**Model Solution**

**Mid-Spring Semester Examination 2023--24**

**High Performance Computer Architecture (CS60003)**

1. What is the asymptotic prediction accuracy of a two-bit bimodal branch predictor on an infinitely repeating pattern of branch outcome: **…TTNTTTTNTTNTNT... TTNTTTTNTTNTNT….** (T=taken, N=not taken). Assume that the predictor is initialized to “strongly not taken” before the start of the program. **[5]**

**Ans: 71.5% - eventually the predictor will hit the strongly-taken state after 4 T’s – after that it will always predict T and be correct 10 of the 14 branch executions.**

1. What is the storage size required (in bits) for implementing a 128-entry (2,2) correlating predictor? Clearly show your workout. **[5]**

**Ans: Size = # entries \* # predictors \* size of predictor = 128 \* 22 \* 2 = 128 \* 4 \* 2 = 1024 bits**

1. In a simple MIPS pipeline a static not taken predictor is used. In case of a misprediction, the instructions being speculatively executed are quashed. A program has 20% branches, out of which 60% are taken branches and 40% are not taken. What is the speed up of a pipeline over a pipeline that uses instruction flushing whenever any branches are encountered? **[5]**

**No Prediction: 1+0.2\*2= 1.4 CPI**

**Not Taken Prediction= 1+0.2\*0.6\*2=1.24 CPI**

**Speedup=1.4/1.24=1.129 0r 12.9%**

1. The inner loop of a program has three branches (b1, b2, and b3) that execute in sequence in every iteration. The outcome pattern for the three branches is shown below. The inner loop always executes for 10 iterations. It resides within an outer loop that executes for thousands of iterations.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1**  **Iteration:**  **b1:** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |  | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **…** | **…** |
| N  **b2:** | N | N | N | T | N | T | T | T | T |  | N | N | N | N | T | N | T | … | … |
| N  **b3:** | T | T | T | N | T | N | N | N | N |  | N | T | T | T | N | T | N | … | … |
| T | T | T | T | T | T | T | T | T | N |  | T | T | T | T | T | T | T | … | … |

In the steady state, what is the prediction accuracy for each of the following predictors on these three branches? Assume that the single bit predictors have been initialized to NT and the 2-bit predictor has been initialized to “strongly not taken”. To concisely present your answer, copy the following table to your answer book and fill in the blanks in the table. **[9]**

**Note:** A 1-entry BHT means that there is only 1 history remembered for the entire processor. A huge BHT means that each branch in this code has its history tracked separately.

|  |  |  |  |
| --- | --- | --- | --- |
| **Predictor** | **Accuracy on b1** | **Accuracy on b2** | **Accuracy on b3** |
| 1bit predictor with 1entry BHT | **6/10** | **1/10** | **5/10** |
| 1-bit predictor with huge BHT | **6/10** | **6/10** | **8/10** |
| 2-bit predictor with huge BHT | **5/10** | **5/10** | **9/10** |

1. Suppose you are designing a new processor that is similar to the 5-stage MIPS processor. However, you observe that the data memory (DM) stage exhibits nearly twice as much latency as the other stages. In this context you want to evaluate two options. **Option 1:** Increase the pipeline cycle time by a factor of two. **Option 2:** Split the data memory (DM) stage into two separate stages: **DM1** and **DM2**. Which design will achieve a higher speed up for executing a program having 10 instructions and by how much? Ignore various pipeline hazards and pipeline overheads. **[10]**

**Ans:**

**Option 1: Increase cycle time**

**Execution time = (5+9) \* 2= 28 cycles Award 4 Marks**

**Option 2: Split DM stage**

**Execution time = (6+9) cycles = 15 cycles Award 4 Marks**

**Speedup with option 2= 28/15=1.87 Award 2 Mark**

1. Consider the following code. It has to be run on a simple MIPS 5-stage pipeline. The processor has no forwarding or control circuits to handle hazards and these are to be taken care appropriately by the compiler by suitably restructuring the code and adding NOOPs.

