

计算机体系结构 lab 6

PB20111704 张宇昂

Tomasulo 模拟器

Q1

周期二：

Tomasulo算法模拟器

使用说明 | 关于我们

第一步：
设置指令和参数，
然后点击“执行”

注1: R[x]表示寄存器x的内容
M[y]表示存储器中存储单元y的内容
注2:

功能部件的执行时间

Load: 2 加/减: 2
乘法: 10 除法: 40

执行 复位

**第二步：用右边的按钮，
控制指令的执行**

步进 退1步 前进5个周期 后退5个周期 执行到底 退出

指令状态

指令	流出	执行	写结果
L.D R6, 0(R2)	1	2	
L.D R2, 0(R3)	2		
MULT.D R0, R2, R4			
ST.D R0, R6, R2			
DIV.D R10, R0, R0			
ADD.D R6, R0, R2			

Load部件

名称	Busy	地址	值
Load1	Yes	R[R2]<1	
Load2	Yes	0	
Load3	No		

当前周期： 2

转移至

保留站

Time	名称	Busy	Op	Vj	Vr	Qj	Qr
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

寄存器

字段	R0	R2	R4	R6	R8	R10	R12	R14	R16	R18	R20	R22	R24	R26	R28	R30
Q1		Load2		Load1												
值																

周期三：

Tomasulo算法模拟器

使用说明 | 关于我们

第一步：
设置指令和参数，
然后点击“执行”

注1: R[x]表示寄存器x的内容
M[y]表示存储器中存储单元y的内容
注2:

功能部件的执行时间

Load: 2 加/减: 2
乘法: 10 除法: 40

执行 复位

**第二步：用右边的按钮，
控制指令的执行**

步进 退1步 前进5个周期 后退5个周期 执行到底 退出

指令状态

指令	流出	执行	写结果
L.D R6, 0(R2)	1	2	
L.D R2, 0(R3)	2	3	
MULT.D R0, R2, R4	3		
ST.D R0, R6, R2			
DIV.D R10, R0, R0			
ADD.D R6, R0, R2			

Load部件

名称	Busy	地址	值
Load1	Yes	R[R2]<1	M[R[R2]<1]
Load2	Yes	R[R2]<0	
Load3	No		

当前周期： 3

转移至

保留站

Time	名称	Busy	Op	Vj	Vr	Qj	Qr
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D		R[R2]		Load2
	Mult2	No					

寄存器

字段	R0	R2	R4	R6	R8	R10	R12	R14	R16	R18	R20	R22	R24	R26	R28	R30
Q1		Mult1	Load2	Load1												
值																

load部件的改动：

- Load 1从内存中取到了值，下个周期可以进行写回阶段
- Load 2进入执行阶段，地址变成了有效地址

Q2

MUL.D开始执行前的系统状态：

第一步：
设置指令和参数，
然后点击“执行”

注1: R[x]表示寄存器x的内容

注2: M[y]表示存储器中存储单元y的内容

M1←M[R[R2]×21]

M2←M[R[R3]×0]

功能部件的执行时间

Load2加/减2

乘法10除法40

执行 复位

第二步：用右边的按钮，
控制指令的执行

步进退1步前进5个周期后退5个周期执行到底退出

指令状态

指令	流出	执行	写结果
L.D F6, Z1(R2)	1	2'3	4
L.D F2, O(R3)	2	3'4	5
MULT.D F0, F2, F4	3	4'	
SUB.D F0, F0, F2	4	5'	
DIV.D F10, F0, F0	5		
ADD.D F6, F0, F2	6		

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
1	Add1	Yes	SUB.D	M1	M2		
	Add2	Yes	ADD.D		M2	Add1	
	Add3	No					
9	Mult1	Yes	MULT.D	M2	R[F4]		
	Mult2	Yes	DIV.D		M1	Mult1	

寄存器

字模	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi	Mult1	Load2		Add2	Add1	Mult2										
值		M2		M1												

Load部件

名称	Busy	地址	值
Load1	No		
Load2	No		
Load3	No		

当前周期：6

转移至 5 GO

在此之前一个周期的系统状态：

第一步：
设置指令和参数，
然后点击“执行”

注1: R[x]表示寄存器x的内容

注2: M[y]表示存储器中存储单元y的内容

M1←M[R[R2]×21]

