More Hazards and False Dependences

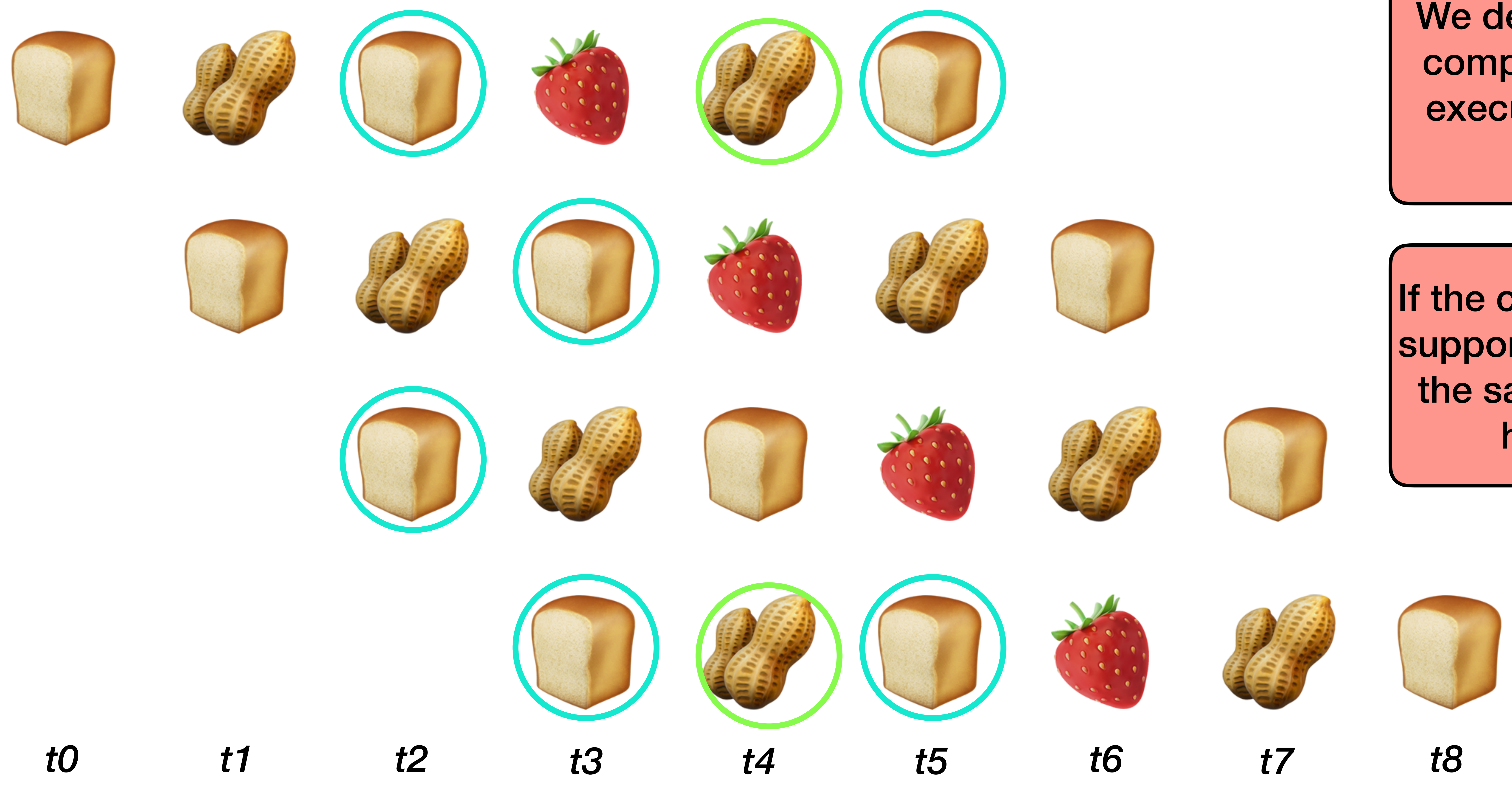
Outline

- Continuing discussion of hazards
- Formalizing and optimizing hazard checking
- The pitfall of hazard checking

(From Wed) Hazards in a Pipelined Processor

- Structural hazards: occurs when a hardware component cannot support the combination of instructions to execute within the same clock cycle
- Data hazards: occurs when there exists a **dependence** between two instructions in the pipeline, where the result from an incomplete instruction is relevant for the execution of another instruction
- Control hazards: occurs when a decision needs to be made about the next instruction to execute before that decision has been computed

(From Wed) Examples of a Structural Hazard



We depend on the same component for different executions at the same cycle!

