x6002  0x2003  0x4cee  0x000c  0xfff8  0x4e5e  0x4e75

'(78     86      0       0       72      231     48      0
  38     46      0       8       36      46      0       12
  47     2       47      3       32      110     0       16
  78     144     74      128     108     4       32      2
  96     2       32      3       76      238     0       12
  255    248     78      94      78      117)
|#
```

DEFINITION:
MAX-CODE
= '(78 86 0 0 72 231 48 0 38 46 0 8 36 46 0 12 47 2 47 3
 32 110 0 16 78 144 74 128 108 4 32 2 96 2 32 3 76
 238 0 12 255 248 78 94 78 117)

`; imax is the Nqthm counterpart of the program (max-code).`

DEFINITION:
max-fn* $(a,\ b)$
= **if** negativep (fn* $(a,\ b)$) **then** b
else a **endif**

`; the computation time of the program.`

DEFINITION: max-t0 $(a,\ b) = 8$

DEFINITION:
max-t $(a,\ b)$
= splus (max-t0 $(a,\ b)$,

$$
\begin{array}{l}
\quad\quad \text{splus}\,(\text{fn*-t}\,(a,\ b), \\
\quad\quad\quad\quad \textbf{if}\ \text{negativep}\,(\text{fn*}\,(a,\ b))\ \ \textbf{then}\ 7 \\
\quad\quad\quad\quad \textbf{else}\ 6\ \textbf{endif}))
\end{array}
$$

```
; the assumptions of the initial state.
```

DEFINITION:
max-sp $(s,\ a,\ b)$

$$
\begin{array}{ll}
= & ((\text{mc-status}\,(s) = \texttt{'running}) \\
& \land\ \ \text{evenp}\,(\text{mc-pc}\,(s)) \\
& \land\ \ \text{rom-addrp}\,(\text{mc-pc}\,(s),\ \text{mc-mem}\,(s),\ 46) \\
& \land\ \ \text{mcode-addrp}\,(\text{mc-pc}\,(s),\ \text{mc-mem}\,(s),\ \text{MAX-CODE}) \\
& \land\ \ (a = \text{iread-mem}\,(\text{add}\,(32,\ \text{read-sp}\,(s),\ 4),\ \text{mc-mem}\,(s),\ 4)) \\
& \land\ \ (b = \text{iread-mem}\,(\text{add}\,(32,\ \text{read-sp}\,(s),\ 8),\ \text{mc-mem}\,(s),\ 4)) \\
& \land\ \ \text{ram-addrp}\,(\text{sub}\,(32,\ 24,\ \text{read-sp}\,(s)),\ \text{mc-mem}\,(s),\ 40))
\end{array}
$$

CONSERVATIVE AXIOM: max-state

$$
\begin{array}{l}
\ (\text{max-statep}\,(s,\ a,\ b) \to \text{fn*-statep}\,(\text{stepn}\,(s,\ \text{max-t0}\,(a,\ b)),\ a,\ b)) \\
\land\ \ (\text{max-statep}\,(s,\ a,\ b) \\
\quad\quad \to\ \ \text{p-disjoint}\,(\text{add}\,(32,\ \text{read-rn}\,(32,\ 15,\ \text{mc-rfile}\,(s)),\ 4294967272), \\
\quad\quad\quad\quad\quad\quad 40, \\
\quad\quad\quad\quad\quad\quad \text{stepn}\,(s,\ \text{max-t0}\,(a,\ b)))) \\
\land\ \ (\text{max-statep}\,(s,\ a,\ b) = (\text{max-sp}\,(s,\ a,\ b) \land \text{comp-loadp}\,(s,\ a,\ b)))
\end{array}
$$

Simultaneously, we introduce the new function symbols *max-statep* and *comp-loadp*.

