计算机组织结构

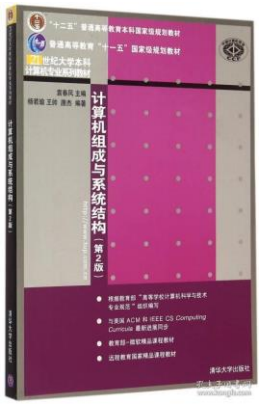
15 指令周期和指令流水线

刘博涵

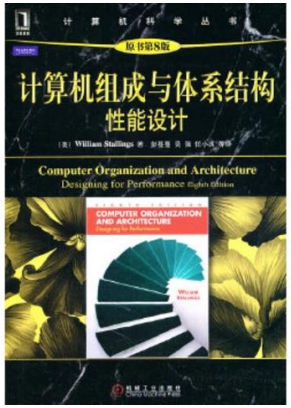
2023年12月7日

南京大学
NANJING UNIVERSITY

教材对应章节



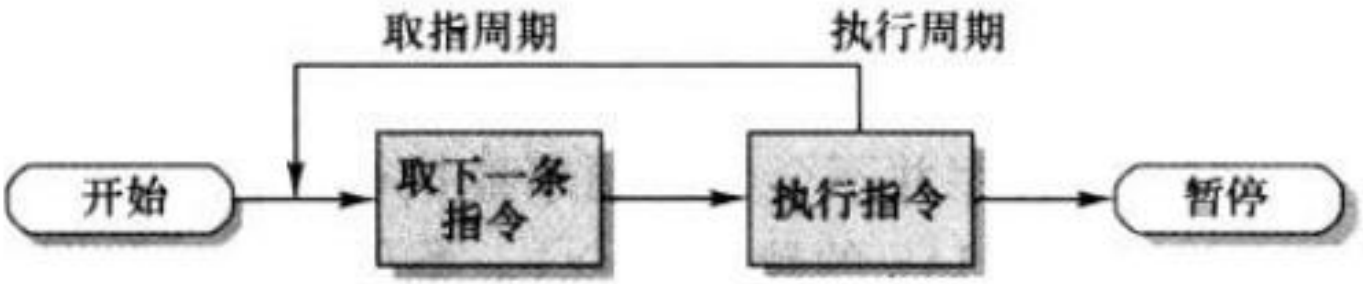
第5章 中央处理器
第6章 指令流水线



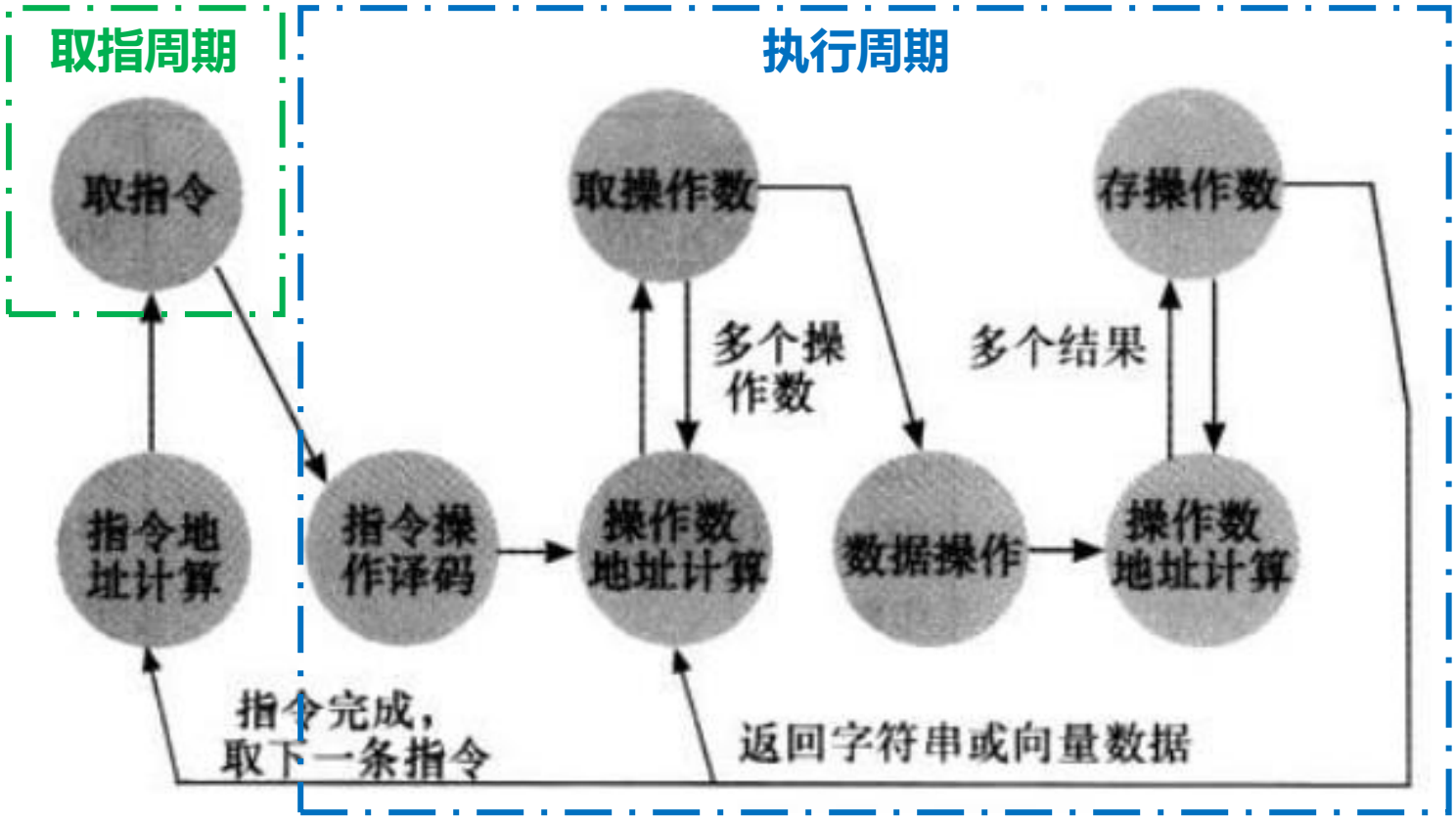
第12章 CPU结构和功能

指令周期

- 指令周期：处理单个指令的过程（时间）
  - 取指周期：从内存中提取一条指令
  - 执行周期：执行所提取的指令
- 只有当机器关闭、发生某种不可恢复的错误或遇到停止计算机的程序指令时，程序执行才会停止

