Assignment-2: Pipelining

Arnab Das, u1014840

February 13, 2018

1 *Pipelining Performance*

i.

In a single cycle processor, with IPC/CPI = 1, the CPU time will be

$$CPU\_time = IC \times CT = 1000 \times 10ns = 10\mu s$$

ii.

We denote the 1ns aditional delay for pipeline registers as

$$T_{ovh} = 1ns$$

Let T denote the cycle time for a single-cycle processor. In this case, $T = 10ns$. Let T_{ovh} denote the additional delay at every pipeline stage. Hence, $T_{ovh} = 1ns$. The cycle time for the unpipelined case will be

$$CT_{unpipe} = T + T_{ovh}$$

and the cycle time for the 10-stage pipelined case will be

$$CT_{pipe} = \frac{T}{10} + T_{ovh}$$

For the unpipelined case, it flushes 1 instruction per cycle since no stalls are required due to absence of hazards. For the 10-stage pipelined case, every 10 cycles req1uires 2 additional stall cycles. Thus it flushes 10 instructions every 12 cycles. Hence the CPI for the pipelined case will be

$$CPI_{pipe} = \frac{\#cycles}{\#instr} = \frac{12}{10} = 1.2$$

Since the instrcution count in both cases is the same, the execution times will be

$$\begin{aligned} Exec\_Time_{unpipe} &= IC_{unpipe} \times CPI_{unpipe} \times CT_{unpipe} = 1000 \cdot 1 \cdot (T + T_{ovh}) \\ Exec\_Time_{pipe} &= IC_{pipe} \times CPI_{pipe} \times CT_{pipe} = 1000 \cdot 1.2 \cdot (\frac{T}{10} + T_{ovh}) \end{aligned}$$

Thus, we get the following speedup

$$\begin{aligned} Speedup &= \frac{Exec\_Time_{unpipe}}{Exec\_Time_{pipe}} \\ &= \frac{1000 \cdot 1 \cdot (T + T_{ovh})}{1000 \cdot 1.2 \cdot (\frac{T}{10} + T_{ovh})} \\ &= \frac{11}{1.2 \cdot (1+1)} = \mathbf{4.58} \end{aligned}$$

2 *Control Hazards*

Let the total number of intructions be x. Given that only branch instructions introduce stalls. The total number of branch instructions
,

$$\#total\ branches = 20\%\ of\ x = \frac{x}{5}$$

Each branch introduces 2 additional stall cycles. Given that in the case of a branch being taken, it is allowed to move two instructions from the taken side into the branch delay slot, wich implies that in case of 60% of the branches being taken some useful work is being done in the stall cycles and hence there is effectively no stall cycles in the *taken* case. The remaining 40% of the branches which results in branches not being taken consequently leads to 2 stall cycles each. Number of branch instructions not taken will be

$$\#\ \text{branches not taken} = 40\%\ of\ \frac{x}{5} = \frac{2x}{25}$$

and total number of stalls introduced will be

$$\#\ total\ stalls = 2 \times \#\ \text{branches not taken} = \frac{4x}{25}$$

Hence, total number of cycles

$$\#\ total\ cycles = \#\ useful\ cycles + \#\ stall\ cycles = x + \frac{4x}{25} = \frac{29x}{25}$$

Hence, the expected CPI will be

$$CPI_{expected} = \frac{\#\ total\ cycles}{\#\ total\ instrs} = \frac{\frac{29x}{25}}{x} = \mathbf{1.16}$$

3 *Multi-cycle Intructions*

The following assumptions are made while developing the timing diagram for the given sequence of instructions.

- Separate instruction and data memory exists, that is, an instruction fetch and a data memory access can happen in the same cycle.
- The register file has two ports. One port is exclusively for read and the other port is exclusively for write. Moreover, data is written to the register file in the first half of the clock and read in the second half of the clock. This implies that a write back and read of the same register location can happen in the same cycle without hazard.
- The pipeline implements all necessary forwarding for the ALU operations.

