```
#|

 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.

|#
```

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

```
;          Proof of the Correctness of the MEMMOVE Function
#|
This is part of our effort to verify the Berkeley string library.  The
Berkeley string library is widely used as part of the Berkeley Unix OS.

This is the source code of memmove function in the Berkeley string library.

typedef int word;               /* "word" used for optimal copy speed */

#define wsize   sizeof(word)
#define wmask   (wsize - 1)

/*
 * Copy a block of memory, handling overlap.
 * This is the routine that actually implements
 * (the portable versions of) bcopy, memcpy, and memmove.
```

```
 */
void *
memmove(dst0, src0, length)
        void *dst0;
        const void *src0;
        register size_t length;
{
        register char *dst = dst0;
        register const char *src = src0;
        register size_t t;

        if (length == 0 || dst == src)          /* nothing to do */
                goto done;

        /*
         * Macros: loop-t-times; and loop-t-times, t>0
         */
#define TLOOP(s) if (t) TLOOP1(s)
#define TLOOP1(s) do { s; } while (--t)

        if ((unsigned long)dst < (unsigned long)src) {
                /*
                 * Copy forward.
                 */
                t = (int)src;   /* only need low bits */
                if ((t | (int)dst) & wmask) {
                        /*
                         * Try to align operands.  This cannot be done
                         * unless the low bits match.
                         */
                        if ((t ^ (int)dst) & wmask || length < wsize)
                                t = length;
                        else
                                t = wsize - (t & wmask);
                        length -= t;
                        TLOOP1(*dst++ = *src++);
                }
                /*
                 * Copy whole words, then mop up any trailing bytes.
                 */
                t = length / wsize;
                TLOOP(*(word *)dst = *(word *)src; src += wsize; dst += wsize);
                t = length & wmask;
                TLOOP(*dst++ = *src++);
```

```
        } else {
                /*
                 * Copy backwards.  Otherwise essentially the same.
                 * Alignment works as before, except that it takes
                 * (t&wmask) bytes to align, not wsize-(t&wmask).
                 */
                src += length;
                dst += length;
                t = (int)src;
                if ((t | (int)dst) & wmask) {
                        if ((t ^ (int)dst) & wmask || length <= wsize)
                                t = length;
                        else
                                t &= wmask;
                        length -= t;
                        TLOOP1(*--dst = *--src);
                }
                t = length / wsize;
                TLOOP(src -= wsize; dst -= wsize; *(word *)dst = *(word *)src);
                t = length & wmask;
                TLOOP(*--dst = *--src);
        }
done:
        return (dst0);
}
```

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

```
0x2550 <memmove>:       linkw fp,#0
0x2554 <memmove+4>:     moveml d2-d4,sp@-
0x2558 <memmove+8>:     movel fp@(8),d3
0x255c <memmove+12>:    movel fp@(16),d2
0x2560 <memmove+16>:    moveal d3,a1
0x2562 <memmove+18>:    moveal fp@(12),a0
0x2566 <memmove+22>:    beq 0x2604 <memmove+180>
0x256a <memmove+26>:    cmpal d3,a0
0x256c <memmove+28>:    beq 0x2604 <memmove+180>
0x2570 <memmove+32>:    bls 0x25bc <memmove+108>
0x2572 <memmove+34>:    movel a0,d1
0x2574 <memmove+36>:    movel d1,d0
0x2576 <memmove+38>:    orl d3,d0
0x2578 <memmove+40>:    movel #3,d4
0x257a <memmove+42>:    andl d4,d0
```

```
0x257c <memmove+44>:    beq 0x25a2 <memmove+82>
0x257e <memmove+46>:    movel d1,d0
0x2580 <memmove+48>:    eorl d3,d0
0x2582 <memmove+50>:    movel #3,d4
0x2584 <memmove+52>:    andl d4,d0
0x2586 <memmove+54>:    bne 0x258e <memmove+62>
0x2588 <memmove+56>:    movel #3,d4
0x258a <memmove+58>:    cmpl d2,d4
0x258c <memmove+60>:    bcs 0x2592 <memmove+66>
0x258e <memmove+62>:    movel d2,d1
0x2590 <memmove+64>:    bra 0x259a <memmove+74>
0x2592 <memmove+66>:    movel #3,d0
0x2594 <memmove+68>:    andl d1,d0
0x2596 <memmove+70>:    movel #4,d1
0x2598 <memmove+72>:    subl d0,d1
0x259a <memmove+74>:    subl d1,d2
0x259c <memmove+76>:    moveb a0@+,a1@+
0x259e <memmove+78>:    subl #1,d1
0x25a0 <memmove+80>:    bne 0x259c <memmove+76>
0x25a2 <memmove+82>:    movel d2,d1
0x25a4 <memmove+84>:    lsrl #2,d1
0x25a6 <memmove+86>:    beq 0x25ae <memmove+94>
0x25a8 <memmove+88>:    movel a0@+,a1@+
0x25aa <memmove+90>:    subl #1,d1
0x25ac <memmove+92>:    bne 0x25a8 <memmove+88>
0x25ae <memmove+94>:    movel #3,d1
0x25b0 <memmove+96>:    andl d2,d1
0x25b2 <memmove+98>:    beq 0x2604 <memmove+180>
0x25b4 <memmove+100>:   moveb a0@+,a1@+
0x25b6 <memmove+102>:   subl #1,d1
0x25b8 <memmove+104>:   bne 0x25b4 <memmove+100>
0x25ba <memmove+106>:   bra 0x2604 <memmove+180>
0x25bc <memmove+108>:   addal d2,a0
0x25be <memmove+110>:   addal d2,a1
0x25c0 <memmove+112>:   movel a0,d1
0x25c2 <memmove+114>:   movel a1,d0
0x25c4 <memmove+116>:   orl d1,d0
0x25c6 <memmove+118>:   movel #3,d4
0x25c8 <memmove+120>:   andl d4,d0
0x25ca <memmove+122>:   beq 0x25ec <memmove+156>
0x25cc <memmove+124>:   movel a1,d0
0x25ce <memmove+126>:   eorl d1,d0
0x25d0 <memmove+128>:   movel #3,d4
0x25d2 <memmove+130>:   andl d4,d0
```

```
0x25d4 <memmove+132>:   bne 0x25dc <memmove+140>
0x25d6 <memmove+134>:   movel #4,d4
0x25d8 <memmove+136>:   cmpl d2,d4
0x25da <memmove+138>:   bcs 0x25e0 <memmove+144>
0x25dc <memmove+140>:   movel d2,d1
0x25de <memmove+142>:   bra 0x25e4 <memmove+148>
0x25e0 <memmove+144>:   movel #3,d4
0x25e2 <memmove+146>:   andl d4,d1
0x25e4 <memmove+148>:   subl d1,d2
0x25e6 <memmove+150>:   moveb a0@-,a1@-
0x25e8 <memmove+152>:   subl #1,d1
0x25ea <memmove+154>:   bne 0x25e6 <memmove+150>
0x25ec <memmove+156>:   movel d2,d1
0x25ee <memmove+158>:   lsrl #2,d1
0x25f0 <memmove+160>:   beq 0x25f8 <memmove+168>
0x25f2 <memmove+162>:   movel a0@-,a1@-
0x25f4 <memmove+164>:   subl #1,d1
0x25f6 <memmove+166>:   bne 0x25f2 <memmove+162>
0x25f8 <memmove+168>:   movel #3,d1
0x25fa <memmove+170>:   andl d2,d1
0x25fc <memmove+172>:   beq 0x2604 <memmove+180>
0x25fe <memmove+174>:   moveb a0@-,a1@-
0x2600 <memmove+176>:   subl #1,d1
0x2602 <memmove+178>:   bne 0x25fe <memmove+174>
0x2604 <memmove+180>:   movel d3,d0
0x2606 <memmove+182>:   moveml fp@(-12),d2-d4
0x260c <memmove+188>:   unlk fp
0x260e <memmove+190>:   rts
```

The machine code of the above program is:

```
<memmove>:       0x4e56  0x0000  0x48e7  0x3800  0x262e  0x0008  0x242e  0x0010
<memmove+16>:    0x2243  0x206e  0x000c  0x6700  0x009c  0xb1c3  0x6700  0x0096
<memmove+32>:    0x634a  0x2208  0x2001  0x8083  0x7803  0xc084  0x6724  0x2001
<memmove+48>:    0xb780  0x7803  0xc084  0x6606  0x7803  0xb882  0x6504  0x2202
<memmove+64>:    0x6008  0x7003  0xc081  0x7204  0x9280  0x9481  0x12d8  0x5381
<memmove+80>:    0x66fa  0x2202  0xe489  0x6706  0x22d8  0x5381  0x66fa  0x7203
<memmove+96>:    0xc282  0x6750  0x12d8  0x5381  0x66fa  0x6048  0xd1c2  0xd3c2
<memmove+112>:   0x2208  0x2009  0x8081  0x7803  0xc084  0x6720  0x2009  0xb380
<memmove+128>:   0x7803  0xc084  0x6606  0x7804  0xb882  0x6504  0x2202  0x6004
<memmove+144>:   0x7803  0xc284  0x9481  0x1320  0x5381  0x66fa  0x2202  0xe489
<memmove+160>:   0x6706  0x2320  0x5381  0x66fa  0x7203  0xc282  0x6706  0x1320
<memmove+176>:   0x5381  0x66fa  0x2003  0x4cee  0x001c  0xfff4  0x4e5e  0x4e75
```

```
'(78      86      0       0       72      231     56      0
  38      46      0       8       36      46      0       16
  34      67      32      110     0       12      103     0
  0       156     177     195     103     0       0       150
  99      74      34      8       32      1       128     131
  120     3       192     132     103     36      32      1
  183     128     120     3       192     132     102     6
  120     3       184     130     101     4       34      2
  96      8       112     3       192     129     114     4
  146     128     148     129     18      216     83      129
  102     250     34      2       228     137     103     6
  34      216     83      129     102     250     114     3
  194     130     103     80      18      216     83      129
  102     250     96      72      209     194     211     194
  34      8       32      9       128     129     120     3
  192     132     103     32      32      9       179     128
  120     3       192     132     102     6       120     4
  184     130     101     4       34      2       96      4
  120     3       194     132     148     129     19      32
  83      129     102     250     34      2       228     137
  103     6       35      32      83      129     102     250
  114     3       194     130     103     6       19      32
  83      129     102     250     32      3       76      238
  0       28      255     244     78      94      78      117)

|#

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

DEFINITION:
MEMMOVE-CODE

```
=   '(78 86 0 0 72 231 56 0 38 46 0 8 36 46 0 16 34 67 32
      110 0 12 103 0 0 156 177 195 103 0 0 150 99 74 34 8
      32 1 128 131 120 3 192 132 103 36 32 1 183 128 120 3
      192 132 102 6 120 3 184 130 101 4 34 2 96 8 112 3
      192 129 114 4 146 128 148 129 18 216 83 129 102 250
      34 2 228 137 103 6 34 216 83 129 102 250 114 3 194
      130 103 80 18 216 83 129 102 250 96 72 209 194 211
      194 34 8 32 9 128 129 120 3 192 132 103 32 32 9 179
      128 120 3 192 132 102 6 120 4 184 130 101 4 34 2 96
      4 120 3 194 132 148 129 19 32 83 129 102 250 34 2
      228 137 103 6 35 32 83 129 102 250 114 3 194 130 103
      6 19 32 83 129 102 250 32 3 76 238 0 28 255 244 78
      94 78 117)
```

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

DEFINITION:
memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
= ((mc-status (s) = 'running)
  ∧ evenp (mc-pc (s))
  ∧ rom-addrp (mc-pc (s), mc-mem (s), 192)
  ∧ mcode-addrp (mc-pc (s), mc-mem (s), MEMMOVE-CODE)
  ∧ ram-addrp (sub (32, 16, read-sp (s)), mc-mem (s), 32)
  ∧ ram-addrp (*str1*, mc-mem (s), n)
  ∧ mem-lst (1, *str1*, mc-mem (s), n, *lst1*)
  ∧ ram-addrp (*str2*, mc-mem (s), n)
  ∧ mem-lst (1, *str2*, mc-mem (s), n, *lst2*)
  ∧ disjoint (sub (32, 16, read-sp (s)), 32, *str1*, n)
  ∧ disjoint (sub (32, 16, read-sp (s)), 32, *str2*, n)
  ∧ (*str1* = read-mem (add (32, read-sp (s), 4), mc-mem (s), 4))
  ∧ (*str2* = read-mem (add (32, read-sp (s), 8), mc-mem (s), 4))
  ∧ (n = uread-mem (add (32, read-sp (s), 12), mc-mem (s), 4))
  ∧ uint-rangep (nat-to-uint (*str1*) + n, 32)
  ∧ uint-rangep (nat-to-uint (*str2*) + n, 32))

