PIPELINING I

WHAT IS PIPELINING?

Pipelining is an implementation technique whereby multiple instructions are overlapped in execution

Do you know what an instruction is, and how it gets executed?

It takes advantage of parallelism that exists among the actions needed to execute an instruction.

Today, pipelining is the key implementation 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)

Assuming A+P+T take 3 hours

Time line	**0-3 hours**	**3-6 Hours**	**6 -9Hours**
Car 1	A+P+T		
Car 2		A+P+T	
Car 3			A+P+T

REAL WORLD ANALOGY

Automobile manufacturing line with pipelining, in 3 stages

Assemble (A), Paint (P), Fix tires (T)

Assuming each stage takes 1 hour

It takes 5 hours with pipelining and 9hours without pipelining

Speedup = 9/5

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

REAL WORLD ANALOGY

Here our assumptions are:

1. There is no time consumed to transition car from one working station on other
2. All the stages consume equal time

However these assumption are not valid

1. Each stage can take different amount of time
    - For example, painting a car takes 2 hours and Assembly and installing types take 0.5 hours each
2. To transfer car from one stage to another will take time.
    - For example, 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 will not be consume in un-pipeline factory as everything will be done is same room (this is the overhead of using pipelining)

REAL WORLD ANALOGY

Automobile manufacturing line with pipelining, in 3 stages

If Assemble (A), Paint (P), Fix tires (T) take different time

If Assembly take 0.5 hour, Paint 2 hours and Tires 0.5 hours

- Paint of Car2 cannot start before 2.5th hour because Paint unit is busy with Car1. So after Assembly Car 2 has to wait for 1.5 Hours
- Total time take to complete 3 instructions =7 hours

Speed Up = 9/7

Time line	0-0.5h	0.5-2.5h	2.5-4.5h	4.5-6.5 h	6.5-7 h
Car 1	A (0.5 h)	P (2h)	T (0.5h)		
Car 2		A (0.5 h) +wait for 1.5 h	P (2h)	T (0.5h)	
Car 3			A (0.5h)+ wait for 1.5 h	P (2h)	T (0.5h)

REAL WORLD ANALOGY

Automobile manufacturing line with pipelining, in 3 stages

If we add time for transition between stages as well

Total time taken to complete 3 instructions = 7h+40m=7.67h

Speed Up = 9/7.67

Time line	0-0.5h	Transition time 10 min	0.5 +10m-2.5h +10m	Transition time 10 min	2.5h+20m-4.5h+20m	Transition time 10 min	4.5h+30m-6.5 h+30m	Transition time 10 min	6.5h+40m -7 h+40m
Car 1	A (0.5 h)		P (2h)		T (0.5h)				
Car 2			A (0.5 h) +wait for 1.5 h		P (2h)		T (0.5h)		
Car 3					A (0.5h)+ wait for 1.5 h		P (2h)		T (0.5h)

REAL WORLD ANALOGY::THROUGHPUT AND LATENCY

In case of car manufacturing **throughput** is number of cars create per hour

- **From slide 8,** throughput **with** pipelining is 3 Cars is 7.67 Hours ~0.4 Cars/hour
- **From slide 4,** throughput **without** pipelining is 3 Cars is 9Hours ~ 0.33 Cars/hour
- So we can see that the throughput with pipelining is higher.

Latency is time taken to complete one car

- **From slide 8,** latency **with** pipelining is 4.5 hours (4.5 hours consumed to create car 2 and 3)
- **From slide 4,** latency **without** pipelining is 3 hours (3 hours consumed to create each car)
- Thus we can see that latency is worse with pipelining

Observation: Pipeline does not improve latency, it only improves throughput (observe the bottlenecks)

