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

本章内容：

7.1 流水线处理

7.2 浮点运算流水线

7.3 指令流水线

7.4 流水线性能测量

7.5 指令流水线的性能提高

7.6 多发射处理器

7.7 指令级并行概念

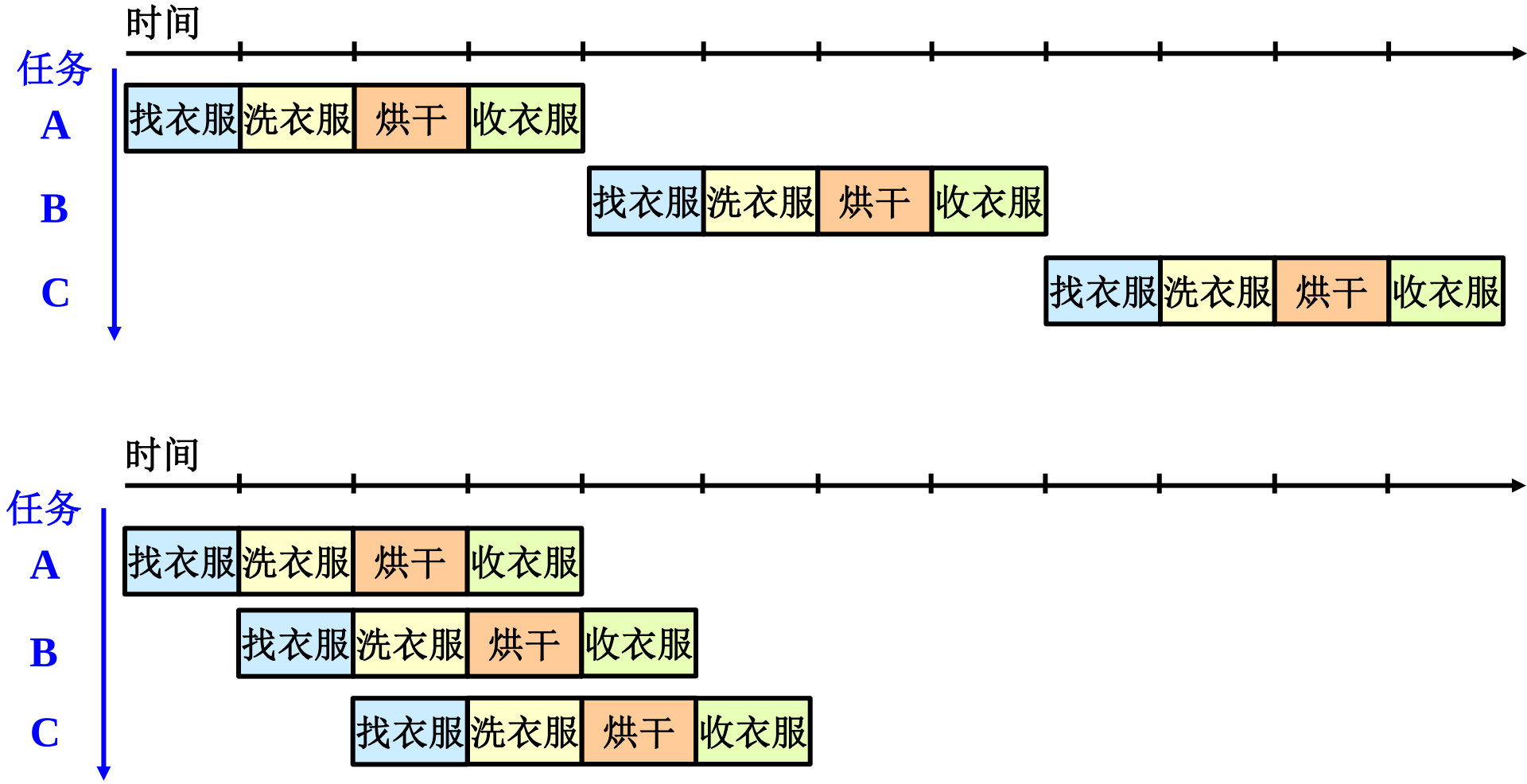
本章第1次课重点

- 流水线处理概念
  - ✓ 一般结构
  - ✓ 类型
  - ✓ 特点
- 浮点运算流水线
- 基本的指令流水线
- 指令流水线设计策略
- 指令流水线设计示例

流水线处理概念

什么是流水线

流水线：以洗衣服为例



流水线方式

7.1 流水线处理

若将一重复的处理过程分解为若干子过程，每个子过程都可在专用设备构成的流水线功能段上实现，并可与其它子过程同时执行，这种技术称为流水技术。

一、流水线的一般结构

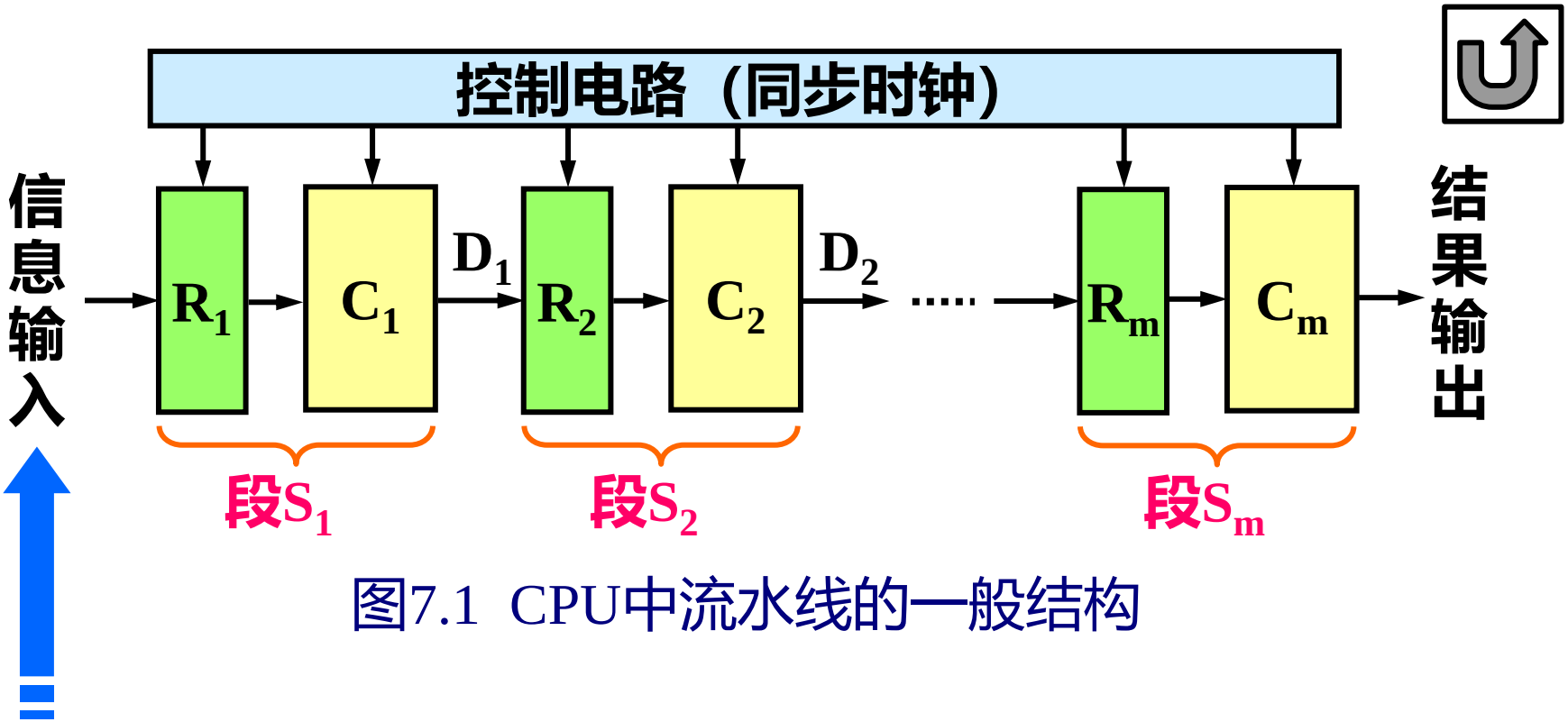


图7.1 CPU中流水线的一般结构

- 若加载的信息是数据就可以构成数据处理或运算流水线（arithmetic pipeline）。
- 若加载的信息是指令就可以构成指令流水线（instruction pipeline）。

