国防科技大学计算机学院

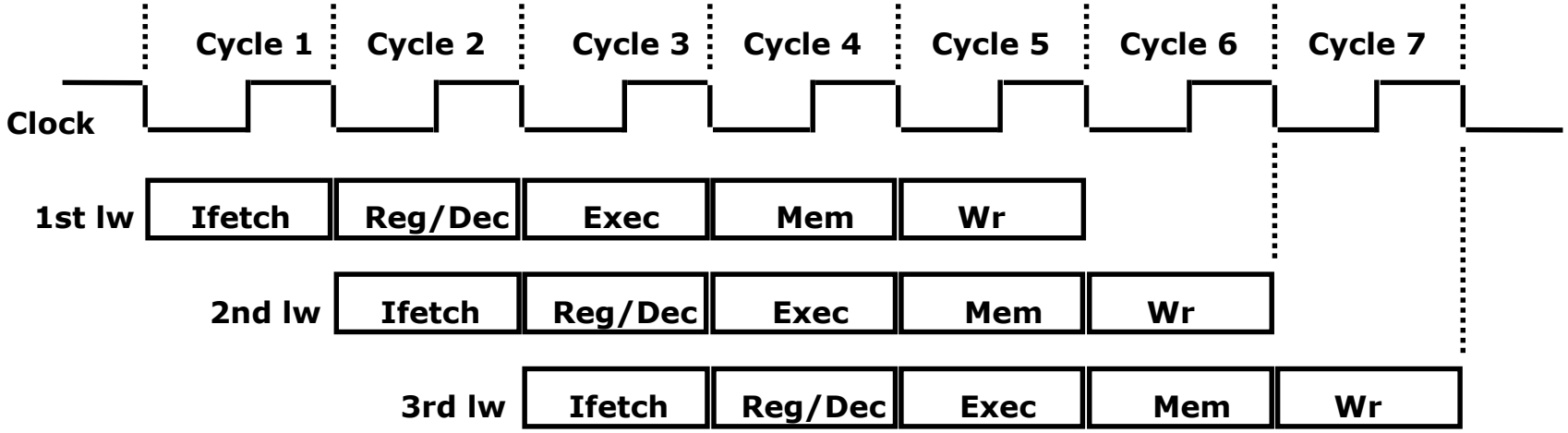
计算机原理

COMPUTER PRINCIPLE

第四章 第四节（2）流水线冒险

1. 流水线数据通路

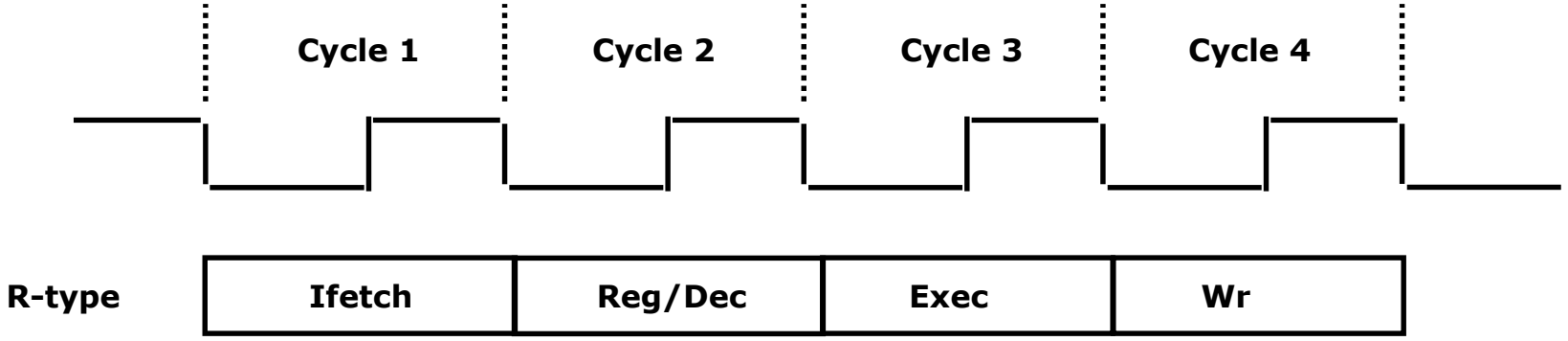
❑ Load指令的流水线



- 每个时钟周期有五个功能部件同时在工作
- 后面指令在前面完成取指后马上开始
- 每个load指令仍然需要五个时钟周期完成
- 但是吞吐率(throughput)提高许多，理想情况下
  - 每个时钟周期有一条指令进入流水线
  - 每个时钟周期都有一条指令完成
  - 每条指令的时钟周期(CPI)为1

