the throwing of a page fault or protection violation is an exceptional but normal occurrence that should be mapped to an instance of `Exception`.

TIP #5: EXCEPTIONS ARE EXCEPTIONAL

Exceptions should be used for exceptional conditions, not just for errors. Using exceptions for flow control in unusual circumstances provides a hint to the VM to optimize for the nonexception path, giving you optimal performance.

In handling page faults and exceptions, we have been slightly cavalier in throwing out two of the usual conventions on the use of exceptions:

- `ProcessorException`, which is the class that represents all Intel processor exception types, extends `RuntimeException`.
- Page faults, and to a lesser extent protection violations, are exceptional conditions, but they are common enough that in order to refrain from sacrificing performance, we throw them using static exception instances.

Both of these decisions were made for good reasons, but were born in large part out of the peculiarities of the JPC project.

Choosing to extend `RuntimeException` was mainly a code beautification decision. Traditionally, runtime exceptions are reserved for exceptions that cannot or should not be handled. Most texts recommend that if thrown, they are allowed to propagate up and cause the termination of the throwing thread. We know that the distinction between runtime and checked exceptions is source "candy" in the same vein as generics, autoboxing, or the for-each loop. Not that we mean to denigrate these constructs, but by observing that the classification of an exception has no effect on the compiled code, we can safely choose the most convenient classification without risking any performance penalty. In our case, `ProcessorException` is a runtime exception in order to avoid having to declare `throws ProcessorException` on a multitude of methods.

Throwing statically initialized exception instances (effectively singleton exceptions) for page faults and protection violations yields benefits at both ends of the exception-handling chain. First, we save ourselves the cost of instantiating a new object every time we want to throw. This is avoids the cost of reconstructing the stack trace of the calling thread (far from trivial in a modern JVM). Second, we save ourselves the cost of determining the nature of the thrown type, as with a limited set of static exceptions determining the thrown type, we only need to perform a sequence of reference comparisons.