计算机组成与系统结构

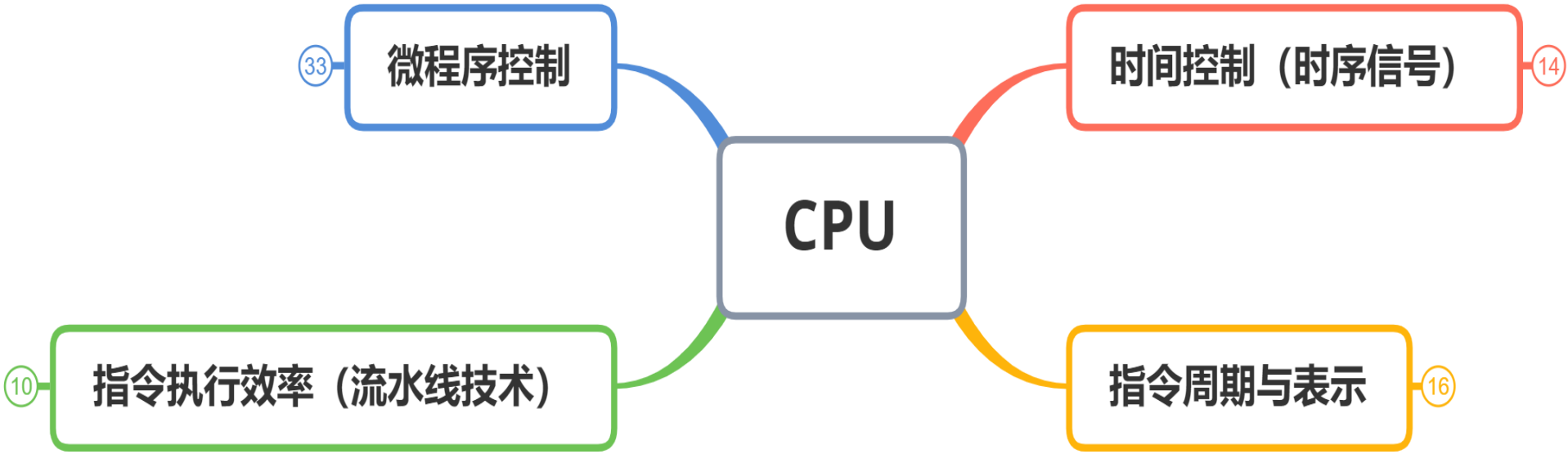
第五章 中央处理机

吕昕晨

lvxinchen@bupt.edu.cn

网络空间安全学院

中央处理机



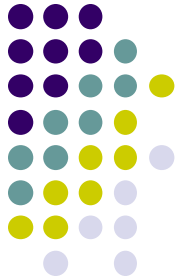
第五章 中央处理机

- 流水线技术与性能分析
  - 流水线与图形化表示
  - 流水线性能指标
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险

并行性

- 并行性概念
  - 问题中具有可以同时进行运算或操作的特性
  - 例：在相同时延的条件下，用n位运算器进行n位并行运算速度几乎是一位运算器进行n位串行运算的n倍（狭义）
- （广义）含义
  - 只要在同一时刻（同时性）或在同一时间间隔内（并发性）完成两种或两种以上性质相同或不同的工作，他们在时间上相互重叠，都体现了并行性

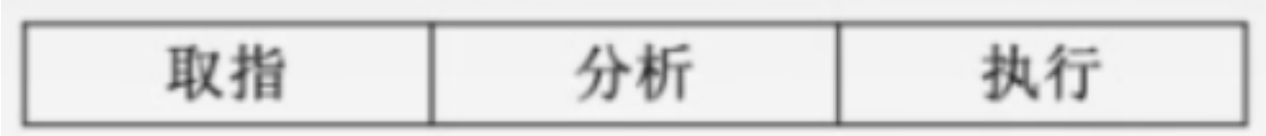
三种并行处理方式



- 三种形式
  - 时间并行（重叠）
    - 流水线技术（浮点运算流水线、交叉体存储器）
    - 让多个处理过程在时间上相互错开，轮流使用同一套硬件设备的各个部件，以加快硬件周转而赢得速度，实现方式就是采用流水处理部件
  - 空间并行（资源重复）：增加冗余部件
    - 超标量技术
    - 体现同时性，VLSI为其提供了技术保证
  - 时间+空间并行
    - 超标量流水线技术

指令执行设计分析实例

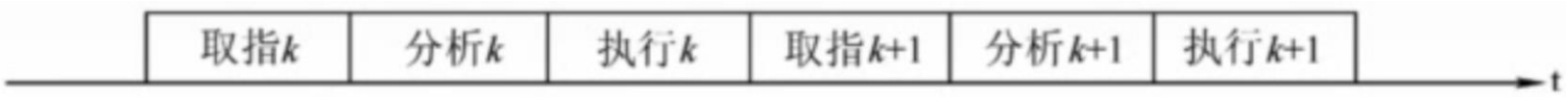
- 指令执行可分为不同阶段，考虑如下三阶段划分



- 注：阶段划分根据计算机系统不一，根据题目分析
- 设三个阶段所需时间相等，均为T
- 问题：
  - 如何**设计指令执行方式**，尽可能在不增加冗余部件情况下**提升指令执行效率**

指令执行设计分析1—顺序方式

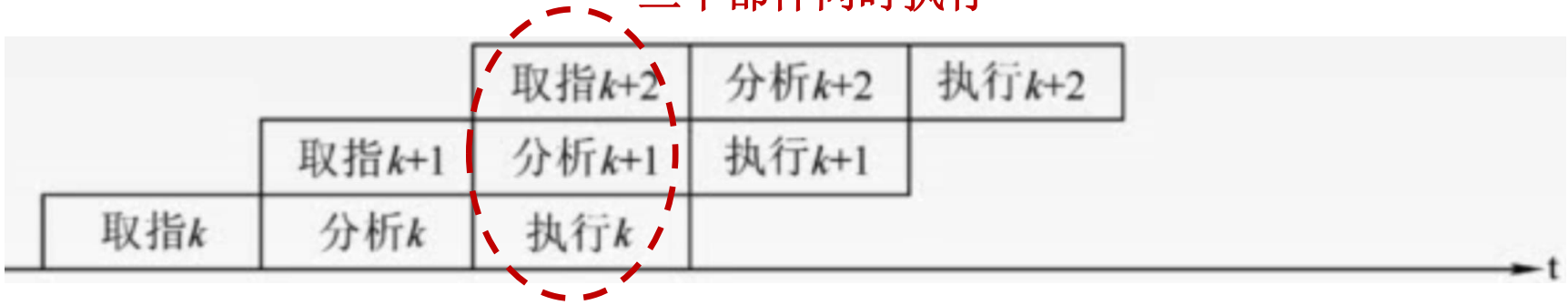
- 顺序执行方式



- 传统冯诺依曼采用执行方式，又称串行方式
- 总耗时：$\mathbf{N \times 3T}$
- 优点：控制简单，硬件容易实现
- 缺点：
  - 任意时刻，仅一条指令执行
  - 任意时刻，总有两个部件处于空闲

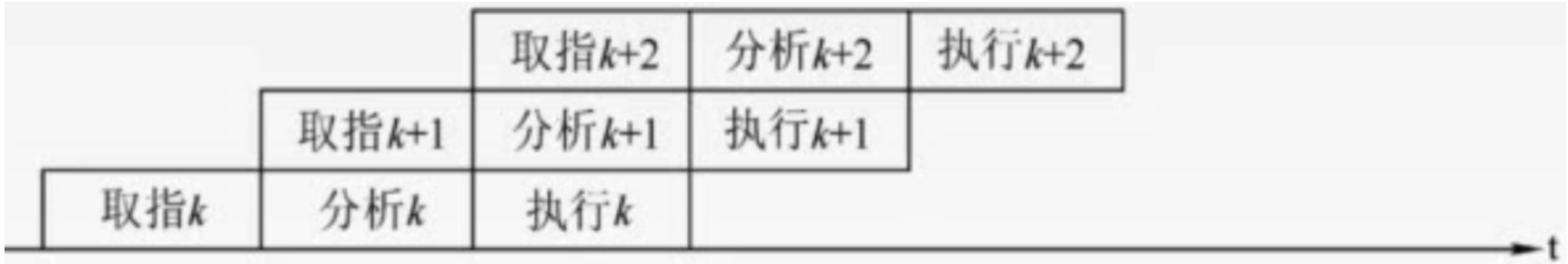
指令执行设计分析2—并行方式

- 并行执行方式



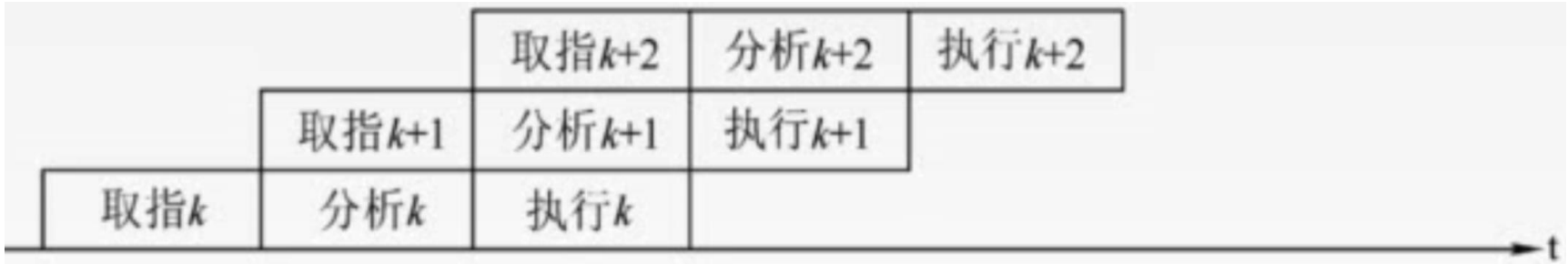
- 各部件独立，时间并行执行，即**流水线方式**
- 总耗时： $3T + (N-1)T$ （理想情况效率提升3倍）
- 优点：指令执行效率高，提升部件利用率
- 缺点：
  - 各部件是否独立？硬件如何实现？

流水线技术原理



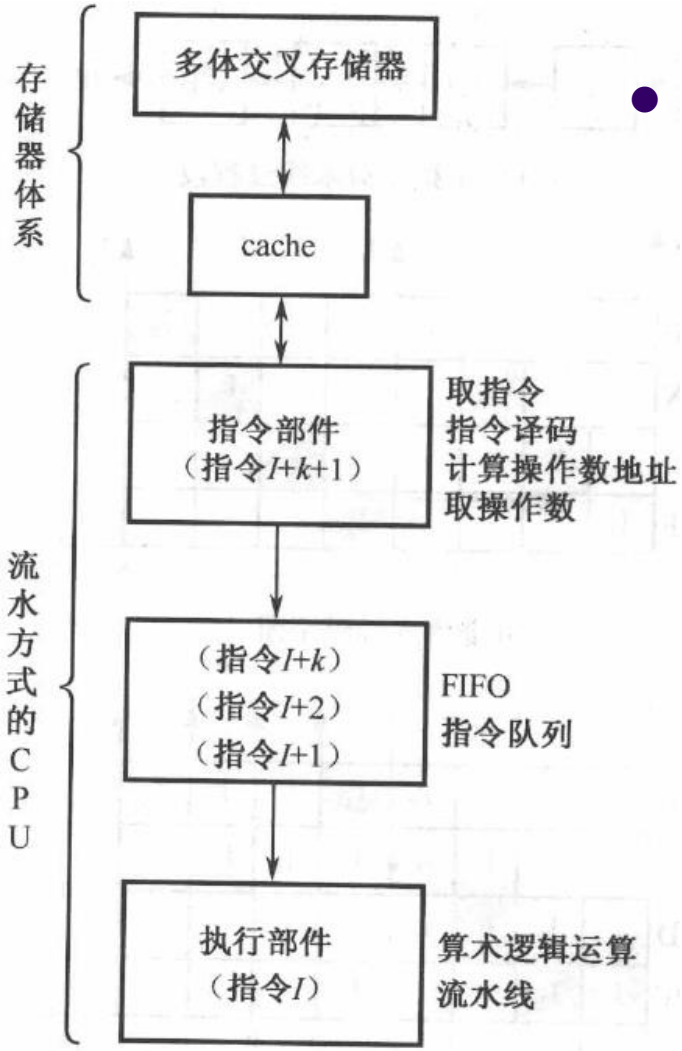
- 把一个任务分解为几个有联系的子任务。每个子任务由一个专门的功能部件实现
  - 部件：取指、分析、执行
- 在流水线中的每个功能部件之后都要有一个缓冲寄存器，或称为锁存器（各级流水独立）
- 结构：取指→缓冲→分析→缓冲→执行