```
; intermediate states.
```

DEFINITION:
memmove-s0p (s, i^*, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
= ((mc-status (s) = 'running)
  ∧ evenp (mc-pc (s))
  ∧ rom-addrp (sub (32, 76, mc-pc (s)), mc-mem (s), 192)
  ∧ mcode-addrp (sub (32, 76, mc-pc (s)), mc-mem (s), MEMMOVE-CODE)
  ∧ ram-addrp (sub (32, 12, read-an (32, 6, s)), mc-mem (s), 32)
  ∧ ram-addrp (*str1*, mc-mem (s), $n_$)
  ∧ mem-lst (1, *str1*, mc-mem (s), $n_$, *lst1*)
  ∧ ram-addrp (add (32, *str2*, i^*), mc-mem (s), $n_ - i$)
  ∧ mem-lst (1, add (32, *str2*, i^*), mc-mem (s), $n_ - i$, mcdr (i, *lst2*))
  ∧ disjoint (sub (32, 12, read-an (32, 6, s)), 32, *str1*, $n_$)
  ∧ disjoint (sub (32, 12, read-an (32, 6, s)), 32, *str2*, $n_$)
  ∧ equal* (read-an (32, 1, s), add (32, *str1*, i^*))
  ∧ equal* (read-an (32, 0, s), add (32, *str2*, i^*))
  ∧ (*str1* = read-dn (32, 3, s))
  ∧ (n = uread-dn (32, 2, s))
  ∧ (nt = uread-dn (32, 1, s))
  ∧ ((nat-to-uint (*str1*) + $n_$) < 4294967296)
  ∧ ((nat-to-uint (*str2*) + $n_$) < 4294967296)
  ∧ (nat-to-uint (*str1*) < nat-to-uint (*str2*))
  ∧ nat-rangep (*str2*, 32)

$\wedge$ $((i + nt + n) \leq n_)$
$\wedge$ $(nt \neq 0)$
$\wedge$ $(i^* \in \mathbf{N})$
$\wedge$ nat-rangep $(i^*, 32)$
$\wedge$ $(i =$ nat-to-uint $(i^*))$
$\wedge$ $(n_ \not\simeq 0))$

DEFINITION:
memmove-s1p $(s, i^*, i, str1, n, lst1, str2, lst2, nt, n_)$
$=$ ((mc-status (s) = 'running)
$\wedge$ evenp (mc-pc (s))
$\wedge$ rom-addrp (sub (32, 88, mc-pc (s)), mc-mem (s), 192)
$\wedge$ mcode-addrp (sub (32, 88, mc-pc (s)), mc-mem (s), MEMMOVE-CODE)
$\wedge$ ram-addrp (sub (32, 12, read-an (32, 6, s)), mc-mem (s), 32)
$\wedge$ ram-addrp $(str1$, mc-mem (s), $n_)$
$\wedge$ mem-lst (1, $str1$, mc-mem (s), $n_$, $lst1)$
$\wedge$ ram-addrp (add (32, $str2$, i^*), mc-mem (s), $n_ - i)$
$\wedge$ mem-lst (1, add (32, $str2$, i^*), mc-mem (s), $n_ - i$, mcdr $(i, lst2))$
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, s)), 32, $str1$, $n_)$
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, s)), 32, $str2$, $n_)$
$\wedge$ equal* (read-an (32, 1, s), add (32, $str1$, i^*))
$\wedge$ equal* (read-an (32, 0, s), add (32, $str2$, i^*))
$\wedge$ $(str1 =$ read-dn (32, 3, s))
$\wedge$ $(n =$ uread-dn (32, 2, s))
$\wedge$ $(nt =$ uread-dn (32, 1, s))
$\wedge$ ((nat-to-uint $(str1) + n_) < 4294967296$)
$\wedge$ ((nat-to-uint $(str2) + n_) < 4294967296$)
$\wedge$ (nat-to-uint $(str1) <$ nat-to-uint $(str2))$
$\wedge$ nat-rangep $(str2, 32)$
$\wedge$ $((i + (4 * nt) + (n \textbf{ mod } 4)) \leq n_)$
$\wedge$ $(nt \neq 0)$
$\wedge$ $(i^* \in \mathbf{N})$
$\wedge$ nat-rangep $(i^*, 32)$
$\wedge$ $(i =$ nat-to-uint $(i^*))$
$\wedge$ $(n_ \not\simeq 0))$

DEFINITION:
memmove-s2p $(s, i^*, i, str1, n, lst1, str2, lst2, nt, n_)$
$=$ ((mc-status (s) = 'running)
$\wedge$ evenp (mc-pc (s))
$\wedge$ rom-addrp (sub (32, 100, mc-pc (s)), mc-mem (s), 192)
$\wedge$ mcode-addrp (sub (32, 100, mc-pc (s)), mc-mem (s), MEMMOVE-CODE)
$\wedge$ ram-addrp (sub (32, 12, read-an (32, 6, s)), mc-mem (s), 32)
$\wedge$ ram-addrp $(str1$, mc-mem (s), $n_)$

$\wedge$ mem-lst (1, *str1*, mc-mem (*s*), $n_$, *lst1*)
$\wedge$ ram-addrp (add (32, *str2*, *i**), mc-mem (*s*), $n_ - i$)
$\wedge$ mem-lst (1, add (32, *str2*, *i**), mc-mem (*s*), $n_ - i$, mcdr (*i*, *lst2*))
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, *s*)), 32, *str1*, $n_$)
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, *s*)), 32, *str2*, $n_$)
$\wedge$ equal* (read-an (32, 1, *s*), add (32, *str1*, *i**))
$\wedge$ equal* (read-an (32, 0, *s*), add (32, *str2*, *i**))
$\wedge$ (*str1* = read-dn (32, 3, *s*))
$\wedge$ (*nt* = uread-dn (32, 1, *s*))
$\wedge$ ((nat-to-uint (*str1*) + $n_$) < 4294967296)
$\wedge$ ((nat-to-uint (*str2*) + $n_$) < 4294967296)
$\wedge$ (nat-to-uint (*str1*) < nat-to-uint (*str2*))
$\wedge$ nat-rangep (*str2*, 32)
$\wedge$ ((*i* + *nt*) $\leq$ $n_$)
$\wedge$ (*nt* $\neq$ 0)
$\wedge$ (*i** $\in$ **N**)
$\wedge$ nat-rangep (*i**, 32)
$\wedge$ (*i* = nat-to-uint (*i**))
$\wedge$ ($n_ \not\simeq$ 0))

DEFINITION:
memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, $n_$)
= ((mc-status (*s*) = 'running)
$\wedge$ evenp (mc-pc (*s*))
$\wedge$ rom-addrp (sub (32, 150, mc-pc (*s*)), mc-mem (*s*), 192)
$\wedge$ mcode-addrp (sub (32, 150, mc-pc (*s*)), mc-mem (*s*), MEMMOVE-CODE)
$\wedge$ ram-addrp (sub (32, 12, read-an (32, 6, *s*)), mc-mem (*s*), 32)
$\wedge$ ram-addrp (*str1*, mc-mem (*s*), $n_$)
$\wedge$ mem-lst (1, *str1*, mc-mem (*s*), $n_$, *lst1*)
$\wedge$ ram-addrp (*str2*, mc-mem (*s*), *i*)
$\wedge$ mem-lst (1, *str2*, mc-mem (*s*), *i*, mcar (*i*, *lst2*))
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, *s*)), 32, *str1*, $n_$)
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, *s*)), 32, *str2*, $n_$)
$\wedge$ equal* (read-an (32, 1, *s*), add (32, *str1*, *i**))
$\wedge$ equal* (read-an (32, 0, *s*), add (32, *str2*, *i**))
$\wedge$ (*str1* = read-dn (32, 3, *s*))
$\wedge$ (*n* = uread-dn (32, 2, *s*))
$\wedge$ (*nt* = uread-dn (32, 1, *s*))
$\wedge$ ((nat-to-uint (*str1*) + $n_$) < 4294967296)
$\wedge$ ((nat-to-uint (*str2*) + $n_$) < 4294967296)
$\wedge$ (nat-to-uint (*str2*) < nat-to-uint (*str1*))
$\wedge$ nat-rangep (*str2*, 32)
$\wedge$ (*i** $\in$ **N**)
$\wedge$ nat-rangep (*i**, 32)

$\wedge$ $(i = \text{nat-to-uint}\,(i^*))$
$\wedge$ $(i \leq n_)$
$\wedge$ $((nt + n) \leq i)$
$\wedge$ $(nt \not\simeq 0)$
$\wedge$ $(i \not\simeq 0)$
$\wedge$ $(n_ \not\simeq 0))$

DEFINITION:
memmove-s4p $(s,\ i^*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$
$=$ ((mc-status (s) = `'running`)
$\wedge$ evenp (mc-pc (s))
$\wedge$ rom-addrp (sub (32, 162, mc-pc (s)), mc-mem (s), 192)
$\wedge$ mcode-addrp (sub (32, 162, mc-pc (s)), mc-mem (s), MEMMOVE-CODE)
$\wedge$ ram-addrp (sub (32, 12, read-an (32, 6, s)), mc-mem (s), 32)
$\wedge$ ram-addrp $(str1$, mc-mem (s), $n_)$
$\wedge$ mem-lst (1, $str1$, mc-mem (s), $n_$, $lst1)$
$\wedge$ ram-addrp $(str2$, mc-mem (s), $i)$
$\wedge$ mem-lst (1, $str2$, mc-mem (s), i, mcar $(i,\ lst2))$
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, s)), 32, $str1$, $n_)$
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, s)), 32, $str2$, $n_)$
$\wedge$ equal* (read-an (32, 1, s), add (32, $str1$, i^*))
$\wedge$ equal* (read-an (32, 0, s), add (32, $str2$, i^*))
$\wedge$ $(str1$ = read-dn (32, 3, $s))$
$\wedge$ $(n$ = uread-dn (32, 2, $s))$
$\wedge$ $(nt$ = uread-dn (32, 1, $s))$
$\wedge$ ((nat-to-uint $(str1) + n_) < 4294967296$)
$\wedge$ ((nat-to-uint $(str2) + n_) < 4294967296$)
$\wedge$ (nat-to-uint $(str2)$ < nat-to-uint $(str1))$
$\wedge$ nat-rangep $(str2$, 32)
$\wedge$ $(i^* \in \mathbf{N})$
$\wedge$ nat-rangep $(i^*$, 32)
$\wedge$ $(i = \text{nat-to-uint}\,(i^*))$
$\wedge$ $(i \leq n_)$
$\wedge$ $(((4 * nt) + (n\ \mathbf{mod}\ 4)) \leq i)$
$\wedge$ $(i \not< 4)$
$\wedge$ $(nt \not\simeq 0)$
$\wedge$ $(n_ \not\simeq 0))$

DEFINITION:
memmove-s5p $(s,\ i^*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$
$=$ ((mc-status (s) = `'running`)
$\wedge$ evenp (mc-pc (s))
$\wedge$ rom-addrp (sub (32, 174, mc-pc (s)), mc-mem (s), 192)
$\wedge$ mcode-addrp (sub (32, 174, mc-pc (s)), mc-mem (s), MEMMOVE-CODE)

$\wedge$ ram-addrp (sub (32, 12, read-an (32, 6, s)), mc-mem (s), 32)
$\wedge$ ram-addrp ($str1$, mc-mem (s), $n_$)
$\wedge$ mem-lst (1, $str1$, mc-mem (s), $n_$, $lst1$)
$\wedge$ ram-addrp ($str2$, mc-mem (s), i)
$\wedge$ mem-lst (1, $str2$, mc-mem (s), i, mcar (i, $lst2$))
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, s)), 32, $str1$, $n_$)
$\wedge$ disjoint (sub (32, 12, read-an (32, 6, s)), 32, $str2$, $n_$)
$\wedge$ equal* (read-an (32, 1, s), add (32, $str1$, $i*$))
$\wedge$ equal* (read-an (32, 0, s), add (32, $str2$, $i*$))
$\wedge$ ($str1$ = read-dn (32, 3, s))
$\wedge$ (nt = uread-dn (32, 1, s))
$\wedge$ ((nat-to-uint ($str1$) + $n_$) < 4294967296)
$\wedge$ ((nat-to-uint ($str2$) + $n_$) < 4294967296)
$\wedge$ (nat-to-uint ($str2$) < nat-to-uint ($str1$))
$\wedge$ nat-rangep ($str2$, 32)
$\wedge$ ($i* \in \mathbf{N}$)
$\wedge$ nat-rangep ($i*$, 32)
$\wedge$ (i = nat-to-uint ($i*$))
$\wedge$ ($i \leq n_$)
$\wedge$ ($nt \leq i$)
$\wedge$ ($nt \not\simeq 0$)
$\wedge$ ($i \not\simeq 0$)
$\wedge$ ($n_ \not\simeq 0$))

```
; enable a few events for this proof.
```

EVENT: Enable disjoint-leq-uint.

EVENT: Enable disjoint-leq1-uint.

EVENT: Enable add-uintxxx.

EVENT: Enable plus-times-sub1.

```
; from the initial state to exit:  s --> sn, when n == 0.
```

THEOREM: memmove-s-sn-1
let sn **be** stepn (s, 11)
in
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$) $\wedge$ ($n \simeq 0$))
$\rightarrow$ ((mc-status (sn) = 'running)
$\wedge$ (mc-pc (sn) = rts-addr (s))

$\wedge$ mem-lst (1, *str1*, mc-mem (*sn*), *n*, *lst1*)
$\wedge$ (read-dn (32, 0, *sn*) = *str1*)
$\wedge$ (read-rn (32, 14, mc-rfile (*sn*)) = read-an (32, 6, *s*))
$\wedge$ (read-rn (32, 15, mc-rfile (*sn*))
= add (32, read-an (32, 7, *s*), 4))) **endlet**

THEOREM: memmove-s-sn-rfile-1
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \simeq 0$)
$\wedge$ ($oplen \leq 32$)
$\wedge$ d2-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 11)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s-sn-mem-1
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \simeq 0$)
$\wedge$ disjoint (*x*, *k*, sub (32, 16, read-sp (*s*)), 32))
$\rightarrow$ (read-mem (*x*, mc-mem (stepn (*s*, 11)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; from the initial state to exit:  s --> sn, when n =\= 0 and str1 == str2.
```

THEOREM: memmove-s-sn-2
let *sn* **be** stepn (*s*, 13)
in
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint (*str1*) = nat-to-uint (*str2*)))
$\rightarrow$ ((mc-status (*sn*) = `'running`)
$\wedge$ (mc-pc (*sn*) = rts-addr (*s*))
$\wedge$ mem-lst (1, *str1*, mc-mem (*sn*), *n*, *lst2*)
$\wedge$ (read-dn (32, 0, *sn*) = *str1*)
$\wedge$ (read-rn (32, 14, mc-rfile (*sn*)) = read-an (32, 6, *s*))
$\wedge$ (read-rn (32, 15, mc-rfile (*sn*))
= add (32, read-an (32, 7, *s*), 4))) **endlet**

THEOREM: memmove-s-sn-rfile-2
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint (*str1*) = nat-to-uint (*str2*))
$\wedge$ ($oplen \leq 32$)
$\wedge$ d2-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 13)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s-sn-mem-2
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
 ∧ ($n \not\simeq$ 0)
 ∧ (nat-to-uint (*str1*) = nat-to-uint (*str2*))
 ∧ disjoint (x, k, sub (32, 16, read-sp (s)), 32))
 → (read-mem (x, mc-mem (stepn (s, 13)), k) = read-mem (x, mc-mem (s), k))

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

THEOREM: memmove-s-s0-1
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
 ∧ ($n \not\simeq$ 0)
 ∧ (nat-to-uint (*str1*) ≠ nat-to-uint (*str2*))
 ∧ (nat-to-uint (*str1*) < nat-to-uint (*str2*))
 ∧ ((nat-to-uint (*str1*) **mod** 4) ≠ (nat-to-uint (*str2*) **mod** 4))
 ∧ (((nat-to-uint (*str1*) **mod** 4) ≠ 0)
  ∨ ((nat-to-uint (*str2*) **mod** 4) ≠ 0)))
 → memmove-s0p (stepn (s, 24), 0, 0, *str1*, 0, *lst1*, *str2*, *lst2*, n, n)

THEOREM: memmove-s-s0-else-1
let *s0* **be** stepn (s, 24)
in
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
 ∧ ($n \not\simeq$ 0)
 ∧ (nat-to-uint (*str1*) ≠ nat-to-uint (*str2*))
 ∧ (nat-to-uint (*str1*) < nat-to-uint (*str2*))
 ∧ ((nat-to-uint (*str1*) **mod** 4) ≠ (nat-to-uint (*str2*) **mod** 4))
 ∧ (((nat-to-uint (*str1*) **mod** 4) ≠ 0)
  ∨ ((nat-to-uint (*str2*) **mod** 4) ≠ 0)))
 → ((linked-rts-addr (*s0*) = rts-addr (s))
  ∧ (linked-a6 (*s0*) = read-an (32, 6, s))
  ∧ (read-rn (32, 14, mc-rfile (*s0*))
   = sub (32, 4, read-sp (s)))
  ∧ (movem-saved (*s0*, 4, 12, 3)
   = readm-rn (32, '(2 3 4), mc-rfile (s)))) **endlet**

THEOREM: memmove-s-s0-rfile-1
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
 ∧ ($n \not\simeq$ 0)
 ∧ (nat-to-uint (*str1*) ≠ nat-to-uint (*str2*))
 ∧ (nat-to-uint (*str1*) < nat-to-uint (*str2*))
 ∧ ((nat-to-uint (*str1*) **mod** 4) ≠ (nat-to-uint (*str2*) **mod** 4))
 ∧ (((nat-to-uint (*str1*) **mod** 4) ≠ 0)
  ∨ ((nat-to-uint (*str2*) **mod** 4) ≠ 0))
 ∧ d5-7a2-5p (rn))

$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 24)))
  = read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s-s0-mem-1
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
$\wedge$ ((nat-to-uint (*str1*) **mod** 4) $\neq$ (nat-to-uint (*str2*) **mod** 4))
$\wedge$ (((nat-to-uint (*str1*) **mod** 4) $\neq$ 0)
  $\vee$ ((nat-to-uint (*str2*) **mod** 4) $\neq$ 0))
$\wedge$ disjoint (*x*, *k*, sub (32, 16, read-sp (*s*)), 32))
$\rightarrow$ (read-mem (*x*, mc-mem (stepn (*s*, 24)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

