

Zadatak 2.1

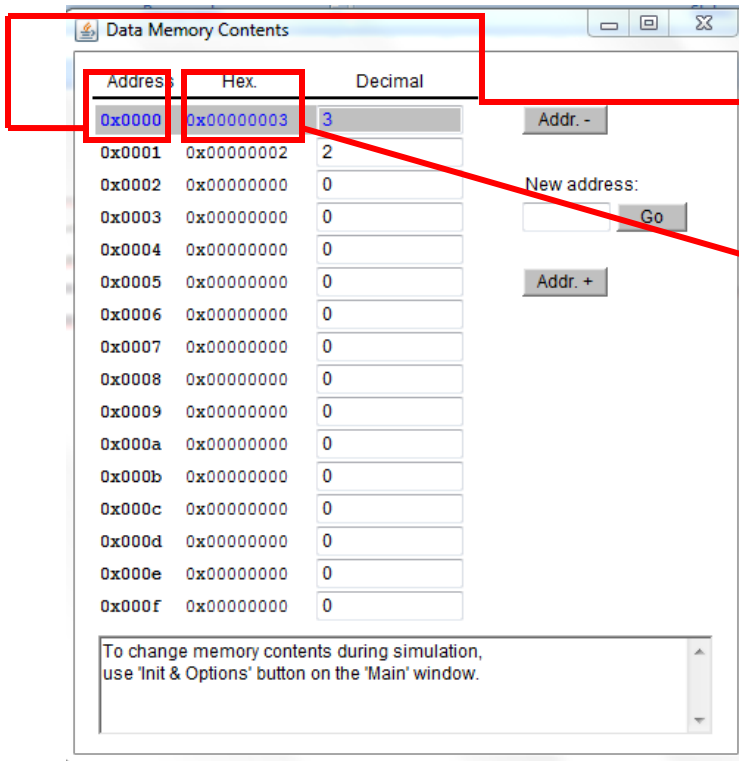
1. T=5:

Jedinica WBSD propušta vred 0000 0003

Potiče iz polja Rwb.LMD

Signal Rwb.WRLMD izaziva propuštanje ove vrednosti

2. Vrednost 0000 0003 koja je pročitana u taktu 5 (prethodno pitanje) predstavlja sadržaj memorijske lokacije (R0+0) = 0000 h i ujedno vrednost kojom se puni reg R1.



Address	Hex	Decimal
0x0000	0x00000003	3
0x0001	0x00000002	2
0x0002	0x00000000	0
0x0003	0x00000000	0
0x0004	0x00000000	0
0x0005	0x00000000	0
0x0006	0x00000000	0
0x0007	0x00000000	0
0x0008	0x00000000	0
0x0009	0x00000000	0
0x000a	0x00000000	0
0x000b	0x00000000	0
0x000c	0x00000000	0
0x000d	0x00000000	0
0x000e	0x00000000	0
0x000f	0x00000000	0

Adresa memorijske lokacije sa koje se čita (za instrukciju lw R1,R0+0)

Vrednost memorijske lokacije 0000 (za instrukciju lw R1,R0+0)

