Systeme 3

Kapitel 10a • Meltdown+Spectre:

Laden ohne zu laden

Winter 2019/20

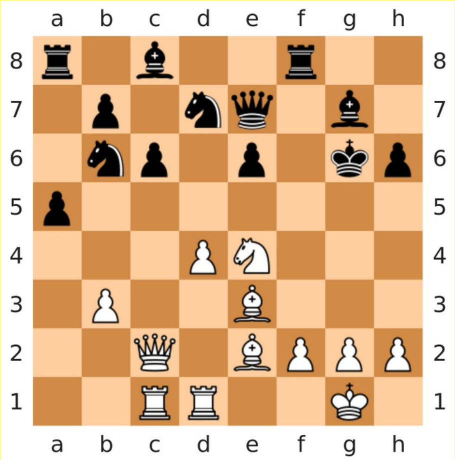
Marcel Waldvogel

Chapter Goals

- How to go beyond physical memory
- How to simplify memory management for the kernel
- How can this be represented with the existing page table structure?
- How to select which page to replace on memory pressure?

Meltdown & Spectre: Overview

CPU bugs → OS/app mitigation

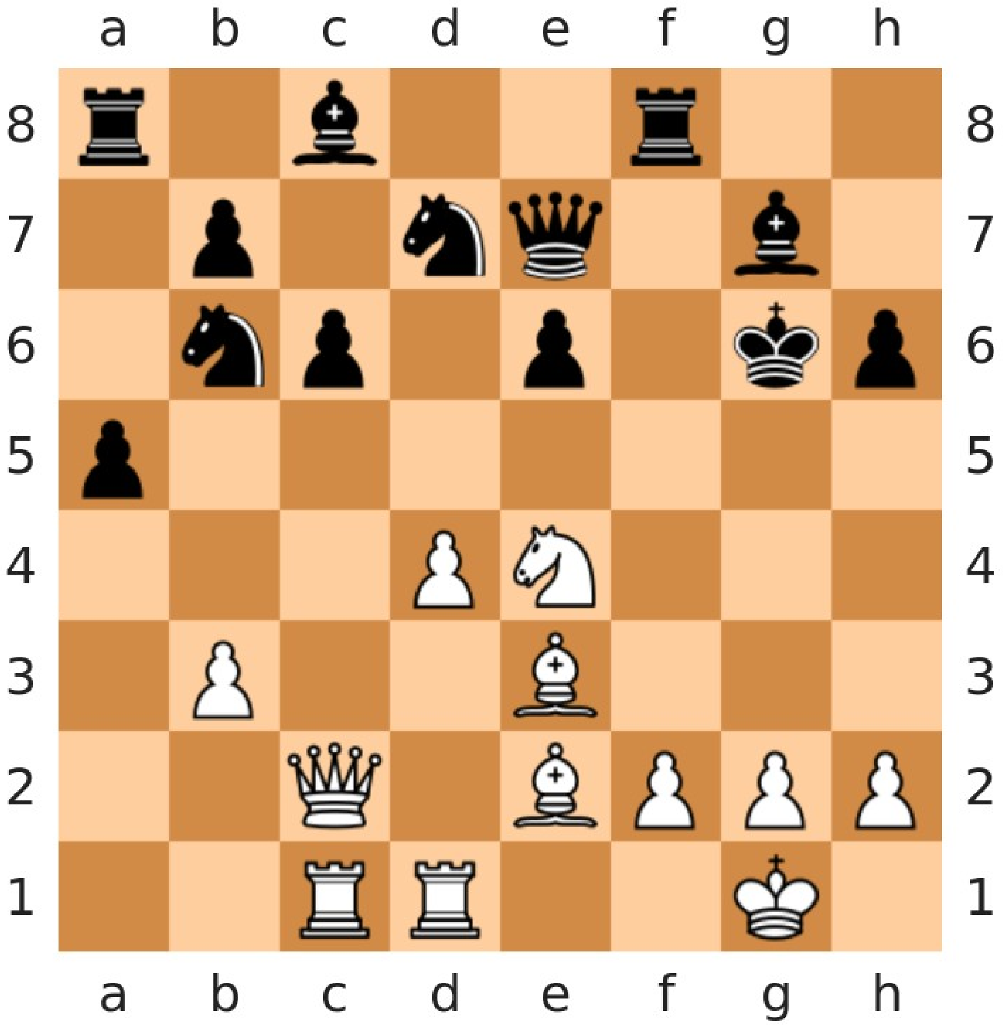


Speculative Execution + Resource Sharing

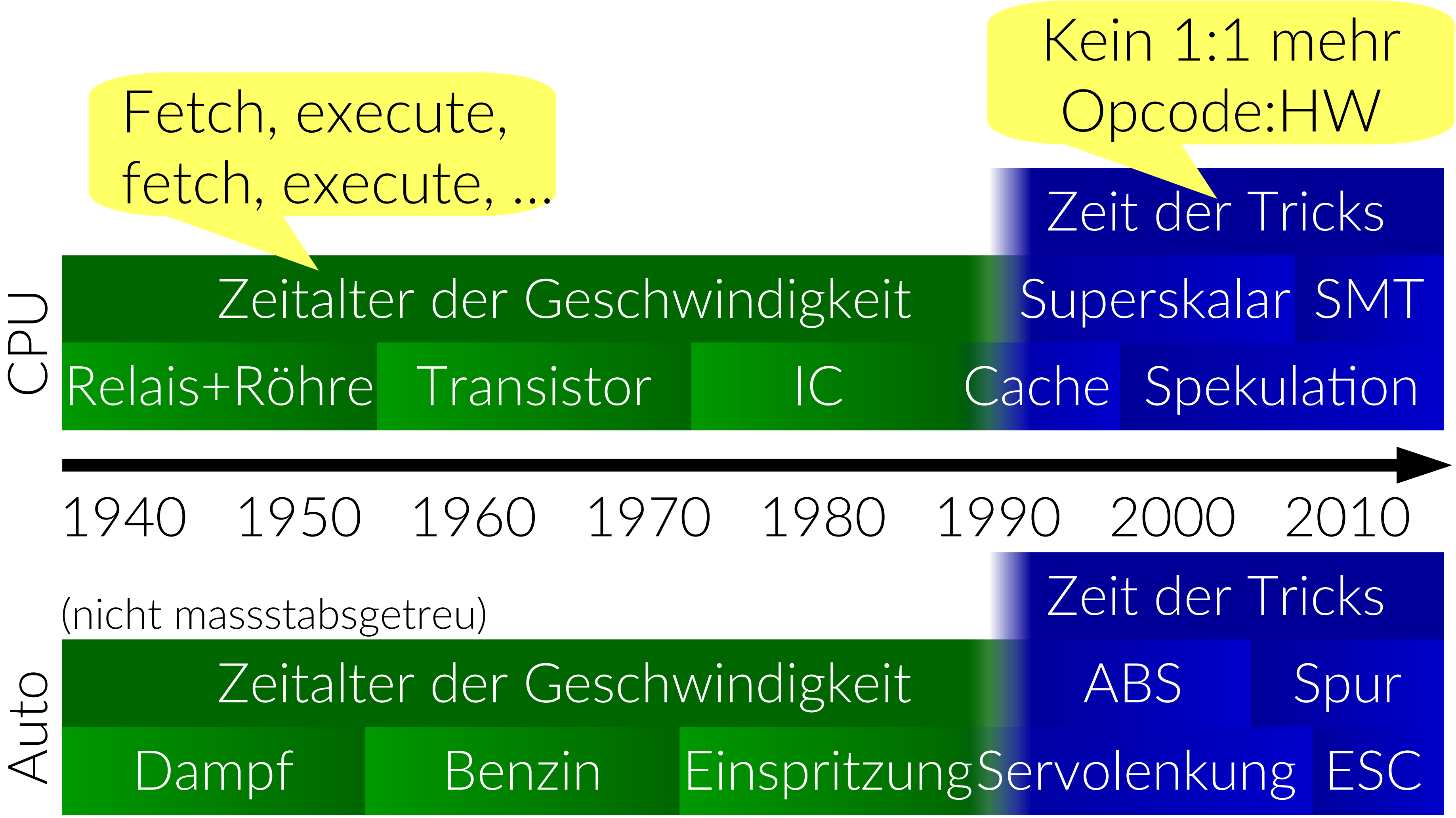
How • Why • Fix: CPU—OS—Apps

Image sources:
Spectre and Meltdown logos: https://meltdownattack.org
Pentium: CPU collection Konstantin Lanzet, CC-BY-SA 3.0 via https://commons.wikimedia.org/wiki/File:KL_Intel_Pentium_4_Northwood.jpg
Chess board: Screenshot from https://en.wikipedia.org/wiki/Chess

Spekulatius



CPU Evolution



Fetch, execute,
fetch, execute, ...

Kein 1:1 mehr
Opcode:HW

CPU

Zeit der Tricks

Zeitalter der Geschwindigkeit

Superskalar

SMT

Relais+Röhre

Transistor

IC

Cache

Spekulation

1940 1950 1960 1970 1980 1990 2000 2010

(nicht massstabsgetreu)

