Zadatak 1.1

1. T=3:
   B= f; Potiče od neposrednog argumenta
2. T=3
   Jedinica EX2 propušta vrednost 0000 000f na ulaz B ALU jedinice stepena EX. Odlučuje na osnovu signala ***Rex.opcode[0]***
3. T=4
   Sa adrese PC=0000 0001
4. T=5
   Prva instrukcija **add R1 R0 15** stiže u WB stepen u **5-tom** taktu – *by the way* događa se **prosleđivanje** iz **MEM -> EX** zbog instrukcije **xori R3, R2, 77** koja koristi odredište inst **subi R2, R1, 0** kao izvorište
5. T= 5
   U taktu 5 javlja se vanredni događaj – **prosleđivanje** iz **MEM -> EX**
6. T=5
   Prosleđuje se iz polja Rmem.ALUOUT na ulaz A jediinice ALU (Stepena EX) i jedinice Zero (pomoću jedinica EXP1 i EX1 koje se koriste za realizaciju logike za prosleđivanje iz MEM,WB,TMP)
7. T=5
   Prosleđivanje je neophodno zbog rešavanja situacije – hazarda podataka (čitanje pogrešne vrednosti registra R2 od strane xori instrukcije)
8. T=5
   Na izlazu EXP1 je ffff ffff.
9. T=5
   Aktivni signali su: Rmem.ALUOUT,EXP1, **EQ3**,**RNEZ**,**Rmem.WRALU**,Rwb. WRALU,Rtmp.WRITE
10. T=
    Jjedinice za prosleđivanje su:
    - **EXP1**
    - **EXP2**
    - **MEMP**
      Signali koji omogućavaju prosleđivanje iz stepena MEM->EX su: **EQ3**,**RNEZ**,**Rmem.WRALU**
11. Zato što nije ispunjen 3ći uslov za prosleđivanje tj.imamo da je EQ2 ≠1 što znači da je Rex.rs1≠Rwb.rd; NAPOMENA: da bi se vršilo prosleđivanje iz stepena WB -> EX moraju biti ispunjena tri uslova: 1) RNEZ=1 (Rex.rs1 ≠0), 2)Rwb.WRALU (u stepenu WB je neka instrukcija koja će upisati vrednost Rmem.ALUOUT u reg fajl) i 3) EQ2=1 (Rex.rs1=Rwb.rd)
12. EQ3 je uključen pod uslovom da je Rex.rs1=Rmem.rd – adresa registra *rs1* u EX stepenu pipline-a je jednaka adresi registra *rd* stepena MEM pipeline-a
13. T=5
    U 5tom taktu **beqz** instrukcija ulazi u pipeline, a predikcija je **nema skoka.**
14. T=5
    HIT = 0 u 5tom taktu. Zato što je poređenjem ulaza u pcache-u sa vrednošću PC utvrđeno da ne postoji ulaz u pcache-u u kome se nalazi vrednost reg PC (ne postoji saglasnot sa PC-om - MISS)

15. T=6
    U taktu 6 je detektovan hazard podataka => rešenje hazarda je **stall**. Rex će biti obrisan, IF & ID će biti zaustavljeni.
16. Neophodno zaustavljanje – zato što instrukcija **beqz R4,10** traži podatak (R4)koji još uvek nije generisan u taktu 6 ,a prosleđivanje je nemoguće jer još nije upisan R4 u Rmem.ALUOUT, pa se zato zaustavlja **beqz R4,10** instrukcija (u stepenu ID) i ove iza **beqz R4,10** dok se ne generiše podatak(R4) od strane instrukcije **lw R4,R0,11** , u stepenu MEM koji koristi instrukcija **beqz R4,10** u EX stepenu.

Drugim rečima – zato što imamo dve uzastopne instrukcije *i* i *i+1* gde *i* generiše podatak u MEM stepenu a instrukcija *i+1* koristi taj podatak u EX stepenu.