```
; an intermediate state.
```

DEFINITION:
max-s0p $(s,\ a,\ b)$

$$
\begin{array}{ll}
= & ((\text{mc-status}\,(s) = \texttt{'running}) \\
& \land\ \ (\text{mc-pc}\,(s) \\
& \quad\quad = \ \text{read-mem}\,(\text{add}\,(32,\ \text{read-an}\,(32,\ 6,\ s),\ 16),\ \text{mc-mem}\,(s),\ 4)) \\
& \land\ \ \text{evenp}\,(\text{rts-addr}\,(s)) \\
& \land\ \ \text{rom-addrp}\,(\text{sub}\,(32,\ 26,\ \text{rts-addr}\,(s)),\ \text{mc-mem}\,(s),\ 46) \\
& \land\ \ \text{mcode-addrp}\,(\text{sub}\,(32,\ 26,\ \text{rts-addr}\,(s)),\ \text{mc-mem}\,(s),\ \text{MAX-CODE}) \\
& \land\ \ \text{ram-addrp}\,(\text{read-sp}\,(s),\ \text{mc-mem}\,(s),\ 40) \\
& \land\ \ \text{equal*}\,(\text{read-sp}\,(s),\ \text{sub}\,(32,\ 20,\ \text{read-an}\,(32,\ 6,\ s))) \\
& \land\ \ (a = \text{iread-dn}\,(32,\ 3,\ s)) \\
& \land\ \ (b = \text{iread-dn}\,(32,\ 2,\ s)) \\
& \land\ \ (\text{iread-mem}\,(\text{add}\,(32,\ \text{read-sp}\,(s),\ 4),\ \text{mc-mem}\,(s),\ 4) = a) \\
& \land\ \ (\text{iread-mem}\,(\text{add}\,(32,\ \text{read-sp}\,(s),\ 8),\ \text{mc-mem}\,(s),\ 4) = b))
\end{array}
$$

```
; an intermediate state.
```

DEFINITION:
max-s1p $(s,\ a,\ b)$
$=$ ((mc-status (s) = `'running`)
  $\wedge$ evenp (mc-pc (s))
  $\wedge$ rom-addrp (sub (32, 26, mc-pc (s)), mc-mem (s), 46)
  $\wedge$ mcode-addrp (sub (32, 26, mc-pc (s)), mc-mem (s), MAX-CODE)
  $\wedge$ ram-addrp (sub (32, 20, read-an (32, 6, s)), mc-mem (s), 40)
  $\wedge$ (a = iread-dn (32, 3, s))
  $\wedge$ (b = iread-dn (32, 2, s))
  $\wedge$ (iread-dn (32, 0, s) = fn* $(a,\ b)$))

CONSERVATIVE AXIOM: max-disjointness
(max-statep $(s,\ a,\ b)$ $\wedge$ max-disjoint $(x,\ k,\ s)$)
$\rightarrow$ p-disjoint (x, k, stepn (s, max-t0 $(a,\ b)$))

Simultaneously, we introduce the new function symbol *max-disjoint*.

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

THEOREM: max-s-s0
max-sp $(s,\ a,\ b)$ $\rightarrow$ max-s0p (stepn (s, max-t0 $(a,\ b)$), a, b)

THEOREM: max-s-s0-else
let *s0* **be** stepn (s, max-t0 $(a,\ b)$)
in
max-sp $(s,\ a,\ b)$
$\rightarrow$ ((linked-rts-addr (*s0*) = rts-addr (s))
  $\wedge$ (linked-a6 (*s0*) = read-an (32, 6, s))
  $\wedge$ (read-rn (32, 14, mc-rfile (*s0*))
    = sub (32, 4, read-sp (s)))
  $\wedge$ (movem-saved (*s0*, 4, 8, 2)
    = readm-rn (32, '(2 3), mc-rfile (s)))) **endlet**

THEOREM: max-s-s0-rfile
(max-sp $(s,\ a,\ b)$ $\wedge$ d4-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, max-t0 $(a,\ b)$)))
  = read-rn (*oplen*, *rn*, mc-rfile (s)))

THEOREM: max-s-s0-mem
(max-sp $(s,\ a,\ b)$ $\wedge$ disjoint (x, k, sub (32, 24, read-sp (s)), 24))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, max-t0 $(a,\ b)$)), k)
  = read-mem (x, mc-mem (s), k))

