University of Nottingham
UK | CHINA | MALAYSIA

Instruction Pipelining

Dr. Heng Yu

AY2022-23, Spring Semester
COMP1047: Systems and Architecture
Week 7

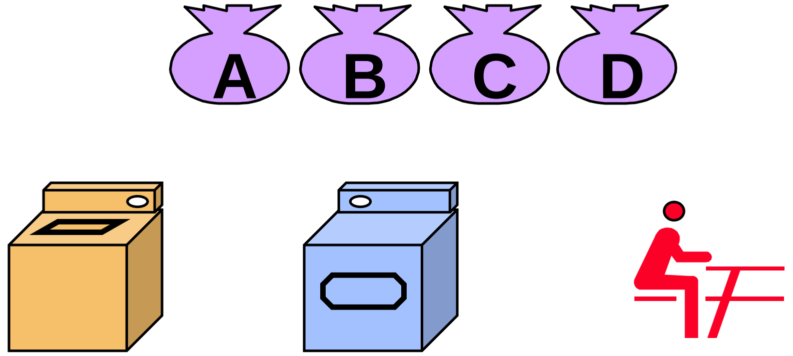
Today’s outline

- An overview of pipelining
- A pipelined datapath
- Pipelined control
- Hazards: types of hazard
  - Handling data hazards
  - Handling branch hazards

Introduction

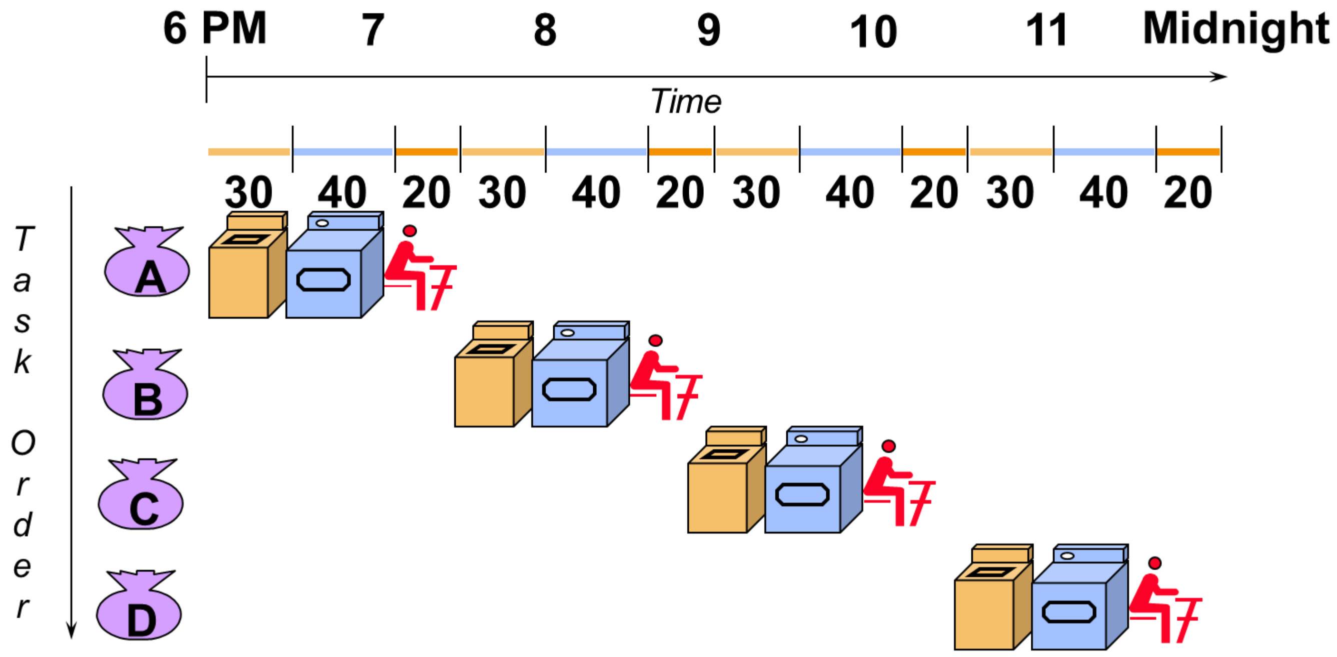
- Pipelining is Natural.

- Laundry example:



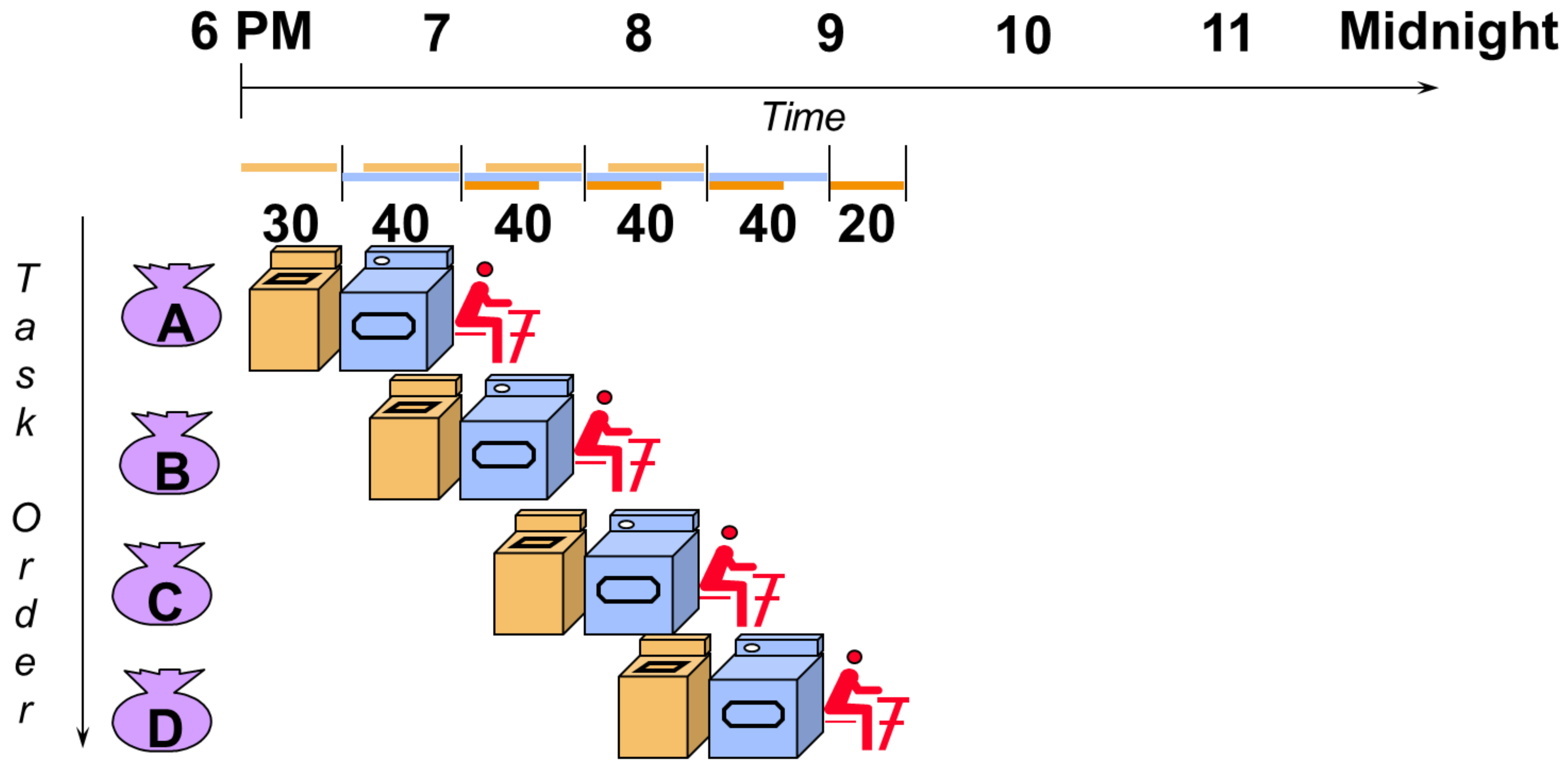
- Anoud, Banan, Chanda, Daania each have one load of clothes to wash, dry, and fold.
  - Washer takes 30 minutes
  - Dryer takes 40 minutes
  - "Folder" takes 20 minutes

- If they do it in sequence, total time is ____ for the four of them to finish.



Introduction

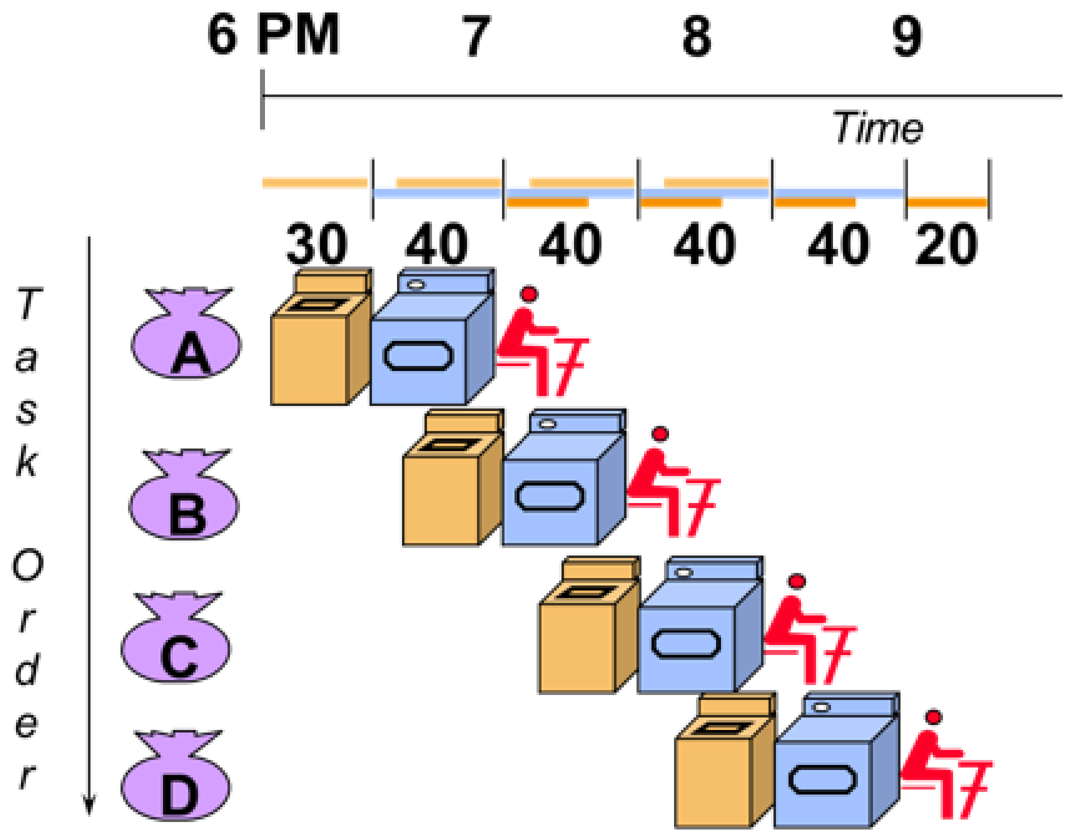
- If they learned pipelining, how long would it take?



Introduction

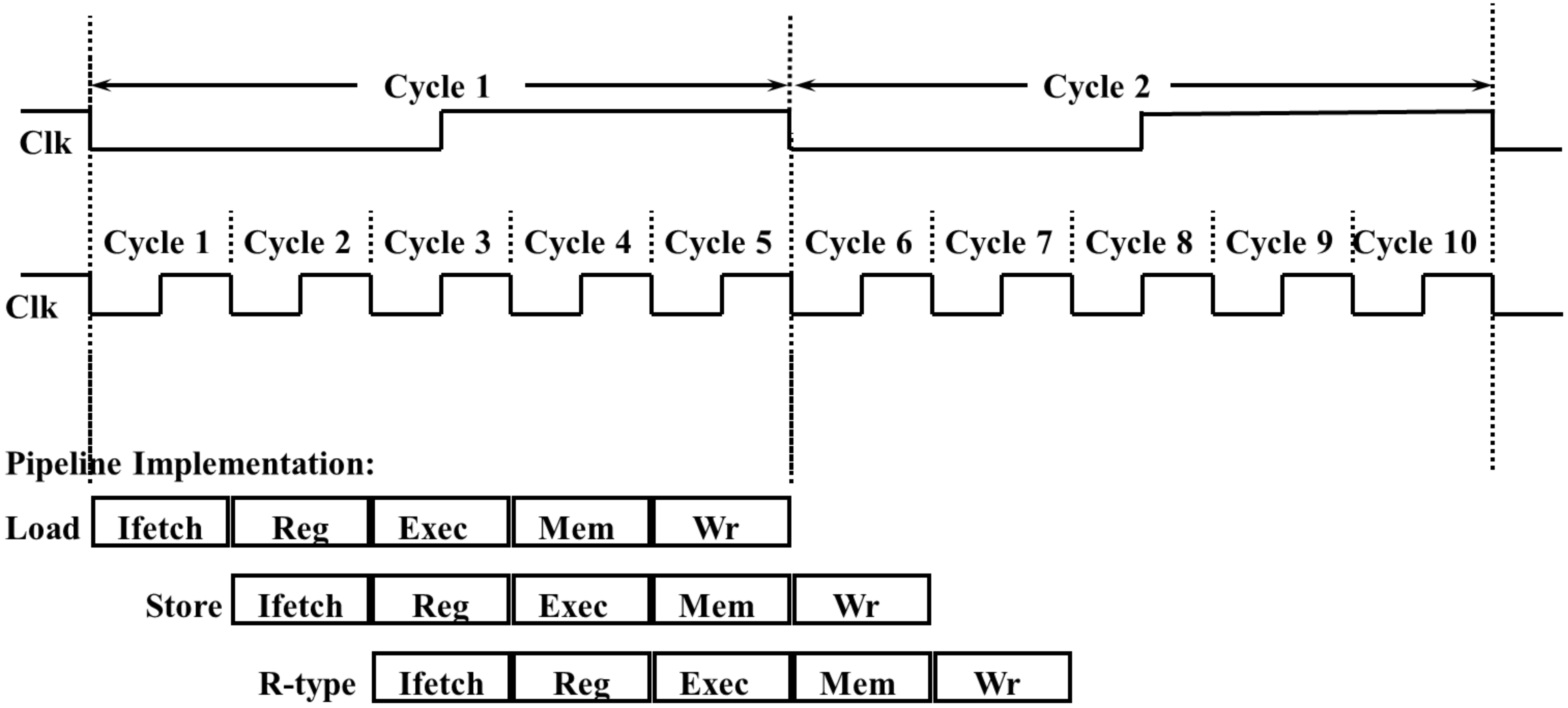
Pipelining lessens:

- Doesn’t help latency of a single task, but the overall throughput is improved.
- Pipeline rate limited by slowest stage
- Multiple tasks working at same time using different resources
- Potential speedup = Number pipe stages
- Both (1) unbalanced stage length, and (2) the process of “filling” & “draining” the pipeline reduce speedup



Introduction

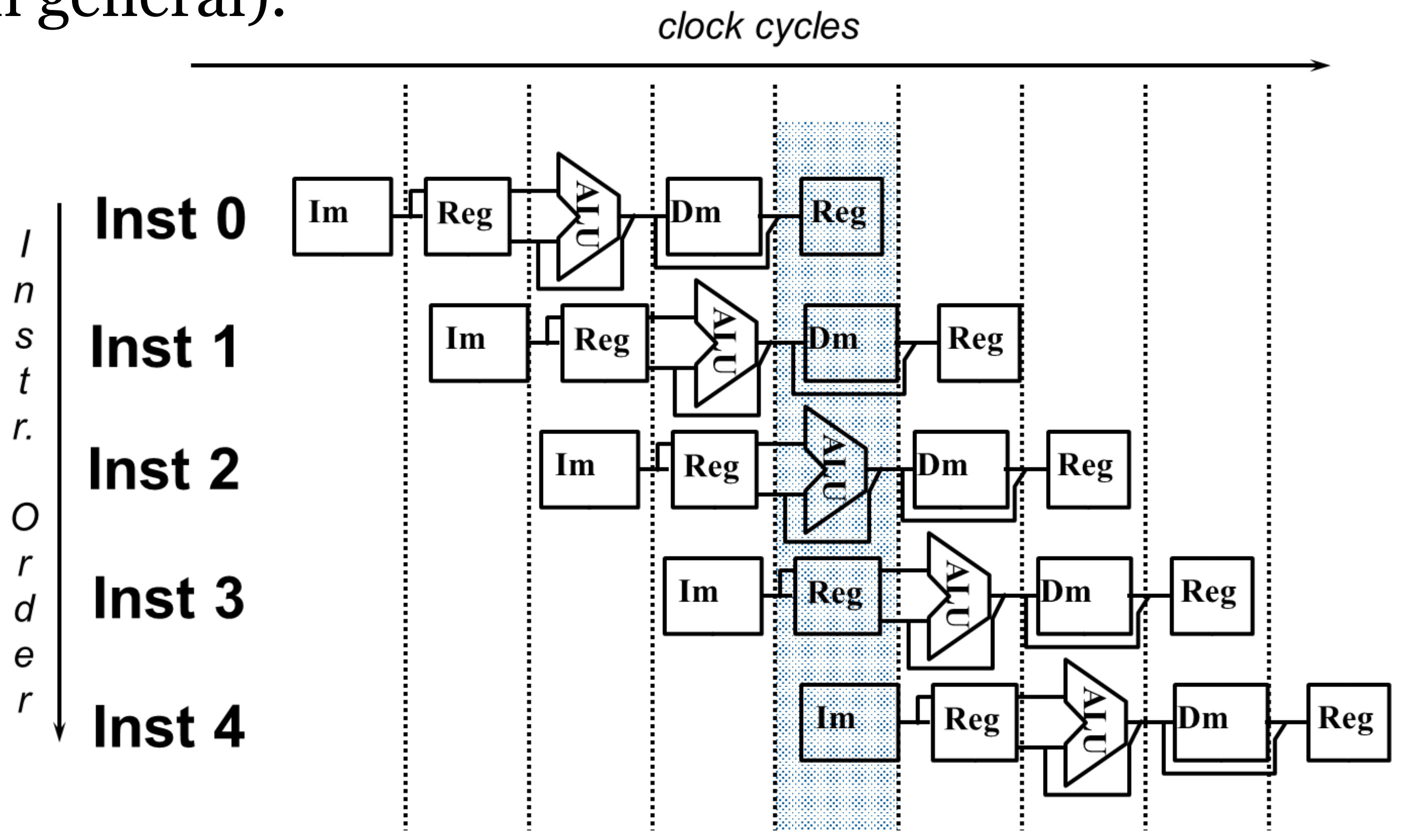
- Similar to the laundry example, instruction executions can also be pipelined.



