

Lab5实验报告

PB18111757 陈金宝

一、 Tomasulo算法模拟器

初始时设置指令和延迟，如下：

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”

指令

L.D	F6	21	R2
L.D	F2	0	R3
MULT.D	F0	F2	F4
SUB.D	F8	F6	F2
DIV.D	F10	F0	F6
ADD.D	F6	F8	F2
NOP	Null	Null	Null
NOP	Null	Null	Null
NOP	Null	Null	Null
NOP	Null	Null	Null

功能部件的执行时间

Load	2	加/减	2
乘法	10	除法	40

执行 复位

1.

周期为2时:

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 F6, 21(R2)	1	2	
L.D F2, 0(R3)	2		
MULT.D F0, F2, F4			
SUB.D F8, F6, F2			
DIV.D F10, F0, F6			
ADD.D F6, F8, F2			

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Mult1	No					
	Mult2	No					

寄存器

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi		Load2		Load1												
值																

Load部件

名称	Busy	地址	值
Load1	Yes	R[R2]*21	
Load2	Yes	0	
Load3	No		

当前周期: 2

转移至 go

此时两条load指令均已经发射。两条load指令分别使用了Load1和Load2部件。其中第一条load指令已经进入执行阶段，第二条load指令刚刚发射，还未进入执行阶段，未得到load所需的地址。

周期为3时:

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 F6, 21(R2)	1	2~3	
L.D F2, 0(R3)	2	3~	
MULT.D F0, F2, F4	3		
SUB.D F8, F6, F2			
DIV.D F10, F0, F6			
ADD.D F6, F8, F2			

Load部件

名称	Busy	地址	值
Load1	Yes	R[R2]+21	M[R[R2]+21]
Load2	Yes	R[R3]+0	
Load3	No		

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
	Add1	No					
	Add2	No					
	Add3	No					
	Multi1	Yes	MULT.D		R[F4]	Load2	
	Multi2	No					

寄存器

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi																
值																

当前周期: 3

转移至

GO

此时load2进入执行阶段，获取要load的memory地址，load1取得memory中的值。

2.

乘法开始时的模拟器状态:

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:

M1=M[R[R2]+21]

M2=M[R[R3]+0]

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 F6, 21(R2)	1	2~3	4
L.D F2, 0(R3)	2	3~4	5
MULT.D F0, F2, F4	3	6~	
SUB.D F8, F6, F2	4	6~	
DIV.D F10, F0, F6	5		
ADD.D F6, F8, F2	6		

Load部件

名称	Busy	地址	值
Load1	No		
Load2	No		
Load3	No		

保留站

Time	名称	Busy	Op	Vj	Vk	Qj	Qk
1	Add1	Yes	SUB.D	M1	M2	Add1	
	Add2	Yes	ADD.D		M2		
	Add3	No					
9	Multi1	Yes	MULT.D	M2	R[F4]		
	Multi2	Yes	DIV.D	M2	M1	Multi1	

寄存器

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi																
值																

当前周期: 6

转移至

GO

此时为第6个周期，MULT.D指令开始执行。相比第5个周期，发生的改动有:MULT.D、SUB.D指令进入执行阶段、ADD.D指令发射。寄存器状态中F6的Qi变为add2，保留站中的add2被ADD.D指令占用。

3.

因为此时存在RAW相关。MULT.D所需的F2（由load获取）还未就绪。此时MULT.D在等待LOAD将对应的F2写入。

4.

