西安电子科技大学
人工智能学院

计算机组成与体系结构

第7章 流水线技术与指令级并行

本章第5次课重点

- 多发射处理器
  - ✓ 超标量处理器
  - ✓ 超长指令字处理器
  - ✓ 超流水处理器
- 指令级并行概念

7.6.1 多发射处理器概念

- 多发射CPU分为：
  - ✓ 超标量处理器(superscalar processor)：在一个时钟周期同时发送多条指令
    - 静态调度：按序执行
    - 动态调度：乱序执行
  - ✓ 超流水处理器(superpipelining processor)：在一个时钟周期分时发送多条指令
  - ✓ 超标量超流水处理器
  - ✓ 超长指令字(very long instruction word, VLIW)处理器
    - 利用编译器实现静态调度
    - Intel、HP：IA-64，显式并行指令计算(explicitly parallel instruction computing，EPIC)
- 多发射处理器的发射宽度或(并行)度(degree)：每时钟周期可以发射的指令数。

7.6.1 多发射处理器概念

- 在多发射处理器中并行地处理指令需要做3项工作：
  - ✓ 检查指令间的相关性，以确定哪些指令可以组合在一起用于并行执行；
  - ✓ 将指令分配（dispatch）给硬件功能单元；
  - ✓ 确定多个指令（或放在一个单字中）的启动时刻。

表7.7 并行处理中硬件及编译器的作用

	指令成组	分配功能单元	启动
动态超标量	硬件	硬件	硬件
静态超标量，EPIC	编译器	硬件	硬件
动态VLIW	编译器	编译器	硬件
VLIW	编译器	编译器	编译器

超标量处理器

7.6.2 超标量处理器

- 为了充分利用度为m的超标量处理器，m条指令必须是可并行执行的，否则流水线将停顿。
- 超标量处理器芯片含有多个独立的执行单元，每个单元采用流水结构。
- 采用乱序发射、重命名技术。
- 每指令时钟数*CPI*实际低于1，通常 *IPC*＝2～6 。

7.6.2 超标量处理器

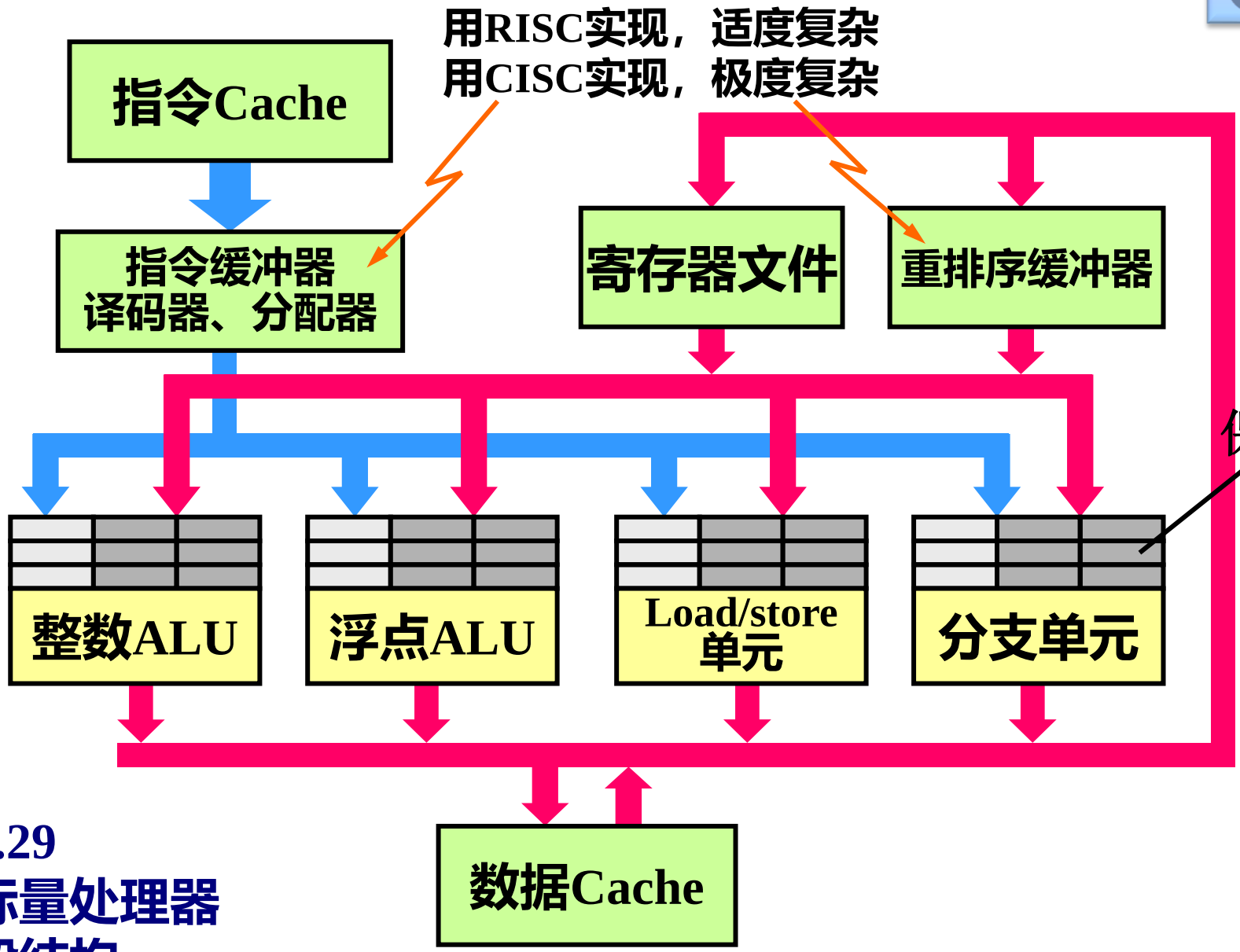
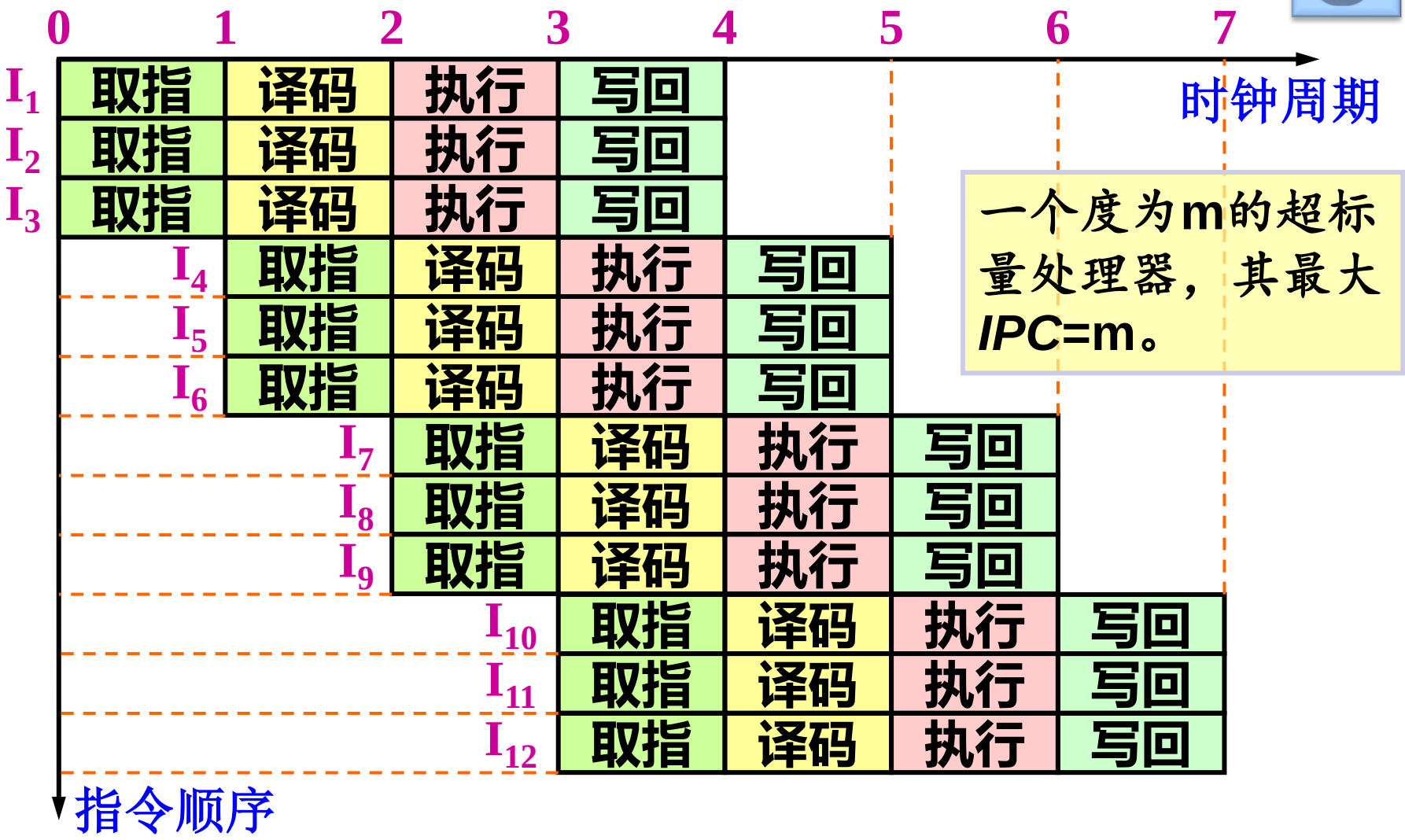


