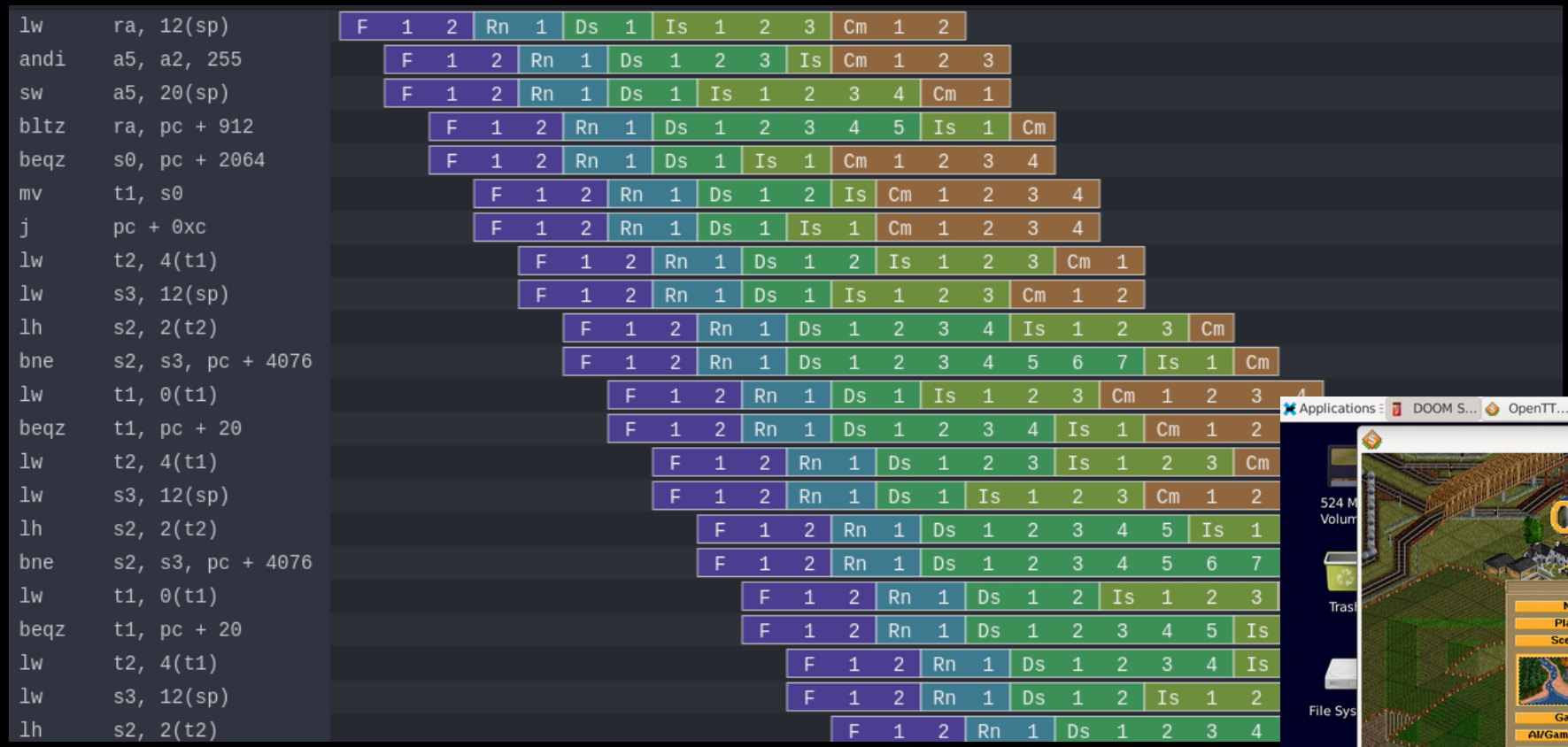
Open CPU / SoC design, all the way up to Debian



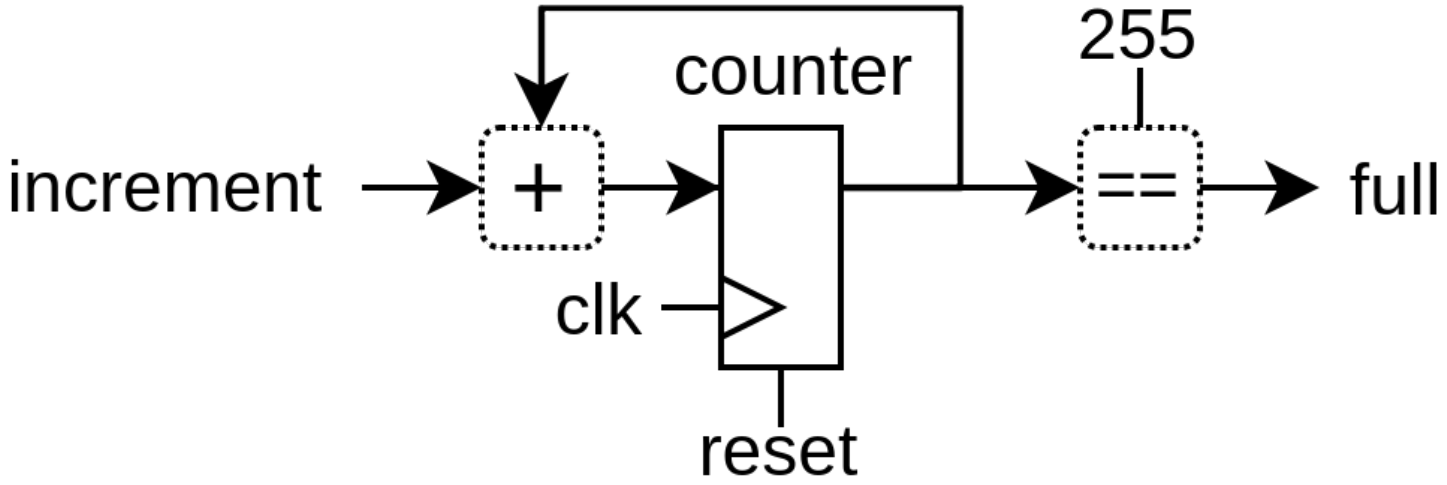
Background / whoami

- Dolu1990 on github
- Active on open/free project
  - SpinalHDL / VexRiscv / NaxRiscv
- Software / Hardware background
  - Without computer science degree
  - With Industrial system / Electronic degree

Introduction

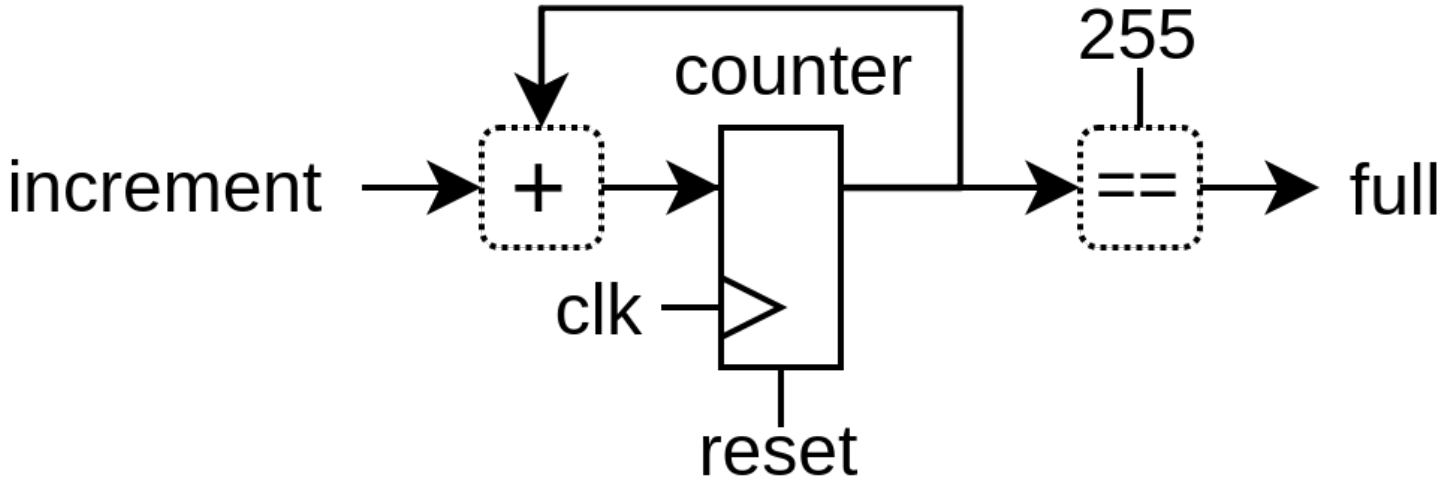
- This talk will be based on NaxRiscv
  - https://github.com/SpinalHDL/NaxRiscv
- What the talk will do
  - Share experience / flow
  - Give tips / keywords

Hardware description



- “There is no alternative” (VHDL / System-Verilog)
  - Scala based : SpinalHDL, Chisel
  - Python based : Migen, Amaranth
  - ...

