

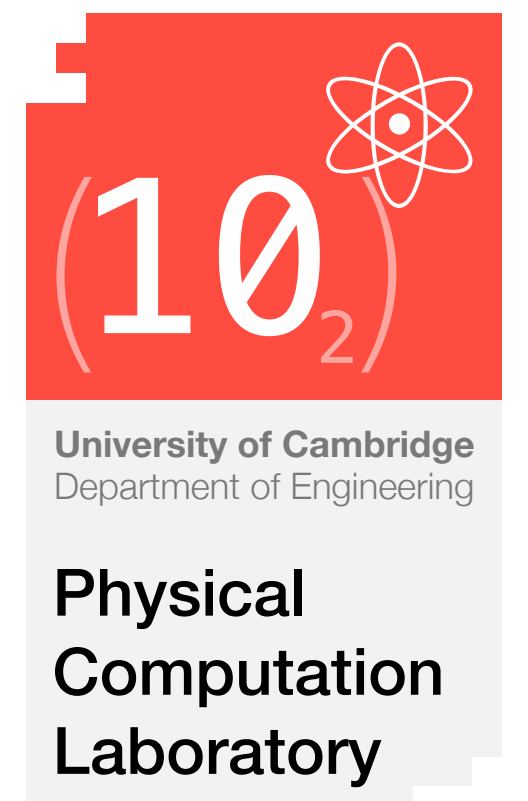
RISC-V Processor Design

Fascicle 1: Introduction to Computer Architecture and RISC-V

(15 minutes)
May 2019

Phillip Stanley-Marbell

Department of Engineering, University of Cambridge
<http://physcomp.eng.cam.ac.uk>



Intended Learning Outcomes for This Fascicle

By the end of this fascicle, you should be able to:

- ❶ Define the primary components of a processor
- ❷ Explain the primary steps involved in executing programs
- ❸ Define the difference between a processor architecture and a microarchitecture
- ❹ Enumerate the 47 RV32I instructions and start looking at the GB3 Verilog RV32I design

Processor Core

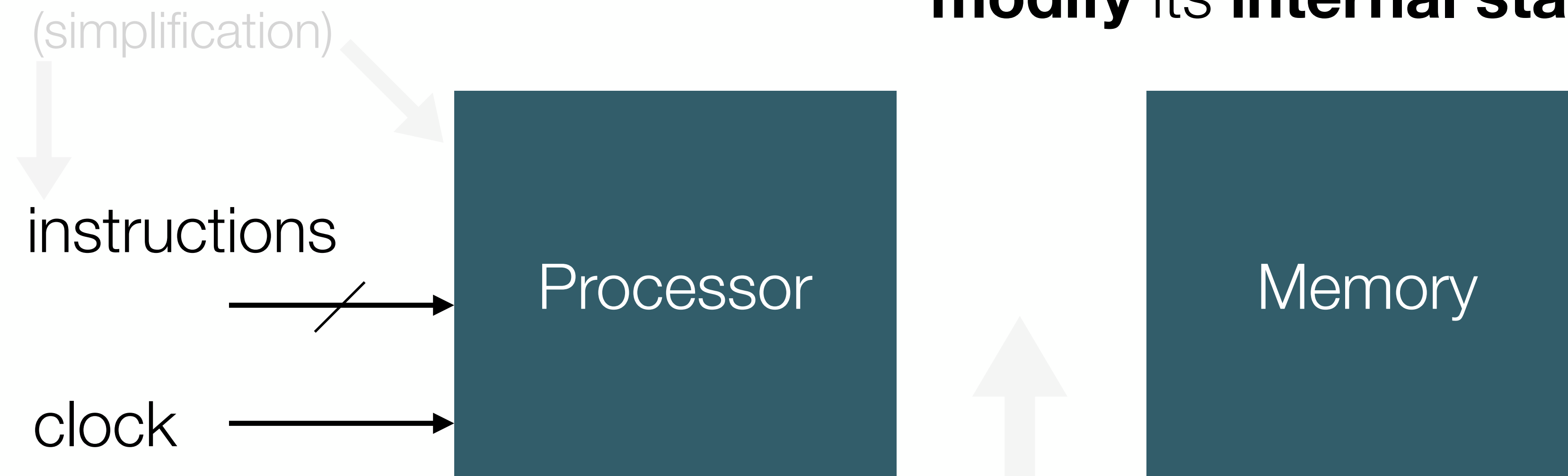
3

15

The processor as a black box

Processor executes instructions

As it executes, these **instructions modify** its **internal state and memory**



Key idea: Some of this internal state is **modified explicitly** by instructions, while other internal state is **modified implicitly**

