CSCI 322
Principles of Concurrent Programming

Filip Jagodzinski
Preliminaries: Bookkeeping

- Labs will be held *most* Thursdays, at the same time as lecture is scheduled
  - CF416 and CF418
  - You do not need to register explicitly for a lab section
http://facultyweb.cs.wwu.edu/~jagodzf/teaching/csci322/

- Two exams
  - Take home midterm
  - In-class final
- Sample midterm and final exams are available on the course website
- Two homework assignments, submitted via Canvas. A mix of on-paper questions and programming tasks
Who has read the course description?
Who has read the course description?

What is this class about?
Why do we care?
What will you learn?
What will you not learn?
Introduction

From the course catalog: Algorithms for mutual exclusion. Synchronization and communication techniques: semaphores; monitors; rendezvous; conditional critical regions. Multi-process and multi-threaded programming. Concurrent programming facilities in HLL's.

Task: Define each of the words in the course catalog description of this course.
Introduction

From the course catalog: Algorithms for mutual exclusion. Synchronization and communication techniques: semaphores; monitors; rendezvous; conditional critical regions. Multi-process and multi-threaded programming. Concurrent programming facilities in HLL's.

An important component of learning is knowing WHY we do things the way we do them.

Why do we need synchronization?
What are semaphores?
What is the difference between parallel and concurrent programming?

If you aren’t asking “why” then you are just memorizing
Motivation: Architecture 101 review

Q: What are the main features of a computer, and how do they work together to transfer information (data) back and forth among different hardware components?

(there’s a reason that architecture is a prerequisite for this course)
Q: What are the main features of a computer, and how do they work together to transfer information (data) back and forth among different hardware components?

Define and explain the Following

- Register
- ALU
- Control Unit
- Opcode
- Cache L1
- Cache L2
- Memory
- Hard Drive
- CPU
Motivation: Architecture 101 review

Q: What are the main features of a computer, and how do they work together to transfer information (data) back and forth among different hardware components?

Task: draw a computer component hierarchy (hard drive, memory, L2 cache, L1 cache, registers, and ALU) which includes an axis for speed of data transfer and data capacity (on the board exercise)
Motivation: Architecture 101 review

Q: What does all of this have to do with concurrent programming?
Motivation: Architecture 101 review

Q: What does all of this have to do with concurrent programming?

Q: How many of you run a SINGLE program each time that you use a computer?

Demo: the top command
Motivation: Architecture 101 review

Q: What does all of this have to do with concurrent programming?

Q: How many of you run a SINGLE program each time that you use a computer?

Q: What are possible issues that might arise, and what are a few possible solutions?

(on the board exercise)
Q: What does all of this have to do with concurrent programming?

Q: How many of you run a SINGLE program each time that you use a computer?

Q: What are possible issues that might arise, and what are a few possible solutions?

You might have said: just build faster and faster computers, which means placing more transistors on a chip ... why is this not feasible?
Q: Assume that the number of transistors on a die keeps on going up ... is memory (access) speed keeping up?
Motivation: Architecture 101 review

Q: Can the number of transistors on a die continue to increase? Why or why not?
Motivation: Architecture 101 review

Q: Can the number of transistors on a die continue to increase? Why or why not?
Hence we are “stuck” ... but we want to run more applications at the same time ... Going back to our architecture hierarchy ...

Q: If we want to run more and more programs (applications) on a single-chip computer, but the number of registers and ALUs asymptotes, what else can we do?

(brainstorm, on-the-board exercise)
Hence we are “stuck” ... but we want to run more applications at the same time ... Going back to our architecture hierarchy ...

Q: If we want to run more and more programs (applications) on a single-chip computer, but the number of registers and ALUs asymptotes, what else can we do?

Structure of this course

Labs: some are “simple,” others are programming assignments in disguise.
Structure of this course

Homework Assignments: a mixture of “book” questions and programming exercises

Many of the lab tasks and homework assignments are open-ended.