It can be pretty / explicit

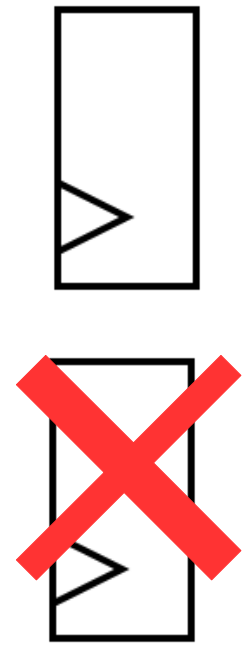



```
class Timer extends Component {
  val increment = in Bool()
  val counter = Reg(UInt(8 bits)) init(0)
  val full = counter === 255

  when(increment){
    counter := counter + 1
  }
}
```

Hardware description API

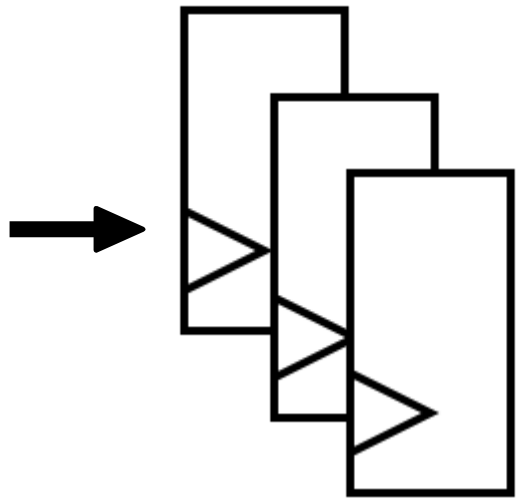
```
if(featureEnabled){
  Reg(UInt(8 bits))
}
```



- Control flow : **if** / for
- Data structures : dynamic array / hash map / hash set
- Lambda function : reduce / fold / map / filter / ...
- OOP : class / software interface
- See https://spinalhdl.github.io/NaxRiscv-Rtd/main/NaxRiscv/abstraction/index.html

Hardware description API

```
for(i <- 0 to 2){
  Reg(UInt(8 bits))
}
```



- Control flow : if / **for**
- Data structures : dynamic array / hash map / hash set
- Lambda function : reduce / fold / map / filter / ...
- OOP : class / software interface
- See https://spinalhdl.github.io/NaxRiscv-Rtd/main/NaxRiscv/abstraction/index.html

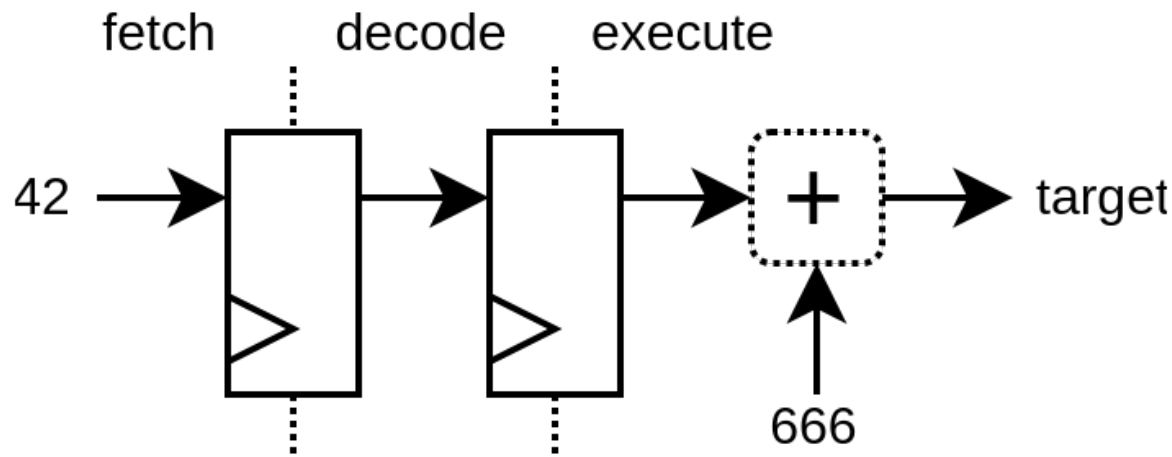
Hardware description API

```
val flushes = ArrayBuffer[Flush]()
flushes += new Flush()
flushes += new Flush()
flushes += new Flush()
```

Index	Value
0	hardware
1	hardware
2	hardware

- Data structures : **dynamic array** / hash map / hash set
- Lambda function : reduce / fold / map / filter / ...
- OOP : class / software interface
- See https://spinalhdl.github.io/NaxRiscv-Rtd/main/NaxRiscv/abstraction/index.html

```
val PC = Hardtype(UInt(32 bits))
val instruction = Hardtype(UInt(32 bits))
val fetch, decode, execute = HashMap[Hardtype, Hardware]()
...
fetch(PC) = U(42)
...
val target = execute(PC) + U(666)
...
autoPipeline(fetch, decode, execute)
```

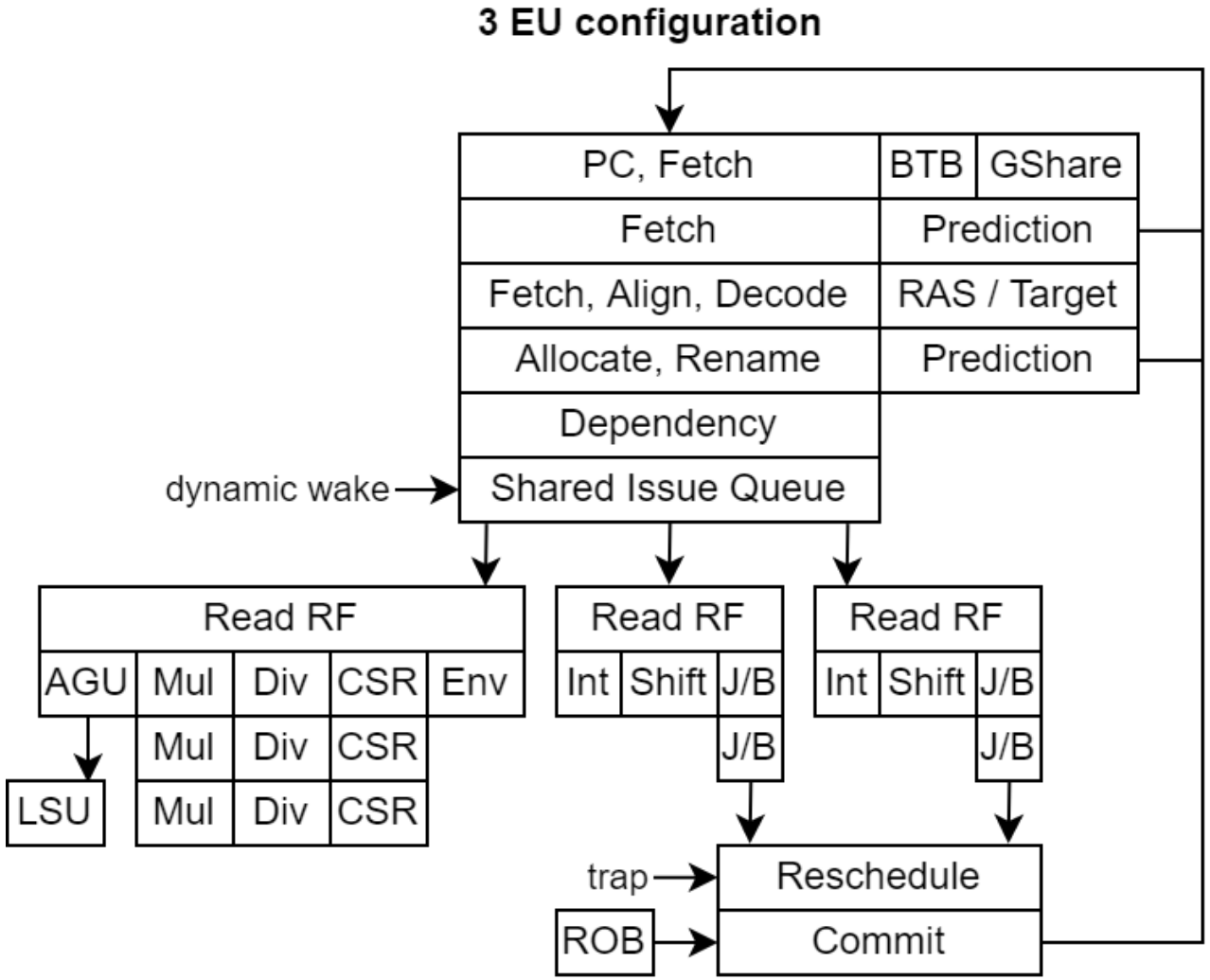


- Data structures : dynamic array / **hash map** / hash set
- Lambda function : reduce / fold / map / filter / ...
- OOP : class / software interface
- See https://spinalhdl.github.io/NaxRiscv-Rtd/main/NaxRiscv/abstraction/index.html

ISA (instruction set architecture)

