

第4章 流水线结构

4.1 流水线结构原理

4.2 线性流水线性能指标

4.3 非线性流水线

4.4 流水线相关处理

4.5 超级流水处理机

目录

- 引言：系统结构中的并行性概念
 - 并行性概念
 - 时间重叠
 - 资源重复
 - 资源共享
- 4.1 流水线结构原理
 - 并行性基本概念，一次重叠、先行控制、流水线结构基本原理、性能分析
 - 基本设计技术
 - 高速缓存一致性, 协议, 网络, 流水线,
- 4.2 线性流水线技术指标
 - 吞吐率 加速比 流水线段数选择
- 4.3 非线性流水线
 - 调度技术
- 4.4 流水线相关处理
- 4.5 超级流水处理机-超标量处理机与超流水线处理机

微处理器发展趋势

- 计算机发展过程中**并行度**的提高是显著趋势
 - 截至1985年: 位级并行: 4-bit -> 8 bit -> 16-bit
 - 20世纪80年代中期32位的微处理器出现后发展趋势变缓
 - 64位芯片;
 - 将32位处理器及高速缓存放在同一芯片上产生了巨大影响
 - 上世纪80年代中期至90年代: **指令级并行**
 - **流水线技术以及精简指令集**+ 编译技术的进步 (精简指令集计算机)
 - 集成在芯片内部的高速缓存和功能单元 => 超标量执行 (superscalar execution)
 - greater sophistication: 乱序执行, 侦测, 预测
 - **解决控制转移以及延迟问题**

系统结构中的并行性概念

- **加快指令的解释过程**

采用高速部件，提高指令内部的并行性，加快单条指令的解释过程；

- **指令间的并行性**

并发解释两条或两条以上的指令，整段程序，提高计算机整体速度。

- **并行性概念**

时间重叠 资源重复 资源共享

并行处理的发展

单机系统中并行处理的发展

- 单处理机并行性开发的主要途径是**时间重叠**。
- 实现时间重叠的基础是**部件功能专用化**。
- 将一件工作按**功能分割**成若干联系的部分，每一部分有指定的专门部件来完成，
- 按时间重叠的原则把各部分执行过程在时间上重叠起来，使所有部件依次分工完成一组同样工作。

4.1 流水线结构原理

提高性能方法：

1. 提高单条指令性能；（硬件和指令系统）
2. 同时处理多条指令。（并行处理）

并行处理方法：

流水线、并行处理机、多处理机。

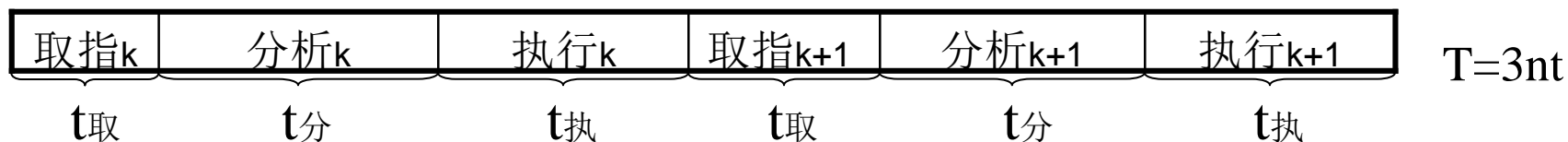
流水线结构原理

- 如何**加快指令的解释过程**是计算机组成设计的基本任务。
- 采用高速部件，**一次重叠，先行控制和流水等控制方式**
- 提高指令的并行性，从而加速指令的解释过程。控制方式分类：
 - 顺序方式
 - 重叠方式
 - 流水方式

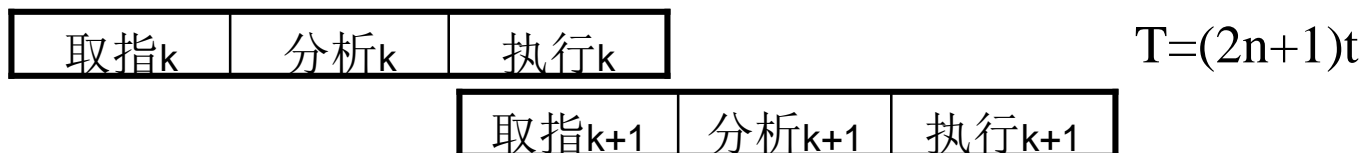
4.1.1 重叠操作

不同部件或同一部件内的各种操作在时间上重叠。

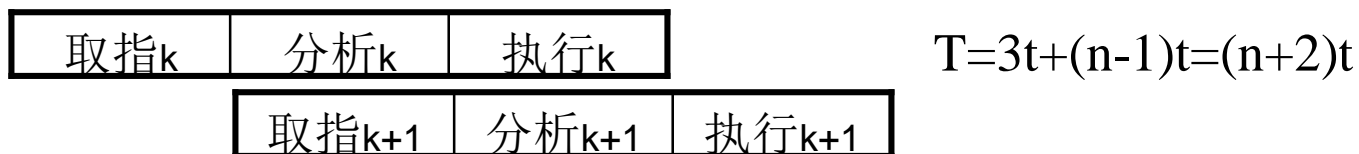
1. 工作方式



(a) 顺序解释执行



(b) 重叠解释执行



(c) 更高重叠程度的解释执行

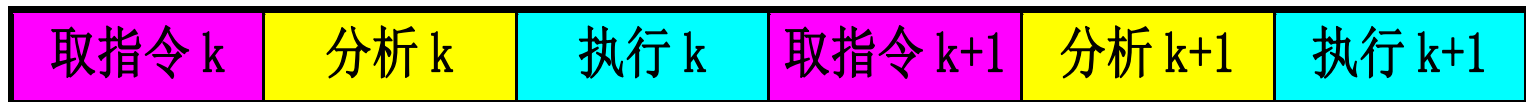
指令的重叠执行方式

1. 顺序执行方式

执行n条指令所用的时间为：

$$T = \sum_{i=1}^n (t_{\text{取指令}i} + t_{\text{分析}i} + t_{\text{执行}i})$$

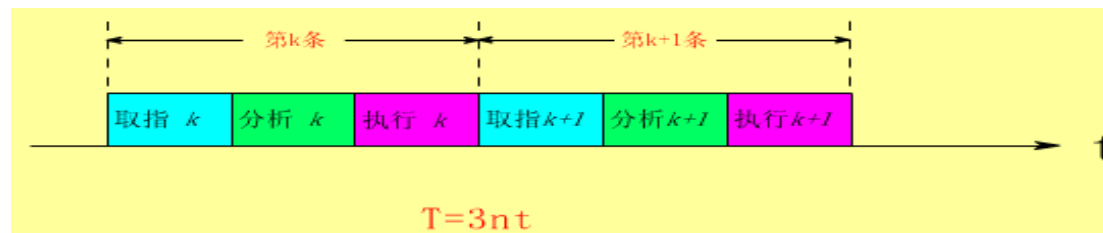
如果每段时间都为t，则执行n条指令所用的时间为： **$T = 3 n t$**



主要优点：**控制简单，节省设备**

主要缺点：**速度慢，功能部件的利用率低**

顺序执行方式



顺序执行方式执行n条指令所用的时间为

$$T = \sum_{i=1}^n (\text{取指}_i + \text{分析}_i + \text{执行}_i)$$

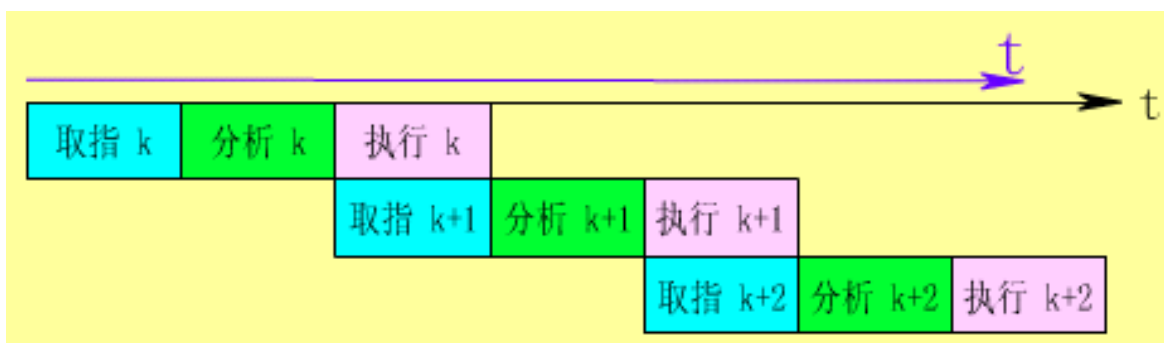
每段的时间都为t

$$T = 3nt$$

一次重叠执行方式

2. 重叠执行方式

- 执行第k条指令与取第k+1条指令同时进行
- 如果执行一条指令的3个阶段**时间均相等**,
则执行n条指令所用的时间为:
- $T = (2n+1)t$



二次重叠执行方式



- 如果执行一条指令的3个阶段时间均相等，
则执行n条指令的所用时间为：

$$T = (2+n)t$$

2. 硬件要求

能同时存取指令与数据： 指令与操作数分存于两个独立编址的存储器中；或采用多体交叉存储器，指令与操作数分存于不同的存储体内。操作数能同时存、取。

预取指令： 增设指令缓冲器。

并行执行： 指令分析、执行部件相互独立；不同部件执行时间的同步与锁存。

3. 存在问题

条件转移，数据相关，**同步**。

4.1.2 先行控制

先行控制结构

现代计算机组成中，**缓冲部件**使用较多，一般设置在两个工作速度不同的部件之间，起到平滑其工作的作用。

缓冲技术是计算机组成设计的一个重要技术。

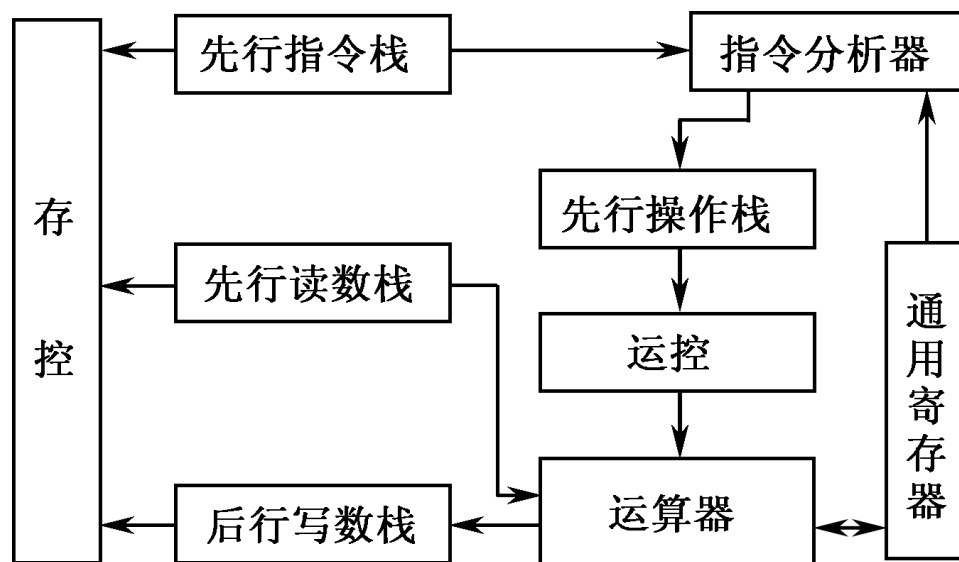
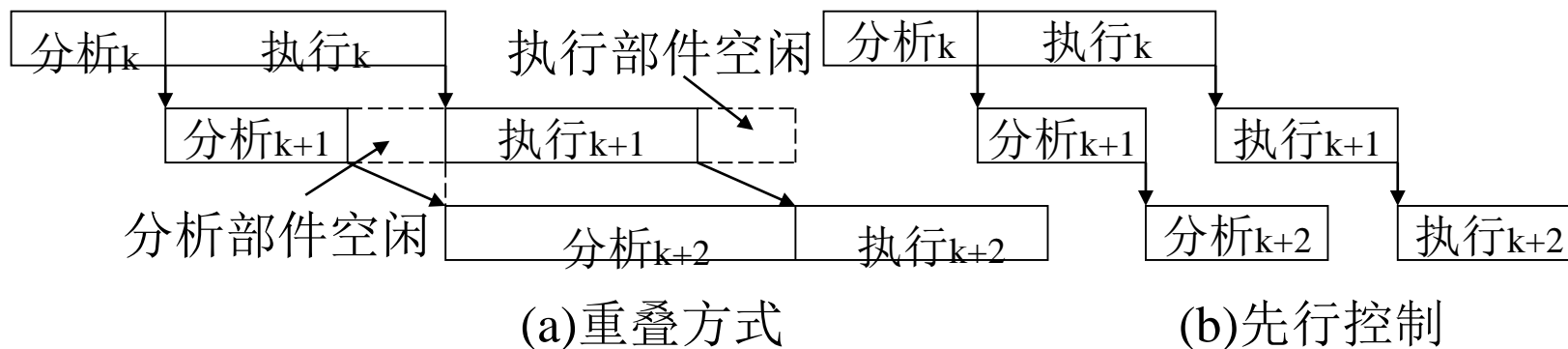


图4-9 采用先行控制方式的结构

4.1.2 先行控制

1. 工作原理

使分析和执行部件分别连续不断地运行，使部件空闲状态减至最低。



$$T_{\text{重}} = t_{\text{分}1} + \sum_{i=2}^n \max(t_{\text{分}i}, t_{\text{执}i-1}) + t_{\text{执}n}$$

$$T_{\text{先}} = t_{\text{分}1} + \sum_{i=1}^n t_{\text{执}i}$$

4.1.2 先行控制

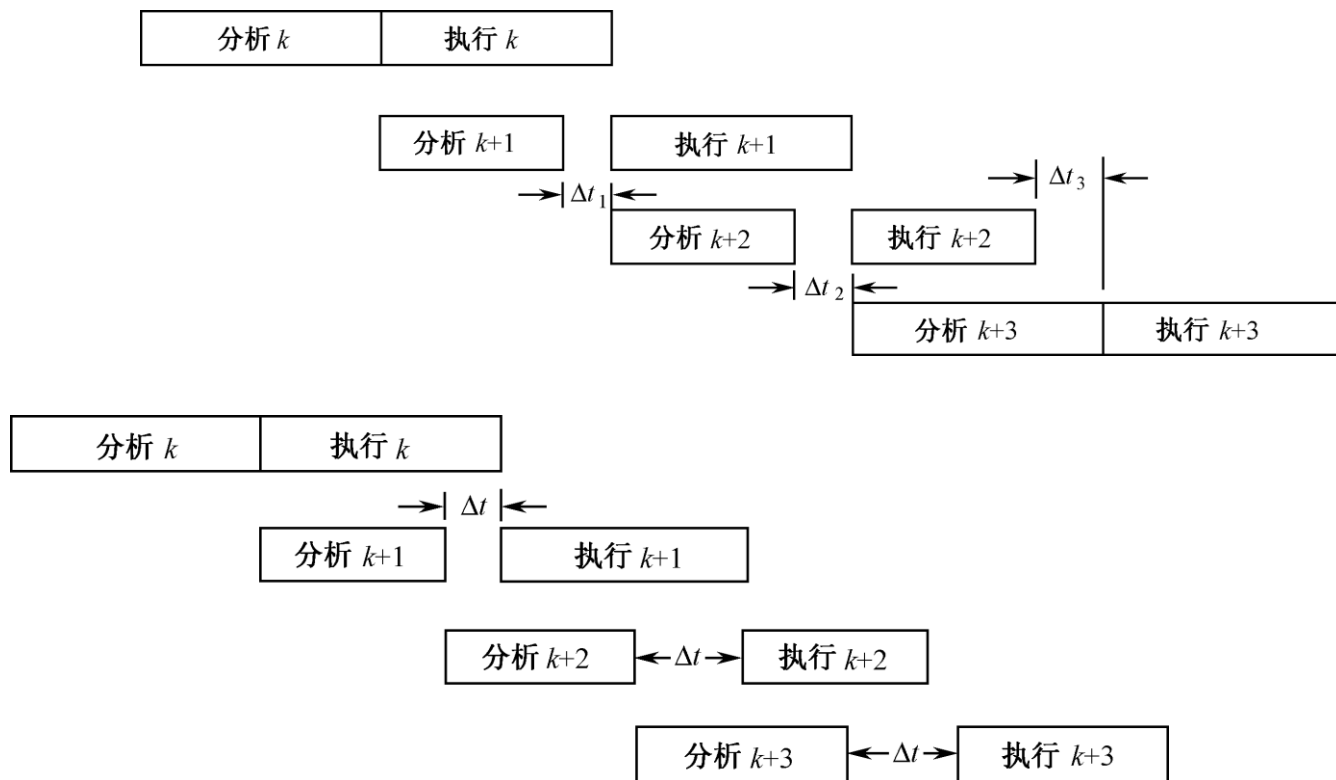


图4-8 先行控制方式的时序 ---能提前的就“先行”

与重叠区别：分析和执行部件可同时处理两条不相邻指令。

采用技术：

缓冲技术+预处理技术

2. 硬件要求

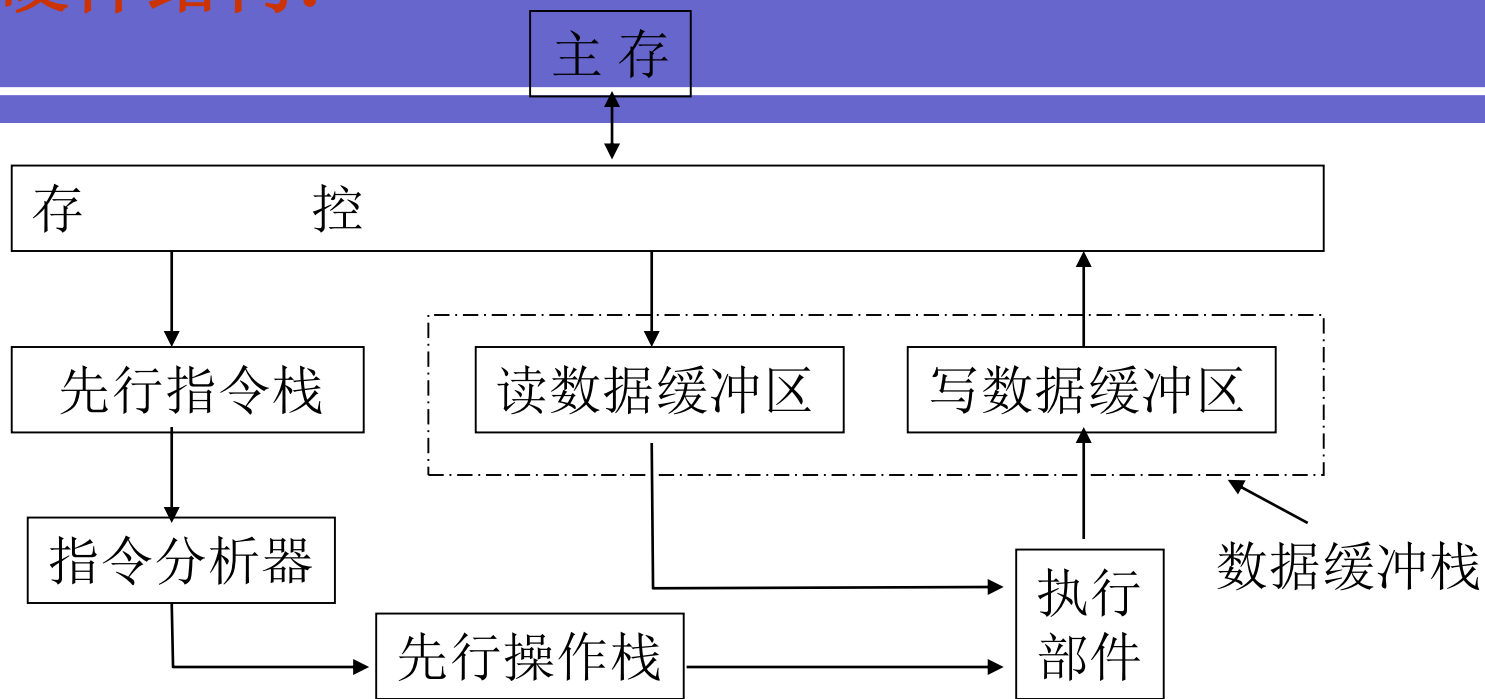
增设指令缓冲栈，消除取指过程；

增设数据缓冲栈，保证不同指令的读、写操作并行；

增设先行操作栈，保证执行部件能连续执行。

结果：解决了分析与执行时间不等长问题。

硬件结构:



栈的深度要求:

$$D_{\text{指缓}} \geq D_{\text{操作}} \geq D_{\text{读栈}} \geq D_{\text{写栈}}$$

3. 相关处理

数据相关、控制相关（条件转移）。

4.1.3流水线处理机

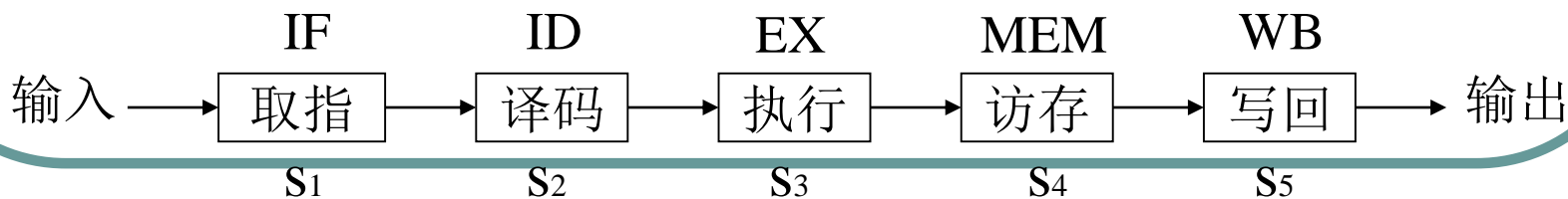
1. 基本思想

延伸重叠方式，使指令解释过程进一步细化，提高各部件的利用率，以提高指令执行速度。

理想目标：完成任务的时间与操作处理过程无关，只与提供操作的速度有关。

2、工作原理

结构：功能部件进一步细化，每段时间为 Δt ，有 m 个段，需要每段间均可重叠，执行 n 条指令。



流水技术原理

- 如果把过程**细分**成多个子过程，并分别由各自独立的条件来实现。每一个 Δt_2 流出一条指令的处理结果。
- 这种工作方式类似于现代工厂的装备流水线，每隔 Δt_i 流水线就流出一个产品，而制造一个产品的时间远远大于 Δt_i ，这就是计算机设计中的“**流水线**”概念。

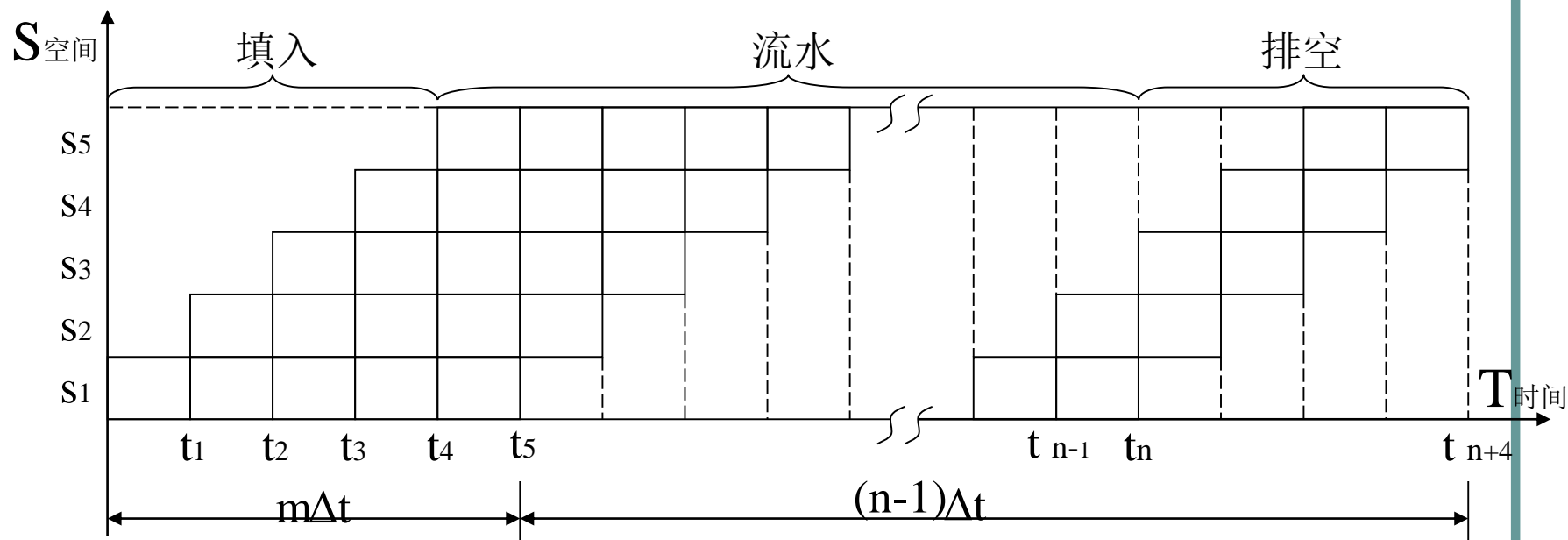
流水线定义

定义：由 k 个处理段(function)线性地逐级串联在一起，外部输入（数据流）馈入流水线的第一段 S_1 ，**处理结果从 S_i 段送到 S_{i+1} 段**（ $i=1, 2, \dots, k-1$ ），最后结果从流水线的最后段 S_k 送出。