(We will talk about the interface between the processor and memory later)

Processor Core

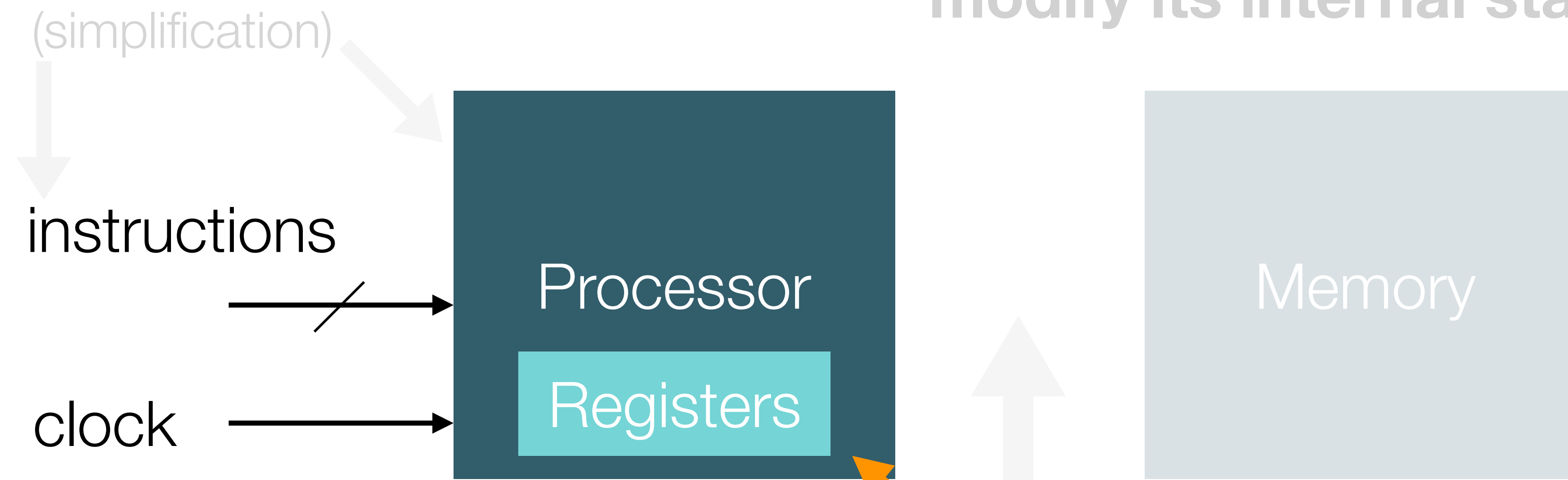
4

15

Looking inside the black box: **Registers and other architectural state**

Processor executes instructions

As it executes, these instructions modify its internal state and memory



Key idea: The part of the internal state that programs explicitly modify is referred to as the **architectural state**

Example of architectural state: registers

(We will talk about the interface between the processor and memory later)

