1. Memory-mapped input and output. This module supplies procedures for reading from and writing to `MMIX` memory addresses that exceed 48 bits. Such addresses are used by the operating system for input and output, so they require special treatment. At present only dummy versions of these routines are implemented. Users who need nontrivial versions of *spec_read* and/or *spec_write* should prepare their own and link them with the rest of the simulator.

Many I/O devices communicate via bytes or wydes or tetras instead of octabytes. So these prototype routines have a *size* parameter, to distinguish between the various kinds of quantities that `MMIX` wants to read from and write to the memory-mapped addresses.

```
#include <stdio.h>
#include "mmix-pipe.h"      /* header file for all modules */
  extern octa read_hex();     /* found in the main program module */
  static char buf[20];
  static char *kind[] = {"byte","wyde","tetra","octa"};
  extern octa shift_left ARGS((octa y, int s));      /* y ≪ s, 0 ≤ s ≤ 64 */
  extern octa shift_right ARGS((octa y, int s, int u));      /* y ≫ s, signed if ¬u */
```

2. If the *interactive_read_bit* of the *verbose* control is set, the user is supposed to supply values dynamically. Otherwise zero is read.

```
  octa spec_read ARGS((octa, int));
  octa spec_read(addr, size)
      octa addr;
      int size;
  {
    octa val;
    size &= #3, addr.l &= -(1 ≪ size);
    if (verbose & interactive_read_bit) {
      printf("** Read %s from loc %08x%08x: ", kind[size], addr.h, addr.l);
      fgets(buf, 20, stdin);
      val = read_hex(buf);
    }
    else val.l = val.h = 0;
    switch (size) {
    case 0: val.l &= #ff;
    case 1: val.l &= #ffff;
    case 2: val.h = 0;
    case 3: break;
    }
    if (verbose & show_spec_bit) {
      printf("   (spec_read ");
      switch (size) {
      case 0: printf("%02x", val.l); break;
      case 1: printf("%04x", val.l); break;
      case 2: printf("%08x", val.l); break;
      case 3: printf("%08x%08x", val.h, val.l); break;
      }
      printf(" from %08x%08x at time %d)\n", addr.h, addr.l, ticks.l);
    }
    return shift_left(val, (8 - (1 ≪ size) - (addr.l & 7)) ≪ 3);
  }
```

3. The default *spec_write* just reports its arguments, without actually writing anything.

```
void spec_write ARGS((octa, octa, int));
void spec_write(addr, val, size)
    octa addr, val;
    int size;
{
  if (verbose & show_spec_bit) {
    size &= #3, addr.l &= -(1 ≪ size);
    val = shift_right(val, (8 - (1 ≪ size) - (addr.l & 7)) ≪ 3, 1);
    printf("␣␣␣(spec_write␣");
    switch (size) {
    case 0: printf("%02x", val.l); break;
    case 1: printf("%04x", val.l); break;
    case 2: printf("%08x", val.l); break;
    case 3: printf("%08x%08x", val.h, val.l); break;
    }
    printf("␣to␣%08x%08x␣at␣time␣%d)\n", addr.h, addr.l, ticks.l);
  }
}
```

4. Incidentally, the combined address a and size s could be transmitted in 64 bits of an actual memory bus, because a is always a multiple of 2^s that is less than 2^{63}. Thus (a, s) can be packed neatly into the 64-bit number $2a + 2^s$. (Think about it.)

5. Index.

November 7, 2022 at 05:31

MMIX-MEM