Safe to the Last Instruction: Automated Verification of a Type-Safe Operating System

Jean Yang
MIT CSAIL

Chris Hawblitzel
Microsoft Research

Safe to the Last Instruction: Auto n of a Type ystem

Jean litzel

MIT CSAIL Microsoft Research

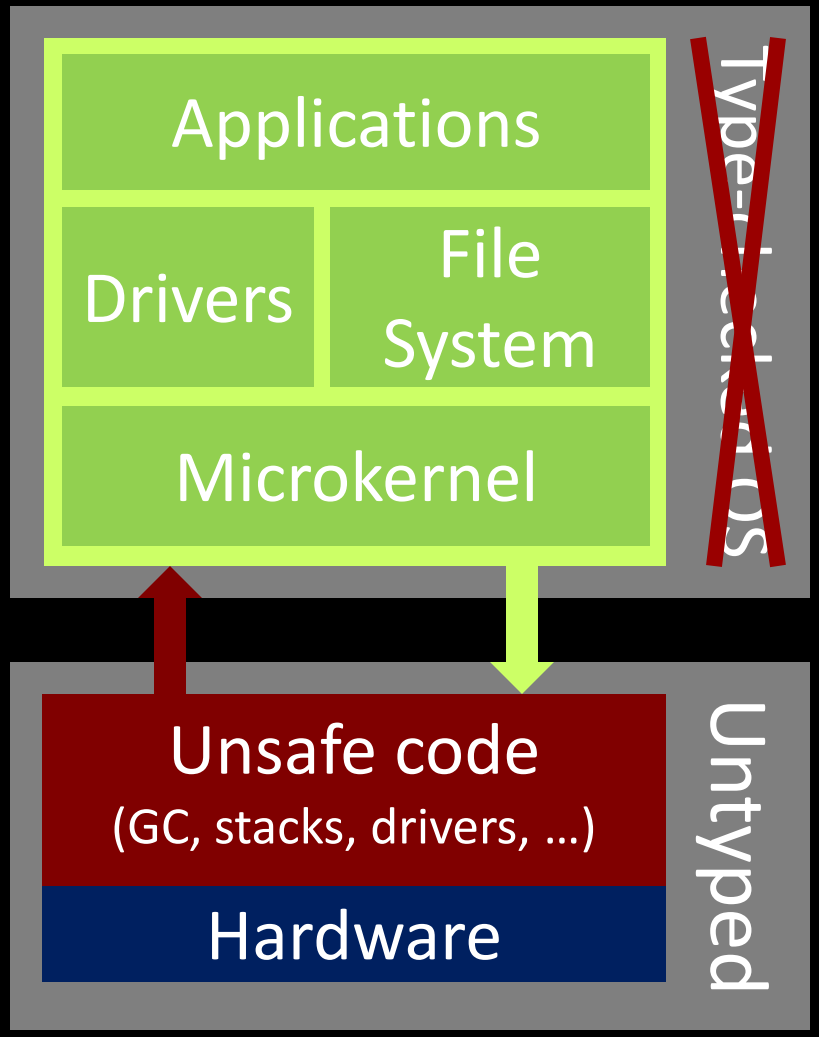

Windows
A fatal exception 0E has occurred at 0028:C0011E36 in VXD VMM(01) +
00010E36. The current application will be terminated.
* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You will
lose any unsaved information in all applications.
Press any key to continue _

