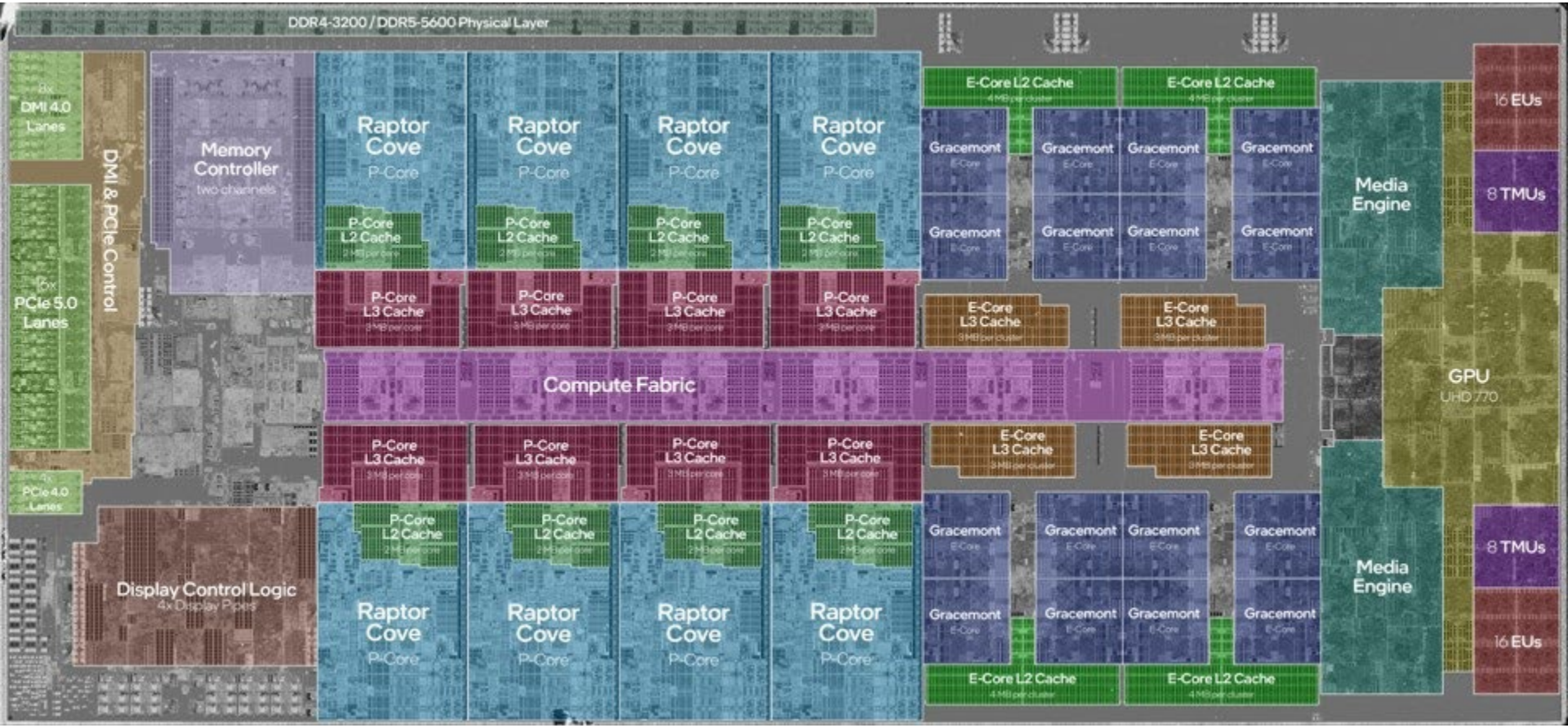
Contemporary Computer Architecture TDSN13

LECTURE 3 – PIPELINE, INSTRUCTION EXECUTION

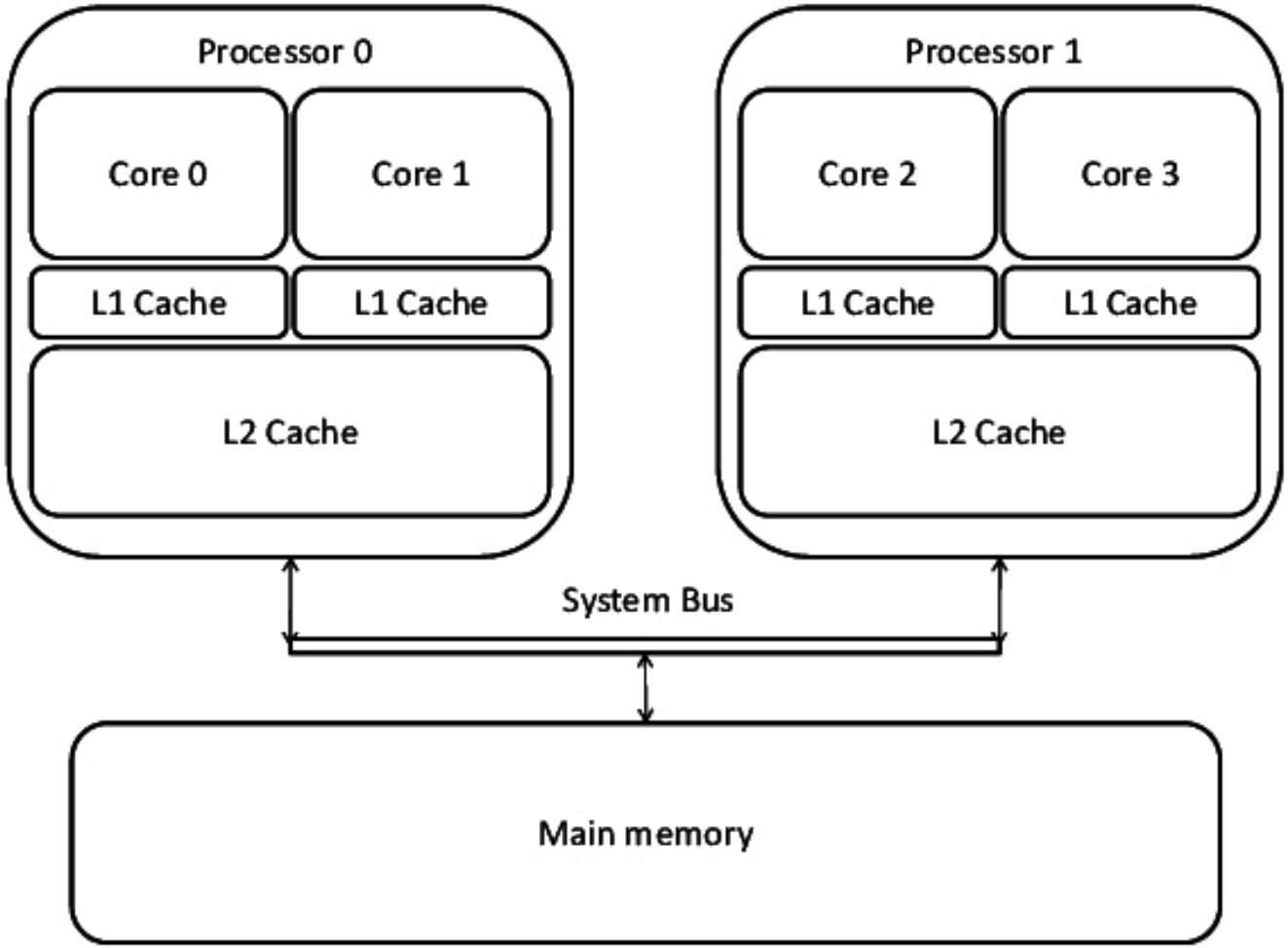
ANDREAS AXELSSON (ANDREAS.AXELSSON@JU.SE)

Intel Core i9-13900K die shot



By JmsDoug and Fritzchens Fritz - Own work based on: Intel@intel7(10nmESF)@RaptorLake@RPL(8P+16E)@i9-13900K@ES DSCx05 poly@5xExt.jpg

Multi-core / Multi-processor



https://www.itrelease.com/2020/07/advantages-and-disadvantages-of-multi-core-processors/

Cache Organization

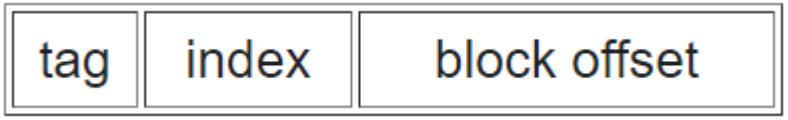
Cache memory divided into **cache lines** or **cache blocks**

