控制冒险

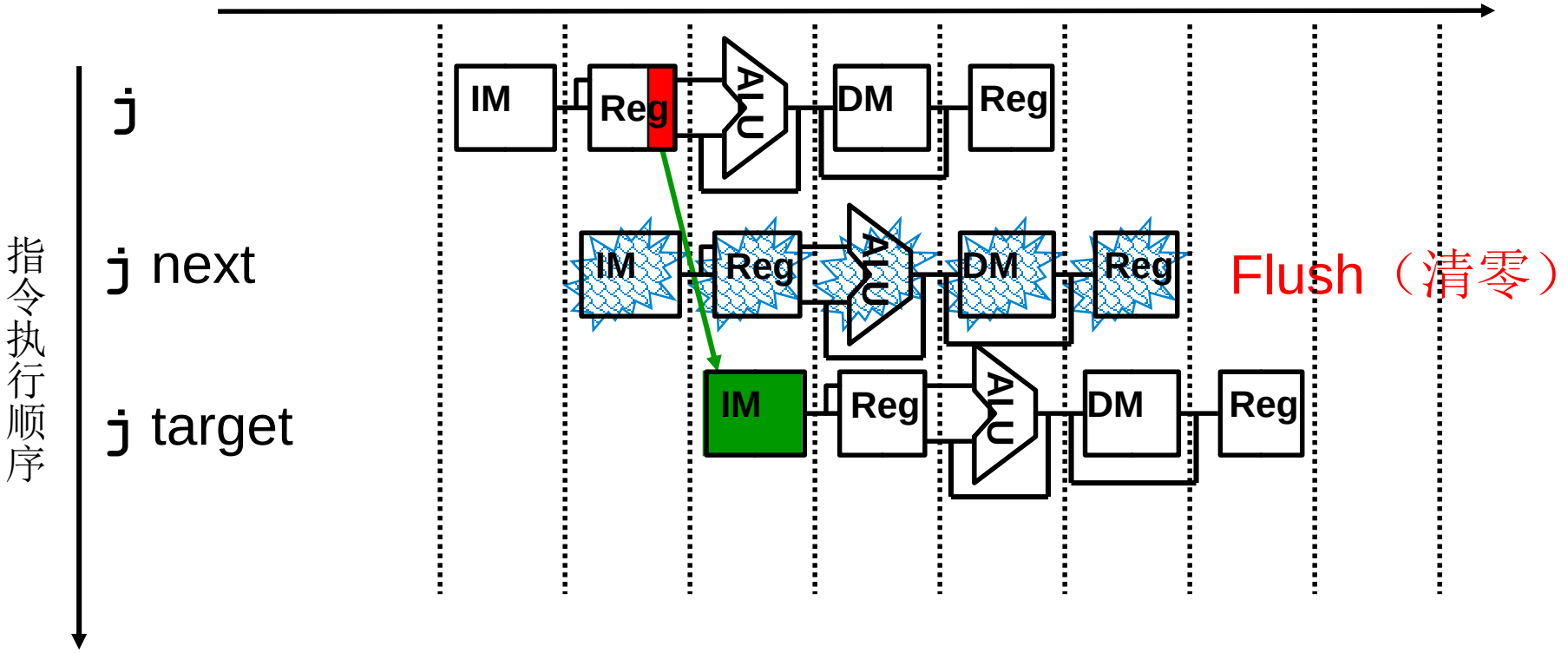
控制冒险

- 控制冒险（control hazards）
  - 由转移指令引起
    - 无条件转移(j, jal, jr)：跳转到指定位置
    - 条件转移(beq, bne)
      - 转移成功：将PC值改变为转移目标地址