周期为15时：

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：
M1=M[R[R2]]+21
M2=M[R[R3]]+0
M3=M1-M2
M4=M3+M2

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, 21(R2)	1	2~3	4
L.D F2, 0(R3)	2	3~4	5
MULT.D F0, F2, F4	3	6~15	
SUB.D F8, F6, F2	4	6~7	8
DIV.D F10, F0, F6	5		
ADD.D F6, F8, F2	6	9~10	11

保留站

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

寄存器

字段	F0	F2	F4	F6	F8	F10	F12	F14	F16	F18	F20	F22	F24	F26	F28	F30
Qi	Mult1	Load2		Add2	Add1	Mult2										
值		M2		M4	M3											

Load部件

名称	Busy	地址	值
Load1	No		
Load2	No		
Load3	No		

当前周期：15

转移至

go

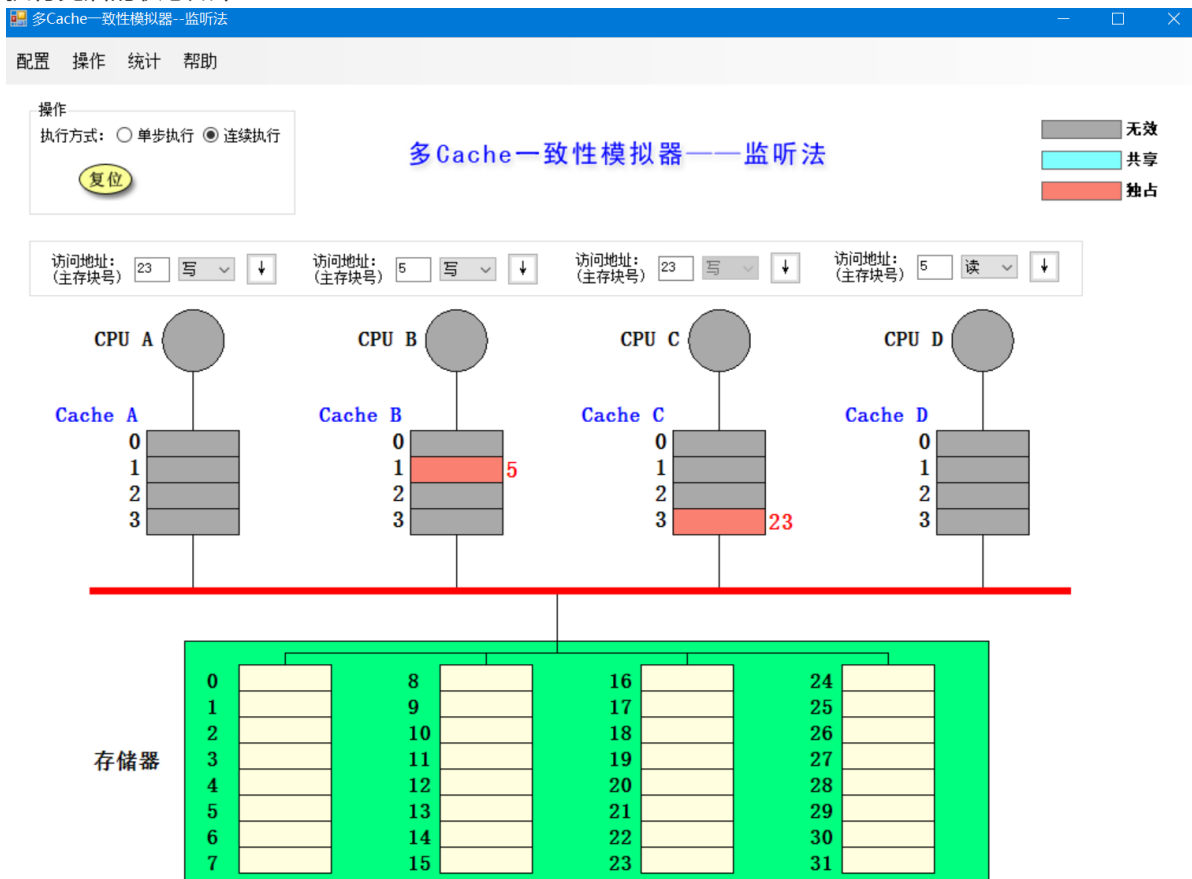
此时MULT.D执行完毕。

周期为16时：

1.