Introduction

Why pipelining the instructions?

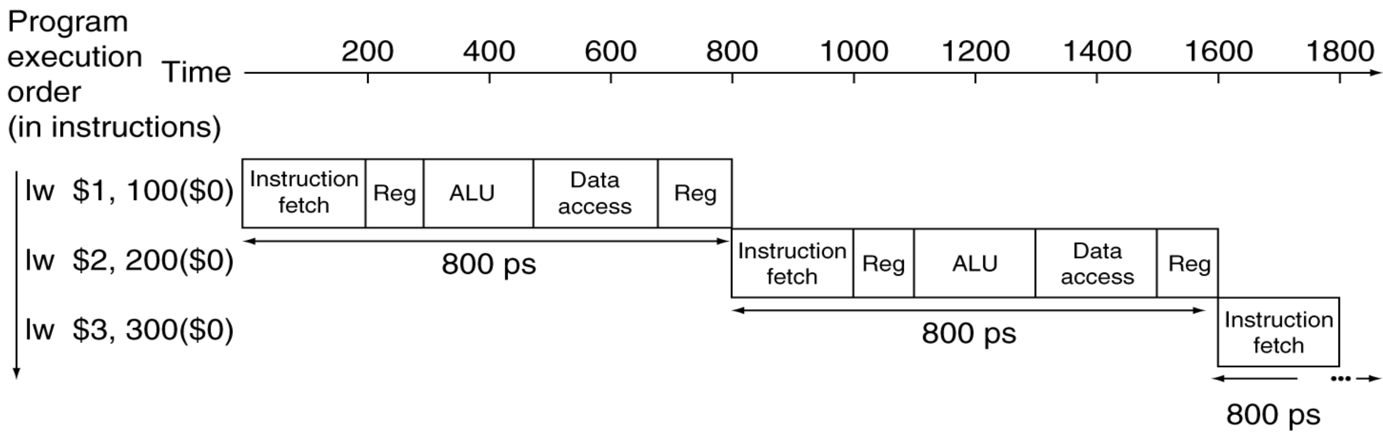
- Different stages of the instructions utilize different hardware blocks (in general).

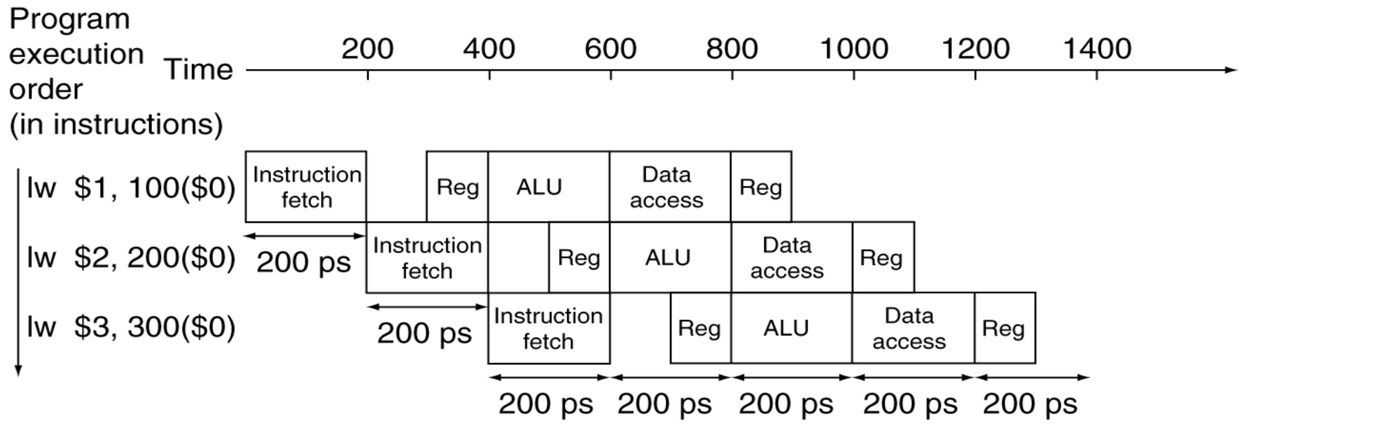


Introduction

- Major advantage of instruction pipelining
  - Saves time, allows higher clock frequency.
  - Note how the ALU dominates Reg in each pipeline stage.
  - Price to pay: extra design effort.

Single-cycle (T_c= 800ps)



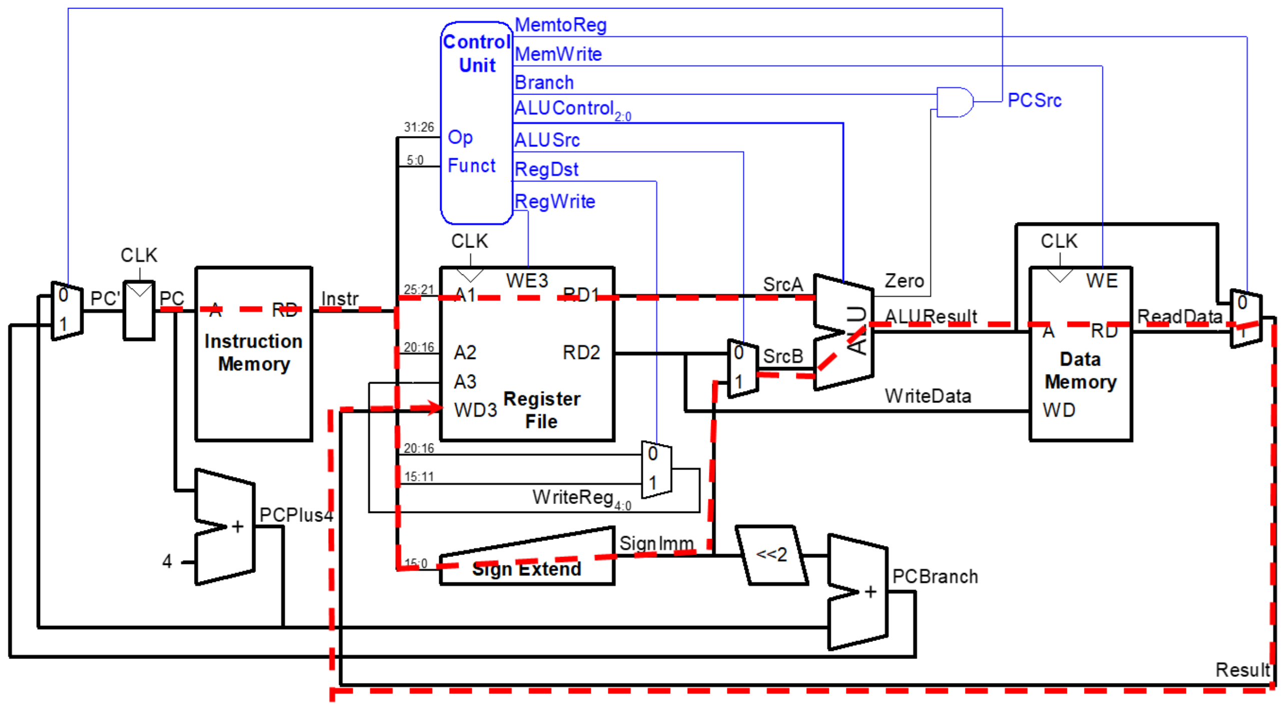


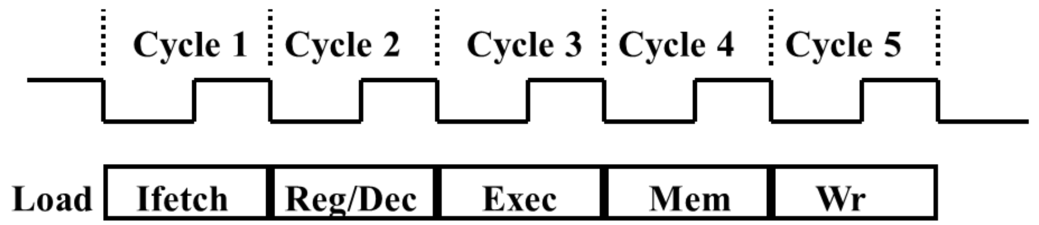
Pipelined (T_c= 200ps)

Pipelined Datapath

Consider `load` instruction

- IF: Instruction Fetch
  - Fetch the instruction from the Instruction Memory
- ID: Instruction Decode
  - Registers fetch and instruction decode
- EX: Calculate the memory address
- MEM: Read the data from the Data Memory
- WB: Write the data back to the register file

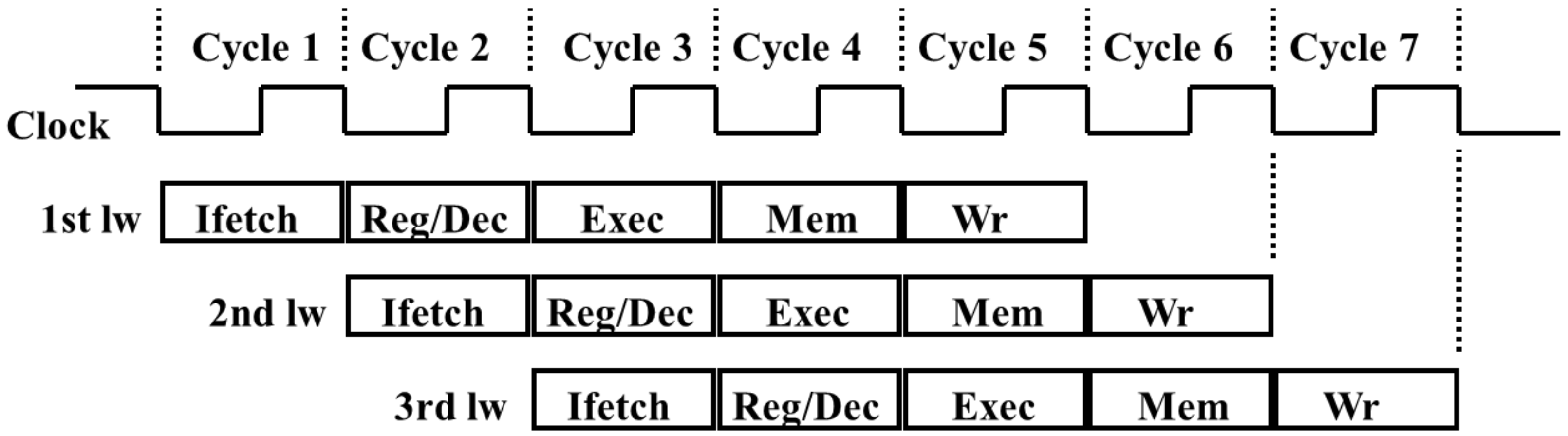




Pipelined Datapath

Pipelining `load`

- 5 functional units in the pipeline datapath:
  - Instruction Memory for the Ifetch stage
  - Register File's Read ports (busA and busB) for the Reg/Dec stage
  - ALU for the Exec stage
  - Data Memory for the MEM stage
  - Register File's Write port (busW) for the WB stage



Today's outline

- An overview of pipelining
- A pipelined datapath
- Pipelined control
- Hazards: types of hazard
    - Handling data hazards
    - Handling branch hazards

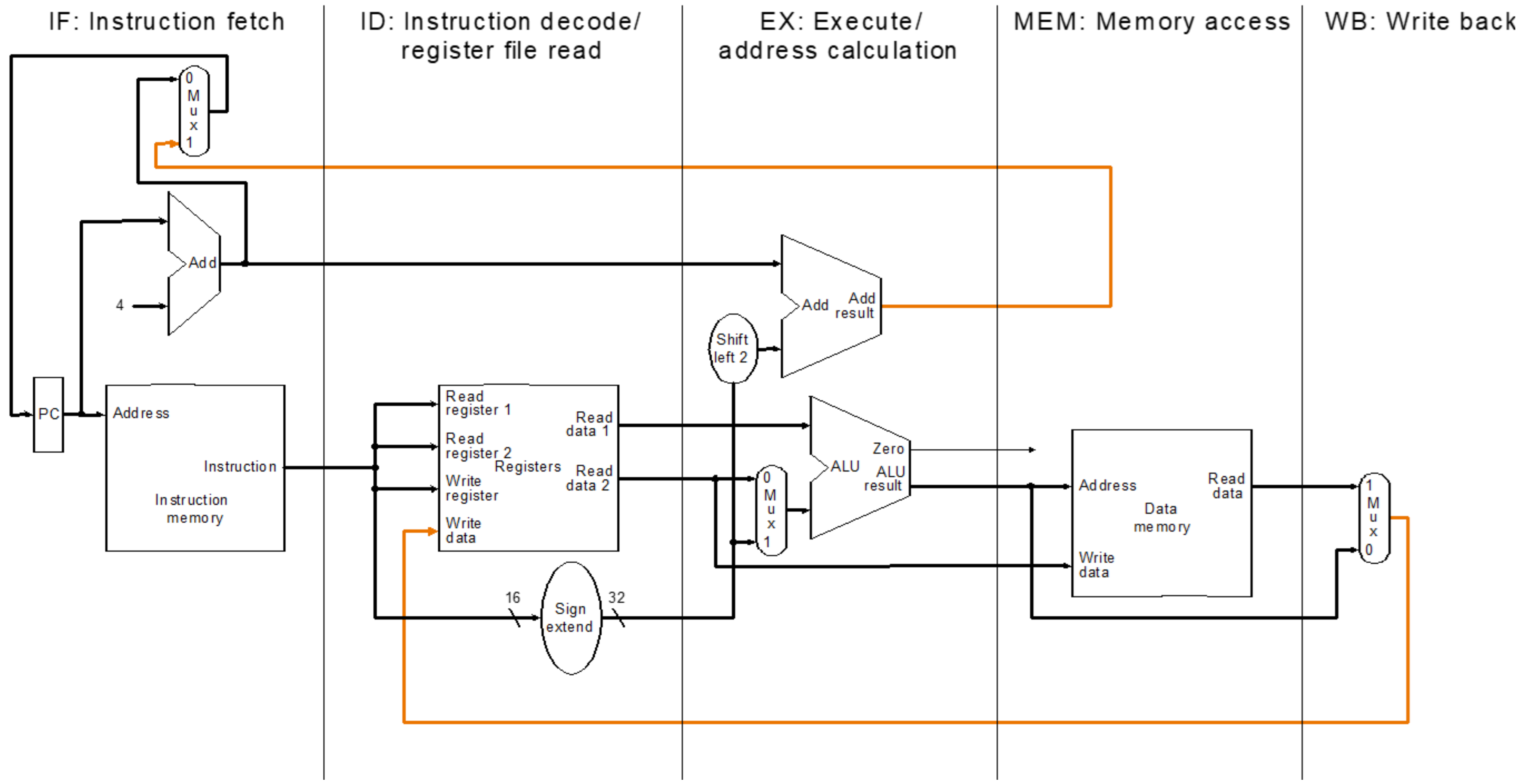
Pipelined Datapath

Designing a Pipelined Processor

- Starting with single cycle datapath
- Partition datapath into stages:
  - IF (instruction fetch), ID (instruction decode and register file read), EX (execution or address calculation), MEM (data memory access), WB (write back)
- Associate resources with stages
- Ensure that flows do not conflict, or figure out how to resolve
- Assert control in appropriate stage

Pipelined Datapath

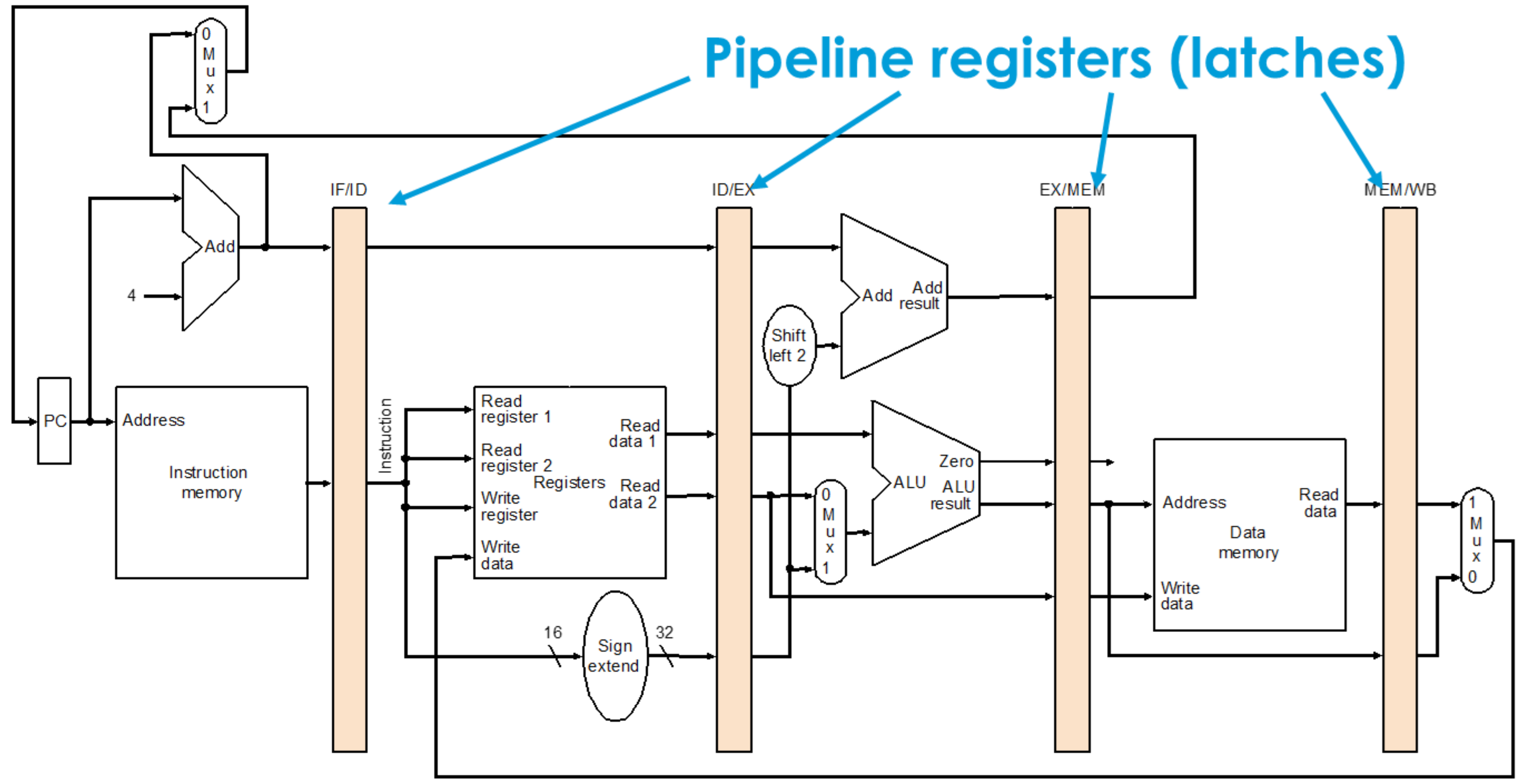
Partitioning datapath into stages



What hardware to add to split the datapath into stages?