**LW R1, 10000(R7)**

**ADD R5, R6, R1**

**BEQZ R1, L1**

**SUB R8, R1, R3**

**L1: ADD R4, R3, R9**

**AND R2, R4, R8**

**ADD R3,R2,R1**

1. Add NOOP instructions wherever required without restructuring the code, so that hazards do not arise. Write the resultant code and also write the number of clock cycles that your code would take to execute. [**3]**
2. Suitably restructure the code and use delayed branch technique. Insert NOPs only when it is unavoidable. Write your restructured code and the number of clock cycles that your code would take to execute. **[7]**

**Ans:**

**a)**

**LW R1, 10000(R7)**

**NOOP**

**NOOP**

**ADD R5, R6, R1**

**BEQZ R1, L1**

**NOOP**

**NOOP**

**SUB R8, R1, R3**

**L1: ADD R4, R3, R9**

**NOOP**

**NOOP**

**AND R2, R4, R8**

**NOOP**

**NOOP**

**ADD R3,R2,R1**

**19 Cycles**

**b)**

**With Code restructuring**

**LW R1, 10000(R7)**

**ADD R4, R3, R9**

**NOOP**

**BEQZ R1, L1**

**ADD R5, R6, R1**

**NOOP**

**SUB R8, R1, R3**

**NOOP**

**NOOP**

**L1 : AND R2, R4, R8**

**NOOP**

**NOOP**

**ADD R3, R2, R1**

**16 Cycles if BEQZ is True and 17 Cycles Otherwise**

1. The performance of a certain multi-cycle non-pipelined processor is evaluated using a program that takes 100 seconds to execute. Of this time, 20% is used for ALU operations, 50% for memory access instructions, and 30% for other types of instructions. A designer targetting to enhance the performance of the processor through two possible improvements: (i) make the ALU instructions run four times faster than before, (ii) make memory access instructions run two times faster than before. **[3+3+4]**
2. What would be the speedup, if the performance of the ALU unit alone is enhanced?

**Ans: 1/(0.8+0.2/4)=1/0.85 =1.1765**

1. What would be the speedup if the performance of the memory access unit alone is enhanced?

**Ans: 1/(0.5+0.5/2)=1/0.75= 1.333**

1. What will the speedup be if both the proposed enhancements are incorporated?

**Ans: 1/(0.3+0.05+0.25)=1/0.6 = 1.666**

1. A simple MIPS processor has no circuit for control hazard detection or mitigation. It has two branch delay slots. An optimizing compiler can fill the first slot 80% of the time and can fill the second slot 20% of the time. The compiler fills the second slot only after the first slot is filled. What is the percentage improvement in performance achieved by this optimizing compiler relative to a compiler that fills the branch delay slots by NOOPS? Assume that a branch occurs once every 7 instructions. **[10]**

**Ans: CPI unoptimized = 9 cycles / 7 instructions = 1.2857 … 2 Mark**

**80% of the time, the first slot is filled, therefore 20% of the time neither slot is filled and there is a 2 cycle delay**

**20% of the time the second slot is filled along with the first slot, therefore there is no delay**

**80%-20%=60% of time only the first slot is filled, therefore there is a 1 cycle delay**

**CPI optimized = (7 + 0.2 x 2 + 0.6 x 1) / 7 = 8 / 7 = 1.14 … 6 Marks**

**%improvement = 11.25% …. 2 Mark**

1. Assume that a MIPS processor with 5 stage instruction pipeline with split cache is being used for executing a program. A not-taken predictor is used in the pipeline. There is no forwarding hardware. Assume that the characteristics of the program being executed are as follows. What is the average CPI? **[10]**

* 10% of the instructions are load instructions, and 40% of loads are used by the immediate next instruction. No other instructions cause a data hazard.
* 20% of the instructions are branches. Of these 10% are unconditional branches. Of the conditional branches, 50% on the average turnout to be taken.

