Pitalica 1.

	1	2	3	4	5	6	7	8
ADD R1, R2,R3	IF	ID	EX	MEM	WB			
OR R2,R3,R4		IF	ID	EX	MEM	WB		
LD R4, (R1)6			IF	ID	EX	MEM	WB	
SW R1, (R5)1				IF	ID	EX	MEM	WB

Hazard podatak postoji između 1 i 3 kao i između 1 i 4 instrukcije:

- Između 1 i 3 instrukcije (ADD i LD) hazard se događa u 4 taktu kada LD čita pogrešnu vrednost registra R1- LD čita pogrešnu vrednost registra R1 zato što ADD tek na kraju 5 takta(WB faza) upisuje novu vred(pravu ili tačnu) R1
- Između 1 i 4 instrukcije (ADD i SW) hazard se događa u 5 taktu kada SW čita pogrešnu vrednost R1 (PAŽNJA: iako u taktu 5 ADD upisuje u R1 pravu vrednost, ona je tek na kraju 5 takta upisana u R1, a ovde je u 5 taktu u praleli čitanje R1 od strane SW instrukcije i upisivanje u R1 od strane ADD)

Pitalica 2.

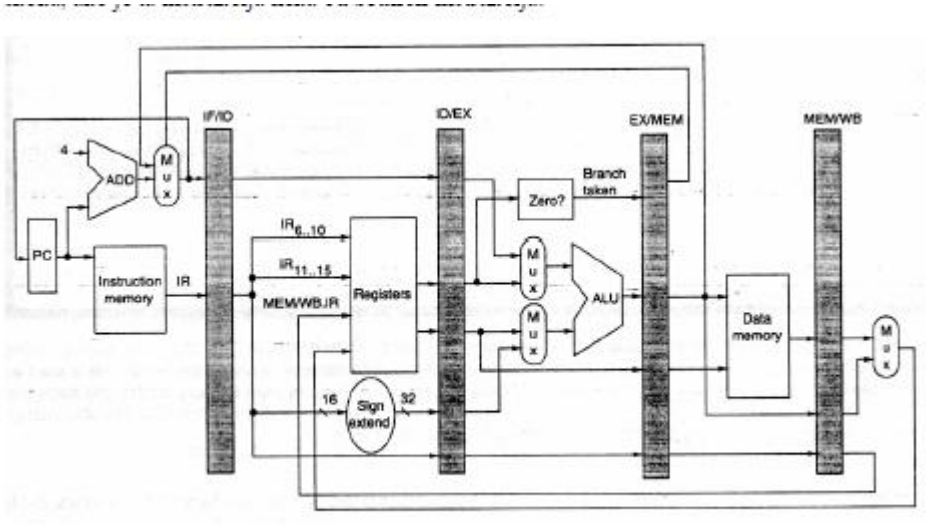
	1	2	3	4	5	6	7	8
ADD R1, R2,R3	IF	ID	EX	MEM	WB			
LD R4, (R5)6		IF	ID	EX	MEM	WB		
SW R4, (R1)1			IF	STALL	ID	EX	MEM	WB

Odgovor je 8 taktova – zato što smo morali u taktu 4 da zaustavimo pipeline kako bi LD instrukcija generisala vrednost reg R4 koju inače SW u tom istom taktu kada LD generiše(još nije izgenerisala)R4 traži vrednost reg R4, a ne može da joj prosledi (to može tek na kraju EX stepena, kada LD izgeneriše tačnu vred R4 reg)

Imamo prosleđivanje u 6 taktu iz MEM -> EX (LD instrukcija prosleđuje tačnu vrednost R4 SW instrukciji)

Pitalica 3.

To je manje više teorijsko pitanje.Odgovor je pod A).



Slika 12 *Pipeline* organizacija procesora

stepen	instrukcija
IF	**sve instrukcije** IF/ID.IR ← Mem[PC]; IF/ID.NPC, PC ← (if EX/MEM.cond then {EX/MEM.ALUOUT} else {PC + 4});
ID	**sve instrukcije** ID/EX.A ← Regs[IF/ID.IR$_{6...10}$]; ID/EX.B ← Regs[IF/ID.IR$_{11...15}$]; ID/EX.NPC ← IF/ID.NPC; ID/EX.IR ← IF/ID.IR; ID/EX.Imm ← (IF/ID.IR$_{16}$)^{16} ## IF/ID.IR$_{16...31}$;
EX	**ALU instrukcije** EX/MEM.IR ← ID/EX.IR; EX/MEM.ALUOUT ← ID/EX.A func ID/EX.B; ili EX/MEM.ALUOUT ← ID/EX.A op ID/EX.Imm; EX/MEM.cond ← 0;
	load/store instrukcije EX/MEM.IR ← ID/EX.IR; EX/MEM.ALUOUT ← ID/EX.A + ID/EX.Imm; EX/MEM.cond ← 0; EX/MEM.B ← ID/EX.B;
	branch instrukcije EX/MEM.ALUOUT ← ID/EX.NPC + ID/EX.Imm; EX/MEM.cond ← (ID/EX.A op 0);
MEM	**ALU instrukcije** MEM/WB.IR ← EX/MEM.IR; MEM/WB.ALUOUT ← EX/MEM.ALUOUT;
	load/store instrukcije MEM/WB.IR ← EX/MEM.IR; MEM/WB.LMD ← Mem[EX/MEM.ALUOUT] ili Mem[EX/MEM.ALUOUT] ← EX/MEM.B;
WB	**ALU instrukcije** Regs[MEM/WB.IR$_{16...20}$] ← MEM/WB.ALUOUT; ili Regs[MEM/WB.IR$_{11...15}$] ← MEM/WB.ALUOUT;
	load instrukcije Regs[MEM/WB.IR$_{11...15}$] ← MEM/WB.LMD;

Slika 13 Operacije koje se izvršavaju u stepenima *pipeline*-a