流水线执行完成一种固定的功能。（指令执行、算术计算、存储器访问等）

工作流程：分填入、流水、排空三个过程。

描述方法：时-空图。



执行n条执行时间： $T = m \Delta t + (n-1) \Delta t$;

当 $n \gg m$ 时, $T = (n-1) \Delta t$ 。

理想效果：完成任务时间与操作处理过程 (m) 无关，只与提供操作的速度 ($1/\Delta t$) 有关！

同步处理：功能部件+锁存器。

指令执行频率（提供指令的速度）：

$$f = \frac{1}{t_s + t_l} = \frac{1}{\max(\Delta t_{si}) + t_l}$$

特点：

完成任务时间只与 Δt 有关，与处理过程 m 无关；
最大限度地提高了部件的利用率。

3. 硬件要求

独立工作的各子功能部件；

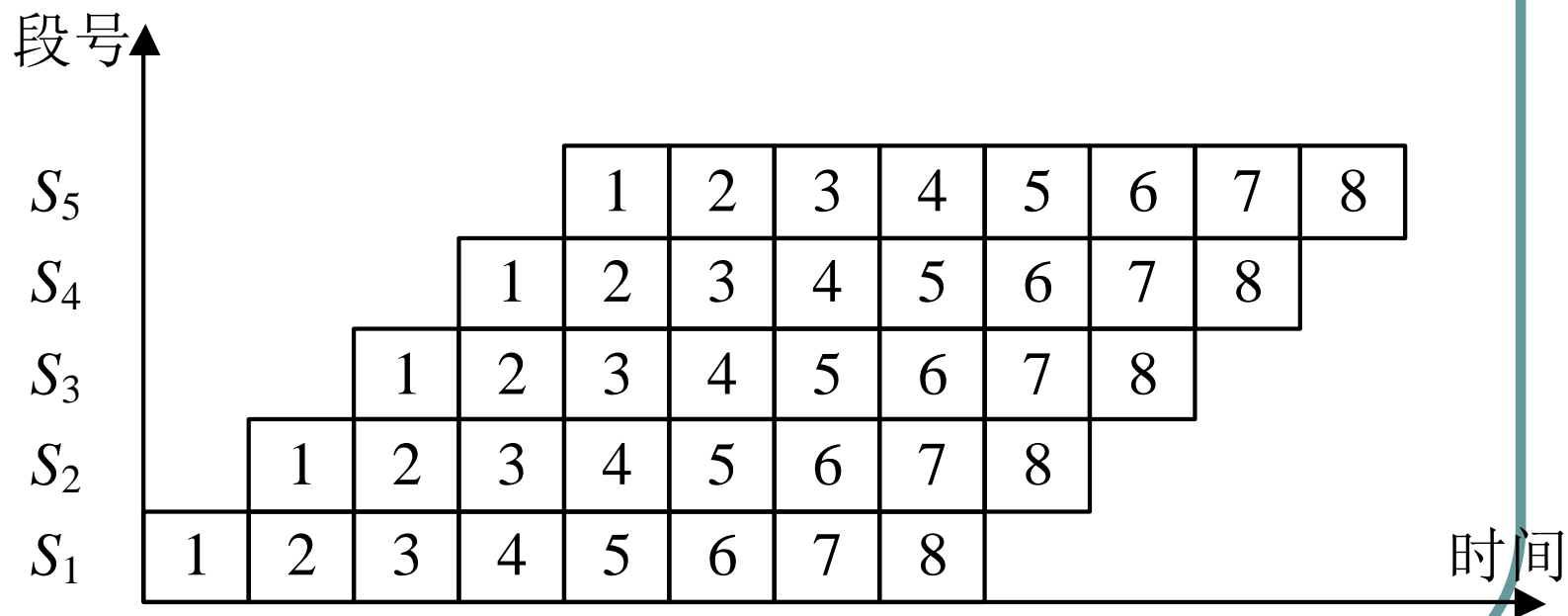
各部件处理时间尽可能相等，争取**最大频率**；

解决同步问题，保证以相同的速度处理；

解决相关问题，包括控制相关和数据相关。

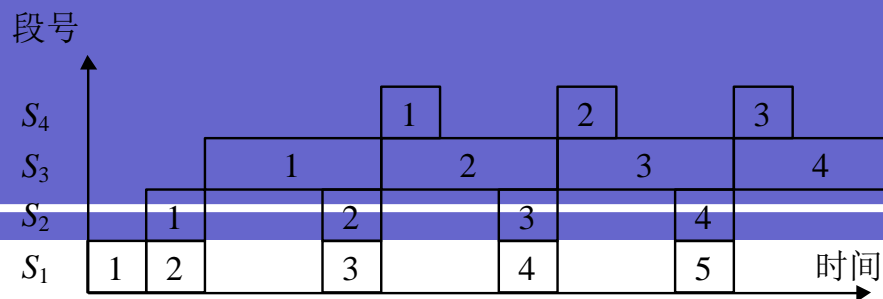
时空图

- 描述流水线的工作过程，采用
时（时间）—空（间）图办法

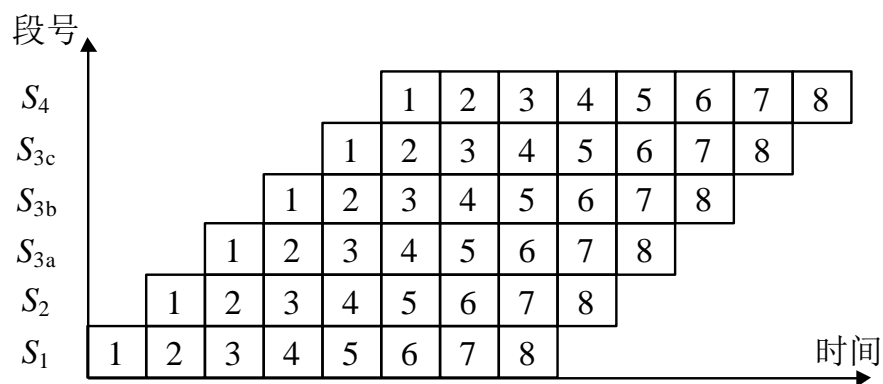


流水线的时空图例子

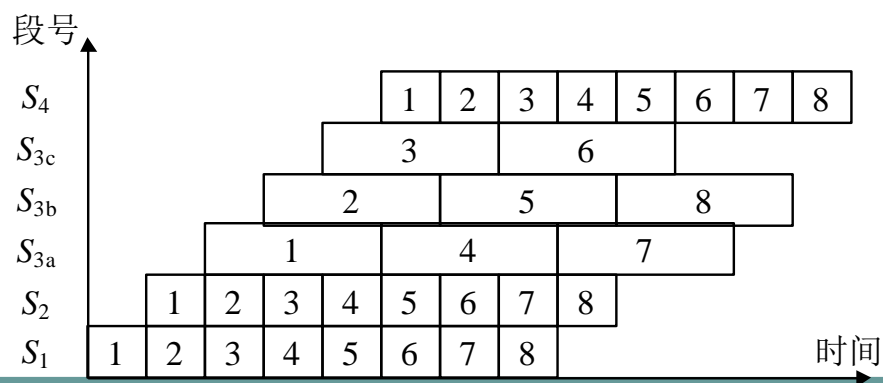
时空图



(a) 原流水线时空图



(b) 流水段细分后的时空图



(c) 流水段重复的时空图

流水技术特点

- (1) **流水线**可以划分为若干个互有联系的子过程（功能段）。每个功能段有专用功能部件实现。
- (2) 实现子过程的功能段所需时间**尽可能相等**，避免因不等产生处理的**瓶颈**，形成流水线“断流”。
- (3) 形成流水处理，需要一段准备时间，称“**通过时间**”。只有在此之后流水过程才能够稳定。
- (4) 指令流发生不能顺序执行时，会使流水过程中断，再形成流水过程，则需通过时间。所以流水过程不应常“断流”，否则效率就不会很高。
- (5) 流水技术适用于大量重复的程序过程，只有在输入端能**连续地提供服务**，流水线效率才能够充分发挥。

● 流水线的主要特点

只有连续提供同类任务才能发挥流水线效率

尽量减少因条件分支造成的“断流”

通过编译技术提供连续的相同类型操作

每个流水线段都要设置一个流水寄存器

时间开销：流水线的执行时间加长

是流水线中需要增加的主要硬件

各流水段的时间应尽量相等

流水线处理机的基本时钟周期等于时间最长的流水段的时间长度。

流水线需要有“装入时间”和“排空时间”

4.流水线分类（属性）

1). 按处理级别分类

操作部件级流水，指令级流水，处理机级流水

2). 按功能分类

单功能流水，多功能流水

3). 按工作方式分类

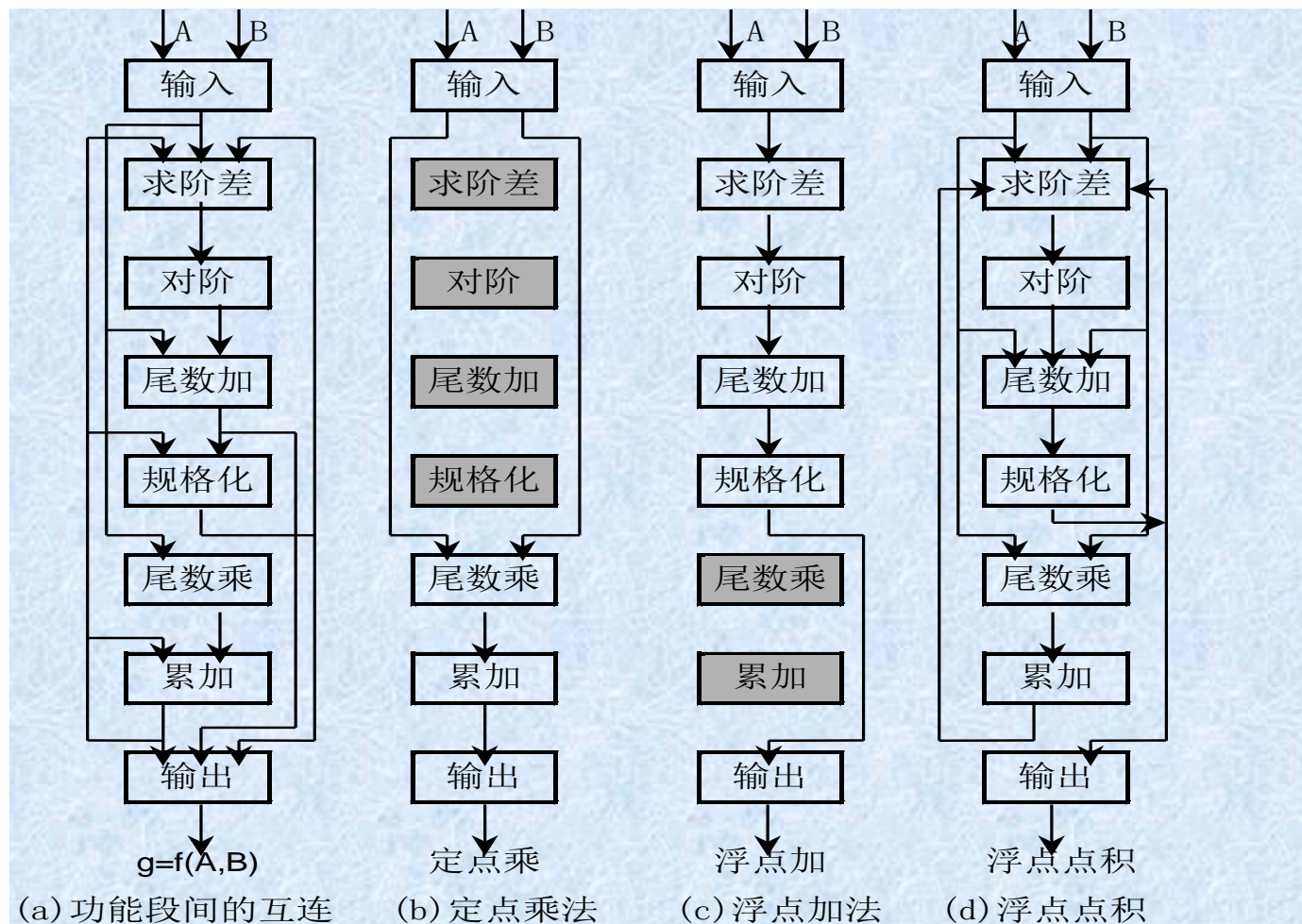
静态流水，动态流水

特点：动态流水线必是多功能流水线；
单功能流水线必是静态流水线。

按功能分类

- **单功能流水线：**只能完成一种功能的流水线。在计算机中要实现多个功能，都采用多个单功能流水线，。
- **多功能流水线：**同一个流水线可有多种连接方式来实现多种功能。
- 一个典型的例子ASC的运算器，它有8个功能段，经适当连接可以实现浮点加、减或乘等功能。

Texas公司的ASC机，8段流水线，能够实现：定点加减法、定点乘法、浮点加法、浮点乘法、逻辑运算、移位操作、数据转换、向量运算等

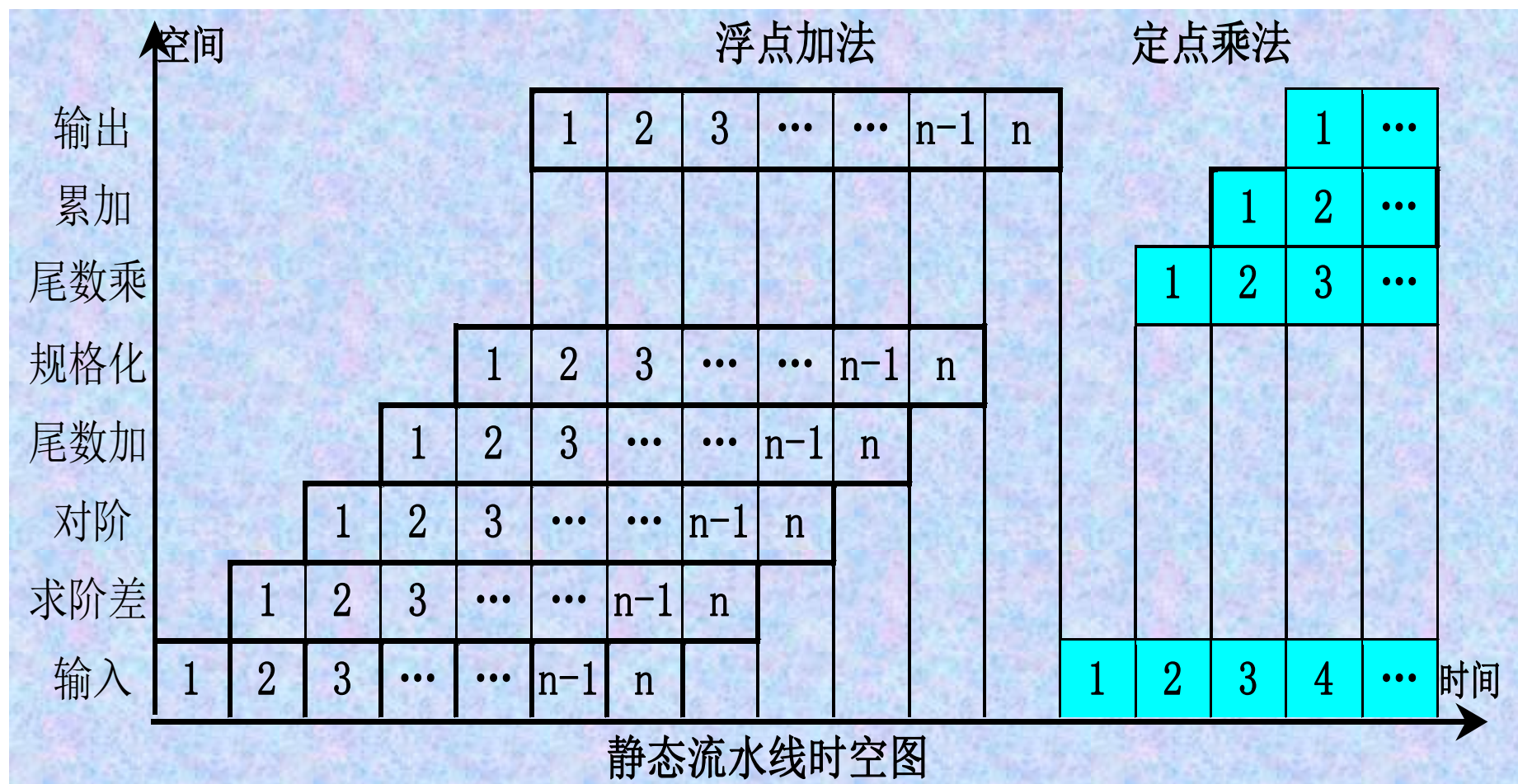


按连接方式分类

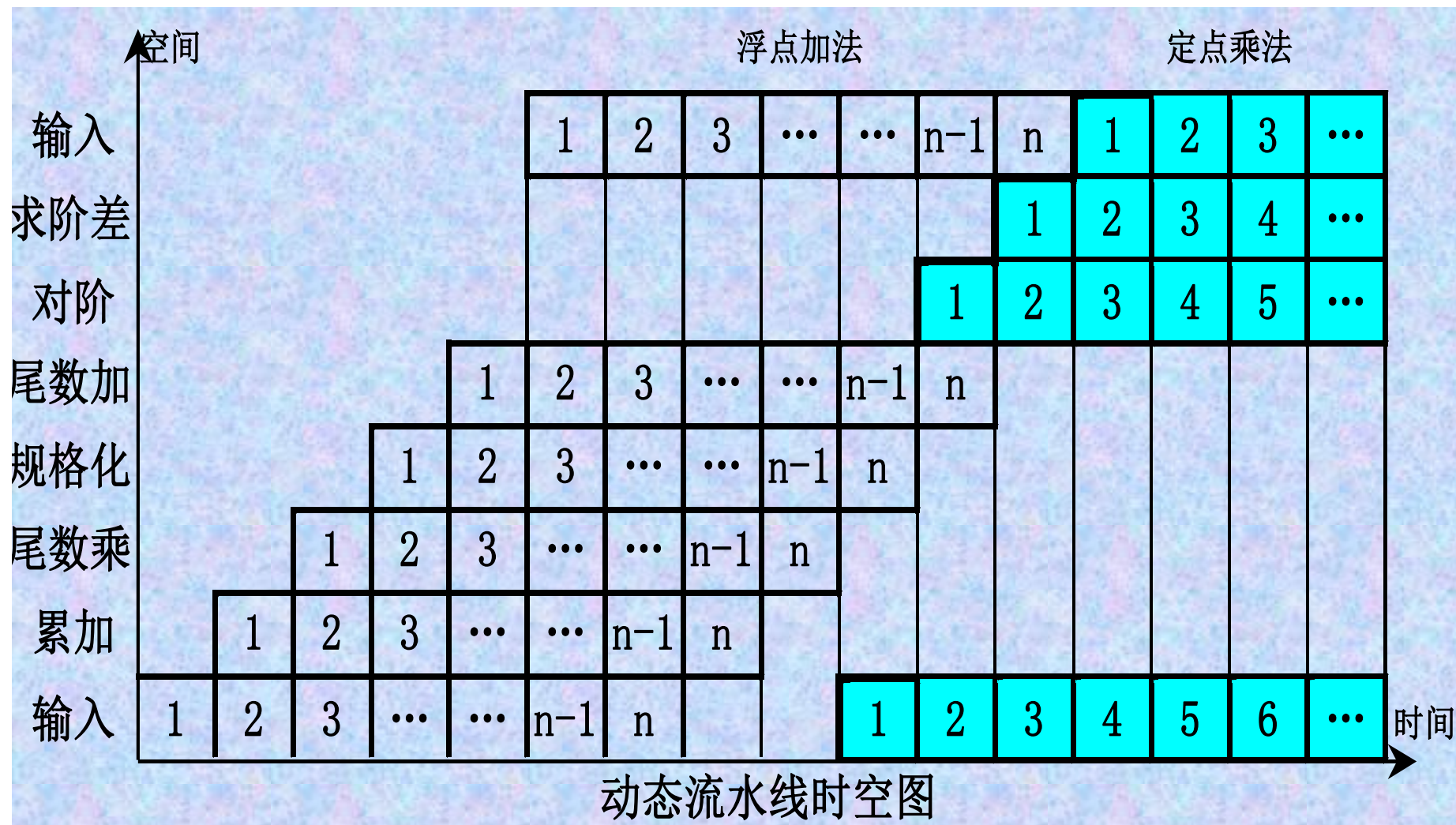
- **静态流水线：** 同一时间内，流水线的各段只能按同一种功能的连接方式工作。
- **动态流水线：** 在同一时间内，流水线的各段可按不同运算的连接方式工作。工作效率提高了，但控制就复杂多了。因此，大多数流水线都是静态的。

静态流水线与动态流水线

静态流水线：同一段时间内，各个功能段只能按照一种方式连接，实现一种固定的功能。



动态流水线：在同一段时间内，各段可以按照不同的方式连接，同时执行多种功能。



按级别分类

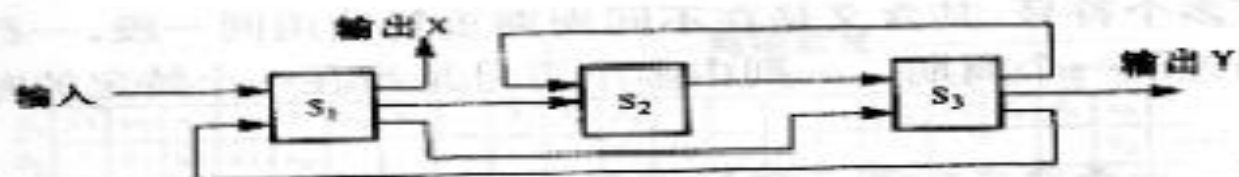
- **部件级流水线**：又称运算器流水线。它是指处理机的算术逻辑部件分段。
- **处理机级流水线**：又称**指令流水线**。它是指在指令解释过程中划分为若干个功能段，按流水方式组织起来。
- **处理机间流水线**：又称宏流水线。它是指两台以上的处理机串行地对同一数据流进行处理，每台处理机完成一个任务。

按数据表示分类

- **标量流水处理机：**只能对标量数据进行处理。
- **向量流水处理机：**它具有向量指令，能对向量的各元素进行流水处理。

按反馈回路分类

- **线性流水线：**流水线各段串行连接，没有反馈回路。
- **非线性流水线：**流水线中除有串行连接通路外，还有反馈回路。在流水过程中，有些段要反复多次使用。



(a) 一条 3 段流水线

→ 时间

	1	2	3	4	5	6	7	8
段 S ₁	X					X		X
段 S ₂		X		X				
段 S ₃			X		X		X	

(b) 功能 X 的预约表

→ 时间

	1	2	3	4	5	6
段 S ₁	Y				Y	
段 S ₂			Y			
段 S ₃		Y		Y		Y

(c) 功能 Y 的预约表

图 6.3 一条有两种不同功能并带有前馈和反馈连接的动态流水线

4.2 线性流水线技术指标

一 流水线相关操作

保持流水线性能条件： 不能停顿或断流。

影响流水线性能因素： 相关和功能切换。
相关类型

1. 局部相关

资源或结构相关；指令相关；数据相关。

结果：流水线停顿，流水线中后续指令有效。

2. 全局相关

转移指令引起的相关。

结果：流水线断流，流水线中后续指令全部作废。

(1) 局部性相关处理

1. 资源冲突（相关）

原因：许多指令争抢同一功能部件。如访存

解决方法：

a. 时间方法——后续指令冲突部件推后一拍执行；

b. 空间方法——重复设置资源。

2. 指令相关

原因：由本指令产生后续指令，即指令可修改引起。

解决方法：

a. 禁止方法——指令不允许修改；

b. 转换方法——把新指令作为“执行”指令的操作数，使指令相关变成数据相关。

3. 数据相关

原因： 对主存数据或通用寄存器数据的操作引起相关。

相关类型一：RAW（先写后读）

有针对存储器RAW和针对通用寄存器RAW两种。

指令 \ 时钟	1	2	3	4	5	6
ADD	IF	ID	EX	MEM	WB	
SUB		IF	ID	EX	MEM	WB

写R1

读R1

相关类型一检测方法：

在流水线读操作数功能段设置一个比较器。

相关类型一解决方法： 延迟执行法（后推法）

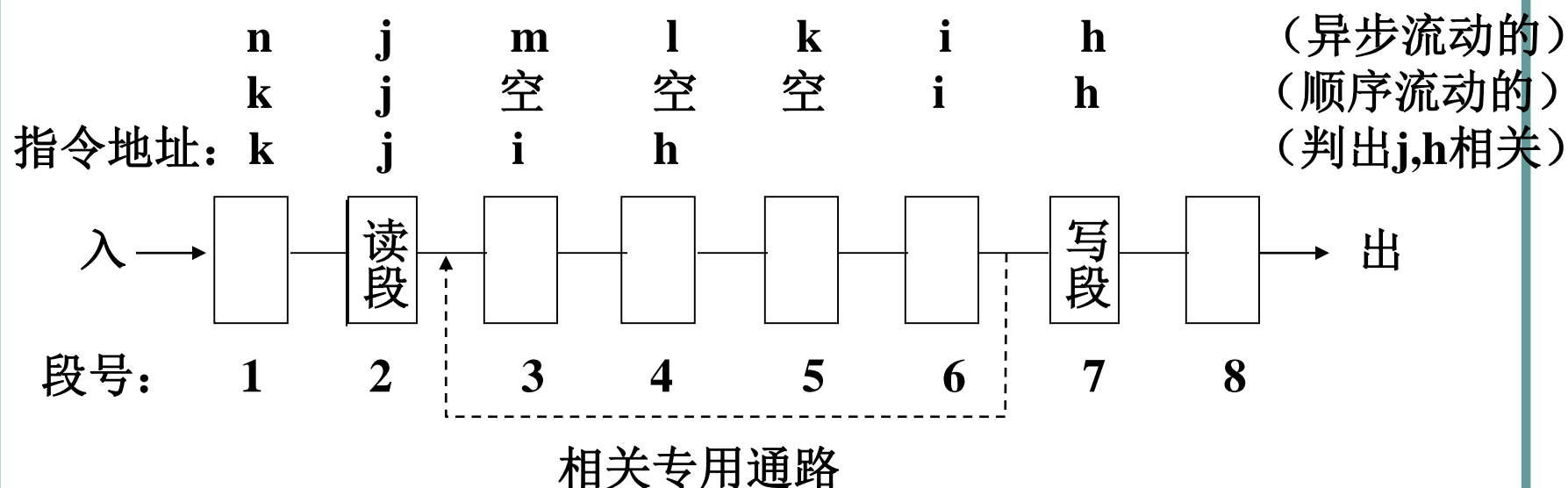
不同拍之间**相关时**，停顿后继指令的运行，直到前面指令结果生成；（R或M相关）

