Lab 1 Overview and Microcode Review

Section 2

2/1/2016

Agenda

- Lab 1
  - Why we're asking you?
  - What we've given you?
  - What you're asked to do?
- Microcode Review
- Miscellaneous (RISC-V, Chisel, etc.) Questions

Computer Architect's Job

- Find the common case(s), and make sure you can support them efficiently
  - Sometimes at the cost of support the uncommon case inefficiently
- Examples
  - Found 20% of VAX instructions responsible for 60% of microcode, but only account for 0.2% of execution time!
  - Register immediate instructions
- How do we know what to do?
  - Intuition, Simulation, Building, Experiments, …

Experiments aren't easy

- Although there are properties that you can prove mathematically, computer architecture is often based on empirical studies
- This means that your data will be application specific
  - Pick your applications carefully! (benchmarks)
- "It depends": Always think about both sides of the argument
  - Keep asking questions to yourself to understand

Lab 1

- Understand how in-order pipelined microarchitecture affects processor
- Guided
  - CPI
  - Instruction Mix
- Open-ended
  - Bypassing
  - CISC microcode
  - General Design
  - Your fun idea!

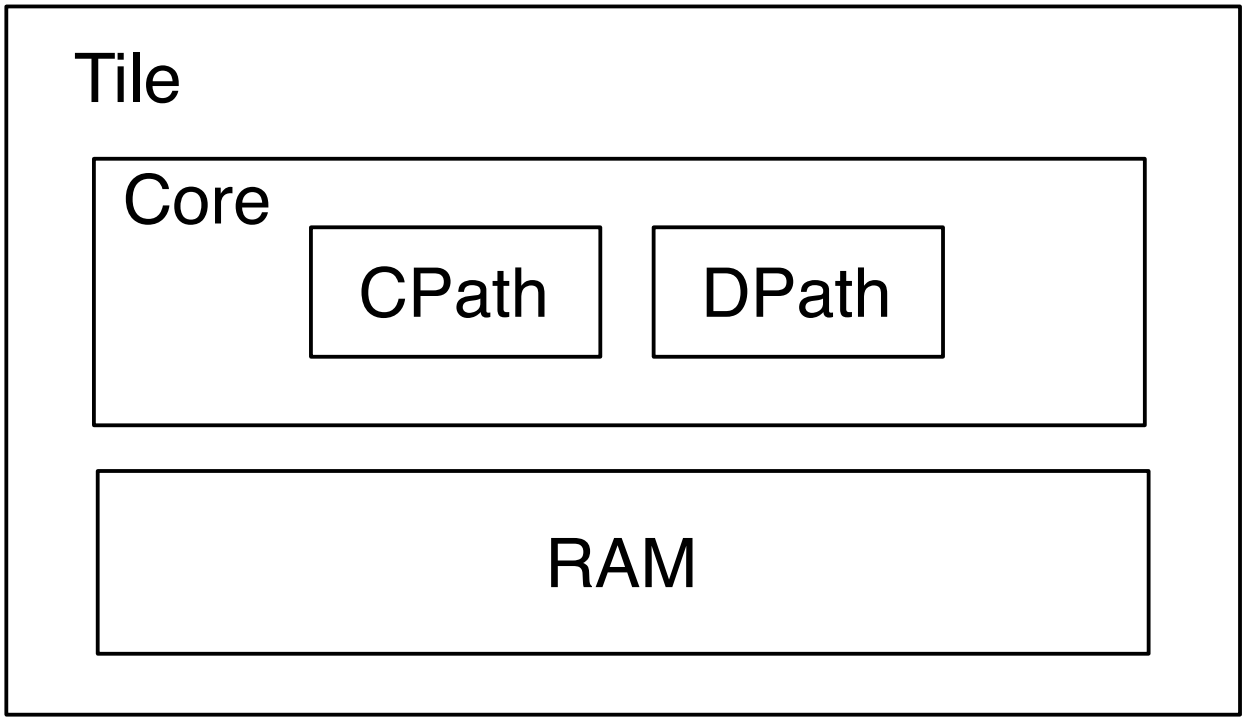
Lab 1 Given

- Provided RISC-V 32I processors
  - 1-stage
  - 2-stage
  - 3-stage
  - 5-stage
    - Fully bypassed
    - Interlocked (stall to resolve all hazards)
  - Micro-coded
- Only 1-stage and 5-stage are used in the directed portion
  - 2-stage 3-stage and micro-code are there for you to investigate or use in the open-ended portion

