Pipelining (part 1)

Chapter 6

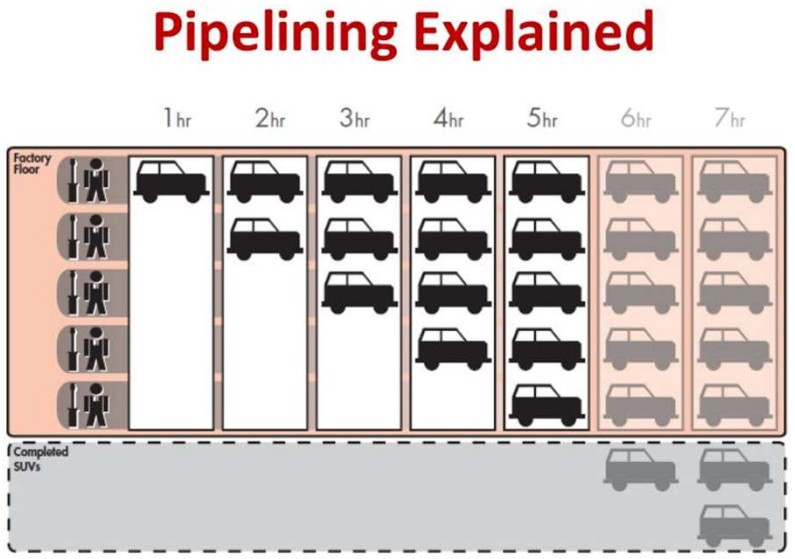
Pipelining

- ★ Pipelining Concept
  - ○ Pipelining in real world
  - ○ Lessons learned
- ★ Potential/Ideal SpeedUp
- ★ Pipelining in Processor
- ★ Representation of Pipeline
- ★ Potential Pipeline Hazards
  - ○ Structural Hazards
  - ○ Data Hazards
  - ○ Control Hazards
- ★ Solutions to Pipeline Hazards

Pipelining in real world

- ★ In industrial, pipelining is used everywhere in the production line.
- ★ The example is the SUV car assembly plant.
- ★ The idea is to start a new task as soon as the resources are available.

- ★ Find more examples of pipelining in real life.

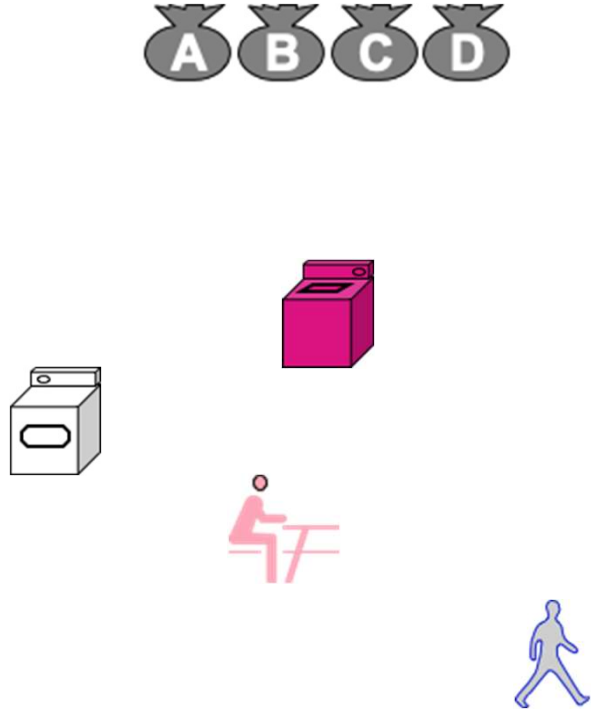


The lifecycle of an SUV in a pipelined factory

Picture taken from
https://images.slideplayer.com/24/7106280/slides/slide_6.jpg

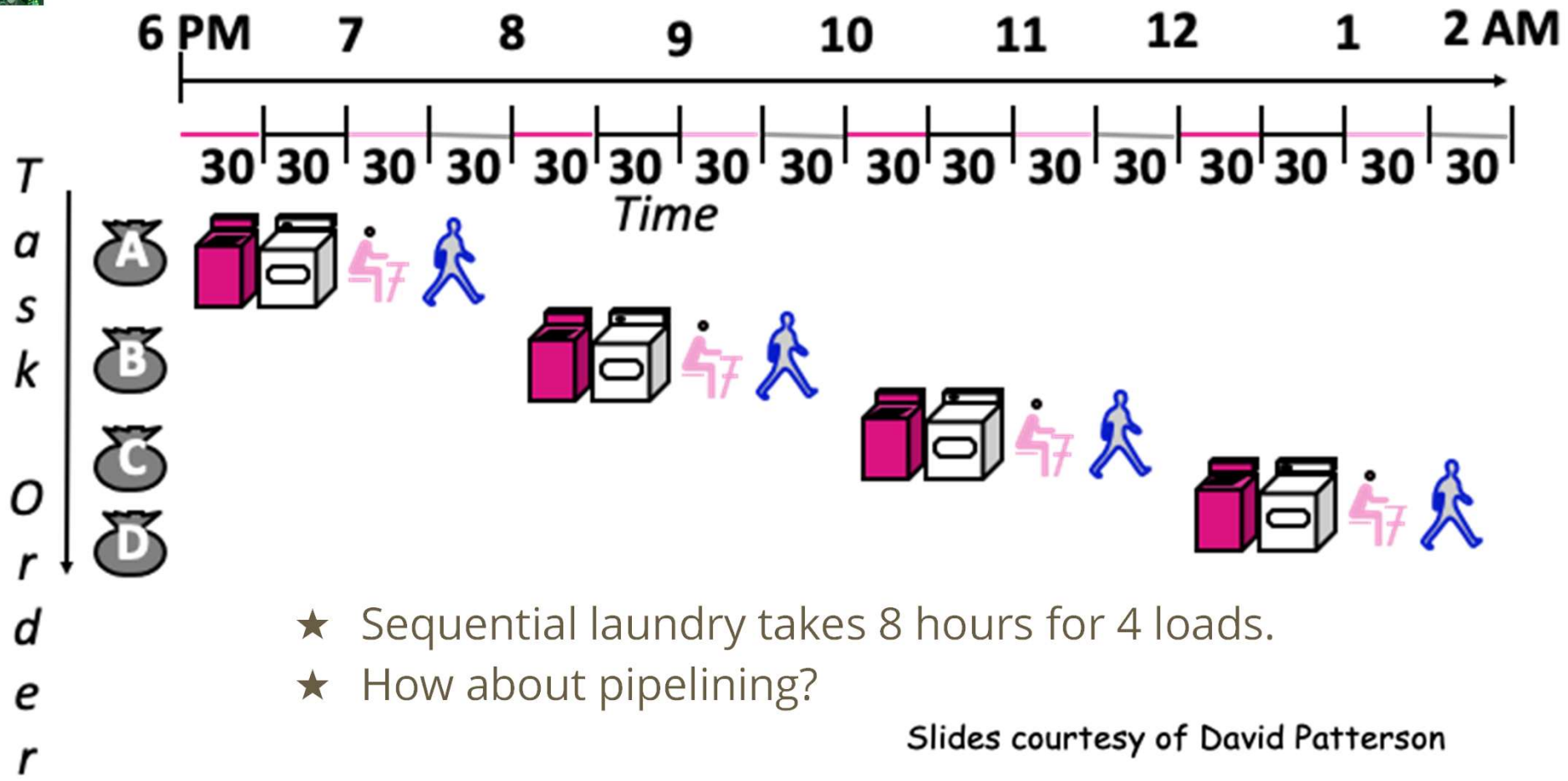
Pipelining in Laundry

- ★ Laundry Example
- ★ Ann, Brian, Cathy, Dave
  each have one load of clothes
  to wash, dry, and fold
- ★ Washer takes 30 minutes
- ★ Dryer takes 30 minutes
- ★ “Folder” takes 30 minutes
- ★ “Stasher” takes 30 minutes
  to put clothes into drawers



