Implementing an Exception Mechanism in C

cs520
Dept. of Computer Science
Univ. of New Hampshire

int catchException(void);

void cancelCatchException(void);

void throwException(int);

So exception is just an int value

multiple calls to catchException can be active

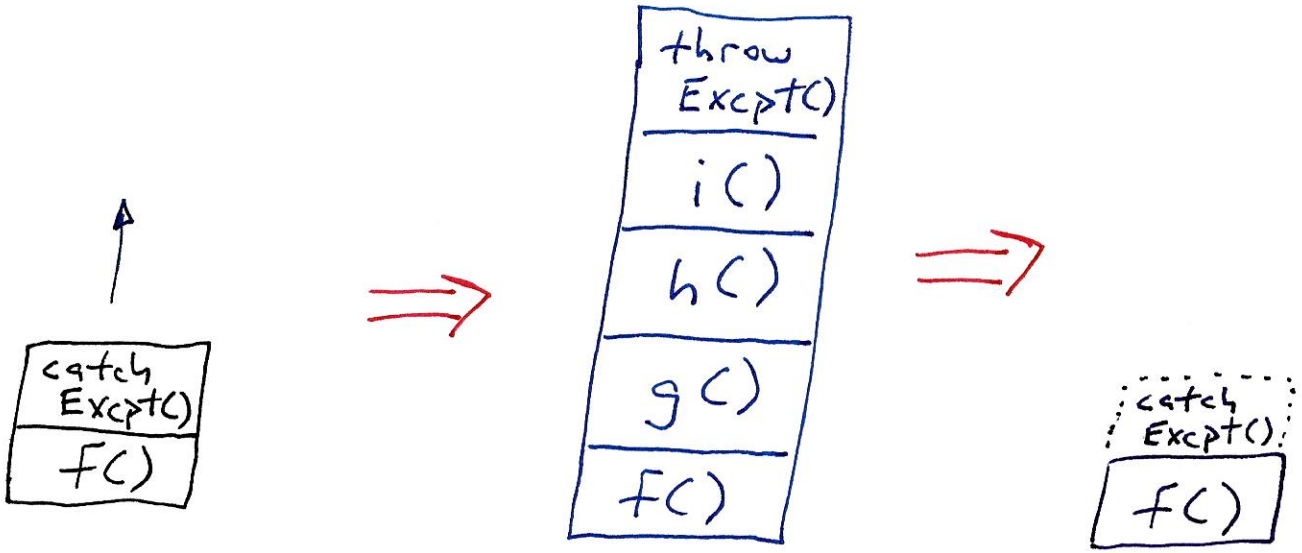
```
try {
  ①
}
catch (e) {
  ②
}
```

```
int e;
if ( e = catchException () ) {
  ②
}
else {
  ①
  cancelCatchException();
}
```

note 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 pare stack back

provide new return from catch Exception

catchException

take "snapshot" of stack

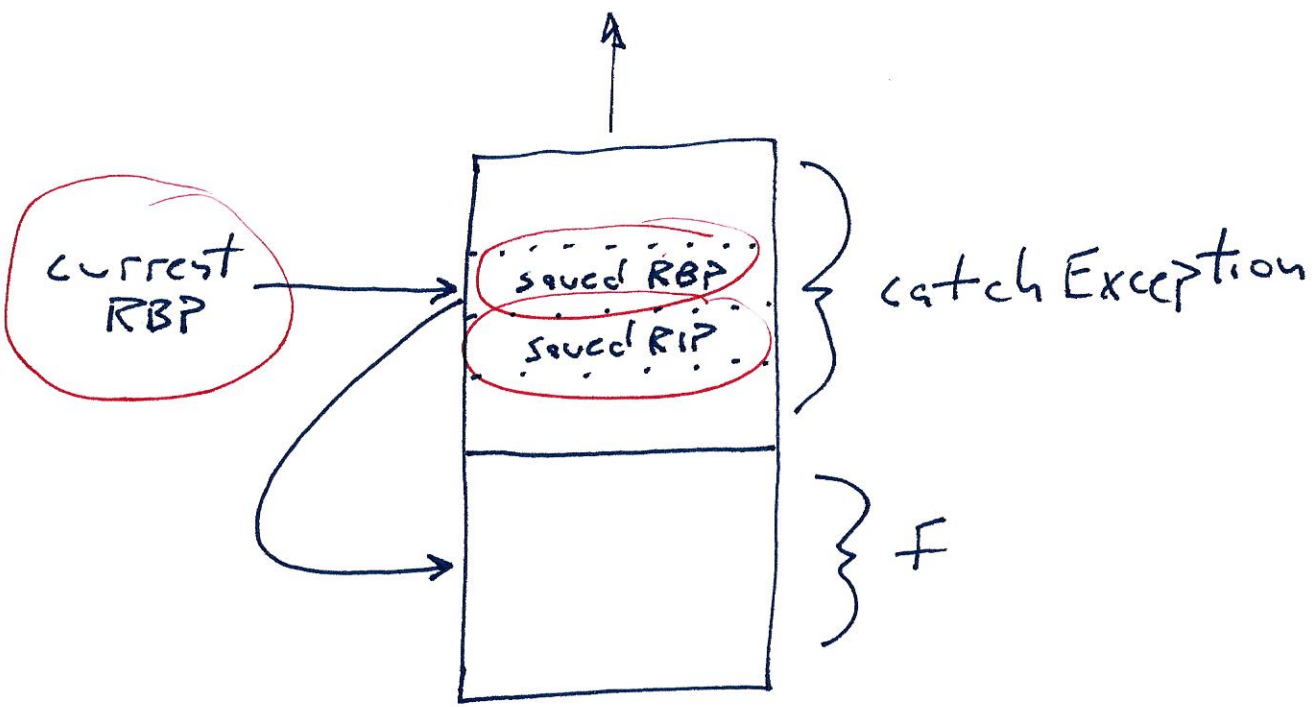
- current RBP & caller's RBP
- saved RIP

