ECE20021 / ITP30003

Homework #2

Due: November 18, 2020

Please submit your file in LMS. Your file should be PDF file. In this homework, you should provide reasonable explanations for your answers.

Q1. Consider three different processors P1, P2, and P3 executing the same instruction set with the clock rates and CPIs given in the following table.

Processor	Clock rate	CPI
P1	2 GHz	1.5
P2	1.5 GHz	1.0
P3	3 GHz	2.5

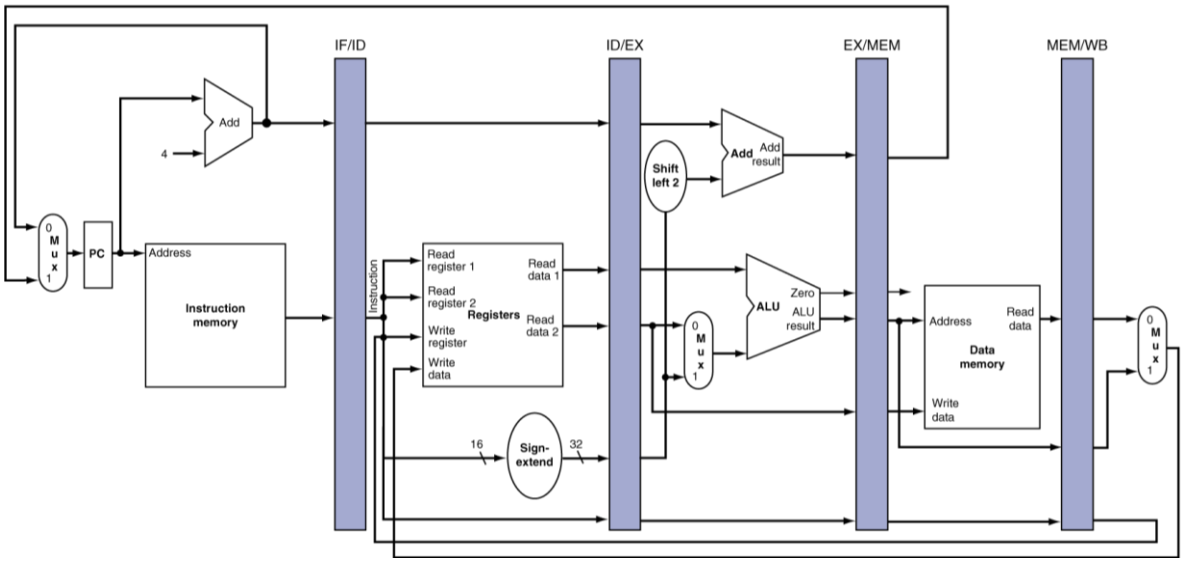
1) Which processor has the highest performance expressed in instructions per second? (5pts)
2) If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions. (5pts)
3) We are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reductions? (5pts)

Q2. Consider two different implementations of the same instruction set architecture. There are four classes of instructions, A, B, C, and D. The clock rate and CPI of each implementation are given in the following table.

Processor	Clock rate	CPI Class A	CPI Class B	CPI Class C	CPI Class D
P1	1.5 GHz	1	2	3	4
P2	2 GHz	2	2	2	2

1) Given a program with 10^6 instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, and 20% class D, which is faster: P1 or P2? (5pts)
2) What is the global CPI for each implementation? (5pts)
3) Find the clock cycles required in both cases. (5pts)

Q3. This exercise is intended to help you understand the cost/complexity/performance tradeoffs of forwarding in a pipelined processor. Below problems refer to pipelined datapaths as shown below. These problems assume that, of all instructions executed in a processor, the following fraction of these instructions have a particular type of RAW (Read After Write) data dependence. The type of RAW data dependence is identified by the stage that produces the result (EX or MEM) and the instruction that consumes the result (1st instruction that follows the one that produces the result, 2nd instruction that follows, or both). We assume that the register write is done in the first half of the clock cycle and that register reads are done in the second half of the cycle, so "EX to 3rd" and "MEM to 3rd" dependences are not counted because they cannot result in data hazards. Also, assume that the CPI of the processor is 1 if there are no data hazards. (RAW dependence



EX to 1st only	MEM to 1st only	EX to 2nd only	MEM to 2nd only	EX to 1st and MEM to 2nd	Other RAW Dependences
5%	20%	5%	10%	10%	10%

Assume the following latencies for individual pipeline stages. For the EX stage, latencies are given separately for a processor without forwarding and for a processor with different kinds of forwarding.