Slides courtesy of David Patterson

Sequential Laundry

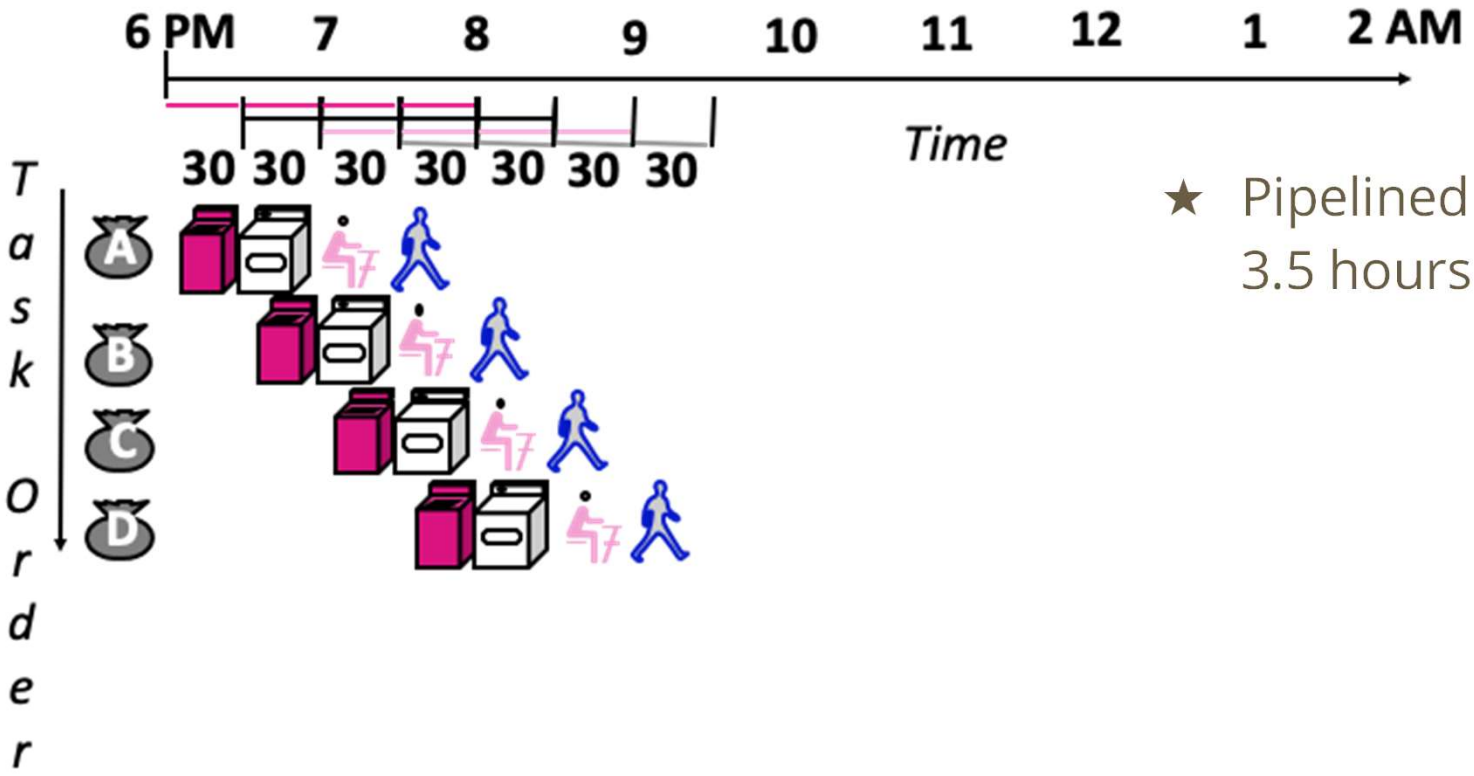


★ Sequential laundry takes 8 hours for 4 loads.

★ How about pipelining?

Slides courtesy of David Patterson

Pipelined Laundry (start work ASAP)

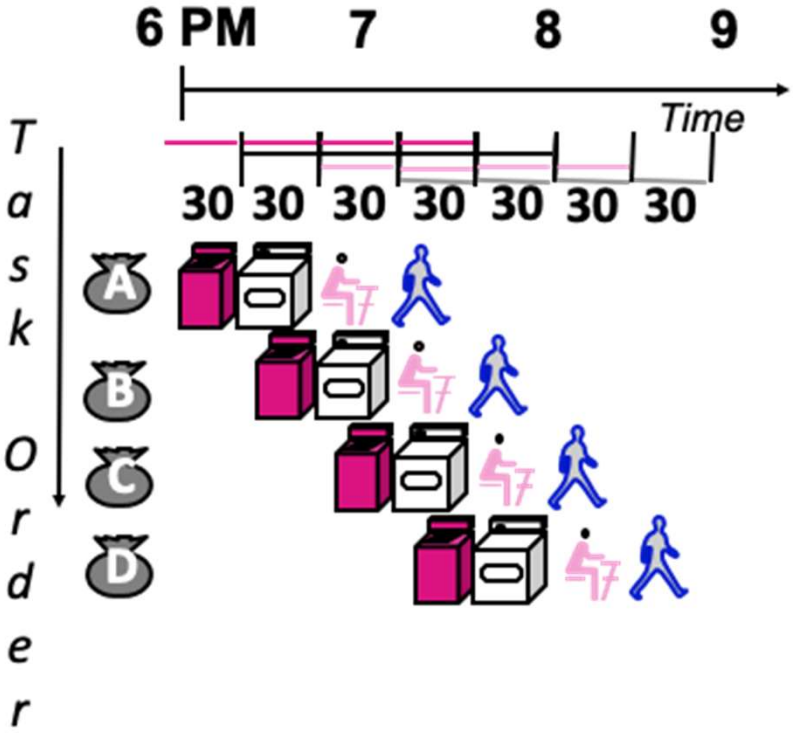


★ Pipelined Laundry taks only 3.5 hours for 4 loads.

Slides courtesy of David Patterson

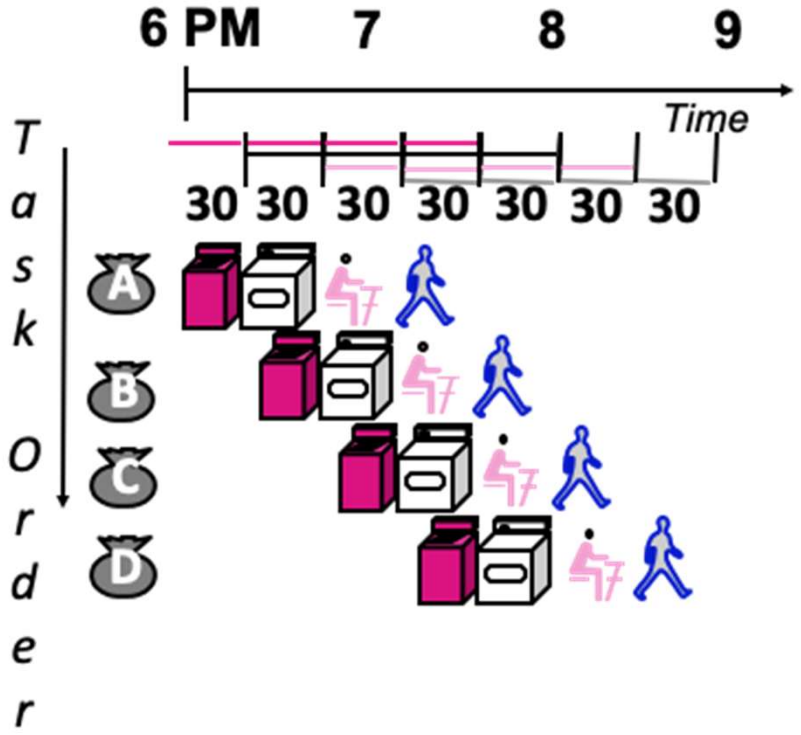
Lessons Learned (1)

- ★ Pipelining does not improve **latency** of a task.
- ★ Pipelining helps **throughput** of entire workload.
- ★ Multiple tasks run simultaneously on different resources.



Slides courtesy of David Patterson

Lessons Learned (2)



★ Pipeline rate limited by the **slowest** stage.

★ Unbalanced lengths of pipe stages reduces speedup.

★ Time to **fill** pipeline and time to **drain** pipeline reduces speedup.

Slides courtesy of David Patterson

Pop Quiz

- With pipelined Laundry, how long does it take to process k loads?
- Hint: Time to fill the pipe and Time for each load

Potential Speedup

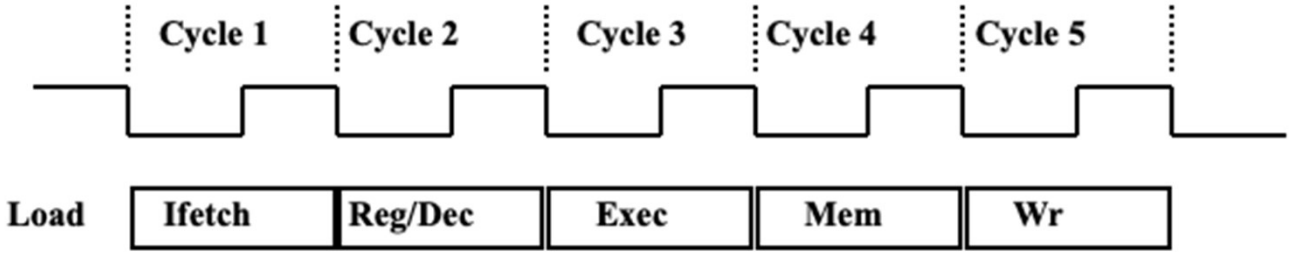
- ★ CPU Time = IC x CPI x Tc
- ★ Multiple Cycle Processor
  - CPU Time = k x 5 x [cycle time]
- ★ Pipelining
  - CPU Time = ([fill cycle] + (k-1) x 1) x [cycle time]
  - [fill cycle] is the number of stage
- ★ Assuming that k is large.
- ★ Potential Speedup is the **number of pipeline stages**

$$n = \frac{CPU\ TIME_{multiple\ cycle}}{CPU\ TIME_{pipeline}} = \frac{k \times 5 \times cycle\ time}{(5+(k-1)\times 1)\times cycle\ time}$$