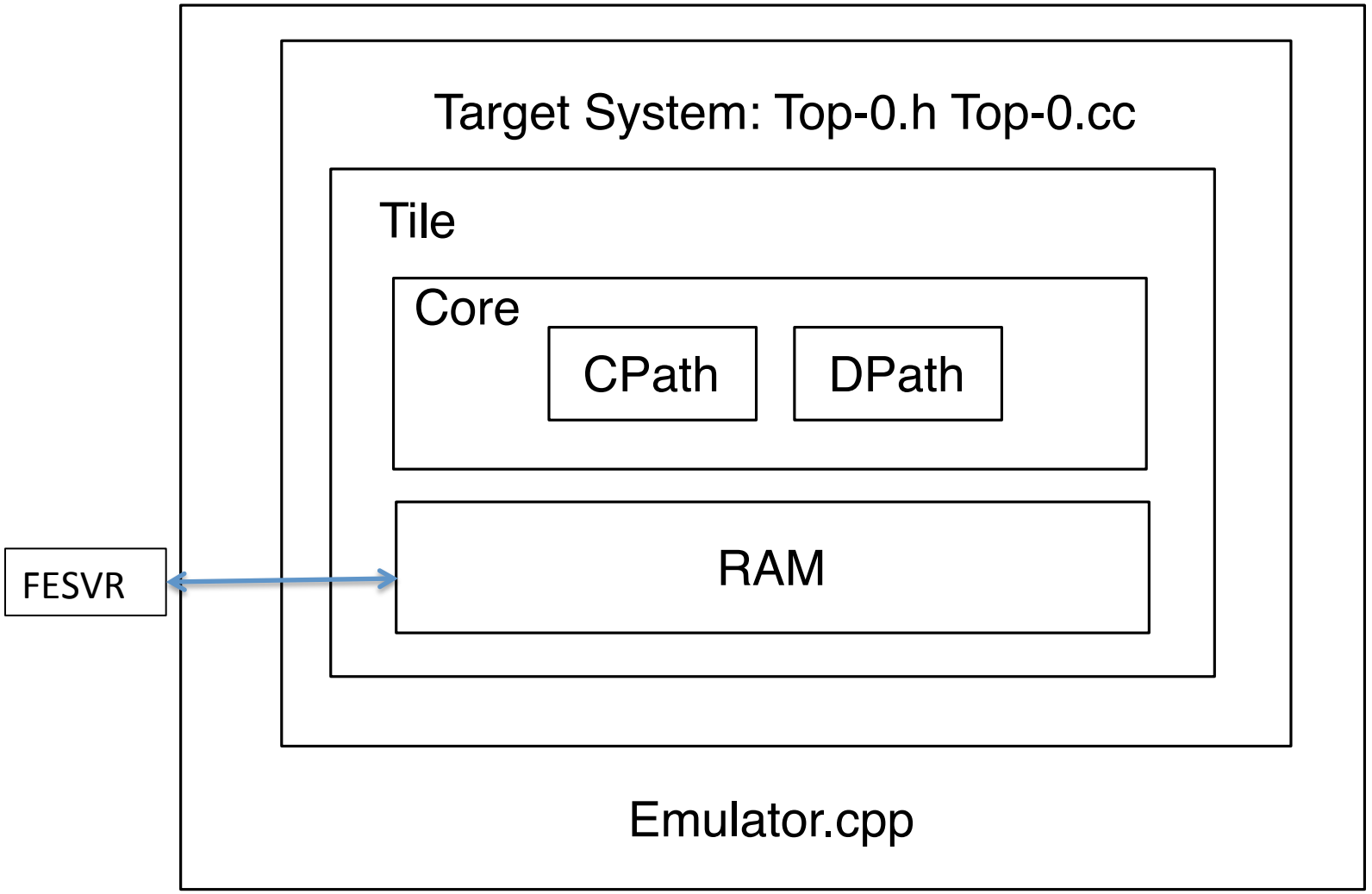
Lab 1 Given other Misc.

