

计算机组织与体系结构

第十三讲

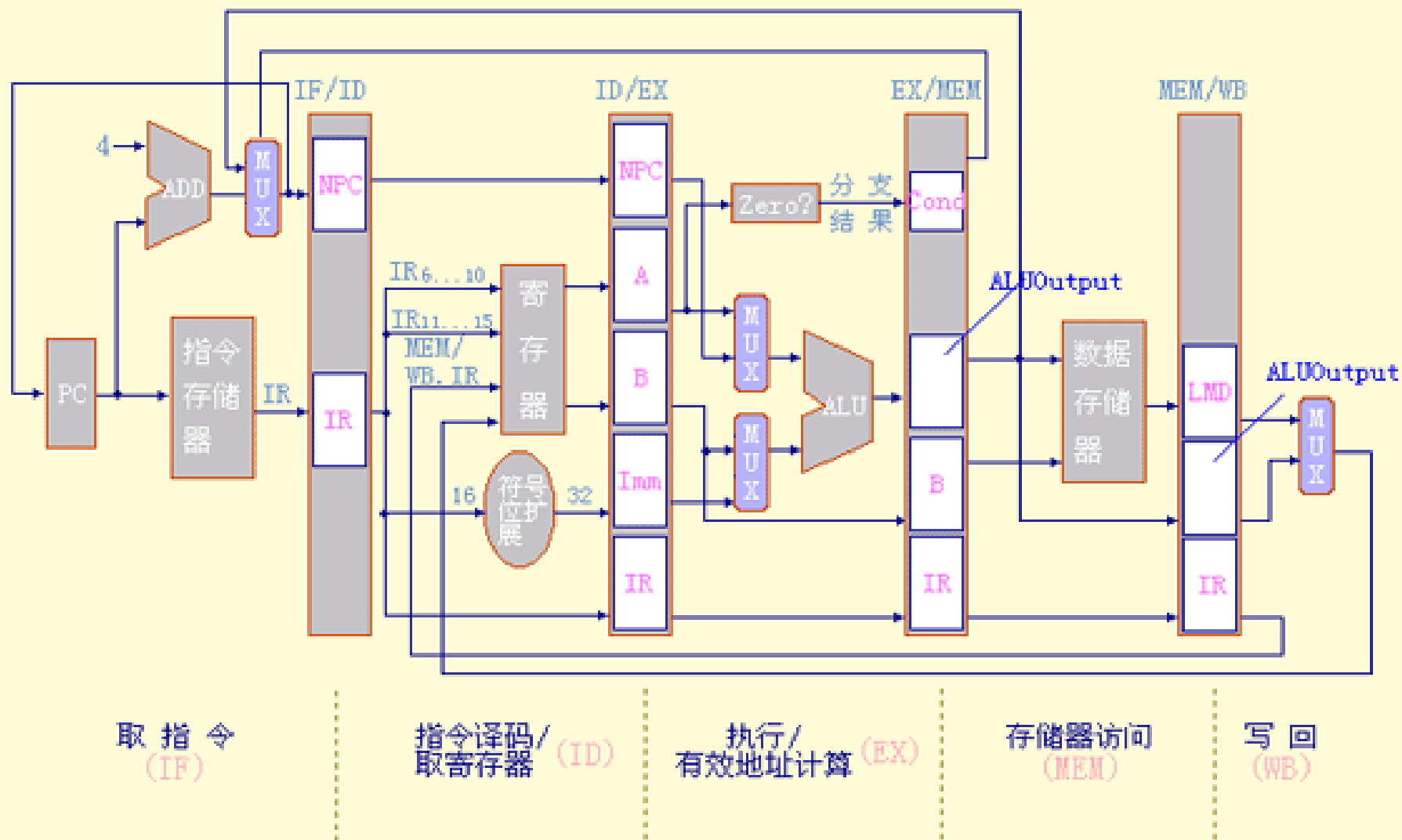
计算机科学与技术学院

舒燕君

Recap

- 流水线的基本概念
 - ✓ 流水技术的定义
 - ✓ 流水线的特点
- 流水线的分类
 - ✓ 实现的功能（单功能、多功能）
 - ✓ 连接方式（静态、动态）
 - ✓ 子过程的粒度（部件级、处理机级和处理机间）
 - ✓ 数据表示（标量、向量）
 - ✓ 反馈回路（线性、非线性）
- **MIPS**基本流水线
 - ✓ 将数据通路划分流水段，加入段间寄存器文件
 - ✓ 每功能段完成指令子过程的相应操作
 - ✓ 多路选择器的控制

DLX 流水线中 对多路寄存器MUX 的控制



6.2 MIPS基本流水线

6.2.1 基本MIPS流水线

6.2.2 流水线性能分析

6.2.2 流水线性能分析

三项性能指标：吞吐率、加速比和效率

1. 吞吐率

是衡量流水线速度的重要指标

- ◆ **吞吐率**是指单位时间内流水线所完成的任务数或输出结果的数量。
- ◆ **最大吞吐率** TP_{max} 是指流水线在达到稳定状态后所得到的吞吐率。
- ◆ 设流水线由 m 段组成，完成 n 个任务的吞吐率称为**实际吞吐率**，记作 TP 。

(1) 最大吞吐率

- ◆ 假设流水线各段的时间相等，均为 Δt_0 ，则：

$$TP_{max} = 1/\Delta t_0$$

- ◆ 假设流水线各段时间不等，第*i*段时间为 Δt_i ，则：

$$TP_{max} = 1/\max\{\Delta t_i\}$$

- 最大吞吐率取决于流水线中最慢一段所需的时间，该段成为流水线的瓶颈
- 消除瓶颈的方法
 - 细分瓶颈段
 - 重复设置瓶颈段

(2) 实际吞吐率

- ◆ 若各段时间相等（假设均为 Δt_0 ），则完成时间

$$T_{\text{流水}} = m\Delta t_0 + (n-1)\Delta t_0 \quad (\text{说明})$$

实际吞吐率

$$TP = \frac{n}{T_{\text{流水}}} = \frac{n}{m \cdot \Delta t_0 + (n-1) \cdot \Delta t_0}$$

$$= \frac{1}{(1 + \frac{m-1}{n}) \Delta t_0} = \frac{TP_{\max}}{1 + \frac{m-1}{n}}$$

时空图

- ◆ 若各段时间不等（假设第*i*段为 Δt_i ），则完成时间

时空图

$$T = \sum_{i=1}^m \Delta t_i + (n-1)\Delta t_j$$

这里， $\Delta t_j = \max\{\Delta t_i\}$

实际吞吐率

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

2. 加速比

- ◆ 加速比是指流水线速度与等功能的非流水线速度之比。
- ◆ 根据定义可知，加速比 $S = T_{\text{非流水}} / T_{\text{流水}}$
- ◆ 若流水线为 m 段，每段时间均为 Δt_0 ，则

$$T_{\text{非流水}} = nm\Delta t_0, T_{\text{流水}} = m\Delta t_0 + (n-1)\Delta t_0$$

$$S = \frac{mn}{m+n-1} = \frac{m}{1+\frac{m-1}{n}}$$

3.效率

- ◆ 效率指流水线的设备利用率。
- ◆ 由于流水线有[通过时间和排空时间](#)，所以流水线的各段并非一直满负荷工作， $E < 1$
- ◆ 若各段时间相等，则各段效率也相等，即 $e_1 = e_2 = e_3 = \dots = n\Delta t_0 / T_{\text{流水}}$ ([解释](#))
- ◆ 整个流水线效率

$$E = \frac{n\Delta t_0}{T_{\text{流水}}} = \frac{n}{m+n-1} = \frac{1}{1+\frac{m-1}{n}}$$

当 $n \gg m$ 时， $E \approx 1$

3.效率

- ◆ 从时-空图上看，效率就是 n 个任务所占的时空区与 m 个段总的时空区之比
- ◆ 根据这个定义，可以计算流水线各段时间不等时的流水线效率

$$E = \frac{n \text{ 个任务占用的时空区}}{m \text{ 个段总的时空区}}$$

4. 吞吐率、加速比和效率的关系

- ◆ $E = n\Delta t_0 / T_{\text{流水}} = mn\Delta t_0 / (T_{\text{流水}}m) = S/m$

效率是实际加速比 S 与最大加速比 m 之比。

- ◆ $E = n\Delta t_0 / T_{\text{流水}} = (n/T_{\text{流水}}) \cdot \Delta t_0 = TP\Delta t_0$

当 Δt_0 不变时，流水线的效率与吞吐率呈正比。
为提高效率而采取的措施，也有助于提高吞吐率。

5. 流水线性能分析实例

例1 在静态流水线上计算 $\sum_{i=1}^m A_i B_i$ ，问吞吐率、加速比、效率各是多少？

解： (1) 确定适合流水处理的计算过程
(2) 画时-空图
(3) 性能计算

$$\text{吞吐率 } TP = 7/20 \cdot \Delta t_0$$

$$\text{加速比 } S = 34 \cdot \Delta t_0 / 20 \cdot \Delta t_0 = 1.7$$

$$\text{效率 } E = (4 \times 4 + 3 \times 6) / (8 \times 20) = 0.21$$

上述方案性能不高！

- 静态多功能流水线在对某种功能进行处理时，总有某些段处于空闲状态
- 功能切换增加了前一种功能的排空时间和后一种功能的通过时间
- 需要把输出回传到输入（冲突）

能否通过动态流水线改进其性能？

- 举例I
- 举例II 这样行不行？ 正确方案

例2 在MIPS的非流水实现和基本流水线中，5个功能单元的执行时间：10/8/10/10/7ns。流水线额外开销为1ns，求相对于非流水指令实现而言，基本MIPS流水线的加速比是多少？

解： $T_{\text{非流水}} = 10 + 8 + 10 + 10 + 7 = 45$

$T_{\text{流水}} = 10 + 1 = 11$

加速比 $S = 45/11 \approx 4.1$

注： 流水线额外开销包括：流水寄存器的延迟（建立时间和传输延迟）以及时钟扭曲

6. 有关流水线性能的若干问题

- ◆ 流水线并不能减少（而且一般是增加）单条指令的执行时间，但**能够提高吞吐率**
- ◆ 增加流水线的深度可以**提高流水线性能**
- ◆ 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寄存器
 - Earle锁存器
- ◆ 指令之间存在的相关，产生了流水线的**冲突**，进而限制了流水线的性能

