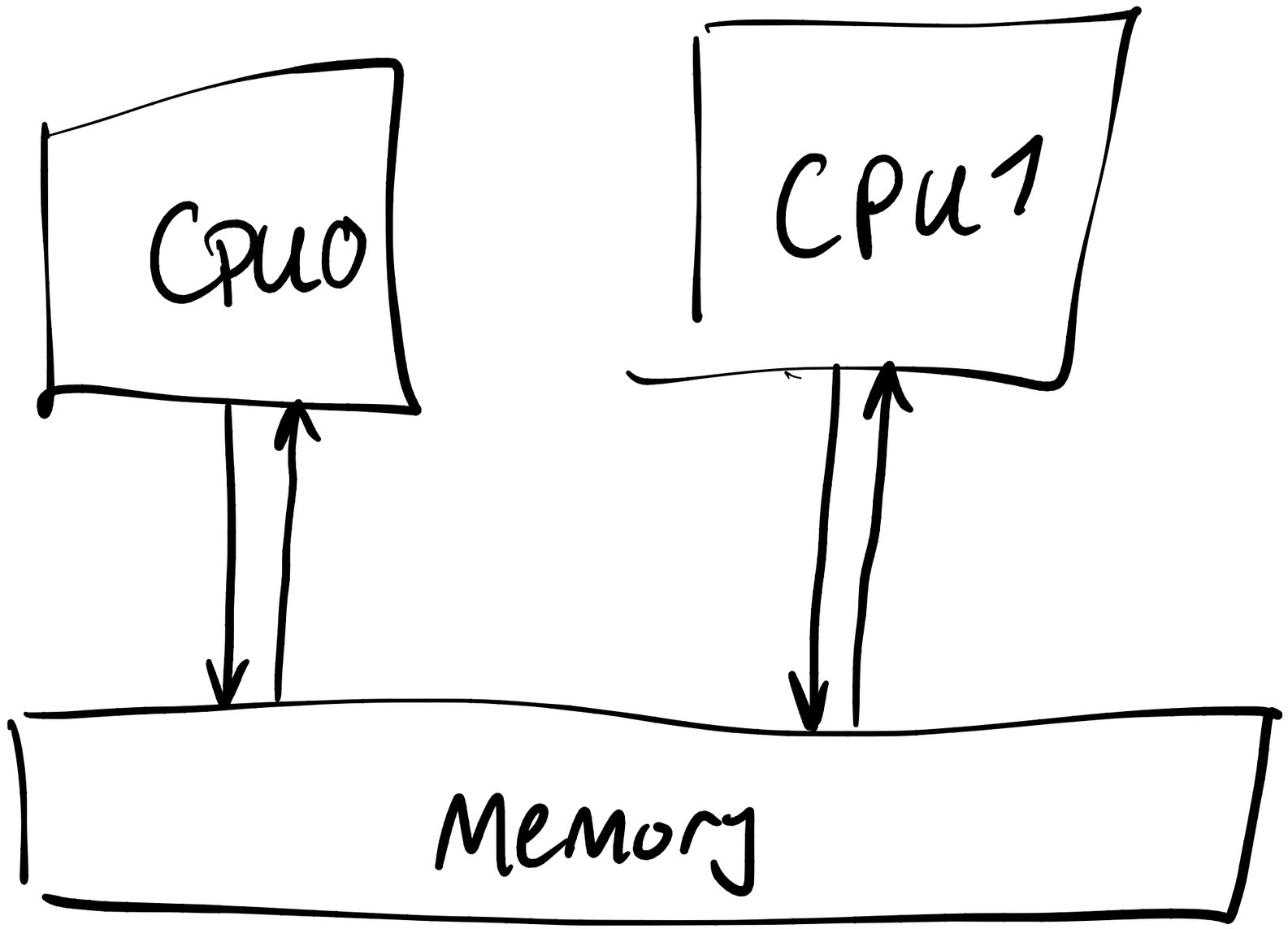
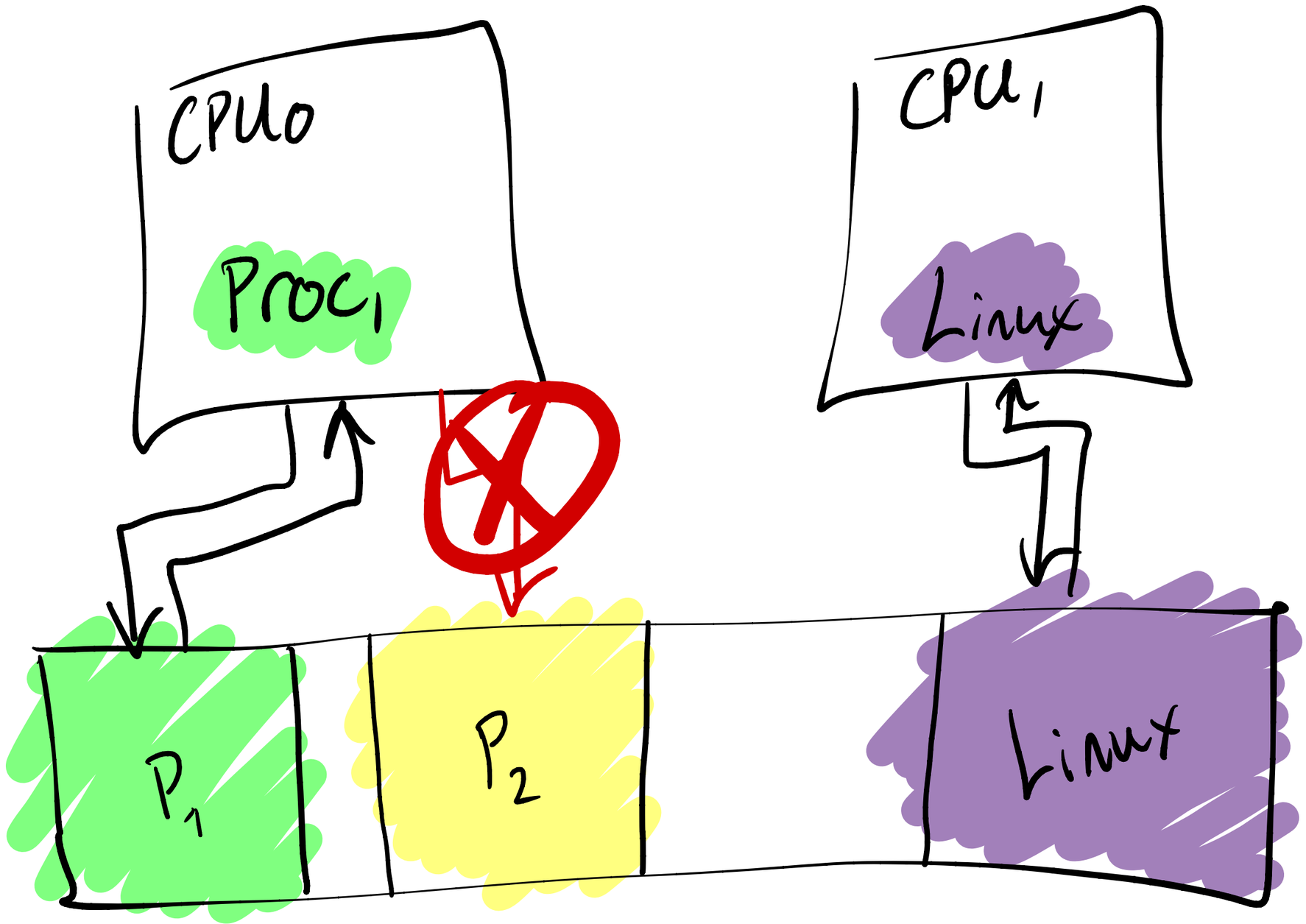
Relaxed Virtual Memory — Armv8 edition —

