Question 1 from 16 July 2020, part 2 (Format OPEN TEXT)
The following loop written in C has multiple types of dependences.

```
for (i=1; i<=100, i++) {
      Y[i] = X[i] + C1 /*S1*/
      X[i] = X[i] + C1 /*S2*/
      Z[i] = Y[i] + C2 /*S3*/
      Y[i] = C1 - Y[i] /*S4*/
}
```

1. Considering only the intra-iteration dependencies, find all type of data dependences, output dependences and anti-dependences for each statement.
2. Eliminate the output dependences and anti-dependences by variable renaming.

1) Solution:

- In S2 there is an anti-dependence with S1 for X[i];
- In S3 there is a data-dependence with S1 for Y[i];
- In S4 there is a data-dependence with S1 for Y[i];
- In S4 there is an anti-dependence with S3 for Y[i];
- In S4 there is an output-dependence with S1 for Y[i].

There are no loop-carried dependencies

The above dependencies are generating the following hazards:

```
for (i=1; i<=100, i++) {
Y[i] = X[i] + C1          /*S1*/
X[i] = X[i] + C1          /*S2   WAR X[i] w. S1 */
Z[i] = Y[i] + C2          /*S3   RAW Y[i] w. S1 */
Y[i] = C1 - Y[i]          /*S4   RAW Y[i] w. S1; WAR Y[i] w.S3; WAW Y[i] w. S1 */
}
```

2) Solution:

A possible variable renaming, yielding a code with only true data dependencies (RAW hazards) is:

```
for (i=1; i<=100, i++) {
      Y1[i] = X[i] + C1;       /* S1  Y[i] renamed to Y1[i] to avoid WAW & WAR*/
      X1[i] = X[i] + C1;       /* S2: X[i] renamed to X1[i] to avoid WAR */
      Z[i] = Y1[i] + C2;       /* S3  RAW Y1[i] w. S1*/
      Y[i] =  C1  - Y1[i];     /* S4: RAW Y1[i] w. S1*/
}
```

Question 1 from 31 Aug. 2020, Part 2 (format open text)

The following loop written in C has multiple types of dependences.

```
for (i=1; i<=100, i++) {
      Y[i] = X[i] + W[i]; /*S1*/
      Z[i] = Y[i] + C1;   /*S2*/
      Z[i] = Z[i] – Y[i]  /*S3*/
}
```

1. Considering only the intra-iteration dependencies, find all type of data dependences, output dependences and anti-dependences for each statement.
2. Eliminate the output dependences and anti-dependences by variable renaming.

1) Solution

- In S2, there is a data dependence with S1 for Y[i];
- In S3, there is a data dependence with S1 for Y[i];
- In S3, there is a true data dependence with S2 for Z[i];
- In S3, there is an output-dependence with S2 for Z[i]

There are no loop-carried dependencies

The above dependencies are generating the following **hazards**:

```
for (i=1; i<=100, i++) {
      Y[i] = X[i] + W[i]; /*S1*/
      Z[i] = Y[i] + C1;   /*S2 RAW Y[i] w. S1 */
      Z[i] = Z[i] – Y[i]  /*S3 RAW Y[i] w. S1; RAW Z[i] w. S2; WAW Z[i] w.S2 */
}
```

2) Solution:

A possible **variable renaming,** yielding a code with only true data dependencies is:

```
for (i=1; i<=100, i++) {
      Y[i] = X[i] + W[i]; /*S1*/
      Z1 = Y[i] + C1;     /*S2 RAW Y[i] w. S1 */
      Z[i] = Z1 – Y[i]    /*S3*/ RAW Y[i] w. S1; RAW Z1 w. S2 */
}
```

Question 8 from 16 July 2020, part 2 (Format OPEN TEXT)

Please consider the program in the following table to be executed on a CPU with dynamic scheduling based on TOMASULO algorithm with all cache HITS:

- 2 RESERVATION STATIONS **(RS1, RS2)** + 2 LOAD/STORE unit **(LDU1, LDU2)** with **latency 4**
- 2 RESERVATION STATIONS **(RS3, RS4)** + 2 ALU/BR FUs **(ALU1, ALU2)** with **latency 2**