第6章 流水线技术

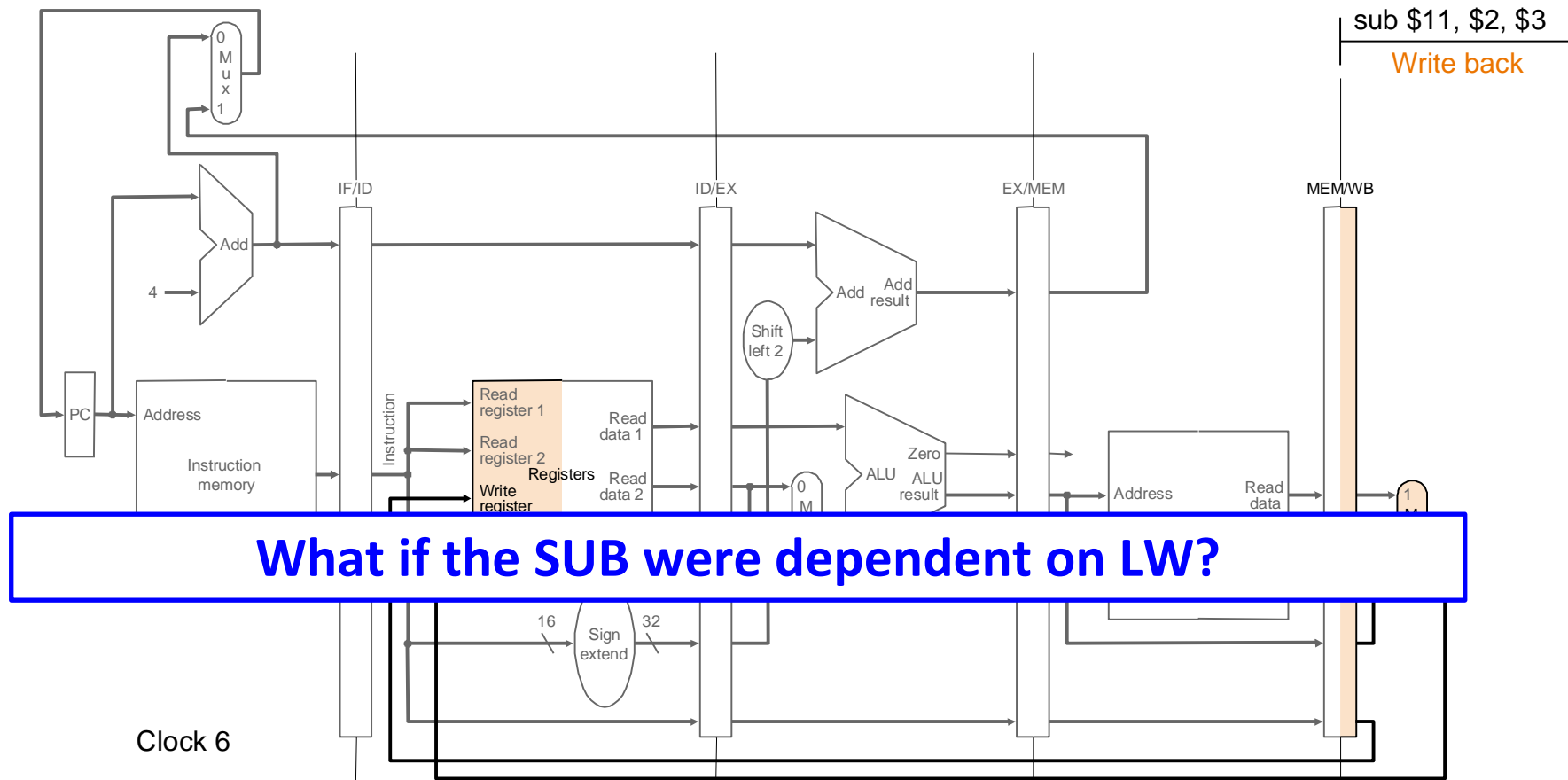
6.1 流水线概述

6.2 MIPS的基本流水线

6.3 流水线中的冲突

6.4 实例分析：MIPS R4000

Pipelined Operation Example



6.3 流水线中的冲突

6.3.1 流水线的结构冲突

6.3.2 流水线的数据冲突

6.3.3 流水线的控制冲突

6.3 流水线中的冲突

1. 什么是流水线冲突 (Pipeline Hazard) ?

流水线冲突是指相邻或相近的两条指令因存在某种关联，后一条指令不能在原先指定的时钟周期开始执行。

◆ 消除冲突的基本方法——**暂停**

暂停流水线中某条指令及其后面所有指令的执行，该指令之前的所有指令继续执行。

6.3 流水线中的冲突

2. 三种不同类型的冲突

- ◆ **结构冲突 (Structural Hazard)** : 当指令在重叠执行过程中, 硬件资源满足不了指令重叠执行的要求而发生的冲突。
- ◆ **数据冲突 (Data Hazard)** : 因一条指令需要用到前面指令的结果, 而无法与产生结果的指令重叠执行时发生的冲突。
- ◆ **控制冲突 (Control Hazard)** : 当流水线遇到分支指令和其它会改变PC值的指令所引起的冲突。

6.3.1 流水线的结构冲突

1. 导致结构冲突的常见原因：

- 功能部件不是全流水
- 重复设置的资源数量不足

2. 实例：当数据和指令存在同一存储器中时，访存指令会引起存储器访问冲突。

解决方法：

- I. 插入暂停周期（时空图）
- II. **将指令存储器和数据存储器分离**

6.3.1 流水线的结构冲突

3. 避免结构冲突的方法：

- 所有功能单元完全流水化
- 设置足够多的硬件资源

但是，硬件代价很大！

4. 有些设计方案允许结构冲突存在

- 降低成本
- 减少功能单元的延迟

例： 当前许多机器都没有将浮点功能单元完全流水，
比如在MIPS实现中，浮点乘需要5个时钟周期，
但对该指令不流水。请分析由此引起的结构冲突
对mdljdp2基准程序在MIPS上运行的性能有何影响？
为简单起见，假设浮点乘法服从均匀分布。

解： mdljdp2中浮点乘法出现的频率约为14%。

最坏情况： 每个浮点乘都无法与其它操作重叠执行，
都需要5个周期，此时CPI为1.56

最好情况： 可以完全重叠执行，仅需要1个周期，
此时没有性能损失

6.3.2 流水线的数据冲突

1. 数据冲突简介

实例: ADD R1, R2, R3
 SUB R4, R1, R5
 AND R6, R1, R7
 OR R8, R1, R9
 XOR R10, R1, R11

产生原因：当指令在流水线中重叠执行时，流水线有可能改变指令读/写操作数的顺序，使之不同于它们在非流水实现时的顺序，这将导致数据冲突。

消除方法：向流水线中插入暂停周期

6.3.2 流水线的数据冲突

2. 通过定向技术减少数据冲突带来的暂停

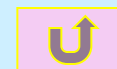
定向(forwarding)，也称为旁路(bypassing)