Situation in Pipeline

Instruction Address	Mnemonic	Clock 1	2	3	4	5	6
0000	addi R1, R0, 15	IF	ID	EX	MEM	WB	
0001	subi R2, R0, 1		IF	ID	EX	MEM	WB
0002	xori R3, R2, 77			IF	ID	EX	MEM
0003	lw R4, R0, 11				IF	ID	EX
0004	beqz R4, 10					IF	ID
0005	add R5, R1, R2						IF

Situation in Pipeline

Instruction Address	Mnemonic	Clock 2	3	4	5	6	7
0001	subi R2, R0, 1	IF	ID	EX	MEM	WB	
0002	xori R3, R2, 77		IF	ID	EX	MEM	WB
0003	lw R4, R0, 11			IF	ID	EX	MEM
0004	beqz R4, 10				IF	(ID)	ID
0005	add R5, R1, R2					(IF)	IF

17. **CLK1** se koristi za pipline registar **Rid**
18. Sa signalom $\overline{mr3}$ =1 brišemo (**flush**-ujemo) pipeline reg **Rex**
19. T=7

    U EX stepenu pipeline procesora nalazi se **instrukcija bez dejstva (**add R0,R0,R0 – gde je R0=0 uvek!**)**

20. Čita vrednost sa adrese R0+11 a to je vrednost NULA (0000 0000)!!!
21. U taktu 8 vrši se prosleđivanje iz WB->EX tačnije sa polja Rwb.LMD na ulaz jedinice Zero i na taj način instrukcija **beqz R4,10** dobija ispravnu vrednost od strane instrukcije **lw R4,R0,11**

Situation in Pipeline

Instruction Address	Mnemonic	Clock 3	4	5	6	7	8
0002	xori R3, R2, 77	IF	ID	EX	MEM	WB	
0003	lw R4, R0, 11		IF	ID	EX	MEM	WB
0004	beqz R4, 10			IF	(ID)	ID	EX
0005	add R5, R1, R2				(IF)	IF	ID
0006	ori R6, R3, 12						IF

22. T=8

    EQ2=1 jedinice EXP1 zato što je –Rex.rs1 jednak Rwb.rd
23. T=8

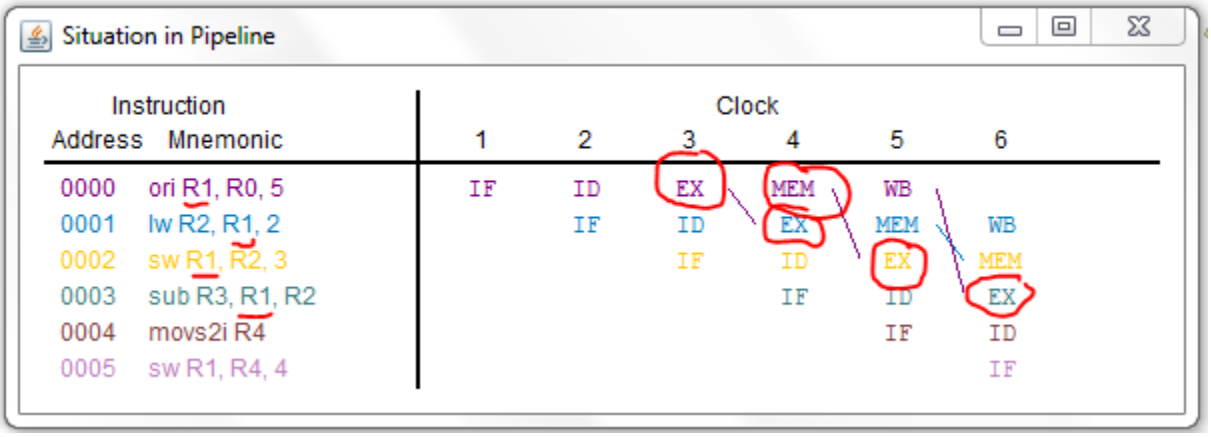
    Na A ulazu ALU jedinice stepena EX je vrednost 4
24. Multiplexesrka jedinica EX1 propušta vrednost 4 na ulaz A ALU jedinice, koristeći upravljački signal Rex.CONDBR za to.
25. T=8

ALU vrši operaciju sabiranja nad operandima A=4 i B=10 i time je rezultat operacije 14 ili hex E – odredišna adresa uslovnog skoka!