7.1 流水线处理　　一、流水线的一般结构

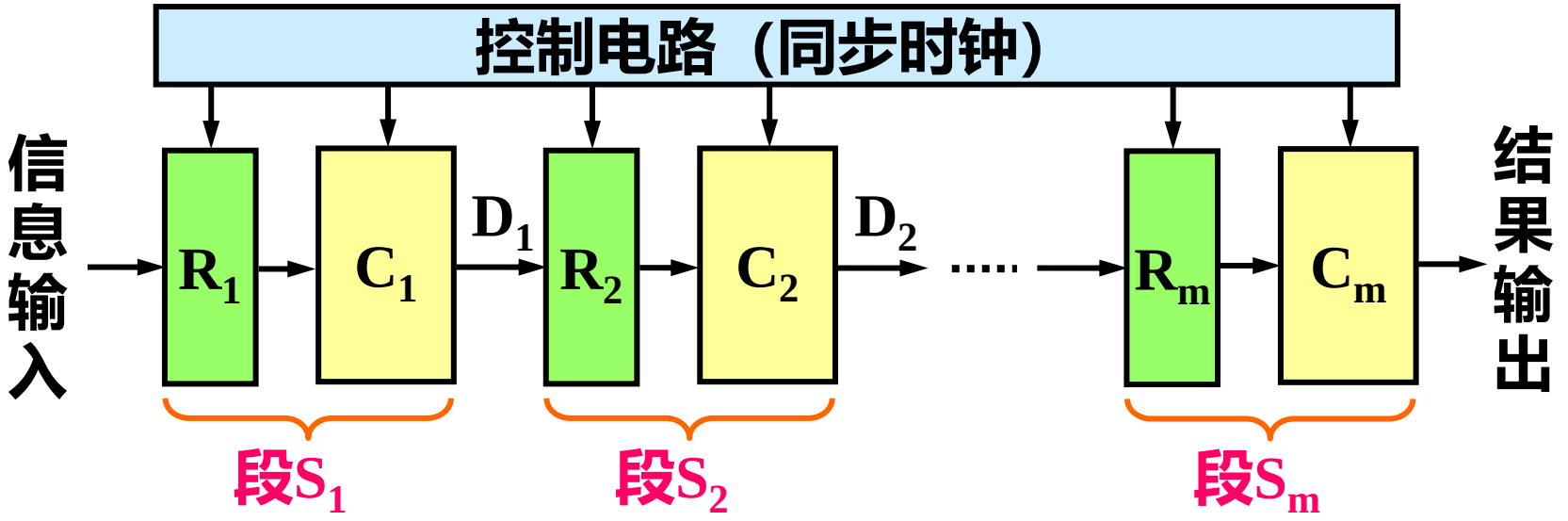


图7.1 CPU中流水线的一般结构

特点：

① 流水过程由多个相联系的子过程组成，每个子过程由专用的功能设备实现，每个子过程称为流水线的“级”或“段”。“级”数称为流水线的“深度”。各段间需要缓冲器隔离。

② 流水线需要有“通过时间/装入时间”，在此之后流水过程才进入稳定工作状态，每一个时钟周期（节拍）流出一个结果。

③ 流水线不能缩短单个任务的执行时间，但可以提高吞吐率；

一、流水线的一般结构

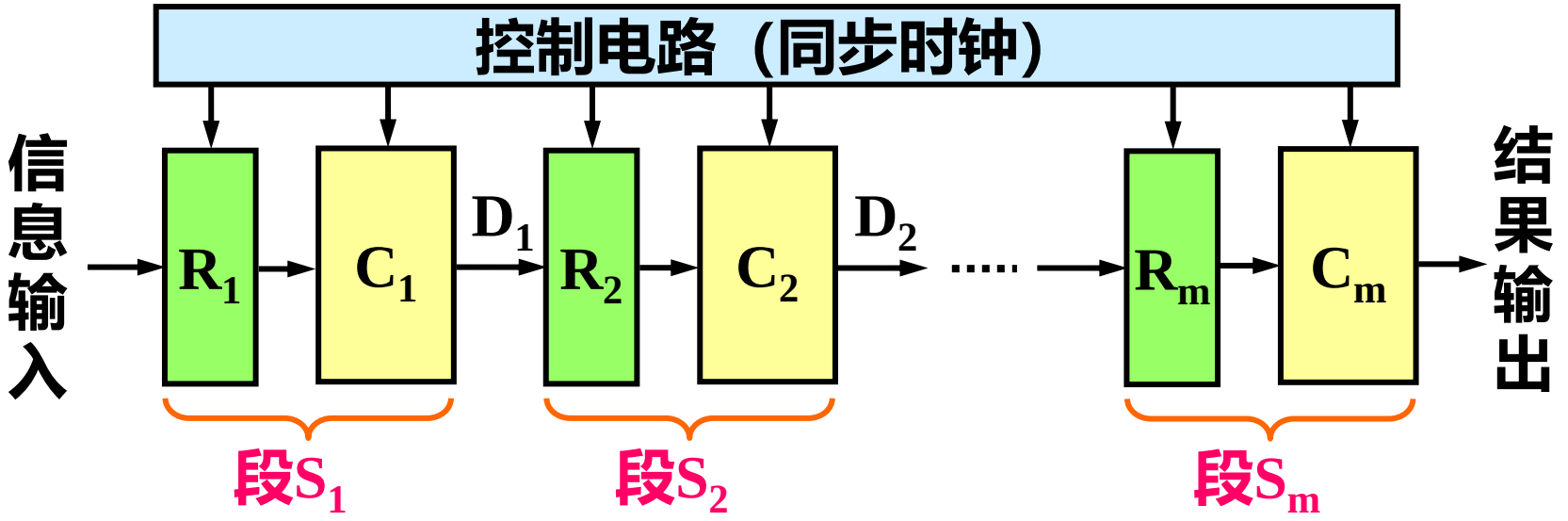


图7.1 CPU中流水线的一般结构

特点：

④ 流水线速度受限于最慢流水线段的运行速度，所以，各个功能段所需时间应尽量相等；

⑤ 流水技术适合于大量重复的处理过程，只有流水线的输入能连续地提供任务，流水线的效率才能充分发挥。

⑥ 流水线中多个任务是并行处理的。

➢ 按流水线位于计算机系统的层次划分：

✓ 系统级流水线/宏流水线：在多（计算）机系统中由多个处理机串行构成的流水线。

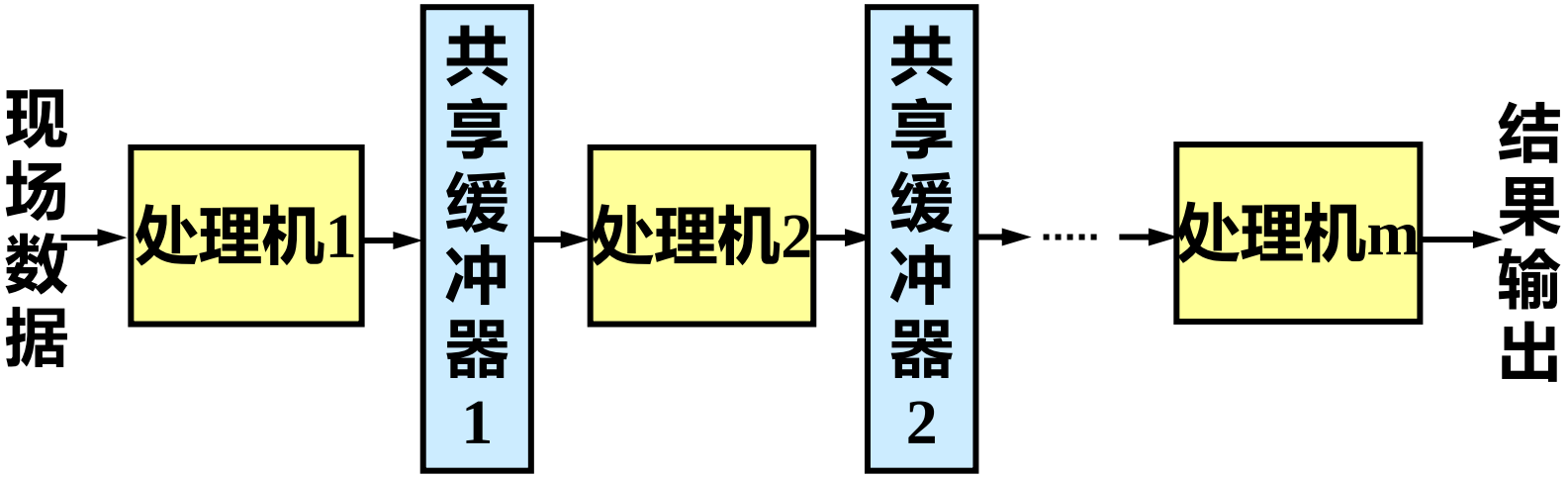


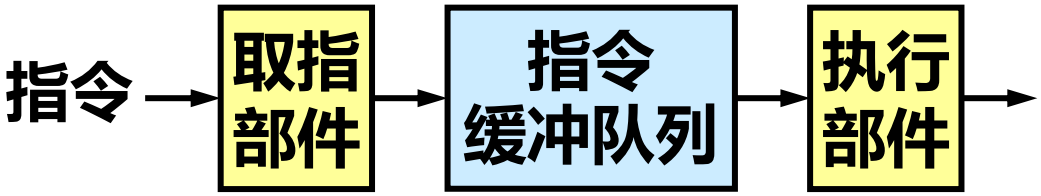
图7.2 系统级流水线

✓ 处理器级流水线

✓ 部件级流水线

二、流水线类型

- 按流水线位于计算机系统的层次划分：
  - ✓ 系统级流水线/宏流水线：在多（计算）机系统中由多个处理机串行构成的流水线。
  - ✓ 处理器级流水线：在处理器内部由多个部件构成的流水线



(a) Intel 8086指令流水线

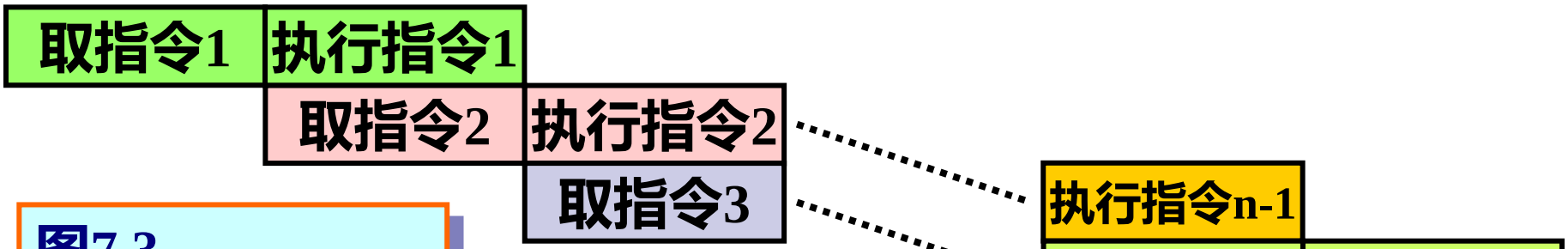


图7.3 处理器级流水线

(b) 流水效果示意图

  - ✓ 部件级流水线

二、流水线类型

- 按流水线位于计算机系统的层次划分：
  - ✓ 系统级流水线/宏流水线：在多（计算）机系统中由多个处理机串行构成的流水线。
  - ✓ 处理器级流水线
  - ✓ 部件级流水线：处理器单个部件
    - 运算操作流水线
    - 控制操作流水线

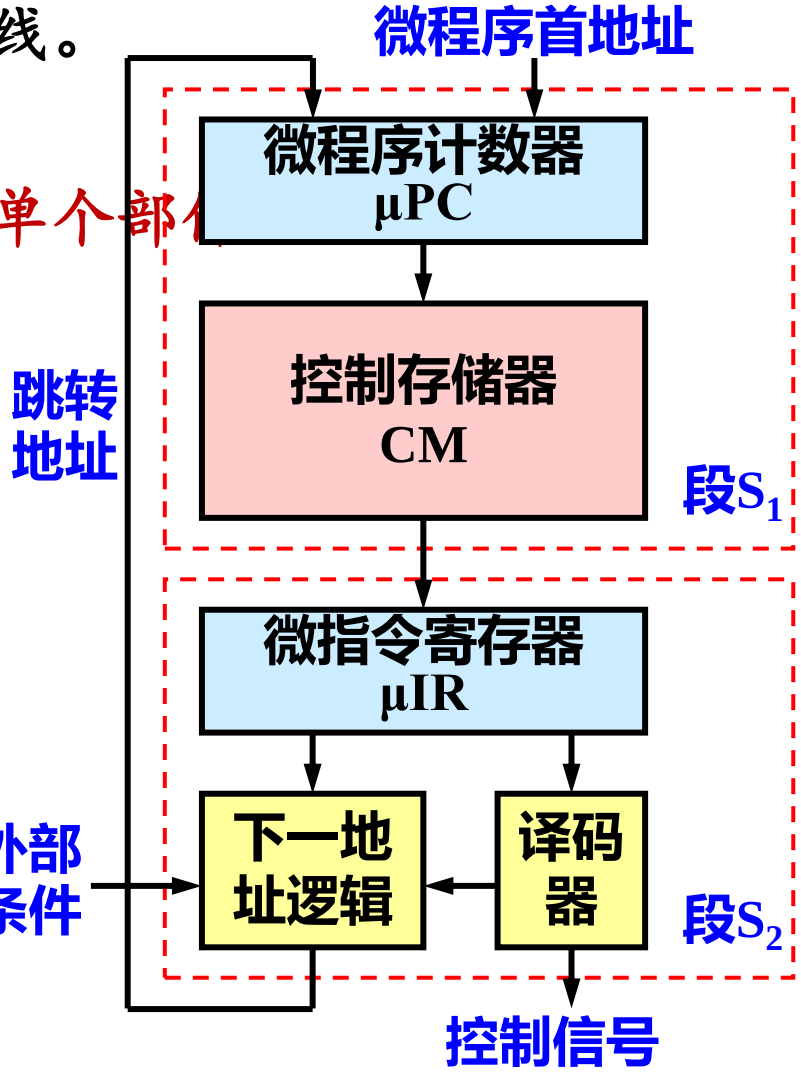
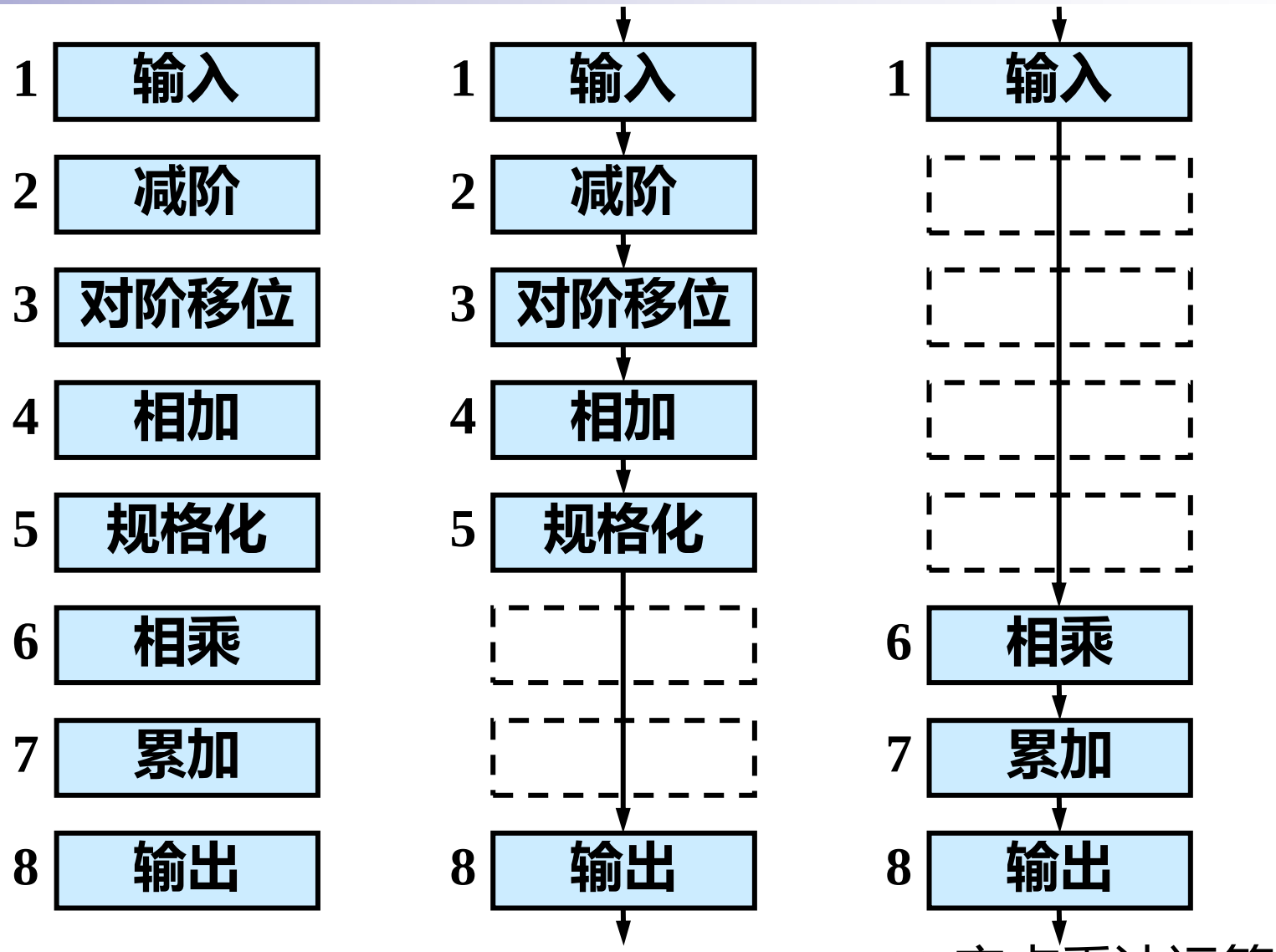


图7.4 部件级流水线
—— 流水的微程序控制单元

- 按流水线功能的强弱划分：
  - ✓ 单功能流水线：只能实现一种功能
    Cray-1向量计算机：160MFLOPS、8兆字节主存储器。有12条单功能运算流水线，分别完成地址加、地址乘、标量加、标量移位、标量逻辑运算、标量计数、向量加、向量移位、向量逻辑运算、浮点加、浮点乘、浮点迭代求倒数。将多个单功能流水线加以组合就可以实现多功能的流水操作。
  - ✓ 多功能流水线：通过各段不同连接方式实现多种功能
    - 静态流水线：在同一时间内流水线的各段只能按同一种功能的连接方式工作。适合处理一组相同的运算操作， TI ASC。
    - 动态流水线：在同一时间内流水线的某些段正在实现某种运算时，而另一些段在实现另一种运算。能提高流水线的效率，使流水线控制变得复杂。