M2←M[R[R3]×0]

功能部件的执行时间

Load2加/减2

乘法10除法40

执行 复位

第二步：用右边的按钮，
控制指令的执行

步进退1步前进5个周期后退5个周期执行到底退出

指令状态

指令	流出	执行	写结果
L.D F6, Z1(R2)	1	2'3	4
L.D F2, O(R3)	2	3'4	5
MULT.D F0, F2, F4	3		
SUB.D F0, F0, F2	4		
DIV.D F10, F0, F0	5		
ADD.D F6, F0, F2			

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	Yes	SUB.D	M1	M2		
	Add2	No					
	Add3	No					
	Mult1	Yes	MULT.D	M2	R[F4]		
	Mult2	Yes	DIV.D		M1	Mult1	

寄存器

字模	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi	Mult1	Load2			Add1	Mult2										
值		M2			M1											

Load部件

名称	Busy	地址	值
Load1	No		
Load2	No		
Load3	No		

当前周期：5

转移至 5 GO

相比周期5的变化：

- 指令状态：
 - 第3, 4条指令开始执行，第六条指令发射
- 保留站：
 - Add2的状态修改为yes，写入第六条指令的信息
 - Mult1得到了操作数并开始执行，Time设为9
- 寄存器：
 - 由于第6条指令的流入，F6的Qi被设为保留站编号Add2，而值保持不变

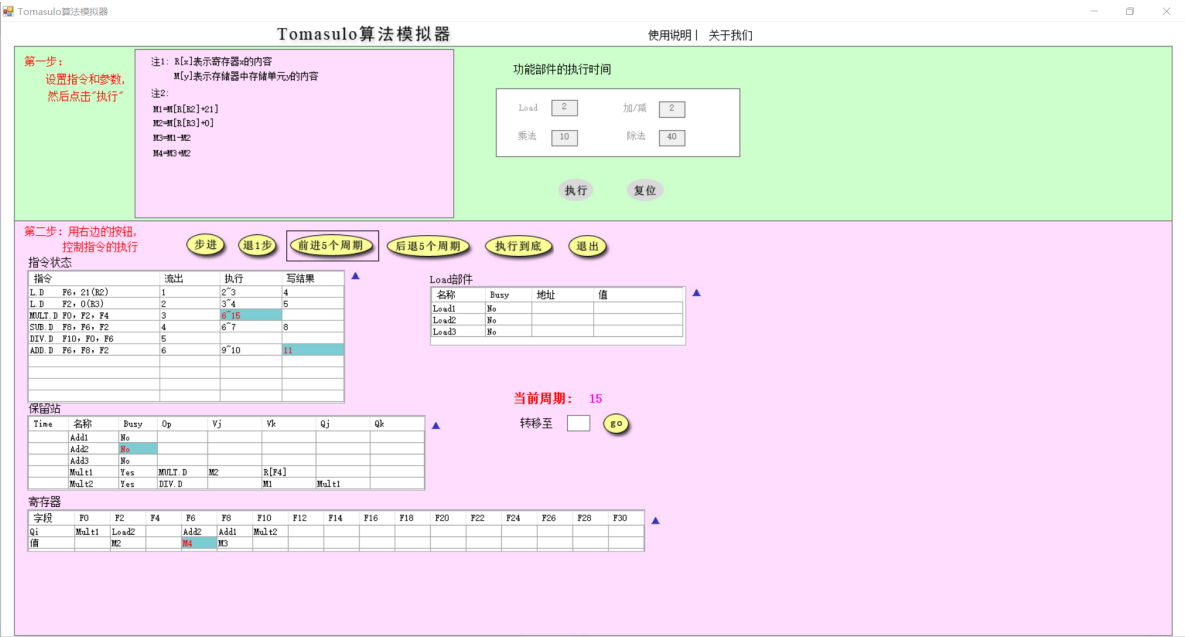
No. 2 / 9

Q3

RAW相关导致 MULT.D 的操作数 F2 需要等待上一条指令写入结果

Q4

第15周期:



第16周期:



系统的变化:

- 指令状态: MULT.D 结束执行阶段, 进入写回阶段
- 保留站:
 - Mult1解除占用状态, Busy状态改为No
 - Mult2的操作数F0变为可用, Qj=0, Vj=M5
- 寄存器: F0变为Not Busy, 值写入M5

