Pipelining I

From Chapter 14, Computer Organization and Architecture by William Stallings

Credits: based on slides by Ms. Noshaba Nasir

What Is Pipelining?

- Pipelining is an implementation technique whereby multiple instructions are overlapped in execution.
- It takes advantage of parallelism that exists among the actions needed to execute an instruction.
- Today, pipelining is the key technique used to make fast CPUs.

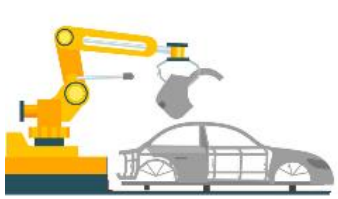
Real World Analogy

- A pipeline is like an assembly line.
- In an automobile manufacturing line, there are many steps, each contributing something to the construction of the car.
- Each step operates in parallel with the other steps, although on a different car.

Real World Analogy

- Automobile manufacturing line **without pipelining**, in 3 stages: Assemble (A), Paint (P), Fix tires (T)
- Assume A+P+T take 3 hours

Timeline	0-3 hours	3-6 hours	6-9 hours
Car 1	A+P+T		
Car 2		A+P+T	
Car 3			A+P+T



Real World Analogy

- Assembly line **with pipelining**
- Assuming each stage takes 1 hour

Time	1st hr	2nd hr	3rd hr	4th hr	5th hr	6th hr	7th hr	8th hr	9th hr
Car 1	A	P	**T**						
Car 2		A	**P**	T					
Car 3			**A**	P	T				

- It takes 5 hours with pipelining and 9 hours without pipelining.
- Speedup = 9/5

Real World Analogy

- Here our assumptions are
  - There is no time consumed to transition car from one working station on other
  - All the stages consume equal time
- However, these assumption are not realistic
  - Each stage can take different amount of time
    - e.g. painting a car takes 2 hours and assembly and installing tires take 0.5 hours each
  - To transfer car from one stage to another will take time.
    - e.g. it takes 10 minutes to transfer a car from assembly room to paint rooms and 10 minute to transfer from paint room to tire fixing room.
    - Note that this time will not be consumed in non-pipelined factory as everything will be done in same room

Real World Analogy

- If Assembly takes 0.5 hour, Paint 2 hours and Tires 0.5 hours
  - Paint of car 2 cannot start before 2.5th hour because Paint unit is busy with car 1.
  - So after Assembly car 2 has to wait for 1.5 hours
- Total time take to complete 3 cars = 7 h
- Speed Up = 9/7

Time	0.5h	2h	2h	2h	0.5h
Car 1	A (0.5h)	P (2h)	T (0.5h)		
Car 2		A (0.5h) + wait for 1.5h	P (2h)	T (0.5h)	
Car 3			A (0.5h) + wait for 1.5h	P (2h)	T (0.5h)

Real World Analogy

- If we add time for transition between stages as well
- Total time taken to complete 3 cars = 7 h 40 m = 7.67 hr
- Speed Up = 9/7.67

Time	0.5h	Transition 10 min	2h	Transition 10 min	2h	Transition 10 min	2h	Transition 10 min	0.5h
Car 1	A (0.5h)		P (2h)		T (0.5h)				
Car 2			A (0.5h) + wait for 1.5h		P (2h)		T (0.5h)		
Car 3					A (0.5h) + wait for 1.5h		P (2h)		T (0.5h)

Cycle Time

- The time required between moving an object one step down the pipeline is a **cycle time**
- Realistically, all stages proceed at the same time
  - There is only one conveyer belt, and it only moves cars to next step when longest stage is complete, i.e. every two hours

Cycle Time

- In the current example, cycle time will be 2 hours
  - Even the car in T stage will be there for 2 hours and will only come out of pipeline when 2 hours of P (of other car) are completed

Time	2 hr	2 hr	2 hr	2 hr	2 hr
Car 1	A	P	**T**		
Car 2		A	**P**	T	
Car 3			**A**	P	T

Ignoring latch (transition) time

Throughput and Latency

- In case of car manufacturing, throughput is number of cars created per unit of time (say, 1 hour)
  - From slide 4, throughput without pipelining is 3 cars in 9 hours, i.e. 0.33 cars/hour
  - From slide 8, throughput with pipelining is 3 cars in 7.67 hours, i.e. 0.4 cars/hour
  - From slide 10, throughput with pipelining is 3 cars in 10 hours, i.e. 0.3 cars/hour
- So, we can see that the throughput with pipelining **can be** higher.

