1.

```
lw r2,0(r1)
label1: beq r2,r0, label2 # not taken once, then taken
lw r3,0(r2)
beq r3,r0,label1 # taken
add r1,r3,r1
label2: sw r1,0(r2)
```

4.14.1 [10] <§4.8> Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.

	1	2	3	4	5	6	7	8	9	10	11	12
lw	F	ID	Ex	Mem	Wb							
beq		F	ID	Ex	Mem	Wb						
sw			F	ID								
lw					F	ID	Ex	Mem	Wb			
beq						F	ID	Ex	Mem	Wb		
beq							F	ID	Ex	Mem	Wb	
sw								F	ID	Ex	Mem	Wb

4.14.2 [10] <§4.8> Repeat 4.14.1, but assume that delay slots are used. In the given code, the instruction that follows the branch is now the delay slot instruction for that branch.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lw	F	ID	Ex	Mem	Wb										
beq		F	ID	Ex	Mem	Wb									
lw			F	ID	Ex	Mem	Wb								
beq					F	ID	Ex	Mem	Wb						
add						F	ID	Ex	Mem	Wb					
beq								F	ID	Ex	Mem	Wb			
lw									F	ID	Ex	Mem	Wb		
sw											F	ID	Ex	Mem	Wb

4.14.3 [20] <§4.8> One way to move the branch resolution one stage earlier is to not need an ALU operation in conditional branches. The branch instructions would be “bez rd,label” and “bnez rd,label”, and it would branch if the register has and does not have a zero value, respectively. Change this code to use these branch instructions instead of beq. You can assume that register R8 is available for you to use as a temporary register, and that an seq (set if equal) R-type instruction can be use"

```
lw r2,0(r1)

label1:
seq r8, r2, r0
bnez r8, label2 # not taken once, then taken

lw r3,0(r2)
seq r8, r3, r0
bnez r8, label1 # taken
add r1,r3,r1

label2: sw r1,0(r2)
```

4.16 This exercise examines the accuracy of various branch predictors for the following repeating pattern (e.g., in a loop) of branch outcomes: T,NT, T, T,NT

4.16.1 [5] <§4.8> What is the accuracy of always-taken and always-not-taken predictors for this sequence of branch outcomes?

Always: 3/5 accuracy

Never: 2/5 accuracy

4.16.2 [5] <§4.8> What is the accuracy of the two-bit predictor for the first 4 branches in this pattern, assuming that the predictor starts off in the bottom left state from Figure 4.63 (predict not taken)?

```
T  -----  T                  3  -----  2
          |                            |
          |                            |
NT ---- NT                   0  -----  1
```

First is T, predicts NT - wrong, to state 1
Second is NT, predicts NT – right, to state 0
Third is T, predicts NT – wrong, to state 1
Fourth is T, predicts NT – wrong, to state 2

1/4 accuracy

4.16.3 [10] <§4.8> What is the accuracy of the two-bit predictor if this pattern is repeated forever?

Fifth is NT, predicts T – wrong, to state 1
Sixth is T, predicts NT – wrong, to state 2
Seventh is NT, predicts T – wrong, to state 1
Eighth is T, predicts NT – wrong, to state 2
Ninth is T, predicts T – right, to state 3
Tenth is NT, predicts T – wrong, to state 2

…. predicts T forever now

Above accuracy is 2/10 or 1/5.

The limit to infinity would round to 3/5 accuracy as the infinite number of the remaining would average to a 3/5 accuracy since it always guesses true. This out weighs the 12 before.