CALab6_Report

PB19051183 吴承泽

一、 Tomasulo算法模拟器

设置好Tomasulo算法模拟器,如下所示:



开始进行仿真:

回答问题:

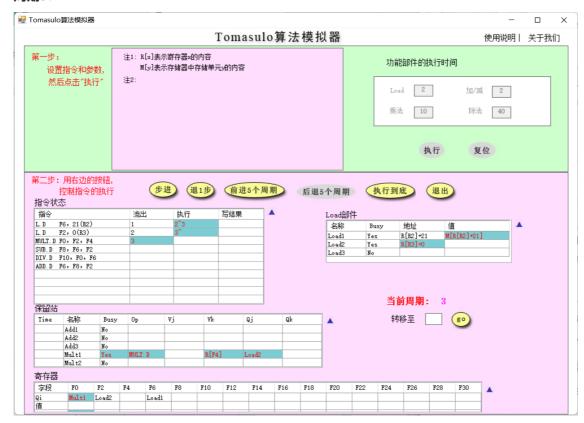
- 1.分别截图 (当前周期2和当前周期3) , 请简要说明load部件做了什么改动
 - 周期2:

					Τ	oma	asulo	算法	模拟	器						使用说明	关于我	妣
	置指令和参 舌点击"执		注1: R[x]表示寄存器始的内容 M[y]表示存储器中存储单元始内容 注2:								[功能	部件的	执行时	间			7
711/2	н											Loa	2		ta/3	或 2		
												乘法	10		除	± 40		
											l							
													1	丸行	3	正位		
	用右边的	balah ker																
指令状态	控制指令(态	的执行	(H	进	图1步	前进	5个周	期	后退5	个周期		执行到	底	退出				
指令			流出	执行		写结果		<u> </u>		Load≧R住	±							
	F6, 21(R2))	流出 1	执行 2~		写结果		A		Load部件 名称		ısy	地址	ſ	直			
L. D 1	F2, O(R3)					写结果		•		名称 Loadi	Bu Ye	5	地址 R[R2]+21		直			
L.D D	F2, O(R3) F0, F2, F4	1	1			写结果		A		名称 Loadi Load2	Bu Ye:	5			直	A		
L.D I L.D I MULT.D I SUB.D I	F2, O(R3) F0, F2, F4 F8, F6, F2	1	1			写结果		•		名称 Loadi	Bu Ye	5	R[R2]+21		直	^		
L.D I L.D I MULT.D I SUB.D I DIV.D I	F2, O(R3) F0, F2, F4	1 2 76	1			写结果	,	•		名称 Loadi Load2	Bu Ye:	5	R[R2]+21		Ē			
L.D I L.D I MULT.D I SUB.D I DIV.D I	F2, O(R3) F0, F2, F4 F8, F6, F2 F10, F0, F	1 2 76	1			写结果		•		名称 Loadi Load2	Bu Ye:	5	R[R2]+21		直			
L. D D D D D D D D D D D D D D D D D D D	F2, O(R3) F0, F2, F4 F8, F6, F2 F10, F0, F F6, F8, F2	1 2 76	1			写结果		•		名称 Loadi Load2	Bu Ye:	5	R[R2]+21		直	•		
L.D 1 L.D 1 MULT.D 1 SUB.D 1 DIV.D 1 ADD.D 1	F2, O(R3) F0, F2, F4 F8, F6, F2 F10, F0, F F6, F8, F2	1 2 76 2	1 2	2~		写结果				名称 Load1 Load2 Load3	Bu Ye:	当前	R[R2]+21			^		
