Q1.

```
Loop: lw    x1, 0(x2)     1
      Addi  x1, x1, 1     2
      Sw    x1, 0(x2)     3
      Addi  x2,x2,44
      Sub   x4,x3,x2      5
      Bnz   x4, Loop      6

      lw    x1, 0(x2)     1+
      Addi  x1, x1, 1     2+
```

Clock	Fetch	Decode	Execute	Memory	Write-Back
1	1				
2	2	1			
3	2		1		
4	2			1	
5	2			1	
6	2			1	
7	2				1
8	3	2			
9	4	3	2		
10	5	4	3	2	
11	5		4	3	2
12	5		4	3	
13	5		4	3	
14	5			4	3

15	5				4
16	6	5			
17	6		5		
18	6			5	
19	6				5
20		6			
21			6		
22			6		
23	1+			6	
24	2+	1+			6
25	2+		1+		

Assuming

- No data forwarding.
- 1 Extra cycle (total 2) for Branch Execute stage
- 2 Extra cycle (total 3) for LW and SW

a) We will lose 3 clock cycles if there is no branch predictor.
b) We will lose only 1 clock cycles if we have a static branch predictor because we will still need to wait for decode. If we have the branch target buffer entry then waiting till decode will suffice.
c) Dynamic branch predictor makes the decision based on history rather than on the branch instruction. So there will be no clock cycle loss.

Q2.

Assuming 1 clock cycle per pipeline stage.
Number of clock cycle per second = 3.4*1000*1000

A)
So for every 100 instructions:

- 80 takes 1 cycle (non-Branch)
- For the 20 branch instructions:
  - 20*0.8 (80% of the cases) = 16 instructions: We will get 2 cycle stall (because we will have to wait to EX to finish before we know that we made a mistake)
  - For the remaining 20*.2 (20% of the cases) = 4 instructions: We will get 1 cycle stall due to correct prediction (We still must wait till decode phase is done for static branch predictor to make any predictor)

So total cycles for 100 instructions is: 80 + (16 * (2+1)) + (4 * (1+1)) = 80 + 48 + 8 = 136

So throughput is 100 instructions every 136 clock cycle
Or 100 instructions every (136/(3.4*1000*1000)) sec
So in 1 sec we can finish 100/ (136/(3.4*1000*1000)) instructions
= ((3.4*1000*1000) * 100) / 136 instructions
= 2,500,000 instructions

B)
So for every 100 instructions:

- 80 takes 1 cycle (non-Branch)
- For the 20 branch instructions:
  - 20*0.8 (80% of the cases) = 16 instructions: We will get 1 cycle stall (because we will have to wait to ID to finish before we know that we made a mistake)
  - For the remaining 20*.2 (20% of the cases) = 4 instructions: We will get 1 cycle stall due to correct prediction (We still must wait till decode phase is done for static branch predictor to make any predictor)

So total cycles for 100 instructions is: 80 + (16 * (1+1)) + (4 * (1+1)) = 80 + 32 + 8 = 120

So throughput is 100 instructions every 120 clock cycle
Or 100 instructions every (120/(3.4*1000*1000)) sec
So in 1 sec we can finish 100/ (120/(3.4*1000*1000)) instructions
= ((3.4*1000*1000) * 100) / 120 instructions
= 2,833,333 instructions

Q3:

```
1          ADD R2, R1, R1
2  Loop:   BNEZ R2, If2
3          ADDI R2, R1, #2
4  If2:    ADDI R2, R1, #-1
5          JUMP Loop
6  Done:
```

Assuming:

00 -> Strong Not Taken
01 -> Weak Not Taken
10 -> Weak Taken
11 -> Strong Taken

The behavior is dependent on the value of R1. Assuming R1 = 2. R2 will have 4.

Assuming a shared branch predictor for all branches.

- As the program hits 2, we get 1 mis-prediction and state changes to 01 => R2 is 4
- As the program hits 2 again we get our 2nd mis-prediction and the state changes to 10 => R2 is 3
- After this we get 1 correct prediction and state changes to 11=> R2 is 2
- Another correct prediction => R2 is 1
- One mis prediction => R2 is 0. Our state changes to 10
- We execute line 3 and 4 (considering we fall through), We end up having R1-1 in R2. So R2 = 1.
- Then we make correct prediction and state goes back to 11 => R2 is 1
- Then we make a wrong prediction state goes to 10 => R2 is 0
- We execute line 3 and 4 (considering we fall through), We end up having R1-1 in R2. So R2 = 1.
- Then we make correct prediction and state goes back to 11 => R2 is 1 ● This goes on….

So given our assumption we will have:

- 2 initial misprediction

- 2 initial correct prediction
- Then alternating correct and wrong prediction

So in a long run…. We will have approx 50% prediction accuracy. THe accuracy will improve if the value of R1 is bigger.

Given an arbitrary value of R1...say X. Assuming Y repetitions of the repeating sequence.

We will have:

- 2 initial misprediction
- (2X-2) correct prediction
- Then Y * (1) mis-prediction
- And, Y* (X-1) correct prediction

Prediction accuracy = ((2X - 2) + (Y* (X-1)))/ (2 + 2X - 2 + Y + YX -Y) If
Y is big…. We converge to: (Y* (X-1))/ (Y + YX -Y)
= (YX - Y) / (YX)
= 1 - (1/X)