# 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 B only and all other students need to answer Section A 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 A **(To be answered by UG and RS Students, Not 2-year MTech Students)**

**1.** Consider a program with **50 Billion** instructions, running on a processor having clock frequency of **4GHz**. An architectural modification enhances the branch instruction so that its *CPI goes from 4 to 2*. **40%** of the instruction is arithmetic with CPI 1, **20%** is branch with CPI 4, **30%** load with CPI 2 and **10%** is store with CPI 3.

a. Calculate the overall speedup. **[2]**

b. As a processor designer, you have been suggested two choices to reduce the overall CPI of the processor. In one case, you reduce the execution time of arithmetic instructions by **20%**, but that leads to a **50%** increase in execution time of load instructions. In the other case, you optimize the memory system such that both load and store execution times are reduced by **20%** but that in turn affects the branches by **10%** (increase in time). So which one will you choose? Or do you prefer to keep the original enhancement? Give reasons for your answer. **[5]**

**2.** Consider a multi-cycle processor **P1** executes load instructions in **10** cycles, store instructions in **8** cycles, arithmetic instructions in **4** cycles, and branch instructions in **4** cycles. Consider an application A, where **20%** of all instructions are *load* instructions, **20%** of all instructions are *store* instructions, **50%** of all instructions are *arithmetic* instructions, and **10%** of all instructions are *branch* instructions.

a. What is the CPI of application A when executing on processor **P1**? Show your work. **[3]**

b. A new design of the processor *doubles the clock frequency of* **P1**. However, the latencies of the *load, store, arithmetic, and branch* instructions increase by **2**, **2**, **2**, and **1** cycles, respectively. We call this *new* processor **P2**. The compiler used to generate instructions for **P2** is the same as for **P1**. Thus, it produces the same number of instructions for program A. What is the CPI of application A when executing on processor **P2**? Show your work. **[5]**

c. Which processor is faster (**P1** or **P2**)? Show by computing the speedup. **[2]**

d. There is some extra area available in the chip of processor **P1**, where extra hardware can fit. You can decide to include in your processor a faster branch execution unit or a faster memory device. The faster branch execution unit reduces the latency of branch instructions by a factor of **4**. The memory device reduces the latency of the memory operations by a factor of **2**. Which design do you choose? Show your work. **[3]**

**3.** The following program is to be run on a **5**-stage pipeline processor of the form **IF-ID-EX-MEM-WB** (as discussed in class).

a. Please identify all data dependencies beside each instruction, in the form: <type> on <register> from <line number> to <line number>. *Example:* **WAW** on **r6** from **I8** to **I10**

```
I1: lw r2, 60(r1)
I2: lw r1, 40(r2)
I3: slt r1, r1, r2     //set r1 = 1, if r1 < r2
I4: sw r1, 20(r2)
```
**[5]**

b. Work out the optimal pipeline schedule using forwarding from **EX** or **MEM** stages to any other stage, then compute the pipeline **CPI**. **[5]**

**4.** Let's compute the degradation of the **RISC-V**'s pipeline performance due to realistic assumptions of hazard situations. A benchmark consists of **27%** *loads*, **10%** *stores*, **45%** *ALU operations*, **15%** *conditional branches*, **3%** *unconditional branches*. We use the version of **RISC-V** with forwarding and branches computed in the ID (decode) stage. Assume **10%** of all loads/stores use indirect addressing (the mode of addressing where the instruction contains the address of the location where the target address is stored), so we have a load followed immediately by a load/store. For example, an instruction like **lw r2, 0(r1)** is preceded by **lw r1, 100** such that the value **100** is first loaded into **r1** and then that particular memory location is read and the respective content is loaded into **r2**.

Assume **40%** of all Loads have an ALU operation immediately afterwards that uses the loaded value. Assume **60%** of all conditional branches immediately follow an ALU operation that places a result in the register to be tested in the branch. Assume a further **10%** of all conditional branches immediately follow a load of the register to be tested in the branch (note that this requires a 2-cycle stall because the data is loaded in the **MEM** stage but needed in the branch's ID stage). An optimizing compiler can schedule code to eliminate **50%** of the indirect memory reference stalls, **60%** of all load-ALU stalls, fill **65%** of the branch delay slots and fill **45%** of the stalls between ALU-branch and **50%** of one stall from the load-branch. Compare the difference between the unoptimized and optimized versions and the optimized and ideal (no stalls). **[10]**

**5.** Let's compare a **CISC** (Complex Instruction Set Computer) machine versus a **RISC** (Reduced Instruction Set Computer) machine on a benchmark. Assume the following characteristics of the two machines.

a. **CISC**: CPI of **4** for load/store, **3** for ALU/branch and **10** for call/return, the CPU clock rate of **3.5** GHz.

b. **RISC**: CPI of **1.3** (the machine is pipelined, the ideal CPI is **1.0**, but overhead and stalls make it **1.3**) and a CPU clock rate of **1.75** GHz.

Since the **CISC** machine has more complex instructions, the total number of instructions for the **CISC** machine is **30%** smaller than that for the **RISC** machine. The benchmark has a breakdown of **38%** loads, **10%** stores, **35%** ALU operations, **3%** calls, **3%** returns and **11%** branches.
Which machine will run the benchmark in less time and by how much? **[10]**

**6.** In the speculative repair mechanism for a **Gshare** predictor, a **FIFO** queue of global registers is accessed at three different times:

- When the branch prediction is made (insertion at the end of the queue).
- When the branch prediction is correct, and the instruction is committed (deletion from the front of the queue).
- When the branch prediction is incorrect at the end of the execute stage (deletion of several consecutive entries in the **FIFO** queue).