If the component does not support multiple utilities in the same cycle, then we have a hazard!

(From Wed) Data Hazard Example

dest

src

Is this a data hazard?

No! No dependence between instructions. If all registers were switched…

```
lw r1, r2
lw r2, r3
lw r3, r4
lw r4, r5
```

```
add r1, r1, r2
mod r1, r1, r1
sub r2, r1, r2
div r2, r2, r2
```

Is this a data hazard?

Yes! Produced output are needed in subsequent instructions

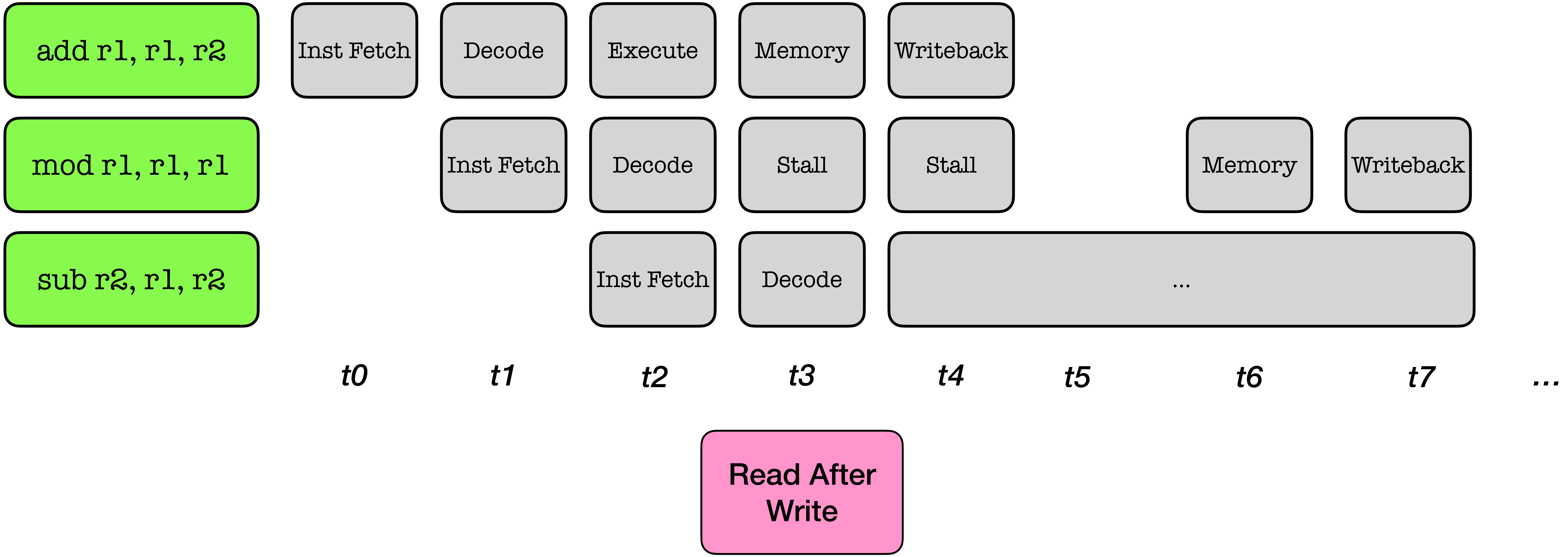
(From Wed) Resolving Data Hazards

- Option 1: Use the compiler to re-organize instructions! This will allow many hazards to resolve themselves before execution
- Option 2: “Bubble” or stall the pipeline when a hazard is detected to wait for the
- Option 3: Forwarding data from one location in the pipeline to another stage that needs the data

```
add r1, r1, r2
```

More instructions here!

(From Wed) Resolving Data Hazards (Bubble)