Stalls for hazards
Stalls to avoid imprecise exception(making in order write backs)

Figure 1: Timing Diagram

Out of order WB with ALU forwarding

	CLOCK CYCLES																
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Load F6, 20(R5)	IF	IS	RR	EX	WB,F6												
Load F2, 28(R5)		IF	IS	RR	EX	WB,F2											
MUL F0, F2, F4			IF	IS		RR	EX	EX	EX	WB,F0							
SUB F8, F6, F3				IF		IS	RR	EX	WB,F8								
DIV F10, F0, F6						IF	IS		RR	EX	EX	EX	EX	WB,F10			
ADD F6, F8, F2							IF		IS	RR	EX	WB,F6					
STORE F8, 50(R5)									IF	IS	RR	EX	WB				

In order WB with ALU forwarding

	CLOCK CYCLES																
Instruction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Load F6, 20(R5)	IF	IS	RR	EX	WB,F6												
Load F2, 28(R5)		IF	IS	RR	EX	WB,F2											
MUL F0, F2, F4			IF	IS		RR	EX	EX	EX	WB,F0							
SUB F8, F6, F3				IF		IS	RR	EX			WB,F8						
DIV F10, F0, F6						IF	IS		RR	EX	EX	EX	EX	WB,F10			
ADD F6, F8, F2							IF		IS	RR	EX				WB,F6		
STORE F8, 50(R5)									IF	IS	RR	EX				WB	

Figure: 1 shows the timing diagram for the given sequence of instructions for both out of order write backs and inorder write backs. The first stall we see is in cycle 5 for MUL instruction. Register read for MUL requires updated **F2**, which is not available in the register file until cycle 6. If register read was issued for MUL in cycle 5, it would cause **data hazard**, since it would read the stale value of $F2$. Note that the stall in cycle 5 for MUL instruction means it is still in the IS stage of the pipeline. Hence, the IF stage of SUB instruction also gets stalled for one more cycle.

For the DIV instruction, its register read includes register $F0$, which is an output of MUL instruction. The register read for DIV in cycle 9 may read the stale value of $F0$ because $F0$ actually becomes available in the pipeline registers at the end cycle 9, and written back to register file in cycle 10. However, since we start the execution of DIV in cycle 10, the available forwarding paths for ALU ops allows the generated $F0$ value to be made available to the execution stage in cycle 10, thus avoiding data hazard and also avoiding a stall.

The second table shows the in order write backs to prevent imprecise exceptions.

Structural and Data Hazards

- **Structural Hazard**
  - If we consider unified memory that is data and instruction reside in the same memory, then we have structural hazards at cycle 4, which involves a instruction fetch and a load memory in the same cycle.

  - Further we delay the IS stage for SUB(similarly for ADD) by 1 cycle else it would have been competing for IS stage of MUL causing structural hazard.

- **Data Hazard**
  - RAW: Reading $F6$ in SUB instruction after writing to $F6$ in the LOAD instruction.
  - RAW: Reading $F6$ in DIV instruction after writing to $F6$ in the LOAD instruction.
  - RAW: Reading $F2$ in MUL instruction after writing to $F2$ in the LOAD instruction.
  - RAW: Reading $F2$ in ADD instruction after writing to $F2$ in the LOAD instruction.
  - RAW: Reading $F0$ in DIV instruction after writing to $F0$ in the MUL instruction.
  - RAW: Reading $F8$ in STORE instruction after writing to $F8$ in the SUB instruction.
  - WAW: Writing to $F6$ in ADD instruction after writing to $F6$ in the LOAD instruction.
  - WAR: No Hazards.
  - RAR: No Hazards.

4 *Points of Production and Consumption*

In the unpipelined processor, it is given that the total circuit delay to execute an instruction is $T = 36ns$, with a latch overhead of $T_{ovh} = 0.5ns$. In the unpipelined case, the distance between POC and POP does not matters since the cycle time includes the duration spent in the entire circuit. Hence, the unpipelined cycle time will be

$$CT_{unpipe} = T + T_{ovh} = 36.5ns$$

and its throughput(IPS) will be

$$IPS_{unpipe} = \frac{1}{36.5 \times 10^{-9}} = \mathbf{27.39 \times 10^6}$$

In case of a 12 stage pipelined processor, the time spent per stage in the combinational circuit(excluding the latching stage) is

$$T' = T/12 = 3ns$$

Thus the 12 ns delay between POP and POC in the unpipelined case maps to 4 cycles in the 12-stage pipelined scenario. Thus the

stage of POP is followed by three stall cycles followed by the stage of POC in the pipelined version. It is given that half the instructions do not introduce data hazard and half the instructions depend on their preceeding instruction. This means each instruction that is a producer is followed by 3 stall cycles followed by a consumer. Thus every 5 cycles it consumes 2 instructions. Hence the IPC will be

$$IPC = \frac{2}{5} = 0.4$$

The cycle time in the 12 stage pipeline will be

$$CT_{pipe} = T/12 + T_{ovh} = 3ns + 0.5ns = 3.5ns$$

Hence the throughput(IPS) will be

$$IPS = \frac{IPC}{CT_{pipe}} = \frac{0.4}{3.5ns} = \mathbf{114.28 \times 10^6}$$