In this context, answer the following:

a. What extra hardware must be added so that repair at execute is possible? If there is a maximum of *m* branches in flight, can the **FIFO** queue be less than *m* entries? If so, what are the consequences? **[5]**

b. A suggestion is to have each **FIFO** entry tagged with the PC of the branch it corresponds to. Then the branch would not have to carry a tag. Why is this not possible? **[5]**

**7.** Assume a **3**-bit global register (initialized to **all 0's**) and a *two-level* **Gshare** predictor.

a. How many entries are there in the **PHT**? **[2]**

b. When the branch prediction is correct, and the instruction is committed (deletion from the front of the queue). Show the contents of the global register and of the **PHT** (where each entry is initialized to the weak not-taken case and follows the state diagram after the following sequence of branch executions has taken place: **1** taken **(T)** branch, **3** not-taken **(NT)** branches, **1 T** branch, **3 NT** branches, **1 T** branch. With pred being the prediction and act the actual outcome as per the given string, show your results for each branch, for example, in the form (**GR**) **(pred) (Updated GR) (act)** (**PHT**). Assume a speculative update for **GR (Global Register)** and a non-speculative one for the **PHT** that occurs before the next branch prediction. On misprediction, restore the **GP** to **(001)**. A typical entry could be:
**(000) (0) (000) (1) (00, 00, 00, 00, 00, 00, 00, 00)**. Considering this as the starting point, complete the table for the entire sequence. **[8]**

| GR | pred | Updated GR | actual | correct? | PHT |
|---|---|---|---|---|---|
| 000 | 0 | 000 | T | No | 00, 00, 00, 00, 00, 00, 00, 00 |

**8.** Consider the following assembly code that searches for an element in an array. An equivalent C code has been provided

| | | |
|---|---|---|
| 0 | `        lw   r1   one          // r1 = 1` | `int arr[] = [15,4,5,24,13]` |
| 1 | `        lw   r5   size        // r5 = size` | `int size = 5;` |
| 2 | `        lw   r3   zero         // r3 = i = 0` | `int i = 0;` |
| 3 | `        lw   r2   search       // r2 = search` | `int search = 24` |
| 4 | `loop:  beq  r5   r3  end     // B0` | `for(i = 0; i < size; ++i)` |
| 5 | `        lw   r4   r3  arr     // r4 = arr[r3]` | `{` |
| 6 | `        beq  r2   r4  end     // B1` | `    if(arr[i] == search)` |
| 7 | `        add  r3   r1  r3` | `        break;` |
| 8 | `        jmp  loop              // B2` | `}` |
| 9 | `end:   halt` | |
| 10 | `zero   .fill   0` | |
| 11 | `one    .fill   1` | |
| 12 | `arr    .fill  15` | |
| 13 | `       .fill   4` | |
| 14 | `       .fill   5` | |
| 15 | `       .fill  24` | |
| 16 | `       .fill  13` | |
| 17 | `size   .fill   5` | |
| 18 | `search .fill  24` | |

a. Compute how many times each branch is executed during the program.

**B0:** **B1:** **B2:** **Total:** **[2]**

b. Suppose a GLOBAL **2**-bit saturating counter is used to predict all branches. Compute the number of correct predictions for each branch by filling out the table below. Assume that the **2**-bit counter is initialized to **2'b11** (strongly taken). Then derive the branch prediction hit rate.

i. Number of correct predictions (**GLOBAL** counter):

**B0:** **B1:** **B2:** **Total:** **[4]**

ii. Branch prediction Hit Rate: ________________ **[2]**

c. Suppose instead each branch has its own local **2**-bit saturating counter. Fill out the table below assuming that all **2**-bit counters are initialized to **2'b00** (strongly not taken). Then derive the branch prediction hit rate.

i. Number of correct predictions (**LOCAL** counter):

**B0:** **B1:** **B2:** **Total:** **[5]**

ii. Branch prediction Hit Rate: ________________ **[2]**

9. Consider the following sequence of branch outcomes for a program: **(N T T N)** where **T** signifies **Branch is Taken** and **N** signifies **Branch is Not Taken**. The accuracy of prediction is defined as the percentage of guesses which are correct.
   Assume that for a program the sequence is repeated forever. The sophisticated branch prediction unit (**BPU**) uses four **3**-bit predictors to predict the sequence.
   a. Draw the State diagram for the predictor (remember the state diagram for **2**-bit counters shown in class). **[6]**
   b. Now, use **2**-bit history with four **3**-bit predictors to predict the sequence. What is the accuracy of prediction assuming all the predictors start in the "**Not Taken**" state? **[6]**
   c. Calculate the space requirement. **[3]**

## Section B (To be answered by 2-Year M.Tech Students)

**Answer all questions.**

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]**

2. 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]**

3. Consider a simple 5 stage MIPS processor. Assume that it uses a unified 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. **Hint:** Memory bandwidth is the number of bytes that can be accessed per cycle from the memory. **[5]**

4. 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.

5. 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]**

6. 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]**

7. 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 of 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 ...
```

a) 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]**

b) 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]**

8. 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]**

9. 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]**

10. 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]**

11. 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.

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

**12.** 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
```

13. 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**

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

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

c) Now assume the following:

i. Data hazards are completely eliminated by compiler. Therefore, the only stalls in this pipeline are due to branch instructions.
ii. Branch instructions account for **25%** of all instructions.
iii. **20%** of branch instructions are taken.
iv. Branch outcome is correctly predicted **90%** of the time.
v. The target address for a taken branch is correctly predicted **80%** of the time.

What is the CPI for this machine? Clearly show your work out. **[6]**

14. 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]**

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]**

c) If a branch target buffer (BTB) 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]**

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