图7.29
超标量处理器
一般结构

7.6.2 超标量处理器

- 超标量处理器的基本执行过程：
  - 从指令Cache中获取多条指令并进行相关检查与分支预测，经过静态或动态调度，对指令重新排序；
  - 将不同类型的指令分配到相应的功能流水线上，由发射单元同时启动在不同功能流水线保留站中的指令开始执行。
  - 各流水线执行结果被送入重排序缓冲器，最终提交的是按原程序顺序排序的指令执行结果。
- 超标量实现需要如下逻辑部件的支持：
  - 同时取多条指令的逻辑；
  - 确定包括寄存器值的真相关逻辑；
  - 传递数据的机构；
  - 并行启动多指令的机构
  - 用于多指令并行执行的资源；
  - 以正确顺序提交处理结果的机构。

7.6.2 超标量处理器



一个度为m的超标量处理器，其最大*IPC*=m。

图7.30 超标量处理器（度m=3）时空图

7.6.2 超标量处理器

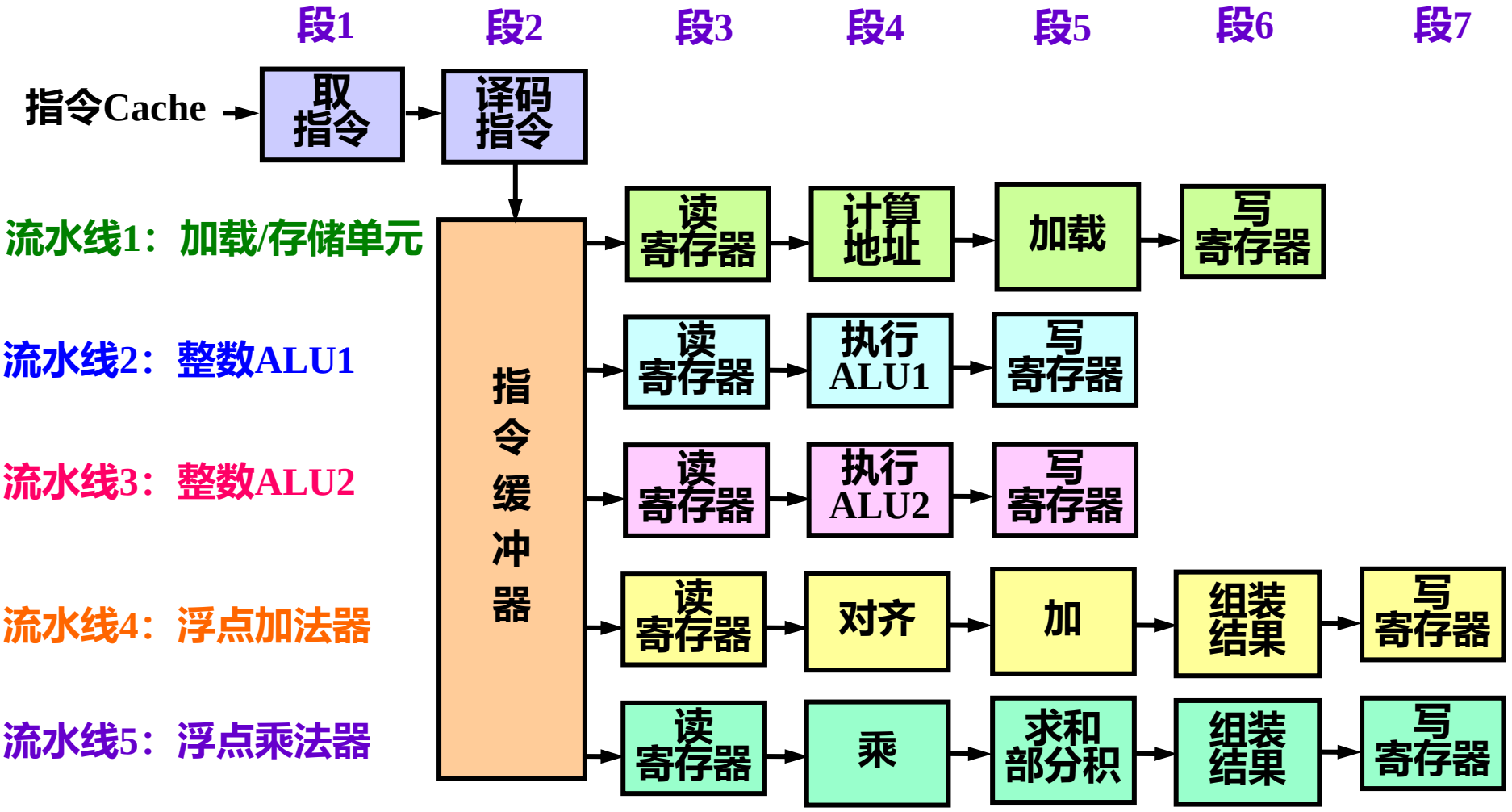


图7.31 SGI/MIPS R10000 的指令流水线

超长指令字处理器

VLIW

7.6.3 超长指令字处理器

比较