Processor Core

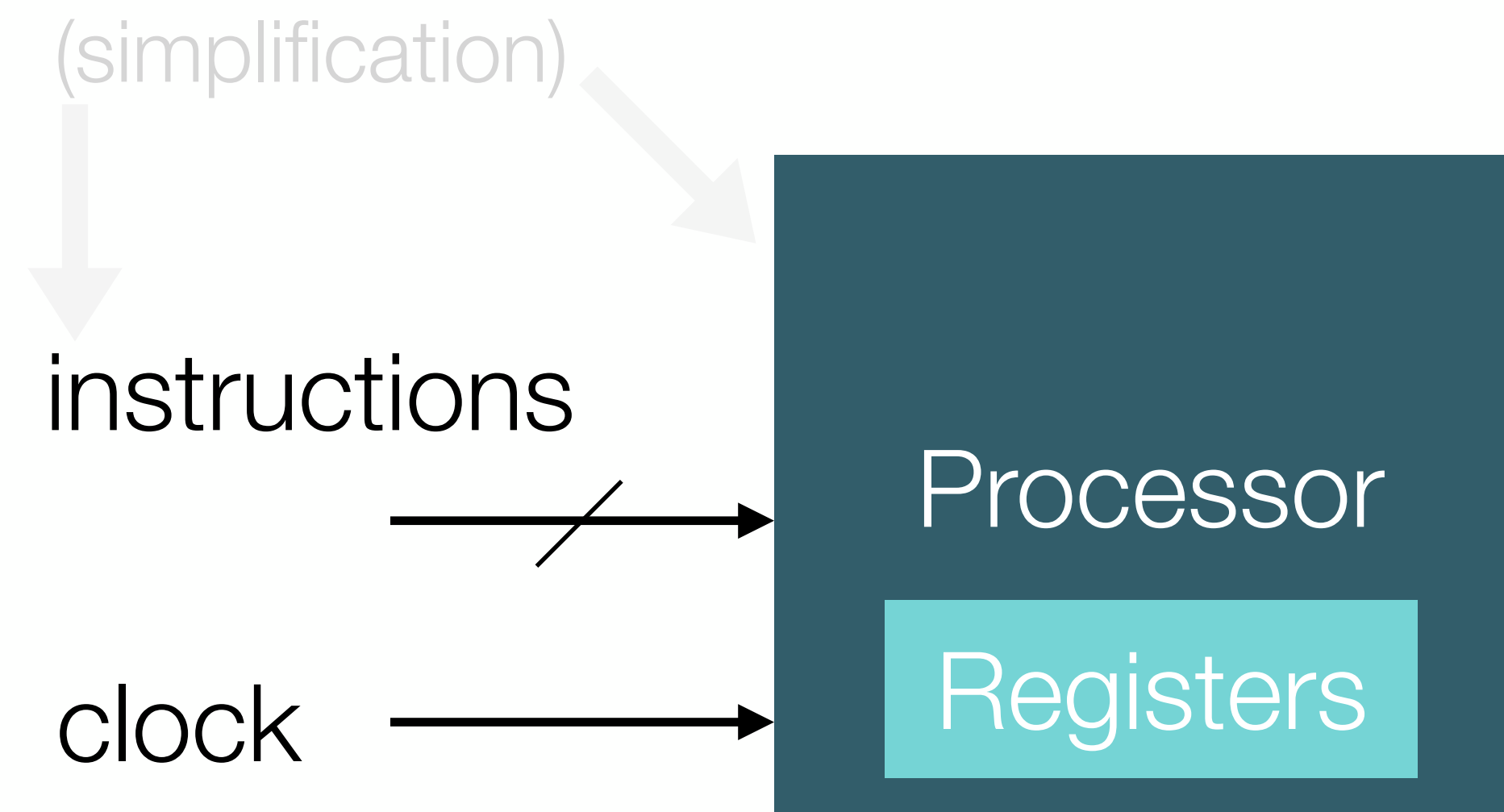
5

15

Looking inside the black box: **Microarchitectural state**

Processor executes instructions

As it executes, these instructions modify its internal state and memory



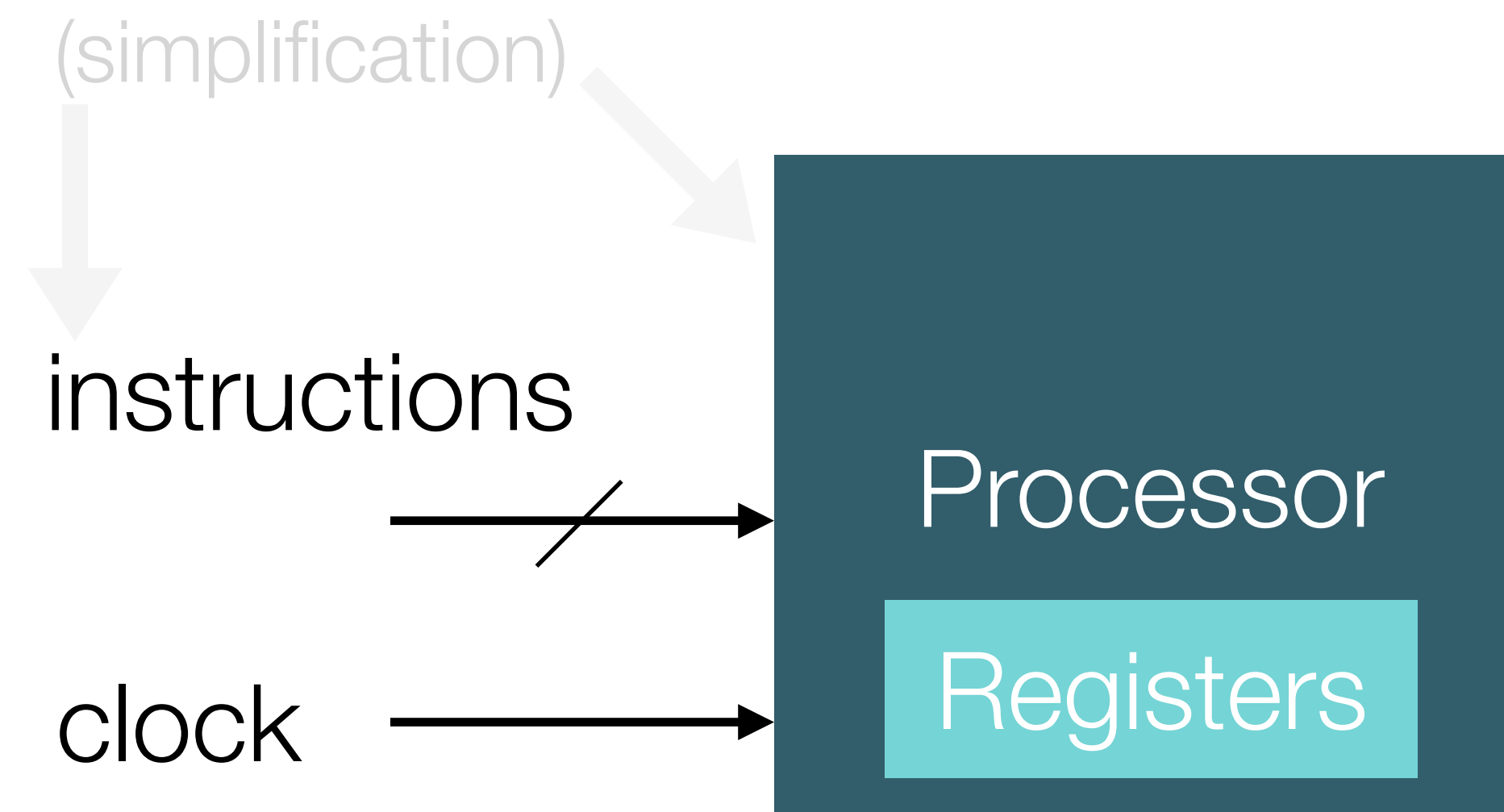
Key idea: The part of the internal state that programs implicitly modify is referred to as the **microarchitectural state**

Example of microarchitectural state: logic values of signals internal to the design, other than the registers

(We will talk about the interface between the processor and memory later)

Architecture versus Microarchitecture

Processor executes instructions



As it executes, these instructions modify its internal state and memory



Key idea ①: The part of the internal state that is **implicitly modified** by programs is referred to as the **microarchitectural state**

Key idea ②: The part of the internal state that is **explicitly modified** by programs is referred to as the **architectural state**