$$n = \frac{k \times 5}{5+(k-1)\times 1} = \frac{5k}{5+k-1} = \frac{5k}{4+k}$$

$$n = \lim_{k \to \infty} \frac{5k}{5+k-1} = 5$$

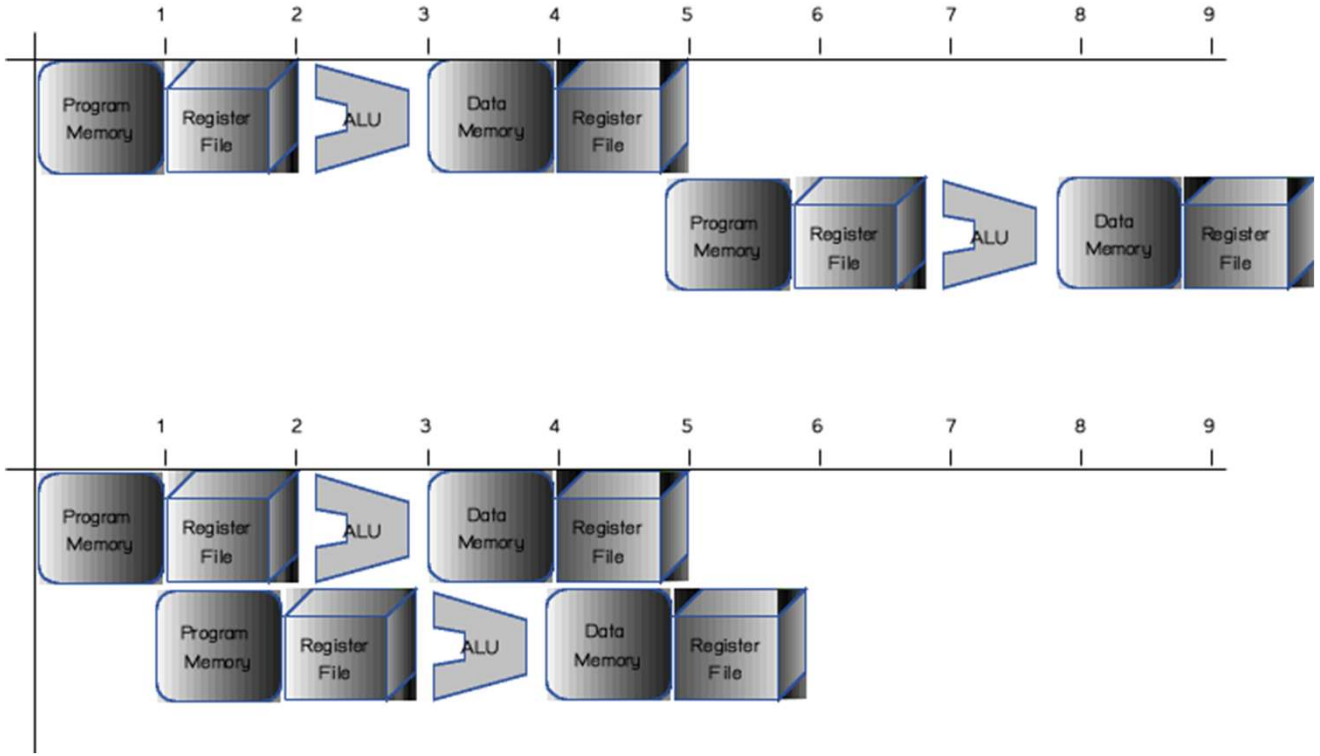
The Five stages of Load



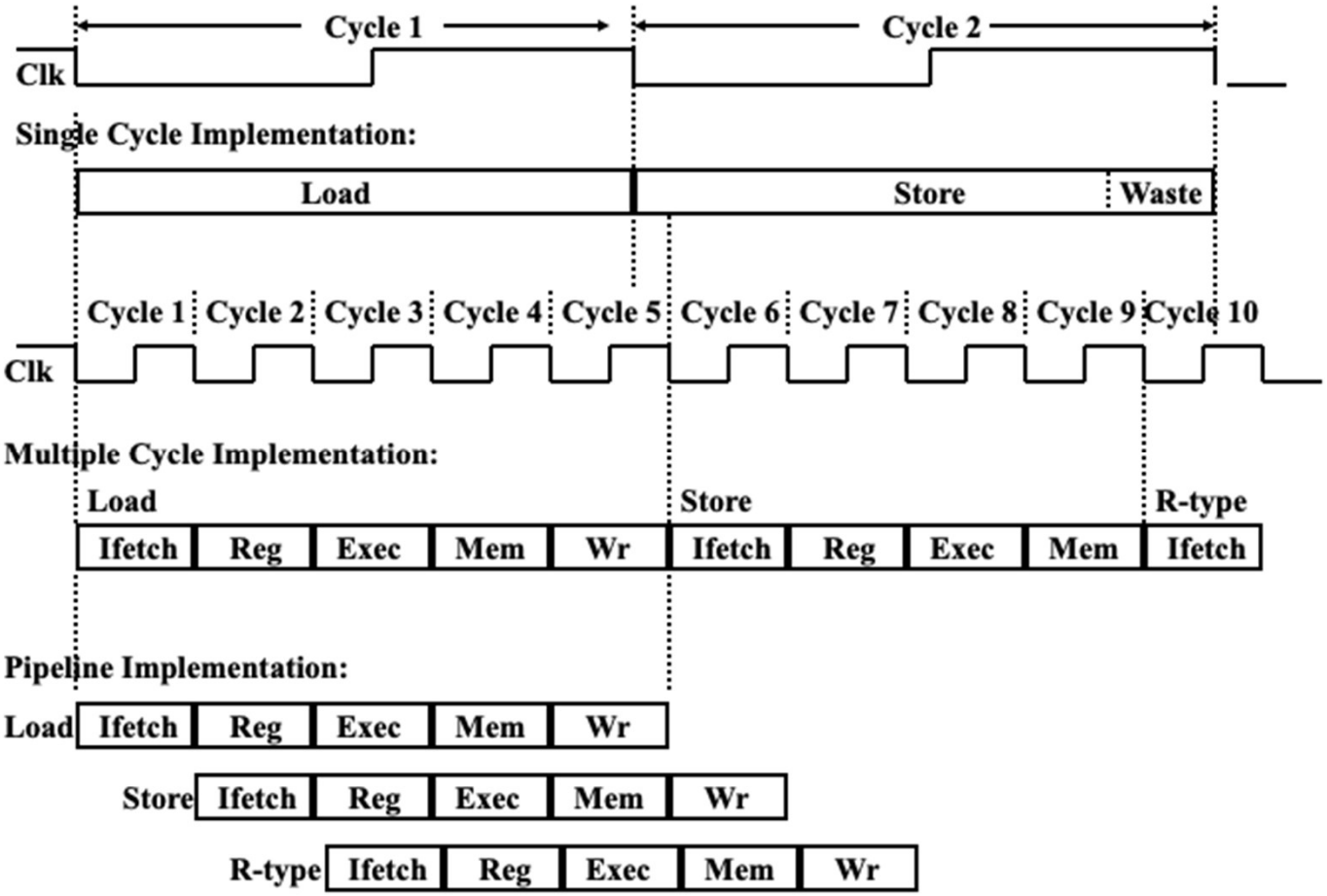
- ★ Ifetch: Instruction Fetch
- ★ Fetch the instruction from the Instruction Memory
- ★ Reg/Dec: Registers Fetch and Instruction Decode
- ★ Exec: Calculate the memory address
- ★ Mem: Read the data from the Data Memory
- ★ Wr: Write the data back to the register file

Representation

- Representation
  - Darker shade on the right-hand side means read.
  - Darker shard on the left-hand side means write.



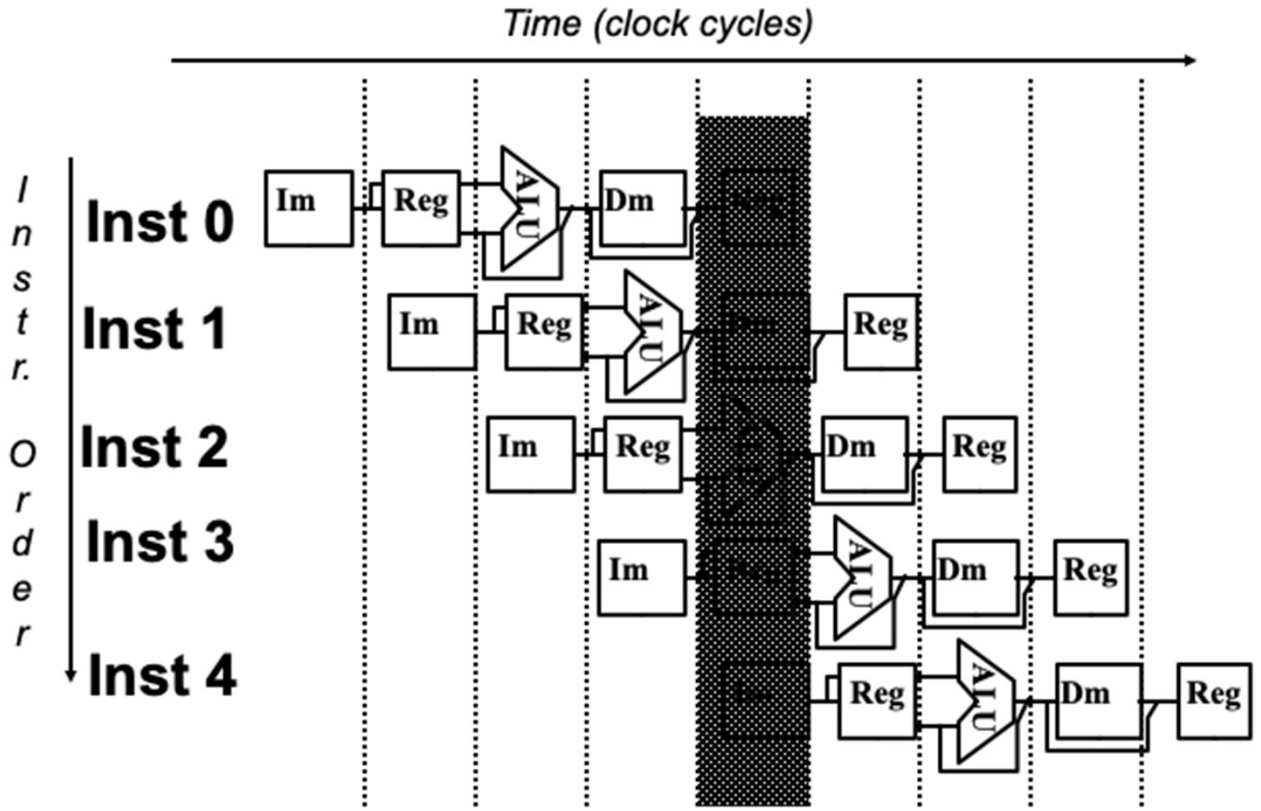
Single Cycle vs. Multiple Cycle vs. Pipeline



