Implementing an Exception Mechanism in C

CS 520
Dept. of Computer Science
Univ. of New Hampshire
int catch Exception (void);
void cancel Catch Exception (void);
void throw Exception (int);

So exception is just an int value
multiple calls to catchException can
be active
try 
  1
  2
  3  
catch (e) 
  1
  2
}

\[ \text{int e;}
\]
\[ \text{if ( e = catchException () ) } \]
\[ \begin{align*}
\text{  1} \\
\text{  2} \\
\text{  3}
\end{align*} \\
\text{else } \]
\[ \begin{align*}
\text{  1} \\
\text{cancel (catchException ());} \\
\text{  3}
\end{align*} \\
\]

\underline{\text{note}} \\
\text{code block 2 could re-throw the exception if it is one it cannot “handle.”}
Key to implementation
understand how Intel 64 stack Frames work

so need to pop stack back
provide new return from catch Exception
catch Exception

- take "snapshot" of stack
  - current RBP & caller's RBP
  - saved R1P

- also save other callee-saved registers
  - rbx & r12-r15

- store the contents of these registers into a struct

- since there can be multiple catch Exceptions alive at one time, need to maintain a stack of their structs
current RBP

squad RBP
squad RIP

catch Exception

low address

high address

long 4

f
throw Exception

restore runtime stack using snapshot
in struct or top of stack of structs
pop the stack of structs
restore registers
perform second return from catch Exception
need assembly language routines

get RBP
get RBX
get R12
get R13
get R14
get R15

} get register contents

→ maybe also routines to
set RBX, set R12, etc?

plus routine to:

parse the stack
restore registers
setup and perform second return
from catch Exception

→ put exception number into RAX
cancel Catch Exception

pop the stack of structs