My expectation of you: ask questions, be thorough, and start right away
### Structure of this course

#### Midterm and final exams

**Sample Midterm 2013 - year 2009**

- This exam is a closed book, closed notes, closed laptops and smartphones, etc. exam.
- All that you can use is either a pen or pencil.
- The last few pages provide formulae, which you may be helpful.
- Do not spend too much time on any one question.

**Name (Print)__________________________**

**House Code statement:** Pledge that this submission is solely my work. I pledge that I have not provided help to anyone. I pledge that I have not received help from anyone.

<table>
<thead>
<tr>
<th>Section</th>
<th>Question Type</th>
<th>Question Number($)</th>
<th>Points Possible</th>
<th>Points Earned</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>True/False</td>
<td>1-5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>II</td>
<td>Multiple Choice</td>
<td>6, 7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>III</td>
<td>Semaphores</td>
<td>8</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>IV</td>
<td>Logic</td>
<td>9, 10</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>Short Answer</td>
<td>1, 15</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>VI</td>
<td>Multiple Answers</td>
<td>16</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>VII</td>
<td>Extra Credit</td>
<td>17, 18</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

**Total Exam Score**

- **I. True/False:** Instructions: Circle either True or False. No partial credit. 3 points each.
  1. True/False: A data hazard arises when there are not enough functional units of the same step of a pipeline to permit multi-threading.
  2. True/False: When a semaphore is created, it must be initialized to 0 (zero).
  3. True/False: Every loop in a program should be parallelized if there are idle CPUs waiting.
  4. True/False: In the context of concurrent programs, partial correctness is achieved when every loop and procedure call terminates.
  5. True/False: Two processes are independent when the write set of each is disjoint from the read and write sets of the other.

