Royal Holloway University of London

# Lecture 9 – Pipelining Hazards

Karl R. Wilcox
Karl@cs.rhul.ac.uk

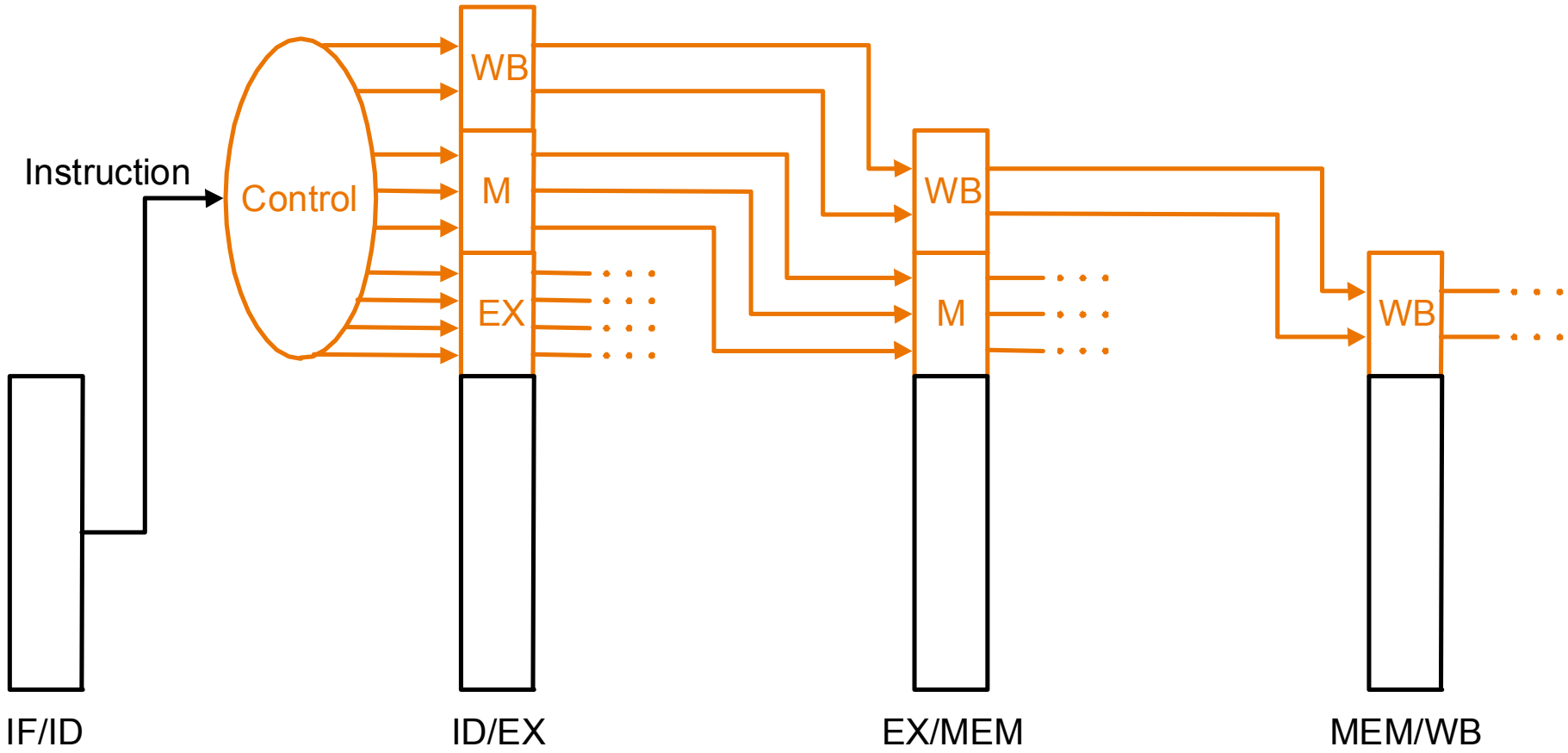
# Objectives

- **In this lecture we will cover**
  - **Pipeline control**
  - **Pipelines in action**
  - **Dealing with Data Hazards**
  - **Dealing with Control Hazards**
  - **(diagrams are from Patterson & Hennessy)**