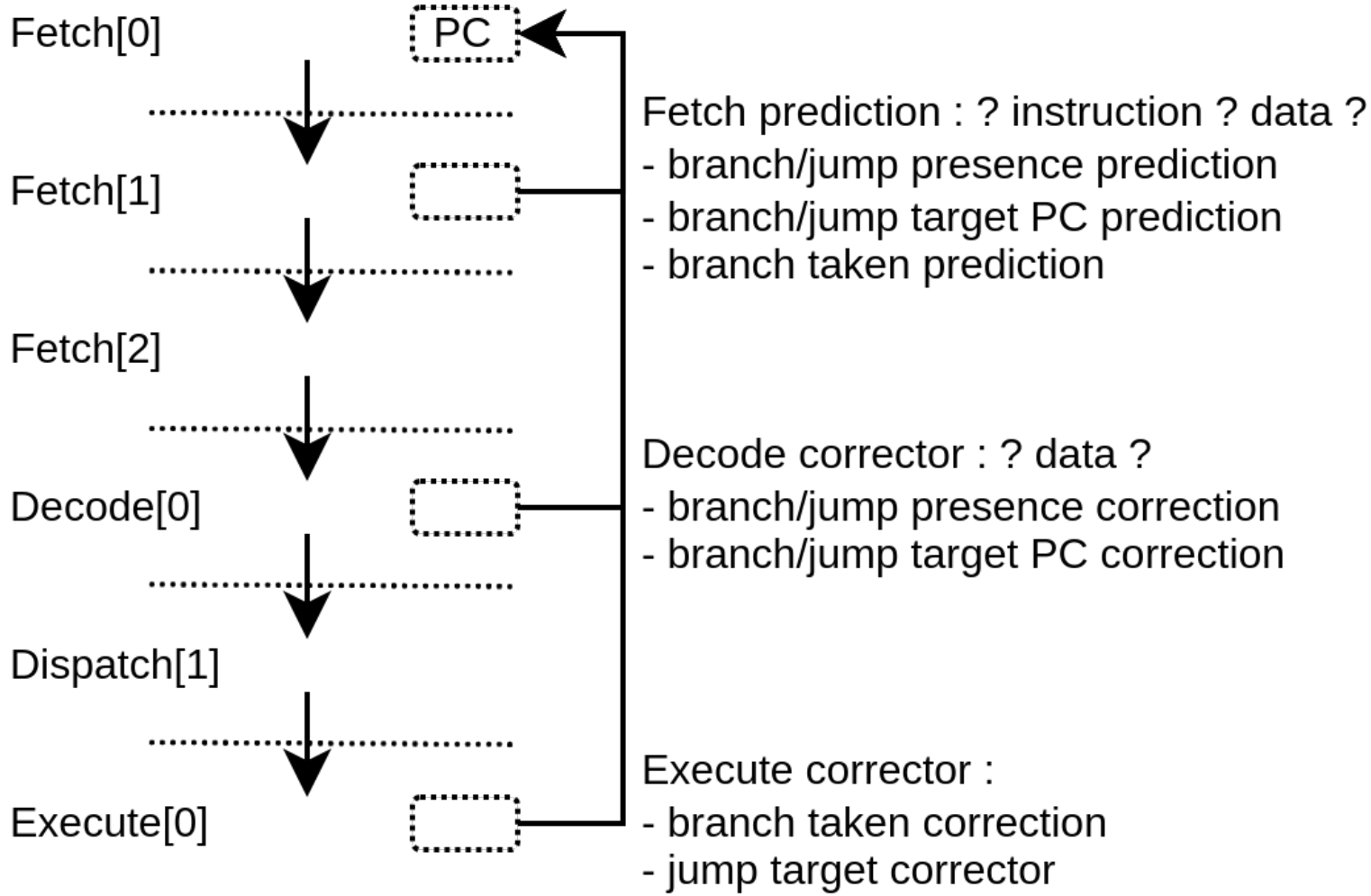
- A few open / free ISA (OpenRISC, , …)
- RISC-V
  - https://riscv.org/technical/specifications/
  - Secretly being embedded in ASICs
  - Mostly bloat free, from a FPGA perspective)
  - GCC / LLVM / Qemu
  - Linux / Debian port
  - …

CPU design (NaxRiscv)



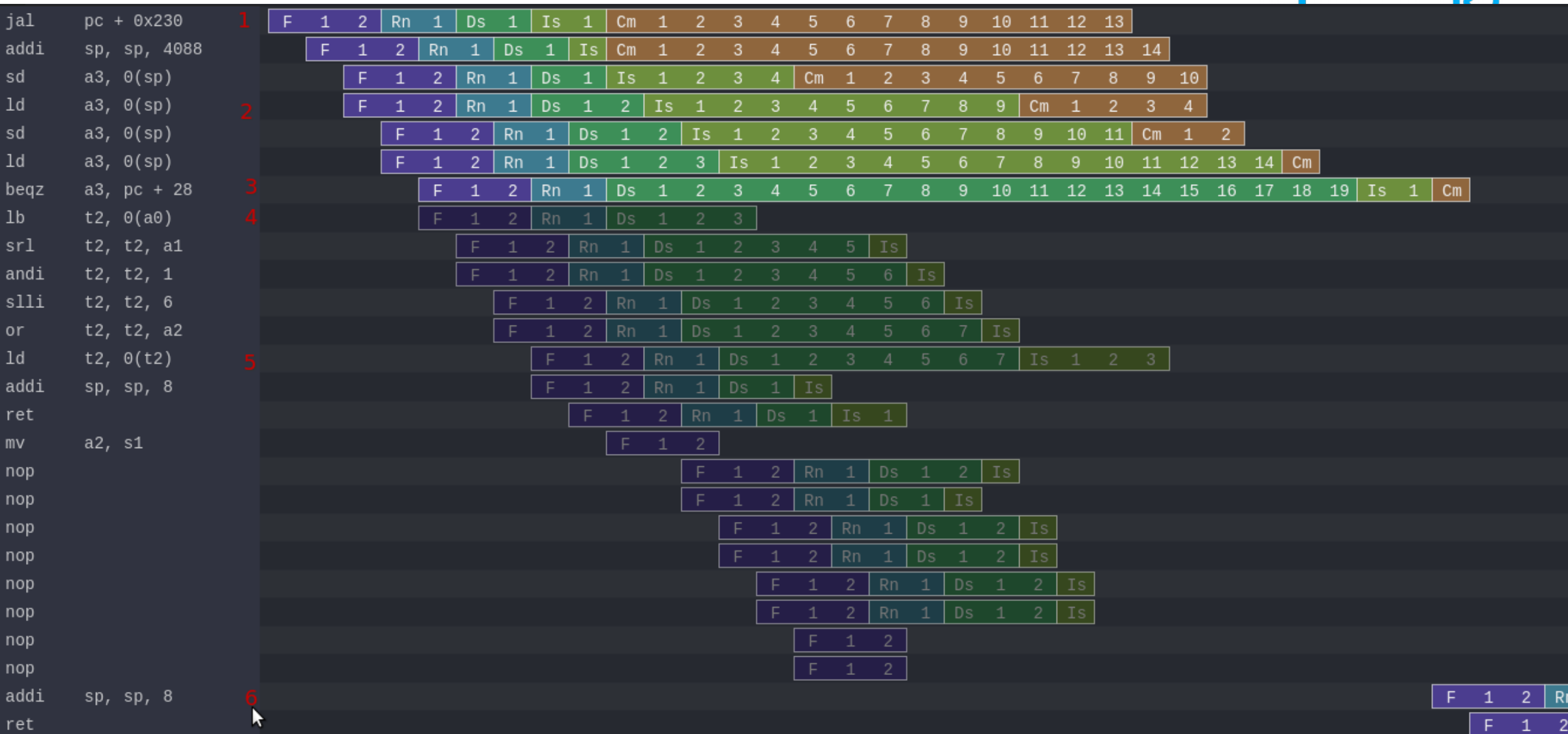
https://github.com/SpinalHDL/NaxRiscv

“Branch prediction”

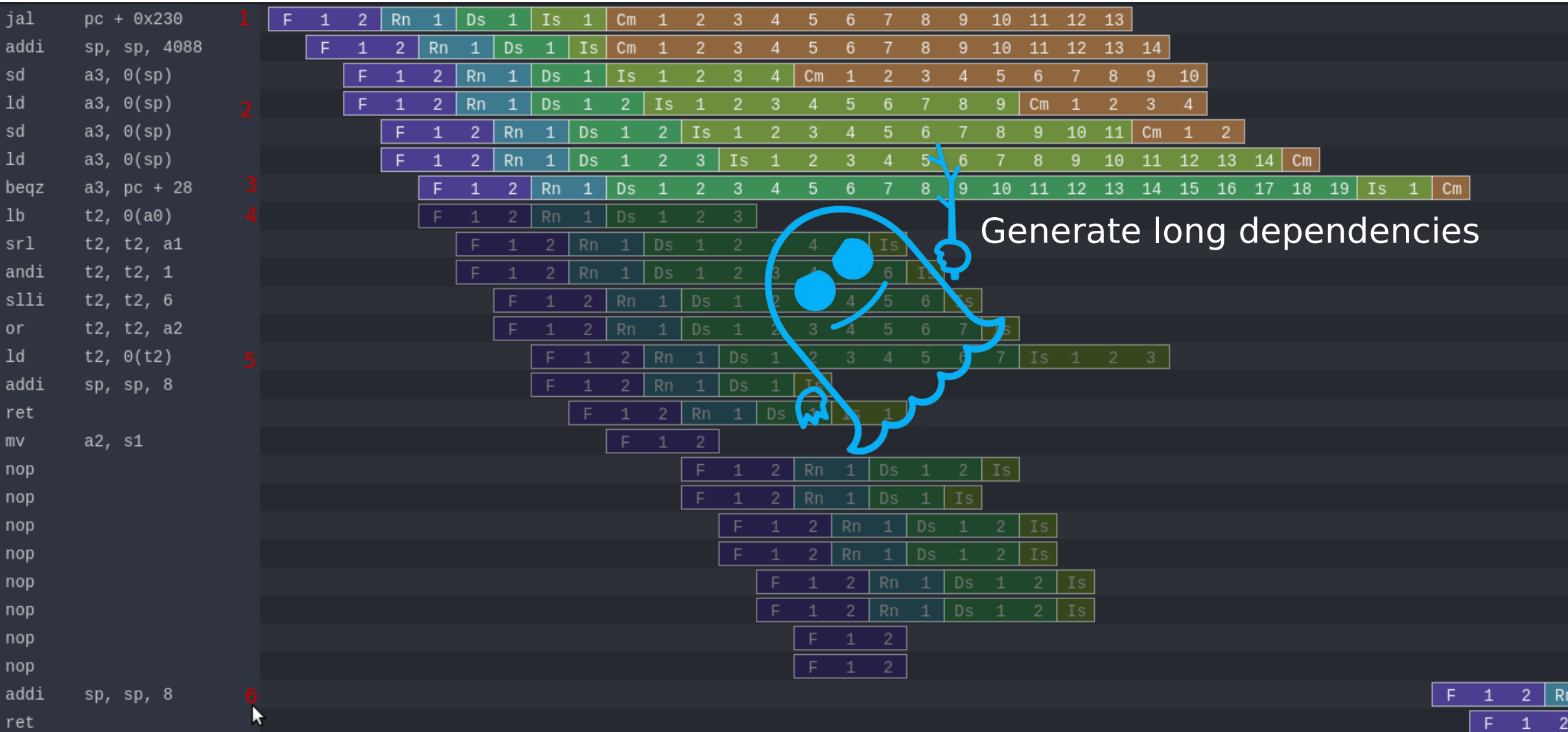


https://spinalhdl.github.io/NaxRiscv-Rtd/main/NaxRiscv/misc/index.html#security

Secret leak (Konata trace)

