计算机组成与系统结构

第五章 中央处理器

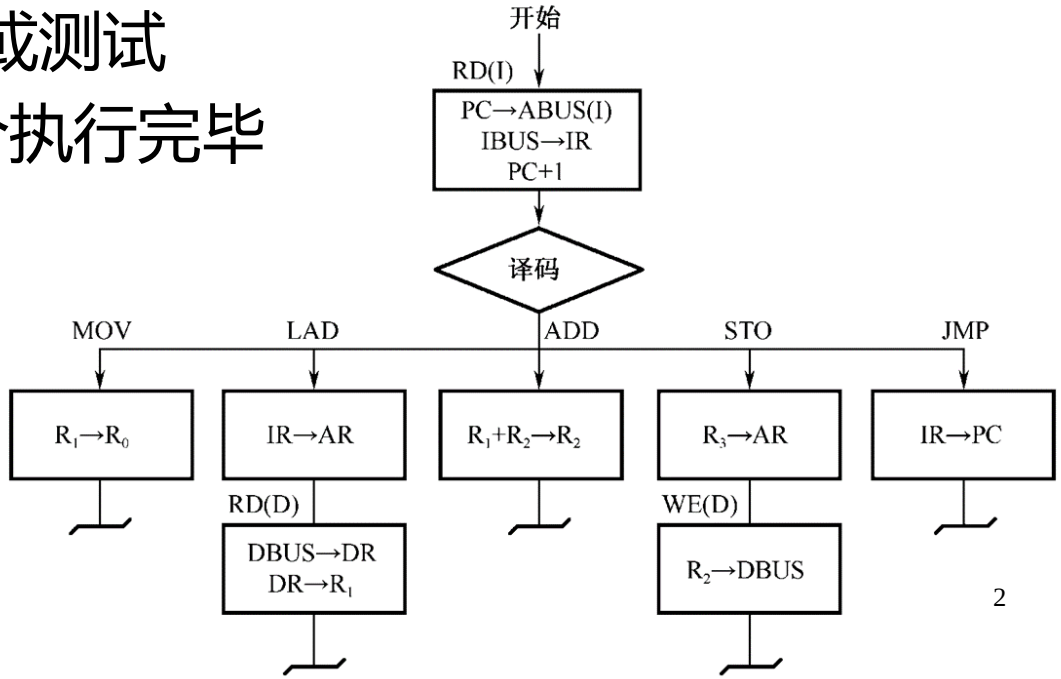
吕昕晨

lvxinchen@bupt.edu.cn

网络空间安全学院

用方框图语言表示的指令周期

- 画数据通路图过程繁琐
- 引入目的主要是为了教学目的（控制器设计）
  - 方框：代表CPU周期（区别指令周期流程图）
    - 方框内内容——数据通路操作或控制操作
  - 菱形符号——判别或测试
  - ~——公操作：指令执行完毕
    - 中断处理等