tag – is part of the adress

data block – containt the actual memory content

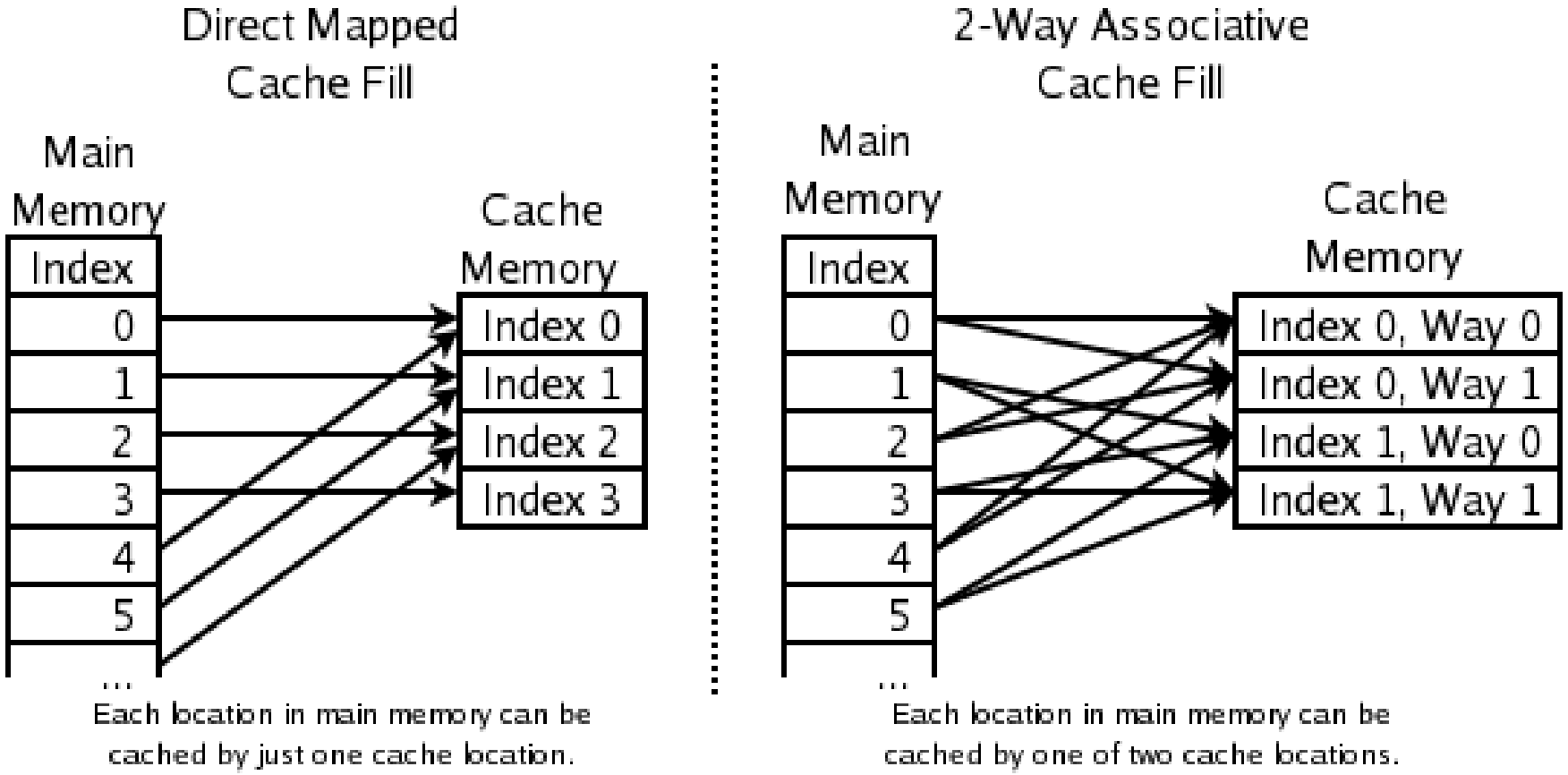
flag bits – containts flags such as if memory has been changed (dirty)

To find correct cache line/block the memory adress that is accessed by the CPU is used as a key.



https://en.wikipedia.org/wiki/CPU_cache

Direct Mapped/2-Way Associative

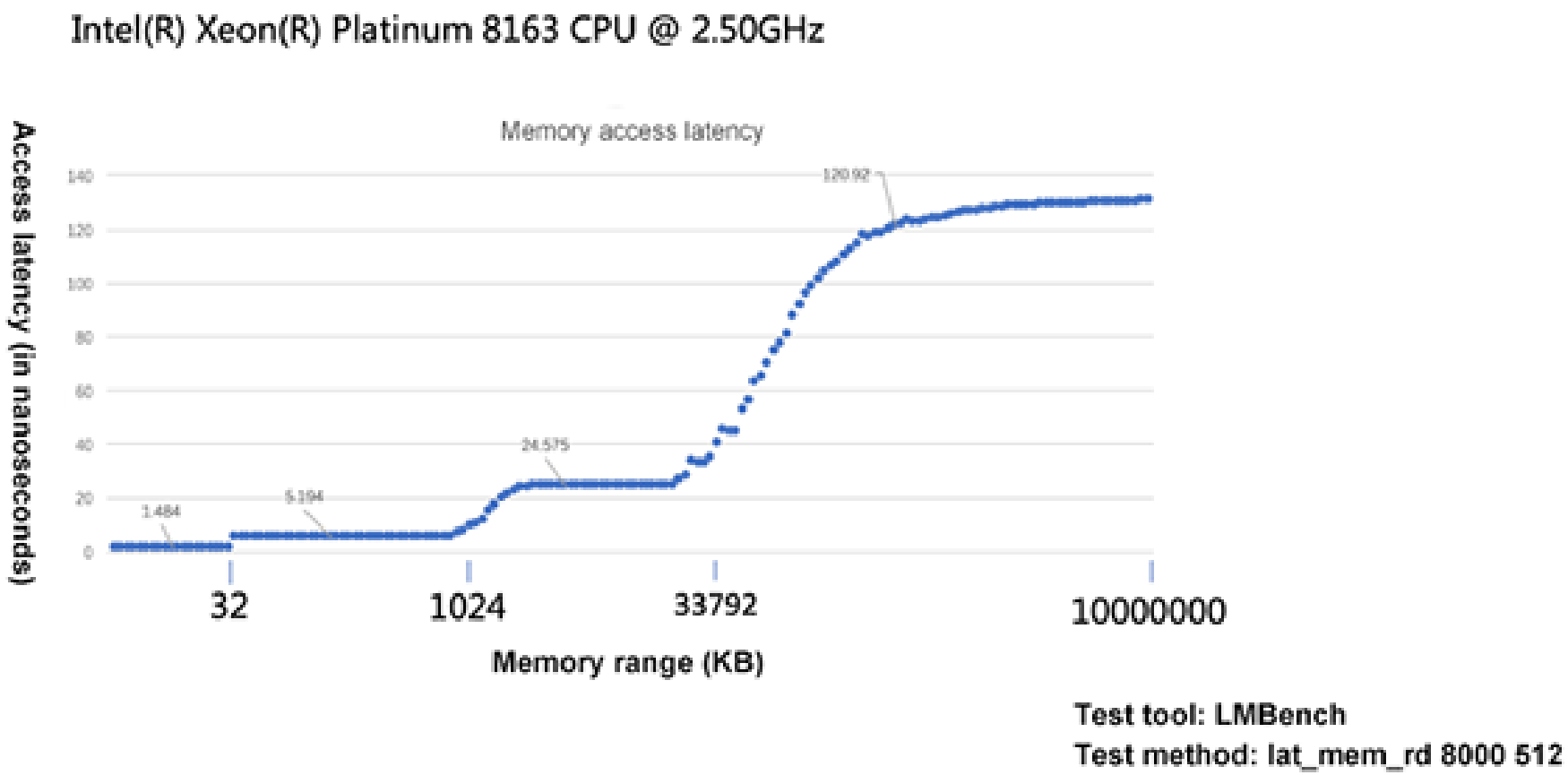


Cache Organization

- Direct mapped cache – good best-case time, but unpredictable in the worst case
- Two-way set associative cache
- Two-way skewed associative cache
- Four-way set-associative cache
- Eight-way set-associative cache, a common choice for later implementations
- 12-way set associative cache, similar to eight-way
- Fully associative cache – the best miss rates, but practical only for a small number of entries

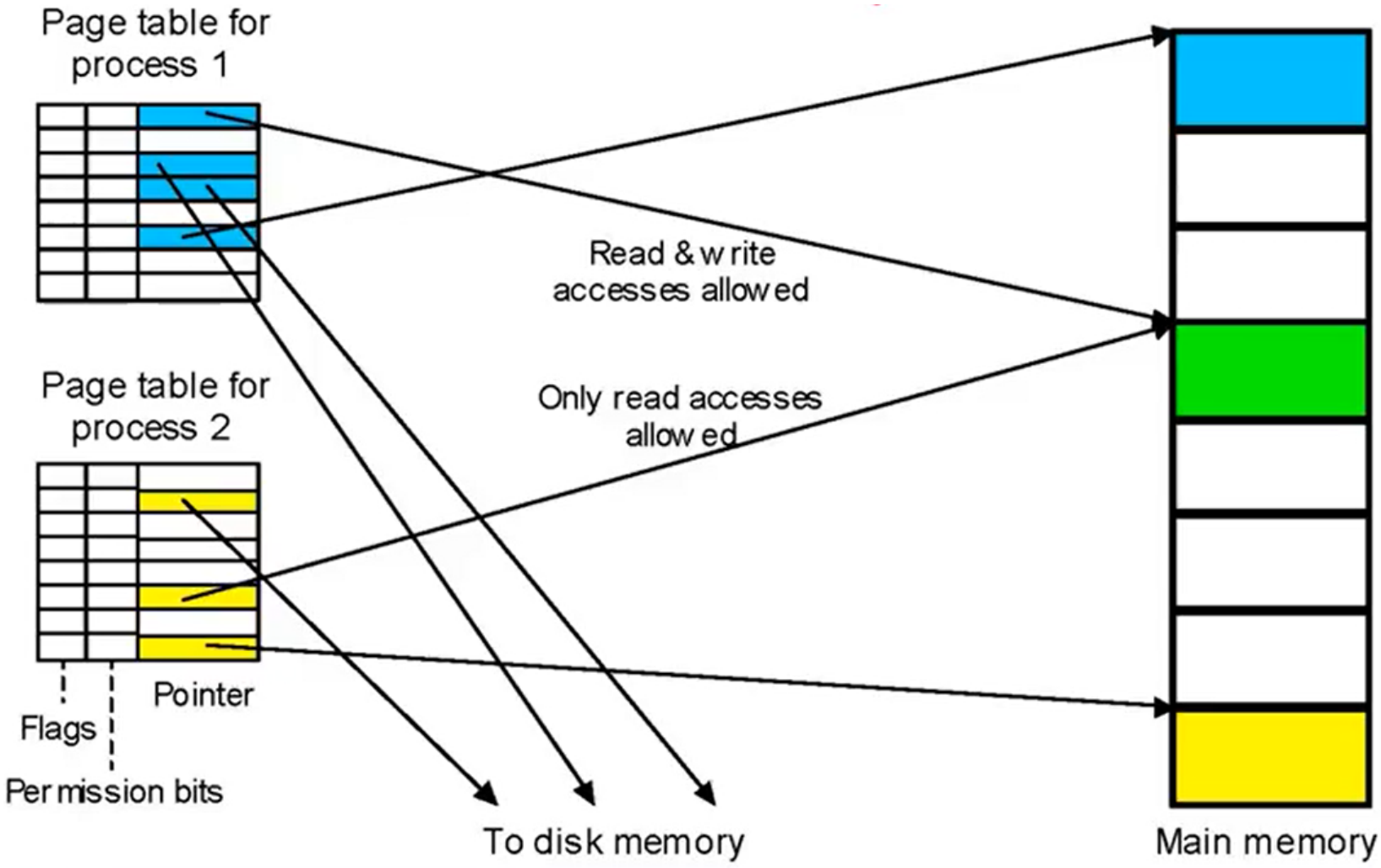
Each of the methods have different rates of **cache hits** or **cache misses**