Please write below the column **HAZARD TYPE** for each instruction:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
`I1: lw $f1,0($r0)`	1	2	6		RS1	LDU1
`I2: lw $f2,0($r0)`	2	3	7		RS2	LDU2
`I3: fadd $f2,$f2,$f1`	3	8	10		RS3	ALU1
`I4: sw $f2,0($r0)`	7	11	15		RS1	LDU1
`I5: lw $f1,4($r0)`	8	9	13		RS2	LDU2
`I6: lw $f2,4($r0)`	14	15	19		RS2	LDU2
`I7: fadd $f2,$f2,$f1`	15	20	22		RS3	ALU1
`I8: sw $f2,4($r0)`	16	23	27		RS1	LDU1

Solution

INSTRUCTION	Hazards Type
`I1: lw $f1,0($r0)`	-
`I2: lw $f2,0($r0)`	-
`I3: fadd $f2,$f2,$f1`	(RAW $f1) RAW $f2
`I4: sw $f2,0($r0)`	STRUCT RS1 RAW $f2
`I5: lw $f1,4($r0)`	-
`I6: lw $f2,4($r0)`	STRUCT RS2
`I7: fadd $f2,$f2,$f1`	(RAW $f1) RAW $f2
`I8: sw $f2,4($r0)`	RAW $f2

Any WAW & WAR are solved by using Tomasulo's register renaming;

Question 8 from 11 Jan. 2021, Part 1

Let's consider the following code sequence executed by the 5 stage MIPS optimized pipeline with forwarding paths:

```
I1:   LW $s1, VECTA ($R6)
I2:   ADD $s1, $s0, $s1
I3:   SUB $s6, $s1, $s6
I4:   OR $s4, $s4, $s6
```

Which one of the following answers is true? ***(Format Multiple choices - SINGLE ANSWER)***

Answer 1: One stall required in ID stage of I2 to avoid RAW $s1 to I1
Answer 2: One stall required in ID stage of I3 to avoid RAW $s1 to I2.
Answer 3: One stall required in ID stage of I4 to avoid RAW $s6 to I3.
Answer 4: No stalls because all hazards solved by forwarding paths .

Solution: Answer 1 is TRUE

Question 2 from 11 Jan. 2021, Part 2 (Format OPEN TEXT)

Please consider the program in the following table executed on a CPU with dynamic scheduling based on **TOMASULO algorithm** with all cache HITS and a single Common Data Bus:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT
`I1: lw $f1,0($r0)`	1	2	6
`I2: fsubi $f1,$f1,C1`	2	7	9
`I3: faddi $f2,$f1,C2`	10	11	13
`I4: sw $f2,0($r0)`	11	14	18
`I5: lw $f1,4($r0)`	19	20	24
`I6: fsub $f2,$f2,$f1`	20	25	27
`I7: sw $f2,4($r0)`	25	28	32

- Please write **how many reservation stations** are present in the Tomasulo architecture and explain **how many structural hazards** are there.

Solution

There are:
*1 RESERVATION STATION **(RS1)** in front of the LOAD/STORE unit (LDU1) with latency 4*
*1 RESERVATION STATION **(RS2)** in front of the ALU/BR FUs (ALU1) with latency 2*
*There are **3 structural hazards:***

1. *one structural hazard RS2 in **I3;***
2. *one structural hazard RS1 in **I5;***
3. *one structural hazard RS1 in **I7;***

Optional explanation:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
`I1: lw $f1,0($r0)`	1	2	6		RS1	LDU1
`I2: fsubi $f1,$f1,C1`	2	**7**	9	**RAW $f1**	RS2	ALU1
`I3: faddi $f2,$f1,C2`	**10**	11	13	**STRUCT RS2** (RAW $f1)	**RS2**	ALU1
`I4: sw $f2,0($r0)`	11	**14**	18	**RAW $f2**	RS1	LDU1
`I5: lw $f1,4($r0)`	**19**	20	24	**STRUCT RS1**	**RS1**	LDU1
`I6: fsub $f2,$f2,$f1`	20	**25**	27	**RAW $f1** (RAW $f2)	RS2	ALU1
`I7: sw $f2,4($r0)`	**25**	**28**	32	**STRUCT RS1** + **RAW $f2**	**RS1**	LDU1

Any WAW & WAR hazard is solved by using Tomasulo's register renaming;

Question 3 from 11 Jan. 2021, Part 2 (Format OPEN TEXT)

Let us consider one iteration of the following loop written in MIPS assembly code:

```
LOOP: LD $F0, 0 ($R1)
      ADDD $F2, $F0, $F0
      SD $F2, ($R1)
      SUBI $R1, $R1, 8
      BNEZ $R1, LOOP
```

Please explain if the store instruction can be rescheduled after the branch or not and explain if the loop iterations are executed correctly.