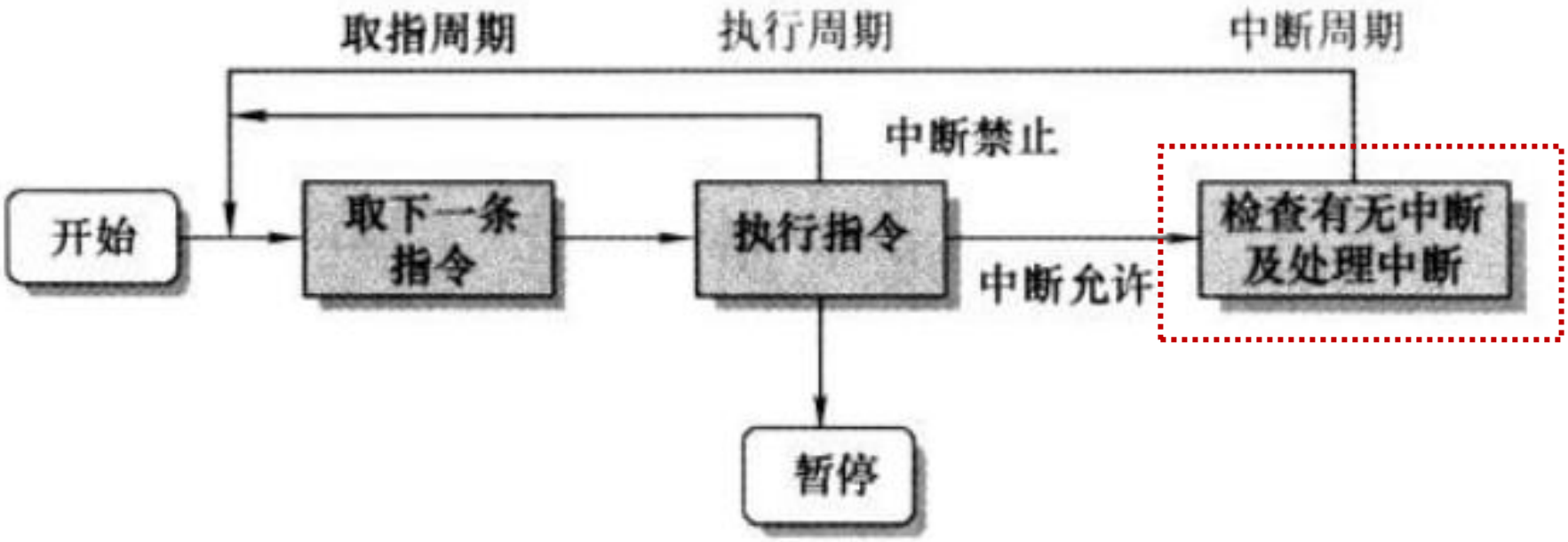


指令周期：状态图

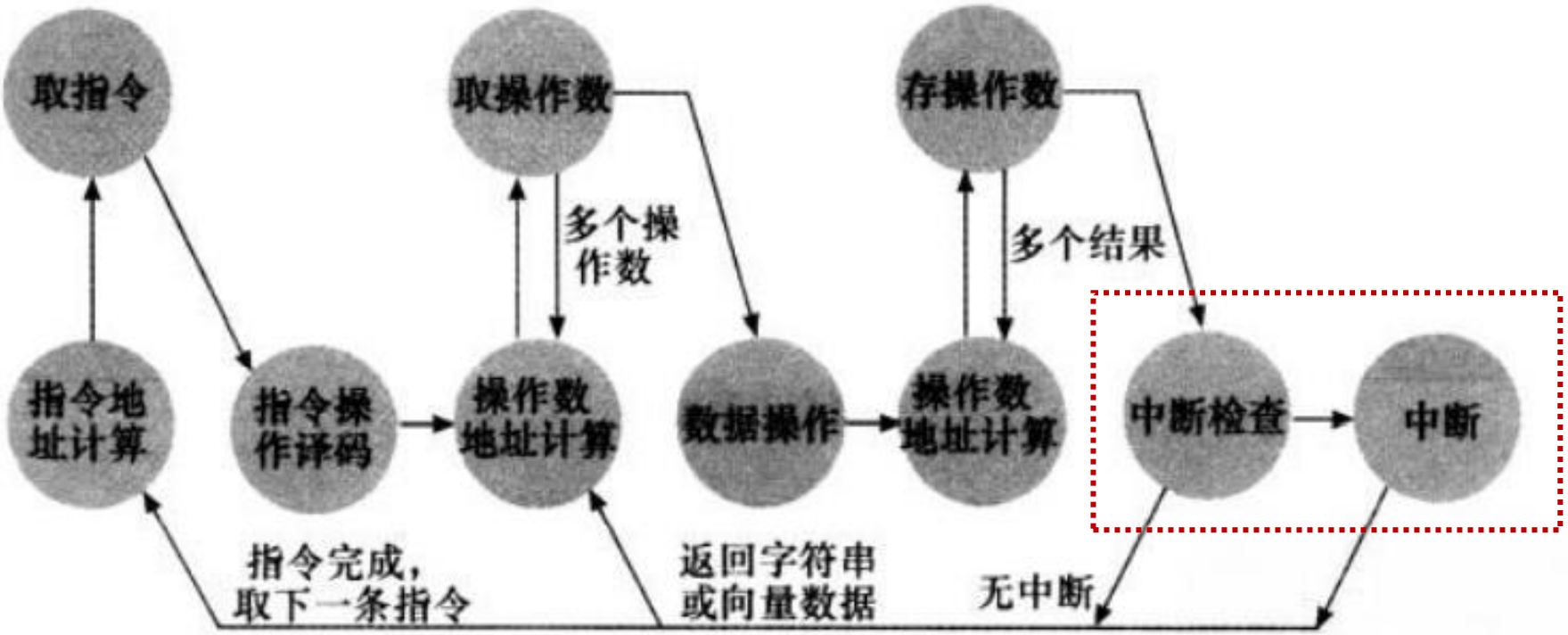


并非所有指令的周期都一样
例如，NOP（空操作）只有取指周期

带中断的指令周期

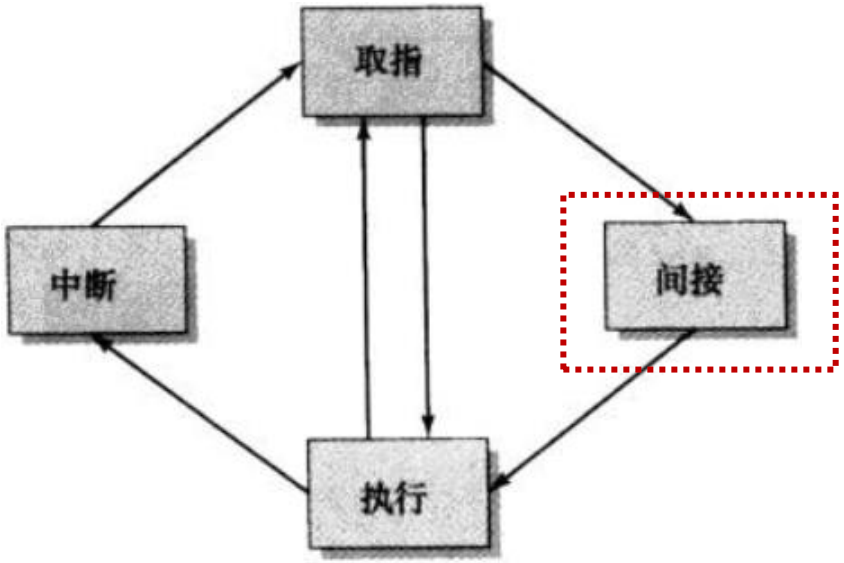


带中断的指令周期：状态图

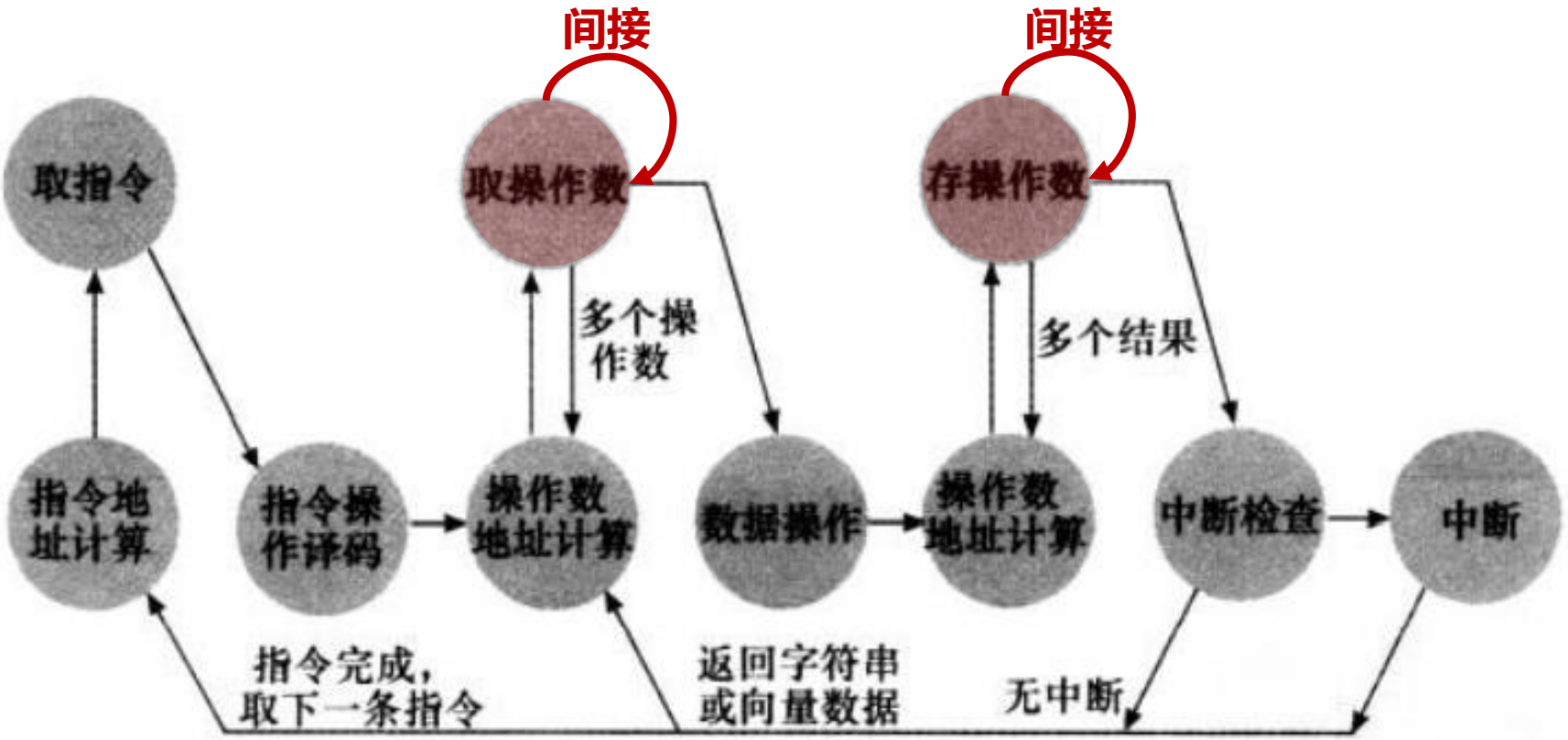


间址周期

- 指令的执行可能涉及一个或多个存储器中的操作数，它们每个都要求一次存储器访问
- 使用间接寻址，还需要额外的存储器访问
- 间址周期：把间接地址的读取看成是一个额外的指令子周期



间址周期：状态图



取操作数发生了2次

- 根据地址取有效地址（间址周期）
- 根据有效地址取操作数

CPU的任务

- **取指令**：CPU必须从存储器（寄存器、cache、主存）读取指令
- **解释指令**：必须对指令进行译码，以确定所要求的动作
- **取数据**：指令的执行可能要求从存储器或输入/输出（I/O）模块中读取数据
- **处理数据**：指令的执行可能要求对数据完成某些算术或逻辑运算
- **写数据**：执行的结果可能要求写数据到存储器或I/O模块

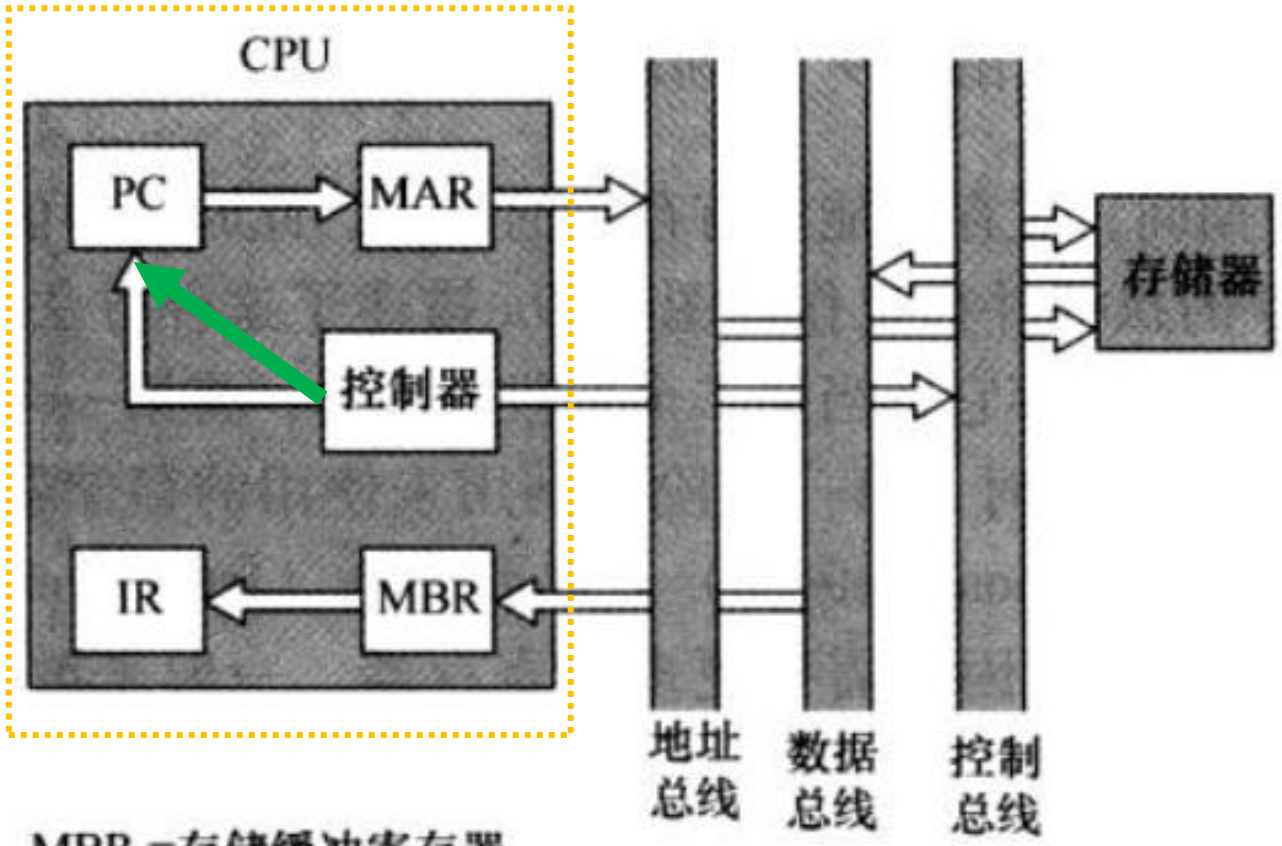
CPU需求：寄存器

- CPU需要在指令周期中临时保存指令和数据
- CPU需要记录当前所执行指令的位置，以便知道从何处得到下一条指令

CPU需要一些小容量的内部存储器

- 假定CPU有：
  - 1个存储地址寄存器（MAR）
  - 1个存储缓冲寄存器（MBR）/ 存储数据寄存器（MDR）
  - 1个程序计数器（PC）
  - 1个指令寄存器（IR）