不同拍之间相关时，停顿后继指令的运行，直到前面指令结果生成；（R或M相关）

RISC计算机中，指令的装载延迟，采用联锁硬件检测，并使流水线停顿，直到相关消除。

[1] 顺序流动:

[2] **异步流动**: 让流水线中相关指令的后续不相关指令先执行, 自动消除相关。

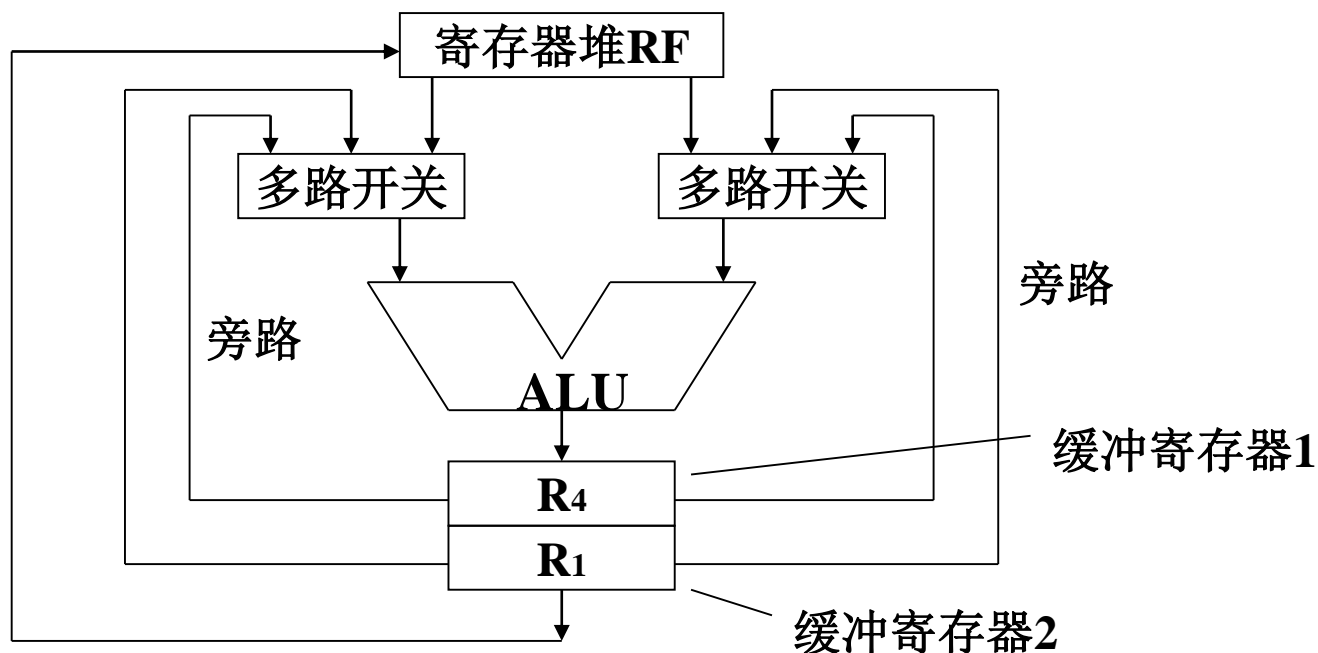


目前, 较多系统采用异步流动的方法。

异步流动会产生新的数据相关类型, 使其控制很复杂。

[3] 相关专用通路

执行结果除写寄存器外，可直接送到ALU的操作数保存栈中。（R相关）



相关类型二：WAR(先读后写)、WAW(写写)

这两种相关只有在异步流动流水线中才会产生。
RAW（先写后读）相关在任何流水线中都会出现。

相关类型二检测方法：

在流水线写操作数功能段设置一个比较器。

相关类型二解决方案：

使用动态调度方法检测和处理相关。
（后续讲解）

(2) 全局性相关处理 相关转移

原因：由转移指令(占总指令的1/4)引起的相关。

解决方法：

程序执行时：

停顿

加速生成结果

猜测结果

程序生成时： 优化延迟转移

[1] 尽早判断转移是否产生，尽早生成转移目标地址
两者相结合，可减少因转移成功产生的停顿时间。

[2] 加快和提前形成条件码

对转移指令的条件码，部分可提前生成；

加快循环内条件码形成。

[3]猜测方法

选取发生概率较高的分支为猜测方向，运行但不写回结果。若猜对，继续执行；否则，作废猜测方向的执行，返回实际转移处。

如何提高猜测命中率

静态猜测法：猜测不成功方向。

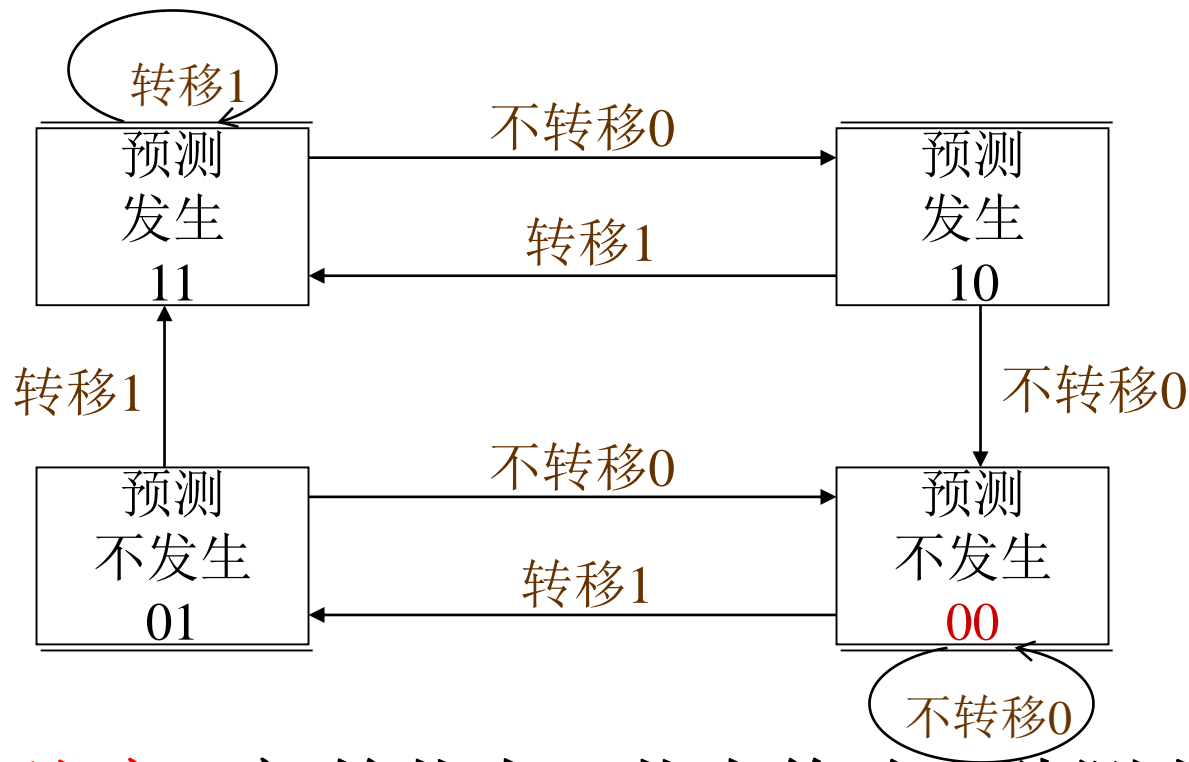
由程序员和编译程序把发生概率高的分支安排在猜测方向。

动态猜测法：根据转移历史猜测。

硬件要求：增设转移目标缓冲器。

转移预测原理:

用2位二进制数记录实际转移状态(历史位)，高位为1时预测转移发生，高位为0时预测转移不发生。



注意: 初始状态、状态修改、猜测方向表示

猜测的后续处理

分支现场的保护及恢复：

采用后援寄存器保存可能被破坏的状态；

猜测执行只完成译码、取操作数或执行，但不写结果（基于硬件的推测执行）。

(3) 流水线的中断处理

中断类型：I/O中断、软件中断、异常中断。

解决问题

中断现场的保护与恢复。

对不同类型中断保护和恢复现场的时间要求不同。

解决方法:

不精确断点法

断点精确到中断请求时最后进入流水线的那条指令

对异常中断的处理:

判断中断产生指令与流水线中其他指令的数据相关性，无数据相关时，其他指令正常流出；否则，流水线中指令全部作废。

对其他中断的处理:

等待流水线中指令执行完毕。

特点: 硬件开销小，控制简单，程序排错不方便。

精确断点法

断点精确到中断请求时已进入流水线的各条指令。

中断处理方法：

平时采用一定数量的后援寄存器，把流水线中所有指令的执行现场保存下来，中断产生时可实现精确断点现场保护与恢复。

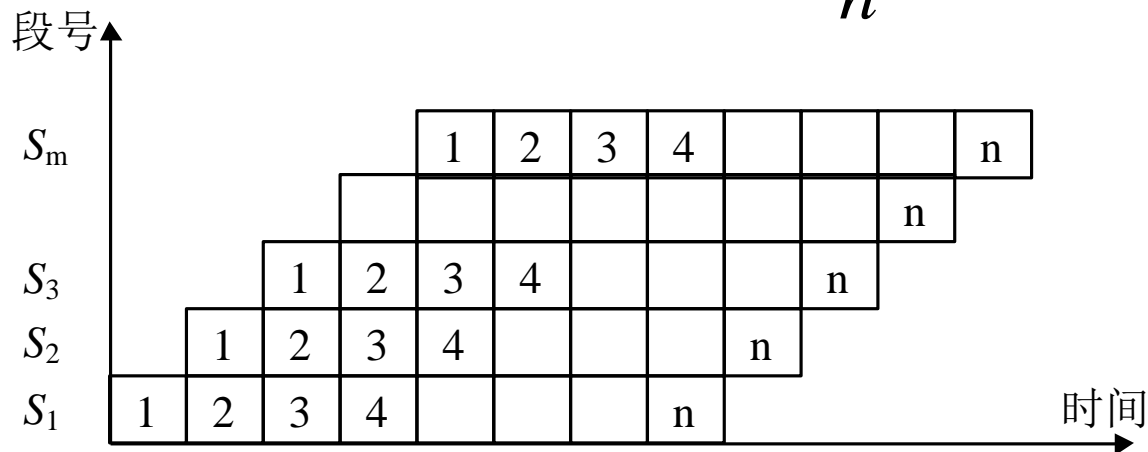
特点：效果理想，硬件代价高，控制复杂。

二 流水线技术指标

1. 吞吐率

对于**线性流水线**，完成 **n** 个任务所需时间为
 $T = m\Delta t + (n-1)\Delta t$ ，实际吞吐率为：

$$T_p = \frac{n}{m\Delta t + (n-1)\Delta t} = \frac{1}{\Delta t(1 + \frac{m-1}{n})} = \frac{T_{p \max}}{1 + \frac{m-1}{n}}$$



线性流水线的性能分析

主要指标：吞吐率、加速比和效率

1. 吞吐率 (Through Put)

流水线吞吐率的最基本公式：

$$TP = \frac{n}{T_k}$$

其中：n为任务数，

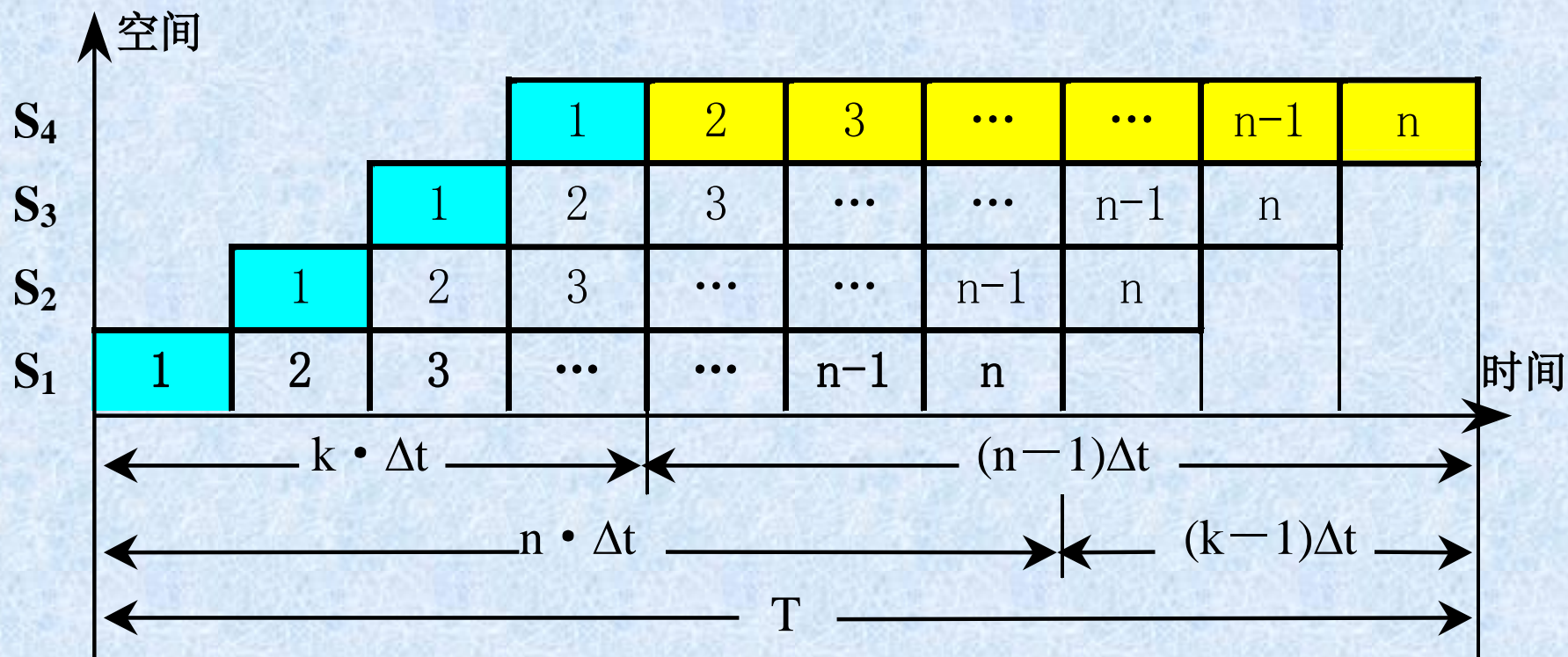
T_k 为完成n个任务所用的时间。

各段执行时间相等，输入连续任务情况下，完成n个任务需要的总时间为：

$$T_k = (k + n - 1)\Delta t$$

其中：k 为流水线的段数， Δt 为时钟周期。

$$T_k = k \cdot \Delta t + (n-1) \Delta t = (k+n-1)\Delta t$$



• 吞吐率为: $TP = \frac{n}{(k+n-1)\Delta t}$

最大吞吐率为: $TP_{\max} = \lim_{n \rightarrow \infty} \frac{n}{(k+n-1)\Delta t} = \frac{1}{\Delta t}$

各段时间不等，完成n个连续任务：

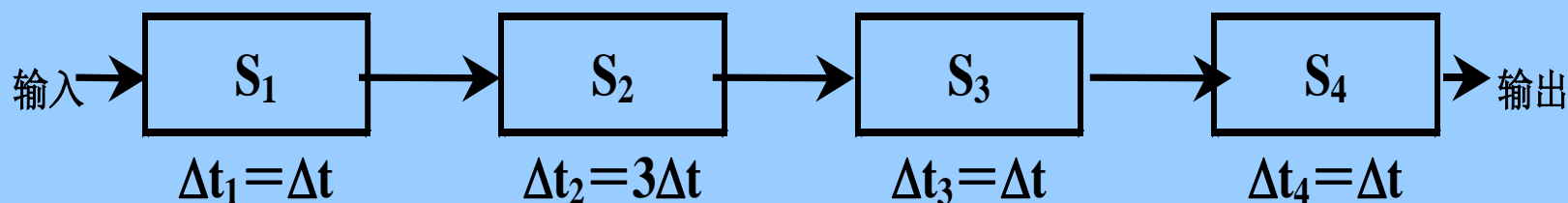
吞吐率：

$$TP = \frac{n}{\sum_{i=1}^k t_i + (n-1) \max(\Delta t_1, \Delta t_2, \dots, \Delta t_k)}$$

最大吞吐率：

$$TP = \frac{1}{\max(\Delta t_1, \Delta t_2, \dots, \Delta t_k)}$$

问题：流水线各段执行时间不相等的解决办法



吞吐率 (Through Put)

单位时间内能处理的指令条数或输出结果的数量

(1) 最大吞吐率

$$T_{P\max} = \frac{1}{\max\{\Delta t_{iexec} + \Delta t_l\}} = \frac{1}{\max\{\Delta t_i\}}$$

当各 Δt_i 相等时, $T_{P\max} = \frac{1}{\Delta t}$

(2) 实际吞吐率

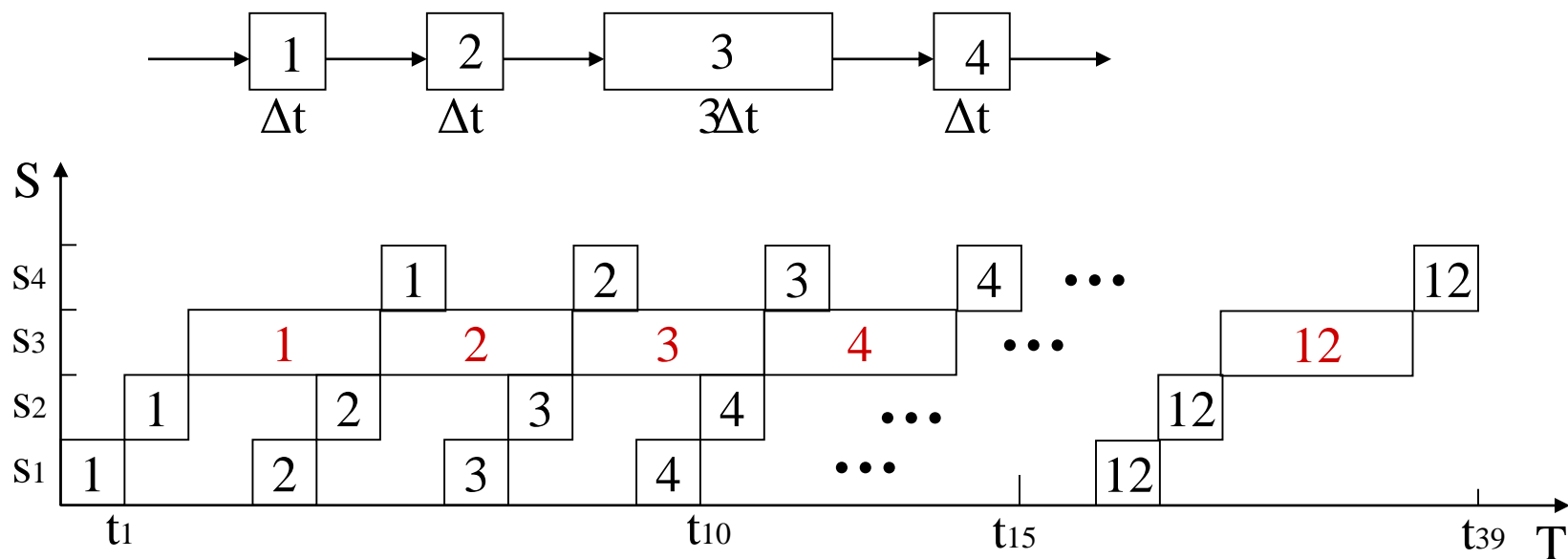
$$T_P = \frac{n}{\sum_{i=1}^m \Delta t_i + (n-1) \max\{\Delta t_i\}}$$

当各 Δt_i 相等时, $T_P = \frac{n}{m\Delta t + (n-1)\Delta t} = \frac{T_{P\max}}{1 + \frac{m-1}{n}}$

