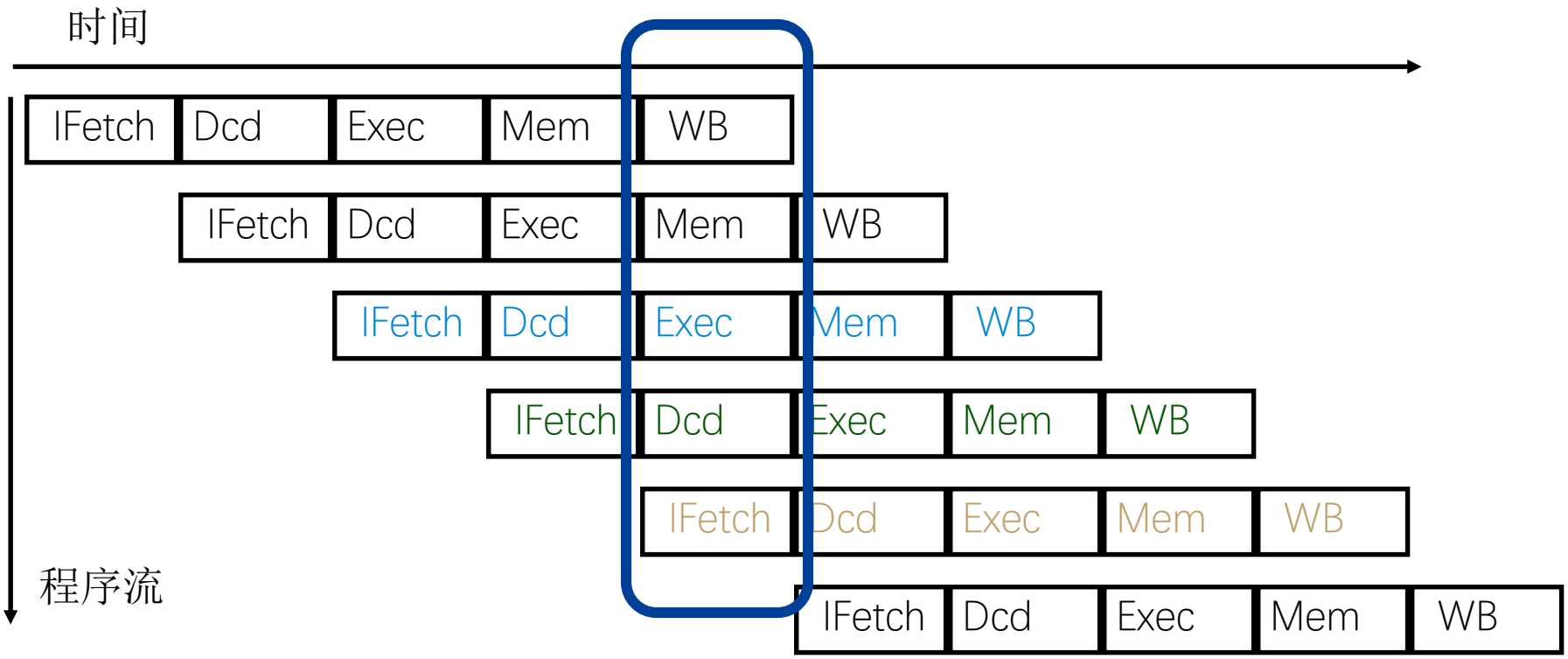
指令流水线的优化

回顾：五阶段指令流水线



理想情况：流水线不停顿

- 每一个周期完成一条指令
- CPI=1;
- CPI：完成一条指令所花的周期数

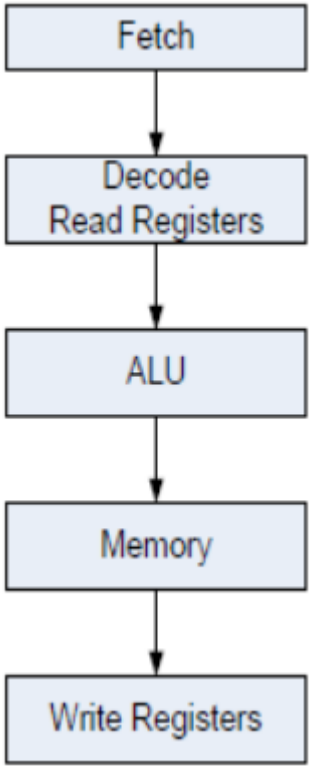
五阶段流水线处理器

优点：

- $CPI_{理想}=1$
- 简单，容易实现

缺点：

- 性能的上限：CPI=1
- 延迟高的指令难以流水化
  - 例如：乘法指令
- 紧耦合：
  - 一条指令停顿，导致后面的指令全部停顿



流水线处理器的性能

- CPU执行时间 =指令数目 x 平均每条指令所花的时钟周期（CPI）
  x 一个时钟周期长度

- $CPI_{ideal}=1$
- $CPI = CPI_{ideal} + CPI_{stall}$
- CPI_{stall} 发生的原因
  - 数据冒险
  - 结构冒险
  - 控制冒险
  - 访存延迟

