Aidan Lopez

Assignment: Pipeline Hazards

Instructor: Elaheh Sadredini

elaheh@cs.ucr.edu

University of California, Riverside

Grading (total 4.2 points)

- Question 1: 0.8 points
- Question 2: 0.4 points (each part 0.2 point)
- Question 3: 0.4 points
- Question 4: 2 points (each part 0.4 point)
- Question 5: 0.4 points
- Question 6: 0.2 points

Question 1: Dependencies

Find the data dependencies between all the instructions below. You answer should be brief.

- For example: I2 has a data dependency with I1 in R2.

I1. sub R2, R5, R4

I2. add R4, R2, R5

I3. lw R5, 100(R2)

I4. sub R2, R5, R4

I5. sw R2, 101(R2)

Answer

- Write answers in the format of the given example

I2 has a dependency with I1 in R2. I3 has a dependency with I1 in R2. I4 has a dependency with I3 and I2 with R5 from I3 and R4 from I2. I5 has a dependency with I4 in R2.

Question 2: EX forwarding paths

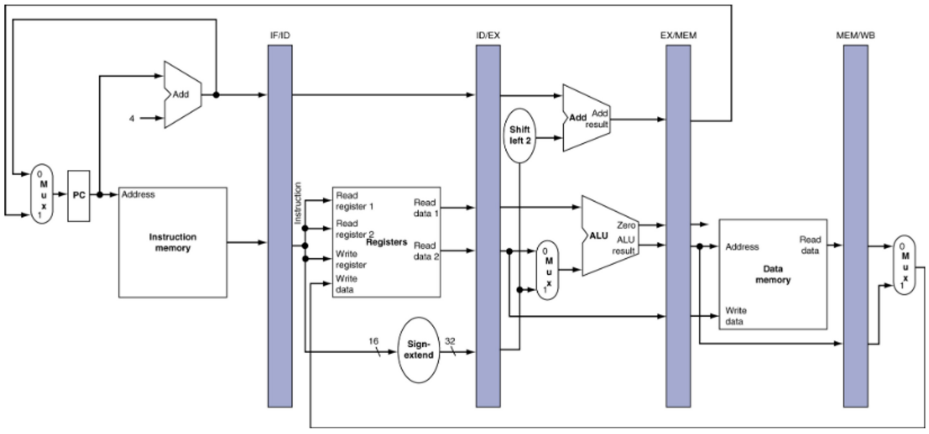
A. Add the necessary forwarding paths for the following instructions in the datapath shown here.

A

1 add R3, R3, R4

2 sub R4, R2, R3

3 add R4, R5, R5



Question 2: EX forwarding paths

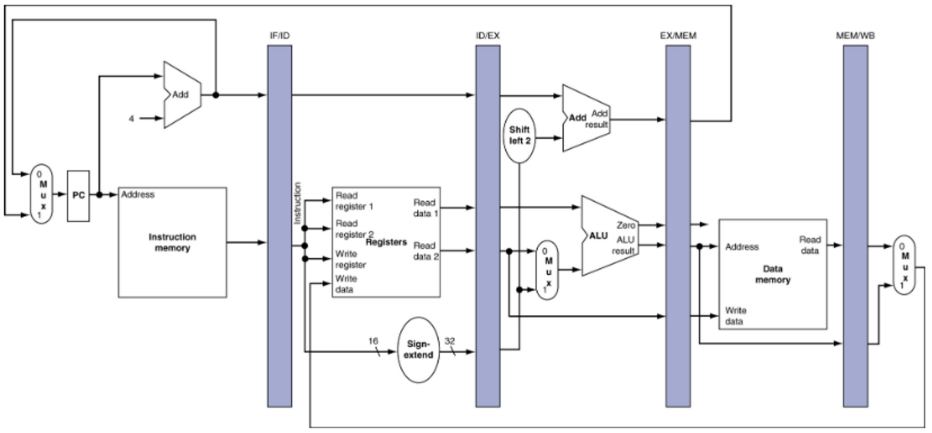
B. Add the necessary forwarding paths for the following instructions in the datapath shown here.