工作过程

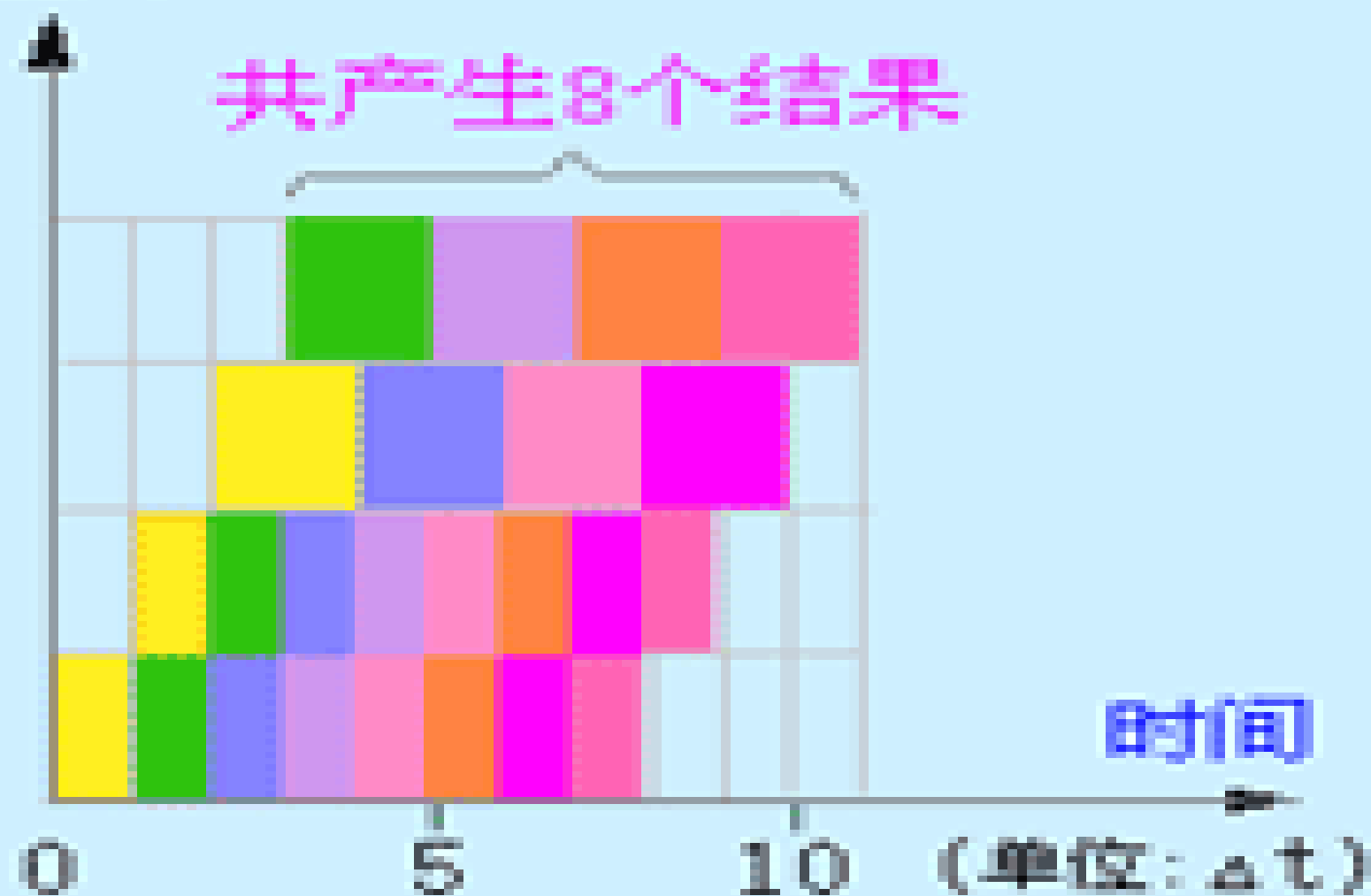
主要思路：将计算结果从其产生的地方直接送到真正需要它的地方，就可以避免暂停。

- 寄存器文件EX/MEM中的ALU运算结果总是回送到ALU的输入寄存器
- 从定向通路得到输入数据的ALU操作不必从源寄存器中读取操作数

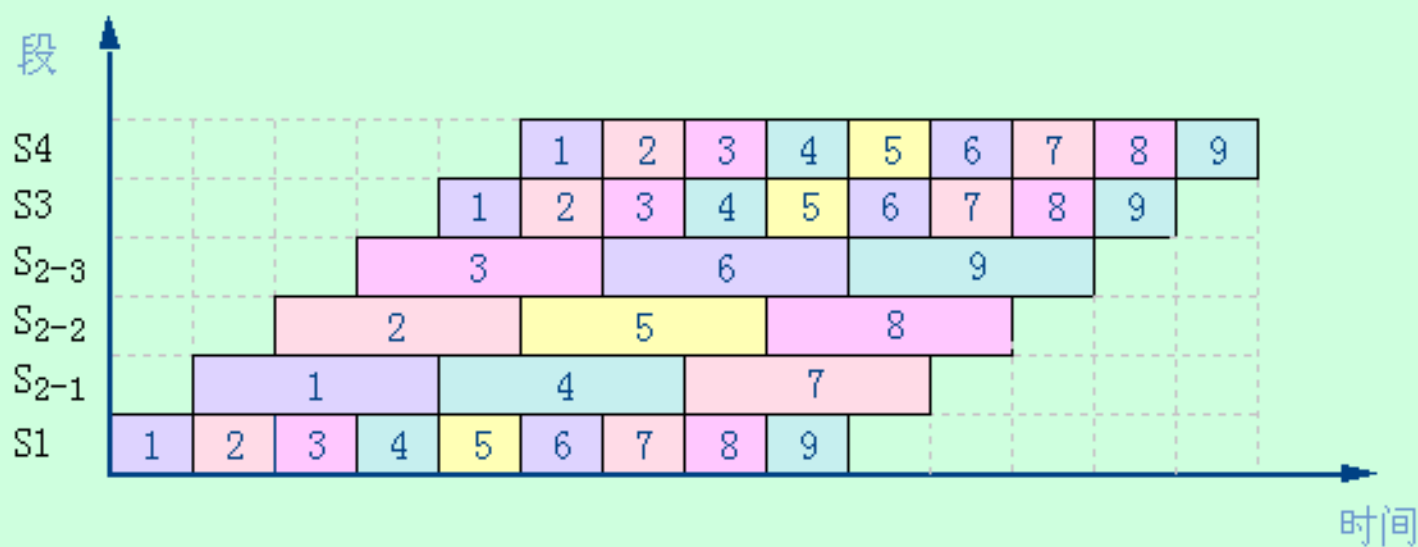
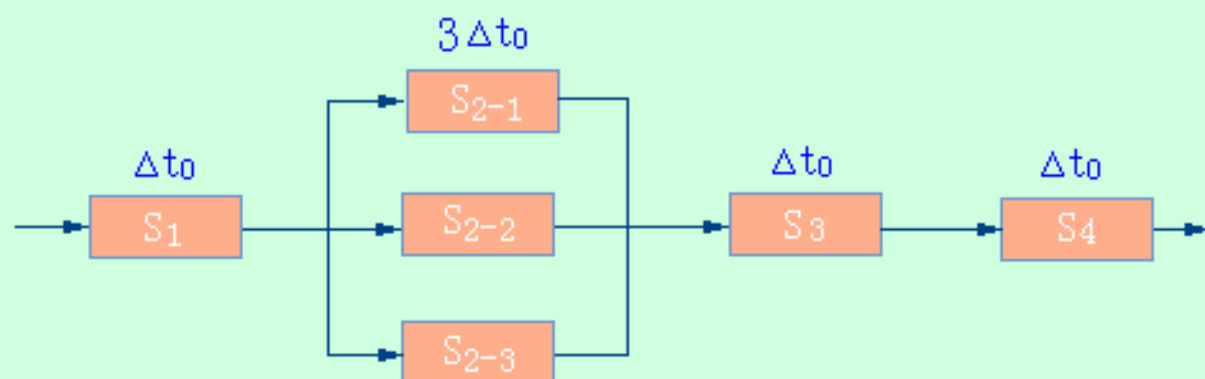
空间



共产生8个结果

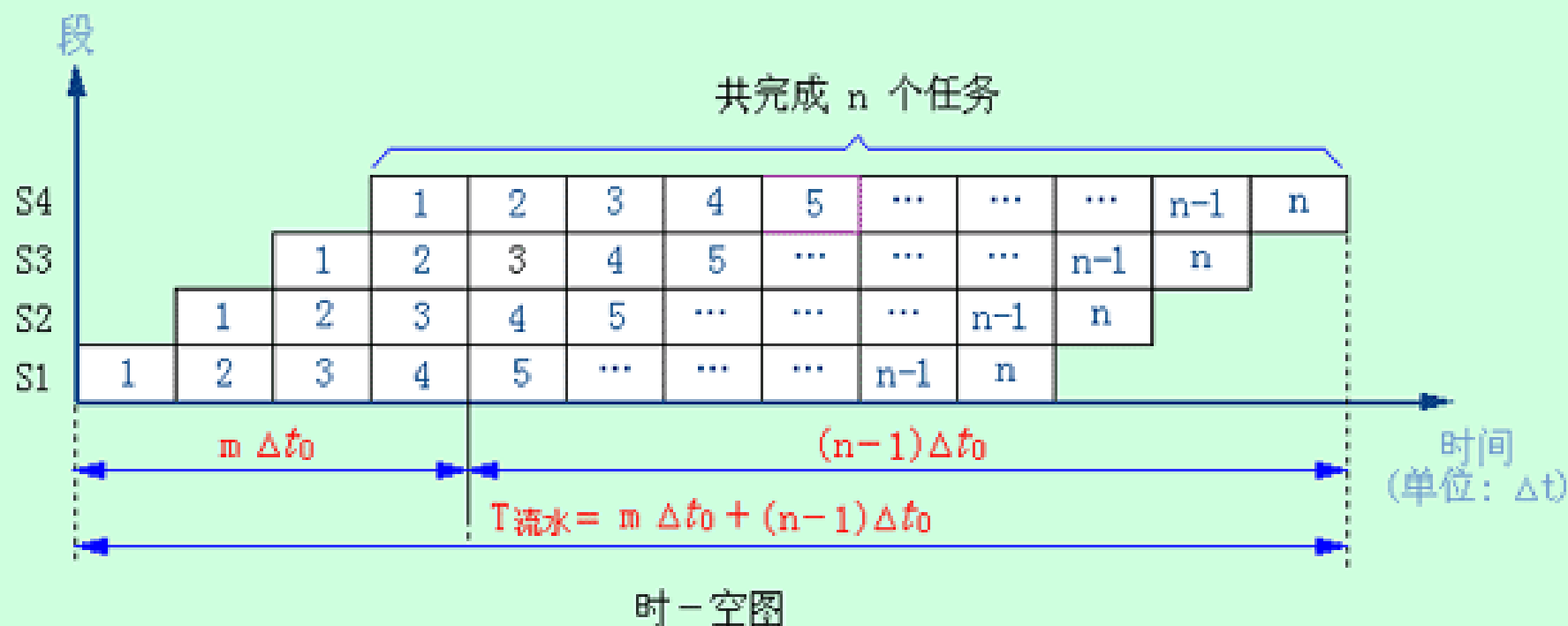
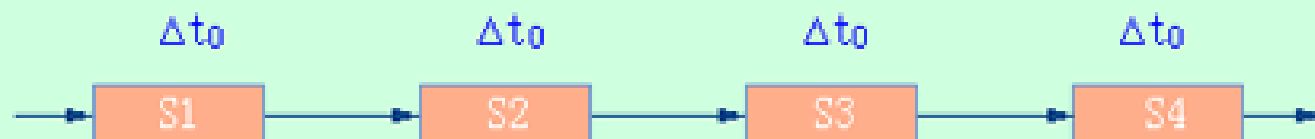