IF	ID	EX (no FW)	EX (full FW)	EX (FW from EX/MEM only)	EX (FW from MEM/WB only)	MEM	WB
150ps	100ps	120ps	150ps	140ps	130ps	120ps	100ps

1) If we use no forwarding, what fraction of cycles are we stalling due to data hazards? (5pts)
(Hint: Compute CPI considering the stall cycle. And then compute the fraction of cycles for stalling, i.e., CPI for stalling only / Total CPI with stalling)

2) If we use full forwarding (forward all results that can be forwarded), what fraction of cycles are we stalling due to data hazards? (5pts)
(Hint: Hint is similar with Q3-1)

3) Let us assume that we cannot afford to have three-input Muxes that are needed for full forwarding. We have to decide if it is better to forward only from the EX/MEM pipeline register (next-cycle forwarding) or only from the MEM/WB pipeline register (two-cycle forwarding). Which of the two options results

in fewer data stall cycles? (5pts)
(Hint: Compare the probabilities of two cases with given hazard probabilities according to pipeline stage.)

4) For the given hazard probabilities and pipeline stage latencies, what is the speed-up achieved by adding full forwarding to a pipeline that had no forwarding? (5pts)
(Hint: Consider clock cycle time with the CPIs already computed in the above problems, Q3-1 and Q3-2.)

5) What would be the additional speed-up (relative to a processor with forwarding) if we added time-travel forwarding that eliminates all data hazards? Assume that the yet-to-be-invented time-travel circuitry adds l00 ps to the latency of the full-forwarding EX stage. (5pts)
(Hint: Time-travel forwarding means that forwarding backward in time. Of course we cannot implement it. This problem comes from an imagination. Please refer to p.12 and p.58 in pdf file of part 2 of chapter 4.)

Q4. The importance of having a good branch predictor depends on how often conditional branches are executed. Together with branch predictor accuracy, this will determine how much time is spent stalling due to mispredicted branches. In this exercise, assume that the breakdown of dynamic instructions into various instruction categories is as follows:

R-Type	beq (branch on equal)	j (jump)	lw (load word)	sw (store word)
40%	25%	5%	25%	5%

Also, assume the following branch predictor accuracies:

Always-Taken	Always-Not-Taken	2-Bit
45%	55%	85%

1) Stall cycles due to mispredicted branches increase the CPI. What is the extra CPI due to mispredicted branches with the always-taken predictor? Assume that branch outcomes are determined in the EX stage, that there are no data hazards. (5pts)

2) Repeat Q4-1 for the “always-not-taken” predictor. (5pts)

3) Repeat Q4-1 for the “2-bit predictor”. (5pts)

4) With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instruction in a way that replaces a branch instruction with an ALU instruction? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced. (5pts)

5) With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instructions in a way that replaced each branch instruction with two ALU instructions? Assume that correctly and incorrectly predicted instructions have the same chance of being replaced. (5pts)

Q5. This exercise explores how exception handling affects pipeline design. The first three problems in this exercise refer to the following two instructions:

Instruction 1	Instruction 2
bne $1, $2, Label	lw $2, 40($3)

1) Which exceptions can each of these instructions trigger? For each of these exceptions, specify the pipeline stage in which it is detected. (5pts)
   (Hint: Possible exceptions in MIPS are overflow, invalid data address, undefined instruction, invalid target address, and hardware malfunction. Which exceptions can each of these instructions trigger?)

2) If there is a separate handler address for each exception, show how the pipeline organization must be changed to be able to handle this exception. You can assume that the addresses of these handlers are known when the processor is designed. (5pts)
   (You don't need to draw the datapath. It is okay to just describe the design directions)

3) If the second instruction in the table is fetched right after the first instruction, describe what happens in the pipeline when the first instruction causes the first exception you listed in Q5-1. Show the pipeline execution diagram from the time the first instruction is fetched until the time the first instruction of the exception handler is completed. (5pts)

4) In vectored exception handling, the table of exception handler addresses is in data memory at a known (fixed) address. Change the pipeline to implement this exception handling mechanism. Repeat Q5-3 using this modified pipeline and vectored exception handling. (5pts)
   (Hint: We need to fetch the address of the exception handler from memory. For the table of exception handler address, table below is one of example. You don't need to draw the datapath. It is okay to just describe the design directions)

Overflow	Invalid data address	Undefined instruction	Invalid instruction address	Hardware malfunction
0xFFFFF000	0xFFFFF100	0xFFFFF200	0xFFFFF300	0xFFFFF400