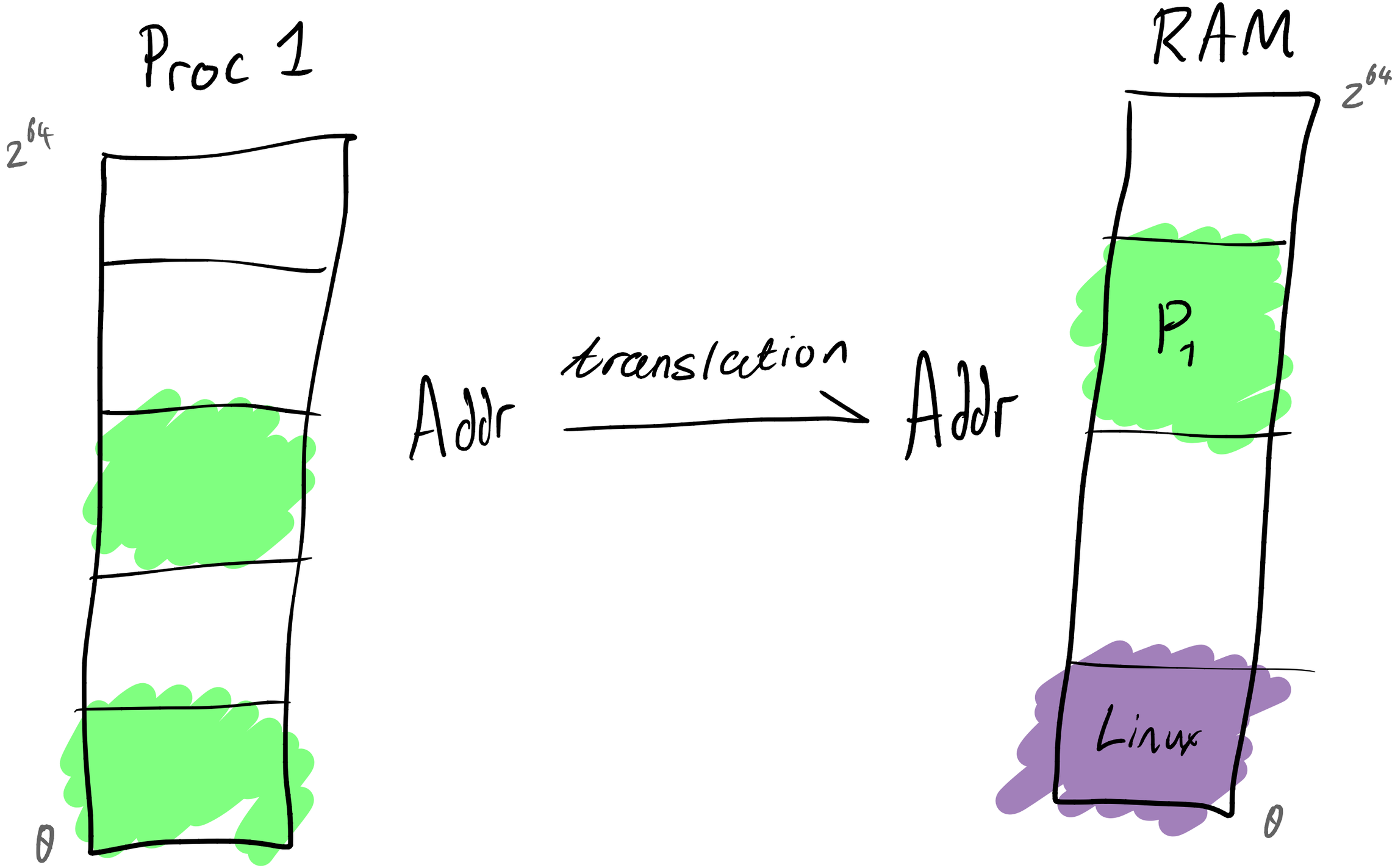
Ben Simner[1], Alasdair Armstrong[1], Jean Pichon-Pharabod[2], Christopher Pulte[1], Richard Grisenthwaite[3], Peter Sewell[1]