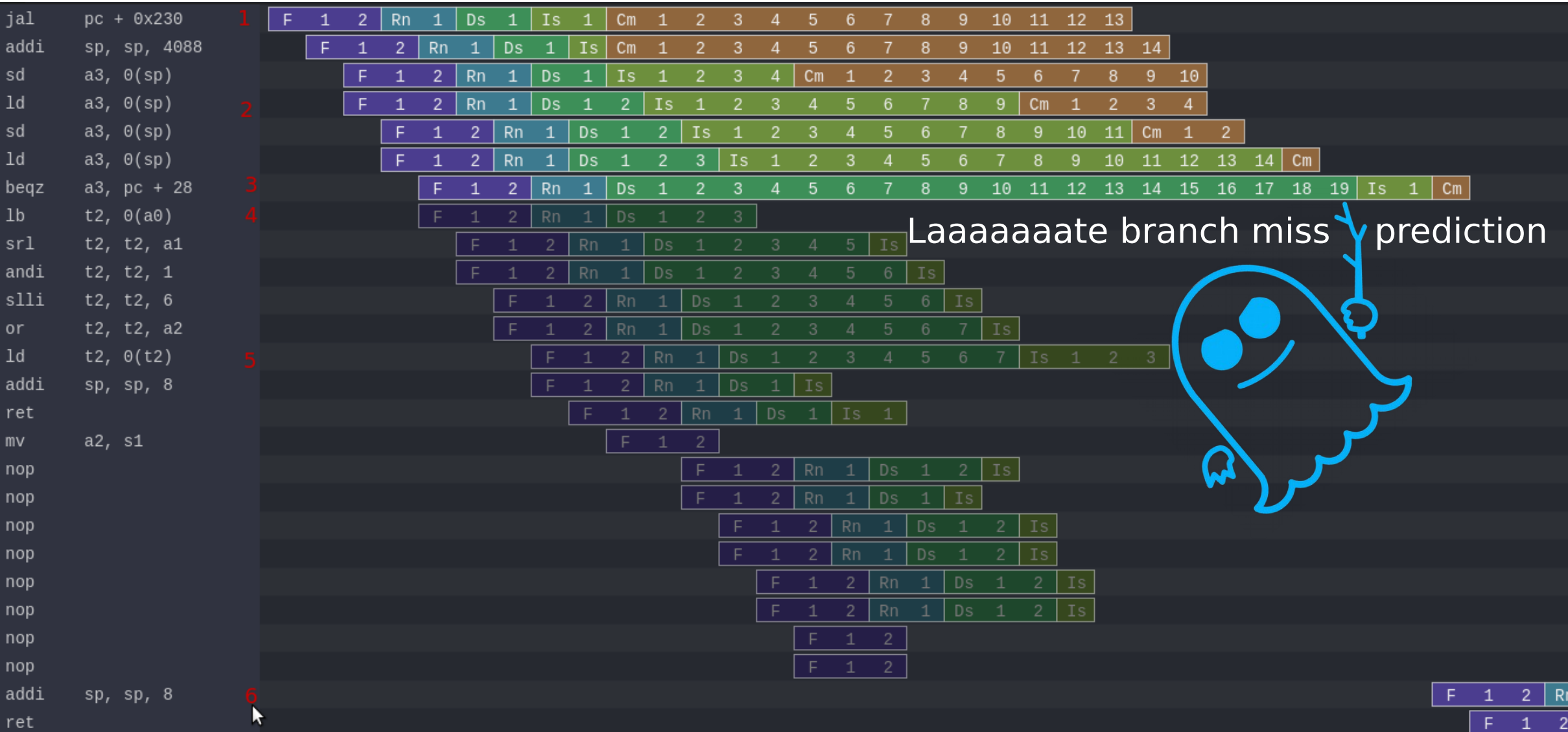


Secret leak

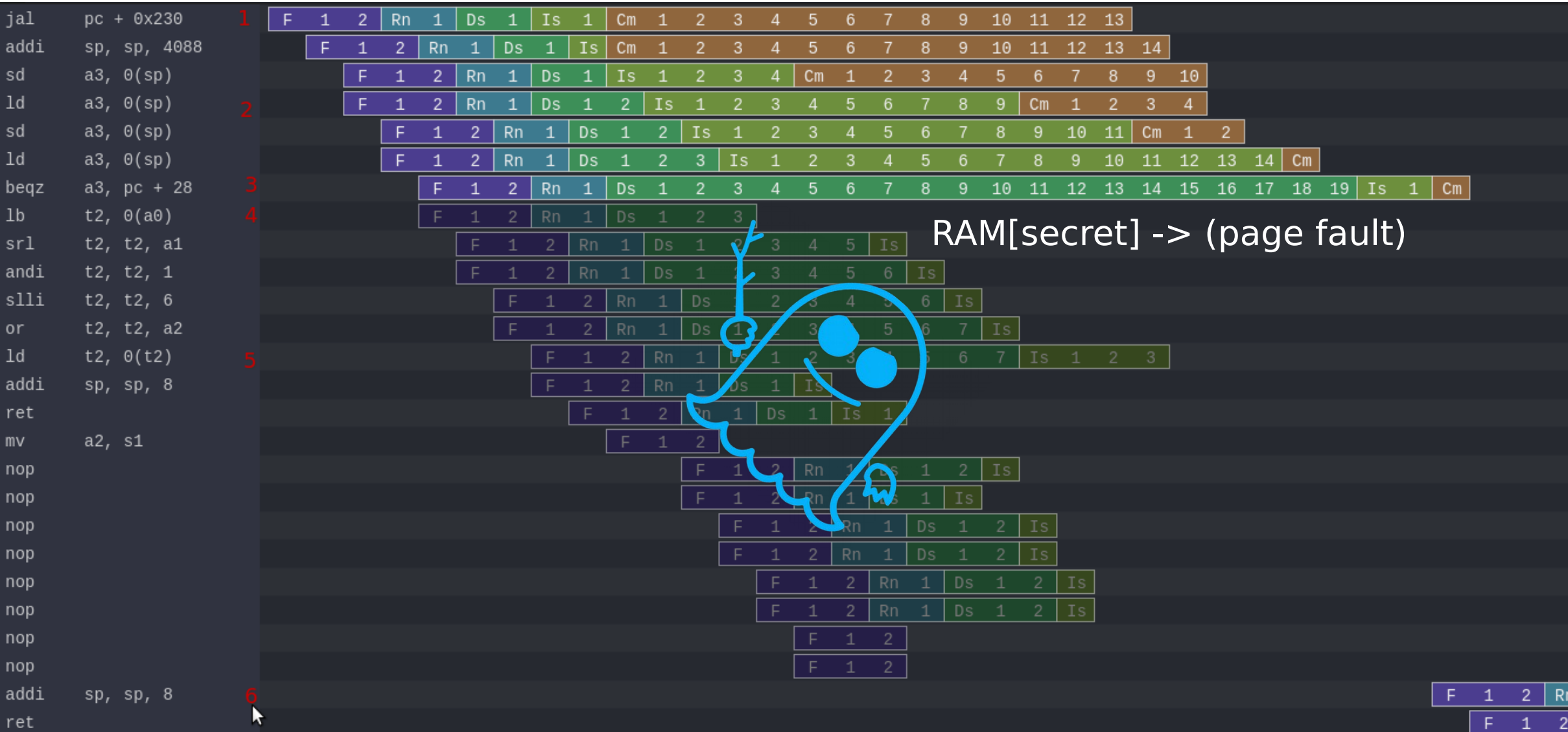

jal pc + 0x230
addi sp, sp, 4088
sd a3, 0(sp)
ld a3, 0(sp)
sd a3, 0(sp)
ld a3, 0(sp)
beqz a3, pc + 28
lb t2, 0(a0)
srl t2, t2, a1
andi t2, t2, 1
slli t2, t2, 6
or t2, t2, a2
ld t2, 0(t2)
addi sp, sp, 8
ret
mv a2, s1
nop
nop
nop
nop
nop
nop
nop
nop
addi sp, sp, 8
ret
1
2
3
4
5
6
Generate long dependencies

