EXERCISE 1(A) – DUAL-ISSUE SUPERSCALAR PIPELINE (4 points)

Given the following loop taken from a high-level program:

```
do {
    BASEC[i] = BASEA[i] + BASEB[i] + INC1 + INC2;
    i++;
}
while (i != N)
```

The program has been compiled in MIPS assembly code assuming that registers $4 and $7 have been initialized with values 0 and 4N respectively.

The symbols BASEA, BASEB and BASEC are 16-bit constant. The processor clock cycle is 2 ns.

```
L1:     lw   $2, BASEA ($4)
        addi $2, $2, INC1     // raw $2, waw $2
        lw   $3, BASEB ($4)
        addi $3, $3, INC2     // raw $3, waw $3
        add  $5, $2, $3       // raw $2, raw $3
        sw   $5, BASEC ($4)   // raw $5
        addi $4, $4, 4        // war $4
        bne  $4, $7, L1       // raw $4
```

Consider the above program be executed on a 2-issue Superscalar MIPS architecture with **Static Branch Prediction BTFNT (BACKWARD TAKEN FORWARD NOT TAKEN)** with Branch Target Buffer

Assume there are the following **optimizations** in the pipeline

- Consider for each instruction issue: **1 ALU/BRANCH** and **1 LOAD/STORE**
- Consider a Register File with **4 read ports**, **2 write** ports. A single read operation and a single write operation both at the same address can be executed;
- **Forwarding**
- Computation of PC and TARGET ADDRESS for branch & jump instructions anticipated in the **ID stage**

Complete the pipeline scheme **by inserting instructions in the proper issue line as well as the stalls** needed to solve the given hazards and by adding an ARROW to indicate the Forwarding paths used

	INSTRUCTION	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	Stalls + Forwarding Path
1 A/B	nop	IF	ID	EX	ME	WB											(CNTR hazard ok)
1 L/S	L1:lw $2, BASEA($4)	IF	ID	EX	ME	WB											
2 A/B																	
2 L/S																	
3 A/B																	
3 L/S																	
4 A/B																	
4 L/S																	
5 A/B																	
5 L/S																	
6 A/B																	
6 L/S																	
7 A/B																	
7 L/S																	
8 A/B																	
8 L/S																	

Express the formula then calculate the following metrics:

- Instruction Count per iteration **(IC)**: **IC =**
- Number of stall per iteration =
- **CPI $_{AS}$ =**

SOLUTION:

Complete the pipeline scheme **by inserting instructions in the proper issue line as well as the stalls** needed to solve the given hazards and by adding an ARROW to indicate the Forwarding paths used

	INSTRUCTION	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	Stalls + Forwarding Path
1 A/B	nop	IF	ID	EX	ME	WB											(CNTR hazard ok)
1 L/S	L1:lw $2, BASEA($4)	IF	ID	EX	ME	WB											
2 A/B	addi $2, $2, INC1		IF	IDS	ID	EX	ME	WB									1 stall + ME-EX $2
2 L/S	lw $3, BASEB($4)		IF	IDS	ID	EX	ME	WB									
3 A/B	addi $3, $3, INC2			IFS	IF	IDS	ID	EX	ME	WB							1 stall + ME-EX $3
3 L/S	nop			IFS	IF	IDS	ID	EX	ME	WB							
4 A/B	add $5, $2, $3					IFS	IF	ID	EX	ME	WB						EX-EX $3
4 L/S	sw $5, BASEC($4)					IFS	IF	ID	EX	ME	WB						EX-ME $5
5 A/B	addi $4, $4, 4							IF	ID	EX	ME	WB					
5 L/S	nop							IF	ID	EX	ME	WB					
6 A/B	bne $4, $7, L1								IF	IDS	ID	EX	ME	WB			1 stall + EX-ID $4
6 L/S	nop								IF	IDS	ID	EX	ME	WB			
7 A/B																	
7 L/S																	
8 A/B																	
8 L/S																	

Express the formula then calculate the following metrics:

- **Instruction Count per iteration (IC): IC = 8**
- **Number of stalls per iteration = 3**
- **CPI $_{AS}$ = (IC + NOPs + 2 * #stalls) / (2 *IC) = (8 + 4 + 2 * 3) / 16 = 18 / 16 = 1.125. The CPI can also be computed as follows:**

EXERCISE 1(C) – VLIW PIPELINE (4 points)