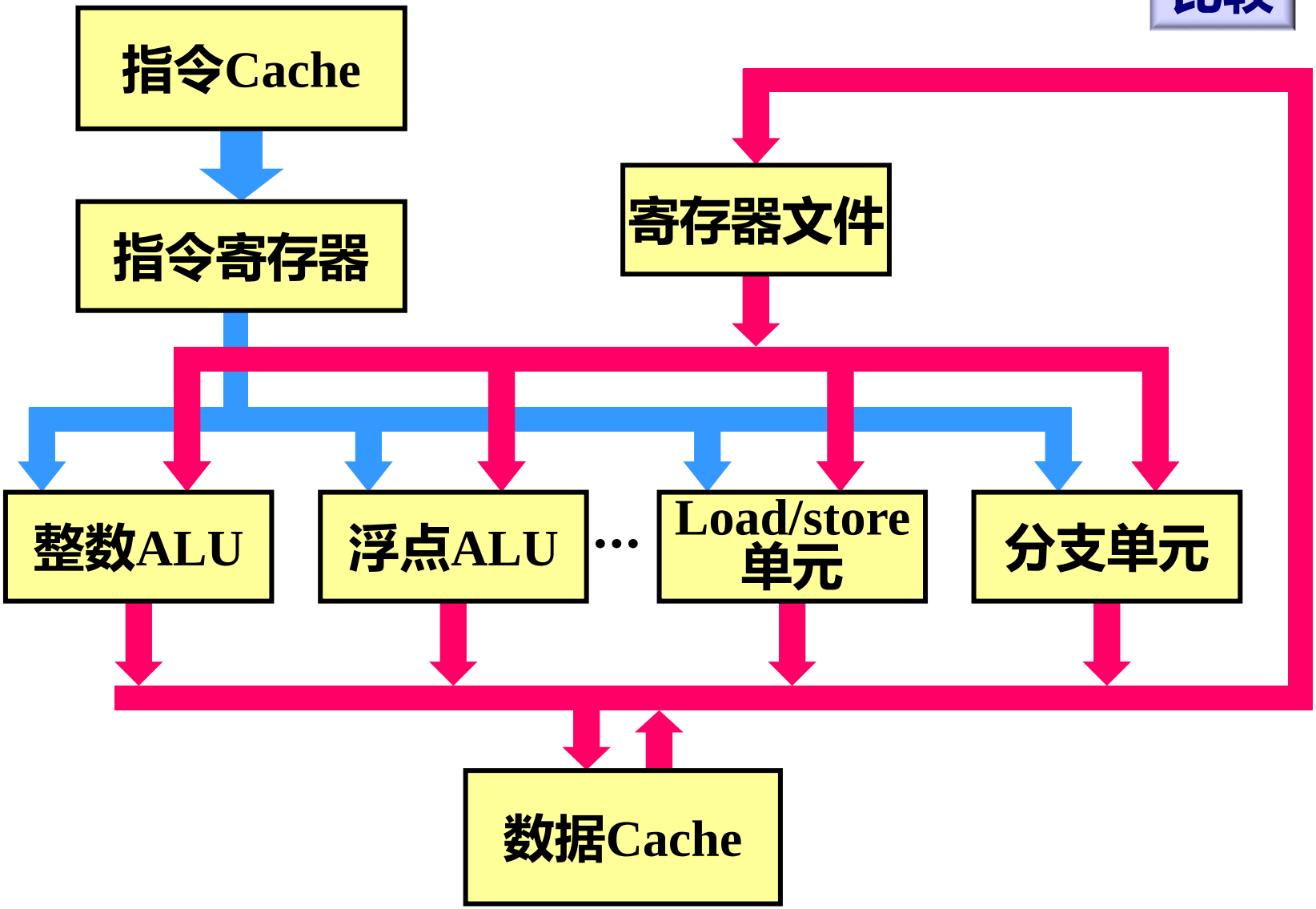


图7.32 VLIW处理器一般结构

7.6.3 超长指令字处理器

- VLIW与超标量处理器结构的不同：
  - VLIW处理器从指令Cache每次取得一条很长的指令字（即VLIW）存入指令寄存器，该字由几个可以并行执行的原始指令组成，每个原始指令在VLIW中所占字段称为一个指令槽或操作槽（slot），每个槽与功能单元一一对应。指令寄存器将VLIW中各原始指令同时派发到各功能单元中开始并行执行。
  - VLIW处理器结构及控制逻辑比超标量处理器简单得多：没有复杂的重排序缓冲器和译码、分配逻辑，功能单元中也可以不设置保留站。

Load/Store	浮点加	浮点乘	分支	…	整数ALU

图7.33（a）VLIW处理器指令格式

7.6.3 超长指令字处理器

- 如何生成VLIW指令：编译器
  - ✓ 对源程序进行相关检查、分支预测、指令调度。
  - ✓ 将一组不相关、可并行执行、可使尽可能多的功能单元处于忙状态的原始指令按约定的指令槽、依代码顺序组装在一条VLIW指令中。
- 典型的VLIW处理器具有数百位的指令长度。
- 为了简化指令译码及处理过程，VLIW指令一般采用固定格式。