当转移指令的结果还未确定定时，无法决定下一步执行哪条指令。

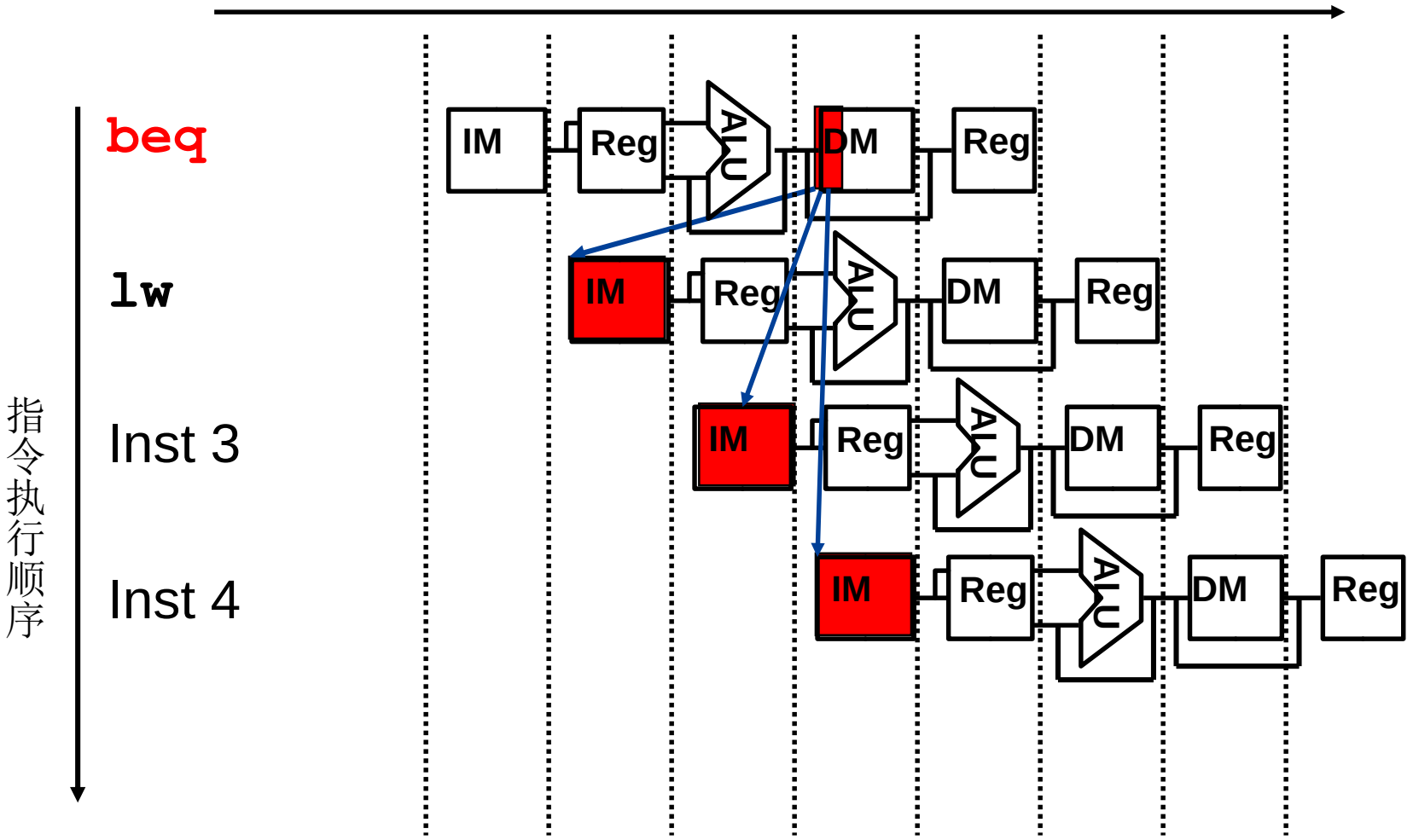
无条件转移引发控制冒险

- Jump 指令
  - 在 ID 段译码，此时IF段已经取了它后面的指令
  - 需要将IF/ID 段中的指令清零（ flush）



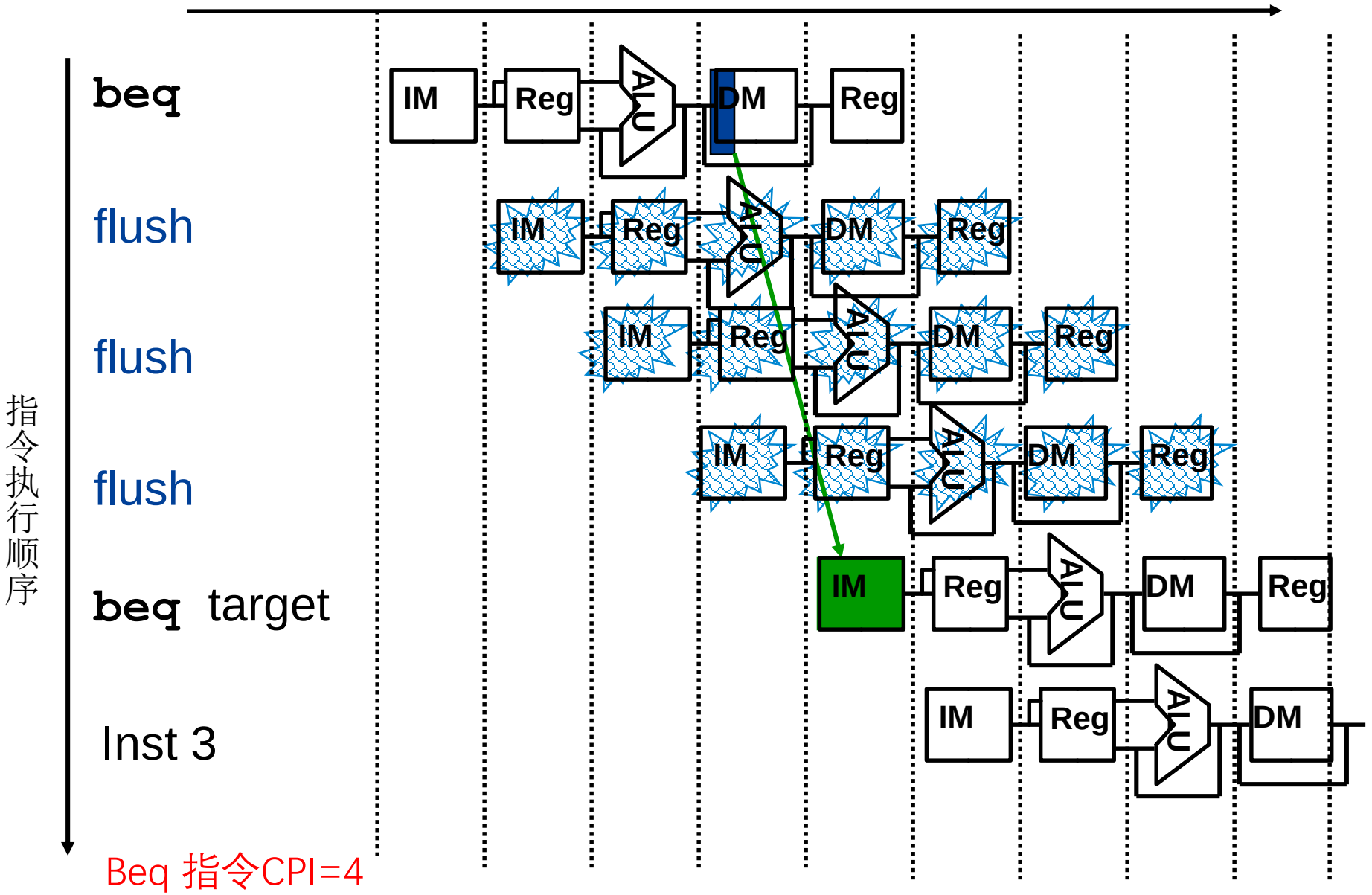
- Jump 指令的 CPI=2

条件转移指令（分支指令）引发控制冒险



- 新PC的值要在MEM段才写入，引发了控制冒险

解决控制冒险最简单的方法：清空



控制冒险对CPI的影响

- 指令流水线:理想CPI＝1
- 若条件转移指令（分支指令）的频率为30%,
- 条件转移语句：如果转移会导致浪费3个时钟周期

  则：实际CPI（cycle per instr.） ＝1＋30%×3≈ 2

- 控制冒险会严重影响CPI
  - 流水段越长，影响越大
  - Pentium 3：转移开销 10周期
  - Pentium 4： 转移开销 20周期

控制冒险的解决方案

- 减少开销的解决方案:
  - 尽量早开始转移决策
  - 转移延迟槽(需要编译器支持)

