All calls are indirect via a *Procedure Entry Point Table*, or *PEP*, which occupies up to the first 512 words of each code space. The last 9 bits of the `PCAL` or `SCAL` instruction are an index in this table.

This approach has dangers and advantages: the kernel uses exactly the same function call methods as user code, which simplifies coding conventions and allows code to be moved between kernel and user space. On the other hand, at least in theory, the `SCAL` instruction enables any user program to call any kernel function.

The system protects access to sensitive procedures based on the `priv` bit in the E register. It distinguishes between three kinds of procedures:

- Nonprivileged procedures, which can be called from any procedure, regardless of whether they are privileged.
- Privileged procedures, which can be called only from other privileged procedures.
- *Callable* procedures, which can be called from any procedure, but which set the `priv` bit once called. They provide the link between the privileged and the nonprivileged procedures.

The distinction between privileged, nonprivileged, and callable procedures is dependent on their position in the PEP. Thus it is possible to have nonprivileged library procedures in the system PEP, sometimes called the *SEP*. The table has the structure shown in Figure 8-6.

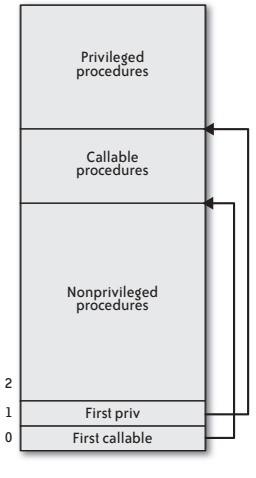

FIGURE 8-6. Procedure entry point table