Q5

第一步：
设置指令和参数，
然后点击“执行”

注1：R[x]表示寄存器x的内容
注2：
M[x]表示存储器中存储单元x的内容
M1=M[R[R2]]*21
M2=M[R[R3]]*0
M3=M1-M2
M4=M3-M2
M5=M2+3[F4]
M6=M5/M1

功能部件的执行时间
Load 2 加/减 2
乘法 10 除法 40
执行 复位

第二步：用右边的按钮，
控制指令的执行

步进 退1步 前进5个周期 后退5个周期 执行到底 退出

指令状态

指令	流出	执行	写结果
L D R6, 21(R2)	1	2 3	4
L D R2, 0(R3)	2	3 4	5
MULT D R0, R2, F4	3	6 15	16
STG D R0, R6, F2	4	6 7	8
DIV D F10, R0, R6	5	17 56	57
ADD D R6, R0, F2	6	9 10	11

Load部件

名称	Busy	地址	值
Load1	No		
Load2	No		
Load3	No		

当前周期： 57

转移至 go

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Multi1	No					
	Multi2	No					

寄存器

字块	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qj	M5	M2		M4	M3	M6										
值																

第57个周期所有指令执行完毕

多Cache一致性算法-监听法

Q1

我个人认为Cache在开始时相当于内存块替换了空块，所以应该算作发生了替换

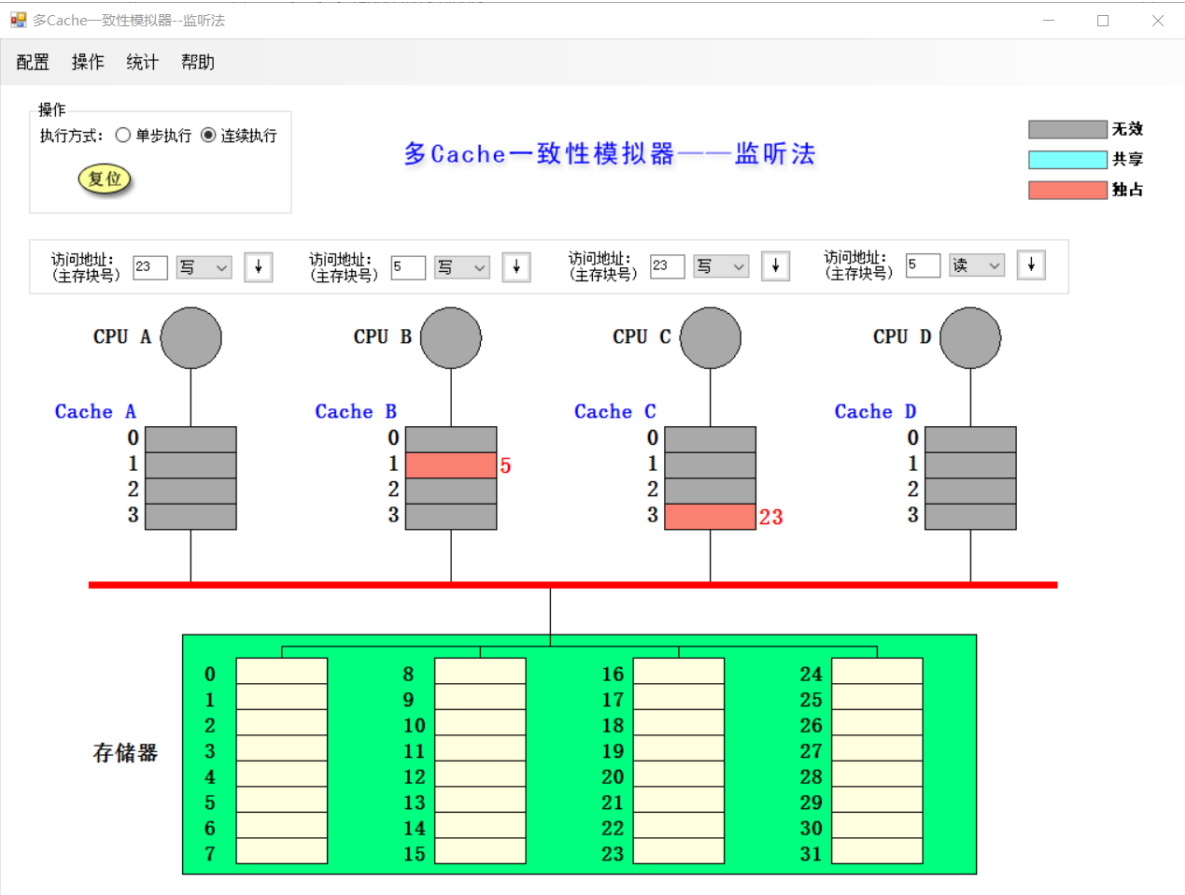
No. 4 / 9