B

1 add R3, R3, R4

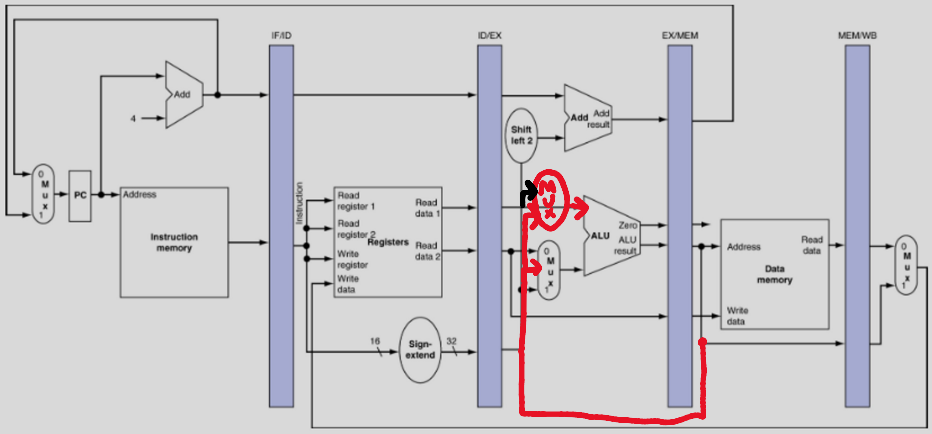
2 sub R4, R2, R4

3 add R4, R5, R3



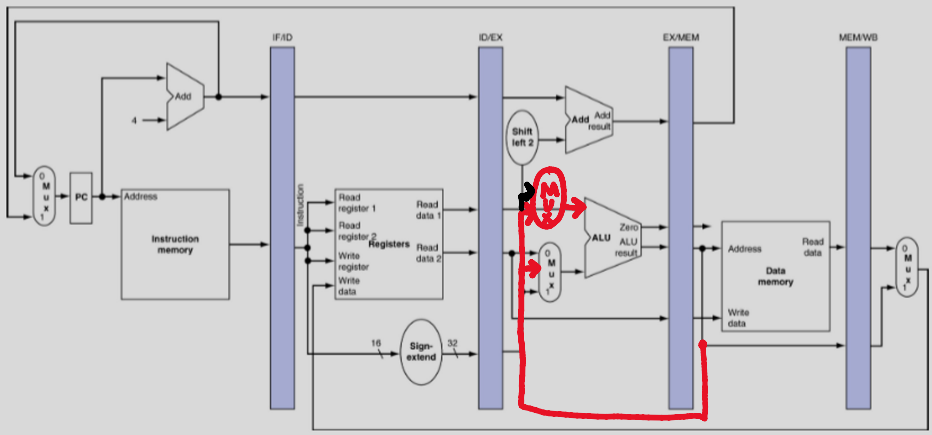
Answer – Part A

- Add the forwarding datapath for part A here



Answer – Part B

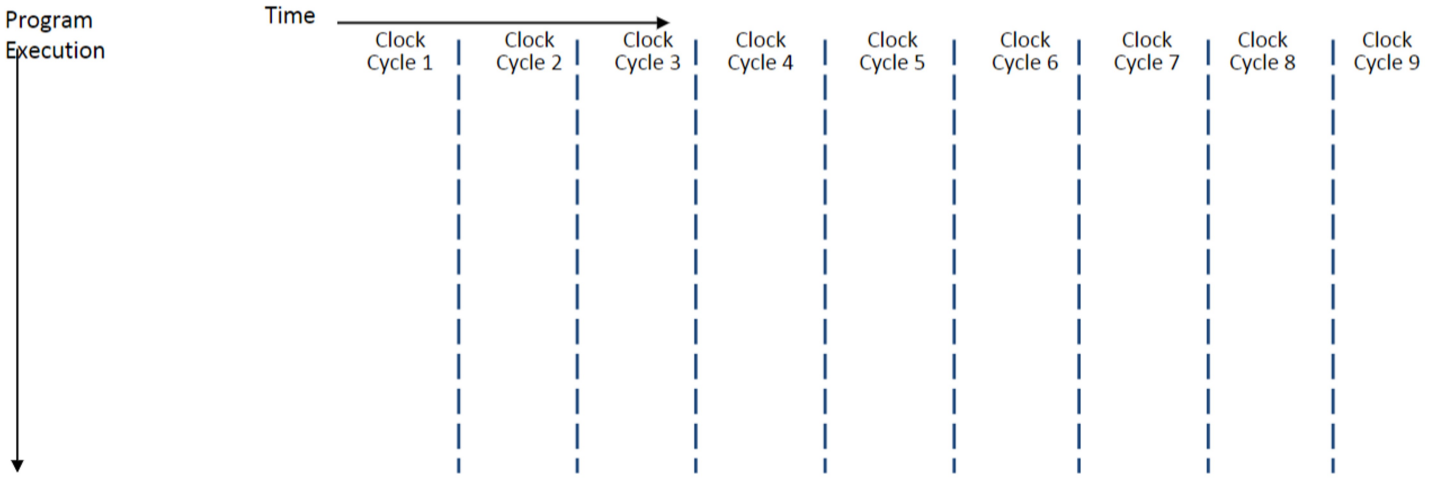
- Add the forwarding datapath for part B here