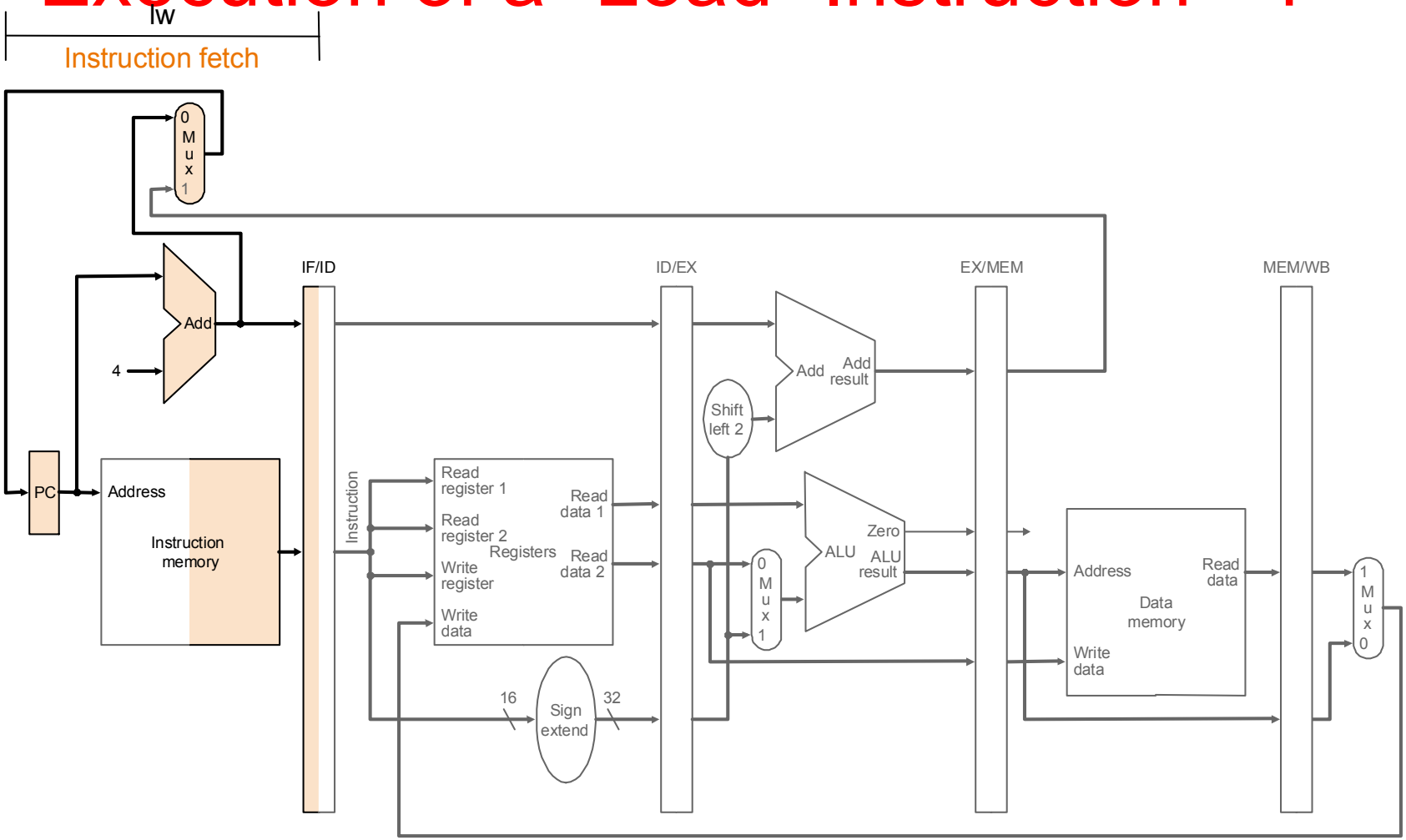
# Pipeline Control

- **In our single cycle datapath (lecture 7) the control signals (for registers, mulitplexors, ALU operation etc.) were generated by a "PLA" type state machine**
- **The same is true for a pipelined datapath but the control signals must be "buffered" so they apply to the correct pipeline step**
  - **Control signals needed for future steps are included in the buffer registers**

