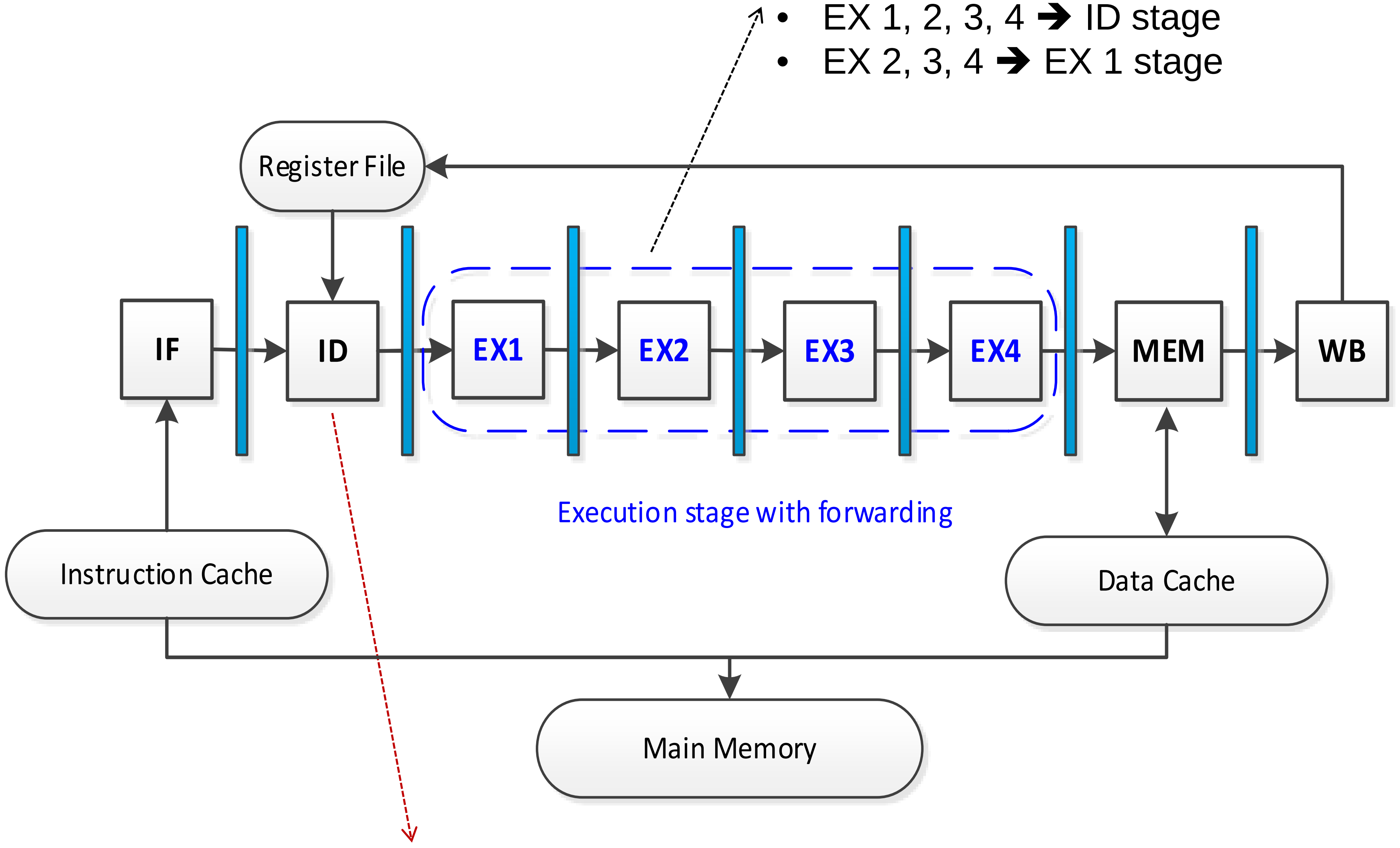
Forwarding

- MEM ➔ the EX 1 & ID stages
- EX 1, 2, 3, 4 ➔ ID stage
- EX 2, 3, 4 ➔ EX 1 stage



