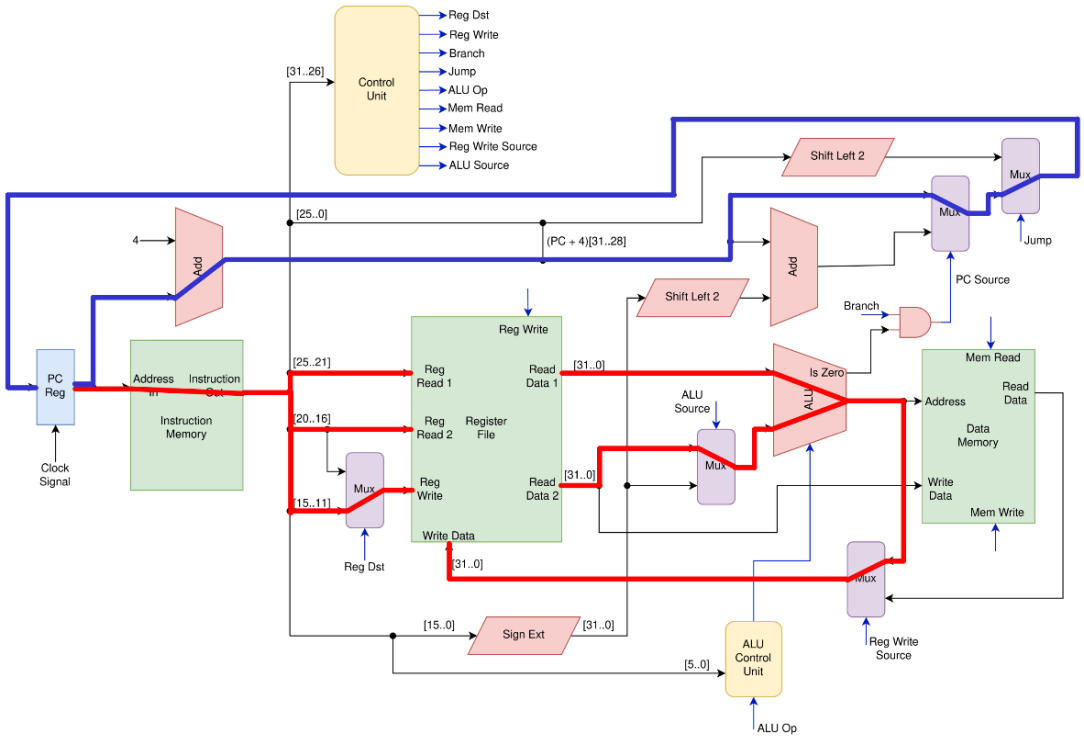
CSE490/590, Summer 2025 Homework 2 Solution

