Yousef Jarrar

CSE 401 – Homework 4

Dr. Gomez

4.8

In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the Datapath have the following latencies:

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

Also, assume that instructions executed by the processor are broken down as follows:

alu	beq	lw	sw
45%	20%	20%	15%

1) For a pipelined processor cycle time is always the latency time by slowest processor. What is the clock cycle time in a pipelined and non-pipelined processor?

Instruction Class	Instruction Decode	Instruction Fetch	Execution	Register write	Memory	Total
Store word	350	250	150		300	1050ps
R-Type	350	250	150	200		950ps
Branch	350	250	150			750ps
Load Word	350	250	150	200	300	1250ps

In the above table it can be easily determined that slowest processor stage is 350ps. This latency is for Instruction decode. Hence, clock cycle time for pipelined processor is 350 ps (picoseconds)

For a non-pipelined processor cycle time can be determined by adding the sum for all stages. Therefore, the sum of all stages is:

350 + 250 +150 + 200 + 300 = 1250

Hence, for non-pipelined processor cycle time will be 1250 ps (picoseconds)

2) What is the total latency of an LW instruction in a pipelined and non-pipelined processor?

The instruction lw uses the all above five stages. If there is no pipelining, then the cycle time has to allow the instruction to pass between the all stages for each cycle. So the total latency for

the lw instruction would be same as the total cycle time for no pipelining or non-pipelined processor as:

250 + 350 + 150 + 300 + 200
= 1250

The instruction lw uses the all above five stages. Pipelining to the all five stage seduce the cycle time to the length of the longest stage. The longest stage is ID with 350ps cycle time.
Therefore **the latency of an lw instruction** for pipelined processor can be 5 cycles at the 350ps per cycle.

It can be calculated as:

5 * 350

= 1750

3) If we can split one stage of the pipelined Datapath into 2 new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle of the processor?

Pipelining to the five stages is reduced the cycle time to the length of the longest stage. Therefore splitting the longest stage is the most usual way to decrease the cycle time. After splitting at the new cycle time is based upon the new longest stage. Thus, the longest stage ID would be split into two new stages. The new clock cycle time of the processor will be 300 ps.

4) Assuming there are no stalls or hazards, what is the utilization of the data memory

Utilization of data memory = LW + SW
= 20 + 15
= 35%

5) Assuming there are no stalls or hazards, what is the utilization of the write–register port of the (registers) unit?

To write the port of register block by the processor is only utilized by the two instructions that are LW for load and ALU for performance in arithmetic and logical operations. This is what is considered in MIPS instruction set architecture.

So utilization of the register block is:
= ALU + IW
= 45 + 20
= 65%

4.12)

1. If we use no forwarding, what fraction of cycles are we stalling due to the data hazards?

Considering that there are dependencies between the first and next instruction, the number of stalls required for the first instruction is to and the number of stalls required for the second instruction is one.

- ➔ CPI processor is equal to one
- ➔ execution to first only is equal to 5% or .05
- ➔ memory to first only equal to 20% or .2
- ➔ execution to first and memory to second is equal to 10% or .1
- ➔ number of clock cycles is equal to 2
- ➔ execution to second only equal to 5% or .05
- ➔ memory to second only equals to 10% or .1
- ➔ execution to first and memory to second is equal to 10% or .1

substituting these values in the formula below:

derive CPI = initial CPI + (EX to 1^{st} only + MEM to first only + EX to 1^{st} and mem to 2^{nd}) * no. Of cycles + (EX to 2^{nd} + MEM to 2^{nd} only + EX to 1^{st} and mem to 2^{nd}) * no. Of cycles

= 1 + (0.05 + 0.2 + 0.1) * 2 + (.05 + .10 + .10) * 1

= 1 * 0.35 * 2 + 0.25 * 1

= 1.95

Calculate the CPI using the following method:

CPI = derived CPI - based CPI

= 1.95-1

= 0.95

the fraction of stalls cycles discarded by:

Stalls cycles = CPI ÷ derive CPI

= 0.95÷1.95

= 0.487

therefore, the fraction of stalls cycles are stalled due to data hazards that are 0.487

2) if we useful forwarding what fraction of cycles are, we stalling due to data hazards?

Calculation of fraction of stalls cycles with full forwarding:

the CPI relates to the given formula below

derive CPI = initial CPI + [(EX of 2^{nd} + MEM to 2^{nd} + EX to 1^{st} and MEM to 2^{nd}) × number of stalls]

= 1+ [(0.05+0.1+0.1) ×1]

= 1+0.25

= 1.25

the derived CPI is 1.25

Calculation of the fraction of stalls cycles is shown as below:

stalls cycle = CPI ÷ derive CPI

= 0.25÷ 1.25

= 20%

the stall cycles are 20%

4.15)

stall cycles due to miss predicted branches increase CPI. What is the extra CPI due to miss predicted branches with the always – taken predictor? Assume that branch outcomes are determined in the EX stage, that there are no data hazards, and that no delay slots are used.

To evaluate the extra CPI due to miss predicted branches, the formula used is

= percentage of missed predictions × percentage of BEQ instructions × number of cycles per misprediction

The branch predictor accuracy with always – taken predictor is 45%. Since the branch predictor accuracy is 45% the miss predicted branches with always taken predictor is:

(100-45) % = 55%

0.55

the breakdown of dynamic instructions into BET you categories 25%

in five stage pipelines, the prediction of whether the instruction is a branch instruction or not gets resolved in the EX cycle. Therefore, the control has it on branches contains two stage stalls: IF, ID. This leads to 2 stall cycles

the number of cycles per misprediction equals to 2.

From the equations one, two and three; the extra CPI due to miss predicted branches with the always – taken predictor is calculated by:

= Percentage of mispredictions × percentage of BEQ instructions × number of cycles per misprediction

= 0.55×0.25×2

= 0.275

= percentage of mispredictions for always – taken predictor × percentage of BEQ instructions × number of cycles per misprediction

= 0.55×0.25×2

= 0.275

therefore, the extra CPI due to misprediction branches with the always – taken predictor is 0.275.

2) it is known that there is an increment in the CPI by misplaced branches in stall cycle. By using the always not taken predictor, this predictor doesn't predict a branch properly will cause three stall cycles

it is valid only when it is supposed that EAX stages are used to determine the branch outcomes and there is no delay slots and no data hazards.

From the table given in the question, it is given that the accuracy of the always – not – taken predictor is 55% or .55

therefore, the accuracy to misprediction for always – not – taken predictor is given by:

(1-0.55) = 0.45

therefore, the extra CPI can be calculated by:

stall cycle ×0.25× misprediction for always – not – taken predictor

3×0.25× (0.45)

= 0.35

hence the increment in the CPI by misplaced branches in stall cycle is 0.34

3) it is known that there is an increment in the CPI by misplaced branches in the stall cycle. By using the two-bit predictor, this predictor doesn't predict a branch properly which will cause three stall cycles

From the given table of branch predictor accuracies, it is given that the accuracy of a two bit is 85% or .85

Therefore, the accuracy to misprediction 2-bit predictor is given by:

(1-0.85) = 0.15

therefore, the extra CPI can be calculating by:

stall cycle ×0.25× misprediction for two-bit predictor

= 3 application signed 0.25× (0.15)

= 0.113

the increment in the CPI by misplaced branches in stall cycle is 0.113