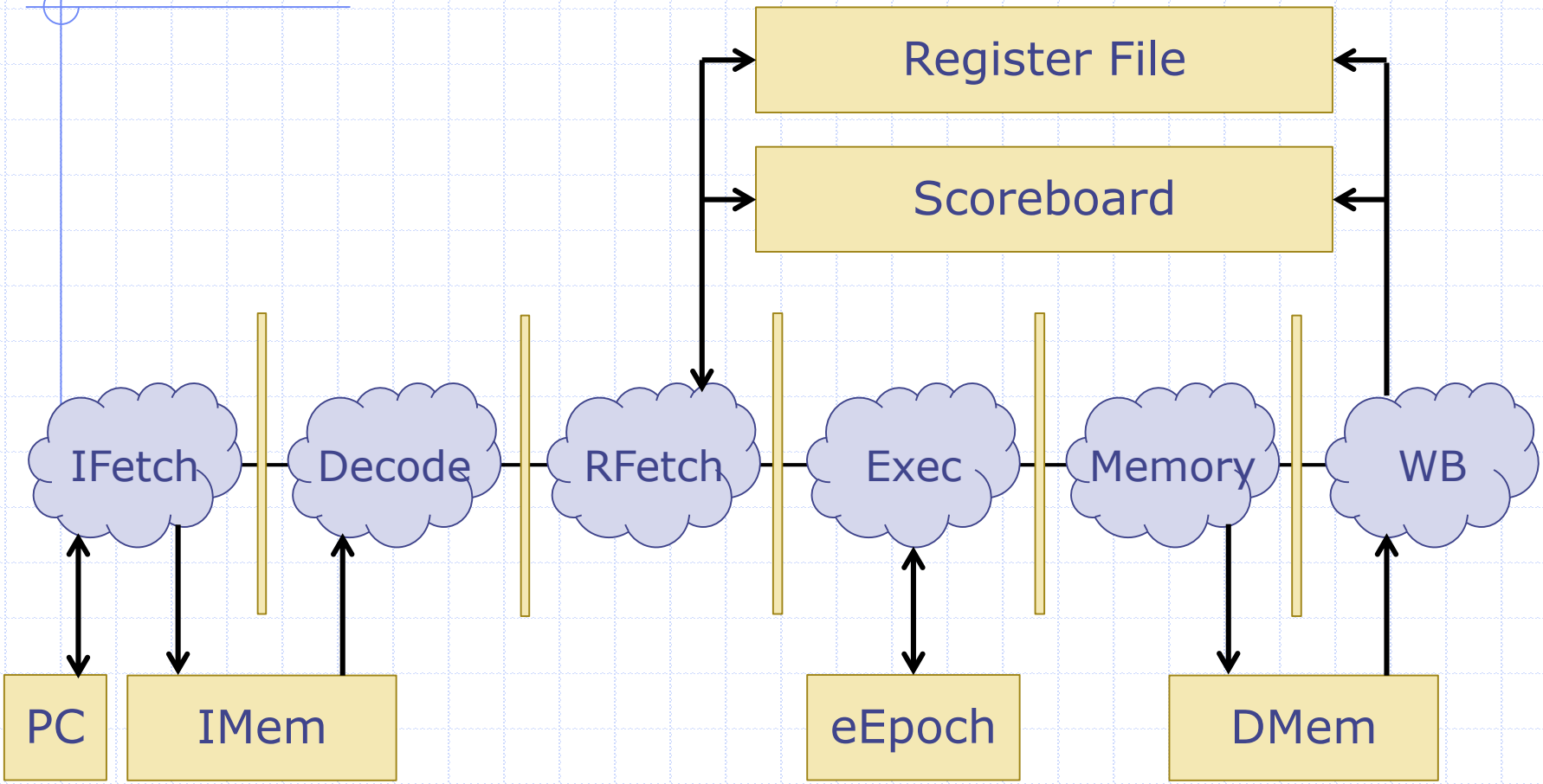
Constructive Computer Architecture

Tutorial 6: Discussion for lab6

Introduction

- Lab 6 involves creating a 6 stage pipelined processor from a 2 stage pipeline
  - This requires a lot of attention to architectural details of the processor, especially at the points of interaction between the stages.