Pipelined Datapath

Partitioning datapath into stages

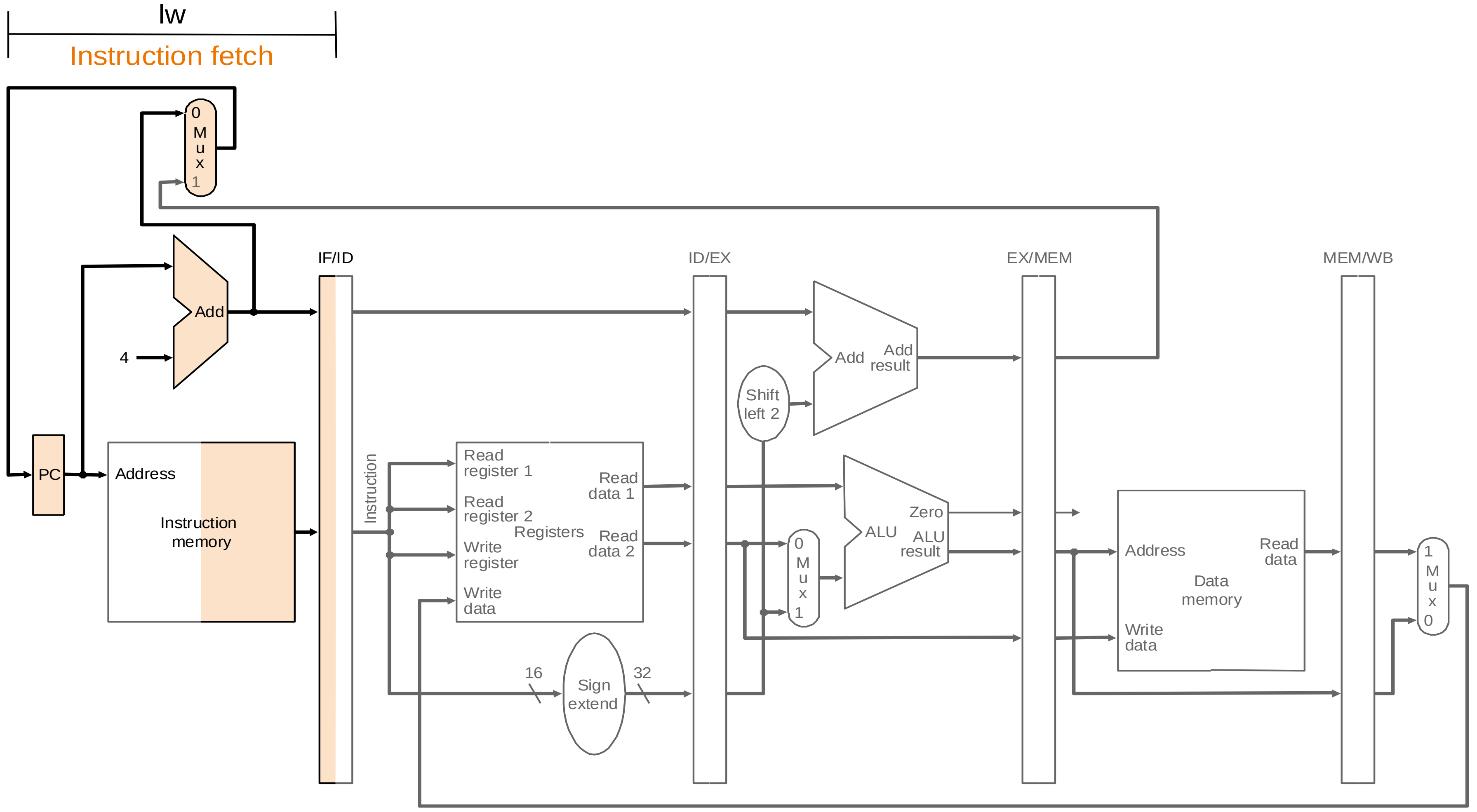


Use registers between stages to carry data and control!

Pipelined Datapath

IF stage of `load`

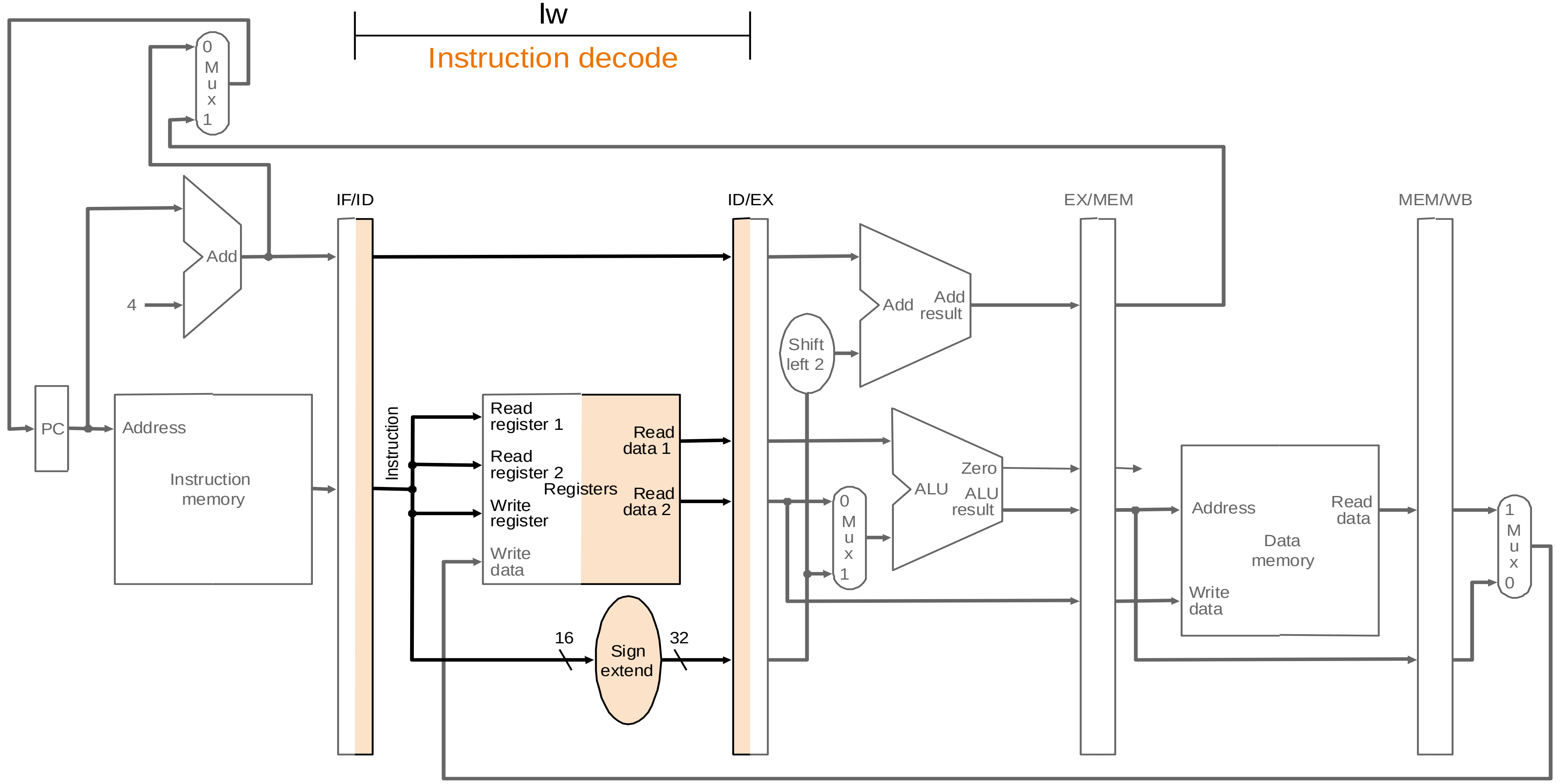
- IR = mem[PC]; PC = PC + 4



Pipelined Datapath

ID stage of `load`

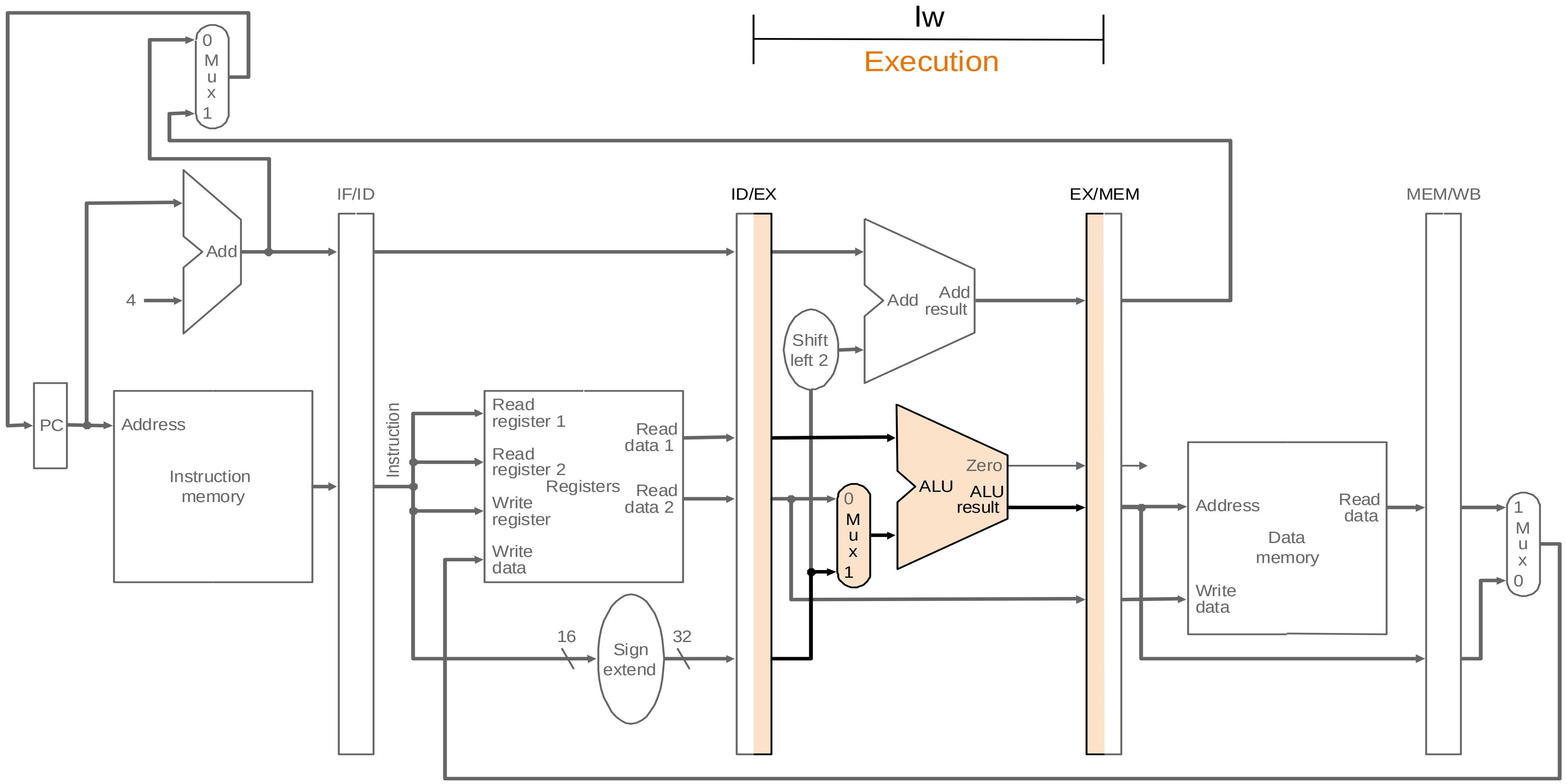
- A = Reg[IR[25-21]]; B = Reg[IR[20-16]];



Pipelined Datapath

EX stage of `load`

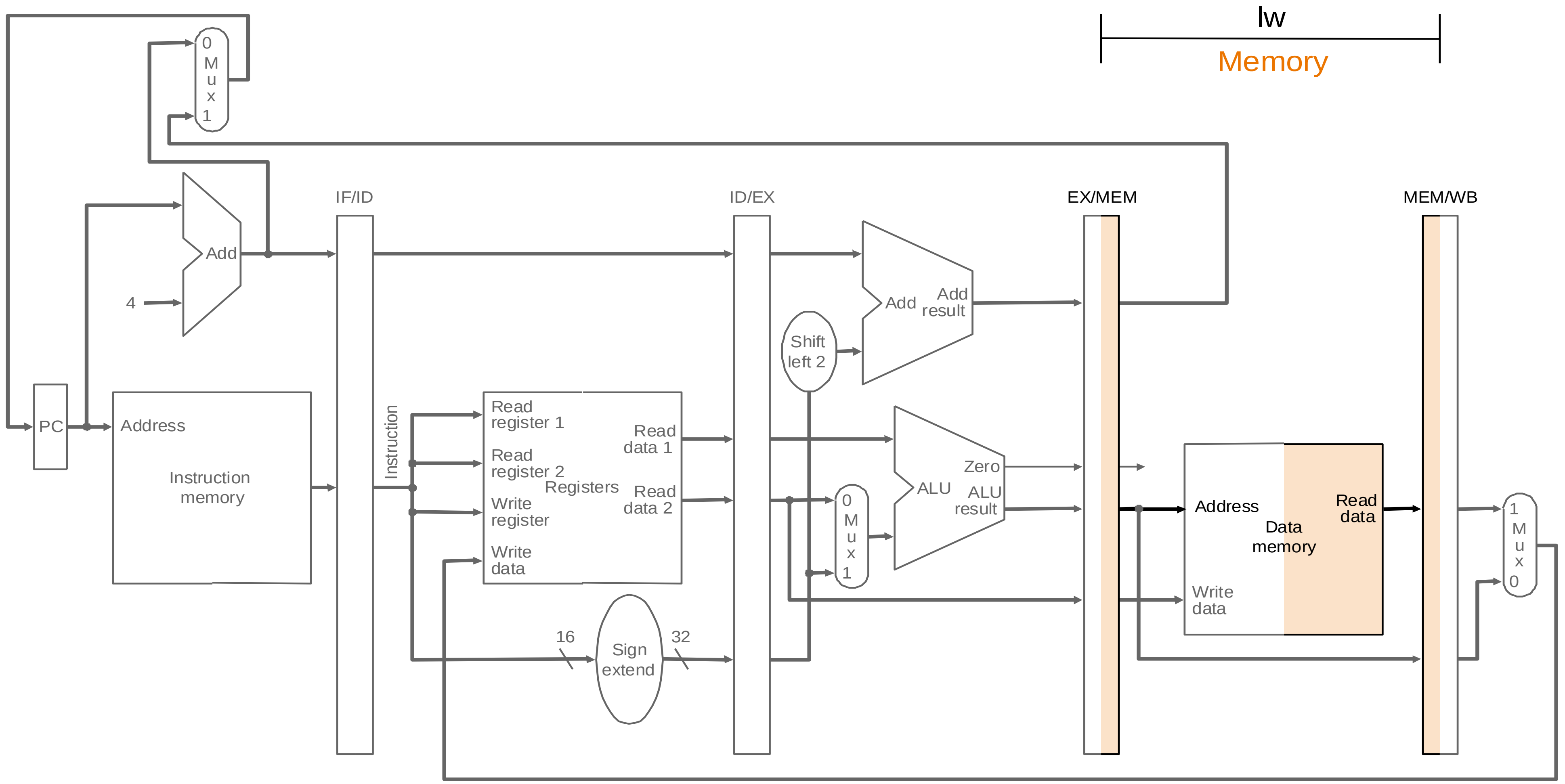
- ALUout = A + sign-ext(IR[15-0])



Pipelined Datapath

MEM stage of `load`

- MDR = mem[ALUout]

