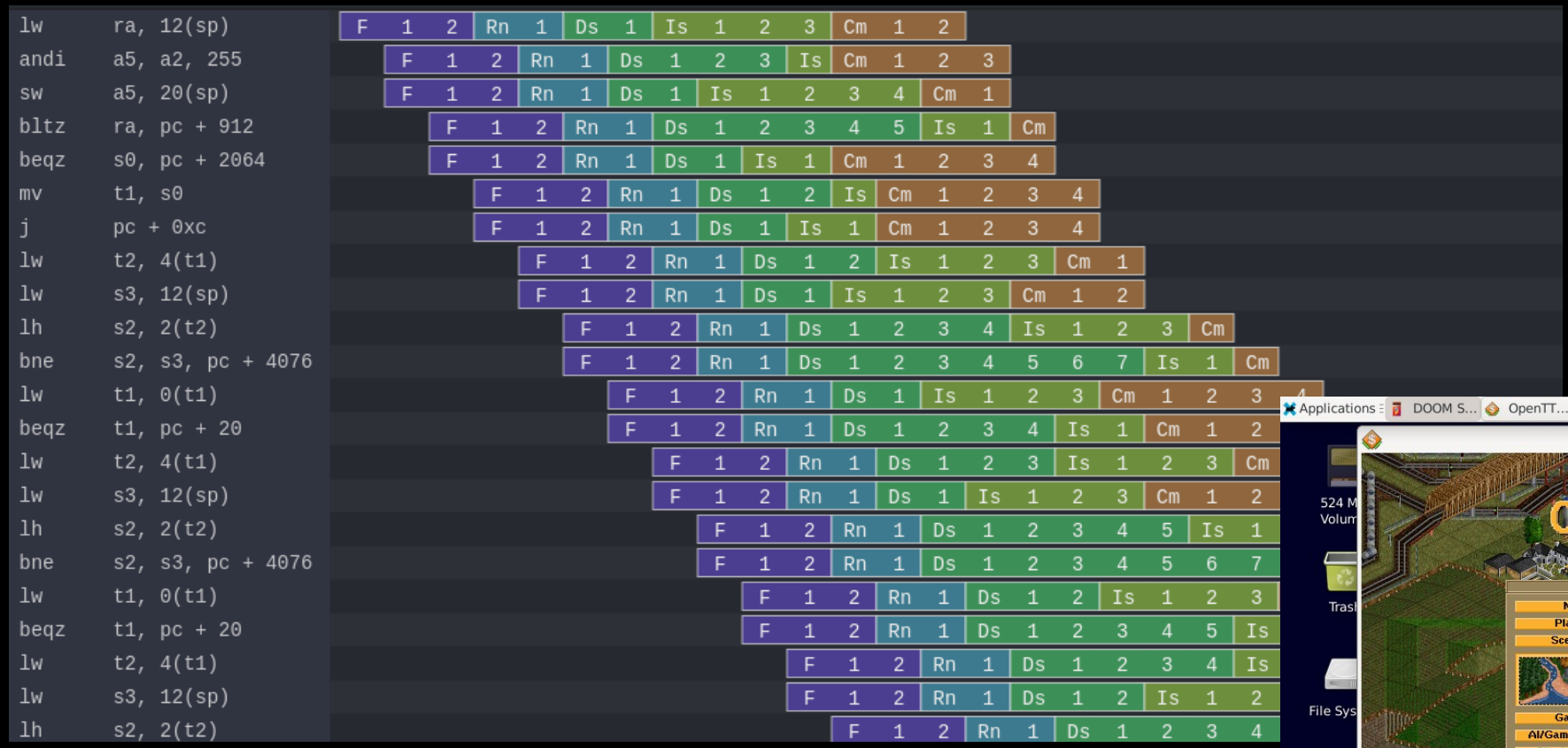
Open CPU / SoC design, all the way up to Debian



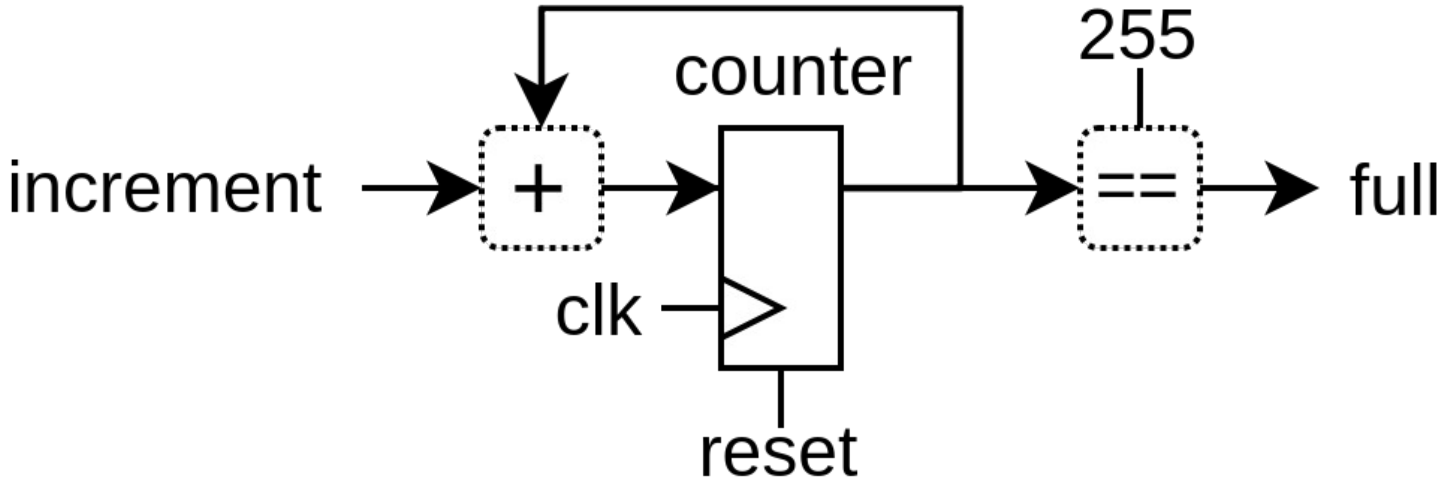
Background / whoami

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

Introduction

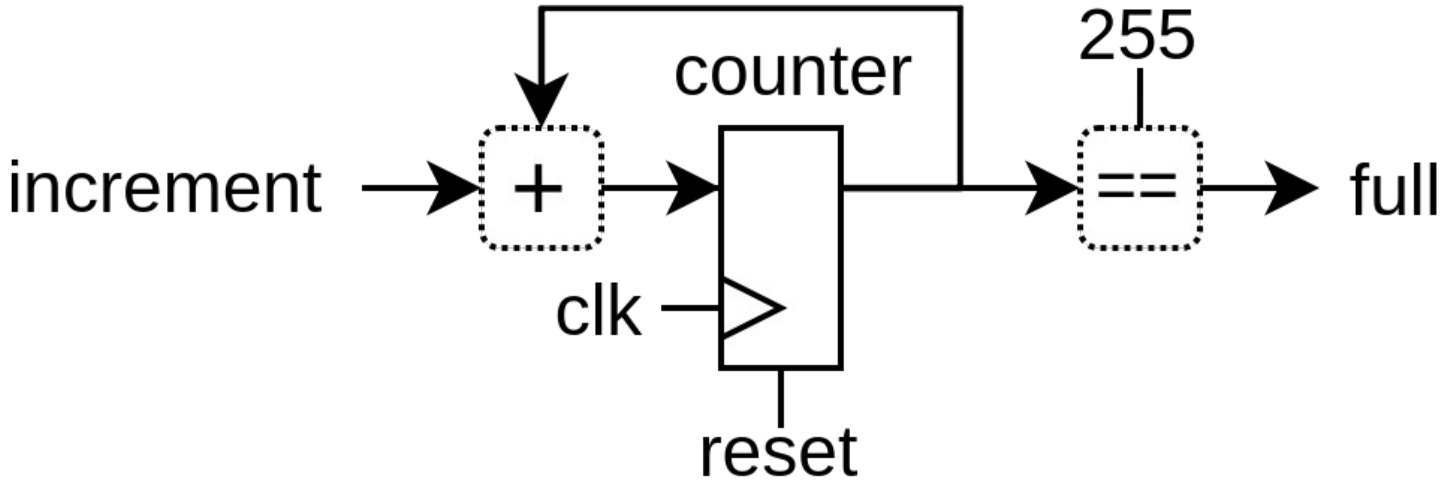
- This talk will be based on NaxRiscv
  - Opensource https://github.com/SpinalHDL/NaxRiscv
    - RISC-V softcore
    - out of order, multi-issue, multi core, memory coherent
- What the talk will do
  - Share experience
  - Give tips / keywords