Auto

Zeit der Tricks

Zeitalter der Geschwindigkeit

ABS

Spur

Dampf

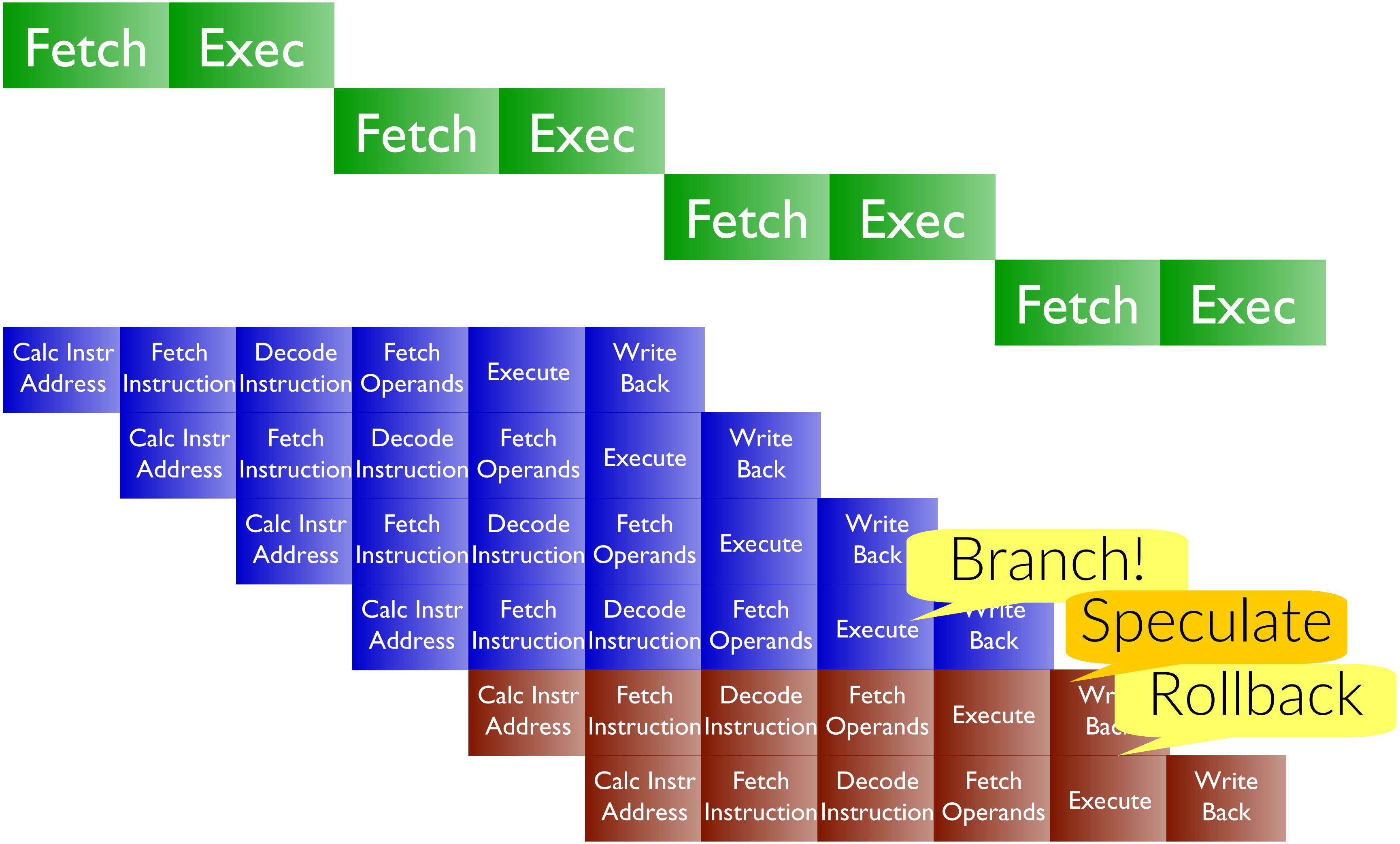
Benzin

Einspritzung

Servolenkung

ESC

Spekulatius II



Fetch Exec

Fetch Exec

Fetch Exec

Fetch Exec

Calc Instr Address | Fetch Instruction | Decode Instruction | Fetch Operands | Execute | Write Back

Calc Instr Address | Fetch Instruction | Decode Instruction | Fetch Operands | Execute | Write Back