Cache performance differences



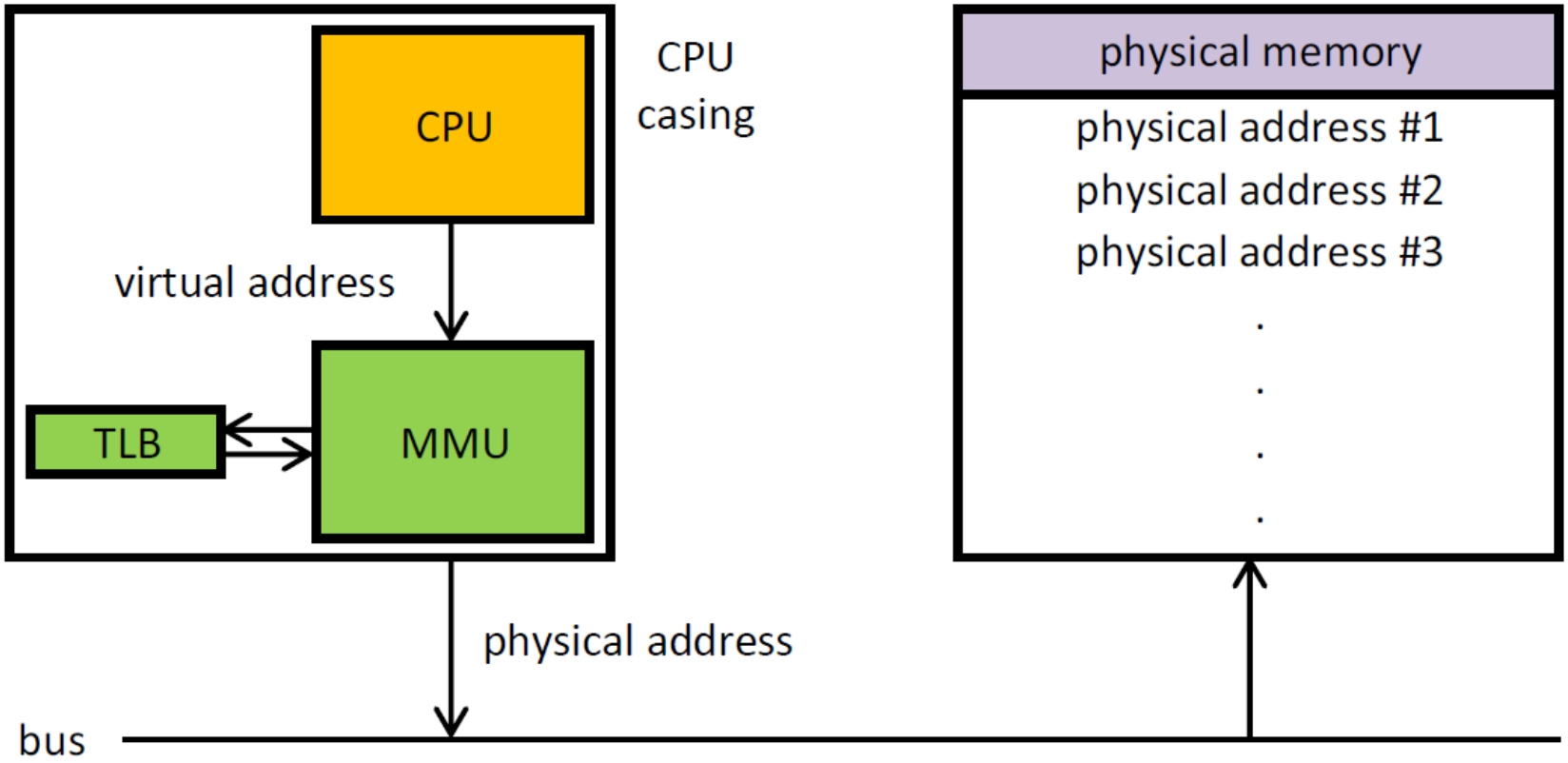
https://www.alibabacloud.com/blog/understanding-memory-performance_594708

Virtual Memory



https://www.youtube.com/watch?v=ICnr87A5m7c

Virtual Memory



CPU: Central Processing Unit

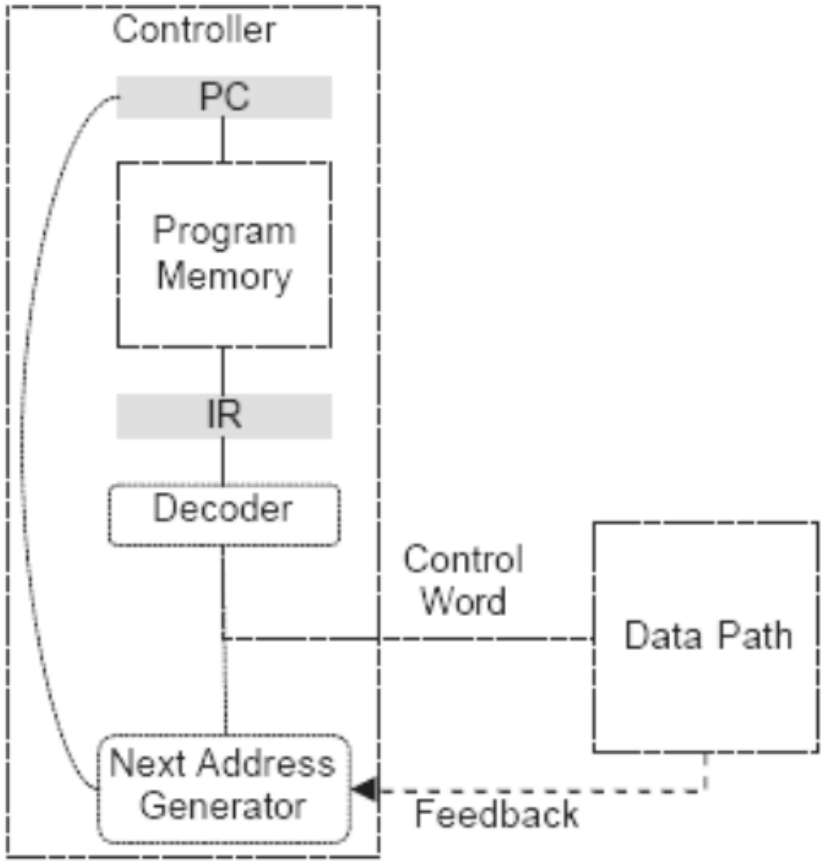
MMU: Memory Management Unit

TLB: Translation lookaside buffer

(TLB can be called an address-translation cache)