Load/Store	浮点加	浮点乘	分支	…	整数ALU

7.6.3 超长指令字处理器

比较

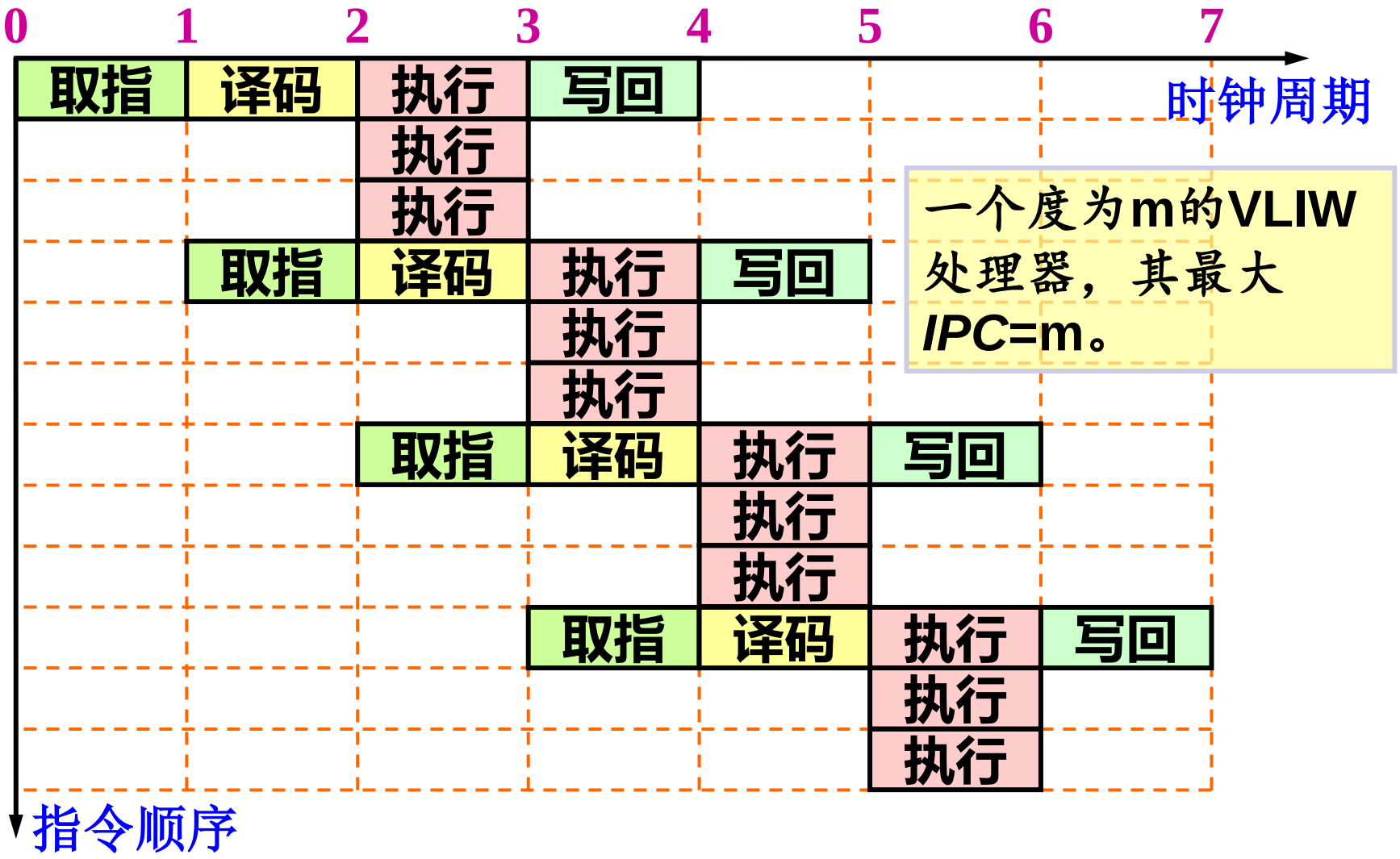


图7.33（b）VLIW处理器流水线时空图（度m=3）

7.6.3 超长指令字处理器

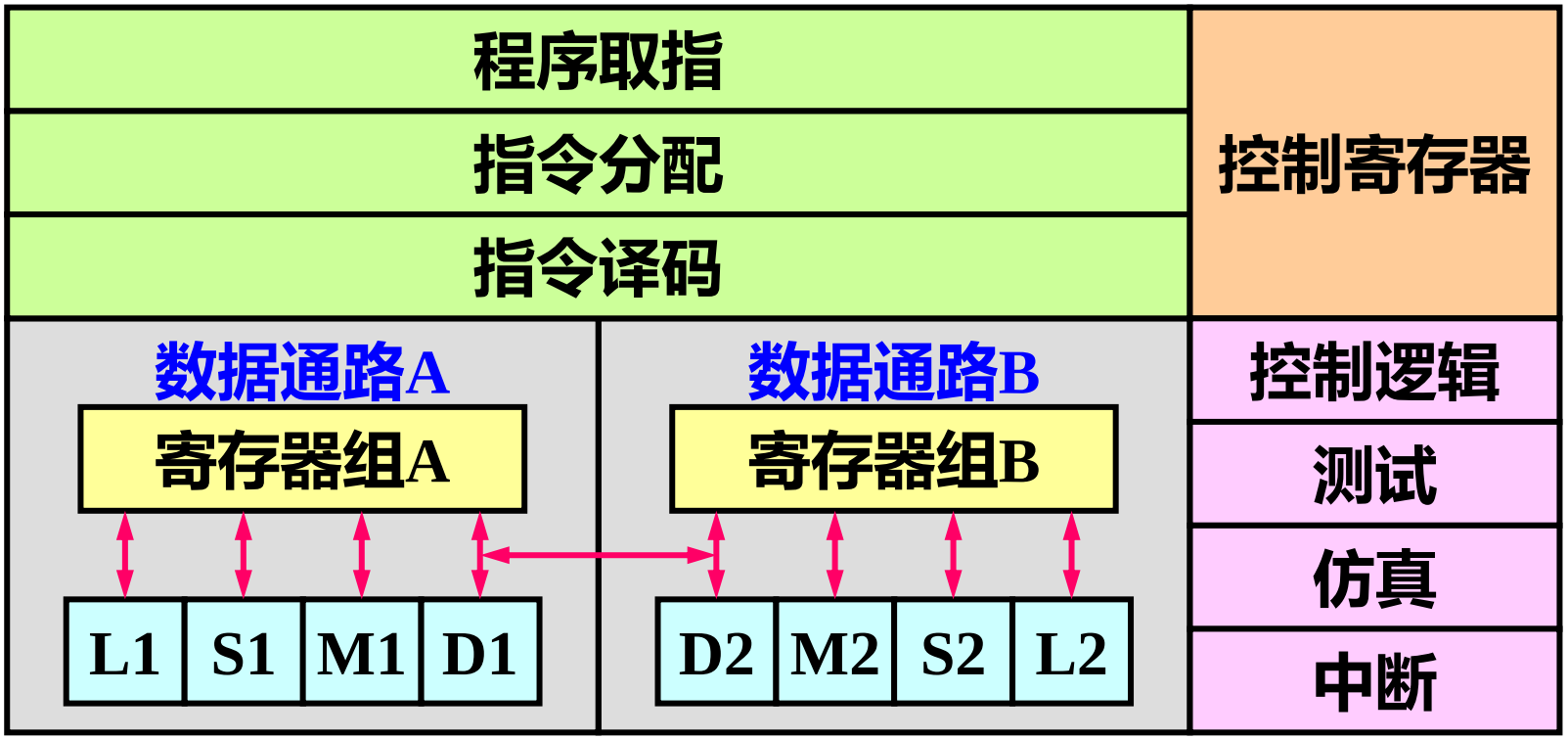


图7.34 TMS320C6200 CPU内部结构

7.6.3 超长指令字处理器

L单元：浮点ALU，同时完成40位整数ALU/比较、位计数、规格化操作

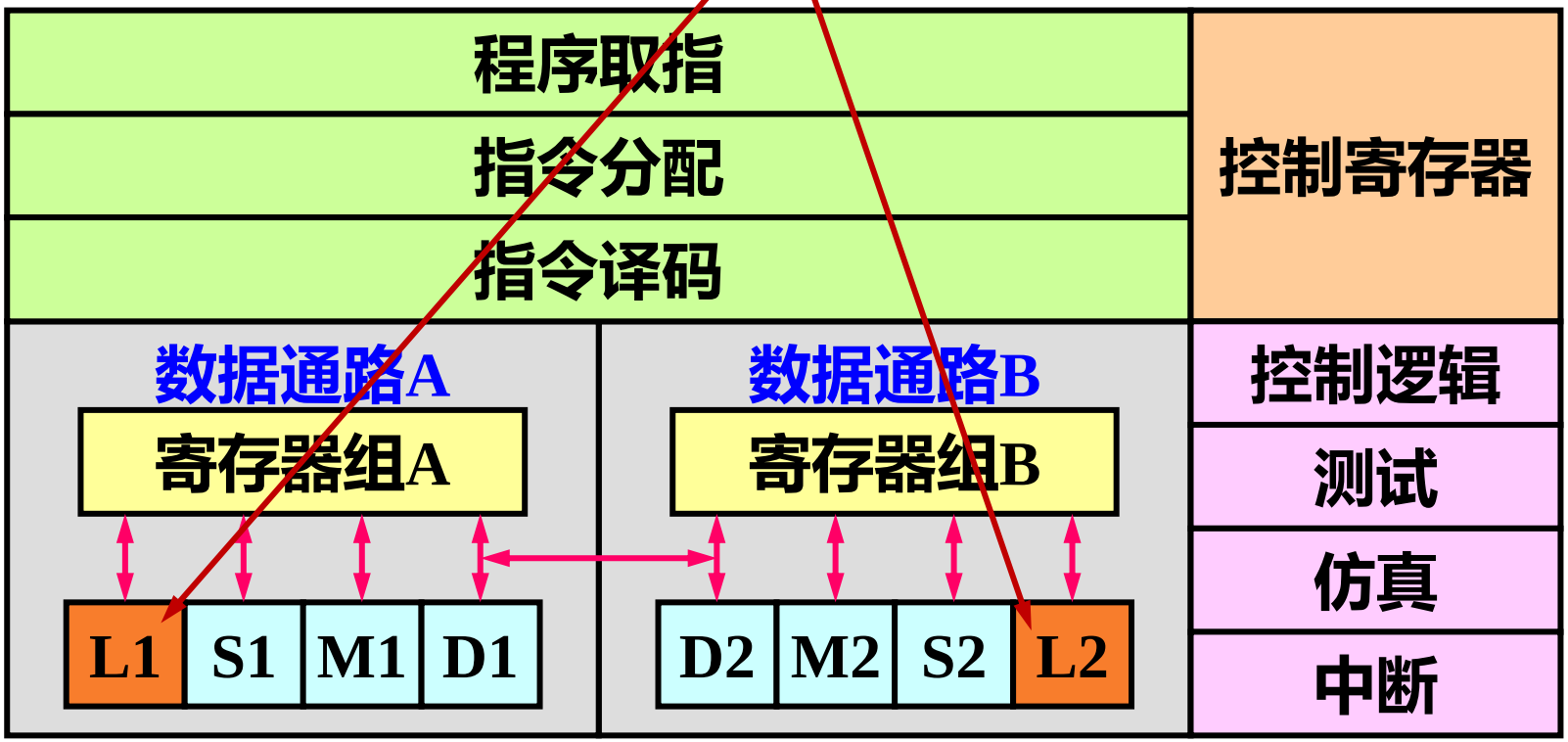


