EXERCISE 1 (A)

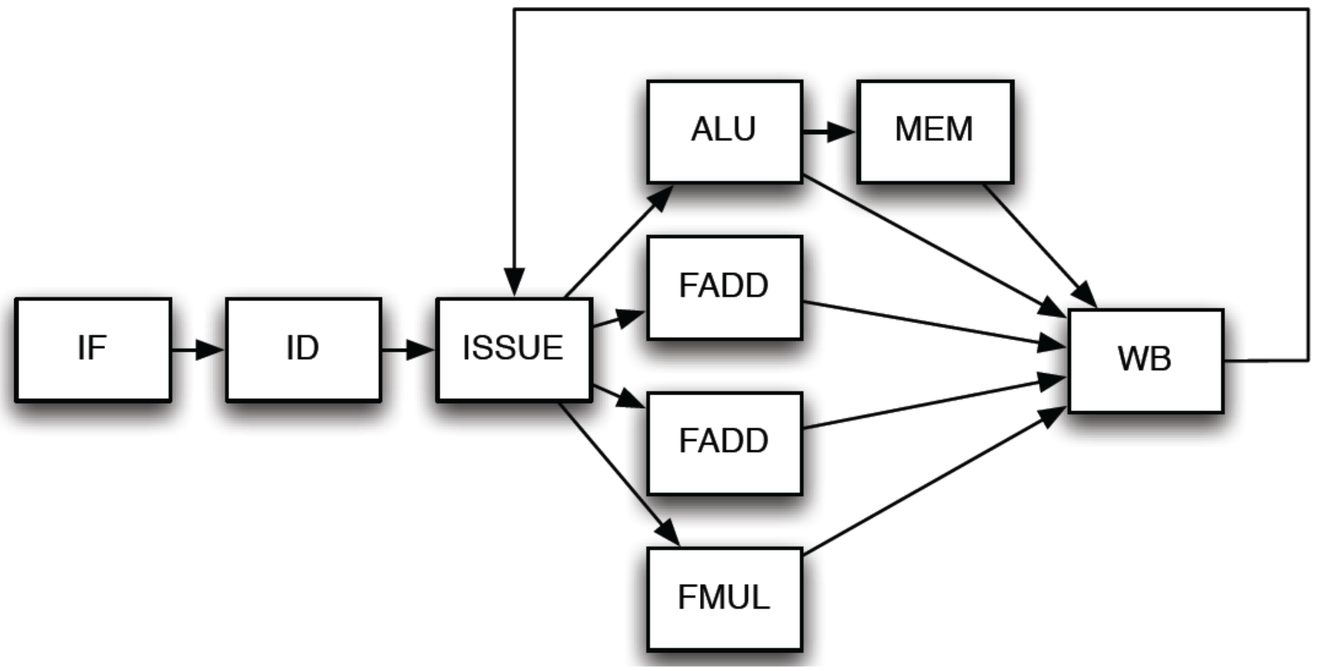
Given the following loop taken from a high level program:

```
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 **$8, $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.d       $f2, BASEA ($8)
        addi.d     $f2, $f2, INC1
        lw.d       $f4, BASEB ($8)
        addi.d     $f4, $f4, INC2
        add.d      $f6, $f2, $f4
        sw.d       $f6, BASEC ($8)
        addi       $8, $8, 4
        bne        $8, $7, L1
```

Let us consider a single iteration of the loop executed by **the following SINGLE-ISSUE OUT-OF-ORDER MIPS pipelined processor:**
The processor clock cycle is T_{clock} = 2 ns (i.e., f_{clock} = 500 MHz).



Let us assume:

- All Functional Units are pipelined;
- Instructions are fetched (IF), decoded (ID) and issued (ISSUE) IN-ORDER;
- The ISSUE stage is an ideal buffer of unlimited length that holds instructions waiting operands to start execution;
- An instruction will enter the ISSUE stage if and only if it does not cause any WAR /WAW hazard;
- Only one instruction can be issued at a time (SINGLE-ISSUE), and in case multiple instructions are ready the oldest one will go first;
- ALU instructions take 1 cycle execution latency (ALU stage);
- Memory instructions take 2 cycles execution latency (including 1 cycle in ALU stage for the computation of the memory address and 1 cycle in MEM stage to access the data cache);
- Floating-point ADD instructions take 4 cycles execution latency by the FADD;
- Floating-point MUL instructions take 5 cycles execution latency by the FMUL;
- STORE instructions allocate the WB stage (even if they are not writing in the Register File);
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage.
- Simultaneous Read and Write in the Register File at the same address can be done