3. T=7

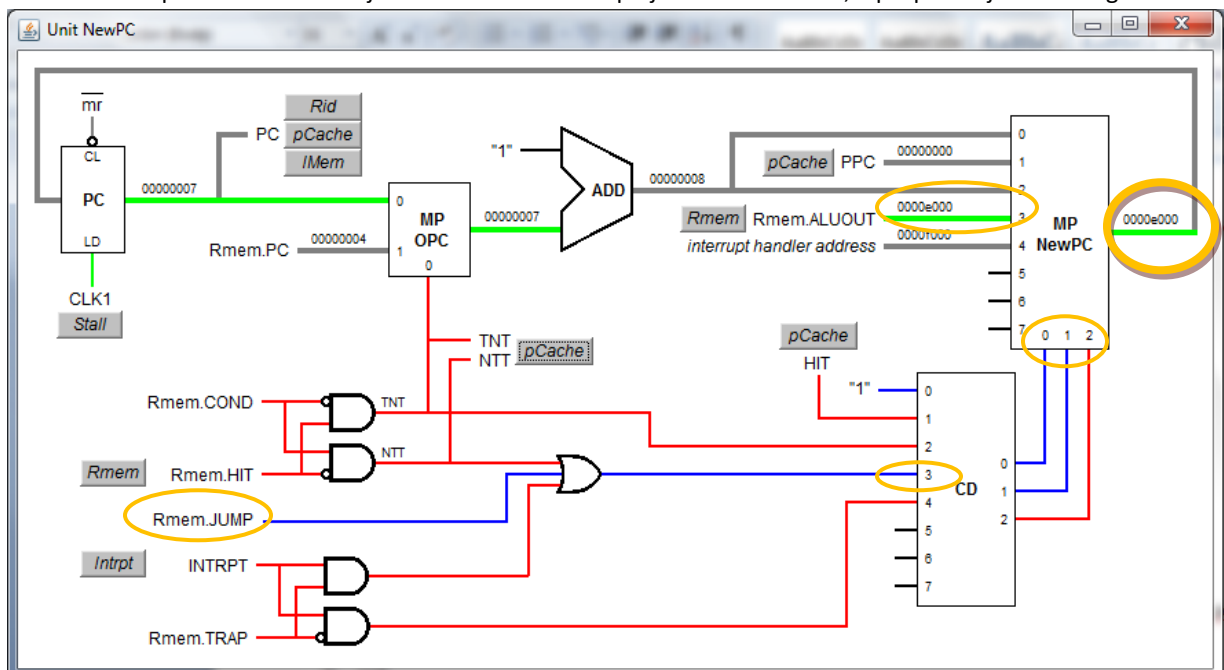
U taktu 7 događa se prva vanredna situacija-prosleđivanje iz stepena MEM->EX, vrši ga jedinica EXP1, iz polja Rmem.ALUOUT na ulaz A jedinice ALU.

4. T=8

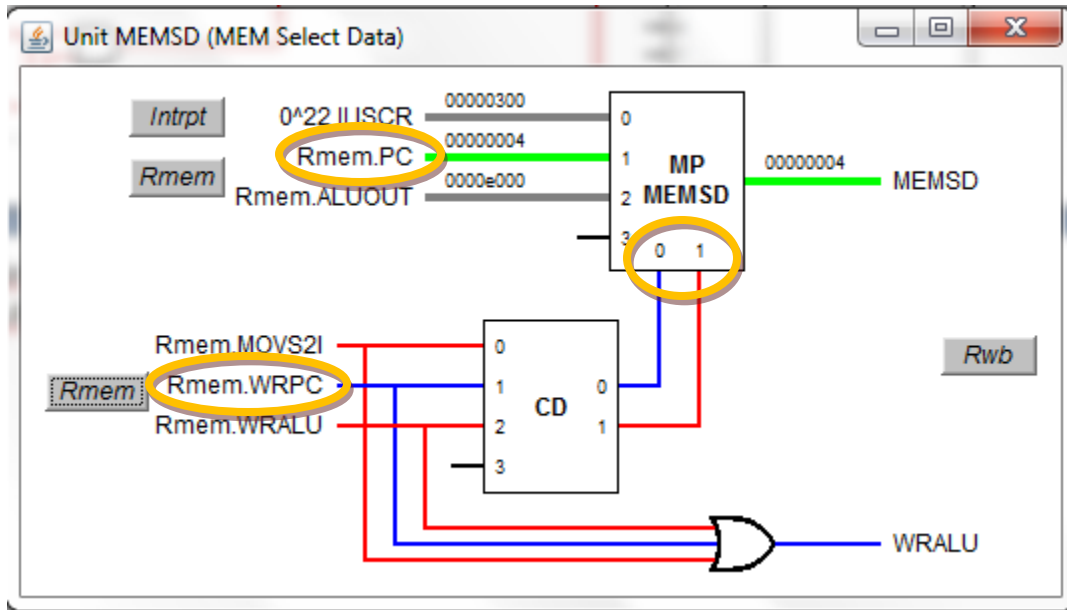
Bezuslovni skok na potprogram ! Registri Rid,Rex i Rmem će biti obrisani na sledećem taktu.

5. T=8

Na izlazu multiplexera MPNewPC je 0000 e000.Potiče iz polja Rmem.ALUOUT, a propuštanje izaziva signal Rmem.JUMP



6. Vrednost 0000 e000 predstavlja adresu prve instrukcije prekidne rutine, a to je instrukcija `sw R29,R1,1`
7. Potiče iz polja `Rmem.PC`, a signal `Rmem.WRPC` izaziva propuštanje.



