Keegan Smith

Computer Architecture and Design

HW 4

11/28/23

1. In this question, we examine how pipelining affects the clock cycle time of the processor. Assume that individual stages of the datapath have the following latencies:

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

Assume also that instructions execute by the process are broken down as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| ALU | BEQ | LW | SW |
| 45% | 20% | 20% | 15% |

1. What is the clock cycle time in a pipelined and non-pipelined processor?

Non-pipelined: 250 + 350 + 150 + 300 + 200 = 1250 ps

Pipelined: 350 ps

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

Non-pipelined: 250 + 350 + 150 + 300 + 200 = 1250 ps

Pipelined: 5 \* 350 = 1750 ps

1. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time?

The stage that I would split would be the ID stage into two, 125 ps stages. This make the new clock cycle dependent on the MEM cycle so, the new time would be 300ps.

1. Assuming there are no hazards, what is the utilization of the data memory?

Utilization = LW + SW = 20% + 15% = 35%

1. Assuming there are no hazards, what is the utilization of the write-register port of the Registers unit?

Utilization = ALU + LW = 45% + 20% = 65%

1. 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 | JMP | LW | SW |
| 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 2 branch outcomes are determined in the EX stage, that there are no data hazards, and that no delay slots are used.

Accuracy of mispredicted always-taken branches = 100 – 45 = 0.55

Extra CPI = 1 + 2 \* 0.55 \* 0.25 = 1.275

1. Repeat 2.a for the “always-not-taken” predictor.

Accuracy of mispredicted always-not-taken branches = 100 – 55 = 0.45

Extra CPI = 1 + 2 \* 0.45 \* 0.25 = 1.225

1. Repeat 2.a for the 2-bit predictor.

Accuracy of mispredicted 2-bit branches = 100 – 85 = 0.15

Extra CPI = 1 + 2 \* 0.15 \* 0.25 = 1.075

1. With the 2-bit predictor, what speedup would be achieved if we could convert half of the branch instructions 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.

CPI without conversion = 1 + 3 \* (1 – 0.85) \* 0.25 = 1.1125

CPI with conversion = 1 + 3 \* (1 – 0.85) \* 0.25 \* 0.5 = 1.05625

Speed up = 1.1125 / 1.05625 = 1.0532

1. 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.

CPI without conversion = 1 + 3 \* (1 – 0.85) \* 0.25 = 1.1125

CPI with conversion = 1 + (1+ 3 \* (1 – 0.85)) \* 0.25 \* 0.5 = 1.18125

Speed up = 1.1125 / 1.18125 = 0.941

1. Some branch instructions are much more predictable than others. If we know that 80% of all executed branch instructions are easy-to-predict loop-back branches that are always predicted correctly, what is the accuracy of the 2-bit predictor on the remaining 20% of the branch instructions?

Accuracy = 0.15 / 0.2 = 75%