分支提前决策

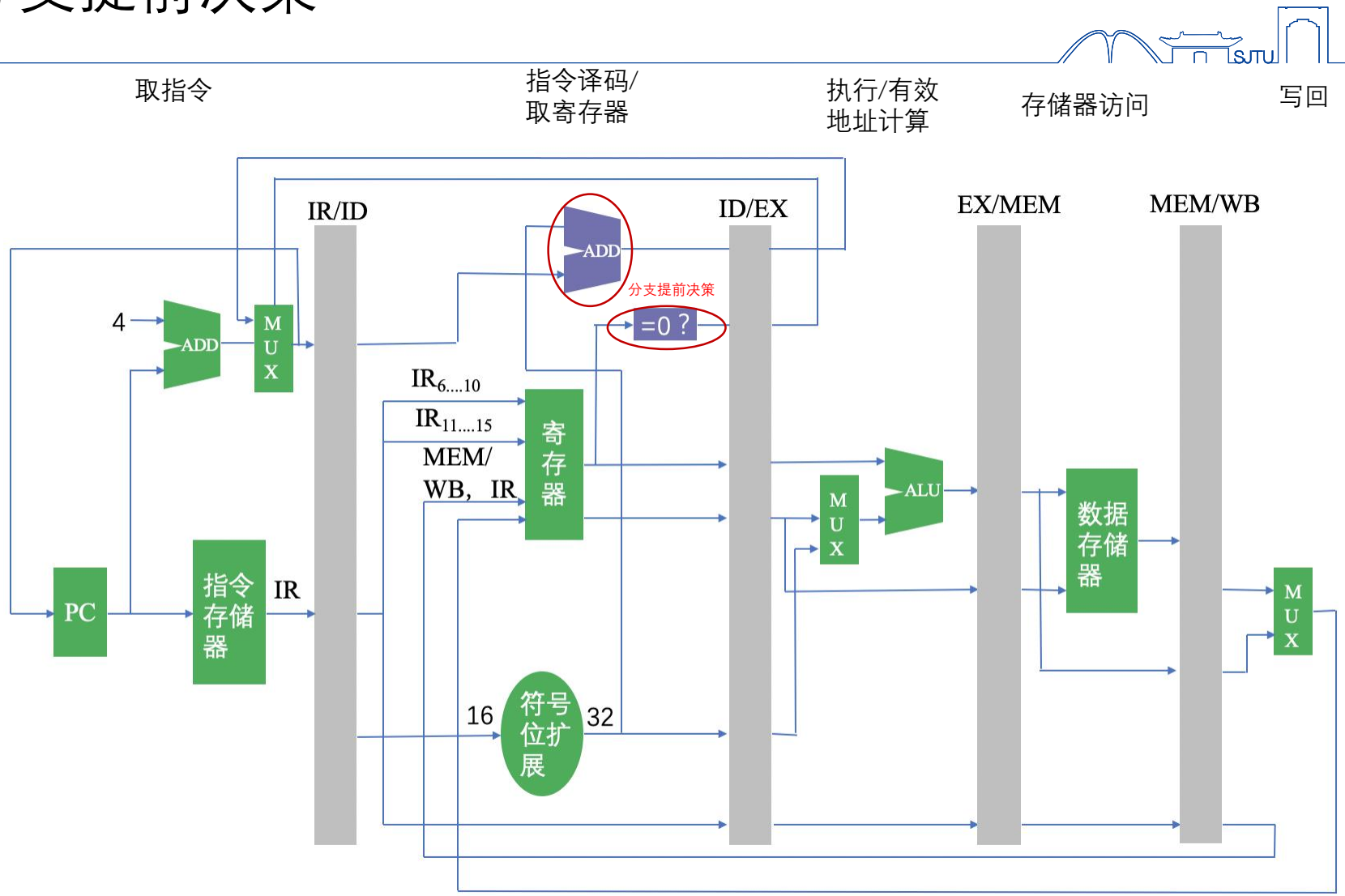
- 在流水线中尽早判断转移条件是否满足
- 并尽早计算出转移目标地址

- 对五阶段指令流水线作以下改进：

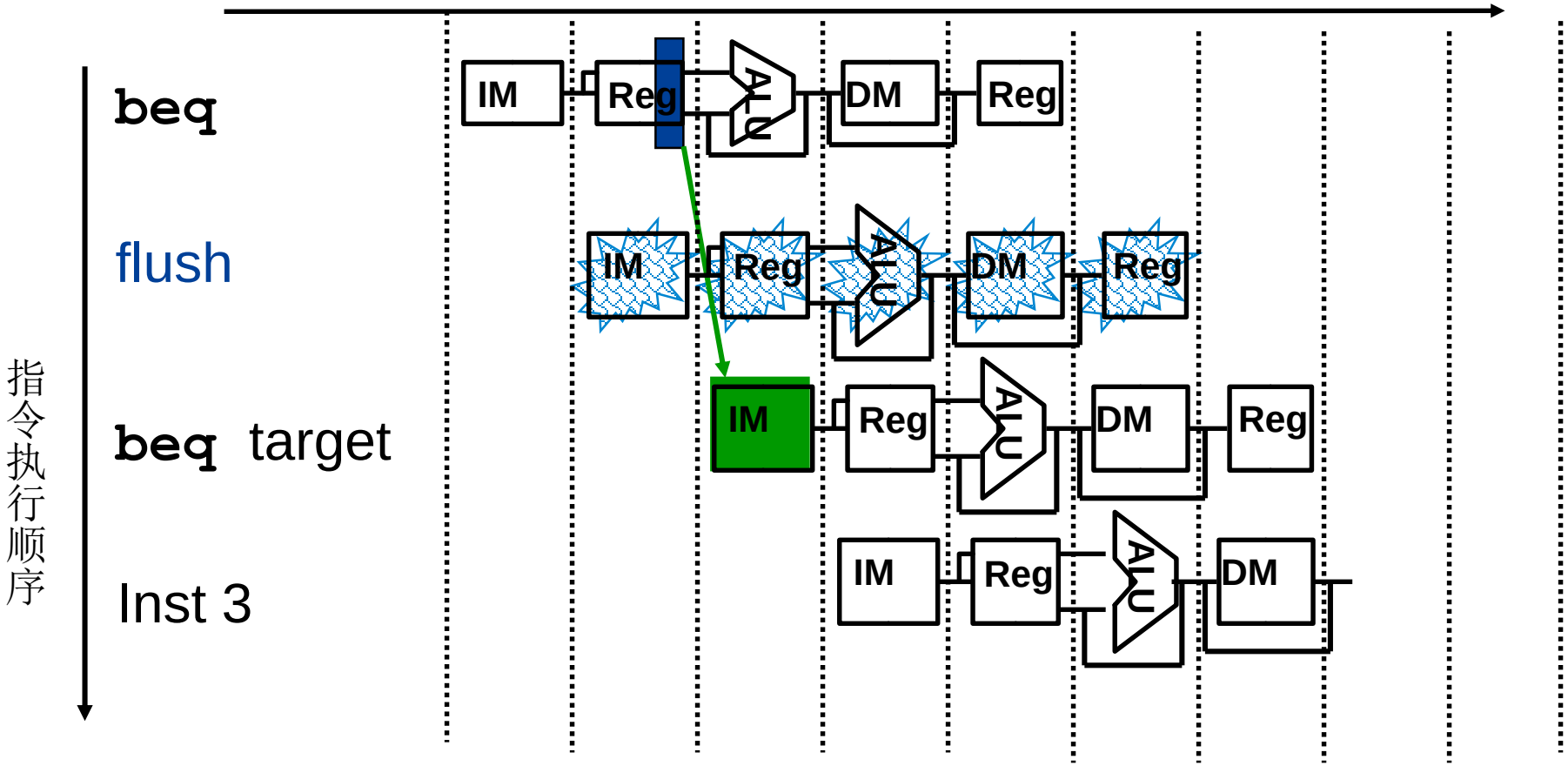
(1) 把“=0？”测试移至ID段；

(2) 在ID段增设一个加法器，计算出转移目标地址

分支提前决策


取指令
指令译码/
取寄存器
执行/有效
地址计算
存储器访问
写回
IR/ID
ID/EX
EX/MEM
MEM/WB
ADD
分支提前决策
=0 ?
4
ADD
M
U
X
IR6....10
IR11....15
MEM/
WB, IR
寄
存
器
PC
指令
存储
器
IR
16
符号
位扩
展
32
M
U
X
ALU
数据
存储
器
M
U
X

