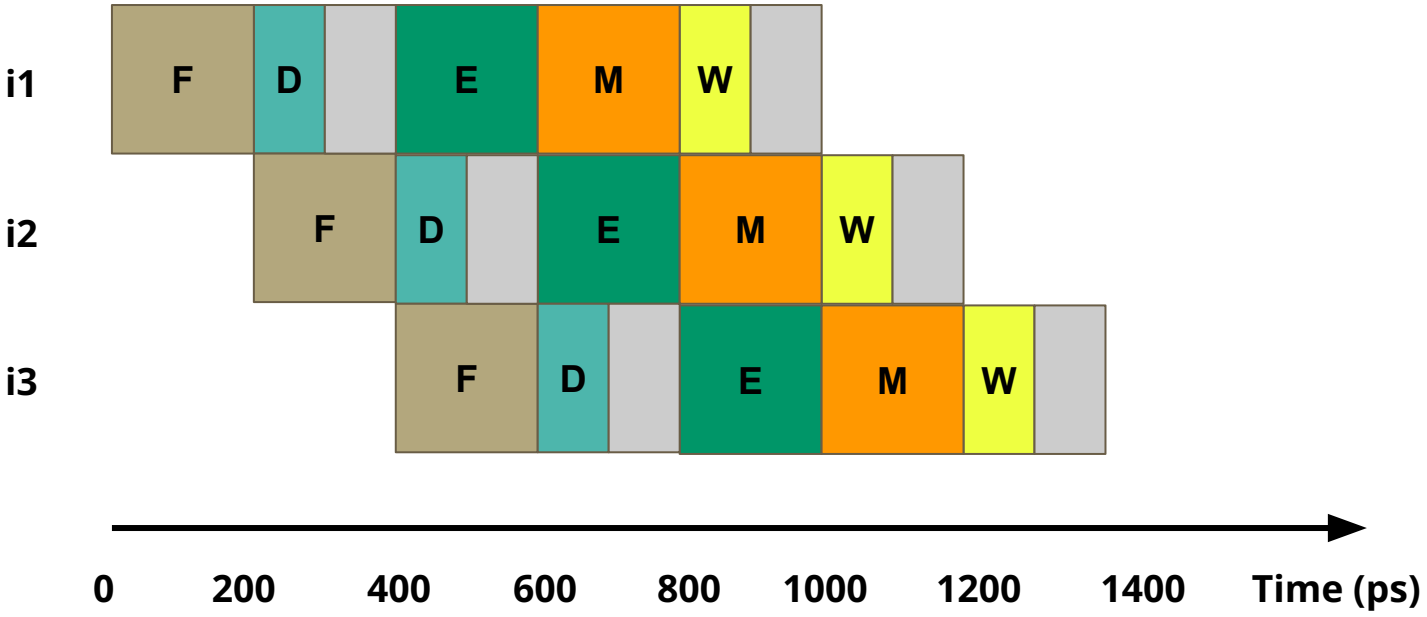
Bits of Architecture

Pipeline Hazards

Recall Pipelining

Execution Over Time

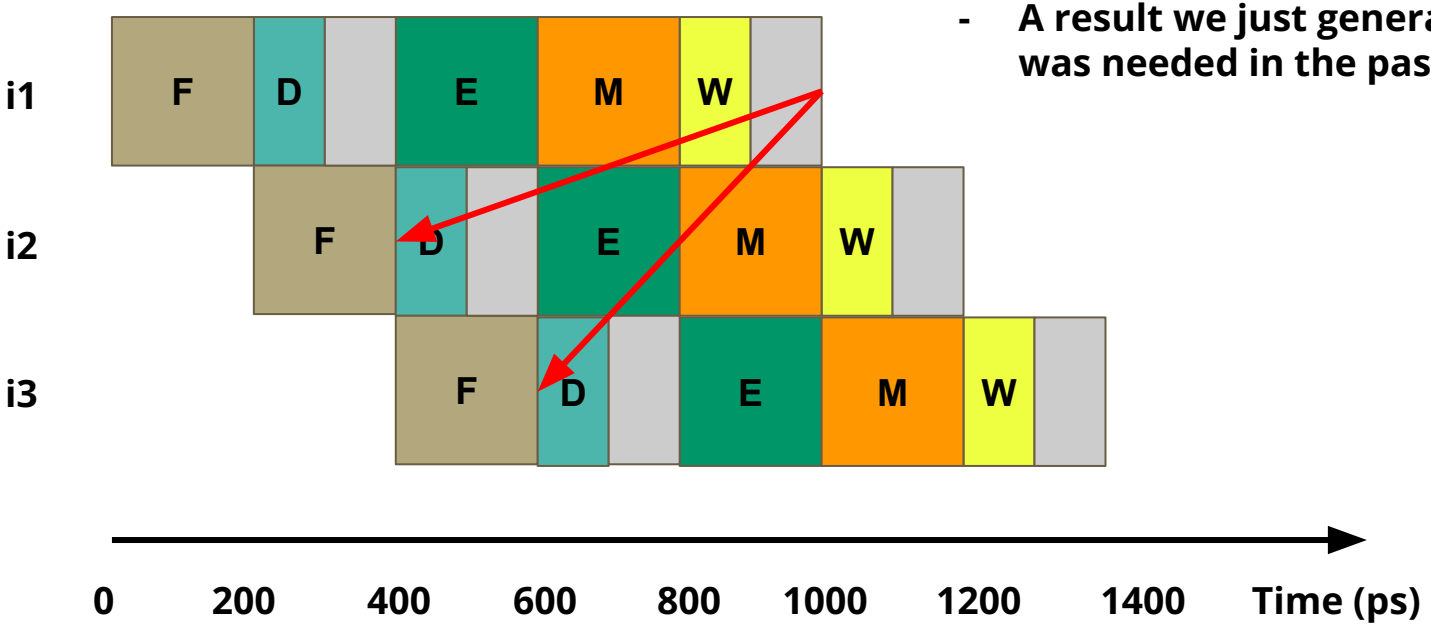


Dependencies

Dependencies

Arrows backward are bad!

- **“Time Travel”**
- **A result we just generated was needed in the past**



How Do We Classify Hazards

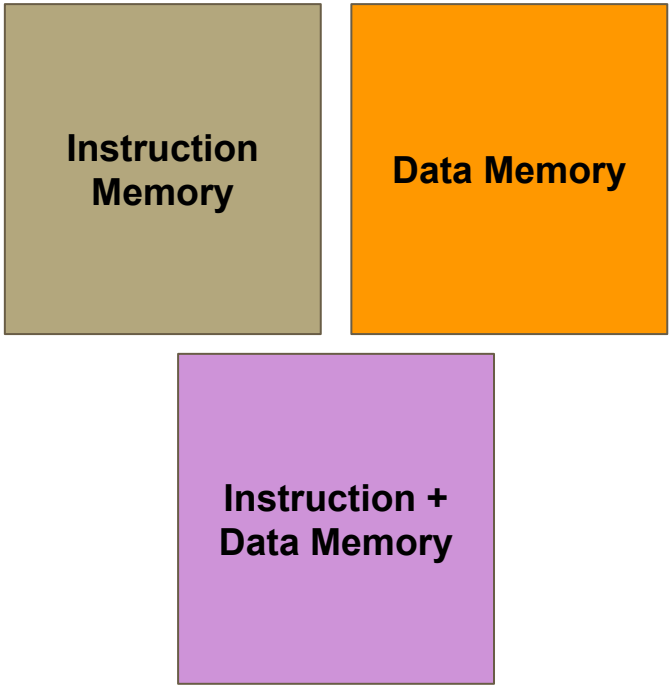
Hazards

- Structural Hazards
  - Multiple Instructions want to use the same hardware structure
- Data Hazards
  - An instruction depends on the result of another instruction
- Control Hazards
  - We can’t fetch the correct instruction in the right clock cycle (e.g., after a branch)

Structural Hazards

Structural Hazards

- Instruction and Data memory don’t have to be separate
- However, this could lead to a hazard
- Example
  - Insn. A is in fetch
  - Insn. B is doing a write
- Depends on the design of our structures