Solution

The STORE instruction can be scheduled after the branch in the Branch Delay Slot because it is always executed (either the branch is taken or untaken). However, to ensure a correct code execution its index must be updated to: SD $F2, 8 ($R1) as follows:

```
LOOP: LD $F0, 0 ($R1)
      ADDD $F2, $F0, $F0
      SUBI $R1, $R1, 8
      BNEZ $R1, LOOP
      SD $F2, 8($R1)          # BRANCH DELAY SLOT
```

Question 1 from 8 Feb. 2021, Part 1

*Let's consider the following code sequence executed by the 5 stage MIPS optimized pipeline with all forwarding paths and **with** early evaluation of the PC:*

```
I1:        lw $t3, VECTA ($t3)
I2:        sw $t0, VECTB ($t3)
I3:        add $t0, $t3, $t0
I4:        sub $t3, $t3, $t0
I5:        beq $t3, $t0, 16
```

Which one of the following answers is true? **(Format Multiple Choice – Multiple answer)**

Answer 1: One stall required between I1 and I2

Answer 2: One stall required between I2 and I3

Answer 3: One stall required between I3 and I4

Answer 4: One stall required between I4 and I5

Solution: Answer1 and Answer 4 are TRUE

Question 2 from 8 Feb. 2021, Part 1

*Let's consider the following code sequence executed by the 5 stage MIPS optimized pipeline with all forwarding paths and **without** the early evaluation of the PC:*

```
I1:        lw $t3, VECTA ($t3)
I2:        sw $t0, VECTB ($t3)
I3:        add $t0, $t3, $t0
I4:        sub $t3, $t3, $t0

I5:        beq $t3, $t0, 16
```

Which one of the following answers is true? ***(*Format Multiple Choice –*SINGLE ANSWER)***

Answer 1: One stall required between I1 and I2

Answer 2: One stall required between I2 and I3

Answer 3: One stall required between I3 and I4

Answer 4: One stall required between I4 and I5

Answer 5: No stalls because all hazards are solved by forwarding paths

Solution: Answer1 is TRUE

Question 2 from 8 Feb. 2021, Part 2 (Format OPEN TEXT)

Please consider the program in the following table executed on a CPU with dynamic scheduling based on **TOMASULO algorithm** with all cache HITS and a single Common Data Bus:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT
I0 LD $F6, VECTA ($R1)	1	2	6
I1 LD $F4, VECTB ($R1)	7	8	12
I2 ADDD $F6, $F6, $F8	8	9	11
I3 SUBD $F4, $F4, $F8	9	13	15
I4 SD $F6, VECTA ($R1)	13	14	18
I5 SD $F4, VECTB ($R1)	19	20	24
I6 JR $R2	20	21	23

How many reservation stations are there in the Tomasulo architecture?

How many structural hazards are there?

Solution:

There are:
1 RESERVATION STATION (RS1) in front of the LOAD/STORE unit (LDU1) with latency 4
2 (or more) RESERVATION STATIONS (RS2, RS3) in front of the ALU/BR FUs (ALU1) with latency 2
There are 3 structural hazards:

1. *one structural hazard RS1in I1;*
2. *one structural hazard RS1 in I4;*
3. *one structural hazard RS1 in I5;*

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
I0 LD $F6, VECTA ($R1)	1	2	6		RS1	LDU1
I1 LD $F4, VECTB ($R1)	**7**	8	12	**STRUCT RS1**	RS1	LDU1
I2 ADDD $F6, $F6, $F8	8	9	11	(RAW $f6)	RS2	ALU1
I3 SUBD $F4, $F4, $F8	9	**13**	15	**RAW $f4**	RS3	ALU1
I4 SD $F6, VECTA ($R1)	**13**	14	18	**STRUCT RS1** (RAW$f6)	RS1	LDU1
I5 SD $F4, VECTB ($R1)	**19**	20	24	**STRUCT RS1** (RAW $f4)	RS1	LDU1
I6 JR $R2	20	21	23		RS2	ALU1

Any WAW & WAR hazard is solved by using Tomasulo's register renaming.