二、流水线类型



(a) 流水线各功能段　(b) 浮点加、减运算　(c) 定点乘法运算

图7.5 TI ASC计算机的运算器流水线

二、流水线类型

- 按流水线是否有反馈回路划分：
  - ✓ 线性流水线
  - ✓ 非线性流水线—流水线调度
- 按流水线输出端任务流出顺序与输入端任务流入顺序是否相同划分：
  - ✓ 顺序流动流水线（入出顺序相同）
  - ✓ 异步流动流水线（入出顺序不同）
    - 无序流水线 / 错序流水线 / 乱序流水线
- 按流水线一次处理对象的数量划分：
  - ✓ 标量流水线
  - ✓ 超标量流水线
  - ✓ 向量流水线
  - ✓ 超长指令字流水线

浮点运算流水线

7.2.1 浮点加/减法器流水线

浮点数加/减运算需要4个操作步骤：

- 对阶

  加载：E1:= X_E，M1:= X_M；E2:= Y_E，M2:= Y_M；

  比较：E:= E1－E2；

  尾数对齐：

  while (E<0) { M1:= right_shift(M1)，E:= E+1}；

  while (E>0) { M2:= right_shift(M2)，E:= E-1}；

- 尾数加/减

  加/减：R:= M1＋M2，E:= max(E1，E2)；

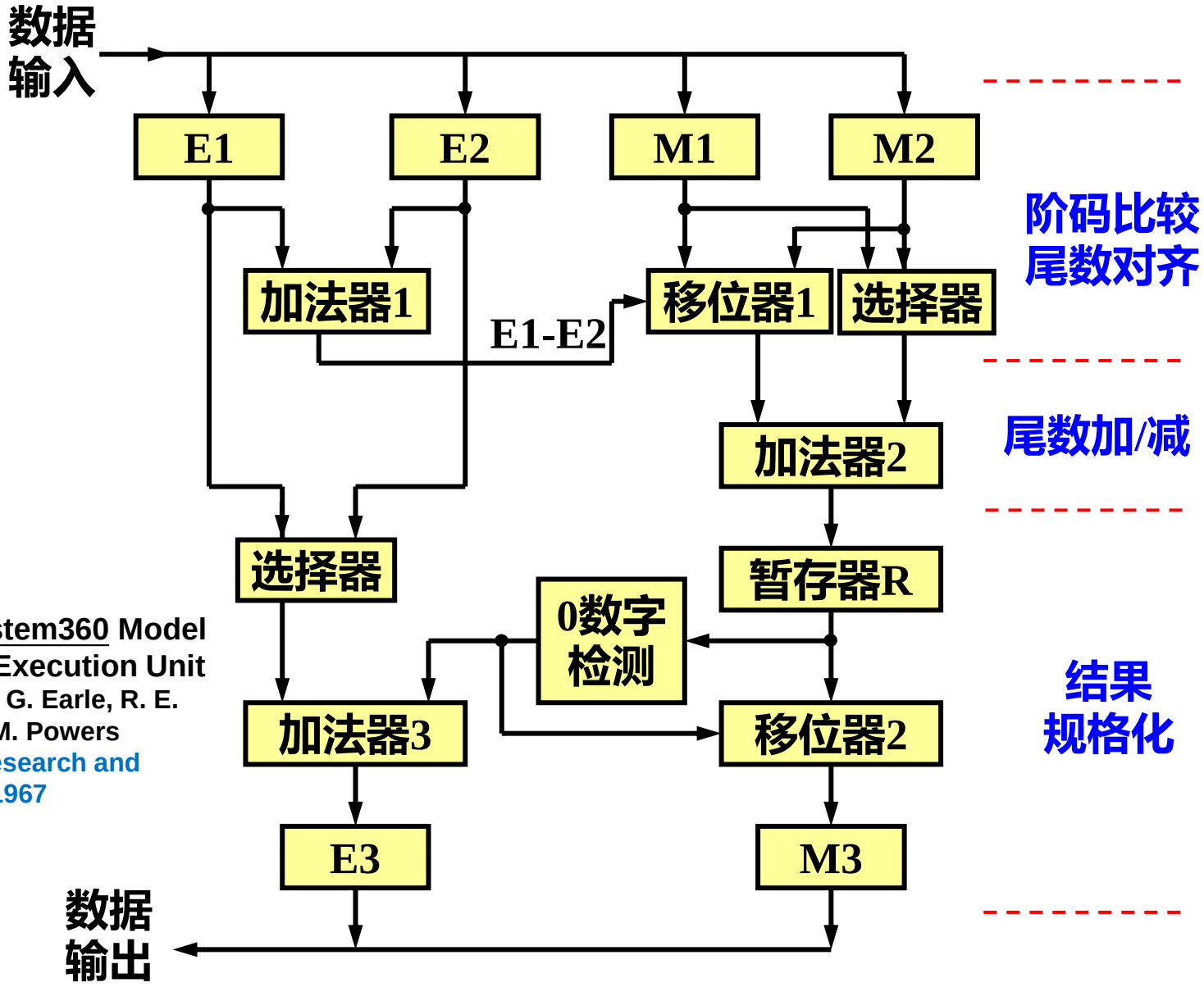
- 结果处理

  溢出、为零、规格化

- 舍入处理

  截断法、末位恒置“1”法、0舍1入法

7.2.1 浮点加/减法器流水线



参考文献

IBM The 91 System360 Model Floating-point Execution Unit
S. F. Anderson, J. G. Earle, R. E. Goldschmidt, D. M. Powers
Ibm Journal of Research and Development 01/1967

图7.6 IBM System/360/91的浮点加/减法单元

7.2.1 浮点加/减法器流水线

为某功能设计流水线电路：

- 寻找一个适当的、可分解为多步骤的、连续运行的算法，该算法的每一步骤应是时间均衡的、可以由硬件电路实现的。
- 将实现算法每一步骤的硬件电路作为流水线的一段，并按照步骤顺序将各段连接起来。
- 在流水线各段之间放置快速缓冲寄存器来分离各段，并利用缓冲寄存器逐段传送处理数据（部分或全部结果）。所有缓冲寄存器受同一时钟控制。

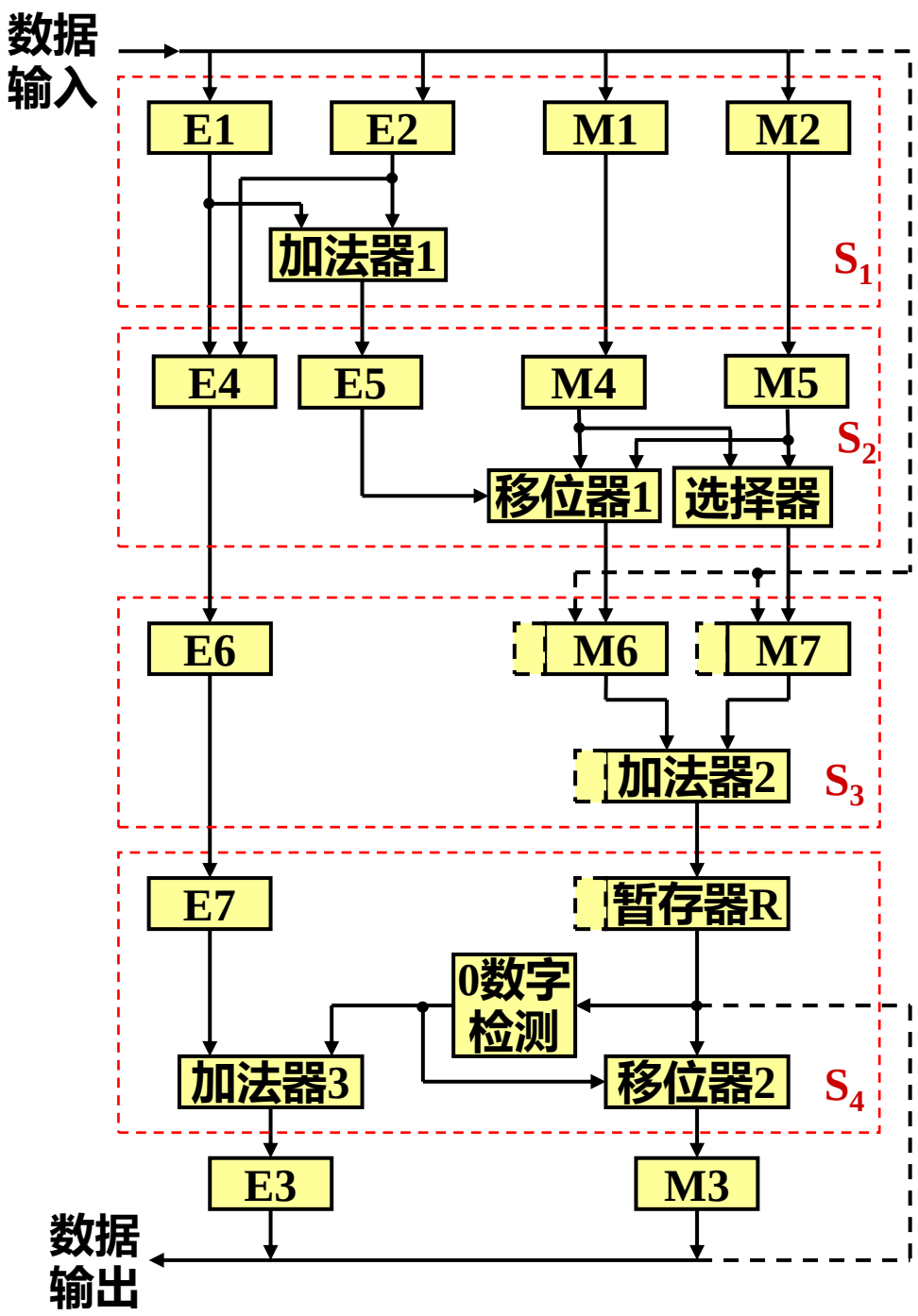


图7.7
浮点加/减法单元的
流水线结构

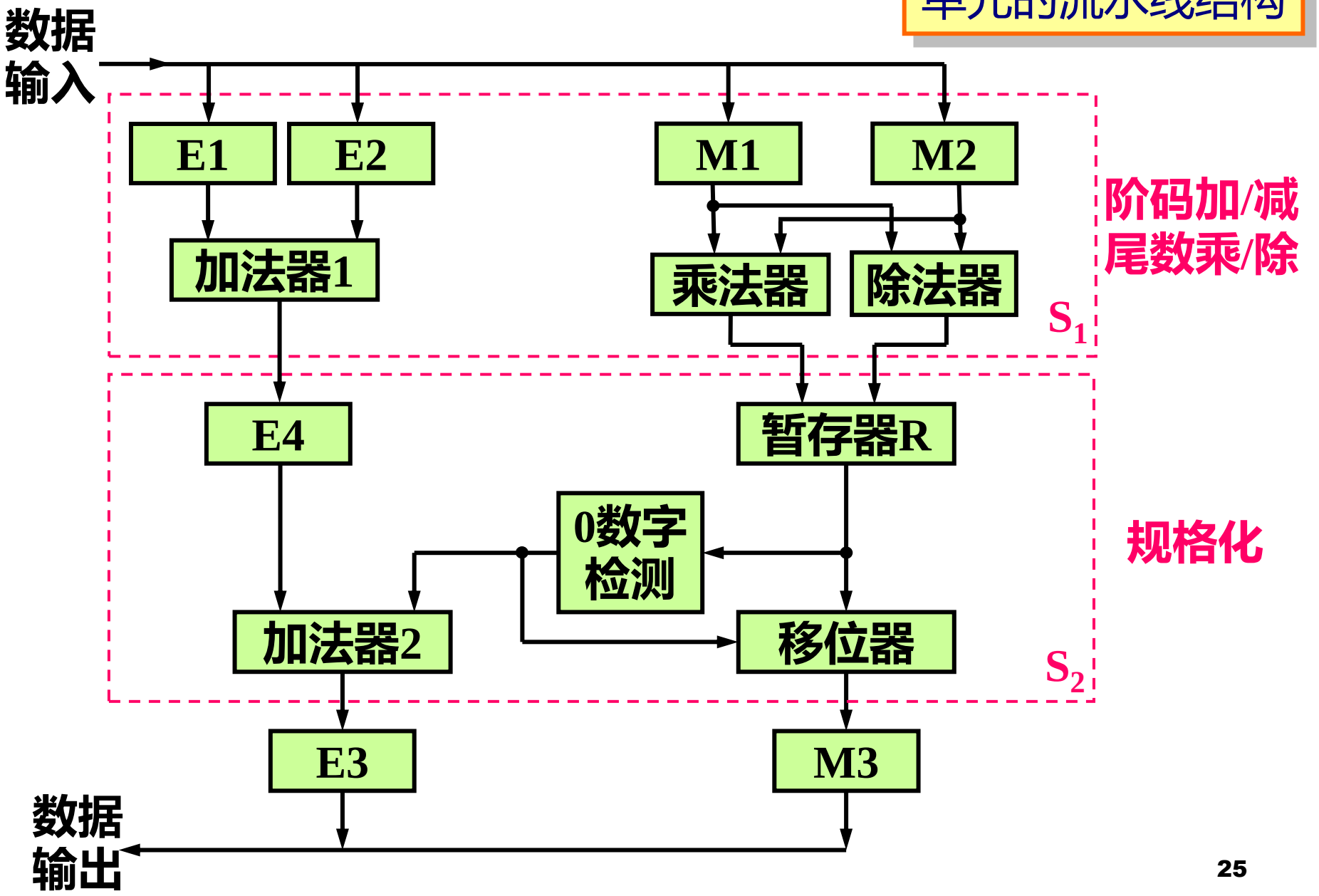
7.2.2 浮点乘/除法器流水线

浮点乘/除操作步骤：

- 阶码加/减
  - 乘法：判断上溢
  - 除法：判断下溢
- 尾数乘/除
- 结果处理
  - 溢出
  - 为0
  - 规格化
- 舍入处理
  - 截断法
  - 末位恒置“1”法
  - 0舍1入法