- Chisel -> C++ simulator (emulator)
- Benchmarks and test programs
- Instruction Tracer to gather stats
  - CPI, instruction mix
- Questions and analysis
  - Make recommendations
  - Propose new designs

Lab 1 Chisel Processor



Lab 1 Processor Emulator



DEMO!

- Add tools to your path

```
$ source ~cs152/sp13/cs152.bashrc
```

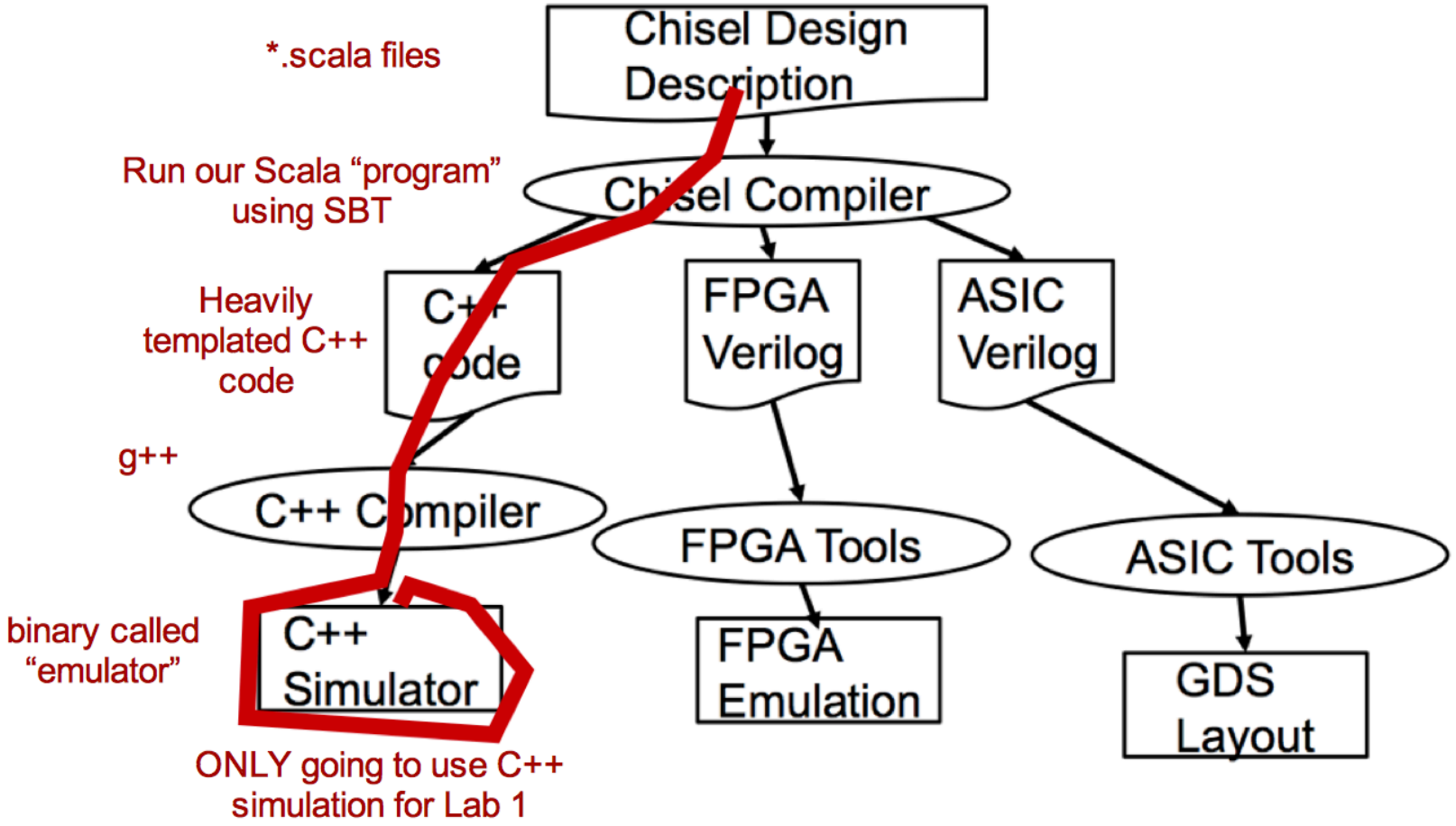
- Copy Lab Files

```
$ cp –R ~cs152/sp13/lab1 .
```

