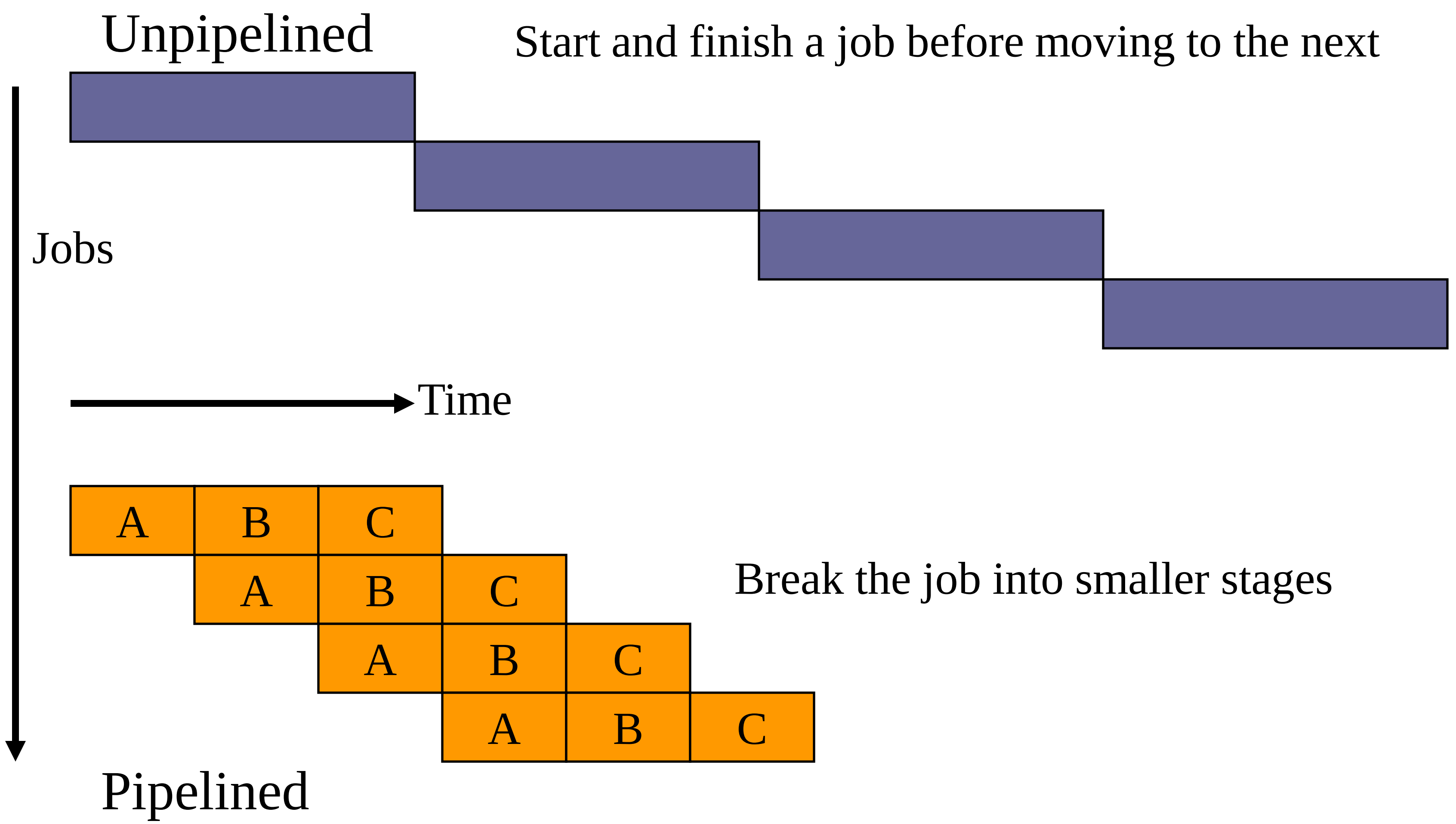
Computer Architecture

Dr. Haroon Mahmood

Assistant Professor

NUCES Lahore

Pipeline


Unpipelined
Start and finish a job before moving to the next
Jobs
Time
A
B
C
A
B
C
A
B
C
A
B
C
Break the job into smaller stages
Pipelined