1. 流水线数据通路

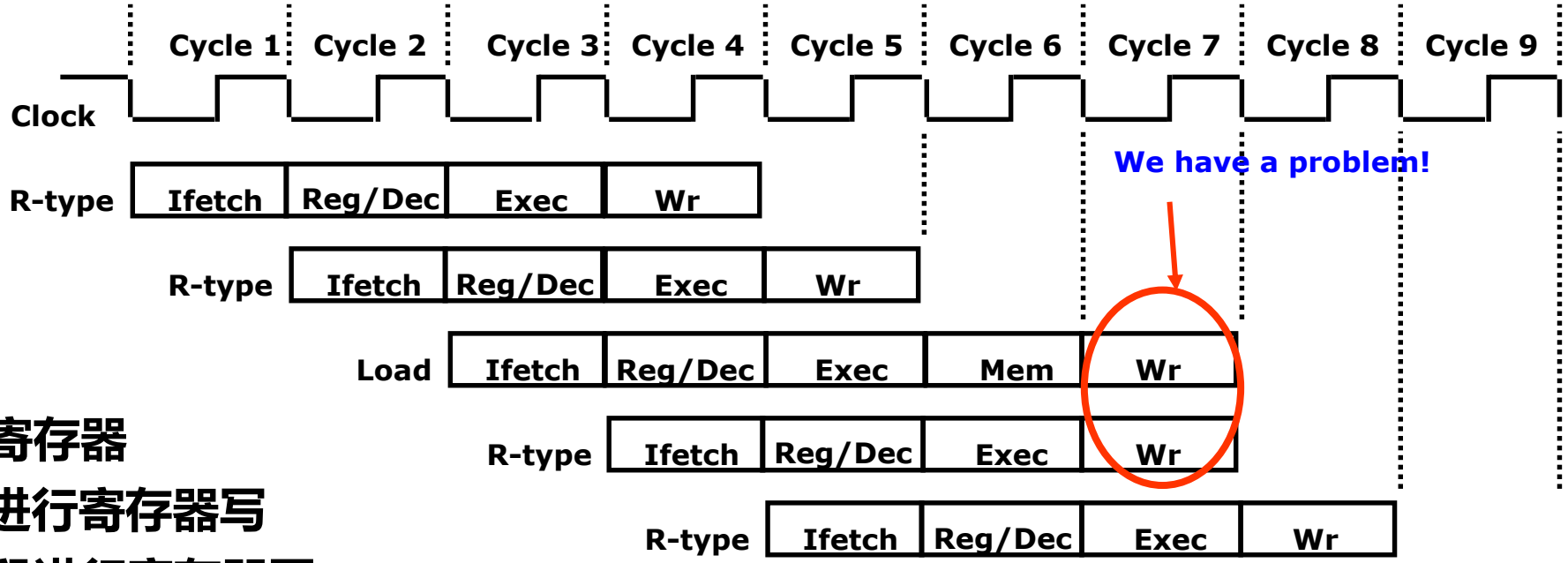
❑ R-型指令的流水线



- ❑ **Ifetch:** 取指令并计算PC+4
- ❑ **Reg/Dec:** 从寄存器取数，同时指令在译码器进行译码
- ❑ **Exec:** 在ALU中对操作数进行计算
- ❑ **Wr:** ALU计算的结果写到寄存器

