```
#|
 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 the ISQRT Program
;
```

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

```
#|
The following C function ISQRT computes the integer square root of a given
nonnegative integer i.

/* computes the integer square root of a given nonnegative integer */
isqrt(int i)
{
  int j;

  j = (i / 2);
  while ((i / j) < j)
    j = (j + (i / j)) / 2;
  return (j);
}
```

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

The program expects an nonnegative integer input which is on the sp stack at address (sp)+4. The local variable j is assigned to d2. The program exits with the answer in register d0 and the program counter equals the return address saved on the sp stack at address (sp).

```
0x2318 <isqrt>:  linkw a6,#0
0x231c <isqrt+4>: movel d2,sp@-
0x231e <isqrt+6>: movel a6@(8),d1
0x2322 <isqrt+10>: movel d1,d2
0x2324 <isqrt+12>: bra 0x232e <isqrt+22>
0x2326 <isqrt+14>: movel d2,d0
0x2328 <isqrt+16>: divsll d1,d0,d0
0x232c <isqrt+20>: addl d0,d1
0x232e <isqrt+22>: tstl d1
0x2330 <isqrt+24>: bge 0x2334 <isqrt+28>
0x2332 <isqrt+26>: addql #1,d1
0x2334 <isqrt+28>: asrl #1,d1
0x2336 <isqrt+30>: movel d2,d0
0x2338 <isqrt+32>: divsll d1,d0,d0
0x233c <isqrt+36>: cmpl d0,d1
0x233e <isqrt+38>: bgt 0x2326 <isqrt+14>
0x2340 <isqrt+40>: movel d1,d0
0x2342 <isqrt+42>: movel a6@(-4),d2
0x2346 <isqrt+46>: unlk a6
0x2348 <isqrt+48>: rts
```

The machine code of the above program is:

```
<isqrt>: 0x4e56 0x0000 0x2f02 0x222e 0x0008 0x2401 0x6008 0x2002
<isqrt+16>: 0x4c41 0x0800 0xd280 0x4a81 0x6c02 0x5281 0xe281 0x2002
<isqrt+32>: 0x4c41 0x0800 0xb280 0x6ee6 0x2001 0x242e 0xfffc 0x4e5e
<isqrt+48>: 0x4e75
```

In the logic, it is going to be a list like:

```
'(78    86    0     0     47    2     34    46
  0     8     36    1     96    8     32    2
  76    65    8     0     210   128   74    129
  108   2     82    129   226   129   32    2
  76    65    8     0     178   128   110   230
```

```
  32      1       36      46      255     252     78      94
  78      117)
|#

; in the logic, the program is defined by (isqrt-code).
```

DEFINITION:
ISQRT-CODE
= '(78 86 0 0 47 2 34 46 0 8 36 1 96 8 32 2 76 65 8 0
   210 128 74 129 108 2 82 129 226 129 32 2 76 65 8 0
   178 128 110 230 32 1 36 46 255 252 78 94 78 117)

THEOREM: mean-lessp-1
$(((b + a) \div 2) < b) = (a < b)$

```
; isqrt1 is a function in the Logic simulating the loop of the above code.
```

DEFINITION:
isqrt1 (i, j)
= **if** $j \simeq 0$ **then** fix (i)
elseif $(i \div j) < j$ **then** isqrt1 $(i, (j + (i \div j)) \div 2)$
else fix (j) **endif**

```
; isqrt is a function in the Logic simulating the above code.
```

DEFINITION: isqrt (i) = isqrt1 $(i, i \div 2)$

```
; the computation time of the loop.
```

DEFINITION:
isqrt1-t (i, j)
= **if** $j \simeq 0$ **then** 0
elseif $(i \div j) < j$
then splus (10, isqrt1-t $(i, (j + (i \div j)) \div 2)$)
else 8 **endif**

DEFINITION: isqrt-t (i) = splus (8, isqrt1-t $(i, i \div 2)$)

EVENT: Enable iplus.