THEOREM: memmove-s-s0-2
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
$\wedge$ ((nat-to-uint (*str1*) **mod** 4) = (nat-to-uint (*str2*) **mod** 4))
$\wedge$ ($n \leq 3$)
$\wedge$ (((nat-to-uint (*str1*) **mod** 4) $\neq$ 0)
  $\vee$ ((nat-to-uint (*str2*) **mod** 4) $\neq$ 0)))
$\rightarrow$ memmove-s0p (stepn (*s*, 27), 0, 0, *str1*, 0, *lst1*, *str2*, *lst2*, *n*, *n*)

THEOREM: memmove-s-s0-else-2
let *s0* **be** stepn (*s*, 27)
in
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
$\wedge$ ((nat-to-uint (*str1*) **mod** 4) = (nat-to-uint (*str2*) **mod** 4))
$\wedge$ ($n \leq 3$)
$\wedge$ (((nat-to-uint (*str1*) **mod** 4) $\neq$ 0)
  $\vee$ ((nat-to-uint (*str2*) **mod** 4) $\neq$ 0)))
$\rightarrow$ ((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, 12, 3)
    = readm-rn (32, '(2 3 4), mc-rfile (*s*)))) **endlet**

THEOREM: memmove-s-s0-rfile-2
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)

$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
$\wedge$ ((nat-to-uint (*str1*) **mod** 4) = (nat-to-uint (*str2*) **mod** 4))
$\wedge$ $(n \leq 3)$
$\wedge$ (((nat-to-uint (*str1*) **mod** 4) $\neq$ 0)
$\vee$ ((nat-to-uint (*str2*) **mod** 4) $\neq$ 0))
$\wedge$ d5-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 27)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s-s0-mem-2
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
$\wedge$ ((nat-to-uint (*str1*) **mod** 4) = (nat-to-uint (*str2*) **mod** 4))
$\wedge$ $(n \leq 3)$
$\wedge$ (((nat-to-uint (*str1*) **mod** 4) $\neq$ 0)
$\vee$ ((nat-to-uint (*str2*) **mod** 4) $\neq$ 0))
$\wedge$ disjoint (*x*, *k*, sub (32, 16, read-sp (*s*)), 32))
$\rightarrow$ (read-mem (*x*, mc-mem (stepn (*s*, 27)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

THEOREM: memmove-s-s0-3
let *r* **be** nat-to-uint (*str1*) **mod** 4
in
(memmove-statep (*s*, *str1*, *n*, *lst1*, *str2*, *lst2*)
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
$\wedge$ ((nat-to-uint (*str1*) **mod** 4) = (nat-to-uint (*str2*) **mod** 4))
$\wedge$ $(3 < n)$
$\wedge$ (((nat-to-uint (*str1*) **mod** 4) $\neq$ 0)
$\vee$ ((nat-to-uint (*str2*) **mod** 4) $\neq$ 0)))
$\rightarrow$ memmove-s0p (stepn (*s*, 29),
0,
0,
str1,
$(n + r) - 4$,
lst1,
str2,
lst2,
$4 - r$,
n) **endlet**

THEOREM: memmove-s-s0-else-3

```
let s0 be stepn (s, 29)
in
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧  (n ≠ 0)
 ∧  (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧  (nat-to-uint (str1) < nat-to-uint (str2))
 ∧  ((nat-to-uint (str1) mod 4) = (nat-to-uint (str2) mod 4))
 ∧  (3 < n)
 ∧  (((nat-to-uint (str1) mod 4) ≠ 0)
     ∨  ((nat-to-uint (str2) mod 4) ≠ 0)))
→  ((linked-rts-addr (s0) = rts-addr (s))
     ∧  (linked-a6 (s0) = read-an (32, 6, s))
     ∧  (read-rn (32, 14, mc-rfile (s0))
          =  sub (32, 4, read-sp (s)))
     ∧  (movem-saved (s0, 4, 12, 3)
          =  readm-rn (32, '(2 3 4), mc-rfile (s)))) endlet
```

THEOREM: memmove-s-s0-rfile-3

```
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧  (n ≠ 0)
 ∧  (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧  (nat-to-uint (str1) < nat-to-uint (str2))
 ∧  ((nat-to-uint (str1) mod 4) = (nat-to-uint (str2) mod 4))
 ∧  (3 < n)
 ∧  (((nat-to-uint (str1) mod 4) ≠ 0)
     ∨  ((nat-to-uint (str2) mod 4) ≠ 0))
 ∧  d5-7a2-5p (rn))
→  (read-rn (oplen, rn, mc-rfile (stepn (s, 29)))
     =  read-rn (oplen, rn, mc-rfile (s)))
```

THEOREM: memmove-s-s0-mem-3

```
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧  (n ≠ 0)
 ∧  (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧  (nat-to-uint (str1) < nat-to-uint (str2))
 ∧  ((nat-to-uint (str1) mod 4) = (nat-to-uint (str2) mod 4))
 ∧  (3 < n)
 ∧  (((nat-to-uint (str1) mod 4) ≠ 0)
     ∨  ((nat-to-uint (str2) mod 4) ≠ 0))
 ∧  disjoint (x, k, sub (32, 16, read-sp (s)), 32))
→  (read-mem (x, mc-mem (stepn (s, 29)), k) = read-mem (x, mc-mem (s), k))
```

```
; from the initial state s to s1:  s --> s1.
```

THEOREM: memmove-s-s1
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
 $\wedge$ ($n \not\simeq$ 0)
 $\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
 $\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
 $\wedge$ ((nat-to-uint (*str1*) **mod** 4) = 0)
 $\wedge$ ((nat-to-uint (*str2*) **mod** 4) = 0)
 $\wedge$ ($n \not< 4$))
$\rightarrow$ memmove-s1p (stepn (s, 19), 0, 0, *str1*, n, *lst1*, *str2*, *lst2*, $n \div 4$, n)

THEOREM: memmove-s-s1-else
let *s1* **be** stepn (s, 19)
in
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
 $\wedge$ ($n \not\simeq$ 0)
 $\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
 $\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
 $\wedge$ ((nat-to-uint (*str1*) **mod** 4) = 0)
 $\wedge$ ((nat-to-uint (*str2*) **mod** 4) = 0)
 $\wedge$ ($n \not< 4$))
$\rightarrow$ ((linked-rts-addr (*s1*) = rts-addr (s))
 $\wedge$ (linked-a6 (*s1*) = read-an (32, 6, s))
 $\wedge$ (read-rn (32, 14, mc-rfile (*s1*))
 = sub (32, 4, read-sp (s)))
 $\wedge$ (movem-saved (*s1*, 4, 12, 3)
 = readm-rn (32, '(2 3 4), mc-rfile (s)))) **endlet**

THEOREM: memmove-s-s1-rfile
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
 $\wedge$ ($n \not\simeq$ 0)
 $\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
 $\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
 $\wedge$ ((nat-to-uint (*str1*) **mod** 4) = 0)
 $\wedge$ ((nat-to-uint (*str2*) **mod** 4) = 0)
 $\wedge$ ($n \not< 4$)
 $\wedge$ d5-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 19)))
 = read-rn (*oplen*, *rn*, mc-rfile (s)))

THEOREM: memmove-s-s1-mem
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
 $\wedge$ ($n \not\simeq$ 0)
 $\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
 $\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
 $\wedge$ ((nat-to-uint (*str1*) **mod** 4) = 0)

$\wedge$ ((nat-to-uint ($str2$) **mod** 4) = 0)
$\wedge$ ($n \not< 4$)
$\wedge$ disjoint (x, k, sub (32, 16, read-sp (s)), 32))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 19)), k) = read-mem (x, mc-mem (s), k))

```
; from s to s2.  s --> s2.
```

THEOREM: memmove-s-s2
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $<$ nat-to-uint ($str2$))
$\wedge$ ((nat-to-uint ($str1$) **mod** 4) = 0)
$\wedge$ ((nat-to-uint ($str2$) **mod** 4) = 0)
$\wedge$ ($n < 4$))
$\rightarrow$ memmove-s2p (stepn (s, 22), 0, 0, $str1$, n, $lst1$, $str2$, $lst2$, n, n)

THEOREM: memmove-s-s2-else
let $s2$ **be** stepn (s, 22)
in
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $<$ nat-to-uint ($str2$))
$\wedge$ ((nat-to-uint ($str1$) **mod** 4) = 0)
$\wedge$ ((nat-to-uint ($str2$) **mod** 4) = 0)
$\wedge$ ($n < 4$))
$\rightarrow$ ((linked-rts-addr ($s2$) = rts-addr (s))
$\wedge$ (linked-a6 ($s2$) = read-an (32, 6, s))
$\wedge$ (read-rn (32, 14, mc-rfile ($s2$))
= sub (32, 4, read-sp (s)))
$\wedge$ (movem-saved ($s2$, 4, 12, 3)
= readm-rn (32, '(2 3 4), mc-rfile (s)))) **endlet**

THEOREM: memmove-s-s2-rfile
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $<$ nat-to-uint ($str2$))
$\wedge$ ((nat-to-uint ($str1$) **mod** 4) = 0)
$\wedge$ ((nat-to-uint ($str2$) **mod** 4) = 0)
$\wedge$ ($n < 4$)
$\wedge$ d5-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 22)))
= read-rn ($oplen$, rn, mc-rfile (s)))

THEOREM: memmove-s-s2-mem
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $<$ nat-to-uint (*str2*))
$\wedge$ ((nat-to-uint (*str1*) **mod** 4) $= 0$)
$\wedge$ ((nat-to-uint (*str2*) **mod** 4) $= 0$)
$\wedge$ ($n < 4$)
$\wedge$ disjoint (x, k, sub (32, 16, read-sp (s)), 32))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 22)), k) $=$ read-mem (x, mc-mem (s), k))

```
; s0 --> s1.
```

THEOREM: memmove-s0-s1
let *s1* **be** stepn (s, 6)
in
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ (($nt - 1$) $= 0$)
$\wedge$ ($n \not< 4$))
$\rightarrow$ memmove-s1p (*s1*,
add (32, $i*$, 1),
$1 + i$,
str1,
n,
put-nth (get-nth (i, *lst2*), i, *lst1*),
str2,
lst2,
$n \div 4$,
$n_$) **endlet**

THEOREM: memmove-s0-s1-else
let *s1* **be** stepn (s, 6)
in
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ (($nt - 1$) $= 0$)
$\wedge$ ($n \not< 4$))
$\rightarrow$ ((linked-rts-addr (*s1*) $=$ linked-rts-addr (s))
$\wedge$ (linked-a6 (*s1*) $=$ linked-a6 (s))
$\wedge$ (read-rn (*oplen*, 14, mc-rfile (*s1*))
$=$ read-rn (*oplen*, 14, mc-rfile (s)))
$\wedge$ (movem-saved (*s1*, 4, 12, 3) $=$ movem-saved (s, 4, 12, 3))) **endlet**

THEOREM: memmove-s0-s1-rfile
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ (($nt - 1$) $= 0$)

```
 ∧   (n ≮ 4)
 ∧   d5-7a2-5p (rn))
 →   (read-rn (oplen, rn, mc-rfile (stepn (s, 6)))
       =   read-rn (oplen, rn, mc-rfile (s)))
```

THEOREM: memmove-s0-s1-mem

```
(memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   (n ≮ 4)
 ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧   disjoint (x, k, str1, n_))
 →   (read-mem (x, mc-mem (stepn (s, 6)), k) = read-mem (x, mc-mem (s), k))
```

```
; s0 --> s2.
```

THEOREM: memmove-s0-s2

```
let s2  be  stepn (s, 9)
in
(memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   (n < 4)
 ∧   (n ≠ 0))
 →   memmove-s2p (s2,
                  add (32, i*, 1),
                  1 + i,
                  str1,
                  n,
                  put-nth (get-nth (i, lst2), i, lst1),
                  str2,
                  lst2,
                  n,
                  n_) endlet
```

THEOREM: memmove-s0-s2-else

```
let s2  be  stepn (s, 9)
in
(memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   (n < 4)
 ∧   (n ≠ 0))
 →   ((linked-rts-addr (s2) = linked-rts-addr (s))
       ∧   (linked-a6 (s2) = linked-a6 (s))
       ∧   (read-rn (oplen, 14, mc-rfile (s2))
             =   read-rn (oplen, 14, mc-rfile (s)))
       ∧   (movem-saved (s2, 4, 12, 3) = movem-saved (s, 4, 12, 3))) endlet
```

THEOREM: memmove-s0-s2-rfile
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ $((nt - 1) = 0)$
$\wedge$ $(n < 4)$
$\wedge$ $(n \not\simeq 0)$
$\wedge$ d5-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 9)))
$=$ read-rn ($oplen$, rn, mc-rfile (s)))

THEOREM: memmove-s0-s2-mem
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ $((nt - 1) = 0)$
$\wedge$ $(n < 4)$
$\wedge$ $(n \not\simeq 0)$
$\wedge$ disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
$\wedge$ disjoint (x, k, *str1*, $n_$))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 9)), k) $=$ read-mem (x, mc-mem (s), k))

```
; s0 --> s0.
```

THEOREM: memmove-s0-s0
let *s0* **be** stepn (s, 3)
in
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ $((nt - 1) \neq 0))$
$\rightarrow$ (memmove-s0p (*s0*,
add (32, $i*$, 1),
$1 + i$,
str1,
n,
put-nth (get-nth (i, *lst2*), i, *lst1*),
str2,
lst2,
$nt - 1$,
$n_$)
$\wedge$ (linked-rts-addr (*s0*) $=$ linked-rts-addr (s))
$\wedge$ (linked-a6 (*s0*) $=$ linked-a6 (s))
$\wedge$ (read-rn ($oplen$, 14, mc-rfile (*s0*))
$=$ read-rn ($oplen$, 14, mc-rfile (s)))
$\wedge$ (movem-saved (*s0*, 4, 12, 3) $=$ movem-saved (s, 4, 12, 3))) **endlet**

THEOREM: memmove-s0-s0-rfile
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ $((nt - 1) \neq 0)$
$\wedge$ d5-7a2-5p (rn))

```
→   (read-rn (oplen, rn, mc-rfile (stepn (s, 3)))
      =   read-rn (oplen, rn, mc-rfile (s)))
```

THEOREM: memmove-s0-s0-mem

```
(memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) ≠ 0)
 ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧   disjoint (x, k, str1, n_))
 →   (read-mem (x, mc-mem (stepn (s, 3)), k) = read-mem (x, mc-mem (s), k))
```

```
; s1 --> s2.
```

THEOREM: memmove-s1-s2

```
let s2 be stepn (s, 6)
in
(memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   ((n mod 4) ≠ 0))
 →   memmove-s2p (s2,
                  add (32, i*, 4),
                  4 + i,
                  str1,
                  n,
                  movn-lst (4, lst1, lst2, i),
                  str2,
                  lst2,
                  n mod 4,
                  n_) endlet
```

THEOREM: memmove-s1-s2-else

```
let s2 be stepn (s, 6)
in
(memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   ((n mod 4) ≠ 0))
 →   ((linked-rts-addr (s2) = linked-rts-addr (s))
      ∧   (linked-a6 (s2) = linked-a6 (s))
      ∧   (read-rn (oplen, 14, mc-rfile (s2))
            =   read-rn (oplen, 14, mc-rfile (s)))
      ∧   (movem-saved (s2, 4, 12, 3) = movem-saved (s, 4, 12, 3))) endlet
```

THEOREM: memmove-s1-s2-rfile

```
(memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   ((n mod 4) ≠ 0)
```

```
 ∧   d5-7a2-5p (rn))
 →   (read-rn (oplen, rn, mc-rfile (stepn (s, 6)))
       =   read-rn (oplen, rn, mc-rfile (s)))
```

THEOREM: memmove-s1-s2-mem

```
 (memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) = 0)
  ∧   ((n mod 4) ≠ 0)
  ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
  ∧   disjoint (x, k, str1, n_))
  →   (read-mem (x, mc-mem (stepn (s, 6)), k) = read-mem (x, mc-mem (s), k))
```

```
; s1 --> s1.
```