7.2.2 浮点乘/除法器流水线

图7.8 浮点乘/除法单元的流水线结构



7.2.2 浮点乘/除法器流水线

一种流水线结构的定点乘法器的实现：

若两个n位定点二进制数为$X=x_{n-1}x_{n-2}\cdots x_0$和$Y=y_{n-1}y_{n-2}\cdots y_0$，则2n位乘积P为：

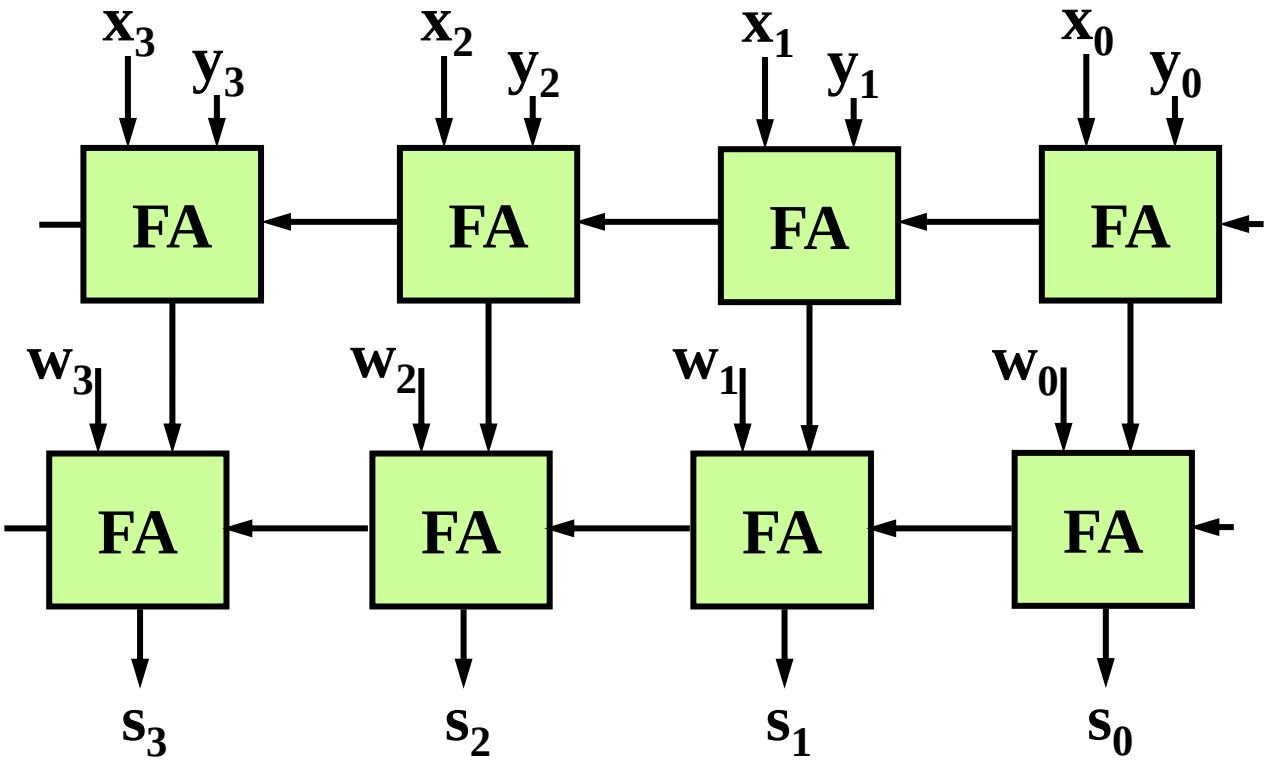
$$P = X \cdot Y = \sum_{i=0}^{n-1} M_i = \sum_{i=0}^{n-1} y_i 2^i X$$

