Closing the Loop on Basic Processor Design

Reminder! HW1 due Friday, HW2 released Wednesday, Check-In 3 on Friday

Image Credit: https://en.wikipedia.org/wiki/IBM_System/360_Model_91

Image Credit: https://www.computer.org/profiles/robert-tomasulo

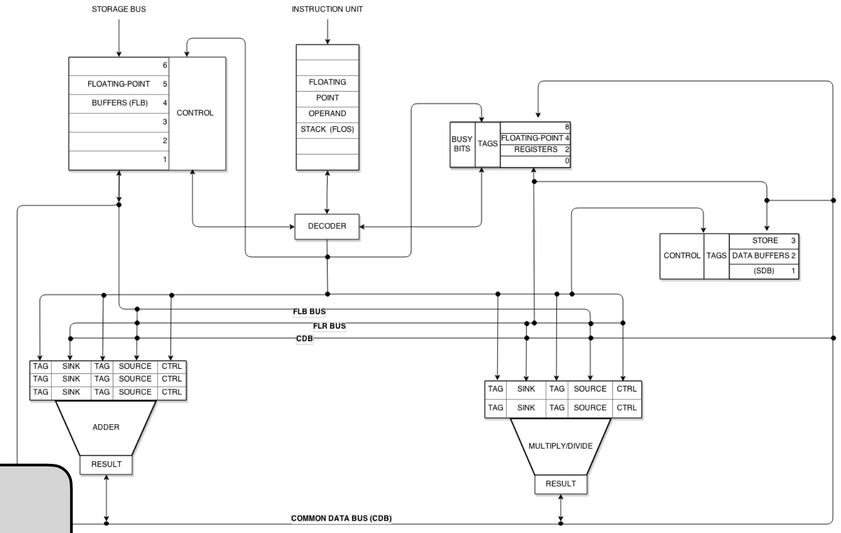
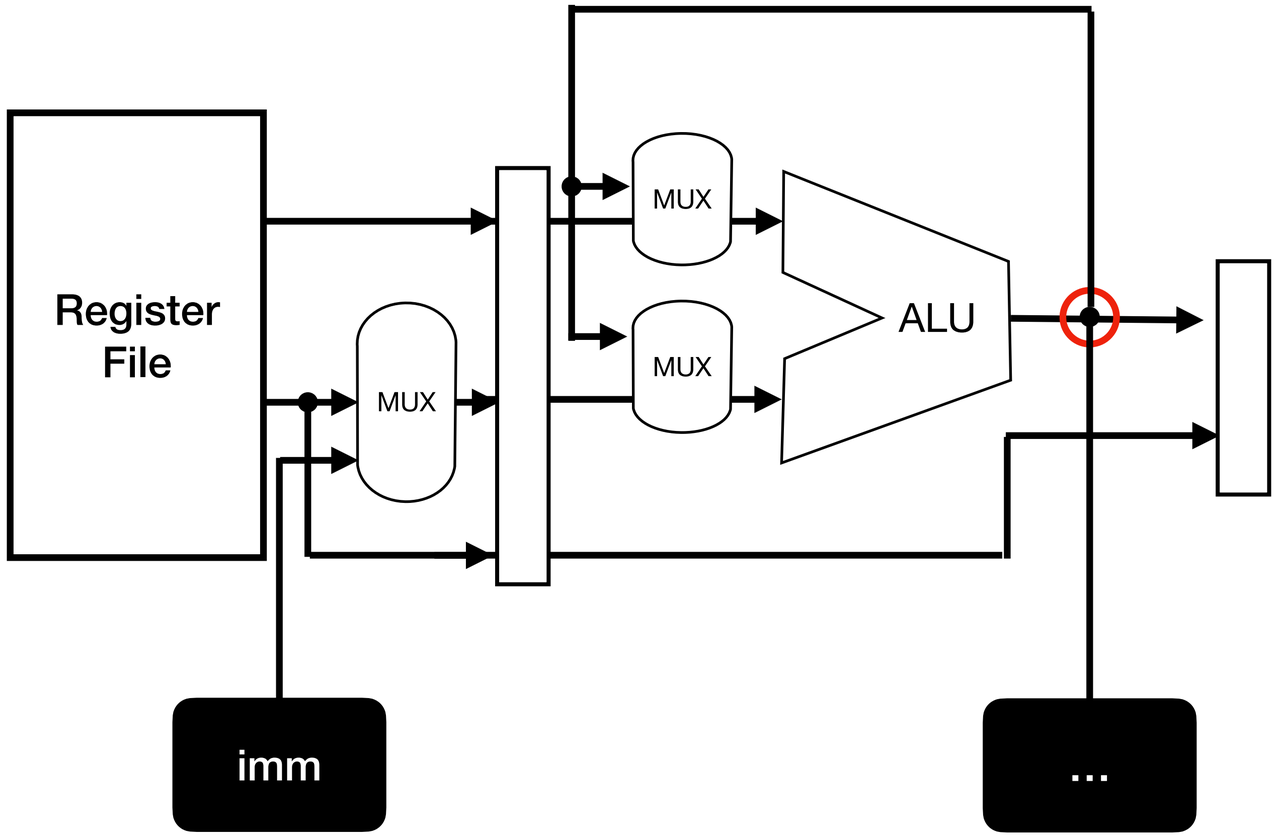