流水线技术原理



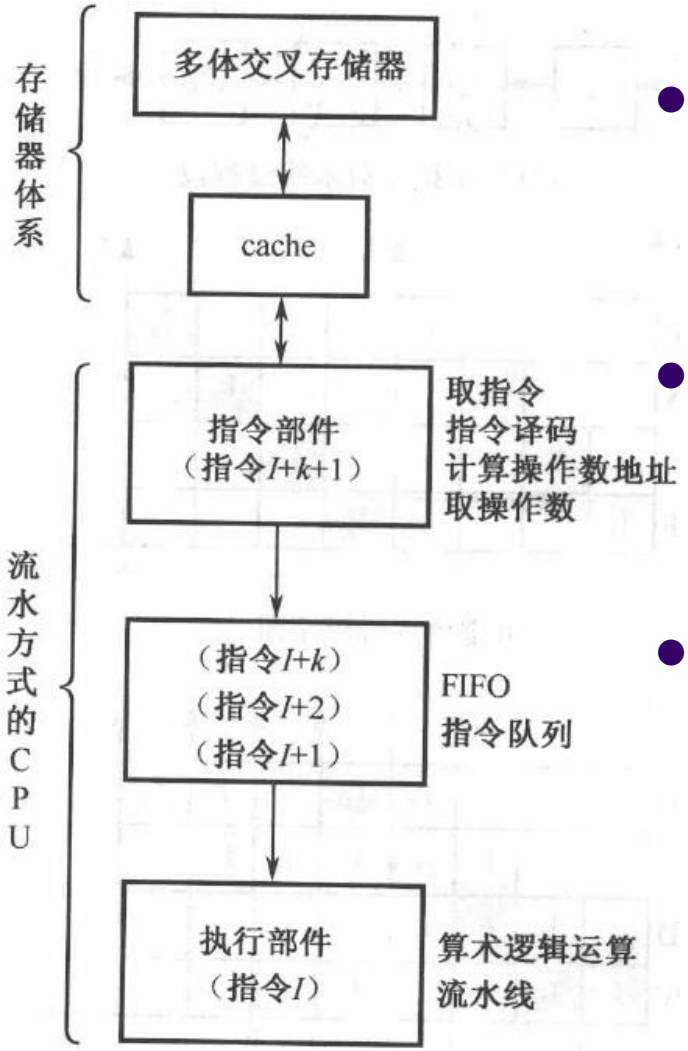
- 结构：取指→缓冲→分析→缓冲→执行
- 流水线中各段的时间应该尽量相等，否则将会引起“堵塞”和“断流”的现象
- 流水线需要有装入时间和排空时间，只有当流水线完全充满时，才能充分发挥效率
- 在流水线中必须是连续的任务，只有不断的提供任务才能充分发挥流水线的效率

流水CPU的结构（1）



- 流水计算机的系统组成
  - 存储器体系
    - 多体交叉存储器
      - 流水线技术对数据依赖性大，采用交叉体存储器
    - Cache
  - 流水方式CPU
    - 指令部件
    - 指令队列
    - 执行部件

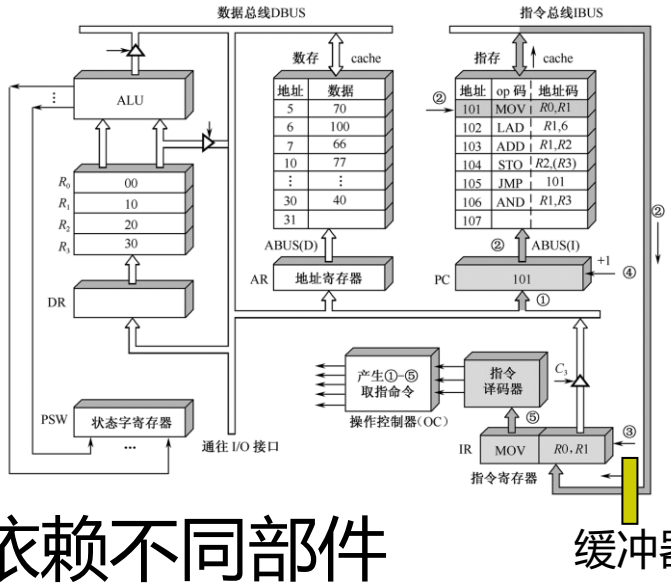
流水CPU的结构（2）

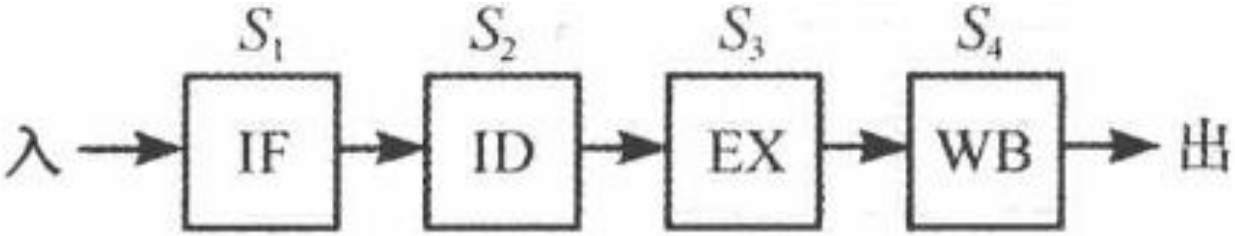


- 指令部件
  - 本身是流水线，包括取指、译码、计算操作数地址、取操作数等
- 指令队列
  - FIFO方式寄存器栈（队列）
  - 存放待执行指令
- 执行部件
  - 多个采用流水线方式构成的算术逻辑部件构成
  - 将定点运算部件和浮点运算部件分开

流水线图形化表示

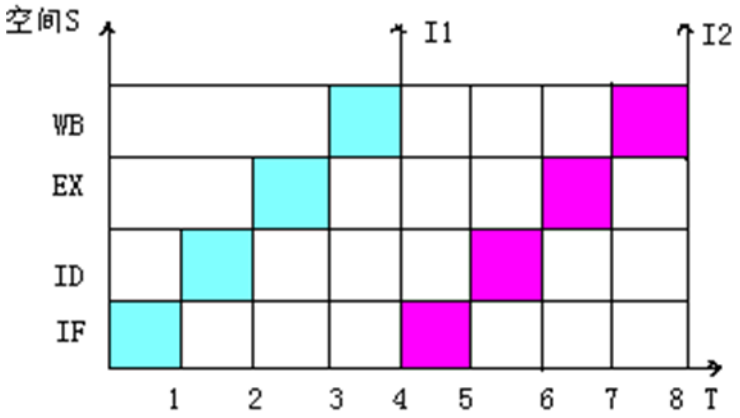
- 考虑一个四周期流水的指令划分（第一章）：
  - 取指：IF（Instruction Fetch）
  - 译码：ID（Instruction Decode）
  - 执行：EX（Execution）
  - 回写：WB（Write Back）
- 流水线基础
  - 取指、译码、执行、回写分别依赖不同部件
  - 可以并行执行（加缓冲、共享总线、Cache）



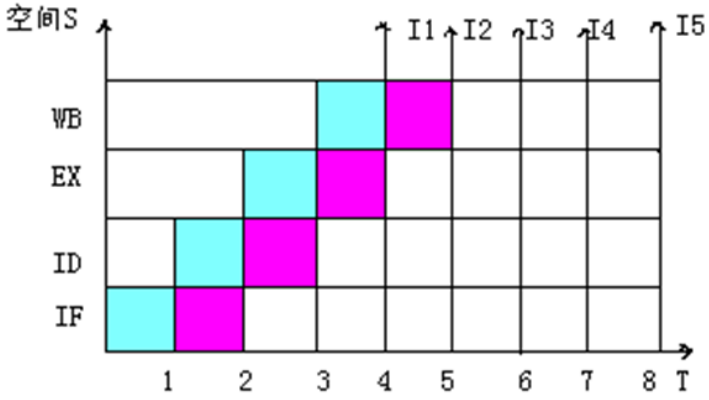


时空图——流水 v.s. 非流水

- 非流水CPU
  - 完成一个指令的四个阶段，再执行下一条指令
  - 未能充分利用各部件并行性
  - 四个周期完成一条指令

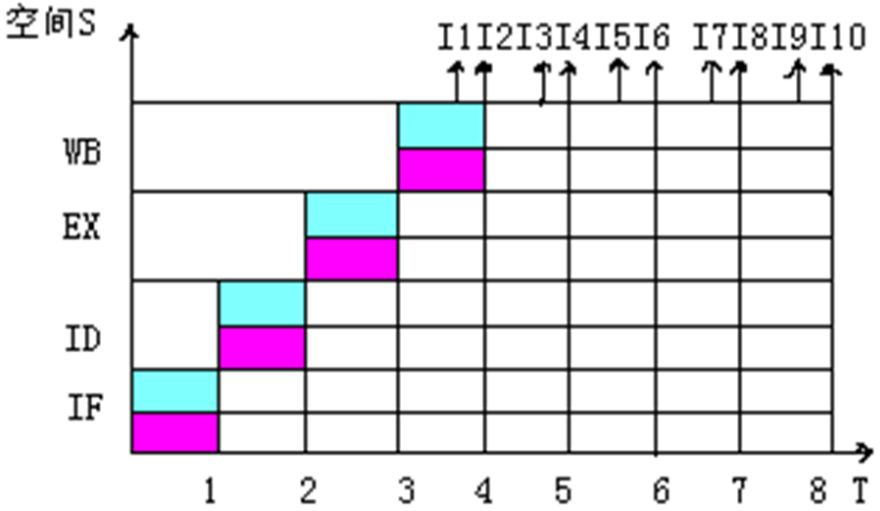


- 流水CPU
  - 四个阶段重叠执行
  - 流水线满载时，每个周期完成一条指令



时空图——超标量流水线

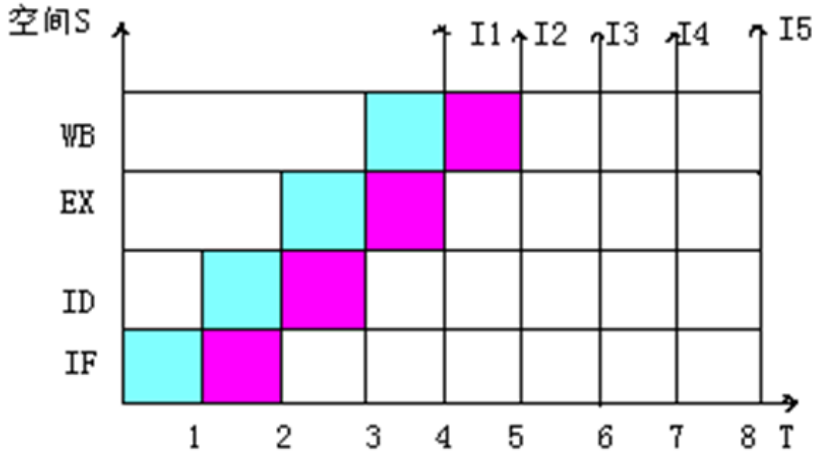
- 采用时间和空间并行技术
- 空间并行：加强各部件能力
  - 利用VLSI技术，提升各部件能力（冗余）
  - 例如，各部件同时执行两条以上指令
- 流水线满载时
  - 每一个时钟周期可以执行2条指令