Digital Hardware description



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

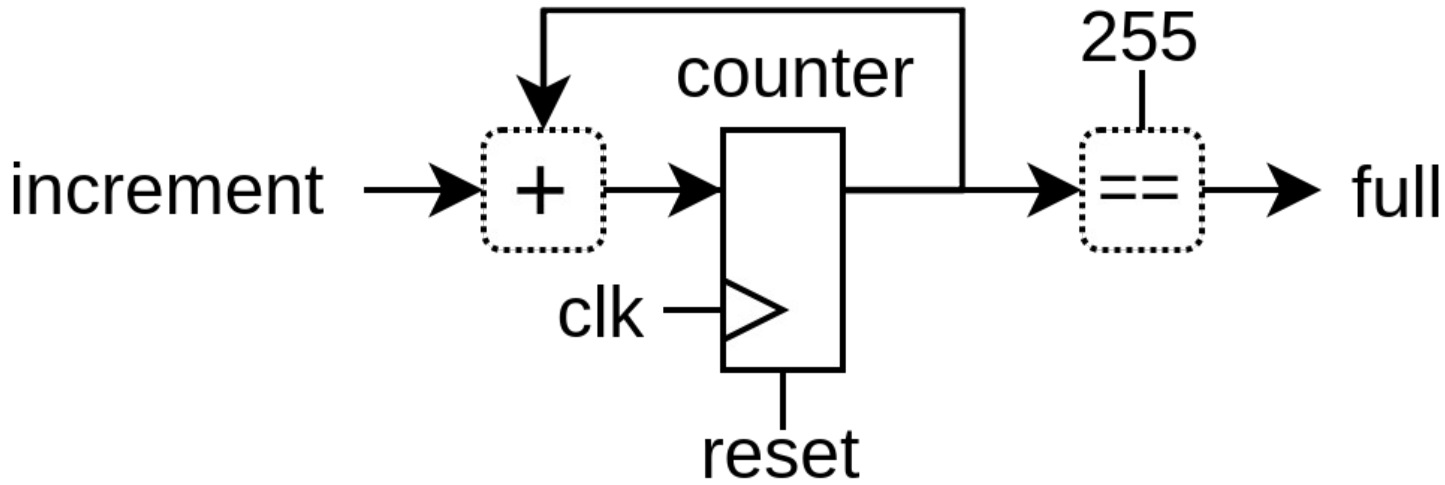
It can be pretty / explicit




```
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
  }
}
```

Using software to elaborate your hardware

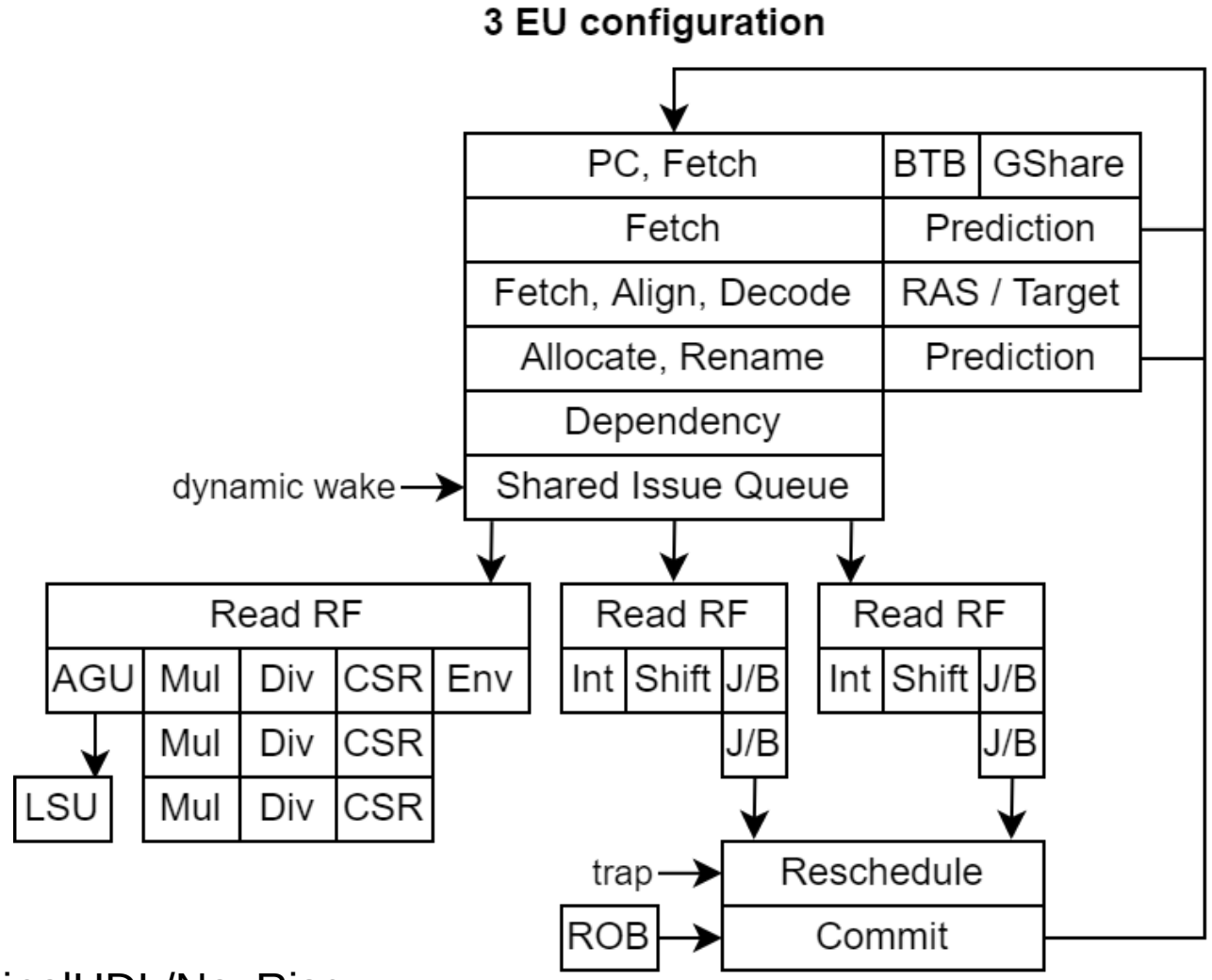


- Control flow : if / for
- Data structures : dynamic array / hash map / hash set
- Lambda function : reduce / fold / map / filter / ...
- OOP : class / software interface
- See https://spinalhdl.github.io/NaxRiscv-Rtd/main/NaxRiscv/abstraction/index.html

CPU instruction set

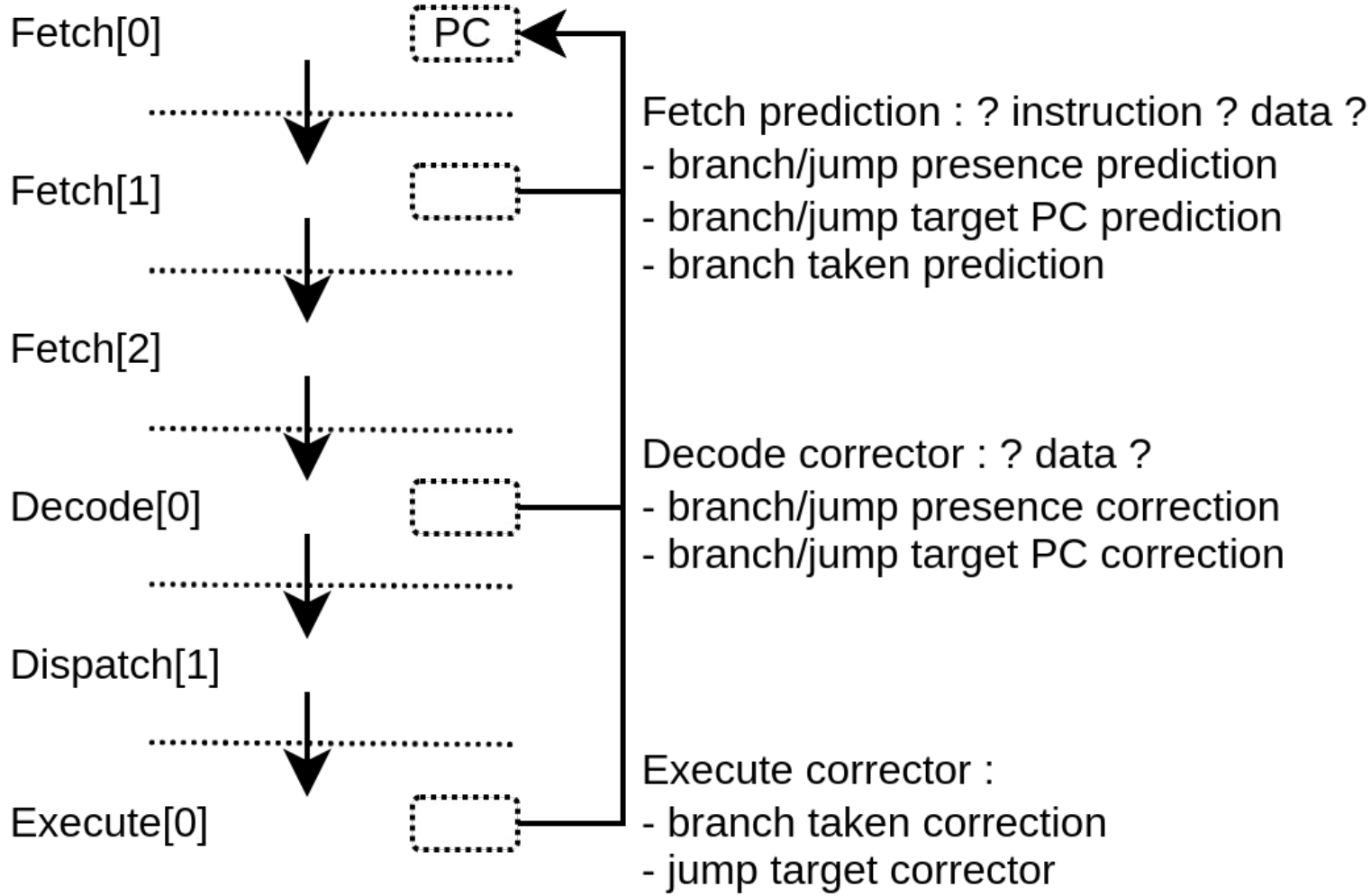
- A few open / free ISA (OpenRISC, , …)
- RISC-V
  - https://riscv.org/technical/specifications/
  - Secretly being embedded in ASICs
  - Bloat free (Mostly, from a FPGA perspective)
  - GCC / LLVM
  - Qemu support
  - Linux / Debian port
  - …