Image Credit: https://en.wikipedia.org/wiki/Tomasulo%27s_algorithm

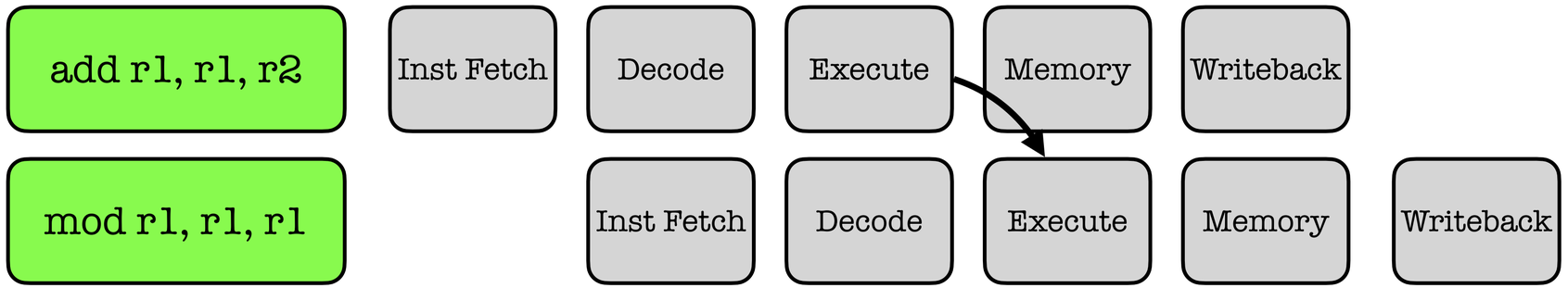
Outline

- False dependencies and the pitfalls of hazard checking
- Summarizing discussion on basic processor design
- Readdressing assumptions and open questions