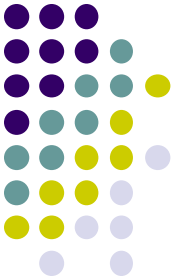
流水线图形化表示总结

- 指令执行过程图（按指令）

- **指令执行时空图（按部件）**



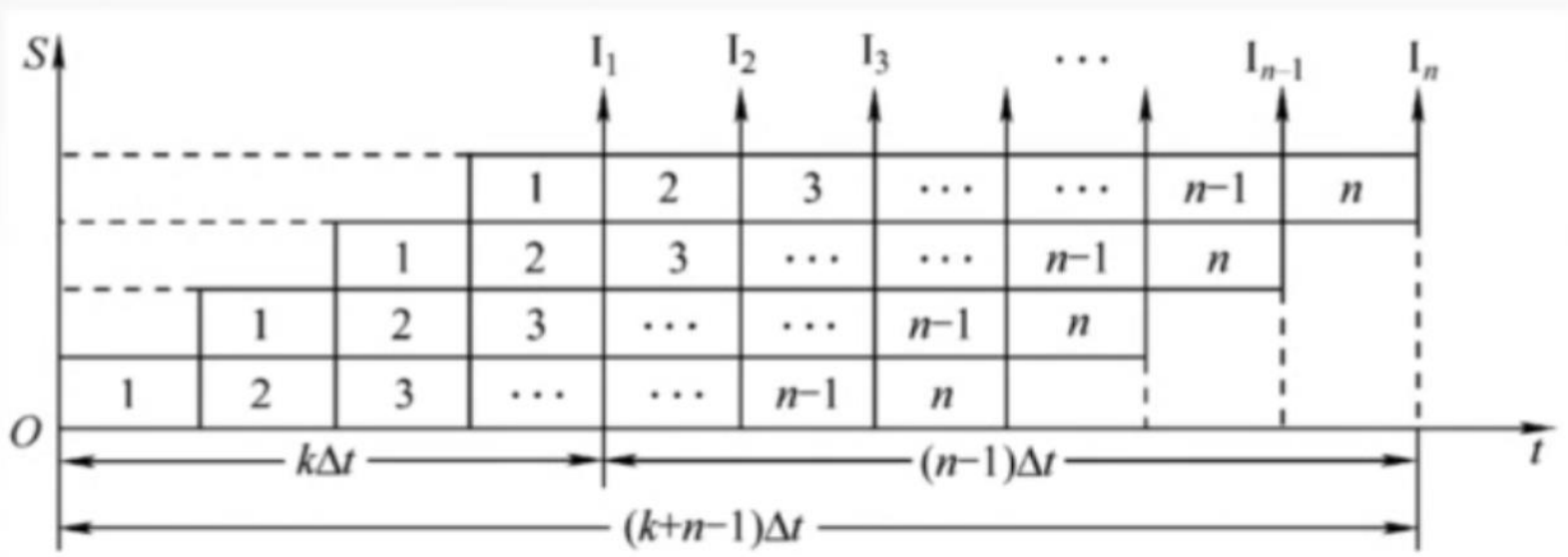
第五章 中央处理机



- 流水线技术与性能分析
  - 流水线与图形化表示
  - 流水线性能指标（吞吐率、加速比、效率）
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险

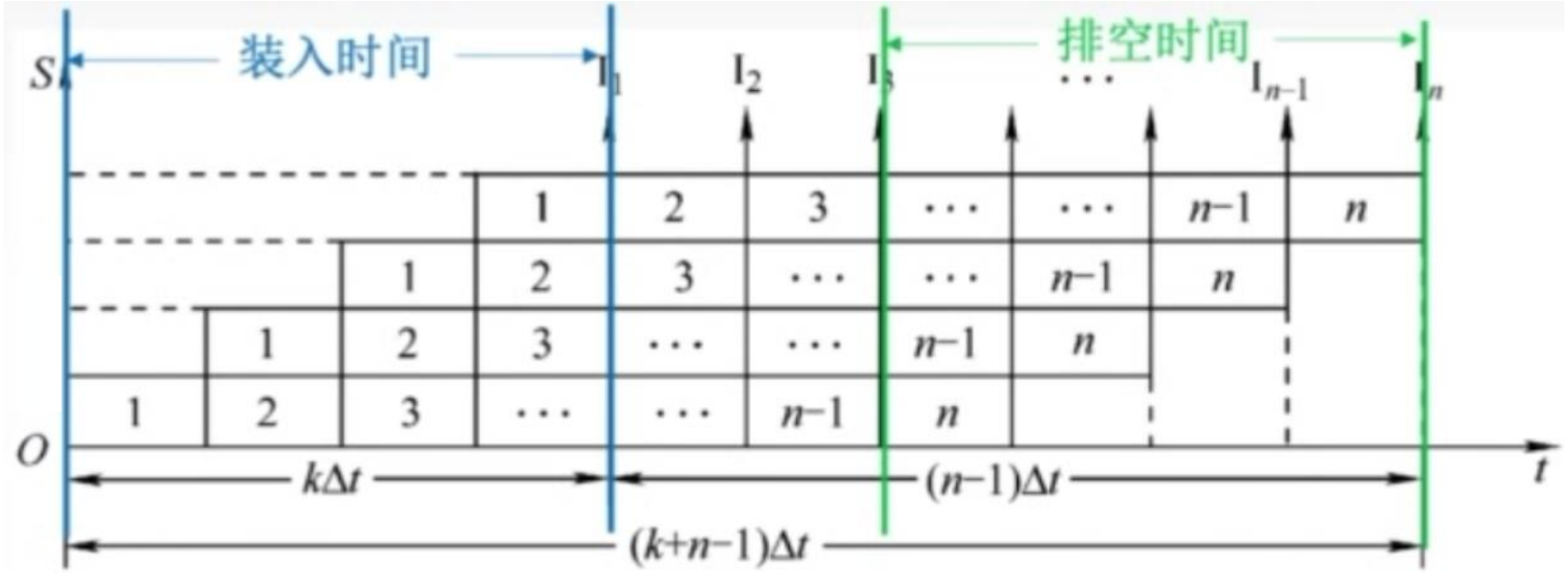
流水线指标1——吞吐率

- 吞吐率定义：单位时间执行指令条数
- 设指令条数为n，每个阶段用时Δt
- 公式：$\mathrm{TP}=\frac{n}{T}=\frac{n}{k\Delta t+(n-1)\Delta t}$ （当$n\to\infty$，取极值）



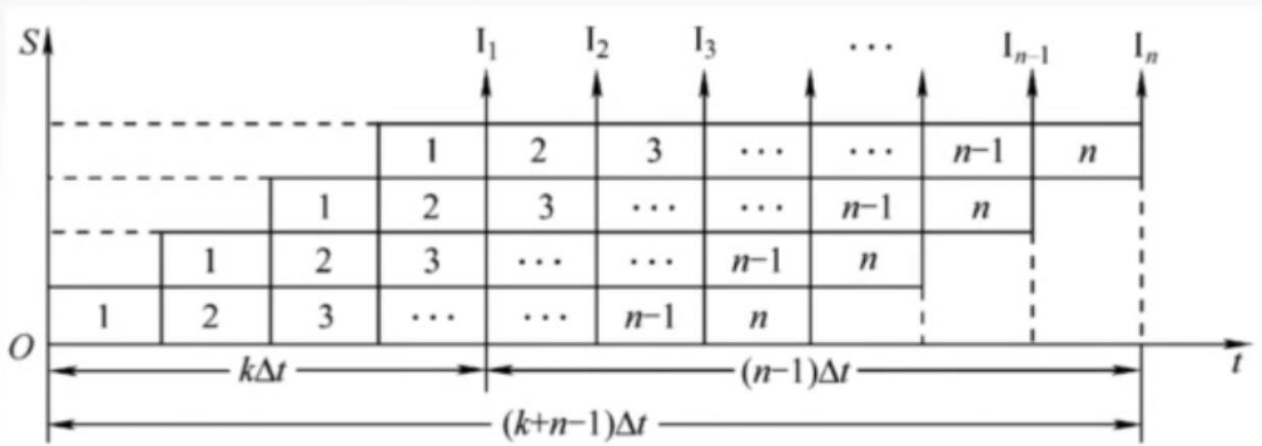
流水线指标1——吞吐率

- 公式：$\mathrm{TP}=\frac{n}{T}=\frac{n}{k\Delta t+(n-1)\Delta t}$ （当$n\to\infty$，取极值）
- 原理：流水线满载时，效率最高

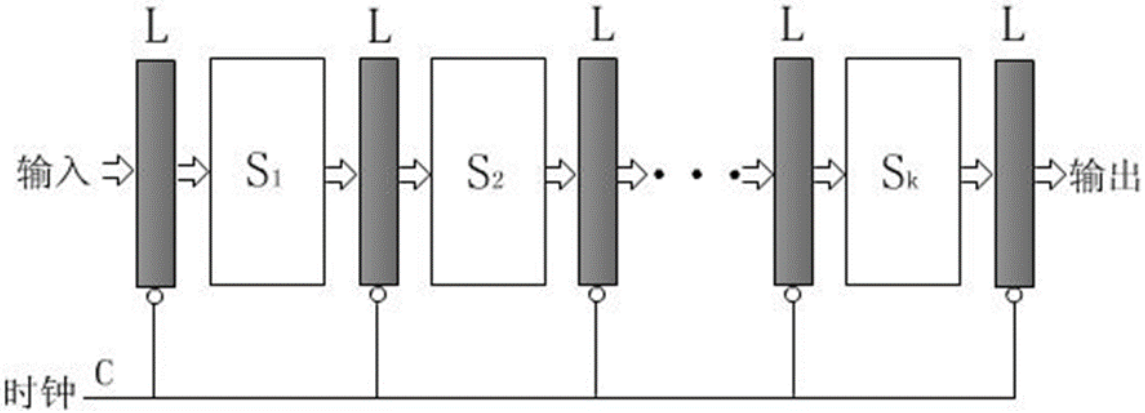


流水线指标2——加速比（理想）

- 加速比定义：非流水执行时间/流水执行时间
- 公式：$S = {}^{T_l}/_{T_k}$（k级流水，执行n条指令，C_k）
  - 非流水线，串行执行$T_L = n\ k\ \Delta t$ （每一级时间）
  - 流水线方式，$T_k = k\ \Delta t + (n-1)\ \Delta t$
  - 加速比，$C_k = T_L/T_k = nk/(k+n-1)$(当$n \to \infty, C_k = k$)



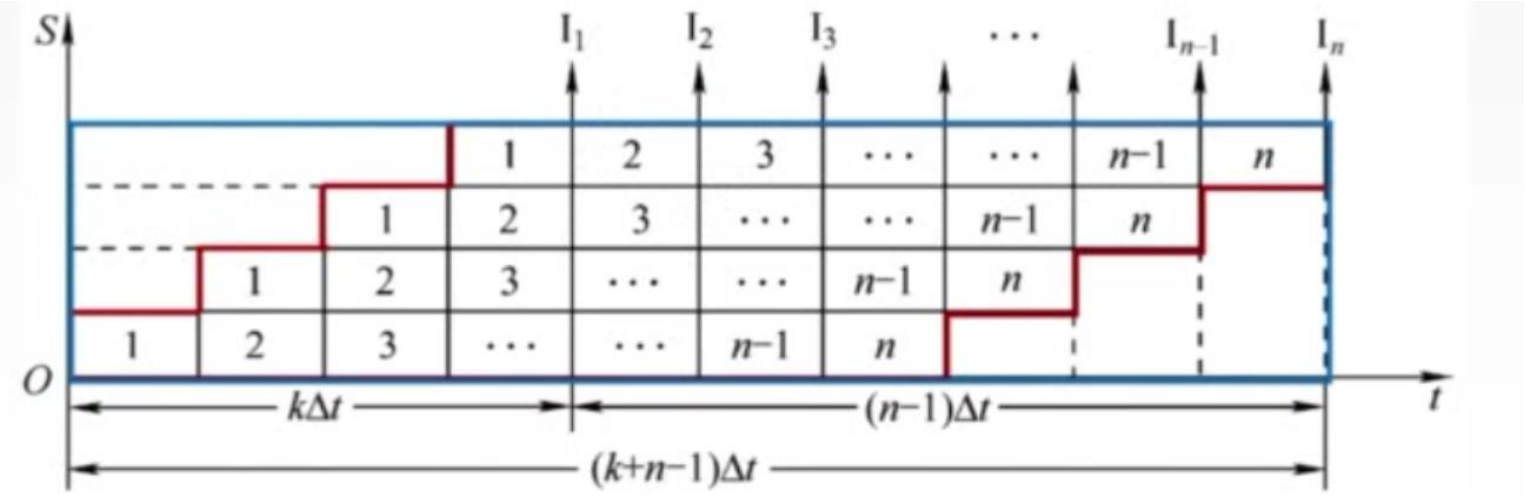
流水线指标2——加速比（实际）



- K级流水CPU
  - 每级流水所需时间为τ_i
  - **缓冲寄存器的延时为τ_l**
- 流水线满载时
  - 每隔$\tau = \max\{\tau_i\} + \tau_l$完成一条指令
  - 完成一条指令所需时延，$T = \sum \tau_i + K\tau_1$
- 无法通过不断划分流水技术提升CPU性能