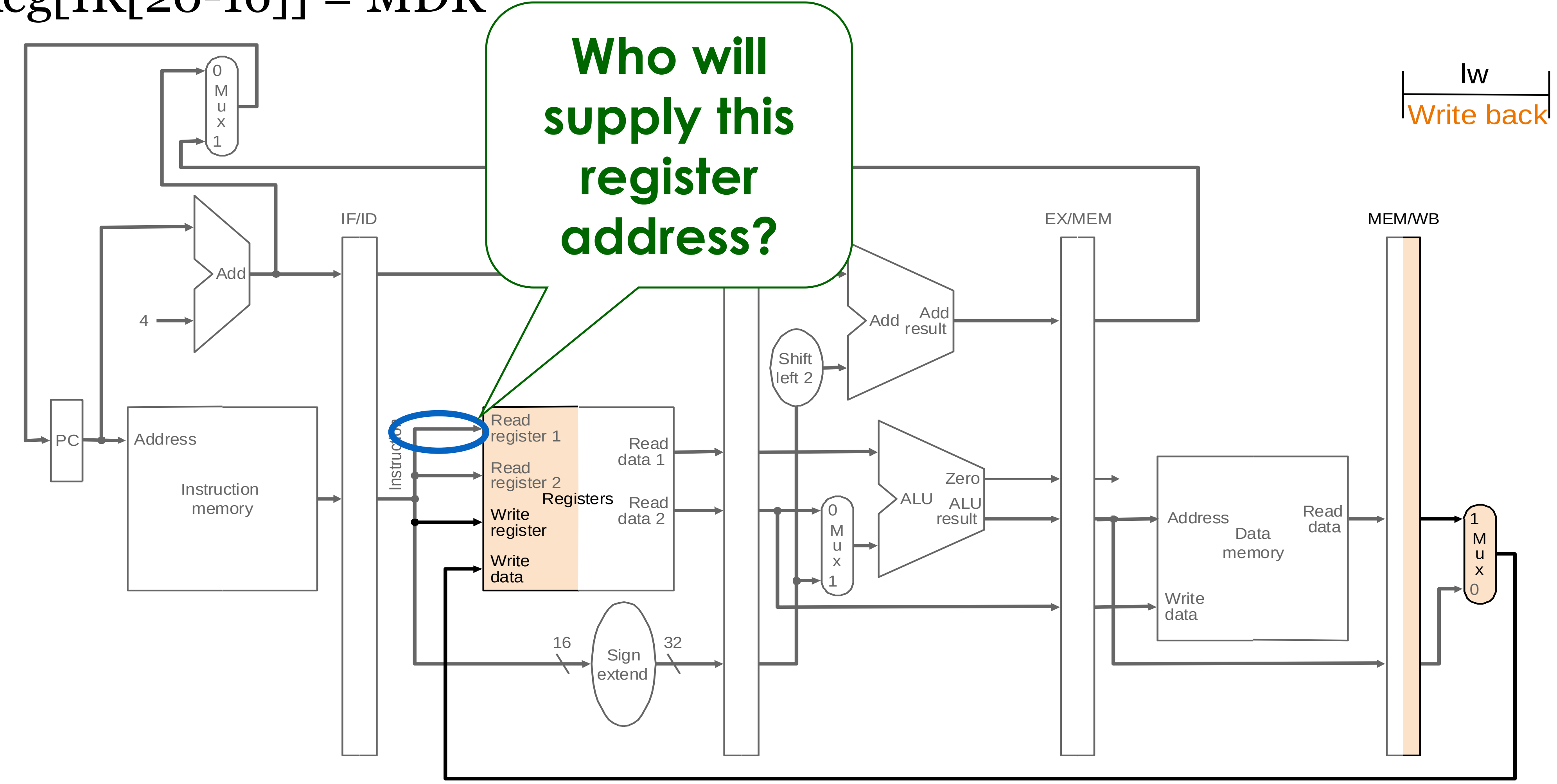


University of Nottingham UK | CHINA | MALAYSIA

Pipelined Datapath

WB stage of `load`

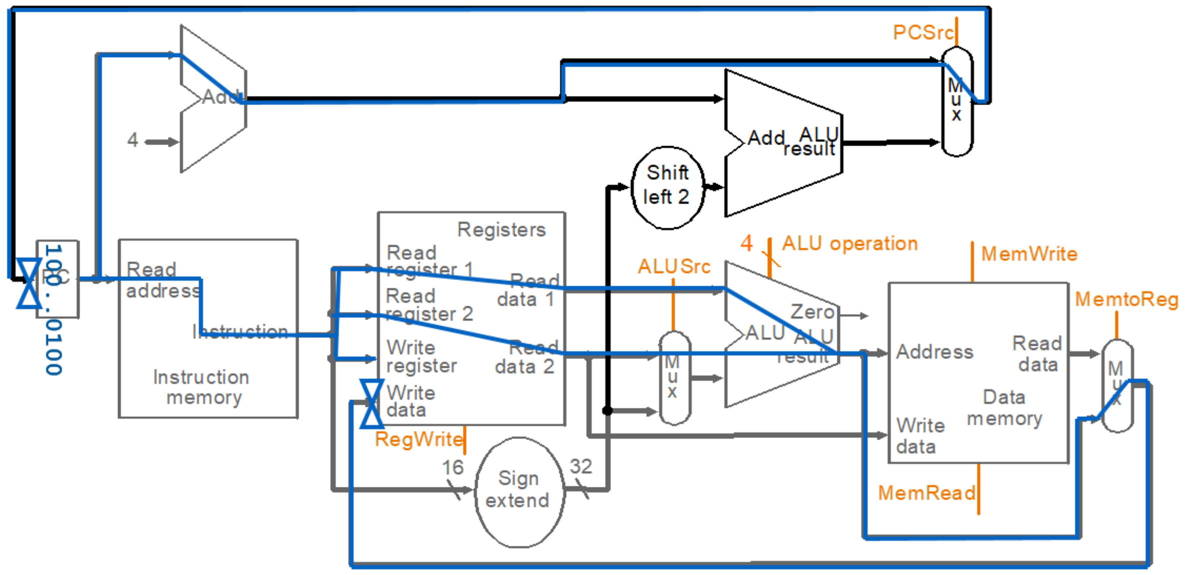
- Reg[IR[20-16]] = MDR

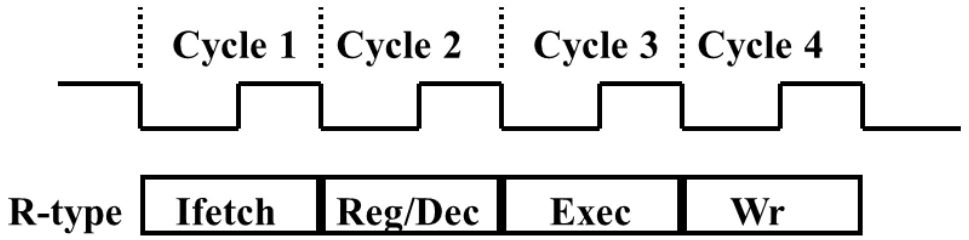


Pipelined Datapath

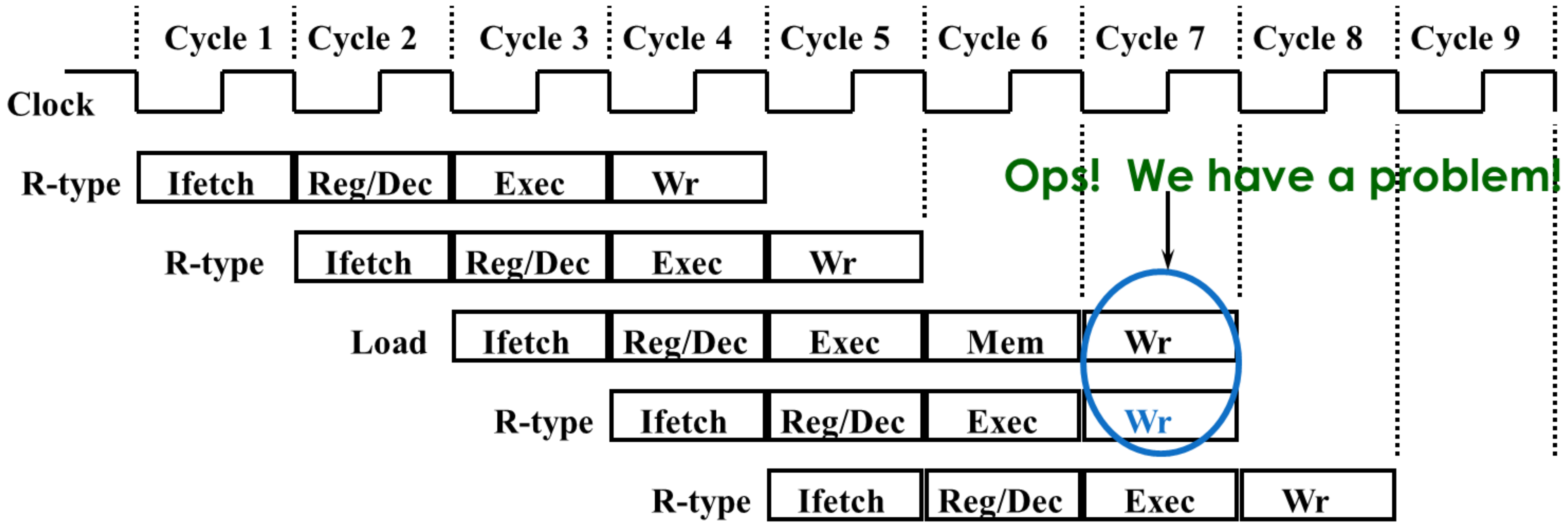
The four stages of R-type instructions

- IF: fetch the instruction from the Instruction Memory
- ID: registers fetch and instruction decode
- EX: ALU operates on the two register operands
- WB: write ALU output back to the register file





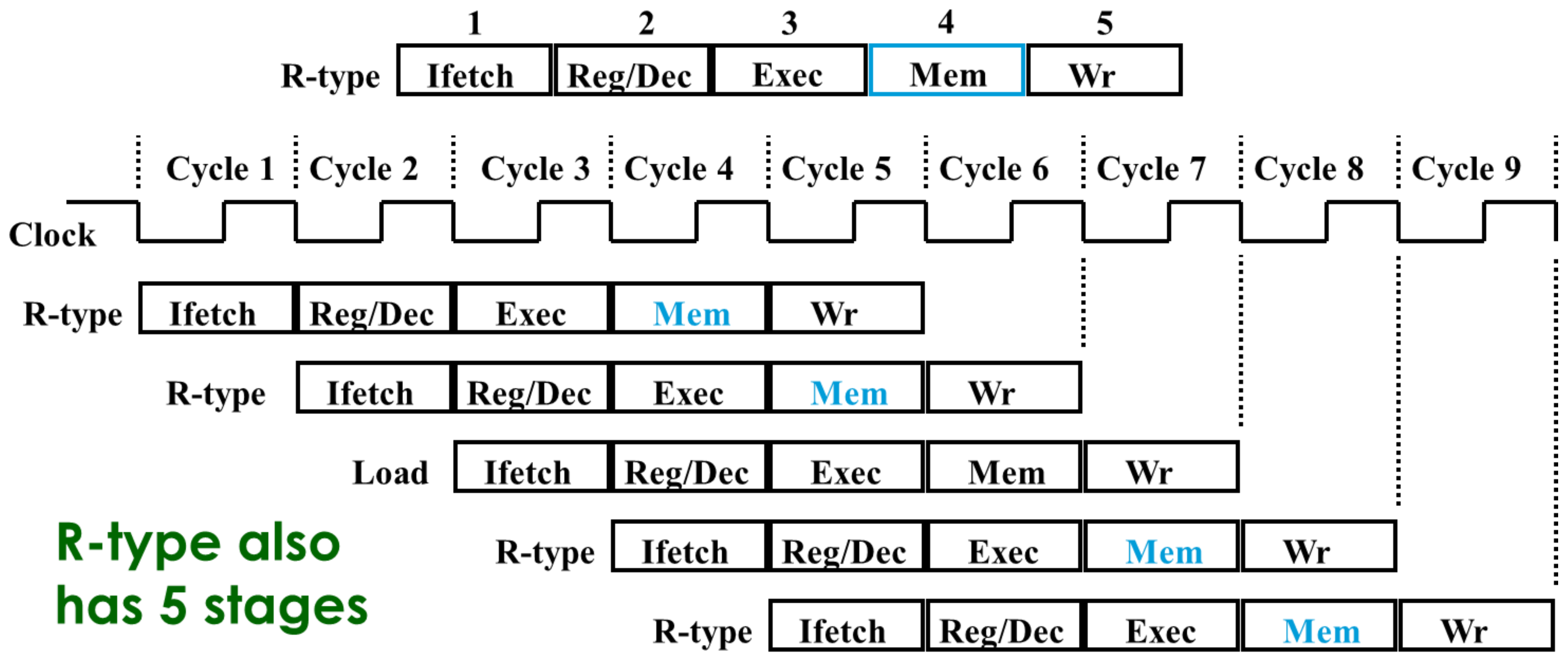
Pipelining R-type instructions and `load`


Cycle 1
Cycle 2
Cycle 3
Cycle 4
Cycle 5
Cycle 6
Cycle 7
Cycle 8
Cycle 9
Clock
R-type Ifetch Reg/Dec Exec Wr
R-type Ifetch Reg/Dec Exec Wr
Load Ifetch Reg/Dec Exec Mem Wr
R-type Ifetch Reg/Dec Exec Wr
R-type Ifetch Reg/Dec Exec Wr
Ops! We have a problem!

- We have a *structural hazard*:
- Two instructions try to write to the register file at the same time!
  - ❖ But only one write port

Solution: Delay R-type's write

- Delay R-type's register write by one cycle:
  - ❖ R-type also use Reg File's write port at Stage 5
  - ❖ MEM is a NOP stage: nothing is executed in this stage.



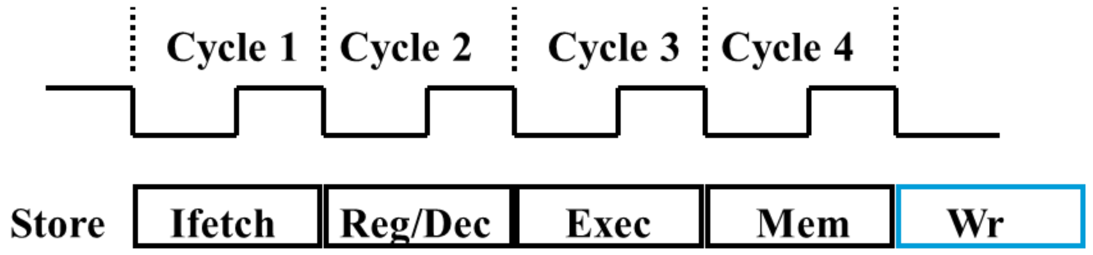
R-type also has 5 stages

Image Link

Pipelined Datapath

The four stages of `store`

- IF: fetch the instruction from the Instruction Memory
- ID: registers fetch and instruction decode
- EX: calculate the memory address
- MEM: write the data into the Data Memory
- Add an extra stage – WB: NOP

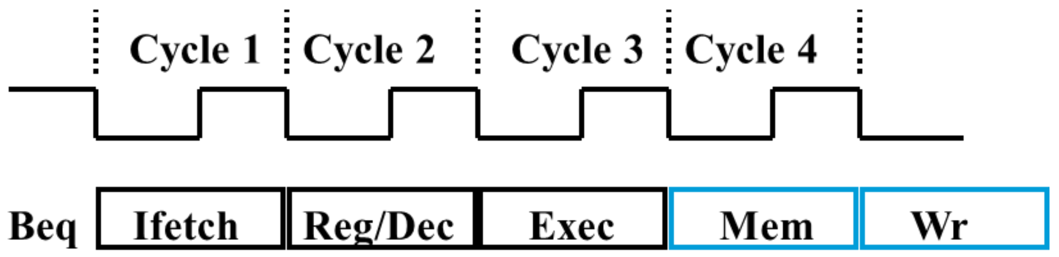


University of Nottingham UK | CHINA | MALAYSIA

Pipelined Datapath

The three stages of beq

- IF: fetch the instruction from the Instruction Memory
- ID: registers fetch and instruction decode
- EX:
  - Compares the two register operands
  - Issue the control signals to
    - Select correct branch target address
    - Latch into PC
- Add two extra stages –
  - MEM: NOP
  - WB: NOP



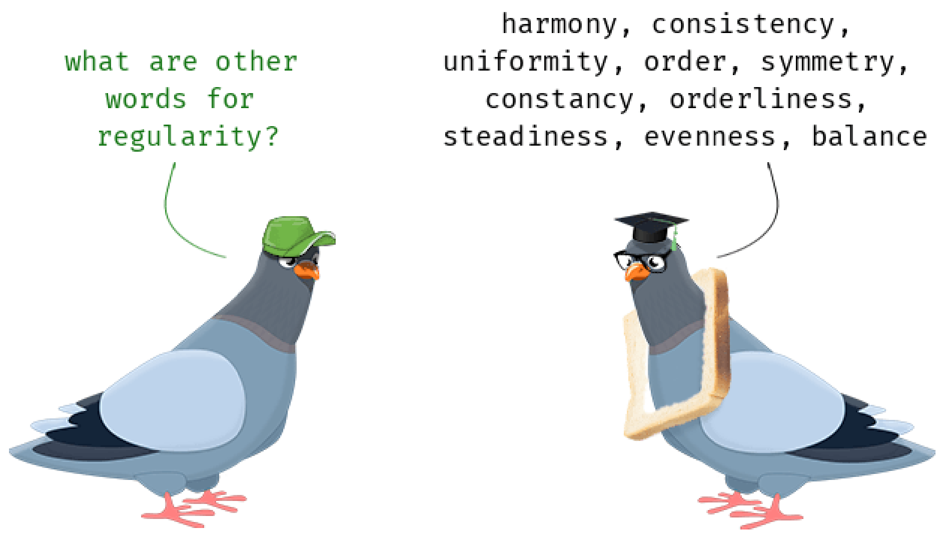

what are other words for regularity?
harmony, consistency, uniformity, order, symmetry, constancy, orderliness, steadiness, evenness, balance