时空图



流水线的时-空图

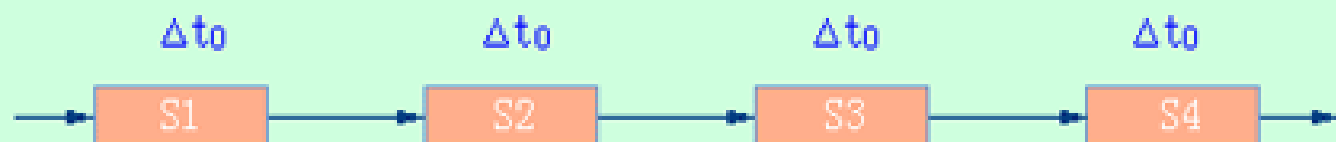
(各段时间相等)



$$T_{\text{流水}} = m \Delta t_0 + (n-1) \Delta t_0$$

流水线的时-空图

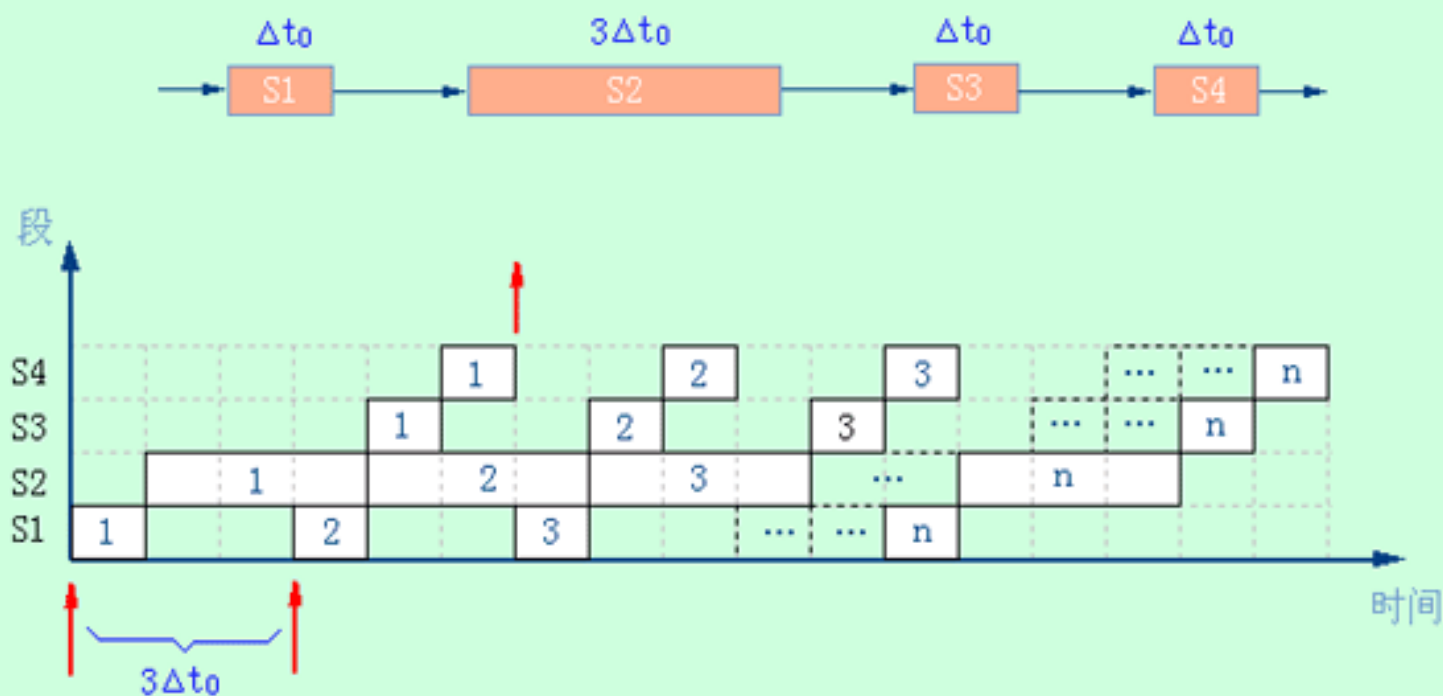
(各段时间相等)



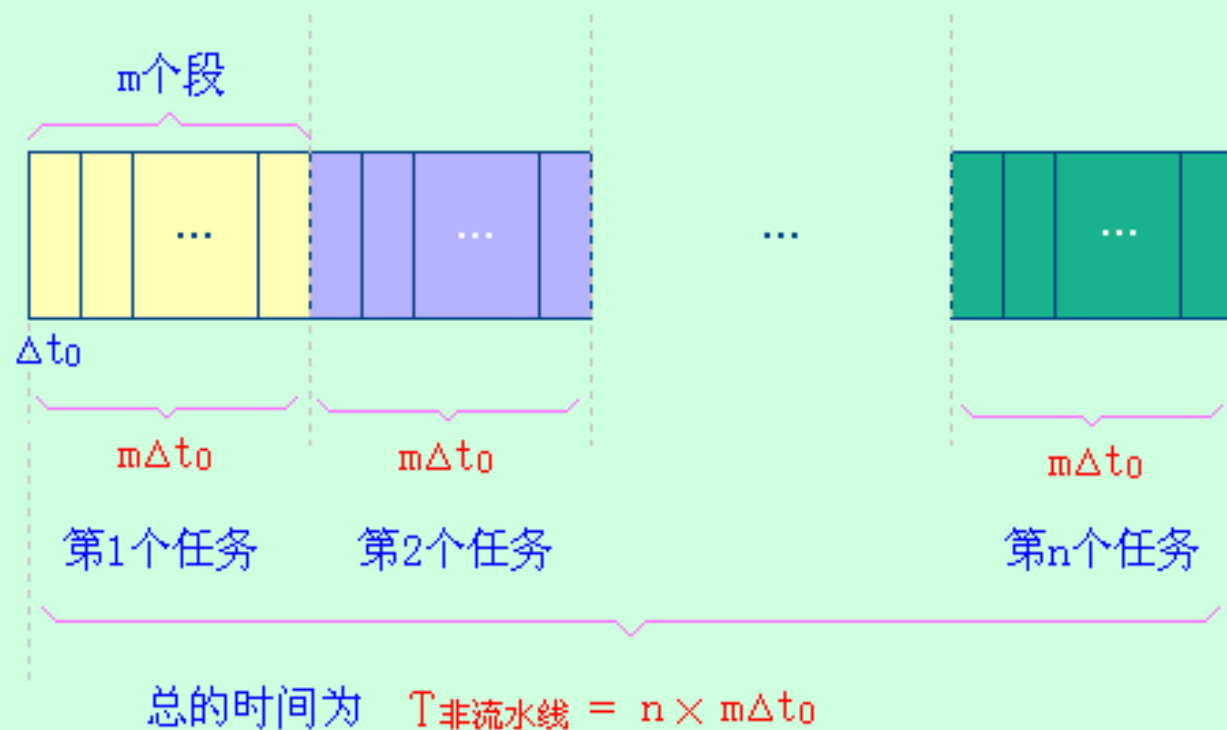
$$\text{吞吐率 } TP = \frac{n}{T_{\text{流水}}}$$

流水线的时-空图

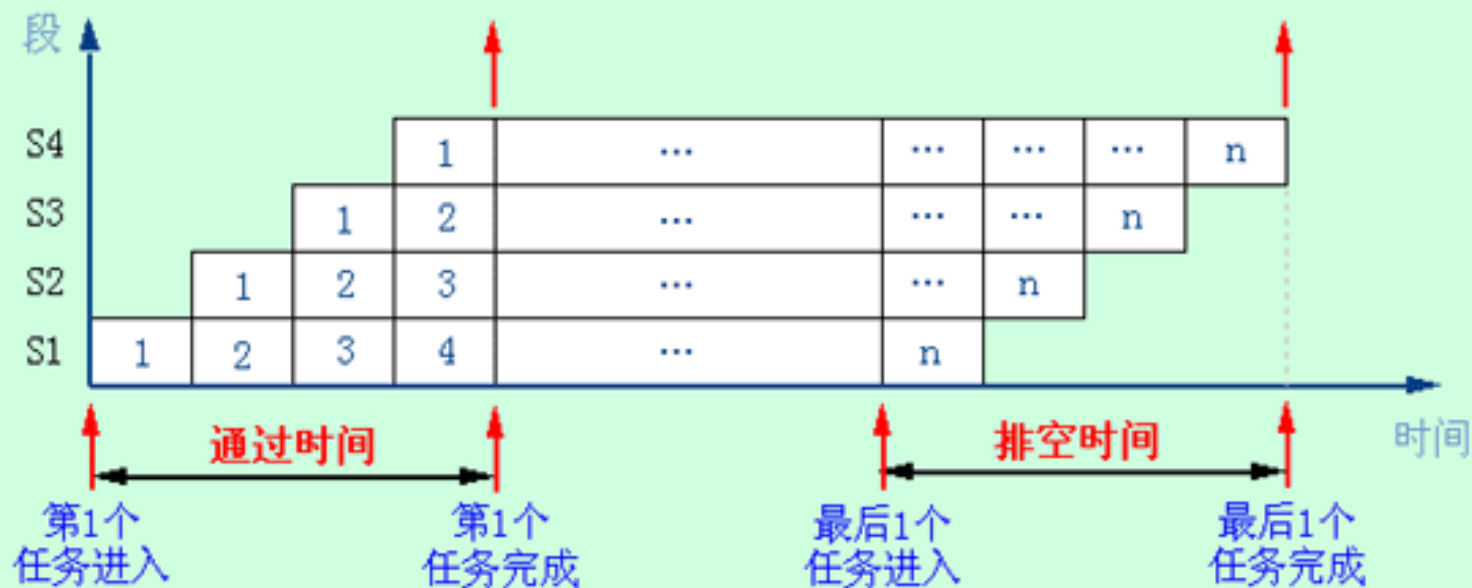
(各段时间不等)