8. T=9: Vrednost `Rmem.PC` koja predstavlja staru vrednost PC-a upisujemo u registar R30, koji se kasnije koristi u instrukciji `jsr R30, 1` za povratak iz potprograma u glavni program nastavljajući od naredne instrukcije posle instrukcije `jsr R30,R17,0-` znači da se `jsr R30, 1` instrukcija koristi kao **INSTRUKCIJA POVRATKA IZ PREKIDNE RUTINE-RTS**

Reg.	Hex.	Decimal	Reg.	Hex.	Decimal
R0	0x00000000	0	R16	0x00000000	0
R1	0x00000003	3	R17	0x0000e000	57344
R2	0x00000002	2	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	0x0000f800	63488
R14	0x00000000	0	R30	0x00000000	0
R15	0x00000000	0	R31	0x00000000	0
			New value of R30:		0x00000004 4

Instruction Address	Mnemonic	4	5	6	7	8	9
0003	li1 R17, R0, 224	IF	ID	EX	MEM	WB	
0004	jsr R30, R17, 0		IF	ID	EX	MEM	WB
0005	add R5, R1, R2			IF	ID	EX	Flush
0006	xori R6, R5, 255				IF	ID	Flush
0007	and R7, R5, R6					IF	Flush
e000	sw R29, R1, 1						IF

`jsr R30,R17,0` instrukcija u stepenu WB vrši upis u reg fajl novu vred reg R30 (vrednost 0000 0004-stara vr PC-a), a u PC upisuje novu vred R17+0 – adresu prve instrukcije potprograma

9. Nakon poziva potprograma izvršen je upis u keš za predikciju (pCache) zato što je ispunjen uslov za skok instrukcije bnez R1,-2 iako je predikcija bila "Not taken"

Entry Address	Branch Instr. Address	Branch Target Address	Tag
0x000	0x00000000	0x00000000	0
0x001	0x00000000	0x00000000	0
0x002	0x00000000	0x00000000	0
0x003	0x00000000	0x00000000	0
0x004	0x00000000	0x00000000	0
0x005	0x00000000	0x00000000	0
0x006	0x00000000	0x00000000	0
0x007	0x00000000	0x00000000	0
0x008	0x00000000	0x00000000	0
0x009	0x00000000	0x00000000	0
0x00a	0x00000000	0x00000000	0
0x00b	0x00000000	0x00000000	0
0x00c	0x00000000	0x00000000	0
0x00d	0x00000000	0x00000000	0
0x00e	0x00000000	0x00000000	0
0x00f	0x00000000	0x00000000	0

New value of location 0x000:
 0x0000e005 0x0000e003 17

10. T=

11. T=14: U taktu 14 bnez ulazi u pipe po prvi put. Predikcija je – NOT TAKEN, zato što se instrukcija bnez R1,-2 ne nalazi u kešu za predikciju (pCache) pa je to analogno predviđanju NOT TAKEN

12. T=17

Predikcija je POGREŠNA u prvom prolasku kroz petlju.

13. Na izlazu MPNewPC je 0000 e003. Potiče iz polja Rmem.ALUOUT. Ta vrednost predstavlja adresu instrukcije add R3,R3,R2 potprograma