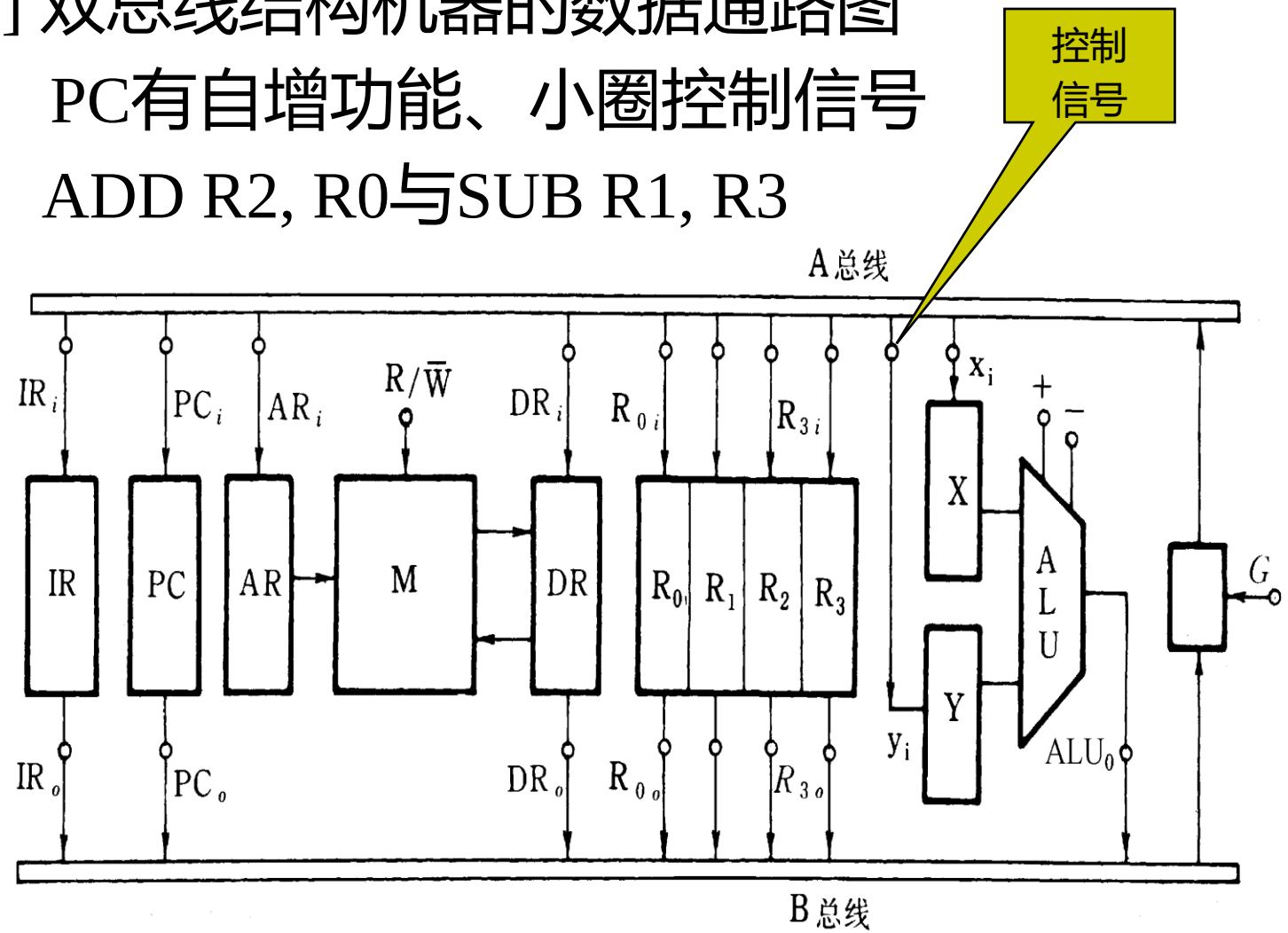


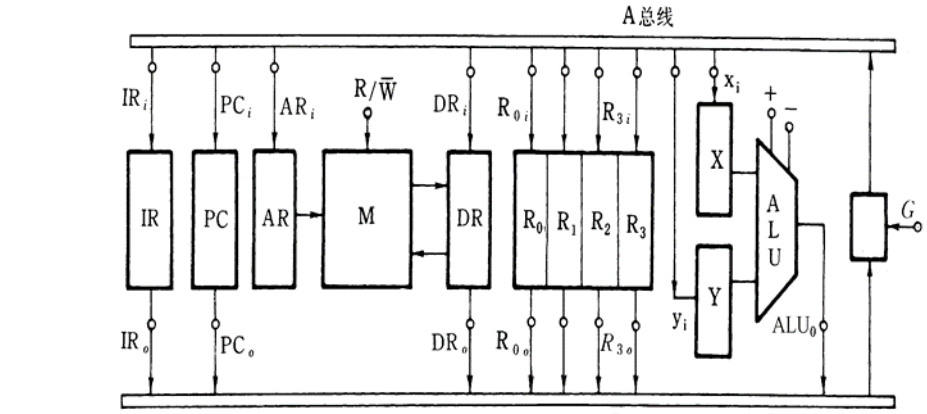
例题

[例1] 双总线结构机器的数据通路图

PC有自增功能、小圈控制信号

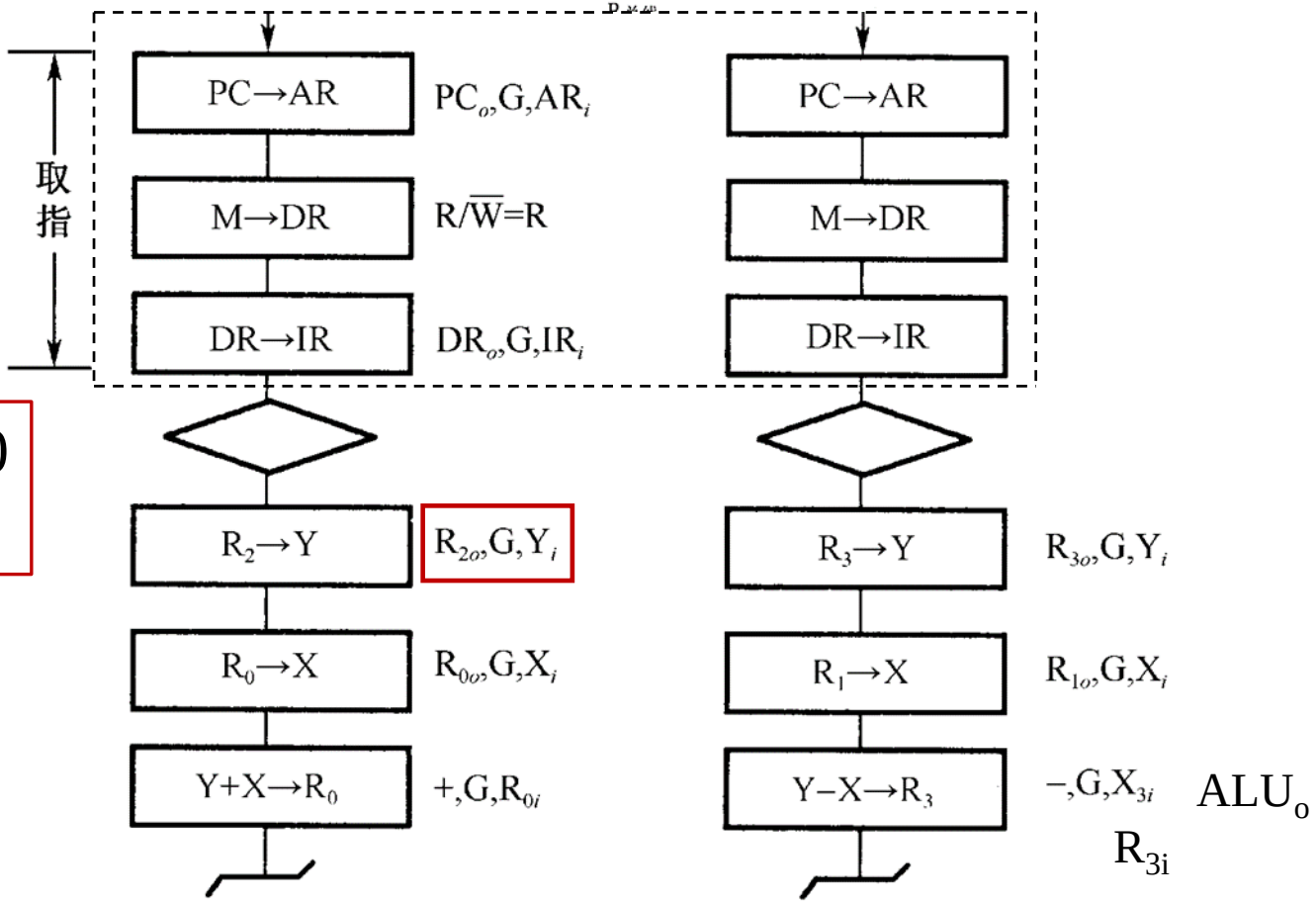
ADD R2, R0与SUB R1, R3

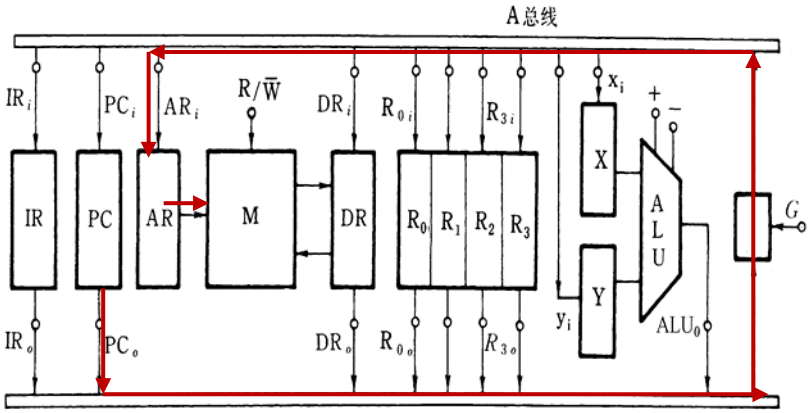




注意微操作控制信号

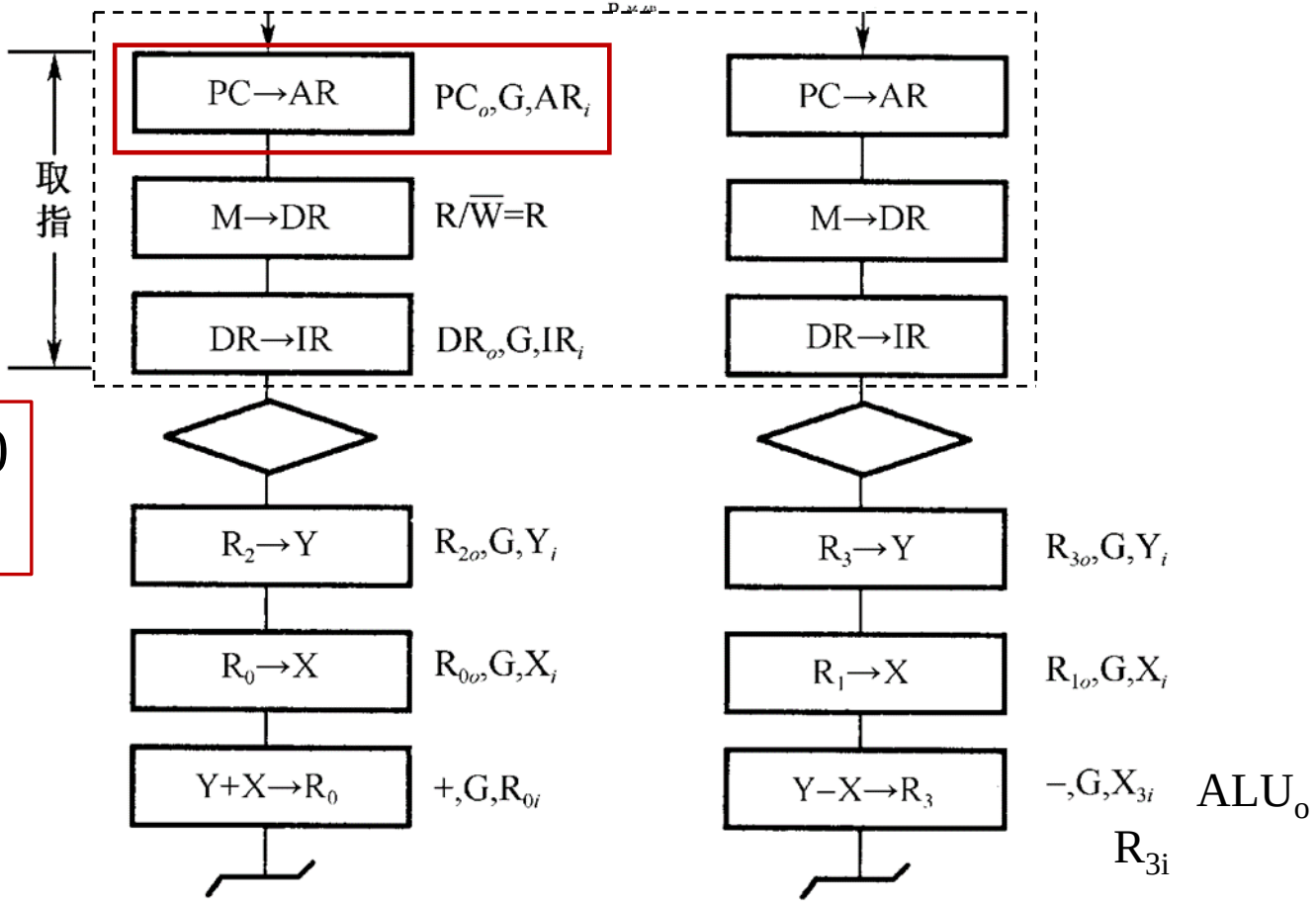
ADD R2, R0
R0+R2→R0

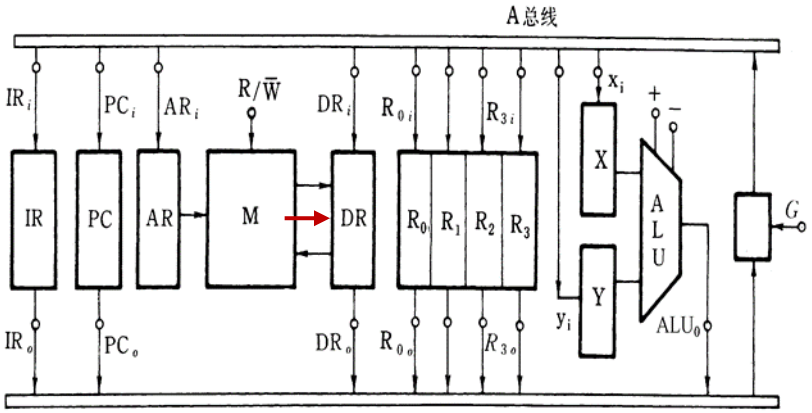




注意微操作控制信号