Processor Speedup of Processor

- ★ Suppose we execute 100 instructions
- ★ Single Cycle Machine has 45 ns/cycle
- ★ Multicycle Machine has 10 ns/cycle with 4.6 (averaged) CPI
- ★ Ideal pipelined machine has 10 ns/cycle with 4 stage to fill.
- ★ Which one is the fastest? (and Speedup?)

Since resources are there! Why not Pipeline?



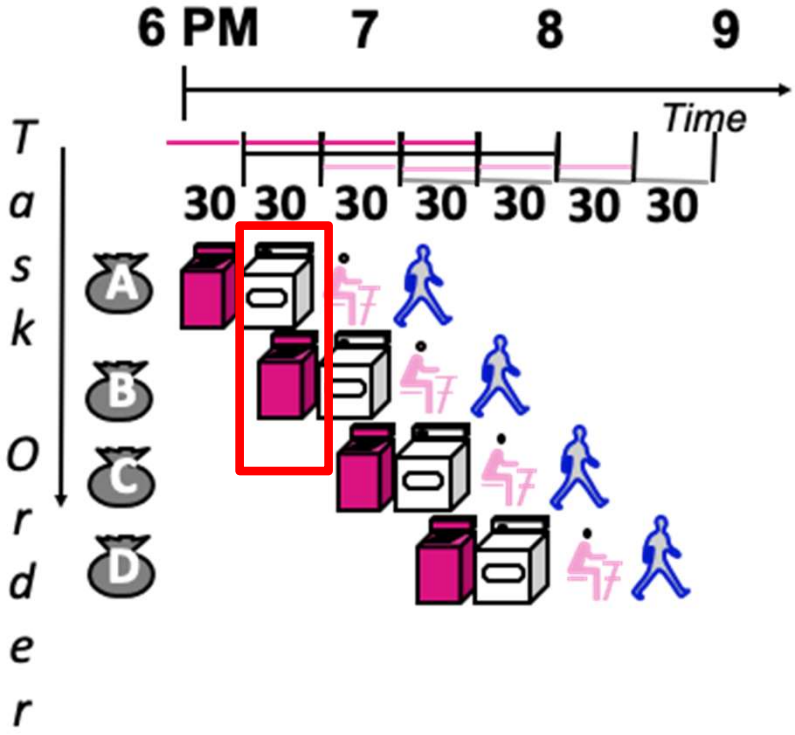
Pipelining is not perfect.

- ★ The world is not perfect. Pipelining can also get us into trouble.
- ★ Pipeline Hazards

- ★ Structural hazards - Limited in resources
- ★ Data hazards - dependency of works/data
- ★ Control hazards - don’t know what to do next

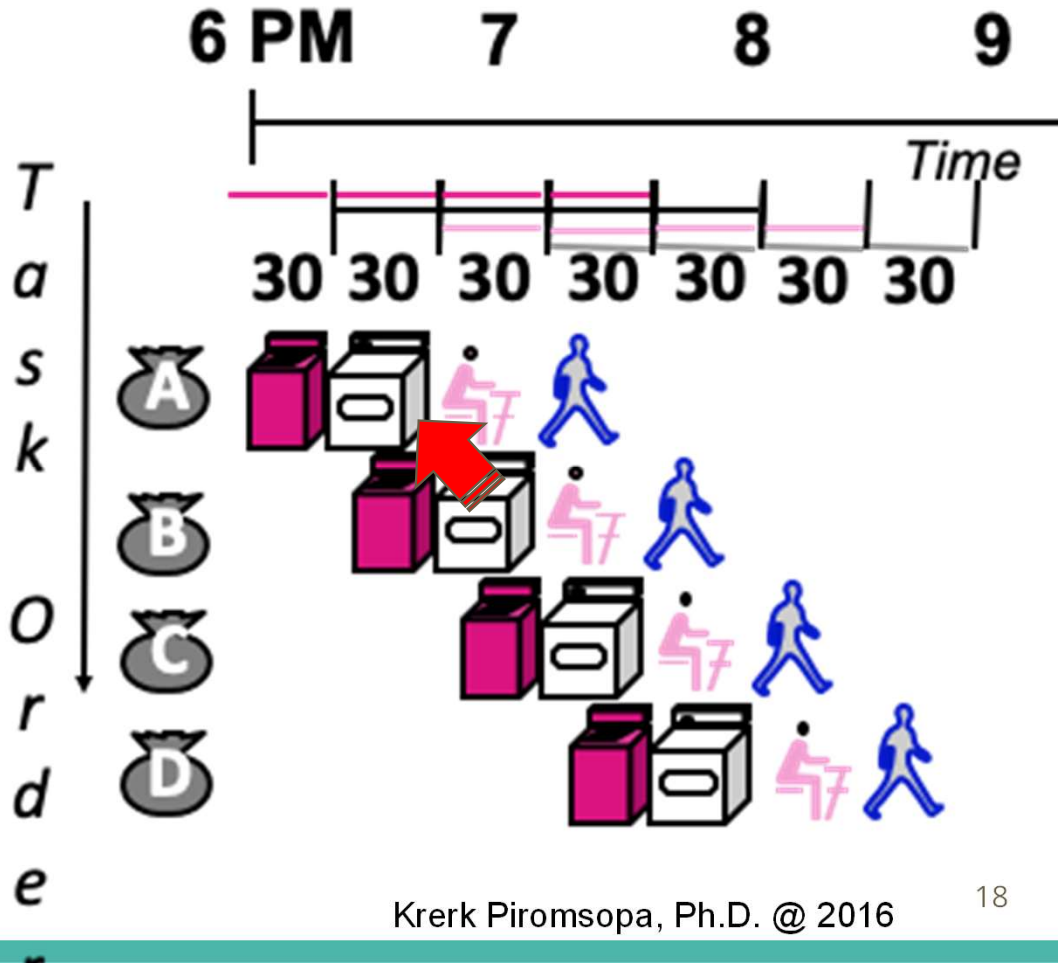
Structural Hazards

- ★ Structural hazards
  - Same resource is required in two (or more) different ways at the same time.
- ★ Example
  - (1) Washer and Dryer is now a same machine. This would be a structural hazard. It is not possible to wash and dry in the same cycle.
  - (2) Folder busy doing something else (e.g. watching TV).



