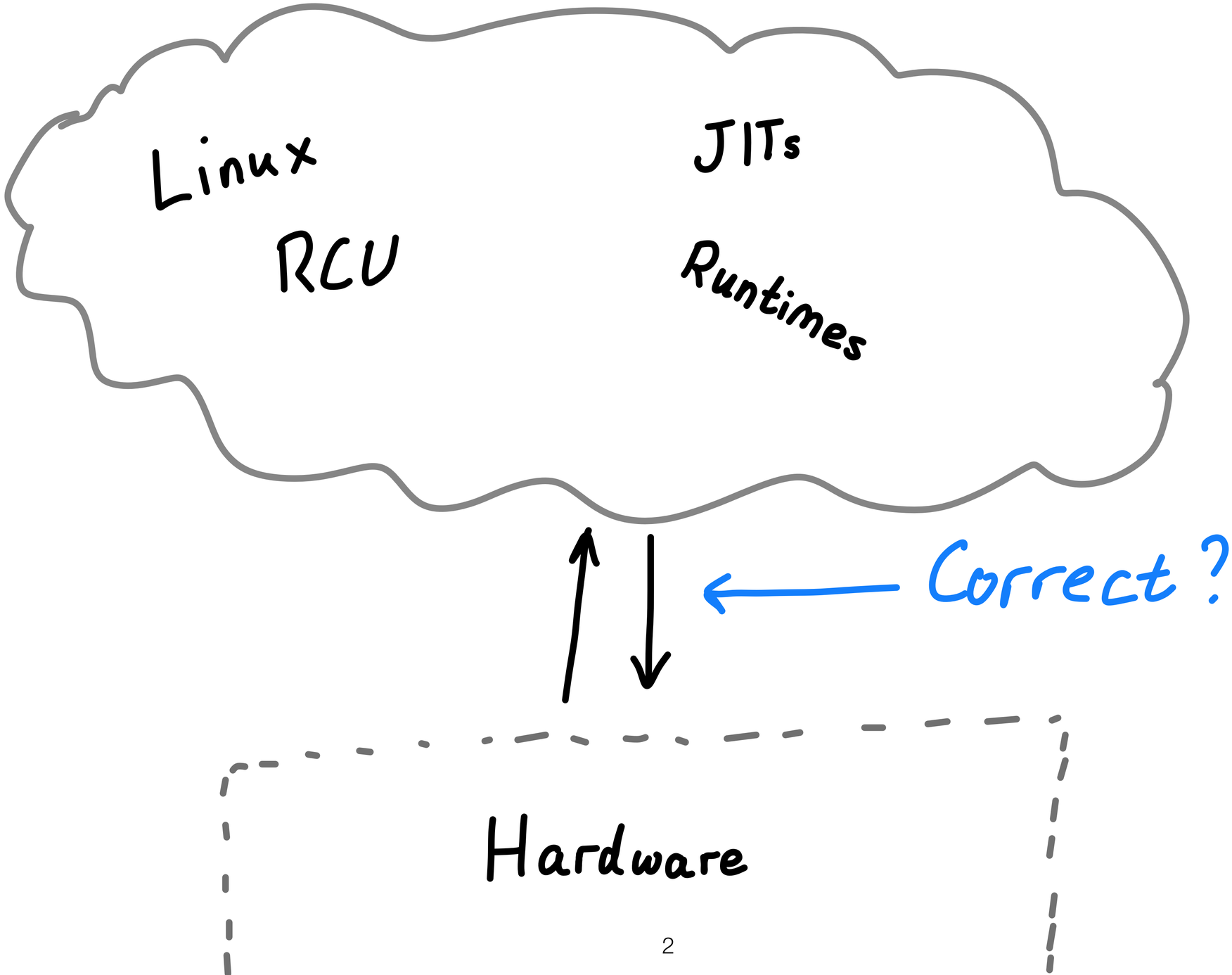
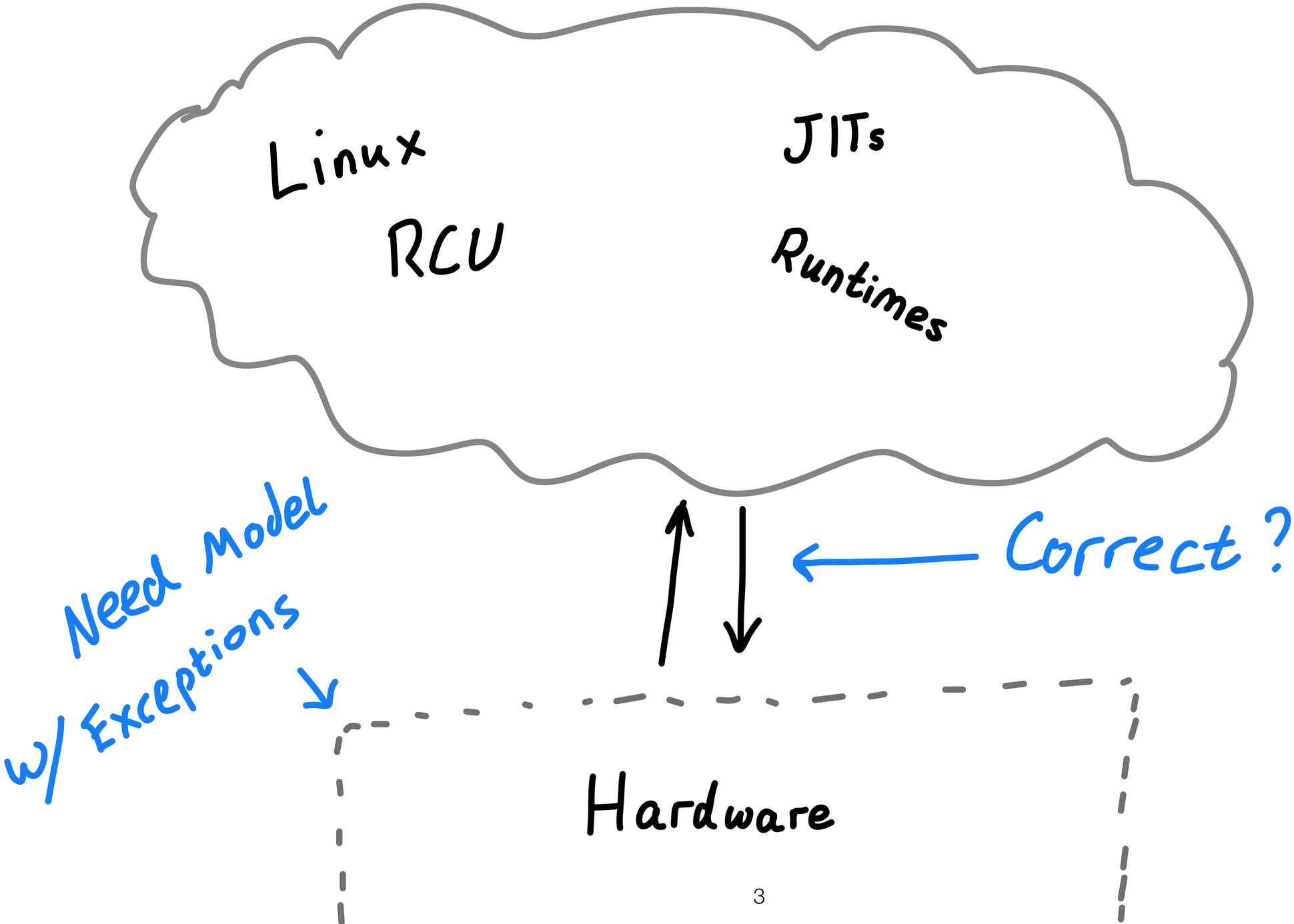
The behaviour of