→ 乘法的实现是通过一组M_i的求和来实现的

→ 进位保留加法（carry-save addition）

7.2.2 浮点乘/除法器流水线

$S=X+Y+W$



常规的两级加法器

7.2.2 浮点乘/除法器流水线

S=X+Y+W+Z

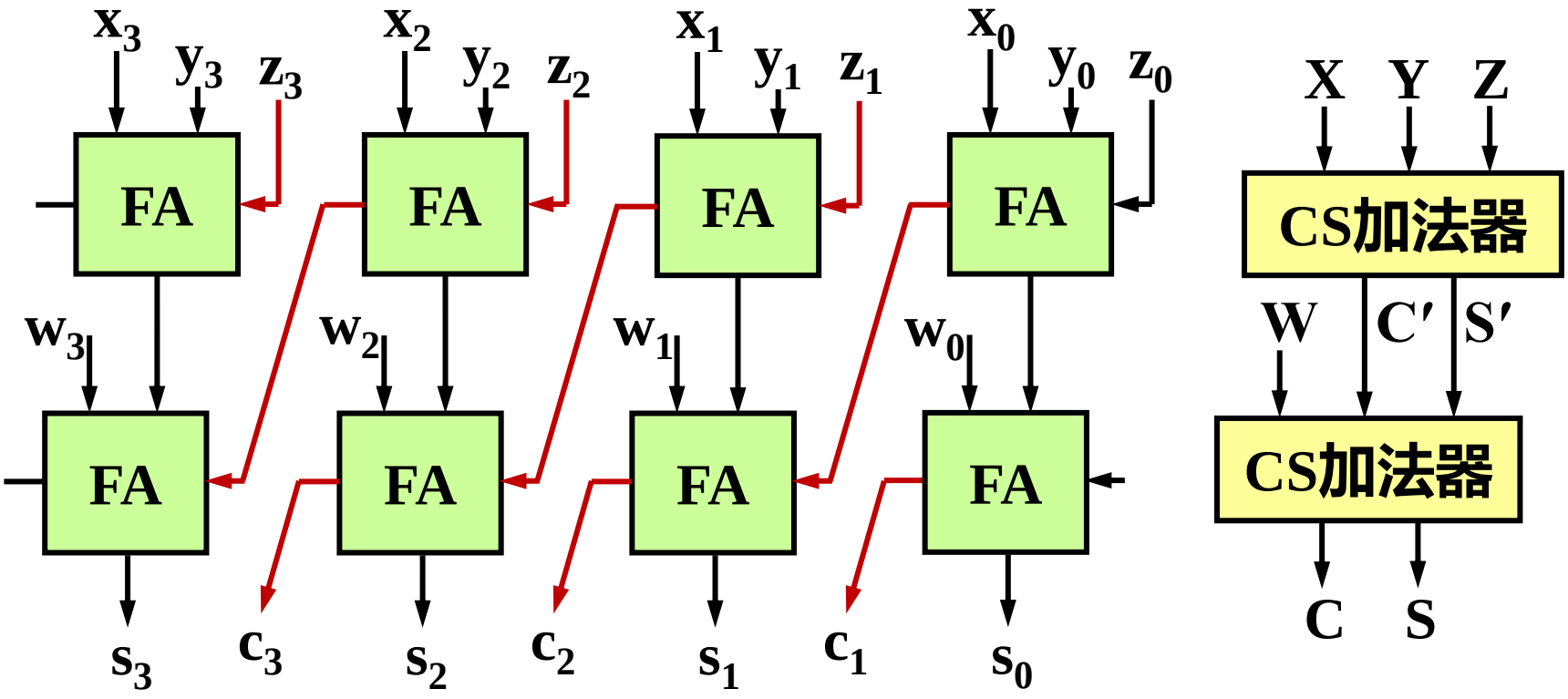


图7.9 两级的进位保留加法器

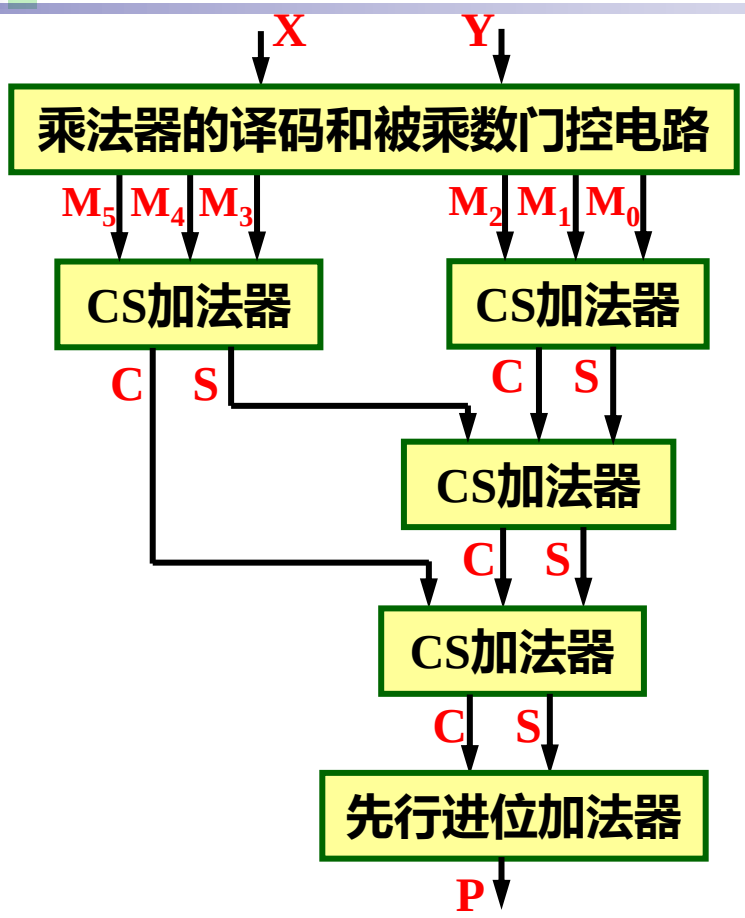


图7.10 进位保留（Wallace树）乘法器

需要CS加法器的数量：n-2

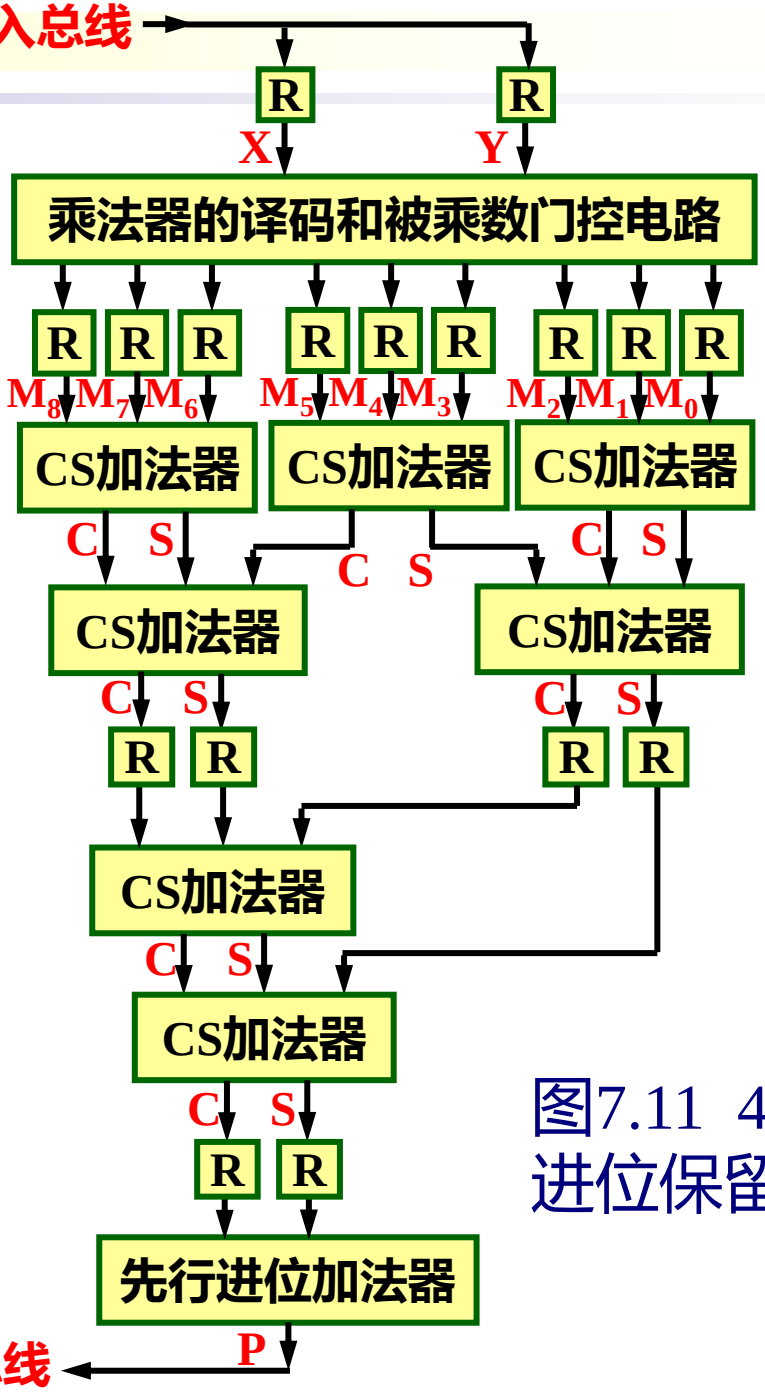


图7.11 4段流水进位保留乘法器

基本的指令流水线

7.3 指令流水线

提高计算机系统速度的途径：

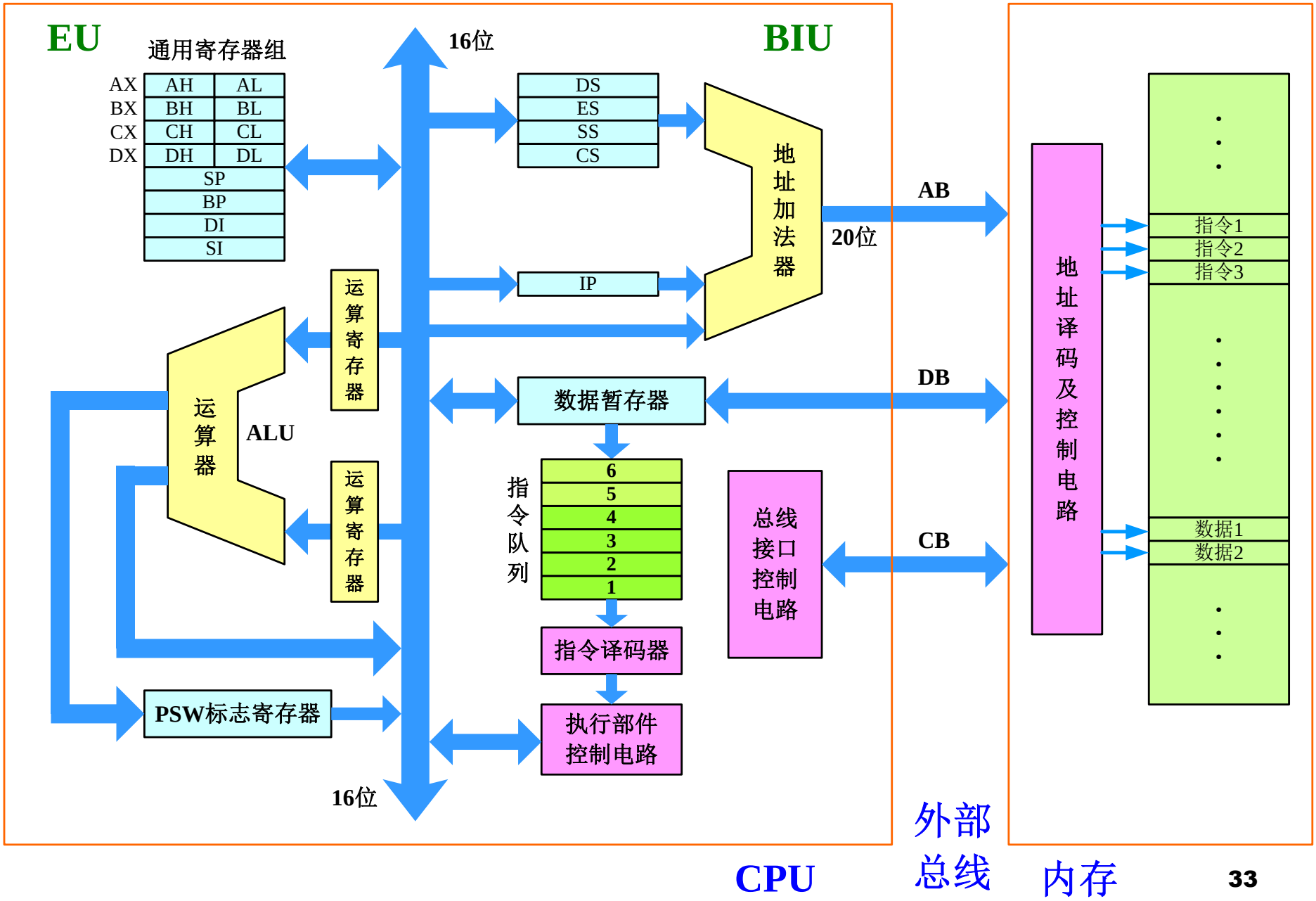
- 更快的电路
- 改进CPU组织结构
  - 减少用于执行指令的时钟周期数
  - 简化组织结构，缩短时钟周期
  - 用多寄存器取代单一的累加器
  - 在存储系统中引入高速缓冲存储器cache
  - 指令流水（instruction pipelining）:指令重叠执行

今天，指令流水线已成为加快处理器速度的关键，并成为现代计算机设计的核心思想。

7.3 指令流水线

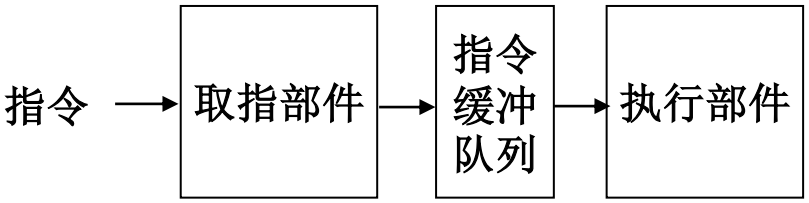
- 指令流水线是利用执行指令时存在的直接并行性实现多条指令重叠执行的一种技术。
- 通常，一条指令流水线（instruction pipeline）是一条多功能的、可重配置的流水线，该流水线通过将指令处理高效重叠的设计，使计算机的运行得到加速。
- 指令流水线对于程序员是不可见的，它由程序编译器和CPU内部程序控制单元自动管理。

8086/8088 CPU

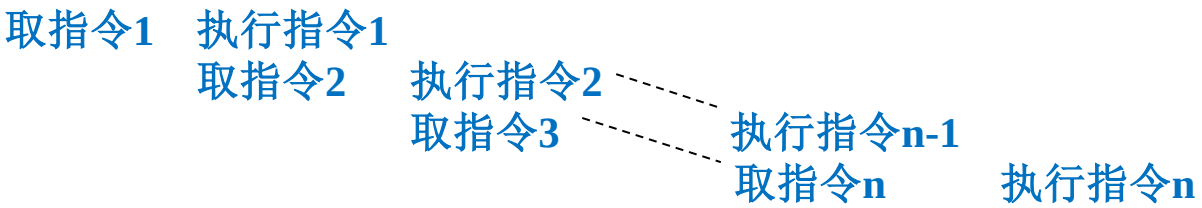


7.3 指令流水线

- 指令流水线是实现多条指令重叠执行的技术。如果将指令的处理简单地分解为两步骤：取指令和执行指令，那么，就可以设计一个两级的指令流水线。



(a) Intel 8086指令流水线



(b) 流水效果示意图

图7.3　两级指令流水线

假设每条指令的指令周期T相同，流水线每段的运行时间 τ 相同，则对于两级指令流水线而言，每经过 τ =T/2 时间，由流水线给出一条指令的执行结果，当执行一段程序时，可以节省近一半的运行时间。

7.3 指令流水线

- 在大多数情况下，取指令和执行指令的操作时间并不相同。
- 如果流水线各段采用同步方式控制各段间信息的同步推进，则流水线同步控制时钟的周期应选定为

$$T_{CLK} = \max\{\tau_i, \tau_i \text{为} S_i \text{段的运行时间}\}$$

这使得流水线中运行快的段将出现等待时间，造成流水线效能降低，速度减慢。

- 解决方案就是将指令的执行步骤再分解，使每一步骤所用时间尽可能均衡，并依此设计一个多级的指令流水线。

7.3.1 基本的指令流水线

【例1】将指令处理分解为以下4步：

1. 指令获取（IF）：从主存或cache中获取指令并对指令进行译码；
2. 操作数加载（OL）：从主存或cache中获取操作数放入寄存器中；
3. 执行指令（EX）：利用ALU等执行部件，对寄存器中的操作数进行处理，结果存于寄存器中；
4. 写操作数（WO）：将寄存器中的结果存入主存或cache中。

7.3.1 基本的指令流水线

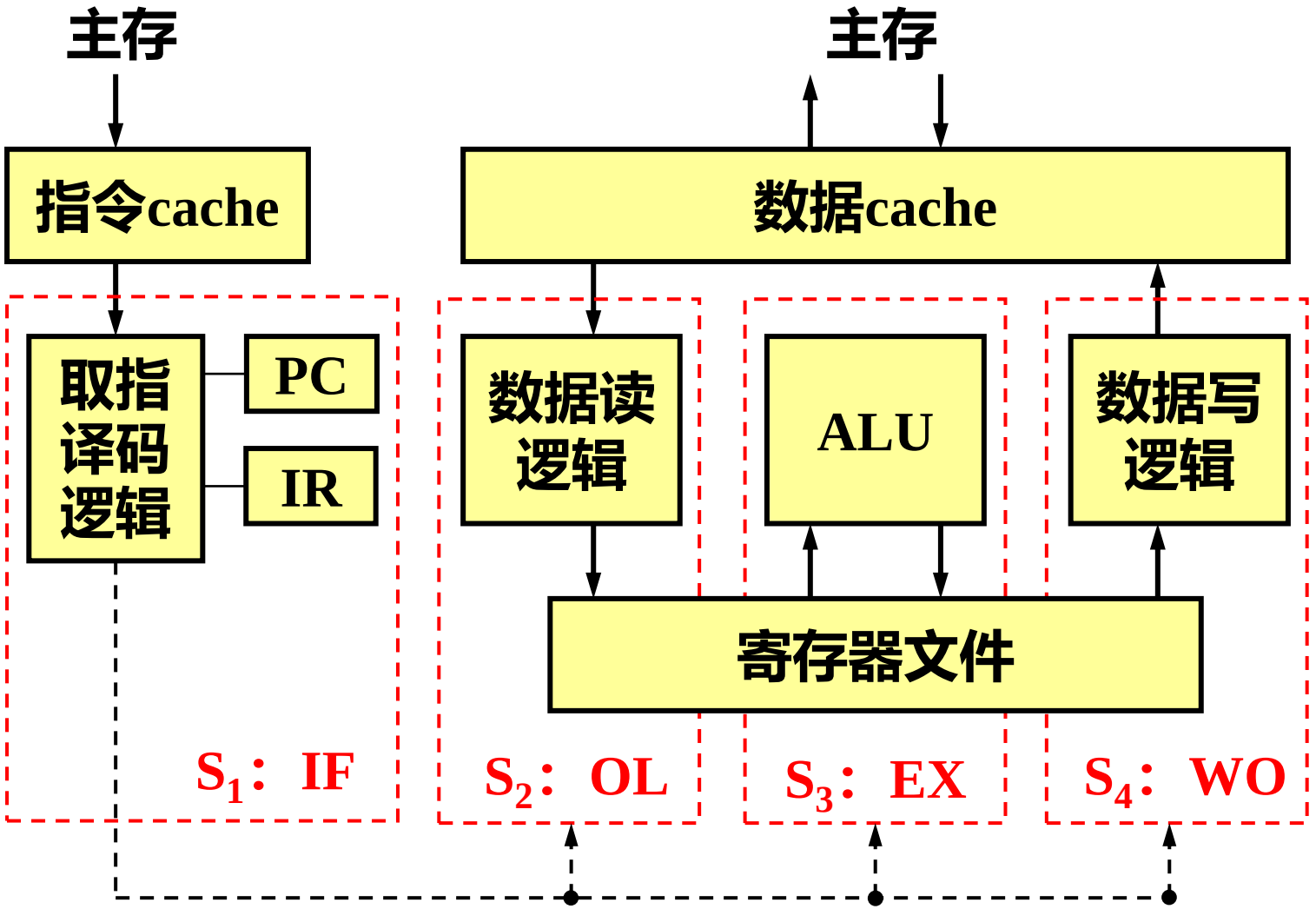
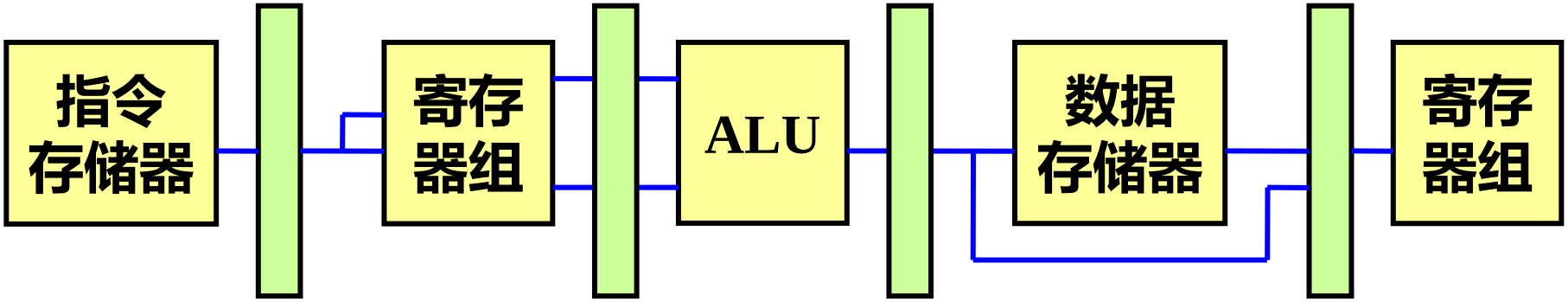


图7.12 由4级指令流水线组织的CPU结构

7.3.1 基本的指令流水线

【例2】RISC-V/MIPS采用的5级指令流水线：

1. IF：利用指令cache获得指令；
2. RD：当对取得的指令进行译码时，从寄存器RF中读取操作数；
   （MIPS指令无复杂的间接等寻址方式）
3. EX：利用ALU和必需的寄存器处理数据；
4. MA：利用数据cache加载或存储操作数；
5. WB：将操作数存储（写回）到寄存器。



RISC-V/MIPS 2/3000指令流水线结构

7.3.1 基本的指令流水线

- 在没有指令分支、cache失效或其他原因引起的延时等条件下，希望标量（单条）流水线CPU的最大执行速率达到理想水平，即$CPI_{理想}=1$。
- 间接寻址：在获取操作数之前，通常需要ALU计算操作数在内存中的地址，使得$CPI>1$。
- ALU完成复杂的运算（如浮点计算）往往需要多个时钟周期，使得$CPI>1$。

如何提高指令流水线的性能?