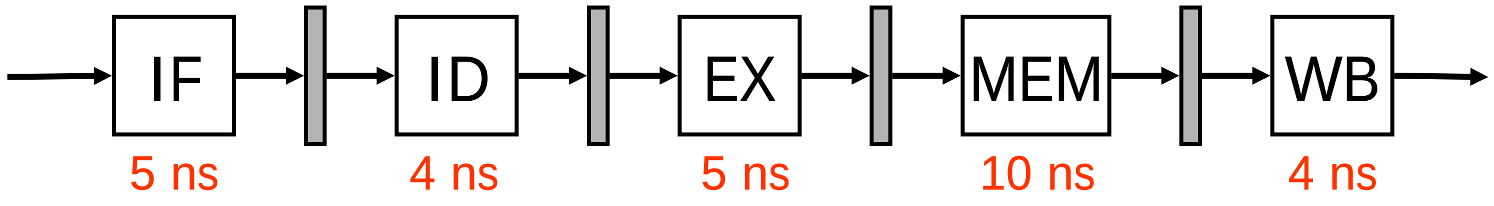
Pipelining

- Pipelining is a key implementation technique used to build fast processors. It allows the execution of multiple instructions to overlap in time.

- A pipeline within a processor is similar to a car assembly line. Each assembly station is called a pipe stage or a pipe segment.

- The throughput of an instruction pipeline is the measure of how often an instruction exits the pipeline.

Pipeline Throughput and Latency

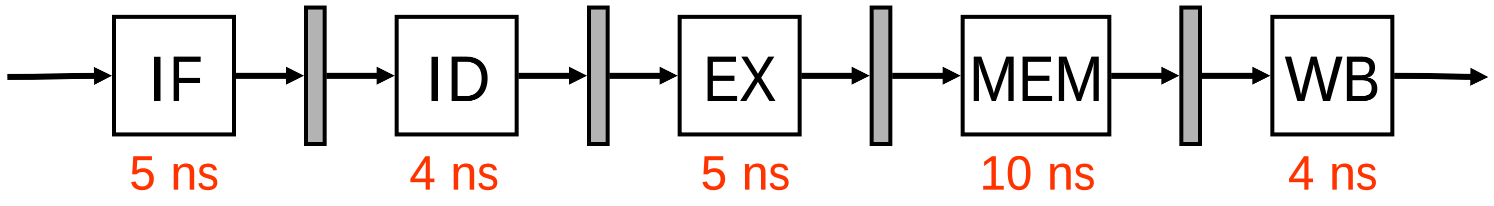


- **Consider the pipeline above with the indicated delays. We want to know what is the pipeline throughput and the pipeline latency.**

- **Pipeline throughput: instructions completed per second.**

- **Pipeline latency: how long does it take to execute a single instruction in the pipeline.**

Pipeline Throughput and Latency



- **Pipeline throughput:**

$$= 1 instr / \max[lat(IF), lat(ID), lat(EX), lat(MEM), lat(WB)]$$
$$= 1 instr / \max[5ns, 4ns, 5ns, 10ns, 4ns]$$
$$= 1 instr / 10ns \quad (ignoring\ pipeline\ register\ overhead)$$