分支提前决策

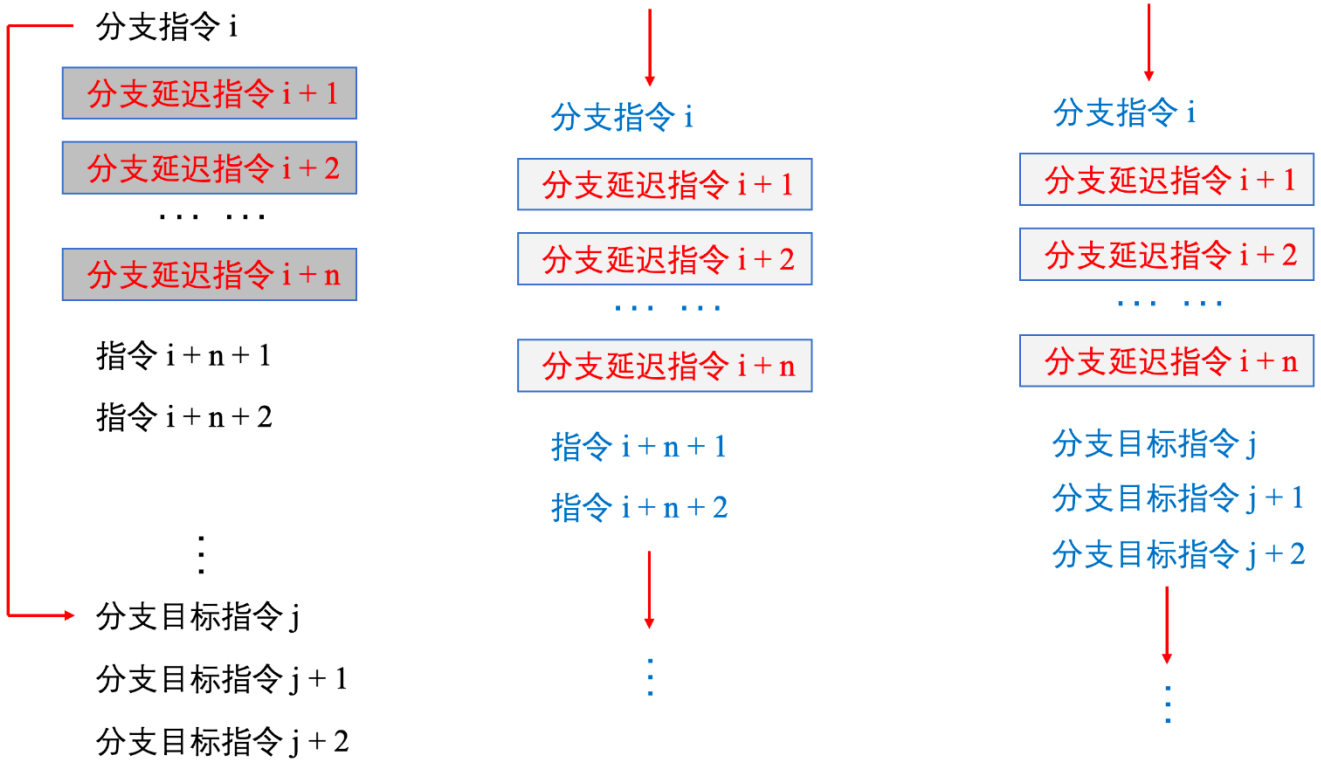


仍然需要停顿一周期：清空一条指令

转移延迟槽（Branch Slot）

- 基本思想:
  - 将一条（或几条）肯定会执行的指令移到branch指令后面，隐藏转移指令的延迟;
  - 需要编译器支持

转移延迟槽大小=n


分支指令 i
分支延迟指令 i + 1
分支延迟指令 i + 2
分支延迟指令 i + n
指令 i + n + 1
指令 i + n + 2
分支目标指令 j
分支目标指令 j + 1
分支目标指令 j + 2
分支失败时
的执行顺序
分支成功时
的执行顺序

转移延迟槽

- 转移延迟槽的大小n
  - 体系结构与编译器之间的约定
  - 一般为分支指令的转移延迟
  - 五段流水无提前决策：n=3
  - 五段流水无提前决策：n=1
  - 流水段级数多的情况下，转移延迟槽远远超过1，这对编译器要求高。

小结

- 控制冒险
- 控制冒险的解决方案
  - 清空 (对性能影响大)
  - 提前转移决策
  - 转移延迟槽