Throughput and Latency

- Latency is time taken to complete a single car
  - From slide 4, latency without pipelining is 3 hours (time consumed to create each car)
  - From slide 8, latency with pipelining is 4.83 hours (time consumed to create car 2 and 3)
  - From slide 10, latency with pipelining is 6 hours (time consumed to create each car)
- Observation: Pipeline **does not improve** latency, it can only improve throughput

Throughput and Latency for n=3

Case	Latency (hr)	Throughput	Speed up
Non pipelined	3	3/9=0.33	NA
Pipelined ideal case (1h each stage)	3	3/5=0.6	9/5
Pipeline with non uniform stages' time (0.5h, 2.5h, 1.5h)	4.5	3/7=0.42	9/7
Pipeline with latch time (10m transition)	4.833	3/7.67=0.39	9/7.67
Pipeline with uniform cycle time restriction (2h each stage)	6	3/10=0.3	9/10

Exercise

- Divide your paint job in 4 stages, 1st coat, 2nd coat, 3rd coat and drying, each phase takes 0.5 hours.
- What will be throughput with these 6 stages for 3 cars? Ignore the latch/transition time.

Time	0.5h	0.5h	0.5h	0.5h	0.5h	0.5h	0.5h	0.5h
Car 1	A	P1	P2	P3	P4	T		
Car 2		A	P1	P2	P3	P4	T	
Car 3			A	P1	P2	P3	P4	T

Observation: Throughput and speedup of pipeline are best if stages are of more or less equal time duration.

Pipeline in Computer

- In a computer pipeline, each step in the pipeline completes a part of an instruction.
- Like the assembly line, different steps are completing different parts of different instructions in parallel.
- Each of these steps is called a pipe stage or a pipe segment.
- The stages are connected one to the next to form a pipe—instructions enter at one end, progress through the stages, and exit at the other end, just as cars would in an assembly line

Pipeline in Computer: Stages

- The number of stages in which one instruction can be divided is different from processor to processor.
- If we only create two stages Fetch Instruction (FI) and Execute Instruction (EI), EI will take a lot more time than FI.
- To get a better throughput and speedup, pipeline must have more stages.
- The pipeline designer’s goal is to balance the length of each pipeline stage

Pipeline in Computer: Stages

- Let us consider the following decomposition of the instruction processing

- **Fetch instruction (FI):** Read the next expected instruction into a buffer.
- **Decode instruction (DI):** Determine the opcode and the operand specifiers.
- **Calculate operands (CO):** Calculate the effective address of each source operand. This may involve displacement, register indirect, indirect, or other forms of address calculation.
- **Fetch operands (FO):** Fetch each operand from memory. Operands in registers need not be fetched.
- **Execute instruction (EI):** Perform the indicated operation and store the result, if any, in the specified destination operand location.
- **Write operand (WO):** Store the result in memory.

Pipeline in Computer: Stages

- Six-stage pipeline can reduce the execution time for 9 instructions from 54 time units to 14 time units.
- Speedup = 54/14
- Note that all instructions might not require going through all these stages
  - e.g. WO is not required in load operations. FO is not required if operand is register