1. 流水线数据通路

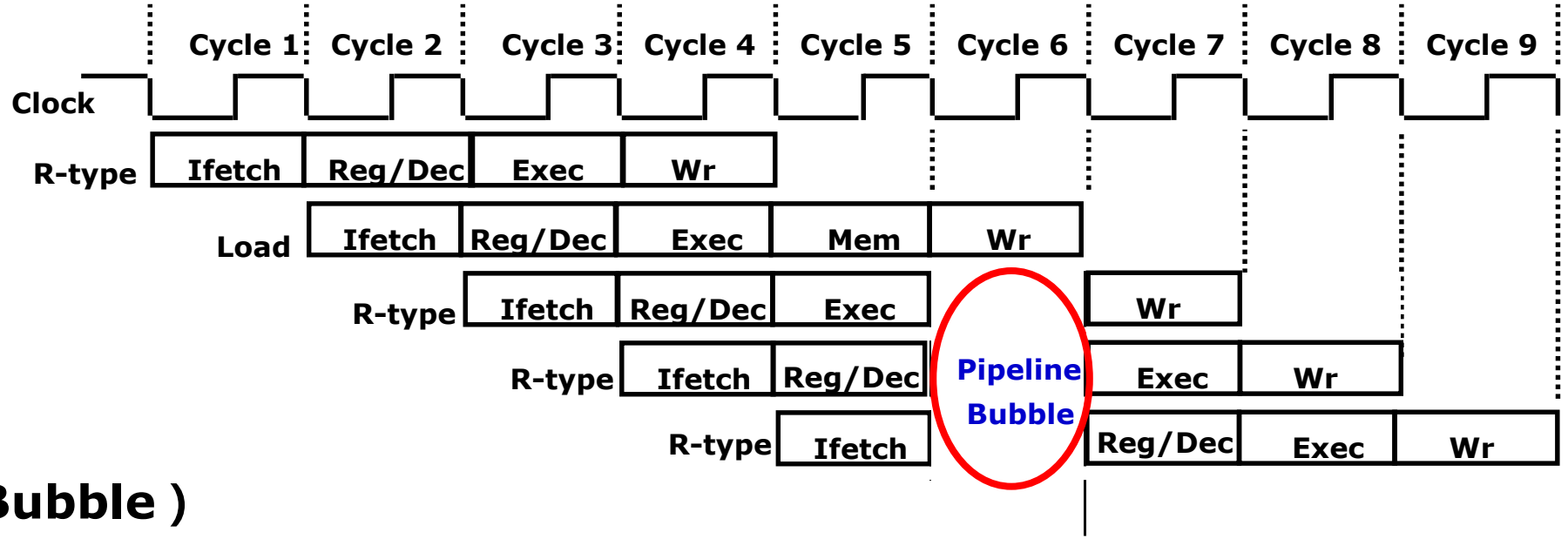
❑ 含有R-型和Load指令的流水线



- ❑ 上述流水线有个问题:
  两条指令试图同时写寄存器
  - ■ Load在第5阶段进行寄存器写
  - ■ R-type在第4阶段进行寄存器写
- ❑ 一个功能部件同时被多条指令使用的现象，称为结构冒险(Struture Hazard)，或称为资源冲突
- ❑ 为了使流水线能顺利工作，规定：
  - ■ 每个功能部件每条指令只能用一次(如不能两次或两次以上使用寄存器文件写端口)
  - ■ 每个功能部件必须在相同阶段被使用(如总是在第五阶段使用寄存器文件写端口)