EVENT: Enable integerp.

EVENT: Enable iquotient.

EVENT: Enable ilessp.

EVENT: Disable remainder.

EVENT: Disable quotient.

THEOREM: isqrt-no-overflow
$(\text{int-rangep}(2 * j,\ n) \land ((i \div j) < j))$
$\rightarrow \quad \text{int-rangep}(j + (i \div j),\ n)$

THEOREM: j-nonzerop
$((1 < i) \land (0 < j)) \rightarrow (((j + (i \div j)) \div 2) \neq 0)$

THEOREM: j-int-rangep
$(\text{int-rangep}(2 * j,\ n) \land ((i \div j) < j))$
$\rightarrow \quad \text{int-rangep}(2 * ((j + (i \div j)) \div 2),\ n)$

`; an induction hint.`

DEFINITION:
isqrt-induct $(s,\ i,\ j)$
$=$ **if** $j \simeq 0$ **then t**
elseif $(i \div j) < j$
then isqrt-induct (stepn $(s,\ 10)$, i, $(j + (i \div j)) \div 2$)
else t endif

DEFINITION:
isqrt-statep $(s,\ i)$
$=$ ((mc-status (s) = 'running)
$\land$ evenp (mc-pc (s)))
$\land$ rom-addrp (mc-pc (s), mc-mem (s), 50)
$\land$ mcode-addrp (mc-pc (s), mc-mem (s), ISQRT-CODE)
$\land$ ram-addrp (sub (32, 8, read-sp (s)), mc-mem (s), 16)
$\land$ (i = iread-mem (add (32, read-sp (s), 4), mc-mem (s), 4))
$\land$ ilessp (1, i))

DEFINITION:
isqrt-s0p $(s,\ i,\ j)$
$=$ ((mc-status (s) = 'running)
$\land$ evenp (mc-pc (s)))
$\land$ rom-addrp (sub (32, 30, mc-pc (s)), mc-mem (s), 50)
$\land$ mcode-addrp (sub (32, 30, mc-pc (s)), mc-mem (s), ISQRT-CODE)
$\land$ ram-addrp (sub (32, 4, read-an (32, 6, s)), mc-mem (s), 16)
$\land$ (i = iread-dn (32, 2, s))
$\land$ (j = iread-dn (32, 1, s))
$\land$ int-rangep ($2 * j$, 32)
$\land$ ilessp (1, i)
$\land$ ilessp (0, j))

```
; from the initial state to s0.
```

THEOREM: initial-j-int-rangep
$\text{int-rangep}(i, n) \rightarrow \text{int-rangep}(2 * (i \div 2), n)$

THEOREM: isqrt-s-s0
$\text{isqrt-statep}(s, i)$
$\rightarrow$ (isqrt-s0p (stepn (s, 8), i, $i \div 2$)
∧ (linked-rts-addr (stepn (s, 8)) = rts-addr (s))
∧ (linked-a6 (stepn (s, 8)) = read-an (32, 6, s))
∧ (read-rn (32, 14, mc-rfile (stepn (s, 8)))
= sub (32, 4, read-sp (s)))
∧ (rn-saved (stepn (s, 8)) = read-dn (32, 2, s)))

THEOREM: isqrt-s-s0-rfile
(isqrt-statep (s, i) ∧ d3-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 8)))
= read-rn ($oplen$, rn, mc-rfile (s)))

THEOREM: isqrt-s-s0-mem
(isqrt-statep (s, i) ∧ disjoint (x, k, sub (32, 8, read-sp (s)), 16))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 8)), k) = read-mem (x, mc-mem (s), k))

```
; from s0 to s0 (induction csae), or from s0 to exit (base case).
; base case:
; the basics we need to prove:
;   0. the machine is still running.
;   1. the pc is at the right position.
;   2. d0 contains the local variable j, the computing square root.
;   3. a6 should have the right value.
;   4. sp(a7) should have the right value.
```