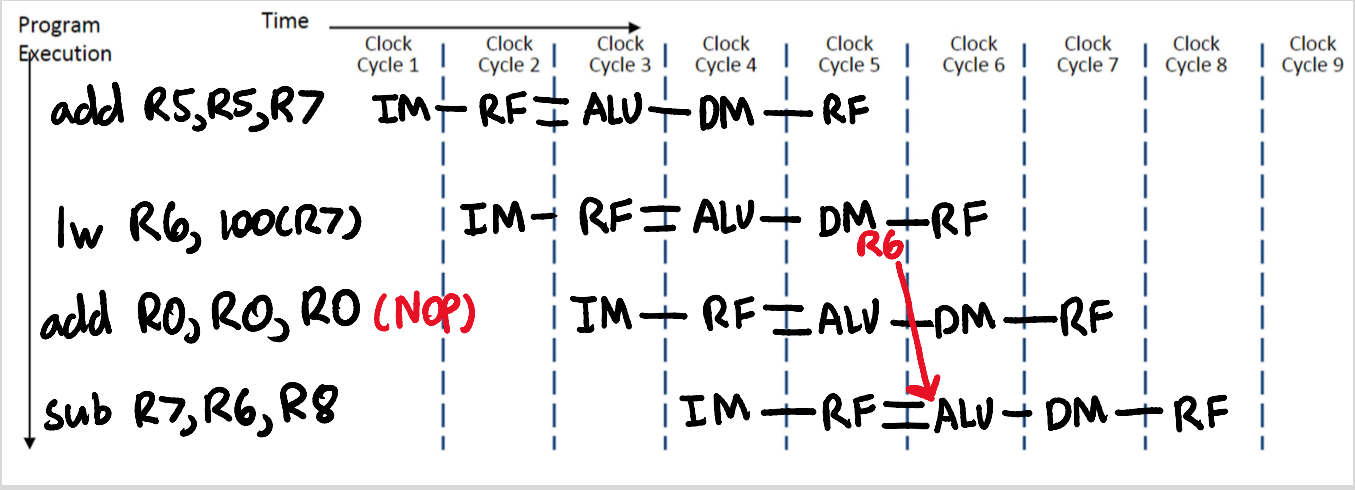
Question 3: Performance with stalls

How long will the following code take on a 5-stage MIPS pipeline with forwarding and appropriate NOPs? A. Fill out the pipeline and draw the forwarding path in the pipeline. B. How many cycles it takes to execute all three instructions?

```
add  R5, R5, R7
lw   R6, 100(R7)
sub  R7, R6, R8
```



Answer



B. It will take 8 clock cycles.

Question 4: Load-to-use delay

The code below was written without regard for the delay slots.
(There may be NOPs missing. The pipeline has a double-pumped register file and forwarding.)

```
lw   R15, 0(R2)
add  R14, R15, R15
lw   R16, 4(R2)
add  R17, R16, R16
```

Answer Part A, B, C, D, and E.