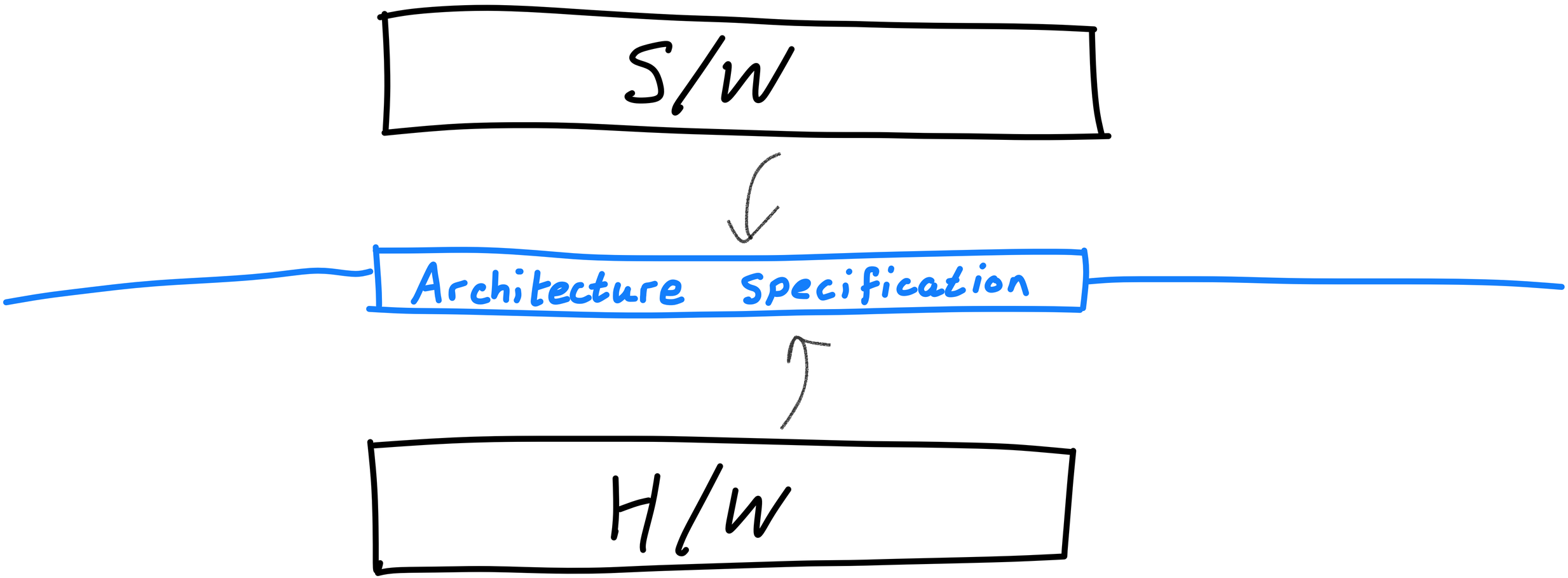
Precise Exceptions in Relaxed Architectures