流水线指标3——效率

- 定义：设备的利用率（时空图占用区域/总区域面积）
- 公式：$E = \frac{(n-1)\Delta t\ \text{（红色）}}{(k+n-1)\Delta t\ \text{（蓝色）}} = 1 - \frac{k}{k+n-1}$
- 当$n \to \infty$，$E = 1$ （满载时效率为1）



一条指令的执行分为k个阶段，每个阶段耗时Δt，一般取Δt=一个时钟周期

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

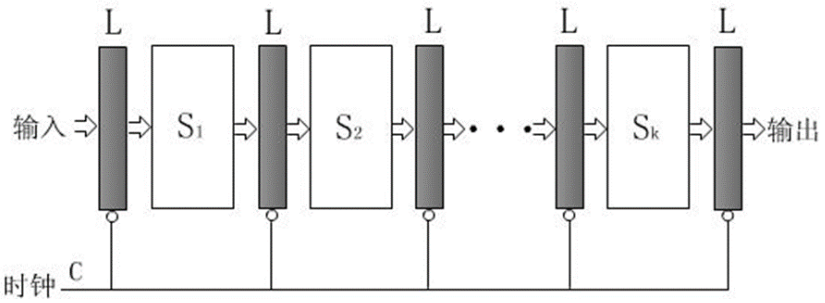
四级流水CPU，每级流水所需时间为200ps，缓冲寄存器延迟50ps
流水线满载时，完成相邻两条指令时间间隔？
完成一条指令延时？

A 800ps, 1000ps

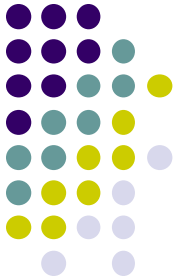
B 200ps, 800ps

C 200ps, 1000ps

D 250ps, 1000ps



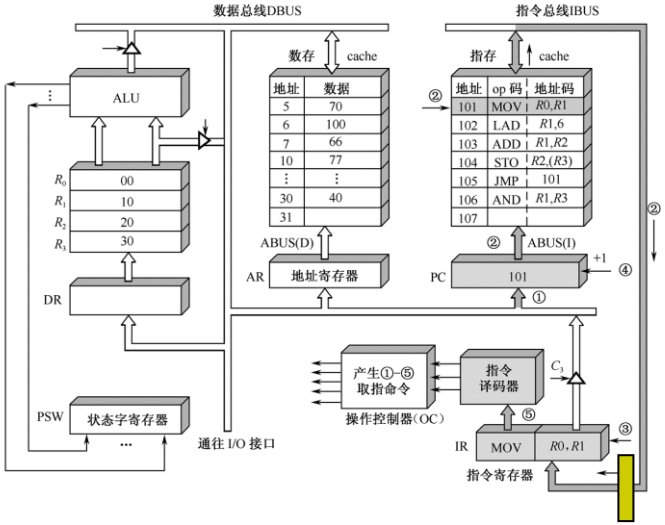
第五章 中央处理机

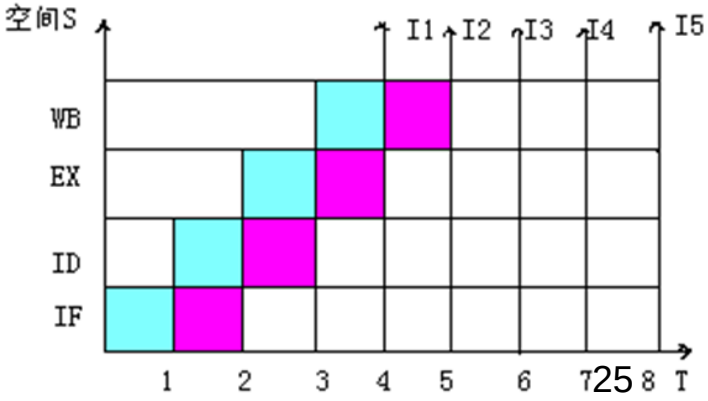


- 流水线技术与性能分析
  - 流水线与图形化表示
  - 流水线性能指标
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险

流水线中的主要问题

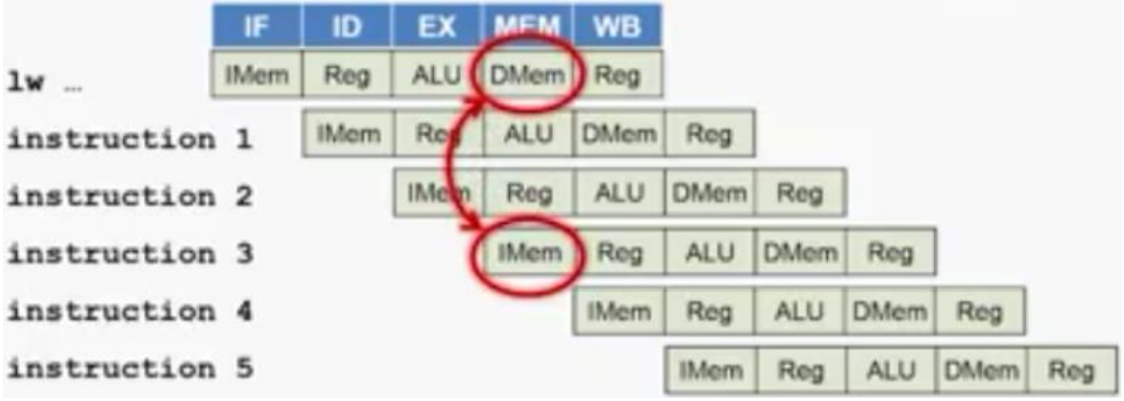
- 流水线基础
  - 取指、译码、执行、回写分别依赖不同部件
  - 加缓冲寄存器隔离，可以并行执行
- 主要问题
  - 共享总线、Cache等
  - 指令顺序，与流水线技术矛盾
  - 指令分支执行
- 三种相关冲突/冒险
  - 资源相关、数据相关、控制相关





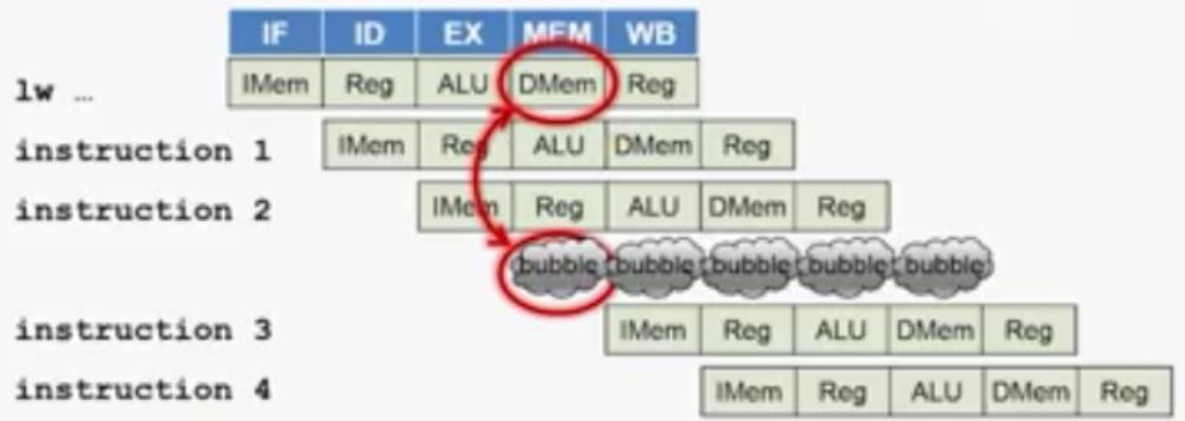
结构冒险

- 结构冒险：资源相关
  - 多条指令在同一时钟周期内争用同一功能部件
- 考虑五级流水指令划分：
  - 取指、译码、执行、访存（MEM）、回写
- 若第一条指令为存数（Load）指令
  - 在第4个时钟周期，会产生结构冒险
    - 指令与数据存放在同一存储器，不能同时读取
    - I3的取指阶段（读）与存数指令的访存（写）冲突



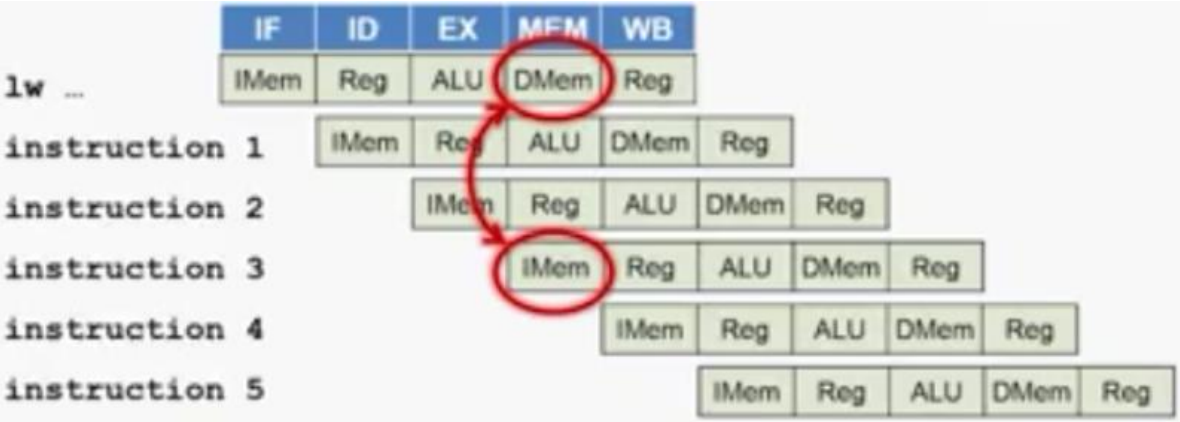
结构冒险解决方法（1）

- 空泡（Bubble）技术
  - 执行空指令（NOP）
  - 流水线产生停顿
  - I3指令向后延迟一个时钟周期，避免产生冲突
- 面临问题
  - 流水线效率降低
  - 若I1指令又是存数指令，还需继续进行空泡

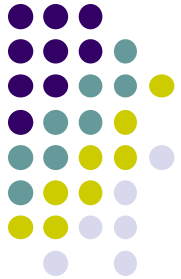


结构冒险解决方法（2）

- 硬件结构修改
  - 新增部件，避免冲突产生
  - 现有CPU中有独立的I-Cache和D-Cache
  - 支持同时进行指令与数据的读写
- 优点
  - 流水线不产生停顿，效率较高



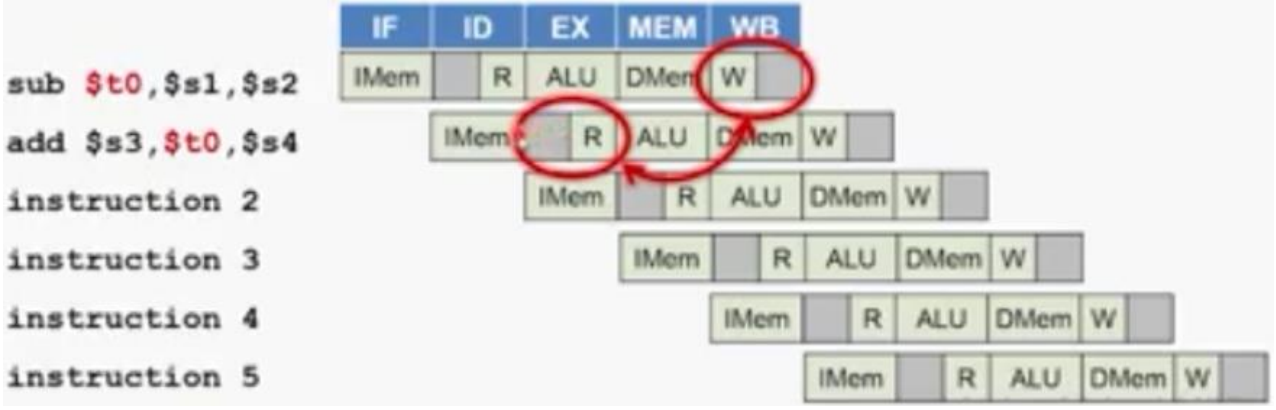
第五章 中央处理机



- 流水线技术与性能分析
  - 流水线与图形化表示
  - 流水线性能指标
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险