14. T=17

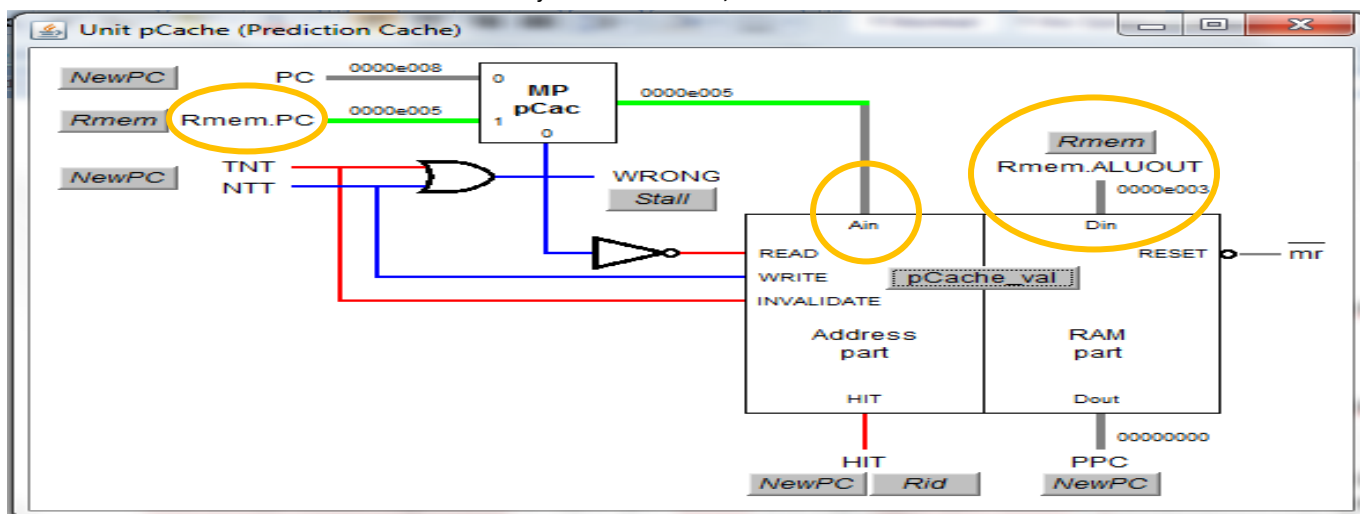
U jedinici NewPC je aktivan NTT signal . On je aktivan pod uslovom da je Rmem.COND=1(signal USLOVA za instrukcije uslovnog skoka: 1-ispunjen uslov, a 0-nije ispunjen uslov) i Rmem.HIT=0 (signal saglasnosti za keš pCache : HIT=1 – ako je instrukcija uslovnog skoka u pCache-u, u suprotnom je HIT=0)

15. T=17

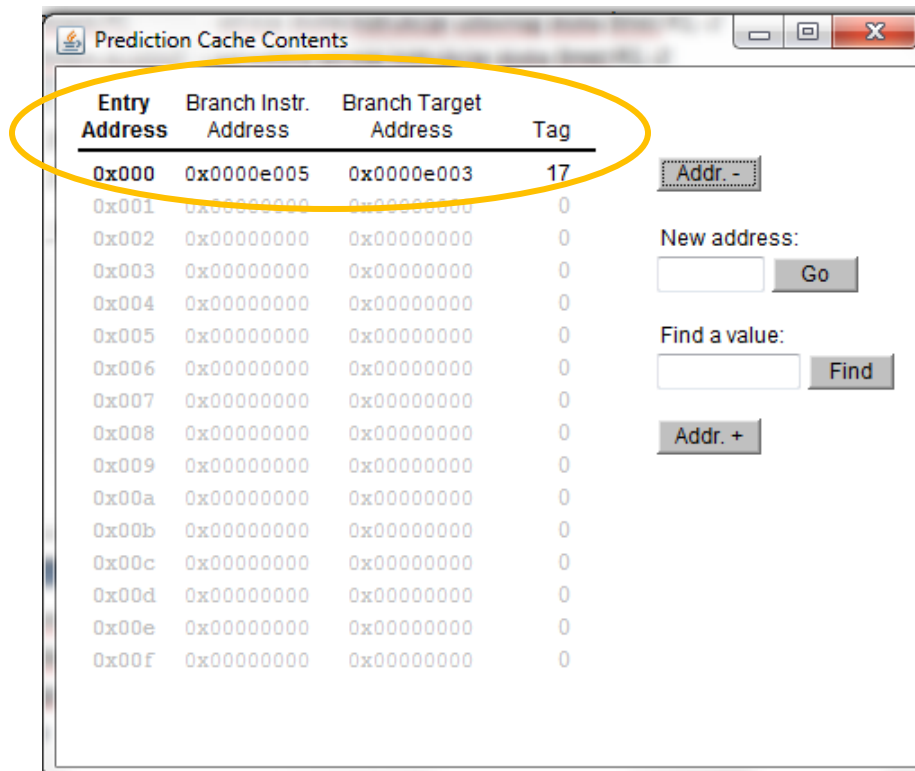
Address & Data ulazi pCache-a:

Ain=Rmem.PC - adresa skoka instrukcije uslovnog skoka *bnez R1,-2*

Din=Rmem.ALUOUT - odredišna adresa instrukcije skoka *bnez R1,-2*



Rmem.PC & Rmem.ALUOUT polja su dovedena na address & data ulaze pCache-a, respektivno zato što je predikcija NOT TAKEN a skok se ipak dogodio ,pa je NTN=1, što znači da se kreira novi ulaz u pCache-u tako što se u odgovarajući ulaz pCache-a upisuje redom adresa instrukcije skoka, odredišna adresa instrukcije, vrednost tag polja