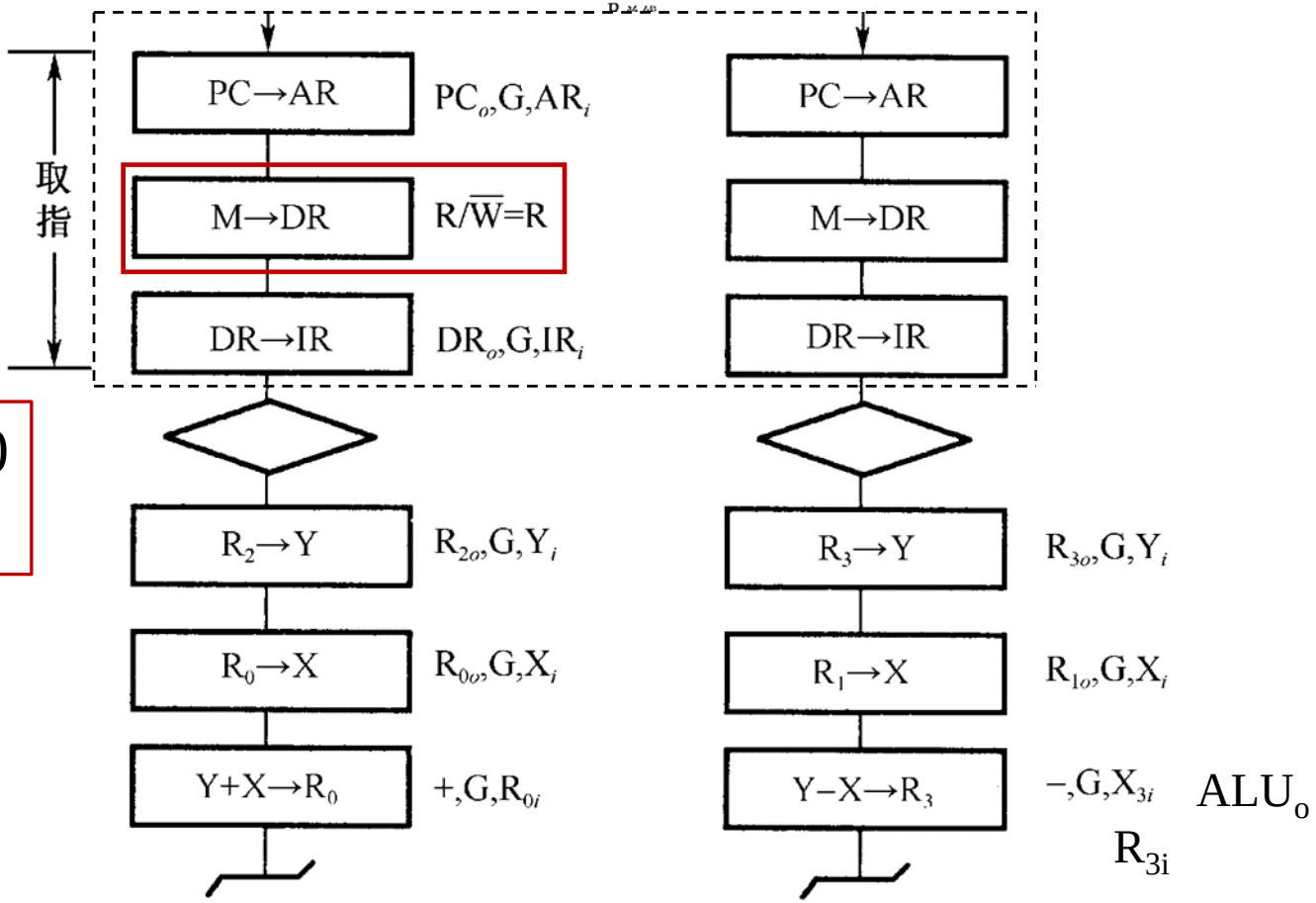
ADD R2, R0
R0+R2→R0

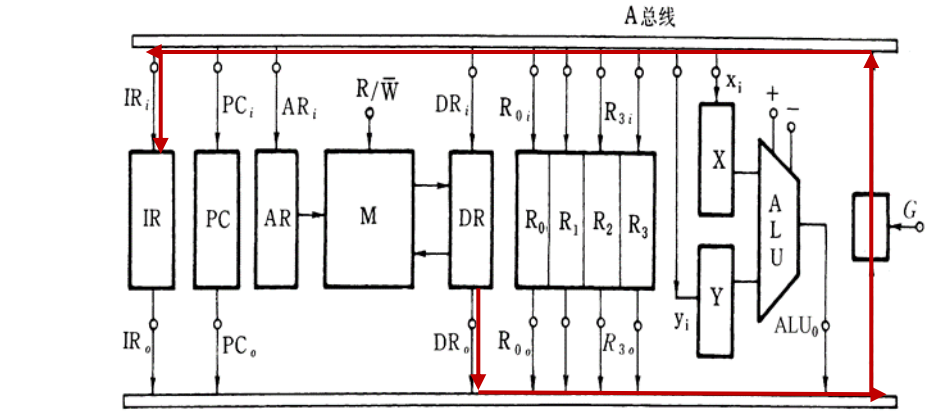




注意微操作控制信号

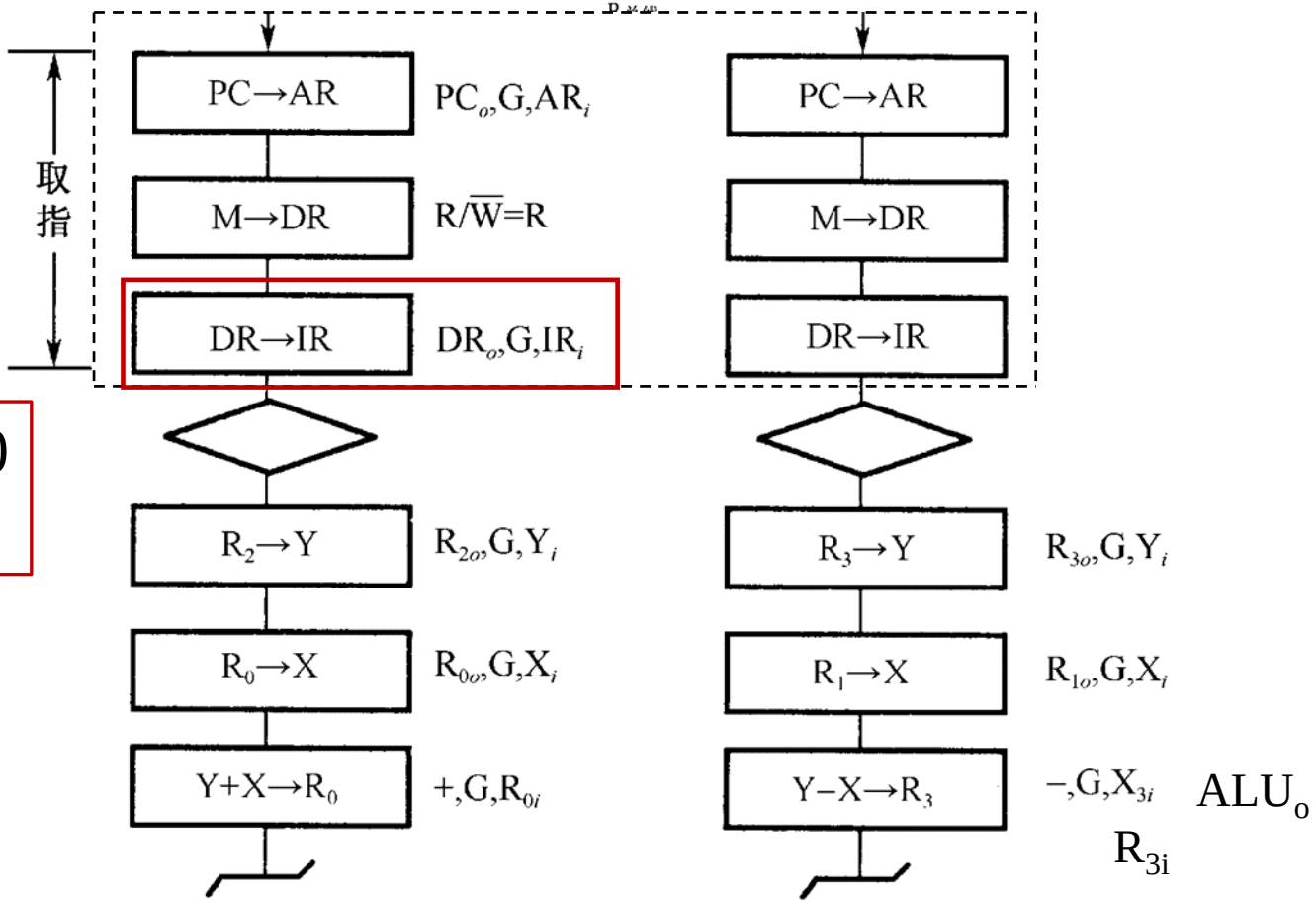
ADD R2, R0
R0+R2→R0

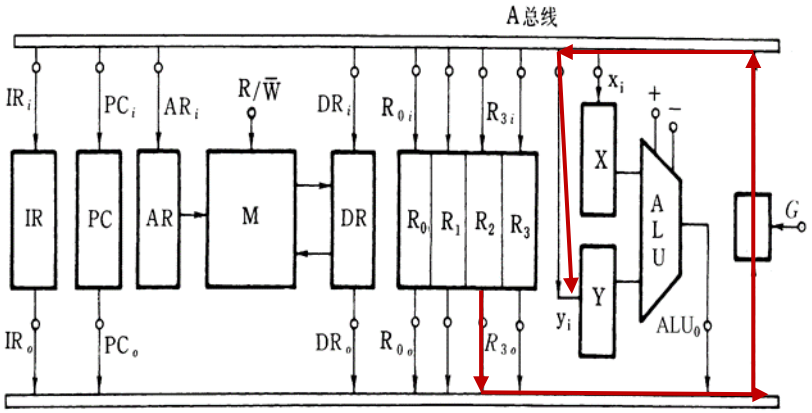




注意微操作控制信号

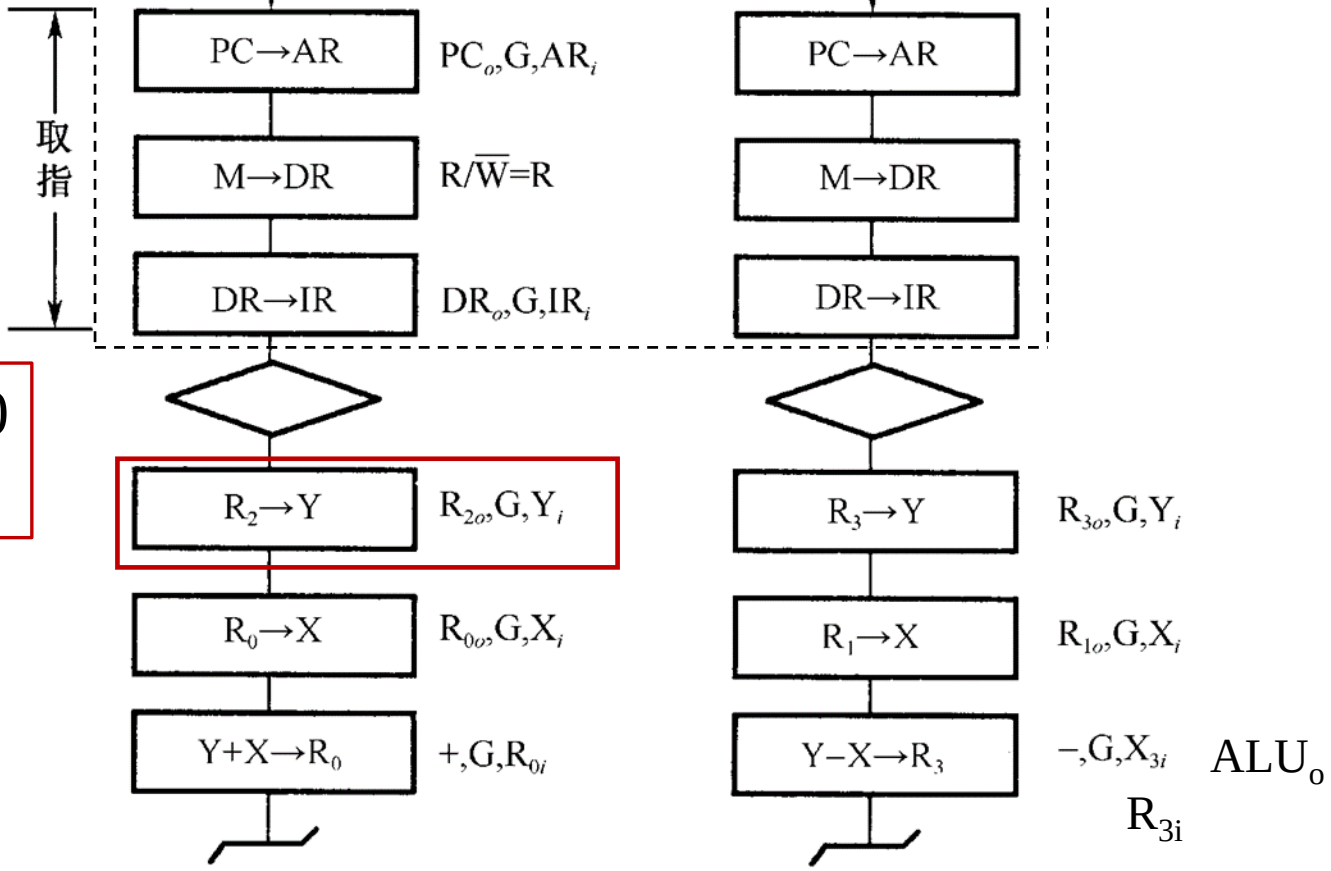
ADD R2, R0
R0+R2→R0

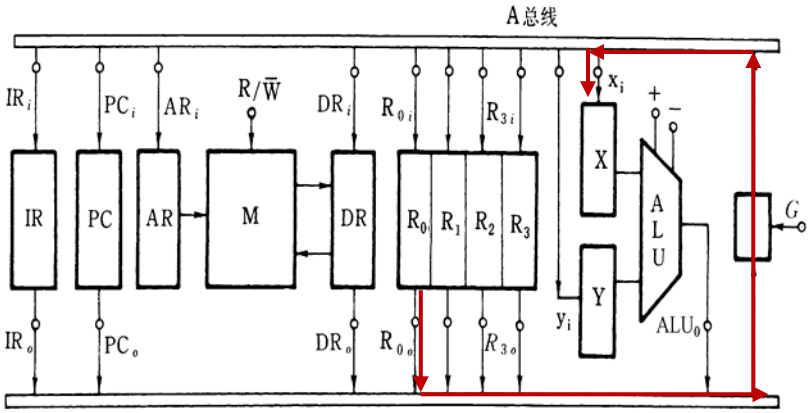




注意微操作控制信号

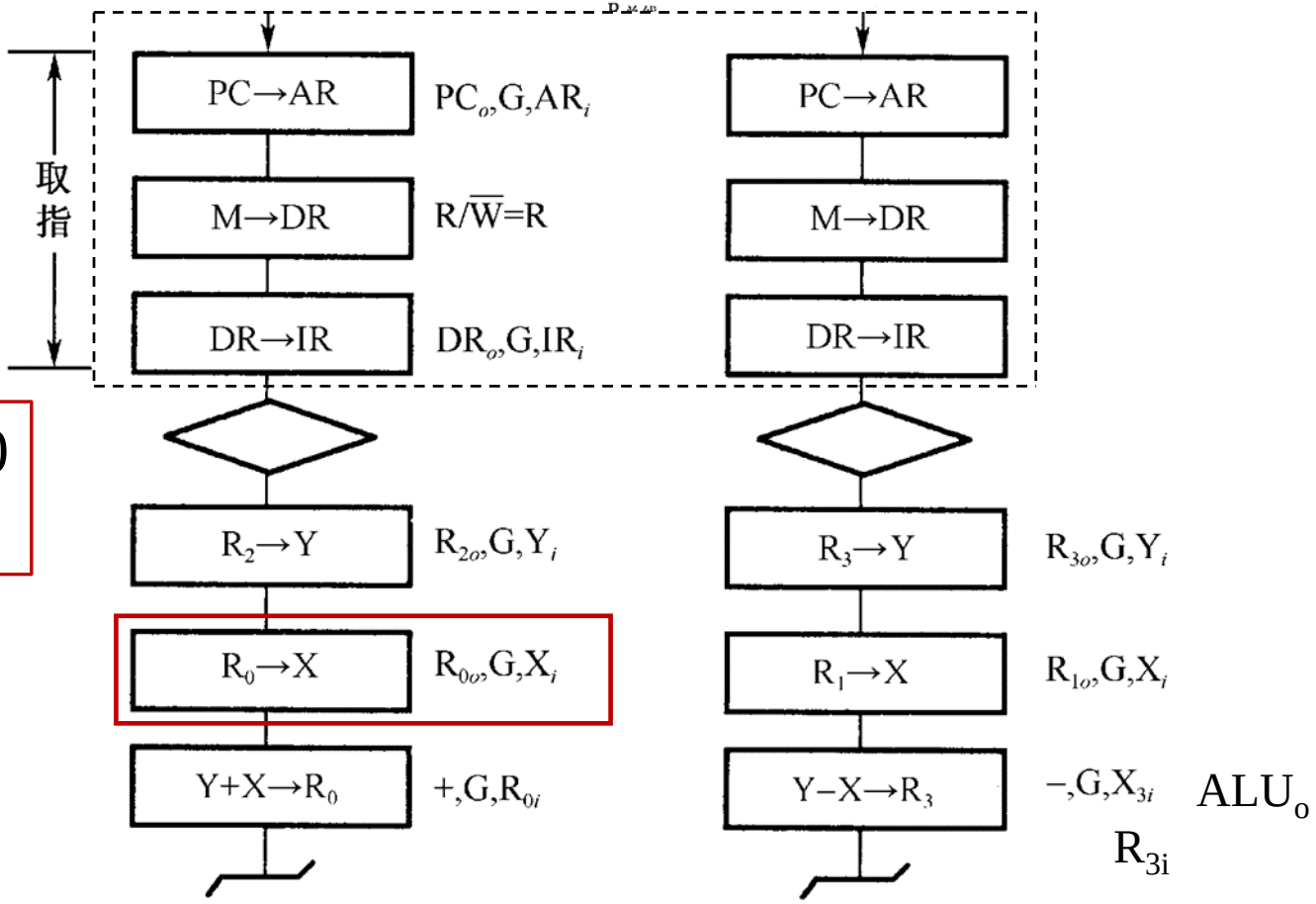