Forwarding Processor

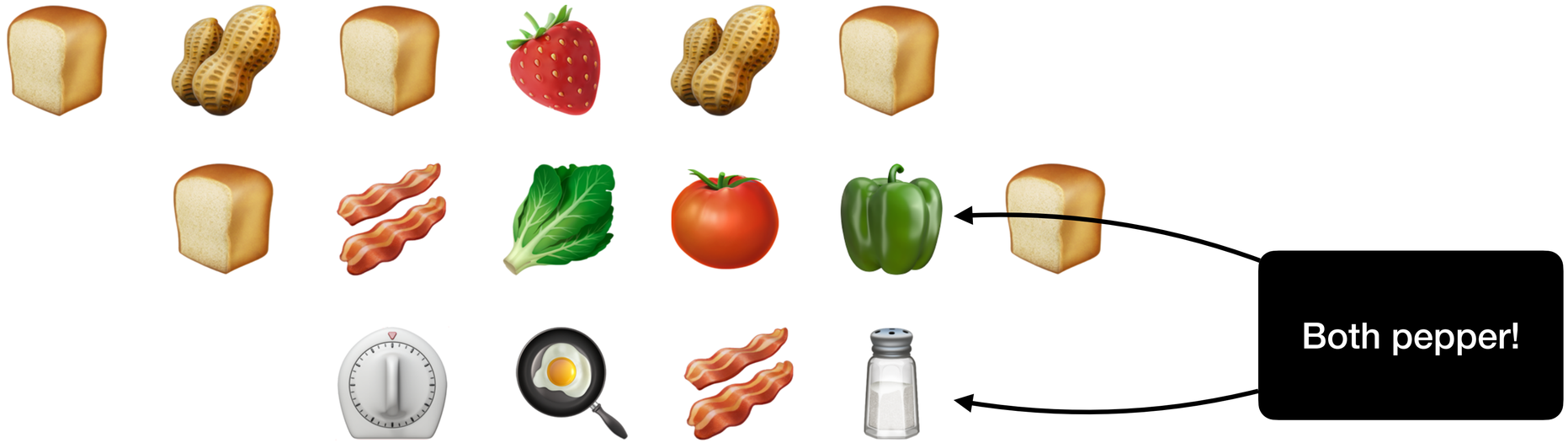


Benefits of Forwarding



No need to stall in this case! Result exists elsewhere in the data path and can be communicated to needed location!

Chat with your neighbor(s)!



Suppose we have a program in which two instructions refer to the same register name, but themselves refer to different variables. Would a hazard checking unit view these instructions as dependent?

False Dependencies

```
int a = 100;
int b = 100;
int c = 100;
int d = 100;
int e = 100;
int f = 100;
int g = 100;
int i = 100;
```

```
ldi r1 100
ldi r2 100
ldi r3 100
ldi r4 100
ldi r5 100
ldi r6 100
ldi r7 100
ldi r1 100
```

Kind of like having two ingredients named pepper!

Takeaways

- Hazards are a natural byproduct of pipelining a task
- Processor hazards can be classified as structural, data, or control
- Structural hazards must be resolved in the design of the data path
- Data hazards can be addressed in software, but hardware must be able to account for any hazard
- Sometimes software produces false dependencies! Hardware will handle instructions as if they are dependent, but in practice they are not… bad news!

