CS422 Mid-semester Examination

1. Each instruction in a certain processor goes through seven stages of processing, namely, IF, ID, RF, EX, MEM1, MEM2, WB with the usual meaning of the stages. The stage latencies are as follows. IF: 2 ns, ID: 1 ns, RF: 2 ns, EX: 1 ns, MEM1: 1.2 ns, MEM2: 1.2 ns, WB: 2 ns. Answer the following questions. Ignore all delays in the pipeline registers.

(a) Consider a program that has 20% branch instructions (complete in EX stage), 15% store instructions (complete in MEM2 stage), and each of the remaining instructions updates some register in the WB stage. Calculate the execution time of this program in single-cycle and multi-cycle unpipelined implementations of this processor. **(3+3 points)**

Solution: In single-cycle implementation, the cycle time is the addition of all stage latencies i.e., 10.4 ns. Let the number of instructions be 100. Therefore, execution time is 100*10.4 ns or 1040 ns. In multi-cycle implementation, the cycle time is determined by the longest pipe stage i.e., 2 ns. Average CPI is (0.2*4+0.15*6+0.65*7) or 6.25. Therefore, execution time is 100*6.25*2 ns or 1250 ns.

(b) Consider a pipelined implementation of this processor. Assume that the load instructions produce values at the end of the MEM2 stage and there is no phased execution of pipeline stages. Further assume that the compiler does not fill any branch delay slots or load delay slots. Enumerate all bypass paths by specifying the source and destination of each bypass path. Describe any interlock logic that must be implemented in the ID stage of the pipeline. **(10+5 points)**

Solution: The first result producing stage is EX. Therefore, the potential bypass sources are EX/MEM1, MEM1/MEM2, MEM2/WB, and WB. Only EX and MEM1 stages initiate new computations. These are the potential bypass destinations. A bypass into the MEM1 stage is required only for a store instruction that picks up the value to be stored from the bypass path. As a result, the WB to MEM1 bypass can also be satisfied by MEM2/WB to EX bypass. Overall, the following bypass paths are necessary: EX/MEM1 to EX, MEM1/MEM2 to EX, MEM2/WB to EX, WB to EX, and MEM2/WB to MEM1. Interlock cycles would be needed for an instruction I in the following conditions: (i) if instruction I is a store and the value to be stored is produced by instruction $I-1$, which is a load; (ii) if instruction I needs a value in EX stage which is produced by instruction $I-1$ or $I-2$, which is a load.

(c) Assume that the processor implements all possible bypass paths enumerated in the last question. There is no branch predictor in the processor and the instruction fetcher simply feeds NOPs into the pipeline after fetching a branch instruction until the branch executes. The compiler team informs the architecture team that it is possible to fill the slot right after a load instruction with an instruction that is independent of the load. Assuming that the average branch frequency is 20% of all instructions and that the compiler guarantees not to put branch instructions in load delay slots, under what condition is it meaningful (from performance viewpoint) for the architecture team to consider fusing the MEM1 and MEM2 stages into a single MEM stage of 2.4 ns latency? **(10 points)**

Solution: Without fusion, cycle time is 2 ns and two-cycle apart MEM2 to EX bypass requires an interlock cycle. Let the percentage of instructions that are loads and have a two-cycle apart EX dependent be $x\%$. Therefore, without fusion, CPI is $1 + (0.2 * 3 + 0.01x)$. With fusion, cycle time is 2.4 ns and CPI is $1 + 0.2 * 3$. Therefore, we want $(1 + 0.2 * 3) * 2.4 < (1 + (0.2 * 3 + 0.01x)) * 2$ or $x > 32$.