Secret leak

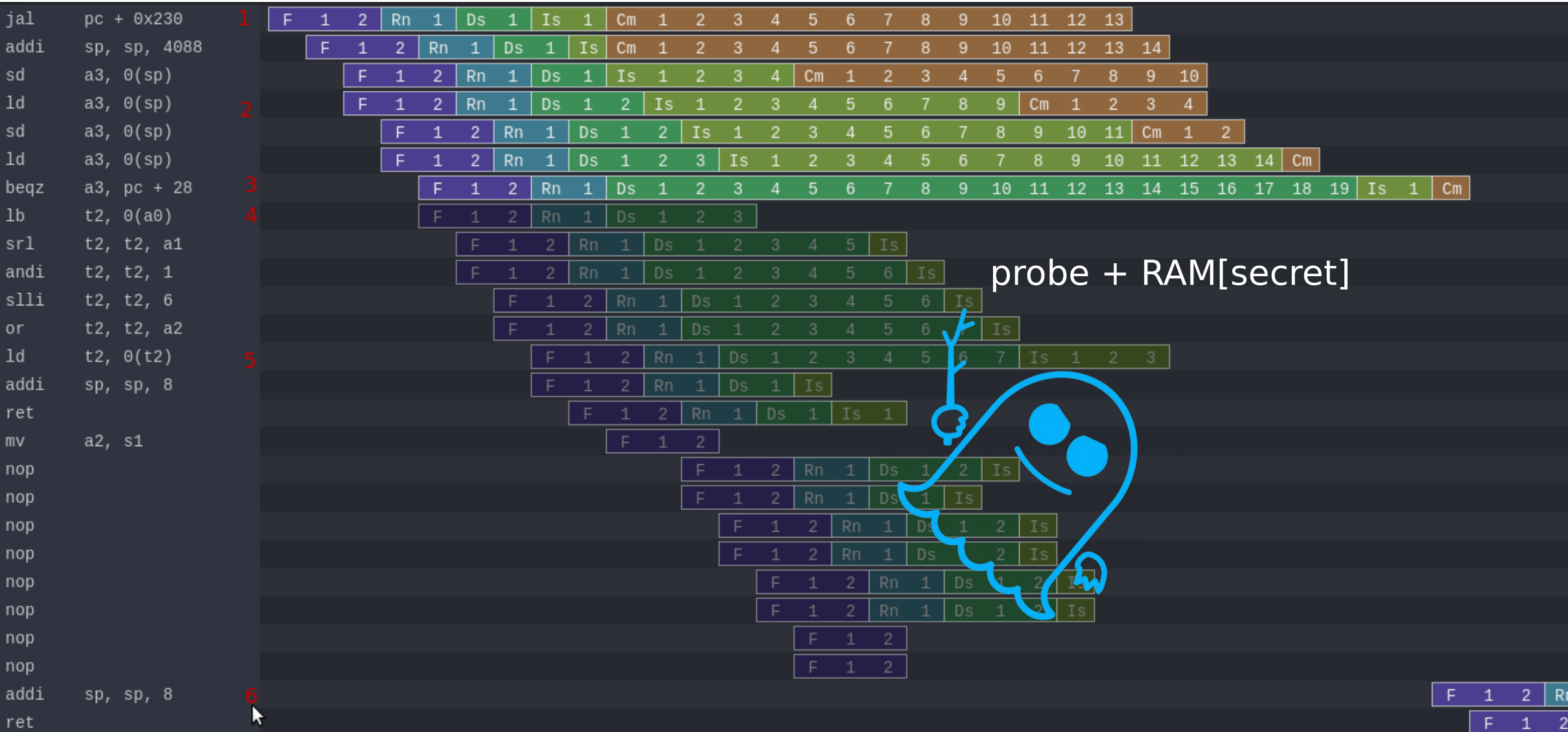
```
jal     pc + 0x230          1
addi    sp, sp, 4088
sd      a3, 0(sp)
ld      a3, 0(sp)           2
sd      a3, 0(sp)
ld      a3, 0(sp)
beqz    a3, pc + 28         3
lb      t2, 0(a0)           4
srl     t2, t2, a1
andi    t2, t2, 1
slli    t2, t2, 6
or      t2, t2, a2
ld      t2, 0(t2)           5
addi    sp, sp, 8
ret
mv      a2, s1
nop
nop
nop
nop
nop
nop
nop
nop
addi    sp, sp, 8           6
ret
```


F 1 2 Rn 1 Ds 1 Is 1 Cm 1 2 3 4 5 6 7 8 9 10 11 12 13
F 1 2 Rn 1 Ds 1 Is Cm 1 2 3 4 5 6 7 8 9 10 11 12 13 14
F 1 2 Rn 1 Ds 1 Is 1 2 3 4 Cm 1 2 3 4 5 6 7 8 9 10
F 1 2 Rn 1 Ds 1 2 Is 1 2 3 4 5 6 7 8 9 Cm 1 2 3 4
F 1 2 Rn 1 Ds 1 2 Is 1 2 3 4 5 6 7 8 9 10 11 Cm 1 2
F 1 2 Rn 1 Ds 1 2 3 Is 1 2 3 4 5 6 7 8 9 10 11 12 13 14 Cm
F 1 2 Rn 1 Ds 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 Is 1 Cm
F 1 2 Rn 1 Ds 1 2 3
F 1 2 Rn 1 Ds 1 2 3 4 5 Is
F 1 2 Rn 1 Ds 1 2 3 4 5 6 Is
F 1 2 Rn 1 Ds 1 2 3 4 5 6 Is
F 1 2 Rn 1 Ds 1 2 3 4 5 6 7 Is
F 1 2 Rn 1 Ds 1 2 3 4 5 6 7 Is 1 2 3
F 1 2 Rn 1 Ds 1 Is
F 1 2 Rn 1 Ds 1 Is 1
F 1 2
F 1 2 Rn 1 Ds 1 2 Is
F 1 2 Rn 1 Ds 1 Is
F 1 2 Rn 1 Ds 1 2 Is
F 1 2 Rn 1 Ds 1 2 Is
F 1 2 Rn 1 Ds 1 2 Is
F 1 2 Rn 1 Ds 1 2 Is
F 1 2
F 1 2
Laaaaaaate branch miss prediction
F 1 2 R
F 1 2
```
```

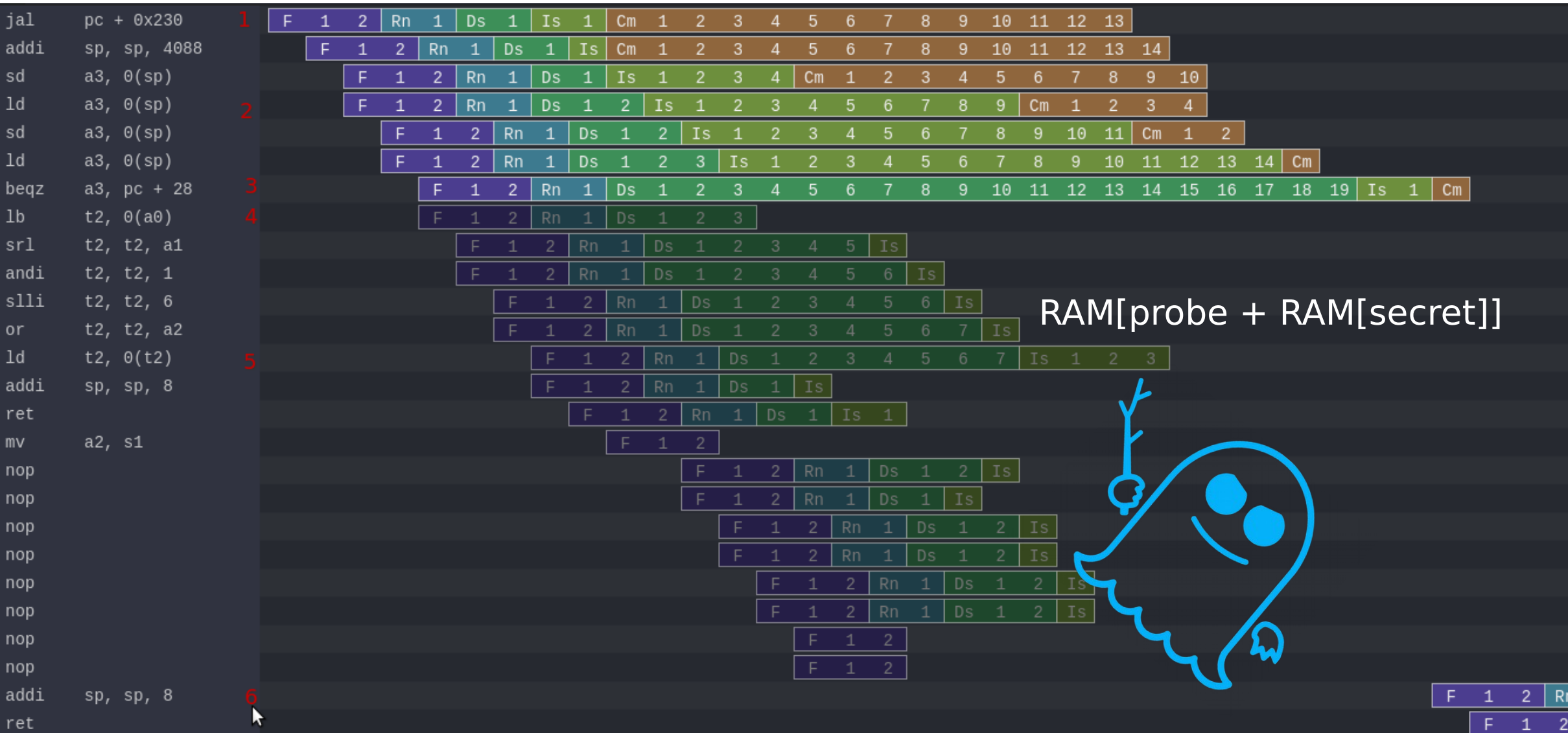
Secret leak


jal pc + 0x230
addi sp, sp, 4088
sd a3, 0(sp)
ld a3, 0(sp)
sd a3, 0(sp)
ld a3, 0(sp)
beqz a3, pc + 28
lb t2, 0(a0)
srl t2, t2, a1
andi t2, t2, 1
slli t2, t2, 6
or t2, t2, a2
ld t2, 0(t2)
addi sp, sp, 8
ret
mv a2, s1
nop
nop
nop
nop
nop
nop
nop
nop
addi sp, sp, 8
ret
1
2
3
4
5
6
RAM[secret] -> (page fault)

Secret leak

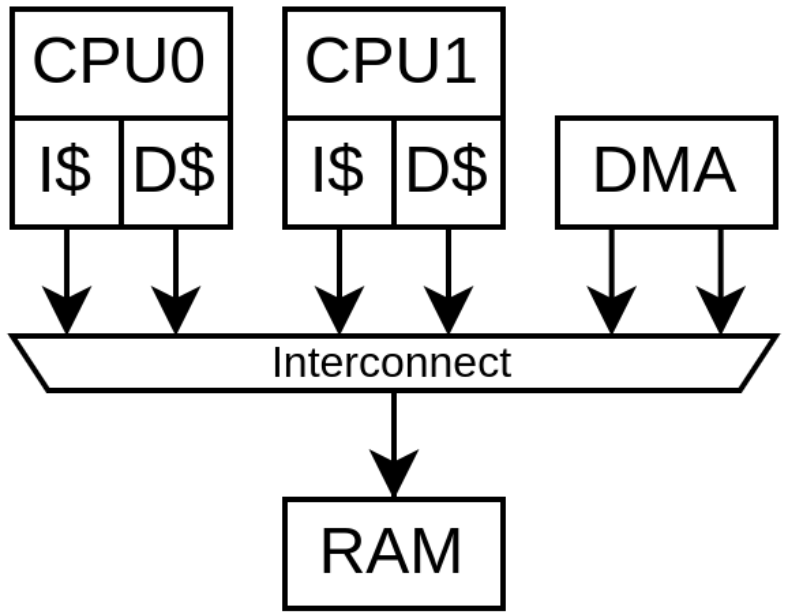

jal pc + 0x230
addi sp, sp, 4088
sd a3, 0(sp)
ld a3, 0(sp)
sd a3, 0(sp)
ld a3, 0(sp)
beqz a3, pc + 28
lb t2, 0(a0)
srl t2, t2, a1
andi t2, t2, 1
slli t2, t2, 6
or t2, t2, a2
ld t2, 0(t2)
addi sp, sp, 8
ret
mv a2, s1
nop
nop
nop
nop
nop
nop
nop
nop
addi sp, sp, 8
ret
1
2
3
4
5
6
probe + RAM[secret]