1. 流水线数据通路

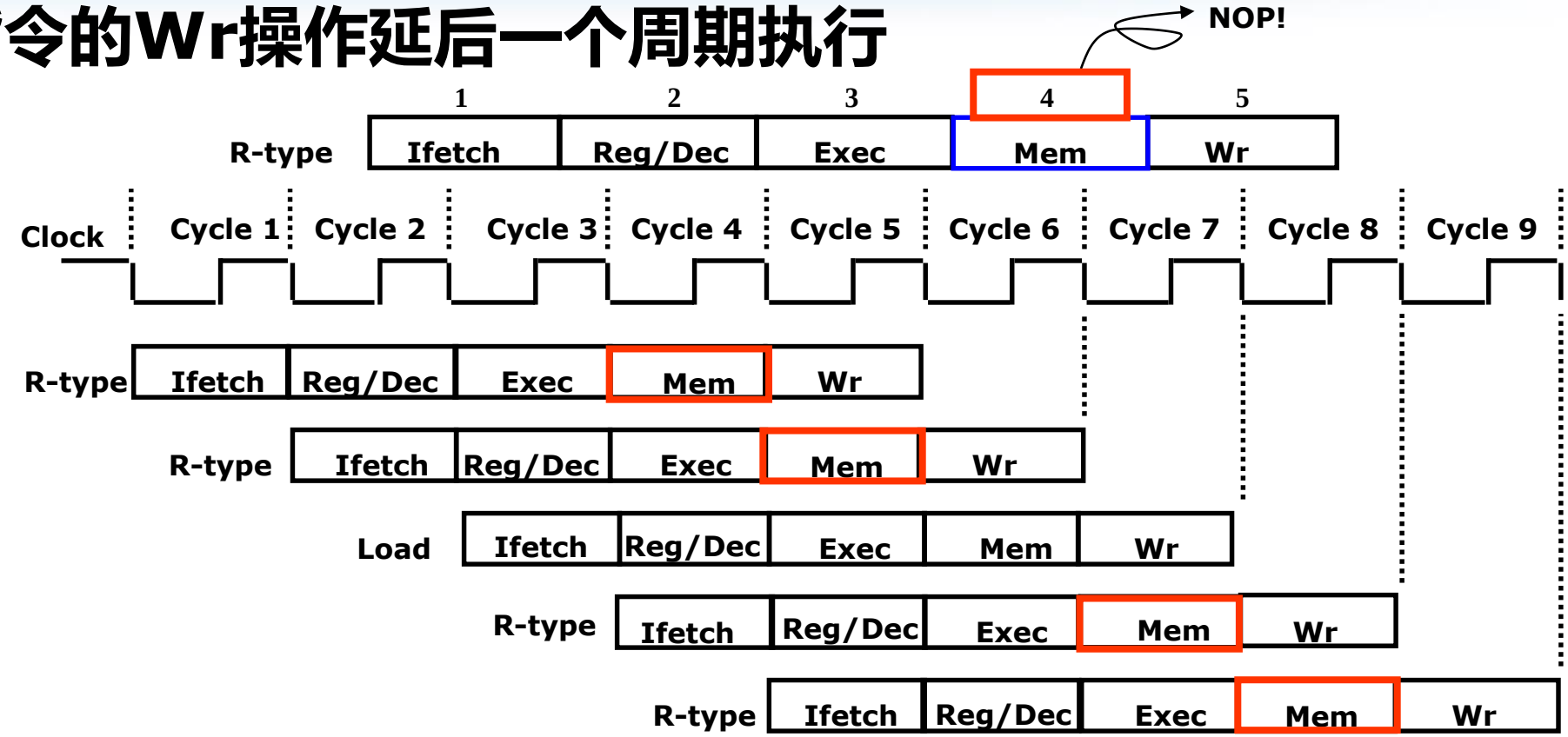
❑ 方法1：在流水线中插入气泡（bubble）



- ❑ 插入一个气泡（Bubble）
  - ■ 抵达/通过某个流水段的指令正常执行，其余指令在各自当前的流水段等待一个周期。
- ❑ 插入“Bubble”到流水线中，以禁止同一周期有两次写寄存器。缺点：
  - ■ 控制逻辑复杂
  - ■ 第5周期没有指令被完成（影响了CPI）