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 读第五块	否	否	将第5块从存储器读入到CacheA, 状态设置为共享
CPU B 读第五块	否	否	将第5块从存储器读入到CacheA, 状态设置为共享
CPU C 读第五块	否	否	将第5块从存储器读入到CacheA, 状态设置为共享
CPU B 写第五块	否	否	写命中, 根据监听协议此时A、C中的块5状态改为无效, B中的块5状态为独占
CPU D 读第五块	否	是	B中的块5写回到存储器, 状态设置为共享。且D读入块5, 状态设置为共享。
CPU B 写第21块	是	否	B中块5被块21替换, 状态设置为独占。
CPU A 写第23块	否	否	将第23块从存储器读入到CacheA, 状态设置为独占
CPU C 写第23块	否	是	写不命中, 根据监听协议此时A中的块23写回到存储器, 状态设置为无效, C读入块23并写, 状态设置为独占。
CPU B 读第29块	是	是	读不命中, B中块21被写回存储器。之后读入块29替换块21。状态设置为共享。
CPU B 写第5块	是	否	写不命中, 根据监听协议, B读入块5替换掉块29, 状态设置为独占。D中的块5状态设置为无效。

执行完后的状态如下:

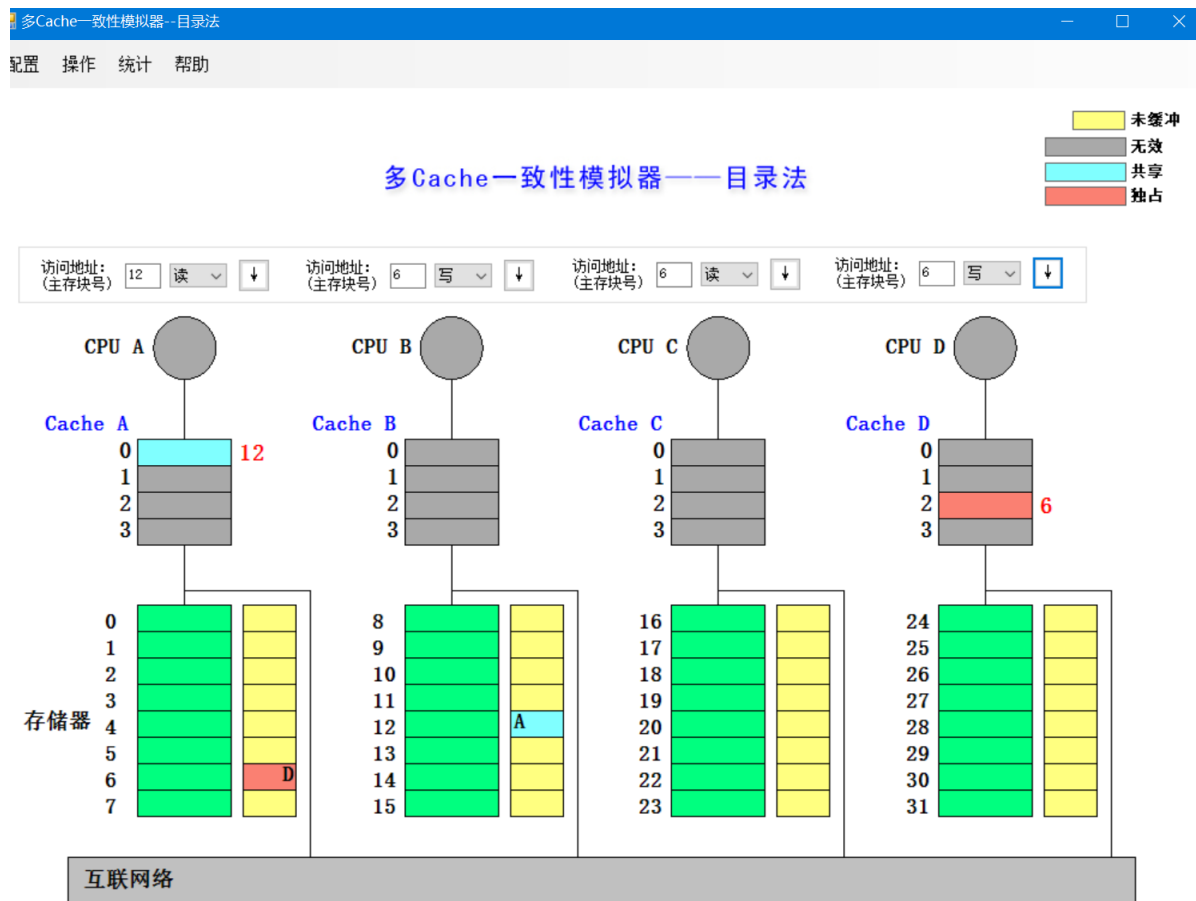


三、多cache一致性算法-目录法

1.

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	A向本地宿主发送读不命中(A,6)消息，宿主收到请求将块6传送给本地结点，状态设置为共享。共享集合为{A}。
CPU B读 第6 块	本地向宿主结点发送读不命中(B,6)消息，宿主将块传送给本地结点，共享集合为{A,B}，状态为共享
CPU D读 第6 块	本地向宿主结点发送读不命中(D,6)消息，宿主将块传送给本地结点，共享集合为{A,B,D}，状态为共享
CPU B写 第6 块	本地向宿主结点发送写命中(B,6)消息，宿主向远程结点A、D发送作废(6)消息。A、D将块状态设置为无效。B将块6状态设置为独占。目录将块6状态设置为独占，共享集合为{B}