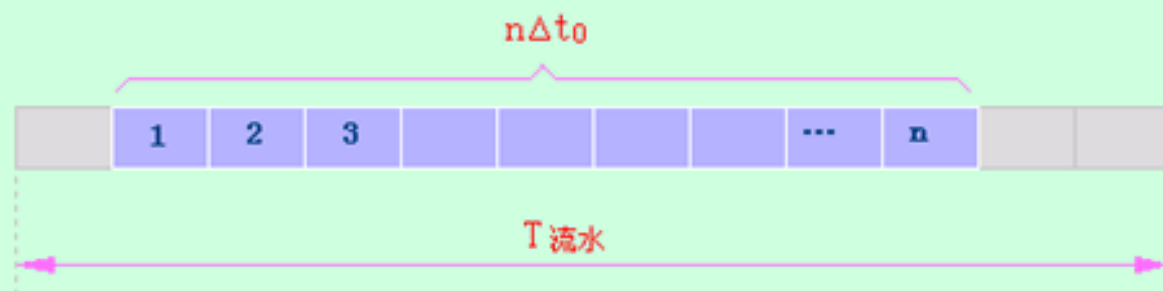
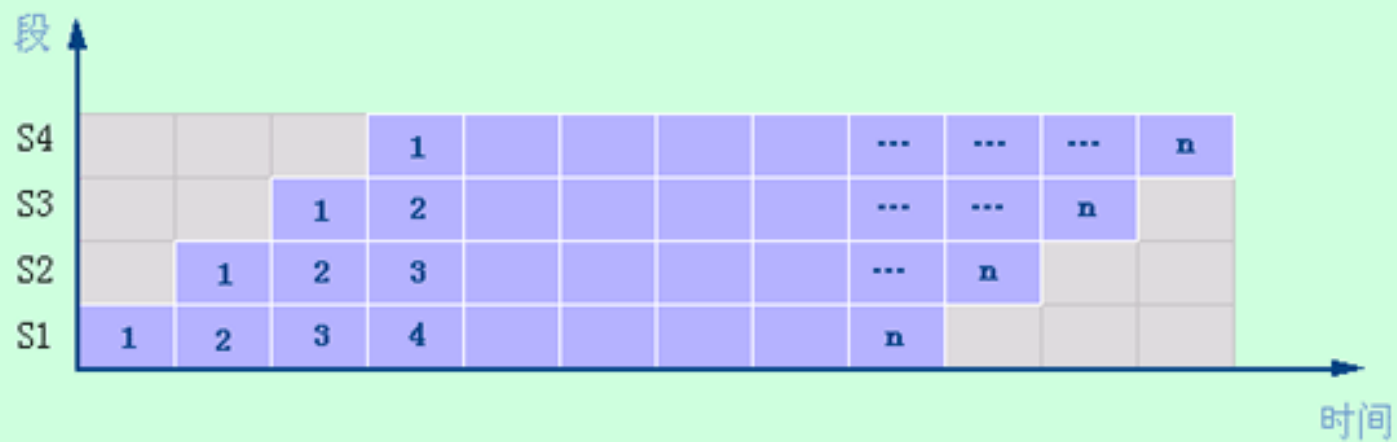
非流水方式所需的时间



通过时间和排空时间



流水段的效率

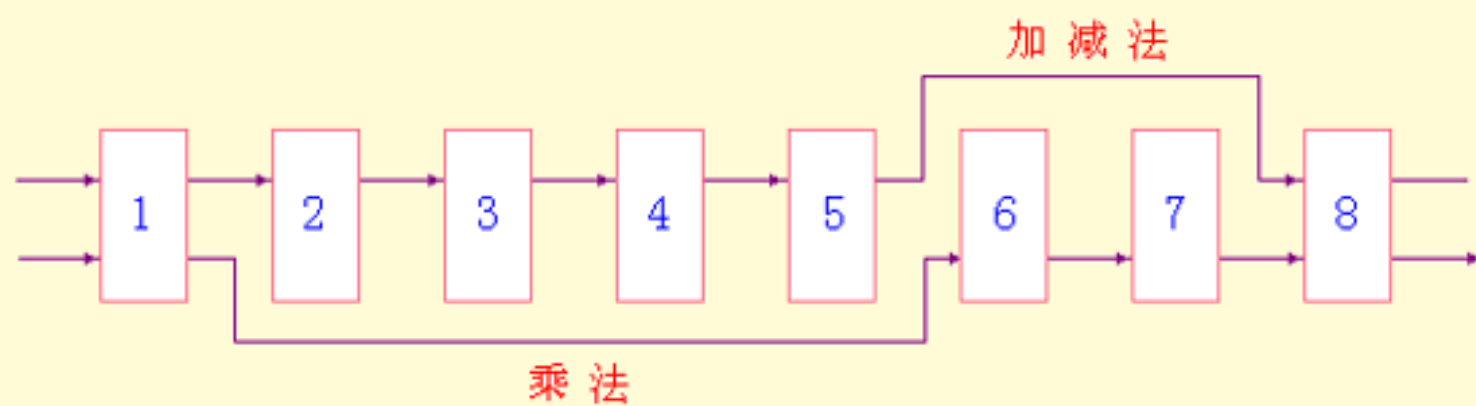


所以

$$e_2 = \frac{n\Delta t_0}{T_{流水}}$$

静态流水线

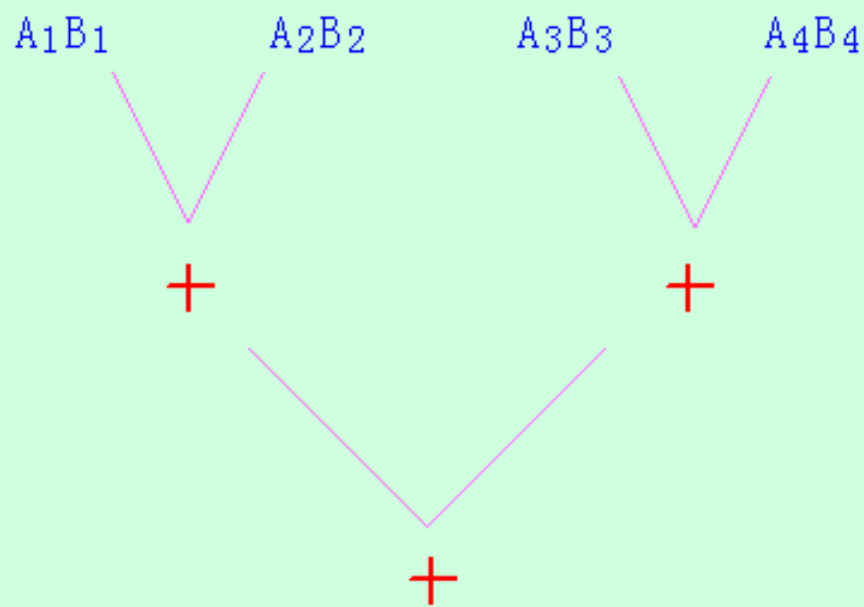
(举例)



$\sum_{i=1}^4 A_i B_i$ 的计算过程

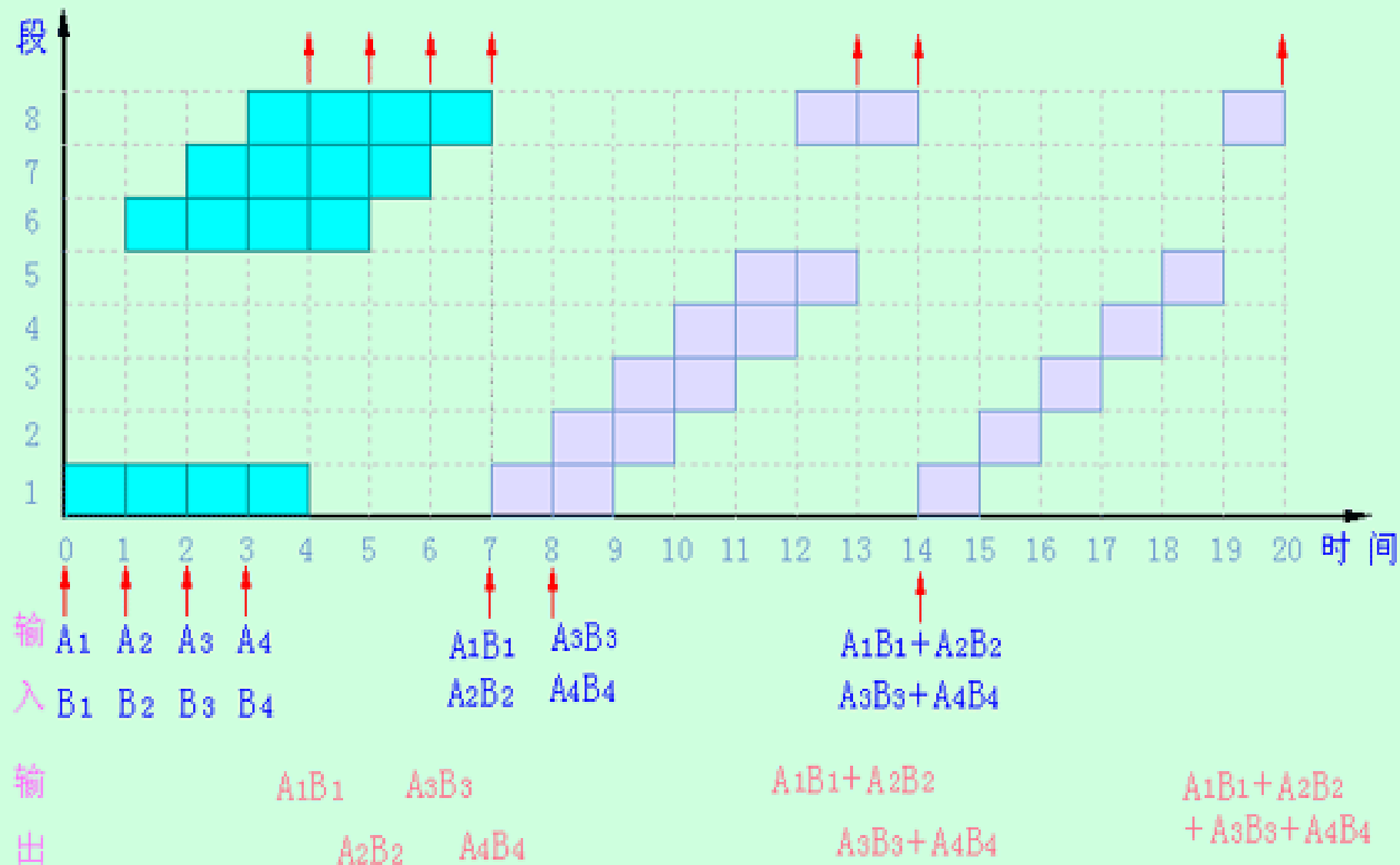
(例 3.1)