1. 流水线数据通路

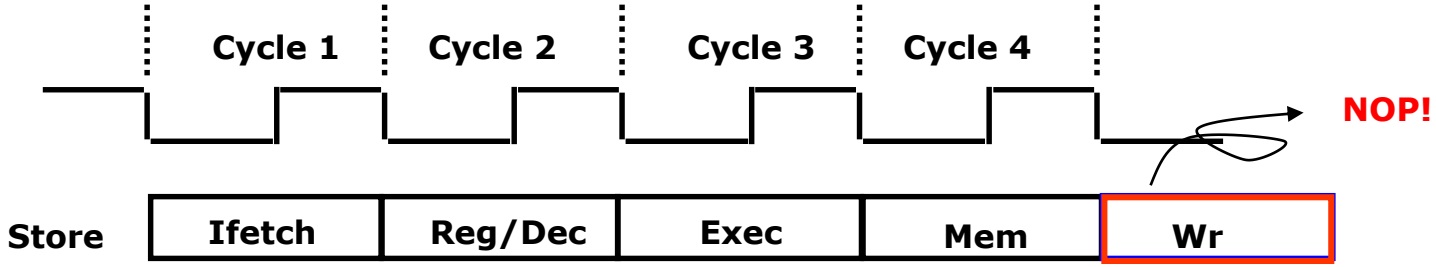
- 方法2：R-型指令的Wr操作延后一个周期执行



- 加一个“NOP”阶段以延迟“写”操作:
  - 把“写”操作安排在第5阶段, 这样使R-Type的Mem阶段为空NOP

1. 流水线数据通路

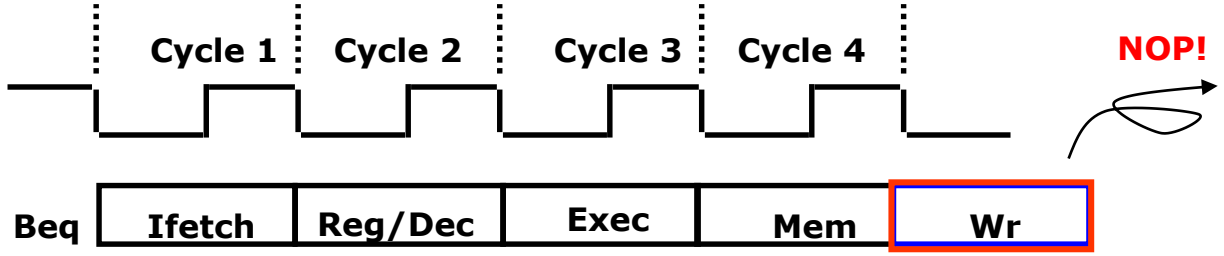
- Store指令的五个阶段



- Ifetch：取指令并计算PC+4
- Reg/Dec： 从寄存器取数，同时指令在译码器进行译码
- Exec：16位立即数符号扩展后与寄存器值相加，计算主存地址
- Mem：将寄存器读出的数据写到主存
- Wr: 加一个空的写阶段，使流水线更规整！