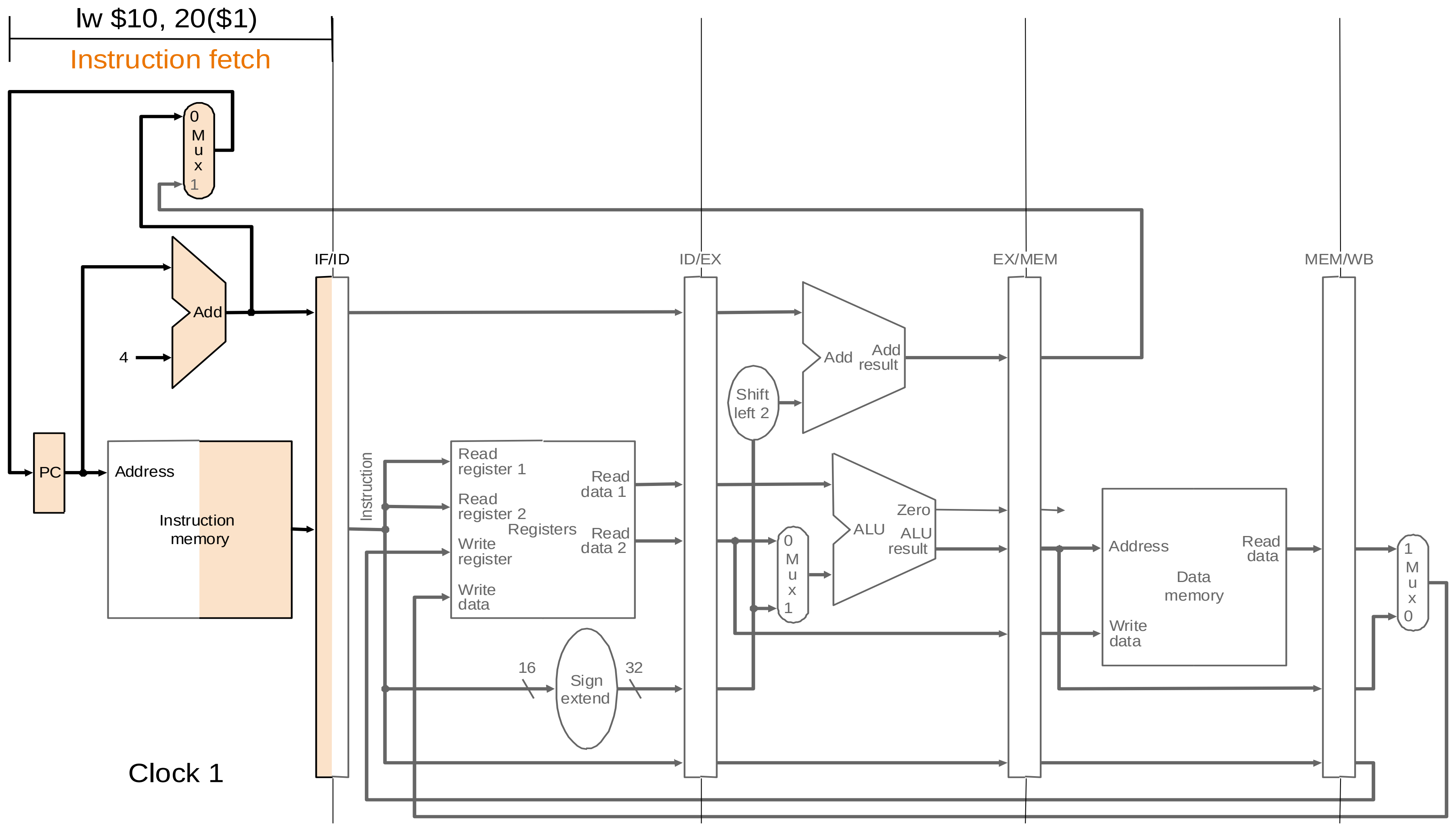
Image Link

University of Nottingham
UK | CHINA | MALAYSIA

Pipelined Datapath

Example: Pipelining `lw $10, 20($1)` and `sub $11, $2, $3`

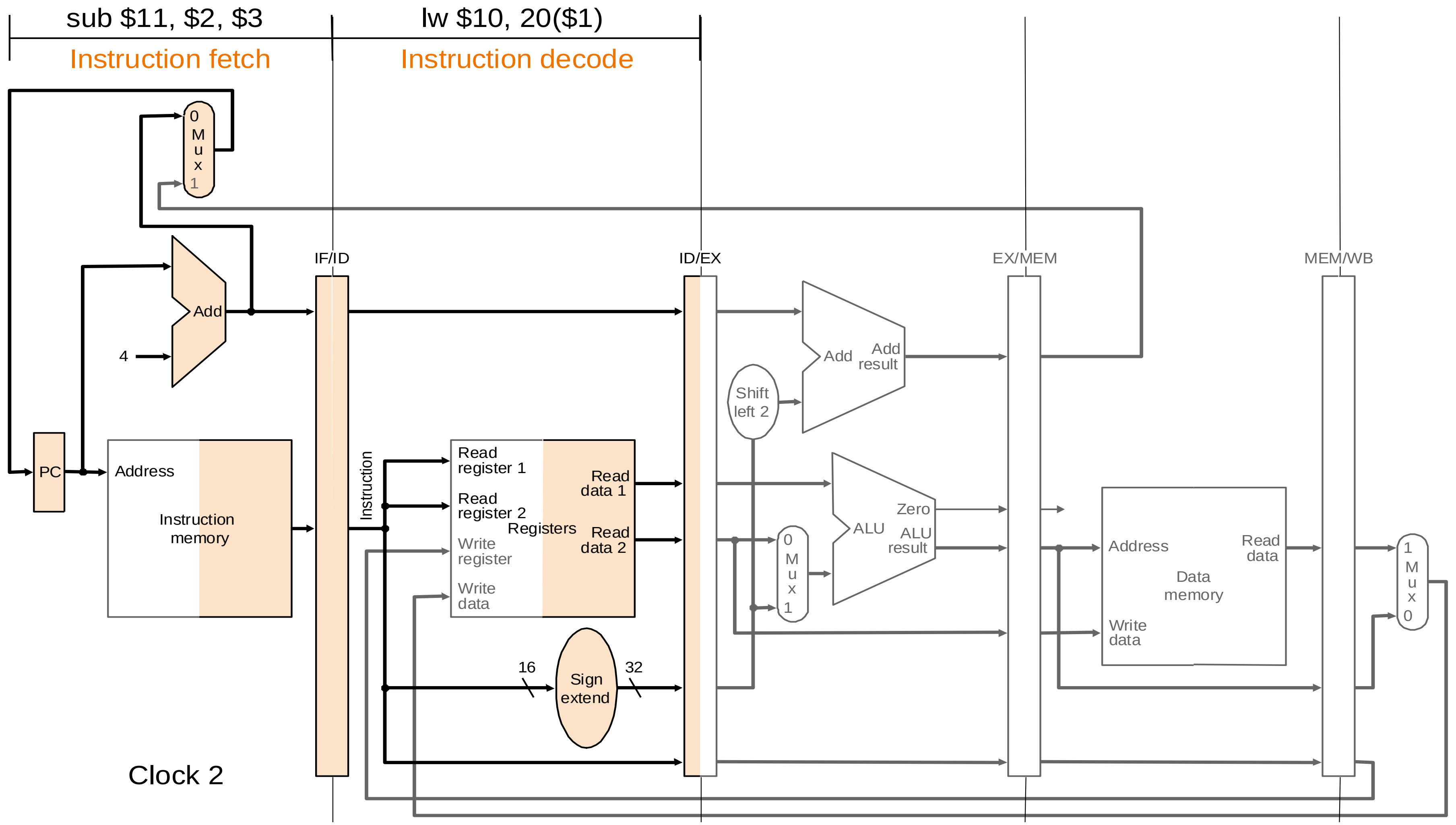
Cycle 1



Pipelined Datapath

Example: Pipelining `lw $10, 20($1)` and `sub $11, $2, $3`

Cycle 2

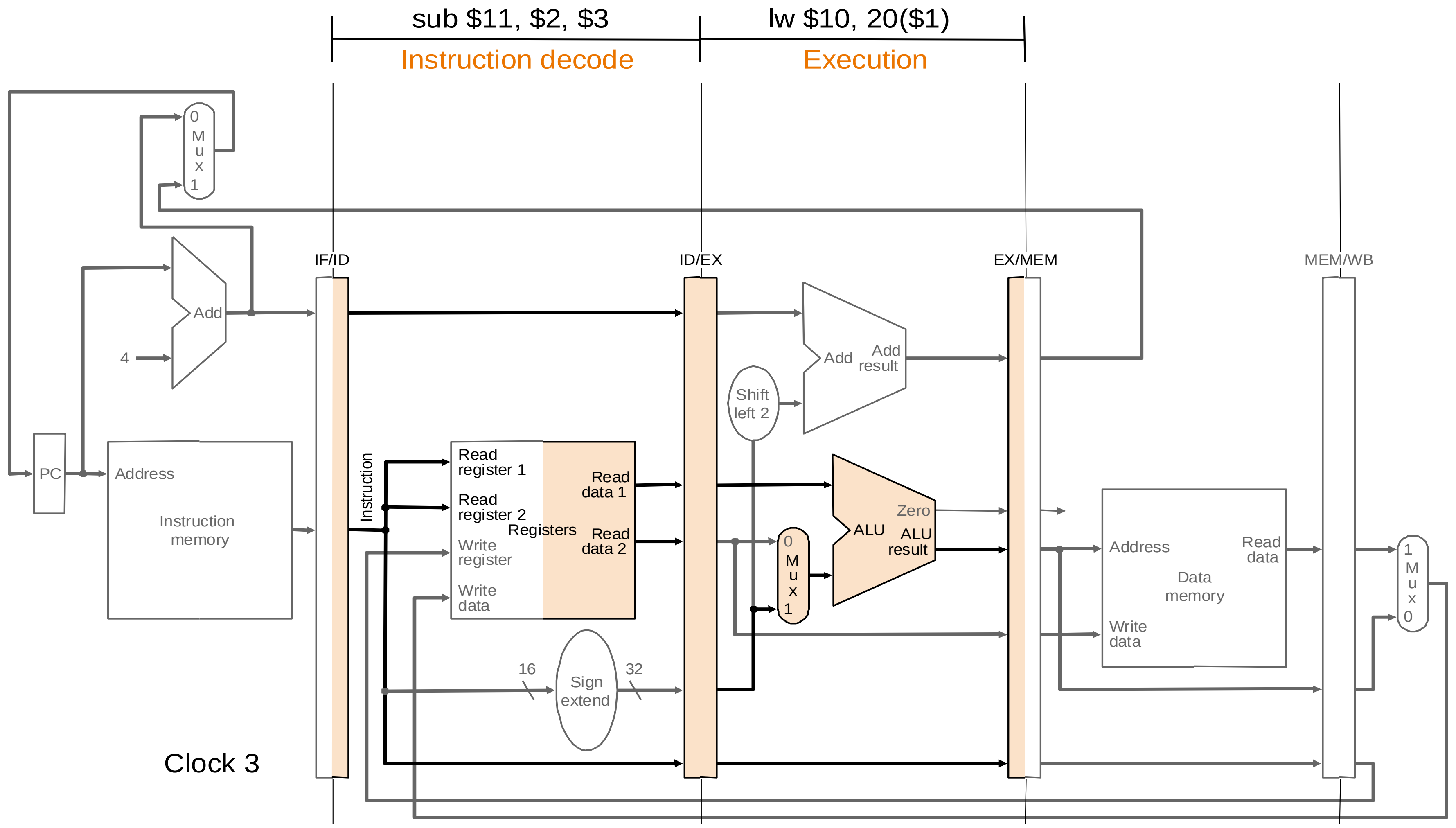


University of Nottingham
UK | CHINA | MALAYSIA

Pipelined Datapath

Example: Pipelining `lw $10, 20($1)` and `sub $11, $2, $3`

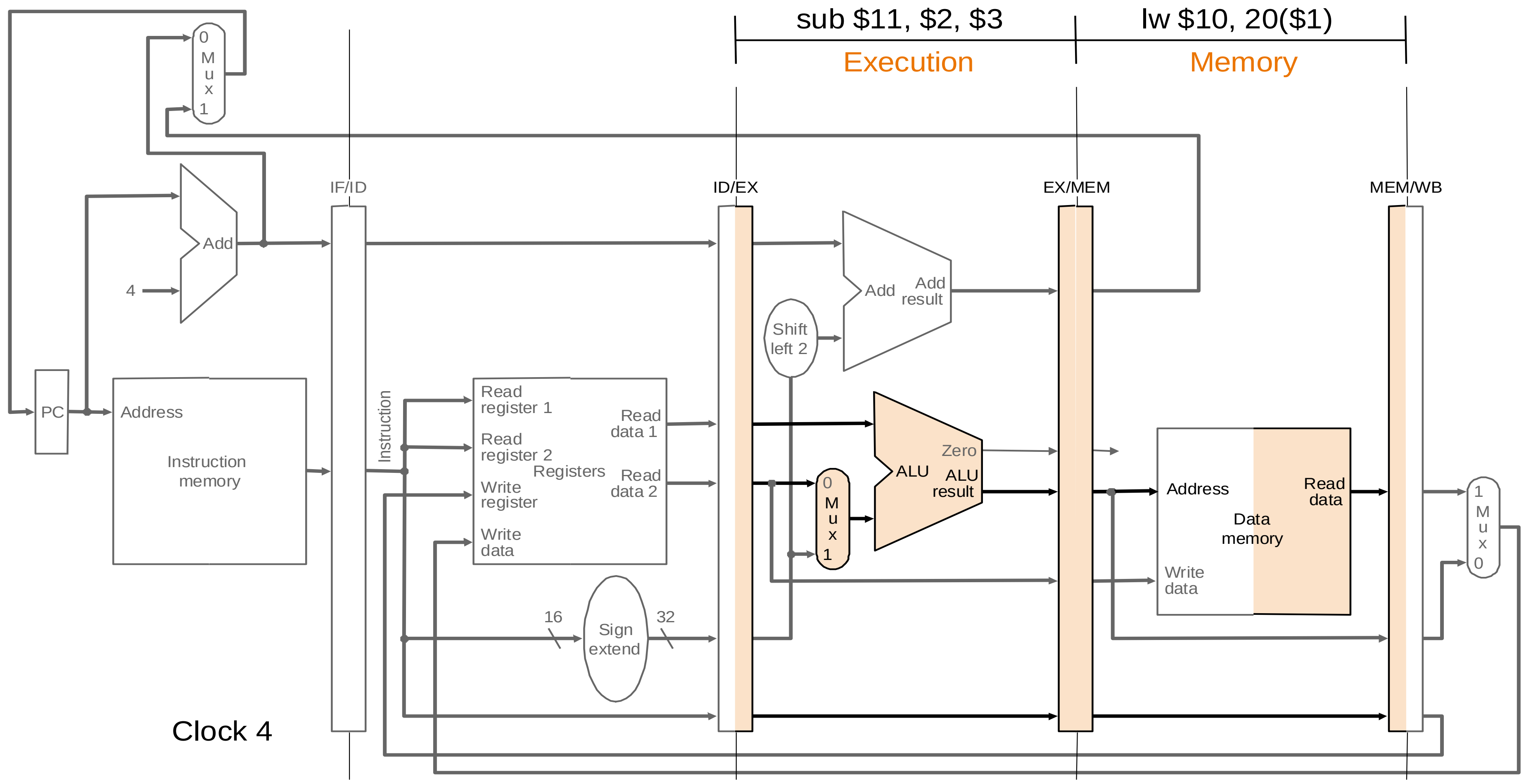
Cycle 3



Pipelined Datapath

Example: Pipelining `lw $10, 20($1)` and `sub $11, $2, $3`

Cycle 4

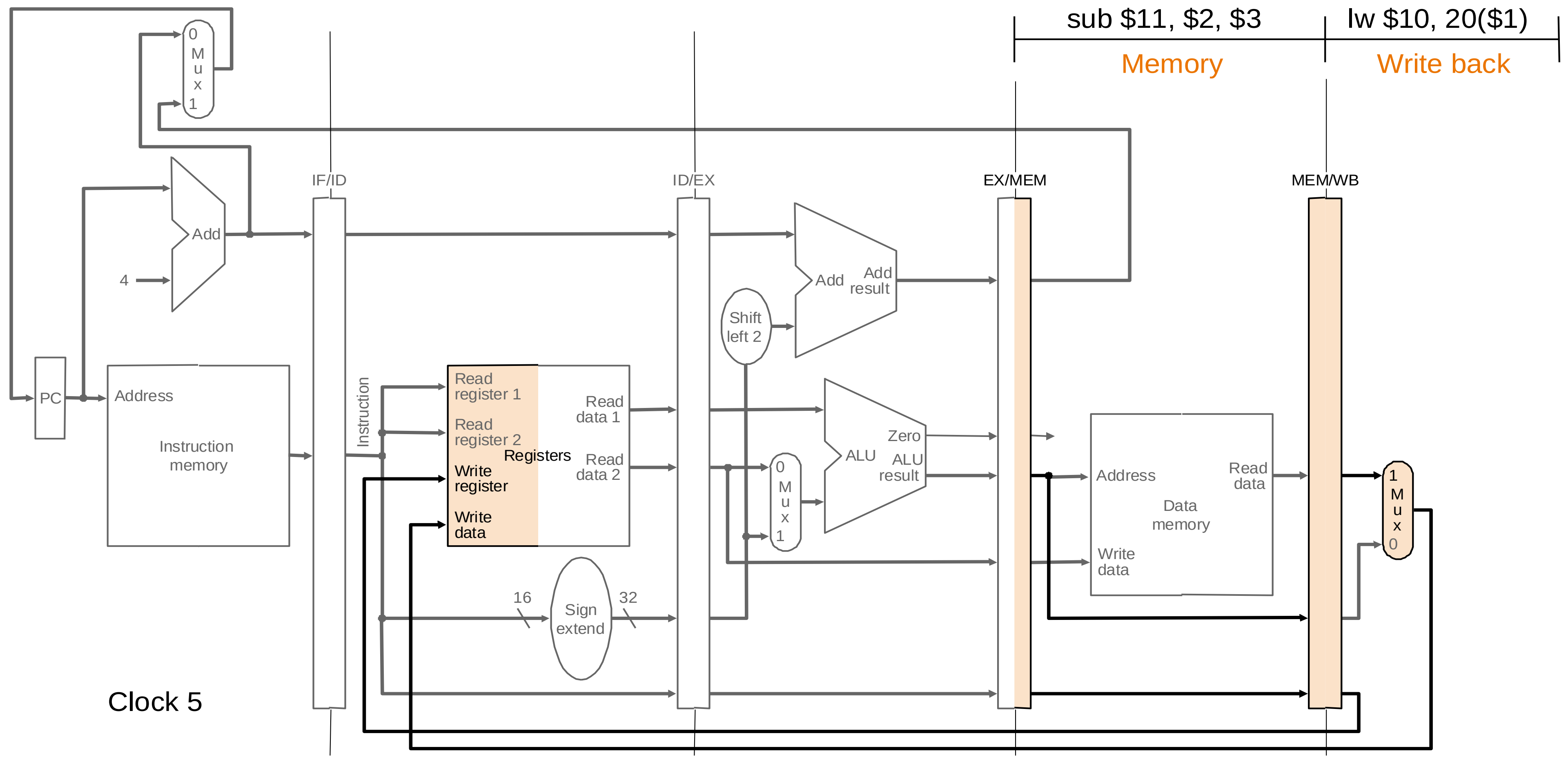


University of Nottingham
UK | CHINA | MALAYSIA

Pipelined Datapath

Example: Pipelining `lw $10, 20($1)` and `sub $11, $2, $3`

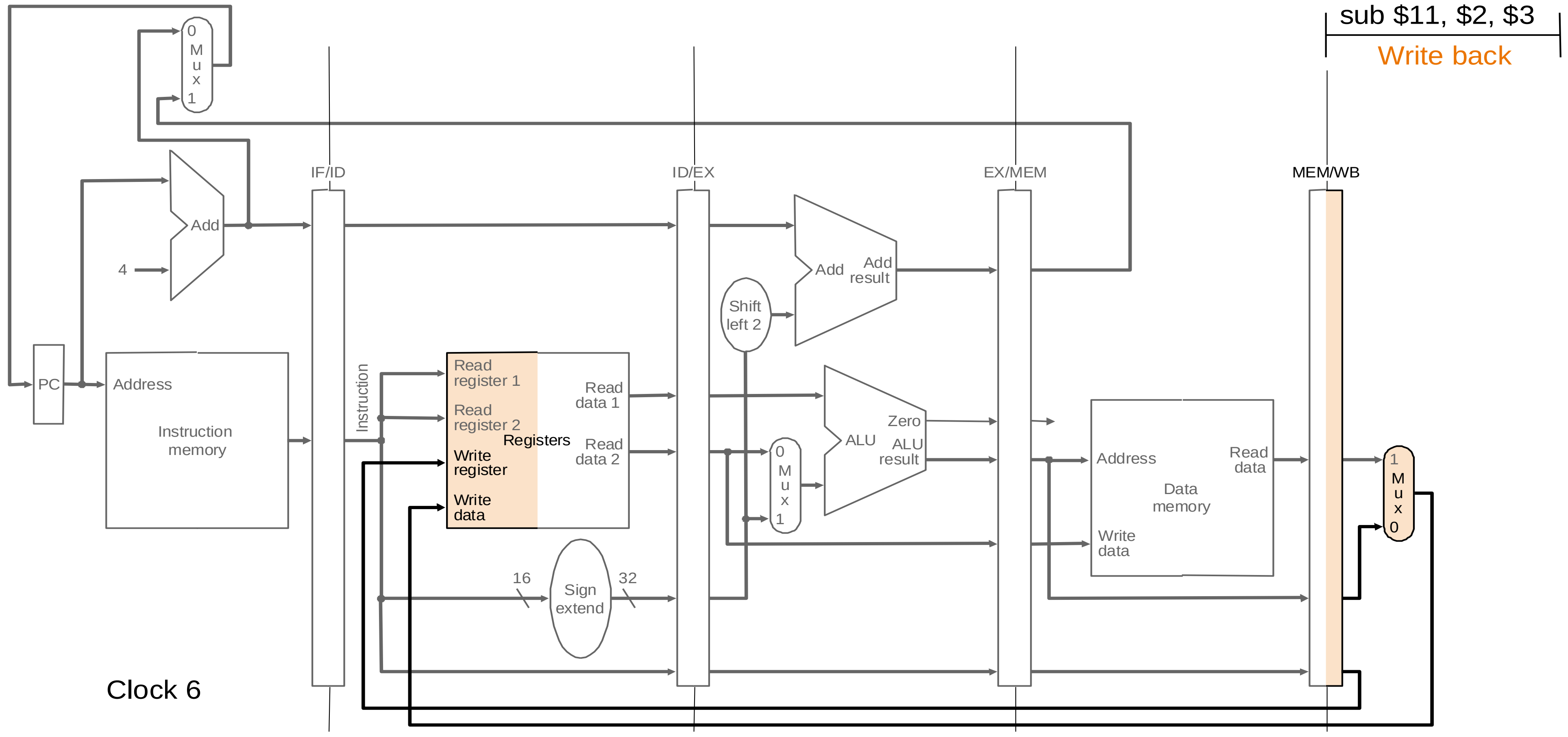
Cycle 5



Pipelined Datapath

Example: Pipelining `lw $10, 20($1)` and `sub $11, $2, $3`

Cycle 6



Today's outline

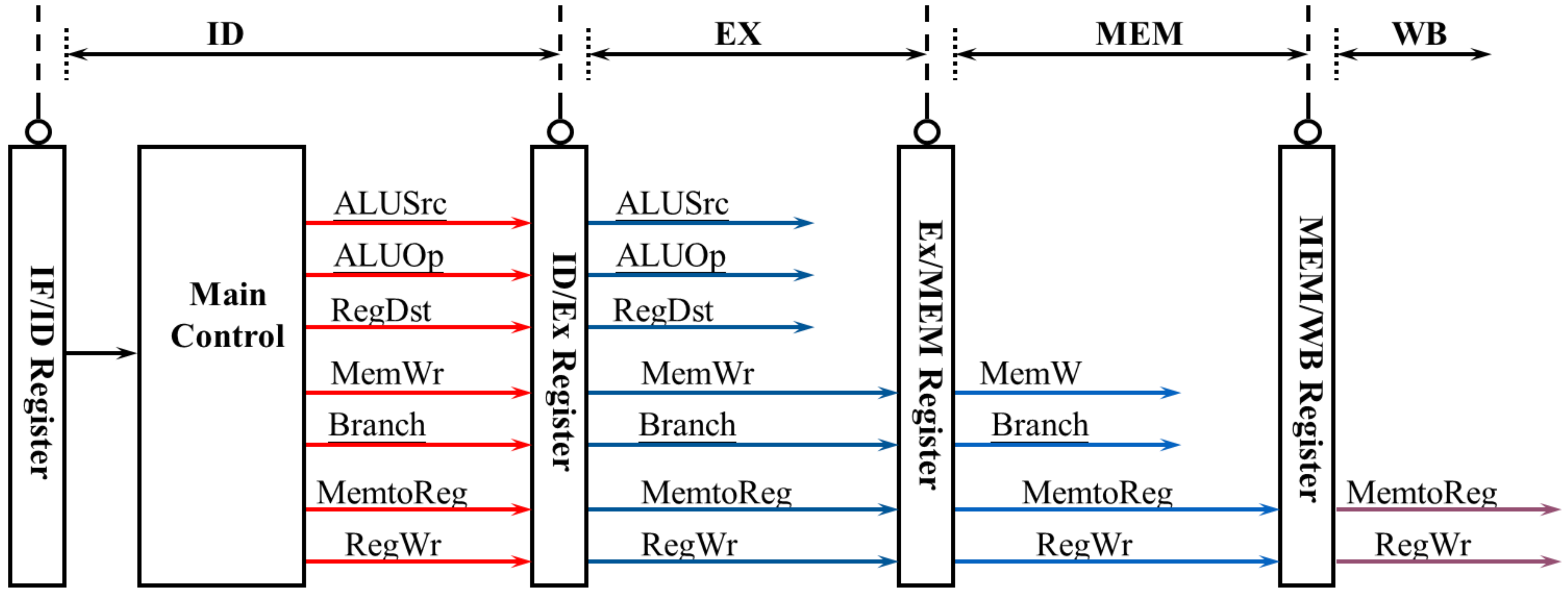
- An overview of pipelining
- A pipelined datapath
- Pipelined control
- Hazards: types of hazard
    - Handling data hazards
    - Handling branch hazards

Pipelined Control

Control signals are pipelined just like the datapath

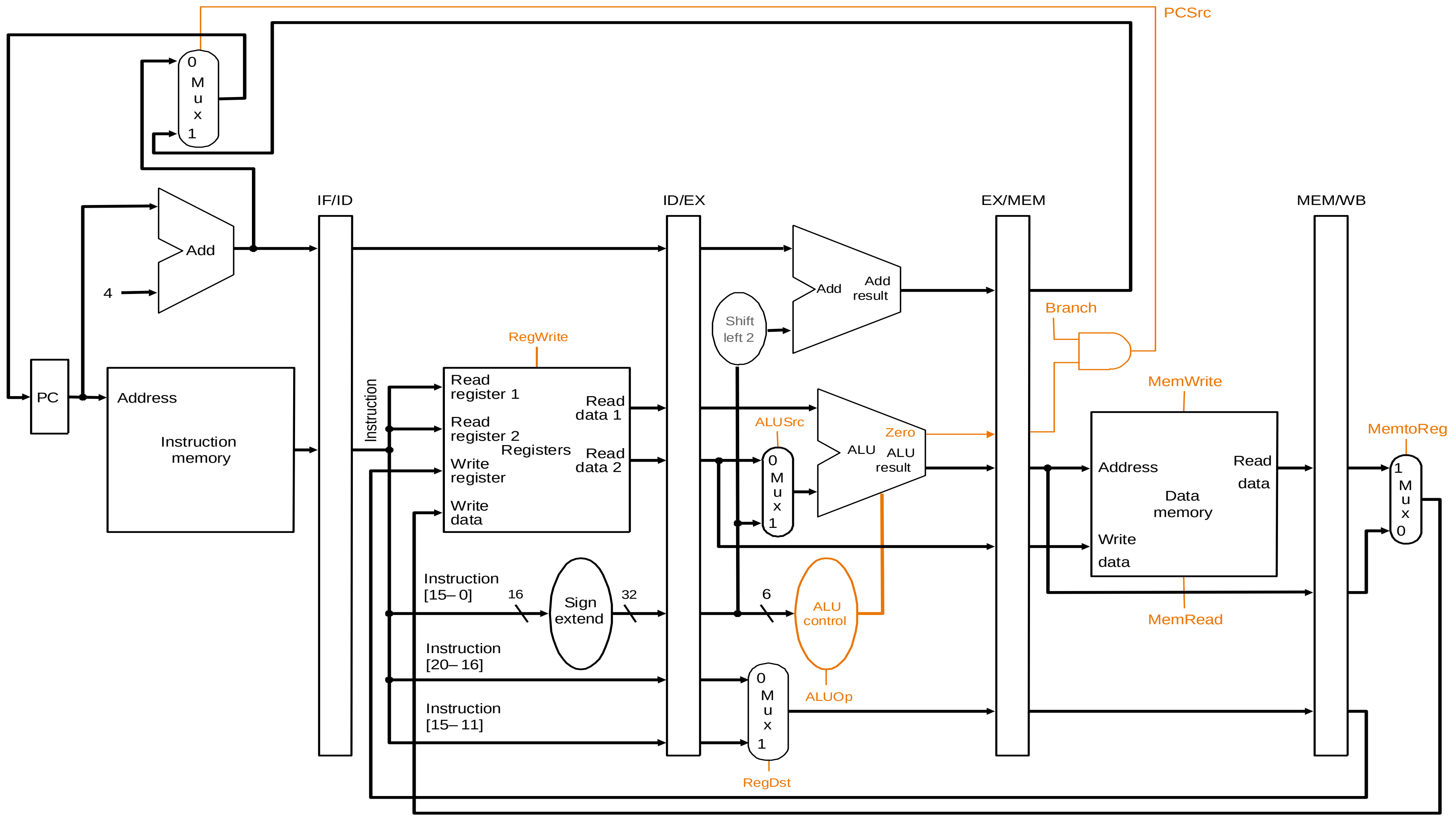
- Main controller generates control signals during ID

- Signals for EX (ExtOp, ALUSrc, ...) are used 1 cycle later
- Signals for MEM (MemWr, Branch) are used 2 cycles later
- Signals for WB (MemtoReg, MemWr) are used 3 cycles later



Pipelined Control

Control signals are pipelined just like the datapath



Summary of pipeline basics

- Pipelining is a fundamental concept
  - Multiple steps using distinct resources
  - Pipelined instruction processing