Data Hazard

- ★ Data Hazards
  - Data is required before it is ready.
  - The data is in the future.
- ★ Example
  - (1) A uniform (shirt and pants) must be folded together. If they are washed and dried in different loads, folding is not possible.
  - (2) One sock of pair in dryer and one in washer; can't fold until get sock from washer through dryer



Control Hazards

- Control Hazards
  - When the resource is available, we want to start a new load. However, the next load is not determined yet.
  - Cannot make a decision before the condition is determined.
- Example
  - Washing football uniforms require proper level of detergent. We need to see it after washer/dryer before putting a next load in.
  - branch instructions

How to solve hazards

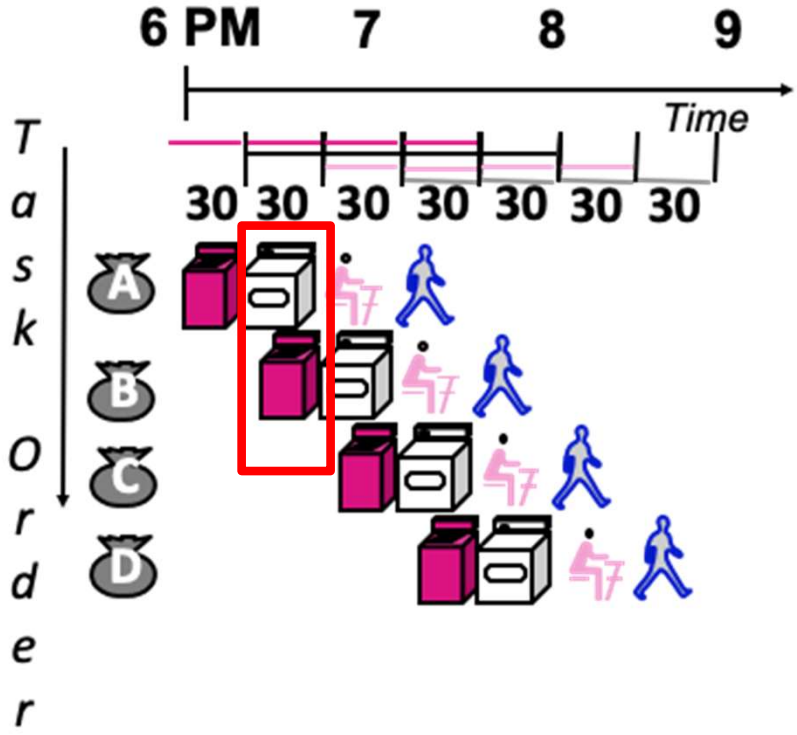
- ★ We can naively solve hazards by **waiting**.
- ★ However, waiting is bad for performance. Maybe there is a workaround solution.

- ★ Pipeline control must detect the hazard
- ★ Take action (or delay action) to resolve hazards

Possible Solutions for Structural Hazards

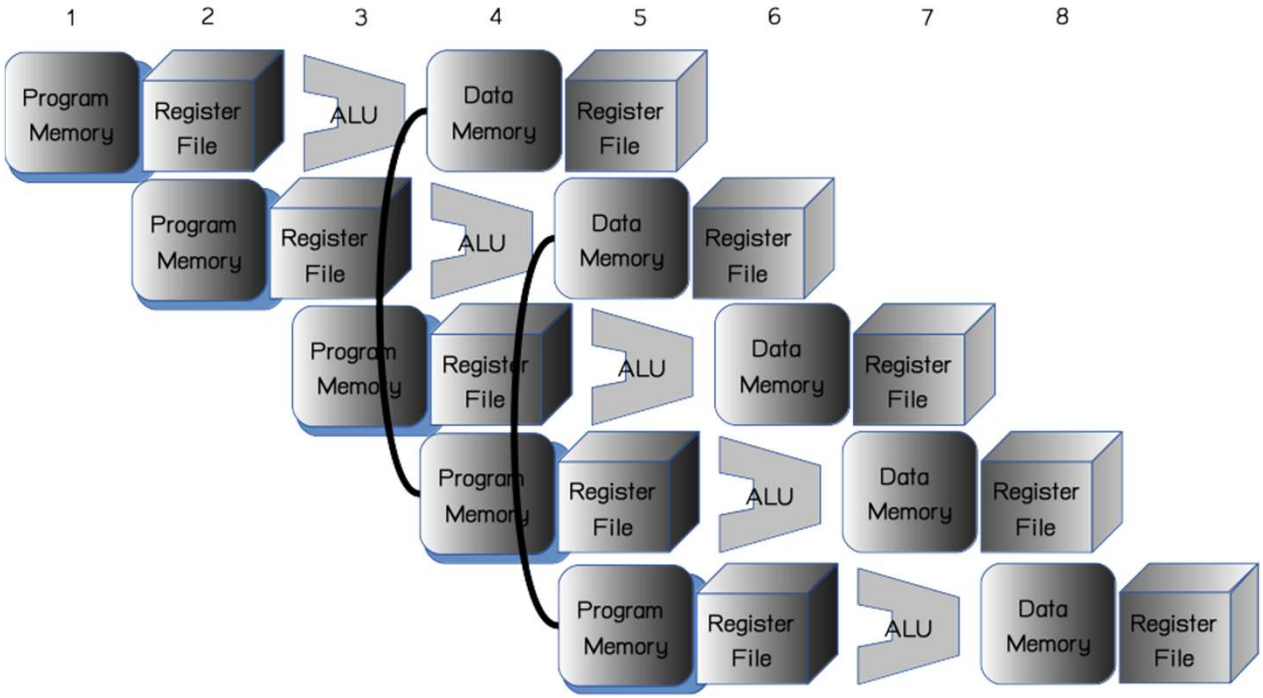
- ★ Limited Resources
- ★ Add more instances (Solve by money?)
- ★ Solve by designs

- ★ For our laundry, buy more machines.



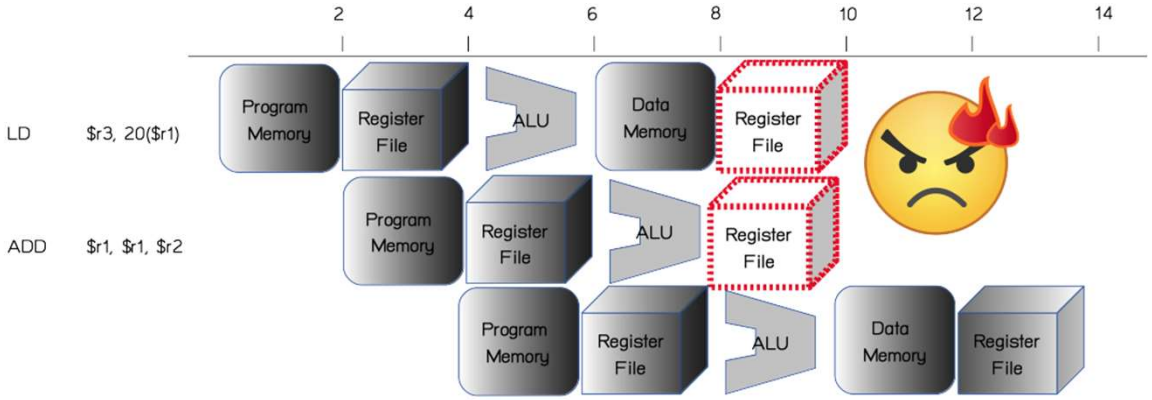
Structural Hazards in Processor (1)

- ★ Single Memory: Data Memory and Program Memory are the same?
- ★ Solution: Make two copies (eg. separate CACHEs). One for program memory, another for data memory.

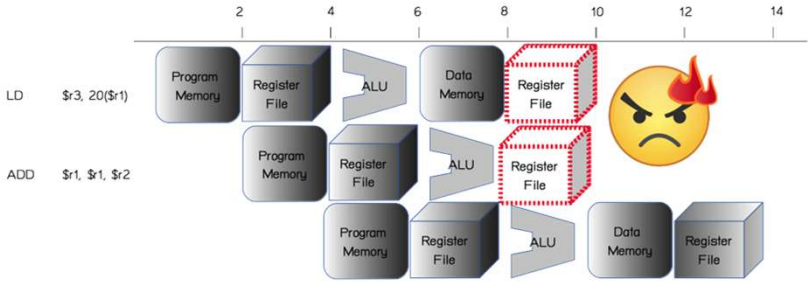


Structural Hazards in Processor (2)

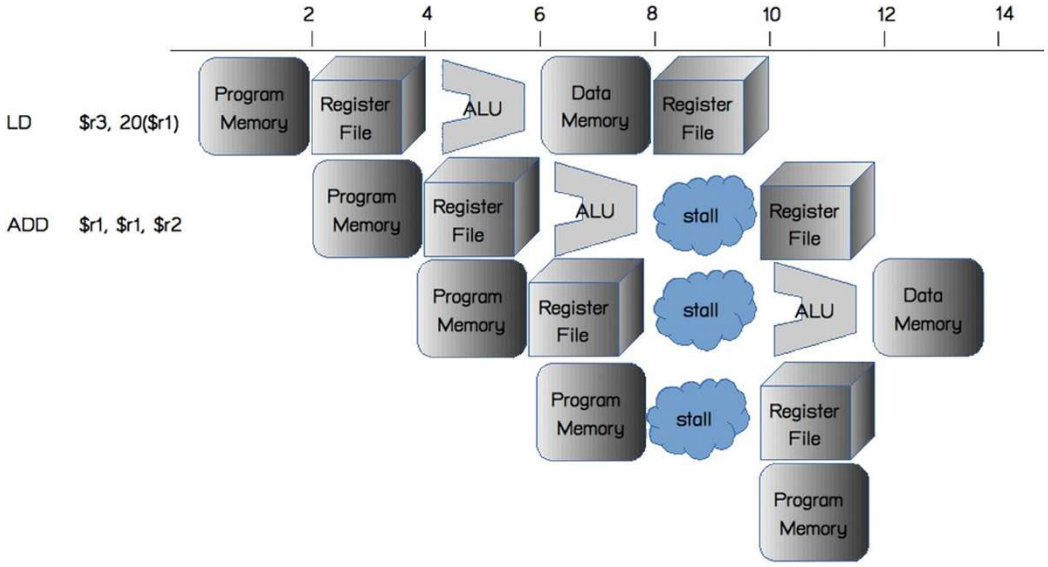
- ★ Mixing 3, 4, 5 cycle instructions together, may lead to using the same resource at the same time.
- ★ LD (5 cycles) follows by ADD (4 cycles) will resulting in WB at the same cycle.
- ★ Solution:
  - (1) stall or
  - (2) redesign to avoid such condition.



