4.13

Without forwarding and stalling:

```
add r5,r2,r1: IF ID EX MEM WB
                  NOP -   -   -   -
                      NOP -   -   -   -
lw r3,4(r5):              IF  ID  EX MEM WB
lw r2,0(r2):                  IF   ID  EX MEM WB
                                   NOP -   -   -   -
or r3,r5,r3:                           IF  ID  EX MEM WB
                                           NOP -   -   -   -
                                               NOP -   -   -   -
sw r3,0(r5):                                       IF  ID  EX MEM WB
```

In this case, we need a total of 5+5-1+5=14 cycles. 2 NOP between add and lw to solve r5 dependency, 1 NOP between lw and or solve r3 dependency. 2 NOP between or and sw to solve r3 dependency.

With forwarding logic:

```
add r5,r2,r1: IF ID EX MEM WB
lw r3,4(r5):    IF ID  EX  MEM WB
lw r2,0(r2):       IF   ID  EX  MEM WB
or r3,r5,r3:          IF   ID  EX   MEM  WB
sw r3,0(r5):             IF  ID    EX    MEM  WB
```

In this case, 1 forward from end of EX in add to begin of EX in lw. 1 forward from end of MEM in lw to begin of EX in or. 1 forward from end of EX to begin of EX in sw. We need a total of 5+5-1=9 cycles.

4.14

Original code segement:

```
label 1:
lw r2,0(r1)
beq r2,r0,label2 # not taken once, then taken
lw r3,0(r2)
beq r3,r0,label1 # taken
add r1,r3,r1
label 2: sw r1,0(r2)
```

First Part:
Branch get executed in EX, get resolved in MEM stage, IF takes place after MEM:

lw r2,0(r1): IF ID EX MEM WB
beq r2,r0,label2: IF ID ID EX MEM WB
label 2: sw r1,0(r2): IF IF ID EX - -
lw r3,0(r2): IF ID EX MEM WB
//penalty for mis-prediction
(Repeat the last line due to formatting issue)
lw r3,0(r2): IF ID EX MEM WB
beq r3,r0,label1: IF ID ID EX MEM WB //predict taken
lw r2,0(r1): IF IF ID EX MEM WB
beq r2,r0,label2: IF ID ID EX MEM WB
label 2: sw r1,0(r2) IF IF ID EX MEM WB

Second part:
Branch get resolved in EX stage, IF takes place after EX:

lw r2,0(r1): IF ID EX MEM WB
beq r2,r0,label2: IF ID ID ID EX MEM WB
label 2: sw r1,0(r2): IF IF IF ID - - -
lw r3,0(r2): IF ID EX MEM WB
//penalty for mis-prediction
(Repeat the last line due to formatting issue)
lw r3,0(r2): IF ID EX MEM WB
beq r3,r0,label1: IF ID ID ID EX MEM WB //predict taken
lw r2,0(r1): IF IF IF ID EX MEM WB
beq r2,r0,label2: IF ID ID ID EX MEM WB
label 2: sw r1,0(r2) IF IF IF ID EX MEM WB

Third Part:
Branch get resolved in ID stage, IF takes place after ID:

lw r2,0(r1): IF ID EX MEM WB
beq r2,r0,label2: IF ID ID ID EX MEM WB
label 2: sw r1,0(r2): IF IF IF - - - -
lw r3,0(r2): IF ID EX MEM WB
//penalty for mis-prediction
(Repeat the last line due to formatting issue)
lw r3,0(r2): IF ID EX MEM WB
beq r3,r0,label1: IF ID ID ID EX MEM WB //predict taken
lw r2,0(r1): IF IF IF ID EX MEM WB
beq r2,r0,label2: IF ID ID ID EX MEM WB
label 2: sw r1,0(r2) IF IF IF ID EX MEM WB

4.16.1
Always taken: 3/5= 60%
Always not taken: 2/5=40%

4.16.2
Two bit predictor go in the path: 00 -> 01 ->00 -> 01-> 10
Only 1 prediction is correct in first 4 branches, accuracy is 25%.

4.16.3
If we do the prediction after 10 times, the predictor will go in a cycle: 10 -> 11 ->10 ->11 ->11 . This cycle has accuracy of 60%. So if the pattern repeats forever the accuracy will approximate 60%.