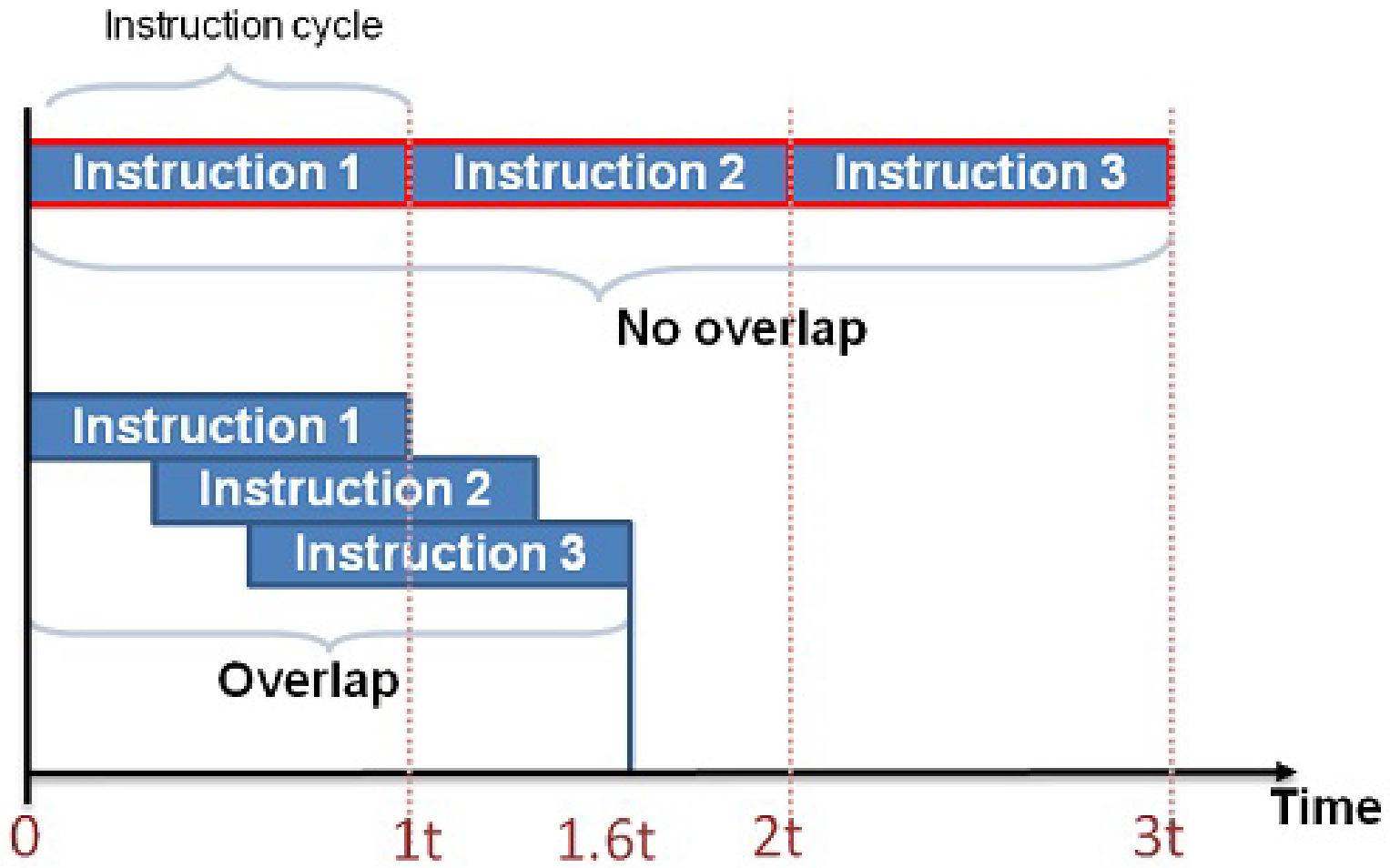
https://en.wikipedia.org/wiki/Memory_management_unit

RISC



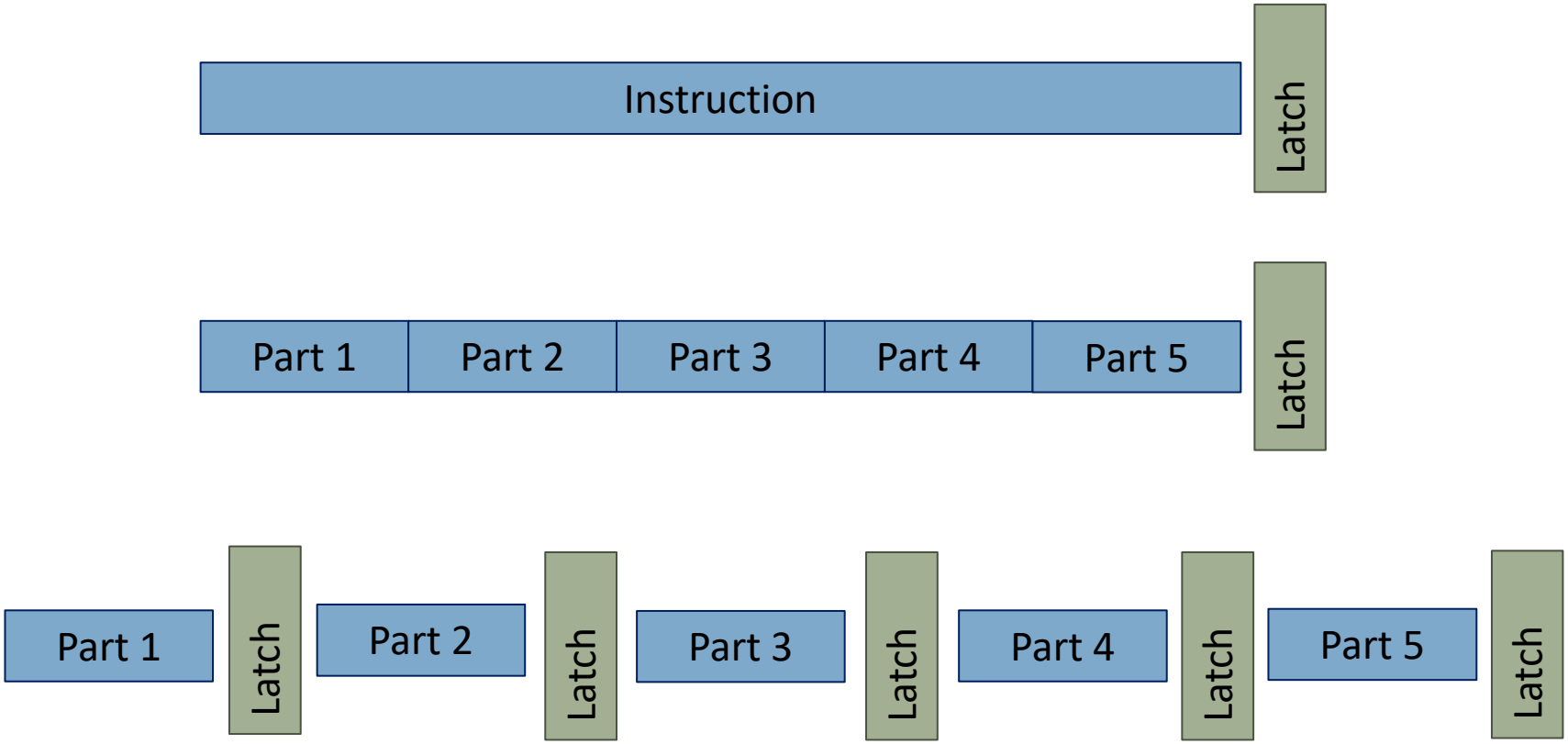
RISC
Simple Instructions
No μ-programming
RISC PM = N x CISC PM

Pipelining introduction



https://witscad.com/course/computer-architecture/chapter/concepts-of-pipelining

Piplining introduction



Piplining introduction

- Unpipelined: time to execute one instruction = T + Toh
- For an N-stage pipeline, time per stage = T/N + Toh
- Total time per instruction = N (T/N + Toh) = T + NToh
- Clock cycle time = T/N + Tovh
- Clock speed = 1 / (T/N + Tovh)
- Ideal speedup = (T + Tovh) / (T/N + Tovh)
- Cycles to complete one instruction = N
- Average IPC (instructions per cycle) = 1

Piplining example problem

1. Assume a non-pipelined instruction takes 5 ns, with additional 0.2 ns for the time to latch the result.
2. We can pipeline the instruction in 5 equally large steps with a latch in between each step

Answer the following:

- What are the cycle times in the two processors?
- What are the clock speeds?
- What are the IPCs?
- How long does it take to finish one instr?
- What is the speedup from pipelining

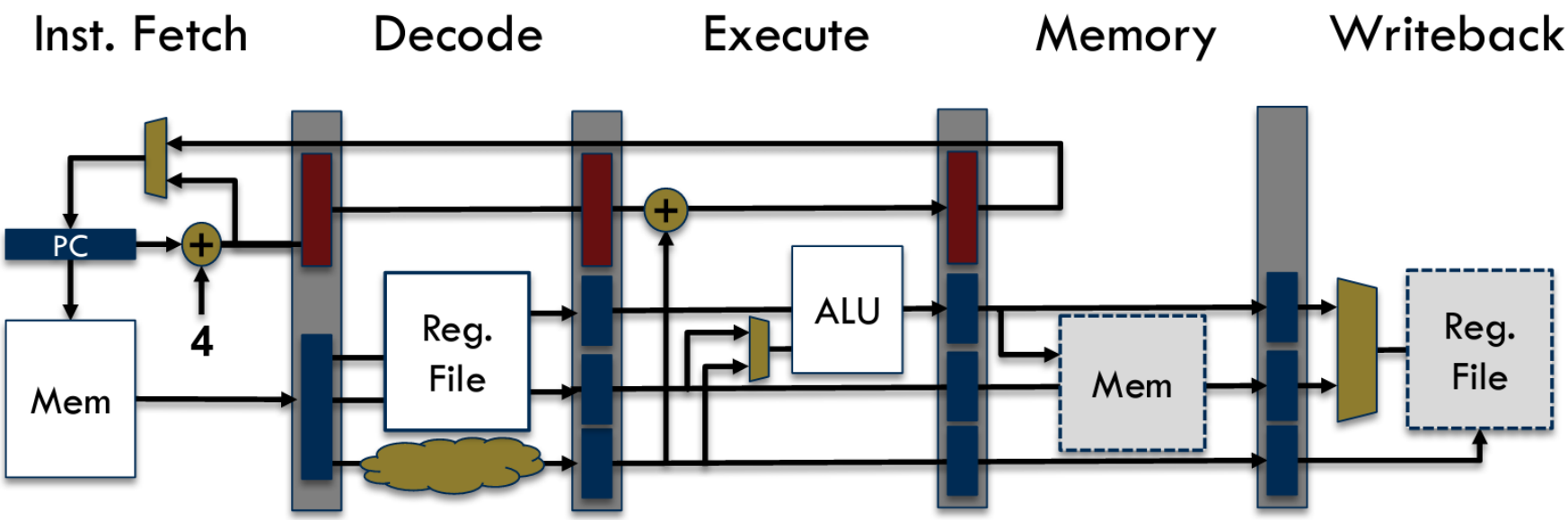
Piplining example problem

1. Assume a non-pipelined instruction takes 5 ns, with additional 0.2 ns for the time to latch the result.
2. We can pipeline the instruction in 5 equally large steps with a latch in between each step