Pipeline Stalls

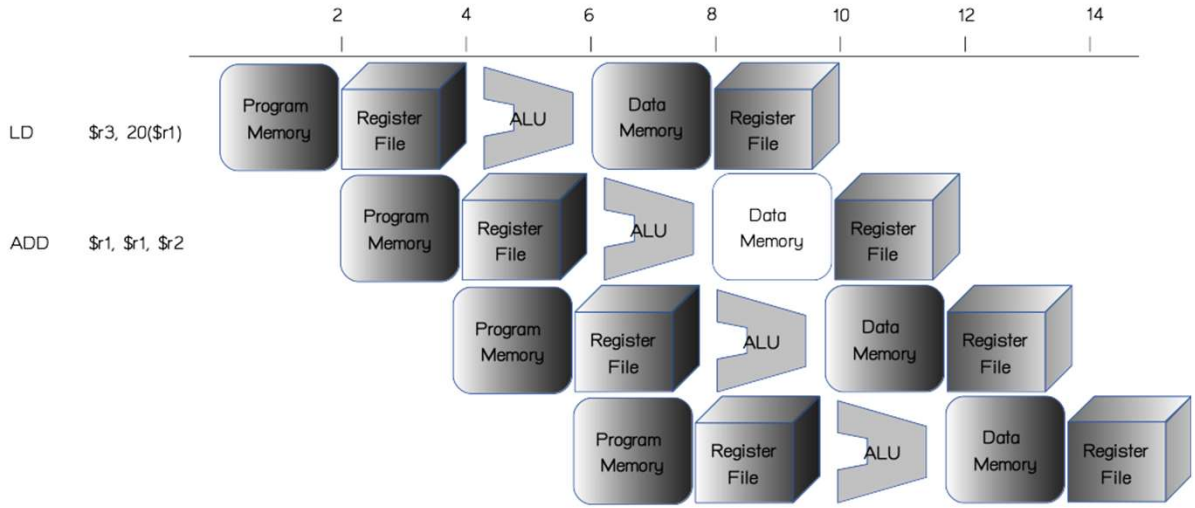


★ Now, pipelining is not ideal.



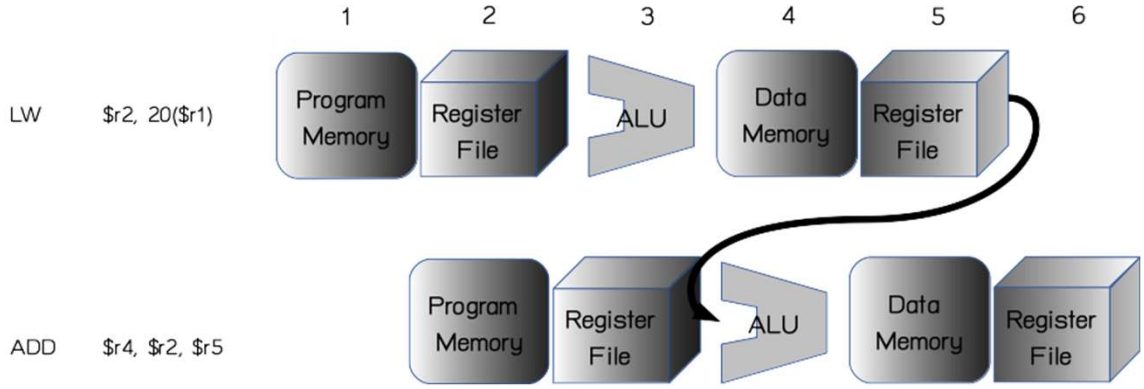
Redesign to avoid stall

- ★ Make R-type to use 5 cycles (add a dummy MEM stage).
- ★ Now, we do not have hazard.
- ★ However, R-type instructions is now 1 cycle slower. (Design tradeoff)



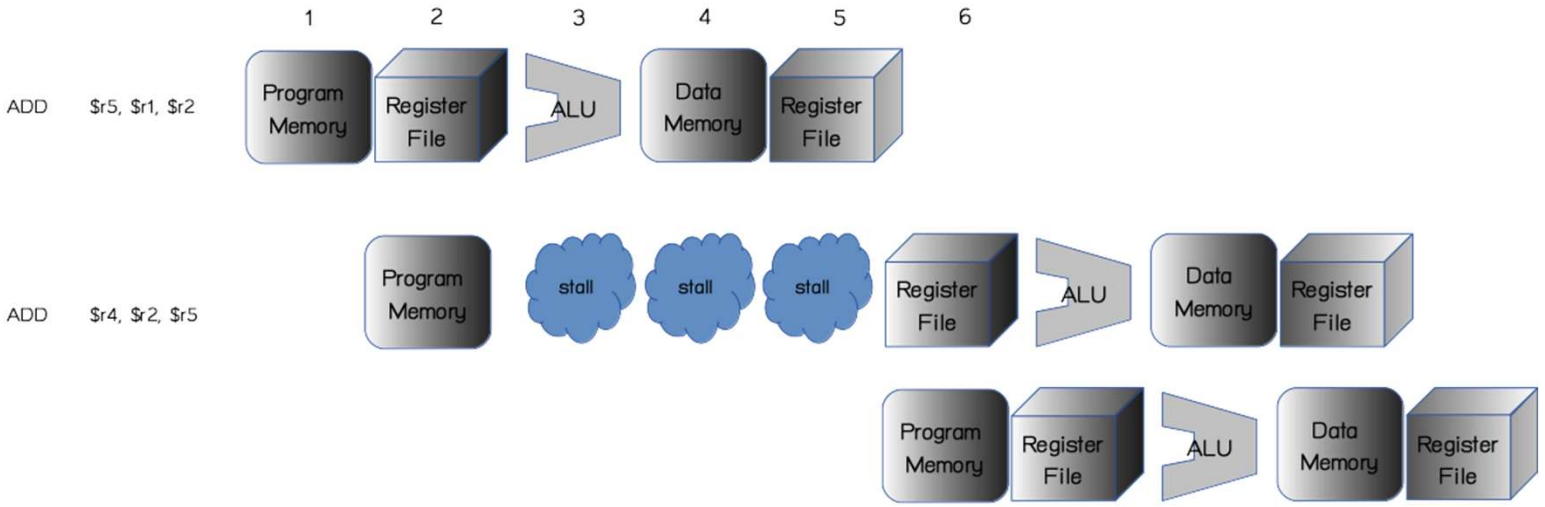
Data Hazards in a Processor

- ★ Reading a data that is a result of previous instruction results in data hazard (read after write).
- ★ ADD $r4, $r2, $r5 after LW $r2, 20($r1). $r2 will be ready after the 5th cycle. However, ADD wants to use $r2 in 4th cycle.



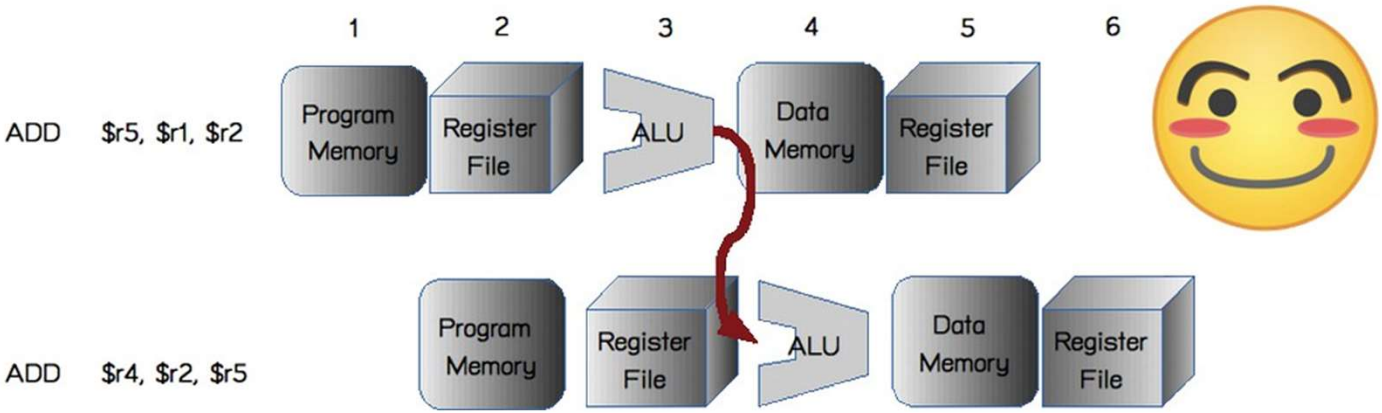
Possible Solutions to Data Hazards (1)

- ★ Wait ?
- ★ Stall ADD for 3 cycles until data is ready.