Memory Safety

Type Safety

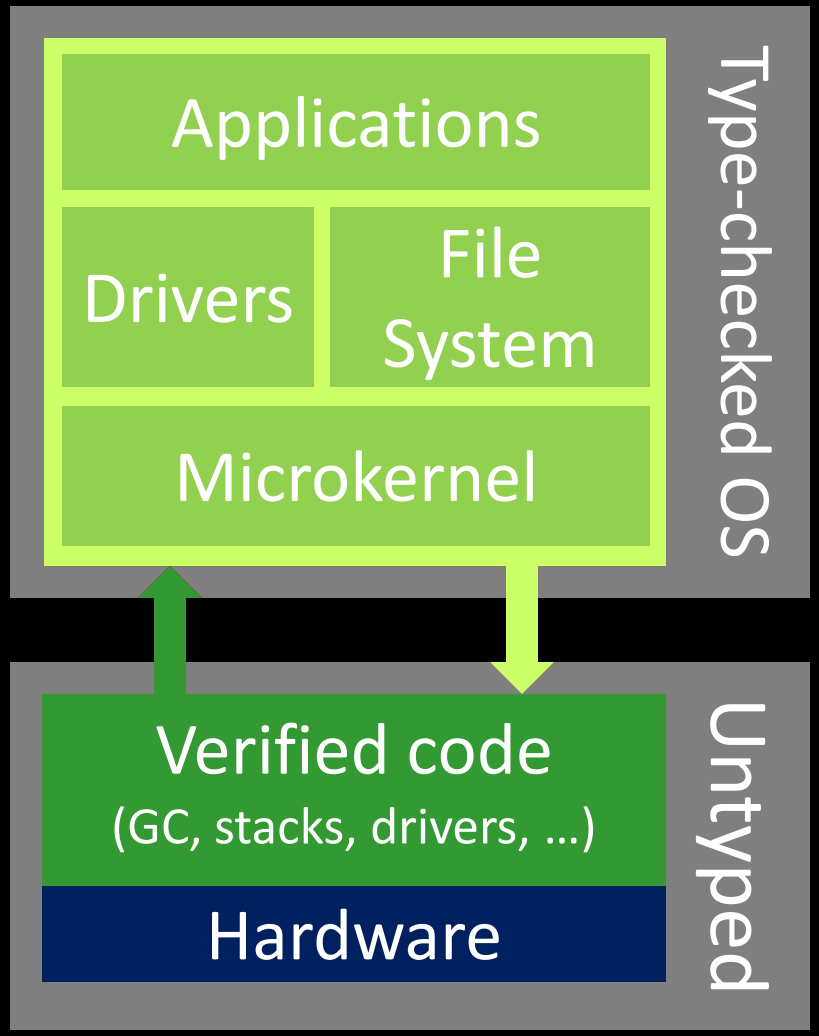
Previously: “Safe” Systems

What currently exists

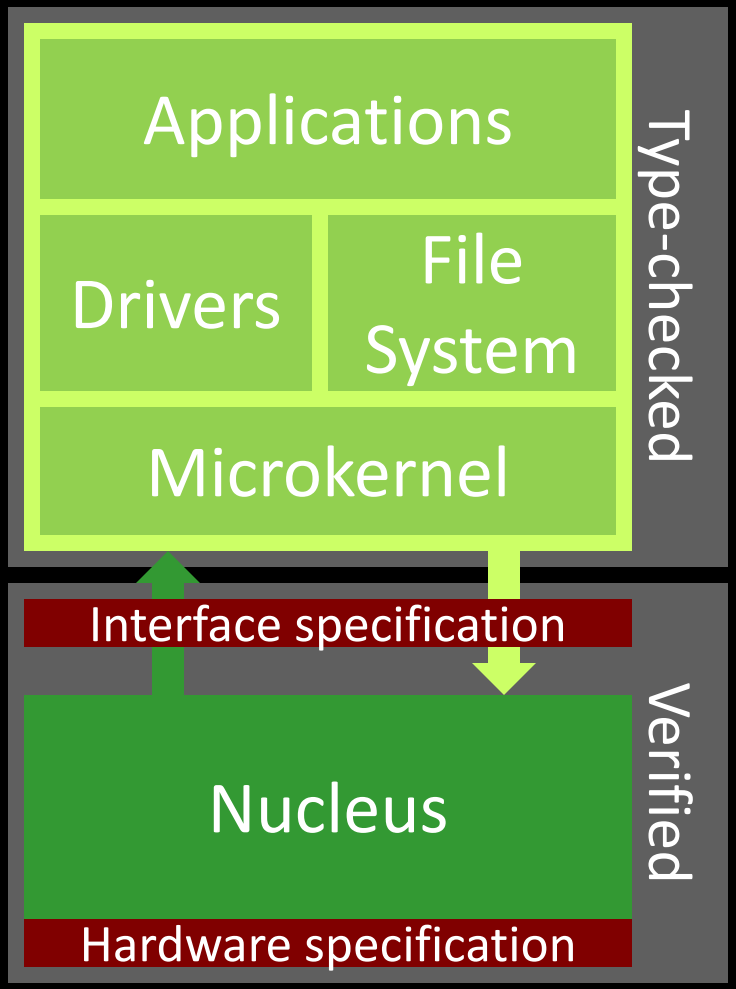


End-to-End Safe Systems

What we want

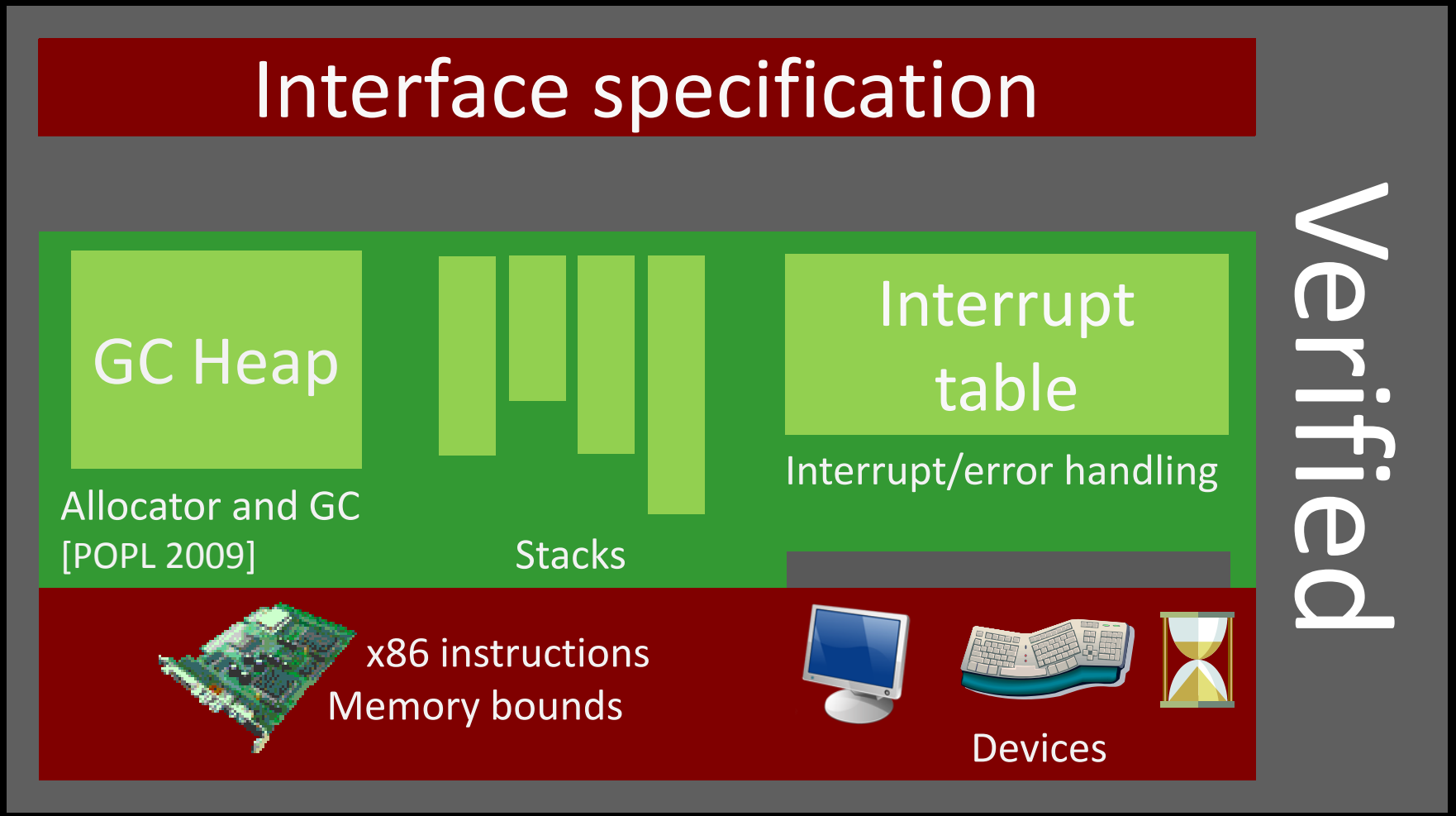


Verve, a Type-Safe OS