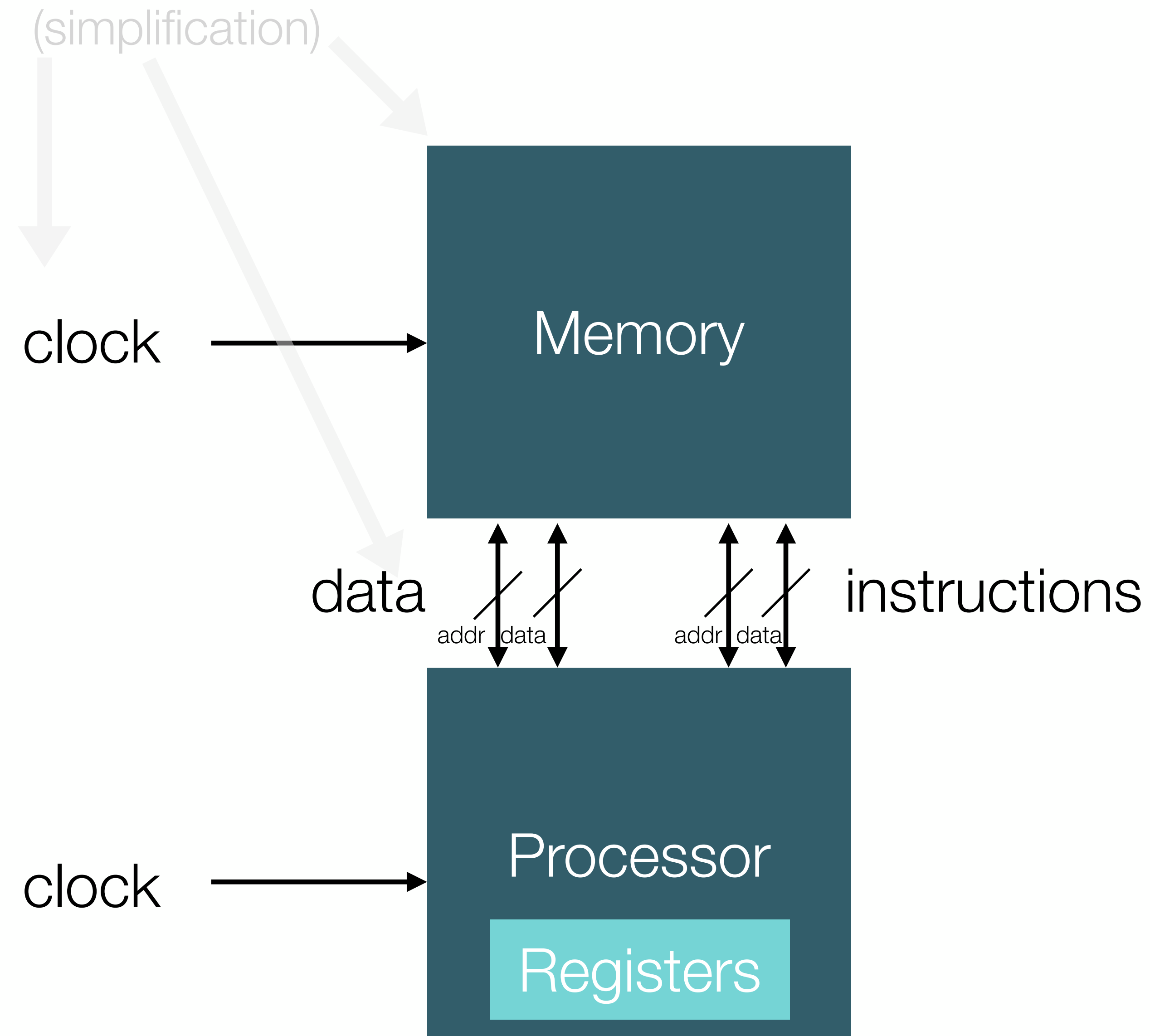
(We will talk about the interface between the processor and memory later)

Processor Core

Processor-Memory Interface

Processor executes instructions