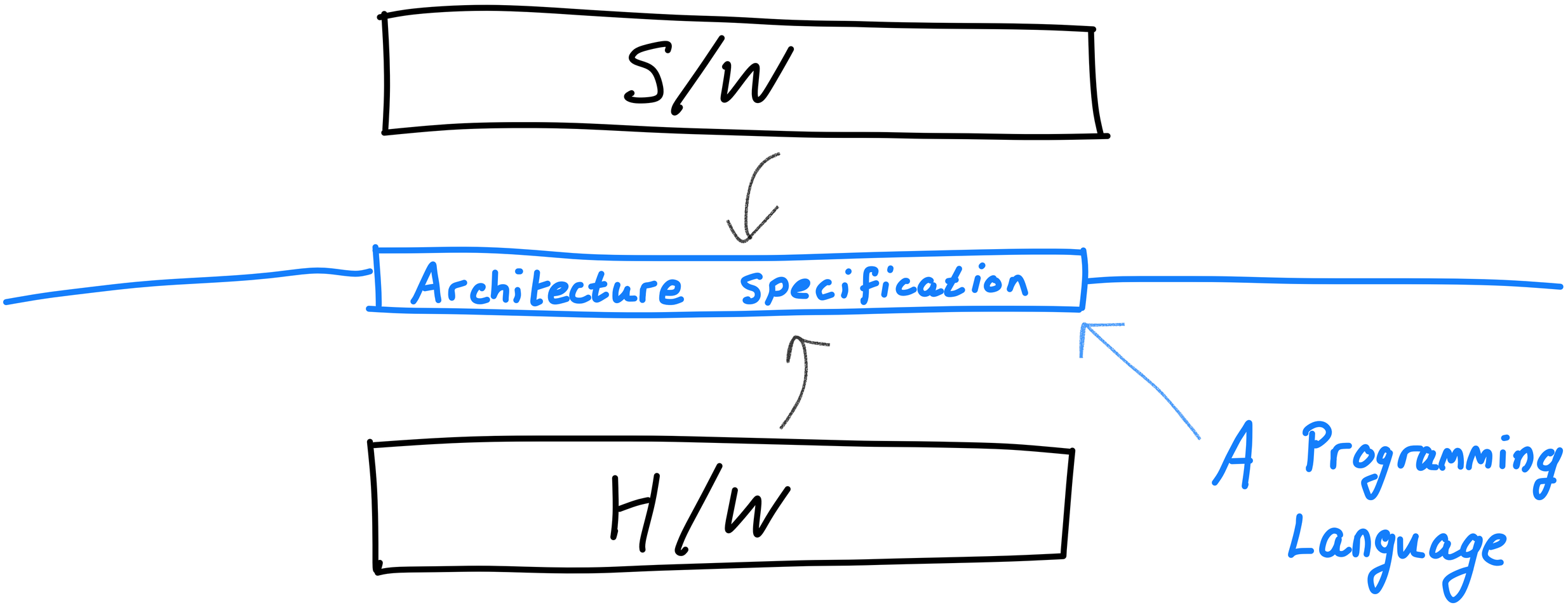
Ben Simner [1]
Alasdair Armstrong [1]
Thomas Bauereiss [1]
Brian Campbell [2]
Ohad Kammar [2]
Jean Pichon-Pharabod [3]
Peter Sewell [1]

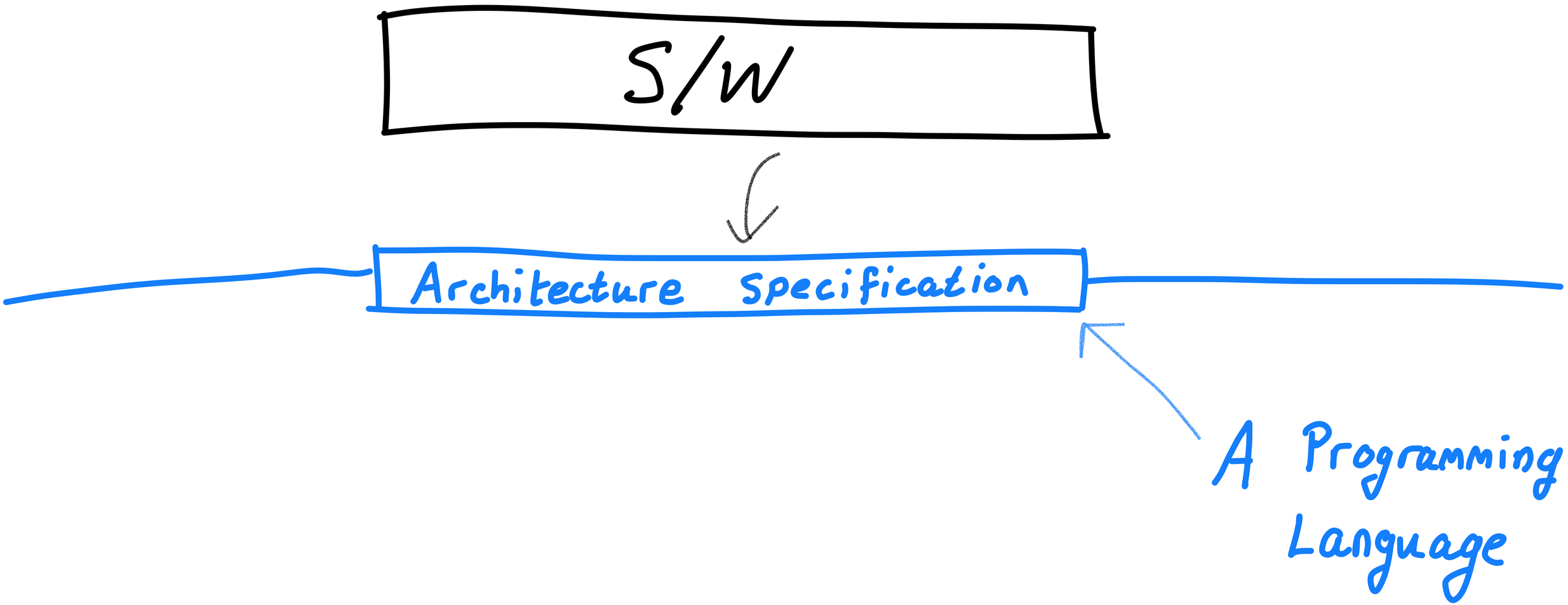
1. U. Cambridge 2. U. Edinburgh 3. U. Aarhus


