In keeping with the aim of reliability and data integrity, the `trap` bit in the E register enables, among other things, traps on arithmetic overflow. There are "logical" equivalents of the arithmetic instructions that do not set the condition codes.

The CPU has a hardware stack addressed by two registers, the *S register* or stack pointer, and the *L register*, which points to the current stack frame. The L register is a relatively new idea: it points to the base of the current frame. Unlike the S register, it does not change during the execution of a procedure.* The stack is limited by addressing considerations to the first 32 kB of the current data space, and unlike some other machines, it grows upward.†

In addition to the hardware stack, there is a *register stack* of eight 16-bit words. The registers are numbered R0 to R7, but the instruction set uses them as a circular stack, where the top of stack is defined by the `RP` bits of the E register. In the following example, RP is set to 3, making R3 the top of stack, referred to as the `A` register; see Figure 8-3.

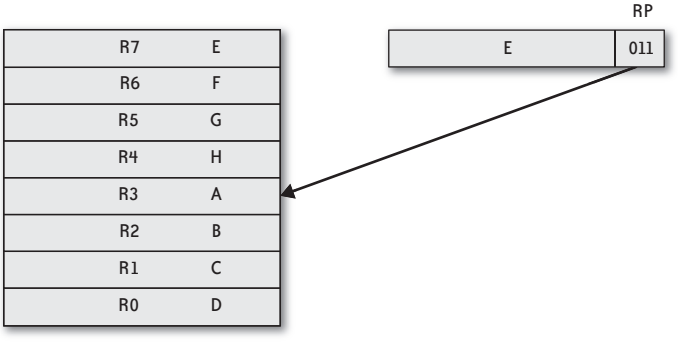

FIGURE 8-3. Register stack

Assuming that the register stack is "empty" at the beginning, a typical instruction sequence might be:

```
LOAD    var^a           -- push var^a on stack (R0)
LOAD    var^b           -- push var^b on stack (R1)
ADD                     -- add A and B (R1 and R0), storing result in R0 (A)
STOR    var^c           -- save A to var^c
```

Instructions are all 16 bits wide, which does not leave much space for an address field: it is only 9 bits wide. To work around this problem, Tandem bases addressing on offsets from a series of registers; see Figure 8-4.

* This is the same thing as the base pointer register used in most 21st-century processors.

† Stacks were quite a new idea in the 1970s. Like its predecessor, the HP 3000, Tandem's support for stacks went significantly beyond that of systems such as DEC's PDP-11, the most significant other stack-based machine of the time.