

《 计算机体系结构 》

实验报告三

|  |  |
| --- | --- |
| 班 级： | **计211** |
| 学 号： | **2103017** |
| 姓 名： | **蒋文强** |
| 指导教师： | **梁建宁** |

信息科学与工程学院

2023年 12月

**实验名称** 重排序缓冲ROB工作原理

**实验地点** 信息楼418 **实验日期** 2023.12.6

1. **实验目的**
2. 加深对指令级并行性及其开发的理解；
3. 加深对基于硬件的猜测的理解；
4. 掌握ROB在指令流出、执行、写回、提交4个阶段所进行的操作；
5. 掌握ROB结构特点；
6. 给定执行代码片段，能写出某个时钟周期，保留站、ROB、寄存器状态表的变化情况。
7. **实验设备**

个人PC机，实验平台采用重排序缓冲ROB模拟器。

1. **实验原理**

1、利用猜测挖掘更多ILP

（1）为了挖掘更多指令并行性，需要通过硬件猜测分支转移的方法克服控制相关，假设猜测是对的就继续执行指令。

（2）基于硬件的猜测的三个主要思想：

·通过动态分支预测来选择要执行的指令；

·猜测可以允许在控制相关解决前执行指令（能够撤销预测错误指令的影响）；

·动态调度基准块的不同组合。

（3）要获得更多的ILP，需要将猜测与动态调度结合起来。没有猜测的动态调度仅仅能够部分重叠执行基本块，因为它要求在执行后续基本块的任意一条指令之前，分支指令必须被判明（转移成功与否）。

（4）硬件猜测+Tomasulo算法的思想：

沿着预测的路径（分支转移方向）执行指令，但是只有预测正确时才确认提交指令结果。

可以将指令执行分成指令执行完成和指令提交两部分。

指令提交就是当指令不再不确定时（分支猜测正确），允许指令更新寄存器堆和存储器。

为了让猜测机制发挥效用，可以允许指令乱序执行，但是要强制循序提交。需要一些额外的硬件（ROB，重排序缓冲区）来阻止任何不可撤回的动作，直到指令确认提交。

2、重排序缓冲区

（1）ROB：当指令执行完毕尚未提交之前，执行结果存放于这里。

（2）ROB包含四项内容：

·指令种类：分支指令/Load/Store/ALU操作；

·目的寄存器、目的内存地址：寄存器编号、存储器编号；

·输出值：指令提交前的执行结果存放于这里；

·准备好：指令完成执行，数值已经准备好。

（3）引入重排序缓冲区后，保留栈也需要进行一些修改：操作数的来源改为ROB，不再是功能部件。ROB类似于Tomasulo原形算法中的RS，也起到扩展寄存器的作用。

（4）直到指令提交时，再将结果写入寄存器和存储器。

（5）对于预测错误：ROB里的相应内容被清除。

（6）对于异常：不予确认直到它对应的指令准备提交。如果一条猜测状态的指令造成一个异常，将异常记录到ROB；如果这条指令抵达ROB的头部，该指令不再是猜测状态，对应的异常准备执行。

（7）基本操作：

·按照FIFO先进先出规则，将指令放置在重排序缓冲区；

·当指令执行完毕，将结果存放在ROB中，并将结果提供其他需要的指令，跟保留栈起到类似作用。不过需要用ROB编号代替RS保留栈编号；

·指令提交：位于ROB头部的数据被写入寄存器或者存储器单元；

·利用ROB，在猜测错误时可以很容易撤销已执行指令。

3、含硬件猜测ROB算法步骤

（1）发射（从指令队列取指令）

如果保留栈和重新排序缓冲区有空闲，发射指令，送操作数以及暂存输出值的ROB编号。

（2）执行（执行指令）

如果操作数准备好就可以开始执行指令；如果操作数没有准备好，监听CDB公共数据通道等待操作数。这一步需要检查RAW冒险。

（3）写回

将结果通过CDB公共数据通道旁路传递给需要的功能部件和ROB。

（4）指令提交

用ROB里的输出结果更新寄存器和内存单元。

当指令位于ROB头部（预测正确，消除不确定性），更新寄存器和内存单元，并将指令从ROB移走。如果预测错误，ROB被刷新，执行过程从该分支的后续正常指令处重新开始。

