1. **(2 points)** For the following code use (1,2) branch predictor to predict branch outcome for b1 and b2. This means that you will be using a correlating predictor that depends on the outcome of 1 previous branch and this predictor will have 2 bits. Also assume that the values of R1 at the entry point of the code are 0, 2, 0, 2, 0, 2 etc. Show the values for both b1 and b2 predictors and note how many misses you have. Assume that start values are 00/00.

   b1:  BZ R1, AL
        ADD R1, R1, #200
   AL:  ADD R1, R1, #4
        LD R3, 100(R1)
        MUL R4, R3, R5
        STORE R3, 100(R1)
        SUB R1, R1, #4

   b2:  BNEZ R1, Loop
        ADD R3, R3, #100
   Loop: SUB R3, R3, #100

2. **(3 points)** Assume a pipeline with stages IF, ID, ALU1, MEM, ALU2, WB. Branch target address is calculated in ALU1 stage and branch outcome is calculated in ALU2 stage. We are using a branch target buffer for branch prediction.
   a. **(0.5 points)** Draw BTB fields and explain how the lookup is done and what is in each field
   b. **(0.5 points)** If BTB is using branch folding, how do fields in BTB change?
   c. **(0.5 points)** What is the penalty for taken branches and a correct prediction with and without branch folding?
   d. **(0.5 points)** What is the penalty for taken branches and an incorrect prediction with and without branch folding?
   e. **(0.5 points)** What is the penalty for not-taken branches and a correct prediction with and without branch folding?
   f. **(0.5 points)** What is the penalty for not-taken branches and an incorrect prediction with and without branch folding?

3. **(4 points)** Show the scheduling of three iterations of the following loop using Tomasulo’s algorithm with and without speculation. The schedules should fit on one sheet of paper each. Note the CPI for each schedule. Assume that there are two CDBs, two instructions can be issued per clock cycle, and there is one ALU both for integer ALU instructions and for address calculation. It takes 1 EX cycle for integer ALU, and 2 EX cycles for load and store (1 for address calculation and 1 for memory access).

   Loop: LD R2, 0(R1)
        DADDI R2, R2,#1
        SD R2, 0(R1)
        DADDI R1, R1, #4
        BNEZ R2, Loop