26. T=8
    Jeste ispunjen uslov jer imamo COND=1
27. Predikcija NIJE BILA ISPRAVNA; PREDIKCIJA – *branch not taken* , a STVARNI ISHOD - *branch taken;*Na sledeći signal takta BRIŠU se pipeline registry Rid,Rex i Rmem - FLUSHING
28. U taktu 9 uključen je ulaz br 0 keša za predikciju (jedinica pCache)
29. T=10: Nastavlja se od adrese 000e (pogledaj PC !); u T=10 u pipeline – u se nalaze samo 2 instrukcije **sub R8,R4,R1** i **beqz R4,10**
30. Keš za predikciju (jedinica Pcache) nije prazan – ima jedan ulaz popunjen (ulaz br 0); Zato što pCache čuva adrese onih instrukcija uslovnog skoka, za koje se dogodio skok (stvarni ishod-taken) prolazeći kroz pipline, a to je upravo slučaj sa instrukcijom **beqz R4,10**

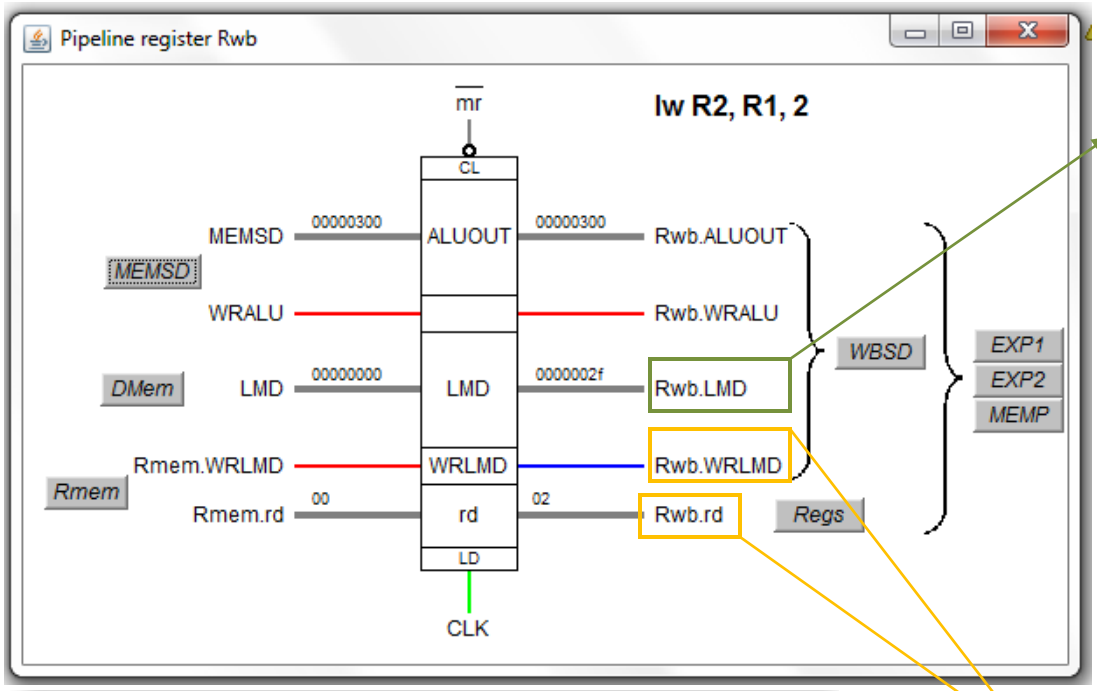
<u>Zadatak 1.2</u>

1. Moguća su 3 istovremena prosleđivanja – jer postoje 3 jedinice za prosleđivanje: EXP1,EXP2,MEMP
2. T=4: U taktu br 4 javlja se vanredni događaj- prosleđivanje iz MEM->EX
3. T=4: U taktu br 4 u EXP1 jedinici ulazi br 0 i br 4 prioritetnog kodera, su aktivni ali zato što je ulaz br 4 većeg prioriteta na izlazu prioCD imamo binarnu vrednost 100 tj decimalno 4 tako što je izlazni signal prioCD[0]=0, prioCD[1]=0,a prioCD[2]=1
