

Subroutines and Control Abstraction

8.2.3 Register Windows

EXAMPLE 8.67

Register windows on the SPARC

As an alternative to saving and restoring registers on subroutine calls and returns, the original Berkeley RISC machines [PD80, Pat85] incorporated a hardware mechanism known as *register windows*. The basic idea is to provide a very large set of physical registers, most of which are organized as a collection of overlapping windows (Figure ©8.12). A few register names (r0–r7 in the figure) always refer to the same locations, but the rest (r8–r31 in the figure) are interpreted relative to the currently active window. On a subroutine call, the hardware moves to a different window. To facilitate the passing of parameters, the old and new windows overlap: the top few registers in the caller’s window (r24–r31 in the figure) are the same as the bottom few registers in the callee’s window (r8–r15 in the figure). On a machine with register windows, the compiler places values of use only within the current subroutine in the middle part of the window. It copies values to the upper part of the window to pass them to a called routine, within which they are read from the lower part of the window. ■

Since the number of physical windows is fixed, a long chain of subroutine calls can cause the hardware to run off the end of the register set, resulting in a “window overflow” interrupt that drops the processor into the operating system. The interrupt handler then treats the set of available windows as a circular buffer. It copies the contents of one or more windows to memory and then resumes execution. Later, a “window underflow” interrupt will occur when control attempts to return into a window whose contents have been written to memory. Again the operating system recovers, by restoring the saved registers and resuming execution. In practice, eight windows appear to suffice to make overflow and underflow relatively rare on typical programs.

Register windows have been used in several RISC processors, but only one of these, the SPARC, is commercially significant today. The more recent Intel IA-64 (Itanium) also uses register windows, though it is not a RISC machine. The advantage of windows, of course, is that they reduce the number of loads and stores required for the typical subroutine call. At the same time, register

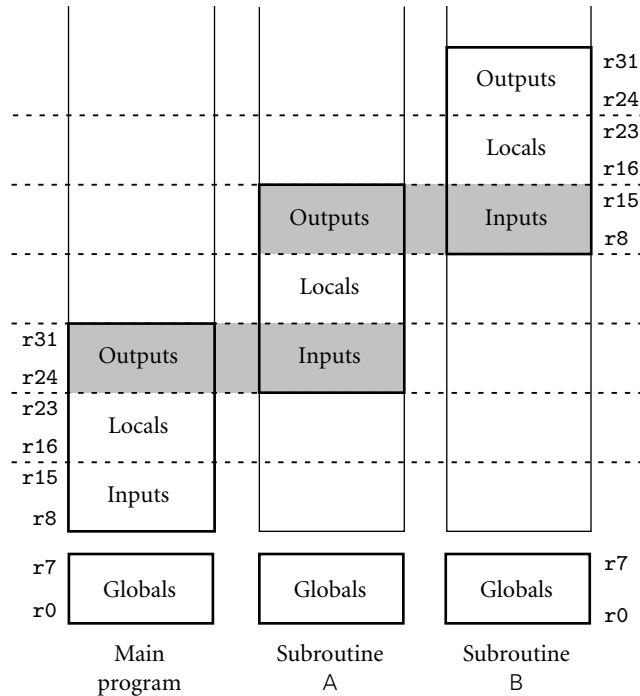


Figure 8.12 Register windows. When the main program calls subroutine A, and again when A calls B, register names `r0–r7` continue to refer to the same locations, but register names `r8–r31` are changed to refer to a new, overlapping window. High-numbered registers in the caller share locations with low-numbered registers in the callee.

windows significantly increase the amount of state associated with the currently running program. When the operating system decides to give the processor to a different application for a while (something that most systems do many times per second), it must save all this state to memory, or arrange for the processor to trap back into the OS if the new process attempts to access an unsaved window. Worse, while register windows nicely capture the referencing environment of a single thread of control, they do not work well for languages that need more than one referencing environment (execution context). Several language features, including continuations (Section 6.2.2), iterators (Section 6.5.3), and coroutines (Section 8.6), are difficult to implement on a machine with register windows, because they require that we save and restore not only the visible registers, but those in other windows as well, when switching between contexts. It is unclear whether the reduction in subroutine call overhead outweighs the extra cost of context switches for typical application workloads, particularly given that loads and stores for parameters are almost always cache hits.

✓ CHECK YOUR UNDERSTANDING

59. What are *register windows*? What purpose do they serve?
 60. Which commercial instruction sets include register windows?
 61. Explain the concepts of register window *overflow* and *underflow*.
 62. Why are register windows a potential problem for multithreaded programs?
-