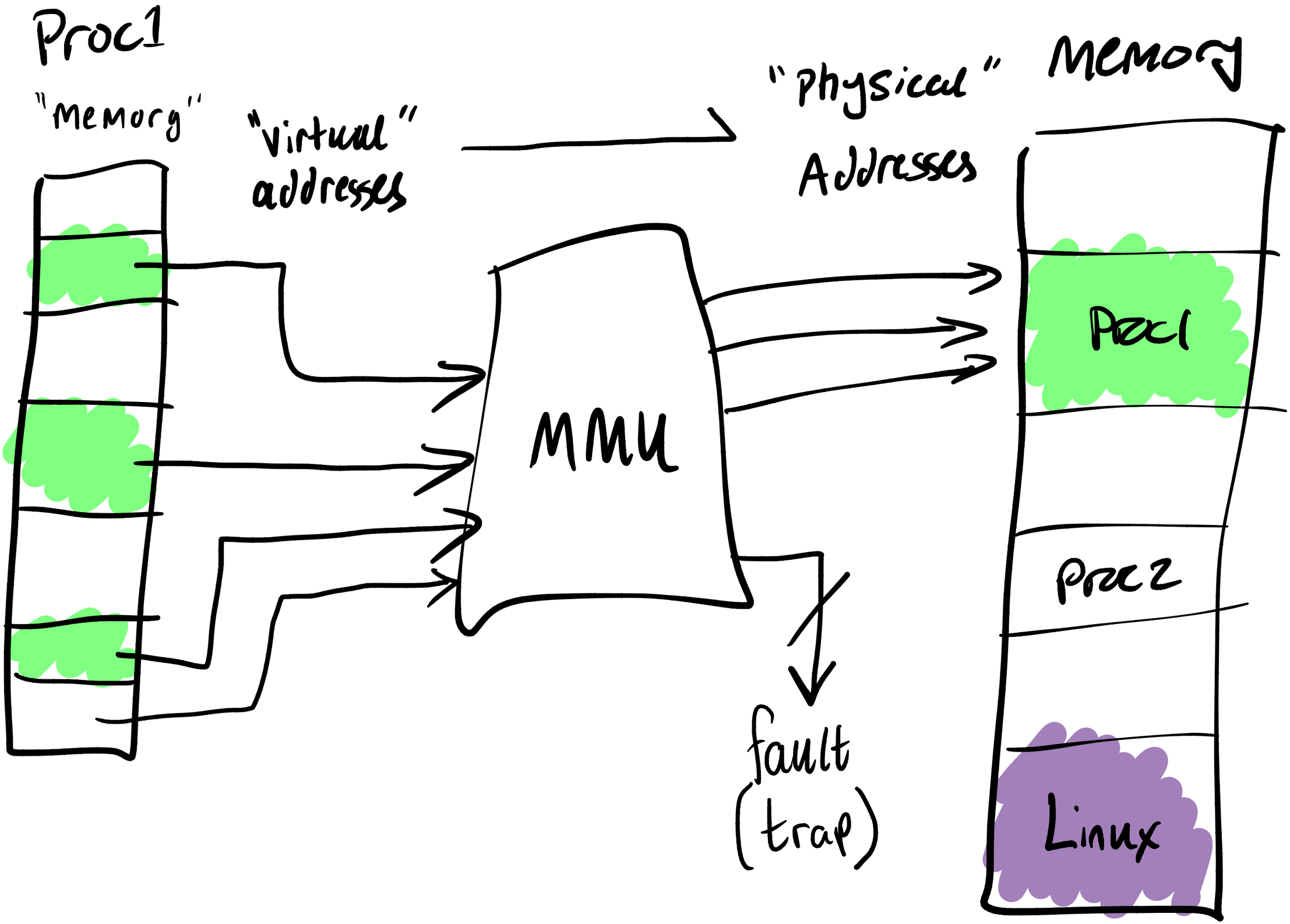
1. Cambridge, UK 2. Aarhus, DK 3. Arm Ltd., UK