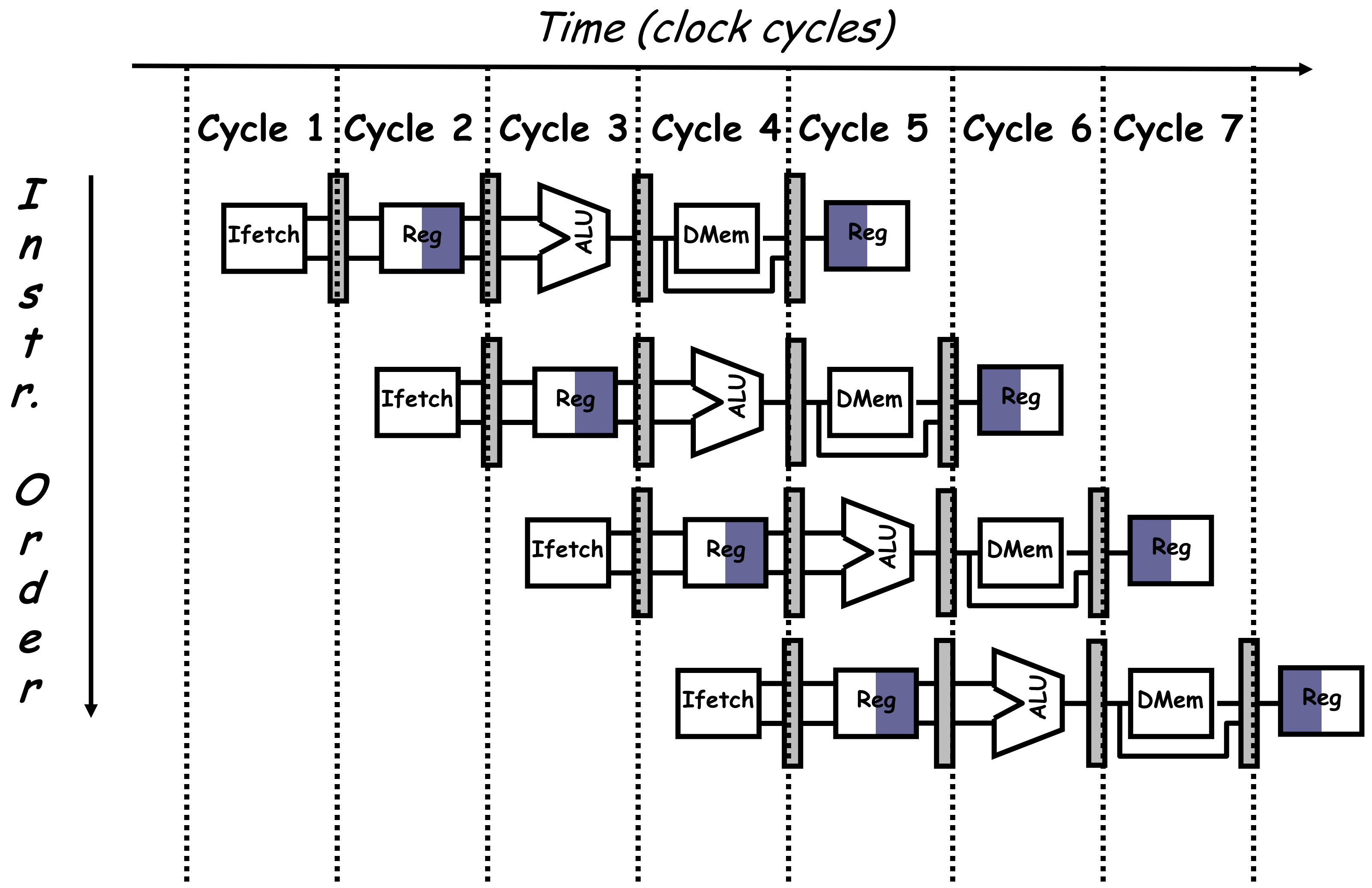
- **Pipeline Instruction latency:**

$$= No.of stages * \max[lat(IF), lat(ID), lat(EX), lat(MEM), lat(WB)]$$
$$= 5 * \max[5ns, 4ns, 5ns, 10ns, 4ns]$$
$$= 50ns \quad (ignoring\ pipeline\ register\ overhead)$$

Note

- Pipelining doesn’t help latency of single task, it helps throughput of entire workload
- Pipeline rate limited by slowest pipeline stage
- Multiple tasks operating simultaneously
- Potential speedup = Number of pipe stages
- Unbalanced lengths of pipe stages reduces speedup
- Time to “fill” pipeline and time to “drain” it reduces speedup