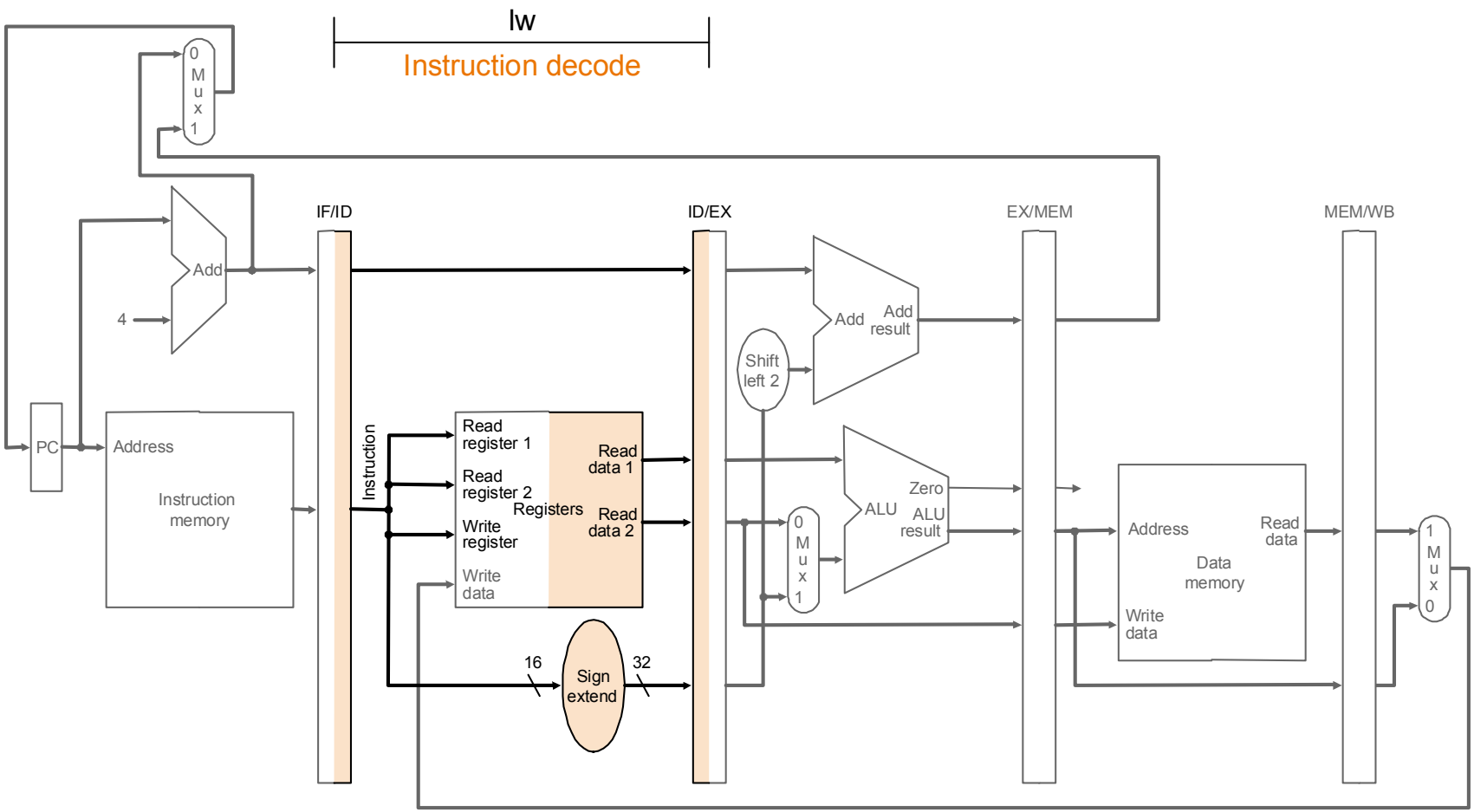
# Pipeline Control Signals



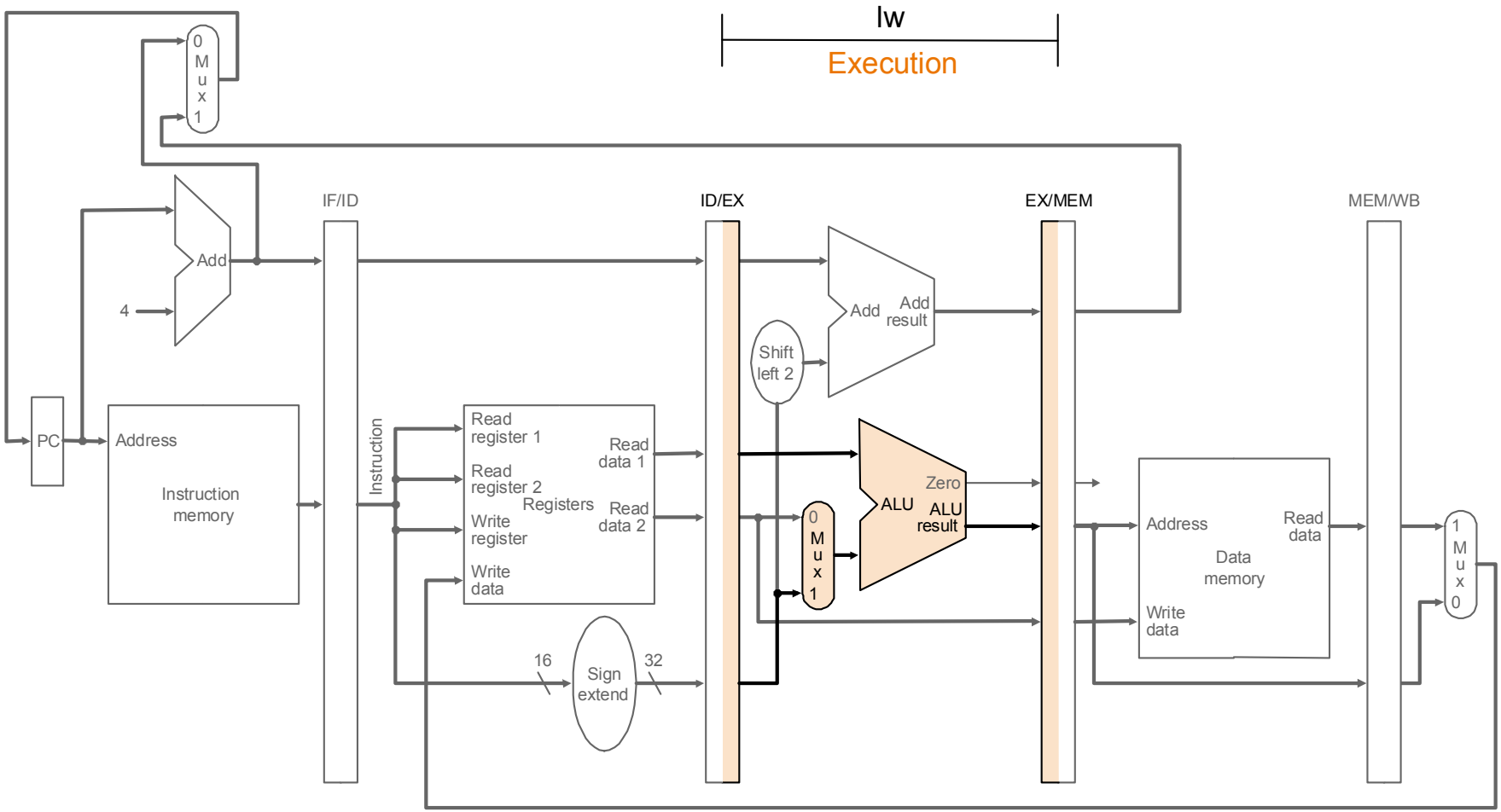
# Execution of a "Load" Instruction - 1



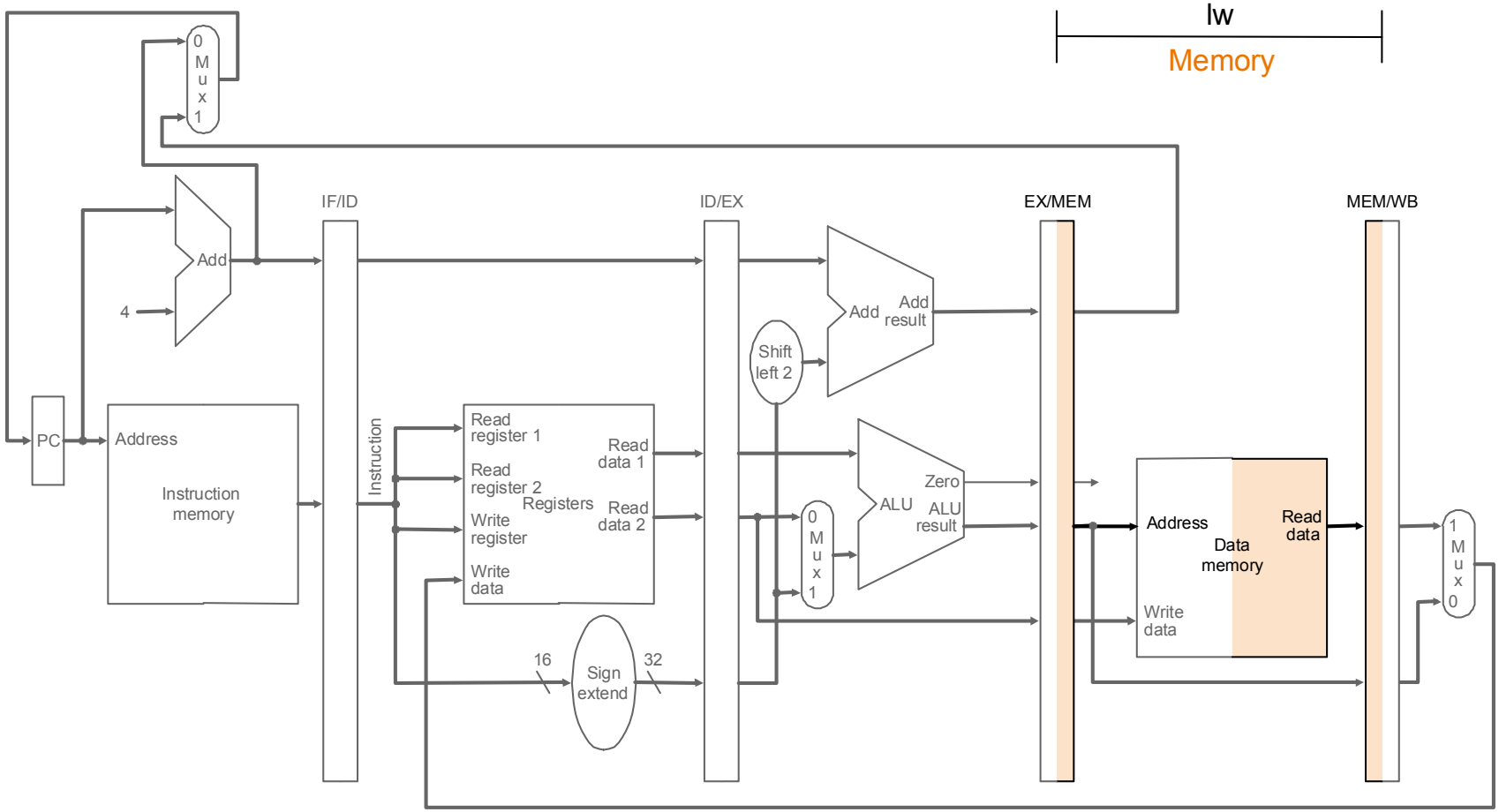