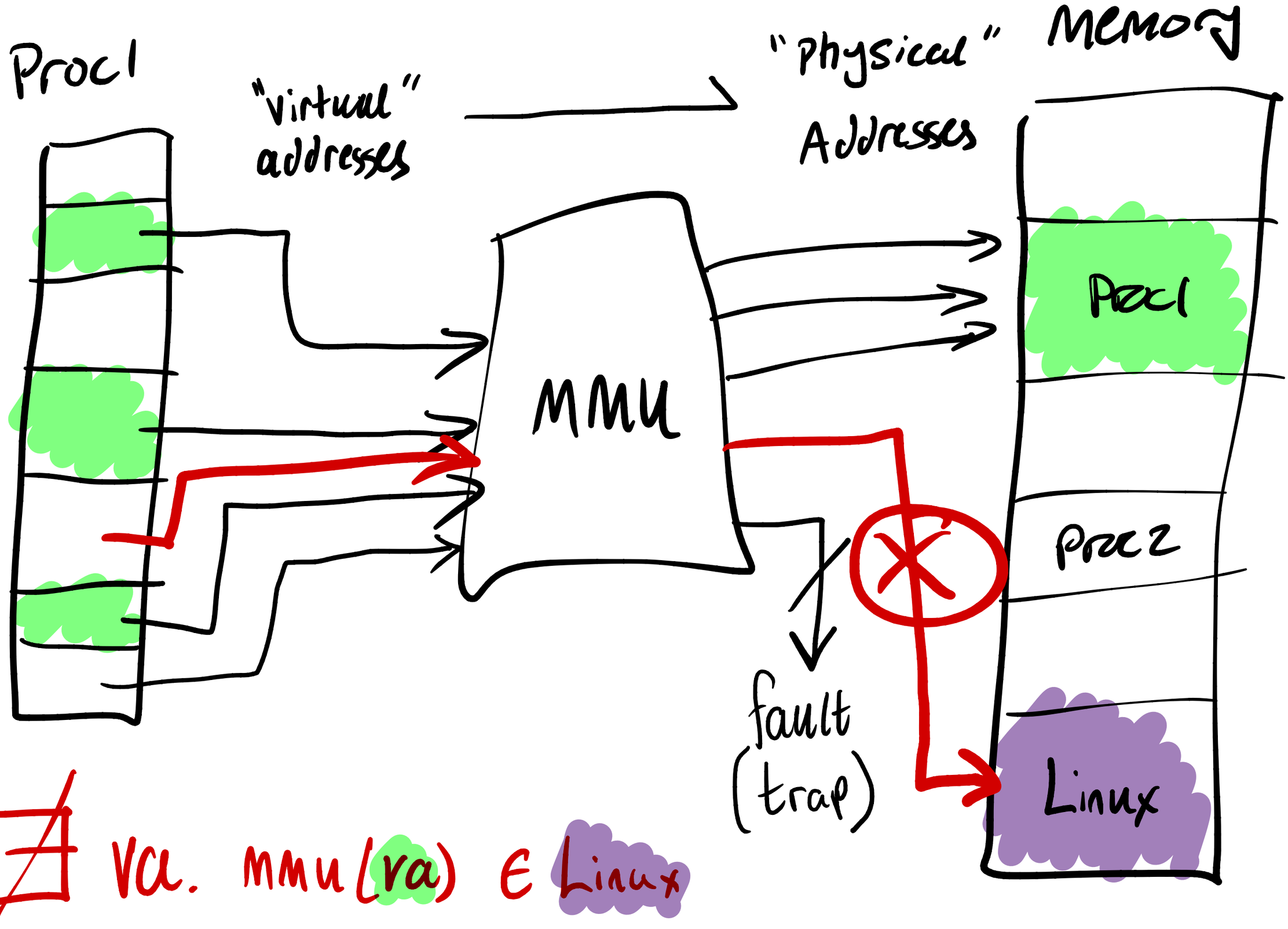

CPU0
CPU1
Memory


CPU0
Proc1
CPU1
Linux
P1
P2
Linux

Virtual Memory




Proc1
"memory"
"virtual" addresses
"physical" Addresses
memory
MMU
fault (trap)
Proc1
Proc2
Linux