Visualizing Pipelining



Some Equations

- Unpipelined: time to execute one instruction = T + Tovh
- For an N-stage pipeline, time per stage = T/N + Tovh
- Total time per instruction = N (T/N + Tovh) = T + N Tovh
- 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 CPI (cycles per instr) = 1

Problem 1

- An unpipelined processor takes 5 ns to work on one instruction. It then takes 0.2 ns to latch its results into latches. We were able to convert the circuit into 5 equal sequential pipeline stages. Answer the following, assuming that there are no stalls in the pipeline.

- 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?

Solution

- An unpipelined processor takes 5 ns to work on one instruction. It then takes 0.2 ns to latch its results into latches. We were able to convert the circuit into 5 equal sequential pipeline stages. Answer the following, assuming that there are no stalls in the pipeline.

- What are the cycle times in the two processors?
  5.2ns and 1.2ns
- What are the clock speeds? 192 MHz and 833 MHz
- What are the IPCs? 1 and 1
- How long does it take to finish one instr? 5.2ns and 6ns
- What is the speedup from pipelining? 833/192 = 4.34

Problem 2

- An unpipelined processor takes 5 ns to work on one instruction. It then takes 0.2 ns to latch its results into latches. I was able to convert the circuits into 5 sequential pipeline stages. The stages have the following lengths: 1ns; 0.6ns; 1.2ns; 1.4ns; 0.8ns. Answer the following, assuming that there are no stalls in the pipeline.

- What is the cycle time in the new processor?
- What is the clock speed?
- What is the IPC?
- How long does it take to finish one instr?
- What is the speedup from pipelining?

Solution

- An unpipelined processor takes 5 ns to work on one instruction. It then takes 0.2 ns to latch its results into latches. I was able to convert the circuits into 5 sequential pipeline stages. The stages have the following lengths: 1ns; 0.6ns; 1.2ns; 1.4ns; 0.8ns. Answer the following, assuming that there are no stalls in the pipeline.

- What is the cycle time in the new processor? 1.6ns
- What is the clock speed? 625 MHz
- What is the IPC? 1
- How long does it take to finish one instr? 8ns
- What is the speedup from pipelining? 625/192 = 3.26

Computer Performance

- Response Time (latency)
- — How long does it take for my job to run?
- — How long does it take to execute a job?
- — How long must I wait for the database query?

- Throughput
- — How many jobs can the machine run at once?
- — What is the average execution rate?
- — How much work is getting done?

Clock

- Computer Designers use clock ticks to measure how fast the hardware can perform basic functions

- Clock rate (frequency) = cycles per second.
  - Measured in Hertz (1 Hz = 1 cycle/s).

- Clock period is the time between ticks of the clock and is measured in seconds per cycle.
  - Period = 1/frequency

- Example: A 200 MHz (MegaHertz) clock has a clock period of 5nanoseconds

Time & Clock Metrics

- Determine effect of design change on performance

- CPU execution time = CPU clock cycles X clock cycle time
- CPU execution time = CPU clock cycles/clock rate

- For some program running on machine X,
  $\text{Performance}_X = 1 / \text{Execution time}_X$

- "X is n times faster than Y"
  $\text{Performance}_X \ / \ \text{Performance}_Y = n$

Problem:

- machine A runs a program in 20 seconds
- machine B runs the same program in 25 seconds

How to improve performance?

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

- **So, to improve performance (everything else being equal) you can either**

reduce________ **the # of required cycles for a program, or**
decrease________ **the clock cycle time or, said another way,**
increase________ **the clock rate.**

$$\text{execution time} = \text{number of cycles} \times \text{clock cycle time}$$

Example

- Our favorite program runs in 10 seconds on computer A, which has a 400 Mhz. clock. We are trying to help a computer designer build a new machine B, that will run this program in 6 seconds. The designer can use new (or perhaps more expensive) technology to substantially increase the clock rate, but has informed us that this increase will affect the rest of the CPU design, causing machine B to require 1.2 times as many clock cycles as machine A for the same program. What clock rate should we tell the designer to target?