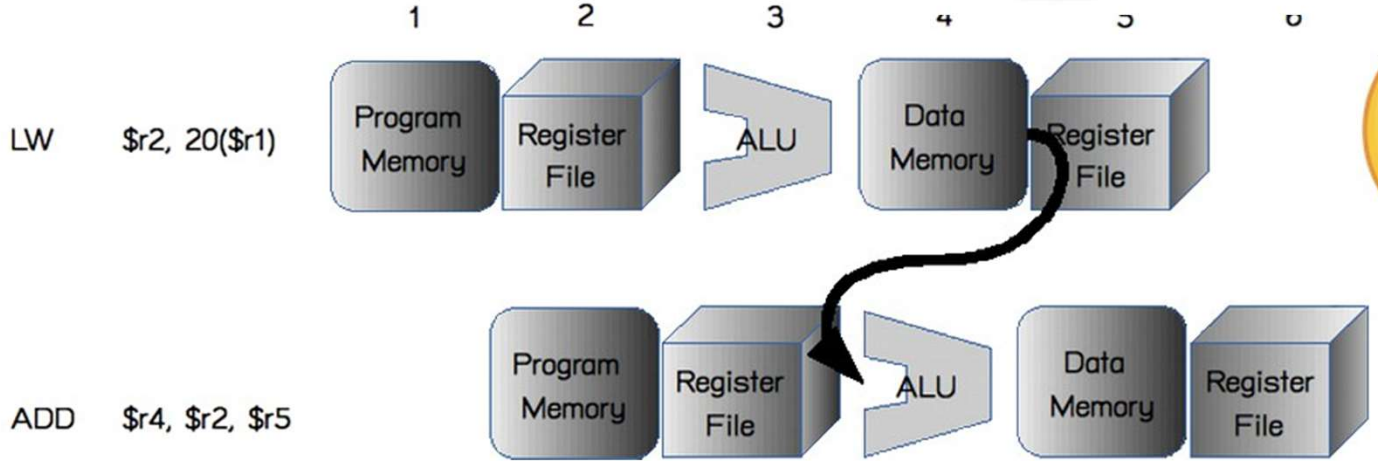
Possible Solutions to Data Hazards (2)

- ★ Hardware Bypass/Forward
- ★ In several cases, Data is there. Rather than waiting, we can just bypass data path.
- ★ Redesign datapath to support forwarding



Possible Solutions to Data Hazards (3)

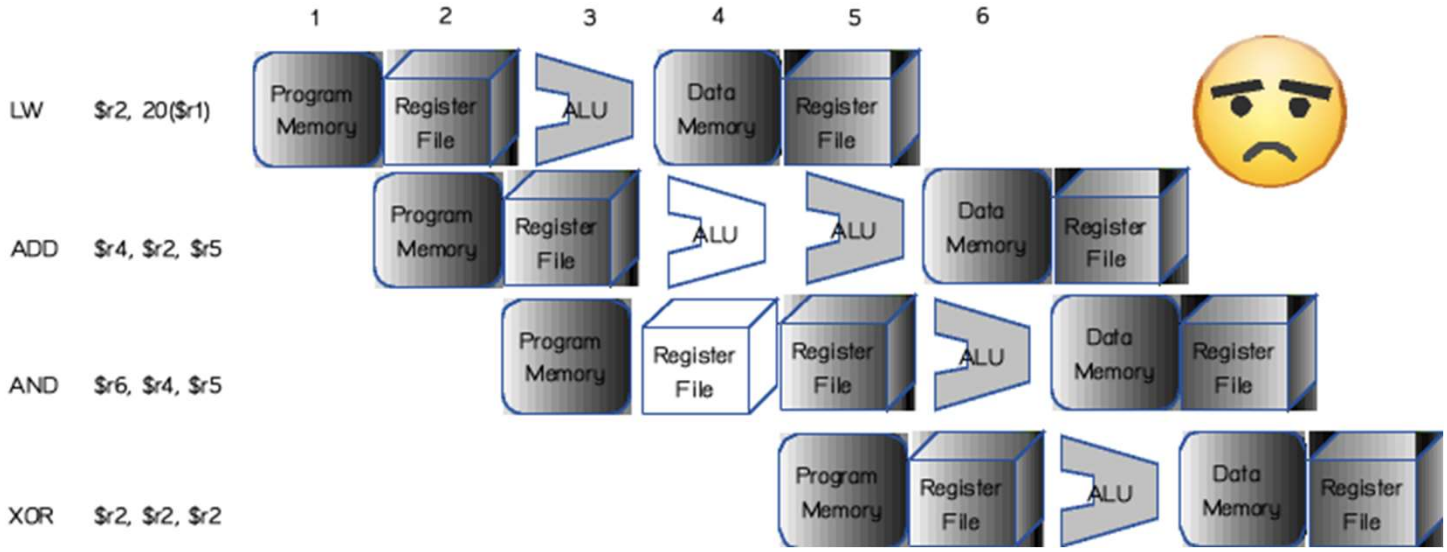
- ★ Forwarding is not always possible. (Unless we have a time machine.)
- ★ Consider ADD after LW.



Pictures from
https://tshop.r10s.jp/hobbyzone/cabinet/02/003/4970381183851-j00.jpg?fitin=330:330
https://www.backtothefuture.store/v/vspfiles/photos/21011-2.jpg

Possible Solutions to Data Hazards (4)

- ★ Ask software for help by rescheduling code to avoid (data) hazards.
- ★ Adding/Moving instruction in between. (Delay Load)
- ★ If none is applicable, insert nop (no operation)



Possible Solutions to Data Hazard (5)

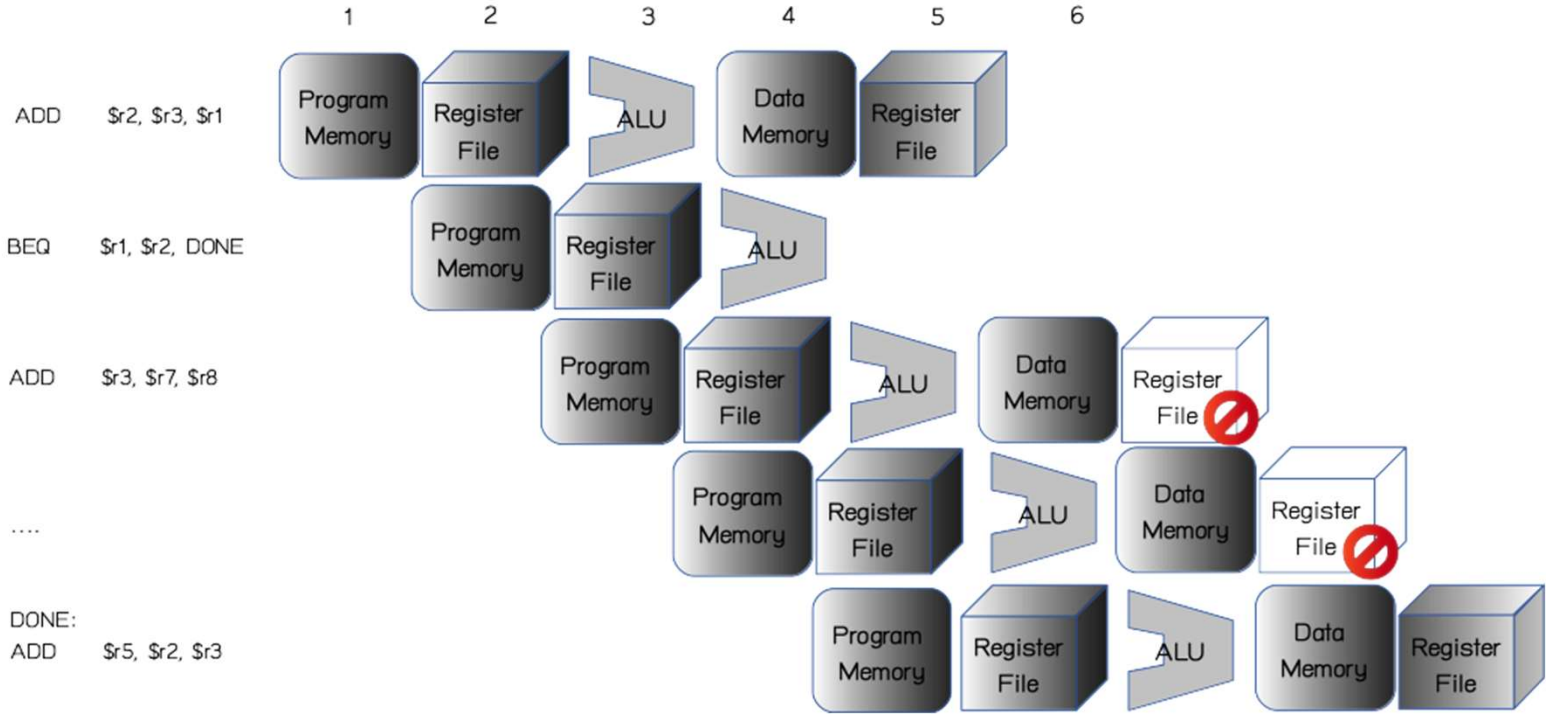
- ★ Drawing a diagram for the two code.
- ★ See how software can play a role in avoiding hazards.
- ★ We will revisit data hazards later.

```
LW      $r2,20($r1)
AND     $r4,$r2,$r5
```

```
LW      $r2,20($r1)
NOP
NOP
NOP
AND     $r4,$r2,$r5
```

