INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR
Long Test 1 - 2022

Date: 23-02-22 **Timing:** 8 am - 9:30 am; server closes at 9:40 am; **Total Marks** = 30
Subject No : CS60003 HIGH PERFORMANCE COMPUTER ARCHITECTURE
Department/Centre/School : Computer Science and Engineering
Answer all questions. In case of reasonable doubt, make assumptions and state them upfront. Marks will be deducted for claims without proper reasoning. Write your answers (with all analysis, justification, calculation etc) in loose sheets, scan them and upload in moodle within the allotted time

---

1. A processor has 32KB instruction cache and 32KB data cache. Also, assume the following quantities.
   - 36% of instructions are of type load/store
   - 74% of memory references are instruction references
   - Misses per 1000 instructions:
     1. 16KB instruction cache: 3.82
     2. 16KB data cache: 40.9
   - Hit time: 1 clock cycle
   - Miss penalty: 100 clock cycles

   a. Compute the overall miss rate of this split cache design.
   b. Compute the average memory access time (AMAT).

   [4+4]

2. Observe the instruction sequence given below and list all dependencies of type RAW, WAW and WAR. Instruction format is (operation, destination, source1, source2)

```
I0 : add r1 r2 r3
I1 : sub r3 r1 r2
I2 : add r4 r1 r3
I3 : mul r1 r2 r3
```

   [3.5]

3. Explain clearly what is the advantage of the following code transformation in terms of performance. Assume x is an integer array, block size in cache = 10×sizeof(int), cache type = direct mapped, cache miss latency = 10 ms, hit latency = 1 ms, no of blocks in cache = 10.

```
/* Before */
for (j = 0; j < 100; j = j+1)
       for (i = 0; i < 5000; i = i+1)
              x[i][j] = 2 * x[i][j];


/* After */
for (i = 0; i < 5000; i = i+1)
      for (j = 0; j < 100; j = j+1)
             x[i][j] = 2 * x[i][j];
```

   [3.5]

4. Consider a 1-bit and a 2-bit predictor. In the table below, indicate the prediction for the branch and whether the prediction is correct . For the 1-bit predictor use the notation "T" for taken, and "N" for not taken states. For the 2-bit predictor, use "ST" for strongly-taken state, "WT" for weakly-taken state, "SN" for strongly-not-taken state, and "WN" for weakly-not-taken state. Assuming the first entry in the table, fill in the predictor's state after branch resolution for each of the remaining branch outcomes.

	1 bit predictor			2 bit predictor		
Outcome	**Prediction**	**Next State**	**Correct?**	**Prediction**	**Next State**	**Correct?**
Taken	N	T	No	N	ST	No
Taken						
Not taken						
Not Taken						
Not Taken						
Taken						
Not Taken						
Taken						
Taken						
Not Taken						

[5]

5. Assume an application where the execution of floating-point instructions on a certain processor P consumes 60% of the total runtime. Moreover, let's assume that 25% of the floating-point time is spent in square root calculations.

   a. Based on the initial research, the design team of the next-generation processor $P2$ believes that it could either improve the performance of all floating point instructions by a factor of 1.5 or alternatively speed up the square root operation by a factor of 8. From which design alternative would the aforementioned application benefit the most?

   b. Instead of waiting for the next processor generation, the developers of the application decide to parallelize the code. What speedup can be achieved on a 16-CPU system, if 90% of the code is perfectly parallelized? What fraction of the code has to be parallelized to get a speedup of 10?

[2+3]

6. Assume a 3-bit global register (initialized to all 0's) and a two-level Global–Global (global register and global PHT) scheme.

   a. How many entries are there in the PHT?

   b. Show the contents of the global register(GR) and of the PHT (where each entry is initialized to the weak not-taken case), 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 *actual* the actual outcome as per the given string, show your results for each branch in the table below in the form (GR) (*pred*) (*Updated* GR) (*actual*) (*correct?*) (PHT). Assume a speculative update for GR and a nonspeculative one for the PHT that occurs before the next branch prediction. On misprediction, restore GPR to (001). **[1 + 4]**

GR	***pred***	***Updated* GR**	***actual***	**correct?**	**PHT**
000	N	000	T	No	01, 01, 01, 01, 01, 01, 01, 01