6 stage pipeline



Bugs cheatseet

- My processor hangs:
  - Does the cpu starts?
  - Do you stall on the scoreboard?
  - Do you do an infinite loop, because there is an actual infinite loop?
  - Do you execute an illegal instruction?
  - Do you fetch forever the same pc?

How do you know?

- Your processor hangs. How do you know why?
  - Does the cpu starts?
  - Do you stall on the scoreboard?
  - Do you do an infinite loop, because there is an actual infinite loop?
  - Do you execute an illegal instruction?
  - Do you fetch forever the same pc?

My processor kind of work: it said PASSED.

- What does it mean?
  - What lead my host computer to print PASSED?
  - What lead my host computer to print FAILED?
- Remark: Your processor should also print a number of cycle that it took to do the test.

My processor kind of work: it said PASSED.

- The numbers for cycle should make sense:
  - What can you tell me if the number of cycle printer for simple is: “200200” and the number of instructions “101101”?

3 Details

- Processor State
- Poisoning Instructions
- ASAP Prediction Correction

Processor State

- The processor state is (PC, RFile, Mem)
- Instructions can be seen as functions of a processor state that return the new processor state
  - addi(PC, RFile, Mem) = (PC+4, RFile', Mem)
    - RFile' is RFile updated with the result of the addi instruction
- The instruction memory can be seen as a function of PC that returns Instructions
  - Imem: (PC) ->
    ((PC, Rfile, Mem) -> (PC, RFile, Mem))