THEOREM: memmove-s1-s1

```
 let s1  be  stepn (s, 3)
 in
 (memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) ≠ 0))
  →   (memmove-s1p (s1,
                    add (32, i*, 4),
                    4 + i,
                    str1,
                    n,
                    movn-lst (4, lst1, lst2, i),
                    str2,
                    lst2,
                    nt − 1,
                    n_)
       ∧   (linked-rts-addr (s1) = linked-rts-addr (s))
       ∧   (linked-a6 (s1) = linked-a6 (s))
       ∧   (read-rn (oplen, 14, mc-rfile (s1))
             =   read-rn (oplen, 14, mc-rfile (s)))
       ∧   (movem-saved (s1, 4, 12, 3) = movem-saved (s, 4, 12, 3))) endlet
```

THEOREM: memmove-s1-s1-rfile

```
 (memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) ≠ 0)
  ∧   d5-7a2-5p (rn))
  →   (read-rn (oplen, rn, mc-rfile (stepn (s, 3)))
       =   read-rn (oplen, rn, mc-rfile (s)))
```

THEOREM: memmove-s1-s1-mem

```
 (memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) ≠ 0)
```

```
 ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧   disjoint (x, k, str1, n_))
 →   (read-mem (x, mc-mem (stepn (s, 3)), k) = read-mem (x, mc-mem (s), k))

; from s2 to exit:  s2 --> sn.
; base case:  s2 --> sn.
```

THEOREM: memmove-s2-sn-base

```
 let sn  be  stepn (s, 8)
 in
 (memmove-s2p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) = 0))
  →   ((mc-status (sn) = 'running)
       ∧   (mc-pc (sn) = linked-rts-addr (s))
       ∧   (read-dn (32, 0, sn) = str1)
       ∧   mem-lst (1,
                    str1,
                    mc-mem (sn),
                    n_,
                    put-nth (get-nth (i, lst2), i, lst1))
       ∧   (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
       ∧   (read-rn (32, 15, mc-rfile (sn))
            =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s2-sn-rfile-base

```
 (memmove-s2p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) = 0)
  ∧   d2-7a2-5p (rn)
  ∧   (oplen ≤ 32))
  →   (read-rn (oplen, rn, mc-rfile (stepn (s, 8)))
       =   if d5-7a2-5p (rn)  then read-rn (oplen, rn, mc-rfile (s))
           else get-vlst (oplen,
                          0,
                          rn,
                          '(2 3 4),
                          movem-saved (s, 4, 12, 3)) endif)
```

THEOREM: memmove-s2-sn-mem-base

```
 (memmove-s2p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) = 0)
  ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
  ∧   disjoint (x, k, str1, n_))
  →   (read-mem (x, mc-mem (stepn (s, 8)), k) = read-mem (x, mc-mem (s), k))

; induction case:  s2 --> s2.
```

THEOREM: memmove-s2-s2
let *s2* **be** stepn (*s*, 3)
in
(memmove-s2p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) ≠ 0))
→ (memmove-s2p (*s2*,
add (32, *i**, 1),
1 + *i*,
str1,
n,
put-nth (get-nth (*i*, *lst2*), *i*, *lst1*),
str2,
lst2,
nt − 1,
*n_*)
∧ (linked-rts-addr (*s2*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s2*) = linked-a6 (*s*))
∧ (read-rn (*oplen*, 14, mc-rfile (*s2*))
= read-rn (*oplen*, 14, mc-rfile (*s*)))
∧ (movem-saved (*s2*, 4, 12, 3) = movem-saved (*s*, 4, 12, 3))) **endlet**

THEOREM: memmove-s2-s2-rfile
(memmove-s2p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) ≠ 0)
∧ d5-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 3)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s2-s2-mem
(memmove-s2p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) ≠ 0)
∧ disjoint (*x*, *k*, sub (32, 12, read-an (32, 6, *s*)), 32)
∧ disjoint (*x*, *k*, *str1*, *n_*))
→ (read-mem (*x*, mc-mem (stepn (*s*, 3)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; put together: s2 --> sn.
```

DEFINITION:
memmove-s2-sn-t (*i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
= **if** (*nt* − 1) = 0 **then** 8
else splus (3,
memmove-s2-sn-t (add (32, *i**, 1),
1 + *i*,
str1,
n,

put-nth (get-nth (i, *lst2*), i, *lst1*),
str2,
lst2,
$nt - 1$,
$n_$)) **endif**

DEFINITION:
memmove-s2-sn-induct (s, $i*$, i, *lst1*, *lst2*, nt)
= **if** $(nt - 1) = 0$ **then t**
else memmove-s2-sn-induct (stepn (s, 3),
add (32, $i*$, 1),
$1 + i$,
put-nth (get-nth (i, *lst2*), i, *lst1*),
lst2,
$nt - 1$) **endif**

THEOREM: memmove-s2p-info
memmove-s2p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
→ $((nt \in \mathbf{N}) \wedge (nt \neq 0))$

THEOREM: memmove-s2-sn
let *sn* **be** stepn (s, memmove-s2-sn-t ($i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$))
in
memmove-s2p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
→ ((mc-status (*sn*) = 'running)
∧ (mc-pc (*sn*) = linked-rts-addr (s))
∧ (read-dn (32, 0, *sn*) = *str1*)
∧ mem-lst (1,
str1,
mc-mem (*sn*),
$n_$,
mmov1-lst (i, *lst1*, *lst2*, nt))
∧ (read-rn (32, 14, mc-rfile (*sn*)) = linked-a6 (s))
∧ (read-rn (32, 15, mc-rfile (*sn*))
= add (32, read-an (32, 6, s), 8))) **endlet**

THEOREM: memmove-s2-sn-rfile
let *sn* **be** stepn (s, memmove-s2-sn-t ($i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$))
in
(memmove-s2p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
∧ d2-7a2-5p (*rn*)
∧ $(oplen \leq 32)$)
→ (read-rn (*oplen*, *rn*, mc-rfile (*sn*))
= **if** d5-7a2-5p (*rn*) **then** read-rn (*oplen*, *rn*, mc-rfile (s))
else get-vlst (*oplen*,

```
                    0,
                    rn,
                    '(2 3 4),
                    movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s2-sn-mem

```
let sn be stepn (s, memmove-s2-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s2p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧  disjoint (x, k, str1, n_))
 →  (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet
```

EVENT: Disable memmove-s2p-info.

```
; from s1 to sn: s1 --> sn.
; base case 1: s1 --> sn.
```

THEOREM: memmove-s1-sn-base-1

```
let sn be stepn (s, 10)
in
(memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  ((n mod 4) = 0))
 →  ((mc-status (sn) = 'running)
     ∧  (mc-pc (sn) = linked-rts-addr (s))
     ∧  (read-dn (32, 0, sn) = str1)
     ∧  mem-lst (1, str1, mc-mem (sn), n_, movn-lst (4, lst1, lst2, i))
     ∧  (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
     ∧  (read-rn (32, 15, mc-rfile (sn))
         =  add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s1-sn-rfile-base-1

```
(memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  ((n mod 4) = 0)
 ∧  d2-7a2-5p (rn)
 ∧  (oplen ≤ 32))
 →  (read-rn (oplen, rn, mc-rfile (stepn (s, 10)))
     =  if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
        else get-vlst (oplen,
                       0,
                       rn,
                       '(2 3 4),
                       movem-saved (s, 4, 12, 3)) endif)
```

THEOREM: memmove-s1-sn-mem-base-1
(memmove-s1p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ (($nt - 1$) = 0)
$\wedge$ ((n **mod** 4) = 0)
$\wedge$ disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
$\wedge$ disjoint (x, k, *str1*, $n_$))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 10)), k) = read-mem (x, mc-mem (s), k))

```
; base case 2: s1 --> s2 --> sn.
```

DEFINITION:
memmove-s1-sn-t0 ($i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
= splus (6,
memmove-s2-sn-t (add (32, $i*$, 4),
4 + i,
str1,
n,
movn-lst (4, *lst1*, *lst2*, i),
str2,
lst2,
n **mod** 4,
$n_$))

THEOREM: memmove-s1-sn-base-2
let *sn* **be** stepn (s, memmove-s1-sn-t0 ($i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$))
in
(memmove-s1p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
$\wedge$ (($nt - 1$) = 0)
$\wedge$ ((n **mod** 4) $\neq$ 0))
$\rightarrow$ ((mc-status (*sn*) = `'running`)
$\wedge$ (mc-pc (*sn*) = linked-rts-addr (s))
$\wedge$ (read-dn (32, 0, *sn*) = *str1*)
$\wedge$ mem-lst (1,
str1,
mc-mem (*sn*),
$n_$,
mmov1-lst (4 + i,
movn-lst (4, *lst1*, *lst2*, i),
lst2,
n **mod** 4))
$\wedge$ (read-rn (32, 14, mc-rfile (*sn*)) = linked-a6 (s))
$\wedge$ (read-rn (32, 15, mc-rfile (*sn*))
= add (32, read-an (32, 6, s), 8))) **endlet**

THEOREM: memmove-s1-sn-rfile-base-2

```
let sn be stepn(s, memmove-s1-sn-t0(i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s1p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  ((n mod 4) ≠ 0)
 ∧  d2-7a2-5p(rn)
 ∧  (oplen ≤ 32))
→  (read-rn(oplen, rn, mc-rfile(sn))
     =  if d5-7a2-5p(rn) then read-rn(oplen, rn, mc-rfile(s))
        else get-vlst(oplen,
                      0,
                      rn,
                      '(2 3 4),
                      movem-saved(s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s1-sn-mem-base-2

```
let sn be stepn(s, memmove-s1-sn-t0(i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s1p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  ((n mod 4) ≠ 0)
 ∧  disjoint(x, k, sub(32, 12, read-an(32, 6, s)), 32)
 ∧  disjoint(x, k, str1, n_))
→  (read-mem(x, mc-mem(sn), k) = read-mem(x, mc-mem(s), k)) endlet
```

```
; put together:  s1 --> s2.
```

DEFINITION:

```
memmove-s1-sn-t(i*, i, str1, n, lst1, str2, lst2, nt, n_)
=  if (nt − 1) = 0
   then if (n mod 4) = 0 then 10
        else memmove-s1-sn-t0(i*, i, str1, n, lst1, str2, lst2, nt, n_) endif
   else splus(3,
              memmove-s1-sn-t(add(32, i*, 4),
                              4 + i,
                              str1,
                              n,
                              movn-lst(4, lst1, lst2, i),
                              str2,
                              lst2,
                              nt − 1,
                              n_)) endif
```

DEFINITION:

```
memmove-s1-sn-induct(s, i*, i, lst1, lst2, nt)
```

```
=   if (nt − 1) = 0 then t
    else memmove-s1-sn-induct (stepn (s, 3),
                               add (32, i*, 4),
                               4 + i,
                               movn-lst (4, lst1, lst2, i),
                               lst2,
                               nt − 1) endif
```

THEOREM: memmove-s1p-info

```
memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
→   ((nt ∈ N) ∧ (nt ≠ 0))
```

THEOREM: memmove-s1-sn

```
let sn be stepn (s, memmove-s1-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
→   ((mc-status (sn) = 'running)
     ∧  (mc-pc (sn) = linked-rts-addr (s))
     ∧  (read-dn (32, 0, sn) = str1)
     ∧  mem-lst (1,
                 str1,
                 mc-mem (sn),
                 n_,
                 memmove-1 (lst1, lst2, i, nt, n))
     ∧  (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
     ∧  (read-rn (32, 15, mc-rfile (sn))
         =  add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s1-sn-rfile

```
let sn be stepn (s, memmove-s1-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s1p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  d2-7a2-5p (rn)
 ∧  (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (sn))
     =  if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
        else get-vlst (oplen,
                       0,
                       rn,
                       '(2 3 4),
                       movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s1-sn-mem

```
let sn be stepn (s, memmove-s1-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
```

(memmove-s1p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
 ∧ disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧ disjoint (x, k, *str1*, $n_$))
→ (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) **endlet**

EVENT: Disable memmove-s1p-info.

```
; from s0 to sn: s0 --> sn.
; base case 1: s0 --> sn.
```

THEOREM: memmove-s0-sn-base-1
let sn **be** stepn (s, 13)
in
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
 ∧ ((nt − 1) = 0)
 ∧ (n ≃ 0))
→ ((mc-status (sn) = `'running`)
 ∧ (mc-pc (sn) = linked-rts-addr (s))
 ∧ (read-dn (32, 0, sn) = *str1*)
 ∧ mem-lst (1,
 str1,
 mc-mem (sn),
 $n_$,
 put-nth (get-nth (i, *lst2*), i, *lst1*))
 ∧ (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
 ∧ (read-rn (32, 15, mc-rfile (sn))
 = add (32, read-an (32, 6, s), 8))) **endlet**

THEOREM: memmove-s0-sn-rfile-base-1
let sn **be** stepn (s, 13)
in
(memmove-s0p (s, $i*$, i, *str1*, n, *lst1*, *str2*, *lst2*, nt, $n_$)
 ∧ ((nt − 1) = 0)
 ∧ (n ≃ 0)
 ∧ d2-7a2-5p (rn)
 ∧ ($oplen$ ≤ 32))
→ (read-rn ($oplen$, rn, mc-rfile (sn))
 = **if** d5-7a2-5p (rn) **then** read-rn ($oplen$, rn, mc-rfile (s))
 else get-vlst ($oplen$,
 0,
 rn,
 `'(2 3 4)`,
 movem-saved (s, 4, 12, 3)) **endif**) **endlet**

THEOREM: memmove-s0-sn-mem-base-1
let *sn* **be** stepn (*s*, 13)
in
(memmove-s0p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
 ∧ ((*nt* − 1) = 0)
 ∧ (*n* ≃ 0)
 ∧ disjoint (*x*, *k*, sub (32, 12, read-an (32, 6, *s*)), 32)
 ∧ disjoint (*x*, *k*, *str1*, *n_*))
 → (read-mem (*x*, mc-mem (*sn*), *k*) = read-mem (*x*, mc-mem (*s*), *k*)) **endlet**

```
; base case 2: s0 --> s1 --> sn.
```

DEFINITION:
memmove-s0-sn-t0 (*i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
= splus (6,
 memmove-s1-sn-t (add (32, *i**, 1),
 1 + *i*,
 str1,
 n,
 put-nth (get-nth (*i*, *lst2*), *i*, *lst1*),
 str2,
 lst2,
 n ÷ 4,
 *n_*))

THEOREM: memmove-s0-sn-base-2
let *sn* **be** stepn (*s*, memmove-s0-sn-t0 (*i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*))
in
(memmove-s0p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
 ∧ ((*nt* − 1) = 0)
 ∧ (*n* ≮ 4))
 → ((mc-status (*sn*) = `'running`)
 ∧ (mc-pc (*sn*) = linked-rts-addr (*s*))
 ∧ (read-dn (32, 0, *sn*) = *str1*)
 ∧ mem-lst (1,
 str1,
 mc-mem (*sn*),
 *n_*,
 memmove-1 (put-nth (get-nth (*i*, *lst2*), *i*, *lst1*),
 lst2,
 1 + *i*,
 n ÷ 4,
 n))
 ∧ (read-rn (32, 14, mc-rfile (*sn*)) = linked-a6 (*s*))
 ∧ (read-rn (32, 15, mc-rfile (*sn*))

```
        =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s0-sn-rfile-base-2

```
 let sn be stepn (s, memmove-s0-sn-t0 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
 in
 (memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) = 0)
  ∧   (n ≮ 4)
  ∧   d2-7a2-5p (rn)
  ∧   (oplen ≤ 32))
 →   (read-rn (oplen, rn, mc-rfile (sn))
       =   if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
           else get-vlst (oplen,
                          0,
                          rn,
                          '(2 3 4),
                          movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s0-sn-mem-base-2

```
 let sn be stepn (s, memmove-s0-sn-t0 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
 in
 (memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) = 0)
  ∧   (n ≮ 4)
  ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
  ∧   disjoint (x, k, str1, n_))
 →   (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet
```

```
; base case 3: s0 --> s2 --> sn.
```

DEFINITION:

```
memmove-s0-sn-t1 (i*, i, str1, n, lst1, str2, lst2, nt, n_)
=   splus (9,
           memmove-s2-sn-t (add (32, i*, 1),
                            1 + i,
                            str1,
                            n,
                            put-nth (get-nth (i, lst2), i, lst1),
                            str2,
                            lst2,
                            n,
                            n_))
```

THEOREM: memmove-s0-sn-base-3

```
 let sn be stepn (s, memmove-s0-sn-t1 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
```

```
in
(memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   (n < 4)
 ∧   (n ≠ 0))
→   ((mc-status (sn) = 'running)
      ∧   (mc-pc (sn) = linked-rts-addr (s))
      ∧   (read-dn (32, 0, sn) = str1)
      ∧   mem-lst (1,
                   str1,
                   mc-mem (sn),
                   n_,
                   mmov1-lst (1 + i,
                              put-nth (get-nth (i, lst2), i, lst1),
                              lst2,
                              n))
      ∧   (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
      ∧   (read-rn (32, 15, mc-rfile (sn))
           =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s0-sn-rfile-base-3

```
let sn be stepn (s, memmove-s0-sn-t1 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   (n < 4)
 ∧   (n ≠ 0)
 ∧   d2-7a2-5p (rn)
 ∧   (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (sn))
      =   if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
          else get-vlst (oplen,
                         0,
                         rn,
                         '(2 3 4),
                         movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s0-sn-mem-base-3

```
let sn be stepn (s, memmove-s0-sn-t1 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   (n < 4)
 ∧   (n ≠ 0)
```

```
  ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
  ∧   disjoint (x, k, str1, n_))
  →   (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet

; put together: s0 --> sn.