- Verify **partial correctness** of low-level **Nucleus** using Hoare logic based on a **hardware spec**.
- Verify an **interface to typed assembly** for end-to-end safety.

The Verve Nucleus



Thread Context Invariant

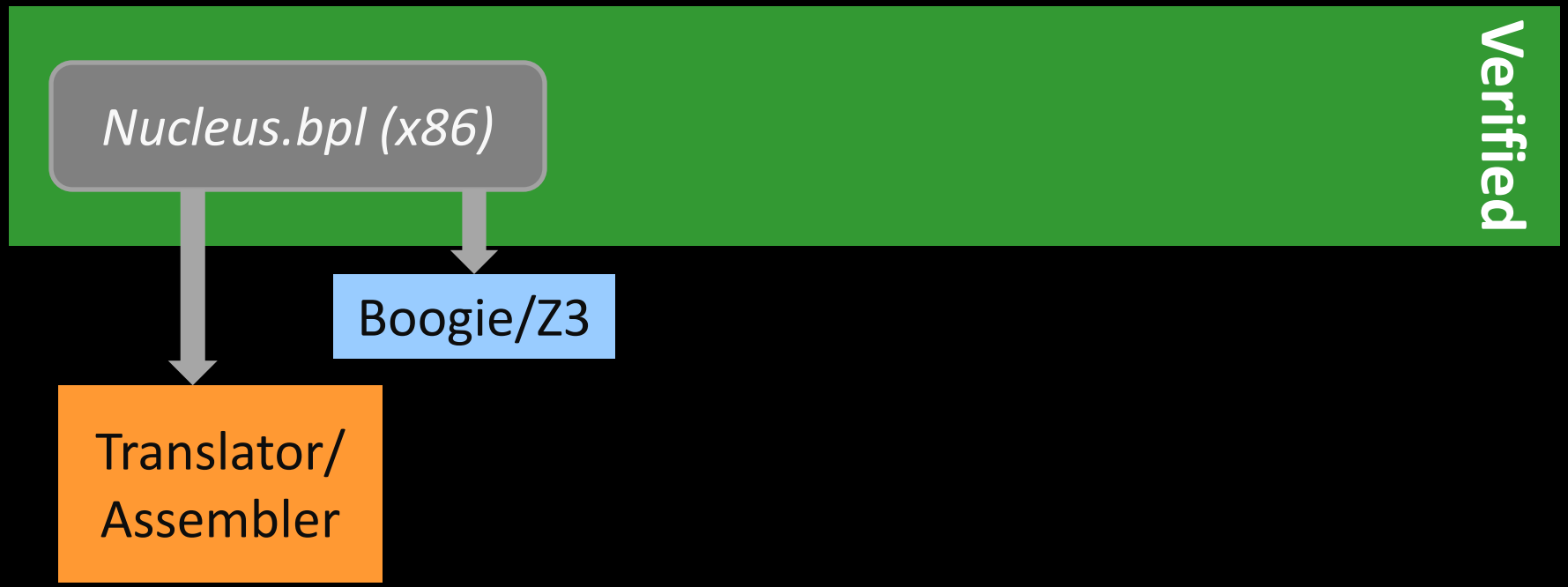
```
function StateInv
   (s:StackID, state:StackState, …)
    returns(bool) {
      (!IsEmpty(state) → …
  && (IsInterrupted(state) → …
  && (IsYielded(state) → …
       && state == StackYielded(
              StackEbp(s, tMems)
            , StackEsp(s, tMems) + 4
            , StackRA(s, tMems, fMems)) && …
}
```

