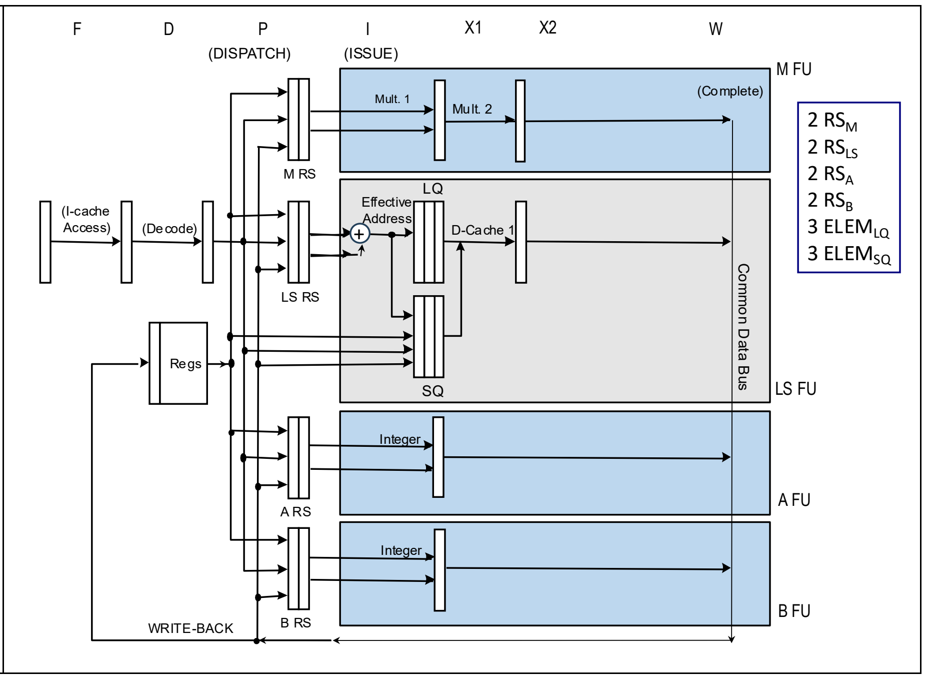
HIGH PERFORMANCE COMPUTER ARCHITECTURE midterm exam 30-10-2025 **MATR.NO.**____________________

SURNAME____________________

FIRST NAME____________________

1) (POINTS 27/30) Consider a **triple-dispatch (3 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
      MULI R2, R2, 3     ; multiplies Xi by 3
      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) and regarding the branch condition; high-performance fetch breaks after fetching a branch
- 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 3 clock cycles**; writes take 1 cycle (this means that stage M is lasting just 2 cycles after the issue)
- 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
M: Integer Multiplication	1	**2**	2

- the functional units TAKE advantage of pipelining techniques internally
- the load queue has 3 slots; the store queue has 3 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 above code fragment in the case of dynamic scheduling with speculation. Also add the instruction that occupies a certain reservation station (one of the 8) as indicated below:

Instr. No..	Instruction name	ALU RS1	ALU RS2	LS RS1	LS RS2	BU RS1	BU RS2	MU RS1	MU RS2	P: disPatch (clock)	I+X:Issue+Exec (start-stop)	M: MEM.ACCESS (start-stop)	W: CDB-write (clock)	C: Commit (clock)	Comments
`I01`	`LW R2,0(R1)`			`I01` `1-1`						1	2-2	3-5	6	7	
...	...														
...	...														

2) (POINTS 3/30) On a Linux system, write the **SINGLE command line** (no semicolon ";" in that line) to perform at the BASH shell prompt the following operation (please note that no intermediate files should be used):
- Send on the standard output, and into the file "aa", the alphabetically sorted lines matching with "ma" in the files having a filename starting with "fa" and ending with "p"

EXERCIZE 1

Instr. No..	Instruction name	ALU RS1 (start-stop)	ALU RS2 (start-stop)	LS RS1 (start-stop)	LS RS2 (start-stop)	BU RS1 (start-stop)	BU RS2 (start-stop)	MU RS (start-stop)1	MU RS2 (start-stop)	P: Dispatch (clock)	I+X: Issue (start-stop)	MEM. ACC. (start-stop)	W: CDB-write (clock)	C: Commit (clock)	Comments
I01	LW R2,0(R1)			I01 1-1						1	2	3-5	6	7	
I02	MULI R2,R2,3							I02 1-6		1	7-8	--	9	10	I waits R2 from 1/LW
I03	SW R2,0(R1)				I03 1-2					1	3	10	--	11	M waits R2 from 1/MULI; I waits issue logic;
I04	ADDI R1,R1,4	I04 2-3								2	4	--	5	12	I waits issue logic;
I05	BNE R2,R0,etic					I05 2-9				2	10	--	--	13	I waits R2 from 1/MULI
I06	LW R2,0(R1)			I06 3-5						3	6	7-9	10	14	I waits R1 from 1/ADDI
I07	MULI R2,R2,3								I07 3-10	3	11-12	--	13	15	I waits R2 from 2/LW
I08	SW R2,0(R1)				I08 3-7					3	8	16	--	17	I waits R1, M waits R2; I waits issue logic; M waits mem
I09	ADDI R1,R1,4	I09 4-8								4	9	--	11	18	I waits R1 from 1/ADDI; I waits issue logic; CDB collision
I10	BNE R2,R0,etic						I10 4-13			4	14	--	--	19	I waits R2 from 2/MULI;
I11	LW R2,0(R1)			I11 6-11						6	12	13-15	16	20	P waits LS-RS; I waits R1 from 2/ADDI;
I12	MULI R2,R2,3							I12 7-16		7	17-18	--	19	21	P waits M-RS; I waits R2 from 3/LW
I13	SW R2,0(R1)				I13 8-12					8	13	22	--	23	P waits LS-RS; I waits R; M waits R2; I waits issue logic; M waits mem
I14	ADDI R1,R1,4		I14 8-14							8	15	--	17	24	I waits R1 from 2/ADDI; I waits issue logic; CDB collision
I15	BNE R2,R0,etic					I15 10-19				10	20	--	--	25	P waits B-RS; I waits R2 from 3/MULI I waits issue logic;
I16	LW R2,0(R1)			I16 12-17						12	18	19-21	22	26	P waits LS-RS; I waits R1; I waits issue logic;
I17	MULI R2,R2,3								I17 12-22	12	23-24	--	25	27	I waits R2 from 4/LW
I18	SW R2,0(R1)				I18 13-18					13	19	26	--	28	I waits R1 from 3/ADDI; M waits R2; I waits issue logic;
I19	ADDI R1,R1,4	I19 13-20								13	21	--	23	29	I waits R1 from 3/ADDI; I waits issue logic; ; CDB collision
I20	BNE R2,R0,etic						I20 14-25			14	26	--	--	30	I waits R2 from 4/MULI

EXERCIZE 2

The requested command line is:

```
grep "ma" fa*p | sort | tee aa
```