ADD R2, R0
R0+R2→R0

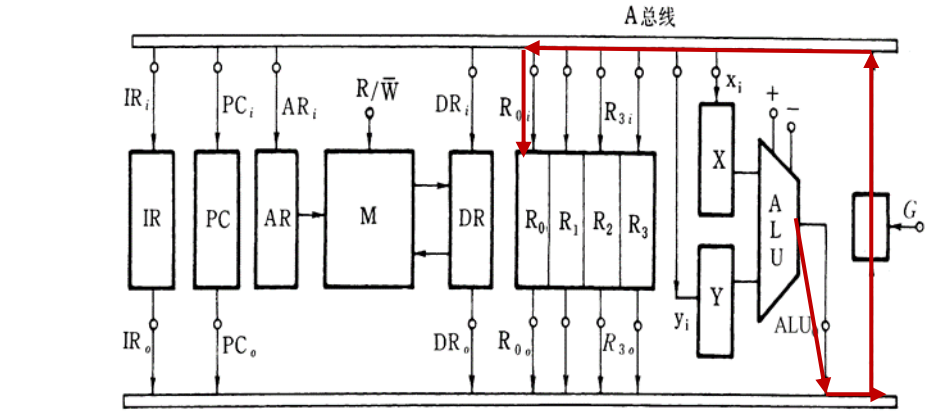




注意微操作控制信号

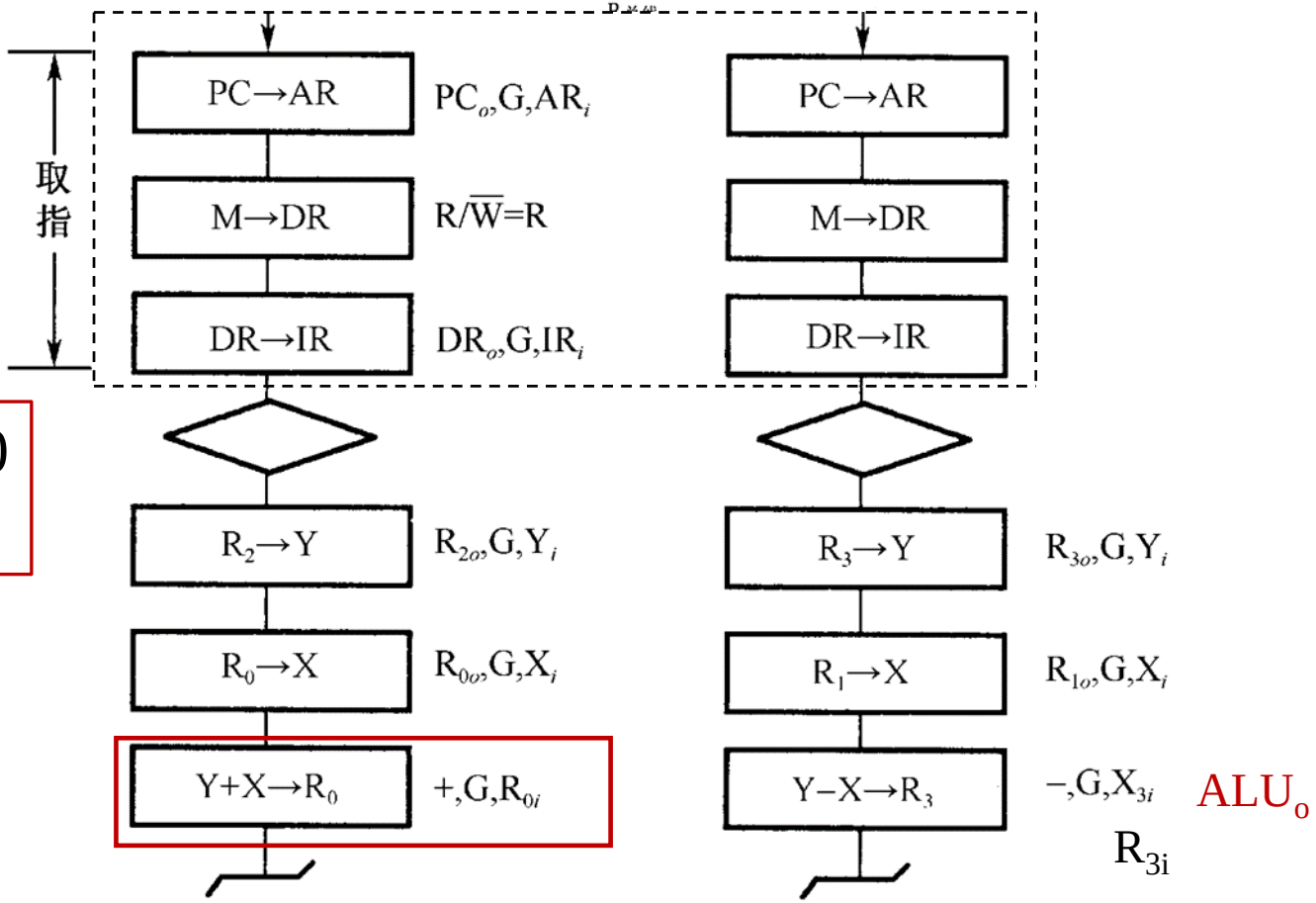
ADD R2, R0
R0+R2→R0





注意微操作控制信号

ADD R2, R0
R0+R2→R0

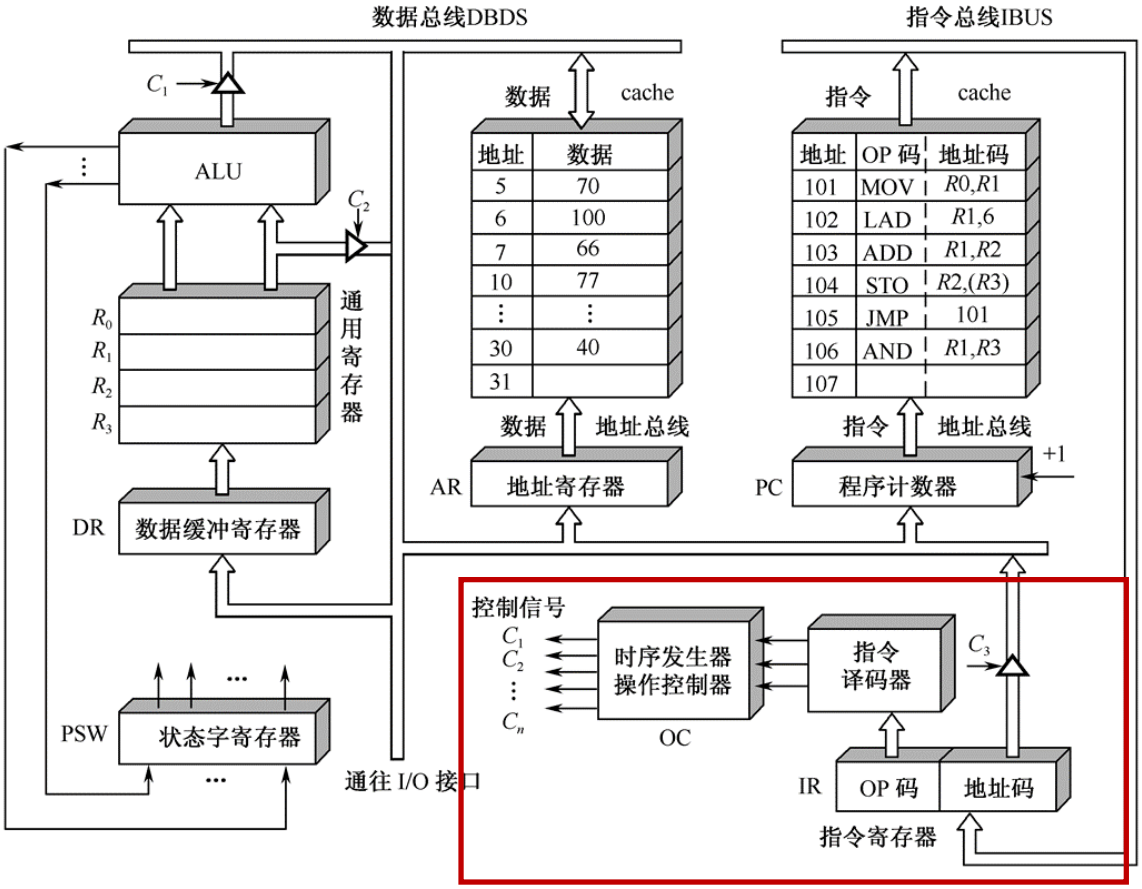


ALU$_o$

R_{3i}

本周教学内容

- CPU组成
  - 运算器
  - I-Cache/D-Cache
  - 控制器
- 本章内容
  - 指令执行
    - 指令周期
  - 时序信号
  - 操作控制器
    - 微程序控制器
  - 流水线设计
  - RISC CPU