Entry Address	Branch Instr. Address	Branch Target Address	Tag
0x000	0x0000e005	0x0000e003	17
0x001	0x00000000	0x00000000	0
0x002	0x00000000	0x00000000	0
0x003	0x00000000	0x00000000	0
0x004	0x00000000	0x00000000	0
0x005	0x00000000	0x00000000	0
0x006	0x00000000	0x00000000	0
0x007	0x00000000	0x00000000	0
0x008	0x00000000	0x00000000	0
0x009	0x00000000	0x00000000	0
0x00a	0x00000000	0x00000000	0
0x00b	0x00000000	0x00000000	0
0x00c	0x00000000	0x00000000	0
0x00d	0x00000000	0x00000000	0
0x00e	0x00000000	0x00000000	0
0x00f	0x00000000	0x00000000	0

16. T=20:

U taktu 20 instrukcija *bnez R1,-2* po drugi put ulazi u pipeline i sada je predikcija TAKEN – zato što smo u IF stepenu pipeline procesora u bloku pCache(keš za predikciju instrukcije uslovnog skoka) poređenjem vrednosti reg PC, sa ulazima pCache utvrdili saglasnost(HIT=1) sa odgovarajućim ulazom za instrukciju (*bnez R1,-2*) što znači da je instrukcija *bnez R1,-2* već jednom ulazila u pipe i da je za nju stvarni ishod skoka bio TAKEN

17. T=20

Na izlazu multiplexera MP NewPC je vrednost 0000 e003, a signal HIT=1 izaziva propuštanje ove vrednosti.

18. T=20:

Vrednost na izlazu multiplexera MP NewPC 0000 e003, predstavlja polje PPC iz bloka pCache i predstavlja predikciju vrednosti registra PC (PPC-Predicted PC), tj. prognozirano odredište skoka (*bnez R1,-2*)

19. T=21:

Pošto je trenutna vrednost reg PC=e003 znači da se sa adrese e003 učitava instrukcija *add R3,R3,R2* koja je u taktu 21 ušla u pipe. Odstupili smo od sekvencijalnog izvršavanja instrukcija zbog predikcije TAKEN (biće skoka) instrukcije uslovnog skoka pa se u PC upisuje odredišna adresa skoka *bnez R1,-2*.

20. Predikcija je TAČNA. Sadržaj keša za predikciju(pCache)ostaje isti – NE MENJA SE!

21. T=22:

U taktu 22 pCache NIJE PRAZAN: na adresi 0000 nalazi se zapis 0000e005(adresa instrukcije uslovnog skoka), 0000 e003(odredišna adresa instrukcije uslovnog skoka) i 17(tag polje)