Answer the following:

- What are the cycle times in the two processors? 5.2 ns and 1.2 ns
- What are the clock speeds? 192.3 MHz and 833.3 MHz
- What are the IPCs? 1 and 1
- How long does it take to finish one instruction? 5.2 ns and 6 ns
- What is the speedup from pipelining? 833.3 / 192.3 = 4.33 times

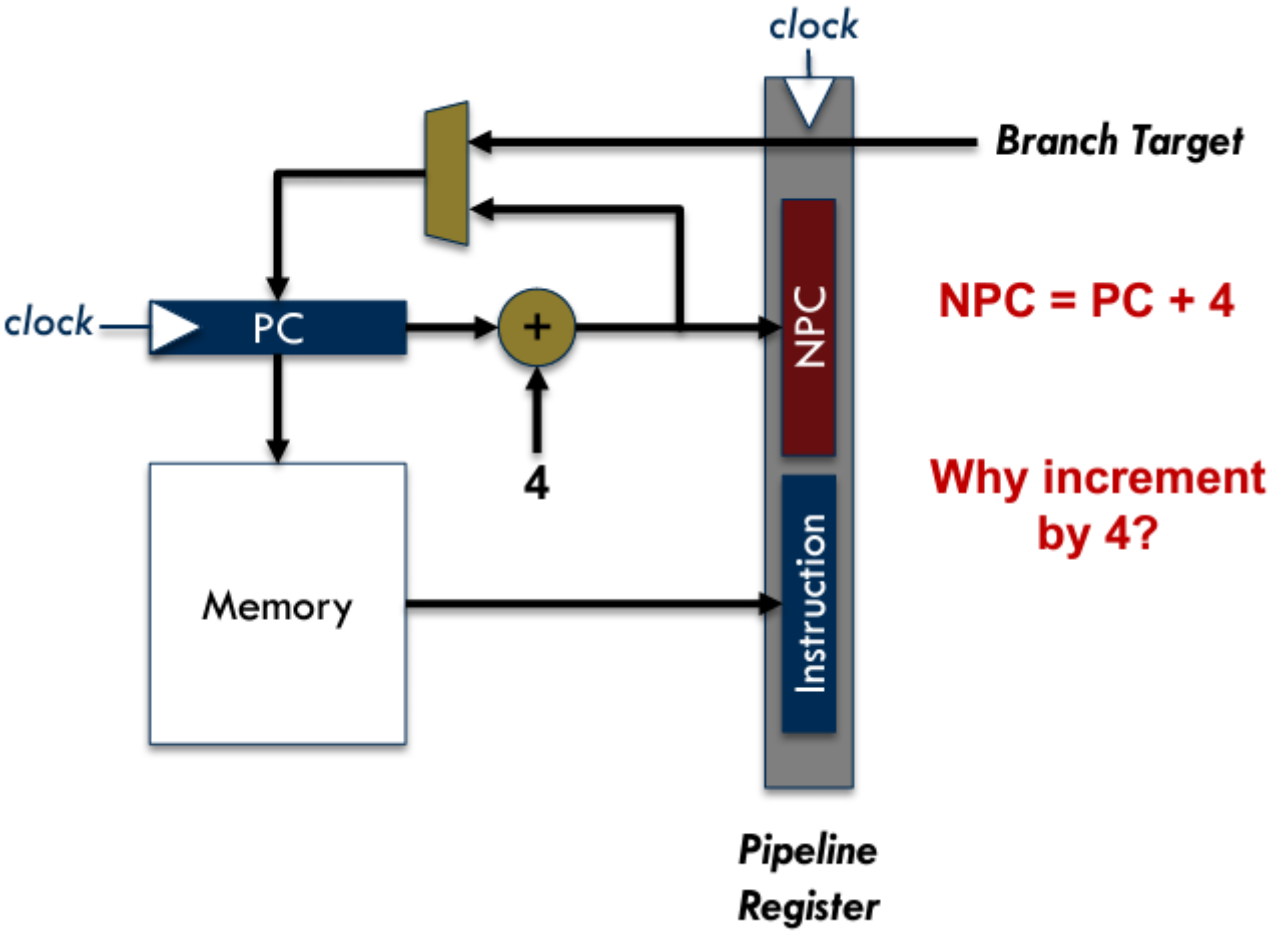
5 Stage Pipelining



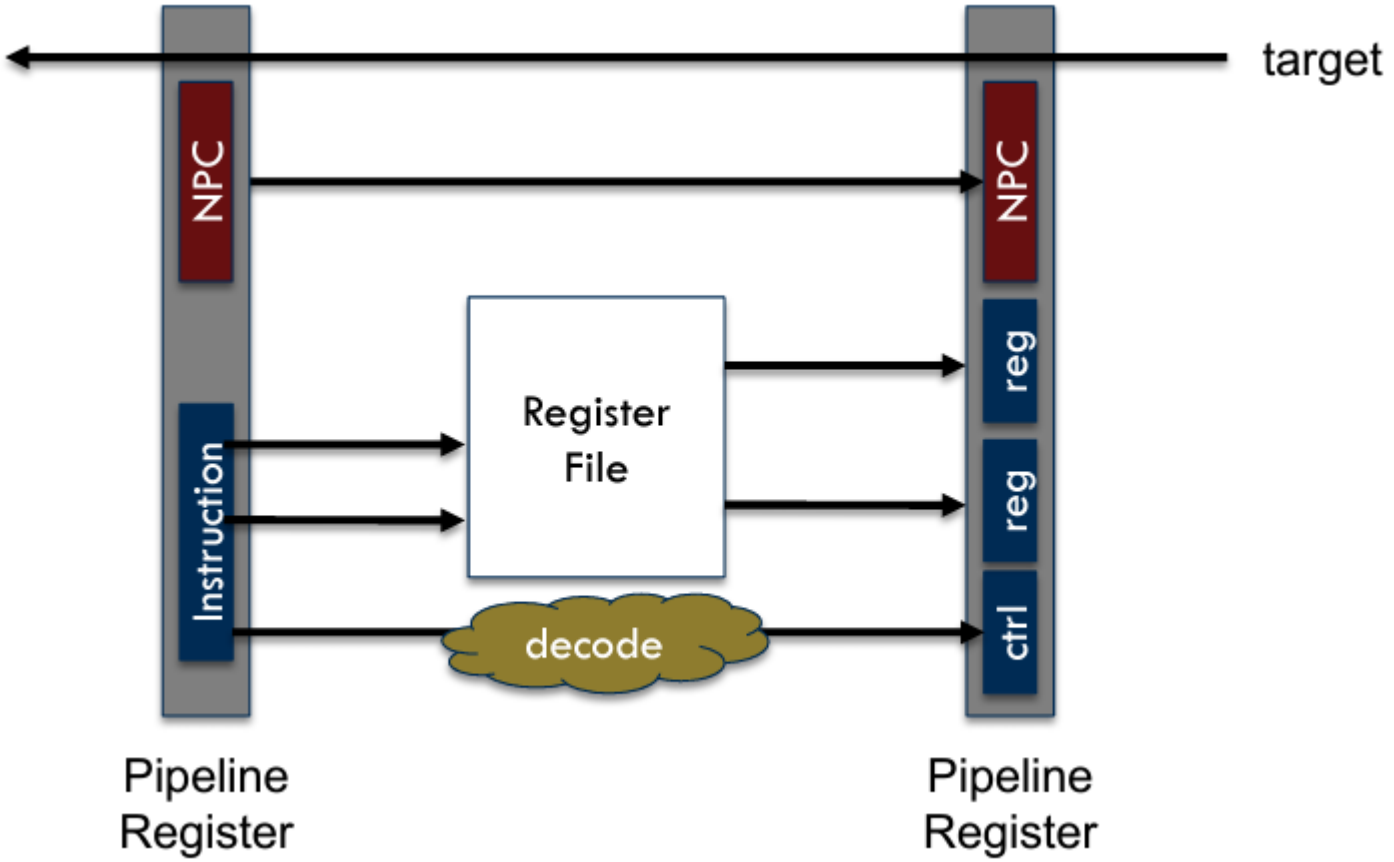
1. Instruction Fetch
2. Decode Instruction
3. Execute Instruction
4. Memory Access
5. Writeback to Registers