**III. Semaphores:** 14 points. Partial credit given.

- The pseudocode below contains three functions, calcA() and calcB(), each with 2 instructions, and main(). All instructions inside calcA and calcB are atomic. Assume Semaphore class is available, which has a constructor Semaphore(int initialValue), and functions increment() and decrement(). Variables x and y are global variables (i.e., shared among all functions) saved in shared memory. Declare semaphores wherever needed, so that when the main method runs to completion, the value of x and y are 7 and 12.

  ```
  V. Short Answer. Provide a concise answer to each question. Partial credit. 6 points each.
  12. Assume a 5 step pipeline with the following stage latencies
      Stage 1: 5ns
      Stage 2: 6ns
      Stage 3: 8ns
      Stage 4: 3ns
      Stage 5: 4ns
  
      Under these conditions, what will be the pipeline’s throughput when the 12,435\(^7\) of 20,000
      instructions is executing, and what will be the pipeline’s latency, also for instruction 12,435\(^7\)?
      Do not assume any stalls other than those that might be imposed by the stage latencies listed
      above.
      
      Pipeline throughput: ________________________
      Pipeline latency: ________________________
      
      13. What cache reuse ratio is needed so that a program that uses cache is 2 times better
          (performance gain) than a program that does not use cache? Assume cache access time of
          1ms and memory access time of 24ns.
      Your Answer: ________________________
Why should you take this course ...

A. Because it’s a required course
B. Because I needed to take an extra class to get above 12 credits to keep my financial aid
C. Because it has real world applications
Why should you take this course ...

A. Because it’s a required course
B. Because I needed to take an extra class to get above 12 credits to keep my financial aid
C. Because it has real world applications
Review of Architecture
Because devices such as the hard drive and main memory have large capacities compared to the registers and ALUs, the constant movement of information among the different computer components experiences bottlenecks.

Q: HOW is information moved back and forth among components?

Q: What is the sequence of steps that are taken to “execute” a single program instruction?
Because devices such as the hard drive and main memory have large capacities compared to the registers and ALUs, the constant movement of information among the different computer components experiences bottlenecks.
Because devices such as the hard drive and main memory have large capacities compared to the registers and ALUs, the constant movement of information among the different computer components experiences bottlenecks.
Let us begin “small.”

Q: How might we “improve” the speed of a computer using the most naïve approach?

Setup: There are three instructions that we want to execute as part of a program ... how are they executed (concurrently?) so that their sum execution time is minimized
Q: What is a computer pipeline?
The computer pipeline

Task: Explain the acronyms IF, ID, EX, and WB
The computer pipeline

Instruction Fetch
The instruction is fetched from physical memory

Instruction Decode
The instruction is decoded (for example if the instruction is to add two numbers, then fetch the values of R1 and R2 (registers) into the appropriate ALU)

Execute
Perform the calculation using the circuitry in the ALU

Write Back
Result of operation performed by ALU is written back to the register(s)
The computer pipeline

Instruction Fetch
The instruction is fetched from physical memory

Instruction Decode
The instruction is decoded (for example if the instruction is to add two numbers, then fetch the values of R1 and R2 (registers) into the appropriate ALU)

Execute
Perform the calculation using the circuitry in the ALU

Write Back
Result of operation performed by ALU is written back to the register(s)

The instruction pipeline (orange, yellow, grey, green) may need to “fetch” information from L1, L2, main memory or the hard drive, so each instruction might require different amounts of time to complete
The computer pipeline

**Goal:** calculate the “speed” or overall execution “time” for processing fully an instruction using the above pipeline

Assume that each step of the pipeline takes a single clock cycle to complete

**Q:** How many clock cycles does it take to complete one instruction?
The computer pipeline

Instruction “queue”

Completed instructions

\[ I_1 \]

Clock cycle  1

**Goal**: calculate the “speed” or overall execution “time” for processing fully an instruction using the above pipeline

Assume that each step of the pipeline takes a single clock cycle to complete

**Q**: How many clock cycles does it take to complete one instruction?
The computer pipeline

Instruction “queue”

Completed instructions

\[ I_1 \]

Clock cycle 2

Goal: calculate the “speed” or overall execution “time” for processing fully an instruction using the above pipeline