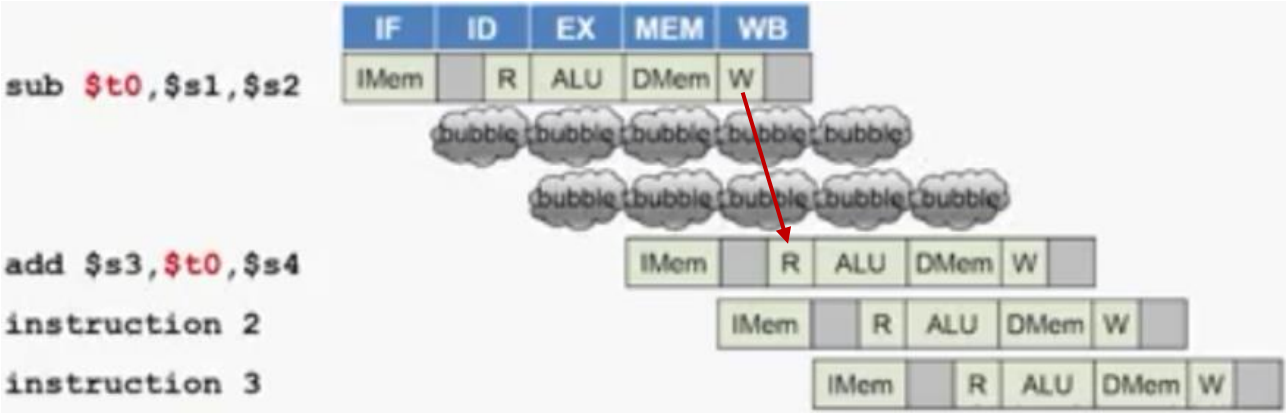
数据冒险

- 数据冒险原因
  - 指令顺序执行，流水线技术破坏上述顺序
- 例如，连续执行两条加减法指令
  - ADD指令在第三个时钟周期需要t0的值
  - SUB指令在第五个时钟周期才能写回寄存器
- 数据冒险分类
  - RAW(Read After Write)
  - WAW(Write After Write)、WAR(Write After Read)



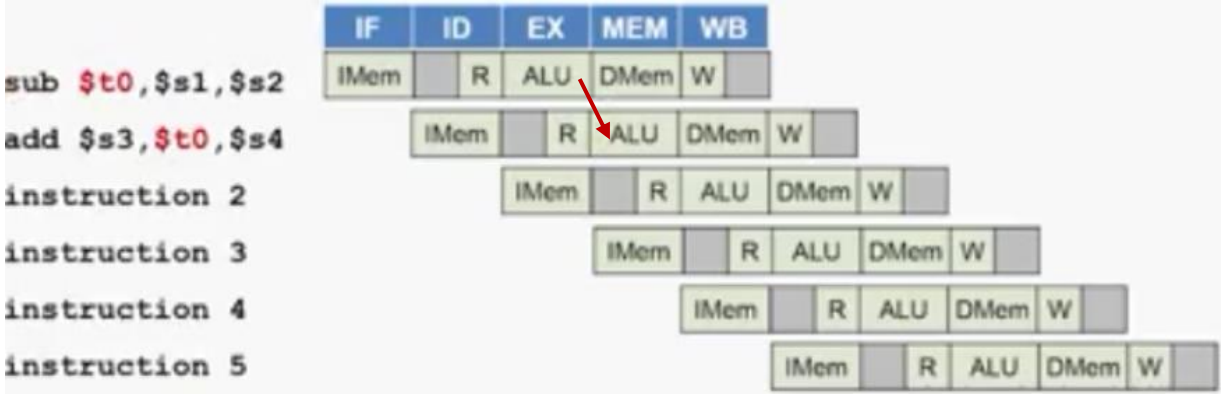
数据冒险解决方法（1）

- 空泡（Bubble）技术
  - 执行空指令（NOP）
  - 流水线产生停顿
  - ADD指令停顿两个时钟周期
- 缺点
  - 流水线效率下降

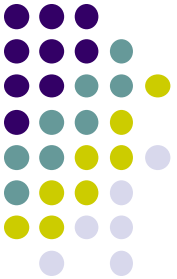


数据冒险解决方法（2）

- 数据前递（Forward）
  - 设置各级（如ALU输出端）与输入端的直接通路
  - 添加额外数据通路（硬件）
  - 由ADD指令ALU的输出直接将$t0寄存器的值传递给SUB指令的ALU的输入端
- 优点
  - 避免流水线停顿，保障效率



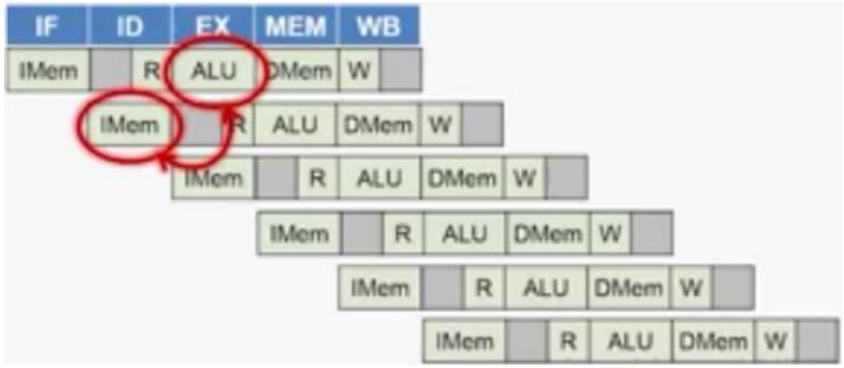
第五章 中央处理机

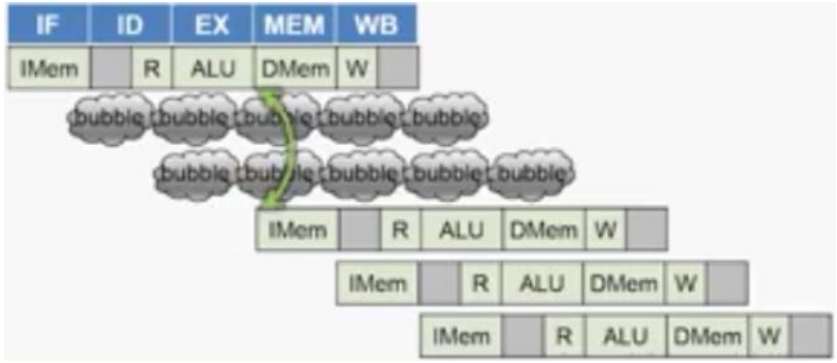


- 流水线技术与性能分析
  - 流水线与图形化表示
  - 流水线性能指标
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险

控制冒险

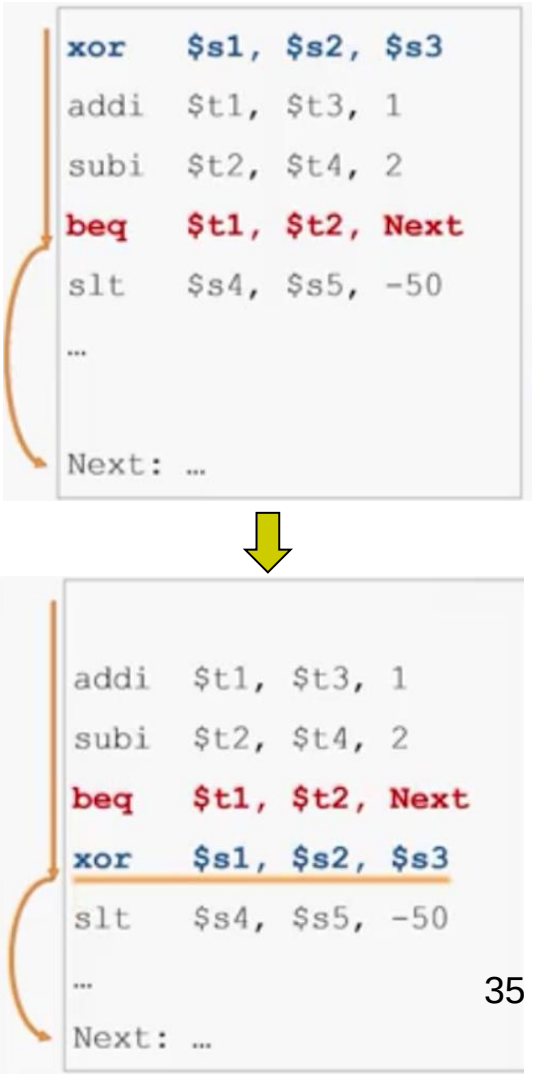
- 控制冒险原因
  - 指令转移破坏流水线结构
  - 无条件/条件转移指令
- 例如，BEQ条件分支指令
  - 下一条指令地址取决于第一条指令判断结果
- 空泡技术
  - 流水线停顿
  - 第二条指令停顿两个时钟周期





控制冒险解决方法

- 延迟转移
  - 基本思想
    - 先执行再转移
    - 转移指令后续指令提前执行
  - 优化
    - 调整指令顺序
- 转移预测法
  - 基本思想
    - 预测未来跳转方式
    - 例如，同时取出转移与顺序指令进入指令队列



流水线冒险例题

【例4】流水线中有三类数据相关冲突：写后读（RAW）相关；读后写（WAR）相关；写后写（WAW）相关。判断以下三组指令各存在哪种类型的数据相关。

(1) I1　ADD R1，R2，R3　；（R2）+（R3）->R1
　　I2　SUB R4，R1，R5　；（R1）-（R5）->R4

(2) I3　STO M（x），R3　；（R3）->M(x)，M(x)是存储器单元
　　I4　ADD R3，R4，R5　；（R4）+（R5）->R3

(3) I5　MUL R3，R1，R2　；（R1）×（R2）->R3
　　I6　ADD R3，R4，R5　；（R4）+（R5）->R3

（1）I1　ADD R1，R2，R3　；（R2）+（R3）->R1

　　I2　SUB R4，R1，R5　；（R1）-（R5）->R4

- 第（1）组指令中，I1指令运算结果应先写入R1，然后在I2指令中读出R1内容。由于I2指令进入流水线，变成I2指令在I1指令写入R1前就读出R1内容，发生RAW相关。

（2）I3　STO M（x），R3　；R3->M(x)，M(x)是存储器单元

　　I4　ADD R3，R4，R5　；（R4）+（R5）->R3

- 第（2）组指令中，I3指令应先读出R3内容并存入存储单元M（x），然后在I4指令中将运算结果写入R3。但由于I4指令进入流水线，变成I4指令在I3指令读出R3内容前就写入R3，发生WAR相关。

（3）I5　MUL R3，R1，R2　；（R1）×（R2）->R3

　　I6　ADD R3，R4，R5　；（R4）+（R5）->R3

- 第（3）组指令中，如果I6指令的加法运算完成时间早于I5指令的乘法运算时间，变成指令I6在指令I5写入R3前就写入R3，导致R3的内容错误，发生WAW相关。

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

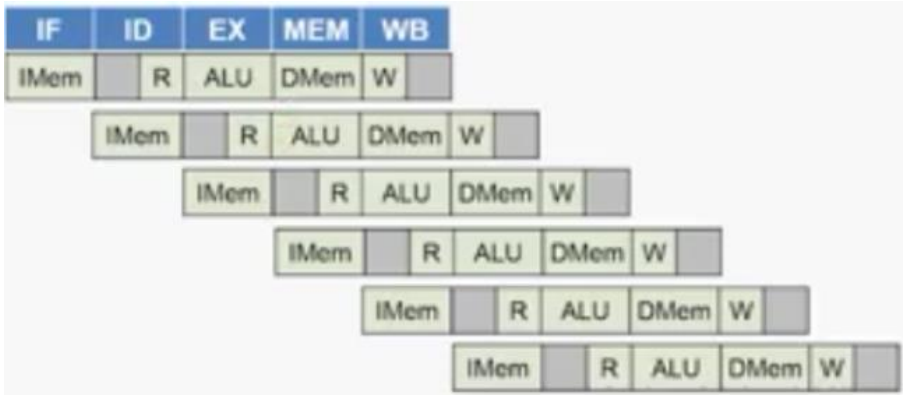
5级流水CPU方式如下，执行
ADD $t0, $s1, $s2; (s1)+(s2)→t0
ADD $s3, $t0, $s2; (t0)+(s2)→s3
是否会发生冒险，冒险方式为?