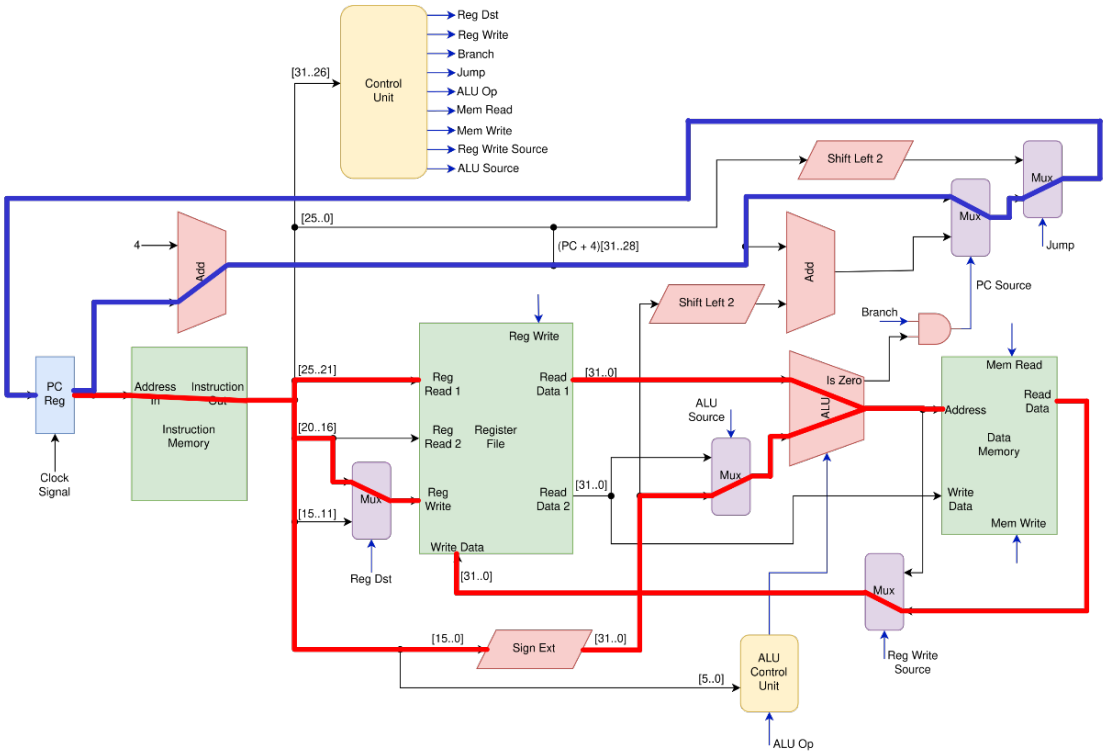
[Question 1]

Highlight the flow of data in the MIPS data path for the instructions:

a. `add $s1, $s2, $s3`



b. `lw $t3, 4($t4)`



CSE490/590, Summer 2025 Homework 2 Solution

[Question 2]

Consider the following sequence of simplified instructions:

Instruction Address:	Instruction:
100	ADD
104	BEQ+200
108	ADD
112	ADD
308	ADD

Assume the following:

- There are 5 pipeline stages: IF (Instruction Fetch), ID (Instruction Decode), EX (Execution), MEM (Memory Access), WB (Write Back).
- The second instruction, `BEQ+200`, takes the branch and jumps to the instruction at the address 308.
- A branch instruction can determine the next PC only at the EX stage.
- The CPU always speculates that it will execute the instruction at (PC + 4) next.
- If a branch or jump needs to be taken, all instructions in the pipeline are killed.

Complete the time table below with the correct stages at each cycle for all instructions. Use "***" to indicate pipeline bubbles.

Instruction Address	Instruction	Time:								
		t0	t1	t2	t3	t4	t5	t6	t7	t8
100	ADD	IF	ID	EX	MEM	WB				
104	BEQ+200		IF	ID	EX	MEM	WB			
108	ADD			IF	ID	***	***	***		
112	ADD				IF	***	***	***	***	
308	ADD					IF	ID	EX	MEM	WB