(d) Suppose that you are allowed to split one or more pipeline stages into two equal parts to improve the frequency of the pipeline shown in part (b). What is the next higher achievable frequency? How deep would the pipeline be to achieve this frequency? How does the performance of this deeper pipeline compare with the shallower original seven-stage pipeline? Your answer can be parametrized in terms of instruction frequencies (e.g., you may assume that $b\%$ of all instructions are branch instructions). Assume that both the pipelines implement all possible bypass paths and in both cases, after fetching a branch instruction, the first pipeline stage keeps feeding NOPs into the pipeline until the branch executes. Also, in both cases, it is possible to fill the slot right after a load instruction with an instruction that is independent of the load and the load delay slots are never filled with branch instructions. **(2+2+10 points)**

Solution: The next achievable cycle time is 1.2 ns (frequency 833 MHz). This can be achieved by splitting the IF, RF, and WB stages into two. The depth of the resulting pipeline is ten. In both pipelines, the penalty due to load delays is same. In the deeper pipeline, the branch penalty increases by two cycles. Let us assume that $b\%$ of all instructions are branch instructions and $x\%$ of all instructions are loads that have a two-cycle apart EX dependent. Therefore, time to execute one instruction in the original pipeline is $(1+0.01b*3+0.01x)*2$ or $2+0.06b+0.02x$. Time to execute one instruction in the new pipeline is $(1 + 0.01b * 5 + 0.01x) * 1.2$ or $1.2 + 0.06b + 0.012x$. It is clear that irrespective of b and x, the deeper pipeline has better performance.

2. Consider a processor with a seven-stage pipeline: IF, ID, RF1, RF2, EX1, EX2, MEM, WB with the pipeline stages having the usual meanings. The branch instructions complete in the EX2 stage. The pipeline stages do not have any phased execution support. The compiler fills the load delay slot with an instruction that is independent of the load instruction. The pipeline has all the necessary bypass paths. The RF1 stage has a direction predictor, which is looked up by conditional branches after they are decoded in the ID stage. The RF1 stage also has an adder to compute the taken target of conditional branches. The conditional branch instruction has a pre-decoded bit in the instruction, which the fetcher checks to detect conditional branches in the IF stage. On such an indication, the IF stage stalls until it is redirected by the direction predictor. A certain conditional branch instruction in a program exhibits 011011011... pattern and the branch executes exactly one thousand times. Calculate the average misprediction penalty of this conditional branch in the presence of two different direction predictors: (a) A bimodal predictor with 2-bit saturating counters, (b) A SAg predictor with 12-bit history in the first-level table and 2-bit saturating counters in the second-level table. Assume that this branch does not alias with any other branches in the bimodal predictor and the first-level table of the SAg predictor. All counters and histories are initialized to zero. The prediction is taken if the saturating counter value is at least two; the prediction is not-taken otherwise. **(10+15 points)**

Solution: The bimodal predictor's counter will show the following values: 0, 0, 1, 2, 1, 2, 3, 2, 3, 3, 2, 3, 3, 2, 3, 3, ... Therefore, the predictions are 0001011111... On every branch, two cycles are always lost due to the stall. There is an additional loss of three cycles if the prediction turns out to be wrong. There are 336 mispredictions. So, the average misprediction penalty is 2+(336*3)/1000 cycles or 3.008 cycles. To calculate the number of mispredictions in the SAg predictor, we need to notice that there three possible steady-state history patterns: 011011011011, 110110110110, 101101101101. Additionally, the initial few branch instances will observe history patterns different from these three: 000000000000, 000000000001, 000000000011, 000000000110, 000000001101, 000000011011, 000000110110, 000001101101, 000011011011, 000110110110, 001101101101. All these initial patterns will emit not-taken predictions. The steady-state pattern 011011011011 will always emit not-taken prediction, which is correct. The steady-state pattern 110110110110 will emit not-taken prediction in the first two instances and the remaining predictions will be taken. The steady-state pattern 101101101101 will emit not-taken prediction in the first two instances and the remaining predictions will be taken. Therefore, the total number of mispredictions is twelve. So, the average misprediction penalty is 2+(12*3)/1000 cycles or 2.036 cycles.