Linux
RCU
JITs
Runtimes
Correct ?
Hardware


Linux
RCU
JITs
Runtimes
Correct ?
Need Model
w/ Exceptions
Hardware

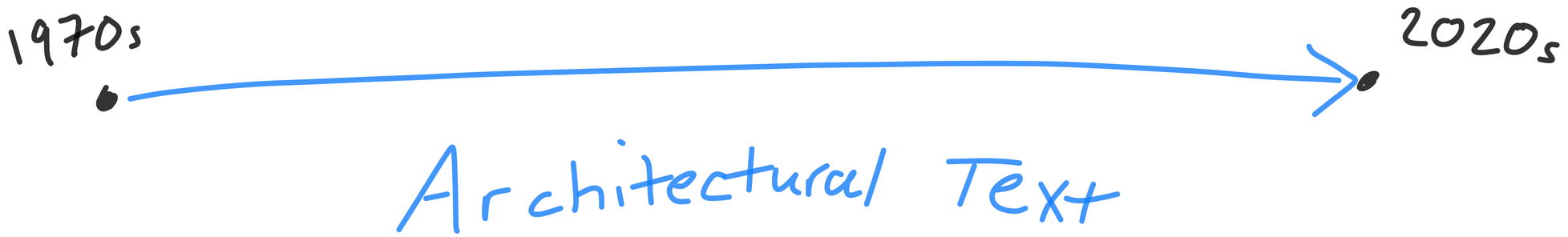

S/W
Architecture Specification
H/W

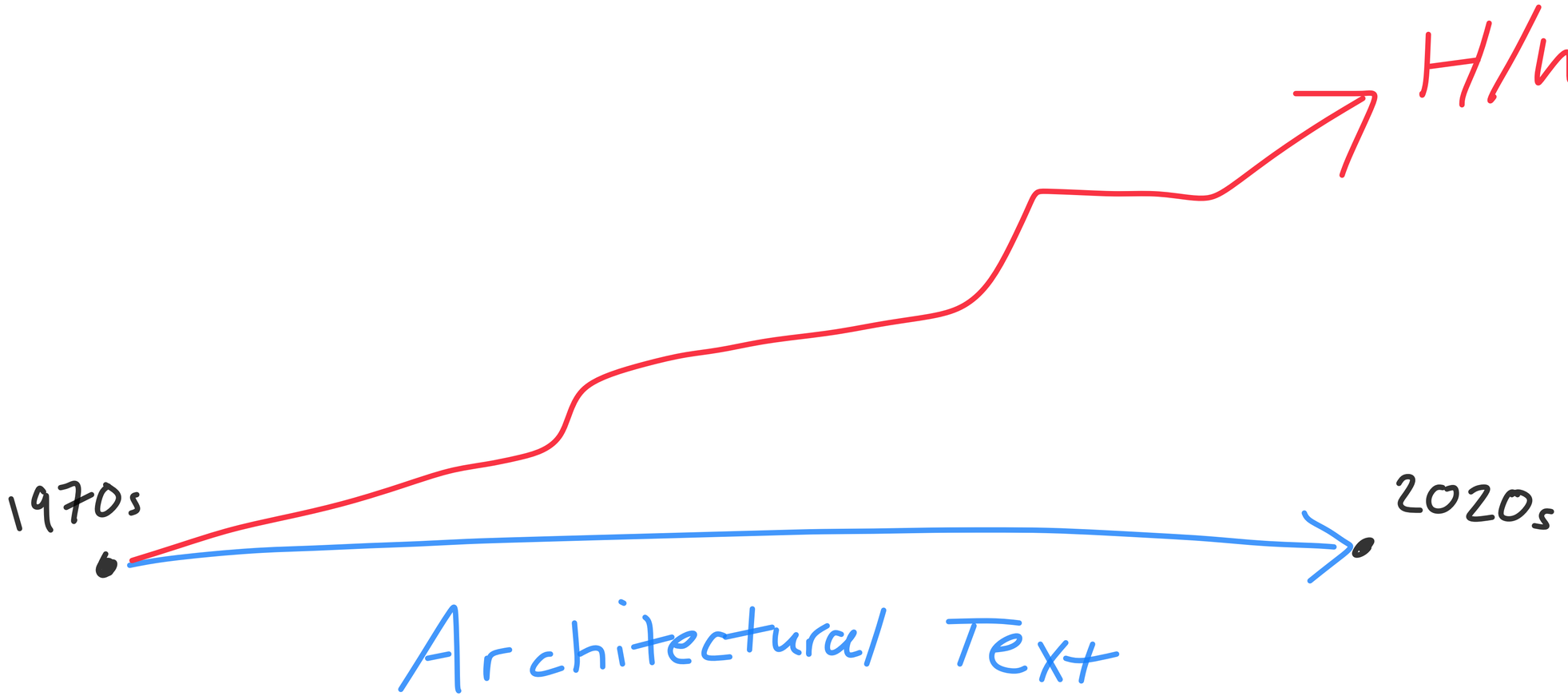

S/W
Architecture Specification
H/W
A Programming Language


S/W
Architecture Specification
A Programming Language

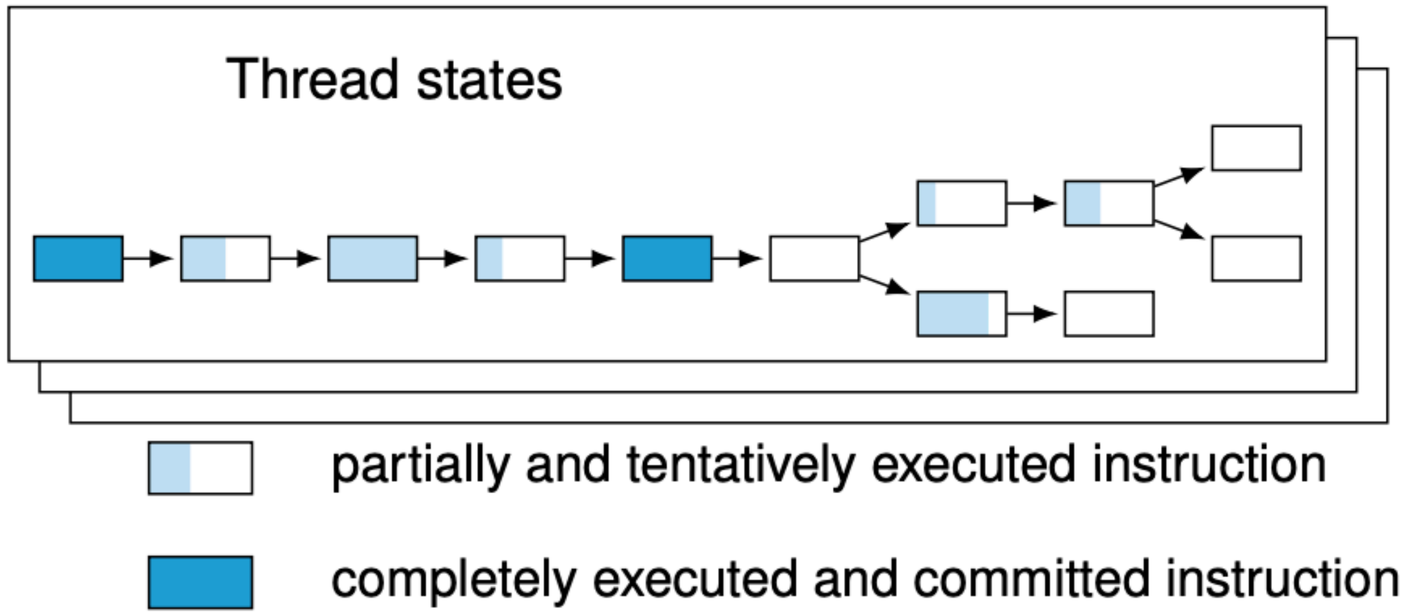
An exception is imprecise if the processor state when an exception is raised does not look exactly as if the instructions were executed sequentially in a strict program order