CSE490/590, Summer 2025 Homework 2 Solution

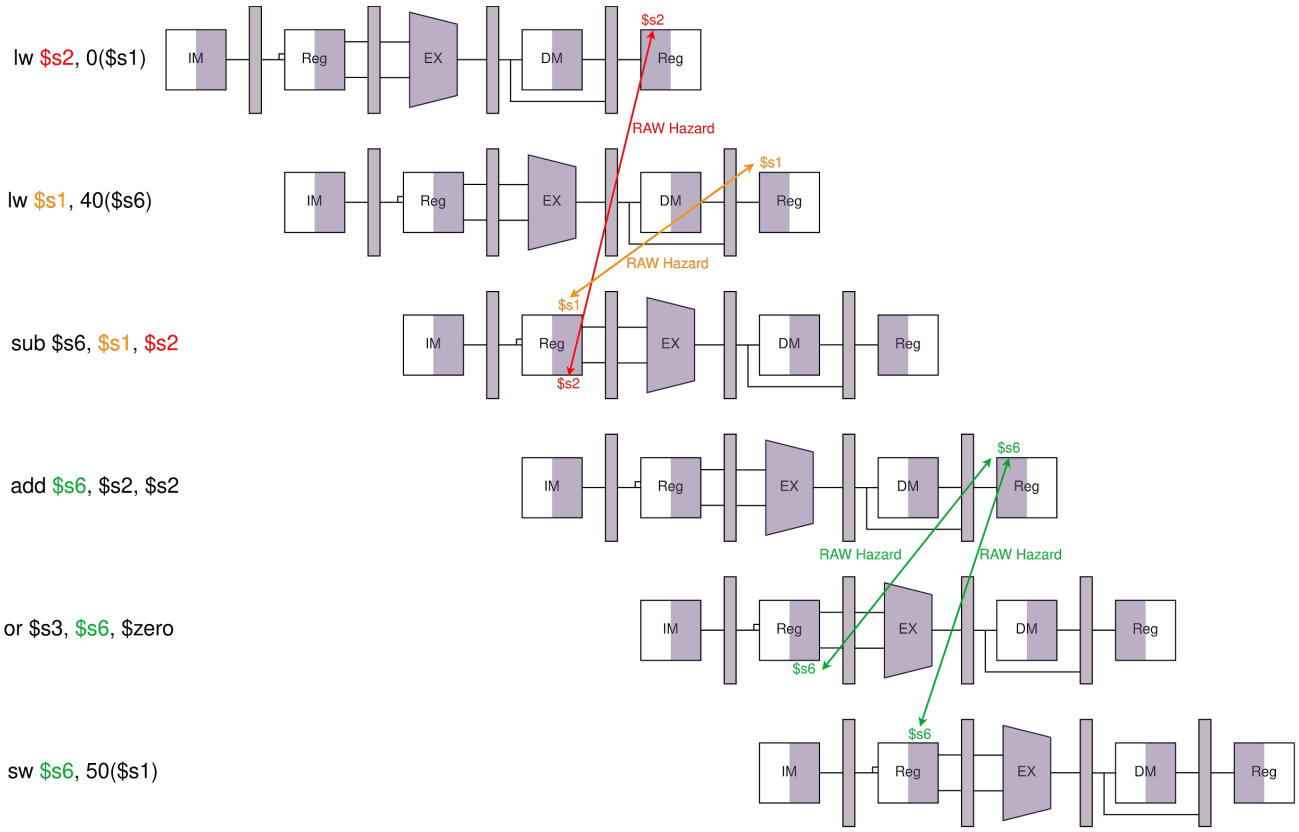
[Question 3]

In the following instruction sequence for a MIPS 5-stage pipelined datapath, list the data hazards:

```
lw $s2, 0($s1)
lw $s1, 40($s6)
sub $s6, $s1, $s2
add $s6, $s2, $s2
or $s3, $s6, $zero
sw $s6, 50($s1)
```

Answer:

```
lw $s2, 0($s1)
lw $s1, 40($s6)
sub $s6, $s1, $s2
add $s6, $s2, $s2
or $s3, $s6, $zero
sw $s6, 50($s1)
```



CSE490/590, Summer 2025 Homework 2 Solution

[Question 4]

Consider the following instruction sequence.

```
add $s5,$s2,$s1
lw $s3,4($s5)
lw $s2,0($s2)
or $s3,$s5,$s3
sw $s3,0($s5)
```

a. Show the pipeline diagram after inserting NOPs to overcome data dependencies

Instruction:	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
add $s5, $s2, $s1	IF	ID	EX	MEM	WB										
NOP		-	-	-	-	-									
NOP			-	-	-	-	-								
lw $s3, 4($s5)				IF	ID	EX	MEM	WB							
lw $s2, 0($s2)					IF	ID	EX	MEM	WB						
NOP						-	-	-	-	-					
or $s3, $s5, $s3							IF	ID	EX	MEM	WB				
NOP								-	-	-	-	-			
NOP									-	-	-	-	-		
sw $s3, 0($s5)										IF	ID	EX	MEM	WB	

b. Show the pipeline diagram after inserting Data Forwarding Unit to overcome data dependencies

