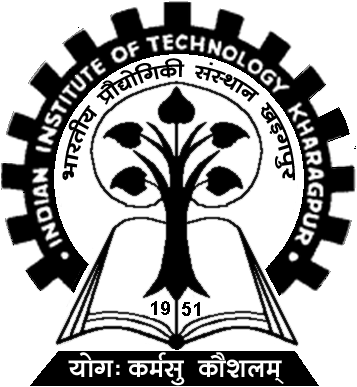
** INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR**

**Mid-Spring Semester 2023**

**High Performance Computer Architecture (CS60003)**

**Time=2 Hours Max Marks=100**

**Important Instructions:**

* **This question paper consists of two sections. You need to answer only one of the sections. 2 Year M.Tech students need to answer Section A only and all other students need to answer Section B only.**
* **No clarification to any of the questions shall be provided. In case you have any queries, you can make suitable assumptions, but please write down your assumptions clearly.**
* All answers should be brief and concise. Lengthy and irrelevant answers will be penalized.

**Section B (for 2-Year M.Tech Students)**

1. Determine the asymptotic prediction accuracy of a two-bit bimodal branch predictor on the following infinitely repeating pattern of branch outcomes: **…TTNTTTTNTTNTNT... TTNTTTTNTTNTNT …. TTNTTTTNTTNTNT…** (**T**=taken, **N**=not taken). Assume that the predictor is initialized to “strongly not taken” before start of program execution. **[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 correctly predict 10 of the 14 branch outcomes.**

1. A processor named **ST1** is has been designed as a simple MIPS pipeline with a static not taken branch predictor. In case of a misprediction, the instructions being speculatively executed are quashed. A program being executed on **ST1** has **20%** branches, out of which **60%** are taken branches and **40%** are not taken. For this program what is the expected speed-up of **ST1** over a pipeline that uses instruction flushing for all branches? **[5]**

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

**Not Taken Prediction= 1+0.2\*0.6\*2=1.24 CPI ..3**

**Speedup=1.4/1.24=1.129 0r 12.9% …. 1 mark**

1. Consider a simple 5 stage MIPS processor. Assume that it uses a single data/instruction cache, and 30% of all instructions issued are of load/store type. The program being executed does not cause any data or control hazards. Determine the increase/decrease in memory bandwidth of the MIPS processor compared to a similar processor that does not use instruction pipelining. Assume that the memory bandwidth is defined as the number of bytes that need to be accessed per cycle. **[5]**

**Ans:**

**Assuming no data and control hazards:**

**For the single data/instruction cache: CPI=1+0.3=1.3 Award 2 Marks**

**For a processor that does not use pipeline: CPI=5 Award 2 Marks**

**Increase in memory bandwidth=5/1.3 =3.84 Award 1 Marks**

1. Assume that a MIPS processor with a simple 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 given below. Compute the average CPI. **[5]**

* 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 you are an engineer at ALEC microsystems and you have designed a pipelined processor with cycle time of **10** **ns**. Your processor exhibits an average CPI of **1.6** on SPECINT2000 program. In the SPECINT2000 program **10%** of the instructions are branches and the branch prediction scheme deployed in the processor is **90%** accurate. Branch misprediction penalty is **2** cycles. You are considering a new processor design where you decrease the cycle time to **9 ns** by increasing the depth of the pipeline. In this new design the cost of a misprediction will increase to **7** clock cycles, but everything else will remain the same. Compute the average CPI on the new processor for the benchmark. Will your benchmark program run faster or slower on this new processor? Show the details of your workout. **[6]**

**Answer: In the new processor the misprediction penalty increases by (7-2) = 5 cycles for 10% of the branches.**

**Hence new CPI = 1.6 + 0.1\*0.1\*5 = 1.65 ….. Award 3 Marks**

**Time taken on workload = Number of Instructions (N) \* CPI \* cycle time**

**Time taken on old machine = N \* 1.6 \* 10 = 16\*N … Award 1 Mark**

**Time taken on new machine = N \* 1.65 \* 9 = (1.65\*9)\*N =14.85N … Award 1 Mark**

**The new machine is faster. … Award 1 Mark**

1. A designer has proposed a hardware optimization for a given processor that would outright eliminate **10%** of instructions of a benchmark program and also decrease the CPI of the remaining instructions by **10%**. Unfortunately, this optimization would result in decreasing the clock rate by **10%**. Is this optimization worth implementing? Show the details of your calculations. **[6]**

**IC(new) = 0.9 \* IC(old) //// 10% of instructions eliminated**

**CPI(new) = 0.9 \* CPI (old) ////CPI decreased by 10%**

**Clock rate(new) = 0.9 \* clock rate(old)**

**Execution time(new)= 0.9\*IC(old) \* 0.9\*CPI(old) \* 1/(0.9 \* clk rate(old))**

**= 0.9 \* IC(old) \* CPI(old) / clk rate(old)**

**= 0.9 \* Execution time(old)**

**Since the new execution time is lower, the optimization is worthwhile**.

1. Consider the following code containing a loop. 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 required to be taken care appropriately by the compiler by suitably restructuring the code and adding NOOPs.