# Execution of a “Load” Instruction - 2



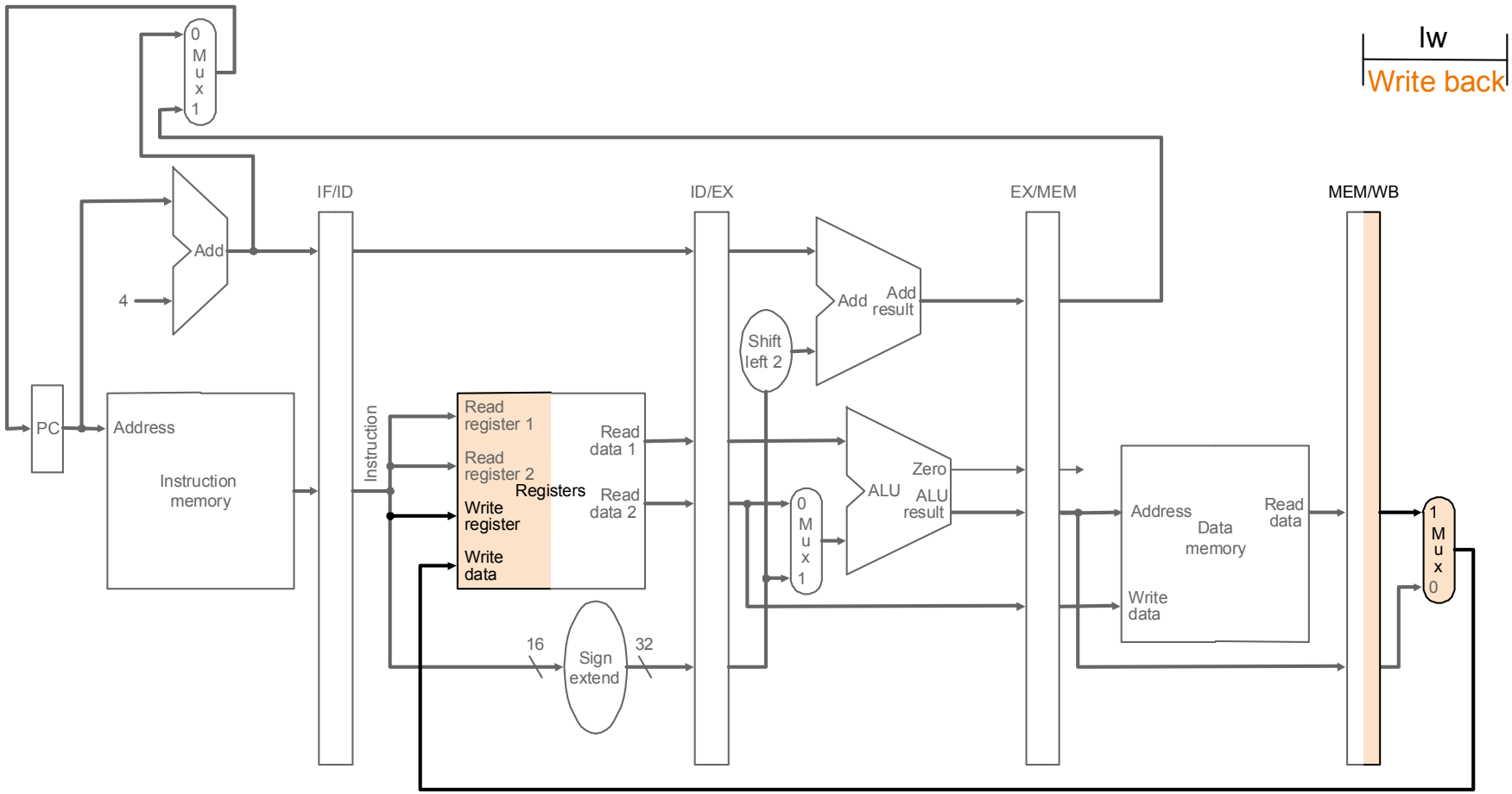
# Execution of a "Load" Instruction - 3



# Execution of a "Load" Instruction - 4



# Execution of a “Load” Instruction - 5



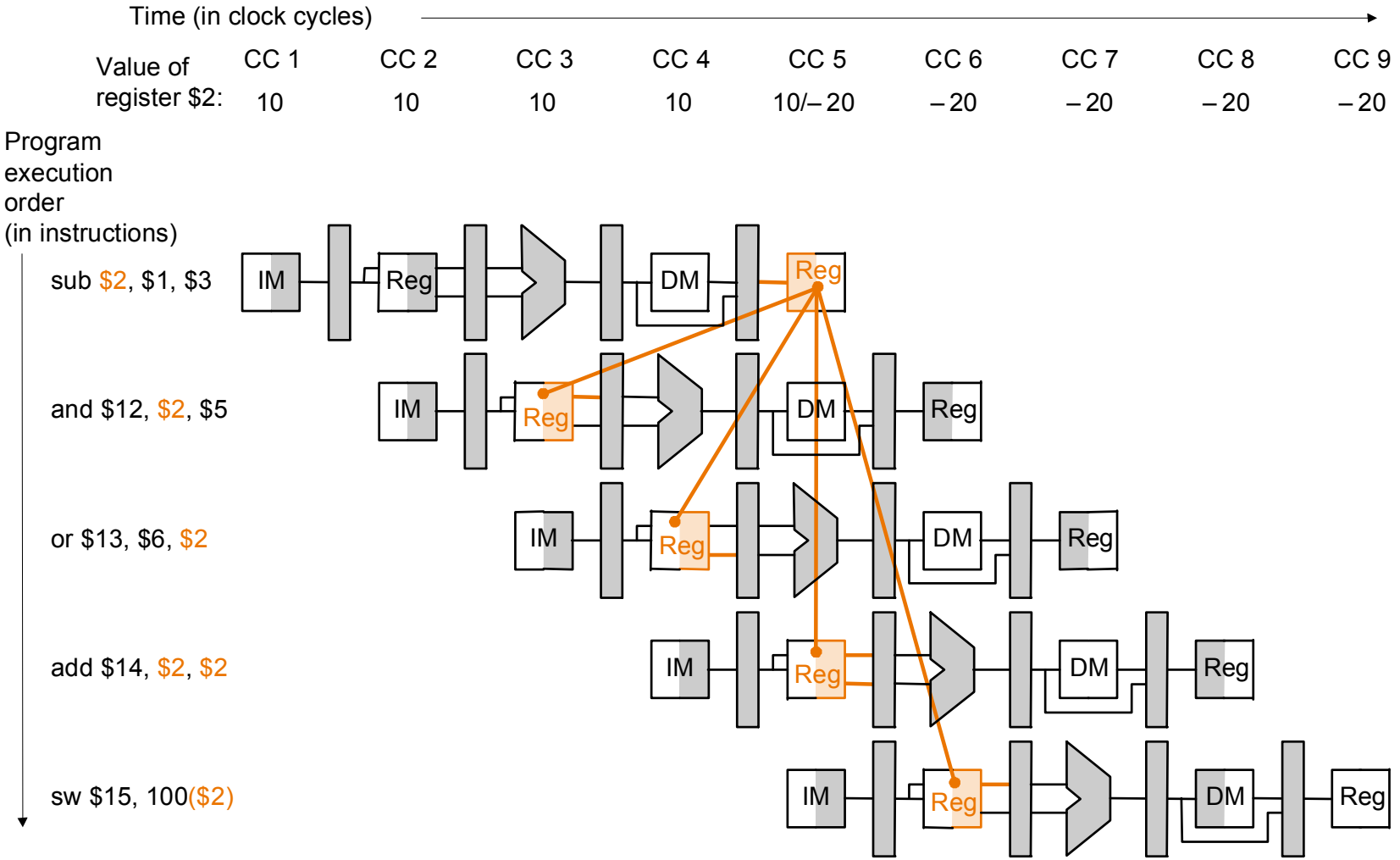
# Recall - Problems with pipelines

- **We can only execute the next instruction overlapped with the current one if:**
  - **That concurrent instructions do not need the same hardware at the same time (STRUCTURAL)**
  - **We know what the next instruction will be (CONTROL)**
  - **That all the operands we require are available (DATA)**
- **A Hazard exists if any of these conditions not met**

# Structural Hazards

- **Are avoided by careful instruction set design**
- **Or by adding extra (duplicate) logic**
  - **For example, if maintaining backwards compatibility with a previous (non-pipelined) instruction set**