图7.34 TMS320C6200 CPU内部结构

7.6.3 超长指令字处理器

S单元：浮点辅助单元，同时完成32位ALU/40位移位、位域、分支操作

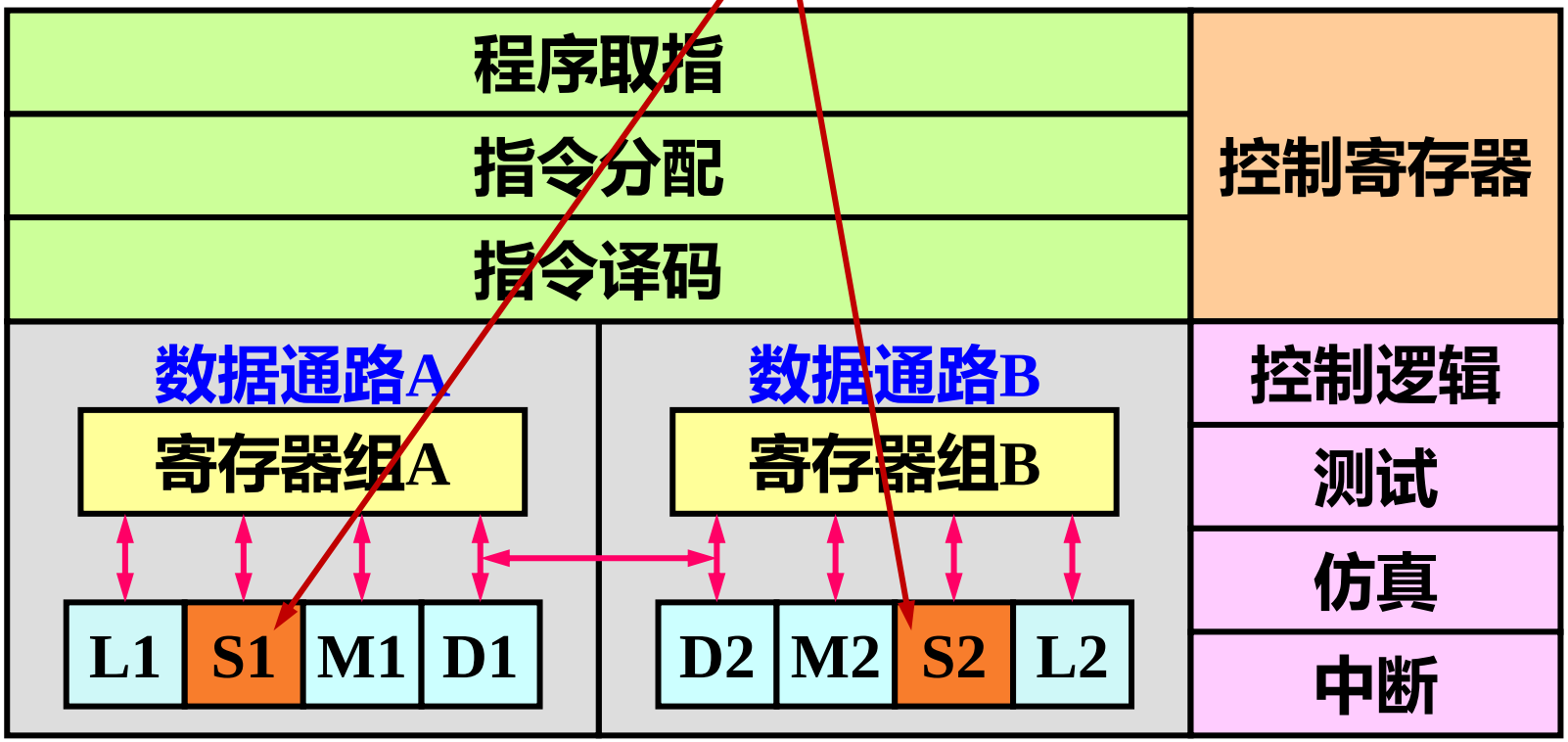


图7.34 TMS320C6200 CPU内部结构

7.6.3 超长指令字处理器

M单元：浮点乘法器，同时完成
16×16到32位的整数乘法

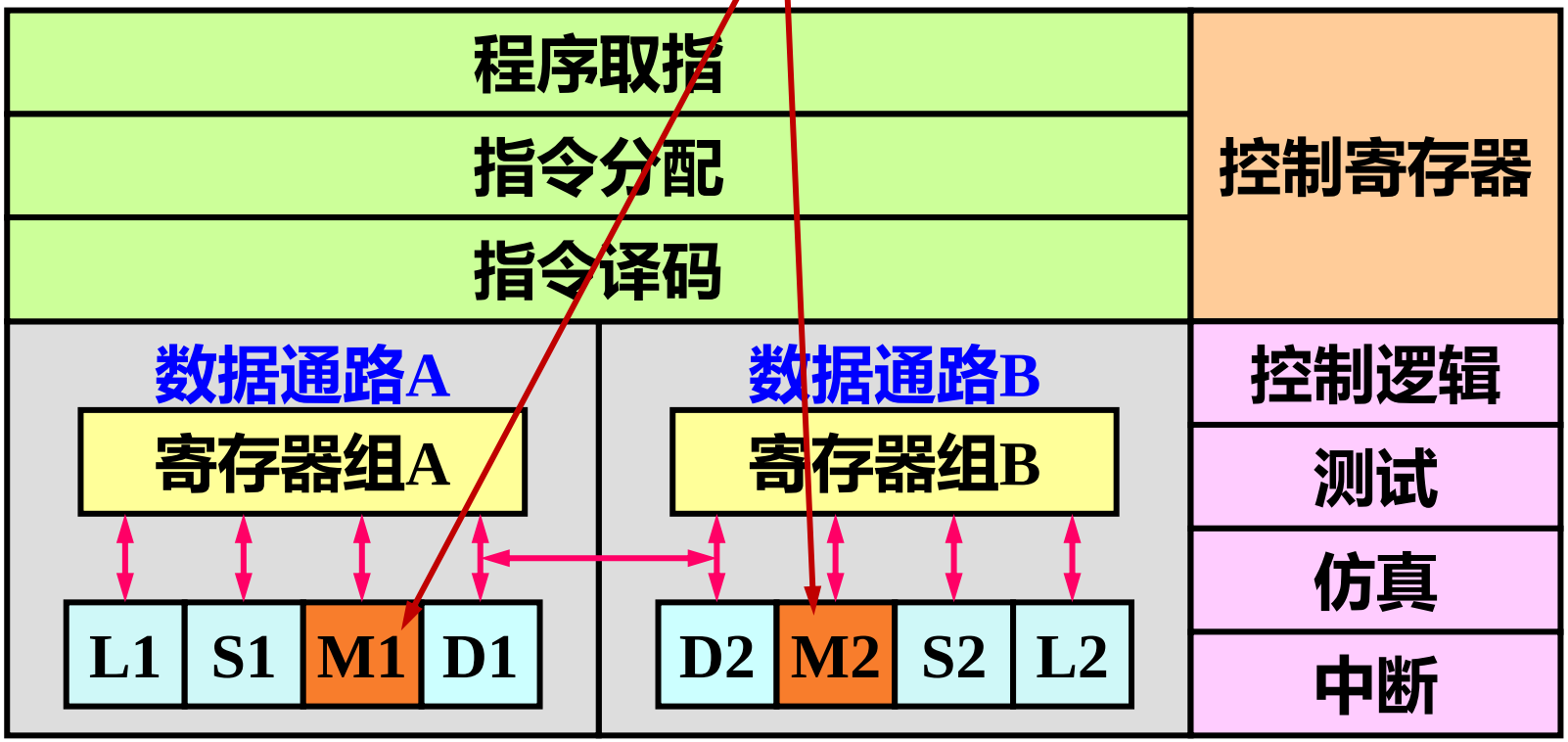


图7.34 TMS320C6200 CPU内部结构

7.6.3 超长指令字处理器