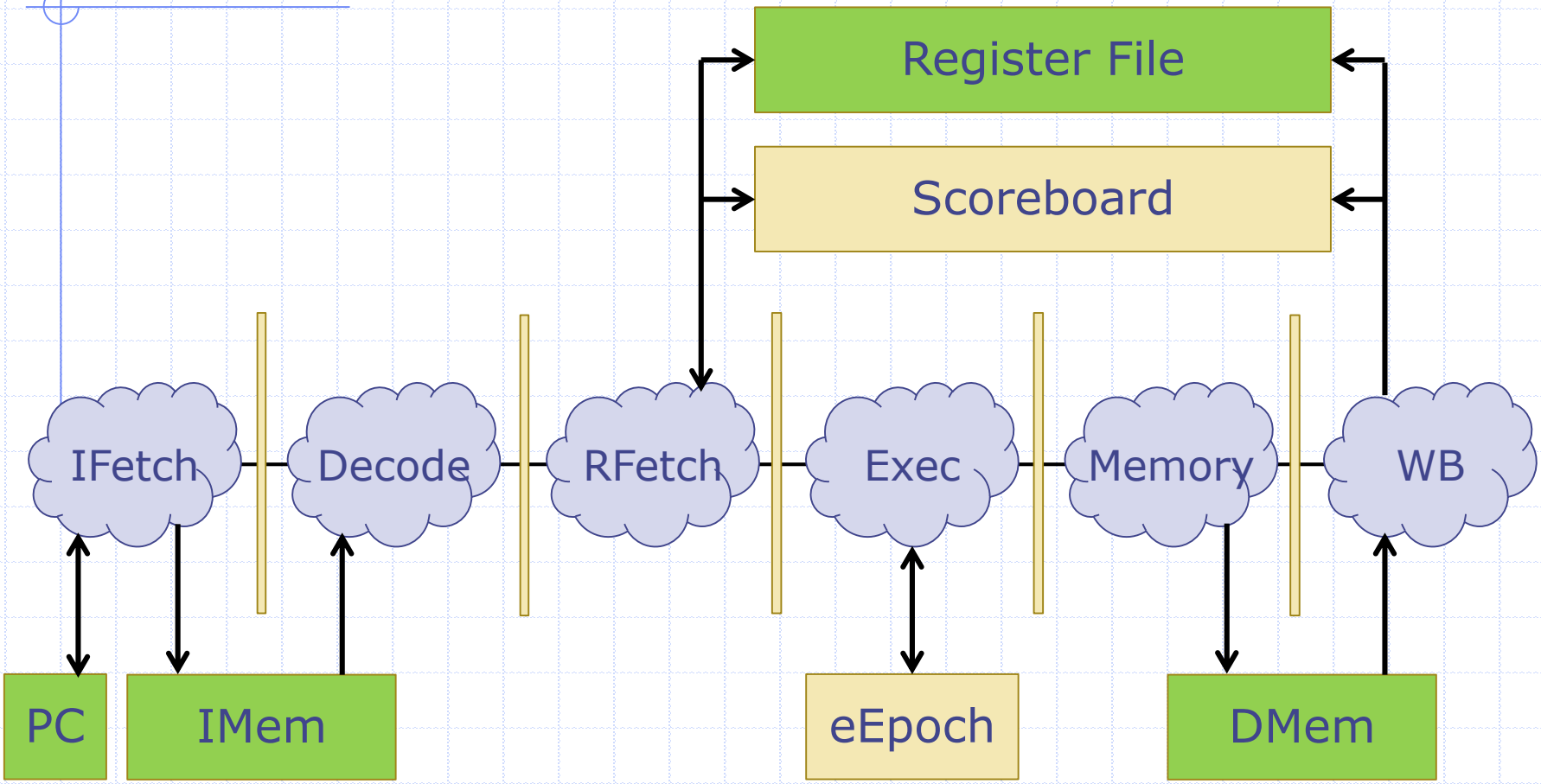
Processor State

- If your processor from lab is not working:
  - Was an instruction executed on the wrong processor state?
    - RAW hazards
    - Not using the right PC in the execute stage
  - Was the wrong instruction executed?
    - A wrong path instruction from branch misprediction updated the processor state
- How do I know what should happen?

What is the right trace?

- You have a reference processor from previous labs.
  - Run the code on the previous core, and look where it diverges.
- Is there a problem with that?
  - You don’t print the same stuff, they are not aligned cycle by cycle.