Basics for an out of order CPU (NaxRiscv)



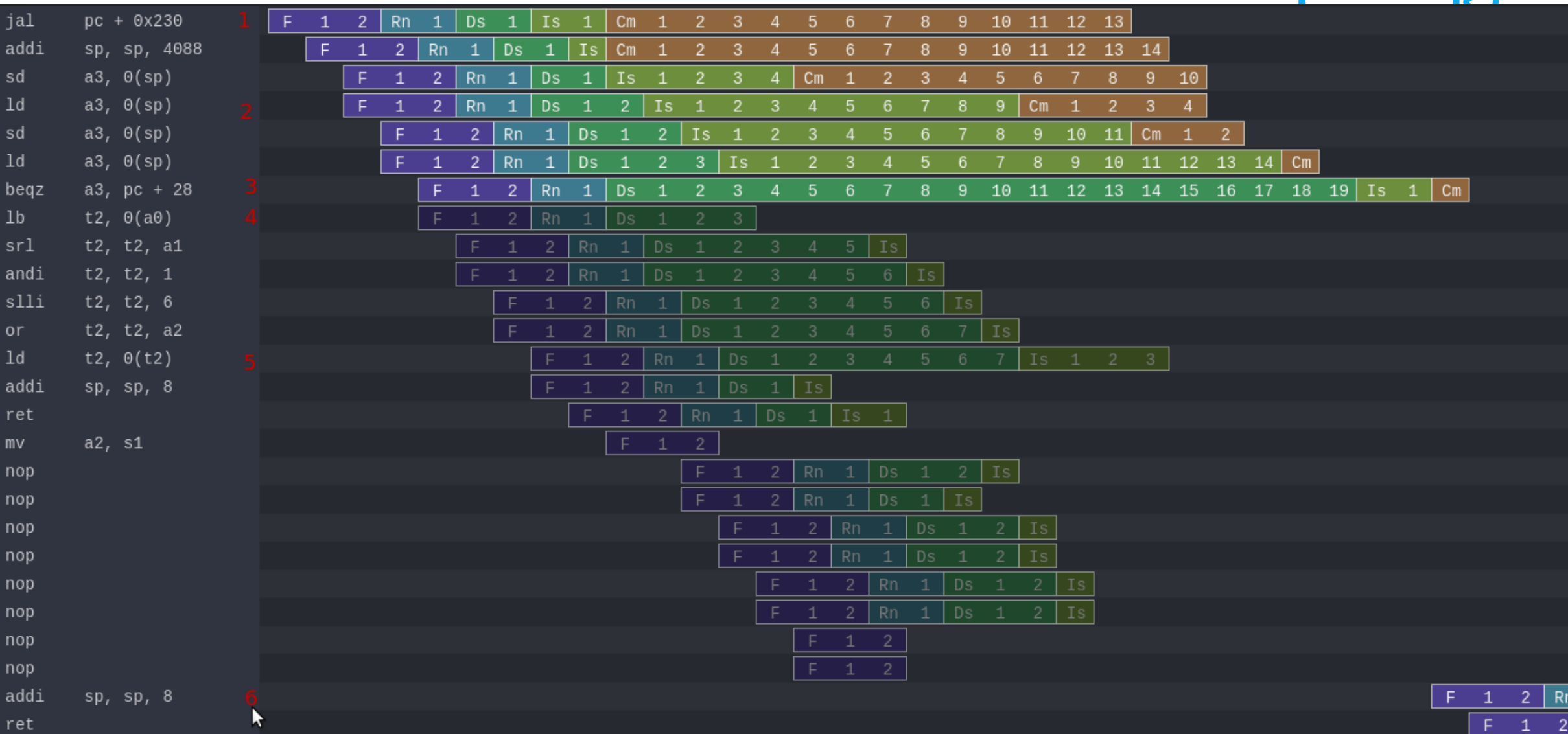
https://github.com/SpinalHDL/NaxRiscv

“Branch prediction”