- Build a Chisel processor, Compile simulator, run all tests & benchmarks

```
$ make run-emulator
```

How is Chisel used in Lab



Lab 1 Questions

Microcode/Lecture 2 Review

- Endianness
- Combinational Path
- Critical Path
- Benefits of Microcode
- Horizontal vs Veritcal
- Nanocoding

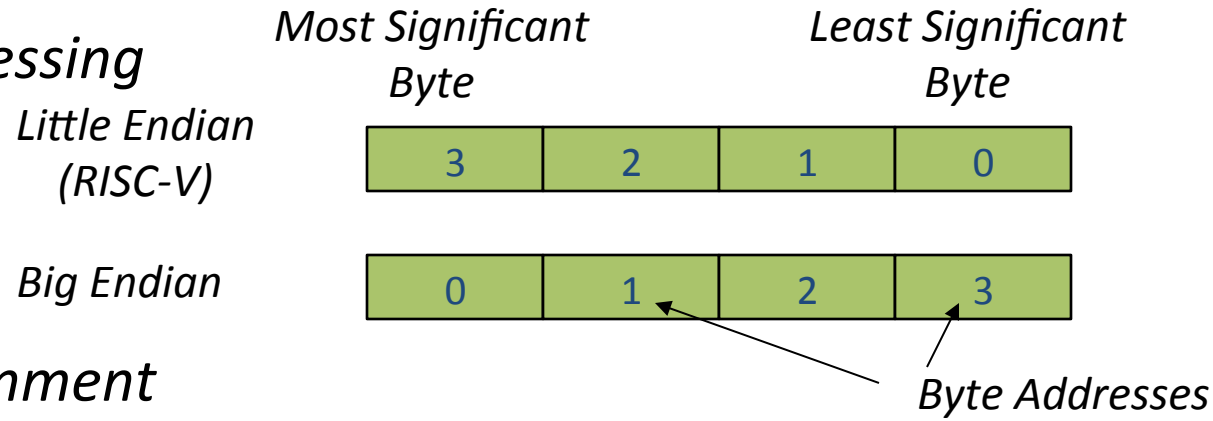
Data Formats and Memory Addresses

Data formats:

8-b Bytes, 16-b Half words, 32-b words and 64-b double words

Some issues

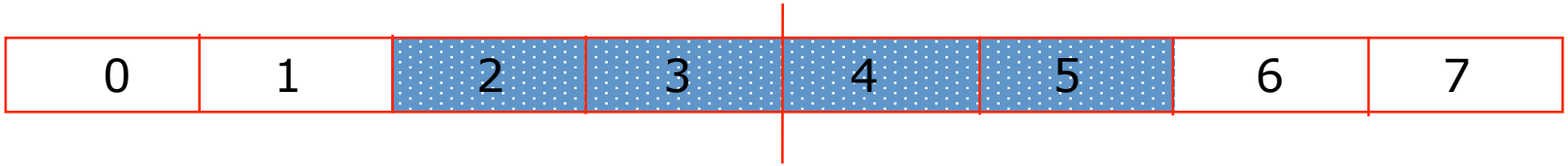
- *Byte addressing*



- *Word alignment*

  Suppose the memory is organized in 32-bit words.

  Can a word address begin only at 0, 4, 8, ... ?