$$\sum_{i=1}^4 A_i B_i = A_1 B_1 + A_2 B_2 + A_3 B_3 + A_4 B_4$$



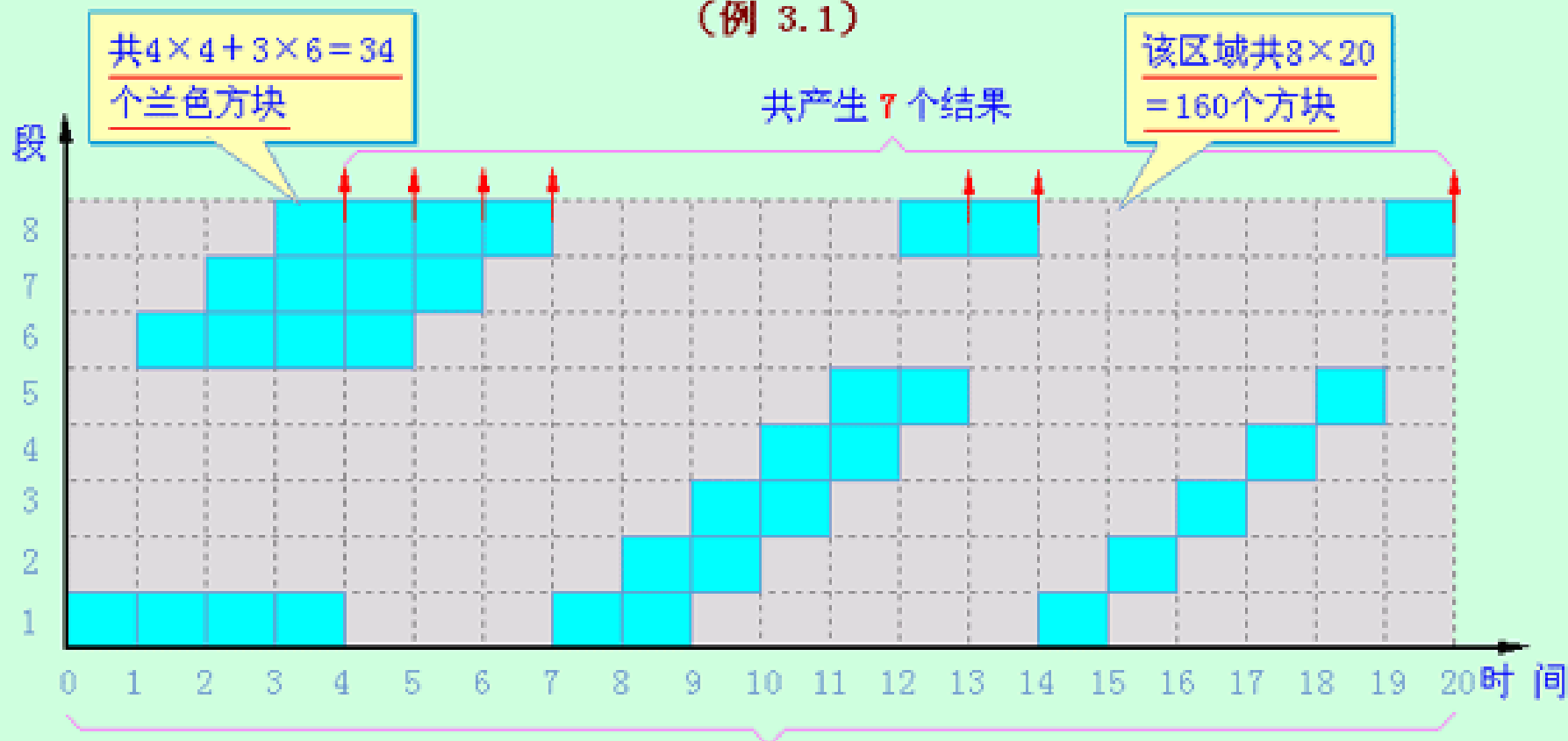
时空图

(例 3.1)



性能计算

(例 3.1)