Question 2 - Midterm test 08/05/2023

Let's consider the following loop code to be executed on a CPU with dynamic scheduling based on **TOMASULO algorithm** with all cache HITS, a single Common Data Bus and:

- *2 RESERVATION STATIONS **(RS1, RS2)** for 2 LOAD/STORE UNITS **(LDU1, LDU2)** with latency **4***
- *2 RESERVATION STATION **(RS3, RS4)** for 2 ALU/BR FUs **(ALU1, ALU2)** with latency **2***
- Static Branch Prediction **BTFNT (BACKWARD TAKEN FORWARD NOT TAKEN)** with Branch Target Buffer

Please complete the following table:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
`FOR1:beq $t6,$t7, END`	**1**	**2**	**4**	None	RS3	ALU1
`lw $t2,VECTA($t6)`						
`lw $t3,VECTB($t6)`						
`addi $t2,$t2,k`						
`sw $t2,VECTA($t6)`						
`add $t4,$t2,$t3`						
`sw $t4,VECTC($t6)`						
`addi $t6,$t6,4`						
`j FOR1`						

*Calculate the **CPI** and the **IPC:***

CPI = _____

IPC= _____

Solution:

INSTRUCTION	ISSUE	START EXEC	WRITE RESULT	Hazards Type	RSi	UNIT
`FOR1:beq $t6,$t7, END`	**1**	**2**	**4**	None	RS3	ALU1
`lw $t2,VECTA($t6)`	2	3	7	(Control solved by BP-NT)	RS1	LDU1
`lw $t3,VECTB($t6)`	3	4	8	None	RS2	LDU2
`addi $t2,$t2,k`	4	8	10*	**RAW $t2**	RS4	ALU2
`sw $t2,VECTA($t6)`	8	11	15	**RAW $t2 + STRUCT RS1**	RS1	LDU1
`add $t4,$t2,$t3`	9	11	13	**RAW $t2** (RAW $t3 ok)	RS3	ALU1
`sw $t4,VECTC($t6)`	10	14	18	**RAW $t4**	RS2	LDU2
`addi $t6,$t6,4`	11	12	14	(WAR $t6 OK)	RS4	ALU2
`j FOR1`	14	15	17	**STRUCT RS3**	RS3	ALU1

(*) The result of ALU2 is forwarded through the CDB to RS1, RS3 and the RF.

CPI = # clock cycles / IC = 18 / 9 = 2

IPC = 1/CPI = 1/ 2 = 0.5

Question 4 - Midterm test 08/05/2023

Let's consider the following LOOP to be executed by a **Speculative Tomasulo** architecture with:

- 2 load buffers (Load1, Load2);
- 2 FP ALU reservation stations (FP1 and FP2);
- 2 integer reservation stations (INT1, INT2);
- 8-entry ROB (ROB0, ROB1, ….. , ROB7).

```
LOOP: LD $F0, 0 ($R1)
      FADD $F4, $F0, $F2
      FADD $F4, $F4, $F2

      SD $F4, AA ($R1)
      SUBI $R1, $R1, 8

      BNEZ $R1, LOOP   // branch prediction taken

EXIT:
```

Let's consider the ROB table by assuming that load of the first iteration is executing a cache miss. Let's continue to issue the next instructions until there are available resources or the ROB becomes full.

Please complete the ROB and the Rename Table

ROB

ROB#	Instruction	Dest.	Ready	Spec.	
ROB0	LD $F0, 0 ($R1) (1^ iteration exec. cache miss)	$F0	No	No	**HEAD**
ROB1					
ROB2					
ROB3					
ROB4					
ROB5					
ROB6					
ROB7					

Rename Table:

$F0	
$F2	
$F4	

Solution

ROB

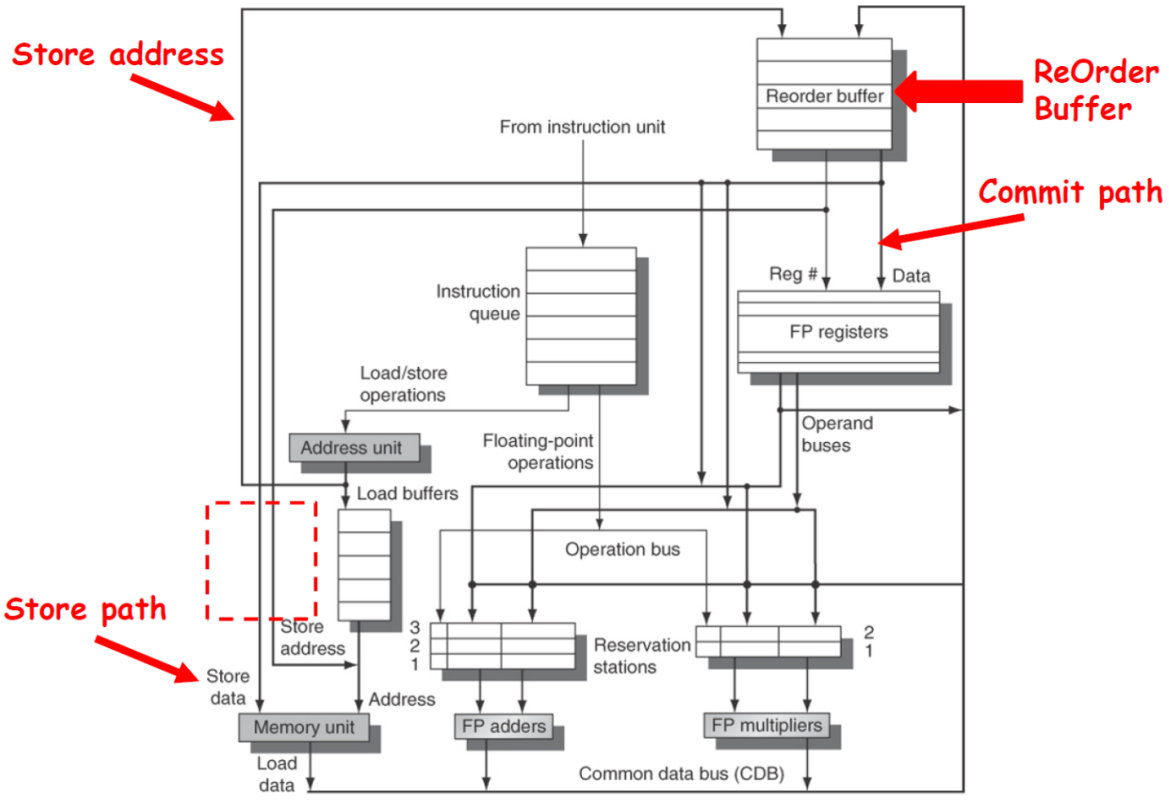
ROB#	Instruction	Dest.	Ready	Spec.	
ROB0	LD $F0, 0 ($R1) (1^ iteration exec. cache miss)	$F0	No	No	**HEAD**
ROB1	FADD $F4, $F0, $F2 (1^ iteration issued)	$F4	No	No	
ROB2	FADD $F4, $F4, $F2 (1^ iteration issued)	$F4	No	No	
ROB3	SD $F4, AA ($R1). (1^ iteration issued)	MEM	No	No	
ROB4	SUBI $R1, $R1, 8 (1^ iteration issued)	$R1	No	No	
ROB5	BNEZ $R1, LOOP (1^ branch predicted as taken)	--	No	No	
ROB6	LD $F0, 0 ($R1) (2^ iteration issued speculatively)	$F0	No	Yes	
ROB7					**TAIL**

Rename Table:

$F0	~~ROB0~~ ROB6
$F2	
$F4	~~ROB1~~ ROB2

We cannot issue instructions until the ROB becomes full because the 2 FP ALU reservation stations are busy. Therefore, the ROB7 is still empty because we cannot issue the FADD (speculatively).

In the Rename Table, $F0 points to ROB6 because the WAW $F0 is solved, while $F4 points to ROB2 because the WAW $F4 is solved.



Question 5 - Midterm test 08/05/2023

Let's consider the following loop where $R1 is set to 0 and $R2 is set to 40:

```
LOOP:  LD $F0, 0($R1)
       FADD $F4, $F0, $F2
       SD $F4, 0($R1)
       ADDI $R1, $R1, 8
       BNE $R1, $R2, LOOP
```

Let's assume to have a 1-bit BHT as dynamic branch predictor initialized as TAKEN.

How much is the **misprediction rate** of this loop?

(SINGLE ANSWER)

Answer 1: 99%

Answer 2: 20%

Answer 3: 100%

Answer 4: 2 %

Answer 5: 25 %

Solution:

Answer 2: 20% (TRUE)

The LOOP is executed 5 times. Being the predictor initialized as TAKEN, there is 1 misprediction at the last iteration of the loop: 1 misprediction out of 5 predictions (20% misprediction rate and 80% success rate).

