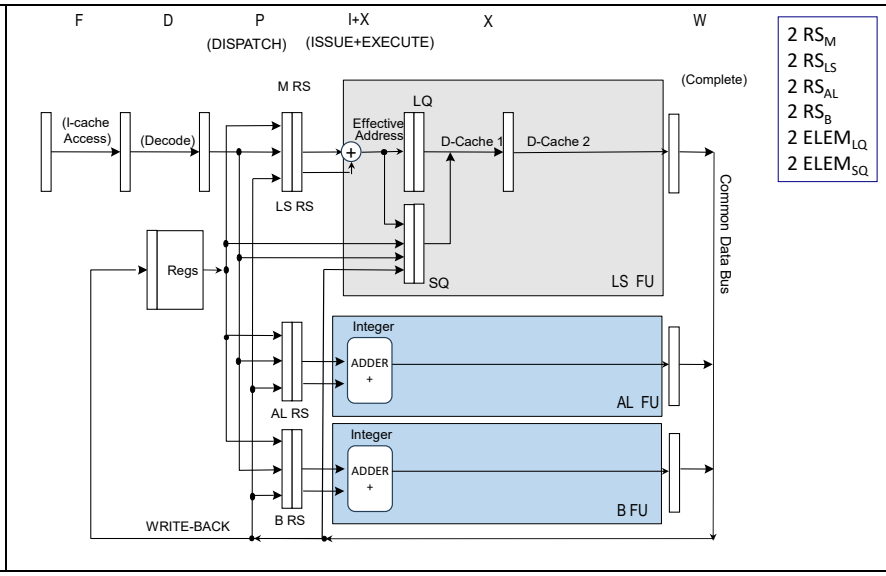
HIGH PERFORMANCE COMPUTER ARCHITECTURE 07-11-2014 **MATRICULATION NO.**_____

(MIDTERM TEST) **SURNAME**_____

REVISED 28-10-2025 **FIRST NAME**_____

1) (POINTS 40/40) Consider a **quad-dispatch (4 instructions per cycle)** processor using Tomasulo's algorithm to perform the dynamic scheduling of instructions on the pipeline shown in the following figure. This pipeline is executing the following program, which performs a search within a vector (initially, R1=0).

```
etic: LW   R2, 0(R1)     ; read Xi
      ADDI R2, R2, 3     ; adds 3 to Xi
      SW   R2, 0(R1)     ; write Xi
      ADDI R1, R1, 4     ; update  R1
      BNE  R2, R0, etic  ; continue to loop if false
```



Working hypothesis:

- the loop executes speculatively in terms of direction (always taken) but not regarding the branch condition; high-performance fetch breaks after fetching a branch; case A) no-speculation on branch condition; case B) speculation on branch condition;
- the issue stage (I) calculates the address of the actual read/write and push it into load/store queues; only 1 instruction is issued per cycle
- **reads require 5 clock cycles**; writes take 1 cycle
- when accessing memory (M), **writes** have precedence over reads and must be executed in-order
- **there is a single CDB**
- dispatch stage (P) and complete stage (W) require 1 clock cycle
- **ASSUME** that the reservation stations could be freed right before the start of issue phase (therefore extending the duration of P stage)
- only 1 instruction is committed (C stage) per cycle
- there are separated integer units: one for the calculation of the actual address, one for arithmetic and logical operations, one of the integer multiplication and one for the evaluation of the branch condition, as illustrated in this table:

Type of Functional Unit	No. of Functional Units	Cycles for stage I+X	No. of reservation stations
LS: Integer (effective addr.)	1	1	2
A: Integer (op. A-L)	1	1	2
B: Integer (branch calc.)	1	1	2

- the functional units do not take advantage of pipelining techniques internally
- the load queue has **2 slots**; the store queue has 2 slots (writes wait for the operand in the store queue, i.e., in the execution stage)

Complete the following chart until the end of the **fourth** iteration of the code fragment above, both in the case of simple dynamic scheduling (case A) that in the case of dynamic scheduling with speculation (case B).

Instr. No..	Instruction name	P: disPatch (start-stop)	I+X:Issue+Exec (clock)	M: MEM.ACCESS (start-stop)	W: CDB-write (clock)	C: Commit (clock)	Comments
I01	**LW R2,0(R1)**	1-1	2	3-7	8	9	
...	...						
...	...						