Assume that each step of the pipeline takes a single clock cycle to complete

Q: How many clock cycles does it take to complete one instruction?
The computer pipeline

**Instruction “queue”**

- **IF**
- **ID**
- **EX**
- **WB**

**Completed instructions**

$I_1$

**Clock cycle** 3

**Goal:** calculate the “speed” or overall execution “time” for processing fully an instruction using the above pipeline

Assume that each step of the pipeline takes a single clock cycle to complete

**Q:** How many clock cycles does it take to complete one instruction?
The computer pipeline

Instruction “queue”               Completed instructions

IF  ID  EX  WB

\[ I_1 \]

Clock cycle  4

**Goal:** calculate the “speed” or overall execution “time” for processing fully an instruction using the above pipeline

Assume that each step of the pipeline takes a single clock cycle to complete

**Q:** How many clock cycles does it take to complete one instruction?
The computer pipeline

Instruction “queue”  
Completed instructions

Clock cycle 4

**Goal**: calculate the “speed” or overall execution “time” for processing fully an instruction using the above pipeline

Assume that each step of the pipeline takes a single clock cycle to complete

**Q**: How many clock cycles does it take to complete one instruction?

We say that the pipeline **latency** of $I_1$ is 4 clock cycles
Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”

Completed instructions

IF ID EX WB

I₂ I₁

If the IF portion of the pipeline requires a single hardware component of the CPU, can more than 1 instructions be in the IF stage of the pipeline?

Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”

Completed instructions

IF  ID  EX  WB

I₂  I₁

Clock cycle  1

Q: How many clock cycles does it take to complete two instructions?
Q: How many clock cycles does it take to complete two instructions?
Q: How many clock cycles does it take to complete two instructions?
Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”

Completed instructions

I_2

I_1

Clock cycle 5

Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”

Completed instructions

IF  ID  EX  WB

$I_2$  $I_1$

Clock cycle  6

Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”  Completed instructions

IF  ID  EX  WB

I₂  I₁

Clock cycle  7

Q: How many clock cycles does it take to complete two instructions?
Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”

Completed instructions

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>WB</th>
</tr>
</thead>
</table>

\[
I_2 \quad I_1
\]

Clock cycle \ 8

This is what a non-pipelined architecture machine does

Q: Can we do better? How? Why?
The computer pipeline

Instruction “queue”

\[ I_2 \quad I_1 \]

Completed instructions

Clock cycle

Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”   Completed instructions

| IF | ID | EX | WB |

\[ I_2 \quad I_1 \]

Clock cycle  1

Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”

Completed instructions

IF ID EX WB

$I_2$ $I_1$

Clock cycle 2

Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”

Completed instructions

IF   ID   EX   WB

\( l_2 \quad l_1 \)

Clock cycle  3

Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”  

Completed instructions

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>WB</th>
</tr>
</thead>
</table>

I_2  I_1

Clock cycle  4

Q: How many clock cycles does it take to complete two instructions?
Q: How many clock cycles does it take to complete two instructions?
The computer pipeline

Instruction “queue”  
Completed instructions

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>EX</th>
<th>WB</th>
</tr>
</thead>
</table>

I₂  I₁

Clock cycle  5

This is what a pipelined architecture machine does

Q: How many clock cycles does it take to complete two instructions?
Another pictorial representation to show how the instructions are pipelined through the 4-step pipeline

We say that the **throughput** of this pipeline is 1 instruction every clock tick (imagine sitting at the “exit” of this pipeline and observing the instructions being completed)
The computer pipeline

In the “real” world, however ...

Q: What simplifications have we assumed, and what are the ramifications?
The computer pipeline

Each step of the pipeline does not take the same amount of time.

Q: What is the pipeline latency* for processing 1 instruction?

Q: What is the pipeline latency* for processing 2 instructions?

Q: What is the pipeline latency* for processing 3 instructions?

* Here “pipeline latency” means “total time that the pipeline is busy working”
The computer pipeline

6ns  3ns  5ns  8ns

Q: How many clock cycles does a second (queued) instruction require?

I instruction: $6 + 3 + 5 + 8 = 22$ ns
Remember that each step of the pipeline can be executing a single instruction only, so instruction 2 must “wait” for the IF stage until instruction 1 is done utilizing the IF stage.

2 instructions: $22 + 6 = 28\text{ns}$

Is this correct? Why or why not?