Processor State

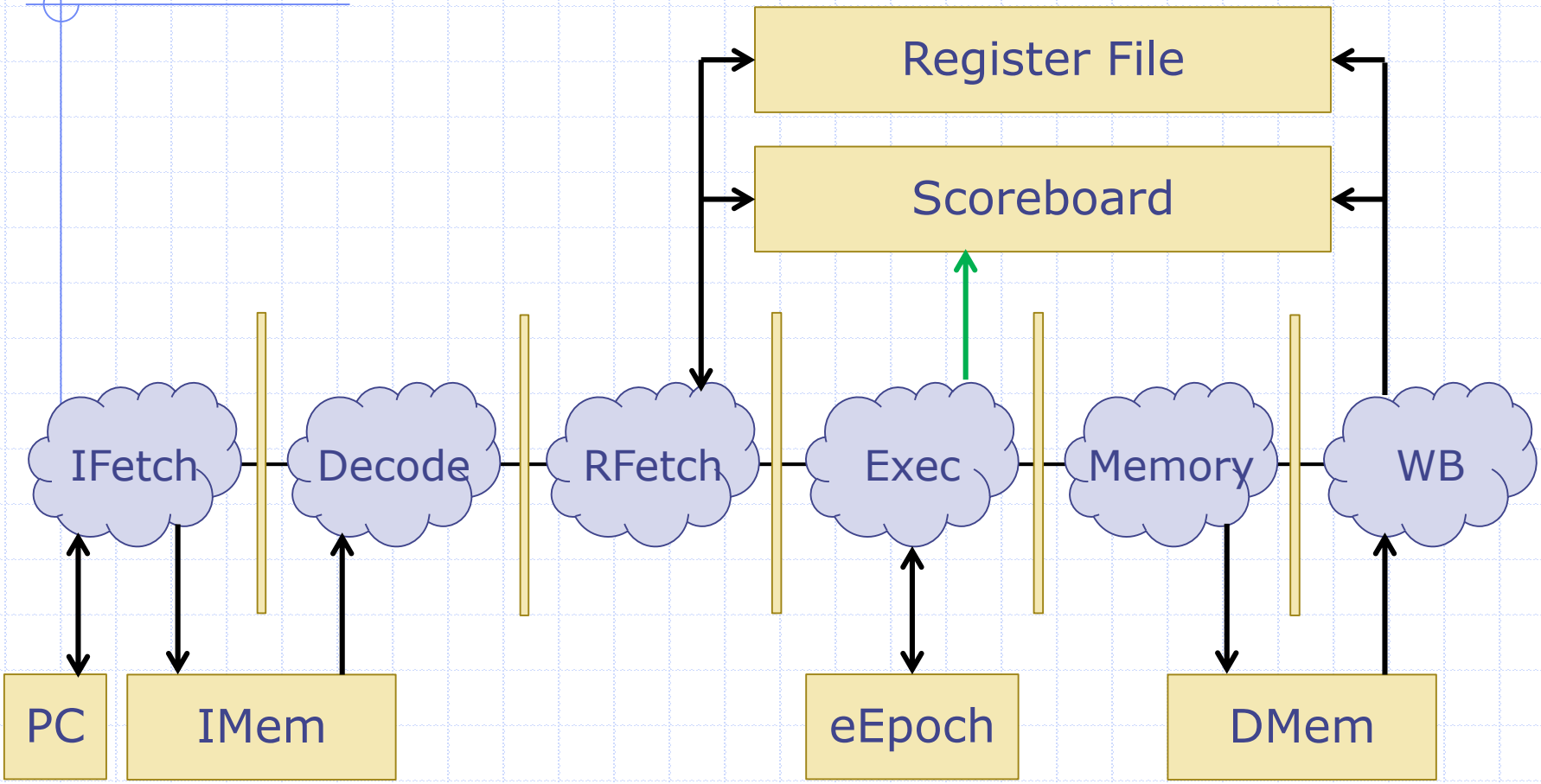


Green blocks make up the processor state. All other state elements make sure the right processor state is used to compute instructions, and to make sure the right instructions are executed.