D单元：64位load部件，同时完成8/16/32位的load/store、32位加减、地址计算的操作

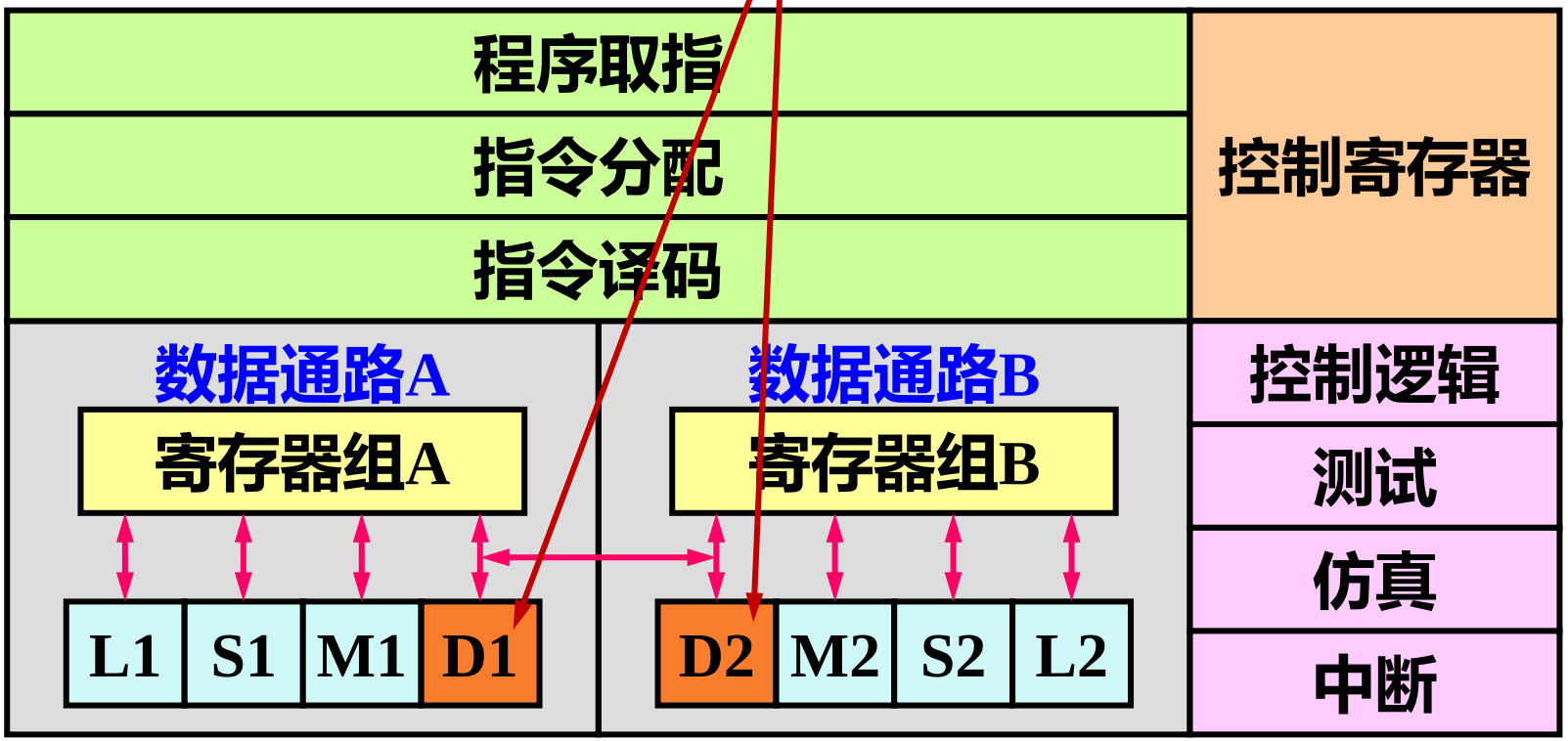


图7.34 TMS320C6200 CPU内部结构

7.6.3 超长指令字处理器

单指令字长32位，8条指令组成一个指令包(VLIW)，总字长为256位

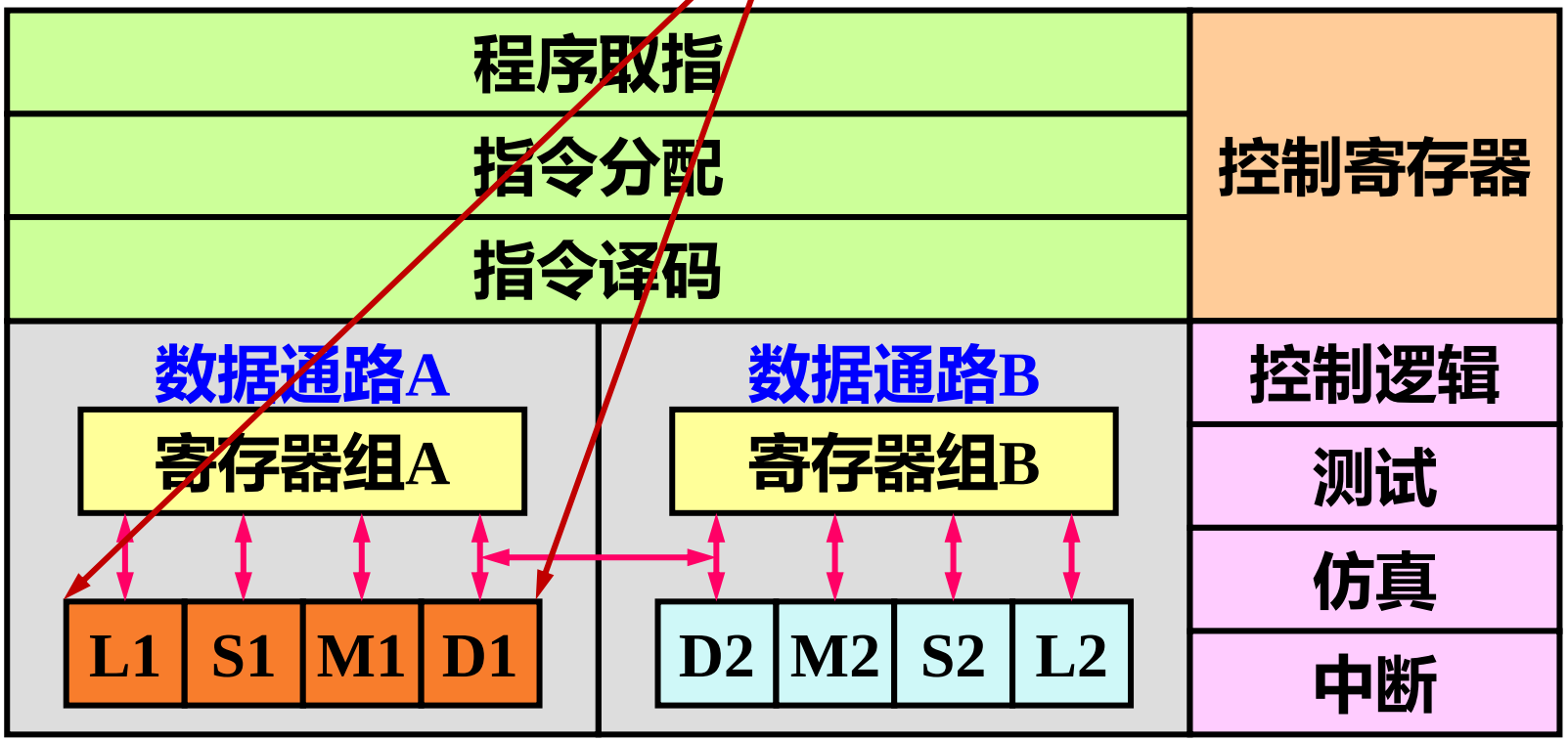


图7.34 TMS320C6200 CPU内部结构

7.6.3 超长指令字处理器

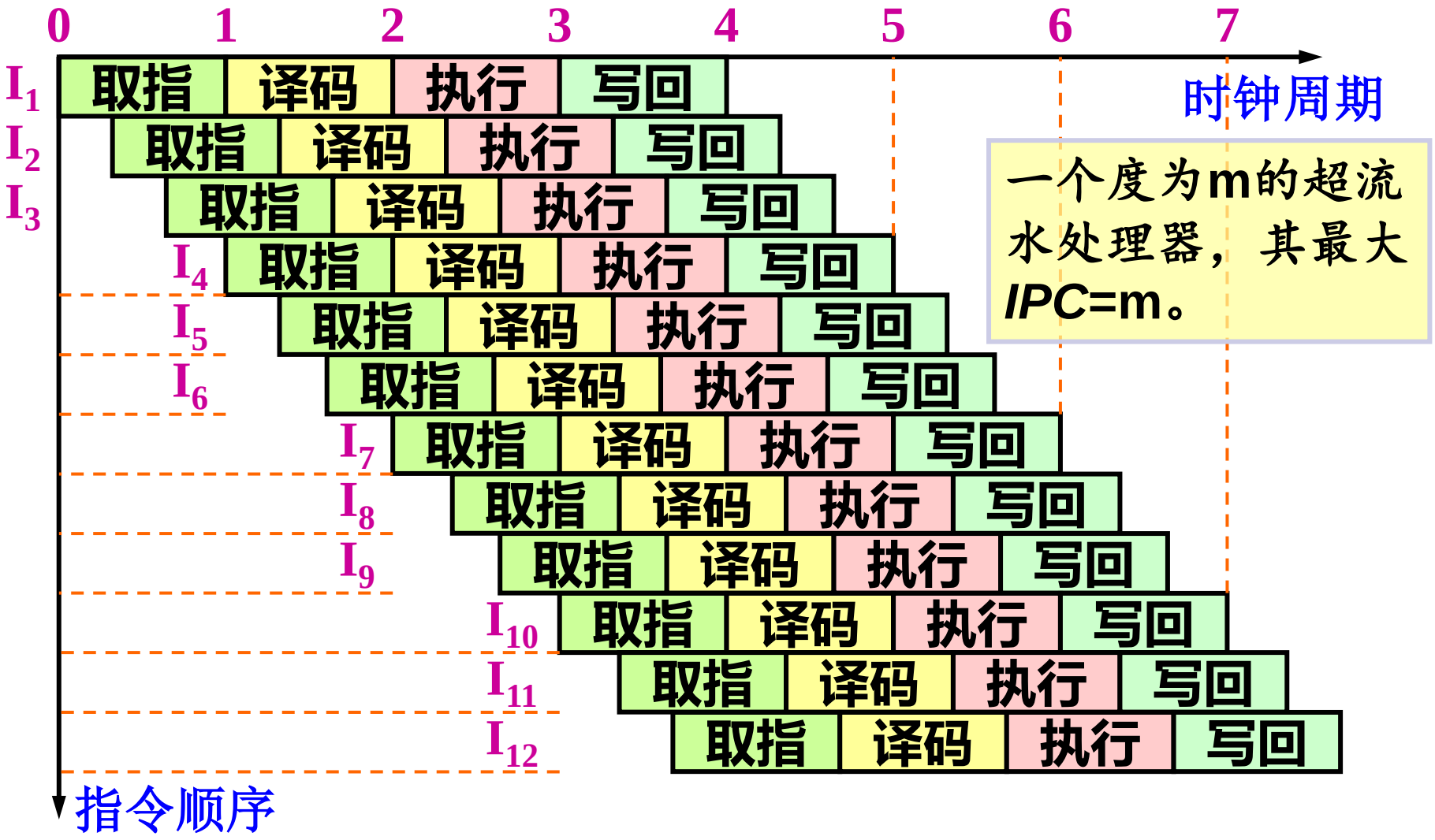
表7.8 VLIW与CISC、RISC的比较

结构特点	CISC	RISC	VLIW
指令长度	可变	固定，通常32位	固定，通常数百位
指令格式	字段放置多样化	规则，一致的字段放置	规则，一致的字段放置
指令语义	从简单变化到复杂，每条指令包含多个非独立操作	几乎总是一个简单操作	多个简单、独立的操作
寄存器	少量，有时专用的	许多，通用的	许多，通用的
涉及存储器	在许多不同类型的指令中与操作捆绑	不与操作捆绑，如load/store结构	不与操作捆绑，如load/store结构
硬件设计焦点	开发微码实现	开发单流水线实现，无微码	开发多流水线实现，无微码，无复杂派发逻辑

