(1)

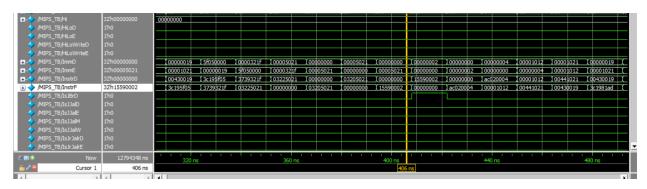


图 1 分支前-1

/MIPS_TB/MemWriteM	1'h0											Į.							
⊢ <pre>/MIPS_TB/NPCD</pre>	32'h00003074		0000305¢	(0000	060 (	00003064	0000306	3 (00	00306c	00003070	00003074		0000307c	0000307c	(000	03084	00003088	(0000308c	(000030
⊢ <pre>/MIPS_TB/OpD</pre>	6'h00	00		(Of	)(	Od	(00						05	(00)	(2b		00		
⊢ <pre>/MIPS_TB/PCF</pre>	32'h00003070		00003058	(0000	05c (	00003060	0000306	4 (00	003068	0000306c	(00003070		00003074	0000307c	(000	03080	00003084	(00003088	(000030
MIPS_TB/PCPlus4D	32'h00003070		00003058	(0000	05c (	00003060	0000306	4 (00	003068	0000306c	00003070		00003074	00003078	000	03080	00003084	(00003088	000030
⊢ <pre>/MIPS_TB/PCPlus4F</pre>	32'h00003074		0000305c	(0000	060 (	00003064	0000306	3 (00	00306c	00003070	00003074		00003078	00003080	(000	03084	00003088	(0000308c	000030
⊢/MIPS_TB/PCPlus8E	32'h00003070		00003058	(0000	05c (	00003060	0000306	4 (00	003068	0000306c	00003070		00003074	00003078	(000	0307c	00003084	(00003088	000030
	32'h0000306c		00003054	(0000	058 (	0000305c	00003060	) (00	003064	00003068	0000306c		00003070	00003074	(000	03078	0000307c	00003084	(000030
	32'h00003068		00003050	(0000	054	00003058	0000305	(00	003060	00003064	00003068		0000306c	00003070	(000	03074	00003078	0000307c	(000030
⊢ <pre>/MIPS_TB/RdD</pre>	5'h00		00	(0b	)(	06	(0a	(00	)	0a	(00						02		(00
⊢/MIPS_TB/RdE	5'h0a	02		(00)	χı	0b	(06	( Oa	1	100	)(0a	T)	(00					(02	

图 2 分支前-2

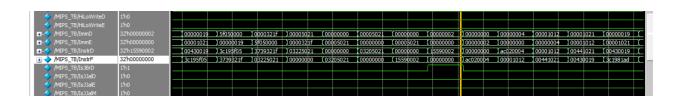


图 3 分支中-1

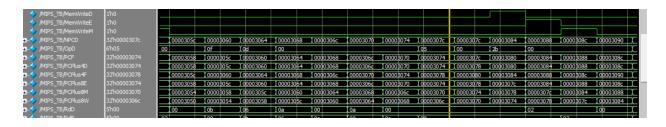


图 4 分支中-2



图 5 分支后-1

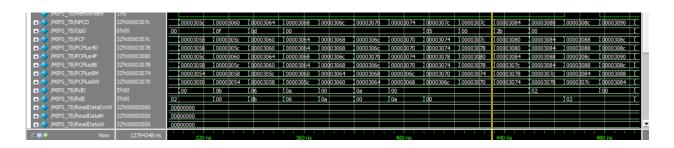


图 6 分支后-2

(2)