$\nexists$ va. mmu(va) $\in$ Linux

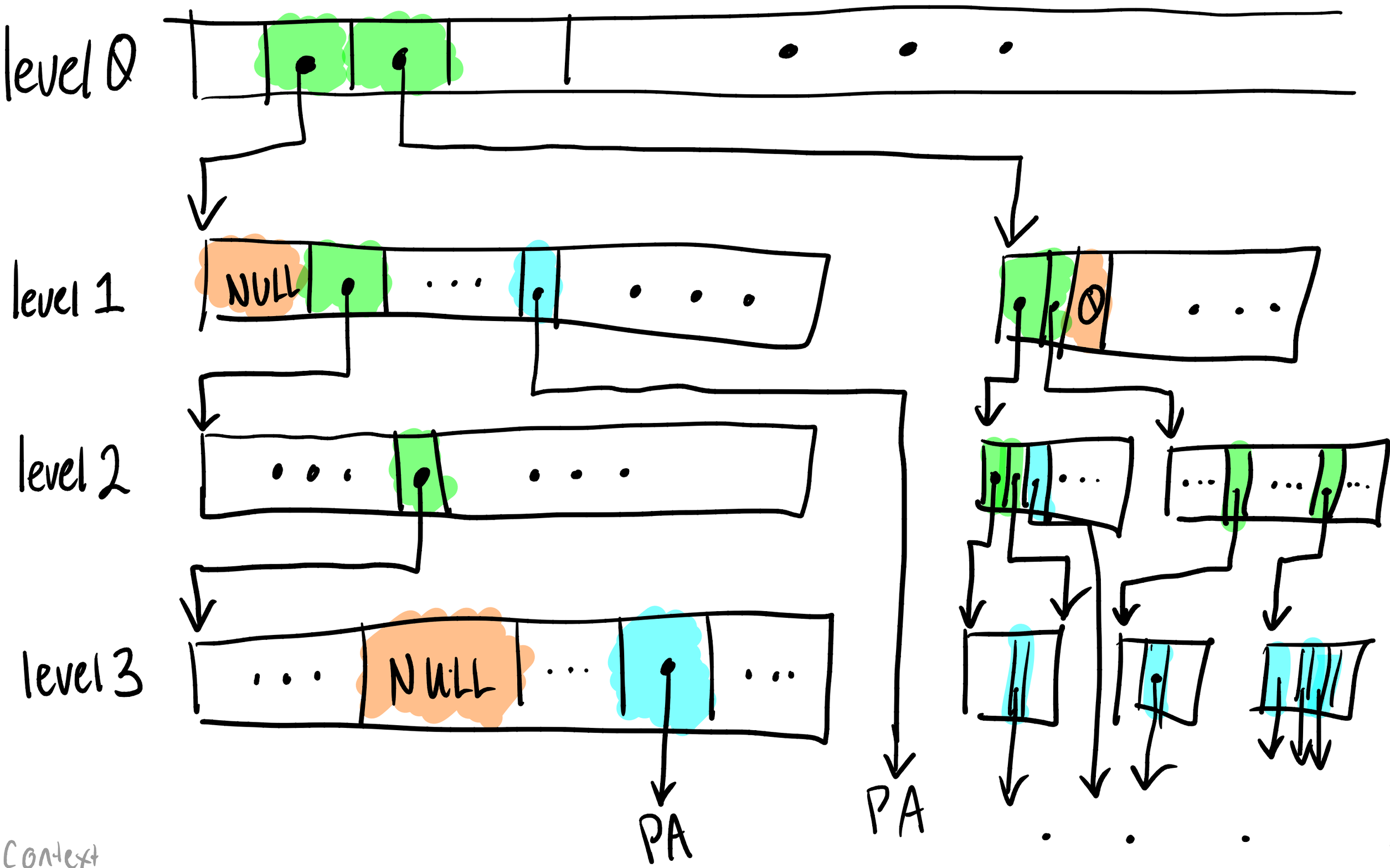
Aims

- Define Formal Semantics
- Build tooling

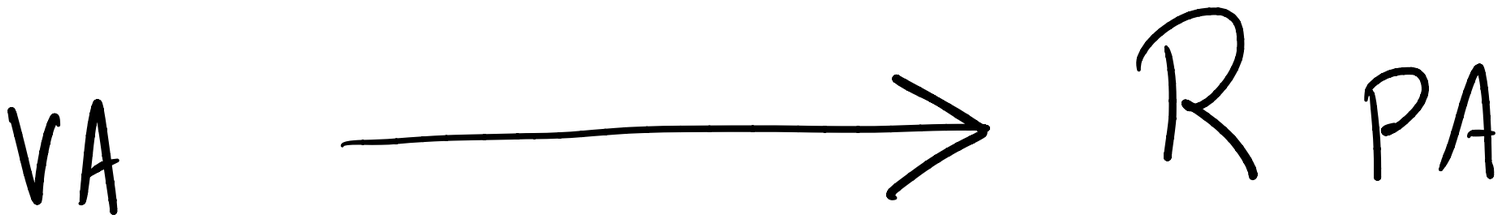
} this work

- Reason about Systems

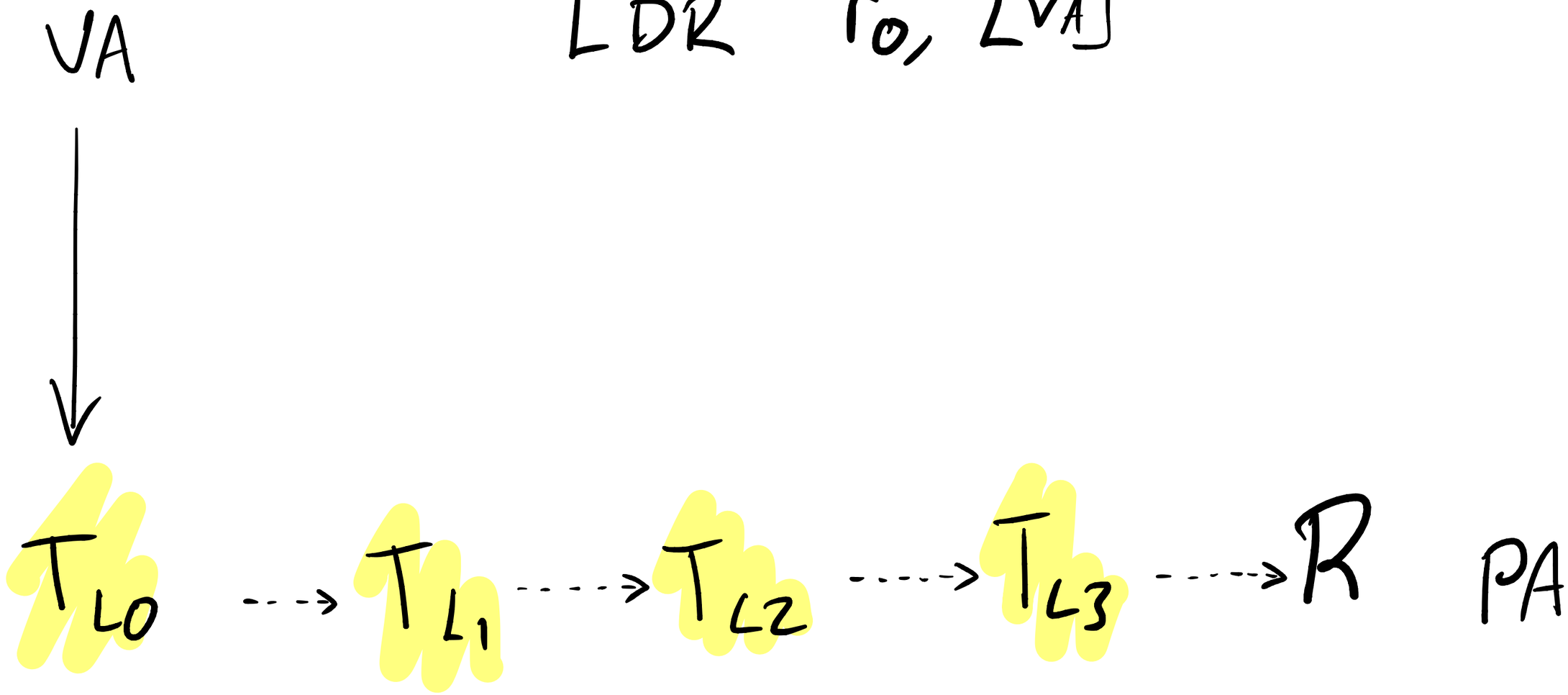
In-memory Data Structure

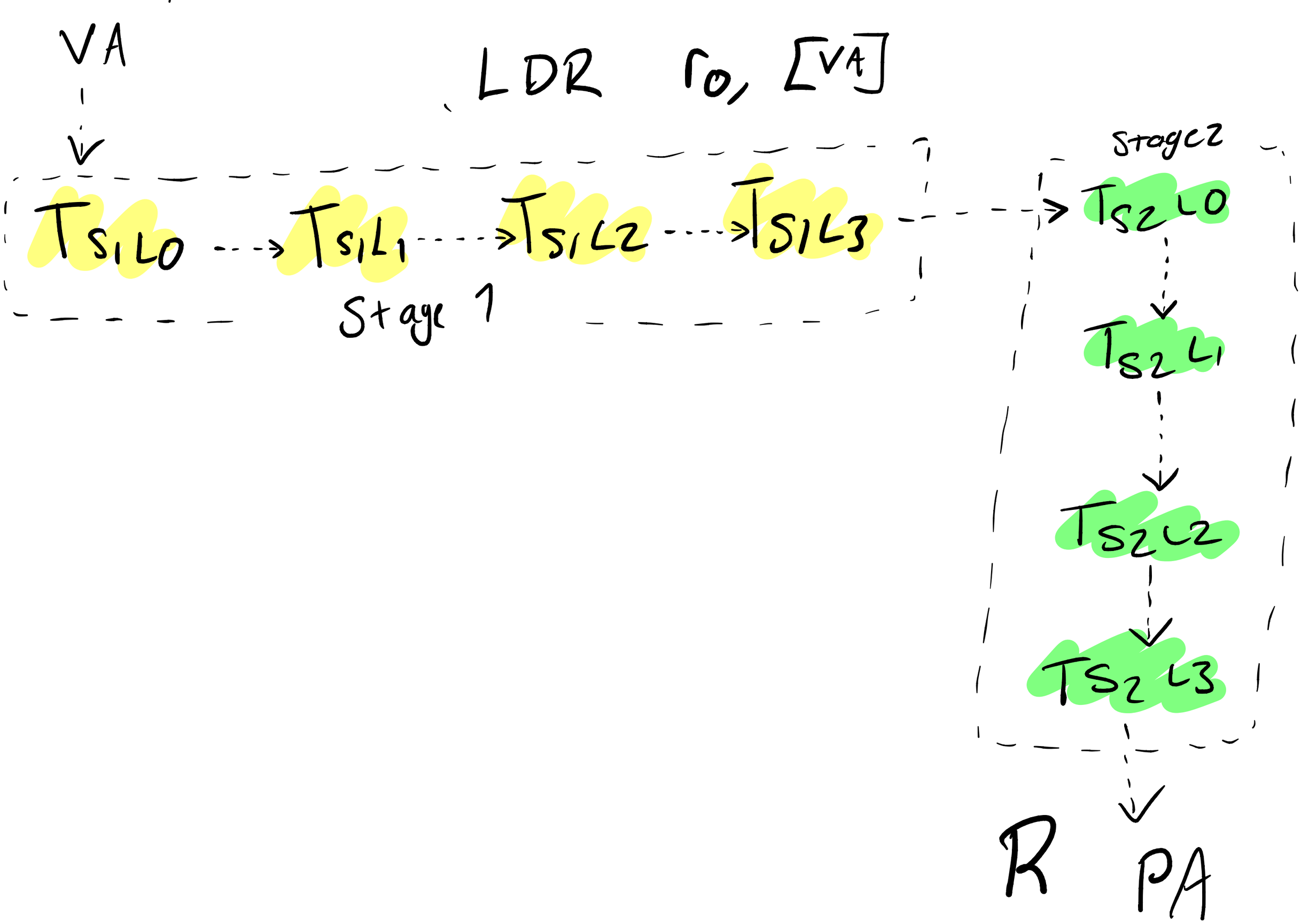


LDR r0, [VA]



LDR r0, [VA]




VA
LDR r0, [VA]
Stage 1
Stage 2
TS1L0
TS1L1
TS1L2
TS1L3
TS2L0
TS2L1
TS2L2
TS2L3
R
PA

LDR r0, [VA]

VA

$T_{S2}L0$ → $T_{S2}L1$ → $T_{S2}L2$ → $T_{S2}L3$ → $T_{S1}L0$

$T_{S2}L0$ → $T_{S2}L1$ → $T_{S2}L2$ → $T_{S2}L3$ → $T_{S1}L1$

$T_{S2}L0$ → $T_{S2}L1$ → $T_{S2}L2$ → $T_{S2}L3$ → $T_{S1}L2$

$T_{S2}L0$ → $T_{S2}L1$ → $T_{S2}L2$ → $T_{S2}L3$ → $T_{S1}L3$

$T_{S2}L0$ → $T_{S2}L1$ → $T_{S2}L2$ → $T_{S2}L3$ → R PA

LDR r0, [VA]