Instruction:	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14
add $s5, $s2, $s1	IF	ID	EX	MEM	WB										
lw $s3, 4($s5)		IF	ID	EX	MEM	WB									
lw $s2, 0($s2)			IF	ID	EX	MEM	WB								
or $s3, $s5, $s3				IF	ID	EX	MEM	WB							
sw $s3, 0($s5)					IF	ID	EX	MEM	WB						

c. Compare performance of (a) vs (b): Option (b) is better in terms of performance because the sequence of instructions is completed in T8 time units (cycles) whereas option (a) takes T13 time units. But option (b) has a tradeoff of using an extra hardware forwarding unit which increases area.

CSE490/590, Summer 2025 Homework 2 Solution

[Question 5]

Assume the time for stages is

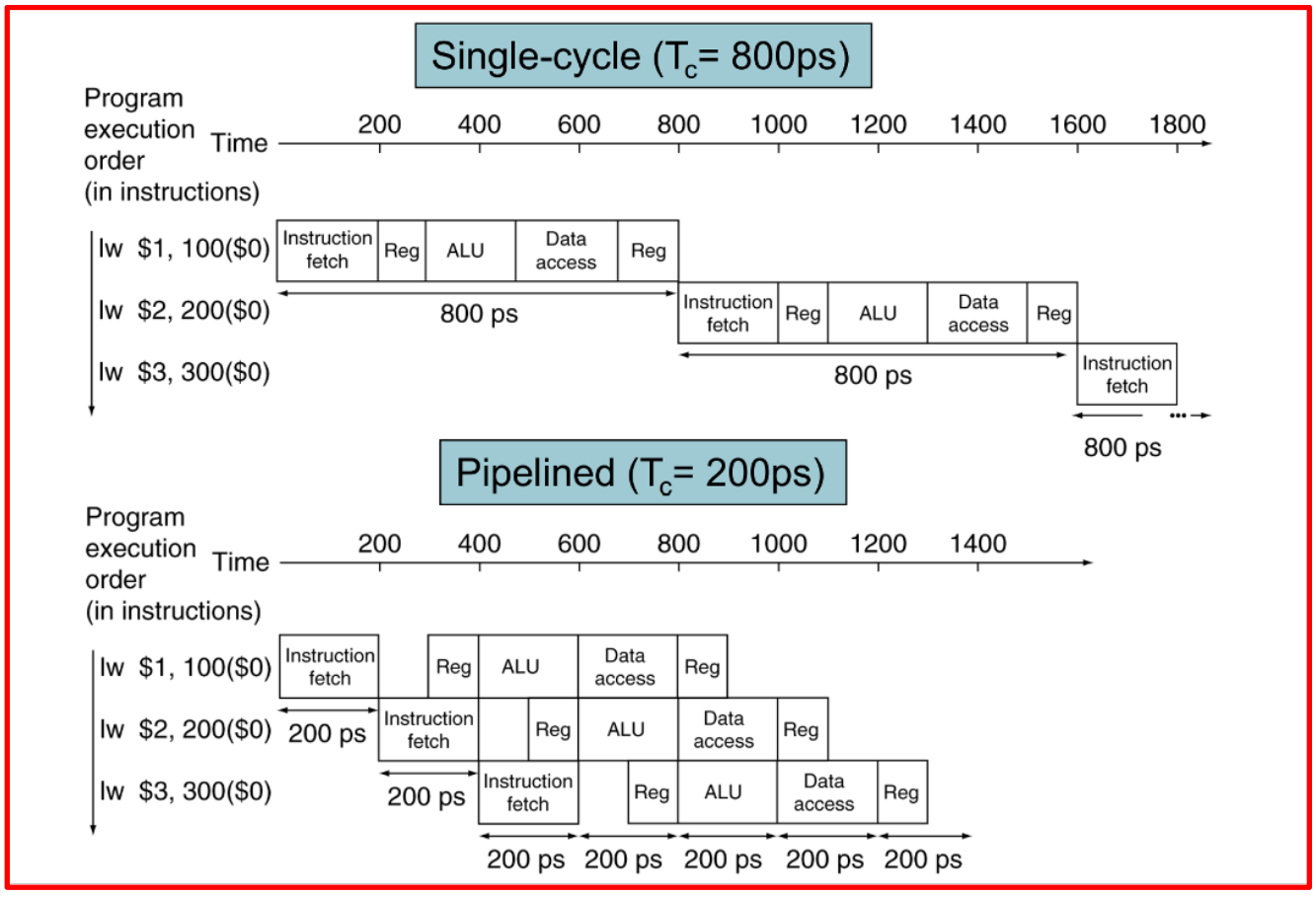
- 100ps for register read or write
- 200ps for other stages