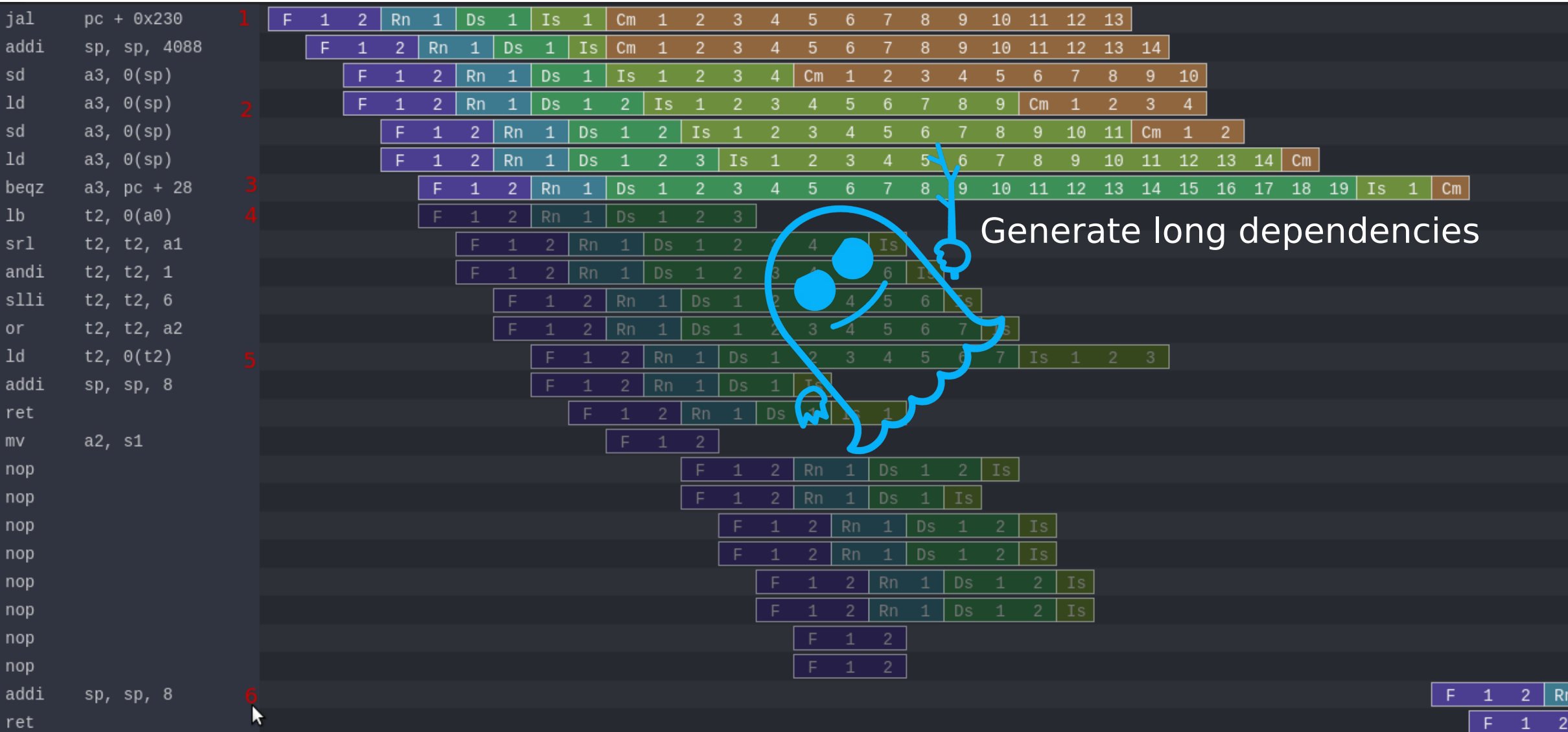


https://spinalhdl.github.io/NaxRiscv-Rtd/main/NaxRiscv/misc/index.html#security

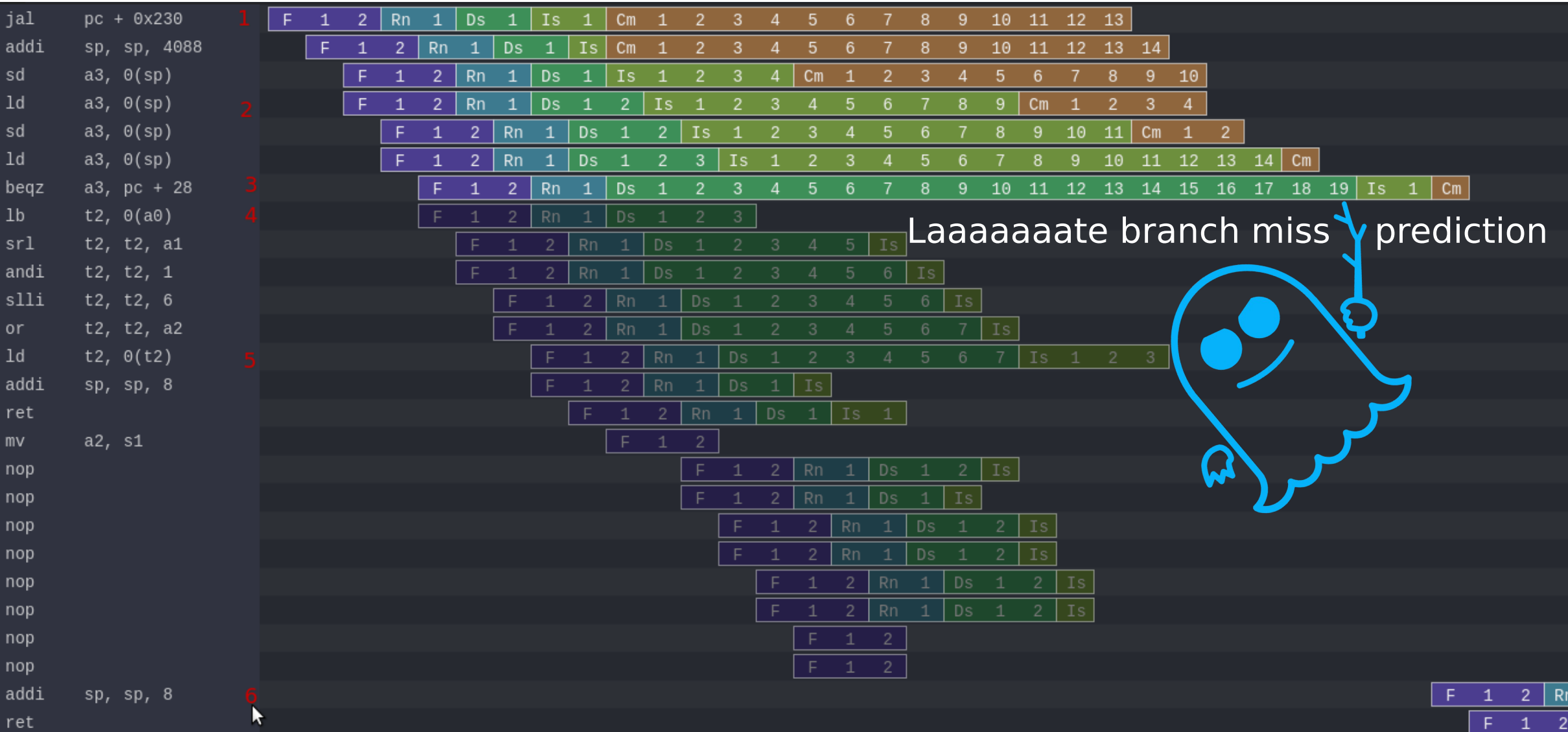
Secret leak (Konata trace)