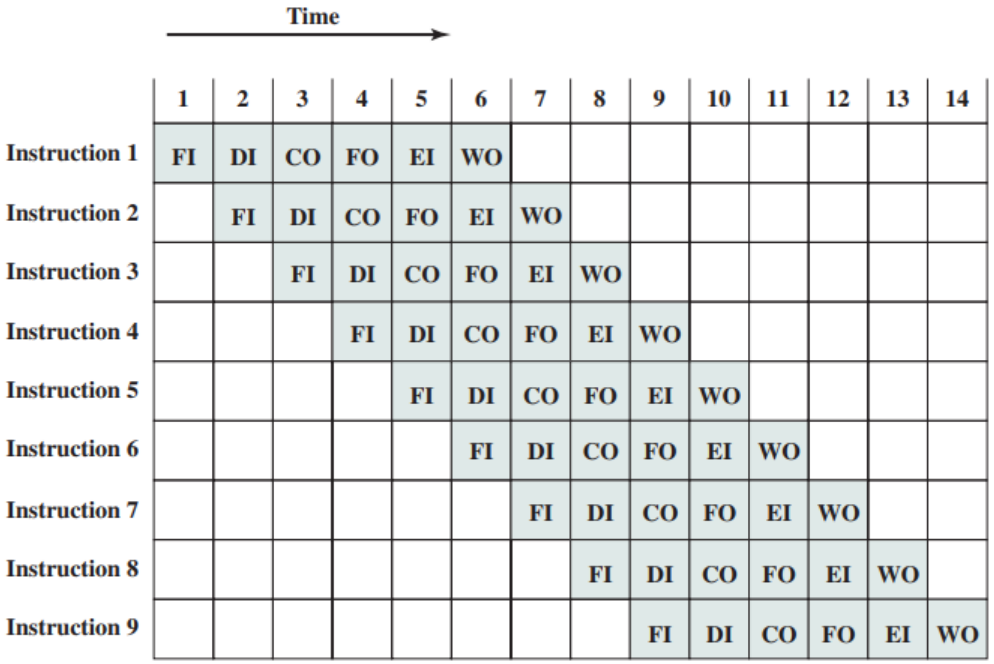


Figure 14.10 Timing Diagram for Instruction Pipeline Operation

Pipeline in Computer: Cycle Time

- The time required between moving an instruction one step down the pipeline is called cycle time
- Because all stages proceed at the same time, the length of a cycle time is determined by the time required for the slowest pipe stage, just as in an auto assembly line
- The longest step would determine the time between advancing the line
- In a computer, this processor cycle is usually 1 clock cycle
  - One cell in figure 14.10 is 1 clock cycle

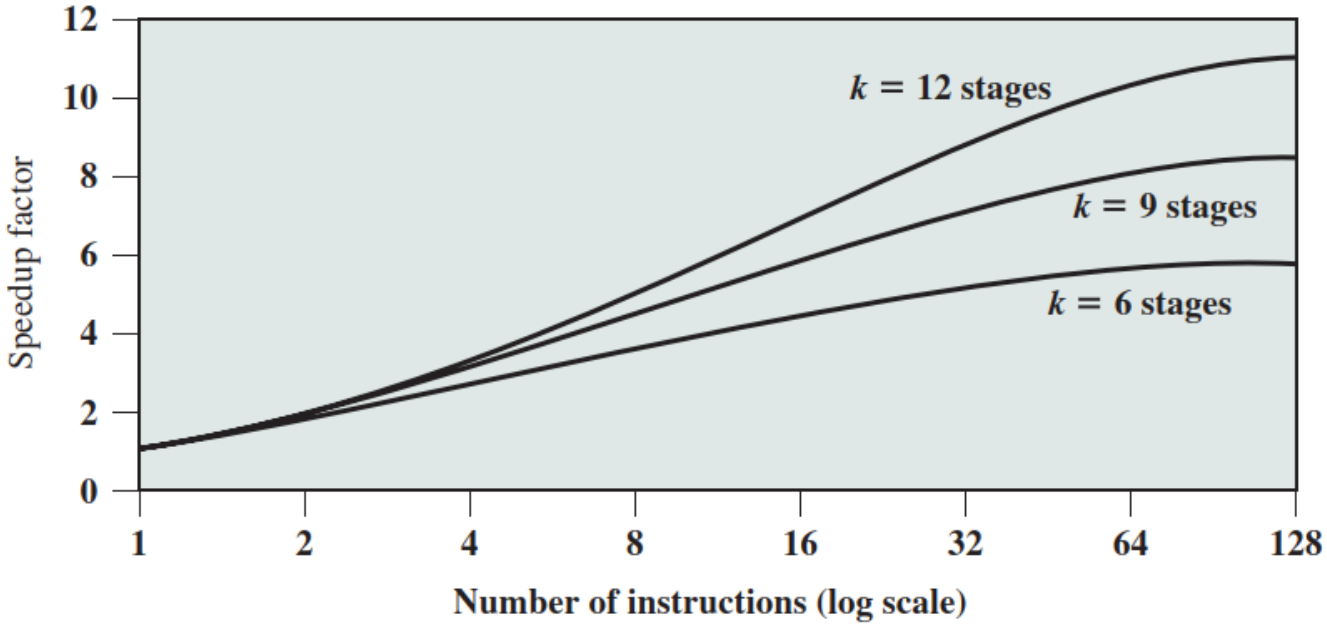
Pipeline in Computer: Cycle Time

- For example, if each stage takes 1 µs
  - Clock cyle should be of 1 µs
  - Although this is the ideal case
- If FI takes 5 µs, DI takes 1 µs, CO takes 1 µs, FO takes 10 µs, EI takes 1 µs, and WO also takes 10 µs
  - Clock cyle should be of 10 µs
- In processor with no pipelining, If one instruction takes 10 µs
  - Clock cycle should 10 µs