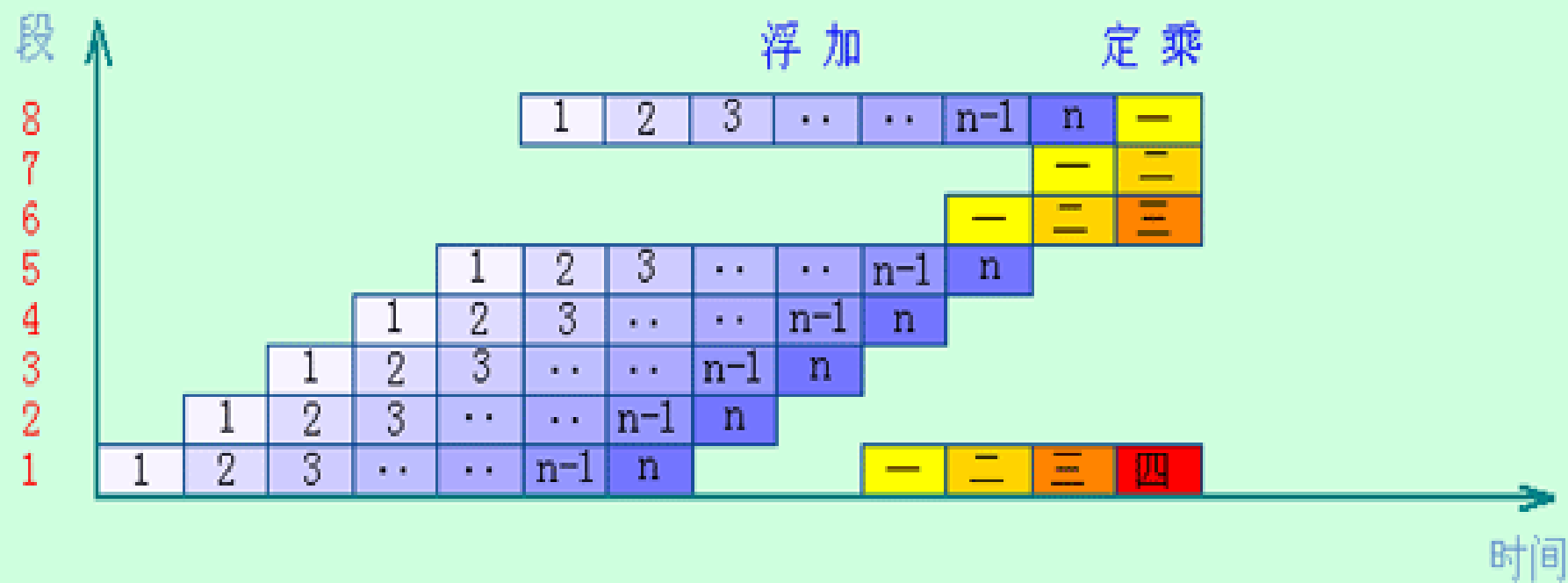
处理时间 = $20 \Delta t$

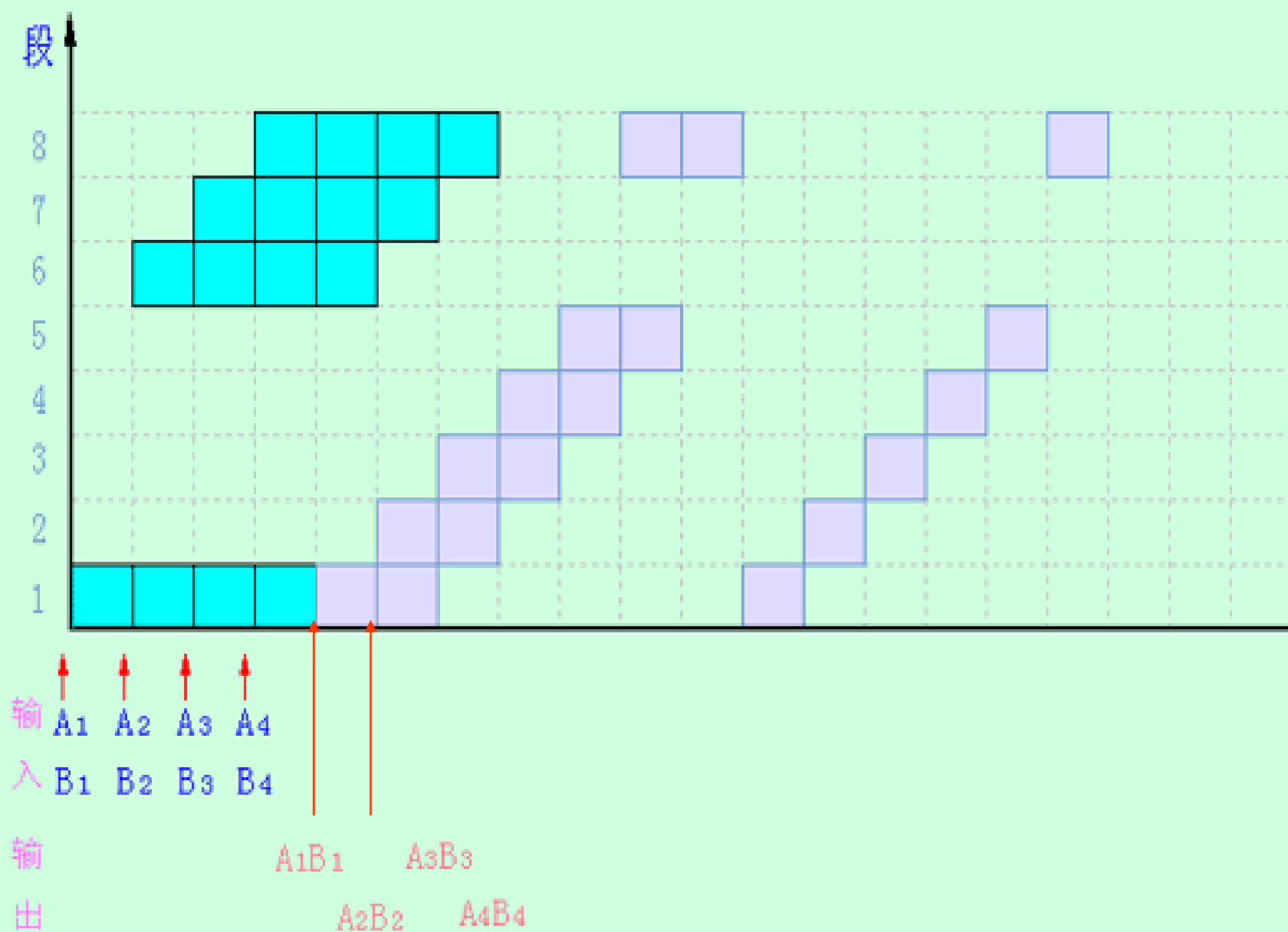
吞吐率 $TP = \frac{7}{20 \Delta t}$

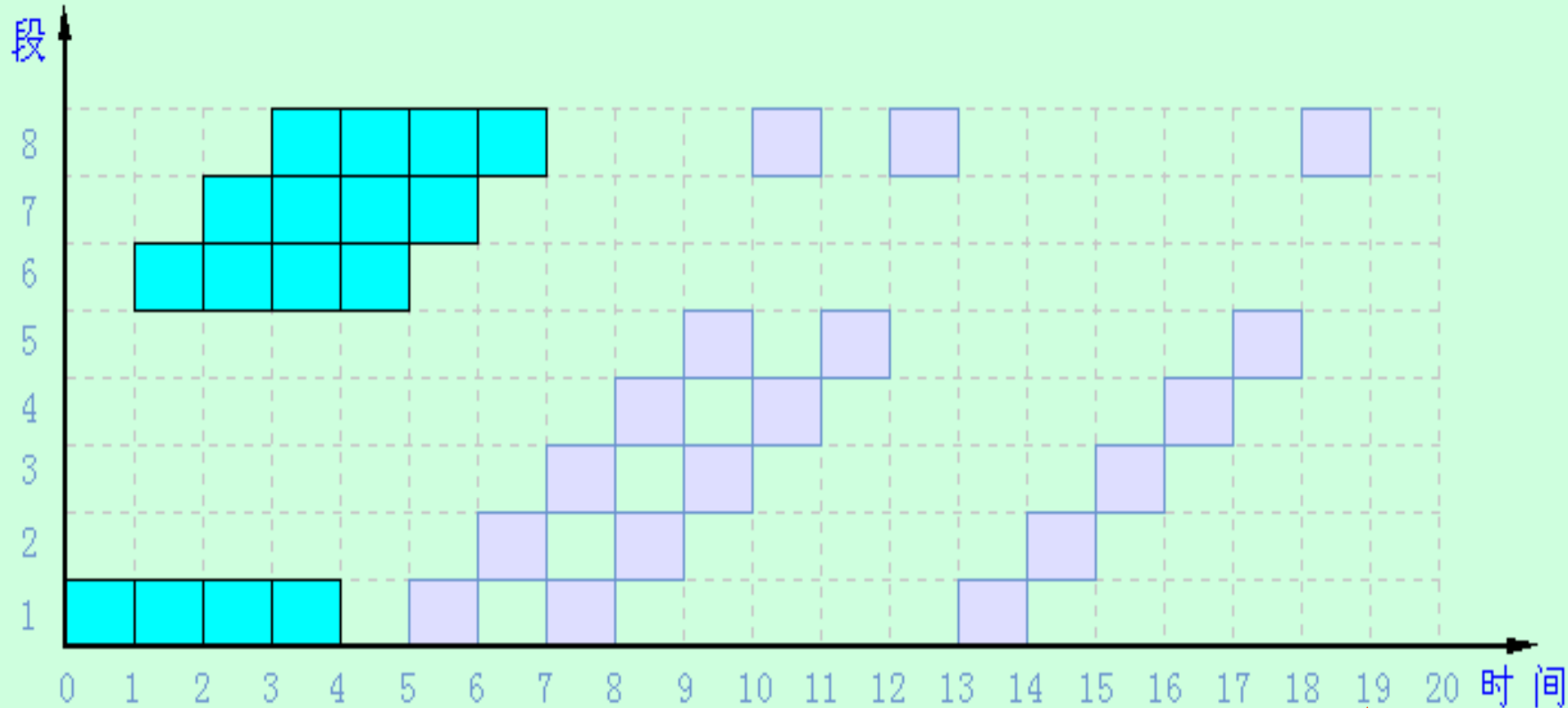
非流水处理所需时间 = $4 \times 4 \Delta t + 3 \times 6 \Delta t = 34 \Delta t$

加速比 $S = \frac{34 \Delta t}{20 \Delta t} = 1.7$

效率 $E = \frac{34}{160} = 0.21$







输入

A_1 A_2 A_3 A_4

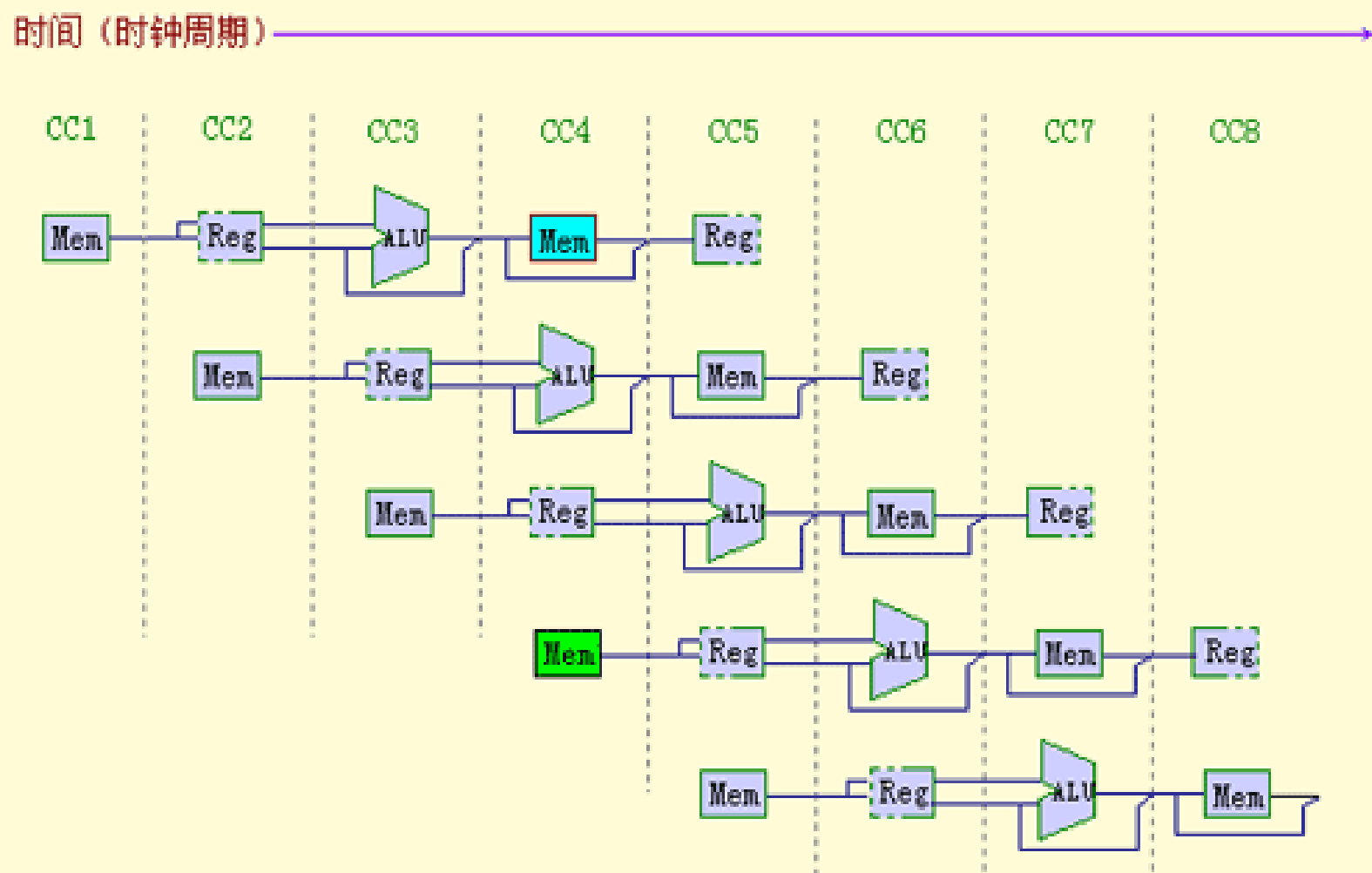
B_1 B_2 B_3 B_4

输出

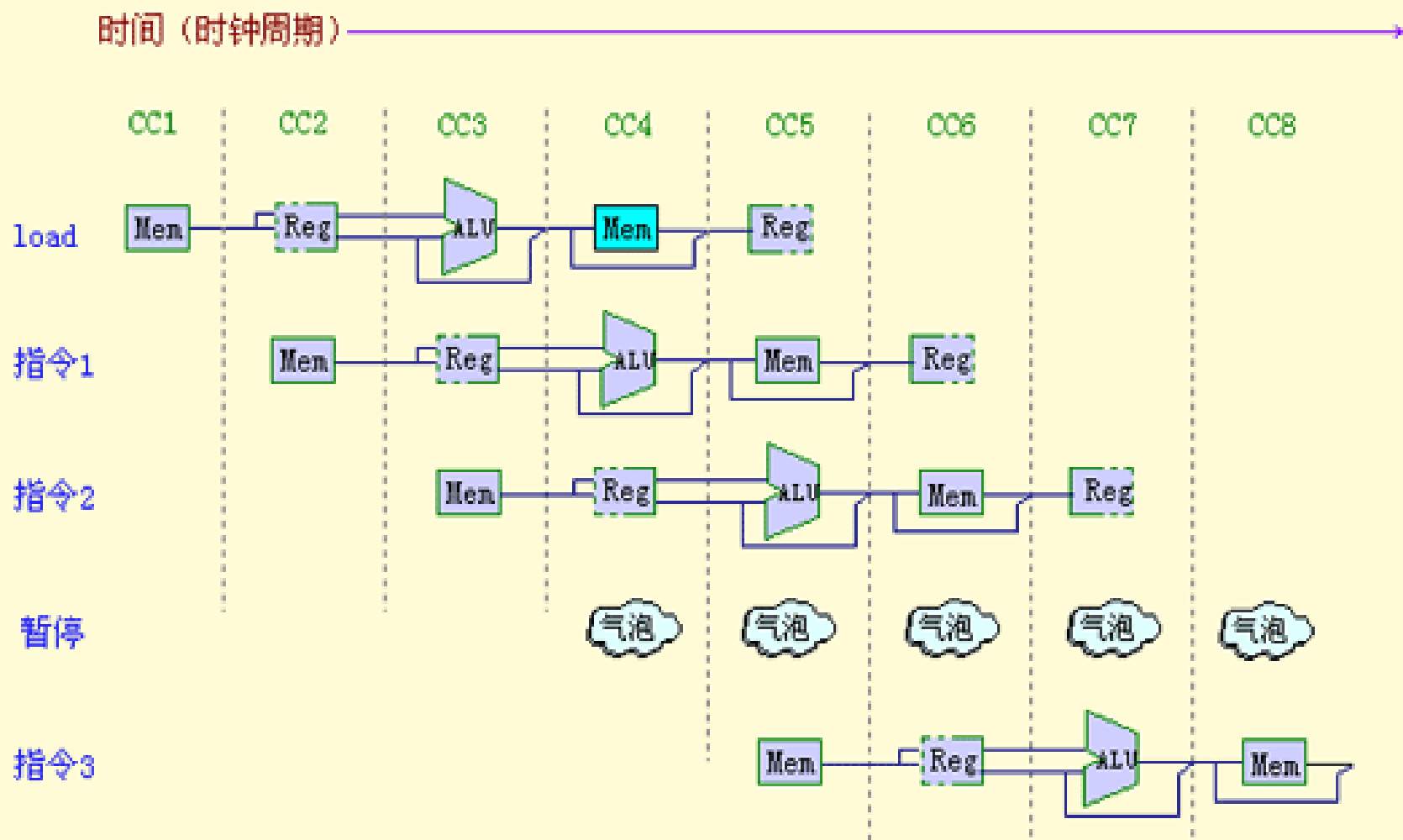
A_1B_1 A_3B_3 $A_1B_1 + A_2B_2$ $A_1B_1 + A_2B_2 + A_3B_3 + A_4B_4$