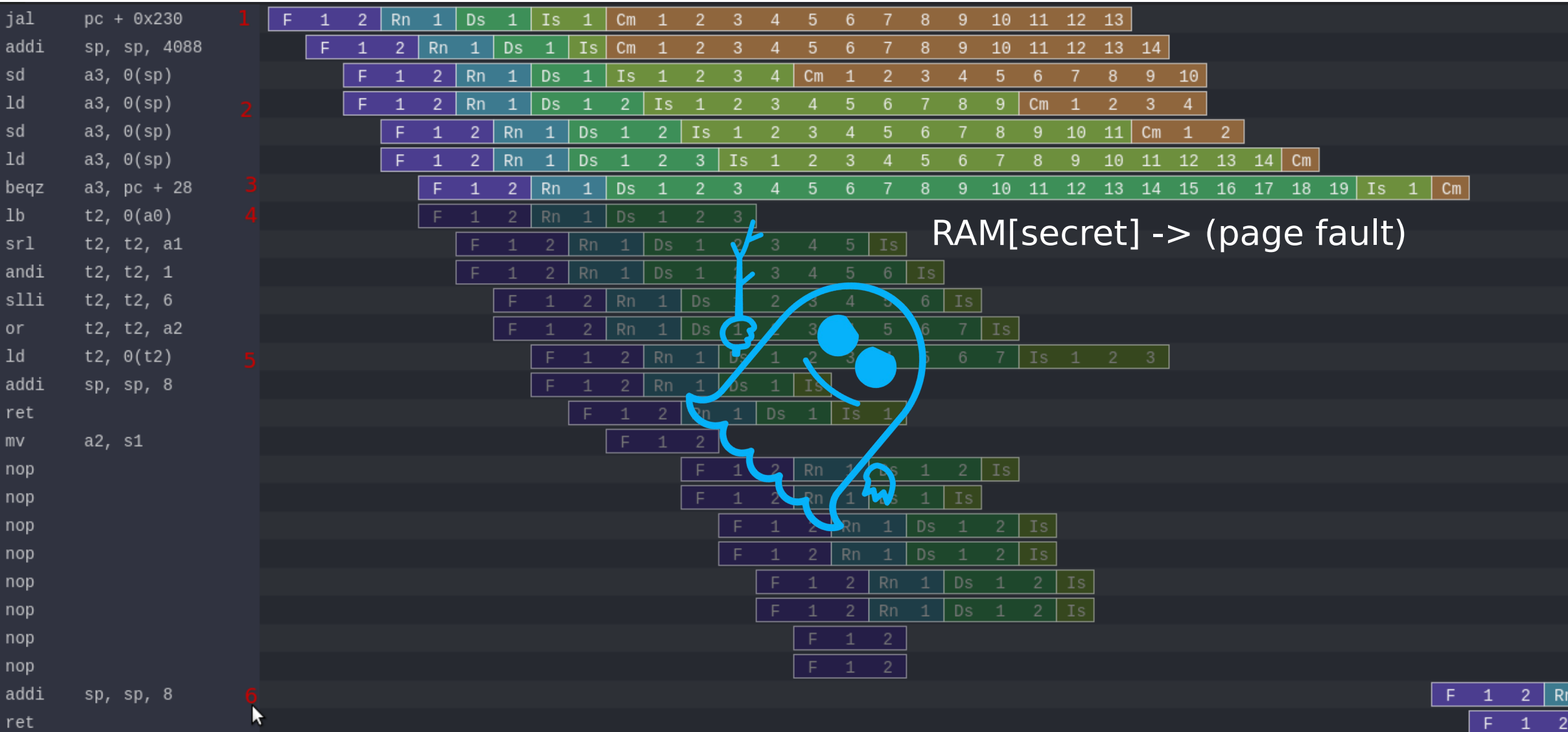
Secret leak


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

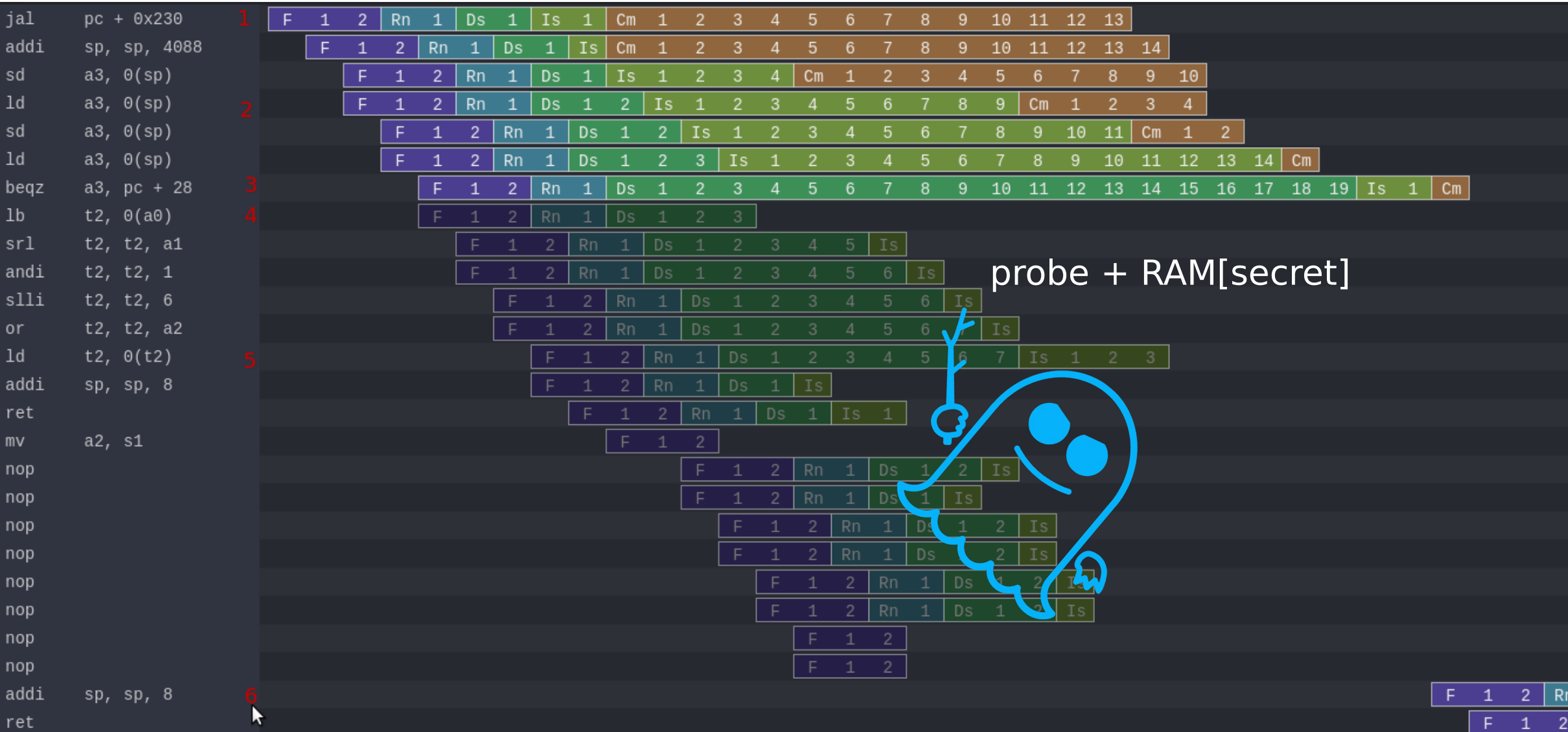
Secret leak


jal pc + 0x230
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
Laaaaaaaate branch miss prediction

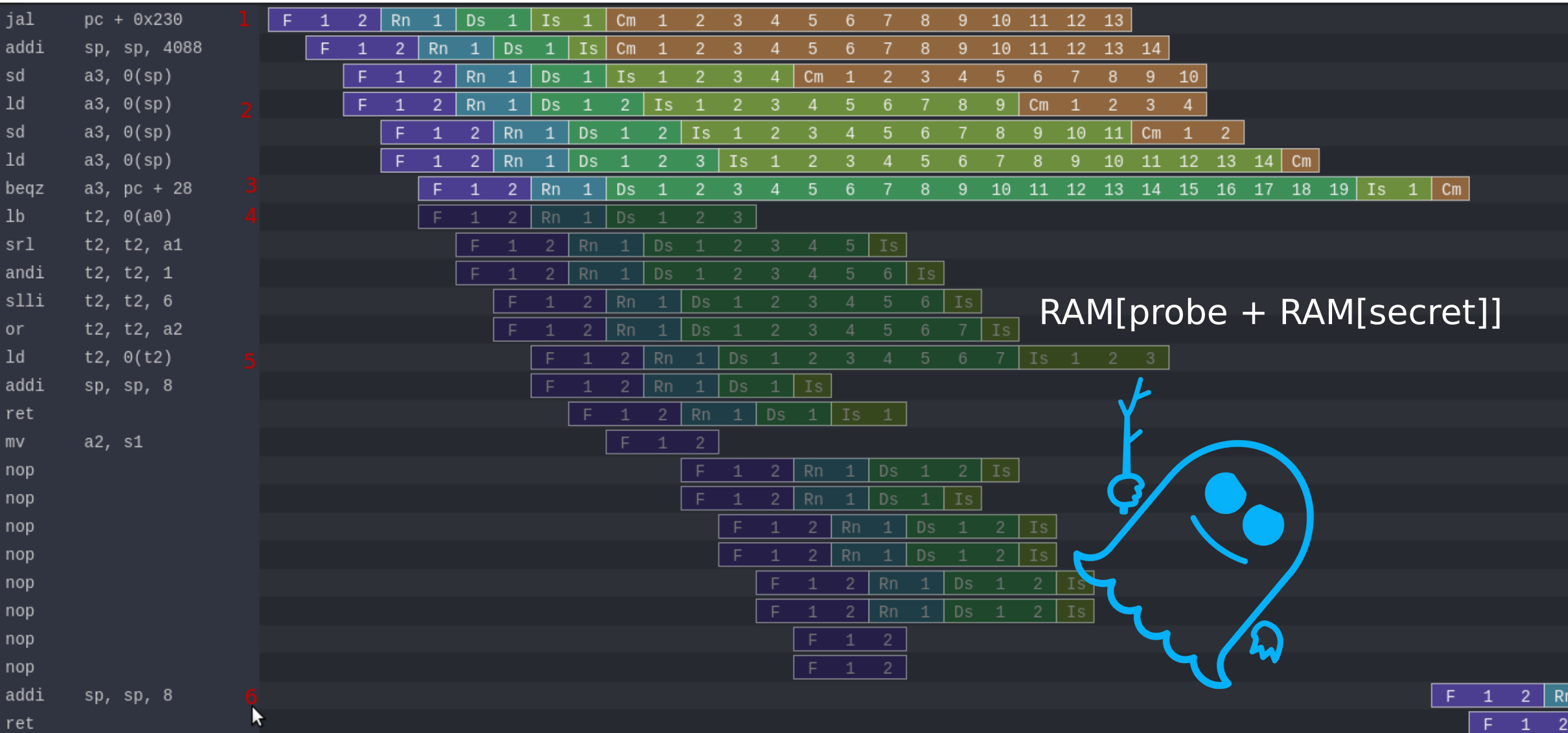
Secret leak


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

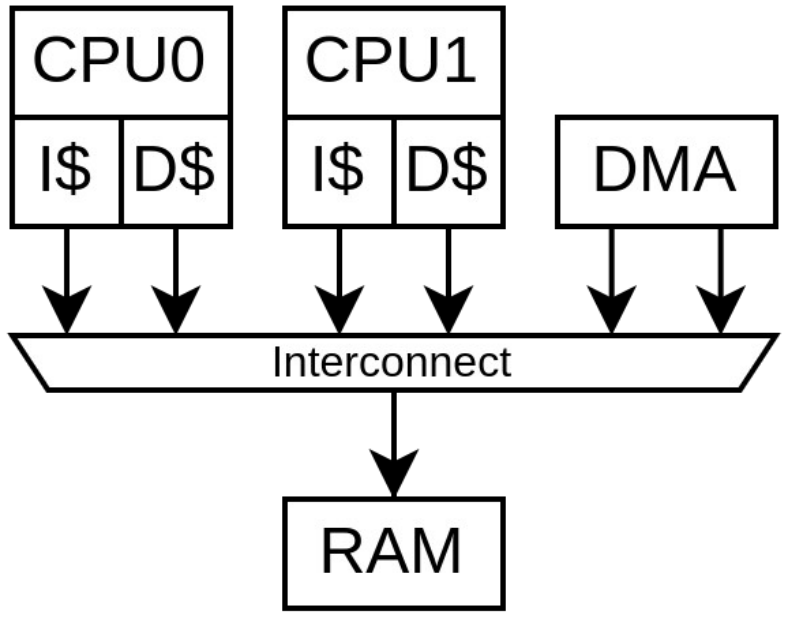

jal pc + 0x230
addi sp, sp, 4088
sd a3, 0(sp)
ld a3, 0(sp)
sd a3, 0(sp)
ld a3, 0(sp)
beqz a3, pc + 28
lb t2, 0(a0)
srl t2, t2, a1
andi t2, t2, 1
slli t2, t2, 6
or t2, t2, a2
ld t2, 0(t2)
addi sp, sp, 8
ret
mv a2, s1
nop
nop
nop
nop
nop
nop
nop
nop
addi sp, sp, 8
ret
1
2
3
4
5
6
F Rn Ds Is Cm
probe +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