Calc Instr Address | Fetch Instruction | Decode Instruction | Fetch Operands | Execute | Write Back

Calc Instr Address | Fetch Instruction | Decode Instruction | Fetch Operands | Execute | Write Back

Calc Instr Address | Fetch Instruction | Decode Instruction | Fetch Operands | Execute | Wr Bac

Calc Instr Address | Fetch Instruction | Decode Instruction | Fetch Operands | Execute | Write Back

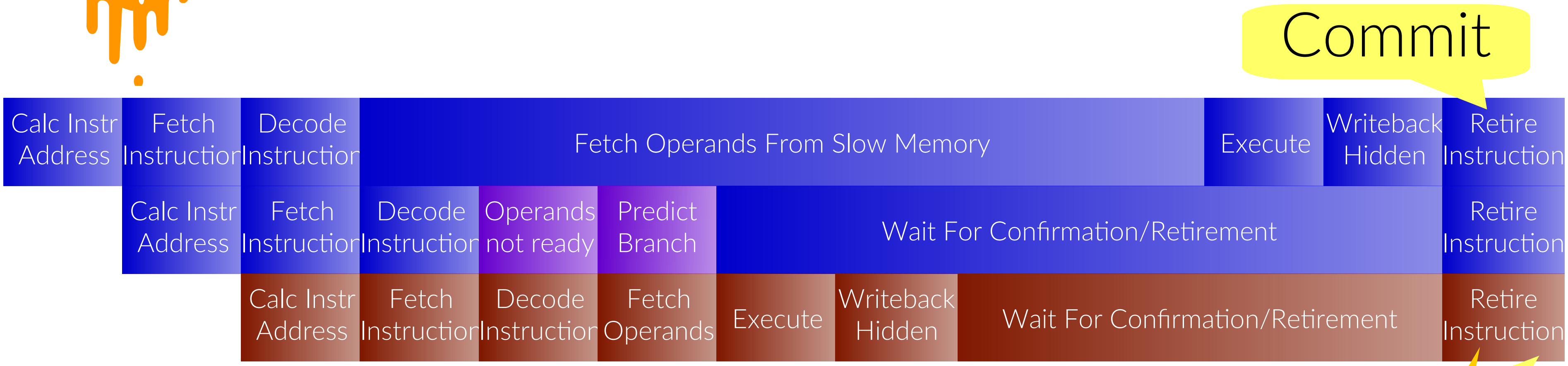
Branch!