指令流水线设计策略

7.3.2 指令流水线策略　1. 增加指令流水线深度

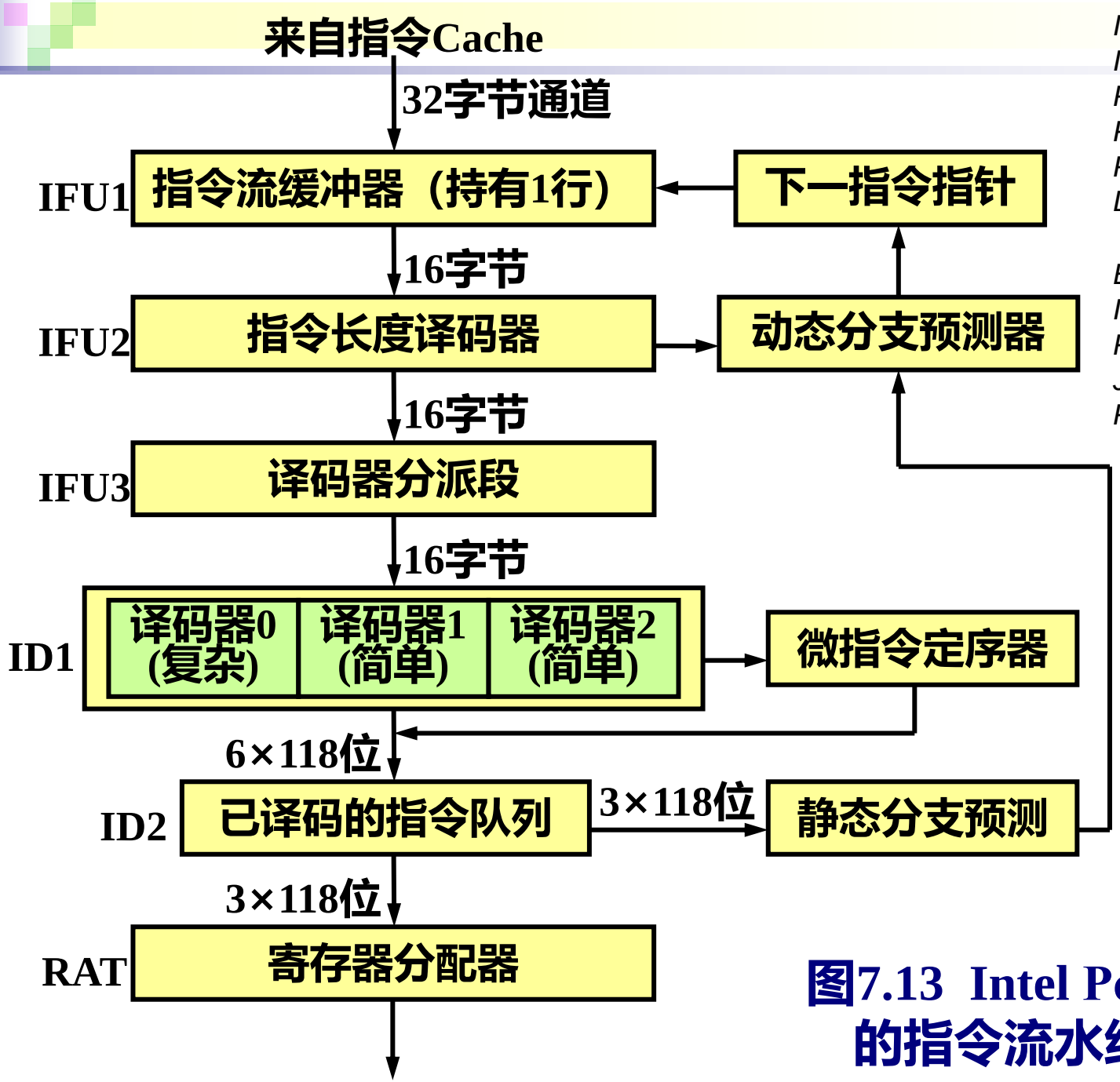
- 采用深度指令流水线结构：将指令的执行过程进一步细化，使流水线的级（段）数变多，而每一级的工作更少（有可能一个时钟周期完成）、时间更均衡。
- 这样做有两个好处：
  - ✓ 流水线级数变多、处理更趋合理，可使单条指令流水线并行执行指令的能力更强；
  - ✓ 每一级的处理时间更短，可以进一步提高处理器的工作频率。
- 深度指令流水线可以使处理器执行指令的速度更快、效率更高。

7.3.2 指令流水线策略

【例3】Amdahl 470V/7：

表7.1 Amdahl 470V/7的指令流水线段

功能	段(级)	名称	操作
指令获取 IF	S_1	指令地址	从存储器控制器请求下条指令
	S_2	启动缓冲器	启动cache读指令
	S_3	读缓冲器	从cache将指令读入到指令单元（I-unit）
	S_4	译码指令	对指令操作码进行译码
操作数加载 OL	S_5	读寄存器	读地址（基址和变址）寄存器
	S_6	计算地址	计算当前存储器操作数的地址
	S_7	启动缓冲器	启动cache读存储器操作数
	S_8	读缓冲器	从cache和寄存器文件（组）读操作数
执行指令 EX	S_9	执行1	传递数据到执行单元（E- unit）并开始指令的执行
	S_{10}	执行2	完成指令的执行
操作数存储 OS	S_{11}	检查结果	执行对结果的错误检查
	S_{12}	写结果	存储结果



IFU (Instruction Fetch Unit)
ID (Instruction Decode)
RAT (Register Allocator)
ROB (Reorder Buffer)
RS (Reorder Buffer)
DIS (Dispatcher)

EX (Execute Stage)
IEU (Integer Execution Unit)
FPU (Float point Unit)
JEU (Jump Execution Unit)
RU (Retire Unit)

图7.13 Intel PentiumII 的指令流水线结构

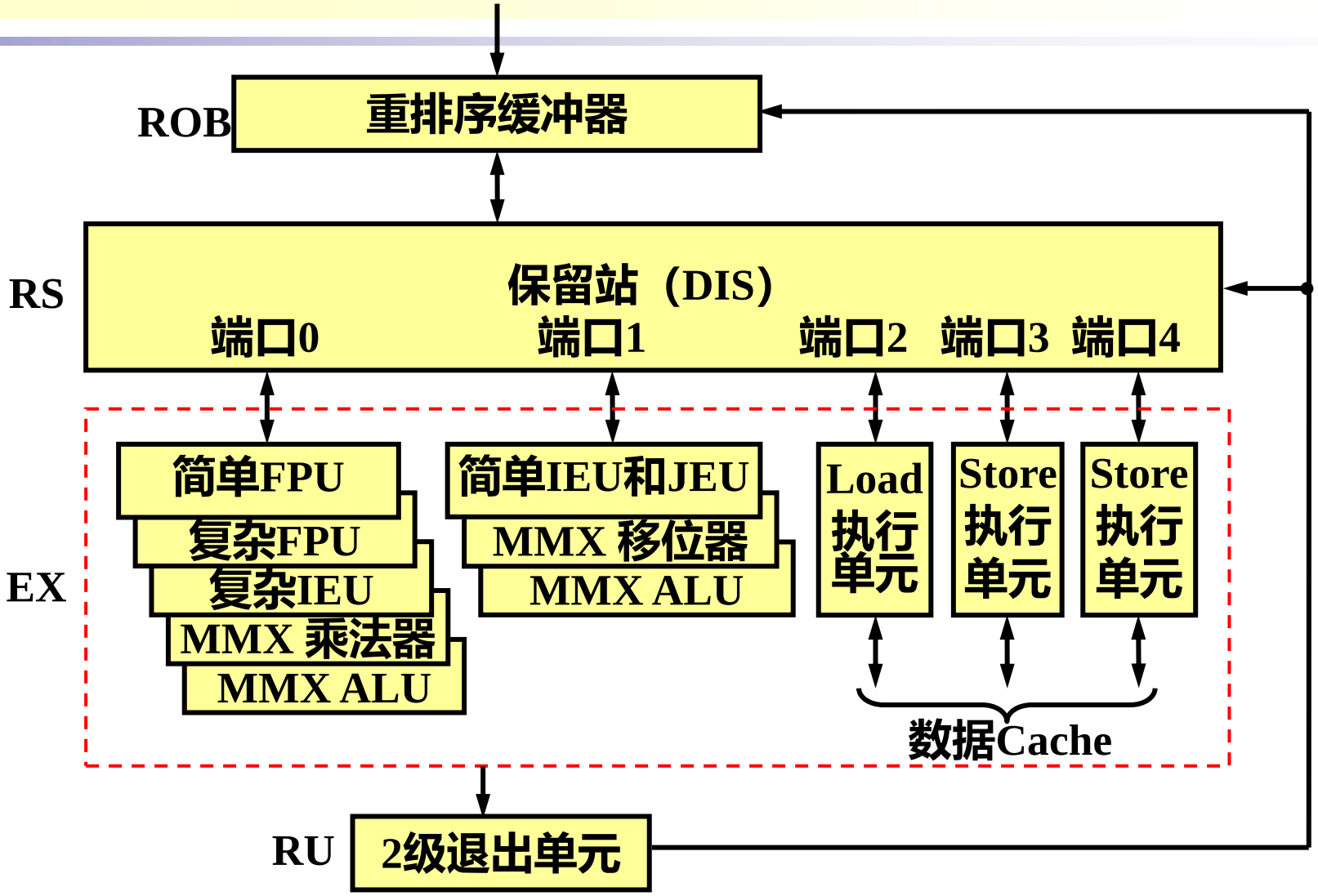


图7.13 Intel PentiumII的指令流水线结构（续）

Intel Pentium 4的Prescott核心(90nm)的CPU有高达31级的流水线

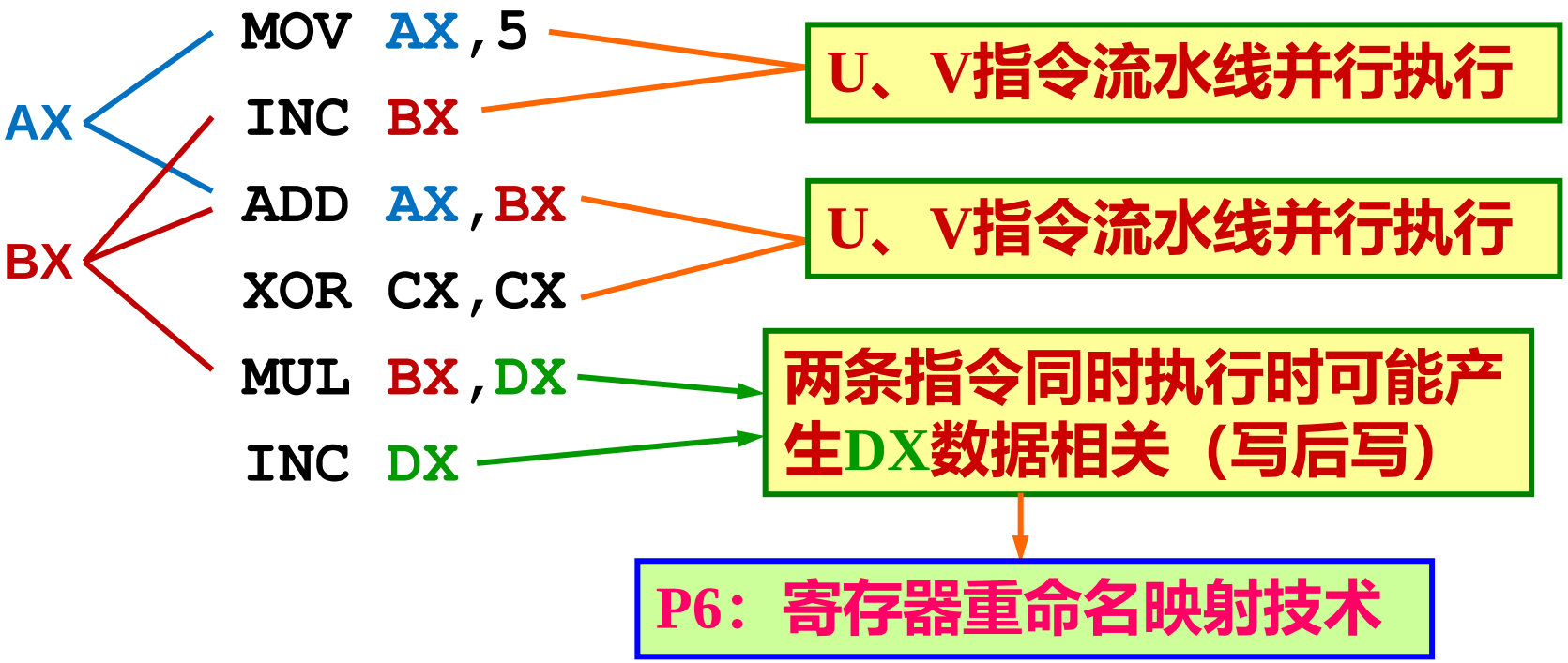
7.3.2 指令流水线策略　1. 增加指令流水线深度

- 增加流水线的深度可以提高流水线的性能，但流水线深度受限于流水线的延迟和额外开销，需要用高速锁存器作为流水线的缓冲寄存器。
- 由J.G.Earle在1965年发明的Earle锁存器是一种具有良好性能的高速锁存器，具有如下优点：
  - ✓ 对时钟扭曲（clock skew）相对而言不敏感，一般是两级门的延迟时间，避免了数据通过锁存器时可能产生的时钟扭曲；
  - ✓ 在锁存器中可以实现两级逻辑运算，而不会增加锁存器的延迟时间，这样，每个流水线中的两级逻辑可以与锁存器重叠，从而隐藏锁存器产生的额外开销。

