American International University-Bangladesh (AIUB)
Faculty of Engineering

SOME COMPUTATIONAL EXAMPLES ON PIPELINE BASED PROCESSING UNIT

QUESTION 1: PIPELINING

The 5 stages of the processor have the following latencies for two instructions 'a' and 'b':

	Fetch	Decode	Execute	Memory	Write back
a.	300ps	400ps	350ps	500ps	100ps
b.	200ps	150ps	120ps	190ps	140ps

Assume that when pipelining, each pipeline stage costs 20ps extra for the registers be- tween pipeline stages.

1. Non-pipelined processor: what is the cycle time? What is the latency of an instruction? What is the throughput?

   Cycle-time: there is no pipelining, so the cycle-time has to allow an instruction to go through all the stages each cycle. Therefore:

   a. CT = 1650ps

   b. CT = 800ps

   The latency for an instruction is also the same, since each instruction takes 1 cycle to go from beginning fetch to the end of write-back. The throughput is similar as $\frac{1}{Cycle\ Time}$ instructions per seconds.

2. Pipelined processor: What is the cycle time? What is the latency of an instruction? What is the throughput?

   Pipelining to 5 stages reduces the cycle time to the length of the longest stage. Additionally, the cycle time needs to be slightly longer to accommodate the register at the end of the stage.

   a. CT = 520ps (500ps+20ps, time taken for inter between pipeline stages)
   b. CT = 220ps (200ps+20ps, time taken for inter between pipeline stages)

The latency for both is $5 * (cycle\ \ time)$, since an instruction needs to go through 5 pipeline stages, spending 1 cycle in each, before it commits. The throughput can still be calculated as, $\frac{1}{Cycle\ Time}$. Throughput increases because the cycle time reduces.

QUESTION 2: PIPELINING

Assume the distribution of instructions that run on the processor is:

- 50% ALU
- 25%: BEQ
- 15%: LW
- 10%: SW

Assuming there are no stalls or hazards, what is the utilization of the data memory? What is the utilization of the register block's write port? (Utilization in percentage of clock cycles used)

Data memory is utilized only by LW and SW instructions in the MIPS ISA. So the utilization is 25% of the clock cycles.

The write port may be utilized by ALU and LW instructions. The utilization is 65% of the clock cycles.

Calculating utilization in terms of *% time* though gives different results, because unless the corresponding stage (data memory or write-back) is the stage dictating cycle time, the circuit is idle for some period of the clock cycle in *all* cycles.

QUESTION 3: STALLING

Sequence of instructions:

1. lw $s2, 0($s1)
2. lw $s1, 40($s6)
3. sub $s6, $s1, $s2
4. add $s6, $s2, $s2
5. or $s3, $s6, $zero
6. sw $s6, 50($s1)

1. Data dependencies: A data-dependency is a dependence of one instruction B on another instruction A because the value produced by A is read by B. So when listing data dependencies, you need to mention A, B, and the location (in this case, the register that causes the dependence). It also doesn't matter how far apart dependencies are. Don't confuse data dependencies and hazards!

   Dependencies:

   3 depends on 1 ($s2)
   3 depends on 2 ($s1)

4 depends on 1 ($s2)
5 depends on 4 ($s6)
6 depends on 2 ($s1)
6 depends on 4 ($s6)

Assume the 5-stage MIPS pipeline with no forwarding and each stage takes 1 cycle. Instead of inserting NOPs, you let the processor stall on hazards. How many times does the processor stall? How long is each stall (in cycles)? What is the execution time (in cycles) for the whole program?

Ignoring the stalls for a moment, the program takes 10 cycles. Remember, pipelines allow multiple instructions to be executing at the same time.

if	id	ex	mem	wb					
	if	id	Ex	mem	wb				
		if	Id	ex		wb			
			If	id	ex		wb		
				if	id	ex		wb	
					if	id	ex	mem	

With the stalls, there are only two stalls – after the 2nd load, and after the add – both are because the next instruction needs the value being produced. Without forwarding, this means the next instruction is going to be stuck in the fetch stage until the previous instruction writes back. These are 2 cycle stalls. So to answer the question, 2 stalls, 2 cycles each, and the total is $10 + 2 * 2 = 14$ cycles to execute the program.

if	id	ex	mem	wb							
	if	id	Ex	mem	wb						
		if	Id	stall	stall	ex		wb			
			If	id	ex		wb				
				if	id	stall	stall	ex		wb	
					if	id	ex	mem			

2. Assume the 5-stage MIPS pipeline with full forwarding. Write the program with nops to eliminate the hazards. (Hint: time travel is not possible!)

if	id	ex	mem	wb							
	if	id	Ex	mem	wb						
		if	Id	stall	ex		wb				
			If	id	ex		wb				
				if	id	ex		wb			
					if	id	ex	mem			

Again, draw a diagram – in the second stall, the result of the add is available at the register at the end of the execute stage when the next instruction wants to move to the execute stage, so you can forward the value of $s6 as the input to the execution stage (as the argument for the or) – this removes the second 2-cycle stall (3-cycle stall in total). However, the loaded value is not ready until the end of the memory stage, so you cannot using forwarding to remove both cycles – you still need to wait 1 cycle. The solution is to place a NOP after the second load. (Note: this is also the reason why MIPS has load delay slots).