第五章 中央处理器

- 流水线技术与性能分析
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险
- 典型CPU简介
  - 奔腾CPU
  - RISC CPU

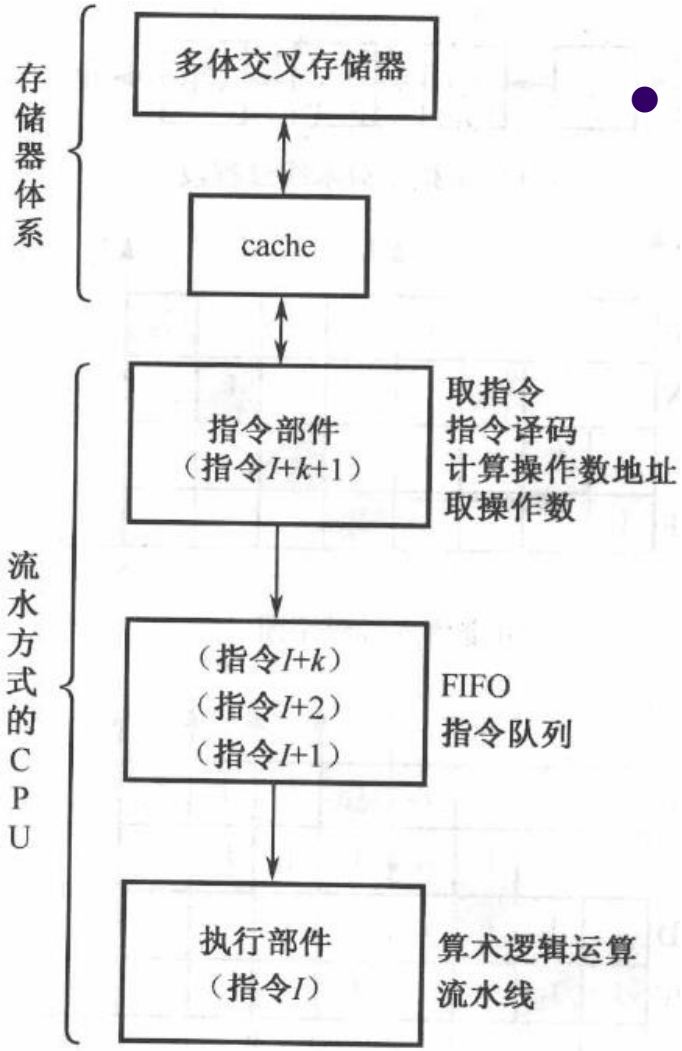
并行性

- 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

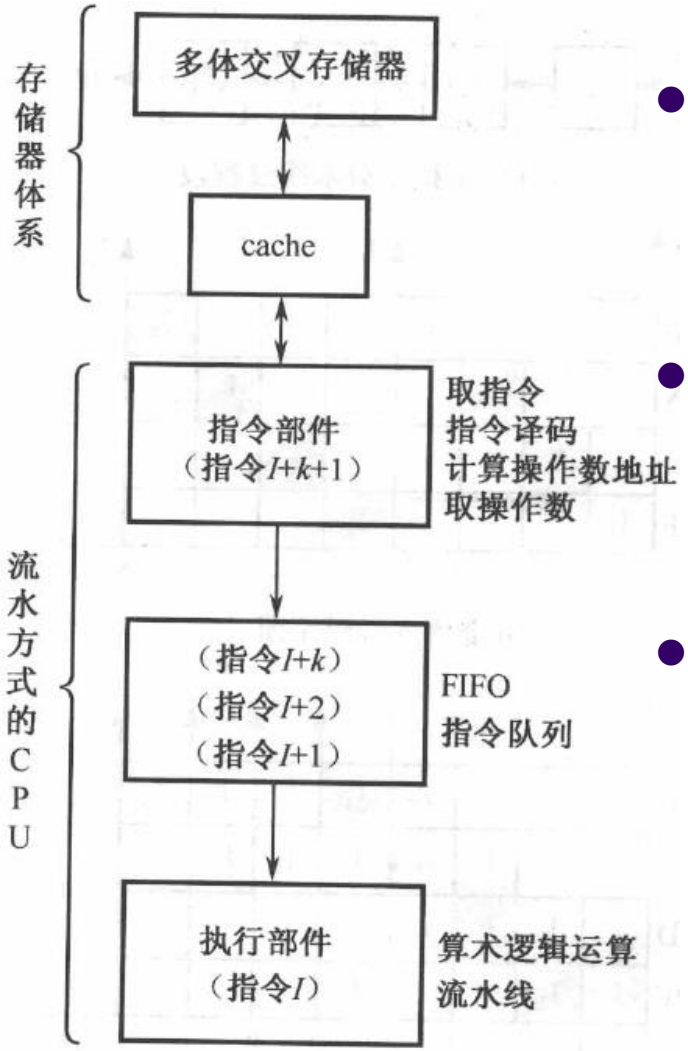
- 三种形式
  - 时间并行（重叠）
    - 流水线技术（浮点运算流水线、交叉体存储器）
    - 让多个处理过程在时间上相互错开，轮流使用同一套硬件设备的各个部件，以加快硬件周转而赢得速度，实现方式就是采用流水处理部件
  - 空间并行（资源重复）：增加冗余部件
    - 超标量技术
    - 它能真正的体现同时性，VLSI为其提供了技术保证
  - 时间+空间并行
    - Pentium中采用了超标量流水线技术

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

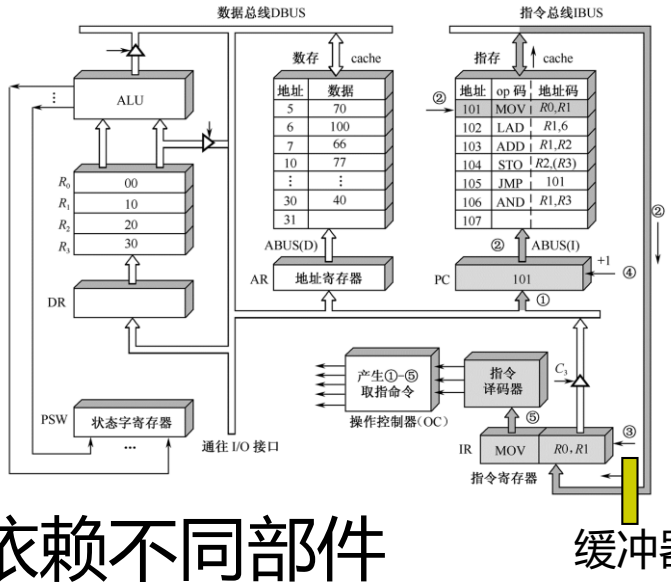
流水CPU的结构（2）

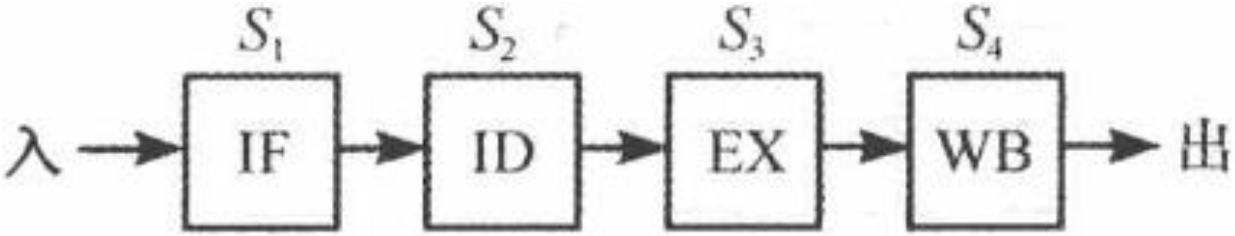


- 指令部件
  - 本身是流水线，包括取指、译码、计算操作数地址、取操作数等
- 指令队列
  - FIFO方式寄存器栈（队列）
  - 存放待执行指令
- 执行部件
  - 多个采用流水线方式构成的算术逻辑部件构成
  - 将定点运算部件和浮点运算部件分开

指令阶段划分

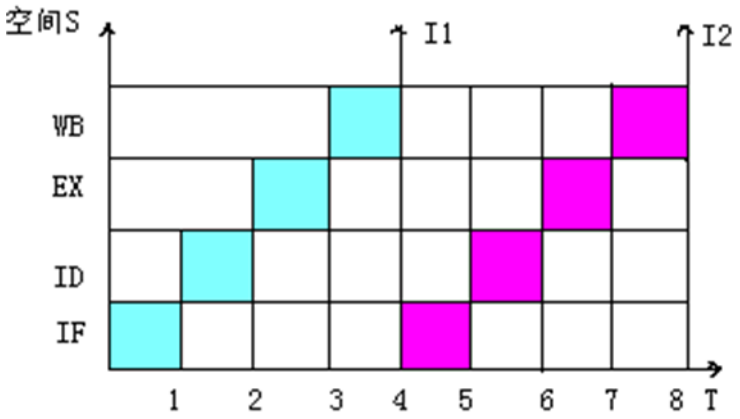
- 考虑一个四周期流水的指令划分（第一章）：
  - 取指：IF（Instruction Fetch）
  - 译码：ID（Instruction Decode）
  - 执行：EX（Execution）
  - 回写：WB（Write Back）
- 流水线基础
  - 取指、译码、执行、回写分别依赖不同部件
  - 可以并行执行（加缓冲、共享总线、Cache）