Question 4: Load-to-use delay

A) Identify dependencies

```
lw   R15, 0(R2)
add  R14, R15, R15
lw   R16, 4(R2)
add  R17, R16, R16
```

B) Insert NOPs to resolve the dependencies

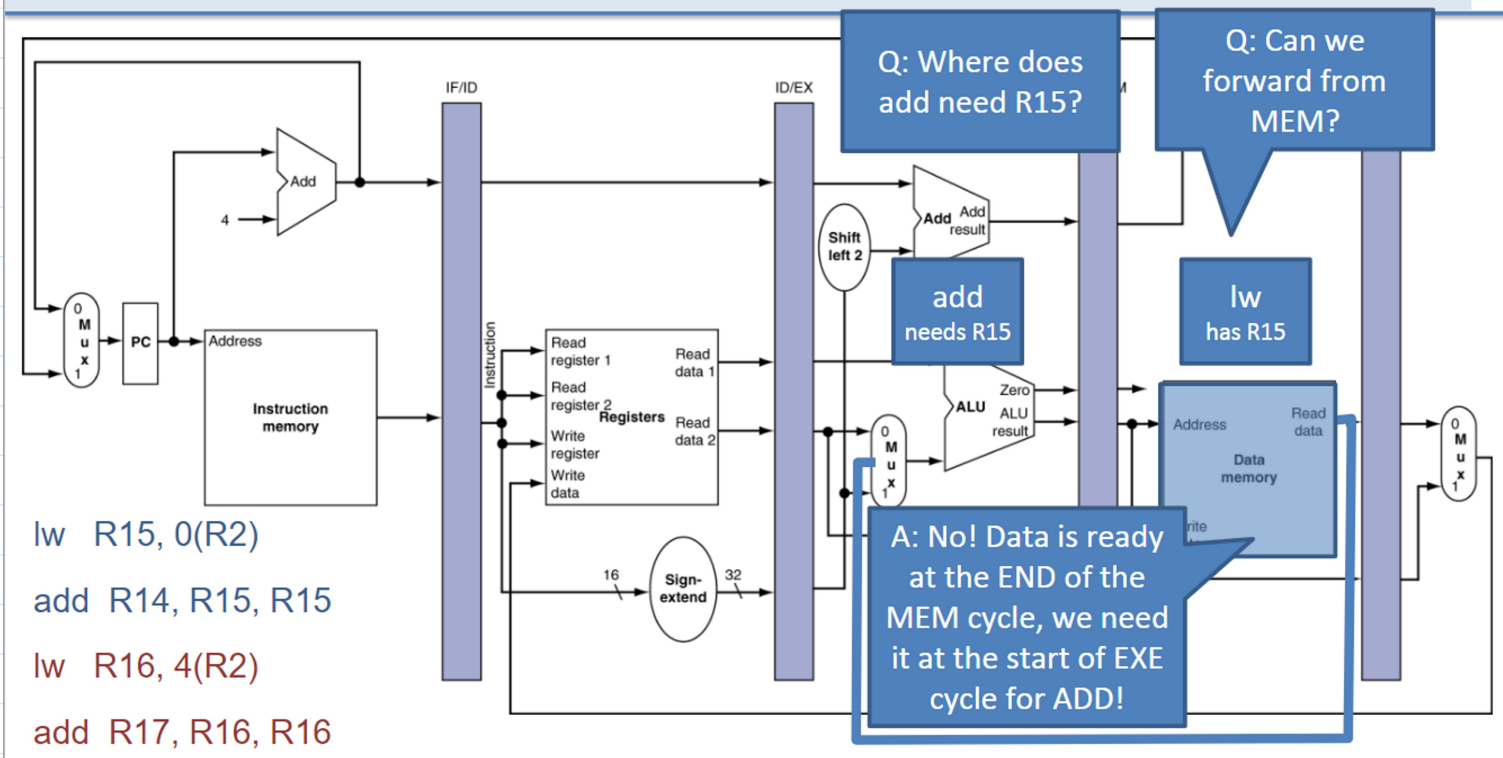
C) After adding NOPs in Part B, add the instructions and draw the forwarding path(s) to the pipeline.

Question 4: Load-to-use delay

D) We would like to keep the pipeline full as much as possible and improve the performance. Can we reorder the instructions to avoid NOPs? If so, how?

E) If the answer to Part D is yes, then add the reordered instructions and draw the forwarding path(s) in the pipeline.