- Consider the same program be executed on a **2-issue VLIW MIPS** (Very Long Instruction Word) architecture **with Forwarding** and **Static Branch Prediction BTFNT (BACKWARD TAKEN FORWARD NOT TAKEN)** with Branch Target Buffer
- Consider for each instruction issue: **1 ALU/BRANCH** and **1 LOAD/STORE**
  - Complete the pipeline scheme **by inserting the NOPS** needed to solve the given hazards and by adding an ARROW to indicate the Forwarding paths used:

	INSTRUCTION	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	Forwarding Path
1 A/B	`nop`	IF	ID	EX	ME	WB											(CNTR hazard ok)
1 L/S	`L1 :lw $2,BASEA($4)`	IF	ID	EX	ME	WB											
2 A/B																	
2 L/S																	
3 A/B																	
3 L/S																	
4 A/B																	
4 L/S																	
5 A/B																	
5 L/S																	
6 A/B																	
6 L/S																	
7 A/B																	
7 L/S																	
8 A/B																	
8 L/S																	
9 A/B																	
9 L/S																	
10 A/B																	
10 L/S																	

Express the formula then calculate the following metrics:

- **Instruction Count per iteration (IC): IC =** **Number of NOPs =**
- **CPI $_{AS}$=**
- **Clock cycles per iteration =**
- **Code efficiency =**

SOLUTION:

- Complete the pipeline scheme **by inserting the NOPS** needed to solve the given hazards and by adding an ARROW to indicate the Forwarding paths used:

	INSTRUCTION	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	Forwarding Path
1 A/B	nop	IF	ID	EX	ME	WB											(CNTR hazard ok)
1 L/S	L1 :lw $2,BASEA($4)	IF	ID	EX	ME	WB											
2 A/B	nop		IF	ID	EX	ME	WB										
2 L/S	lw $3,BASEB($4)		IF	ID	EX	ME	WB										
3 A/B	addi $2, $2, INC1			IF	ID	EX	ME	WB									ME-EX $2
3 L/S	nop			IF	ID	EX	ME	WB									
4 A/B	addi $3, $3, INC2				IF	ID	EX	ME	WB								ME-EX $3
4 L/S	nop				IF	ID	EX	ME	WB								
5 A/B	add $5, $2, $3					IF	ID	EX	ME	WB							EX-EX $3 ME-EX $2
5 L/S	nop					IF	ID	EX	ME	WB							
6 A/B	addi $4, $4, 4						IF	ID	EX	ME	WB						
6 L/S	sw $5,BASEC($4)						IF	ID	EX	ME	WB						EX-EX $5
7 A/B	nop							IF	ID	EX	ME	WB					
7 L/S	nop							IF	ID	EX	ME	WB					
8 A/B	bne $4, $7, L1								IF	ID	EX	ME	WB				EX-ID $4
8 L/S	nop								IF	ID	EX	ME	WB				
9 A/B	*																
9 L/S																	

(*) branch delay slot to be used for next iteration because there is branch prediction

Express the formula then calculate the following metrics:

- Instruction Count per iteration **(IC)**: **IC = 8** **Number of NOPs = 8**
- **CPI $_{AS}$= (IC + #nops) / (2 *IC) = (8 + 8) / 16 = 1**
- **Clock cycles per iteration = 8 VLIW cycles / 1 iteration = 8**
- **Code efficiency = IC/ #VLIW cycles * # issues = 8 / (8 * 2) = 0.5 (there are 8 instructions and 8 NOPs in the VLIW code)**

The VLIW schedule can be represented synthetically in the following table:

	SLOT1: LOAD/STORE ops	SLOT2: ALU/BRANCH ops
C1	`lw $2,BASEA($4)`	`NOP`
C2	`lw $3,BASEB($4)`	`NOP`
C3	`NOP`	`addi $2, $2, INC1`
C4	`NOP`	`addi $3, $3, INC2`
C5	`NOP`	`add $5, $2, $3`
C6	`sw $5,BASEC($4)`	`addi $4, $4, 4`
C7	`NOP`	`NOP`
C8	`NOP`	`bne $4, $7, L1`
C9	`NOP`	`(br. delay slot)`

2-issue VLIW architecture with:

1 LD/ST UNIT with latency 2;

1 ALU/BR UNIT with latency 1 to next ALU/LD/ST and with latency 2 to next Branch,

The Branch op. completes with 1 cycle delay slot (branch solved in ID stage)

If there is branch prediction, the branch delay slot can be used for next iteration

In the RF, it is possible to read & write at the same address in the same clock cycle