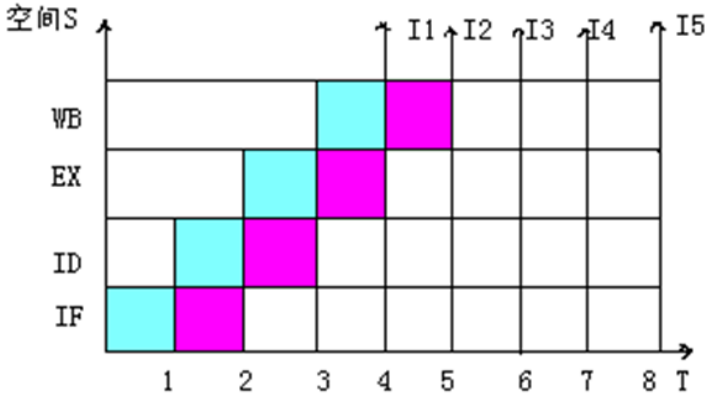


流水线时空图

- 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

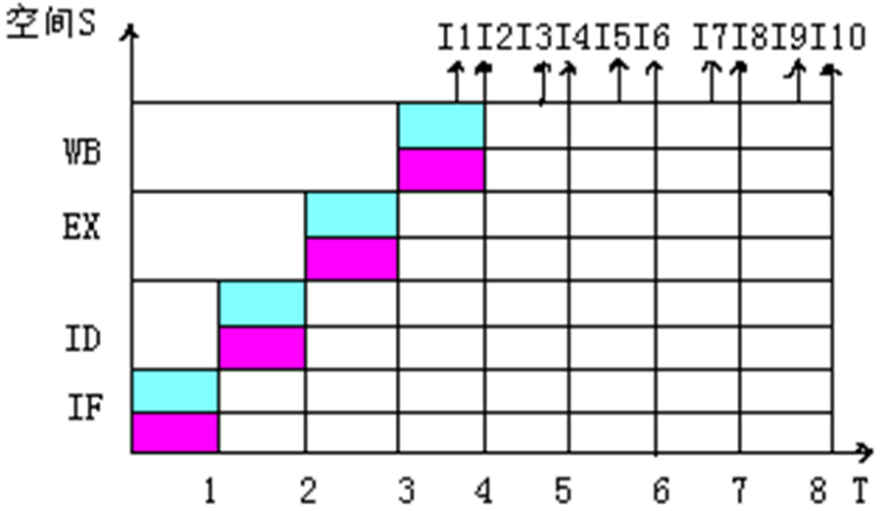


- 流水CPU
  - 四个阶段重叠执行
  - 流水线满载时，每个周期完成一条指令

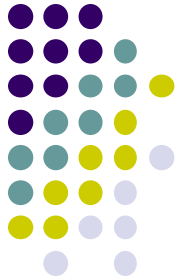


超标量流水线

- 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

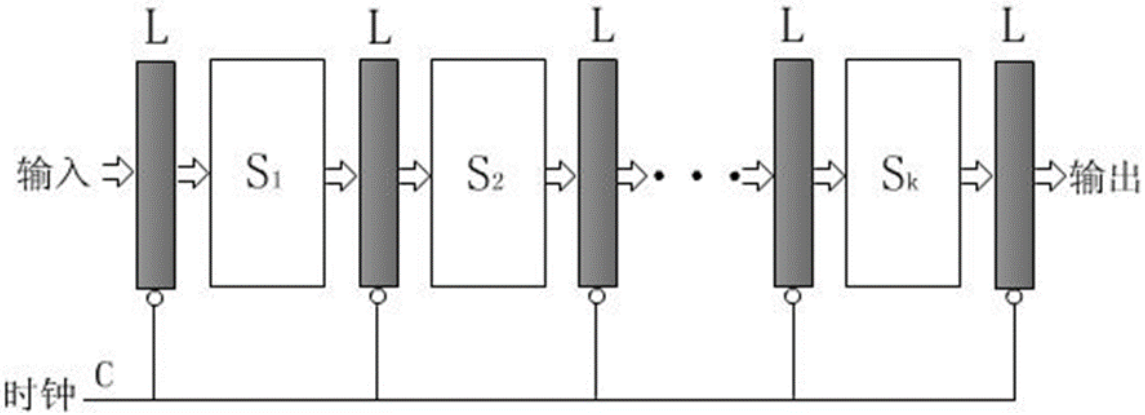


流水线技术原理



- 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
- 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
- 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）
- 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

流水线性能分析



- 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，$T = \sum \tau_i + K\tau_1$
- 无法通过无限划分流水技术提升CPU性能

单选题 1分

设置

此题未设置答案，请点击右侧设置按钮

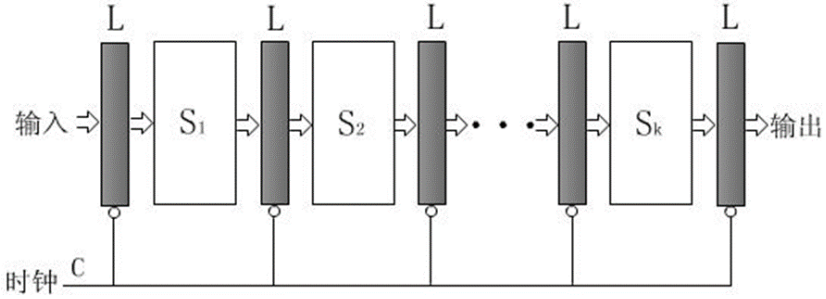
四级流水CPU，每级流水所需时间为200ps，缓冲寄存器延迟50ps
流水器满载时，完成相邻两条指令时间间隔？完成一条指令延时？

A 800ps, 1000ps

B 200ps, 800ps

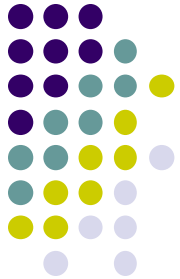
C 200ps, 1000ps

D 250ps, 1000ps



提交

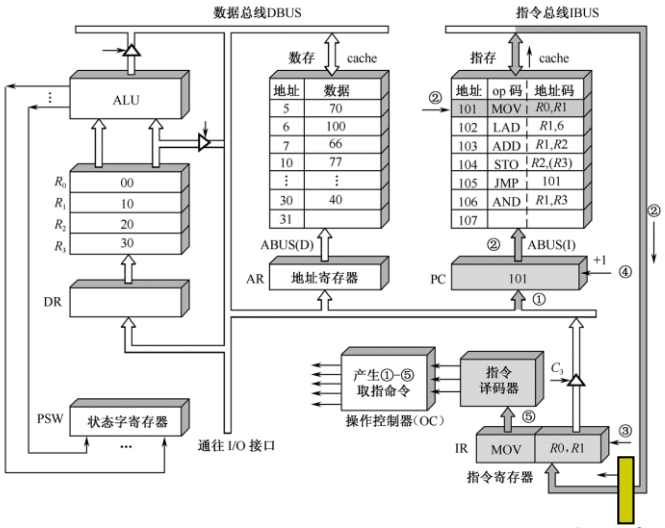
第五章 中央处理器

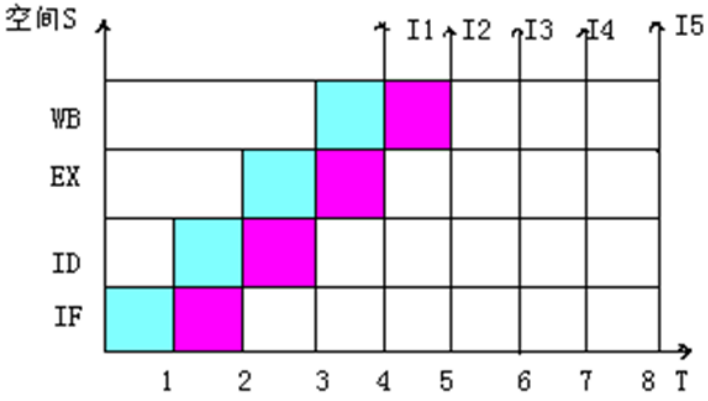


- 流水线技术与性能分析
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险
- 典型CPU简介
  - 奔腾CPU
  - RISC CPU

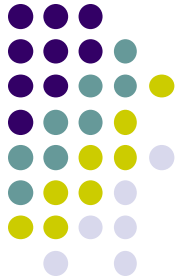
流水线中的主要问题

- 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





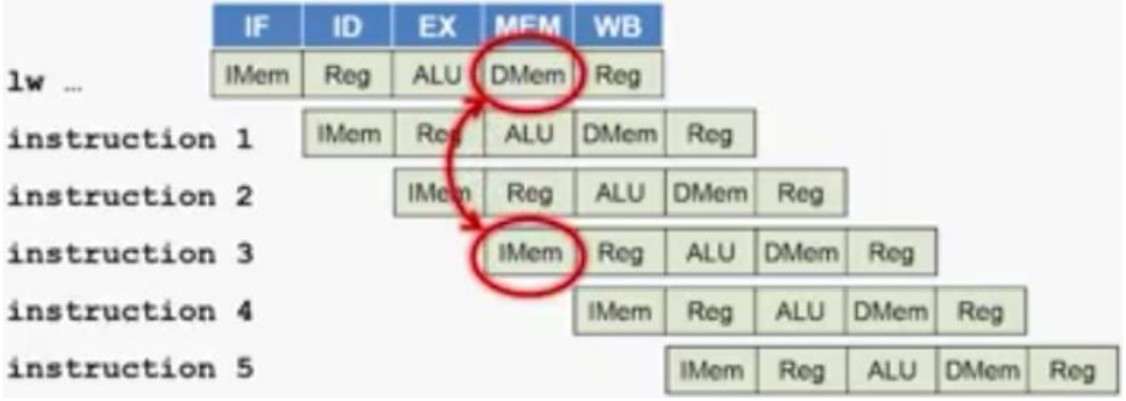
第五章 中央处理器



- 流水线技术与性能分析
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险
- 典型CPU简介
  - 奔腾CPU
  - RISC CPU