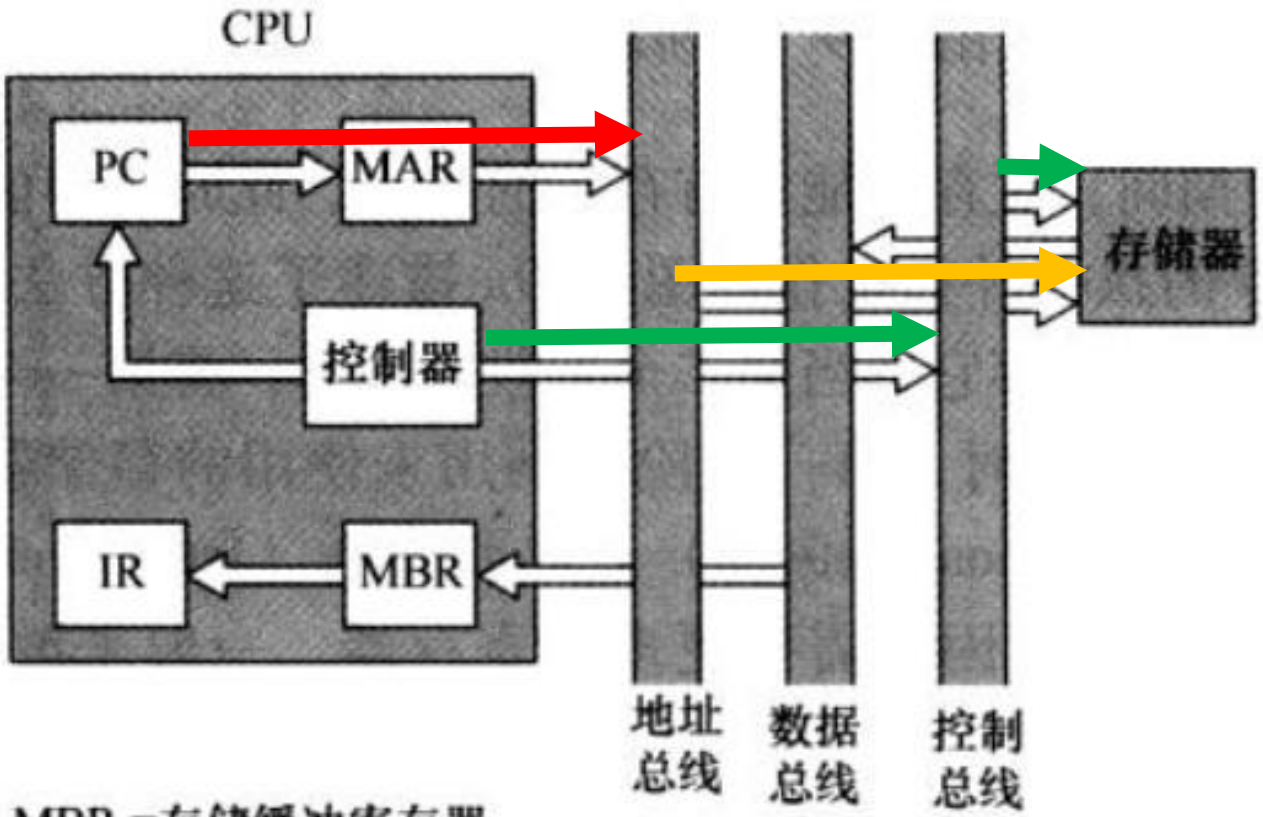
数据流：取指周期



MBR =存储缓冲寄存器
MAR =存储地址寄存器
IR = 指令寄存器
PC =程序计数器

控制器下达指令：取指周期的开始

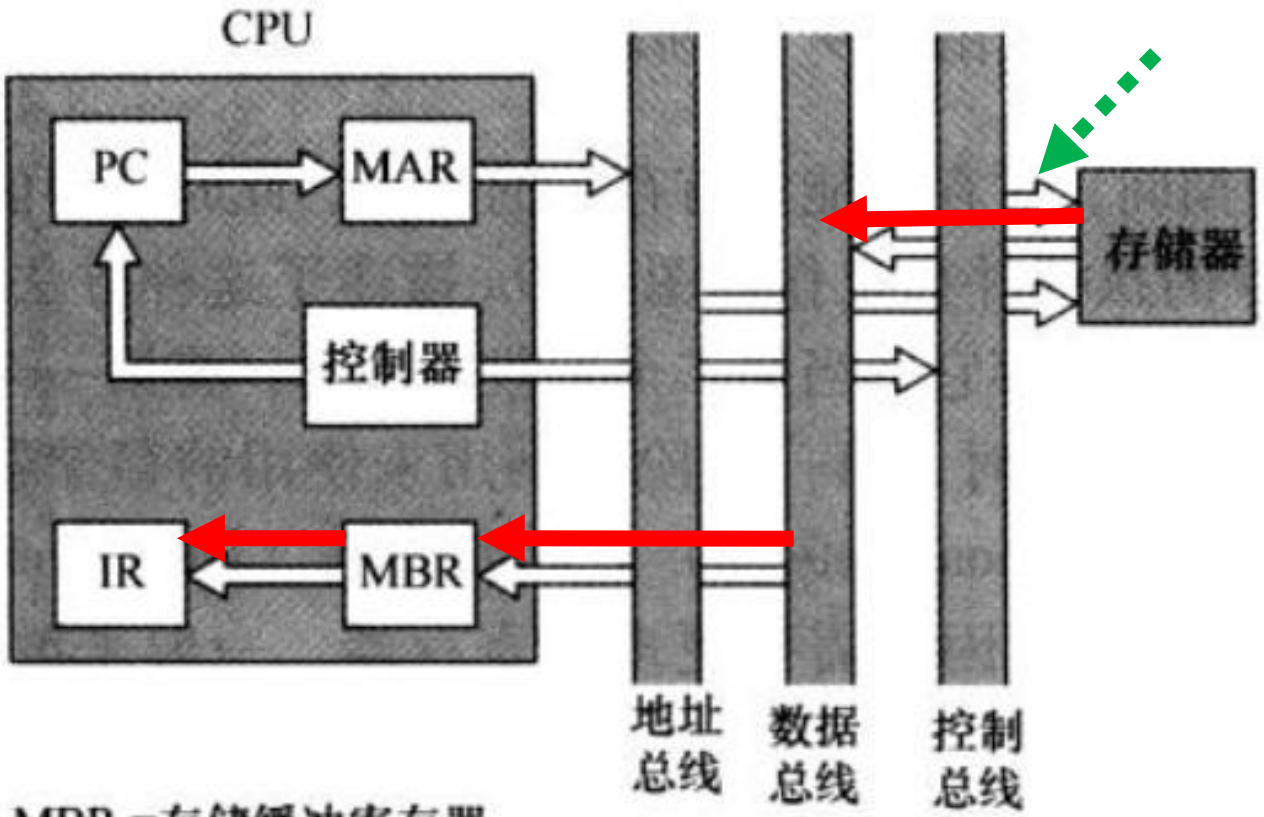
数据流：取指周期（续）



MBR =存储缓冲寄存器
MAR =存储地址寄存器
IR = 指令寄存器
PC =程序计数器

通过MAR将地址传入地址总线
控制器通过控制线通知存储器地址就绪
存储器读取地址

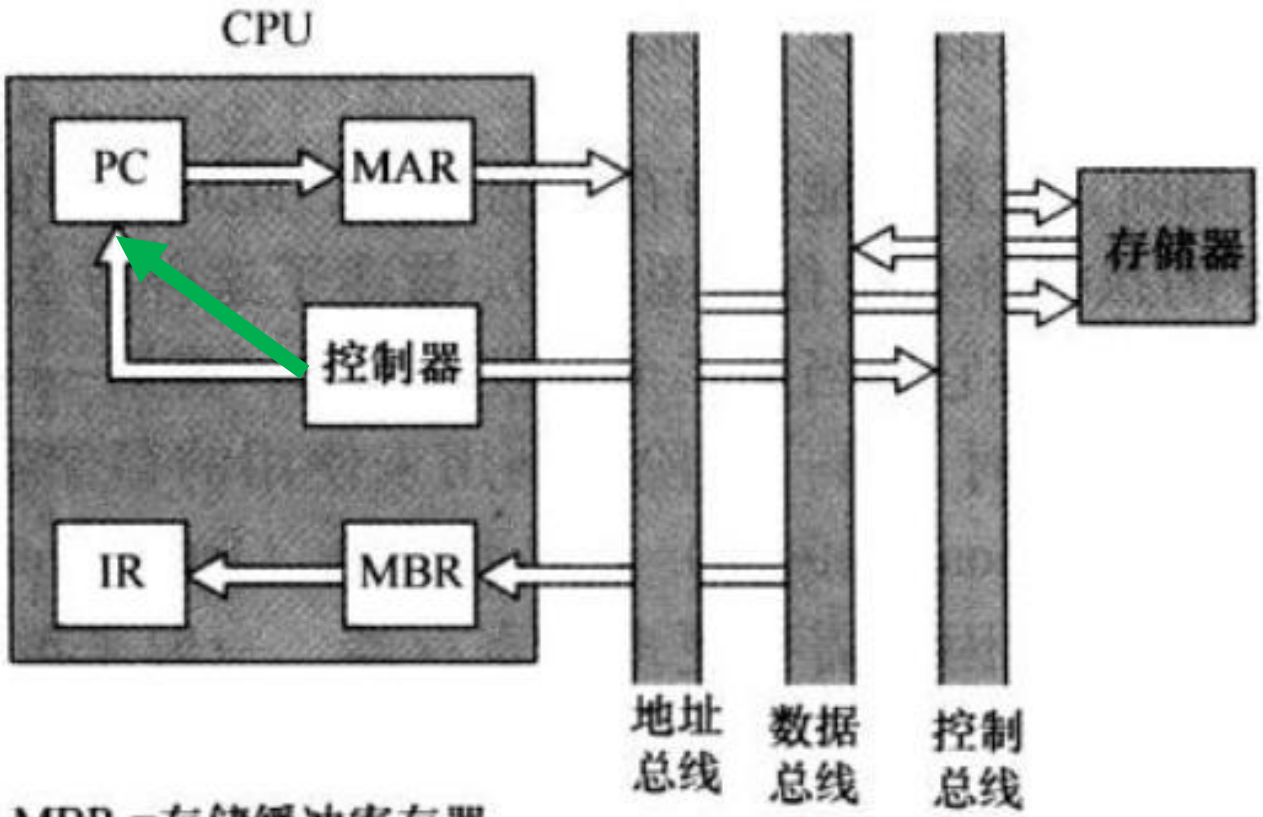
数据流：取指周期（续）



MBR =存储缓冲寄存器
MAR =存储地址寄存器
IR = 指令寄存器
PC =程序计数器

存储器通过数据总线将数据发送给MBR
如果是异步总线，存储器提供反馈

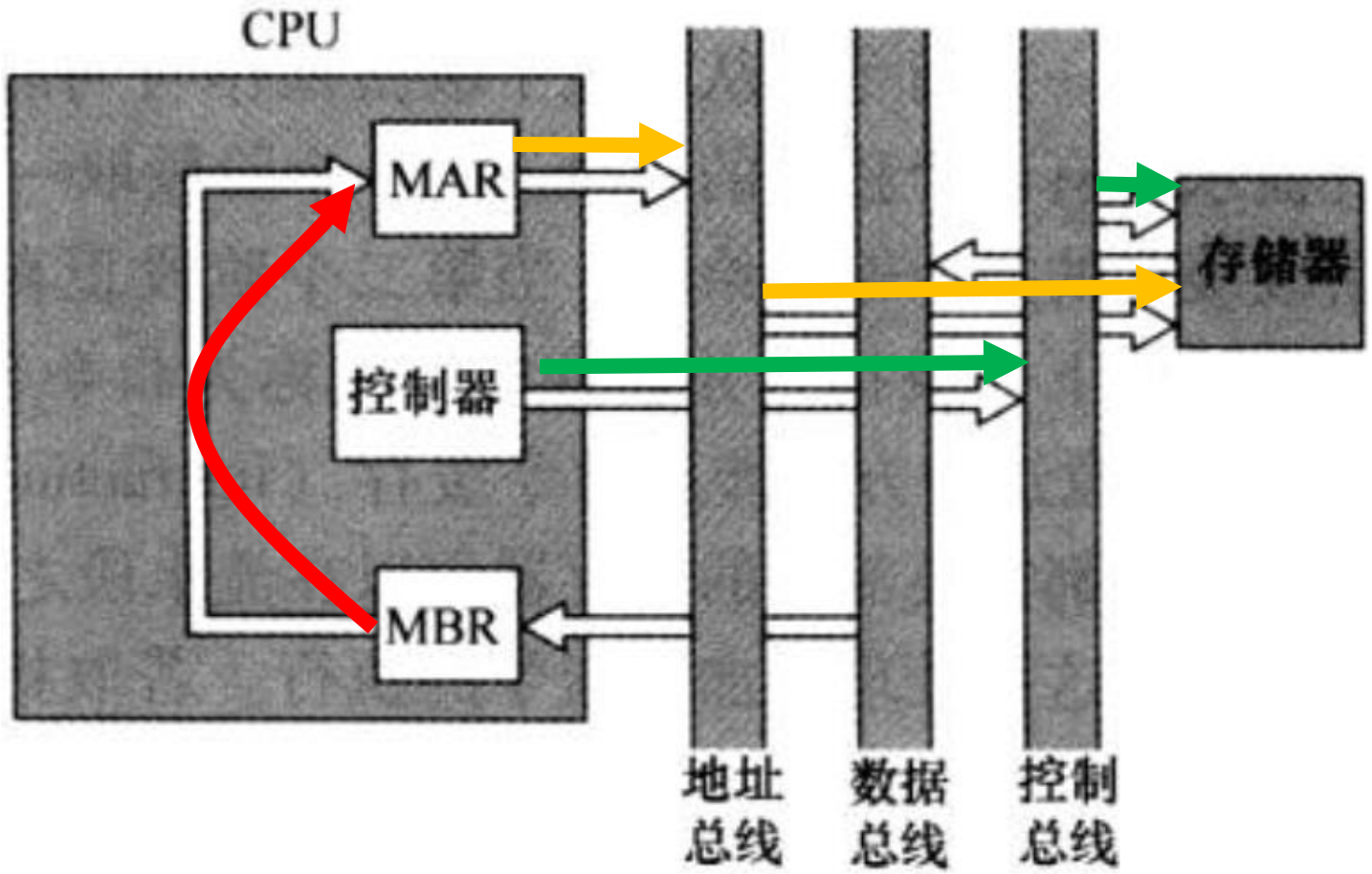
数据流：取指周期（续）



MBR =存储缓冲寄存器
MAR =存储地址寄存器
IR = 指令寄存器
PC =程序计数器

指令取回来后，PC+“1”

