EndSem
CSE/ECE 511 Computer Architecture

INSTRUCTIONS:

Total Marks = 50
Time Duration = (1hr and 30mins) solving + 10 mins uploading

1. **The duration of the exam is (1hr and 30mins), and 10 mins for scanning and uploading the solutions. No further extension of time will be given regarding this. Any late submission will be awarded 0 marks.**
2. **The question paper will be uploaded in google classroom. Do not forget to turn it in. Solutions submitted by any other means (email etc.) won't be considered for evaluation.**
3. **Students are required to switch on their cameras and mute themselves. Make sure you are sitting in a well-lit room so that we are able to see your faces clearly. If you are not clearly visible, you will be awarded 0 marks.**
4. **The answers should be in your own handwriting and submission should be in PDF format only.**
5. **Write any assumption clearly, if any. Needless to say, only reasonable assumptions will be considered if any ambiguity is found in the question.**
6. **During the exam, if you have any queries, write them in the meet chatbox. It will be taken into notice by us. Don't unnecessarily unmute your mic for it creates a disturbance to others.**
7. **Calculators are NOT allowed during exam time. ONLY use pen and paper for writing the exam.**
8. **Students need to be present and visible for the whole exam duration (till the end of solution uploading time) even if they upload the solution before time.**
9. **NAMING CONVENTION - <Name>_<Roll number>_EndSem.pdf.**
   Example Abc_Def_2020123_EndSem.pdf
10. **Show your intermediate calculations and justifications in each question.**

Q1. The main memory access latency with no cache requires 200 cycles. By introducing a single L1 cache in the system, access latency reduces to 10 cycles if access results in a hit, otherwise it results in 210 cycles on a miss. Calculate the highest miss rate for L1 cache after which the use of cache will be disadvantageous? **[4 Marks]**

Q2. Consider the code snippet below. Suppose that it is executed on a system with a 2-way set associative 16KB data cache with 32-byte blocks, 32-bit words, and an LRU replacement policy. Assume that int is word-sized. Also assume that the address of 'a' is 0x0, 'i' and 'x' are in registers, and that the cache is initially empty. How many data cache misses are there? How many hits are there? **[8 Marks]**

```
int i;
int a[1024*1024];
int x=0;
for (i=0; i<1024; i+=2){
      x+ = a[i]
}
for(i=0;i<1024;i++){
      x+ = a[1024*i]
}
```

Q3. Consider the following code snippet.

```
      ADDI  R1, R0, n
      ADDI  R2, R0, 0
loop:
      LD    R3, A[ R2 ]
      BEQZ  R3, if
      ADDI  R3, R3, 2
      ST    R3, A[ R2 ]

If:
      ADDI  R2, R2, 4
      ADDI  R1, R1, -1
      BNEZ  R1, loop
```

Consider R0 always stores 0. A[R2] points to a location R2 in an array A. Consider a ONE-entry BHT, where each entry in the BHT has a 2-bit counter. The counter is initialized to 00

state in the beginning. The working of the 2-bit counter is as follows: The state of the counter decides whether the branch is predicted as 'TAKEN (T)' or 'NOT TAKEN (N)'.If the branch is actually TAKEN, the counter increments to the next state. If the branch is actually NOT TAKEN, the counter decrements to the previous state.

State No.	State	Prediction
0	00	Not Taken
1	01	Not Taken
2	10	Taken
3	11	Taken

Consider the array elements to be {5, 4, 3, 2, 1} and n= 5. What is the prediction accuracy for all the branches present in the code snippet for 5 loop iterations. Show the intermediate steps. **[10 Marks]**

Q4.

High Level Code-

```
For (int I = 0 ; i <100; i++){
        D[i] = A[i] + B[i] * C ;
}
```

Corresponding Assembly Code-

```
Addi   R2, R0, 100
Addi.f FC, F0 , C
Addi   R1, R0, 0
Loop :
Ld     F1, 0(RA)
Ld     F2, 0(RB)
Mul.f  F3, F2 , FC
Add.f F4 , F1, F3
St     F4, 0(RD)
Addi  RA, RA, 4
Addi  RB, RB, 4
Addi  RD, RD, 4
Addi  R1 , R1, 1
Bne   R1, R2, Loop
```

You are given a VLIW instruction of the following format in the table below along with the latency of each type of operation in parenthesis. Assume LEQ type scheduling. For instructions with more than 1 cycle latency, assume registers are read in the first cycle itself. Assume R0 and F0 hard-coded to 0. RA, RB and RD store the Starting addresses of Arrays A, B , D.

Int (1)	**Int (1)**	**Add.f(3)**	**Mul.f(4)**	**MEM (4)**	**MUL(3)**

a. Schedule the above mentioned code on a VLIW processor considering the given latencies without unrolling or pipelining. **[5 Marks]**
b. Now, Unroll the code 2 ways and schedule the resulting program . **[5 Marks]**

Clearly mention the entire instruction inside the box while scheduling.

Q5. Assume an (IO2I) processor which automatically predicts branches to be not taken. Branches are resolved in the execution stage. In case of conflict after Branch X0 stage, squash invalid instructions when branch commits. Assume full bypassing
Assume Multiplication takes 4 cycles(Y0, Y1, Y2, Y3), Other arithmetic operations and branches take 1 cycle (X0). Assume all registers initialized with its index number (R0 → 0, R1 → 1 and so on).

a) Draw a pipeline timing diagram. **[9 Marks]**
b) Mention the number of cycles . **[1 Mark]**

BGE X , Y , L → goto L if R[X] >=R[Y]

Code Snippet:

```
1:    ADD R1, R0, R2
2:    MUL R3, R2, R4
3:    ADDI R3 , R3 , -3
4:    ADD R7, R8, R9
5:    BGE R3, R4, label
6:    MUL R14, R10, R11
7:    ADD R5, R14, R6
8:    MUL R3. R1, R2
9:    BEQZ R1, label
10:   ADDI R5, R5, 1
11:   MUL R5, R11, R6
12:   label: MUL R8, R3, R4
13:   ADD R1, R2, R3
```

Q6. Consider a single issue out-of-order processor with split Instruction window/ ROB design and has the following pipeline stages: Fetch(F), Decode/Rename (D), Issue (I), Register File read (R), execute (X), Writeback(W).

- The ALU operations take 1 cycle, Memory operations take 3 cycles, Floating point multiply takes 4 cycles, floating-point add takes 2 cycles.
- There are three parallel execution units: 1) Floating point add 2) Floating point multiply. 3) ALU and memory operations.
- There are two domains of registers: integer registers (R1,R2…) and floating-point registers(F1,F2).
- The Fetch and Issue stages have buffers to hold infinite no. of instructions to get executed.
- The instruction will exit the issue stage only when there is no WAR or WAW hazard. New instructions must wait in Issue stage until older instructions finish reading from the register file.
- No bypassing between functional units.
- You are allowed to perform register renaming and there are unlimited register renaming resources.

```
1:    ADD   R4,   R2,   R3
2:    LD.f  F1,   0(R1)
3:    MUL.f F2,   F3,   F1
4:    LD.f  F1,   0(R2)
5:    SUB.f F4,   F2,   F1
6:    ST.f  F4,   0(R2)
7:    ADDI  R2,   R2,   3
8:    SUBI  R4,   R1,   4
```

The instruction with ".f" indicates floating point instruction. Draw the pipeline timing diagram for the above code snippet. **[8 Marks]**