— Hennessy & Patterson, 5th ed, 2012

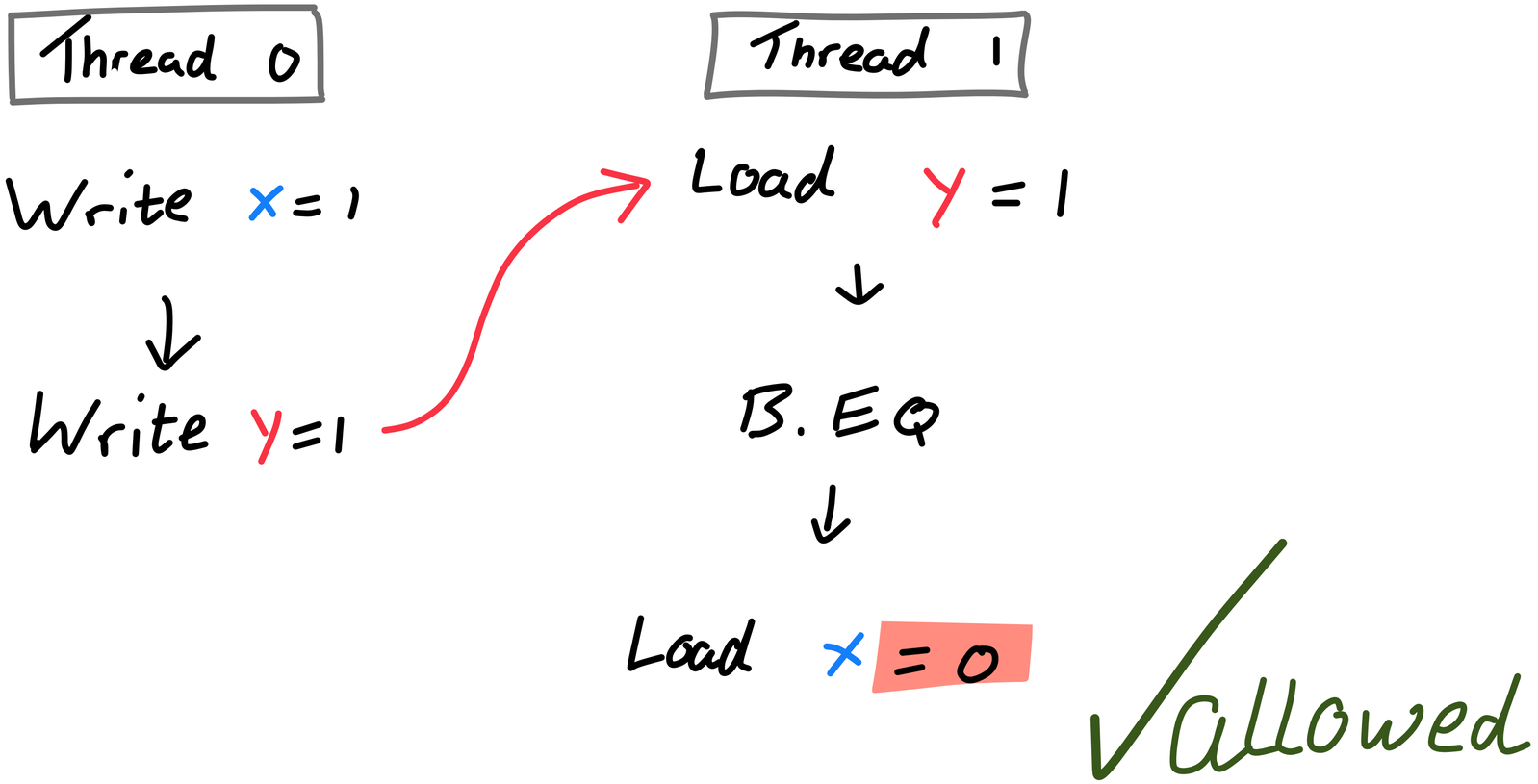

1970s
2020s
Architectural Text

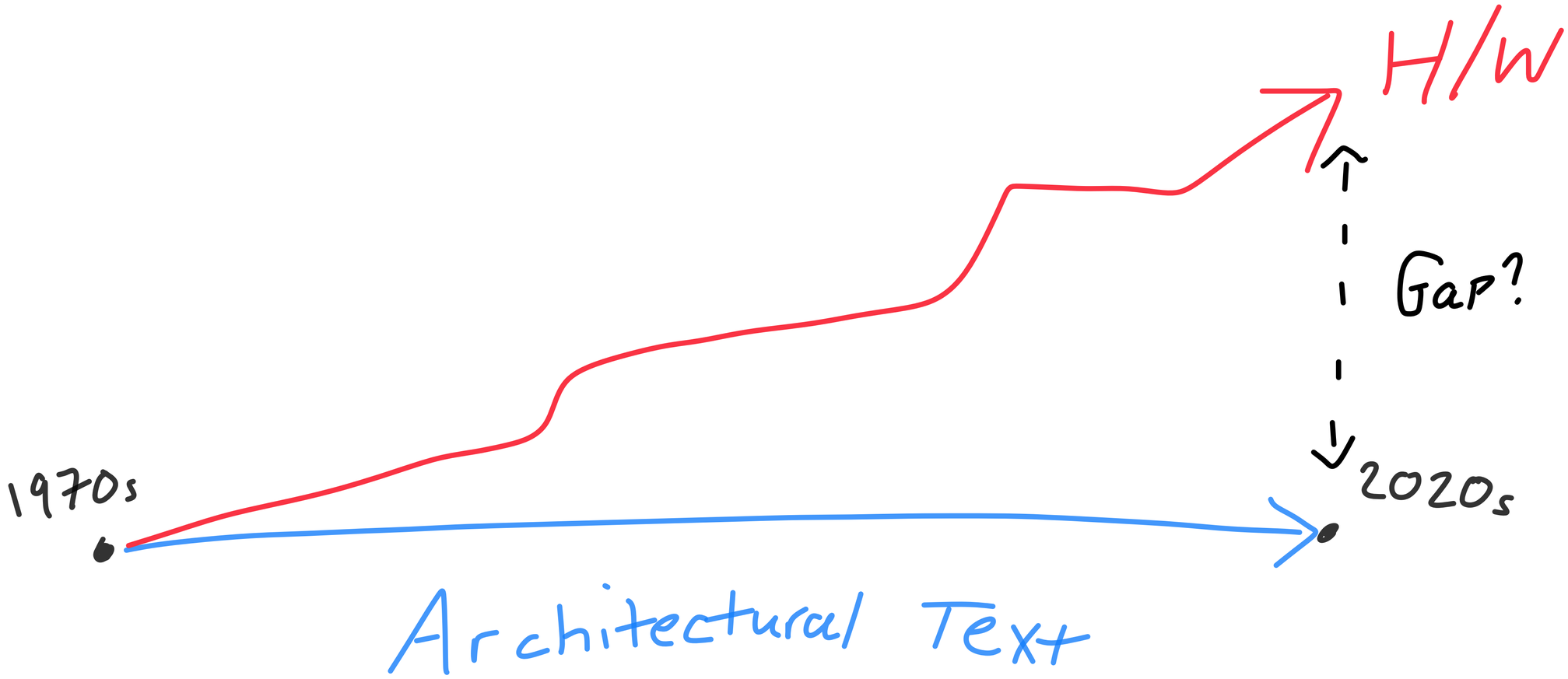

H/W
1970s
2020s
Architectural Text

Weak Memory — Observable Out-of-order Commit



Weak Memory — Observable Out-of-order Commit (Arm)




H/W
Gap?
1970s
2020s
Architectural Text

A Closer Look: Arm

An exception is precise if on taking the exception, the [register and memory] state is consistent with [...] having executed all the instructions up to [...] where the exception was taken, and none afterwards

— The Arm Architecture Reference Manual

A Closer Look: Arm

An exception is precise if on taking the exception, the [register and memory] state is consistent with [...] having executed all the instructions up to [...] where the exception was taken, and none afterwards