A 否

B 是；控制冒险

C 是；结构冒险

D 是；数据冒险（RAW）



流水线综合例题

现有四级流水线，各部件分别完成取指、译码与取数、执行、回写操作。现假设各操作完成时间分别为100ns、100ns、80ns、50ns，忽略缓冲器时延。试回答如下问题：

（1）流水线操作周期如何设计？该流水线满载情况下的加速比为多少？

（2）试分析如下指令是否会产生冒险，若有，说明对应冒险种类

ADD R1，R2，R3　　；（R2）+（R3）->R1

SUB R4，R1，R5　　；（R1）-（R5）->R4

（3）如果在硬件上加以改进（避免对应冒险），请说明改进方法，并说明流水线至少应推迟多少时间？

流水线综合例题——解答

现假设各操作完成时间分别为100ns、100ns、80ns、50ns，忽略缓冲器时延

（1）流水线操作周期如何设计？该流水线满载情况下的加速比为多少？

答：100ns、Ck=330ns/100ns=3.3

（2）试分析如下指令是否会产生冒险，若有，说明对应冒险种类

ADD R1，R2，R3　　；（R2）+（R3）->R1

SUB R4，R1，R5　　；（R1）-（R5）->R4

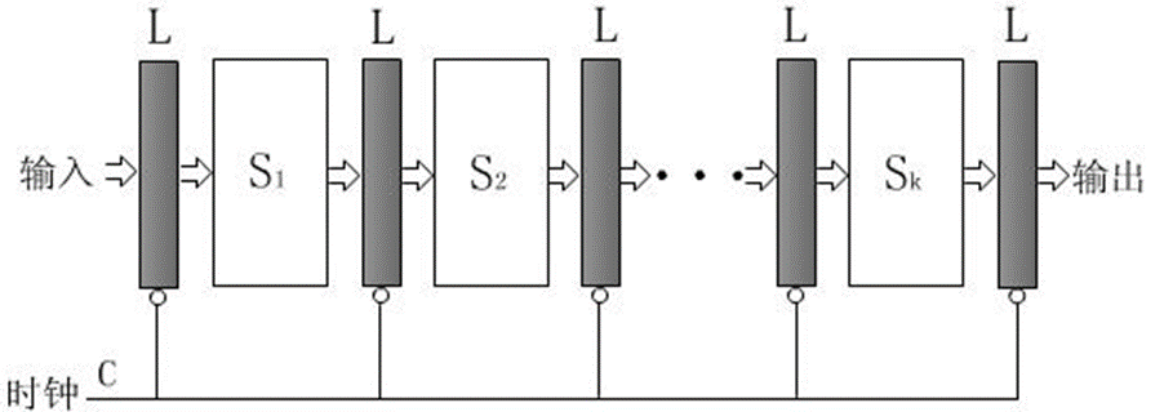
答：数据冒险（RAW）

（3）如果在硬件上加以改进（避免对应冒险），请说明改进方法，并说明流水线至少应推迟多少时间?

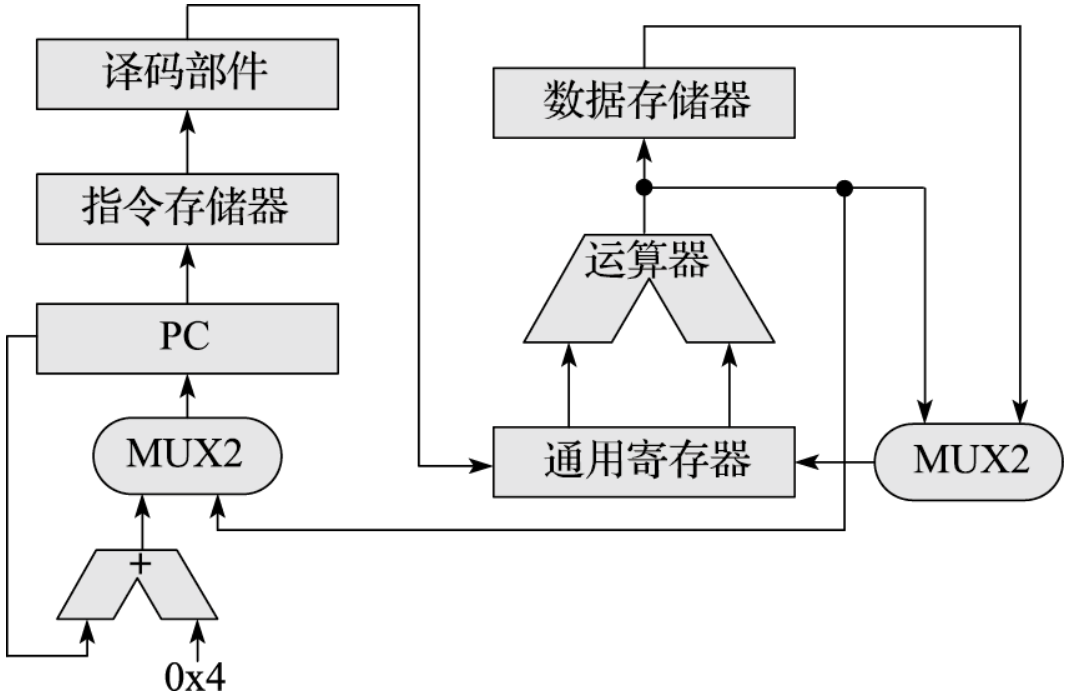
答：采用数据前递技术，不需要推迟时间

流水线技术 vs. 数据通路

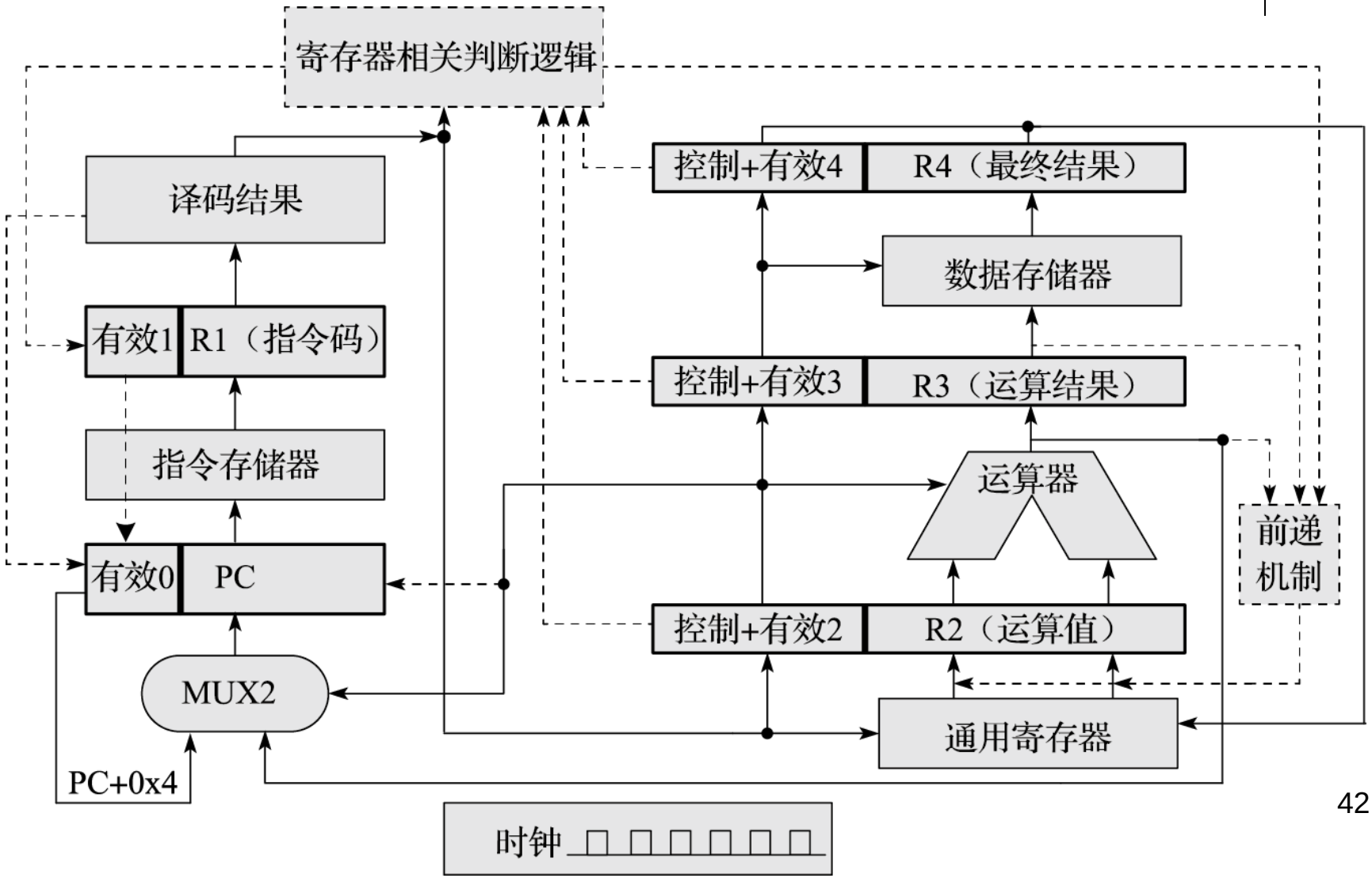
流水线



基本数据通路图



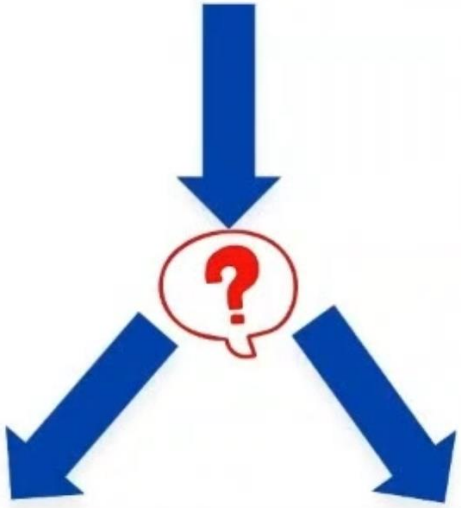
流水线数据通路 vs. 数据前递



扩展1：控制冒险如何缓解

- **分支预测（Branch Prediction）用来预测下一条指令，提高CPU流水线的效率**
- 当指令流水线中出现**条件分支**时，流水线中的指令需要**等待判断结果**，这就会导致了**流水线停顿**，对处理器性能有很大影响。分支预测提前预判结果，减少流水线停顿，提高执行效率

向左？向右？



静态分支预测

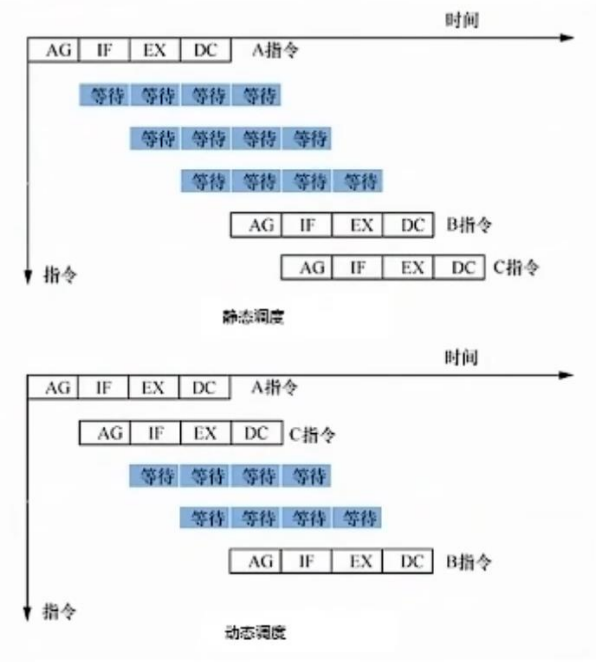
- 静态预测采用既定的策略进行预测，因此受策略本身的精确度及程序类型影响较大
- 实现起简单、成本低，预测正确率的波动范围很大
- 静态预测有always not taken，always taken，BFTN（backward taken，forward not taken等

动态分支预测

- 动态分支预测：根据指令的不同及历史信息作出相应的预测
- 相比静态预测，引入了分支历史的结果作为参考，预测准确率得到提高
- 动态预测策略有Saturating counter，Two-level predictor，Local branch prediction，Global branch prediction等