Performance of Pipeline

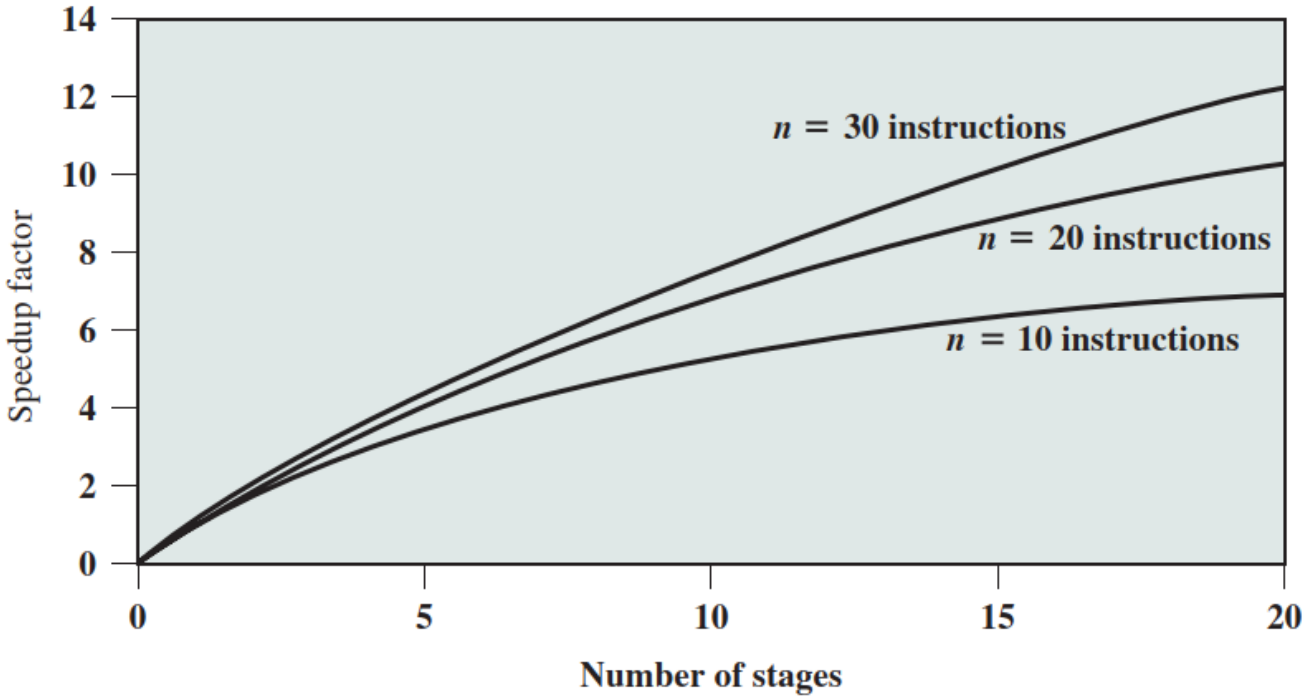
- Assume
  - **Ideal Case:** All stages take equal amount of time T, Latch time = 0
  - Stages = k
  - Number of instructions in program = n
- Without pipeline
  - Time to complete one instruction (i.e. all stages) = kT
  - Time taken to complete n instructions = nkT
  - Throughput for n instructions = $\frac{n}{nkT} = \frac{1}{kT}$
- With pipeline
  - Time taken to complete n instructions = $[k + (n - 1)]\ T$
  - Throughput for n instructions = $\frac{n}{[k+(n-1)]T}$
  - Speedup = $\frac{nkT}{[k+(n-1)]T} = \frac{nk}{k + (n-1)}$