https://users.cs.utah.edu/~bojnordi/classes/6810/s18/slides/05-pipelining.pdf

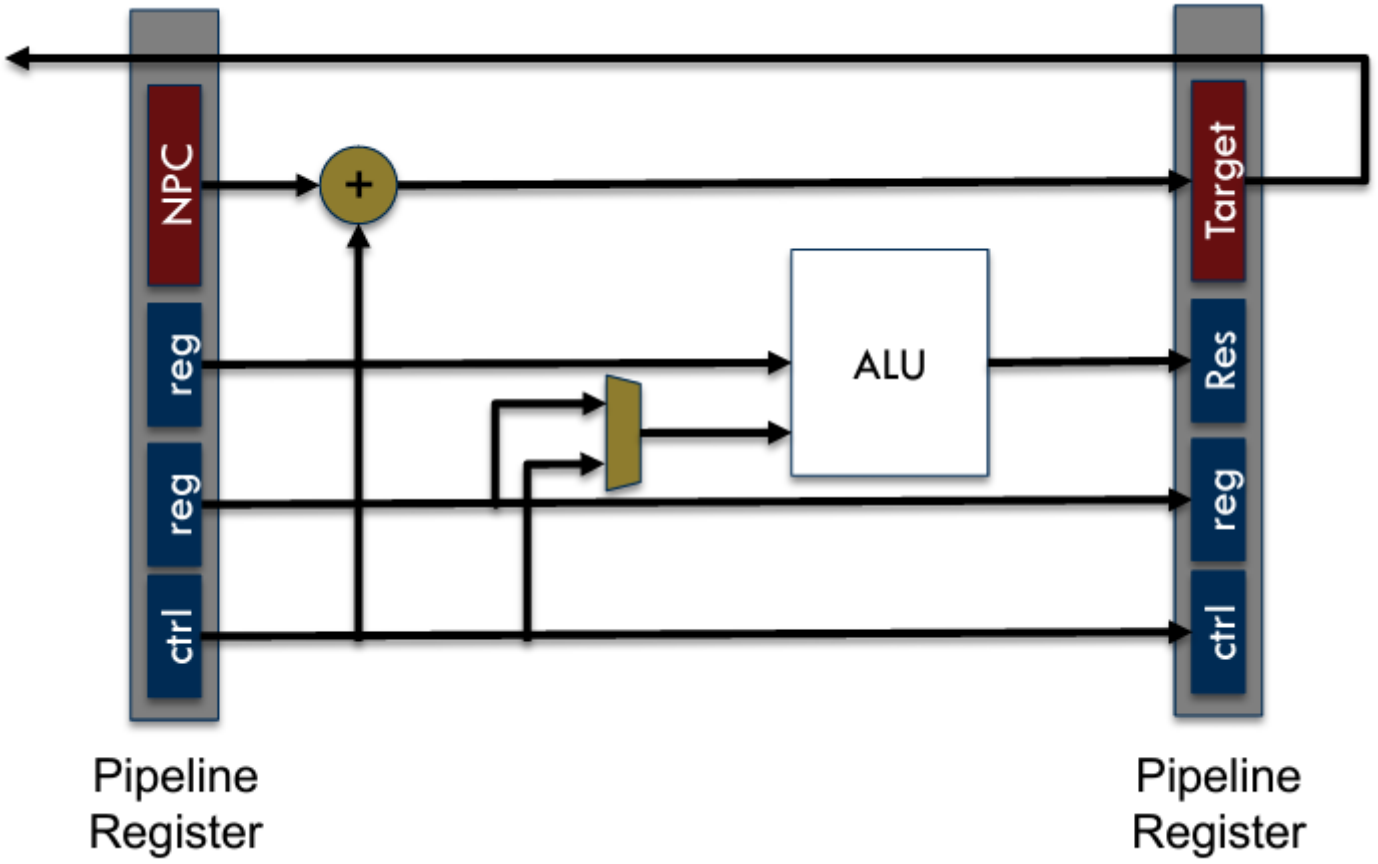
Instruction Fetch



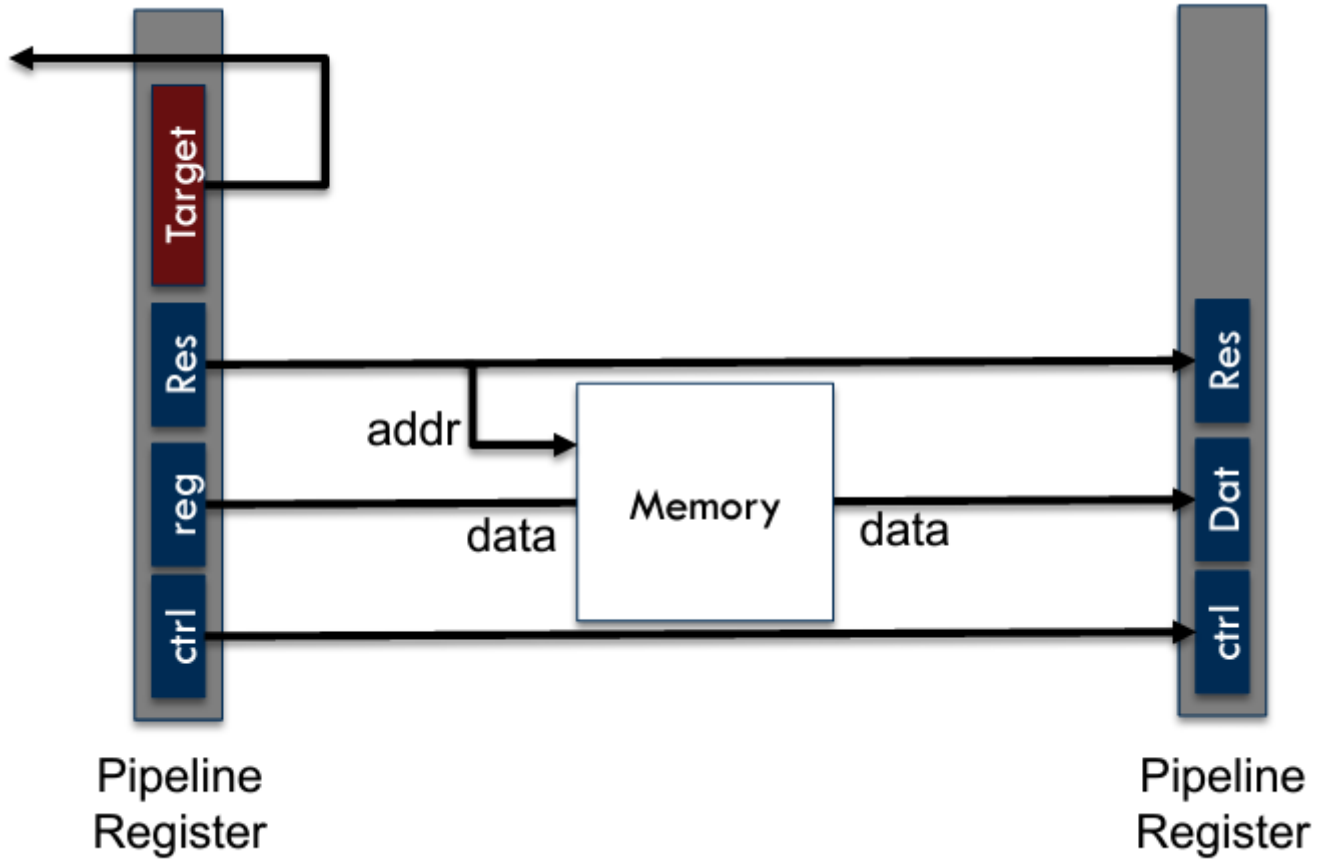
Instruction Decode



Execute Stage



Memory Access

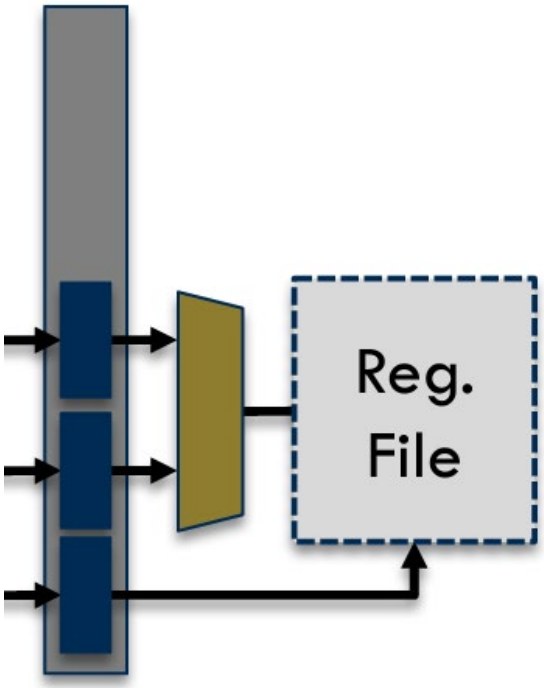


Register Write Back

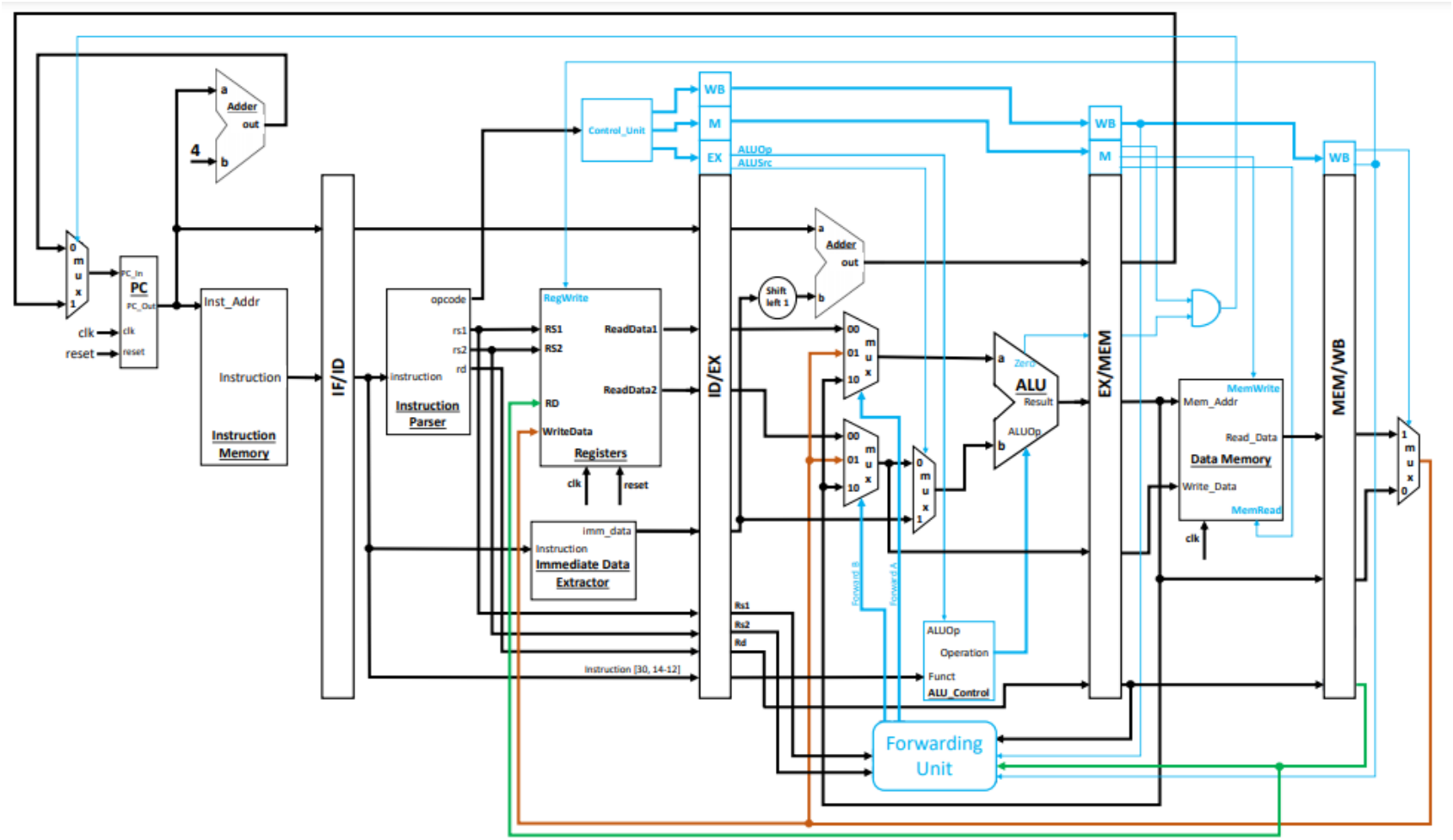
Update register file

Control signals determine if a register write is needed

- Write the ALU result to the destination register, or
