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

(MIDTERM TEST) **SURNAME**_____

REVISED 23-10-2023 **FIRST NAME**_____

(POINTS 40/40) Consider the following snippet of code running on a quad-dispatch (4 instructions per cycle) using Tomasulo's algorithm to perform the dynamic scheduling of instructions. The program performs a search within a vector. Initially, R1 = 0.

```
etic:   LW      R2, 0(R1)       ; read Xi
        ADDI    R2, R2, 1       ; increments 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 reads and writes; only 1 instruction is issued per cycle
- reads require 5 clock cycles; writes take 0 cycles (they are written in a write-buffer + split-cache)
- when accessing memory (M), writes have precedence over reads and must be executed in-order
- there's only one CDB
- dispatch stage (P) and complete stage (W) require 1 clock cycle
- ASSUME that the reservation stations could be freed right after the issue phase
- only 1 instruction is committed (C stage) per cycle
- there are separated integer units for the calculation of the actual address, for arithmetic and logical operations, for the evaluation of the branch condition
- 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)
- the rest of the processor and has the following characteristics

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

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).

Iteraz.	Instruction	P: Dispatch (clock)	I+X: Issue+Exec (start-stop)	M: MEM ACCESS (clock)	W: CDB-write (clock)	C: Complete (clock)	Commenti
1	`LD R2,0(R1)`						
...	...						
...	...						

HIGH PERFORMANCE COMPUTER ARCHITECTURE 07-11-2014

SOLUTION

REVISED 23-10-2023

EXERCISE 1

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

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

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

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