DEFINITION:
memmove-s0-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_)
=   if (nt − 1) = 0
    then if n < 4
         then if n ≃ 0 then 13
              else memmove-s0-sn-t1 (i*,
                                     i,
                                     str1,
                                     n,
                                     lst1,
                                     str2,
                                     lst2,
                                     nt,
                                     n_) endif
         else memmove-s0-sn-t0 (i*, i, str1, n, lst1, str2, lst2, nt, n_) endif
    else splus (3,
                memmove-s0-sn-t (add (32, i*, 1),
                                 1 + i,
                                 str1,
                                 n,
                                 put-nth (get-nth (i, lst2), i, lst1),
                                 str2,
                                 lst2,
                                 nt − 1,
                                 n_)) endif

THEOREM: memmove-s0p-info
 memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 →   ((nt ∈ N) ∧ (nt ≠ 0))

THEOREM: memmove-s0-sn
 let sn be stepn (s, memmove-s0-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
 in
 memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 →   ((mc-status (sn) = 'running)
      ∧   (mc-pc (sn) = linked-rts-addr (s))
      ∧   (read-dn (32, 0, sn) = str1)
      ∧   mem-lst (1,
                   str1,
```

```
                    mc-mem (sn),
                    n_,
                    memmove-0 (lst1, lst2, i, nt, n))
    ∧   (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
    ∧   (read-rn (32, 15, mc-rfile (sn))
          =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s0-sn-rfile

```
 let sn  be  stepn (s, memmove-s0-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
 in
 (memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   d2-7a2-5p (rn)
  ∧   (oplen ≤ 32))
 →   (read-rn (oplen, rn, mc-rfile (sn))
       =   if d5-7a2-5p (rn)  then read-rn (oplen, rn, mc-rfile (s))
           else get-vlst (oplen,
                          0,
                          rn,
                          '(2 3 4),
                          movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s0-sn-mem

```
 let sn  be  stepn (s, memmove-s0-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
 in
 (memmove-s0p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
  ∧   disjoint (x, k, str1, n_))
 →   (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet
```

EVENT: Disable memmove-s0p-info.

```
; from the initial state s to s3:  s --> s3.
```

THEOREM: memmove-s-s3-1

```
 let x1  be  nat-to-uint (str1) + n,
     x2  be  nat-to-uint (str2) + n
 in
 (memmove-statep (s, str1, n, lst1, str2, lst2)
  ∧   (n ≠̸ 0)
  ∧   (nat-to-uint (str1) ≠ nat-to-uint (str2))
  ∧   (nat-to-uint (str1) ≮ nat-to-uint (str2))
  ∧   ((x1 mod 4) ≠ (x2 mod 4))
  ∧   (((x1 mod 4) ≠ 0) ∨ ((x2 mod 4) ≠ 0)))
 →   memmove-s3p (stepn (s, 26),
```

uint-to-nat (n),
n,
str1,
0,
lst1,
str2,
lst2,
n,
n) **endlet**

THEOREM: memmove-s-s3-else-1
let *x1* **be** nat-to-uint (*str1*) + n,
x2 **be** nat-to-uint (*str2*) + n,
s3 **be** stepn (s, 26)
in
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $\not<$ nat-to-uint (*str2*))
$\wedge$ ((*x1* **mod** 4) $\neq$ (*x2* **mod** 4))
$\wedge$ (((*x1* **mod** 4) $\neq$ 0) $\vee$ ((*x2* **mod** 4) $\neq$ 0)))
$\rightarrow$ ((linked-rts-addr (*s3*) = rts-addr (s))
$\wedge$ (linked-a6 (*s3*) = read-an (32, 6, s))
$\wedge$ (read-rn (32, 14, mc-rfile (*s3*))
= sub (32, 4, read-sp (s)))
$\wedge$ (movem-saved (*s3*, 4, 12, 3)
= readm-rn (32, '(2 3 4), mc-rfile (s)))) **endlet**

THEOREM: memmove-s-s3-rfile-1
let *x1* **be** nat-to-uint (*str1*) + n,
x2 **be** nat-to-uint (*str2*) + n
in
(memmove-statep (s, *str1*, n, *lst1*, *str2*, *lst2*)
$\wedge$ ($n \not\simeq 0$)
$\wedge$ (nat-to-uint (*str1*) $\neq$ nat-to-uint (*str2*))
$\wedge$ (nat-to-uint (*str1*) $\not<$ nat-to-uint (*str2*))
$\wedge$ ((*x1* **mod** 4) $\neq$ (*x2* **mod** 4))
$\wedge$ (((*x1* **mod** 4) $\neq$ 0) $\vee$ ((*x2* **mod** 4) $\neq$ 0))
$\wedge$ d5-7a2-5p (*rn*))
$\rightarrow$ (read-rn (*oplen*, *rn*, mc-rfile (stepn (s, 26)))
= read-rn (*oplen*, *rn*, mc-rfile (s))) **endlet**

THEOREM: memmove-s-s3-mem-1
let *x1* **be** nat-to-uint (*str1*) + n,
x2 **be** nat-to-uint (*str2*) + n

```
in
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧   (n ≉ 0)
 ∧   (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧   (nat-to-uint (str1) ≮ nat-to-uint (str2))
 ∧   ((x1 mod 4) ≠ (x2 mod 4))
 ∧   (((x1 mod 4) ≠ 0) ∨ ((x2 mod 4) ≠ 0))
 ∧   disjoint (x, k, sub (32, 16, read-sp (s)), 32))
 →   (read-mem (x, mc-mem (stepn (s, 26)), k)
       =   read-mem (x, mc-mem (s), k)) endlet
```

THEOREM: memmove-s-s3-2

```
let x1  be  nat-to-uint (str1) + n,
    x2  be  nat-to-uint (str2) + n
in
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧   (n ≉ 0)
 ∧   (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧   (nat-to-uint (str1) ≮ nat-to-uint (str2))
 ∧   ((x1 mod 4) = (x2 mod 4))
 ∧   (n ≤ 4)
 ∧   (((x1 mod 4) ≠ 0) ∨ ((x2 mod 4) ≠ 0)))
 →   memmove-s3p (stepn (s, 29),
                  uint-to-nat (n),
                  n,
                  str1,
                  0,
                  lst1,
                  str2,
                  lst2,
                  n,
                  n) endlet
```

THEOREM: memmove-s-s3-else-2

```
let x1  be  nat-to-uint (str1) + n,
    x2  be  nat-to-uint (str2) + n,
    s3  be  stepn (s, 29)
in
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧   (n ≉ 0)
 ∧   (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧   (nat-to-uint (str1) ≮ nat-to-uint (str2))
 ∧   ((x1 mod 4) = (x2 mod 4))
 ∧   (n ≤ 4)
```

$\wedge$ $(((x1 \textbf{ mod } 4) \neq 0) \vee ((x2 \textbf{ mod } 4) \neq 0)))$
$\rightarrow$ ((linked-rts-addr $(s3)$ = rts-addr (s))
$\wedge$ (linked-a6 $(s3)$ = read-an $(32, 6, s)$)
$\wedge$ (read-rn $(32, 14,$ mc-rfile $(s3))$
= sub $(32, 4,$ read-sp $(s)))$
$\wedge$ (movem-saved $(s3, 4, 12, 3)$
= readm-rn $(32,$ '(2 3 4), mc-rfile $(s))))$ **endlet**

THEOREM: memmove-s-s3-rfile-2
let $x1$ **be** nat-to-uint $(str1) + n$,
$x2$ **be** nat-to-uint $(str2) + n$
in
(memmove-statep $(s, str1, n, lst1, str2, lst2)$
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint $(str1) \neq$ nat-to-uint $(str2))$
$\wedge$ (nat-to-uint $(str1) \not< $ nat-to-uint $(str2))$
$\wedge$ $((x1 \textbf{ mod } 4) = (x2 \textbf{ mod } 4))$
$\wedge$ $(n \leq 4)$
$\wedge$ $(((x1 \textbf{ mod } 4) \neq 0) \vee ((x2 \textbf{ mod } 4) \neq 0))$
$\wedge$ d5-7a2-5p $(rn))$
$\rightarrow$ (read-rn $(oplen, rn,$ mc-rfile (stepn $(s, 29)))$
= read-rn $(oplen, rn,$ mc-rfile $(s)))$ **endlet**

THEOREM: memmove-s-s3-mem-2
let $x1$ **be** nat-to-uint $(str1) + n$,
$x2$ **be** nat-to-uint $(str2) + n$
in
(memmove-statep $(s, str1, n, lst1, str2, lst2)$
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint $(str1) \neq$ nat-to-uint $(str2))$
$\wedge$ (nat-to-uint $(str1) \not< $ nat-to-uint $(str2))$
$\wedge$ $((x1 \textbf{ mod } 4) = (x2 \textbf{ mod } 4))$
$\wedge$ $(n \leq 4)$
$\wedge$ $(((x1 \textbf{ mod } 4) \neq 0) \vee ((x2 \textbf{ mod } 4) \neq 0))$
$\wedge$ disjoint $(x, k,$ sub $(32, 16,$ read-sp $(s)), 32))$
$\rightarrow$ (read-mem $(x,$ mc-mem (stepn $(s, 29)), k)$
= read-mem $(x,$ mc-mem $(s), k))$ **endlet**

THEOREM: memmove-s-s3-3
let $x1$ **be** nat-to-uint $(str1) + n$,
$x2$ **be** nat-to-uint $(str2) + n$,
nt **be** $(n +$ nat-to-uint $(str1))$ **mod** 4
in
(memmove-statep $(s, str1, n, lst1, str2, lst2)$
$\wedge$ $(n \not\simeq 0)$

$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $\not<$ nat-to-uint ($str2$))
$\wedge$ (($x1$ **mod** 4) = ($x2$ **mod** 4))
$\wedge$ (4 < n)
$\wedge$ ((($x1$ **mod** 4) $\neq$ 0) $\vee$ (($x2$ **mod** 4) $\neq$ 0)))
$\rightarrow$ memmove-s3p (stepn (s, 29),
uint-to-nat (n),
n,
$str1$,
$n - nt$,
$lst1$,
$str2$,
$lst2$,
nt,
n) **endlet**

THEOREM: memmove-s-s3-else-3
let $x1$ **be** nat-to-uint ($str1$) + n,
$x2$ **be** nat-to-uint ($str2$) + n,
$s3$ **be** stepn (s, 29)
in
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$)
$\wedge$ ($n \not\simeq$ 0)
$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $\not<$ nat-to-uint ($str2$))
$\wedge$ (($x1$ **mod** 4) = ($x2$ **mod** 4))
$\wedge$ (4 < n)
$\wedge$ ((($x1$ **mod** 4) $\neq$ 0) $\vee$ (($x2$ **mod** 4) $\neq$ 0)))
$\rightarrow$ ((linked-rts-addr ($s3$) = rts-addr (s))
$\wedge$ (linked-a6 ($s3$) = read-an (32, 6, s))
$\wedge$ (read-rn (32, 14, mc-rfile ($s3$))
= sub (32, 4, read-sp (s)))
$\wedge$ (movem-saved ($s3$, 4, 12, 3)
= readm-rn (32, '(2 3 4), mc-rfile (s)))) **endlet**

THEOREM: memmove-s-s3-rfile-3
let $x1$ **be** nat-to-uint ($str1$) + n,
$x2$ **be** nat-to-uint ($str2$) + n
in
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$)
$\wedge$ ($n \not\simeq$ 0)
$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $\not<$ nat-to-uint ($str2$))
$\wedge$ (($x1$ **mod** 4) = ($x2$ **mod** 4))

$\wedge$ $(4 < n)$
$\wedge$ $(((x1 \textbf{ mod } 4) \neq 0) \vee ((x2 \textbf{ mod } 4) \neq 0))$
$\wedge$ d5-7a2-5p $(rn))$
$\rightarrow$ (read-rn $(oplen,\ rn,$ mc-rfile (stepn $(s,\ 29)))$
$=$ read-rn $(oplen,\ rn,$ mc-rfile $(s)))$ **endlet**

THEOREM: memmove-s-s3-mem-3
let $x1$ **be** nat-to-uint $(str1) + n$,
$x2$ **be** nat-to-uint $(str2) + n$
in
(memmove-statep $(s,\ str1,\ n,\ lst1,\ str2,\ lst2)$
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint $(str1) \neq$ nat-to-uint $(str2))$
$\wedge$ (nat-to-uint $(str1) \not< $ nat-to-uint $(str2))$
$\wedge$ $((x1 \textbf{ mod } 4) = (x2 \textbf{ mod } 4))$
$\wedge$ $(4 < n)$
$\wedge$ $(((x1 \textbf{ mod } 4) \neq 0) \vee ((x2 \textbf{ mod } 4) \neq 0))$
$\wedge$ disjoint $(x,\ k,$ sub $(32,\ 16,$ read-sp $(s)),\ 32))$
$\rightarrow$ (read-mem $(x,$ mc-mem (stepn $(s,\ 29)),\ k)$
$=$ read-mem $(x,$ mc-mem $(s),\ k))$ **endlet**

```
; from the initial state s to s4:  s --> s4.
```

THEOREM: memmove-s-s4
(memmove-statep $(s,\ str1,\ n,\ lst1,\ str2,\ lst2)$
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint $(str1) \neq$ nat-to-uint $(str2))$
$\wedge$ (nat-to-uint $(str1) \not< $ nat-to-uint $(str2))$
$\wedge$ $(((n +$ nat-to-uint $(str1)) \textbf{ mod } 4) = 0)$
$\wedge$ $(((n +$ nat-to-uint $(str2)) \textbf{ mod } 4) = 0)$
$\wedge$ $(n \not< 4))$
$\rightarrow$ memmove-s4p (stepn $(s,\ 21)$,
uint-to-nat (n),
n,
$str1$,
n,
$lst1$,
$str2$,
$lst2$,
$n \div 4$,
$n)$

THEOREM: memmove-s-s4-else
let $s4$ **be** stepn $(s,\ 21)$
in