Another example of microarchitectural property: Interface between processor core and memory. This example shows separate buses for data and instructions.



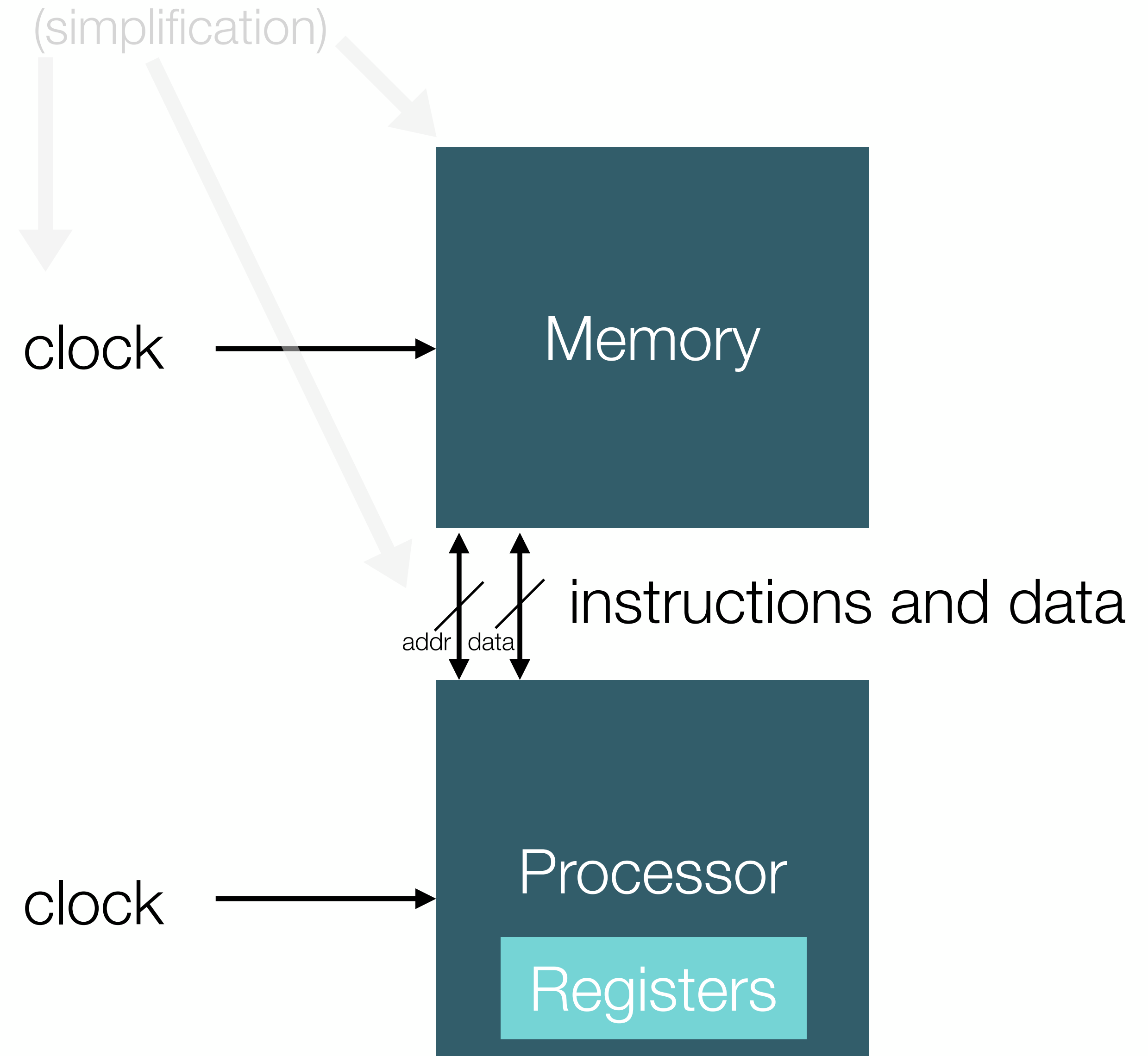
(We won't talk about other I/O interfaces for now)