— The Arm Architecture Reference Manual

New

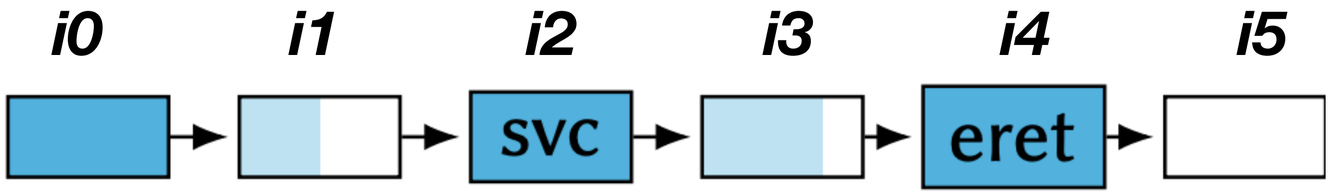
A Closer Look: Arm

Our Collected Data

Name	m6g	m7g	m8g	odroid	m2	pi3	pi4	pi5
MP+dmb+ctrl-svc	0/16M	0/24M	0/12M	0/329M	0/360M	0/10M	0/230M	0/136M
MP+dmb+ctrlelr	0/16M	0/24M	0/12M	0/329M	0/360M	0/30M	0/318M	0/130M
MP+svc-eret+addr	U0/16M	U0/24M	U0/12M	149K/328M	U0/360M	376/9M	U0/228M	12/136M
MP.EL1+dmb+dataesrsvc	0/16M	0/24M	0/12M	0/16M	0/0	0/4M	0/14M	0/27M
S+dmb+svc	U0/16M	U0/24M	U0/12M	U0/328M	U0/360M	U0/41M	U0/222M	U0/101M
SB+dmb+eret	60/16M	120/24M	213/12M	262/328M	12K/360M	203K/41M	946K/222M	4K/100M
SB+dmb+rfisvc-addr	4/16M	235/24M	1K/12M	305K/328M	12/360M	1M/30M	7K/316M	197K/128M
MP+dmb+fault	0/16M	0/24M	0/12M	0/74M	0/0	0/2M	0/46M	0/80M

$$\text{Data} = \frac{\#\,\text{seen}}{\#\,\text{runs}}$$

A Closer Look: Arm



Q: Behaviour = OoO commit
(as if exception not there?)