数据流：间址周期

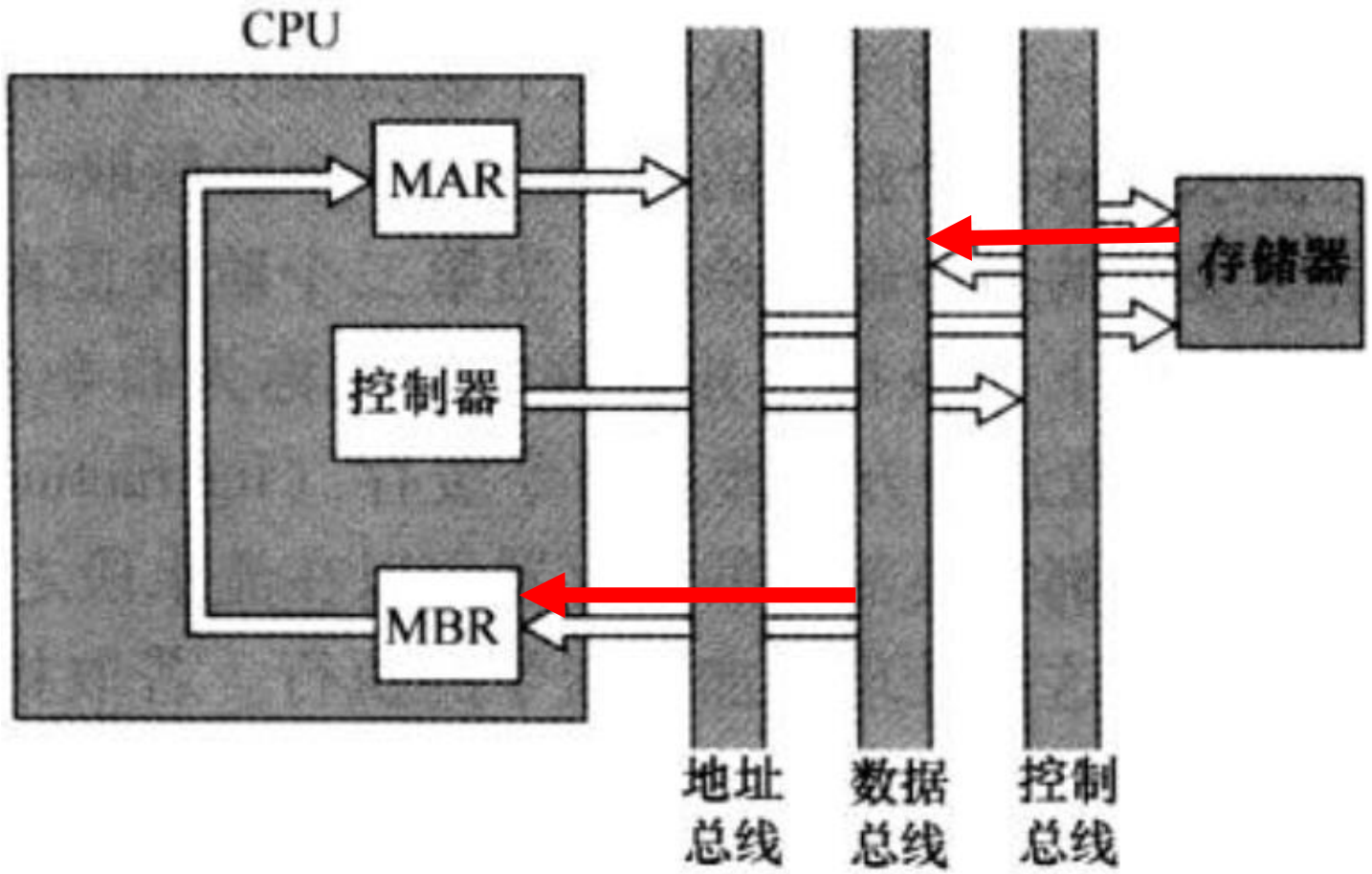


将MBR中的地址引用送入MAR得到地址

MAR将地址传入地址总线

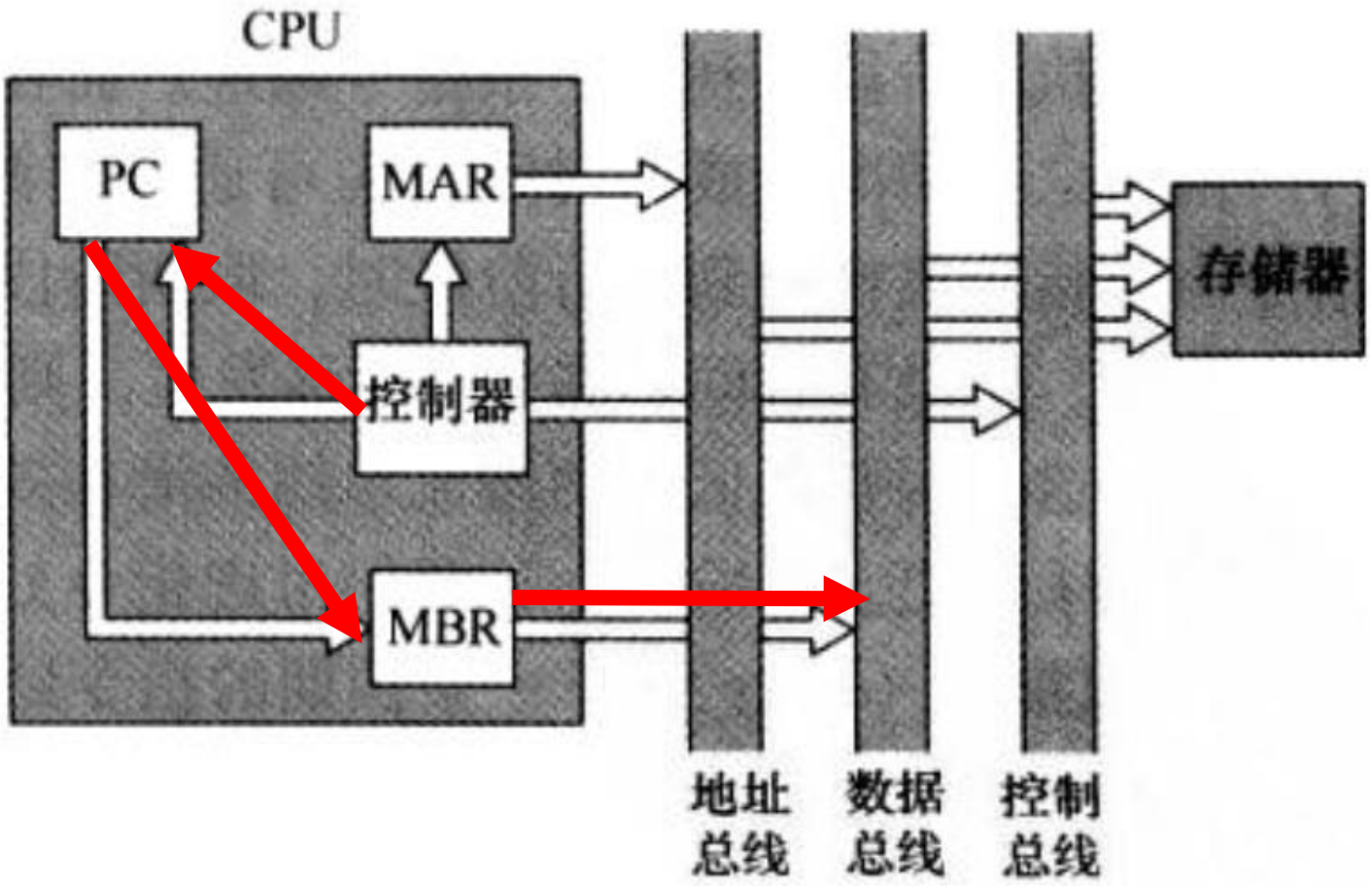
控制器通知存储器取地址

数据流：间址周期（续）



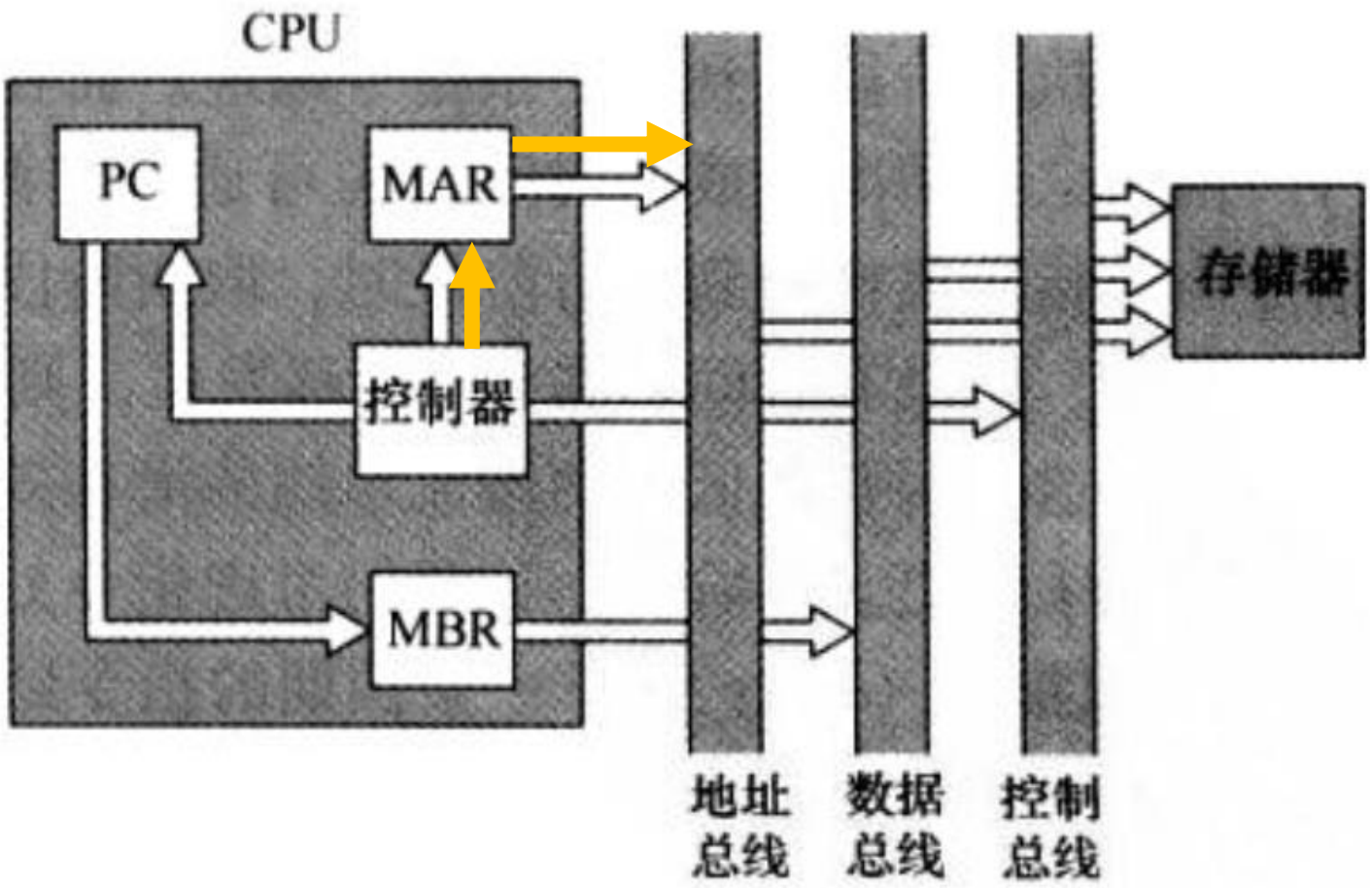
存储器通过数据总线将有效地址发送给MBR

数据流：中断周期



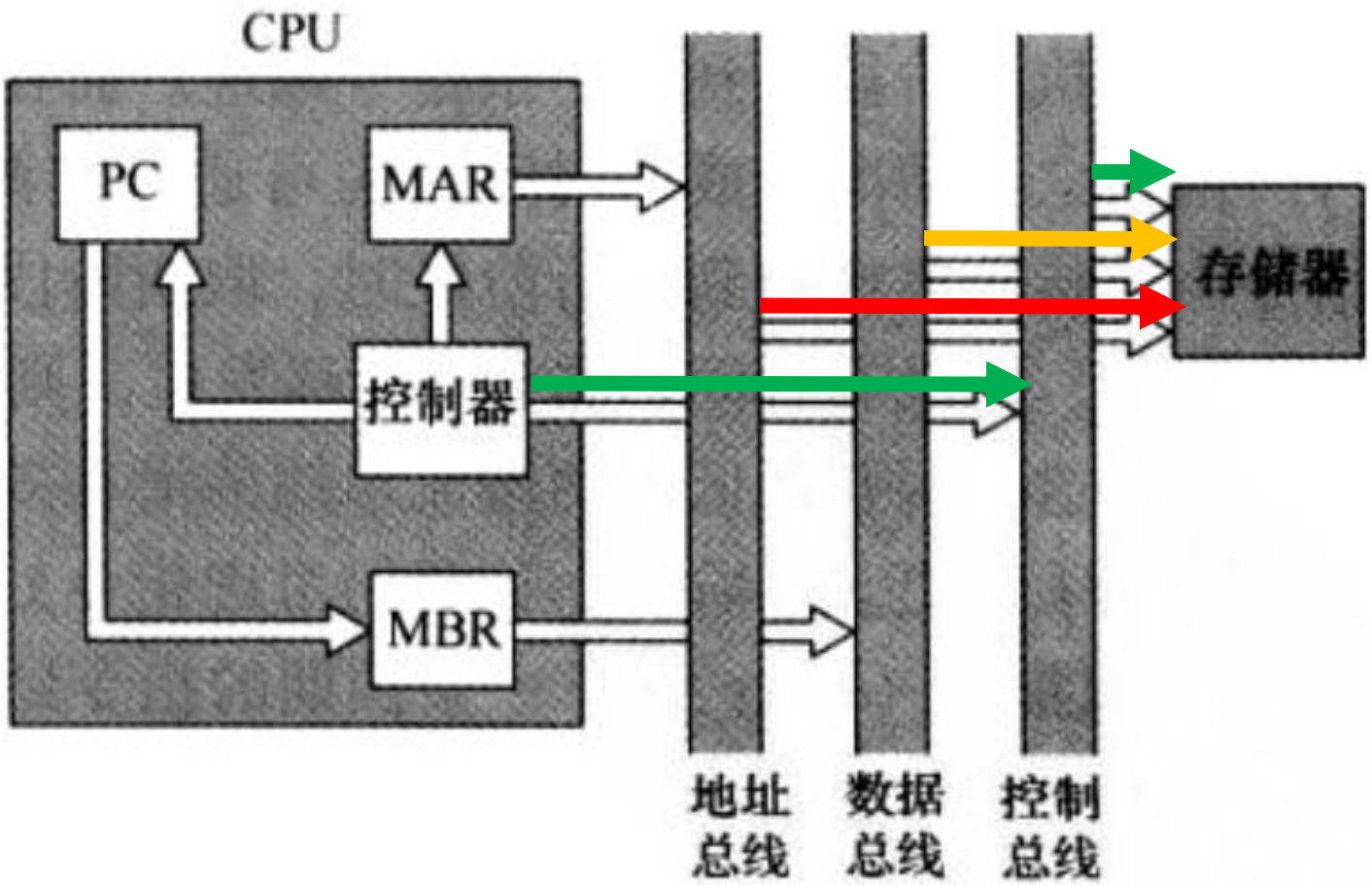
处理中断前，将下一条指令放到MBR中，再放到数据总线上

数据流：中断周期（续）



控制器将下一条指令的取值地址通过MAR放到地址总线上

数据流：中断周期（续）



控制器通知存储器获得数据

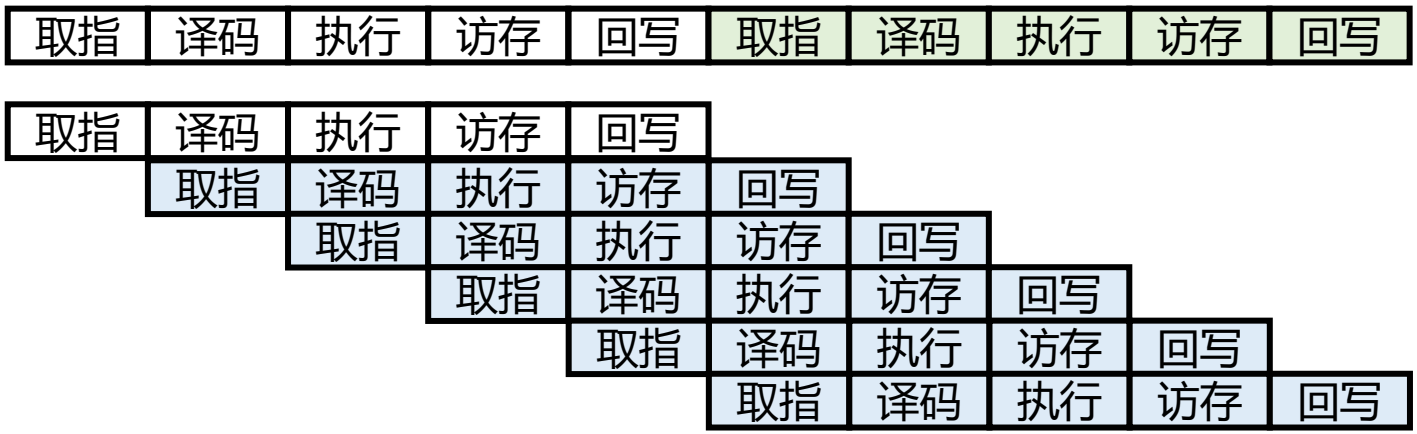
存储器从地址线获得地址

存储器从数据线获得数据，并将数据写入到获得的地址

指令流水线

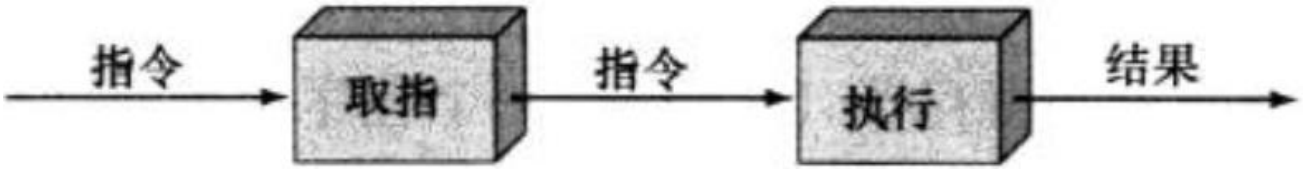
- **流水处理（pipelining）**
  - 如果一个产品要经过几个制作步骤，通过把制作过程安排在一条装配线上，多个产品能在各个阶段同时被加工

- **指令流水线**：一条指令的处理过程分成若干个阶段，每个阶段由相应的功能部件完成



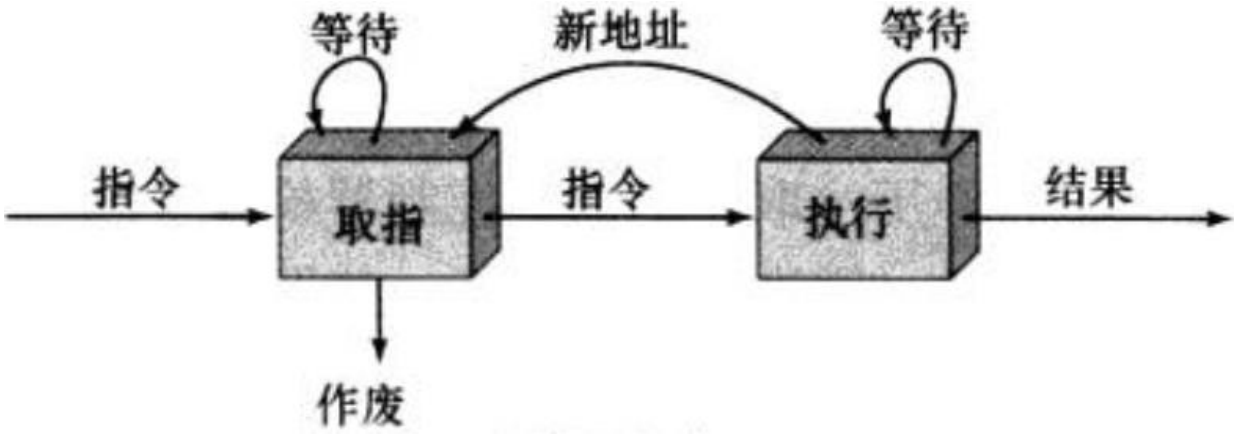
两阶段方法

- 将指令处理分成两个阶段：取指令和执行指令
- 在当前指令的执行期间取下一条指令
- **问题：执行时间一般要长于取指时间**



两阶段方法（续）

- 更多问题
  - 主存访问冲突
  - 条件分支指令使得待取的下一条指令的地址是未知的



六阶段方法

- 为了进一步的加速，流水线必须有更多的阶段
  - **取指令（Fetch instruction，FI）**：读下一条预期的指令到缓冲器
  - **译码指令（Decode instruction，DI）**：确定操作码和操作数指定符
  - **计算操作数（Calculate operands，CO）**：计算每个源操作数的有效地址
  - **取操作数（Fetch operands，FO）**：从存储器取出每个操作数，寄存器中的操作数不需要取
  - **执行指令（Execute instruction，EI）**：完成指定的操作。若有指定的目的操作数位置，则将结果写入此位置
  - **写操作数（Write operand，WO）**：将结果存入存储器
- 各个阶段所需要的时间几乎是相等的

六阶段方法（续）

- **例**：将9条指令的执行时间由54个时间单位减少到14个时间单位

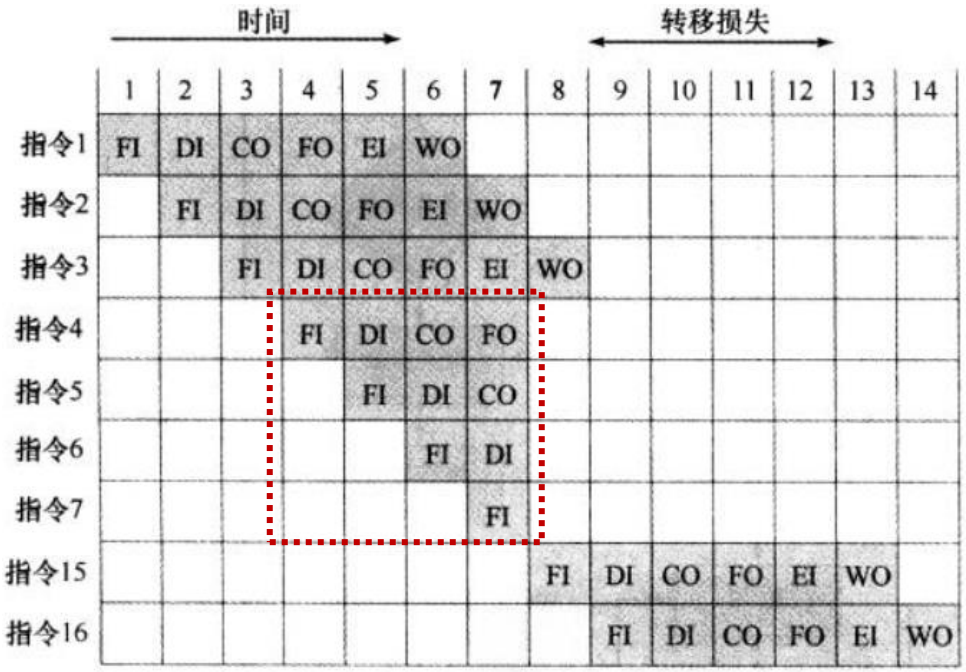
六阶段方法（续）

问题：

- 不是所有指令都包含6个阶段
  - 例：一条LOAD指令不需要WO阶段
  - 为了简化流水线硬件设计，在假定每条指令都要求这6个阶段的基础上来建立时序
- 不是所有的阶段都能并行完成
  - 例：FI、FO和 WO都涉及存储器访问
- 若6个阶段不全是相等的时间，则会在各个流水阶段涉及某种等待