Processor Core

Processor-Memory Interface

Processor executes instructions

Another example of microarchitectural property: Interface between processor core and memory. In implementing a processor, the processor designer chooses which memory microarchitecture would be better for her target design objectives. This example shows a single (unified) bus for both instructions and data.



Processor-Memory Microarchitectures

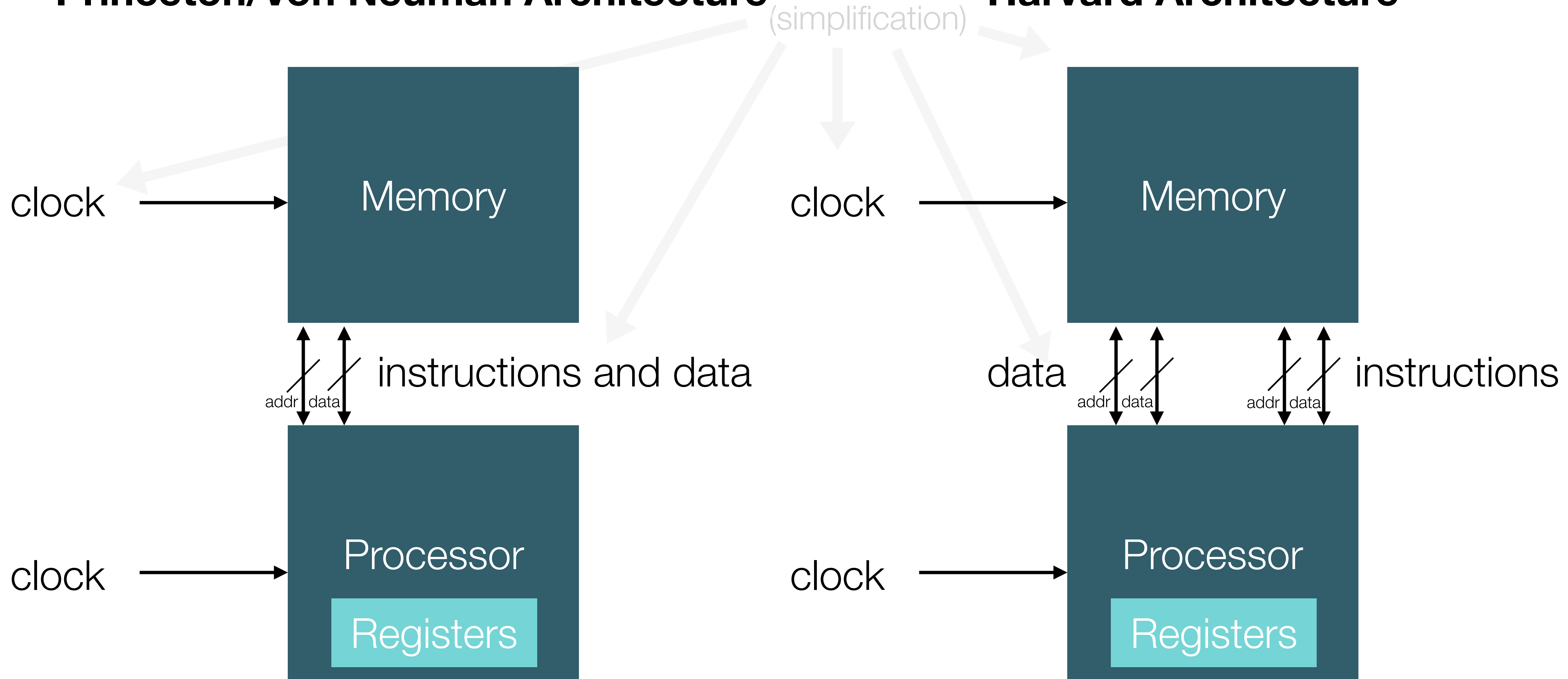
9

15

Historically, the following two memory interface microarchitectures have different names:

Princeton/Von Neuman Architecture

Harvard Architecture



Instruction Set Architectures (ISAs)

10

15

The ISA is the most important interface in a computing system

The boundary of architecture versus microarchitecture is well defined:

Everything visible from the instruction set architecture (**ISA**) such as **registers**, is part of the architecture. Everything else (under the hood¹) is part of the microarchitecture.

¹ That is, “under the bonnet” in the UK

Components of a Microarchitecture

11

15

Five components needed in any microarchitecture:

Logic to retrieve (fetch) instructions from memory

Logic to decode instructions

Logic to execute the instructions once we know what they are

Logic to access memory if the instructions access memory

Logic to access registers (most instructions affect register values)

These components also form the natural boundary for stages in a simple pipeline (Fascicle 5)

► There are often additional hardware structures (branch predictors, caches, reorder buffers, reservation stations, etc.) to make these steps more efficient. Since it is separated from programs via the ISA, a microarchitecture is free to implement whatever it wishes provided it honors ISA

The RISC-V Architecture

12
15

32 Registers: **x0** to **x31** in addition to a dedicated program counter register (**pc**)

Each of the registers has a pseudonym. More on that later.

We will focus only on the smallest subset of RISC-V, known as RV32I

- ▶ 32-bit data path
- ▶ Only integer instructions (no floating-point)
- ▶ No multiply or divide instructions
- ▶ No vector instructions