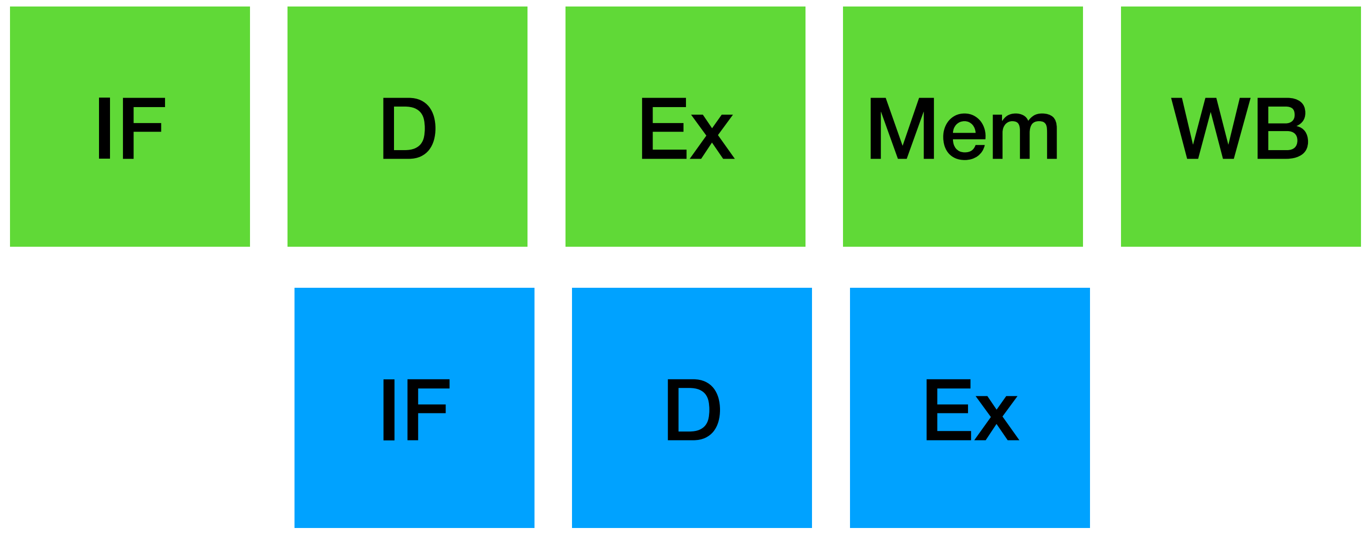
Hazard Checking Unit

- To detect if there is a data hazard, the processor needs to deploy a *hazard checking unit*
- The hazard checking unit examines the state of the *pipeline registers* to look for inter-instruction dependencies!
- If such a dependence exists, then the processor needs to signal the appropriate components to “do nothing” until the hazard is resolved — only at this point should the signals be reset with the next steps of execution
- This means components may be highly under-utilized!