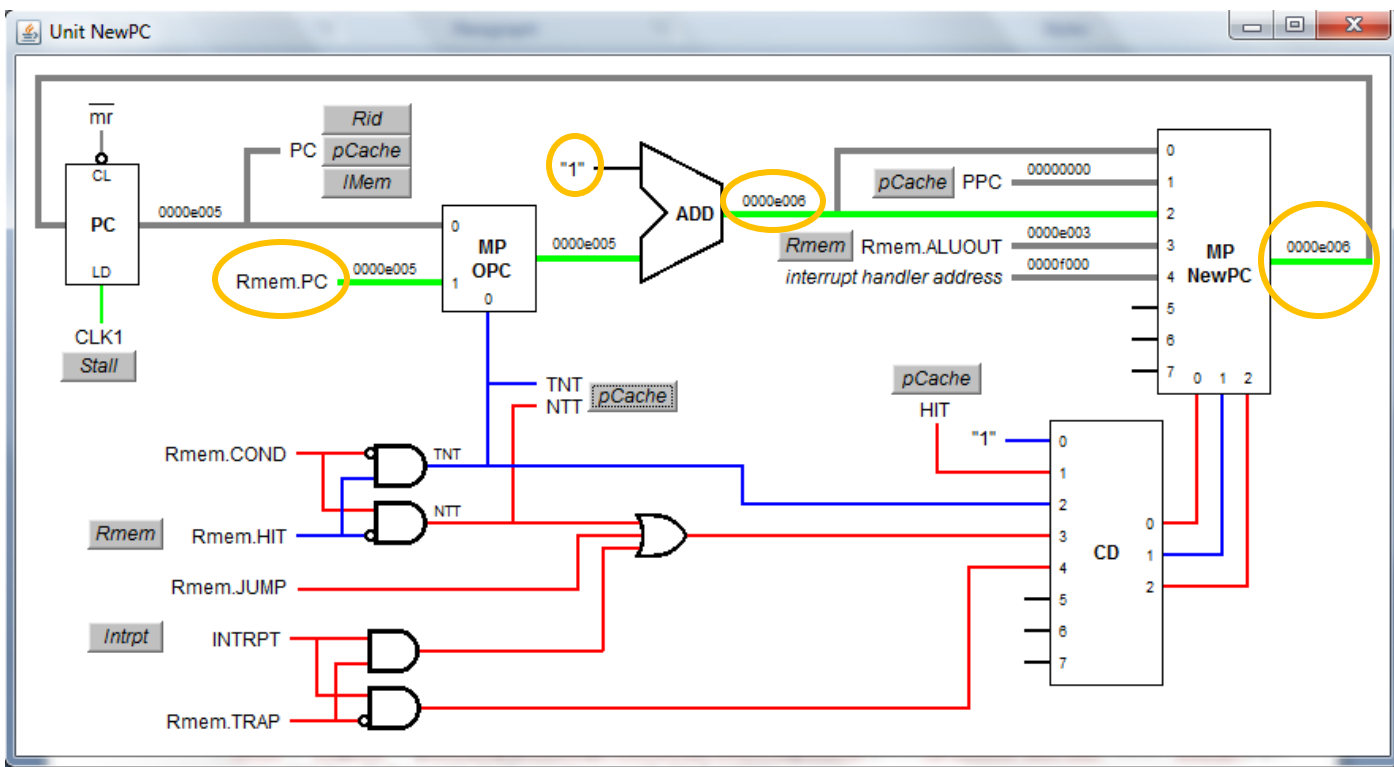
22. T=23:

Predikcija instrukcije uslovnog skoka *bnez R1,-2* je TAKEN (biće skoka)- zato što imamo HIT=1 u pCache-u

23. T=26:

U taktu 26 događa se **promašaj predikcije** instrukcije uslovnog skoka, pa se na sledeći takt radi FLUSH-ovanje pipe registra Rid,Rex & Rmem tj . brišu se stepeni ID, EX & MEM.