Question 6 - Midterm test 08/05/2023

Let's consider the following loop where $R1 is set to 0 and $R2 is set to 80:

```
LOOP:  LD $F0, 0($R1)
       FADD $F4, $F0, $F2
       SD $F4, 0($R1)
       ADDI $R1, $R1, 8
       BNE $R1, $R2, LOOP
```

Let's consider the BACKWARD TAKEN FORWARD NOT TAKEN static branch prediction.

How much is the **misprediction rate** of this loop?

(SINGLE ANSWER)

Answer 1: 99%

Answer 2: 10%

Answer 3: 100%

Answer 4: 1 %

Answer 5: 20 %

Solution

Answer 2: 10% (TRUE)

The LOOP is executed 10 times. For this backward branch, the static prediction is TAKEN. Therefore, we have 1 misprediction only at the last iteration of the loop: 1 misprediction out of 10 predictions (10% misprediction rate and 90% success rate).

Question 7 - Midterm test 08/05/2023

Let's consider the following code where $R0 has been initialized to 0:

```
INIT:   ADDI $R1, $R0, 0
        ADDI $R2, $R0, 80
        ADDI $R3, $R0, 0
        ADDI $R4, $R0, 40

LOOP1: LD $F0, 0($R1)
        FADD $F4, $F0, $F2
        SD $F4, 0($R1)

LOOP2: LD $F6, 0($R3)
        FADD $F8, $F6, $F2
        SD $F8, 0($R3)
        ADDI $R3, $R3, 8
        BNE $R3, $R4, LOOP2

        ADDI $R1, $R1, 8
        BNE $R1, $R2, LOOP1
```

Answer to the following questions:

Question 1: How many iterations of LOOP1 and LOOP2 are executed?

Solution

The outer loop LOOP1 is executed 10 times.

The inner loop LOOP2 is executed 5 times for each iteration of LOOP1
=> Globally LOOP2 is executed 50 times.

Question 2: Let's assume to have a 1 entry 1-bit BHT as dynamic branch predictor (where the two branch instructions collide) initialized as Taken.
How many mispredictions are we going to observe? Please explain.

Solution

Being the predictor initialized as TAKEN, we have a misprediction only at the last iteration (exit) of the inner LOOP2 and the prediction bit is turned to NT. So, for LOOP 2, we have 1 misprediction out of 5 predictions (misprediction rate 20% for LOOP2). Exiting from the inner LOOP2 with the NT prediction generates a misprediction on the BNE-LOOP1 for 9 iterations (except for the last iteration).

When re-entering in LOOP1, the prediction bit was turned to TAKEN when re-entering in the inner LOOP2 as before.

Counting the number of mispredictions, we have 1 misprediction for the BNE-LOOP2 only when exiting from LOOP2 for 10 iterations of the outer LOOP1 therefore 10 mispredictions. For the outer LOOP1, we have 9 mispredictions for BNE-LOOP1 (except for the last iteration). Globally there are (10 + 9) = 19 mispredictions.

Given 19 mispredictions while the branch predictor has been used 60 times (50 times for BNE-LOOP2 and 10 times for BNE-LOOP1) => there are 19 mispredictions out of 60 predictions => 31.67% misprediction rate.

Question 3: Let's assume to have a 1 entry 2-bit BHT as dynamic branch predictor (where the two branch instructions collide) initialized Strongly Taken.
How many mispredictions are we going to observe? Please explain.

Solution:

Being the predictor initialized as Strongly Taken, the predictor fails and changes to the Weakly Taken state only at the last iteration (exit) of the inner LOOP2, but the prediction bit is still Taken So, for LOOP 2, we have 1 misprediction out of 5 predictions (misprediction rate 20% for LOOP2). Exiting from the inner LOOP2 as Weakly Taken, the prediction as TAKEN is correct for the BNE-LOOP1 for 9 iterations (except for the last iteration).

When re-entering in LOOP 1, the prediction state was turned to Strongly Taken when re-entering in the inner LOOP2 as before.

Counting the number of mispredictions, we have 1 misprediction for the BNE-LOOP2 only when exiting from LOOP2 for 10 iterations of the outer LOOP1 therefore 10 mispredictions. For the outer LOOP1, we have only 1 misprediction for BNE-LOOP1 at the last iteration. Globally, there are (10 + 1) = 11 mispredictions.

Given 11 mispredictions while the branch predictor has been used 60 times (50 times for BNE-LOOP2 and 10 times for BNE-LOOP1) => there are 11 mispredictions out of 60 predictions => 18.33% misprediction rate. As expected, the behavior of the 2-bit BHT is better than the previous 1-bit BHT.