A Tour of Arm: Context Sychronisation (1/3)

① update context (write Sysreg)

② Take/Return from exception

③ Later instructions guaranteed to see new context

A Tour of Arm: Context Sychronisation (1/3)

① update context (write Sysreg)

② Take/Return from exception ⇒ Wait for Control-flow

③ Later instructions guaranteed to see new context

A Tour of Arm: Context Sychronisation (1/3)

① update context (write Sysreg)

② Take/Return from exception ⇒ Wait for Control-flow

③ Later instructions guaranteed to see new context

S/W relies on

A tour of Arm: External Aborts (2/3)

① LD from memory

② ST to memory

A tour of Arm: External Aborts (2/3)

Can fail ⟶ ① LD from memory

② ST to memory

A tour of Arm: External Aborts (2/3)

can fail ⟶ ① LD from memory

② ST to memory

if: can fail sync

then: No OoO Propagate

A tour of Arm: External Aborts (2/3)

can fail ⟶ ① LD from memory

② ST to memory

if: can fail sync

then: No OoO Propagate

order!

A tour of Arm: imprecise inconsistency (3/3)

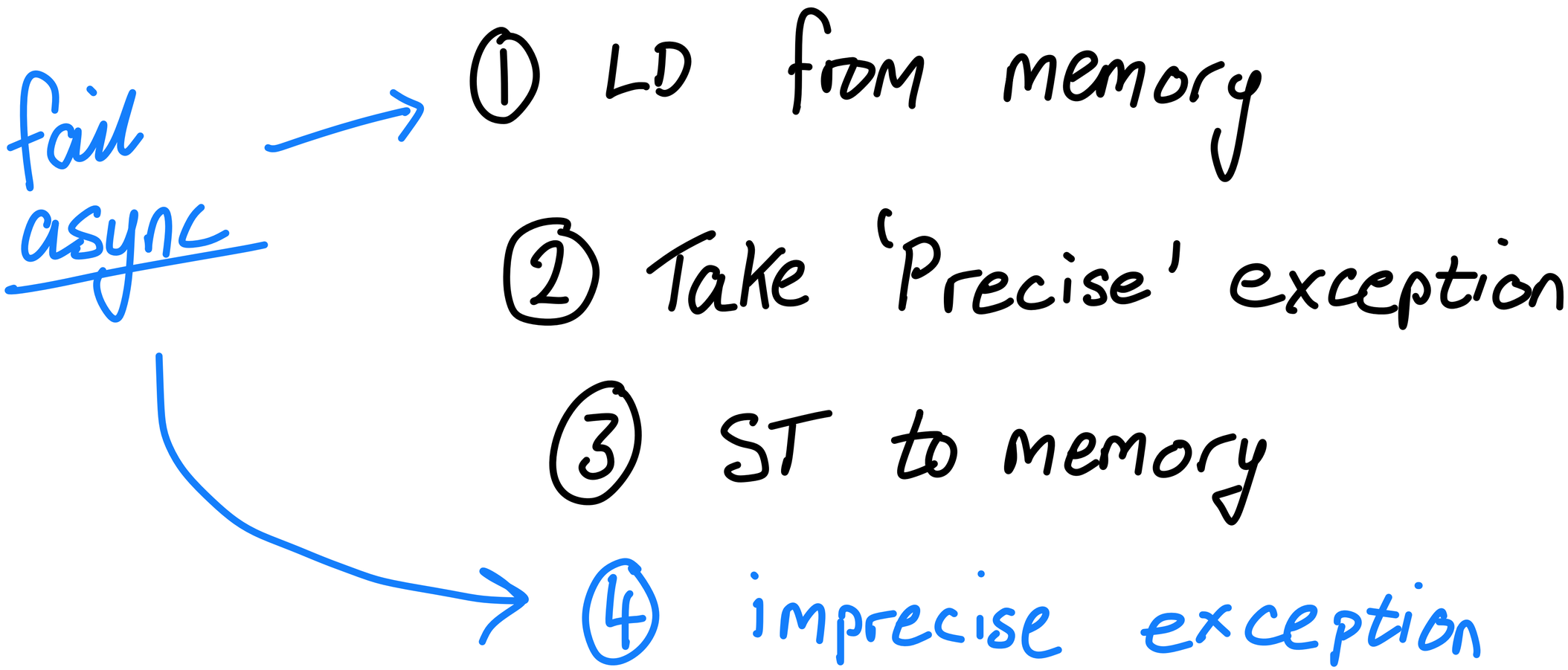
fail async →

① LD from memory

② Take 'Precise' exception

③ ST to memory

A tour of Arm: imprecise inconsistency (3/3)



A tour of Arm: imprecise inconsistency (3/3)

fail async

① ~~LD from memory~~ Never completes

② Take 'Precise'? exception

③ ST to memory

④ imprecise exception

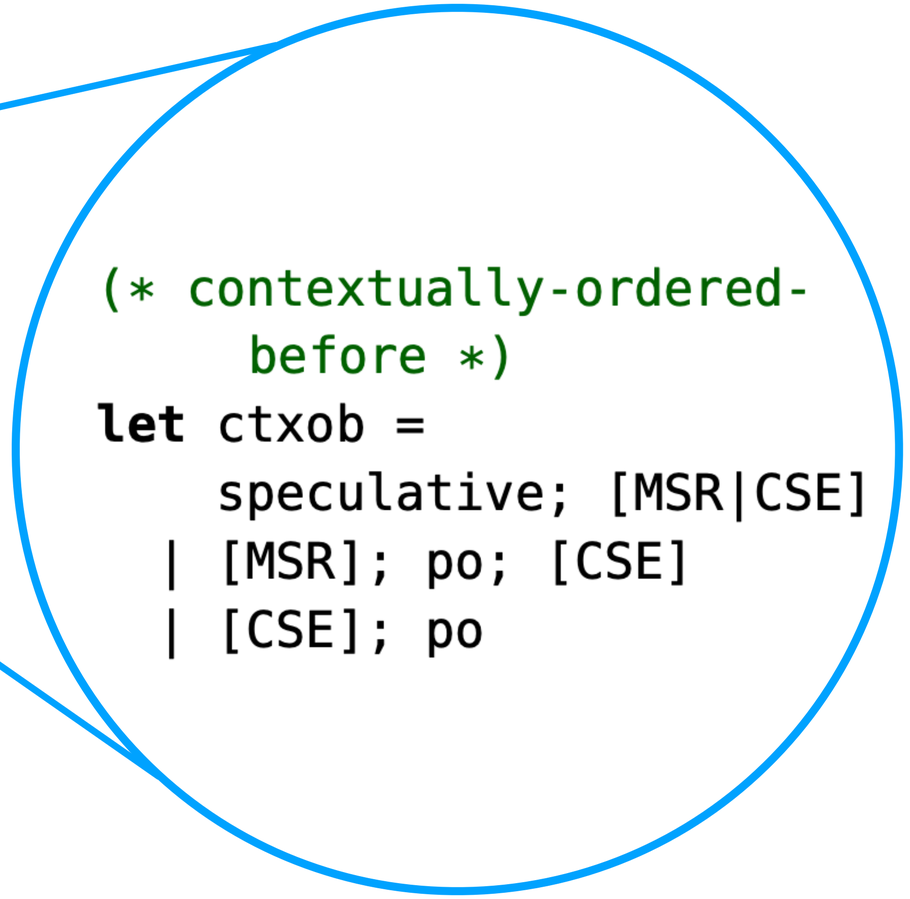
What is precision ?