CPU C读 第6 块	本地向宿主结点发送读不命中(C,6)消息，宿主向远程结点发送数据块6的消息(读请求)，远程结点将数据块6传送给宿主结点，宿主结点将数据块6传送给本地结点。此时状态均设置为共享，共享集合为{B,C}
CPU D写 第 20 块	本地向宿主结点发送写不命中(D,20)消息，宿主将数据库传送给本地。此时块20状态均为独占，共享集合为{D}
CPU A写 第 20 块	本地向宿主结点发送写不命中(A,20)消息，宿主给远程结点D发送取并作废(20)的消息，远程将数据块传送给宿主结点，将CacheD中的块20设为无效。宿主将块20传送给本地结点。状态均为独占。共享集合为{A}
CPU D写 第6 块	本地向宿主结点发送写不命中(D,6)消息，宿主向远程结点B、C发送作废(6)的消息。B、C将块6设置为无效。宿主将块6发送给本地结点。状态均为独占。共享集合为{B}
CPU A读 第 12 块	本地向被替换的宿主结点发写回并修改(A,20)的消息。A将块20写回存储器。之后本地向宿主结点发送读不命中(A,12)消息，宿主结点将块12传送给本地结点。状态均为共享，共享集合为{A}

执行完后的状态如下:



四、综合问答

1.

- 监听法:
优：监听协议较为简单，且具有较快的速度
劣：性能瓶颈取决于总线，总线带宽不足时会影响性能。
- 目录法
优：多个CPU的大型分布式cache可以稳定运行
劣：小型体系使用目录法会较为复杂

2.

- 异:
Tomasulo通过寄存器换名来解决WAW, WAR相关。执行前若操作数未准备好则等待。(解决RAW)。且处理和缓存是分布到各个保留站的, 是分布式的。而ScoreBoard通过检测是否具有相同的目的(源)寄存器, 若有则停止发射(或暂停执行)。且处理和缓存是集中式的。一般而言, Tomasulo效率更高, 停顿的周期更少, 可以发射更多的指令
- 同:
都能够通过动态指令流调度的方法来实现乱序执行, 一定程度上实现ILP。

3.

结构相关：监视保留站。当保留站中对应的功能部件有空闲(非Busy)时再发射对应的指令。

RAW：监视CDB。当操作数就绪（写回CDB）时再开始执行。

WAW,WAR：使用寄存器换名策略。