Hint: Load-to-use delay



Answer

- Enter answer for Part A

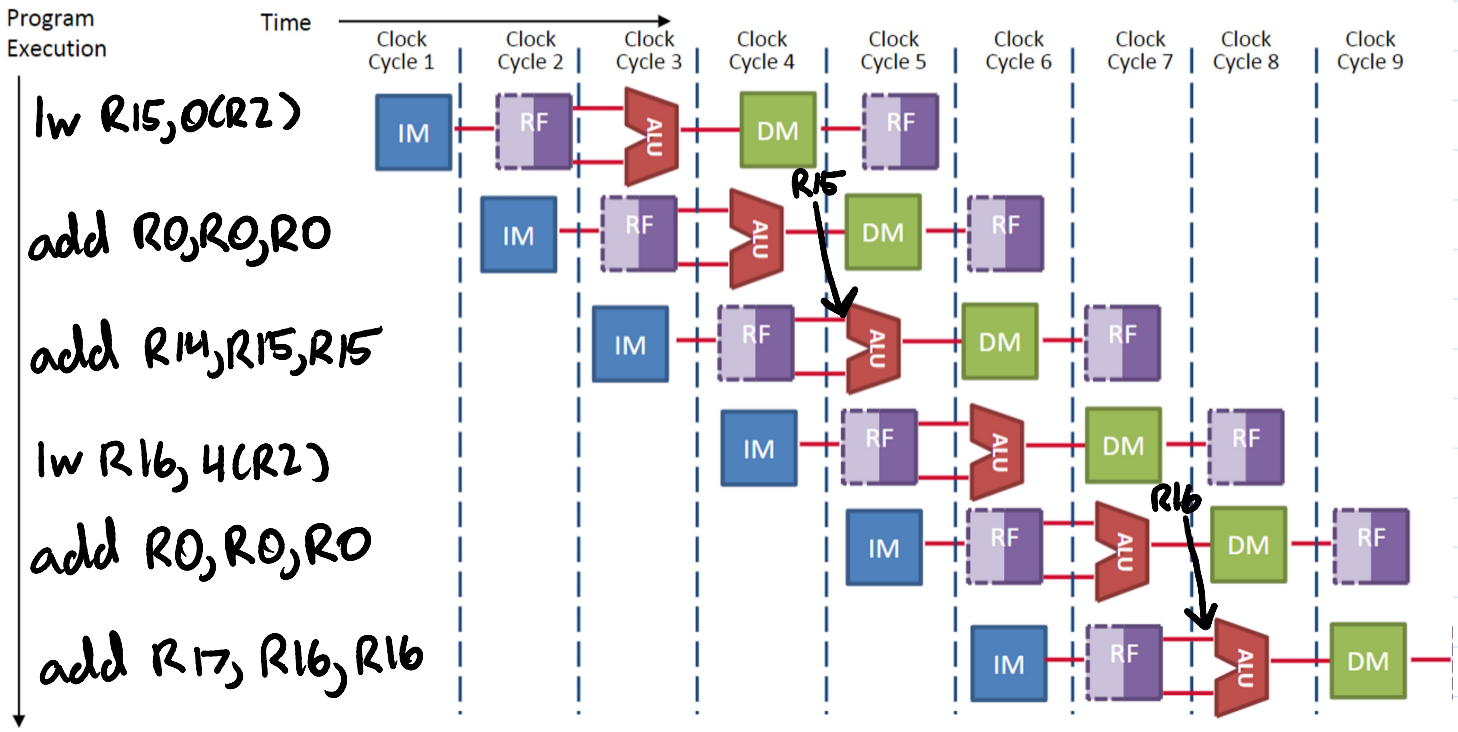
I2 has a dependency with I1 in R15. I4 has a dependency with I3 in R16.