图 7 分支前-1

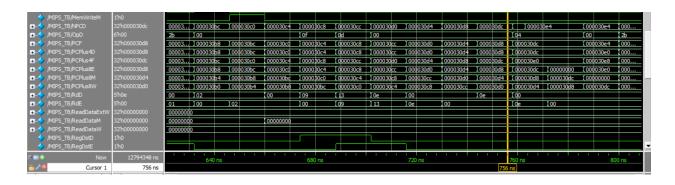


图 8 分支前-2



图 9 分支中-1

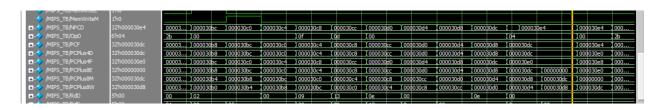


图 10 分支中-2

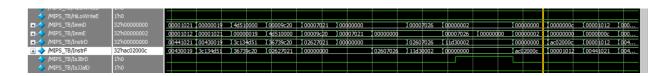


图 11 分支后-1

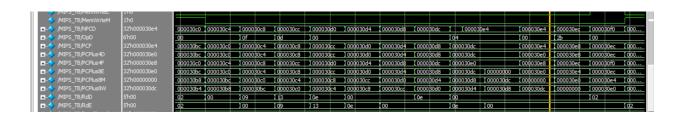
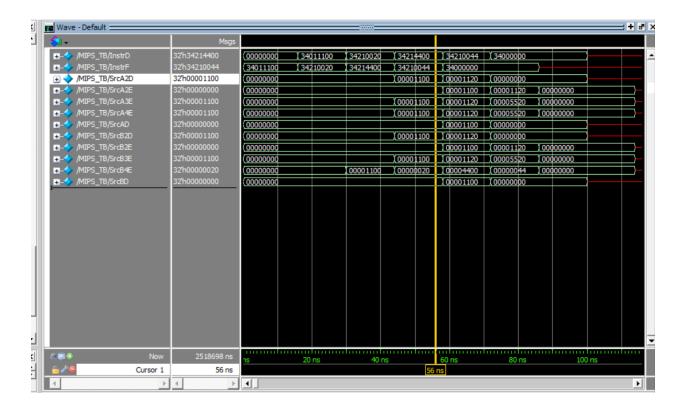


图 12 分支后-2

在EX阶段判决是否分支,因此会引起流水线暂停,损耗一个周期。 若在ID阶段已经判断是否分支,则不会流水线暂停。



## 图 13 数据前推

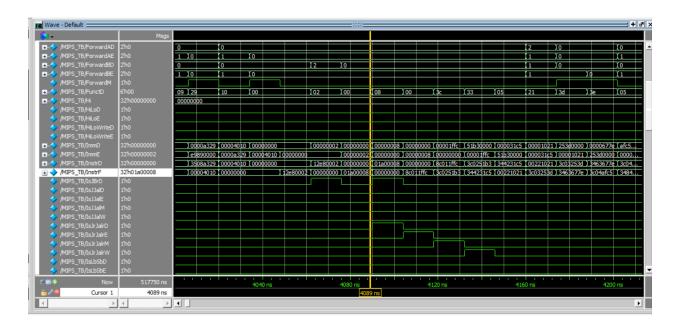


图 14 跳转前-1

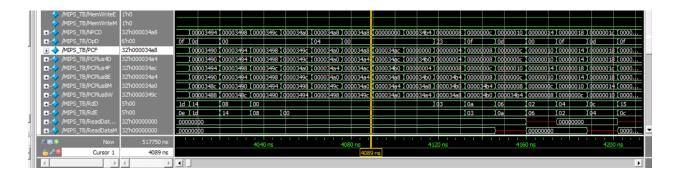


图 15 跳转前-2

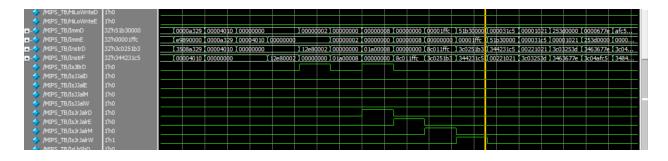


图 16 跳转后-1



图 17 跳转后-2

控制相关是指流水线中的转移指令或者其他需要改写 PC 的指令造成的相关。这些指令改写了 PC 的值,所以导致后面已经进入流水线的几条指令无效,比如:如果转移指令在流水线的执行阶段进行转移条件判断,在发生转移时,会导致当前处于取指、译码阶段的指令无效,需要重新取指。

也就是说,在流水线执行阶段进行转移判断,并且转移发生,那么会有2条无效指令,导致浪费了两个时钟周期。为了减少损失,规定转移指令后面的指令位置为"延迟槽",延迟槽中的指令被称为"延迟指令"(也可称之为"延迟槽指令")。延迟指令总是被执行,与转移发生与否没有关系。

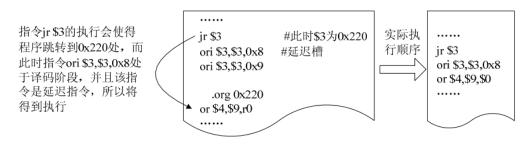


图 8-2 引入延迟槽以减少转移带来的损失

- ●取指:取出指令存储器中的指令, PC 值递增,准备取下一条指令。
- ●译码:对指令进行译码,依据译码结果,从32个通用寄存器中取出源操作数,有的指令要求两个源操作数都是寄存器的值,比如 or 指令,有的指令要求其中一个源操作数是指令中立即数的扩展,比如 ori 指令,所以这里有两个复用器,用于依据指令要求,确定参与运算的操作数,最终确定的两个操作数会送到执行阶段。
- ●执行阶段:依据译码阶段送入的源操作数、操作码,进行运算,对于 ori 指令而言,就是进行逻辑"或"运算,运算结果传递到访存阶段。

- ●访存阶段:对于 ori 指令,在访存阶段没有任何操作,直接将运算结果向下传递到回写阶段。
  - ●回写阶段:将运算结果保存到目的寄存器。