    - Start next instruction while working on the current one
    - Limited by length of longest stage
    - Need to detect and resolve hazards
- What makes it easy in MIPS?
  - All instructions are of the same length
  - Just a few instruction formats
- What makes pipelining hard? hazards

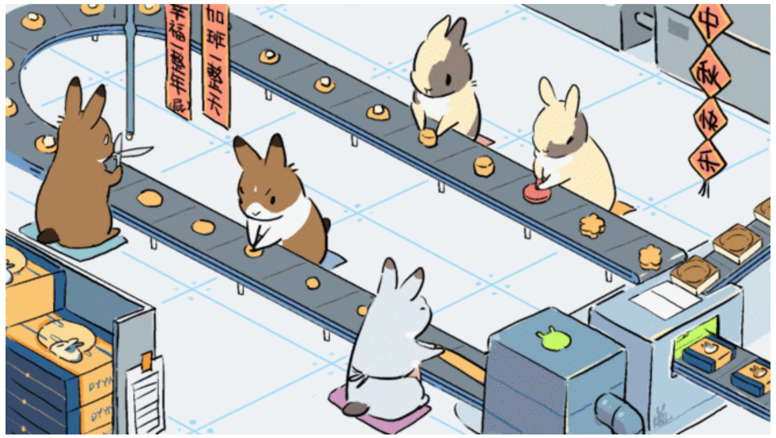


Image Link

Today's outline

- ✓ An overview of pipelining
- ✓ A pipelined datapath
- ✓ Pipelined control
- ✓ Hazards: types of hazard
  - ✓ Handling data hazards
  - ✓ Handling branch hazards

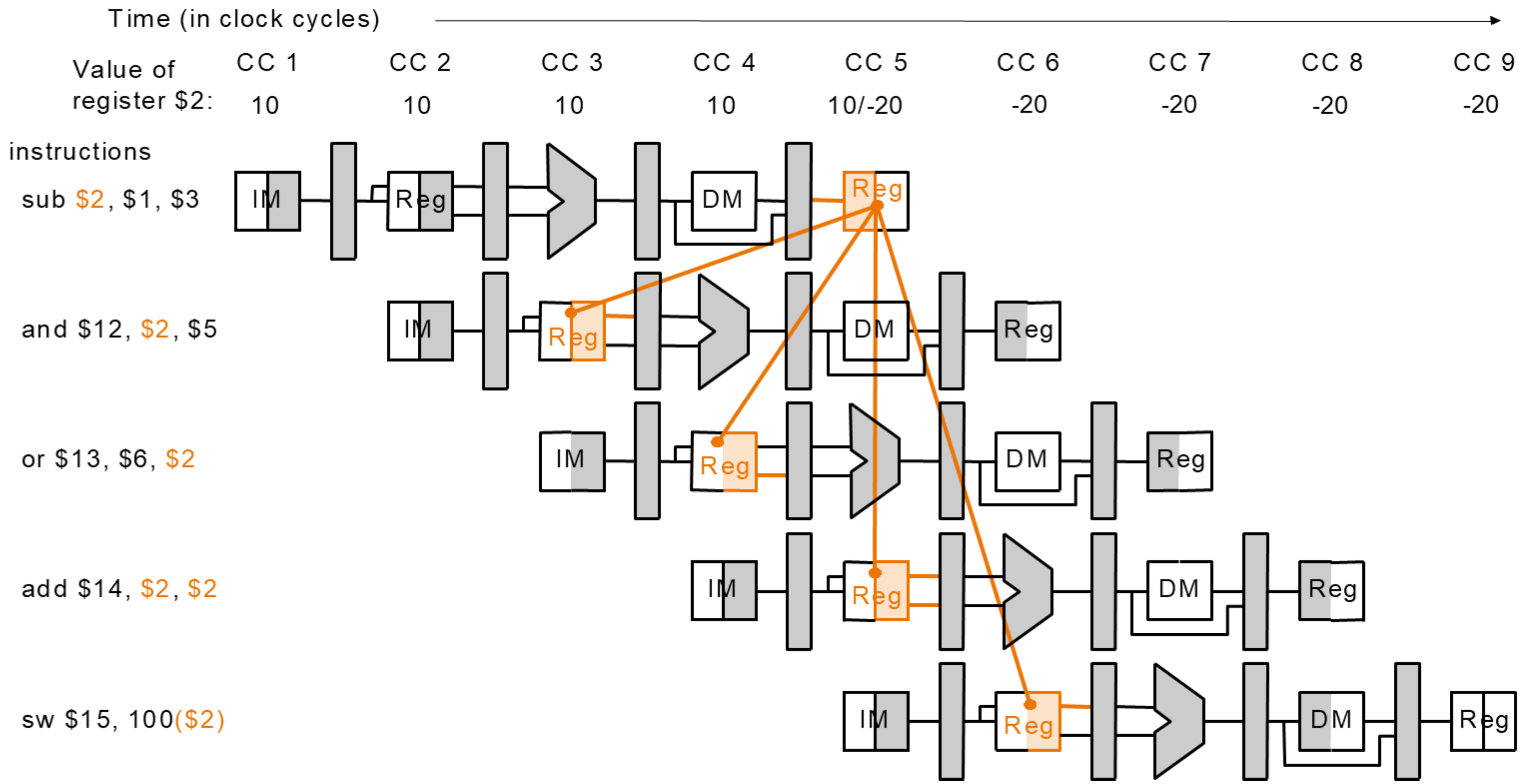
Pipeline Hazards

Pipeline Hazards:

- Structural hazards
  - Attempt to use the same resource in two different ways at the same time
- Data hazards
  - Attempt to use the data value before available
  - Instruction depends on result of prior instruction still in the pipeline
- Control hazards
  - Attempt to make decision before condition is evaluated
  - Most often happens in branch instructions
- Can always resolve hazards by waiting
  - Pipeline controller must detect the hazard
  - Take action (or delay action) to resolve hazards

Data Hazards

- Data hazards in MIPS
  - Attempt to use the data value before available
  - Instruction depends on result of prior instruction still in the pipeline
  - RAW (Read-after-write)



Today's outline

- ✓ An overview of pipelining
- ✓ A pipelined datapath
- ✓ Pipelined control
- ✓ Hazards: types of hazard
  - ✓ Handling data hazards
  - ✓ Handling branch hazards

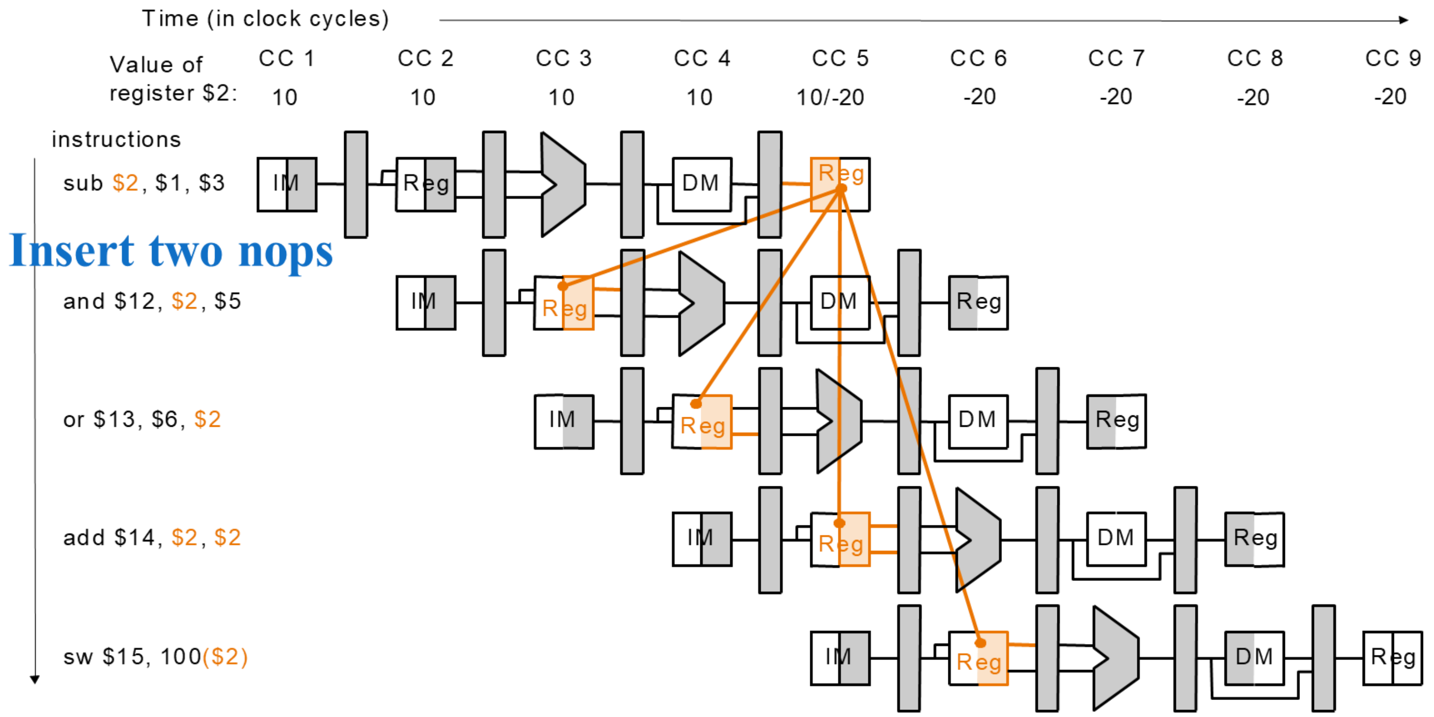
Data Hazards

Method 1: Inserting NOP

- Software solution, use compiler to guarantee no hazards
- Where do we insert the NOPs?

```
sub     $2,  $1, $3
and     $12, $2, $5
or      $13, $6, $2
add     $14, $2, $2
sw      $15, 100($2)
```