1. **实验操作及运行结果**

1、假设浮点功能部件的延迟时间（处理时间）为：加法2个时钟周期，乘法10个时钟周期，除法40个时间周期，Load部件2个时钟周期。

①有以下代码段，给出当指令MUL.D即将确认时，保留站、ROB和浮点寄存器状态表的内容。

L.D F6, 24(R2)

L.D F2, 12(R3)

MUL.D F0, F2, F4

SUB.D F8, F6, F2

DIV.D F10, F0, F6

ADD.D F6, F8, F2

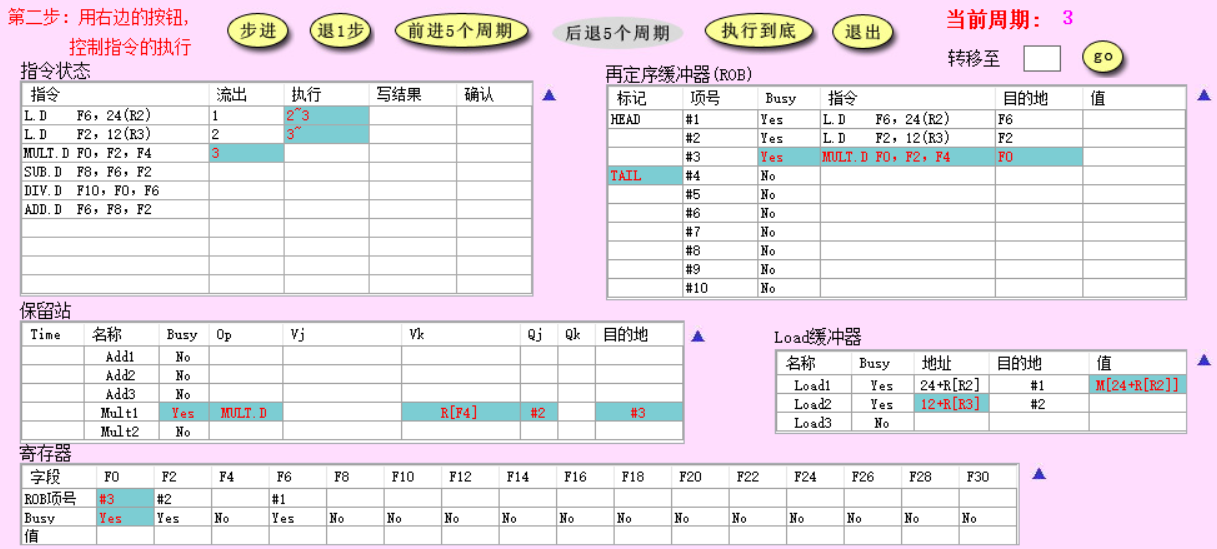
在第15个时钟周期时，指令MUL.D即将确认（该指令在第16个时钟周期时确认），其保留站、ROB和浮点寄存器状态表的内容如下图所示：



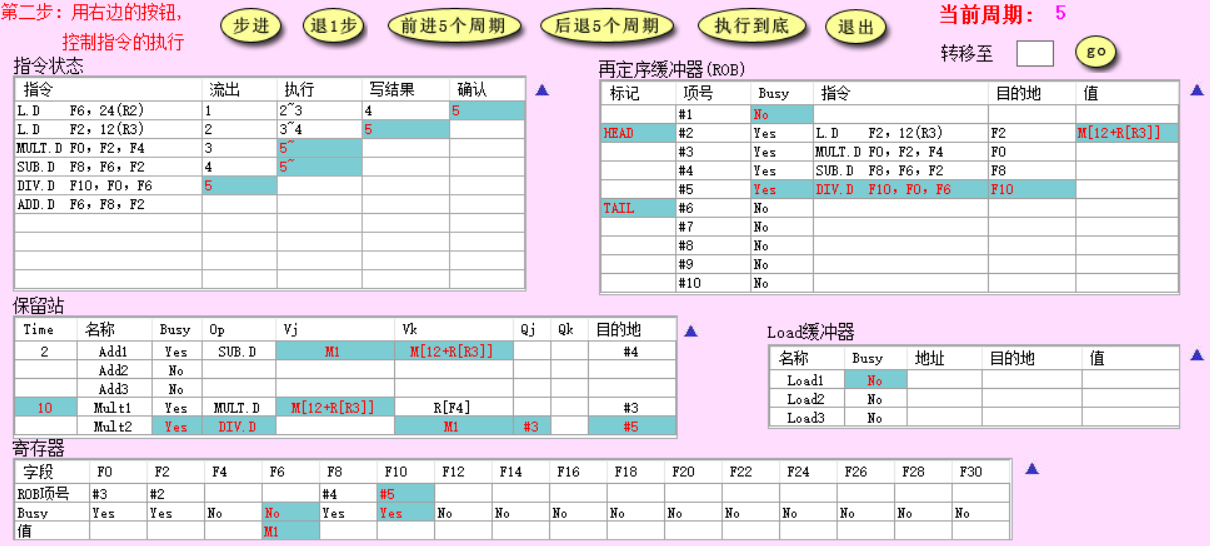
②按步进方式执行上述代码，利用模拟器的“小三角按钮”的对比显示功能，观察每一个时钟周期前后保留站、ROB和浮点寄存器状态的内容变化情况。**（针对步进过程中的一些比较重要的周期，在截图下方都附有文字解释）**