```
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧   (n ≉ 0)
 ∧   (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧   (nat-to-uint (str1) ≮ nat-to-uint (str2))
 ∧   (((n + nat-to-uint (str1)) mod 4) = 0)
 ∧   (((n + nat-to-uint (str2)) mod 4) = 0)
 ∧   (n ≮ 4))
→    ((linked-rts-addr (s4) = rts-addr (s))
      ∧   (linked-a6 (s4) = read-an (32, 6, s))
      ∧   (read-rn (32, 14, mc-rfile (s4))
            =   sub (32, 4, read-sp (s)))
      ∧   (movem-saved (s4, 4, 12, 3)
            =   readm-rn (32, '(2 3 4), mc-rfile (s)))) endlet
```

THEOREM: memmove-s-s4-rfile

```
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧   (n ≉ 0)
 ∧   (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧   (nat-to-uint (str1) ≮ nat-to-uint (str2))
 ∧   (((n + nat-to-uint (str1)) mod 4) = 0)
 ∧   (((n + nat-to-uint (str2)) mod 4) = 0)
 ∧   (n ≮ 4)
 ∧   d5-7a2-5p (rn))
→    (read-rn (oplen, rn, mc-rfile (stepn (s, 21)))
      =   read-rn (oplen, rn, mc-rfile (s)))
```

THEOREM: memmove-s-s4-mem

```
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧   (n ≉ 0)
 ∧   (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧   (nat-to-uint (str1) ≮ nat-to-uint (str2))
 ∧   (((n + nat-to-uint (str1)) mod 4) = 0)
 ∧   (((n + nat-to-uint (str2)) mod 4) = 0)
 ∧   (n ≮ 4)
 ∧   disjoint (x, k, sub (32, 16, read-sp (s)), 32))
→    (read-mem (x, mc-mem (stepn (s, 21)), k) = read-mem (x, mc-mem (s), k))
```

```
; from s to s5.  s --> s5.
```

THEOREM: memmove-s-s5

```
(memmove-statep (s, str1, n, lst1, str2, lst2)
 ∧   (n ≉ 0)
 ∧   (nat-to-uint (str1) ≠ nat-to-uint (str2))
 ∧   (nat-to-uint (str1) ≮ nat-to-uint (str2))
 ∧   (((n + nat-to-uint (str1)) mod 4) = 0)
```

$\wedge$ $(((n + \text{nat-to-uint}(str2))\ \textbf{mod}\ 4) = 0)$
$\wedge$ $(n < 4))$
$\rightarrow$ memmove-s5p (stepn (s, 24), uint-to-nat (n), n, $str1$, n, $lst1$, $str2$, $lst2$, n, n)

THEOREM: memmove-s-s5-else
let $s0$ **be** stepn (s, 24)
in
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$)
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $\not<$ nat-to-uint ($str2$))
$\wedge$ $(((n + \text{nat-to-uint}(str1))\ \textbf{mod}\ 4) = 0)$
$\wedge$ $(((n + \text{nat-to-uint}(str2))\ \textbf{mod}\ 4) = 0)$
$\wedge$ $(n < 4))$
$\rightarrow$ ((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, 12, 3)
= readm-rn (32, '(2 3 4), mc-rfile (s)))) **endlet**

THEOREM: memmove-s-s5-rfile
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$)
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $\not<$ nat-to-uint ($str2$))
$\wedge$ $(((n + \text{nat-to-uint}(str1))\ \textbf{mod}\ 4) = 0)$
$\wedge$ $(((n + \text{nat-to-uint}(str2))\ \textbf{mod}\ 4) = 0)$
$\wedge$ $(n < 4)$
$\wedge$ d5-7a2-5p (rn))
$\rightarrow$ (read-rn ($oplen$, rn, mc-rfile (stepn (s, 24)))
= read-rn ($oplen$, rn, mc-rfile (s)))

THEOREM: memmove-s-s5-mem
(memmove-statep (s, $str1$, n, $lst1$, $str2$, $lst2$)
$\wedge$ $(n \not\simeq 0)$
$\wedge$ (nat-to-uint ($str1$) $\neq$ nat-to-uint ($str2$))
$\wedge$ (nat-to-uint ($str1$) $\not<$ nat-to-uint ($str2$))
$\wedge$ $(((n + \text{nat-to-uint}(str1))\ \textbf{mod}\ 4) = 0)$
$\wedge$ $(((n + \text{nat-to-uint}(str2))\ \textbf{mod}\ 4) = 0)$
$\wedge$ $(n < 4)$
$\wedge$ disjoint (x, k, sub (32, 16, read-sp (s)), 32))
$\rightarrow$ (read-mem (x, mc-mem (stepn (s, 24)), k) = read-mem (x, mc-mem (s), k))

```
; s3 --> s4.
```

THEOREM: memmove-s3-s4
let *s4* **be** stepn (*s*, 6)
in
(memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0)
∧ (*n* ≮ 4))
→ memmove-s4p (*s4*,
sub (32, 1, *i**),
i − 1,
str1,
n,
put-nth (get-nth (*i* − 1, *lst2*), *i* − 1, *lst1*),
str2,
lst2,
n ÷ 4,
*n_*) **endlet**

THEOREM: memmove-s3-s4-else
let *s4* **be** stepn (*s*, 6)
in
(memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0)
∧ (*n* ≮ 4))
→ ((linked-rts-addr (*s4*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s4*) = linked-a6 (*s*))
∧ (read-rn (*oplen*, 14, mc-rfile (*s4*))
= read-rn (*oplen*, 14, mc-rfile (*s*)))
∧ (movem-saved (*s4*, 4, 12, 3) = movem-saved (*s*, 4, 12, 3))) **endlet**

THEOREM: memmove-s3-s4-rfile-base
(memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0)
∧ (*n* ≮ 4)
∧ d5-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 6)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s3-s4-mem-base
(memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0)
∧ (*n* ≮ 4)
∧ disjoint (*x*, *k*, sub (32, 12, read-an (32, 6, *s*)), 32)
∧ disjoint (*x*, *k*, *str1*, *n_*))
→ (read-mem (*x*, mc-mem (stepn (*s*, 6)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; s3 --> s5.
```

THEOREM: memmove-s3-s5
let *s5* **be** stepn (*s*, 9)
in
(memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
 ∧ ((*nt* − 1) = 0)
 ∧ (*n* < 4)
 ∧ (*n* ≠ 0))
→ memmove-s5p (*s5*,
 sub (32, 1, *i**),
 i − 1,
 str1,
 n,
 put-nth (get-nth (*i* − 1, *lst2*), *i* − 1, *lst1*),
 str2,
 lst2,
 n,
 *n_*) **endlet**

THEOREM: memmove-s3-s5-else
let *s5* **be** stepn (*s*, 9)
in
(memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
 ∧ ((*nt* − 1) = 0)
 ∧ (*n* < 4)
 ∧ (*n* ≠ 0))
→ ((linked-rts-addr (*s5*) = linked-rts-addr (*s*))
 ∧ (linked-a6 (*s5*) = linked-a6 (*s*))
 ∧ (read-rn (*oplen*, 14, mc-rfile (*s5*))
 = read-rn (*oplen*, 14, mc-rfile (*s*)))
 ∧ (movem-saved (*s5*, 4, 12, 3) = movem-saved (*s*, 4, 12, 3))) **endlet**

THEOREM: memmove-s3-s5-rfile
(memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
 ∧ ((*nt* − 1) = 0)
 ∧ (*n* < 4)
 ∧ (*n* ≠ 0)
 ∧ d5-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 9)))
 = read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s3-s5-mem
(memmove-s3p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
 ∧ ((*nt* − 1) = 0)

$\wedge$ $(n < 4)$
$\wedge$ $(n \not\simeq 0)$
$\wedge$ disjoint $(x,\ k,\ \text{sub}\,(32,\ 12,\ \text{read-an}\,(32,\ 6,\ s)),\ 32)$
$\wedge$ disjoint $(x,\ k,\ str1,\ n_))$
$\rightarrow$ $(\text{read-mem}\,(x,\ \text{mc-mem}\,(\text{stepn}\,(s,\ 9)),\ k) = \text{read-mem}\,(x,\ \text{mc-mem}\,(s),\ k))$

```
; s3 --> s3.
```

THEOREM: memmove-s3-s3
let *s3* **be** stepn $(s,\ 3)$
in
(memmove-s3p $(s,\ i*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$
$\wedge$ $((nt - 1) \neq 0))$
$\rightarrow$ (memmove-s3p $(s3,$
sub $(32,\ 1,\ i*)$,
$i - 1$,
str1,
n,
put-nth $(\text{get-nth}\,(i - 1,\ lst2),\ i - 1,\ lst1)$,
str2,
lst2,
$nt - 1$,
$n_)$
$\wedge$ $(\text{linked-rts-addr}\,(s3) = \text{linked-rts-addr}\,(s))$
$\wedge$ $(\text{linked-a6}\,(s3) = \text{linked-a6}\,(s))$
$\wedge$ $(\text{read-rn}\,(oplen,\ 14,\ \text{mc-rfile}\,(s3))$
$= \text{read-rn}\,(oplen,\ 14,\ \text{mc-rfile}\,(s)))$
$\wedge$ $(\text{movem-saved}\,(s3,\ 4,\ 12,\ 3) = \text{movem-saved}\,(s,\ 4,\ 12,\ 3)))$ **endlet**

THEOREM: memmove-s3-s3-rfile
(memmove-s3p $(s,\ i*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$
$\wedge$ $((nt - 1) \neq 0)$
$\wedge$ d5-7a2-5p $(rn))$
$\rightarrow$ $(\text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(\text{stepn}\,(s,\ 3)))$
$= \text{read-rn}\,(oplen,\ rn,\ \text{mc-rfile}\,(s)))$

THEOREM: memmove-s3-s3-mem
(memmove-s3p $(s,\ i*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$
$\wedge$ $((nt - 1) \neq 0)$
$\wedge$ disjoint $(x,\ k,\ \text{sub}\,(32,\ 12,\ \text{read-an}\,(32,\ 6,\ s)),\ 32)$
$\wedge$ disjoint $(x,\ k,\ str1,\ n_))$
$\rightarrow$ $(\text{read-mem}\,(x,\ \text{mc-mem}\,(\text{stepn}\,(s,\ 3)),\ k) = \text{read-mem}\,(x,\ \text{mc-mem}\,(s),\ k))$

```
; s4 --> s5.
```

THEOREM: memmove-s4-s5
let *s5* **be** stepn (*s*, 6)
in
(memmove-s4p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0)
∧ ((*n* **mod** 4) ≠ 0))
→ memmove-s5p (*s5*,
sub (32, 4, *i**),
i − 4,
str1,
n,
movn-lst (4, *lst1*, *lst2*, *i* − 4),
str2,
lst2,
n **mod** 4,
*n_*) **endlet**

THEOREM: memmove-s4-s5-else
let *s5* **be** stepn (*s*, 6)
in
(memmove-s4p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0)
∧ ((*n* **mod** 4) ≠ 0))
→ ((linked-rts-addr (*s5*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s5*) = linked-a6 (*s*))
∧ (read-rn (*oplen*, 14, mc-rfile (*s5*))
= read-rn (*oplen*, 14, mc-rfile (*s*)))
∧ (movem-saved (*s5*, 4, 12, 3) = movem-saved (*s*, 4, 12, 3))) **endlet**

THEOREM: memmove-s4-s5-rfile
(memmove-s4p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0)
∧ ((*n* **mod** 4) ≠ 0)
∧ d5-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 6)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s4-s5-mem
(memmove-s4p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0)
∧ ((*n* **mod** 4) ≠ 0)
∧ disjoint (*x*, *k*, sub (32, 12, read-an (32, 6, *s*)), 32)
∧ disjoint (*x*, *k*, *str1*, *n_*))
→ (read-mem (*x*, mc-mem (stepn (*s*, 6)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; s4 --> s4.
```

THEOREM: memmove-s4-s4
let *s4* **be** stepn (*s*, 3)
in
(memmove-s4p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) ≠ 0))
→ (memmove-s4p (*s4*,
sub (32, 4, *i**),
i − 4,
str1,
n,
movn-lst (4, *lst1*, *lst2*, *i* − 4),
str2,
lst2,
nt − 1,
*n_*)
∧ (linked-rts-addr (*s4*) = linked-rts-addr (*s*))
∧ (linked-a6 (*s4*) = linked-a6 (*s*))
∧ (read-rn (*oplen*, 14, mc-rfile (*s4*))
= read-rn (*oplen*, 14, mc-rfile (*s*)))
∧ (movem-saved (*s4*, 4, 12, 3) = movem-saved (*s*, 4, 12, 3))) **endlet**

THEOREM: memmove-s4-s4-rfile
(memmove-s4p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) ≠ 0)
∧ d5-7a2-5p (*rn*))
→ (read-rn (*oplen*, *rn*, mc-rfile (stepn (*s*, 3)))
= read-rn (*oplen*, *rn*, mc-rfile (*s*)))

THEOREM: memmove-s4-s4-mem
(memmove-s4p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) ≠ 0)
∧ disjoint (*x*, *k*, sub (32, 12, read-an (32, 6, *s*)), 32)
∧ disjoint (*x*, *k*, *str1*, *n_*))
→ (read-mem (*x*, mc-mem (stepn (*s*, 3)), *k*) = read-mem (*x*, mc-mem (*s*), *k*))

```
; from s5 to exit:  s5 --> sn.
; base case: s5 --> sn.
```

THEOREM: memmove-s5-sn-base
let *sn* **be** stepn (*s*, 7)
in
(memmove-s5p (*s*, *i**, *i*, *str1*, *n*, *lst1*, *str2*, *lst2*, *nt*, *n_*)
∧ ((*nt* − 1) = 0))

```
→   ((mc-status (sn) = 'running)
     ∧  (mc-pc (sn) = linked-rts-addr (s))
     ∧  (read-dn (32, 0, sn) = str1)
     ∧  mem-lst (1,
                 str1,
                 mc-mem (sn),
                 n_,
                 put-nth (get-nth (i − 1, lst2), i − 1, lst1))
     ∧  (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
     ∧  (read-rn (32, 15, mc-rfile (sn))
         =  add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s5-sn-rfile-base

```
(memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  d2-7a2-5p (rn)
 ∧  (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (stepn (s, 7)))
     =  if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
        else get-vlst (oplen,
                       0,
                       rn,
                       '(2 3 4),
                       movem-saved (s, 4, 12, 3)) endif)
```

THEOREM: memmove-s5-sn-mem-base

```
(memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧  disjoint (x, k, str1, n_))
→   (read-mem (x, mc-mem (stepn (s, 7)), k) = read-mem (x, mc-mem (s), k))
```

```
; induction case:  s5 --> s5.
```

THEOREM: memmove-s5-s5

```
let s5 be stepn (s, 3)
in
(memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) ≠ 0))
→   (memmove-s5p (s5,
                  sub (32, 1, i*),
                  i − 1,
                  str1,
                  n,
                  put-nth (get-nth (i − 1, lst2), i − 1, lst1),
```

```
                    str2,
                    lst2,
                    nt − 1,
                    n_)
    ∧   (linked-rts-addr (s5) = linked-rts-addr (s))
    ∧   (linked-a6 (s5) = linked-a6 (s))
    ∧   (read-rn (oplen, 14, mc-rfile (s5))
          =   read-rn (oplen, 14, mc-rfile (s)))
    ∧   (movem-saved (s5, 4, 12, 3) = movem-saved (s, 4, 12, 3))) endlet
```