扩展1：控制冒险如何缓解

乱序执行（Out-of-Order Execution），又称为动态执行，是指在CPU内部执行过程中，指令执行的实际顺序可能和软件中的顺序不同。根据实际情况重新排布，乱序执行极大地**提高了处理器的资源利用率和性能**。特点是“**有序取指、重新排列执行顺序、有序结束**”



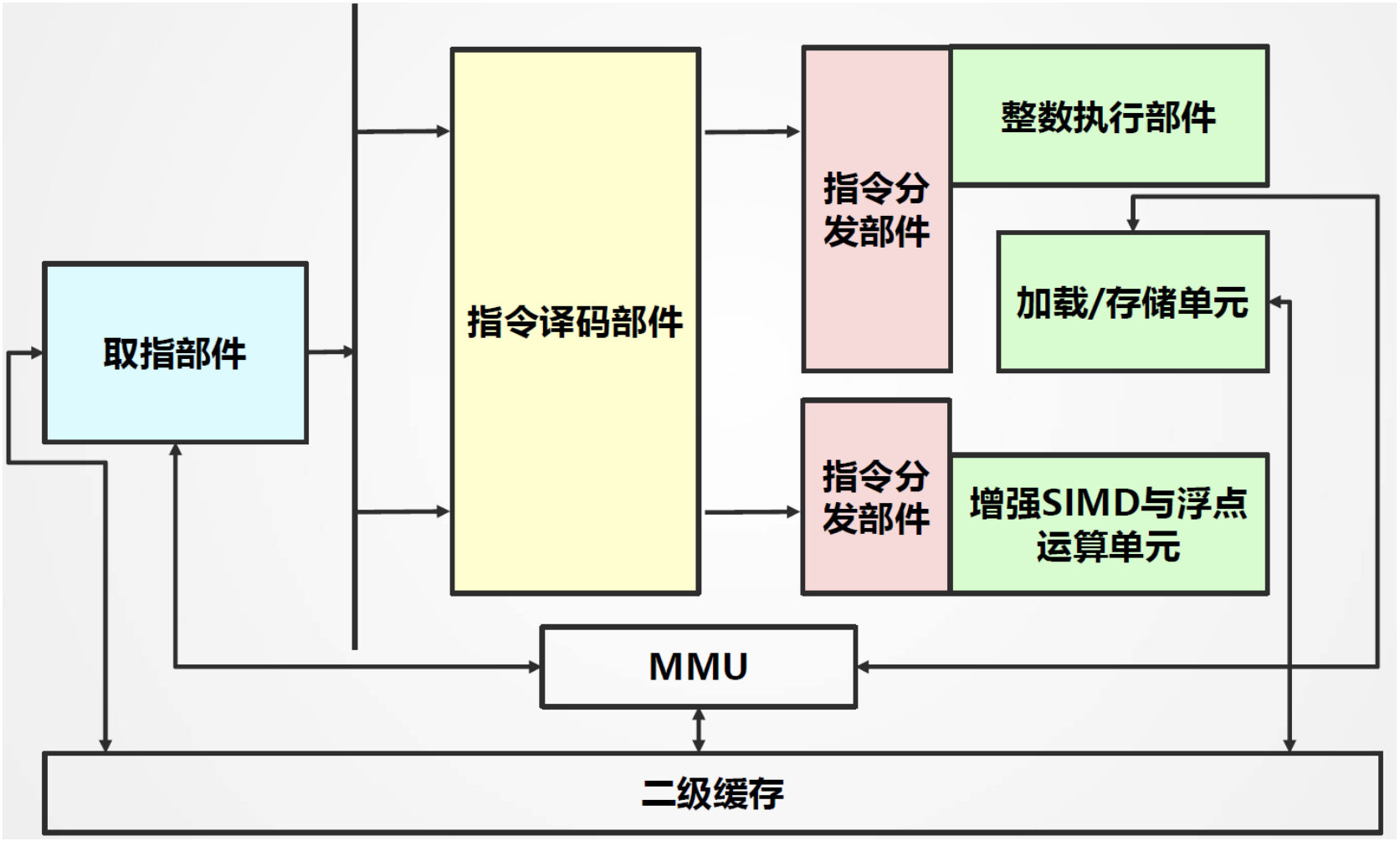
乱序执行的典型例子

主要步骤

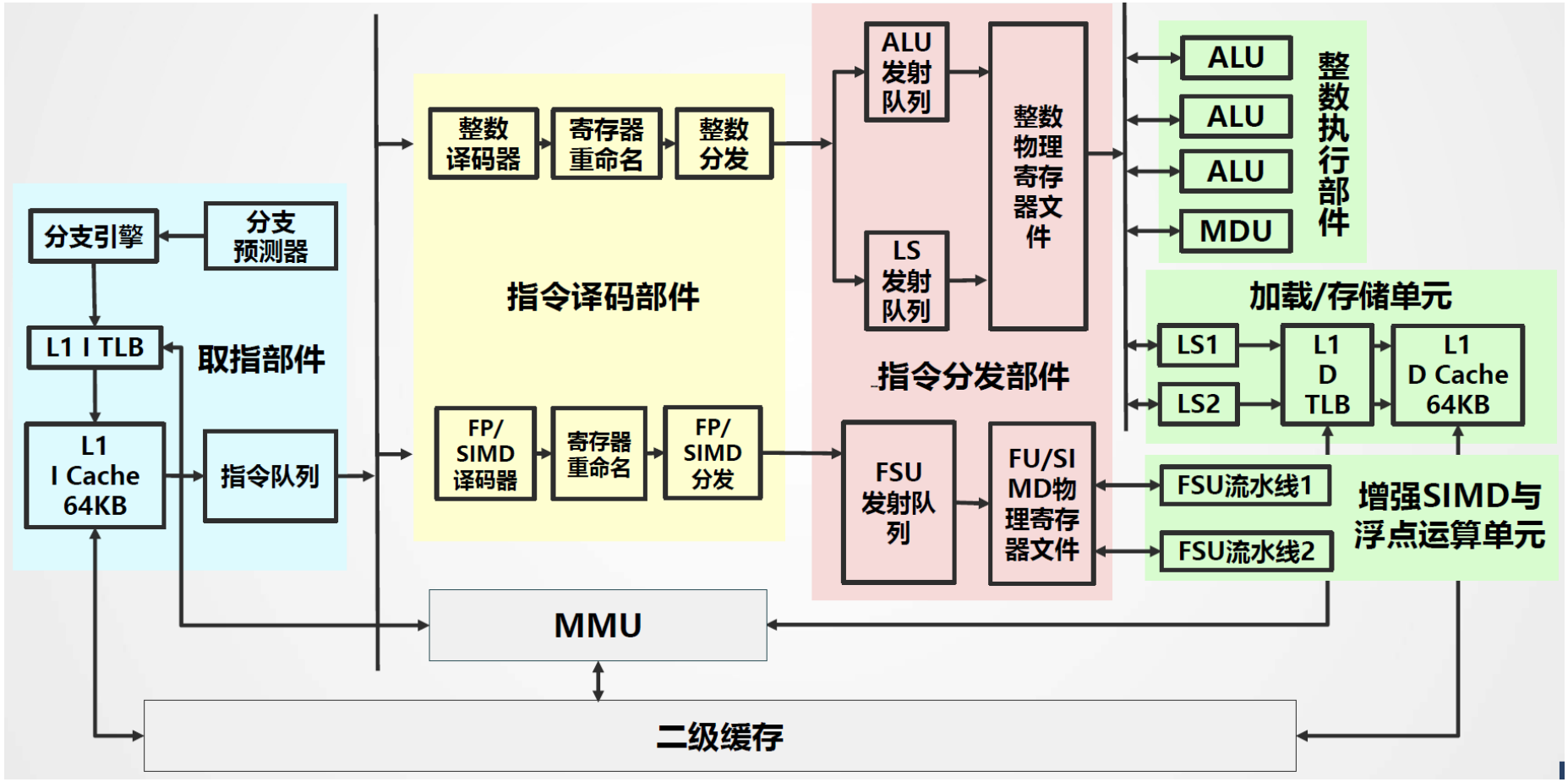
优点

- **高效利用资源**：乱序执行技术允许处理器在等待数据或者资源时执行其他的指令，大大提高了资源的使用效率。
- **提升性能**：乱序执行能够让处理器尽可能地并行执行多个指令，提高了处理器的运行速度

扩展2：鲲鹏处理器流水线结构



扩展2：鲲鹏处理器流水线结构



扩展：硬件安全—研究背景

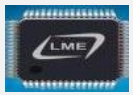
硬件是计算机系统的基础

硬件芯片

处理器


FPGA

GPU



物联网


加解密

车联网


通信


导航定位

固件接口

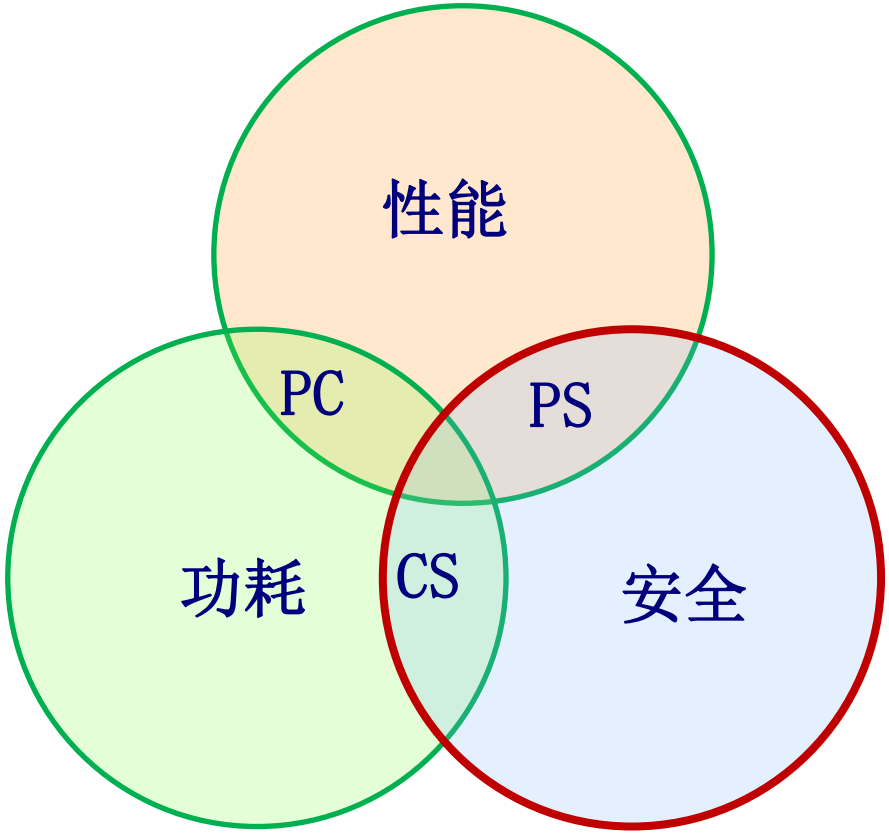
系统软件

计算设备

硬件攻击非常致命

如果硬件安全无法保障，与之紧密关联的软件和网络安全将无从谈起

扩展3：硬件安全—设计因素

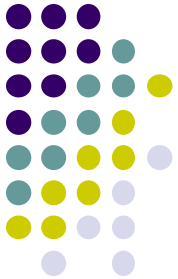


现有漏洞

- 分支预测类
- 乱序执行类
- 值预测类
- 侧信道类

在追求处理器性能、功耗的同时，也需要更加关注硬件的安全性

扩展3：硬件安全—微架构



■ **微架构**是遵循指令集架构的具体硬件实现方案。**处理器微架构技术点众多**，包括执行、内存管理、并行和并发、安全、能源效率等等。在选择或者设计处理器时，这些技术需要综合考虑以满足性能、功耗、成本等多方面需求

执行技术：

- **流水线执行**（Pipeline Execution）
- 指令级并行（Instruction Level Parallelism）
- 数据级并行（Data Level Parallelism）
- **乱序执行**（Out-of-Order Execution）
- 超标量执行（Superscalar Execution）
- 寄存器重命名（Register Renaming）
- 矢量计算和SIMD指令（Vector Processing and SIMD Instructions）