Data Hazards

Data Hazard

- Instructions can depend on the result of an earlier instructions
- Example
  - **i1** writes to **x2**
  - **i2** reads from **x2**
  - **i2** must wait for **i1** to write the result to the register **x2** before reading it

Example

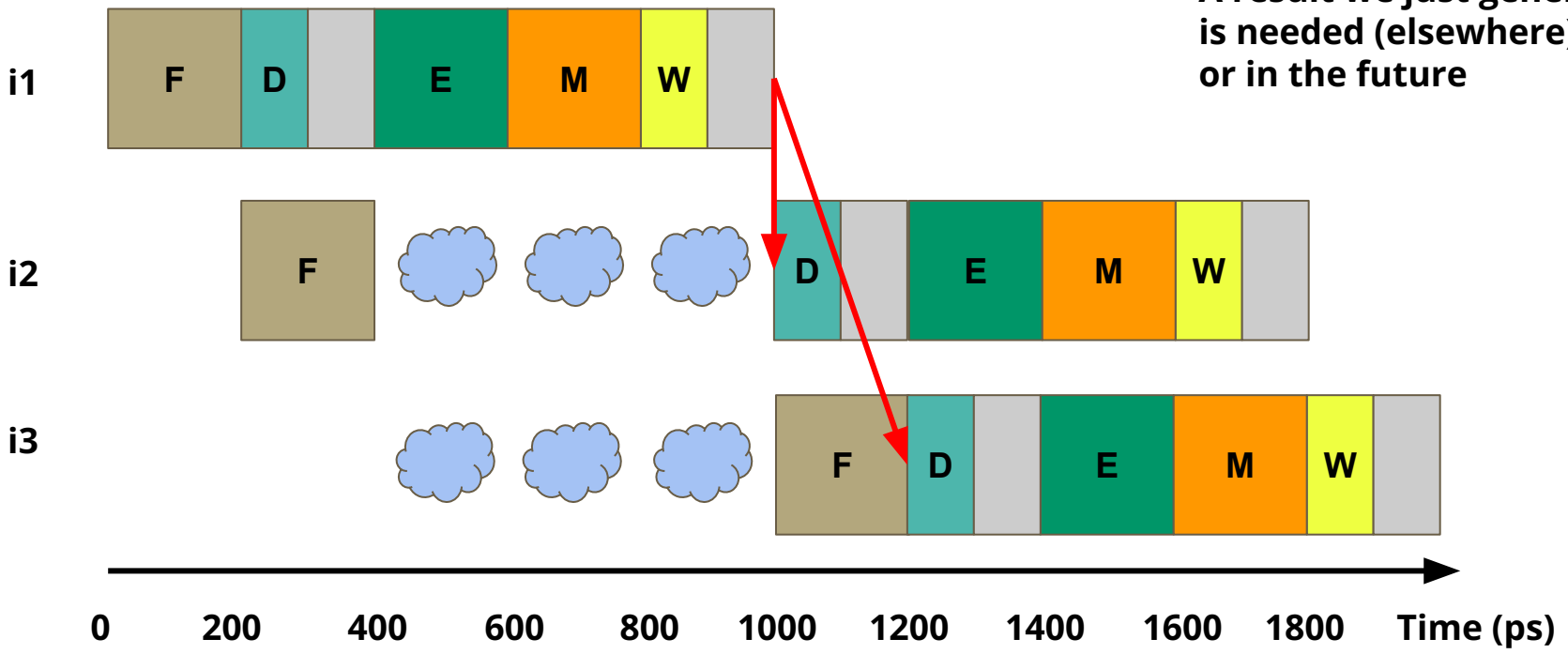
i1. add x2, x1, x0

i2. add x3, x2, x4

Execution Over Time

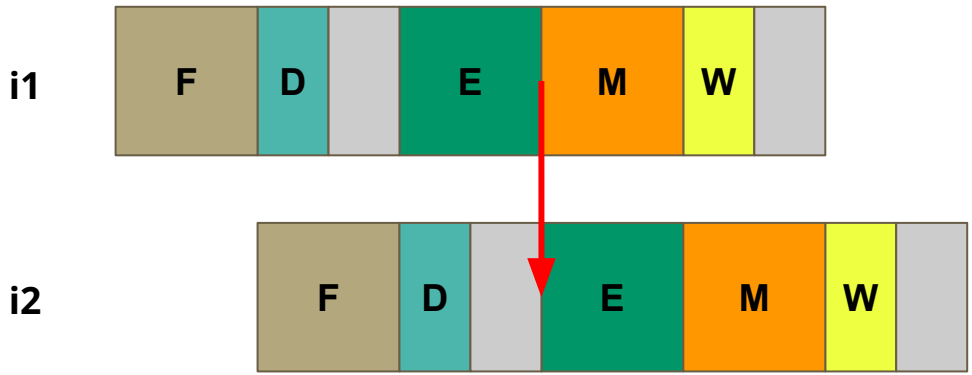
Arrows forward/down are OK!

- A result we just generated is needed (elsewhere) now or in the future



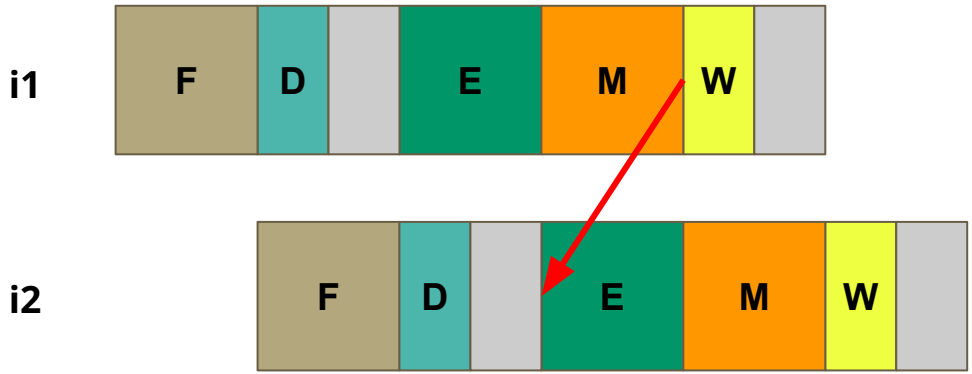
Forwarding/Bypassing

- We can add extra logic to bypass values to earlier stages in the pipeline
  - Feed ALU output back in as an input
- Eliminates stalls between R-Type instructions



Forwarding/Bypassing

- What about between Loads and R-Type?
  - Guaranteed stall!
- Can still forward to avoid some of the stall cycles
  - M->E



Control Hazards

Control Hazard

- Not every instruction is 4B away
  - Branches
- Example
  - **i1** does a conditional branch
  - What do we fetch for **i2** in the next clock cycle?
- Easy solution
  - Just stall!
  - Bad for performance...