Performance of Ideal Pipeline



As $n \rightarrow \infty$, speed up factor $\rightarrow k$

Performance of Ideal Pipeline



Performance of Pipeline: Example

- Suppose
  - It takes 6 µs to complete one instruction in non-pipelined processor
  - We were able to convert the circuit into 6 equal sequential pipeline stages.
  - Assume latch time is 0
- Answer the following, assuming that there are no stalls in the pipeline.
  - What are the clock cycles in the two processors?
  - How long does it take to finish one instruction in pipelined and non-pipelined processor (latency)?
  - What is the throughput for 100 instructions without pipelining?
  - What is the throughput for 100 instructions with pipelining?
  - What is the speedup from pipelining for 1 instructions?
  - What is the speedup from pipelining for 100 instructions?

Performance: Non-Ideal Case I

- Where all stages do not take same amount of time and ignoring latch time
  - Speedup= $\frac{n\,T_1}{[k+(n-1)]\,T_2}$
  - where n is number of instructions, k is number of stages, T_1 is time required by one instruction to complete in non pipelined processor and T_2 is max time required by any stage in pipelined processor.

Time	T_1	T_1	T_1
Car 1	A+P+T		
Car 2		A+P+T	
Car 3			A+P+T
..			

Non pipelined

Time	T_2	T_2	T_2	T_2	T_2
Car 1	A	P	T		
Car 2		A	P	T	
Car 3			A	P	T
..				…	…

Pipelined

Non-Ideal case I example

- Suppose
  - It takes 6 µs to complete one instruction in non-pipelined processor
  - We were able to convert the circuit into 6 sequential pipeline stages.
  - Stage 1 and 2 take 2 µs each and stage 3-6 take 0.5 µs each
  - Assume latch time is 0
- Answer the following, assuming that there are no stalls in the pipeline.
  - What are the clock cycles in the two processors?
  - How long does it take to finish one instruction in pipelined and non pipelined processor (latency)?
  - What is the throughput for 100 instructions without pipelining?
  - What is the throughput for 100 instructions with pipelining?
  - What is the speedup from pipelining for 1 instructions?
  - What is the speedup from pipelining for 100 instructions?

Performance: Non-Ideal Case II

- Also including latch time
  - Speedup = $\frac{n\,T_1}{[k+(n-1)](T_2+T_3)}$
  - where n is number of instructions, k is number of stages, T_1 is time required by one instruction to complete without pipeline, and T_2 is max time required of all the stages, T_3 is latch (transition) time

Time	T_2	T_3	T_2	T_3	T_2	T_3	T_2	T_3	T_2	T_3
Car 1	A	Latch time	P	Latch time	T	Latch time		Latch time		Latch time
Car 2			A		P		T			
Car 3					A		P		T	
..							…		…	

Non-Ideal case II example

- Suppose
  - It takes 6 µs to complete one instruction in non-pipelined processor
  - We were able to convert the circuit into 6 sequential pipeline stages.
  - Stage 1 and 2 take 2 µs each and stage 3-6 take 0.5 µs each
  - Latch time is 2 µs
- Answer the following, assuming that there are no stalls in the pipeline.
  - What are the clock cycles in the two processors?
  - How long does it take to finish one instruction in pipelined and non pipelined processor (latency)?
  - What is the throughput for 100 instructions without pipelining?
  - What is the throughput for 100 instructions with pipelining?
  - What is the speedup from pipelining for 1 instructions?
  - What is the speedup from pipelining for 100 instructions?

Another example

- Suppose
  - It takes 5 µs to complete one instruction in non-pipelined processor
  - We were able to convert the circuit into 5 equal duration sequential pipeline stages.
  - Latch time is 0.2 µs
- Answer the following, assuming that there are no stalls in the pipeline.
  - What are the clock cycles in the two processors?
  - How long does it take to finish one instruction in pipelined and non pipelined processor (latency)?
  - What is the throughput for 100 instructions without pipelining?
  - What is the throughput for 100 instructions with pipelining?
  - What is the speedup from pipelining for 1 instructions?
  - What is the speedup from pipelining for 100 instructions?

Exercise

Question 1

- If processor 1 completes n instructions in 15 µs and processor 2 completes same n instructions in 10 µs, what speedup is achieved by processor 2 as compared to processor 1?

Question 2

- If latency with pipelining with 5 stages is 6 µs and latch time is 0 what is the clock cycle time?