(3) 瓶颈段处理

瓶颈段： Δt_i 最大的那一个段。

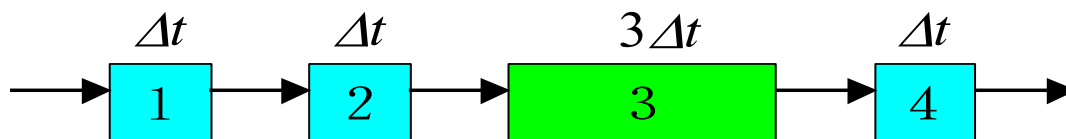
假设流水线结构如下图：



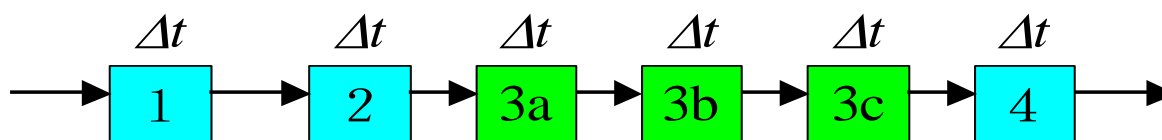
T_P 的具体计算以时空图为准。

$$\text{实际吞吐率 } T_P = \frac{12}{39\Delta t}$$

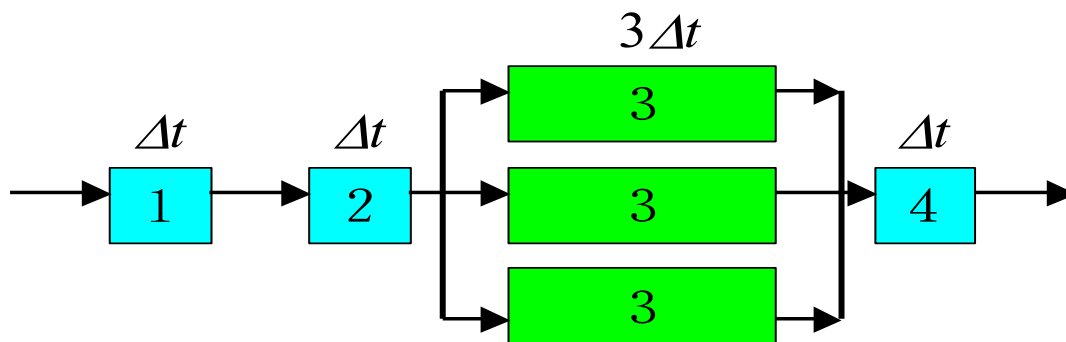
流水线瓶颈的解决 2种方法



(a) 原流水线，第3段为瓶颈



(b) 瓶颈段细分

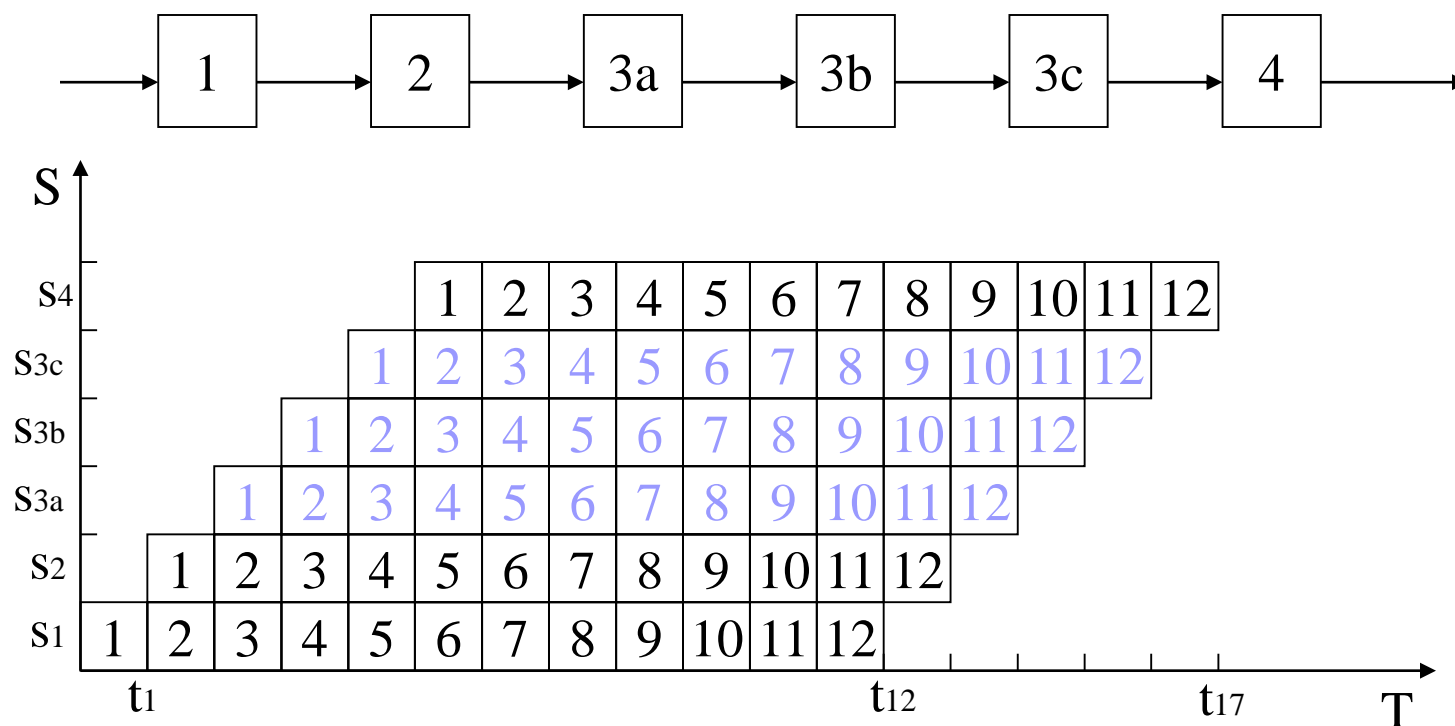


(c) 瓶颈段重复设置部件

消除流水线瓶颈段的两种方法

处理方法一：

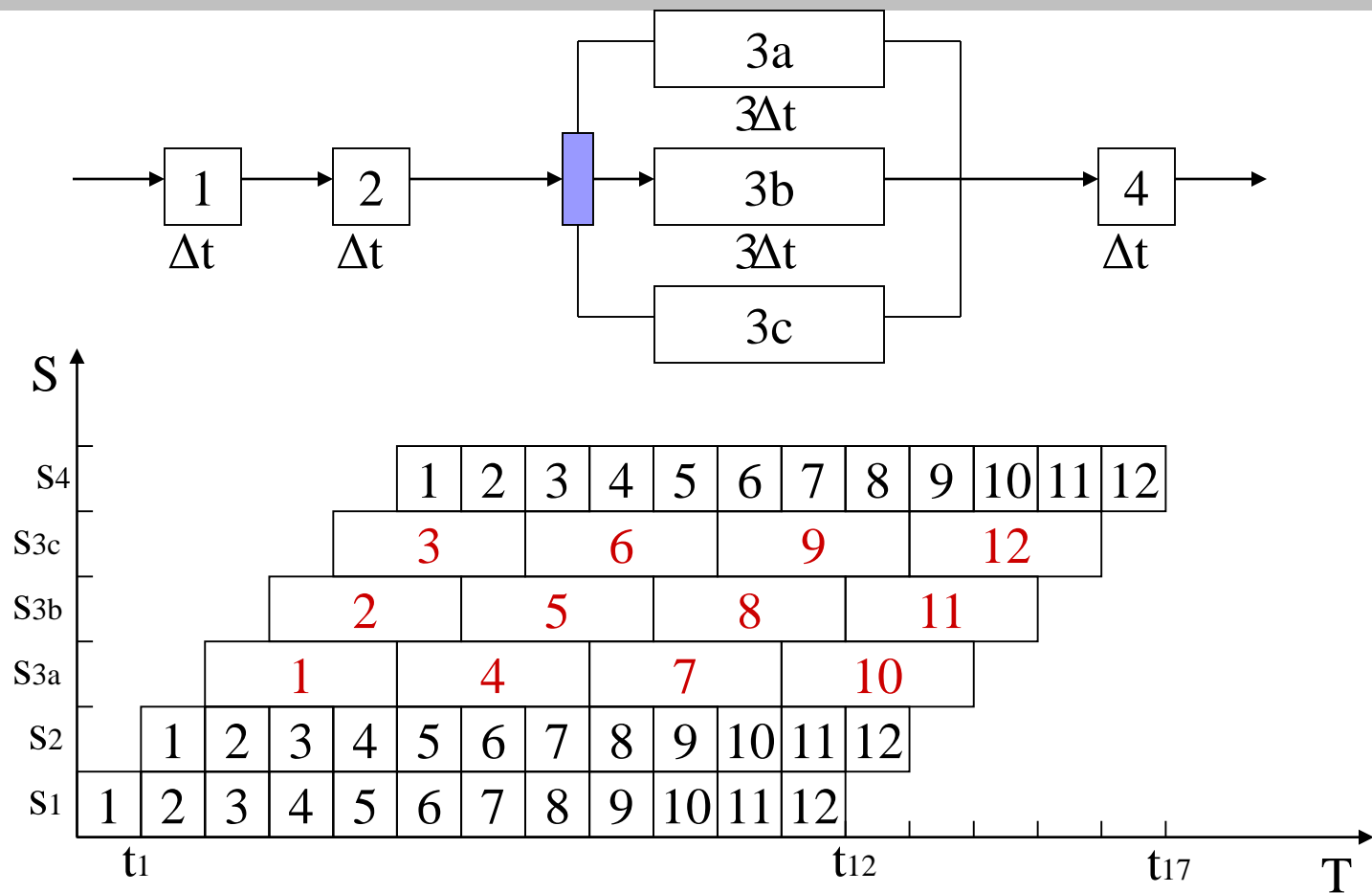
将此段进一步细化，保证细化后的段与其它段处理时间相同或相近；



$$\text{实际吞吐率 } T_P = \frac{12}{17\Delta t}$$

处理方法二：

并联设置多套功能部件，通过控制器进行调度。



提高吞吐率方法：加大流水线深度 m （可减小 Δt ）。

(4) 流水线最佳段数选择

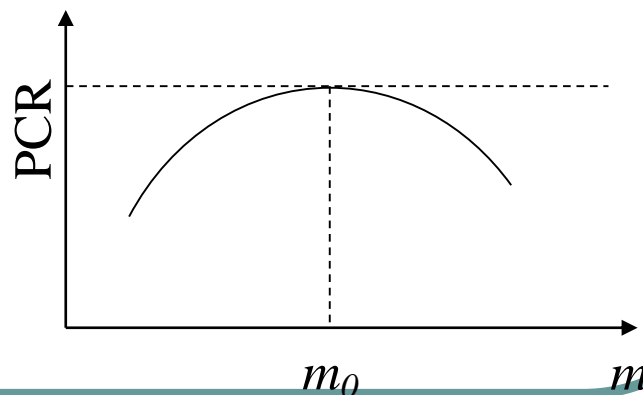
假设流水线各功能段总延迟为 t ，流水线共分 m 个段，
段间同步锁存延迟为 d ；功能段总价格为 a ，每个锁存器价格为 b

$$T_P = \frac{1}{t/m+d}, \quad C=a+bm$$

$$PCR = \frac{T_P}{C} = \frac{1}{t/m+d} \bullet \frac{1}{a+bm}$$

对 m 求导，得性能/价格
PCR 的极大值，此时

$$m_0 = \sqrt{\frac{t \bullet a}{d \bullet b}}$$

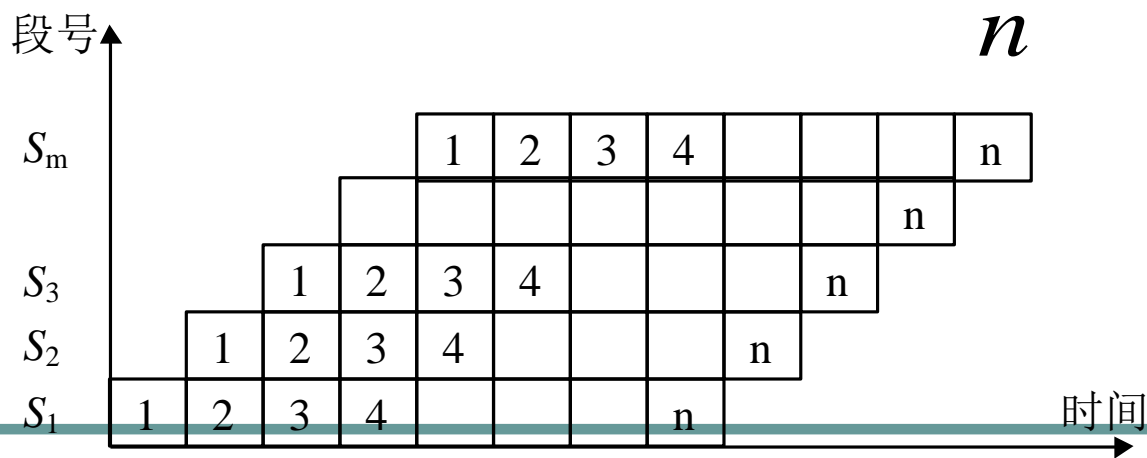


2. 加速比

—流水方式的工作速度与等效的顺序工作方式时间的比值。
对于线性流水线：

$$S = \frac{\text{顺序执行时间 } T_0}{\text{流水线执行时间 } T_k}$$

$$S_p = \frac{T_l}{T_k} = \frac{nm}{m+n-1} = \frac{m}{1 + \frac{m-1}{n}}$$



流水方式相对于串行方式工作速度的比值

$$S_P = \frac{T_{\text{非流水}}}{T_{\text{流水}}} = \frac{n \sum_{i=1}^m \Delta t_i}{\sum_{i=1}^m \Delta t_i + (n-1) \max\{\Delta t_i\}}$$

当各 Δt_i 相等时,
$$S_P = \frac{m\Delta t \bullet n}{m\Delta t + (n-1)\Delta t} = \frac{m}{1 + \frac{n-1}{m}}$$

当 $n \gg m$ 时, $S_P \approx m$ 。

S_P 的具体计算以时空图为准。

如前例流水线 $S_P = \frac{72}{17}$

提高加速比性能方法：加大流水线深度 m 。

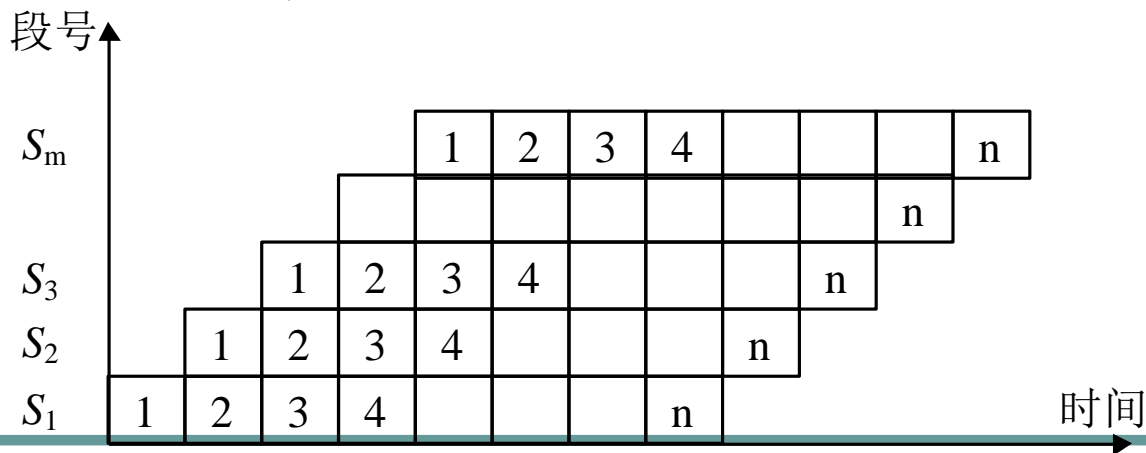
3.使用效率:

工作时间的时空区与流水线中各段总的时空区之比。

对于线性流水线:

$$E = \frac{n \text{个任务占用的时空区}}{k \text{个流水段的总的时空区}} = \frac{T_0}{k \cdot T_k}$$

$$E = \frac{nm\Delta t}{m(m+n-1)\Delta t} = \frac{n}{m+n-1} = \frac{S_p}{m} = T_p\Delta t$$



流水线中设备的利用率

$$E = \frac{n \text{个任务占时空区}}{\text{流水线}n\text{个任务总时空区}} = \frac{n \sum_{i=1}^m \Delta t_i}{m \left[\sum_{i=1}^m \Delta t_i + (n-1) \max \{ \Delta t_i \} \right]}$$

$$\text{当各 } \Delta t_i \text{ 相等时, } E = \frac{n \bullet m \Delta t}{m(m+n-1)\Delta t} = \frac{S_P}{m} = T_P \bullet \Delta t$$

当 $n \gg m$ 时, $E \approx 1$ 。

特点：仅当流水线为线性且 Δt_i 相等时，效率与吞吐率、加速比呈线性关系；否则三者间不存在关系。

E的具体计算以时空图为准。 如前例 $E = 12/17$

提高效率性能方法：增加流水线中任务数量。

- Instruction Set Architecture
 - Intel 80x86
 - MIPS

Intel IA-32 (80x86/Pentium/Xeon)

- Intel 80x86 Registers

Names		31	0	Use
EAX				GPR 0
ECX				GPR 1
EDX				GPR 2
EBX				GPR 3
ESP				GPR 4
EBP				GPR 5
ESI				GPR 6
EDI				GPR 7
	CS			Code segment pointer
	SS			Stack segment pointer
	DS			Data segment pointer 0
	ES			Data segment pointer 1
	FS			Data segment pointer 2
	GS			Data segment pointer 3
EIP				Instruction pointer (PC)
EFLAGS				Condition codes

Typical 80x86 Instruction Formats

- 指令不规整，不利于译码及实现流水

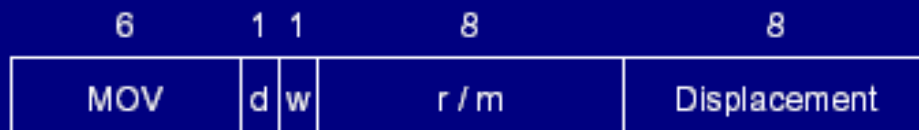
(1) JE EIP + displacement



(2) CALL



(3) MOV EBX, [EDI + 45]

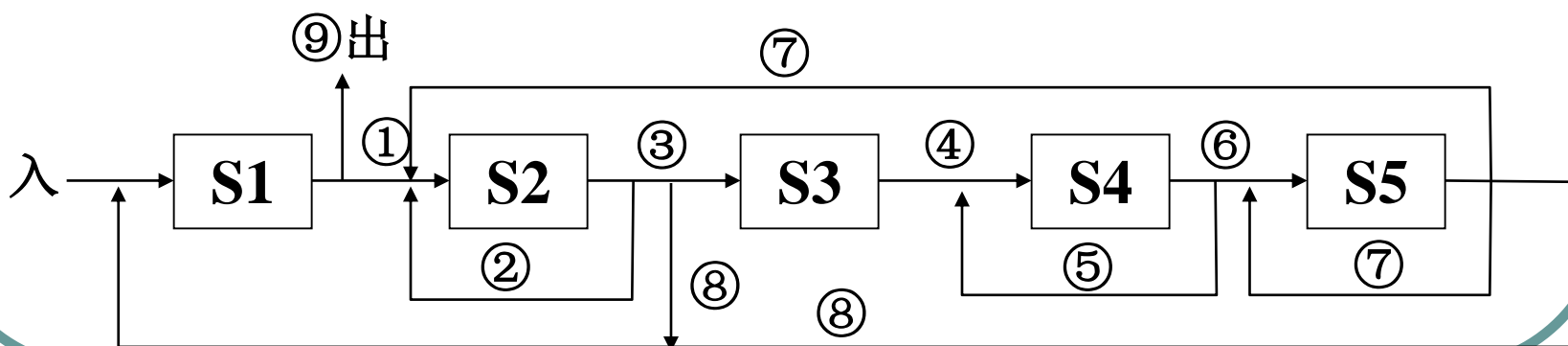


4.3 非线性流水线的调度技术

线性流水线：简单（每拍流入一条指令）；

非线性流水线：因功能段使用冲突，调度相对复杂。

某流水线结构如下：



非线性流水线的调度

非线性流水线调度的任务是要找出一个最小的循环周期，按照这周期向流水线输入新任务，流水线的各个功能段都不会发生冲突，而且流水线的吞吐率和效率最高。

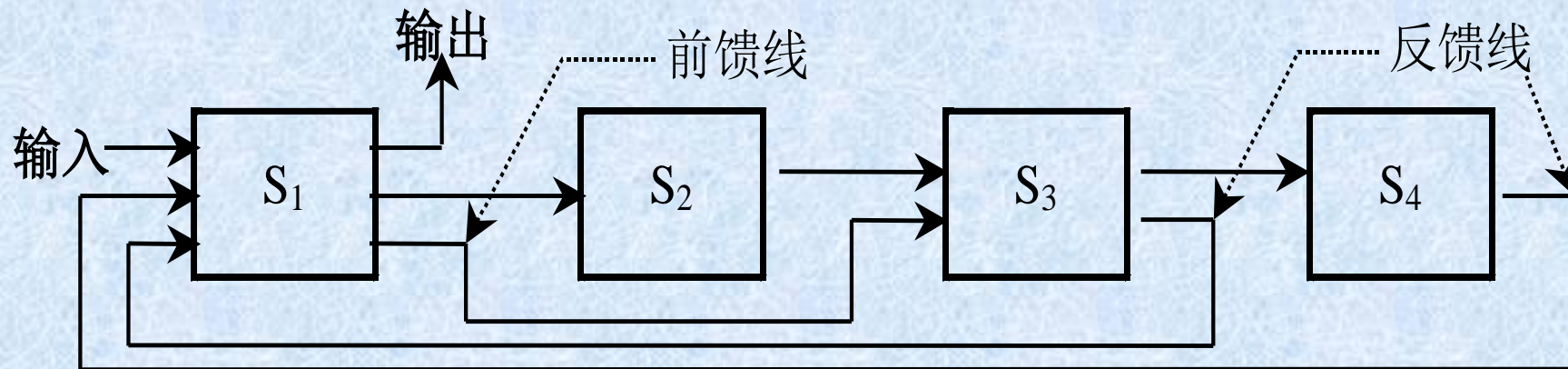
1. 非线性流水线的表示

线性流水线能够用流水线连接图唯一表示

对于非线性流水线，连接图不能唯一表示工作流程，因此，引入流水线预约表

例如：

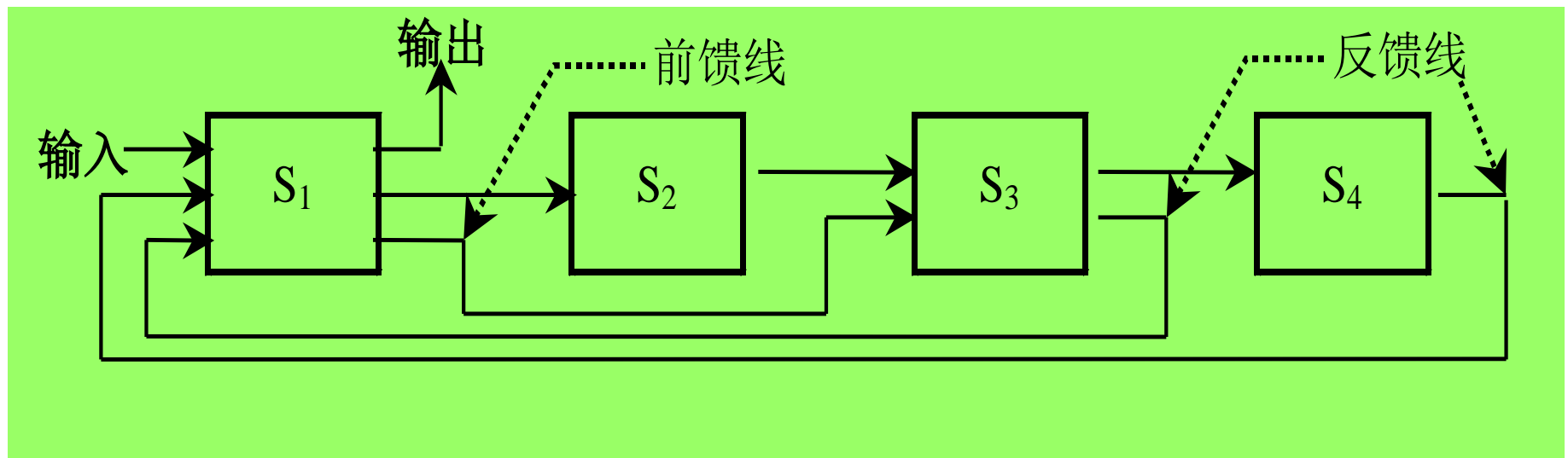
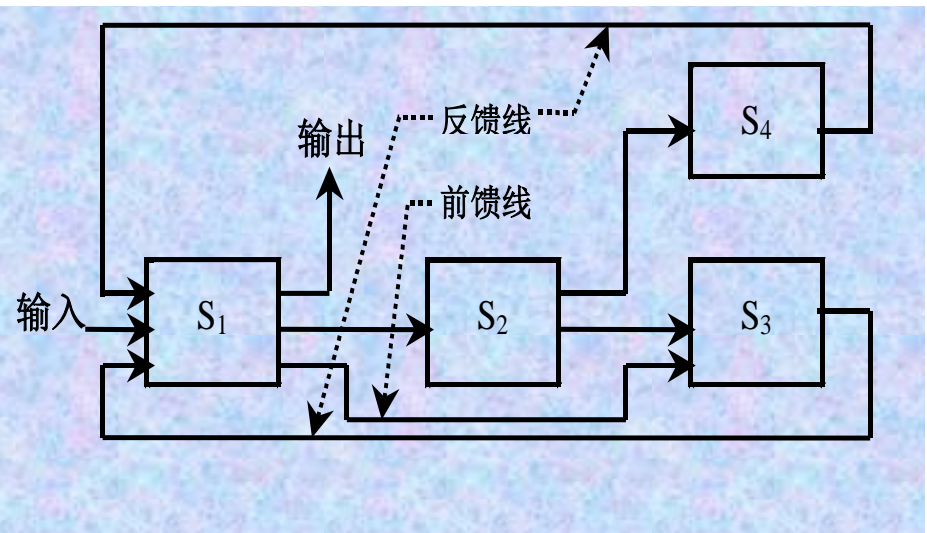
非线性流水线的连接图和预约表



时间 / 功能段	1	2	3	4	5	6	7
S_1	×			×			×
S_2		×			×		
S_3		×				×	
S_4			×				

一张预约表可能与多个流水线连接图相对应

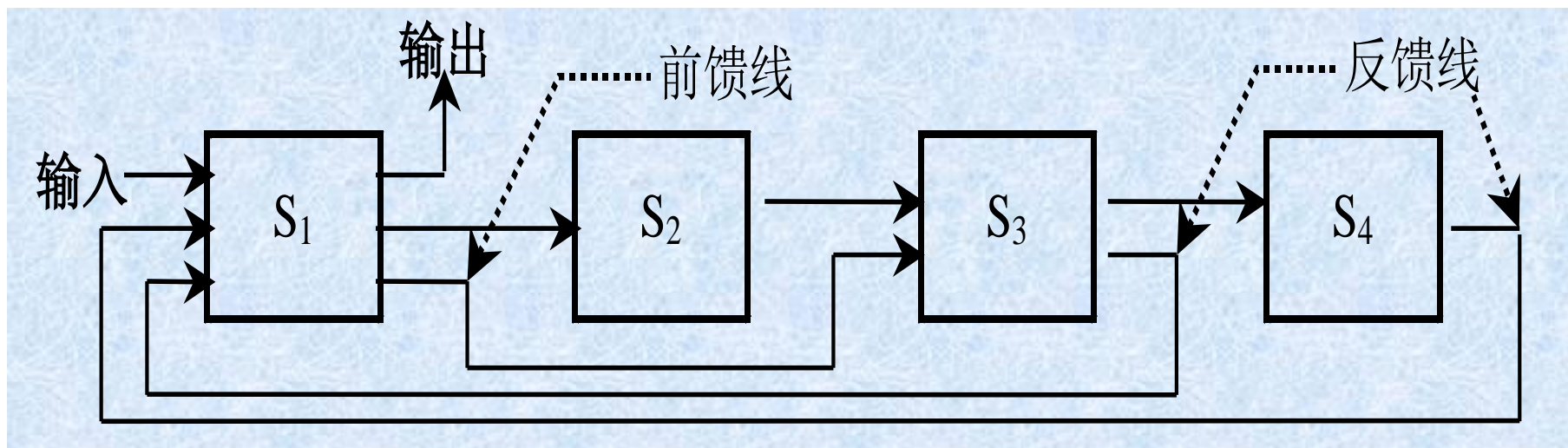
时间 功能段	1	2	3	4	5	6	7
S_1	×			×			×
S_2		×			×		
S_3		×				×	
S_4			×				



一个流水线连接图对应与多张预约表

时间 功能段	1	2	3	4	5	6	7
S ₁	×			×			×
S ₂		×			×		
S ₃		×				×	
S ₄			×				

时间 功能段	1	2	3	4	5	6	7
S ₁	×				×		×
S ₂		×					
S ₃			×			×	
S ₄				×			



2. 非线性流水线的冲突

启动距离：连续输入两个任务之间的时间间隔

流水线冲突：几个任务争用同一个流水段

启动距离为 3 的流水线冲突情况

时间 功能段	1	2	3	4	5	6	7	8	9	10	11	...
S ₁	X ₁			X ₁ X ₂			X ₁ X ₂ X ₃			X ₂ X ₃ X ₄		...
S ₂		X ₁			X ₁ X ₂			X ₂ X ₃			X ₃ X ₄	...
S ₃		X ₁			X ₂	X ₁		X ₃	X ₂		X ₄	...
S ₄			X ₁			X ₂			X ₃			...