- Write the loaded data into the register file

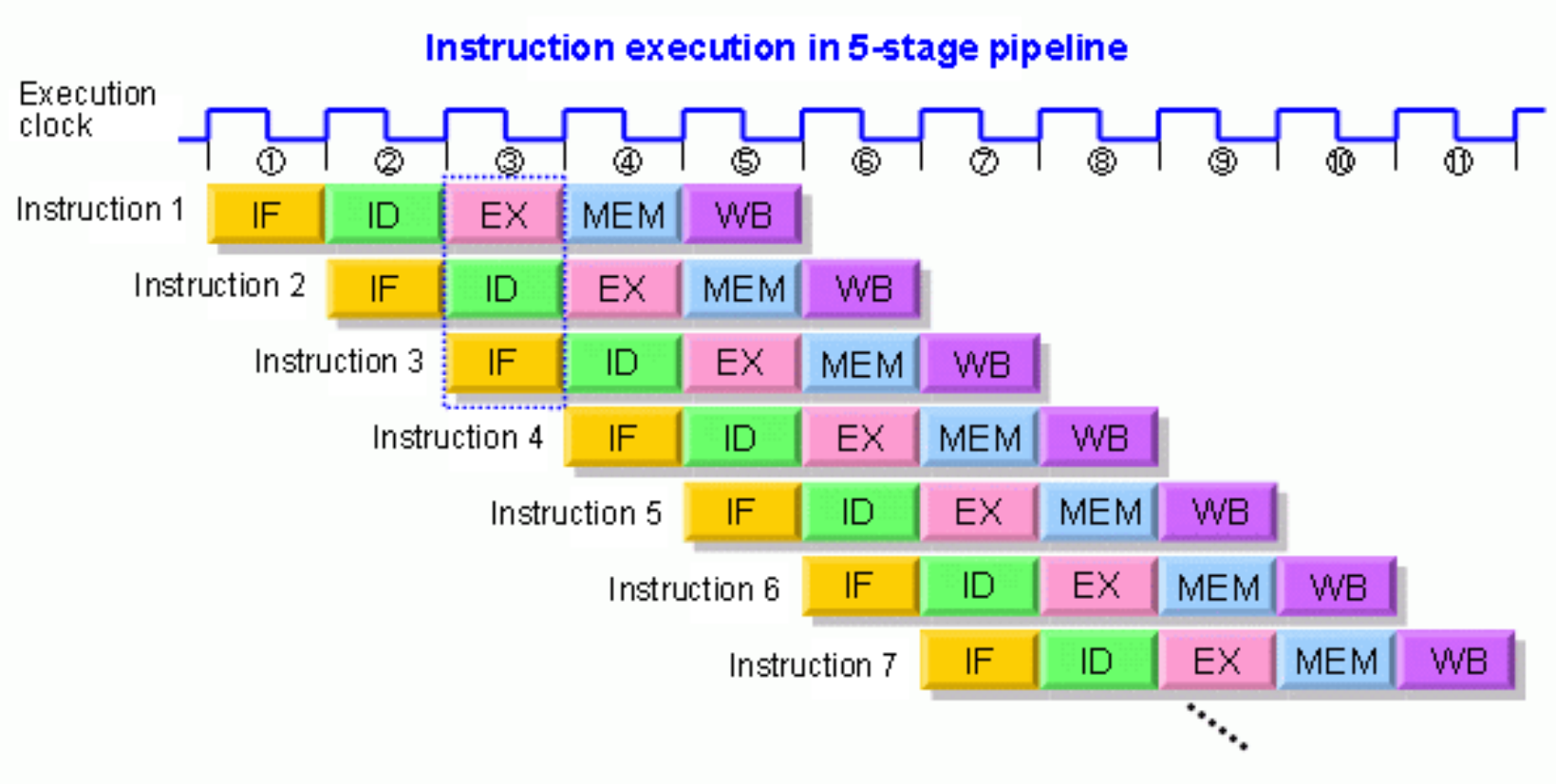


5 Stage Pipelining (Risc-V)



https://github.com/AkeelMedina22/RISC-V-Pipelined-Processor

5 Stage Pipelining



http://www.cs.iit.edu/~cs561/cs350/CPU/5stage.html

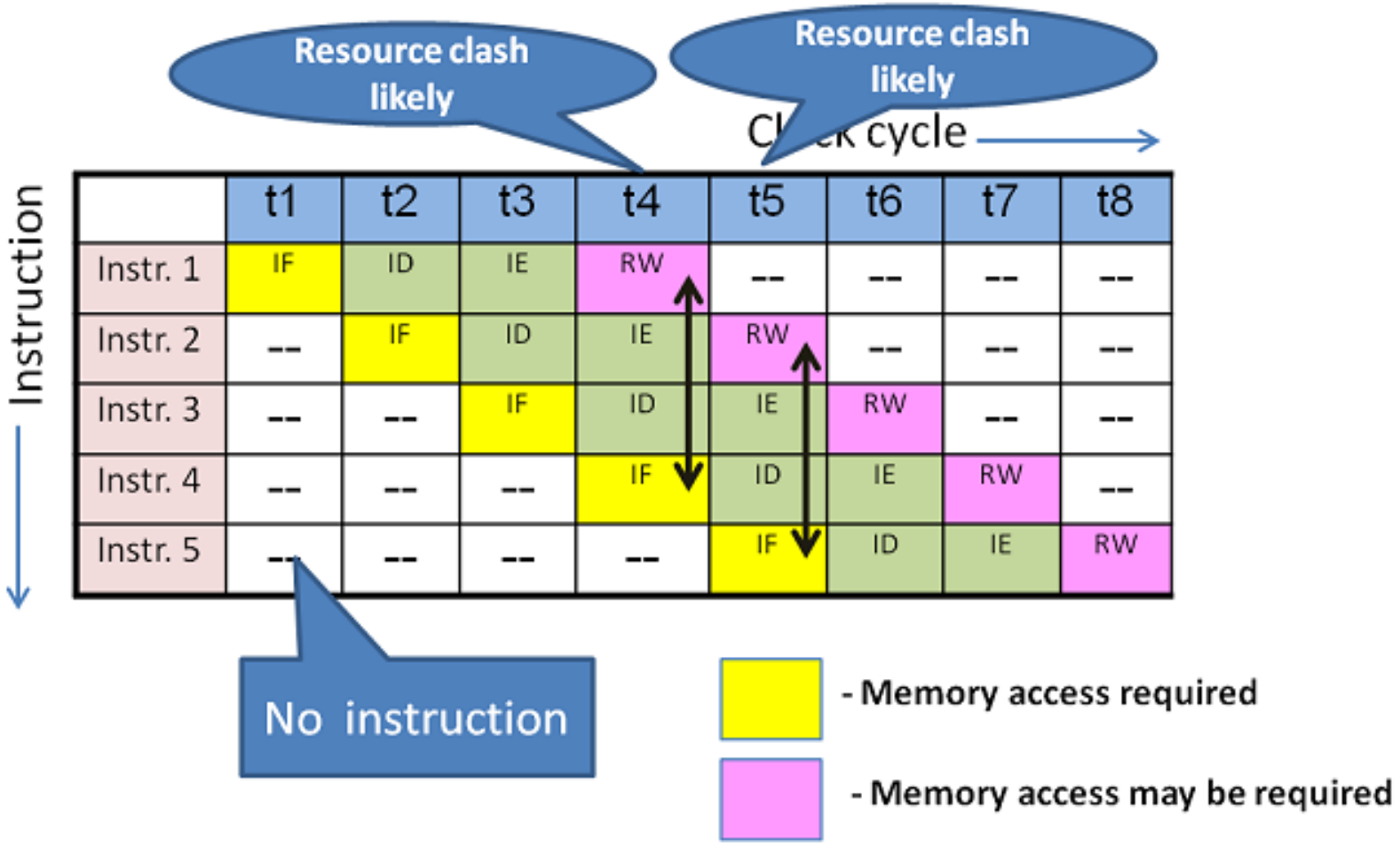
Pipeline Hazards

Structural hazards: Multiple instructions compete for the same resource

Data hazards: A dependent instruction cannot proceed because it needs a value that hasn't been produced