# Data Hazards



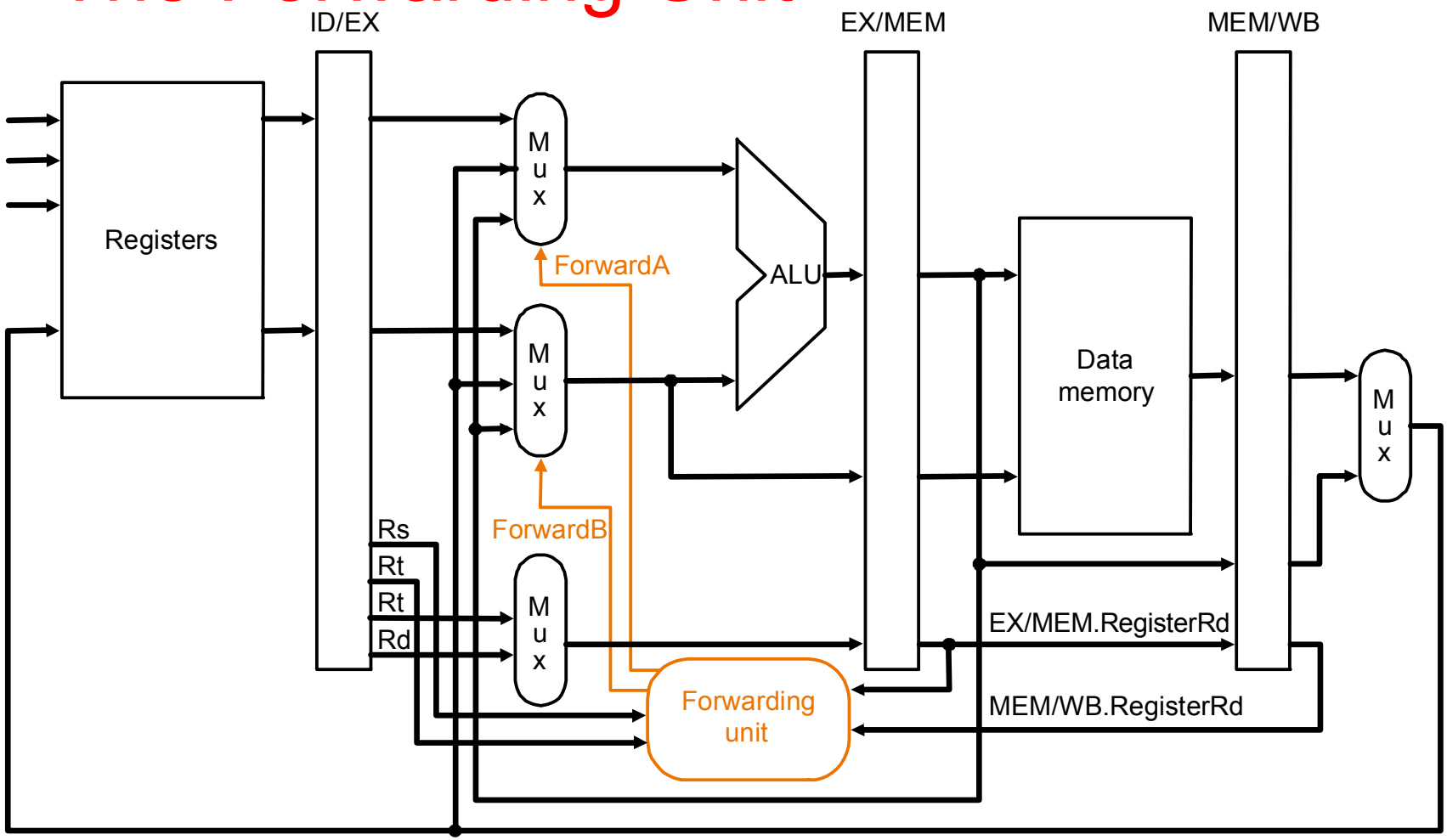
# Data Hazard Detection

- **Those dependencies that go "backwards" in time are data hazards**
- **We can detect them by looking at control signals from each step of the pipeline**
  - **These tell us which registers are being read / written**
  - **We can detect where a register is being read before it has been written**

# Data Hazard Resolution

- **Two approaches:**
  - **We could stall the pipeline until the register has been written**
    - **We have to do this if the hazard involves a "load" instruction since the data is only available after step 5 anyway**
    - **(See lecture 7)**
  - **We could take the required data directly from the buffer register**
    - **We would not need to wait until the writeback stage has completed**
    - **We would need additional inputs on multiplexors and control signals to select the appropriate source**
    - **This is done by the Forwarding Unit**

# The Forwarding Unit



Royal Holloway University of London

# Control Hazard Detection/Resolution

- **Control hazards are caused by conditional jumps**
  - **We do not know what the next instruction will be**
- **We could make an assumption about branch**
  - **Then stall & flush pipeline if we were wrong**
- **Additionally, we could evaluate the branch earlier**
  - **We do not need to wait until the writeback step to stall the pipeline**
  - **We know the answer at the end of the execution step**
  - **We could add hardware to evaluate loops at the instruction identification stage**
  - **This means we have to discard fewer steps if predict wrongly**

# Dynamic Branch Prediction

- We could maintain a history of how branches were taken
- Use a branch buffer (1 = taken, 0 = not taken), indexed by low order bits of address
  - Not perfect, may re-use addresses
- This could mis-predict twice on each loop
  - Once as a "left over" from the last time the loop was executed
  - Once when the loop is exited
- To avoid this use a 2 bit prediction scheme
  - Prediction must be wrong twice before changing

# Summary

- Pipeline control signals need to be buffered along with the data between each pipeline step
- When we watch a pipeline in operation we can see the hardware requirements for each step
- Data hazards are detected by monitoring the control signals, resolved by stalling or forwarding
- Control hazards are caused by conditional jumps, resolved by stalling & flushing, or prediction

Royal Holloway University of London

# Next Lecture

- The MIPS branch delay slot
- Exception handling
- Applicability of pipelining techniques
- Next Laboratory session, To be arranged