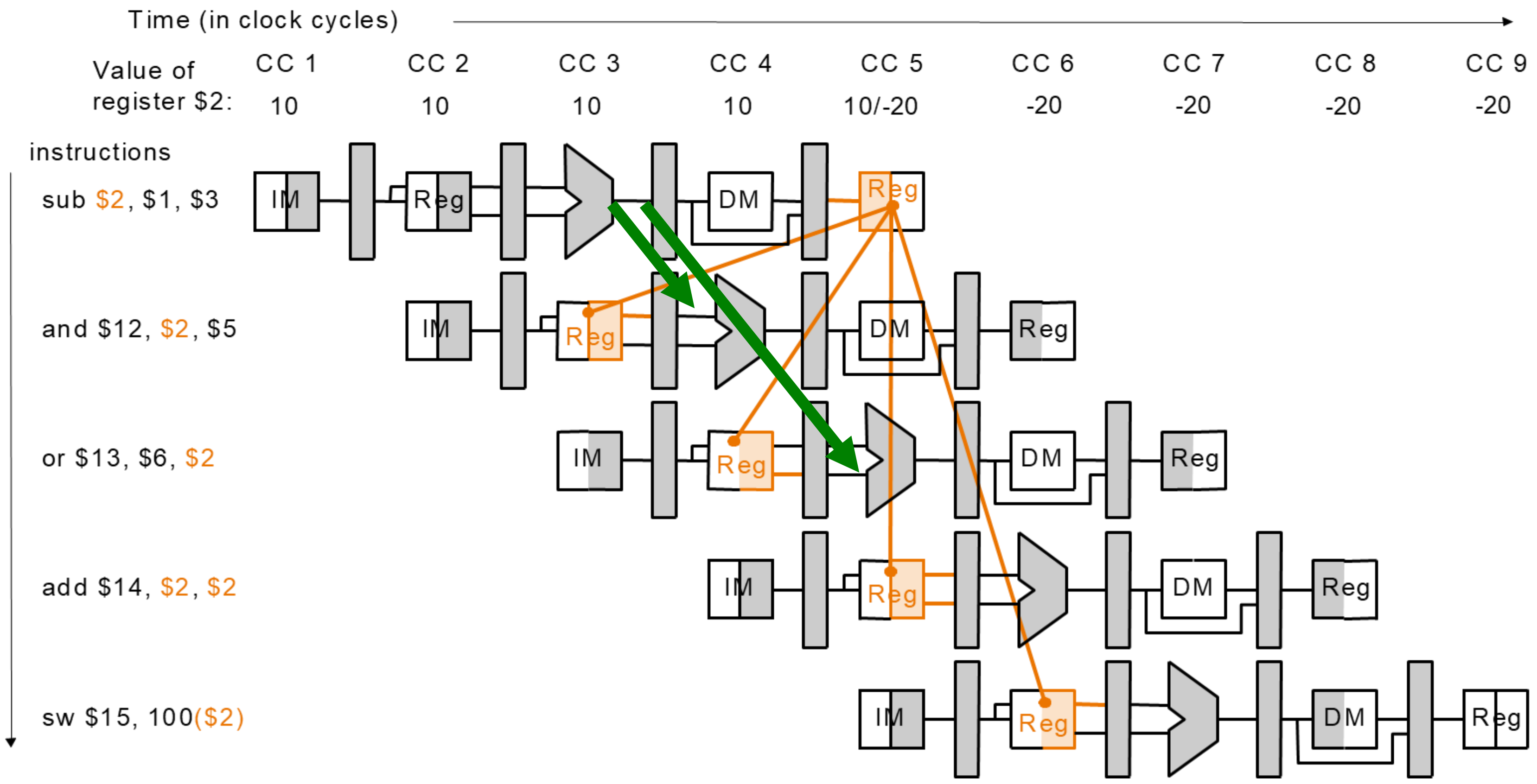
- Problem: this really slows the CPU down!



Data Hazards

Method 2: Forwarding (for R-type instructions)

- Hardware solution, feed the $2 values directly to ALU read-in ports. Don't have to wait until the WB stage.

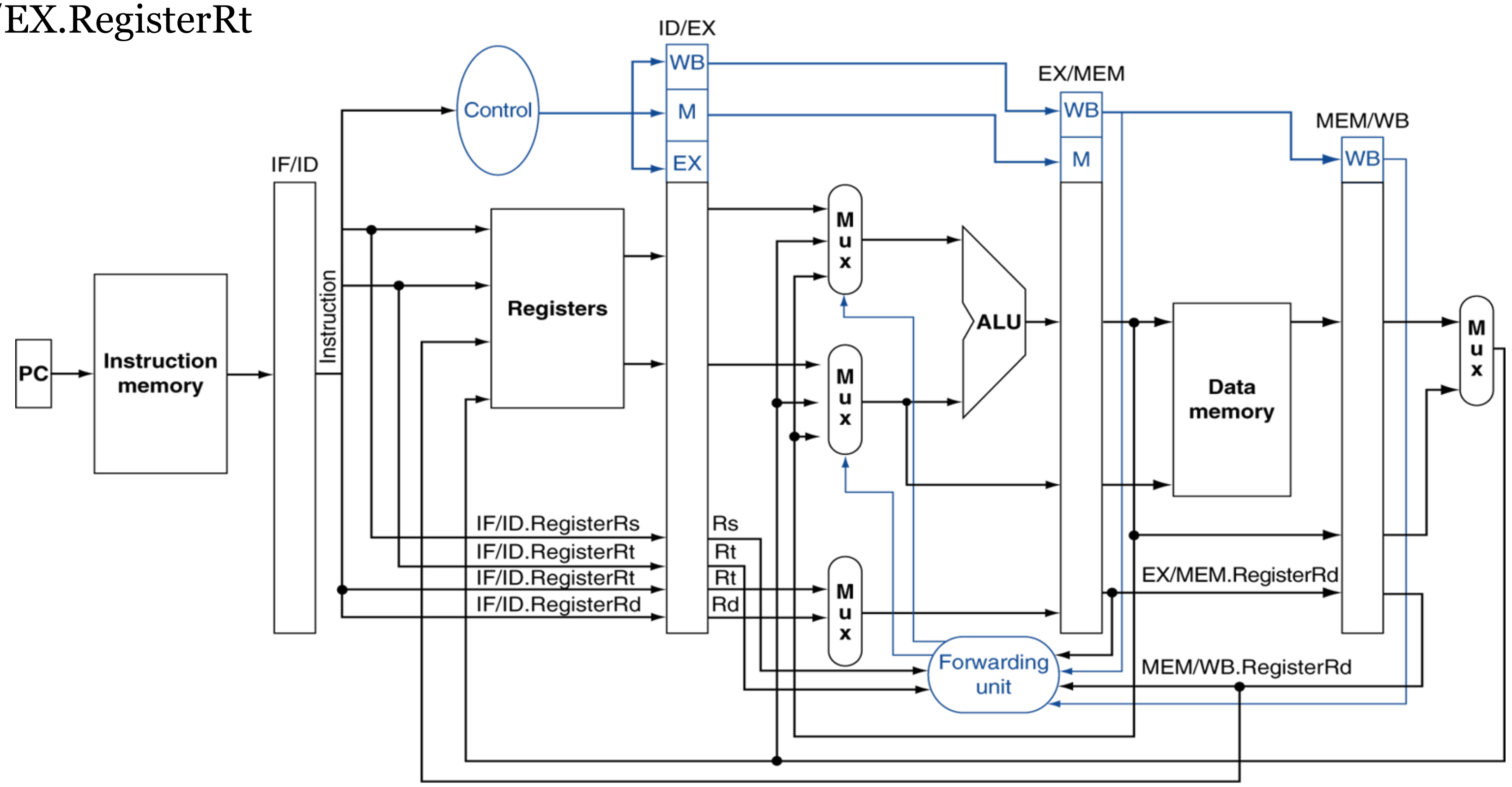


Data Hazards

Forwarding conditions

- Hazard conditions:
  - EX/MEM.RegisterRd = ID/EX.RegisterRs
  - EX/MEM.RegisterRd = ID/EX.RegisterRt
  - MEM/WB.RegisterRd = ID/EX.RegisterRs
  - MEM/WB.RegisterRd = ID/EX.RegisterRt

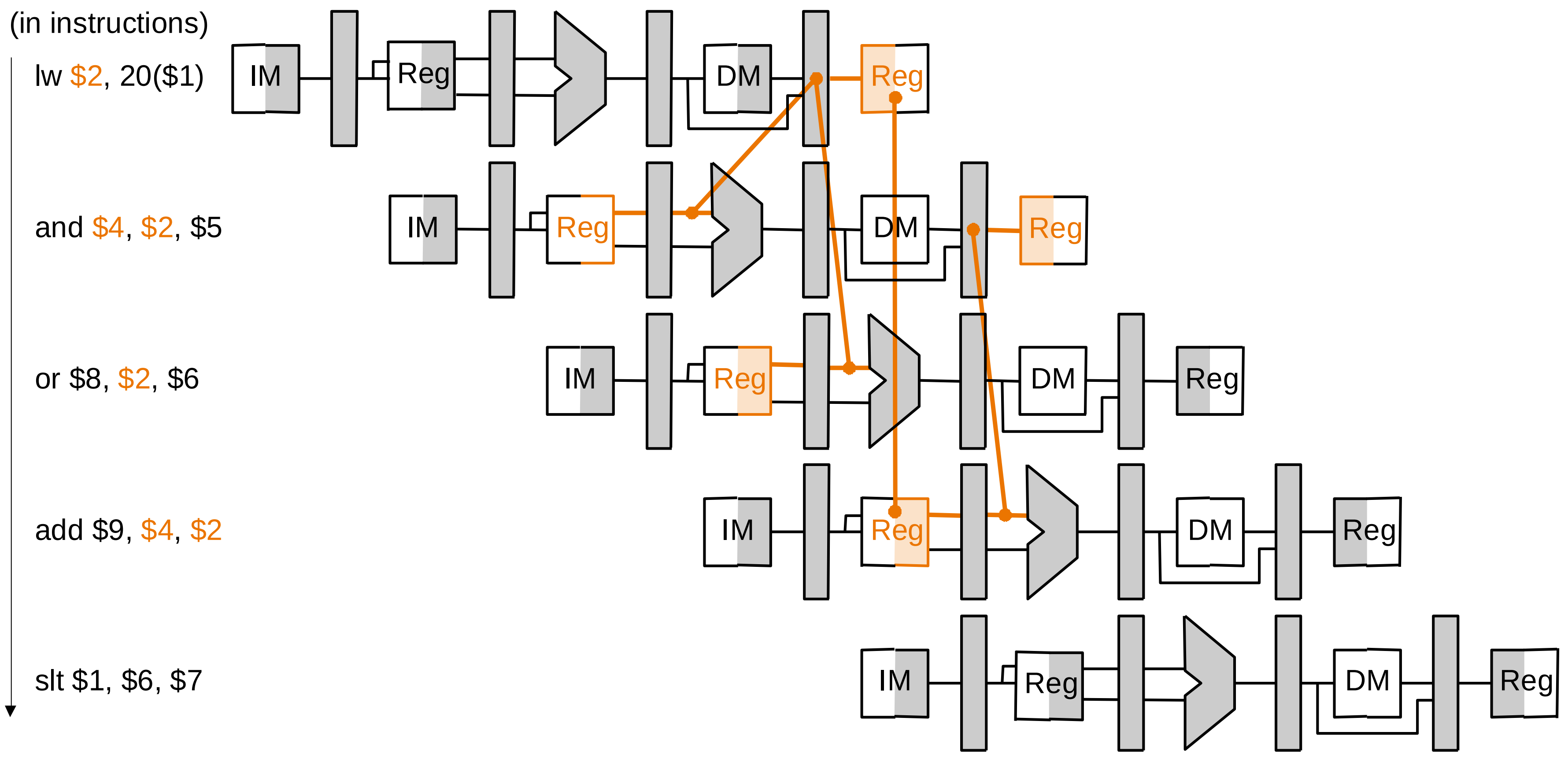
- Two optimizations:
  - Don't forward if instruction does not write register => check if RegWrite is asserted
  - Don't forward if destination register is $0 => check if RegisterRd = 0



Data Hazards

Method 3: Stalling (for load instructions)

- `lw` can still cause a data hazard, if it is followed by an instruction to read the loaded register ($2 in this example)

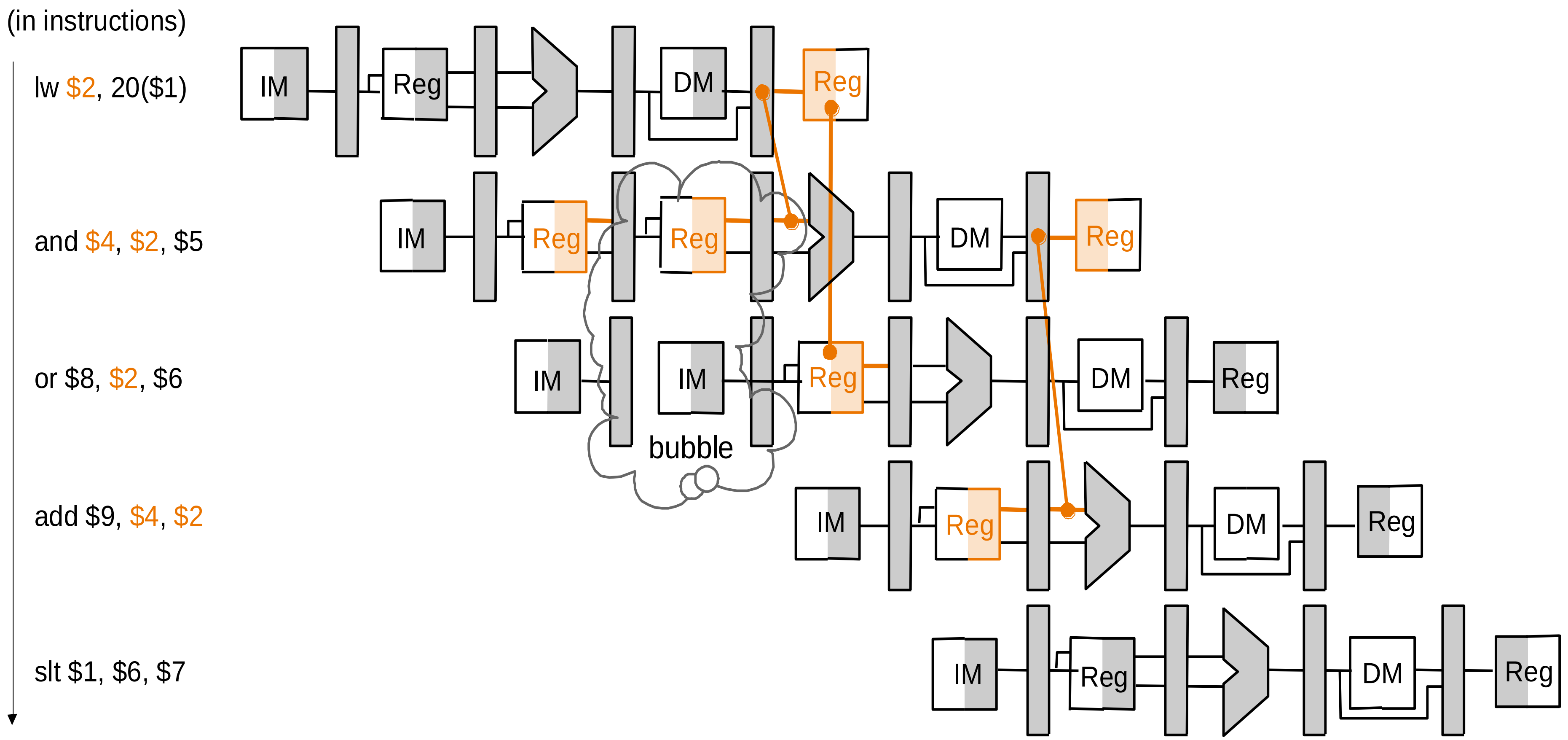


University of Nottingham UK | CHINA | MALAYSIA

Data Hazards

Stalling

- Stall pipeline by “freezing” or duplicating the control/data values. Namely, inserting a bubble in the pipeline.