L. D D D D D D D D D D D D D D D D D D D	F2, O(R3) F0, F2, F4 F8, F6, F2 F10, F0, F F6, F8, F2	1 2 2 6 6 2 2 Busy	1		Vk	写结果	Qj	Qk		名称 Loadi Load2	Bu Ye:	当前	R[R2]+21)		go go	^		
L.D 1 L.D 1 MULT.D 1 SUB.D 1 DIV.D 1 ADD.D 1	F2, O(R3) F0, F2, F4 F8, F6, F2 F10, F0, F F6, F8, F2	Busy	1 2	2~		写结果				名称 Load1 Load2 Load3	Bu Ye:	当前	R[R2]+21			^		
L.D 1 L.D 1 MULT.D 1 SUB.D 1 DIV.D 1 ADD.D 1	F2, O(R3) F0, F2, F4 F8, F6, F2 F10, F0, F F6, F8, F2	1 2 2 6 6 2 2 Busy	1 2	2~		写结果				名称 Load1 Load2 Load3	Bu Ye:	当前	R[R2]+21			•		
L.D 1 L.D 1 MULT.D 1 SUB.D 1 DIV.D 1 ADD.D 1	F2, O(R3) F0, F2, F4 F8, F6, F2 F10, F0, F F6, F8, F2	Busy No	1 2	2~		写结果				名称 Load1 Load2 Load3	Bu Ye:	当前	R[R2]+21			•		
L.D 1 L.D 1 MULT.D 1 SUB.D 1 DIV.D 1 ADD.D 1	F2, O(R3) F0, F2, F4 F8, F6, F2 F10, F0, F F6, F8, F2 名称 Add1 Add2 Add3	Busy No No	1 2	2~		写结果				名称 Load1 Load2 Load3	Bu Ye:	当前	R[R2]+21			^		
L.D 1 L.D 1 MULT.D 1 SUB.D 1 DIV.D 1 ADD.D 1	F2, O(R3) F0, F2, F4 F8, F6, F2, F10, F0, F, F6, F8, F2 Add1 Add2 Add2 Mult1	Buzy No No No	1 2	2~		写结果				名称 Load1 Load2 Load3	Bu Ye:	当前	R[R2]+21			•		
L. D 1 L. D 1 MULT. D 1 SUB. D 1 ADD. D 1 REST	F2, O(R3) F0, F2, F4 F8, F6, F2, F10, F0, F, F6, F8, F2 Add1 Add2 Add2 Mult1	Buzy No No No	1 2	2~		写结果			F18	名称 Load1 Load2 Load3	Bu Ye:	当前	R[R2]+21					
L. D 1 L. D 1 MULT. D 1 SUB. D 1 DIV. D 1 ADD. D 1 R留站	F2, 0(R3) F0, F2, F4 F8, F6, F2 F8, F6, F2 F10, F0, F8, F2 Add1 Add1 Add2 Add3 Mult1 Mult1	Busy No No No	0p	Vj	Vk		Qj	Qk		名称 Load1 Load2 Load3	Bu Ye Ye No	当前	R[R2]+21 ウ周期 多至 [: 2	go	^		