**ADDI R1, R0, 100**

**L1: ADD R2, R2, 1**

**ADD R3, R3, R2**

**ADDI R1, R1, -4**

**BNEZ R1, L1**

**….. other code …**

1. For the given code, how many NOOPs would have to be added, so that hazards do not arise? Write the code with NOOP instructions added wherever required without restructuring the code. **[3]**
2. Suitably restructure the code given below and also add NOOP instructions wherever required so that that hazards do not arise and the number of NOOPs is minimized. **[4]**

**Ans:**

1. **ADDI R1, R0, 100**

**L1: ADD R2, R2, 1**

**NOOP**

**NOOP**

**ADD R3, R3, R2**

**ADDI R1, R1, -4**

**NOOP**

**NOOP**

**BNEZ R1, L1**

**NOOP**

**NOOP**

**----Other code ----**

**ii) Restructured code:**

**ADDI R1, R0, 100**

**L1: ADD R2, R2, 1**

**ADDI R1, R1, -4**

**NOOP**

**NOOP**

**BNEZ R1, L1**

**ADD R3, R3, R2**

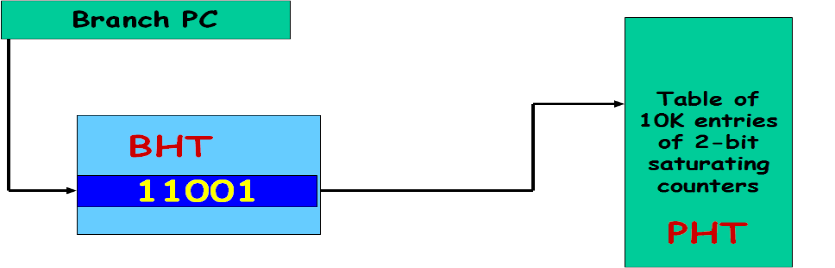
**NOOP**

**----Other code ----**

1. Assume that you are designing a 2-level (correlating) local predictor a certain pipelined processor. In this predictor, the last 6 bits of the PC address will be used to select the appropriate branch history register (BHR). The size of each BHR is 6 bits. The branch history will be used to select an appropriate pattern history table (PHT) of 2-bit saturating counters. Each PHT maintains 1024 entries. Determine the cache size required to implement the 2-level local predictor.  **[7]**

**Ans: PHT= 2\*26\*26\*210 = 8\*220 = 8Mb, …. 5 marks**

**BHT= 6\*26 = 384b ≈ 0.5Kb… 2 marks**



1. A simple MIPS processor has two branch delay slots. An optimizing compiler can fill the first slot **85%** 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 does not fill any of the branch delay slots? Assume that a branch occurs once every **7** instructions. **[7]**

**Ans: CPI unoptimized = 9 cycles / 7 instructions = 1.2857 … 1 Mark**

**85% of the time, the first slot is filled, therefore 15% 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**

**Then 85% x 20% of the time there is no delay and 85% x 80% there is a 1 cycle delay**

**CPI optimized = (7 + 0.15 x 2 + 0.85 x 0.8 x 1) / 7 = 7.98 / 7 = 1.14 … 5 Marks**

**%improvement = 12.8% …. 1 Mark**

1. An engineer is trying to use a branch predictor for a processor designed based on the simple MIPS pipeline. In the program being run on the processor, branches constitute **20%** of all instructions. The engineer has essentially two branch predictors to choose from: PicoPruner and ChartChooser. For both these predictors, the branch mispredict penalty is **2** cycles. Branches that are correctly predicted incur no penalty. Simulation of PicoPruner shows **10%** misprediction rate, but implementing PicoPruner will increase the cycle time by **20%**. Simulation of ChartChooser shows **20%** misprediction rate, but its implementation will increase the cycle time by **10%** . Which predictor should the designer choose?Show the details of your computations. **[7]**

**Ans:**

**CTPicoPruner =1.2**

**CPIPicoPrunerer =1+ 0.2 ∗ 0.1 ∗ 2 = 1.04**

**ETPicoPruner = CTPicoPruner ∗ CPIPicoPruner =1.2 ∗ 1.04 = 1.248 Award 3 marks**

**CTChartChooser =1.1**

**CPIChartChooser =1+0.2 ∗ 0.2 ∗ 2= 1.08**

**ETChartChooser = CTChartChooser ∗ CPIChartChooser =1.1 ∗ 1.08 = 1.188 Award 3 marks**

**ChartChooser is the better branch predictor. Award 1 mark**

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 ten thousand 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. Consider the following program being run on a MIPS processor deploying a single-level (non-correlating) bimodal predictor of 2-bit saturating counters. What is approximate prediction accuracy for each of the three branches (at labels L1, L2, and L3) in the program code? **[9]**

**ADD $t1, $R0,$R0**

**ADD $t2, $R0,$R0**

**ADDI $t3, $R0,2**

**ADDI $t4, $R0,3**

**SLL $t5,$t4,20 #Bits in $t4 shifted left logical by 20 places**

**LOOP: ADDI $t1, $t1,1**

**L1: BLE $t1,$t3,L2 #if $t1<= $t3 branch to L2**

**ADDI $t2, $t2,1**

**L2: BLE $t1,$t4, L3 #if $t1<= $t4 branch to L3**

**ADD $t1, $R0,$R0**

**L3: BLE $t2,$t5, LOOP #if $t2<= $t5 branch to LOOP**

**Ans: Accuracy for branch at L1: The pattern for this branch is: TTNN, so 0% accuracy for bimodal. … 3 marks**

**Accuracy for branch at L2: The pattern for this branch is: TTTN, so 75% accuracy for bimodal. . … 3 marks**

**Accuracy for branch at L3:The pattern for this branch is: TTTT, so 100% accuracy for bimodal. . … 3 marks**

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. Now assume the following
2. Data hazards are completely eliminated by compiler. Therefore, the only stalls in this pipeline are due to 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? **[6]**

**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. **[4]**

**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. **[5]**

**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 … 1 Mark**

**Penalty for wrong prediction = 0.9 \* 0.5 \* 4 = 1.8 … 1 Mark**

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

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