Action of the PCAL and SCAL Instructions

The PCAL instruction performs the following actions:

- If the priv bit in the E register is not set (meaning that the calling procedure is nonprivileged), check the "first priv" value (word 1 of the code space). If the offset in the instruction is greater or equal, the procedure is trying to call a privileged procedure. Generate a protection trap.
- If the priv bit in the E register is not set, check the "first callable" value (word 0 of the code space). If the offset in the instruction is greater or equal, set the priv bit in the E register.
- Push the current value of the P register (program counter) onto the stack.
- Push the *old* value of the E register onto the stack.
- Push the current L register value onto the stack.
- Copy the S register (stack pointer) to the L register.
- Set the RP field of the E register to 7 (empty).
- Load the contents of the PEP word addressed by the instruction into the P register.

The SCAL instruction works in exactly the same way, except that it also sets the SC bit in the E register, thus ensuring that execution continues in kernel space. The data space does *not* change.

The PCAL and SCAL instructions are very similar, and the programmer normally does not need to distinguish between them. That is done by the system at execution time. Thus library procedures can be moved between user code and system code with no recompilation.

The Interprocessor Bus

All communication between CPUs goes via the *interprocessor bus*, or *IPB*. There are in fact two buses, called X and Y (see Figure 8-1), in case one fails. Unlike other components, both buses are used in parallel when they're up.

Data is passed across the bus in fixed-length packets of 16 words. The bus is fast enough to saturate memory on both CPUs, so the client CPU performs it synchronously in the *dispatcher* (scheduler) using the SEND instruction. The destination (server) CPU reserves buffer space for a single transfer at boot time. On completion of the transfer, the destination CPU receives a *bus receive* interrupt and handles the packet.

Input/Output

Each processor has a single I/O bus with up to 32 controllers. All controllers are dual-ported and connected to two different CPUs. At any one time, only one CPU has access to any specific controller. This relationship between CPU and controller is called *ownership*: the controlling