```
; from s0 to s1.  s0 --> s1.  This is basically a fact about the
; subroutine comp.
```

THEOREM: max-s0-s1
$(\text{max-s0p}(s, a, b) \land \text{fn*-statep}(s, a, b)) \rightarrow \text{max-s1p}(\text{stepn}(s, \text{fn*-t}(a, b)), a, b)$

THEOREM: max-s0-s1-else
$$
\begin{array}{l}
(\text{max-s0p}(s, a, b) \\
\;\land\; \text{fn*-statep}(s, a, b) \\
\;\land\; \text{p-disjoint}(\text{sub}(\texttt{32}, \texttt{20}, \text{read-an}(\texttt{32}, \texttt{6}, s)), \texttt{40}, s)) \\
\rightarrow\; ((\text{linked-rts-addr}(\text{stepn}(s, \text{fn*-t}(a, b))) = \text{linked-rts-addr}(s)) \\
\qquad \land\; (\text{read-rn}(\texttt{32}, \texttt{14}, \text{mc-rfile}(\text{stepn}(s, \text{fn*-t}(a, b)))) \\
\qquad\qquad = \text{read-rn}(\texttt{32}, \texttt{14}, \text{mc-rfile}(s))) \\
\qquad \land\; (\text{linked-a6}(\text{stepn}(s, \text{fn*-t}(a, b))) = \text{linked-a6}(s)) \\
\qquad \land\; (\text{movem-saved}(\text{stepn}(s, \text{fn*-t}(a, b)), \texttt{4}, \texttt{8}, \texttt{2}) \\
\qquad\qquad = \text{movem-saved}(s, \texttt{4}, \texttt{8}, \texttt{2})))
\end{array}
$$

THEOREM: max-s0-s1-rfile
$$
\begin{array}{l}
(\text{max-s0p}(s, a, b) \land \text{fn*-statep}(s, a, b) \land (oplen \leq \texttt{32}) \land \text{d2-7a2-5p}(rn)) \\
\rightarrow\; (\text{read-rn}(oplen, rn, \text{mc-rfile}(\text{stepn}(s, \text{fn*-t}(a, b)))) \\
\qquad = \text{read-rn}(oplen, rn, \text{mc-rfile}(s)))
\end{array}
$$

THEOREM: max-s0-s1-mem
$$
\begin{array}{l}
(\text{max-s0p}(s, a, b) \land \text{fn*-statep}(s, a, b) \land \text{p-disjoint}(x, k, s)) \\
\rightarrow\; (\text{read-mem}(x, \text{mc-mem}(\text{stepn}(s, \text{fn*-t}(a, b))), k) \\
\qquad = \text{read-mem}(x, \text{mc-mem}(s), k))
\end{array}
$$