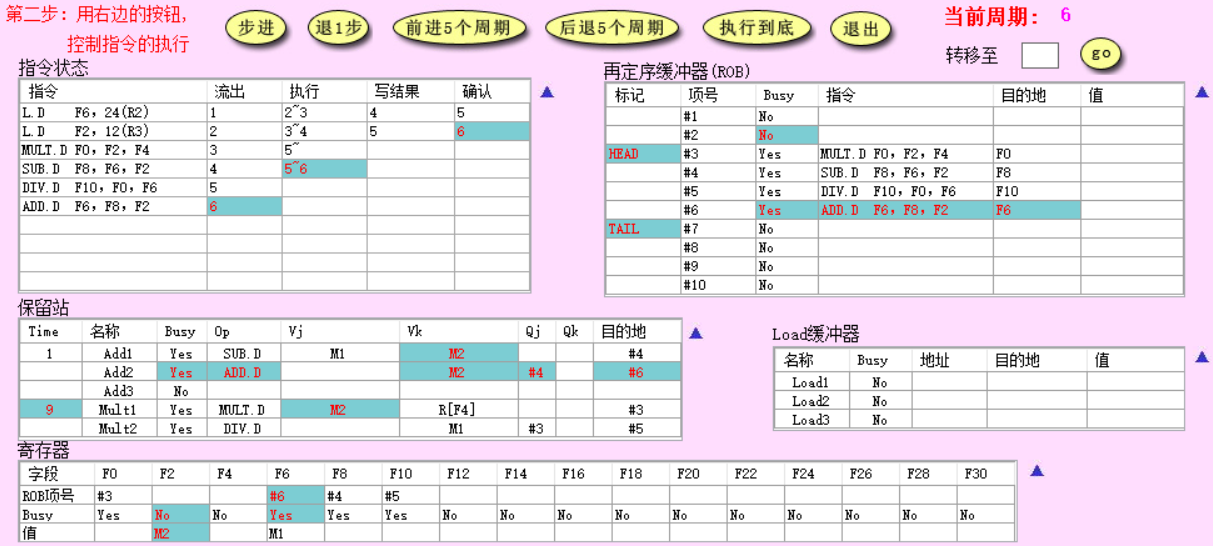




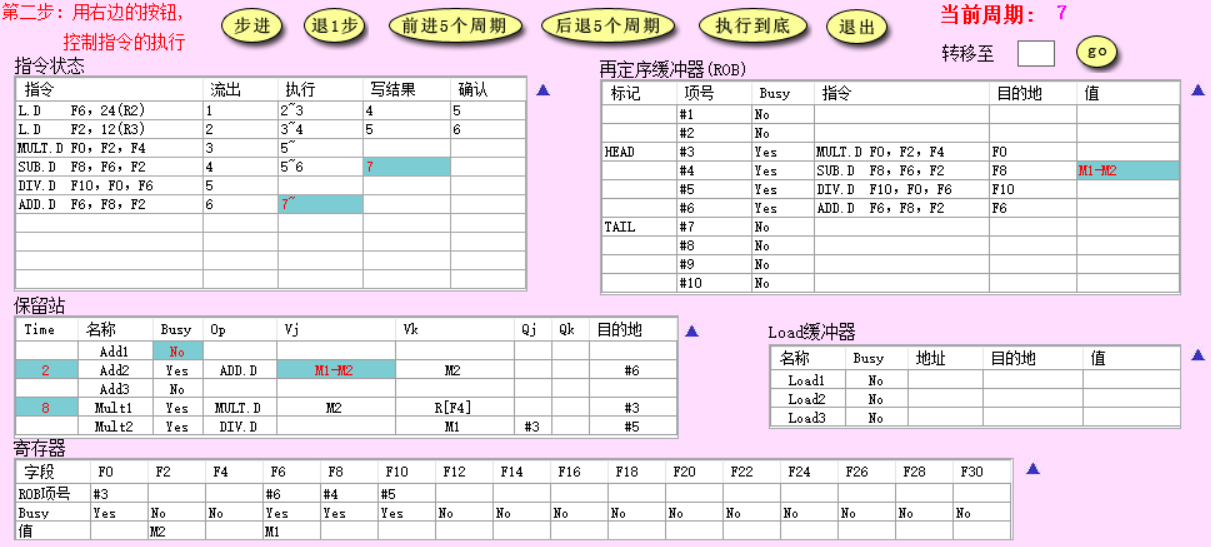




前5个周期中，第1条load指令完成了流出-执行-写结果-确认的过程，第2条load指令即将确认，MULT与SUB指令均在执行阶段。寄存器F6、F2、F0、F8、F10先后作为各指令在ROB中的目的地，项号为#1、#2、#3、#4、#5，这是ROB与动态调度中Tomasulo算法不同的地方。ROB中根据项号缓存了前5条指令的目的地与值，由于第一条load指令已经完确认，所以更新寄存器F6与存储器中的值后，从ROB中移走。保留栈中也暂存了相关指令的操作数信息。虽然第2条Load指令与MULT指令和SUB指令之间存在数据相关（寄存器F2），但是到第5周期该load指令已经完成了写结果，F2值已旁路转发到ROB和相应功能部件中，MULT与SUB指令可从中获得源操作数，进入执行阶段。