第二条L.D指令占用了Load2部件,将Busy置为Yes状态

第一条L.D指令更新访存地址为R[R2] + 21, 并储存在Load1部件中

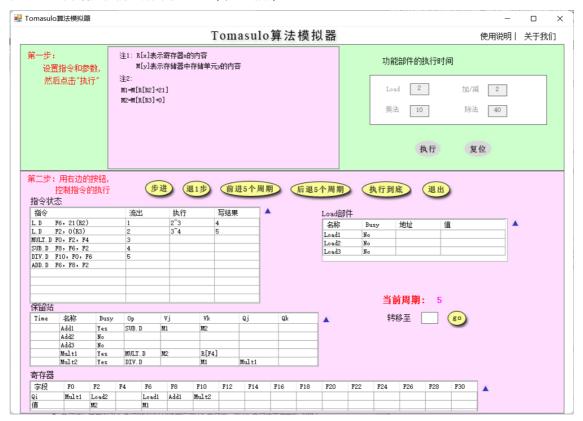
• 周期3:



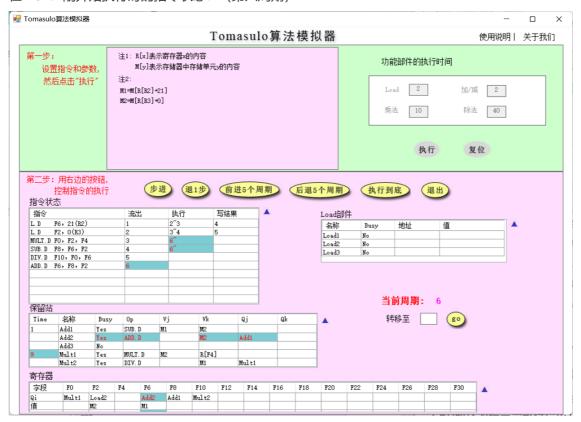
第一条L.D指令将Load1部件中的地址在内存中的值存入Load1的值寄存器中 第二条L.D指令更新访存地址为R[R3]+0,并存入Load2部件的地址寄存器中

2.请截图 (MUL.D刚开始执行时系统状态) , 并说明该周期相比上一周期整个系统发生了哪些改动 (指令状态、保留站、寄存器和Load部件)

• 在MUL.D开始执行前的指令状态: (第五周期)



• 在MUL.D刚开始执行时的指令状态: (第六周期)



参数改动如下:

- 1. 指令状态: 第六条指令进入流出状态, 第3、4条指令进入执行状态。
- 2. 保留站: Add2保留站中被第六条指令所占有, 且第三条指令与第四条指令开始执行Time计时。
- 3. **寄存器**: Add2保留站所占有了原来对于Load1所占有的F6寄存器,即第六条指令将写回F6寄存器。

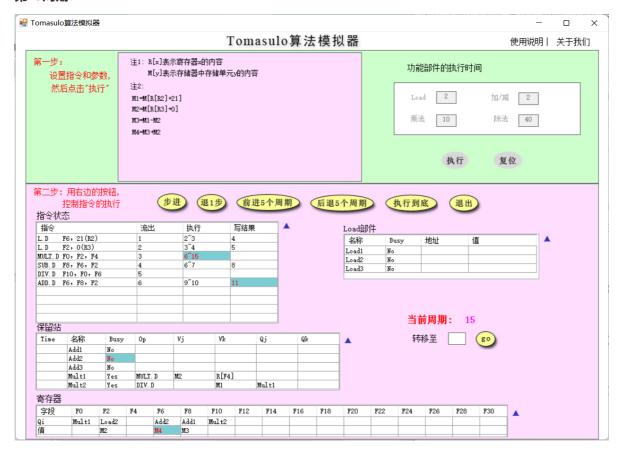
4. Load部件:没有任何变化

3.简要说明是什么相关导致MUL.D流出后没有立即执行

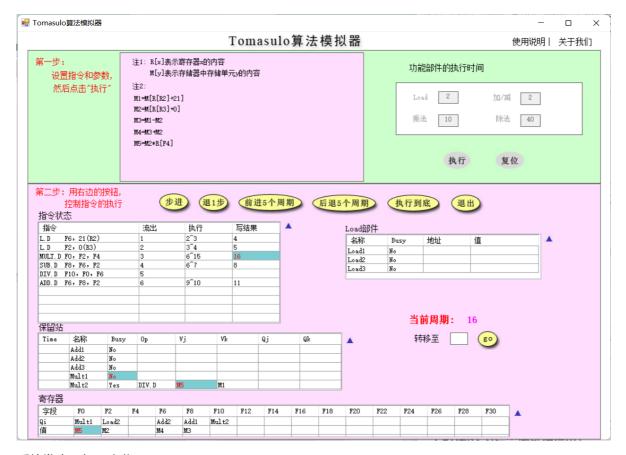
数据相关,第二条指令还未写入F2寄存器,此时会产生RAW相关。

4.请分别截图(15周期和16周期的系统状态),并分析系统发生了哪些变化

第15周期:



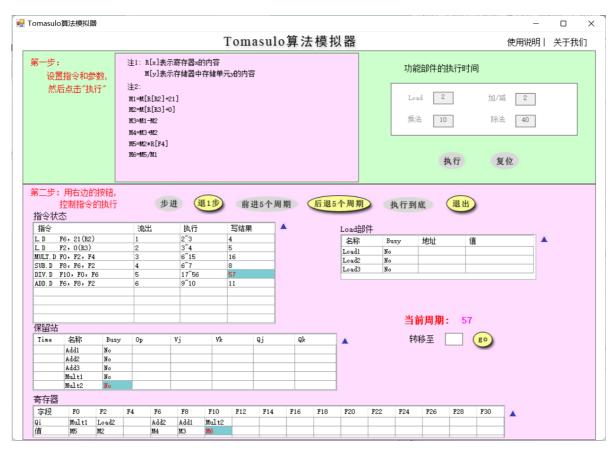
第16周期:



系统发生了如下变化:

1. 第三条指令MULT.D开始执行写结果阶段,释放了保留站中Mult1的资源,Busy改为No,将寄存器中F0的值更新为M5,并将Mult2保留站中的Vj更新为M5操作数,此时DIV.D指令的操作数准备就绪。

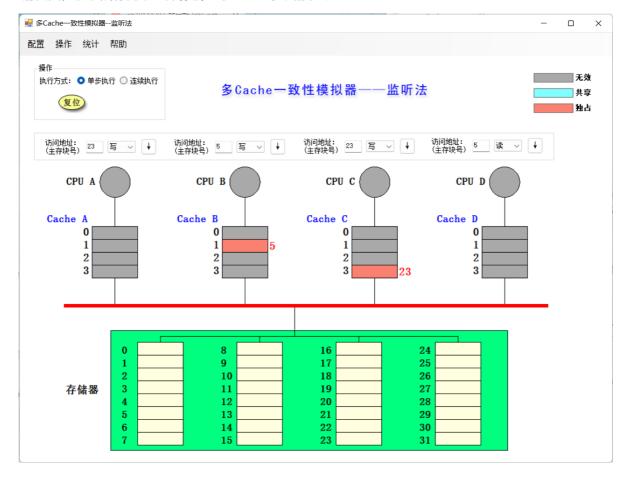
5.回答所有指令刚刚执行完毕时是第多少周期,同时请截图(最后一条指令写CBD时认为指令流执行结束)



二、多cache一致性算法-监听法

所进 行的 访问	是否发生 了替换?	是否 发生 了写 回?	监听协议进行的操作与块状态改变
CPU A读 第5 块	是,替换 Cache A 的块 1	否	Cache A读Miss,内存将块5传输至Cache A,并将Cache A块1 修改为共享
CPU B读 第5 块	是,替换 Cache B 的块 1	否	Cache B读Miss,内存将块5传输至Cache B,并将Cache B块1 修改为共享
CPU C读 第5 块	是,替换 Cache C 的块 1	否	Cache C读Miss,内存将块5传输至Cache C,并将Cache C块1 修改为共享
CPU B写 第5 块	未发生替换	否	Cache B修改块1的值并将Cache B将块1修改为独占,将Cache A和Cache C变为无效
CPU D读 第5 块	是,替换 Cache D 的块1	是	Cache D读Miss,Cache B写回第五块,内存将块5传输至 Cache D,并将Cache B 和 Cache D块1修改为共享
CPU B写 第21 块	是,替换 Cache B 的块1	否	Cache B 写失效,将内存中第21块传送到Cache B的块1上,将块1修改为独占
CPU A写 第23 块	是,替换 Cache A 的块3	否	Cache A 写失效,将内存中第23块传送到Cache A的块3上,将块3修改为独占
CPU C写 第23 块	是,替换 Cache C 的块3	是	Cache C 写失效,Cache A写回第23块,Cache C将内存中第23 块传送到Cache C的块3上,将块3修改为独占,并使Cache A块 3修改为失效
CPU B读 第29 块	是,替换 Cache B 的块1	是	Cache B 写回第21块且读失效,内存将第29块传输至CacheB的块1上,并设置块1为共享
CPU B写 第5 块	是,替换 Cache B 的块 1	否	Cache B写失效,内存将第5块传输至Cache B ,并将Cache B的块1设置为独占,Cache D 的块1变为无效