THROUGHPUT AND LATENCY FOR N=3

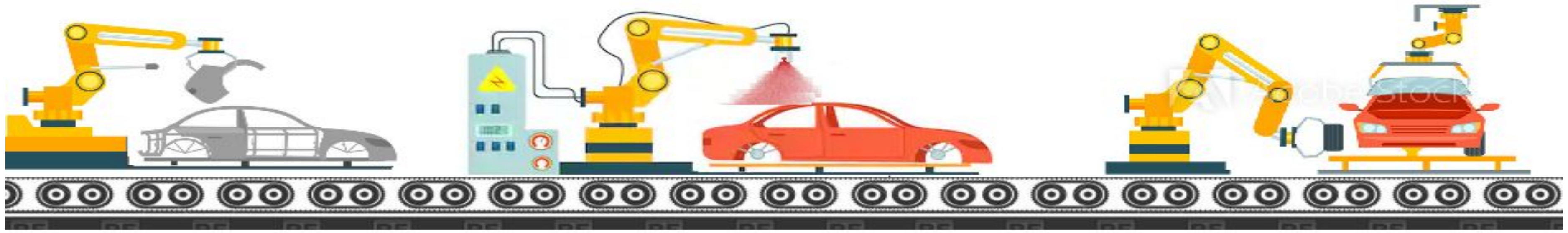
Case	Latency	Throughput	Speed up
Non pipeline	3	3/9=0.33	NA
Pipeline ideal case	3	3/5=0.6	9/5
Pipeline with all stages not of equal time	4.5	3/7=0.42	9/7
Non ideal case with latch time	4.833	3/7.67=0.39	9/7.67

REAL WORLD ANALOGY:: PROCESS CYCLE

The time required between moving an object one step down the pipeline is a **processor cycle**

Assume that all stages procced 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.
- Processor cycle will be 2 hours (See we're trying to define it – think of the computer clock – the smallest unit needs to be followed)
- 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 instruction are completed)



REAL WORLD ANALOGY:: PROCESS CYCLE

Consider process cycle to be 2 hours, throughput with pipelining is 3 Cars/ 10 Hours ~0.3 Cars/hour

- This is even worse than throughput as compared with an un-pipelined version

This issue can be resolve if stages are more or less of equal time duration (paint job can be further divided into smaller stages)

Time line	2 hours	2 hours	**2 hours**	2 hours	2 hours
Car 1	A	P	**T**		
Car 2		A	**P**	T	
Car 3			**A**	P	T

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 4 stages for 3 cars? 1.5 hrs

Ignore the latch/transition time

PIPELINING II

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.

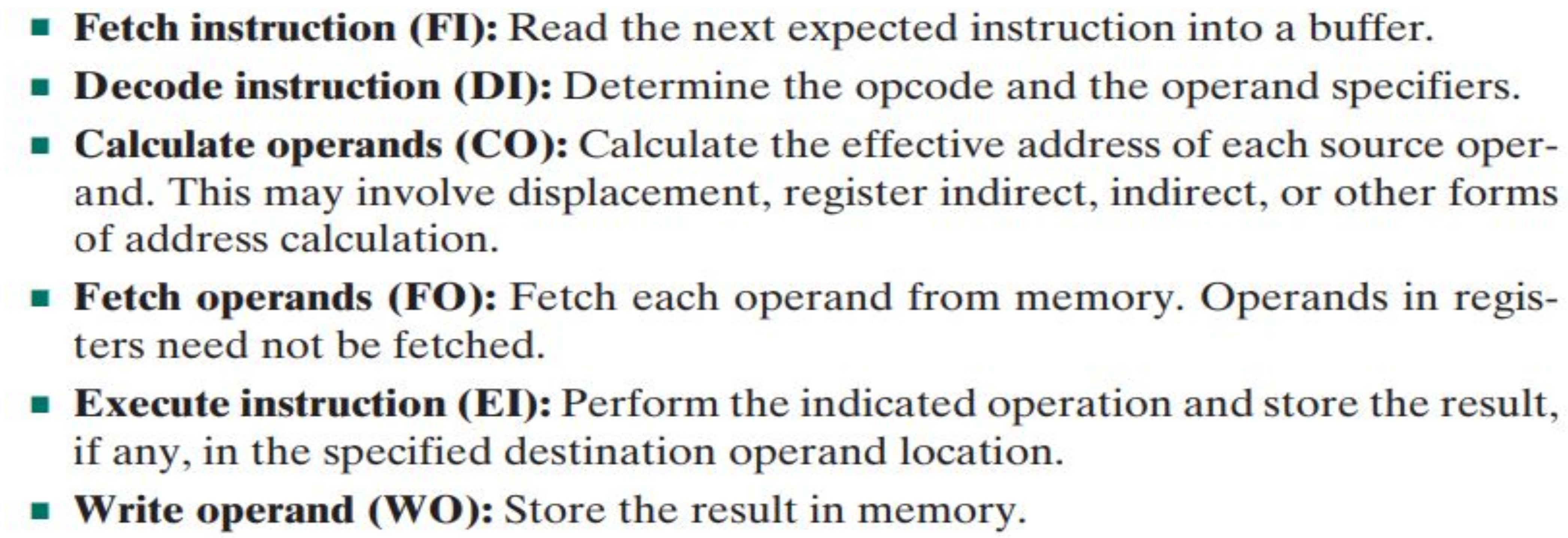
If we only create two stages Fetch Instruction (FI) and Execute Instruction (EI) – *the bottleneck creator*, EI will take a lot more time than FI.