Performance Issues

Microprogrammed control

=> multiple cycles per instruction

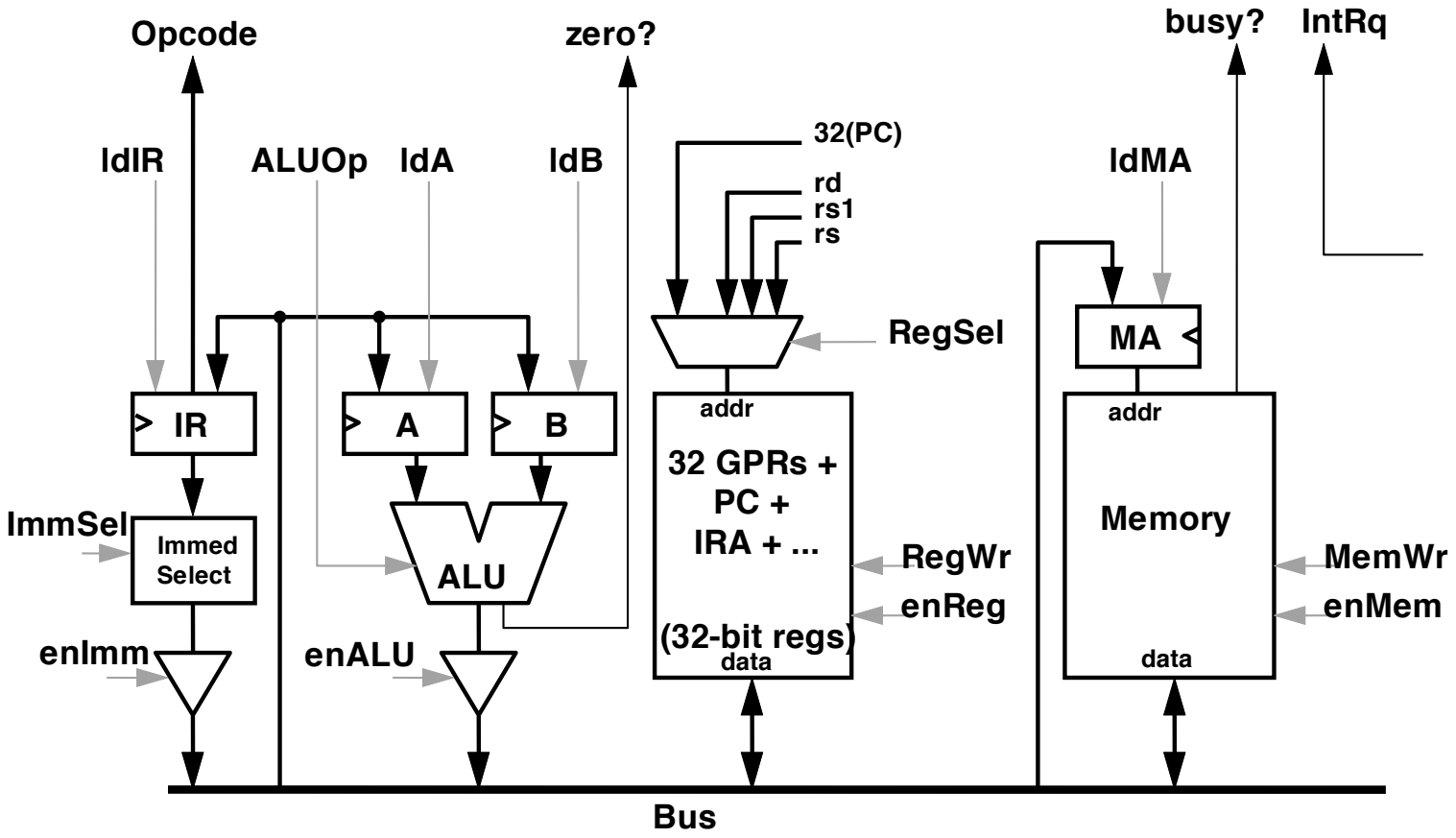
Cycle time ?