Poisoning Instructions

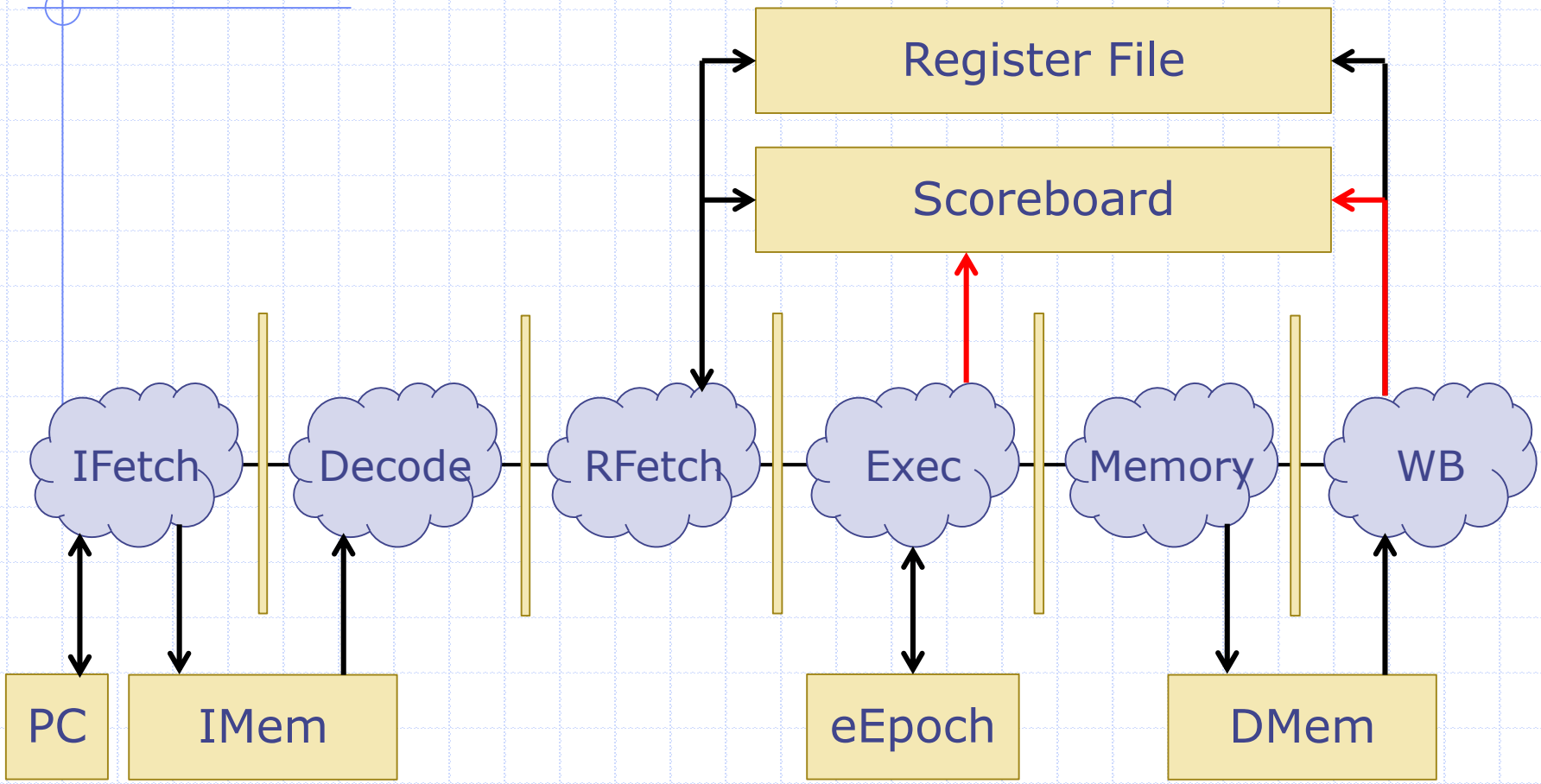
- Why poison? It's a way to mark that an instruction should be killed at a later stage.
  - This mark could be as simple as using an invalid value in a maybe data type
- Instructions are poisoned when epochs don't match
- Why not kill in place?