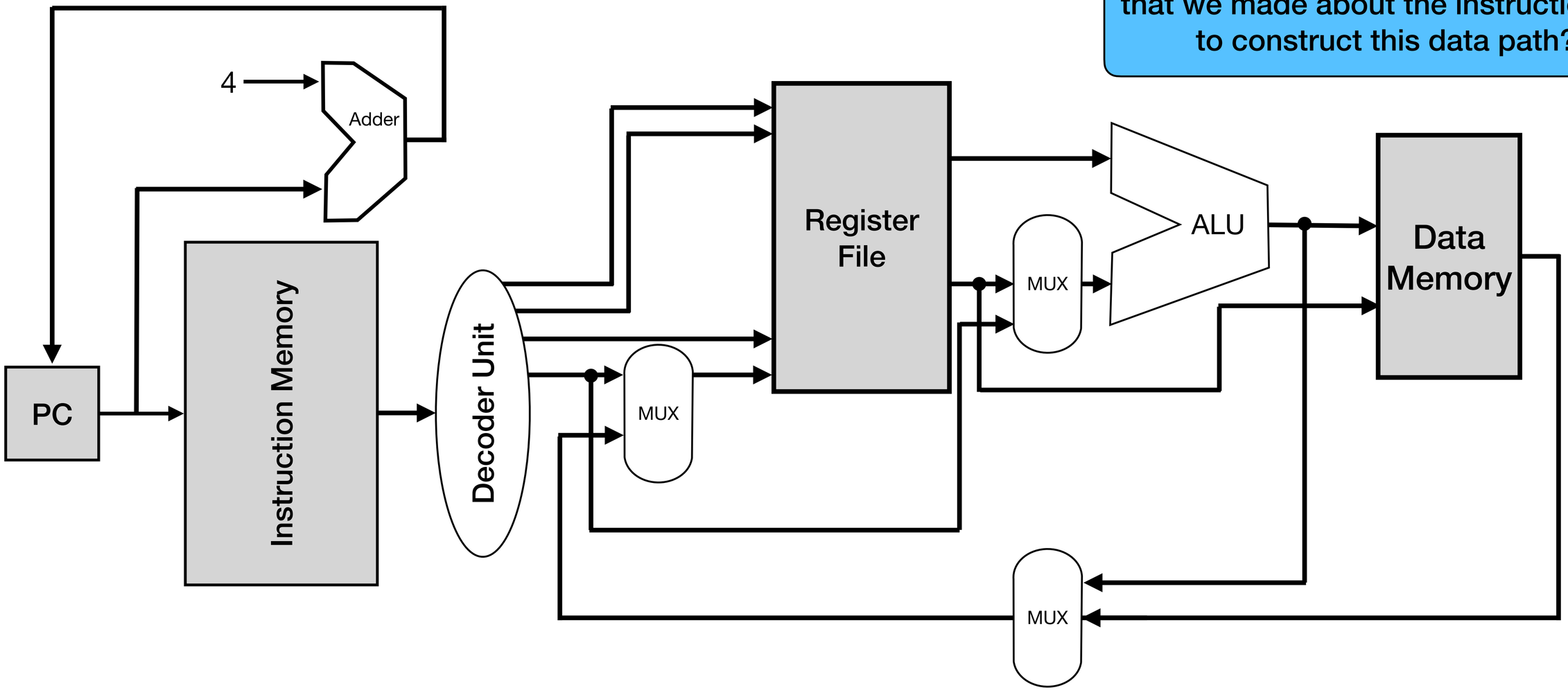
Data Path Construction

- We needed to build connections between components to implement all instructions in the ISA
- The more complicated the ISA, the more complicated the data path
- Our modeled ISA was perhaps overly simplistic…

Basic Processor Data Path

Chat with your neighbor(s)!

What are some simplifying assumptions that we made about the instruction set to construct this data path?



Processor Control

- In the single cycle data path, control fields from an instruction were communicated directly from the decoder unit to the appropriate components directly
- In a pipelined processor, the control fields from an instruction are communicated between pipeline registers so that each stage has the relevant control information to execute the instruction at that stage

Control Instructions

- In practice, we will need to have a more robust protocol to update our PC state to handle dynamic control through the program!
- Topics still to come when we revisit processor design…
  - Branch prediction
  - Transient execution
  - Speculative execution attacks!

Multiprocessors

- In practice, there is more than one processor to execute one or more programs in parallel!
- Different processors may exhibit different characteristics
- Coordination of “multithreaded” processes coordinated by the operating system and the processor hardware

```
(base) stum2025@STUM2025-MAC29 work-log % sysctl -a | grep -e perflevel -e cpu
kern.sched_rt_avoid_cpu0: 0
kern.cpu_checkin_interval: 5000
hw.ncpu: 10
hw.activecpu: 10
hw.perflevel0.physicalcpu: 4
hw.perflevel0.physicalcpu_max: 4
hw.perflevel0.logicalcpu: 4
hw.perflevel0.logicalcpu_max: 4
hw.perflevel0.l1icachesize: 196608
hw.perflevel0.l1dcachesize: 131072
hw.perflevel0.l2cachesize: 16777216
hw.perflevel0.cpusperl2: 4
hw.perflevel0.name: Performance
hw.perflevel1.physicalcpu: 6
hw.perflevel1.physicalcpu_max: 6
hw.perflevel1.logicalcpu: 6
hw.perflevel1.logicalcpu_max: 6
hw.perflevel1.l1icachesize: 131072
hw.perflevel1.l1dcachesize: 65536
hw.perflevel1.l2cachesize: 4194304
hw.perflevel1.cpusperl2: 6
hw.perflevel1.name: Efficiency
hw.physicalcpu: 10
hw.physicalcpu_max: 10
hw.logicalcpu: 10
hw.logicalcpu_max: 10
hw.cputype: 16777228
hw.cpusubtype: 2
hw.cpu64bit_capable: 1
hw.cpufamily: 1867590060
hw.cpusubfamily: 2
hw.nperflevels: 2
machdep.cpu.cores_per_package: 10
machdep.cpu.core_count: 10
machdep.cpu.logical_per_package: 10
machdep.cpu.thread_count: 10
machdep.cpu.brand_string: Apple M4
```

Other Advanced Processor Features

- We can start multiple instructions at the same cycle with a *multi-issue* processor, which will increase the likelihood of completing an instruction at every cycle
- Tomasulo's Algorithm allows processors to think about maintaining multiple "reservation stations" of components, and dynamically assigning instructions to these reservation stations (basis of *out-of-order* execution)
- A single instruction can correspond to multiple steps of execution, sometimes via *very large instruction word* (VLIW) processor

Memory Operations

- “Instruction memory” and “data memory” are not single cycle components… Instead, they are members of a whole memory system!
- If memory operations (instruction fetch, data memory) can take multiple cycles to execute, our processor needs to be able to handle these longer operations!
- As when describing data hazards, all memory systems need to have well defined “consistency policies” (i.e., the rules that dictate the allowable behaviors for interacting with the same location in memory)
- We will talk about this more on Wednesday!