THEOREM: memmove-s5-s5-rfile

```
 (memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) ≠ 0)
  ∧   d5-7a2-5p (rn))
  →   (read-rn (oplen, rn, mc-rfile (stepn (s, 3)))
        =   read-rn (oplen, rn, mc-rfile (s)))
```

THEOREM: memmove-s5-s5-mem

```
 (memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) ≠ 0)
  ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
  ∧   disjoint (x, k, str1, n_))
  →   (read-mem (x, mc-mem (stepn (s, 3)), k) = read-mem (x, mc-mem (s), k))
```

```
; put together: s5 --> sn.
```

DEFINITION:

```
memmove-s5-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_)
=   if (nt − 1) = 0  then 7
    else splus (3,
                memmove-s5-sn-t (sub (32, 1, i*),
                                 i − 1,
                                 str1,
                                 n,
                                 put-nth (get-nth (i − 1, lst2),
                                          i − 1,
                                          lst1),
                                 str2,
                                 lst2,
                                 nt − 1,
                                 n_)) endif
```

DEFINITION:

```
memmove-s5-sn-induct (s, i*, i, lst1, lst2, nt)
=   if (nt − 1) = 0  then t
```

```
else memmove-s5-sn-induct (stepn (s, 3),
                           sub (32, 1, i*),
                           i − 1,
                           put-nth (get-nth (i − 1, lst2),
                                    i − 1,
                                    lst1),
                           lst2,
                           nt − 1) endif
```

THEOREM: memmove-s5p-info

```
memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
→   ((nt ∈ N) ∧ (nt ≠ 0))
```

THEOREM: memmove-s5-sn

```
let sn be stepn (s, memmove-s5-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
→   ((mc-status (sn) = 'running)
     ∧   (mc-pc (sn) = linked-rts-addr (s))
     ∧   (read-dn (32, 0, sn) = str1)
     ∧   mem-lst (1,
                  str1,
                  mc-mem (sn),
                  n_,
                  mmov1-lst1 (i, lst1, lst2, nt))
     ∧   (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
     ∧   (read-rn (32, 15, mc-rfile (sn))
          =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s5-sn-rfile

```
let sn be stepn (s, memmove-s5-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   d2-7a2-5p (rn)
 ∧   (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (sn))
     =   if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
         else get-vlst (oplen,
                        0,
                        rn,
                        '(2 3 4),
                        movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s5-sn-mem

```
let sn be stepn (s, memmove-s5-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
```

```
in
(memmove-s5p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧   disjoint (x, k, str1, n_))
 →   (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet
```

EVENT: Disable memmove-s5p-info.

```
; from s4 to sn: s4 --> sn.
; base case 1: s4 --> sn.
```

THEOREM: memmove-s4-sn-base-1

```
let sn be stepn (s, 10)
in
(memmove-s4p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   ((n mod 4) = 0))
 →   ((mc-status (sn) = 'running)
      ∧   (mc-pc (sn) = linked-rts-addr (s))
      ∧   (read-dn (32, 0, sn) = str1)
      ∧   mem-lst (1,
                   str1,
                   mc-mem (sn),
                   n_,
                   movn-lst (4, lst1, lst2, i − 4))
      ∧   (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
      ∧   (read-rn (32, 15, mc-rfile (sn))
          =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s4-sn-rfile-base-1

```
(memmove-s4p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   ((n mod 4) = 0)
 ∧   d2-7a2-5p (rn)
 ∧   (oplen ≤ 32))
 →   (read-rn (oplen, rn, mc-rfile (stepn (s, 10)))
      =   if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
          else get-vlst (oplen,
                         0,
                         rn,
                         '(2 3 4),
                         movem-saved (s, 4, 12, 3)) endif)
```

THEOREM: memmove-s4-sn-mem-base-1

```
(memmove-s4p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   ((n mod 4) = 0)
 ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧   disjoint (x, k, str1, n_))
 →   (read-mem (x, mc-mem (stepn (s, 10)), k) = read-mem (x, mc-mem (s), k))
```

```
; base case 2: s4 --> s5 --> sn.
```

DEFINITION:

```
memmove-s4-sn-t0 (i*, i, str1, n, lst1, str2, lst2, nt, n_)
=   splus (6,
          memmove-s5-sn-t (sub (32, 4, i*),
                           i − 4,
                           str1,
                           n,
                           movn-lst (4, lst1, lst2, i − 4),
                           str2,
                           lst2,
                           n mod 4,
                           n_))
```

THEOREM: memmove-s4-sn-base-2

```
let sn be stepn (s, memmove-s4-sn-t0 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s4p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   ((n mod 4) ≠ 0))
 →   ((mc-status (sn) = 'running)
      ∧   (mc-pc (sn) = linked-rts-addr (s))
      ∧   (read-dn (32, 0, sn) = str1)
      ∧   mem-lst (1,
                   str1,
                   mc-mem (sn),
                   n_,
                   mmov1-lst1 (i − 4,
                               movn-lst (4, lst1, lst2, i − 4),
                               lst2,
                               n mod 4))
      ∧   (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
      ∧   (read-rn (32, 15, mc-rfile (sn))
          =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s4-sn-rfile-base-2

```
let sn be stepn (s, memmove-s4-sn-t0 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
```

```
in
(memmove-s4p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  ((n mod 4) ≠ 0)
 ∧  d2-7a2-5p(rn)
 ∧  (oplen ≤ 32))
 →  (read-rn(oplen, rn, mc-rfile(sn))
      =  if d5-7a2-5p(rn) then read-rn(oplen, rn, mc-rfile(s))
         else get-vlst(oplen,
                       0,
                       rn,
                       '(2 3 4),
                       movem-saved(s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s4-sn-mem-base-2

```
let sn be stepn(s, memmove-s4-sn-t0(i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s4p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  ((n mod 4) ≠ 0)
 ∧  disjoint(x, k, sub(32, 12, read-an(32, 6, s)), 32)
 ∧  disjoint(x, k, str1, n_))
 →  (read-mem(x, mc-mem(sn), k) = read-mem(x, mc-mem(s), k)) endlet
```

```
; put together:  s4 --> sn.
```

DEFINITION:

```
memmove-s4-sn-t(i*, i, str1, n, lst1, str2, lst2, nt, n_)
=  if (nt − 1) = 0
   then if (n mod 4) = 0 then 10
        else memmove-s4-sn-t0(i*, i, str1, n, lst1, str2, lst2, nt, n_) endif
   else splus(3,
              memmove-s4-sn-t(sub(32, 4, i*),
                              i − 4,
                              str1,
                              n,
                              movn-lst(4, lst1, lst2, i − 4),
                              str2,
                              lst2,
                              nt − 1,
                              n_)) endif
```

DEFINITION:

```
memmove-s4-sn-induct(s, i*, i, lst1, lst2, nt)
=  if (nt − 1) = 0 then t
```

```
else memmove-s4-sn-induct (stepn (s, 3),
                           sub (32, 4, i*),
                           i − 4,
                           movn-lst (4, lst1, lst2, i − 4),
                           lst2,
                           nt − 1) endif
```

THEOREM: memmove-s4p-info

```
memmove-s4p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
→   ((nt ∈ N) ∧ (nt ≠ 0))
```

THEOREM: memmove-s4-sn

```
let sn be stepn (s, memmove-s4-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
memmove-s4p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
→   ((mc-status (sn) = 'running)
     ∧  (mc-pc (sn) = linked-rts-addr (s))
     ∧  (read-dn (32, 0, sn) = str1)
     ∧  mem-lst (1,
                 str1,
                 mc-mem (sn),
                 n_,
                 memmove-4 (lst1, lst2, i, nt, n))
     ∧  (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
     ∧  (read-rn (32, 15, mc-rfile (sn))
         =  add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s4-sn-rfile

```
let sn be stepn (s, memmove-s4-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s4p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  d2-7a2-5p (rn)
 ∧  (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (sn))
     =  if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
        else get-vlst (oplen,
                       0,
                       rn,
                       '(2 3 4),
                       movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s4-sn-mem

```
let sn be stepn (s, memmove-s4-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s4p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
```

```
 ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧   disjoint (x, k, str1, n_))
 →   (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet
```

EVENT: Disable memmove-s4p-info.

```
; from s3 to sn: s3 --> sn.
; base case 1: s3 --> sn.
```

THEOREM: memmove-s3-sn-base-1

```
 let sn be stepn (s, 13)
 in
 (memmove-s3p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) = 0)
  ∧   (n ≃ 0))
  →   ((mc-status (sn) = 'running)
       ∧   (mc-pc (sn) = linked-rts-addr (s))
       ∧   (read-dn (32, 0, sn) = str1)
       ∧   mem-lst (1,
                    str1,
                    mc-mem (sn),
                    n_,
                    put-nth (get-nth (i − 1, lst2), i − 1, lst1))
       ∧   (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
       ∧   (read-rn (32, 15, mc-rfile (sn))
            =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s3-sn-rfile-base-1

```
 let sn be stepn (s, 13)
 in
 (memmove-s3p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
  ∧   ((nt − 1) = 0)
  ∧   (n ≃ 0)
  ∧   d2-7a2-5p (rn)
  ∧   (oplen ≤ 32))
  →   (read-rn (oplen, rn, mc-rfile (sn))
       =   if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
           else get-vlst (oplen,
                          0,
                          rn,
                          '(2 3 4),
                          movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s3-sn-mem-base-1

```
let sn be stepn(s, 13)
in
(memmove-s3p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  (n ≃ 0)
 ∧  disjoint(x, k, sub(32, 12, read-an(32, 6, s)), 32)
 ∧  disjoint(x, k, str1, n_))
 →  (read-mem(x, mc-mem(sn), k) = read-mem(x, mc-mem(s), k)) endlet
```

```
; base case 2: s3 --> s4 --> sn.
```

DEFINITION:

```
memmove-s3-sn-t0(i*, i, str1, n, lst1, str2, lst2, nt, n_)
=  splus(6,
         memmove-s4-sn-t(sub(32, 1, i*),
                         i − 1,
                         str1,
                         n,
                         put-nth(get-nth(i − 1, lst2), i − 1, lst1),
                         str2,
                         lst2,
                         n ÷ 4,
                         n_))
```

THEOREM: memmove-s3-sn-base-2

```
let sn be stepn(s, memmove-s3-sn-t0(i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s3p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  (n ≮ 4))
 →  ((mc-status(sn) = 'running)
     ∧  (mc-pc(sn) = linked-rts-addr(s))
     ∧  (read-dn(32, 0, sn) = str1)
     ∧  mem-lst(1,
                str1,
                mc-mem(sn),
                n_,
                memmove-4(put-nth(get-nth(i − 1, lst2),
                                  i − 1,
                                  lst1),
                          lst2,
                          i − 1,
                          n ÷ 4,
                          n))
     ∧  (read-rn(32, 14, mc-rfile(sn)) = linked-a6(s))
```

```
    ∧   (read-rn (32, 15, mc-rfile (sn))
          =   add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s3-sn-rfile-base-2

```
let sn be stepn (s, memmove-s3-sn-t0 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s3p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   (n ≮ 4)
 ∧   d2-7a2-5p (rn)
 ∧   (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (sn))
      =   if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
          else get-vlst (oplen,
                         0,
                         rn,
                         '(2 3 4),
                         movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s3-sn-mem-base-2

```
let sn be stepn (s, memmove-s3-sn-t0 (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s3p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧   ((nt − 1) = 0)
 ∧   (n ≮ 4)
 ∧   disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧   disjoint (x, k, str1, n_))
→   (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet
```

```
; base case 3: s3 --> s5 --> sn.
```

DEFINITION:

```
memmove-s3-sn-t1 (i*, i, str1, n, lst1, str2, lst2, nt, n_)
=   splus (9,
           memmove-s5-sn-t (sub (32, 1, i*),
                            i − 1,
                            str1,
                            n,
                            put-nth (get-nth (i − 1, lst2), i − 1, lst1),
                            str2,
                            lst2,
                            n,
                            n_))
```

THEOREM: memmove-s3-sn-base-3

```
let sn be stepn(s, memmove-s3-sn-t1(i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s3p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  (n < 4)
 ∧  (n ≠ 0))
→  ((mc-status(sn) = 'running)
     ∧  (mc-pc(sn) = linked-rts-addr(s))
     ∧  (read-dn(32, 0, sn) = str1)
     ∧  mem-lst(1,
                str1,
                mc-mem(sn),
                n_,
                mmov1-lst1(i − 1,
                           put-nth(get-nth(i − 1, lst2),
                                   i − 1,
                                   lst1),
                           lst2,
                           n))
     ∧  (read-rn(32, 14, mc-rfile(sn)) = linked-a6(s))
     ∧  (read-rn(32, 15, mc-rfile(sn))
         =  add(32, read-an(32, 6, s), 8))) endlet
```

THEOREM: memmove-s3-sn-rfile-base-3

```
let sn be stepn(s, memmove-s3-sn-t1(i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s3p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  ((nt − 1) = 0)
 ∧  (n < 4)
 ∧  (n ≠ 0)
 ∧  d2-7a2-5p(rn)
 ∧  (oplen ≤ 32))
→  (read-rn(oplen, rn, mc-rfile(sn))
     =  if d5-7a2-5p(rn) then read-rn(oplen, rn, mc-rfile(s))
        else get-vlst(oplen,
                      0,
                      rn,
                      '(2 3 4),
                      movem-saved(s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s3-sn-mem-base-3

```
let sn be stepn(s, memmove-s3-sn-t1(i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s3p(s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
```

$\wedge$ $((nt - 1) = 0)$
$\wedge$ $(n < 4)$
$\wedge$ $(n \not\simeq 0)$
$\wedge$ disjoint $(x,\ k,$ sub $(32,\ 12,$ read-an $(32,\ 6,\ s)),\ 32)$
$\wedge$ disjoint $(x,\ k,\ str1,\ n_))$
$\rightarrow$ (read-mem $(x,$ mc-mem $(sn),\ k) =$ read-mem $(x,$ mc-mem $(s),\ k))$ **endlet**

```
; put together: s3 --> sn.
```

DEFINITION:
memmove-s3-sn-t $(i*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$
= **if** $(nt - 1) = 0$
then if $n < 4$
then if $n \simeq 0$ **then** 13
else memmove-s3-sn-t1 $(i*,$
$i,$
$str1,$
$n,$
$lst1,$
$str2,$
$lst2,$
$nt,$
$n_)$ **endif**
else memmove-s3-sn-t0 $(i*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$ **endif**
else splus $(3,$
memmove-s3-sn-t $($sub $(32,\ 1,\ i*),$
$i - 1,$
$str1,$
$n,$
put-nth $($get-nth $(i - 1,\ lst2),$
$i - 1,$
$lst1),$
$str2,$
$lst2,$
$nt - 1,$
$n_))$ **endif**

THEOREM: memmove-s3p-info
memmove-s3p $(s,\ i*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$
$\rightarrow$ $((nt \in \mathbf{N}) \wedge (nt \neq 0))$

THEOREM: memmove-s3-sn
let sn **be** stepn $(s,$ memmove-s3-sn-t $(i*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_))$
in
memmove-s3p $(s,\ i*,\ i,\ str1,\ n,\ lst1,\ str2,\ lst2,\ nt,\ n_)$

```
→   ((mc-status (sn) = 'running)
     ∧  (mc-pc (sn) = linked-rts-addr (s))
     ∧  (read-dn (32, 0, sn) = str1)
     ∧  mem-lst (1,
                 str1,
                 mc-mem (sn),
                 n_,
                 memmove-3 (lst1, lst2, i, nt, n))
     ∧  (read-rn (32, 14, mc-rfile (sn)) = linked-a6 (s))
     ∧  (read-rn (32, 15, mc-rfile (sn))
         =  add (32, read-an (32, 6, s), 8))) endlet
```