Secret leak

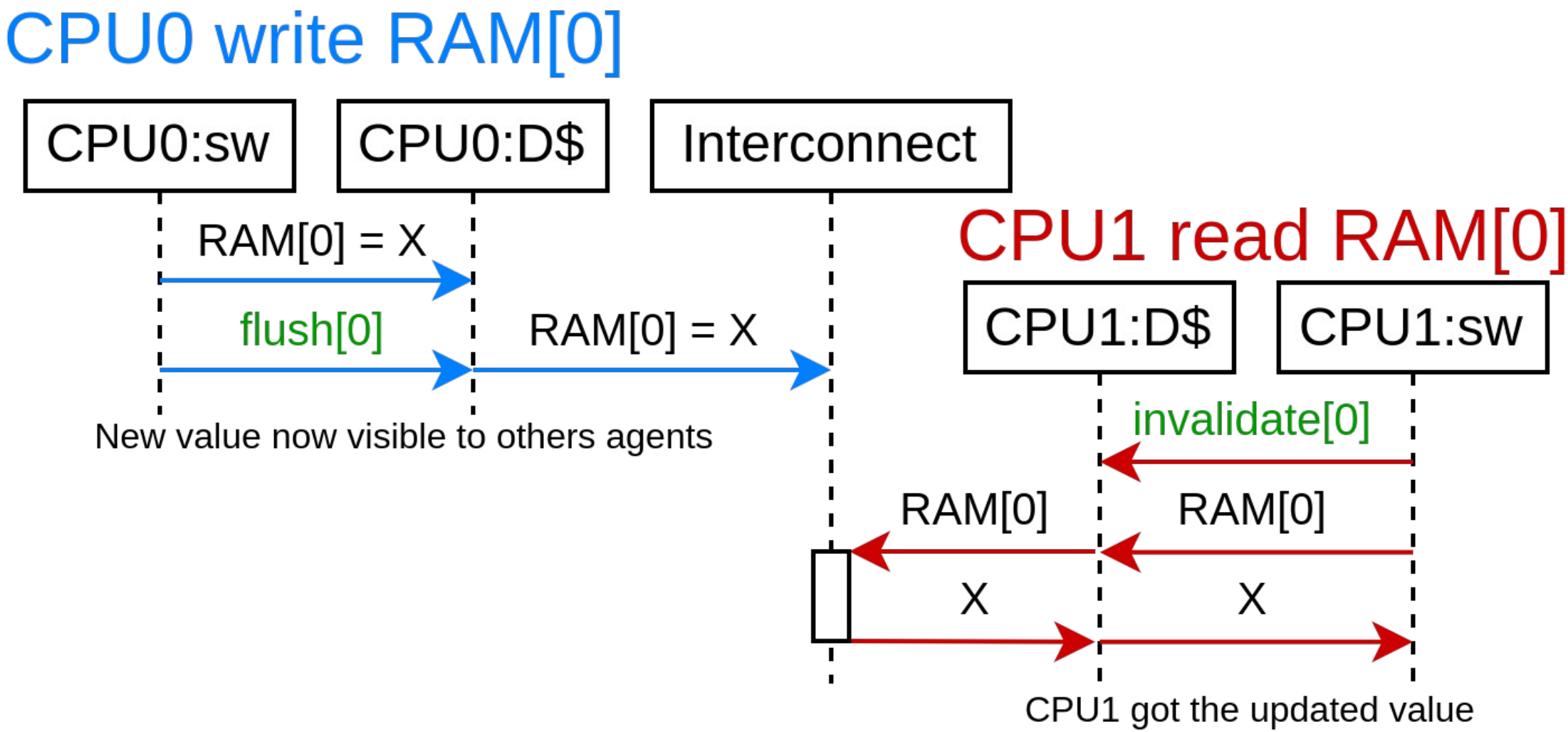

jal pc + 0x230
addi sp, sp, 4088
sd a3, 0(sp)
ld a3, 0(sp)
sd a3, 0(sp)
ld a3, 0(sp)
beqz a3, pc + 28
lb t2, 0(a0)
srl t2, t2, a1
andi t2, t2, 1
slli t2, t2, 6
or t2, t2, a2
ld t2, 0(t2)
addi sp, sp, 8
ret
mv a2, s1
nop
nop
nop
nop
nop
nop
nop
nop
addi sp, sp, 8
ret
1
2
3
4
5
6
RAM[probe + RAM[secret]]

Multi core requirements (RISC-V linux)