7.3.2 指令流水线策略　2. 增加指令流水线条数

- 增加指令流水线的深度的局限：
  - 指令执行过程的细化是有限度的
  - 随着流水线深度的增加，流水线段之间的缓冲器增多，延迟加大，使流水线的性能提高受到阻碍
- 解决：多指令流水线结构
  - Intel Pentium 处理器：
    - U指令流水线（主流水线）
    - V指令流水线（副流水线）
  - IBM PowerPC 601
    - 定点运算流水线
    - 浮点运算流水线
  - 多发射CPU
  - 多核CPU

- 多指令流水线结构
  - Intel Pentium 处理器：
    - U指令流水线（主流水线）
    - V指令流水线（副流水线）



- 多指令流水线结构
  - ✓ IBM PowerPC 601

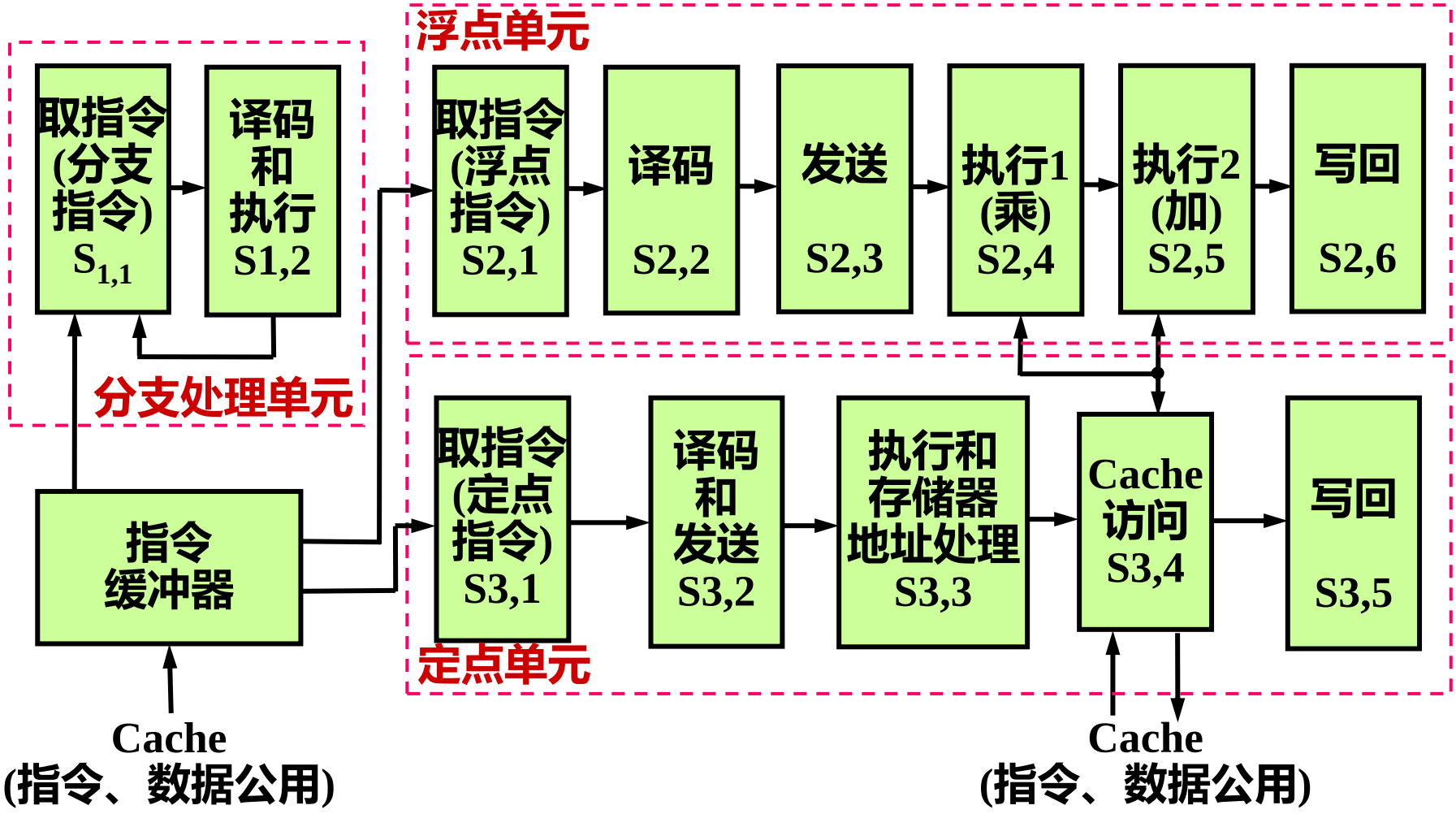
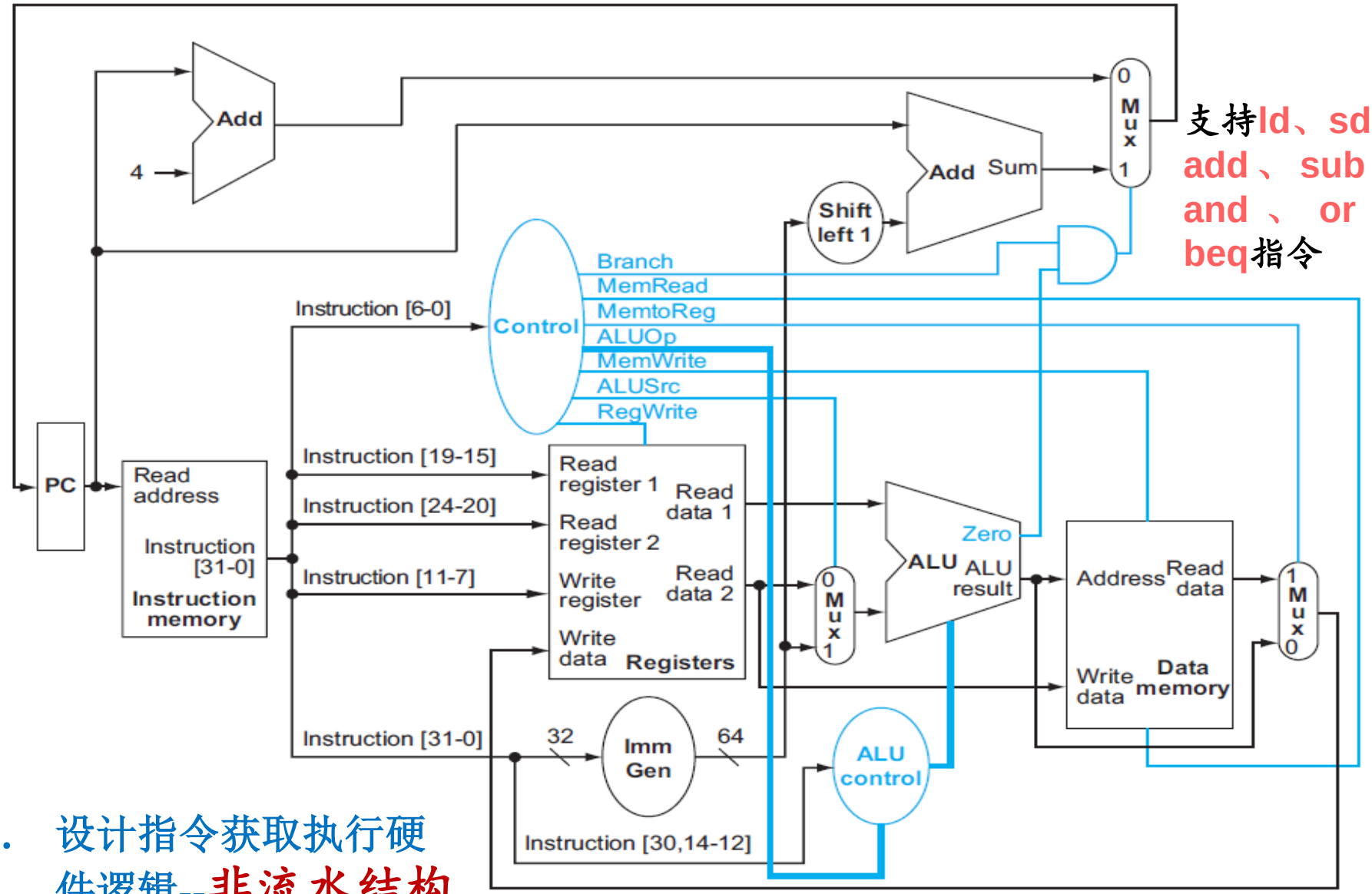