Compare pipelined datapath with single-cycle datapath for the following instruction sequence:

```
lw $1,100($0)
lw $2,200($0)
lw $3,300($0)
```

The following table provides how much time is spent in each stage by a specific instruction:

Instr	Instr fetch	Register read	ALU op	Memory access	Register write	Total time
lw	200ps	100 ps	200ps	200ps	100 ps	800ps
sw	200ps	100 ps	200ps	200ps		700ps
R-format	200ps	100 ps	200ps		100 ps	600ps
beq	200ps	100 ps	200ps			500ps



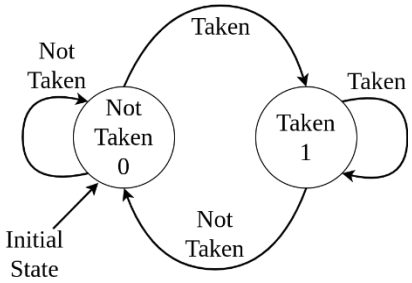
[Question 6]

CSE490/590, Summer 2025 Homework 2 Solution

Consider the following list of instructions. Assume that the initial values for `$s1`, `$s2`, `$s3`, `$s4`, and `$s5` are all 0:

```
loop:     addi $s2, $s1, -2
          bne $s5, $s2, target1
          addi $s3, $s3, 0
target1:  addi $s1, $s1, 1
          addi $s4, $s1, -3
          bne $s5, $s4, loop
```

Assume that we have a 1-bit branch predictor that stores the result of the last branch and makes the prediction based on the result. Show the results of all predictions throughout the execution. (Use T/N to represent Taken/ Not Taken)



Branch	Prediction (T/N)	Actual Result (T/N)
`1st bne->target1`	N	T
`2nd bne->loop`	T	T
`3rd bne->target`	T	T
`4th bne->loop`	T	T
`5th bne->target1`	T	N
`6th bne->loop`	N	N

The first Branch in the table is filled for you. The Prediction was Not Taken (N) and the Actual Result was Taken (T). The Prediction of current branch depends on the Prediction and Actual Result of previous branch. Here, N and T (prediction and actual results of branch 1) would produce a predicted result of T for branch 2.

Calculation of Branch 2's Actual Result:

(Note that the initial values for $s1, $s2, $s3, $s4 and $s5 are all 0)

After branch 1 execution jumps to "target1". As the instructions execute, $s1 is set to 1 and $s4 is set to -2. The next branch instruction is reach, since $s5 is not equal to $s4, the branch is taken which causes execution to jump to "loop", thus the Actual Result is Taken (T) for branch 2. T and T from branch 2 gives T as the predicted result for branch 3. The rest of the table follows the same pattern.