THEOREM: memmove-s3-sn-rfile

```
let sn be stepn (s, memmove-s3-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s3p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  d2-7a2-5p (rn)
 ∧  (oplen ≤ 32))
→   (read-rn (oplen, rn, mc-rfile (sn))
     =  if d5-7a2-5p (rn) then read-rn (oplen, rn, mc-rfile (s))
        else get-vlst (oplen,
                       0,
                       rn,
                       '(2 3 4),
                       movem-saved (s, 4, 12, 3)) endif) endlet
```

THEOREM: memmove-s3-sn-mem

```
let sn be stepn (s, memmove-s3-sn-t (i*, i, str1, n, lst1, str2, lst2, nt, n_))
in
(memmove-s3p (s, i*, i, str1, n, lst1, str2, lst2, nt, n_)
 ∧  disjoint (x, k, sub (32, 12, read-an (32, 6, s)), 32)
 ∧  disjoint (x, k, str1, n_))
→   (read-mem (x, mc-mem (sn), k) = read-mem (x, mc-mem (s), k)) endlet
```

EVENT: Disable memmove-s3p-info.

```
; the correctness of memmove.
```

DEFINITION:

```
memmove-t (str1, n, lst1, str2, lst2)
=   if n ≃ 0 then 11
    else let x1 be nat-to-uint (str1),
             x2 be nat-to-uint (str2)
         in
```

```
if x1 = x2 then 13
elseif x1 < x2
then if ((x1 mod 4) = 0)
        ∧   ((x2 mod 4) = 0)
     then if n < 4
          then splus (22,
                      memmove-s2-sn-t (0,
                                       0,
                                       str1,
                                       n,
                                       lst1,
                                       str2,
                                       lst2,
                                       n,
                                       n))
          else splus (19,
                      memmove-s1-sn-t (0,
                                       0,
                                       str1,
                                       n,
                                       lst1,
                                       str2,
                                       lst2,
                                       n ÷ 4,
                                       n)) endif
     elseif (x1 mod 4) = (x2 mod 4)
     then if 3 < n
          then splus (29,
                      memmove-s0-sn-t (0,
                                       0,
                                       str1,
                                       (n + (x1 mod 4))
                                       −   4,
                                       lst1,
                                       str2,
                                       lst2,
                                       4 − (x1 mod 4),
                                       n))
          else splus (27,
                      memmove-s0-sn-t (0,
                                       0,
                                       str1,
                                       0,
                                       lst1,
```

```
                                        str2,
                                        lst2,
                                        n,
                                        n)) endif
      else splus (24,
                  memmove-s0-sn-t (0,
                                   0,
                                   str1,
                                   0,
                                   lst1,
                                   str2,
                                   lst2,
                                   n,
                                   n)) endif
else let y1  be  n + x1,
         y2  be  n + x2
     in
     if ((y1 mod 4) = 0)
       ∧   ((y2 mod 4) = 0)
     then if n < 4
          then splus (24,
                      memmove-s5-sn-t (uint-to-nat (n),
                                       n,
                                       str1,
                                       n,
                                       lst1,
                                       str2,
                                       lst2,
                                       n,
                                       n))
          else splus (21,
                      memmove-s4-sn-t (uint-to-nat (n),
                                       n,
                                       str1,
                                       n,
                                       lst1,
                                       str2,
                                       lst2,
                                       n ÷ 4,
                                       n)) endif
     elseif (y1 mod 4)
            =   (y2 mod 4)
     then if 4 < n
          then splus (29,
```

```
                    memmove-s3-sn-t (uint-to-nat (n),
                                     n,
                                     str1,
                                     n − (y1 mod 4),
                                     lst1,
                                     str2,
                                     lst2,
                                     y1 mod 4,
                                     n))
          else splus (29,
                      memmove-s3-sn-t (uint-to-nat (n),
                                       n,
                                       str1,
                                       0,
                                       lst1,
                                       str2,
                                       lst2,
                                       n,
                                       n)) endif
     else splus (26,
                 memmove-s3-sn-t (uint-to-nat (n),
                                  n,
                                  str1,
                                  0,
                                  lst1,
                                  str2,
                                  lst2,
                                  n,
                                  n)) endif endlet endif endlet endif
```

THEOREM: memmove-correctness

```
let sn be stepn (s, memmove-t (str1, n, lst1, str2, lst2))
in
memmove-statep (s, str1, n, lst1, str2, lst2)
→   ((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))
     ∧   ((d2-7a2-5p (rn) ∧ (oplen ≤ 32))
           →   (read-rn (oplen, rn, mc-rfile (sn))
                 =   read-rn (oplen, rn, mc-rfile (s))))
     ∧   ((disjoint (x, k, sub (32, 16, read-sp (s)), 32)
```

$\wedge$ disjoint (x, k, *str1*, n))
$\rightarrow$ (read-mem (x, mc-mem (sn), k)
$=$ read-mem (x, mc-mem (s), k)))
$\wedge$ (read-dn (32, 0, sn) $=$ *str1*)
$\wedge$ mem-lst (1,
str1,
mc-mem (sn),
n,
memmove (*str1*, *str2*, n, *lst1*, *lst2*))) **endlet**

EVENT: Disable memmove-t.

```
; some properties of memmove.
; see file cstring.events.
```

EVENT: Make the library "memmove" and compile it.

Index

add, 7–12, 19–36, 49, 51–53, 55, 56, 58, 59, 61, 64

d2-7a2-5p, 12, 24, 26, 27, 29–31, 33, 34, 36, 49, 51, 52, 54–56, 58, 59, 61, 64
d5-7a2-5p, 13, 15–18, 20, 21, 23–27, 29–31, 33, 34, 36, 37, 39, 41–52, 54–56, 58, 59, 61
disjoint, 7–16, 18–25, 27–29, 31–33, 35, 36, 38, 39, 41–44, 46–50, 52–54, 56–58, 60, 61, 64, 65

equal*, 7–11
evenp, 7–10

get-nth, 19–21, 24–26, 31–35, 44–46, 49–51, 56–60
get-vlst, 24, 27, 29–31, 33, 34, 36, 49, 51, 52, 54–56, 58, 59, 61

linked-a6, 13, 14, 16–28, 30–32, 34, 36, 37, 39, 40, 42–53, 55–57, 59, 61
linked-rts-addr, 13, 14, 16–28, 30–32, 34, 35, 37, 39, 40, 42–53, 55–57, 59, 61

mc-mem, 7–16, 18–36, 38, 39, 41–44, 46–61, 65
mc-pc, 7–12, 24, 26–28, 30–32, 34, 35, 49, 51–53, 55–57, 59, 61, 64
mc-rfile, 12–34, 36, 37, 39–59, 61, 64
mc-status, 7–12, 24, 26–28, 30–32, 34, 35, 49, 51–53, 55–57, 59, 61, 64
mcar, 9–11
mcdr, 7–9
mcode-addrp, 7–10
mem-lst, 7–12, 24, 26–28, 30–32, 34, 36, 49, 51–53, 55–57, 59, 61, 65
memmove, 65
memmove-0, 36
memmove-1, 30, 32
memmove-3, 61
memmove-4, 55, 57
memmove-code, 6–10
memmove-correctness, 64
memmove-s-s0-1, 13
memmove-s-s0-2, 14
memmove-s-s0-3, 15
memmove-s-s0-else-1, 13
memmove-s-s0-else-2, 14
memmove-s-s0-else-3, 16
memmove-s-s0-mem-1, 14
memmove-s-s0-mem-2, 15
memmove-s-s0-mem-3, 16
memmove-s-s0-rfile-1, 13
memmove-s-s0-rfile-2, 14
memmove-s-s0-rfile-3, 16
memmove-s-s1, 17
memmove-s-s1-else, 17
memmove-s-s1-mem, 17
memmove-s-s1-rfile, 17
memmove-s-s2, 18
memmove-s-s2-else, 18
memmove-s-s2-mem, 19
memmove-s-s2-rfile, 18
memmove-s-s3-1, 36
memmove-s-s3-2, 38
memmove-s-s3-3, 39
memmove-s-s3-else-1, 37
memmove-s-s3-else-2, 38
memmove-s-s3-else-3, 40
memmove-s-s3-mem-1, 37
memmove-s-s3-mem-2, 39
memmove-s-s3-mem-3, 41
memmove-s-s3-rfile-1, 37
memmove-s-s3-rfile-2, 39

memmove-s-s3-rfile-3, 40
memmove-s-s4, 41
memmove-s-s4-else, 41
memmove-s-s4-mem, 42
memmove-s-s4-rfile, 42
memmove-s-s5, 42
memmove-s-s5-else, 43
memmove-s-s5-mem, 43
memmove-s-s5-rfile, 43
memmove-s-sn-1, 11
memmove-s-sn-2, 12
memmove-s-sn-mem-1, 12
memmove-s-sn-mem-2, 13
memmove-s-sn-rfile-1, 12
memmove-s-sn-rfile-2, 12
memmove-s0-s0, 21
memmove-s0-s0-mem, 22
memmove-s0-s0-rfile, 21
memmove-s0-s1, 19
memmove-s0-s1-else, 19
memmove-s0-s1-mem, 20
memmove-s0-s1-rfile, 19
memmove-s0-s2, 20
memmove-s0-s2-else, 20
memmove-s0-s2-mem, 21
memmove-s0-s2-rfile, 21
memmove-s0-sn, 35
memmove-s0-sn-base-1, 31
memmove-s0-sn-base-2, 32
memmove-s0-sn-base-3, 33
memmove-s0-sn-mem, 36
memmove-s0-sn-mem-base-1, 32
memmove-s0-sn-mem-base-2, 33
memmove-s0-sn-mem-base-3, 34
memmove-s0-sn-rfile, 36
memmove-s0-sn-rfile-base-1, 31
memmove-s0-sn-rfile-base-2, 33
memmove-s0-sn-rfile-base-3, 34
memmove-s0-sn-t, 35, 36, 62, 63
memmove-s0-sn-t0, 32, 33, 35
memmove-s0-sn-t1, 33–35
memmove-s0p, 7, 13–15, 19–22, 31–36
memmove-s0p-info, 35
memmove-s1-s1, 23
memmove-s1-s1-mem, 23
memmove-s1-s1-rfile, 23
memmove-s1-s2, 22
memmove-s1-s2-else, 22
memmove-s1-s2-mem, 23
memmove-s1-s2-rfile, 22
memmove-s1-sn, 30
memmove-s1-sn-base-1, 27
memmove-s1-sn-base-2, 28
memmove-s1-sn-induct, 29, 30
memmove-s1-sn-mem, 30
memmove-s1-sn-mem-base-1, 28
memmove-s1-sn-mem-base-2, 29
memmove-s1-sn-rfile, 30
memmove-s1-sn-rfile-base-1, 27
memmove-s1-sn-rfile-base-2, 29
memmove-s1-sn-t, 29, 30, 32, 62
memmove-s1-sn-t0, 28, 29
memmove-s1p, 8, 17, 19, 22, 23, 27–31
memmove-s1p-info, 30
memmove-s2-s2, 25
memmove-s2-s2-mem, 25
memmove-s2-s2-rfile, 25
memmove-s2-sn, 26
memmove-s2-sn-base, 24
memmove-s2-sn-induct, 26
memmove-s2-sn-mem, 27
memmove-s2-sn-mem-base, 24
memmove-s2-sn-rfile, 26
memmove-s2-sn-rfile-base, 24
memmove-s2-sn-t, 25–28, 33, 62
memmove-s2p, 8, 18, 20, 22, 24–27
memmove-s2p-info, 26
memmove-s3-s3, 46
memmove-s3-s3-mem, 46
memmove-s3-s3-rfile, 46
memmove-s3-s4, 44
memmove-s3-s4-else, 44
memmove-s3-s4-mem-base, 44
memmove-s3-s4-rfile-base, 44
memmove-s3-s5, 45
memmove-s3-s5-else, 45

memmove-s3-s5-mem, 45
memmove-s3-s5-rfile, 45
memmove-s3-sn, 60
memmove-s3-sn-base-1, 56
memmove-s3-sn-base-2, 57
memmove-s3-sn-base-3, 59
memmove-s3-sn-mem, 61
memmove-s3-sn-mem-base-1, 57
memmove-s3-sn-mem-base-2, 58
memmove-s3-sn-mem-base-3, 59
memmove-s3-sn-rfile, 61
memmove-s3-sn-rfile-base-1, 56
memmove-s3-sn-rfile-base-2, 58
memmove-s3-sn-rfile-base-3, 59
memmove-s3-sn-t, 60, 61, 64
memmove-s3-sn-t0, 57, 58, 60
memmove-s3-sn-t1, 58–60
memmove-s3p, 9, 37, 38, 40, 44–46, 56–61
memmove-s3p-info, 60
memmove-s4-s4, 48
memmove-s4-s4-mem, 48
memmove-s4-s4-rfile, 48
memmove-s4-s5, 47
memmove-s4-s5-else, 47
memmove-s4-s5-mem, 47
memmove-s4-s5-rfile, 47
memmove-s4-sn, 55
memmove-s4-sn-base-1, 52
memmove-s4-sn-base-2, 53
memmove-s4-sn-induct, 54, 55
memmove-s4-sn-mem, 55
memmove-s4-sn-mem-base-1, 53
memmove-s4-sn-mem-base-2, 54
memmove-s4-sn-rfile, 55
memmove-s4-sn-rfile-base-1, 52
memmove-s4-sn-rfile-base-2, 53
memmove-s4-sn-t, 54, 55, 57, 63
memmove-s4-sn-t0, 53, 54
memmove-s4p, 10, 41, 44, 47, 48, 52–55
memmove-s4p-info, 55
memmove-s5-s5, 49
memmove-s5-s5-mem, 50
memmove-s5-s5-rfile, 50
memmove-s5-sn, 51
memmove-s5-sn-base, 48
memmove-s5-sn-induct, 50, 51
memmove-s5-sn-mem, 51
memmove-s5-sn-mem-base, 49
memmove-s5-sn-rfile, 51
memmove-s5-sn-rfile-base, 49
memmove-s5-sn-t, 50, 51, 53, 58, 63
memmove-s5p, 10, 43, 45, 47–52
memmove-s5p-info, 51
memmove-statep, 7, 11–19, 36–43, 64
memmove-t, 61, 64
mmov1-lst, 26, 28, 34
mmov1-lst1, 51, 53, 59
movem-saved, 13, 14, 16–25, 27, 29–31, 33, 34, 36, 37, 39, 40, 42–52, 54–56, 58, 59, 61
movn-lst, 22, 23, 27–30, 47, 48, 52–55

nat-rangep, 7–11
nat-to-uint, 7–19, 36–43, 61

put-nth, 19–21, 24–26, 31–35, 44–46, 49–51, 56–60

ram-addrp, 7–11
read-an, 7–14, 16–18, 20–37, 39, 40, 42–44, 46–61
read-dn, 7–12, 24, 26–28, 30–32, 34, 35, 49, 51–53, 55–57, 59, 61, 65
read-mem, 7, 12–16, 18–25, 27–29, 31–33, 35, 36, 38, 39, 41–44, 46–50, 52–54, 56–58, 60, 61, 65
read-rn, 12–34, 36, 37, 39–59, 61, 64
read-sp, 7, 12–19, 37–43, 64
readm-rn, 13, 14, 16–18, 37, 39, 40, 42, 43
rom-addrp, 7–10
rts-addr, 11–14, 16–18, 37, 39, 40, 42, 43, 64

splus, 26, 28, 29, 32, 33, 35, 50, 53, 54, 57, 58, 60, 62–64
stepn, 11–61, 64
sub, 7–25, 27–29, 31–33, 35–58, 60, 61, 64

uint-rangep, 7
uint-to-nat, 37, 38, 40, 41, 43, 63, 64
uread-dn, 7–11
uread-mem, 7