Other Types of Data Hazards

```
int a = *addr1;
a = 100;
```

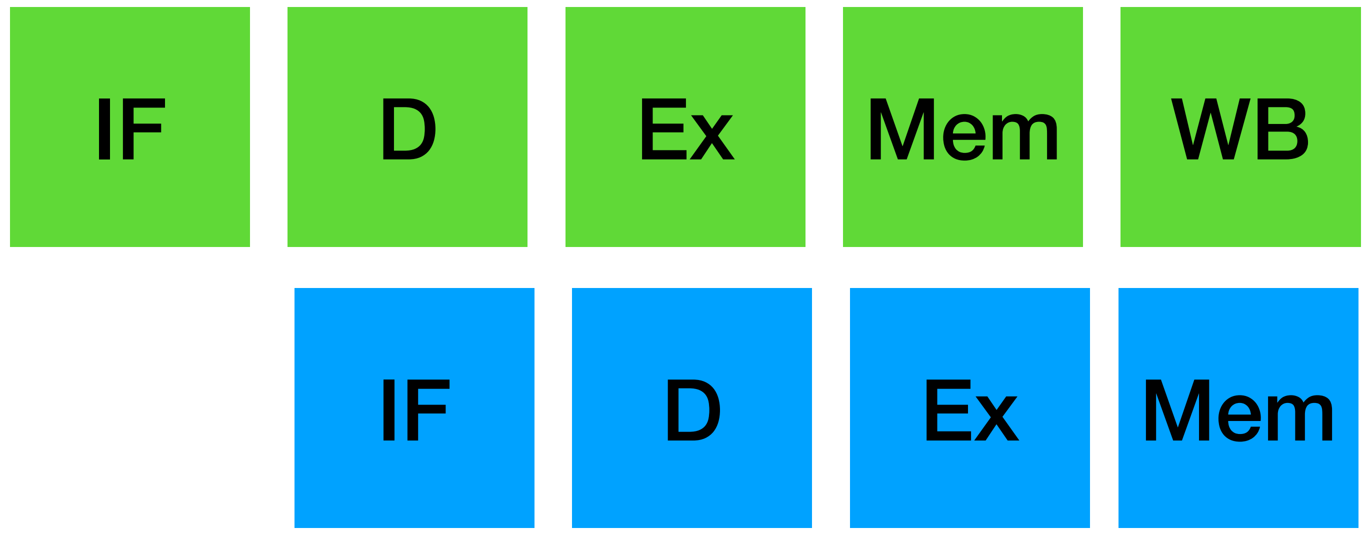
```
lw r1, r2
ldi r1, 100
```



Write After Write

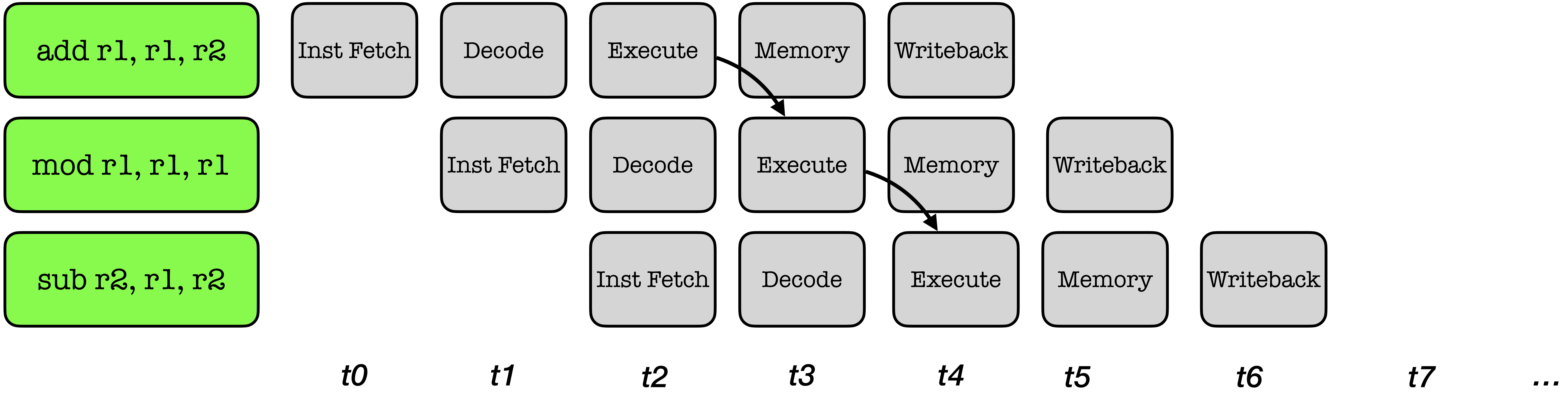
```
int a = *addr1;
*addr1 = 100;
```

```
lw r1, r2
sw r2, 100
```