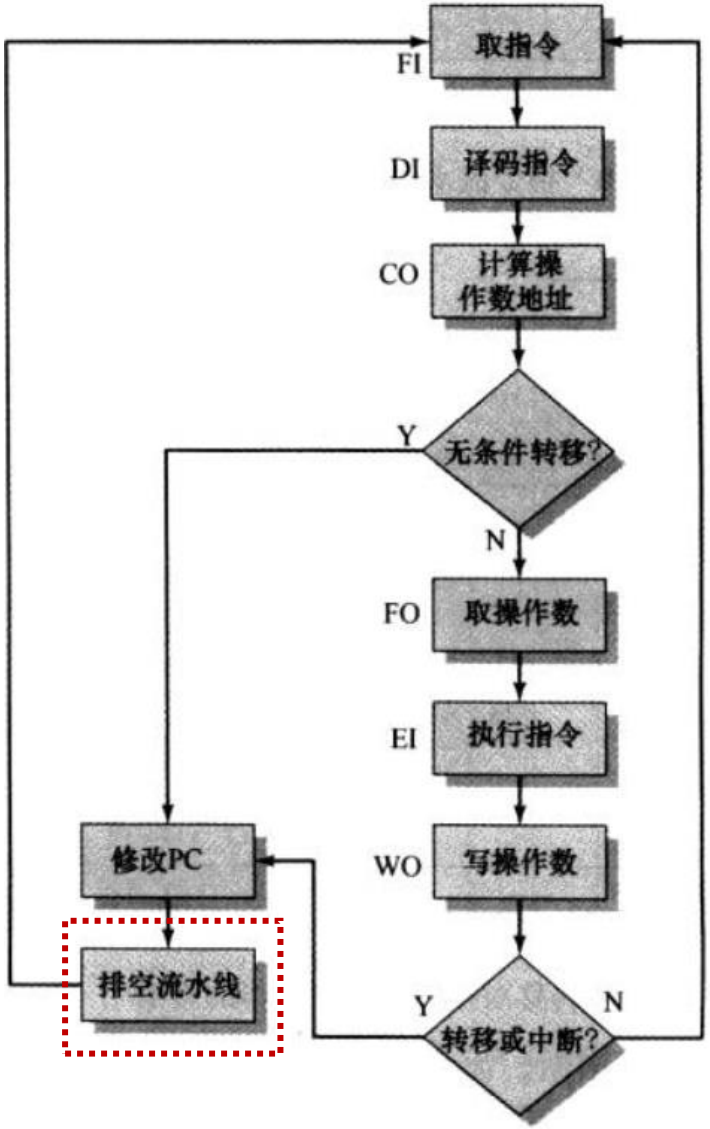
六阶段方法（续）

- **限制：条件转移指令能使若干指令的读取变为无效**



六阶段方法（续）

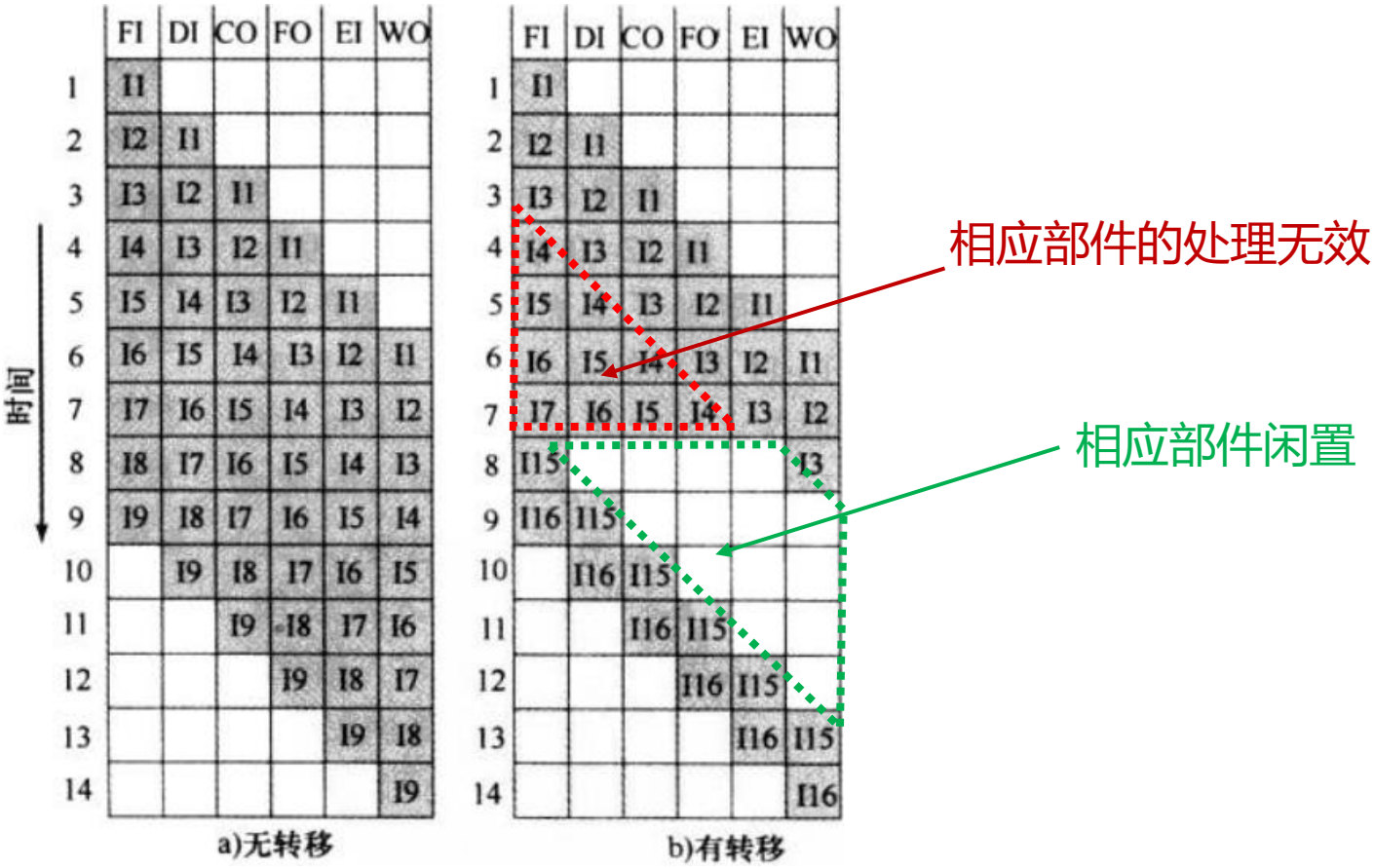
- **限制：中断**



提前处理到一半的过程都需要被清除

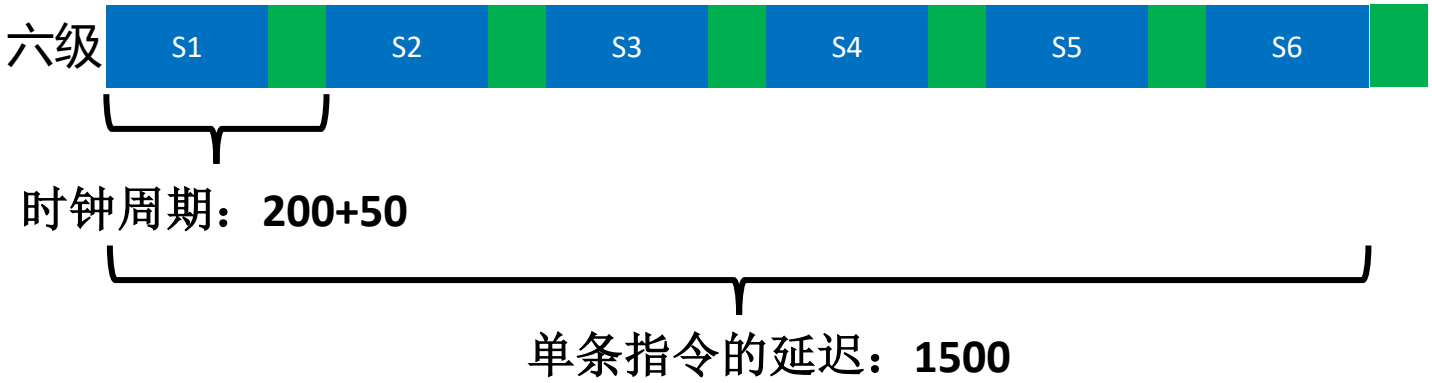
六阶段方法（续）

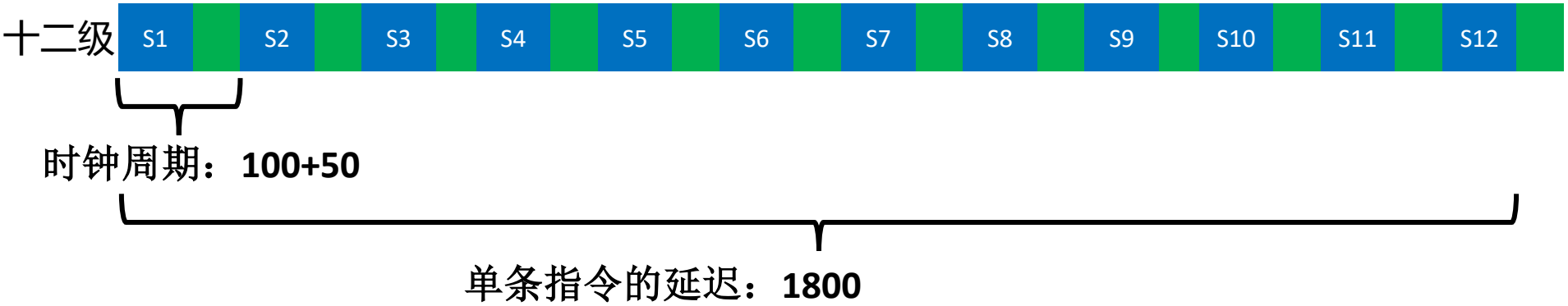
- 另一种描述方式（转90度，处理部件的视角）



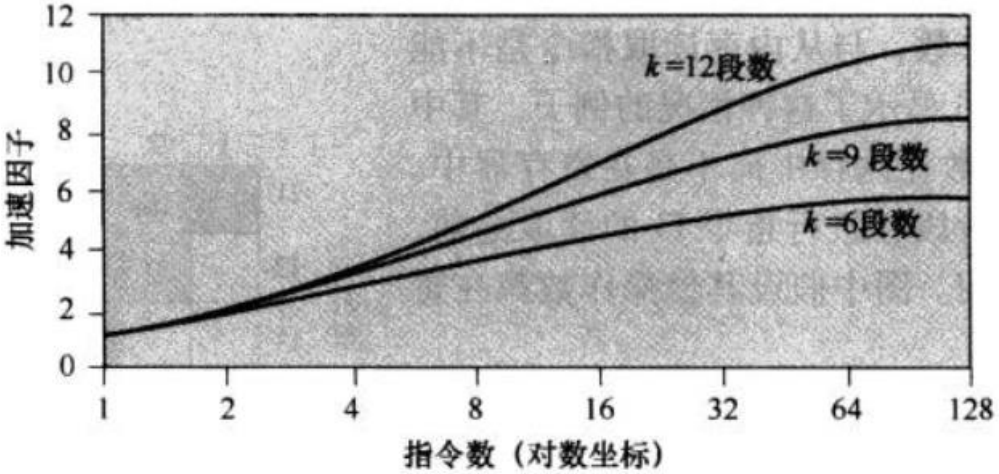
超流水线（Super-pipelining）

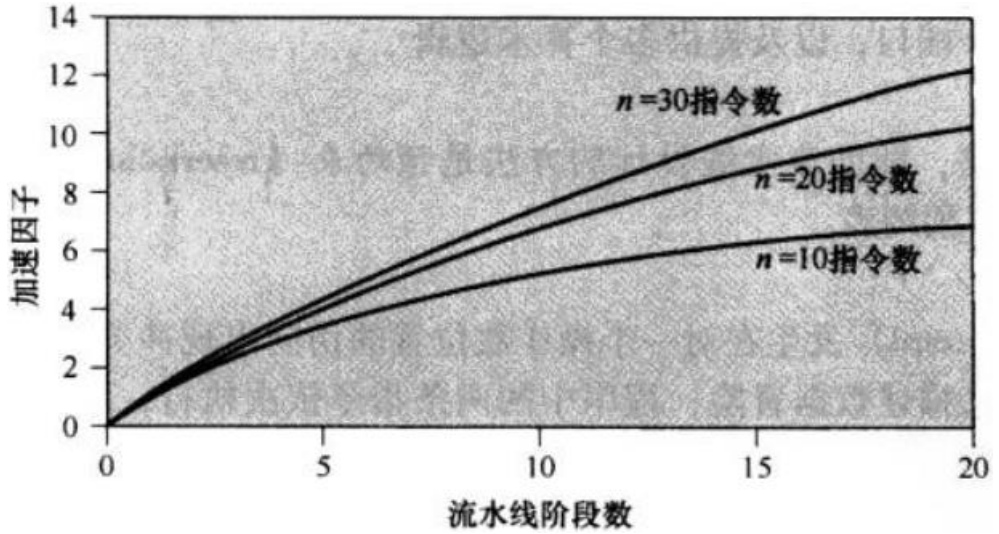
- 将六级流水线细分为更多的阶段，增加流水线的**深度**
- 提升时钟频率，从而提高指令吞吐率





流水线性能（续）


12
10
8
6
4
2
0
加速因子
k=12段数
k=9 段数
k=6段数
1
2
4
8
16
32
64
128
指令数（对数坐标）


14
12
10
8
6
4
2
0
加速因子
n=30指令数
n=20指令数
n=10指令数
0
5
10
15
20
流水线阶段数

流水线性能

- **假设**
  - t_i: 流水线第i段的电路延迟时间
  - t_m: 最大段延迟（通过耗时最长段的延迟）
  - k: 指令流水线段数
  - d: 锁存延时（数据和信号从上一段送到下一段所需的段间锁存接收时间）