第6周期结束，第2条load指令完成确认，更新寄存器F2与存储器的值，然后从ROB中移走。F2更新为M2值，保留站中的F2内容也相应更新。

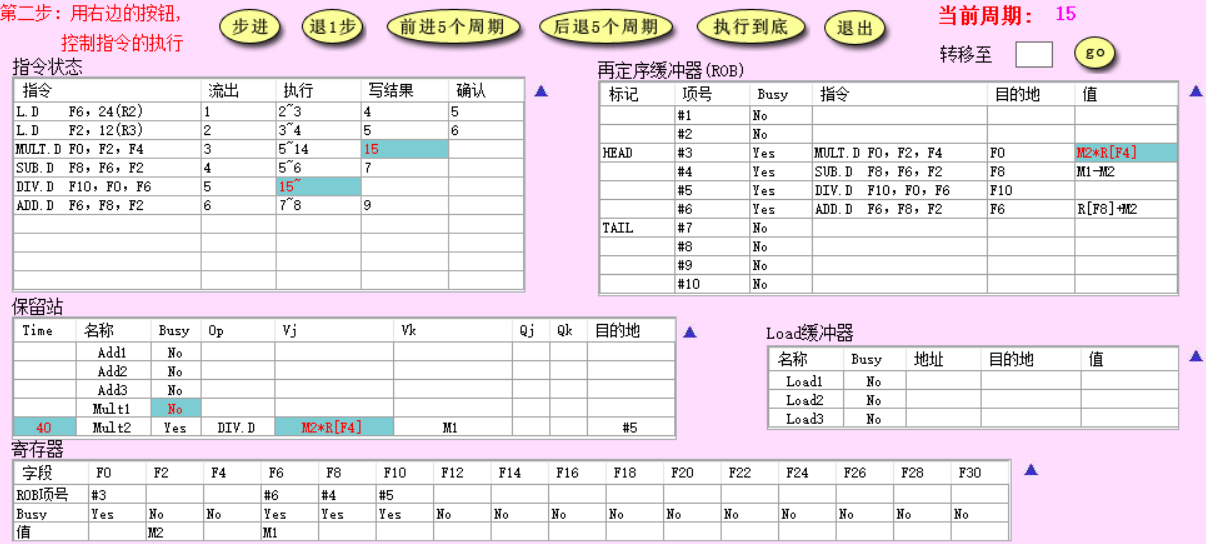


到第7周期SUB指令已完成写结果，F8值已旁路转发到ROB和相应功能部件中，ADD指令可从中获得源操作数F8的值，进入执行阶段。

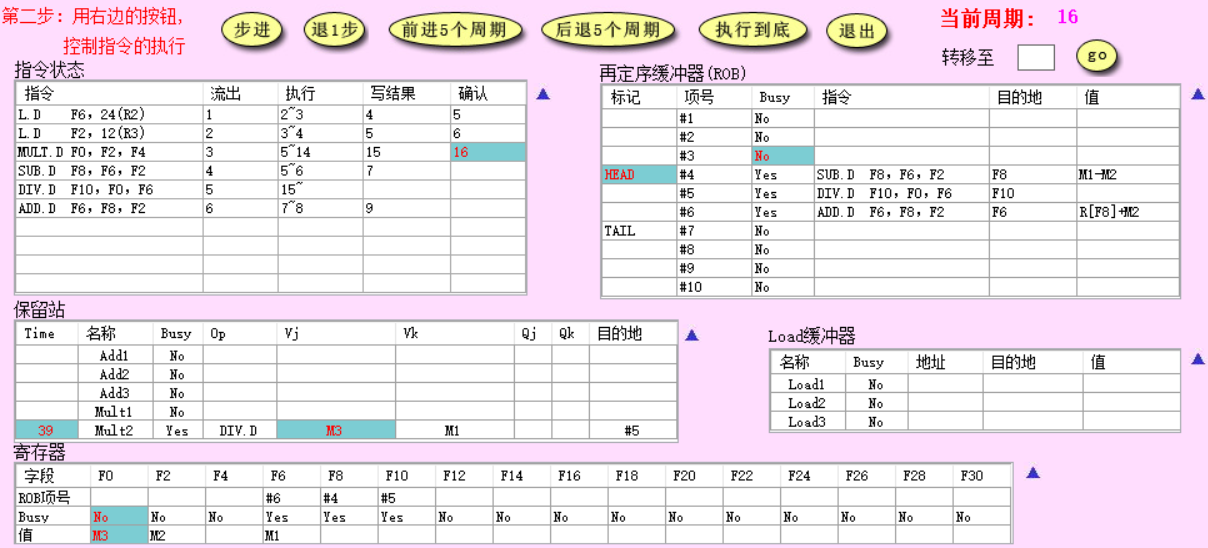




此处10-14周期的截图省略。第5-14周期是MULT的执行阶段。



到第15周期，MULT指令完成写结果，F0值已旁路转发到ROB和相应功能部件中，DIV指令可从中获得源操作数F0的值，进入执行阶段。



到第16周期结束，MULT指令完成确认，更新寄存器F0与存储器的值，然后从ROB中移走。F0更新为M3值，保留站中的F0内容也相应更新。



到第17周期结束，SUB指令完成确认，更新寄存器F8与存储器的值，然后从ROB中移走。F8更新为M4值。

此处第18-54周期的截图省略，第15-54周期是DIV指令的执行阶段。