To get a better throughput and thus the 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.

With this decomposition, the various stages will be of more nearly equal duration. For the sake of illustration, let us assume equal duration. Using this assumption, next slide shows that a six-stage pipeline can reduce the execution time for 9 instructions from 54 time units to 14 time units.

PIPELINE IN COMPUTER:: STAGES

Six-stage pipeline can reduce the execution time for 9 instructions from 54 time units to 14 time units.

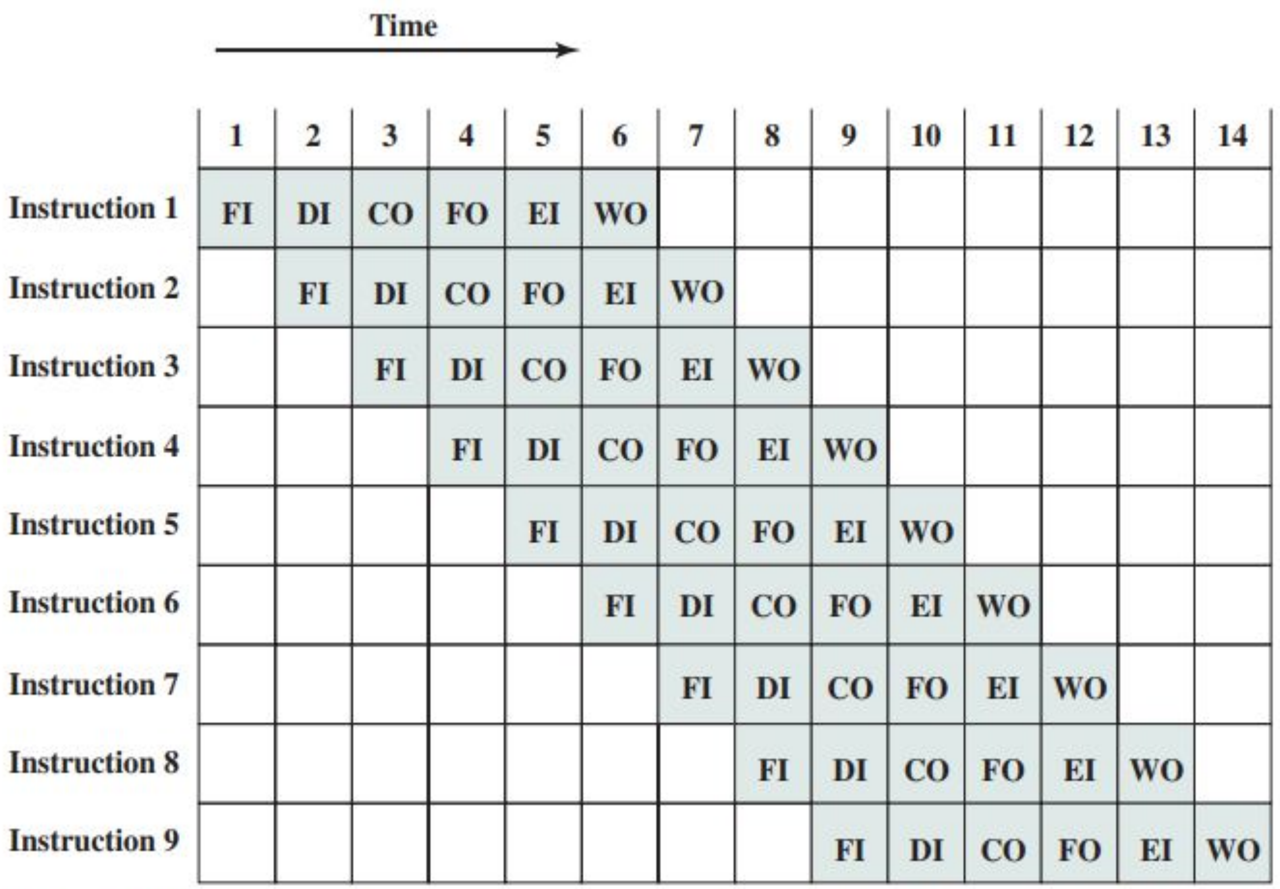
Throughput with pipeline= 9/14

Throughput without pipeline= 9/54

Speedup= 54/14

Note that the figure is for ideal case. In reality, all the instructions might not require to go through all these stages