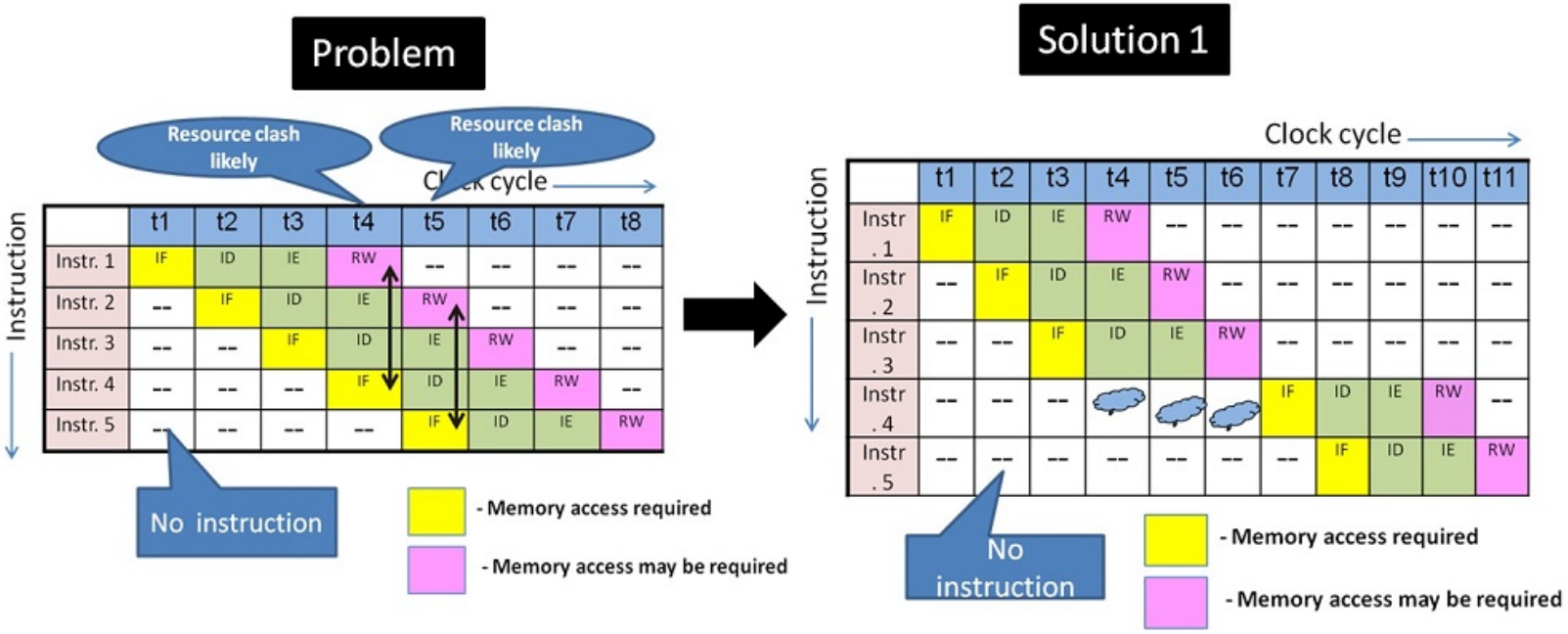
Control hazards: The next instruction cannot be fetched because the outcome of an earlier branch is unknown

Structural Hazards



https://witscad.com/course/computer-architecture/chapter/pipeline-hazards

Structural Hazards



Solution 1: Insert stalls to avoid clash

Solution 2: Use separate instruction and data memories

…

https://witscad.com/course/computer-architecture/chapter/pipeline-hazards

Data Hazards

Consider the following set of instructions in a 5-stage pipeline.
Operands are read in ID.
MEM is memory Write for result; RW is Register Write for result

R3 is accessed in READ mode; expect the result of ADD to be available in **R3**

ADD **R3**, R6, R5	- Results to be written in R3
SUB R4, **R3**, R5	- R3 has one of the operand
OR R6, **R3**, R7	- R3 has one of the operand
AND R8, **R3**, R7	- R3 has one of the operand
XOR R12, **R3**, R10	- R3 has one of the operand

But result of ADD written in R3 at t5

	t1	t2	t3	t4	t5	t6	t7	t8	t9
ADD R3, R6, R5	IF	ID	IE	MEM	RW R3	--	--	--	--
SUB R4, R3, R5	--	IF	ID R3	IE	MEM	RW	--	--	--
OR R6, R3, R7	--	--	IF	ID R3	IE	MEM	RW	--	--
AND R8, R3, R9	--	--	--	IF	ID R3	IE	MEM	RW	--
XOR R10, R3, R11	--	--	--	--	IF	ID R3	IE	MEM	RW

Note when each instruction is accessing R3

https://witscad.com/course/computer-architecture/chapter/pipeline-hazards

Data Hazards

Consider the following set of instructions in a 5-stage pipeline.
Operands are read in ID.
MEM is memory Write for result; RW is Register Write for result

R3 is accessed in READ mode; expect the result of ADD to be available in **R3**

ADD **R3**, R6, R5	- Results to be written in R3
SUB R4, **R3**, R5	- R3 has one of the operand
OR R6, **R3**, R7	- R3 has one of the operand
AND R8, **R3**, R7	- R3 has one of the operand
XOR R12, **R3**, R10	- R3 has one of the operand

But result of ADD written in R3 at t5

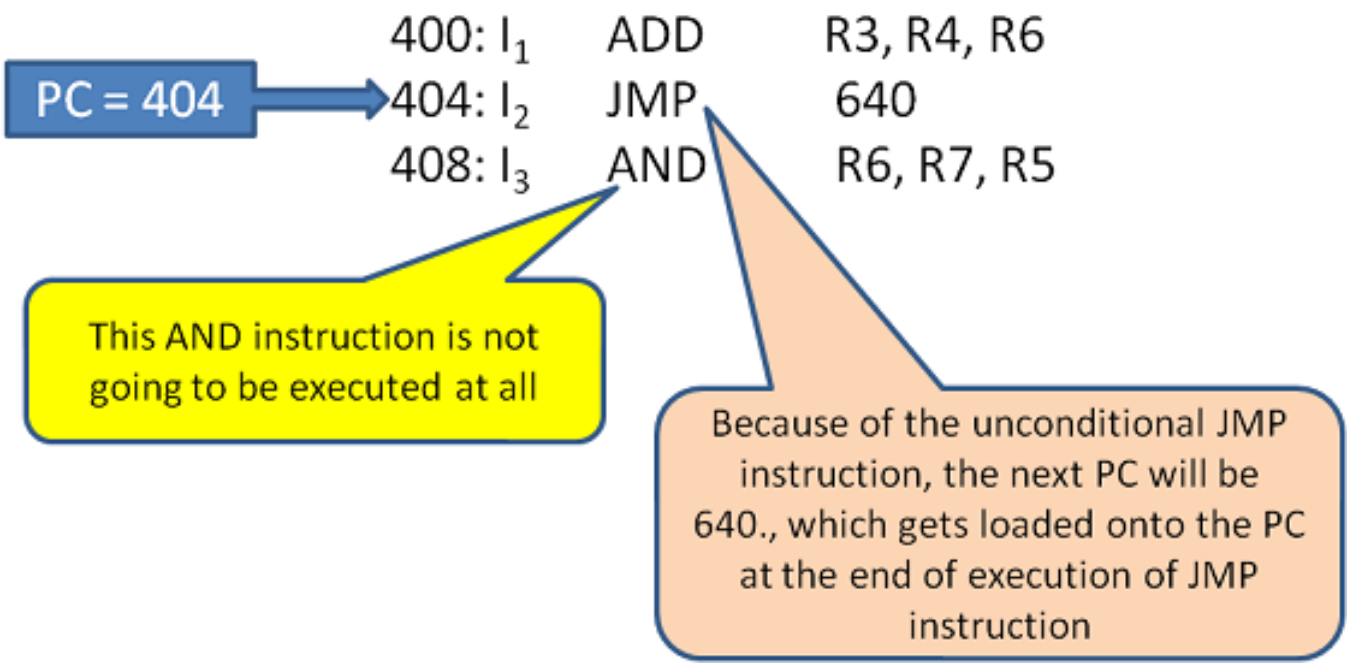
	t1	t2	t3	t4	t5	t6	t7	t8	t9
ADD R3, R6, R5	IF	ID	IE	MEM	RW R3	--	--	--	--
SUB R4, R3, R5	--	IF	ID R3	IE	MEM	RW	--	--	--
OR R6, R3, R7	--	--	IF	ID R3	IE	MEM	RW	--	--
AND R8,									
XOR R10,									

Note when each instruction is accessing R3

Solution 1: Insert stalls (bubbles) after SUB instruction fetch
Solution 2: Data forwarding
Solution 3: Compiler optimizes instruction order
Solution 4: Insert software stall (NOP)

https://witscad.com/course/computer-architecture/chapter/pipeline-hazards

Control Hazards



Solutions:

1. Stall
2. Prediction
3. Dynamic Branch Prediction
4. Reordering instructions

https://witscad.com/course/computer-architecture/chapter/pipeline-hazards

Introduction

Questions?

Contact information

Andreas Axelsson

Email: andreas.axelsson@ju.se

Mobile: 0709-467760