- Branches are resolved in the ID stage
- “not-taken prediction” will be used in IF stage

Instruction Class	Instruction Mnemonic
Data Transfers	LW, SW
Arithmetic/ logical	ADD, ADDI, MULT, MULTI, SUB, SUBI, AND, ANDI, OR, ORI, LI, LUI
Control	BEQ, BNE, J
Special purpose	HLT (to halt the simulation)

The table below shows the number of cycles each instruction takes in the EX stage.

Number of Cycles	Instructions
0 Cycle	J, BEQ, BNE, LI, LUI
1 Cycle	AND, ANDI, OR, ORI, LW, SW
2 Cycles	ADD, ADDI, SUB, SUBI
4 Cycles	MULT, MULTI

Additional Features & Assumptions

- Instructions and data are stored in memory starting at address 0x0 and 0x100, respectively.
- Load and store instructions use word-aligned addresses when accessing data.
- Both conditional and unconditional jump instructions can be forward and backward. You can assume that a program will not create a closed loop.
- An instruction stalled for data hazard in the ID stage can get the values in the same cycle WB takes place.
- The HLT instruction will mark the end of the program, i.e., fetching will seize as soon as the HLT instruction is decoded. In your implementation you can assume that the program will have two HLT instructions at the end in order to stop accessing the cache once the first HLT reaches the ID stage, i.e., the second HLT instruction will be terminated at that time.

Example

```
       LI     R1, 100h          # addr = 0x100;
       LW     R3, 0(R1)         # boundary = *addr;
       LI     R5, 1             # i = 1;
       LI     R7, 0h            # sum = 0;
       LI     R6, 1h            # factorial = 0x01;
LOOP:  MULT   R6, R5, R6        # factorial *= I;
       ADD    R7, R7, R6        # sum += factorial;
       ADDI   R5, R5, 1h        # i++;
       BNE    R5, R3, LOOP
       HLT
       HLT
```

Example: Without Memory Hierarchy

			1	2	3	4	5	6	7	8	9	10	11	12	13	14
	LI	R1, 100h	IF	ID	EX1	EX2	EX3	EX4	ME	WB						
	LW	R3, 0(R1)		IF	ID	EX1	EX2	EX3	EX4	ME	WB					
	LI	R5, 1			IF	ID	EX1	EX2	EX3	EX4	ME	WB				
	LI	R7, 0h				IF	ID	EX1	EX2	EX3	EX4	ME	WB			
	LI	R6, 1h					IF	ID	EX1	EX2	EX3	EX4	ME	WB		
Loop:	MULT	R6, R5, R6						IF	ID	EX1	EX2	EX3	EX4	ME	WB	
	ADD	R7, R7, R6							IF	ID	stall	stall	stall	EX1	EX2	EX3
	ADDI	R5, R5, 1h														
	BNE	R5, R3, Loop														
	HLT															

			7	8	9	10	11	12	13	14	15	16	17	18	19	20
	LI	R1, 100h	ME	WB												
	LW	R3, 0(R1)	EX4	ME	WB											
	LI	R5, 1	EX3	EX4	ME	WB										
	LI	R7, 0h	EX2	EX3	EX4	ME	WB									
	LI	R6, 1h	EX1	EX2	EX3	EX4	ME	WB								
Loop:	MULT	R6, R5, R6	ID	EX1	EX2	EX3	EX4	ME	WB							
	ADD	R7, R7, R6	IF	ID	stall	stall	stall	EX1	EX2	EX3	EX4	ME	WB			
	ADDI	R5, R5, 1h		IF	stall	stall	stall	ID	EX1	EX2	EX3	EX4	ME	WB		
	BNE	R5, R3, Loop			stall	stall	stall	IF	ID	stall	ID					
	HLT								IF	stall	stall	ID				
	HLT											IF				