```
; from s1 to exit.  s1 --> exit.
```

THEOREM: max-s1-sn-1
$$
\begin{array}{l}
(\text{max-s1p}(s, a, b) \land \text{negativep}(\text{fn*}(a, b))) \\
\rightarrow\; ((\text{mc-status}(\text{stepn}(s, \texttt{7})) = \texttt{'running}) \\
\qquad \land\; (\text{mc-pc}(\text{stepn}(s, \texttt{7})) = \text{linked-rts-addr}(s)) \\
\qquad \land\; (\text{iread-dn}(\texttt{32}, \texttt{0}, \text{stepn}(s, \texttt{7})) = b) \\
\qquad \land\; (\text{read-rn}(\texttt{32}, \texttt{14}, \text{mc-rfile}(\text{stepn}(s, \texttt{7}))) = \text{linked-a6}(s)) \\
\qquad \land\; (\text{read-rn}(\texttt{32}, \texttt{15}, \text{mc-rfile}(\text{stepn}(s, \texttt{7}))) \\
\qquad\qquad = \text{add}(\texttt{32}, \text{read-an}(\texttt{32}, \texttt{6}, s), \texttt{8})) \\
\qquad \land\; (\text{read-mem}(x, \text{mc-mem}(\text{stepn}(s, \texttt{7})), k) \\
\qquad\qquad = \text{read-mem}(x, \text{mc-mem}(s), k)))
\end{array}
$$

THEOREM: max-s1-sn-rfile-1
$$
\begin{array}{l}
(\text{max-s1p}(s, a, b) \\
\;\land\; \text{negativep}(\text{fn*}(a, b)) \\
\;\land\; (oplen \leq \texttt{32}) \\
\;\land\; \text{d2-7a2-5p}(rn)) \\
\rightarrow\; (\text{read-rn}(oplen, rn, \text{mc-rfile}(\text{stepn}(s, \texttt{7}))) \\
\qquad = \textbf{if } \text{d4-7a2-5p}(rn) \textbf{ then } \text{read-rn}(oplen, rn, \text{mc-rfile}(s)) \\
\qquad\quad \textbf{else } \text{get-vlst}(oplen, \texttt{0}, rn, \texttt{'(2 3)}, \text{movem-saved}(s, \texttt{4}, \texttt{8}, \texttt{2})) \textbf{ endif})
\end{array}
$$

THEOREM: max-s1-sn-2

(max-s1p $(s,\ a,\ b)$ $\wedge$ ($\neg$ negativep (fn* $(a,\ b)$)))
$\rightarrow$ ((mc-status (stepn $(s,$ `6`)) = `'running`)
$\wedge$ (mc-pc (stepn $(s,$ `6`)) = linked-rts-addr (s))
$\wedge$ (iread-dn (`32`, `0`, stepn $(s,$ `6`)) = a)
$\wedge$ (read-rn (`32`, `14`, mc-rfile (stepn $(s,$ `6`))) = linked-a6 (s))
$\wedge$ (read-rn (`32`, `15`, mc-rfile (stepn $(s,$ `6`)))
= add (`32`, read-an (`32`, `6`, s), `8`))
$\wedge$ (read-mem (x, mc-mem (stepn $(s,$ `6`)), k)
= read-mem (x, mc-mem (s), k)))

THEOREM: max-s1-sn-rfile-2

(max-s1p $(s,\ a,\ b)$
$\wedge$ ($\neg$ negativep (fn* $(a,\ b)$))
$\wedge$ ($oplen \leq$ `32`)
$\wedge$ d2-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn $(s,$ `6`)))
= **if** d4-7a2-5p (rn) **then** read-rn ($oplen$, rn, mc-rfile (s))
else get-vlst ($oplen$, `0`, rn, `'(2 3)`, movem-saved (s, `4`, `8`, `2`)) **endif**)