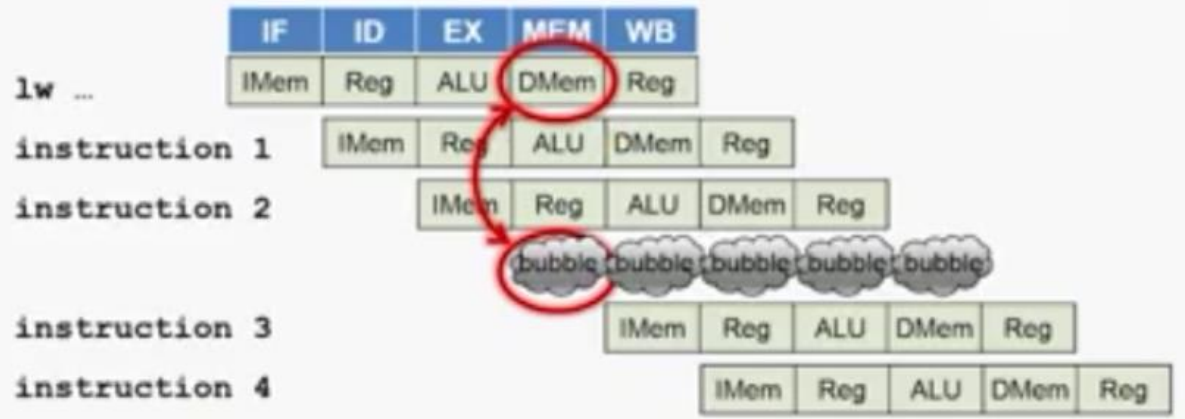
结构冒险

- 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



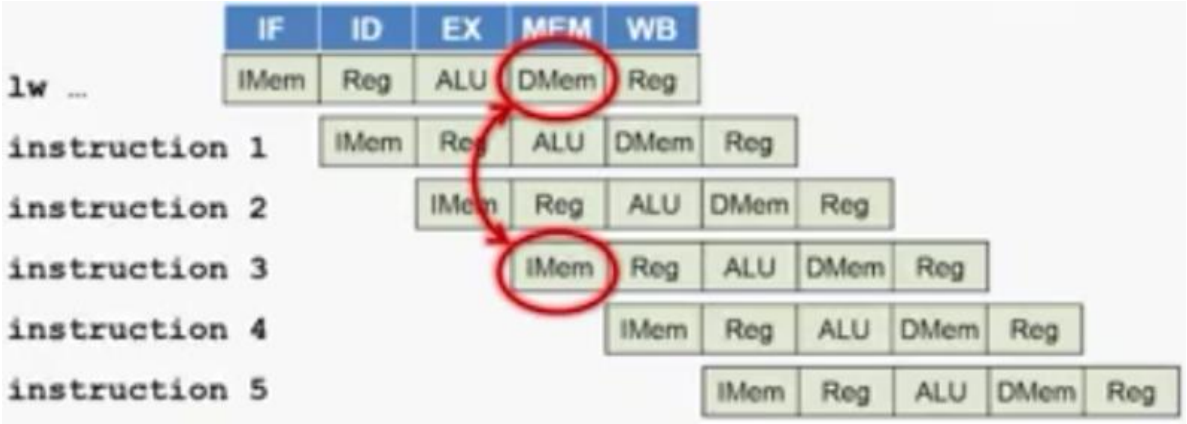
结构冒险解决方法

- 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

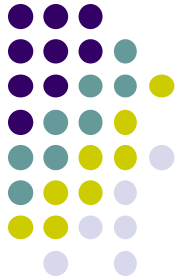


结构冒险解决方法（2）

- 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



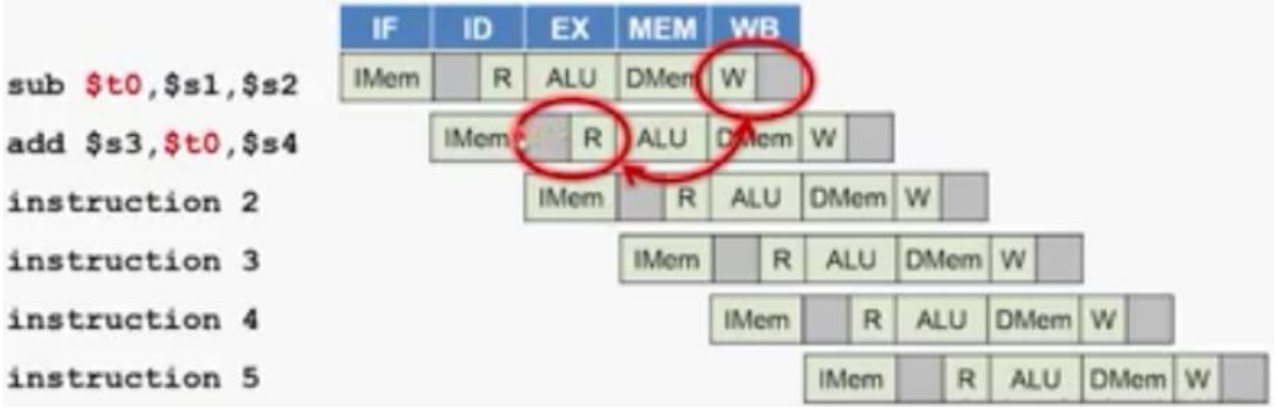
第五章 中央处理器



- 流水线技术与性能分析
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险
- 典型CPU简介
  - 奔腾CPU
  - RISC CPU

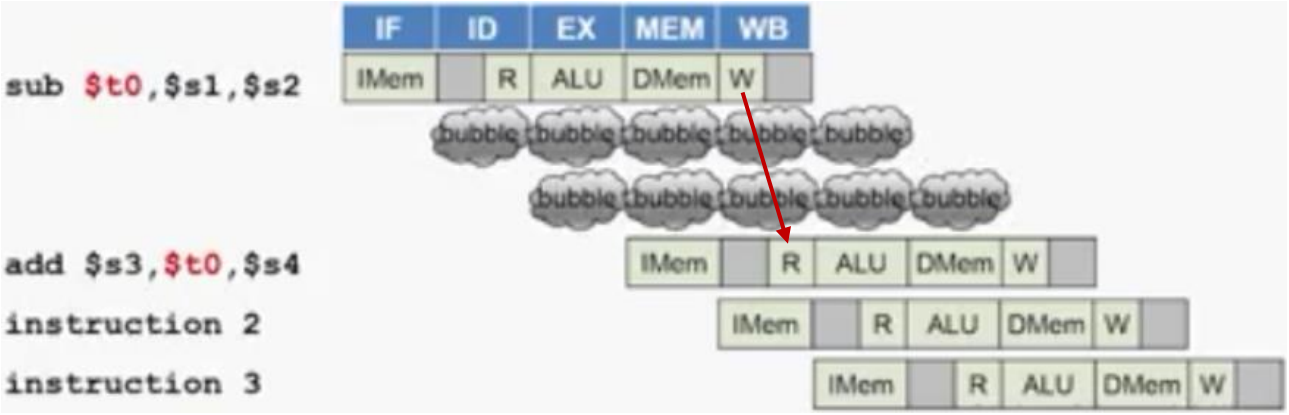
数据冒险

- 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)



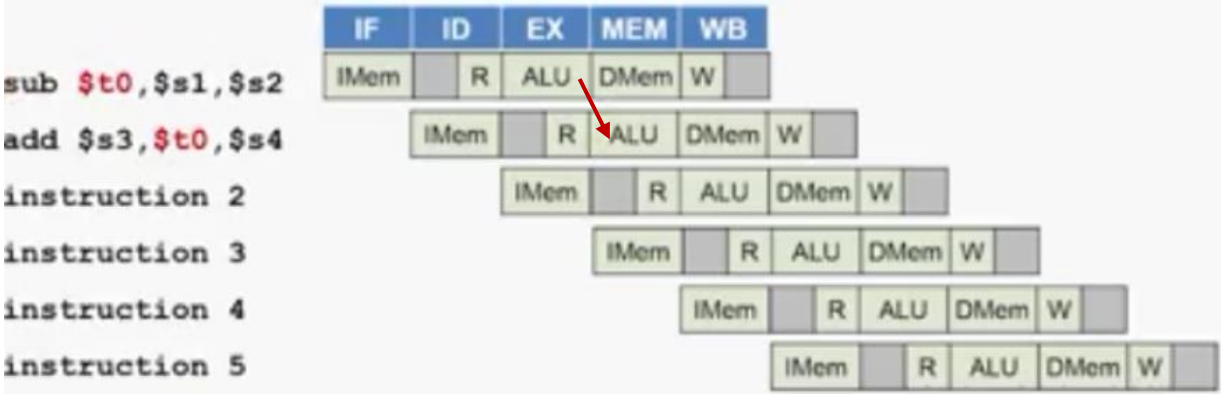
数据冒险解决方法（1）

- 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

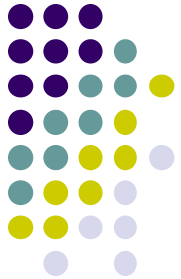


数据冒险解决方法（2）

- 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



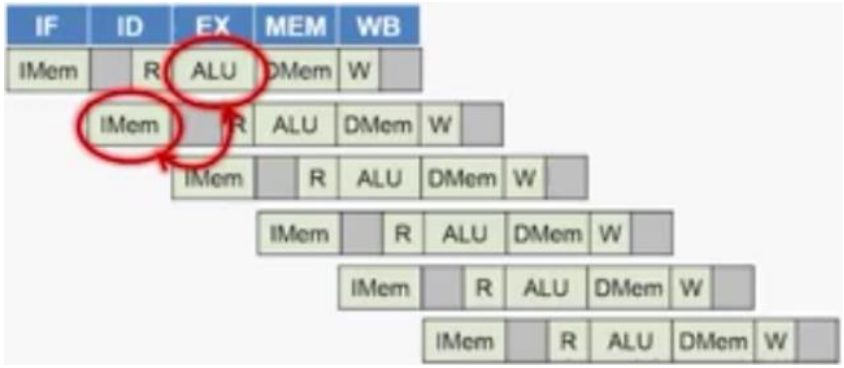
第五章 中央处理器

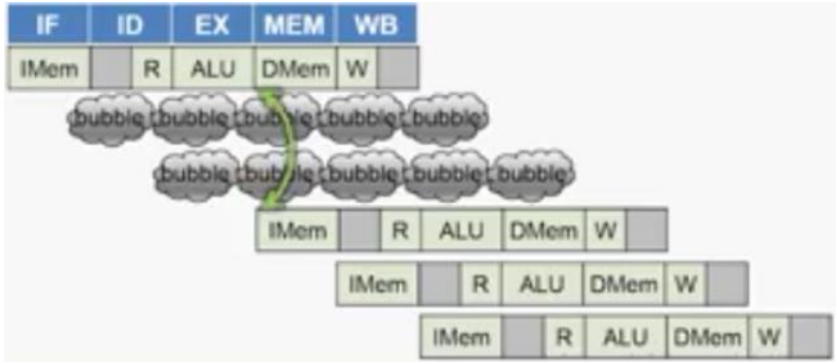


- 流水线技术与性能分析
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险
- 典型CPU简介
  - 奔腾CPU
  - RISC CPU

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

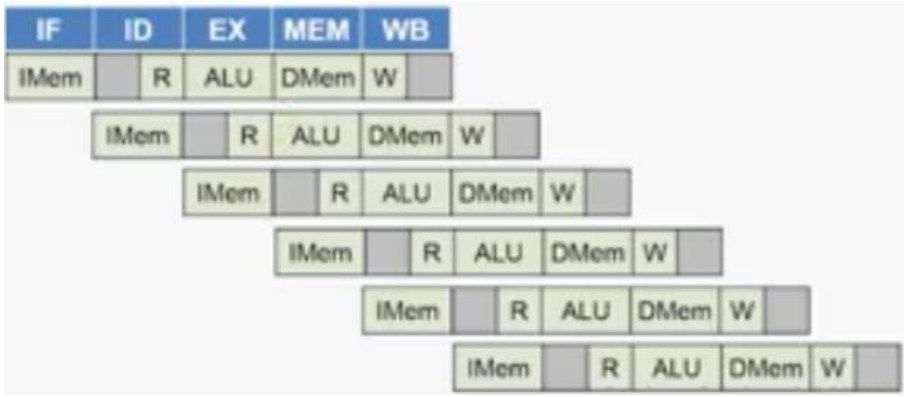
5级流水CPU方式如下，执行
ADD $t0, $s1, $s2; (s1)+(s2)→t0
ADD $s3, $t0, $s2; (t0)+(s2)→s3
是否会发生冒险，冒险方式为?

A 否

B 是；控制冒险

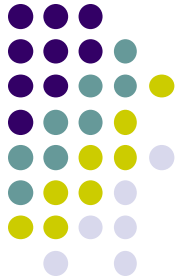
C 是；结构冒险

D 是；数据冒险（RAW）



提交

第五章 中央处理器



- 流水线技术与性能分析
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险
- 典型CPU简介
  - 奔腾CPU
  - RISC CPU

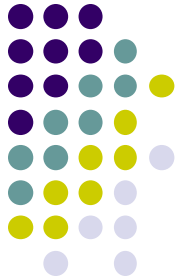
奔腾CPU

- Pentium CPU （第一代）
  - 1989年初0.8um工艺，310万晶体管
  - 有60M和66MHz外频两种版本
  - 5V电压，功耗20W
  - 超标量流水线结构
    - 486有一条流水线
    - Pentium有U和V两条指令流水线
      - U流水线可以执行所有的整数和浮点指令
      - V流水线可以执行简单的整数和FXCH浮点指令

奔腾CPU

- 双重分离式Cache，减少了等待和搬移数据时间
- 32位CPU，外部数据总线宽度为64位，外部地址总线宽度为36位
- 非固定长度指令格式，9种寻址方式，191条指令，兼具有RISC和CISC特性，不过我们还是将其看成CISC
- 提供了更加灵活的存储器寻址结构，可以支持传统的4k大小的页面，也可以支持4M大小的页面
- 动态转移预测技术