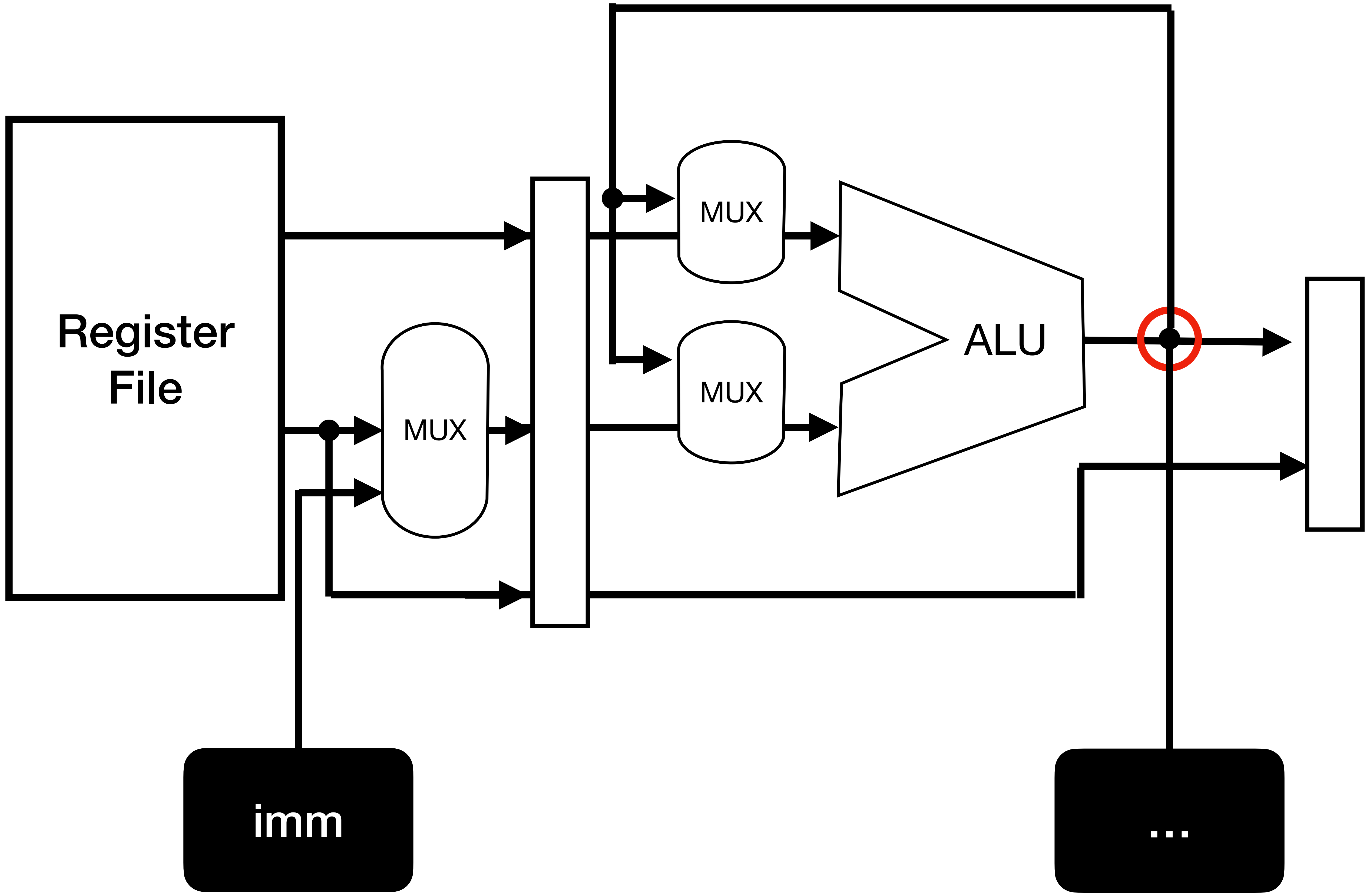
Write After Read

Resolving Data Hazards (Forwarding)



Forwarding Processor

Forwarding Processor



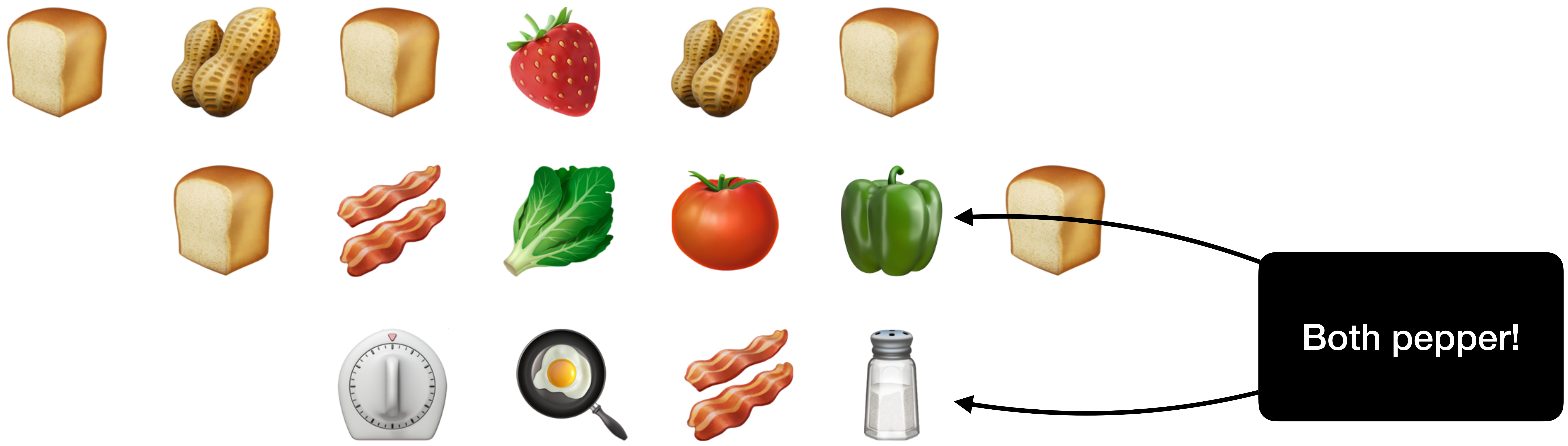
Chat with your neighbor(s)!

Where does the selection of the newly introduced multiplexors come from?

Hazard Checking Summary

- Structural hazards are resolved in the pipeline design
- Data hazards are resolved at runtime by either stalling or forwarding
- Control hazards are resolved by… stay tuned!

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!

Exit Ticket

https://forms.cloud.microsoft/r/7KVf6ex7f5