“Load” Specification

```
procedure Load(ptr:int)
  returns (val:int);
  requires memAddr(ptr);
  requires Aligned(ptr);
  modifies Eip;
  ensures word(val);
  ensures val == Mem[ptr];
```

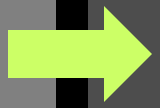
Assembling Verve

- Source file
- Verification tool
- Compilation tool



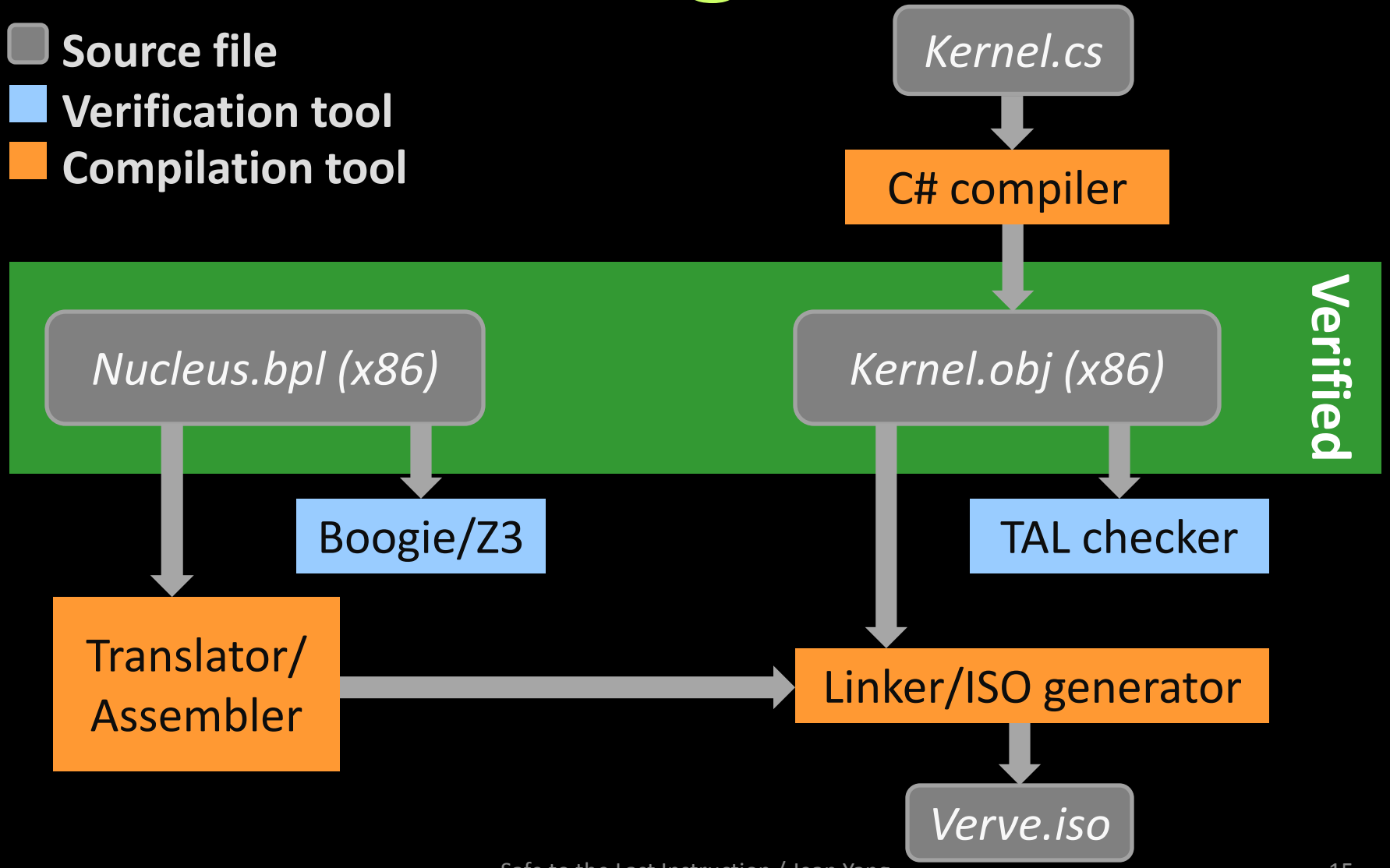
Boogie to x86

```
implementation ReadKeyboard(){
  call KeyboardStatusIn8();
  call eax := And(eax, 1);
  if (eax != 0) { goto proc; }
  call eax := mov(256);
  return;
  proc:
  call KeyboardDataIn8();
  call eax := And(eax, 255);
  return;
}
```



```
ReadKeyboard proc
  in al, 064h
  and eax, 1
  cmp eax, 0
  jne ReadKeyboard$proc
  mov eax, 256
  ret
ReadKeyboard$skip:
  in al, 060h
  and eax, 255
  ret
```