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	Yes	No	CPU A 读取未命中，向内存中读取第 5 块，第五块放入 CPU A 的 Cache 中，状态设置为共享状态
CPU B 读第 5 块	Yes	No	CPU B 读取未命中，向内存中读取第 5 块，第五块放入 CPU B 的 Cache 中，状态设置为共享状态
CPU C 读第 5 块	Yes	No	CPU C 读取未命中，向内存中读取第 5 块，第五块放入 CPU C 的 Cache 中，状态设置为共享状态
CPU B 写第 5 块	No	No	CPU B 写入 Cache 中的第 5 块，随后向总线中发送作废信号，使得 CPU A 和 CPU C 的 Cache 中的第五块变为无效状态，CPU B 的第五块变为独占状态
CPU D 读第 5 块	Yes	Yes	CPU D 读取未命中，CPU B 的 Cache 将第 5 块写回内存，状态变为共享，D 读取内存中的第五块并保存到 Cache 中，且 Cache 中该块状态变为共享
CPU B 写第 21 块	Yes	No	CPU B 写不命中，读取内存中第 21 块替换 Cache 中的第五块，状态变为独占
CPU A 写第 23 块	Yes	No	CPU A 写不命中，A 读取内存中第 23 块替换到 Cache 中，状态变为独占
CPU C 写第 23 块	Yes	Yes	CPU C 写不命中，A 将第 23 块写回内存，状态变为无效，C 读取第 23 块到 Cache 中并写入，状态变为独占
CPU B 读第 29 块	Yes	Yes	CPU B 写回第 21 块，随后读第 29 块不命中，向内存中读取第 29 块，29 块写回 B 的 Cache，状态变为共享

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 B 写第5块	Yes	No	CPU B读取第5块，状态变为独占，替换第29块，CPU D的第五块状态变为无效

