INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR
Mid-Spring Semester 2018-19

High Performance Computer Architecture (CS60003)

Time=2 Hours **Max Marks=50**

Important Instructions:

- **This question paper consists of two sections. You need to answer only one of the sections. All the two Year M.Tech students need to answer only Section A, and all other students need to answer only Section B.**
- **Use of simple non-programmable calculators permitted.**
- **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 (Only two Year MTech students need to answer)

1. Consider a simple MIPS pipeline that does not have any forwarding circuitry. Determine the minimum number of comparators that the pipeline control circuitry must use to determine whether an instruction at the decode stage would cause any data hazard and needs to be stalled. Briefly explain your answer. **[1+1]**

2. Consider that on a processor deploying a simple MIPS pipeline, the following C code is being executed.

```
for (i=0; i<10000; i++)
    for (j=0; j<3; j++){
        p=p+1; q=q+p;}
```

   a) Assume that the pipeline deploys 1-level (non-correlating) 1-bit branch predictors. In this case, what will be the percentage misprediction of the branch corresponding to the inner loop? Assume that the size of the BHT (Branch History Table) is large. **[2]**

   b) Suppose the pipeline incorporates 1-level (non-correlating) 2-bit bimodal predictors. What will be the percentage misprediction of the branch corresponding to the inner loop? Assume that the size of the BHT is large. **[3]**

3. Assume that you are designing a 2-level (correlating) local branch 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. **[5]**

4. Consider the following code segment. It is to be run on a processor deploying a simple MIPS pipeline that does not have any interlocking hardware, however it incorporates full forwarding.

```
loop:   lw      $t2, 0($t1)     # $t2 = A[k]
        add     $t4, $t1, $t2   # $t4 = address of B[k]
        lw      $t5, 0($t4)     # $t5 = B[k]
        add     $t5, $t5, $t3   # $t5 = A[k] + B[k]
        add     $t3,$t1,$r0     # $t3 = $t1
        sw      $t5, 0($t3)     # A[k] = A[k] + B[k]
        addi    $t1, $t1, -4    # k--
        bne     $t1, $t0, loop  # loop termination test
```

   a) Determine the number of cycles required for the loop to execute 1000 iterations. **[2]**

b) Restructure the code to reduce data hazard and also reduce control hazard using the delayed branch technique. Highlight the specific code restructuring done by you. Determine the number of cycles required for the loop to execute 1000 iterations. **[2+1]**

5. Consider a simple MIPS processor that deploys full forwarding. For a specific program being executed on this processor, 20% of the instructions are loads (half of which are followed immediately by a dependent instruction), 20% are branches (75% of which are not taken), the remaining instructions are simple single-cycle arithmetic operations. The processor deploys a static not taken predictor. What is the average CPI of the given program on this processor? **[5]**

6. 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? **[2+2+1]**

```
        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
```

7. Consider a simple 5 stage MIPS pipeline in which a dynamic predictor with target prediction is being used. The programs that run on this processor have on the average 20% branches. Of these, on the average 60% are taken and 40% are not taken. All branch outcomes are predicted with 90% accuracy. The branch target is predicted with 90% accuracy. On account of data hazards, the pipeline stalls 30% of time. What would be the CPI for this processor? Clearly show all steps of your computation. **[5]**

8. A certain 6 stage pipelined processor has two branch delay slots. An optimizing compiler can fill the first slot 80% of the time and can also fill the second slot 20% of the time. The second slot is filled only when the first slot is filled. Of the filled slots, 10% are eventually discarded on account of being taken from the fall through path. 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 on the average. **[5]**

9. 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. **[5]**

10. A team of designers is trying to implement the StoneBear processor based on the simple MIPS pipeline. The team has come up with the following latencies for the pipeline stages: IF: 1.5 ns, ID: 1 ns, EX: 0.75 ns, MEM: 2.25 ns, WB: 1 ns. The latch overhead is negligible. A dynamic branch predictor along with target prediction

has been implemented. Assume that all target predictions are done with 100% accuracy. The processor incorporates a well-designed forwarding circuitry. For a certain program to be run on this processor, it was determined that:

- 1% of the program instructions are mispredicted branches.
- 15% of the program instructions are loads.
- There is 40% chance that the first instruction following a load depends on that load. An additional 20% of the time, the second instruction after the load depends on it.

a) What is the clock cycle time for this design taking data and control hazards into account? **[1]**

b) What is the CPI for this design? **[2]**

c) Later the designers decided to split the MEM stage into two stages: MEM0 and MEM1 with latencies 1.5 ns and 0.75 ns respectively. What is the new clock cycle time? **[1]**

d) What would be the CPI for the modified design taking data and control hazards into account? Show the steps of your calculation. **[2]**

e) Based on your answers above, is it a good idea for the designers to split the MEM stage into MEM0 and MEM1? Justify your answer. **[2]**

PLEASE SEE NEXT PAGE FOR PART B (ONLY TO BE ATTEMPTED BY DUAL DEGREE STUDENTS AND RESEARCH SCHOLARS)

High Performance Computer Architecture (CS60003)

Section B

To be answered by Dual Degree MTechs and Research Scholars

1. Answer the following questions. **[2+3+5]**

   (a) When running an integer benchmark on a RISC machine, the average instruction mix was as follows:

Instructions	**Average Frequency**
Load	26%
Store	9%
Arithmetic	14%
Compare	13%
Conditional Branch	16%
Unconditional Branch	1%
Call/Return	2%
Shift	4%
Logical	9%
Miscellaneous	6%

   The following measurements of average CPI for individual instruction categories were made:

Instruction Type	**Average CPI (clock cycles)**
All ALU instructions	1
Load-store	1.4
Conditional branches:	
Taken	2.0
Not Taken	1.5
Jumps	1.2

   Assume that 60% of the conditional branches are taken and that all instructions in the Miscellaneous category are ALU instructions. What is the CPI of the benchmark on this RISC machine?

   (b) Consider two implementations M1 and M2 of the same ISA. We are interested in the performances of two programs P1 and P2, which have the following instruction mixes:

Instructions	**P1**	**P2**
Load/Store	40%	50%
ALU	50%	20%
Branches	10%	30%

   The CPIs for each machine are:

Instructions	**M1**	**M2**
Load/Store	2	2
ALU	1	2
Branches	3	2

i. Assume that the clock rate of M1 is 2 GHz. What should be the clock rate of M2 so that both machines have the same execution time for P1?

ii. Assume now that both machines have the same clock rate and that P1 and P2 execute the same number of instructions. Which machine is faster for a workload consisting of equal runs of P1 and P2?

iii. Find a workload (using only P1 and P2) that makes M1 and M2 have the same performance when they have the same clock rate.

(c) A processor M-5 has a five-stage pipeline and a clock cycle time of 10ns. A newer implementation of the same instruction set in a processor M-7 uses a seven-stage pipeline and a cycle time of 7.5ns.

i. Consider a loop of five instructions (four arithmetic instructions and a branch to the beginning of the loop). There is a dependency between two consecutive arithmetic instructions in the loop. This dependency induces a 1 cycle stall in M-5 and 2-cycle stall in M-7. The branch induces 2-cycle stall on M-5 and 4-cycle stall on M-7. Which processor executes the loop faster in "steady state" (i.e., you do not have to consider the time it takes to start up the pipeline in the first iteration and/or drain it in the last iteration)?

ii. Assume now that the loop is unrolled once, i.e., instead of n iterations of four instructions and a branch, we have now $n/2$ iterations of eight instructions and a branch. Instead of one data dependency per loop, we now have two data dependencies per unrolled loop. Which processor executes the unrolled loop faster?

2. Answer the following questions. **[1+2+2+5]**

(a) Consider the following equation for execution time of a program in a CPU.

$$EX_{CPU} = Number\ of\ instructions \times CPI \times cycle\ time.$$

Of the three factors which is most influenced by:

i. The Compiler.

ii. The Computer Architect.