Secret leak

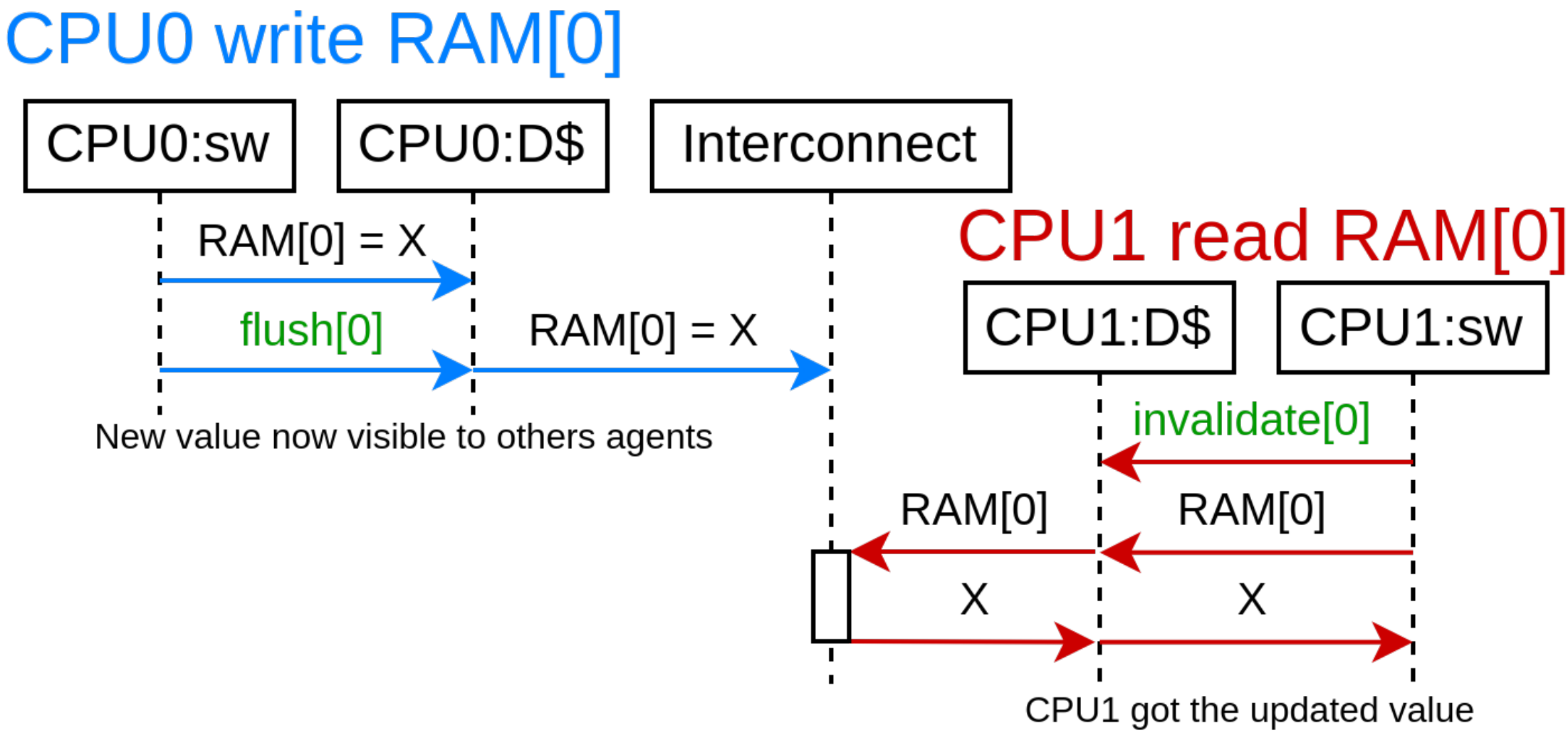

jal pc + 0x230
addi sp, sp, 4088
sd a3, 0(sp)
ld a3, 0(sp)
sd a3, 0(sp)
ld a3, 0(sp)
beqz a3, pc + 28
lb t2, 0(a0)
srl t2, t2, a1
andi t2, t2, 1
slli t2, t2, 6
or t2, t2, a2
ld t2, 0(t2)
addi sp, sp, 8
ret
mv a2, s1
nop
nop
nop
nop
nop
nop
nop
nop
addi sp, sp, 8
ret
1
2
3
4
5
6
RAM[probe + RAM[secret]]

Multi core requirements (RISC-V linux)

- Hardware cache coherency
- Inter-CPU software interrupts

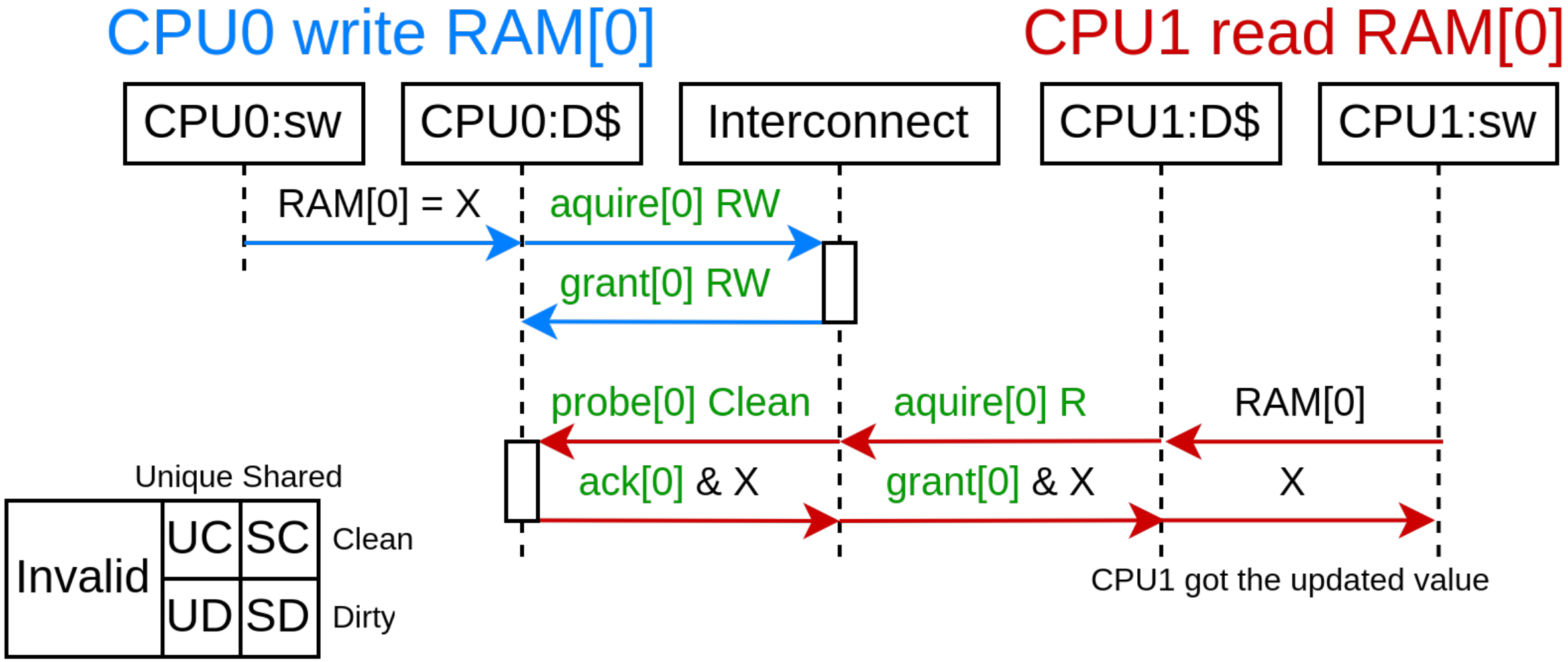


Software cache coherency



Hardware cache coherency

with memory block copy and permissions



Hardware cache coherency

And its side effects

```
volatile int array[2];

void thread_0() {
  while(1) {
    array[0] += 1;
  }
}

void thread_1() {
  while(1) {
    array[1] += 1;
  }
}
```

```
Same memory block (64 bytes): 18486 ps/iteration
Different memory blocks     :   902 ps/iteration

(on a 2 years old desktop CPU)
```