- For example: in load operations WO is not required
- If operand are register FO is not required



Time →

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

Figure 14.10 Timing Diagram for Instruction Pipeline Operation

PIPELINE IN COMPUTER:: CLOCK CYCLE

The time required between moving an instruction one step down the pipeline is a processor cycle

Because all stages proceed at the same time, the length of a processor cycle 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 (previous slide) is 1 clock cycle

PIPELINE IN COMPUTER:: CLOCK CYCLE

For example in if

- FI DI CO FO EI WO each takes 1 us
- Clock cyle should be of 1us
- Although this is the ideal case

For example:

- FI takes 5 us, DI takes 1us, CO takes 1us, FO takes 10us, EI takes 1us, and WO also takes 10us
- Clock cyle should be of 10us

For example, in processor with no pipelining

- If **one instruction** takes 10us then clock cycle should 10us

PIPELINE IN COMPUTER:: FREQUENCY

Computer processors can execute one or more instructions per **clock cycle**, depending on the type of processor.

Frequency of computer is 1/clock cycle

For example: if 1 clock cycle= 10 ms;

- **Frequency = 1/10ms= 1MHZ**
- **i.e. 1 000 000 cycles / second**

Frequency is AKA as clock speed

PIPELINE IN COMPUTER:: PERFORMANCE

In ideal scenario i.e., all the stages require same amount of time and ignoring time to transfer data from one stage to another

- Speedup= n*k/ k+(n-1)
- where n is number of instructions and k is number of stages

Time line	**T**	**T**	**T**	**T**	**T**
Car 1	A	P	**T**		
Car 2		A	**P**	T	
Car 3			**A**	P	T
..				…	..

PIPELINE IN COMPUTER:: PERFORMANCE IDEAL CASE (DERIVATIONS)

If

- All stages take equal amount of time T
- Latch time = 0
- Stages = k
- Number of instructions = n

Then

- Clock cycle of pipeline = T , clock cycle of non pipeline= k*T
- Frequency of pipeline=1/T , Frequency of non pipeline=1/k*T
- Time taken to complete n instructions without pipeline= n*k*T= n*clock cycle of non pipeline
- Through put for n instructions without pipeline= n/ n*k*T
- Time taken to complete n instructions with pipeline= (k+n-1)*T= (k+n-1)*(Clock Cycle of pipeline)
- Through put for n instructions with pipeline= n/ (k+n-1)*T
- Speedup for n instructions = n*k*T/ (k+n-1)*T = n*k/ (k+n-1)
- Latency without pipelining = k*T
- Latency with pipelining= k*T

PIPELINE IN COMPUTER:: PERFORMANCE IDEAL CASE

Numerical:

- It takes 6us to complete one instruction in non-pipeline 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 cycle in the two processors?

What are the clock speeds(frequency) in two processors?

How long does it take to finish one instruction in pipeline and no pipeline (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?

PIPELINE IN COMPUTER:: PERFORMANCE

NON-IDEAL CASE I

In non ideal case where all stages do not take same amount of time and ignoring latch time

- Speedup= n* T_1 / (k+(n-1))* T_2
- Where n is number of instruction, k is number of stages, T_1 time require by one instruction to complete in non pipeline processor and T_2 is max time require by any stage in pipeline processor

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

Without pipelining

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
..				…	..

With pipelining

PIPELINE IN COMPUTER:: PERFORMANCE NON IDEAL CASE (DERIVATIONS)

If all stages do not take same time

- Non pipeline processor takes T1 time to complete one instruction
- In pipeline processor max time take by any stage is T2
- Latch time=0
- Stages =k
- Number of instructions = n

Then

- Clock cycle of pipeline processor = T2, Clock Cycle of processor without pipelining = T1
- Frequency of pipeline processor =1/T2, Frequency of processor without pipelining = 1/T1
- Time taken to complete n instructions without pipeline= n*T1= n*clock cycle of non pipeline
- Through put for n instructions without pipeline= n/ n*T1
- Time taken to complete n instructions with pipeline= (k+n-1)*T2 =(k+n-1)*(Clock Cycle of pipeline)
- Through put for n instructions with pipeline= n/ (k+n-1)*T2
- Speedup for n instructions = n*T1/ (k+n-1)*T2
- Latency without pipelining = T1
- Latency with pipelining= K*T2

PIPELINE IN COMPUTER:: PERFORMANCE

Example NON-IDEAL

- It takes 6us to complete one instruction in non-pipeline processor
- We were able to convert the circuit into 6 sequential pipeline stages.
- Stage 1 and 2 take 2us each and stage 3-6 take 0.5us each
- Assume latch time is 0
- Answer the following, assuming that there are no stalls in the pipeline.