Building Verve



Verve Performance

Verve functionality	Cycles
Round-trip yield	98
Round-trip wait + signal	216

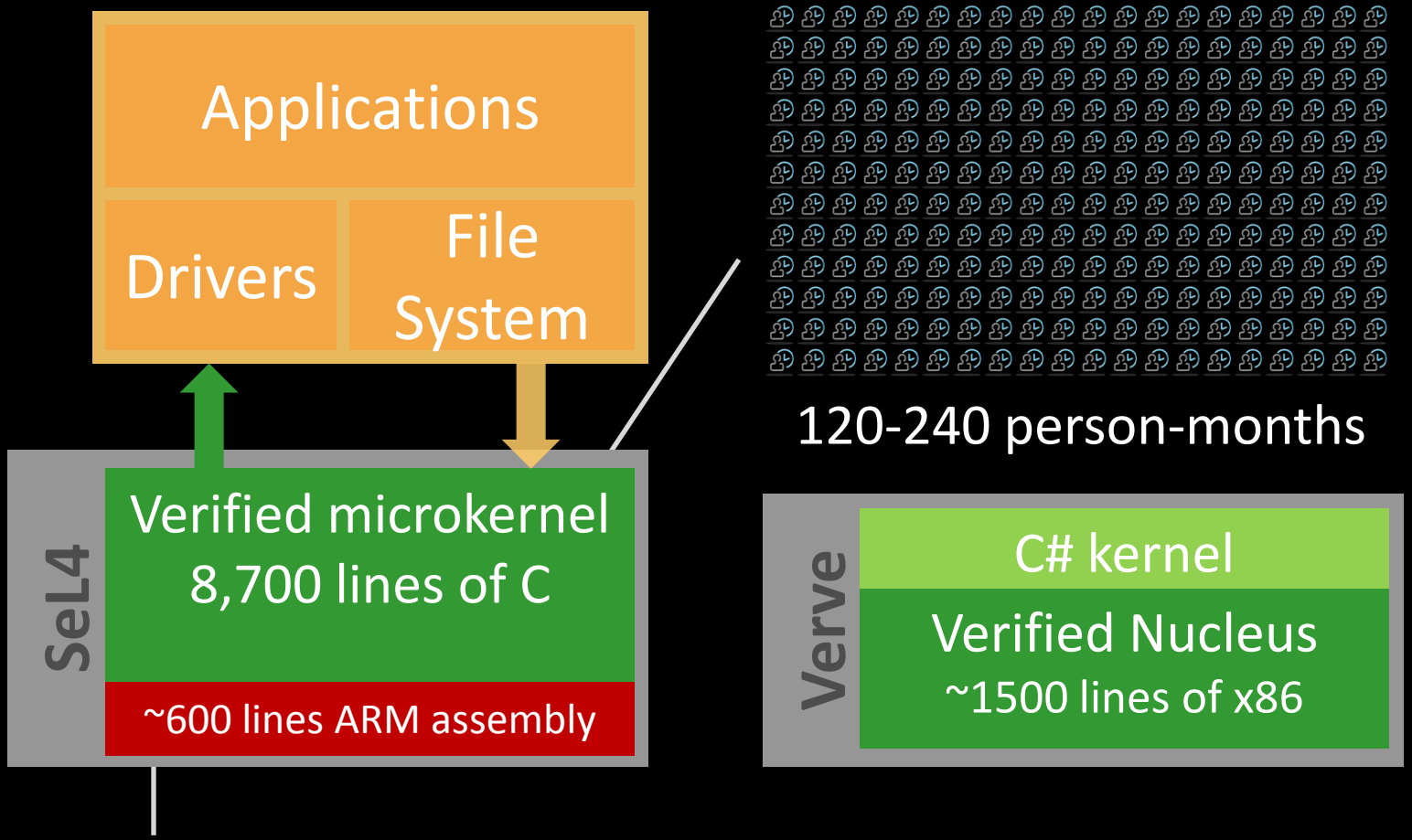
Comparisons	Cycles
L4 (IPC)	224
SeL4 (IPC)	448
Singularity (yield)	2156
Linux (yield)	2390
Windows (yield)	3554