Cache coherency

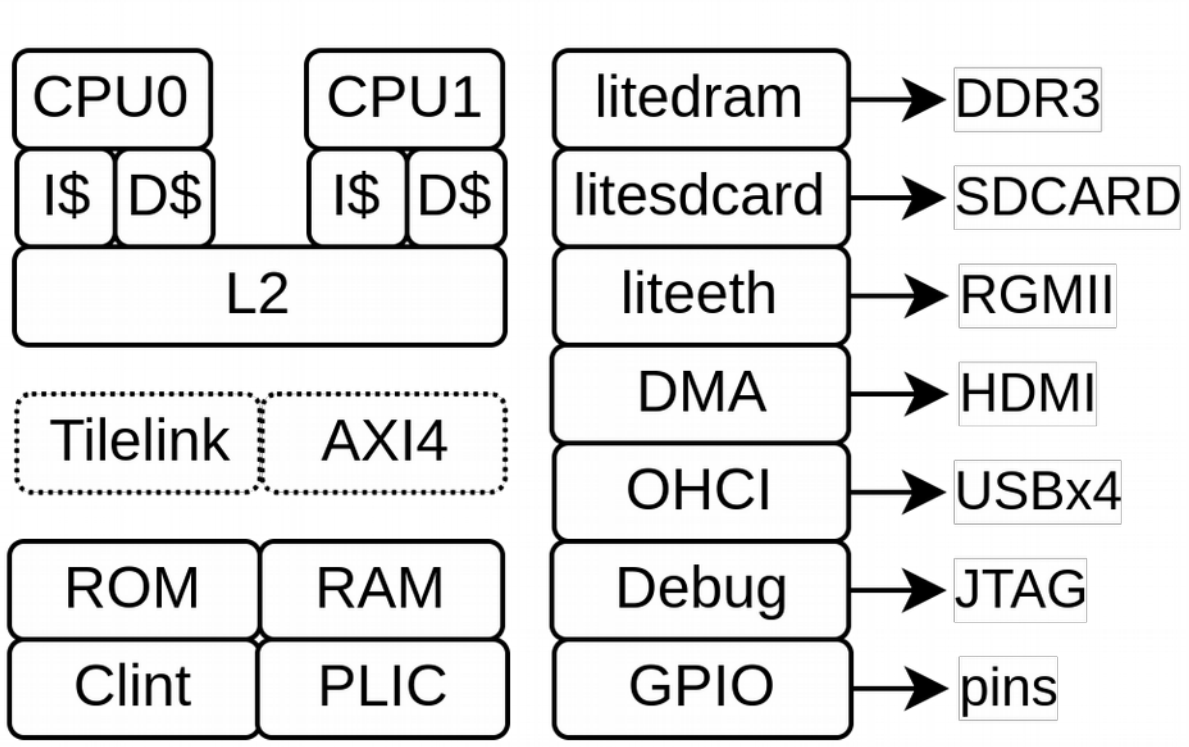
- Open specification : Tilelink
- Multiple open source implementations, ex :
  - https://github.com/SpinalHDL/SpinalHDL/tree/dev/lib/src/main/scala/spinal/lib/bus/tilelink
- Coherent L2 design
  - https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Libraries/Bus/tilelink/tilelink_fabric.html
  - Exposes a lot of performance tricks
    - 1 pending transaction per 64 bytes block
    - 64 bytes per burst max
    - Keep bursts aligned
    - Alignment Aliasing

Memory / Peripherals / Deployment

- LiteX
  - Open-source SoC framework (in Python)
  - Integrate many peripherals
    - DDRx controller
    - SDCARD
    - Ethernet, USB host
    - SATA, PCIe, ...

Memory / Peripherals / Deployment

- python3 -m litex_boards.targets.digilent_nexys_video --cpu-type=naxriscv --bus-standard axi-lite --with-video-framebuffer --with-coherent-dma --with-sdcard --with-ethernet --xlen=64 --scala-args='rvc=true,rvf=true,rvd=true,alu-count=2,decode-count=2' --with-jtag-tap --sys-clk-freq 100000000 --cpu-count 2 --build –load
- + stuff for USB / JTAG



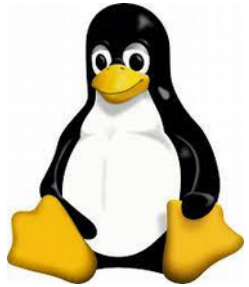
USB host support

- USB host support =>
  - Keyboard / mouse / audio / flash drive / sdcard / uart / ethernet / bluetooth
- Open specification :
  - Open Host Controller Interface Specification for USB (OHCI)
  - USB 12 Mb/s, up to 15 ports, 2 FPGA GPIO per port
- Open implementation :
  - https://spinalhdl.github.io/SpinalDoc-RTD/master/SpinalHDL/Libraries/Com/usb_ohci.html
  - https://github.com/SpinalHDL/SpinalHDL/blob/dev/lib/src/main/scala/spinal/lib/com/usb/ohci/UsbOhci.scala
  - https://github.com/litex-hub/pythondata-misc-usb_ohci/tree/master/pythondata_misc_usb_ohci/verilog

Synthesis / Place / Route

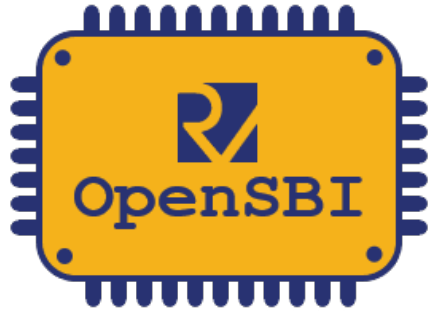
- Some open source tools for
  - Xilinx 7-Series
  - Lattice ice40 / ecp5
  - ...
- Synthesis : Yosys
- Place and route : Nextpnr
- ...

Running Linux

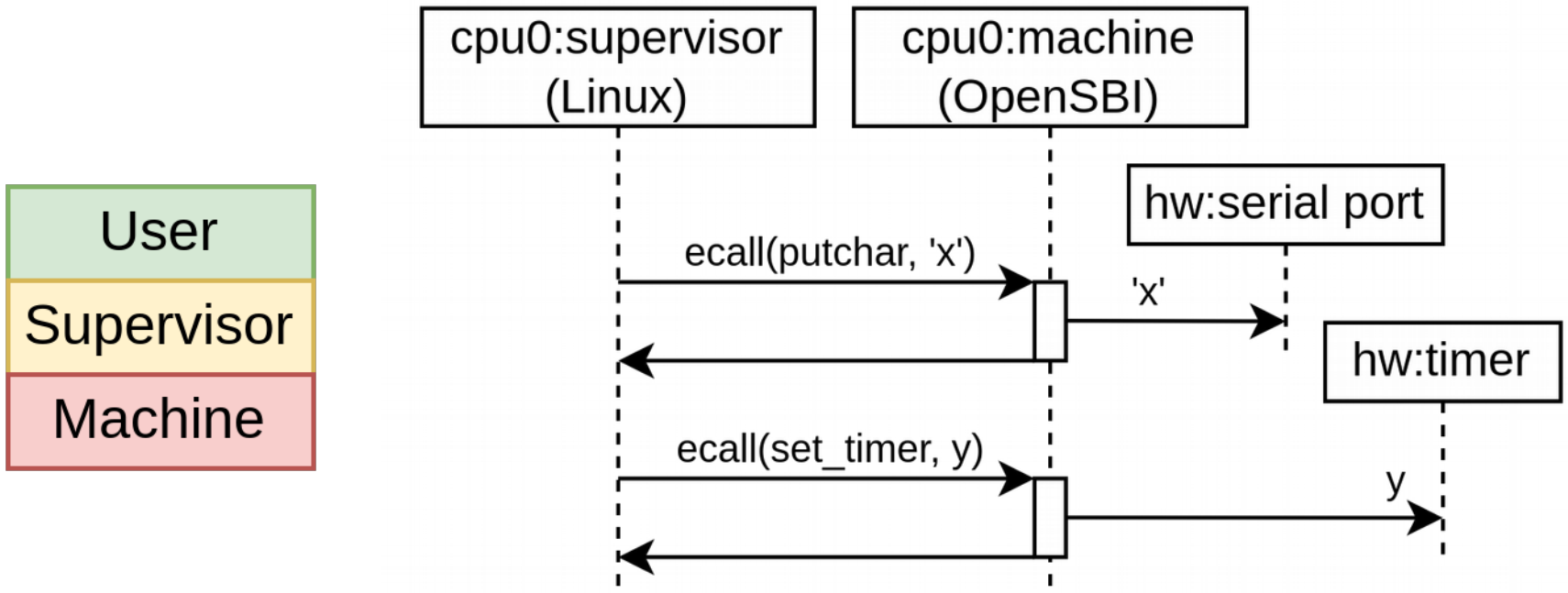


- Requirements on RISC-V
  - Hardware cache coherency (between CPUs + DMA)
  - Machine/Supervisor/User modes (riscv-privileged spec)
  - MMU / Atomic
  - SBI (Supervisor Binary Interface, kinda like BIOS)

SBI (Supervisor Binary Interface)