- Draw the pipeline scheme by marking in RED the possible RAW (Read After Write) data hazards and in BLUE the possible control and structural hazards:

Ins	**Instruction**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**Possible Hazards**
I1	`L1: lw.d $f2,BASEA($8)`	IF	ID	IS	ALU	M	**WB**							
I2	`addi.d $f2,$f2,INC1`		IF	**ID**	**IS**	FA1	FA2	FA3	FA4	**WB**				**WAW $f2 RAW $f2**
I3	`lw.d $f4,BASEB($8)`			IF	ID	IS	ALU	M	**WB**					
I4	`addi.d $f4,$f4,INC2`				IF	**ID**	**IS**	FA1	FA2	FA3	FA4	**WB**		**WAW $f4 RAW $f4**
I5	`add.d $f6,$f2,$f4`					IF	ID	**IS**	FA1	FA2	FA3	FA4	**WB**	**RAW $f4 + RAW $f2**
I6	`sw.d $f6,BASEC($8)`						IF	ID	**IS**	ALU	M	**WB**		**RAW $6 + Struct WB**
I7	`addi $8, $8, 4`							IF	ID	IS	ALU	**WB**		**(WAR $8 ok) Struct WB**
I8	`bne $8,$7,L1`								IF	**ID**	**IS**	ALU	**WB**	**RAW $8 Struct WB**
										IF				**CNTR**

- Insert in the following pipeline scheme the stalls needed to solve the previous data, control and structural hazards:

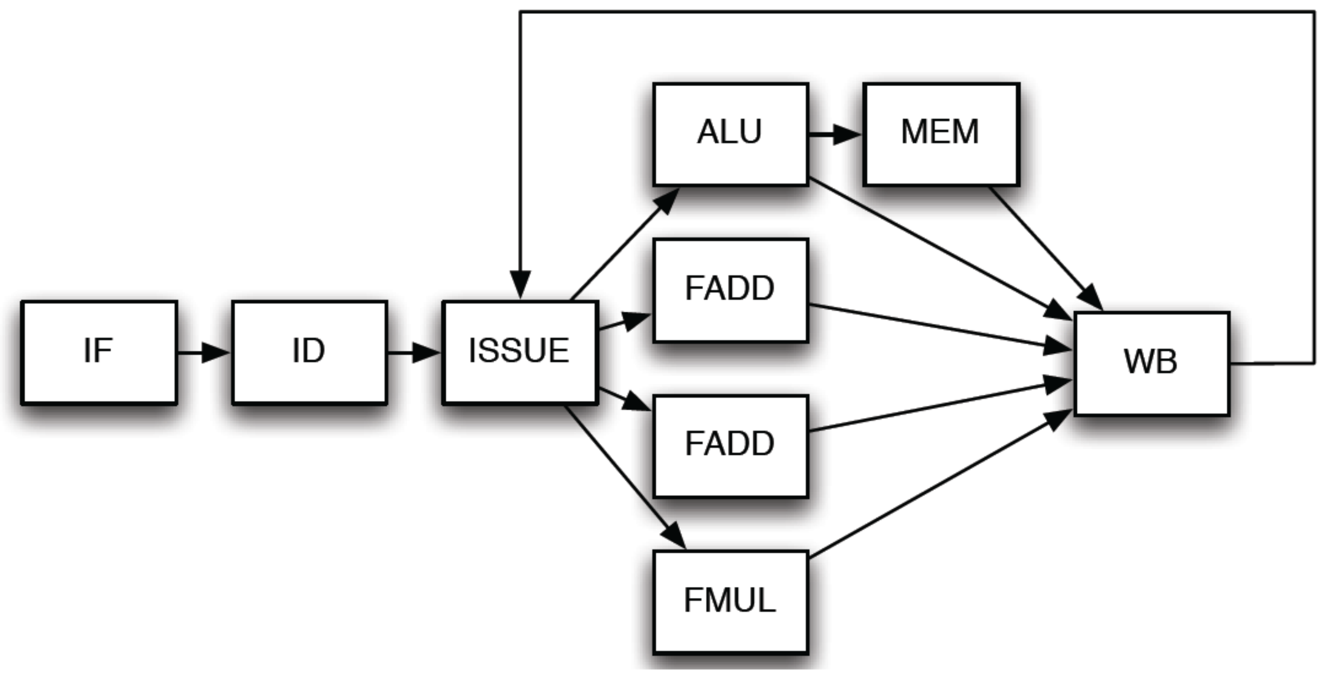
Ins	C 1	C 2	C 2	C 4	C 5	C 6	C 7	C 8	C 9	C 10	C 11	C 12	C 13	C 14	C 15	C 16	C 17	C 18	C 19	C 20	C 21	C 22	C 23	C 24	C 25	C 26	C 27	C 28	C 29	C 30
I1	IFS	IFS	IF	ID	IS	ALU	M	WB																						
I2				IF	IDS	IDS	IDS	ID	IS	FA1	FA2	FA3	FA4	WB																
I3					IFS	IFS	IFS	IF	ID	IS	ALU	M	WB																	
I4									IF	IDS	IDS	IDS	ID	IS	FA1	FA2	FA3	FA4	WB											
I5										IFS	IFS	IFS	IF	ID	ISS	ISS	ISS	ISS	IS	FA1	FA2	FA3	FA4	WB						
I6														IF	IDS	IDS	IDS	IDS	ID	ISS	ISS	ISS	ISS	IS	ALU	M	WB			
I7															IFS	IFS	IFS	IFS	IF	IDS	IDS	IDS	IDS	ID	IS	ALU	WBS	WB		
I8																				IFS	IFS	IFS	IFS	IF	ID	ISS	ISS	IS	ALU	WB