- Hardware cache coherency
- Inter-CPU software interrupts

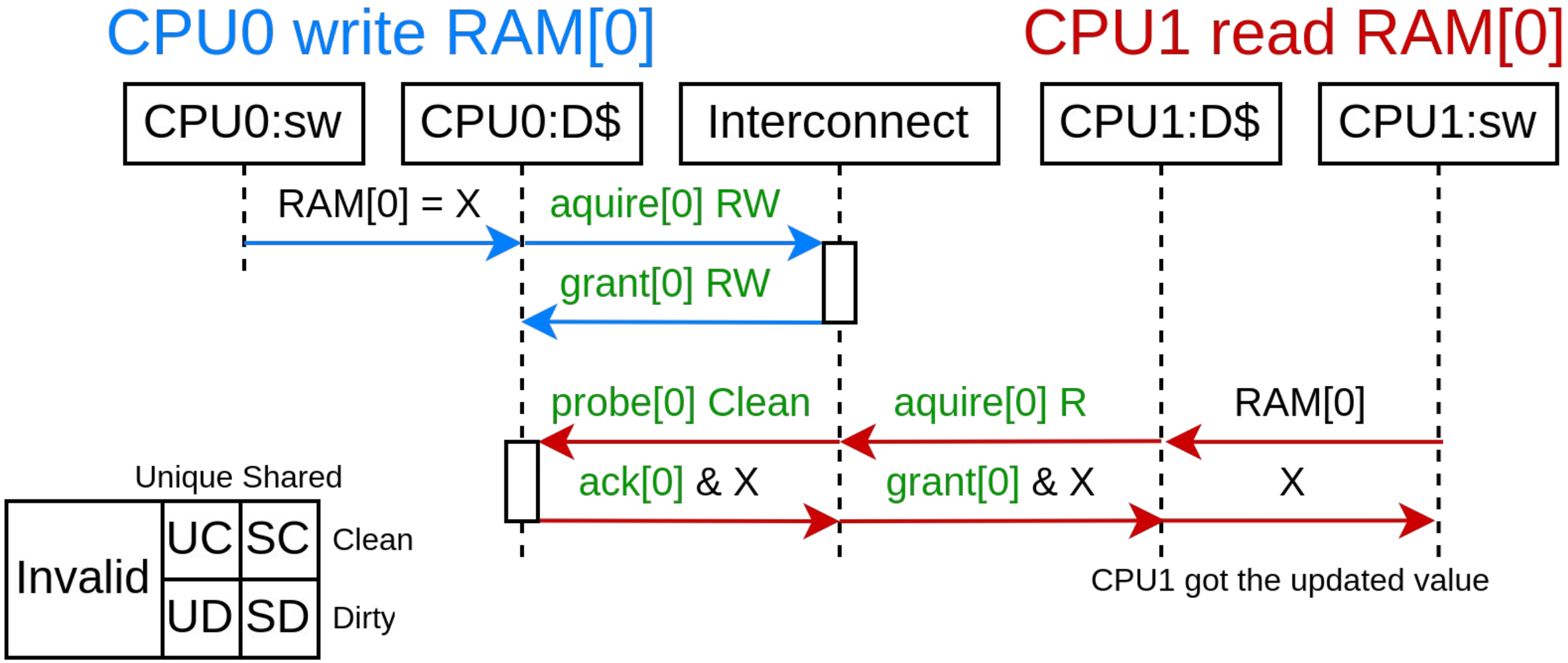


Software cache coherency



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 CPU)
```

Cache coherency

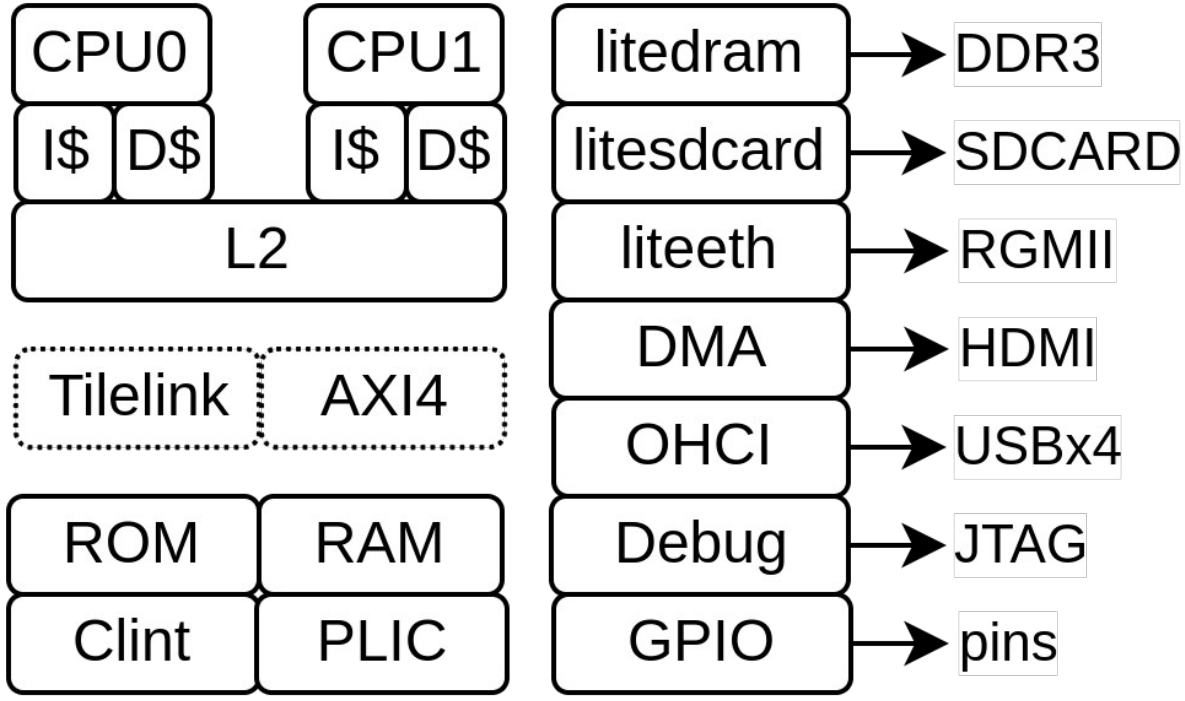
- Open specification : Tilelink
- Multiple open source implementations, ex :
  - https://github.com/SpinalHDL/SpinalHDL/tree/dev/lib/src/main/scala/spinal/lib/bus/tilelink
- Coherent L2 design
  - https://github.com/SpinalHDL/SpinalHDL/blob/dev/lib/src/main/scala/spinal/lib/bus/tilelink/coherent/Cache.scala
  - Exposes a lot of performance tricks
    - 1 pending transaction per 64 bytes block
    - 64 bytes per burst max
    - Keep bursts aligned

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
- + minor 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

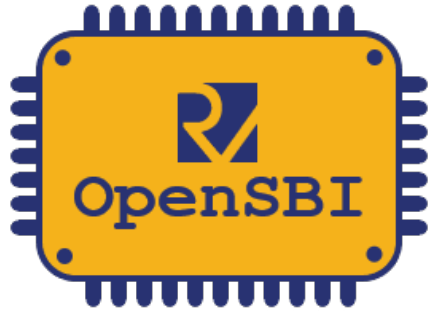
Synthesis / Place / Route

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

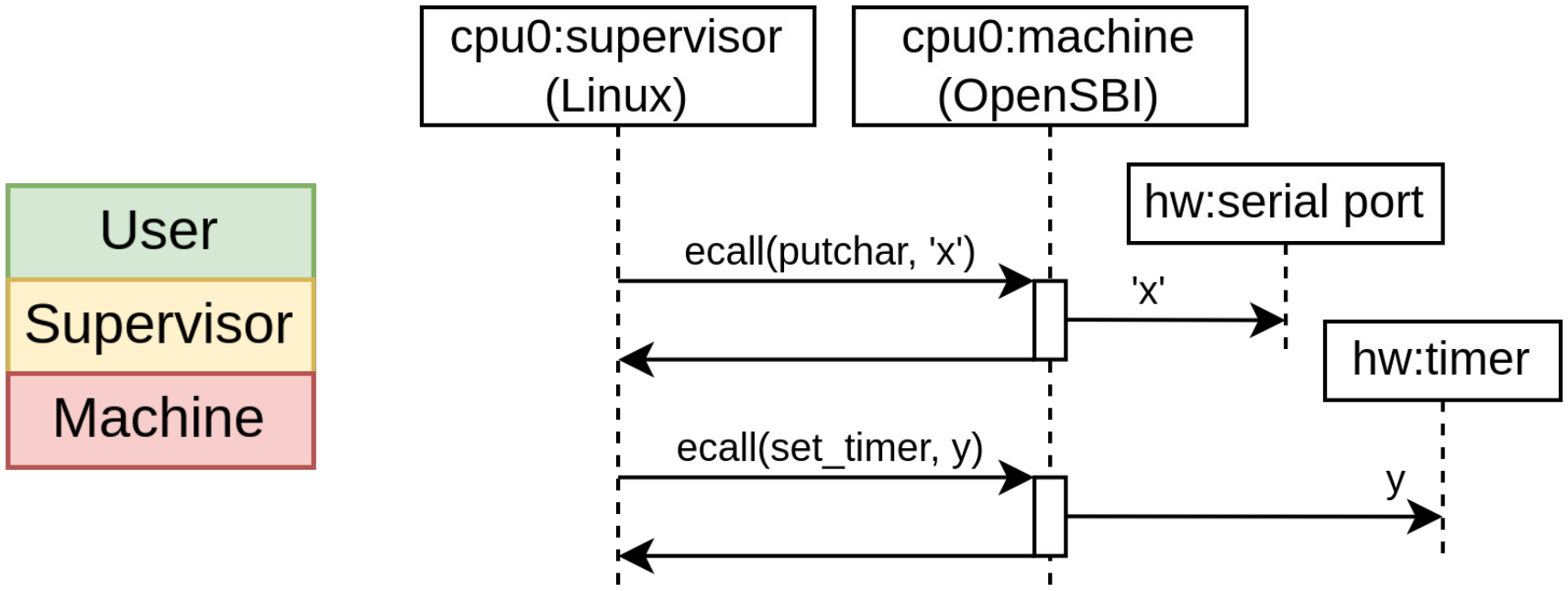
Running Linux

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

SBI (Supervisor Binary Interface)

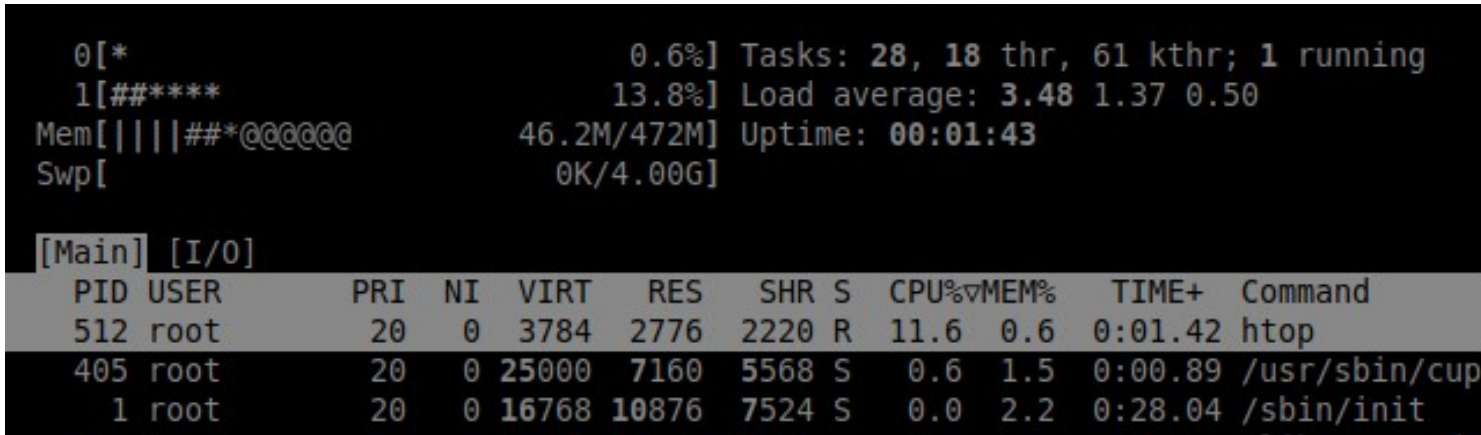


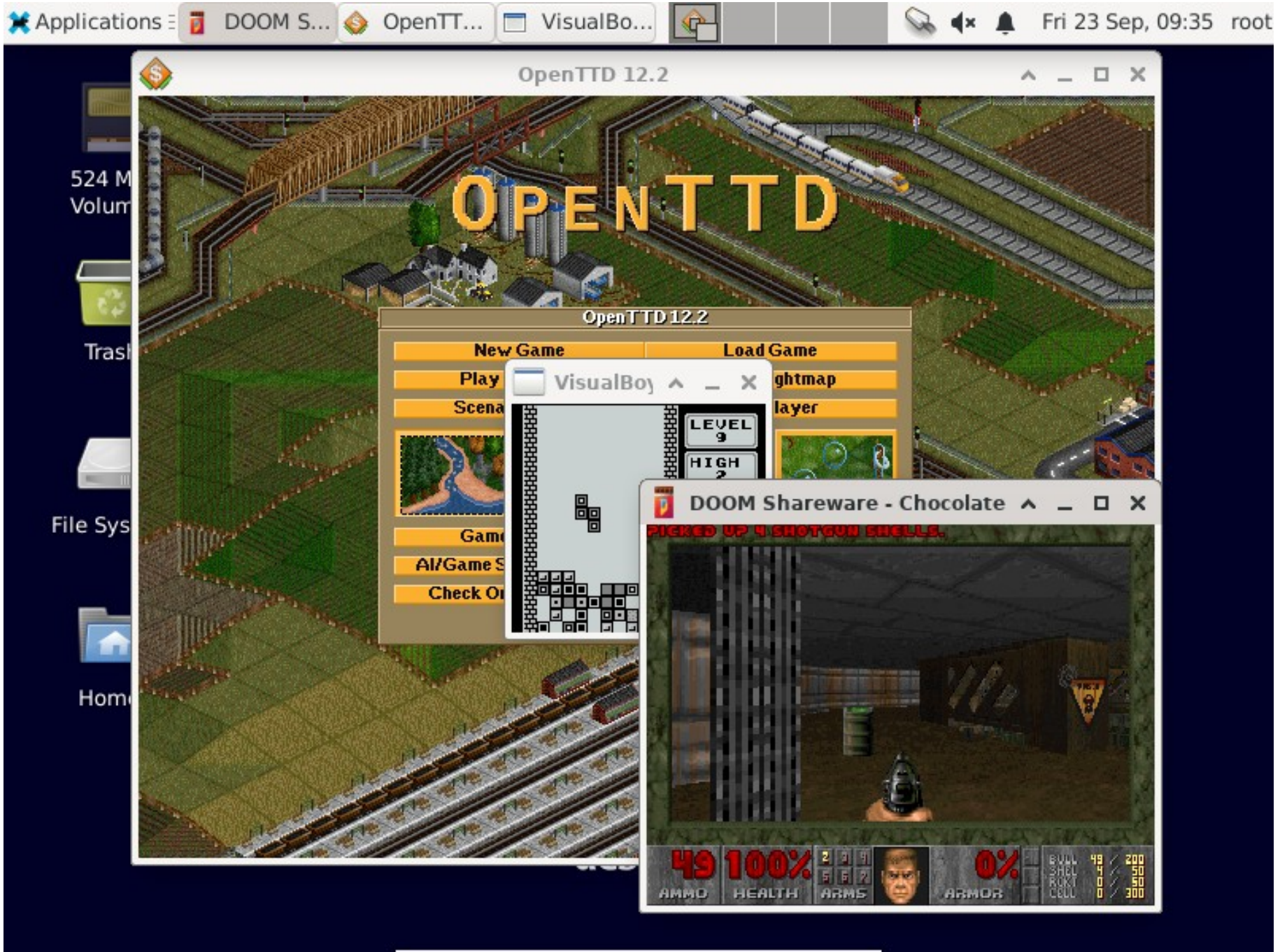
- https://github.com/riscv-non-isa/riscv-sbi-doc