also save other callee-saved registers

- rbx & R12-R15

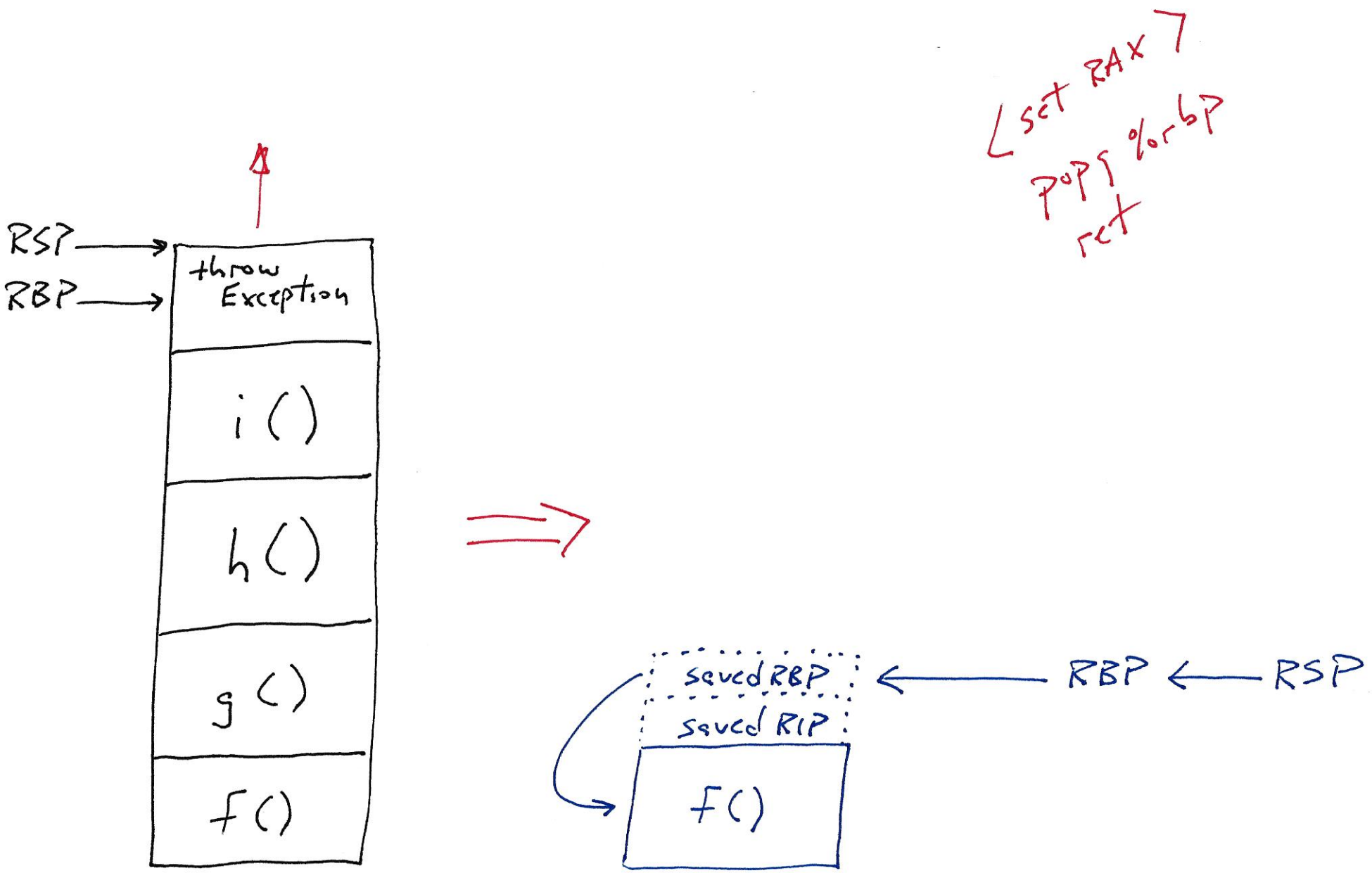
store the contents of these registers into a struct

since there can be multiple catchExceptions alive at one time, need to maintain a stack of these structs


low address
lows *
current RBP
saved RBP
saved RIP
catch Exception
f
high address

throwException

restore runtime stack using snapshot
    in struct on top of stack of structs
pop the stack of structs
restore registers
perform second return from catchException


set RAX
pop %rbp
ret
RSP
RBP
throw Exception
i()
h()
g()
f()
saved RBP
saved RIP
f()
RBP
RSP

need assembly language routines

getRBP
getRBX
getR12
getR13
getR14
getR15

} get register contents

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

plus routine to:

pare the stack
restore registers
setup and perform second return from catchException

↳ put exception number into RAX

cancel Catch Exception

pop the stack of structs