**Solution:**

**40% loads are used by next instruction.**

**Average CPI= 1+ Load stall Cycles + Uncond stall cycles+ Cond stall cycles**

**= 1+ 0.1×0.4×2+ 0.2×0.1×2+0.2×0.9×0.5×2**

**=1+0.08+0.04+0.18**

=**1.3**

1. Suppose that a processor uses a 10-stage instruction pipeline with the following stages

**F1 – start the fetch, predict if the instr. is a branch, whether it is taken or not, and if taken, the target address.**

**F2 – complete the fetch**

**D1 – decode the instruction – know it’s a branch at the end of D1,**

**D2 – complete decode**

**RR – read the registers– branch target address available at the end of this stage**

**A1 – start ALU operation; resolve branch condition**

**A2 – complete ALU operation**

**M1 – start memory operation**

**M2 – complete memory operation**

**WB – write the result to registers**

1. What is the penalty for a mispredicted branch? **[2]**

**Ans: 5 cycles**

1. What is the penalty when a branch is correctly predicted as taken, but the branch target address is incorrectly predicted? **[2]**

**Ans: 4 cycles:**

1. Assume the following
2. There are no stalls in this pipeline except for branch instructions.
3. Branch instructions account for 25% of all instructions
4. 20% of branch instructions are taken
5. Branch outcome is correctly predicted 90% of the time
6. The target address for a taken branch is correctly predicted 80% of the time

What is the CPI for this machine? **[7]**

**Ans: CPI = .75\*1 + .25 \* (.2\*(.9\*.8\*1+.8\*.2\*5+.1\*6) +.8\*(.9\*1+.1\*6)) = 1.161**

1. Consider an 8-stage instruction pipeline, consisting of the stages: IF1, IF2, ID, EX1, EX2, M1, M2, WB. The branch addresses are resolved at the end of the ID stage and branch conditions are resolved at the end of the EX2 stage. The typical workload has 20% conditional branches with 75% of the conditional branches taken, on the average. Assume that only control hazards cause pipeline stalls and the pipeline does not stall on account of any other issue.

a) What is the CPI if a statically "predict-not-taken" scheme is used? **[3]**

**Ans: Each wrong prediction (taken branch) will cause a penalty of 4 cycles**

**CPI = 1 + 0.2 \* 0.75 \* 4 = 1.6**

b) What is the CPI if a "predict-taken" scheme is used? Assume that no Branch Prediction Buffer/Table is used and that the target address can only be known at the end of the ID stage. **[5]**

**Ans: The branch address is only known in ID, at this point we start to fetch from the target address. There are already two instructions in the pipe (will have to be removed if the branch is actually taken).**

**When the branch condition is resolved in EX2, there are 4 instructions in the pipe, two from the normal flow and two from the target address. The two instructions in ID and EX1 will have to be removed if the branch is taken and the two in IF1 and IF2 will have to be removed if the branch is not taken.**

**CPI = 1 + 0.2 (0.75 \* 2 + 0.25 \* 2) = 1 + 0.2 (2) = 1 + 0.4 = 1.4**

c) If a branch target buffer is added so that the branch address is predicted during the IF1 stage, what would be the CPI? Assume that the branch target buffer does not give any prediction 10% of the time, in which case, a "predict-not-taken" scheme is applied. For the 90% of the time where a prediction is given, the prediction is correct with a probability of 50%. Assume that when a branch condition is predicted correctly, the target address is also predicted correctly. **[7]**

**Penalty if no prediction = 0.1 \* 0.75 \* 4 = 0.3 (predict not taken applies in this case) ..1 Mark**

**Penalty for correct prediction = 0.9 \* 0.5 \* 0 = 0 … 2 Mark**

**Penalty for wrong prediction = 0.9 \* 0.5 \* 4 = 1.8 … 2 Mark**

**CPI=1+0.2(0.3+1.8) = 1.42 … 3 Marks**

**--- The End---**