```
#|

 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 FOO Function
```

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

```
#|
Here is a trivial example to illustrate our ability to handle embedded
assembler in a high level language.

foo returns either a or b depending on the memory value at location 10000.

int foo (int a, int b)
{
  asm("tstl 10000:w ");
  asm("beq l1 ");
  asm("movl a6@(12), d0 ");
  asm("jmp end ");
  asm("l1: movl a6@(8), d0 ");
  asm("end: nop ");
}
```

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

0x243a <foo>:           linkw fp,#0
0x243e <foo+4>:         tstl @#0x2710
0x2442 <foo+8>:         beq 0x244e <foo+20>
0x2446 <foo+12>:        movel fp@(12),d0
0x244a <foo+16>:        jmp 0x2452 <foo+24>
0x244e <foo+20>:        movel fp@(8),d0
0x2452 <foo+24>:        nop
0x2454 <foo+26>:        unlk fp
0x2456 <foo+28>:        rts

The machine code of the above program is:

<foo>:      0x4e56  0x0000  0x4ab8  0x2710  0x6700  0x000a  0x202e  0x000c
<foo+16>:   0x4efa  0x0006  0x202e  0x0008  0x4e71  0x4e5e  0x4e75

'(78      86      0       0       74      184     39      16
  103     0       0       10      32      46      0       12
  78      250     0       6       32      46      0       8
  78      113     78      94      78      117)
|#

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

DEFINITION:
FOO-CODE
= `'(78 86 0 0 74 184 39 16 103 0 0 10 32 46 0 12 78 250 0 6 32 46 0 8 78 113 78 94 78 117)`

```
; the Nqthm counterpart of foo.
```

DEFINITION:
foo (a, b, x)
= **if** $x = 0$ **then** a **else** b **endif**

```
; the computation time of the program.
```

DEFINITION:
foo-t (x)
= **if** $x = 0$ **then** 7 **else** 8 **endif**

```
; the preconditions of the initial state.
```

DEFINITION:
foo-statep (s, a, b)
= ((mc-status (s) = 'running)
∧ evenp (mc-pc (s))
∧ rom-addrp (mc-pc (s), mc-mem (s), 30)
∧ mcode-addrp (mc-pc (s), mc-mem (s), FOO-CODE)
∧ ram-addrp (sub (32, 4, read-sp (s)), mc-mem (s), 16)
∧ ram-addrp (10000, mc-mem (s), 4)
∧ disjoint (10000, 4, sub (32, 4, read-sp (s)), 16)
∧ (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)))

```
; from the initial state to exit: s --> exit.
```

THEOREM: foo-correctness
let x **be** iread-mem (10000, mc-mem (s), 4)
in
foo-statep (s, a, b)
→ ((mc-status (stepn (s, foo-t (x))) = 'running)
∧ (mc-pc (stepn (s, foo-t (x))) = rts-addr (s))
∧ (read-rn (32, 14, mc-rfile (stepn (s, foo-t (x))))
= read-rn (32, 14, mc-rfile (s)))
∧ (read-rn (32, 15, mc-rfile (stepn (s, foo-t (x))))
= add (32, read-an (32, 7, s), 4))
∧ (d2-7a2-5p (rn)
→ (read-rn ($oplen$, rn, mc-rfile (stepn (s, foo-t (x))))
= read-rn ($oplen$, rn, mc-rfile (s))))
∧ (disjoint (x, k, sub (32, 4, read-sp (s)), 16)
→ (read-mem (x, mc-mem (stepn (s, foo-t (x))), k)
= read-mem (x, mc-mem (s), k)))
∧ (iread-dn (32, 0, stepn (s, foo-t (x))) = foo (a, b, x))) **endlet**

Index