流水线处理器的性能优化（1）

- CPU执行时间 =指令数目 x 平均每条指令所花的时钟周期（CPI）
  x 一个时钟周期长度
- 策略1：减少一个时钟周期的长度（提高时钟频率）
- 超级流水(superpipelining)
  - 增加流水线的段数，提升时钟频率、从而提高指令吞吐率

时钟周期=200ps + 50ps(流水段寄存器的延迟） = 250ps

时钟周期=100ps+ 50ps=150ps

流水线的深度

- 流水线的级数是越多越好吗？ 当然不!

时钟周期：200ps+50ps=250ps
单条指令的延迟：1250ps
流水段寄存器延迟所占比例：50ps / 250ps = 20%

时钟周期：100ps+50ps=150ps
单条指令的延迟：1500ps （增加了！）
流水段寄存器延迟所占比例：50ps / 150ps =33%

原因：流水线越深，流水段寄存器延迟就越多，一条指令的延迟就越大!

深度流水导致的问题

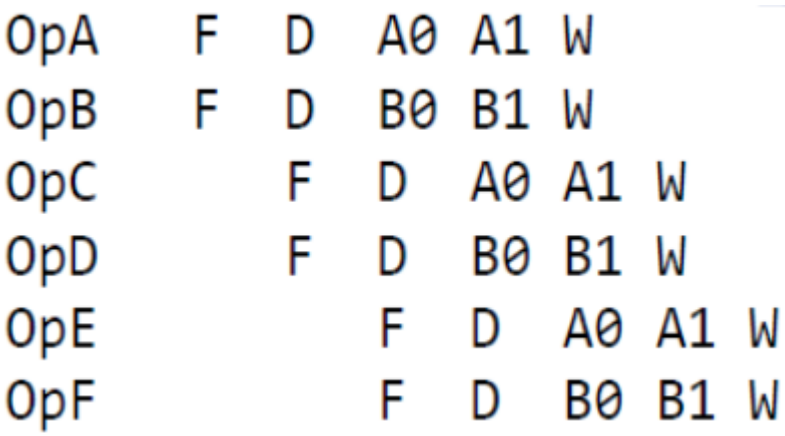
- 开销和复杂度增加
  - 流水段越多，前向通路越多
- 性能下降
  - 流水段寄存器个数增加，一条指令的延迟就越大
  - 重叠执行的指令越多 → 可能出现的相关性越多 → 停顿的可能性就越大
- 功耗
  - 时钟频率高，功耗越大

处理器流水线深度的变化

- 1986年，MIPS R2000: 5级
- 1988年，MIPS R3000 5级
- 1991年，MIPS R4000（64位）: 8级
- 1997年，ARM9: 5级
- 2002年，ARM11: 8级
- 2009年，Cortex-A8：13级
- 2011年，Cortex-A15：15级
- 2013年，Cortex-A57：15级
- 1993年，Pentium: 5级
- 1995年，Pentium Pro: 12级
- 2004年，Pemtium 4(Prescott)：**31级**
- 2006年，Core 2 Duo(Merom)：14级
- 2008年，Core i7(Nehalem)：16级
- 2013年，Core i7(Haswell)：14级

流水线处理器的性能优化（2）

- CPU执行时间 =指令数目 x 平均每条指令所花的时钟周期（CPI）
  x 一个时钟周期长度
- 策略2：进一步减小CPI
- 多发射：让多条指令在流水线中并行执行
  - 一次取出并执行多条指令



- 双发射流水线
- 一个周期能发射两条指令，完成两条指令
- CPI_{ideal} =0.5

多发射

- Multiple Issue Processor(多发射处理器)
- 利用指令级并行性（Instruction Level Parallism: ILP ）获得 CPI < 1
  - static multiple issue (静态多发射：编译时确定多发射)
    - 编译器在编译时，决定哪些指令可以同时发射
  - dynamic multiple issue (动态多发射)
    - 超标量处理器（ superscalar processors ）
    - 执行时确定哪些指令并行发射
      - 可以是按序（ in-order ）执行的
      - 也可以是乱序（ out-of-order ）的超标量处理器

小结

谢谢！