Answer

- Enter answer for Part B

```
lw R15, 0(R2)
add R0, R0, R0
add R14, R15, R15
lw R16, 4(R2)
add R0, R0, R0
add R17, R16, R16
```

Answer for Part C

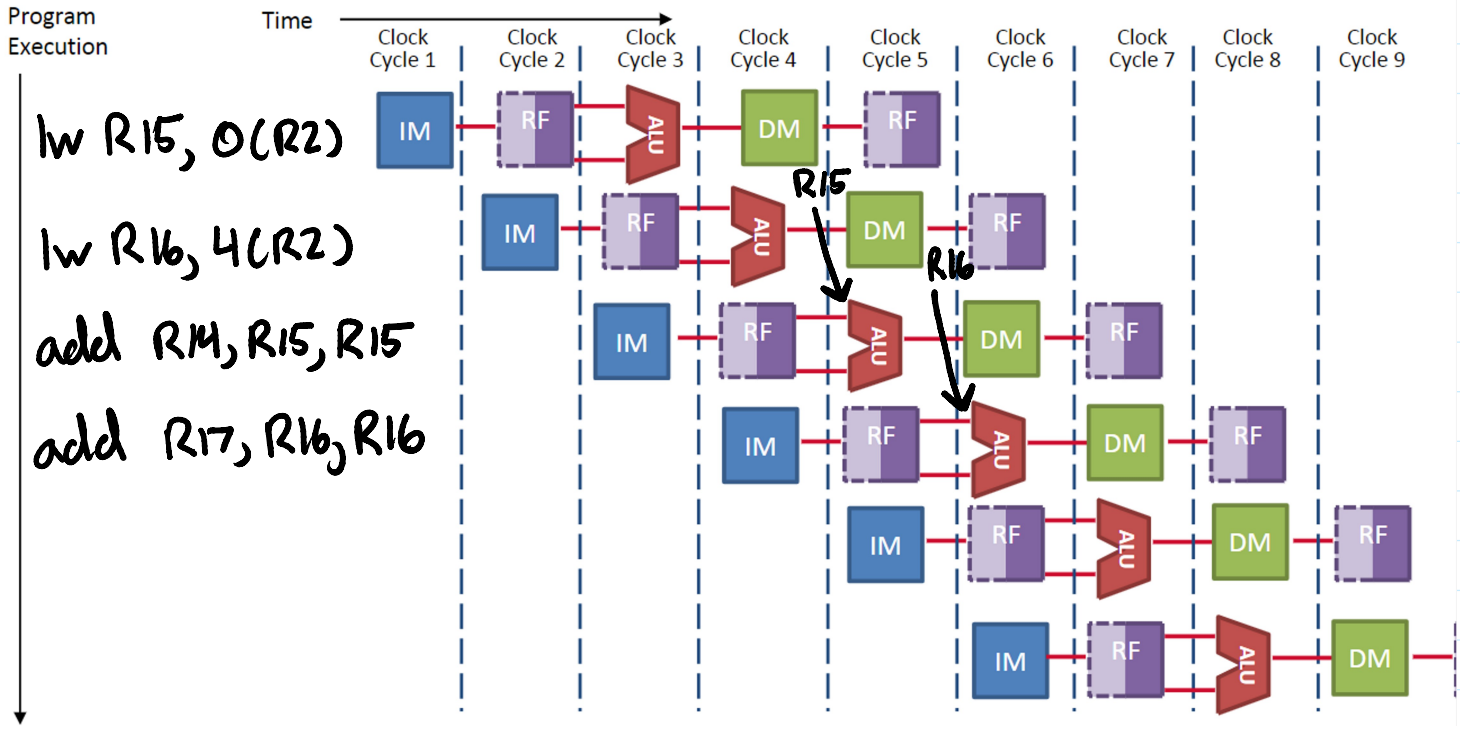


Answer

- Enter answer for Part D

Yes, we can reorder the instructions to avoid using NOPs by having the lw instructions first and the add instructions after.

Answer for Part E



Question 5: MEM-to-MEM stalls

Fill in the pipeline for the following code assuming forwarding and double-pumped registers. Show the forwarding path(s) in the pipeline. If there is a dependency that cannot be resolved with forwarding, then add **bubbles** to the pipeline.