1. 流水线数据通路

❑ 条件分支指令（Beq）的五个阶段

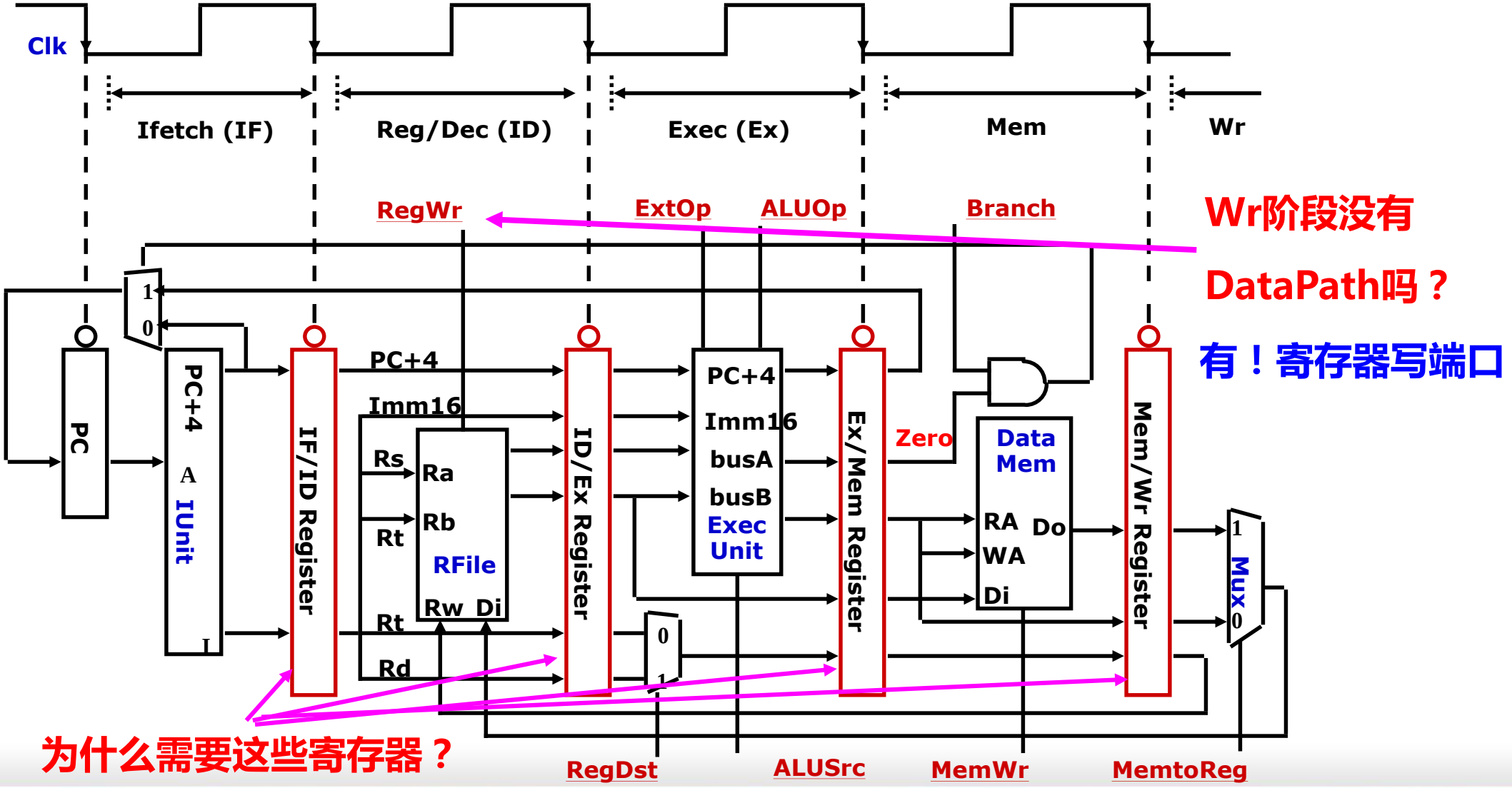


- ❑ **Ifetch:** 取指令并计算PC+4
- ❑ **Reg/Dec:**从寄存器取数，同时指令在译码器进行译码
- ❑ **Exec:** 执行阶段
  - ■ ALU中比较两个寄存器大小(做减法)
  - ■ Adder中计算转移地址
- ❑ **Mem:** 如果比较相等，则：
  - ■ 转移目标地址写到PC
- ❑ **Wr:** 加一个空的写阶段，使流水线更规整！