```
; the correctness of max.
```

EVENT: Disable max-t0.

THEOREM: max-correctness

let sn **be** stepn (s, max-t $(a,\ b)$)
in
max-statep $(s,\ a,\ b)$
$\rightarrow$ ((mc-status (sn) = `'running`)
$\wedge$ (mc-pc (sn) = rts-addr (s))
$\wedge$ (read-rn (`32`, `14`, mc-rfile (sn))
= read-rn (`32`, `14`, mc-rfile (s)))
$\wedge$ (read-rn (`32`, `15`, mc-rfile (sn))
= add (`32`, read-sp (s), `4`))
$\wedge$ ((($oplen \leq$ `32`) $\wedge$ d2-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (sn))
= read-rn ($oplen$, rn, mc-rfile (s))))
$\wedge$ ((disjoint (x, k, sub (`32`, `24`, read-sp (s)), `40`)
$\wedge$ max-disjoint $(x,\ k,\ s)$)
$\rightarrow$ (read-mem (x, mc-mem (sn), k)
= read-mem (x, mc-mem (s), k)))
$\wedge$ (iread-dn (`32`, `0`, sn) = max-fn* $(a,\ b)$)) **endlet**

THEOREM: disjoint-la4

(disjoint (*a*, *m*, *b*, *n*)
∧ ((*j* + index-n (*a1*, *a*)) ≤ *m*)
∧ ((*l* + index-n (*b1*, *b*)) ≤ *n*))
→ disjoint (*a1*, *j*, *b1*, *l*)

`; instantiation.`

DEFINITION:
max-gt-statep (*s*, *a*, *b*)
= **let** *comp* **be** read-mem (add (32, read-sp (*s*), 12), mc-mem (*s*), 4)
in
(mc-status (*s*) = `'running`)
∧ evenp (mc-pc (*s*))
∧ rom-addrp (mc-pc (*s*), mc-mem (*s*), 46)
∧ mcode-addrp (mc-pc (*s*), mc-mem (*s*), MAX-CODE)
∧ (*a* = iread-mem (add (32, read-sp (*s*), 4), mc-mem (*s*), 4))
∧ (*b* = iread-mem (add (32, read-sp (*s*), 8), mc-mem (*s*), 4))
∧ evenp (*comp*)
∧ rom-addrp (*comp*, mc-mem (*s*), len (GT-CODE))
∧ mcode-addrp (*comp*, mc-mem (*s*), GT-CODE)
∧ ram-addrp (sub (32, 28, read-sp (*s*)), mc-mem (*s*), 44) **endlet**

DEFINITION:
max-gt-t (*a*, *b*)
= splus (max-t0 (*a*, *b*),
splus (gt-t (*a*, *b*),
if negativep (gt (*a*, *b*)) **then** 7
else 6 **endif**))

DEFINITION:
max-gt (*a*, *b*)
= **if** negativep (gt (*a*, *b*)) **then** *b*
else *a* **endif**

THEOREM: max-gt-statep-s0
let *s0* **be** stepn (*s*, max-t0 (*a*, *b*))
in
max-gt-statep (*s*, *a*, *b*)
→ ((mc-status (*s0*) = `'running`)
∧ (mc-pc (*s0*)
= read-mem (add (32, read-sp (*s*), 12), mc-mem (*s*), 4))
∧ (read-rn (32, 15, mc-rfile (*s0*))
= sub (32, 24, read-sp (*s*)))
∧ (iread-mem (add (32, read-rn (32, 15, mc-rfile (*s0*)), 4),
mc-mem (*s0*),

```
                    4)
            =   a)
      ∧   (iread-mem (add (32, read-rn (32, 15, mc-rfile (s0)), 8),
                    mc-mem (s0),
                    4)
            =   b)) endlet
```

THEOREM: max-gt-correctness

```
 max-gt-statep (s, a, b)
→   let sn be stepn (s, max-gt-t (a, b))
    in
    (mc-status (sn) = 'running)
    ∧   (mc-pc (sn) = rts-addr (s))
    ∧   (read-rn (32, 14, mc-rfile (sn))
          =   read-rn (32, 14, mc-rfile (s)))
    ∧   (read-rn (32, 15, mc-rfile (sn))
          =   add (32, read-sp (s), 4))
    ∧   (((oplen ≤ 32) ∧ d2-7a2-5p (rn))
          →   (read-rn (oplen, rn, mc-rfile (sn))
                =   read-rn (oplen, rn, mc-rfile (s))))
    ∧   ((disjoint (x, k, sub (32, 24, read-sp (s)), 40)
           ∧   disjoint (x, k, sub (32, 28, read-sp (s)), 4))
          →   (read-mem (x, mc-mem (sn), k)
                =   read-mem (x, mc-mem (s), k)))
    ∧   (iread-dn (32, 0, sn) = max-gt (a, b)) endlet
```

Index