- Indicate synthetically in the first column the **NUMBER OF STALLS** to be inserted before (or during) each instruction to solve data, control and structural hazards:

Num. of Stalls	**Instruction**	**Type of Hazards**
2	`L1: lw.d $f2,BASEA($8)`	**CNTR**
3	`addi.d $f2,$f2,INC1`	**WAW $f2**
	`lw.d $f4,BASEB($8)`	
3	`addi.d $f4,$f4,INC2`	**WAW $f4**
4	`add.d $f6,$f2,$f4`	**RAW $f4**
4	`sw.d $f6,BASEC($8)`	**RAW $6**
1	`addi $8, $8, 4`	**Struct WB**
1	`bne $8,$7,L1`	**RAW $8**
Total 18		

- **Express the formulas, then calculate the following metrics for one iteration:**
  - Instruction Count per iteration **(IC)** = **8**
  - Number of stalls per iteration = **18**
  - **CPI** per iteration: **CPI = Number of clock cycles / IC = 30 / 8 = 3.75**
  - **Throughput** (expressed in **MIPS**) per iteration: **MIPS =** f_{CLOCK} / (CPI * 10^6) = (500 * 10^6) / (3.75 * 10^6) = 133.33
- **Express the formulas, then calculate the following metrics for n iterations, where n -> ∞ :**
  - Asymptotic **CPI** (**N** cycles): **CPI** $_{AS}$= (IC + # stalls) / IC = (8 + 18) / 8 = 3.25
  - Asymptotic **Throughput** (expressed in **MIPS**) (**N** cycles): **MIPS**$_{AS}$ = f_{CLOCK} / (CPI$_{AS}$ * 10^6) = (500 * 10^6) / (3.25 * 10^6) = 154

EXERCISE 1 (B)



Let us assume:

- All Functional Units are pipelined;
- Instructions are fetched (IF), decoded (ID) and issued (ISSUE) IN-ORDER;
- The ISSUE stage is an ideal buffer of unlimited length that holds instructions waiting operands to start execution;
- An instruction will enter the ISSUE stage if and only if it does not cause any WAR /WAW hazard;
- Only one instruction can be issued at a time (SINGLE-ISSUE), and in case multiple instructions are ready the oldest one will go first;
- ALU instructions take 1 cycle execution latency (ALU stage);
- Memory instructions take 2 cycles execution latency (including 1 cycle in ALU stage for the computation of the memory address and 1 cycle in MEM stage to access the data cache);
- Floating-point ADD instructions take 4 cycles execution latency by the FADD;
- Floating-point MUL instructions take 5 cycles execution latency by the FMUL;
- STORE instructions DO NOT allocate the WB stage;
- Program Counter calculation for branches and jumps has been anticipated in the ISSUE stage.
- No simultaneous Read and Write in the Register File at the same address