(b) With sequential execution occurring 15% of the time, what is the maximum speedup with an infinite number of processor?

(c) Consider an experiment where you have run the SPEC CPU2006 integer benchmark in two different machines and noted execution times of each program in the suite. Indicate which of the (weighted) arithmetic, harmonic, or geometric mean you would use to compare the speeds of the two machines?

(d) You have a system that contains a special processor for doing floating-point operations. You have determined that 60% of your computation time can use the floating-point processor. When a program uses the floating-point processor, the speedup of the floating-point processor is 40% faster than when it does not use it.

i. Calculate overall speedup by using the floating-point processor.

ii. In order to improve the speedup consider two options:

- **Option 1:** Modify the compiler so that 70% of the computation time can use the floating-point processor. Cost of this option is $50K.
- **Option 2:** Modify the floating-point processor. The speedup of the floating-point processor is 100% faster than when it does not use it. Assume in this case that 50% of the computation time can use the floating-point processor. Cost of this option is $60K.

Which option would you recommend? Justify your answer quantitatively.

3. Answer the following questions. **[5+5]**

(a) Consider the following instructions.

```
I1:  R6 ← MEM[R2]
I2:  R7 ← R6 + R4
```

Find out what type of dependency is present in these two instructions. Explain with diagram (including required modifications) how this can be resolved in the standard 5-stage pipelined architecture.

(b) What modifications are necessary for the pipelined design and forwarding paths of the architecture from previous answer (part (a)) if load instructions include auto-increment instructions of the form:

```
I3:  R1 ← MEM[R2+offset]
I4:  R2 ← R2 + c
```

where the two operations are considered to be part of the same instruction and c is a constant that depends on the size of the data that are loaded. Auto-increment loads will have their own opcodes.

4. Answer the following questions. **[5+5]**

(a) Consider the following assembly code instructions, which is executed in a 5-stage pipelined architecture and the branch is resolved at 3^{rd} stage.

```
I1:   LABEL 2:   BEQ R1, R2, LABEL 1
I2:              LD R3, 0(R4)
I3:              ADD R2, R3, 1
I4:   LABEL 1:   DIV R2, R2, 2
I5:              JUMP LABEL 2
```

Fill-up the following table (with Yes or No) for dependency and hazards between different pairs of instructions.

	RAW?	**WAR?**	**WAW?**	**Hazard?**
I1 → I2				
I1 → I3				
I1 → I4				
I2 → I3				
I2 → I3				
I3 → I4				

(b) Consider the following sequence of branch outcomes for a program, where `T` signifies *Branch is Taken* and `NT` signifies *Branch is Not Taken*. The accuracy of prediction is defined as the percentage of guesses which are correct.

```
T, T, NT, NT, T
```

What is the accuracy of prediction considering 2-bit history with four 2-bit counters if the sequence is repeated forever? Assume all the predictors start in the "Strong Not Taken" state.

5. Answer the following questions. **[5+5]**

(a) Show a state diagram for a scheme where the 2-bit counter is replaced by a 3-bit shift register keeping track of the last three executions of the branch and predicting according to the majority decision of these three executions. Is this a reasonable alternative to a 2-bit saturating counter in terms of performance and cost?

```
1000:   BRANCH   <condA>   1018
1004:   BRANCH   <condB>   1014
1008:   BRANCH   <condC>   1014
100C:   ADD
1010:   MUL
1014:   BRANCH   <condE>   1004
1018:   SUB
```

(b) Consider the following pseudo-code snippet (address in hexadecimal, followed by pseudo assembly code).

Suppose we are using a *gshare* predictor with a history of length 2. Suppose also that we have a 32 entry table of counters. Our gshare predictor hashes into this table by taking bits [6:2] of the current PC, and XOR-ing with the current 2-bit history. Construct two paths through the code, ending at different branch instructions, but that cause the predictor to hash to the same table entry. [**Clarification:** History register left shifts, the older history bit gets XOR-ed with PC[3], the newer history bit gets XOR-ed with PC[2].]