启动距离为 2 的流水线冲突情况

时间 功能段	1	2	3	4	5	6	7	8	9	10	11	...
S ₁	X ₁		X ₂	X ₁	X ₃	X ₂	X ₁ X ₄	X ₃	X ₂ X ₅	X ₄	X ₃ X ₆	...
S ₂		X ₁		X ₂	X ₁	X ₃	X ₂	X ₄	X ₃	X ₅	X ₄	...
S ₃		X ₁		X ₂		X ₁ X ₃		X ₂ X ₄		X ₃ X ₅		...
S ₄			X ₁		X ₂		X ₃		X ₄		X ₅	...

启动距离为 5 时的流水线不冲突

时间 功能段	1	2	3	4	5	6	7	8	9	10	11	...
S ₁	X ₁			X ₁		X ₂	X ₁		X ₂		X ₃	...
S ₂		X ₁			X ₁		X ₂			X ₂		...
S ₃		X ₁				X ₁	X ₂				X ₂	...
S ₄			X ₁					X ₂				...

← 启动周期 → ← 重复启动周期 →

启动距离为 (1, 7) 循环时的流水线预约表

时间 功能段	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
S ₁	X ₁	X ₂		X ₁	X ₂		X ₁	X ₂	X ₃	X ₄		X ₃	X ₄		X ₃	X ₄	...
S ₂		X ₁	X ₂		X ₁	X ₂				X ₃	X ₄		X ₃	X ₄			...
S ₃		X ₁	X ₂			X ₁	X ₂			X ₃	X ₄			X ₃	X ₄		...
S ₄			X ₁	X ₂							X ₃	X ₄					...

← 启动周期 → ← 重复启动周期 →

3. 无冲突调度方法

由E. S. Davidson及其学生于1971年提出

禁止向量：预约表中每一行任意两个“×”之间距离的集合。上例中为 **(3, 4, 6)**

冲突向量： $C = (C_m C_{m-1} \dots C_2 C_1)$

其中： m 是禁止向量中的最大值。

如果 i 在禁止向量中，则 $C_i = 1$ ，否则 $C_i = 0$ 。

上例中 **$C = (101100)$**

时间 功能段	1	2	3	4	5	6	7
S ₁	×			×			×
S ₂		×			×		
S ₃		×				×	
S ₄			×				

例1 一条4功能段的非线性流水线，每个功能段的延迟时间都相等，它的预约表如下：

- (1) 写出流水线的禁止向量和初始冲突向量。
- (2) 画出调度流水线的状态图。
- (3) 求最小启动循环和最小平均启动距离。
- (4) 求平均启动距离最小的恒定循环。

时间 功能段	1	2	3	4	5	6	7
S ₁	X						X
S ₂		X				X	
S ₃			X		X		
S ₄				X			

解:

(1) 禁止向量为: (2, 4, 6)

初始冲突向量: $S = 101010$

(2) 构造状态图

S逻辑右移2、4、6位时, 不作任何处理,
逻辑右移1、3、5和大于等于7时:

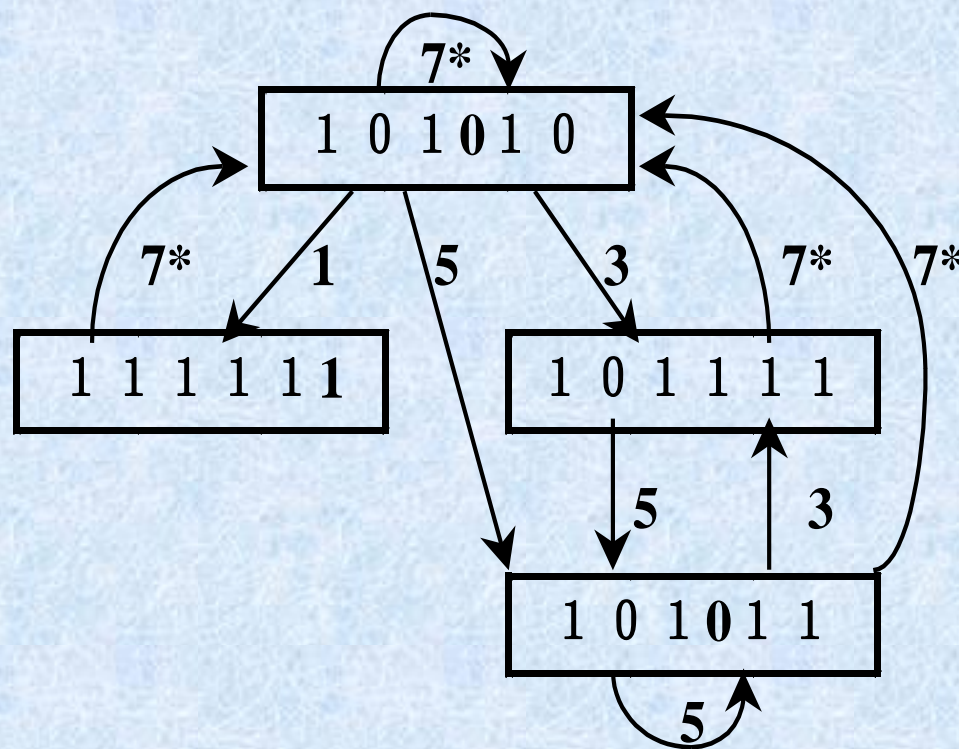
S右移1位之后: $010101 \vee 101010 = 111111$,

S右移3位之后: $000101 \vee 101010 = 101111$,

S右移5位之后: $000001 \vee 101010 = 101011$,

S右移7位或大于7位后: 还原到它本身。

101111右移5位之后: $000001 \vee 101010 = 101011$,
 101011右移3位之后: $000101 \vee 101010 = 101111$,
 101011右移5位之后: $000001 \vee 101010 = 101011$ 。



非线性流水线的状态图

简单循环： 状态图中各种冲突向量只经过一次的启动循环。

(3) 最小的启动循环为 (1, 7) 和 (3, 5) ,
平均启动距离为 4。

(4) 启动距离最小的恒定循环为 (5)

简单循环	平均启动距离
(1, 7)	4
(3, 7)	5
(5, 7)	6
(3, 5, 7)	5
(5, 3, 7)	5
(3, 5)	4
(5)	5
(7)	7

最小启动循环(3, 5)的流水线工作状态

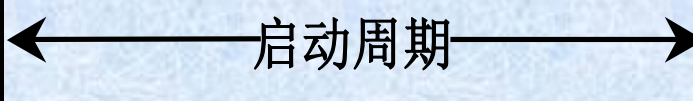
时间 功能段	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
S ₁	X ₁			X ₂			X ₁		X ₃	X ₂		X ₄			X ₃	...
S ₂		X ₁			X ₂	X ₁			X ₂	X ₃			X ₄	X ₃		...
S ₃			X ₁		X ₁	X ₂		X ₂			X ₃		X ₃	X ₄		...
S ₄				X ₁			X ₂					X ₃			X ₄	...

最小启动循环(1, 7)的流水线工作状态

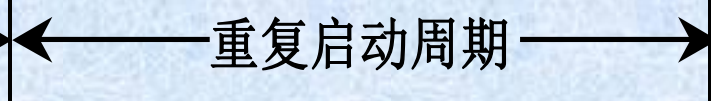
时间 功能段	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
S ₁	X ₁	X ₂					X ₁	X ₂	X ₃	X ₄					X ₃	...
S ₂		X ₁	X ₂			X ₁	X ₂			X ₃	X ₄			X ₃	X ₄	...
S ₃			X ₁	X ₂	X ₁	X ₂					X ₃	X ₄	X ₃	X ₄		...
S ₄				X ₁	X ₂							X ₃	X ₄			...

恒定启动循环(5)的流水线工作状态

时间 功能段	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	...
S_1	X_1					X_2	X_1				X_3	X_2				...
S_2		X_1				X_1	X_2				X_2	X_3				...
S_3			X_1		X_1			X_2		X_2			X_3		X_3	...
S_4				X_1					X_2					X_3		...



启动周期

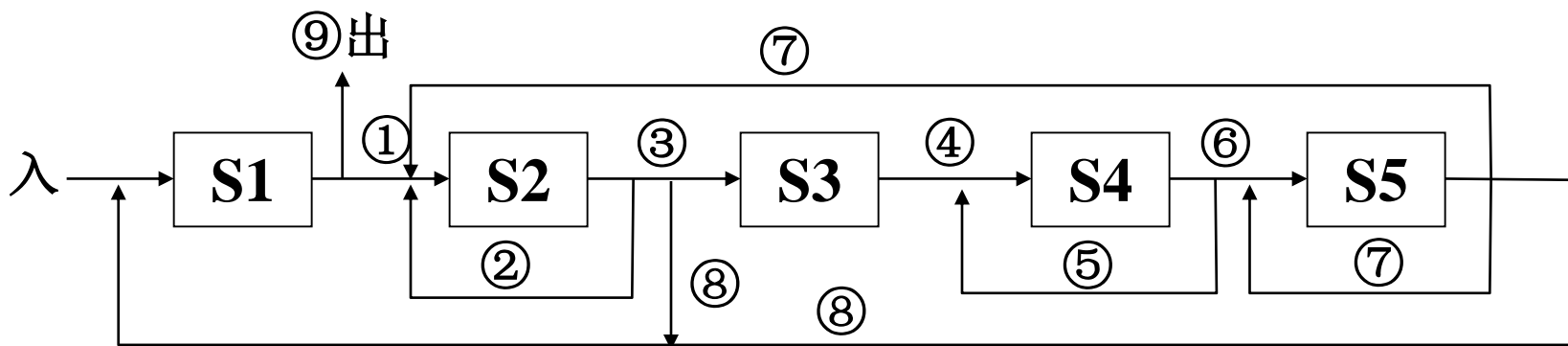


重复启动周期

例2：非线性流水线的调度技术

单功能非线性流水线调度

某流水线结构如下：



流水线调度方案步骤如下：

- ①形成预约表；
- ②由预约表形成禁止表；
- ③由禁止表形成初始冲突向量；
- ④由初始冲突向量形成状态转换图；
- ⑤由状态转换图中找出最佳调度方案。

(1) 形成预约表

方法：指令总拍数为 n ，流水线有 k 个段，则形成 $n \times k$ 的预约表，段的使用情况用“ \times ”表示。

预约表如下：

$\begin{matrix} t \\ \backslash \\ S \end{matrix}$	1	2	3	4	5	6	7	8	9
1	\times								\times
2		\times	\times					\times	
3				\times					
4					\times	\times			
5							\times	\times	

- 与时空图区别？
- 动态流水线可以由一张以上的预约表来表示。
- 每张预约表表示的是某一功能求值的数据通过流水线的时空流
- 预约表一行可以有多个符号,含义是在不同周期重复使用同一段；一列中多个符号是指在一个特定的时钟周期内并行使用多个周期。

(2) 由预约表形成禁止表F

- 等待时间：一条流水线的两次启动之间的时间单位数（时钟周期）
- **冲突**：一条流水线的两次启动之间的资源冲突
- 禁止等待时间：有些等待时间将会引起冲突
- 禁止等待时间测算方法：检查预约表中同一行中任意两个符号之间的距离
- 例如（表）：**8, 6, 5, 1**
- $F = \{\text{各段中冲突间隔拍数}\}$
- $F = \{8, 6, 5, 1\}$

s1	9-1=8		
s2	8-2=6	8-3=5	3-2=1
s3			
s4	6-5=1		
s5	8-7=1		

$s \backslash t$	1	2	3	4	5	6	7	8	9
1	×								×
2		×	×					×	
3				×					
4					×	×			
5							×	×	

1

1

6

1

5

8

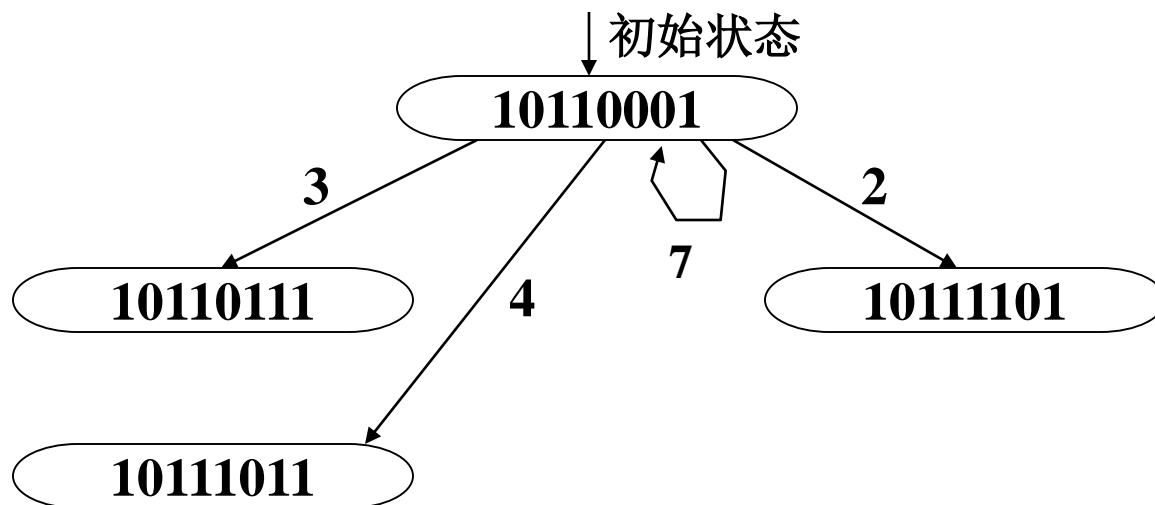
s1	9-1=8		
s2	8-2=6	8-3=5	3-2=1
s3			
s4	6-5=1		
s5	8-7=1		

(3) 由禁止表F形成初始冲突向量 C_0

- 冲突向量 C_0 是一个 m 位的二进制
- 对于 n 列的预约表最大禁止等待时间 $m \leq n-1$
- 可允许等待时间 p $1 \leq p \leq m-1$
- $C_0 = (c_N \dots c_0)$, $c_i=1$ 冲突, $=0$ 不冲突。
- 本例: $C_0 = (10110001)$
- C_0 产生方法根据禁止表F的禁止等待时间在 C_0 相应位置1, 其余位置0

位	8	7	6	5	4	3	2	1
8	1							
6			1					
5				1				
1								1
C0	1	0	1	1	0	0	0	1

(4) 由初始冲突向量 C_0 形成状态转换图

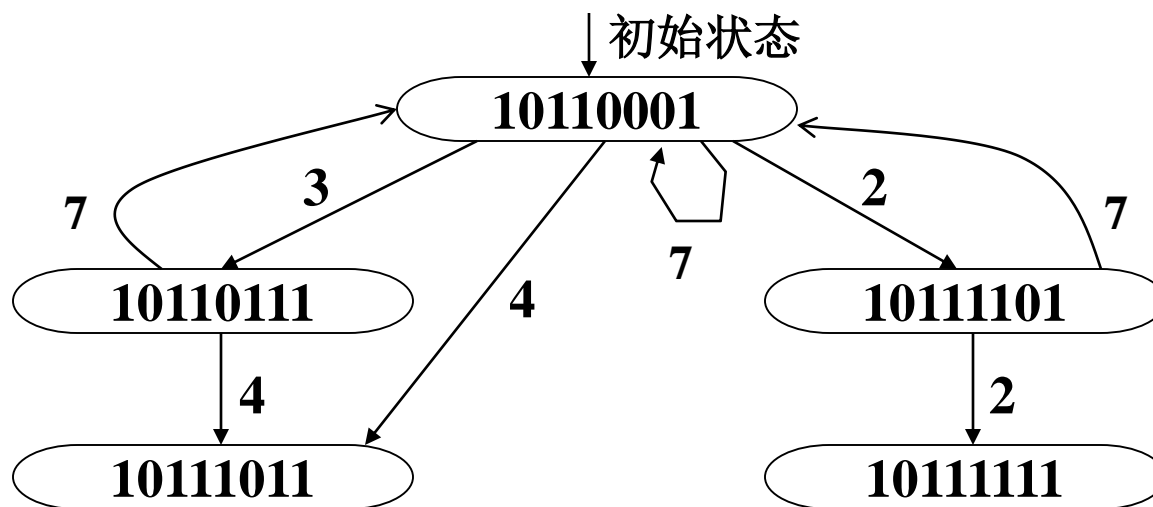


- a. C_0 每过一拍**逻辑右移一位，若移出0**，则允许后续指令进入流水线，再与 C_0 按位“**或**”，形成新的冲突向量 C_i ；
- （根据禁止表F的允许等待时间，
 - 本例：2，3，4，7，就是 C_0 中0的位置）

- 方法：冲突向量C₀经过移位后和后续指令（仍然是冲突向量C₀）求或，得出新的冲突向量C_i

C ₀	1	0	1	1	0	0	0	1
移位3				1	0	1	1	0
或	1	0	1	1	0	1	1	1
C ₀	1	0	1	1	0	0	0	1
移位4					1	0	1	1
或	1	0	1	1	1	0	1	1
C ₀	1	0	1	1	0	0	0	1
移位2			1	0	1	1	0	0
或	1	0	1	1	1	1	0	1
C ₀	1	0	1	1	0	0	0	1
移位7								1
或	1	0	1	1	0	0	0	1

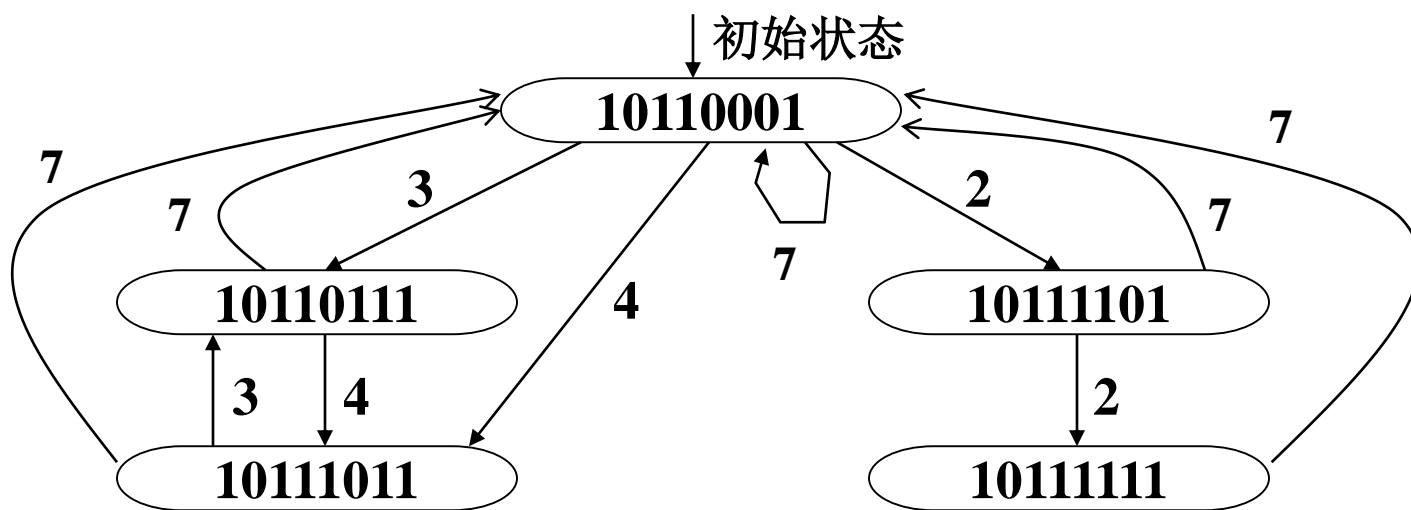
- b. 各 C_i 再每过一拍逻辑右移一位，若移出0，允许后续指令进入，再与 C_0 按位“或”，形成新的冲突向量 C_{ij} ；（如何判断？可以根据 C_i 中的0）



● **方法：**冲突向量 C_i 经过移位后（可以根据 C_i 中的0是第几位就移几位）和后续指令（仍然是冲突向量 C_0 ，因为每次新启动的指令都是具有冲突向量 C_0 ）求或，得出新的冲突向量 C_{ij}

C0	1	0	1	1	0	0	0	1
Ci移位4					1	0	1	1
或	1	0	1	1	1	0	1	1
C0	1	0	1	1	0	0	0	1
Ci移位7								1
或	1	0	1	1	0	0	0	1
C0	1	0	1	1	0	0	0	1
Ci移位2			1	0	1	1	0	0
或	1	0	1	1	1	1	0	1
C0	1	0	1	1	0	0	0	1
Ci移位7								1
或	1	0	1	1	0	0	0	1

- c. 重复上一步骤，直到不再生成新的冲突向量为止。
(如何判断？可以根据C中的0，移位，移出0，
但不产生新冲突向量)



(5) 找出最佳调度方案

从各个**闭合回路**中找出平均间隔最小的一个。

● 本例中调度方案如下：

(3, 4) 是
(10110111)
和 (10111011)

调度策略	平均间隔拍数	调度策略	平均间隔拍数
(2, 7)	4.50	(3, 4, 3, 7)	4.25
(2, 2, 7)	3.67	(4, 3, 7)	4.67
(3, 4)	3.50	(4, 7)	5.50
(3, 7)	5.00	(7)	7.00
(3, 4, 7)	4.67		

最小平均等待时间

- 最小平均等待时间（minimal average latency---MAL）
- **最佳调度方案为（3，4），MAL=3.5**。对非 C_0 开始的调度方案由**流水线控制器**完成控制的过渡。
- 简单循环：每种状态只出现一次的等待时间循环
- 简单循环中一些是**迫切循环**。
- **迫切循环**：从各自的初始状态输出的边缘都具有最小等待时间的循环。这样的循环首先是简单循环，并且他们的平均等待时间必须比其他简单循环的更小。
- 从迫切循环中确定MAL
- **MAL应小于或等于状态图中任一迫切循环的平均等待时间**
- **上限是初始冲突向量的1个数再+1**

应用举例

某单功能流水线预约表如下：

请确定最佳调度方案。

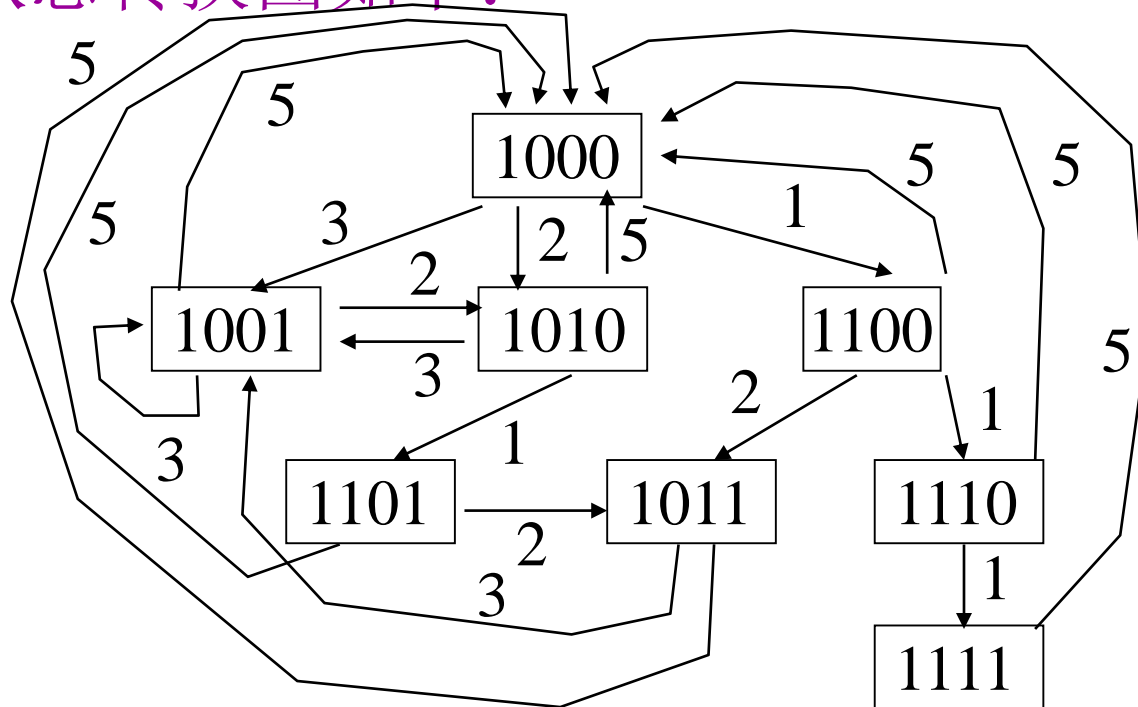
按此方案输入8个指令时，性能指标如何？

	t1	t2	t3	t4	t5	t6
S1	×				×	
S2		×				×
S3			×			
S4				×		

解:

- $F=\{4\}$; $C_0=(1000)$;

- 状态转换图如下:



不要忘记流水线排空后进入（5拍）。

4+1=5 (1111) 相当于 (01111) 第5位

调度方案如下:

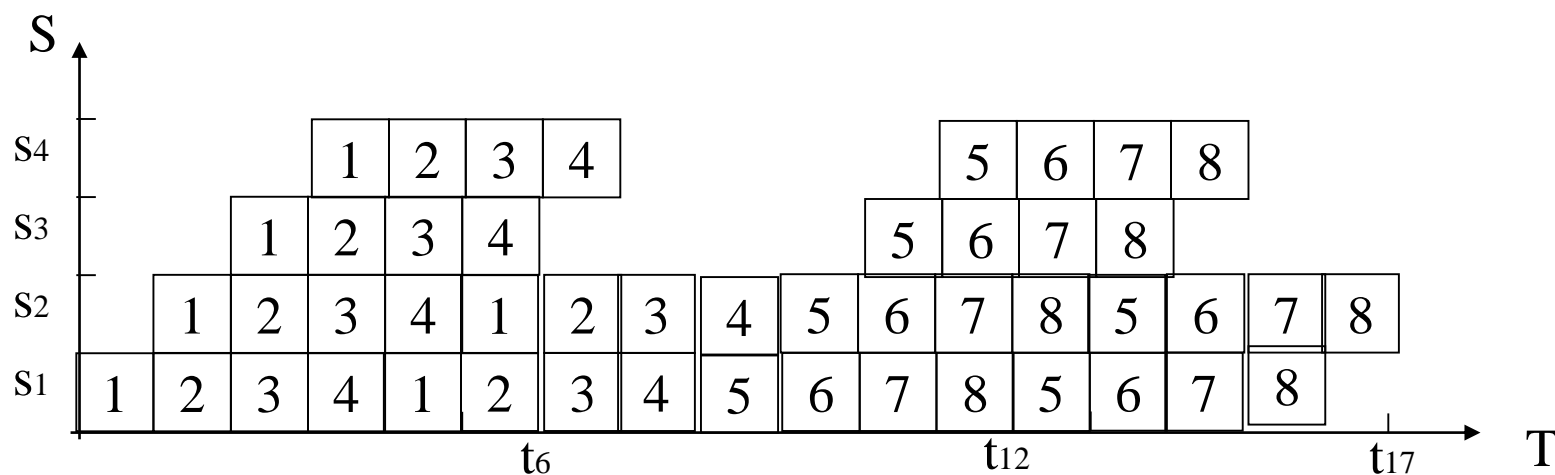
1,5	6/2	2,1,2,5	10/4
1,1,5	7/3	2,3,5	10/3
1,1,1,5	8/4	3,5	8/2
1,2,5	8/3	3	3
1,2,3,5	11/4	3,2,5	10/3
2,5	7/3	3,2,1,5	11/4
2,1,5	8/3	2,3	5/2

最佳调度方案: 1, 1, 1, 5

- 平均最少延时: $MAL = 2$ 拍

时空图表示

注意：每种调度方案均要枚举到；
不要忘记流水线排空后进入（5拍）。
8个指令进入流水线的时空图如下：



性能指标计算

吞吐率 每个时钟周期启动任务的平均数
或：指令数/运行时间

$$P = 8 / (17 \Delta t);$$

加速比

$$S = (6 \Delta t \times 8) / (17 \Delta t) = 48/17;$$

效率

$$E = (6 \Delta t \times 8) / (17 \Delta t \times 4) = 12/17;$$

假如：n条指令，效率如何？

n个指令进入流水线的时空图如下：

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	...
s4				1	2	3	4					5	6	7	8					9	10	11	12					13	14	15	...
s3			1	2	3	4					5	6	7	8					9	10	11	12					13	14	15	...	
s2		1	2	3	4	1	2	3	4	5	6	7	8	5	6	7	8	9	10	11	12	9	10	11	12	13	14	15	...		
s1	1	2	3	4	1	2	3	4	5	6	7	8	5	6	7	8	9	10	11	12	9	10	11	12	13	14	15	...			