- Draw the pipeline scheme by marking in RED the possible RAW (Read After Write) data hazards and in BLUE the possible control and structural hazards:
-

Ins	Instruction	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	Possible Hazards
I1	`L1: lw.d $f2,BASEA($8)`	IF	ID	IS	ALU	M	WB							
I2	`addi.d $f2,$f2,INC1`		IF	ID	IS	FA1	FA2	FA3	FA4	WB				WAW $f2 RAW $f2
I3	`lw.d $f4,BASEB($8)`			IF	ID	IS	ALU	M	WB					
I4	`addi.d $f4,$f4,INC2`				IF	ID	IS	FA1	FA2	FA3	FA4	WB		WAW $f4 RAW $f4
I5	`add.d $f6,$f2,$f4`					IF	ID	IS	FA1	FA2	FA3	FA4	WB	RAW $f4 + RAW $f2
I6	`sw.d $f6,BASEC($8)`						IF	ID	IS	ALU	M	WB		RAW $6 + Struct WB
I7	`addi $8, $8, 4`							IF	ID	IS	ALU	WB		(WAR $8 ok) Struct WB
I8	`bne $8,$7,L1`								IF	ID	IS	ALU	WB	RAW $8 Struct WB
										IF				CNTR

- Insert in the following pipeline scheme the stalls needed to solve the previous data, control and structural hazards:

Ins.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24	C25	C26	C27	C28	C29	C30	C31	C32
I1	IFS	IFS	IF	ID	IS	ALU	ME	WB																								
I2				IF	IDS	IDS	IDS	ID	IS	FA1	FA2	FA3	FA4	WB																		
I3					IFS	IFS	IFS	IF	ID	IS	ALU	ME	WB																			
I4									IF	IDS	IDS	IDS	ID	IS	FA1	FA2	FA3	FA4	WB													
I5										IFS	IFS	IFS	IF	ID	ISS	ISS	ISS	ISS	ISS	IS	FA1	FA2	FA3	FA4	WB							
I6														IF	IDS	IDS	IDS	IDS	IDS	ID	ISS	ISS	ISS	ISS	ISS	IS	ALU	ME	WB			
I7															IFS	IFS	IFS	IFS	IFS	IF	IDS	IDS	IDS	IDS	IDS	ID	IS	ALU	WB			
I8																					IFS	IFS	IFS	IFS	IFS	IF	ID	ISS	ISS	IS	ALU	WB

- Indicate synthetically in the first column the NUMBER OF STALLS to be inserted before (or during) each instruction to solve data, control and structural hazards:

Num. of Stalls	Instruction	Type of Hazards
2	`L1: lw.d $f2,BASEA($8)`	CNTR
3	`addi.d $f2,$f2,INC1`	WAW $f2
	`lw.d $f4,BASEB($8)`	
3	`addi.d $f4,$f4,INC2`	WAW $f4
5	`add.d $f6,$f2,$f4`	RAW $f4
5	`sw.d $f6,BASEC($8)`	RAW $6
	`addi $8, $8, 4`	
2	`bne $8,$7,L1`	RAW $8
Total 20		

- **Express the formulas, then calculate the following metrics per one iteration:**
  - Instruction Count per iteration **(IC)** = **8**
  - Number of stalls per iteration = **20**
  - **CPI** per iteration: **CPI = Number of clock cycles / IC = 32 / 8 = 4**
  - **Throughput** (expressed in **MIPS**) per iteration: **MIPS =** f_{CLOCK} **/ (CPI * 10^6) = (500 * 10^6) / (4 * 10^6) = 125**
- **Express the formulas, then calculate the following metrics per n iteration, where n -> ∞ :**
  - Asymptotic **CPI** (**N** cycles): **CPI** $_{AS}$**= (IC + # stalls) / IC = (8 + 20) / 8 = 3.25**
  - Asymptotic **Throughput** (expressed in **MIPS**) (**N** cycles): **MIPS**$_{AS}$ **=** f_{CLOCK} **/ (CPI$_{AS}$ * 10^6) = (500 * 10^6) / (3.25 * 10^6) = 154**
  -