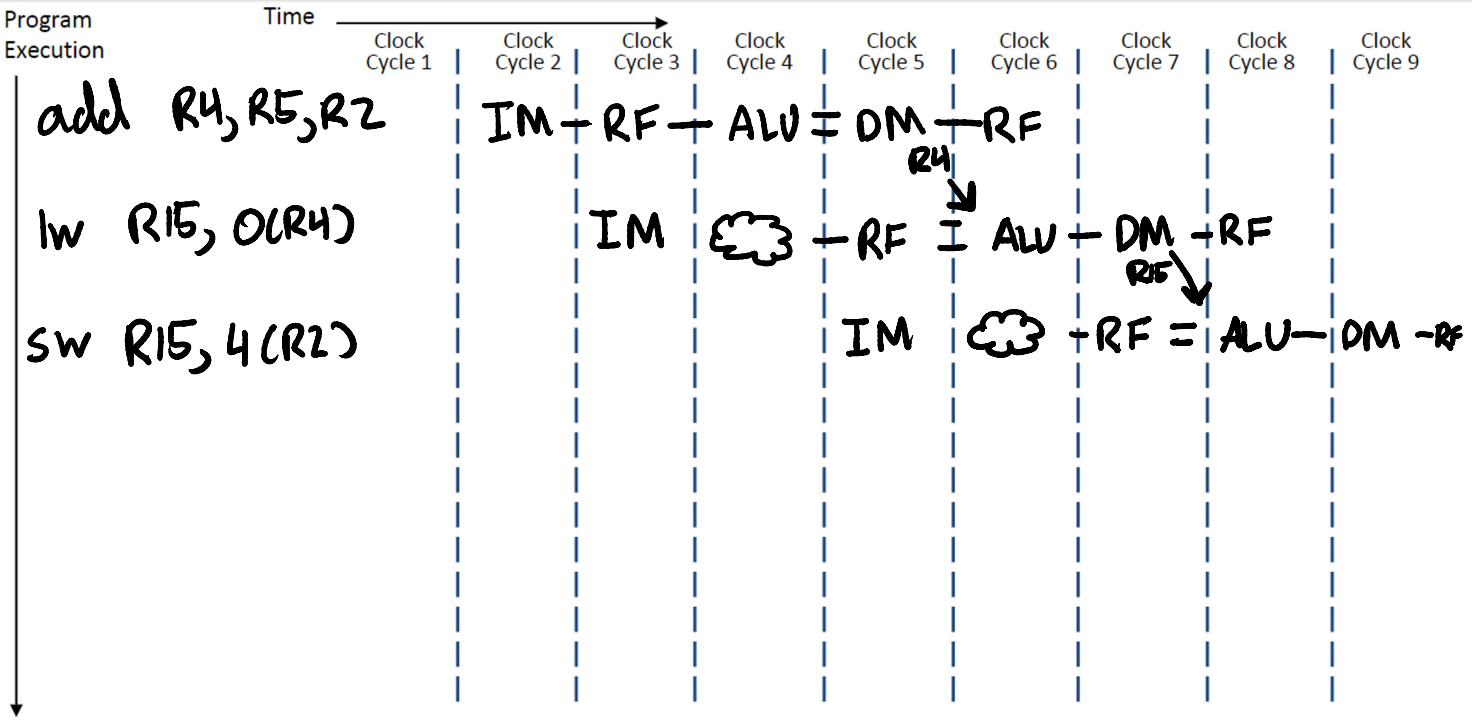
```
add  R4, R5, R2
lw   R15, 0(R4)
sw   R15, 4(R2)
```

IM RF ALU DM RF

IM RF ALU DM RF

IM RF ALU DM RF

Answer



Question 6: Branch delay slot

The code in Part A has two data dependencies (R3 and R4), and can be resolved using forwarding and double-pumped register file. However, we still have control dependency for "beq" instruction. To solve the control dependency issue, we add a "nop" after the "beq" instruction, as showsn in Part B.

Question: We would like to keep the pipeline full as much as possible and improve the performance. Can we reorder the instructions to avoid NOPs? If so, how?

A) Identify dependencies

```
add  R3, R3, R4
sub  R4, R2, R3
beq  R0, R2, end
addi R4, R4, 12
end:
```

B) insert NOPs

```
add  R3, R3, R4
sub  R4, R2, R3
beq  R0, R2, end
nop
addi R4, R4, 12
```

C) reorder for performance

Answer

- Write the reordered instructions.

Answer

- Write the reordered instructions.

```
add R3, R3, R4
beq R0, R2, end
sub R4, R2, R3
addi R4, R4, 12
end:
```