- **周期时间**

$$t = \max[t_i] + d = t_m + d$$

流水线性能（续）

- 令$T_{k,n}$为 k 阶段流水线执行所有 n 条指令所需的**总时间**

$$T_{k,n} = [k + (n-1)]t \qquad \text{（理想情况）}$$

- **加速比**

没有使用流水线

$$S_k = \frac{T_{1,n}}{T_{k,n}} = \frac{nkt}{[k + (n-1)]t} = \frac{nk}{k + (n-1)} = \frac{n}{1 + \frac{n-1}{k}} > 1$$

使用流水线

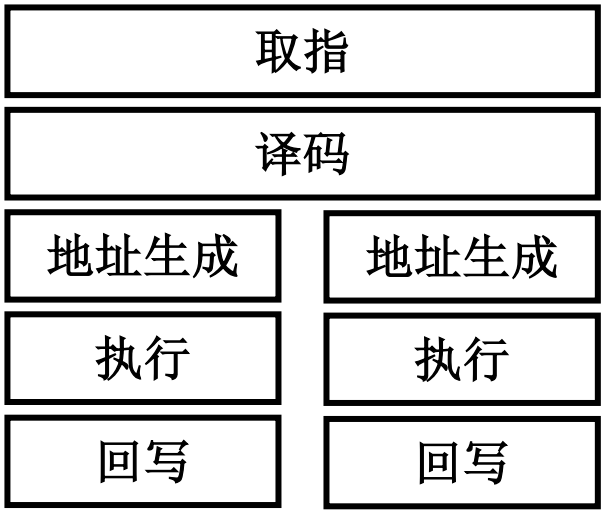
当考虑流水线失效时，分母不同

流水线性能（续）

- **误解**
  - 流水线中的阶段数越多，执行速度越快
- **原因**
  - 在流水线的每个阶段，将数据从一个缓冲区移动到另一个缓冲区以及执行各种准备和传递功能都涉及一些开销（比如锁存延时）
  - 处理内存和寄存器依赖以及优化管道使用所需的控制逻辑数量随着阶段的增加而急剧增加

超标量流水线（Superscalar）

- **超标量结构**：具有两条或两条以上并行工作的流水线结构



双发射5级流水线（第一代奔腾处理器）

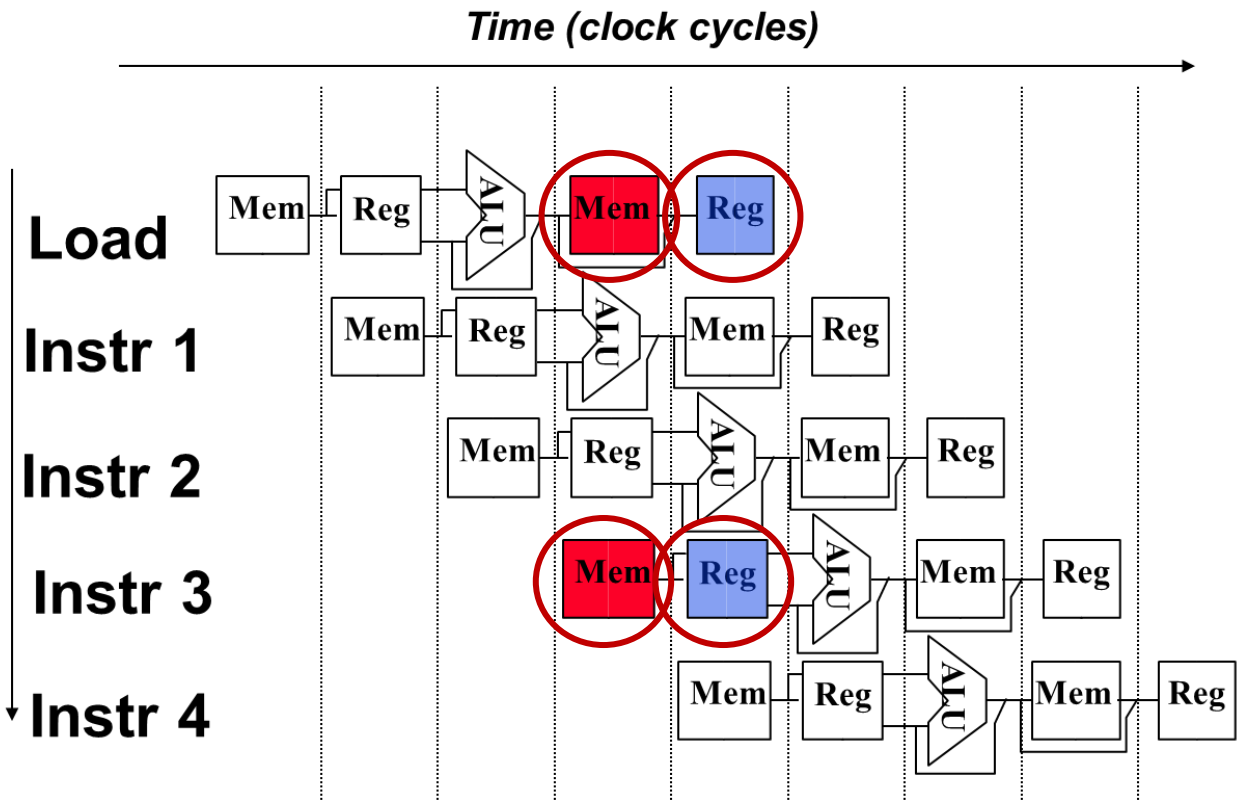
- **单周期→标量流水线**： **时间并行性**的优化， 主要是对现有硬件的分段
- **标量流水线→超标量流水线**： **空间并行性**的优化， 需成倍增加硬件资源
- **多核CPU**：一个CPU芯片中集成多个超标量处理器核

冒险（Hazard）

- 在某些情况下，指令流水线会阻塞或停顿（stall），导致后续指令无法正确执行
- **类型**
  - **结构冒险**（Structure hazard） / 硬件资源冲突
    - 不同指令同时使用相同的硬件资源，比如访存
  - **数据冒险**（Data hazard） / 数据依赖性
    - 有些数据要等前序计算完成
  - **控制冒险**（Control hazard）
    - 比如条件转移

