1. Indicate all of the data hazards and branch delays in the above program.

And R5,R3,#1 ; test for even or odd

Data hazard since R3 might not be written to memory/register

Branch\_if\_[R5]=0 EVEN

delayed to read R5 because still writing the actual value for R5

Multiply R3,R3,#3 ; Calculate 3x+1

if using delay and not needed, changes R3 for Hazard

Add R3,R3,#1

Wait for previous instruction to be done to get correct R3

ENDLP: Add R6,R6,#1 ; increment counter

data hazard since R3 might not be updated

Branch\_if\_[R3]!=[R4] LP ;

delayed to get correct R3 value

1. Suppose we were running on a computer with a branch delay slot. Write a modified version of the program that exploits the delay slots as much as possible.

Move R3,#20 ; set an initial value

Move R4,#1 ; end condition

Move R6,#1 ; set counter

LP: And R5,R3,#1 ; test for even or odd

Branch\_if\_[R5]=0 EVEN

; the number is currently odd, so set it to 3x+1

DoNothing ; delay cannot be used since it changes the needed value

Multiply R3,R3,#3 ; Calculate 3x+1

Add R3,R3,#1

Branch ENDLP

; the number is currently even, so set it to x/2

EVEN: ShiftRight R3,R3,#1 ; divide by 2

; count the # of times we did the loop, see if we are done

ENDLP: Add R6,R6,#1 ; increment counter

Branch\_if\_[R3]!=[R4] LP ; loop back up if R3 is not 1

Store R6,COUNT ; save counter. This can be done since it will not hurt the program

to do so

1. Suppose the machine has a four-state branch prediction algorithm (Figure 6.12b) that initially starts in the LNT state. For each time a conditional branch is reached, is the prediction correct, and what is the new state of the predictor? Assume there is a separate predictor for each conditional branch instruction.
   1. Even Predictor
      1. LNT 20
      2. False, new is ST 20
      3. True, new is ST 10
      4. False, new is LT 5
      5. True, new is ST 16
      6. True, new is ST 8
      7. True, new is ST 4
      8. True, new is ST 2
   2. LP Predictor
      1. LNT
      2. True, new is SNT 10
      3. True, new is SNT 5
      4. True, new is SNT 16
      5. True, new is SNT 8
      6. True, new is SNT 4
      7. True, new is SNT 2
      8. False, new is LNT 1
2. How much faster would this program run on the superscalar pipeline in Figure 6.13 compared to a single five-stage pipeline?
   1. Assuming the superscalar pipieline only has the drawn connections, the buffers and no forwarding or passing between the ALU pipeline, then the pipeline will only save a few clock cycles on the store portion. This is because so much of this program is dependent on previous steps to calculate numbers, and without being able to communicate the correct number to another instruction, the pipeline will have to be cycled through. Also, the program only uses one store and no load operations, so the store/load portion of the ALU cannot be used very often. And even if it could, the arithmetic operations are dependent on the results of a load being written to memory to obtain the correct, updated information.