吞吐率 $P = 1/MAL = 1/(2 \Delta t)$

效率 $E = (24 \Delta t) / (8 \Delta t \times 4) = 3/4;$

三 向量流水处理

提高标量流水线性性能方法：

增加流水线段数，以减少 Δt ；

每个时钟同时启动多条指令；

减少相关，减少功能变换次数，增加处理

向量操作特点：

向量元素间操作相互独立，无数据相关；

向量元素间操作相同，无功能变换；

相当于标量循环，对指令访问带宽的要求不高。

向量操作很适合于流水处理或并行处理。

以向量操作 $D=A \times (B+C)$ 为例。

1、横向处理方式

循环处理 $d_i = a_i \times (b_i + c_i)$ 。

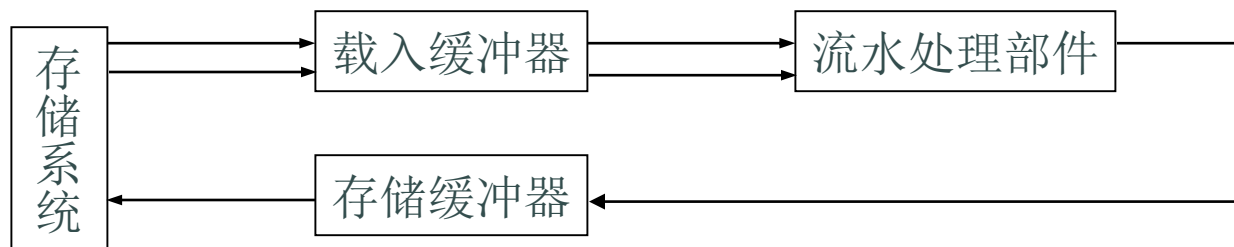
特点：产生 N 次相关、 $2N$ 次功能变换；不宜用于向量处理。

2、纵向处理方式

循环处理 $e_i = b_i + c_i$ ，再循环处理 $d_i = a_i \times e_i$ 。

特点：产生1次相关、1次功能变换，适合用于向量处理。

思考：针对不同的 N ，硬件可采用哪些结构实现？

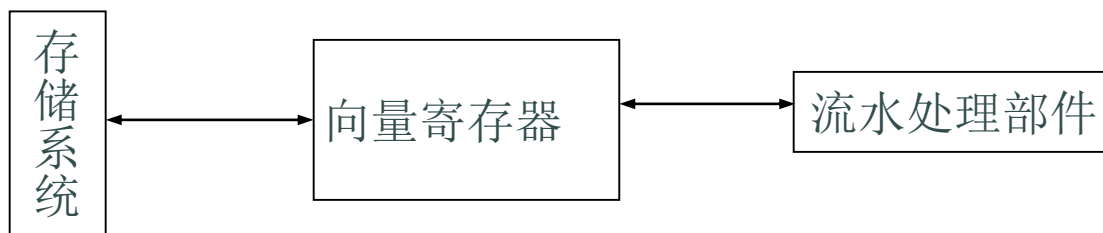


3、纵横处理方式

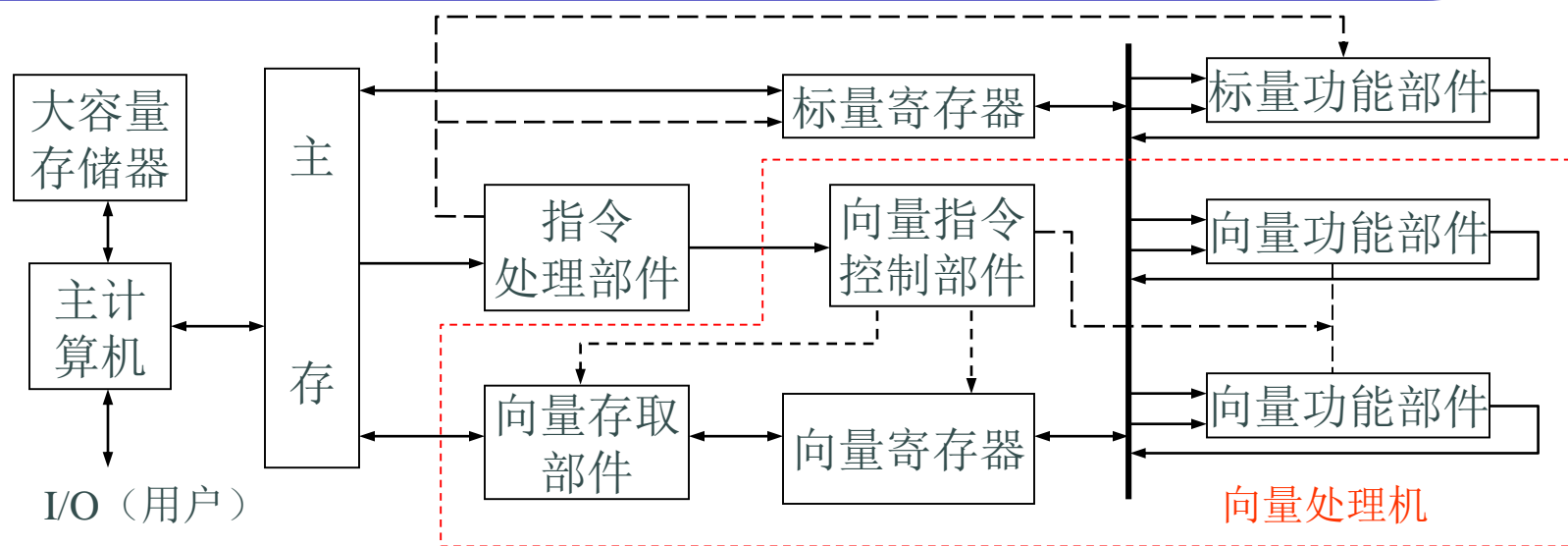
对向量分组(n 个元素), 组内纵向处理、组间横向处理。

特点:

很适合于**REG-REG**型结构的处理方式。



4. R-R向量处理机结构



指令处理部件：标量指令直接控制，向量指令传递控制；

向量寄存器: V_i

向量存取部件：设置多级缓冲器，缓冲主存与向量寄存器速度差，**M-M型**向量处理机简单得多(只需缓冲几个分量)。

2、存储系统设计

(1) 提高存储系统带宽

对MEM的带宽需求相对M-M型向量处理机要低得多；
一般采用单总线即可。

(2) 提高向量存取性能

- ①向量存储从无序到错位存储，提高向量存取性能；
- ②设置指令、数据缓冲器，减少存取频率；
- ③设置地址缓冲器，实现成组交换，提高存取性能。

(3) 解决向量访问冲突

采用错位存储，从空间上解决访问的存储体冲突；
增加可变延迟器，从时间上解决访问的存储体冲突；
设置向量存取指令，指令串行执行可解决访问的操作冲突。

V₇ 向量寄存器组(8×64个)



地址寄存器A($8 \times 24\text{bit}$), 中间寄存器B ($64 \times 24\text{bit}$), 中间寄存器T(64个64bit)。

第4章 流水线结构

4.5 超级流水处理机

- 超标量处理机与超流水线处理机
- 指令调度
- 多流水线调度

超标量处理机概念

- 一般流水线处理机：

一条指令流水线，一个多功能操作部件，每个时钟周期平均执行指令的条数小于1。

- 多操作部件处理机：

一条指令流水线，多个独立的操作部件，操作部件可以采用流水线，也可以不流水。多操作部件处理机的指令级并行度小于1。

- 超标量处理机典型结构：

多条指令流水线、多个功能部件。先进的超标量处理机有：定点处理部件CPU，浮点处理部件FPU，图形加速部件 GPU，大量的通用寄存器，两个一级高速Cache

- 超标量处理机的指令级并行度大于1

概念 指令级流水线并行

指令级的并行度：每个时钟同时启动的指令或操作的数量。

表示： $ILP(m, n)$, Instruction Level Parallel
 m —同时启动的指令或操作，
 n —每个时钟启动指令或操作的次数。

特征： $ILP(m, n) \times CPI = 1$ 。

指令级并行处理机

超标量处理机 超流水线处理机 VLIW

超标量超流水线处理机

流水线结构

- **单发射结构**流水线

做到每个周期能平均执行一条指令，即 $IPC=1$ 。
但由于转移相关与数据相关，还有其他的资源冲突，使得 IPC 远小于1。

- 通过指令重组，可以提高 IPC 并使之接近于1，
但单发射的RISC的 IPC 不可能大于等于1

- 流水线执行和相关性问题 “**气泡**”， “**冲刷**”

流水线受阻 第 $n+1$ 指令与第 $n+2$ 条指令数据相关

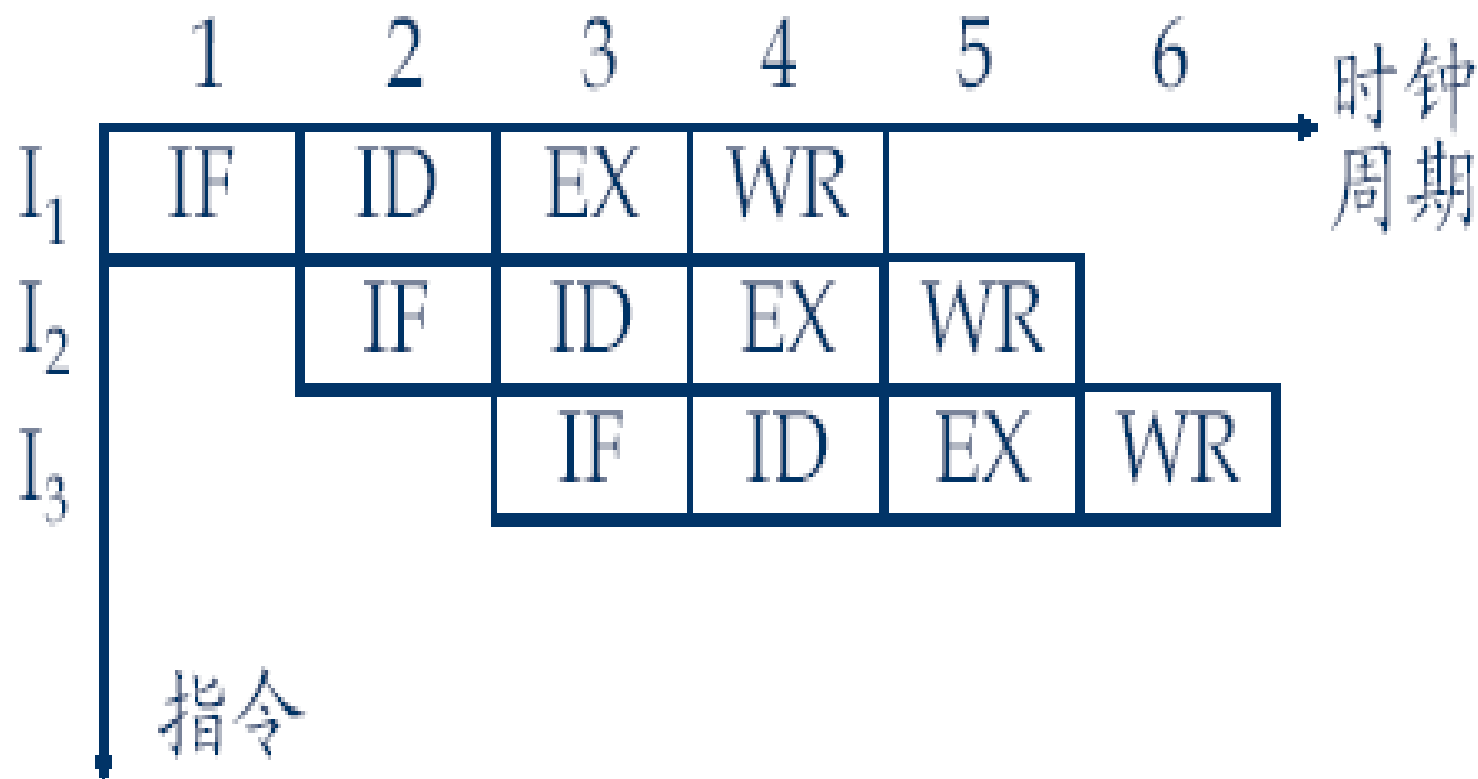
四种流水线的主要特性

机器类型	k段流水线 基准标量 处理机	m度 超标量 处理机	n度 超流水线 处理机	(m,n) 度 超标量 超流水线 处理机
机器流水线 周期	1	1	1/n	1/n
同时发射 指令条数	1	m	1	m
指令发射 等待时间	1	1	1/n	1/n
指令级 并行度ILP	1	m	n	$m*n$

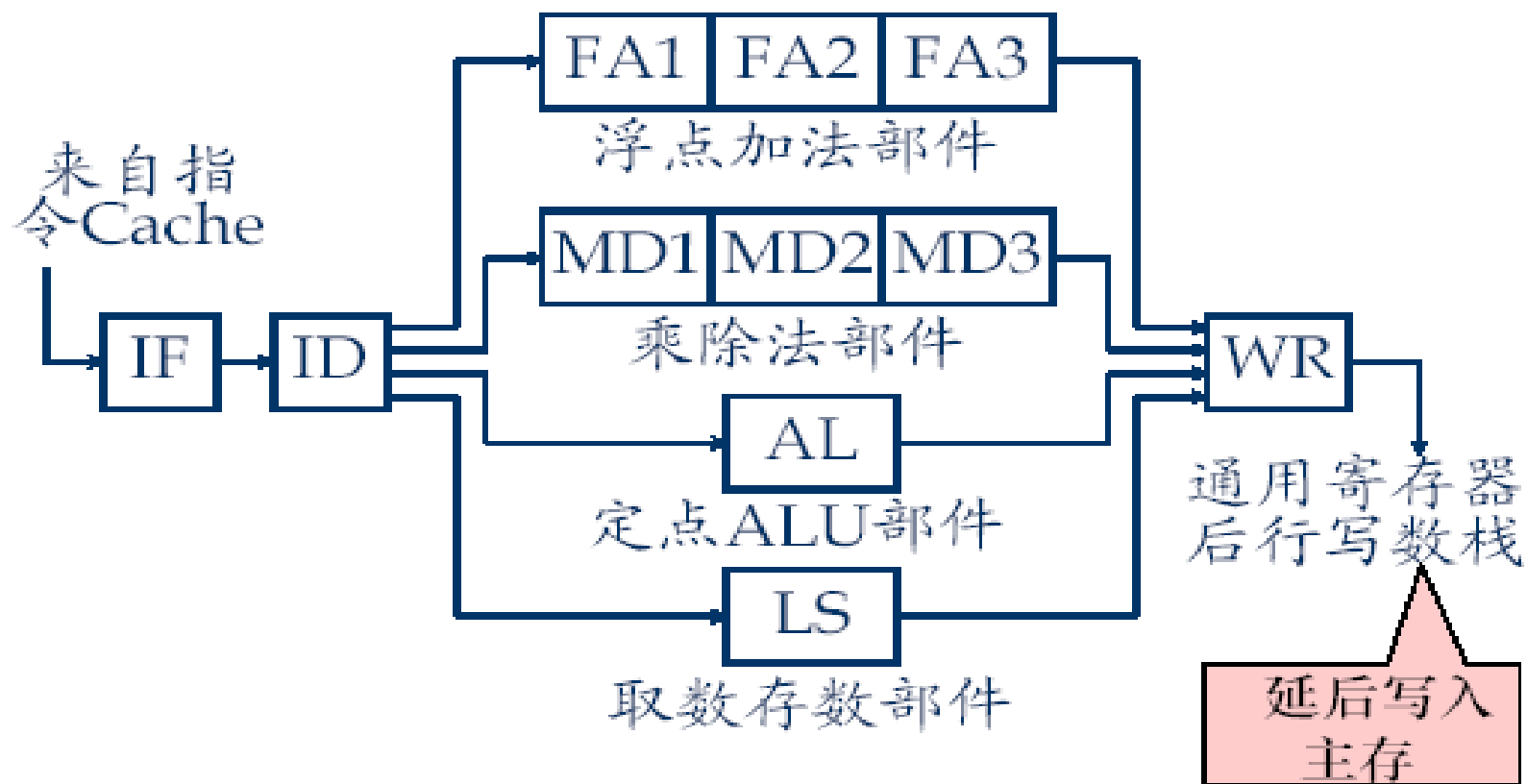
- **多发射结构**

- 为了使 $IPC > 1$ ，自然会联想到能否在一个周期内发出多条指令。
- 多发射结构 多发射结构处理器常见的有超标量、超流水线和VLIW结构。
- 数据流结构也属于多发射结构
- 超标量、超流水线和VLIW的正常流水线结构分别见图（教材）。

单发射指令处理器流水线时空图



单发射指令流水线

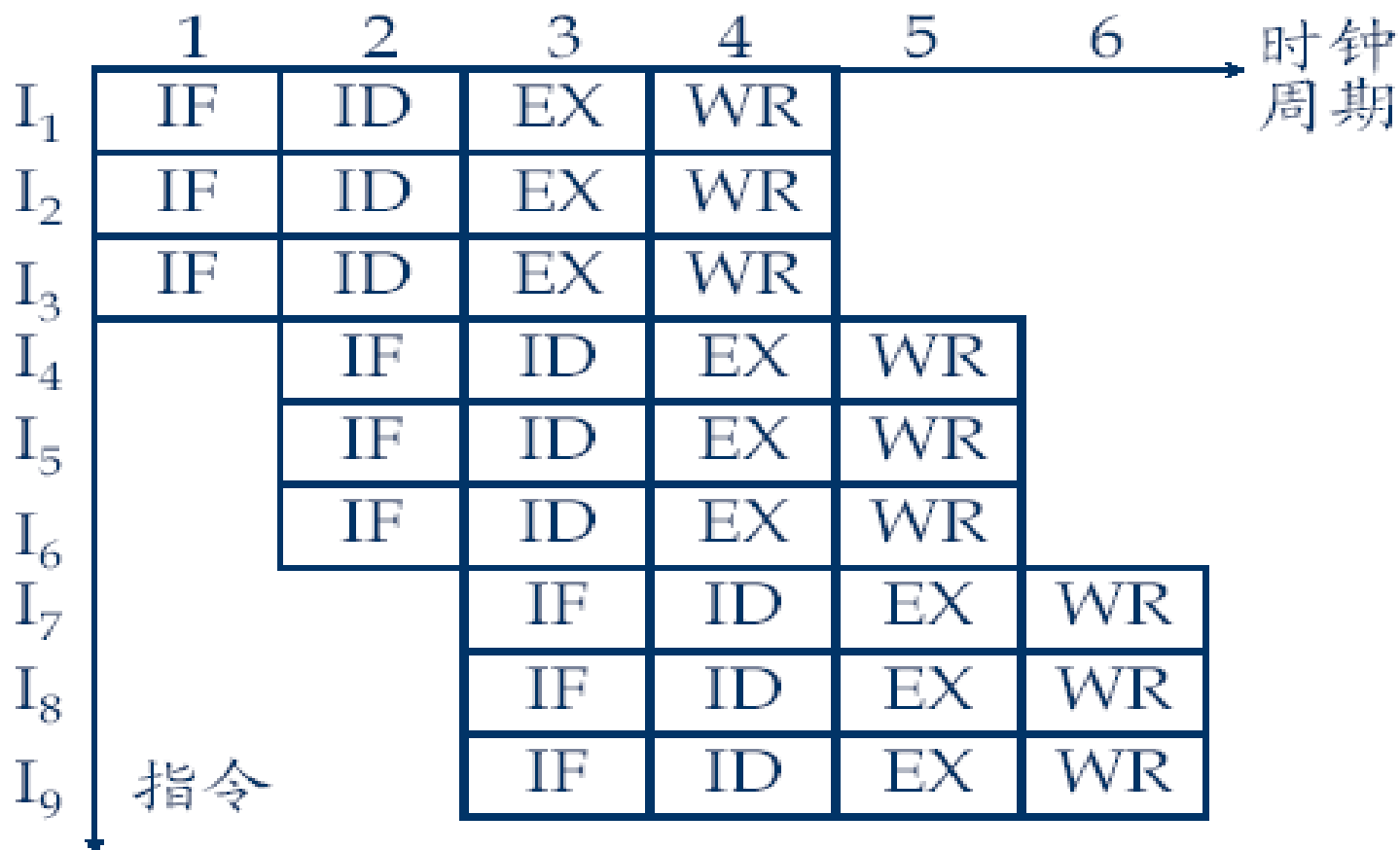


由4个操作部件组成的单发射处理器

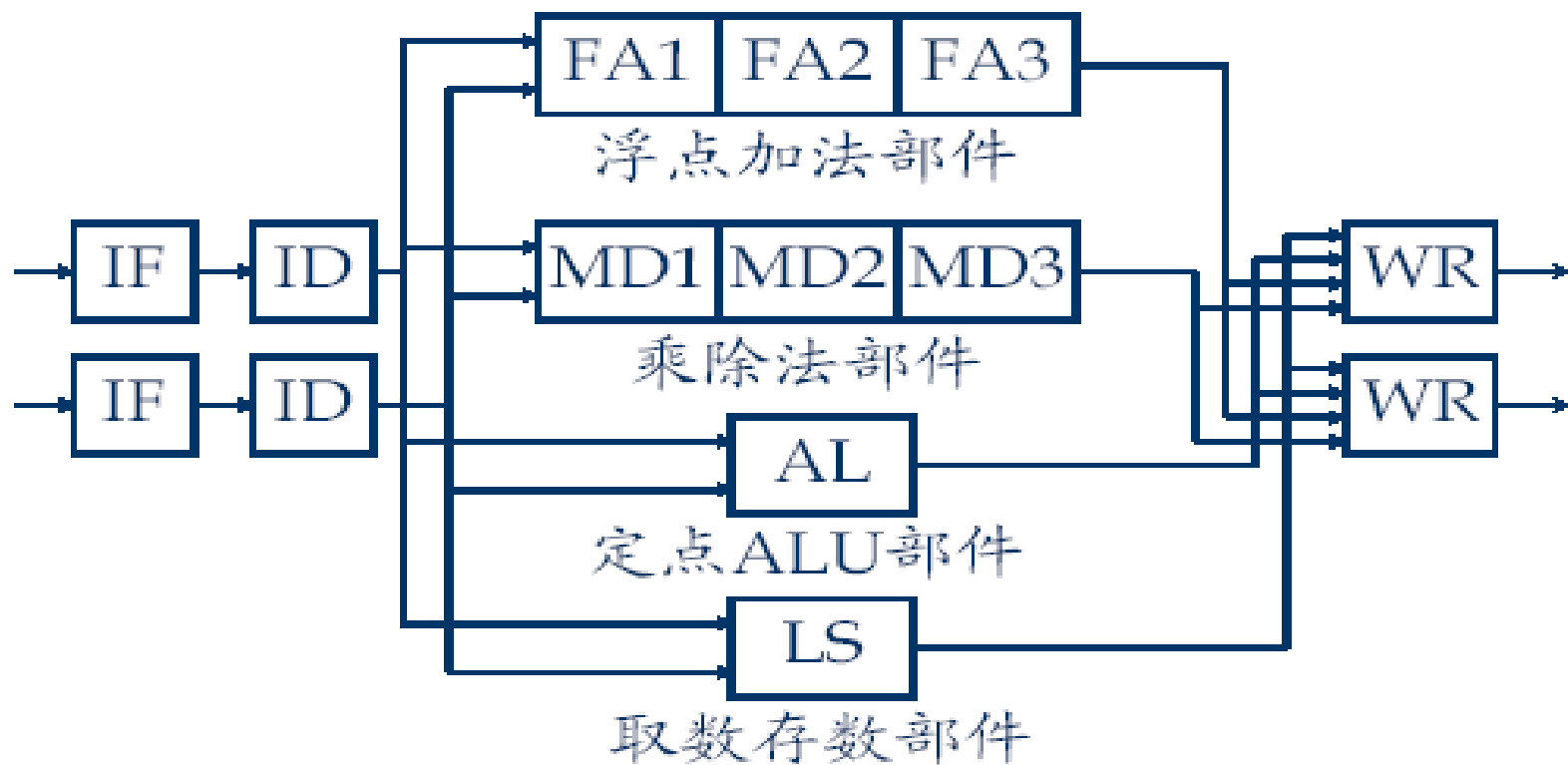
多发射处理机

- 每个周期同时取多条指令、同时译码多条指令，同时执行多条指令，同时写回多个运算结果
- 需要多个取指令部件，多个指令译码部件和多个写结果部件
- 设置多个指令执行部件，复杂的指令执行部件一般采用流水线结构
- 设计目标是每个时钟周期平均执行多条指令，**ILP**的期望值大于**1**