第五章 中央处理器



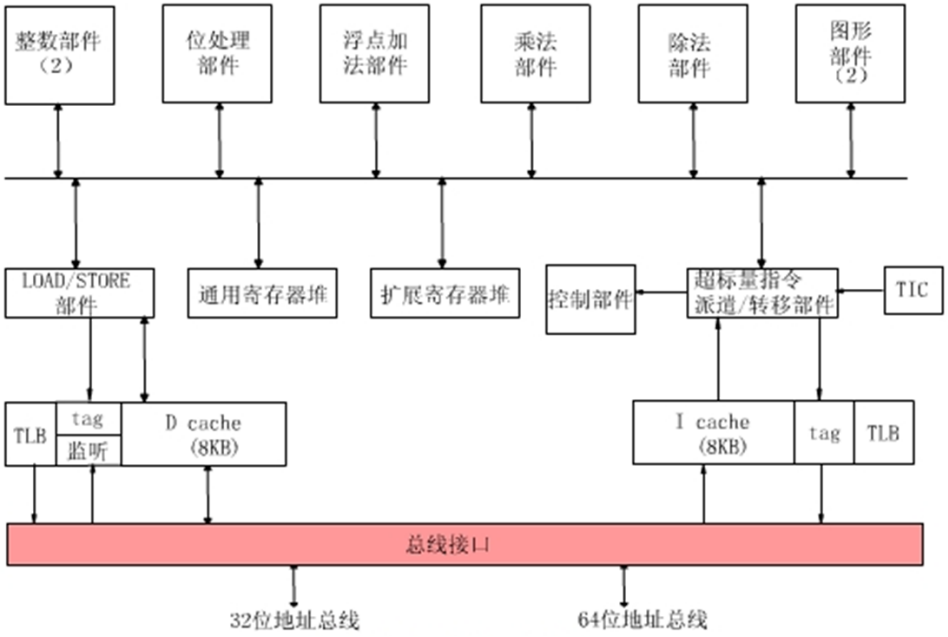
- 流水线技术与性能分析
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险
- 典型CPU简介
  - 奔腾CPU
  - RISC CPU

RISC特点

- 特点
  - （采用流水线技术）
  - 简单而统一格式的指令译码；
  - 大部分指令可以单周期执行
  - 只有LOAD/STORE可以访问存储器
  - 简单的寻址方式
  - 采用延迟转移技术
  - 采用LOAD延迟技术
  - 三地址指令格式
  - 较多的寄存器
  - 对称的指令格式
  - 其他

RISC CPU实例

- 实例 MC88110
  - CPU结构框图
    - 12个执行功能部件
    - 3个Cache（指令，数据和目标指令）
    - 两个寄存器堆（通用寄存器堆、扩展寄存器堆）
    - 六条80位宽的内部总线



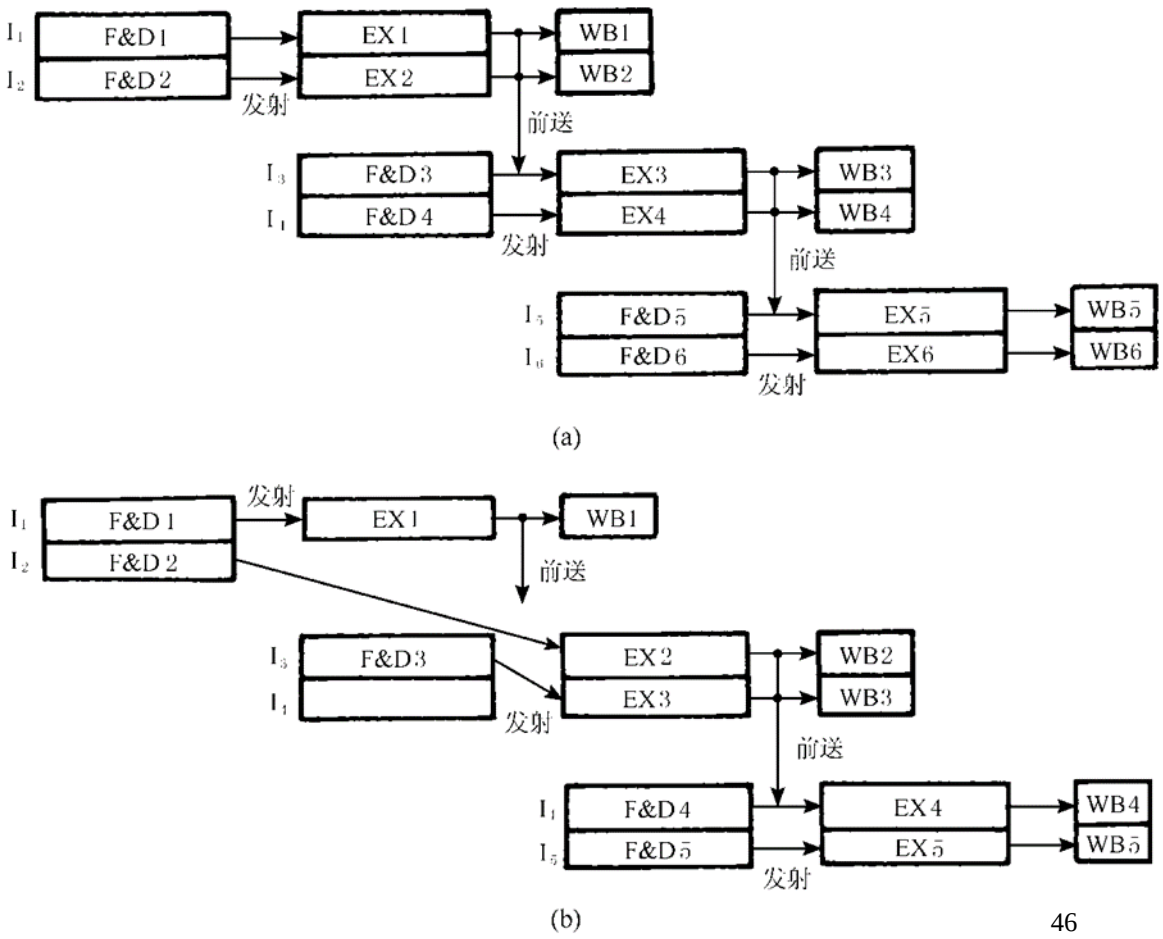
MC88110的指令流水线

- 超标量流水线CPU
  - F&D：取指和译码段需要一个时钟周期，
  - EX：执行段，大都只需要一个时钟周期，
  - WB：写回段，只需要时钟周期的一半
  - 采用了直接通路（Forwarding）技术

F&D	EX	WB

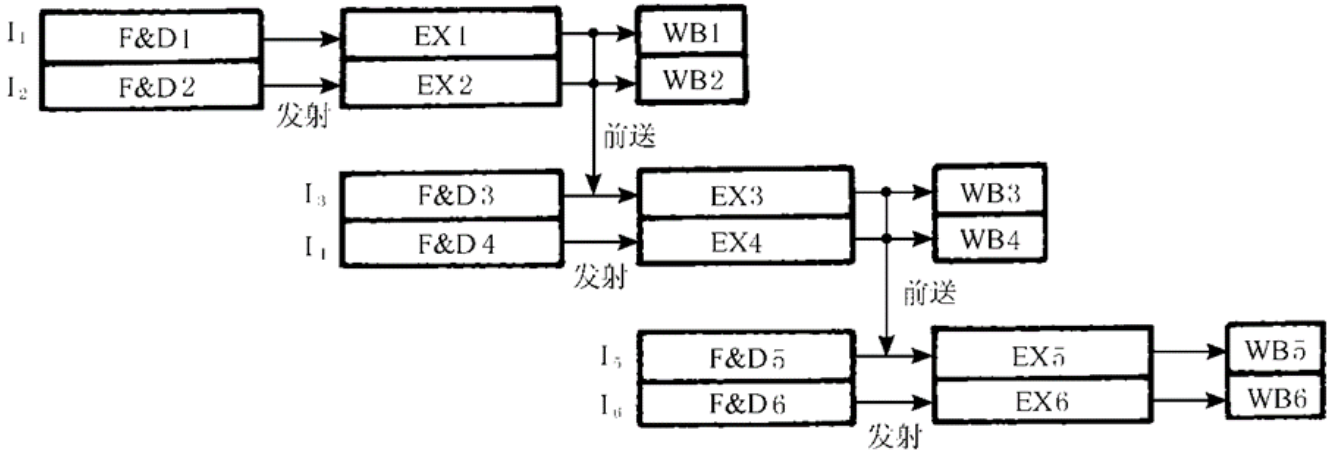
指令动态调度策略

- 指令动态调度策略
  - 按序发射
    - 取两条指令，配对发送
    - 一个周期可以有两条指令执行完毕



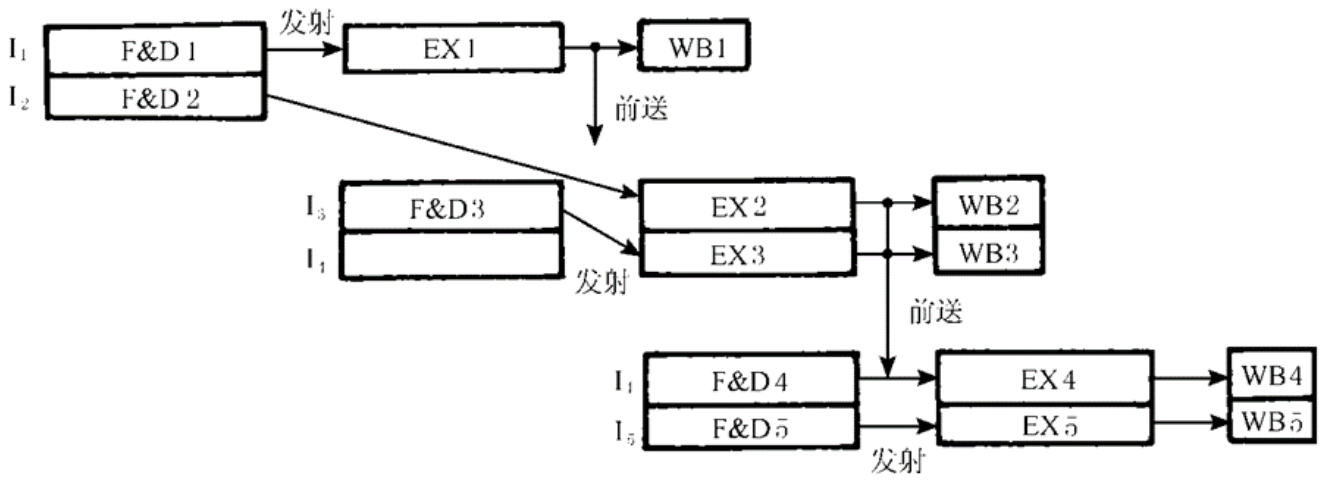
指令动态调度策略（1）

- 指令动态调度策略
  - 按序发射
    - 取两条指令，配对发送
    - 一个周期可以有两条指令执行完毕

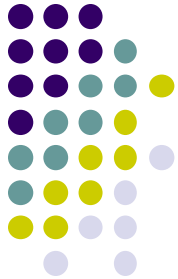


指令动态调度策略（2）

- 指令配对
  - 第一条指令由于资源相关或数据相关，则这两条指令都不发射
  - 若第一条指令能发射，第二条不能发射，只发射第1条指令到EX段，第二条指令等待并新取一条指令与之配对等待发射



指令动态调度策略（3）



- 几个问题：
  - 怎样判断能否发射呢?
    - 可以采用计分牌的方法
    - 位向量，表示寄存器等是否繁忙
    - 可以发射条件：指令所有相关寄存器记分牌已清除
  - 如何保证按序完成?
    - FIFO指令队列
  - 如何对待控制相关（转移指令）？
    - 采用延迟转移法
    - 目标指令cache法（TIC）

总结

- 流水线技术与性能分析
- 流水线冒险分析
  - 结构冒险
  - 数据冒险
  - 控制冒险
- 典型CPU简介
  - 奔腾CPU
  - RISC CPU