Instruction 1 and 2 are attempting to use the WB stage of the pipeline at the same time. This is not allowed.

**How do we fix this?**
The computer pipeline

2 instructions: $20 + 2 + 8 = 30\text{ns}$

Instruction 2 is delayed by instruction 1

Q: What is the pipeline latency for processing 3 instructions?
Instruction 3 is delayed by instruction 2, which was delayed by instruction 1

Q: Is the maximum number of stalls that an instruction experiences always 1?
The computer pipeline

For the 4-stage pipeline shown above, and for 3 instructions queued ...

- What is the pipeline latency for instruction 1?
- What is the pipeline latency for instruction 2?
- What is the pipeline latency for instruction 3?
- How many stalls (ns) does instruction 1 experience?
- How many stalls (ns) does instruction 2 experience?
- How many stalls (ns) does instruction 3 experience?
- What is the pipeline throughput for the 3 instructions?

Assume that there is only one of each architecture component capable of performing the IF, ID, EX and WB stages

(In-class exercise, on-the-board calculations)
Pipeline Calculations, Speedup, Latency

How does this analysis scale to programs with thousands of instructions?
Pipeline Calculations, Speedup, Latency

Q: An instruction is completed every _____ ns
Q: A single instruction takes ______ ns to complete
Q: An instruction is completed every _____ ns
Q: A single instruction takes ______ ns to complete

These are “it depends” questions. Because, the answer depends on how many instructions are being processed.

For a single instructions, the answers to these questions are straight-forward.

Q: What are the answers to these 2 questions for a single instruction?
Q: An instruction is completed every _____ ns
Q: A single instruction takes ______ ns to complete

These are “it depends” questions. Because, the answer depends on how many instructions are being processed.

For a single instructions, the answers to these questions are straight-forward.

Q: If there are 20,000 instructions to be completed, then what is the throughput and what is the latency of our 4-stage pipeline. Assume that as soon as a stage is “free”, then the next instruction “in line” is immediately piped into that stage. Also assume that all instructions are independent of the others.
**Pipeline Calculations, Speedup, Latency**

Throughput: The pipeline is governed by the slowest step, which in this case is the WB, so the throughput is 1 instruction / 8ns.
Pipeline Calculations, Speedup, Latency

Q: An instruction is completed every _____ ns

Q: A single instruction takes ______ ns to complete

Latency: 6+3+5+8=22ns. Is that correct for 20,000 instructions? Why or why not?
Q: An instruction is completed every _____ ns

Q: A single instruction takes _____ ns to complete

Latency: 6+3+5+8=22ns. Is that correct for 20,000 instructions? Why or why not?

NO! We’ve already seen that stalls needs to be introduced. So what is the latency?
Pipeline Calculations, Speedup, Latency

Latency: $6+3+5+8=22\text{ns}$ more than 22ns

After a large number of instructions have been processed, the stalls will have back-logged, and delays will begin to accumulate.

Thus .... throughput AND latency are governed by the slowest pipeline stage.
Q: How much faster is the pipelined machine architecture versus a non-pipelined machine architecture (assume a large number of instructions)?

Remember that a pipelined machine architecture means that instructions are processed and interwoven, while a pipeline refers to the stages that are involved.
Pipeline Calculations, Speedup, Latency

Q: How much faster is the pipelined machine architecture versus a non-pipelined machine architecture (assume a large number of instructions)?

Remember that a pipelined machine architecture means that instructions are processed and interwoven, while a pipeline refers to the stages that are involved.

For the first homework be able to compute:

Execution time for pipeline architecture:
Execution time for non-pipelined architecture:
Speedup:

For 20,000 instruction

On the board discussion
**Q: How much faster is the pipelined machine architecture versus a non-pipelined machine architecture (assume a large number of instructions)?**

Remember that a pipelined machine architecture means that instructions are processed and interwoven, while a pipeline refers to the stages that are involved.

For the first homework be able to compute:

- Execution time for pipeline architecture: \( 20,000 \times 8\text{ns} = 160,000\text{ns} \)
- Execution time for non-pipelined architecture: \( 20,000 \times 22\text{ns} = 440,000\text{ns} \)
- Speedup: \( \frac{440,000}{160,000} = 2.75 \)

This is an approximation, because at time zero, when the pipeline is not yet fully loaded, output is not 1 instruction every 8 ns. For large numbers of instructions, this approximation approaches very closely the “real” execution time.
Pipeline Calculations, Speedup, Latency