多发射时空图



多发射指令流水线

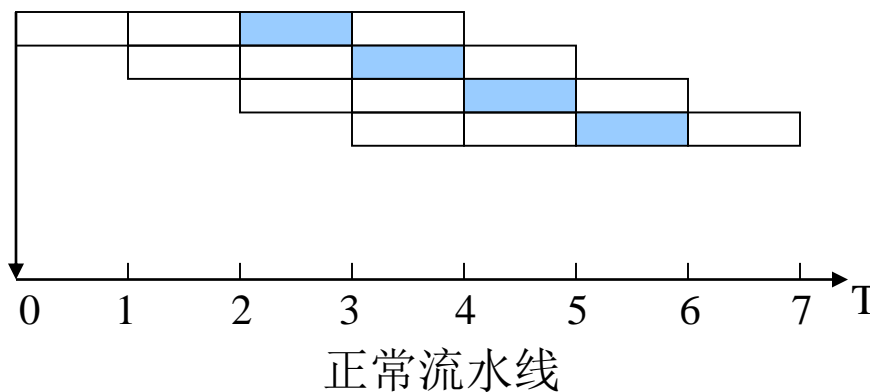


1. 超标量结构

(借助于资源重复)

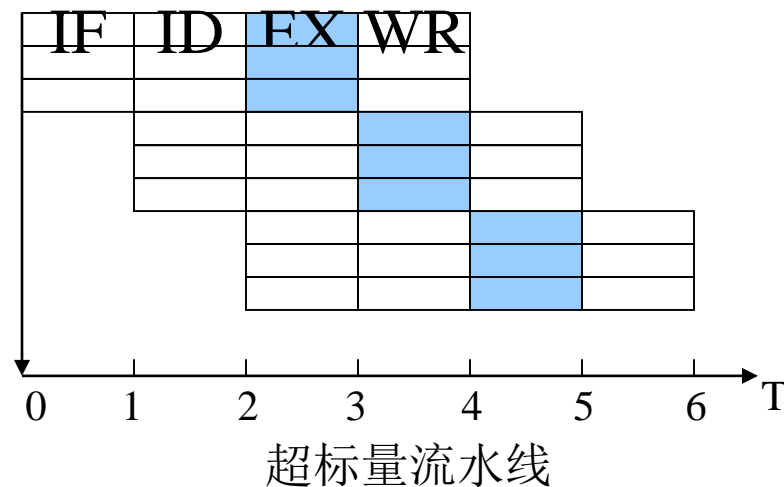
思想：多条流水线并行工作，提高指令级并行度。

超标量工作原理：



RISC: ILP (1, 1)

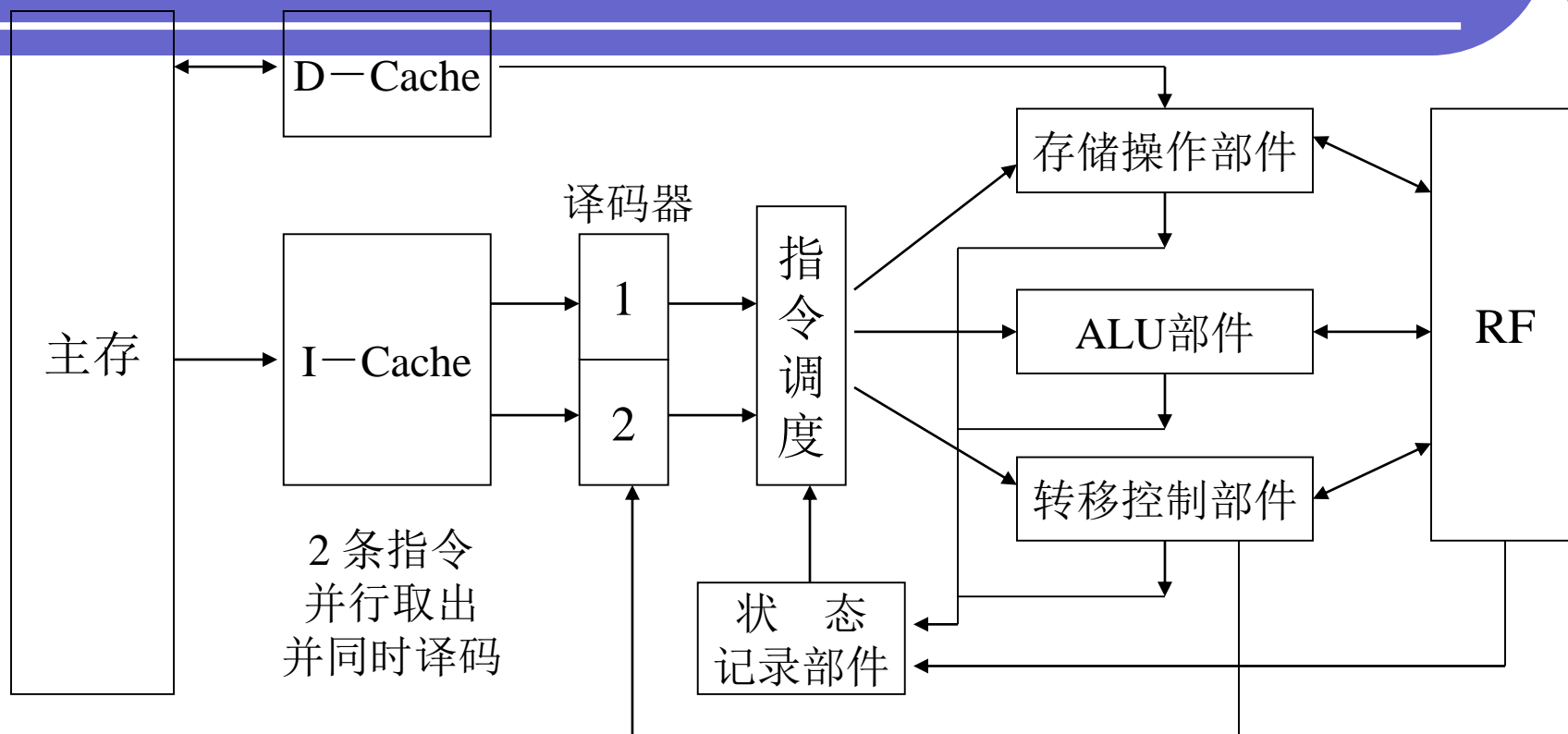
硬件要求：



超标量: ILP(3, 1)

多套独立译码电路、多条独立流水线、控制器

超标量机结构一——集中式调度：



超标量流水性能：指令并行度 ≤ 2 。

超标量机结构——分布式调度：

PII结构

超标量流水线性能：

普通标量单流水性能：

（假设指令间没有相关）

$$T(1,1) = (k + N - 1)\Delta t$$

超标量流水线性能：

$$T(m,1) = (k + \frac{N-m}{m})\Delta t$$

相对单流水加速比：

$$S(m,1) = \frac{T(1,1)}{T(m,1)} = \frac{m(k+N-1)}{N+m(k-1)}$$

当 $N \rightarrow \infty$ 时， $S(m, 1) = m$ 。

影响超标量流水线性能因素：

多条指令同时存取，数据通道可能会成为瓶颈；

多条指令间不能有相关，否则部分流水线空转；

相关专用通道只能针对多条指令后的指令进行。

指令调度策略（发射策略和流水线调度策略）：

发射：控制器判别多条指令间是否存在相关，不相关可并行执行，相关时重新调度指令或不相关指令进入流水线；

分类：

顺序发射顺序完成—静态调度策略；（Pentium）

顺序发射乱序完成—动态调度策略；（PII）

乱序发射乱序完成—动态调度策略。

实现：

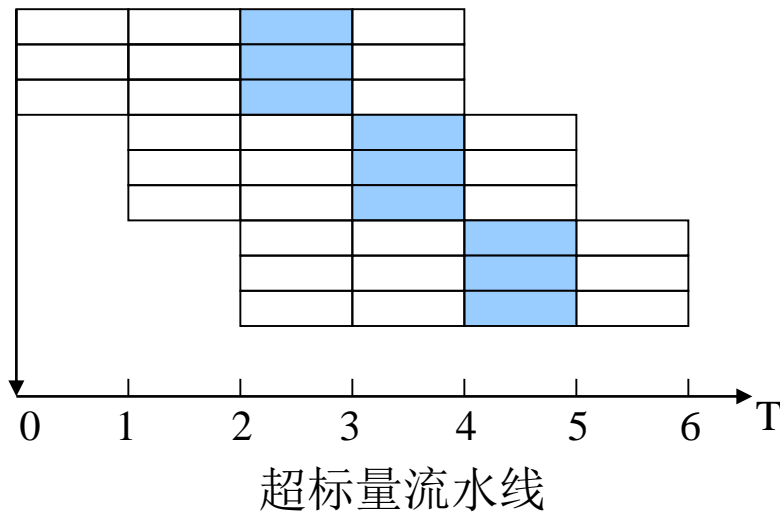
发射策略由译码控制器完成，

流水线调度由执行控制器完成。

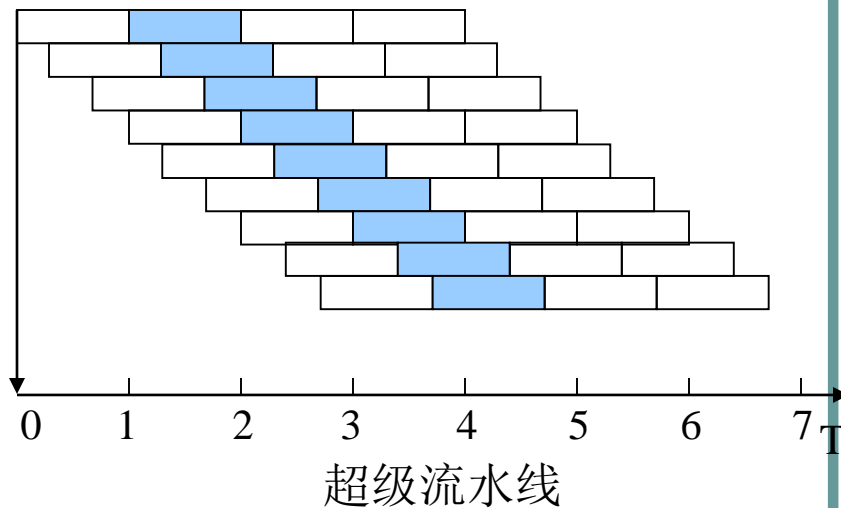
2. 超级流水方法

思想：单条流水线的启动速度**小于1拍**，来提高指令级流水速度。

工作原理：



超标量：ILP(3, 1)



超流水：ILP(1, 3)

硬件要求:

采取多种手段特别处理好指令相关，否则一旦相关，产生的延迟会比普通流水线大。

超标量超流水线:

同时采用超标量技术和超级流水技术的流水线。

超标量超级流水线性能:

$$T(m, n) = \left(\frac{k}{n} + \frac{N-m}{mn} \right) \Delta t$$

相对单流水线加速比:

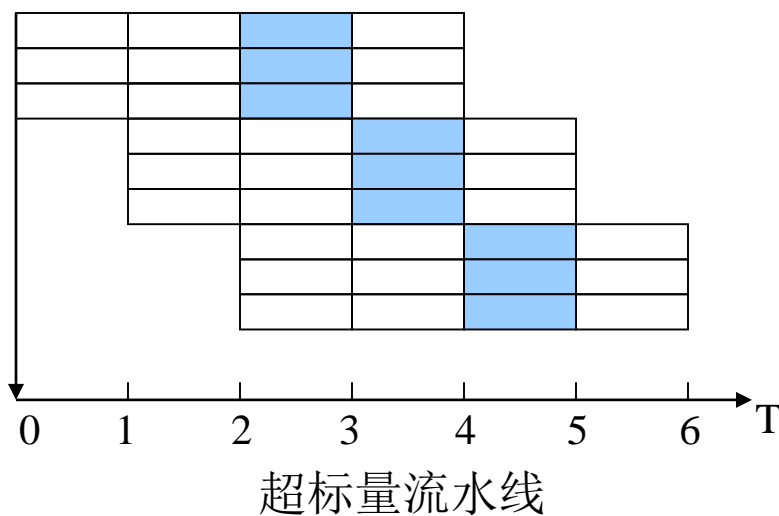
$$S(m, n) = \frac{T(1,1)}{T(m,n)} = \frac{mn(k+N-1)}{mk+N-m}$$

$$\text{当 } N \rightarrow \infty \text{ 时, } S(m, n) = mn。$$

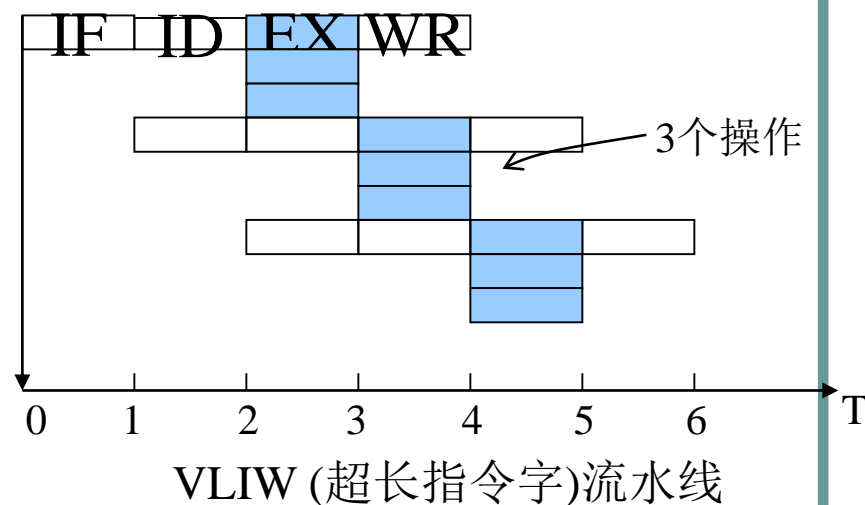
3. 超长指令字 (VLIW) 技术

思想： 设置多个执行部件，同时完成执行 (EX) 功能，提高操作级并行度。

VLIW流水工作原理：

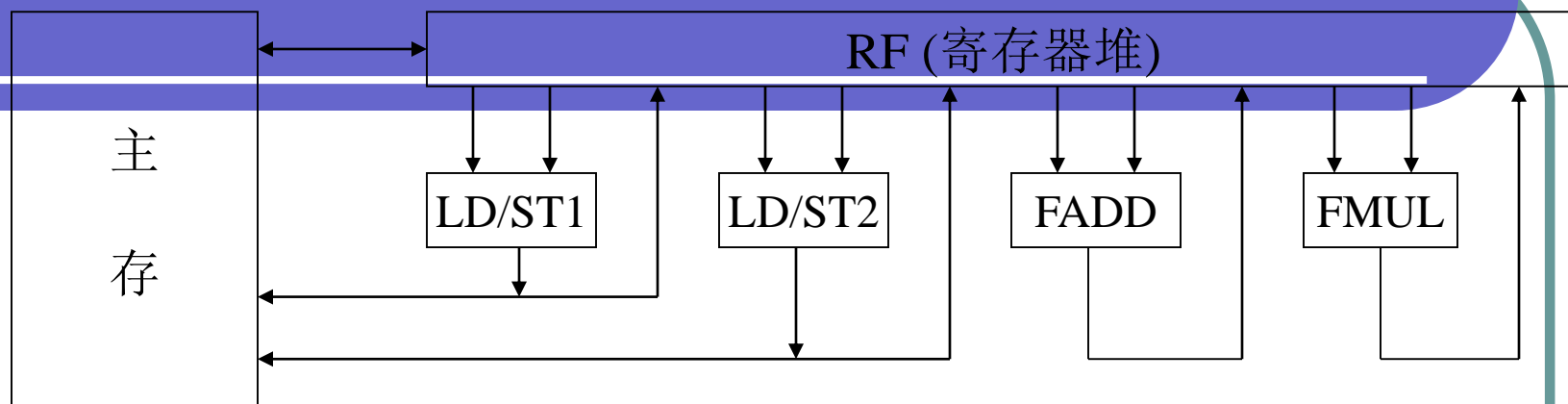


超标量: ILP (3, 1)



VLIW: ILP (3,1)

VLIW结构:



VLIW 中的操作字段:

LD/ST1	LD/ST2	FADD	FMUL
存/取1	存/取2	浮点加	浮点乘

性能: 操作并行度 ≤ 4 。

特点:

控制器为单一的控制流，指令中不同控制字段控制相应的功能部件；

并行调度在编译阶段完成，非执行时完成。

VLIW优化编译技术:

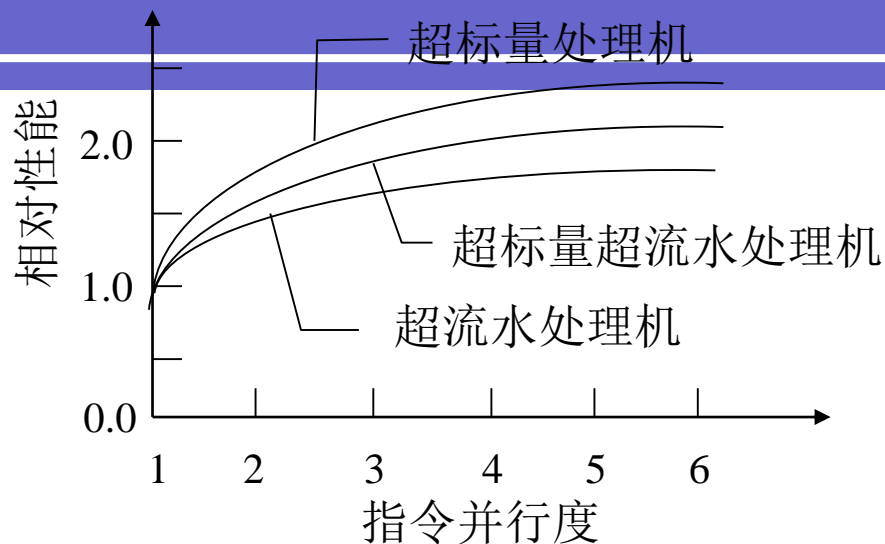
局部调度: 表调度法, 在基本块范围内调度。

全局调度: 路径调度法, 渗透调度法和软件流水法。

VLIW与超标量流水特点比较:

	译码	并行性检测	代码密度	可移植性
VLIW	简单	编译	较差	不可移植
超标量	复杂	执行	较好	可移植

几种指令级流水线性能比较:



原因分析:

- a. 超级流水线的启动延迟、条件转移造成的损失、执行部件的冲突程度均比超标量大;
- b. 指令级并行度较小时, 处理机性能提高较快;
- c. 处理机的性能与程序的调度算法有很大关系。

4 多流水线调度

- **超标量处理机：**

一个时钟周期内能够同时发射多条指令的处理机称为超标量处理机，必须有两条或两条以上能够同时工作的指令流水线。

- **先行指令窗口：**

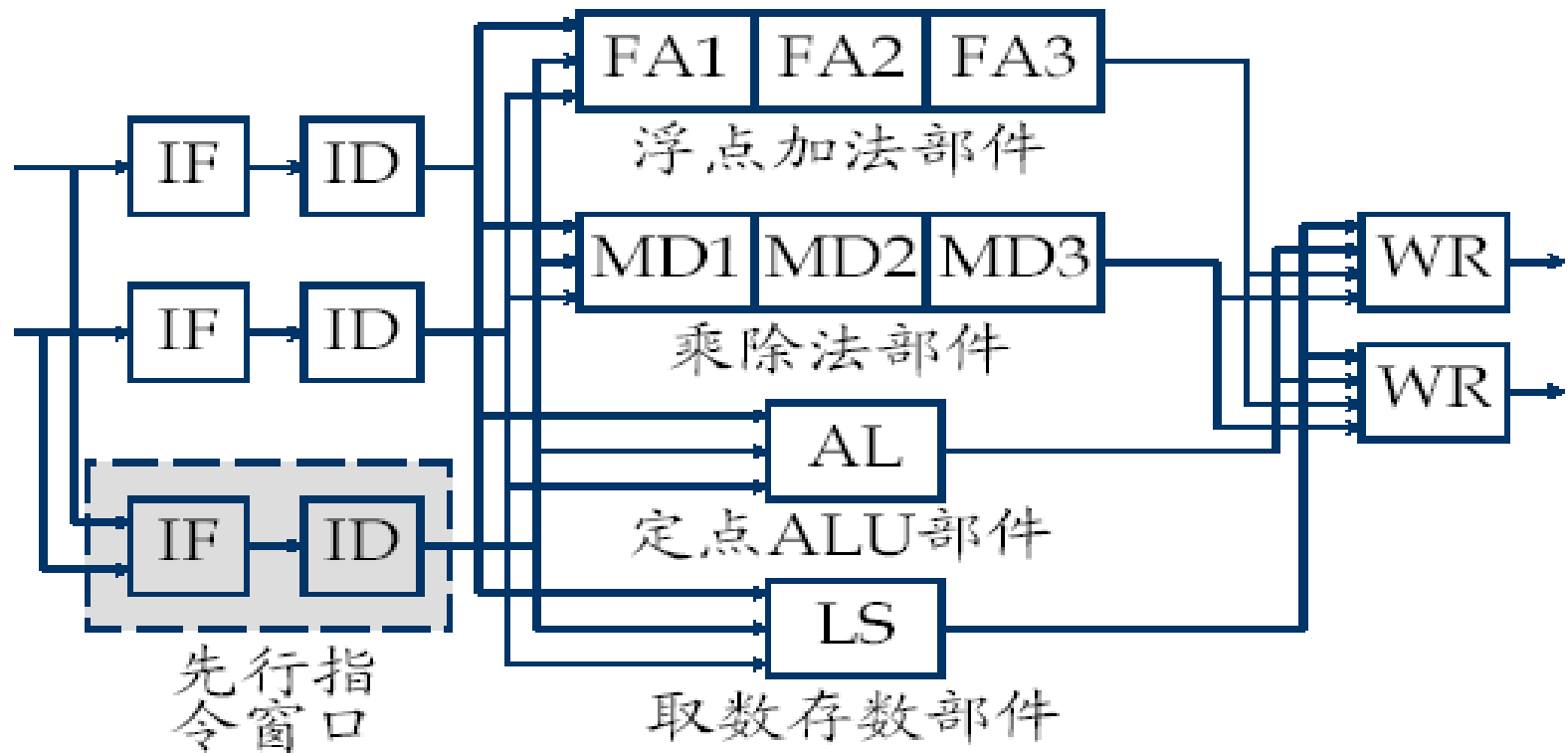
能够从指令Cache中预取多条指令，能够对窗口内的指令进行数据相关性分析和功能部件冲突的检测。窗口的大小：一般为2至8条指令。