4. T=4: Nad registrom R1.
5. T=4: Prosleđivanje je neophodno da ne bi instrukcija lw R2,R1,2 pročitala pogrešnu vrednost registra R1
6. T=5: Vanredni događaj- Prosleđivanje iz WB->EX
7. T=5: Prosleđuje se iz polja Rwb.ALUOUT na ulaz A, ALU jedinice EX stepena i ulaz Zero jedinice
8. T=5: ALU jedinica stepena EX radi operaciju ADD (sabiranja) operanada čije su vrednosti 0000 0005 (ulaz A ALU jedinice) i 0000 0003 (ulaz B ALU jedinice) i time se izračunava adresa memorijske lokacije (R1+3) u koju se smešta sadržaj registra R2
9. T=5: Čita se sa adrese 0000 0007 a pročitana vrednost je 0000 002f
10. T=6:
    Prosleđivanje WB->MEM
    Prosleđivanje iz Rtmp ->EX
    Prosleđivanje WB->EX
11. T=6:
    Nad R2 i R1.
12. T=6:
    Pomoću MEMP vrši se prosleđivanje iz Rwb.LMD na ulaze jedinica DMem i jedinice INtrpt
13. T=6
    R1 se prosleđuje na ulaz A ALU jedinice, EX pipeline stepena, preko jedinice EXP1 za prosleđivanje i multiplexerske jedinice EX1.



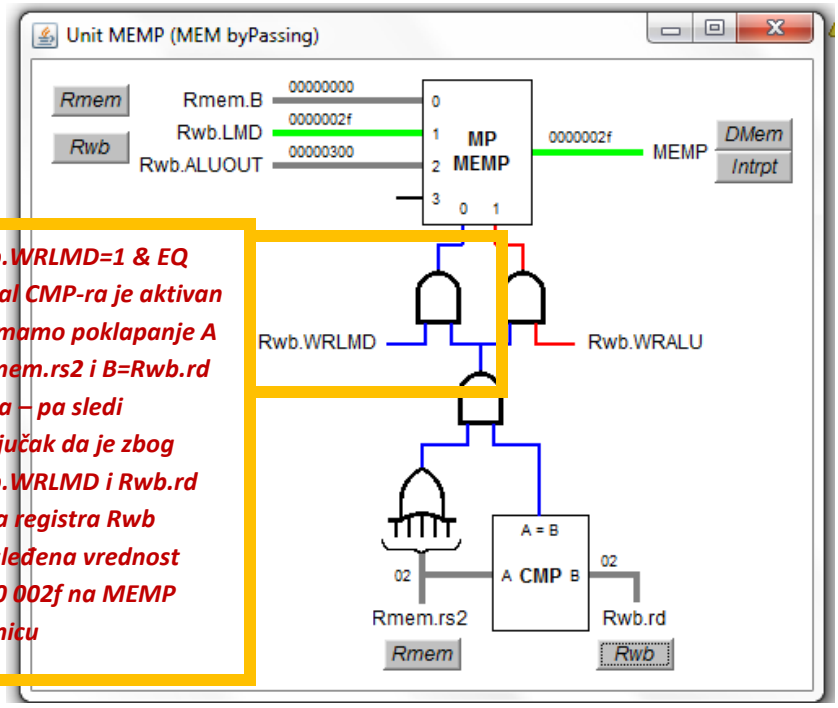
14. T=6

Polja **Rwb.WRLMD** i **Rwb.rd**, pipeline registra Rwb izazivaju prosleđivanje u jedinici MEMP. Vrednosti polja su: **Rwb.WRLMD= 1 i Rwb.rd=02.**



Ovo je **posledica** – prosleđivanje vrednosti 0000 002f !