Low Annotation Burden

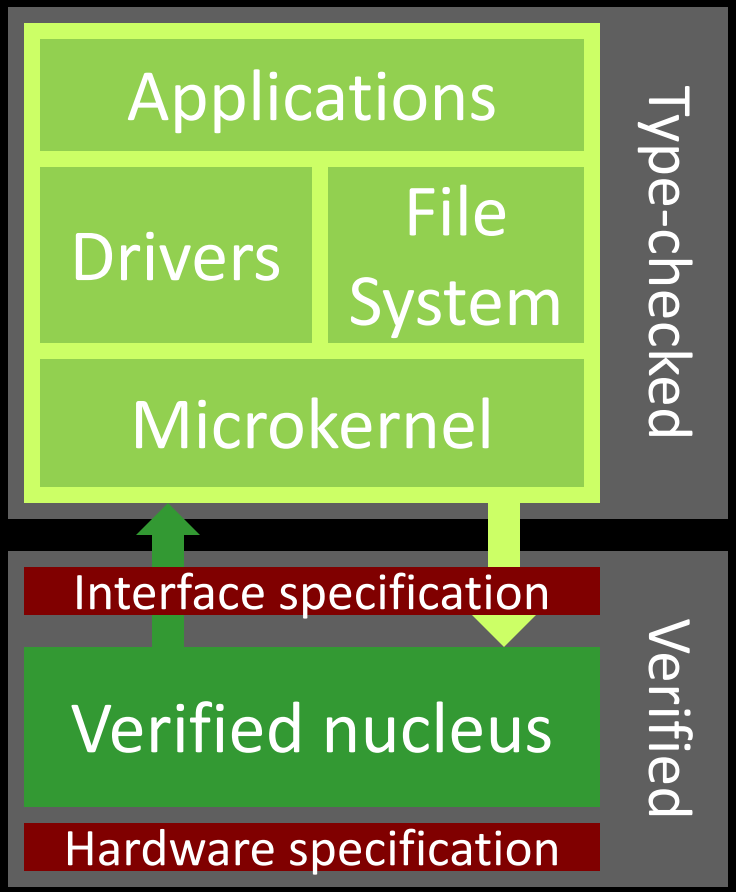
	Copying	Mark-sweep
Specification Boogie lines	1185	
Verified Boogie lines **3x code**	4309	4854
x86 instructions	1377	1489

9 person-months

Verve vs. SeL4?



Contributions



- First **automatically, mechanically verified** OS for **type safety**.
- **Real system** running on x86 with **efficient code**.
- Approach for using **automated techniques** to verify safety.

http://www.codeplex.com/**singularity**