请截图,展示执行完以上操作后整个cache系统的状态。



三、多 cache 一致性算法-目录法

所进 行的 访问	监听协议进行的操作与块状态改变
CPU A 读 第6块	Cache A读失效,并将读未命中(A,6)发送给存储器,将块6送至Cache A块2上,并在存储器块6中记录A, Cache A块 2 设为共享, 存储器中块6共享目录上为{A}
CPU B读 第6块	Cache B读失效,并将读未命中(B,6)发送给存储器,存储器将第六块传输至CacheB块2上,并将CacheB块2状态设为共享,存储器中块6共享目录上为{A,B}
CPU D读 第6块	Cache D读失效,并将读未命中(D,6)发送给存储器,存储器将第六块传输至CacheD块2上,并将Cache D块2状态设为共享,存储器中块6共享目录上为{A,B,D}
CPU B写 第6块	Cache B写命中,并将命中的信息传输到存储器,此时存储器将作废发送至Cache A与Cache D中,此时Cache A与Cache D的块2变为无效, Cache B中的块变为独占,存储器上的块6共享目录为{B}
CPU C 读 第6块	Cache C 读失效,并将消息发送到存储器,存储器将Cache B写的写入存储器块6中,并将存储器块6的值写入Cache C,将Cache B 与 Cache C都设为共享, 块6共享目录为{B,C}
CPU D写 第20 块	Cache D 写失效,存储器将块20存入CacheD中, CacheD中的块0设为独占,块20的共享目录为{D}
CPUA 写第 20块	Cache A写失效,存储器将 CacheD中的值写回存储器中,并将CacheD块0设为无效,将存储器块20传入Cache A,Cache A的块0状态改为独占,块20的共享目录为{A}
CPU D写 第6块	Cache D写失效(D,6),存储器向Cache B、Cache C发送消息,将其块2置为无效,将存储器中的第六块传输至CacheD,Cache D的块2状态修改为独占,块6存储器的共享目录为{D}
CPU A读 第12 块	Cache A写回至存储器块20,将Cache A的块0置为无效,并发送读失效(A,12)至存储器中,存储器将第12块传送到Cache A,并将Cache A 块0设置为共享,并将块12 的共享目录修改为{A}

请截图,展示执行完以上操作后整个cache系统的状态。



四、综合问答

1. 目录法和监听法分别是集中式和基于总线,两者优劣是什么? (言之有理即可)

监听法优点:实现较为简单,在小型体系中效率较高。**缺点**:对总线带宽等要求较高,扩展性较差,总线压力大,在大型机中效果较差。

目录法优点:易于支持大型机,通过互联网络降低总线的压力。**缺点**:需要专门的目录空间,空间 开销大,且实现复杂。

2. Tomasulo算法相比ScoreBoard算法有什么异同? (简要回答两点: 1.分别解决了什么相关, 2.分别是分布式还是集中式) (参考第五版教材)

解决相关:

同: Tomasulo算法和ScoreBoard算法均解决了结构相关和数据相关,且都是通过动态调度的方式解决RAW相关。

异: Tomasulo算法可以直接将结果写入保留站中以供使用, ScoreBoard算法只能将结果先写入寄存器中, 再将寄存器中的数据读入使用。

分布式还是集中式:

Tomasulo算法是分布式算法,而ScoreBoard算法是集中式算法。

3. Tomasulo算法是如何解决结构、RAW、WAR和WAW相关的? (参考第五版教材)

结构相关: 当资源非Busy时才可以发射指令。

RAW相关:在保留站中记录每个功能部件的操作数,当操作数均就绪时,才会读取执行,否则会Stall住,防止产生RAW相关。

WAR, WAW相关:寄存器重命名解决WAR, WAW相关。