24. T=26:

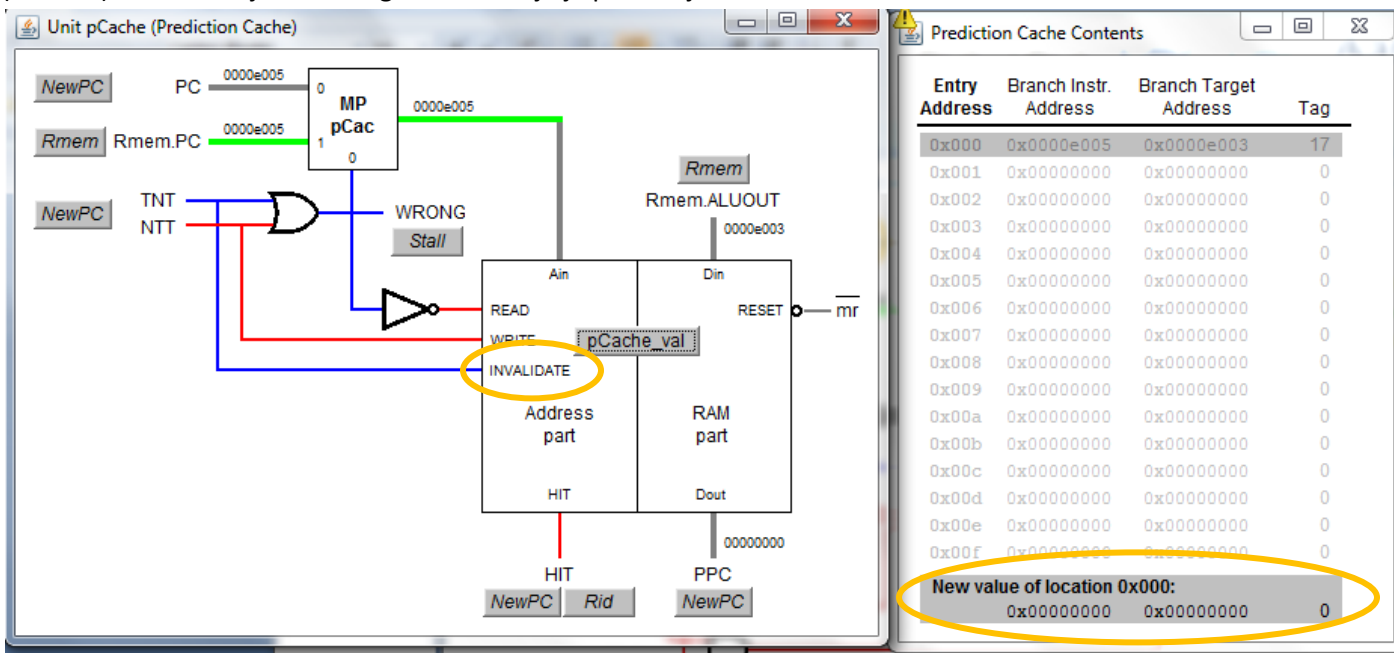


Na izlazu MP NewPC je e006, a dobija se na izlazu bloka ADD koji sabira Rmem.PC=e005 i 1 i predstavlja vrednost koja će se na sledećem taktu upisati u PC i to će biti adresa instrukcije lw R1, R29, 0, koja ulazi u pipe

25. T=26: to je signal TNT- on je uključen pod uslovom da je **Rmem.HIT=1 & Rmem.COND=0**

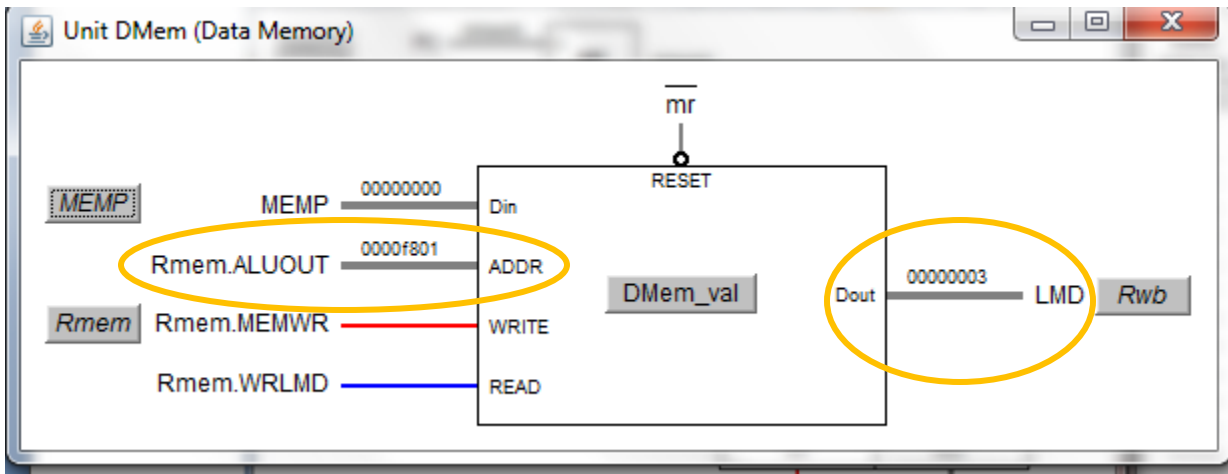
26. T=26:

Kontrolni ulaz INVALIDATE je uključen ! INVALIDATE se uključuje kako bi poništili odgovarajući ulaz u pCache-u (briše se) za instrukciju uslovnog skoka za koju je predikcija bila TAKEN a stvarni ishod NOT TAKEN.



27. T=30:

Sa adrese 0000 f801. Pročitana vrednost je 0000 0003. Svrha instrukcije lw je da napuni R1 reg vrednošću 3_{DEC}.