THEOREM: isqrt-s0-sn
(isqrt-s0p (s, i, j) ∧ (($i \div j$) $\not<$ j))
$\rightarrow$ ((mc-status (stepn (s, 8)) = `'running`)
∧ (mc-pc (stepn (s, 8)) = linked-rts-addr (s))
∧ (iread-dn (32, 0, stepn (s, 8)) = j)
∧ (read-rn (32, 14, mc-rfile (stepn (s, 8))) = linked-a6 (s))
∧ (read-rn (32, 15, mc-rfile (stepn (s, 8)))
= add (32, read-an (32, 6, s), 8)))

```
; 5. d2, which is used by this program, should be restored.
```

THEOREM: isqrt-s0-sn-d2
(isqrt-s0p (s, i, j) ∧ (($i \div j$) $\not<$ j) ∧ ($oplen \leq 32$))
$\rightarrow$ (read-rn ($oplen$, 2, mc-rfile (stepn (s, 8))) = head (rn-saved (s), $oplen$))

```
; and 6. those registers untouched by this program should still
; have their original contents.
```

THEOREM: isqrt-s0-sn-rfile

$(\text{isqrt-s0p}(s,\ i,\ j) \land ((i \div j) \not< j) \land \text{d3-7a2-5p}(rn))$
$\rightarrow\ (\text{read-rn}(oplen,\ rn,\ \text{mc-rfile}(\text{stepn}(s,\ 8)))$
$=\ \text{read-rn}(oplen,\ rn,\ \text{mc-rfile}(s)))$

```
; and 7. the memory is correctly changed.
```

THEOREM: isqrt1-s0-sn-mem

$(\text{isqrt-s0p}(s,\ i,\ j) \land ((i \div j) \not< j))$
$\rightarrow\ (\text{read-mem}(x,\ \text{mc-mem}(\text{stepn}(s,\ 8)),\ k) = \text{read-mem}(x,\ \text{mc-mem}(s),\ k))$

THEOREM: isqrt-s0p-j_nonzerop

$\text{isqrt-s0p}(s,\ i,\ j) \rightarrow ((j \neq 0) \land (j \in \mathbf{N}))$

```
; induction case: s0 --> s0.
; we need to prove:
;  0. the state predicate at s0 is still satisfied.
;  1. a6 is unchanged.
;  2. the content of original a6 on the sp stack is not changed.
;  3. the return address on the sp atack is not changed.
;  4. the content of original d2 on the sp stack is not changed.
```

THEOREM: isqrt-s0-s0

$(\text{isqrt-s0p}(s,\ i,\ j) \land ((i \div j) < j))$
$\rightarrow\ (\text{isqrt-s0p}(\text{stepn}(s,\ 10),\ i,\ (j + (i \div j)) \div 2)$
$\land\ (\text{read-rn}(oplen,\ 14,\ \text{mc-rfile}(\text{stepn}(s,\ 10)))$
$=\ \text{read-rn}(oplen,\ 14,\ \text{mc-rfile}(s)))$
$\land\ (\text{linked-a6}(\text{stepn}(s,\ 10)) = \text{linked-a6}(s))$
$\land\ (\text{linked-rts-addr}(\text{stepn}(s,\ 10)) = \text{linked-rts-addr}(s))$
$\land\ (\text{rn-saved}(\text{stepn}(s,\ 10)) = \text{rn-saved}(s)))$

```
; and 5. the registers that are not modified by this program still have
; their previous values.
```

THEOREM: isqrt-s0-s0-rfile

$(\text{isqrt-s0p}(s,\ i,\ j) \land ((i \div j) < j) \land \text{d3-7a2-5p}(rn))$
$\rightarrow\ (\text{read-rn}(oplen,\ rn,\ \text{mc-rfile}(\text{stepn}(s,\ 10)))$
$=\ \text{read-rn}(oplen,\ rn,\ \text{mc-rfile}(s)))$

```
; and 6. the memory is correctly changed.
```

THEOREM: isqrt-s0-s0-mem

$(\text{isqrt-s0p}(s,\ i,\ j) \land ((i \div j) < j))$
$\rightarrow\ (\text{read-mem}(x,\ \text{mc-mem}(\text{stepn}(s,\ 10)),\ k) = \text{read-mem}(x,\ \text{mc-mem}(s),\ k))$

EVENT: Disable isqrt-statep.

