Exception Handling Implementation

Daniel Haris (xharis00)
xharis00@stud.fit.vutbr.cz

Filip Bajaník (xbajan01)
xbajan01@stud.fit.vutbr.cz

Exception handling is the process of responding to the occurrence of exceptions – anomalous or exceptional conditions requiring special processing – often changing the normal flow of program execution. It is provided by specialized programming language constructs, computer hardware mechanisms like interrupts or operating system IPC facilities like signals. In general, an exception breaks the normal flow of execution and executes a pre-registered exception handler. The details of how this is done depend on whether it is a hardware or software exception and how the software exception is implemented.

The implementation of exception handling in programming languages typically involves a fair amount of support from both a code generator and the runtime system accompanying a compiler.

First approach – *dynamic handler registration*, generates code that continually updates structures about the program state in terms of exception handling. Typically, this adds a new element to the stack frame layout that knows what handlers are available for the function or method associated with that frame. If an exception is thrown, a pointer in the layout directs the runtime to the appropriate handler code.

The second approach is a *table-driven approach* (the one implemented in many production-quality C++ compilers). This creates static tables at compile time and link time that relate ranges of the program counter to the program state with respect to exception handling. Then, if an exception is thrown, the runtime system looks up the current instruction location in the tables and determines what handlers are in play and what needs to be done.