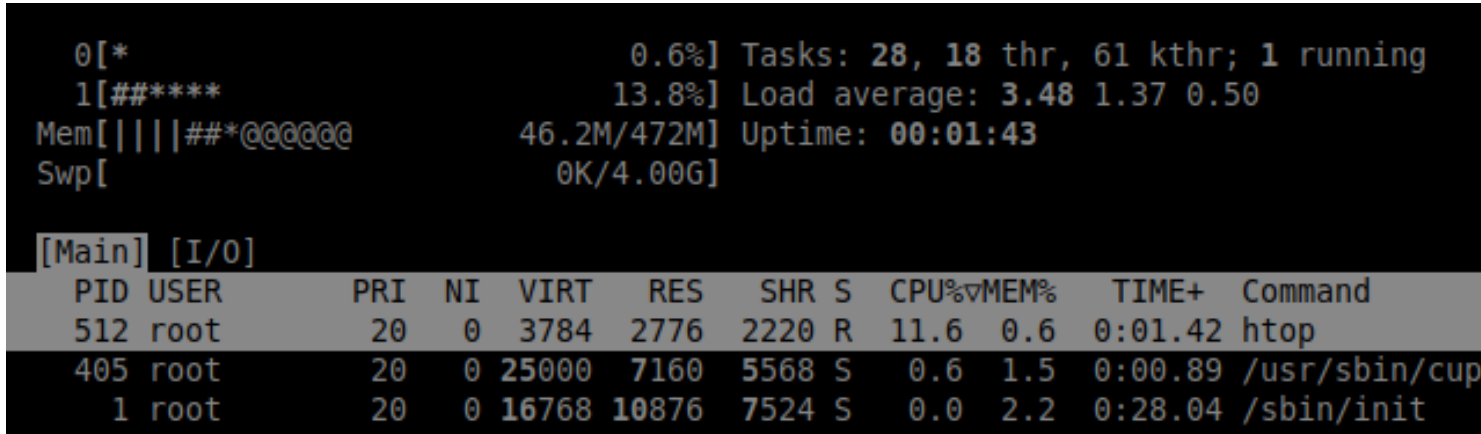


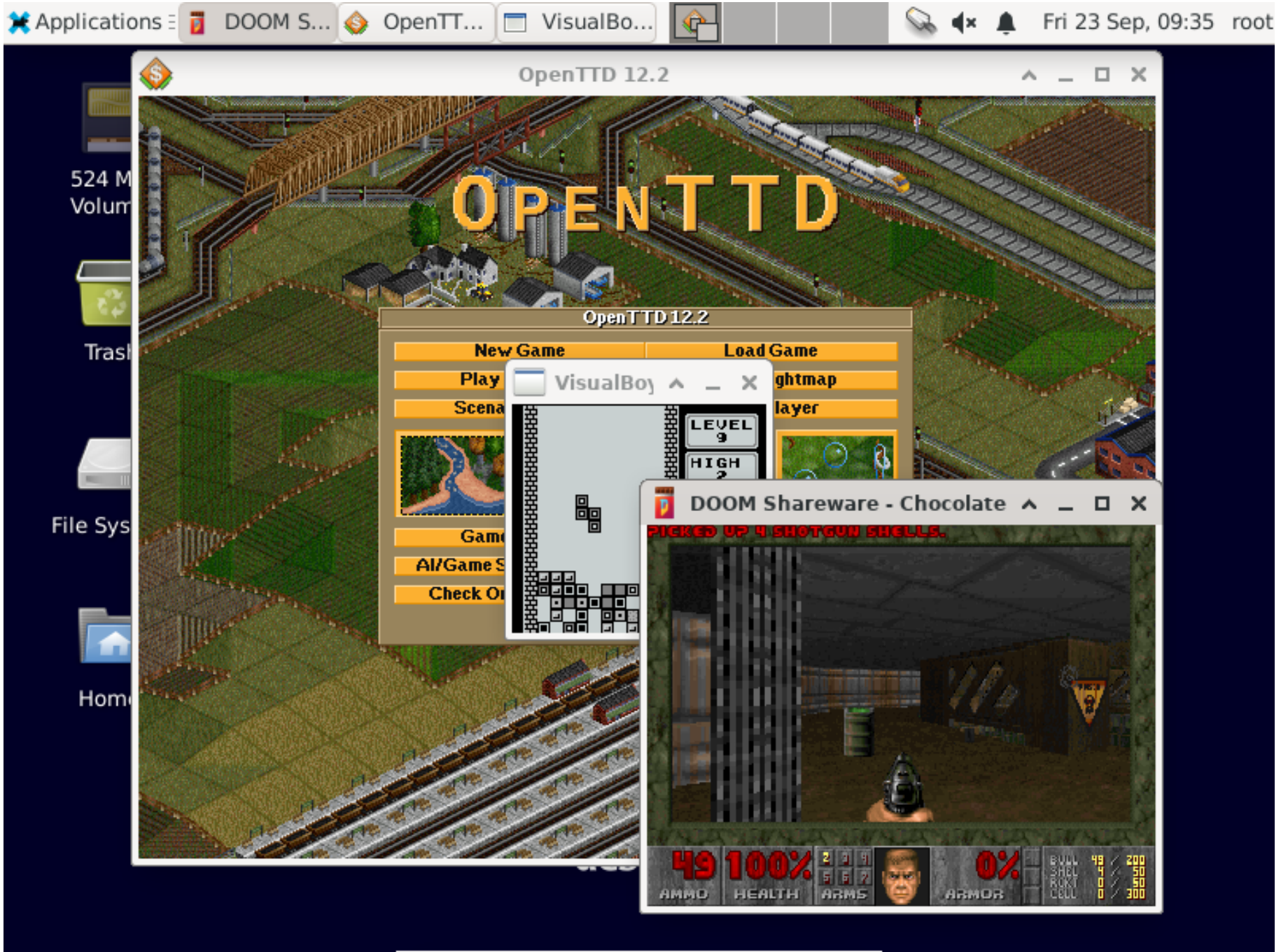
- https://github.com/riscv-non-isa/riscv-sbi-doc
- https://github.com/riscv-software-src/opensbi



Running Debian

- Requirements on RISC-V
  - RV64IMAFDC
  - Linux kernel
  - Some storage (ex: SDCARD)
- Doc : https://github.com/SpinalHDL/NaxSoftware/tree/main/debian_litex
- Video : https://x.com/dolu1990/status/1712848731349889108




Applications
DOOM S...
OpenTT...
VisualBo...
Fri 23 Sep, 09:35
root
OpenTTD 12.2
OPENTTD
New Game
Load Game
Play
Scena
ghtmap
layer
Game
AI/Game S
Check Or
VisualBoy
LEVEL 9
HIGH 2
DOOM Shareware - Chocolate
PICKED UP 4 SHOTGUN SHELLS.
49
AMMO
100%
HEALTH
ARMS
0%
ARMOR
524 M
Volum
Trash
File Sys
Hom

Verification / Debugging

- May create rituals
- May exhaust your sanity
- May never end
- May crush your soul

Verification / Debugging

- May create rituals
- May exhaust your sanity
- May never end
- May crush your soul

```
[1377614.579833] rcu: INFO: rcu_sched self-detected stall on CPU
[1377614.579845] rcu: 18-....: (2099 ticks this GP) idle=54e/1/0x4000000000000002
[1377614.579895] (t=2100 jiffies g=155867385 q=20879)
[1377614.579898] Task dump for CPU 18:
[1377614.579899] CPU 1/KVM R running task 0 1030947 256019 0x06000004
[1377614.579902] Call Trace:
[1377614.579912] ([<0000001f1f4b4f52>] show_stack+0x7a/0xc0)
[1377614.579918] [<0000001f1ec8e96c>] sched_show_task.part.0+0xdc/0x100
[1377614.579919] [<0000001f1f4b7248>] rcu_dump_cpu_stacks+0xc0/0x100
[1377614.579924] [<0000001f1ecdd10c>] rcu_sched_clock_irq+0x75c/0x980
[1377614.579926] [<0000001f1eceb26c>] update_process_times+0x3c/0x80
[1377614.579931] [<0000001f1ecfcfea>] tick_sched_handle.isra.0+0x4a/0x70
[1377614.579932] [<0000001f1ecfd28e>] tick_sched_timer+0x5e/0xc0
[1377614.579933] [<0000001f1ecec294>] __hrtimer_run_queues+0x114/0x2f0
[1377614.579935] [<0000001f1ececfdc>] hrtimer_interrupt+0x12c/0x2a0
[1377614.579938] [<0000001f1ebecb6a>] do_IRQ+0xaa/0xb0
[1377614.579942] [<0000001f1f4c6d08>] ext_int_handler+0x130/0x134
[1377614.579945] [<0000001f1ec0af10>] ptep_zap_key+0x40/0x60
```

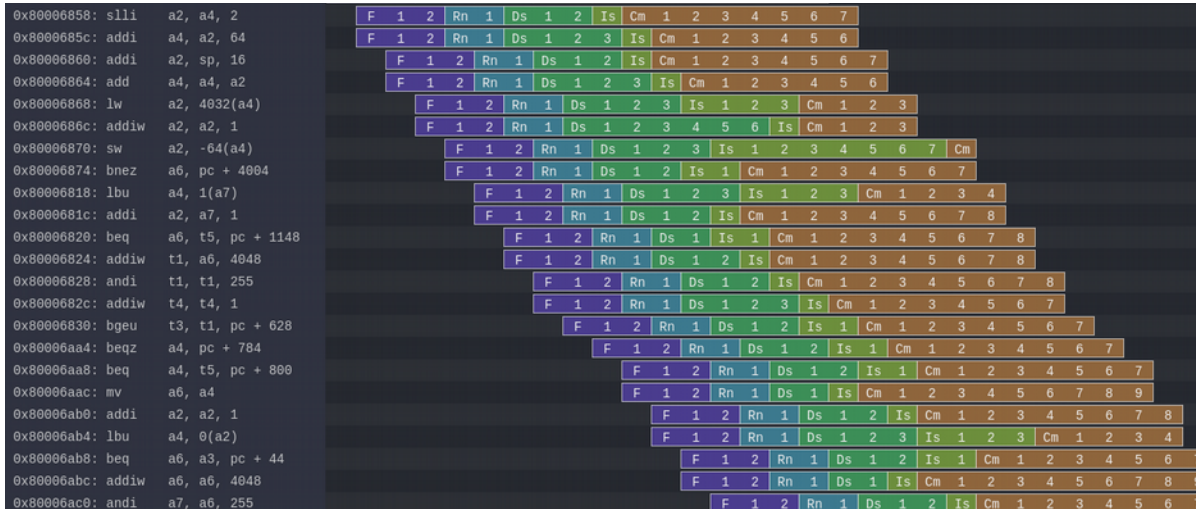
Hello darkness my old friend

Debugging – example (a real one)

- Dual core Debian freeze (RCU error) after ~1h (360’000’000’000 cycles)
  - Impossible to reproduced in simulation (~40 days of runtime @ 100 Khz)
- Hope you get clues
  - Core was still alive on JTAG => OpenOCD
  - PC was in linux “_raw_spin_lock”
  - Forcing it to unlock => Everything back on track
- What it was :
  - CPU D$ doing bad memory coherency
  - (releasing data permissions once again after being probed out)

Simulation

- A few open source simulation tools :
  - GHDL, IVerilog, Verilator, ...
- Based on Verilator (~150-200 Khz on a 4 years old CPU)
- Lock-step checks against Spike / RVLS
- Konata traces



0xAA55

- I hope you had a good time
- If you are looking for open-source project funding :
  - NLnet Foundation (they helped me a lot)