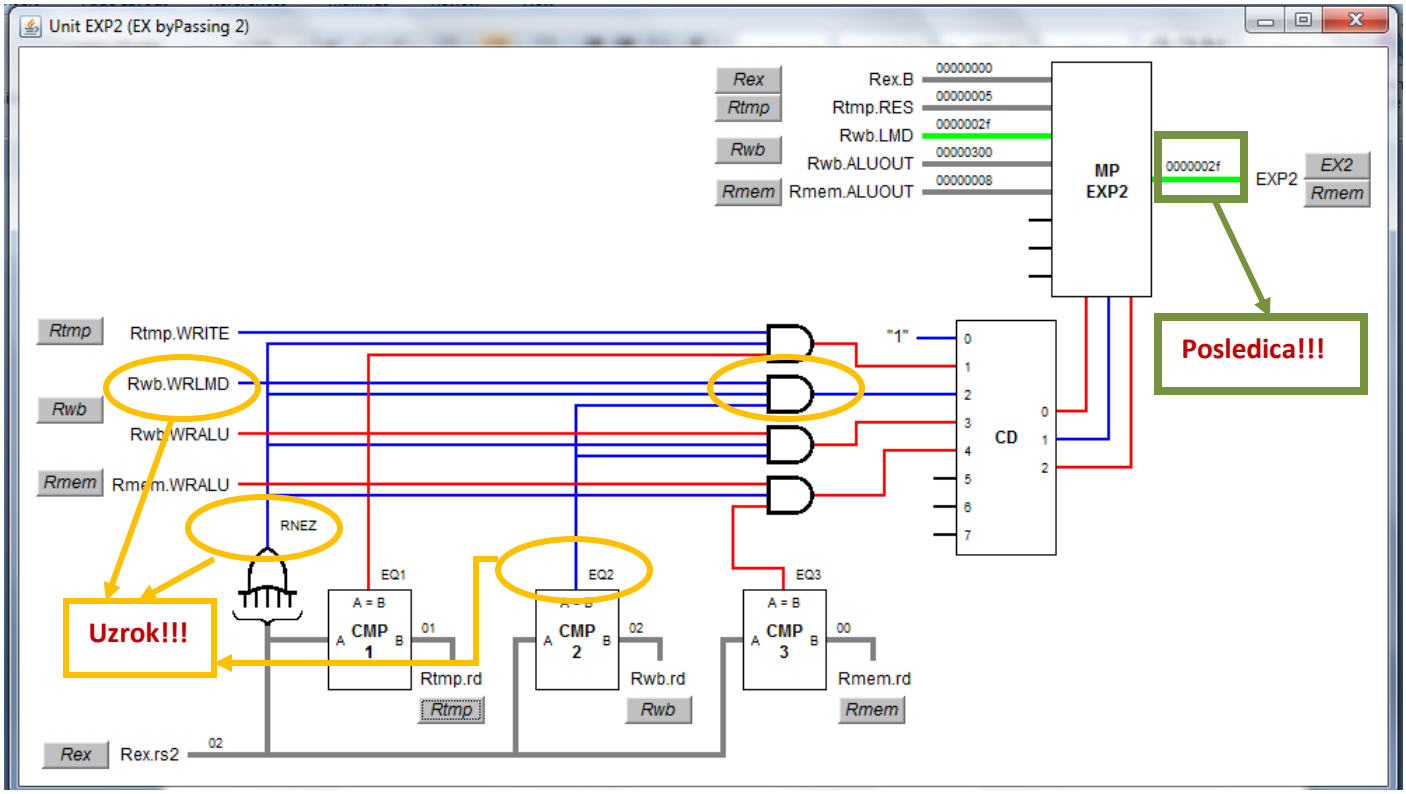
Ovo je **uzrok**-ima ih dva!



Rwb.WRLMD=1 & EQ signal CMP-ra je aktivan jer imamo poklapanje A = Rmem.rs2 i B=Rwb.rd ulaza – pa sledi zaključak da je zbog Rwb.WRLMD i Rwb.rd polja registra Rwb prosleđena vrednost 0000 002f na MEMP jedinicu

15. T=6

Signali **Rwb.WRLMD,RNEZ,EQ2** dovode do prosleđivanja u jedinici EXP2.