A_2B_2 A_4B_4 $A_3B_3 + A_4B_4$

结构相关举例——访存冲突



流水线气泡

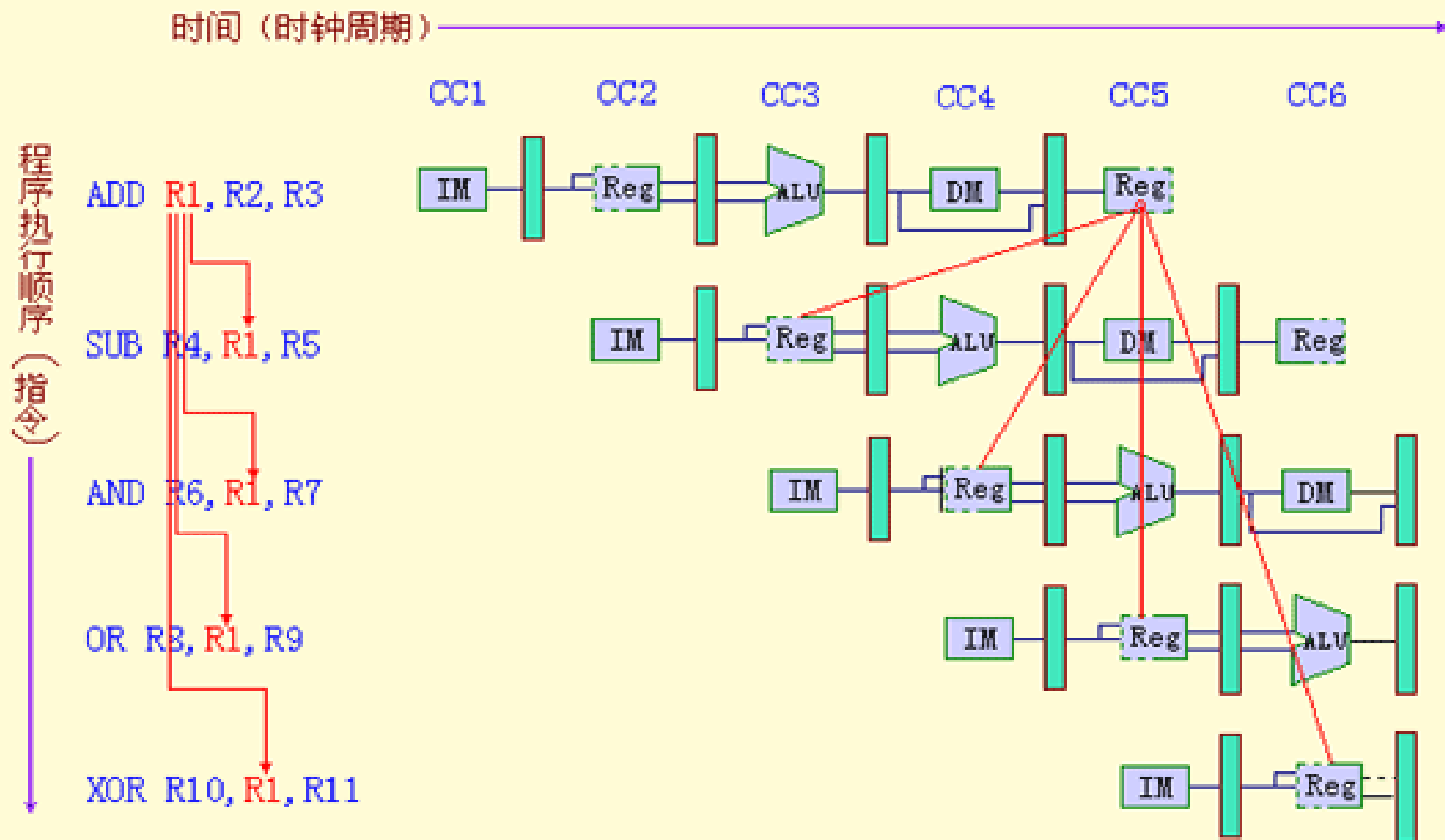


引入暂停后的流水线时空图

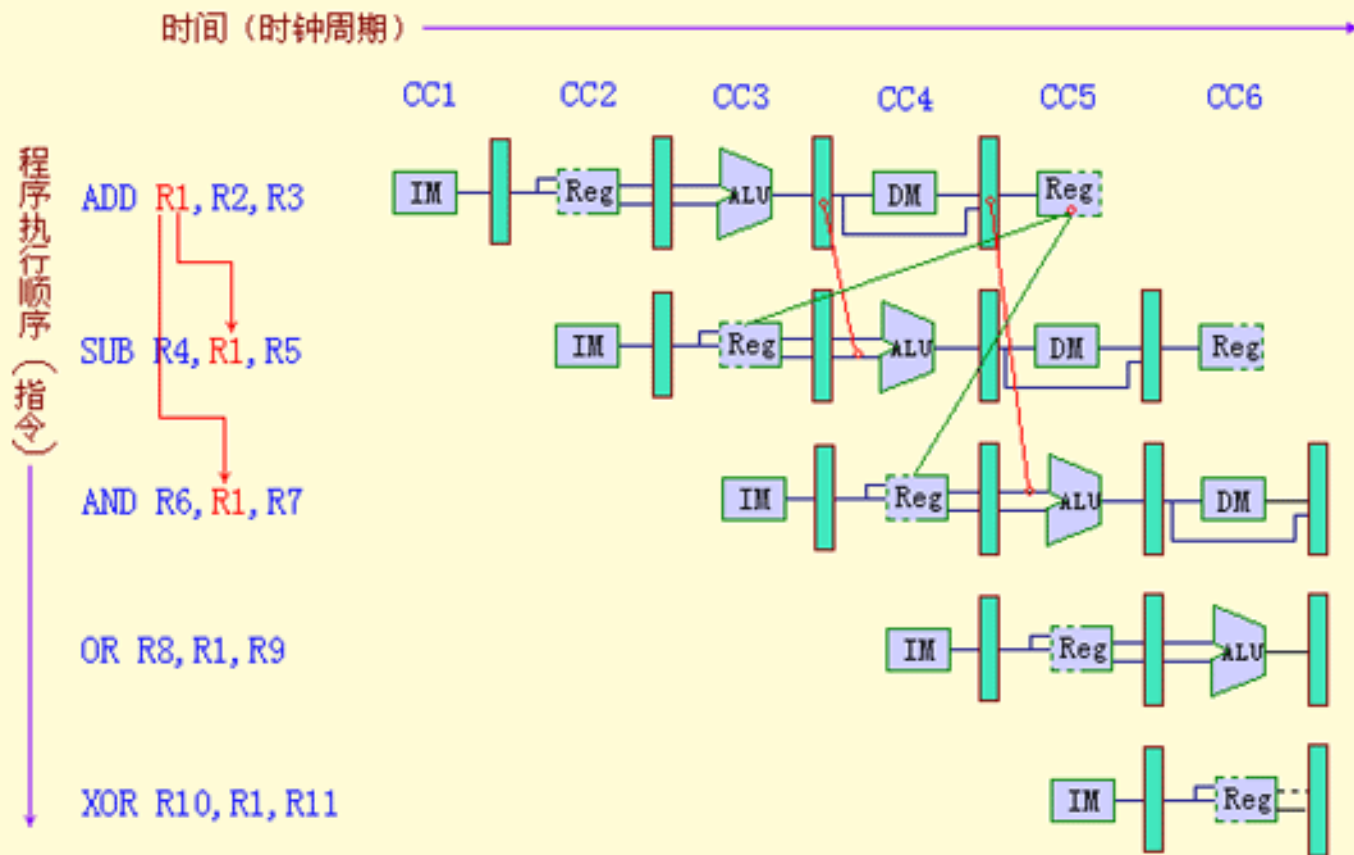
| | 时钟周期 | | | | | | | | |
|-------|------|----|----|-------|-----|-----|----|-----|-----|
| 指令编号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 指令i | IF | ID | EX | MEM | WB | | | | |
| 指令i+1 | | IF | ID | EX | MEM | WB | | | |
| 指令i+2 | | | IF | ID | EX | MEM | WB | | |
| 指令i+3 | | | | stall | IF | ID | EX | MEM | WB |
| 指令i+4 | | | | | | IF | ID | EX | MEM |

WB

数据相关举例



采用定向技术消除数据相关



采用定向技术后的工作过程

- 指令1 ADD R1, R2, R3
- 指令2 SUB R4, R1, R5
- 指令3 AND R6, R1, R7
- 指令4 OR R8, R1, R9
- 指令5 XOR R10, R1, R11

