instruction's behavior. The other compiler phases at the MIR level are concerned with ensuring that calling, exception, and other conventions are adhered to.

Factored control flow graph

Java programs create runtime exceptions if a null pointer is used to access memory or if an array index is out-of-bounds. These runtime exceptions control the flow of code and so should end basic blocks. This results in small basic blocks with less scope for local optimizations (described earlier in "HIR"). To increase the size of basic blocks, the control-flow dependence of runtime exceptions is turned into an artificial data dependence at the HIR and LIR levels. This dependence ensures that operations are ordered correctly for exception semantics, and therefore the basic block may be left during mid-execution to handle a runtime exception. When the intermediate form has exceptional exit points in the middle of blocks, the control flow graph is said to be factored (Choi et al. 1999).

Instructions are created that explicitly test the runtime exceptions and generate synthetic guard results. Instructions that then require ordering make use of the guard. An instruction that can generate an exception is known as a *Potentially Exceptioning Instruction* (PEI). All PEIs generate a synthetic guard value, as do instructions that can be used to remove PEIs if they are redundant. For example, a branch that tests for null makes null pointer tests on the same value redundant. The guard from the branch is used in place of the guard from the null pointer tests to ensure that instructions cannot be reordered to before the branch.

Figure 10-7 shows a single array assignment having its constituent runtime exceptions turned into PEIs and the guard dependencies that ensure the code is executed in the correct order.

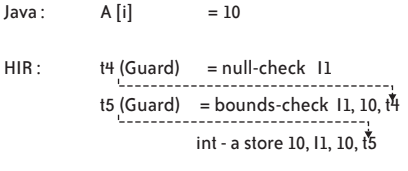

FIGURE 10-7. An example of instructions in a factored control flow graph

Scalar and extended array SSA forms

Static Single Assignment (SSA) form reduces the dependencies that compiler optimizations need to be concerned about. The form ensures that any registers (more commonly known as variables; in the HIR and LIR phases of Jikes RVM, all variables will be held in registers) are written to at most once. A compiler transformation must handle three kinds of dependence:

True dependence
: Where a register is written and then read.