Example

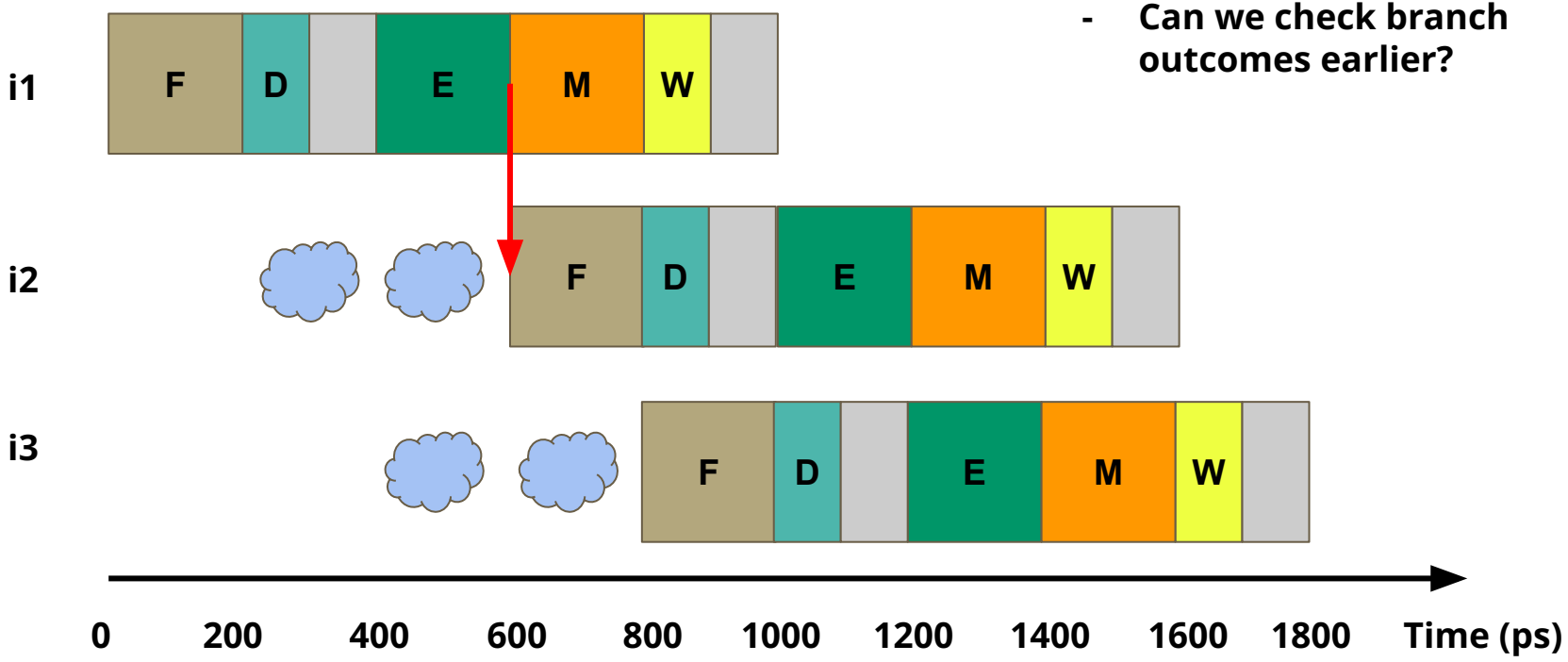
i1. beq x2, x1, 0x40

i2. ???

Execution Over Time

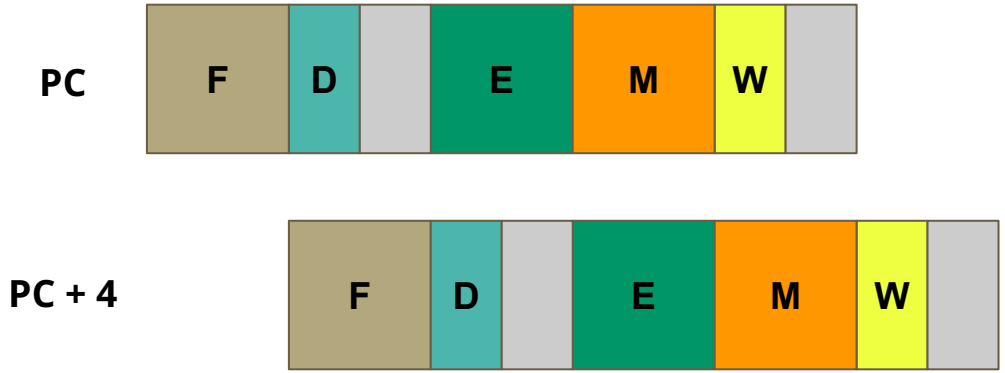
We don't know our branch result until after execute

- Can we check branch outcomes earlier?



Workarounds For Control Hazards

- Branch Prediction
  - Speculative Execution
- Choose a path to follow without knowing for certain it is correct
  - Must be able to recover from bad predictions



Workarounds For Control Hazards

- Simple strategy
  - Always predict not-taken for branches
- No Penalty if our prediction is correct!
- More complex branch prediction strategies
  - A topic for another time