Our Contributions

- Catalogued a bunch of behaviours for Arm
  - ↳ clarifying the intent
- Unearthing reality of Exceptions today
  - ↳ Model + Test harness

```
"Arm-A exceptions"

include "cos.cat"
include "arm-common.cat"

(* might-be speculatively
     executed *)
let speculative =
    ctrl
  | addr; po
  | if "SEA_R" then [R]; po
      else 0
  | if "SEA_W" then [W]; po
      else 0

(* context-sync-events *)
let CSE =
    ISB
  | if "FEAT_ExS" & ~"EIS"
      then 0 else TE
  | if "FEAT_ExS" & ~"EOS"
      then 0 else ERET

let ASYNC =
  TakeInterrupt

(* observed by *)
let obs = rfe | fr | co

(* dependency-ordered-
     before *)
let dob =
    addr | data
  | speculative ; [W]
  | speculative ; [ISB]
  | (addr | data); rfi

(* atomic-ordered-before *)
let aob =
    rmw
 | [range(rmw)]; rfi; [A|Q]
```

```
(* barrier-ordered-before
     *)
let bob =
    [R] ; po ; [dmbld]
  | [W] ; po ; [dmbst]
  | [dmbst]; po; [W]
  | [dmbld]; po; [R|W]
  | [L]; po; [A]
  | [A | Q]; po; [R | W]
  | [R | W]; po; [L]
  | [dsb]; po

(* contextually-ordered-
     before *)
let ctxob =
    speculative; [MSR|CSE]
  | [MSR]; po; [CSE]
  | [CSE]; po

(* async-ordered-before *)
let asyncob =
    speculative; [ASYNC]
  | [ASYNC]; po

(* Ordered-before *)
let ob = (obs | dob | aob |
  bob | ctxob | asyncob)+

(* Internal visibility
     requirement *)
acyclic po-loc | fr | co |
     rf as internal

(* External visibility
     requirement *)
irreflexive ob as external

(* Atomic: Basic LDXR/STXR
     constraint to forbid
     intervening writes. *)
empty rmw & (fre; coe) as
     atomic
```



```
"Arm-A exceptions"

include "cos.cat"
include "arm-common.cat"

(* might-be speculatively
     executed *)
let speculative =
    ctrl
  | addr; po
  | if "SEA_R" then [R]; po
      else 0
  | if "SEA_W" then [W]; po
      else 0

(* context-sync-events *)
let CSE =
    ISB
  | if "FEAT_ExS" & ~"EIS"
      then 0 else TE
  | if "FEAT_ExS" & ~"EOS"
      then 0 else ERET

let ASYNC =
  TakeInterrupt

(* observed by *)
let obs = rfe | fr | co

(* dependency-ordered-
     before *)
let dob =
    addr | data
  | speculative ; [W]
  | speculative ; [ISB]
  | (addr | data); rfi

(* atomic-ordered-before *)
let aob =
    rmw
 | [range(rmw)]; rfi; [A|Q]
```

```
(* barrier-ordered-before
     *)
let bob =
    [R] ; po ; [dmbld]
  | [W] ; po ; [dmbst]
  | [dmbst]; po; [W]
  | [dmbld]; po; [R|W]
  | [L]; po; [A]
  | [A | Q]; po; [R | W]
  | [R | W]; po; [L]
  | [dsb]; po

(* contextually-ordered-
     before *)
let ctxob =
    speculative; [MSR|CSE]
  | [MSR]; po; [CSE]
  | [CSE]; po

(* async-ordered-before *)
let asyncob =
    speculative; [ASYNC]
  | [ASYNC]; po

(* Ordered-before *)
let ob = (obs | dob | aob |
  bob | ctxob | asyncob)+

(* Internal visibility
     requirement *)
acyclic po-loc | fr | co |
     rf as internal

(* External visibility
     requirement *)
irreflexive ob as external

(* Atomic: Basic LDXR/STXR
     constraint to forbid
     intervening writes. *)
empty rmw & (fre; coe) as
     atomic
```

```
(* contextually-ordered-
     before *)
let ctxob =
    speculative; [MSR|CSE]
  | [MSR]; po; [CSE]
  | [CSE]; po
```

Model of interactions —

Not definition of Precision

Call to ISCA

- Use the model
  - Verify H/w?
  - Model-check S/W
- Other architectures?
  - Catalogue x86/RISC-V
    (Separately ☹)

Call to ISCA

- Use the model
  - Verify H/w?
  - Model-check S/W
- Other architectures?
  - Catalogue x86/RISC-V (Separately ☹)
  - Unified Definition?

Call to ISCA

- Use the model
  - Verify H/w?
  - Model-check S/w
- Other architectures?
  - Catalogue x86/RISC-V (Separately ☹)
  - Unified Definition?

Thanks!