Now that we know that the latency and throughput are both governed by the slowest pipeline step, how might a computer architect modify the pipeline stages to improve execution time when large numbers of instructions are being processed?
Now that we know that the latency and throughput are both governed by the slowest pipeline step, how might a computer architect modify the pipeline stages to improve execution time when large numbers of instructions are being processed?

Q: What effect does this have on pipeline throughput? And latency?
Pipeline Calculations, Speedup, Latency

Compare and contrast these two pipelines by answering:

**Q:** For an initially empty pipeline, what is the pipeline latency and throughput of a single instruction?

**Q:** For an initially empty pipeline, what is the pipeline latency and throughput for the 50,000\(^{th}\) instruction (assume instructions are independent and are queued at time 0, and the architecture is a pipelined architecture)
I’ve mentioned that complications arise, but I’ve only discussed one of them. What’s the other one?

Complication 1: Structural hazards (governed by slowest stage)
Complication 2: ________________________________
Data Hazards ...

From the “code” perspective

Instruction 1: \( a = b + c \)
Instruction 2: \( d = a + e \)

Q: How quickly can these two instructions be processed by a pipeline?

This seems like innocent-enough code. **Q: But what could go wrong?** Remember to “think” like a computer processes instructions.
Data Hazards ...

From the “code” perspective

Instruction 1: \( a = b + c \)
Instruction 2: \( d = a + e \)

When we see such code in a program, we assume that instruction 2 is “executed” after instructions 1 is “executed,” which is a requirement, because the operand in instruction 2 is the output of instruction 1.

Because CPUs employ pipelines, it’s not that straight-forward.

To see how/why data hazards come up, let’s think about this from a register perspective
Data Hazards ...

From the “code” perspective
Instruction 1: \( a = b + c \)
Instruction 2: \( d = a + e \)

From the “register” perspective
Instruction 1: \( R3 <- R1 + R2 \)
Instruction 2: \( R4 <- R3 + R5 \)

Q: What does the above register terminology specify?
Data Hazards ...

From the “code” perspective

Instruction 1: \( a = b + c \)
Instruction 2: \( d = a + e \)

From the “register” perspective

Instruction 1: \( R3 <- R1 + R2 \)
Instruction 2: \( R4 <- R3 + R5 \)

Q: What does the above register terminology specify?

Q: Assuming each stage of the 4-stage pipeline consumes a single clock cycle, how many clock cycles are needed to complete the above 2 instructions?
Data Hazards ...

Clock cycle = 0

Instruction 1 : R3 <- R1 + R2

Instruction 2 : R4 <- R3 + R5

At time 0, which stages of the pipeline are being utilized?
Clock cycle = 1

Instruction 1: R3 <- R1 + R2

Instruction 2: R4 <- R3 + R5

At clock cycle 1, which stage(s) of the pipeline are being utilized?
Data Hazards ...

Clock cycle = 1

Instruction 1 : R3 <- R1 + R2

Instruction 2 : R4 <- R3 + R5

At clock cycle 1, which stage(s) of the pipeline are being utilized?
Data Hazards ...

Clock cycle = 2

Instruction 1 : \( R3 \leftarrow R1 + R2 \)

Instruction 2 : \( R4 \leftarrow R3 + R5 \)

At clock cycle 2, which stage(s) of the pipeline are being utilized?
At clock cycle 2, which stage(s) of the pipeline are being utilized?

Are there any issues (yet)?
Data Hazards ...

Clock cycle = 3

Instruction 1: R3 <- R1 + R2
Instruction 2: R4 <- R3 + R5

At clock cycle 3, which stage(s) of the pipeline are being utilized?
Data Hazards ...

Clock cycle = 3

Instruction 1 : R3 <- R1 + R2
Instruction 2 : R4 <- R3 + R5

At clock cycle 3, which stage(s) of the pipeline are being utilized?

Are there any issues (yet)?
Data Hazards ...

At clock cycle 3, which stage(s) of the pipeline are being utilized?

Are there any issues (yet)?

During clock cycle 3, instruction 2 is being decoded, which states, “get values from registers 3 and 5” ... but instructions 1 has not yet completed, because its WB stage has not yet been performed!

This is called a Read After Write (RAW) Data hazard

Notice that this is NOT a structural hazard

How do we fix this?
Data Hazards ...

Clock cycle = 3

Instruction 1: R3 ← R1 + R2
Instruction 2: R4 ← R3 + R5

We do NOT perform ID of Instruction 2 until AFTER WB of instruction 1 completes
Data Hazards ...

Instruction 1: \( R3 \leftarrow R1 + R2 \)  
Instruction 2: \( R4 \leftarrow R3 + R5 \)

Clock cycle = 4

We do NOT perform ID of Instruction 2 until AFTER WB of instruction 1 completes
Data Hazards...

Clock cycle = 5

Instruction 1: R3 <- R1 + R2

Instruction 2: R4 <- R3 + R5

We do NOT perform ID of Instruction 2 until AFTER WB of instruction 1 completes
Data Hazards ...

Clock cycle = 6

Instruction 1: $R3 \leftarrow R1 + R2$

Instruction 2: $R4 \leftarrow R3 + R5$

We do NOT perform ID of Instruction 2 until AFTER WB of instruction 1 completes
Data Hazards ...

Instruction 1: R3 ← R1 + R2

Instruction 2: R4 ← R3 + R5

We do NOT perform ID of Instruction 2 until AFTER WB of instruction 1 completes
Stalls
Instruction Dependencies