$t_C > \max(t_{reg\text{-}reg}, t_{ALU}, t_{\mu ROM})$

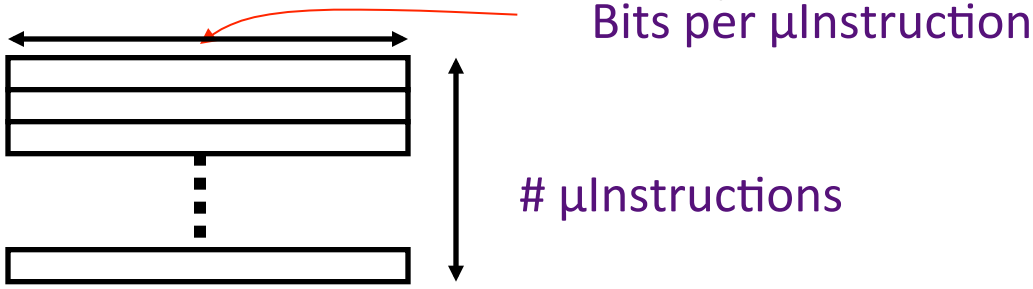
Suppose $10 * t_{\mu ROM} < t_{RAM}$

Good performance, relative to a single-cycle hardwired implementation, can be achieved even with a CPI of 10

Cycle Time, Combinational Path, Critical Path



Horizontal vs Vertical μCode



- Horizontal μcode has wider μinstructions
  - Multiple parallel operations per μinstruction
  - Fewer microcode steps per macroinstruction
  - Sparser encoding ⇒ more bits
- Vertical μcode has narrower μinstructions
  - Typically a single datapath operation per μinstruction
    - separate μinstruction for branches
  - More microcode steps per macroinstruction
  - More compact ⇒ less bits
- Nanocoding
  - Tries to combine best of horizontal and vertical μcode

Nanocoding

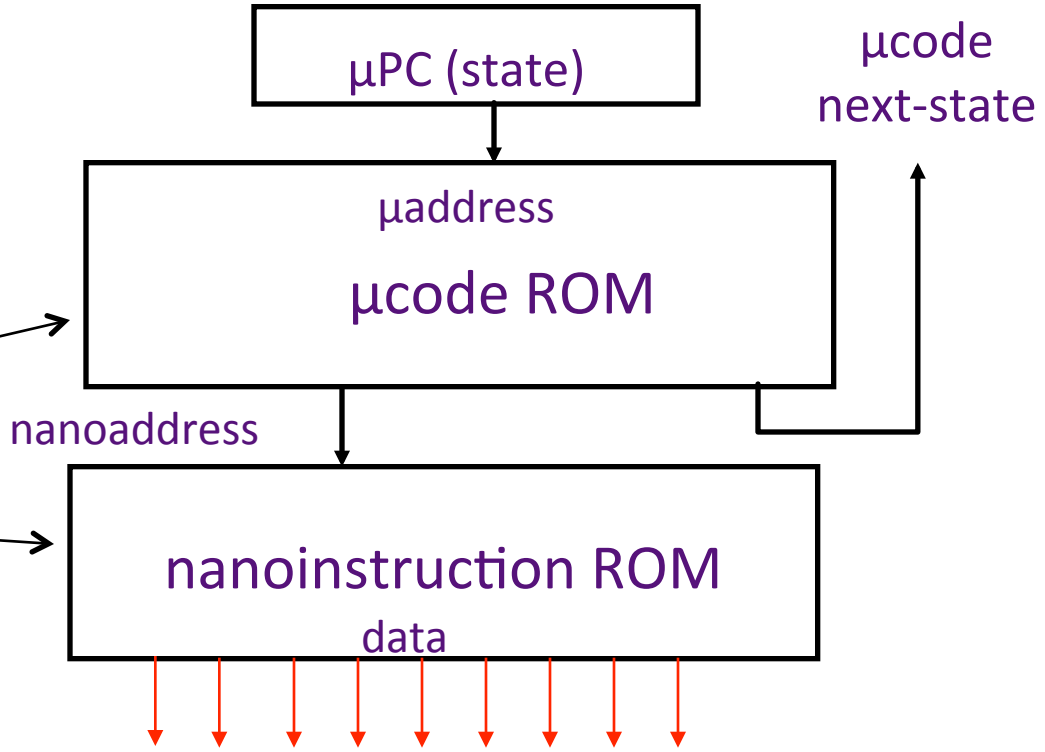
Exploits recurring control signal patterns in µcode, e.g.,

ALU_0 A <= Reg[rs1]

...

$ALUi_0$ A <= Reg[rs1]

...



- MC68000 had 17-bit µcode containing either 10-bit µjump or 9-bit nanoinstruction pointer
  - Nanoinstructions were 68 bits wide, decoded to give 196 control signals

Questions