EVENT: Disable isqrt-s0p.

THEOREM: isqrt1-correctness

$$
\begin{array}{l}
\text{isqrt-s0p}\,(s,\ i,\ j) \\
\rightarrow \quad ((\text{mc-status}\,(\text{stepn}\,(s,\ \text{isqrt1-t}\,(i,\ j))) = \texttt{'running}) \\
\qquad \land \quad (\text{mc-pc}\,(\text{stepn}\,(s,\ \text{isqrt1-t}\,(i,\ j))) = \text{linked-rts-addr}\,(s)) \\
\qquad \land \quad (\text{iread-dn}\,(\texttt{32},\ \texttt{0},\ \text{stepn}\,(s,\ \text{isqrt1-t}\,(i,\ j))) = \text{isqrt1}\,(i,\ j)) \\
\qquad \land \quad (\text{read-rn}\,(\texttt{32},\ \texttt{14},\ \text{mc-rfile}\,(\text{stepn}\,(s,\ \text{isqrt1-t}\,(i,\ j)))) \\
\qquad\qquad = \quad \text{linked-a6}\,(s)) \\
\qquad \land \quad (\text{read-rn}\,(\texttt{32},\ \texttt{15},\ \text{mc-rfile}\,(\text{stepn}\,(s,\ \text{isqrt1-t}\,(i,\ j)))) \\
\qquad\qquad = \quad \text{add}\,(\texttt{32},\ \text{read-an}\,(\texttt{32},\ \texttt{6},\ s),\ \texttt{8})) \\
\qquad \land \quad (\text{read-mem}\,(x,\ \text{mc-mem}\,(\text{stepn}\,(s,\ \text{isqrt1-t}\,(i,\ j))),\ k) \\
\qquad\qquad = \quad \text{read-mem}\,(x,\ \text{mc-mem}\,(s),\ k)))
\end{array}
$$

THEOREM: isqrt1-d2

$$
\begin{array}{l}
(\text{isqrt-s0p}\,(s,\ i,\ j) \land (oplen \leq \texttt{32})) \\
\rightarrow \quad (\text{read-rn}\,(oplen,\ \texttt{2},\ \text{mc-rfile}\,(\text{stepn}\,(s,\ \text{isqrt1-t}\,(i,\ j)))) \\
\qquad = \quad \text{head}\,(\text{rn-saved}\,(s),\ oplen))
\end{array}
$$

THEOREM: isqrt1-rfile

$$
\begin{array}{l}
(\text{isqrt-s0p}\,(s,\ i,\ j) \land \text{d3-7a2-5p}\,(rn)) \\
\rightarrow \quad (\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(\text{stepn}\,(s,\ \text{isqrt1-t}\,(i,\ j)))) \\
\qquad = \quad \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s)))
\end{array}
$$

```
; after an execution of this program,  the machine state satisfies:
;  0. normal exit.
;  1. the pc is returned to the next instruction of the caller.
;  2. the result -- ISQRT(i), is stored in D0.
;  3. a6, used by LINK, is restored to its original content.
;  4. the stack pointer sp(a7) is updated correctly to pop off one frame.
```