图7.14 PowerPC 601的指令流水线

指令流水线设计示例

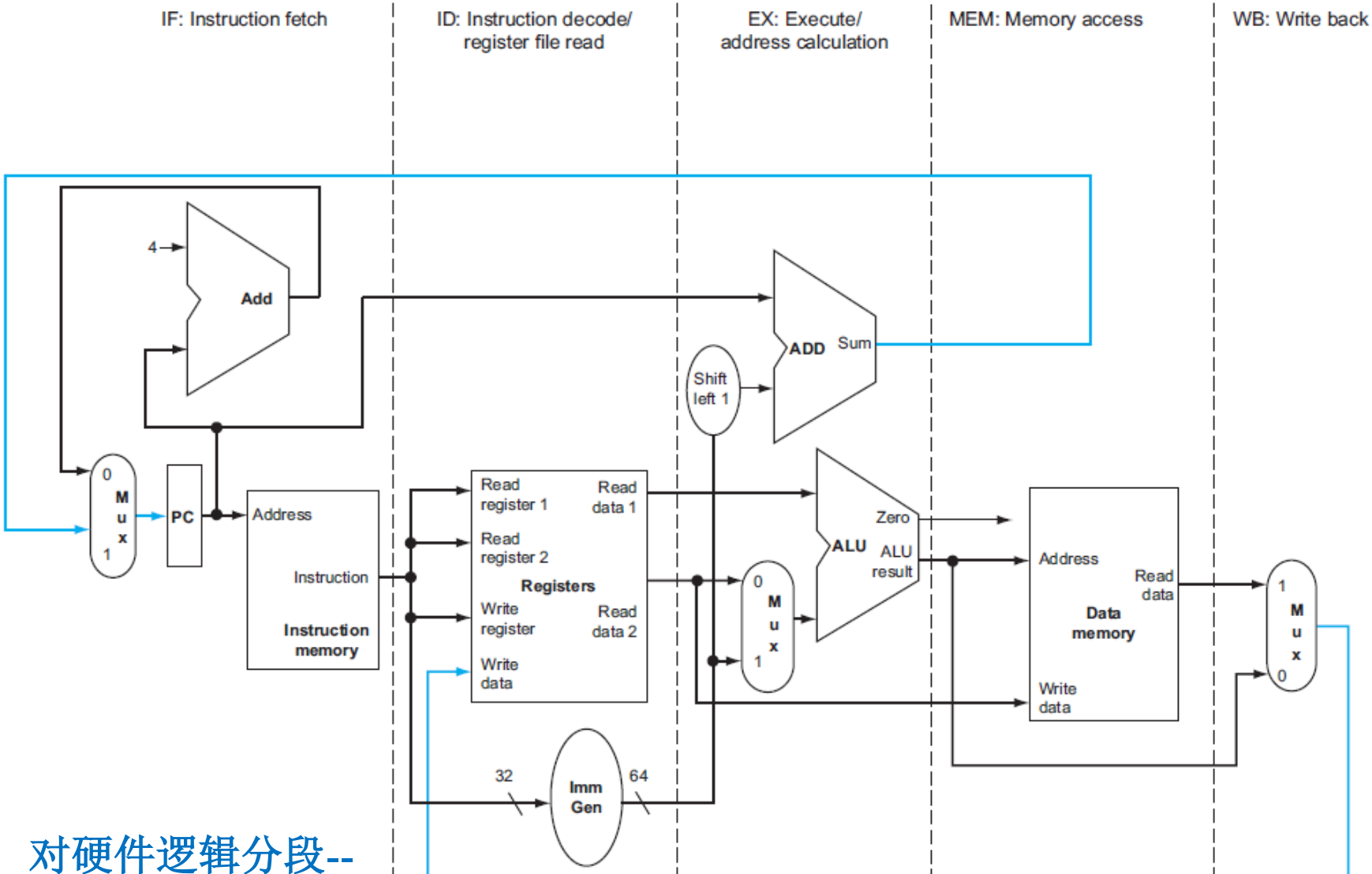
7.3.3 流水线设计示例（RISC-V/MIPS）



支持ld、sd、add、sub、and、or、beq指令

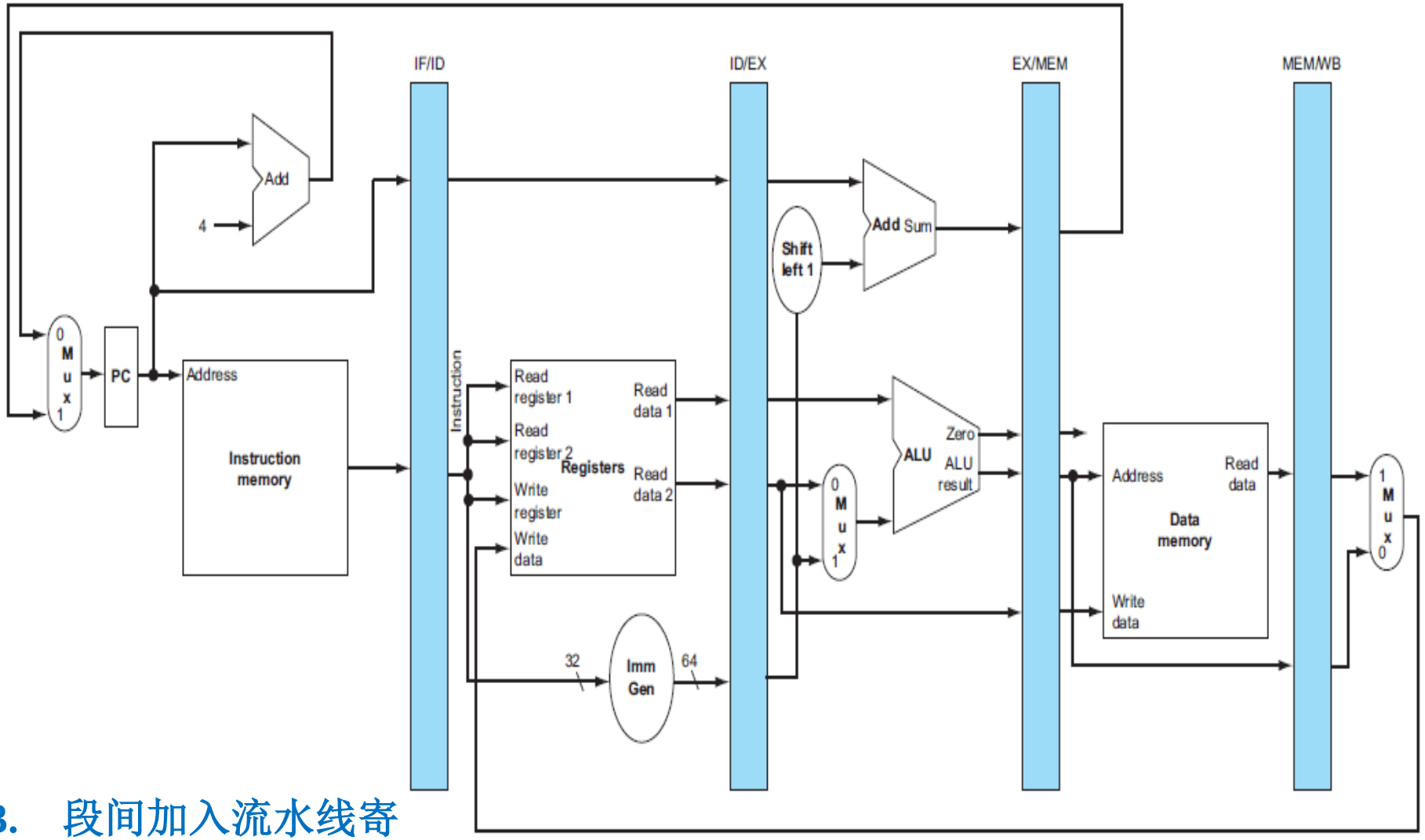
1. 设计指令获取执行硬件逻辑--非流水结构

7.3.3 流水线设计示例（RISC-V/MIPS）



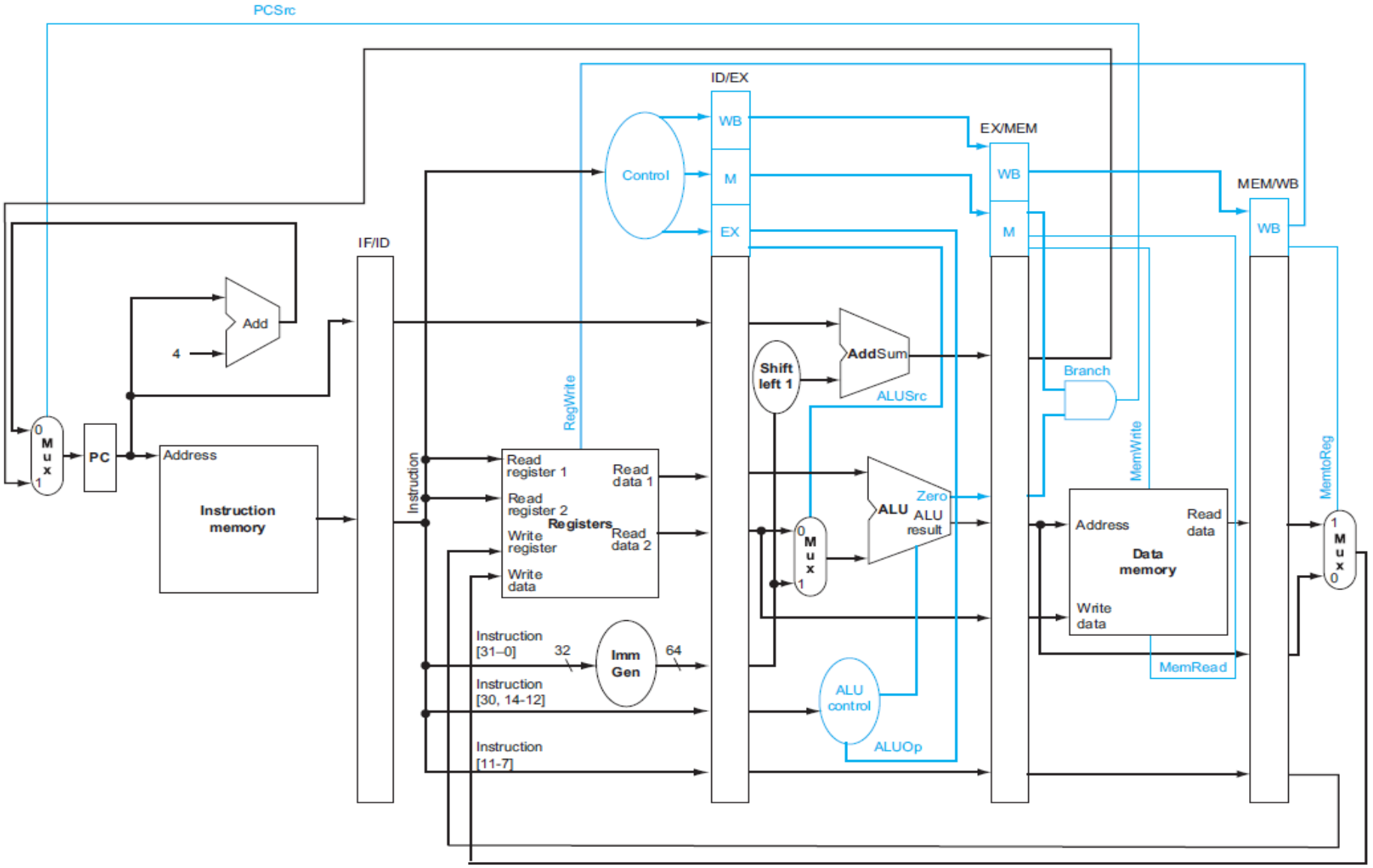
2. 对硬件逻辑分段--
分段非流水结构

7.3.3 流水线设计示例（RISC-V/MIPS）



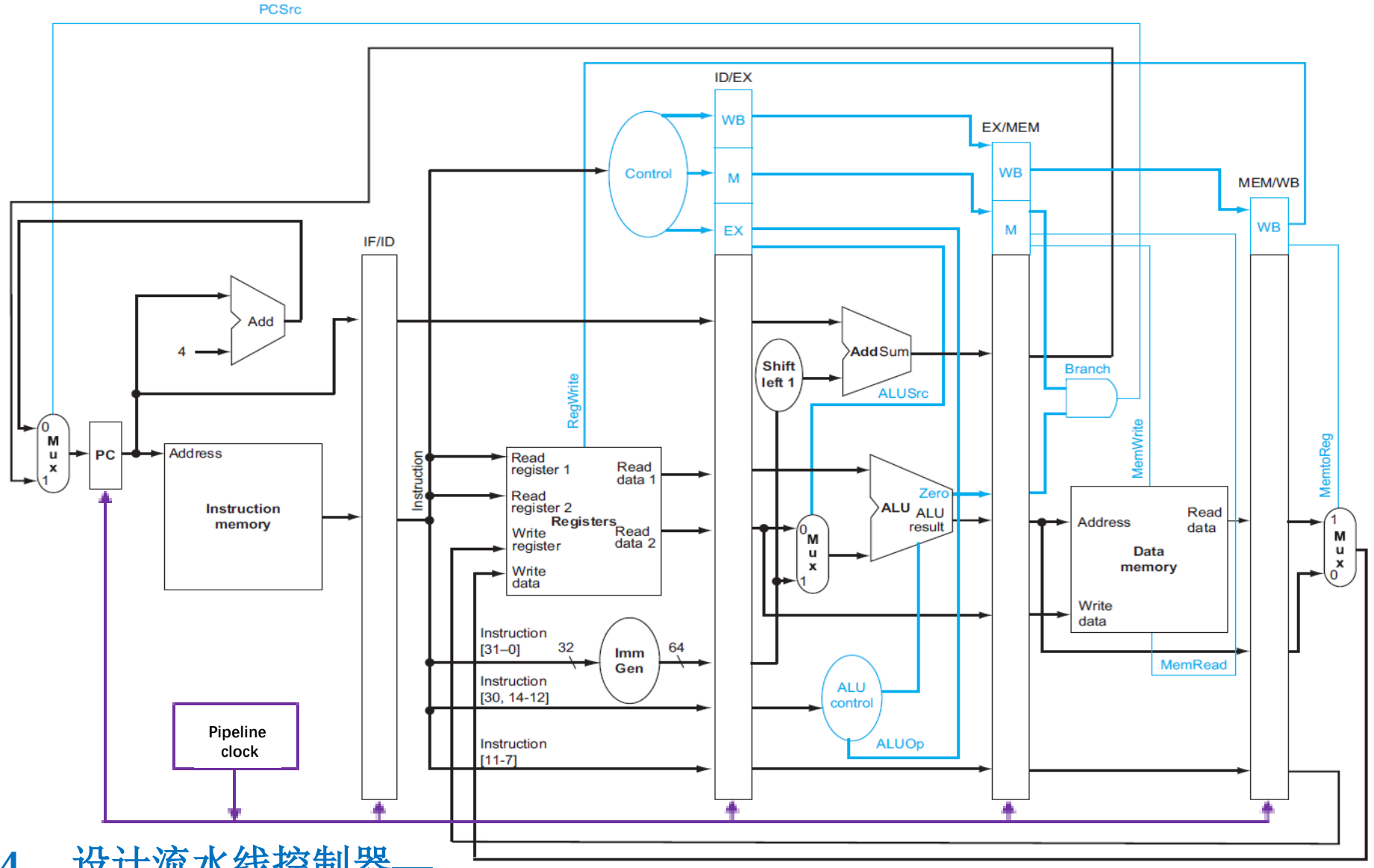
3. 段间加入流水线寄存器--流水线结构

7.3.3 流水线设计示例（RISC-V/MIPS）



带控制信号的流水结构

7.3.3 流水线设计示例（RISC-V/MIPS）



4. 设计流水线控制器——

有时钟控制的流水结构

7.3.4 指令系统对流水线设计的支持

与x86指令集相比，RISC-V指令集为流水执行而设计

- **指令长度**
  - 所有RISC-V指令长度相同，这一限制使得在IF段和ID段取指令与译码指令变得更加容易。
  - x86指令系统中的指令从1~17字节不等，实现流水线的难度要大得多。实际上，现代x86体系结构是将x86指令转换为类似于RISC-V指令的简单操作，然后将简单操作（而不是原生x86指令）流水化。
- **指令格式**
  - RISC-V只有几种指令格式，源和目标寄存器字段位于每个指令的相同位置。
- **加载/存储结构**
  - 主存操作数仅出现在RISC-V的加载或存储指令中，这个限制意味着可以使用EX段来计算主存地址，然后在MEM段访问主存。
  - 如果像x86那样访问主存操作数，那么EX段和MEM段将需要扩展到地址计算段、主存访问段、执行段，甚至需要更多的流水段。
  - 过长的流水线往往会带来负面效应，除了流水线寄存器的延迟可能会变得不可忽略，会明显增加流水线上指令间的相关性。
- **哈佛存储结构**
  - 信息流流向一致，便于流水实现。