到第55周期，DIV指令完成写结果，F10值已旁路转发到ROB和相应功能部件中。



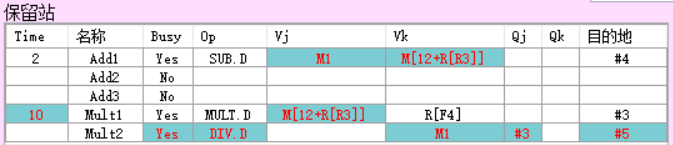
到第56周期结束，DIV指令完成确认，更新寄存器F10与存储器的值，然后从ROB中移走。F10更新为M5值。



到第57周期结束，ADD指令完成确认，更新寄存器F6与存储器的值，然后从ROB中移走。F6更新为M6值。

2、对于与上面相同的延迟时间和代码段：

①给出在第5个时钟周期时，保留站的内容。



②步进5个时钟周期，ROB的内容有哪些变化？

在第5个周期时，ROB的内容如下：



在此基础上再步进5个周期，此时为第10个时钟周期，ROB的内容如下。

第2条load指令完成确认，更新寄存器F2及存储器以后，从ROB中移走；SUB和ADD指令先后完成写结果，将计算得到的目的地F8、F6的值先旁路转发给了ROB和相应功能部件，供给寄存器相关的指令使用，实现了基于硬件的猜测。等待确认（即指令提交）以后再更新寄存器和存储器的值。