HIGH PERFORMANCE COMPUTER ARCHITECTURE 07-11-2014

SOLUTION

REVISED 28-10-2025

EXERCISE 1

CASE A (no speculation on branch condition: dispatch WAITS for branch condition verification):

Iter..	Instruction	Dispatch (P) (start-stop)	Issue (I) (clock)	MEM (M) (start-stop)	CDB-write (W) (clock)	Commit (C)(clock)	Comments
1	LW R2,0(R1)	1-1	2	3-7	8	9	
1	ADDI R2,R2,1	1-8	9	--	10	11	I waits R2 from 1/LW
1	SW R2,0(R1)	1-2	3	11	--	12	I waits issue logic; M waits R2
1	ADDI R1,R1,4	1-3	4	--	5	13	I waits issue logic;
1	BNE R2,R0,etic	2-10	11	--	--	14	I waits R2 from 1/ADDI-R2
2	LW R2,0(R1)	12-12	13	14-18	19	20	P waits branch target; I waits R1;
2	ADDI R2,R2,1	12-19	20	--	21	22	I waits R2 from 2/LW;
2	SW R2,0(R1)	12-13	14	22	--	23	I waits issue logic; I waits R1; M waits R2
2	ADDI R1,R1,4	12-14	15	--	16	24	I waits issue logic;
2	BNE R2,R0,etic	13-21	22	--	--	25	I waits R2 from 2/ADDI-R2;
3	LW R2,0(R1)	23-23	24	25-29	30	31	P waits branch target; I waits R1;
3	ADDI R2,R2,1	23-30	31	--	32	33	I waits R2 from 2/LW
3	SW R2,0(R1)	23-24	25	33	--	34	I waits issue logic; I waits R1; M waits R2;
3	ADDI R1,R1,4	23-25	26	--	27	35	I waits issue logic;
3	BNE R2,R0,etic	24-32	33	--	--	36	I waits R2
4	LW R2,0(R1)	34-34	35	36-40	41	42	P waits branch target; I waits R1;
4	ADDI R2,R2,1	34-41	42	--	43	44	I waits R2
4	SW R2,0(R1)	34-35	36	44	--	45	I waits issue logic; I waits R1; M waits R2
4	ADDI R1,R1,4	34-36	37	--	38	46	I waits issue logic; I waits R1;
4	BNE R2,R0,etic	35-43	44		--	47	I waits R2

CASE B (speculation: dispatch DOES NOT WAIT for branch condition verification):

Iter.	Instruction	Dispatch (P) (start-stop)	Issue (I) (clock)	MEM (M) (start-stop)	CDB-write (W) (clock)	Commit (C)(clock)	Comments
1	LW R2,0(R1)	1-1	2	3-7	8	9	
1	ADDI R2,R2,1	1-8	9	--	10	11	I waits R2 from 1/LW
1	SW R2,0(R1)	1-2	3	13	--	14	I waits issue logic; M waits R2 M waits mem
1	ADDI R1,R1,4	1-3	4	--	5	15	I waits issue logic;
1	BNE R2,R0,etic	2-10	11	--	--	16	I waits R2 from 1/ADDI-R2
2	LW R2,0(R1)	3-5	6	8-12	13	17	I waits R1; M waits mem
2	ADDI R2,R2,1	4-13	14	--	15	18	P waits A-RSs; I waits R2 from 2/LW;
2	SW R2,0(R1)	4-6	7	19	--	20	I waits R1; I waits issue logic; M waits R2; M waits mem
2	ADDI R1,R1,4	9-9	10	--	11	21	P waits A-RSs;
2	BNE R2,R0,etic	9-15	16		--	22	I waits R2 from 2/ADDI-R2;
3	LW R2,0(R1)	10-11	12	14-18	19	23	I waits R1; M waits mem
3	ADDI R2,R2,1	10-19	20	--	21	24	P waits A-RSs; I waits R2 from 3/LW
3	SW R2,0(R1)	10-14	15	25	--	26	I waits R1; I waits issue logic; M waits R2; M waits mem; SQ FULL
3	ADDI R1,R1,4	14-16	17	--	18	27	P waits A-RSs; I waits issue logic;
3	BNE R2,R0,etic	14-21	22	--	--	28	I waits R2 from 3/ADDI-R2
4	LW R2,0(R1)	15-16	19	20-24	25	29	I waits R1; M waits mem;
4	ADDI R2,R2,1	17-25	26	--	27	30	P waits A-RSs; I waits R2 from 4/LW
4	SW R2,0(R1)	17-20	21	28	--	31	I waits R1; I waits issue logic; M waits R2; M waits mem;
4	ADDI R1,R1,4	20-22	23	--	24	32	P waits A-RSs; I waits issue logic @19 and @20;
4	BNE R2,R0,etic	20-27	28		--	33	I waits R2 from 4/ADDI-R2