所有指令都按照最复杂的“load”指令所需的五个阶段来划分，不需要的阶段加一个“NOP”操作

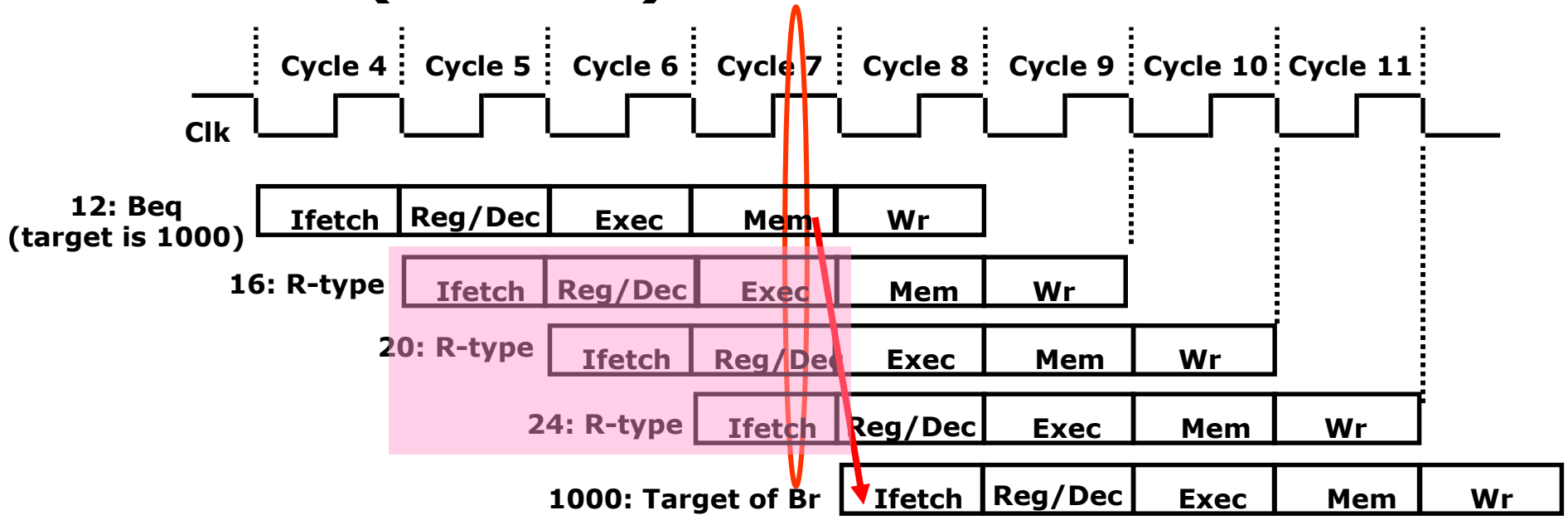
1. 流水线数据通路

❑ 五段流水线的数据通路



2. 流水线冒险

❑转移分支指令(Branch)引起的“延迟”现象

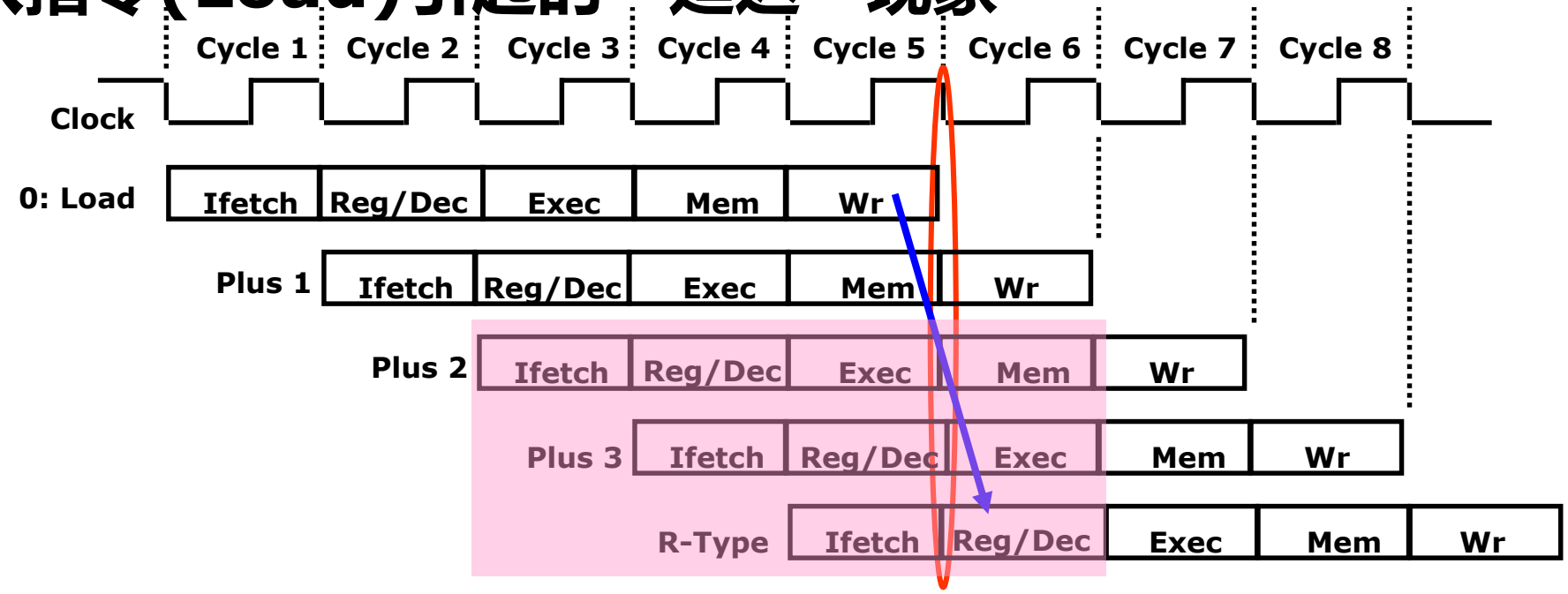


- ❑ 虽然Beq指令在第四周期取出，但：
  - ■ 目标地址在第七周期才被送到PC的输入端
  - ■ 第八周期才能取出目标地址处的指令执行

  结果：在取目标指令之前，已有三条指令被取出，取错三条指令！
- ❑ 这种现象称为控制冒险(Control Hazard)，亦称为分支冒险或转移冒险(Branch Hazard)

2. 流水线冒险

❑装入指令(Load)引起的“延迟”现象



- ❑ 尽管Load指令在第一周期就被取出，但：
  - ■ 数据在第五周期结束才被写入寄存器
  - ■ 在第六周期时，写入的数据才能被用

  结果：在Load指令结果有效前，已经有三条指令被取出(若它们要用到Load数据，需要延迟！)
- ❑ 这种现象称为 数据冒险 (Data Hazard)，亦称为数据相关(Data Dependency)

3. 流水线的三种冒险

- 冒险（Hazard）：指流水线遇到无法正确执行后续指令或执行了不该执行的指令的现象。
  - Structural hazards(resource conflicts)

    现象：同一个部件同时被不同指令所使用
  - Data hazards(data dependencies)

    现象：后面指令用到前面指令结果，但结果还没产生
  - Branch hazards(changes in program flow)

    现象：转移或异常改变执行流程，顺序执行指令在目标地址产生前已被取出