Speculate

Rollback

Meltdown

- Intel hit hardest
- Permission check too late



Commit

Calc Instr Address | Fetch Instruction | Decode Instruction | Fetch Operands From Slow Memory | Execute | Writeback Hidden | Retire Instruction

Calc Instr Address | Fetch Instruction | Decode Instruction | Operands not ready | Predict Branch | Wait For Confirmation/Retirement | Retire Instruction

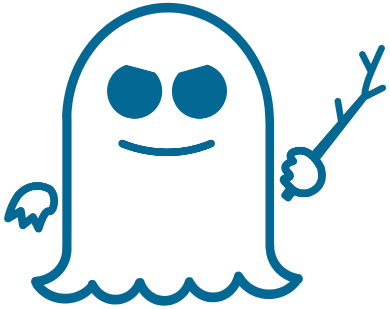
Calc Instr Address | Fetch Instruction | Decode Instruction | Fetch Operands | Execute | Writeback Hidden | Wait For Confirmation/Retirement | Retire Instruction

Commit or rollback

Permission check!

- Read from forbidden address
- Read from some allowed address
- Check which „some“ was read

Spectre



- Most modern processors
- Read from own address space
- Not interesting, is it?
- Find out what would have been read in non-executed (wrongly speculated) code
- Interpreters/JITs everywhere

Spectre I:
Array Bounds Checks

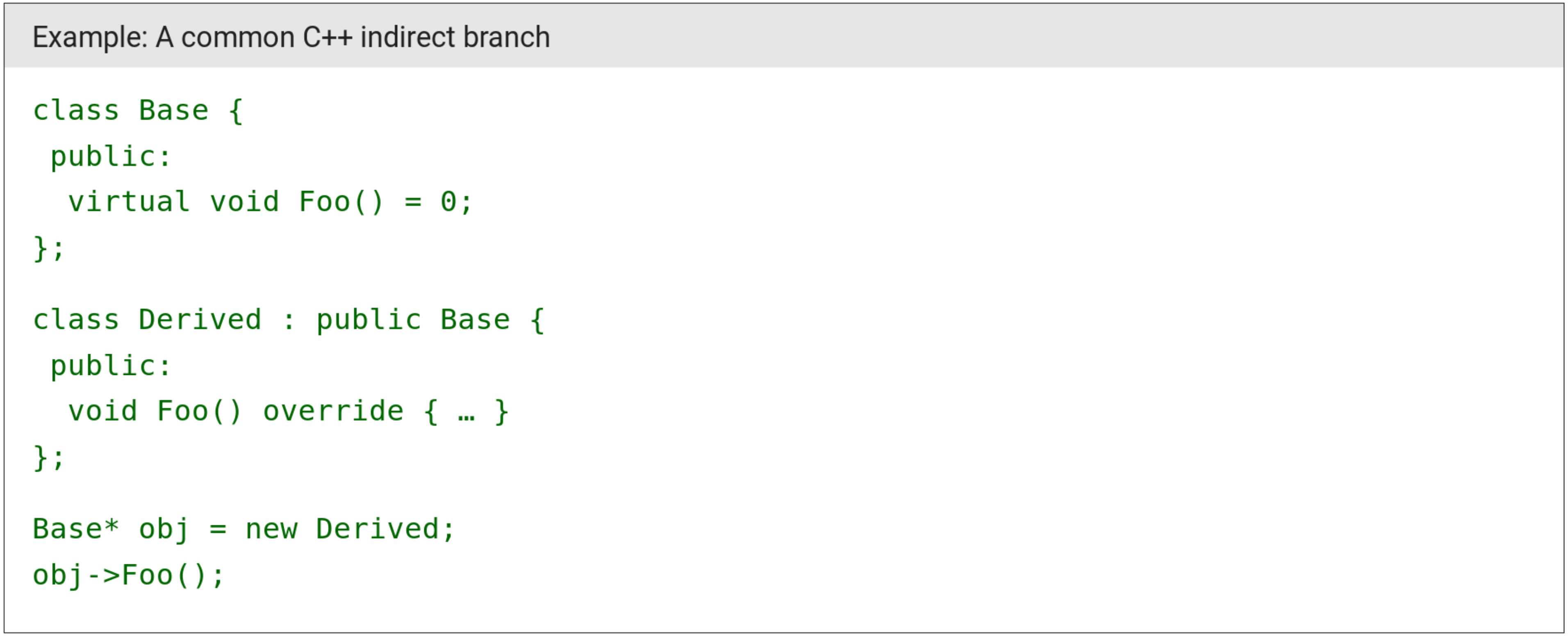
```
struct array {
 unsigned long length;
 unsigned char data[];
};
struct array *arr1 = ...;
unsigned long untrusted_offset_from_caller = ...;
if (untrusted_offset_from_caller < arr1->length) {
 unsigned char value = arr1->data[untrusted_offset_from_caller];
 ...
}
```