- https://github.com/riscv-software-src/opensbi



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

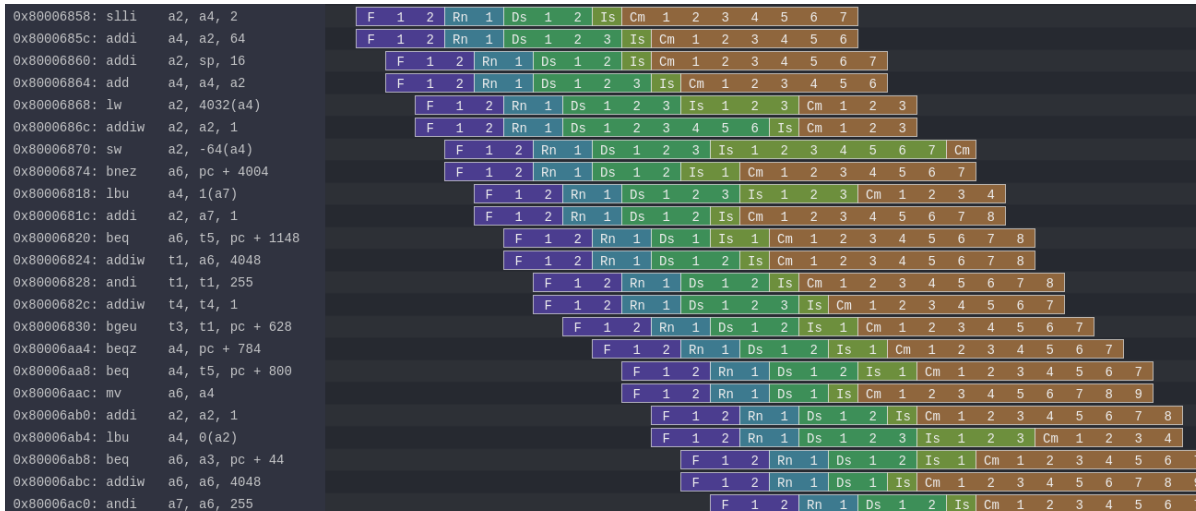
Hello darkness my old friend

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

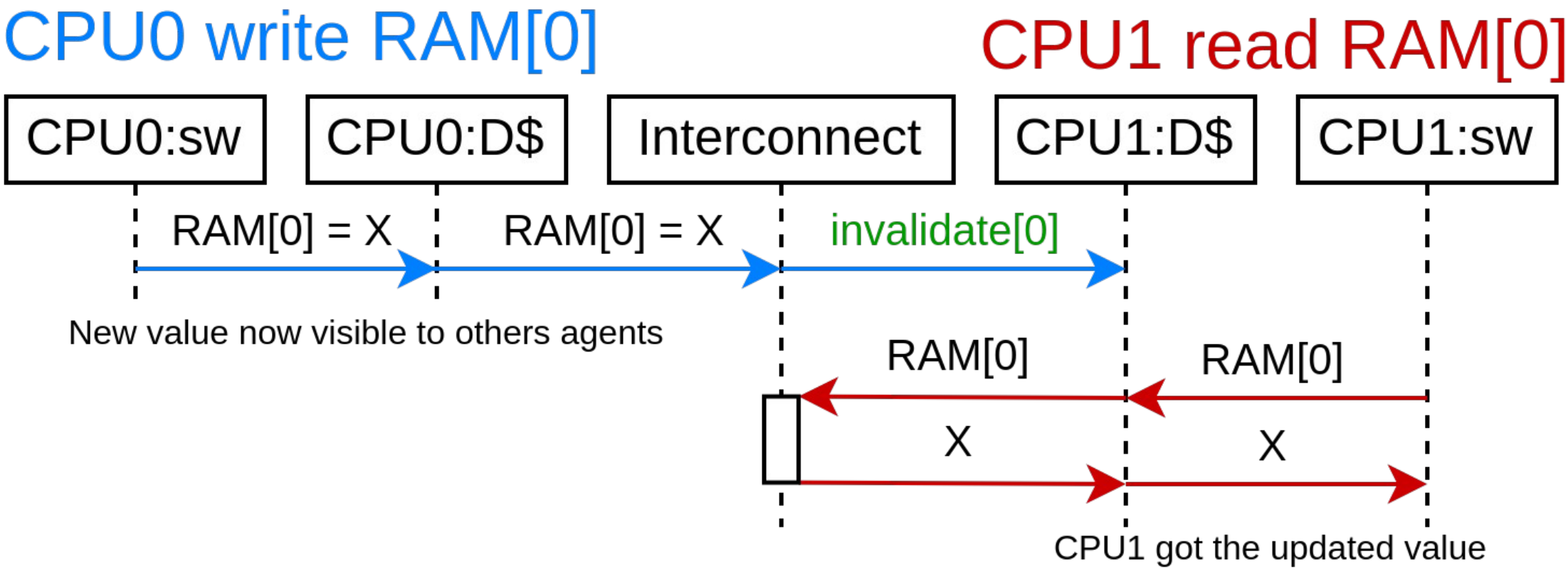
- I hope you had a good time
- If you are looking for open-source project funding :
  - NLnet Foundation (they helped me a lot)

Speculative execution

- Execute as much as you can in advance
- Bad speculation → no side effect on the **ISA states**
  - x1, x2, .. , x31
  - CSR
  - Global memory (RAM)
- But what’s about all the others **hardware states**
  - Branch predictors (BTB, RAS, GShare)
  - I$ D$ tags (is address X in the cache or not)
  - Those are easy to probe

Hardware cache coherency

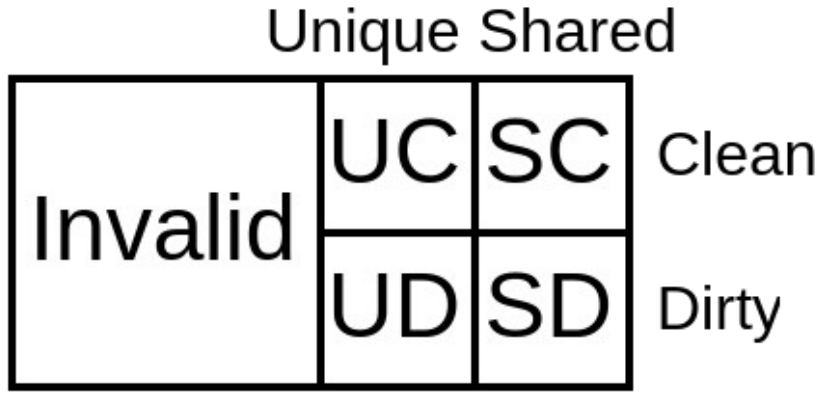
without memory block ownership



Hardware cache coherency

with memory block ownership

- Caches have to acquire memory block permissions
  - read [write] permissions
  - 64 bytes per block (typically), aligned
- Cache line states
  - Valid / Invalid
  - Clean / Dirty
  - Unique / Shared



Boot sequence

- FPGA bitstream
- Litex bios
- OpenSBI
- [Uboot]
- Linux
- Buildroot / Debian / ...