28. Odredišna adresa instrukcije jr R30,1 je 0000 0005. Ova instrukcija predstavlja instrukciju povratka iz potprograma

Reg.	Hex.	Decimal	Reg.	Hex.	Decimal
R0	0x00000000	0	R16	0x00000000	0
R1	0x00000000	0	R17	0x0000e000	57344
R2	0x00000002	2	R18	0x00000000	0
R3	0x00000006	6	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	0x00005001	63489
R14	0x00000000	0	R30	0x00000004	4
R15	0x00000000	0	R31	0x00000000	0

Instrukcija jr R30,1 čita sadržaj reg R30, sabira tu vrednost sa 1-com i na kraju upisuje rezultat (R30+1) u PC – to je adresa instrukcija koja se ubacuje u pipeline

29. Potprogram počinje sa adrese e000:

Address	Hex.	Mnemonic	Comment
0xe000	0x2183a101	sw R29, R1, 1	
0xe001	0x02f7a001	addui R29, R29, 1	
0xe002	0x000c0000	add R3, R0, R0	
0xe003	0x000c6200	add R3, R3, R2	R3 <- R3 + R2
0xe004	0x01842001	subi R1, R1, 1	R1 <- R1 - 1
0xe005	0x298020fe	bnez R1, -2	skok na pocetak petlje
0xe006	0x2087a000	lw R1, R29, 0	
0xe007	0x03f7a001	subui R29, R29, 1	
0xe008	0x2a83c001	jr R30, 1	
0xe009	0x00000000	add R0, R0, R0	

Petlja počinje od adrese e003 do e005 :
loop
add R3,R3,R2
subi R1,R1,1
bnez R1,-2
end_loop

- (1) sw – čita R1 i tu vred(3_{DEC}) upisuje u MemLok[R29+1]=MemLok[f801] =3
- (2) addui - upisuje u R29=R29+1 tj R29=f8001 (takt 14)
- (3) add – R3=R0+R0 = 0 (takt 15)
- (4) add – R3=R3+R2 = 0+2 = 2 (takt 16)
- (5) subi – R1=R1-1 = 3-1=2
- (6) bnez – (R1 != 0)? true => PC=PC-2=e005-2=e003; false => PC=PC+1= e006
- (7) add – R3=R3+R2=2+2=4
- (8) subi – R1=R1-1=2-1=1
- (9) bnez - (R1 != 0)? True =>PC=PC-2=e003
- (10) add – R3=R3+R2=4+2=6
- (11) subi – R1=R1-1=1-1=0
- (12) bnez - (R1 != 0)?false => PC=e006
- (13) lw – čita R29 = f801, sabira sa 0 i upisuje u R1=3 - vred MemLok[f801]
- (14) subui – R29=R29-1=f801-1=f800
- (15) jr – čita R30 (R30=4), sabira tu vred sa 1 (R30+1) i upisuje rezultat u PC (PC=R30+1=0005 – adresa sledeće instr u glavnom programu, posle jsr R30,R17,0 instrukcije)

I put se izvršava

II put se izvršava

III put se izvršava

Na kraju potprograma imamo sadržaj reg u register fajlu:

Values of General-Purpose Registers					
Reg.	Hex.	Decimal	Reg.	Hex.	Decimal
R0	0x00000000	0	R16	0x00000000	0
R1	0x00000003	3	R17	0x0000e000	57344
R2	0x00000002	2	R18	0x00000000	0
R3	0x00000006	6	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	0x0000f800	63488
R14	0x00000000	0	R30	0x00000004	4
R15	0x00000000	0	R31	0x00000000	0

R29 – se koristi kao **STEK POINTER**

R30 – se koristi da pamti adresu **povratka iz potprograma** (!!! ne iz prekidne rutine: za adresu povratka iz prekidne rutine se koristi R31 koji se ne može menjati)

Iz priloženog sledi :

Kada se ulazi u potprogram jsr računa adresu skoka i pamti u R17(početna adresa potprograma e000) i upisuje u R30 povratnu adresu iz potprograma . u potprogramu se na Mem Lok[f801] upisuje vred reg R1 a to je vred 3. Menja se sadržaj reg R29 (stek pointer)sa F800 na F801 – sada SP pokazuje na sledeću slobodnu lok u memoriji a to je f801.Menja sadržaj reg R3 (upisuje 0),a onda ulazi u petlju (e003-e005) u kojoj se R3 puni vred 2, dec se R1i proverava R1 da li je !=0 (loop se izvršava 3 puta. Kada R1=0 čita se sa vrha steak(R29=f800) vrednost 3 i upisuje u R1, ažurira R29(stek pointer – sada pokazuje na f800) i izvršava instrukcija povratka iz potprograma tako što čita R30 i u PC upisuje R30+1

Zaključak : Potprogram vrti ovu petlju tri puta - to je glavni deo operacije

30. T=40

Nema zapisa u pCache-u zato što je poslednji put jedan jedini ulaz INVALIDOVAN u p Cache-u zbog promašene predikcije TNT