Q2

最后整个Cache系统的状态如下：



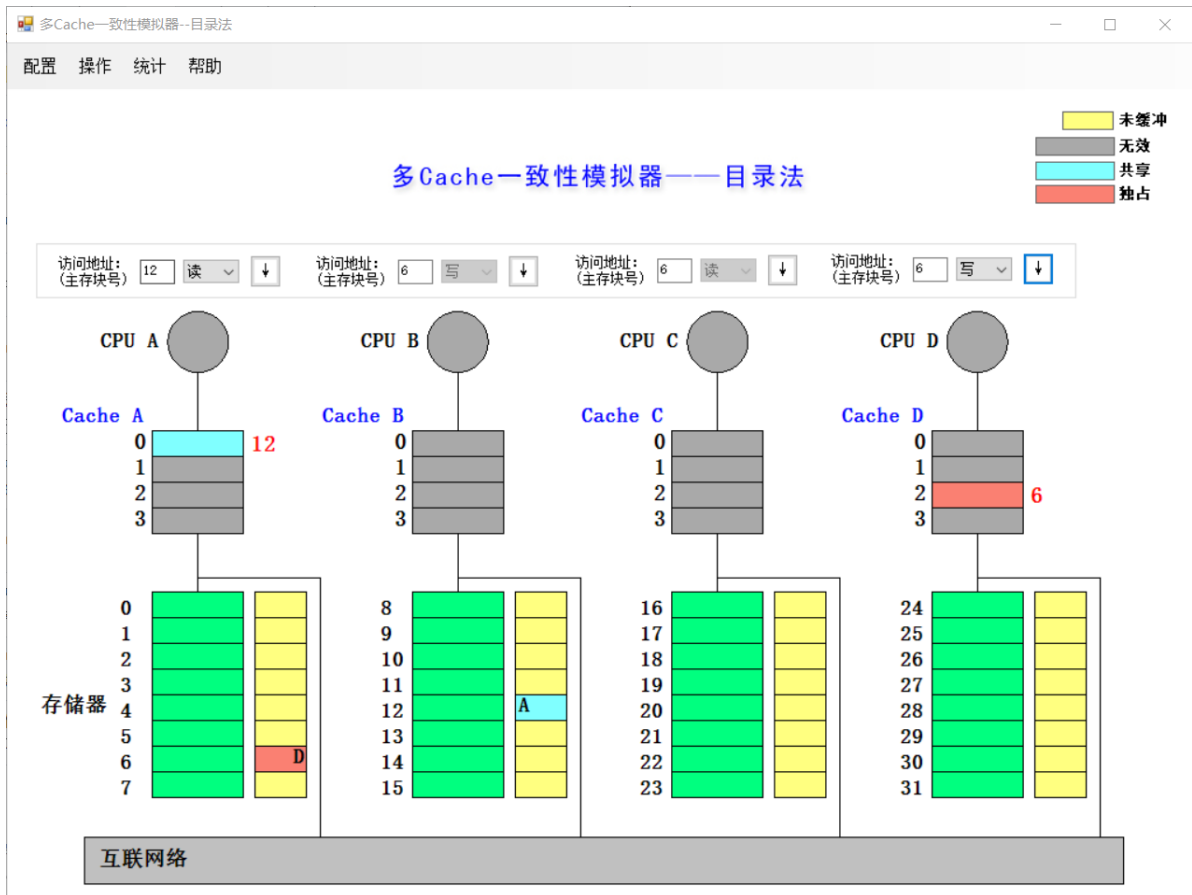
多Cache一致性算法-目录法

Q1

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	CPU A读不命中，读取第6块，存储器状态改为共享，目录记A为共享者，将第6块写入Cache A
CPU B 读第6块	CPU B读不命中，读取第6块，目录记AB为共享者，将第6块写入Cache B
CPU D 读第6块	CPU D读不命中，读取第6块，目录记ABD为共享者，将第6块写入Cache D
CPU B 写第6块	CPU B写入第六块，写命中后将存储器目录改为B独占，将A，D的Cache中的第六块变为无效状态
CPU C 读第6块	CPU C读不命中，存储器读取B的第6块，B将第六块写回，状态改为共享，存储器将第六块发回到C的Cache，目录中记BC为共享者
CPU D 写第20块	CPU D写不命中，存储器向D的Cache发送第20块，将存储器目录改为D独占
CPU A 写第20块	CPU A写不命中，存储器读取D中的第20块并存入存储器，将D中的第20块变为无效，目录中记A为独占者，向A写入第20块
CPU D 写第6块	CPU D写不命中，存储器将B，C的Cache中第6块作废，存储器向D发送第6块，目录记D为独占，D写入第六块
CPU A 读第12块	A将第20块写回，向存储器发送第20块，第20块所在存储器将第20块目录置空，A读不命中，存储器将第12块发送到A的Cache中，目录记A为共享者

Q2

整个Cache系统状态如下图所示：



综合问答

Q1

目录法：

- 优点：可扩展性高，可以用于大型多处理器系统；也能够通过维护更多的状态信息来提高一致性协议的效率
- 缺点：需要一个额外的中心控制器，这可能会成为系统的瓶颈

监听法：

- 优点：不需要一个额外的中心控制器，并且可以在总线上处理一致性协议
- 缺点：能会导致总线拥塞，尤其是在大型多处理器系统中，因为每个处理器都会在总线上发送大量的请求和响应消息

Q2

两种算法都解决了结构相关、RAW、WAR、WAW相关，都是通过动态调度的方法解决相关问题的；但是Tomasulo通过寄存器重命名来解决WAR和WAW相关；而ScoreBoard是通过插入stall暂停的方式解决这两种相关的

Tomasulo是分布式；ScoreBoard是集中式

Q3

Tomasulo解决这几种相关的方式：

- 结构相关：部件流水化，所有功能部件有序访问存储器
- RAW相关：通过CDB检测源操作数是否可用，若所有源操作数均可用则让指令进入执行阶段
- WAR/WAW相关：采用寄存器重命名的方法，在发射阶段保留站空闲时才发射指令和操作数