Kill-In-Place Pipeline



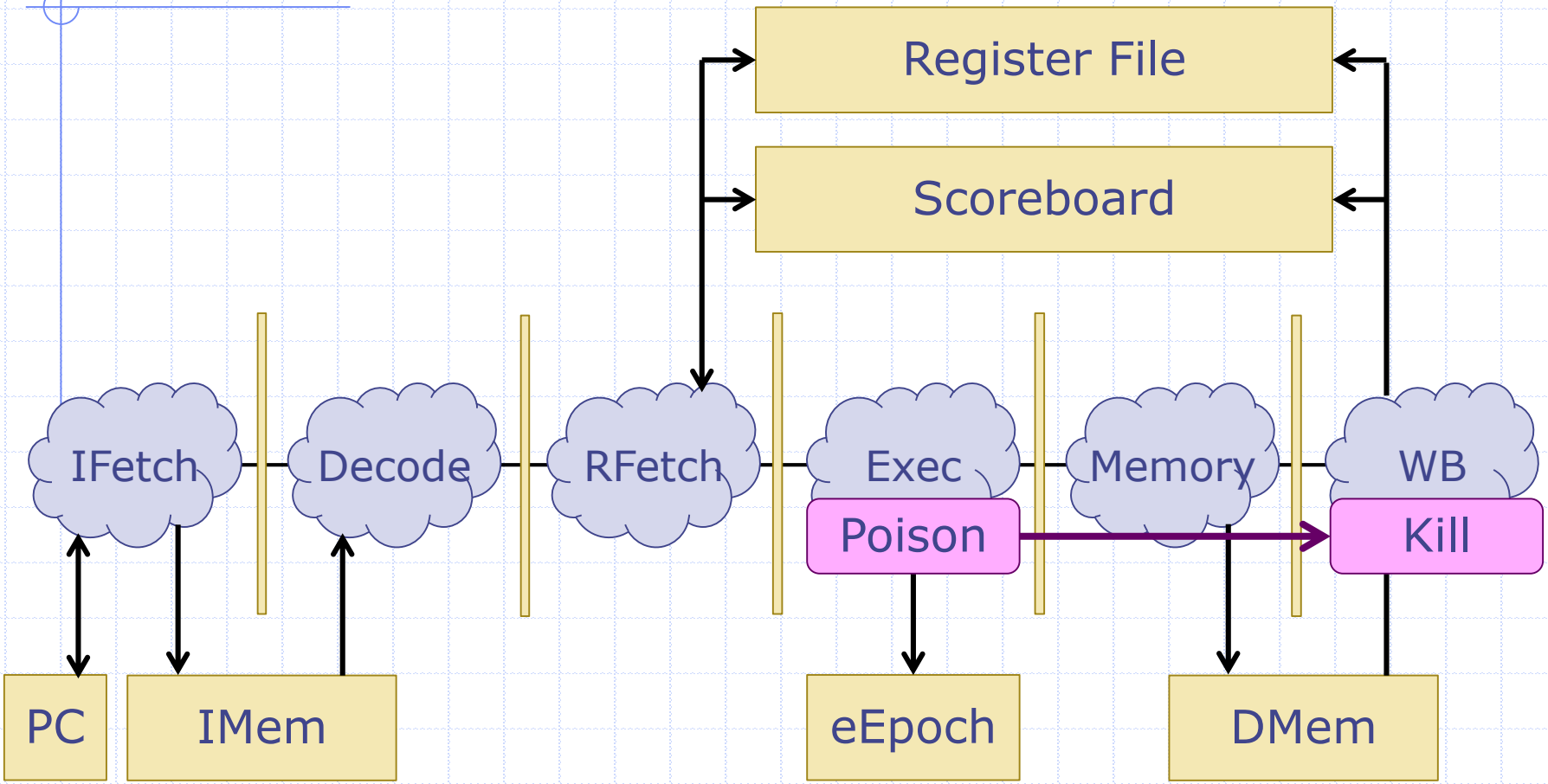
Scoreboard entries need to be removed when instructions are killed.

Kill-In-Place Pipeline



Both Exec and WB try to call sb.remove(). This will cause Exec to conflict with WB. Also, the scoreboard implementation doesn't allow out-of-order removal.