超流水处理器

7.6.4 超流水处理器

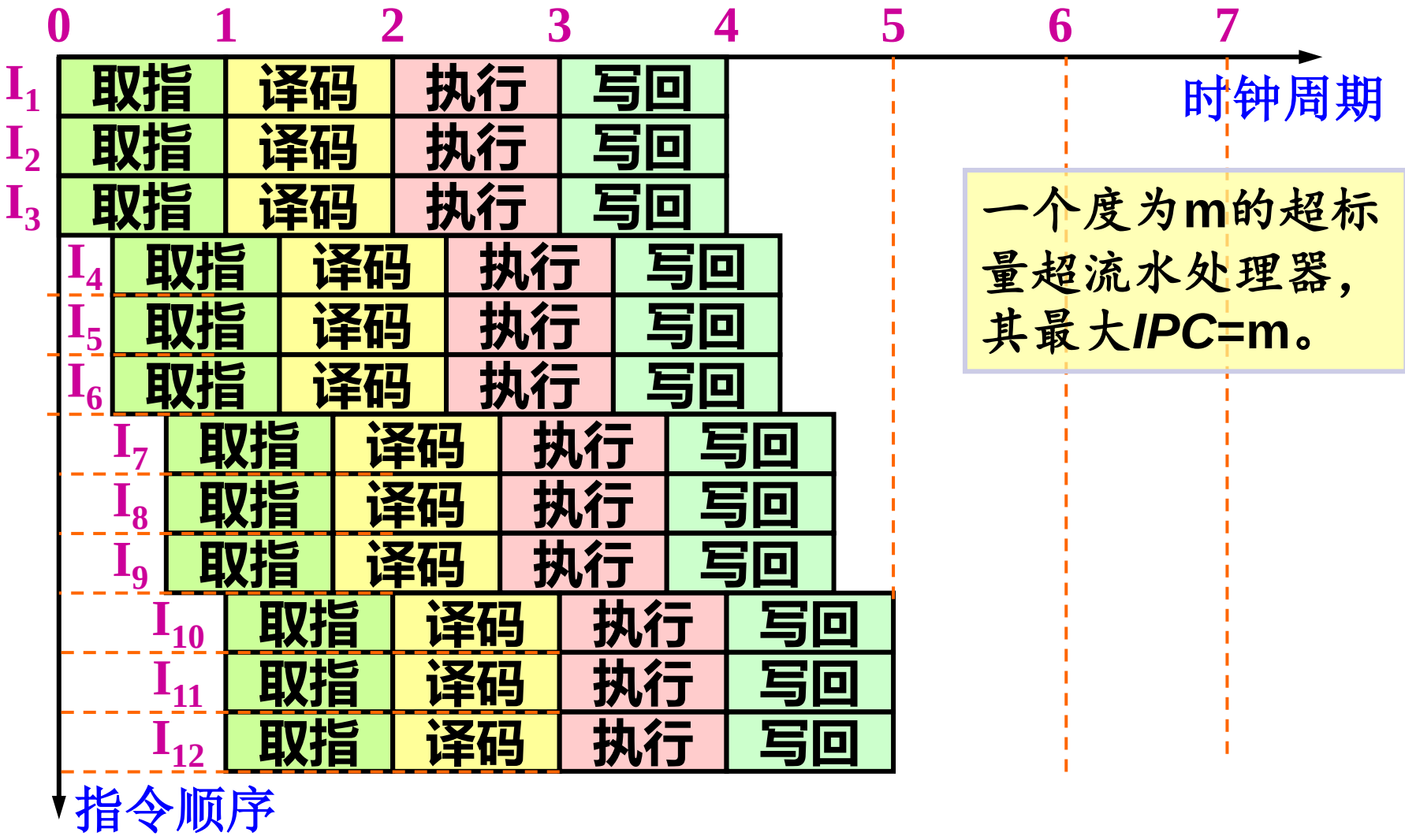
比较



一个度为m的超流水处理器，其最大*IPC*=m。

超流水处理器（度m=3）时空图

7.6.4 超标量超流水处理器



一个度为m的超标量超流水处理器，其最大*IPC*=m。

超标量超流水处理器（度m=3×3）时空图

7.6.4 多发射处理器的限制

- 超标量结构
  - ✓ 控制逻辑复杂
  - ✓ 多指令译码、发射机制实现困难
  - ✓ ILP度一般不超过8
- VLIW结构
  - ✓ 需要强大的编译技术支持；可利用的并行机制有限
  - ✓ 为了避免相关，有时需要在VLIW指令槽中插入空操作指令，而较低的槽利用率会浪费宝贵的处理器资源
  - ✓ 如何有效地设计VLIW指令格式及编码是设计中的难题
- 解决：将指令级并行上升到线程级并行——多核处理器
  - ✓ 目前可以替代并超越超标量处理器和超长指令字处理器的最佳选择

指令级并行概念

7.7 指令级并行概念

- 当指令不相关时，它们在流水线中重叠执行，这种指令间潜在的并行性称为指令级并行(Instruction Level Parallelism, ILP）。
- 实现指令级并行的关键技术是流水线技术。
- 开发指令级并行的方法：
  - ✓ 依赖于硬件，动态地发现和开发指令级并行。Intel的Pentium系列
  - ✓ 依赖于软件技术，在编译阶段静态地发现并行。Intel的Itanium处理器

7.7.1 指令流水线的限制

- 增加指令发射的宽度和指令流水线的深度，要求复杂硬件电路和高频率时钟的支持。这导致CPU功耗的上升。
- 目前已逐渐形成的共识是，功耗是限制当代处理器发展的首要因素。

7.7.2 突破限制的途径

- 流水线深度
- 从更深层次地解决流水线中可能存在的各种相关性
- 多核CPU，多指令流水线
- 现代处理器中，比较成熟的提高指令级并行的技术：

7.7.2 突破限制的途径

表7.9 提高指令级并行的技术

技术	简要说明	主要解决问题
直通和旁路	在流水线段间建立直接的连接通路	潜在的数据相关停顿
简单转移调度	冻结流水线，预取分支目标，多流，循环缓冲器等	控制相关停顿
延迟分支	利用编译器调度，填充延迟槽	控制相关停顿
基本动态调度（记分板）	乱序执行	真相关引起的数据相关停顿
重命名动态调度	WAW和WAR停顿，乱序执行	数据相关停顿、反相关和输出相关引起的停顿
分支预测	动态分支预测，静态分支预测	控制相关停顿
多指令发射	多指令流出（超标量和超长指令字）	理想CPI
硬件推测	用于多指令发射，使用重排序缓存	数据相关和控制相关停顿

7.7.2 突破限制的途径

表7.9 提高指令级并行的技术（续）

技术	简要说明	主要解决问题
循环展开	将循环展开为直线代码，消除判断、分支开销，加速流水	控制相关停顿
基本编译器流水线调度	对数据相关指令重排序	数据相关停顿
编译器相关性分析	利用编译器发现相关	理想CPI，数据相关停顿
软件流水线，踪迹调度	软件流水：对循环进行重构，使得每次迭代执行的指令是属于原循环的不同迭代过程的。踪迹调度：跨越IF基本块的并行度。	数据相关和控制相关停顿
硬件支持编译器推测	软硬件推测结合	理想CPI，数据相关停顿，转换相关停顿

7.7.3 指令级并行的限制

- 从单一线程中已经不太可能提取更多的指令级并行性，主要原因：
  - ✓ 不断增加的芯片面积提高了生产成本
  - ✓ 设计和验证所花费的时间变得更长
- 主流商业应用一般都具有较高的线程级并行性（Thread Level Parallelism, TLP）

  → 两种新型体系结构：
  - ✓ 单芯片多处理器（CMP）
  - ✓ 同时多线程处理器（Simultaneous Multithreading, SMT）
- TLP处理的基本思想：

  当某一个线程由于等待内存访问结构而空闲时，可以立刻导入其他的就绪线程来运行。处理器流水线就能够始终处于忙碌的状态，系统的处理能力提高了，吞吐量也相应提升。