What are the clock cycle in the two processors?

What are the clock speeds(frequency) in two processors?

How long does it take to finish one instruction in pipeline and no pipeline (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?

PIPELINE IN COMPUTER:: PERFORMANCE

Including latch time

- Speedup= n* T_1 / (k+(n-1))* (T_2+ T3)
- Where n is number of instruction, k is number of stages, T_1 time require by one instruction to complete without pipeline, and T_2 is max time require of all the stages, T3 is latch time (same as transfer time as explained in slide 8)

Time	**T_2**	**Latch time T3**	**T_2**	**Latch time T3**	**T_2**	**Latch time T3**	**T_2**	**Latch time T3**	**T_2**	**Latch time T3**
Car 1	A		P		**T**					
Car 2			A		**P**		T			
Car 3					**A**		P		T	
..							…		..	

PIPELINE IN COMPUTER:: PERFORMANCE NON IDEAL CASE WITH LATCH TIME (DERIVATIONS)

If all stages do not take same time

- Non pipeline processor takes T1 time to complete one instruction
- In pipeline processor max time take by any stage is T2
- Latch time=T3
- Stages =k
- Number of instructions = n

Then

- Clock cycle of pipeline processor = T2+T3, Clock Cycle of processor without pipelining = T1
- Frequency of pipeline processor =1/(T2+T3), Frequency of processor without pipelining = 1/(T1)
- Time taken to complete n instructions without pipeline= n*T1= n*clock cycle of non pipeline
- Through put for n instructions without pipeline= n/ n*T1
- Time taken to complete n instructions with pipeline= (k+n-1)*(T2+T3)= (k+n-1)*(Clock Cycle of pipeline)
- Through put for n instructions with pipeline= n/ (k+n-1)*(T2+T3)
- Speedup for n instructions = n*T1/ (k+n-1)*(T2+T3)
- Latency without pipelining = T1
- Latency with pipelining= K*(T2+T3)

This slide covers all the formulas that you have seen earlier, for example of latch time 0 then you will get same equations are given on slide 26

PIPELINE IN COMPUTER:: PERFORMANCE

Example NON-IDEAL CASE with latch time

- It takes 6us to complete one instruction in non-pipeline processor
- We were able to convert the circuit into 6 sequential pipeline stages.
- Stage 1 and 2 take 2us each and stage 3-6 take 0.5us each
- Latch time is 2us
- Answer the following, assuming that there are no stalls in the pipeline.

What are the clock cycle in the two processors?

What are the clock speeds(frequency) in two processors?

How long does it take to finish one instr in pipeline and no pipeline (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?

PIPELINE IN COMPUTER:: PERFORMANCE

Another example

- It takes 5us to complete one instruction in non-pipeline processor
- We were able to convert the circuit into 5 equal duration sequential pipeline stages.
- Latch time is 0us
- Answer the following, assuming that there are no stalls in the pipeline.

What are the clock cycle in the two processors?

What are the clock speeds(frequency) in two processors?

How long does it take to finish one instr in pipeline and no pipeline (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?

PIPELINE IN COMPUTER:: PERFORMANCE

Another example

- It takes 5us to complete one instruction in non-pipeline processor
- We were able to convert the circuit into 5 equal duration sequential pipeline stages.
- Latch time is 0.2us
- Answer the following, assuming that there are no stalls in the pipeline.

What are the clock cycle in the two processors?

What are the clock speeds(frequency) in two processors?

How long does it take to finish one instr in pipeline and no pipeline (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 throughput of one processor1 for n instruction is 10us and through put of processor 2 is 15us for same n instruction what speed up is achieved by processor2 are compare to processor 1?

Question 2: If Latency with pipelining with 5 stages is 6us and latch time is 0 what is the clock cycle time?