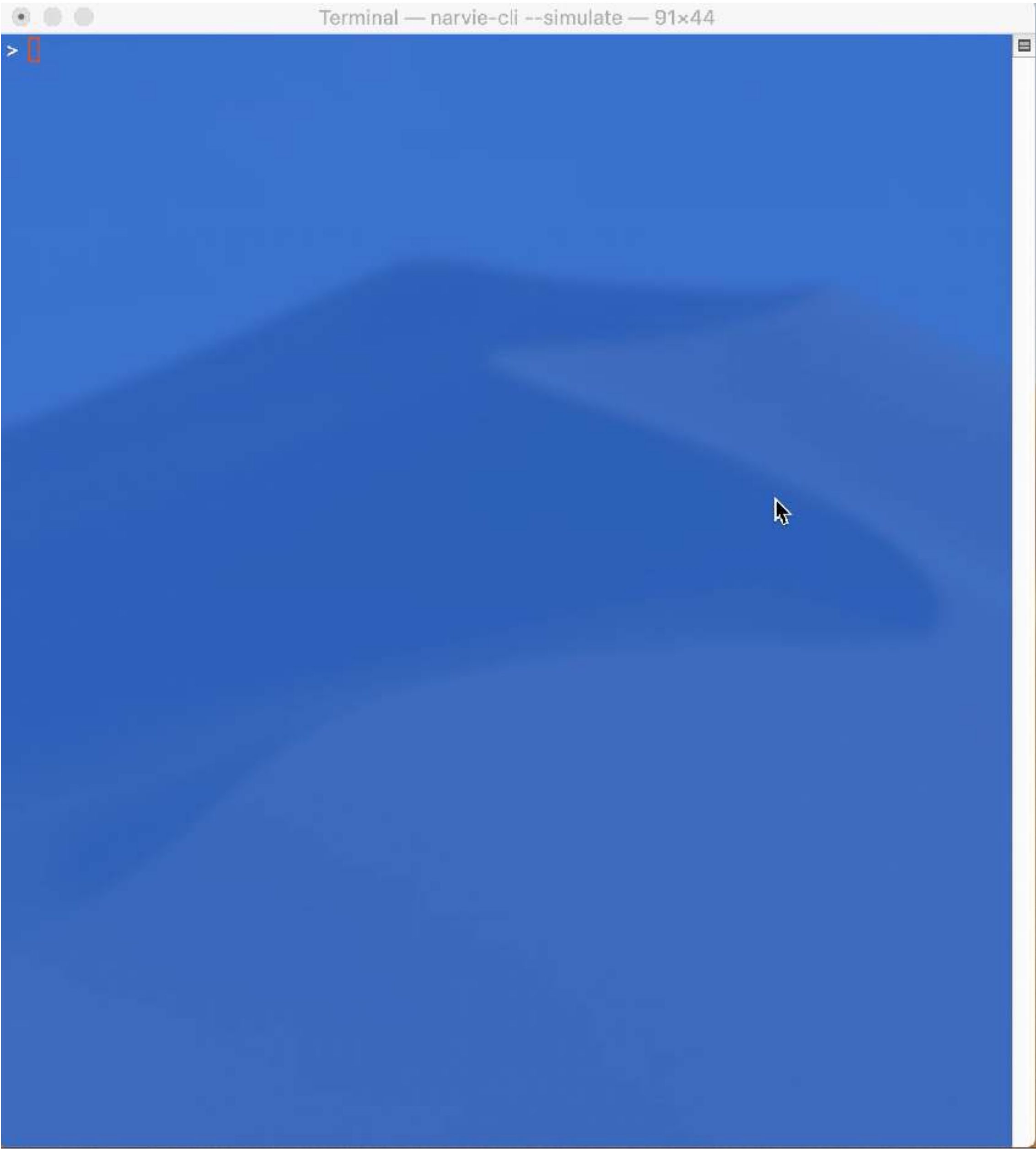
RV32I has 47 instructions:

add addi sub subi sll slli sra srai srl srli be bne fence
and andi or ori xor xori lui auipc slt slti sltiu fence.i
bge bgeu blt bltu jal jalr lb lh lw sb sh sw lbu lhu ebreak
csrrw csrrs csrrc csrrwi csrrsi csrrci ecall

(That's it!)

Example Using Narvie (narvie-cli --simulate)

13
15



nop

li

add

add

t0, 0x10

t1, t0, x0

t1, t1, t1

Does nothing

Load immediate of value 16 into t0

Copies t0 to t1, since x0 is always 0

Double t1

operation

destination register

source register 1

source register 2

Start Poking Around the RV32I Implementation

14

15

From the root of the course repository:

```
$ cd verilog/hardware/processor/  
$ make
```

(The RV32I processor implementation is in the subdirectory `sail-core`)

Things to Do

15

15

❶ Drop off “**muddiest point**” sheets (pages 20, 26, 31, 34, 81) anytime this week

❷ **Read the article** “Understanding some simple processor performance limits” by P. Emma (<docs/ussppl-journal-paper.pdf>)

As you read it, think about: (1) what is a “benchmark”? (2) what is a “basic block”? (3) What are the three kinds of data dependencies? (4) what is one way in which the statement “event frequencies being independent of microarchitecture” could be false?

Team Member 1

Write a RISC-V assembly program to loop through a range of addresses and compute the arithmetic mean

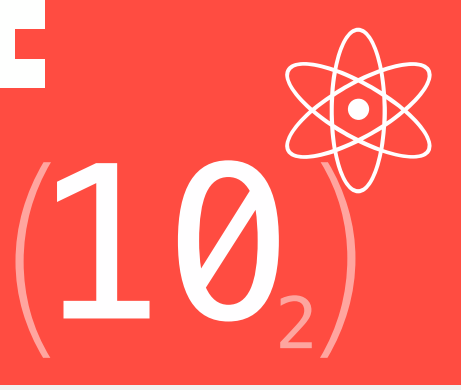
Team Member 2

Write a RISC-V assembly program to loop through a range of addresses and compute the variance. Assume register R30 contains the mean

Team Member 3

Write a RISC-V assembly program to loop through a range of addresses and compute the coefficient of variation. Assume register R30 contains the variance

(Remember, RV32I has no multiply or divide and no floating-point)



University of Cambridge
Department of Engineering

**Physical
Computation
Laboratory**