- 采用目前的指令调度技术，每个周期发射2至4条指令

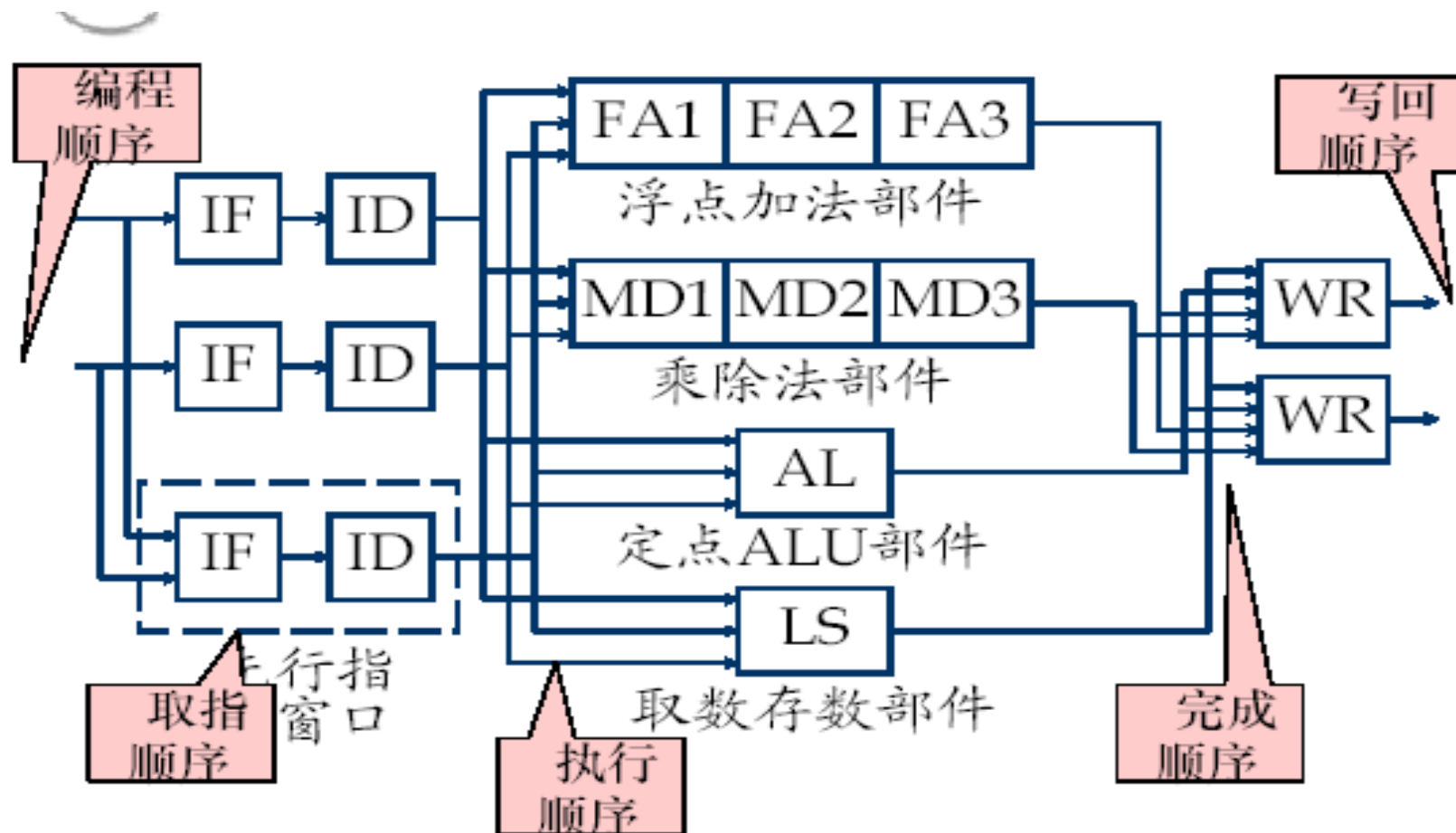
举例

- **Intel**公司的**Pentium**处理机， **Motolora**公司的**MC88110**处理机，
- **IBM**公司的**Power 6000**处理机等每个周期都发射两条指令
- **TI**公司生产的**SuperSPARC**处理机以及**Intel**的 **Pentium III**处理机等每个周期发射三条指令
- 操作部件的个数多于每个周期发射的指令条数。**4**个至 **16**个操作部件
- 超标量处理机的指令级并行度： $1 < \text{ILP} < m$ ；
m为每个周期发射的指令条数。

有先行指令窗口的多发射流水线结构



多流水线如何调度



多流水线如何调度

- **顺序流动** 一串连续任务在流水线中是一个接一个地在各个流水段中间流过的。任务流出流水线的顺序与任务流入流水线的顺序完全相同。
- **乱序流动** 为了充分发挥流水线的效率，在发生数据相关时，要允许没有数据相关的后续指令进入相关指令所占据的流水段，并超越相关的指令继续往前流动。指令流入流水线的顺序和流出流水线的顺序可以是不一样的。

多流水线调度

- 指令排列顺序
- 指令发射顺序
- 指令完成顺序
- 顺序发射：发射顺序=排列顺序，
否则，乱序发射
- 顺序完成：指令完成顺序=排列顺序，
否则，乱序完成
- 多流水线调度：指令在多条流水线上如何执行，
以提高执行效率。

多流水线调度方法

- (1) . 顺序发射顺序完成
- (2) . 顺序发射乱序完成
- (3) . 乱序发射乱序完成
- 程序举例说明

5.5.3 多流水线调度

顺序发射(in-order issue)与乱序发射(out-order issue): **指令发射顺序是按照程序中指令排列顺序进行的称为顺序发射**

顺序完成(in-order completion)与乱序完成(out-order completion): **指令完成顺序是按照程序中指令排列顺序进行的称为顺序完成**

多流水线的调度主要有三种方法:

顺序发射顺序完成

顺序发射乱序完成

乱序发射乱序完成

以如下6条指令组成的程序为例，说明这三种调度方法

I_1 : LOAD R1, A ; $R1 \leftarrow (A)$
 I_2 : FADD R2, R1 ; $R2 \leftarrow (R2) + (R1)$
 I_3 : FMUL R3, R4 ; $R3 \leftarrow (R3) \times (R4)$
 I_4 : FADD R4, R5 ; $R4 \leftarrow (R4) + (R5)$
 I_5 : DEC R6 ; $R6 \leftarrow (R6) - 1$
 I_6 : FMUL R6, R7 ; $R6 \leftarrow (R6) + (R7)$

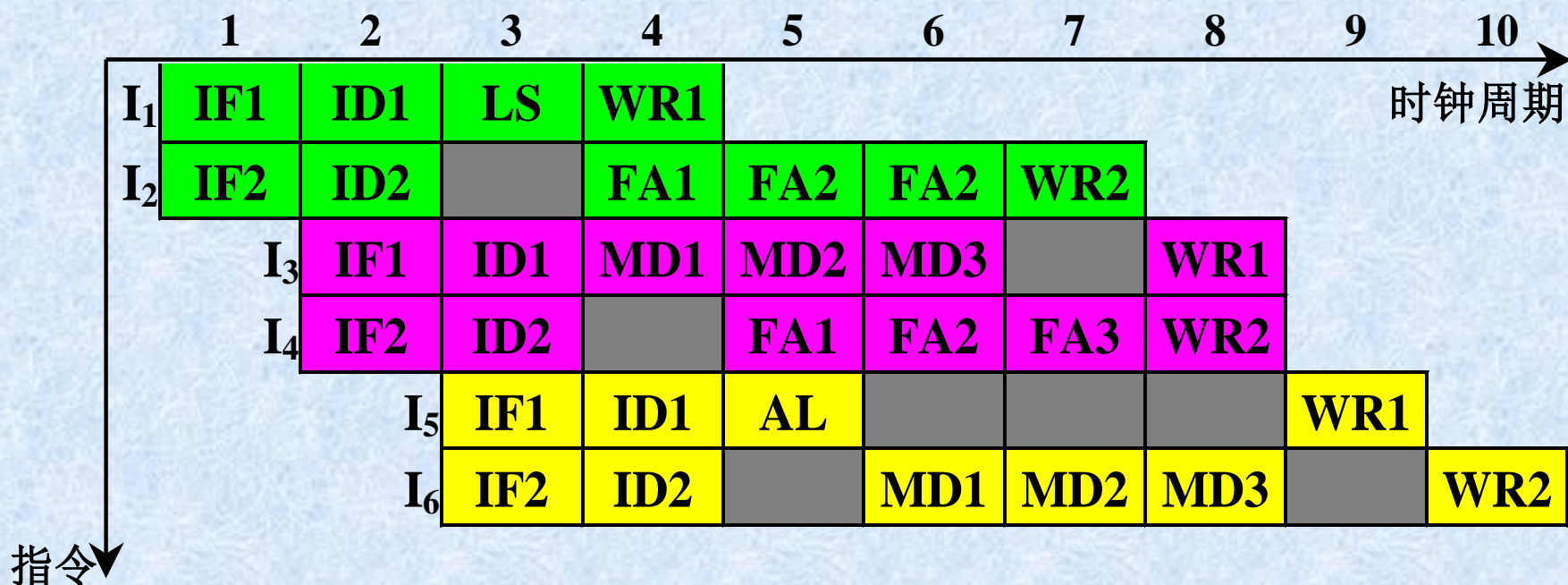
6条指令中有4个数据相关，包括2个写读相关，1个读写相关和1个写写相关。

1. 顺序发射顺序完成

共用10个时钟周期完成

还有8个空闲的时钟周期

顺序发射顺序完成的指令流水线时空图



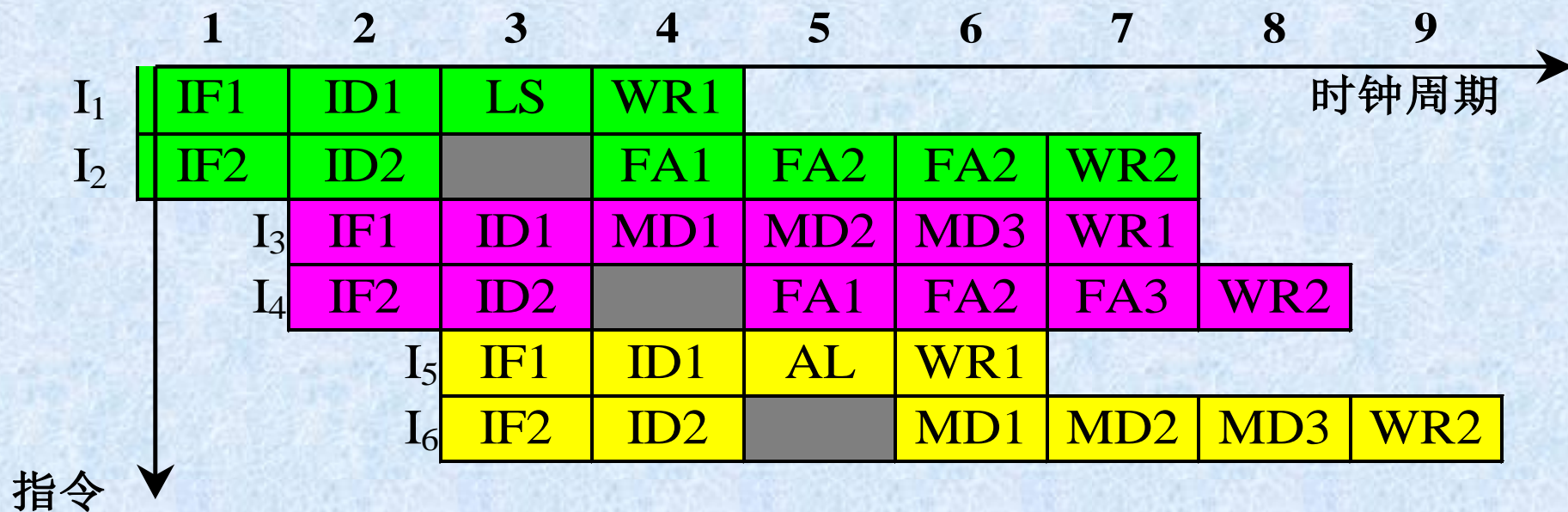
IF: 取指令, ID: 指令译码, LS 取数存数, FA: 浮点加减法运算,
MD: 乘除法运算, AL: 定点算术逻辑运算 WR: 写回运算结果

2. 顺序发射乱序完成

总的执行时间为9个时钟周期，

节省了一个时钟周期。少了5个空闲时钟周期

顺序发射乱序完成的流水线时空图



顺序发射乱序完成的指令完成次序

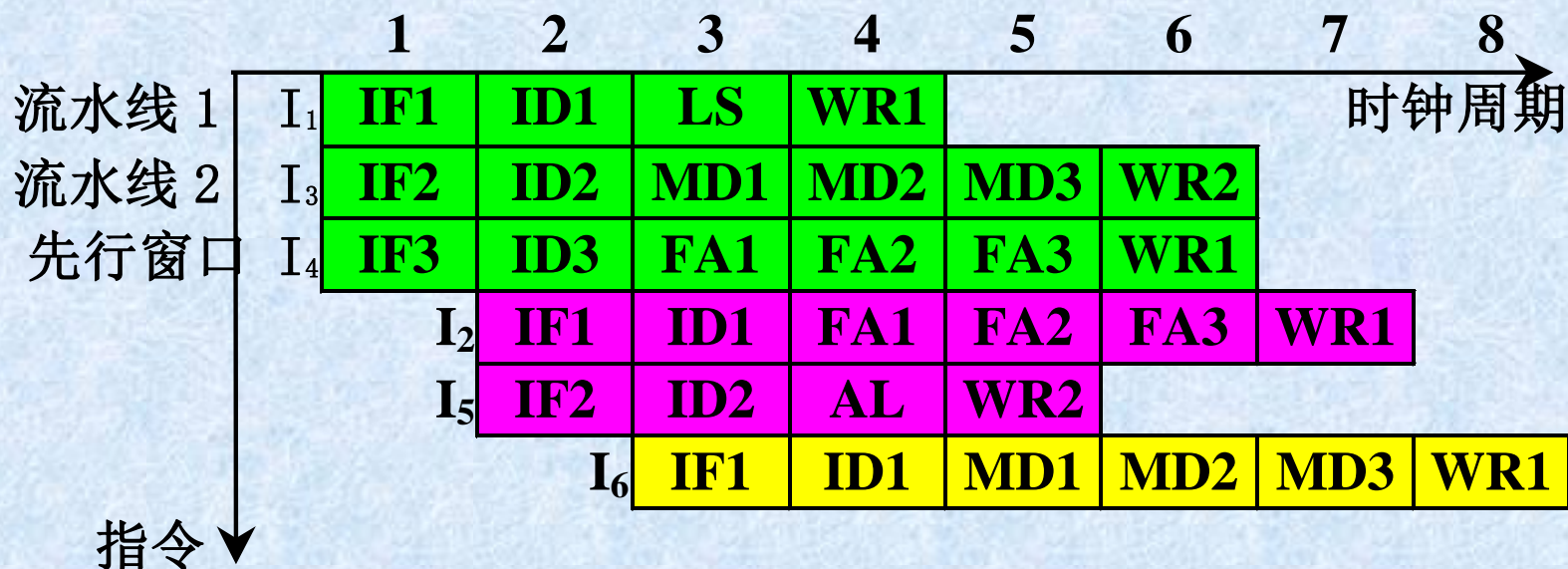
时钟周期	4	5	6	7	8	9
流水线 1	I ₁		I ₅	I ₃		
流水线 2				I ₂	I ₄	I ₆

3. 乱序发射乱序完成

没有空闲周期，功能部件得到充分利用。

总的执行时间为**8**个周期，节省**2**个周期。

乱序发射乱序完成调度方法的流水线时空图



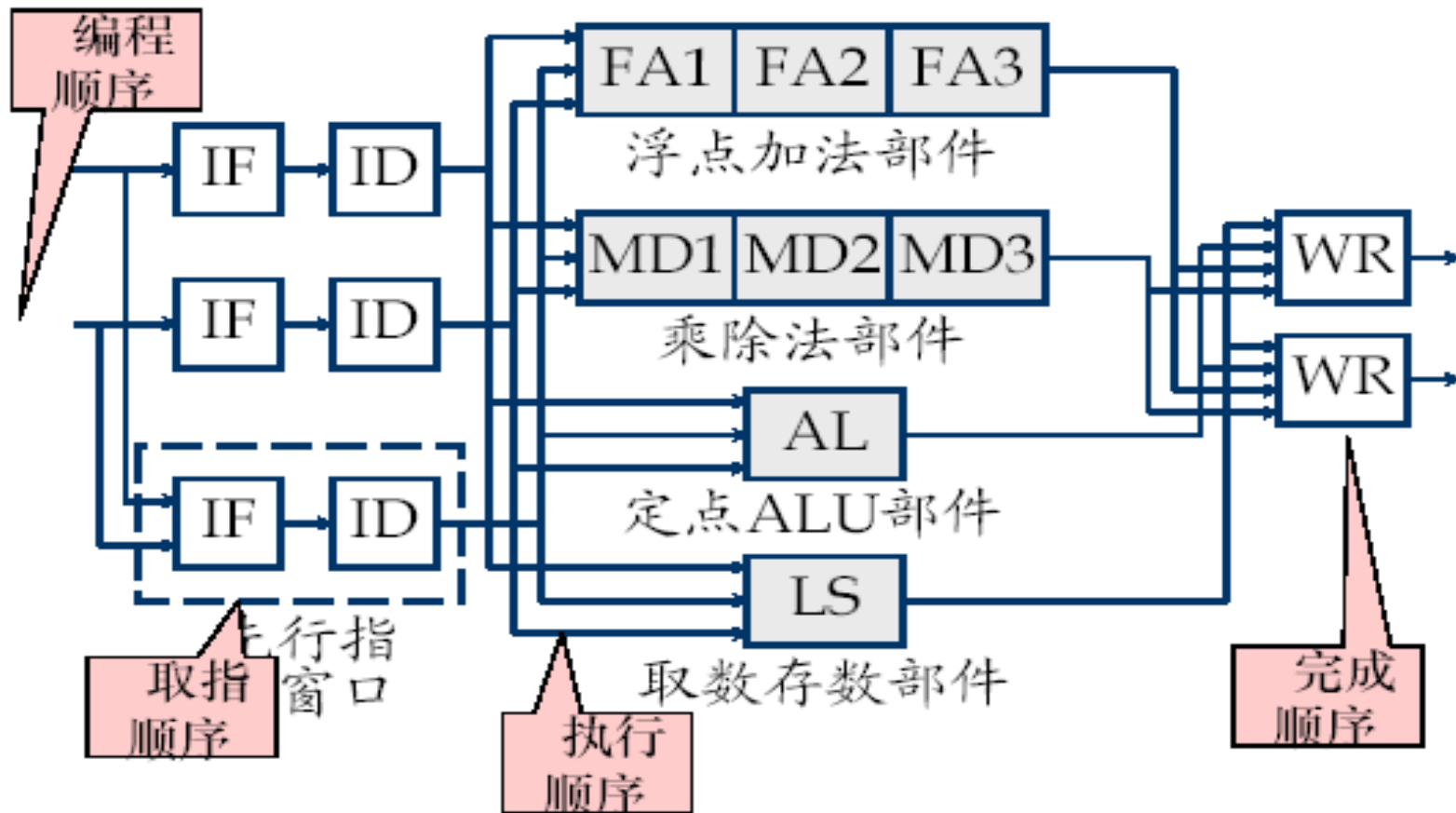
指令在流水线中的发射次序

时钟周	1	2	3
流水线 1	I ₁	I ₂	I ₆
流水线 2	I ₃	I ₅	
先行窗口	I ₄		

指令在流水线中的完成次序

时钟周期	4	5	6	7	8
流水线 1	I ₁		I ₄	I ₂	I ₆
流水线 2		I ₅	I ₃		

5. 资源冲突： 多条指令使用相同的资源



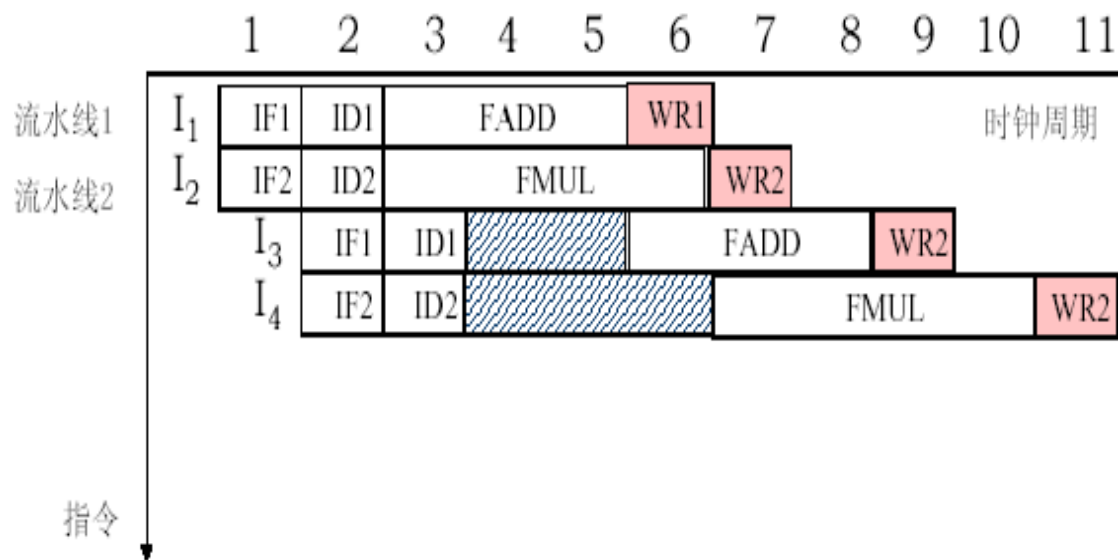
资源冲突：执行部件非流水结构

I1: FADD R0, R1

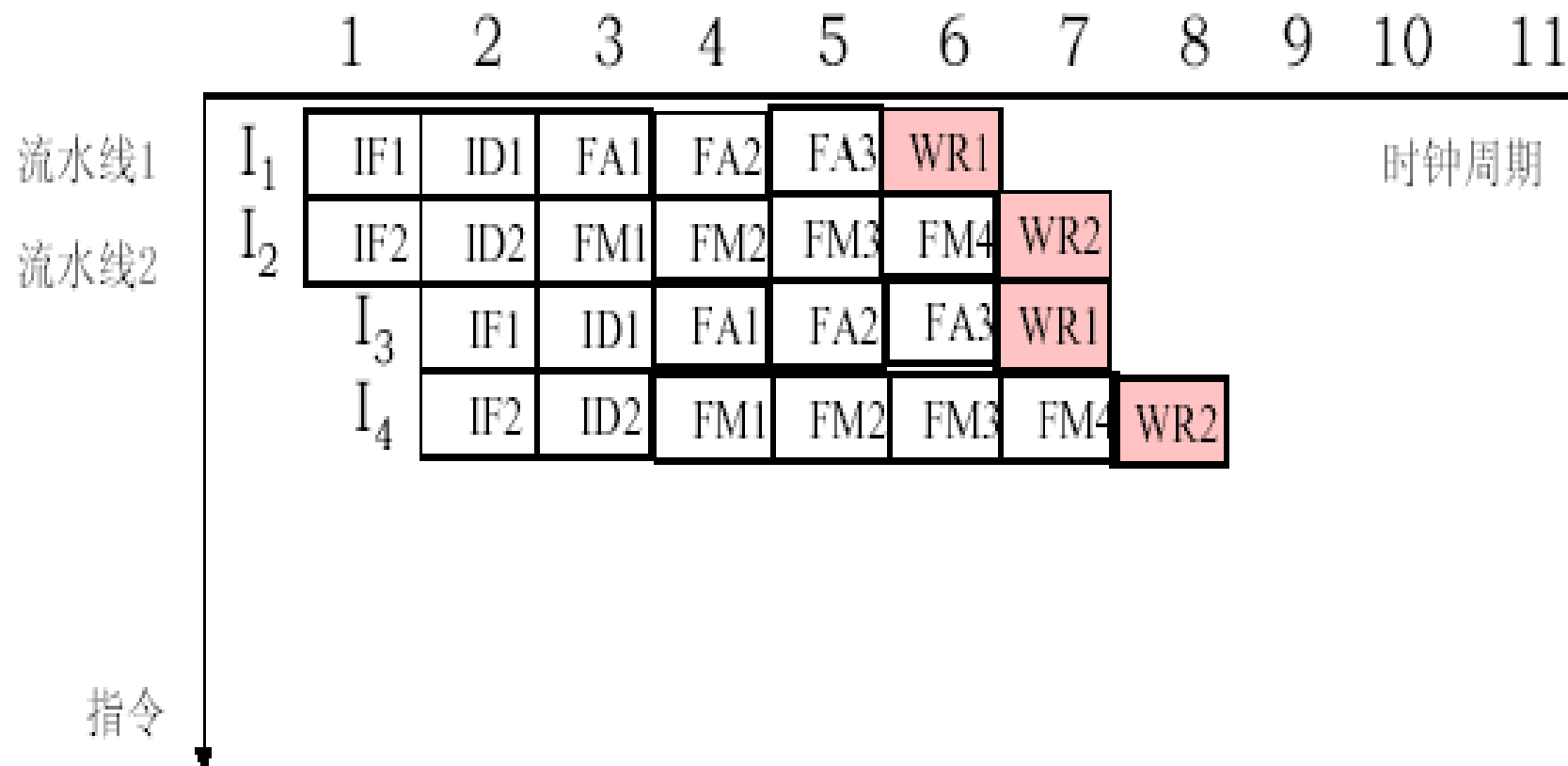
I2: FMUL R2, R3

I3: FADD R4, R5

I4: FMUL R6, R7



资源冲突：执行部件流水结构



小结

- 超级流水处理机

- 超标量处理机与超流水线处理机

- 多流水线调度

- 单发射结构：每个周期只发射一条指令；
 - 多发射结构：在流水线结构基础上，每个周期可以发射多条指令。
 - 比如：超标量 超流水线 VLIW
 - 顺序发射 乱序发射

- P6微结构

- Intel P6 Pentium Pro
 - 扩展指令集 MMX SSE

总结

- **4.1 流水线结构原理**
- **4.2 线性流水线性能指标**
- **4.3 非线性流水线**
- **4.4 流水线相关处理**
- **4.5 超级流水处理机**