16. T=6

U bloku ADD ALU jedinice se JAVLJA PRENOS i to ULAZNI PRENOS (Cin=1), zato što- kad god se pojavi operacija SUB (oduzimanje) blok LOG (nalazi se u bloku ADD jedinice ALU stepena EX)generiše komplement jedinice one vrednosti koja se oduzima (to je u ovom slučaju vred 0000 002f,a njen komplement jedinice je ffff ffd0) da bi se sabiranjem sa prenosom Cin=1 dobila celobrojna vrednost sa znakom u komplementu dvojke. Dakle, formiranje celobrojne vrednosti sa znakom u komplementu dvojke se u ovom procesoru realizuje na sledeći način: LOG blok izračunava komplement jedinice vrednosti EX2 (ona koja se oduzima => vred registra R2 instrukcije *sub*), to se dalje signalom SUB=1 propušta kroz multiplexer MPADD bloka ADD jedinice ALU i šalje na ulaz Din2 sabirača ADD, na ulazu Din1 sabirača ADD se šalje vrednost EX1 (ona od koje se oduzima=>R1 instrukcije *sub*) i na kraju na izlazu ADDOUT , sabirača ADD dobijamo vrednost (Din1+Din2+Cin),koja predstavlja rezultat oduzimanja R1-R2 instrukcije *sub R3,R1,R2.*

ZAKLJUČAK: Kada god imate SUB=1 (oduzimanje) imaćete i Cin=1, da bi dobili komplement dvojke one vrednosti koja se oduzima (EX2)

NAPOMENA:Metoda predstavljanja celog broja sa znakom- ***Komplement jedinice*** nekog binarnog broja se dobija tako što svi bitovi koji su bili 0 postaju 1, a svi koji su bili 1 postaju 0. Primer: komplement jedinice binarnog broja 1001 je 0110. Primer: broj $+51_{10}$ ima reprezentaciju 00110011, a broj -51_{10} 11001100. Kodiranje negativnih celih brojeva komplementom jedinice opisano je izrazom: -X = not(X). Danas se isključivo koristi **komplement dvojke za predstavljanje celog broja sa znakom =>-** X = not(X) + 1

17. T=6

Na ovo pitanje sam odgovorio u 16-tom pitanju:

– Log vrši XOR operaciju sa vrednostima Din1=ffff ffff i Din2= 0000 002f bloka XOR jedinice ALU , čime generiše vrednost EX2 u komplementu jedinice, da bi se sabiranjem te vrednosti (Dout bloka XOR jedinice ALU) sa Cin=1 dobila vrednost EX2 u komplementu dvojke na izlazu ADDOUT bloka ADD jedinice ALU – celobrojna vrednost sa znakom

18. T=7

Instrukcija *sub R3,R1,R2* se nalazi u MEM stepenu pipline-a i pročitana je sa adrese 0000 0003 h(Rmem.PC vrednost).

19. T=8
    Fali još signal EQ2
20. T=9
    Prosleđivanje iz WB u MEM.
21. T=9
    Jedinica MEMP vrši prosleđivanje iz polja Rwb.ALUOUT na ulaze jedinica DMEM i Intrpt
22. T=9
    Na data ulazu jedinice DMem je vrednost 0000 0300. Da nema prosleđivanja na tom ulazu bila bi vrednost 0000 0000 (to je vrednost polja Rmem.B)
23. U taktu br 5 **vrši se upis** vred reg R1 tj. vrednost 0000 0005, a u taktu 6 je **upisana** vred reg R1.