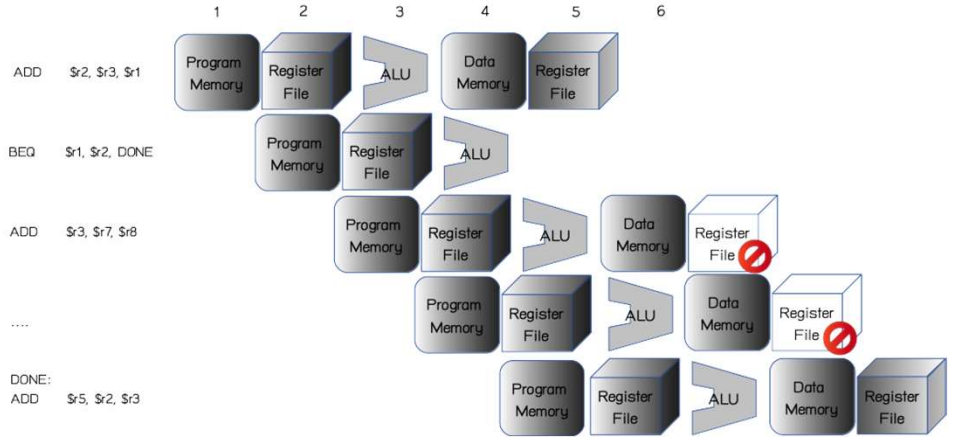
Control Hazards in a Processor

- ★ Control Hazard (aka. Branch Hazard)
- ★ Branch would resolve in 3 cycles. So, what should we do in the next cycle?



Possible Solutions to Control Hazards (1)

- ★ Stall?
  - 2 cycles slower
- ★ Guess (Branch Prediction)
  - Predict taken or not taken
  - If correct, no delay
  - If incorrect, 1 cycle delay?
  - At least 50% changes of being correct
  - More advance scheme 90% correct (e.g. Branch history)



Possible Solutions to Control Hazards (1)

- ★ Redefine branch with branch delay slot
- ★ A slot after is not related to branch. Branch is 1 cycle later.
- ★ Software/Compiler can take benefit of the extra slot.

Without delay slot

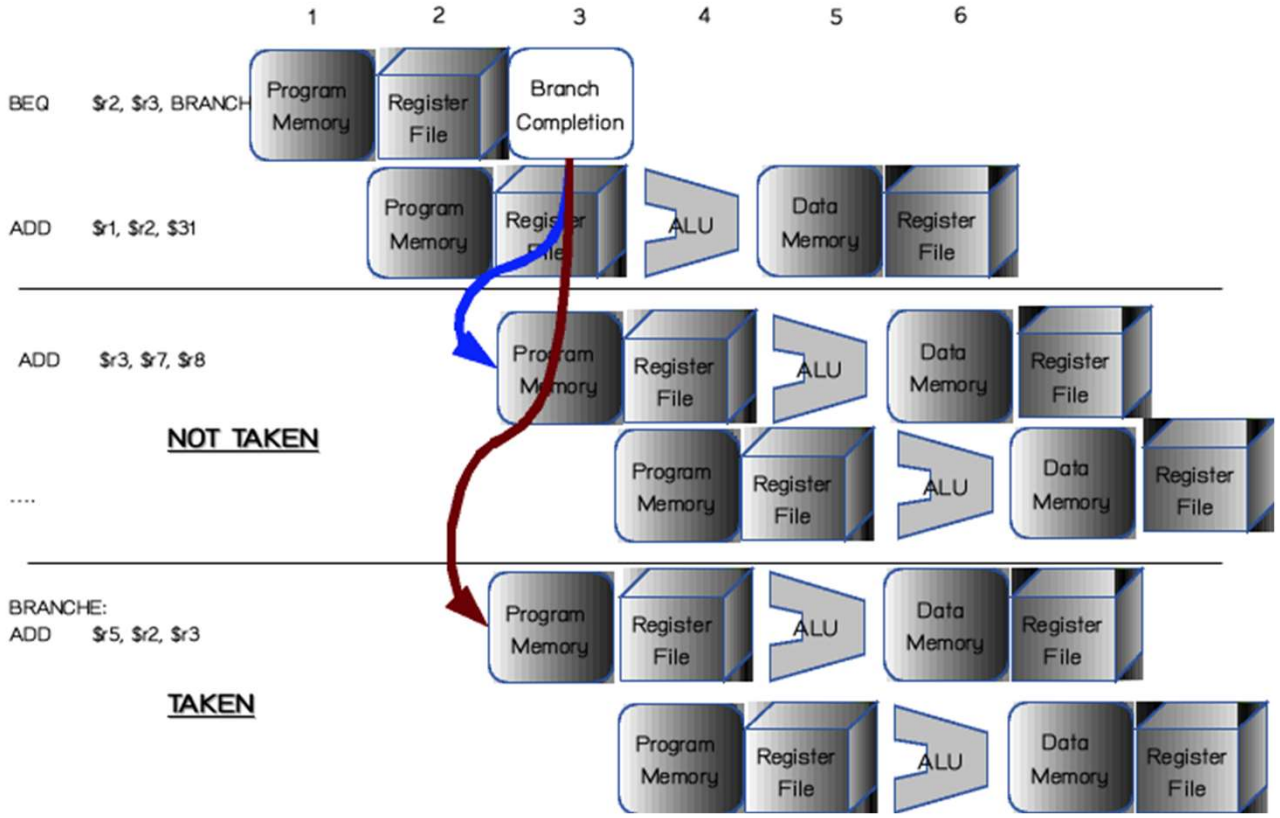
```
(1)       ADD     $r1, $r2, $r3
(2)       BEQ     $r2, $r3, BRANCH
(3)       ORI     $r4, $r5, #100
BRANCH:
(9)       SW      $r1, 100($r3)
```

With delay slot

```
(2)       BEQ     $r2, $r3, BRANCH
(1)       ADD     $r1, $r2, $r3
(3)       ORI     $r4, $r5, #100
BRANCH:
(9)       SW      $r1, 100($r3)
```

Branch Delay Slot

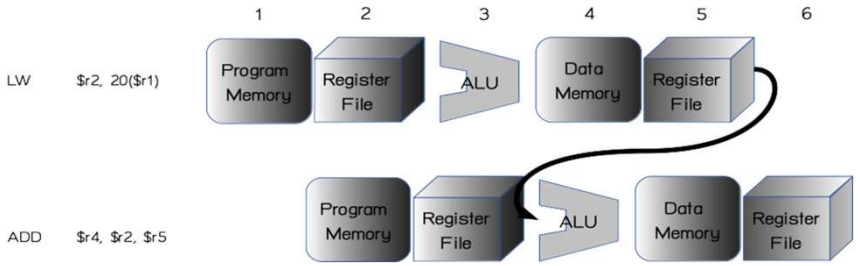
★ Need support from software.

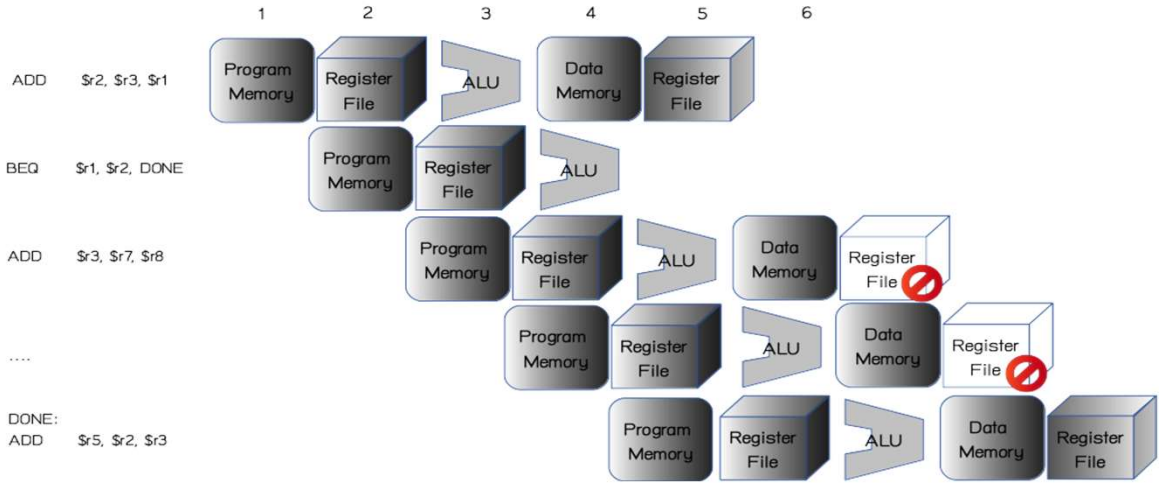


Exercises

Mixing CPI and Structural Hazards

★ Why mixing SW, BEQ, and JMP with R-Type (ADD, ORI, ORUI) does not result in a structural hazard? Please provide your analysis.





Pipelining Representation

- ★ Please draw a diagram showing the pipelining of the following code. (Please also provide correct shade for reading and writing).
  - ADD $r10,$r0,$r1
  - LW $r12,20($r5)
  - ADD $r3,$r12,$r12
  - SW $r3,30($r10)
  - ADD $r10,$r10,$r1
- ★ There are some hazards in the given pipeline. Can you identify some of them?

End of Chapter 6 (part 1)