VA

TS2 L0

TS2 L1

TS2 L2

TS2 L3

TS1 L0

TS2 L0

TS2 L1

TS2 L2

TS2 L3

TS1 L1

TS2 L0

TS2 L1

TS2 L2

TS2 L3

TS1 L2

TS2 L0

TS2 L1

TS2 L3

TS2 L0

TS2 L1

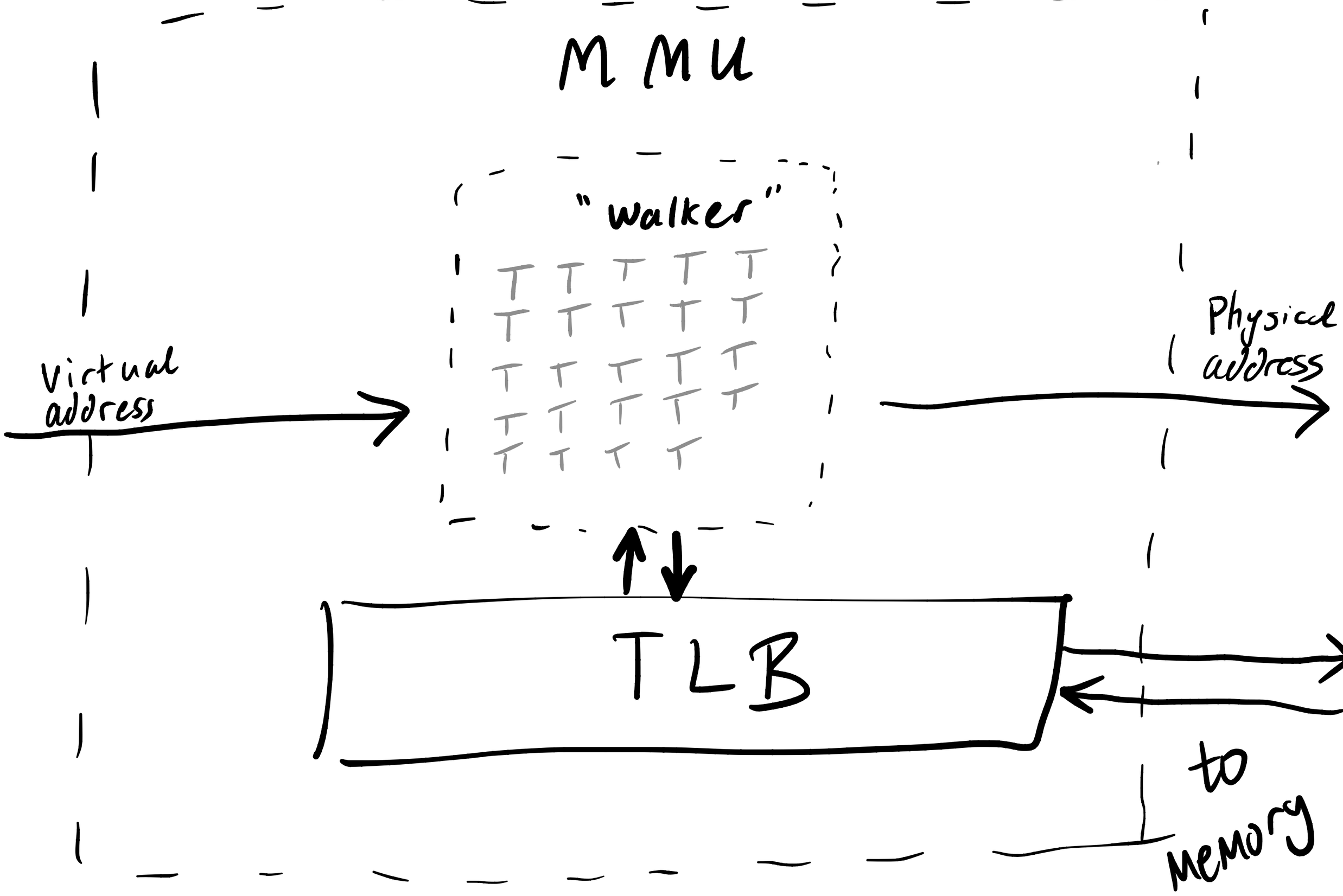
TS2 L2

TS2 L3

R

PA

+24


MMU
"walker"
Virtual address
Physical address
TLB
to memory

Complexity

- TLB, Pipeline & cache effects
- Concurrency

} 11,530 page Arm Manual

Behaviour — Pipeline

Init: $x \mapsto$ NULL

Code:

STR New, [pte(x)]

LDR r0, [x]

Can see old.

Execution:

W pte(x) = New

↓ po

T x= NULL → Fault

↑ trf

allowed!

Behaviour — Aliasing

init: x ↦ P, y ↦ P,
P = 0

---

code:

STR 1, [x]

LDR r0, [y]

Can't see old

$T_{x=P} \longrightarrow W_{P=1}$

↓ Po

rf

$T_{y=P} \longrightarrow R_{P=0}$

(forbidden)

Behaviour — TLB

```
Init: x ↦ P
-----------
Code:
  STR NULL, [ptec(x)]
  DSB SY
  ISB
  LDR r0, [x]
```

Thread 0

W ptec(x) = NULL

↓ flush-pipe

$T_{x=P}$ → R old

allowed!

Behaviour — Break-before-make

Thread 0

W pte(x)=NULL

↓

DSB SY

↓

TLBI

↓

DSB SY

↓

W pte(x)=New

Thread 1

$T_x \longrightarrow R$

... Lots of Interactions

- Spontaneous walks
- Coherent TLB fills
- Order within a translation table walk
- Speculative translations.
- Micro-TLBs
- Multi-copy Atomicity
- ETS
- Break-before-make
- write-forwarding
- ASIDs and VMIDs

Axiomatic Model

TLB

```
let tlb-affects =
  ...

let TLB_barrier =
  ([TLBI] ; tlb-affects ; [T] ; tfr ; [W])^-1 & wco

let maybe_TLB_cached =
  ([T] ; trf^-1 ; wco ; [TLBI-S1]) & tlb-affects^-1

let tcache1 = [T & Stage1] ; tfr ; TLB_barrier
let tcache2 = [T & Stage2] ; tfr ; TLB_barrier
```