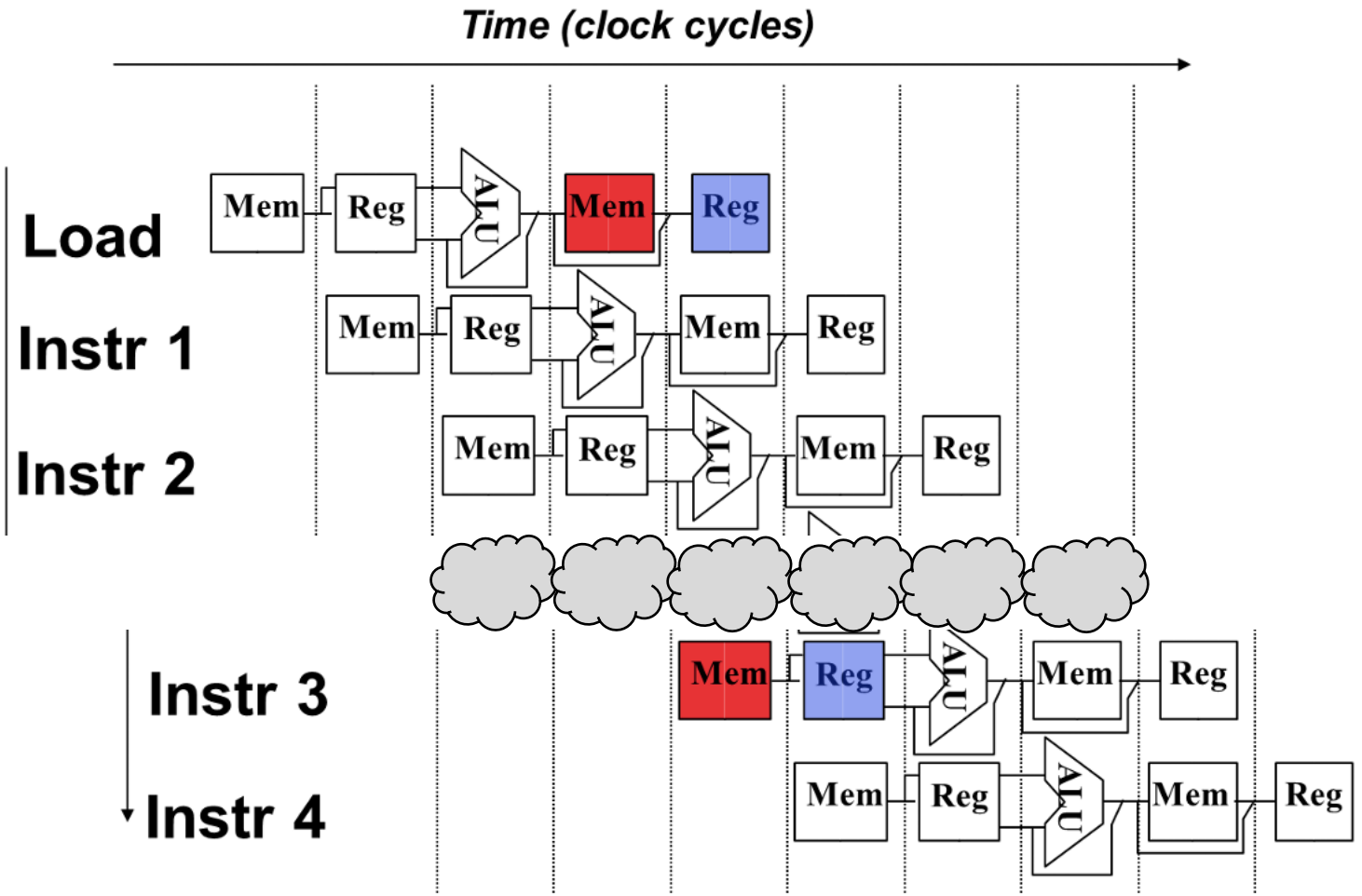
结构冒险

- 原因：已进入流水线的不同指令在同一时刻访问相同的硬件资源



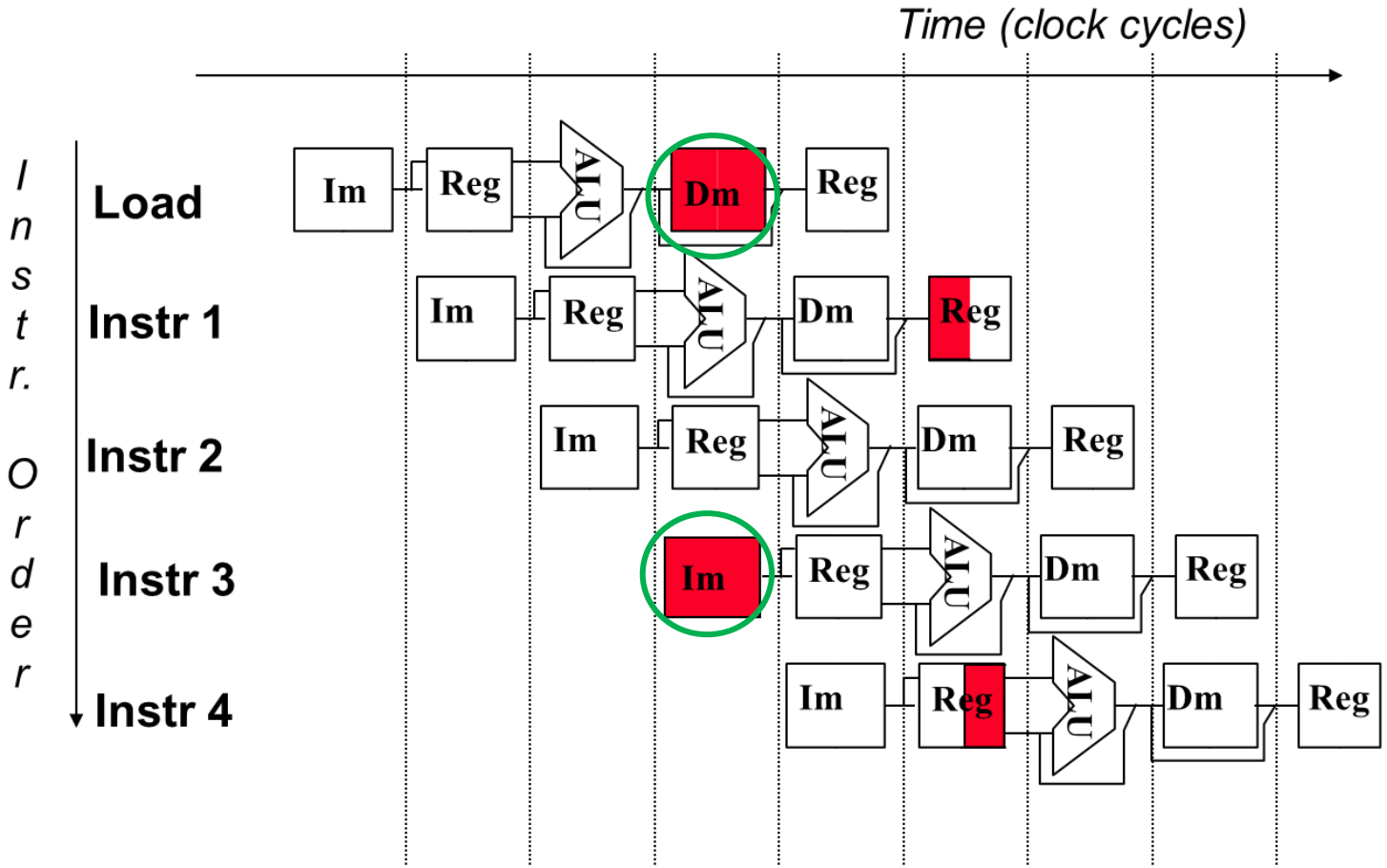
结构冒险

- 解决方案1：流水线停顿（stall），插入空泡（bubble）



结构冒险

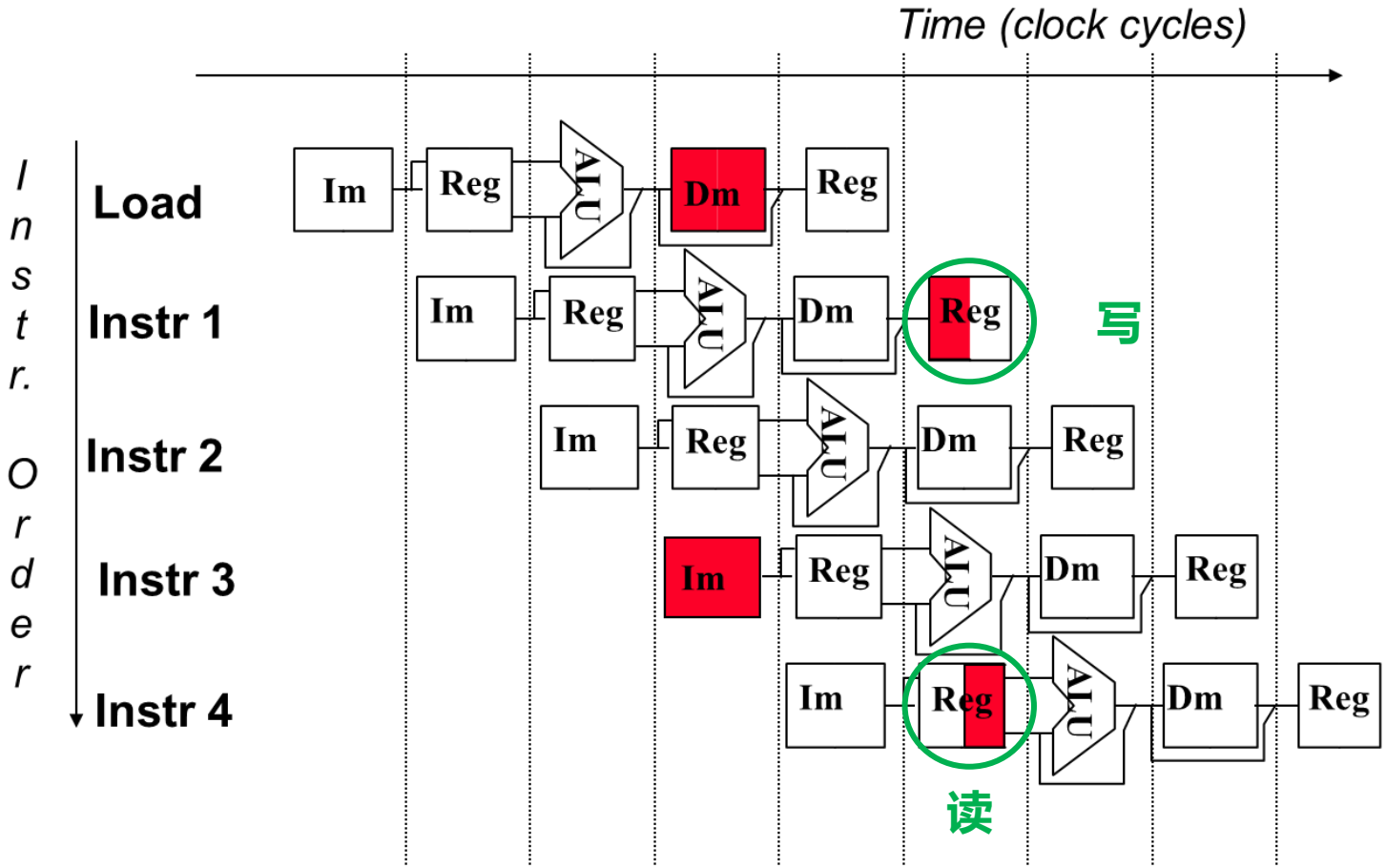
- **解决方案2：使用不同用途的多个存储器**



例如：指令和数据放在不同的Cache中

结构冒险

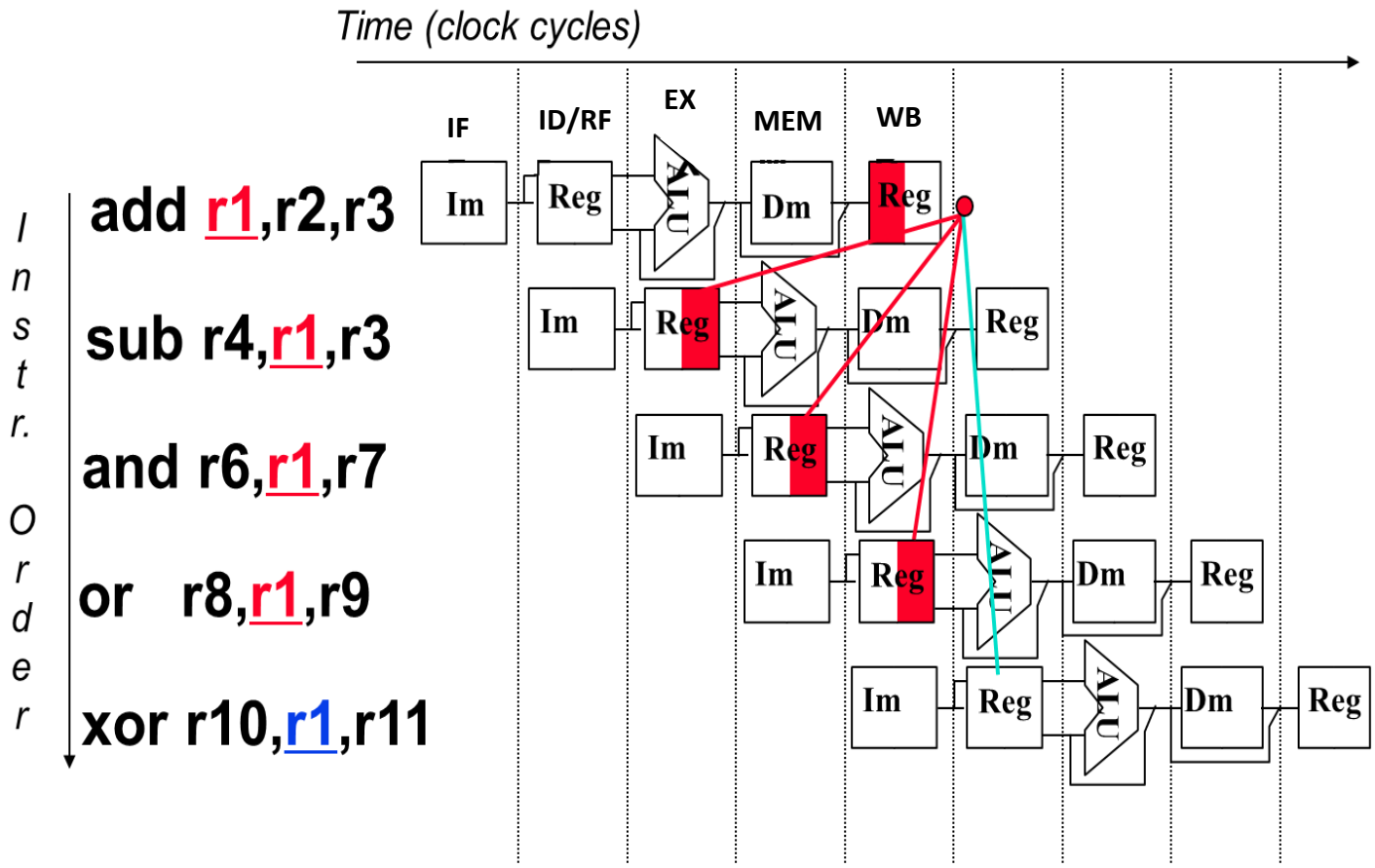
- **解决方案3：同一个存储器提供分时处理**



例如：寄存器时钟上升沿写，时钟下降沿读

数据冒险

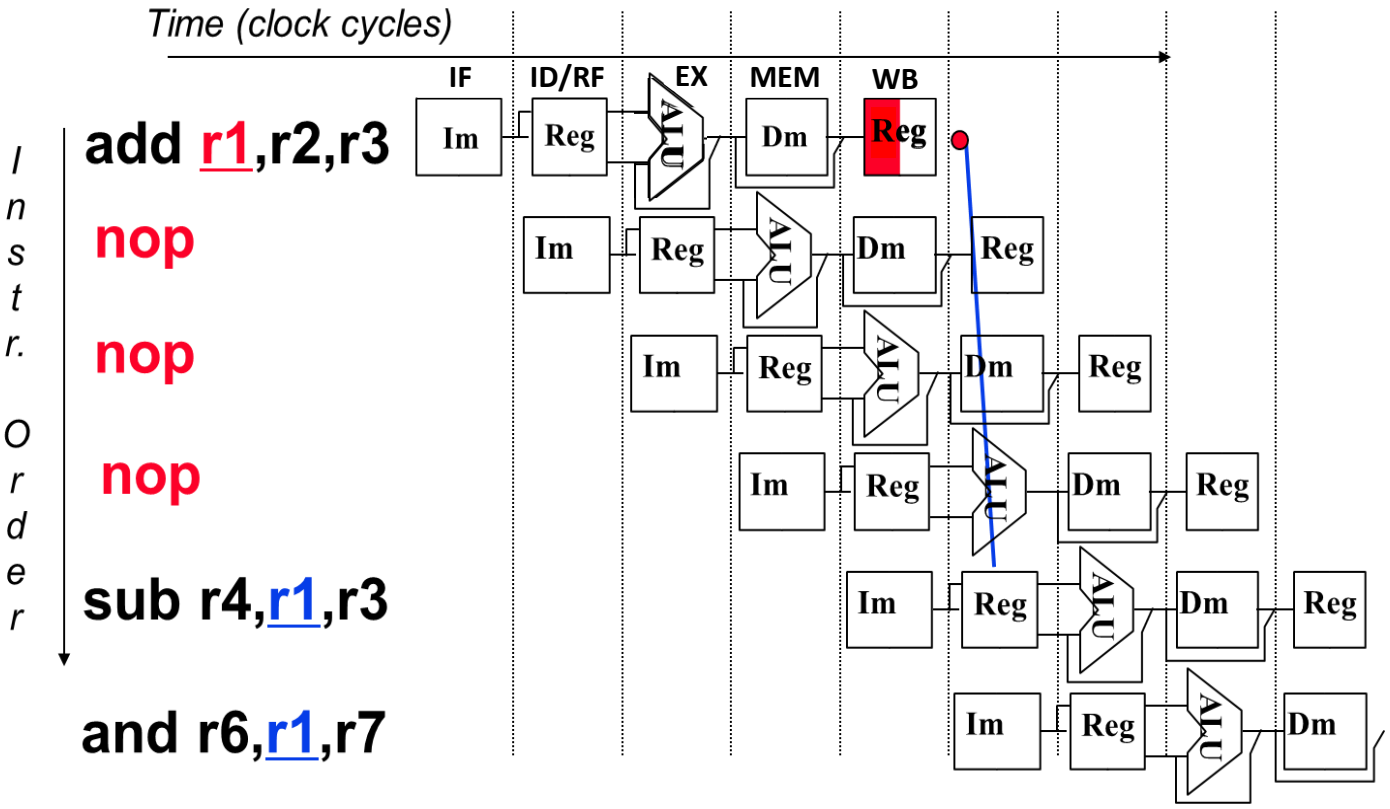
- **原因：未生成指令所需要的数据**



例如：一条指令需要使用之前指令的运算结果，但是结果还没有写回

数据冒险（续）

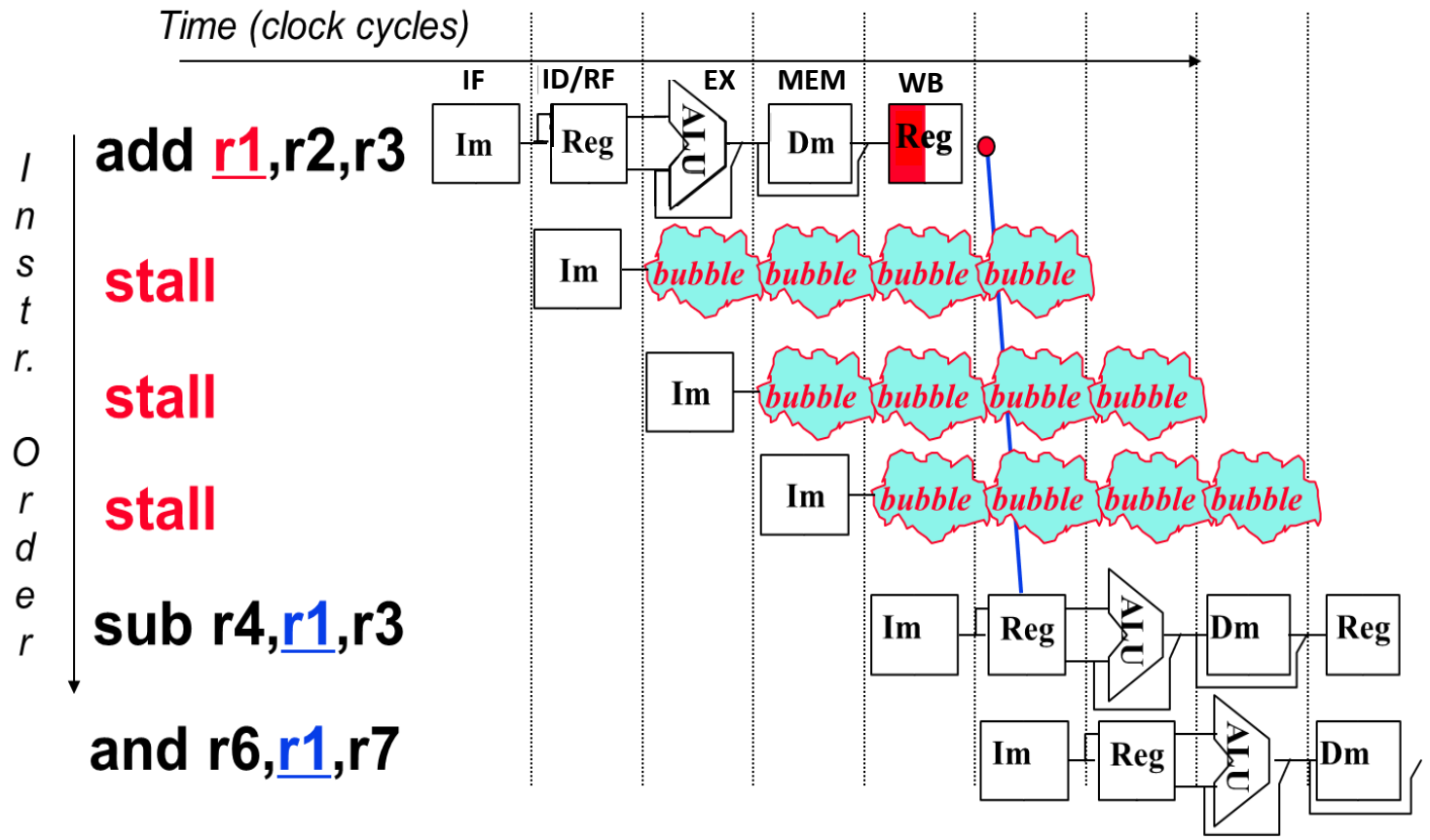
- **解决方案1：插入nop指令（软件）**



问题：如果计算机的结构发生了变化，需要更新软件

数据冒险（续）

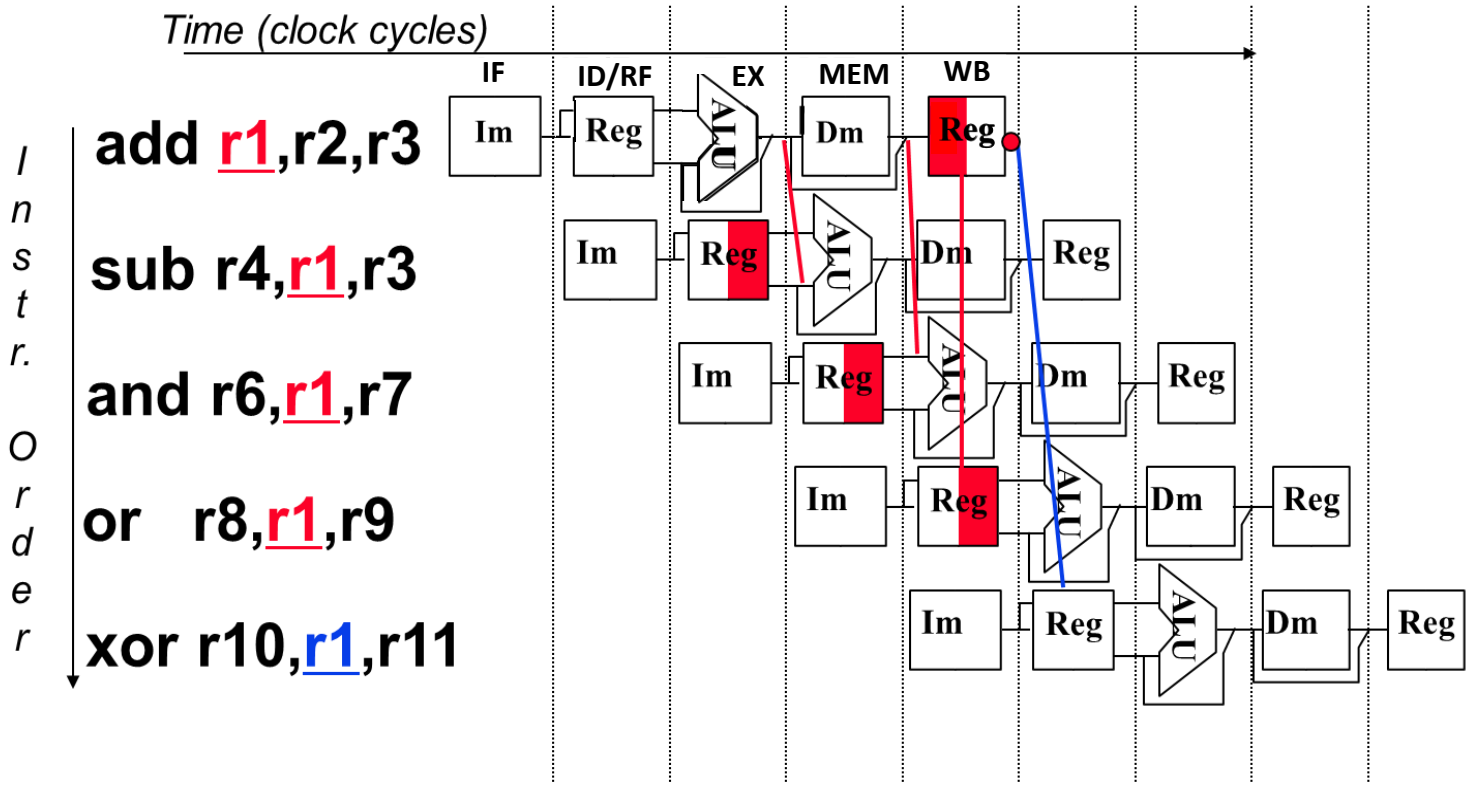
- **解决方案2：插入 bubble（硬件）**



问题：效率低下，流水线失去意义

数据冒险（续）

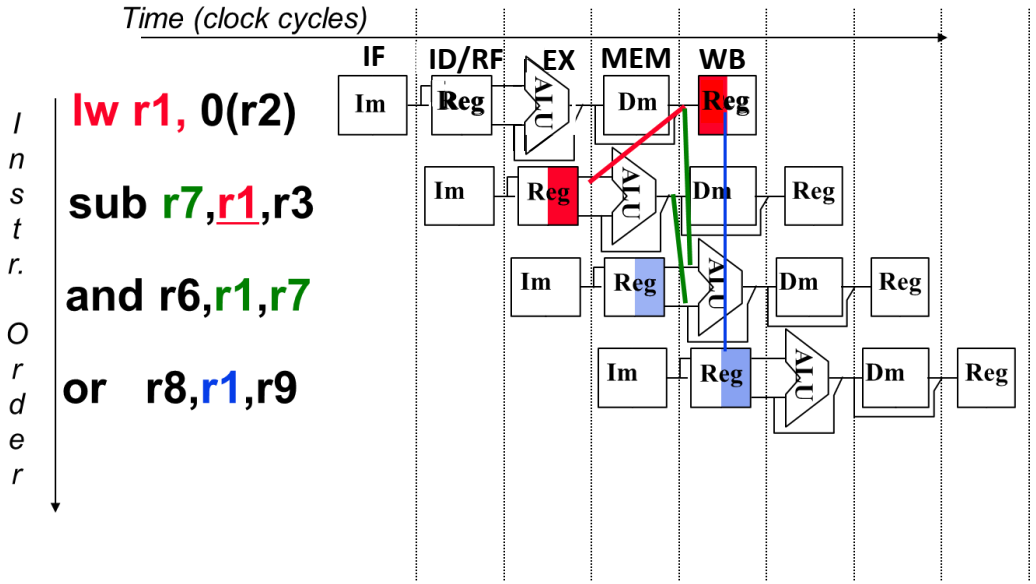
- 解决方案3：前递（forwarding） / 旁路（bypassing）



无法解决：一条指令需要使用之前指令的访存结果(Load-Use Harzard)

数据冒险（续）

- **解决方案4：交换指令顺序**



Slow code:

```
lw    $2, b
lw    $3, c
add   $1, $2, $3
sw    a, $1
lw    $5, e
lw    $6, f
sub   $4, $5, $6
sw    d, $4
```

Fast code:

```
lw    $2, b
lw    $3, c
lw    $5, e
add   $1, $2, $3
lw    $6, f
sw    a, $1
sub   $4, $5, $6
sw    d, $4
```

控制冒险

- **原因：指令的执行顺序被更改**
  - 转移（Transfer）：分支（branch），循环（loop），…
  - 中断（Interrupt）
  - 异常（Exception）
  - 调用 / 返回（Call / return）

- **影响：**
  - 转移指令占比15%～25%（平均每隔4～7条指令）
  - 转移平均损失10个周期
  - 流水线越深，超标量数越多，转移指令的影响越大