Abhilfe:

- Händisch LFENCE
- Zukünftige Compiler?

Examples taken from: https://googleprojectzero.blogspot.de/2018/01/reading-privileged-memory-with-side.html

Spectre II:
Indirect Jumps



Example: A common C++ indirect branch

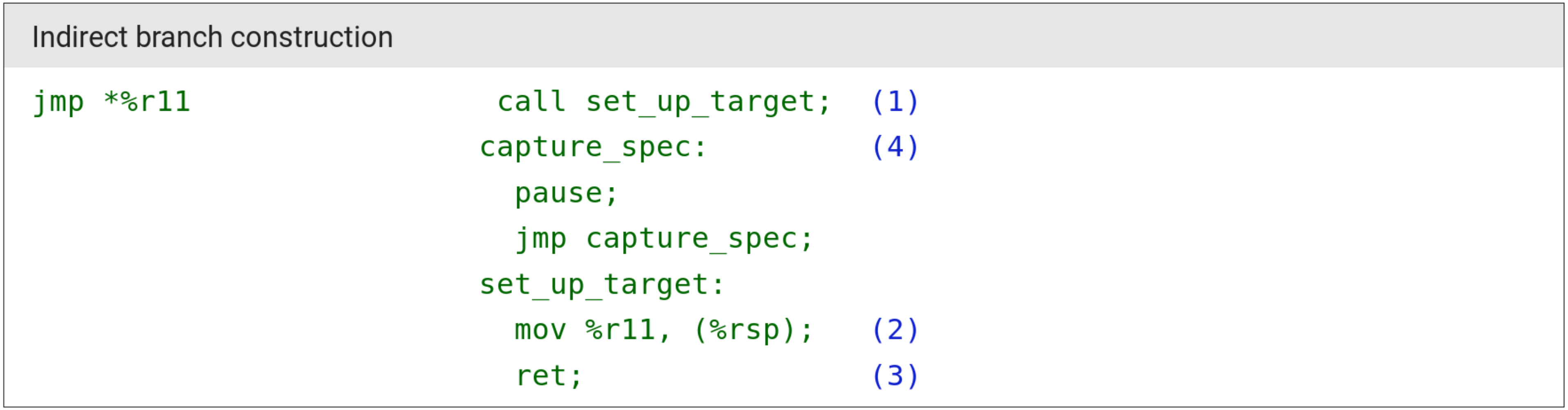
```
class Base {
 public:
  virtual void Foo() = 0;
};

class Derived : public Base {
 public:
  void Foo() override { … }
};

Base* obj = new Derived;
obj->Foo();
```

Examples taken from: https://support.google.com/faqs/answer/7625886

Spectre II: Indirect Jumps

Abhilfe:

- RetPoline (Compiler)



Indirect branch construction

```
jmp *%r11                  call set_up_target;   (1)
                          capture_spec:          (4)
                            pause;
                            jmp capture_spec;
                          set_up_target:
                            mov %r11, (%rsp);    (2)
                            ret;                 (3)
```

Examples taken from: https://support.google.com/faqs/answer/7625886