Walk + Speculation

```
let speculative =
    ctrl
  | addr; po
  | [T] ; instruction-order
(* translation-ordered-before *)
let tob =
    [T_f] ; tfre
  | ([T_f] ; tfri) & (po ; [DSB.SY] ; instruction-order)^-1
  | [T] ; iio ; [R|W] ; po ; [W]
  | speculative ; trfi
(* observed by *)
let obs = rfe | fr | wco
  | trfe
(* ordered-before TLBI and translate *)
let obtlbi_translate =
    tcache1
  | tcache2 & (iio^-1 ; [T & Stage1] ; trf^-1 ; wco^-1)
  | (tcache2 ; wco? ; [TLBI-S1]) & (iio^-1 ; [T & Stage1] ;
    maybe_TLB_cached)

(* ordered-before TLBI *)
let obtlbi =
    obtlbi_translate
  | [R|W|Fault] ; iio^-1 ; (obtlbi_translate & ext) ; [TLBI]

(* context-change ordered-before *)
let ctxob =
    speculative ; [MSR]
  | [CSE] ; instruction-order
  | [ContextChange] ; po ; [CSE]
  | speculative ; [CSE]
  | po ; [ERET] ; instruction-order ; [T]

(* ordered-before a translation fault *)
let obfault =
```

MCA

TLB maintenance

Pipeline

FEAT_ETS

```
    data ; [Fault & IsFromW]
  | speculative ; [Fault & IsFromW]
  | [dmbst] ; po ; [Fault & IsFromW]
  | [dmbld] ; po ; [Fault & (IsFromW|IsFromR)]
  | [A|Q] ; po ; [Fault & (IsFromW | IsFromR)]
  | [R|W] ; po ; [Fault & IsFromW & IsReleaseW]

(* ETS-ordered-before *)
let obETS =
    (obfault ; [Fault]) ; iio^-1 ; [T_f]
  | ([TLBI] ; po ; [dsb] ; instruction-order ; [T]) & tlb-affects

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

(* 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]
  | [F | C]; po; [dsbsy]
  | [dsb] ; po

(* Ordered-before *)
let ob = (obs | dob | aob | bob | iio | tob | obtlbi | ctxob |
    obfault | obETS)^+

(* Internal visibility requirement *)
acyclic po-loc | fr | co | rf as internal
(* External visibility requirement *)
irreflexive ob as external
(* Atomic requirement *)
empty rmw & (fre; coe) as atomic
(* Writes cannot forward to po-future translates *)
acyclic (po-pa | trfi) as translation-internal
```

Axioms

Contrib - Model

isla – axiomatic

isla-axiomatic.cl.cam.ac.uk/#

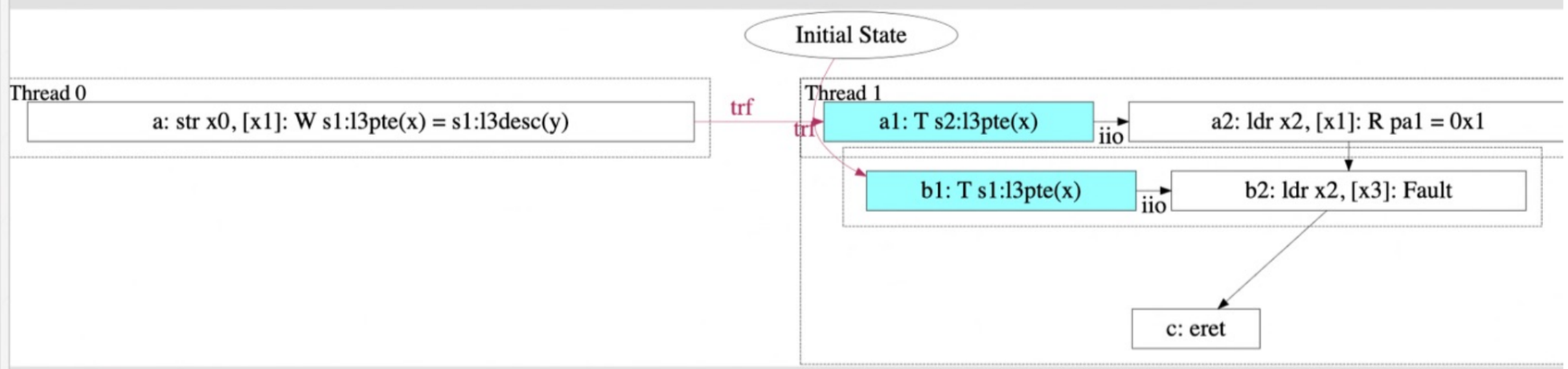
Isla | W / aarch64-vmsa-strong | Litmus file | Memory model | Sail architecture (*aarch64-vmsa*) | **Run test** | Options | Share

W.toml | Memory model

```
    identity 0x1000 with code;
"""

[thread.0]
init = {}
code = """
    STR X0,[X1]
"""

[thread.0.reset]
R0 = "desc3(y, page_table_base)"
R1 = "pte3(x, page_table_base)"

[thread.1]
init = {}
code = """
    LDR X2,[X1]
    MOV X0,X2
    LDR X2,[X3]
"""

[thread.1.reset]
R1 = "x"
R3 = "x"
VBAR_EL1 = "extz(0x1000, 64)"

"PSTATE.SP" = "0b0"
"PSTATE.EL" = "0b00"

[section.thread1_el1_handler]
address = "0x1400"
code = """
    MOV X2,#0

    MRS X13,ELR_EL1
    ADD X13,X13,#4
    MSR ELR_EL1,X13
    ERET
"""

[final]
assertion = "1:X0=1 & 1:X2=0"
```

EventGraph ×

Relations ← 1 of 1 →



Contrib-.tools

Recap

① Clarified Architecture

② Defined Axiomatic model

③ Built tooling

Future

use the model!

more Arm features

... more architectures?