University of Nottingham
UK | CHINA | MALAYSIA

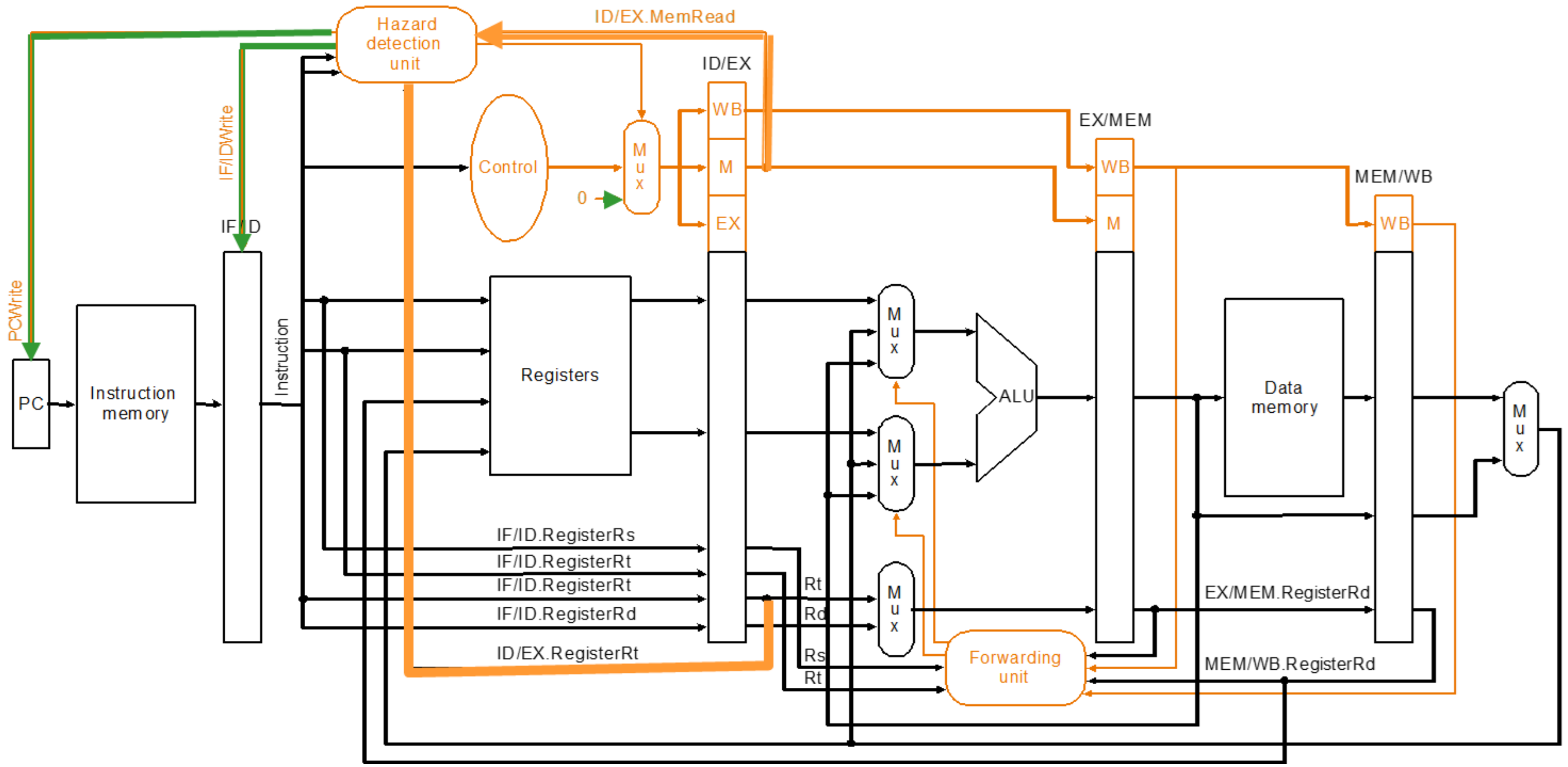
Data Hazards

Stalling conditions

- if (ID/EX.MemRead *AND*
  ((ID/EX.RegisterRt = IF/ID.RegisterRs) *OR*
  (ID/EX.RegisterRt = IF/ID.registerRt))

- ID/EX.MemRead=1 indicates a load instruction

- Stall instruction in IF and ID: not change PC and IF/ID => the stages re-execute the instructions



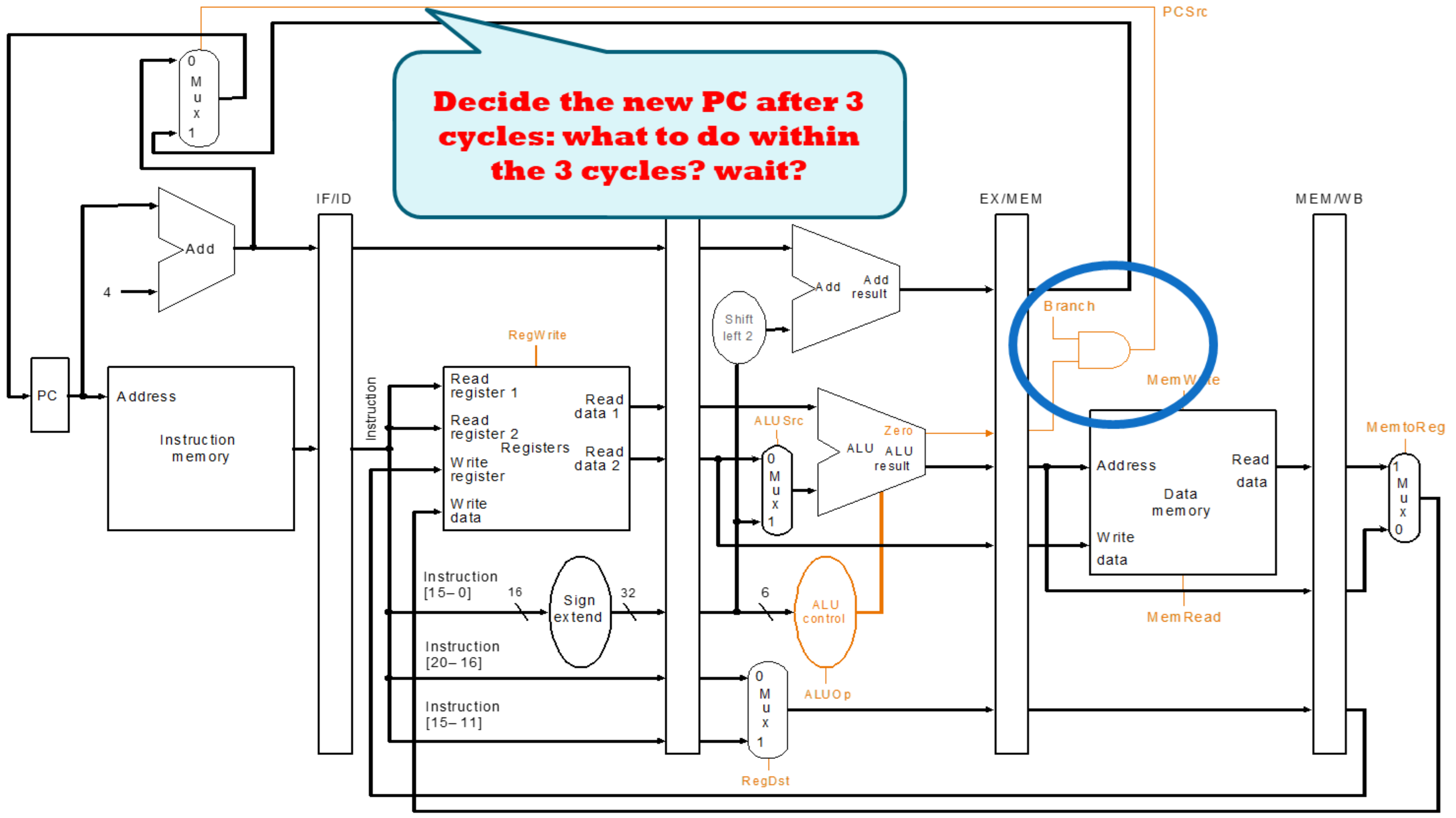
Today’s outline

- ✓ An overview of pipelining
- ✓ A pipelined datapath
- ✓ Pipelined control
- ✓ Hazards: types of hazard
  - ✓ Handling data hazards
  - ✓ Handling branch hazards

University of Nottingham UK | CHINA | MALAYSIA

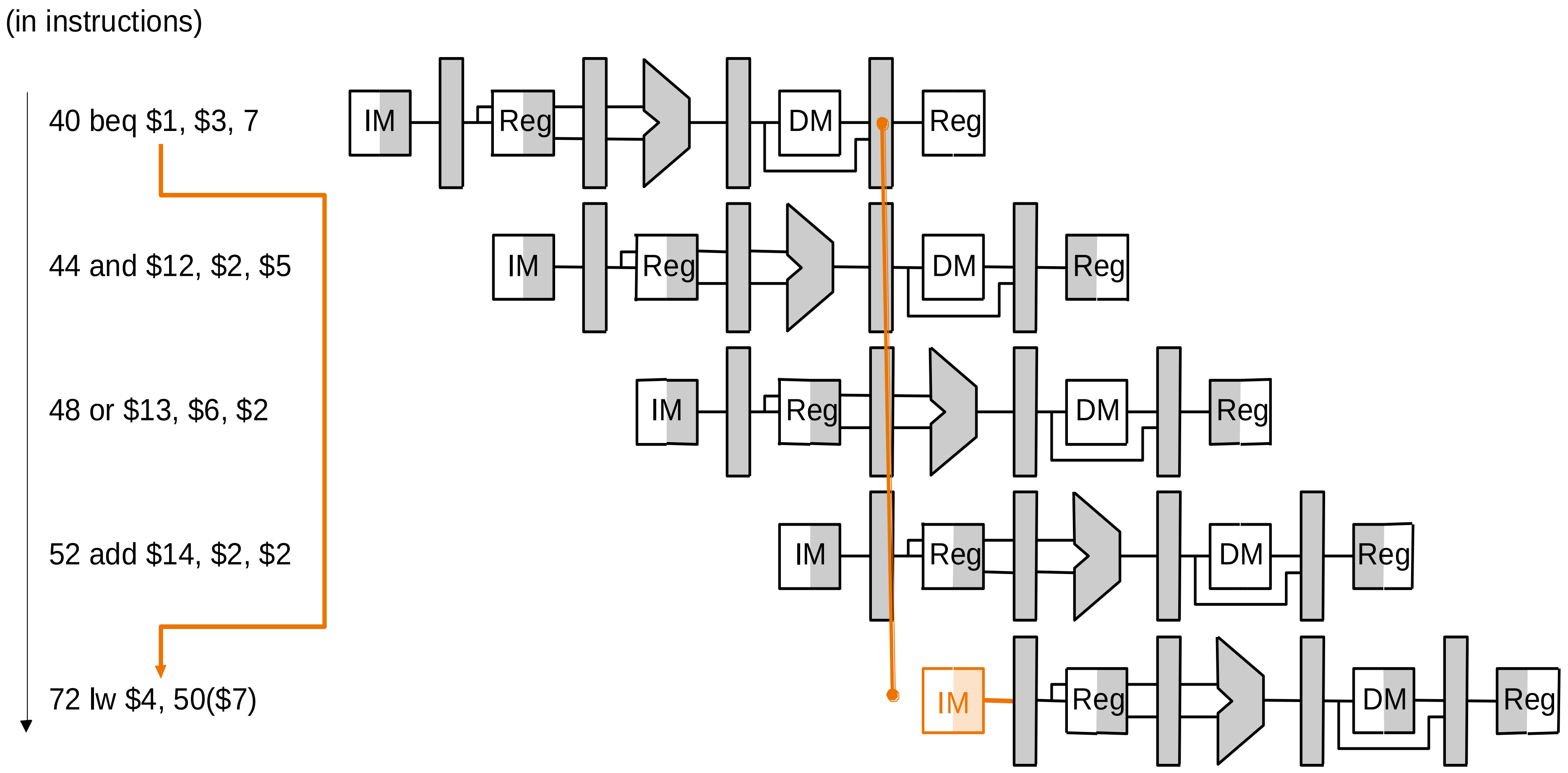
Control Hazards

- Pipeline datapath with Control hazards



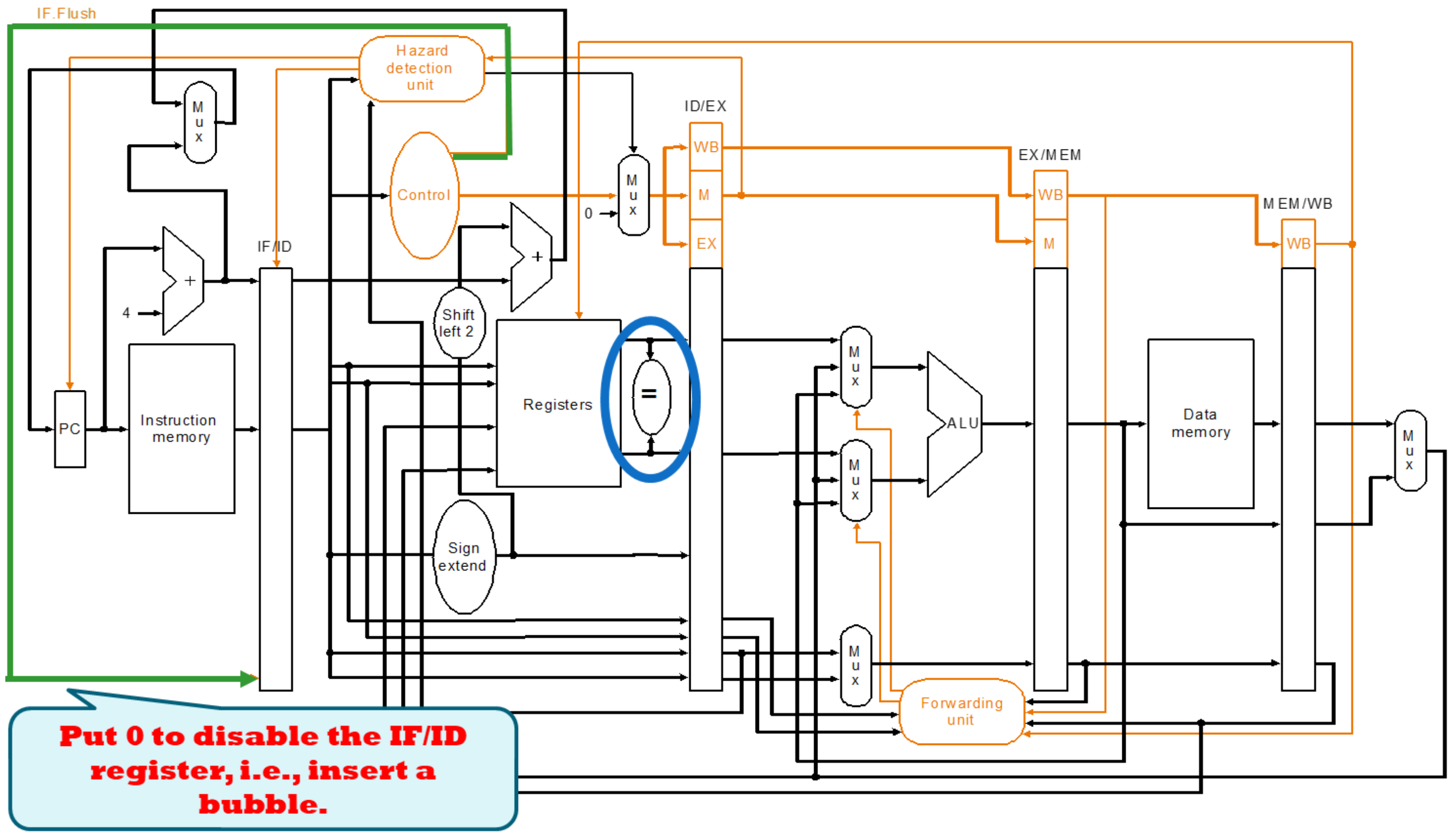
Data Hazards

- Pipeline datapath with Control hazards
  - When decided to branch, 3 cycles already passed!



Control Hazards

- Remedy: Reduce delay of taken branch by moving branch execution earlier in the pipeline
  - Move up branch address calculation to ID
  - Check branch equality at ID (using XOR) by comparing the two registers read during ID
  - Branch decision made at ID => only one instruction to waste
  - Add a control signal, IF.Flush, to zero instruction field of IF/ID => making the instruction an NOP



Dynamic Branch Prediction

- In deeper and superscalar pipelines, branch penalty is more significant
- Use dynamic prediction (e.g. loop)
  - Branch prediction buffer (i.e., branch history table)
  - Indexed by recent branch instruction addresses
  - Stores outcome (taken/not taken)
  - To execute a branch
    - Check table, expect the same outcome
    - Start fetching from fall-through or target
    - If wrong, flush pipeline and flip prediction

Think about it:

What AI techniques can be adopted to help branch prediction?

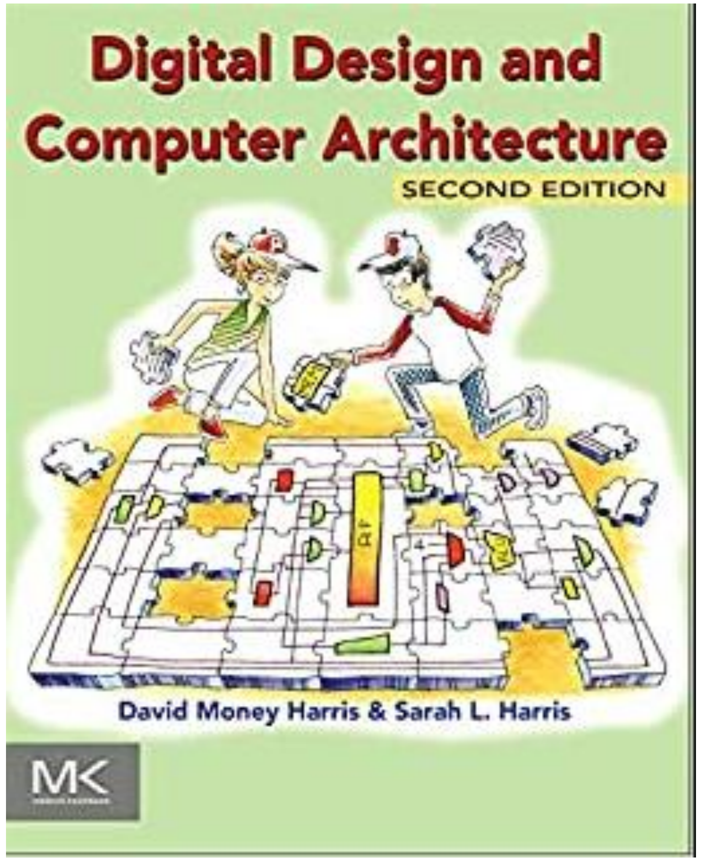
Summary

- Single cycle CPU => CPI=1, clock cycle time long
- Pipelined CPU => CPI > 1, needs additional hardware to control

- This module (as well as CSF) serves as the beginning of the track: Computer Engineering.
  - Industrial example on computer architecture: Alibaba's Xuantie (Black iron) 910, in RISC-V.

- Subsequent courses in this direction:
  - Real-time embedded systems, Operating systems, Multiprocessor systems, Parallel/Distributed computing, VLSI design, Advanced digital design, etc.

- Application fields
  - AI + Edge computing (AI on small devices), Internet-of-things (AI on everything connected), etc.


Digital Design and Computer Architecture
SECOND EDITION
David Money Harris & Sarah L. Harris
MK

University of Nottingham
UK | CHINA | MALAYSIA

Thank you.