Values of General-Purpose Registers

Reg.	Hex.	Decimal	Reg.	Hex.	Decimal
R0	0x00000000	0	R16	0x00000000	0
R1	0x00000000	0	R17	0x00000000	0
R2	0x00000000	0	R18	0x00000000	0
R3	0x00000000	0	R19	0x00000000	0
R4	0x00000000	0	R20	0x00000000	0
R5	0x00000000	0	R21	0x00000000	0
R6	0x00000000	0	R22	0x00000000	0
R7	0x00000000	0	R23	0x00000000	0
R8	0x00000000	0	R24	0x00000000	0
R9	0x00000000	0	R25	0x00000000	0
R10	0x00000000	0	R26	0x00000000	0
R11	0x00000000	0	R27	0x00000000	0
R12	0x00000000	0	R28	0x00000000	0
R13	0x00000000	0	R29	0x00000000	0
R14	0x00000000	0	R30	0x00000000	0
R15	0x00000000	0	R31	0x00000000	0
			New value of R1:	0x00000005	5

Situation in Pipeline

Instruction Address	Mnemonic	Clock 1	2	3	4	5
0000	ori R1, R0, 5	IF	ID	EX	MEM	WB
0001	lw R2, R1, 2		IF	ID	EX	MEM
0002	sw R1, R2, 3			IF	ID	EX
0003	sub R3, R1, R2				IF	ID
0004	movs2i R4					IF

U CLK=5 u stepenu WB vrši se upis nove vred R1 u reg fajl

24. Instrukcija sw R1,R4,4 čita ispravnu vred reg R1 bez prosleđivanja. Instrukcija sw čita R1 i to je vred 0000 0005 u taktu 7
25. Instrukcije lw R2, R1,2, sw R1,R2,3, sub R3, R1, R2 čitaju pogrešne vrednosti reg R1 iz reg fajla

26. U taktu 6 se upisuje konačna vrednost reg R2 i to 0000 002f

Values of General-Purpose Registers

Reg.	Hex.	Decimal	Reg.	Hex.	Decimal
R0	0x00000000	0	R16	0x00000000	0
R1	0x00000005	5	R17	0x00000000	0
R2	0x00000000	0	R18	0x00000000	0
R3	0x00000000	0	R19	0x00000000	0
R4	0x00000000	0	R20	0x00000000	0
R5	0x00000000	0	R21	0x00000000	0
R6	0x00000000	0	R22	0x00000000	0
R7	0x00000000	0	R23	0x00000000	0
R8	0x00000000	0	R24	0x00000000	0
R9	0x00000000	0	R25	0x00000000	0
R10	0x00000000	0	R26	0x00000000	0
R11	0x00000000	0	R27	0x00000000	0
R12	0x00000000	0	R28	0x00000000	0
R13	0x00000000	0	R29	0x00000000	0
R14	0x00000000	0	R30	0x00000000	0
R15	0x00000000	0	R31	0x00000000	0

New value of R2: 0x0000002f 47

Situation in Pipeline

Instruction Address	Mnemonic	Clock 1	2	3	4	5	6
0000	ori R1, R0, 5	IF	ID	EX	MEM	WB	
0001	lw R2, R1, 2		IF	ID	EX	MEM	WB
0002	sw R1, R2, 3			IF	ID	EX	MEM
0003	sub R3, R1, R2				IF	ID	EX
0004	movs2i R4					IF	ID
0005	sw R1, R4, 4						IF

U CLK=6 u stepenu WB vrši se upis nove vred R2 u reg fajl

27. Nijedna instrukcija ne čita pravu vrednost R2 reg bez prosleđivanja
28. Instrukcije sw R1,R2,3 i sub R3,R1,R2 čitaju pogrešne vred reg R2
29. Protočna obrada NE MORA DA SE ZAUSTAVI jer se hazard podataka u ovoj situaciji rešava PROSLEĐIVANJEM
30. Kada bi se premutovali reg argument instrukcije sw R1,R2,3 odgovor na prethodno pitanje **ne bi bio isti** već bi glasio – pipeline MORA DA SE ZAUSTAVI u taktu 4, zato što lw instrukcija u CLK=4 u EX stepenu još uvek računa novu vrednost reg R2 (nije je generisala pa ne može ni da prosleđuje tu vrednost instrukciji sw); zaustavljanjem pipe-a u 4 taktu zaustavlja instrukciju sw i sve druge instrukcije iza sw instrukcije jednu periodu signala takta dok instrukcija lw u stepenu EX ne izračuna R1+2 (nova vred R2)