These address modes are encoded in the first few bits of the address field of the instruction; see Figure 8-5.

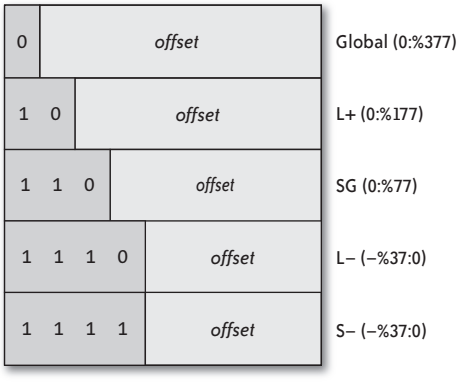

FIGURE 8-5. Address format

The % symbol represents octal numbers, like %377 (decimal 255, or hexadecimal 7F). Tandem does not use hexadecimal.

The instruction format also provides single-level indirection: if the I bit is set in the instruction, the retrieved data word is taken as the address of the final operand, which has to be in the same address space. The address space and the data word are both 16 bits wide, so there is no possibility for multilevel indirection.

One problem with this implementation is that the unit of data is a 16-bit word, not a byte. The instruction set also provides "byte instructions" with a different addressing method: the low-order bit of the address specifies the byte in the word, and the remainder of the address are the low-order 15 bits of the word address. For data accesses, this limits byte addressability to the first 32 kB of the data space; for code access, it limits the access to the same half of the address space as the current instruction. This has given rise to the restriction that a procedure cannot span the 32 kB boundary in the code space.

There are also two instructions, LWP (*load word from program*) and LBP (*load byte from program*), that can access data in the current code space.

Procedure Calls

Tandem's programming model owes much to the Algol and Pascal world, so it reserves the word *function* for functions that return a value, and uses the word *procedure* for those that do not. Two instructions are provided to call a procedure: PCAL for procedures in the current code space, and SCAL for procedures in the system code space. SCAL fulfills the function of a *system call* in other architectures.