Example

Let C = number of cycles

Execution time = C X clock cycle time = C/ clock rate

On computer A,

C/ 400 MHz = C/ 400 X 10^6 = 10 seconds => C = 400 X 10^7

On computer B, number of cycles = 1.2 X C

What should be B’s clock rate so that our favorite program has smaller execution time?

1.2 X C/ clock rate = 6 => 1.2 X 400 X 10^7 / 6 = clock rate

I.e. clock rate = 800 MHz

CPU performance equation

- An alternative to "number of clock cycles" is "number of instructions executed" or Instruction Count (IC).
- Given both the "number of clock cycles" and IC of a program, the average clock cycles per instruction (CPI) is given by:

$$\mathrm{CPI} = \frac{\text{CPU Clock Cycles of a Program}}{\mathrm{IC}}$$

$$\text{CPU time} = \mathrm{IC} \times \mathrm{CPI} \times \text{Clock cycle time} = \frac{\mathrm{IC} \times \mathrm{CPI}}{\text{Clock rate}}$$

Example

- Suppose we have two implementations of the same instruction set architecture. Machine A has a clock cycle time of 1 ns (nanoseconds) and a CPI of 2.0 for some program, and machine B has a clock cycle time of 2 ns and a CPI of 1.2 for the same program.

- Which machine is faster for this program, and how much faster is it?

Solution

- We know that each machine executes the same number of instructions for the same program; let's call this number I
- CPU clock cyclesA = I x 2.0 CPU
- clock cyclesB = I x 1.2
-

  CPU timeA = CPU clock cyclesA x Clock cycle timeA
  = I x 2.0 x 1 ns = 2 x I ns
- CPU timeB = I x 1.2 x 2 ns = 2.4 x I ns
- Machine A is faster. How much?

$$\frac{\text{CPU Performance}_A}{\text{CPU Performance}_B} = \frac{\text{Execution time}_B}{\text{Execution time}_A} = \frac{2.4 \times I \text{ ns}}{2 \times I \text{ ns}} = 1.2$$

- Machine A is 1.2 times faster than machine B

Example

- A compiler designer is trying to decide between two code sequences for a particular machine. Based on the hardware implementation, there are three different classes of instructions: Class A, Class B, and Class C, and they require one, two, and three cycles (respectively).
- The first code sequence has 5 instructions: 2 of A, 1 of B, and 2 of C
- The second sequence has 6 instructions: 4 of A, 1 of B, and 1 of C.

- Which sequence will be faster? How much?
- What is the CPI for each sequence?

Solution

$$\text{CPI for sequence } 1 = \frac{2\times1+1\times2+2\times3}{(2+1+2)} = \frac{10}{5}$$

$$\text{CPI for sequence } 2 = \frac{4\times1+1\times2+1\times3}{(4+1+1)} = \frac{9}{6}$$

$$\text{CPU cycles for sequence } 1 = 10/5\times5 = 10$$

$$\text{CPU cycles for sequence } 2 = 9/6\times6 = 9$$

- **Sequence 2 is 11 % faster than sequence 1**

Remember

- A given program will require
  - some number of instructions (machine instructions)
  - some number of cycles
  - some number of seconds

- We have a vocabulary that relates these quantities:
  - cycle time (seconds per cycle)
  - clock rate (cycles per second)
  - CPI (cycles per instruction)

CPU Performance

- For a given instruction set architecture, increases in CPU performance can come from three sources:
  1. lower the instruction count or generate instructions with a lower average CPI
  2. lower the CPI
  3. Increases in clock rate

	Instruction Count	CPI	Clock cycle time
Program	X	X	
Compiler	X	X	
Instruction Set	X	X	
Organization		X	X
Technology			X