Poisoning Pipeline



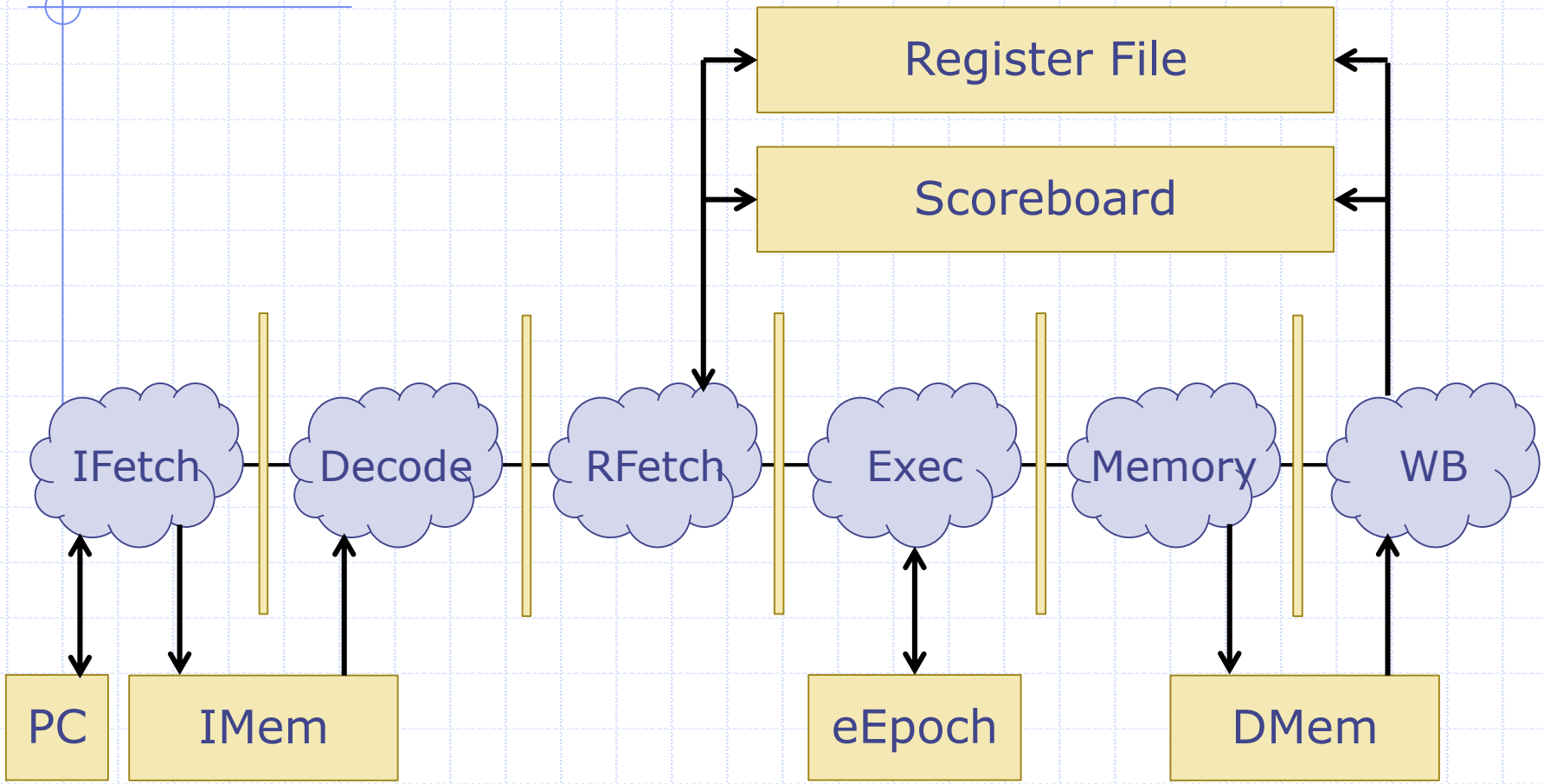
Register stalling error:

- Make sure you don’t drop instructions on the floor when you stall.
  - E.g you always dequeue from the fifo.

Performance: < 1 IPC

- How does the following parameters influence the performance:
  - Depth of the pipeline
  - Efficiency of the branch predictor
  - Number of branches/jump.
- Problem with firing several rule simultaneously.