THEOREM: isqrt-correctness

$$
\begin{array}{l}
\text{isqrt-statep}\,(s,\ i) \\
\rightarrow \quad ((\text{mc-status}\,(\text{stepn}\,(s,\ \text{isqrt-t}\,(i))) = \texttt{'running}) \\
\qquad \land \quad (\text{mc-pc}\,(\text{stepn}\,(s,\ \text{isqrt-t}\,(i))) = \text{rts-addr}\,(s)) \\
\qquad \land \quad (\text{iread-dn}\,(\texttt{32},\ \texttt{0},\ \text{stepn}\,(s,\ \text{isqrt-t}\,(i))) = \text{isqrt}\,(i)) \\
\qquad \land \quad (\text{read-an}\,(\texttt{32},\ \texttt{6},\ \text{stepn}\,(s,\ \text{isqrt-t}\,(i))) = \text{read-an}\,(\texttt{32},\ \texttt{6},\ s)) \\
\qquad \land \quad (\text{read-an}\,(\texttt{32},\ \texttt{7},\ \text{stepn}\,(s,\ \text{isqrt-t}\,(i))) \\
\qquad\qquad = \quad \text{add}\,(\texttt{32},\ \text{read-an}\,(\texttt{32},\ \texttt{7},\ s),\ \texttt{4})))
\end{array}
$$

```
;  5. d2, used for local variable j, is restored to its original value.
```

THEOREM: isqrt-d2
(isqrt-statep $(s,\ i) \wedge (oplen \leq 32))$
$\rightarrow$ (read-rn $(oplen,\ 2,$ mc-rfile (stepn $(s,$ isqrt-t $(i))))$
$=$ read-dn $(oplen,\ 2,\ s))$

```
; and 6. the registers that are not modified by this program still have
; their original values.
```

THEOREM: isqrt-rfile
(isqrt-statep $(s,\ i) \wedge$ d3-7a2-5p $(rn))$
$\rightarrow$ (read-rn $(oplen,\ rn,$ mc-rfile (stepn $(s,$ isqrt-t $(i))))$
$=$ read-rn $(oplen,\ rn,$ mc-rfile $(s)))$

```
; and 7. the memory is correctly changed.  i.e. specify clearly the portions
; of the memory were changed and prove that is a truth.
```

THEOREM: isqrt-read-mem
(isqrt-statep $(s,\ i) \wedge$ disjoint $(x,\ k,$ sub $(32,\ 8,$ read-sp $(s)),\ 16))$
$\rightarrow$ (read-mem $(x,$ mc-mem (stepn $(s,$ isqrt-t $(i))),\ k)$
$=$ read-mem $(x,$ mc-mem $(s),\ k))$

EVENT: Disable isqrt-t.

```
#|
; we next need to prove that isqrt does compute the integer square root.
; the following is the same as the proof in the file isqrt-ada.events.
(defn sq (j)
  (times j j))

(prove-lemma isqrt1-lower-bound (rewrite)
     (implies (not (zerop j))
      (not (lessp i (sq (isqrt1 i j))))))

(prove-lemma quotient-by-2 (rewrite)
     (not (lessp (plus (quotient x 2) (quotient x 2))
 (sub1 x))))

(prove-lemma main-trick (rewrite)
     (not (lessp (sq (add1 (quotient (plus j k) 2)))
 (plus (times j k) j)))
     ((induct (difference j k))))

(prove-lemma sq-add1-non-zero (rewrite)
     (not (equal (sq (add1 x)) 0)))
```

```
(prove-lemma main (rewrite)
     (implies (not (zerop j))
      (lessp i
     (sq (add1 (quotient (plus j (quotient i j)) 2)))))
     ((disable sq)))

(prove-lemma isqrt1-upper-bound (rewrite)
     (implies (lessp i (sq (add1 j)))
      (lessp i (sq (add1 (isqrt1 i j)))))
     ((disable sq)))

(prove-lemma isqrt->isqrt1 (rewrite)
   (implies (lessp 1 i)
    (lessp i (sq (add1 (quotient i 2))))))

; (isqrt i) is the square root of i:  (isqrt i)^2 <= i < [(isqrt i)+1]^2.
(prove-lemma isqrt-logic-correctness ()
     (implies (lessp 1 i)
      (and (lessp i (sq (add1 (isqrt i))))
   (not (lessp i (sq (isqrt i))))))
     ((disable sq isqrt1)))

|#
```

Index