QUESTION 4: MORE PIPELINES

You are given a non-pipelined processor design which has a cycle time of 10ns and average CPI of 1.4. Calculate the latency speedup in the following questions.

1. What is the best speedup you can get by pipelining it into 5 stages?

In the best case, pipelining will reduce CT to 2ns considering CPI to 1.

Speed up = $\frac{CT_{old}}{CT_{new}} = \frac{10\ ns}{2\ ns}$ = 5 times.

2. If the 5 stages are 1ns, 1.5ns, 4ns, 3ns, and 0.5ns, what is the best speedup you can get compared to the original processor?

The cycle time is limited by the slowest stage, so CT = 4 ns.

Speed up = $\frac{CT_{old}}{CT_{new}} = \frac{10\ ns}{4\ ns}$ = 2.5 times

3. If each pipeline stage added also adds 20ps due to register setup delay, what is the best speedup you can get compared to the original processor?

Adding register delay to the cycle time because of pipeline registers, you get CT = 4.02 ns.

Speed up = $\frac{CT_{old}}{CT_{new}} = \frac{10\ ns}{4.02\ ns}$ = 2.49 times

4. The pipeline from Q4.3 stalling adds 20% of the total execution time that leads CPI increment to 1 cycle and another 5% of the total execution time that leads CPI increment to 1 cycle 2 cycles. What is the new CPI? What is the speedup compared to the original processor?

For 20% of the total execution time, CPI increases to $1.4 + 1 = 2.4$

For another 5% of the total execution time, CPI increases to $1.4 + 2 = 3.4$

And the rest 75% of the total execution time, CPI remains same 1.4

New average CPI: $2.4 * 0.2 + 3.4 * 0.05 + 1.4 * 0.75 = 1.7$

Speed up = $\frac{CT_{old}}{CT_{new}} \times \frac{CPI_{old}}{CPI_{new}} = \frac{10\ ns \times 1.4}{4.02\ ns \times 1.7}$ = 2.049 times

QUESTION 5: CPI (CYCLE PER INSTRUCTION)

Cycle Per Instruction (CPI): Cycle Per Instruction is defined how many cycles are required to complete one instruction. In non-pipelined processing unit, CPI defined by each instruction type. Mathematically, CPI is calculated as:

$$\text{CPI} = \frac{no.\#\ of\ total\ cycles\ per\ instruction}{\text{no.\# of stages take active participation } per\ instruction}$$

No.	Instruction
1.	lw $s2, 0($s1)
2.	lw $s1, 40($s6)
3.	sub $s6, $s1, $s2
4.	add $s6, $s2, $s2
5.	or $s3, $s6, $zero
6.	sw $s6, 50($s1)

If every stage requires one (1) cycle to complete, thereby, each instruction's CPI will be 1 for non-pipelined processing unit.

E.X.

For instruction 1, $CPI = \frac{5}{5} = 1$

For instruction 5, $CPI = \frac{4}{4} = 1$

No.	Instruction	Stages Overview									
1.	lw $s2, 0($s1)	if	id	ex	mem	wb					
2.	lw $s1, 40($s6)		if	id	Ex	mem	wb				
3.	sub $s6, $s1, $s2			if	Id	ex	wb				
4.	add $s6, $s2, $s2				If	id	ex	wb			
5.	or $s3, $s6, $zero					if	id	ex	wb		
6.	sw $s6, 50($s1)						if	id	ex	mem	

If every stage requires one (1) cycle to complete, thereby, CPI will be 1 again for pipelined processing unit without taking an account of hazard. Be remembered: Pipelined Processing Unit takes account only the instruction in a program that is slowest. [In above program, '**lw**' will account]

Now, if the hazards are taken in consideration, then the pipeline processing is changing as follows:

No.	Instruction	Stages Overview									
1.	lw $s2, 0($s1)	if	id	ex	mem	wb					
2.	lw $s1, 40($s6)		if	id	Ex	mem	wb				
3.	sub $s6, $s1, $s2			if	Id	stall	stall	ex	wb		
4.	add $s6, $s2, $s2				If	id	ex	wb			
5.	or $s3, $s6, $zero					if	id	stall	ex	wb	
6.	sw $s6, 50($s1)						if	id	ex	mem	

CPI of the program will be counted with the slowest instruction like '3'. And, CPI becomes 1.4 (7 cycle/5 stages).