③再步进5个时钟周期，给出这时保留站、ROB以及浮点寄存器状态表的内容。

在②的基础上再步进5个周期，此时为第15个时钟周期，保留站、ROB以及浮点寄存器状态表的内容如下：



3、假设浮点功能部件的延迟为：加减法3个时钟周期，乘法8个时钟周期，除法40个时钟周期。自己编写一段程序（要在实验报告中给出），重复上述步骤2的工作。

（1） 自己编写的程序如下：

L.D F6, 24(R2)

L.D F2, 12(R3)

MUL.D F0, F2, F4

ADD.D F8, F2, F6

DIV.D F10, F0, F6

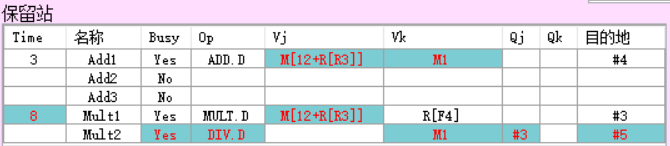
SUB.D F6, F8, F2

（2）重复上述步骤2中的工作：

先在ROB算法模拟器中重新输入指令，然后根据要求修改各个浮点功能部件的延迟时间，随后开始逐步执行程序。



① 给出在第5个时钟周期时，保留站的内容。



②步进5个时钟周期，ROB的内容有哪些变化？

在第5个周期时，ROB的内容如下：



在此基础上再步进5个周期，此时为第10个时钟周期，ROB的内容如下。

第2条load指令完成确认，更新寄存器F2及存储器以后，从ROB中移走；ADD指令完成写结果，将计算得到的目的地F8的值先旁路转发给了ROB和相应功能部件，供给寄存器相关的指令（SUB指令）使用，实现了基于硬件的猜测。等待确认（即指令提交）以后再更新寄存器F8和存储器的值。



③再步进5个时钟周期，给出这时保留站、ROB以及浮点寄存器状态表的内容。

在②的基础上再步进5个周期，此时为第15个时钟周期，保留站、ROB以及浮点寄存器状态表的内容如下：



1. **实验中出现的问题和解决方法**

·实验中关于重排序缓冲（ROB）算法模拟器的使用

**问题：**该实验中运用ROB算法模拟器，主要是为了更清楚地学习掌握ROB算法的执行流程，体会该算法在基于硬件猜测过程中的优越性；而如果在输入代码段后直接采用“执行到底”按钮，则只能得到执行结果，无法观察到程序执行的细节部分，不利于我们对指令执行过程的掌握。

**解决方案**：先尽量多使用ROB算法模拟器上的“步进”按钮，进行步进形式执行，仔细观察每一个时钟周期前后各信息表中的内容变化情况，了解ROB算法中各程序执行的详细过程；有需要的情况下再灵活使用“退1步”“前进5个周期”“后退5个周期”等按钮，辅助观察程序执行情况。

实验总结：

当使用ROB（重排序缓冲器）算法来执行指令时，与Tomasulo算法相比，ROB最大的不同之处在于增加了再定序缓冲器。

再定序缓冲器可以看作是一个循环队列，使用HEAD和TAIL来标记队列的首尾。每次只能提交队列首部的指令，而新加入的指令只能放入队列尾部，这样就保证了指令的顺序提交，实现了精确中断。

另外，再定序缓冲器还可以方便地清空预先执行但不应执行的指令（可能由于分支预测错误），而不提交它们，以避免对程序正确性产生影响。这样就可以消除控制相关性问题。

由于指令执行完成后不直接将结果写入寄存器，ROB还负责在指令完成后到指令提交阶段为其他指令提供数据。此外，保留站中记录的目的地以及Qj和Qk（源操作数标记）都变成了ROB的项号。这意味着保留站中的指令执行完成后会直接写入ROB而不是写入寄存器，并且保留站会在遇到原始数据相关冲突时，在公共数据总线（CDB）上监听写入ROB特定项号的数据，以获得所需的源操作数。

ROB的其余执行流程与Tomasulo算法相似，包括指令调度、保留站的占用与释放、指令的执行和结果提交等步骤。整体上，ROB通过引入再定序缓冲器来提供更精细的指令顺序控制，解决了控制相关性和数据相关性的问题，从而提高了指令执行的效率和并行性。