控制冒险（续）

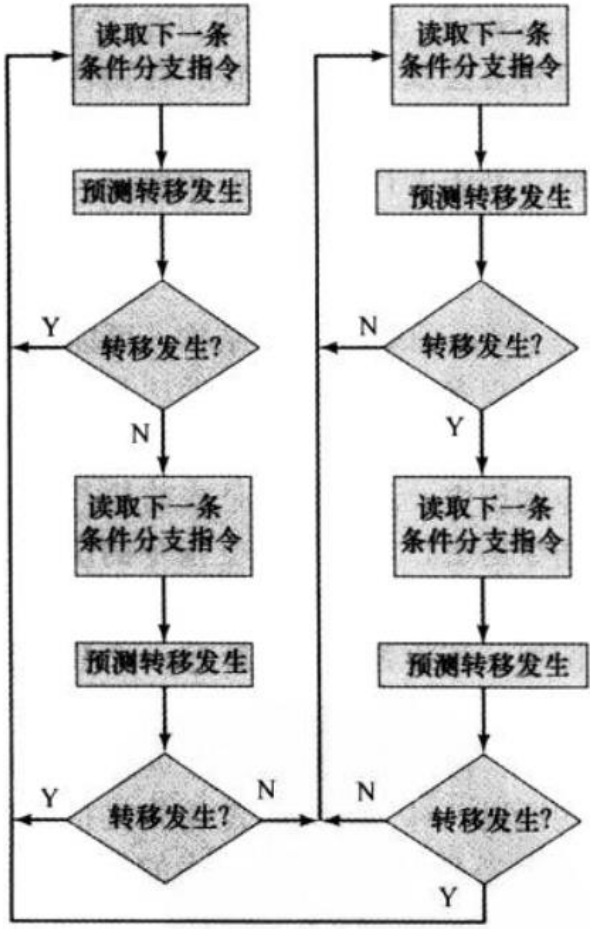
- **解决方案1：取多条指令**
  - **多个指令流：**复制流水线的开始部分，并允许流水线同时取这两条指令，使用两个指令流
  - **预取分支目标：**识别出一个条件分支指令时，除了取此分支指令之后的指令外，分支目标处的指令也被取来
  - **循环缓冲器：**由流水线指令取指阶段维护的一个小的但极高速的存储器，含有 n 条最近顺序取来的指令

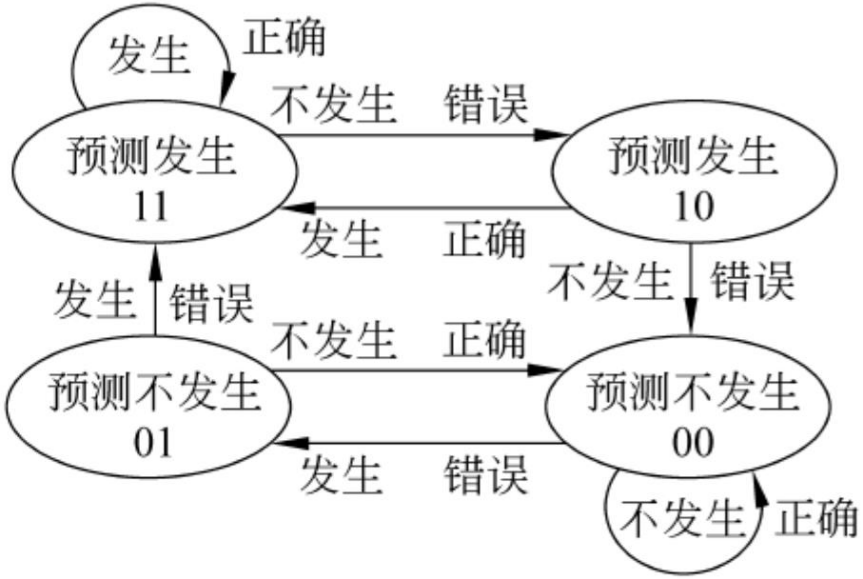
控制冒险（续）

- **解决方案2：分支预测**
  - 静态预测（规则不变）
    - 预测绝不发生跳转 → 按顺序取下一条指令
    - 预测总是发生跳转 → 到目标地址取指令
    - 依操作码预测
  - 动态预测（规则变化）
    - 发生 / 不发生切换
    - 转移历史表

控制冒险（续）

- **动态分支预测：发生 / 不发生切换**



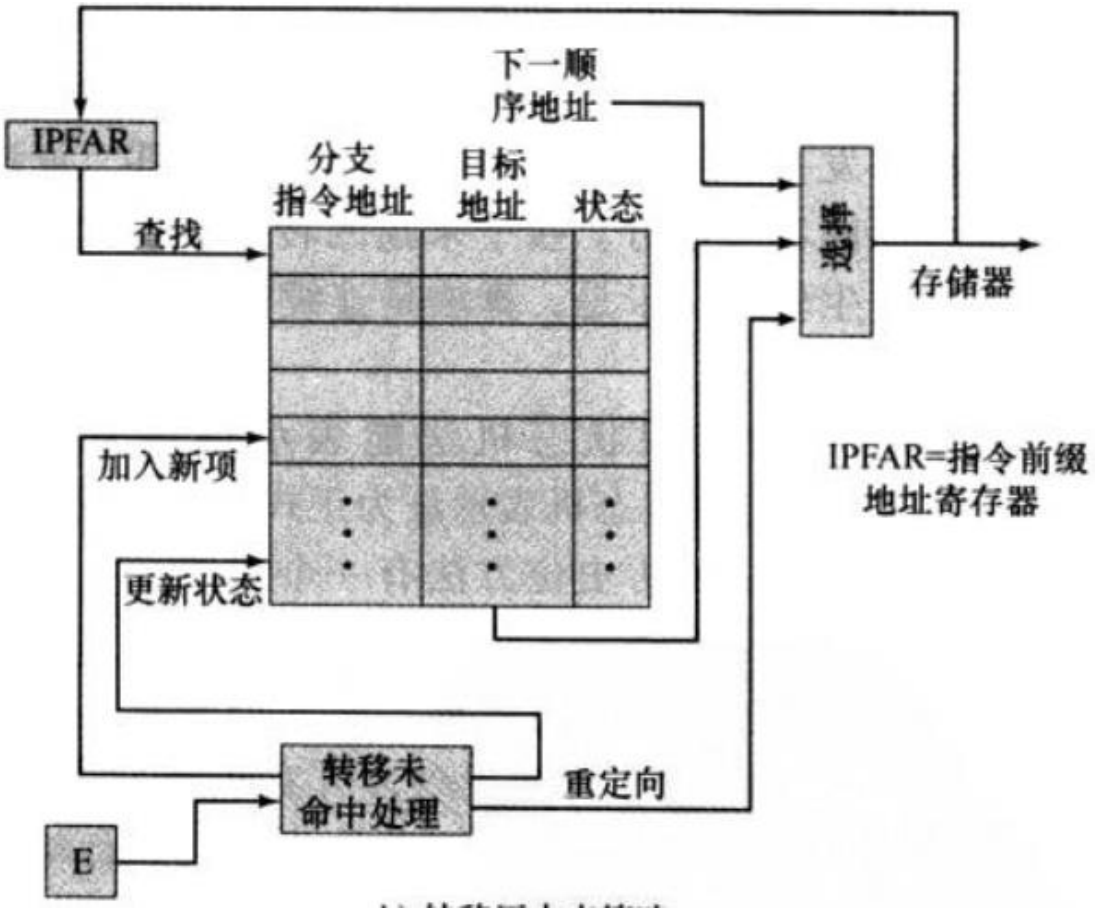


仅在连续发生两次错误时改变状态

```
for (i=0, i<10, i++) {
    for (j=0, j<10, j++) {
    }
}
```

控制冒险（续）

- **动态分支预测：转移历史表**



控制冒险（续）

- **解决方案3：提前判断**
  - 直接无条件转移：例如 j Target
    - 在取指阶段即可获得转移目标地址（流水线不停顿）
  - 间接无条件转移：例如 jr $r1
    - 在译码阶段才能获得转移目标地址（流水线停顿1周期）
  - 直接有条件转移：例如 beq $r1, $r2, Target
    - 在执行阶段才能获得转移目标地址（流水线停顿2周期）
    - 在寄存器堆输出端增加额外的比较电路（流水线停顿1周期）

控制冒险（续）

- 解决方案4：交换指令顺序

```
xor     $r1, $r2, $r3
addi    $r4, $r5, 1
subi    $r6, $r7, 2
beq     $r4, $r6, Next
slt     $r5, $r7, 10
…


Next…
```

```
addi    $r4, $r5, 1
subi    $r6, $r7, 2
beq     $r4, $r6, Next
xor     $r1, $r2, $r3
slt     $r5, $r7, 10
…


Next…
```

总结

- **指令周期**
  - 取指周期，执行周期，中断周期，间址周期
  - 数据流
- **流水线**
  - 两阶段，六阶段
  - 超流水线
  - 流水线性能
  - 超标量流水线
  - 冒险：结构冒险，数据冒险，控制冒险

谢谢

bohanliu@nju.edu.cn

南京大學
NANJING UNIVERSITY