预测和优化技术：

- **分支预测**（Branch Prediction）
- 指令预取（Instruction Prefetching）
- 数据预取（Data Prefetching）
- 动态频率和电压调整（Dynamic Frequency and Voltage Scaling）

并行和并发技术：

- **多线程**（Multithreading）
- 多核（Multicore）

内存和存储技术：

- **多级缓存**（Multi-Level Caching）
- 写回缓存（Write-Back Cache）
- 写穿缓存（Write-Through Cache）
- 内存管理单元（Memory Management Unit）
- 虚拟内存（Virtual Memory）

可靠性和安全性技术：

- 错误检测和纠正（Error Detection and Correction）
- 中断处理（Interrupt Handling）
- 内存屏障（Memory Barrier）
- 硬件锁定（Hardware Locking）
- 硬件虚拟化支持（Hardware Virtualization Support）
- 安全扩展，如Intel SGX（Security Extensions）

功耗和热管理技术：

- 电源管理（Power Management）
- 散热管理（Thermal Management）

扩展3：硬件安全—常见漏洞

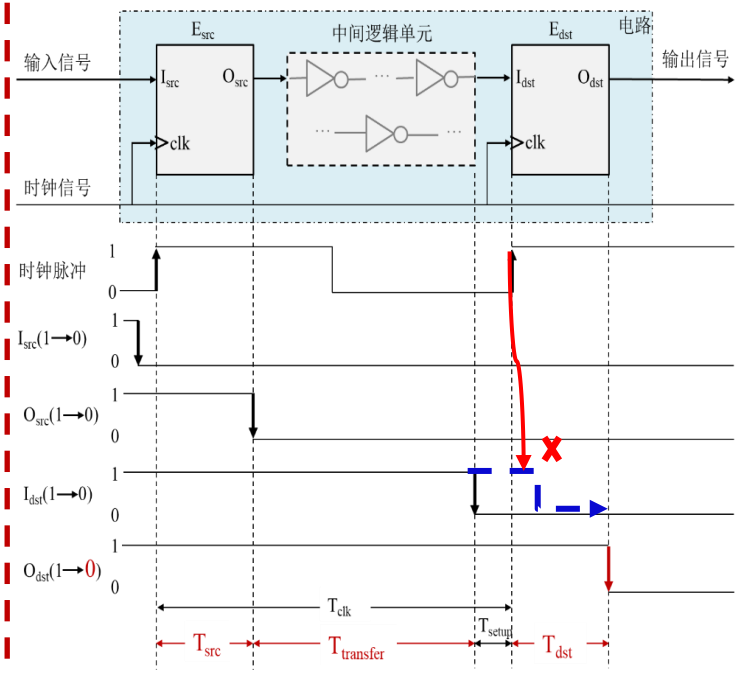
微架构攻击	内容
■ 利用处理器的微架构特性（如乱序执行、预测执行等）来获取信息 ■ 攻击过程完成使用软件实现但是不依赖于任何软件漏洞	■ 乱序执行攻击（以**熔断攻击**为例，CPU会忽略对内存权限的检查，访问受保护内存区域） ■ 预测执行攻击（以**幽灵攻击**为例，CPU执行分支预测时，会将预测执行的数据加载到缓存，不清除）

CPU眼里的：分支预测_哔哩哔哩_bilibili

扩展3：硬件安全—常见漏洞

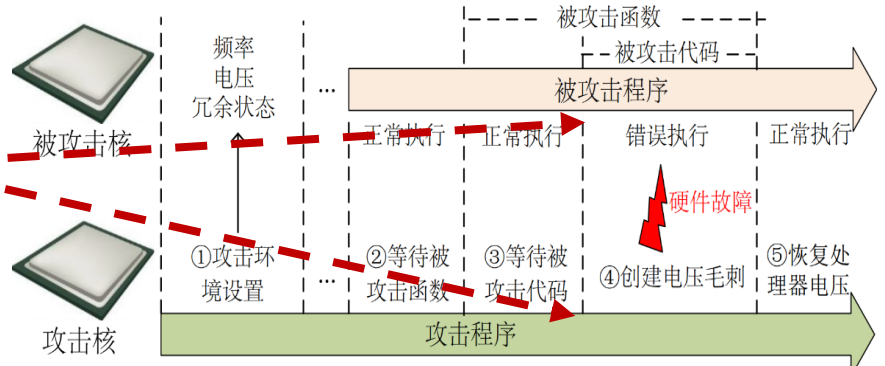
获取核心秘钥、
运行恶意程序等

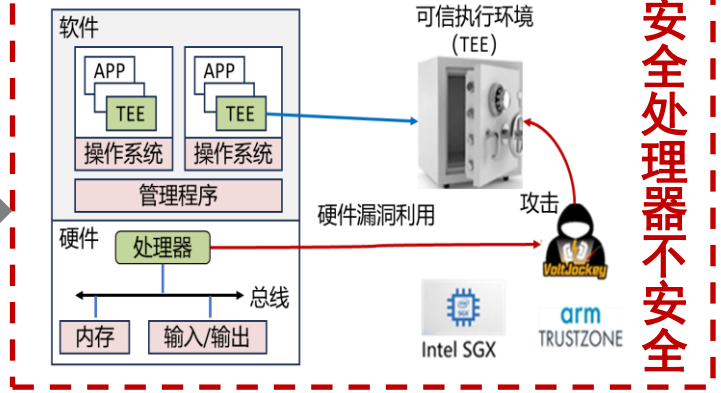
成熟的低功耗技术存在安全漏洞



$$T_{src} + T_{transfer} \leq T_{clk} - T_{setup} - T_{\varepsilon}$$

新一类硬件漏洞

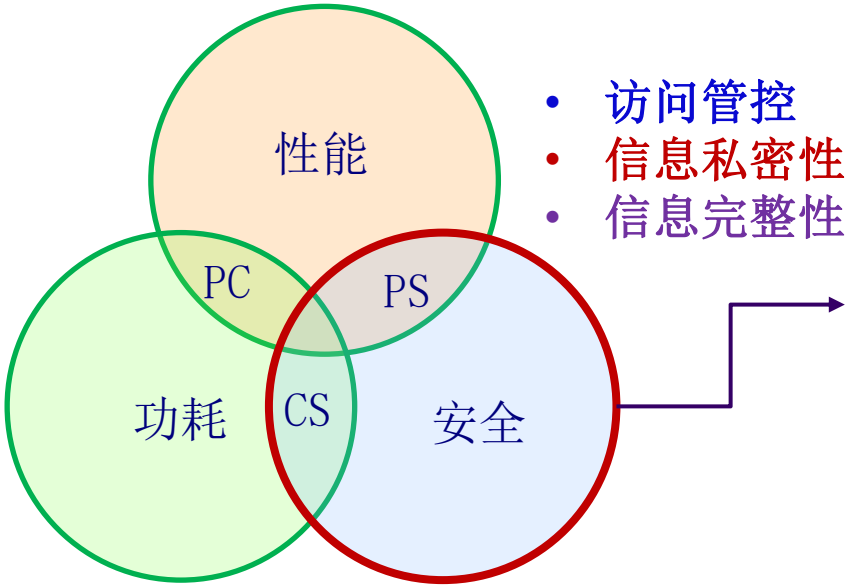




- 国内发现的首个处理器硬件漏洞、首个被国际漏洞库CVE收录的低功耗硬件漏洞

扩展3：硬件安全—安全设计

在“性能、功耗、安全”的芯片设计三角约束下获得更加合理的解决方案

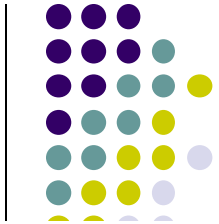


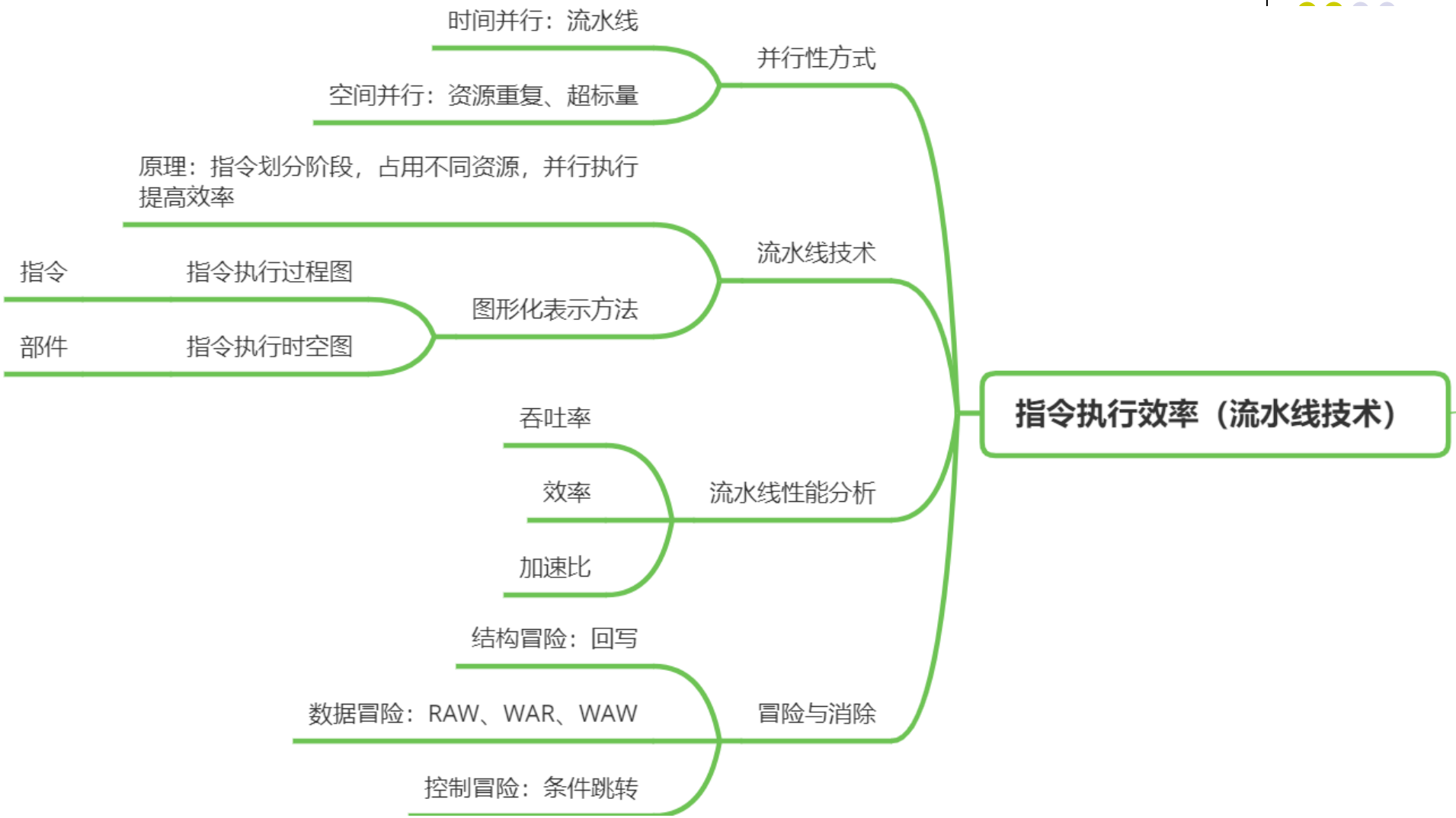
- 访问管控
- 信息私密性
- 信息完整性

安全高效的处理器架构

	设计规则概要
访问控制	1. 穷举组合测试 2. 形式化验证
乱序	1. 瞬态访问控制检查 2. 必要时回滚Cache 3. 页表隔离
预测	1. 预测执行组件不共享 2. 加密预测执行时使用的数据 3. 延迟执行潜在数据泄露指令
侧信道	1. 添加额外的缓存单元 2. 对Cache进行分区 3. 随机化Cache映射
故障注入	1. 驱动隔离 2. 核心数据完整性检查 3. 处理器工作参数异常预警

总结




指令执行效率（流水线技术）
并行性方式
时间并行：流水线
空间并行：资源重复、超标量
流水线技术
原理：指令划分阶段，占用不同资源，并行执行
提高效率
图形化表示方法
指令
指令执行过程图
部件
指令执行时空图
流水线性能分析
吞吐率
效率
加速比
冒险与消除
结构冒险：回写
数据冒险：RAW、WAR、WAW
控制冒险：条件跳转