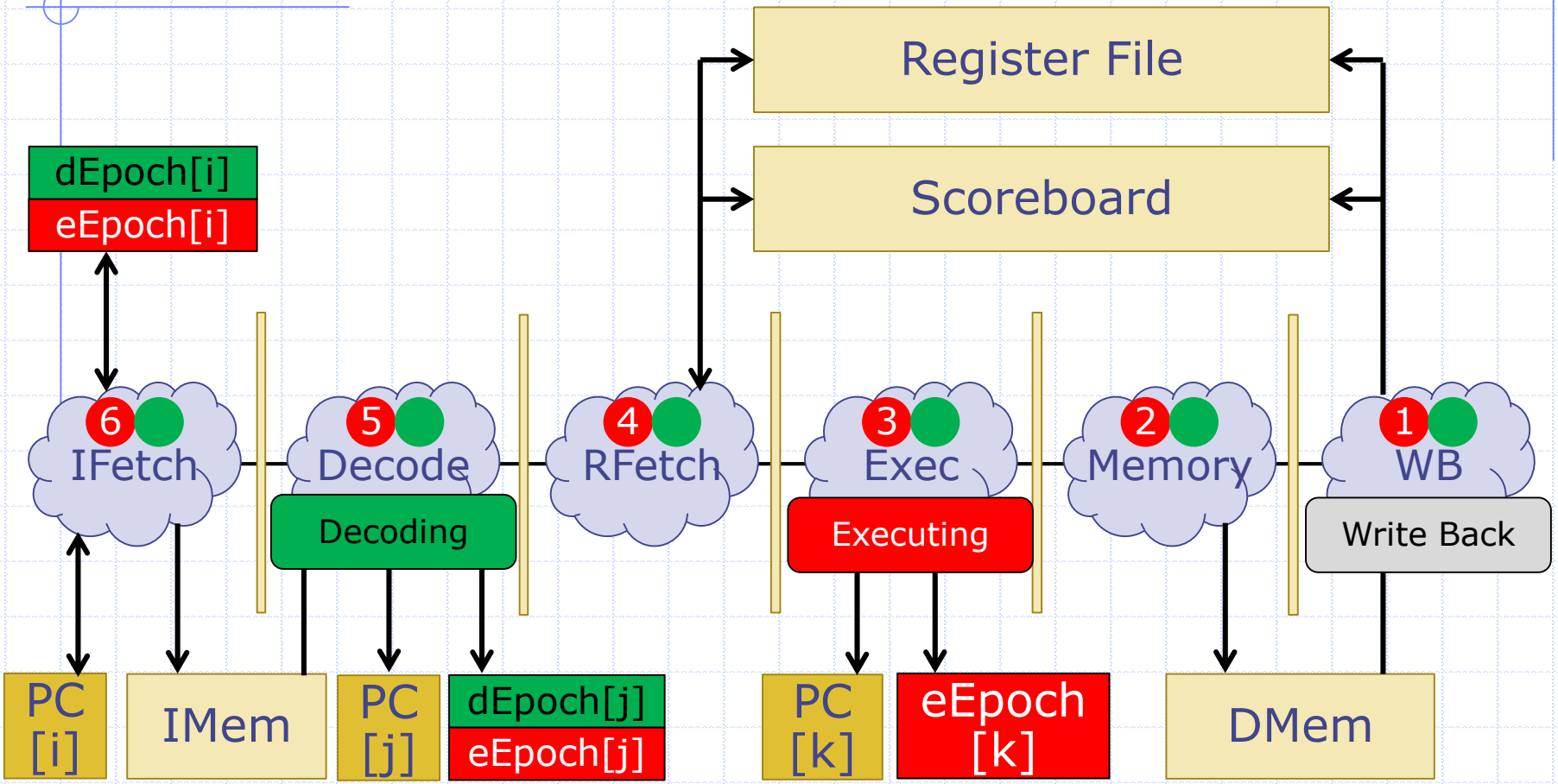
6 stage pipeline



ASAP Prediction Correction

- Different instructions that affect the program flow can be resolved at different times
  - Absolute Jumps – Decode
  - Register Jumps – RFetch
  - Branches – Exec
- You can save cycles on each misprediction by correcting the PC once you have computed what the next PC should have been.

Implementing Global Epoch States with EHRs



Make PC an EHR and have each pipeline stage redirect the PC directly

Important point of the cartoon picture

- No canonicalize rule.
- Only update direct to a pc ehr.
- Very few stage check the epoch:
  - Only stages that updates the epochs and do redirection.
  - Why is it not required for other stages?

Questions?